A Survey Of Architectural Techniques for Managing Process Variation

Sparsh Mittal, Oak Ridge National Laboratory

Process variation —deviation in parameters from their nominal specifications— threatens to slow down and even pause technological scaling and mitigation of it is the way to continue the benefits of chip miniaturization. In this paper, we present a survey of architectural techniques for managing process variation (PV) in modern processors. We also classify these techniques based on several important parameters to bring out their similarities and differences. The aim of this paper is to provide insights to the researchers into the state-of-art in PV management techniques and motivate them to further improve these techniques for designing PV resilient processors of tomorrow.

Categories and Subject Descriptors: A.1 [**General Literature**]: Introductory and Survey; H.3.4 [**Systems and Software**]: Performance evaluation (efficiency and effectiveness); B.8.0 [**Hardware**]: Performance and Reliability

General Terms: Design, Performance

Additional Key Words and Phrases: Review, classification, die-to-die (D2D), within-die (WID), core-to-core (C2C), delay and leakage variation, parametric variation, resilience, GPU, CPU, 3D processor, non-volatile memory (NVM), DRAM

ACM Reference Format:
S. Mittal, "A Survey Of Architectural Techniques for Managing Process Variation", 20xx. *ACM Computing Surveys* 0, 0, Article 0 (2015), 31 pages.
DOI: http://dx.doi.org/10.1145/XX0000000.0000000

1. INTRODUCTION

As process technology scales to small feature sizes, precise control of fabrication process is becoming increasingly difficult. As a result, process variation (PV) has exacerbated greatly. For several years, it was generally believed that mitigation of PV can be done exclusively at circuit/device-level and the architectural policies could assume PV-free devices/components and thus, focus on higher-level objectives such as performance and energy optimization. In fact, until nearly 350nm technology node, PV had negligible effect on processors [Ghosh and Roy 2010], since the magnitude of variation was insignificant compared to the device size.

However, with ongoing process scaling, the effect of PV can be seen on all metrics of interest, such as yield, performance and energy. For example, with process technology scaling from 350nm to 90nm, chip yields have reduced from nearly 90% to mere 50% [Ozdemir et al. 2006] and with 45nm, a study reports the yield to be around 30% [Agarwal et al. 2005] (note that in addition to PV, other factors such as manufacturing limitations may also contribute to decreasing yields). PV leads to large variation in leakage power and maximum frequency of processor chips [Borkar et al. 2003; Gupta

Support for this work was provided by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research.
Author's address: 1 Bethel Valley Road, ORNL, TN, United States; email: mittals@ornl.gov.

DOI: http://dx.doi.org/10.1145/XX0000000.0000000

et al. 2013] and hence, if left unaddressed, PV can wipe out the performance gain obtained from an entire process technology generation [Bowman et al. 2002]. These factors necessitate special provisions for alleviating the effect of PV on commercial processors [Bowhill et al. 2015].

Driven by these factors, the perspective on strategies for addressing PV has shifted tremendously in recent years. It is now widely acknowledged that management of PV at architecture level is not only attractive, but even imperative. Architecture-level techniques can leverage the properties of the processing unit itself (e.g. CPU or GPU), specific processor components (e.g. cache or main memory) and memory technologies (e.g. NVM or DRAM) and exercise tradeoff between desired optimization targets (e.g. performance, fairness, energy or yield) which may not be possible for circuit-level techniques. Incorporating PV-awareness into architectural techniques is also important for them to account for 'ground reality' of underlying variation, since the efficacy of architectural techniques can be vastly different at different magnitudes of PV [Chen et al. 2014b; Jing et al. 2013; Liang et al. 2007; Schechter et al. 2010; Wang et al. 2012; Yoon et al. 2011]. Thus, PV-aware architecture-level techniques can complement circuit-level techniques, and reduce the requirement of stringent manufacturing/testing procedures, leading to improved revenue. In recent years, several techniques have been proposed which address this crucial need.

In this paper, we present a survey of techniques for managing PV in computing systems. Figure 1 shows the overall organization of this paper. We first present a background on PV in Section 2 and discuss the impact of PV and its types. In Table I, we classify the research works based on the the PV-affected parameters (e.g. latency, power and write endurance) and the type of PV studied by these works. To motivate this paper, we also discuss the magnitude of PV in real processors and the importance of managing PV in computing systems. In Section 3, we discuss some key ideas which are commonly used in PV-management techniques. In Section 4, we review the techniques for integrating PV-awareness in different system management schemes (Table II). From the point of view of application developers, we also classify the works based on their optimization objective and the search/optimization heuristic used by them (Table III).

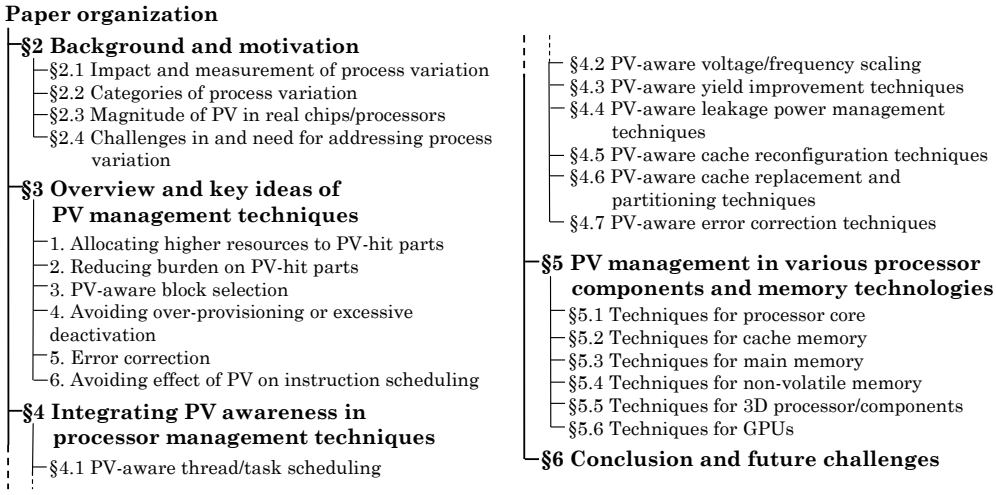

Fig. 1. Organization of the paper in different sections

Further, in Section 5, we discuss the techniques for managing PV in various processing units (e.g. CPU or GPU), processor components (e.g. core, cache), memory technologies (e.g. non-volatile memory) and fabrication techniques (e.g. 3D stacking) and present this classification in Table IV. We also mention the evaluation platform (e.g. simulator or real processor) used in a research work. Finally, Section 6 discusses the future challenges and also presents the conclusion.

PV affects a large range of devices and hence, for sake of keeping a balance between coverage and details, in this paper, we do not discuss PV in some emerging devices (e.g. FinFET, domain wall memory) and the effect of near-threshold voltage operating region on PV[1]. Also, we mainly focus on architecture and system-level techniques and not on circuit-level techniques, although it is noteworthy that some of the works discussed here span across these boundaries. It is hoped that this paper will be useful for researchers, application developers, system designers and others.

2. BACKGROUND AND MOTIVATION

Parametric variations can be caused by either dynamic factors (e.g. temperature and voltage variations arising due to application's runtime characteristics) or static factors (e.g. process variation). Of these, in this paper, we focus on process variation (PV). We now provide a brief background on PV and refer the reader to prior work for further details [Borkar et al. 2003; Bowman et al. 2002; Ghosh and Roy 2010; Ozdemir et al. 2006; Romanescu et al. 2007].

2.1. Impact and measurement of process variation

Due to PV, several device parameters may vary, for example, gate width, device threshold voltage, channel length and oxide thickness. This variability information can be obtained/estimated in several ways, such as manufacturer's datasheets, statistical models [Lu and Agrawal 2007; Sarangi et al. 2008a; Zhang et al. 2009] and temperature/performance measurements using sensors [Chan et al. 2014; Kursun and Cher 2008; Lai et al. 2014; Mahfuzul et al. 2012; Mercati et al. 2014; Sasan et al. 2012; Teodorescu and Torrellas 2008; Yan et al. 2010; Zhang et al. 2009].

For example, Chan et al. [2014] develop design-dependent ring oscillators for monitoring circuit performance and sensitivities of chips to PV. Since critical path delay sensitivities tend to occur in clusters, by synthesizing one monitor to match the delay sensitivities of each cluster, their technique captures the design-specific delay sensitivities of the chip. Mahfuzul et al. [2012] propose process-sensitive ring oscillators to estimate D2D variation in gate length and PMOS/NMOS threshold voltages based on on-chip measurement values. Lai et al. [2014] propose an in-situ monitoring approach which works by inserting timing slack monitors at selected nets (including nets along critical paths) and using these, the delay failures due to PV can be measured.

2.2. Categories of process variation

PV manifests across wafer-to-wafer (W2W), die-to-die (D2D) and within-die (WID) (W2W variation is generally lumped into D2D variation [Fu et al. 2009]). In multicore processors, WID variation may manifest itself as core-to-core (C2C) variation [Humenay et al. 2007; Papa and Mutyam 2008]. WID variation is classified as random and systematic fluctuation [Bowman et al. 2002], where the random component of PV

[1]We use the following acronyms frequently in this paper: chip multiprocessor (CMP), general purpose GPU (GPGPU), functional unit (FU), error correcting code (ECC), instruction-per-cycle (IPC), memory based computing (MBC), non-volatile memory (NVM), single-level cell (SLC), multi-level cell (MLC), phase change memory (PCM), spin transfer torque RAM (STT-RAM), embedded DRAM (eDRAM), negative bias temperature instability (NBTI), built-in self-test (BIST).

varies arbitrarily and hence, shows no correlation across devices. The systematic fluctuation exhibits spatial correlations and thus, leads to similar properties in devices that are laid out close by. Table I classifies the works based on the type of PV modeled by them and from the table, it is clear that mitigation of WID variation has received significant amount of attention from research community. Since PV manifests as variation in different architectural parameters such as latency, power or vulnerability (e.g. retention period in DRAM/eDRAM, write endurance in NVM or failure rate in SRAM), Table I also classifies the works based on whether they study/address the variation occurring primarily in latency, vulnerability or power.

2.3. Magnitude of PV in real chips/processors

To get an estimate of magnitude of process variation encountered in real-world chips, we now briefly review the measurement studies conducted on real chips. Note that, the variations reported in these studies may arise due to a combination of PV and other factors, such as voltage and temperature variation.

Dighe et al. [2011] observe that due to WID variations, the maximum clock frequency of different cores in an 80-core Intel processor vary between 5.7 GHz and 7.3 GHz. Chandrasekar et al. [2014] measure critical DRAM timings for 1Gb Micron DDR3-800 device and note that due to PV, these values can be up to 66% lower than that mentioned in the datasheet. Gupta et al. [2013] found the sleep power variation in ten instances of ARM Cortex M3 processor to be as high as 9×. They also observe up to 22% variation in power consumption of six identical Core i5-540M processors for a benchmark.

Balaji et al. [2012] evaluate power variation in six instances of Intel's 32nm Core i5-540M mobile processor using several applications. They note that depending on application characteristics and processor configuration options such as turbo boost and hyper-threading, the variability may range from 7 to 17 percent. Fraternali et al. [2014] observe up to 15% difference in energy between nodes of the Eurora supercomputer. Zhang et al. [2009] note up to 20% variation in leakage power between two cores of Intel Core 2 Duo E6420 processor. Kursun and Cher [2008] observe up to 7 degree temperature difference between two cores of a PV-affected test chip.

From these studies, it is clear that computing systems ranging from mobile processors to supercomputers show a high magnitude of process variation.

2.4. Challenges in and need for addressing process variation

As we show below, presence of PV brings several challenges in processor management and hence, addressing it is of vital importance.

2.4.1. Difficulties in accurate determination. PV management techniques rely on estimation of PV present in the chip. However, this is challenging since it may require costly, specialized and intrusive testing procedures [Chan et al. 2014; Lai et al. 2014]. Further, processor designers may not always divulge device-level information about PV and due to the random nature of PV, software-based techniques may require recompilation of a program for every hardware chip to account for its specific variation map.

2.4.2. PV exacerbated by ongoing scaling. In older technology generations, PV led to D2D variations, which could be more easily addressed by conventional techniques such as clock binning. However, ongoing process scaling has led to increased WID variations [Bowman et al. 2002; Chun et al. 2008], which demand more complicated management techniques.

2.4.3. Effect on processor design and performance . PV affects crucial architectural design decisions, such as optimal floorplan [Herbert and Marculescu 2009a; Humenay

Table I. A classification of research works based on type of PV modeled and PV-affected parameter

Classification	References
Type of PV addressed/modeled	
D2D	[Agarwal et al. 2005; Ansari et al. 2009; Bhunia et al. 2007; Bowman et al. 2002; Chandra et al. 2012; Dong et al. 2011; Ferri et al. 2008; Garg and Marculescu 2013; Juan et al. 2011; Kapadia and Pasricha 2014; Kozhikkottu et al. 2014b; Liang et al. 2007; Mahfuzul et al. 2012; Mahmood and Kim 2010; Meng and Joseph 2006; Ozdemir et al. 2010; Palermo et al. 2012; Pan et al. 2009; Paul et al. 2011; Tschanz et al. 2002; Yun et al. 2014; Zhang et al. 2009; Zhang and Li 2009; Zhao et al. 2009]
WID	[Agarwal et al. 2005; Agrawal et al. 2014; Aguilera et al. 2014; Ansari et al. 2009; Bennaser et al. 2008; Bhunia et al. 2007; Bowman et al. 2002; Chandra et al. 2012; Chen et al. 2014b; Chun et al. 2008; Das et al. 2007; Dighe et al. 2011; Dong et al. 2011; Ferri et al. 2008; Fu et al. 2008, 2009; Garg and Marculescu 2013; Goudarzi and Ishihara 2010; Goudarzi et al. 2008; Gupta et al. 2010; Herbert and Marculescu 2009a,b; Hong and Kim 2013; Hussain and Mutyam 2008; Jiang et al. 2011; Juan et al. 2011; K and Mutyam 2008; Kadayif et al. 2013; Kapadia and Pasricha 2014; Koh et al. 2009; Kozhikkottu et al. 2014a,b; Lee et al. 2011; Lee and Kim 2009; Liang et al. 2007; Mahmood and Kim 2010; Meng and Joseph 2006; Mittal et al. 2014; Mujadiya 2009; Ozdemir et al. 2010, 2006; Palermo et al. 2012; Pan et al. 2009; Paterna et al. 2012; Paul et al. 2011; Raghunathan et al. 2013; Rangan et al. 2011; Sarangi et al. 2008b,a; Teodorescu and Torrellas 2008; Tiwari et al. 2007; Tschanz et al. 2002; Yoon et al. 2011; Yun et al. 2014; Zhang et al. 2009; Zhang and Li 2009; Zhao et al. 2009]
C2C	[Das et al. 2007; Herbert and Marculescu 2009b; Humenay et al. 2007; Kursun and Cher 2008; Lee et al. 2011; Lee and Kim 2009; Papa and Mutyam 2008; Raghunathan et al. 2013; Teodorescu and Torrellas 2008]
W2W	[Juan et al. 2011; Kong and Chung 2012; Ozdemir et al. 2010]
Study of PV manifesting primarily as variation in	
Latency (or frequency)	[Aguilera et al. 2014; Bathen et al. 2012; Bennaser et al. 2008; Bowman et al. 2002; Chan et al. 2014; Chandrasekar et al. 2014; Chen et al. 2014a; Das et al. 2008; Dighe et al. 2011; Ferri et al. 2008; Goudarzi et al. 2008; Hong and Kim 2013; Hong et al. 2009; Hussain and Mutyam 2008; Jain et al. 2011; K and Mutyam 2008; Kadayif et al. 2013; Kozhikkottu et al. 2014b; Lee et al. 2011; Lee and Kim 2009; Mercati et al. 2014; Mittal et al. 2011; Momtazpour et al. 2011; Mujadiya 2009; Mutyam and Narayanan 2007; Ozdemir et al. 2006; Pan et al. 2009; Rangan et al. 2011; Reda et al. 2009; Rehman et al. 2014; Romanescu et al. 2008; Sarangi et al. 2008b; Teodorescu et al. 2007; Teodorescu and Torrellas 2008; Tschanz et al. 2002; Yan et al. 2010; Zhao et al. 2009; Zhou et al. 2013]
Power	[Balaji et al. 2012; Bathen et al. 2012; Dighe et al. 2011; Donald and Martonosi 2006; Ferri et al. 2008; Fraternali et al. 2014; Gupta et al. 2013; Herbert et al. 2012; Herbert and Marculescu 2009b; Juan et al. 2011; Kannan et al. 2008; Lee and Kim 2009; Meng and Joseph 2006; Momtazpour et al. 2011; Ozdemir et al. 2006; Reda et al. 2009; Teodorescu et al. 2007; Teodorescu and Torrellas 2008; Tschanz et al. 2002; Wanner et al. 2015; Zhang et al. 2009]
Vulnerability (failure, limited endurance or retention period)	[Agarwal et al. 2005; Agrawal et al. 2014; Ansari et al. 2009; Chen et al. 2014b; Cintra and Linkewitsch 2013; Dong et al. 2011; Jiang et al. 2011; Jing et al. 2013; Kong and Chung 2012; Liang et al. 2007; Lorente et al. 2013; Mahmood and Kim 2010; Mittal et al. 2014; Paul et al. 2011; Schechter et al. 2010; Tiwari and Torrellas 2008; Wang et al. 2012, 2014; Wilkerson et al. 2010; Yoon et al. 2011; Yun et al. 2014; Zhang and Li 2009; Zhao et al. 2014]

et al. 2007], optimal device technology (e.g. SRAM versus eDRAM for designing memory [Jing et al. 2013; Liang et al. 2007]), distribution of 3D cache over different layers [Kong and Chung 2012], among others [Henkel et al. 2013]. A PV-unaware approach can lead to reduced throughput, thermal emergencies (e.g. when computation-intensive tasks are scheduled on cores with high leakage power) and missed real-time deadlines [Mittal 2015]. This forces the designers to employ guard-banding, which trades-off performance for functional correctness and reliability. However, these margins are getting wider with ongoing scaling. PV-aware techniques can reduce the mar-

gin required, and improve yield by avoiding the rejection of otherwise acceptable designs [Gupta et al. 2013].

2.4.4. Effect of memory and stacking technologies. Emerging memory technologies are seen as a solution to power issues [Vetter and Mittal 2015], however, they do not provide solution to the PV issue. For example, due to PV, the write latency of NVM can vary between 19 to 35 cycles [Zhou et al. 2013]. PV can increase the programming power of NVMs by 96% and reduce the endurance by 50X [Zhang and Li 2009]. Given the low write endurance of NVMs and non-uniform write pattern of typical applications, wear-leveling techniques are required for achieving reasonable NVM device lifetimes [Mittal et al. 2015]. However, a PV-unaware wear-leveling technique may blindly remap the data to low-endurance NVM cells. Also, in absence of a fault-tolerance technique, a device may have to be discarded on the first failure of its cell that is most affected by PV.

In DRAM and eDRAM, PV affects the retention time and the cell with shortest retention time determines the refresh period of the entire device, which reduces device availability and increases energy consumption [Mittal and Vetter 2015a]. In 3D die-stacked processors, bonding of dies from multiple wafers can lead to large W2W variation and this may aggravate further with increasing number of layers [Kong and Chung 2012]. This necessitates use of die-stacking schemes that minimize the effect of PV in 3D processors [Ferri et al. 2008; Garg and Marculescu 2013; Juan et al. 2011; Kong and Chung 2012; Ozdemir et al. 2010].

3. OVERVIEW AND KEY IDEAS OF PV MANAGEMENT TECHNIQUES

In this section, we discuss some essential ideas which are used in different techniques discussed in the paper. Note that these ideas are not mutually exclusive.

(1) **Allocating higher resources to PV-hit parts:** The heterogeneity in parameters due to PV can be mitigated by allocating higher resources to PV-affected parts, so that PV-induced loss is compensated, e.g. using cache partitioning to give higher cache quota [Kozhikkottu et al. 2014a], pipeline adaptation for giving extra slack [Tiwari et al. 2007], adaptive body biasing (ABB) to improve frequency or reduce leakage [Teodorescu et al. 2007; Tschanz et al. 2002], performing extra refresh operations [Wang et al. 2014] and using voltage/frequency scaling (refer Table II).
(2) **Reducing burden on PV-hit parts:** Conversely, the heterogeneity can be mitigated by directing smaller "load" to PV-affected parts so that their further degradation is avoided. Several approaches have been proposed for this.
   — *Reducing writes or stress:* Writes can be reduced using wear-leveling techniques [Cintra and Linkewitsch 2013; Dong et al. 2011; Mittal et al. 2014; Yun et al. 2014; Zhao et al. 2014] or data compression [Zhang and Li 2009]. Also, programming current [Jiang et al. 2011; Zhang and Li 2009] or NBTI stress [Abella et al. 2007; Fu et al. 2008; Tiwari and Torrellas 2008] can be reduced.
   — *Deprioritizing PV-affected parts:* To remove the effect of PV on hot data and/or critical path (and ultimately on performance), use of PV-hit parts can be avoided, e.g. by avoiding low-retention eDRAM cache lines [Liang et al. 2007] and slow segments of FUs [Fu et al. 2008].
   Similarly, by skewing the replacement policy [Jain et al. 2011; Liang et al. 2007], using explicit data movement [Liang et al. 2007; Zhao et al. 2009; Zhou et al. 2013], register renaming [Romanescu et al. 2008], functional unit assignment [Romanescu et al. 2008], resource binding [Mittal et al. 2011], invalidating (but not turning-off) slow cache lines [Goudarzi et al. 2008], etc., PV-affected parts can be deprioritized and PV-robust parts can be preferentially used.

  - —*Substituting PV-hit parts:* Accesses to the disabled blocks can be remapped to non-faulty/spare blocks [Agarwal et al. 2005; Ansari et al. 2009; Cintra and Linkewitsch 2013; Goudarzi and Ishihara 2010; Mahmood and Kim 2010; Schechter et al. 2010]. Presence of narrow data values can facilitate substituting PV-hit parts [Fu et al. 2008; Kong and Chung 2012; Paul and Bhunia 2011] (refer Section 5.1). In case of faulty FUs, their functionality can instead be achieved by memory based computing [Hajimiri et al. 2013; Paul and Bhunia 2011] (refer Section 5.1).
  - —*Deactivating PV-hit parts:* PV-affected parts can be disabled for improving yield, reliability or fault-tolerance [Agarwal et al. 2005; Chun et al. 2008; Gottscho et al. 2014; Goudarzi and Ishihara 2010; Gupta et al. 2010; Ozdemir et al. 2006]. FUs with high leakage or latency can be turned-off [Kannan et al. 2008; Mujadiya 2009] for improving efficiency. Also, to allow fast cores to run at high frequency, slow cores can be disabled [Rangan et al. 2011], especially when application parallelism is low [Lee et al. 2011].
  - —*Taking backup of data in PV-hit parts:* Contents of PV-hit parts can be copied in PV-robust spare memory, which obviates the need of accessing PV-hit parts [Das et al. 2008; Goudarzi et al. 2008; Hong and Kim 2013; Kadayif et al. 2013; Romanescu et al. 2008].

(3) **PV-aware block selection:** Techniques which select blocks for some optimization can do so based on PV-awareness, e.g. cache reconfiguration techniques can preferentially turn-off high leakage blocks [Hussain and Mutyam 2008; Meng and Joseph 2006] and techniques for meeting power budget can preferentially turn-off high leakage cores [Donald and Martonosi 2006; Lee and Kim 2009; Raghunathan et al. 2013].

(4) **Avoiding over-provisioning or excessive deactivation:** PV-affected parts can be disabled at fine-granularity to avoid disabling a whole block/page for a single failure [Koh et al. 2009; Kong and Chung 2012; Mahmood and Kim 2010; Wang et al. 2012; Yoon et al. 2011]. Similarly, PV-hit parts can be refreshed or restored at fine-granularity to avoid incurring this cost at coarse granularity [Pan et al. 2009]. Also, to improve efficiency, operational margins can be tightened to provision 'bare minimum' margin [Chandrasekar et al. 2014, 2013]. As a generalization, varying levels of error-correction [Paul et al. 2011] or refresh [Agrawal et al. 2014; Jing et al. 2013] can be provided. These techniques allow continuing the execution despite failures, instead of discarding a cache or main memory on the first failure (due to the most vulnerable cell).

(5) **Error correction**: In some works, first an aggressive energy saving approach is used and then, the errors in PV-hit parts due to this approach is corrected using ECC [Jiang et al. 2011; Paul et al. 2011; Wilkerson et al. 2010].

(6) **Avoiding effect of PV on instruction scheduling:** Some techniques seek to avoid the effect of PV-induced delay variation on processor pipeline [Bennaser et al. 2008; Chen et al. 2014a; Kadayif et al. 2013; Mutyam and Narayanan 2007]. Compiler may be used to facilitate this [Kadayif et al. 2013].

We discuss the use of these and other key ideas in PV-management techniques in Sections 4 and 5. In these sections, we classify the works from several perspectives to bring out their similarities and differences. We then discuss many of these works by roughly organizing them into several groups. Although many of these techniques fall into multiple groups, we discuss them in a single group only.

4. INTEGRATING PV AWARENESS IN PROCESSOR MANAGEMENT TECHNIQUES

Presence of PV has significant impact on the efficacy of several processor management techniques, for example, task scheduling, power management schemes, cache replacement policy, NVM lifetime improvement schemes, DRAM refresh management schemes among others. Hence, computer architects and system designers must account for PV while designing these techniques. In Table II, we classify the research works based on PV-aware architectural and system-level management techniques proposed by them. Further, in Table III, we classify the works based on their study/optimization objective and the search heuristics/strategies used by them.

Table II. A classification of works based on PV-aware processor management techniques

Classification	References
Thread/task scheduling	[Aguilera et al. 2014; Dighe et al. 2011; Hong et al. 2009; Kursun and Cher 2008; Mercati et al. 2014; Momtazpour et al. 2011; Paterna et al. 2012; Raghunathan et al. 2013; Rahimi et al. 2015; Rangan et al. 2011; Teodorescu and Torrellas 2008; Yan et al. 2010; Zhang et al. 2009]
(Dynamic) voltage/ frequency scaling (DVFS)	[Bathen et al. 2012; Chandra et al. 2012; Dighe et al. 2011; Fraternali et al. 2014; Gottscho et al. 2014; Herbert et al. 2012; Herbert and Marculescu 2009a,b; Humenay et al. 2007; Kozhikkottu et al. 2014b; Lee et al. 2011; Lee and Kim 2009; Mercati et al. 2014; Papa and Mutyam 2008; Park et al. 2012; Raghunathan et al. 2013; Sarangi et al. 2008b; Sasan et al. 2012; Teodorescu and Torrellas 2008; Tiwari et al. 2007]
Replacement policy	[Jain et al. 2011; Liang et al. 2007; Zhao et al. 2014; Zhou et al. 2013]
Cache partitioning	[Hajimiri et al. 2013; Kozhikkottu et al. 2014a]
Cache reconfiguration	[Hajimiri et al. 2013; Hussain and Mutyam 2008; Meng and Joseph 2006; Ozdemir et al. 2006]
Cache block remapping	[Hussain and Mutyam 2008; Jain et al. 2011; Kadayif et al. 2013; Mutyam and Narayanan 2007; Zhou et al. 2013]
Error correction schemes	[Cintra and Linkewitsch 2013; Jiang et al. 2011; Paul et al. 2011; Schechter et al. 2010; Wilkerson et al. 2010]
NVM wear-leveling	[Cintra and Linkewitsch 2013; Dong et al. 2011; Mittal et al. 2014; Yun et al. 2014; Zhao et al. 2014]
(e)DRAM refresh policy	[Agrawal et al. 2014; Liang et al. 2007; Wang et al. 2014]

4.1. PV-aware thread/task scheduling

Yan et al. [2010] present a technique to avoid timing emergencies caused due to process, temperature and voltage variations in a multicore processor. Their technique uses delay sensors to measure delays in each core and conducts frequency analysis to estimate the contribution of each of the three sources of variation. In the frequency spectrum, the magnitude of DC (direct current) and low frequency components show the strength of temperature and process variations, whereas the components with high frequency show the strength of voltage variations. Based on this information, the application threads that cause large voltage fluctuations are migrated to faster or cooler cores, since running them on hot or slow cores aggregates their impact, which makes the cores susceptible to timing errors. In frequency domain, this is equivalent to interchanging the high frequency components which removes the variation.

With ongoing process scaling, power-wall allows only a small and decreasing fraction of transistors on a chip to be simultaneously operational at any time. Raghunathan et al. [2013] note that this phenomenon allows 'which' cores on the chip to turn on, and hence, the selection of cores can be done based on the knowledge of PV. Since PV

Table III. A classification of works based on optimization objective and search heuristic used

Classification	References
Study/optimization objective	
Performance improvement	[Aguilera et al. 2014; Bathen et al. 2012; Bennaser et al. 2008; Chandrasekar et al. 2014; Chun et al. 2008; Dighe et al. 2011; Ferri et al. 2008; Gupta et al. 2010; Hajimiri et al. 2013; Herbert and Marculescu 2009a,b; Hong and Kim 2013; Hong et al. 2009; Hussain and Mutyam 2008; Jain et al. 2011; Kadayif et al. 2013; Koh et al. 2009; Kozhikkottu et al. 2014a; Lee et al. 2011; Liang et al. 2007; Mercati et al. 2014; Mittal et al. 2011; Mutyam and Narayanan 2007; Ozdemir et al. 2010; Palermo et al. 2012; Park et al. 2012; Raghunathan et al. 2013; Rahimi et al. 2015; Rangan et al. 2011; Romanescu et al. 2008; Sarangi et al. 2008b; Teodorescu et al. 2007; Teodorescu and Torrellas 2008; Tiwari et al. 2007; Tiwari and Torrellas 2008; Tsai et al. 2005; Yan et al. 2010; Zhao et al. 2009; Zhou et al. 2013]
Dynamic energy	[Ansari et al. 2014; Chandrasekar et al. 2014; Das et al. 2007; Dighe et al. 2011; Donald and Martonosi 2006; Gupta et al. 2010; Hajimiri et al. 2013; Herbert et al. 2012; Herbert and Marculescu 2009a,b; Jiang et al. 2011; Jing et al. 2013; Kozhikkottu et al. 2014b; Liang et al. 2007; Lorente et al. 2013; Momtazpour et al. 2011; Ozdemir et al. 2006; Paterna et al. 2012; Paul et al. 2011; Sasan et al. 2012; Teodorescu and Torrellas 2008; Tiwari et al. 2007; Zhang and Li 2009; Zhou et al. 2013]
Leakage energy	[Dighe et al. 2011; Donald and Martonosi 2006; Gottscho et al. 2014; Goudarzi and Ishihara 2010; Goudarzi et al. 2008; Gupta et al. 2010; Hajimiri et al. 2013; Herbert et al. 2012; Herbert and Marculescu 2009a,b; Hussain and Mutyam 2008; Kannan et al. 2008; Kong and Chung 2012; Kursun and Cher 2008; Lorente et al. 2013; Lu and Agrawal 2007; Meng and Joseph 2006; Momtazpour et al. 2011; Mujadiya 2009; Ozdemir et al. 2006; Paterna et al. 2012; Paul et al. 2011; Sasan et al. 2012; Shafique et al. 2014; Teodorescu et al. 2007; Teodorescu and Torrellas 2008; Tsai et al. 2005]
Yield improvement	[Agarwal et al. 2005; Ansari et al. 2009; Chandra et al. 2012; Das et al. 2008; Ferri et al. 2008; Garg and Marculescu 2013; Gottscho et al. 2014; Juan et al. 2011; Kapadia and Pasricha 2014; Koh et al. 2009; Kong and Chung 2012; Mahmood and Kim 2010; Momtazpour et al. 2011; Ozdemir et al. 2010, 2006; Pan et al. 2009; Paul and Bhunia 2011; Reda et al. 2009; Tsai et al. 2005; Tschanz et al. 2002; Wang et al. 2014]
Reliability	[Abella et al. 2007; Fu et al. 2008, 2009; Gottscho et al. 2014; Henkel et al. 2013; Lorente et al. 2013; Mahmood and Kim 2010; Paul and Bhunia 2011; Paul et al. 2011; Rehman et al. 2014; Schechter et al. 2010; Tiwari and Torrellas 2008; Yoon et al. 2011]
Thermal management	[Herbert and Marculescu 2009a,b; Juan et al. 2011; Kursun and Cher 2008; Mujadiya 2009; Park et al. 2012]
Fairness	[Rangan et al. 2011; Yan et al. 2010]
NVM lifetime improvement	[Chen et al. 2014b; Cintra and Linkewitsch 2013; Dong et al. 2011; Jiang et al. 2011; Mittal et al. 2014; Schechter et al. 2010; Wang et al. 2012; Yoon et al. 2011; Yun et al. 2014; Zhang and Li 2009; Zhao et al. 2014]
Search/optimization heuristics	
Linear programming	[Ferri et al. 2008; Lu and Agrawal 2007; Mittal et al. 2011; Paterna et al. 2012; Teodorescu and Torrellas 2008; Zhang et al. 2009]
Simulated annealing	[Momtazpour et al. 2011; Teodorescu and Torrellas 2008]
Others	Genetic algorithm [Hajimiri et al. 2013], Dynamic programming [Kozhikkottu et al. 2014a], Fuzzy controller [Sarangi et al. 2008b], Graph coloring [Ansari et al. 2009], Response surface modeling [Palermo et al. 2012]

manifests as C2C variation in frequency and power, for any application, their technique selects the optimum subset of cores such that performance is maximized in a given power budget. Their technique also decides the frequency of each core and on which cores the sequential and parallel threads should be mapped to. They also show that as the percentage of turned-off fraction of chip increases, the performance benefit obtained from the ability to select cores from a large pool increases further.

Teodorescu and Torrellas [2008] present PV-aware thread scheduling algorithms for improving throughout or saving power. The first algorithm aims to save power and works by mapping threads having the largest dynamic power to the cores which dissipate the smallest amount of static power. The second algorithm aims to improve throughout and works by mapping threads with the highest IPC to cores with the highest frequency. They also propose three PV-aware DVFS methods which complement these algorithms and aim to maximize throughput at a fixed power budget (P_{budget}). These DVFS methods, which are used with the second technique above are, (a) reducing voltage/frequency of cores in round-robin manner to meet P_{budget}, (b) using linear programming and (c) simulated annealing to find the best frequency and voltage values for every core to meet P_{budget}. They show that their techniques improve CMP throughput and reduce energy-delay^2-product (ED^2P).

Rangan et al. [2011] note that although, in a PV-affected multicore processor, per-core clocking allows each core to optimize performance by running at its maximum frequency, it precludes the impression of uniform performance (shown in terms of *sales* frequency) for all cores. They propose using the apparent mean frequency (AMF) over all the cores as the single chip sales frequency. They evaluate different PV-aware scheduling policies that guarantee application performance close to that of mean frequency. The round-robin scheduling works by running processes on each core in circular order and the fairness-driven scheduling works by first predicting a process' performance at AMF and then scheduling to meet its AMF performance level. Throughput-driven scheduling works by scheduling compute-bound program phases on high frequency cores. They propose throughput-driven fairness scheduling which works by assigning configurable weights to throughput- and fairness-scheduling policies to achieve the best of both. This policy achieves performance equal or close to that expected at AMF which gives the impression of uniform performance to the end-user.

Bathen et al. [2012] present a memory virtualization approach for improving performance and/or energy efficiency by exploiting both on-chip and off-chip memory variability. Their technique allows the users to logically divide the virtual address space of the application into different regions. For each region, different mapping policies can be designed to meet different targets such as performance or energy. Based on this, their dynamic memory management module maps the program data to the most suitable memory resource using the knowledge of PV map, for example, the frequently-used data are mapped to the low-power memory. They show that their technique reduces dynamic power consumption and execution time.

Hong et al. [2009] present a technique to address PV in multicore processors by using an intelligent thread-mapping approach. They target data intensive applications where a series of loops iterate many times during execution. Their technique operates in two phases. In the first phase, the variation in execution latencies of different threads is estimated by executing one iteration of the loop nest being optimized. For this, the total load of each thread is expressed in terms of the cycles taken by the processor and the cycles spent in accessing the L1 cache. Then the threads are sorted based on their load. In the second phase, the thread with the largest load is mapped to the processor-cache pair with the fastest frequency and lowest cache latency combination. They show that by virtue of allowing every core to operate at its peak frequency,

their technique provides large performance improvement compared to a PV-unaware thread mapping scheme.

Kursun and Cher [2008] use thermal sensors to generate thermal-profile of the processor chip under PV. This information is used for power/thermal management such that hot threads are mapped on cold cores (and vice versa) and computation is moved from hotter cores to cooler cores.

In manycore processors of tomorrow, the thread-to-core combinations will increase exponentially. In face of this, ensuring effectiveness and keeping low implementation overhead will be a key challenge for future PV-aware thread scheduling techniques.

4.2. PV-aware voltage/frequency scaling

Herbert and Marculescu [2009a] note that for a CMP with large number of cores, thermal variations and WID variations present themselves as tile-to-tile variations[2]. For CMPs with fine-grain frequency islands (FIs), they propose adjusting the per-core clock frequencies to address tile-to-tile variations. Compared to a globally-clocked design, an FI-based design provides the flexibility to independently-clock each FI and thus, the frequency of each FI can be adjusted depending on the variation, without the need of adjusting the frequency globally. They also show that integration of thermally-aware frequency scaling into their approach is necessary to offset the performance loss arising due to partitioning of fully-synchronous design into different frequency islands.

Herbert et al. [2012] present PV-aware DVFS algorithms for saving energy with minimal performance loss. They note that in a multicore processor with per-core voltage/frequency islands (VFIs), the variation in leakage power of different VFIs can be large. Based on this, they propose using lower voltage/frequency levels for leaky VFIs and higher voltage/frequency levels for less leaky VFIs. This shifts the work to less leaky VFIs, with saves power and also facilitates more uniform power profile across the cores.

Das et al. [2007] propose a PV-aware power saving technique which uses one or more voltage islands for cores of a multicore processor. They note that due to C2C variations in a multicore processor, different cores achieve optimal power consumption level at different supply voltages. They study voltage islands of multiple granularities, such as an island for a core or an island for a group of cores. By studying the power saving for different island configurations, they show that while per-core voltage setting provides the largest power saving, use of even a few voltage islands can provide reasonably large power saving.

Dynamic voltage scaling approaches for cache energy saving [Mittal 2014] work by scaling supply voltages of cold blocks that have not been used in recent past. Sasan et al. [2012] propose to add PV-awareness to these techniques to improve energy saving without sacrificing performance or reliability. Their technique applies voltage scaling based on both the cache access pattern and the vulnerability of the memory cells to PV. The vulnerability of memory cells is recorded using a BIST circuitry and is updated on a change in voltage, frequency or temperature. In their technique, any of the following three voltages can be used for a cache way. The lowest voltage is applied to cold ways which are found based on the cache access pattern. The remaining two voltage levels are applied depending on whether a cache way can work correctly on that voltage. Thus, weaker cells are supplied with higher voltage to avoid reliability issues. They show that their technique saves energy while maintaining correctness and performance.

[2]Different from small-scale CMPs, large-scale CMP architectures are designed with similar or near-similar basic blocks, which are termed as tiles. A CMP may include processor tiles (e.g. core and private L1 caches) and memory (e.g. L2 caches) tiles, connected using the on-chip network.

As the supply voltage has been steadily scaling (e.g. from 5V at $800nm$ to $\sim$1.1V at $32nm$ process technology), the scope of energy saving from DVFS is also reducing. Tackling this issue will be essential for porting existing PV-aware DVFS solutions to future miniaturized processor-chips.

4.3. PV-aware yield improvement techniques

In processor design, if the frequency of the longest pipeline stage is below the designated minimum processor frequency, the chip is marked as failed which reduces the yield. Pan et al. [2009] propose a technique to reduce the L1 cache access latency if the higher cache latency causes a yield loss. Their technique boosts the supply voltage of wordlines of only the failing rows. This helps in reducing the cache access time with low power and area overheads. They show that their technique leads to significant improvement in the yield level.

Ozdemir et al. [2006] present two techniques that minimize yield losses due to excessive leakage and excessive latency. Their first technique uses way-based cache reconfiguration to deactivate a way that violates maximum allowed latency constraint. Also, if the power consumed by the whole cache exceeds a limit, the cache way that consumes the largest leakage power is deactivated. A deactivated way is not used throughout the lifetime of a chip. Since turning off cache ways causes performance loss, their second technique uses variable latency cache architecture, such that slower ways can be kept enabled by allowing additional latency for completion of accesses to them. This technique uses special buffers in FUs to allow dependents of a load operation to stall whenever a delay is observed in a load operation. Thus, correct execution can be ensured despite the delay incurred on slow ways. They also show that by combining these two techniques, even larger improvement in yield can be obtained.

Das et al. [2008] present a cache architecture to improve the yield. They use a small-sized fully-associative array with each L1 cache way. This array replicates the data stored in PV-affected high latency lines. Later on, these data values need not be accessed from the high-latency lines in L1 cache, which allows the cache to operate at higher frequency. Thus, by allowing more number of chips to operate at higher frequencies, their technique reduces yield loss due to delay variation.

For improving the yield of a PV-affected cache, Mahmood and Kim [2010] propose disabling fault locations at the fine-granularity of cache-word level and not cache line/way/set level. In their technique, accesses to faulty words are remapped to a word-length buffer having small number of entries. They show that their technique improves parametric yield and allows graceful performance degradation.

It has been estimated that 1% reduction in yield can lead to 12% reduction in profit [Flamm 2010]. Since cost efficiency is the primary determinant in ongoing development of a processor design technology, in coming years, novel solutions at device, architecture, system and programming level will be crucial for mitigating the impact of PV on yield.

4.4. PV-aware leakage power management techniques

Kannan et al. [2008] note that leakage energy consumption of functional units varies due to temperature and PV. Using leakage sensors, they measure the leakage energy dissipation of each FU. Using these values, the decision about the number of FUs to power-gate is taken. Also, most leaky FUs are preferentially chosen for power-gating to minimize the total energy consumption. This approach also helps in reducing the variation in the leakage of FUs.

Donald and Martonosi [2006] note that both PV and application characteristics can lead to higher leakage power in some cores in a multicore processor. They derive an analytical model to find out when to turn-off a core with high power consumption such

that the 'performance per watt' is maximized. They further extend their model to a parallel computing scenario where they use Amdahl's law to estimate how a change in active core count affects the energy efficiency of a parallel program.

The transistor-count on modern chips has exceeded 10 billion [Morgan 2014], however, not all the transistors are expected to be used simultaneously. Since the problem of unused transistors dissipating leakage power is getting worse with ongoing scaling, PV-aware leakage management will be increasingly important for meeting power budgets in future systems.

4.5. PV-aware cache reconfiguration techniques

Cache reconfiguration techniques are specific forms of leakage energy saving approaches, which leverage the properties of cache, such as turning-off unused data for saving leakage energy and later fetching those data from main memory. We now discuss some PV-aware cache reconfiguration techniques.

Hussain and Mutyam [2008] propose a technique to save leakage energy in a PV-affected cache with minimal performance loss. Their technique rearranges the blocks such that there are nearly equal number of PV-hit high latency blocks in all the sets. Further, since applications with small working set size do not require full associativity, the high latency blocks are turned-off for saving leakage energy. Since the turned-on blocks are low latency blocks, the performance loss remains minimal.

Way-based cache reconfiguration approaches [Mittal 2014] work by turning-off some cache ways based on program working set size, such that leakage energy is saved with minimal performance loss. Meng and Joseph [2006] note that since PV can lead to difference in leakage power dissipation of different ways, in a way-based reconfiguration approach, the choice of 'which' (and not just 'how many') ways are turned-off can have significant influence on the total energy saved. Their technique uses a hardware register which records relative ordering of physical ways based on decreasing leakage power and another register which records absolute leakage of corresponding physical way. Using information from these registers, on a cache reconfiguration, the decision about turning on/off ways can be made to strike a balance between energy saving and performance loss.

Due to the latency-critical nature of L1 cache, cache reconfiguration techniques are not suitable for it. Using cache reconfiguration in last level cache increases off-chip accesses and hence, while accounting for increasing PV, future reconfiguration techniques should also achieve low performance overhead to justify their use. This may necessitate integrating reconfiguration techniques with performance optimization techniques such as data compression.

4.6. PV-aware cache replacement and partitioning techniques

In a PV-affected eDRAM cache, the retention period of different cache blocks in a set may be different. In a naive "no-refresh" policy, no cache block is refreshed and hence, any block whose retention time is lower than a threshold is evicted. However, with this policy, the blocks with low retention period are likely to be free most of the times and hence, they may be repeatedly used by cache replacement policy, which is undesirable. To address this issue, Liang et al. [2007] propose three PV-aware replacement policies for eDRAM caches. The first policy always avoids blocks with zero retention period. The second policy works on the observation that since a large fraction of cache accesses happen in a short duration after a data-item is loaded, loading the new data-item into largest retention period block can greatly reduce its invalidation due to expired retention period. Based on it, this policy logically arranges the blocks in every set in decreasing retention period order. The new data-item is stored in the block with largest retention period and the data-item in that block is migrated to the block with the next

largest retention period. In the third policy, the block with the longest retention period is used for storing the most-recently referenced data-item. Thus, on each write or read, the blocks are suitably shuffled to maintain the order.

Jain et al. [2011] present a PV-aware version of LRU (least recently used) replacement policy which aims to reduce the number of accesses to blocks having high latency due to PV. On cache hit to a high latency block, their technique migrates the data to a low latency block in the same set to try to use low latency blocks for storing repeatedly-used data items. They show that their technique eliminates nearly all the performance loss due to PV.

Kozhikkottu et al. [2014a] propose a technique which partitions shared system resources to accelerate the threads which are most affected by PV. Their technique addresses WID variations by unequally allocating shared resources. They target multi-threaded applications where the slowest thread determines the performance of the application due to the presence of synchronization barriers. In their technique, in the first phase, the performance of each thread is characterized by varying its cache quota and deducting the time consumed in stalling at synchronization barriers. In the second phase, the cache is partitioned at way-granularity with a view to maximize the performance of the slowest thread and this also improves the overall performance.

Future work in this area should further enhance PV-aware cache replacement and partitioning techniques to account for system-level goals such as fairness, quality of service, energy efficiency, etc., in addition to performance.

4.7. PV-aware error correction techniques

Schechter et al. [2010] note that while ECC is used to correct soft errors, due to the low endurance of NVMs, these codes reach their endurance limit much sooner than the cells they seek to correct. They propose error-correcting pointers (ECPs), in place of ECC, for correcting hard errors (due to limited write endurance) in NVMs. In ECC scheme, coded bits are used with each data block, while in ECP scheme, addresses of failed cells are encoded and stored and spare cells are used for replacement. Thus, ECP scheme can tolerate errors in known memory bit positions by maintaining pointers to these bit positions and using the spare cells. They show that even under severe PV, their technique improves lifetime of non-volatile memory significantly.

Wilkerson et al. [2010] propose using higher refresh period than that required by the weakest cell and then using strong ECC for tolerating failures in the weak cells. For example, a baseline eDRAM cache requires refresh operations every 30μS, while for keeping the same failure probability, using SECDED (single error correction, double error detection) code and their technique (which uses 5EC6ED code, i.e. 5 error correction, 6 error detection) allow increasing this period to 150μS and 440μS, respectively. Thus, at the cost of higher complexity of error correction and encoding/decoding overhead, stronger ECC allow increasing the refresh period for saving power while maintaining reliability. To reduce the storage overhead of ECC, they design a low-cost ECC circuit and amortize ECC cost over large data words (e.g. 5EC6ED for a 1KB line instead of a 64B line).

Paul et al. [2011] note that due to PV, vulnerability of different cache blocks to failures (e.g. read, write and hold failures) becomes different and hence, providing homogeneous ECC protection to every memory block is inefficient. They propose a technique where ECC allocated to a block is determined by its relative vulnerability towards runtime failures. Logic for encoding/decoding ECC with different error correction capability are incorporated at design time and are selected at runtime. They also propose combining their adaptive ECC approach with dynamic voltage scaling for saving energy, such that stronger ECC is used to tolerate the errors introduced due to voltage scaling.

Reliability mechanisms such as ECC and redundant execution are critical for applications demanding very high reliability, e.g. medical and space applications. In the wake of rising impact of PV, highly efficient ECC schemes will be crucial for meeting the conflicting goals of high reliability and high performance in these systems.

5. PV MANAGEMENT IN VARIOUS PROCESSOR COMPONENTS AND MEMORY TECHNOLOGIES

Different processing units (i.e. CPU or GPU), processor components (cache, memory, register file or functional unit) and memory technologies (SRAM, DRAM/eDRAM or NVM) have different properties and hence, PV management techniques in them need to address different tradeoffs and/or can benefit from their unique architectural characteristics. To highlight this, Table IV categorizes the research projects based on the processor, its component, memory technology and the fabrication technique. Since first and last level caches offer different latency/capacity/area tradeoffs [Mittal 2014; Mittal et al. 2015], the PV-management strategy used in those caches may be different and hence, Table IV also classifies the techniques based on the cache level for which it is targeted (or is implemented in).

Both real chips and simulators are indispensable evaluation platforms for PV-related studies, since the real chips provide valuable insights and data about magnitude and pattern of PV observed in the field and simulators offer flexibility to study different PV-management techniques for various processor components in a repeatable manner. Hence, Table IV also classifies the research works based on their evaluation platform.

In a processor, PV can arise in different registers, TLB entries , functional units and different cells of cache or main memory. We discuss the techniques for addressing PV in these components in Section 5.1 through Section 5.3.

5.1. Techniques for processor core

Tiwari et al. [2007] present a technique to address PV in processor pipeline where PV leads to imbalance in speed of pipeline stages. In their technique, the clock arrival times are skewed to the latching elements which effectively transfers the time slack of faster stages to the slow stages. To insert skew in the signal reaching each pipeline register, their technique utilizes tunable delay buffers in the clock network. By virtue of this, the clock period of a pipeline stage is determined by the average stage delay and not the worst case delay.

Kadayif et al. [2013] present a technique for mitigating the effect of PV on instruction fetches. To reduce the impact of TLB latency variation, their technique uses an extra register as a 'cache' for TLB. This register records the result of latest virtual-to-physical address translation. When the next instruction access demands the same translation as stored in the register, an access to TLB is avoided. Since instruction accesses exhibit high locality, the register greatly reduces the accesses to TLB and thus, the impact of TLB latency variation on performance is minimized. To address the latency variation among different cache lines, they analyze the application using a compiler and generate its control flow graph. The cache access latency value of every subsequent instruction is estimated based on latency of the cache set which will be accessed for that instruction, assuming that the latency values are available through suitable tests at compile time. This value is recorded in the unused bit positions of the instruction. When an instruction is fetched from the cache, the latency value is also read which helps in estimating when the cache can be accessed for the succeeding instruction.

Fu et al. [2008] present techniques to leverage the positive interaction of PV and NBTI and avoid their negative interaction. NBTI leads to increased PMOS threshold

Table IV. A classification of research works based on processor component and memory technology

Classification	References
Processing Unit	
GPU	[Aguilera et al. 2014; Jing et al. 2013; Lee et al. 2011; Lee and Kim 2009]
CPU	almost all others
Processor component	
First level caches	[Agarwal et al. 2005; Ansari et al. 2009; Bennaser et al. 2008; Chen et al. 2014a; Das et al. 2008; Fu et al. 2008; Gottscho et al. 2014; Goudarzi et al. 2008; Hajimiri et al. 2013; Hong and Kim 2013; Hong et al. 2009; Hussain and Mutyam 2008; Jain et al. 2011; Kadayif et al. 2013; Koh et al. 2009; Liang et al. 2007; Lorente et al. 2013; Mahmood and Kim 2010; Mutyam and Narayanan 2007; Ozdemir et al. 2010, 2006; Pan et al. 2009; Romanescu et al. 2008; Sasan et al. 2012]
Last level caches	[Agrawal et al. 2014; Ansari et al. 2009; Ferri et al. 2008; Gottscho et al. 2014; Hajimiri et al. 2013; Koh et al. 2009; Kong and Chung 2012; Meng and Joseph 2006; Mittal et al. 2014; Ozdemir et al. 2010; Paul et al. 2011; Wang et al. 2012; Wilkerson et al. 2010; Zhao et al. 2009; Zhou et al. 2013]
Main memory	[Chandrasekar et al. 2014, 2013; Chen et al. 2014b; Dong et al. 2011; Jiang et al. 2011; Schechter et al. 2010; Wang et al. 2014; Yoon et al. 2011; Yun et al. 2014; Zhang and Li 2009; Zhao et al. 2014]
Register file	[Fu et al. 2008; Jing et al. 2013; Ozdemir et al. 2010; Romanescu et al. 2008]
Other components	[Ferri et al. 2008; Fu et al. 2008, 2009; Gupta et al. 2010; Hong et al. 2009; K and Mutyam 2008; Kannan et al. 2008; Kursun and Cher 2008; Mujadiya 2009; Romanescu et al. 2008; Sarangi et al. 2008b; Tiwari et al. 2007]
Memory technology	
SRAM	[Agarwal et al. 2005; Ansari et al. 2009; Bathen et al. 2012; Bennaser et al. 2008; Bhunia et al. 2007; Chen et al. 2014a; Das et al. 2008; Gottscho et al. 2014; Goudarzi and Ishihara 2010; Goudarzi et al. 2008; Hajimiri et al. 2013; Hong and Kim 2013; Jing et al. 2013; Koh et al. 2009; Kong and Chung 2012; Liang et al. 2007; Lorente et al. 2013; Mahmood and Kim 2010; Meng and Joseph 2006; Ozdemir et al. 2010, 2006; Pan et al. 2009; Sasan et al. 2012]
eDRAM	[Agrawal et al. 2014; Jing et al. 2013; Liang et al. 2007; Lorente et al. 2013; Wilkerson et al. 2010]
DRAM	[Bathen et al. 2012; Chandrasekar et al. 2014, 2013; Wang et al. 2014; Zhao et al. 2009]
NVM	[Chen et al. 2014b; Cintra and Linkewitsch 2013; Dong et al. 2011; Jiang et al. 2011; Mittal et al. 2014; Schechter et al. 2010; Wang et al. 2012; Yoon et al. 2011; Yun et al. 2014; Zhang and Li 2009; Zhao et al. 2014; Zhou et al. 2013]
Fabrication technique	
3D processor	[Ferri et al. 2008; Garg and Marculescu 2013; Juan et al. 2011; Kong and Chung 2012; Ozdemir et al. 2010; Reda et al. 2009; Zhao et al. 2009]
Evaluation platform/approach	
Real chip/processor	[Balaji et al. 2012; Borkar et al. 2003; Bowhill et al. 2015; Chandrasekar et al. 2014, 2013; Dighe et al. 2011; Fraternali et al. 2014; Gottscho et al. 2014; Gupta et al. 2013; Kursun and Cher 2008; Tschanz et al. 2002; Zhang et al. 2009]
Simulation/ analytical	almost all others

voltage which manifests as a slow-down of transistors and possible timing violations [Abella et al. 2007; Tiwari and Torrellas 2008]. NBTI recovery is done by setting positive voltage at PMOS transistor gate [Fu et al. 2008]. In a multi-ported register file, read access time constitutes the largest part of the delay and hence, in their first technique, larger utilization is assigned to ports with smaller read access time. The reduced utilization of ports with longer access time helps in reducing NBTI degradation on them. Thus, this technique shifts more NBTI degradation to ports which are less affected by PV. Their second technique exploits narrow-width values to alleviate the impact of NBTI loss in FUs. Narrow-width values refer to those which occupy only a

fraction of storage space in a data block. They logically partition a 64-bit FU into four 16-bit segments, where 16-bit operations can be done by each segment independently, but all four segments need to be involved for data values greater than 16-bits. Since PV affects the delay of different segments differently, their second technique uses the current fastest segment for executing a narrow-width data value. They also use runtime detection schemes to measure the total effect of PV and NBTI, since it changes the delay of different segments at runtime.

Memory based computing (MBC) refers to the computing strategy where functions are evaluated by retrieving their response values from the lookup tables stored in the main memory. Since PV can induce failures in FUs of the processor, MBC can be used for improving reliability of computing systems. For such multicore processors, Hajimiri et al. [2013] propose a cache reconfiguration and partitioning technique to improve their performance and energy efficiency. They apply MBC to achieve the functionality of integer execution unit in every core. Both the core-private L1 cache and shared unified L2 cache can be partitioned. Partitioning is done statically for conventional instruction/data access and MBC-related accesses. They use genetic algorithm to find the suitable L1/L2 cache quota that should be allocated for instruction/data accesses and for each MBC operation, for example, in an 8-way cache, 4 ways may be allocated for regular instruction/data accesses and 3 and 1 ways (respectively) may be allocated for addition and multiply lookup tables for MBC. They show that their technique provides energy and performance gains without sacrificing reliability.

In VLIW (very long instruction word) architectures, the compiler controls the instruction scheduling and hence, by using the information about underlying variation, FUs can be assigned at compile time to mitigate the effects of PV. Mujadiya [2009] present two compile-time techniques for addressing PV-induced delay variation in integer FUs (IFUs) in VLIW architectures. They note that in some portions of VLIW application execution, the number of available IFUs becomes much larger than the number of operations that can be performed in a cycle. This provides the opportunity to avoid the PV-affected slow IFUs. Based on this, their first technique turns-off all the slow IFUs. The second technique turns of some of the slow IFUs whenever the IPC becomes greater than the number of normal latency IFUs. They show that their techniques achieve nearly the performance as a VLIW processor with homogeneous latencies. Further, turning off the IFUs saves leakage energy and reduces peak temperature compared to the case where worst-case latencies are used for all components.

Romanescu et al. [2008] present techniques for mitigating the effect of PV on determination of processor frequency. Their technique for processor registers identifies critical instructions and renames their destinations to fast registers. Their technique for functional units (FUs) preferentially assigns critical instructions to the fast FUs. This removes slow FUs from the critical execution path of an application. To handle different cycle latencies due to PV, the instruction scheduling logic is modified, such that depending on the speeds of FUs (which is measured at production time and is stored in processor), a multiplexer chooses between regular and slow (due to PV) scenario. Based on it, the scheduling logic waits for suitable time period before dependent instructions are scheduled.

Many existing systems use coarse-grain redundancy (e.g. triple modular) and/or deactivation of PV-affected components/cores to ensure reliability and maintain high frequency. Although conservative, such an approach leads to huge wastage and does not scale to high failure rates. Designing fine-grain redundancy mechanisms will be important to continue to scale performance of future processor cores.

5.2. Techniques for cache memory

Mutyam and Narayanan [2007] present a technique to minimize the effect of PV on cache performance. Since PV affects latency of blocks, presence of both high and normal latency blocks in a set makes it difficult to predict cache access latency. This complicates scheduling of dependent instructions and to avoid this, their technique aims to map and group together PV-affected blocks in few sets only, so that the number of sets having 'both' high and normal latency is minimized. For this, the physical locations of cache blocks are rearranged such that for an M-way cache, the M blocks of a set can be in multiple rows instead of a single row as used in conventional caches.

In PV-affected caches, assuming the worst-case delay as the delay of all the cache lines leads to an overly-conservative approach, since most cache lines show much lower latency. Bennaser et al. [2008] present a technique which finds the delay of each cache line and uses this value to reduce waiting cycles in the pipeline. They use a BIST circuitry to test the cache access latency of entire cache and then store the result into delay storage unit of every cache line. Since the latency of cache access is known accurately, use of the worst-case value is not required. This leads to better scheduling of dependent instructions and an improvement in the performance.

Zhou et al. [2013] note that in NVM (e.g. STT-RAM) caches, the read and write latencies are asymmetric and the write latency is significantly higher than the read latency [Poremba et al. 2015]. Under PV, the variation in write latency is higher than that in read latency. Hence, based on the write latency (and not read latency), their technique classifies the cache lines into two groups, viz. fast and slow. The access latency of any line in a group is equal to the slowest line in that group. Their technique migrates frequently accessed data in fast cache lines to improve performance. They also use a cache block remapping scheme [Hussain and Mutyam 2008] to make the ratio of fast and slow lines uniform across different sets.

Hong and Kim [2013] propose a technique to tolerate PV-induced access time failures in the cache. They note that modern processors can tolerate a small increase in L1 cache latency due to features such as out-of-order execution, accurate branch prediction, among others. Thus, they tradeoff a slight increase in L1 cache latency to eliminate cache access time failures. A cache access involves three operations, viz., address decode, cell access (CA), and data out. Due to PV, CA can take up to two cycles. Their technique already allows two cycles to CA operation, to give it enough timing margin, so that a timing error does not happen. To partially recover the performance loss, they use sub-banking scheme which allows multiple access requests to be served at the same time if they go to different banks. They also use a buffer to store recently accessed data which decreases accesses to the cache to further reduce the impact of PV-induced delay on performance.

Agarwal et al. [2005] present a cache architecture for detecting and replacing faulty cells in a PV-affected cache. They use a runtime BIST circuitry to identify PV-affected faulty cells. Using this information, when an access happens to a faulty block, their technique alters the column address and thus, forces the column multiplexer to choose a working (i.e. healthy) block from the same row, in place of the faulty block. This cache resizing is not visible to the core, and hence, the core can access the cache using the same address as in the baseline cache. They show that their technique improves the yield significantly.

Koh et al. [2009] propose a technique for improving performance and yield of PV-affected caches. They note that for typical fault rates, the probability of seeing faults in more than one bit of a 512 bit (64B) cache line is very low. Hence, disabling a complete cache line for a single bit failure is wasteful. Their technique pairs a faulty cache line with a different faulty line in the same set if their faulty bit locations do not overlap.

This helps in retaining a functional line instead of losing two lines. They show that their technique helps in graceful degradation of performance and yield of the cache in the presence of faults. Romanescu et al. [2008] present a technique for L1 caches which prefetches data into small-sized L1-data and L1-instruction prefetch buffers that are guaranteed to be unaffected by PV. This removes the effect of PV-affected blocks on performance.

Gottscho et al. [2014] note that in presence of PV, SRAM bits that fail at a particular supply voltage also fail at all lower voltages and using this, a fault-map for multiple voltages can be built with little additional overhead compared to a fault-map for single voltage. Based on this, their techniques scale voltage of data array while turning-off faulty blocks to save cache power. Their static technique selects the optimal voltage by using boot-time knowledge of faulty-blocks, such that at least 99% blocks are without faults. The dynamic technique trades-off capacity with energy saving at runtime. At low voltage, the miss-rate becomes high and in such case, their technique increases the voltage to enhance capacity and performance by regulating the miss-rate. When miss-rate becomes low, voltage is reduced again to exploit energy saving opportunity.

In a multicore system with shared cache, degradation in cache capacity due to PV can have unexpected impact on performance of applications running on different cores. Since previous studies have focused on performance impact of PV on single-core systems only, detailed studies on impact of PV on multicore systems are definitely required.

5.3. Techniques for main memory

Zhang and Li [2009] present techniques to address PV in PCM main memory. In their first technique, the PCM array is divided into multiple domains and the voltage level (and the corresponding programming current) is adapted for different domains. For each domain, the lowest current magnitude is selected that can successfully program all cells within that domain. In the second technique, memory pages are classified into hot and cold pages based on the frequency of page updates. Then, hot-modified pages are mapped to the regions which are positively affected by PV and cold-modified pages are mapped to regions which are negatively affected by PV. Also, to save energy, in writes to the cold-modified pages, data comparison write is performed to write only those bits which have changed. In the third technique, when an evicted cache line is written back, if the write is performed to a region positively affected by PV, a partial write (i.e. writing only the dirty words) is performed. By contrast, if write is performed to a region negatively affected by PV, data are written in compressed form if doing so leads to smaller number of bit-writes compared to the partial write scheme.

Jiang et al. [2011] note that the endurance of PCM cells is crucially affected by the RESET current, which varies due to PV. Hence, use of a single RESET current value for all cells leads to over-programming which shortens the lifetime of NVM significantly. They propose a technique which regulates RESET current at the granularity of a single cache line. They note that in every line, only a few cells are difficult-to-reset, and based on this, their technique intentionally reduces the RESET current for these cells to let them fail temporarily and then uses error-correction schemes to recover them. With this support, the RESET current for remaining cells can be greatly reduced which leads to improvement in the lifetime of non-volatile memory.

Wang et al. [2014] propose a PV-aware refresh management policy for DRAM-based main memory. Their technique manages refresh operations for PV-affected low-retention cells, without reducing refresh period for the entire memory. In their technique, DRAM records the low-retention cells and observes when some of these rows need supplementary refreshes. In such a case, DRAM behaves like a memory operation requester and sends out activate commands to the memory controller. The mem-

ory controller inserts activate commands into command queues for future scheduling. Thus, DRAM proactively ensures that low-retention rows receive extra refresh, which avoids their failure. This improves the yield, alleviates the need of redundant rows and provides higher performance and energy gains than using worst-case refresh period for all the rows.

Changes in main memory management affect application and OS policies. Hence, future work should focus on designing PV-management techniques which are transparent to application and OS, so that legacy software may be used without significant modifications.

5.4. Techniques for non-volatile memory

In context of NVM, we now discuss techniques for PV-aware wear-leveling and for continuing execution despite early failure of PV-affected blocks (refer Section 2.4.4).

Dong et al. [2011] propose a PV-aware wear-leveling technique which works by balancing wear-rate (computed as 'writes divided by endurance'), instead of only wear (i.e. writes) across different NVM domains. For this, they record number of writes to each logical domain and endurance for each real domain in two different lists. Then these lists are sorted and finally, the logical domain with the largest write traffic is mapped to the real domain having the largest endurance. This minimizes the maximum wear-rate of every domain and thus, balances the wear-rates across the NVM.

Yun et al. [2014] propose a PV-aware wear-leveling techniques that uses low-overhead Bloom filters to record both write count and endurance variation information at fine-granularity. To reduce the number of false positives in Bloom filter, they use multiple carefully-designed hash functions. Using this, when hot data are mapped to those NVM cells that have seen a large number of writes or have inferior write endurance, the data are remapped to another NVM cell that has seen lesser wear-out.

Wang et al. [2012] present a technique for tolerating write-endurance induced errors in NVM caches. When an NVM byte fails due to limited write endurance, their technique discards only the faulty byte and not the whole cache line. The locations of faulty bytes are stored in the non-faulty bytes of the cache line. As the number of failures in a set increases, the number of available ways in that set reduce, however, the cache can still function and this leads to improvement on NVM cache lifetime. Their technique ensures graceful performance degradation as more memory cells reach their endurance limit. They also show that with increasingly severe PV, their technique provides larger improvement in lifetime.

Yoon et al. [2011] present a fine-grained remapping technique to protect NVM main memory from both soft and hard errors [Mittal and Vetter 2015b]. A conventional technique disables and remaps a block when the number of wear-out failures exceed the correction capability. Their technique stores the redirection information in the non-faulty cells of worn-out memory blocks. The spare area is dynamically created within the main memory. To avoid chained remapping on failure of a remapped block, their technique stores the final remapping destination address in the original failed block. By virtue of fine-grained remapping, failure of a 64B block does not lead to remapping or disabling of an entire 4KB page. They show that relative advantage of their technique increases further with increasing magnitude of PV.

Depending on whether multiple or single data bit(s) are stored in an NVM cell, it is referred to as MLC or SLC [Mittal et al. 2015]. Zhao et al. [2014] note that while MLC provides capacity advantage, its write-endurance is 2-3 orders of magnitude smaller than that of SLC. PV-induced endurance variation and non-uniform write-distribution can further aggravate the problem of limited endurance. They propose a PV-aware wear-leveling technique, which aims to bring together the capacity benefit of MLC and endurance benefit of SLC. Their technique transitions the MLC pages seeing high

write-traffic and having low endurance (due to PV) into SLC to benefit from high speed and endurance of SLC. This avoids the complexity and overhead of redirecting writes to different pages in the memory. They show that their technique improves the memory lifetime significantly while retaining the capacity advantage of MLC by carefully controlling the number of SLC conversions.

Another promising direction for addressing NVM endurance is the design of SRAM-NVM hybrid caches [Mittal and Vetter 2015c], where write-intensive blocks can be redirected to SRAM for achieving higher device lifetime and performance. However, since SRAM is also vulnerable to PV and consumes high leakage power, architecting intelligent policies for SRAM-NVM hybrid caches will be crucial for bringing together the best of both SRAM and NVMs.

5.5. Techniques for 3D processor/components

Kong and Chung [2012] present a technique for tolerating PV-induced failures in 3D SRAM last level cache by exploiting narrow-width values. They classify 64-bit data values into four types, viz. 64-bit full-width values and 48-bit, 32-bit and 16-bit narrow-width values. They note that partitioning of cache across 3D layers can be done either across sets, ways or bits, where different sets, ways or bits of each block (respectively) are mapped to each layer. Of these, they suggest use of bit-partitioning scheme since it leaves several cache lines usable with narrow-width values even when some layers suffer from severe PV. Four cache bit subblocks (each 16-bit) which constitute a 64-bit cache word subblock are assigned to four different layers. Their technique maintains a map of faults at the granularity of 16-bit cache bit subblocks and not at coarse granularity of 64-byte cache line. Thus, even if some bit subblocks in a cache line are faulty, a data value may still fit in this line if the data value is narrow-enough. Thus, their technique provides larger lifetime than a naive technique which discards a cache line on occurrence of even a single cache cell failure.

Reda et al. [2009] develop a PV-aware framework for estimating the effect of mapping a 2D design onto a 3D stack. Using statistical analysis, they show that by splitting a 2D design into multiple dies that are manufactured independently and then stacked together without any parametric testing, the timing and leakage variability can be reduced. Further, when individual die parametric testing information is available, the dies can be stacked optimally to further reduce the impact of PV. Thus, they show that 3D stacking can be an effective approach for combating PV.

Ferri et al. [2008] propose integration strategies for improving yield of 3D processors. They consider a 3D processor where the upper wafer holds L2 cache die and the substrate wafer holds CPU die. The first integration strategy pairs the fastest CPUs with the fastest L2 caches and the second pairs the fastest CPUs with the slowest L2 caches. The third strategy finds the optimal assignment by modeling this problem as an integer linear programming problem and the fourth strategy pairs the dies randomly. They show that on using the first and the third strategy, the 3D chips in the fastest speed bins increase which improves the yield.

Ozdemir et al. [2010] note that a die-stacked chip may use layers manufactured in different wafers and hence, due to W2W variation, the properties of different layers can be significantly different. Since the processor performance is determined by the slowest component, placing all the critical components in a single layer would increase the probability of a slow layer reducing the overall performance of the processor. Since the critical path generally lies in caches and register files, they propose splitting these critical components into different layers. Using this approach, the yield loss due to 3D stacking can be nullified and parametric variations between different layers can be exploited for improving yield ratios. Further, their approach improves the average frequency (performance) of the chips.

Zhao et al. [2009] present a technique to address PV-induced variation in access times of different banks in a DRAM last level cache. Since DRAM provides large capacity, the working set of an application may fit in few banks only. Based on this, their technique migrates data from slow banks to fast banks for reducing the average access time. To reduce the contention in the fastest banks, they divide banks into tiers and migrate data from a slow tier to the fastest tier for evenly distributing the migrations. Also, to reduce migration overhead, they propose a variant of this technique where migration happens only within each column so that shorter interconnection wires provided by stacking can be leveraged to reduce energy of data movement. They show that their technique brings the performance very close to that of an ideal PV-free memory.

Juan et al. [2011] show that compared to a 2D processor, an equivalent 3D implementation sees much larger susceptibility in temperature profile due to PV-induced leakage variations. They present a method to accurately predict how the maximum steady-state temperature of a 3D processor varies with the total leakage power consumption of every tier. Using this temperature modeling scheme, they select the optimal tier-stacking order such that the tiers which consume the highest leakage power are placed closer to the heat sink to minimize overall temperature.

In 3D-stacked cache or main memory, stacking strategies (e.g. whether a rank is within one layer or is split across multiple layers) can have significant influence on performance/energy/thermal efficiency and data-migration policies. Integrating awareness of this into conventional PV-management solutions will be important for enhancing their effectiveness for 3D-stacked processors.

5.6. Techniques for GPUs

Lee et al. [2011] present two techniques for mitigating the effect of C2C frequency variations on GPGPU performance. They note that several GPGPU applications have no or few synchronizations between thread blocks. Using this, their first technique allows each core to run at its own maximum operating frequency (F_{max}) determined by PV. They also note that in some GPGPU applications, all the cores are not utilized efficiently. Since the slowest core limits the F_{max} of GPGPU, their second technique disables the slowest core limiting the F_{max} of GPGPU. Their first technique is suitable for a per-core clocking design and the second technique is suitable for a global-clocking design.

Lee and Kim [2009] show that in the presence of C2C frequency and leakage variations in a GPU, per-core clocking scheme provides higher energy efficiency than the global-clocking scheme. Further, when the demanded throughput is lower than the maximum possible throughput such that only a subset of cores need to be chosen, moderately fast and leaky cores minimize total power consumption under both clocking schemes. This is because very fast cores also consume large leakage energy, while very slow cores necessitate selection of large number of cores for achieving the demanded throughput which lowers the energy efficiency.

Aguilera et al. [2014] note that in a spatially-multitasking GPGPU (where multiple applications execute simultaneously with each one using a subset of the GPU resources) which uses a PV-aware per-core clocking scheme (e.g. as used in [Lee et al. 2011]), assignment of cores to applications can have significant impact on performance. For example, some cores benefit from higher core frequencies while others can tolerate lower frequencies. For such GPGPUs, they propose a performance-improvement technique which works by assigning faster cores to compute-bound and slower cores to memory-bound applications. The classification of applications into compute or memory-bound is done based on whether their performance does or does not increase (respectively) with increasing core frequency.

Jing et al. [2013] compare the impact of PV on SRAM and eDRAM when they are used for designing register file in a GPU. They note that for the SRAM-based register file, PV leads to non-uniform access speeds of different cells and hence, the slowest SRAM cell in the memory determines the overall frequency. For eDRAM, PV manifests itself as variation in retention time of different eDRAM cells under the same nominal speed. Based on this, separate counters can be used for each register entry to record their individual refresh periods and using this, the impact of the cell with the smallest retention time can be restricted to the register entry in which it resides and the entire register file is not affected. They also show that with increasing severity of PV, the energy efficiency of an eDRAM-based register file degrades at much smaller rate than that of an SRAM-based register file.

Most of the existing PV-management techniques have been developed for CPUs. GPUs feature markedly different architecture compared to CPU, and as they become first-class citizens of the computing world, porting existing techniques to and designing novel techniques for GPUs and CPU-GPU heterogeneous systems will be absolutely vital and yet challenging.

6. CONCLUSION AND FUTURE CHALLENGES

We believe that until radically new manufacturing technologies are invented and become commercially viable, architecture- and system-level management of PV is essential for continued performance scaling. In this paper, we highlighted the need of managing PV in computing systems and reviewed several techniques proposed for this purpose. We also underscored the similarities and differences between these techniques. We now conclude with a brief discussion of some challenges that lie ahead in this field.

Most of the works in this field have focused on addressing the effect of PV on individual components. Moving forward, a comprehensive evaluation of how the variation present in multiple components of a system interacts with each other is definitely required. Similarly, since PV-management techniques will be employed in conjunction with other techniques, such as those for addressing power/bandwidth issues, ensuring their synergistic integration will be important to enable their integration in product systems.

The choice of most suitable memory technology (e.g. SRAM vs eDRAM) in future systems depends on the characteristics of the memory technology itself (e.g., PV-susceptibility, density, latency etc.) and the processor component (e.g., first level cache, last level cache, register, scratchpad, main memory, etc.). In addition, application properties, such as inherent error-tolerance of visual graphics applications can allow reducing refresh and ECC requirement under PV. A thorough evaluation of these factors must be performed for making optimal decisions.

From hand-held embedded systems consuming a fraction of a watt, to large-scale data centers and supercomputers consuming tens of megawatts, modern computing systems come in a variety of size and shape. Each of these systems has different architecture, design objective and performance/power/reliability/cost tradeoffs. Accounting for these in PV-management techniques is an interesting step ahead to design techniques optimized for individual platforms.

As the quest of ongoing process technology scaling confronts the formidable challenge of rising process variation, the design of computing systems is likely to undergo a major overhaul. We look forward to an exciting near future where computer architects cross-over this obstacle and design the variation-resilient computing systems of tomorrow.

REFERENCES

Jaume Abella, Xavier Vera, and Antonio Gonzalez. 2007. Penelope: The NBTI-aware processor. In *International Symposium on Microarchitecture*. 85–96.

Amit Agarwal, Bipul Chandra Paul, Hamid Mahmoodi, Animesh Datta, and Kaushik Roy. 2005. A process-tolerant cache architecture for improved yield in nanoscale technologies. *IEEE Transactions on VLSI Systems* 13, 1 (2005), 27–38.

Aditya Agrawal, Amin Ansari, and Josep Torrellas. 2014. Mosaic: Exploiting the Spatial Locality of Process Variation to Reduce Refresh Energy in On-Chip eDRAM Modules. In *International Symposium on High Performance Computer Architecture*. 84 – 95.

Paula Aguilera, Jungseob Lee, Amin Farmahini-Farahani, Katherine Morrow, Michael Schulte, and Nam Sung Kim. 2014. Process variation-aware workload partitioning algorithms for GPUs supporting spatial-multitasking. In *Proceedings of the conference on Design, Automation & Test in Europe*. 176.

Amin Ansari, Shantanu Gupta, Shuguang Feng, and Scott Mahlke. 2009. ZerehCache: Armoring cache architectures in high defect density technologies. In *Int. Symposium on Microarchitecture*. 100–110.

Amin Ansari, Anadi Mishra, Jianping Xu, and Josep Torrellas. 2014. Tangle: Route-oriented dynamic voltage minimization for variation-afflicted, energy-efficient on-chip networks. In *International Symposium on High Performance Computer Architecture (HPCA)*. 440–451.

Bharathan Balaji, John McCullough, Rajesh K Gupta, and Yuvraj Agarwal. 2012. Accurate characterization of the variability in power consumption in modern mobile processors. In *USENIX conference on Power-Aware Computing and Systems (HotPower)*.

Luis Angel D Bathen, Nikil D Dutt, Alex Nicolau, and Puneet Gupta. 2012. VaMV: Variability-aware memory virtualization. In *Design, Automation & Test in Europe Conference*. 284–287.

Mahmoud Bennaser, Yao Guo, and Csaba Andras Moritz. 2008. Data memory subsystem resilient to process variations. *IEEE Transactions on VLSI Systems* 16, 12 (2008), 1631–1638.

Swarup Bhunia, Saibal Mukhopadhyay, and Kaushik Roy. 2007. Process variations and process-tolerant design. In *International Conference on VLSI Design*. 699–704.

Shekhar Borkar, Tanay Karnik, Siva Narendra, Jim Tschanz, Ali Keshavarzi, and Vivek De. 2003. Parameter variations and impact on circuits and microarchitecture. *Design Automation Conference* (2003), 338–342.

Bill Bowhill, Blaine Stackhouse, Nevine Nassif, Zibing Yang, Arvind Raghavan, Charles Morganti, Chris Houghton, Dan Krueger, Olivier Franza, Jayen Desai, Jason Crop, Dave Bradley, Chris Bostak, Sal Bhimji, and Matt Becker. 2015. The Xeon® processor E5-2600 v3: A 22nm 18-core product family. In *IEEE International Solid-State Circuits Conference (ISSCC)*. 1–3.

Keith A Bowman, Steven G Duvall, and James D Meindl. 2002. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *Journal of Solid-State Circuits* 37, 2 (2002), 183–190.

Tuck-Boon Chan, Puneet Gupta, Andrew Kahng, and Liangzhen Lai. 2014. Synthesis and analysis of design-dependent ring oscillator (DDRO) performance monitors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 22, 10 (2014), 2117–2130.

Saumya Chandra, Anand Raghunathan, and Sujit Dey. 2012. Variation-aware voltage level selection. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 20, 5 (2012), 925–936.

Karthik Chandrasekar, Sven Goossens, Christian Weis, Martijn Koedam, Benny Akesson, Norbert Wehn, and Kees Goossens. 2014. Exploiting expendable process-margins in DRAMs for run-time performance optimization. In *Design, Autom. Test Eur.* 1–6.

Karthik Chandrasekar, Christian Weis, Benny Akesson, Norbert Wehn, and Kees Goossens. 2013. Towards variation-aware system-level power estimation of DRAMs: an empirical approach. In *Design Automation Conference*. 23:1–23:8.

Hu Chen, Sanghamitra Roy, and Koushik Chakraborty. 2014a. Exploiting static and dynamic locality of timing errors in robust L1 cache design. In *International Symposium on Quality Electronic Design (ISQED)*. 9–15.

Jie Chen, Guru Venkataramani, and H. Huang. 2014b. Exploring Dynamic Redundancy to Resuscitate Faulty PCM Blocks. *J. Emerg. Technol. Comput. Syst.* 10, 4, Article 31 (2014), 23 pages.

Eric Chun, Zeshan Chishti, and TN Vijaykumar. 2008. Shapeshifter: Dynamically changing pipeline width and speed to address process variations. In *International Symposium on Microarchitecture*. 411–422.

Marcelo Cintra and Niklas Linkewitsch. 2013. Characterizing the impact of process variation on write endurance enhancing techniques for non-volatile memory systems. In *International conference on Measurement and modeling of computer systems (SIGMETRICS)*. 217–228.

Abhishek Das, Serkan Ozdemir, Gokhan Memik, and Alok Choudhary. 2007. Evaluating voltage islands in CMPs under process variations. In *Int. Conference on Computer Design*. 129–136.

Abhishek Das, Berkin Ozisikyilmaz, Serkan Ozdemir, Gokhan Memik, Joseph Zambreno, and Alok Choudhary. 2008. Evaluating the effects of cache redundancy on profit. In *International Symposium on Microarchitecture*. 388–398.

Saurabh Dighe, Sriram R Vangal, Paolo Aseron, Shasi Kumar, Tiju Jacob, Keith A Bowman, Jason Howard, James Tschanz, Vasantha Erraguntla, Nitin Borkar, Vivek De, and Shekhar Borkar. 2011. Within-die variation-aware dynamic-voltage-frequency-scaling with optimal core allocation and thread hopping for the 80-core teraflops processor. *IEEE Journal of Solid-State Circuits* 46, 1 (2011), 184–193.

James Donald and Margaret Martonosi. 2006. Power efficiency for variation-tolerant multicore processors. In *International Symposium on Low Power Electronics and Design (ISLPED)*. 304–309.

Jianbo Dong, Lei Zhang, Yinhe Han, Ying Wang, and Xiaowei Li. 2011. Wear rate leveling: lifetime enhancement of PRAM with endurance variation. *Design Automation Conference* (2011), 972–977.

Cesare Ferri, Sherief Reda, and R Bahar. 2008. Parametric yield management for 3D ICs: Models and strategies for improvement. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 4, 4 (2008), 19.

Kenneth Flamm. 2010. The Impact of DRAM Design Innovation on Manufacturing Profitability. *Future Fab International* 35 (2010).

Francesco Fraternali, Andrea Bartolini, Carlo Cavazzoni, Giampietro Tecchiolli, and Luca Benini. 2014. Quantifying the Impact of Variability on the Energy Efficiency for a Next-generation Ultra-green Supercomputer. In *International Symposium on Low Power Electronics and Design (ISLPED)*. 295–298.

Xin Fu, Tao Li, and Jose Fortes. 2008. NBTI tolerant microarchitecture design in the presence of process variation. In *International Symposium on Microarchitecture*. 399–410.

Xin Fu, Tao Li, and José AB Fortes. 2009. Soft error vulnerability aware process variation mitigation. In *International Symposium on High Performance Computer Architecture*. 93–104.

S. Garg and D. Marculescu. 2013. Mitigating the Impact of Process Variation on the Performance of 3-D Integrated Circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21, 10 (2013), 1903–1914.

Swaroop Ghosh and Kaushik Roy. 2010. Parameter variation tolerance and error resiliency: New design paradigm for the nanoscale era. *Proc. IEEE* 98, 10 (2010), 1718–1751.

Mark Gottscho, Abbas BanaiyanMofrad, Nikil Dutt, Alex Nicolau, and Puneet Gupta. 2014. Power/capacity scaling: Energy savings with simple fault-tolerant caches. In *Design Automation Conference*. 1–6.

Maziar Goudarzi and Tohru Ishihara. 2010. SRAM leakage reduction by row/column redundancy under random within-die delay variation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 18, 12 (2010), 1660–1671.

Maziar Goudarzi, Tadayuki Matsumura, and Tohru Ishihara. 2008. Cache power reduction in presence of within-die delay variation using spare ways. In *IEEE Computer Society Annual Symposium on VLSI*. 447–450.

Puneet Gupta, Yuvraj Agarwal, Lara Dolecek, Nikil Dutt, Rajesh K Gupta, Rakesh Kumar, Subhasish Mitra, Alexandru Nicolau, Tajana Simunic Rosing, Mani B Srivastava, Steven Swanson, and Dennis Sylvester. 2013. Underdesigned and opportunistic computing in presence of hardware variability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 1 (2013), 8–23.

Shantanu Gupta, Amin Ansari, Shuguang Feng, and Scott Mahlke. 2010. StageWeb: Interweaving pipeline stages into a wearout and variation tolerant CMP fabric. In *International Conference on Dependable Systems and Networks (DSN)*. 101–110.

Hadi Hajimiri, Prabhat Mishra, and Swarup Bhunia. 2013. Dynamic cache tuning for efficient memory based computing in multicore architectures. In *Int. Conference on VLSI Design*. 49–54.

Jörg Henkel, Lars Bauer, Nikil Dutt, Puneet Gupta, Sani Nassif, Muhammad Shafique, Mehdi Tahoori, and Norbert Wehn. 2013. Reliable on-chip systems in the nano-era: lessons learnt and future trends. In *Design Automation Conference*. 99:1–99:10.

Sebastian Herbert, Siddharth Garg, and Diana Marculescu. 2012. Exploiting process variability in voltage/frequency control. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 20, 8 (2012), 1392–1404.

Sebastian Herbert and Diana Marculescu. 2009a. Mitigating the impact of variability on chip-multiprocessor power and performance. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 17, 10 (2009), 1520–1533.

Sebastian Herbert and Diana Marculescu. 2009b. Variation-aware dynamic voltage/frequency scaling. In *International Symposium on High Performance Computer Architecture*. 301–312.

Seokin Hong and Soontae Kim. 2013. AVICA: An access-time variation insensitive L1 cache architecture. In *Design, Automation & Test in Europe (DATE)*. 65–70.

Shengyan Hong, Sri Hari Krishna Narayanan, M Kandemir, and Özcan Özturk. 2009. Process variation aware thread mapping for chip multiprocessors. In *Conference on Design, Automation and Test in Europe*. 821–826.

Eric Humenay, David Tarjan, and Kevin Skadron. 2007. Impact of process variations on multicore performance symmetry. In *Design, automation and test in Europe*. 1653–1658.

Mohammed Abid Hussain and Madhu Mutyam. 2008. Block remap with turnoff: a variation-tolerant cache design technique. In *Asia and South Pacific Design Automation Conference*. 783–788.

Aarul Jain, Aviral Shrivastava, and Chaitali Chakrabarti. 2011. LA-LRU: A latency-

aware replacement policy for variation tolerant caches. In *International Conference on VLSI Design (VLSID)*. 298–303.

Lei Jiang, Youtao Zhang, and Jun Yang. 2011. Enhancing phase change memory lifetime through fine-grained current regulation and voltage upscaling. In *international symposium on Low-power electronics and design*. 127–132.

Naifeng Jing, Yao Shen, Yao Lu, Shrikanth Ganapathy, Zhigang Mao, Minyi Guo, Ramon Canal, and Xiaoyao Liang. 2013. An energy-efficient and scalable eDRAM-based register file architecture for GPGPU. *International Symposium on Computer Architecture* (2013), 344–355.

Da-Cheng Juan, Siddharth Garg, and Diana Marculescu. 2011. Statistical thermal evaluation and mitigation techniques for 3D chip-multiprocessors in the presence of process variations. In *Design, Automation & Test in Europe*. 1–6.

Raghavendra K and Madhu Mutyam. 2008. Process Variation Aware Issue Queue Design. *Design, Automation and Test in Europe (DATE)* (2008), 1438–1443. DOI:http://dx.doi.org/10.1145/1403375.1403722

Ismail Kadayif, Mahir Turkcan, Seher Kiziltepe, and Ozcan Ozturk. 2013. Hardware/software approaches for reducing the process variation impact on instruction fetches. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 18, 4 (2013), 54:1–54:23.

Deepa Kannan, Aviral Shrivastava, Vipin Mohan, Sarvesh Bhardwaj, and Sarma Vrudhula. 2008. Temperature and process variations aware power gating of functional units. In *International Conference on VLSI Design*. 515–520.

Nishit Kapadia and Sudeep Pasricha. 2014. Process Variation Aware Synthesis of Application-Specific MPSoCs to Maximize Yield. In *Int. Conference on VLSI Design*. 270–275.

Cheng-Kok Koh, Weng-Fai Wong, Yiran Chen, and Hai Li. 2009. Tolerating process variations in large, set-associative caches: The buddy cache. *ACM Transactions on Architecture and Code Optimization (TACO)* 6, 2 (2009), 8.

Joonho Kong and Sung Woo Chung. 2012. Exploiting narrow-width values for process variation-tolerant 3-D microprocessors. In *Design Automation Conference*. 1197–1206.

Vivek Kozhikkottu, Abhisek Pan, Vijay Pai, Sujit Dey, and Anand Raghunathan. 2014a. Variation Aware Cache Partitioning for Multithreaded Programs. In *Design Automation Conference*. 1–6.

Vivek Kozhikkottu, Swagath Venkataramani, Sujit Dey, and Anand Raghunathan. 2014b. Variation tolerant design of a vector processor for recognition, mining and synthesis. In *International symposium on Low power electronics and design*. 239–244.

Eren Kursun and Chen-Yong Cher. 2008. Variation-aware thermal characterization and management of multi-core architectures. In *International Conference on Computer Design*. 280–285.

Liangzhen Lai, V. Chandra, R.C. Aitken, and P. Gupta. 2014. SlackProbe: A Flexible and Efficient In Situ Timing Slack Monitoring Methodology. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 33, 8 (2014), 1168–1179.

Jungseob Lee, Paritosh Pratap Ajgaonkar, and Nam Sung Kim. 2011. Analyzing throughput of GPGPUs exploiting within-die core-to-core frequency variation. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 237–246.

Jungseob Lee and Nam Sung Kim. 2009. Optimizing total power of many-core processors considering voltage scaling limit and process variations. In *International Symposium on Low power electronics and design*. 201–206.

Xiaoyao Liang, Ramon Canal, Gu-Yeon Wei, and David Brooks. 2007. Process variation

tolerant 3T1D-based cache architectures. In *International Symposium on Microarchitecture*. 15–26.

Vicente Lorente, Alejandro Valero, Julio Sahuquillo, Salvador Petit, Ramon Canal, Pedro López, and José Duato. 2013. Combining RAM technologies for hard-error recovery in L1 data caches working at very-low power modes. In *Conference on Design, Automation and Test in Europe*. 83–88.

Yuanlin Lu and Vishwani D Agrawal. 2007. Statistical leakage and timing optimization for submicron process variation. In *International Conference on VLSI Design*. 439–444.

Islam Mahfuzul, Akira Tsuchiya, Kaoru Kobayashi, Hidetoshi Onodera, and others. 2012. Variation-sensitive monitor circuits for estimation of global process parameter variation. *IEEE Transactions on Semiconductor Manufacturing* 25, 4 (2012), 571–580.

Tayyeb Mahmood and Soontae Kim. 2010. Fine-grained fault tolerance for process variation-aware caches. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 46–51.

Ke Meng and Russ Joseph. 2006. Process variation aware cache leakage management. In *International symposium on Low power electronics and design*. 262–267.

Pietro Mercati, Francesco Paterna, Andrea Bartolini, Luca Benini, and Tajana Simunic Rosing. 2014. Dynamic variability management in mobile multicore processors under lifetime constraints. In *International Conference on Computer Design*. 448–455.

Kartikey Mittal, Arpit Joshi, and Madhu Mutyam. 2011. Timing variation-aware scheduling and resource binding in high-level synthesis. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 16, 4 (2011), 40.

Sparsh Mittal. 2014. A Survey of Architectural Techniques For Improving Cache Power Efficiency. *Elsevier Sustainable Computing: Informatics and Systems* 4, 1 (March 2014), 33–43.

Sparsh Mittal. 2015. A Survey Of Architectural Techniques for Near-Threshold Computing . *ACM Journal on Emerging Technologies in Computing Systems* (2015).

Sparsh Mittal and Jeffrey Vetter. 2015a. A Survey Of Techniques for Architecting DRAM Caches. *IEEE Transactions on Parallel and Distributed Systems (TPDS)* (2015).

Sparsh Mittal and Jeffrey Vetter. 2015b. A Survey of Techniques for Modeling and Improving Reliability of Computing Systems. *IEEE Transactions on Parallel and Distributed Systems* (2015).

Sparsh Mittal and Jeffrey S Vetter. 2015c. AYUSH: A Technique for Extending Lifetime of SRAM-NVM Hybrid Caches. *IEEE Computer Architecture Letters* (2015).

Sparsh Mittal, Jeffrey S Vetter, and Dong Li. 2014. *LastingNVCache: Extending the Lifetime of Non-volatile Caches using Intra-set Wear-leveling*. Technical Report ORNL/TM-2014/374. Oak Ridge National Laboratory, USA.

Sparsh Mittal, Jeffrey S Vetter, and Dong Li. 2015. A Survey Of Architectural Approaches for Managing Embedded DRAM and Non-volatile On-chip Caches. *IEEE Transactions on Parallel and Distributed Systems (TPDS)* (2015).

Mahmoud Momtazpour, Mahboobeh Ghorbani, Maziar Goudarzi, and Esmaeil Sanaei. 2011. Simultaneous variation-aware architecture exploration and task scheduling for MPSoC energy minimization. In *Great lakes symposium on VLSI*. 271–276.

Timothy Prickett Morgan. 2014. http://www.enterprisetech.com/2014/08/13/oracle-cranks-cores-32-sparc-m7-chip/. (2014).

Nayan V Mujadiya. 2009. Instruction scheduling for VLIW processors under variation scenario. In *International Symposium on Systems, Architectures, Modeling, and Simulation*. 33–40.

Madhu Mutyam and Vijaykrishnan Narayanan. 2007. Working with process variation aware caches. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1–6.

Serkan Ozdemir, Yan Pan, Abhishek Das, Gokhan Memik, Gabriel Loh, and Alok Choudhary. 2010. Quantifying and coping with parametric variations in 3D-stacked microarchitectures. In *Design Automation Conference*. 144–149.

Serkan Ozdemir, Debjit Sinha, Gokhan Memik, Jonathan Adams, and Hai Zhou. 2006. Yield-aware cache architectures. In *International Symposium on Microarchitecture*. 15–25.

Gianluca Palermo, Cristina Silvano, and Vittorio Zaccaria. 2012. A variability-aware robust design space exploration methodology for on-chip multiprocessors subject to application-specific constraints. *ACM Transactions on Embedded Computing Systems (TECS)* 11, 2 (2012), 29.

Yan Pan, Joonho Kong, Serkan Ozdemir, Gokhan Memik, and Sung Woo Chung. 2009. Selective wordline voltage boosting for caches to manage yield under process variations. In *Design Automation Conference*. 57–62.

Abu Saad Papa and Madhu Mutyam. 2008. Power management of variation aware chip multiprocessors. In *ACM Great Lakes symposium on VLSI*. 423–428.

Junyoung Park, H Mert Ustun, and Jacob A Abraham. 2012. Run-time prediction of the optimal performance point in DVS-based dynamic thermal management. In *International Conference on VLSI Design (VLSID)*. IEEE, 155–160.

Francesco Paterna, Andrea Acquaviva, Alberto Caprara, Francesco Papariello, Giuseppe Desoli, and Luca Benini. 2012. Variability-aware task allocation for energy-efficient quality of service provisioning in embedded streaming multimedia applications. *IEEE Trans. Comput.* 61, 7 (2012), 939–953.

Somnath Paul and Swarup Bhunia. 2011. Dynamic transfer of computation to processor cache for yield and reliability improvement. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 19, 8 (2011), 1368–1379.

Somnath Paul, Fang Cai, Xinmiao Zhang, and Swarup Bhunia. 2011. Reliability-driven ECC allocation for multiple bit error resilience in processor cache. *IEEE Trans. on Computers* 60, 1 (2011), 20–34.

Matthew Poremba and others. 2015. DESTINY: A Tool for Modeling Emerging 3D NVM and eDRAM caches. In *IEEE Design Automation and Test in Europe*. 1543 – 1546.

Bharathwaj Raghunathan, Yatish Turakhia, Siddharth Garg, and Diana Marculescu. 2013. Cherry-picking: exploiting process variations in dark-silicon homogeneous chip multi-processors. In *Conference on Design, Automation and Test in Europe*. 39–44.

Abbas Rahimi, Daniele Cesarini, Andrea Marongiu, Rajesh K Gupta, and Luca Benini. 2015. Task Scheduling Strategies to Mitigate Hardware Variability in Embedded Shared Memory Clusters. *Design Automation Conference* (2015).

Krishna K Rangan, Michael D Powell, Gu-Yeon Wei, and David Brooks. 2011. Achieving uniform performance and maximizing throughput in the presence of heterogeneity. In *Int. Symposium on High Performance Computer Architecture*. 3–14.

Sherief Reda, Aung Si, and R Bahar. 2009. Reducing the leakage and timing variability of 2D ICs using 3D ICs. In *International Symposium on Low Power Electronics and Design (ISLPED)*. 283–286.

Semeen Rehman, Florian Kriebel, Duo Sun, Muhammad Shafique, and Jörg Henkel. 2014. dTune: Leveraging reliable code generation for adaptive dependability tuning under process variation and aging-induced effects. *Design Automation Conference* (2014), 1–6.

Bogdan F Romanescu, Michael E Bauer, Sule Ozev, and Daniel J Sorin. 2007. Vari-

aSim: simulating circuits and systems in the presence of process variability. *ACM SIGARCH Computer Architecture News* 35, 5 (2007), 45–48.

Bogdan F Romanescu, Michael E Bauer, Sule Ozev, and Daniel J Sorin. 2008. Reducing the impact of intra-core process variability with criticality-based resource allocation and prefetching. In *conference on Computing frontiers*. 129–138.

Smruti Sarangi, Brian Greskamp, Abhishek Tiwari, and Josep Torrellas. 2008b. EVAL: Utilizing processors with variation-induced timing errors. In *Int. Symposium on Microarchitecture*. 423–434.

Smruti R Sarangi, Brian Greskamp, Radu Teodorescu, Jun Nakano, Abhishek Tiwari, and Josep Torrellas. 2008a. VARIUS: A model of process variation and resulting timing errors for microarchitects. *IEEE Trans. on Semiconductor Manufacturing* 21, 1 (2008), 3–13.

Avesta Sasan, Kiarash Amiri, Houman Homayoun, Ahmed M Eltawil, and Fadi J Kurdahi. 2012. Variation trained drowsy cache (VTD-cache): A history trained variation aware drowsy cache for fine grain voltage scaling. *IEEE Transactions on VLSI Systems* 20, 4 (2012), 630–642.

Stuart Schechter, Gabriel H Loh, Karin Straus, and Doug Burger. 2010. Use ECP, not ECC, for hard failures in resistive memories. In *SIGARCH Computer Architecture News*, Vol. 38. 141–152.

Muhammad Shafique, Lujo Bauer, and Jorg Henkel. 2014. Adaptive energy management for dynamically reconfigurable processors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33, 1 (2014), 50–63.

Radu Teodorescu, Jun Nakano, Abhishek Tiwari, and Josep Torrellas. 2007. Mitigating parameter variation with dynamic fine-grain body biasing. In *International Symposium on Microarchitecture*. 27–42.

Radu Teodorescu and Josep Torrellas. 2008. Variation-aware application scheduling and power management for chip multiprocessors. In *ACM SIGARCH Computer Architecture News*, Vol. 36. 363–374.

Abhishek Tiwari, Smruti R Sarangi, and Josep Torrellas. 2007. ReCycle:: pipeline adaptation to tolerate process variation. In *ACM SIGARCH Computer Architecture News*, Vol. 35. 323–334.

Abhishek Tiwari and Josep Torrellas. 2008. Facelift: Hiding and slowing down aging in multicores. In *International Symposium on Microarchitecture*. 129–140.

Yuh-Fang Tsai, Narayanan Vijaykrishnan, Yuan Xie, and Mary Jane Irwin. 2005. Influence of leakage reduction techniques on delay/leakage uncertainty. In *Int. Conf. on VLSI Design*. 374–379.

James W Tschanz, James T Kao, Siva G Narendra, Raj Nair, Dimitri Antoniadis, Anantha P Chandrakasan, Vivek De, and others. 2002. Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage. *IEEE Journal of Solid-State Circuits* 37, 11 (2002), 1396–1402.

Jeffrey Vetter and Sparsh Mittal. 2015. Opportunities for Nonvolatile Memory Systems in Extreme-Scale High Performance Computing. *Computing in Science and Engineering* 17, 2 (2015), 73 – 82.

Jue Wang, Xiangyu Dong, and Yuan Xie. 2012. Point and discard: a hard-error-tolerant architecture for non-volatile last level caches. In *Design Automation Conference*. 253–258.

Jue Wang, Xiangyu Dong, and Yuan Xie. 2014. ProactiveDRAM: A DRAM-initiated retention management scheme. In *International Conference on Computer Design (ICCD)*. 22–27.

Lucas Wanner and others. 2015. NSF expedition on variability-aware software: Recent results and contributions. *it-Information Technology* 57, 3 (2015), 181–198.

Chris Wilkerson, Alaa R Alameldeen, Zeshan Chishti, Wei Wu, Dinesh Somasekhar, and Shih-lien Lu. 2010. Reducing cache power with low-cost, multi-bit error-correcting codes. *ACM SIGARCH Computer Architecture News* 38, 3 (2010), 83–93.

Guihai Yan, Xiaoyao Liang, Yinhe Han, and Xiaowei Li. 2010. Leveraging the core-level complementary effects of PVT variations to reduce timing emergencies in multi-core processors. In *ACM SIGARCH Computer Architecture News*, Vol. 38. 485–496.

Doe Hyun Yoon, Naveen Muralimanohar, Jichuan Chang, Parthasarathy Ranganathan, Norman P Jouppi, and Mattan Erez. 2011. FREE-p: Protecting non-volatile memory against both hard and soft errors. *Int. Symposium on High Performance Computer Architecture* (2011), 466–477.

J. Yun, S. Lee, and S. Yoo. 2014. Dynamic Wear Leveling for Phase-Change Memories With Endurance Variations. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems,* (2014). DOI:http://dx.doi.org/10.1109/TVLSI.2014.2350073

Lide Zhang, Lan S Bai, Robert P Dick, Li Shang, and Russ Joseph. 2009. Process variation characterization of chip-level multiprocessors. In *Design Automation Conference*. 694–697.

Wangyuan Zhang and Tao Li. 2009. Characterizing and mitigating the impact of process variations on phase change based memory systems. In *IEEE/ACM International Symposium on Microarchitecture*. 2–13.

Bo Zhao, Yu Du, Youtao Zhang, and Jun Yang. 2009. Variation-tolerant non-uniform 3D cache management in die stacked multicore processor. In *IEEE/ACM international Symposium on Microarchitecture*. 222–231.

Mengying Zhao, Lei Jiang, Youtao Zhang, and Chun Jason Xue. 2014. SLC-enabled Wear Leveling for MLC PCM Considering Process Variation. In *Design Automation Conference*. 1–6.

Yi Zhou, Chao Zhang, Guangyu Sun, Kun Wang, and Yu Zhang. 2013. Asymmetric-access aware optimization for STT-RAM caches with process variations. In *Great Lakes symposium on VLSI*. 143–148.

Received ab; revised cd; accepted ef