Decoupled Control and Data Processing for Approximate Near-Threshold Voltage Computing

The authors exploit the intrinsic error tolerance of emerging Recognition, Mining, and Synthesis (RMS) applications to mitigate variation. RMS applications can tolerate errors emanating from data-intensive program phases rather than control. Thus, the authors reserve reliable cores for control and execute error-tolerant data-intensive phases on error-prone cores. They also provide a design space exploration for decoupled control and data processing to mitigate variation at near-threshold voltages.

Ismail Akturk
University of Minnesota, Twin Cities

Nam Sung Kim
University of Illinois at Urbana–Champaign

Ulya R. Karpuzcu
University of Minnesota, Twin Cities

Contemporary technology scaling deviates from Dennard's prospects[1] by the escalating power density. The chip power budget can't keep up with the growing power density because of cooling and power delivery limitations. As a result, the number of cores that we can integrate on chip surpasses the number of cores that we can use simultaneously.[2] A promising way to integrate more cores into the available power budget is to reduce the operating voltage, V_{DD}. If V_{DD} remains slightly above the threshold voltage, V_{th}, power consumption decreases by more than an order of magnitude.[3] This unconventional regime, near-threshold voltage computing (NTC), enables more cores to operate simultaneously. Power reduction increases with the proximity of V_{DD} to V_{th}. However, as V_{DD} approaches V_{th}, both the operating frequency, f, and the resilience to parametric variation (that is, the deviation of transistor parameters from design specifications) are reduced.

The lower operating speed restricts NTC's applicability. However, NTC can sustain throughput performance by operating more cores at lower f. Power reduction at near-threshold voltages (NTVs) can easily exceed the power cost of more cores participating in computation. Accordingly, the limited parallel scalability of applications is more likely to restrict the use of more cores than the power budget.[4]

Even if applications featured perfect parallel scalability, parametric variation would

Published by the IEEE Computer Society

preclude aggressive V_{DD} reductions. In each technology generation, manufacturing imperfections exacerbate vulnerability to parametric variation. At conventional, super-threshold voltages (STVs), variation already results in slower cores and ample speed differences among the cores. At lower V_{DD}, transistor speed becomes more sensitive to variation. Therefore, NTC accentuates variation-induced slowdown and speed differences. At the same time, variation-induced timing errors become more likely (see Figure 1). Timing errors emerge if variation slows down logic to prevent operation at the designated clock *f*.

A common STV design practice for eliminating timing errors is to operate at lower than the nominal speed (that is, the sustainable clock *f* if there was no variation). This slowdown to guarantee error-free execution constitutes the timing guardband. Figure 2 depicts the timing guardband as a function of V_{DD}, considering contemporary (22 nm) and near-future (11 nm) technology nodes. The guardband quickly grows as V_{DD} approaches V_{th}, where the nominal *f*—or the sustainable *f* if there was no variation—is already low. Thus, relying on the worst-case guardband is not practical at NTV. A further difficulty stems from the diminishing efficacy of state-of-the-art STV variation mitigation techniques when adapted at NTV.[5,6] Moreover, at NTV, more cores should contribute to computation to counter performance degradation. The corresponding expansion of the chip area is likely to further exacerbate the already intensified vulnerability to variation.

We need NTC-specific variation mitigation techniques. Otherwise, a safe V_{DD} to guarantee error-free execution can barely reach the near-threshold region. To this end, we exploit the intrinsic error tolerance of emerging Recognition, Mining, and Synthesis (RMS) applications that process massive, yet noisy and redundant, input data by probabilistic, often iterative, algorithms. RMS applications can tolerate errors emanating from data-intensive program phases as opposed to control.[7] Therefore, our proposed architecture, Accordion, reserves reliable cores for control and executes error-tolerant data-intensive phases on error-prone cores.

In this article, we provide a design space exploration for decoupled control and data processing to mitigate the impact of variation at NTV. We refine our exploration to three promising Accordion design points[9]: three clustered many-core architectures featuring

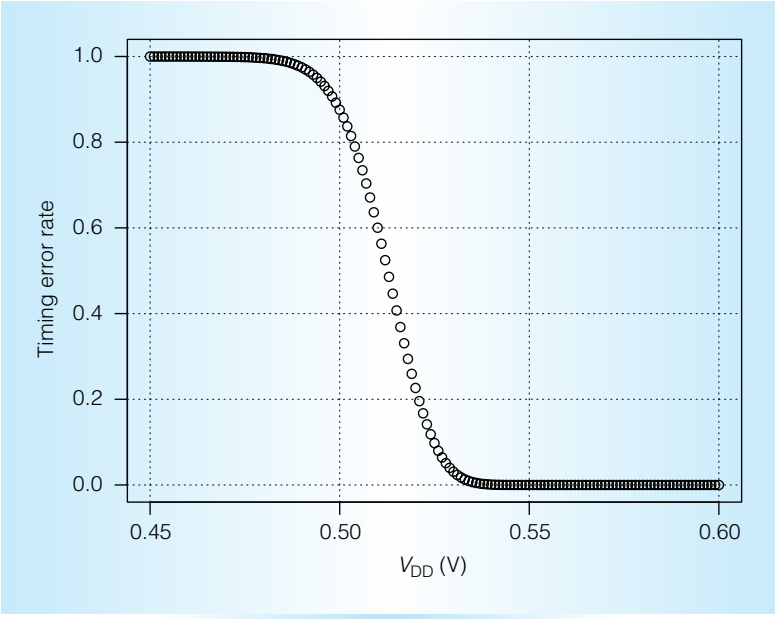

Figure 1. Variation-induced timing error rate. Such errors become inevitable as operating voltage (V_{DD}) reaches to V_{th}.

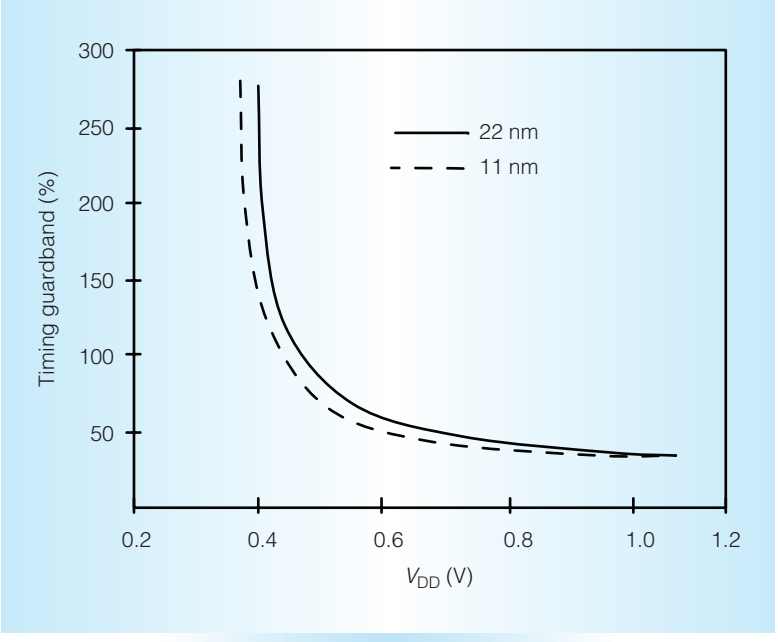

Figure 2. Timing guardband to mask variation-induced errors. As operating voltage (V_{DD}) reaches the threshold voltage (V_{th}), the timing guardband required exceeds practical limits. The situation exacerbates with technology scaling.[8]

different types of heterogeneity. Here, we compare and contrast these three design points in terms of energy efficiency and complexity.

Exploiting algorithmic error tolerance

RMS applications can mask errors in data-intensive program phases as opposed to control. However, to be able to embrace errors,

- errors should be confined to error-tolerant data-intensive phases, such that they manifest as degradation in the accuracy of computation; and
- the degradation in accuracy should remain within acceptable boundaries.

To meet the first condition, our NTV hardware executes error-tolerant data-intensive phases on error-prone cores and reserves reliable cores for control. In other words, we enforce variation-induced errors to be contained where they can be tolerated: within data-intensive program phases. Thread decomposition of most RMS algorithms already conforms to decoupled data and control, in order to distinguish data-intensive parallel tasks from control-intensive ones. For this study, we relied on a coarse-grained characterization, assuming that the main thread in charge of distributing and collecting data to and from worker threads is error free, where worker threads are error prone. This model complies with bulk-synchronous or pipelined parallelism.

The second condition, on the other hand, demands the capability to configure the accuracy of computation explicitly. We can devise an application-specific set of input parameters such as time-step granularity or resolution to serve the purpose. Such input parameters usually associate with the problem size[9–11]: by expanding the problem size, we can configure the application to generate an output of higher accuracy. A larger problem size can facilitate a lower near-threshold V_{DD} by engaging more cores to computation. A lower V_{DD} increases the vulnerability to variation, but the larger problem size can mask an accuracy loss due to variation-induced errors, provided that we meet the first condition.

Decoupling control from data processing

Accordion runs all cores engaged in data-intensive computation at the same f to ensure that parallel tasks make similar progress. This typically leads to faster overall execution by eliminating any synchronization overhead incurred if cores operated at different speeds.

An expanded problem size activates more cores to operate at a lower f. As the number of active cores expands, cores suffering from substantial variation-induced slowdown become more likely to participate in computation. These cores can limit the overall operating f. A compressed problem size, on the other hand, activates fewer cores to operate at a higher f. The lower number of active cores gives the runtime more freedom in picking the cores. Because of the higher operating f and the increased likelihood of including more resilient and faster cores, a compressed problem size does not necessarily degrade the accuracy severely.

Because Accordion is tailored to operate at high error rates, it enables operation at higher f than a designated safe f (to guarantee error-free execution). As a safety net, we can rely on checkpoint/recoveries, yet at a reduced complexity: due to the masking of (a subset of) errors, we anticipate less frequent checkpoints and recoveries.

We hold variation-induced errors where the application can tolerate them. Accordion executes error-tolerant data-intensive program phases on error-prone data cores (DCs) and reserves more reliable control cores (CCs) for control. CCs and DCs work in master-slave mode. Each CC coordinates computation on a designated set of DCs.

CCs exclude any type of error. In order to prevent catastrophic failures or hangs, CCs should comprise robust transistors and circuits, by construction. For enhanced resilience, CCs can operate at higher V_{DD}. Alternatively, a variation-afflicted NTV chip can reserve the most resilient cores for CCs. CCs periodically check whether DCs have completed the assigned computation. CCs are in charge of housekeeping; once DCs finish computations, the master CCs merge or reduce results from different DCs. To detect potential crashes or hangs of DCs, CCs keep watchdogs on a per-DC basis. To prevent error propagation from DCs, CCs never rely on data produced by DCs for control. CCs communicate with DCs over a dedicated memory location, both to find out whether slave DCs are done with computation

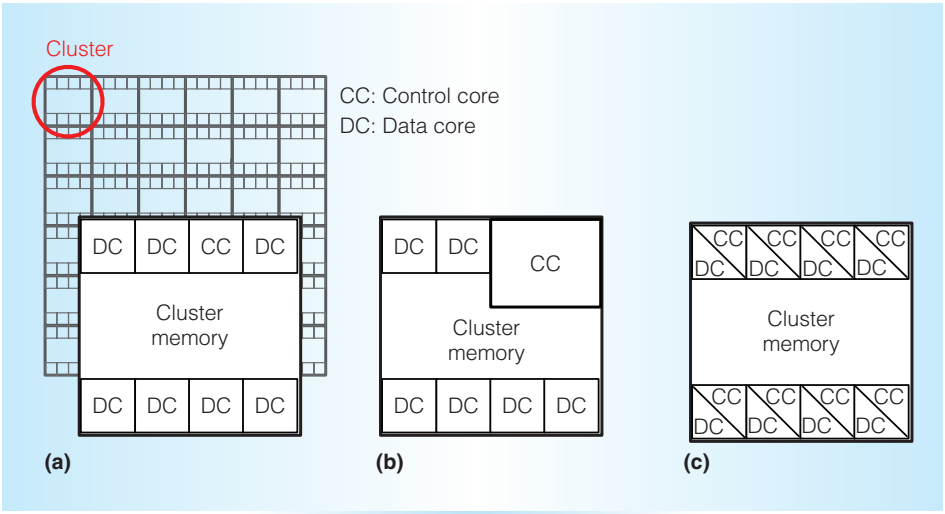

Figure 3. Design space exploration. (a) Spatiotemporal heterogeneity. All cores are identical by design, and the cores identified as most reliable post manufacturing testing are assigned as control cores. (b) Design-driven heterogeneity. Each cluster has a dedicated and specialized control core by design. (c) Temporal heterogeneity. All cores are identical by design, but unlike in spatiotemporal heterogeneity, control cores are not fixed post manufacturing testing; a core can serve both as a control core and a data core.

and to collect DC results. DCs can read only; they cannot modify the data produced by master CCs.

To facilitate effective coordination with CCs, DCs feature fast reset and restart hardware. A DC has accesses to a private read-write memory where the DC can write (either organized as a scratchpad or to communicate with other cores), in addition to a read-only memory where shared data managed by master CCs reside. To avoid error propagation, DCs cannot write to the private space of CCs or other DCs. Instead, a dedicated memory location serves intra-DC communications.

Design space exploration

Figure 3 demonstrates a hypothetical NTV chip, clustered to enhance scalability. A few cores with per-core private memories and a shared cluster memory constitute each cluster. Various options exist to differentiate control cores from data cores.

In spatiotemporal heterogeneity (ST-H), each cluster accommodates identical cores (Figure 3a). There is no difference in the design of CCs and DCs. ST-H distinguishes CCs from DCs spatiotemporally: CCs correspond to the fastest, most reliable cores (that is, the cores of minimum slowdown under variation). This option is simpler from a hardware perspective, yet we should program all the semantics discussed earlier. Still, the organization is flexible in that we can configure the number of CCs (although the example from Figure 3a depicts one CC per cluster). Because variation governs the f of CCs, the range of CC frequencies may differ across chips.

In design-driven heterogeneity (D-H), CCs and DCs per cluster represent different types of cores by design (Figure 3b). DCs and CCs specialize; we can implement the semantics discussed earlier directly in hardware. Under D-H, the number of CCs can easily become a bottleneck. The example from Figure 3b assumes one CC per cluster. Depending on the application, a higher or a lower CC to DC ratio could be favorable. We anticipate that CCs will consume more area than DCs due to the control-intensive specialization and the demand for enhanced reliability. D-H can employ single-instruction, multiple-data (SIMD) DCs to further improve energy efficiency without sacrificing serial performance.

In temporal heterogeneity (T-H), CCs and DCs are identical by design (Figure 3c). T-H distinguishes CCs from DCs temporally,

by time-multiplexing each core between CC and DC functionality during execution. This option provides a better use of hardware resources; however, it complicates the design because of the required support for different memory protection domains.

Evaluation

To quantify how the different design points affect the energy efficiency versus accuracy tradeoff, we deployed a hypothetical NTV chip of eight clusters (Figure 3) at 22 nm. Each cluster comprises eight cores, one of which acts as the CC at any one time. Each core is a single-issue engine wherein memory accesses and computation can overlap. Per-core memories represent private L1 cache (64-Kbyte write through, four-way, 64-byte line), the cluster memory, a shared L2 cache across all eight cores (2-Mbyte write back, 16-way, 64-byte line). The coherence protocol is a fully mapped, directory-based MESI with each pointer corresponding to one cluster. A bus inside a cluster and a 2D torus across clusters constitute the network. The nominal values of V_{DD} and f at NTV are 0.6 V and 1.2 GHz (which approximately correspond to 0.85 V and 2.4 GHz at STV), respectively.

Variation model

We deployed VARIUS-NTV to determine the operating f and estimate the timing error rate as a function of f at the designated near-threshold V_{DD}.[12] VARIUS-NTV captures both spatially correlated and random variation due to manufacturing imperfections. VARIUS-NTV parameter values are $(\sigma/\mu)V_{th} = 10\%$ and $\varphi = 0.1$. We do not consider V_{DD} noise. Because variation causes a slowdown in logic, whether or not core logic fails totally depends on the operating V_{DD} and f. Accordingly, we first extract the minimum V_{DD}, V_{DDMIN}, that each cluster requires to remain functional at NTV. If clusters operate below V_{DDMIN}, memory blocks might not be able to hold or change state. Per-cluster V_{DDMIN} is defined as the maximum V_{DDMIN} across all memory blocks within a cluster. We designate the maximum per cluster V_{DDMIN} as the chip-wide near-threshold V_{DD}.

We assign tasks to cores at the granularity of clusters. Per-cluster phase-locked loops (PLLs) generate per-cluster f. PLLs share one separate analog voltage domain similar to many commercial processors. PLL frequency is not a function of voltage but PLL settings. Thus, we can adjust each PLL's frequency as needed. Accordion runs all clusters assigned to a parallel application at the same f to ensure that threads make similar progress.

Power and performance models

To evaluate performance, we used the Sniper microarchitectural simulator.[13] The power analysis relies on McPAT.[14] We set the power budget to 100 W. For ST-H and T-H, CCs and DCs differ by their operating V_{DD} and f. Accordingly, we fed our McPAT-based power model with different (V_{DD}, f) configurations. Such (V_{DD}, f) pairs depend on the variation profile.

Benchmarks

We experimented with select RMS benchmarks from Parsec[15] and Rodinia[16] suites. The benchmarks cover emerging application domains. Table 1 captures benchmark characteristics. For each benchmark, we identify at least one application input parameter to govern both the problem size and the output accuracy.[9] We change the problem size by adjusting these input parameters. Such changes in the problem size do not result in a different problem—we still solve the very same problem, but with different accuracy. We deploy application-specific metrics to quantify the changes in output accuracy.

Error model

To model the application-layer manifestation of timing errors, we conservatively ignored potential masking of errors at various system-stack layers. Variation renders a different timing error probability per cycle, P_{err}, for each core. We assume that any timing error of probability higher than $P_{err} = 10^{-12}$ reaches the application layer. In this case, we drop the computation (that is, ignore the end result all together) of the thread executing on the erroneous core.

This model can capture close-to-worst-case error manifestation under Accordion's decoupled execution model. Per-core errors can manifest as the following:

Table 1. Evaluated benchmarks

Benchmark	Application domain	Input parameter	Accuracy metric
canneal (Parsec)	Optimization	Swaps per temperature step Number of temperature steps	(Relative) routing cost
ferret (Parsec)	Similarity search	Size factor	Number of common images
bodytrack (Parsec)	Computer vision	Number of annealing layers	Sum of square differences
hotspot (Rodinia)	Physics simulation	Number of iterations	Sum of square differences
srad (Rodinia)	Image processing	Number of iterations	Peak signal-to-noise ratio (PSNR)

- no termination due to crashes or hangs,
- termination with excessive accuracy loss, or
- termination with acceptable accuracy loss.

We rely on CCs to detect the first type of error—for example, by deploying watchdog timers. The application layer perceives the error as the computation (thread) on the erroneous core being dropped. In the second case, timing errors affect data-intensive phases, which results in unacceptable accuracy. We expect CCs to capture the second type of error by enforcing preset limits on maximum accuracy loss. CCs can drop computations not conforming to such limits. On the other hand, we do not require CCs' intervention under the third type of error. We expect the accuracy loss in this case to remain less than the accuracy loss incurred by the first type of error.

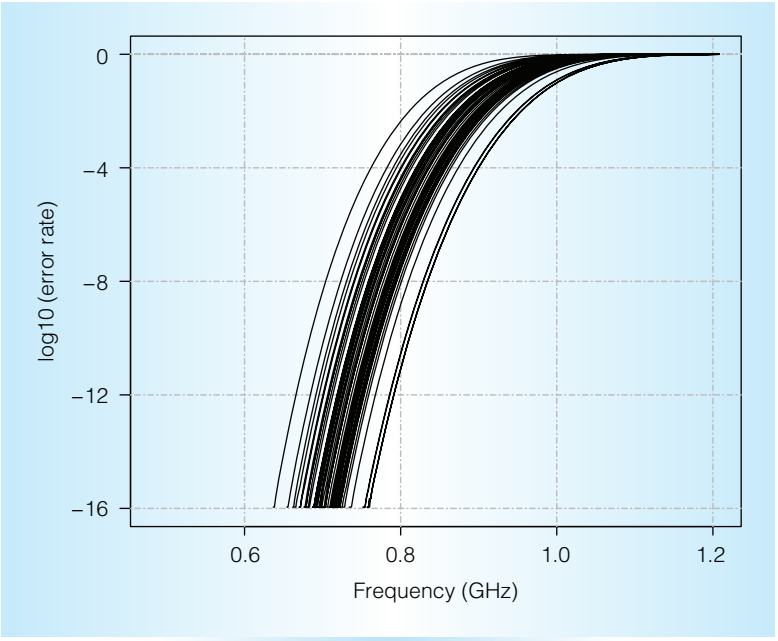

Figure 4. Impact of parametric variation. Parametric variation renders ample differences between the safe operating frequencies of the cores.

Impact of variation

Figure 4 depicts the variation-induced timing error rate per cycle, P_{err}, at the nominal near-threshold V_{DD} of 0.6 V (slightly higher than the maximum per cluster V_{DDMIN} across chip), as a function of f. The data reflects the median chip out of 100 chips we experimented with. The figure demonstrates each core's P_{err} curve, rendering 64 curves for 64 cores. The P_{err} axis is in log scale. P_{err} values rapidly increase to reach 1, as f increases beyond 0.8 GHz. At acceptably low P_{err} of $[10^{-16}, 10^{-12}]$, the majority of the cores cannot operate at the nominal f of 1.2 GHz (which represents the sustainable f were there no variation). P_{err} in the range of $[10^{-16}, 10^{-12}]$ induces a timing error every $[10^{12}, 10^{16}]$ cycles, respectively. At $P_{err} = 10^{-16}$, the f of the slowest core becomes 0.56 times the nominal f.

Design space exploration

Next, we examined how the tradeoff space of energy efficiency versus accuracy differs across the three design points. We considered different problem sizes for the benchmarks from Table 1. As we changed the problem size, we modulated the number of active cores such that the execution time at NTV comes as close to the execution time at STV as possible—we imposed a threshold of no more than 30 percent slowdown. The STV

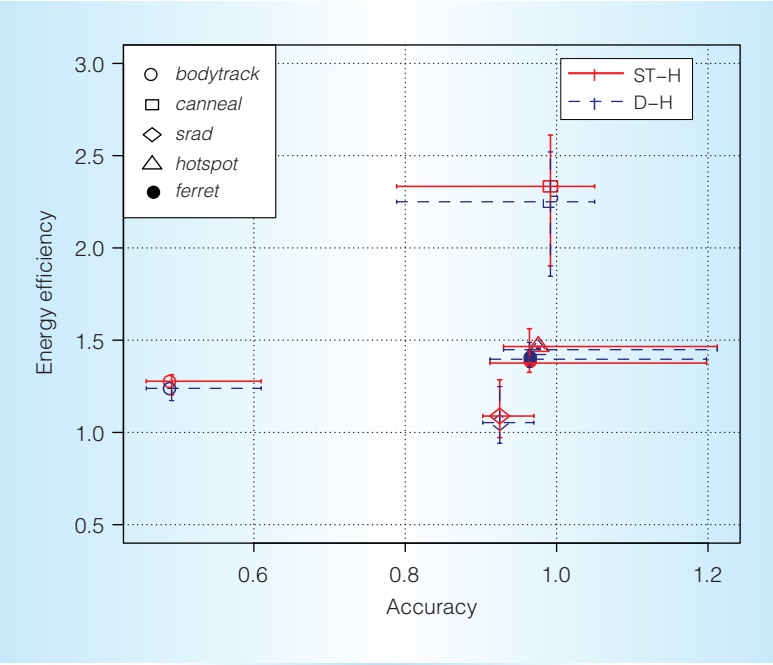

Figure 5. Energy efficiency versus accuracy for spatiotemporal (ST-H) and design-driven (D-H) heterogeneity. The vertical (horizontal) error bars depict how the relative energy efficiency (accuracy) evolves as we sweep the problem size.

baseline is an eight-core design operating at nominal superthreshold V_{DD} and f. We omitted variation-induced slowdown at STV to determine a lower bound for energy efficiency gain at NTV.

ST-H assigns the fastest core in each cluster to operate as the CC. The slowest of the CCs across all clusters determines the operating f of the entire chip at the designated near-threshold V_{DD}, f_{ST-H}. In this manner, we prevent error onset in CCs. On the other hand, none of the DCs can operate at this f_{ST-H} without error onset, by construction. We consult Figure 4 to determine each DC's P_{err}. We drop the thread running on any DC with a P_{err} value exceeding 10^{-12}. Under variation, f_{ST-H} could cause all computation to be dropped if all DCs rendered higher P_{err} than 10^{-12} at f_{ST-H}. If this is the case, we adjust (that is, reduce) f_{ST-H} such that no more than one-eighth of the threads is dropped.

The f assignment under D-H proceeds similarly, except that the CC per cluster is fixed by design in this case. The slowest of the CCs at the designated near-threshold V_{DD} determines f_{D-H}. If none of the DCs can operate at f_{D-H} without the P_{err} exceeding 10^{-12}, we reduce f_{D-H} such that no more than one-eighth of the threads is dropped. We assume that CCs are 25 percent larger than DCs.

Figure 5 compares spatiotemporal heterogeneity (ST-H, marked with a solid line) and design-driven heterogeneity (D-H, marked with a dashed line). Energy efficiency in MIPS/watt (y-axis) and accuracy (x-axis) are both normalized to the nominal STV operating point. Each point in the graph corresponds to the default problem size (*simsmall* or equivalent). The vertical (horizontal) error bars depict how the relative energy efficiency (accuracy) evolves as we sweep the problem size. For each application, we experimented with five representative problem sizes within 0.5 to 2.3 times of the default problem size for *bodytrack, canneal, srad,* and *hotspot,* and within 0.85 to 2.9 times for ferret, respectively. Except for *srad* and *ferret,* all applications from Figure 5 finish within the STV execution time. The slowdown of *srad* is approximately 27 percent; for *ferret,* approximately 13 percent, both under ST-H and D-H. *canneal* and *bodytrack* use 64 cores; *srad* and *hotspot,* 32 cores; and *ferret,* only 16 cores. We observe that the accuracy is more sensitive to changes in the problem size when compared to the energy efficiency. Figure 5 captures how energy efficiency changes with P_{err}, because each accuracy value maps to a particular P_{err}: the higher the P_{err}, the lower the accuracy. However, depending on the application characteristics, in what way and how fast a Δ increase in P_{err} reduces the accuracy varies. For a per-benchmark characterization of accuracy versus P_{err}, refer to our previous work.[9] For some applications, the relative accuracy exceeds 1 at larger problem sizes: this indicates that we can operate at even higher f (which render higher P_{err}, according to Figure 4) and translate the excess accuracy into higher energy efficiency. Overall, ST-H delivers a slightly better energy efficiency/accuracy tradeoff than D-H.

Figure 6 depicts energy efficiency versus accuracy under temporal heterogeneity (T-H). Energy efficiency and accuracy are normalized to the STV operating point. This time, we experiment with the default problem size (*simsmall* or equivalent) only. Each point of

Figure 6 characterizes a different assignment of cores to operate as CCs. T-H differentiates between CCs and DCs at runtime. Accordingly, the energy efficiency evolves with the overhead of the runtime algorithm to orchestrate the assignments. In the following, we omit such algorithmic overheads and report how energy efficiency versus accuracy changes across the entire space of feasible CC assignments. Recall that we experimented with one CC per cluster. We let all clusters operate at the same f, which is set by the slowest CC among all clusters. In Figure 6, for each point, a different core represents the slowest CC, CC_{slwst}, across the chip. The f of CC_{slwst} at the designated near-threshold V_{DD}, f_{T-H}, sets the operating f of all DCs and CCs assigned to a parallel application under T-H. If none of the DCs can operate at f_{T-H} without the P_{err} exceeding 10^{-12}, we reduce f_{T-H} such that no more than one-eighth of the threads is dropped.

However, not all cores can represent CC_{slwst}. For example, no point in Figure 6 has the fastest core on the chip as CC_{slwst}. If we set the fastest core on chip as CC_{slwst}, we would not be able to prevent error onset in the remainder of the CCs on chip, violating Accordion's execution semantics. We observe that T-H does not deliver a better tradeoff when compared to ST-H, which renders ST-H the most complexity-effective design point.

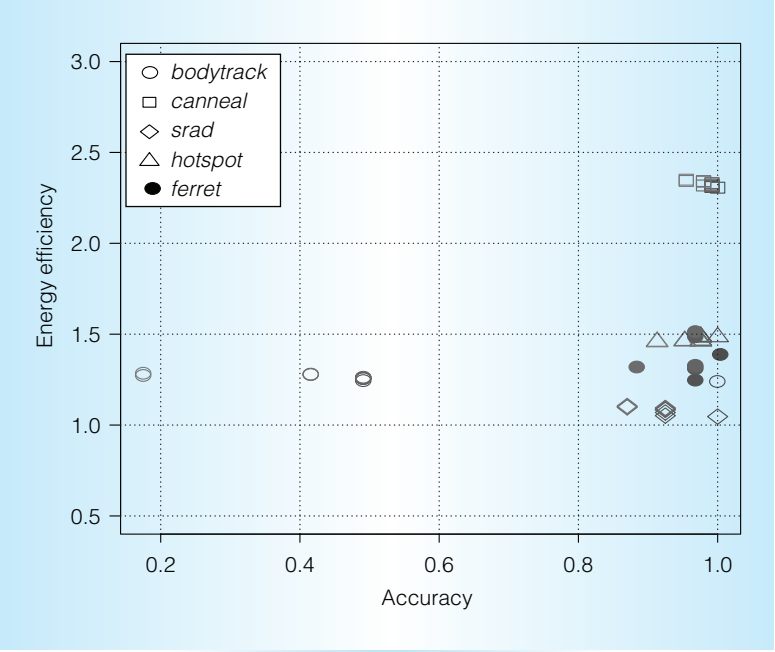

Figure 6. Energy efficiency versus accuracy for temporal heterogeneity (T-H). Each point characterizes a different assignment of cores to operate as control cores. The slowest control core determines the operating voltage of all of the clusters.

In this study, we deployed a conservative error model. Regarding error propagation at the application level, we are exploring whether recurring error patterns apply that we can exploit to develop more accurate error models. A lack of recurring patterns is likely to render statistical fault injection as the only means of assessing manifestation of errors at higher system-stack levels. In this case, rather than modeling, scalable and fast statistical fault injection should be explored. Statistical fault injection is inevitable in characterizing system-level manifestation of errors and assessing the fidelity of existing models.

Scaling of the problem size with the core count does not always translate into fixed per-thread work, as weak scaling in the strict sense would imply. Applications strictly conforming to weak scaling would benefit most from Accordion operation. However, for the selected RMS benchmarks we deployed, per-thread work tends to increase with problem size. We are extending our study to strict weak scaling, considering novel application domains. We are also examining how to orchestrate Accordion operation dynamically at runtime. Although we can change the number of cores assigned to computation midst execution, the problem size does not always lend itself well to such fine-grained adaptation. For this study, we let resource allocation and the assigned operating point apply for the entire duration of execution. However, both the application phases and the hardware resources can experience changes in resiliency during execution. At the very least, data- and control-centric phases can be interleaved in numerous ways. How to capture such fine-grained temporal changes in resiliency is another open research question. MICRO

Acknowledgments

This work was supported in part by generous grants from the NSF (CCF-1421988, CCF-0953603) and DARPA (HR0011-12-2-0019). Nam Sung Kim has a financial interest in AMD.

References

1. R. Dennard et al., "Design of Ion-Implanted MOSFETs with Very Small Physical Dimensions," *IEEE J. Solid-State Circuits*, Oct. 1974, pp. 256–267.
2. M.B. Taylor, "Is Dark Silicon Useful? Harnessing the Four Horsemen of the Coming Dark Silicon Apocalypse," *Proc. 49th ACM/EDAC/IEEE Design Automation Conf.*, 2012, pp. 1131–1136.
3. R.G. Dreslinski et al., "Near-Threshold Computing: Reclaiming Moore's Law through Energy Efficient Integrated Circuits," *Proc. IEEE*, vol. 98, no. 2, 2010, pp. 253–266.
4. N. Pinckney et al., "Assessing the Performance Limits of Parallelized Near-Threshold Computing," *Proc. 49th Design Automation Conf.*, 2012, pp. 1143–1148.
5. R.G. Dreslinski et al., "Reevaluating Fast Dual-Voltage Power Rail Switching Circuitry," presented at 10th Ann. Workshop Duplicating, Deconstructing, and Debunking, 2012.
6. U.R. Karpuzcu et al., "EnergySmart: Toward Energy-Efficient Manycores for Near-Threshold Computing," *Proc. IEEE 19th Int'l Symp. High Performance Computer Architecture*, 2013, pp. 542–553.
7. H. Cho et al., "ERSA: Error Resilient System Architecture for Probabilistic Applications," *Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 4, 2012, pp. 1560–1565.
8. L. Chang et al., "Practical Strategies for Power-Efficient Computing Technologies," *Proc. IEEE*, Feb. 2010, pp. 215–236.
9. U.R. Karpuzcu et al., "Accordion: Toward Soft Near-Threshold Voltage Computing," *Proc. 20th Int'l Symp. High Performance Computer Architecture*, 2014, pp. 72–83.
10. V. Chippa et al., "Dynamic Effort Scaling: Managing the Quality-Efficiency Tradeoff," *Proc. 48th Design Automation Conf.*, 2011, pp. 603–608.
11. M. de Kruijf et al., "Relax: An Architectural Framework for Software Recovery of Hardware Faults," *Proc. 37th Ann. Int'l Symp. Computer Architecture*, 2010, pp. 497–508.
12. U.R. Karpuzcu et al., "VARIUS-NTV: A Microarchitectural Model to Capture the Increased Sensitivity of Manycores to Process Variations at Near-Threshold Voltages," *Proc. 42nd Ann. IEEE/IFIP Int'l Conf. Dependable Systems and Networks*, 2012, pp. 1–11.
13. T.E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the Level of Abstraction for Scalable and Accurate Parallel Multi-core Simulation," *Proc. Int'l Conf. High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 1–12.
14. S. Li et al., "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," *Proc. 42nd Ann. IEEE/ACM Int'l Symp. Microarchitecture*, 2009, pp. 469–480.
15. C. Bienia et al., *The PARSEC Benchmark Suite: Characterization and Architectural Implications*, tech. report TR-811-08, Dept. of Computer Science, Princeton Univ., 2008.
16. S. Che et al., "Rodinia: A Benchmark Suite for Heterogeneous Computing," *Proc. IEEE Int'l Symp. Workload Characterization*, 2009, pp. 44–54.

Ismail Akturk is a PhD student in the Department of Electrical and Computer Engineering at the University of Minnesota, Twin Cities. His research focuses on energy-efficient computer architecture and approximate computing. Akturk has an MS in computer engineering from Bilkent University, Turkey. Contact him at aktur002@umn.edu.

Nam Sung Kim is an associate professor in the Department of Electrical and Computer Engineering at the University of Illinois at Urbana–Champaign. His research focuses on devices, circuits, and architectures for power-efficient computing. Kim has a PhD in computer science and engineering from the University of Michigan. Contact him at nskim@illinois.edu.

Ulya R. Karpuzcu is an assistant professor in the Department of Electrical and Computer Engineering at the University of Minnesota, Twin Cities. Her research focuses on the impact of process technology on computing systems. Karpuzcu has a PhD in computer engineering from the University of Illinois at Urbana–Champaign. Contact her at ukarpuzc@umn.edu.