Phase-difference based logic: principle and applications

A.Yakovlev, A. Bystrov, D. Sokolov, J. Murphy
(University of Newcastle upon Tyne, UK)
V. Varshavsky, V. Marakhovsky, Advanced Logic Projects and The University of Aizu, Inc. Japan

Talk outline

- Motivation
- PDBL: principles and examples
- Applications: security, testing, decoherence
- Design flow and tool
- Cryptography hardware case study

Motivation

- Traditional classification based on timing is along the line: clocked vs self-timed. It caters for the way how computations are controlled, either on GLOBAL VALID (clocked) or DISTRIBUTED VALID (self-timed) signal.
- This approach helps in finding ways of battling clock-distribution problems, modularity, robustness, power-saving …
- However, this approach *does not reflect properties of the switching activity of individual nodes* in circuits, which may be important in a range of application domains, e.g. security, testability, decoherence.

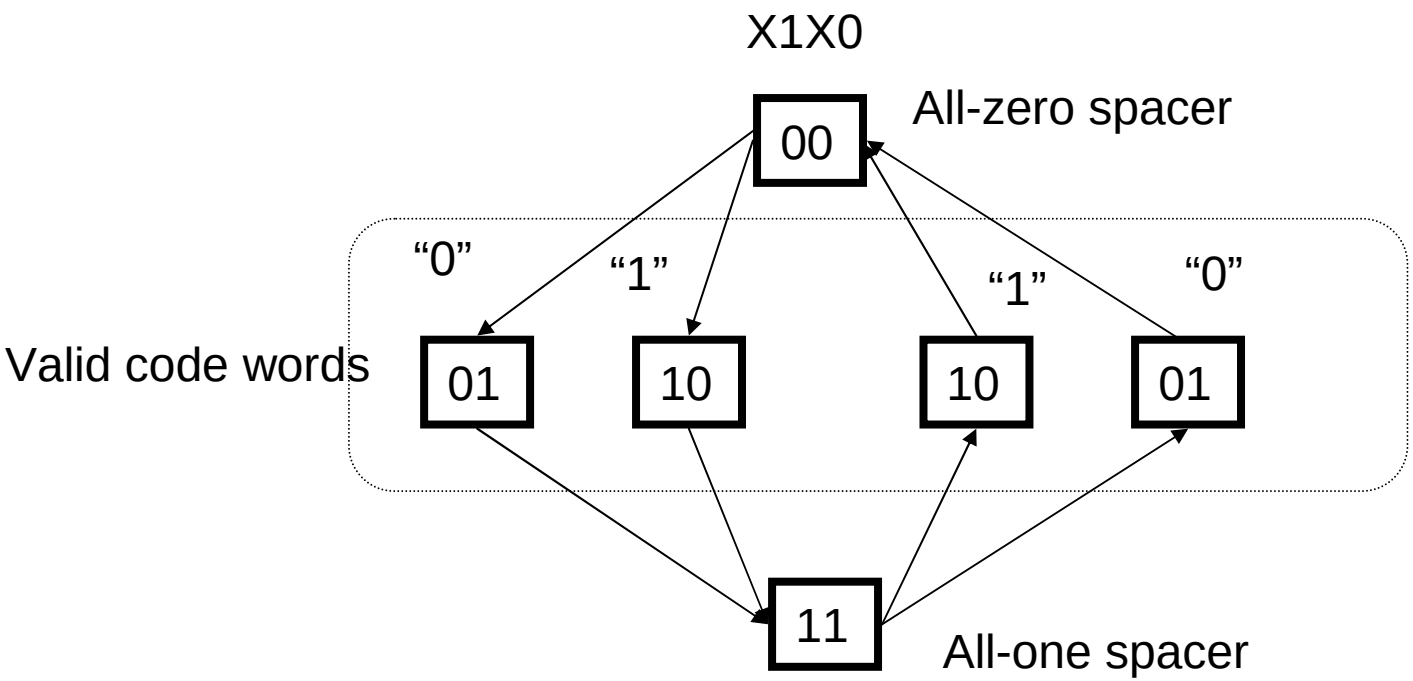
Motivation

- Another classification is w.r.t. to switching activity of the nodes in a logic circuit: arbitrary vs predictable in some sense. Eg:
- Can we build circuits whose switching activity is invariant to processed data? (good for security)
- Can we build circuits whose Idd measured on a short time interval can fully characterise the absence or presence of faults (good for testing)
- Can we build circuits whose nodes have stable predictable periodicity in switching? (good for decoherence/refreshing)

Phase Difference Based Logic

- Uses dual-rail representation (other also possible but probably not so efficient)
- Both rails must switch in every operational cycle, regardless of the data value. How?
- Bit X: (X1,X0):
  - Valid states (0,1) for “0” and (1,0) for “1”
  - Spacer states (0,0) and (1,1)

PDBL protocol



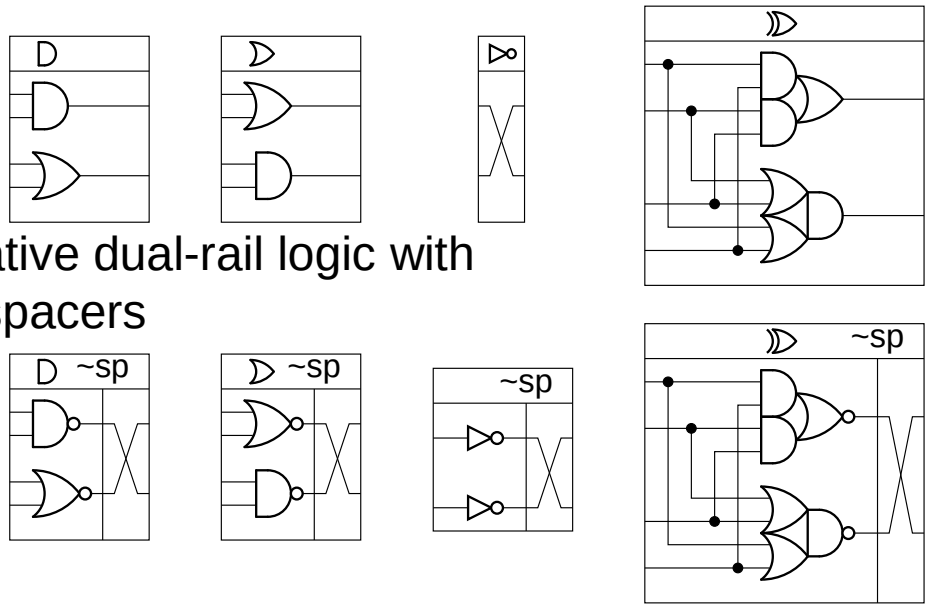
PDBL vs conventional dual-rail

- Conventional dual-rail uses ONE spacer only, typically the ‘all-zero’ (e.g. NCL)
- Compare switching activity between PDBL and single-spacer logic. E.g. for a sequence 0001 on X
  - Single-spacer logic: (x0+,x0-,x0+,x0-,x0+,x0-,x1+,x1-), the total 3*X0+,3*X0-,1*X1-,1*X1- (depends on the sequence)
  - PDBL: (x0+,x1+,x1-,x0-,x0+,x1+,x0-,x1-), the total 2*X0+,2*X0-,2*X1+,2*X1- (does not depend on the sequence)
- In PDBL switching activity of the bit X taken on the sequence duration is independent on the values. The *data is encoded in phase differences* between X0 and X1. However, the difference of course remains within the single cycle of operation.

How to implement PDBL?

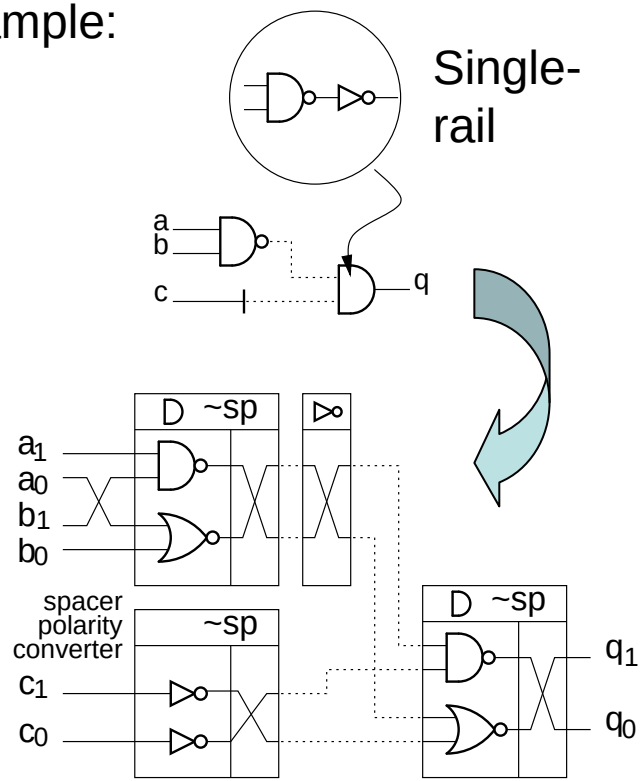
- Combinational logic is built in the same way as conventional hazard-free dual-rail

Positive dual-rail logic with (0,0) spacer



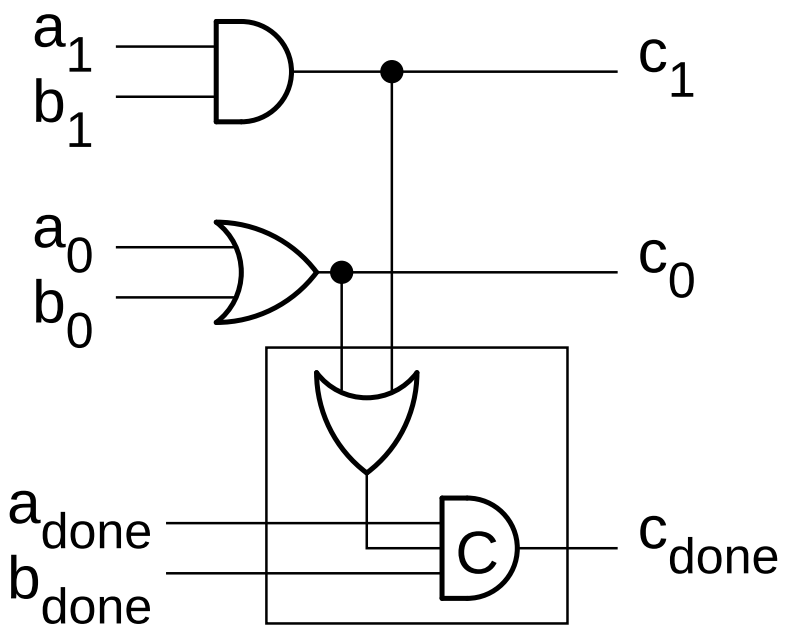
Negative dual-rail logic with two spacers

Example:



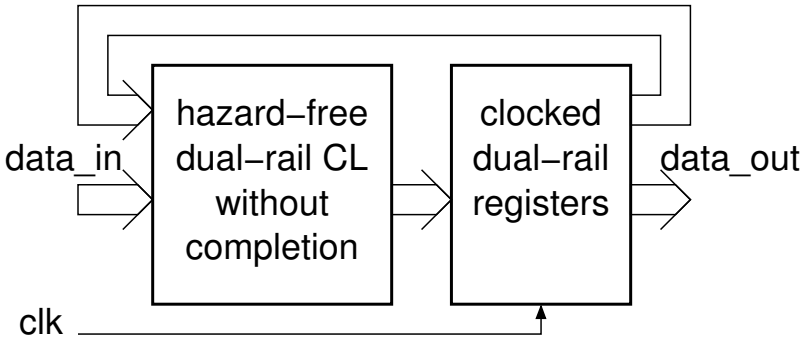
PDBL Combinational Logic

- If necessary completion detection can be added on a per gate basis (cf. Kondratyev & Lwin, IEEE D&T, Jul/Aug 2002):

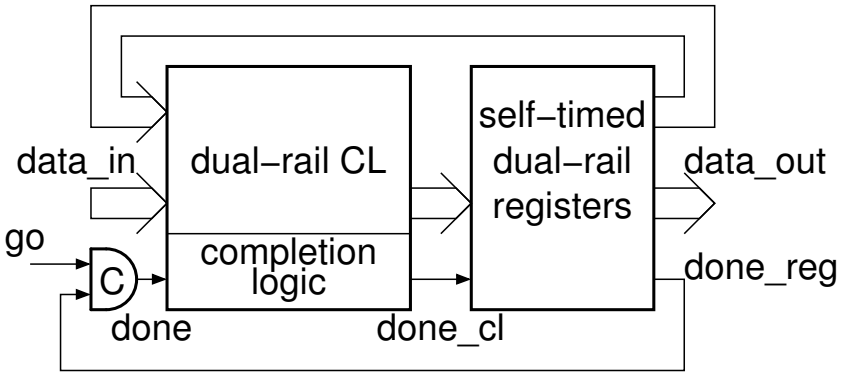


Possible architectures

Clocked dual-rail

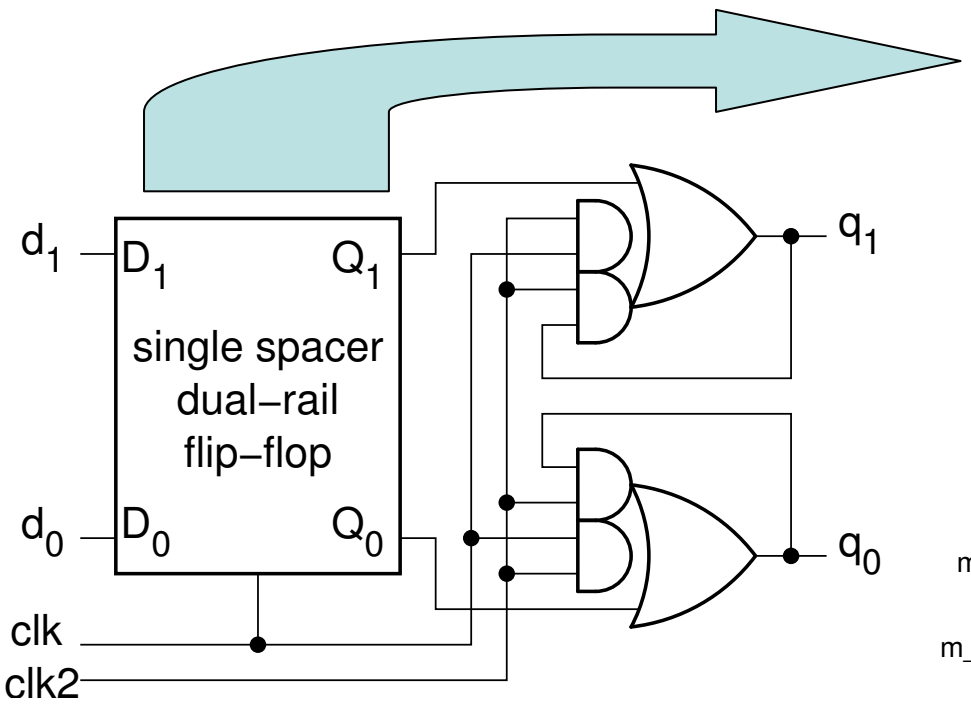


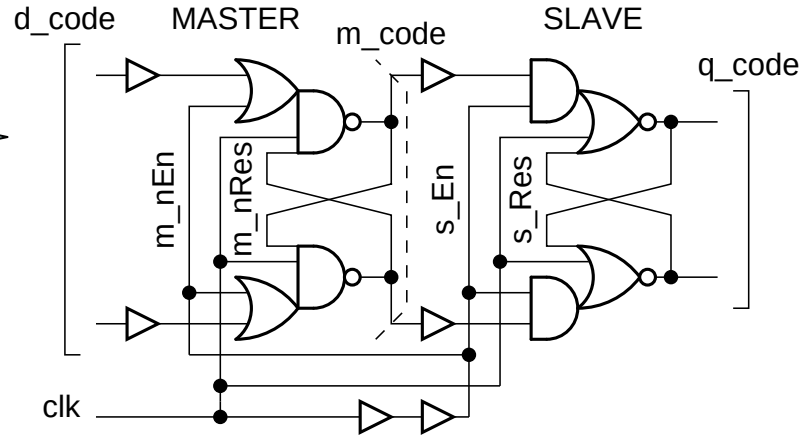
Self-timed dual-rail

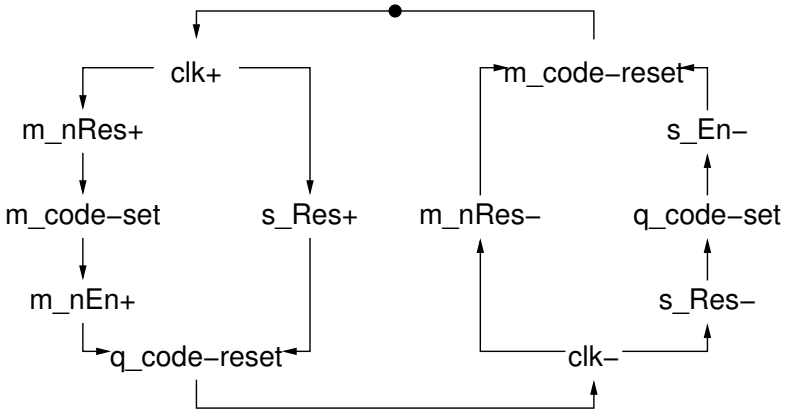


Registers for PDBL

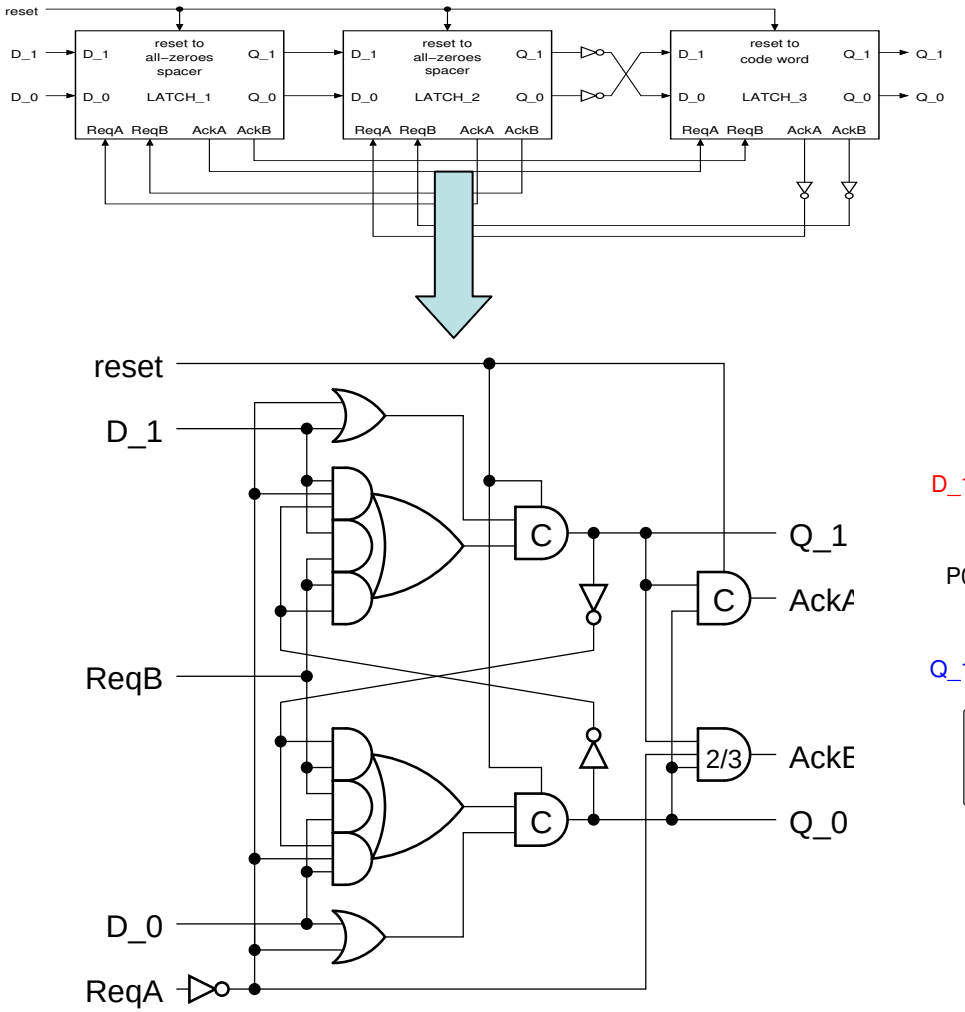
For clocked architecture:



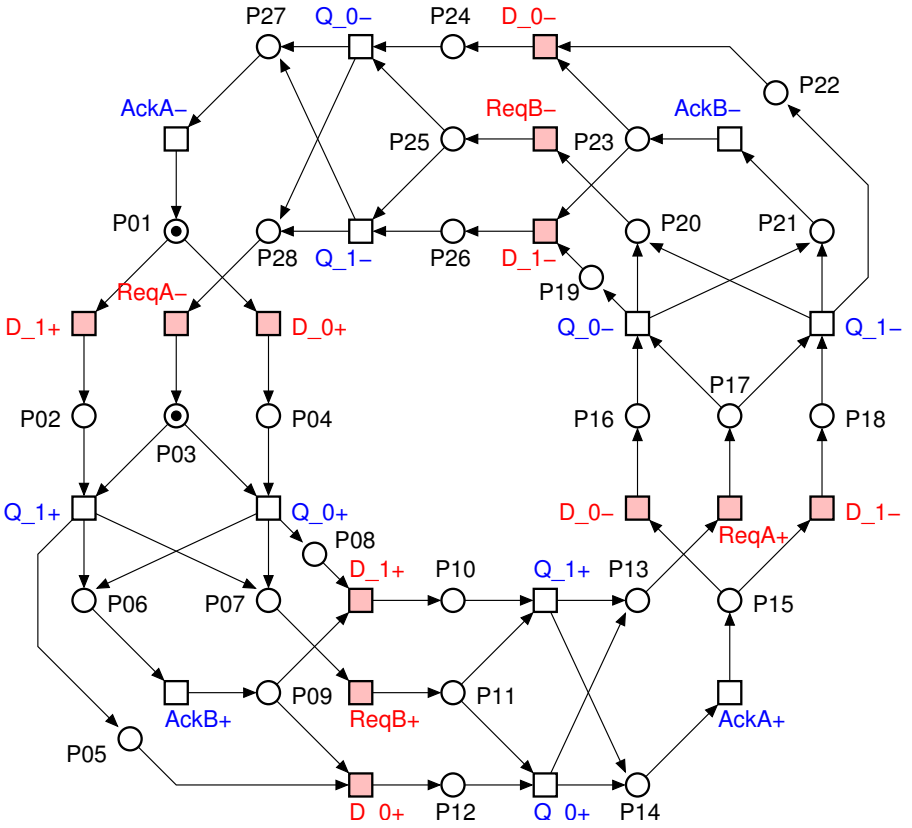




Registers for PDBL

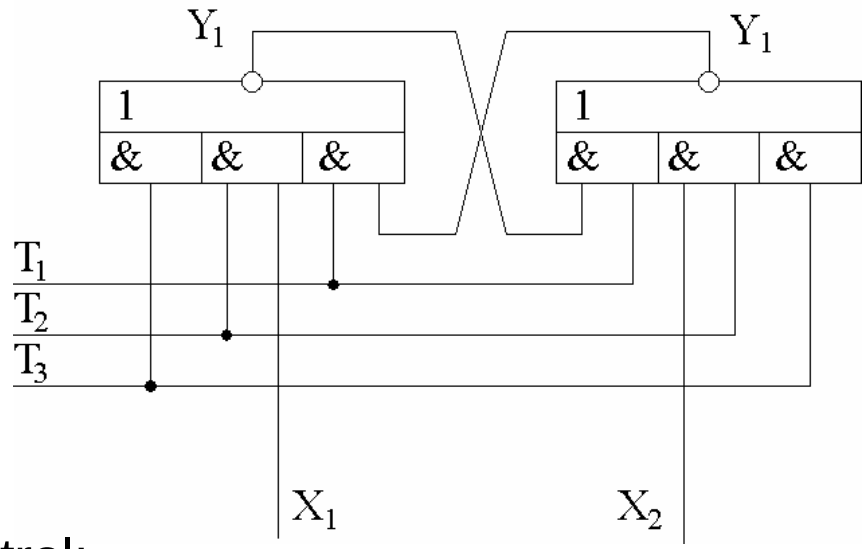


For self-timed architecture:



PDBL architectures

- Another possible architecture is based on Synchro-strata following V. Varshavsky & V. Marakhovsky, GALA (Globally Asynchronous - Locally Arbitrary) Design, Concurrency and Hardware Design Advances in Petri Nets, LNCS 2549, pp. 61-107.



Flip-Flop timing control:

T_3	T_2	T_1	Action
0	0	0	Spacer [1,1]
0	0	1	Stable memory
0	1	0	Enable
0	1	1	Transition from Enable to Memory
1	*	*	Spacer [0,0]

Security Issues: imbalance

- Security of logic gates. Imbalance is measured as variation in energy consumed by a circuit when processing different data:

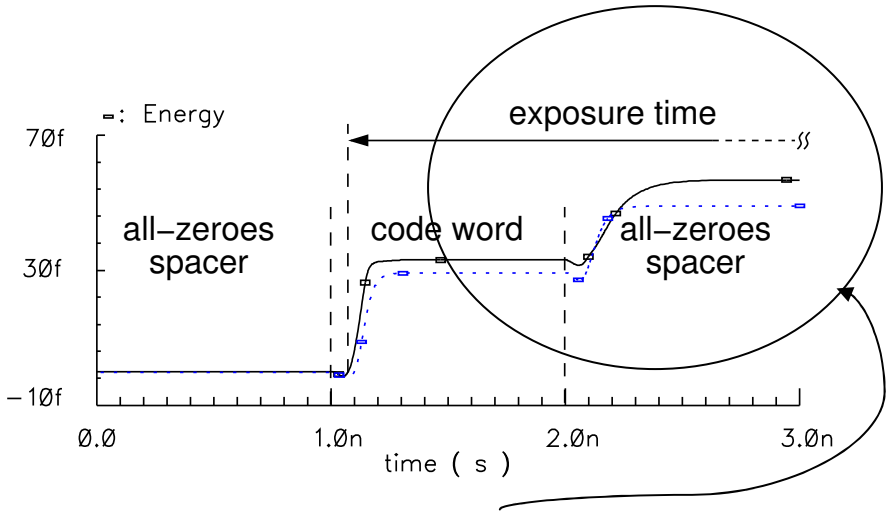
$$d = \frac{|e_1 - e_2|}{e_1 + e_2} * 100\%$$

where e_1 and e_2 are the energy consumptions of two input patters

For 3-input NAND and NOR gates implementing the two rails of a dual rail NAND/NOR, the imbalance (for Vcc=3.3V,1ns pulse and 150ps rise and fall times switching) was 10.7% for unloaded gates and 2.1% for realistic loads

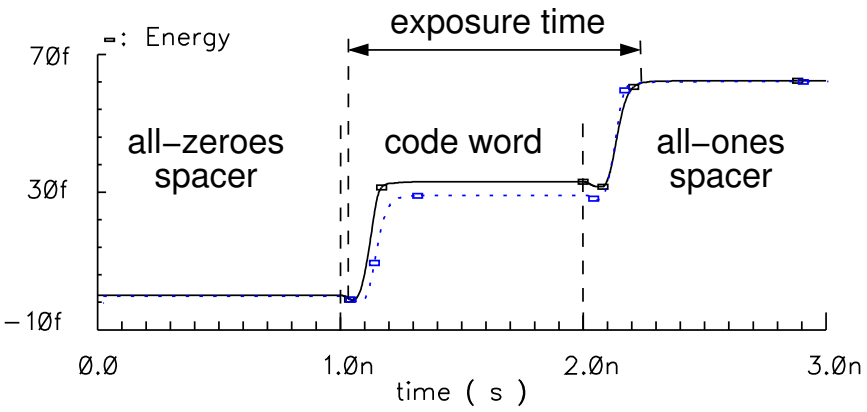
Security issues: exposure time

Single spacer protocol



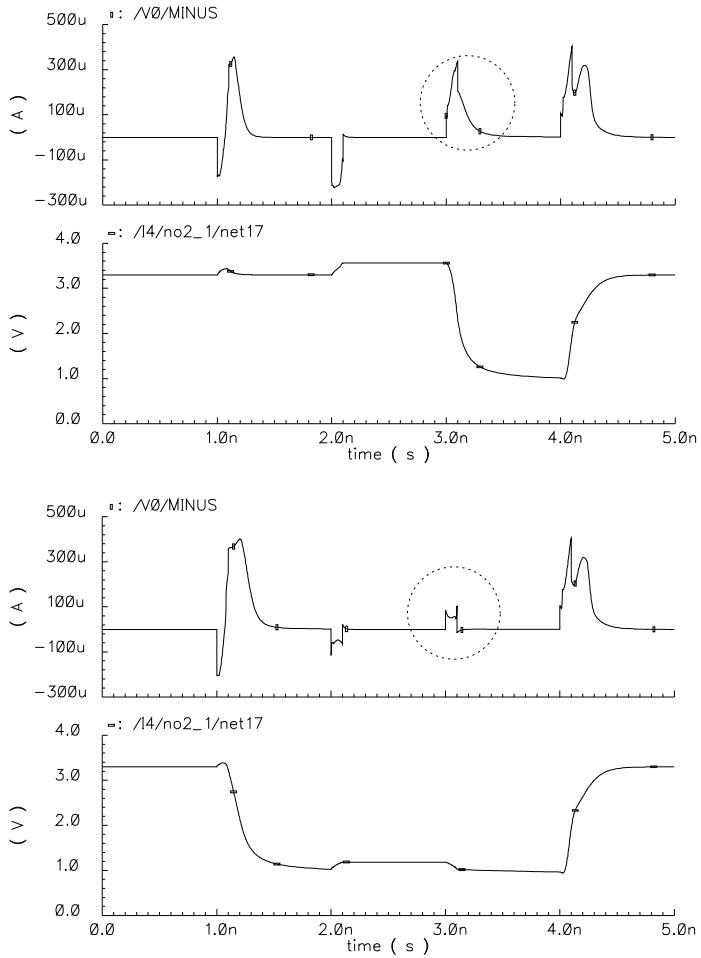
The imbalance accumulates

Dual spacer protocol (PDBL)



Security: early propagation and memory effect

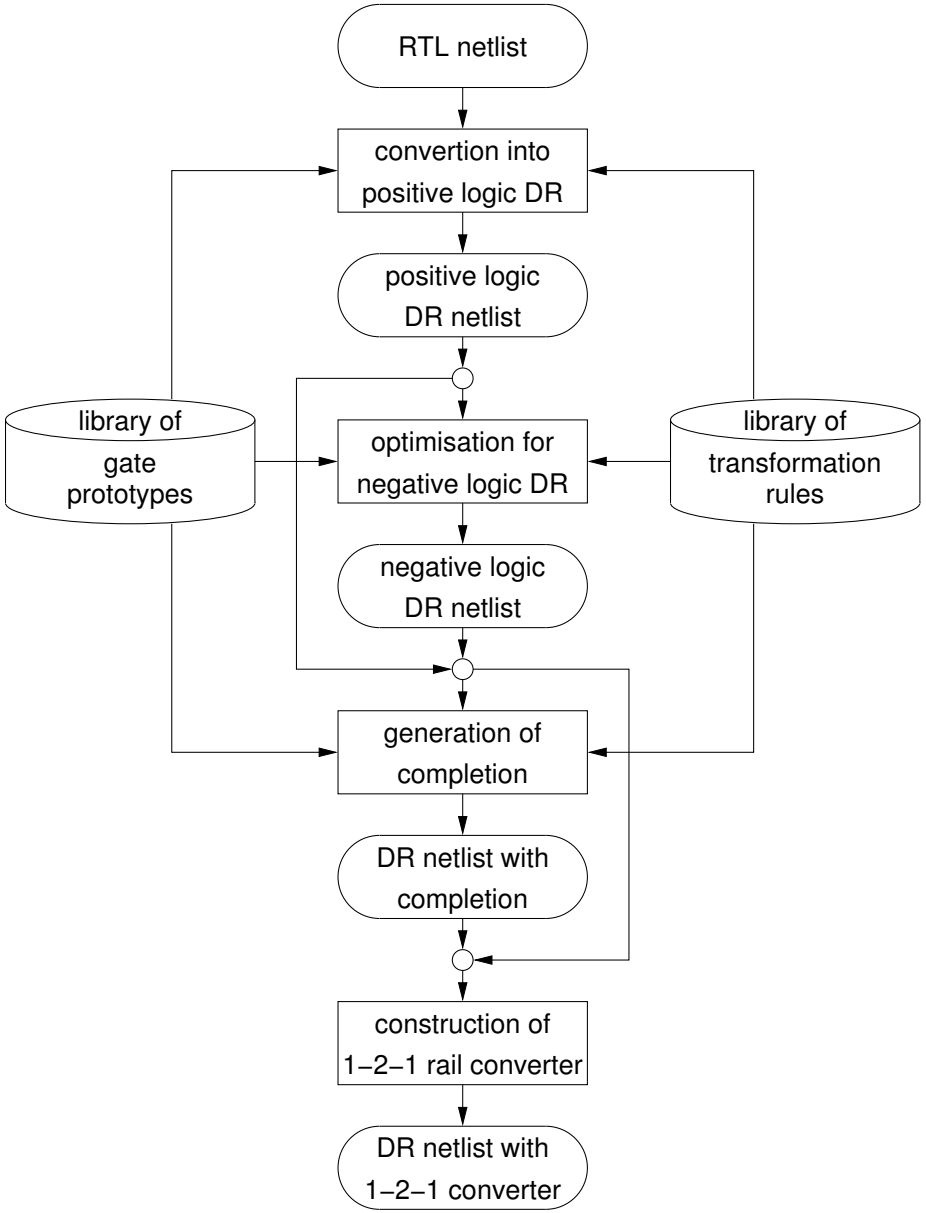
- Due to the inherent presence of OR-causality in level-based logic gates (eg. rising edge on NOR and falling edes on NAND), early propagation can cause data dependency imbalance. The exposure time can be reduced to within one dual-rail gate if we use completion detection at the gate level (cf. NCL-D)
- Gates keep charges on parasitic capacitances within transistor stacks (eg., p-stack for NOR). This can be partly battled by making two parallel stacks, <ab> and <ba> instead of one (cost: interfere with gate library!)



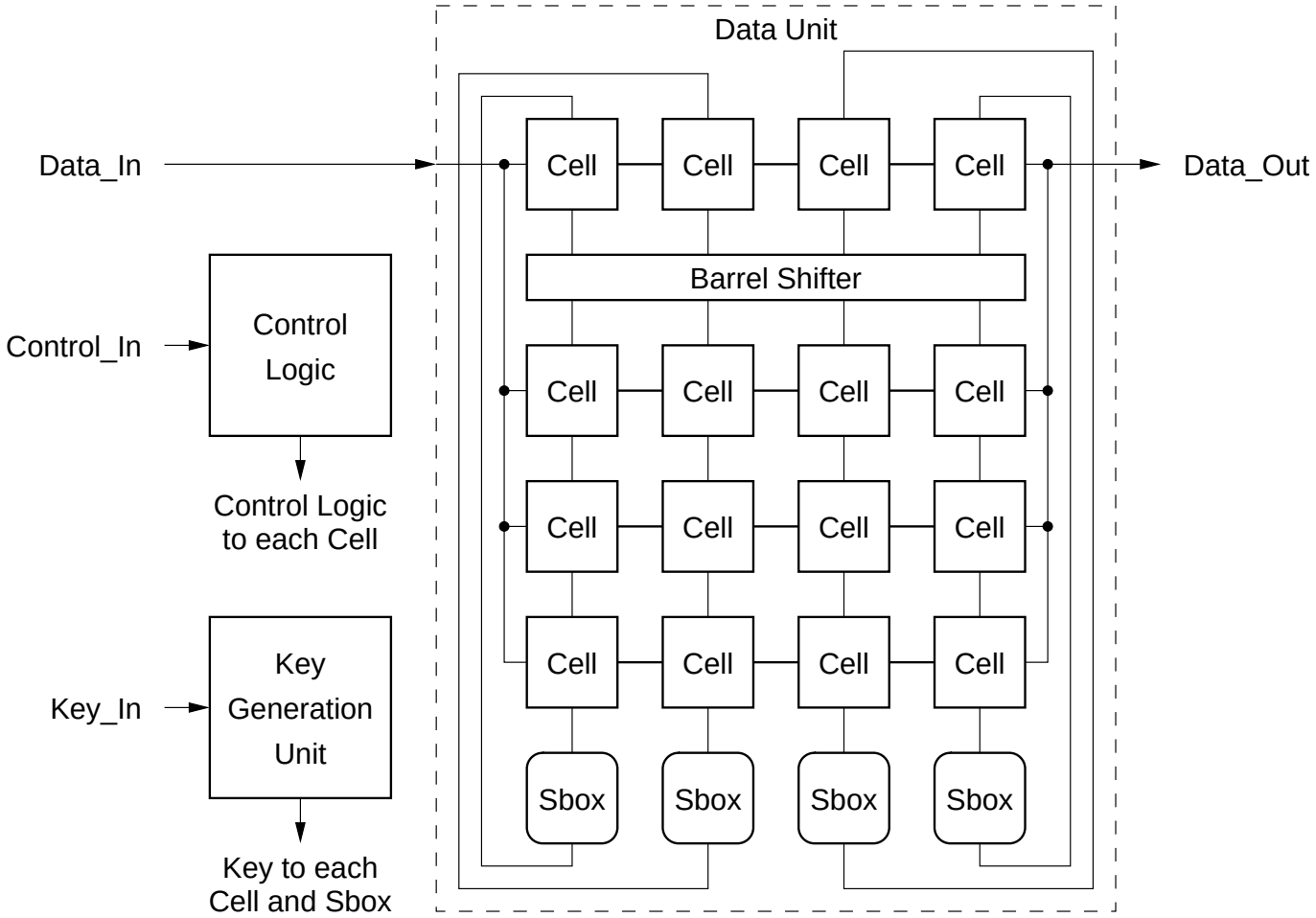
Design tool

One of the main goals is to carry out all conversions within the standard industrial (Verilog) RTL design flow

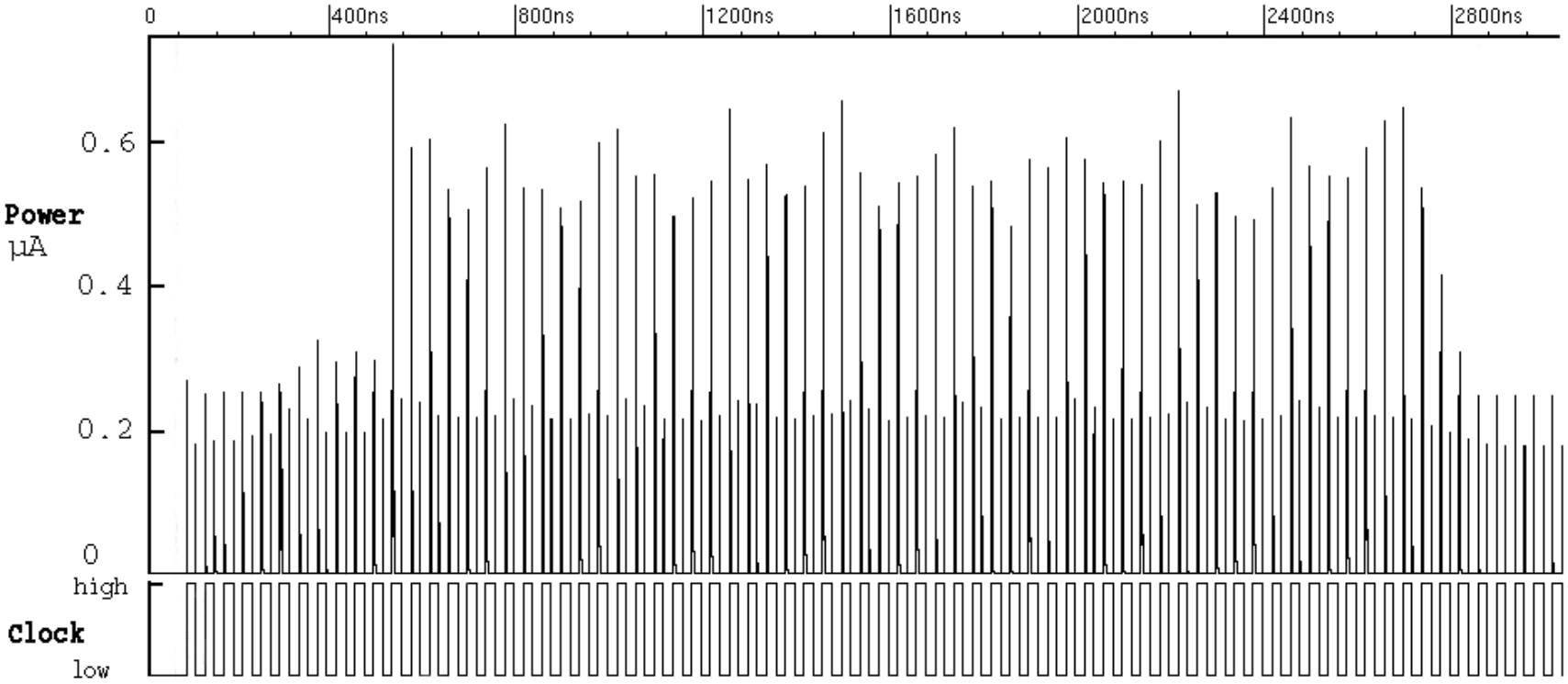
Additionally, target an easy inclusion into async design flow



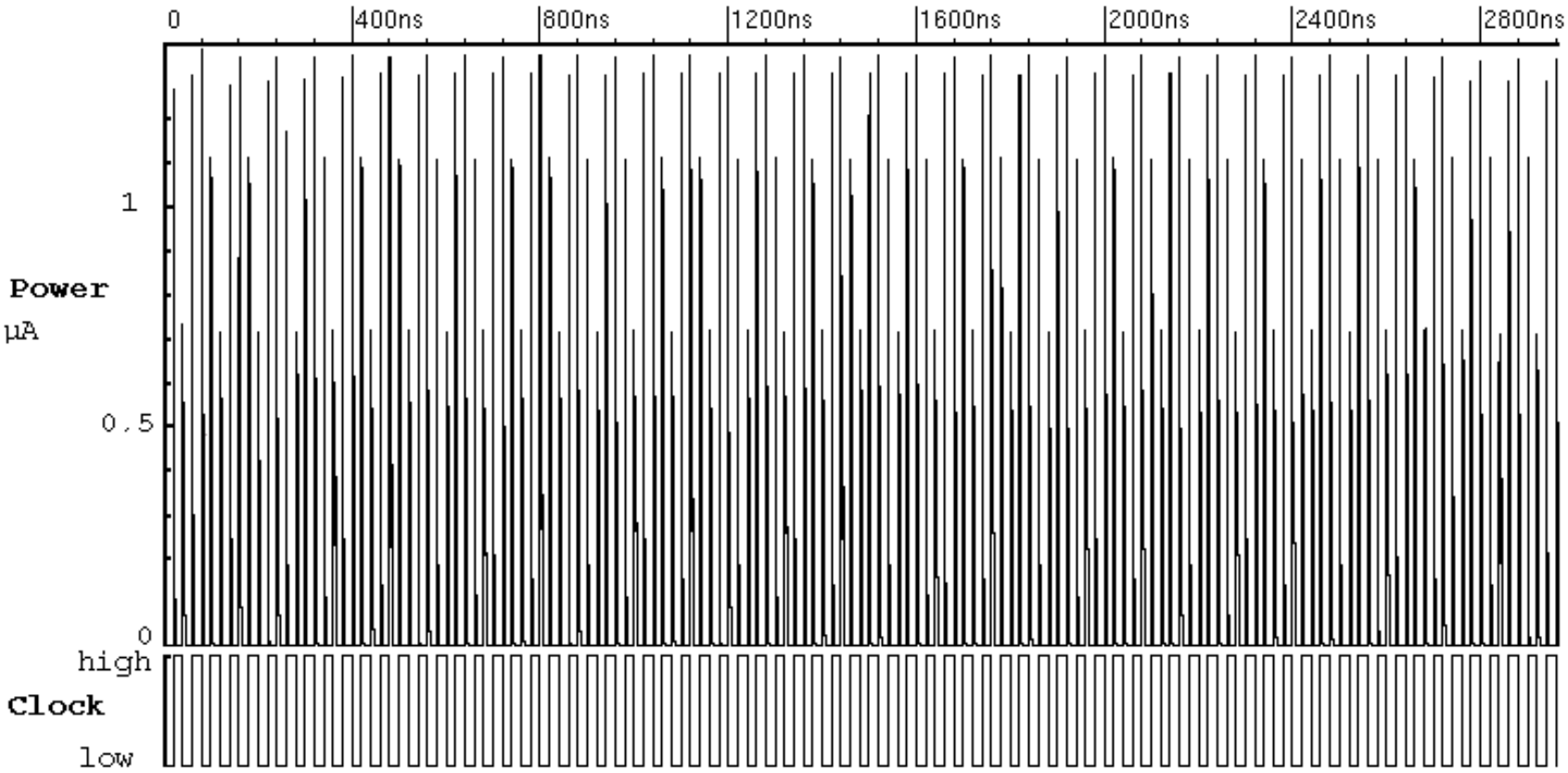
AES design case study



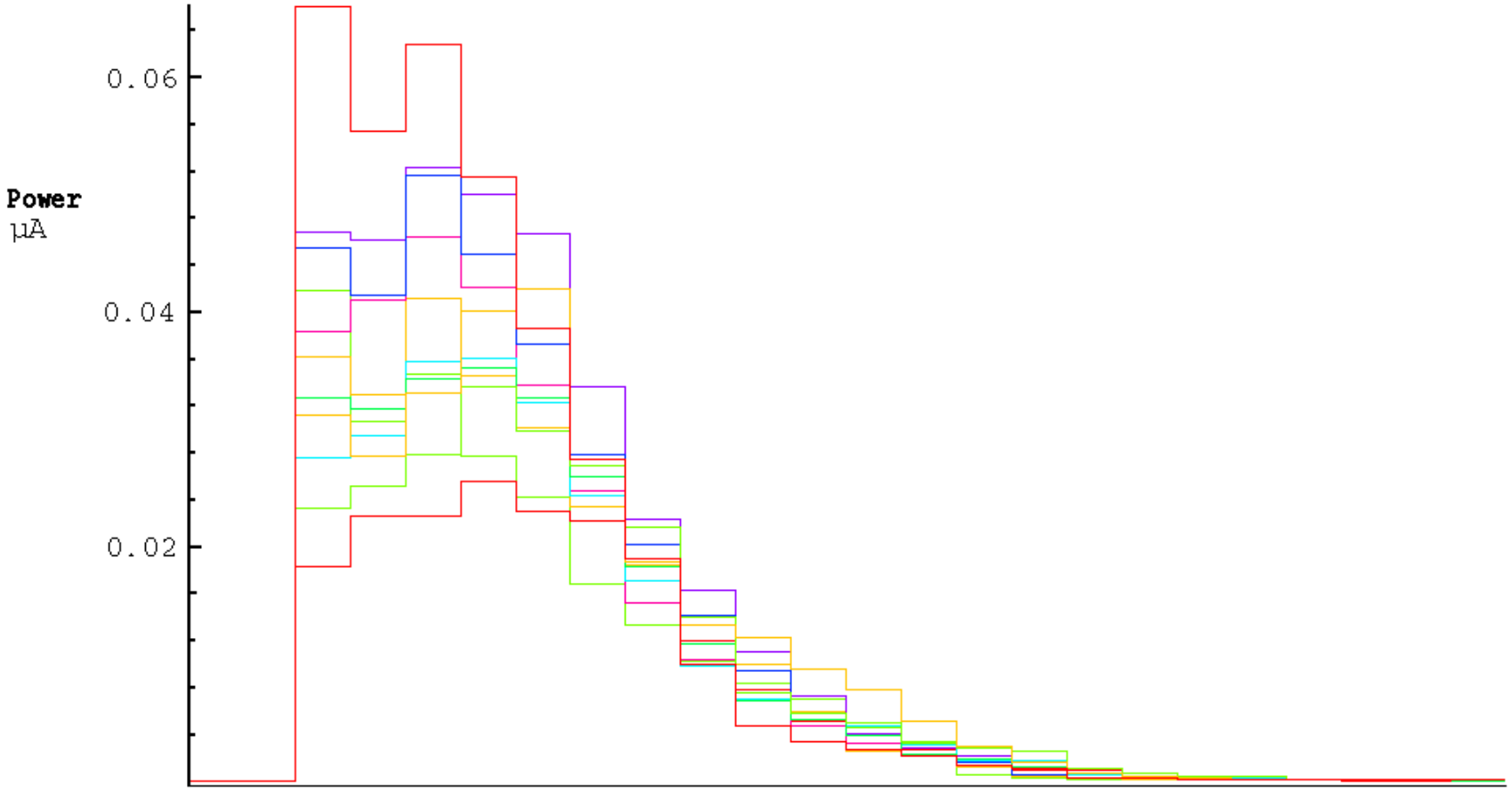
Power signature simulations for AES (single rail)



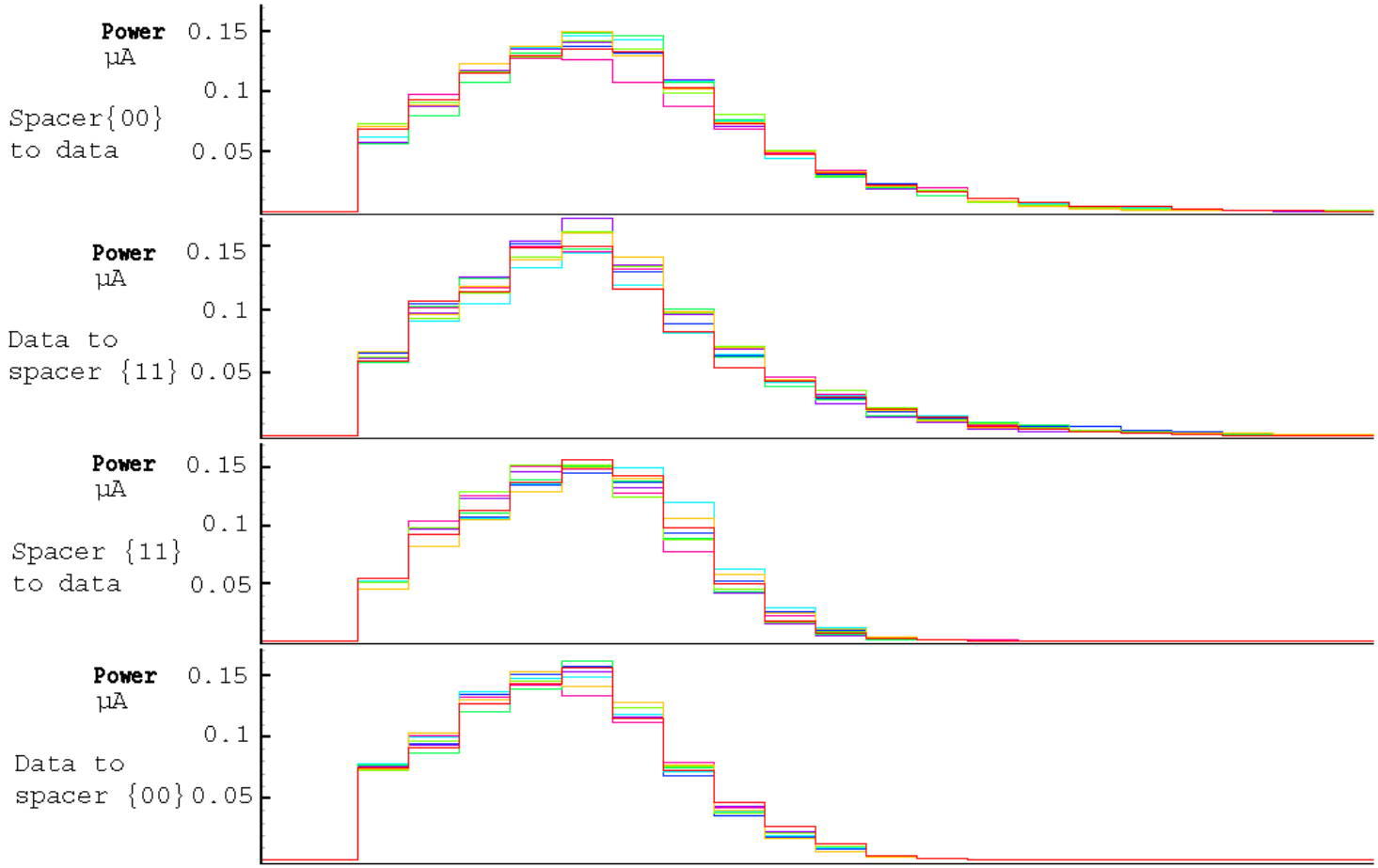
Power signature simulations for AES (dual rail PDBL)



Power signature simulations for AES (sbox for different data, single rail)



Power signature simulations for AES (sbox for different data, PDBL)



AES: design area

benchmark name		negative gate count	transistor count	wire count	estimated area	
					CL	FF
sbox (open core)	single-rail	655	3,180	482	44,593	0
	dual-rail	1,523	6,672	1,180	101,364	0
	overhead	133%	110%	145%	127%	0
sbox (computable)	single-rail	634	2,362	400	32,975	0
	dual-rail	1,164	4,628	868	68,603	0
	overhead	84%	96%	117%	108%	0
ciphet (open core)	single-rail	12,752	68,184	9,980	873,175	142,370
	dual-rail	26,396	139,828	24,367	1,925,190	466,870
	overhead	107%	105%	144%	120%	228%
ciphet (computable)	single-rail	10,372	50,344	5,936	580,046	118,678
	dual-rail	19,510	95,066	13,055	1,237,260	462,021
	overhead	88%	89%	120%	113%	289%

AES: switching activity

benchmark name		switching activity	
		single spacet	alternating spacet
ciphet (enctyption)	tail_1	8,388	6,505
	tail_0	4,622	6,505
	imbalance	29%	0%
ciphet (dectyption)	tail_1	8,572	6,505
	tail_0	4,438	6,505
	imbalance	32%	0%

benchmark name		switching activity (hazards)		
		min	avg	max
sbox (open core)	single-tail	0 (0)	162 (33)	277 (124)
	dual-tail	1,180	1,180	1,180
	overhead	∞	628%	326%
sbox (computable)	single-tail	0 (0)	525 (345)	936 (746)
	dual-tail	868	868	868
	overhead	∞	65%	-17%
ciphet (open core)	single-tail	0	9,147	13,236
	dual-tail	41,285	41,285	41,285
	overhead	∞	351%	211%
ciphet (computable)	single-tail	0 (0)	3,810 (2,013)	6,140 (3,682)
	dual-tail	13,055	13,055	13,055
	overhead	∞	242%	112%

Conclusions

- PDBL is a way to build circuits with data-independent (at logic level) switching activity – a circuit's structure determines its behaviour
- Can be naturally combined with self-timing (desyncronization is an orthogonal aspect)
- Application to secure circuit design (industrial exploitation is under way)
- Other applications are in testability and periodic refreshing (cf. decoherence) are to be studied next
- Automatic design flow exists
- Main problem is with high power consumption on its own. To battle this use PDBL selectively or within a clever clock gating context