2014 47th Annual IEEE/ACM International Symposium on Microarchitecture

Voltage Noise in Multi-core Processors: Empirical Characterization and Optimization Opportunities

Ramon Bertran*, Alper Buyuktosunoglu*, Pradip Bose*, Timothy J. Slegel†, Gerard Salem‡, Sean Carey†, Richard F. Rizzolo†, Thomas Strach§
*IBM Research, Yorktown Heights, NY, USA
†IBM Systems & Technology Group, Poughkeepsie, NY, USA
‡IBM Systems & Technology Group, Burlington, VT, USA
§IBM Systems & Technology Group, Boeblingen, Germany
Email: {rbertra,alperb,pbose,slegel,gsalem,smcarey,rizzolo}@us.ibm.com, strach@de.ibm.com

Abstract—**Voltage noise characterization is an essential aspect of optimizing the shipped voltage of high-end processor based systems. Voltage noise, i.e. variations in the supply voltage due to transient fluctuations on current, can negatively affect the robustness of the design if it is not properly characterized. Modeling and estimation of voltage noise in a pre-silicon setting is typically inadequate because it is difficult to model the chip/system packaging and power distribution network (PDN) parameters very precisely. Therefore, a systematic, direct measurement-based characterization of voltage noise in a post-silicon setting is mandatory in validating the robustness of the design.**

In this paper, we present a direct measurement-based voltage noise characterization of a state-of-the-art mainframe class multi-core processor. We develop a systematic methodology to generate noise stressmarks. We study the sensitivity of noise in relation to the different parameters involved in noise generation: (a) stimulus sequence frequency, (b) supply current delta, (c) number of noise events and, (d) degree of alignment or synchronization of events in a multi-core context. By sensing per-core noise in a multi-core chip, we characterize the noise propagation across the cores. This insight opens up new opportunities for noise mitigation via workload mappings and dynamic voltage guard-banding.

Keywords-**dI/dt; inductive noise; voltage droop; stressmark generation; noise-aware workload mapping; dynamic guard-banding; multi-core hardware measurements**

I. Introduction

For mainframe-class systems like the IBM System z series, zero-fault reliability is an essential part of the design specification. Continued technology scaling and the well known 'power-wall' has added pressure to the historically conservative design margins used for reliable operation. Specifically, in the face of aggressive supply voltage scaling and power management solutions to reduce the power burden, the need to characterize the voltage noise in precise detail has become imperative. Thus, a systematic approach is needed to define and validate an operating voltage point that ensures robust performance and zero-fault reliability, without being overly conservative.

Modeling and characterization of voltage noise in a pre-silicon setting is typically inadequate. This is because it is difficult to model the chip/system packaging and power distribution network (PDN) parameters very precisely. Therefore, a systematic, direct measurement-based characterization of voltage noise in a post-silicon setting (during the processor testing process) is required to define the adequate voltage levels and package characteristics that ensure robust functionality.

A key part of such post-silicon processor testing process is the use of specially crafted benchmarks that maximize the voltage noise, known as dI/dt stressmarks[1]. Manual, expert-driven generation (and fine-tuning) of such dI/dt stressmarks can be a tedious and error-prone process. This is specially true in a multi-core setting, in which inter-core synchronization of noise events and experimental discovery of the system resonance frequencies can require hundreds (or even thousands) of test runs with hand-crafted programs.

In this paper, we use a latest generation IBM System z multi-core processor (IBM zEnterprise™ EC12 processor chip) to illustrate a systematic, automation-driven voltage noise characterization methodology. This methodology is deemed to be a key post-silicon aid in determining the optimal voltage levels and package characteristics to ensure the robust performance and ultra-reliable functionality that is targeted for IBM mainframe systems. In particular, we present a tight bounds analysis to quantify maximum voltage noise on the targeted system. This includes a characterization of *noise propagation* across the cores in the multi-core chip. The main contributions of the paper are:

- A systematic methodology to generate all types of dI/dt stressmarks. The methodology, in contrast to previous works (e.g. [25], [26], [28]), permits the control of different parameters affecting the voltage noise generated: amount of ΔI, synchronization level of ΔI events, number of consecutive ΔI events and frequency of ΔI events. This 'white-box' approach provides a rich toolbox to perform detailed noise characterizations.
- A complete noise sensitivity analysis of the different parameters affecting voltage noise: the amount of ΔI, the synchronization level of ΔI events, the number of consecutive ΔI events and the frequency of ΔI events. This detailed characterization allows us to: (a) validate the robustness of the zEC12 power delivery system, (b) confirm the shift of the '1st droop' resonance frequency to lower bands (around the 2MHz band) due to the usage of deep trench technology and large on-chip eDRAMs, (c) empirically quantify the relative importance of each parameter involved in the noise generation. We show that the amount of ΔI and the synchronization of ΔI events are the main contributors of noise and the number of consecutive ΔI events and their frequency are secondary factors. Overall, we present a significantly more comprehensive noise characterization of multi-core systems compared to prior art (e.g. [2], [27], [31], [40], [41], [55]), which do not study all the parameters related to noise.

[1]In this work we use the terms dI/dt and ΔI interchangeably and we refer to a dI/dt event or ΔI event as a fast transition in power consumption.

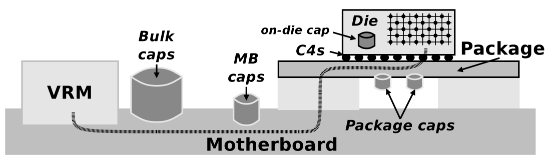


Figure 1: Schema of a power distribution network (PDN).

- An analysis of the noise propagation on multi-core processors based on per-core noise hardware measurements. We describe a method to empirically evaluate the effects of physical layout, process variation and other power delivery system design parameters on the noise perceived by each core. For instance, we show that the L3 cache —with a relatively large capacitance— isolates the noise coming from different cores. Thus, it acts as a damping element.
- A discussion of the optimization opportunities that arise from noise propagation behavior. We provide a conceptual overview of the potential benefits of a noise-aware workload mapping policy and a utilization-based dynamic voltage guard-banding.

To the best of our knowledge, there is no previous work on complete noise characterization of multi-core processors in which both intra- and inter-core noise behavior is captured and interpreted in the context of a real, high-end mainframe system. Accurate noise characterization and robustness testing is a pre-requisite to shipping systems to customers in this high-end market segment. Hence, this work is of significant and tangible value to a commercial systems vendor.

The rest of the paper is organized as follows: In Section II we review the fundamentals of voltage noise. Section III presents the stressmark generation methodology. Section IV details the experimental set up. Section V and VI present the analysis of noise behavior and inter-core noise characteristics respectively. Section VII discusses the optimization opportunities. Section VIII presents the related work and we conclude in Section IX.

II. Background

This section provides the basic background to understand the reasons for the existence of voltage noise and its negative effects on energy efficiency and reliability. For further details, we refer the reader to the several previous works on the topic [16], [20], [22], [24], [30], [34], [38], [43], [46].

A. Problem Overview

In a computer system, the electrical power has to be distributed from the main power source to all the devices present in the micro-processor. This is done through the power distribution network (PDN). Figure 1 shows a schematic diagram of a PDN. First, a voltage regulator module (VRM) attached in the motherboard reduces the voltage to a proper package voltage. Then, the power is distributed through the motherboard to the package, where controlled collapse chip connections (C4s) —also known as flip chips— connect the package with the micro-processor connection pads to distribute the power inside the chip. Finally, the on-chip PDN distributes the power across all the devices in the micro-processor.

In an ideal scenario, the on-chip devices would observe the voltage level generated by the VRM, V_{nom}. However, the

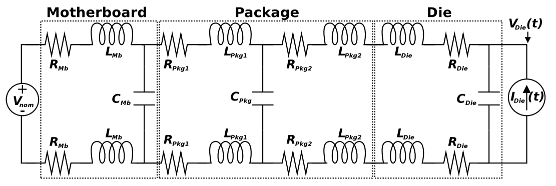


Figure 2: Model of a power distribution network (PDN).

parasitic components present in the PDN produce variations in the voltage supply level perceived by the devices (V_{Die}). Figure 2 shows a simplified model of the power distribution network that includes the components (resistances and impedances) present on the motherboard, package and die. Resistive components (R_{Mb}, R_{Pkg1}, R_{Pkg2}, R_{Die}) produce a voltage droop across the network referred to as *IR-droop* (V_{droop}), whereas inductive components (L_{Mb}, L_{Pkg1}, L_{Pkg2}, L_{Die}) produce voltage fluctuations across the network. We refer to these fluctuations as *didt-droop* (V_{didt}), which in conjunction with the V_{droop} constitute the *voltage noise* (V_{noise}). Thus, the $V_{Die} = V_{nom} - V_{droop} - V_{didt} = V_{nom} - V_{noise}$.

The maximum magnitude that V_{noise} can achieve defines the voltage margin (V_{margin}) required for a reliable operation. Designers mitigate the V_{noise} magnitude by adding decoupling capacitance —also known as *decap*— at different points in the PDN (C_{Mb}, C_{Pkg} and C_{Die} in Figure 2) [6]. However, since the amount of decap is limited by area constraints, the V_{noise} cannot be nullified. In this work, we focus our analysis in characterizing the Δ in V_{Die} (V_{noise}) depending on the activity being generated in a multi-core processor ($I_{die}(t)$).

B. Fundamentals

In order to understand the reasons of such V_{noise}, we have to review some fundamentals on circuit theory. In reactive RLC circuits with time-varying signals —like the PDNs— Ohm's law is generalized to:

$$\Delta V_{Die} = V_{noise} = \Delta I \cdot Z \tag{1}$$

where the Z denotes the impedance, i.e. the complex generalization of resistance. Z can be further decomposed to:

$$Z = \sqrt{R^2 + (X_L - X_C)^2} \tag{2}$$

where R is the resistance in ohms of the resistor elements, X_L is the reactance of the inductor elements and X_C is the reactance of the capacitor elements. Given the following definition of reactance of the inductor and capacitor elements that assumes a sinusoidal input signal[2]:

$$X_L = 2\pi \cdot f \cdot L \tag{3}$$
$$X_C = (2\pi \cdot f \cdot C)^{-1} \tag{4}$$

where f is the frequency of the signal, L is the inductance in henries, C the capacitance in farads; we observe that both X_L and X_C depend on the input signal frequency. As a result, for a given R, L, C, the V_{noise} can be expressed as a function of the ΔI generated and the frequency (f) at which this ΔI is generated.

[2]Fourier analysis can be used to approximate any arbitrary signal as a sum of sinusoids.

The resonance frequency (f).: Parallel RLC circuits have the property of resonance at specific frequencies. Resonance is the disposition of the circuit to have higher magnitude oscillations for some frequencies than others. Resonance occurs because energy is kept in two distinct ways in the circuit: in the magnetic field of the inductors and in the charge of capacitors. The rate at which the energy is transferred from one to the other in a periodic fashion and where the magnitude of oscillations is maximum is referred to as resonant frequency. The PDN model shown in Figure 2 can be decomposed into a set of parallel RLC circuits. Therefore, this results in various resonant frequencies:

$$f_r = (2\pi\sqrt{L \cdot C})^{-1} \tag{5}$$

that are the consequence of the interaction of the different inductances and decoupling capacitances present in the PDN. One effect of stimulating a parallel RLC circuit at resonant frequency is that the impedance (Z) is maximized. As a result, the V_{noise} generated is also maximized (Equation 1).

Since the activity in the device can eventually generate ΔI at such resonant frequencies, the impedance Z of the system must be defined not only based on the operating clock frequency, but also on the spectrum of frequencies where current fluctuations can exist. This definition is done during the package design process, when PDN impedance (Z) profiles and decap maps are generated. In that process, package designers ensure that a target maximum impedance Z is not surpassed for any given frequency by placing enough decaps in parallel. This guarantees that V_{noise} remains within a constrained magnitude, allowing for affordable and reliable voltage margins (V_{margin}).

Then, one could define the V_{margin} based on impedance Z profiles generated during the package design process [2], [14]. However, that would lead to a pessimistic design since, after all, the V_{margin} defined by this process is based on an in-lab generated worst-case maximum ΔI. Therefore, to improve the efficiency of the design, V_{margin} must be squeezed to the limits using possible real-world worst-case situations, but without impacting the reliability of the design.

According to the formulas mentioned above, in order to check the reliability of the design it is sufficient to validate that the voltage noise is within the limits ($V_{noise} < V_{margin}$) when a maximum ΔI event is generated at resonant frequency. This is done through stressmarks, which are specifically tailored to generate such (possible but improbable) worst case conditions. Generating them is a challenging task that gained focus in recent years [25], [26], [28]. In this work, besides presenting a methodology to generate these stressmarks, we focus our discussion on exploring, in detail, the effects of the ΔI magnitude and the ΔI frequency on the V_{noise}.

C. V_{noise} in Multi-core Processors

In the previous section we have seen that the V_{noise} can be expressed as a function of the ΔI generated by the activity in the processor and the frequency (f) at which this ΔI is generated. Ideally, in a multi-core processor, this could be expressed as:

$$V_{noise_{proc}} = \max_{0 \le i \le \#cores}(V_{noise_i}) = \max_{0 \le i \le \#cores}(\gamma_i(\Delta I_i, f_{\Delta I_i})) \tag{6}$$

where the total $V_{noise_{proc}}$ is defined as the maximum of the local noise V_{noise_i} generated in each core. This local noise is a function (γ_i) of the ΔI_i that the core generates and the frequency at which it is generated ($f_{\Delta I_i}$). This definition assumes that noise on each core is independent from each other. However, this is not accurate since the on-chip PDN is shared across the cores in order to minimize the noise [21]. Therefore, a more realistic definition of the local noise on a given core i (V_{noise_i}) would be:

$$V_{noise_i} = \gamma_i(\Delta I_i, f_{\Delta I_i}) + \eta_i(V_{noise_j}, \forall j \neq i) \tag{7}$$

where the interaction of local noises generated by different cores (second element of the formula) is taken into account. Notice that we define noise function γ and interaction function η on a per core basis in order to express that different cores can show different noise levels and interactions depending, for instance, on their physical layout or due to process variations. This simple formulation of a much more complex system provides a high level view of the complexity of performing a sound *voltage noise* characterization of a multi-core processor.

In this work, we focus our analysis in understanding the V_{noise} depending on the activity being generated on each core in the processor. Specifically, we first examine in detail the different parameters of interest in noise generation: (a) the amount of ΔI, (b) the frequency of ΔI, (c) the alignment of ΔI events generated on each core, and (d) the number of consecutive ΔI events generated. Then, we analyze the interactions of local noises generated by different cores in order to identify possible trends resulting from a particular physical layout or decap placement. This is done by analyzing data gathered from noise measurement macros implemented on each core of the processor. This is the first work performing such local vs. global noise characterization on a production system. This type of characterization efforts provide a fundamental piece of data to researchers and system architects in order to define the requirements of voltage noise mitigation mechanisms [4], [15], [27], [29], [49].

III. Experimental Framework

This section describes the experimental platform as well as the different methods used to gather the experimental data presented in this paper. Unless otherwise stated, experiments have been run on different processors multiple times to check their reproducibility, and arithmetic average values are reported.

Evaluation platform: The experimental platform is an IBM zEnterprise$^{\text{TM}}$ EC12 system (zEC12) [44], [51]. Each processor chip (CP) in the system contains six super-scalar out-of-order processor cores running at 5.5 GHz. Each core contains private instruction and data L1 and L2 SRAM caches with 256B cache lines. The center of the chip contains a 48 MB eDRAM L3 cache that is shared by all 6 cores. The DDR3 interface is on the left side of the chip (MCU), and the I/O bus controller (GX) is on the right. Figure 3 shows the chip layout. For the purpose of this work, various CP chips of zEC12 systems were measured.

Software environment: When running experiments, the platform runs SUSE Linux Enterprise Server 11 SP3 with linux kernel version 3.0.82-0.7. This version provides the standard PCL API [8] to access hardware performance counters. We use this standard interface to gather performance counter data to assess the generation of dI/dt stressmarks.

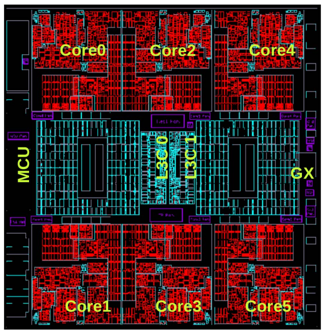


Figure 3: IBM zEC12 CP chip layout. Each unit, i.e. the cores, the MCU, the GX and the nest, implements a skitter macro that allows measurement of voltage noise.

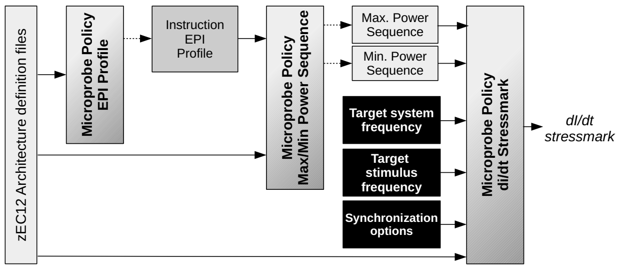
Voltage control: Each IBM zEC12 system is provided with a management console, i.e. a service element, from which control and monitoring commands can be sent to the different devices in the system. Specifically, chip voltage can be controlled in steps of 0.5% of the nominal voltage. This fine-grained granularity enables accurately finding the failure point when errors start to occur. Errors are detected using the recovery unit (R-Unit) that is implemented in these highly reliable systems [45], [52].

In order to find the available voltage margin, i.e. the voltage at failure [26], [27], the operating voltage is slowly decreased until first failure happens. This process, commonly named V_{min} experiment, is the ultimate bullet-proof method to check the available voltage margin. The drawback of this method is its long turn-around time. This is because voltage is decreased slowly —0.5% every two minutes— until first failure. In addition, for each experiment, the system performs failure checks and has to be rebooted. Therefore, since we had other means to evaluate the voltage noise generated (see below), we only used this mechanism to double-check the stressmarks generated and to gather the extra system information, e.g. the circuit paths within the design macros that fail first.

Power measurement: The service element can monitor the power consumption of each device in the system. Specifically, power measurements are done at the chip level by reading the current and voltage of the input rails. The granularity of the readings is in milliwatts. This power consumption data has been used extensively to assess the generation of the dI/dt stressmarks.

Voltage noise measurement: The measurement of the voltage noise on the different cores of the system is done through the skitter macros implemented in the different units of the zEC12 CP chip [13], [42]. We also performed oscilloscope measurements of key experiments to confirm the voltage behavior. Next, we provide an overview about the skitters. We refer the reader to [13], [42] for an in-depth explanation of their operation and usability.

Skitter macros contain a latched-tapped delay line of 129 inverters, for the best timing resolution, with a nominal delay between 5 and 8 ps depending on the threshold voltage and technology. These delay lines and sampling latches form an edge-capture circuit that captures the rising and falling edges of the clock signal. The sampling latches take a snapshot of the state of the inverter chain every cycle, forming a 129 bit string of 0's with 1's where the edges are detected. If the inverter delays are constant and there is no clock jitter, then the clock edges (the 1's) are captured in the same latches (same bit in the string). When there is clock jitter, the location of the captured edges varies.


zEC12 Architecture definition files
Microprobe Policy EPI Profile
Instruction EPI Profile
Microprobe Policy Max/Min Power Sequence
Max. Power Sequence
Min. Power Sequence
Target system frequency
Target stimulus frequency
Synchronization options
Microprobe Policy dI/dt Stressmark
dI/dt stressmark
Figure 4: dI/dt stressmark generation methodology. First, the EPI profile of the target platform is generated. Then, the profile is used to obtain max./min. power instruction sequences. Finally, instruction sequences, system and stimulus frequencies, and synchronization options are used to generate the desired dI/dt stressmark.

Besides measuring the clock jitter, skitter macros implicitly take into account the effect of voltage changes on the clock jitter and skew. This is because the delay line is very sensitive to voltage changes. In zEC12 processor chip, skitters were placed in locations that were known to have the potential for large ΔI events. Additional skitters were then placed to add visibility throughout the design. In this study, we provide results for the skitter macros implemented on the cores.

Skitter macros can be configured to run in sticky-mode. In that case, the latches record all locations where one or more clock edges occur within a period of time. This allows the worst-case timing uncertainty (noise) to be measured while running any application. We used this mode for measuring worst-case voltage noise while running our dI/dt stressmarks.

Results of skitter macros are reported in percentage peak-to-peak variation (%p2p). The %p2p value is directly related to the voltage droop in the PDN. That is, the higher the %p2p noise, the higher the voltage droop. These characteristics enable us to quickly and reliably evaluate the voltage droops in the system, avoiding the time-consuming V_{min} experiments.

IV. dI/dt Stressmark Generation Methodology

In order to profile the voltage noise of a given architecture, several thousands of dI/dt stressmarks —with different properties, such as the target stimulus frequency, ΔI synchronization method or alignment constraints— have to be generated. Manual implementation is impractical. Therefore, we require a systematic methodology to generate dI/dt stressmarks with the different desired properties.

Figure 4 shows a high-level view of the methodology used to generate dI/dt stressmarks. The methodology uses the Microprobe [3] micro-benchmark generation framework as the underlying infrastructure to generate the dI/dt stressmarks. To that end, before starting this noise characterization effort, a back-end knowledge base for the zEC12 architecture had to be implemented via target definition files [3].

As discussed in Section II, there are various parameters that affect the noise generated. One parameter is the magnitude of

Rank #	Instr.	Description	Power
1	CIB	Compare immediate and branch (32<8)	1.58
2	CRB	Compare and branch (32)	1.57
3	BXHG	Branch on index high (64)	1.57
4	CGIB	Compare immediate and branch (64<8)	1.55
5	CHHSI	Compare halfword immediate (16<16)	1.55
1297	DDTRA	Divide long DFP with rounding mode	1.01
1298	MXTRA	Multiply extended DFP with rounding mode	1.01
1299	MDTRA	Multiply long DFP with rounding mode	1
1300	STCK	Store clock	1
1301	SRNM	Set rounding mode	1

Table I: First and last five instructions in the zEC12 EPI profile. Instruction power is normalized to SRNM instruction.

the core ΔI. In order to maximize it, maximum and minimum power instruction sequences first need to be defined, and then concatenated. Another parameter is the frequency at which core ΔIs are generated, controlling it permits the detection of the resonant regions of the system. One more parameter is the instant at which the core ΔIs are generated, by synchronizing them the overall voltage droop is maximized. Finally, the last parameter that affects the overall noise generated is the number of consecutive ΔI events generated. The rest of the section describes how we generate these parametrizable dI/dt stressmarks.

A. Energy-per-instruction profile

The first step required to produce dI/dt stressmarks is the generation of an energy-per-instruction (EPI) profile [3]. This profile allows ranking all the ISA instructions by their power usage. The rank is used afterwards to select the instruction candidates when searching for maximum and minimum power instruction sequences.

This systematic methodology for generating an EPI profile consists in generating a micro-benchmark for each and every instruction in the ISA. The micro-benchmark skeleton is an endless loop with 4000 repetitions of the instruction, without dependencies. Micro-benchmarks are run for a few seconds and power and performance metrics are gathered. Metrics are post-processed afterwards in order to generate the EPI profile.

The fact that all instructions are profiled provides a complete picture of the ISA characteristics. This is useful to detect potential instruction candidates for low and high power instruction sequence generation that would not be considered by an expert. For instance, in Table I, which shows the first and last five instructions in the instruction ranking, we see the non-intuitive case where a compare immediate instruction (CHHSI) is in the Top 5 of most energy consuming instructions. Besides the detection of these non-intuitive cases, the profile also provides important feedback to optimize future implementations of the architecture.

B. Maximum and minimum power instruction sequences

In order to generate the maximum power instruction sequence, we implement a methodology based on the one presented in [3]. In that work, the authors propose to select instruction candidates by functional unit basis —the top instruction for each functional unit was selected— and then evaluate all the possible combinations of a given length to select the highest power one. We adapt that approach to the extra complexity of the zEC12 CISC architecture.

Figure 5 shows the methodology used to generate the maximum power instruction sequence. It includes the following steps:

1) *Instruction candidate selection*: We use the EPI profile to categorize the instructions by their functional unit usage and issue class. From each category, we select the top most power-consuming instructions. Categories with low power or low IPC[3] are discarded to reduce the number of instruction candidates to nine, avoiding a design space explosion problem.
2) *Sequence candidate generation*: We generate all possible combinations of length six of these nine instructions ($9^6 = 531\,441$). Length six is selected because it is twice the dispatch group size in the zEC12 architecture. Thus, it provides the best trade-off between combinations explored and experimental time.
3) *Microarchitectural filtering*: The 531 441 combinations are filtered using constraints based on the microarchitecture details implemented in Microprobe. For instance, sequences that are known to not have an average dispatch group size of 3 —the maximum possible in zEC12— are filtered out because they will not exhibit a high IPC. Other constraints, such as the maximum number of branches or prefetches per sequence are also added. This results in a reduction of the design space from 531 441 sequences to 32000.
4) *IPC filtering*: Since it is still not practical to evaluate the power of the 32000 remaining candidate sequences, we filter them based on their IPC. We select the top thousand sequences showing the highest IPC. Two reasons motivated us to choose IPC filtering: (a) it is well-known that IPC is directly related to power and, (b) IPC evaluations are much faster than the power ones because: (i) it is possible to run them in parallel using different cores and machines and, (ii) a run of a few seconds is enough to obtain the IPC. This is much faster than the power evaluations, which have to be done on the same processor with the same experimental conditions (e.g. chip temperature) between runs for a fair comparison.
5) *Power evaluation*: We evaluate the power consumption of the remaining thousand sequence candidates. The sequence showing the highest power is selected as the maximum power instruction sequence. We validate the sequence on different processors to confirm its high power consumption.

Finally, we also rely on the EPI profile to define the minimum power sequence. We select the last instruction of the instruction rank as the minimum power sequence. Note that, the no-operation instruction (*nop*) is not the optimal candidate. Instead, long-latency instructions (such as divisions or decimal instructions) are better candidates because they stall all parts of the processor. The EPI profile provides an evidence of that fact, confirming previous findings [22].

[3]We refer to IPC as the micro-operations executed per cycle, which in a CISC architecture might differ from the general IPC definition of instruction committed per cycle.

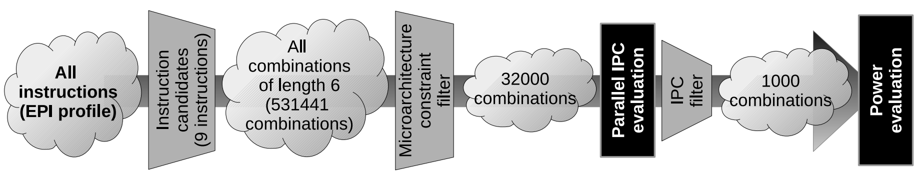

All instructions (EPI profile)
Instruction candidates (9 instructions)
All combinations of length 6 (531441 combinations)
Microarchitecture constraint filter
32000 combinations
Parallel IPC evaluation
IPC filter
1000 combinations
Power evaluation

Figure 5: Maximum power instruction sequence generation methodology. First, instruction candidates are selected from the EPI profile. Second, several combinations are generated and filtered using microarchitecture constraints. Then, IPC is evaluated to select the top thousand instruction sequence candidates which will proceed to the power evaluation stage.

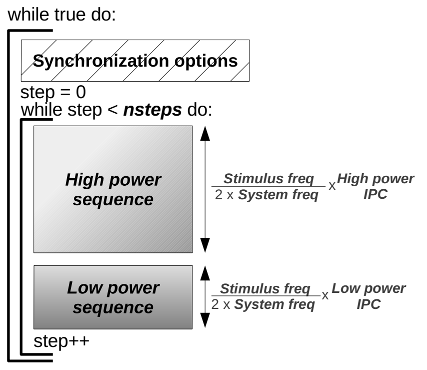


Figure 6: dI/dt stressmark skeleton. Rich number of configurable parameters to control the noise generation: number of steps, synchronization options, low and high power instruction sequences and their IPCs, stimulus frequency and target system frequency.

C. dI/dt stressmark generation

In order to understand in detail the voltage noise in multi-core systems and present studies such as the ones presented in Sections V and VI, several dI/dt stressmarks with different properties are needed. For instance, we need dI/dt stressmarks generating different amounts of ΔI, with different stimulus frequencies —i.e. duration between ΔI events—, with different number of consecutive ΔI events, with specific alignment between ΔI events, etc. Therefore, the dI/dt stressmark generation should be highly configurable.

We have implemented a dI/dt stressmark generation policy on top of the Microprobe framework. As shown in the last step of Figure 4, the policy takes as input the high and low power sequences and their IPCs, the target system frequency, the target stimulus frequency, the number of consecutive ΔI events to generate and the synchronization options. With that information, one can derive the length of high and low power sequences to generate low/high activity at the given stimulus frequency. Then, high and low power sequences are concatenated within a loop to generate a dI/dt stressmark as shown in Figure 6. We explored the addition of intruction dependencies between high and low power sequences to ensure a sharper activity change but results were similar.

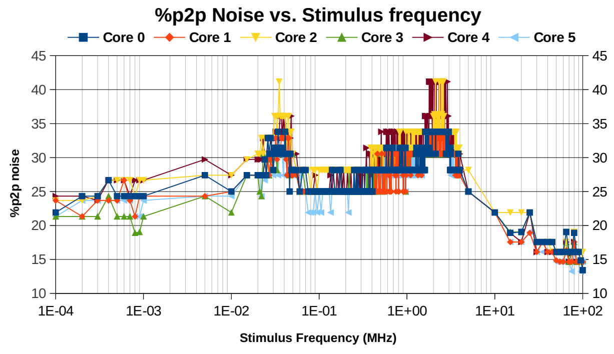
As explained in Section II-C, an important challenge to overcome in multi-core systems is how to synchronize the stressmarks running on different cores. This is required to maximize the ΔI generated in a given instant. Although probabilistic approaches exist to ensure an eventual alignment of ΔI events within a time window [26], we implemented a deterministic approach. We leverage the advanced state-of-the-art timing facilities present in the zEC12 architecture to perform perfect cycle-accurate alignment within the stressmarks. The timing facilities provide a global 64-bit Time-Of-Day (TOD) register that allows for the control of the misalignment between stressmarks in steps of 62.5 nanoseconds. This cycle-accurate capability of controlling the misalignment permits the analysis of the sensitivity of noise to ΔI event misalignments presented in Section V-B. Note that without the right architecture support the perfect control of alignment would not be possible.

During the dI/dt stressmark generation methodology definition, we also studied the introduction of disruptive (e.g. branch/cache/TLB misses) events and memory hierarchy activity to maximize the ΔI generated. However, we decided to generate core-contained sequences without disruptive events because: (a) disruptive events showed small differences in power consumption with respect to the minimum power sequence (b) the introduction of memory activity in the maximum power sequence did not improve the maximum power significantly; (c) disruptive events and memory activity in shared resources limit the capacity to control the stimulus frequency generated by the workload in a multi-core context.

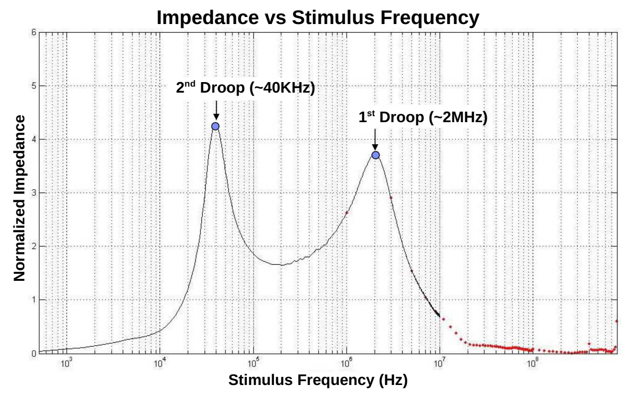
Overall, the flexibility of dI/dt stressmark generation methodology allows us to generate any type of dI/dt stressmarks by controlling every relevant aspect involved in the noise generation: the magnitude of the ΔI, the frequency of ΔI events, the number of consecutive ΔI events and the alignment between the ΔI events. This 'white-box' approach provides to the user all the knobs to control the noise. This approach complements the existing solutions that are only focused in finding the worst-case noise scenario. It would be possible to implement optimization algorithms —such as the genetic algorithms employed in previous works [26]— on top of the presented solution. The following sections present detailed noise analysis that would not be possible without such dI/dt stressmark generation flexibility.

V. Understanding voltage noise

In this section, we present sensitivity analyses for each parameter related to the voltage noise. This permits us to: (a) validate the dI/dt stressmark generation policy, (b) gain detailed insights on noise behavior in multi-core systems, (c) understand the conditions where worst case noise scenario occurs, and (d) confirm the robustness of the zEC12 design. Although we provide per-core noise measurements, this section analyzes the global behavior of the noise at chip level. Section VI studies the noise propagation across cores and the inter-core interactions. These detailed characterizations are used to motivate the optimization opportunities presented in Section VII.



(a) Maximum per core noise measured when running the dI/dt stressmarks with different stimulus frequencies



(b) Post-silicon impedance (Z) profile generated via direct measurements during package characterization

Figure 7: Noise sensitivity to stimulus frequency. The detected resonance bands in (a) are in accordance with the Z profile (b).

A. Noise sensitivity to stimulus frequency

The first question to answer is if the skitter-measured noise generated by the dI/dt stressmarks is magnified at the expected resonant bands. For this purpose, we explore a wide range of stimulus frequencies. For each stimulus frequency, we generate a dI/dt stressmark without synchronization and we run one copy of it on each core.

Figure 7a shows the per-core noise measurements for different stimulus frequencies. Note that the step function of noise readings is due to the discrete nature of the skitter bit-string. The measurements confirm two resonant bands around the 40KHz region and 2MHz region, respectively. This agrees with the post-silicon impedance profile generated via direct measurements during package testing process (Figure 7b).

The six cores show the same trends in the entire range of the frequency spectrum. This is expected because the PDN is shared and therefore, the noise is propagated across cores. Nevertheless, some cores exhibit more noise than others. For instance, the maximum noise seen is ∼41%p2p variation in cores 2 and 4. These differences in the noise seen on each core are mainly due to manufacturing process variation, although —as we present later on— some other factors such as the physical layout or inter-core noise propagation can also contribute.

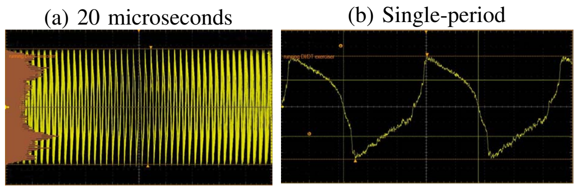


Figure 8: Oscilloscope shot of voltage noise on core 0 when executing the maximum dI/dt stressmark at ∼2MHz.

One important thing to note is the displacement of the 'first droop' to the ∼2MHz region[4]. The traditional 'first droop' in older systems was at frequencies around 30-100 MHz and it was caused by the charge exchange oscillation between the chip and the surrounding module capacitors. The path inductance between chip and module capacitors is fairly small, and therefore the 'first droop' oscillation time is short. Modern IBM systems use deep trench technology for implementing the large on-chip eDRAM. This augmented the on-chip capacitance by 40x. As a result, the resonant bands shifted to a much lower frequency, around the ∼2MHz region in Figure 7a. The outcome is that there is no longer an oscillatory power noise behavior at frequencies above 5 MHz, making the design more robust.

Finally, although the noise levels seen are confirming the correctness of the dI/dt stressmarks generated, we performed oscilloscope measurements to double-check the activity being generated. Figure 8a shows oscilloscope measurements of the voltage in core 0 when running the noisiest dI/dt stressmark (∼2MHz). The repetition of the sinusoidal form in the 20 microseconds shot confirms the correctness of the stressmark. A single period is shown in Figure 8b. The measurements consistently show large peak-to-peak variations in the voltage supply. Despite these large noise events, the zEC12 platform did not trigger any recovery mechanism, confirming its robust design.

B. Noise sensitivity to alignment

In this section, we repeat the same experiment but adding the synchronization mechanism in order to evaluate the effect of synchronization on the ΔI events happening on each core. We expect that synchronizing the ΔI events will increase the voltage noise but we would like to quantify, empirically, that increase in order to discern the relative importance of the synchronization effect versus other parameters.

We generate dI/dt stressmarks with synchronization for a wide spectrum of stimulus frequencies and we run a copy of each stressmark on every core. The synchronization code is configured to loop until the low-order bits of the clock value (TOD register) are zero; this happens every 4ms. After that, the dI/dt loop is executed for a thousand iterations (thousand ΔI events) before re-synching again. This ensures the stressmarks are in sync and minimizes the synchronization overhead. We evaluate the noise sensitivity to the number of consecutive ΔI events in Section V-E.

Figure 9 shows the noise perceived by the cores. As expected, the noise increased significantly confirming the effectiveness of the synchronization mechanism. We see a general increase of around 20 %p2p points. For instance, the maximum noise

[4]One can also say that the 'first droop' disappeared.

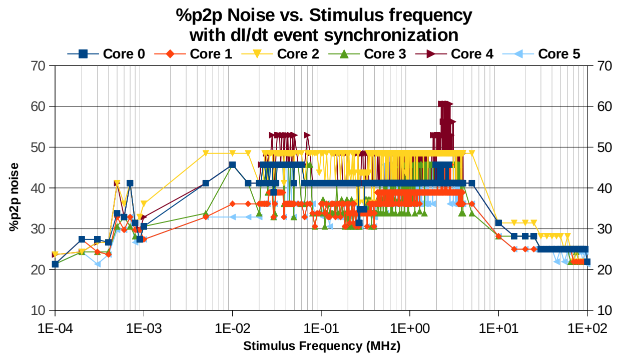


Figure 9: Maximum per core noise measured for different stimulus frequencies when running the dI/dt stressmarks that are synchronized every 4ms. Synchronization enlarged the noise levels seen in Figure 7a.

seen at the resonant region around the 2MHz point was 41% (Figure 7a), while with synchronization it is 61% (Figure 9). The noise is not only magnified around the resonant regions. It is also magnified in the rest of the stimulus frequencies explored. Actually, we measured higher noise in non-resonant bands with synchronization enabled than in the resonant bands without the synchronization (region between 40KHz and 2MHz stimulus frequencies in Figure 9). This is because the effect due to the large ΔI events resulting from synchronization exceeds the effect of resonance. This suggests that ΔI event synchronization is a more important factor to avoid than ΔI events occurring at resonance frequency.

C. Noise sensitivity to misalignment

Since ΔI event synchronization is an important factor to consider, we want to evaluate 'how much' misalignment is necessary to diminish the synchronization effect. By quantifying, empirically, the time window where ΔI events should not happen simultaneously, one can define the requirements of noise reduction mechanisms on future architecture generations.

For this experiment, we generate dI/dt stressmarks with a stimulus frequency of 2MHz (within the resonant band) and synchronizing every 4ms after a thousand ΔI events. However, in this case, we generate different versions varying the exit condition of the synchronization loop. For instance, one stressmark exits the synchronization loop when bits 39–63 of the TOD register are zero, while the other does so when bits 39–62 of the TOD register are zero and bit 63 is one. This small modification ensures a 62.5ns misalignment between the stressmarks.

Figure 10 shows the average %p2p noise seen for each core versus the maximum allowed misalignment. To create a given maximum allowed misalignment, the stressmarks are distributed evenly within the misalignment range. For instance, for a maximum allowed misalignment of 125ns, 2 stressmarks are synchronized at $t = 0ns$, 2 at $t = 62.5ns$ and 2 at $t = 125ns$. Since there are multiple possible stressmarks to core mappings, we execute all of them and report the average values. The results show that a small misalignment of 62.5ns is sufficient to reduce the noise to levels similar to the ones without synchronization. For this study, we were limited by the 62.5ns granularity ensured by the particular implementation of

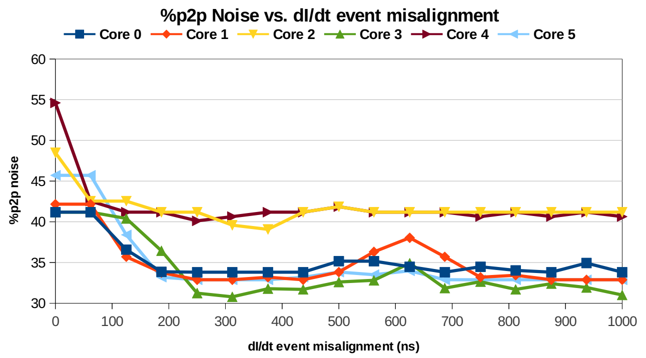


Figure 10: Sensitivity of noise to ΔI event alignment. X-axis shows the maximum misalignment allowed between the stressmarks. A small misalignment is sufficient to diminish the synchronization effect.

the architecture, but it is likely that smaller misalignments, in the order of a few nanoseconds, will be enough to nullify the alignment effect.

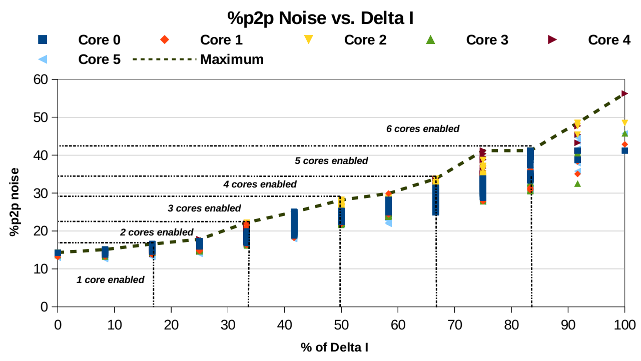
In conclusion, the fact is that synchronized ΔI events generate large voltage noise but the strict alignment requirement considerably reduces their likelihood to happen. Moreover, this likelihood will decrease in the future with the addition of more cores to the system. Nevertheless, a platform with such high reliability standards as the zEC12 cannot take the risk of not taking into account these improbable but possible noise events.

D. Noise sensitivity to ΔI

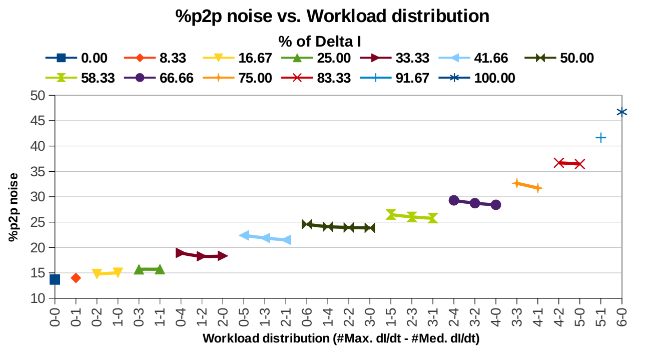
As explained in Section II, voltage noise is directly related to the amount of ΔI generated in a given instant. In this section we quantify that relation. For this experiment, we use three types of workloads: nothing (idle), medium dI/dt and maximum dI/dt. The dI/dt ones have a stimulus frequency of 2MHz and use the synchronization mechanism in order to maximize the noise. To generate the medium dI/dt stressmark we use an instruction sequence that consumes exactly the average between the maximum and the minimum power sequence. In other words, two medium dI/dt stressmarks generate the same ΔI as one maximum dI/dt stressmark. We run all possible workload to core mappings ($6\ cores\ \&\ 3\ workloads \Rightarrow 3^6$ combinations) and gather skitter noise measurements.

Figure 11a shows the maximum noise generated with respect to the percentage of the maximum possible ΔI that can be generated. As expected, the noise grows with the amount of ΔI. This quantification is crucial to accurately define the requirements of noise reduction mechanisms. For instance, if we want to implement a noise mitigation mechanism to keep %p2p noise below 30%, we should not allow more than 60% ΔI.

In addition, we want to know if the way the ΔI is generated affects the noise. That is, we want to answer the following question: *what workload distribution creates more noise? A single stressmark generating a given* ΔI *or two stressmarks generating* $\Delta I/2$*?* Figure 11b shows the results of Figure 11a organized by the workload distribution. In this case, noise is averaged across cores and mappings with the same workload distribution and ΔI. The results depict a trend showing that when the source of the noise is more spread out, the noise is



(a) Maximum per core noise measurements with respect to the percentage of maximum possible ΔI that can be generated. Each point shows, for a given % of ΔI and core number, the maximum %p2p noise seen across all workload mappings generating that particular % of ΔI. The dotted regions designate the minimum number of cores needed to generate a particular noise level.



(b) Average noise measurements with respect to ΔI generated and workload distribution. For instance, point 1-4 shows the average noise across all possible workload mappings of 1 maximum dI/dt stressmark and 4 medium dI/dt stressmarks (and 1 core idling).

Figure 11: Noise sensitivity to ΔI. We observe in (a) that noise is directly related to the amount of ΔI and the number of cores enabled. We also detect in (b) a trend showing that when the sour[illegible]

higher. For instance, in Figure 11b we see that, for 50% ΔI, the noise slightly decreases when going from a 0-6 configuration (all cores running the medium dI/dt workload) to a 3-0 configuration (3 cores running the dI/dt workload). Nevertheless, the trend is not significant. This suggests that the important factor is the amount of ΔI generated and not the source of the ΔI.

E. Noise sensitivity to number of consecutive ΔI events

To wrap up this sensitivity analysis section, we evaluate the sensitivity of noise regarding the number of consecutive ΔI events generated. We want to know if a single ΔI event is enough to generate large noise or if we need more. We generate maximum dI/dt stressmarks with different stimulus frequencies and synchronization enabled while varying the number of consecutive ΔI events between synchronization points. We also generate stressmarks for these different stimulus frequencies without synchronization, i.e. ∞ number of consecutive ΔI events. The stimulus frequencies selected are at resonant bands (35KHz and 2.5MHz) and their surroundings. For this last sensitivity experiment, instead of skitter measurements we run V_{min} experiments in order to measure the available margin empirically.

Figure 12 shows the available margin —the amount of V_{bias}[5] required to get the first failure— per stimulus frequency and number of consecutive ΔI events. Results are normalized to the worst case, i.e. highest V_{bias} to fail. The results show that neither the number of consecutive ΔI events nor the stimulus frequency affect the voltage margin available significantly. This is because the noise generated with just a single synchronized ΔI event is large enough so that the noise amplification due to resonance becomes relatively small. Note that although the skitters showed much higher %p2p noise at resonant frequencies, the fact that we are in the high region of noise —where the linearity between V_{noise} and skitter measurements diminishes [13], [42]— translates this to similar V_{bias} levels. When the ΔI event synchronization is disabled (∞ events/no synch in Figure 12), the margin available with respect to the worst case is more

[5]$V_{min} = V_{bias} \times V_{nominal}$

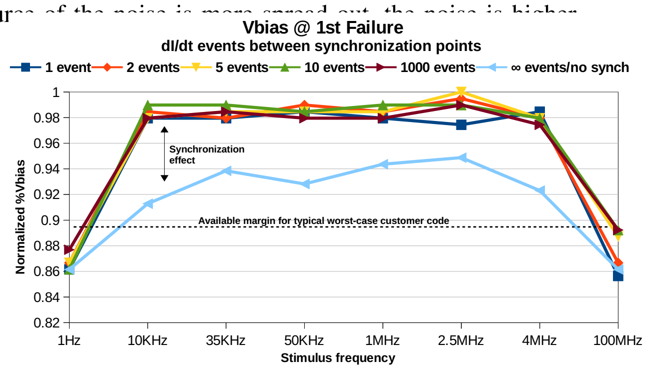


Figure 12: Available margin for different number of consecutive ΔI events and stimulus frequencies. The synchronization of ΔI events is essential to reduce the available margin. The stimulus frequency is a secondary factor.

than doubled (from 0–2% range to 5–7%). This confirms the importance of alignment in the noise formation. Finally, 1Hz and 100MHz frequency points show higher margins because either the stressmarks are misaligned[6] or the stimulus frequency is too high to generate ΔI events.

In order to put these available margins in context, we depict a line showing what we would consider the worst case available margin for a typical customer code. The extrapolation assumes the following: (a) ΔI events are not synchronized —synchronized ΔI events are very unlikely to happen— and (b) the magnitude of the ΔI events generated on each core is around $\sim$80% of the maximum possible ΔI. This is based on the fact that, historically, maximum power stressmarks showed $\sim$20% higher than worst case regular user codes. Therefore, in this context, there is plenty of margin for optimization opportunities. One can apply dynamic guard-banding to reduce the margins dynamically (like in POWER7/POWER7+ systems [11], [12], [29]) or noise reduction techniques to reduce the maximum noise that can be generated by a workload [27].

[6]The fact that frequency (1Hz) is much larger than the synchronization interval (4ms) leads to stressmarks being aligned at different 4ms interval points.

F. Sensitivity summary

We quantified the sensitivity of noise with respect to different parameters. We confirm that the shift of the 'first droop' to lower frequencies —around the 1-5MHz range— is due to the on-chip capacitance increase because of the implementation of large eDRAMs using deep trench technology. This could potentially affect some prior works on noise reduction techniques that are too tailored to reduce high frequency noise events (30-100MHz range).

In addition, we conclude that the key enabler for noise generation is the amount of ΔI generated in a given instant of time. The synchronization of ΔI events on different cores is crucial in that regard. If a mechanism is implemented to avoid the synchronization of ΔI events happening on different cores, the noise can be reduced by 2-3x. The results also show that any mechanism implemented to reduce the noise should be implemented on a chip-wide basis. Two observations support that fact: (a) large intra-core ΔI events happening on a few number of cores do not lead to high noise and, (b) relatively small ΔI events happening simultaneously to all cores could lead to a large V_{noise}. This global requirement for noise reduction techniques imposes significant implementation challenges since prediction/detection and reduction of ΔI events should be done globally. The next generation processor chip for System z mainframes will include a mechanism to globally monitor/reduce noise if necessary.

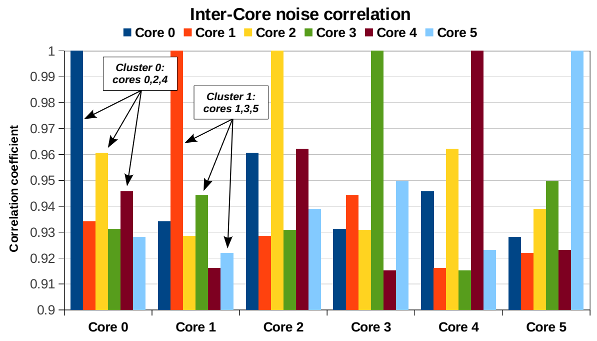
Finally, we also confirm experimentally that the stimulus frequency and the number of consecutive ΔI events are secondary factors. We do not see any value in implementing mechanisms to avoid resonance or a given number of consecutive ΔI events if the main factors (magnitude of ΔI and synchronization of ΔI events) are not solved first. Throughout this section, we analyzed the noise globally and we proved the robustness of the zEC12 power delivery system. The next section provides detailed analyses on inter-core noise propagation.

VI. Noise propagation on multi-cores

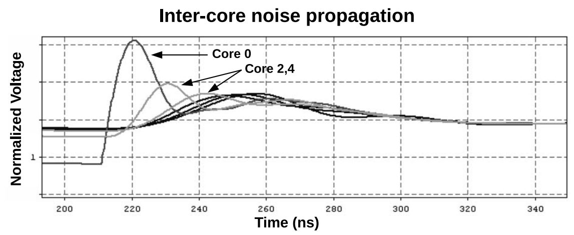
In this section, we analyze how the noise is propagated in a multi-core system. We want to discern all patterns (if any) that could arise as a result of the system design such as core placement, decap placement or overall chip layout. Detecting such patterns is important for: (a) validating the design, e.g. ensuring that there is no particular core/region more vulnerable to noise, (b) detecting optimization opportunities. For example, if the current delivery system is not robust and some cores are more correlated in terms of noise, then one can avoid scheduling tasks on them at the same time if other cores are available. This would reduce the worst-case noise.

To evaluate the inter-core noise relations, we use the same experimental setup as in Section V-D. We also evaluate all the possible workload mappings to cores. This provides the complete picture of possible core activities. By performing statistical analyses on the dataset, one can detect trends or anomalies.

First, in Figure 13a, we compute the correlation factor between the noise seen in all the possible mappings for each pair of cores. This tells us if some cores are more related to others. As expected, inter-core noise correlation factors are high (>0.91).



(a) Correlation coefficient between noise observed by each core for all possible workload mappings.



(b) Simulation of a ΔI event on core 0 confirming that the noise from core 0 is transferred faster and more strongly to cores 2 and 4 than to the other cores.

Figure 13: Inter-core noise propagation. We detect the existence of two noise-related core clusters resulting from the chip layout.

This is because noise is globally propagated across the PDN. Therefore, if there is noise in the PDN, all the cores are affected. However, we also see that some cores are more correlated than others.

As shown in Figure 13a, we detect two clusters of cores: cores 0,2,4 and cores 1,3,5. Two main reasons explain this result: First, if we revisit Figure 3, the clusters correspond to the upper three cores and the lower three cores, respectively. Since the L3 —with a relatively large capacitance— sits between the clusters, it slightly isolates the noise from one cluster to another, acting as a damping element. Second, the PDN topology also affects how the noise is propagated. The two clusters are on different on-chip voltage domains sharing a single package voltage domain.

In order to confirm these findings, we use our in-house PDN simulation set up based on Cadence/Sigrity Speed 2000 software [5]. We evaluate the effects of a large ΔI event on Core 0, while the other cores are idling. Figure 13b shows the simulated voltage on the six cores over time. We see that the noise in the cores 0, 2, 4 on one side of the chip is larger than the noise in the cores on the opposite side of the chip (cores 1,3,5). This correlates with our observations. Moreover, we also see that the noise from core 0 is transferred faster to cores 2 and 4 than to the other cores.

We show an example of how these inter-core relations can affect the noise generated in Figure 14. The figure shows two different mappings of three dI/dt stressmarks on the six-core processor. In Figure 14a, the dI/dt stressmarks are mapped on cores 1,4,5 and we see a worst-case noise of 24.6 %p2p on core 1. In contrast, if we map the three dI/dt stressmarks in one of the clusters detected (as shown in Figure 14b), the worst

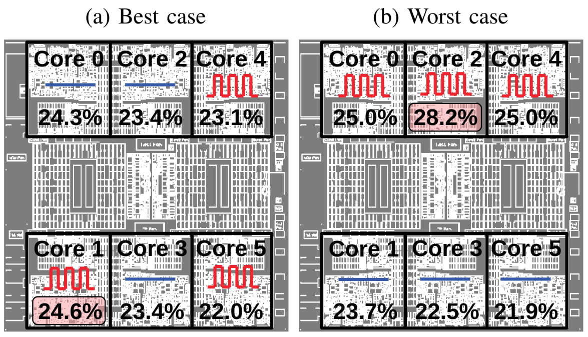


Figure 14: Two different mappings of 3 worst-case dI/dt stressmarks. Each core shows the workload and the %p2p noise. Worst-case noise is highlighted on each mapping to spotlight that the worst-case noise depends on workload to core mapping.

case noise increases to 28.2 %p2p (on core 2). The main reason for this effect is the clustering relation mentioned above. In addition, the noise seen in core 2 of Figure 14b is amplified because it sits in the middle of two other 'noisy' cores (core 0 and 4).

To summarize, we have seen that technology and physical layout as well as other PDN design parameters —such as the number of power domains— can affect how the noise is propagated across the different cores in a multi-core chip. The amount of propagation depends on the distance/capacitance between elements. Therefore, there are some isolation benefits related to element layout. However, their significance has to be evaluated on each particular design. These interactions will likely be higher in the future due to the higher process variation and number of cores. This suggests that there might be increasing opportunities to decrease worst-case noise via appropriate workload-mapping techniques. The next section discusses some of them.

VII. Optimization opportunities

Based on the knowledge learned during the characterization of noise performed in previous sections, we infer two optimization opportunities: noise-aware workload mapping and utilization-based dynamic guard-banding. In this section, we discuss the potential gains of implementing them in order to build a strong case for further research on the topic. Detailed proposals and evaluations are needed to assess their implementability on future generations of processors. In any case, we believe that they could serve as a guide for future research on noise-related techniques. Note also that, given the robustness of the power delivery system implemented in IBM zEnterprise^{TM} EC12 systems (as shown throughout the paper), we do not foresee the necessity to implement noise reduction techniques for maintaining the system reliability.

A. Noise-aware workload mapping

As we have seen in Section VI, the worst case noise for a given number of workloads is not equal for all possible mappings. That is, depending on the mapping used, the worst case noise would be different. This opens up the possibility of implementing noise-aware workload mapping policies. In other words, one can implement a task mapping policy with the objective of minimizing the worst-case noise. Then, one can proactively squeeze the available voltage margin accordingly or just let other hardware-implemented mechanisms such as critical path monitors [11], [12], [29] to reap the benefits of lower noise automatically.

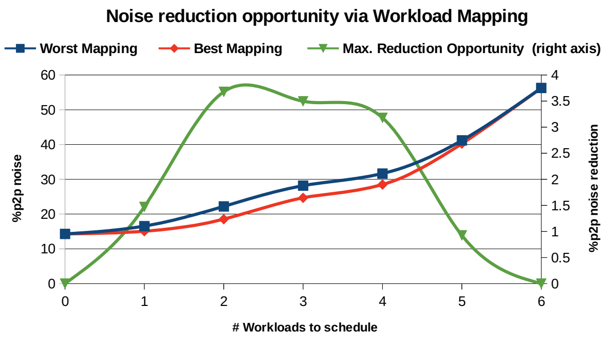


Figure 15: Noise reduction opportunity to reduce worst case noise via workload mapping.

Figure 15 shows the maximum benefits that could arise from noise-aware workload mapping. We compare for each number of workloads (dI/dt stressmarks) to schedule, the mapping showing the highest core noise (Worst Mapping in Figure 15) vs. the one showing the lowest core noise (Best Mapping in Figure 15). The difference between them is shown in the secondary y-axis. When we have 2, 3 or 4 workloads to schedule there are mappings that could reduce the worst-case noise between 2 and 3 %p2p points. For low or high number of workloads to schedule, the benefits are smaller because the potential ΔI is too small or too high. Overall, these results show a small potential for the approach on current systems. However, we foresee that with the increasing number of cores and process variation, the optimization opportunities will increase. This is because the number of possible combinations (mappings) will grow exponentially as well as the variation among them.

B. Utilization-based dynamic guard-banding

From Figure 11a, it is clear that the worst case noise is directly related to the amount of ΔI generated. At the same time, the amount of ΔI that can be generated is bounded by the number of cores that are executing a workload (regions depicted in Figure 11a). If the hardware, or the software managing the resources such as the OS or the firmware, is aware of the number of cores that can execute a workload, then it could safely adapt the available margin accordingly. That is, once a new core is requested to execute some workload, the hardware would raise the voltage to maintain the safety margin because the worst-case noise is higher. Similarly, when a core is freed from execution, the hardware would decrease the voltage to ensure that the margin is not over-provisioned. Although mechanisms such as critical path monitors [11], [12], [29] already do that implicitly, they can benefit from this worst-case bound approach because the dynamic range where they actuate will be tailored to the real worst case for each situation. In the end, the benefits of this simple mechanism depend on the utilization rates of the processor on real environments, with potential huge impact on energy efficiency when the system is not fully utilized.

VIII. Related work

Several works have explored ways to optimize the PDN and minimize voltage noise. This section summarizes some of them.

Architectural solutions: Several architecture-level solutions target the reduction of voltage noise events. Powell *et al.* [35] propose a pipeline muffling technique to avoid transient and large change in resource utilization. In [39], the authors present a voltage noise prediction mechanism, a signature-based noise reduction scheme and an efficient implementation of them based on bloom-filters. Gupta *et al.* [18] presents a delay-commit mechanism that allows rolling-back the execution if noise emergencies have been detected. In [36], the authors use pipeline throttling knobs to change the frequency of voltage noise, avoiding the resonant frequency band. Different works [32], [33], [47] target the problem of minimizing the noise when activating/deactivating, or clock-gating, functional units. In [22], the authors propose techniques to eliminate voltage emergencies based on control theory. The authors extend their work in [23] by using wavelet-based approach for identifying voltage levels at run-time. SMT aspect of voltage noise is studied in [7]. In that work, the authors propose intelligent hardware thread management to control voltage noise. In a broader scope, the authors in [9], [10] propose a reliable architecture that eliminates the need of voltage margins.

Software-based noise reduction: Voltage noise can not only be mitigated using hardware solutions, it can also be reduced by appropriate software control. In [48], the authors propose a noise-aware instruction scheduling algorithm for compilers. Types of noise events are categorized and code transformations to avoid them are proposed in [17]. Dynamic software-based optimization systems are presented in [19], [37]. Based on continuous hardware profiling, the solutions use compiler-based transformations to reduce the frequency of noise events. In [31], the authors propose a synchronization methodology to mitigate the noise related to thread synchronization primitives. Our work foresees two software-based solutions related to voltage noise that were not explored in prior art: noise-aware workload mapping and utilization-based dynamic guard-banding. Although these solutions do not directly target the reduction of noise, they alleviate the guard-banding that is required to be noise-safe.

Noise in production processors: Empirical quantification of noise on real hardware provides fundamental insights and direct data to the research community. Reddi *et al.* [41], [40] analyze the noise behavior on an Intel® Core^{TM}2 Duo processor. They use a technique —based on removing package capacitors— to extrapolate voltage noise in future systems. The extrapolation is used to analytically evaluate a temporal-based thread scheduling policy that minimizes voltage emergencies. In [27], Kim *et al.* analyze the floating point throttling mechanism implemented in AMD Orochi processors. After analyzing the power, performance and reliability trade-offs, they propose a noise reduction solution —controlling, dynamically, the floating-point unit throttling— that increases system performance. In [1], the authors use the error correction mechanisms already built into modern processors to dynamically adapt voltage margins while keeping a safe operating voltage. Our work provides a step forward in the understanding of voltage noise in production processors by providing insights on intra-/inter-core noise behavior, crossing —for the first-time— the chip-level measurement barrier. In addition, we also present a complete characterization of all the parameters related to voltage noise.

dI/dt stressmark generation: Correct dI/dt stressmark generation is crucial to understand the noise behavior [22]. In [28], Kim *et al.* propose a systematic methodology to generate of dI/dt stressmarks based on genetic algorithm-based optimization. In [26], the same authors validate the methodology on different processors, AMD Buldozer and Phenom, by proposing a dithering technique to align the threads in a multi-core system. Our novelty on systematic dI/dt generation is that we provide a fully configurable method to generate all kinds of dI/dt stressmarks, not only the worst case. Besides, we show a deterministic approach leveraging the time facilities to solve the thread alignment issue for multi-core systems, which also allows to control the miss-alignment between threads.

Noise modeling and PDN optimization: It is also important to the community to understand the trends of the noise in future processors in order to anticipate possible limitations. To that end, exploratory PDN and noise modeling are crucial. In [55], the authors model the noise in a future 10nm technology processor running at near-threshold voltages. They highlight the importance of accurate voltage noise characterizations, like the one presented in this paper, to avoid the large guard-band requirements they predict for future near-threshold processors. The same issue is addressed in [53], where the authors use a PDN optimization model to anticipate that future processors will have I/O pads limited due to the increase of pads required to maintain reliable levels of voltage noise. This pad budgeting problem is further analyzed in [50], [54].

IX. Conclusions

This paper addresses the issue of generating complete voltage noise characterizations on current, state-of-the-art, multi-core processors. We presented a configurable 'white-box' methodology to systematically generate dI/dt stressmarks with different characteristics. Then, we showed how that methodology enabled the detailed characterization of the different parameters involved in voltage noise generation. This permits the validation of the power delivery system design and the quantification of the relative importance of each parameter involved in the noise generation. In addition, we presented a noise propagation analysis on multi-core systems, detecting empirically the different noise interactions between the cores. Finally, we discussed potential optimization opportunities emerging from this new understanding of noise behavior.

Overall, we proved the robustness of the power delivery system implemented in IBM zEnterprise^{TM} EC12 systems. The empirical and detailed characterization of the noise behavior enables an accurate definition of the requirements for noise monitoring/mitigation mechanisms.

Acknowledgment

This work has been partially sponsored by Defense Advanced Research Projects Agency (DARPA), Microsystems Technology Office (MTO), under contract no. HR0011-13-C-0022. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S.

Government. This document is: Approved for Public Release, Distribution Unlimited.

Many individuals across IBM made this work possible. We extend particular thanks to Michael D. Campbell, Andrew Z. Muszynski, Jochen Supper, Hubert Harrer, David S. Hutton, Douglas C. Mershimer, Ervin Conn Jr, Karl E. Anderson, Scott B. Swaney, Stanley F. Lutek, Andreas Krebbel and Sreekala Anandavally. Finally, we would like to thank the anonymous reviewers for their comments and feedback.

REFERENCES

[1] A. Bacha and R. Teodorescu, "Dynamic reduction of voltage margins by leveraging on-chip ecc in itanium ii processors" in *ISCA-40*, Jun 2013, Tel-Aviv, Israel.
[2] W. Becker *et al.*, "Mid-frequency simultaneous switching noise in computer systems" in *ECTC-47*, May 1997, San Jose, CA, USA.
[3] R. Bertran *et al.*, "Systematic Energy Characterization of CMP/SMT Processor Systems via Automated Micro-Benchmarks" in *MICRO-45*, Dec 2012, Vancouver, BC, Canada.
[4] K. Bowman *et al.*, "A 22nm dynamically adaptive clock distribution for voltage droop tolerance" in *VLSI*, Jun 2012, Honolulu, HI, USA.
[5] "Cadence/sigrity speed 2000." [Online]. Available: http://www.cadence.com/products/sigrity/speed2000
[6] R. Downing *et al.*, "Decoupling capacitor effects on switching noise" in *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, vol. 16, no. 5, Aug 1993.
[7] W. El-Essawy and D. Albonesi, "Mitigating inductive noise in SMT processors" in *ISLPED*, Aug 2004, Newport Beach, CA, USA.
[8] S. Eranian, "Linux Has a Generic Performance Monitoring API!" *CSCADS Workshop*, Tahoe City, California, USA, Jul 2009.
[9] D. Ernst *et al.*, "Razor: Circuit-level correction of timing errors for low-power operation" in *IEEE Micro*, vol. 24, no. 6, Nov 2004.
[10] D. Ernst *et al.*, "Razor: a low-power pipeline based on circuit-level timing speculation" in *MICRO-36*, Dec 2003, San Diego, CA, USA.
[11] M. S. Floyd *et al.*, "Introducing the adaptive energy management features of the power7 chip" in *IEEE Micro*, vol. 31, no. 2, Mar 2011.
[12] M. S. Floyd *et al.*, "Adaptive energy-management features of the ibm power7 chip" in *IBM JRD*, vol. 55, no. 3, May 2011.
[13] R. Franch *et al.*, "On-chip Timing Uncertainty Measurements on IBM Microprocessors" in *ITC*, Oct 2008, Santa Clara, CA, USA.
[14] B. Garben *et al.*, "Mid-frequency delta-i noise analysis of complex computer system boards with multiprocessor modules and verification by measurements" in *IEEE Trans. on Adv. Packaging*, vol. 24, no. 3, Aug 2001.
[15] A. Grenat *et al.*, "Adaptive clocking system for improved power efficiency in a 28nm x86-64 microprocessor" in *ISSCC*, Feb 2014, San Francisco, CA, USA.
[16] M. Gupta *et al.*, "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network" in *DATE*, Apr 2007, Nice, France.
[17] M. Gupta *et al.*, "Towards a software approach to mitigate voltage emergencies" in *ISLPED*, Aug 2007, Portland, OR, USA.
[18] M. Gupta *et al.*, "Decor: A delayed commit and rollback mechanism for handling inductive noise in processors" in *HPCA*, Feb 2008, Salt Lake City, UT, USA.
[19] K. Hazelwood and D. Brooks, "Eliminating voltage emergencies via microarchitectural voltage control feedback and dynamic optimization" in *ISLPED*, Aug 2004, Newport Beach, CA, USA.
[20] D. Herrell and B. Beker, "Modeling of power distribution systems for high-performance microprocessors" in *IEEE Trans.s on Adv. Packaging*, vol. 22, no. 3, Aug 1999.
[21] N. James *et al.*, "Comparison of Split-Versus Connected-Core Supplies in the POWER6 Microprocessor" in *ISSCC*, Feb 2007, San Francisco, CA, USA.
[22] R. Joseph *et al.*, "Control techniques to eliminate voltage emergencies in high performance processors" in *HPCA*, Feb 2003, Anaheim, CA, USA.
[23] R. Joseph *et al.*, "Wavelet analysis for microprocessor design: Experiences with wavelet-based di/dt characterization" in *HPCA*, Feb 2004, Madrid, Spain.
[24] G. Katopis, "Delta-i noise specification for a high-performance computing machine" in *Proceedings of the IEEE*, vol. 73, no. 9, Sep 1985.
[25] M. Ketkar and E. Chiprout, "A microarchitecture-based framework for pre- and post-silicon power delivery analysis" in *MICRO-42*, Dec 2009, New York, NY, USA.
[26] Y. Kim *et al.*, "AUDIT: Stress Testing the Automatic Way" in *MICRO-45*, Dec 2012, Vancouver, BC, Canada.
[27] Y. Kim *et al.*, "Performance boosting under reliability and power constraints" in *ICCAD*, Nov 2013, San Jose, CA, USA.
[28] Y. Kim and L. John, "Automated di/dt stressmark generation for microprocessor power delivery networks" in *ISLPED*, Aug 2011, Fukuoka, Japan.
[29] C. Lefurgy *et al.*, "Active management of timing guardband to save energy in POWER7" in *MICRO-44*, Dec 2011, Porto Alegre, Brazil.
[30] B. McCredie and W. Becker, "Modeling, measurement, and simulation of simultaneous switching noise" in *IEEE Transactions on Components, Packaging, and Manufacturing Technology*, vol. 19, no. 3, Aug 1996.
[31] T. N. Miller *et al.*, "Vrsync: Characterizing and eliminating synchronization-induced voltage emergencies in many-core processors" in *ISCA-39*, Jun 2012, Portland, OR, USA.
[32] M. Pant *et al.*, "An architectural solution for the inductive noise problem due to clock-gating" in *ISLPED*, Aug 1999, San Diego, CA, USA.
[33] M. Pant *et al.*, "Inductive noise reduction at the architectural level" in *VLSI-13*, Jan 2000, Calcutta, India.
[34] S. Pant and E. Chiprout, "Power grid physics and implications for cad" in *DAC-43*, Jul 2006, San Francisco, CA, USA.
[35] M. Powell and T. N. Vijaykumar, "Pipeline muffling and a priori current ramping: architectural techniques to reduce high-frequency inductive noise" in *ISLPED*, Aug 2003, Seoul, Korea.
[36] M. Powell and T. N. Vijaykumar, "Exploiting resonant behavior to reduce inductive noise" in *ISCA-31*, Jun 2004, Munich, Germany.
[37] V. J. Reddi *et al.*, "Eliminating voltage emergencies via software-guided code transformations" in *TACO*, vol. 7, no. 2, Oct. 2010.
[38] V. J. Reddi *et al.*, "Voltage Noise: Why Its Bad, and What To Do About It" in *SELSE-5*, Mar 2009, Stanford, CA, USA.
[39] V. J. Reddi *et al.*, "Voltage emergency prediction: Using signatures to reduce operating margins" in *HPCA-15*, Feb 2009, Raleigh, NC, USA.
[40] V. J. Reddi *et al.*, "Voltage noise in production processors" in *IEEE Micro*, vol. 31, no. 1, Jan 2011.
[41] V. J. Reddi *et al.*, "Voltage smoothing: Characterizing and mitigating voltage noise in production processors via software-guided thread scheduling" in *MICRO-43*, Dec 2010, Atlanta, GE, USA.
[42] P. Restle *et al.*, "Timing uncertainty measurements on the Power5 microprocessor" in *ISSCC*, Feb 2004, San Francisco, CA, USA.
[43] R. Senthinathan and J. Prince, *Simultaneous Switching Noise and CMOS Devices and Systems*, vol. 249, no. 1, Jan 1994.
[44] C. Shum *et al.*, "IBM zEC12: The Third-Generation High-Frequency Mainframe Microprocessor" in *IEEE Micro*, vol. 33, no. 2, Mar 2013.
[45] T. Slegel *et al.*, "IBM's S/390 G5 microprocessor design" in *IEEE Micro*, vol. 19, no. 2, Mar 1999.
[46] L. Smith *et al.*, "Power distribution system design methodology and capacitor selection for modern CMOS technology" in *IEEE Trans. on Adv. Packaging*, vol. 22, no. 3, Aug 1999.
[47] Z. Tang *et al.*, "Ramp up/down functional unit to reduce step power" in *First International Workshop on Power-Aware Computer Systems*, Nov 2001, Cambridge, MA, USA.
[48] M. C. Toburen, "Power analysis and instruction scheduling for reduced di/dt in the execution core of high-performance microprocessors" Master's thesis, North Carolina State University, Raleigh, NC, USA, Jun 1999.
[49] C. Tokunaga *et al.*, "A graphics execution core in 22nm CMOS featuring adaptive clocking, selective boosting and state-retentive sleep" in *ISSCC*, Feb 2014, San Francisco, CA, USA.
[50] K. Wang *et al.*, "Walking pads: Fast power-supply pad-placement optimization" in *ASP-DAC-19*, Jan 2014, Singapore.
[51] J. Warnock *et al.*, "Circuit and Physical Design of the zEnterprise$^{\text{TM}}$ EC12 Microprocessor Chips and Multi-Chip Module" in *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, Jan 2014.
[52] C. Webb, "IBM z10: The Next-Generation Mainframe Microprocessor" in *IEEE Micro*, vol. 28, no. 2, Mar 2008.
[53] R. Zhang *et al.*, "Some limits of power delivery in the multicore era" in *WEED-4 workshop at ISCA-39*, Jun 2012, Portland, OR, USA.
[54] R. Zhang *et al.*, "Architecture implications of pads as a scarce resource" in *ISCA-41*, Jun 2014, Minneapolis, MN, USA.
[55] X. Zhang *et al.*, "Characterizing and evaluating voltage noise in multi-core near-threshold processors" in *ISLPED*, Sep 2013, Beijing, China.