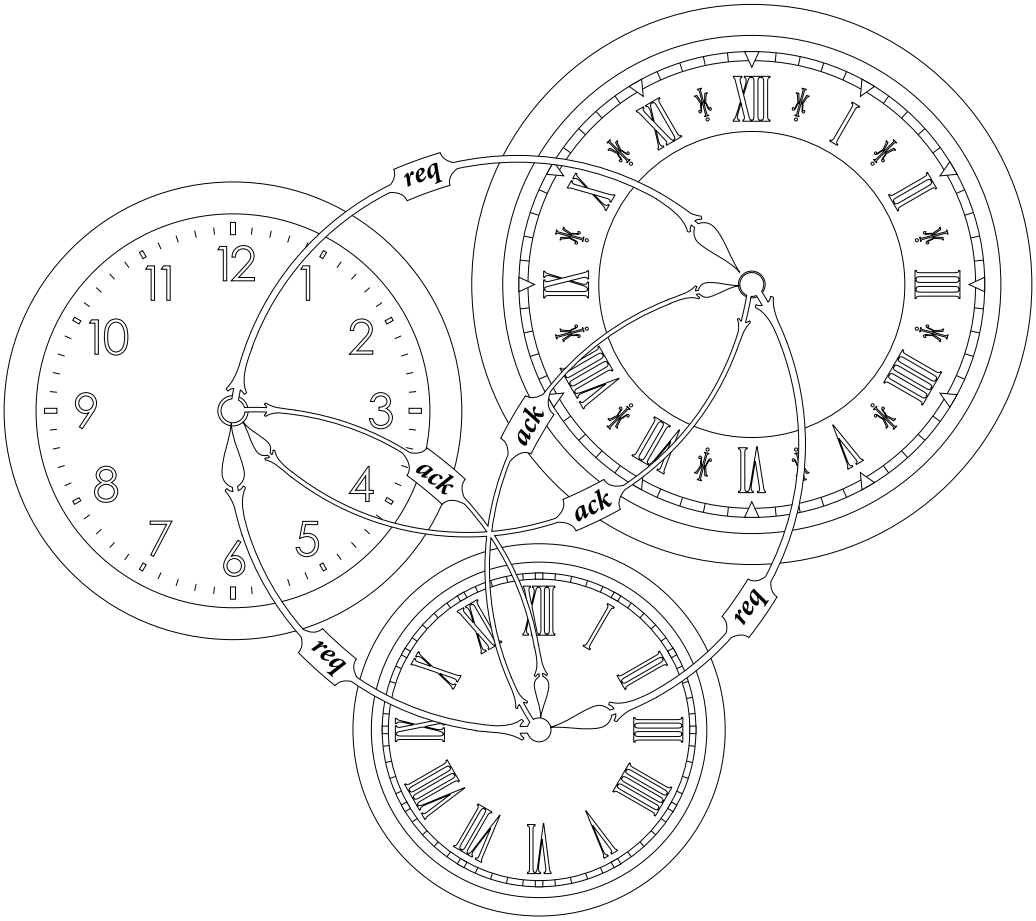
This Asynchronous World

Essays dedicated to Alex Yakovlev on the occasion of his 60th birthday



Editor:

Andrey Mokhov
Newcastle University
School of Electrical and Electronic Engineering
Newcastle upon Tyne, NE1 7RU, UK
Email: andrey.mokhov@ncl.ac.uk

Cover art by:

Ashur Rafiev
Newcastle University
School of Computing Science
Newcastle upon Tyne, NE1 7RU, UK
Email: ashur.rafiev@ncl.ac.uk

Book production:

Ghaith Tarawneh
Newcastle University
Institute of Neuroscience
Newcastle upon Tyne, NE1 7RU, UK
Email: ghaith.tarawneh@ncl.ac.uk

Publisher:

Newcastle University
Newcastle upon Tyne, NE1 7RU, UK
Website: www.ncl.ac.uk

ISBN: 978-0-7017-0257-1

Publication date: 20 July 2016

Preface

The world was in a very unstable state before this Festschrift. It waited, impatiently, for a lightest touch to explode with thousands of messages flown across the globe. The messages have turned into stories, and the stories have turned into this book.

As the reader is undoubtedly aware, this book is dedicated to Alex Yakovlev on the occasion of his 60^{th} birthday, but let us not confuse the occasion with the underlying cause, which is: *everybody loves Alex!* His open mind, infectious enthusiasm, and positive thinking are driving the community, and we – his students, colleagues and friends – thank him here, in writing.

It is important to note that not everyone was able to complete their writing in time to be included into the first edition of the book. But fear not: as a self-appointed editor I declare this book fully asynchronous and unbounded. Let us celebrate the variability of the contributions with respect to their completion time (which is likely to follow Alex's favourite long-tail distribution), as well as their topic and length, and welcome new contributors, who will become part of future editions of the book[1].

One of the many lessons I learned from Alex is that 'when' is just as important as 'what' and 'why'. We present our stories to Alex on his 60^{th} anniversary; at best, I can only half understand the magnitude of this number in my 30s, therefore let me delegate the task of arguing that "60 is not just a number" to a philosopher in our community:

> *Professor Alex Yakovlev.*
>
> *The days of our years are threescore years and ten. Respect our forefathers, as you become one. Marvel at their wise words and institutions; the giant shoulders to our improbable but superlative technical edifice. The next 40yrs will be as exciting as the last, but you will not lead its charge. Time now to set aside personal aspiration and firm the shoulder for those who follow. As life takes you to this new and scary place, remember you are not alone in the dark.*
>
> *Though ships that pass, I am very pleased to have known, shared and laughed with you o'er years beyond number. And equally pleased to count you friend and colleague.*
>
> *60 is not just a number; it's the end of childhood.*
>
> *Ian Phillips (ARM)*

I would like to thank everyone who contributed to this book, sent their words of encouragement and support, and helped with the editing. And with no further ado, ladies and gentlemen, I invite you to join Alex and turn this page.

Andrey Mokhov
July 2016, Newcastle

[1] Please contact me by email andrey.mokhov@ncl.ac.uk if you would like to contribute a paper or for any other queries.

Table of Contents

Resilient Bundled-Data Design: Motivation, Results to Date, and Remaining Challenges*

Peter A. Beerel[1] and Ney L. V. Calazans[2]

[1] University of Southern California - Los Angeles, California, USA,
pabeerel@usc.edu
[2] Pontifícia Universidade Católica do Rio Grande do Sul - Porto Alegre, Brazil,
ney.calazans@pucrs.br

Abstract. The periodic nature of the global clock in traditional synchronous designs forces circuits to be margined for the worst possible case of process, voltage, temperature, and data conditions. This constrains the silicon to operate at worst-case frequencies and at conservative supply voltages. Resilient architectures promise to remove these margins, by detecting and correcting timing errors when they occur, thereby creating the potential to achieve real average-case operation. However, synchronous resilient schemes previously proposed can suffer from multiple issues, including being susceptible to metastability and requiring often complex changes to the architecture to support replay-based recovery from timing errors. These problems respectively lead to circuit failures and/or incur high timing penalties when errors occur. This paper reviews a recently proposed resilient bundled-data template called Blade that is robust to metastability issues, requires no replay-based logic, and has low timing error penalties. It also describes some open issues and new research opportunities this template presents, including automation problems to target average-case operation, specific circuit optimizations to minimize resiliency overhead, and the need for new test procedures to tune delay lines and screen out bad chips.

1 Introduction and Related Work

Traditional synchronous designs must incorporate timing guardbands to ensure correct operation under worst-case delays caused by process, voltage, and temperature (PVT) variations as well as data-dependency [6]. This is particularly important in low-power low-voltage designs, as performance uncertainty due to PVT variations grows from around 50% at nominal supply to around 2,000% in the near-threshold domain [14]. To address this problem, many synchronous design techniques for resilient circuits have been proposed that address delay variations. For example, canary FFs predict when the design is close to a setup timing failure (see e.g., [41]). Designs can then adjust their supply voltage or clock frequency either statically or dynamically to ensure correct operation at

* This is an extended and updated version of an ECCTD 2015 invited paper on the same topic.

the edge of failure. The adverse impact of variations on hold margins are significantly more challenging to manage because changing the clock frequency and voltage does not typically resolve hold problems and thus these must be very conservatively managed. Hold constraints are typically resolved by preemptively adding hold buffers to all "short" paths in the design. Unfortunately, at low voltages, the number of hold buffers needed can be much larger than at nominal voltages, because the increased delay variation causes: (1) clock uncertainty to grow; (2) a larger fraction of paths to be identified as potentially short, due to the possible decrease in delays resulting from variability; and (3) the hold buffers themselves possibly being unexpectedly fast. All these margins translate into considerable increases in energy consumption.

Several research groups have explored adding a degree of *timing resiliency* into the design to detect and then recover from setup violations [9, 12, 25, 27]. There are two general approaches: architecturally dependent, or "replay-based" approaches, and architecturally independent. The former includes Razor II [12] and the Intel approach described in [6]. The problem with these approaches is that they work much like pulsed latch circuits: the wider the pulse, the more resiliency is obtained, at the cost of worsening hold time margins [15]. Moreover they require synchronizers in the control path, incurring long delays to identify whether an error occurred, and demand complex replay and recovery mechanisms [6, 12, 27]. Granted, the area overhead of these can be amortized by reusing existing recovery logic (e.g., for resuming after a mispredicted branch), but the techniques remain architecturally invasive and thus a design challenge. In contrast, architecturally independent approaches like Bubble Razor [15] and TIMBER [9] require no architectural changes and can be automatically generated from standard RTL specifications. The flow involves replacing flip-flops with retimed latches that have non-overlapping clocks, mitigating hold time problems. Bubble Razor, for example, avoids replay and recovery by immediately stalling neighboring stages via clock gating, and solves timing errors on the fly and locally. However, the template assumes that metastability can be resolved within one clock period, which is often unrealistic and leads to poor mean-time-between-failures rates [3]. More recent work [25] proposes to borrow time only from the following stage by quickly boosting its supply voltage to accommodate for the borrowed time. Unfortunately, this approach requires fast error detection and dynamically adjustable supply voltages which limits its applicability.

Different asynchronous templates have also been proposed to address the excessive margining problem (e.g., [42]). Quasi-delay-insensitive (QDI) templates use completion signal logic which makes them robust to delay variations at the cost of increased area (often 4x larger than synchronous counterparts or even more) and high switching activity due to a return to zero paradigm (e.g., [16, 38]). Bundled-data templates (e.g., micropipelines [39]) use delay lines matched to single-rail combinational logic, providing a low area, low switching activity asynchronous solution (e.g., [10]). However, the delay lines must be implemented with sufficiently large margins in the presence of on-chip variations [11], reducing the advantages of this approach. Researchers have proposed different solutions to mitigate these margins, such as duplicating the bundled-data delay lines [8],

constraining the design to regular structures such as PLAs [24] and using soft latches [28]. Others have suggested current-based completion sensing techniques (e.g., [1, 29]) that rely on analog current sensors, which can be prohibitively power hungry.

This work focuses on a recently proposed asynchronous design template that couples the architectural benefits of resilient techniques with the flexibility of asynchronous bundled-data pipelines. The template, called Blade, minimizes hold time issues, requires no replay-based logic, and is supported by an automatic translation flow from synchronous RTL specifications. It is not only safe from metastability issues but also takes advantage of the low *average* metastability resolution times, which leads to low timing error penalties compared to synchronous alternatives. It thus provides significantly higher potential performance and voltage scaling power benefits.

The paper reviews Blade principles and operation, comparing and contrasting the approach to synchronous alternatives. Its recent application to the design of a MIPS OpenCore processor illustrates techniques to reduce overheads and maximize performance and power benefits. The paper also discusses the range of designs for which this design style is likely to provide the biggest overall benefit, as well as some of the open problems that must be solved to maximize the opportunity to use Blade and make the method commercially attractive.

2 The Blade Bundled-Data Architecture

As Figure 1 shows, pipeline stages in Blade use single-rail logic followed by Transition Detector with Time Borrowing (TDTB) error detecting latches (EDLs) [6,32], Q-Flops [35], and two reconfigurable delay lines. The stage-to-stage delay line is of duration δ and controls when the TDTB goes transparent and begins to propagate data at the output of the combinational logic to the next stage. According to the timing diagram depicted in Figure 2, the asynchronous controller speculatively assumes data at the output of the TDTB latch is stable and triggers the request to the next stage via the standard bundled data request channel consisting of $R.req$ and $R.ack$. The second delay line is of duration Δ and defines a time window during which late transitions that violate this assumption (i.e. timing errors) are allowed, which is called the *timing resiliency window* (TRW). While Δ is elapsing, CLK is high (i.e. the Data Latch is transparent).

Error detecting latches are responsible for triggering an error if a timing violation occurs during the TRW. While there are several EDL implementations (e.g., [6,9,15,32]), Blade employs a custom design [32] based on TDTB latches [6]. The basic design requirement is this component triggers an error on its E output in response to any transition or glitch during the TRW that is significant enough to also propagate to its data output [32]. In this way, no timing violation is missed.

In addition to the push data channel L, Blade uses a second pull *error channel* formed by signals $RE.req$ and $RE.ack$ to manage potential timing violations. Near the end of the TRW, after receiving a request on the $RE.req$ signal, the controller will trigger a signal that directs the Q-Flop to sample the E signal,

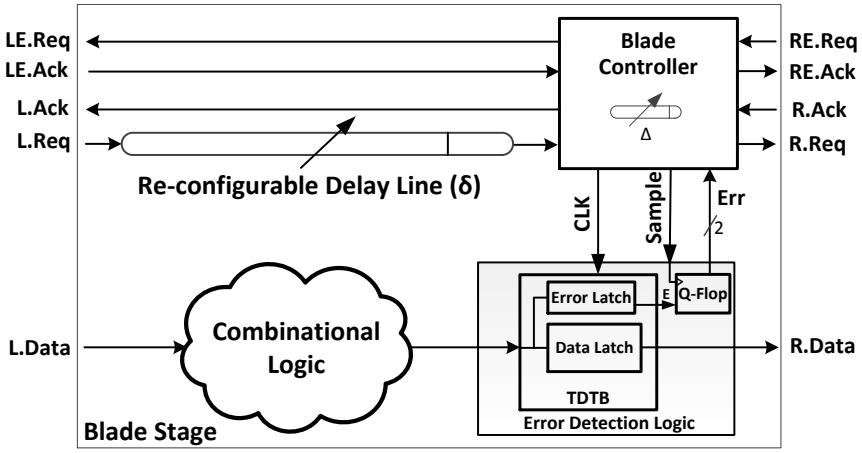


Fig. 1. The Blade architecture typical stage structure.

determining whether or not a timing error occurred during the TRW. If an error did not occur $RE.ack$ is immediately asserted, else Δ is triggered and only after that $RE.ack$ is asserted. Because the setup time of the TDTB Error Latch may be violated, the E signal may be metastable during sampling. To cope with this, the Q-Flop has a built-in metastability filter that guarantees metastability does not propagate to its Err output. In fact, this output is intentionally made a dual-rail signal that only becomes valid after the Q-Flop has safely determined if an error occurred or not. The controller simply waits for this to happen before acknowledging the error channel request via the $RE.ack$ signal. This ensures that metastability, while possibly causing an instantaneous cycle slowdown, does not propagate to the main control path. This is in stark contrast to synchronous schemes, which must wait for a fixed, larger metastability resolution time set to guarantee a sufficiently large mean time between failures (MTBF).

There are two main delay lines that affect the performance of Blade, δ and Δ. Compared to a traditional synchronous circuit, with clock period C, we set $C = \delta + \Delta$. The TRW (defined by Δ) must be large enough to capture even the worst-case datapath delay. However, a trade off in setting these values emerges, as increasing Δ allows δ to be smaller and the system to operate faster if no timing violations (errors) occur; on the other hand, the shorter stage-to-stage delay means that more transitions will occur while the latch is transparent, thereby increasing the frequency of errors that force subsequent pipeline stages to be delayed by the now larger Δ value. The optimal Δ depends greatly upon the amount of total variation (due to data and PVT variations) that can be expected in the design, and can range from 20% to over 60% [18] of the stage total delay. This is in contrast to Bubble Razor whose optimal error rate is less than 5% percent [15] and synchronous replay-based Razor schemes whose optimal error rate is less than 1% [7].

When Δ is sufficiently smaller than δ, the next stage has time to check whether the previous stage has an error before it makes its own latch transpar-

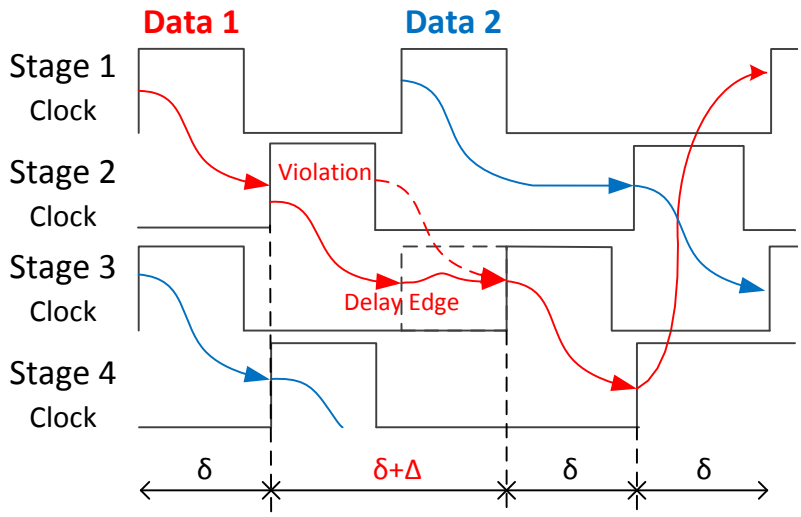


Fig. 2. Typical timing diagram for the Blade template.

ent, delaying the transparency phase if the previous stage had an error. Stage clocks will thus remain non-overlapping, as illustrated in Figure 2, making it easy to satisfy hold times. This is again in contrast to most synchronous resiliency schemes that make meeting hold time margins harder. Supporting larger values of Δ (w.r.t. δ) is also possible and is beneficial when data/process yield high variability. However, the result is that the transparency phases of neighboring stages clocks will overlap, and this may cause hold time issues similar to those seen in synchronous approaches (see [15] for an encompassing analysis). Managing these hold time issues in synchronous resiliency approaches is particularly challenging, as they cannot be fixed by slowing down the clock. Accordingly, hold times need to margined to a higher degree than setup times. As mentioned earlier, these hold margins are typically satisfied by adding hold buffers to the datapath, but the higher margins may make the number of added buffers impractically large for designs with high variability. In contrast, an asynchronous solution like Blade can easily add an additional programmable delay line to the backward control path, actively managing the degree of transparency overlap, which makes such extra margins unnecessary. In both cases the flexibility of the asynchronous solution makes managing hold time issues far more practical.

Lastly, note that Blade also uses programmable delay lines, because under significant PVT variations it may be difficult to achieve the optimal TRW, which captures the delay of all worst-case paths via static design analysis and optimization. Programmable delay lines allow customizing the actual delay post-silicon. In particular, the authors expect that during chip characterization delay lines are analyzed and optimally configured for every chip produced, subject to some quantization error. In particular, quantization errors in δ may lead to a non-optimal expected error rate, but the overall performance will remain close to

optimal [18]. Any additional margin needed to account for worst-case paths under PVT variations can be added only to the Δ delay line. Given the average frequency of timing violations can be in the range of 20%-40%, the impact of the added margin is only experienced 20-40% of the time, greatly reducing the percentage drop in performance. This is in contrast to non-resilient bundled-data designs (e.g., [10]) in which the added margin affects performance 100% of the time. As an example, a 10% increase in variation due to PVT can result in up to 30% margin penalty in synchronous designs; however, even considering a 40% rate of timing violations, the computed performance impact on Blade is less than 13% [19].

3 Preliminary CAD Flow

The authors' teams developed a preliminary flow to automatically convert single CLK domain, synchronous RTL designs to the Blade template using industry standard synthesis tools. The flow consists of various Tcl and shell scripts that drive the tools and a library of custom cells (e.g., the TDTB error latch), needed to make the template efficient.

In addition, to further reduce area and power overheads of the error detection logic, two microarchitectural optimizations are used. First, not every pipeline stage need be error-detecting, and non error-detecting stages can time borrow. Time-borrowing stages permit data to pass through the latch during the entire time it is transparent without flagging violations. The authors found that alternating between error-detecting and time-borrowing stages can work well as this effectively halves the overhead of error detection logic while still providing sufficient resiliency. Secondly, only latches that terminate near-critical paths [19] need to be error detecting, further reducing the number of EDLs in the entire design.

As Figure 3 illustrates, the flow has five main steps:

1) **Synchronous Synthesis:** The synchronous RTL is synthesized to a flip-flop (FF-based) design for given clock.
2) **FF to Latch Conversion:** FFs are converted to master-slave latches by synthesizing the design using a fake library of standardized D flip-flops (DFFs) that can be easily mapped to standard cell latches.
3) **Latch Retiming:** The latch-based netlist is retimed using a target TRW, where the combined path delay constraint of any two stages equals the given clock period. The purpose is to split the critical path in two parts, which enables hiding inter-stage Blade handshaking overheads.
4) **Resynthesis:** The retimed netlist is then resynthesized to reduce the number of TDTBs and increase performance of the final resilient netlist. In particular, re-synthesizing the logic happens such that the delay to a subset of latches is sufficiently fast to guarantee that data is stable before the latches go transparent (i.e., is not near-critical). This means that the latches do not need to be error-detecting, reducing the EDL overhead, and potentially reduces the error rate at the expense of increasing the datapath logic area.

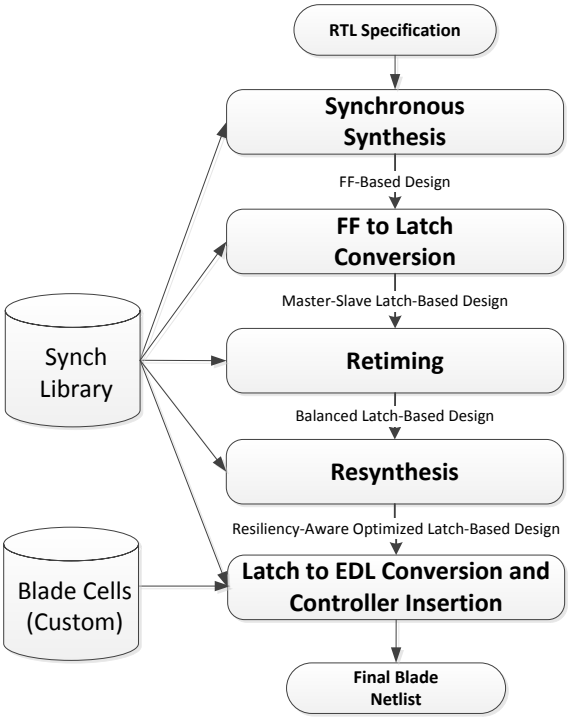


Fig. 3. The Blade design flow.

Targeting latches that cause the most errors in typical applications can lead to significant reductions in error rates with marginal increase in area. In [19] the authors employ a simple brute-force search, but more powerful means of identifying which subset of latches to speed up is an interesting area of future work.

5) **Blade Conversion:** The resynthesized latch-based netlist is then converted to the Blade template, by removing clock trees and replacing these with Blade controllers. The control logic, delay lines, and error detection logic are also inserted to create a final Blade netlist. There are many ways to implement the control logic [5]; using burst-mode specifications was explored in [19].

The authors' preliminary pre-P&R flow was tested and evaluated on a 3-stage version of Plasma [33], a MIPS OpenCore CPU, targeting a 28nm FD-SOI technology. The gate-level Blade design was compared to the equivalent synchronous design, and post-synthesis results demonstrate that for an area overhead of 8.4%, the Blade version of Plasma achieves a 19% average performance boost with a timing resiliency window of 30%. Out of the 8.4% area overhead, 32% is due to the use of EDLs and to the FF to latch conversion. With the removal of synchronous PVT margins, it led to an estimated 30%-40% improvement in performance [19].

4 Recent Developments and Open Research Problems

The technology is being commercialized by Reduced Energy Microsystems [34], a semi-conductor start-up company co-founded by William Koven, Dylan Hand, and Eleazar Vega-Gonzalez. They recently designed and fabricated a Blade-based design for light-weight encryption cyphers [20] and have plans on using Blade to build energy-efficient processors for the Internet of Things market. The success of the commercialization of Blade, however, will likely require solving several research challenges which we outline here.

4.1 Scope and Scale of Design

REM's recent work involved extending the flow illustrated in Figure 3 to start from a synthesizable-subset of SystemVerilogCSP [36] and include back-end place-and-route. In particular, they leveraged the USC-developed tool CSP2RTL to automatically decompose CSP designs into blocks of synthesized unconditional logic surrounded by efficient SEND/RECEIVE primitives. This tool allows asynchronous designers to couple the benefits of hierarchical decomposition and conditional communication with resilient pipelined designs and is based on a similar framework used for industrial-scale QDI designs [4].

Interestingly, the work in [20] represents just the beginning of what is possible when the scope of synthesizable CSP specifications is expanded. For example, we envision supporting arbitration in CSP where the CSP2RTL tool would automatically insert arbitrated merge blocks to enable the automatic design of complex routers and NoC designs. Moreover, we believe we can support mixed-timing interfaces in SystemVerilogCSP for which the tool will automatically insert clock-domain-crossing circuits. This will enable seamless integration of resilient bundled-data blocks within otherwise synchronous designs.

4.2 Area and Energy Efficiency

A CSP-based resilient design flow can support energy-efficient hardware design for a large variety of applications, but the added complexity associated with hierarchical design and conditional communication must be managed. In particular, over-decomposition can lead to increased overheads in terms of extra pipeline stages, excess error-detecting logic, and more delay lines and control logic. To properly navigate this increased design space, high-level energy and power estimation tools that guide decomposition will be important. In addition, the use of slack-less controllers such as in [20] can often provide much of the benefits of conditionality without the area overhead of a distinct pipeline stage. This design exploration should be guided by a notion of average-performance and average-energy consumption that considers the probabilities implicit in the conditional communication and the probabilities associated with error-rates at each error-detecting pipeline stage. In particular, understanding the implication of clustering latches into pipeline stages will likely be critical for a comprehensive flow that can accommodate industry-scale applications. Too few pipeline

stages and the design would loose the benefits of average-case data whereas too many pipeline stages would lead to too high overhead. Finding the right balance is likely design and use-case specific and will demand new tools to guide the design.

Even for simple RTL-based resilient bundled-data designs, numerous advances in area and power efficiency will likely be essential to the success of the technology. In particular, the design of the delay line and error-detecting logic is critical to efficiency. We expect one of the significant advantages of this design style is its ability to naturally support dynamic voltage scaling [21, 40]. Unlike a synchronous design in which adjusting voltages often requires stalling the pipeline for many clock cycles while the clock source is reconfigured, bundled-data circuits can be designed to automatically adapt to voltage scaling. If the delay line tracks the delay of the combinational logic, it need not be re-programmed when the supply voltage is changed. To support this strategy, we recently proposed a framework for delay line design [40] in which we minimize average energy subject to two-sided voltage scaling constraints. In fact, we anticipate we will need to build a library of delay elements and a CAD tool that chooses which delay elements to use based on an analysis of the likely critical path of a pipeline stage and its voltage scaling properties.

Minimizing the cost of error-detecting logic latches is also important. There are two distinct approaches to this problem. The first approach is re-synthesis in which new constraints are added to the logic synthesis / physical design to make certain latches non-critical, thereby saving the overhead of making them error-detecting. In [23], we developed a geometric programming based mathematical algorithm that guides re-synthesis to minimize the total area of the design. We have found that this often reduces error rates, but explicitly modeling and considering average-case performance is interesting future work. Moreover, we are exploring resilient-aware, latch-based retiming. Recall that the Blade CAD flow described above involves replacing FFs with a pair of latches and retiming of the slave latches to create a balanced latch based design. This balance aids in hiding the performance overhead of the asynchronous control and mitigating hold-time problems. Commercial tools support retiming of sequential elements, including latches, but the results are often sub-optimal for resilient designs as their retiming algorithm does not understand the inherent trade-off associated with near-critical paths and the error-detecting/non-error-detecting latch at which the path ends. The second approach is to design efficient multi-bit error-detecting latches. Amortizing the cost of memorizing whether an error occurs can lead to significant benefits in terms of area and power [22].

While most of the circuit design research has focused on super and near-threshold design, another important domain for Blade designs may be sub-threshold operation, particularly for the sub-set of the market in which performance is not important. Sub-threshold design, however, introduces new challenges in guaranteeing reasonable static noise margins and minimizing leakage currents. Fortunately, techniques to achieve efficient sub-threshold designs for synchronous circuits are well-known [2]. Using these techniques to design efficient efficient asynchronous control circuits, delay lines, and error-detecting

latches for the sub-threshold operation is an interesting and important area of research.

Finally, numerous researchers have developed bundled-data design flows using commercially-supported physical design tools [11, 16, 17] and REM has developed a prototype flow for Blade circuits [20]. To extend these to complex Blade designs, however, a few more enhancements will be necessary. First, new standard-cells must be designed, including efficient error-detecting latches and mutual exclusion elements. In addition, supporting the non-standard timing constraints and trade-offs associated with the introduction of programmable delay lines in complex Blade designs is novel and challenging. A naïve implementation can lead to a quadratic explosion of delay lines between interacting Blade stages. Instead, an intelligent sharing of delay lines is needed to guarantee using only a linear number of delay lines and this sharing should be guided by a variety of factors.

4.3 Design for Test, Debug, and Manufacturability

Traditional synchronous testing methodologies are based on an implicit assumption of statically controlled voltages and clocks and that the associated control logic is minimal (an on-chip PLL and off-chip voltage regulator). Traditional test methodologies have thus focused on the max-delay constraints in the core digital logic and relied on functional tests to cover the control logic. However, bundled-data resilient designs are more complex as they have programmable delay lines in every pipeline stage and have error-detecting logic that indicates when setup failures occur. One recent study explored the testability of the Blade template and found while many faults were implicitly testable by the error-detecting logic, other faults led to excessive errors or disabled the error-detecting capability of the circuit [26]. The complex nature of testing these circuits warrants the study of a holistic test methodology that encompasses new resiliency-aware fault models, test coverage, test generation, and design for test. This will include test methods for optimally tuning the programmable delay lines based perhaps on *in situ* error rate monitoring, as well as means to identify and discard chips with delay variations too large to correct.

5 Discussion and Conclusions

Asynchronous design has become an increasingly attractive alternative to synchronous design in several applications for a variety of reasons. For example, Intel showed that high-performance quasi-delay-insensitive (QDI) design is sufficiently robust and effective for high performance networking chips [13]. Moreover, the challenges of managing a global clock in large neuromorphic chips, have driven IBM [31] and Stanford [30] to adopt an asynchronous mostly QDI interconnect. Other academic researchers have found that built-in flow-control in bundled-data network-on-chips lead to significant benefits in terms of latency and area compared to synchronous counterparts [16]. However, efforts to commercialize bundled-data pipelines for processors demonstrated only marginal performance

benefits [10]. We hope that adding resiliency opens the door for much larger performance advantages to a broader range of applications.

Generally speaking, the range of architectures and applications for which resiliency adds value depends on two factors: the overhead one can expect from the error-detecting latches and the variance of the data and PVT dependent delays [37]. The benefits of a resilient design are higher when the fraction of combinational to sequential area is large, because the relative overheads of the TDTBs is smaller. Thus, resilient design favors less pipelined designs. Moreover, an architecture where the difference between average and worst-case delay is large will likely benefit more than a well balanced architecture and even more likely if the worst-case paths are rarely executed [37]. For example, architectures that involve complex logic with rarely executed long carry chains will benefit more than balanced designs consisting of many regular structures (e.g., memories). Fortunately, there are many architectural decisions that can be made to favor timing resilient templates [37].

Lastly, it is important to emphasize that the advantages of asynchronous resilient designs are difficult to approximate in synchronous architectures. In particular, asynchronous resilient designs adapt to the quite low average-case time it takes for metastability to resolve, which in principle can be unbounded. In contrast, the periodic nature of the clock forces synchronous alternatives to be designed for a much larger fixed resolution time, set by an acceptable MTBF. This difference enables our solution to be architecturally-independent, whereas existing robust synchronous solutions are forced to be based on recover and replay logic to obtain reasonable MTBF.

Thus, we believe asynchronous resiliency is a promising research direction to obtain efficient designs which adapt to the combination of PVT and data variations and naturally supports voltage scaling. We believe that a good initial market for this technology is the Internet of Things market, but the higher energy efficiency may very well be attractive to more general computing domains.

References

1. Akgun, O., Leblebici, Y., Vittoz, E.: Current Sensing Completion Detection for Subthreshold Asynchronous Circuits. In: European Conference on Circuit Theory and Design (ECCTD). pp. 376–379 (Aug 2007)
2. Alioto, M.: Ultra-low power VLSI circuit design demystified and explained: A tutorial. IEEE Transactions on Circuits and Systems - I: Regular Papers 59(1), 3–29 (Jan 2012)
3. Beer, S., Cannizzaro, M., Cortadella, J., Ginosar, R., Lavagno, L.: Metastability in Better-Than-Worst-Case Designs. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). pp. 101–102 (Apr 2014), (Fresh Ideas Workshop)
4. Beerel, P.A., Dimou, G.D., Lines, A.M.: Proteus: An ASIC flow for GHz asynchronous designs. IEEE Design & Test of Computers 28(5), 36–51 (2011)
5. Beerel, P., Ozdag, R., Ferreti, M.: A Designer's Guide to Asynchronous VLSI. Cambridge University Press (2010)

6. Bowman, K., Tschanz, J., Kim, N., Lee, J., Wilkerson, C., Lu, S., Karnik, T., De, V.: Energy-Efficient and Metastability-Immune Resilient Circuits for Dynamic Variation Tolerance. IEEE Journal of Solid State Circuits 44(1), 49–63 (Jan 2009)
7. Bowman, K., Tschanz, J., Lu, S., Aseron, P., Khellah, M., Raychowdhury, A., Geuskens, B., Tokunaga, C., Wilkerson, C., Karnik, T., De, V.: A 45 nm resilient microprocessor core for dynamic variation tolerance. IEEE JSSC 46(1), 194–208 (Jan 2011)
8. Chang, I.J., Park, S.P., Roy, K.: Exploring Asynchronous Design Techniques for Process-Tolerant and Energy-Efficient Subthreshold Operation. IEEE Journal of Solid State Circuits 45(2), 401–410 (Feb 2010)
9. Choudhury, M., Chandra, V., Aitken, R., Mohanram, K.: Time-Borrowing Circuit Designs and Hardware Prototyping for Timing Error Resilience. IEEE Transactions on Computers 63(2), 497–509 (Feb 2014)
10. Cortadella, J., Kondratyev, A., Lavagno, L., Sotiriou, C.: Desynchronization: Synthesis of Asynchronous Circuits From Synchronous Specifications. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25(10), 1904–1921 (Oct 2006)
11. Cortadella, J., Lupon, M., Moreno, A., Roca, A., Sapatnekar, S.: Ring Oscillator Clocks and Margins. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). pp. 19–26 (May 2016)
12. Das, S., Tokunaga, C., Pant, S., Wei-Hsiang, M., Kalaiselvan, S., Lai, K., Bull, D., Blaauw, D.: Razor II: In Situ Error Detection and Correction for PVT and SER Tolerance. IEEE Journal of Solid State Circuits 44(1), 32–48 (Jan 2009)
13. Davies, M., Lines, A., Dama, J., Gravel, A., Southworth, R., Dimou, G., Beerel, P.: A 72-Port 10G Ethernet Switch/Router using Quasi-Delay-Insensitive Asynchronous Design. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). pp. 103–104 (May 2014), (Industrial Track)
14. Dreslinski, R., Wieckowski, M., Blaauw, D., Sylvester, D., Mudge, T.: Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits. Proceedings of the IEEE 98(2), 253–266 (Feb 2010)
15. Fojtik, M., Fick, D., Kim, Y., Pinckney, N., Harris, D., Blaauw, D., Sylvester, D.: Bubble Razor: Eliminating Timing Margins in an ARM Cortex-M3 Processor in 45 nm CMOS Using Architecturally Independent Error Detection and Correction. IEEE Journal of Solid State Circuits 48(1), 66–81 (Jan 2013)
16. Ghiribaldi, A., Bertozzi, D., Nowick, S.: A Transition-signaling Bundled Data NoC Switch Architecture for Cost-effective GALS Multicore Systems. In: Design & Test in Europe (DATE). pp. 332–337 (Mar 2013)
17. Gibiluka, M., Moreira, M.T., Calazans, N.L.V.: A Bundled-Data Asynchronous Circuit Synthesis Flow Using a Commercial EDA Framework. In: Euromicro Conference on Digital System Design (DSD). pp. 79–86 (2015)
18. Hand, D., Huang, H., Cheng, B., Zhang, Y., Moreira, M.T., Breuer, M., Calazans, N.L.V., Beerel, P.A.: Performance Optimization and Analysis of Blade Designs under Delay Variability. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). pp. 61–68 (May 2015)
19. Hand, D., Moreira, M., Huang, H., Chen, D., Butzke, F., Li, Z., Gibiluka, M., M., B., Calazans, N.L.V., Beerel, P.A.: Blade - A Timing Violation Resilient Asynchronous Template. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). pp. 21–28 (May 2015)
20. Hand, D., Katzin, A., Koven, W.: Adding Conditionality to Resilient Bundled-Data Designs. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC) (May 2016), (Industrial Track)

21. Heck, G., Heck, L.S., Singhvi, A., Moreira, M.T., Beerel, P.A., Calazans, N.L.V.: Analysis and Optimization of Programmable Delay Elements for 2-Phase Bundled-Data Circuits. In: International Conference on VLSI Design (VLSI). pp. 321–326 (Jan 2015)
22. Hua, W., Tadros, R., Beerel, P.: Low area, low power, robust, highly sensitive error detecting latch for resilient architectures. In: International Symposium on Low Power Electronics and Design (ISLPED) (Aug 2016)
23. Huang, H.H.H., Cheng, H., Chu, C.C., Beerel, P.A.: Area optimization of resilient designs guided by a mixed integer geometric program. In: Design Automation Conference (DAC) (Jun 2016)
24. Jayakuma, N., Garg, R., Gamache, B., Khatri, S.: A PLA based Asynchronous Micropipelining Approach for Subthreshold Circuit Design. In: Design Automation Conference (DAC). pp. 419–424 (Jun 2006)
25. Kim, S., Seok, M.: Variation-Tolerant, Ultra-Low-Voltage μP With a Low-Overhead, Within-a-Cycle In-Situ Timing-Error Detection and Correction Technique. IEEE Journal of Solid State Circuits 50(6), 1478–1490 (Jun 2015)
26. Kuentzer, F., Amory, A.: Fault classification of the error detection logic in the blade resilient templates. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). pp. 37–42 (May 2016)
27. Kulkarni, J.P., Tokunaga, C., Aseron, P.A., Nguyen, T., Augustine, C., Tschanz, J.W., De, V.: A 409 GOPS/W Adaptive and Resilient Domino Register File in 22 nm Tri-Gate CMOS Featuring In-Situ Timing Margin and Error Detection for Tolerance to Within-Die Variation, Voltage Droop, Temperature and Aging. IEEE Journal of Solid State Circuits 51(1), 117–129 (Jan 2016)
28. Liu, J., Nowick, S., Seok, M.: Soft MOUSETRAP: A Bundled-Data Asynchronous Pipeline Scheme Tolerant to Random Variations at Ultra-Low Supply Voltages. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). pp. 1–7 (May 2013)
29. Liu, T.T., Alarcon, L., Pierson, M., Rabaey, J.: Asynchronous Computing in Sense Amplifier-Based Pass Transistor Logic. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). pp. 105–115 (Apr 2008)
30. Merolla, P., Arthur, J., Alvarez, R., Bussat, J.M., Boahen, K.: A Multicast Tree Router for Multichip Neuromorphic Systems. IEEE Transactions on Circuits and Systems - I: Regular Papers 61(3), 820–833 (Mar 2014)
31. Merolla, P., et al.: A Million Spiking-neuron Integrated Circuit with a Scalable Communication Network and Interface. Science 345(6197), 668–673 (2014)
32. Moreira, M.T., Hand, D., Calazans, N.L.V., Beerel, P.A.: TDTB Error Detecting Latches: Timing Violation Sensitivity Analysis and Optimization. In: International Symposium on Quality Electronic Design (ISQED). pp. 379–383 (Mar 2015)
33. Plasma CPU, 2014. Available: http://opencores.org/project,plasma
34. Reduced Energy Microsystems. http://www.remicro.com, Accessed: March, 2016
35. Rosenberger, F., Molnar, C., Chaney, T., Fang, T.P.: Q-modules: Internally Clocked Delay-insensitive Modules. IEEE Transactions on Computers 37(9), 1005–1018 (Sep 1988)
36. Saifhashemi, A., Beerel, P.A.: SystemVerilogCSP: Modeling digital asynchronous circuits using systemverilog interfaces. In: Proceedings of Communicating Process Architectures - WoTUG-33 (Jun 2011)
37. Sartori, J., Kumar, R.: Exploiting Timing Error Resilience in Processor Architecture. ACM Transactions on Embedded Computing Systems 12(2), 89:1–89:25 (May 2013)

38. Stevens, K.S., Gebhardt, D., You, J., Xu, Y., Vij, V., Das, S., Desai, K.: The Future of Formal Methods and GALS Design. Electronic Notes in Theoretical Computer Science 245(0), 115–134 (2009)
39. Sutherland, I.E.: Micropipelines. Communications of the ACM 32(6), 720–738 (Jun 1989)
40. Tadros, R., Hua, W., Gibiluka, M., Moreira, M.T., Calazans, N.L.V., Beerel, P.A.: Analysis and design of delay lines for dynamic voltage scaling applications. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC) (May 2016)
41. Tschanz, J., Bowman, K., Walstra, S., Agostinelli, M., Karnik, T., De, V.: Tunable Replica Circuits and Adaptive Voltage-Frequency Techniques for Dynamic Voltage, Temperature, and Aging Variation Tolerance. In: Symposium on VLSI Circuits. pp. 112–113 (2009)
42. Yakovlev, A., Vivet, P., Renaudin, M.: Advances in Asynchronous Logic: From Principles to GALS & NoC, Recent Industry Applications, and Commercial CAD Tools. In: Design & Test in Europe (DATE). pp. 1715–1724 (Mar 2013)

From an extended study of asynchronous controllers to system level design: application to the design of digital and analog interfaces

E. Beigne*, P. Vivet* and M. Renaudin**

* CEA LETI, Minatec Campus, 38054 Grenoble, France
** Tiempo, 38330 Montbonnnot, France Grenoble, France
edith.beigne@cea.fr, pascal.vivet@cea.fr,
marc.renaudin@tiempo-ic.com

Abstract. Asynchronous circuits have been proposed for many years and have successfully shown their robustness to delay variations making them a key solution for digital and mixed-signal circuits. More specifically, asynchronous circuits have demonstrated their efficiency in solving control, arbiter and interfaces issues. In order to make them more widely adopted, modeling and synthesis of asynchronous circuits have been proposed through different methodologies that we will firstly detail and discuss in this paper. We will then demonstrate the use of these specific methodologies for the synthesis and design of digital and analog-to-digital interfaces. The overall paper aims at providing a study and analysis of specific interfaces and their dedicated controllers for synchronization and energy management applications while discussing the asynchronous community positioning accordingly.

Keywords: Asynchronous circuits, controllers, interfaces, digital, analog, modelling, synthesis, verification, STGs, Petri Nets.

1 INTRODUCTION: TOOLS TO SOLVE INTERFACES ISSUES

This is now a cliché to say that there is a lack of tools for the design and verification of asynchronous circuits. But is that true? Not really, if we consider how challenging it is to enumerate all the tools that were developed and/or are still in use: Sis, Assassin, Forcage, Meat, Verdect, Syn, Versify, Tangram, Balsa, Teak, Cast, NCL, Ack, Tast, Pipefitter, Chp2Vhdl, Verilog2Stg, Desynch, Class, Haste, Tiempo-Tools, Petrify, Minimalist, Proteus, 3D…

So, why are there so many of them, and why people are complaining about a lack of tool? This is probably because there is still no consensus, neither about a modeling-language nor about a circuit-style. Therefore, many modeling languages and many asynchronous circuit styles are in use, not giving the opportunity of visible ones to emerge.

Prof. Yakovlev vision is focused, the language has to be suited to model, verify and synthesize asynchronous controllers, i.e. control circuits that are reactive to their input signals and able to drive their output signals, following a formal specification of concurrency, causality and choice. With this respect, a high level language is not necessary, there's no need for communicating processes, nor channels, but instead there is the need for modeling signals' behavior formally using concurrency and choice at a very fine grain. Exit Tangram, Balsa, CHP, HDLs, etc. and welcome Petri Nets [1] [2], STGs [3], or FSMs, even if there might be some links between them [4]. The spectrum of formalisms being narrowed, what is the appropriate graph based methodology and the supporting modeling language? PN, STGs, FSMs all target speed independent circuits which are robust enough to accommodate the variations the controllers have to resist, but they are not equivalent with respect to their ability to efficiently model and synthesize controllers.

Prof. Yakovlev and his team contributed to a major evolution of these graph-based approaches with the introduction of CPOG [5], because it prevents the designer from explicitly enumerating all the signal events and their traces together with their causal relations. CPOG is based on a description of the scenarios, i.e. the events traces, using a compact functional form. The signal encodings are specified apart, thus simplifying the controller's behavioral specification, and at the same time enabling synthesizing the controller with different encodings but keeping the same structural specification. Both aspects brought significant improvements to the modeling and design of asynchronous controllers.

So, now it's fine, let's develop design tools using all the concurrency theory, the computer science and electrical engineering knowledge and skills underneath, in order to apply and prove that all this research is solving relevant real and physical problems. With this respect Prof. Yakovlev vision is very wide, especially in the domain of interfaces we want to focus on in this paper. He proposed the specification, implementation and verification of many arbiters, synchronizers and controllers for the communication between digital circuits both synchronous and asynchronous, or between digital and analog circuits, and also controllers dedicated to phase encodings and analog circuits, etc…

Prof. Yakovlev scientific contribution is significant and well recognized; it was used and continues to inspire many works in different domains, such as the design of interfaces we choose to briefly address in this paper, because it illustrates very well the broad scientific knowledges involved.

2 DIGITAL INTERFACES FOR SYNCHRONIZATION

In advanced digital systems with nowadays large System-on-Chips and multicore architectures, one of the primary concern is the interconnect infrastructure, and how to implement efficient system communication for on-chip or off-chip communication. With tens of cores, hundreds of different clocks, and due to the always larger delay in wires compared to delay in gates, communication architecture and its clean implementation is a great challenge. Deep pipelining must be implemented to transport

information from one die side to its other side. It is more costly to exchange data than performing computation.

In this context, Prof. Yakovlev was a precursor and very early applied his knowledge of asynchronous controller design to the design of generic and abstract interfaces to implement system interfaces. In [6], he early proposed the model and design of controllers to implement system communication of arbitrary protocols. It was applied to a simple fifo interface but claimed to be generic for application to any kind of interfaces: “a unified solution to the problem of synthesizing logic implementations from abstract specifications. Although easily applied to control circuits, it is suitable for all types of interface hardware”. It is worth to notice how abstract models can help to derive logic interfaces implemented in asynchronous logic. Prof. Yakovlev studied many kind of interfaces, such as [7] proposing system communication using STG, allowing reader & writer to indefinitely wait for an answer, compared to earlier work using fundamental mode : such interfaces was then latency insensitive (at protocol and system level) and Speed Independent or QDI at signal level. In [8], the proposed asynchronous interfaces could cope with various timing and protocol interfaces.

One of the primary aspects of system communication is obviously synchronization: a clean synchronization of low level signals must be performed to ensure robust communication. On this domain, Alex Yakovlev has made on the long time a very wide study of synchronization and its formalization, both at electrical level with the study of synchronizer cell elements, and at logical level with the design of a full range of asynchronous arbiters, using various specification methods and proposing different services. He proposed for instance flat arbiters [9], priority arbiters [10], and parallel multi-resource arbiters [11], among many other ones. More recently, he also proposed opportunistic merge elements, as a way to efficiently control asynchronous events within a switch cap converter. One can notice that smart digital interface implementing synchronization may also be required for efficient control of the analog domain, as it will be presented in section III.

These contributions on interface synchronization and arbiter design have been a regular source of inspiration for the design of large scale system level interconnects. In the 2000’s, the Network-on-Chip paradigm has been introduced and many flavors where proposed, in conjunction with the Globally Asynchronous Locally Synchronous (GALS) paradigm as early introduced by D. Chapiro. The NoC architecture offers first of all a clear separation of computation and communication, compared to previous bus-based topologies, and NoC perfectly fits the GALS paradigm, where computations (core, accelerators, etc…) are implemented using synchronous methods while system communication are implemented using asynchronous methods. Many flavors of GALS NoCs are then possible: either multi-synchronous, or mesochronous, or fully asynchronous, using either FIFO like interfaces or pausable clocks interfaces. Many asynchronous Network-on-Chip have been introduced and implemented, such as Chain, QNoC, Mango, ANOC, Hermes-A and more recently a 2-phase bundle-data NoC.

For the CEA-LETI ANOC architecture, the proposal is based on wormhole packet switching, it includes virtual channels for Quality of Service, while routers and links

are implemented using Quasi Delay Insensitive logic with 4-phase / 4-rail protocol for robust system communications. The early arbiters design within the ANOC router were initiated by early works from Marc Renaudin and by priority arbiters [9] from Prof. Yakovlev proposal. Within CEA-LETI, a full series of circuit demonstrators have been developed using ANOC protocol, which was continuously improved by adding new features, such as on-chip and off-chip NoC interfaces, Design-for-Test wrapper, automatic router power down using activity detection, integration of DVFS scheme within NoC units, and with an automated flow for timing analysis and physical implementation. An extensive comparative study showed a clear benefit of ANOC versus its synchronous counterparts with a gain in power consumption by a ratio of 5.

For NoC topologies, a challenge is the design of efficient and robust NoC links using dense encodings. Alex Yakovlev proposed original contributions on protocol signaling for system communication. For instance in [12], he proposed a novel self-timed communication protocol based upon phase-modulation of a reference signal. The reference and the data are sent on the same transmission lines and the data can be recovered observing the sequence of events on the same lines phase, offering a compact code, while being robust to faults. In the same period, for system communication, people tried also to extend and optimize the existing 2-phase protocol (for reduced number of transitions compared to 4-phase), but using denser codes. For examples, the LETS code offers a generic solution for 2-phase multi-rail codes, their protocols and their associated protocol converters. More recently, a 1-of-T multi-rail code was proposed for 2phase signaling with efficient 2-phase/4-phase protocol converters.

These few examples of asynchronous protocols and corresponding protocol converters can be applied to 3D architecture. 3D technology using so-called Through-Silicon-Via (TSV) offer a new paradigm for further system integration, with strong benefits in terms of yield and system partitioning, power reduction due to shorter connections, and a full spectrum of new architecture by using heterogeneous technologies (logic, memory, NVM, MEMs, etc). Again, in such 3D architecture, scalable and modular system communication is a challenge, and it can be elegantly implemented as an asynchronous 3D Network-on-Chip, by offering power efficient and robust 3D asynchronous communications without any global 3D clocking.

To conclude this section, asynchronous logic offers an efficient way to implement digital interfaces at system level. GALS has still a long story to play. The main design challenges are still the same: synchronization, arbitration, protocol communication and signaling. Detailed modeling of the interfaces is mandatory to ensure reliability and robust design. More automation would help but due to the large spectrum of all these interfaces, it will be difficult [16]. Even if mature solutions exist, research can always be carried-on on these complex topics to continue innovation on protocol signaling and associated control schemes!

3 MIXED-SIGNAL AND ANALOG INTERFACES

Interfaces issues are more and more problematic today with the advent of power management circuit techniques and mixed-signal systems of the Internet-of-Things (IoT). In the first case, voltage and frequency generators are analog circuits, integrated on-chip, but controlling digital circuits. They require internal control loops and also need to be efficiently interfaced with digital blocks. IoT systems, on their side, are mixed-signal circuits in which many different interfaces are managed. In that kind of circuit, analog information is coming from sensors, wireless transfer or even energy harvesters. In that context, Prof. Yakovlev proposed many novel solutions from mixed-signal circuit workflow using asynchronous control [13] to Buck-boost DC-DC converters controls [14] and specific efficient IT controller called opportunistic merge element [15]. In the following we will discuss some of these works and put in perspective our current works using energy efficient asynchronous design.

3.1 The advent of mixed-signal circuits

To start with, it is obvious to say that, today, most of our circuits are mixed-signal. We use to call mixed-signal, a circuit which contains an analog and a digital part, interfaced with Analog-to-Digital or Digital-to-Analog converters. The reality is more complex: small digital circuits are necessary to control analog blocks. It can be implemented as control loops or finite-state-machines. Prof. Yakovlev proposed a framework for the design of mixed-signal systems with asynchronous control [13]. It enables formal verification of AMS systems with asynchronous control and the workflow is efficiently demonstrated on a buck-boost converter. In our lab, we implemented in 2007 an integrated micro battery charger which can be charged by a thermo-generator's DC/DC output or by an HF converter. The power supply manager consisted in a specific unit along with an asynchronous finite state machine to manage priority between the two different sources. Electrical simulations were performed but no framework was available at that time to formally verify our circuit's functionality. This could have been solved by Prof. Yakovlev work and the proposed AMS methodology [13] and would have improved a lot our verification step.

3.2 Power management circuits and limitations

One of the main mixed-signal components in today's circuits is related to power management and more precisely to voltage generation. Prof. Yakovlev demonstrated that power control can significantly benefit from the use of asynchronous logic [14]. They demonstrated how to design and verify a speed-independent multi-phase buck-boost converter. It has been shown to be extremely robust to the changes in power demand. In the same topic, in our lab, we first tried to get rid of classical synchronous DC-DC converter by proposing a Vdd-Hopping solution based on voltage dithering. In this scheme, two or three supply voltages are available on chip and delivered through power switches to the digital block. The main advantage is very high power efficiency as it only consumes during transitions. To avoid any undershoots and overshoots on

the digital block power supply, a local control loop is implemented. It was originally designed using synchronous logic, but we finally realized that an asynchronous loop would have probably been more robust to voltage changes as demonstrated by Prof. Yakovlev in a paper entitled “Design and Verification of Speed-Independent Multi-phase Buck Controller”. Another interesting application of asynchronous circuits at the interface of power supply and energy harvesters is presented in our lab in 2010. An asynchronous state-machine controls the energy levels between different energy sources (Photo-voltaic, Thermoelectric, etc.). The control is based on voltage level crossings on capacitances representing the energy level of the sources. We are still working on this topic today as, ideally; it could bring infinite energy autonomy to IoT circuits and systems making them fully energy-driven (similarly to asynchronous data-driven circuits).

3.3 Low power IoT circuits and systems

We have just discussed interfaces and control for mixed-signal circuits related to power management issues. In addition, in emerging IoT systems, many analog sensors have to interface with digital controllers. Events coming from these sensors are purely asynchronous and it would be inefficient in terms of energy to continuously sample them as they can occur in very different time scales. An efficient solution is to handle these events as asynchronous Interrupts (ITs) computed by an asynchronous IT controller (IT-Ctrl) as shown in our very recent works. An asynchronous IT-Ctrl manages all the peripherals events and sends the priority number to a decoder. Interruptions are sorted by interruption numbers with the highest priority on interrupt number 0 and weakest priority on interrupt number 31. This scheme is interesting in terms of wake-up as, due to its asynchronous implementation, it immediately reacts on an incoming interrupt. However, the priority could be treated in a different way to improve this controller’s energy efficiency. Indeed, Prof. Yakovlev’s work proposes an element named opportunistic merge element which could greatly improve our IT controller performances. This innovative asynchronous component merges two or more request-acknowledge channels into one and is allowed to opportunistically send requests if they arrive close to each other. Our future works aim at improving our current IT controller with the help of an opportunistic scheme as described [15] by Prof. Yakovlev work.

4 CONCLUSION

We have discussed in this paper interfaces issues in digital and mixed-signal circuits. These interfaces can be implemented between synchronous domains or between analog to digital and digital to analog parts of circuits. They are also seen as a key issue between synchronous and asynchronous timing domains. We have shown how Prof Yakovlev’s work has been focused, in part, to solve these issues by proposing many different methodologies and innovative implementation schemes. Our aim has been to put in perspective our own work with respect to his research all along the paper and to

show that most of our proposals are complementary or could have been even greatly improved by Prof Yakovlev's proposals.

References

1. A. Yakovlev, A. Koelmans, A. Semenov, D. Kinniment, "Modelling, analysis and synthesis of asynchronous control circuits using Petri Nets", Integration: the VLSI Journal, 1996, Vol. 21, N°. 3, pp. 143-170.
2. A. Yakovlev, A. Koelmans, "Petri Nets and Digital Hardware design", in W. Reisig, G. Rosenberg ed., Lectures on Petri Nets II: Applications, Advances in Petri Nets, Berlin, New York, Springer-Verlag, 1998, 154-236.
3. D. Sokolov, A. Bystrov, A. Yakovlev, "STG optimization in the direct mapping of asynchronous circuits", Design Automation and Test in Europe (DATE), 2003, Munich, Germany.
4. M. Renaudin, A. Yakovlev, "From Hardware Processes to Asynchronous Circuits via Petri Nets: an Application to Arbiter Design", Workshop on Token Based Computing (ToBaCo), part of the 25th International Conference on Applications and Theory of Petri Nets (ICATPN'04), 2004.
5. A. Mokhov, A. Yakovlev, "Conditional Partial Orders Graphs: Models, Synthesis, and Application", IEEE Transactions on Computers, 2010, Vol. 59, N°. 11, 1480-1493.
6. A. Yakovlev, A. Koelmans, L. Lavagno, "High-level modeling and design of asynchronous interface logic", IEEE Design & Test of Computers, 1995, vol. 12, issue 1, pp. 32-40.
7. F. Xia, A. Yakovlev, D. Shang, A. Bystrov, A. Koelmans, D. Kinniment, "Asynchronous communication mechanisms using self-timed circuits", Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems, 2000, pp. 150-159.
8. F. Xia, A. Yakovlev, I. G. Clark, D. Shang, "Data communication in systems with heterogeneous timing", IEEE Micro, 2002, Vol. 22, Issue 6, pp. 58-69.
9. A. Mokhov, V. Khomenko, A. Yakovlev, "Flat Arbiters", Fundamenta Informaticae, 2011, 108(1-2), pp. 63-90.
10. A. Bystrov, D. Kinniment, A. Yakovlev, "Priority Arbiters", International Symposium on Advanced Research in Asynchronous Circuits, April 2000, pp. 128-137.
11. D. Shang, F. Xia, A. Yakovlev, "Highly parallel multi-resource arbiters", Proceedings of 2010 IEEE International Symposium on Circuits and Systems, 2010, pp. 4117-4120.
12. C. D'Alessandro, Delong Shang, A. Bystrov, A. Yakovlev, O. Maevsky, "Multiple-rail phase-encoding for NoC", 12th IEEE International Symposium on Asynchronous Circuits and Systems, 2006, pp.110-116.
13. V. Dubikhin, C. J. Myers, A. Yakovlev, D. Sokolov, "Design of Mixed-signal Systems with Asynchronous Control", IEEE Design & Test, 2016, 10.1109/MDAT.2016.2555916.
14. D. Sokolov, V. Khomenko, A. Mokhov, A. Yakovlev and D. Lloyd, "Design and Verification of Speed-Independent Multiphase Buck Controller," Asynchronous Circuits and Systems (ASYNC), IEEE International Symposium on, Mountain View, CA, 2015, pp. 29-36.
15. A. Mokhov, V. Khomenko, D. Sokolov and A. Yakovlev, "Opportunistic Merge Element," Asynchronous Circuits and Systems (ASYNC), 2015 21st IEEE International Symposium on, Mountain View, CA, 2015, pp. 116-123.
16. A. Yakovlev, P. Vivet, M. Renaudin, "Advances in Asynchronous Logic: from Principles to GALS & NoC, Recent industry Applications, and Commercial CAD Tools", in Conference on Design, Automation and Test in Europe (DATE'13), 2013, Grenoble, France.

A Community of Asynchronauts: 20+ Years of the ASYNC Conference

Erik Brunvand

School of Computing
University of Utah
Salt Lake City, UT 84112

Abstract. Since its founding in 1994, the IEEE Symposium on Asynchronous Circuits and Systems has been a premiere venue for publishing results from the asynchronous research community. Perhaps more importantly, it has also been an annual meeting where people gather, form and renew friendships, and build a strong sense of community. In this paper I will give a brief history of the ASYNC Symposium with a special focus on the social events that have contributed so much to the tremendous sense of community we enjoy among asynchronous researchers.

1 Introduction

Researchers interested in asynchronous circuits and systems have always been a bit on the fringe of the computer engineering world. Although there were interesting examples of asynchronous approaches in the early days of computer design, the codification of a synchronous design style, and the subsequent support for that design style in computer-aided design tools, resulted in the vast majority of digital systems using a synchronous timing regime. In spite of that inertia, an intrepid group of researchers has continued to be intrigued by the possibilities of asynchronous approaches, both in terms of design (at circuit and system levels) and analysis/theory.

Like researchers in many "niche" areas, their results were sometimes not appreciated by the larger research community, and often had to struggle to be recognized at the larger conferences and journals. As sometimes happens, when the critical mass of research becomes great enough in an area, this spawns a new conference series devoted more specifically to that area of study.

In the case of asynchronous design, this critical mass was reached in the early 1990's. Leading up to this point researchers such as Chuck Seitz at Caltech [36, 39, 38], Charles Molnar, Tom Chaney and Wes Clark at Washington University in Saint Louis [13, 12, 14, 25], Steven Unger at Columbia [49, 50], and Victor Varshavsky at the St. Petersburg Electrical Engineering Institute [51–53] were doing foundational work from the late 1960's to the early 1980's, without the benefit of a specific conference venue.

One standout conference series that included a nice set of early asynchronous and self timed papers was the Caltech Conference on VLSI which would become

the Advanced Research in VLSI (ARVLSI) conference series. The very first Caltech Conference in 1979, for example, had three seminal papers on asynchronous subjects [45, 48, 37] and the 1983 version of that conference included another set of important papers on asynchronous circuits and systems [16, 17, 40].

In the 1980's researchers such as Alain Martin at Caltech [19–21], Bob Sproull and Ivan Sutherland at Caltech and Carnegie Mellon University [41, 47, 15], and Theresa Meng at Stanford [23, 22, 24] were extending the work of earlier pioneers, and they, along with the pioneers, were producing a new generation of asynchronous researchers including Peter Beerel [2, 3, 1], Erik Brunvand [10, 7–9], Chris Myers [27, 28, 26], Steven Nowick [29–31], and Ken Stevens [44, 42, 43], Kees van Berkel [4–6], Alex Yakovlev [54, 18, 55] , and many others (note that references chosen in this section are specifically the early works from these selected researchers).

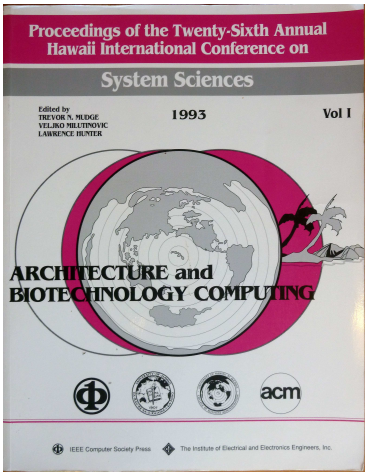
With the backdrop of the expanding world of asynchronous and self-timed research, the time was right for a conference devoted to this research area.

2 Pre-history: HICSS 1993

The first foray into thinking about a discipline-specific conference was to propose and organize a special session on Asynchronous and Self-Timed Circuits and Systems at an existing conference. Erik Brunvand and Ganesh Gopalakrishnan from the University of Utah took up the challenge and organized just such a special session at the 1993 Hawaiian International Conference on System Sciences (HICSS) (see Figure 1). This conference had the double benefit of being open to such a special session on a niche topic, and also being held in Maui, Hawaii in January. The conference "mini-track" was a huge success with 14 papers accepted and presented by most of the leading researchers in the area. In fact, the Hawaiian conference organizers were apparently not prepared for the success of the mini-track, and for the avid nature of the asynchronous research community. The conference session was held in a tiny room that was overflowing with attendees anxious to participate in the session and hear about the research. The organizers of the conference were apparently thinking that people would be at the beach rather than listen to papers on such a topic! But the interest and enthusiasm was clear. That session was enough to encourage the organizers to think in grander terms and organize a whole conference dedicated to the subnect of asynchronous and self-timed circuits and systems research.

3 Async 1994: Salt Lake City, UT, USA

The first official IEEE Symposium on Advanced Research in Asynchronous Circuits and Systems was organized and held in November 1994 in Salt Lake City at the University of Utah. The conference was primarily organized by: General Chairs Erik Brunvand (University of Utah), Al Davis (University of Utah), and Program Chairs Ganesh Golpalakrishnan (University of Utah) and Steven Nowick (Columbia University). The name for the conference was a bit of a mouthful,



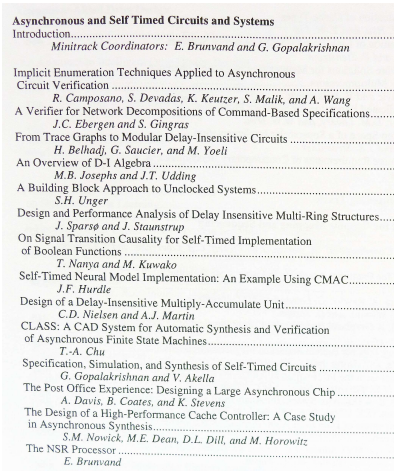


Fig. 1. The proceedings cover, and the session contents, for the Special Session on Asynchronous and Self-Timed Circuits and Systems at the 1993 Hawaiian International Conference on Systems Science held on Maui, Hawaii.

but chosen to echo the name of the premiere VLSI conference of the day, the conference on Advanced Research in VLSI (ARVLSI). The Async conference name was eventually shortened to remove the "Advanced Research in" portion of the name.

The conference was organized with support from the IEEE Technical Committee on VLSI - support that it continues to have to this day. It was also started with a small grant from the National Science Foundation to support student attendance at the conference, and also non-monetary support from IFIP Working Groups 10.2 and 10.5. The November time frame of the original conference was designed to be loosely compatible with the International Conference on Computer Aided Design (ICCAD) so that attendees could plausibly come to both conferences one after the other, ICCAD being held in November 1994 in Santa Clara, California.

The program committee for the first Async conference in 1994 reads like a "who's who" of asynchronous and self-timed researchers at the time: Graham Birtwistle, Steven Burns, Raul Camposano, Tam-Anh Chu, David Dill, Steven Furber, Luciano Lavagno, Bill Lin, Alain Martin, Teresa H.-Y. Meng, Charles Molnar, Martin Rem, Jens Sparsø, Robert Sproull, Pasupathi (Subra) Subramanyam, Jan Tijmen Udding, Steven Unger, Kees van Berkel, Peter Vanbekbergen, and Alex Yakovlev. The conference registration fee for IEEE members was $225, and students could register for $65. The conference included 25 refereed papers and one invited paper. The keynote speaker was Ivan Sutherland (Sun Microsystems).

Perhaps the most memorable keepsake from this first Async conference was the "Async wallet card" (see Figure 2). This card, given to all conference atten-

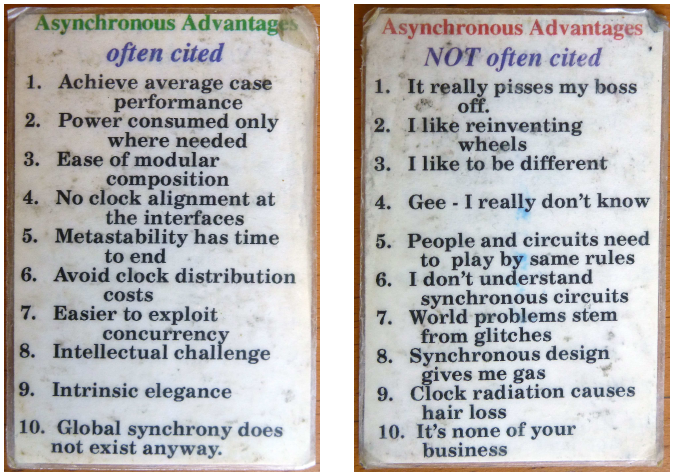


Fig. 2. The front and back sides of the wallet card given to Async 1994 attendees. This card was designed and produced by Al Davis and Erik Brunvand.

dees, was designed and produced by Al Davis and Erik Brunvand back in the day when color printing and laminating had to happen at a commercial printing shop. The card featured a list of the most common asynchronous advantages that we had seen repeated in virtually every paper, along with humorous "advantages" not often cited in papers. For many years afterwards at future Async conferences a challenge was given by Ivan Sutherland to hold up your Async wallet card if you still had it with you.

Although there was no organized outing at this first Async conference, it was definitely a starting point for the community of Asynchronauts. Friendships and research collaborations initiated at this very first Async conference persist to this day. From my clearly biased perspective, it was a rousing success!

4 Async 1996: Aizu, Japan

Although the continuation of any new conference is not a sure thing, the success of the first Async Symposium meant that this conference series would definitely continue. For the second conference, a group of Russian ex-pats who had found an academic home, for the moment at least, in Aizu-Wakamatsu, Japan, would host the second conference, along with their Japanese colleagues. The General Chair for the second conference was Tosiyasu Kunii, a largely ceremonial position for a senior researcher at the University of Aizu. The actual Conference Chairs were Takashi Nanya (Tokyo Institute of Technology) and Alex Kondratyev (University of Aizu). The Program Chairs were Luciano Lavagno (Polytecnico di Torino) and Alexander Taubin (University of Aizu). There were 24 papers accepted, and two embedded (invited) talks by Rajit Manohar (Caltech) and Alain Martin (Caltech), and Steve Furber (Univrsity of Manchester).

Fig. 3. Dinner group from the conference excursion to the Japanese baths at Async 1996. Attendees in this photo are (clockwise from left): Takashi Nanya, Chris Myers, (unknown - perhaps Peter Beerel?), Bill Richardson, Steve Furber, Erik Brunvand, and Doug Edwards

While the symposium format did not change much, the time frame did. Having the first conference in November was an attempt to let attendees amortize travel to Async 1994 and to ICCAD 1994. This didn't make as much sense for a conference held in Japan, and the organizers wanted to move to a spring conference schedule. So, the second conference was held from March 18-21 1996. The year-and-a-half gap meant that there would be no Async 1995 in the series, but the conference would actually be on-schedule for its second year.

One major social outing for the second conference was having the conference banquet in a Japanese bath. The conference attendees enthusiastically embraced the opportunity to experience this Japanese tradition. After washing carefully, the attendees soaked in hot baths, recovered in cool baths, and donned traditional Japanese robes for dinner at the baths. This was possibly the most relaxed collection of researchers ever assembled at a conference banquet!

After the conference a second outing was organized to a local ski area. Many conference participants assembled at the ALTS Bandai Resort near Aizu. The skiing conditions were good, but quite spring-like including a short bout of rain on the slopes, but great fun was had by all the skiers in the group. The resort's ski rental facilities, however, were greatly stressed by a group of large western visitors all requesting large ski boot sizes! The rental facility ran out of large ski boots and had to have more boots shipped in from a nearby resort.

5 Async 1997: Eindhoven, The Netherlands

The third Async conference was held in another hotbed of asynchronous research: The Netherlands. Starting with Martin Rem [33–35] there had been a flurry of papers from Dutch researchers that had been influential especially in the area of synthesizing circuits from program descriptions. The third conference was held in Eindhoven, The Netherlands, from April 7-10, 1997, with cooperation from Eindhoven University of Technology and Philips Research Labs where some of the Dutch researchers had landed. The conference General Chair was Martin Rem (Eindhoven University of Technology) with Co-Chair Peter Hilbers (Eindhoven University of Technology). The Program Chairs were Kees van Berkel (Philips Research Laboratories) and Mark Josephs (South Bank University, UK). The conference included 25 accepted papers and a set of five invited keynote lectures from Cees Niessen (Philips Research Labs), Roger Brockett (Harvard University), Steve Furber (University of Manchester), Ivan Sutherland (Sun Microsystems), and Hiroaki Terada (Osaka University). The conference banquet was held at the Philips Evoluon: a Philips showcase housed in a building shaped like a flying saucer (really!).

Fig. 4. Pre-conference outing during Async 1997. Attendees are (from left): Erik Brunvand, Ken Yun, and Ken Stevens.

The conference was actually held at a conference center in Veldhoven, a small town close to Eindhoven. The conference center had all the facilities for the conference under one roof including a fascinating set of sports and recreation facilities. The recreation opportunities included swimming, sauna, darts,

billiards, and a wonderful 9-pin bowling alley. This style of bowling is a European version, played since medieval times, and is quite different from the North American 10-pin bowling. There are, as you might expect, nine pins, arranged in a diamond shape and having strings on the top of each pin to hoist them back to their starting positions. The ball is 16cm in diameter and has no finger holes. Async conference attendees were enthusiastic bowlers, especially after a few drinks, in the evenings at Async 1997.

6 Async 1998: San Diego, CA, USA

Returning to the United States, the fourth Async conference was held in SanDiego, California from March 30-April 2 1998. The General Chair was David Dill (Stanford University) and the Program Co-Chairs were Peter Beerel (University of Southern California) and Ken Yun (University of California, San Diego). The conference program featured 23 papers, and keynotes by Steven Unger (Columbia University), Ivan Sutherland (Sun Microsystems), and Mark Horowitz (Stanford University). The conference social events included a visit to Qualcom, a respected high tech company headquartered in San Diego, a visit to the historic Hotel del Coronado, a San Diego landmark, and a California beach party for the conference banquet. This conference also marks many attendees' first encounter with a San Diego delicacy that was relatively unknown at the time, but has since become a wide success, the "fish taco."

Fig. 5. Included in the conference registration at Async 1998 was a pair of "Async socks" with the C-element control circuit for a micropipeline and the conference date.

A standout souvenir from the Async 1998 conference was the "Async socks" given to each conference attendee (see Figure 5). These socks were sponsored by Sun Microsystems and featured Async98 and a series of C-elements organized into a micropipeline-style two-phase control circuit [47]. This basic circuit structure of "half-cocked" C-elements connected into a FIFO-like circuit would become a common theme in future Async conference logos with some characteristic glyph representing the conference location taking the place of the C-element.

Fig. 6. Winners of the Best Paper Award at Async 1999 posing with their trophy: "Sync: the evil dragon that must be destroyed" (based on an original sculpture by Gaudí). They are (from left) Rakefet Kol, Chris Myers, Ken Stevens, Ran Ginosar, and Peter Beerel. Not shown are additional authors Shai Rotem and Ken Yun. (photographer unknown)

7 Async 1999: Barcelona, Spain

In 1999 the Async conference moved back to Europe to Barcelona, Spain from April 18-22. the General Chairs were Jordi Cortadella (Universitat Politècnica de Catalunya) and Mark Josephs (South Bank University). The Program Chairs were Steven Nowick (Columbia University) and Alex Yakovlev (University of Newcastle upon Tyne). The program consisted of 21 papers and keynotes from Richard Lyon (Apple Computer), Mike Gordon (Cambridge University), and Wesley Clark (Washington University). Dr. Clark's lecture, entitled "Asynchronous

Macromodules Were a Pain to Build but a Joy to Use" was especially memorable. The Macromodules project at Washington University in St. Louis, MO, USA in the 1960's involved the design and implementation of shoebox sized asynchronous computing modules connected through asynchronous interconnections [32, 46]. They were large physical versions of what we would imagine as VLSI circuit modules today, and were used to build many examples of "... arbitrarily large and complex computers that work." [32] Dr. Clark even brought some original macromodules to the conference to show.

While in Barcelona the conference attendees were treated to not only wonderful Catalan food, but tours of the city showing off some of the famous sites, including many sites designed by perhaps the most famous Catalan artist Antoni Gaudí (although painter Joan Miró might dispute that claim). The city of Barcelona has many wonderful buildings and sculptures that were designed by Gaudí in his characteristic flamboyant style. The Best Paper award at Async 1999 was a small reproduction of Gaudí's famous lizard sculpture. The original lizard sculpture is found in Parc Güell in Barcelona and is known locally as "El Drac" (The Dragon). It was made in collaboration with another artist, Joseph Maria Jujo, out of concrete and ceramic tiles. For the Best Paper award, the smaller version was christened "Sync, the evil dragon that must be destroyed" (See Figure 6).

8 Async 2000: Eilat, Israel

Ran Ginosar (Technion) organized the 2000 Async conference as General Chair in Eilat, Israel. The Program Chairs were Steve Furber (University of Manchester) and Mike Kishinevsky (Intel). The conference program featured 20 papers and keynote addresses by Avinoam Kolodny (Intel), Shimon Even (Technion), and Udi Shapiro (Weizmann Institute of Science). The conference also presented a full day of hands-on tool demos showing off a set of tools that would come to be seen as hugely influential in the asynchronous research community: ATACS by Chris Myers (University of Utah), Petrify by Jordi Cortadella (Universitat Politècnia de Catalunya), Minimalist by Steven Nowick (Columbia University), and Balsa by Doug Edwards (University of Manchester).

As exciting as the tool demos and conference program were, perhaps the highlights of the Israel conference were the location in Eilat, and the excursions organized for conference attendees. Eilat is a resort town at the southern tip of Israel on the Red Sea. The conference hotels were essentially right on the beach, and the scuba and snorkeling opportunities were tremendous. The conference banquet was held at an undersea restaurant where the windows from the dining tables looked out to an underwater scene with curious fishes looking back at the dinner guests.

On Friday after the conference there was a one-day excursion to the ancient city of Petra in neighboring Jordan. This 2000 year old city (established as early as 312 BCE) was the capital of the Arab Nabataens. The buildings in Petra are carved out of living rock in the sandstone cliffs of the Petra valley (Figure 8).

Fig. 7. A reunion at Async 2000 in Eilat, Israel of some members of the asynchronous research group of Victor Varshavsky. From left: Alex Kondratyev, Alexander Taubin, Mike Kishinevsky, Victor Varshavsky, Alex Yakovlev, and Masha Yakovlev. (photographer unknown)

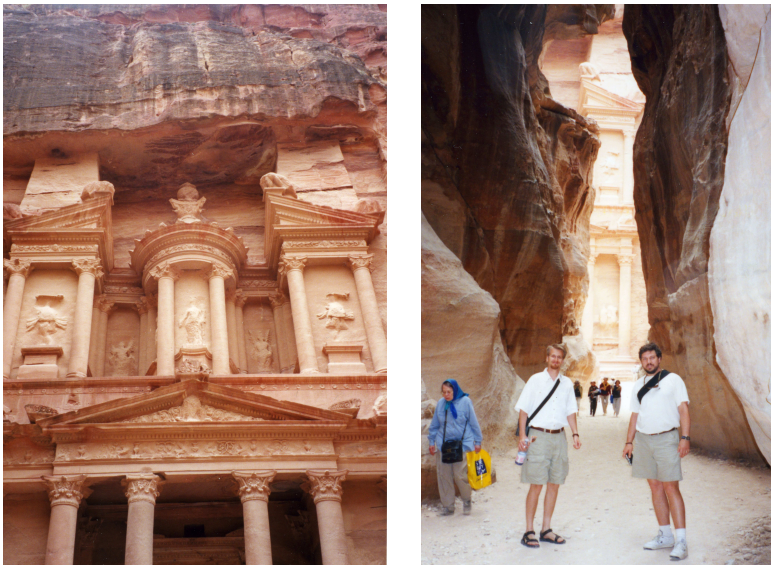


Fig. 8. Images from the excursion to Petra, Jordan during Async 2000. On the left is a building from the archeological site of Petra. On the right are attendees Hans Jacobsen and Erik Brunvand in the narrow canyon (the Siq) leading to the main Petra city.

Amazingly, this spectacular site (a UNESCO World Heritage Site since 1985) was unknown to the western world until 1812 when word of the city was spread by a Swiss explorer. In a poem by John William Burgon Petra was described as "a rose-red city half as old as time." [11] The conference attendees who went on this excursion were awed by the city, and slightly worried as the buses were driving back to the border with Israel trying to make sure they reached the border before sundown and the beginning of the Jewish sabbath when the border would close (the buses made it with minutes to spare).

The second excursion was a two-day tour of the "best of" Israel featuring the Dead Sea (see Figure 9 with some Async attendees "floating like a cork" in the Dead Sea), the ancient fortress of Masada (the last stronghold of ancient Israel that fell to the Romans in 70 CE), and Jerusalem (Holy city to at least three major world religions: Judaism, Christianity, and Islam). The conference time of Spring 2000 was a calm time in the Middle East and was a perfect time for the conference, and the Asynchronauts to visit.

Fig. 9. Async 2000 conference attendees floating in the Dead Sea. The floaters are, in the front row (left to right) Kees van Berkel and Charlie Molnar, and in the back row Jo Ebergen and (unknown). (photo by Jo Ebergen)

9 Async 2001: Salt Lake City, UT, USA

In 2001 the Async conference returned to Salt Lake City and the University of Utah. This was the first time that a location for the conference had been

repeated, and the conference organization included some familiar names from the original 1994 conference. The General Chair was Erik Brunvand, and the Program Co-Chairs were Chris Myers and Al Davis (all from the University of Utah). Other names in the 2001 Symposium Committee that were also in the original 1994 Committee included Ganesh Gopalakrishnan (Finance Chair) and Steven Nowick (Best Paper Chair). The conference program included 20 papers and keynotes from Bill Athas (Apple Computer), Kevin Normoyle (Sun Microsystems), and Ajay Koche (Agilent Laboratories) (who had attended the original 1994 conference as a student). In the Message from the Chairs it was noted that the original conference name (IEEE Symposium on Advanced Research in Asynchronous Circuits and Systems) was meant to evoke the history of the highly influential Advanced Research in VLSI (ARVLSI) conference series. For this occasion in 2001 the ARVLSI conference was co-located with Async 2001 with Async being the first two days (March 12-13, 2001, a shared day in the middle (March 14) and then ARVSI taking over for the final two days (March 15-16). The banquet on Wednesday March 13 was combined for the both conferences. Ironically, 2001 was also the year that the Async Steering Committee voted to shorten the official name of the Async conference to leave out the "Advanced Research" description. This was also, sadly, the final offering of the venerable ARVLSI conference.

Fig. 10. The multi-colored scarf given to conference attendees in Salt Lake City at Async 2001. This scarf proved very valuable for finding other Asynchronauts during the ski outing to Park City Mountain Resort. The combined logo shows that Async was co-located with the 2001 Conference on Advanced Research in VLSI (ARVLSI).

The conference attendees at Async 2001 were given a multi-colored scarf with the logos of both Async 2001 and ARVLSI 2001 conferences (Figure 10). These scarves were greatly appreciated by the attendees and proved to be a valuable way of identifying conference skiers during the ski excursion. The Wednesday afternoon excursion took skiers and sightseers to the resort town of Park City, Utah, just 40min from the conference site. Good skiing was had by all with no rain this time (see the 1996 excursion description...).

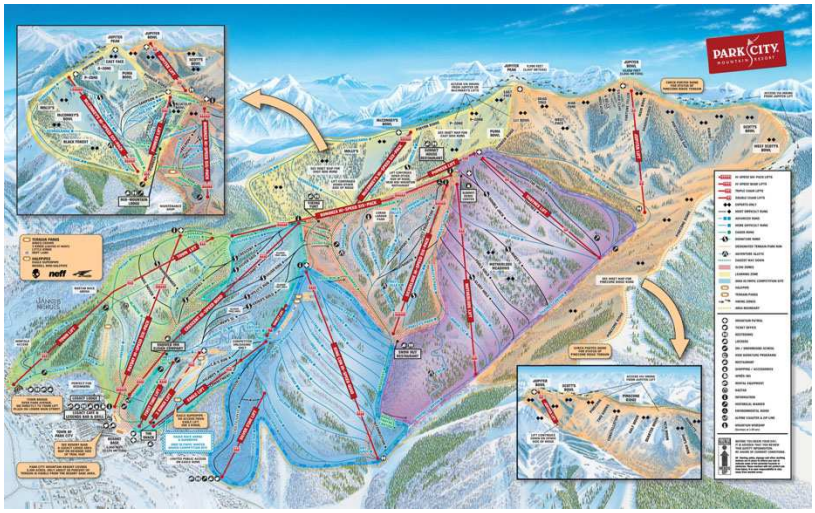


Fig. 11. Trail map from Park City Mountain resort - site of the ski excursion from Async 2001.

10 Async 2002: Manchester, U.K.

In 2002 the conference was held in Manchester, U.K. organized by General Chair Steve Furber (University of Manchester) and Program Co-Chairs Marly Roncken (Intel) and Simon Moore (University of Cambridge). The program consisted of 21 papers and keynotes by Robin Saxby (ARM Ltd.), Russel Cowburn (University of Durham), Nick Foggin (Orange), Andrew Lines (Fulcrum Microsystems), and Uri Cummings (Fulcrum Microsystems). The conference signature gift was an "Async umbrella" with the conference logo. Sadly, in some sense, the weather at the conference (April 8-11, 2002) was beautiful and sunny with no need for an umbrella!

The conference banquet was held at the Manchester Museum of Science in the "steam hall" surrounded by large, impressive steam engines, many of which were fired up and running earlier in the visit. The other notable exhibit was the

faithful replica of the Manchester Small-Scale Experimental Machine (SSEM), nicknamed "Baby," and arguably the worlds first stored program computer. The Baby was designed and built in Manchester in 1948 as a testing interface for the Williams Tube CRT-based memory system also being developed at the time. In 1998 a working replica of the SSEM was built to celebrate the 50th anniversary of the running of its first program. The banquet speaker at Async 2002, Chris Burton, described the rebuilding effort and demonstrated the Baby replica in operation. Interestingly, he related that the most difficult part of the reconstruction was not finding the vacuum tubes or the other electrical components, but finding original examples of the metal racks that housed the Baby. Apparently almost all such racks had been scrapped after the project was originally completed. The final rack used in the reconstruction had been found in a local farmer's barn being used to hold farm equipment.

Fig. 12. Banquet talk by Chris Burton of the Computer Conservation Society at Async 2002. Chris led the rebuilding work of the Manchester SSEM - the world's first stored program computer - which was reconstructed for its 50th anniversary in 1998. The replica is now on display in the Museum of Science and Industry and Chris demonstrated the machine in operation before the banquet.

11 Async 2003: Vancouver, BC, Canada

The ninth Async conference was held in Vancouver, B.C., Canada from May 12-16, 2003. This was the first time the conference had been held in Canada,

and the latest in the year the conference had been held. The General Chairs were Mark Greenstreet (University of British Columbia) and Jo Ebergen (Sun Microsystems). The Program Co-Chairs were Jo Ebergen (in a rare dual role) and David Kinniment (Newcastle University). The program consisted of 21 papers and keynotes by Fred Brooks (University of North Carolina at Chapel Hill), Ted Williams (Morphics Technology), Barbara Chappell (Intel), and Rajiv Joshi (IBM).

Fig. 13. The SkyRide aerial tram at Grouse Mountain taking the Async 2003 attendees to the top of the mountain, the banquet site, and the brown bear enclosure.

The conference included two different outings: one to the Stanley Park and the Vancouver Aquarium, and one to nearby Grouse Mountain ski resort. The Aquarium included a huge variety of fishes and marine life including some very friendly Beluga whales. The banquet was held at the top of Grouse Mountain, accessed by an aerial tram ride (Figure 13). The attractions at the top of the mountain included a large bear enclosure where a friendly (?) pair of grizzly bears could be seen frolicking.

12 Async 2004: Crete, Greece

The 2004 conference was hosted by General Chair Christos Sotiriou (ICS-FORTH) on the beautiful Mediterranean island of Crete from April 19-23, 2004. The Program Co-Chairs were Ran Ginosar (Technion) and Ken Stevens (Intel). The

Fig. 14. Banquet group at Async 2003 including Alexander Yakovlev in the center with hand raised

conference program included 21 papers, and keynotes by Christer Svensson (Linköping University), Martin Jenkner (Infineon Technologies), and Ad Peeters (Handshake Solutions). The conference attendees were confronted with a difficult choice of attending conference sessions, or enjoying the lavish surroundings of the Aldemar Knossos Royal Village Hotel.

The conference included two separate excursions: one to the Palace of Knossos and visit to the FORTH research center, and one to the museum village of Arolithos. The Palace of Knossos was the ceremonial and political centre of the Minoan civilization and culture dating to the bronze age, 1380 – 1100 BCE. In Greek mythology, King Minos dwelt in a palace at Knossos. He had Daedalus construct a labyrinth; a very large maze in which to retain his son, the Minotaur. The Arolithos museum is a site that celebrates the tradition and history of the Cretan way of life. Dinner there was capped off by a performance of traditional Greek music played on the Bouzouki by conference General Chair Christos Sotirou (see Figure 16).

13 Async 2005: New York City, NY, USA

The 11th Async conference was held in the "Big Apple," New York City, NY, USA from March 14-16, 2005. The conference General Chairs were Steven Nowick (Columbia University) and José Tierno (IBM). The Program Chairs were

Fig. 15. Greek traditional dancing at the Async 2004 banquet. Alex Yakovlev is in the far left of this picture. (photo by Doug Edwards)

Fig. 16. Async 2004 general chair Christos Sotiriou (and friends) entertaining the excursion attendees with some excellent Bouzouki playing.

Prabhakar Kudva (IBM) and Rajit Manohar (Cornell University). 20 papers were presented along with keynotes from Bob Colwell (R. E. Colwell and Associates), Ivan Sutherland (Sun Microsystems) and Robert Drost (Sun Microsystems), with an additional invited tutorial by Phil Restle (IBM Research) and Ken Shepard (Columbia University).

Fig. 17. Banquet group from the Async 2005 Manhattan circle-cruise including on the left, Keith Heron, and on the right, David Kinniment, Jo Ebergen, and Gaurav Gulati.

The banquet/outing for the 2005 conference was held on a boat that encircled the island of Manhattan during dinner. Attendees were treated to spectacular views of the city, the statue of liberty, Brooklyn Bridge, and other New York sites. One "secret" about the cover of the Async 2005 proceedings is that if you look in the lower right of the cover you can see two birds flying away from the city. These birds represent the twin towers of the World Trade Center that had come down in terrorist attacks in 2001.

14 Async 2006: Grenoble, France

The conference returned to Europe in 2006 being held in Grenoble, France and hosted by General Chair Marc Renaudin (Institut Polytechnique de Grenoble). The Program Co-Chairs were Alex Yakovlev (University of Newcastle upon Tyne), and Jens Sparsø (Technical University of Denmark). The conference program included 19 papers and keynotes by Nobuo Karaki (Seiko Epson Corp.),

Jean-Pierre Schoellkopf (STMicroelectronics), and Ferdinand Peper (National Institute of Information and Communications Technology, Japan). The official conference social outing was a banquet at a local restaurant Le Chateau de la Baume. In addition to the excellent food, one notable feature of the banquet location was a virtual-reality golf simulator in the adjoining hall. This simulator let attendees hit real golf balls with real golf clubs into a net and the tracking system would tell you how far and in what direction the shot went, and show the next picture from that location. Great fun was had by the participants, many of whom had never before swung a golf club.

Fig. 18. The intrepid Async ski adventurers from Async 2006 at the peak of the Aiguille du Midi ready to descend in the the Vallée Blanche.

The after-conference outing was perhaps the most memorable of any Async outing (at least from my perspective). A group of intrepid skiers embarked on a guided descent of the famous Vallée Blanche. This famous ski adventure involves a tram ride to the top of the Aiguille du Midi - a 3,842 m / 12,605 ft peak in the Mont Blanc massif within the French Alps. From there skiers descend through the Vallée Blanche - a 20 km long, unmarked off-piste ski route which begins very steeply from the Aiguille du Midi station and continues across crevassed glaciated terrain. Figure 18 shows the Async group at the peak ready to start out. The view from that peak includes alps in France, Switzerland, and Italy. The lunch hut halfway down the Valléee Blanche (Figure 19) included superb views of the glacier that the group had just descended, and the pathway to the

Mer du Glace, the largest glacier in France, 7km long and 200m deep and one of the biggest attractions in the Chamonix Valley.

Fig. 19. Lunch halfway down in the Vallée Blanche at Async 2006. From left: Peter Beerel (note the 2001 Async scarf, and Async 2006 hat), John Bainbridge, Alex Yakovlev, and Keith Heron

15 Async 2007: Berkeley, CA, USA

Async 2007 returned to North America to be held in Berkeley, California. The General Chairs were Peter Beerel (University of Southern California) and Marly Roncken (Intel). The Program Co-Chairs were Mark Greenstreet (University of British Columbia) and Montek Singh (University of North Carolina at Chapel Hill). The program included 18 papers and keynotes from James T. Kajiya (Microsoft Research), Carlo H. Sequin (University of California, Berkeley), Steven Jacobsen (Sarcos Inc.), and Kevin Nowka (IBM). The talk by Carlo Sequin was especially appreciated by attendees as he talked about "Thinking Outside the Box in Geometry and Art" including many examples of how mathematical functions can be the basis for sculpture and how the then-new capabilities of 3D printing could be used to teach 3D geometry (see Figure 20).

The conference banquet was held in the Steinhart Aquarium, echoing a previous banquet from Vancouver also held in an aquarium. The post-confernece

Fig. 20. A crowd at Async 2007 gathers around keynote speaker Carlos Sequin after his talk about geometric modeling for 3D printing.

outing was to the nearby city of San Fransisco featuring transportation on the famous cable cars. The tour started from the cable car turntable at Powell/Market Street in San Fransisco. Turntables are the endpoints of a cable car line. After the cable car has arrived, the passengers get off, and then the cable car is pushed onto the turntable and turned around 180 degrees by human power.

16 Async 2008: Newcastle, UK

The 14th conference returned to the U.K., this time being held in Newcastle and hosted by General Chair Alex Yakovlev (University of Newcastle upon Tyne). Conference Program Co-Chairs were Jordi Cortadella (Universitat Politcnica de Catalunya) and Alexander Taubin (Boston University). One notable feature of the 2008 conference is that for the first time Async was co-located with the emerging Networks on Chip Symposium (NoCS) hosted by Co-General Chairs Alex Yakovlev and John Bainbridge (Silistix). These two conferences were truly co-located with Async and NoCS sessions intermingled during each day of the combined conferences (April 7-11, 2008). The Async program consisted of 15 papers invited tutorials by David Kinniment (Newcastle University), Sachin S. Sapatnekar (University of Minnesota), and Mike Kishinevsky (Intel), and keynotes (shared by Async and NoCS) by Ad Peeters (Handshake Solutions), Arjan Bink (Handshake Solutions), David May (University of Bristol), Asen Asenov (University of Glasgow), Ian H. White (University of Cambridge), and Richard V. Penty (University of Cambridge).

Fig. 21. Ivan Sutherland "carries coal to Newcastle" at Async 2008 presenting a gift of coal to conference general chair Alex Yakovlev

One notable event at the conference was Ivan Sutherland "carrying coal to Newcastle" and presenting it to conference General Chair Alex Yakovlev (Figure 21). This phrase is an idiom describing a foolhardy or pointless action. It refers to the fact that historically the economy of Newcastle upon Tyne was heavily dependent on the distribution and sale of coal and therefore any attempt to bring coal to Newcastle from elsewhere for profit would be doomed to failure.

The major social event at the 2008 conference was an excursion to the open-air museum of Beamish. This museum preserves an example of everyday life in urban and rural North East England at the height of the industrial revolution in the early 20th century. It includes a mixture of translocated, original and replica buildings; a huge collection of artifacts, working vehicles and equipment; as well as livestock and costumed interpreters. Async attendee Simon Moore was intrigued by one of the historic coal cars advertising "S. Moore & Co" as the proprietors (Figure 22).

17 Async 2009: Chapel Hill, NC, USA

In 2009, from May 17-20, the conference was held in North Carolina in the college town of Chapel Hill. The General Chair was Montek Singh (University of North Carolina, Chapel Hill). The Program Co-Chairs were Ran Ginosar (Technion) and Luciano Lavagno (Politecnico di Torino). The program featured 21 papers and keynotes by David Tennenhouse (New Venture Partners), Bill

Fig. 22. Simon Moore finds a long-lost relative in the coal business in the Beamish open air museum during the Async 2008 excursion.

Dally (NVIDIA Research), and Chuck Seitz (Myricom, Inc.). The first social event was, unfortunately, rained out. The plan was for the conference attendees to attend a baseball game of the local minor-league baseball team the Durham Bulls. Unfortunately, that game, and the social event, had to be cancelled.

The second outing was much more successful - a trip to Fearrington Village - a mixed-use community located on farmland dating back to the 18th century in Pittsboro, North Carolina. Started in 1974, the community has grown to include over 1800 residents, an award-winning country inn and restaurant (The Fearrington House), and a variety of shops. The conference banquet was held in the Fearrington House and included entertainment by a local Bluegrass music trio. It turns out that Async Steering Committee Chair (at the time) Erik Brunvand plays in a Bluegrass band in Salt Lake City, Utah, and convinced the band to let him join the group on upright bass for two songs (see Figure 24). Along with the band, Prof. Brunvand sang two songs - "Walking the Dog," and "On and On" - Bluegrass songs that both Brunvand and the band knew. The conference attendees were somewhat astonished to see this impromptu performance!

18 Async 2010: Grenoble, France

In 2010 (May 3rd through 6th) the conference returned again to Grenoble, France. This time the General Co-Chairs were Pascal Vivet (Cae-Leti) and Marc Renaudin (Tiempo). The Program Co-Chairs were Alex Yakovlev (University of Newcastle upon Tyne) and Ken Stevens (University of Utah). The conference

Fig. 23. Async 2009 attendees relaxing before the banquet at Fearrington Village. On the right, Alex Yakovlev and Chuck Seitz. On the left (from the left) Hao Zheng, Ian Jones, Steven Nowick, and Jens Sparsø.

Fig. 24. Async 2009 attendee (and Async Conference Steering Committee Chair at the time) Erik Brunvand joins the Bluegrass band after the banquet at Fearrington Village. He played two songs with the band: "Walking the Dog," and "On and On." This image is a capture from a video of the performance (thus the blur). (videographer unknown)

was once again co-located with NoCS. The Async 2010 program featured 17 papers and there were three (shared) keynotes: Mohamad Sawan (University of Montreal), Keren Bergman (Columbia University), and Alessandro Cremonesi (STMicroelectronics).

Fig. 25. Asynchronauts at the Async 2010 conference in Grenoble. From left: Ran Ginosar, Erik Brunvand, Ken Stevens, Alex Yakovlev, and Pascal Vivet.

The main social event at the conference itself was dinner at La Bastille - a small fortified mountain located at the crossroad of three valleys. The route we took to the restaurant was the famous "bubbles" of Grenoble - the connected set of five spherical gondolas whisking conference attendees from the center of town to the top of the mountain. Because the conference was held in May, there was no skiing to be had this time. Instead a post-conference sightseeing excursion was planned including the Chartreuse monastery and the Voiron Cave where the monks produce traditional liqueurs.

19 Async 2011: Ithaca, NY, USA

In 2011 the 17th Async conference returned to the state of New York in the USA, but this time was held at Cornell University in Ithaca. the General Chair was Erik Brunvand (University of Utah) and Program Co-Chairs were John Bainbridge (Silistix) and Ian Jones (Oracle Labs). The conference program consisted of

11 accepted papers, with keynotes from David Albonesi (Cornell University), and Yannis Tsividis (Columbia University). The program was filled out with three invited industrial papers that addressed current industrial approaches to leveraging asynchrony.

Fig. 26. Async 2011 conference attendees at the conference banquet. From left: Alex Yakovlev, Jens Sparsø, Ivan Sutherland and Peter Beerel (facing away).

The main social event at the conference (April 27-29) was dinner at a local winery. With good food and good wine, the conference attendees were put into an excellent mood with an evening full of socializing.

20 Async 2012: Lyngby, Denmark

In 2012 General Chair Jens Sparsø (Technical University of Denmark) hosted the conference in Lyngby, Denmark. Program Co-Chairs were Pascal Vivet (Cea-Leti) and Montek Singh (University of North Carolina at Chapel Hill). The conference program included 18 regular papers, and keynotes by Kwabena Boahen (Stanford University), Steve Furber (Univrsity of Manchester), and Mogens Balsby (Oticon). The conference was co-located (for a third time) with the NoCS conference - this time having the two conferences run back to back with Async on May 7-9 and NoCS from May 9-11. Tutorials were also held on the day before the main conference and were presented by Eslam Yahya (American University in Cairo), Laurent Fesquet (TIMA), and Marc Renaudin (Tiempo).

Fig. 27. Async conference regular Montek Singh makes sure that no wine is left over after the Async 2011 banquet ...

The conference outing/banquet involved a boat tour of Copenhagen followed by dinner. With drinks flowing on the boat before dinner, attendees arrived in a good mood for dinner. There were also opportunities to explore beautiful Copenhagen as the hotels were located in Copenhagen with a short bus ride in the morning to arrive at the conference site in Lyngby.

21 Async 2013: Santa Monica, CA, USA

2013 brought the conference back to California in the USA - this time in Santa Monica in the Los Angeles area. Santa Monica is on the Pacific and enjoys wonderful weather, and seaside activities including a famous pier with amusements. The conference was hosted by General Chair Peter Beerel (University of Southern California) from May 19-22. The Program Co-Chairs were Tomohiro Yoneda (National Institute of Informatics, Japan) and Ran Ginosar (Technion). The program consisted of 22 papers and keynotes by Vivek De (Intel), and Jeanne Trinko Mechler (IBM). The conference also featured talks by three representatives from asynchronous-related startup companies on the state of their chips: Michel Lawrence from Octasic Inc, Richard Terrill from Wave Semiconductor, and Chuck Moore from Green Arrays, Inc.

The banquet was in a restaurant on the beach and featured a talk by Erik Brunvand on the folklore of “hacking” - specifically featuring the on-line legend

Fig. 28. A portrait of the Async conference steering committee taken at the Async 2012 conference in Lyngby, Denmark. From left - Front row: Ian Jones, Steven Nowick, Jens Sparsø, Marly Roncken, and Rajit Manohar. Second row: Montek Singh, Mark Greenstreet, Andrew Lines, John Banbridge, and Peter Beerel. Third row: Pascal Vivet, Alex Yakovlev, Tomohiro Yoneda, Erik Brunvand, and Mark Renaudin. (photo by Peter Beerel)

Fig. 29. Async 2012 attendees enjoy the boat ride around Copenhagen. From left: Alex Yakovlev, Montek Singh, and Ivan Sutherland.

of "Mel, a real programmer." This is a story that has been circulated on the network starting with the USENET in the 1980's about what it means to be a "real programmer" and features a hero named Mel who played some amazing tricks on a drum-memory based machine called the LGP-30 in the early days of computers.

Fig. 30. Conference attendees at Async 2013 in Santa Monica, Alex Yakovlev and Ran Ginosar, enjoy (?) a shot of fresh squeezed wheat grass - a quintessential California treat. (photo by Ran Ginosar)

22 Async 2014: Potsdam, Germany

Milos Krstic (IHP) and Eckhard Grass (IHP and Humboldt University) hosted the 20th Async conference as Co-General Chairs in Potsdam Germany from May 12-14, 2014. The Program Co-Chairs were Marly Roncken (Portland State University) and Andreas Steininger (Vienna University of Technology). The program consisted of 12 papers and keynotes from Jan Rabey (University of California, Berkeley), Joseph Sylvester Chang (Nanyang Technology University, Singapore), and Paul Mitcheson, (Imperial College, London). In addition to regular papers and keynotes, the conference included industry reports about the latest impacts of asynchronous design concepts in industrial products and prototypes, and a "Fresh Ideas" session to provide a forum for controversial statements and unconventional ideas that are not yet fully explored.

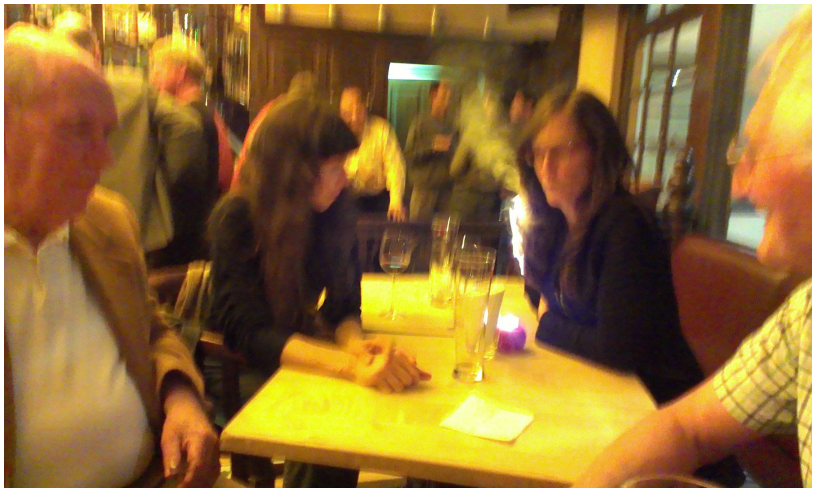


Fig. 31. Excitement at the pre-conference reception at Async 2014 in Potsdam, Germany. Sandy Brunvand's hair catches fire in the bar while posing for pictures with (from left) Ivan Sutherland, Marly Roncken, Sandy Brunvand, and Graham Birtwistle. Photo is a still from a video (thus the blur).

Fig. 32. Async 2014 conference attendees (from left): Alex Yakovlev, Marly Roncken, Marios Elia, and Luciano Lavagno.

The conference outing/banquet started with a boat trip covering seven lakes surrounding Potsdam with a view on historical and natural attractions. This trip Included a buffet dinner with local German culinary specialties, on the boat. At the dinner, Erik Brunvand gave a presentation on "Twenty Years of Async" which provided a history of, and reminiscences about, 20 years of the IEEE Symposium on Asynchronous Circuits and Systems - and was the impetus for this very article.

23 Async 2015: Silicon Valley, CA, USA

In some ways it is surprising that it took 21 iterations of the conference before it found its way to the heart of Silicon Valley. For the 21st conference, General Chair Ian Jones (Oracle Labs) organized the conference in Silicon Valley - Mountain View, California to be precise. Program Co-Chairs were Jens Sparsø (Technical University of Denmark) and Eslam Yahya (Benha University, Egypt). The conference program consisted of 18 papers, two industrial short papers, nine "Fresh Ideas" papers. There were also three keynotes by Bob Iannucci (Carnegie Mellon University, Silicon Valley), Paul Cunningham and Steev Wilcox (Cadence), and Ron Ho (Altera).

Fig. 33. The venue at Async 2015 in Silicon Valley was the Portuguese Cultural Center in Mountain View, California - a great old building with oak walls and benches along the sides.

The conference was held from May 4-6 near downtown Mountain View in the S.F.V Lodge. This is a Portuguese Heritage Center with a lovely main auditorium featuring old oak floors, walls, and benches (see Figure 33). Afternoon treats were provided by an old-fashioned ice-cream truck (see Figure 34). The conference banquet was at a restaurant in downtown Mountain View - a surprisingly quaint downtown area in the middle of bustling Silicon Valley.

24 Async 2016: Porto Alegre, Brazil

In 2016 the conference found its way for the first time to South America - hosted by General Chair Ney L. V. Calazans (PUCRS, Brazil) in Porto Alegre, Brazil

Fig. 34. Async 2015 conference attendees Montek Singh and Erik Brunvand enjoy an ice cream treat from the afternoon break at the conference.

from May 8-11. The Program Chairs were Peter Beerel (University of Southern California) and Julian J. H. Pontes (ARM). The program consisted of 12 regular track papers, four industrial track papers, and eight "Fresh Ideas" papers. The keynote speakers were Patrick Groeneveld (Synopsys), and Paulo A. dal Fabro (Chipus Microelectronics).

Sadly, this is the first Async conference that I personally was not able to attend. Up until this point there were only two researchers who had attended **every** Async conference: Peter Beerel and Erik Brunvand. With the 2016 conference that list has narrowed to only one stalwart conference attendee who has been at every single conference. Peter Beerel is now the sole member of that club.

25 Conclusions

When I, along with colleagues, started this conference series in 1994 (or 1993 if you count the HICSS special session) I had no way of knowing whether it would be a long-lasting conference series, or run its course in a few years. It is immensely gratifying that the conference is, if not growing, at least still vital and healthy some 23 years later. The conference series has had a wonderful core of researchers, some of whom have been involved from the very start. If we look at the Organizing and Program Committees from the very first conference in 1994, we can see the following names still on the conference committee in

Fig. 35. Web splash screen for the 2016 Async conference in Porto Alegre, Brazil

Fig. 36. Erik Brunvand modeling some Async conference swag from over the years. Included are a scarf from Async 2001 (Salt Lake City), an umbrella from Async 2002 (Manchester), an insulated cup from Async 2005 (New York City), a hat from Async 2006 (Grenoble), and a rain jacket from Async 2008 (Newcastle).

some capacity in 2016: Erik Brunvand, Luciano Lavagno, Alain Martin, Steven Nowick, Jens Sparsø, and Alex Yakovlev.

The conference has also attracted a growing corps of younger researchers interested in the field, and who have made good connections at the conference, and found mentors and collaborators there for their research.

Strong research collaborations and strong friendships have been forged over the many years of this conference. It truly seems like one of the friendliest and congenial of the research conferences of which I am familiar. Through the Async conference series we have developed and maintained a community of Asynchronauts that has remained close throughout the years. Perhaps there is no better legacy for a conference series than that.

Acknowledgments. The IEEE Symposium on Asynchronous Circuits and Systems would not be the congenial conference series that it is without the support of a great many wonderful researchers, academics, students, and supporters. Photographs not otherwise credited are by Erik Brunvand (or Erik Brunvand's camera).

References

1. Beerel, P., Meng, T.Y.: Automatic gate-level synthesis of speed-independent circuits. In: Proc. International Conf. Computer-Aided Design (ICCAD). pp. 581–587. IEEE Computer Society Press (Nov 1992), `http://jungfrau.usc.edu/pub/iccad92.ps`
2. Beerel, P., Meng, T.: Semi-modularity and self-diagnostic asynchronous control circuits. In: Séquin, C.H. (ed.) Advanced Research in VLSI. pp. 103–117. MIT Press (Mar 1991)
3. Beerel, P.A., Meng, T.H.Y.: Testability of asynchronous self-timed control circuits with delay assumptions. In: Proc. ACM/IEEE Design Automation Conference. pp. 446–451. IEEE Computer Society Press (Jun 1991)
4. Berkel, C.H.K.v., Niessen, C., Rem, M., Saeijs, R.W.J.J.: VLSI programming and silicon compilation. In: Proc. International Conf. Computer Design (ICCD). pp. 150–166. IEEE Computer Society Press, Rye Brook, New York (1988)
5. Berkel, C.H.K.v., Saeijs, R.W.J.J.: Compilation of communicating processes into delay-insensitive circuits. In: Proc. International Conf. Computer Design (ICCD). pp. 157–162. IEEE Computer Society Press (1988)
6. Berkel, K.v., Kessels, J., Roncken, M., Saeijs, R., Schalij, F.: The VLSI-programming language Tangram and its translation into handshake circuits. In: Proc. European Conference on Design Automation (EDAC). pp. 384–389 (1991)
7. Brunvand, E.: Translating Concurrent Communicating Programs into Asynchronous Circuits. Ph.D. thesis, Carnegie Mellon University (1991)
8. Brunvand, E.: Designing self-timed systems using concurrent programs. Journal of VLSI Signal Processing 7(1/2), 47–59 (Feb 1994)
9. Brunvand, E.: Low latency self-timed flow-through FIFOs. In: Dally, W.J., Poulton, J.W., Ishii, A.T. (eds.) Advanced Research in VLSI. pp. 76–90. IEEE Computer Society Press (1995)

10. Brunvand, E., Sproull, R.F.: Translating concurrent programs into delay-insensitive circuits. In: Proc. International Conf. Computer-Aided Design (ICCAD). pp. 262–265. IEEE Computer Society Press (Nov 1989)
11. Burgon, J.W.: Petra, a Poem: To Which a Few Short Poems Are Now Added. Oxford: F. MacPherson, 2 edn. (1846)
12. Chaney, T.J., Molnar, C.E.: Anomalous behavior of synchronizer and arbiter circuits. IEEE Transactions on Computers C-22(4), 421–422 (Apr 1973)
13. Clark, W.A.: Macromodular computer systems. In: AFIPS Conference Proceedings: 1967 Spring Joint Computer Conference. vol. 30, pp. 335–336. Academic Press, Atlantic City, NJ (1967)
14. Clark, W.A., Molnar, C.E.: Macromodular computer systems. In: Stacy, R.W., Waxman, B.D. (eds.) Computers in Biomedical Research, vol. IV, chap. 3, pp. 45–85. Academic Press (1974)
15. Dill, D.L., Nowick, S.M., Sproull, R.F.: Automatic verification of speed-independent circuits with Petri net specifications. In: Proc. International Conf. Computer Design (ICCD). pp. 212–216. IEEE Computer Society Press (1989)
16. Frank, E.H., Sproull, R.F.: A self-timed static RAM. In: Bryant, R. (ed.) Proceedings of Third Caltech Conference on VLSI. pp. 275–285. Computer Science Press (1983)
17. Hayes, A.B.: Self-timed IC design with PPL's. In: Bryant, R. (ed.) Proceedings of Third Caltech Conference on VLSI. pp. 257–274. Computer Science Press (1983)
18. Kondratyev, A., Rosenblum, L., Yakovlev, A.: Signal graphs: A model for designing concurrent logic. In: Briggs, F.A. (ed.) Proc. International Conference on Parallel Processing. vol. 1, pp. 51–54 (1988)
19. Martin, A.J.: The design of a self-timed circuit for distributed mutual exclusion. In: Fuchs, H. (ed.) Proceedings of the 1985 Chapel Hill Conference on VLSI. pp. 245–260. Computer Science Press (1985)
20. Martin, A.J.: Compiling communicating processes into delay-insensitive VLSI circuits. Distributed Computing 1(4), 226–234 (1986)
21. Martin, A.J.: A synthesis method for self-timed VLSI circuits. In: Proc. International Conf. Computer Design (ICCD). pp. 224–229. IEEE Computer Society Press, Rye Brook, NY (1987)
22. Meng, T.H.Y.: Asynchronous Design for Digital Signal Processing Architectures. Ph.D. thesis, UC Berkeley (1988)
23. Meng, T.H.Y., Brodersen, R.W., Messerschmitt, D.G.: A synthesis method for self-timed VLSI circuits. In: Proc. International Conf. Computer-Aided Design (ICCAD). pp. 514–517 (1987)
24. Meng, T.H.Y., Brodersen, R.W., Messerschmitt, D.G.: Automatic synthesis of asynchronous circuits from high-level specifications. IEEE Transactions on Computer-Aided Design 8(11), 1185–1205 (Nov 1989)
25. Molnar, C.E., Fang, T.P.: Synthesis of reliable speed-independent circuit modules: I. general method for specification of module-environment interaction and derivation of a circuit realization. Technical Memorandum 297, Computer Systems Laboratory, Institute for Biomedical Computing, Washington Univ., St. Louis, MO (1983)
26. Myers, C.J., Rokicki, T.G., Meng, T.H.Y.: Automatic synthesis and verification of gate-level timed circuits. Tech. Rep. CSL-TR-94-652, Stanford University (Jan 1995)
27. Myers, C., Meng, T.H.Y.: Synthesis of timed asynchronous circuits. In: Proc. International Conf. Computer Design (ICCD). pp. 279–282. IEEE Computer Society Press (Oct 1992)

28. Myers, C.J., Meng, T.H.Y.: Synthesis of timed asynchronous circuits. IEEE Transactions on VLSI Systems 1(2), 106–119 (Jun 1993)
29. Nowick, S.M., Dill, D.L.: Practicality of state-machine verification of speed-independent circuits. In: Proc. International Conf. Computer-Aided Design (ICCAD). pp. 266–269. IEEE Computer Society Press (Nov 1989)
30. Nowick, S.M., Dill, D.L.: Automatic synthesis of locally-clocked asynchronous state machines. In: Proc. International Conf. Computer-Aided Design (ICCAD). pp. 318–321. IEEE Computer Society Press (Nov 1991)
31. Nowick, S.M., Yun, K.Y., Dill, D.L.: Practical asynchronous controller design. In: Proc. International Conf. Computer Design (ICCD). pp. 341–345. IEEE Computer Society Press (Oct 1992)
32. Ornstein, S.M., Stucki, M.J., Clark, W.A.: A functional description of macromodules. In: AFIPS Conference Proceedings: 1967 Spring Joint Computer Conference. vol. 30, pp. 337–355. Academic Press, Atlantic City, NJ (1967)
33. Rem, M.: Concurrent computations and VLSI circuits. In: Broy, M. (ed.) Control Flow and Data Flow: Concepts of Distributed Programming, NATO ASI Series, vol. F14, pp. 399–437. Springer-Verlag (1985)
34. Rem, M.: Trace theory and systolic computations. In: de Bakker, J.W., Nijman, A.J., Treleaven, P.C. (eds.) PARLE: Parallel Architectures and Languages Europe, Vol. I. Lecture Notes in Computer Science, vol. 258, pp. 14–33. Springer-Verlag (1987)
35. Rem, M.: The nature of delay-insensitive computing. In: Birtwistle, G. (ed.) IV Higher Order Workshop, Banff 1990. pp. 105–122. Springer-Verlag (1991)
36. Seitz, C.L.: Asynchronous machines exhibiting concurrency (1970), record of the Project MAC Concurrent Parallel Computation
37. Seitz, C.L.: Self-timed VLSI systems. In: Seitz, C.L. (ed.) Proceedings of the 1st Caltech Conference on Very Large Scale Integration. pp. 345–355. Caltech C.S. Dept., Pasadena, CA (Jan 1979)
38. Seitz, C.L.: Ideas about arbiters. Lambda 1(1, First Quarter), 10–14 (1980)
39. Seitz, C.L.: System timing. In: Mead, C.A., Conway, L.A. (eds.) Introduction to VLSI Systems, chap. 7. Addison-Wesley (1980)
40. Snepscheut, J.L.A.v.d.: Deriving circuits from programs. In: Bryant, R. (ed.) Proceedings of Third Caltech Conference on VLSI. pp. 241–256. Computer Science Press (1983)
41. Sproull, R.F., Sutherland, I.E.: Asynchronous Systems. Sutherland, Sproull and Associates, Palo Alto (1986), vol. I: Introduction, Vol. II: Logical effort and asynchronous modules, Vol. III: Case studies
42. Stevens, K., Ginosar, R., Rotem, S.: Relative timing. In: Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems. pp. 208–218 (Apr 1999)
43. Stevens, K., Rotem, S., Burns, S.M., Cortadella, J., Ginosar, R., Kishinevsky, M., Roncken, M.: CAD directions for high performance asynchronous circuits. In: Proc. ACM/IEEE Design Automation Conference. pp. 116–121 (1999)
44. Stevens, K.S.: Practical Verification and Synthesis of Low Latency Asynchronous Systems. Ph.D. thesis, Dept. of Computer Science, University of Calgary, Canada (Sep 1994)
45. Stucki, M.J., Cox, J.J.R.: Synchronization strategies. In: Seitz, C.L. (ed.) Proceedings of the First Caltech Conference on Very Large Scale Integration. pp. 375–393 (1979)

46. Stucki, M.J., Ornstein, S.M., Clark, W.A.: Logical design of macromodules. In: AFIPS Conference Proceedings: 1967 Spring Joint Computer Conference. vol. 30, pp. 357–364. Academic Press, Atlantic City, NJ (1967)
47. Sutherland, I.E.: Micropipelines. Communications of the ACM 32(6), 720–738 (Jun 1989)
48. Sutherland, I.E., Molnar, C.E., Sproull, R.F., Mudge, J.C.: The trimosbus. In: Seitz, C.L. (ed.) Proceedings of the First Caltech Conference on Very Large Scale Integration. pp. 395–427 (1979)
49. Unger, S.H.: Asynchronous Sequential Switching Circuits. Wiley-Interscience, John Wiley & Sons, Inc., New York (1969)
50. Unger, S.H.: Asynchronous sequential switching circuits with unrestricted input changes. IEEE Transactions on Computers 20(12), 1437–1444 (Dec 1971)
51. Varshavsky, V.: Collective Behavior of Automata. Nauka, Moscow (1973), (In Russian), German Edition: W. I. Warshawsky, Kollektives Verhalten von Automaten, (Akademie-Verlag, Berlin, 1978)
52. Varshavsky, V.: Aperiodic automata with self-synchronization. In: Discrete System: Proceedings of International symposium IFAC. vol. 1. Riga (1974)
53. Varshavsky, V.I.: Hardware support of parallel asynchronous processes. Research Reports Series A, No. 2, Digital Systems Laboratory, Helsinki Univ. of Technology, Otaniemi, Otakaari 5 A, SF-02150 ESPOO 15, Finland (Sep 1987)
54. Yakovlev, A.: Designing self-timed systems. VLSI Systems Design 6, 70–90 (Sep 1985)
55. Yakovlev, A.V.: On limitations and extensions of STG model for designing asynchronous control circuits. In: Proc. International Conf. Computer Design (ICCD). pp. 396–400. IEEE Computer Society Press (Oct 1992)

Investigating the side-effects of synchronisers on GALS circuits

Frank Burns

Newcastle University, Newcastle Upon Tyne, UK,
frank.burns@newcastle.ac.uk

Abstract. A choice of synchroniser may be crucial for the correct operation of a GALS circuit. GALS NOCs currently require thousands of synchronisers to communicate information accross clock boundaries. Carefully designed synchronisers are capable of mitigating the effects of metastability errors within a chips lifetime. An arbitrary choice of synchroniser, however, may be affected by metastability or timing issues early on in a chips lifetime, resulting in detrimental side-effects and ultimately failure. This, however, is largely dependent on the circuit design. This paper models GALS communication circuits using xMAS models to investigate the potential side-effects of different types of synchroniser on a variety of xMAS circuits. Different xMAS models are analysed to quantify and classify the level of robustness and the exposure to side-effects based on the synchroniser selection.

Keywords: xMAS models, GALS, Synchronisers, Verification

1 Introduction

Whilst there has been a lot of interest in researching new architectures for GALS [1][2][3], there have been few attempts at providing synthesis solutions for GALS communication. Thus, generation of GALS from specifications has been limited to hardware description languages such as Verilog, VHDL, SystemC [4][5] or synchronous programming languages such as C or ESTEREL [6]. Models for communication logic in the past have relied on standard languages, e.g. Verilog, which require a significant amount of "glue logic" to connect communication primitives together. This kind of modelling tends to be unwieldy and non-intuitive. xMAS [7][8][9][10][11] represents a significant improvement in the representation and modelling of communication systems. It provides a set of graphical communication primitives which are more natural and their higher level of abstraction enables them to be easily understood.

Circuit Petri nets [12] provide a natural means for translation of the xMAS equations and they are also well suited to the visualisation of distributed models of local machines in terms of concurrency. For verification they capture a complete knowledge in the unfolding hence providing a represention of the full causality. In [13] basic techniques for GALS synthesis to Circuit Petri nets for xMAS were presented offering some distinct advantages: they are well suited to

the visualisation of distributed models of local machines in terms of concurrency and for verification they capture a complete knowledge in the unfolding hence providing a representation of the full causality. Basic techniques for GALS verification were also presented including unfolding to occurrence nets and deadlock analysis.

In [13] an additional xMAS synchroniser primitive was introduced to provide a synchronisation wrapper for synthesising a range of "glue" solutions e.g. asynchronous, mesochronous, etc. The system is capable of detecting the side-effects of synchronisation problems through unfolding and verification and signalling a potential shutdown. To improve metastability MTBF, designers can take specific measures. For example, they can change the metastability settling time by adding extra register stages to synchronization register chains. The timing slack on each additional register-to-register connection is added to the metastability settling time value. Designers commonly use two registers to synchronize a signal, but some companies recommend using a standard of three registers for better metastability protection. However, adding a register adds an additional latency stage to the synchronization logic, so arises a trade-off between logic and robustness. Also the choice of a specific synchroniser can have a significant impact depending on the particular design.

This paper models GALS communication circuits using xMAS models to investigate the potential side-effects of different types of synchroniser on a variety of xMAS circuits. An arbitrary choice of synchroniser may cause metastability problems and another may not. This, however, is largely dependent on the circuit design. Different xMAS models are analysed to quantify and classify the level of robustness and the exposure to side-effects based on the synchroniser selection.

The main contributions of this work are:

- Analysis of deadlocks in xMAS models due to synchronisers;
- investigaton of the potential side-effects of different types of synchroniser using a variety of xMAS circuits;
- testing the level of robustness of a design based on synchroniser selection.

2 xMAS modelling

2.1 xMAS Primitives

xMAS models are based on a set of communication primitives which have inputs and outputs and which can be glued together according to the equations which define them [7]. There are eight communication primitives altogether and these are depicted in Fig. 1.

The Source and the Sink primitives are used for inputting and outputting information in the form of packets or tokens. These are the ports of the xMAS model which allow the model to be interfaced to its environment. The equations governing the Source and Sink are shown below

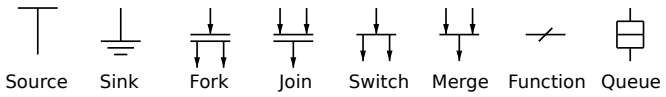


Fig. 1. xMAS primitives.

```
Source:
o.irdy = oracle or pre(o.irdy and not o.trdy)
o.data = e

Sink:
i.trdy = oracle or pre(i.trdy and not i.irdy)
```

The Source is parameterised by a constant expression e : α. Each cycle, it non-deterministically attempts to send a packet e through its output port o : α. In the equations **pre** is the standard synchronous operator that returns the value of its (Boolean) argument in the previous cycle and the value zero in the first cycle. The signals *irdy* and *trdy* stand for initiator ready to send and target ready to receive. The Source and the Sink have a number of different types of operation:

- *eager* - always ready to send or receive packets;
- *dead* - never ready to send or receive packets;
- $non - deterministic$ - the value of the oracle is set randomly.

The Fork and Join primitives are the basic synchronisation primitives. The equations governing the Fork and Join are shown below:

```
Fork:
a.irdy = i.irdy and b.trdy  a.data = f(i.data)
b.irdy = i.irdy and a.trdy  b.data = g(i.data)
i.trdy = a.trdy and b.trdy

Join:
a.trdy = o.trdy and b.irdy
b.trdy = o.trdy and a.irdy
o.irdy = a.irdy and b.irdy
o.data = h(a.data, b.data)
```

A Fork coordinates the input i and outputs a, b so that a transfer only takes place when the input is ready to send and the outputs are ready to receive. A Join primitive operates as the inverse of the fork in which the roles of the *irdy* and *trdy* signals are reversed.

The Switch and Merge primitives are used for routing and selection of packets or tokens through the xMAS circuit. The Switch primitive is governed by the following equations:

```
Switch:
a.irdy = i.irdy and s(i.data)
b.irdy = i.irdy and not s(i.data)
a.data = i.data  b.data = i.data
i.trdy = (a.irdy and a.trdy) or (b.irdy and b.trdy)
```

Informally, the Switch applies s to a packet x at its input, and if $s(x)$ is true, it routes the packet to port a, and otherwise it routes it to port b.

The Merge primitive is used for modelling arbitration by selecting one packet among multiple competing input packets.

```
Merge:
a.trdy = mg and o.trdy and a.irdy
b.trdy = not mg and o.trdy and b.irdy
o.irdy = a.irdy or b.irdy
o.data = a.data if mg and a.irdy
           b.data if not mg and b.irdy
```

A merge has multiple input ports and one output port. Requests for a shared resource are modelled by sending packets to a merge, and a grant is modelled by the selected packet. A local Boolean state variable mg is used to ensure fairness [7].

The Function primitives are used for representing functions. The xMAS equations for the function are shown below.

```
Function:
o.irdy = i.irdy    o.data = f(i.data)
i.trdy = o.trdy
```

In xMAS storage is implemented by queues. The equations for the queue are shown below.

```
Queue:
hd = if (o.irdy and o.trdy) then inc(pre(hd))
      else pre(hd)
tl = if (i.irdy and i.trdy) then inc(pre(tl))
      else pre(tl)
where inc(x) = if x=k-1 then 0 else x+1
       o.irdy = not qempty  i.trdy = not qfull
For j = 0 to k-1
   mem_j = if (i.irdy and i.trdy and j=pre(tl))
           then i.data else pre(mem_j)
```

The queue is characterised by a non-negative integer k that indicates the capacity of the queue. It has one input port i which is connected to the target end of a channel that is used to write data into the queue. Likewise the output of the queue is connected to the initiating end of the channel that reads data out of the queue. The elements in the queue are stored in an array called mem of size k. These are indexed by head (hd) and tail (tl) pointers used for reading and writing.

2.2 GALS Asynchronous Primitive

We have developed a modelling tool in WORKCRAFT [14] for graphical entry of xMAS diagrams. It incorporates an xMAS module for constructing the xMAS models. In addition to the symbols for all the basic primitives a new asynchronous synchronisation primitive has been added to the basic set of primitives shown in Fig. 2. The primitive is used for inserting asynchronous "glue" components in communication channels that cross clock domains. The interface signals are defined using the xMAS format so that it can be interfaced to other xMAS primitives.

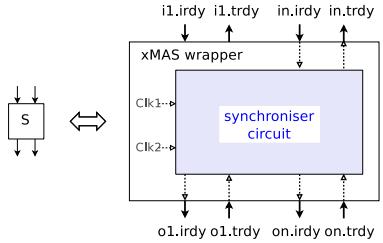


Fig. 2. xMAS synchronisation primitive.

A synchronisation primitive is used for communication between two islands. The synchronisation primitive accepts a variable number of send signals, $i1.irdy$ .. $in.irdy$, from the incoming primitives from one island and returns the required number of receive signals, $i1.trdy$ .. $in.trdy$. Similarly it communicates with the target island by issuing the required number of send signals, $o1.irdy$ .. $on.irdy$ and by accepting the required number of receive signals, $o1.trdy$ .. $on.trdy$. The new asynchronous primitive is generic and incorporates a number of synchronisation schemes. A black box is used to house the specific implementation style used for synchronisation, which is designed to accommodate different GALS implementation styles: asynchronous, mesochronous, pausible clocking, etc.

2.3 Synchroniser modelling

For modelling synchronisers [15] in WORKCRAFT the user connects the communicating GALS modules by means of synchronisation primitives and subsequently from a selection menu chooses the implementation style for each synchroniser. This enables the user to make a decision with regard the internal details based on the GALS style that is required. The GALS style is chosen from a selection of available GALS implementation schemes [16]

The basic synchroniser schemes provided by the tool are as follows:
asynchronous - an implementation based on the use of synchronisers to transfer signals arriving from an outside timing domain to the local timing domain e.g.

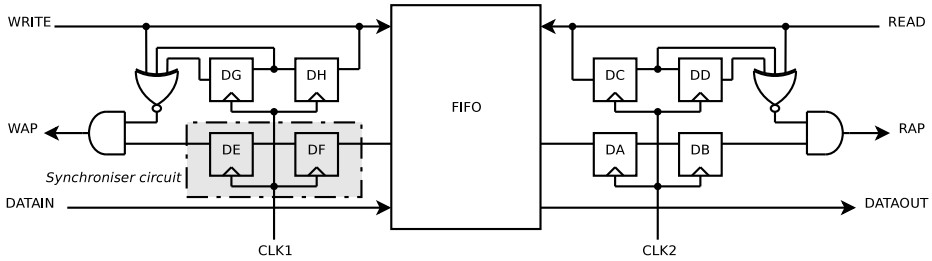


Fig. 3. Asynchronous synchronisation.

two flip-flops to synchronise signal with local clock; *mesochronous* - an implementation in which clocks are derived from the same source and the bounds on the frequencies of communicating blocks are exploited to meet the timing requirements; *pausible* - an implementation based on ring oscillators in which each locally synchronous block generates its own clock with a ring oscillator.

An implementation style that is provided for the asynchronous scheme is shown in Fig. 3. The implementation in Fig. 3 uses a FIFO and synchroniser circuits to transfer signals between the global timing domain and the local timing domain. In this implementation the FIFO buffer handshake signals may be asserted at any time relative to the transmitter or receiver clocks. The implementation uses two flip-flops to synchronise a signal with the local clock. To account for the synchronisers delay, the wait signal generated by the gates prevents the transmitter from sending until the FIFO buffer status following the previous write operation has propagated through the synchroniser.

The synchroniser is used to synchronise the asynchronous communication signal with the local clock. The synchroniser circuit, which is a two-flop synchroniser, is designed to protect the communication signal when it synchronises with the clock from metastability errors. If the synchroniser and clock edges arrive too close together the synchroniser can become metastable with a probability which is related to Mean Time Before Failure (MTBF) [15].

The MTBF for a specific signal transfer, or all the transfers in a design, can be calculated using information about the design and the device characteristics. The MTBF of a synchroniser chain is calculated with the following formula and parameters:

$$\frac{e^{t_{MET}}/C_2}{C_1 \cdot f_{CLK} \cdot f_{DATA}} \tag{1}$$

where the C_1 and C_2 constants depend on the device process and operating conditions; f_{CLK} is the clock frequency of the receiving clock domain; f_{DATA} is the toggling frequency of the input data signal and the t_{MET} parameter is the metastability settling time. For a synchroniser chain t_{MET} is the sum of the output timing slacks for each register in the chain.

The overall design MTBF can be determined by the MTBF of each synchroniser chain in the design. The failure rate for a synchroniser is $1/MTBF$, and the failure rate for the entire design is calculated by adding the failure rates for each synchroniser chain, as follows:

$$Failrate_{design} = \frac{1}{MTBF_{design}} = \sum_{n=1}^{nochains} \frac{1}{MTBF_i} \tag{2}$$

For the two-flop synchroniser a failure could result in the addition of a clock cycle to the latency.

For each implementation style details of the clocking are entered by the user. Inside the tool menus are provided which allow the clocking details to be modified for each synchroniser. Frequencies are set as relative values to reflect changes across module boundaries. The clocking details entered are used later in the verification. Potential synchronisation problems due to metastability are exploited in the unfolding by varying or altering the clock cycles. This is used as a margin of error for the two-flop synchroniser to investigate the effect of a change in the latency.

3 Modelling of deadlocks in Synchronisers

The modelling of the GALS circuits and deadlock analysis is conducted using the Workcraft tool. In [13] we described a methodology and approach for analysing deadlocks in GALS communication circuits using an unfolding algorithm. In [14] the analysis method has been augmented using deadlock relations which are derived from Communication Structured Occurrence Nets $CSONs$ [17].

For unfolding the GALS model is mapped to Structured Occurrence nets and the local modules L_N are mapped to ordinary occurrence nets. The GALS unfolding enables mapping by assigning occurrence nets to divisions corresponding to local module boundaries; occurrence nets are generated automatically for each local module and the individual ONs are subsequently connected using communication channels.

3.1 Deadlock relations

Deadlock relations are derived from the nets. The advantage of deadlock relations is they are more compact and they can be used inside the tool to relay critical information to the user in the form of statements about the type and causality of the blocking i.e. which queue is the source of the blocking for another queue in a particular module. Deadlock relations can be specified either locally or globally.

Deadlock relations can be defined in terms of queue blocking or idleness. A queue which is found to be blocked in local module L_A may cause a queue to be blocked in L_B. Correspondingly a queue which is found to be idle in

local module L_A may cause a queue to be idle in L_B. The following definitions introduce deadlock relations for local queue blocking and local queue idleness.

Definition 1 *A blocking deadlock relation occurs locally between two queues on the same path in module L_A if a queue $q1^{L_A}$ is blocked thereby causing a queue that precedes it $q2^{L_A}$ to be blocked. This relation is expressed as follows $q2^{L_A} \overset{B}{\leftarrow} q1^{L_A}$.*

Definition 2 *An idle deadlock relation occurs locally between two queues on the same path in module L_A if a queue $q1^{L_A}$ is idle thereby causing a queue that follows it $q2^{L_A}$ to be idle. This relation is expressed as follows $q1^{L_A} \overset{I}{\rightarrow} q2^{L_A}$.*

For the GALS models the process can be extended to analyse which queue in a local module causes blocking or idleness in a synchroniser. The following definitions introduce the different types of deadlock relations for synchronisers.

Definition 3 *A blocking deadlock relation occurs between a synchroniser S and queue that precedes it in module L_A connecting the queue if the synchroniser is blocked thereby causing the connecting queue $q1^{L_A}$ to be blocked. This is expressed using the relation $q1^{L_A} \overset{B}{\leftarrow} S1$. The reverse relation of this can be expressed using $S1 \overset{B}{\leftarrow} q1^{L_A}$.*

Definition 4 *An idle deadlock relation occurs between a synchroniser and a queue in module L_B that follows it connecting the synchroniser if the synchroniser is idle thereby causing the connecting queue $q1^{L_B}$ to be idle. This is expressed using the relation $S1 \overset{I}{\rightarrow} q1^{L_B}$. The reverse relation of this can be expressed using $q1^{L_B} \overset{I}{\rightarrow} S1$.*

The above relations can be chained together. The following equations show examples of chained relations. Equation (3) shows a deadlock relation between a synchroniser $S0$ and its two connecting queues $Q1$ and $Q2$ from local modules L_A and L_B. Equation (4) shows an internal local blocking relation between queues $Q2$ and $Q3$ in module L_B, in conjunction with blocking relations between the synchroniser $S0$ and corresponding local connecting queues.

$$q1^{L_A} \overset{I}{\rightarrow} S0 \overset{I}{\rightarrow} q2^{L_B} \tag{3}$$

$$q1^{L_A} \overset{B}{\leftarrow} S0 \overset{B}{\leftarrow} (q2^{L_B} \overset{B}{\leftarrow} q3^{L_B}) \tag{4}$$

The following definitions are used to define deadlock relations for queues which are connected on the same path.

Definition 5 *A bde is a set of queues connected via the same communication path in which contiguous communicating queue pairs exhibit blocking deadlock relations.*

Definition 6 *An ide is a set of queues connected via the same communication path in which contiguous communicating queue pairs exhibit idle deadlock relations.*

Equation (3), above, is an example of an *ide* relation and equation (4) is an example of a *bde* relation.

Using the deadlock relations a relational map is generated to show complete instances of deadlock activity inside the model. This is achieved by deriving all the deadlock relations from the unfolding to analyse the activity across the channel links and internally inside the local modules. This is expressed in terms of sets of blocking *bde* equations and idle *ide* equations. A complete set of *bde* and *ide* equations is generated by the analyser.

Indirect relations can also be formed between *ide* and *bde* providing relational links between blocking and idle paths. Here the queues on an *ide* and *bde* may not be in direct communication with each other but may be influenced by the communication links between. The causality between an *ide* and a *bde* is established by analysing the corresponding cross-communication links via the net. Using this information it is possible to analyse a number of unique solutions and trace the set of the original source(s) of the deadlocks.

Applying the relational model it becomes practicable to query the effects between different queues and synchronisers. The querying process uses transitivity to establish links between specific queues. Transitivity may be applied to equation (3), for example, to produce equation (5), reflecting the relation between $q1^{L_A}$ and $q2^{L_B}$:

$$q1^{L_A} \xrightarrow{I} S0 \cdot S0 \xrightarrow{I} q2^{L_B} \implies q1^{L_A} \xrightarrow{B} q2^{L_B} \tag{5}$$

Hence, it becomes possible using the relational model to query directly point-to-point causality between queues in different modules.

3.2 Modelling of deadlocks due to synchroniser problems

Deadlocks related to the synchroniser can be split into two types: (i) direct i.e. the deadlock is due to a synchroniser handshake failure. This can be caused by an error in the synchroniser or its environment due to handshake problems [18]. (ii) indirect: i.e. timing problems due to the latency. This is a result of setup time and metastability problems which can result in latency mismatch and subsequent functional errors in the adjoining modules.

Direct deadlock The example below is based on a direct deadlock error caused by a synchroniser $S0$. In this example module L_B communicates with two modules L_A and L_C via asynchronous channels. L_B transmits packets to L_A. As a result of a synchronisation handshake error in synchroniser $S0$, $S0$ fails to communicate with L_B causing it to become idle resulting in a shutdown in communication between L_A and L_B. However, due to its design L_B only partially shuts down and still manages to communicate packets with L_C.

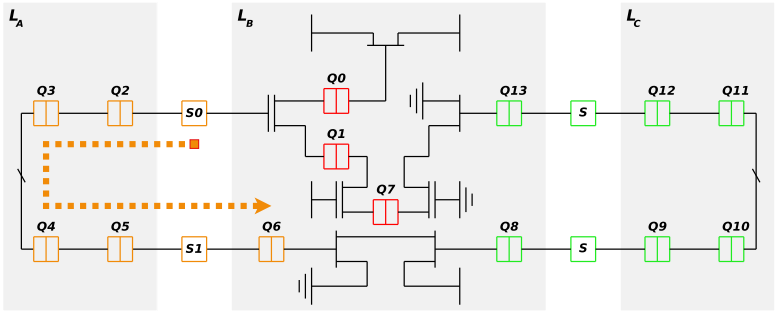


Fig. 4. Direct deadlock example.

The equation showing the synchroniser deadlocks are shown below.

$$S0 \xrightarrow{I} (q2 \xrightarrow{I} q3 \xrightarrow{I} q4 \xrightarrow{I} q5)^{L_A} \tag{6}$$

$$q5^{L_A} \xrightarrow{I} S1 \xrightarrow{I} q6^{L_B} \tag{7}$$

Here an indirect *idle* deadlock occurs in $S1$ due to the chain of *ide* deadlock relations.

Indirect deadlock The example below, in Fig. 5, is based on deadlock due to timing mismatch issues caused by a synchroniser. Module L_A communicates with L_B accross an asynchronous channel. L_B merges its own internal source with the incoming stream from L_A and a switch is used to filter all external packets upwards and all native packets downwards. All sources in the example are eager.

The circuit on the right requires a specific relative timing between the information flows to operate properly. Specifically the feedback from $q8^{L_B}$ and $q9^{L_B}$ are used to limit the upward and downward packet flow so that the upward and downward transfers become balanced. Queue $q4^{L_B}$ represents a common channel. Due to the setup and MTBF time window for the synchroniser being larger than the restricted flow limit will allow, the common channel as a consequence will sequence too many native packets. Thus, when $q9^{L_B}$ is emptied this channel

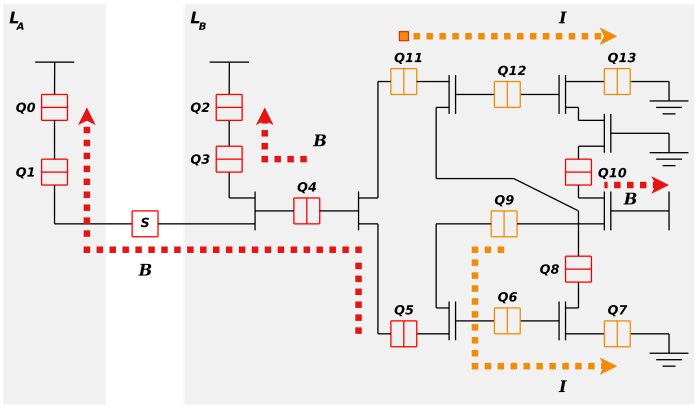


Fig. 5. Synchroniser deadlock example.

becomes blocked. If the synchroniser is removed or replaced by an ordinary queue the balance requirements of the circuit are met so it will operate according to the flow requirements. The deadlock is indirectly caused by the synchroniser due to latency problems with setup and MTBF resulting in downstream functional errors. The equations related to the deadlock are.

$$q9^{L_B} \xrightarrow{I} q6^{L_B} \xrightarrow{I} q7^{L_B} \tag{8}$$

$$(q0^{L_B} \xleftarrow{B} q1^{L_B}) \xleftarrow{B} S \xleftarrow{B} (q4^{L_B} \xleftarrow{B} q5^{L_B}) \tag{9}$$

An indirect relation between (8) and (9) via the join means that when $q9$ is emptied $q5$ becomes blocked.

For the same circuit a mesochronous synchroniser can be substituted in place of the asynchronous synchroniser and the synchroniser will only deadlock with a level of probability which is significantly lower. The level of robustness of the implementation of the mesochronous case can be approximated based on an estimate of the relative timing gap in relation to metastability.

This estimate is measured in terms of the following equation:

$$\sum_{n=1}^{nochains} \frac{1}{k \cdot e^{-(t' \cdot n)/t_i}} \tag{10}$$

where $t' \cdot n$ provides a measure of the relative timing gap.

4 Analysis and experiments

A set of experiments was conducted using a variety of xMAS circuits and different synchronisers. Verification proceeds by searching for direct deadlocks or those

Table 1. xMAS Verification Results [k=2]

Example	i	s	n	asynch	dlk	lrb	time(s)
PC1	4	4	34	asynch	2	0	0.291
PC2	3	1	30	asynch	1	0	0.430
PC3	3	1	30	mesoch	0	0.055	0.341
Agent1	6	2	50	asynch	2	0	0.402
Agent2	6	2	50	mesoch	0	0.028	0.360
Agent3	6	2	52	mesoch	0	1.515	0.368
Mesh1	8	4	104	asynch	4	0	1.550
Mesh2	8	8	104	asynch	0	0.007	1.628
Mesh3	16	12	228	asynch	0	0.004	5.770

due to setup problems which are reported immediately if they are present. In the presence of deadlocks the level of robustness is set to 0. In the absence of deadlocks the level of robustness is measured in terms of a metric based on the metastability gap. The level of robustness of the designs was measured based on its relative level to the circuit and the choice of synchroniser. A trade-off is possible in certain cases based on the choice of synchroniser versus the level of robustness.

To limit the verification effort experiments were conducted using a mixed-mode consisting of eager and non-deterministic. In this mode the sources are varied between eager and non-deterministic. This mode is significant because it is faster than full non-deterministic which in conjunction with a non-deterministic limit generates a more efficient unfolding leading to faster verification in which the analysis can be performed more efficiently. The experiments were conducted using an Intel Core i7 3.4GHz processor.

For the experiments a number of different xMAS circuits were tested. The results of the verification are shown in Table 1. The results are shown in terms of the queue size $k = 2$, the number of sources i, the number of synchronisers s, the number of xMAS primitives n, the type of GALS implementation, the number of synchroniser deadlocks dlk, the level of robustness lrb, and the time in seconds it takes to calculate the results.

The first set of experiments are producer consumer examples. These are basic communication examples using point-to-point communication only. The first example calculates a direct deadlock in 0.291s. The second example $PC2$ uses an asynchronous synchroniser has 1 deadlock due to setup problems and its level of robustness is 0. The third example $PC3$ which uses a mesochronous synchroniser for the same design, is deadlock free, and, therefore, its level of robustness is is estimated. This appears as 0.055 in Table 1.

The next set of experiments, shown in Table 1, are agent examples in which the GALS modules are structurally designed so that varying numbers of communicating agents communicate with each other. The first example uses an asynchronous synchroniser has 2 deadlocks due to setup problems. The second ex-

Table 2. xMAS Verification Results [k=3]

Example	i	s	n	asynch	dlk	lrb	time(s)
PC1	4	4	34	asynch	2	0	0.452
PC2	3	1	30	asynch	1	0	1.135
PC3	3	1	30	mesoch	0	0.055	1.062
Agent1	6	2	50	asynch	2	0	1.165
Agent2	6	2	50	mesoch	0	0.028	1.129
Agent3	6	2	52	mesoch	0	1.515	1.142
Mesh1	8	4	104	asynch	4	0	4.520
Mesh2	8	8	104	asynch	0	0.007	4.692
Mesh3	16	12	228	asynch	0	0.004	18.532

ample *Agent*2 uses a mesochronous synchroniser has 0 deadlocks and its level of robustness is 0.028. The third example *Agent*3 which uses two mesochronous synchronisers is deadlock free but the level of robustness is much higher 1.515 due to the estimate for the metastability gap being larger.

Finally, the examples Mesh1 to Mesh4 are mesh structures comprising more than 100 nodes. These were split into two sizes using more complex structures consisting of many intra-modular and inter-modular loops. The number of synchronisation units was varied for each experiment. These experiments were used to test the scalability of the verification. The results for the experiments show the level of robustness is much lower for an increase in the number of synchronisation units used. For the larger examples it takes significantly longer to test the level of robustness to synchronisation problems.

Table 2 shows results for the same set of experiments using a different queue size [k=3] which shows a comparison of times.

5 Conclusions

We have provided a GALS synthesis and verification environment for xMAS. This has been used for analysing problems caused by synchronisation. It is based on unfolding and deadlock analysis which allows for both checking and visualisation of different types of synchroniser deadlocks. A unique deadlock analysis approach using relations has been described for verifying the examples.

The verification approach is flexible and adaptable to the timing of alternate GALS implementations. Different GALS synchronisers can be selected based on the chosen implementation style. The approach taken enables the investigaton of the potential side-effects of different types of synchroniser using a variety of xMAS circuits. The approach allows for testing the level of robustness of a design based on synchroniser selection.

Acknowledgments

Acknowledgments to Alex Yakovlev et. al.

References

1. Suhaib, S., Mathaikutty, D., Shukla, S.: Dataflow Architectures for GALS. *ACM Journal. Electronic Notes in Theoretical Computer Science (ENTCS)*, Vol. 200, No. 1, 33–50, 2008.
2. Fan, X., Krstic, M., Grass, E., Sanders, B., Heer, C.: Exploring pausible clocking based GALS design for 40-nm system integration. *Proceedings of DATE'2012*, 118–121, 2012.
3. Jungeblut, T., Ax, J., Porrmann, M., Ruckert, U.: A TCM-based architecture for GALS NoCs. *Proceedings of ISCAS'2012*, 2721–2724, 2012.
4. Yakovlev, A., Vivet, P., Renaudin, M.: Advances in Asynchronous logic: from Principles to GALS and NOC, Recent Industry Applications, and Commercial CAD tools. *Proceedings of DATE'2013*, 2013.
5. L. Janin and D. Edwards, "AsipIDE Tutorial - Bringing together GALS design and open-source tools in a hardware-software-FPGA co-simulation flow," *Tutorial at Conference ASYNC-NOCS*, 2010.
6. Koch-Hofer, C., Renaudin, Y., Thonnart, Y., Vivet, P.: ASC, a System C Extension for Modelling Asynchronous Systems, and its Application to an Asynchronous NOC. *Proc. on Networks-on-Chip NOCS'2007*, 295–306, 2007.
7. Chatterjee, S., Kishinevsky, M., Ogras, U.: xMAS: Quick Formal Modelling of Communication Fabrics to Enable Verification. *IEEE Design and Test of Computers*, Vol 29, no. 3, 80–88, 2012.
8. Chatterjee, S., Kishinevsky, M.: Automatic Generation of Inductive Invariants from High-Level Microarchitectural Models of Communication Fabrics. *Proc. of Intl. Conf. on Computer Aided Verificationi, CAV'2010*, 2010.
9. Gotmanov, A., Chatterjee, S., Kishinevsky, M.: Verifying deadlock-freedom of communication fabrics. *Proc. VMCIA*, 214–231, 2012.
10. S. Joosten and J. Schmaltz, "Generation of inductive invariants from register transfer level designs of communication fabrics," *Proc. Formal Methods and Models for Codesign (MEMOCODE) 2013*, pp. 57–64, 2013.
11. S. J. Joosten and J. Schmaltz, "Automatic Extraction of Micro-Architectural Models of Communication Fabrics from Register Transfer Level Designs," *Design and Test Europe (DATE'15)*, Grenoble, France, March, pp. 9–13, 2015.
12. Yakovlev, A., Gomes, L., Lavagno, L.: Hardware Design and Petri Nets. *Springer*, (2000)
13. F. Burns, D. Sokolov and A. Yakovlev, "GALS Synthesis and Verification for xMAS models," *Proceedings of DATE'2015*, pp. 1419–1424, 2015.
14. WORKCRAFT homepage. *http://workcraft.org/*
15. D. Kinniment, "Synchronization and Arbitration in Digital Systems," *Wiley Publishing*, 2008.
16. M. Krstic, M. Grass, E. Gurkaynak, F. and P. Vivet, "Globally Asynchronous, Locally Synchronous Circuits, Overview and Outlook" *Design and Test of Computers, IEEE*, Vol. 24, No. 5, pp. 430–441, 2007.
17. Koutny, M., Randell, B.: Structured Occurrence Nets: A Formalism for Aiding System Failure Prevention and Analysis Techniques. *Proc. ACM Fundamenta Informaticae*, 41–91, 2009.
18. F. Verbeek, S. Joosten and J. Schmaltz, "Formal Deadlock Verification for Click Circuits," *19th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'13)*, Santa Monica, May, pp. 19–22, 2013.

Analytical Derivation of the Reliability Metric for Digital Circuits

A. Bystrov and M. A. Abufalgha

Newcastle University

1 Summary of the method

Two traditional approaches to evaluation of digital circuit reliability are Monte Carlo simulations and physical testing of a prototype, both being quite expensive and unsuitable for circuit optimisation in the course of logic synthesis. Therefore, a new method is proposed, which is based on two levels of characterisation: the platform-level stochastic interference model and the circuit-level model for "translation" of the former model into the reliability metric of the digital circuit. The platform-level interference model is fixed for a design library and environmental conditions. For example, it may include a *probability density function* (PDF) of neutron energy, and a model of the current pulse in the transistor as a function of the neutron energy, transistor size, type, source-drain voltage, temperature, etc. Its purpose is to represent the interference, possibly expressed in non-electrical terms (e.g. particle energy distribution), as electrical effects (e.g. pulses of current having their magnitude, duration and arrival time stochastically described). This is done just once and is universal for every block in a SoC.

The translation model is the core idea of the method. This model converts the stochastic description of the electrical interference, e.g. the current pulse caused by neutron strike, into the probability of error at the circuit output. This is done by finding the critical values for the interference parameter, e.g. the parameters of the above current pulse, beyond which the parameter causes an error, e.g. an incorrect output value written into a flip-flop. The critical values are found by a series of analogue simulation runs on the circuit, but not the Monte Carlo method. Then, in the knowledge of the critical values of the interference parameter, it becomes possible to analytically recalculate the stochastic model of the interference into the probability of an output error or correct operation (reliability).

This method can be combined with the analysis of performance and energy consumption of a circuit, thus contributing to the methodology of energy-modulated computing, whose major problem is provision of reliable operation under randomly modulated, i.e. unreliable, power supply. First results of the proposed method are obtained. They show that a complex tradeoff exists between energy, performance and reliability of digital circuits, and that the traditional dynamic voltage-frequency scaling can be improved by taking the reliability into account.

2 Circuit under test

In this paper only a simple form of a combinational circuit is considered – a long chain of inverters. It is intended to mimic a single path through an arbitrary logic circuit used as a part of a synchronous clocked automaton operating under voltage-frequency scaling. The frequency is chosen as a performance metric. It is determined for each value of the voltage supply Vdd by simulating the circuit and measuring the propagation delay, no margins added. The circuit includes 205 identical inverters implemented with UMC 90nm foundry design kit, all transistors are 80nm in length (standard for this library), pull-down transistor is 400nm, pull-up is 800nm (these values as similar to those used in a commercial standard-cell library), standard threshold voltage, standard use $Vdd = 1V$. Between the inverters there are wires, whose parasitic capacitance we simulate as $2fF$ capacitors (typical capacitance of a short interconnect wire). In our experiments we estimate the reliability of only four inverters in this long chain, as illustrated in Figure 1, then show that the values for all of them are very similar, while a minor difference is observed only in the last stage. Therefore, the reliability of all inverters in the path, except the last one, can be accepted to be the same.

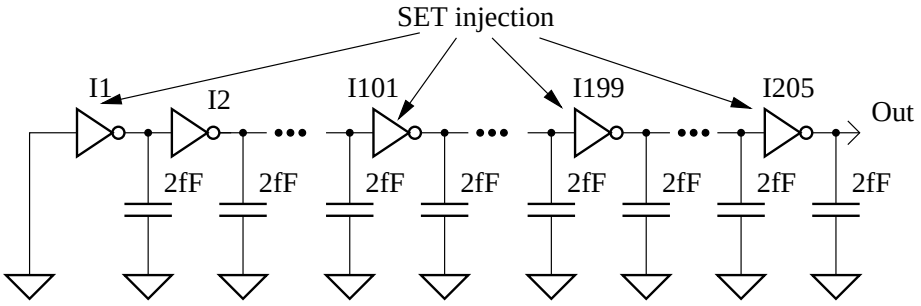


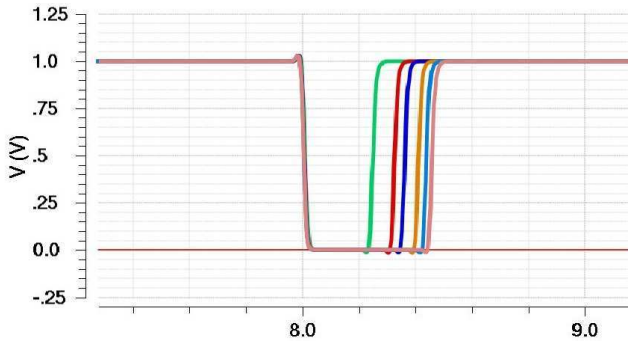
Figure 1: Circuit under test

3 Fault model

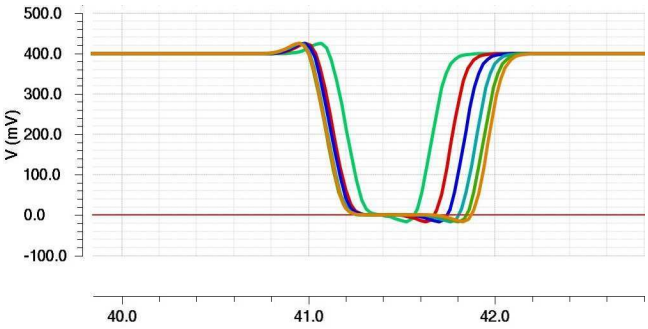
A strike of a neutron is chosen as the cause of faults in our example. The neutron penetrates silicon and may collide with an atom, thus producing secondary charged ions and, eventually, the holes and electrons around the sensitive part of a transistor, known as *error zone* [15]. The holes and the electrons injected in the material around a transistor flow towards the PN junctions, recombine and create a current pulse, which in the circuit of a logic gate, to which the transistor in question belongs, presents itself as a pulse of voltage at the gate output. This is a very crude overview of a complex physical process studied in

[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]. The flow of neutrons can be modelled as a stochastic energy distribution by a classical Maxwell-Boltzmann PDF and the rate [15].

After the model for the primary cause of errors have been specified, one has to "translate" its model into the pulses of voltage on wires. In this work a method described in [16] is used. In this method the effect of a neutron strike is modelled as a dependent current source included into a BSIM4 Spice model of a MOSFET transistor. A result of application of this method is a number of families of waveforms for the voltage at the output of an inverter for a range of Vdd voltages and a range of values of neutron energy. The particle energy is expressed as a metric of *linear energy transfer* (LET) [1,11,12,13,14,15]; this is because we are interested not in the neutrons themselves, but rather in the effect of their interaction with the transistor. The pulses of voltage at the gate outputs caused by neutron strikes are called *single event transients* (SET), because they are temporary logic errors resolving themselves after a short interval of time. The simulated families of SETs for different LET and Vdd is shown in Figure 2.



(a) $Vdd = 1.0V$, LET=5, 15, 25, 50, 75, 100(the widest)



(b) $Vdd = 0.4V$, LET=5, 15, 25, 50, 75, 100(the widest)

Figure 2: Two families of SETs for different LET and Vdd values

Constructing a fault model is an important stage of the reliability estimation method, because this model provides the primary information which is subsequently converted into the probabilities of error or absence of such, i.e. the reliability. As one can see, this model is specific to the technology [15] and the stochastic description of the neutron flux. In the same time, it is not aware of the logic circuit constructed with the gates. Therefore, this is the platform-level stochastic interference model, a characterisation stage performed just once for a given technology library and radiation conditions, and is not repeated for each particular design utilising the technology library. This stage is expensive, because it requires conducting physical experiments in order to determine various parameters involved in modelling the SET as in [15].

4 Analytical calculation of reliability

This section describes the core of the method which does not require Monte Carlo simulations for gaining statistics on the output errors. Instead, the stochastic fault model is converted into the reliability value through the properties of a circuit.

The first objective of this stage is to find whether an SET (e.g. on of those shown in Figure 2) would cause an output error of the whole circuit comprising multiple gates (a long chain of inverters in our example) or not. The second objective is to calculate the probability of error-free operation or reliability.

An output error is defined as a *single event upset* (SEU) [1,2,17], which is an effect of an SET if the latter becomes latched in a flip-flop connected to the output of the combinational circuit with the SET on it. A difficulty here is that not all SETs result in an SEU. Some SETs disappear before the clock signal, or appear too late w.r.t. it. Furthermore, the magnitude of an SET may be below the threshold of the flip-flop sampling the output, its duration may be insufficient or it may disappear while propagating through the path due to individual stages exhibiting *inertial delay* behaviour, and suppressing the short duration pulses.

The first objective is achieved by identifying a vector of parameters of the *interference* (in this experiment it is a SET characterised with two parameters – the LET and arrival time) and simulating the circuit in order to determine the critical values of this vector, which separate the erroneous from error-free behaviour at the output. We repeat this for different Vdd and arrival time values in order to see how reliability changes under voltage-frequency scaling (the clock period is adjusted to the propagation delay under each Vdd value).

In Figure 3 the critical values of the interference vector are displayed for $Vdd = 1V$ and the faulty stage number 101; the clock period defined as a propagation delay without any margins is $4.06ns$. It is easy to adjust the results to any timing margins used in a particular design, but it is not included in this paper. For the other stages in the path the diagrams are very similar, just shifted left for the low stage numbers and right for the high numbers.

The second objective is achieved by using the graph in Figure 3 to calculate the probability P_{err} of the system being in the error zone. For this we use the

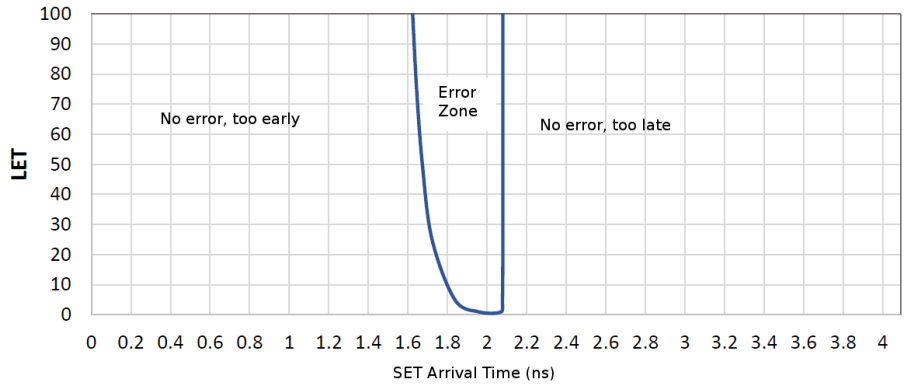


Figure 3: Critical values of the interference vector

PDF function f_x for LET x_{LET} and the PDF function f_t for SET arrival time t_a; the former known from the fault model, the latter having uniform distribution due to asynchronous nature of SET events. P_{err} in (1) is calculated for a single clock cycle.

$$P_{err} = \frac{\iint_{error\ zone} f_x(x_{LET}) \cdot f_t(t_a) \cdot dx_{LET} \cdot dt_a}{\int_{t=0}^{T} \int_{e_{LET}=0}^{\infty} f_x(x_{LET}) \cdot f_t(t_a) \cdot dx_{LET} \cdot dt_a}. \tag{1}$$

The integrals in (1) are computed numerically. Note, the PDF of the arrival time is constant, i.e. $f_t(t_a) = 1/T \cdot r_{SET}$, where T is the clock period, and r_{SET} is a constant representing SET rate. Instead of the infinite integration limit for e_{LET} we choose 100, as the probability of exceeding this limit is negligible [18,19]. The PDF for LET is defined as Maxwell–Boltzmann formula (2).

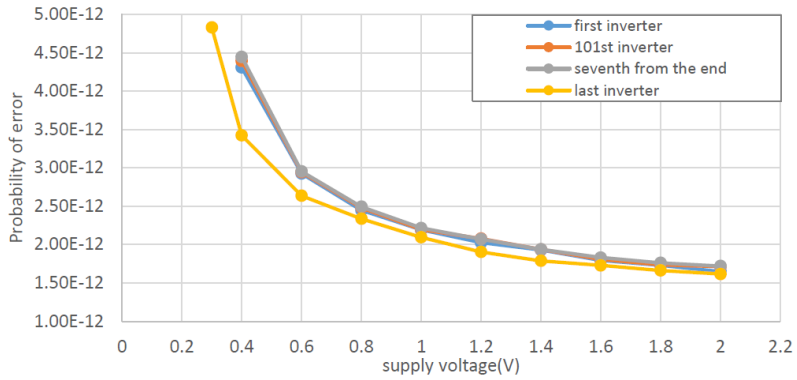
$$f_x = \sqrt{\frac{2}{\pi}} \cdot \frac{x_{LET}^2 e^{-x_{LET}^2/(2a^2)}}{a^3}, \tag{2}$$

with the constant a a calculated as 25.06.

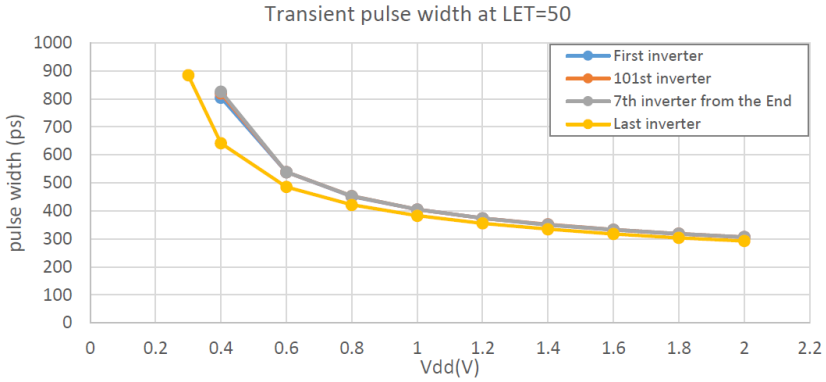
This is for the probability of error when an SET is injected in the stage 101 of the path. The same procedure was repeated for the other stages, and the computed figures were the same apart from the last five stages, where the probability of error was gradually reduced towards the end, and the last stage produced 10%-20% lower error probability (depending on Vdd). For low SET rates it is reasonable to assume that not more than a single SET can take place in the path in any particular clock cycle, which leads to the formula (1) being applicable to the path error, and r_{SET} becomes the SET rate in the path. The reliability can be calculated as absence of error, i.e. $P_{reliability} = 1 - P_{err}$.

5 Results

The above method was applied under a range of Vdd values, the error probabilities were calculated and plotted in Figure 4(a). The SET rate was chosen as $r_{SET} = 20h^{-1}$, i.e. 20 neutrons per hour hitting one of the inverters in the path, which is abnormally high, as such a rate is usually applied to the whole chip rather than a small circuit. It is interesting that the error probability is reduced if SET is injected in the last stage. This is an effect of the SET expanding when propagating along the path. This expansion only happens when SET is long, i.e. the LET causing it is high. There is no path attached to the last stage, hence no expansion, and lower error probability as a result. It is seen in Figure 4(b), which plots the transient pulse duration for a range of Vdd values and point of SET injection.



(a) Error probability



(b) Transient pulse duration

Figure 4: Error probability and transient pulse duration vs. Vdd

Note that in these diagrams the probability of error is calculated per a single clock cycle, rather than per second of operation. This metric is relevant to completion of fixed computational tasks. If this metric is changed to the probability of error per unit of time, then the figures for low voltages will look by far better – it is a common oversight in low-power design.

A 3D diagram in Figure 5 depicts a three way trade-off between energy, reliability and performance, which is one of main results in this paper. It shows that in the low-energy corner both the reliability and performance drop rapidly, which results in a recommendation to avoid this corner. A similar diagram can be generated for any design and without lengthy Monte Carlo simulations and used for selecting an operating point.

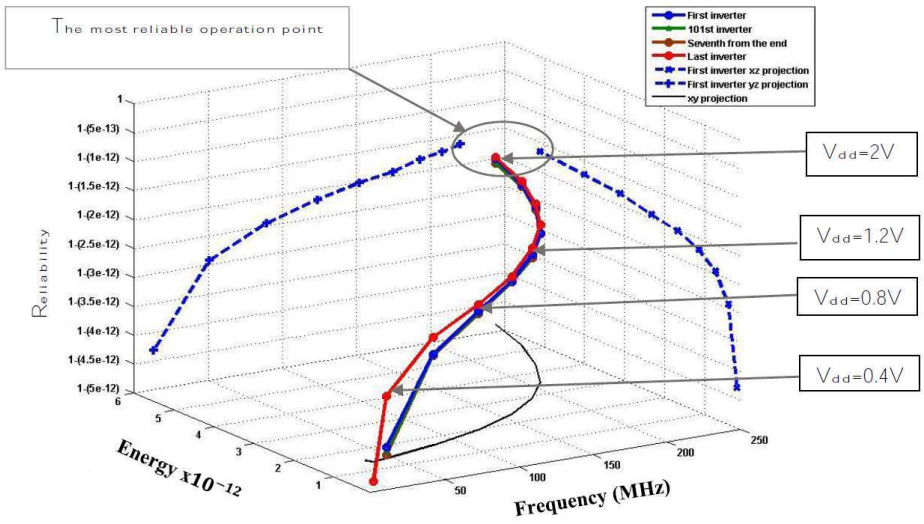


Figure 5: Energy-reliability-performance tradeoff

6 Conclusions

Two main achievements reported in this paper are a new method of analytical derivation of reliability metric for digital circuits, and a tree-way energy-reliability-performance tradeoff demonstrated by by the above method.

The reliability metric is derived without extremely expensive Monte Carlo simulations or physical experiments, which makes its inclusion into ECAD logic synthesis tools possible. This method is possibly a future enabler for achieving the reliability closure on a system at an early design stage, similar to how the timing closure is addressed.

The method includes two stages. At the first stage the technology library is characterised under a chosen interference model, e.g. a neutron flux with a particular energy distribution, and then "translated" into the electrical domain as an SET model. This is done just once and not repeated for each circuit in the project. At the second stage critical values for the vector of interference parameters are derived for a particular circuit under test by a limited number of simulations. The critical values are the border between the erroneous and error-free operation. Then, the probability of error or absence of it, i.e. the reliability, is calculated.

The explored three-way tradeoff is extending the traditional static or dynamic voltage-frequency scaling concepts by adding the reliability metric. It will help to select the operating point for circuits. It is also an enabler for a new generation of power management which controls the reliability dynamically – power or energy reliability management, PRM or ERM.

References

1. D. A. Black, W. H. Robinson, I. Z. Wilcox, D. B. Limbrick, and J. D. Black, "Modeling of Single Event Transients With Dual Double-Exponential Current Sources: Implications for Logic Cell Characterization," IEEE Transactions on Nuclear Science, vol. 62, pp. 1540-1549, 2015.
2. S. Sayil, A. Shah, M. Zaman, and M. Islam, "Soft Error Mitigation using Transmission Gate with varying Gate and Body Bias," IEEE Design & Test, vol. PP, pp. 1-1, 2015.
3. V. Ferlet-Cavrois, L. W. Massengill, and P. Gouker, "Single Event Transients in Digital CMOS—A Review," IEEE Transactions on Nuclear Science, vol. 60, pp. 1767-1790, 2013.
4. Z. Bin, W. Wei-Shen, and M. Orshansky, "FASER: fast analysis of soft error susceptibility for cell-based designs," in Quality Electronic Design, 2006. ISQED '06. 7th International Symposium on, 2006, pp. 6 pp.-760.
5. N. Miskov-Zivanov and D. Marculescu, "MARS-C: modeling and reduction of soft errors in combinational circuits," in 2006 43rd ACM/IEEE Design Automation Conference, 2006, pp. 767-772.
6. R. R. Rao, K. Chopra, D. T. Blaauw, and D. M. Sylvester, "Computing the Soft Error Rate of a Combinational Logic Circuit Using Parameterized Descriptors," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 26, pp. 468-479, 2007.
7. M. Nicolaidis, "Soft Errors in Modern Electronic Systems," 2011.
8. B. Narasimham, M. J. Gadlage, B. L. Bhuva, R. D. Schrimpf, L. W. Massengill, W. T. Holman, et al., "Characterization of Neutron- and Alpha-Particle-Induced Transients Leading to Soft Errors in 90-nm CMOS Technology," Device and Materials Reliability, IEEE Transactions on, vol. 9, pp. 325-333, 2009.
9. R. Liu, A. Evans, Q. Wu, Y. Li, L. Chen, S. J. Wen, et al., "Analysis of advanced circuits for SET measurement," in Reliability Physics Symposium (IRPS), 2015 IEEE International, 2015, pp. SE.7.1-SE.7.7.
10. M. Ebrahimi, A. Evans, M. B. Tahoori, E. Costenaro, D. Alexandrescu, V. Chandra, et al., "Comprehensive Analysis of Sequential and Combinational Soft Errors in an Embedded Processor," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, pp. 1586-1599, 2015.

11. D. Munteanu and J. L. Autran, "Modeling and Simulation of Single-Event Effects in Digital Devices and ICs," Nuclear Science, IEEE Transactions on, vol. 55, pp. 1854-1878, 2008.
12. S. M. Jahinuzzaman, "Modeling and Mitigation of Soft Errors in Nanoscale SRAMs," 2008.
13. L. Artola, M. Gaillardin, G. Hubert, M. Raine, and P. Paillet, "Modeling Single Event Transients in Advanced Devices and ICs," Nuclear Science, IEEE Transactions on, vol. 62, pp. 1528-1539, 2015.
14. G. I. Wirth, M. G. Vieira, E. H. Neto, and F. G. L. Kastensmidt, "Single Event Transients in Combinatorial Circuits," in 2005 18th Symposium on Integrated Circuits and Systems Design, 2005, pp. 121-126.
15. J.-L. Autran and D. Munteanu, "SOFT ERRORS FROM PARTICLES TO CIRCUITS," 2015.
16. J. S. Kauppila, A. L. Sternberg, M. L. Alles, A. M. Francis, J. Holmes, O. A. Amusan, et al., "A Bias-Dependent Single-Event Compact Model Implemented Into BSIM4 and a 90 nm CMOS Process Design Kit," Nuclear Science, IEEE Transactions on, vol. 56, pp. 3152-3157, 2009.
17. M. Slimani and L. Naviner, "A tool for transient fault analysis in combinational circuits," in 2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS), 2015, pp. 125-128.
18. C. Geng, J. Liu, Z.-G. Zhang, K. Xi, S. Gu, M.-D. Hou, et al., "Modeling the applicability of linear energy transfer on single event upset occurrence," Chinese Physics C, vol. 37, p. 066001, 2013.
19. R. Doering and Y. Nishi, "Handbook of semiconductors manufacturing technology," 2008.

From Digital Timing Diagrams to Natural Language and Back

Josep Carmona

Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract. Digital Timing Diagrams have been and are and effective visualization aid for the understanding of a digital circuit. However, in case of complex circuits, the interplay between signals and the corresponding hidden dependencies may be missed. In this paper we consider the textual representation of digital timing diagrams, as an alternative way of describing a digital circuit. We provide ideas on how to transform a digital timing diagram into a textual description, and the (more challenging) opposite problem: obtaining a digital timing diagram from a textual description.

1 Introduction

I will always remember the first time I met Alex Yakovlev: Newcastle, (very cold) winter of 1998, my (at that time, future) PhD. advisor Jordi Cortadella took me to the ACID workshop to convince me to do a PhD. with him on asynchronous circuits synthesis. The first night I discovered two important things: first, the warm character of Alex and his family, who hosted me in a great dinner, which remarkably included dancing at the end. Since then I have met Alex in several conferences, and have collaborated with his group in different topics (asynchronous circuits, theory of regions, process mining). And the second thing I discovered ... Jordi is a great dancer!!!

In this paper I sketch some recent ideas I had about the use of Natural Language Processing (NLP) techniques to support the analysis and elicitation of timing diagrams. I shall tell a secret: my very first paper was on NLP techniques, but at some point I got into the dark side of formal methods and never went back to NLP. It is funny that I recently got again attracted for working on the NLP area.

A digital timing diagram is a representation of a set of signals in the time domain. A timing diagram can contain many rows, usually one of them being the clock (but we do not deal always with clocks, as Alex knows very well). It is a tool that is ubiquitous in digital electronics, hardware debugging, and digital communications. Besides providing an overall description of the timing relationships, the digital timing diagram can help find and diagnose digital logic hazards.

2 Motivating Example

Let us consider the following textual description of the timing diagram of Figure 1.

Example 1 (Asynchronous Circuit). The circuit contains signals A, B and C. First, signal A goes high, which causes signal B to go high. Then, the rising of signals B and C causes signal C to go low. Afterwards, the rising of signal C causes signals A and B to go high. Then the falling of signal A causes signal B to go low.

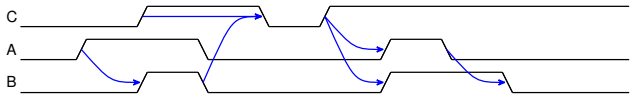


Fig. 1: Timing Diagram of an Asynchronous Circuit

Also, an alternative would be to describe the same behavior in a clocked way, as depicted in Figure 2:

Example 2 (Clocked Circuit). The synchronous circuit contains signals A, B and C. First, signal A goes high, which causes signal B to go high in the next clock edge. Then, the rising of signals B and C causes signal C to go low two clock cycles afterwards. In the next clock cycle, the rising of signal C causes signals A and B to go high one cycle afterwards. One clock cycle later, the falling of signal A causes signal B to go low.

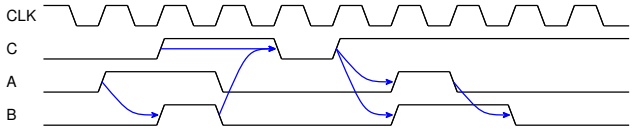


Fig. 2: Synchronous version of the circuit of Fig. 1

Next sections would illustrate how to go from the textual to the graphical description and back.

3 From Timing Diagrams to Natural Language

There are several formal descriptions of digital timing diagrams. We take a simple one, which is used by the tool `TimingDrawer` [1]. For instance, the timing diagram shown in Fig. 2 is simply specified with the following instructions:

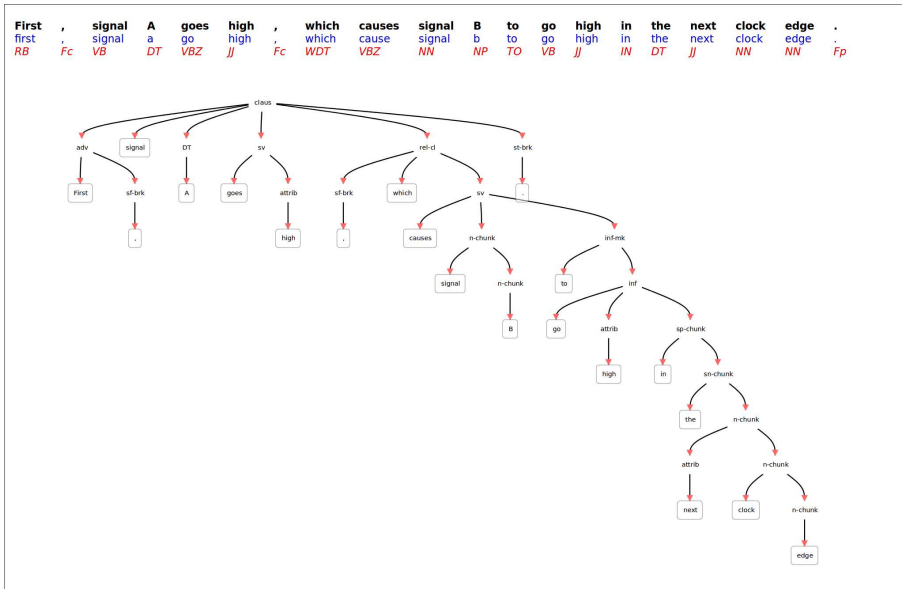


Fig. 3: Parsing for a sentence of the timing diagram textual description.

```
CLK=clock;C=0;A=0;B=0.
# Dependency from previous A edge to new value of B
A=1.
A=>B=1;
# Dependency from multiple signals
C=1.A=0;B=0.
C,B=>C=0.
# Dependency to multiple signals
C=1.
C=>A=1,B=1.
# Vertical dependency
A=0.
A=>B=0.
```

Given a text file in the previous format, one can generate a textual explanation by: i) parsing the file in order to get a tree-like structure of the timing diagram, and ii) traversing the tree to generate the corresponding explanation in natural language, using template sentences that may be instantiated with the real names found in the tree.

4 From Natural Language to Timing Diagrams

The opposite problem to the one faced in the previous section is a challenging one: given a text describing the main behavior of a digital timing diagram, derive

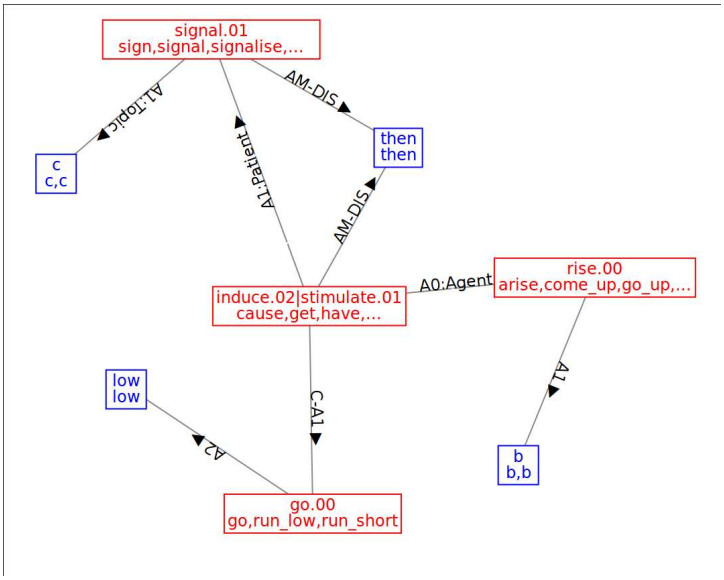


Fig. 4: Semantic Graph for a sentence of the timing diagram textual description.

a formal representation (e.g., the one used by `TimingDrawer`) of it. Inspired by the approach presented in [2] to obtain process diagrams from textual descriptions, *Natural Language Processing* (NLP) techniques can be used to tackle this problem.

Likewise it is done in [2] for the case of process diagrams, the generation of a digital timing diagram can be done in three steps. However, given the narrower focus considered in this paper, some of the steps can be significantly simplified. Below we provide an informal description of each one of the three steps considered.

Step 1: Sentence Level Analysis Using NLP techniques (e.g., tokenization, parsing, and similar), sentences in the input text can be analyzed to decompose the input into phrases with clear actors (signals, in our case) and the actions corresponding to them. Also, irrelevant or unrelated sentences are filtered. Morphosyntactic analysis is one of the prominent techniques to apply, that may derive a categorization as provided in Figure 3.

Step 2: Text Level Analysis Then the output of the previous phase is analyzed, taking into account the relationship between signals and/or signal actions across different phrases (a phenomena well known as *anaphora resolution*). For each sentence, a *semantic graph* describing these relations can be obtained. Figure 4 depicts an example of the semantic graph obtained from some of the sentences describing the digital circuit of the previous section. Nodes in this graph denote semantic meanings of the words in the sentences, and arcs represent the semantic

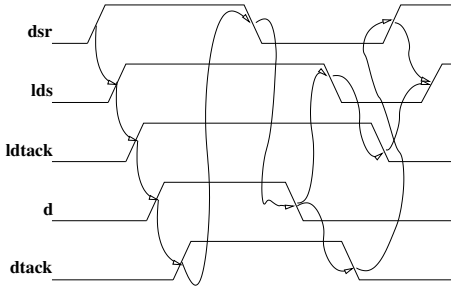


Fig. 5: VME Bus Timing Diagram.

relations between them. For instance, in Figure 4 it can be extracted that the rise of B (arc between concept "rise" and concept "B", with "B" as main actor denoted by "A1"), causes signal C (arc between concept "induce" and "signal" concept, with semantic arc "A1:Patient" denoting the signal to be the result) to go low (arc between concept "induce" and concepts "go" and "low", through arcs "C-A1" and "A2", respectively).

Step 3: Timing Diagram Generation Given the previous analyses, traversing the corresponding data structures would allow to generate the timing diagram corresponding to the input textual description. The idea would be to select the meaningful nodes/arcs from the semantic graph that can be translated into causalities in the timing diagram, generating a formal description as the one used by `TimingDrawer`.

5 Discussion

In the last decades, Alex has been a key person in the field of asynchronous circuits. One of the first works that I read from Alex was describing a VME bus controller with Signal Transition Graphs, which are labeled Petri nets representing the behavior of a digital circuit. The timing diagram of the read cycle of the controller is shown in Figure 5. In my research on asynchronous circuits (CSC encoding, synthesis), I was using this example all the time. I hope Alex can consider this paper as a way to pay back his enormous influence on the area, and in particular, on my work.

References

1. Salokin, V.: TimingDrawer Tool. https://sourceforge.net/p/timingdrawer/wiki/Home/ (2016)
2. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: Advanced Information Systems Engineering - 23rd International Conference, CAiSE 2011, London, UK, June 20-24, 2011. Proceedings. (2011) 482–496

Are Asynchronous ideas useful in FPGAs?

Peter Y. K. Cheung

Department of EEE, Imperial College London
London SW7 2BT
p.cheung@imperial.ac.uk

Abstract. Professor Yakovlev has contributed to the field of asynchronous designs and methodologies for over three decades. I have worked in the area of reconfigurable circuits and systems for almost as long. On this happy occasion celebrating his 60th birthday, it is fitting to make an attempt to examine possible synergy between the two fields and reflect upon where we might find a common intersection if one exists.

Keywords: asynchronous circuits, reconfigurable, FPGAs

1 Introduction

I have been looking forward to celebrate Professor Alex Yakovlev's academic achievements with him at his Festschrift for the past three years, ever since my own Festschrift held in 2013 [1]. At that time, he contributed a thoughtful article and spoke eloquently at the workshop, and I have been asking him when his own Festschrift will take place ever since.

Although I have known Alex and admired his work for over two decades, particularly in his contributions towards the modelling of asynchronous systems using Petri nets [2] [3]], he and I have not worked together on a project until very recently. I have dabbled in asynchronous circuits in the past, but my own area of research has, for the past two decades, been in the field of reconfigurable systems, particularly those using FPGAs. Therefore it is obvious for me to contribute on this happy occasion an article that focuses on both asynchronous ideas and FPGAs.

I have chosen to write this article completely from scratch. I want to take this opportunity to pontificate on the subject of asynchrony in the context of configurable and reconfigurable digital circuits. The nice thing about writing for a Festschrift is that I can be less rigorous and scientific, more reflective and opinionated, without the need to support my hypotheses and thoughts with objective and experimental data. This is like a columnist writing an article for the Financial Times, such as my favourite economist Mr John Kay, whose many insightful essays have taught me much.

2 My journey into Asynchronous Designs

Before I attempt to relate asynchronous circuits to FPGAs, it is worth putting my thoughts in context. My own experience in asynchronous circuits is limited. I was first introduced to the subject when Ivan Sutherland spent a 15-months sabbatical leave in our Research Group at Imperial College in the mid-80s. At that time, he was working on the design of an asynchronous multiplier [4] which also led to him developing the method of logical effort [5]. I was a young lecturer at the time trying to establish myself in the competitive world of academia. Through Ivan and his research assistant Ian Jones, I learned the basic principles of asynchronous circuits, such as asynchronous FIFO, the C-element, dual-rail signaling etc.. Subsequently, I recruited three research students to work with me on the subject and published a few insignificant papers [6] [7] [8]. We spent many hours debating among ourselves the pros and cons of asynchronous circuits, and through our discussions and arguments, we came to the conclusion that we wanted to work in the area of composability of asynchronous systems.

A few years later, my friend and colleague Wayne Luk from Oxford joined Imperial College. Mainly because of this, I moved my research to the area of reconfigurable systems.

3 Asynchronous Reconfigurable Logic

Early attempts to put asynchronous circuits in conventional synchronous FPGAs failed miserably. FPGA fabrics are fundamentally register-rich in architecture and early flip-flops in FPGAs were particularly prone to metastability problems. Asynchronous components such as the C-element could be implemented using fine-grain architectures on those early generations of FPGA fabrics such as the XC6200 [9] and Actel's ACT-1 devices [10]. Unfortunately implementing reliable asynchronous systems of reasonable size were very challenging. This prompted researchers to design asynchronous FPGAs with special circuit elements dedicated to the asynchronous design style.

There have been various attempts in designing asynchronous FPGA fabrics. One of the earliest efforts was by Scott Hauck and Carl Ebeling [11]. Montage was the first FPGA that included dedicated support for both asynchronous and synchronous circuits. It even came with mapping software to support designers. It was shortly followed by the work of Payne [12], Traver [13], Teifel [14] and Martin [15]. As someone who has a passing interest in asynchronous circuits and techniques, and made his research career in FPGA, I have frequently asked the question: why asynchronous FPGAs never took traction either in industry or in academia? The following is a brief exposition of my personal views in the matter.

4 The potential and reality of asynchronous in reconfigurable

On paper asynchronous circuits have many attractive features [16]. The idea of doing away with rigid timing imposed by synchronous circuit's clock signal can be appealing - even liberating! However, are these potential advantages relevant and applicable to FPGAs?

4.1 Clock skews

Potential - Eliminating the need to distribute clock signals throughout a chip also removes the problem of clock skews. Clock networks in modern FPGAs are complex and occupy significant amount of silicon area. Unlike ASIC designs, FPGA devices need to accommodate arbitrary application circuits not known to the FPGA architect at the time the clock networks are designed. The loading on nodes in a clock tree are also not known before hand. This exacerbates the clock skews problem even further.

Reality - Indeed handling clock skews on large FPGAs is a problem. It may be possible to measure the skews and mitigate their effects at run-time as suggested in [17]. The effectiveness of such an approach is unproven. However, having clocks in FPGAs is so fundamental to the entire FPGA design flow that industry simply chooses to solve the clock skews problem, no matter how challenging. Until recently, the approach taken has been to carefully design fixed, highly buffered, clock tree networks and to develop accurate timing models of the reconfigurable fabric including the programmable routing resources. The timing model is tightly coupling to place-and-route algorithms in the CAD tools.

Recently, Altera has taken a much more radical approach to this problem [18] in their latest Stratix-10 family of FPGAs. Instead of using fixed clock networks, they use a highly configurable, routable approach to customize the clock network based on the user's design. This approach potentially reduces the impact of clock skews and makes the technique scalable for future larger FPGAs. Xilinx has taken a different approach in their Ultrascale architecture [19]. Distribution of clocked is based on clock regions which, unlike Stratix-10, are not customizable. However, the clock grid comprises of routing tracks and distribution tracks, which provide optimal routing of the clock signals from the centre of the region (called clock root).

Both Altera and Xilinx have successfully tackled the clock skews problem in different ways. In both cases, clock skews will not be a show stopper as we move into future generations of larger, faster, and denser FPGAs.

4.2 Timing closure

Potential - Any designer who has grappled with fitting a large design onto an FPGA will tell you that timing closure is often a big headache. FPGA designs

relies heavily on the flexible, programmable interconnect resources. The unpredictability in the delays of interconnects often presents many challenges. Even a minor change in a previously working design could require many hours of work in order to achieve timing closure again. The self-timed nature of asynchronous FPGAs could do away with this problem.

Reality - As size and density of FPGA devices keep increasing, achieving timing closure is expected to be increasingly difficult. However, FPGA manufacturers have made great strides in easing this through two advances. Xilinx has pushed stacked silicon interconnect technology that promises to deliver much higher capacity, higher data bandwidth and power efficiency [20]. Altera recently announced their highly pipelined Stratix-10 architecture where they include programmable pipeline registers within the routing fabric. Coupled with their CAD tools that are designed to exploit this architectural innovation, they mitigate the timing closure problem by simplifying circuit retiming with improved performance [21].

4.3 Reduced power

Potential - Having to distribute a clock signal everywhere, even when to those registers where the input signals remain unchanged consumes unnecessary power. Even though clock gating helps in reducing unnecessary clock propagation to inactive regions, the idea that data drives circuit activities can be very attractive. FPGAs are already much larger and consume more power than their ASIC equivalent. Asynchronous circuit could regain some of the power efficiency loss.

Reality - Although asynchronous circuits have the potential of consuming less power than their synchronous counterpart, FPGAs have never really been the technology of choice where power is the primary concern. One would not find many FPGAs used in battery operated mobile devices. Therefore reduced power consumption has never been a significant driver in pushing the asynchronous agenda in FPGAs. Furthermore, the advances in process technology and the continual reduction in power supply voltages have alleviated the concern of overheating in large FPGAs. For example, using TSMC 28nm low power (HPL) process, Xilinx has introduced three separate families of devices to meet the market's requirement: the Artix-7 devices for low power and low cost, the Kintex-7 devices for a balance in price and performance, and the Virtex-7 devices for high performance and high capacity [22]. In the meantime, Altera introducing Aria-10 and Stratix-10 families of FPGAs, both claiming significant improvements in power consumption for a given workload when compared with previous generations of FPGAs [23].

4.4 Improved performance

Potential - It is well known that asynchronous circuits allow *average-case* instead of *worst-case* performance. Synchronous FPGAs, for example, would have

to wait for the ripple-carry chain to reach the most significant bit in an add operation before the sum is available. Asynchronous FPGAs with its inherent completion detection and signalling would allow on average faster operations.

Reality - The arguments for improved performance using asynchronous techniques have never been proven in industrial designs. Neither Sun Microsystem's asynchronous Sparc processor nor AMULET, the asynchronous ARM processor, demonstrated performance gain over their synchronous counterpart implemented using the technology. In any case, having better *average-case* performance gain is no advantage in many real-time applications where a guaranteed completion time is more important than having faster operation some of the times. Deterministic behaviour is often more desirable than indeterministic, data-dependent delay or latency. In any case, in the world of FPGAs, advancement in process technology, new transistor devices (e.g. FinFET transistors) and new FPGA architectures have continued to push the boundary of FPGA performances. In addition, the use of 2.5D and 3D integration in FPGAs significantly reduces interconnect delays, which are particularly important for FPGA designs where interconnect delays are often the main limitation to system level performance.

4.5 Composability and reusability

Potential - Delay insensitivity in asynchronous circuits allows the assembly and integration of modules forming a complete digital system with ease. This promotes the divide-and-conquer approach to design and allows previously designed and verified modules to be reused with minimal effort, very much like the way software library functions can be used in building a large programme. Therefore asynchronous techniques permit FPGA designers easier routes to perform reconfiguration of virtual hardware blocks even during runtime.

Reality - There have been much effort devoted to the idea of run-time reconfiguration (RTR) in FPGAs so that modules could be swapped in and out of the FPGA fabric, very much like the way we swap blocks of memory in a virtual memory system. The asynchronous paradigm where each module is self-contained with its own timing appears to be a perfect match to this particular vision of FPGAs. Unfortunately RTR has not really been widely used in industry beyond design upgrade or system retargeting. There are many reasons for this including the overhead in reconfiguration if it is done frequently, the difficult in verification of such a system (similar to the reason why self-modifying code is not encouraged in software), and the lack of good CAD tools to handle such a RTR system.

However, with ever increasing size and density in FPGAs, partial reconfiguration is now becoming not just a reality but a necessity. Putting a working design on an FPGA device with 2 million logic cells requires a design flow that allows composability and resuability. The FPGA industry is currently solving this problem, not with the asynchronous paradigm, but with high-level synthesis methodology.

There is now a widespread adoption of OpenCL as the language of choice for FPGA designs, particularly after the integration of powerful embedded CPU into the FPGA fabrics as found in Xilinx's and Altera's latest family of devices. This approach is totally opposite of that used in the asynchronous paradigm. In asynchronous, a successful design requires high degree of skills and understanding of hardware design down to the minute details. The approach taken by the FPGA industry is to push the hardware design skills required to a minimum and makes the FPGA fabric invisible. I personally will not bet against the scenario that very soon one could put designs on FPGAs without touching any low level descriptions such as Verilog or VHDL.

The introduction by Altera of highly pipelined flexible registers in its routing resources helps this process even further [21]. Now an OpenCL compiler can deal with the hardware compilation task without worrying about timing. The flexible routing registers can then be allocated subsequently to optimise the performance and the latency of a design. In my view, this is one of the most important advancements in FPGA architectures that enables effective high-level synthesis in recent years.

4.6 Tolerance to process and environmental variations

Potential - As technology scaling continues to advance and supply voltage continues to drop, uncertainties in delays in FPGA circuits due to variations in process, temperature, voltage and noise are beginning to post a challenge. Asynchronous circuits adapts to delay variation automatically. This is particularly significant if and when future electronic devices degrade faster with age. Older asynchronous FPGA chips will continue to function correct with reduced speed, while the synchronous counterpart could fail completely.

Reality - Mitigating the effect of variability with reconfigurable hardware has been the focus of my research in recent years. Alex contributed an article at my Festschrift on this very same topic [24]. FPGAs are inherently adaptable. It is therefore reasonable to ask the question how could one exploit the flexibility offered by FPGAs to mitigate the ever increasing variability factors found in modern integrated circuits?

Asynchronous paradigm is one possibility but this does not really need the configurability offered by FPGAs. The approach I have taken so far has been to perform online measurements on FPGAs devices with embedded instrumentations. One can then exploit such information to adapt the electronic systems in order to deal with variations by exploiting the configurability inherent in FPGAs. For example, we have recently successful demonstrated efficient and effective circuits to measure delays and timing slacks in arbitrary circuit paths [25] [26] and power consumption in modules within FPGAs [27]. These techniques

are particularly suitable for FPGA based designs for the following reasons. Not only are FPGA reconfigurable, they also come with fixed amount of hardware resources on a chip. Provided that a design does not fill the entire chip, there are generally spare hardware resources that one could deploy for such embedded instrumentations. Putting the unused resources to provide better adaptation not only improves performance, reduces power consumption, it also improves the reliability of the system. The challenge is in how to add such instruments without impacting on the already complicated design flow. For this, we can learn from the experience in the adoption of boundary scans in test. Boundary scan register insertion in ASIC is now an automatic and, at least to the designer, a more or less invisible process. It is my view that one day on-chip embedded instrumentations will be the same.

5 Concluding Remarks

When I first set out to write this article, I was intending to focus on the synergy between asynchronous and reconfigurable. As I was completing the article, I realised that my conclusion is rather negative towards asynchronous. This is both a surprise and a disappointment. I have always found the asynchronous paradigm rather beautiful. I also admire those who have made this their lifelong work. However, no matter how much I want to make asynchronous ideas relevant and useful to FPGAs, everywhere I turned, I was confronted with strong counter arguments. The reality of industrial practices also suggests that I am probably right.

Notwithstanding, I strongly believe that there is one aspect where the asynchronous paradigm is useful to FPGAs and this is in the integration of modules via the globally asynchronous, locally synchronous (GALS) methodology. If I look back in the history of electronics and the one asynchronous idea that we still use everyday is the serial communication using universal asynchronous receiver/transmitter (UART). Indeed communicates between multiple FPGAs are already relying on the very fast serial transceivers found on all modern FPGAs.

I have attending a few conferences on asynchronous in the past. Just like the FPGA community, the asynchronous community has spent many years searching for the "killer application" for their technology. May be the answer is that there is *no killer application*, but asynchronous is still a very useful glue. The paradigm, particularly with Alex's original field of research into modelling using Petri net, can act as the catalyst to many innovations in electronics.

In conclusion, while I still believe that asynchronous is useful in the field of reconfigurable, we are still searching for this intersection between the two fields. May be the recent effort of companies such as Achronix will show us the way. I am still waiting.

6 Acknowledgements

I want to thank Professor Alex Yakovlev for his friendship and encouragements that he generously offered to me over many years. I have enjoyed our many open and honest discussions, both technical and non-technical. I look forward to working with you for many years to come. May be we can even start something completely new in the future.

I also want to acknowledge the generous support of EPSRC in the following grants: PRiME: Power-efficient, Reliable, Many-core Embedded Systems (EP/K034448) and Variation-Adaptive Design in FPGAs (EP/H013784).

References

[1] Luk, W. and Constantinides, G.A., Transforming Reconfigurable Systems: A Festschrift Celebrating the 60th Birthday of Professor Peter Cheung. Imperial College Press, 2015

[2] J. Cortadella and M. Kishinevsky and L. Lavagno and A. Yakovlev, Synthesizing Petri nets from state-based models. Computer-Aided Design, 1995. ICCAD-95. Digest of Technical Papers., 1995 IEEE/ACM International Conference on, pp.164-171, 1995

[3] J. Cortadella and M. Kishinevsky and L. Lavagno and A. Yakovlev, Deriving Petri nets from finite transition systems. IEEE Transactions on Computers, 47 (8), pp. 859-882, 1998

[4] Sutherland, I. E., Micropipelines. Commun. ACM, 32 (8), pp. 720–738, 1989

[5] Sutherland, Ivan E. and Sproull, Robert F., Logical Effort: Designing for Speed on the Back of an Envelope. Proceedings of the 1991 University of California/Santa Cruz Conference on Advanced Research in VLSI, pp. 1-16, 1991

[6] Molina, P. A., Cheung, P. Y. K. Quai delay-insensitive bus for fully asynchronous systems. IEEE International Symposium in Circuits and Systems, Vol. 4, pp. 189-192, 1996

[7] Bormann, D. S., Cheung, P. Y. K. Asynchronous wrapper for heterogeneous systems. International Conference on Computer Design, ICCD'97 pp. 307-314, 1997

[8] Royal, A., Cheung, P. Y. K. Globally Asynchronous Locally Synchronous FPGA Architectures International Workshop on Field Programmable Logic and Applications, LNCS 2778, pp. 355-364, 2003

[9] Brebner, Gordon, A virtual hardware operating system for the Xilinx XC6200. Proceedings of the 6th International Workshop on Field-Programmable Logic and Applications, pp. 327–336, FPL'96, 1996

[10] Brunvand, Erik A Cell Set for Self-Timed Design using Actel FPGAs Report UUCS-91-013, Department of Computer Science, University of Utah August 10, 1991

[11] Hauck, S., Burns, S., Borriello, G., Ebeling, C. An FPGA for Implementing Asynchronous Circuits. IEEE Design & Test of Computers, 11 (3), pp. 60-69, 1994

[12] Payne, R. E. Self-Timed FPGA Systems. 5th International Workshop on Field Programmable Logic and Applications, pp. 21-35, 1995

[13] Traver, C., Reese, R. B., Thornton, M. A. Cell Designs for Self-Timed FPGAs. 14th Annual IEEE International ASIC/SOC Conference pp. 175-179, 2001

[14] Teifel, J., Manohar, R. Highly Pipelined Asynchronous FPGAs Proceedings of the 2004 ACM International Symposium on FPGA pp. 133- 142, 2004.

[15] Wong, C. G., Martin, A. J., Thomas, P. An Architecture for Asynchronous FPGAs. IEEE International Conference on Field-Programmable Technology pp. 170-177, 2003
[16] Hauck, S. Asynchronous Design Methodology: An Overview Proceedings of the IEEE, 83 (1), pp. 69 - 93, January 1995
[17] Sedcole, P., Wong, J. S., Cheung, P. Y. K. Characterisation of FPGA Clock Variability IEEE Symposium on VLSI, pp. 322 - 328, 2008
[18] Ebeling, C., How, D., Lewis, D., Schmit, H. Stratix 10 High Performance Routable Clock Networks Proceedings of the 2016 ACM International Symposium on FPGA, pp. 64-73, 2016
[19] Xilinx Inc. Ultrascale Architecture clocking Resources - User Guide. *www.xilinx.com/support/documentation/user_guides/ug572-ultrascale-clocking.pdf*
[20] Saban, S. Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency. White Paper WP380, Xilinx Inc., Dec 11, 2012
[21] Lewis, D., Chiu, G., Chromczak, J., Galloway, D., Gamsa, B. The Stratix 10 Highly Pipelined FPGA Architecture Proceedings of the 2016 ACM International Symposium on FPGA, pp. 159-168, 2016
[22] Mohson, E. Reducing System Power and Cost with Artix-7 FPGAs. White Paper WP423, Xilinx Inc., Dec 17, 2014
[23] Lim, S. Expect a Breakthrough Advantage in Next-Generation FPGAs. White Paper WP-01199, Altera Corp., June 2015
[24] Yakovlev, A. Enabling Survival Instincts in Electronic Systems: An Energy Perspective. In Transforming Reconfigurable Systems: A Festschrift Celebrating the 60th Birthday of Professor Peter Cheung Imperial College Press, 2015
[25] Wong, J. S. J., Cheung, P. Y. K. Timing Measurement Platform for Arbitrary Black-Box Circuits Based on Transition Probability. IEEE Transactions on VLSI Systems, 21 (12), pp. 2307-2320, 2013
[26] Levine, J. M., Stott, E., Cheung, P. Y. K. Dynamic voltage and frequency scaling with online slack measurement. Proceedings of the 2014 ACM International Symposium on FPGA pp. 65-74, 2014
[27] Hung, E., Davis, J. J., Levine, J. M., Stott, E. A., Cheung, P. Y. K., Constantinides, G. A. KAPow: A System Identification Approach to Online Per-module Power Estimation in FPGA Designs. Proceedings of the 24th IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2016

Event splitting:
the way to survive when regions fail

Jordi Cortadella

Department of Computer Science
Universitat Politècnica de Catalunya, Barcelona

Abstract. The theory of regions, originated from the work by Ehrenfeucht and Rozenberg, established the bridge between transition systems and Petri nets and a path towards the friendly visualization of concurrent behaviors. Unfortunately, not every transition system can be represented with a Petri net in which every event corresponds to a single transition. However, a Petri net can always be found if events can be split and represented by multiple transitions. When applying event splitting, an exponential space of solutions arises, each one delivering a different Petri net. Selecting one with gracious properties is a challenge and an open problem. This paper will informally illustrate the impact of event splitting using simple examples and will discuss directions of research for this problem.

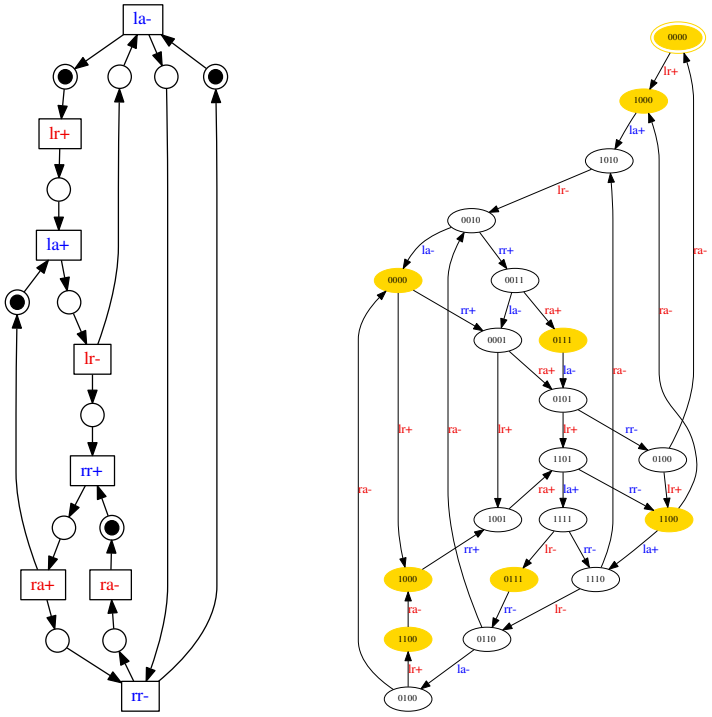
1 Motivation

Since I first met Alex Yakovlev in 1994, I have had the pleasure to share many unforgettable experiences in our professional and personal lives. Petri nets, asynchronous circuits and design automation have been permanent *leitmotifs* in our academic research. Along with our endearing colleagues, Mike Kishinevsky, Alex Kondratyev and Luciano Lavagno, we managed to solve challenging problems and many of them ended up enhancing the functionality of `petrify` [4], a tool that is still alive today.

During the nineties, our passion was to introduce automation in the design and verification of asynchronous circuits. We had a devotion for Petri nets and their interpretation as Signal Transition Graphs (STGs) [11]. Since logic synthesis techniques required state information with explicit values for the binary signals [5], we were often generating transition systems (TSs) from Petri nets. At that time, we decided to use Binary Decision Diagrams [2] for representing sets of states symbolically and handle the state explosion problem in a manageable way.

Figure 1 shows a classical synthesis example. Figure 1a depicts the specification of an asynchronous controller with an STG[1], whereas the corresponding TS is shown in Fig. 1b. Each state has an associated binary code with as many

[1] Most drawings in this paper have been automatically generated by `graphviz` [9], a tool set that can be found in `www.graphviz.org`.



(a) Signal Transition Graph. (b) TS with encoding conflicts.

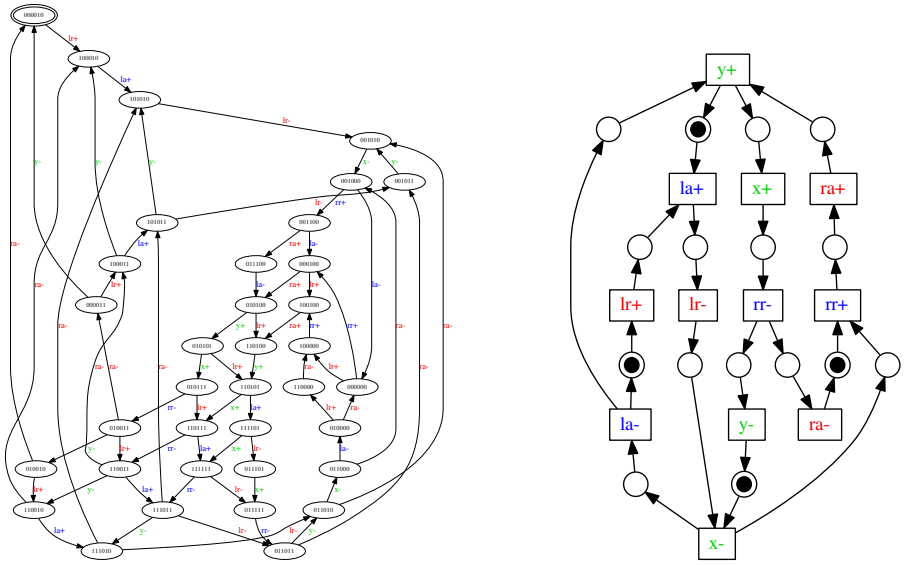
Fig. 1. Generation of a binary-encoded TS from an STG.

digits as signals in the system. The shadowed states represent encoding conflicts, i.e., multiple states sharing the same binary code.

The state encoding problem was one of the many problems we had to solve for the synthesis of asynchronous controllers [6]. We managed to find a satisfactory solution adding new signals and transforming the TS, as shown in Fig. 2a. In this particular example, two new internal signals, x and y, are introduced to disambiguate the state encoding conflicts.

We were often facing problems that required transformations at the level of TS: hiding signals, adding new state signals, reducing concurrency, etc. However, evaluating the impact of those transformations was a challenge, since the visualization and analysis of TSs was an extremely arduous task.

The TSs we had to manage often had hundreds or thousands of states, with a lot of concurrency embedded in their semantics. Unfortunately, we were no able of analysing the results of the transformations with a reasonable human effort. We were asking ourselves: why should not we be able to visualise the behaviour of a TS as an STG? It was then when the theory of regions came into play [8, 1] as the instrument to return to the Petri net world from a TS.



(a) TS without encoding conflicts. (b) STG with the new signals.

Fig. 2. Specification of the asynchronous controller after solving the encoding problem.

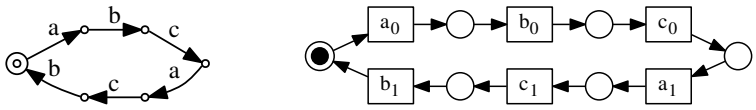
At that time, we were fascinated by the theory proposed by Ehrenfeucht and Rozenberg, that gave us the vehicle to synthesise STGs after manipulating our TSs. The implementation of that theory was the genesis of `petrify` [7]. With that vehicle we could generate the STG shown in Fig. 2b in which the causality relations of the new signals could be cleanly observed and analysed.

2 Petri net synthesis is not that simple

We immediately realised that things were not as simple as expected. For a TS to be representable as a Petri net, with each event represented by a single transition, a set of conditions must be fulfilled: the TS must be *elementary* [8]. In [7] we presented a more general concept, *excitation closure*, that defined a set of conditions to generate a Petri net with bisimilar behaviour [10]. A TS fulfilling those conditions is called an Excitation-Closed TS (ECTS). Unfortunately, most TSs are not ECTSs. A very simple example is shown in Fig. 3a.

However, the synthesis of a bisimilar Petri net is always possible by applying event splitting, i.e., allowing each event to be represented by more than one transition in the Petri net. For example, an event a could be represented by two transitions with labels a_0 and a_1 (the subindices represent different instances of the same label).

In the worst case, a degenerated solution consisting of a Petri net structurally isomorphic to the TS can be constructed: each place corresponds to one state

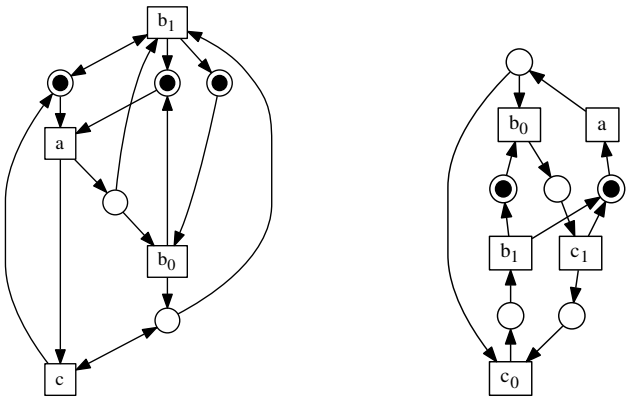


(a) Non-elementary TS. (b) Petri net obtained after splitting all events.

Fig. 3. Synthesis of a non-elementary transition system by event splitting.

of the TS and each transition to an arc. The initial state is represented by a marked place, as shown in Fig. 3b.

Unfortunately, this solution is quite uninteresting since it does not bring any additional information for the analysis of the system. The obvious question that comes to our mind is: can we minimise the number of events that need to be split to guarantee the synthesis of a Petri net?



(a) Splitting event b. (b) Splitting events b and c.

Fig. 4. Different solutions obtained after event splitting. Arcs with double arrow ($\bigcirc \leftrightarrow \square$) represent two arcs ($\bigcirc \rightleftarrows \square$).

Figure 4 depicts two bisimilar solutions obtained by splitting a different set of events. If we also consider the Petri net in Fig. 3b, we end up by having three different solutions with the following characteristics:

Figure	Places	Transitions	Arcs
3b	6	6	12
4a	5	4	18
4b	6	5	14

In this particular example, no solution with three transitions exists, since the original TS is not an ECTS. By analysing the previous table it is obvious to

realise that minimizing the number of transitions (event splits) is not always the best choice.

Even in the case of an ECTS, event splitting can be an option to generate a better visualisation of the behaviour. Figure 5 shows an ECTS with five events. A bisimilar Petri net with five transitions is depicted in Fig. 6a. The intricate relationship between places and transitions makes the analysis of this structure very tortuous. Instead, by simply splitting event a, the Petri in Fig. 6b is obtained, which clearly visualises the concurrency of events b and c and the choice between d and e.

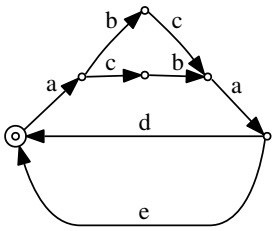
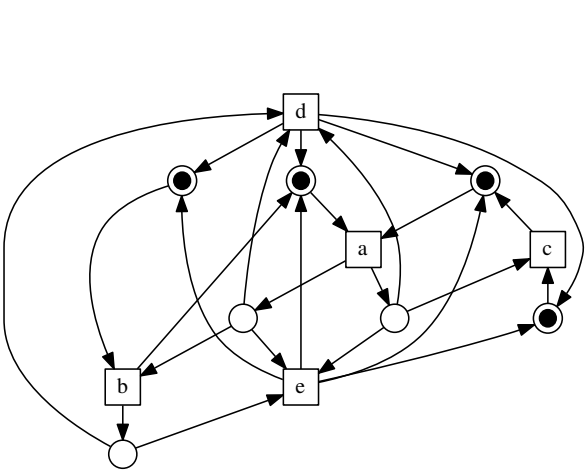
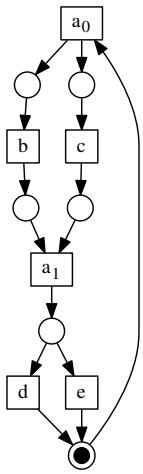


Fig. 5. Transition system.



(a) Without event splitting.



(b) Splitting event a.

Fig. 6. Two bisimilar Petri nets representing the behavior of the TS in Fig. 5.

3 Which events to split?

Event splitting is at the core of the Petri net synthesis problem. As we can suspect from the previous examples, the space of solutions can be exponential on the size of the TS. An essential question is the following:

> *How to measure the quality of a solution after selecting a set of events for splitting?*

As shown in Fig. 6, minimising the number of transitions may not always be the best choice, but doing a frenetic splitting may result in uninformative solutions. There is no clear answer for such question, but some directions for exploration are next discussed.

3.1 Minimising the number of transitions

Minimising the number of transitions is a strategy that may lead to more compact models. A possible approach could consist of finding a small set of events for splitting. This set would generate new regions to enforce the excitation closure for all events.

Figure 7a depicts a transition system and the set of minimal regions, $\{r_1, \ldots, r_6\}$, represented as shadowed sets of states. The excitation closure holds for all events except e. The only pre-region of e is r_4, but e is only enabled in one of the states.

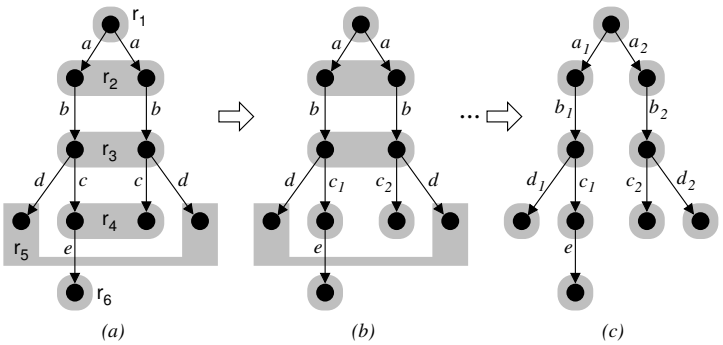


Fig. 7. The domino effect of greedy splitting (example from [3]).

A greedy approach, as proposed in [3], would consist in creating another region that would *separate* the two states in r_4. This could be achieved by splitting event c into c_1 and c_2 (Fig. 7b).

However, this myopic view solves the problem for e but creates another problem for c_1 and c_2. Applying the same strategy, excitation closure for c_1 and c_2 could be enforced by splitting b, etc, thus unleashing a domino effect that, in this particular case, would produce a complete event splitting, as shown in Fig. 7c.

This example shows that using naïve strategies for splitting may result in poor quality solutions.

An alternative approach could be proposed by globally analysing the excitation closure problem and resorting to the concept of state separation from the theory of regions [8]. This is illustrated in Fig. 8.

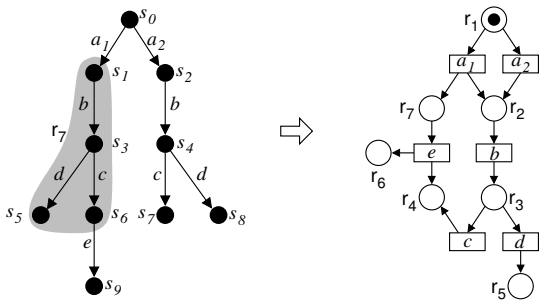


Fig. 8. New region (r_7) and Petri net synthesis after splitting event a.

By analysing the set of minimal regions $\{r_1, \ldots, r_6\}$ in Fig. 7a, we can calculate the pairs of non-bisimilar states that cannot be distinguished by the regions. Two states cannot be distinguished if there is no region such that one state is in the region and the other is not. In the example, the set of non-bisimilar pairs is:

$$\{(s_1, s_2), (s_3, s_4), (s_6, s_7)\}$$

Proposing heuristics to find set of states that *separate* these pairs is an interesting direction of research.

For example, the set $\{s_1, s_3, s_6\}$ could be a good candidate, but it would require two event splits to become a region: a and d. Instead, the set $\{s_1, s_3, s_5, s_6\}$ would guarantee the same separation with only one event split. This set corresponds to region r_7 in Fig. 8 and leads to the Petri net on the right, in which each place is annotated with the corresponding region.

3.2 Simplifying the structure of the Petri net

Having a nice visualisation contributes to giving a graphical intuition of the relationship between events. For example, producing series-parallel graphs or minimising the number of arc crossings in a picture is always a desired property for a Petri net.

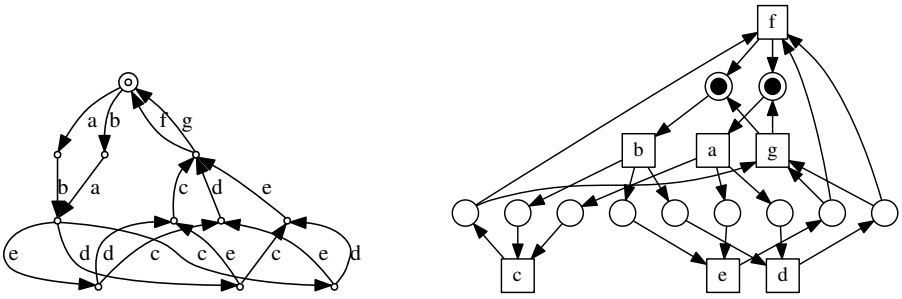
Analysing the connectivity and the causality relations between events can help to decompose sets of states in which an event is enabled. The TS in Fig. 5 is one example. Event a has two dispersed states in which the event is enabled. Additionally, each state is triggered by different sets of events ($\{b, c\}$ in one state and $\{d, e\}$ in the other state). The separation of enabling sets and the distinction

of trigger sets may be a criterion to split events. In this case, splitting event a results in the Petri net shown in Fig. 6b, which has nice structural and graphical properties.

3.3 An unexpected guest: τ

An alternative strategy to simplify the structure of a Petri net is to intentionally insert silent events (τ). In some cases, a new event can *collect* causality information between groups of events and contribute to better visualize their relationship.

An example is depicted in Fig. 9a, where two concurrent events (a and b) trigger another group of three concurrent events (c, d and e). This is represented by a two-way diamond preceding another three-way diamond. Similarly, c, d and e trigger two events in conflict (f and g) that later trigger events a and b.



(a) Transition system. (b) Petri net with bisimilar behaviour.

Fig. 9. TS with complex relationships between events.

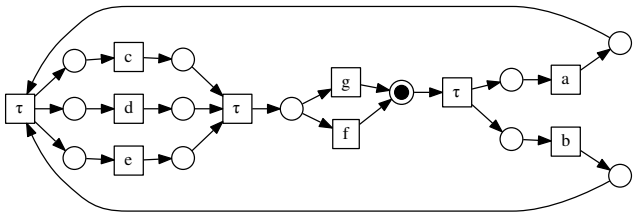


Fig. 10. Petri net after the insertion of τ events.

In a Petri net, a group of n concurrent events triggering another group of m concurrent events requires a set of $n \times m$ places representing the cross product

of relationships between pairs of events (see Fig. 9b). This situation can be analysed at the level of TS and insert silent events in strategic states that separate relationships between multiple events.

Figure 10 depicts a Petri net exhibiting the same behaviour as the one in Fig. 9b, but including some τ transitions. To be more precise, both Petri nets are weakly bisimilar [10]. The following table reports a comparative study about the structural properties of both Petri nets.

Figure	Places	Transitions	Arcs	Arcs/Nodes	Arc crossings
9b	11	7	27	1.50	12
10	12	10	26	1.18	0

For a fair comparison, both layouts have been generated by `graphviz`. The number of arc crossings is reported by the tool and gives an idea of the intricateness of the layout. Clearly, the layout shown in Fig. 10 is much more graphically intuitive. Quantitatively speaking, the ratio of arcs per node and the number of arc crossings are parameters highly related to the visualisation of the layout.

4 Surprise, surprise, ...

During a discussion on the problem of duplicate tasks in process mining, an interesting example came up. In process mining, the goal is to obtain a formal model from a set of behaviours represented by an event log.

Figure 11 shows a TS obtained by the event log at the left. Each one of the traces from the log corresponds to a different trajectory in the TS. The figure also shows a bisimilar Petri net after event splitting.

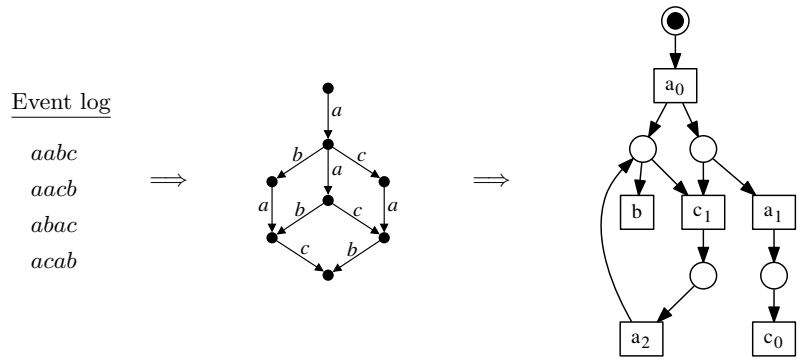


Fig. 11. TS and Petri net obtained from an event log.

It was interesting to realize that a much simpler Petri net, shown in Fig. 12, was able to generate exactly the same language. However, there is a small subtlety that differentiates them. The underlying TSs are not bisimilar, but they still are trace equivalent.

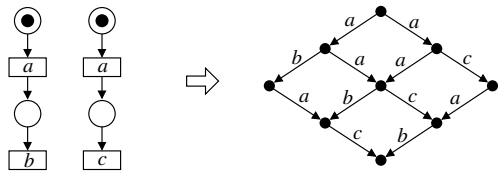


Fig. 12. Petri net generating the same language as the event log in Fig. 11.

The previous example suggests that event splitting can be solved in different ways depending on the equivalence that needs to be preserved during the synthesis of a Petri net.

5 Conclusions

For many years I have had the opportunity to share many exciting discussions about a large variety of research problems with Alex Yakovlev. After such a long time, some of the problems we have tackled have been solved, many of them are still open and many others are still unknown.

This paper just showed one of the open problems that will surely draw the attention of some researchers in the near future. My only hope is that the examples shown in the paper stimulate new ideas and discussions such as the one we had with Alex in Dresden (March 2016) when chatting about this topic and, more in particular, about the last example shown in the previous section.

References

1. Eric Badouel, Luca Bernardinello, and Philippe Darondeau. *Petri Net Synthesis*. Springer-Verlag, 2015.
2. Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, 35(8):677–691, August 1986.
3. Josep Carmona. The label splitting problem. *Trans. Petri Nets and Other Models of Concurrency*, 6:1–23, 2012.
4. J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE Transactions on Information and Systems*, E80-D(3):315–325, March 1997.
5. J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. *Logic Synthesis of Asynchronous Controllers and Interfaces*. Springer-Verlag, 2002.
6. Jordi Cortadella, Michael Kishinevsky, Alex Kondratyev, Luciano Lavagno, and Alexandre Yakovlev. A region-based theory for state assignment in speed-independent circuits. *IEEE Transactions on Computer-Aided Design*, 16(8):793–812, August 1997.
7. Jordi Cortadella, Michael Kishinevsky, Luciano Lavagno, and Alexandre Yakovlev. Deriving Petri nets from finite transition systems. *IEEE Transactions on Computers*, 47(8):859–882, August 1998.

8. Andrzej Ehrenfeucht and Grzegorz Rozenberg. Partial (set) 2-structures, parts i-ii. *Acta Informatica*, 27:315–368, 1990.
9. John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. Graphviz - Open Source Graph Drawing Tools. *Graph Drawing*, pages 483–484, 2001.
10. Robin Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
11. Leonid Ya. Rosenblum and Alexandre Yakovlev. Signal graphs: From self-timed to timed ones. In *International Workshop on Timed Petri Nets*, pages 199–206, Washington, DC, USA, 1985. IEEE Computer Society.

The Scientific Craftsperson: Beauty, Engineering and the Bohemian Researcher

Crescenzo D'Alessandro

Independent submission

Abstract. "Engineering" and "Beauty" seem to be diametrically opposed concepts: one concerned with hard reality, experimentation and evidence; the other generally associated with subjective, culturally-shaped experiences. And yet science and beauty are often discussed together, and many parallels have been drawn between the two disciplines. Engineering solutions are often considered beautiful, at least by engineers (think of a Phase Lock Loop, for instance) - but are they? Can the pursuit of beauty help the engineering endeavour? What about elegance - does this concept have any more bearing on engineering? Pushing engineering outside its traditional boundaries, this paper attempts to show that in spite of the dangers of aesthetic rules inevitably steering the judgement of the clinically-minded engineer, beauty and elegance do indeed have an often undermined, but positive effect on the practice of engineering and these values should be nurtured instead of discounted.

1 Introduction

Science and Beauty have walked hand in hand since the beginning of civilization, but their relationship has been a complex one, which continues to evolve through the times. In an age where the natural, irregular, "organic" beauty of nature dominates the everyday life of early populations, man-made precision, which hinted at the metaphysical world of perfection of the gods, becomes "beautiful"; this precision is expressed for example in perfect shapes and perfect colours to replicate the unconscious beauty of nature. Both types of perfection require refined techniques and technologies, which in turn require ever more refined science to allow these techniques and technologies to blossom.

In an era of man-made precision, the concept of Beauty is extended and becomes less literal. In order to progress on this, we will refer to "Beauty" as the sense of awe and pleasure which fills us when observing (in the widest possible sense) something. Returning to Nature we can find beauty in the exploration and description of a Nature-puzzle to be solved and exploited, or in the awe of discovering a sophisticated mathematical fabric underpinning the whole (set of) Universe(s), or simply in the visually attractive patterns created by water as it freezes over a windscreen – indeed the mathematical-physical description of such patterns. In these examples science is either itself the source of Beauty or the mechanism which unveils it from beneath the seemingly unpredictable behaviour of invisible forces. What does it mean to talk about beauty in science? The

easy descriptions of beauty as visually pleasing when applied to 3-dimensional fractal patterns is unsatisfactory: many lay people would consider the symmetry embedded in some mathematical functions, or the sound of a pulsar many million light years away as "beautiful". Often we hear that $E = mc^2$ is "the most beautiful formula in physics", but the answer to "why" will probably be very confused and/or confusing.

Indeed, what makes it beautiful? Perhaps more importantly, is it beautiful for everybody or could you find someone who will be prepared to say that it is "ugly"? Does it depend on the level of understanding of the formula? And if we talk about beauty in Science, what about beauty in Engineering: could this combination of words be even acceptable, or are they such a dichotomy that putting them together cancel each other out?

In this essay I will attempt to link Beauty to the scientific endeavour in a way that allows engineering in the mix, in a pragmatic and indeed practice-oriented way: I will show that an aesthetic assessment of each own work is, if not pragmatically a necessity, an aspiration to keep in mind during any scientific and engineering-related activity. In order to achieve this goal, I will define Elegance as a precursor to Beauty, which I consider a real requirement for good engineering: this will provide the ground on to which to build the thesis I propose.

Of course, this essay does not purport to appear as an authoritative contribution in the philosophical arena: more humbly, it is a set of reflections and musing by the author. I am indebted for the contribution of a number of people who have read and commented on the work, particularly my long-suffering wife Claudia, who, as an artist, has provided me with deep insights in this field and has provided many valuable comments, and my friend Ugo Concilio. Every responsibility for inaccuracies and misrepresentations remain however with the author.

1.1 How not to close a discussion

This essay was conceived just after a discussion with a manager at a company at which I was employed as a design engineer. We were discussing some circuitry to be included into an ASIC (Application-Specific Integrated Circuit) which was responsible for generating a Sigma-Delta modulated stream which, when low-pass filtered, would provide the necessary voltage for a VCO (Voltage-Controlled Oscillator) to maintain a frequency with a fixed, known fractional relationship with respect to a reference frequency. It was a disarmingly simple device, all digital (apart from the low-pass filter) and very small in terms of silicon area; my task consisted in integrating this device into the rest of the IC. Discussing some possible solutions, I mentioned that one would be very beautiful but would need a bit of time to develop, so I will concentrate on the "rougher" basic solution: his answer to that was, in a very serious voice which was meant to indicate that A Very Important Concept was being imparted upon me: "We Don't Do Beautiful". That exchange remained in my head, and is still there – and I think it is destined to remain there as a "foundation memory". My initial reaction was a vague sense of disappointment: the reason why I do electronics is precisely because I find it

"beautiful"; to be told that it is anything but was quite a blow! I then thought that probably he was trying to hurrying me up with my task, so in the coming months I managed to probe further his statement, and indeed found that it was not a spur-of-the-moment thought, but a real conviction. I question now as I did then the strength of that conviction: can you really become a manager of something you don't find beautiful – in fact he is an excellent manager, and I can't really believe that. But that answer hinted at something which I have experienced over and over again: between "beauty" and quick solutions, in industry we always tend to go for the quick solution, however inelegant (in the traditional sense) that is (well, there are limits there, of course). We never did have a discussion about this: the conversations I tried to start about beauty were always closed in sacrifice to the god of Time. Instead, I went away and did what I like doing: I tried to make sense of what I observed.

2 A Concise Treatise on Beauty

Everyone can easily put forward statements about beauty in paintings, poetry, music, sculpture, regardless of our philosophical abilities: Croce points out in his "Breviario di Estetica"[3] that "common people" could easily make the philosopher flush with in-depth discussion on beauty and art, these two subjects being so close to our hearts. However, relating this to science is more difficult territory: can the scientist, often imagined in the typical Hollywoodesque white coat, crazy hair and nerdy attitude, really produce "beauty"? And what about the engineer: this is often seen as an even less plausible actor in the development of beauty, so engrossed with practical issues to solve to lift his/her head to look at the world around. The problem may stem from the romantic idea of the artist as a "damned hero", so embedded in our society that anything else seems rather incongruous: in my experience, when asked to name an artist, most people would mention artists with troubled life histories, who produced high-impact art seemingly at the expense of their own sanity or health[1].

The relationship between science and beauty has been explored by many philosophers, and has become more relevant with the advances of physics which have opened avenues in science towards aesthetics which were not available before. Arthur I. Miller points out that Einstein in 1905 "introduced aesthetics into modern physics by arguing that the "profound formal distinction" scientists made that particles of electrons emit waves of light was unwarranted [...] his discovery that light could also be a particle emerged from his minimalist aesthetic"[6]. And in mathematics in particular, aesthetics has been part of the discourse from a very early age: Greek philosophers found beauty in the perfection of the simple formulae describing complex natural features. More recently, Dirac famously said that "it is more important to have beauty in one's equations than have them fit the experiments"[4].

[1] It is also my experience that those versed in art history typically make different choices

The first problem is to work out whether beauty and science are compatible. In order to address this point we initially equate art and beauty, by identifying producing art as the act of pursuing beauty[2]. Most thinkers, and indeed a first sight most people, would introduce a necessary distinction between art and science: Croce indicates the necessary contraposition between art-intuition and science-classification, where art has an unconscious quality and science is fully conscious. Indeed, Croce considers the two to be diametrically opposed and incompatible, relating to different "esprits". This distinction is to me artificial and indicates a distance from the scientific pursuit: while experimentalism is indeed far from art when a mechanical repetition of actions (itself necessary to the advancement of science in some – most? – cases), history of science is punctuated with examples of beautiful theories/proofs/insights which are akin, for me, to art works. Examples include Maxwell's equations, quantum mechanics, Fourier transforms.

An additional pitfall which needs to be identified in order to avoid distractions is the distinction which needs to be made between the underlying beauty of the natural phenomenon described by a scientific theory and the beauty of the theory itself[3]. The point here is to distinguish, for example, between the "beauty" of the DNA molecule and the environment around it, which allows life to proceed from such a system, and the theories and experiments which led to its discovery: let us, for instance, consider the DNA molecule "beautiful" from a visual perspective, observing the double-helix configuration as our primary focus; the danger is to consider anything related to DNA similarly beautiful. Confusing the two (the observed and the observation) is disastrous for our discussion, because it will lead us into the traps of believing that science can only be beautiful "by reflection", i.e., like a planet which shines of the reflected light of the sun, science can only be considered beautiful if the matter studied is itself beautiful according to some criteria. Quite apart from being rather disheartening for those scientist who work on "ugly" phenomena (again, according to some aesthetic criteria), it diverts us from our quest, because it hides what is really beautiful about science, which in my opinion transcends the subject matter of the scientific produce.

Another distraction is the identification of criteria for beauty which refer to senses; for instance, using "symmetry" as a criteria confuses the issue because it cannot applied to all scientific produce. I don't mean to limit symmetry to what is visible, as this quality can be applied abstractly to a number of mathematical representations when we extend symmetry to indicate formulae which don't change following rotations and translations; rather, I question the perspective of some to draw the conclusion that "symmetry is a criterion for beauty" be-

[2] This is not necessarily a satisfactory equation, but it will serve as a starting point for the discussion. Indeed, we can use it in this context where we are not concerned with the relationship between art and beauty; thus the equation can be useful to introduce a shortcut between "beauty" and "the pursuit of beauty"

[3] Of course, for "theory" one can substitute any scientific endeavour. I will often refer to "solutions", in a reference to my previous mention of the Nature-puzzle

cause most people would consider symmetrical objects beautiful, confusing the pleasure obtained through a visual experience with a more general idea.

It could be said that as scientific theories are representations of reality, truthfulness is a requirement for beauty; this, however, would draw us back into the trap of believing that beauty can only refer to nature, again confusing the observed with the observation.

So, what is beautiful? Is beauty objective – and whether it is or not, can we identify a set of criteria to draw a line (hard or otherwise) between "beautiful" and "ugly"? As I indicated above, "aesthetic" is a branch of philosophy with a long history (although the term is relatively modern, and appeared in the middle of the 18th century as the title of a work by Baumgarten); as such, many diverse theories have been put forward to identify, precisely, "what is beauty", from the most intuitive idea that "beauty is something which gives you pleasure for the senses", which we classify as "hedonistic", to beauty (or rather art, at one time identified with pursuit of beauty) as expression of the divine (medieval aesthetic, St. Augustine) or the "absolute" (Schelling, Hegel), to the modern perspective which refutes a normative definition of beauty and instead focuses on the piece of art itself. Alas, a proper dissertation on the subject would be beyond the focus of this essay, and will be left to reader to investigate the history of aesthetic further. However, it is important to mark some points in this rich history. The first point to observe is that, while modern aesthetic attempts to refute a normative approach[4], this position is unsatisfactory for our investigation[5]. An observation which we come back to is from Kant: something beautiful appears "purposive without purpose"[6], i.e. it appears to have a purpose, but no specific purpose can be found. This point is important and will be discussed later. A more important contribution to note for our discussion is Denis Dutton's proposal of six universal signatures for human aesthetic[2]:

1. Expertise or virtuosity. Technical artistic skills are cultivated, recognized, and admired
2. Nonutilitarian pleasure. People enjoy art for art's sake, and don't demand that it keep them warm or put food on the table
3. Style. Artistic objects and performances satisfy rules of composition that place them in a recognizable style
4. Criticism. People make a point of judging, appreciating, and interpreting works of art
5. Imitation. With a few important exceptions like abstract painting, works of art simulate experiences of the world
6. Special focus. Art is set aside from ordinary life and made a dramatic focus of experience.

[4] Pareyson for instance proposes that a piece of art is successful according to a norm (or set of norms) defined within the piece itself, and the norm defines the invention and the criteria for the artwork

[5] The reader will forgive me for not pursuing this aspect further in this paper

[6] "Critique of Judgment"

One could easily draw parallels between these signatures and science: "beautiful" science is considered to be "difficult"; "style" could refer to different schools of thoughts on a particular subject; criticism can be found in the peer-review process, for instance; imitation can go in parallel with re-use. As for non-utilitarian pleasure and special focus, although scientific discoveries can be said to be driven by "need", in reality the theory which describes the phenomenon and the experiments devised to investigate nature are, in my view, entirely works of the mind and in themselves can be enjoyed for their own sake and are set aside from ordinary life. Consider for instance the theory according to which dark matter permeates the Universe: assuming it "beautiful", it does not provide me with shelter or food, and indeed is set aside from ordinary life, but it does provide me with a dramatic focus of experience. And indeed we can apply the above signatures to the theory and consider it beautiful. But Dutton's signatures are not the end of the story: being universal they risk encompassing too much. However, the strength of the signatures is that it bridges the objective and subjective parts of beauty: if I cannot appreciate the virtuosity of a piece of art I cannot consider it beautiful; thus, a sense of "responsibility" is required on the part of the beholder and his/her knowledge becomes part of the appreciation of beauty not just as fruition of beauty but also as a determining conscious act.

Karl Popper delivered in 1953[7] a lecture[7] where he attempts to draw a line between science and pseudo-science, and arrives at a set of conclusions on the nature of science[8], which I use as the basis for the bridge between aesthetics and science:

- Science is courageous: it must allow the risk of being refuted
- A good scientific theory is a prohibition
- Irrefutability is a vice of science

The first thing to notice in this list is that it applies mostly to quite revolutionary steps in the progress of science: Popper was extremely impressed by Einstein's proof of General Relativity by Sir Arthur Eddington who organized an expedition to observe the solar eclipse of 1919 and showed that light is indeed bent by gravitational fields as expected by Einstein's theory. Kuhn correctly points out that most science is instead an accretion on existing knowledge: while Popper describes the "romantic hero" of science (see previous section), Kuhn talks about the "busy bees" who make up most of the knowledge by contributing small parts to greater ideas. It appears to me that Popper introduces some sense of aesthetic in the scientific discourse, in particular when he talks about risk. Observe that this position is very different from that of those philosophers who consider science to be the result of mechanical observations thus precluding science from any aesthetic activity for the lack of intuition. Rather, Popper seals the non-mechanical element of intuition and consequently an aesthetic perspective onto the scientific

[7] This lecture was subsequently published in the reference indicated in the bibliography

[8] Popper also indicates additional criteria more related to testability of theories, which I don't think are relevant in the current discussion

progress, warning of the risks of confusing the two aspects, mechanical accretion of knowledge and pure intuition not supported by observation. Kuhn counters that in reality the work of the scientist is far from the beautiful theories, and focuses instead on experiments: "the exploits of Copernicus or Einstein make better reading than those of a Brahe or Lorentz"[1]. Is beauty lost here? No: in my opinion, the aspiration of the scientist is still the beautiful theory, the ground-shifting discovery, the Higgs boson with the "wrong" weight, the Fourier transform. The other observation is that science and art are courageous in similar ways, pushing the boundaries of what's known and exploring the impact of new theories on existing knowledge, imposing distinctions and prohibitions and being open to debate/tests.

3 Beauty in STEM

"Armed" with the basic tools introduced above, we can finally approach Beauty in the STEM subjects (Science, Technology, Engineering and Mathematics) in steps or degrees, and this will be helpful to our final aim. We first look at a subset of what is beautiful, and I will use the term "elegance" for the quality of these instances. I will consider "elegance" a subset of beauty because in my definition something elegant might not be beautiful, while something inelegant cannot be beautiful. Of course, I am "overloading"[9] this term for my purposes here, but the word and its everyday meaning is in line with my aims.

I will use a normative approach for the definition of elegance and beauty in science: I intend these "norms" however more as guidelines than rules.

We can identify a set of criteria for elegance in science:

1. Clarity or effortlessness – an elegant solution, experiment, proof, theory can be understood by those familiar with the context, and the general idea can be grasped by those not familiar. This is the quality of an elegant work to appear as self-evident
2. Generality – it applies not only to the problem at hand, but can be re-used with the necessary modifications in other contexts
3. Control or coverage – an elegant solution covers the problem completely. This does not mean that it does not have limitations, rather that the limitations do not preclude breadth of application
4. Adherence to Occam's Razor – "entities must not be multiplied beyond necessity", the solution is economical and efficient

These four criteria can be applied to science and engineering works at various levels: experiments, theories, but also computer programs or computer languages themselves. I have chosen these criteria from my experience and they are therefore open to criticism (i.e. they risk to be refuted, as a good theory should).

Beauty extends elegance and we introduce the following additional criteria:

[9] A term borrowed from Computing, where a function can be overloaded if it has different semantics based on the number and type of input parameter and output

5. Virtuosism – a beautiful solution resolves a difficult problem, and/or challenges our preconceptions and knowledge
6. Intuition – a beautiful solution hints at something beyond the problem at hand and has far-reaching consequences (an instantiation of Kant's "purposive without purpose")

3.1 The Fourier Transform

As an example of beauty in science, I propose the Fourier transform. This is a mathematical tool which describes an arbitrary function[10] in terms of sinusoids of different frequency and amplitude. It is a remarkable method, and applied to technologies we use in everyday life; it also enables scientific methods and observations which further our understanding of physics, chemistry, astronomy to name a few disciplines. We can apply the criteria defined previously:

1. Clarity - The idea of a signal as composed of sinusoids of different frequencies and amplitudes is surprising to begin with, but becomes simple to relate to when observing a diagram. Most people use a graphic equalizer for their HiFi stereo - a direct application of the Fourier Transform
2. Generality - Fourier discovered this method while working on heat transfer and yet the very same method is used for anything from audio processing to radio communication
3. Control/Coverage - the constraints applied to the method are not a major limitation and the Transform is applied successfully in most applications
4. Occam's Razor - the Fourier Transform is powerful in its "economicity"; consider for instance the ability to deploy a Fast Fourier Transform (FFT), a method which simplifies the traditional Fourier Transform to make it available for deployment in digital computation platforms - or even by hand
5. Virtuosism - The idea of translating a complex, arbitrary waveform into its frequency components is such a divergence from the sensory perception of reality that it challenges our knowledge and preconceived assumptions. Moving from the actual observed light into analysing the colour components, astrophysicists are able to determine the composition of planets' atmospheres and stars, which in turn enables them to infer a history of the celestial bodies
6. Intuition - the "magic" of the Fourier Transform is that it opens a window on a completely different way to observe reality: the Transform itself is a mere mathematical tool, but the implication that every signal we observe (or make devices to observe) is composed of simple repeating waveforms of different amplitude and frequency has a metaphysical quality to it, and inspires us to consider corollaries which bring us far away from the original proposition

3.2 The Phase-Locked Loop

Elegance in Engineering is a more slippery concept to an extent, but as an example consider the Phase-Locked Loop (PLL). This device (which can be developed

[10] The arbitrary nature of the function is in fact limited by a number of constraints

mechanically or electronically) is able to track an incoming waveform and keep the output phase locked to the input phase. It is an important development in technology, enabling for example efficient and affordable modulation and demodulation of radio signals. It is a fundamental building block of many electronic devices, used for instance in clock generation.

Applying the criteria:

1. Clarity - The function of the PLL can be easily explained and is typically understood by undergraduates of engineering courses
2. Generality - The PLL is applied very widely in industry
3. Control/Coverage - The PLL device has very limited caveats in its deployment
4. Occam's Razor - The PLL is "economical" in that it requires limited boundary conditions to operate

However, I contend that the PLL does not completely satisfy the additional two criteria:

5. Virtuosism – This is covered, as the PLL does indeed resolve a complex problem which would require significant effort to resolve with alternative solutions (consider radio modulation and demodulation before the PLL)
6. Intuition - this is the criteria which I believe the PLL does not completely satisfy. Under scrutiny, the PLL operation can be described completely and it does not appear to open up different ways to consider reality

Thus, I propose that the PLL is an Elegant solution rather than Beautiful.

3.3 Beauty and Elegance: Necessity or Indulgence?

We now come to a fundamental tenet of this essay. I contend that "beauty" and "elegance", and the pursuit of these, are not only desirable aspects of the scientific and engineering endeavour; they are *necessary*:

- Beautiful and elegant solutions are clear and easy to understand intuitively. Examples are software coding, electronics design, chemical processes etc.
- This clarity makes portability and enhancements more reliable and predictable
- Such solutions also require less maintenance and additional follow-up work, as they have generality and coverage as defining features
- They are optimal as they adhere to the Occam's Razor rule
- Beautiful scientific solutions have an element of virtuosism and appeal to our intuition, which fills us with awe at the unending complexity of Nature, intended in the widest sense

A suitable metaphor which can be used here equates the scientist to the traditional "artist" and the engineer to the traditional "craftsperson". The artist explores the boundaries and challenges assumptions about aesthetic; the craftsperson exploits and develops ideas to turn them into practical deployments. It is

important to point out that the two are related: the two agents influence each other's work, enabling a complex interplay between artistic élan, technological constraints, business opportunities, customer appreciation. The parallels between the artistic and scientific endeavour are very appropriate to bring the scientific pursuit of beauty and the engineering focus on elegance on terms more closely related to everyday life.

Just as the artist and craftsperson uses and develops the available technology to their pursuits, so do the scientist and the engineer. In this context, the Engineer is a "scientific craftsperson": this agent invents new solutions to technical problems, translates the ideas of the "bohemian researcher" into practical products, provides the bohemian researcher with new and refined technologies to pursue new, beautiful science.

In summary, I consider that what moves us to further knowledge is science is not just our insatiable curiosity, but also a pursuit of aesthetic satisfaction which transcends the senses - in a way, scientific pursuit is *essentially beautiful* in a pure sense. Notice an intriguing parallel with (post-)modern art - a parallel we will not explore further in this paper.

4 Conclusions

Looking back, strictly speaking that (ex-)manager was indeed right: we (engineers) don't do beautiful - instead, we engineers "do elegant". But I don't believe the attitude of considering the idea of "beauty" in engineering with disdain, as an unnecessary distraction from the necessary work at hand, is correct. Introducing "elegance" is a way to introduce aesthetic appreciation in the engineering practice, however, seeing it as a burden, an incidental by-product of the work of the snowed-under engineer is limiting its application. What is the alternative to elegant work? Inelegant work requires regular re-work; re-use and extension of inelegant work require understanding of the original caveats and limitations, hampered by lack of full coverage of a problem; its use outside the strict letter of its manual (itself a work of engineering, thus subject to the same elegance criteria!) can have unexpected results. Thus inelegant work requires unnecessary resources which could be saved by more work on the aesthetic value.

And what about science, and in particular research? I believe that researchers have the privilege of aspiring towards beauty as an everyday endeavour; this privilege should indeed inform the choices and any work undertaken. It is true that a balance needs to be struck between what I call the bohemian aspect of the research and the day-to-day drudgery of literature reviews, computations, analysis etc. However, keeping in mind the higher ideal of Beauty during this work can only help in making the work more appealing and may lead to unexpected developments.

Finally, a necessary digression. I have avoided deliberately mentioning "truth" throughout the essay, but this is indeed an important concept in this context, especially to explain an aspect discussed previously. A pitfall which should be avoided at all costs is to equate beauty and truth: this equation we have already

discounted previously, but apart from the mistake of inferring beauty from truth, an even more dangerous mistake is to infer truth from beauty. Ian Glynn in the epilogue to his book "Elegance in Science"[5] reports an excellent example of this mistake related to the way data is encoded onto the DNA molecule (a warmly recommend read for every scientist). I mentioned before that "scientific pursuit is essentially beautiful in a pure sense": the price to pay for that is that at the end, truth defines the success or otherwise of scientific work. Art has no such shackles and is therefore free to explore beauty - a privilege not afforded to the scientist.

So my closing remarks are that beauty is not simply incidental to science, but a fundamental aspect of it. Engineers are the scientific craftspeople informed by aesthetic just as much as by science. A reassessment of the aesthetic value of the scientific and engineering endeavour would make us more efficient, more effective and, more importantly of all, *happier and more satisfied individuals.*

References

1. *Citicism and the Growth of Knowledge*. Cambridge University Press, Cambridge.
2. *The Routledge Companion to Aesthetics*. Routledge, 2002.
3. B. Croce. *Breviario di Estetica*. Adelphi Edizioni S.p.A., Milano, 2007.
4. P. Dirac. A thing of beauty. *Scientific American*, 1963.
5. I. Glynn. *Elegance in Science*. Oxford University Press, 2010.
6. Arthur I. Miller. A thing of beauty. *New Scientist*, 189(2537), 2006.
7. K. Popper. *Conjectures and Refutations*. Routledge and Kegan Paul, London, 1963.

Every Accomplishment Starts With the Decision to Try

Sohini Dasgupta[1], Praneet Bhatnagar[2]

[1] Imagination Technologies,
[2] Mclaren Applied Technologies

Abstract. A stimulating intellectual environment and access to resources are of paramount importance in creating an ideal learning and research atmosphere conducive to quality scholarly work. Being in a system that rewards academic contributions through publications and funding garners commitment towards the chosen field of research. Inspirational role models help to shape students and thereby enable them to pursue academic disciplines they can excel in. We have been lucky to be mentored/supervised by Prof. Alex Yakovlev during our Masters and PhD degree courses in Newcastle University. Through his work he has shown how satisfying and gratifying research can be once you start on this path. His objective for research made an impression on many professional careers towards a more fulfilling objective in life rather than just material and personal gain. In developing countries learning is always a goal to achieve personal gains rather than actually understanding and spreading the knowledge. In this paper, we will aim to provide an Indian student's perspective on learning and research in India and how it compares to Britain and how role models like Alex helped shape us into what we are today.

1 Introduction

As a student, coming to study in a different country has a lot of excitement associated with it. The excitement of living in a new society, with people from various countries and culture, is foremost for new students. Most students find this experience, a background learning process which enhances their appreciation of other cultures. The first hand exposure of this new society is something which cannot be experienced through portrayals in British dramas and movies. However, along with the excitement there is a sense of apprehension regarding the challenges to be encountered in a new education system. Education system for higher degrees in the UK universities is operated in a very professional manner. The way universities are set up makes the initial experience a very welcoming one. Even in a one year course the initial days are spent just enjoying the whole university experience rather than immersing the students in a classroom. Once the taught modules begin, the lecturer and students exchange starts at a very personal level. Most lecturers practice an open door policy to the students and are ready to share their experiences. In this paper we try to highlight the key advantages that we experienced in the UK University over our experience in an Indian University.

2 Experience

Our first-hand experience of being taught by Prof Alex Yakovlev was during our Masters programme. We both were coming from an education background in India which had a different approach to teaching compared to our experience working with Alex Yakovlev. His teaching style was more a subjective approach as compared to having an objective approach. This was a refreshing change where the details of each concept were analysed and explained for understanding of the students. This meant the involvement of the student in getting curious about learning. This also enabled students to extrapolate from established theories and be creative about the subject.

Owing to Alex, our university education became the beginning of our un-schooling. This is where we began to love learning and not lose ourselves in the production line. Much of what we feel are the weaknesses of the Indian education system comes from the fact that emphasis is solely given on a student's grade when considering admission to a particular field of study. In the UK, lecturers like Alex, look beyond grades and identify the potential of student through their skills, knowledge, aptitude and aspirations.

One of the major things that alienated the school education system was the expectation that students can memorise large amounts information in order to reproduce these during examinations. On the contrary, in the UK, lecturers pushed students to think and know that there can be two right answers to a question.

Another key strength of UK education system is the emphasis on original research and publication. Many University departments in India have poor government and corporate funding which hinders student participation in workshops and conferences. During our tenure at Newcastle University we have always been encouraged to attend workshops, seminars and conferences which helped us bounce off ideas against each other and helped us grow as a researcher.

3 Conclusion

The development of a discipline in any country is closely related with the overall economic and political stability. This has a deep impact on the available resources to invest in higher education and the resulting infrastructure within universities and research institutions. Although the Indian education system had its shortcomings we feel we still benefited from it in many other ways. The Indian education system provided us with good theoretical background which enables us to pursue higher education in UK. The competitive nature of the Indian education system also enabled us to adapt quickly to the education system in UK. We are enriched in our experiences due to our university experience in UK. Our experiences in two different worlds have made us better professionals.

This paper is dedicated to Prof Alex Yakovlev and how his teaching and mentoring style has left a deep imprint on our lives. We feel his teaching style has had an impact on many students before us and will continue to have on many students in the future.

The Story of the Amulet: A Brief History of Asynchronous Events in Manchester

Doug Edwards, Will Toms, Steve Temple, Luis Plana, Jim Garside and Steve Furber

University of Manchester

Abstract. This presents an overview of the significant achievements in Asynchronous Logic over the last quarter of a century at the University of Manchester. Including the Amulet (and subsequent) asynchronous Microprocessors, the synthesis tools Balsa and Teak, the MARBLE and Chain on-chip Interconnects, together with research arising from external collaborations with the University of Newcastle.

1 Pre-history - an Overview by Steve Furber

The Amulet story begins in 1989. Through the 1980s I (SBF) worked at Acorn Computers Ltd in Cambridge, UK, where I was a designer of the BBC Microcomputer and the ARM (then "Acorn RISC Machine") 32-bit RISC microprocessor. We didn't publish much, but one paper we published was on the ARM3 - the first ARM with an on-chip cache - at VLSI'89 in Munich [7]. At that conference I was struck by a paper [14] presented by Craig Mudge, then CEO of Austek Microsystems Ltd based in Adelaide. This was my first exposure to asynchronous design, and of course Mudge's paper cited Ivan Sutherland's Turing Award paper [19] which really sold the concept to me.

Over the following year I doodled some ideas on the possible design of an asynchronous version of the ARM based on Sutherland's micropipelines approach. This was around the time the EU launched the OMI - Open Microprocessor systems Initiative - which was in part stimulated by input from Acorn. When I accepted the ICL Chair at Manchester, to which I moved on 1st August 1990, Acorn was generous in allowing me to take with me to Manchester a part of an early OMI project that I had been involved in developing at Acorn - the OMI-MAP project. OMI-MAP provided the early funding that got the Amulet (asynchronous ARM) research off the ground at Manchester. The University allocated a lectureship appointment linked to my chair, to which Jim Garside was appointed, and OMI-MAP funded the appointment of Nigel Paver and Paul Day as post-docs. In addition, existing academic staff joined in - Doug Edwards, Linda Brackenbury and Viv Woods, bringing in additional perspectives and new funding, all of which enabled the Amulet group to grow to critical mass.

Why "Amulet"? Well, in searching for a group or project name you have to start somewhere, and writing down keywords on a whiteboard is one way to start. "Asynchronous", "Manchester University","Low-Energy Technology"?

The name stuck and worked well, even though the acronym expansion did not see much light-of-day!

2 Amulet Processor Series

Before engineers had learned the "correct way" to build digital circuits - by inventing a global clock to slow everything down - there was considerable experimentation with different design styles. However to attempt anything complex in a clockless circuit was clearly a bad idea. Was it even possible to challenge the decades of synchronous development with such a radical alternative?

Few people had tried asynchronous VLSI, largely because the two terms are from different eras; asynchronous digital circuits largely predated VLSI. However, conventional devices were exhibiting some worrying trends, particularly increasing power consumption in parallel with the rise of mobile computing devices and shedding the clock seemed like a way to lose one of the major drains on a battery, particularly as the clock waggled continuously but did no actual computing.

Requiring a demonstrator, a microprocessor seemed an appropriate size. Regarding an architecture, a custom ISA was already being addressed at Caltech [13] so cloning a commercial ISA seemed to pose some additional - and well-defined - challenges. Respecting Steve Furber's background, and arguably influenced by already being a low-power champion - the demonstrator ended up as an ARM.

Thus was born Amulet 1, an attempt to produce an ARM7 including everything but the clock. With respect to the literature, particularly Sutherland's 'Micropipelines' paper [19] this was done using transition (two-phase) signalling and lots of "exotic" standard cells such as a variety of Muller C-elements [15]. Time elapsed and the the device appeared, made largely 'by hand' on a 1 µm 2LM process. Happily it was functional and could run standard ARM code with various asynchronous features such as varying execution speed as the supply voltage was changed. Rather less happily it did this at about half the speed of a contemporary ARM7, despite managing a 'CPI' figure of 0.0. On the plus side, energetically it was at least comparable with the synchronous circuit; as a minus the transition signalling proved extremely painful to interface to.

Somewhat undaunted, Amulet 2 followed, hopefully incorporating the lessons learnt. This involved going "four phase" to simplify the circuit design and incorporating more features - notably a cache memory (operating asynchronously, of course) on chip. This achieved its objectives in that when the chip arrived it worked and was a lot easier to use. It was also somewhat faster; unfortunately so were the contemporary synchronous ARMs though which meant it was still about half their speed. However there was another interesting observation made which was that the lack of temporal coherence reduced the electromagnetic noise emitted by the system.

It was this last characteristic which was a tipping point in having a third attempt at the architecture, named, unsurprisingly, Amulet3. This was done in

parallel with Hagenuk GmbH for integration onto Draco - a Digital Radio Controller. This time only half the chip (the Manchester half) was clockless but this included the processor, memories, DMAC etc. all communicating across MARBLE [2], an asynchronous bus. This was now to compete with the ARM9 and more features - such as branch prediction and out-of-order, parallel execution - were incorporated into the microarchitecture. The goal was to achieve a commercially available system before the year 2000. The chip was manufactured in time; unfortunately also in time for Hagenuk to go out of this business. However the technical design was vindicated in that this would run at effectively the same speed as an ARM9 manufactured on the same process. Honours on energy use was about even; the EMI was startlingly low.

Amulet 4 may have addressed more sophisticated architectural issues such as superscalar execution ... but it never happened. In the synchronous world new techniques - such as gating the annoying, power-consuming clock - were being adopted and it was becoming apparent that the potential advantages of future asynchronous processors were unlikely to be great enough to cause a major switch in the industry. However the chips did demonstrate that most, perhaps all, synchronous microarchitecture can be duplicated competitively without a clock.

3 SPA - A Synthesised Amulet

The Amulet processors followed the design style used in the early, synchronous ARM hard macrocells - full-custom datapaths with standard cell control blocks. A different approach was required for the next collaborative project, which investigated the ability of asynchronous systems to offer improved security through increased resistance to non-invasive attacks on smartcards, such as power and timing analysis. A new processor, based on a more secure asynchronous technology, was needed and there was no time to design it in the rather laborious, full-custom style used previously. Balsa had proved its value in the Amulet3-based DRACO chip and presented the group with the opportunity to produce a synthesised, asynchronous, ARM-compatible core, named SPA [16]. In fact, a complete asynchronous smartcard System-on-Chip, shown in figure 1, was synthesised using Balsa. The only clock used in the system is the smartcard interface clock, driven by the smartcard reader. All the components of the system-on-chip were connected through CHAIN, the on-chip interconnect technology also developed at Manchester. Balsa was used to describe and synthesise the system-on-chip, which incorporated synchronous memories with asynchronous wrappers. Cadence CAD tools were used to implement the chip in TSMC 180nm technology. The chip, shown in figure 2, occupied an area of approximately 33 mm^2. Prototypes were received from the manufacturer in October 2002 and, after a connection in the memory wrapper was repaired using FIB technology, were fully functional. SPA required a completely different design approach from the Amulets. The strategy to make SPA robust against power and timing attacks was to make sure that all operations consumed the same energy and took the same

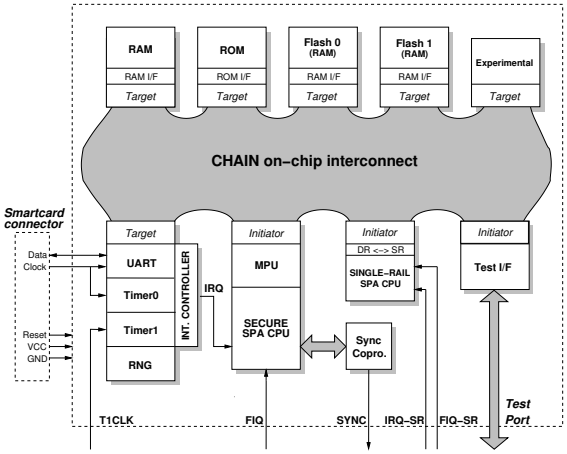


Fig. 1. SPA Block Diagram

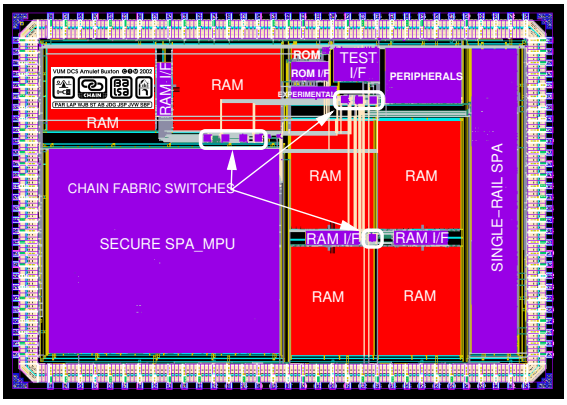


Fig. 2. SPA Chip Plot

time, irrespective of the actual data. This went against a basic asynchronous designer's strategy to take advantage of average case performance. The only sensible way to achieve this was to design the processor to operate in worst-case time and energy consumption. SPA was a success from the point of view of security but was distinguished with the unenviable accolade of being the slowest ARM ever.

4 Latch Controllers

Amulet1 was an implementation of the ARM based pretty faithfully on Sutherland's micropipelines style. However, we found the two-phase control pretty messy and slow for complex control structures, so for Amulet2 we decided to try four-phase micropipelines - a very similar bundled-data approach but with return-to-zero (RTZ) signalling. After some early exploratory work [6] we decided that the number of options in terms of the interleaving of the two handshakes was too complex to do entirely by hand, so with a little help from Alex we adopted an STG approach to devise a range of solutions with performance/complexity trade-offs. The paper published in IEEE Trans.VLSI [8] remains Steve Furber's most cited paper of all time, with just under 300 citations to date pretty evenly spread over the last 20 years! Further developments optimised the controllers for pipelines using dynamic logic [10]. Steve pulled a lot of this together in the unpublished (though still cited!) "A small compendium of 4-phase micropipeline latch control circuits", which has kept Graham Birtwistle and Ken Stevens (constructively) amused for many years since exploring the outer limits of this space.

As our designs became more complex, the need to use formal tools to get things right increased. On Amulet3 [9] we acknowledge extensive use of Alex's Petrify tool in the design. Although a wide range of latch controllers are used in the design, the most terrifying aspect of it was the register reorder buffer. Here Jim Garside came up with an initial circuit by hand, but no-one else could understand it, so Steve used Petrify to verify Jim's design. This turned out to be non-trivial - specifying an STG with as much concurrency as Jim's brain was a very challenging and incremental process! But in the end, with a little help from Alex, we got there. Petrify proved Jim's design correct, and even managed to remove one transistor from it!

5 AntiTokens and Wagging Logic

While working in the optimisation of dual-rail self-timed logic, Charles Brej formulated the idea of "anti-tokens" [5]. To fully exploit the benefits of "early-evaluation" (or "Or-causality") anti-tokens are sent in the opposite direction to the flow of data to destroy data that is no longer required. The concept was subsequently adopted by Cortadella et al for use in Elastic Systems.

Charles also developed the "Wagging-Logic" style where logic-blocks are replicated (into slices) and each successive datum are applied to successive blocks in a round-robin fashion. This allows the return-to-zero phase of a slice to be

overlapped with the data-phase of the succeeding slice. By extending the concept to the units of a Microprocessor, data forwarding occurs between slices, reducing the interlocking penalty and maximising the throughput of independent instructions. This design style cumulated in the Utopium, an 11-way wagging 8051 microcontroller. Sceptical comments from reviewers were etched into the top layer metal (figure 3)!

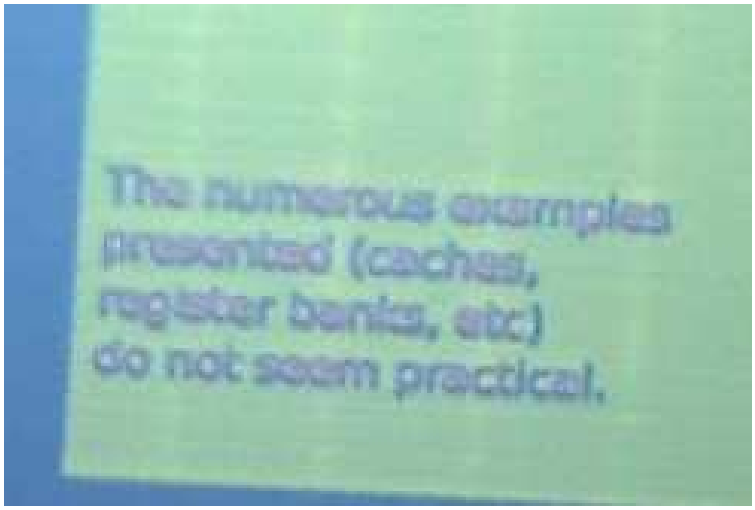


Fig. 3. Utopium Die Plot

6 Balsa and Teak

As part of the EU funded project, EXACT, we were introduced to Tangram, developed by Philips Research Labs, wihich was a CSP-based tool using handshake circuits to synthesise complex asynchronous hardware. it was clear that Tangram was likely to be extremely useful for the design of the systems that the Amulet group might be interested in building. However, it had a number of disadvantages: the language was unfamiliar to most engineers; more importantly, it was proprietary software and was not possible to experiment with language extensions and optimisations and it was not clear if Manchester would be able to use it after the end of EXACT project and. Doug Edwards, who led the Manchester effort on EXACT, persuaded Andrew Bardsley, then a final year student, to become interested in the idea. Andrew produced a prototype for his prize-winning project and then produced a complete working system, Balsa [4], for his M.Phil.

Balsa was used to design the DMA controller for Amulet3. This was a matter of necessity: for various reasons, it would not have been possible for a hand-

designed controller to meet the tape-out deadline; the design that Andrew produced in short order was a convincing proof of the effectiveness and reliability of the tool. Balsa was then used to successfully design a complete processor - SPA by an engineer, Peter Riocreux, who had no previous experience of the Balsa paradigm. SPA has a reputation of being slow, but this is in part due to the design requirements of the processor.

Although there were good reasons for SPA's performance, strenuous efforts were made by Andrew and Luis Tarazona to improve the speed of Balsa generated circuits resulting in an improvement of more than an order of magnitude. Nevertheless, performance could not match that achieved by a conventional synchronous implementation. Attention was directed towards generating data-driven circuits rather than the control-driven circuits of Balsa. Sam Taylor's system, based on Balsa, gave promising results, but unfortunately existing circuit descriptions could not be automatically translated into his new language.

Andrew then began developing Teak which is capable of transforming the timeless concurrent specifications in the CSP-based Balsa language into a data-driven network. The Teak generated dataflows enjoy a set of architectural properties in communication and computation including slack elasticity, distributed control and data-driven behaviour which pave the way for further optimisations such as retiming and re-synthesis. These all make Teak a practical EDA framework in the asynchronous domain suitable for exploring fine-grained elasticity toward tackling the energy issue in large-scale SoCs

Recently eTeak has been introduced to enable the synchronous designers to exploit the powerful properties of the Teak networks including scalability. eTeak adopts a synchronous library (specifically the synchronous elastic protocol) to introduce a common timing discipline to the asynchronous dataflows of Teak [12]. This way clocked commercial EDA is employable for retiming and resynthesis in the synchronous domain. Latest explorations toward automatic GALS synthesis leverage these advantages, particularly retiming of eTeak circuits, to study the impact of the fine-grained partitioning on performance.

Balsa research has had many points of intersection with Alex Yakovlev's group at Newcastle, from STG specification of handshake circuit to collaboration on funded research projects such as GAELS, SEDATE, VERDAD. It has been used as a teaching and research tool in many institutions globally.

7 Asynchronous Interconnect

The Manchester AsynchRonous Bus for Low Energy (MARBLE) [2] was developed by John Bainbridge for use in the Amulet3H system. MARBLE was a dual-channel pipelined bus with centralised arbitration and address decoding, using an asynchronous four-phase bundled data protocol. The interfaces (figure 4) were specified using STG's and speed-independent implementations were synthesised using Newcastle University's Petrify tool, except for two signalling modules in which timing assumptions were unavoidable. John was awarded the BCS Distinguished Dissertation award for his PhD based on MARBLE.

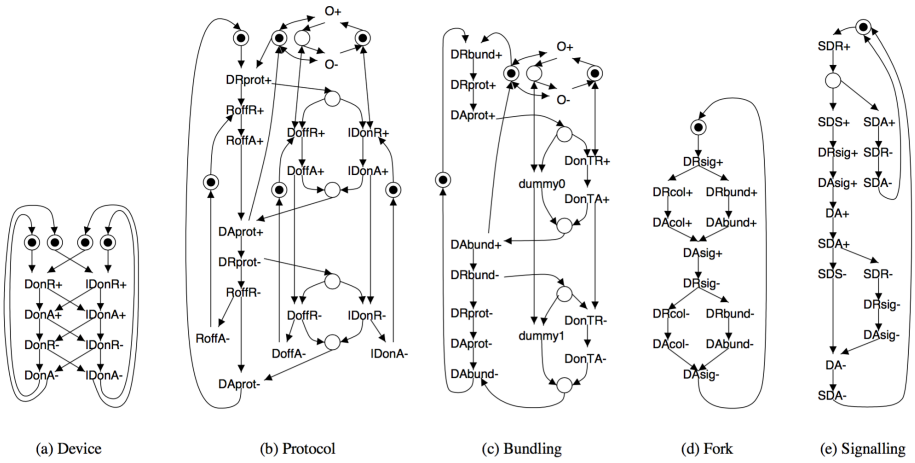


Fig. 4. MARBLE Interface STGs

To overcome the limitations of a shared bus, MARBLE was re-implemented as a packet-switched Network-on-Chip, using a Delay-Insensitive 1-of-4 data encoding [3]. This was further developed in to the CHip Area INterconnect (CHAIN) Network fabric [1] using a range of incomplete m-of-n codes designed to minimise encoding and decoding overhead. The CHAIN Network became the core of Silistix, which produced a range of AMBA and AXI compatible asynchronous network-on-chip solutions.

8 SpiNNaker

In addition to leading the research into asynchronous systems, in the late 1990s Steve Furber also became actively interested in how the brain functions. This was to lead to the SpiNNaker project [11] which received its first funding in 2006 and has continued to the present time. SpiNNaker is based around a multi-core processing chip [17] which has a novel routing system for small packets (40 or 72 bits) which represent the spikes emitted by real neurons when they fire. The chips are designed to tile together to create a massively parallel compute engine for simulating networks of spiking neurons. Currently, a 500,000 core machine has been constructed, which consumes around 40kW when running flat out. It is planned to extend this machine to 1 million cores.

It was clear from the outset that the design effort in making the SpiNNaker chip required the use of as much pre-existing IP as possible. Our Amulet cores were not readily process-portable so in collaboration with ARM we obtained processor and memory controller IP (synchronous, of course) and from Silistix, which was spun out from the group to commercialise the CHAIN system, we obtained asynchronous networking IP. This meant that we could concentrate our

efforts on the novel IP required for SpiNNaker which was based around a custom router which facilitated efficient multicast routing of the small packets used to represent neural spikes. The chip was built as a GALS system which meant that we could harden the various synchronous IP blocks and obtain timing closure on them relatively easily, while the GALS architecture meant that timing closure at the top level was much easier than a fully synchronous system of similar size.

So while SpiNNaker generally uses synchronous IP there is a significant asynchronous element to the design. The chip has two independent asynchronous networks (figure 5). The System NoC is the main system bus for all of the cores, on-chip peripherals and memories. It was generated using Silistix's tools and based around multiple 3-of-6 RTZ channels running in parallel to achieve the necessary bandwidth, which is of the order of 1 Gbyte/s. Silistix provided synchronous interface IP for this network which presented ARM AMBA interfaces (AXI, AHB and APB) to the synchronous IP blocks. The second asynchronous

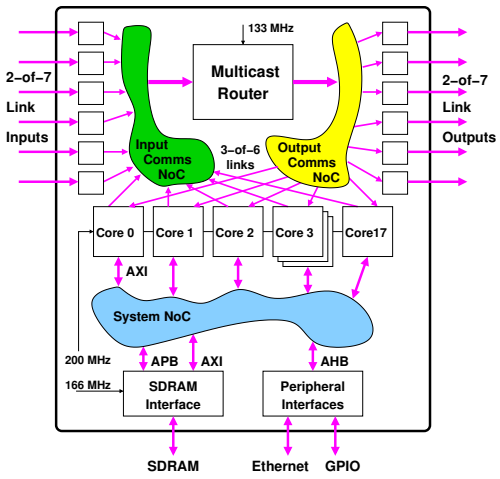


Fig. 5. SpiNNaker asynchronous NoCs

network, the Communications NoC, is used to carry spike packets around the chip both between cores and the router and between the router and the off-chip packet interfaces which transport packets between chips. The Comms. NoC links are single 3-of-6 RTZ channels as the bandwidth requirement for packets is relatively modest. To carry packets from chip to chip, we convert 3-of-6 to an asynchronous interface based on 2-of-7 NRZ signalling. This only requires 3 transitions (two data and acknowledge) to convey data as a 4-bit flit and provides a very low-power interconnect, albeit at the cost of 8 pins per channel. The bandwidth across these links is around 250 Mbit/s and significant design effort was

expended in making them resistant to glitching (and therefore deadlock-free) as they may span significant distances on a PCB [18].

Two chips were fabricated in a UMC 130nm process through Europractice. A prototype (MPW) chip in 2009 was followed by the production chip in 2011. The production chip figure 6 is approximately 10x10mm and houses 18 ARM968 processing cores each with 96 Kbytes of local RAM. The packet router is in the centre of the die and there is also an SDRAM interface block at the left hand side which interfaces to a 128 Mbyte LPDDR memory die which is housed in the same package as the SpiNNaker die. As some of the on-chip asynchronous links span considerable distances on the die, pipelined repeaters were inserted every 0.5mm to maintain throughput. The System NoC is laid out as a sea of gates surrounding the router. Standard synchronous CAD tools were used throughout the design with a minimum of manual intervention to maintain correct operation of the asynchronous components. An estimated 30 man-years went into the

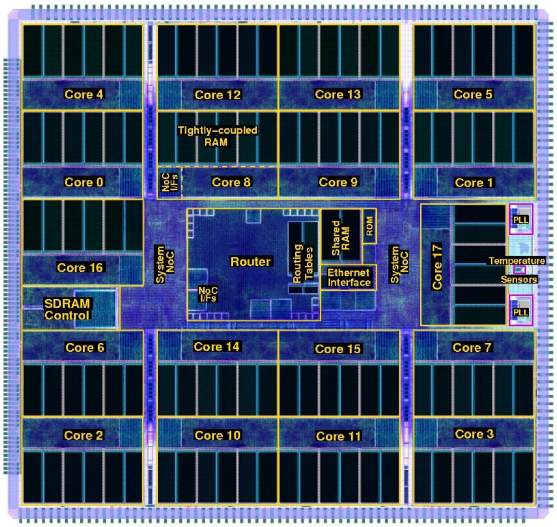


Fig. 6. SpiNNaker chip die plot

design of the SpiNNaker chip, a figure which will soon seem insignificant when the associated software effort is calculated!

References

1. Bainbridge, J., Furber, S.: Chain: a delay-insensitive chip area interconnect. IEEE Micro (5), 16–23 (2002)
2. Bainbridge, W., Furber, S.B.: Asynchronous macrocell interconnect using MARBLE. In: Advanced Research in Asynchronous Circuits and Systems, 1998. Proceedings. 1998 Fourth International Symposium on. pp. 122–132. IEEE (1998)

3. Bainbridge, W., Furber, S.B.: Delay insensitive system-on-chip interconnect using 1-of-4 data encoding. In: Asynchronus Circuits and Systems, 2001. ASYNC 2001. Seventh International Symposium on. pp. 118–126. IEEE (2001)
4. Bardsley, A., Edwards, D.: The Balsa asynchronous circuit synthesis system. In: Proceedings FDL 2000. pp. 37–44 (2000)
5. Brej, C., Garside, J.: Early output logic using anti-tokens. In: International Workshop on Logic Synthesis. pp. 302–309 (2003)
6. Day, P., Woods, J.V.: Investigation into micropipeline latch design styles. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 3(2), 264–272 (1995)
7. Furber, S., Thomas, A., Oldham, H., Howaid, D., Urquhart, J., Wilson, A.: ARM3-32b RISC processor with 4kbyte on-chip cache. In: Proceedings IFIP VLSI'89 international conference. vol. 10, pp. 35–44 (1989)
8. Furber, S.B., Day, P.: Four-phase micropipeline latch control circuits. IEEE Transactions on Very Large Scale Integration(VLSI) Systems 4(2), 247–253 (1996)
9. Furber, S.B., Edwards, D.A., Garside, J.D.: AMULET3: a 100 MIPS asynchronous embedded processor. In: Computer Design, 2000. Proceedings. 2000 International Conference on. pp. 329–334. IEEE (2000)
10. Furber, S.B., Liu, J.: Dynamic logic in four-phase micropipelines. In: Advanced Research in Asynchronous Circuits and Systems, 1996. Proceedings., Second International Symposium on. pp. 11–16. IEEE (1996)
11. Furber, S.B., Galluppi, F., Temple, S., Plana, L.A.: The SpiNNaker project. Proceedings of the IEEE 102(5), 652–665 (2014)
12. Mamaghani, M.J., Garside, J., Edwards., D.: De-elastisation: from asynchronous dataflows to synchronous circuits. In: Proceedings of the 2015 Design, Automation and Test in Europe. EDA Consortium (2015)
13. Martin, A.J.: The Design of a Delay-Insensitive Microprocessor: An Example of Circuit Synthesis by Program Transformation. In: Leeser, M., Brown, G. (eds.) Hardware Specification, Verification and Synthesis: Mathematical Aspects. vol. 408, pp. 244–259 (1989)
14. Mudge, J.C.: An Illustration of Micropipelines using Two-Dimensional Fourier Transform Architectures . In: Musgrave, G., Lauther, U. (eds.) Proceedings of VLSI 89. pp. 359–368 (1989)
15. Muller, D.E.: Asynchronous Logics and Application to Information Processing. In: Symposium on the Application of Switching Theory to Space Technology. Stanford University Press (1962)
16. Plana, L., Riocreux, P., Bainbridge, W., Bardsley, A., Garside, J., Temple, S.: SPA-a synthesisable Amulet core for smartcard applications. In: Asynchronous Circuits and Systems, 2002. Proceedings. Eighth International Symposium on. pp. 201–210. IEEE (2002)
17. Plana, L.A., Clark, D., Davidson, S., Furber, S., Garside, J., Painkras, E., Pepper, J., Temple, S., Bainbridge, J.: SpiNNaker: Design and Implementation of a GALS Multicore System-on-Chip. J. Emerg. Technol. Comput. Syst. 7(4), 17:1–17:18 (Dec 2011), http://doi.acm.org/10.1145/2043643.2043647
18. Shi, Y., Furber, S.B., Garside, J., Plana, L.A.: Fault tolerant Delay Insensitive Inter-Chip Communication. In: 2009 15th IEEE Symposium on Asynchronous Circuits and Systems. pp. 77–84. IEEE (2009)
19. Sutherland, I.E.: Micropipelines. Communications of the ACM 32(6), 720–738 (1989)

Partially-Ordered Event-Triggered Systems (POETS)

Steve Furber[1], Andrew Brown[2]

1: School of Computer Science, University of Manchester, M13 9PL
2: Department of Electronics & Computer Science, University of Southampton, SO17 1BJ

Abstract: Event-triggered computing systems have long formed the basis of real-time embedded systems in industrial plant control, automotive and aerospace system, to name but a few. Each of these application domains comes with its own challenges, but - generalising wildly - there are numerically few inputs and the timing constraints are such that the system can be realised with typically a few cores. Within the last decade, technology has moved on to the point where "the core" - once a central and important component in any computing system - has become as commoditised as the transistor did forty years ago. They have become negligibly cheap, and this change of value has brought with it a change of design perspective: In the past, complex data structures had to be constructed to allow a machine to operate efficiently on large numbers of "problem components", and if the resource was insufficient, multiple cores would be bought into play, their interactions choreographed explicitly by expert software architects. It is now possible to create systems where the atomic elements of a datastructure are spread evenly and thinly over a huge number of small, simple cores, and the necessary computations executed by *cores local to the data*, rather than moving the data to the cores. In this paper we discuss realisations of this idea: the SpiNNaker engine, a custom system designed to simulate the behaviour of a billion mammalian neurons in real time - a feat made possible by a bespoke communications infrastructure, asynchronously and independently transporting tiny packets of information; we then go on to generalise the concept and describe the POETS computing system, which allows a far greater range of application domains to be addressed than does SpiNNaker.

The model

Introduction

Here we begin to develop a formal system model that can be used to describe the operation of biological neural systems (such as the brain) and computational models of such systems.

This work is motivated by a desire to find useful ways to think about information processing in the brain, and by a desire to produce a formal semantics that can underpin reliable operation of event-based machines.

We introduce three models at different levels of abstraction, progressing from the biology of neural systems down to the details of the SpiNNaker machine.

A hybrid-system model

The system is a set of dynamical processes $P = \{P_i\}$ that communicate purely though event communications using a set of event channels $E = \{E_j\}$.

Each dynamical process evolves in time under the influence of received events, so $P_i = p_i(t, E)$, where p_i is a function that may have internal state and t is time. Typically P_i will depend on a subset of the event channels, not all of them.

Each event channel carries events that are either generated by a process, or come from the environment, to all the other processes that depend on them. So for an internal event channel $E_i = e_i(P_j)$ for some j. Normally an "event" is a pure asynchronous event that carries no information other than that is has occurred, so it can be thought of as a time series of identical impulses:

$$E_i = \sum_k \delta(t - t_k),$$

where t_k are the times the events occur on this channel. Sometimes it might prove useful to be able to modulate the size of the impulse.

We can consider the event channel to be instantaneous, so that events arrive at all of their destinations at the same time that they are generated by their source process, though causality allows us to view this as "after" they are generated, albeit by a vanishingly small delay. Likewise, if an incoming event causes a process to generate an outgoing event this causality is captured by the output being "after" the input.

The outputs from the model are simply a subset of the total set of events, E.

Biological neurons

Biological neurons are complex living cells that have a cell body (the *soma*), a single output (the *axon*) that carries action potentials, and a complex multi-branched input structure (*dendrites*) that collect inputs. The axon from one neuron couples to the dendrite of another through a *synapse*, which is a complex adaptive component in its own right.

Action potentials are sustained and propagated by electro-chemical processes in the axon that allow them to be viewed as pure asynchronous events.

Long axons incur significant delays, but these can be rolled into the transmitting and/or receiving process. Where there are different delays from a single source to different targets, for example a short delay to proximal targets and a long delay to distal targets, the hybrid-system model allows this to be captured either by different delays in the receiving process or by the source transmitting separate events with different source delays, or some combination of these.

We therefore claim that the hybrid-system model captures the essential features of biological neurons that exchange information principally through action potentials.

Action potentials are not the whole story, however. Some neurons produce chemical messages, for example dopamine, that modulate the activity of other neurons within a physical region. Some neurons make analogue dendritic connections with their neighbours. These phenomena are outside the hybrid-system model, but we hope that their principal effects can be captured through back-channel processes of some sort.

In addition, biological systems do not have static connectivity – they develop and grow, gaining and losing neurons and connections to their "event channels". These happen slowly relative to the real-time information flow, and such dynamic topology changes many be modelled through back-channel processes.

Biological systems are also very noisy, but we can accommodate this by using noisy processes.

An abstract computational model

We cannot compute a continuous process exactly as in the hybrid-system model, so for efficiency it is important to approximate the process in some way. Most neuron models are some form of system of differential equations, so it is common practice to compute these using a form of integration over discrete time-steps.

For *real-time* modeling, the integration can be implemented by introducing an additional "time-step" event, E_t. Now time is just another, regular (e.g. 1ms) event, from an external source, and t can be removed from the model.

We can, at least in principle if our computer is sufficiently fast, ignore the time taken for a process to handle an event. Each event is handled as it arrives, and each process is simply a set of rules defining how that process's state is changed by every possible input event. Thus:

$$P_i = [E_j \Rightarrow S_i \leftarrow p_i(S_i, j)] \forall j : E_j \in I_i$$

where S_i is the state of P_i and I_i is the set of events, now including E_t, that are inputs to P_i. This is the **event-triggered** aspect of POETS.

It is clear that a process is active only in response to an input event, and therefore any output events it generates must also occur at the same time as (though causally after) an input event. Note that this does not preclude an internal time delay between a neural input and the output it causes: the input can change the state of the process, which then progresses through several time-step events before producing an output. But the output will eventually be produced in response to, and at the same time as, a time-step event. As the only representation of time in the system is the time-step event, time is discretized.

Since the time-step event, E_t, connects to many (if not all) processes, there may be many events generated just after it. These events, from different processes, have no implicit order. This gives rise to the **partially-ordered** aspect of POETS. Each process to which some of these concurrent events are inputs will impose an arbitrary order on their reception (at notionally the same time), and as a consequence the system behaviour is non-deterministic at this point.

A SpiNNaker computational model

SpiNNaker is a massively-parallel system with an interconnect fabric designed specifically to convey events generated by a program running on one processor to all of the processors to which that event is an input. The SpiNNaker fabric must initially be configured to put the necessary connections in place, but once so configured the

hardware looks after the event connections. Processors then receive events intended for them and issue events with no knowledge of where they are destined to go.

Unfortunately the processors on SpiNNaker aren't infinitely fast, so a process takes a finite time to complete its response to an input event. While it is running another event may arrive, demanding pre-emption. The time-step event may not be synchronized across the machine (although near synchronization is possible using a technique such as fire-fly synchronization).

A further complication is that SpiNNaker processors keep some of their state in off-chip SDRAM, access to which incurs high latency costs. In general we aim to hide this latency by exploiting a DMA subsystems attached to each processor to handle SDRAM transfers while the processor gets on with other stuff.

These (and other) niceties apart, SpiNNaker aims to implement the abstract computational model as faithfully as it can, subject to all of the constraints of the physical system, delivering a reasonably efficient solution, and minimizing energy consumption.

SpiNNaker models may attempt to implement the abstract computational model faithfully, in which case they will aim to synchronize the (notional) 1ms time-step across the machine and complete all the work in every 1ms to stay in lock-step across the machine. In this case the ***peak*** process load must complete within the 1ms for correct operation. Alternatively, they may adopt an asynchronous model where there is no attempt to align a 1ms period in one process with that in another, in which case the ***average*** process load must complete within 1ms for correct operation.

Spiking neurons on SpiNNaker

Each processor on a SpiNNaker machine handles one process, where each process models a number of neurons. As incoming events from other processes are very similar they are handled by one event handler. The simplest model of a SpiNNaker process then handles two event types:

1. Incoming neuron event: locate and process synaptic data, updating local neural state accordingly.
2. Time-step event: perform integration step for all local neurons, possibly generating outgoing events.

As an implementation detail the neuron event handler will usually invoke a DMA transfer to bring the synaptic connectivity data in from SDRAM, but as this is internal to the process we hope to hide the DMA as much as possible from the application code.

This model does not handle the important aspect of synaptic plasticity, but already creates some interesting data consistency issues if a type 2 event occurs while the (fairly long) event 1 process is running and pre-empts it. These data consistency issues are avoided if no input is allowed to affect state that is used in the current time step, which amounts to imposing a minimum axonal delay of 1ms.

In general a SpiNNaker implementation uses a very simple real-time kernel of some sort, with drivers for the event communication system, DMA, etc. It includes

queue management, priority scheduling, buffer overflow procedures, and so on. This notwithstanding, it maintains a strongly event-driven nature, spending any idle time in a low-power wait-for-interrupt state.

What can a formal model offer?

A validated formal model can answer various important questions about the SpiNNaker system. At the low level:

- Is the run-time software a robust implementation of the computational model?

And at the high level:

- Is system activity at a stable level, or will it grow uncontrollably (as in epilepsy) or die away?
- How does the processing of neural information through successive layers "add value"? For example, in vision we start from pixels, which are processed (in the retina) into centre-surround signals. The primary part of the visual cortex is known as V1, (**V**isual area **one**), which processes these signals into edge/–corner/orientation, and up through further layers into "car" or "tiger". (Both hemispheres of the brain contain visual cortex, one for each visual field.) How can we quantify the benefits of each layer, preferably in information-theoretic terms?

Generalising....

Changing the narrative perspective significantly and moving it back out, Moore's Law has given us a doubling of logic density every eighteen months or so for over four decades. It has enabled microelectronics to move from a narrow professional niche into the hands and pockets of every consumer in the world. However, as process geometries continue to shrink towards the scale of the atom, we face the emergence of fundamental limits which the scaling of current methodology can no longer easily overcome; increasingly, far ranging architectural - both hardware and software - changes are required to utilise the potential of the technology. Four major challenges can be identified:

- **Power dissipation:** it is already not possible to power all parts of a chip at the same time (the *dark-silicon* problem). It has been demonstrated that multiple small CPUs are correspondingly more power efficient than fewer large ones, so the deployment of large cohorts of small CPUs is an obvious way forward.
- **Reliability:** As process geometries continue to shrink, issues of reliability and robustness inevitably emerge. In a system of millions of cores (not unreasonable today), it is unrealistic to expect 100% functionality 'out of the box'; equally, cores will inevitably fail over the lifetime of the system.
- **Communication vs computation:** A traditional argument against moving to large numbers of cores is the relative cost of computation and communication. A core can typically perform several thousand operations in the time taken to

get a single word out of memory and made available to a core. Ever deeper caches and convoluted pipelines can help alleviate the problem, but with conventional architectures, bottlenecks are still almost unavoidable.

- **Programming:** In the past, processor time (core hours) was a valuable resource, and much work went into understanding how to optimise the scheduling of a workload on parallel machines. The automatic (high-level) parallelisation of general-purpose codes remains a 'holy grail' of computer science, but fine-grain parallelisation is frequently signposted by the underlying mathematics. The problem has been in the past that solutions emerging naturally from a numerical solution technique do not map well (cheaply) onto existing architectures, partly because cores were *relatively* scarce, compared to the granularity of a discrete solution. Today, processing is effectively a free resource: cores do not have to be 'kept busy'.

These considerations form another set of constraints on a design space that is already extremely complex. However, they *also* open the way to new approaches: design space may become more convoluted, but it also gets bigger.

Whilst there is no way **through** Amdahl's Law *(The proportion of code that cannot be parallelised will ultimately limit the advantages accrued from more processors)*, the Gustafson-Barsis Law does permit a way **around** it: *(If you can have an **arbitrary** number of processors, the **total** amount of work performed by the system may be increased arbitrarily **at no extra cost**)*.

POETS technology exploits this and explicitly addresses all these points simultaneously.

What is POETS?

POETS - ***P***artial ***O***rdered ***E***vent ***T***riggered ***S***ystems - technology is based on the idea of an extremely large number of small cores, embedded in a fast, hardware, parallel communications infrastructure - the **core mesh**. Inter-core communication is effected by small, fixed size, hardware data **packets** (a few bytes) - aka **messages**.

This proposal describes research to investigate and prototype a software methodology and associated hardware platform to realise the potential of this architecture.

The physical implementation of such a system imposes a fixed and finite topology on the core graph, but a thin (hardware) layer on top of the cores allows the user to virtualise an *arbitrary* connectivity graph on top of the physical one. Once this is done, the mapping of problem domain to processor mesh follows naturally.

For example, a surprising number of industrial problems map naturally and ultimately to solution of the matrix equation [A]x = [B], and the efficient solution of this *prima facie* simple problem for large (say, rank 1000) and ill-defined systems is still the subject of current research. Using POETS technology, ***each matrix element can be mapped onto its own core***: textbook solution techniques become possible because element-element communication is truly (hardware) parallel across the entire matrix. Traditionally, calculations of this type require polynomial time; POETS can perform the calculation in linear time - a massive difference with large industrial problem sets.

Why now?

Because we can - ten years ago it was not possible.

In 1965 Gordon Moore published his famous prediction: that the number of transistors on a chip would double every 18 months or so. This is not a law, just a market prediction, yet it has become a self-fulfilling prophecy that has guided industry for decades. However, it is an exponential prediction, and no exponential is sustainable indefinitely in nature.

Moore's Law is coming to an end, gradually, not because of any one particular show-stopping physical limit, but because of a host of effects, each one in isolation probably capable of resolution, but taken together present an insuperable barrier: *it simply isn't worth it any more*.

But: if we focus on the last few years of this line, and recalibrate the axes in terms of cores/chip instead of transistors/chip, we see the beginnings of a new law: the number of cores/chip is increasing by some multiple/year. Yes, it is an exponent, and so it won't last, but while it does, we should exploit it.

POETS fits into the landscape described above in innovative ways:

- **Power dissipation:** POETS is an *event-driven* system. Cores carry out small calculations in response to the arrival of a message, based on a state subset held in local memories. These calculations may/may not result in the emission of further messages, which are immediately swept up by the communication infrastructure and delivered asynchronously, via hardware, to their target core. The target core is woken (by the hardware delivering the message), acts upon it - as above - and *returns to quiescence*, awaiting another stimulus. POETS is intrinsically energy frugal - you only power calculations when you perform calculations. The design intention is that for a significant portion of time, each core is asleep. This is a programming model of immense power and enormous potential, and is completely orthogonal to conventional architectures.
- **Reliability:** POETS architecture is intrinsically resilient in the face of hardware failure for two reasons: (1) one way of thinking about a POETS core is to view it as an asynchronous finite state machine. Like its conventional counterpart, there is no reason why its state transition graph cannot be disjoint - POETS cores can multi-task at an event level, and so can run inconspicuous system integrity checks in parallel with anything else, allowing possible recovery and/or graceful performance degradation in the face of core or communication fabric failure. (2) is rather more subtle, and not applicable to all problem domains. The dominant use intention of the system is that a fine-grained mathematical model is mapped to the core mesh for subsequent processing - usually but not always some kind of simulation. Failure of a core (or part of the communication fabric) therefore has the effect of compromising the simulation model (specifically the state subset held in the failed area), rather than the algorithm, which is distributed over the entire system. For a certain subset of problems (notably relaxation-based simulations), this perturbation is minimal, localised and does not propagate.

- **Communication vs computation:** POETS sidesteps this tension by 'embedding' the cores in a hardware communications fabric (which is truly parallel) and in which the messages are small and of fixed size (a few bytes). (This is one of the core outcomes of the SpiNNaker project.) With POETS machines, the burden of high-level message choreography is completely removed (there are none): systems trade cores against complexity in both compute and communications.
- **Programming:** This is the area where the largest research challenges lie. Our work with SpiNNaker has demonstrated the validity of the POETS concepts, but the use cases to date have all been hand-crafted. The challenge here is to find a way in which domain-specific specialists - who neither know nor care about the underpinning technology - can use the system to attack large, industrially important problems, focussing on the problem, without the distraction of the solution technique and details.

Taken together, these attributes represent a significant sea-change in the way in which large, industrially relevant problem sets may be attacked. POETS is not a general-purpose architecture, but nor is it a corner-case; it is elegantly suited to a wide variety of industrial problems:

- Finite difference and finite element problems
- Computational chemistry
- Particle & field
- Image processing
- Neural synthesis and simulation (Human Brain Project)
- Drug screening
- Discrete system simulation

In fact, anything where the underlying mathematics naturally formulates as a large graph with large numbers of small, parallel interactions, and no overarching synchronisation requirement.

For some - not all - industrial problems, POETS architectures are capable of delivering orders of magnitude speed increases.

What needs to be done?

Our work with SpiNNaker delivered the first large-scale existence proof of the power of this concept. If we are to exploit this hitherto underexplored and unconventional computing technology, there is still much research to be done. For nearly every step of the development trajectory to date, almost every tool and technique that a conventional software developer takes for granted has had to be re-engineered from scratch. Conventional support tools do not work with this system. We need standardised input formalisms, we need command, control, internal visibility and debug tools, we need to know where the limits are of an instance of the architecture, and how difficult and expensive it is to move these limits.

Why should we bother?

Despite decades of attack, the general purpose parallelisation problem remains one of the most elusive 'holy grails' of computer science. Inspired - possibly - by what we achieved by simply ignoring difficult general problems in our past neural simulation work, and focussing on the functionality we actually *needed*, our thesis here is that POETS is incredibly well-suited to an unexpectedly wide range of important engineering and physics problems, most of which are traditionally the domain of large, extremely resource hungry supercomputers. Event-driven programming, using thousands to millions of small, cheap, energy frugal cores is by far the best platform for some massive engineering problems that traditionally consume millions - tending to billions - of core-hours *and* watts, both of which translate directly into money. **Proponents of exascale computing need event-driven machines if budgets are to remain sub-exascale.**

The problem is not *creating* large cohorts of processors, but how they might be productively *used* to perform or enable the sort of analyses that users of big compute demand. The project focuses on the potential of the hardware architectural point on the scale represented by the earlier SpiNNaker work. It will look at the areas of work where the hardware architecture would be well suited, how those needs can be supported on this architecture by methods and tools, and how the architecture may be further optimised, with the objective of providing the basis of knowledge to support valuable commercial exploitation opportunities anticipated to emerge for commodity HPC.

POETS is a different type of computing architecture; no mature tools or techniques currently exist to exploit it fully, and physical implementations are not yet commonplace. However, this will change: the architecture is unusual but has the attraction of being the choice of evolution - it is the architecture of all neural cortexes, including our own brains. It is highly functional, extremely power efficient and very fault tolerant. Whilst it demonstrably can be programmed, research is needed to make it a commodity capability on a par with the architectures found in almost every electronic system today.

Aims of the research

The phrase "Technology Readiness Level (TRL)" has several definitions, not all of which are mutually consistent, but in essence POETS is currently squarely at TRL 1: the basic principles have been observed and reported. The goal at the project end is to take the concept at least to TRL 4: (Validation of the concept in laboratory conditions), preferably, with the assistance of the project partners, to TRL 5: (Validation of the concept in a relevant domain-specific environment.)

The long-term strategy is to be in a position, at the end of the project, to be able to approach tool vendors and specialist product providers with a solution technique - ***cast in domain-specific terms that they are familiar with, demonstrating solutions to real problems that they care about*** - and make a case for the commercial uptake of the developed technology.

SpiNNaker and POETS:

SpiNNaker is a distributed multi-core system, consisting of a network of 65 000 **nodes**, (each containing 18 200MHz ARM9 cores), embedded in a bespoke (hardware) message-passing infrastructure. The nodes are triangularly connected in a two-dimensional (2D) planar mesh, the edges of which are identified with each other and the whole plane wrapped onto the surface of a torus. Cores communicate via hardware packets of 72 bits. Each node also contains a router unit to control all packet movements, both inter- and intra-node. The design of SpiNNaker explicitly disregards three of the central planks of computer architecture dogma:

1. There is no central synchronising system clock, and all the inter-node (and much intra-node) communication is asynchronous.

2. There is no attempt made to enforce overall memory coherency. Each core has its own private memory, which is not visible to any other core.

3. The message passing infrastructure is non-deterministic (and may, under certain circumstances, be non-transitive).

SpiNNaker is designed for use as a neural simulator. At the level of abstraction utilised by SpiNNaker, a neuron consists of a multi-input, multi-output unidirectional discrete component, communicating with its peers via **action potentials**, modelled as discrete events. A neural system or circuit is represented as a graph of neurons, mapped onto the physical core mesh. A thin hardware layer (the routing system) enables transparent neuron-neuron communication over the underlying hardware - the biological model does not 'see' the underlying electronics.

SpiNNaker is intended to simulate neural aggregates in real time: the biological information contained within a packet resides in the wallclock time of arrival; the 72 bits interact with the underlying routing system to ensure the right packet gets to the right neuron model.

POETS: SpiNNaker is extremely good at the task for which it was designed: neural simulation. The idea of virtualising an abstract arbitrary graph and mapping it to hardware via a thin, hardware, parallel routing layer is immensely powerful, and opens the door to a large array of application domains. However, SpiNNaker is an ASIC, and contains aspects - that we cannot change - that make it unsuitable for the generalisations we wish to explore in POETS.

- In biology, the existence of a spike (packet) contains the only biological data. It is here at this time, or it isn't; this is how mammalian neural systems work. For more diverse applications, we need to be able to put more data into a packet. To keep the speed advantage, the packet size must be small and fixed size, but the bit length of SpiNNaker is crippling for other domains. A few dozen bytes would be fine; 72 bits is not enough.
- Data exfiltration: SpiNNaker relies on gross neural activities for I/O - this is how biological systems work. We need to be able to extract global state data reliably.
- Global synchronisation: Biological systems do not support this behaviour -

neither does SpiNNaker. There exists a wide set of circumstances - in diverse application areas - where this functionality is essential.

Event triggered computing - key points:

Architecture

- *Extremely* large numbers (1000000+) of *extremely* simple cores
- Short (a few bytes), uniform messages
- Hardware massively parallel communications network (on and off-chip)

Disadvantages

- Not a general purpose architecture
- Cannot port existing codebases
- No existing support toolsets

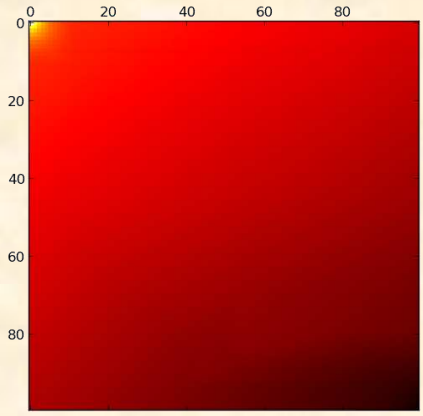
Advantages

- *Massive* speedups for certain classes of problem: $\mathbf{O(n^m) \rightarrow O(k)}$
- Highly fault tolerant
- Low power: **25000 cores < 13A**

Use cases

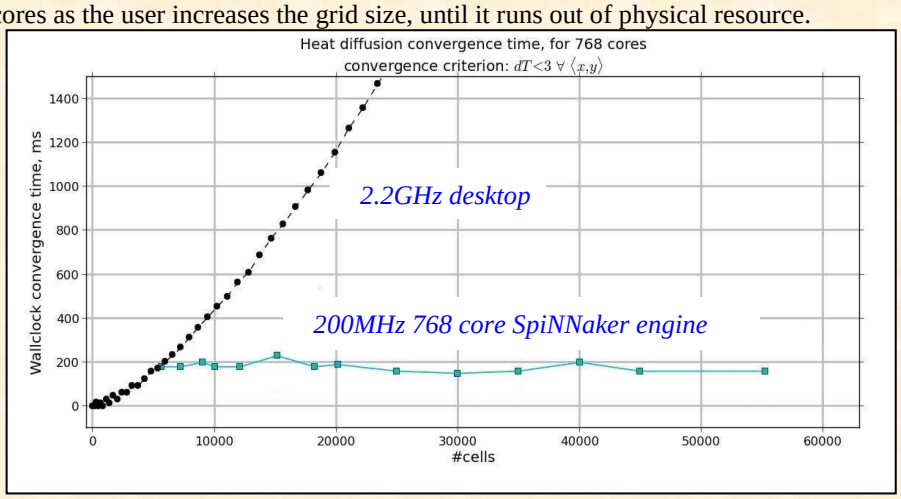
A large proportion of real engineering problems can be broken down to a discrete graph, albeit one with sometimes millions of nodes. If we have millions of cores and a fast communications infrastructure, we can trade cores off against computational complexity, and exploit the near-perfect parallelism of the hardware interconnect. The Use case portfolio suggests some of the industrial application areas for POETS technology.

Use case: Finite difference calculations

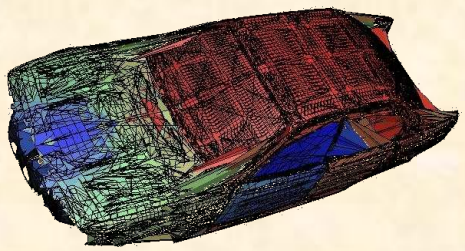


Consider the canonical finite difference heat equation on a 2D square grid: ***each grid point is represented by an individual core***, which holds the grid point state (temperature) plus ghosts of the immediate neighbouring states, and communicates only with its direct neighbours by messages. On receipt of a message - any message - a grid point recomputes its state; if this has changed, it broadcasts the new state value to its neighbours. All the cores do this simultaneously (asynchronously), triggered by the arrival of messages. Pinning the opposite corner temperatures and letting heat flow freely produces the obvious result.

The interesting point is the wallclock solution time as a function of grid size: the algorithm, as cast, will continue to operate in constant time, using more and more cores as the user increases the grid size, until it runs out of physical resource.

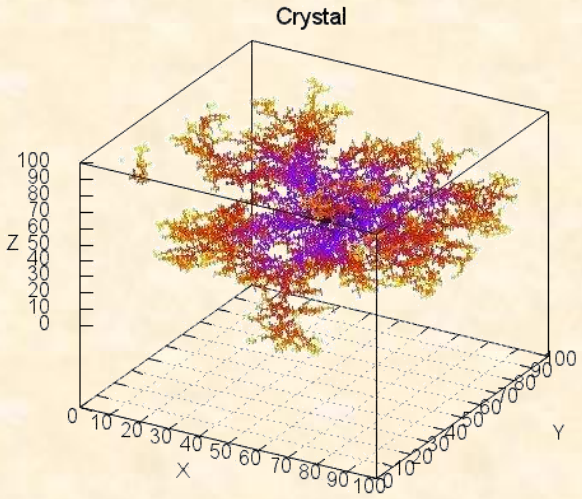


There is, obviously, no reason why we need restrict our analyses to a uniform grid:

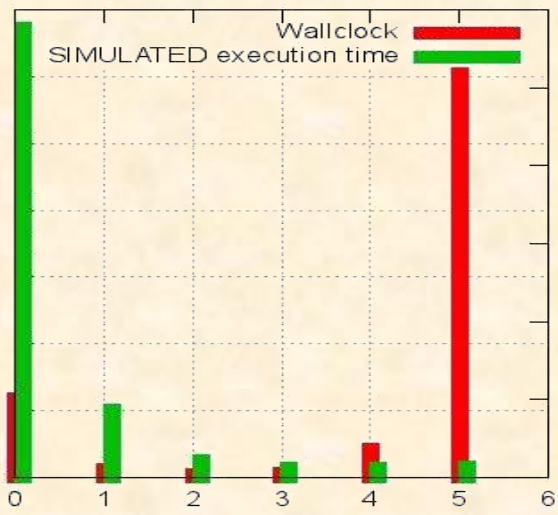


Use case: Neuron synthesis

Large scale neural simulations - which underpin almost all of computational neuroscience - require *realistic* models to simulate, and the generation of these models is not trivial or computationally cheap. In biology, $1mm^3$ of neural matter contains around 10^5 cell bodies, 4km of axons, 5.10^6 dendrites and 7.10^9 synapses. *Each* neuron is represented as a space-filling tree, which does not intersect with itself or any other neuron. Vasculature - essential for accurate modelling - approximately doubles the complexity of the space.



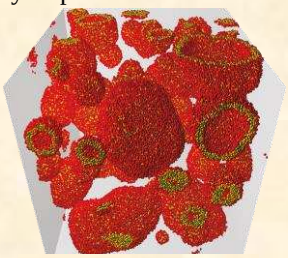
A popular way of approaching this problem is to tile space (the universe) with three-dimensional cubes, populated with 'virtual neurites'. These move about randomly, condensing (and sticking) onto a seed neuron whenever they touch one.



Using POETS methodology, we can allocate ***each spatial 3-cube to an individual core***, and handle the passage of neurites and the growth of neurons across cube boundaries by passing messages. Run 0 below shows the universe modelled by one POETS core; run 5 by 32 x 32 x 32 (= 32768) cores. (Intermediate data points are for 2^3, 4^3, 8^3 and 16^3 cores.) The figure itself is a simulation, but nevertheless the speedup trend is clear and impressive.

Use case: Spatio-temporal simulation of stochastic biochemical processes

Biochemical processes are increasingly being modelled in-silico, where a low-level description of chemical interactions is used to drive a simulation of higher-level biological activities; these models have been made possible by improved abilities to automatically extract individual molecular pathways. Modelling the interactions within an entire cell is computationally infeasible, due to the large number of molecules, and the huge number of interactions needed for the cell to make enough progress to be interpreted at a high level.



*J. Shillcock, Langmuir 2012, **28**, 541-547*

Approximations are used to interpret and capture low-level processes as coarse behaviour, which have recently allowed the creation of whole-cell models for simple bacteria. However, there remains the question of whether some important behaviour is only captured by the low-level interactions, so there is still a need to perform high-fidelity simulations of chemical processes which track individual molecules.

Current cell simulation techniques provide an efficient method for simulating systems with tens of thousands of reactants, but current compute systems are too slow to come close to the speed needed to model all interactions within a cell. The spatial nature of the problem, and the heavy reliance on local lightweight communication, means that space can be discretised and ***each cube mapped onto a core***:

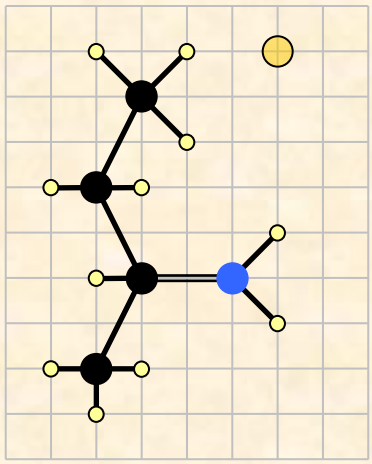
- *Loose temporal coupling:* the notion of time within a simulated system is intrinsically fuzzy, and only local causality matters.
- *Local fault tolerance:* as long as molecules can propagate between local volumes within a spatial region, the failure of one or two volumes within that region is largely irrelevant.
- *Scaling via spatial decomposition:* due to the huge number of molecules involved, the problem can be decomposed spatially until all available CPUs are occupied, achieving good utilisation of all available CPUs.

As well as being a good fit for the architecture, stochastic chemical simulation also presents some interesting challenges and research opportunities:

- *Dynamic molecule balancing:* during the simulation molecules naturally migrate around the system, potentially requiring cores to negotiate the size of the volume they manage.
- *Dynamic rate balancing:* the rate at which reactions within a volume occur depends on the number and balance of local molecules, and "hot" areas will eventually limit the rate of progress within the entire system.

Overall we can expect to see this application scaling linearly with the number of cores in the system; where a traditional multi-core or GPU simulation becomes communication limited, the intrinsic spatial communication capabilities of POETS means the bottleneck is removed.

Use case: Particle & field



Advances in conventional computer technology have made it feasible to simulate the mutual interactions of huge ensembles of particles, but at a massive core-hour cost. By employing sometimes innovative and sometimes brutal approximations, it is possible, for example, to model the migration of proteins through a cell wall at the level of individual particles (where a particle is a group of atoms - a sort of sub-molecule). These computational experiments push at the boundaries of what is possible today - and the further introduction of long-range forces into the experimental regime (for example electrical charge) places many interesting and useful studies out of range. The difficulty here lies in the fact that non-trivial forces extend over many particle-particle separations, making the computational graph necessary to solve the system almost a clique. Any attempt to parallelise such a system computationally rapidly becomes communication bound.

POETS, however, offers a (partial) solution to this. Whilst particle-particle analyses will not map usefully onto a POETS system, an alternative representation, particle & field, does. In a particle & field analysis, ***space is tiled, and each core "owns" a volume***, managing the particles that inhabit that volume. Particles do not, however, interact with each other, they interact with a global field (which may be multi-valued in space):

Particles tell the field how to deform, and
the field tells the particles how to move.

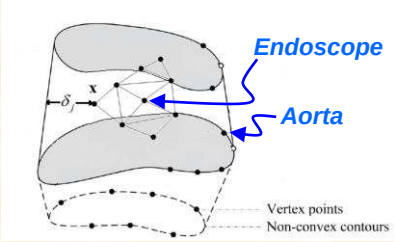
The big difference - from POETS point of view - is that deformations in the field can spread out from their source via core-core communication, and the intensity of the field can be calculated locally (and simply) by the local core - no reference back to the originator of the perturbation is necessary. Particles derive the force incident on them from the local field: again, information that is to hand.

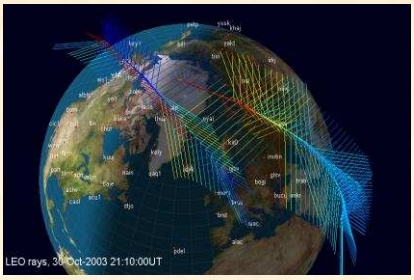
Use case: Industrial image processing

At their core, many industrial problems resolve to **[A]x = [B]** or similar. Whilst matrix solution techniques are the stuff of undergraduate textbooks, industry is interested in matrices of massive ranks (thousands), which are often sparse and ill-conditioned. Further, *sequences* of matrices that represent continuous processes are often mutually inconsistent.

Mapping a core to each matrix element allows the inversion of matrix equations in O(n) time, better than any low-thread solution on a conventional machine. This opens the door to a host of real-time image based applications:

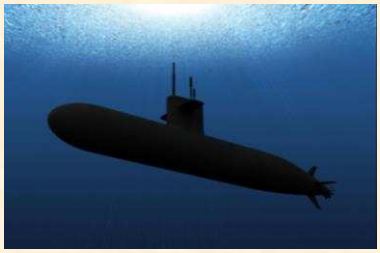
Medical: detailed non-invasive tomographic imaging of biological structures - bones, brains, vascular systems; image-guided surgery. The last requires the reconstruction of images that are noisy and fast moving (typically around 10^6 points/s), where inaccuracy can easily cause death.



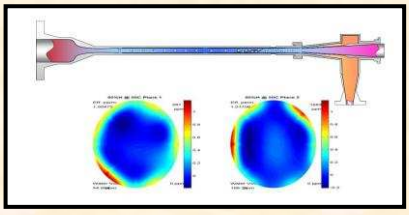


C.N. Mitchell, University of Bath

Measuring ionospheric weather: can decrease the error of GPS fixes by a meter. So what? GPS guided ocean oil drilling costs around 10^6 £m^{-1}.



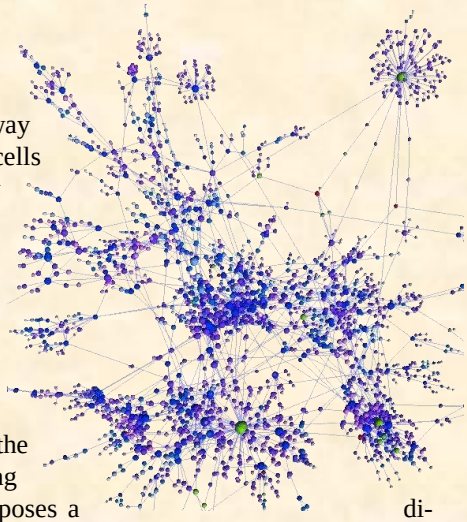
Inverse field problems: detection and location of submerged cylindrical magnetohydrodynamic anomalies



W. Yang, University of Manchester

Production line quality control: mixing efficiency, void detection, structural integrity

Use case: Drug screening



Computational chemistry has a long way to go before the interactions of drugs and cells can be modelled and simulated accurately and usefully at a molecular level. The difficulty arises from the sheer volume of computation necessary to model the interactions of the millions of atoms comprising even the simplest biological drug-relevant system.

The natural response of the simulation engineer in this situation is to increase the level of granularity of the system, modelling at larger and larger resolutions, which exposes a dilemma: the higher the modelling level, the more tractable the total problem, but on the other hand, by coarsening the level of modelling abstraction, interactions were discarded that may turn out to be dominant in some unexpected way; and the ultimate object of simulation is to illuminate interactions that were unexpected. The *art* of modelling for simulation - in any discipline - consists of finding ways to capture relevant interactions as simply as possible without compromising (too much) the representation of reality embodied in the model.

Drug discovery is the process through which potential new medicines are identified. It is traditionally slow and labour intensive, but remains a vital step in the identification of new medicines and treatments. A difficulty arises from the fact that even the simplest drug interacts not only with its primary target (cell), but also with secondary structures - other proteins in other cells. These also interact with each other, in complex ways, making the prediction of the impact of a specific pharmaceutical intervention an almost impossible task, unless the system is modelled at infeasible levels of granularity.

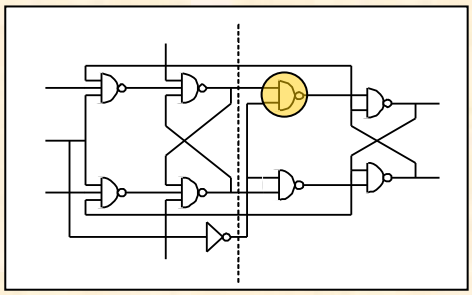
One attempt to overcome this bottleneck employs a radically different methodology to represent a cell and its constituent proteins: a cell is represented by a graph. The nodes of the graph are the proteins contained within the cell, and the edges of the graph the known protein interactions. These edges may themselves be quite complex, to model known adjuvant and chaperoning effects. In a similar manner, a potential drug may be represented by a graph, modelling the effects of the drug on specific proteins. The screening process then involves running a "cross-product" between the biological cell library and the putative drug, analysing the effects by looking at the topology of the affected cell graphs.

None of the steps in this process are particularly individually demanding, but again the difficulties arise from the sheer size of the graphs: a cell graph can contain tens of thousands of nodes. ***Mapping the model graph nodes onto POETS cores*** opens the way to parallelising the graph-graph interactions, with a potentially dramatic impact on computational throughput.

Use case: Discrete simulation

In 1979 - four years before the PC became available - a paper about discrete simulation (*K.M. Chandy and J. Misra, IEEE-T Software Engineering, SE-5 no 5 1976 440-452*) was published by the IEEE, where the authors stated in the abstract:

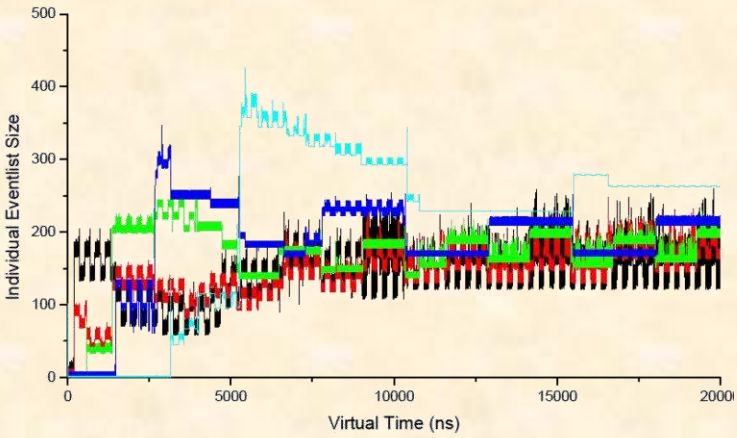
... We propose a distributed solution where processes communicate only through messages with their neighbours; there are no shared variables and there is no central process for message routing or process scheduling. Deadlock is avoided in this system despite the absence of global control. Each process in the solution requires only a limited amount of memory.....



They were talking abut POETS, thirty-six years ago.

The match to the POETS technology is quite remarkable; ***mapping one logical device to each core*** - something unimaginable in 1979 - allows the simulation of industrially relevant systems today. The perennial problem of maintaining overall simulation causality is elegantly overcome by the introduction of timing events that are broadcast along the same signal paths as contained in the circuit under simulation; thus the overhead is an approximate doubling of the signal traffic, a negligible cost considering the speed gearing from all the cores.

Further levels of sophistication are possible: where the circuit under simulation has more devices than the POETS engine has cores - it can happen - we can map multiple devices to a single core, and further, allow the POETS engine to dynamically modify this mapping, load-balancing the simulation on the fly.

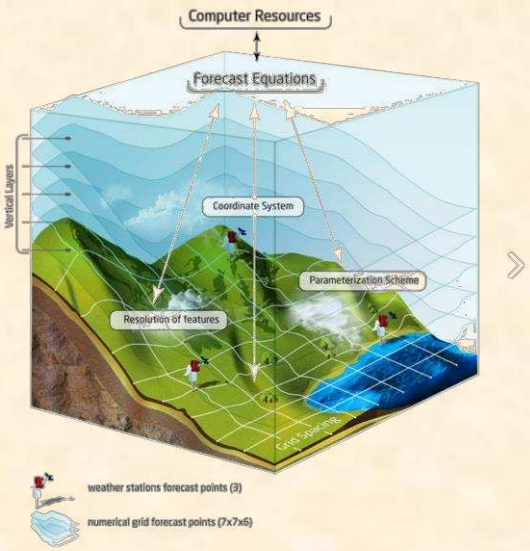


Use case: Weather modelling

Weather modelling involves predicting the interactions of wind, solar radiation, ground conditions, pollution and a host of other features into a numerical model whose state is capable of extrapolation into the future in a computational timeframe that is faster than real time - there is little point in coming up with an accurate prediction of tomorrows weather if it takes two days to do it.

Current methodologies decompose the atmosphere into a non-uniform (multi-scale) grid, and solve the equations concerning the movement of air between grid cells, ensuring continuity of pressure, temperature, density et al across the cell boundaries. The atmosphere over the UK is divided into cells around 1.5 km on a side (giving a UK - based cell count of around 6.10^7 cells); over Europe around 4 km on a side and the rest of the world is modelled at a resolution of around 17 km.

The solution technique revolves around mapping the atmospheric cells onto the available cores of whatever machine is being used to solve the system - there is a tradeoff between cell size (accuracy - pushes the cell size down) and the inter-core traffic load (speed - which pushes the cell size up). Much effort is required to find the 'sweet-spot', resulting in the best accuracy from the fastest cell configuration. (Much effort is also expended in finding better models to represent the atmospheric behaviour, but POETS solves equations, it does not derive them.)



Using ever smaller cells - and ***mapping only a handful of these to each POETS core*** - provides two-fold benefits: the cell-cell traffic maps comfortably onto the hardware routing fabric of the engine (which in any case in POETS engines is hardware and fast); and the equations governing the behaviour of the atmospheric model can become much simpler, as the range scale of the non-linearities intrinsic to the physics become comparable to the new, reduced cell resolution.

The Effects of BTI Aging on the susceptibility of On-Chip Communication Schemes to Soft Errors in Nano-Scale CMOS Technologies

Basel Halak, ECS, University of Southampton, Southampton SO17 1BJ, UK
Email: bh9@ecs.soton.ac.uk

This work is done in celebration of the 60th Birthday of Professor Alex Yakovlev, my mentor and my friend. Happy Birthday Alex ☺

Abstract. The continuous scaling of semiconductor devices has introduced new circuit failure mechanisms such as bias temperature instability, and made existing reliability problems more severe such as single event transients (SET), variability and crosstalk. This work provides a frame work to quantify the soft errors induced by crosstalk and radiation hits in on-chip communication schemes and investigates the effects of aging on the susceptibility of on-chip communication to these transient failures. It also provides a comprehensive comparison between synchronous and asynchronous communication methods in terms of their robustness against soft errors. Our results based on SPICE level simulations in 90 nm technologies indicate that BTI aging increases the probability of delay-induced soft errors but mitigate the effect of glitches-triggered errors.

1. Introduction

One of the major challenges facing the SoC designers of dependable and/or safety critical systems is the intrinsic unreliability of the communication infrastructure in Nano scale CMOS technologies [1, 2]. There are number physical mechanisms which may cause soft errors in on-chip communication links, examples include: crosstalk and radiation, and power bounce [3]. Such problems are further worsened by Variability and CMOS aging. The former refers to the inaccuracies in manufacturing pro-

cesses and within-die voltage-temperature variations that lead to fluctuations in circuit performance and power consumption. It arises from scaling very large-scale integrated (VLSI) circuit technologies beyond the ability to control specific performance-dependent and power-dependent parameters [4, 5]. Variability has become a first order limitation to continued scaling. At deep submicron technology nodes, the achievement of parameter precision becomes exponentially more difficult due to the limitations imposed by quantum mechanics. The various intrinsic sources of variability such as random dopant distribution cannot be reduced by better process control and their effect is generally random, this is shown by atomistic modelling in [6, 7]. Therefore, the impact of variability is expected to be significant in future technologies [5, 8], making variations an unavoidable characteristic of future VLSI circuits.

CMOS aging is another major concern for modern VLSI designers, It refers to a slow progressive degradation in the performance of MOS transistors; it is caused by several failure mechanisms; namely: Bias Temperature Instability (BTI); Hot Carrier Injection (HCI); and Time- Dependent Dielectric Breakdown (TDDB) [9, 10]. BTI is often cited as the primary reliability concern in modern processes [11]. Negative bias temperature instability (NBTI) in PMOS transistors is more dominant than positive bias temperature instability in NMOS transistors in the latest process technology especially after the introduction of nitrogen into gate stacks, which reduces boron penetration and gate leakage, but leads to worse NBTI degradation [12]. This mechanism is characterized by a positive shift in the absolute value of the threshold voltage of the PMOS device, such a shift is typically attributed to hole trapping in the dielectric bulk and the breakage of Si-H bonds at the gate dielectric interface [13]. Positive bias temperature instability (PBTI) in NMOS transistors also has the same effect but is only considered crucial in devices which use high K-dielectrics. BTI aging causes delay to increase, which can eventually lead to timing errors and system failure [9, 10, 14]. The effects of aging on the performance and reliability of systems on chips have been extensively addressed in the literature [15-20]. In addition, there are extensive works in the literature in the area of reliability enhancement techniques for soft errors. The authors of [21] proposed a data-redundancy-based fault tolerance method for self-timed phase-encoded channels in order to mitigate the effect of crosstalk-related errors. In [22] Lin et al proposed a radiation hardened register design for protecting microprocessors pipelines from alpha particles hits. The use of error correction and detection codes has also been proposed to enhance the reliability of on-chip communication [23, 24]. However to the best of our knowledge, there are no previous studies on the effects of BTI aging on the susceptibility of on-chip communication schemes to soft errors. The contributions of this work are as follows:

1) It provides a framework to quantify the soft errors induced by crosstalk and radiation hits in on-chip communication schemes.
2) It investigates the effects of aging on the susceptibility of on-chip communication to soft errors.
3) It provides a comprehensive comparison between synchronous and asynchronous communication methods in terms of their robustness against transient failure mechanisms.

The rest of the paper is organized as follows. Section 2 and 3 outline the computation of closed form expression to quantify the soft error caused by crosstalk and radiation hits respectively. The experimental setups and the analysis methodology are explained in section 4. Simulation results are presented and discussed in section 5. Conclusions are drawn in section 6.

2. Computation of Crosstalk-Induced Error Rate

The soft errors caused by crosstalk can be classified into two types, namely:

1. Glitch-induced soft errors: these are caused by static noise pulses induced on a quiet victim net due to switching of neighbouring aggressors. If such a pulse propagates through logic gates and reaches storage elements, it can mistakenly be considered as a valid data item
2. Delay-induced soft errors: these are due to violations in the timing constraints which occur when the timing of a stage (i.e. gate and interconnect) becomes uncertain due to coupling from the switching activity of neighbouring stages (i.e. Miller effect). This results in a change in the total capacitance of the wire, hence dynamic delay.

The following two subsections provide closed form expressions that can be used to compute the probability of crosstalk-induced soft errors in synchronous and asynchronous channels, respectively.

2.1. Crosstalk Error Rate in Synchronous Links

Synchronous design methodologies globally distribute a timing signal (a clock) to all parts of the circuit. Transitions (rising and/or falling depending on the design) on this clock line indicate moments at which the data signals are stable and therefore ready to be sampled. In synchronous links, crosstalk can cause both delay and/or glitch-induced soft errors; this depends on the data transition state on the bus.

To illustrate crosstalk effects on a synchronous communication channel, consider the link shown in figure 1, it is a typical D-flip-flop (DFF) pipelining stage. Data is

normally transmitted every clock cycle where the latches are triggered at the clock rising edge.

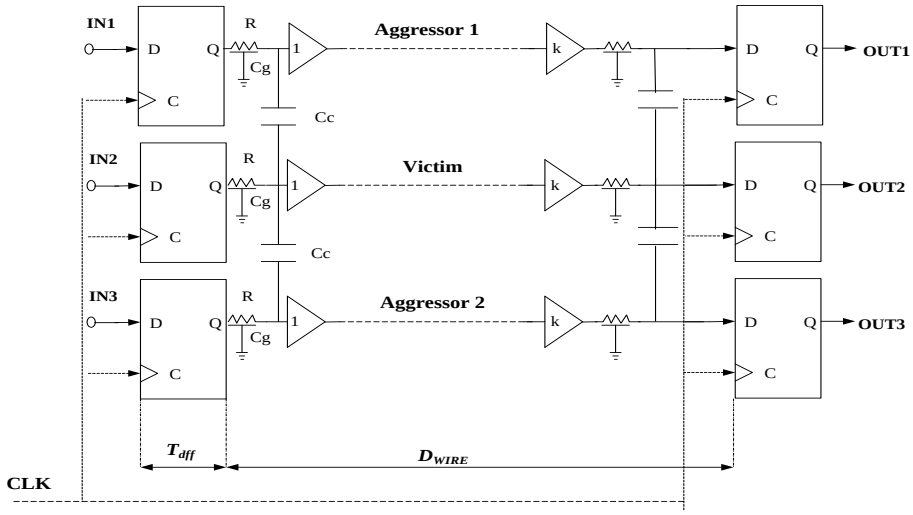


Figure 1: A Synchronous Communication Link

To ensure correct data transmission, intermediate values of logic signals or glitches must settle down before the next clock edge such that there is a stable logic value at the input of each register. Crosstalk-Delay-induced soft errors can only occur on a victim wire during a data transition event (0=> 1 or 1=>0), whereas glitch-induced soft error occurs on a victim wire when the data on it is stable and there are transitions on neighbouring lines. For example in a three-wire link as shown in figure 1, each wire has four possible transition states (UP (0=> 1), Down (1=>0), Stable(1), Stable(0)), so there are (4^3 =64) possible transition states. In 32 of which, the middle line is vulnerable to crosstalk delay errors. In 14 of which are, the middle line is vulnerable to crosstalk glitch errors. In the remaining 18 state, the middle line is not vulnerable to any crosstalk-induced soft errors.

The average probability of a crosstalk delay soft error (CDSE) on a victim line per transmission cycle (i.e. per clock cycle in this case) can be computed as follows:

$$CDSE = \sum_{I=1}^{SD} TER_i * Oi \quad (1)$$

The average probability of a crosstalk glitch soft error (CGSE) on a victim line per transmission cycle can be computed as follows

$$CGSE = \sum_{j=1}^{SG} GER * Oj \quad (2)$$

Where:

TERi is the probability of crosstalk-delay induced soft error during a bus transition state i

GERj is the probability of crosstalk-delay induced soft error during a bus transition state j

SD is the number of transition states on the bus in which delay errors are possible

SG is the number of transition states on the bus in which glitch errors are possible

Oi is the occurrence probability of a transition state i

Detailed closed form expressions to compute *TER* and GER are provided in our previous work [25].

2.2. Crosstalk Error Rate in Delay-Insensitive Asynchronous Links

Asynchronous methodologies typically encode the timing information in the data line activity itself, therefore there is no need for a separate timing signal [26, 27]. Delay-insensitive channels are an important class of asynchronous communication scheme, these channels are self-timed, and hence, insensitive to signal propagation delays, in other words, they have no timing constraints. Therefore, they are inherently resilient to crosstalk-delay soft errors. However, they are still prone to crosstalk glitch soft errors. There are many possible delay insensitive coding methods such as M-out-of-N, Burger, and Knuth codes [26, 27]. The average probability of a crosstalk-glitch soft error (CGSE) per transmission cycle can be computed using equation 2 above.

3. Computation of Radiation Hits-Induced Error Rate

The probability of a radiation-induced soft error is the product of two factors, namely: the probability of energetic radiation hits and the probability of a functional failure caused by such a strike. The first factor depends on the environment under which the circuit is operating, and is independent of the architecture and the design of the circuit; therefore, it is not relevant in a comparative analysis of various design techniques, and it is not going to be addressed in this work. On the other hand; the

probability of a radiation-induced glitch turning into a soft error is a function of the circuit architecture and fabrication technology; therefore, it can be used as a metric to estimate the susceptibility of an on-chip communication schemes to radiation-induced soft errors. The following two subsections provide closed form expressions that can be used to compute the probability of a radiation glitch soft error (RGSE) for both synchronous and asynchronous channels:

3.1. Radiation Error Rate in Synchronous Links

A radiation-induced glitch at the input of an edge triggered flip flop can cause a functional failure if it is erroneously sampled or if it forces the devise to go into metastability state. The occurrence time of the glitch and its width determine whether or not it generates an error. Consider the case of a synchronous link in figure 1, assume a glitch is generated by a repeater due to a radiation hit. In order for this glitch propagate to cause an error, it has to satisfy the following three conditions:

1) Glitch amplitude Condition (GA): the amplitude of the glitch should exceed the threshold voltage of the receiving flip-flop
2) Glitch Timing condition(GT): The glitch should coincide with the sampling clock in order for it to be latched
3) Glitch Width Condition (GW): it should be sufficiently wide i.e. (its amplitude should remain higher than the threshold voltage while the flip-flop input is being sampled.)

Therefore, given a radiation hit has actually happened, the probability of a soft error caused by such the radiation-induced transient pulse in synchronous links is given as follows:

$$RGSE = GA * GT * GW \quad (3)$$

Detailed closed form expressions to compute *GA*, *GT,* and *GW* are provided in our previous work [28].

3.2. Radiation Error Rate in Delay-Insensitive Asynchronous Links

In this type of links, a radiation-induced glitch can lead to the generation of illegal code words, for example in (1-of-4) based channel, a code word (0001) can be received as (0011). Depending on the design of the receiver, this erroneous data symbol can be disregarded or wrongly interpreted as a different symbol. In order for this a radiation-induced transient to cause such a failure, it should only satisfy the glitch

amplitude condition (ga). This is because data can be sampled at any time in delay insensitive links, so there is a high probability for a glitch to cause an error if it exceeds the threshold voltage of the logic gates. Therefore, the probability of a soft error caused by a radiation glitch in asynchronous links is given as follows:

Therefore, given a radiation hit has actually happened, the probability of a soft error caused by such the radiation-induced transient pulse in Asynchronous delay insensitive links is given as follows:

$$RGSE = GA \tag{4}$$

4. Analysis Method and Experimental Setups

In this section we first outline the aging model we have used, and then summarize our analysis method and experimental setups

4.1. Bias Temperature Instability Model

BTI consists of Negative Bias Temperature Instability (NBTI) in pMOS transistors and Positive Bias Temperature Instability (PBTI) in nMOS transistors. The reaction diffusion model has been developed in [29] to allow designers to estimate the drift of Vth (ΔVth) induced by BTI effects as a function of technology parameters, operating conditions and time. However, the drift of Vth does not depend on the frequency of input signals, but only on the total amount of the stress time, therefore a closed form analytical model has recently been proposed which allow designers to estimate long term threshold voltage shift as follows [30, 31] :

$$\Delta V_{th} = \chi K \sqrt{C_{ox}(Vdd - Vth)} \exp\left(\frac{E_a}{k_B T_A}\right) (\alpha t)^{1/6} \tag{5}$$

C_{ox} is the oxide capacitance;

t: is the operating time;

α: is the fraction of the operating time during which a MOS transistor is under a stress condition. It has a value between 0 and 1. α = 0 if the MOS transistor is always OFF (recovery phase), while α = 1 if it is always ON (stress phase);

E_a: is the activation energy ($E_a \cong 0.1$eV);

k_B: is the Boltzmann constant;

T_A: is the aging temperature;

χ: is a coefficient to distinguish between PBTI and NBTI. Particularly, χ equals 0.5 for PBTI, and 1 for NBTI;

K: lumps technology and environmental parameters.

4.2. Analysis Methodology

In order to estimate the impact of BTI aging on the susceptibility of on-chip links to soft errors, we have adopted the following approach:
First, we have considered a synchronous communication link as shown in figure 1. We have also considered three delay-insensitive channels: (1-of-4), (3-of-6) and (2-of-7). The physical layouts of all communication schemes considered are the same in all experiments. Second, we estimated the soft error rate induced by crosstalk and radiation hits expression provided in equations (1), (2), (3) and (4) based on SPICE level simulations in 90nm technology. Third, in order to estimate the impact of BTI aging, we computed the degradation in electrical parameter of transistors based on the model presented in Section 4.1 The ΔVth value obtained for each considered operating time interval was then utilized to customize the SPICE device model and simulate the communication link with the proper BTI degradation. Particularly, we have considered three operation points after 1 day, 1 year, and 5 years. Fourth, we estimated the soft error rate induced by crosstalk and radiation hits have been estimated again using circuits with degraded electrical parameters.

4.3. Experimental Setups

The communication medium considered in this study is top metal layer in the 90nm technology. The interconnect structure used is three signal lines between two grounded shields as shown in figure 1. Each wire is modelled as a distributed RC network with four Π-model segments. The resistive and capacitive parasitic elements of the wires are calculated using equations from [32]. To model process variations, we have considered seven sources of variations, namely: wire width (w), metal thickness (t), interlayer dielectric thickness (h), metal resistivity (ρ), gate length (Leff), supply voltage (Vdd) and temperature (T). The variability of these parameters are assumed to be Gaussian and mutually independent. We have also assumed that

wire width (w) and spacing(s) are perfectly negatively correlated, letting w to be the independent variable. Table 1 summarises the mean and variations for each of these parameters. It is worth mentioning that the wire width shown in Table 1 is the minimum wire width in the considered technology.

Table 1: The Nominal Values and Variations of Circuit Parameters

Parameter	Nominal Value	Variations
Interlayer Dielectric height	0.94 *um*	20%
Metal resistivity	21.8 *nΩ.m*	20%
Wire thickness	0.81 *um*	20%
Wire Width	0.56 *um*	20%
Transistor length	90 *nm*	10%
Power Supply	1.2 *V*	10%
Temperature	27 ^{o}C	12 – 43 ^{o}C

In all of our simulations we have considered a channel with a 10 mm wire length, each line has four equally spaced and minimally sized repeaters. We used a clock frequency of 1.5 GHz for simulating the synchronous link.

For crosstalk error rate commutation of synchronous links, the delay of the victim wire has been estimated using spice-level simulation for all possible transition activities. The amplitude and the width of the induced crosstalk glitch the victim line glitch have also been estimated in the same manner. To estimate the impact of variability on the delay and glitch characteristics, we have employed the DoE statistical analysis approach to device a set of experiments in order to estimate the mean value and statistical distributions of measured parameters as shown in our previous work[33]. The above experiments are then repeated using circuits with degraded electrical parameters according to the aging model explained in section 4.1 For the estimation of crosstalk error of delay-insensitive links we followed the same approach above using the code [34]. The recorded values from each simulation are used to compute the crosstalk error rate according to the equations presented in section 2. For the computation of radiation errors, the single event transient (SET) values for 90nm technology presented in [35]. The mean and standard variation of the SET width is 500 and 150 ps respectively. The mean and standard variation of the SET amplitude is 0.1v and 0.32v respectively , these values are in agreement with the statistical distributions of the SET characteristics presented in [36].

5. Results and Discussions

5.1. The Impact of Aging on Crosstalk Error Rate

The results shown in figure 2 indicate the BTI aging leads to an increase in the susceptibility of synchronous links to crosstalk –induced delay errors. In this case the probability of a soft error is 10^{-8} which mean, every 100 million clock cycle we expect to have one soft error. After five years of operation the probability of a soft error is $3* 10^{-5}$ which is there order of magnitude larger. This sever degradation can be attributed to the induced delay degradation of the repeaters and flip-flops due to BTI aging in these links, which increases the probability of violating the timing constraints. On the other hand, the results shown in figure 3 indicate that the effect of crosstalk glitches on the reliability of synchronous links become less pronounced with aging, because the latter leads to an increase in the threshold voltages of CMOS transistors, therefore some of the crosstalk glitches will no longer be able to propagate and cause soft errors. A similar trend is also observed for asynchronous delay insensitive links as shown in figure 4. One interesting observation from figure 4 is the fact that some delay-insensitive code are more susceptible to crosstalk errors, this is because crosstalk errors are dependent on the transition activities and the data patterns on the bus, both of these factors are affected by the choice of a delay insensitive code. In particular 2-of-7 links are most prone to crosstalk-related errors at any time, so any glitch that exceeds the threshold voltage of a logic gate can propagate

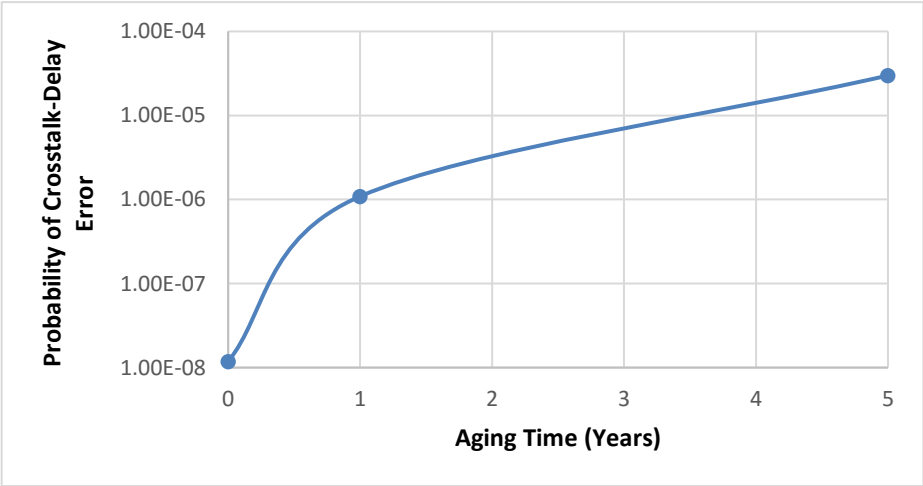


Figure 2: The Impact of BTI Aging on the Rate of Crosstalk-Induced Delay Errors in Synchronous Links

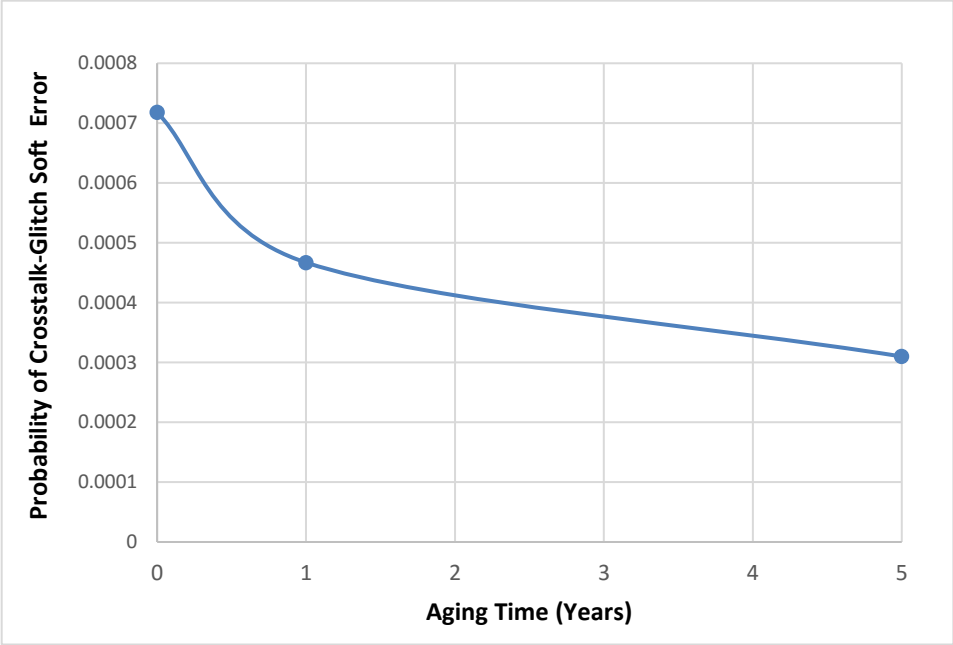


Figure 3: The Impact of BTI Aging on the Rate of Crosstalk-Induced Glitch Errors In Synchronous Links

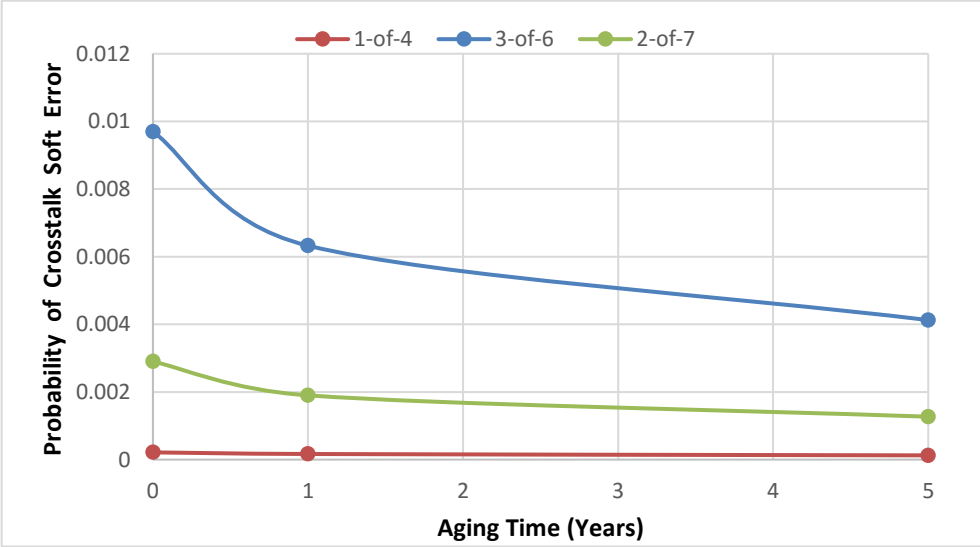


Figure 4: The Impact of BTI Aging on the Rate of total Crosstalk Errors in Asynchronous Links

5.2. The Impact of Aging on SET Error Rate

Figure 5 indicates that the effect of radiation-induced SET on the reliability of communication links become less pronounced with aging, this is mainly due to the fact that aging lead to an increase in the threshold voltages of CMOS transistors, therefore some of the SET glitches will no longer be able to propagate and cause a soft error. Again, asynchronous links seems be more vulnerable to this type of soft errors than synchronous channels due to their inherent nature of sampling logic values

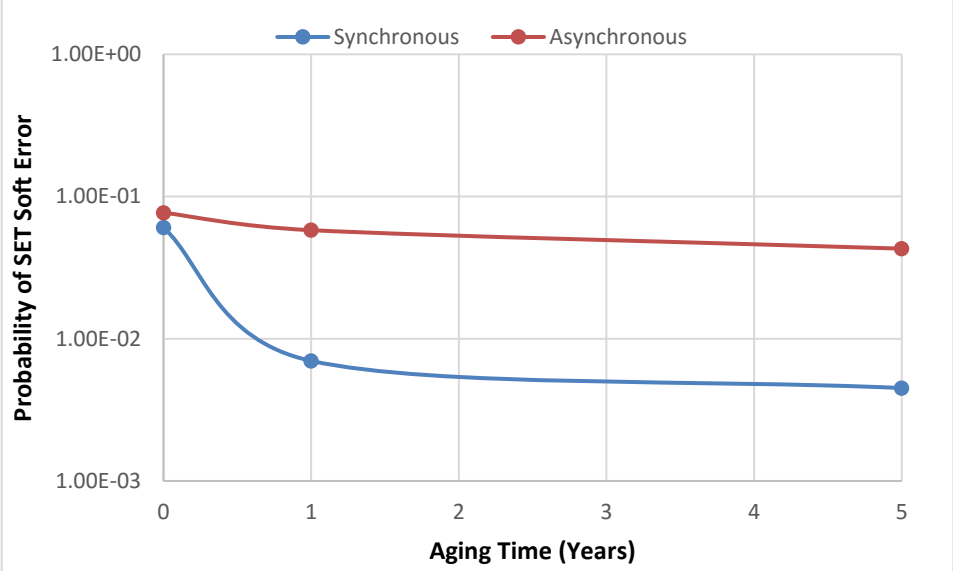


Figure 5: A Comparison of Single Transition Event Errors-Susceptibility (Synchronous vs. Asynchronous Links)

6. Conclusions

BTI CMOS aging is major concern for modern VLSI designers, It refers to a slow progressive degradation in the performance of MOS transistors; this mechanism is characterized by a positive shift in the absolute value of the threshold voltage of the PMOS device, such a shift is typically attributed to hole trapping in the dielectric bulk and the breakage of Si-H bonds at the gate dielectric interface. This work shows that aging can significantly affect the susceptibility of on-communication schemes to soft errors caused by cross-talk and/or radiation hits. In particular, BTI aging can lead to sharp increase in the rate of delay-induced soft errors; this is a major concern for synchronous

links. On the other hand, aging seems to mitigate the probability of a soft error caused by a noise pulse (due to crosstalk or SET), this means the robustness of asynchronous communication schemes against may become better as the circuit ages. On the other hand, Synchronous communication links seem to become more prone to delay-induced soft errors with aging but better protected against glitches and noise pulses.

7. Acknowledgement

I would like to thank my colleagues Daniele Rossi, Yang Lin and Mark Zwolinski for sharing their expertise.

8. References

[1] C. Grecu, A. Ivanov, R. Saleh, and P. P. Pande, "NoC Interconnect Yield Improvement Using Crosspoint Redundancy," in *2006 21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2006, pp. 457-465.

[2] P. Magarshack and P. G. Paulin, "System-on-chip beyond the nanometer wall," in *Design Automation Conference, 2003. Proceedings*, 2003, pp. 419-424.

[3] K. L. Shepard and V. Narayanan, "Noise in deep submicron digital design," in *Computer-Aided Design, 1996. ICCAD-96. Digest of Technical Papers., 1996 IEEE/ACM International Conference on*, 1996, pp. 524-531.

[4] S. R. Stg, J. Srivatsava, and R. Narahari Tondamuthuru, "Process Variability Analysis in DSM Through Statistical Simulations and its Implications to Design Methodologies," *International Symposium on Quality Electronic Design,* pp. 325-329, 2008.

[5] S. Nassif, "Delay variability: sources, impacts and trends," *Solid-State Circuits Conference,* pp. 368-369, 2000.

[6] C. Millar, D. Reid, G. Roy, S. Roy, and A. Asenov, "Accurate Statistical Description of Random Dopant-Induced Threshold Voltage Variability," *Electron Device Letters, IEEE,* vol. 29, pp. 946-948, 2008.

[7] C. Alexander, G. Roy, and A. Asenov, "Random-Dopant-Induced Drain Current Variation in Nano-MOSFETs: A Three-Dimensional Self-Consistent Monte Carlo Simulation Study Using (Ab initio)Ionized Impurity Scattering," *Electron Devices, IEEE Transactions on,* vol. 55, pp. 3251-3258, 2008.

[8] "International Technology Roadmap for Semiconductors (www.itrs.net)."

[9] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, "Predictive Modeling of the NBTI Effect for Reliable Design," in *Custom Integrated Circuits Conference, 2006. CICC '06. IEEE*, 2006, pp. 189-192.

[10] R. Vattikonda, W. Wenping, and C. Yu, "Modeling and minimization of PMOS NBTI effect for robust nanometer design," in *Design Automation Conference, 2006 43rd ACM/IEEE*, 2006, pp. 1047-1052.

[11] D. Rossi, M. Omaña, C. Metra, and A. Paccagnella, "Impact of bias temperature instability on soft error susceptibility," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems* vol. 23, pp. 743 - 751, 2015.

[12] H. I. Yang, C. T. Chuang, and W. Hwang, "Impacts of Contact Resistance and NBTI/PBTI on SRAM with High-? Metal-Gate Devices," in *Memory Technology, Design, and Testing, 2009. MTDT '09. IEEE International Workshop on*, 2009, pp. 27-30.

[13] B. C. Paul, K. Kunhyuk, H. Kufluoglu, M. A. Alam, and K. Roy, "Impact of NBTI on the temporal performance degradation of digital circuits," *IEEE Electron Device Letters,* vol. 26, pp. 560-562, 2005.

[14] S. Chakravarthi, A. Krishnan, V. Reddy, C. F. Machala, and S. Krishnan, "A comprehensive framework for predictive modeling of negative bias temperature instability," in *Reliability Physics Symposium Proceedings, 2004. 42nd Annual. 2004 IEEE International*, 2004, pp. 273-282.

[15] V. G. Rao and H. Mahmoodi, "Analysis of reliability of flip-flops under transistor aging effects in nano-scale CMOS technology," in *Computer Design (ICCD), 2011 IEEE 29th International Conference on*, 2011, pp. 439-440.

[16] Y. Wang, X. Chen, W. Wang, V. Balakrishnan, Y. Cao, Y. Xie*, et al.*, "On the efficacy of input Vector Control to mitigate NBTI effects and leakage power," in *Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design*, 2009, pp. 19-26.

[17] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "Adaptive Techniques for Overcoming Performance Degradation Due to Aging in CMOS Circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 19, pp. 603-614, 2011.

[18] J. Keane, X. Wang, D. Persaud, and C. H. Kim, "An All-In-One Silicon Odometer for Separately Monitoring HCI, BTI, and TDDB," *IEEE Journal of Solid-State Circuits,* vol. 45, pp. 817-829, 2010.

[19] V. Huard, C. Parthasarathy, A. Bravaix, C. Guerin, and E. Pion, "CMOS device design-in reliability approach in advanced nodes," in *Reliability Physics Symposium, 2009 IEEE International*, 2009, pp. 624-633.

[20] M. Agarwal, V. Balakrishnan, A. Bhuyan, K. Kim, B. C. Paul, W. Wang, *et al.*, "Optimized Circuit Failure Prediction for Aging: Practicality and Promise," in *Test Conference, 2008. ITC 2008. IEEE International*, 2008, pp. 1-10.

[21] B. Halak and A. Yakovlev, "Fault-Tolerant Techniques to Minimize the Impact of Crosstalk on Phase Encoded Communication Channels," *Computers, IEEE Transactions on,* vol. 57, pp. 505-519, 2008.

[22] Y. Lin, M. Zwolinski, and B. Halak, "A Low-Cost, Radiation-Hardened Method for Pipeline Protection in Microprocessors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 24, pp. 1688-1701, 2016.

[23] S. R. Sridhara and N. R. Shanbhag, "Coding for system-on-chip networks: a unified framework," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 13, pp. 655-667, 2005.

[24] B. Halak, "Partial coding algorithm for area and energy efficient crosstalk avoidance codes implementation," *IET Computers & Digital Techniques,* vol. 8, pp. 97-107, 2014.

[25] B. Halak and A. Yakovlev, "Statistical analysis of crosstalk-induced errors for on-chip interconnects," *IET Computers & Digital Techniques,* vol. 5, pp. 104-112, 2011.

[26] T. Verhoeff, "Delay Insensitive Codes - An Overview," *Distributed Computing,* pp. 1-8, 1987.

[27] W. J. Bainbridge, W. B. Toms, D. A. Edwards, and S. B. Furber, "Delay-insensitive, point-to-point interconnect using m-of-n codes," *International Symposium on Asynchronous Circuits and Systems,* pp. 132-140, 2003.

[28] Y. Lin, M. Zwolinski, and B. Halak, "A low-cost radiation hardened flip-flop," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014, pp. 1-6.

[29] H. K. M. A. Alam, D. Varghese, and S. Mahapatra, "A comprehensive model for pmos nbti degradation: Recent progress," *Microelectronics Reliability,* vol. 47, pp. 853–862, 2007.

[30] M. Fukui, S. Nakai, H. Miki, and S. Tsukiyama, "A dependable power grid optimization algorithm considering NBTI timing degradation," *IEEE 9th International New Circuits and Systems Conference (NEWCAS),* pp. 370-373., 2011.

[31] K. Joshi, S. Mukhopadhyay, N. Goel, and S. Mahapatra, "A consistent physical framework for N and P BTI in HKMG MOSFETs," in *Reliability Physics Symposium (IRPS), 2012 IEEE International*, 2012, pp. 5A.3.1-5A.3.10.

[32] B. Halak and A. Yakovlev, "Throghput Optimisation for Area-Constrained Links with Crosstalk Avoidance Methods," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* (Forth Coming Issue).

[33] B. Halak, S. Shedabale, H. Ramakrishnan, A. Yakovlev, and G. Russell, "The impact of variability on the reliability of long on-chip interconnect in the presence of crosstalk," *International Workshop on System-Level Interconnect Prediction,* pp. 65-72, 2008.

[34] Y. Shi, S. B. Furber, J. Garside, and L. A. Plana, "Fault Tolerant Delay Insensitive Inter-chip Communication," in *Asynchronous Circuits and Systems, 2009. ASYNC '09. 15th IEEE Symposium on*, 2009, pp. 77-84.

[35] T. Assis, F. L. Kastensmidt, G. Wirth, and R. Reis, "Measuring the effectiveness of symmetric and asymmetric transistor sizing for Single Event Transient mitigation in CMOS 90nm technologies," in *2009 10th Latin American Test Workshop*, 2009, pp. 1-6.

[36] F. Lochon, "SINGLE EVENT EFFECTS TEST REPORT " *Texas Instruments Test Report,* 2010.

Proving timing properties with the Leibnizian time model

Alexei Iliasov

Newcastle University

Abstract. We present a novel approach to the description of real-time requirements in Event-B, based on the relativistic time model of Gottfried Liebniz. The approach is surprisingly useful, and has led to some significant results. We illustrate the approach with several modelling recipes for the specification of real-time systems in Event-B.

1 Introduction

In the design and modelling of systems from user specifications, it is common to find some proportion of the user requirements expressed in terms of real-time. In work on business information systems, for example, real-time requirements are a natural way to express high-level constraints on business processes [8]. In scheduling or performance analysis, real-time is the natural language for stating requirements.

We use the Event-B language [5] to explore an alternative model of time, the Leibnizian model [11]. According to Leibniz, time is not a fundamental dimension, but is used to distinguish the changes in an observed entity. In the Newtonian model time is an observable attribute of an entity, and may be used to distinguish an entity in the past from an entity in the future, even if the entities are otherwise identical. In the Leibnizian model, in which time is not a directly observable attribute, these may only be distinguished if some other observable attribute has changed. In other words, in the Leibnizian model, time-related changes are transformations of the entity itself. If nothing changes, time is not observed to pass, and therefore (to the observer) time does not pass. The Newtonian model permits time to change without a change in the observed entity.

The dichotomy of the Leibnizian model, in which two separate entities are necessary in order to define the notion of time, suggests that all the time-related properties may be isolated in the observer part leaving the part being observed to deal with functional properties. This has important practical implications: the formulation of timing constraints does not have to be notationally tied with the description of behaviour so that existing methods, semantics and tools may be employed in specifying functional properties.

This difference in time interpretation has significant consequences for the definition of a timed semantics, and for the specification of timing constraints in Event B. We present the semantic model briefly, using examples to illustrate the

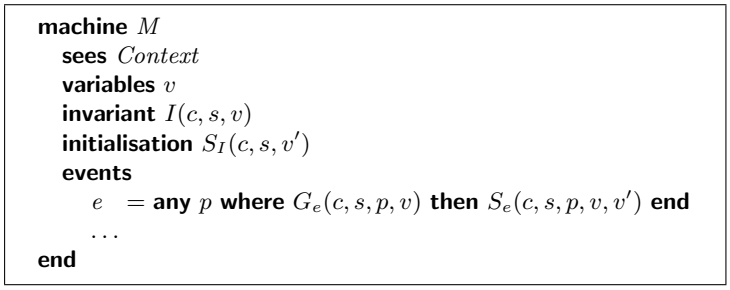


Fig. 1. Event-B model structure.

important points. We also give a series of "recipes" to show how the Leibnizian time model could be used by a model developer to introduce time into Event-B developments.

2 Background

An Event-B development starts with a compact, often trivial abstraction. The cornerstone of the Event-B method is a stepwise development that facilitates a gradual design of a complex system via a number of correctness-preserving *refinement* steps. The general form of an Event-B model (or *machine*) is shown in Fig. 1. A machine encapsulates a state space, defined by *machine variables*, and provides transitions on the state, as described by *machine events*. Events are characterised by a list of parameters p, a state predicate G called an *event guard*, and a next-state relation S.

The **invariant** clause defines the properties of a system, expressed as state predicates, that must be preserved during the system lifitem. The states defined by an invariant are called the *safe states* of a system. A correct model is proven to never leave its safe states. Data types s, constants c and relevant axioms are defined in a separate component called a *context*, and included into a machine with the **sees** clause.

The consistency of a machine as well as the correctness of refinement steps is demonstrated by discharging relevant *proof obligations* which, collectively, define the Event-B *proof semantics* [5]. The Rodin Platform [18], a tool supporting Event-B, is an integrated environment that automatically generates necessary proof obligations and providers a number of automated provers and solvers along with an interactive proof environment.

An Event-B machine defines a state transition system. Let $\Omega = \{v \mid I(c, s, v)\}$ be the (safe) states of a machine where v and $I(c, s, v)$ are the variables and the invariant of a machine. The relational form of an event e is $[e]_R \equiv \{v \mapsto v' \mid \exists p \cdot (G_e(c, s, p, v) \wedge S_e(c, s, p, v, v'))\}$.

Definition 1 (Event-B transition system). *A machine defines a transition system* (Ω, f, ω_0) *where* $f : \Omega \rightarrow \mathbb{P}(\Omega)$ *is defined as* $f = (\bigcup_e [e]_R)$*; the set of initial states* $\omega_0 \subseteq \Omega$ *is defined by the initialisation predicate* S_I*:* $\omega_0 = \{v' \mid S_I(c, s, v')\}$.

3 Leibnizian Time

In this section we formally define some essential concepts of the Leibnizian time model. We illustrate them with a timed specification of a lossless buffer, which we return to throughout the paper. For brevity, we omit the theorem proofs. Proofs and machine-checked models of the example are available at [3].

A fundamental concept is that of a process, which we define as a transition system.

Definition 2 (Process). *A process* P *is a tuple* $(\alpha\mathsf{P}, p, \iota\mathsf{P})$ *where* $\alpha\mathsf{P}$ *is a process* alphabet*,* $p \subset \alpha\mathsf{P} \times \alpha\mathsf{P}$ *is a* transition relation *and* $\iota\mathsf{P}$ *is the set of initial states.*

Time only appears when we put together two processes and let them interact in a certain way. The nature of the interaction is what intuitively may be regarded as an observation of one process by another.

Definition 3 (Observation connection). *An* observation connection *between processes* C *and* S *is a relation* $\varphi \subseteq \alpha\mathsf{S} \times \alpha\mathsf{C}$.

A timed system is formed of pair of processes where one process, an *observer*, is said to observe another process, a *subject*. In the definition above, C is an observer and S is a subject.

Definition 4 (Timed system). *An observer process* C*, a subject process* S *and an observation connection* φ *define a* timed system $\mathsf{C} \cdot \varphi \cdot \mathsf{S}$.

The first technique we give extends an untimed Event-B model to a timed system, by defining a timed observer in an associated context. We illustrate this technique in Example 1.

Recipe 1 (Event-B timed system) An timed Event-B system $\mathsf{C} \cdot \varphi \cdot \mathsf{S}$ is a pair of a machine S and context C of the following form.

machine S
  sees C
  variables v
  invariant $I(V)$
  initialisation $R(v')$
  events
    $E_i =$ **any** p_i **where**
        $G_i(p_i, v)$
      then
        $S_i(p_i, v, v')$
      end
end

context C
  sets $\alpha\mathsf{C}$
  constants $c, \varphi, \iota\mathsf{C}$
  axioms
    $\iota\mathsf{C} \subseteq \alpha\mathsf{C}$
    $c \subseteq \alpha\mathsf{C} \times \alpha\mathsf{C}$
    $\varphi \subseteq \{v \mid I(v)\} \times \alpha\mathsf{C}$
    $\ldots$
end

Subject S is an arbitrary Event-B machine defining a vector of variables v. Set $\{v \mid I(v)\}$ defines the possible states of the machine. Observer C is axiomatically defined in a context. The context defines a sort $\alpha\mathsf{C}$, a transition relation c and an observation connection φ which relates states from set $\{v \mid I(v)\}$ to observer states. Further axioms and theorems may added, to more precisely characterise the observer model. □

Example 1 (Buffer). A *lossy* buffer with the capacity to store one element of type V is defined by machine BUF, as shown below.

machine BUF
  sees def, C_0
  variables b
  invariant $b \in V$
  initialisation $b :\in V$
  events
    $wr =$ **any** v **where**
        $v \in V1$
      then
        $b := v$
      end
    $rd \ =$ **begin** $b := nil$ **end**
end

context C_0
  $\iota\mathsf{C}_0 = V$
  $c \subseteq V \times V \setminus (V1 \times V1)$
  $\varphi = V \lhd \mathrm{id}$
end

The constant $nil \in V$ and sets $V1 = V \setminus \{nil\}$, $V1 \neq \varnothing$ are defined in context def. Event wr updates the value of the stored element; event rd consumes a buffered element and sets the buffer contents to nil to indicate that the buffer is now empty. The events are always enabled and thus BUF permits arbitrary interleavings of the operations. Such operations may be implemented by unsynchronised concurrent activities. The write operation may happen arbitrary often thus potentially overwriting a previous value before it is read.

A *lossless* buffer is defined with the following timed Event-B system.

$$\mathsf{C}_0 \cdot \varphi \cdot \mathsf{BUF}$$

The observation model rules out the possibility of event wr writing into a non-empty buffer. We shall substantiate this claim in Example 2. □

An interpretation of a timed system gives a precise meaning to the phenomenon of observation. Essentially, an observation prohibits behaviours that an observer does not expect to see.

Definition 5 (Interpretation of a timed system). *Given a timed system* $\mathsf{C} \cdot \varphi \cdot \mathsf{S}$ *where* $\mathsf{S} = (\alpha\mathsf{S}, s, \iota\mathsf{S})$ *and* $\mathsf{C} = (\alpha\mathsf{C}, c, \iota\mathsf{C})$*, its interpretation is a process*

$$\mathbb{I}(\mathsf{C} \cdot \varphi \cdot \mathsf{S}) \equiv (\varphi, \tau(\mathsf{C} \cdot \varphi \cdot \mathsf{S}), (\iota\mathsf{S} \times \iota\mathsf{C}) \cap \varphi)$$

where transition relation $\tau(\mathsf{C} \cdot \varphi \cdot \mathsf{S}) \subseteq (\alpha\mathsf{S} \times \alpha\mathsf{C}) \times (\alpha\mathsf{S} \times \alpha\mathsf{C})$ *is such that a mapping* $(u \mapsto t) \mapsto (u' \mapsto t') \in (\alpha\mathsf{S} \times \alpha\mathsf{C}) \times (\alpha\mathsf{S} \times \alpha\mathsf{C})$ *belongs to* $\tau(\mathsf{C} \cdot \varphi \cdot \mathsf{S})$ *if and only if the following properties hold*

(a) $u \mapsto u' \in s$ *(a transition of a subject process)*
(b) $t \mapsto t' \in c$ *(a transition of an observer process)*
(c) $u \mapsto t, u' \mapsto t' \in \varphi$ *(subject and observer transitions are linked via the observation connection)*

One could say that an observer is a historian with a preconceived idea about subject process behaviour. An observer would not tolerate a subject that does not follow a certain plan or timetable. Note the use of $\varphi \subseteq \alpha\mathsf{S} \times \alpha\mathsf{C}$ to define the alphabet of a timed system interpretation. Whenever we speak about a timed system we always imply, unless specifically indicated otherwise, that the timed system permits an interpretation.

It is essential to note that (despite the nomenclature) the observer is an integral part of the timed system, and does not have a merely passive role. The observer characterises the timing constraints that the developer wishes to impose on an otherwise untimed system, and permits only interpretations that conform to these constraints.

Recipe 2 (Consistency) It may happen that a proof of liveness and timing properties is merely a consequence of an incompatibility between the observer and the subject process. This incompatibility results in a vacuous interpretation of a timed system that defines no common state transitions. To avoid this problem, it is sufficient to exhibit an initialisation of the timed system. For a timed system $\mathsf{C} \cdot \varphi \cdot \mathsf{S}$ one needs to prove that

$$\exists x, y \cdot x \mapsto y \in \iota\mathsf{S} \times \iota\mathsf{C} \wedge x \mapsto y \in \varphi \tag{1}$$

Condition 1 is called the *consistency proof obligation* of a timed system. □

The consistency condition holds for the system in Example 1; one possible witness is mapping $nil \mapsto nil$.

We give now the condition under which an event may be safely removed from a timed system without affecting the overall behaviour.

Recipe 3 (Relation empty) Consider a timed system $\mathsf{C} \cdot \varphi \cdot \mathsf{S}$ with Event-B machine S defining some event E_i:

$$E_i = \textbf{any } p_i \textbf{ where } G_i(p_i, v) \textbf{ then } S_i(p_i, v, v') \textbf{ end}$$

Let S' be a machine identical to S except that E_i is suppressed:

$$E_i = \textbf{any } p_i \textbf{ where } \bot \textbf{ then } S_i(p_i, v, v') \textbf{ end}$$

Timed systems $\mathsf{C} \cdot \varphi \cdot \mathsf{S}$ and $\mathsf{C} \cdot \varphi \cdot \mathsf{S}'$ are equivalent provided the following condition is satisfied

$$(\varphi[\mathsf{before}(E_i)] \times \varphi[\mathsf{after}(E_i)]) \cap c = \varnothing \tag{2}$$

where $\mathsf{before}(e)$ corresponds to the enabling states defined by an event guard and $\mathsf{after}(e)$ is a set of possible new states computed by an event:

$$\mathsf{before}(e) = \{v \mid I(v) \wedge \exists p_i \cdot G_i(p_i, v)\}$$
$$\mathsf{after}(e) = \{v' \mid I(v) \wedge \exists p_i \cdot (G_i(p_i, v) \wedge S_i(p_i, v, v'))\}$$

The technique allows one to prove that after removing event E_i the overall timed system does not become less live since the E_i is already prevented from occuring by an observer.

Example 2 (Buffer, contd.). We can apply the event removal technique to prove that timed system $\mathsf{C_0} \cdot \varphi \cdot \mathsf{BUF}$ from Example 1 does indeed define a lossless buffer.

To make the buffer lossless, we need to rule out the posibility of event *wr* writing into a non-empty buffer. That is, event *wr* should not happen when $b \neq nil$. Event *wr* may be represented (via a trivial case of refinement) by the following two events.

$$wr = \textbf{refines } wr \textbf{ any } v \textbf{ where } \mathbf{b = nil} \wedge v \in V1 \textbf{ then } b := v \textbf{ end}$$
$$owr = \textbf{refines } wr \textbf{ any } v \textbf{ where } \mathbf{b \neq nil} \wedge v \in V1 \textbf{ then } b := v \textbf{ end}$$

It is possible to prove that *owr* is not a part of the timed system $\mathsf{C_0} \cdot \varphi \cdot \mathsf{BUF}$ by showing that Condition 2 holds for *owr*:

$$(\varphi[\mathsf{before}(owr)] \times \varphi[\mathsf{after}(owr)]) \cap c = \varnothing$$

which expands to $\varphi[\{b \mid b \in V1 \wedge (\exists v \cdot v \in V1)\}] \times \varphi[\{b' \mid b \in V1 \wedge (\exists v \cdot v \in V1 \wedge b' = v)\}] \cap c = \varnothing$. Since $V1$ is not empty we have that $\exists v \cdot v \in V1 \Leftrightarrow \top$ and also $V1 = \{b \mid b \in V1\}$. The condition simplifies to $\varphi[V1] \times \varphi[V1] \cap c = \varnothing \Leftrightarrow \varphi[V1] \times \varphi[V1] \cap (V \times V \setminus (V1 \times V1)) = \varnothing \Leftrightarrow \top$. Hence, we can replace machine BUF in $\mathsf{C_0} \cdot \varphi \cdot \mathsf{BUF}$ with the following machine $\mathsf{BUF'}$:

machine BUF′
  ...
  events
    wr = **any** v **where** $\mathbf{b = nil} \wedge v \in V1$ **then** $b := v$ **end**
    rd = **begin** $b := nil$ **end**
end

It is trivial to see that $\mathsf{BUF'}$ defines a lossless buffer. Hence, $\mathsf{C_0} \cdot \varphi \cdot \mathsf{BUF}$ is also a lossless buffer. □

It is often advantageous to deal with an observer that is cooperative enough to completely accept any execution of a subject process. Then one knows a priori that something happens in a subject process for every possible point of time defined by an observer.

Definition 6 (Strictness). *A timed system* $\mathcal{A} = (\alpha\mathsf{C}, c, \iota\mathsf{C}) \cdot \varphi \cdot (\alpha\mathsf{S}, s, \iota\mathsf{S})$ *is strict if for every* $u \mapsto t \in \alpha\mathsf{S} \times \alpha\mathsf{C}$ *and* $t \mapsto t' \in c$ *there exists some* u' *such that* $(u \mapsto t) \mapsto (u' \mapsto t') \in \tau\mathcal{A}$ *and* $\iota\mathsf{C} \subseteq \varphi[\iota\mathsf{S}]$.

In a system with a strict observer, an observation connection is also a simulation relation [3].

Example 3 (Buffer, contd.). Observer C_0 permits a concise abstraction however there is an even simpler observer that achieves the same effect. Notice that $\mathsf{C_0} \cdot \varphi \cdot \mathsf{BUF}$ defines three transitions classes: reading a value and setting buffer to 0 ($V^+ \times \{nil\}$); reading an empty buffer ($\{nil \mapsto nil\}$); writing into an empty buffer ($\{nil\} \times V^+$). We shall exploit this property and define a new observer C_1 such that these three classes are the kernels of new observation connection φ_1:

context $\mathsf{C_1}$
  extends *def*
  sets $\alpha\mathsf{C_1}$
  constants $c_1, \varphi_1, \iota\mathsf{C_1}, \mathrm{E}, \mathrm{F}$
  axioms
    $\mathrm{partition}(\alpha\mathsf{C_1}, \{\mathrm{E}, \mathrm{F}\})$
    $\iota\mathsf{C_1} = \{\mathrm{E}, \mathrm{F}\}$
    $c_1 = \{\mathrm{E} \mapsto \mathrm{E}, \mathrm{E} \mapsto \mathrm{F}, \mathrm{F} \mapsto \mathrm{E}\}$
    $\varphi_1 = V1 \times \{\mathrm{F}\} \cup \{nil\} \times \{\mathrm{E}\}$
end

It is not hard to see that event removal condition also holds for $\mathsf{C_1} \cdot \varphi_1 \cdot \mathsf{BUF}$: $\varphi_1[\mathsf{before}(owr)] \times \varphi_1[\mathsf{after}(owr)] \cap c_1 = \varnothing \Leftrightarrow (\{\mathrm{F}\} \times \{\mathrm{F}\}) \cap c_1 = \varnothing \Leftrightarrow \top$. It is easy to see that, unlike $\mathsf{C_0} \cdot \varphi \cdot \mathsf{BUF}$, system $\mathsf{C_1} \cdot \varphi_1 \cdot \mathsf{BUF}$ is *strict*. □

The fourth recipe allows a developer to show that a state which is possible in the untimed process is ruled out by the timing constraints. We give the theory of the technique and demonstrate it with a simple example.

Recipe 4 (Point empty) Consider a timed system $\mathsf{C} \cdot \varphi \cdot \mathsf{S}$ and a subject state $w \in \alpha\mathsf{S}$. If one can show that φ does not project w into anything at all in $\alpha\mathsf{C}$ then, by the Definition 5 of timed system interpretation, any state $\chi \in \alpha\mathsf{C} \times \alpha\mathsf{S}$ where $\mathrm{prj}_2[\{\chi\}] = \{w\}$ is not a state of $\mathsf{C} \cdot \varphi \cdot \mathsf{S}$.

Thus, a subject state not projected by φ is not reachable in a timed system. A proof that assumes the existence of such a state may be discharged by deriving a contradiction with the following rule.

$$\forall W \cdot W \subseteq \Omega \wedge \varphi[W] = \varnothing \Rightarrow \bot \tag{3}$$

where $\Omega = \{v \mid I(v)\}$ is the set of subject states. □

Example 4 (Mutex). In this example we describe a very simple mutual exclusion algorithm that works due to a rigid scheduling of the involved threads. The state of a thread p is defined by $s(p)$ and is one of the following values: 'out', denoting

that p is outside of a critical section and not trying to enter it; 'prep', telling that the thread is about to enter the critical section; and 'in' for the states when the thread is in the critical section.

machine MTX
  variables s
  invariant
    inv1 : $s \in P \to \{\text{out}, \text{prep}, \text{in}\}$
    inv2 : $\text{card}(s^{-1}[\{\text{in}\}]) \leq 1$
  initialisation $s := P \times \{\text{out}\}$
  events
    prepare = **any** p **where** $p \in P \wedge s(p) = \text{out}$ **then** $s(p) := \text{prep}$ **end**
    enter = **any** p **where** $p \in P \wedge s(p) = \text{prep}$ **then** $s(p) := \text{in}$ **end**
    leave = **any** p **where** $p \in P \wedge s(p) = \text{in}$ **then** $s(p) := \text{out}$ **end**
end

where set P of processes is finite. Invariant inv2 expresses the property of mutual exclusion. We employ the following observer process to define that no two processes may be, at the same time, at stages 'prep' and 'in':

context C
  ...
  $c \subseteq \alpha\mathsf{C} \times \alpha\mathsf{C}$
  $S = \mathbb{P}(P \times \{\text{out}, \text{prep}, \text{in}\})$
  $\varphi \subseteq S \times \alpha\mathsf{C}$
  axm5 : $\forall t, q \cdot t, q \in P \wedge t \neq q \Rightarrow [\![s(t) = \text{prep} \wedge s(q) = \text{in}]\!] = \varnothing$
end

where $[\![P(\omega)]\!] \equiv \varphi[\{\omega \mid P(\omega)\}]$. The only non-trivial proof obligation in this model is the preservation of inv2 by event *enter*. It asks to prove, for some process p, that entering the critical does not violate safety invariant inv2.

$$\text{card}(s^{-1}[\{\text{in}\}]) \leq 1 \wedge s(p) = \text{prep} \vDash \text{card}((s \lhd\!\!\!\!- \{p \mapsto \text{in}\})^{-1}[\{\text{in}\}]) \leq 1$$

The condition cannot be discharged within the scope of the subject model alone. We need to bring in the constraints of the observer model to demonstrate the condition. We proceed by replacing $\text{card}((s \lhd\!\!\!\!- \{p \mapsto \text{in}\})^{-1}[\{\text{in}\}]) \leq 1$ with a stronger goal $s^{-1}[\{\text{in}\}] = \varnothing$ and continue with a proof by contradiction. The negation of $s^{-1}[\{\text{in}\}] = \varnothing$ in hypothesis gives

$$s(p) = \text{prep} \wedge s(x) = \text{in} \wedge x \neq p \vDash \bot$$

A state where one process is in the critical section and the other is about to enter the critical section is disallowed by the observer (axm5) so that the point empty technique may be used to discharge the condition. Instantiating axm5 with $t = p, q = x$ we have $\varphi[\{a \cdot a \in S \wedge a(p) = \text{prep} \wedge a(x) = \text{in} \mid a\}] = \varnothing$ which gives us set W to instantiate Condition 3 and derive a contradiction in hypothesis.

One way to realise observer C is by defining it to be cyclic scheduler that allows processes to access the critical section at fixed time intervals. □

4 Realisability

According to Definition 5, a timed system is a transition (or a process, as it is defined in Definition 2). Hence, a timed system may itself be employed in the role of subject or observer and one can define a complex timed system made of subsystems which are also timed systems. One application of the compositionality property is a structure called *time log*. A time log is timed system observed by an external observer. Informally, the external observer make a record of observation using its own timekeeping device.

Definition 7 (Time log). *Time log* $\mathbb{T}(\mathsf{C}\cdot\varphi\cdot\mathsf{S})_{\mathsf{T},\omega}$ *of timed system* $\mathcal{A} = \mathsf{C}\cdot\varphi\cdot\mathsf{S}$ *is a process*

$$\mathbb{T}(\mathcal{A})_{\mathsf{T},\omega} = (\varphi; \omega, L[\tau(\mathsf{T}\cdot\omega\cdot\tau(\mathcal{A}))], \iota\mathsf{T})$$

where $\mathsf{T}\cdot\omega\cdot\tau(\mathcal{A})$ *is strict,* ω *is total and functional; projection* L *removes states of process* C*:* $L[X] = \{((a,b),\{c\}) \mapsto (a,c) \mid ((a,b),\{c\}) \in X\}$*. Also,* $L[\iota\mathcal{A}\times\iota\mathsf{T}]\cap\omega \neq \varnothing$*.*

A time log defines a timed system (or a process) which does not reference states of observer C. A time log is itself a transition system hence it is sometimes possible to replace a timed system with its time log. One common reason to do this is to separate the proof of logical properties relevant to timing constraints from the proof of how these properties may be expressed in a specific, implementation-oriented form, e.g., hard real-time constraints.

Not all timed system may be realised in physical reality. If the object of a formal development is a piece of software of hardware it is necessary to check, at the level of a concrete design, that certain properties are respected by an observer process. These properties, called realisability conditions, are as follows:

- time advance is monotonic
- infinite subject activities take infinitely long time to observe

Instead of checking these properties directly on an observer model, it is more convenient to consider yet another observer and study the observations defined by the new observer and the original timed system. The first realisability condition demands that the time model described in an observer process may be mapped to a monotonic time model. The second condition prohibits a situation where an unterminating activity of a subject process is timed to terminate by a certain deadline.

An animation is a time log satisfying the the realisability conditions. Let bounded(X,Y) denote the fact that there exist lower and upper bounds w.r.t. the relation of process $\mathsf{Y} = (\alpha\mathsf{Y}, <)$, bounded$(X) \equiv X \subseteq \alpha\mathsf{Y} \wedge (\exists l, u \cdot l, u \in \alpha\mathsf{Y} \wedge \forall p \cdot p \in P \Rightarrow l < p < b)$.

Definition 8 (Animation of timed system). *An animation of* $\mathsf{C} \cdot \varphi \cdot \mathsf{S}$ *is a time log* $\mathbb{T}(\mathsf{C} \cdot \varphi \cdot \mathsf{S})_{\mathsf{T},\omega}$ *such that* T *is monotone and every bounded subset of* $\alpha\mathsf{T}$ *maps to a finite sequence of subject actions:* $\forall P \cdot \text{bounded}(P, \mathsf{T}) \Rightarrow \text{finite}(\omega^{-1}; \varphi^{-1}[P])$.

Definition 9 (Realisability). *A timed system is realisable if it admits at least one animation.*

According to Definition 5, a timed system is a transition (or a process, as it is defined in Definition 2). Hence, a timed system may itself be employed in the role of subject or observer and one can define a complex timed system made of subsystems which are also timed systems. One application of the compositionality property is a structure called an *animation*. An animation is a timed system formed by observed another timed system. Informally, the external observer make a record of observation using its own timekeeping device.

Definition 10 (Animation). *Animation* $\mathbb{T}(\mathsf{C} \cdot \varphi \cdot \mathsf{S})_{\mathsf{T},\omega}$ *of timed system* $\mathcal{A} = \mathsf{C} \cdot \varphi \cdot \mathsf{S}$ *is a process*

$$\mathbb{T}(\mathcal{A})_{\mathsf{T},\omega} = (\varphi; \omega, L[\tau(\mathsf{T} \cdot \omega \cdot \tau(\mathcal{A}))], \iota\mathsf{T})$$

where $\mathsf{T} \cdot \omega \cdot \tau(\mathcal{A})$ *is strict,* ω *is total and functional; projection* L *removes states of process* C*:* $L[X] = \{((a,b),\{c\}) \mapsto (a,c) \mid ((a,b),\{c\}) \in X\}$. *Also,* $L[\iota\mathcal{A} \times \iota\mathsf{T}] \cap \omega \neq \varnothing$.

An animation defines a timed system (or a process) which does not reference states of observer C. An animation is itself a transition system hence it is sometimes possible to replace a timed system with its animation. One common reason to do this is to separate the proof of logical properties relevant to timing constraints from the proof of how these properties may be expressed in a specific, implementation-oriented form, e.g., hard real-time constraints.

Recipe 5 (Real-time constraints) The form of timing constraints of a concrete design is dictated by the practical necessity to validate constraints via some form of static analysis (i.e., worst-case execution time) or by observing execution runs of a software or hardware implementation. For the former, it may be necessary to present constraints in the form of durations of elementary execution steps.

In a timed Event-B we suggest to use an observer based on a dense linear order (DLO) to realise real-time constraints. A linear order c is dense if it satisfies condition $\forall x, y \cdot x \mapsto y \in c \Rightarrow (\exists z \cdot x \mapsto z \in c \wedge z \mapsto y \in c)$. Intuitively, with a dense order one is able to define durations and time points with an arbitrary precision[1]. It also means one is able to interpolate between any two time points. An example of an Event-B model of a DLO may be found in [1].

Assume a process $\mathsf{C} = (\alpha\mathsf{C}, c, \iota\mathsf{C})$ where c is a DLO. This process will be employed to define an animator for some timed system $\mathcal{A}$. We introduce a layer

[1] This is what, we believe, is usually meant as a property distinguishing a "real-valued clock" from a "discrete" clock.

of syntactic shorthand to express properties of C. Let $t \in \alpha\mathsf{C}$ be the current state of animator C, P a predicate defined on set $\gamma^{-1}[\alpha\mathsf{C}]$ and γ an animation relation.

- $\mathsf{at}(P,t)_\gamma \equiv P(\gamma^{-1}(t))$: event (defined by predicate) P happens at time t;
- $\mathsf{during}(P,i)_\gamma \equiv \exists t \cdot t \in i \Rightarrow \mathsf{at}(\mathrm{s},t)_\gamma$: event P happens at least once during time interval i;
- $\mathsf{within}(\Delta,P,t)_\gamma \equiv \exists t' \cdot t' > t \wedge t' - t \leq \Delta \wedge P(\gamma^{-1}(t'))$: event P happens within Δ time units after t;
- $\mathsf{after}(\Delta,P,t)_\gamma \equiv \neg\mathsf{within}(\Delta,P,t)_\gamma \wedge \mathsf{within}(\infty,P,t)_\gamma$: event P happens not sooner than Δ time units (but happens eventually) after t.

The following statement says that, at all times, whenever there happens a stimulus event ($\mathbf{s}$), it is followed by a response event ($\mathbf{r}$) within Δ time units

$$\forall t \cdot t \in \alpha\mathsf{C} \wedge \mathsf{at}(\mathbf{s},t)_\gamma \Rightarrow \mathsf{within}(\Delta,\mathbf{r},t)_\gamma$$

One may elect to be more precise and state that, should a stimulus happen at time t, a response follows within interval $[t+\Delta_1, t+\Delta_2]$

$$\forall t \cdot t \in \alpha\mathsf{C} \wedge \mathsf{at}(\mathbf{s},t)_\gamma \Rightarrow \mathsf{during}([t+\Delta_1, t+\Delta_2],\mathbf{r})_\gamma$$

which is the same as

$$\forall t \cdot t \in \alpha\mathsf{C} \wedge \mathsf{at}(\mathbf{s},t)_\gamma \Rightarrow \neg\mathsf{within}(\Delta_1,\mathbf{r},t)_\gamma \wedge \mathsf{within}(\Delta_2,\mathbf{r},t)_\gamma$$

As a further illustration, the following are statements about a simple traffic light.

- $A_\gamma(t) \equiv \mathsf{within}(\infty,\mathbf{green},t)_\gamma$: the green aspect is eventually lit;
- $B_\gamma(t) \equiv \neg\mathsf{within}(\infty,\mathbf{green}\wedge\mathbf{red},t)_\gamma$: the green and red aspects are never lit at the same time;
- $C_\gamma(t) \equiv \mathsf{at}(\mathbf{red},t) \Rightarrow \mathsf{after}(\Delta_1,\mathbf{green},t)_\gamma$: the green aspect follows the red aspect within Δ_1 time units;
- $D_\gamma(t) \equiv \mathsf{at}(\mathbf{red},t) \Rightarrow \neg\mathsf{after}(\Delta_2,\mathbf{yellow},t)_\gamma$: the yellow aspect might be lit after red aspect might happen but not sooner than Δ_2 time units.

Any traffic light implementation must respect these properties. In an animator specification this is expressed by addign an axiom that animator C may never violate these properties:

$$\iota\mathsf{C} \subseteq \mathbf{v}_\gamma \qquad c[\mathbf{v}_\gamma] \subseteq \mathbf{v}_\gamma \tag{4}$$

For the traffic light example, it must hold that the initial states $\iota\mathsf{C}$ satisfies properties $A-D$ and the animation relation c preserves properties $A-D$ so that $\mathbf{v}_\gamma$ is the set of all valid time points: $\mathbf{v}_\gamma = \{t \mid t \in \alpha\mathsf{C} \wedge A_\gamma(t) \wedge B_\gamma(t) \wedge C_\gamma(t) \wedge D_\gamma(t)\}$.

The verification effort is in showing that an animator does indeed animate a given timed system. For this we consider the animation relation γ connecting the timed system observer with the animator C. If one can prove that γ exists as an

animation relation that one may take the animation as a timed specification that respects both the abstract scheduling properties of the original timed system and the real-time constraints of the animator. □

Example 5 (Buffer, contd.). In this example we show how to construct an animation of the lossless buffer timed system $\mathsf{C_1} \cdot \varphi_1 \cdot \mathsf{BUF}$ in the terms of the relative speeds of the write and read operations. For this, we reinterpret the timing requirements with an animator that explicitly defines operation delays and time-outs.

Consider a DLO animator $\mathsf{T} = (\alpha\mathsf{T}, <, \{zero\})$ and animation relation $\omega \in \{\mathrm{E}, \mathrm{F}\} \to \alpha\mathsf{T}$ such that for all $x, y \in \alpha\mathsf{T}$ it holds that

(a) $P_1(t) \equiv \mathsf{within}(\Delta_R, \lambda c \cdot c = \mathrm{E}, t)_\omega$ (a read happens within Δ_R time units)
(b) $P_2(t) \equiv \neg\mathsf{within}(\Delta_W, \lambda c \cdot c = \mathrm{F}, t)_\omega$ (a write happens not sooner than Δ_W time units from now)
(c) $P_3 \equiv \Delta_R \leq \Delta_W$ (reader is quicker than writer)
(d) $\omega[\{\mathrm{E}\}] = \{zero\}$ (system starts at time $zero \in \alpha\mathsf{T}$)

In this example, properties $P_1 - P_3$ also sufficiently constrain the animation relation γ. Delays Δ_R and Δ_W define the durations of **wr** and **rd**. We prove that T animates $\mathsf{C_1} \cdot \varphi_1 \cdot \mathsf{BUF}$ with animation connection ω. Let $\mathbf{v}_\gamma = \{t \mid P_1(t) \wedge P_2(t)\}$ (we omit P_3 as it is not time-sensetive) and *less* be a prefix form of $<$. From Condition 4 we derive the following hypothesis

$$\{zero\} \subseteq \mathbf{v}_\omega \qquad less[\mathbf{v}_\omega] \subseteq \mathbf{v}_\omega$$

The initialisation condition expands to

$$\begin{array}{l}(\exists t' \cdot t' > zero \wedge t' - zero \leq \Delta_R \wedge \omega^{-1}(t') = \mathrm{E}) \wedge \\ \neg(\exists t' \cdot t' > zero \wedge t' - zero \leq \Delta_W \wedge \omega^{-1}(t') = \mathrm{F})\end{array} \tag{5}$$

Statement $less[\mathbf{v}_\omega] \subseteq \mathbf{v}_\omega$ gives

$$\begin{array}{l}\forall t_0, t_1 \cdot t_0 t_1 \in \alpha\mathsf{T} \wedge t_0 < t_1 \wedge \\ ((\exists t' \cdot t' > t_0 \wedge t' - t_0 \leq \Delta_R \wedge \omega^{-1}(t') = \mathrm{E}) \wedge \\ \neg(\exists t' \cdot t' > t_0 \wedge t' - t_0 \leq \Delta_W \wedge \omega^{-1}(t') = \mathrm{F})) \Rightarrow \\ \quad ((\exists t' \cdot t' > t_1 \wedge t' - t_1 \leq \Delta_R \wedge \omega^{-1}(t') = \mathrm{E}) \wedge \\ \quad\quad \neg(\exists t' \cdot t' > t_1 \wedge t' - t_1 \leq \Delta_W \wedge \omega^{-1}(t') = \mathrm{F}))\end{array} \tag{6}$$

We sketch the proof for the fact that ω is an animation relation: for any $t < t'$ it holds that $\omega^{-1}(t) < \omega^{-1}(t')$. The only case when $\omega^{-1}(t) \geq \omega^{-1}(t')$ is $\omega^{-1}(t) = \mathrm{F}$ and $\omega^{-1}(t') = \mathrm{F}$. Assume that such t and t' exist. From Condition 5 there exists $t_0 < t$ and $\omega^{-1}(t_0) = \mathrm{E}$. From Condition 6 we have the existence of t_1 such that $t_1 > t_0 \wedge t_1 - t_0 \leq \Delta_R \wedge \omega^{-1}(t_1) = \mathrm{E}$ and $t_1 > t'$. The $t_1 > t'$ part may be shown by induction: since t_0 and t are a finite distance apart there exists $t_i, \omega^{-1}(t_i) = \mathrm{E}$ such that there does not exist $t_{i+1}, \omega^{-1}(t_i) = \mathrm{E}$ and $t_{i+1} < t$. Also, from Condition 6, we have that $t' - t \geq \Delta_W$ which leads to a contradiction due to condition (c). □

Recipe 6 (Point merge) This technique is a generalisation of the empty point technique. It is used to derive a contradiction when a subject state, defined by the intersection of states of two or more concurrent threads disagrees with the observation model. The following lemma states how to make a transition from a set of statements about individual thread states to a statement about a time point when such a state configuration may be observed.

Lemma 1 (Point merge). *Let $\mathcal{W}$ and $\mathcal{P}_i$ be non-empty subject process states such that $\mathcal{W} = \{v \mid W(v)\}$, $\mathcal{P}_i = \{v \mid P_i(v)\}$ where $W(v)$ and $P_i(v)$ are predicates over subject process state space and it holds that $W \Rightarrow \bigwedge_i P_i$. Then there exist time points $t_i \in [\![P_i]\!] \cap [\![W]\!]$ such that $\forall i, j \cdot t_i = t_j$.*

Proof. See [3] (a Rodin Toolkit proof).

The proof technique is to show that no two states from P_i and P_j, $i \neq j$ may be observed at the same time (due to some timing conditions). Then the existence of a time point common for the two states P_i and P_j gives a contradiction.

We have applied the point merge technique in the proof of Fischer's timing-based algorithm of mutual exclusion [19, 4]. The complete Event-B development of the algorithm is available at [2]. □

5 Related Work

One closely related work is that of Abadi and Lamport [4] which shows that timing constraints may be expressed directly in TLA without syntactic or semantic extensions. Timed automata [6] offers a formal framework for specifying real-time properties by enriching the state of an automata with a number real-valued clocks. The UPPAAL[7] tool offers support for automated verification of timed automata. Timed process algebras have been researched extensively and there is a large variety of notations and semantics. The timed extensions of CSP [19] and CCS[16] are two notable examples.

Although Event-B lacks any native support of time, some form of timed modelling may be done directly in the Event-B notation. The basic principle - a clock variable employed to keep track of time - is fairly intuitive and has been applied in various state-based and proces algebraic methods. One example is Tock-CSP [19] which uses the standard CSP notation and measures the passage of time by counting the occurence of a *tock* event. A state-based equivalent is having a dedicated variable *now* to track the passage of time and express timing conditions [4, 15].

Previous work on modelling time in B uses a clock variable which records the current value of a clock, and an operation is given to advance time [9]. This approach is taken up again for Event B in [10, 17]. In [14] the concept of time is embedded into the B notation itself. Time is modelled by equipping a machine with a clock and assuming that an event execution is not instanteneous.

We briefly discuss the general ideas behind a clock variable technique and show how it may related to our approach. A machine with a clock variable t has the following form

machine m
  variables v, t
  invariant $I(v) \wedge P(v,t)$
  initialisation $R(v', t')$
  events
    sys = **when** $G(v,t)$ **then** $S(v,t,v')$ **end**
    $tick$ = **when** $H(v,t)$ **then** $T(t,v,t')$ **end**
end

Timing constraints are encoded as a safety invariant $P(v,t)$. A clock variable t is usually defined as $t \in \mathbb{N}$ to imply an unbounded discrete clock. The clock variable is updated by event $tick$; an update would either increment t by one or 'jump' time to some interesting point in future (i.e., next deadline). The behaviour of a system is then cumulatively defined by some event sys which may not update t but may refer to t in its guard. An informal interpretation is the following: if activity sys must happen within interval $[a(v), b(v)]$, guard $G(v,t)$ should not allow sys happen before $a(v)$ while the clock guard $H(v,t)$ should prevent time from progressing beyond $b(v)$ until sys has happened. Sometimes intervals are singular and one speaks about deadlines [17].

Verification conditions for time properties are invariant preservation theorems for $P(v,t)$:

$$I(v) \wedge P(v,t) \wedge G(v,t) \wedge S(v,t,v') \Rightarrow P(v',t)$$
$$I(v) \wedge P(v,t) \wedge H(v,t) \wedge T(t,v,t') \Rightarrow P(v,t')$$

On the left-hand side, $P(v,t)$ is stated on an old state and on the right-hand side on a new state produceed by respective events. What such theorems show is that, if properly initialised, the system is guaranteed to stay within the bounds set by predicate $P(v,t)$.

This technique allows one to demonstarte a range of progress properties with a heavy reliance on Event-B refinement principles. In Event-B one is able to refine a previously atomic transition into a sequence or a terminating loop of new transitions. Atomicity refinement - as this technique is known - allows one to prove certain progress properties by constructing suitable refinement relations. For instance, one can prove that activity A completes before activity B (each comprising several events) by showing that A and B are derived from abstract events a and b and, at that abstraction level, it is somehow known that a always precedes b. Such kind of properties of a and b may be demonstrated by encoding an ordering relation with an auxiliary variable or generating a special proof obligation [12, 13]. A model with a clock variable restates the atomicity refinement technique in terms of deadlines and intervals.

It is easy to convert the tick model into Leibnizian time. Rather than giving a timed system that defines an equivalent transitions system we define a timed system that, as we believe, corresponds to the intended purpose of an Event-B

machine with a *tick* event. Let $\Upsilon = \{t \mid \exists v \cdot P(v,t)\}$ and $\Omega = \{v \mid \exists t \cdot I(v) \wedge P(v,t)\}$. Then machine m corresponds to a timed system $\mathsf{C} \cdot \varphi \cdot \mathsf{S}$ such that

- $\mathsf{C} = (\Upsilon, c, \iota\mathsf{C})$ where $c \subseteq \Upsilon \times \Upsilon$ and $c \cap \mathrm{id}(\Upsilon) = \varnothing$
- $\mathsf{S} = (\Omega, s, \iota\mathsf{S})$ where $s = \{v \mapsto v' \mid \exists t \cdot G(v,t) \wedge S(v,t,v')\}$
- $\varphi = \{v \mapsto t \mid G(v,t) \wedge \neg H(v,t)\} \cap \{(v,t) \mid P(v,t)\}$

Note that $\mathsf{C} \cdot \varphi \cdot \mathsf{S}$ does not depend on the definition of $T(t,v,t')$. This is because in an Event-B model there is no meaning to $T(t,v,t')$ in the sense that no proof obligation constraints $T(t,v,t')$ beyond requiring that $T(t,v,t')$ is safe and irreflexive (should be imposed by the proof of convergence of *tick*) and $T(t,v,t') \subseteq \{(v,t) \mid P(v,t)\} \times \{(v,t) \mid P(v,t)\}$. The guard $H(v,t)$ of *tick* potentially matters as it would be a part of the deadlock freeness condition. In practice, the convergence and deadlock freeness of *tick* are hard to prove and are rarely attempted. In the Leibnizian time model the $P(v,t)$ constraint is placed in the observation connection and the irreflexivity property is a part of the observer model leaving subject S to contend with functional properties.

6 Discussion

We have presented a summary of our ideas on how the Leibnizian model of time may be used to construct timed Event-B specifications. Our approach offers a homogenous technique to time modelling where properties of timed models are expressed and proven in a gradual, refinement-based manner. The approach is a conservative extension of Event-B. No notational or semantical changes are necessary and the existing modelling tools have proven adequate.

Our technique does not dictate any specific time domain: we let a modeller choose the most appropriate abstraction of time – a simple scheduler, a fictious integer clock or a dense time clock. Both dense and discrete time domains are supported so that the approach may be used as a part of a toolchain with a wide range of potential roles including expressing scheduling properties and hard real-time constraints. The approach has proven to be quite effecient and intuitive: we were able to tackle several large case studies and, as far as we are aware, our models are simpler and require a lower verification effort while all proofs are completely machine-checked.

Due to space constraints, we did not present a larger case study although one such case study is available at [2]. Many recipes were not discussed. These include rules for demonstrating the realisability of a timed specification and several refinement-related recipes. We plan to provide a plug-in to the Rodin Toolkit [18] for automated generation of the timed systems proof obligations and a template-based assistant for constructing various kinds of observer processes.

References

1. A. Iliasov and J. Bryans. Dense linear order; An Event-B context encoding. online at http://www.iliasov.org/fischer/observer.pdf.

2. A. Iliasov and J. Bryans. Event-B development of Fischer's algorithm. online at http://iliasov.org/fischer.
3. A. Iliasov and J. Bryans. Supplementary material: proofs and models. online at http://iliasov.org/ltime.
4. M. Abadi and L. Lamport. An old-fashioned recipe for real time. In *Proceedings of the Real-Time: Theory in Practice, REX Workshop*, London, UK, UK, 1992. Springer-Verlag.
5. J.-R. Abrial. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, 2010.
6. R. Alur and D. L. Dill. Automata for modeling real-time systems. In *Proceedings of the seventeenth international colloquium on Automata, languages and programming*. Springer-Verlag New York, Inc., 1990.
7. J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, and W. Yi. Uppaal - a tool suite for automatic verification of real-time systems. In *Proceedings of the DIMACS/SYCON workshop on Hybrid systems III : verification and control*, pages 232–243. Springer-Verlag New York, Inc., 1996.
8. J. W. Bryans, J. S. Fitzgerald, A. Romanovsky, and A. Roth. Patterns for modelling time and consistency in business information systems. In *15th IEEE International Conference on Engineering of Complex Computer Systems*. IEEE Computer Society, 2010.
9. M. Butler and J. Falampin. An approach to modelling and refining timing properties in B. In *Refinement of Critical Systems (RCS)*, January 2002.
10. D. Cansell, D. Méry, and J. Rehm. Time Constraint Patterns for Event B Development. In *Formal Specification and Development in B, 7th International Conference of B Users*, 2007.
11. M. J. Futch. *Leibniz's Metaphysics of Time and Space*. Springer-Verlag GmbH, 2008.
12. Thai Son Hoang and Jean-Raymond Abrial. Reasoning about liveness properties in event-b. In *ICFEM*, pages 456–471, 2011.
13. A. Iliasov. Use case scenarios as verification conditions: event-b/flow approach. In *Proceedings of the Third international conference on Software engineering for resilient systems*, SERENE'11, 2011.
14. K. Lano. *The B Language and Method: A Guide to Practical Formal Development*. Springer-Verlag New York, Inc., 1996.
15. Nancy Lynch and Frits Vaandrager. Forward and backward simulations - part ii: Timing-based systems. *Information and Computation*, 128.
16. F. Moller and C. Tofts. A temporal calculus of communicating systems. In *Proceedings on Theories of concurrency : unification and extension: unification and extension*, CONCUR '90, pages 401–415. Springer-Verlag New York, Inc., 1990.
17. J. Rehm. A method to refine time constraints in event B framework. In *AVoCS*, 2006.
18. RODIN. Event-B Platform. http://www.event-b.org/, 2009.
19. S. Schneider. *Concurrent and Real Time Systems: The CSP Approach*. John Wiley & Sons, Inc., New York, NY, USA, 1999.

Regions of Affine Nets

Jetty Kleijn[1], Maciej Koutny[2], and Marta Pietkiewicz-Koutny[2]

[1] LIACS, Leiden University, PO Box 9512, 2300 RA, The Netherlands
[2] School of Computing Science, Newcastle University, NE1 7RU, UK

Abstract. Regions of transition systems provide a versatile and effective tool for the synthesis of Petri nets from behavioural specifications. Intuitively, a region captures a single net place through essential behavioural characteristics as encoded in the transition system, including marking information and its connectivity with all the transitions. One of the key advances in the design of region based solutions for a variety of synthesis problems has been the development of a general approach for dealing with region based synthesis of Petri nets. It is founded on so-called τ-nets and corresponding τ-regions.
In this paper, we discuss a region based synthesis procedure for affine nets, a class of Petri nets, in which the number of tokens produced by firing transitions depends linearly on the current marking. We then show that the notion of a τ-region can be suitably adapted to fit the semantics of affine nets.
Keywords: concurrency, theory of regions, transition system, synthesis problem, Petri net, affine net, localities, locally maximal step semantics

1 Introduction

The intended or observed behaviour of a concurrent system may be captured using a step transition system such as that depicted in Figure 1. It has six different states, including the initial state *init*, and a number of directed arcs labelled by multisets of executed actions representing possible transitions among these states.

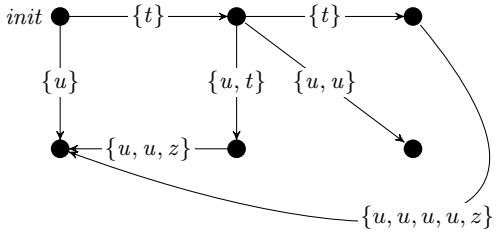


Fig. 1. A step transition system which cannot be generated by a PT-net.

Suppose that one would like to construct a Place/Transition net (PT-net) net N with its concurrent reachability graph isomorphic to the step transition system in Figure 1. Such an attempt would fail for the following reasons:

- The presence of an arc labelled by $\{u, t\}$ necessarily implies the presence of an arc outgoing from the same state labelled by $\{u\}$, and that is not true of the step transition system in Figure 1.
- The effect of executing $\{t\}\{u, t\}\{u, u, z\}$ and $\{t\}\{t\}\{u, u, u, u, z\}$ should in N be the same as both step sequences lead from the initial state to the same state of the transition system. Hence, assuming that W is the weight function of N, for every place p we would have:

  $$3 \cdot (W(u,p) - W(p,u)) + 2 \cdot (W(t,p) - W(p,t)) + (W(z,p) - W(p,z)) = \\ 4 \cdot (W(u,p) - W(p,u)) + 2 \cdot (W(t,p) - W(p,t)) + (W(z,p) - W(p,z)) \; .$$

  As a result, $W(u,p) = W(p,u)$ which means that executing u would not change the marking of N, a contradiction with the fact that executing $\{u\}$ in the initial state leads to a different state.

The latter of the above two problems is related to the fact that arc weights in PT-nets are constant and, as we demonstrated above, no net model with this property can generate the step transition system in Figure 1. We therefore need a more expressive model, and in this paper we show that a suitable formal model for behavioural descriptions like that in Figure 1 are affine nets with localities (AL-nets). *Affine* nets [13] are an example of Petri net models where arc weights depend linearly on the current marking. They are syntactically related to nets with whole-place operations [1] (WPO-nets) and transfer/reset nets [10], but they have a distinct execution semantics. In this paper, we extend the original model of [13] with step sequence semantics and transition localities. The latter feature supports the definition of the locally maximal execution semantics, allowing one to model GALS (Globally Asynchronous Locally Synchronous) systems [8, 12].

Grouping net transitions in different localities and introducing execution semantics that allows only the maximal multisets of enabled net transitions to 'fire' within a given locality will help us to address the first problem mentioned above. Allowing the weights of connections between places and transitions depend on the current marking will address the second problem.

The synthesis of an AL-net from a transition system specification will be based on the notion of a region of a transition system [11, 3, 2] suitably adapted to AL-nets, and the notion of locally maximal step semantics, a special kind of *step firing policy* (see [7, 16]).

Synthesising systems from behavioural specifications is an attractive way of constructing implementations which are correct-by-design and thus requiring no costly validation efforts. The synthesis problem was solved for many specific classes of nets, e.g., [18, 17, 4, 20, 5, 19]. Later, a general approach was developed within the framework of τ-nets that take a *net-type* as a parameter [3]. In this context, [7] introduced a general approach for dealing with step firing policies, including the locally maximal execution semantics.

In this paper, we focus on the problem of synthesising AL-nets from behavioural specifications provided by step transition systems. A solution to the synthesis problem for the WPO-nets was outlined in [14], and for WPO-nets with localities in [15] and we use some of the ideas introduced there in the proposed treatment of affine nets with localities.

The paper is organised as follows. The next section recalls some basic notions concerning step transition systems and τ-nets. Section 3 introduces AL-nets, and Section 4 presents regions as an essential ingredient for a solution to the synthesis problem for AL-nets, treating them as a special kind of τ-nets.

2 Preliminaries

An *abelian monoid* is a set $\mathbb{S}$ with a commutative and associative binary operation $+$, and an identity element $\mathbf{0}$. The result of composing n copies of $s \in \mathbb{S}$ is denoted by $n \cdot s$, and so $\mathbf{0} = 0 \cdot s$. An example of an abelian monoid is the free abelian monoid $\langle T \rangle$ generated by a set T, the elements of which will represent *steps* of nets with transition set T. $\langle T \rangle$ can be seen as the set of all the multisets over T, e.g., $aab = aba = baa = \{a, a, b\}$. We use $\alpha, \beta, \gamma, \ldots$ to range over the elements of $\langle T \rangle$. For $t \in T$ and $\alpha \in \langle T \rangle$, $\alpha(t)$ denotes the multiplicity of t in α, and so $\alpha = \sum_{t \in T} \alpha(t) \cdot t$. Then $t \in \alpha$ whenever $\alpha(t) > 0$, and $\alpha < \beta$ whenever $\alpha \neq \beta$ and $\alpha(t) \leq \beta(t)$ for all $t \in T$.

Transition systems. A *(deterministic) transition system* $\langle Q, \mathbb{S}, \delta \rangle$ over an abelian monoid $\mathbb{S}$ consists of a set of *states* Q and a partial *transition function* $\delta : Q \times \mathbb{S} \to Q$ such that $\delta(q, \mathbf{0}) = q$ for all $q \in Q$. An *initialised* transition system $\langle Q, \mathbb{S}, \delta, q_0 \rangle$ is a transition system with an *initial* state $q_0 \in Q$ such that each state $q \in Q$ is *reachable* from the initial state, i.e., there are $s_1, \ldots, s_n$ and $q_1, \ldots, q_n = q$ $(n \geq 0)$ with $\delta(q_{i-1}, s_i) = q_i$, for $1 \leq i \leq n$. For every state q, we denote by $enb_{TS}(q)$ the set of all s which are *enabled* at q, i.e., $\delta(q, s)$ is defined. TS is *bounded* if $enb_{TS}(q)$ is finite for every state q of TS. Moreover, such a TS is *finite* if it has finitely many states. In diagrams, $\mathbf{0}$-labelled arcs are omitted.

Initialised transition systems $\mathcal{T}$ over free abelian monoids — called *step transition systems* or *concurrent reachability graphs* — represent behaviours of Petri nets. *Net-types* are non-initialised transition systems τ over abelian monoids and used to define various classes of nets.

Two step transition systems, $\mathcal{T} = \langle Q, \langle T \rangle, \delta, q_0 \rangle$ and $\mathcal{T}' = \langle Q', \langle T \rangle, \delta', q_0' \rangle$, are *isomorphic* if there is a bijection f with $f(q_0) = q_0'$ and

$$\delta(q, \alpha) = q' \Leftrightarrow \delta'(f(q), \alpha) = f(q'), \text{ for all } q, q' \in Q \text{ and } \alpha \in \langle T \rangle \ .$$

Petri nets defined by net-types. A net-type $\tau = \langle \mathcal{Q}, \mathbb{S}, \Delta \rangle$ specifies the values (markings) that can be stored in places ($\mathcal{Q}$), the operations and tests (inscriptions on the arcs) that a net transition may perform on these values ($\mathbb{S}$), and the enabling condition and the newly generated values for steps of transitions (Δ).

A *τ-net* is a tuple $N = \langle P, T, F, M_0 \rangle$, where:

- P and T are respectively disjoint sets of places and transitions;
- $F : P \times T \to \mathbb{S}$ is a *flow mapping*; and
- M_0 is an *initial marking* belonging to the set of *markings*, i.e., mappings from P to $\mathcal{Q}$.

For many classes of Petri nets, including the affine nets, $\mathcal{Q}$ is the set of natural numbers $\mathbb{N} = \{0, 1, 2, \dots\}$ or its subset.

N is *finite* if both P and T are finite. For all $p \in P$ and $\alpha \in \langle T \rangle$, we denote $F(p, \alpha) = \sum_{t \in T} \alpha(t) \cdot F(p, t)$. Then a step $\alpha \in \langle T \rangle$ is *enabled* at a marking M if, for every $p \in P$, $F(p, \alpha) \in enb_\tau(M(p))$. We denote this by $\alpha \in enb_N(M)$. *Firing* such a step produces the marking M', for every $p \in P$ defined by $M'(p) = \Delta(M(p), F(p, \alpha))$. We denote this by $M[\alpha\rangle M'$. The *concurrent reachability graph* $CRG(N)$ of N is formed by firing inductively from M_0 all possible enabled steps, i.e., $CRG(N) = \langle [M_0\rangle, \langle T\rangle, \delta, M_0\rangle$ where

$$[M_0\rangle = \{M_n \mid \exists \alpha_1, \dots, \alpha_n \; \exists M_1, \dots M_{n-1} \; \forall 1 \le i \le n \; : \; M_{i-1}[\alpha_i\rangle M_i\}$$

is the set of *reachable* markings, and $\delta(M, \alpha) = M'$ *iff* $M[\alpha\rangle M'$.

3 Affine Nets with Localities

Assuming an ordering of places, markings can be represented as vectors, with the i-th component of a vector $\mathbf{x}$ being denoted by $\mathbf{x}^{(i)}$. For $\mathbf{x} = (x_1, \dots, x_n)$ and $\mathbf{y} = (y_1, \dots, y_n)$, $(\mathbf{x}, 1) = (x_1, \dots, x_n, 1)$ and $\mathbf{x} \otimes \mathbf{y} = x_1 \cdot y_1 + \dots + x_n \cdot y_n$. Moreover, $\otimes$ also denotes the multiplication of two-dimensional arrays.

Before introducing AL-nets, we first give the definition of affine nets and their step sequence semantics. An *affine net* (A-net), not yet considered as a τ-net, is a tuple $N = \langle P, T, W, \mathbf{m}_0\rangle$, where:

- $P = \{p_1, \dots, p_n\}$ is a finite set of implicitly ordered *places*;
- T is a finite set of *transitions* disjoint with P;
- W is a *weight* function with domain $(P \times T) \cup (T \times P)$ such that, for all $p \in P$ and $t \in T$, $W(p, t) \in \mathbb{N}$ and $W(t, p) \in \mathbb{N}^{n+1}$; and
- $\mathbf{m}_0$ is an *initial marking* belonging to the set $\mathbb{N}^n$ of *markings*.

It is convenient to specify the output weights using linear expressions involving the p_i's. For example, if $n = 3$ then $W(t, p_3) = (2, 0, 1, 4)$ can be written down as $2 \cdot p_1 + p_3 + 4$. In diagrams, arcs are annotated with their weights; arcs with weight 0 are dropped; and annotations '1' are not explicitly shown. A place p_j $(1 \le j \le n)$ is a *whole-place* if $W(t, p)^{(j)} > 0$, for some $p \in P$ and $t \in T$. In such a case we also write $p_j \rightsquigarrow p$.

For $p \in P$ and $\alpha \in \langle T\rangle$, $W(p, \alpha) = \sum_{t \in T} \alpha(t) \cdot W(p, t)$ and $W(\alpha, p) = \sum_{t \in T} \alpha(t) \cdot W(t, p)$. Then α is *enabled* at a marking $\mathbf{m}$ if, for every $p \in P$,

$$\mathbf{m}(p) \ge W(p, \alpha) \; . \tag{1}$$

We denote this by $\alpha \in enb_N(\mathbf{m})$. An enabled α can be *fired* leading to a new marking such that, for every $p \in P$,

$$\mathbf{m}'(p) = \mathbf{m}(p) - W(p,\alpha) + (\mathbf{m} - (W(p_1,\alpha),\ldots,W(p_n,\alpha)),1) \otimes W(\alpha,p)\ . \quad (2)$$

We denote this by $\mathbf{m}[\alpha\rangle\mathbf{m}'$, and define the *concurrent reachability graph* $CRG(N)$ of N as one built by firing inductively from $\mathbf{m}_0$ all possible enabled steps. Note that in (2), the number of tokes deposited in places depends linearly on the marking of the net places *after* the tokens were removed from them by the transitions of the step being executed. In contrast, in WPO-nets [1] the number of deposited tokens is calculated on the basis of the marking before the execution of the step (in addition, the number of tokens removed from the places also depends on the current marking).

An *affine net with localities* (AL-net) is a tuple $N = \langle P, T, W, \mathbf{m}_0, \ell\rangle$ such that $\langle P, T, W, \mathbf{m}_0\rangle$ is the underlying A-net, $\ell : T \to \mathbb{N}$ is the *locality mapping* of N, and $\ell(T)$ are the *localities* of N. In diagrams, nodes representing transitions assigned the same locality are shaded in the same way, as illustrated in Figure 2 for transitions z and u.

AL-nets are executed under the *locally maximal* rule. A step $\alpha \in \langle T\rangle$ is *resource enabled* at a marking $\mathbf{m}$ if, for every $p \in P$, the inequality (1) is satisfied (i.e., if α is enabled in the underlying A-net). A resource enabled step α is then *control enabled* at $\mathbf{m}$ if there are no $t \in T$ and $u \in \alpha$ (not necessarily different from t) such that $\ell(t) = \ell(u)$ and the step $t + \alpha$ is resource enabled at $\mathbf{m}$. A control enabled step α can be then fired leading to the marking $\mathbf{m}'$, for every $p \in P$ given by the formula (2) (i.e., as in the underlying A-net). The *concurrent reachability graph* $CRG_{lmax}(N)$ of N is then formed by firing inductively from $\mathbf{m}_0$ all possible control enabled steps. The concurrent reachability graph of the AL-net in Figure 2 is isomorphic to the step transition system shown in Figure 1.

The concurrent reachability graph of an AL-net can be finite even if the concurrent reachability graph of the underlying A-net is infinite. For example, the underlying A-net of the AL-net in Figure 2 generates infinitely many step sequences $\underbrace{\{t\}\{t\}\{z\}\ldots\{t\}\{t\}\{z\}}_{k \text{ times}}$, each of which leads to a different marking.

In general, execution semantics such as local maximal concurrency can be formulated in terms of *step firing policies* (see [7]). A step firing policy is given by a *control disabled steps* mapping $cds : 2^{\langle T\rangle} \to 2^{\langle T\rangle\setminus\{\mathbf{0}\}}$ that, for a set of resource enabled steps at some reachable marking, returns the set of steps disabled by this policy at that marking. For the locally maximal step firing policy this mapping is given by:

$$cds_{lmax}(X) = \{\alpha \in X \setminus \{\mathbf{0}\} \mid \exists \beta \in X\colon \ell(\beta) = \ell(\alpha)\ \wedge\ \alpha < \beta\}\ .$$

4 Synthesising affine nets with localities

We will now discuss how to construct an AL-net with a concurrent reachability graph that is isomorphic to a given step transition system $\mathcal{T} = \langle Q, \langle T\rangle, \delta, q_0\rangle$.

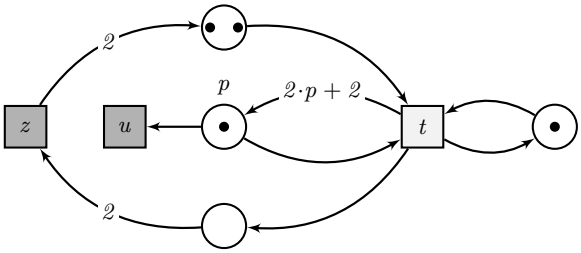


Fig. 2. An AL-net generating the step transition system of Figure 1.

For this net synthesis problem, a general approach was developed for generically defined τ-nets, each such class of nets being represented by its own net-type τ. Note that a key aspect of any solution to a net synthesis problem is to discover all the necessary net places from $\mathcal{T}$ and their connections with transitions of T from τ.

4.1 Net-type for affine nets

AL-nets employ arc weights that depend on the current marking of all places. This may be too general, e.g., in the case of systems where places are distributed among remote neighbourhoods and thus are not capable to exert direct influence on each other. This can be captured by restricting the number of places which can influence arc weights.

A *k-restricted* AL*-net* (k-AL-net, $k \geq 1$) is a AL-net N for which there is a partition $P_1 \uplus \cdots \uplus P_r$ of the set of places such that each P_i comprises at most k places and, for all $p \in P_i$ and $p' \in P_j$ $(i \neq j)$, we have $p \not\rightsquigarrow p'$. That is, there is no exchange of current marking information between different clusters of places P_i.

Although k-AL-nets are not τ-nets in the sense of the original definition, they still broadly speaking adhere to the ideas behind the definition of τ-nets. All we need to do is to define a suitably extended net-type capturing the behaviour of sets of clusters of places rather than the behaviour of single places. More precisely, for each $k \geq 1$, the *k-affine-net-type* is a transition system:

$$\tau^k_{aff} = \langle \mathbb{N}^k, \mathbb{N}^k \times (\mathbb{N}^{k+1})^k, \Delta^k_{aff} \rangle$$

where

$$\Delta^k_{aff} : \mathbb{N}^k \times (\mathbb{N}^k \times (\mathbb{N}^{k+1})^k) \to \mathbb{N}^k$$

is a partial function such that $\Delta^k_{aff}(\mathbf{x}, (X, Y))$ is defined if $\mathbf{x} \geq X$ and, if that is the case,

$$\Delta^k_{aff}(\mathbf{x}, (X, Y)) = (\mathbf{x} - X) + (\mathbf{x} - X, 1) \otimes Y \ .$$

Note that here we treat tuples of vectors in $(\mathbb{N}^{k+1})^k$ as $(k+1) \times k$ arrays.

A *τ^k_{aff}-net* is a tuple $N = \langle \mathcal{P}, T, F, M_0, \ell \rangle$, where:

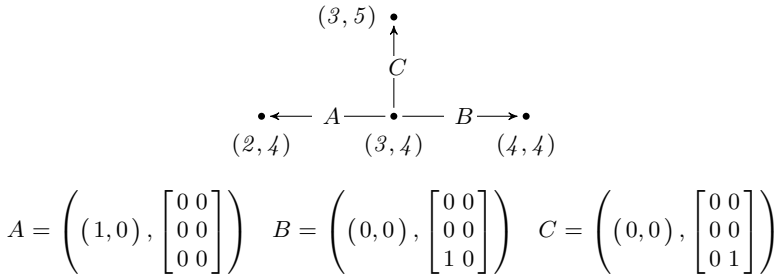


Fig. 3. A fragment of the infinite net-type τ^2_{aff}.

- $\mathcal{P} = \{P_1, \ldots, P_r\}$ is a set of disjoint sets of implicitly ordered places comprising exactly k places each;
- T is a set of transitions being different from the places in the sets of $\mathcal{P}$;
- $F : \mathcal{P} \times T \to \mathbb{N}^k \times (\mathbb{N}^{k+1})^k$ is a *flow mapping*;
- M_0 is an *initial marking* belonging to the set of *markings* defined as mappings from $\mathcal{P}$ to $\mathbb{N}^k$; and
- ℓ is a location mapping for the transitions in T.

For all $P_i \in \mathcal{P}$ and $\alpha \in \langle T \rangle$, we denote $F(P_i, \alpha) = \sum_{t \in T} \alpha(t) \cdot F(P_i, t)$. Then a step $\alpha \in \langle T \rangle$ is *resource enabled* at a marking M if, for every $P_i \in \mathcal{P}$, $F(P_i, \alpha) \in enb_{\tau^k_{aff}}(M(P_i))$. Such a step is then *control enabled* if

$$\alpha \in enb_{N, cds_{lmax}}(M) = enb_N(M) \setminus cds_{lmax}(enb_N(M)) \; . \tag{3}$$

Firing a control enabled step produces the marking M', for every $P_i \in \mathcal{P}$, defined by $M'(P_i) = \Delta^k_{aff}(M(P_i), F(P_i, \alpha))$. We denote this by $M[\alpha\rangle M'$, and then define the *concurrent reachability graph* $CRG_{lmax}(N)$ of N as the step transition system formed by firing inductively from M_0 all possible control enabled steps.

4.2 From transition systems to nets

First we need to express a k-AL-net $N = \langle P, T, W, \mathbf{m}_0, \ell \rangle$, with a set of places $P = \{p_1, \ldots, p_n\}$ and clusters $P_1, \ldots, P_r$, as a τ^k_{aff}-net with localities. Suppose that each set P_i in the partition has exactly k places. (If any of the sets P_i has $m < k$ places, we can always add to it $k - m$ fresh dummy empty places disconnected from the original transitions and places.) We then define $N' = \langle \mathcal{P}, T, F, M_0, \ell \rangle$ so that $\mathcal{P} = \{P_1, \ldots, P_r\}$ and, for all $P_i \in \mathcal{P}$ and $t \in T$:

- $F(P_i, t) = (X, Y)$, where $X = (W(p_1, t), \ldots, W(p_n, t))$ is a vector, and Y is the array $[W(t, p_1), \ldots, W(t, p_n)]$ (the $W(t, p_i)$'s are column vectors), both obtained by deleting the rows and/or columns corresponding to the places in $P \setminus P_i$;

– $M_0(P_i)$ is obtained from $\mathbf{m}_0$ by deleting the entries corresponding to the places in $P \setminus P_i$.

It is straightforward to check that the concurrent reachability graphs of N and N' are isomorphic (when we apply the cds_{lmax} policy, or ignore it, in both nets). Conversely, one can transform any τ^k_{aff}-net with localities into an equivalent k-AL-net and, trivially, each AL-net is a $|P|$-AL-net. Hence k-AL-net synthesis can be reduced to the following two synthesis problems for τ^k_{aff}-net with localities.

Problem 1 (feasibility) *Let $\mathcal{T} = \langle Q, \langle T\rangle, \delta, q_0\rangle$ be a bounded step transition system, k be a positive integer, and ℓ be a locality mapping for T.*
Provide necessary and sufficient conditions for $\mathcal{T}$ to be realised by some τ^k_{aff}-net with the locality mapping ℓ, i.e., $\mathcal{T}$ is isomorphic with the concurrent reachability graph of the net executed under the cds_{lmax} policy defined by ℓ.

Problem 2 (effective construction) *Let $\mathcal{T} = \langle Q, \langle T\rangle, \delta, q_0\rangle$ be a finite step transition system, k be a positive integer, and ℓ be a locality mapping for T.*
Decide whether there is a finite τ^k_{aff}-net with the locality mapping ℓ realising $\mathcal{T}$. Moreover, if the answer is positive construct such a net.

To address Problem 1, we define a *τ^k_{aff}-region* of $\mathcal{T}$ as a pair:

$$\langle \sigma : Q \to \mathbb{N}^k, \eta : T \to \mathbb{N}^k \times (\mathbb{N}^{k+1})^k \rangle \tag{4}$$

such that, for all $q \in Q$ and $\alpha \in enb_{\mathcal{T}}(q)$,

$$\eta(\alpha) \in enb_{\tau^k_{aff}}(\sigma(q)) \;\text{ and }\; \Delta^k_{aff}(\sigma(q), \eta(\alpha)) = \sigma(\delta(q, \alpha)) \;,$$

where $\eta(\alpha) = \sum_{t\in T} \alpha(t) \cdot \eta(t)$. Moreover, for every state q of Q, we denote by $enb_{\mathcal{T},\tau^k_{aff}}(q)$ the set of all steps α such that $\eta(\alpha) \in enb_{\tau^k_{aff}}(\sigma(q))$, for all τ^k_{aff}-regions $\langle \sigma, \eta\rangle$ of $\mathcal{T}$ (intuitively, in this case α is *region* enabled).

In the context of the synthesis problem, a τ^k_{aff}-region represents a cluster of places whose local states (in τ^k_{aff}) are consistent with the global states (in $\mathcal{T}$). Then, to deliver a realisation of $\mathcal{T}$, one needs to find *enough*[3] τ^k_{aff}-regions to construct a τ^k_{aff}-net with localities realising $\mathcal{T}$ (under the cds_{lmax} policy). The need for the existence of such τ^k_{aff}-regions is dictated by the following two *regional axioms*:

Axiom 1 (state separation) *For any pair of states $q \neq r$ of $\mathcal{T}$, there is a τ^k_{aff}-region $\langle \sigma, \eta\rangle$ of $\mathcal{T}$ such that $\sigma(q) \neq \sigma(r)$.*

Axiom 2 (forward closure) *For every state q of $\mathcal{T}$, $enb_{\mathcal{T}}(q) = enb_{\mathcal{T},\tau^k_{aff}}(q) \setminus cds_{lmax}(enb_{\mathcal{T},\tau^k_{aff}}(q))$.*

[3] We need here only a subset of all possible regions, called admissible regions in [9], that act as 'witnesses' for the satisfaction of every instance of the regional axioms.

The above axioms provide a full characterisation of realisable transition systems. The first axiom links the states of $\mathcal{T}$ with markings of the net to be constructed, making sure that a difference between two states of $\mathcal{T}$ is reflected in a different number of tokens held in the two markings of the net representing the said states. The second axiom means that, for every state q and every step α in $\langle T \rangle \setminus enb_{\mathcal{T}}(q)$, we have that:

1. there is a τ^k_{aff}-region $\langle \sigma, \eta \rangle$ of $\mathcal{T}$ such that $\eta(\alpha) \notin enb_{\tau^k_{aff}}(\sigma(q))$ (the step α is not *region enabled*), or
2. $\alpha \in cds_{lmax}(enb_{\mathcal{T},\tau^k_{aff}}(q))$ (the step α is not *control enabled*, meaning that it is rejected by the cds_{lmax} policy).

Note that when a τ^k_{aff}-net with localities realises $\mathcal{T}$, every cluster of places of the net still determines a corresponding τ^k_{aff}-region of the transition system, without taking cds_{lmax} into account.

For Problem 1, by suitably adapting the proofs developed in [15] for the wpo-nets with localities, one can show that $\mathcal{T}$ can be realised by a τ^k_{aff}-net ($k \geq 1$) executed under cds_{lmax} *iff* Axioms 1 and 2 are satisfied.

To address Problem 2 using the feasibility result provided by the above statement we need to find an effective representation of the τ^k_{aff}-regions of $\mathcal{T}$. Similarly as in [14], one can define a system $\mathcal{S}_{\mathcal{T}}$ of equations and inequalities encoding the conditions defining τ^k_{aff}-regions. Then, all the non-negative integer solutions of $\mathcal{S}_{\mathcal{T}}$ are in one-to-one correspondence with the τ^k_{aff}-regions of $\mathcal{T}$. Therefore, Axioms 1 and 2 can be checked using the solutions of $\mathcal{S}_{\mathcal{T}}$.

In general, the (homogenous) system $\mathcal{S}_{\mathcal{T}}$ is quadratic. In practice, one might often want to impose bounds on the allowed range of the whole-place coefficients used in arc annotations. In such a case, Problem 2 has a solution since one can replace $\mathcal{S}_{\mathcal{T}}$ by finitely many linear systems that can be dealt with using the techniques developed for pt-nets that employ the results of [6]. One can also consider modified versions of Problem 2, where there is no need to resort to bounding the whole-place coefficients, and still obtain a solution, see, e.g., [14, 15].

5 Conclusions

In this paper, we extended the notions of τ-nets and τ-regions to the class of affine nets. We also discussed how these two notions can be used to develop a synthesis procedure for affine nets with locally maximal step semantics.

Among possible directions for future work, we single out two challenges. The first is to investigate the relationship between the locality mapping and the grouping of the places into clusters. The second is effective construction without the locality mapping being given as input.

Acknowledgement

We would like to thank Alex Yakovlev for a long standing collaboration on topics related to the modelling, analysis, and synthesis of concurrent systems.

References

1. Abdulla, P.A., Delzanno, G., Van Begin, L.: A Language-Based Comparison of Extensions of Petri Nets with and without Whole-Place Operations. Lecture Notes in Computer Science **5457**, Springer (2009) 71–82
2. Badouel, E., Bernardinello, L., Darondeau, P.: Petri Net Synthesis. Texts in Theoretical Computer Science. An EATCS Series. Springer (2015)
3. Badouel, E., Darondeau, P.: Theory of Regions. Lecture Notes in Computer Science **1491**, Springer (1998) 529–586
4. Bernardinello, L., De Michelis, G., Petruni, K., Vigna, S.: On the Synchronic Structure of Transition Systems. In: Structures in Concurrency Theory, J.Desel (ed.) (1995) 69–84
5. Busi, N., Pinna, G.M.: Synthesis of Nets with Inhibitor Arcs. Lecture Notes in Computer Science **1243**, Springer (1997) 151–165
6. Chernikova, N.: Algorithm for Finding a General Formula for the Non-negative Solutions of a System of Linear Inequalities. USSR Computational Mathematics and Mathematical Physics **5** (1965) 228–233
7. Darondeau, P., Koutny, M., Pietkiewicz-Koutny, M., Yakovlev, A.: Synthesis of Nets with Step Firing Policies. Fundamenta Informaticae 94 (2009) 275–303
8. Dasgupta, S., Yakovlev, A.: Comparative Analysis of GALS Clocking Schemes. IET Computers and Digital Techniques **1** (2007) 59–69
9. Desel, J., Reisig, W.: The Synthesis Problem of Petri Nets. Acta Informatica 33 (1996) 297–315
10. Dufourd, C., Finkel, A., Schnoebelen, P.: Reset Nets Between Decidability and Undecidability. Lecture Notes in Computer Science **1443**, Springer (1998) 63–115
11. Ehrenfeucht, A., Rozenberg, G.: Partial 2-structures; Part I: Basic Notions and the Representation Problem, and Part II: State Spaces of Concurrent Systems. Acta Informatica **27** (1990) 315–368
12. Fernandes, J., Koutny, M., Mikulski, L., Pietkiewicz-Koutny, M., Sokolov, S., Yakovlev, A.: Persistent and Non-violent Steps and the Design of GALS Systems. Fundamenta Informaticae **137** (2015) 143–170
13. Finkel, A., McKenzie, P., Picaronny, C.: A Well-structured Framework for Analysing Petri Net Extensions. Information and Computation **195** (2004) 1–29
14. Kleijn, J., Koutny, M., Pietkiewicz-Koutny, M., Rozenberg, G.: Applying Regions. Theoretical Computer Science (2016)
15. Kleijn, J., Koutny, M., Pietkiewicz-Koutny, M.: Synthesis of Petri Nets with Whole-places and Localities. (submitted)
16. Koutny, M., Pietkiewicz-Koutny, M.: Synthesis of Petri Nets with Localities. Sci. Ann. Comp. Sci. **19** (2009) 1–23
17. Mukund, M.: Petri Nets and Step Transition Systems. International Journal of Foundations of Computer Science **3** (1992) 443–478
18. Nielsen, M., Rozenberg, G., Thiagarajan, P.S.: Elementary Transition Systems. Theoretical Computer Science **96** (1992) 3–33
19. Pietkiewicz-Koutny, M.: Transition Systems of Elementary Net Systems with Inhibitor Arcs. Lecture Notes in Computer Science **1248**, Springer (1997) 310–327
20. Schmitt, V.: Flip-Flop Nets. Lecture Notes in Computer Science **1046**, Springer (1996) 517–528

Asynchronous Design Methods for Dark Silicon Chips

Milos Krstic

IHP, Frankfurt (Oder), Germany

krstic@ihp-microelectronics.com

Dedicated to the dear friend and colleague Alex Yakovlev at the occasion of his 60th birthday.

1 Introduction

The scaling of CMOS technologies continues to enable each two years more and more transistors to the ASIC designers and eventually application users. In theory this would mean that the processing architectures could also exponentially develop in the direction of further many-processor systems scaling. Nevertheless there are significant issues which may disable such scenario with the current (20 nm, 14 nm) and future (10 nm, 8 nm and beyond) technology nodes. The power consumption which can be provided to the silicon unfortunately cannot scale further. This leads to the phenomenon named as dark silicon [1]. As a consequence, even if the increased number of available transistors is used to implement additional processor cores, all available cores cannot be powered at the same time, in order not to overload the thermal budget of the chip. Dark silicon is potentially very significant issue, practically disabling further simple scaling of homogenous processor architectures.

There are several consequences of dark silicon paradigm:

- The business as usual" strategy of scaled complex chips is not possible anymore and novel architectures and strategies are required. The future designs shell instead of parallel operation of many homogenous programmable processing cores, rather use large number of dedicated co-processors which will execute the dedicated tasks in a power-optimal manned only when needed and during idle time stay unpowered [2].
- The advanced power reduction methods are becoming completely unavoidable. The use of power gating, dynamic frequency and voltage scaling and even adaptive voltage scaling are the cornerstones of successful dark silicon ASIC implementation.
- The applied methods/architectures need to be utilized in a dynamic adaptive way, ensuring that the performance is available when applications needs it, but that it can be dramatically reduced in case of inactive operation or it can be utilized with additional robustness again when needed.

– The intelligent power management is needed to utilize the rich processing architectures and advanced power control mechanisms in an optimal and reliable way.

Use of standard synchronous methodologies is quite challenging with dark silicon limitation. Continuous clocking of the circuits adds increase to the power budget and creates the need for extensive clock gating which comes with its own overhead. Moreover the advanced power reduction methods, such as adaptive voltage scaling and use of the circuits in Near-Threshold (NT) modes, becomes to be very critical or suboptimal in synchronous systems due to the large on-chip variability. One important alternative is the partial or intensive use of the asynchronous logic in dark silicon systems.

2 Power Optimal Asynchronous Circuits

Asynchronous circuits are for many years proposed as low-power alternative to the standard synchronous approach. The main difference between the synchronous and asynchronous paradigm is that in the synchronous case there is only one global control signal - clock, which is periodically active regardless whether there is a need for processing in particular pipeline stage or not. In case of asynchronous design, the local activity is there only if there is a valid control initiator (token) which activates the local pipelines. With this methodology the dynamic power consumption could be significantly reduced compared to the synchronous counterparts.

The limitations of asynchronous methods, namely complicated design and purely developed test flow, disabled the pervasive use of the methodology for the applications beyond the academic demonstrators and few industry examples.

In the recent years the novel asynchronous design methods have been proposed, including the concept of desynchronization [3]. Desynchronization enables the seamless conversion of arbitrary synchronous circuit into the bundled data asynchronous design. Using such approaches it has been shown that the significant advantages of the asynchronous circuits in ultra-low power domains can be obtained. In particular, according to [4], in near threshold regimes, due to lack of the global timing and avoidance of worst case paradigm, 40% improved power consumption can be achieved. This can be traded also for more performance under the same power budget. Moreover, the novel aggressive fault tolerant voltage scaling approaches on the asynchronous side, such as recently proposed BLADE [5] show important improvements in comparison with state of the art synchronous power saving architectures such as Bubble-RAZOR.

One of the most interesting concepts for power reduction using asynchronous logic comes from the group of Prof. Alex Yakovlev at the University of Newcastle. He introduces the term "energy modulated computing" [6, 7], indicating the ability of asynchronous logic to use the quant of the energy which is currently available, i.e. to self-adjust the performance to the energy level which is currently available. This concept has been utilized in the various valuable architectures focused on ultra-low power usage in for example wireless sensor nodes [8].

Based on the prior work, it has been shown that the asynchronous circuit design methods are very effective for low-power applications in the scaled CMOS world especially for the applications with extreme power reduction requirements, such as Internet of Things (IoT). Nevertheless, the power requirements are not important only to mobile and IoT applications. The high-performance computing is also affected and dark silicon issues and causes the need to the radical paradigm change. In this context there is a significant chance for the asynchronous logic design methods to address dark silicon problems.

3 Asynchronous Design for Dark Silicon

In order to address the challenges imposed by the dark silicon issues the asynchronous logic design could be effectively utilized in several ways.

The future architectural concept of dedicated, power-optimal and power-controlled co-processor based design is much more suitable for the use of asynchronous logic then the generic homogenous many-core processing which was a main trend some years ago. The asynchronous co-processors could be effectively power-controlled, they do not need the distribution of the clock source, and could be fully event driven. Also utilization of the modern low-power techniques such as adaptive voltage scaling and power gating can be seamlessly integrated in the design of such co-processors. In general the concept of co-processor is based on irregular activation of the dedicated accelerated hardware processing which again in the event driven manner provides back the processed information. In its nature this function is elastic and does not need exact global synchronization and cycle based processing. Consequently, the use of the asynchronous logic seems to be the natural choice. As an example, the recent study [9] has shown that asynchronous co-processor for Elliptic Curve Cryptography (ECC) can reduce the power consumption in comparison to the synchronous counterpart by 1/3.

Moreover, the use of asynchronous logic is in general beneficial for the operations in the ultra-low power regimes of the operations such as Near Threshold Voltage Computing [4]. This can be additionally utilized in co-processor design in dark silicon chips to increase the performance and/or reduce the power consumption. Since synchronous design style still has significant merits when it comes to the high performance, due to the maturity of the design tools and simpler control protocol, it is plausible to propose also the use of mixed-mode synchronous-asynchronous logic, as illustrated in Fig. 1. In the context of dark silicon the specific co-processors can be designed in such way that their pipelines can be with control signal (Mode in Fig. 1) turn into asynchronous (bundled-data) pipelines. This asynchronous mode of operation could be used in near threshold voltage mode to further reduce the power consumption of the system. In high-performance mode, the regular synchronous pipeline could be activated. The overhead of this technique could be limited to the additional control gates which can be in scaled technologies, with billions of gates available, tolerated in many applications.

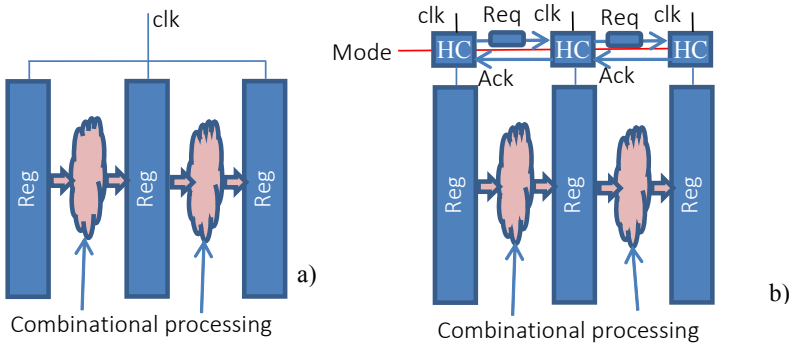


Fig. 1. Standard synchronous pipeline (a), and mixed-mode synchronous-asynchronous pipeline (b)

Finally, the inevitable part of the future dark silicon systems will be interconnects. The concepts of Networks of Chips have be proposed already many years ago, but mainly driven from the academic research. In this context it has been also shown that asynchronous logic can be utilized in much power efficient way compared to the synchronous approach. The power consumption of the asynchronous switches could be reduced by using asynchronous methods from 45% up to 91%, respectively in active and idle operation phases, compared with the clock gated synchronous design [10]. The industry standards (such as AMBA AXI, OCP) and applications push the interconnect development more into point-to-point link direction. However, in this case the use of the asynchronous interconnects is even more natural and simpler than in the case of bus or crossbar interconnect topologies. It is therefore reasonable to expect that in dark silicon chips the role of asynchronous interconnects will be significant. The power management of asynchronous blocks is simpler and more efficient then in case of the synchronous design. In general the free voltage adaptation is always possible and the processing performance will be self-adjusted to the current PVT (process, voltage, temperature) setting, without the need for PLL or similar blocks. Moreover, the blocks may have extremely fast event-based activation which doesn't require active clock being enabled all the time. Finally, there are no additional obstacles in integrating power gating with asynchronous logic.

4 Conclusions

Dark silicon issues impose significant challenges to the design and architectures of the future complex system-on-chip. In this paper it has been emphasized that the asynchronous design methods could be utilized as effective design methods to address the challenges of the dark silicon. Those design methods could be utilized to in general improve the power budget of the system both at the component and at the interconnect level. Moreover, such methods will enable more aggressive voltage scaling techniques and utilization of near-threshold voltages. Finally, the

event based computing is very suitable to the co-processor based architectures which expect to be the main stream solution for the future dark silicon systems.

References

1. H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark Silicon and the End of Multicore Scaling," *SIGARCH Comput. Archit. News*, vol. 39, no. 3, pp. 365–376, Jun. 2011. [Online]. Available: http://doi.acm.org/10.1145/2024723.2000108
2. Q. Zheng, N. Goulding-Hotta, S. Ricketts, S. Swanson, M. B. Taylor, and J. Sampson, "Exploring Energy Scalability in Coprocessor-Dominated Architectures for Dark Silicon," *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 4s, pp. 130:1–130:24, Apr. 2014. [Online]. Available: http://doi.acm.org/10.1145/2584657
3. J. Cortadella, A. Kondratyev, L. Lavagno, and C. P. Sotiriou, "Desynchronization: Synthesis of Asynchronous Circuits From Synchronous Specifications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 1904–1921, Oct 2006.
4. J. Cortadella, L. Lavagno, D. Amiri, J. Casanova, C. Macin, F. Martorell, J. A. Moya, L. Necchi, D. Sokolov, and E. Tuncer, "Narrowing the margins with elastic clocks," in *2010 IEEE International Conference on Integrated Circuit Design and Technology*, June 2010, pp. 146–150.
5. D. Hand, M. T. Moreira, H. H. Huang, D. Chen, F. Butzke, Z. Li, M. Gibiluka, M. Breuer, N. L. V. Calazans, and P. A. Beerel, "Blade – A Timing Violation Resilient Asynchronous Template," in *Asynchronous Circuits and Systems (ASYNC), 2015 21st IEEE International Symposium on*, May 2015, pp. 21–28.
6. A. Yakovlev, "Energy-modulated computing," in *2011 Design, Automation Test in Europe*, March 2011, pp. 1–6.
7. F. Xia, A. Mokhov, Y. Zhou, Y. Chen, I. Mitrani, D. Shang, D. Sokolov, and A. Yakovlev, "Towards power-elastic systems through concurrency management," *IET Computers Digital Techniques*, vol. 6, no. 1, pp. 33–42, January 2012.
8. X. Zhang, D. Shang, F. Xia, and A. Yakovlev, "A Novel Power Delivery Method for Asynchronous Loads in Energy Harvesting Systems," *J. Emerg. Technol. Comput. Syst.*, vol. 7, no. 4, pp. 16:1–16:22, Dec. 2011. [Online]. Available: http://doi.acm.org/10.1145/2043643.2043646
9. S. Zeidler, M. Goderbauer, and M. Krstić, "Design of a Low-Power Asynchronous Elliptic Curve Cryptography Coprocessor," in *Proc. of the IEEE International Conference on Electronics, Circuits, and Systems (ICECS'13), Abu Dhabi, UAE*, Dec. 2013, pp. 569–572.
10. A. Ghiribaldi, D. Bertozzi, and S. M. Nowick, "A transition-signaling bundled data NoC switch architecture for cost-effective GALS multicore systems," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, March 2013, pp. 332–337.

State Recovery of Coarse-Grained TMR Circuits based on Scan Chains and Clock Gating

Jakob Lechner

RUAG Space GmbH, Vienna, Austria
jakob.lechner@ruag.com

Abstract. Triple modular redundancy (TMR) is a wide-spread technique for mitigating soft-errors in digital circuits. Replication of synchronous circuits is often performed at gate-level, where a single clock tree needs to be maintained and a specialised redundant implementation of the circuit is necessary. In case of replication at module-level on the other hand, any non-redundant module implementation can be easily triplicated without modification. The replicas can then be placed at physically isolated locations on the die for optimal fault tolerance. The drawback of this approach is that voting can only be performed at the module outputs and extra effort is required to recover the internal state of a compromised replica. In this paper we therefore present a scan chain-based recovery mechanism, which allows for scrubbing the internal state. The replicated modules do not need to be within a single clock domain, albeit they are likely to operate with the same clock frequency. Clock gating is used to compensate for run-time differences, whenever a state recovery needs to be performed.

1 Introduction

Fault-tolerant circuit architectures are essential for applications that have high reliability requirements and/or are exposed to harsh environmental conditions such as radiation in space. In space electronics, e.g. a shift towards feature sizes of 65 nm and below can be observed due to increasing processing demands. With decreasing feature sizes, however, soft error rates due to radiation are increasing. Furthermore, there is currently a large momentum for small-satellite platforms that rely on inexpensive commercial-of-the-shelf (COTS) parts, which obviously cannot offer the same radiation-hardness as space-grade semiconductor ICs.

All these trends demand for fault-tolerant solutions that can be implemented at architectural level. Triple-modular redundancy is a widely used technique to improve circuit reliability. Replication can be done at gate level, where individual gates and flip-flops are triplicated (fine-grained TMR), or by replicating entire modules, where each module instance is effectively an unchanged copy of the non-redundant circuit implementation (coarse-grained TMR). Both solutions have their upsides and downsides and depending on the specific application needs one might match better than the other. Fine-grained TMR requires EDA tools to perform the desired replication and to insert voters that mask errors after

flip-flops. Due to these voters SEUs in flip-flops cannot spread within the circuit and are recovered in the next clock cycle that overwrites the affected flip-flop. This obviously requires that the entire replicated circuit belongs to a single clock domain with one common clock tree, which remains a single point of failure. The use of voters inside a fine-grained TMR circuit increases the complexity of the interconnect and thereby has direct implications on the physical implementation of the modular redundant circuit. Limited routing capacities and strict timing constraints enforce a compact circuit layout, where replicated components have to be placed in close proximity. This might reduce the reliability of the resulting system, especially if multiple-bit upsets have to be considered.

Coarse-grained or module-level replication on the other hand can be done manually by the RTL designer, simply by instantiating the reliability-critical modules threefold and by implementing voting functions for the module outputs. As the replicated modules are not internally connected via voters, physical separation is possible, minimising the probability of *spatial proximity faults* [1]. Furthermore, the circuit layout of the replicas can remain identical to a non-redundant implementation avoiding any timing penalties that might occur due to the increased routing complexity of a fine-grained solution. The disadvantage of a coarse-grained setup, however, is the fact that SEU in flip-flops are not implicitly scrubbed, since voting is only performed outside the modules. A single upset thus might quickly poison the state of other flip-flops in the affected module replica, whose outputs might then become fully inconsistent to the remaining two healthy replicas. In this situation the failure probability is twice as high compared to a non-redundant system, since a soft error in one of the two healthy modules will cause a system failure. Therefore, a recovery mechanism is required that quickly brings a faulty module replica back to a correct and consistent state. In this paper we want to discuss the concept for such a recovery mechanism.

2 Related Work

In [2] Yu et.al. presented a roll-forward mechanism to improve the reliability of TMR circuits. They propose a state restoration scheme where faulty registers are overwritten with the data values provided by correct registers. Like in our solution recovery is performed at predetermined checkpoints.

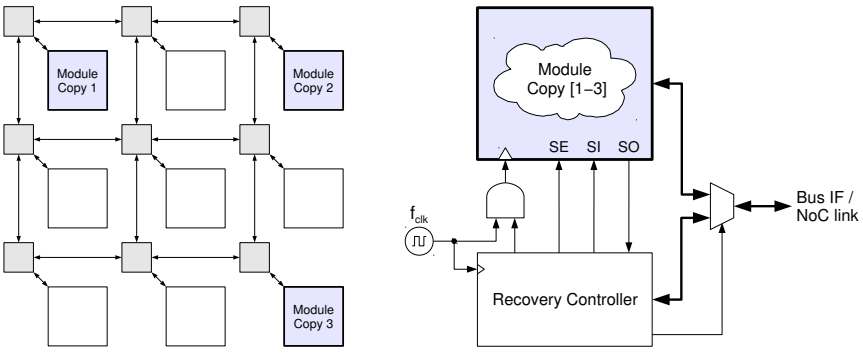
Ebrahimi et.al. [3] employ a voting mechanism to detect faults at the outputs of triplicated components. If an error is detected, the recovery process is triggered. Like in our solution register scan chains are used to recover a faulty state. In [4] the same authors propose an extension of their previous work to recover from multiple transient faults in one or two replicas.

In [5, 6] we have already introduced a scan-chain based recovery mechanism for TMR modules in GALS systems. In such a system the entire module communication is performed over asynchronous interfaces. Synchronization is performed with stoppable clock generators built from on-chip ring oscillators. These clock generators are also used to stall the module operation during the recovery pro-

cess, i.e. they fulfil the same purpose as the clock gating mechanism in this paper. However, the design of robust ring oscillators with low jitter also constitutes a major challenge for practical circuit implementations.

3 Concept

Integrated circuits are typically composed of several modules or IP cores, which are performing various tasks needed for the operation of the chip. They usually communicate over standardised bus systems or a network on chip (NoC). When replicating a critical module there are two alternatives how the module copies can be connected to the bus/NoC: 1) Via a single bus interface or NoC link, where a voter protects the system against faulty accesses to the interconnection network, or 2) via individual, independent interfaces, where each copy can directly provide output data for other components of the system. In first approach failures of a single module copy will remain completely transparent to the rest of the system, whereas in the second solution components that process output data of the triplicated modules need to acquire all the redundant outputs and perform voting themselves. The benefit of the latter approach, however, is that maximum independence of the replicated modules can be achieved. The chip designer can place them anywhere on the die, e.g. in different corners of an NoC mesh. Figure 1 illustrates this idea for an NoC-based chip architecture. Note that the redundant modules do not necessarily need to be in a single clock domain, since they do not exchange data directly. In a mesochronous NoC [7], e.g. each module would likely be operated with the same clock frequency but their clock signals could have an arbitrary (constant) phase offset.



(a) NoC-based chip architecture. (b) Module with recovery controller.

Fig. 1. Coarse-grained TMR concept.

To support the required recovery process we need a mechanism to access state information and exchange it among the replicas. In [5] we therefore proposed to use a scan-chain based approach to tackle this problem. Scan chains provide a

simple mechanism to access the state of a module's flip-flops, both for reading and writing their stored values. Considering that scan chains are available anyway in most ASIC designs for testability reasons, they provide an area-neutral means to perform state recovery. All we need are recovery controllers, one for each replicated module, which perform a readout of their module's scan chain, exchange data with the other replicas and perform majority voting on these data bit by bit and shift the voting results back into the scan chain. Obviously, this process can only be executed when the redundant modules have the exact same state, at least for the flip-flops that store data relevant for the execution after the recovery. Also the recovery controller needs to be in a single clock domain with the module it is associated to.

The data exchange can be done by re-using the module's local link to the system bus or NoC. Depending on the link width a certain number of bits can be read from the scan chain and then transmitted. Concurrently, the recovery controller waits for the same data chunk to arrive from the other replicas. This could take a few clock cycles since the replicated modules are likely to experience a different number of wait cycles when accessing a shared bus or a network-on-chip with other communicating nodes (note that the replicated modules could already be off by some cycles due to different timing of I/O operations during the regular execution). While a recovery controller waits for incoming state recovery data, clock gating can be used to stall the scan-chain operation.

Attention has to be paid to the fact that the recovery controllers themselves might be affected by SEUs. A recovery controller could deadlock or become inconsistent to the other controllers and therefore not join the recovery process at the time it is supposed to. Since the other controllers wait for data (while gating the clock), the healthy module replicas would get stuck. To avoid this a watchdog timer can be implemented in every recovery controller, which detects such a deadlock situation. If a timeout happens in the middle of the recovery process, the healthy controllers can abort the recovery and finish the scan and shift back operation locally without voting. This brings the healthy replicas back into a state where they can go on with their regular operation.

To recover a deadlocked or inconsistent recovery controller – a necessary action to bring the TMR ensemble back to a fully functional state – a mechanism needs to be implemented for the healthy controllers to force their erroneous counterpart into the recovery operation. This can be done with a control signal that is asserted by every (functional) recovery controller at the beginning of the recovery process and is routed to the other redundant controllers. Consider this signal to be a *recovery request*, which is mutually exchanged among the controllers. By voting over local and remote request signals, every controller can determine, if the majority of controllers wants to start the recovery. This information activates another timeout-controlled circuit, which drives the recovery controller's state machine into recovery mode when it fails to do so on its own before the timeout occurs. The replicated recovery controllers therefore form a TMR system themselves. Note that the recovery request signals are mesochronous inputs and have to be synchronized to the clock domain of the receiving controller. Furthermore,

timeout values need to be carefully selected in order for this mechanism to work. For details the interested reader is referred to [6].

4 Conclusion

In this paper we presented the concept of a coarse-grained TMR implementation. Replicated modules provide their outputs to other modules over a bus system or NoC. The replicas do not need to be synchronized to each other and therefore can be implemented independently with no physical constraints on their location on the die. The state recovery with scan-chains does not add extra circuits to the modules and the recovery controllers are simple state machines, which would have a very small area footprint.

Acknowledgements. The ideas for this work stem from the concepts and methods developed for my PhD thesis. A cornerstone for my PhD thesis and the better part of my scientific work so far was a lecture that Alex Yakovlev gave in 2011 at Vienna University of Technology. The lecture obviously was on asynchronous circuits and systems design and helped me a lot to better understand the asynchronous design philosophy. But most of all Alex was able to electrify me with his immense enthusiasm for the field. Thank you so much for your friendship and support, Alex!

References

1. Kopetz, H.: Real-Time Systems: Design Principles for Distributed Embedded Applications. 2st edn. Springer New York Dordrecht Heidelberg London (2011)
2. Yu, S.Y., McCluskey, E.: On-line testing and recovery in tmr systems for real-time applications. In: Test Conference, 2001. Proceedings. International. (2001) 240 –249
3. Ebrahimi, M., Miremadi, S., Asadi, H.: Sctmr: A scan chain-based error recovery technique for tmr systems in safety-critical applications. In: Design, Automation Test in Europe Conference Exhibition (DATE), 2011. (march 2011) 1 –4
4. Ebrahimi, M., Miremadi, S.G., Asadi, H., Fazeli, M.: Low-cost scan-chain-based technique to recover multiple errors in tmr systems. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on **PP**(99) (2012) 1
5. Lechner, J., Veeravalli, V.S.: Modular redundancy in a gals system using asynchronous recovery links. In: Asynchronous Circuits and Systems (ASYNC), 2013 IEEE 19th International Symposium on. (2013) 23–30
6. Lechner, J.: Building Robust GALS Circuits – Fault-Tolerant and Variation-Aware Design Techniques for Reliable Circuit Operation. PhD thesis, Vienna University of Technology (April 2014)
7. Ludovici, D., Strano, A., Gaydadjiev, G.N., Bertozzi, D.: Mesochronous noc technology for power-efficient gals mpsocs. In: Proceedings of the Fifth International Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip. INA-OCMC '11, New York, NY, USA, ACM (2011) 27–30

Where the Light is Coming From

Terrence Mak
terrencemak@gmail.com

Abstract - Newcastle upon Tyne, is a place where my academic career started and is a place that I met my teacher, mentor and friend – Prof. Alex Yakovlev. It was a cold day with plenty of snow covering all the pieces of stones in the college. People who live here are calling themselves Geordies and saying "Aye" instead of "Yes". It is also a place as the guardian for England with the famous Hadrian wall. Strong, ancient but tough castles can be easily found all along the seaside to defend England from intruders. That was the time when I was almost finished my PhD at Imperial College London. When all my colleagues started to look for jobs in London or looking for academic positions back to their home countries, Prof Alex Yakovlev offered me a lectureship at Newcastle University and that was the beginning of my career as a Lecturer. "Terrence, are you interested to join us?" Alex asked with an enthusiastic smile.

1 Alex as a Teacher

"Terrence, this is Ra'ed, one of my PhD student. I let you two to have chat," This is the first week when I joint Newcastle. I was struggling where to look for PhD students and where to looking for research assistants. Ra'ed was a guy with Moustache and a bit like Zappa. Not too sure about his age but plenty of smile and can speak fluent English. He began by explaining where he was coming from and what was his research.

It was not easy at all to initiate and maintain a good motivation throughout the PhD, and it was even more difficult to direct the student into a research area that was rough and unclear. However, there was a strange connection between myself, Alex and Ra'ed. The efficiency between myself and Ra'ed was like the speed of light and we began to work out what problems to tackle, to use what kind of tools and techniques. Ra'ed was quickly identified the research topics. One special highlight was that both the descriptions and illustration of research, and drawing and sketching of figures from Ra'ed were totally professional, given the fact that he was at his first year PhD. Especially, he could draw a figure using five to six different colours, in order to make sure everybody knew what he was talking about.

Only within six months, we finished a conference paper submission. Sooner, the great news came back and his paper had been accepted at the DATE'11 (one of the most prestigious conference in the world) and, later on, the paper was awarded the Best Paper Award 2011 in the conference. Need to spell out that this conference had at least 800 submissions and only one paper could obtain this award. Not only a de-

lightful smile was given by Ra'ed, I began to enjoy working with the inspirational teacher, Alex.

Fig. 1. (Left) Both Alex and myself were receiving the Best paper award at DATE'2012. (Right) Ammar, Ra'ed, Alex and myself are visiting Vancouver to attend the networks-on-chip architecture conference in 2013. Yet, we were visiting the famous forrest in Vancouver.

Later on, we had received multiple awards including one highlight, which was called the 2015 IET Computer & Digital Techniques Premium Award. This is a really outstanding award and only one journal paper is selected each year in this top-in-the-field magazine. This prize was based on an IET publication together with our co-supervise student, Dr Nizar Dahir, who was a PhD student and was also co-supervised with with Alex. Together with Alex, we have more than eight PhD students graduated and they are all now working at different exciting places in the world.

2 Alex as a learner

"I am learning," said by Alex. At a quarter past three, if you go to the tea room at level 3 of the Electrical and Electronic Engineering Building, you will find Alex who is there and talking to students, colleagues and may be the waitresses. The tea room in the building is the discussion hotspot. Especially in the afternoon, no one would miss the time to go there and have a chat. Alex is one of the usual customer. "If we can provide an architecture as 3-dimensional structure and spinning like a spiral, the energy saving will be tremendous, yet …", a group of students were holding their drinks and listening to Alex. In fact, that tea room at level 3 is a scientific and engineering meeting place.

Fig. 2. "Shall we go for a tea?" Alex asked. "Of course, it is time to discuss." I replied.

We discussed a lot and, most of the time, the discussion was not only limited by our research. Diversity from different area and discussion on different religions were in our discussions. Drinking an afternoon tea at level 3, you wouldn't miss the political or even spiritual view from Alex. Commentary on various political groups and leaders in the history were even more fascinating. You could certainly obtain a feeling based on the thought and perspective from Alex. But the most appreciated character that naturally speared with Alex is his inclusiveness. His students are with different ages, from different countries, having different religions and especially with various characters.

Alex said, "My students are my teachers and I can learn a lot from them". Alex is truly a model of researcher, a teacher and a sincere friend.

3 The most important is Happiness

"The most important thing is happiness. Happiness is the bridge between you and your friends, is the mirror of your success and is synchronous to your awards and asynchronous to your learning and experiences", Alex said in his 60's birthday party. We all congratulated Alex's birthday and make unstoppable clapping that evening. Happiness might be something that you are looking for. But to Alex, happiness was naturally rooted in his blood and illuminating in his face. It is not only the emotion, but a strong character or even a naturally instinct to interact with people. This is the most respectful character to learn from experiences. Because, happiness is the reward to someone who never stop learning. This is Alex is my mentor, my teacher and is my very good friend as well.

Fig. 3. With the same hat and jacket, I can sense the power of happiness from Alex.

Reassessing the causes of asynchronous systems performance

Alain J. Martin
California Institute of Technology

Abstract. In the course of the past 25 years of research in asynchronous VLSI, several complex systems, mainly microprocessors, have been designed and successfully fabricated to demonstrate the proposed technology.

Most prototypes have displayed a set of unique advantages. Some advantages, such as improved robustness to parameter variations, can be directly attributed to the quasi delay-insensitivity of circuits without clocks. But the remarkable performance of some large asynchronous chips in the combined space of speed and power has been offset by the mediocre behavior of some other comparable asynchronous prototypes. This suggests that some other factor may be at play.

The question raised here is the following. What if the good performance of complex asynchronous systems were not due to the absence of a clock?

Having been for many years an ardent proponent of asynchronous logic, I will happily slip into the role of the iconoclast and propose that the excellent performance of several asynchronous systems may not be (mainly) due to asynchrony (the absence of clock).

My argument concerns the design of at least moderately complex systems, typically a microprocessor. (I am not talking about single components such as a FIFO or an adder.) A sufficient number of such complete asynchronous systems have now been produced, and we can draw some conclusions. I will use as examples the projects I know best: the asynchronous systems developed at Caltech between 1988 and today. Other labs have produced asynchronous prototypes following an essentially similar approach: the Cornell asynchronous microcontrollers, the Grenoble ASPRO series, and the Fulcrum switches are the closest in design style.

What characterizes those approaches is a systematic top-down synthesis starting from a simple and, in principle, easily verifiable version of the device. I will call this approach “ab initio”, because no attempt should be made at the start to guess the final structure of the design. Rather, starting from this simple initial version, the final structure is reached through a sequence of semantic-preserving transformations. The goal of the transformations is usually to increase concurrency (pipeline depth, for example) and at the same time to replace large monolithic components with a number of small-

er and smaller ones. In the end, each component has a standard, easily implementable structure. This first set of transformations is not part of the compiling phase. The source and result of a transformation are in the same formalism. At Caltech, this formalism is the programming language CHP. It is only at the end of this first phase, when all components are of the appropriate size and the decomposition is expected to deliver the target performance, that the second phase, which can be called compilation, transforms each CHP component into an asynchronous circuit.

The quality of the formalism (CHP, for example) is crucial to the method, as it must allow an algebraic style of manipulations of the representation of the system, and it requires us to look at a programming language not so much as a "description language," but rather as a formal notation.

Other formalisms, such as Petri Nets or STG ("signal transition graphs") have been used with success as well. Their graph-base model gives them a flexibility that language-based models sometimes miss. My personal reticence with such tools is that they do not scale well with the size of the systems to be represented. Who can give me the Petri Net or STG describing a complete microprocessor?

All systems engineered following this general approach have produced excellent results in terms of energy efficiency, robustness to parameter variations, combined energy and delay performance. All were functioning correctly on first silicon, often over a wide range of power supplies, including sub-threshold. Other groups using a similar approach have reported similarly good results.

But several asynchronous systems developments have followed a different approach. Whereas the final implementation of the components as asynchronous circuits followed a method essentially similar to the first one, the overall system decomposition was quite different. The starting point was usually a pre-existing synchronous architecture – or at least a standard, generic, synchronous pipeline. Without mentioning specific cases, the general observation is that asynchronous systems developed along this path often produced unremarkable results, not better and sometimes worse than their synchronous counterpart – except where it concerns attributes specific to asynchrony, such as robustness to parameter variations.

The discrepancy between the two methods has been a surprise to me, as it should be to all those interested in the technology. Once, a proponent of the second approach was so surprised by the results claimed for the Caltech MiniMIPS (an asynchronous version of a MIPS R3000 microprocessor) that he suggested at an NSF meeting that the performance figures we reported could absolutely not be true, and must be fraudulent…

So what may be causing the performance gap between the two approaches? Both use asynchronous logic as circuit implementation, albeit with some significant differences: the circuits of the ab-initio approach are usually of the QDI flavor, whereas the

circuits of the second approach are usually not. But that difference is not significant enough to explain the discrepancy. So if the excellent performances of the ab-initio approach are not due to asynchrony, they may be the result of the drastically different system-level design.

At the system level, the main design issue is complexity – and, in particular, in the case of a modern microprocessor, complexity due to intricate concurrency and communication, which are crucial to achieving any kind of performance. In a traditional approach, the engineer confronted with this complexity has only one safe way out: to stick as closely as is possible to what he or she already knows might work. In the absence of a correct-by-construction method, drastically new explorations of alternative solutions are too risky and too time consuming.

In the ab-initio approach, many (possibly all) alternative solutions can be generated with the emboldening knowledge that they are all correct. A thorough exploration of the solution space is achievable by the algebraic manipulation of an abstract object (a CHP program) that is a correct representation of the complete design. Certainly, asynchrony greatly facilitates the application of this synthesis method by postponing dealing with the physical issues as long as possible.

Another argument is that, by positioning the high-level design squarely within the realm of distributed computing, all the tools (program constructs, proof methods) and techniques of concurrency can be brought to bear on the problem with much better results than those achieved by traditional HDLs.

In the end, it is conceivable that asynchrony is mainly a powerful enabler for a concurrency-based approach to digital system design – and that this approach itself represents a fundamental paradigm shift from traditional methods.

Asynchronous BDD-like structures

Oleg Mayevsky[1], Andrey Mokhov[2], Danil Sokolov[2]

[1] oleg-maevsky@yandex.ru
[2] Newcastle University, United Kingdom
andrey.mokhov@ncl.ac.uk, danil.sokolov@ncl.ac.uk

Abstract. Another attempt to apply BDD-like structures to build digital systems. The close relationship of BDDs with elementary schemes of computational algorithms is used. A simple conceptual option for decomposition of asynchronous computational processes on sub-machines with BDD-like structure is seen in several examples.

Keywords: binary decision diagrams, asynchronous circuits

1 Introduction

An important property of Binary Decision Diagrams (BDD) is their ability to canonically and compactly represent Boolean functions, which is attractive in digital circuit design, and in some cases makes BDDs preferable to other representations [1]. A lot of research has been dedicated to the use of BDDs for synthesis and analysis of circuits in different technologies [2–4]. BDDs were typically used for implementing the combinational part of control automata in an 'orthogonal way' that came both with benefits, such as testability and race freedom, as well as drawbacks, such as the requirement for separate descriptions of the control and operational parts of the system, and, perhaps more importantly, the sequentiality of described processes. The sequentiality was caused by the fact that a single BDD could only represent a single Boolean function, which forced the designer to decompose the combinational part of control into a set of independently synthesised components that could not be used simultaneously.

In this work we use BDD-like diagrams for the compact representation of concurrent computations. Thanks to the orthogonality of obtained descriptions, it is possible to directly map them into efficient asynchronous controllers of large computation systems. We show that the size of such descriptions significantly depends on the decomposition between the control and operational parts of the system.

Section 2 demonstrates a correspondence between BDDs and simple logic computations on an example. To emphasise the correspondence we show how BDDs can be converted to *Orthogonal Canonical Parameterised Computation Graphs (OCPCGs)* that inherit orthogonality and canonicity – the two key properties of BDDs. We then discuss OCPCG-based control descriptions and show how they can be translated to Petri Nets. The rest of the section is dedicated to graph decomposition, concurrency and synchronisation issues.

Section 3 discusses ways of reducing the size of OCPCG descriptions by restricting the labellings of their elements. We use several examples to show how the size and properties of OCPCGs describing the control part of a system depend on the properties of its operational part, and study restrictions on OCPCG labellings that preserve the canonicity.

Section 4 concludes the paper with an example of using OCPCGs in the implementation of the control part of a 5-bit multiplier. It is known that the corresponding Boolean functions cannot be compactly represented by BDDs, however we show that OCPCGs admit a simple reformulation of this problem leading to significant savings in terms of the size of the obtained representations.

2 BDDs and logic computations

Consider a BDD in Fig. 1 describing a Boolean function $y = F(x_1, x_2, x_3, x_4, x_5)$ whose truth table is shown in Tab. 1. The *terminal* nodes 0 and 1 at the bottom of the BDD correspond to the value that the function has on a particular set of input variables. This diagram can also be thought of as a control part of a system, whose task is to execute a particular operation under the condition $y = 1$ and skip it when $y = 0$.

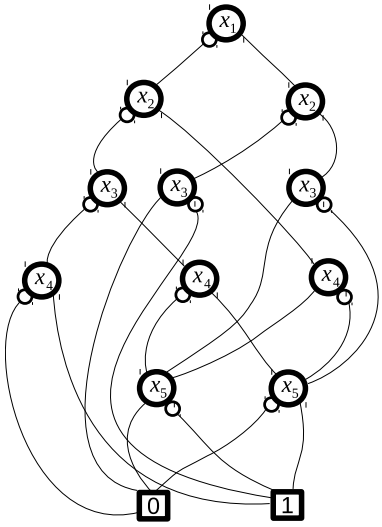


Fig. 1: BDD corresponding to the function from Tab. 1.

Let us modify the diagram in Fig. 1 by labelling the arcs leading to the two terminals by 0 and 1 according to their targets (0 labels may also be omitted on diagrams, as one can always infer them from remaining 1 labels). We then

x_1	x_2	x_3	x_4	x_5	y
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	1
0	0	0	1	1	1
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	1

x_1	x_2	x_3	x_4	x_5	y
0	1	0	0	0	0
0	1	0	0	1	1
0	1	0	1	0	1
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	0

x_1	x_2	x_3	x_4	x_5	y
1	0	0	0	0	1
1	0	0	0	1	1
1	0	0	1	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	0	1	0
1	0	1	1	0	0
1	0	1	1	1	0

x_1	x_2	x_3	x_4	x_5	y
1	1	0	0	0	0
1	1	0	0	1	1
1	1	0	1	0	0
1	1	0	1	1	1
1	1	1	0	0	1
1	1	1	0	1	0
1	1	1	1	0	1
1	1	1	1	1	0

Table 1: Truth table of a Boolean function of 5 variables.

merge the terminals into a new node end, and add another node begin at the top of the diagram for symmetry. See the result in Fig. 2.

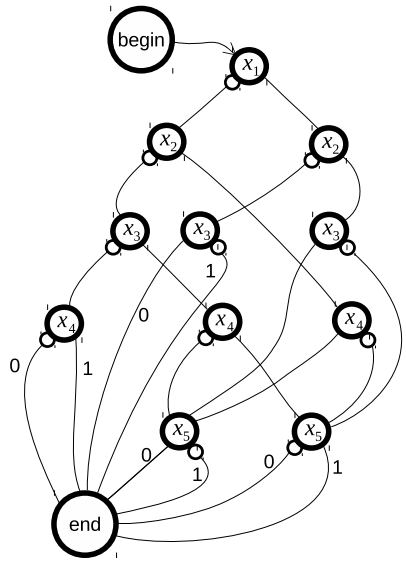


Fig. 2: A computation graph derived from the BDD in Fig. 1.

The newly added nodes begin and end have a different semantics from that of terminals 0 and 1 in BDDs: instead of denoting the final computed value, they simply indicate the start and end of a computation process. Note that the resulting computation graph in Fig. 2 may be *local*, that is, it may be a (sequential) part of a larger computation system, which may contain concurrency and/or cycles, as illustrated in Fig. 3.

All nodes of a computation graph can take part in the computation, similar to Petri Nets. As an example, we can place a *token* in the begin node, that

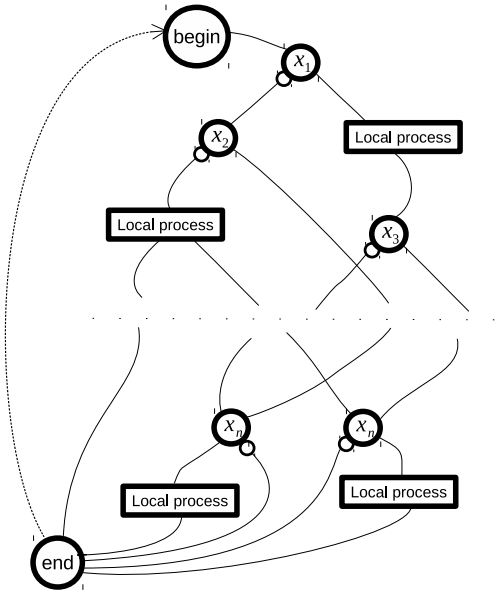


Fig. 3: A cyclic computation graph comprised of local processes.

will then travel along the arcs of the graph following the local *routing* decisions made in individual nodes according to the values of input variables. Each arc could in fact correspond to a Petri Net that consumes a single token from a node, performs a computation process according to usual Petri Net rules, and then signals about the completion by releasing the token to the next node. In fact, the whole computation graph can be modelled by a Petri Net if we create suitable Petri Net fragments corresponding to individual computation nodes, e.g. as shown in Fig. 4.

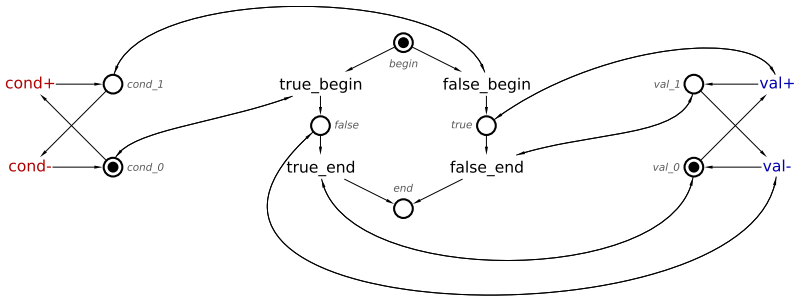


Fig. 4: An example fragment Petri Net.

3 Labellings

Let us come back to the graph in Fig. 2. The upper half of the graph is dedicated to the computation of the necessary logic conditions that are used by the lower half of the graph for actually executing the action associated with $y = 1$. The complexity of the computation part eventually determines the complexity of the hardware that implements it. It is well-known that BDDs often lead to overly complex circuits when used directly to implement logic computations, compared to conventional logic synthesis. However, one can consider alternative ways of labelling the elements of the computation graph in Fig. 2, which can lead to a reduction in the number of its vertices and therefore in a lower complexity of the resulting hardware.

As an example, let us change the semantics of the labels in our computation graph. The label of 1 will now correspond to adding 1 to the current value of y modulo 2, that is, $y \leftarrow y \oplus 1$. The label of 0 will correspond to $y \leftarrow y \oplus 0$, which is essentially a *no-op*. With this approach, one can imagine the token to start in the node begin with the initial value $y = 0$, and travel along the arcs of the graph, undergoing the transformations corresponding to the labels on route to the destination node end. When the token reaches end, the value of y becomes the final outcome of the computation. There may be several equivalent labellings that compute the same Boolean function. Fig. 5 gives an example of a labelling that is equivalent to the one shown in Fig. 2.

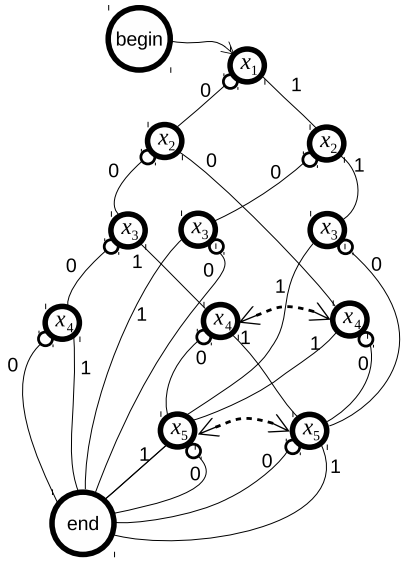


Fig. 5: An equivalent labelling of the graph from Fig. 2.

The existence of equivalent labellings means that our new computation graphs lost an important property inherited from BDDs – the canonicity of the representation of Boolean functions. Fortunately, it is possible to recover it.

Indeed, there are certain rules we can follow to relabel a computation graph in order to bring it back to a canonical form. For example, consider an arbitrary node x_n with arcs labelled a, b, c and d, as shown in Fig. 6(left).

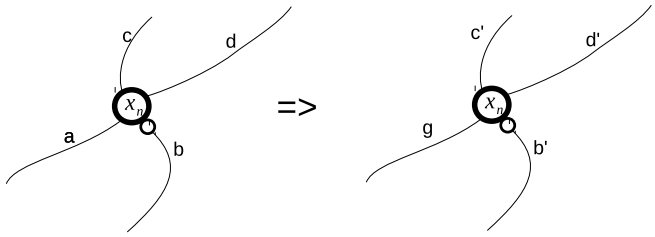


Fig. 6: Equivalent relabelling.

Let us try to change the label a to g. Labels b, c and d may need to be changed too in order to preserve the equivalence. To find out admissible new labels b', c' and d' we can solve the following system of equations:

$$\begin{cases} g \oplus c' = a \oplus c, \\ g \oplus d' = a \oplus d, \\ b' \oplus c' = b \oplus c, \\ b' \oplus d' = b \oplus d. \end{cases} \tag{1}$$

The equations above use modulo 2 addition $\oplus$, but in general any additive group can be used. If the set of labels does not form an additive group with respect to the chosen addition operator, then the resulting system of equations may have no solutions or multiple solutions, therefore limiting the freedom of graph relabelling and/or making the derivation of a canonical labelling more challenging. On the other hand, if the system of equations is guaranteed to always have a unique solution, as in the case of (1), then one can fix the label of a particular arc, e.g. fix the $x_n = 0$ branch to always have label 0, leading to a canonical labelling of the computation graph.

Once a canonical relabelling is performed, one can reduce the resulting graph by merging equivalent nodes, similar to the reduction of BDDs. Fig. 5 shows two pairs of equivalent nodes by dashed arrows; after merging them we obtain the reduced computation graph shown in Fig. 7.

The system of equations of the form (1) will be analogous for any additive group, including non-commutative ones, such as real or complex numbers, vectors, matrices, etc. The issue of commutativity in this context is related to concurrency. Indeed, if the labels do not commute then the computation order is important and the graph corresponds to a sequential computation process. On

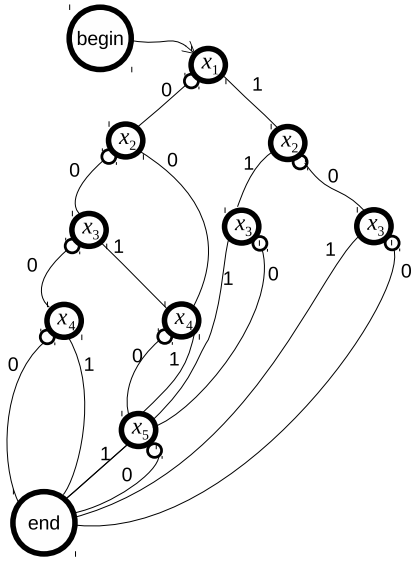


Fig. 7: Reduced computation graph from Fig. 5.

the other hand, if labels commute, e.g. $a + b = b + a$, then the order in which a and b are performed does not matter and they may be performed concurrently. Most real systems are mixed and involve both sequential and concurrent computations, which can be represented by general models such as Petri Nets. The following example illustrates how commutative labels can also be used for finding further reduction opportunities in computation graphs.

Let us remove the node conditions x_n from the graph in Fig. 7 and place them on the incoming arcs instead, as shown in Fig. 8. In other words, nodes no longer contain any conditions, and the latter actually become arc labels, which means variable comparisons are now performed while travelling along arcs. We therefore have a new set of arc labels $\{x_1, x_2, ...\}$ that do not interact in any way with existing labels $\{0, 1\}$, and therefore commute with them. However, we can no longer relabel arcs as described above, because we have not yet defined the group operation that acts on the new labels. Despite this, we can already identify new equivalent nodes that can be merged, as indicated by dashed arrows. The graph obtained by merging these nodes is shown in Fig. 9.

Let us refer to the new labels $\{x_1, x_2, ...\}$ as *condition labels*. Can we move these labels from one arc to another? Yes, we can! Below we give one possible relabelling method, which is not as simple as we would like, but does provide further intuition into labelled computation graphs.

Let us keep all condition labels in a queue, which is initially empty, and supports two operations: i) extracting a label at the front of the queue, ii) adding a new label to the queue, either combining it an with existing label corresponding to the same control variable (using a group operation defined below), or adding

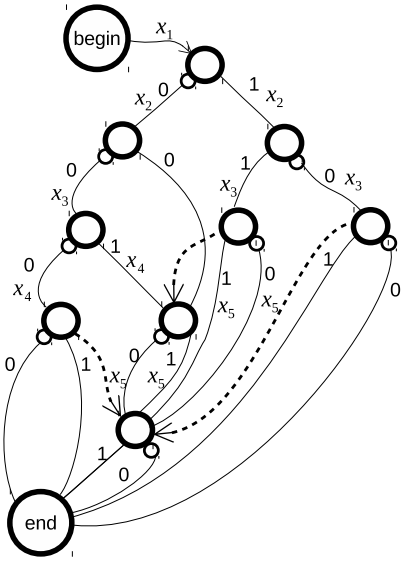


Fig. 8: Moving conditions from nodes to arcs.

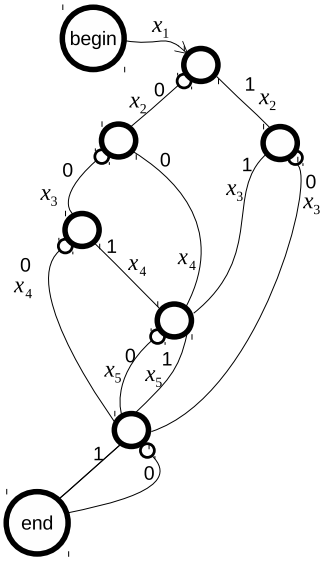


Fig. 9: Reducing the graph in Fig. 8.

it to the back of the queue if it is the first occurrence of the variable in the queue. Note that labels corresponding to different variables do not interact with each

other, therefore it is sufficient to define an operation only on labels corresponding the the same variable.

Let x stand for a control variable. Then we can define a group operation $+$ for combining labels corresponding to this variable according to Tab. 2. This is an infinite commutative group.

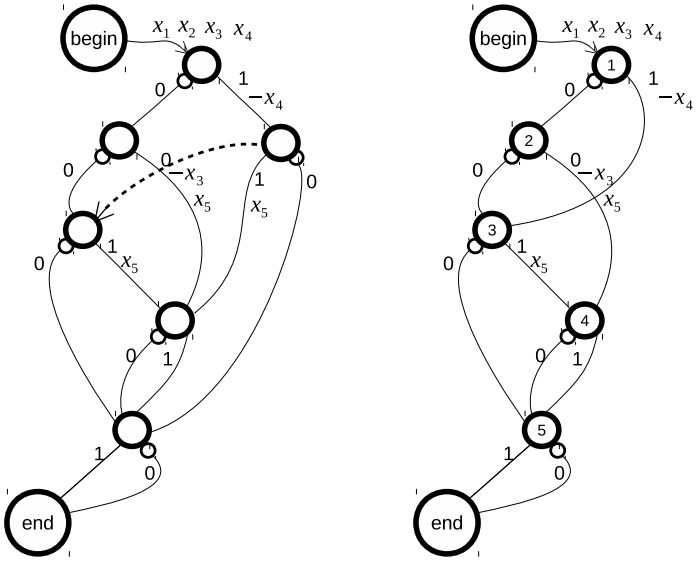
$+$	0	x	$-x$	$2 \cdot x$	$-2 \cdot x$	$\ldots$
0	0	x	$-x$	$2 \cdot x$	$-2 \cdot x$	$\ldots$
x	x	$2 \cdot x$	0	$3 \cdot x$	$-x$	$\ldots$
$-x$	$-x$	0	$-2 \cdot x$	x	$-3 \cdot x$	$\ldots$
$2 \cdot x$	$2 \cdot x$	$3 \cdot x$	x	$4 \cdot x$	0	$\ldots$
$-2 \cdot x$	$-2 \cdot x$	$-x$	$-3 \cdot x$	0	$-4 \cdot x$	$\ldots$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$

Table 2: Combining condition labels corresponding to variable x

Label x in this group means: *apply x as the branching condition at the current node*. Label $2 \cdot x$ (here the dot is not a group operation, it simply denotes the multiplicity of the label) means: *apply x as the branching condition at the current node as well as the next node*. Label $-x$ means: *skip x and use the next label in the queue as the branching condition*. This is a complex rule and will likely incur significant hardware cost in the implementation. Furthermore, to further reduce the size of the computation graph it may be necessary to allow multiple labels on each arc in the graph, which is also costly. Nevertheless, this approach may be practically beneficial for certain applications, where the aim is to minimise the number of branches. The graph obtained by labelling arcs using the group operations in Tab. 2 and its reduced variant are shown in Fig. 10.

To obtain the graph shown in Fig. 10(left) we use the following relabelling approach. Our end goal is to have 0 labels on all false arcs of the graph. To achieve that we first move variable x_1 to the topmost arc. We then remove label x_2 from the subsequent false arc by subtracting x_2 from both outgoing branches, and adding x_2 to the incoming arcs. This procedure is then repeated for all occurrences of x_3, until all false arcs are free from it, and so forth.

Let us now demonstrate how the resulting graph can be used for computation. Consider variable assignment 11010 (x_1 corresponds to the leftmost bit). Starting at the arc leaving node begin, we extract the first element from the queue and since $x_1 = 1$ we follow the true arc of the subsequent node of the graph (see Fig. 10b). The arc holds two labels: the first one tells us that we need to update the value of y by adding 1 to it: $y \leftarrow y \oplus 1 = 0 \oplus 1 = 1$, the second one tells us that condition variable x_4 will have to be skipped, therefore we erase it from the queue and proceed. These two markers do not interact and hence commute. Once both of them are handled, we arrive at the next node (denoted as 3 in Fig. 10b), and the next variable in the queue that we check is x_2. Since $x_2 = 1$ we travel along the arc leading to node 4, updating $y \leftarrow y \oplus 1 = 1 \oplus 1 = 0$



(a) Initial transformation. (b) Reduced graph.

Fig. 10: Graph Fig. 9 transformed using condition labels.

again and adding x_5 to the back of the control label queue. The next variable to check is $x_3 = 0$, hence we follow the false arc, which contains label 0 only (as a consequence of our relabelling algorithm). We do nothing and now check $x_5 = 0$ (remember, we erased x_4 from the queue), which leads us directly to node end, where we report the computation outcome: $y = 0$ (upon checking Tab. 1 we see that this is the correct value).

3.1 Labellings with two operations

The use of independent (commutative) labels allows us to model a system of Boolean functions with a single computation graph. However, the more independent labels we can have on an arc and the more different values a single label can take, the fewer opportunities for graph reduction will be available, because the number of 'unmergeable' nodes grows. As we have seen in the previous subsection, by introducing a richer algebraic structure to labels sometimes allows us to find new opportunities for graph reductions without sacrificing the canonicity of the representation. We can take this idea further, and consider labels that form richer algebraic structures with two operations, that still guarantee unique solutions to the system of equations arising in the process of graph relabelling. For finite sets of labels such algebras are *Galois fields*; for infinite sets – *division rings*.

An algebraic structure with two operations is a powerful tool for graph relabelling. We can associate pairs of labels with an arc of a computation graph: one

for the additive component, and another for the multiplicative one. The additive component is added to the labels reachable during the computation, while the multiplicative one is multiplied by them.

The relabelling process is arranged in two stages. In the first stage we associate additive labels with all arcs making sure that false arcs contain 0 values (as before). In the second stage, we add multiplicative labels to all arcs except for those pointing to node `end`. We then normalise the pairs so that false arcs contain exactly the pair $(0, 1)$, the additive zero and the multiplicative identity.

Let us clarify the above using an example of a system of four Boolean functions of four input variables $x_1..x_4$ describing a 2-bit binary multiplier. Tab. 3 shows the truth table for all four functions $y_1..y_4$ of the multiplier.

x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Table 3: Truth table of a 2-bit binary multiplier

Let us use 4-bit labels with addition $+$ being component-wise addition modulo 2, and multiplication $*$ defined modulo an irreducible polynomial $e^4 + e + I$ where $e = 0010$ is the generator. Below are all $2^4 = 16$ labels of the resulting algebraic structure:

- $e = 0010$ – the generator,
- $e^2 = 0100$,
- $e^3 = 1000$,
- $e^4 = e + I = 0011$,
- $e^5 = e^2 + e = 0110$,
- $e^6 = e^3 + e^2 = 1100$,
- $e^7 = e^3 + e + I = 1011$,
- $e^8 = e^2 + I = 0101$,

- $e^9 = e^3 + e = 1010$,
- $e^{10} = e^2 + e + I = 0111$,
- $e^{11} = e^3 + e^2 + e = 1110$,
- $e^{12} = e^3 + e^2 + e + I = 1111$,
- $e^{13} = e^3 + e^2 + I = 1101$,
- $e^{14} = e^3 + I = 1001$,
- $e^{15} = I = 0001$ – the identity element,
- $\emptyset = 0000$ – the zero element.

As the first step we build a computation graph according to the specification given in Tab. 3 using only additive labelling, that is using only the modulo 2 addition operation for combining labels along computation paths. The resulting graph is shown in Fig. 11.

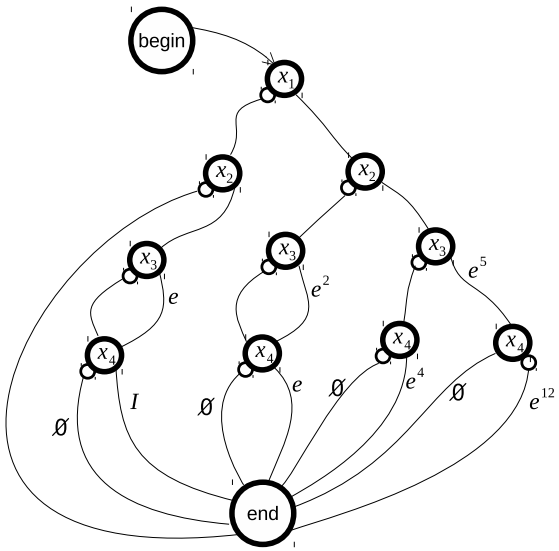


Fig. 11: Additive labelling.

There are no pairs of equivalent nodes, hence the obtained graph cannot be reduced. However, we can now relabel the graph using the multiplication operation, by factoring out common factors from outgoing arcs and moving them up to the incoming arcs in a canonical manner. After applying the relabelling to the lower layer of the graph we obtain a new graph shown in Fig. 12. We explain the used notation for new labels below.

We use so-called *Polish prefix notation* for compact representation of the effect (function) that additive and multiplicative labels have on values travelling along arcs. For example, label $+\emptyset * I$ should be interpreted as $y \leftarrow \emptyset + I * y$, where y represents the current value of the computation, just as in the examples

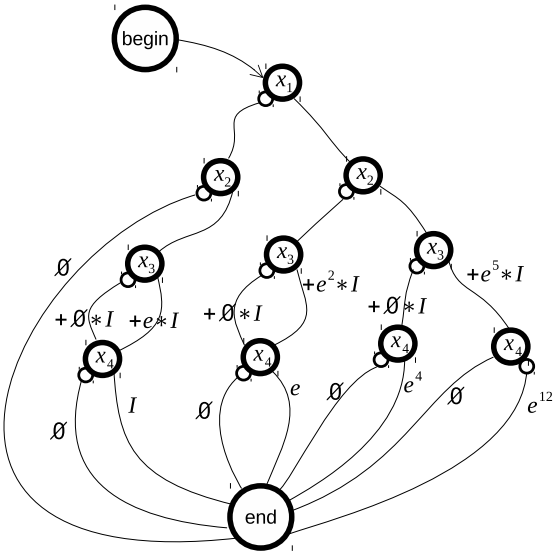


Fig. 12: Multiplicative labelling of the lower layer of the graph in Fig. 11.

before. The use of Polish notation allows us to avoid parentheses on the arcs of the graph. Alternatively, one can use *lambda calculus* for a more general and familiar (yet somewhat more verbose) representation.

Let us now define the relabelling rule using the properties of addition and multiplication of our Galois field. As before, consider a node x_n with incoming and outgoing arcs as shown in Fig. 13. If we would like to relabel the graph by changing a to g, how do we need to change the other labels?

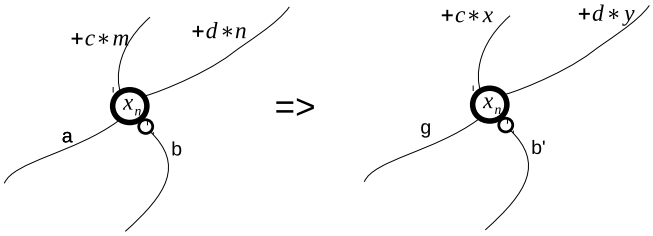


Fig. 13: Relabelling with two operations.

The following four equations need to be satisfied:

$$\begin{cases} c + x * g = c + a * m, \\ d + y * g = d + a * n, \\ c + x * b' = c + b * m, \\ d + y * b' = d + b * n. \end{cases} \tag{2}$$

Unique solutions of (2) have the following form:

$$\begin{cases} x = g^{-1} * a * m, \\ y = g^{-1} * a * n, \\ b' = g * a^{-1} * b. \end{cases} \tag{3}$$

Using the obtained solution we can complete the relabelling procedure of the lower layer of our example graph, as shown in Fig. 14.

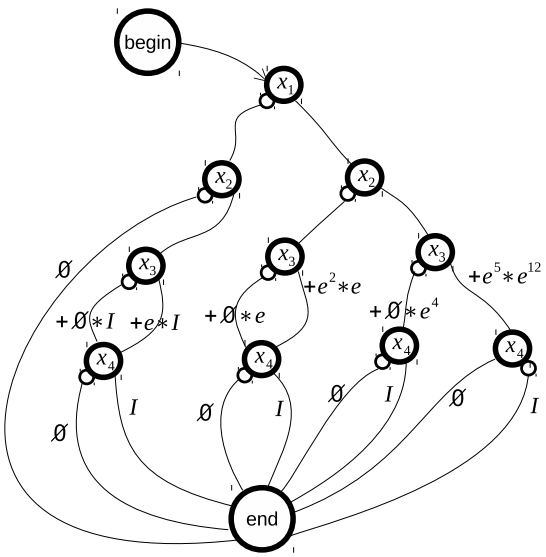


Fig. 14: Canonical relabelling of the lower layer of the graph in Fig. 12.

After the canonical relabelling all nodes of the lower layer become equivalent and can therefore be merged as shown in Fig. 15.

We continue the process by relabelling the previous (third) layer, leading the graph shown in Fig. 16 containing two new equivalent nodes that will further be merged. By proceeding analogously we complete the graph relabelling obtaining the graph shown in Fig. 17.

How do we use the obtained computation graph? We start at the node begin and traverse the graph according to the values of conditions $x_1..x_4$ until we reach

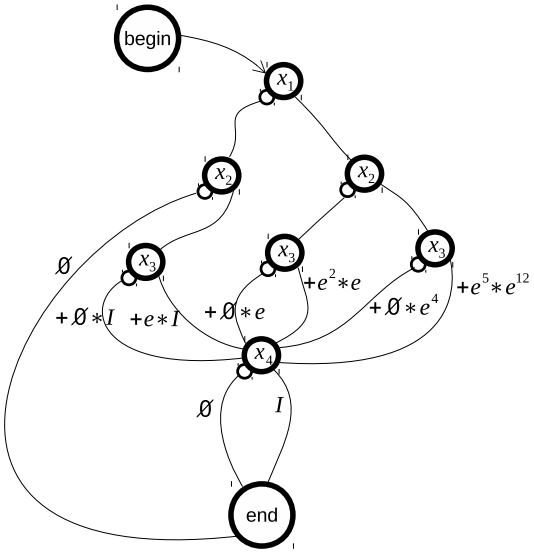


Fig. 15: Merging equivalent nodes of the lower layer in the graph in Fig. 14.

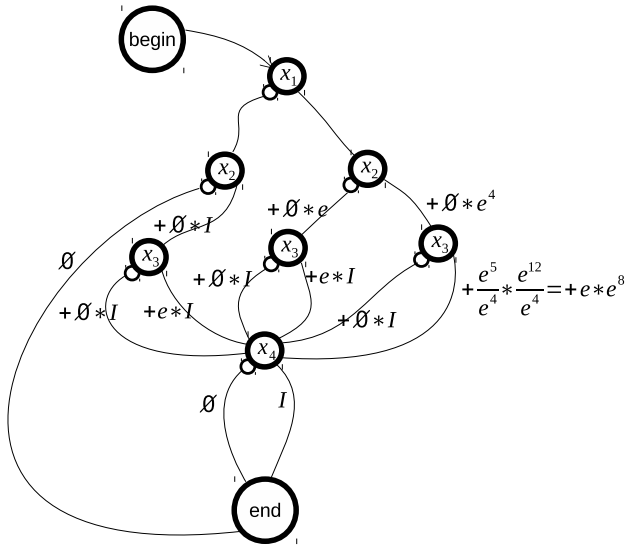


Fig. 16: Relabelling of the third layer.

the node end, writing down the computation result in the form of an expression in Polish prefix notation. We then evaluate the expression and interpret the result as a 4-bit Boolean vector $y_1..y_4$. For example, let $x_1 = x_2 = x_3 = x_4 = 1$, i.e. the input Boolean vector is 1111. This vector corresponds to the rightmost

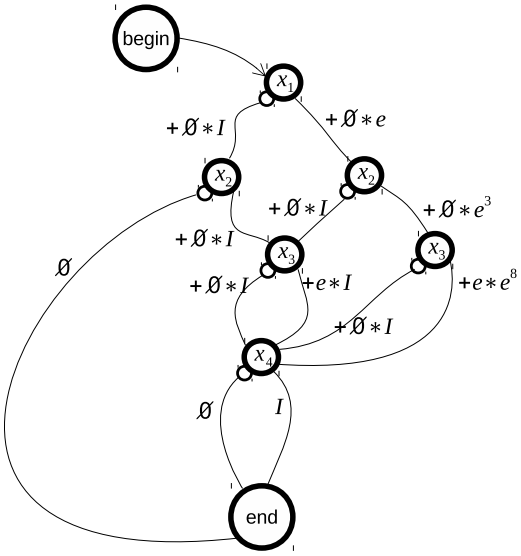


Fig. 17: Complete graph relabelling.

computation path in the graph in Fig. 17. The resulting expression is $(+\emptyset * e + \emptyset * e^3 + e * e^8)I$. We clarify the evaluation of the expression in Fig. 18. The resulting Boolean vector is 1001, which matches the specification in Tab. 3.

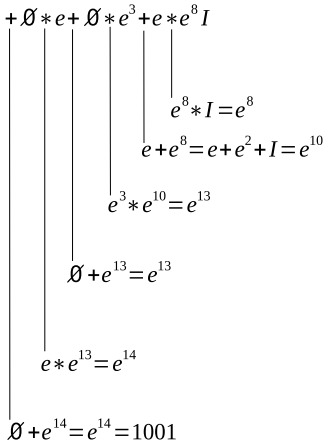


Fig. 18: Evaluating of the resulting expression for input 1111.

The presented approach to the reduction of the size of computation graphs is inspired by *differential BDDs* [5].

b1	b2	b3	b4	b5	Operation
0	0	0	0	0	$r_1; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset$
				1	$r_1; r_2; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset$
			1	0	$r_1; r_2; r_3; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset$
				1	$r_1; r_2; r_3; r_2; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset$
		1	0	0	$r_1; r_2; r_3; r_3; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset$
				1	$r_1; r_2; r_3; r_3; r_2; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset$
			1	0	$r_1; r_2; r_3; r_2; r_3; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset$
				1	$r_1; r_2; r_3; r_2; r_3; r_2; \emptyset; \emptyset; \emptyset; \emptyset$
	1	0	0	0	$r_1; r_2; r_3; r_3; r_3; \emptyset; \emptyset; \emptyset; \emptyset; \emptyset$
				1	$r_1; r_2; r_3; r_3; r_3; r_2; \emptyset; \emptyset; \emptyset; \emptyset$
			1	0	$r_1; r_2; r_3; r_3; r_2; r_3; \emptyset; \emptyset; \emptyset; \emptyset$
				1	$r_1; r_2; r_3; r_3; r_2; r_3; r_2; \emptyset; \emptyset; \emptyset$
		1	0	0	$r_1; r_2; r_3; r_2; r_3; r_3; \emptyset; \emptyset; \emptyset; \emptyset$
				1	$r_1; r_2; r_3; r_2; r_3; r_3; r_2; \emptyset; \emptyset; \emptyset$
			1	0	$r_1; r_2; r_3; r_2; r_3; r_3; r_2; \emptyset; \emptyset; \emptyset$
				1	$r_1; r_2; r_3; r_2; r_3; r_2; r_3; r_2; \emptyset; \emptyset$
1	0	0	0	0	$r_1; r_2; r_3; r_3; r_3; r_3; \emptyset; \emptyset; \emptyset; \emptyset$
				1	$r_1; r_2; r_3; r_3; r_3; r_3; r_2; \emptyset; \emptyset; \emptyset$
			1	0	$r_1; r_2; r_3; r_3; r_3; r_2; r_3; \emptyset; \emptyset; \emptyset$
				1	$r_1; r_2; r_3; r_3; r_3; r_2; r_3; r_2; \emptyset; \emptyset$
		1	0	0	$r_1; r_2; r_3; r_3; r_2; r_3; r_3; \emptyset; \emptyset; \emptyset$
				1	$r_1; r_2; r_3; r_3; r_2; r_3; r_3; r_2; \emptyset; \emptyset$
			1	0	$r_1; r_2; r_3; r_3; r_2; r_3; r_2; r_3; \emptyset; \emptyset$
				1	$r_1; r_2; r_3; r_3; r_2; r_3; r_2; r_3; r_2; \emptyset$
	1	0	0	0	$r_1; r_2; r_3; r_2; r_3; r_3; r_3; \emptyset; \emptyset; \emptyset$
				1	$r_1; r_2; r_3; r_2; r_3; r_3; r_3; r_2; \emptyset; \emptyset$
			1	0	$r_1; r_2; r_3; r_2; r_3; r_3; r_2; r_3; \emptyset; \emptyset$
				1	$r_1; r_2; r_3; r_2; r_3; r_3; r_2; r_3; r_2; \emptyset$
		1	0	0	$r_1; r_2; r_3; r_2; r_3; r_2; r_3; r_3; \emptyset; \emptyset$
				1	$r_1; r_2; r_3; r_2; r_3; r_2; r_3; r_3; r_2; \emptyset$
			1	0	$r_1; r_2; r_3; r_2; r_3; r_2; r_3; r_2; r_3; \emptyset$
				1	$r_1; r_2; r_3; r_2; r_3; r_2; r_3; r_2; r_3; r_2$

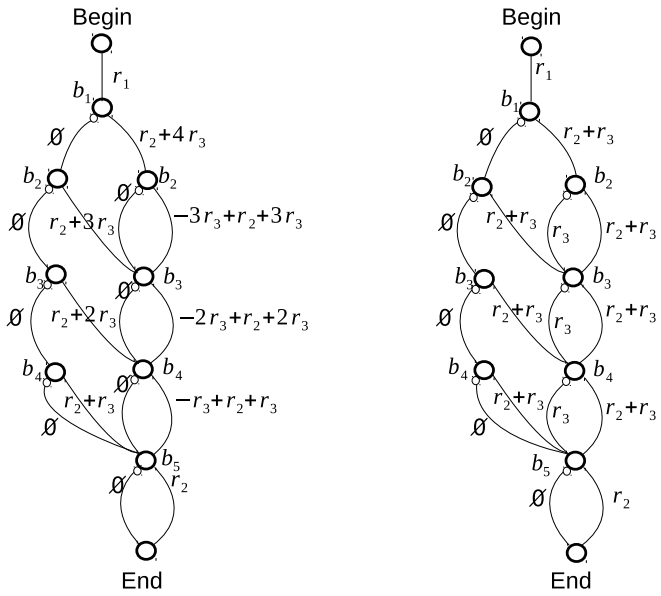
Table 4: Decision tree for a 5-bit sequential multiplier.

4 Example of a sequential computation process

Consider a sequential computation process corresponding to the multiplier of 5-bit non-negative integer numbers. We have a 10-bit asynchronous accumulating register Y that supports three operations: i) add a given 5-bit value A to its current content: $Y \leftarrow Y + A$, ii) double the currently stored value: $Y \leftarrow Y + Y$, and iii) reset to zero: $Y \leftarrow 0$. The complete binary decision tree for all input vectors is shown in Tab. 4; the table uses the following notation for brevity:

- $\emptyset$ corresponds to the no-op (doing nothing),
- r_1 corresponds to the reset operation: $Y \leftarrow 0$,
- r_2 corresponds to addition: $Y \leftarrow Y + A$,
- r_3 corresponds to doubling: $Y \leftarrow Y + Y$.

A list of operations separated by semicolons corresponds to the sequential execution of listed operations from left to right.



(a) Canonical. (b) Not canonical.

Fig. 19: Computation graphs for the 5-bit multiplier.

The leafs of the decision tree are all distinct, therefore the only way to achieve any reduction in the size of the computation graph is to introduce an algebraic structure on the arc labels. Let us first consider label r_1, which stands for the reset of the register. To define an additive group we use $\emptyset$ as the zero label, and introduce the negative label $-r_1$, which stands for *undo the reset* operation. Supporting such undo operations in hardware may be overly expensive, therefore we do not want them to appear the final computation graph, however we are still free to use them analytically in intermediate derivations. We extend the additive group on labels r_2 and r_3 in a similar manner, although their negative labels have more straightforward hardware implementations: $-r_2$ means *subtract A from* Y: $Y \leftarrow Y - A$, while $-r_3$ means *divide* Y *by 2*: $Y \leftarrow Y/2$. The resulting group is defined by the following equations:

- $r_1 + r_1 = 2 \cdot r_1,\ r_1 - r_1 = \emptyset, -r_1 - r_1 = -2 \cdot r_1,\ \ldots$;
- $r_2 + r_2 = 2 \cdot r_2,\ r_2 - r_2 = \emptyset, -r_2 - r_2 = -2 \cdot r_2,\ \ldots$;
- $r_3 + r_3 = 2 \cdot r_3,\ r_3 - r_3 = \emptyset, -r_3 - r_3 = -2 \cdot r_3,\ \ldots$.

Using the relabelling system of equations similar to (1), one can derive the canonical computation graph shown in Fig. 19a. We have not considered the

optimality with respect to the register operations, however, and one can see that some of the resulting computation labels contain redundant shifting operations (doubling labels r_3 followed by immediate division by 2 labels $-r_3$). This is a cost of choosing a particular canonical relabelling. If canonicity can be sacrificed, it is possible to obtain a more efficient labelling shown in Fig. 19b, which avoids negative labels.

References

[1] Alex Semenov, Alex Yakovlev: "Combining Partial Orders and Symbolic Traversal for Efficient Verification of Asynchronous Circuits", 1995.

[2] "Binary Decision Diagram (BDD) adiabatic charging logic circuit", EP 1331738 A1, http://www.google.com.tr/patents/EP1331738A1?cl=en

[3] Aiqun Cao, Cheng-Kok Koh: "Non-crossing ordered BDD for physical synthesis of regular circuit structure", School of Electrical and Computer Engineering, Purdue University West Lafayette, IN 47907-1285, 2003.

[4] Robert Wille, Oliver Keszocze, Clemens Hopfmuller, Rolf Drechsler: "Revers BDD-based synthesys for splitter-free optical circuits", Institute of Computer Science, University of Bremen, Germany, Cyber Physical Systems.

[5] Anuchit Anuchitanukul, Zohar Manna, Toms E. Uribe: "Differential BDDs", In J. van Leeuwen, ed, Computer Science Today , Lecture Notes in Computer Science, vol. 1000, pp. 218-233, Springer-Verlag, 1995.

Asynchronous Clocks

Simon Moore
University of Cambridge

Abstract. Asynchronous circuits typically operate in a clock-free manner. That said, low-level timing characteristics like equipotential regions and matched delays are often employed in self- timed circuits, a class of asynchronous circuits. This paper takes this a step further and reviews approaches to generating clocks inspired by asynchronous circuits, from frequency distribution using Muller C-element chains through to pausible clocks and asynchronously oscillating grids.

1 Introduction

After many years of discussion with Professor Alex Yakovlev and Professor David Kinniment in Newcastle, and other members of the asynchronous circuits community, I am fortunate to have gained a deeper understanding timing in circuits. With that understanding brings enlightenment but not always back-and-white clarity. The question of what is a clock and what is not a clock is a grey area when one looks closely. Proponents of asynchronous (or *self-timed*) circuits believe that clocks are an evil and that clock-less circuits have many virtues. This paper reviews the heretical approach of using asynchronous (clock-less) circuits to generate clocks, and how the boundary between asynchrony and synchrony can be blurred.

2 Clocking basics

In its simplest form, a clock for a digital circuit comes from a precision timing source like a quarts crystal. The precision timing source is then distributed across a chip, to the clock inputs of components like the D flip-flop (DFF). The DFFs provide storage of state and also control the rate of data propagation by delaying data output until the next clock edge. Thus, data is advanced on the clock edge. To provide the illusion that all state updates happen simultaneously (so called *synchronous digital circuits* or *clocked digital circuits*), the clocks to each DFF are expected to arrive simultaneously (synchronously). This provides the illusion of discrete (digital) time to go with the discrete (digital) signal levels. In practise, a truly synchronous clock does not exist. Instead we must be satisfied with a close approximation that, within tolerances (e.g. setup and hold times of the DFFs), provides an accurate enough implementation of the desired synchronous abstraction, which arguably makes the circuit designer's life easier.

3 Asynchronous clock source

Many clocked circuit designers like to provide an external clock locked to a highly stable quartz crystal. It is ironic that the performance of their circuits will vary with temperature, so the clock frequency has to be set against the worst case path delay between synchronising elements at the worst case temperature. Much performance is, therefore, thrown on the floor when the circuits are operating at more typically temperatures. But this approach does preserve the digital time abstraction.

External crystals typically operate at a much lower rate than the desired clock frequency. A phase locked loop is often used to multiply this lower frequency stable clock up to a higher frequency on-chip clock.

An alternative clock generation strategy is to use a delay-line. Some low-cost microcontrollers simply use an inverter ring to provide a clock frequency. Typically the resulting clock frequency varies significantly between devices and with device temperature. We investigated the possibility of constructing a tuneable delay-line that can be self-calibrated from a low-frequency and power-efficient watch crystal [7]. An overview is presented in Figure 1 with details of the delay-line cell in Figure 2.

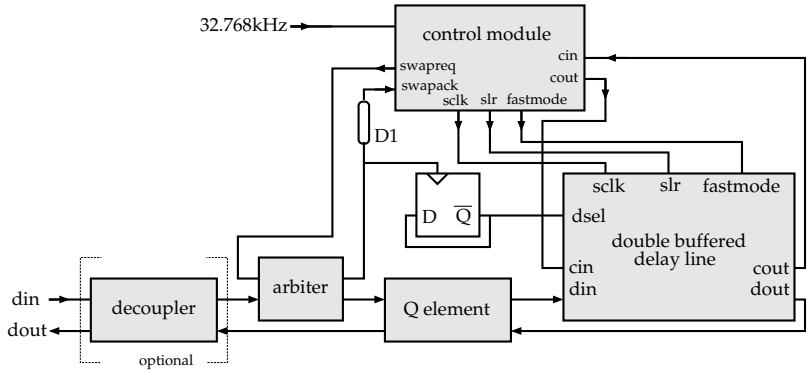


Fig. 1. Overview of an asynchronous self-calibrating delay-line (from [7])

Critical to the functional behaviour is the Q-element [5] used to send both rising and falling events through the delay line before acknowledging dout. The Q-element ensures that the arbiter is not released until the delay line has been through both rising and falling edge phases. Analysis of the behaviour of our Q-element implementation (Figure 4) was undertaken using signal transition graphs (STGs) [8, 9], a form of Petri net, with assistance from the Petrify tool [2]. Professor Yakovlev was pivotal in establishing STGs and the creation of the Petrify tool.

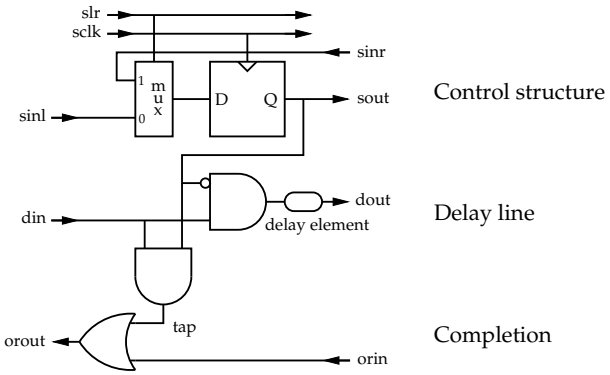


Fig. 2. Asynchronous delay-line cell (from [7])

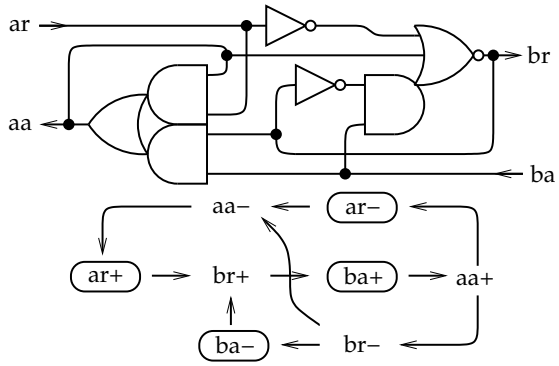


Fig. 3. Asynchronous decoupler with behaviour as an STG (from [7])

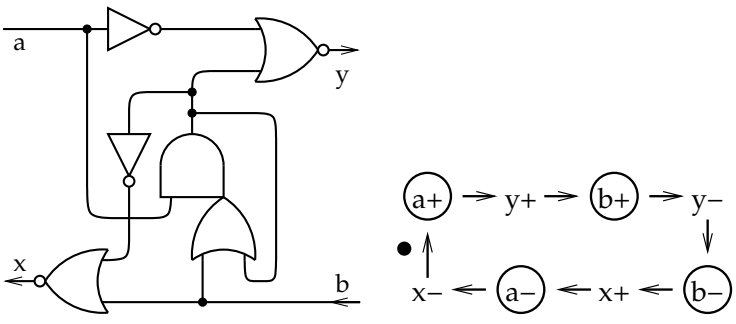


Fig. 4. Asynchronous Q-element with behaviour as an STG (from [7])

4 Asynchronous clock distribution

Clock distribution is the art of broadcasting a clock across a chip so that its frequency and phase appear identical at every clocked element (e.g. DFF). Frequency distribution, in contrast is rather easier. One could, for example, construct a long chain of inverters (Figure 5a) and arrange them in a serpentine manner over the surface of the chip. This would (almost) manage to broadcast the frequency. I say "almost" because a pulse proceeding down an inverter chain will undergo pulse shrinkage, so it is unlikely to reach the end of a long chain. On the other hand, an asynchronous micropipeline made of Muller-C elements (Figure 5b) will successfully distribute the frequency and will guarantee that pulse shrinkage will never obliterate a pulse as it carefully copies the pulse (or *token*) to the next Muller C-element before destroying the source.

A conventional clock distribution approach uses a H-tree fractal over the surface of the chip. This works quite well, though still presents potential discontinuities in clock phase (e.g. see nodes A and B in Figure 6 which are clocked from different branches but are physically adjacent). Self-calibration in the tree can help. Also, sometimes a grid is used at the lowest level to crowbar the H-tree leaves together.

Rather than drive a grid from a H-tree, Dr Scott Fairbanks and I investigated the use of a micropipeline structure laid out as a grid to form a self-oscillating clock distribution system (both frequency and phase) [3]. This originated from earlier work on the one-dimensional asP micropipline structure [1] (see Figure 7a) and was evolved into a two-dimensional structure (Figure 7c). The grid inputs are mixed using the circuit in Figure 8. Pull-up and pull-down nodes are alternated across a grid. Pull-up nodes use the mixer to identify when the majority of inputs are low and then pulls high. Pull-down nodes do the inverse. Thus the grid oscillates in unison and measurements indicate very low skew even in the presence of device variability.

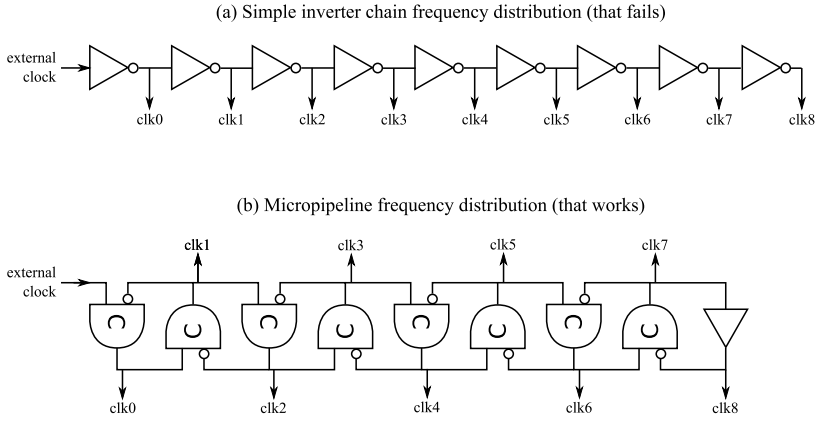


Fig. 5. (a) inverter and (b) micropipeline clock frequency distribution

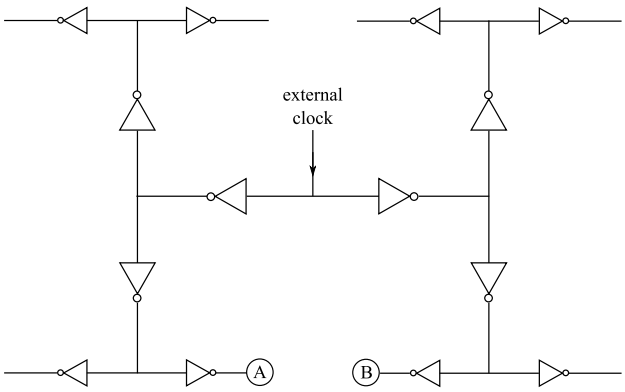


Fig. 6. Simplified H-tree clock distribution

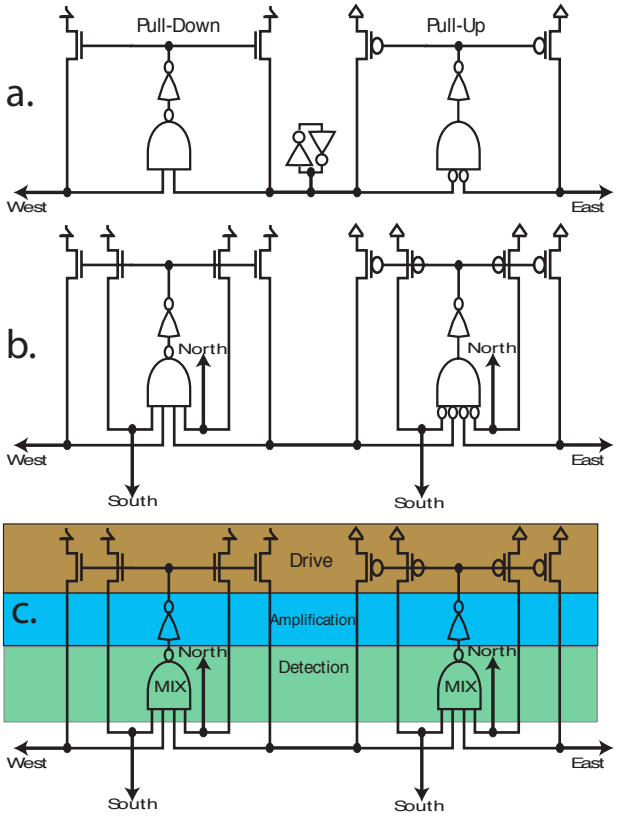


Fig. 7. Evolution from dynamic asP to a distributed clock generator (from [3])

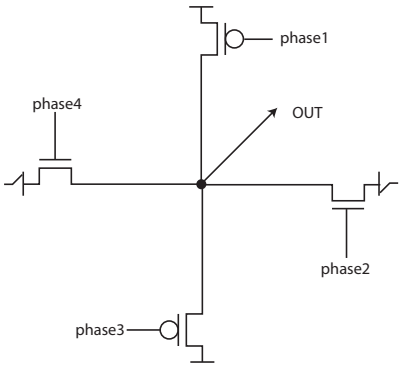


Fig. 8. Clock mixer for the distributed clock generator (from [3])

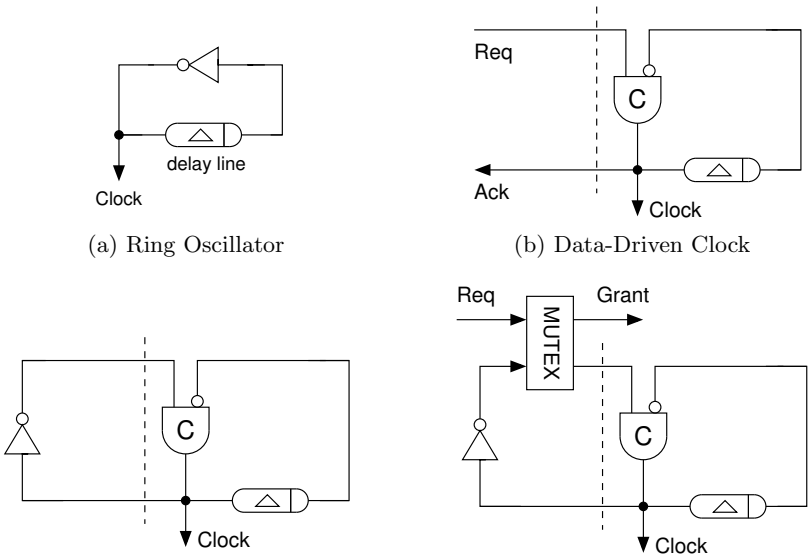
5 Globally asynchronous but locally synchronous circuits

Given that global synchronisation is difficult to achieve, one option is to build chips which are globally asynchronous but locally synchronous (GALS). Since local synchrony is easier to achieve, it allows the clock (synchronous) design method to be used in the small (e.g. a processor core) with asynchronous interconnect between these clocked islands. Global frequency distribution might still be used to control the rate of transfer of information between blocks, making it easy to use credit-based flow control.

Moving data between synchronous domains is not without its problems, however. Sampling a "data ready" bit or some other flow control information coming from another clock domain is likely to result in metastability in the sampling flip-flop. Using a two-flop synchroniser is one approach and with careful design it is possible to reduce the mean-time between failure (MTBF) to once in the lifetime of the universe [4]. However, with incorrect design, or device variability reducing the performance of the sampling flip-flop, the MTBF can easily become less than a minute.

In order to avoid metastability altogether, it is possible to use pausible clocks to ensure completely safe data transfer. Dr Robert Mullins and I undertook a great deal of work in this area with the key final paper being *Demystifying Data-Driven and Pausible Clocking Schemes* [6]. Dr Robert Mullins and I were delighted to collaborate with Professor David Kinniment and Professor Alex Yakovlev on the book *Synchronization and Arbitration in Digital Systems* [4] with several circuits from [6] being reproduced.

Pausible clocks are based around the use of a delay line clock source (Figure 9a) that can be transformed into a data driven clock (Figure 9b) where a local clock signal is produced whenever there is new input data. This is, however, rather restrictive since one typically requires that the local clock continues to oscillate regardless of whether there is new data or not. To this end, the circuit in Figure 9d (an evolution from the circuits in Figure 9a–c) can be used so that



(a) Ring Oscillator (b) Data-Driven Clock

(c) Ring Oscillator (II) (d) Pausible Clock

Fig. 9. Pausible and Data-Driven Local Clocks (from [6])

the clock is only paused to safely transfer new data. Using this basic concept, a complete GALS system can be produced (see Figure 10). For further details, see [6].

Conclusions

Just as digital circuits abstract the analog world into discrete ones and zeros, clocked synchronous circuits abstract continuous time into discrete ticks. In much the same way that it can be useful to analyse digital circuits in their true analog form, it can also be helpful to analyse the true asynchronous (or analog-time) behaviour using techniques like STGs that Professor Yakovlev has been pivotal in creating. Moreover, the ability to mix clocked and asynchronous circuits enables a broader range of design tradeoffs. As we face challenges in clock distribution and device variability for future CMOS circuits, asynchronous techniques may well become critical to meet design requirements. Finally, it should be noted that we can use asynchronous techniques to control and generate clocks, blending synchrony with asynchrony.

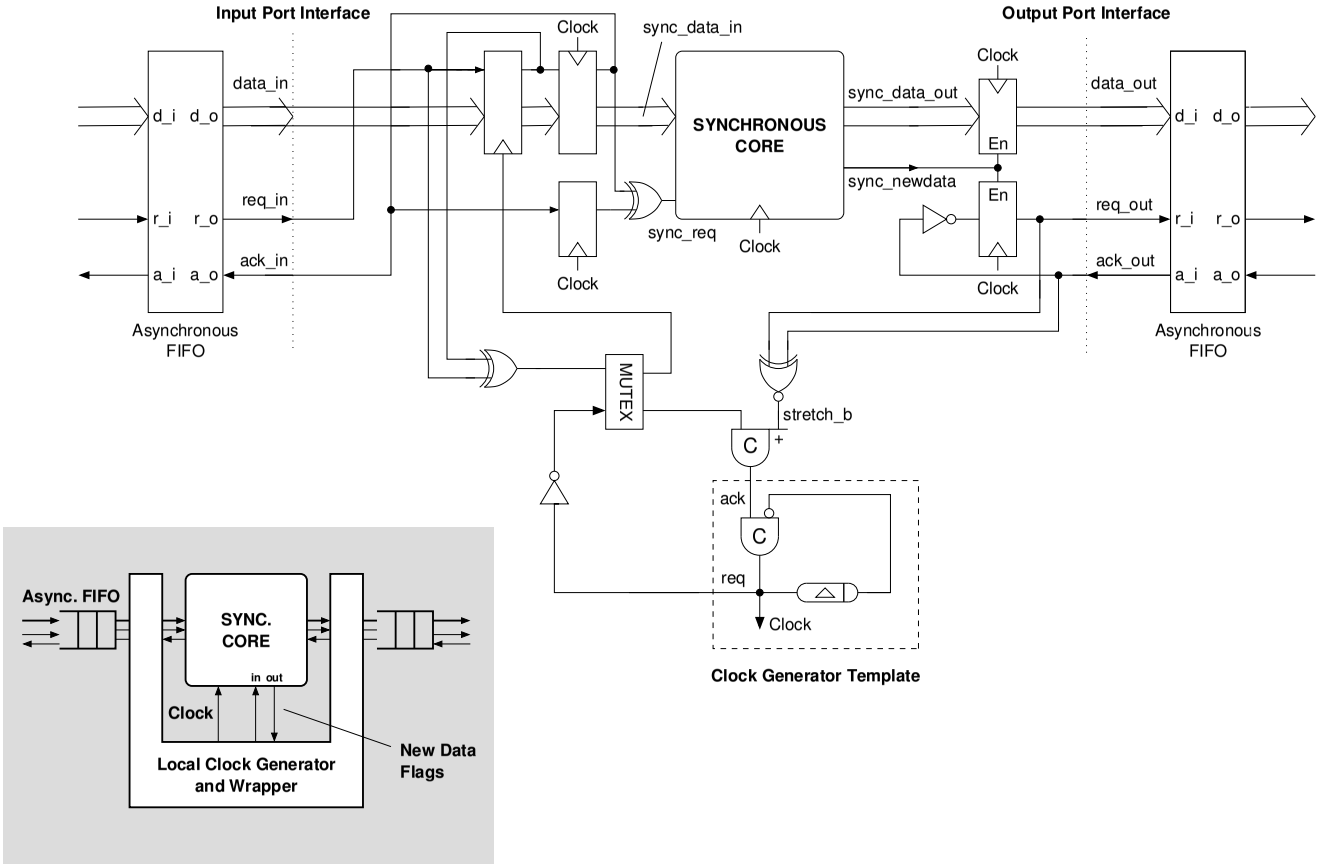


Fig. 10. A locally-clocked synchronous block with a pausible-clock input port and registered/stretchable-clock output port (from [6])

References

1. Control structure for a high- speed asynchronous pipeline (1999)
2. Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. IEICE Transactions on information and Systems 80(3), 315–325 (1997)
3. Fairbanks, S., Moore, S.W.: Self-timed circuitry for global clocking. In: 11th IEEE International Symposium on Asynchronous Circuits and Systems. pp. 86–96 (March 2005)
4. Kinniment, D.J.: John Wiley & Sons, Ltd (2008)
5. Martin, A.J.: Synthesis of asynchronous VLSI circuits. In: Straunstrup, J. (ed.) Formal Methods for VLSI Design, chap. 6, pp. 237–283. North-Holland (1990)
6. Mullins, R., Moore, S.: Demystifying data-driven and pausible clocking schemes. In: 13th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'07). pp. 175–185 (March 2007)
7. Taylor, G., Moore, S., Wilcox, S., Robinson, P.: An on-chip dynamically recalibrated delay line for embedded self-timed systems. In: Advanced Research in Asynchronous Circuits and Systems, 2000. (ASYNC 2000) Proceedings. Sixth International Symposium on. pp. 45–51 (2000)
8. Yakovlev, A., Lavagno, L., Sangiovanni-Vincentelli, A.: A unified signal transition graph model for asynchronous control circuit synthesis. In: Computer-Aided Design, 1992. ICCAD-92. Digest of Technical Papers., 1992 IEEE/ACM International Conference on. pp. 104–111 (Nov 1992)
9. Yakovlev, A., Lavagno, L., Sangiovanni-Vincentelli, A.: A unified signal transition graph model for asynchronous control circuit synthesis. Formal Methods in System Design 9(3), 139–188 (1996)

Asynchronous Circuit Design and Beyond

Chris J. Myers[1]

University of Utah, Salt Lake City, UT 84112, USA,
myers@ece.utah.edu,
WWW home page: http://www.async.ece.utah.edu/Myers

Abstract. In the 80s, there was a resurgence of interest in asynchronous circuit design. Technology challenges being faced by synchronous designers led researchers to reconsider the clock-less option. This work led to both new theoretical insights and computational design methods that were ultimately validated in a number of successful demonstration designs. After more than 30 years, there has been some industrial uptake of this research work, but the revolution that we envisioned has not occurred. Instead, the asynchronous mindset has had impact in a number of other domains from formal methods applied to mixed-signal and cyber-physical systems to design methods for synthetic biology. This paper will give a brief account of the asynchronous renaissance and its lasting impact.

Keywords: asynchronous circuit design, Petri nets, formal methods

1 A Brief History of Asynchronous Circuit Design

The universe doesn't allow perfection.
– Stephen Hawking, A Brief History of Time

Asynchronous circuit design has a long history. Many of the earliest mainframe computers including the ILLIAC and ILLIAC II designed at the University of Illinois and the Atlas and MU-5 designed at the University of Manchester in the 1950s and 1960s utilized asynchronous circuits. Asynchronous circuit design was chosen to improve reliability and provide easier maintenance [3]. Asynchronous circuit design though can be challenging due to the need to ensure that every transition on a signal wire is meaningful [14]. An asynchronous computation is coordinated using a *handshaking protocol* in which a circuit is *requested* to perform an operation, and it *acknowledges* completion of the operation. Even a single unwanted glitch on a request or acknowledge signal wire, known as a *logic hazard*, could lead to unintended behavior. A simple way to address this hazard problem is to utilize *synchronous circuit design*, which employs a periodic clock signal to filter out these glitches by allowing sampling of signals to occur only at prescribed times. For this reason, synchronous circuit design has become the dominant design methodology.

During the 1980s, however, asynchronous circuit design experienced a resurgence of interest. In particular, new effective design methodologies were developed to address the challenges to produce hazard-free asynchronous circuits.

There were two main camps: the language-based and the graph-based researchers. The language-based researchers included Alain Martin of Caltech [11] and Martin Rem of Eindhoven University [2]. They and their students developed design methods that started with behavioral descriptions of asynchronous designs written in a high-level language inspired by Hoare's *communication channels* [7], which are then compiled through a series of well defined transformations into a hazard-free asynchronous circuit implementation. The graph-based researchers included, among others, Tam-Anh Chu, a PhD student from the Massachusetts Institute of Technology [4], Teresa Meng, a PhD student from the University of California at Berkeley [13], and Alexander Yakovlev and Victor Varshavsky of the Leningrad Electrical Engineering Institute [19, 23]. These researchers and their colleagues developed design methods that started with graphical models, typically some variation on a *Petri net* [17], for their asynchronous designs. Using these models, all reachable states would be considered, sometimes implicitly, to produce asynchronous logic circuits free of hazards.

The late 80s and early 90s witnessed several exciting demonstrations of these new methodologies to produce asynchronous circuits with improved performance, power consumption, and robustness. The first fully asynchronous microprocessor was designed by Martin's group at Caltech in 1989 [12]. Utilizing techniques inspired by Ivan Sutherland's Turing Award paper on *micropipelines* [21], Steve Furber's group at the University of Manchester designed the first asynchronous microprocessor that was code-compatible with an existing synchronous processor [24]. At Phillips Research Laboratories, former students and colleagues of Martin Rem designed several low power circuits for commercial applications [1]. Finally, between 1995 and 1999, researchers working with Intel on the RAPPID project designed an asynchronous instruction-length decoder for the Pentium processor with three times better performance while consuming half the power [20].

These successes enabled several startup companies to explore asynchronous circuit design during the late 90s and early 2000s. For example, Alain Martin created Situs Logic, while his former students created Fulcrum Microsystems and Achronix Semiconductor. Other startup companies originated from Steve Furber's group including Cogency and Silistix. Finally, the Phillips group spun out a company called Handshake Solutions. These companies though faced the tremendous inertia in the semiconductor industry. The asynchronous solution had to provide not only substantial benefits, but it also must solve problems that could not be solved by synchronous methods. Therefore, the success of these startup companies has been quite limited to date.

2 The Asynchronous Mindset

> They are really smart, and we can teach them to do something real.
> – Manpreet Khaira, Top 10 Reasons to Hire an Asynchronous Designer

After the success of the RAPPID project, Intel hired numerous asynchronous researchers to join their Strategic CAD Labs that was led by Manpreet Khaira.

The quote above was from a panel discussion at the 1999 Symposium on Advanced Research in Asynchronous Circuits and Systems held in Barcelona Spain. This was the same conference in which the RAPPID results were first published and won the best paper award. Although perhaps a bit insulting to an asynchronous designer, I would argue that it need not be. The "asynchronous mindset" is indeed a very powerful tool that can be applied to a wide variety of applications. The world around us is inherently asynchronous, so those trained to reason in this way are perfectly equipped to be successful. This section briefly highlights three such research areas, ordered in increasing distance from electronic circuits, that we have applied, with some success, an "asynchronous mindset".

2.1 Formal Verification of Analog/Mixed-Signal Circuits

In order to design asynchronous circuits, one must reason about time as a continuous rather than a discrete variable. *Analog/mixed-signal* (AMS) circuits take this one step further and also require values to be represented as continuous variables. In this domain, we have extended a Petri net graph-based modeling formalism utilized for asynchronous design to represent these continuous voltages and currents [9]. This has enabled AMS circuits to be modeled and verified using techniques originally developed for the verification of asynchronous circuits [6]. Most recently, in collaboration with Alexander Yakovlev's group at Newcastle University, we have been closing the design loop by leveraging these formal techniques to optimize the design of asynchronous digital controllers for analog circuits [5].

2.2 Formal Verification of Cyber-Physical Systems

Cyber-physical systems (CPS) are any systems that tightly integrate computation, communication, and control with the physical world. These can include anything from nuclear power plants, to robotic surgeons, to autonomous vehicles. The physical world that these systems must interact with is inherently noisy, stochastic, continuous, and asynchronous. Since these systems are clearly safety critical, it is essential that they be formally verified to avoid catastrophic failures. Once again, a Petri net inspired modeling formalism, similar to the one used for asynchronous and AMS systems, can be efficiently employed to model and verify these systems [22]. We have also recently used a channel-level language inspired by the one that we used for asynchronous design [14] to model and formally verify a fault-tolerant asynchronous routing protocol for an automotive application [25].

2.3 Genetic Design Automation

Synthetic biology is an exciting area of research in which scientists are attempting to engineer new biological systems to solve a wide variety of environmental,

energy, and health problems. *Genetic design automation* (GDA) research is providing the design methods and tools to support this discipline [15, 16]. Since biological systems do not have a global clock, it only makes sense that these methods should be adopted from the asynchronous domain. To this end, we have abstracted biochemical models into asynchronous logical models, once again using an adapted Petri net formalism, enabling orders of magnitude more efficient analysis [8, 10]. We have also adapted asynchronous logic synthesis techniques to produce genetic circuit designs automatically [18]. Finally, we hope to some day potentially draw inspiration from the design in this very noisy environments to produce more robust asynchronous electronic circuits.

3 Discussion

While the asynchronous revolution that many of us hoped for when we began working in this area of research has not yet occurred, I believe we can take solace in the fact the "asynchronous mindset" is and will continue to have tremendous impact in a wide variety of fields. A continued emphasis in instilling this view of the world in our students make them better equipped to tackle the wide variety of problems that they will face in the ever changing asynchronous world.

References

1. Berkel, K.v., Burgess, R., Kessels, J., Peeters, A., Roncken, M., Schalij, F.: A fully-asynchronous low-power error corrector for the DCC player. IEEE Journal of Solid-State Circuits 29(12), 1429–1439 (Dec 1994)
2. Berkel, K.v., Rem, M.: VLSI programming of asynchronous circuits for low power. In: Birtwistle, G., Davis, A. (eds.) Asynchronous Digital Circuit Design. pp. 152–210. Workshops in Computing, Springer-Verlag, New York (1995)
3. Brearley, H.C.: ILLIAC II: A short description and annotated bibliography. IEEE Transactions on Computers 14(6), 399–403 (Jun 1965)
4. Chu, T.A.: Synthesis of Self-Timed VLSI Circuits from Graph-Theoretic Specifications. Ph.D. thesis, MIT Laboratory for Computer Science (Jun 1987)
5. Dubikhin, V., Myers, C.J., Yakovlev, A., Sokolov, D.: Design of mixed-signal systems with asynchronous control. IEEE Design & Test Magazine (2016)
6. Fisher, A., Batchu, S., Jones, K., Kulkarni, D., Little, S., Walter, D., Myers, C.: Lema: A tool for the formal verification of digitally-intensive analog/mixed-signal circuits. In: Circuits and Systems (MWSCAS), 2014 IEEE 57th International Midwest Symposium on. pp. 1017–1020 (Aug 2014)
7. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall, Englewood Cliffs, NJ (1985)
8. Kuwahara, H., Myers, C., Barker, N., Samoilov, M., Arkin, A.: Automated abstraction methodology for genetic regulatory networks. Trans. Comp. Syst. Biol. VI, 150–175 (2006)
9. Little, S., Walter, D., Myers, C., Thacker, R., Batchu, S., Yoneda, T.: Verification of analog/mixed-signal circuits using labeled hybrid petri nets. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 30(4), 617–630 (2011)

10. Madsen, C., Zhang, Z., Roehner, N., Winstead, C., Myers, C.: Stochastic model checking of genetic circuits. J. Emerg. Technol. Comput. Syst. 11(3), 23:1–23:21 (Dec 2014), http://doi.acm.org/10.1145/2644817
11. Martin, A.J.: Programming in VLSI: From communicating processes to delay-insensitive circuits. In: Hoare, C.A.R. (ed.) Developments in Concurrency and Communication. pp. 1–64. UT Year of Programming Series, Addison-Wesley, Reading, MA (1990)
12. Martin, A.J., Burns, S.M., Lee, T.K., Borkovic, D., Hazewindus, P.J.: The design of an asynchronous microprocessor. In: Seitz, C.L. (ed.) Advanced Research in VLSI. pp. 351–373. MIT Press, Cambridge, MA (1989)
13. Meng, T.H.Y., Brodersen, R.W., Messerschmitt, D.G.: Automatic synthesis of asynchronous circuits from high-level specifications. IEEE Transactions on Computer-Aided Design 8(11), 1185–1205 (Nov 1989)
14. Myers, C.: Asynchronous Circuit Design. John Wiley & Sons (2001)
15. Myers, C.J.: Engineering Genetic Circuits. Chapman and Hall/CRC (2009)
16. Myers, C.J.: Computational synthetic biology: Progress and the road ahead. Multi-Scale Computing Systems, IEEE Transactions on 1(1), 19–32 (2015)
17. Petri, C.A.: Communication with automata. Tech. Rep. RADC-TR-65-377, Vol. 1, Suppl. 1, Applied Data Research, Princeton, NJ (1966)
18. Roehner, N., Myers, C.J.: Directed acyclic graph-based technology mapping of genetic circuit models. ACS Synthetic Biology 3(8), 543–555 (2014), pMID: 24650240
19. Rosenblum, L.Y., Yakovlev, A.V.: Signal graphs: From self-timed to timed ones. In: Proc. of International Workshop on Timed Petri Nets. pp. 199–207. IEEE Computer Society Press, Los Alamitos, CA, Torino, Italy (Jul 1985)
20. Stevens, K., Rotem, S., Ginosar, R., Beerel, P., Myers, C., Yun, K., Kol, R., Dike, C., Roncken, M.: An asynchronous instruction length decoder. IEEE Journal of Solid-State Circuits 35(2), 217–228 (Feb 2001)
21. Sutherland, I.E.: Micropipelines. Communications of the ACM 32(6), 720–738 (Jun 1989)
22. Thacker, R., Jones, K., Myers, C., Zheng, H.: Automatic abstraction for verification of cyber-physical systems. In: Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems. pp. 12–21. ICCPS '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1795194.1795197
23. Varshavsky, V.I. (ed.): Self-Timed Control of Concurrent Processes: The Design of Aperiodic Logical Circuits in Computers and Discrete Systems. Kluwer Academic Publishers, Boston, Dordrecht, The Netherlands (1990)
24. Woods, J.V., Day, P., Furber, S.B., Garside, J.D., Paver, N.C., Temple, S.: AMULET1: An asynchronous ARM processor. IEEE Transactions on Computers 46(4), 385–398 (Apr 1997)
25. Zhang, Z., Serwe, W., Wu, J., Yoneda, T., Zheng, H., Myers, C.: An improved fault-tolerant routing algorithm for a network-on-chip derived with formal analysis. Science of Computer Programming (2016)

Beyond Carrying Coal To Newcastle: Dual Citizen Circuits

Marly Roncken[1], Chris Cowan[1], Ben Massey[1], Swetha Mettala Gilla[1], Hoon Park[1], Robert Daasch[1], Anping He[2], Yong Hei[3], Warren Hunt Jr.[4], Xiaoyu Song[1], and Ivan Sutherland[1]

[1] Portland State University, Portland, Oregon, USA
[2] School of Information Science & Engineering, Lanzhou University, Lanzhou, China
[3] Institute of Microelectronics Chinese Academy of Sciences, Beijing, China
[4] The University of Texas at Austin, Austin, Texas, USA

Abstract. This paper makes self-timed circuits dual citizens by providing a clocked mode of operation in addition to their self-timed mode. The clocked or synchronous mode of operation re-uses the self-timed fabric and protocols, and thereby — beneficially — inherits the elasticity of the self-timed or asynchronous mode of operation. In exchange, clocked circuit operations can build confidence in self-timed circuit operations or replace aging or erratic self-timed circuit operations that need more time to finish. Once confidence is gained, the self-timed mode of operation can serve as a turbo mode to obtain better latency, throughput, energy, robustness to delay variations, or electro-magnetic compatibility. The dual citizen circuits in this paper have individual action control. As a result, the circuits can either run in a fixed mode — self-timed or clocked — and switch modes on the fly, or run in both modes concurrently.

Keywords: self-timed circuits, asynchronous, synchronous, dual mode

Figure 1 Carrying coal to Newcastle, ASYNC 2008.

Foreword

The title of this paper reflects an incident at the ASYNC 2008 conference chaired by Alex. Ivan, having a British mother, grew up knowing the futility of "Carrying Coal to Newcastle." Because ASYNC 2008 was in Newcastle, Ivan seized on the opportunity actually to carry coal to Newcastle. Being at pains to find coal in the San Francisco Bay area where it is a rare household fuel, he finally got a plastic sandwich bag of coal imported from Utah to California. He labeled it "mineral samples" to pass international inspection, and duly delivered it to Alex — see Figure 1.

1 Introduction

This paper expands [8] by not only naturalizing self-timed circuits but by turning them into dual citizens as well. Below, we explain what this means.

The "naturalized communication and testing" view [8] separates self-timed building blocks into links and joints. Links store and transport data. Joints serve as meeting points for links to coordinate state and exchange data. The actions of a self-timed system start in joints, and can be enabled or frozen selectively using separate *go* control signals in each joint. Joints act only when input links are full and output links are empty and *go* is enabled. Actions can be conditional or nondeterministic. For ease of explanation, this paper uses simple FIFO actions — see Figure 2.

Figure 2 shows a joint as a stick figure with data flowing in the direction of the arrow, and links as rectangles. Each link-joint-link triple represents a FIFO with an input link called *in*, and an output link called *out*. The most important property of a link is whether it is full or empty, just as the most important property of a parking place is whether or not it is occupied. We color the inside of a full link blue (grey in black and white print) and leave an empty link white.

Each link reports its full or empty state at both ends. It accepts a fill command at its input end and a drain command at its output end. Fill and drain commands change the state of a link. The impact of fill and drain commands is observed immediately at the near end of the link but may take time to traverse the length of the link before appearing at the link's far end.

The joint in Figure 2 acts only when its input link, *in*, is full and its output link, *out*, is empty, and its *go* signal is enabled. When it acts, it starts three concurrent operations that (1) copy the data from *in* to *out*, (2) drain link *in* — leaving it empty, and (3) fill link *out* — leaving it full. Note that when *go* is disabled, the joint is "frozen," and there is no action. Note also that the data value shown in Figure 2 as 60 stays in link *in* even after being copied — it may stay there until link *in* is filled with new data. By making link *in* empty and link *out* full, the action enables neighboring joints to act while it disables itself. This is what makes a self-timed circuit "tick."

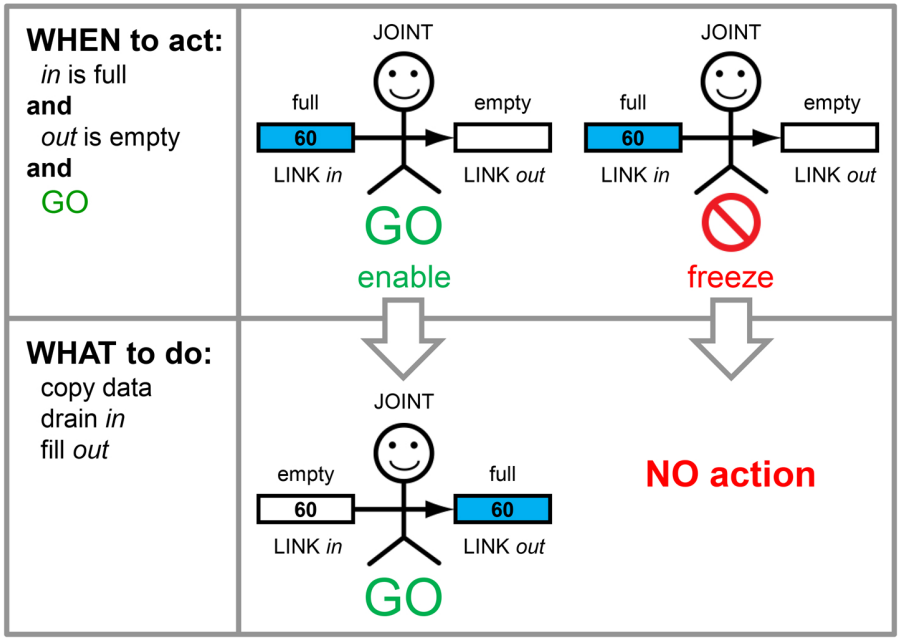


Figure 2 Self-timed FIFO action.

The full-empty protocol in Figure 2 works regardless of its handshake implementation. We advocate using it as standard interface to facilitate mixing and matching self-timed designs from different circuit families. The "naturalized" link and joint view offered in [8] captures the essence of self-timed systems.

This paper expands that view by adding a clocked or *synchronous* mode of operation to a naturalized self-timed circuit. Clock signals that retard the self-timed or *asynchronous* operation can be part of links or part of joints — we show circuits for each. A clocked link announces changes in its full or empty state only at times specified by its clocks. Likewise, a clocked joint acts only at times specified by its clocks. Not only do both self-timed and clocked modes of operation work, but simulations reported here also confirm mixed mode operation.

By providing a self-timed circuit with a clocked mode of operation, we aspire to increase the level of familiarity, comfort, and confidence of VLSI designers to integrate self-timed circuits into their systems. We regard the resulting circuit as both a self-timed and a clocked circuit — just as a "dual citizen" is regarded as a citizen of two countries. We therefore call this circuit a *dual citizen* circuit.

This paper is organized as follows. Section 2 shows a naturalized FIFO implementation for Figure 2 in Click, taken from [8], for use as reference design. Section 3 expands this reference design in two ways, by adding a clocked mode of operation to (1) its joint and (2) its links. Section 4 presents simulations for fixed mode operation, mode switching, and mixed mode operation. Section 5 discusses related work. Section 6 concludes the paper.

2 Naturalized Self-Timed Circuits

Most design methods for self-timed circuits use handshake protocols to encode the full or empty status of a link and validity of the link's data. Figure 3 shows a two-phase single-rail handshake [9] — or as we say a "two-phase non-return-to-zero handshake with bundled data" — and the way it encodes full, empty, and data validity. This protocol is used by the Click self-timed circuit family [6].

Click is the *most synchronous* asynchronous circuit family that we know. Its implementation style was chosen to resemble clocked or synchronous circuits as much as possible. It uses flipflops in every loop and it uses flipflops to store data. The flipflops facilitate the use of conventional optimization, timing, and test tools used for clocked circuits.

Figure 4 shows a self-timed circuit implementation for Figure 2, based on Click and two-phase non-return-to-zero handshake signaling with bundled data, but adapted for "naturalized communication and testing" [8]. The circuit has been adapted by moving the link-joint interface. The original interface separated links and joints at the handshake request, acknowledge, and data signals — here named R, A, and *Dstored*. The new interface separates the links and joints at their natural communication signals: *full*, *drain*, and *Dstored* for links carrying data into a joint, and *empty*, *fill*, and D for links carrying data away from the joint. The new interface takes full advantage of the handshake protocol without exposing it. It thereby diverts any "Tower of Babel" effect that a multitude of handshake protocols in use [9] might create if their handshake signals were exposed to each other.

By "naturalizing" the communication we gain translation-free communication. Moreover, we gain it whilst keeping the peculiarities of each handshake protocol and the specific skills that it supports to create circuits with better latency, throughput, energy, robustness, or electro-magnetic compatibility [1, 4, 5].

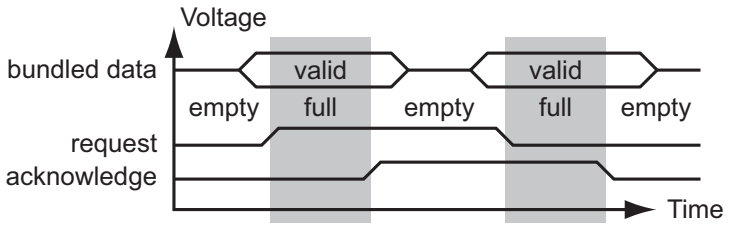


Figure 3 Example of a two-phase non-return-to-zero handshake with bundled data. This protocol has two control signals, *request* and *acknowledge*, and zero or more data signals, also known as *bundled data*. A link using this protocol is full when the voltage levels of its request and acknowledge differ, and empty otherwise. Its data signals are valid when the channel is full. A full channel may be drained, i.e. made empty, and an empty channel may be filled, i.e. made full.

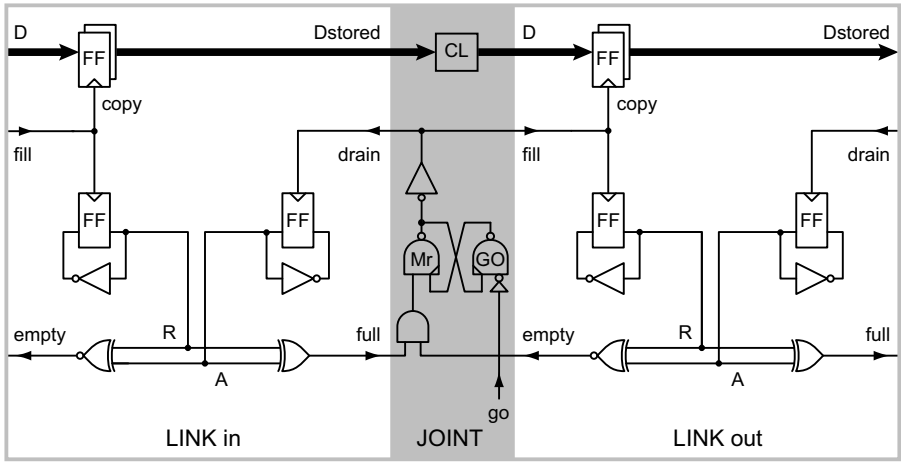


Figure 4 Naturalized Click FIFO circuit presented at ASYNC 2015 [8].

The links in Figure 4 use edge-triggered flipflops to store their full or empty state and the data they transfer. Combinational logic (CL) for datapath operations is kept in the joint. A FIFO that merely fills its output link with data copied from its input link uses simple wire connections for combinational logic.

Each link stores its full or empty state on two signals, a request signal, R, and an acknowledge signal, A. Each fill operation, performed as soon as signal *fill* goes high, changes R, making it differ from A. Each drain operation, performed as soon as signal *drain* goes high, changes A, making it match R. XOR and XNOR gates generate the full or empty state of the link by comparing the signal values of R and A. They report this state to signals *full* and *empty*. The link changes each R and A signal by complementing its value. Because R and A are separate signals, the link has separate flipflops to store the old and new values.

The joint in Figure 4 contains an AND function and the combinational logic for the datapath. The AND function combines the *full* and *empty* signals of links *in* and *out* with a *go* signal. When all three signals are high, the AND function "acts" by making signals *drain* and *fill* both high. Thus the action starts concurrently (1) a drain operation in link *in*, and (2) a fill operation in link *out* that copies the data from link *in*. In turn, the fill and drain operations make both *full* and *empty* signals low, thus disabling the AND function and causing both *drain* and *fill* to go low, which ends the action.

The *go* signal comes with its own arbiter to decide what to do when *go* is low. When *go* is low, the arbiter decides cleanly whether to stop at once or to complete a pending or ongoing action in the joint. The arbitrated circuit is called MrGO, pronounced "Mister GO" — see Figure 5(a). To control actions selectively, each MrGO has its own *go* signal. *We use MrGO for single-step, multi-step, and at-speed test and debug as explained in [8], and for switching between self-timed and clocked modes of operation as explained later in this paper.*

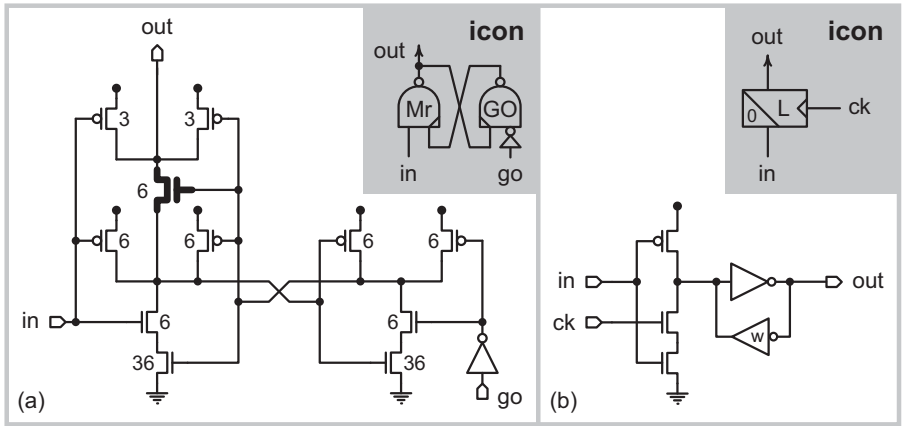


Figure 5 Transistor-level details for (a) MrGO and (b) a zero-passing latch.
(a) *MrGO*
The schematic for MrGO with its icon inset in the grey area is copied from [8], except that here we use correctly matching *in* and *go* parts. When *go* is high, MrGO acts as an inverter from *in* to *out*. When *go* is low, MrGO uses arbitration to decide cleanly whether or not to make *out* high. The bold central transistor delays active-low signal *out* by conducting only after metastability ends in favor of a low *out* signal. Metastability can occur during arbitration decisions, when a high-to-low (falling) transition on *go* to make and keep *out* high concurs with a low-to-high (rising) transition on *in* to make *out* low. Transistors are sized to reduce the logical effort from *in* to *out*. Split pull-up transistors avoid a floating *out* signal. **We use MrGO for single-step, multi-step, and at-speed test and debug as explained in [8], and for switching between self-timed and clocked modes of operation as explained in this paper.**
(b) *Zero-passing latch*
The latch stops a high input signal *in* from propagating until clock signal *ck* is high. When both *in* and *ck* are high, or when *in* is low, output signal *out* copies the value of *in*. Back to back inverters on *out* keep its value. To reduce drive fights, the reverse inverter of the keeper is weak.

3 Dual Citizen Circuits

The link and joint model of a self-timed circuit extends naturally to two solutions with a clocked mode of operation: clock the joint, as in Figures 6, or clock the link, as in Figure 7. The two *dual citizen* circuit solutions in Figures 6–7 both extend the Click circuit of Figure 4. Both circuits re-use the self-timed fabric and protocols. As a result, even in clocked mode, each circuit inherits the elasticity of the self-timed mode of operation to act only when and where needed. This is beneficial not only because it reduces power and saves energy, but also because it simplifies scheduling of clocked operations. With protocols rather than clock cycles in charge of the flow of control, the clocked operations of a dual citizen circuit can function correctly even when operating out of lockstep.

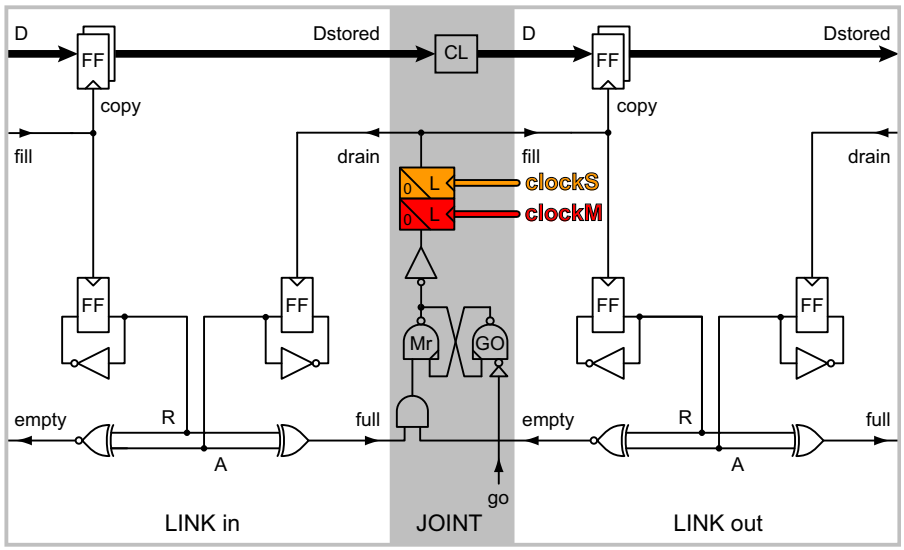


Figure 6 Dual citizen version of Figure 4 with clocks in the Joint.

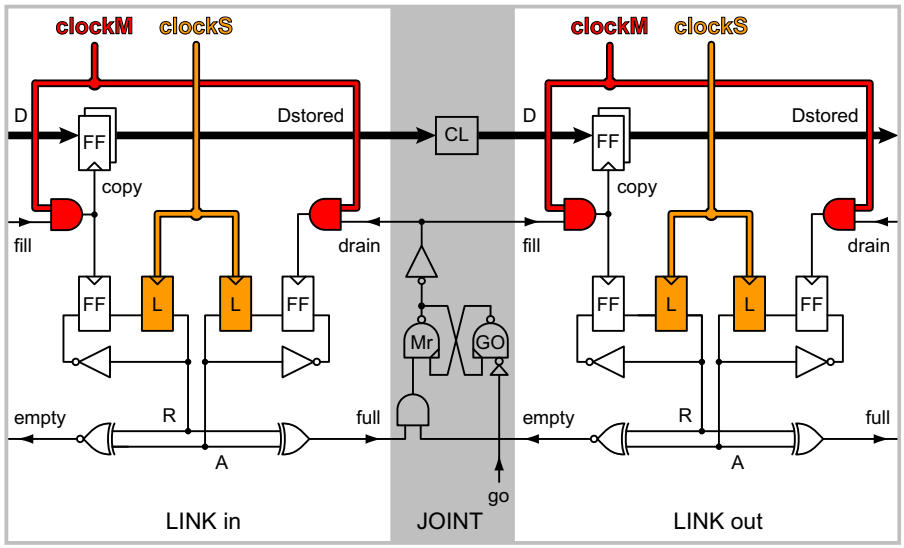


Figure 7 Dual citizen version of Figure 4 with clocks in each Link.

Because they re-use the self-timed fabric, the clocked circuit operations can build confidence in the self-timed circuit operations or replace aging or erratic self-timed circuit operations that need more time to finish.

Once confidence in the correctness of the self-timed operation is established, the self-timed mode can serve as "turbo mode" to obtain better latency, throughput, energy, robustness to delay variations, or electro-magnetic compatibility.

Both circuits use master and slave clocks, *clockM* and *clockS*. For self-timed operation, both clocks remain high. For clocked operation, a high pulse on *clockM* is followed by a high pulse on *clockS*.

The circuit in Figure 6 adds *clockM* and *clockS* at the end of the AND function in the joint where the clocks control a serial pair of zero-passing latches. For a transistor level schematic of a zero-passing latch, see Figure 5(b). In self-timed mode, *clockM* and *clockS* both remain high and the latches remain transparent to amplify the output signal of the MrGO controlled AND function. In clocked mode, each latch passes a low incoming signal by making its output low, but keeps its output as is when the incoming signal is high and its clock is low. A high incoming signal propagates only during a high pulse of the latch clock. Zero-passing latches allow the reset part of the joint action to run unhindered to completion — even in the clocked mode of operation. Because neighboring joints act in mutual exclusion, allowing each action to complete by making its *copy*, *fill*, and *drain* signals low before starting another action in the next clock cycle, facilitates the use of latches instead of edge-triggered flipflops in the datapath.

The key advantages of the dual citizen circuit in Figure 6 are (1) its simplicity and (2) its generality: many self-timed circuit families use the same joints [8]. Its key disadvantage is that the clocks leave some self-timed loops free running:

- The inverting loops of the link flipflops run freely, even in clocked mode. As a result, hold violations on these flipflops, due to an exceedingly fast data inversion loop, will affect both self-timed and clocked modes of operation.
- The action, once started, runs freely to completion. As a result, active-high pulse width violations on copy, fill, and drain signals are beyond clock control, and will affect both self-timed and clocked modes of operation.

The dual citizen circuit in Figure 7 adds *clockM* and *clockS* in each link. This circuit also allows the reset part of a joint's action to run to completion. But it does so while keeping a firm grip on all self-timed loops. The circuit in Figure 7 can repair all aging or erratic self-timed circuit operations that need more time to finish by switching to a clocked mode of operation, and by setting the high and low pulse widths for *clockM* and *clockS* as wide as needed.

In Section 4, we show simulation waveforms of dual citizen circuits interacting in self-timed and in clocked mode. Some interactions use MrGO to control joints selectively. In clocked operations, we change *go* signals during the low phase of *clockM*. None of the simulation scenarios in this paper require the arbitration function of MrGO. But if needed, arbitration in MrGO can be avoided in clocked operations by changing the *go* signal only when both *clockM* and *clockS* are low.

4 Simulation Experiments

We use four simulation scenarios to illustrate what one can do with dual citizen circuits. We simulate a ripple FIFO with ten joints, Joint 1 to Joint 10, and eleven links, Link 0 to Link 10, where Link 0 and Link 10 are connected to the external environment — see Figure 11. We assume that initially all links are empty (low) and all input signals and signal values stored in latches or flipflops are zero (low). For details on initialization, see [8].

All simulations were done in Verilog and use discrete delay models. For gate delays we model the number of signal inversions: each inversion counts as one time step. More precisely: signal changes through INVERTER and NAND gates take one time step. It takes two time steps to go through AND, X(N)OR, FLIPFLOPS, and LATCHES. The environment takes five time steps to respond. Environment actions that fill and drain a link are synchronized with a link's slave clock whenever the link operates in clocked mode, and are self-timed otherwise.[5]

All simulation waveforms shown in this paper are generated using the dual citizen Click FIFO with clocks added to the links, as shown in Figure 7, and with 6-bit wide data signals and simple wire connections for combinational logic. If instead of clocking the links we clock the joints, as shown in Figure 6, the Verilog test benches produce similar waveforms with the same test stimuli and responses.

Sections 4.1–4.3 below discuss the following simulation scenarios:

- Run in fixed mode mode, either self-timed or clocked.
- Switch modes after starting a self-timed operation to finish it clocked.
- Mix modes by running self-timed and clocked operations concurrently.

4.1 Run in Fixed Mode — Self-Timed or Clocked

The simulation waveforms in Figure 8 show the FIFO operating in self-timed mode. Those in Figure 9 show the FIFO operating in clocked mode.

The horizontal axis at the top of both Figures shows the progression of time throughout the course of the operation. The signal waveforms are displayed vertically, row by row. The vertical axis on the left shows the signal names. The signal called *start* indicates the end of initialization — we use it to start the operation cleanly. Any grey-colored, i.e. undefined, waveform values and any waveform changes prior to *start* going high can be ignored.

The master and slave clocks, *clockM* and *clockS*, are both high in Figure 8, as required for self-timed operation, while in Figure 9 they start ticking as soon as *start* goes high. Signals *in_empty*, *in_fill*, and *in_D* go between Link 0 and the environment. Likewise, signals *out_D*, *out_full*, and *out_drain* go between Link 10 and the environment. The remaining signals, *Dstored0* to *Dstored9*, are the data signals stored in Link 0 to Link 9.

[5] The reset part of fill and drain actions remains self-timed at all times.

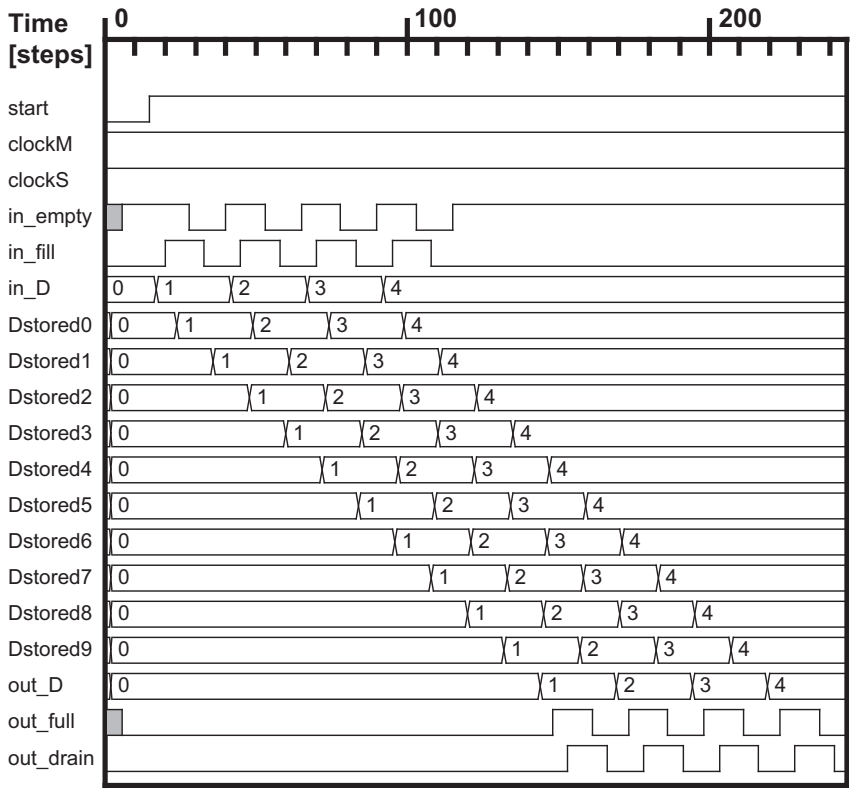


Figure 8 Self-timed FIFO operation transferring four data items.

As soon as *start* goes high, the environment starts communicating with the FIFO, using the protocols on Link 0 and Link 10. The entire operation consists of: (1) the environment sending four data items, with successive values 1 to 4, through Link 0 into the FIFO, (2) the FIFO forwarding these data items from Link 0 to Link 10, and (3) the environment collecting the data items at Link 10.

Note that data values stay in the links until they are overwritten by new values. This is particularly visible for the initial data values of 0 and for the final data values of 4. The FIFO's output data, *out_D*, for instance, keeps its initial value 0 for approximately 140 time steps in Figure 8, which is how long it takes a data item to ripple through the FIFO in self-timed mode. In the clocked mode of operation shown in Figure 9, this takes approximately 700 time steps. Likewise, the value 4 of the last data item stays on all the links after all four data items have rippled through the FIFO.

Figures 8–9 show that both the self-timed and the clocked operation are immune to old data values lingering on links. This is as expected, because both modes of operation obey the same dataflow protocols. Both use the full or empty status of a link to decide when to pay attention to and when to ignore the link's data.

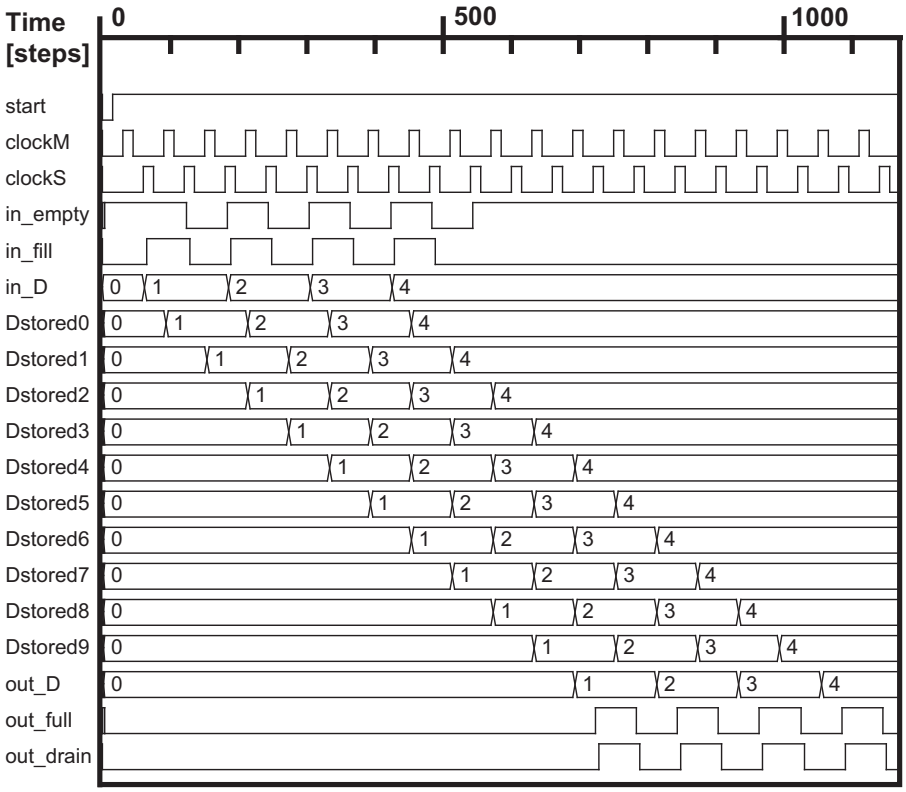


Figure 9 Clocked FIFO operation transferring four data items.

As a result — omitted from Figure 9 — the master and slave clocks in the clocked operation can keep ticking after the operation completes, without jeopardizing any of the final values of status or data signals.

As noted earlier, the clocked mode of operation shown in Figure 9 is slower than the self-timed operation in Figure 8. The clocked operation is slower because the clocks retard the self-timed fabric and protocols. The clock periods must be longer than the worst-case cycle times of the self-timed fabric and protocols.

4.2 Switch Modes — from Self-Timed to Clocked

The simulation waveforms in Figure 10 show the FIFO first operating in self-timed mode and then switching mode to operate in clocked mode. The overall operation is similar to each of the operations shown in Figures 8–9: four data items pass through the FIFO. Approximately the first 250 time steps of the simulation run self-timed. In self-timed mode, we execute the first half of the operation, which consists of (1) the environment sending four data items, with

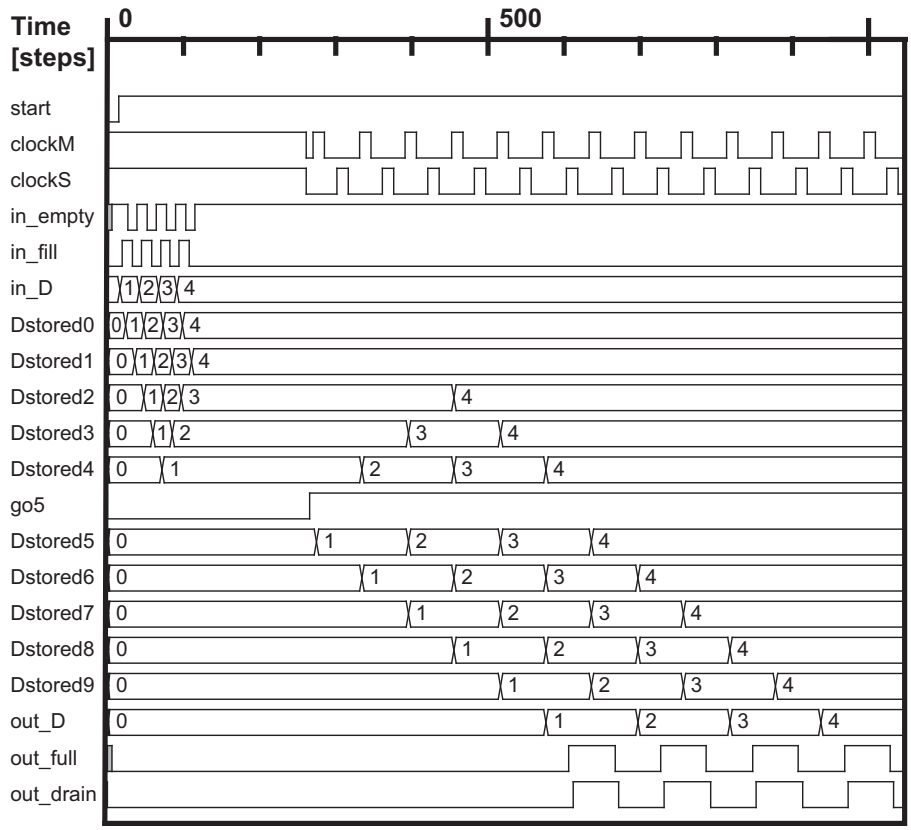


Figure 10 From self-timed to clocked FIFO operation passing four data items.

successive values 1 to 4, through Link 0 into the FIFO, and (2) the FIFO storing these data items in Link 0 to Link 4. The remaining time steps of the simulation run clocked. In clocked mode, we execute the second half of the operation, which consists of (3) the FIFO forwarding the data items stored in Link 0 to Link 4, and (4) the environment collecting the data items at Link 10. In between, the mode of operation switches from self-timed to clocked.

To split the operation in two and switch the mode of execution between the two halves, we deploy MrGO [8]. Specifically, we use go control signal *go5* of Joint 5. Signal *go5* is the only new signal that we inserted into the waveform display of Figure 10 — just below the center. All other signals match those of Figures 8–9.

A pictorial view of the role of *go5* in splitting the operation follows in Figure 11. First, we freeze Joint 5, by making *go5* low — see Figure 11(a). This prevents Joint 5 from acting. Then we run the first half of the operation in self-timed mode. The operation ends in a stable state in which Link 1 to Link 4 are full and the other links are empty — see Figure 11(b). The stable state allows us to switch

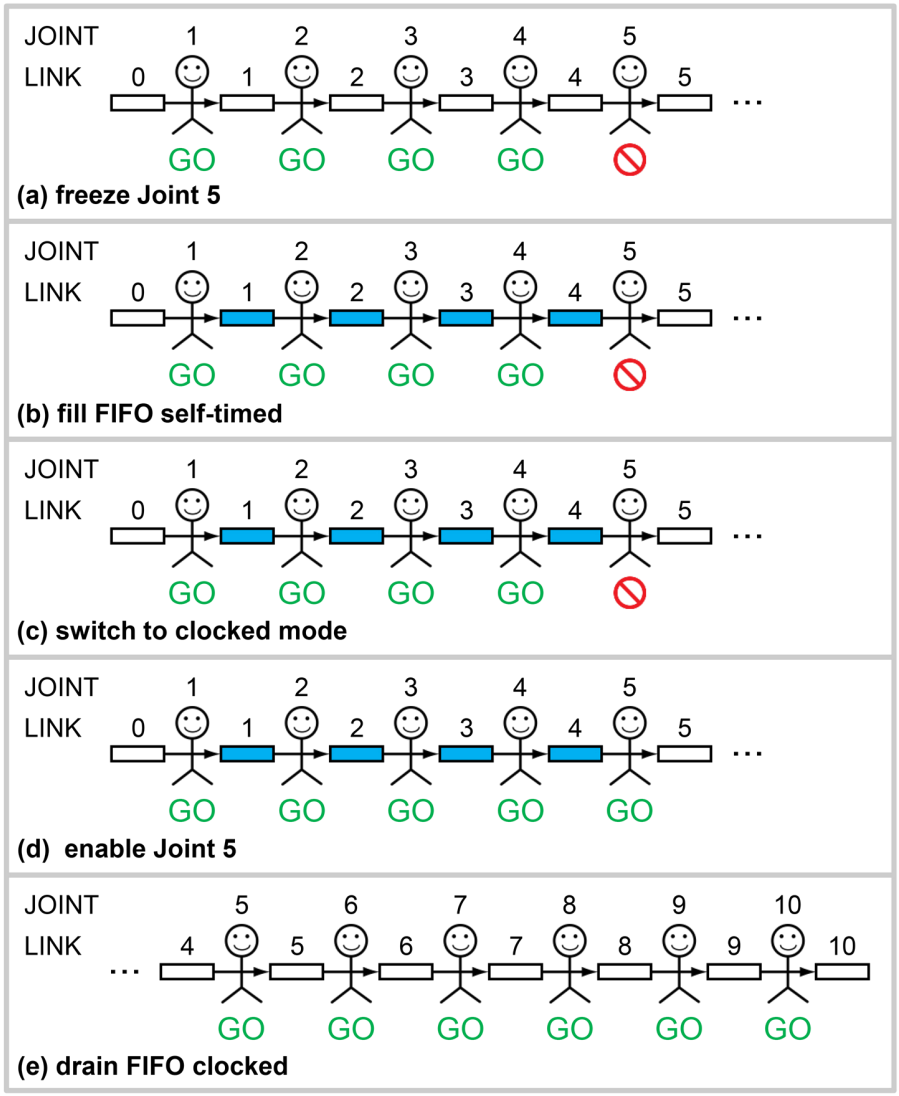


Figure 11 Pictorial view of how to switch modes from self-timed to clocked. We color full links blue (or grey) and empty links white — see also Figure 2.

the mode of operation reliably from self-timed to clocked — see Figure 11(c). Next, we enable Join 5 by making *go5* high — see Figure 11(d). To do this safely, *go5* must change from low to high during the low phase of the master clock, *clockM*. The waveforms in Figure 10 show a safe low to high transition for *go5* sufficiently in advance of the first high pulse on *clockM*, around 275 time steps into the simulation. The second and clocked half of the operation can now forward and drain the four data items from the FIFO — see Figure 11(e).

4.3 Mix Modes — Self-Timed and Clocked

The waveforms in Figure 12 show the FIFO operating simultaneously in both self-timed and clocked modes. In addition to showing waveforms for the signal names introduced earlier for Figures 8–10, Figure 12 also includes the waveforms for go control signal *go7* of Joint 7, and for status signals *empty5* and *empty6* of Link 5 and Link 6. The FIFO is partitioned into three regions that share the same source clocks but can run in different modes of operation:

- **Region 1**, the FIFO's input region, covers Link 0 to Link 4, and operates continuously in self-timed mode. Signals clockM1, clockS1 refer to its clocks.
- **Region 2**, the FIFO's *airlock*, covers Joint 5 to Joint 7. It switches mode repeatedly. Signals clockM2 and clockS2 refer to its clocks.
- **Region 3**, the FIFO's output region, covers Link 7 to Link 10, and operates continuously in clocked mode. Signals clockM3 and clockS3 refer to its clocks.

We call the middle region, Region 2, "the FIFO's airlock" because it permits status, control, and data to pass reliably between the input and output regions of the FIFO, Region 1 and Region 3, just as an airlock permits safe passage of people and objects between environments of different air pressures.

A pictorial view of how the airlock provides safe passage of status, control, and data between Region 1 and Region 3 follows in Figure 13. To operate the airlock, we deploy the two MrGO circuits in Joint 5 and Joint 7 — the two joints that separate the airlock from its neighbors. By freezing or enabling Joint 5, using go control signal *go5*, we disconnect the airlock from or engage it with its predecessor region in the FIFO, Region 1. Likewise, freezing or enabling Joint 7, via *go7*, disconnects the airlock from or engages it with its successor region in the FIFO, Region 3. The use of go control signals to accommodate the airlock operation is an extension of their use in the mode-switching operation depicted in Figure 11.

The waveform and pictorial views in Figures 12–13 relate to each other as follows:

- **Figure 13(a) — fill airlock self-timed**
  Initially, Joint 5 is enabled and Joint 7 frozen, because *go5* is high and *go7* low. This engages the airlock with Region 1 for self-timed filling. The fill operation ends with Links 0 to 6 full, around 200 time steps into Figure 12.
- **Figure 13(b) — engage airlock with clocked successor**
  The stable state of the airlock allows us to disconnect the airlock safely from its self-timed predecessor, Region 1, which we do by freezing Joint 5, i.e. by making *go5* low. Now, we can switch the airlock's mode of operation reliably from self-timed to clocked. Next, we engage the airlock with its clocked successor, Region 3, by enabling Joint 7, i.e. by making *go7* high. To do this safely, *go7* must change during the low phase of the master clock. The waveforms in Figure 12 show a safe low to high transition for *go7* sufficiently in advance of the engaging high pulse on *clockM2* or *clockM3*, now the same, around 275 time steps into the simulation.

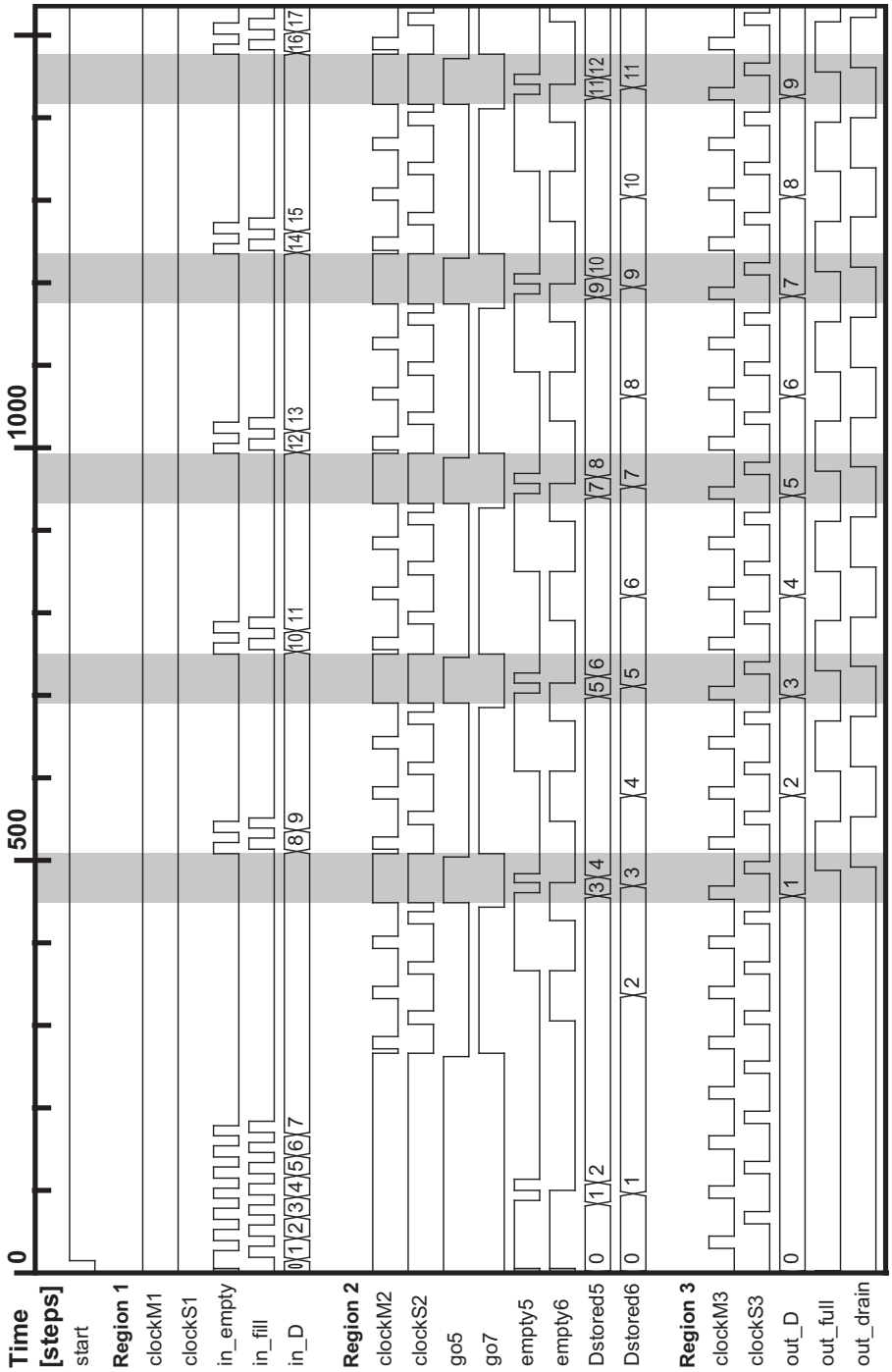


Figure 12 Mixed-mode self-timed and clocked FIFO operation with airlock.

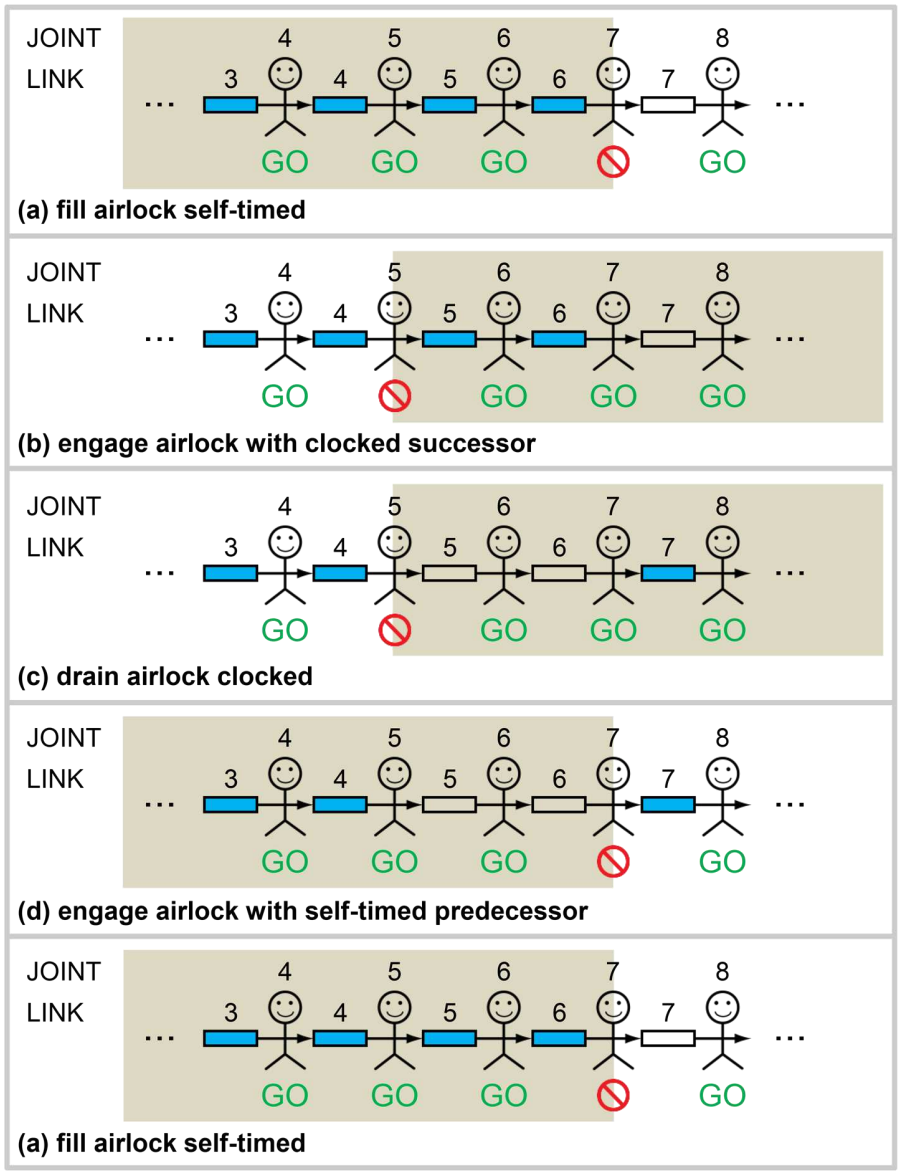


Figure 13 Pictorial view of the airlock from Joint 5 to Joint 7 passing data. We color full links blue (or grey) and empty links white — see also Figure 2.

- **Figure 13(c) — drain airlock clocked**
  We can now drain the airlock using a clocked mode of operation. The drain operation forwards data value 1 on *Dstored6*, stored in Link 6, followed by data value 2 on *Dstored5*, stored in Link 5, draining both links in the process. The drain operation ends in a stable airlock state with Link 5 and Link 6 both empty, around 450 time steps into Figure 12.
- **Figure 13(d) — engage airlock with self-timed predecessor**
  The stable airlock state allows us to disconnect the airlock safely from its successor, Region 3, which we do by freezing Joint 7, by making *go7* low. Next, we engage the airlock with its self-timed predecessor, Region 1, by making *clockM2* and *clockS2* both high to enable self-timed operation, and by making *go5* high to enable Joint 5. These engagement steps happen shortly after the airlock becomes empty, about 450 time steps into Figure 12.
- **Figure 13(a) — fill airlock self-timed**
  We can now fill the airlock using a self-timed mode of operation, just like we did initially. The fill operation ends with Link 0 to Link 6 full, and with a data value of 3 on *Dstored6* in Link 6 and a data value of 4 on *Dstored5* in Link 5 — about 500 time steps into Figure 12. This specific fill operation is marked by the first grey-colored vertical band in Figure 12. Similar grey-colored bands mark similar fill operations further along in the simulation.

Note that each grey-colored band in Figure 12 starts with *empty5* and *empty6* both high, and ends with *empty5* and *empty6* both low. This indicates that each grey-colored band starts with an empty airlock and ends with a full airlock. The airlock is filled using a self-timed mode of operation within a grey band, and is drained using a clocked mode of operation between grey bands.[6]

Note also that the data exchange rate at the output of the FIFO is constant. The output environment receives a new data value for *out_D* for every two pairs of non-overlapping *clockM3*–*clockS3* pulses, i.e. every two clock cycles. This is the fastest clock cycle time that the simulated dual citizen Click circuit can support. *The output environment works at full speed, and can be completely agnostic of the existence of the self-timed input environment and the FIFO's airlock!*

5 Comparison to Related Work

The idea of clocking a self-timed circuits is not new by itself. Prior work published in [7, 10, 3, 2] explains how to combine clocks with handshake protocols or with other forms of elastic protocols. Only the circuits published in [10] and [3] use both a clocked and a self-timed mode of operation, as do we, but neither publication discusses running a system in both modes concurrently as we do in Figures 12–13. Below follows a more specific comparison.

The work reported in [7] adds clocks to initially self-timed handshake circuits but then optimizes the circuits for synchronous operation by simplifying those

[6] Note that *clockM2* and *clockS2* remain high within a grey band, for self-timed filling, and match *clockM3* and *clockS3* between grey bands, for clocked draining.

parts of the circuits that provide flow control for self-timed operation but that are redundant under clocked operation. The resulting circuits, though generated with the same design flow, are no longer self-timed and may have lost some of their elasticity, but can be used for FPGA mappings or for integration into a completely synchronous system.

The circuits presented in [2] remain elastic when clocked, and will thus tolerate variations in computation and communication delays when clocked. The supporting design, analysis, and optimization techniques described in the paper can be used for clocked as well as for self-timed circuit designs. As such, the choice "to clock or not to clock" can be deferred until late in the design process. The paper gives no examples nor any indication of keeping both choices, clocked and self-timed, available to the final circuit implementation.

The dual-mode synchronous/asynchronous CORDIC processor for wireless broadband communication presented in [3] can select its mode of operation to fit system demands and application needs. For instance, when the received signal is weak, the processor can be switched into self-timed mode to reduce electro-magnetic interference. The clocks in [3] bypass the handshake control circuits. Consequently, clocked operation of the CORDIC forfeits the elasticity provided by the handshake protocols. Also, in bypassing the handshake control, the CORDIC's clocked mode of operation will be of marginal use for building confidence in the CORDIC's self-timed operations.

In contrast to [3], the clocked or synchronous mode of operation implemented in [10] re-uses the self-timed fabric and protocols — as do we. The resulting level-sensitive synchronous mode of operation thus inherits the elasticity of the self-timed mode of operation. As a result, the synchronous mode of operation can be used to build confidence in the self-timed circuit operations, which is one of the key reasons for us to add it, though this is not addressed in [10] which mentions only its potential use for system-level diagnosis and debug. The paper provides a systematic solution for adding clocks and test inputs to a self-timed circuit. The use of clocks to run the circuit in synchronous mode acts as a stepping stone in that solution. The key feature of [10] is the systematic addition of a clocked scan test mode of operation.

The dual citizen circuits that we present in this paper offer a new approach to clocking self-timed circuits, because the circuits are built around the ideas of naturalized communication and testing [8]. By differentiating *links* from *joints* the design solutions for adding a clocked mode of operation fall naturally into solutions that clock the links versus solutions that clock the joints. By differentiating *actions* from *states* we can avoid adding energy-costly slave latches into the datapath that an action-agnostic approach like [10] would add, because we know that the joints at opposite ends of a link act in mutual exclusion. Last but not least, we can control actions individually, using MrGO. By enabling or freezing selective actions at run time, different parts of the circuit can be made to (1) run in different modes, (2) switch modes reliably, and (3) exchange data without the need for synchronizers.

6 Conclusion

The "naturalized communication and testing" view [8] unifies thinking about a wide variety of self-timed circuit families and facilitates mixing and matching these families within a single system. In this paper, we extend this unity to embrace clocked circuits; we add a clocked or synchronous mode of operation to self-timed circuits. We call the resulting circuits *dual citizen* circuits.

As reference circuit, we chose a ripple FIFO implemented in Click [6] but adapted for naturalized communication and testing [8]. Its dual citizen solutions and simulation results apply broadly to other self-timed designs and circuit families.

Clock signals that retard self-timed operation can be part of *links* or of *joints*. Links store and transport data. Joints act on the data. We have shown how to add clocking to each. Each of our clocking additions re-uses the self-timed protocols, and thereby inherits their elasticity to act only when and where needed. Need-driven action is beneficial not only because it saves energy, but also because it simplifies scheduling of operations. With protocols rather than clock cycles in charge of flow control, the clocked operations of a dual citizen circuit can function correctly even when operating out of lockstep.

By recognizing joint *actions*, we can avoid adding latches and clocked gates into the datapath As a result, dual citizen circuits operating in self-timed mode can maintain the energy-efficiency of the original self-timed circuit.

By enabling or freezing selective actions at run time, using MrGO, different parts of the circuit can (1) run in different modes, (2) switch modes reliably, and (3) exchange data without the need for synchronizers. We have shown simulations of fixed mode operation, mode switching, and mixed mode operation.

The clocked mode of dual citizen circuits can bolster confidence in the correct functionality of the self-timed mode — or vice versa — in various ways.

- Engineers most comfortable with clocked systems can easily see how dual citizen circuits work when clocked. The datapath is identical in both modes and so may be understood in either mode. The self-timed control fabric and protocols are shared in both modes. Seeing the control work when clocked can therefore build confidence in its self-timed behavior.
- Simulations reported here exhibit mixed mode behavior. Data received in self-timed mode may be delivered to a clocked destination and vice versa. The ability to change between clocked and self-timed modes of operation can bolster confidence in the correctness of either mode.

Because their self-timed mode of operation is faster than their clocked mode, the clocked mode of operation can be used as a "crutch" to support aging or erratic self-timed circuit operations that need more time to finish. The link-based clocking addition makes a good crutch because it keeps a firm grip on self-timed loops and can retard any of these as much as needed by using wider clock pulses. On the other hand, the self-timed mode of operation can provide a "turbo" performance boost when needed, to obtain better latency, throughput, energy, robustness to delay variations, or electro-magnetic compatibility.

Moving safely from clocked circuits through self-timed circuits and then back again provides a path for synchronous designers to embrace self-timed design incrementally. We clear this path by providing self-timed circuits with a clocked mode of operation. This approach deserves thorough study, so the costs in terms of design, analysis, verification, and engineering can be quantified, and — we hope — proven competitive with the state of the art in distributed VLSI design.

Acknowledgement

We thank corporate and private sponsors of the Asynchronous Research Center at Portland State University. We also thank the National Natural Science Foundation of China for sponsoring part of this work under Grant No. 61402121. Last but not least, we thank our friend and colleague Alex Yakovlev, for his unforgettable "Terminator" accent — now either lost or too familiar to us — and for the many years of camaraderie, open-mindedness, and fresh ideas and students he carried with him to and from Newcastle. To strengthen our ties, this paper ends with a family recipe for Pavlova from the New Zealand branch of the Sutherland clan. Face shots are in order of authors. Happy birthday, Alex!

References

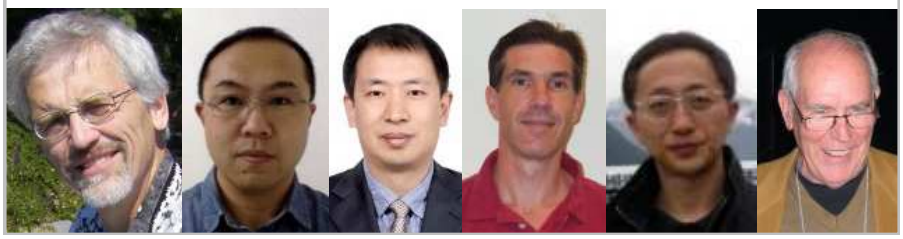
1. Peter Beerel and Marly Roncken. Low Power and Energy Efficient Asynchronous Design. *Journal of Low Power Electronics (JOLPE)*, 3(3):234–253, 2007.
2. Joseph Carmona, Jordi Cortadella, Mike Kishinevsky, and Alexander Taubin. Elastic Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(10):1437–1455, 2009.
3. Eckhard Grass, Bodhisatya Sarker, and Koushik Maharatna. A Dual-Mode Synchronous/Asynchronous CORDIC Processor. In *Asynchronous Circuits and Systems (ASYNC)*, pages 76–83, 2002.
4. Steven Nowick and Montek Singh. Asynchronous Design — Part 1: Overview and Recent Advances. *IEEE Design & Test*, 32(3):5–18, 2015.
5. Steven Nowick and Montek Singh. Asynchronous Design — Part 2: Systems and Methodologies. *IEEE Design & Test*, 32(3):19–28, 2015.
6. Ad Peeters, Frank te Beest, Mark de Wit, and Willem Mallon. Click Elements: An Implementation Style for Data-Driven Compilation. In *Asynchronous Circuits and Systems (ASYNC)*, pages 3–14, 2010.
7. Ad Peeters and Kees van Berkel. Synchronous Handshake Circuits. In *Asynchronous Circuits and Systems (ASYNC)*, pages 86–95, 2001.
8. Marly Roncken, Swetha Mettala Gilla, Hoon Park, Navaneeth Jamadagni, Chris Cowan, and Ivan Sutherland. Naturalized Communication and Testing. In *Asynchronous Circuits and Systems (ASYNC)*, pages 77–84, 2015.
9. Jens Sparsø and Steve Furber (Eds.). *Principles of Asynchronous Circuit Design — A Systems Perspective*. Kluwer Academic Publishers, 2001.
10. Kees van Berkel, Ad Peeters, and Frank te Beest. Adding Synchronous and LSSD Modes to Asynchronous Circuits. In *Asynchronous Circuits and Systems (ASYNC)*, pages 161–170, 2002.

Pavlova photo by Hazel Fowler, Wikimedia Commons

Pavlova

3 (old) egg whites
9 oz castor sugar
1 tsp vanilla
1 tsp vinegar
1 pinch salt
cream
fruit

Beat egg whites with a pinch of salt
until stiff enough to peak.
Fold in sugar, vanilla, and vinegar.
Place on baking paper on greased tray.
Bake slowly about 1-1.5 hours at 250F.
Dress with fresh whipped cream,
kiwi fruit, strawberries, or similar.
BON APPÉTIT !



Significance-Driven Computing for Big Data Applications: From *Buttery* Discussions to *Serious* Research

Rishad Shafik
Microsystems Research Group, School of Electrical and Electronic Engineering
Newcastle University, Newcastle upon Tyne, NE1 7RU
e-mail: Rishad.shafik@ncl.ac.uk

Abstract. Sometimes the best of the ideas evolve from informal discussions over coffee in cafes or places of retreat, such as *Buttery*. In this festschrift article we reflect on how one of these ideas inspired serious research on the development of a new generation of intelligent and energy-efficient processors by the Microsystems Research Group (jointly led by Alex and I) at Newcastle University.

1 Pretext

Continued technology scaling and engineering innovations have made digital services ever more affordable, thereby revolutionising the industrial age of data. A number of applications have emerged, which use deeply-embedded sensors to collect data and process them continuously, otherwise known as big data. Examples of these applications include smart computer vision, machine learning, e-governance and financial analytics. However, with widespread adoption of these services and applications the dimensionality and density of data are increasing drastically, rendering an unprecedented resource proliferation. Such proliferation of resources is likely to cause an uncontrolled energy consumption challenge for computing hardware systems, particularly to technology and service providers. For achieving transformational energy efficiency while also coping with increased performance needs a key solution is to design adaptive approximate computing systems. Recently, extensive research efforts have been initiated by the Microsystems Research Group (jointly led by Alex and I) at Newcastle University to design such computing systems. The aim is to learn the data significance during runtime and to process them dynamically adapting to their significance using novel logic design and system-level interactions. This rest of this article will report how these ideas evolved from informal discussions over coffee at Buttery and how they have inspired serious research on the design, implementation and validation of a new generation of intelligent and energy-efficient processors.

2 Prologue: *Buttery* Discussions

Buttery is a retreat space, popular among the students and staff members of the School of Electrical and Electronic Engineering (EEE) at Newcastle University. When I joined the school as a Lecturer in Electronic Systems (in September 2015), this was the first place where Alex invited me to have a coffee with him and discuss my research ambitions. During our rather informal discussions, the words 'adaptive', and 'approximate' were pronounced in very many contexts, mostly within the remits of electronic computing systems. Our discussions made good strides in understanding how the recent computing systems have evolved and how the future systems needed to emerge with magical adaptability features to play the perfect tradeoff game between energy consumption and performance every time.

In *Buttery* we started a little tradition of round-robin payment system for the coffee there – e.g. if Alex paid for the coffee the day earlier day I'd be the one to pay the next day and so on. Many a times, we would not remember who paid in the previous day. In those times we would vaguely try to remember by tagging with some 'significant' ideas that have been discussed in the previous day. We could then find out who paid for the coffee through backtracing techniques, much like *GCC's debugging features*. We both appreciated how our brains tag 'significance' to our everyday activities and process them accordingly. We also recognised how more 'significant' activities are stored by our brain in fast access memory for us to quickly recapitulate them.

Strangely enough, these little appreciations and observations triggered us to think and ask – 'why shouldn't our computing systems and storage subsystems also work likewise – i.e. modulate and adapt computation and storage efforts based on the data significance?' Finding "the" answer meant more work and investigation as outlined in the following sections.

3 Parode: Energy Consumption and Performance Tradeoffs

To find "the" answer it was important to understand the underlying challenges with existing computing systems. The increasing performance demands and resulting energy were the first issues that needed a bit of retrospective analysis without getting into details of big data just yet.

The performance needs of modern embedded computing systems have evolved dramatically over the years due to many emerging applications. According to Koomey's law, the performance per unit watt has doubled every 1.57 years [1], which is faster than the originally predicted 1.8 years by Moore's law [2]. The performance improvement is being enabled by technology scaling and innovative parallelisation techniques. However, the power consumption is also increasing uncontrollably as demonstrated by Dennard's scaling law [3] and numerous experimental observations [4-5]. Indeed, achieving scalable performance improvement with energy-efficiency is highly challenging for current and future generations of computing systems.

To minimise energy consumption, traditional approach is to reduce the supply voltage [6]. However, due to capacitive load imbalance, this also necessitates

lowering the operating frequencies [7]. Such reduction of supply voltage, coupled with the operating frequency is generally known as dynamic voltage/frequency scaling (DVFS), which ensures energy minimisation at the cost of degraded performance [6]. Over the years, significant research works have been carried out to demonstrate energy and performance tradeoffs in computing systems [8-11].

To improve performance at low energy consumption, an effective approach is to operate each processor core at low voltage/frequencies and also parallelise the computation tasks between multiple cores [12]. Significant research has been carried out in the recent past for understanding the best possible schemes and architectures to parallelise application tasks, including both compute- and memory-intensive ones. These works have revealed that the best energy efficiency is exhibited when compute-intensive parallel application tasks are exercised with higher number of cores operating at low voltage/frequency. The memory-intensive parallel workloads tend to favour lower number of cores at higher operating voltage/frequency for energy efficiency [13].

However, modern application workloads cannot always be statically labelled as CPU- or memory-intensive. Workloads change dynamically in these applications all the time. Often the same task or parallel thread can exhibit compute- and memory-intensive contexts at different times [13]. To address such dynamic variations, continuous runtime adaptation approaches have also been demonstrated recently by researchers. These approaches use feedback from the processor performance counters to adjust number of parallel threads, cores, architectural configurations and/or operating voltage/frequency to achieve energy-efficiency [9-10].

Existing system-level approaches have established the relationships between application workloads and power control knobs to achieve energy efficiency. However, modern big data applications are posing new challenges energy efficiency at required performance levels. This is because these applications are typically characterised by high volume and velocity (i.e. real-time processing needs) of data, which will require unprecedented resource allocations (for both computation and storage) using the existing approaches [14].

It is clear that existing computing approaches will hardly be enough to meet the growing performance needs for big data applications. Specific details of this are highlighted next.

4 Agôn: Key Challenges of Big Data Computing

Achieving energy efficiency for computing with big data applications is highly challenging using the existing approaches due to the following three major reasons:

Data proliferation: Existing big data applications have been characterised with a volume growth of several hundreds of petabytes per day. It is envisioned that such expansive growth will continue for the foreseeable future, generating many orders of magnitude higher volume of data. Current research suggests that the typical energy consumption of computing these data will soon approach the complexity of $O(N^3)$ or higher, where N is the number of data samples [15].

Undue performance scaling: To compute such a large volume of data at the required performance, currently existing computing systems exploit system-level controls, e.g. increased number of parallel cores and high operating frequencies through DVFS. However, these controls eventually cause diminishing returns in terms of increased energy consumption and complexity in the systems design with large area. In some cases, to meet the high performance demands custom designed accelerators are also used, which less flexible in terms of design, adaptability and programmability.
Indiscriminate Data Processing: The raw data of these applications acquired from the sources (e.g. sensors or humans) are processed identically in existing computing systems, ignoring the underlying informational value, i.e. significance, of the data. However, in reality the significance of acquired data varies dynamically over time and space depending on the application [16]. As a result, existing computing systems exhibit indiscriminate efficiency in data processing, resulting in large energy costs.

To foster the growth of this technology with the required energy reductions a paradigm shift is much needed from the existing significance-agnostic computing to significance-driven computing. Our research efforts in this direction is briefly outlined in the next section.

5 Parabasis: Significance-Driven Computation Research

Recently, *extensive* and *serious* research works have been initiated by the Microsystems Research Group at Newcastle University, led by Alex and I, to design such computing systems, including logic-level and system-level approaches. Our works at these levels are aimed at achieving holistic energy-efficiency through adaptive computation approach that can intelligently infer the significance of underlying information (i.e. bits and/or data). We give a brief account of the summary of our research to date in the following key areas, as follows.

5.1. Significance-Driven Low-level Logic Design:

In existing data processing logic design, there is no notion of modulating processing effort based on their bit-level significance. All bits are treated equally to generate a precise output. However, many emerging applications are inherently tolerant to imprecisions in less signigicant bits, such as computer vision, data mining and machine learning. This gives a unique opportunity to design next-generation processing logic such that computation efforts can adapted to the significance at bit-levels for achieving energy efficiency.

To this end, our low-level logic design is aimed at developing novel arithmetic and logical data processing subsystems. The more significant bits are treated with progressively higher precision through traditional computation, while bits with lower significance are compressed using variable clustering (i.e. vertical grouping of partial terms). As a result of such logic design, complexity of computation in terms of logic cell counts and length of the critical paths are drastically reduced.

A number of multipliers using this approach have recently been designed using SystemVerilog and synthesised using EDA tools. Our post-synthesis experiments with a 128-bit multiplier showed that up to 60% less energy consumption and 53%

performance improvement can be achieved, when compared with traditional Dadda and Wallace multipliers. These gains are achieved at a low loss of accuracy due to significance-driven bit processing - with up to 30% inaccuracies for small valued operands and exponentially reduced imprecision for higher values. We are currently designing real application demonstrators using this approach to show the comparative advantages of our approach.

5.2. Extracting Data Significance at System-level:

Data are crucial parts of big data applications. However, not all data carry the same informational value. Traditional computing systems are agnostic of such values as all data are processed indiscriminately and equally. As a result, a large energy cost is incurred for processing a large volume of data, much of which have little or no significance.

To extract the informational value of data at operational time we have taken initiatives to modify processor architectures underpinning the theory and practices of approximate computing and machine learning. These involve designing a data inference engine as middleware, which will use application domain-specific knowledge to evaluate measurable significance of data that are being processed in parallel. The aim is to use the measured significance to dynamically scale the computation efforts, e.g. data with higher significance will be processed using accurate data processing unit and standard data flow (prefetch, decode and execute), while those with less significance will be processed using low-complexity and inaccurate data processing unit using already existing cache data (i.e. avoiding further prefetch flow). This will eventually result in significant energy reductions for data-intensive applications.

To date we have already carried out proof-of-concept designs, currently also carrying out implementation-ready logic design. We will follow this by integrated processor design and optimisation, including cache localisation for fast middleware routines and DVFS features to corroborate energy reductions. The new processor will be validated using real case study big data applications for practical demonstrations to key industries and academia in the UK and beyond. The overarching goal will be to create a critical mass in this important area.

5.3. Significance-driven Big Data Storage:

Memory constitutes a major component in modern computing systems. The energy efficiency or performance of these systems cannot be achieved in isolation without considering the memory effect. As such, our future research plan will include significance-driven memory management, including cache optimisation and development of fast memory systems for significant data.

6 Exode: Conclusions

Our rather informal *Buttery* discussions over such a short period time have given us the impetus to combine our expertise synergistically to carry out *serious* research on a new breed of intelligent processors. So this *Exode* is really just a beginning rather

than an end. We will continue to engage in further fruitful discussions in the future, potentially involving other interested academics and industrial peers to advance the understanding of research needs further.

We expect that our research will be a small but important contribution to UK's world-leading portfolio in low power systems engineering. A key differentiator for maintaining this portfolio and enhancing it further will be to be able to design a new breed of intelligent processors with control over computation efforts. It is a relatively new area, which combines and advances the theory and practices of traditional approximate computing and machine learning. Our innovations in this space will be crucial to enable many emerging applications of profound impact on our businesses and society.

References

1. Koomey, Jonathan; Berard, Stephen; Sanchez, Marla; Wong, Henry; (March, 2010). Implications of Historical Trends in the Electrical Efficiency of Computing. IEEE Annals of the History of Computing 33 (3): 46−54..
2. Moore, Gordon E. (April 1965). "Cramming more components onto integrated circuits". Electronics 38 (8): 1-4.
3. Dennard, Robert H.; Gaensslen, Fritz; Yu, Hwa-Nien; Rideout, Leo; Bassous, Ernest; LeBlanc, Andre (October 1974). Design of ion-implanted MOSFET's with very small physical dimensions. IEEE Journal of Solid State Circuits SC–9 (5): 668-678.
4. Schaller, R. R. (1997). Moore's law: past, present and future. IEEE spectrum, 34(6), 52-59.
5. Moore, G. E. (1995, May). Lithography and the future of Moore's law. In SPIE's 1995 Symposium on Microlithography (pp. 2-17). International Society for Optics and Photonics.
6. Shafik, R. A., Al-Hashimi, B. M., & Chakrabarty, K. (2010, March). Soft error-aware design optimization of low power and time-constrained embedded systems. In Proceedings of the Conference on Design, Automation and Test in Europe (pp. 1462-1467). European Design and Automation Association.
7. Wang, L., Von Laszewski, G., Dayal, J., & Wang, F. (2010, May). Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS. In Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on (pp. 368-377). IEEE.
8. Shafik, R. A., Al-Hashimi, B. M., Kundu, S., & Ejlali, A. (2009). Soft Error-Aware Voltage Scaling Technique for Power Minimization in Application-Specific Multiprocessor System-on-Chip. Journal of Low Power Electronics, 5(2), 145-156.
9. Das, A., Kumar, A., Veeravalli, B., Shafik, R., Merrett, G., & Al-Hashimi, B. (2015, March). Workload uncertainty characterization and adaptive frequency scaling for energy minimization of embedded systems. In Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (pp. 43-48). EDA Consortium.
10. Shafik, R. A., Yang, S., Das, A., Maeda-Nunez, L. A., Merrett, G. V., & Al-Hashimi, B. M. (2016). Learning transfer-based adaptive energy minimization in embedded systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 35(6), 877-890.
11. Kim, S. G., Eom, H., Yeom, H. Y., & Min, S. L. (2014). Energy-centric DVFS controlling method for multi-core platforms. Computing, 96(12), 1163-1177.
12. Shafik, R. A., Das, A., Yang, S., Merrett, G., & Al-Hashimi, B. M. (2015, January). Adaptive energy minimization of OpenMP parallel applications on many-core systems. In

Proceedings of the 6th Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures (pp. 19-24). ACM.

13. A Aalsaud, R Shafik, A Rafiev, F Xia, S Yang & A Yakovlev (September 2016). Power-Aware Performance Adaptation of Concurrent Applications in Heterogeneous Many-Core Systems. In Proceedings of International Symposium on Low Power Electronics and Design (ISLPED) (in press). IEEE.
14. Burke, D., Shafik, R., and Yakovlev, A. (March 2016). Challenges and Opportunities in Research and Education of Heterogeneous Many-Core Applications, European Symposium on Microelectronics Education (EWME) (in press).
15. Kambatla, K., Kollias, G., Kumar, V., & Grama, A. (2014). Trends in big data analytics. Journal of Parallel and Distributed Computing, 74(7), 2561-2573.
16. Mohapatra, D., Karakonstantis, G., & Roy, K. (2009, August). Significance driven computation: a voltage-scalable, variation-aware, quality-tuning motion estimator. In Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design (pp. 195-200). ACM.

Workcraft: Ten Years Later

Danil Sokolov, Victor Khomenko, Andrey Mokhov

Newcastle University, Newcastle upon Tyne, UK

Abstract A large number of models that are employed in the field of concurrent systems' design, such as Petri nets, Signal Transition Graphs, gate-level circuits, dataflow structures have an underlying static graph structure. Their semantics, however, is defined using additional entities, e.g. tokens or node/arc states, which collectively form the overall state of the system. We jointly refer to such formalisms as *Interpreted Graph Models* (IGMs).
Workcraft is a framework for capturing, simulation, synthesis and analysis of IGMs. It provides an extendible cross-platform plugin-based front-end to a variety of computationally-intensive command-line back-end tools. This paper gives the developers' perspective on Workcraft and overviews its evolution.

1 Introduction

We want our research to be used to make the world a better place. However, technology transfer is challenging in practice due to a number of obstacles. One of the primary research outcomes is scientific publications. However, engineers do not have time to read them and do not normally have a necessary background to comprehend and apply that knowledge, or even find the necessary publications.

One of the ways to make this knowledge more accessible is to encapsulate it in software tools. Ideally, these tools should be usable by non-experts. In practice, however, the situation is very different. A research tool is typically developed up to a point when it can produce a table of results for a research paper. The motivation to develop it any further is diminished afterwards – in the current "publish or perish" academic culture it is usually more advantageous to start exploring new topics for writing another paper, and it is also more interesting than polishing old tools. Moreover, research funding is usually granted for a very limited period of time and so the academics, researchers, and PhD students who developed the tool leave or get allocated to different projects. Hence, much of research software has only a command-line interface with cryptic options, poor documentation, limited error handling, requires modifying the source code to adjust the tool to a particular variant of a problem and is not maintained.

Integrating several research tools into a coherent flow presents further challenges. File formats are often invented by the creators of the tools and are non-standard. Furthermore, there are often gaps in the flow that have to be patched to complete it. This requires a large amount of slog with no perspective of publishing its results. Therefore, using research software in a real industrial flow is

often infeasible. The net result of the above is that much knowledge remains buried in publications or "experts-only" tools and is not accessible to practitioners.

Workcraft is unusual in several respects. There have been a series of funded projects related by the common topic of application of Petri nets to circuit design. This allowed to have relatively stable group of developers and sufficient time to make the tool usable. Furthermore, early versions of the tool attracted industrial interest, which motivated putting more effort into user interface development. In addition, several previously developed command-line tools were deployed within Workcraft as back-ends. In turn, the success of Workcraft and the perspectives of industrial exploitation motivated the developers of back-ends to maintain and enhance the functionality of these tools.

As the result, Workcraft opens access to the goodness hidden in research tools. The main enabling factors for this to happen are:

- *Availability* – open-source front-end and plugins, permissive freeware licenses for back-end tools, as well as frequent releases with bug fixes and features requested by users.
- *Usability* – elaborated GUI that was developed with much feedback from the users.
- *Portability* – it runs on Windows, Linux, and Mac OS X operating systems.
- *Extendibility* – the framework is designed to easily include new IGMs and interfaces to back-end tools as plugins.
- *Automation* – several complete design flows have been implemented by bridging the gaps between back-ends and converting file formats.

The focus on availability, usability, portability and extendibility, along with extensive networking, has proven effective – there is a large and diverse user base. For example, in 2015 there were 4.4k downloads from 1.2k unique IPs and 11.1k visits to `http://workcraft.org/`, 4.7k of them from unique IPs. During that year there were 4 releases with 49 bug fixes and 23 new features. We do not know all Workcraft users, but one can identify at least the following categories: developers, industrial users, undergraduate students, academics, researchers, and PhD students.

In this paper we give the developers' perspective on Workcraft. The main principles of Workcraft architecture and its design flow are outlined in Sections 2 and 3. Supported IGMs together with relevant case studies use are presented in Sections 4 and 5. We analyse the categories of Workcraft users in Section 6 and overview the timeline of Workcraft evolution in Section 7.

2 Workcraft philosophy

In this section we discuss several basic principles/ideas underlying Workcraft. Some of them are visible to the user and aimed at enhancing the user experience. Others are concerned with the internal organisation and aimed at simplifying the integration of new research tools.

2.1 Interpreted Graph Models

A large number of models that are employed in the field of concurrent systems' design, such as Petri nets, Signal Transition Graphs, gate-level circuits, dataflow structures have an underlying static graph structure. Their semantics, however, is defined using additional entities, e.g. tokens or node/arc states, which collectively form the overall state of the system. We jointly refer to such formalisms as *Interpreted Graph Models* (IGMs) [21].

The similarities between the interpreted graph models allow for links between different formalisms to be created, either by means of adapter interfaces or by conversion from one model type into another. This greatly extends the range of applicable modelling and analysis techniques.

Workcraft is designed to provide a flexible common framework for development of interpreted graph models, including visual editing, (co-)simulation and analysis. The latter can be carried out either directly or by mapping a model into a behaviourally equivalent model of a different type (usually a Petri net or Signal Transition Graph). Hence the user can design a system using the most appropriate formalism (or even different formalisms for the subsystems), while still utilising the power of Petri net analysis techniques. In Section 4 there is a summary of the currently supported IGMs.

2.2 Front-end vs. back-end

Workcraft provides front-end to a number of command-line back-end tools, such as Petrify [10,2] and UnfoldingTools toolkit [3]. The calls to back-end tools are transparent to the user: Workcraft automatically chooses the correct command-line parameters, parses the output of the tools and presents it to the user in an appropriate graphic form. For example, to check whether a digital circuit conforms to its environment the user needs to click a single menu item. In response the following sequence of actions is performed by the front-end:

1. The circuit is converted to an equivalent STG.
2. The internal signal transitions in the environment STG (it models the contract between the circuit and its environment) are replaced by dummies – this is required for technical reason.
3. The STGs obtained in the previous two steps are composed by calling PComp back-end with appropriate command line parameters.
4. The front-end expresses the conformation property as an expression in Reach language. Parts of this expression are specific to the circuit under test and need to be calculated by the front-end.
5. The composed STG is unfolded by calling Punf back-end.
6. The resulting unfolding prefix and Reach expression are passed to MPSat back-end that performs verification.
7. The verification results are parsed by the front-end. If the property holds then an appropriate message is displayed. Otherwise the violation trace of the composed STG reported by MPSat is projected to the circuit, and the

user can execute it step-by-step to debug the problem. All the capabilities of the front-end simulator are available, e.g. navigation within the trace, branching, etc.

Each of the above steps looks trivial and "boring" (and hence unpublishable) from the research point of view. However, checking conformation is a frequent task during the circuit design. Performing all the above steps manually would have been very tedious and error-prone, discouraging the casual user from applying formal verification. Hence using WORKCRAFT makes it feasible for the user to harness the power of research tools, which helps to catch the bugs early in the design process, and reduces the risk of an incorrect circuit going into production.

2.3 Plugin-based architecture

Extendibility is an important part of WORKCRAFT philosophy and this is reflected in its plugin-based architecture [23]. In particular, there is a framework for adding new IGMs and integrating new back-end tools. This framework provides a number of standard services available to the plugins, such as (de-)serialisation of IGMs, common editing features (creation of nodes and connections, undo-redo, copy-paste, etc.) and model visualisation.

To add a new IGM the developer has to implement a small set of Java interfaces for mathematical and visual representation of the IGM. Most of the functionality has default implementations provided by the WORKCRAFT core and only "unusual" features of a new model need to be explicitly implemented.

For integration of a back-end tool the developer needs to provide versions of the tool for the supported operating systems (this is usually not a problem because console applications are relatively easy to port), and implement a simple Java interface that specifies how to run the tool, interpret its output, which IGMs it is applicable to, and which menu to integrate it into.

There is also a possibility to add more complicated plugins that interact with visual representation of the IGMs, e.g. the simulation plugin.

3 WORKCRAFT design flow

The WORKCRAFT design flow is modelled by the Petri net in Figure 1. A typical way of designing a circuit is as follows.

1. The STG specification (place specification) is created in the WORKCRAFT editor (transition edit) or perhaps imported from a *.g file (transition import).
2. The user verifies various properties of this specification, such as consistency, deadlock freeness, output persistency, input properness, complete state coding (CSC) and some custom design specific properties (transition verify).
3. The verification report from a back-end tool (place report) is then presented to the user in a convenient form, e.g. a violation trace can be simulated (transition simulate), CSC conflict cores can be visualised as a core map or a core density map (transition visualise). This helps the user to debug the STG.

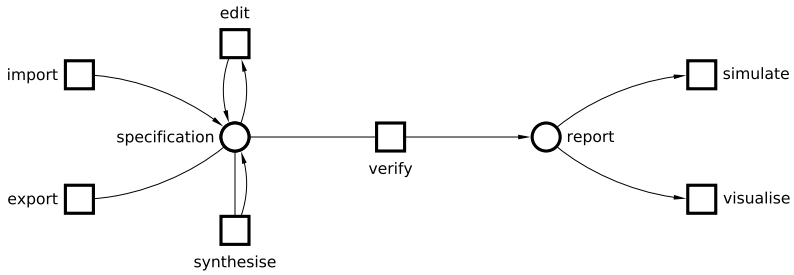


Figure 1: WORKCRAFT design flow.

4. Once a correct specification is obtained, it can be implemented as a circuit. At this point the CSC property may still be violated, and so a new STG where the conflicts are resolved by inserting new internal signals can be automatically created by synthesis back-ends (transition synthesise; hence place specification will contain two tokens representing the original and modified STGs).
5. At this point the user can synthesise a digital circuit (transition synthesise; hence place specification will contain three tokens representing the two STGs and the circuit).
6. The user can manually alter the circuit, e.g. by improving the layout or changing the polarity of some internal signals (transition edit).
7. The circuit must be verified against the initial specification, as synthesis tools are complicated and may have bugs and manual editing is error-prone (transition verify). Verification report is presented to the user in a convenient form.
8. The created models can be exported, e.g. as Verilog netlist for circuits and *.g files for STGs (transition export). In addition models can be exported in a variety of graphic formats for inserting into documentation or research papers.

4 WORKCRAFT models

As explained in Section 2, WORKCRAFT supports several IGMs, and new models are added from time to time. Some of the most popular IGMs are described below. A crucial aspect of WORKCRAFT is interaction and synergy between different types of IGMs.

A popular formalism for capturing the behaviour of a concurrent system is *Finite State Machines (FSMs)*. The advantage of FSMs is their relative simplicity compared to the alternative formalisms. However, they represent concurrency by multi-dimensional interleaving 'diamonds' which is unnatural and leads to exponential blow-up in the size of the model [27].

Petri nets (PNs) [19] are a well-known 'true concurrency' formalism which is much more convenient for practical modelling. WORKCRAFT supports conversions between FSMs and PNs: one can construct the reachability graph of a

PN or, vice versa, synthesise a PN as a compact representation of a behaviour expressed as an FSM – see Vending Machine case study in Section 5.1 for FSM and PN.

Signal Transition Graphs (STGs) [9,24] are a kind of PNs where transitions are labelled by rising and falling edges of signals. STGs are often used to specify the behaviour of speed-independent *Digital Circuits* [18,12], which is another IGM supported by WORKCRAFT. Many kinds of interactions and conversions between these two models are supported. For example, one can synthesise an STG as a circuit using several implementation styles or convert a circuit to an STG – this is necessary for composing it with the environment expressed as an STG for subsequent verification of various standard and custom correctness properties [22]. Section 5.2 presents a case study on designing a speed-independent circuit using STGs.

Dataflow Structures (DFSs) [25] and *xMAS Circuits* [8] are high-level models for designing pipelines, in particular data paths of circuits. WORKCRAFT supports the simulation and analysis of these IGMs by converting them to STGs and utilising the established functionality. Some model-specific functionality such as finding bottlenecks, cycle analysis, and performance optimisation using *wagging* [7] are also supported. Section 5.3 showcases DFS functionality using a baseband transmitter pipeline.

Conditional Partial Order Graphs (CPOGs) [15] is a formalism for specifying a collection of behavioural scenarios, and combining them into a compact graph representation using the optimal encoding. For example, CPOGs can be used for synthesis of application-specific microprocessor instruction sets, see Section 5.4 for an ARM Cortex-M0 case study.

Structured Occurrence Nets (SONs) [14] is a model for capturing and analysis of causality and concurrency in families of execution traces. They can be used to represent the current state of crime or accident investigation – see Section 5.5 for the use of SONs to model Ladbroke Grove rail crash.

The diagram in Figure 2 shows the relationships between the currently supported IGMs. There are several categories of automatic conversions. *Synthesis*, e.g. from STGs to Digital Circuits or from FSMs to PNs, is a computationally intensive procedure whose resulting graph is structurally very different from the input graph. *Translation* is a relatively simple transformation yielding a structurally similar graph. *Lossless* translation, e.g. from PNs to STGs, does not lose information, i.e. the original model can be restored from the result of the conversion. *Lossy* translation, e.g. from STGs to PNs, loses some information, in this case the signal information attached to transitions.

5 Case studies

The applications of WORKCRAFT are wide-ranging: from modelling concurrent algorithms and biological systems to designing asynchronous circuits and investigating crimes. In this section we present several examples of how WORKCRAFT can be used.

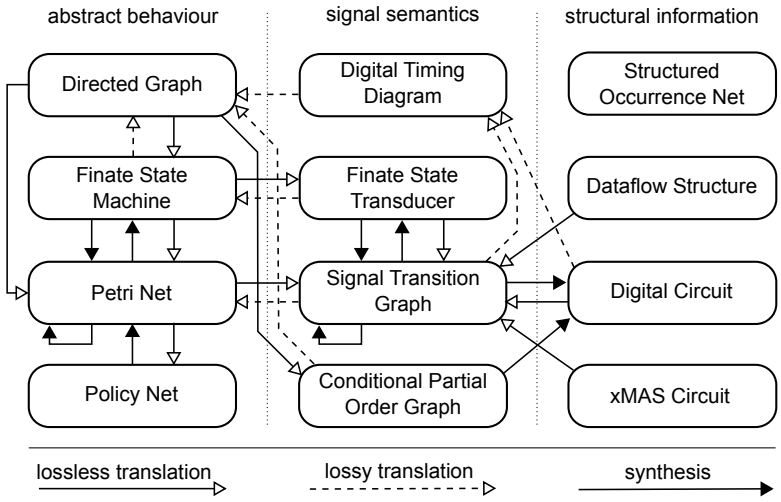
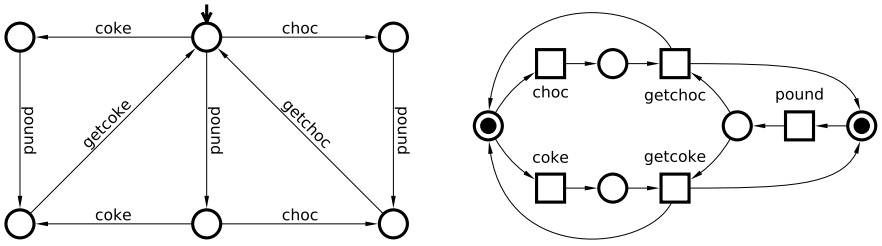


Figure 2: Relationships between WORKCRAFT models.

5.1 Modelling Concurrent Systems: Vending Machine

In this case study WORKCRAFT is used to capture the behaviour of a concurrent vending machine as an FSM shown in Figure 3a. It allows the user to insert a £1 coin (action pound) concurrently with making an order (actions coke and choc). Note that the concurrency between actions pound and coke as well as pound and choc is represented by *interleaving* – there are two corresponding diamonds in this FSM, and the layout is chosen so as to highlight them.



(a) FSM specification. (b) Synthesised Petri net.

Figure 3: Concurrent vending machine.

A PN can often be automatically obtained from the initial FSM model by the process called synthesis (`Conversion→Net synthesis [Petrify]` menu item of FSM model). The resultant PN is shown in Figure 3b; one can validate that be reachability graph of this PN coincides with the original FSM. Note that

transitions pound and coke as well as pound and choc are now truly concurrent in the PN.

5.2 Design of Asynchronous Circuits: VME Bus Controller

In this case study WORKCRAFT is used to formally specify and derive a speed-independent implementation of VME bus controller. A controller for VME bus provides an interface between a data bus and a slave device, as shown in Figure 4.

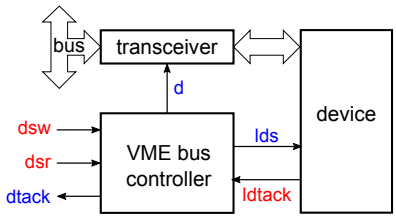


Figure 4: VME bus interface.

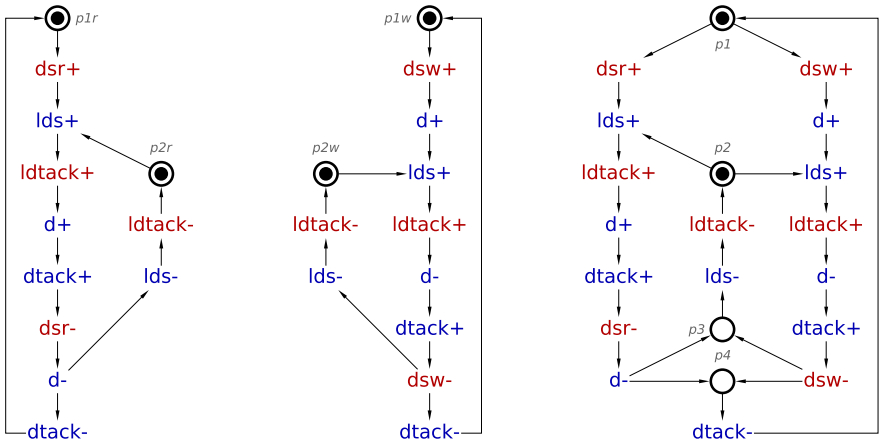
The controller has two modes of operation: reading from the device into the bus (activated by dsr+) and writing from the bus into the device (activated by dsw+). In the reading mode, a request to read data form the device is made through lds+. When the device has the data ready and this is acknowledged by ldtack+, the controller opens the transceiver by d+ and notifies the bus that data is ready for transfer by dtack+. After the read operation is complete, all the signals return to the initial state.

In the writing mode, once the data is stable on the bus, the transceiver is opened by d+, and the write request is made by lds+. When the device acknowledges the receipt of data by ldtack+, the transceiver is closed with d-, thus isolating the device from the bus, and the bus is notified that the write operation is complete by dtack+. After that all the signals return to the initial state.

The read and write modes of VME control are captured in WORKCRAFT by the STGs in Figures 5a and 5b respectively. These two STGs describe the behaviour of the same circuit and need to be combined into one specification by merging their initial states. Note that transitions ldtack-, lds- and dtack- occur in both branches of the choice and can also be merged. For merging places and transitions one can use the corresponding operations in the `Transformations` menu. The complete STG specification of VME bus controller is shown in Figure 5c.

Before proceeding to synthesis the STG needs to satisfy the following soundness properties (`Verification` menu of WORKCRAFT enables checking all these properties with a single click):

– *Deadlock freeness* – every reachable marking enables at least one transition.



(a) Read operation. (b) Write operation. (c) Combined operation.

Figure 5: STG specification of VME bus controller.

- *Consistency* – the '+' and '–' transitions of every signal alternate in every execution, always starting with the same sign.
- *Input properness* – an input cannot be disabled by an output or internal signal, and cannot be triggered by an internal signal.
- *Output persistency* – an enabled output or internal signal cannot be disabled by any other signal.

The STG specification can now be synthesised into an asynchronous circuit. A complex-gate solution is as follows (`csc0` signal was automatically inserted by PETRIFY back-end to resolve a CSC conflict):

```
INORDER = dsr dsw ldtack d dtack lds csc0;
OUTORDER = [d] [dtack] [lds] [csc0];
[d] = dsr ldtack csc0' + dsw (csc0 + ldtack');
[dtack] = d' csc0' (dsr' + dsw) + dsw' d;
[lds] = csc0';
[csc0] = dsr' d' (csc0 + dsw') + ldtack csc0;
```

This solution uses complex gates that do not usually exist in real gate libraries. Such gates need to be decomposed to map them to existing library gates – this is done by logic decomposition that preserves speed-independence of the circuit. The library of available gates can be passed to WORKCRAFT in SIS GENLIB format.

The result of technology mapping into TSMC gate library (with an addition of C-elements) is shown in Figure 6. One can verify (via `Verification` menu) that the circuit implementation is deadlock-free, hazard-free, and conforms to the original STG specification.

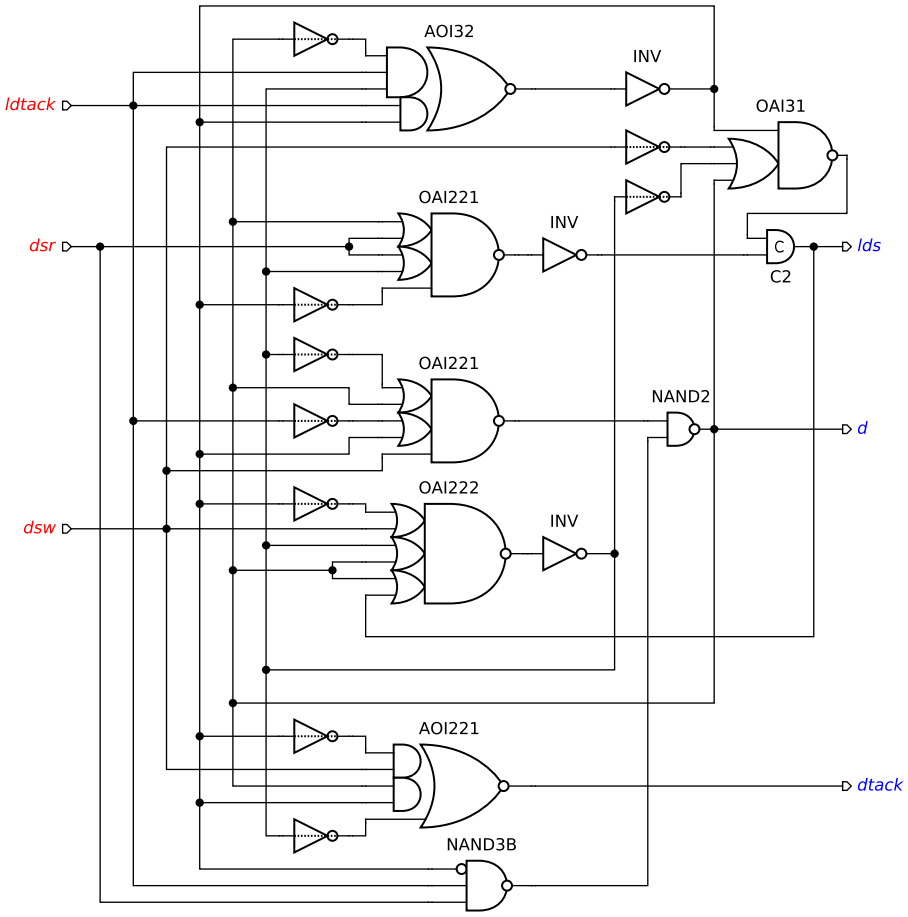


Figure 6: Implementation of VME bus controller. The dotted lines through the inverters express the timing assumption that their delays are smaller than any other gate delay in the circuit.

5.3 Analysis of Asynchronous Pipelines: Baseband Transmitter

Pipelines can be modelled in WORKCRAFT using the Dataflow Structure (DFS) formalism. This abstraction separates the structure and the function of the system from the implementation details of its components.

The possibility of formally modelling and reasoning about the system at this architectural level is crucial, as the design decisions made at this level will affect all the subsequent stages of the design. Moreover, optimisations performed at this level are likely to have a much stronger impact than micro-optimisations applied towards the end of the design process.

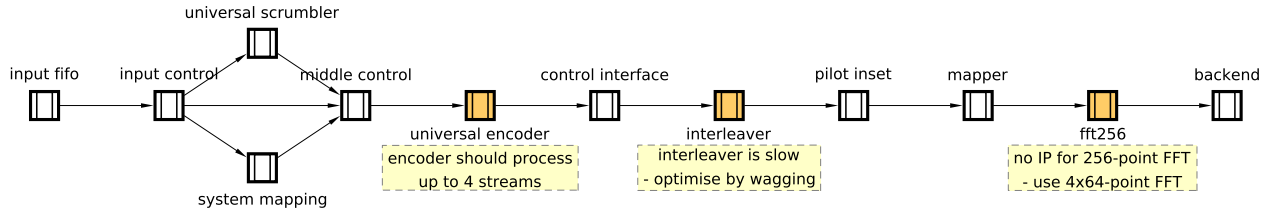
A DFS model in Figure 7a represents pipeline stages of a baseband transmitter at a rather high level of abstraction. Even at this level important design decisions can be made about specific implementation of the pipeline components. For example, information about the maximum number of streams to encode (up to 4 streams), available components for Fast Fourier Transform (maximum 64-points) and the preliminary performance estimates (interleaver is the bottleneck as it is 3 times slower than the other pipeline stages) define the refinement of the DFS model shown in Figure 7b. Note that the refinement of the interleaver stage into three concurrent slices was obtained automatically using the 3-way *wagging* [7] operation in the `Transformations` menu.

5.4 Instruction Set Architecture: ARM Cortex-M0+

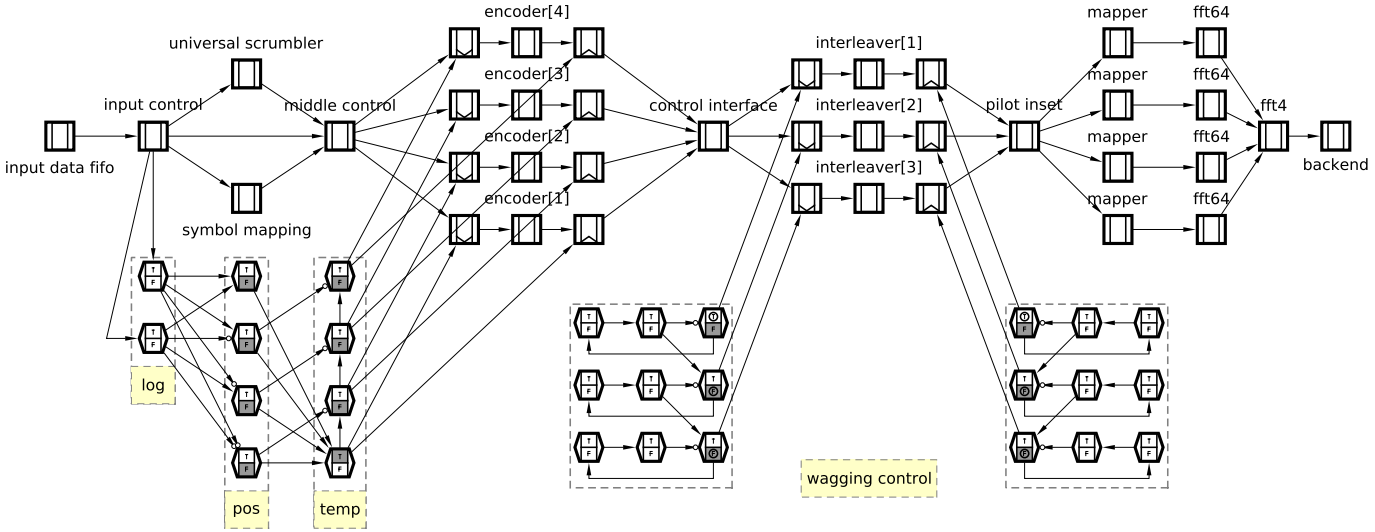
WORKCRAFT can be used to specify and explore processor Instruction Set Architectures (ISAs), and synthesise efficient hardware implementations for their microcontrollers. In this section we use WORKCRAFT to specify a subset of ARM Cortex-M0+ instructions, automatically derive optimal opcodes for them, and synthesise a Verilog netlist for the corresponding microcontroller. The presented approach relies on CPOGs as the modelling formalism [15] that provides the designer with a convenient ISA visualisation notation as well as analysis methods.

Figure 8 shows 11 partial orders corresponding to instruction classes of ARM Cortex-M0+ processor [1][11]. The events in these partial orders correspond to primitive computation steps performed during instruction execution:

- PCIU stands for the Program Counter Increment Unit. It is used to advance the program counter when fetching instruction opcodes and operands from the program memory.
- The Instruction Fetch Unit (IFU) loads instruction opcodes and immediate instruction operands from the program memory into the processor instruction register.
- The data memory can be accessed using the Memory Access Unit (MAU), which transfers data between the processor registers and the data memory.
- The Arithmetic Logic Unit (ALU) is capable of performing basic computations such as addition, multiplication, comparison, bitwise Boolean logic operations, etc.



(a) Abstract DFS model.



(b) Refined DFS model.

Figure 7: Self-timed baseband transmitter.

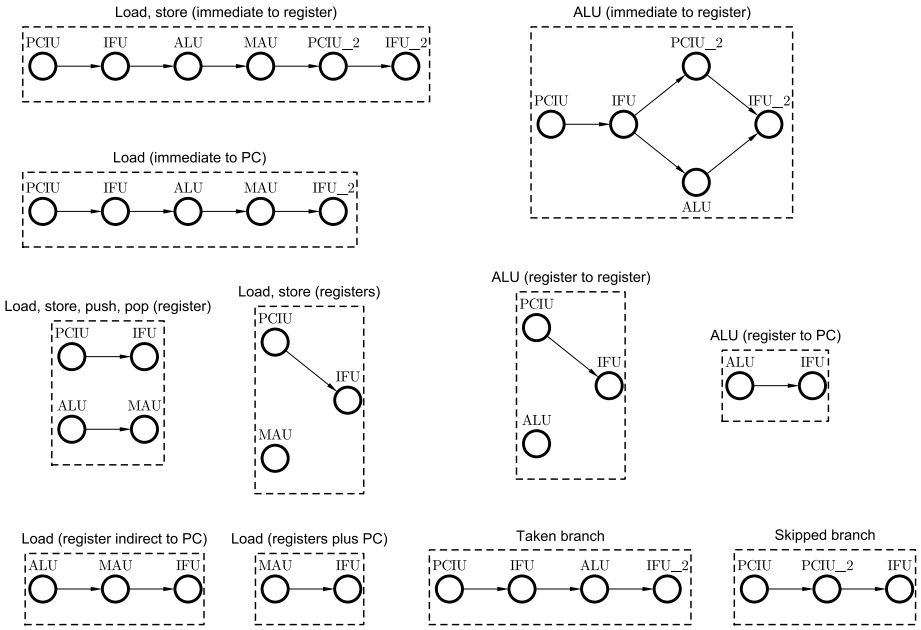


Figure 8: ARM Cortex M0+ instruction classes.

The partial orders can be created in WORKCRAFT either by individually placing and connecting events, or by using the Parameterised Graphs Algebra [17] plugin, which allows one to specify partial orders algebraically (as an example, the partial order *ALU (register to register)* in Figure 8 can be specified by the expression `PCIU -> IFU + ALU`).

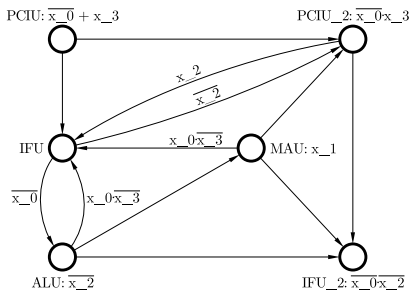


Figure 9: Compact representation of ARM Cortex M0+ instructions.

The CPOG encoding plugin SCENCO [11] can be used to automatically derive instruction opcodes minimising the area of the resulting microcontroller.

The plugin allows one to set a desired opcode length and reserve opcode bits in certain instructions. Several encoding algorithms are supported – see Table 1. Note that choosing good opcodes has a significant impact on the area of the resulting microcontroller. Having computed the optimal opcodes for the given partial orders, it is possible to represent them compactly as a CPOG, see Figure 9. Furthermore, it is possible to synthesise the processor microcontroller that can execute all 11 instruction classes. The microcontroller can be automatically synthesised as a Digital Circuit in WORKCRAFT – see Figure 10, or exported as a Verilog netlist for processing with traditional EDA toolkits.

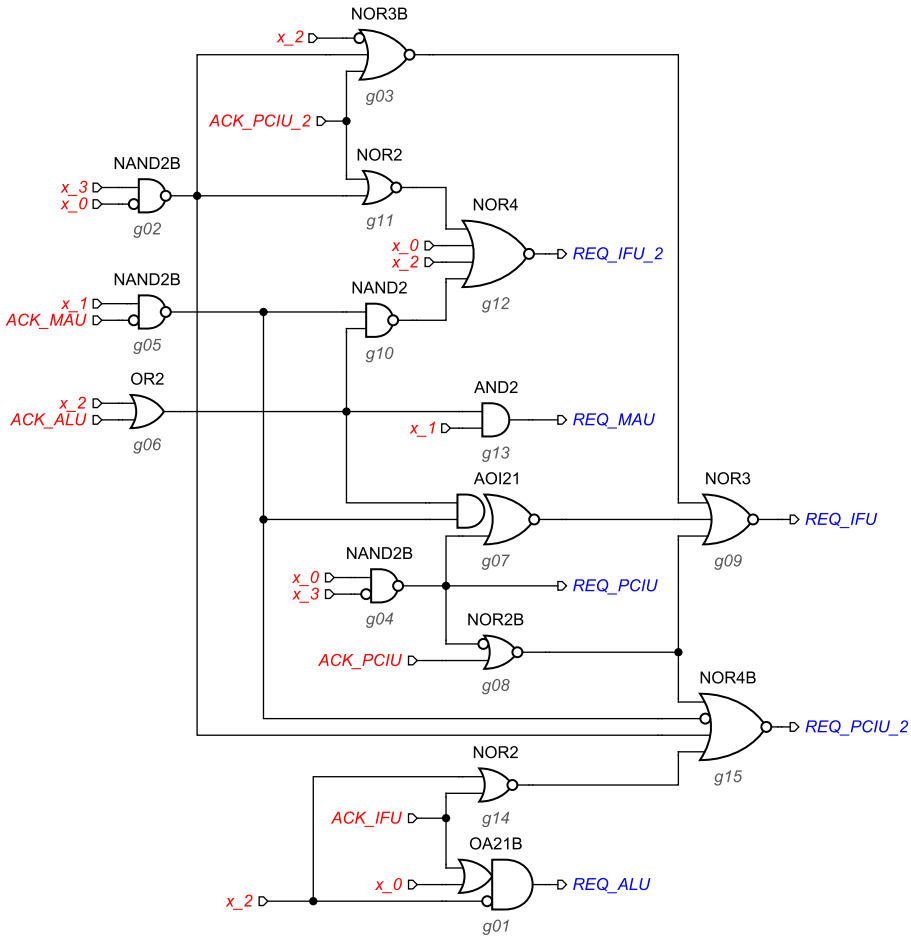


Figure 10: Synthesised microcontroller.

Encoding algorithm	Microcontroller area, μm^2	Gate count
Sequential assignment	468	28
Random, 10 samples	436	28
Random, 100 samples	364	24
Random, 200 samples	372	24
Heuristic [11], 10 seeds	312	21
Heuristic, 100 seeds	256	16
Heuristic, 200 seeds	**248**	**15**
SAT-based [16]	256	16

Table 1: Comparison of CPOG encoding algorithms.

5.5 Accident Investigation: Ladbroke Grove Rail Crash

In this case study WORKCRAFT is used for modelling Ladbroke Grove rail crash [20]. Ladbroke Grove, London, was the scene of a serious railway accident in October 1999. An outbound diesel train collided with a high speed train at the combined velocity of 130mph, with 31 people killed and over 500 injured. The immediate cause of the disaster was that the diesel train passed signal SN109 at red, although there were many other contributing factors.

The SON model in Figure 11 captures the details of the Ladbroke Grove rail crash. It consists of five occurrence nets that represent separate parties of the accident:

- signals – represents the track signals that diesel train passed by in sequence. The first signal SN43 is green (proceed). The next two signals SN63 and SN87 are both yellow (caution). The last one SN109 is red (stop).
- driver – shows the behaviours of the diesel train driver. It captures the operation of the train speed control with seven speed notches (1-7) and the brake actions.
- diesel train – models the speed of diesel train as a reaction to driver actions.
- control centre – is the behaviours of the signaller who was in charge of monitoring the situation.
- HST – models the high speed train shortly before collision.

WORKCRAFT allows one to verify several kinds of properties of SON models, including behavioural, structural and temporal consistency.

6 WORKCRAFT users

User-centred design process is at the heart of WORKCRAFT philosophy. WORKCRAFT is used for different purposes, such as education, research, and industrial circuit design to list a few. Therefore there are several categories of users: taught students, academics and research students, industrial engineers. The developers of WORKCRAFT also use it intensively in their research. It is not trivial to develop a tool that suits all these categories. Hence the developers made a point not

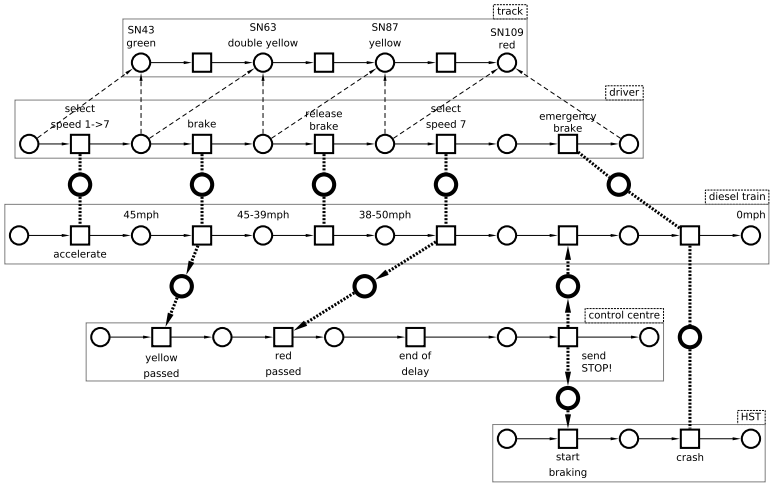


Figure 11: SONs model of the Ladbroke Grove rail crash.

to impose their design decisions on the users, but rather listen to their feedback and try to understand their needs. This resulted in an intuitive GUI. Historically there have been many reports of features as bugs because the users were confused due to lack of experience and understanding. Rather than dismissing these reports, an effort was made to improve the interface and address the sources of confusion.

Plenty of resources to support Workcraft users are available from the `http://workcraft.org/` website:

- binary distributions for Windows, Linux, Mac OS X and a link to the source code at GitHub;
- educational materials in the form of tutorials and case studies;
- user manuals;
- guidelines for developers of new IGMs, plugins, and back-ends;
- news track and announcement of the training events.

6.1 Education

Workcraft and the developed educational materials have been deployed in the learning process at Newcastle University as a part of undergraduate module *CSC3324: Understanding Concurrency* for Stage 3 students within the *Research-led Teaching* initiative. Not only this enhanced the learning process for the students, but the developers have also benefited by gathering invaluable data about

- how novice users attempted to install and use Workcraft;
- unexpected use patterns;
- features which students found difficult or confusing;

– bug reports;
– suggestions for improvements;
– feature requests.

This data helped to improve the stability and usability of Workcraft. The developers directly engaged with the students and tried to resolve the issues as soon as possible. In particular, several versions of Workcraft were released in the course of the module, so that the students could see their feedback making real difference.

Workcraft is also used by other universities to support teaching of asynchronous circuit design, in particular by Technical University of Denmark for *02204: Design of Asynchronous Circuits* course and by Southampton University for the *MSc Systems on Chip* course.

6.2 Research

Researchers use Workcraft for formal modelling or designing various systems. Accessible simulation, synthesis and verification features enhance this process and automate some routine and tedious parts of it. Researchers occasionally come up with new IGM formalisms. The extendible architecture of Workcraft makes it possible to implement them as plugins, benefiting from the functionality that is already present in the Workcraft core and existing plugins.

An important part of researcher survival is funding applications. Workcraft has been used, and is planned to be used, in several research proposals. There are several ways in which Workcraft is useful for this purpose:

– As a platform for developing solutions for the problems stated in the project proposal.
– As an impact and dissemination channel: the research results are encapsulated in Workcraft to enable non-expert users to reap the benefits.
– Establishing industrial links which strengthen the impact cases. Industry also helped to focus on industrially relevant topics and discover new areas of research (e.g. analogue-to-asynchronous interfaces) generating ideas for research proposals.
– In the REF2014 UK national exercise, one of the Newcastle socio-economic impact case studies was *Worldwide Adoption of Asynchronous Circuits and Improved Business Process Modelling* – it was judged to be world-leading. This case study included the development of techniques and tools for the synthesis of asynchronous systems (these tools are now incorporated into Workcraft as back-ends). This achievement can be leveraged in future funding applications.

On the practical side in research papers related to Petri nets and asynchronous circuits one usually needs a number of diagrams. Using LaTeX primitives or general-purpose graphics editors is tedious and time-consuming. Workcraft export feature allows one to produce high quality diagrams of the supported IGMs in a variety of graphic formats with minimum effort. This also improves the

consistency between the published diagrams and the models that were actually used in the research.

6.3 Industry

Interaction with industry is of paramount importance for the researchers living in the ivory tower of academia, and industrial uptake is a crucial criterion of success for research software. We are proud that Workcraft is used by several hardware design companies to develop real-life electronic circuits.

Interaction with industry proved to be of enormous use. It provided real-life case studies, gave insight into problems faced by industry where research is necessary and what kind of solutions can be actually implemented in silicone. We had a number of joint projects and publications with industrial partners and exchanged numerous visits. Several tutorials and teaching materials were developed to address educational needs of engineers. The feedback and feature requests submitted by the engineers helped to improve the usability of Workcraft in industrial context as well as identify and implement missing links in the design flow.

7 Workcraft timeline

Workcraft started in 2006 as a PhD research project of Ivan Poliakov to provide a modelling and simulation tool for PNs and STGs. Additional requirements were to make it cross-platform, user-friendly, and easily extendible. The result proved to be a convenient framework for a range of PN-like models (subsequently called IGMs), and other researchers started adding their favourite formalisms to Workcraft.

Figure 12 captures the history of Workcraft as a SON model. The central part of the diagram represents the development of Workcraft versions, highlighting some major features associated with each release. The shaded vertices denote new IGMs introduced in the corresponding release. Developers of the codebase and their contribution to specific features are listed at the top. The main dissemination results in terms of exhibition demonstrations, training sessions, and teaching are at the bottom of the diagram.

The main milestones in Workcraft development are as follows:

- Workcraft 1 series (A New Hope) was implemented in Java with native libraries for OpenGL graphics. It supported PNs, STGs, ACMs and Digital Circuits models. It also worked as a sandpit for trying different flavours of self-timed datapath models with various token game semantics. This lead to formalisation of a folklore static DFS model and its extension with dynamic elements. Figure 13 shows Workcraft v1.0 when simulating a motor control system with asynchronous communication between slow speed controller, fast torque controller, and the slowest adaptive parameter tuner [13].

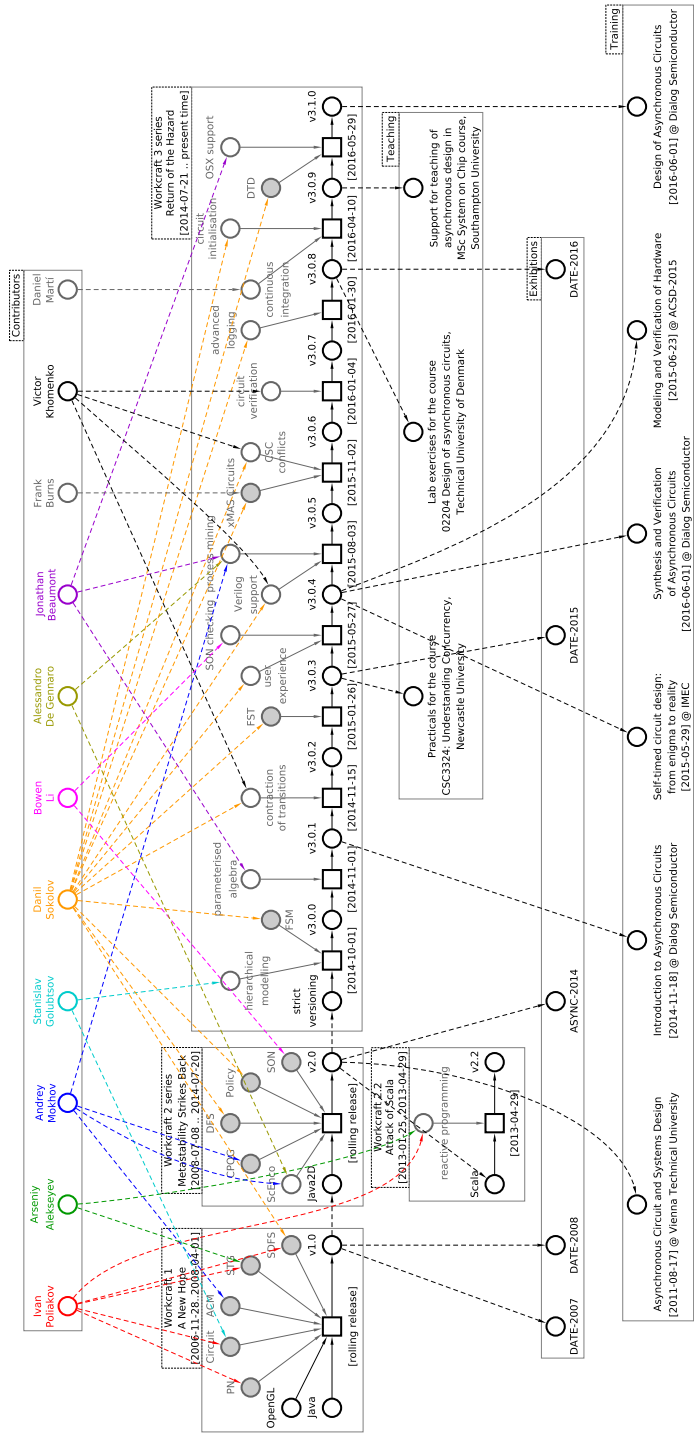


Figure 12: WORKCRAFT timeline expressed in SONs.

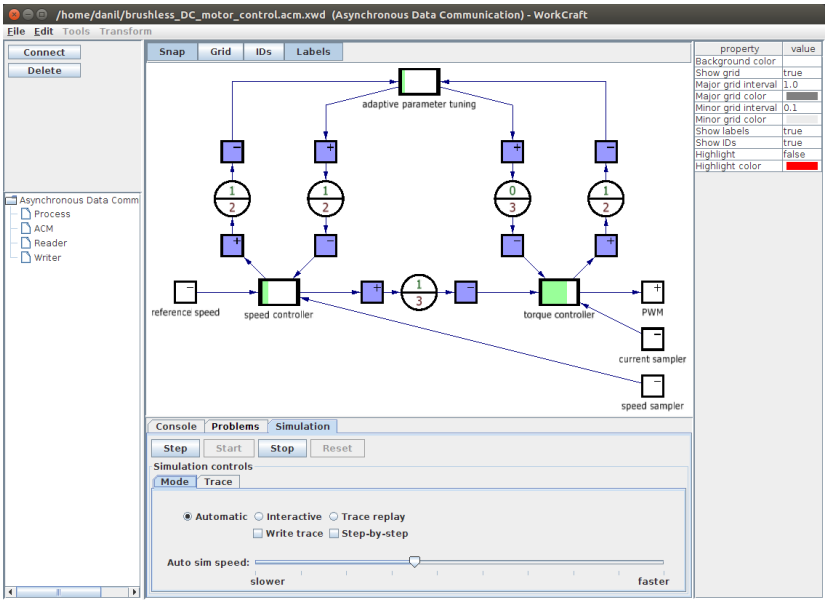


Figure 13: Modelling asynchronous communication in WORKCRAFT v1.0.

– WORKCRAFT 2 series (Metastability Strikes Back) was a major revision of the core functionality, plugin architecture, file exchange format, and rendering engine. Several new IGMs we added in this series, namely CPOGs, SONs and Policy nets. A screenshot in Figure 14 shows a development of Petri net representation of algorithm for generating a pair of public and private keys used in AES encryption in WORKCRAFT v2.0 [26].

– WORKCRAFT 2.2 branch (Attack of Scala) was a heroic attempt to direct the development into reactive programming paradigm using Scala functional programming language. Due to inherent complexity of the design concepts only few researchers could contribute to this development. As a result this branch was abandoned.
– WORKCRAFT 3 series (Return of the Hazard) is currently under active development and follows regular schedule of releases (as opposed to rolling releases in the previous development). The focus of this series is on improving user experience and developing tutorial materials. Several new IGMs were also added, namely FSMs, FSTs and DTDs. Figure 15 shows a screenshot of WORKCRAFT v3.1.0 when designing an asynchronous controller for a basic buck converter [5].

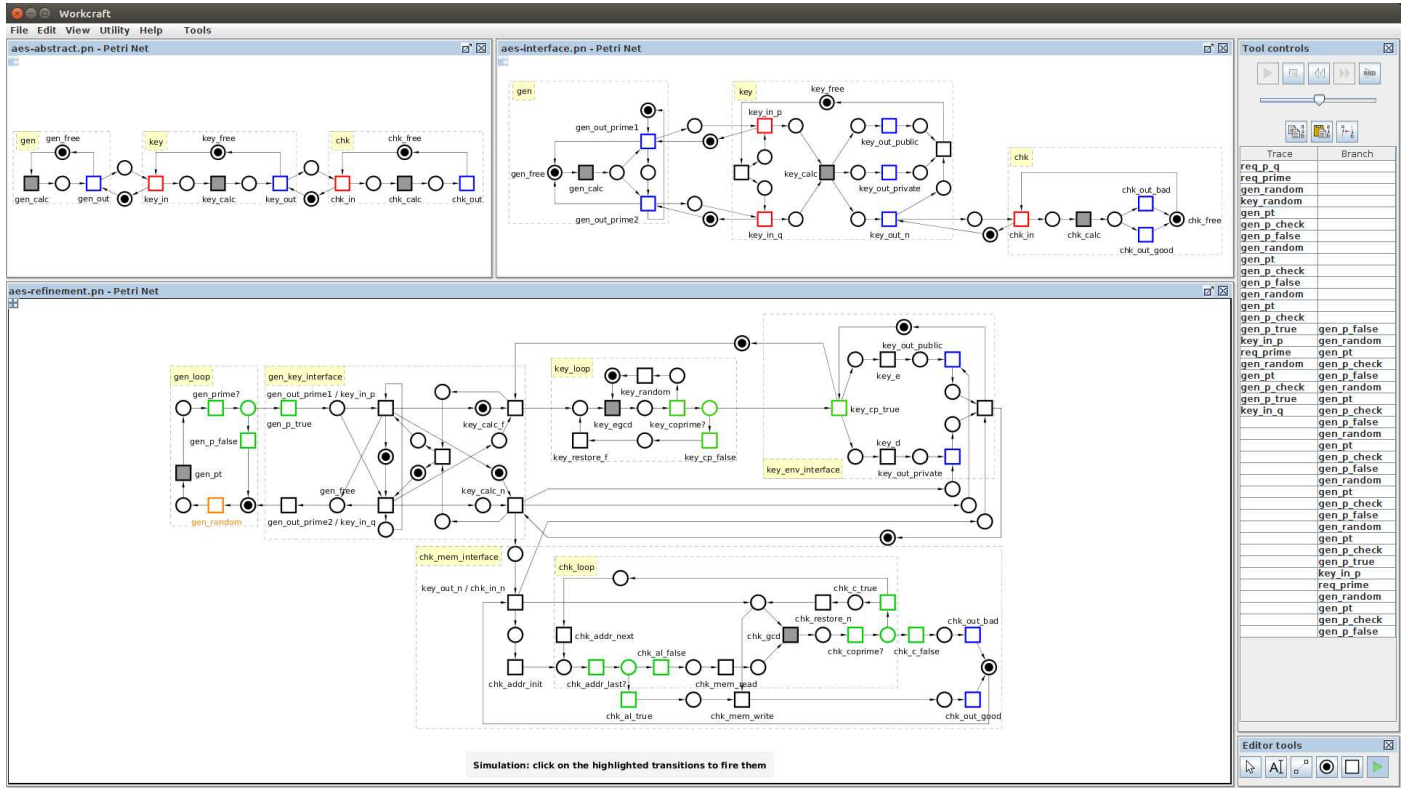


Figure 14: Refinement of algorithm for generation of AES keys using Petri nets in WORKCRAFT v2.0.

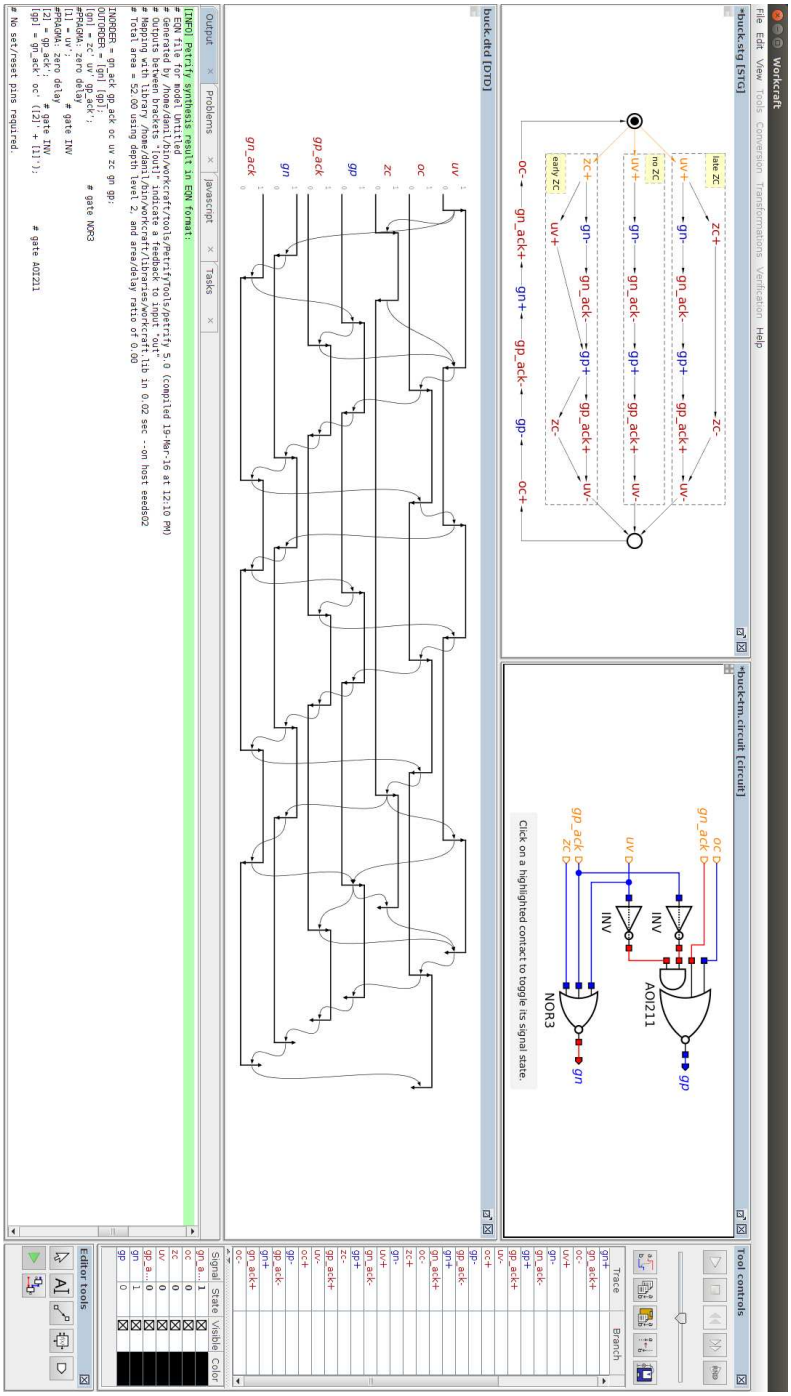


Figure 15: Design of a controller for basic buck converter in WORKCRAFT v3.1.0.

8 Conclusions

Workcraft is a versatile framework for capturing, simulation, synthesis and analysis of IGMs. It opens access to all the goodness hidden in command-line academic tools. This has significantly increased the user base, including industrial users, and thus the research impact of this software. The future plans include:

- Promoting Workcraft to an even wider audience. This will be achieved by creating and delivering tutorials to academic and industrial audience, development of the website [4], and improving the user experience based on their feedback.
- Encouraging other researchers to integrate their tools as back-ends.
- Extending Workcraft by adding new IGMs and implementing new features for existing ones. This will often be based on user requests. For example, waveforms model will be added in near future based on a request from an industrial partner.
- Finding other areas of application for the supported methods. For example, techniques based on STGs and speed-independent circuits appear to be well suited for modelling and analysis of Genetic Regulatory Networks [6].

Acknowledgements

The work on Workcraft was partially supported by Dialog Semiconductor, Maxeler Technologies, Royal Society (Research Grant "Computation Alive"), and the following EPSRC research grants:

- SElf-timed DATapath synthEsis (SEDATE) – EP/D053064/1
- Self-Timed Event Processor (STEP): EP/E044662/1
- Globally Asynchronous Elastic Logic Synthesis (GAELS) – EP/I038551/1
- Parallel Model Checking – GR/M99293/01
- Design And Verification of Asynchronous Circuits (DAVAC) – EP/C53400X/1
- Asynchronous design for Analogue electronics (A4A) – EP/L025507/1
- UNderstanding COmplex system eVolution through structurEd behaviouRs (UNCOVER) – EP/K001698/1
- Dataflow Computation à la Carte – Newcastle University Impact Acceleration Account EP/K503885/1

References

1. ARM Cortex-M0+ technical reference manual. `http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0432c/index.html`.
2. Petrify homepage. `http://www.cs.upc.edu/~jordicf/petrify/`.
3. UnfoldingTools homepage. `http://homepages.cs.ncl.ac.uk/victor.khomenko/tools/UnfoldingTools/current/`.
4. Workcraft homepage. `http://workcraft.org/`.

5. Towards asynchronous power management. In *Proc. IEEE Faible Tension Faible Consommation(FTFC)*, 2014.
6. R. Banks, V. Khomenko, and J. Steggles. A case for using signal transition graphs for analysing and refining genetic networks. *Electron. Notes Theor. Comput. Sci.*, 227:3–19, 2009.
7. C. Brej. Wagging logic: Implicit parallelism extraction using asynchronous methodologies. In *Proc. Application of Concurrency to System Design (ACSD)*, pages 35–44. IEEE Computer Society, 2010.
8. F. Burns, D. Sokolov, and A. Yakovlev. GALS synthesis and verification for xMAS models. In *Proc. Design, Automation & Test in Europe Conference & Exhibition*, pages 1419–1424, 2015.
9. T.-A. Chu. *Synthesis of self-timed VLSI circuits from graph-theoretic specifications*. PhD thesis, MIT Laboratory for Computer Science, 1987.
10. J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. PETRIFY: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE Transactions on Information and Systems*, E80-D(3):315–325, 1997.
11. Alessandro de Gennaro, Paulius Stankaitis, and Andrey Mokhov. A heuristic algorithm for deriving compact models of processor instruction sets. In *International Conference on Application of Concurrency to System Design (ACSD)*, pages 100–109. IEEE, 2015.
12. V. Varshavsky (Editor). *Self-Timed Control of Concurrent Processes*. Kluwer Academic Publishers, 1990.
13. F. Hao. *Asynchronous Data Communication Mechanism Models in Matlab and Their Applications*. PhD thesis, Newcastle University, 2007.
14. M. Koutny and B. Randell. Structured occurrence nets: A formalism for aiding system failure prevention and analysis techniques. *Fundam. Inform.*, 97(1-2):41–91, 2009.
15. A. Mokhov. *Conditional Partial Order Graphs*. PhD thesis, Newcastle University, 2009.
16. A. Mokhov, A. Alekseyev, and A. Yakovlev. Encoding of processor instruction sets with explicit concurrency control. *IET Computers and Digital Techniques*, 5(6):427–439, 2011.
17. A. Mokhov and V. Khomenko. Algebra of parameterised graphs. *ACM Transactions on Embedded Computing*, 13(4s), 2014.
18. D. Muller and W. Bartky. A theory of asynchronous circuits. In *Proc. Int. Symp. of the Theory of Switching*, pages 204–243, 1959.
19. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
20. The Rt Hon Lord Cullen PC. The Ladbroke Grove rail inquiry, 2001.
21. I. Poliakov, V. Khomenko, and A. Yakovlev. WORKCRAFT – a framework for interpreted graph models. In *Proc. Application and Theory of Petri Nets (ATPN)*, volume 5606 of *Lecture Notes in Computer Science*, pages 333–342. Springer-Verlag, 2009.
22. I. Poliakov, A. Mokhov, A. Rafiev, D. Sokolov, and A. Yakovlev. Automated verification of asynchronous circuits using circuit Petri nets. In *Proc. Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, pages 161–170, 2008.
23. I. Poliakov, D. Sokolov, and A. Mokhov. WORKCRAFT: a static data flow structure editing, visualisation and analysis tool. In *Proc. Application and Theory of Petri Nets (ATPN)*, pages 505–514, 2007.

24. L. Rosenblum and A. Yakovlev. Signal graphs: from self-timed to timed ones. In *International Workshop on Timed Petri Nets, Torino, Italy, 1985*, 1985.
25. D. Sokolov, I. Poliakov, and A. Yakovlev. Analysis of static data flow structures. *Fundamenta Informaticae*, 88(4):581–610, 2008.
26. D. Sokolov and A. Yakovlev. GALS partitioning by behavioural decoupling expressed in Petri nets. In *Proc. Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, pages 17–26, 2014.
27. A. Valmari. The state explosion problem. In *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, pages 429–528. Springer, 1998.

Fifty Shades of Synchrony

Andreas Steininger

TU Wien, Embedded Computing Systems Group E182-2,
Treitlstr. 3, A-1040 Vienna,
steininger@ecs.tuwien.ac.at

Abstract. This paper reflects on how, from the view of a VLSI designer, we coordinate activities in our daily lives. Example cases from transportation and recreational activities illustrate that both, synchronous and asynchronous approaches are in use, and there are good reasons for that. Interestingly, there is rarely a "black or white" – combined approaches can be found in many places.

It is the author's hope that this paper can serve two purposes, namely (a) to illustrate the differences in the fundamental principles of the two timing paradigms and their implications for a doctor's consultation, e.g., and (b) to inspire the reader to leverage the proven "grey" approaches from everyday life for solving problems in the VLSI domain.

1 Introduction

Since several decades more or less all reasonably complex logic devices are clocked, which means their timing is based on the synchronous paradigm. This approach is very efficient in both, design and implementation. In parallel to that, a relatively small research community of "asynchronous revolters" has kept on elaborating conceptually elegant methods for designing circuits that operate in a "self-timed" fashion, i.e. without a global clock. The continuous debate on which approach, synchronous or asynchronous, is preferable has been fueled by the increasing visibility of shortcomings of the dominant synchronous approach caused by the proceeding miniaturization of device structures. Recent examples for such deficiencies are power consumption/heat dissipation, and parameter or voltage variations. More often than not, this debate unveils substantial misunderstandings on the differences between asynchronous and synchronous paradigm. It is the aim of this paper to contribute to a clarification here that is comprehensible for non-experts. To this end we will go away from the domain of VLSI systems and look for comparable problems we encounter in everyday life. There we will investigate which solutions – synchronous or asynchronous – we employ there. Beyond (hopefully) providing a good illustration for the key concepts of the approaches, such a comparison may inspire new solutions for VLSI that leverage the wealth of (intuitive) knowledge and experiences of generations that shaped these everyday solutions.

2 The VLSI designer's world

In the VLSI domain the problem is to coordinate some kind of cooperative execution, like shared/distributed processing of a task (like in pipelined or parallel execution units), access to a shared resource, or simply a data exchange. In the latter, e.g., the receiver must capture the data at a point in time when they are actually provided by the sender. Synchronization is required and becomes non-trivial, as soon as the partners operate concurrently, i.e. each partner operates at its own speed, determined by the complexity of its service (which may itself vary with input data or operating mode), communication delays, tolerances and jitter, and, most notably, coordination with other partners. Especially the latter can lead to extremely complex transitional dependences in the speed of operation.

There are two fundamental ways to establish a synchronization: In (the most pronounced form of) *the synchronous paradigm*, all partners are provided with the same notion of time, and a global schedule determines which partner has to perform which activity at which point in time. In fact, all partners receive the same clock signal whose edges define a discrete time base. A partner's schedule can be as simple as "capture your input data with every rising clock edge", or it may be a very involved behavior, determined by a protocol state machine. The important concept here is that each partner performs its activities *based on its local view only*, without caring about the others. The global schedule (circuit design) takes care that those local activities, in their combination, provide the desired overall service. The clock period is usually constant, and each partner has to finish its local task by a given deadline (usually the next active clock edge), where most often being too early does not pose a problem but does not yield a benefit either.

The asynchronous approach is based on an explicit communication between the partners, which is, unlike the actual application data flow that may occur between them as well, just dedicated to the coordination of their activities. This so called handshake is composed of some partner initiating an activity (request) and another one (or more) confirming its completion (acknowledge). For a data exchange, the sender will, e.g., provide the data to the receiver's input, then request the exchange; and the receiver will capture the data and acknowledge their reception. The important point here is that the partners *communicate* to establish the coordination, so there must be suitable channels available for that (the handshake signals)[1]. This communication allows taking care of the partner's state. In fact, the handshake establishes a closed-loop timing control between the partners that adapts the timing of the local activities to the abilities of the respective partner(s). In order to consider the partners' state, one needs to know all partners involved in a cooperative execution. In a multicast communication, e.g., the sender can remove its data only after having received the acknowledges of all intended receivers.

[1] The handshake may be intertwined with the data.

It is often non-trivial to directly identify the completion of a local event which is needed to trigger a request or acknowledge signal. For the simple example of a data exchange, e.g., it is not obvious when exactly the data have arrived at the receiver's input, neither when exactly their capturing by the receiver is finished. The lack of such exact knowledge compromises the otherwise very beneficial closed-loop behavior of the "delay-insensitive" asynchronous approach. In contrast, the asynchronous "bounded delay approach" simply accepts this imperfection and uses a time delay as an indirect measure for completion, thus often saving significant overheads.

3 Synchrony and asynchrony in daily life

As a first real-life example let us look at transportation. Here the service is to have a group of persons moved from location A to B jointly, and synchronization is required to have these persons on the transportation medium upon departure.

The synchronous incarnation of this problem is a train: There is a global schedule known to everyone, which gives a deadline for boarding. The train will depart as scheduled, irrespective of who has boarded, and, as the author knows well from his daily ride to work, without mercy for latecomers. No handshake, no communication. The passengers are acting concurrently in the sense that each of them has his own history of getting up in the morning, not finding his skirt, meeting a friend on the way[2], getting stuck in a traffic jam[3], etc. Therefore they arrive at the train at different times, and they are "synchronized" by the joint departure.

The asynchronous counterpart would be a family returning with the car from a hike. Dad will start driving home with the car, as soon as mom, Tom and Nelly are comfortably sitting in the car. As long as Nelly is still flirting with the incredibly nice boy she just met on the trail, there is no way for dad to start going – even if since days he had insisted on being back home again before 7pm, in order to make it for a beer with his friend Gordon.

Some key differences in the concepts become clearly visible here: For the train it is neither known nor important who will join – the alignment of all passengers' largely different concurrent activities to one closed control loop would cause unmanageable complexity. Here the synchronous approach that abandons all history and just sticks to the global schedule is an essential means of decomposing this enormously complicated system; essentially moving the responsibility to each single passenger.

Even if he wished, this does not work for dad's problem, since he cares to have the whole family on board.

An interesting hybrid approach is the airplane, where the luggage check-in is ruled by a strict static time schedule (synchronous), while then the plane will not depart without any of the checked-in passengers (asynchronous) – at that point the airline knows each single passenger and, "for security reasons",

[2] This is an example of the transitional dependence mentioned above.
[3] This is an example of a shared resource requiring/enforcing synchronization.

cares to have everyone on board. While it was easy for dad to see that the family is complete, the airline companies check completion by means of barcode scanners, accompanied by manual counting through the flight attendants. This fussy procedure nicely illustrates that completion detection can be quite tricky. Interestingly, even in real life "bounded delay approaches" are sometimes being pursued to circumvent completion detection. One (for the author particularly annoying) example is the microwave oven: The user has to enter (and know!) a cooking time instead of the desired temperature, simply because time is cheaper to measure.

Another interesting aspect we can study with these transportation examples is fault tolerance: If Nelly drove away in her new boyfriend's car without saying a word (handshake!), then dad would search the whole surroundings for her instead of driving home. So asynchronous systems tend to deadlock in case of faults. This property is deeply buried in their "wait for all" philosophy imposed by the closed loop. Obviously such a concept will never work for a train. The synchronous approach can easily handle the absence of a partner – it simply does not care. Another way of viewing this is that the asynchronous paradigm has no flexibility with the number of completion messages but is arbitrarily flexible about their arrival times, while in the synchronous case there is no flexibility in time but arbitrary flexibility in the number of completing partners.

However, if the train is late for whatever reason, some of its next track segments will be used by other trains already[4], and passengers will miss their connections. Here the coordinated mission fails, as on that level of hierarchy the underlying global schedule essentially relies on the correct behavior of all partners. So the don't care philosophy fails if there are partners that are mission critical (like the professor in a lecture).

In practice one often finds solutions between these extremes, like an asynchronous approach with time-out (waiting for a colleague to drop by in the office during the afternoon, but at some point then going home when he still does not show up), or a combined synchronous/asynchronous approach. An example for the latter will be presented in the next section.

Before that, let us briefly turn to a different example, namely meeting friends to go for a bike tour. Some 30 years ago (like when the author did such tours) all partners simply agreed on a time and a street corner to meet, and everyone tried to be 5 minutes earlier because he knew the others will be unnecessarily waiting at the street corner if he is late – or even leave without him. Today (like when the author's son is doing such tours), for the same simple purpose of going on a bike tour, there is an incredible amount of communication on cell phone, skype or whatsapp like "Shall we meet at 4 or better at 5?", "Will Andrey also join?", "I will be 20 minutes late, wait for me" or "Decided not to come, don't wait for me", to negotiate for the actual meeting time. Why this change?

[4] To avoid crashes in these cases, sensors and signals are installed to provide the communication infrastructure required for a switch to the asynchronous paradigm at that point.

In earlier times, in the absence of mobile communication, there was no choice but resorting to the common notion of time which was the only (mobile) available relevant infrastructure then[5]. This forced us to use the synchronous paradigm. Nowadays the mobile phones provide the cheap mobile peer-to-peer communication channels required for the asynchronous approach. And for the reasons mentioned earlier this indeed seems to be the preferable approach for reasonably small groups.

4 The king of timing optimization

An extremely enlightening construction is the doctor's appointment. It starts with an "asynchronous" call at the doctor's office[6], upon which a date and time (synchronous!) are assigned for a consultation. However, when arriving synchronously, e.g. before the deadline, at the doctor's office, the patient waits until being called (handshake), and this can take a substantial amount of time. This can be regarded as a locally asynchronous globally synchronous approach, and although the abbreviation "LAGS" does not sound as nice as GALS[7], this principle is surprisingly often encountered in our everyday life. It can be interpreted as follows: We use a synchronous "appointment" approach to roughly schedule our daily agendas – that would simply be too complicated to be done asynchronously, "on demand". In a perfect world that alone would be sufficient, but in reality we cannot precisely meet all deadlines, as we have, e.g., to accommodate asynchronous requests as well (like accidentally meeting an old friend on the way), so we use an asynchronous approach for fine-tuning. This also relieves us from being too pessimistic about our deadlines, thus saving waiting time and increasing performance.

This brings up the issue of *waiting times* in general. Fundamentally, coordination of activities among concurrently operating partners implies some form of waiting for each other. In a synchronous setting, the rendezvous time must be chosen such that each partner (we care about) can be in time even under the worst circumstances. That is why we usually plan for some safety margin when going to an important date like our wedding – the idea is to make sure we make the deadline even if we get caught in a traffic jam or the car breaks[8]. Clearly, this causes unnecessary waiting in all but the worst case situations, for all partners. It is the responsibility of each partner to decide which margin he wants to have. Each partner's waiting time is then the difference between the considered worst case and the actual case. Under good conditions all partners will thus be early and wait. The train will normally not leave the station before schedule even if all passengers have already boarded.

[5] Note how immensely important global time as a common infrastructure is for our society, exactly for this type of synchronous coordination

[6] Synchronous doesn't make sense here unless one regularly gets ill

[7] "Globally Asynchronous Locally Synchronous", an approach recently used in VLSI

[8] That's probably the only reasonable argument why airlines urge passengers to be at the airport 2 hours before departure

In an asynchronous setting the completion/arrival of the slowest/latest partner determines the start of the joint activity. The slowest partner therefore does not have to wait, while all others wait. Recall that in the synchronous setting the rendezvous time was determined such that even the slowest partner can reach it under worst case conditions. Therefore, in the average case, even the slowest partner will arrive earlier than that, and all partners save waiting time. That is why asynchronous systems are said to have higher performance, namely average case performance, than synchronous ones with their worst-case performance.

However, why then do we notoriously have to wait for hours in the doctor's waiting room? This is because the doctor abuses this scheme: In the synchronous domain, he gives you a date for which he knows that he will never make it. This forces you to arrive at a time when he is not ready for sure, and at that time he switches to the asynchronous mode, which means you will always be the waiting partner and he will never wait (or have you ever seen the doctor welcome you with the words "I have been waiting for you already!"). Through that unfair trick the doctor increases your waiting time beyond the one you already had in the synchronous domain – so this system is not beneficial for the patient[9]. The doctor is using it because he considers himself a precious resource whose waiting time has thus to be minimized. Admittedly, there are no good alternatives: A purely asynchronous approach (visit the doctor on demand, scheduling by priority), is necessarily practiced in emergency stations, but has proven to cause even higher waiting times in practice for the lower priority cases. And a purely synchronous system with conservative estimation of the time per patient would severely decrease the doctor's throughput, and also disallow him to take care of sporadic emergency cases.

Actually, the doctor uses another trick to avoid waiting times on his side: buffering. Most often there is more than one patient waiting patiently, so should the doctor be faster with processing one patient, there is already a next one waiting (who of course also got a ridiculously optimistic rendezvouz time). This nicely illustrates the function of a buffer (waiting room) for compensating fluctuations in the processing time. Such a FIFO buffer – sometimes even signified by the fact that you literally get a number – is to the benefit of all patients, as it increases the doctor's throughput. However, it is yet another unfair trick of doctors to introduce some dubious kind of priority (for emergency cases like his brother-in-law's cousin) sometimes.

Obviously there is no point in using the LAGS scheme for starting a rock concert, where the sheer number of participants clearly calls for a synchronous solution. And on the family hike dad won't make friends by always looking at the watch and telling "Hurry up, we must be back at the car at 5pm!" or not starting the ride back at 4:45pm even if everyone is sitting in the car – this is clearly an asynchronous case. However, even the friends going on a bike tour from the example above (at least the classic version) employ a LAGS solution by agreeing on a time to meet and then wait until everyone (they care about) is here. In fact, we use this scheme many times a day. We use the synchronous schedule

[9] Interestingly this word's interpretation as an adjective becomes symptomatic here

for planning our day (you simply cannot plan if you give all partners arbitrary freedom), i.e. scheduling ourselves as a kind of shared resource, but then add flexibility by switching to the asynchronous mode, accounting for the fact that the interactions between the myriads of parallel and interacting processes on our planet are so complicated in detail that we call them unpredictable.

Note that in LAGS we often need to assign an ID to each of the partners. As an example, imagine the doctor using a schedule of the patients, and the patients establishing a FIFO order by pulling numbers as they arrive. The patients will (even in the absence of dubious priority) not know in which order to enter the doctor's room, and the doctor will clearly call them by name. In technical terms, when the jitter of arrival gets larger than the synchronous period (planned processing time per patient), out-of-order arrival is possible and identifiers need to be introduced. This is neither necessary in a purely synchronous approach ("Take the 8:30 train") nor in a purely asynchronous approach with strict FIFO ordering (pulling a number at a government office).

5 Conclusion

We have discussed real-life scenarios where either pure synchronous or pure asynchronous timing is beneficially used, and these have essentially confirmed the experiences from the VLSI domain. However, it turned out that in our daily lives we often use a locally asynchronous globally synchronous (LAGS) approach, which is quite in contrast to the GALS approach found in the VLSI domain. What VLSI designers might take away from this observation is that (a) the large complexity found on system level might be much easier to handle in a (coarse-grained) synchronous approach than with interacting asynchronous control loops, while (b) using continuous time for fine-tuning the local timing might be more appropriate than using ridiculously high clock frequencies to avoid losing performance when performing that task in the synchronous domain.

Admittedly, these are just relatively vague ideas, and some of the parallels drawn may even turn out wrong upon closer investigation. It is not the author's claim that this is a scientific paper (after all it is lacking related work and references), but it is hoped that this different view at the problem of VLSI timing may inspire new solutions. At minimum, however, the paper shall provide something to contemplate about during the next hours the reader spends in the doctor's waiting room.

Can Metastable States be Trapped?

Ghaith Tarawneh (ghaith.tarawneh@ncl.ac.uk)

Institute of Neurosience, Newcastle University, UK

Abstract. Synchronizers (commonly implemented as chains of two flip-flops) are used to reduce the probability that metastable states, emanating from asynchronous input transitions, propagate to state/data flip-flops and cause unwieldy catastrophes. Another way to think of this is that synchronizers "trap" metastability, preventing it from being observed by others circuits until a failure rate criteria is satisfied. Flip-flop chains are certainly one way of doing it but are there other ways in which metastability can be trapped? More importantly, can a circuit perform useful computations while trapping metastability in the same way that a synchronizer does? This paper will explore the theoretical prospects (and, potentially foolishly, break the long- held taboos) of allowing metastability inside generic Moore machines and finding ways to trap it and manage its effects.

1 Introduction

This paper will explore the theoretical possibility of creating synchronous state machines that meet a given functional specification despite being occasionally metastable. The motivation behind this perusal is to determine whether it is possible (at least in principle) to create multi-clock systems where signals can be transferred across clock domain boundaries without latency. As the reader may well be aware, there is a formal proof (using dynamical system theory) that a fundamental trade-off between latency and reliability is inherent in any Newtonian system that attempts to make a binary decision based on an analogue quantity [1]. It is therefore important to note outright that this paper will not attempt the impossible by trying to remove synchronization altogether. Instead, what will be considered is the more realistic (but perhaps equally questionable) possibility of turning a state machine into a large synchronizer housing many flip-flops and logic gates. To develop the intuition behind this idea, the paper will start by considering a synchronization conundrum in a hypothetical system.

2 The Island

Legend has it that in a time, so long ago, the best philosophers in all parts of the world, tired of wars and petty human affairs, decided to leave the main lands and live in a distant island, known to the rest of the world only as *Synchrona*.[1]

[1] Incidentally, philosophers in that era, known for their modest attire, were colloquially called by common people *flip-flops* in reference to their (often worn out) sandals.

The inhabitants of Synchrona sought to resolve all disorder by establishing strict rules to govern their interactions and day-to-day affairs. One such rule mandated that all philosophers meet each day, precisely at noon, to share any new philosophical insights they have had overnight. This daily routine worked well for the philosophers but was occasionally disturbed by the untimely arrival of small boats from the main lands, carrying messages from common people who sought the philosophers' wisdom. When a boat arrived during a meeting, a very peculiar thing often happened: the philosophers developed conflicting views about what was said during the interrupted meeting. These problematic disagreements were often resolved during the next one or two meetings (although they often led philosophers to reach nonsensical conclusions about certain matters).[2]

Even though disagreements were settled peacefully and had no long-term ill effects on Synchrona, any messenger boats dispatched carrying conflicting replies from the philosophers stirred doubt and greater trouble when received by common people on the main lands. The philosophers were aware of the dangers of propagating disagreements and, realizing that boat arrivals at Synchrona, the interruptions of their meetings and consequently their disagreements could not be prevented, decided to add one more rule to their routine. The new rule said that all discussions preceding the drafting of a reply to common people's questions may not depend on prior discussions on which the philosophers are still in disagreement. Knowing that deciding which discussions involved disagreements was itself a discussion prone to disagreement, the philosophers decided on a most unusual way to enact their new rule. After unloading their cargo, all inbound boats are to be forced into a two-day sail around Synchrona before docking to collect drafted replies and returning to people on the main lands. This way, the philosophers reasoned, any disagreements arising from the arrival of boats would have had enough time to be settled by the time a reply is to be shipped.

Ever since the new rule came into force, the common people stopped receiving the occasional news about the philosophers' disagreements. Rumor circulated that the philosophers devised a most ingenious way to abolish disagreements of all sorts (although the more educated remained skeptical of this). Unusual notes that did not make much sense such as "$1 + 2 = 7$" were still received from the island but all carried the signature of each philosopher known to be living there.

3 The Island as a State Machine

As the philosophers living on Synchrona have argued, metastable states can be "trapped" in a synchronous component if we ensure that value changes on its asynchronous input ports cannot be observed on the outputs within n cycles (where n corresponds to a given MTBF criterion). Synchronizers are the simplest circuits that satisfy this property; it takes n cycles for an input change to be observed at the output of an n-stage synchronizer. Gray counters satisfy

[2] Scientists inhabiting the island noted that the probability such disagreements persisted for t more seconds was reduced by a factor of $e^{t/\tau}$ (where τ depended on how well-fed were the involved philosophers).

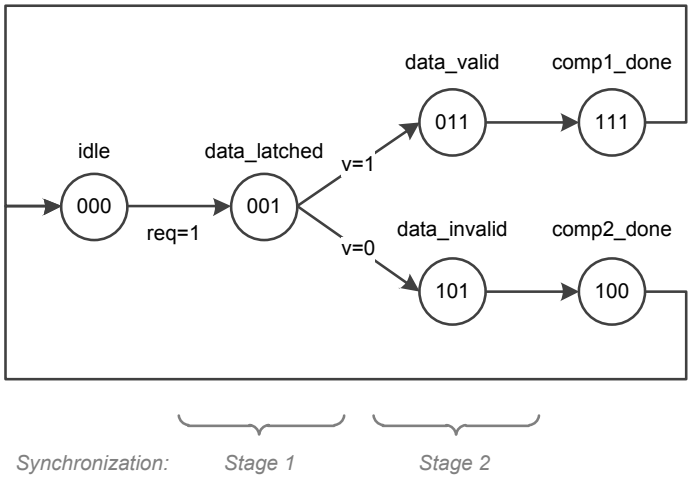


Fig. 1. A state graph of Synchrona

this property too but, like synchronizers, are not typically used for anything beyond counting. In this section we aim to present a toy example of a datapath controller that traps metastability in the same way that a synchronizer does. The controller processes a data item received asynchronously and synchronizes its associated request signal *simultaneously*. Our aim here is not to provide a rigorous methodology for designing such components but to present and argue for the correctness of at least one non-trivial example.

To continue our analogy from the previous section, our datapath controller is assumed to be Synchrona itself. Figure 1 shows a state graph of the island in which three "binary philosophers" (A, B and C) are used to encode its state (with A being the most significant philosopher). The graph has two branches that correspond to a sequence of computations to be performed based on whether the received data satisfies a certain condition v, where v is computed by the datapath after data has been latched. When `req` is asserted, the island is kicked into a series of transitions along one of the two main branches. State codes are chosen such that any two consecutive states within 2 transitions from the idle state differ by only 1 bit. This Gray-based encoding scheme ensures that any metastable state bit in the system will have a *single* sensitized combinational path to a destination state bit. When the island is metastable, therefore, only a single state bit is open to misinterpretation and the island will either transition to the next state or safely roll back to its existing one.

But what about the signal v? If v is generated by the datapath based on a data item that is received asynchronously then isn't v itself vulnerable to becoming metastable? No, the data bundling constraint implies that data bits arrive well before their corresponding `req` transition and so the controller state bits are the single point of failure in the system [2].

So far we have presented arguments that the controller will function correctly (more strictly, the arrival of `req` will trigger a chain of state transitions as per Figure 1, although occasionally the controller will stay in one of the states 000 or 001 for an additional cycle). Can the controller be safely integrated into a larger system without causing metastability failures? Yes if we make it impossible for external observers that monitor the controller's output ports to know when the controller is metastable (just like the philosophers did to hide news of their disagreements from common people). We do this by making the controller's output ports have the same values in the states 000 and 001. In our example we assume the controller has a single output `valid` that is logic high when the controller is in either state 011 or 101 and logic low otherwise (i.e. `valid` $= (A \oplus B) \wedge C$). `valid` will go high once synchronization is complete and is therefore a safe signal for external circuits to sample.

In effect, our controller has two "built-in" synchronizer chains: CB (when $v = 1$) and CA (when $v = 0$). The value of `valid` is asserted once the transition of `req` appears at the second synchronizer flip-flop (either B or A) and so the controller satisfies our criterion for functioning as a synchronizer. In the state graph, entering the state 001 corresponds to the completion of the first stage of synchronization and entering either state 011 or 101 corresponds to the completion of the second.

4 Conclusion

The main point of the informal discussion presented here is to show that there exists more than meets the eye in the design space of metastable synchronous components. Although it is impossible to prevent the occurrence or propagation of metastable states in synchronous modules, our toy datapath controller demonstrates few key things that we still *can* do. First, we can create more generic forms of synchronizer circuits that are neither flip-flop chains nor Gray counters. In these designs, the propagation of metastable states can be restricted to specific chains of flip-flops. Second, these generic synchronizers can be state machines that complete a number of state transitions in sequence. There remains a non-deterministic uncertainty of 1 cycle in how long these transitions take but they are nonetheless guaranteed to be completed *eventually*.

References

1. Michael Mendler and Terry Stroup. Newtonian arbiters cannot be proven correct. *Formal Methods in System Design*, 3(3):233–257, 1993.
2. Ghaith Tarawneh, Alex Yakovlev, and Terrence Mak. Eliminating synchronization latency using sequenced latching. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 22(2):408–419, 2014.

Quantitative modelling of asynchronous variables

Fei Xia and Ian Clark

School of EEE, Newcastle University, NE1 7RU, UK
fei.xia@newcastle.ac.uk, ian.clark@newcastle.ac.uk

Abstract. Variables being passed between processes not synchronized for the communication may be affected by the lack of synchrony between the processes and such passing of variables may also affect the nominal asynchrony between the communicating processes. There exists a large body of research on the data communication between asynchronous processes exemplified by Lamport's atomic registers and Simpson's multi-slot asynchronous communication mechanisms (ACMs). Many of the existing solutions try to reduce the effects of the fundamental problem by reducing the timing independence to variables of very small size. For instance, Boolean and ternary control variables have been used to protect the usually larger data structures being passed. However, ultimately, the control variables must deal with the asynchrony between the communicating processes in some way. A Boolean variable (single bit) between an asynchronous reader-writer pair cannot avoid metastability or mutual exclusion protection, for instance. Existing models using formalisms such as Petri nets and process algebra and solutions based on state-space analysis provide a very good understanding in the qualitative behavior of such variables. In this paper we aim to expand this understanding to the quantitative by developing models in stochastic activity networks (SANs) with which quantitative investigations may be made with regard to such variables.

Keywords: asynchronous data communication, metastability, stochastic activity networks.

1 Introduction

In digital systems, before the entire world's systems can be synchronized on the same truly global clock, inevitably it would be necessary to communicate outside a particular clock domain. With continued increase of VLSI integration, the physical size of clock domains have become smaller, not larger, and the overall number of clock domains has also increased. For instance, whereas it used to be that everything on the motherboard of a desktop computer ran off a single global clock, now within a single chip there tends to be multiple clock domains as a single chip packs more computation power than multiple classical computers.

Crossing clock domain boundaries with data can be implemented in many ways. It can be done fully synchronously, through the temporary synchronization between two clock domains for the duration of data transfer. This can be found in many globally asynchronous locally synchronous (GALS) solutions where stretchable and/or pausi-

ble clocks are used [1]. It can also be done with a certain degree of asynchrony between the communicating processes, for instance through the use of data buffers. This can be found in the majority of network communications including networks on chip (NoC) solutions [2].

One of the ultimate examples of asynchronous data communication is the so-called fully asynchronous communication where there is no synchronization either actively administered (e.g. the use of GALS-style synchronization) or implied (e.g. through the buffer full or empty states, or the mutual exclusion/critical section protection of data), which ideally allows the processes to possess full temporal independence not affected by the act of communication. Lamport's atomic register followed by Simpson's multi-slot ACMs attempt to solve this problem [3, 4].

From this wide spectrum of problem statements and solutions, it is clear that there exist two fundamental desirable properties. These are:

- Asynchrony: Minimal obliged waiting for either the reader or the writer processes. Fully asynchronous communication aims for zero waiting on either side.
- Data transfer: Maximal quality for the data eventually read. This is usually described by a number of metric parameters, such as data coherence, data freshness, data sequencing, etc. and is different for different application scenarios. An intuitive understanding of data coherence, for instance, is that the writer, or anything else, should not be allowed to corrupt half-read data.

And the large number of existing solutions arrive at various trade-off points between these two qualities [8].

A substantial amount of research exists in this field, with a large number of attempts at provide qualitative modelling so that a solution may be tested for whether it violates data coherence, process asynchrony or any other metric and if so under what circumstances [8].

However there has been a total absence of any quantitative modelling method with which different solutions may be more precisely placed relative to each other in a quantitative map of trade-off.

1.1 Contributions and organization

This work is the first attempt at achieving quantitative models of asynchronous data communication. The language chosen is SANs, which provides opportunities of properly representing such phenomena as metastability quantitatively according to well accepted models [9]. Given that many solutions remove the problems caused by inter-process asynchrony away from potentially large data to usually small control variables, this work concentrate on modelling Boolean and ternary variables and their usual implementation using hardware latches.

The rest of the paper is organized as follows: Section 2 introduces the concept of the asynchronous variable, and describes quantitative models for the two essential properties for Boolean asynchronous variables. Section 3 describes more complex asynchronous variables, their implementation and modelling. Section 4 describes case

studies where the models of asynchronous control variables are used to derive behaviors of larger systems in which they are used. Section 5 concludes the paper.

2 The Boolean asynchronous variable

Fig. 1 shows the basic concept of two asynchronous processes intercommunicating with one (writer) providing the data and the other (reader) making use of it. This is both a general description of all such data communications and a specific description of the passing of control variables. The difference is in what the data is and how it is meant to be used.

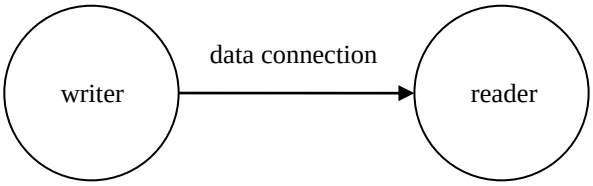


Fig. 1 Unidirectional inter-process data communication.

In a control variable situation, it may happen that the writer of the overall data communication may be the reader of a specific control variable, which is written by the reader process in the overall communication. In other words, although a specific instance of asynchronous data communication is usually defined as unidirectional, i.e. data passing from the writer to the reader, to support this communication some of the control variables may go the other way, to allow the reader to inform the writer of its current state, for instance. In the rest of the paper, unless otherwise stated, 'reader' and 'writer' pertain to the variable being discussed, which in most cases are binary and ternary control variables and not the main data.

It is clear from Fig. 1 that we assume each individual control variable to be unidirectional and cannot be written to by both sides. This simplifies the problem without limiting the solution space, as demonstrated by numerous existing work.

The smallest control variable is the smallest digital variable, i.e. a single binary bit. This is the subject of this section.

Such a variable can be transmitted from the writer to the reader in a number of ways:

(a) Fully synchronously: The reader and the writer need to be synchronized for a single clock period during which the reader directly reads a copy of this Boolean value from the writer's bus or output port. There needs to be no shared memory, just shared wires [5].
(b) With a single shared memory location to provide some degree of asynchrony: The single space FIFO buffer may be guarded with a MUTEX making it accessible by one process at any time [5].

(c) With an unguarded single space FIFO buffer: A fully asynchronous solution allows buffer access by both sides at the same time [4, 8].

All three methods face the following two issues:

- Metastability: The phenomenon of metastability, where a nominally Boolean signal takes a value that is neither 0 nor 1 which nevertheless may persevere for non-trivial amounts of time, is inevitable when you have two independently timed processes accessing the same memory element at the same time in certain conflicting ways, such as reading and writing at the same time or making requests to a MUTEX element at the same time [6]. The simplest 1-bit memory is a latch and synchronizers, MUTEXes and Boolean variables are implemented using circuits which could be classified as some type of latch. The first method therefore cannot avoid metastability at the synchronizer, the second method must face it at the MUTEX, and the third directly on the data bit.
- The relative timing of access from both sides is not specified. Reading may take the same time as, or a radically different time from, writing. And this potentially has an impact on the behaviors of all three methods.

The effects of these challenges on the different methods may be reasoned about qualitatively using existing research results and techniques [7]. For example, even though the fully asynchronous solution may sound unsafe because the metastability is on the data and not controlled by a MUTEX, or mitigated by multi-flop synchronizers, for a lot of control variables used in asynchronous data communications this causes, in practice, some non-deterministic delay [8]. Since the variable being communicated is a binary bit, the worst case scenario, i.e. the reader and the writer both accessing it at the same time, is that metastability may happen. Pragmatically, it is sensible to assume that once metastability happens, the variable eventually settles non-deterministically to one of the digital values: either 0 or 1. Metastability can only happen if the writer is in the process of changing the value of the bit when the reader attempts to access it. Hence either one of the settled values should be valid, for any sensible communication algorithm and implementation. The only thing the designer need to do is to make sure that the control variable is used after some time of its reading to provide it with enough probability to settle before use, as determined by the mean time between failure (MTBF) requirement of the design.

However, when a designer is making a decision on choosing one method over another, a quantitative exploration may be desirable in addition to qualitative considerations.

In this section, we develop quantitative models for both challenges, metastability and independent timing of reader and writer processes.

2.1 Quantitative modelling of metastability in a Boolean variable

To study the metastability behavior of a binary bit being passed from one asynchronous process to another, we assume it is implemented in the way described by Fig. 2. This is the passing of a binary variable from the writer to the reader, such that

the reader's input variable *y* takes on the value of the writer's output variable *x* when the clock/control signal *cl* is set. In other words, *cl*: *y*=*x*.

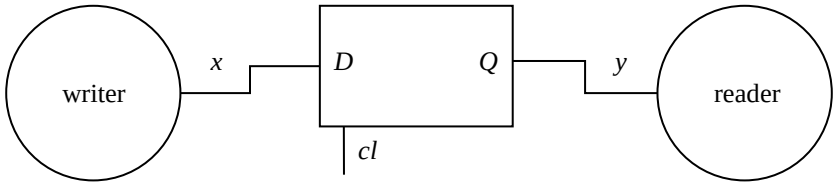


Fig. 2 A binary asynchronous variable.

In order to have metastability in a latch of this type, the clock or control signal *cl* is usually activated by some entity not temporally related to the writer, in other words, the signals *x* and *cl* may change very close in time causing metastability to happen at signal *y*. This directly corresponds with method (c) described above as a latch like this forms the unguarded FIFO buffer used in that method. On the other hand, since synchronizers are constructed out of essentially the same kind of circuit with the same metastability behavior, we can describe the metastability encountered by method (a) using the same technique.

A binary variable that may become metastable, and settling out of metastability

Representing the metastable value of a nominally Boolean variable as a distinct marking allows the convenient tracking of metastability and its effects. The SAN model of a Boolean variable that may become metastable is shown in Fig. 3.

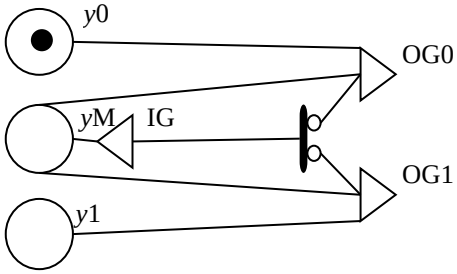


Fig. 3 SAN model of a Boolean variable *y* that may become metastable, including the process of metastability settling.

The process of the natural settling of metastability is usually regarded as stochastic with an average speed entirely dependent on the hardware implementation of the variable. This can be represented by a timed transition whose firing takes the value of the variable to 0 or 1 based on pre-determined probabilities. Established theory on metastability describes the settling as following an exponential process [5]. This can be represented by the timed transition having an exponential timing distribution whose mean rate λ can be found through hardware experimentation [5]. It is usual practice to

assume that metastability settles to 0 or 1 with the same probability, i.e. 50%. However, the model in Fig. 3 allows arbitrary pairs of probabilities p_0=1-p_1 to be chosen, if experiments on hardware show a bias in one way or the other.

Coincidentally, for the purpose of using this model for analysis, exponential timing in the timed transition means that this part of the model does not introduce anything non-Markovian. An entirely Markovian system model usually allows not only simulations but also analytical reasoning [9].

The logic of the input and output gates in the model is defined as follows:

Gate	Predicate	Function
IG	`Mark(yM)==1`	

OG0		`Mark(yM)=0;`
		`if (Mark(y0)==0 and Mark(y1)==0)`
		`then Mark(y0)=1;`

OG1		`Mark(yM)=0;`
		`if (Mark(y0)==0 and Mark(y1)==0)`
		`then Mark(y0)=1;`

The settling countdown starts immediately when the variable enters metastability. At the end of the model-determined settling time, the marking in place *y*M is set to zero. However, when updating the variable to a digital value, the model needs to determine whether the settling process is at this moment still in charge of the value of the variable – it is entirely possible that during the expected duration of metastability settling time, when the settling transition is in the process of firing, the variable has otherwise been set to a secure digital value through other means such as having been successfully assigned a value by another operation. The functions of the output gates ensure that only when no such thing has happened (i.e. both digital places still have the marking of 0) the completion of the settling transition would set the expected digital value. Otherwise nothing is done as at the end of the expected settling time, the variable has already otherwise achieved a secure digital value.

Actively changing the variable value

In addition to the settling of metastability, which is a passive process, the value of a nominally Boolean variable may also be changed actively. An example of this is the setting of variable *y* to the value of variable *x* in Fig. 2.

To model metastability and its effect fully, we need to consider the following situations:

- When *cl* comes, *x* is stable at either 0 or 1 → *y* takes the value of *x*
- When *cl* comes, *x* is itself metastable, i.e. having the value of M → *y* has a probability of becoming M;
- When *cl* comes, *x* is being changed → *y* has a probability of becoming M;

This means that we need a place or places whose marking(s) indicate that *x* is being changed between the two digital values 0 and 1. In addition, *cl* itself is a signal

whose change may take some time. This is best represented by having any *cl* change indicated by a marking in a place.

The SAN formalism facilitates the compact representation of such conditional relationships, once such states as '*x* is being changed' and '*cl* is coming' are represented by markings. Similar to the metastability settling speed, the probabilities of *y* getting a value of 0, M or 1 in any particular situation may be obtained through hardware experiments. Hardware characterization is the best method for generating the quantitative parameters for these models.

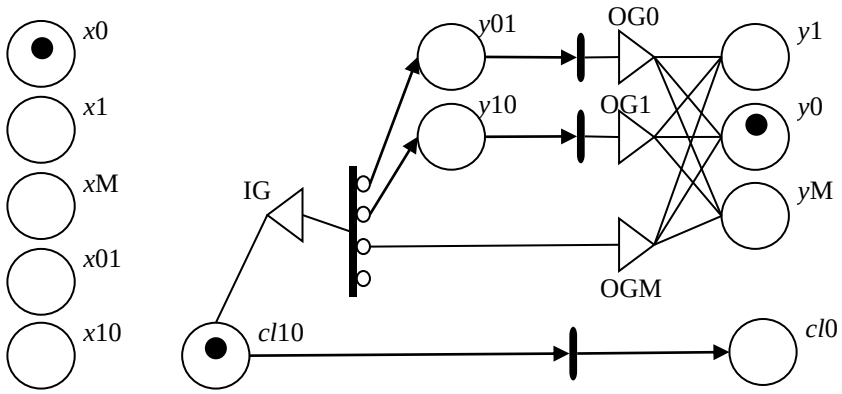


Fig. 4 Changing the value of an asynchronous Boolean variable.

Fig. 4 shows the structure of the SAN model of changing the value of a Boolean variable implemented using a circuit of the type shown in Fig. 2. Places named *x*0, *y*M, *cl*0, etc. denote that the signal takes a particular value. Places named with the signal name followed by two values denote that the signal is transitioning from the first value to the second, i.e. *x*01 denotes *x*: 0→1.

The logic of the input and output gates is as follows:

Gate	**Predicate**	**Function**
IG	`Mark(cl10)==1`	

OG0		`Mark(y=M)=0;`
		`Mark(y=0)=0;`
		`Mark(y=1)=1;`

OG1		`Mark(y=M)=0;`
		`Mark(y=1)=0;`
		`Mark(y=0)=1;`

OGM		`Mark(y=0)=0;`
		`Mark(y=1)=0;`
		`Mark(y=M)=1;`

The process of changing the value of *y* starts when *cl* is being reset (falling edge trigger on the clock, place *cl*10 marked). At the end of the process, the value of *y* is

set according to what branch of the SAN the model has been progressing. The first transition on the left hand side is where the logic is that determines how the value of *y* will be set, and as such must correctly specify the probabilities of each of its branches, changing from 0 to 1, changing from 1 to 0, setting to M, and do nothing (keeping the old value of *y*).

This depends on the markings of the places listed on the left hand side of Fig. 4.

For instance, case 4 of the activity, do nothing, has a probability of 1 when x=0 and y=0, and a probability of 0 when x=0 and y=1. The probability of case 3 is non-zero if one of the places *x*M, *x*01 and *x*10 are marked. The probability of case 3 under different conditions when it is not zero can be determined through hardware characterization experiments [5].

This model has a relatively low precision as it assumes a constant probability of *y* entering metastability if, when *cl* changes, *x* is in the process of change. However, accepted theory of metastability indicates that the probability of *y* entering metastability is related to how close the *cl* and *x* changes are in time [5]. Assuming the same probability for all cases of overlapping access may not be precise enough for certainly analysis.

Timing issues

The precision of the above model can be improved by representing one of the changing processes, either *x* changing or *cl* falling, as constituting multiple steps. This extended representation is shown in Fig. 5.

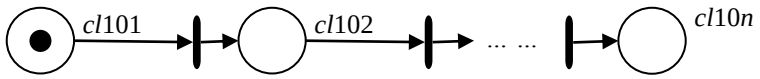


Fig. 5 Dividing a value change into multiple steps.

During a change of the signal *cl*, how much this change has progressed is indicated by where the token is in the chain. Usually variable value changes take deterministic time related to the speed of the hardware in which the process that makes this change is mapped. Such a deterministic delay can easily be divided into a sequence of deterministic delays as represented by the timed transitions in Fig. 5. For stochastic delays, this method is less effective and accurate unless the distributions of the sequence of smaller delays can be derived from the distribution of the overall delay, which is not always possible. In other words, if metastability is involved in any signal value change, this method of dividing into multiple smaller changes may be less effective and using a single probability as in the previous section may be unavoidable.

Knowing where a signal is in its changing process when the other signal is also changing allows the distance in time between these two changes to be represented, up to the precision of the stages in the model shown in Fig. 5. The model can therefore be made as precise as the time resolution of the hardware characterization data.

The other timing issue, that of the relative durations of time each of the reader and writer processes makes accesses to a shared resource, is automatically represented in

models developed according to the methods given in this section. Together, the metastability settling time, rising time, and falling time of each signal fully describe overall process lengths such as how long a shared variable is accessed by the writer or the reader.

2.2 MUTEX and arbitration

For method (b), which protects the shared memory location with a mutual exclusion arrangement avoiding simultaneous accesses by both the reader and writer. The metastability and timing modelling can be derived based on the models presented in Section 2.1. This is because MUTEXes are usually constructed out of similar circuits, i.e. a single bit memory. However, since the input signals to MUTEXes are not the somewhat asymmetric data and clock with different functions, but fully symmetric requests, the models need to be modified to reflect this.

A typical MUTEX consists of an SR-latch followed by a metastability resolver. It functions as follows:

- Be ready to receive requests from two different processes when no grant is outstanding;
- Issue a grand to the process which has just produced a request;
- Withdraw a grant after receiving a reset of a request signal – a requesting process is assumed to reset its request once the granted resource has been made use of;
- When requests from both processes arrive close together, the latch may go into metastability, but the metastability resolver makes sure that no grants will be issued until the metastability has been resolved;
- A requesting process is assumed to hold up its request until a grant is issued – there is no withdrawal of requests without grants.

The model for the metastable state and its settlement is similar to Fig. 3, if a Boolean variable is used to issue grants, for instance y=0 grants process 0 and y=1 grants process 1, as is normal for MUTEX arbitration.

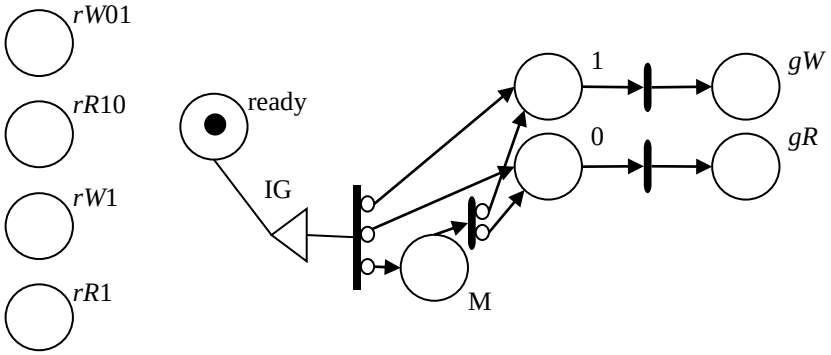


Fig. 6 Responding to requests

The model in Fig. 6 is based on the assumption that the internal MUTEX state of 1 corresponds to granting to writer and 0 corresponds to granting to reader, without losing generality. Once the internal state 0 or 1 is secured, the corresponding grant may take non-zero time to appear, represented by the timed transitions on the right hand side. If this degree of representation is not needed, these may be replaced with instantaneous transitions. The internal state M causes the entire process to delay with the only activity being the metastability settling. Note that this settling sub-net is less complex than Fig. 3 as in a MUTEX it is not possible for external influences to hard set M to one of the digital states. Because requesters would not withdraw outstanding requests, the only route out of the M state is via settlement.

The internal state is only represented between the start of a grant computation (the left hand transition firing) and the next grant commitment (one of the right hand transitions starting to fire). Although the actual circuit would maintain an internal state always, it is only of functional relevance in between those events. Hence the model is simplified this way.

The input gate IG has the following predicate and function:

Gate	Predicate	Function
IG	`Mark(ready)==1 &&` `(Mark(rW1)==1 || Mark(rR1)==1)`	`Mark(ready)=0`

This means that grant computation starts when the place ‘ready’ is marked and at least one of the request signals is present (*rW*1 and/or *rR*1 marked). And starting a round of grant computation unmarks the place ‘ready’. Place ‘ready’ is marked again once a requesting process has finished using its granted access.

The probabilities in the grant computation starting transition need to reflect the following protocol:

- If neither *rR*1 nor *rR*01 is marked, i.e. the reader request is not present nor being issued, grant computation goes to the branch leading to internal place ‘1’ with a probability of 1. This leads towards granting writer access.
- If neither *rW*1 nor *rW*01 is marked, i.e. the writer request is not present nor being issued, grant computation goes to the branch leading to internal place ‘0’ with a probability of 1. This leads to granting reader access.
- If one of *rR*1/*rR*01 and one of *rW*1/*rW*01 are marked at the same time, i.e. both reader and writer requests are either present or being issued, grant computation goes to all three branches with appropriate probabilities, which may be obtained from experimental characterization of hardware.

Granting is represented by either place *gW* or place *gR*. The appropriate process taking its grant by taking the token from its corresponding grant place, and puts a token back to place ‘ready’ after the end of the access.

The normal assumptions of MUTEX arbitration apply. For instance, a requester is not supposed to withdrawn a request before a grant is issued in its favor. In order to represent all possible causes of metastability properly, request signals are modelled as taking non-zero time to set up (‘being issued’ is at least one distinct state). More pre-

cise modelling by having a multi-stage setup process for one or both of the request signals is possible with the method shown in Fig. 5, if necessary.

3 The ternary asynchronous variable

Larger control variables may be needed for more complex ACMs. Here we discuss the ternary control variable used in certain existing ACM algorithms, to show that larger control variables can be constructed out of Boolean asynchronous variables.

A ternary (base-3) variable taking three possible values (e.g., 0, 1, 2) may be implemented in a number of ways. The most straightforward is to use one-hot encoding and binary circuits. In other words, using three wires, with a maximum of one of then having a signal value of 1 at any time. Wire 0 being 1 and the other two wires being 0 indicate the value of the variable being 0, wire 1 being 1 and the other two wires being 0 indicate the value of the variable being 1, and wire 2 being 1 and the other two wires being 0 indicate the value of the variable being 2:

Signals on wires	**Variable value**
{1, 0, 0}	0
{0, 1, 0}	1
{0, 0, 1}	2
{0, 0, 0}	potential spacer (see below)
other values are not allowed	undefined

When such a variable is being sent from one independently timed process to another, the simplest way of avoiding confusion and reducing potential errors is to make use of spacers. In other words, when changing the value of such a variable, the current wire holding 1 is pulled down to 0 first, before another wire is set from 0 to 1. Briefly, the three wires hold the spacer signal {0, 0, 0}, which indicates the fact that the value of the ternary variable is being changed.

For instance, the following steps change the variable value from 0 to 2:

$\{1, 0, 0\} \rightarrow \{0, 0, 0\} \rightarrow \{0, 0, 1\}$

With such an implementation, or any other implementation using binary logic, the models derived in Section 2 can be directly used, as each ternary variable is implemented with binary signals.

It is worth noting that when reading the value of such a ternary variable, a spacer should cause a wait on the reader's part as the value is not functionally valid. In other words, reading a ternary variable should be done using the following procedure:

```
wait until (v0=1 or v1=1 or v2=1);
read v;
```

In such a scheme, metastability can only happen when changing to spacer or changing out of spacer. The spacer scheme qualitatively trades spacer delay for additional variable value safety.

Ternary logic based on using three different analogue signal values to represent the three variable values is conceptually possible. It is worth noting that these circuits have been ignored in general by researchers in the field of asynchronous data communication. Modelling such implementations is out of the scope of this paper.

4 Case study

In this section we investigate the usage of the asynchronous variable models in a wider context, i.e. when these variables are used as control variables in ACMs. Here the examples used are the 'pool' or 'RR-OW' type, with a single logical buffer space, mechanisms that aim to provide the reading and writing processes with full timing independence. This means that overwriting of previously unread data stored in the buffer, and rereading of previously read data, must be allowed. This is intuitive if the buffer has a limited number of spaces (in this case a single space), and potentially allow multiple cycles of reader access in between two writer accesses and multiple writer accesses in between two reader accesses.

The fundamental assumption of these ACMs is that they serve as data connectors between the communicating processes between which a sequence of data items of the same type are transferred. In each cycle of writing and reading access, the relevant communicating process transfers one item of data to or from the ACM.

4.1 A two-slot ACM

The two-slot pool ACM, proposed by Simpson in [4], attempts to accommodate the full asynchrony between the reading and writing accesses by using two physical data memory spaces, each enough to contain one item of data. When an access happens, the other access should maximally be able to occupy one slot, hence the intuition is that whenever an access needs to happen it always has a slot to point to. Control variables are used to make sure that reading accesses 'chase' writing accesses – each reading access tries to read from the slot containing the newest completely written data item, as indicated by the writing access's previous round. How the writing access chooses its slot can be more interesting. A seemingly totally safe method is to always avoid the reading access by going to the other slot not currently being read. This however has been shown to create the possibility that the reading access will keep rereading the same item of data whilst the writing access keeps overwriting to the other slot, if the two accesses are matched in their speed. Simpson's method is to have the writing access point to alternating slots in successive rounds regardless of where the reading is happening. This has been shown to lead to clashes on the same slot by both processes and violate data coherence, under certain circumstances.

The algorithm for the two-slot mechanism can be written as follows:

Write access	**Read access**
`w=!(l);`	`r=l;`
`d[w]=input;`	`Output=d[r];`
`l=w;`	

In this algorithm the two data slots are arranged into an array *d*[0..1], whose access is managed through the shared control variable *l*, which is Boolean and indicates the last complete written slot. The writing access uses its private control variable *w* to choose the next slot to be written. The reading access uses its private control variable *r* to choose the next slot to be read.

In the writing access the sequence of actions are: choosing the slot not accessed by the previous writing access, writing to that slot, indicating that slot to be the last written one. It is then assumed that the writing process ends and the master process that contains the writing process will execute a sequence of actions which includes the preparation of the next writing access round. Then the writing access will start again from the first statement. This is assumed to form a forever loop as long as the ACM continues to be used.

The reading access has the following sequence of actions: choosing the slot to read from, read from that slot. After a round of reading access ends, the master process that includes the reading access is assumed to perform actions not related to accessing this ACM, including making use of the data just read. A forever loop situation similar to the writing side is also assumed.

The ACM accesses can then be viewed as procedures or functions called by their respective master processes in each cycle of action.

A single Boolean control variable is shared between the two access procedures, *l*, which is written by the writing access and read by the reading access. It is used for the writing side to indicate to the reading side, and to itself, the immediate previous completely written slot. The passing of this Boolean variable through the reader statement `r=l` can be modelled with the method described in Section 2.

Here we investigate the mode of operation where the two-slot ACM works if *l* is always correctly read. This is guaranteed if the duration between two writing accesses is longer than a reading access. If this is not the case, the two-slot ACM can violate data coherence even if *l* is always correctly read and there is not much point in analyzing what happens when, for instance, *r* becomes metastable.

As the value of *l* itself is changed entirely during a writer internal statement unrelated to the reader, it cannot be metastable. As a result the model for *r* and the synchronizer is simpler in methods (a) and (c) as there is no such thing as metastability propagation between the two sides. For method (b), the MUTEX protects variable *l*, and both `l=w;` and `r=l;` need to be preceded by requests to the MUTEX and followed by request resets. The writer example is as follows:

```
rW=1;
l=w;
rW=0;
```

4.2 Quantitative model explorations

Models in SANs were constructed within the environment of the Möbius tool [10] and quantitative explorations of the behavior of the two-slot ACM studied in the same environment. In this first explorative attempt, the following assumptions are made:

- The smallest step, that of assigning the value of a Boolean variable, is set to unit time, called τ – this usually corresponds with somewhere in the picosecond range in current CMOS technology given typical latches;
- Both reader and writer processes have deterministic delays in their statements, to emulate real-time programs whose timings have deterministic specifications;
- The reader process is started after a stochastic delay after the writer process, to emulate non-deterministic phase differences between the two processes, the resolution of this phase difference is 0.1τ so that processes can by desynchronized by less time than a full Boolean variable value change;
- Test cases with both `l=w;` and `r=l;` taking τ, with `l=w;` taking τ and `r=l;` taking 10τ, as well as with `l=w;` taking 10τ and `r=l;` taking τ are explored;
- Reading is assumed to take 10 times the time as a reader binary variable statement and writing is assumed to take 10 times the time as a writer binary variable statement;
- The time distance between two write accesses and that between two read accesses is assumed to take 1000 times the time as a binary variable statement.

Basically we cover the cases where the writer is 10 times faster than, the same speed as, and 10 times slower than the reader. The only non-deterministic delays in the study comes from metastability settlement. We also tried one case where the writer's speed is related to τ, and hence that of the reader, by a random non-integer value, but that did not show up any new results or trends.

We only explored methods (b) – MUTEX protection for *l* and (c) – unprotected fully asynchronous access to *l* by both sides.

The quantitative results we obtained, collated from a number of experiments, are listed as follows:

Method	**Data error min**	**Data error mean**	**Data error max**	**Delay min**	**Delay mean**	**Delay max**
(b)	0	0	0	0	0.013τ	∞[1]
(c)	0	0.025%	100%[2]	0	0	0

These explorations are based on realistic assumptions of very low probabilities of the onset of metastability even with overlapping accesses (e.g. 1%) and fast metastability settlement (e.g. the mean settlement time $1/\lambda = 0.1\tau$). Data error is measured by how many reads produced output values that have not been written, and 'delay' in the above results relate to additional delay either side has to suffer because the two state-

[1] These are theoretical values derived from [5].

ments `l=w;` and `r=l;` took longer than normal time due to the asynchrony. The results confirm the intuitive notion that the two methods trade delay with correctness as the MUTEX method protects the variable by paying potential non-deterministic delay as a price, and the fully asynchronous method guarantees full delay predictability by paying potential data corruption. Although these qualitative points can be derived from existing methods this work provides a systematic way of generating quantitative trade-off maps for designers.

5 Conclusions and future work

Quantitative models of asynchronous variables including metastability and its settlement are developed using the formalism stochastic activity networks (SANs) and their use initially demonstrated through a case study conducted using the Möbius tool. Whilst the actual numbers obtained from the work so far may not have any practical significance, the fact that they can be obtained using the methods provided represents a new development in the modelling of variables being passed between two non-fully synchronized processes.

The models can be extended and used on the entire class of existing ACM solutions and the hypothesis that the method may be used for developing new ACMs are promising topics of future work.

References

1. M. Krstic, E. Grass, F. Gürkaynak, P. Vivet, "Globally asynchronous, locally synchronous circuits: overview and outlook," in IEEE Design & Test of Computers, 24,(5), pp.430-441, 2007.
2. L. Benini, G. De Micheli, "Networks on chips: a new SoC paradigm," in Computer, 35,(1), pp. 70-78, 2002.
3. Lamport, L., 'On interprocess communication: Parts I and II', Distrib. Comput., 1,(2), pp.77-101 1986.
4. Simpson, R., "Four-slot fully asynchronous communication mechanism", IEE Proceedings, Pt. E, 137,(1), pp.17-30, 1990.
5. D. Kinnement, Synchronization and arbitration in digital systems, Wiley, 2007.
6. Kleeman, L., Cantoni, A., "On the unavoidability of metastable behavior in digital systems", IEEE Trans. Comput, 36,(1), pp.109-112, 1987.
7. Clark, I., Xia, F., Yakovlev, Y., Davies, A., 'Petri net models of latch metastability', Electronics Letters 34,(7), pp.635–636, 1998.
8. Xia, F, Yakovlev, A., Clark, I., Shang. D., 'Data communication in systems with heterogeneous timing', IEEE Micro, 22,(6), pp. 58-69, 2002.
9. Sanders, W. and Meyer J., 'Stochastic activity networks: formal definitions and concepts', LNCS 2090, pp.315-343, 2000.
10. The Möbius tool, available at: https://www.mobius.illinois.edu/.

Working with Petri Nets and Asynchronous Circuits

Tomohiro Yoneda

National Institute of Informatics, Tokyo 101-8430, Japan

Abstract. Petri nets and asynchronous circuits are very important and special topics in my research career. Petri nets have been a useful tool for me to formally verify and synthesize asynchronous circuits. In this paper, I especially focus on handling timing issues in such frameworks, and would like to summarize my research experiences built on both time Petri nets and timed asynchronous circuits.

1 Time Petri nets and timed circuits

A *time Petri net* [1] is one of timed extensions of a Petri net where timing constraints are given to their transition firings. Two nonnegative rational numbers are assigned to each transition of a time Petri net as shown in Fig. 1. In this example, suppose that a transition t_0 becomes enabled (*i.e.*, a marking where both its input places p_0 and p_1 have tokens is reached) at time point T. Then, t_0 cannot fire before time $T + a$, and t_0 must fire before or at time $T + b$.

These timing constraints given to transitions of time Petri nets are useful to model a *timed circuit*, where each gate in it has bounded delays, as shown in Fig. 2 (a). In this paper, it is considered that the timed gate shown in Fig. 2 (a) is equivalent to a gate with an inertial delay element as shown in Fig. 2 (b). The two parameters of the delay element show its lower and upper bounds of the delay, that is, the actual delay of the gate is non-deterministic, but is always within the range. Furthermore, since it is an inertial delay element, a pulse whose width is shorter than a always disappears (*i.e.*, it cannot go through the delay element). A wider pulse may or may not disappear, if its width is shorter than or equal to b.

The timed gate shown in Fig. 2 can be modeled by a time Petri net as shown in Fig. 3. This time Petri net is extended in several points: (1) Each transition

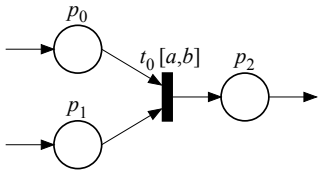


Fig. 1. A time Petri net.

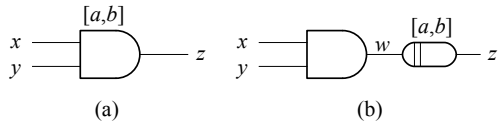


Fig. 2. A timed gate.

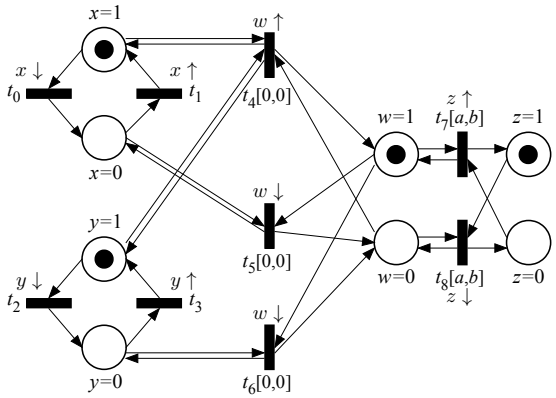


Fig. 3. A time Petri net representing the timed gate shown in Fig. 2.

is labeled with a rising or falling change of a signal. The firing of a transition represents the change of the corresponding signal. (2) Each place is labeled with a value of a signal. If a place has a token, the corresponding signal has the value at the place. (3) Some transitions have no timing constraints (*e.g.*, t_0, t_1, t_2, and t_3). These transitions are called *input transitions*. The remaining transitions are called *output transitions*. When a set of time Petri nets representing gates or modules are considered, the firings of input transitions are invoked in synchronization with the firings of the corresponding output transitions.

2 Partial order reduction

State space enumeration of a formal model is an essential technique needed for both formal verification and asynchronous circuit synthesis. However, it is not easy to complete the state space enumeration of a system unless the system is very small. Thus, we focused on the fact that our purpose can be achieved by checking whether some specific properties hold or not. One such simple example of the properties is whether a specific transition can fire in some reachable state. This property can easily be checked, if the whole reachable state space is examined, but it is also possible to check it in a reduced state space. The partial order reduction is a technique to obtain such a reduced state space for the given property. Stubborn set method [2] is one of such techniques, and we extended similar ideas for time Petri nets [3, 4].

Let's consider a time Petri net shown in Fig. 4 (a), and assume that we want to check whether transition t_5 can fire in some reachable state. After firing t_0, t_1 can fire concurrently with t_3 and t_4 from their timing constraints. Two transitions t_2 and t_5 are *in conflict*, which means that firing one of those transitions disables the other. After firing t_1, t_2 becomes enabled. Again, from the timing constraints, t_2 can fire at the state, and its firing disables t_5. Similar situation happens for

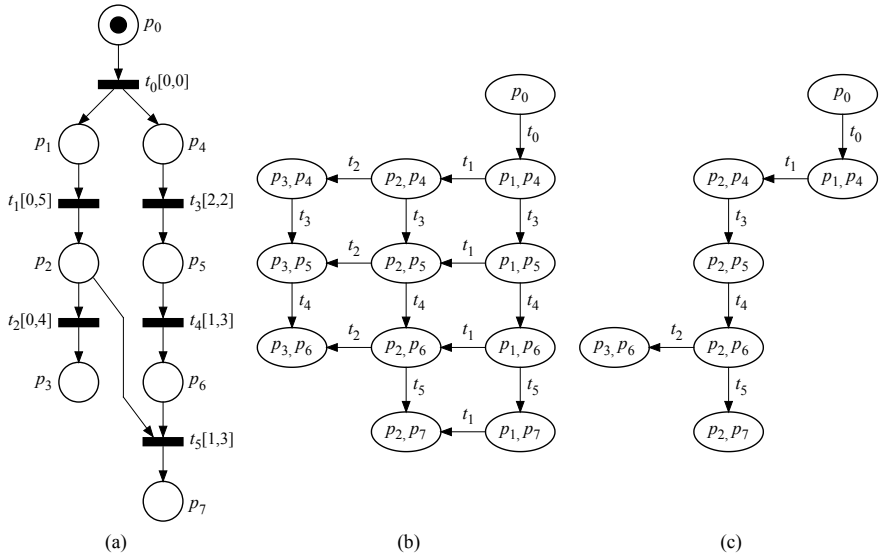


Fig. 4. (a) A time Petri net, (b) its full state space, and (c) its reduced state space.

t_5. Hence, we have a reachable state graph as shown in Fig. 4 (b). This graph is obtained by firing every transition that can fire in each state. This is the full state enumeration.

In order to check the above property without constructing the full state graph, we usually fire one enabled and *firable* transition in each state. Intuitively, an enabled transition is firable, if it can fire with respect to its timing constraints. After firing t_0, both t_1 and t_3 satisfy this condition. Since we choose one such transition arbitrarily, assume that t_1 is chosen. After firing t_1, t_2 and t_3 are such transitions. This state is special, because firing t_2 takes away the possibility to fire t_5. Thus, firing only t_2 in this state may lead to an incorrect decision. Instead, enabled and firable transitions that may eventually make t_5 enabled in time are searched. In this case, it is t_3. Our algorithm requires that t_3 should also be fired when t_2 is fired from this state. On the other hand, t_3 is an enabled and firable transition in this state, and it conflicts with no other transitions. Thus, when t_3 is fired in this state, it is not required to fire any other transitions. For this reason, our algorithm chooses to fire t_3 in this state. t_4 is fired similarly, and then, both t_2 and t_5 become enabled. In this state, there is no other option except for firing t_2 or t_5, and the reduced state space shown in Fig. 4 (c) is obtained.

A key idea of our algorithm is that in case where firing a transition t_a is considered and a disabled transition t_b is in conflict with t_a, enabled and firable transitions that may eventually make t_b enabled in time are searched. Firing such transitions (as well as t_a) prevents us from missing possible transition firings. By the way, the above "in time" is also important. For example, if the

timing constraints of t_3 is $[8, 8]$ instead of $[2, 2]$, t_3 cannot make t_5 enabled in time, because t_2 must fire before time $T(t_0) + 9$ (let $T(u)$ denote the time when transition u fires).

I mainly worked with Holger Schlingloff for this research topic. We both stayed at Carnegie Mellon University in 1990-1991 as visiting researchers of Prof. E. M. Clarke. We published 7 co-authored papers. An incomplete list of researchers with whom I communicated on this topic is B. Berthomieu, P. Godefroid, K. L. McMillan, D. Peled, and A. Valmari.

3 Formal verification of timed asynchronous circuits

Our framework of formal verification of timed asynchronous circuits is a conformance checking. In this framework, a *safety failure* for given specification and circuit is a situation where a possible behavior in either side (specification or circuit) is not possible in the other side. This situation usually happens when the circuit produces a bad output that is not specified in the specification. For this purpose, the given specification is first modeled by a time Petri net. This time Petri net specifies when the input signals of the circuit can change, with the timing constrains given to its output transitions. It also specifies when the output signals of the circuit can change, with its input transitions. For example, Fig. 5 (b) is a specification of a C-element in Fig. 5 (a). Note that transitions labeled with τ are *dummy* transitions that are output transitions, but have no corresponding input transitions.

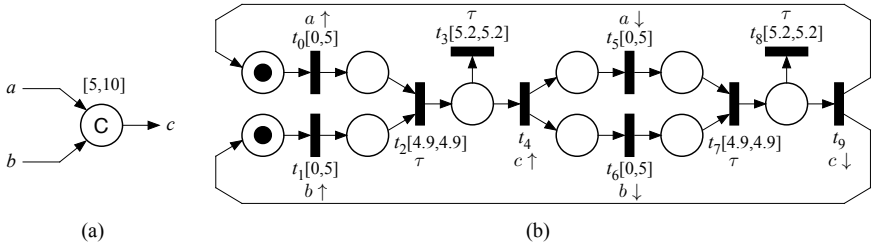


Fig. 5. (a) a circuit, and (b) its specification.

A circuit is modeled by a set of time Petri nets. As shown in Fig. 3, a simple gate is straightforwardly modeled by a time Petri net. For connecting several gates to compose a circuit, it is only needed to prepare a set of gates. Two important roles of our verification algorithm are (1) to synchronize the firings of corresponding input and output transitions to explore the state space of the system (*i.e.*, a set of time Petri nets for a specification and gates), and (2) to check whether for any firable output transition there exists a corresponding enabled input transition. By the role (1), when an output of a gate is connected to an input of another gate, its connection is represented by firing those input and

output transitions in synchronization with each other. The role (1) also works for specifying how a circuit should be used with respect to the specification. Remember that the output transitions of the specification decide how the corresponding input signals of a circuit behave, by synchronizing the corresponding output transitions in a specification and the input transitions in a circuit. That is, the specification stimulates the circuit. The role (2) is exactly for detecting the safety failures.

As mentioned in the previous section, it is not usually possible to explore the full state space. Thus, we have applied the partial order reduction for the conformance checking. For this purpose, we need to consider a property for the partial order reduction, such that a safety failed state is reachable in the reduced state space if and only if a safety failed state is reachable in the full state space. Remember that the simple partial order reduction algorithm mentioned in Section 2 searches enabled and firable transitions that make t_b enabled in time, where t_b is in conflict with t_a and t_a is fired at the current state. For handling the conformance checking, this search is extended such that an input transition t_c is considered for the search similarly to t_b, where t_d is an output transition that corresponds to t_c and t_d is made enabled by t_a (see Fig. 6).

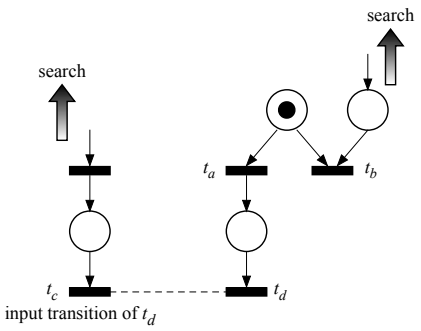


Fig. 6. Extended search of enabled and firable transitions.

I mainly worked with Chris Myers for this research topic. We had a joint travel grant for 3 years, and published 15 co-authored papers related to this topic. An incomplete list of researchers with whom I communicated on this topic is P. A. Beerel, J. Cortadella, D. Dill, K. L. McMillan, and H. Schlingloff.

4 Synthesis of asynchronous circuits

Synthesis of an asynchronous circuit is to obtain an asynchronous circuit M for a given specification G such that M behaves identically to G. In our case, G is given as a timed signal transition graph, which is a kind of a time Petri net. By using timed signal transition graphs for specifications, several behaviors that are

impossible due to the given timing constraints are ignored, and thus, a smaller circuit is expected to be finally synthesized.

If the full state space of G is obtained, and it satisfies several properties, M can be obtained by a standard logic synthesis algorithm for asynchronous circuits. Such logic synthesis algorithm is implemented in several tools. The most popular tool may be Petrify [5]. Again, it is not easy to explore the full state space of the given specifications, especially when those specifications are obtained in a high-level synthesis process.

Our approach to this problem is a decomposition based synthesis [6, 7]. In our approach, for an output x of the given G, some subset V of signals of G and $\mathsf{abs}(G, V, x)$ are first obtained, where $\mathsf{abs}(G, V, x)$ and G have equivalent behaviors with respect to signals in V, and $\mathsf{abs}(G, V, x)$ is synthesizable. Then, a standard logic synthesis algorithm is applied to $\mathsf{abs}(G, V, x)$ instead of G, which generates a sub-circuit for an output x. This process is repeated for each output of G. Usually, the state space of $\mathsf{abs}(G, V, x)$ is much smaller than that of G. Thus, the synthesis time can be dramatically reduced. The construction of V is, however, not trivial. Actually, it can easily be obtained from the full state space of G, but it does not help us. Our key contribution is an algorithm to efficiently obtain V using a technique similar to the partial order reduction for time Petri nets. Note that this approach uses a restricted information through a subset V of signals of G. Thus, the synthesized circuits by our method may not be optimal. According to the experimental results, however, this overhead is pretty small [6, 7].

I worked with Chris Myers also for this research topic. We published 7 co-authored papers related to this topic. An incomplete list of researchers with whom I communicated on this topic is J. Cortadella, V. Khomenko, M. Schaefer, and W. Vogler.

5 Conclusion

In this paper, I summarized my previous work related to time Petri nets. I have chosen such research topics, because I had many chances to talk with Alex Yakovlev through those topics in related conferences such as ASYNC and ACSD. Recently, I have been more interested in designing asynchronous circuits such as asynchronous NoCs. At ACSD conference in 2011, Alex invited me as a key-note speaker, and I was greatly honored to be able to give a talk on asynchronous NoCs there.

References

1. P. Merlin and D. J. Faber. Recoverability of communication protocols. *IEEE Trans. on Communication*, COM-24(9):1036–1043, 1976.
2. A. Valmari. Stubborn sets for reduced state space generation. *LNCS 483 Advances in Petri Nets*, pages 491–515, 1990.

3. T. Yoneda, A. Shibayama, H. Schlingloff, and E. M. Clarke. Efficient verification of parallel real-time systems. *LNCS 697 Computer Aided Verification*, pages 321–332, 1993.
4. T. Yoneda and H. Schlingloff. Efficient verification of parallel real-time systems. *Formal Method in System Design*, pages 187–215, 1997.
5. J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE Transactions on Information and Systems*, E80-D(3):315–325, March 1997.
6. T. Yoneda, H. Onda, and C. Myers. Synthesis of speed independent circuits based on decomposition. *Proc. of 10th International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 135–145, 2004.
7. T. Yoneda and C. J. Myers. Synthesis of timed circuits based on decomposition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(7):1177–1195, July 2007.

Opportunities and Challenges for Ultra-Low Voltage Digital IC Design

Jun Zhou, Scientist / PhD Supervisor
Institute of Microelectronics, A*STAR, Singapore
zhouj@ime.a-star.edu.sg

Graduated in Nov. 2008
from Microelectronics System Design Group, Newcastle University

***Abstract*—Ultra-low voltage digital IC design is promising in achieving ultra-low power consumption for emerging applications such as IoT, smart sensor and wearable computing. This paper discusses the opportunities and challenges of ultra-low voltage digital IC design by reviewing and discussing the major design techniques for enabling ultra-low voltage operation, including ultra-low voltage device sizing, ultra-low voltage level shifter design, ultra-low voltage SRAM design and variation-resilient techniques for ultra-low voltage design.**

1 INTRODUCTION

Emerging applications such as IoT, smart sensor and wearable computing require advanced built-in digital signal processing and controlling capability for "intelligent" and "connected" devices. In the meanwhile, ultra-low power consumption is demanded to prolong the battery life or achieve perpetual operation with energy harvester. This makes the design of ultra-low power digital ICs a must. In the past, ultra-low voltage design has been proved to be able to reduce the power consumption by 5-10×[1][2], making it a very promising candidate for the power-constrained emerging applications. This paper reviews and discusses the major design techniques for the ultra-low voltage digital IC design. In Section 2, the ultra-low voltage device sizing methods are reviewed. In Section 3, the design techniques of ultra-low voltage level shifter are discussed. In Section 4, the design techniques of ultra-low voltage SRAM are discussed. In Section 5, the variation resilience design techniques for ultra-low voltage operation are discussed, and in Section 6, the conclusions are drawn.

2 Ultra-Low Voltage Device Sizing for Logic Gates

Previous studies [2][3] have shown that the logic gates with low fan-in (e.g. fan-in less than 4) usually do not have functionality problems at ultra-low voltage even under

PVT variations. However, in terms of performance they are not optimal, as the device size is optimized for super-threshold operation. For example, the PMOS is usually sized around twice as big as NMOS in order to achieve balanced rise and fall delay. However, in the near/sub-threshold region, the strength ratio of PMOS and NMOS varies due to different current-voltage characteristic, leading to unbalanced rise and fall delay [2][3][4]. Another issue is that the impact of the parasitic effects becomes different in the near/sub-threshold region. For example, the impact of drain-induced barrier lowering (DIBL) become less due to reduced drain-to-source voltage and the impact of reverse short-channel effect (RSCE) becomes stronger due to the exponential dependence of current on threshold voltage. Sub-threshold device sizing methods considering the change of current-voltage characteristic and parasitic effects of have been proposed to achieve optimal performance for the near/sub-threshold operation. In [4], RSCE is utilized to boost the current by using non-minimum transistor length. This also mitigate the impact of process variations due to increased transistor area. In [5], inverse narrow width effect (INWE) is utilized to increase the current by using minimum-size finger. With constant current this results in reduced load capacitance and area. The same device sizing method is combined with minimum sizing in a dual-width sub-threshold standard cell library [6]. The INWE-aware cells are used in critical paths to achieve small delay while the minimum-sizing cells are used in non-critical paths to reduce the power and area. Based on these work, future ultra-low voltage device sizing methods may further explore circuit partitioning and hybrid device sizing method to achieve co-optimized delay, power and area.

3 Ultra-Low Voltage Level Shifter

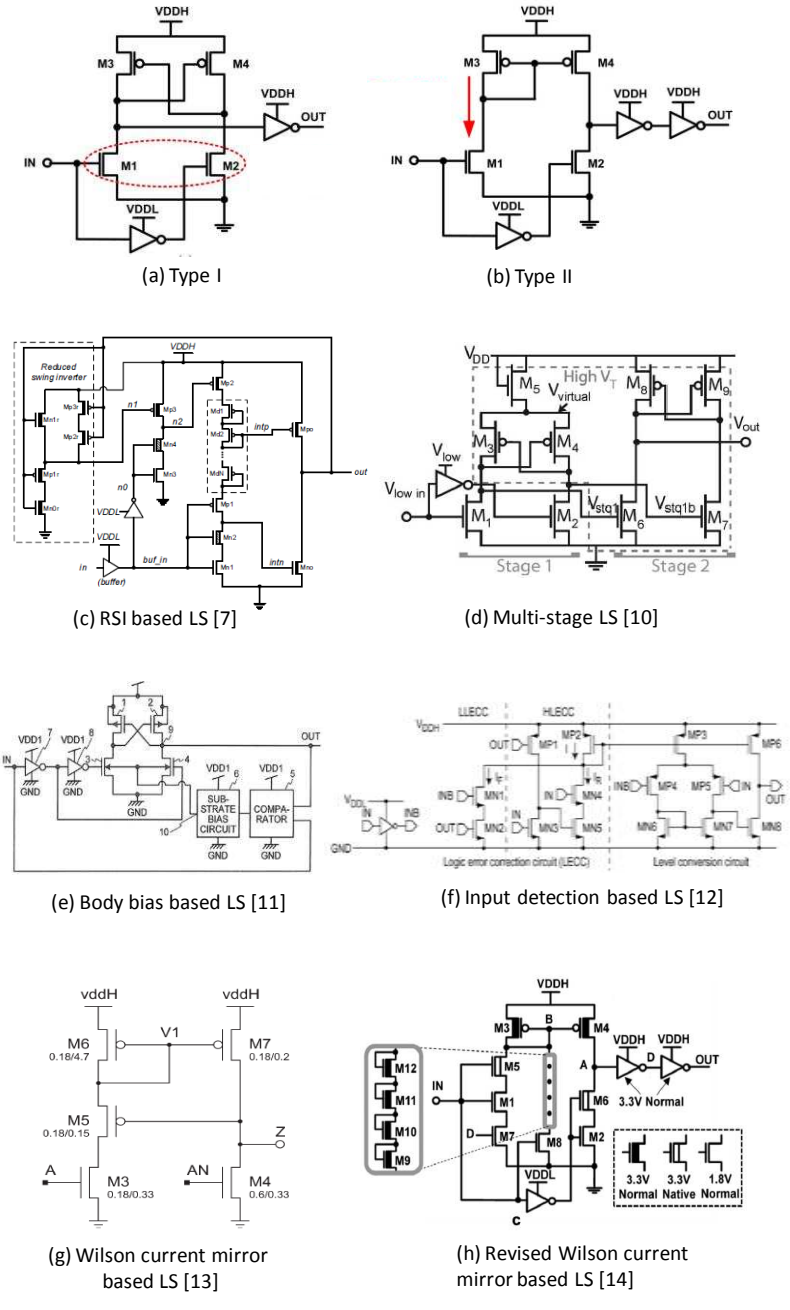


Fig. 1. Ultra-low voltage level shifters.

Level shifter is a crucial component in ultra-low voltage digital ICs for voltage conversion between different voltage domains including core-to-core and core-to-I/O. The conventional level shifter topologies for super-threshold level shifting have functionality or performance issues when operating at ultra-low voltage. For example, for the cross-coupled level shifting structure (i.e. Type I) as shown in Fig. 1(a), when the input voltage is extremely low, the pull-down NMOS cannot overcome the strength of the pull-up PMOS even after heavy upsizing, which will cause functional failures. For the current-mirror level shifting structure (i.e. Type II) as shown in Fig. 1(b), the static source current causes significant standby power, which will diminish the power saving from ultra-low voltage operation. To address these issues, some ultra-low voltage level shifters have been proposed in the past as shown in Fig. 1(c)-(h). In [7][8] the pull-up network is weakened by using reduced swing inverter (RSI) so as to prevent functional failure in Type I level shifters. However, in this topology the delay is not scalable with supply voltage as the pull-up network is constantly weakened. In[9][10], multi-stage level shifter is used to reduce the effort for wide-range level shifting. This effectively avoids the heavy upsizing of the pull-down NMOS while resulting in increased complexity and relatively long delay compared with single-stage level shifters. In [11], forward body bias is applied to help the level shifting at the price of increased area and power due to body bias control. For Type II level shifter, the major effort are spent on reducing the static current. In [12] the source current is enabled/disabled based on the detection of input transition. This significantly reduce the standby power while increasing the delay and dynamic energy due to the operation of the detection circuits. In [13], feedback control is used to cut off the source current after the output of the level shifter flips. However, the feedback structure causes output drop and charging sharing issues, resulting in non-optimal delay and energy consumption. The issues were addressed in [14] via a revised Wilson-current mirror based level shifter, which also uses mixed-V_T devices to achieve wide-range voltage conversion up to I/O voltage. While focusing on the performance optimization for ultra-low voltage operation, what is missing in the existing work is how to achieve optimal operation over a wide range of supply voltages across sub-threshold and super-threshold region. This need to be considered especially for wide-range dynamic voltage scaling (DVS)applications.

4 Ultra-Low Voltage SRAM

SRAM is heavily used in digital ICs. Conventional 6T SRAM cannot work at ultra-low voltage due to several issues such as read disturbance, degraded sensing margine and writability. To address these issues, the SRAM cell and write/read circuits need to be re-designed. In this section, several techniques are presented for energy efficient ultra-low voltage SRAMs.

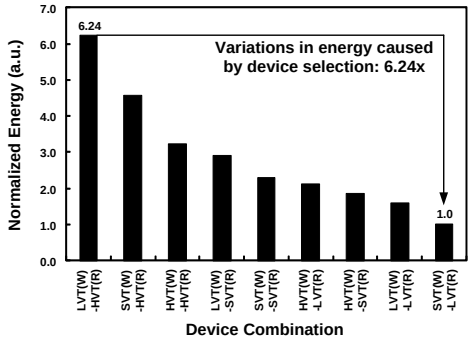


Fig. 2.Summary of normalized minimum energy consumption over various device combinations [15]

4.1 MTCMOS

Multi-threshold CMOS (MTCMOS) devices are commonly utilized in advanced CMOS technologies. Utilization of those devices properly can improve energy efficiency significantly. In general, devices with higher threshold voltage are used in non-critical paths while devices with lower threshold voltage are employed in critical paths. This improves the energy efficiency by reducing the leakage in the non-critical paths and maximizing the performance of the critical paths. In SRAMs, read delay is larger than write delay. Therefore, higher-V_{th} devices are preferred in the write paths while lower-V_{th} devices are better in the read paths. The variations in the energy of various device combinations are illustrated in Fig. 2. Note that the maximum energy occurs at the device combination of standard-V_{th} devices in the write paths and lower-V_{th} devices in the read paths (SVT(W)-LVT(R)), which is 6.24× better than that of LVT(W)-HVT(R). This indicates that proper device selection is not trivial in point of energy efficiency. The optimal device combination can be also affected by various circuit techniques for leakage reduction, dynamic power reduction, etc.

4.2 Read Assist Circuits

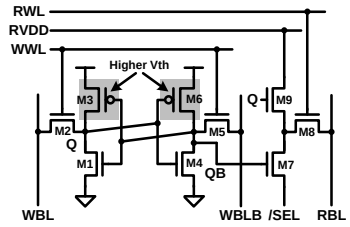


Fig. 3. Proposed SRAM cell with equalized bitline [15].

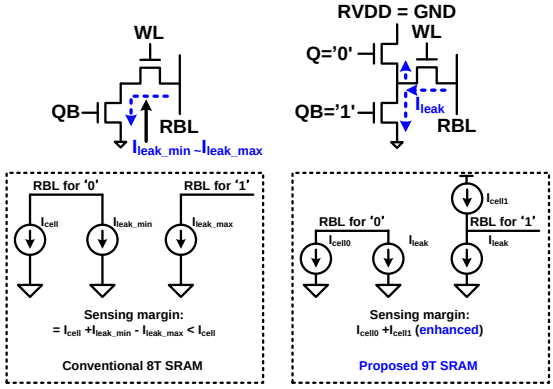


Fig. 4. Principle of the equalized bitline[15].

Scaling supply voltage degrades I_{on}-to-I_{off} ratio, which affects read bitline sensing margin. This limits the number of SRAM cells per bitline, maximum temperature, operating supply voltage, etc. One technique to reduce the impact of bitline leakage on sensing is to equalize the bitline leakage. Fig. 3 shows an 9T SRAM that can generate same leakage regardless of the stored data. In unselected SRAM cells, either M7 or M9 is turned on while RVDD and /SEL are grounded. Compared to the conventional bitline sensing (Fig. 4(left)) where sensing margin is affected by the amount of data-dependent bitline leakage, the equalized leakage always provides sensing margin (Fig. 4(right)) with the equalized bitline leakage. Another technique for improving sensing margin is to realize static bitline. Fig. 5 explains the principle of the static bitline. Unlike the conventional dynamic-operation-based read operation, the static bitline is implemented by turning on the pull-up PMOS devices with proper strength adjustment. This prevents read bitline from being fully discharged to GND. The final bitline voltage levels are determined by the strength of the pull-down paths and that of the pull-up pths, which achieves static bitlines. As shown in Fig. 6, the static bitline provides larger sensing margin and sensing timing window compared to the conventional bitline structure. However, this requires additional power during read operation. Therefore, it is neces-

sary to turn off the pull-up devices or read wordline (RWL) quickly after completing read operation.

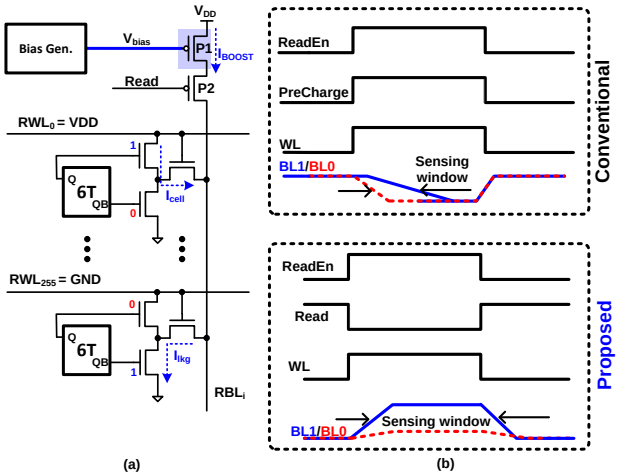


Fig. 5. (a) Schematic of the boosted bitline scheme [16] (b) timing diagram during a read operation of the conventional 8T and boosted bitline 8T.

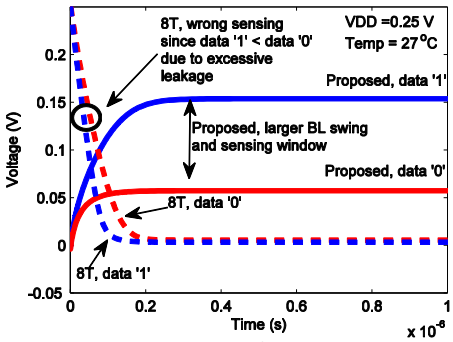


Fig. 6. Simulated proposed RBL waveforms and RBL swing of the conventional 8T at 27 ºC. RBL levels of data '1' is higher than data '0' in the proposed design. However it is reversed in the 8T design, indcating a wrong sensing[16].

4.3 Write Assist Circuits

Write operation is equally critical for reliable ultra-low voltage operation. Circuit techniques such as boosted wordline [17] and floating supply [17] improves write margin. However, they also exacerbate the half-selected cell stability. One technique for enhancing write margin without stability degradation is to use write-back techniques. The write-back operation is achieved by executing read operation before write operation. By writing read data into the write bitlines of unselected columns, the cell

stability disturbance caused by the conventional half-selection issue can be eliminated. However, this requires additional delay for executing the inserted read operation. However, by reducing the read delay through the hierarchical bitline structures, the overall performance of the write-back operation is comparable to the read performance. Therefore, the performance of the overall SRAM is not degraded. Fig. 7 explains the above write-back scheme, which is called 'fast local write-back'. A sample read/write timing diagram of fast local write-back scheme is presented in Fig. 8.

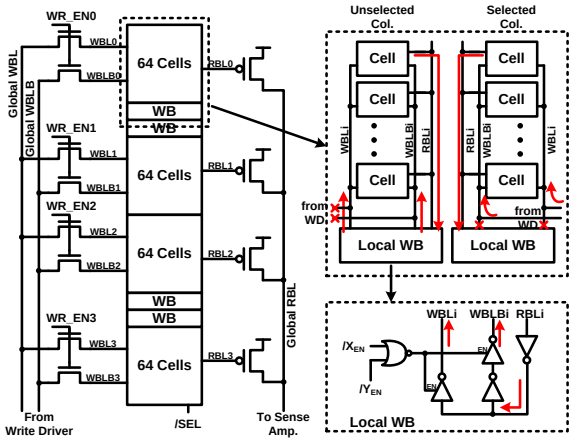


Fig. 7. Fast local write-back for improving cell stability.[15]

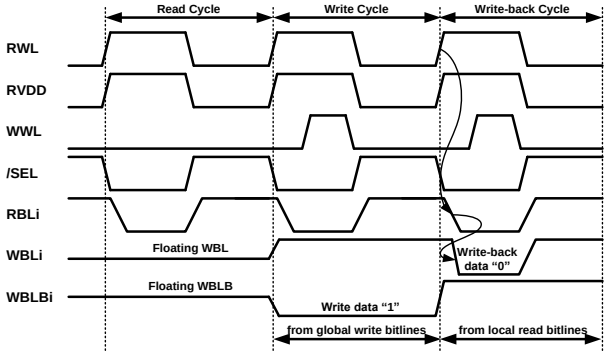


Fig.8. Timing diagram of the proposed fast local write-back scheme.[15]

5 Variation-Resilient Design

One of the largest challenges for ultra-low voltage digital IC design is the dramatically increased delay variations, which can be up to 100× compared to that for nominal voltage operation [1]. The conventional worse-case design method will result in significant design overhead in this case. Earlier the issue was addressed by on-chip timing error monitoring using the replica critical paths and adaptive clock/voltage tuning.

However, this cannot capture the local variations in the actual critical paths. In-situ timing monitoring techniques have been proposed to tackle this problem. In [19], razor technique is used to capture the late arrival signals (i.e. timing errors) by a shadow flipflop and correct them by architectural replay. However, this requires the minimal path delay to be increased to differentiate the late arrival and early arrival signals, leading to significant overhead due to buffer insertion. Also, its application is limited to high performance processors where architectural replay is available. In [20], canary flipflop technique is used to predict the error by monitoring artificially delayed signal. This eliminates the need for increasing minimum path delay. The drawback is there are some errors it cannot correct such as errors caused by fast variation or suddenly activated critical paths. In [21], half-path error monitoring is proposed to address the disadvantages of the razor and canary flipflop techniques. As the error is detected before the clock rising edge. It does not need to differentiate the late and early arrival signals, reducing the overhead of buffer insertion. Also, it is able to deal with errors caused by fast variations and suddenly activated critical path. Another advantage is that it is applicable to any digital designs as the error correction is done through general clock gating. For variation-resilient ultra-low voltage design techniques, the most important considerations include overhead, effectiveness and compatibility with standard design flow, which determine how the technique will be welcomed by major industry.

6 Conclusions

In this paper the major design techniques for ultra-low voltage digital IC design are reviewed. The device sizing methods considering current behavior and parasitic effects in near/sub-threshold region are reviewed and discussed. The ultra-low voltage level shifter design techniques employing revised cross-couple and current mirror structures are discussed. Various design techniques for ultra-low voltage SRAM are reviewed, including adoption of Multi-threshold CMOS device, read assist circuit and write assist circuits. The variation-resilient design techniques for ultra-low voltage operation are also reviewed and discussed.

References

[1] R. G. Dreslinski, et al., "Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits," Proceedings of the IEEE , vol.98, no.2, pp.253,266, Feb. 2010

[2] B.H. Calhoun, et al., "Modeling and sizing for minimum energy operation in subthreshold circuits," Solid-State Circuits, IEEE Journal of , vol.40, no.9, pp.1778,1786, Sept. 2005

[3] J. Kwong, et al., "Variation-Driven Device Sizing for Minimum Energy Sub-threshold Circuits," Low Power Electronics and Design, 2006. ISLPED'06. Proceedings of the 2006 International Symposium on , vol., no., pp.8,13, 4-6 Oct. 2006

[4] T. Kim, et al., "Utilizing Reverse Short Channel Effect for Optimal Subthreshold Circuit Design," Low Power Electronics and Design, 2006. ISLPED'06. Proceedings of the 2006 International Symposium on , vol., no., pp.127,130, 4-6 Oct. 2006

[5] J. Zhou, et al., "A 40 nm inverse-narrow-width-effect-aware sub-threshold standard cell library," Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE , vol., no., pp.441,446, 5-9 June 2011
[6] J. Zhou, et al., "A 40 nm Dual-Width Standard Cell Library for Near/Sub-Threshold Operation," Circuits and Systems I: Regular Papers, IEEE Transactions on , vol.59, no.11, pp.2569,2577, Nov. 2012
[7] Y. S. Lin, et al., "Single stage static level shifter designfor subthreshold to I/O voltage conversion," in Proc. ACM/IEEE Int.Symp. Low Power Electron. Design (ISLPED), Aug. 11–13, 2008, pp.197–200.
[8] I. J. Chang, et al., "Robust level converterfor subthreshold/super-threshold operation: 100 mV to 2.5 V," IEEETrans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 8, pp.1429–1437, Aug. 2011.
[9] B. Zhai, et al., "Energyefficientsubthreshold processor design," IEEE Trans. Very Large ScaleIntegr. (VLSI) Syst., vol. 17, no. 8, pp. 1127–1137, Aug. 2009.
[10] S. N. Wooters, et al., "An energy-efficientsubthreshold level converter in 130-nm CMOS," IEEE Trans. CircuitsSyst. II, Exp. Briefs, vol. 57, no. 4, pp. 290–294, Apr. 2010.
[11] "Level shifter circuit," U.S. Patent 7924080, 2011, Toshiba.
[12] Y. Osaki, et al., "A low-power level shifter with logic error correction for extremely low-voltage digital CMOS LSIs," IEEE J. Solid-State Circuits, vol. 47, no. 7, pp.1776–1783, Jul. 2012.
[13] S. Lutkemeier, et al., "A subthreshold to above-threshold level shifter comprising a wilson current mirror," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 57, no. 9, pp. 721–724, Sep. 2010.
[14] J. Zhou, et al., "An Ultra-Low Voltage Level Shifter Using Revised Wilson Current Mirror for Fast and Energy-Efficient Wide-Range Voltage Conversion from Sub-Threshold to I/O Voltage," Circuits and Systems I: Regular Papers, IEEE Transactions on , vol.62, no.3, pp.697,706, March 2015
[15] B. Wang, et al., "Maximization of SRAM Energy Efficiency Utilizing MTCMOS Technology," Asia Symposium on Quality Electronic Design (ASQED), pp. 35-40, July 2012
[16] Q. Li,et al., "A 5.61 pJ, 16 kb 9T SRAM with Single-ended Equalized Bitlines and Fast Local Write-back for Cell Stability Improvement," IEEE European Solid-State Device Research Conference (ESSDERC), pp. 201-204, Sept. 2012
[17] A. Do, et al., "A 32kb 9T SRAM with PVT-tracking Read Margin Enhancement for Ultra-low Voltage Operation," IEEE International Symposium on Circuits and Systems (ISCAS), May pp. 2553-2556, 2015
[18] B.H. Calhoun, et al., "A 256-kb 65-nm Sub-threshold SRAM Design for Ultra-Low-Voltage Operation," Solid-State Circuits, IEEE Journal of , vol.42, no.3, pp.680,688, March 2007
[19] S. Das, et al., "Razor II: In situ error detection and correction for PVT and SER tolerance," JSSC, vol. 44, no. 1, pp. 32–48, Jan. 2009.
[20] H. Fuketa, et al., "Adaptive performance compensation with in-situ timing error predictive sensors for subthreshold circuits," TVLSI, vol. 20, no. 2, pp. 333–343, Feb. 2012.
[21] J. Zhou, et al., "HEPP: A new in-situ timing-error prediction and prevention technique for variation-tolerant ultra-low-voltage designs," Solid-State Circuits Conference (A-SSCC), 2013 IEEE Asian , vol., no., pp.129,132, 11-13 Nov. 2013

Author Index