Recovery-Driven Design

Andrew B. Kahng[†+], Seokhyeong Kang[†], Rakesh Kumar[‡] and John Sartori[‡]
[†]ECE and [+]CSE Departments, University of California at San Diego
[‡]Coordinated Science Laboratory, University of Illinois

***Abstract*—This paper presents an *error rate-optimal* approach to reducing processor power whereby a processor is designed from the ground up to deliberately allow voltage overscaling-based timing errors during nominal operation, while a software or hardware-based error resilience technique is used to tolerate these errors. This approach represents a shift from the traditional design philosophy of designing a processor core for correctness and then correcting any errors that occur under various operating conditions.**

We use our *recovery-driven* design methodology, in which a processors is optimized with an error resilience mechanism in mind, to optimize for Razor [2] hardware error tolerance and for error resilient applications [9]. We also show that a processor core-level methodology can be used for the design of *heterogeneously reliable multi-core processors*, i.e., chip multiprocessor designs where different cores are power-optimized for different reliability targets. We show that recovery driven processors and heterogeneously reliable multi-core processors have substantial power benefits over their conventional single-core and multi- core counterparts.

I. Introduction

Power has been, for some time now, a first order design constraint for microprocessors [16]. In fact, performance, yield, and functionality are routinely sacrificed for power considerations ([5], [11]) today.

Processors are often designed conservatively to allow correct operation under worst-case conditions. Applying a power reduction technique such as voltage scaling reduces power, but the benefits are limited by the inherently conservative nature of the baseline worst-case design [1]. Some better-than-worst-case (BTWC) techniques [1] have recently been proposed to eliminate guardbands against worst-case conditions. However, the processor is still designed for correct operation in the average case, limiting how small the operating voltage can be for a given frequency, and thereby limiting the power reduction that is possible.

A recent study [8] has suggested that significant power benefits may be possible for a hardware module from a CAD-level methodology that minimizes the power of the module for a target timing error rate. The outcome of the CAD methodology is a modified design for a hardware module that targets an error rate and consumes less power than the baseline.

This paper extends the module-level methodology in [8] to the processor level to enable a novel approach to reducing single-core and multi-core processor power – designing a processor from the ground up to deliberately allow voltage overscaling-based timing errors ([4],[2]) during nominal operation, while using a software or hardware-based error-resilience technique to tolerate these errors.

The proposed *error rate-optimal* approach to processor design has several architectural implications. For example, one can now design *recovery-driven processors*, i.e., single-core processors whose power is optimized for a target error recovery mechanism. The target error recovery mechanism can either be timing speculation-based or it may simply be an application with inherent error tolerance ([4],[12]).

Similarly, the proposed approach enables *heterogeneously reliable multi-core designs* where each core is power-optimized for a different reliability requirement or recovery technique. Applications are mapped to the appropriate core based on their robustness in the face of errors. Such heterogeneously reliable multi-core architectures adapt to diversity in applications and allow higher power savings for similar levels of performance.

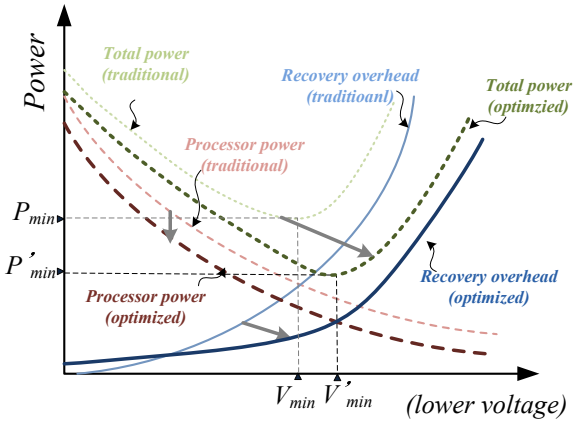

Fig. 1. Design for a target error rate ensures that the target error rate is exceeded at a substantially lower voltage of an already lower power processor, significantly reducing the power consumption at the target error rate, lowering recovery overhead for hardware-based error tolerance and decreasing degradation in output quality for software-based error tolerance.

II. Minimizing Power of a Circuit Module for a Target Error Rate

Details on the *PowerOptimizer* heuristic, which minimizes the power consumption of a circuit module for a given error rate, can be found in [8].

III. An Architecture-aware Methodology for Processor Power Reduction

The *PowerOptimizer* procedure optimizes a hardware *module* to minimize power consumption while ensuring that the error rate of the module will be not exceed a specified target. A processor-level recovery-driven design heuristic must choose which modules to optimize, as well as the error rate targets and operating voltage for the optimized modules such that processor power is minimized.

Algorithm 1 shows a heuristic for minimizing processor power for a target error rate. The first step of the power-minimization heuristic involves characterizing the modules of the processor core in terms of their power consumption at different error rate and voltage targets. These data are provided by *PowerOptimizer* and used to select the optimal operating voltage(s) for the processor core as well as the error rate targets to assign to the processor modules.

The next step in the processor-level heuristic is to use the data from *PowerOptimizer* to solve an optimization problem. The optimization objective is to minimize the power of the processor core subject to the constraint that the processor error rate must be less than the chosen target rate. Using the data from *PowerOptimizer*, we can formulate expressions for the power and error rate of the processor core in terms of the module error rates and the operating voltage. Thus, the goal of the optimization problem for a particular voltage is to find the assignment of error rate targets to modules that satisfies the optimization objective. We used a disjunctively-constrained knapsack-based [15] approach to solve the optimization problem. Finally, the heuristic selects the voltage and error rate assignment for which power of the processor core is minimized and performs Engineering Change Order (ECO) for each module using the optimized netlist for the target voltage and assigned error rate.

Algorithm 1 Processor-level Design Heuristic.

Procedure $OptimizeProcessor(ER_{target}, MODULES, DOMAINS)$

1. **for each** module m in the optimization list of MODULES **do**
2. **for each** error rate $ER < ER_{target}$ **do**
3. $PowerOptimizer(N(m), ER)$;
4. **end for**
5. Use the results from $PowerOptimizer$ to characterize $P_m(V, ER)$
6. **end for**
7. **for each** voltage $V \in V_{range}$ **do**
8. Minimize $P_{core}(V) = \Sigma(P_m(V, ER))$ s.t. $ER_{core}(ER_{module_1}, ..., ER_{module_M}) \leq ER_{target}$
9. Record minimum power $P^{min}_{core}(V)$ and module error rate assignment $S(V) = [ER_{module_1}, ..., ER_{module_M}]$
10. **end for**
11. Select the voltage V_{opt} at which power P^{min}_{core} is minimized
12. Let $V^*(S(V)[m])$ be the voltage that minimizes power for module m at $ER = S(V)[m]$
13. Locate the $DOMAINS$ neighbors $\{V_1, ..., V_{DOMAINS}\}$ nearest to the set of voltages $V^*(S(V_{opt}))$
14. Assign each module m to the voltage domain $V_D[m] \in \{V_1, ..., V_{DOMAINS}\}$ that minimizes power $P_m(V_D[m], S(V_{opt})[m])$
15. Perform ECO for each module $m \in$ MODULES with netlist $N(m, V_D[m], S(V_{opt})[m])$;

Fig. 2. Processor core power-minimization heuristic.

IV. Architectural Implications

In this section, we discuss two classes of processor designs that are enabled by the proposed error rate-optimal processor design methodology.

A. Recovery-driven Processors

The proposed design methodology enables *recovery-driven processors* – single core processors that are optimized to deliberately produce timing errors at a rate that can be gainfully tolerated by a hardware or software-based error tolerance technique.

1) Case Study: Circuit-level Timing Speculation: One popular hardware-based scheme for error detection and correction is circuit-level timing speculation. Razor [2] is one good example of a circuit-level timing speculation-based scheme.

A recovery-driven processor design targeted for Razor takes into account the frequency of errors that can be gainfully tolerated by Razor (determined by error recovery overhead) as well as the number of latches in which an error is allowed (which determines the cost of making the circuit robust to errors).

The methodology for producing a recovery-driven processor targeted for Razor begins with an initial estimate of the optimal target error rate is made, based on characterization of the costs and benefits of the technique with respect to error rate. This involves estimating the power savings afforded by voltage scaling and the power cost of error recovery and finding the voltage (and corresponding error rate) at which the cost and benefit equalize. Next, the processor is optimized for the selected target error rate using the *OptimizeProcessor* heuristic described in Section III. Finally, we characterize the optimized design and check for good matching between the error rate for which power is minimized and the target error rate for which the design was optimized. We create a feedback loop, and the difference between these error rates drives the optimization flow until a good matching is achieved and power is minimized. Similar methodology can be used for other timing speculation-based techniques such as Intel's EDS [13].

2) Case Study: Application Noise Tolerance: Error-tolerant applications [14] represent an opportunity to save power and increase performance by allowing errors to propagate to the application level rather than expending power to detect and correct them at the hardware level. For several such applications, data errors simply result in reduced output quality, instead of program failure.

Designing a recovery-driven processor for error tolerant applications requires several considerations. First, the set of processor modules is partitioned into two subsets – one containing modules that produce errors that the applications can tolerate, and another containing modules that should not allow errors to propagate to the application level. For the class of error-tolerant applications that we consider in this paper, errors in the arithmetic units (i.e. ALU, FPU) can be tolerated. For this class of applications (which relies heavily on numerical computation), the arithmetic units account for approximately 35% of the dynamic power consumption of the processor.

In addition to the list of modules to optimize, the *OptimizeProcessor* procedure requires a target error rate. The error rate is chosen such that all applications in the class have acceptable quality for the target error rate.

B. Heterogeneously Reliable Multi-core Processors

Different applications have different levels of intrinsic robustness and different activity profiles. Some applications cannot tolerate errors, while others can seamlessly tolerate datapath errors at the expense of output quality [12]. Ideally, a processor core should be matched to the robustness of the application it is running.

Just as single-ISA heterogeneous multi-core processors [10] were proposed to efficiently meet the varying performance needs of different classes of applications, we propose *heterogeneously reliable multi-core processors* (see Figure 3) in which the cores on the processor are designed for different reliability targets using our processor-level design heuristic. For a reliability-diverse workload, a heterogeneously reliable CMP can potentially achieve higher power and energy efficiency than homogeneous CMPs by mapping an application with a specific reliability requirement to an appropriate core on the processor. A homogeneous CMP will waste power or performance by either over-provisioning for error correction when it is unnecessary or under-provisioning when protection is necessary and suffering a performance loss or power increase to ensure reliability.

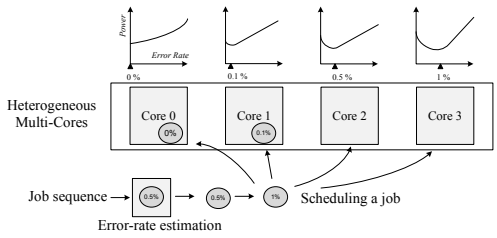

Fig. 3. The heterogeneously reliable CMP matches each application to the core that has minimum power for the application's reliability requirement.

An example heterogeneously reliable dual-core CMP consists of one core designed for hardware error tolerance with Razor, and one core relying on application-level error tolerance that is designed to allow errors in arithmetic units under nominal conditions. Error tolerant applications are mapped to the core designed to allow arithmetic errors, while applications that cannot tolerate errors are mapped to the core designed for Razor-based error tolerance. When the workload only consists of applications that cannot tolerate errors, the frequency of the second core is reduced to prevent timing errors.

Another example heterogeneously reliable multi-core processor may have different cores optimized for applications with different levels or types of software-based error resilience. For instance, one core may be optimized for applications that can tolerate floating point errors, while the other core is optimized for applications that can tolerate errors in the SAD (sum-of-absolute-difference) unit.

V. Experimental Methodology

Our methodology for demonstrating the benefits of exploiting error resilience for single core and multi-core processor design has two parts – a design-level methodology to characterize the power and reliability of circuit modules optimized for different voltage and error rate targets, and an architecture-level methodology to estimate processor power and performance when the proposed design-level techniques are applied at the processor

level. The design-level methodology is described in [8], and the architecture-level methodology is described in [6].

VI. Results and Analysis

A. Minimizing Power for a Target Error Rate

To demonstrate the benefits of minimizing power for a target error rate, we run experiments for five implementation cases at an operating frequency of 0.8GHz (the highest frequency at which no module produces timing errors) – Traditional P&R with a loose clock frequency target (0.7GHz), Traditional P&R with a tight clock frequency target (1.4GHz), BlueShift PCT [3], Slack optimizer [7], [6], and Power optimizer (i.e., designing for a target error rate) Note that the Power Optimizer produces a different design for every error rate target, while other implementations simply produce one design each that is evaluated at different error rates.

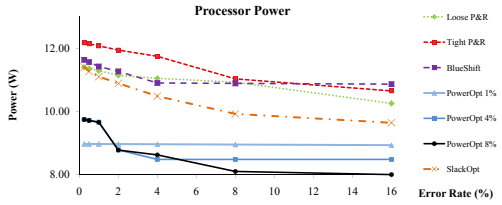

Fig. 4. Processors optimized for different error rate targets consume the least power out of all alternative designs, especially around the error rate target for which they were optimized.

Figure 4 compares the power consumption of processors optimized for different error rate targets against alternative design approaches. The results confirm that the processors designed for target rates of 1, 4, and 8% indeed have the lowest power consumption at those error rates. Note that *Power Optimizer* significantly outperforms BlueShift and Slack Optimizer at all error rates. This is not surprising, considering that BlueShift and Slack Optimizer target different objectives, and therefore result in processors that are overdesigned for specific target error rates.

Notice also that in many cases the power optimizer produces a design that has lower power even for an error rate of 0%. Difference in power consumption at 0% error rate between conventional SPR and PowerOpt/SlackOpt is due to the fact that conventional SPR does not use functional information, and therefore, optimizes even false paths and dormant paths at the expense of area/power. SlackOpt has a high power overhead relative to PowerOpt for 0% as it optimizes for a range of error rates, which requires SlackOpt to upsize on more paths, resulting in higher area, and consequently, power overhead.

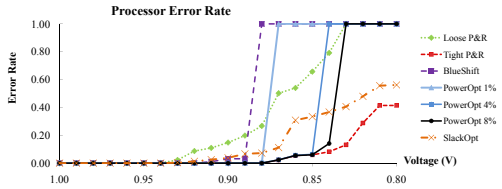

Fig. 5. The voltage at which a design reaches its target error rate determines how much power can be reduced through voltage scaling.

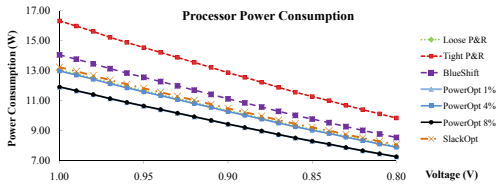

Fig. 6. The power of a processor design at a given voltage shows the relative ordering of designs in terms of power overhead.

Figures 5 and 6 further explain why designs optimized for error rate targets are able to achieve lower power than alternative approaches. Figure 5 shows that designing for a target error rate minimizes the voltage at which that target error rate occurs. Thus, as the error rate target is increased, correctness is relaxed to a greater extent, allowing more cell downsizing, less restriction on voltage scaling due to resulting faulty paths, and a correspondingly lower voltage for the target error rate, which translates into lower power consumption. Note that tightly constrained SP&R produces a design with an error rate even lower than the power optimizer for a given voltage. However, as Figure 6 demonstrates, the power overhead of this approach is substantially higher than that of the power optimizer due to additional area overhead that offsets the benefits of additional voltage scaling, allowing Power Optimizer to emerge as the most efficient power reduction technique for a given target error rate.

B. Recovery-driven Processors

In this section, we demonstrate the benefits of designing processors for specific hardware and software-based error tolerance mechanisms.

1) Circuit-level Timing Speculation: Figure 7 compares the power consumption of processors designed to produce errors that are tolerable by Razor against the power consumption of processors designed for other objectives, such as gradual slack or BlueShift, and against processors that have been designed for correctness but use the traditional Razor methodology to save power.

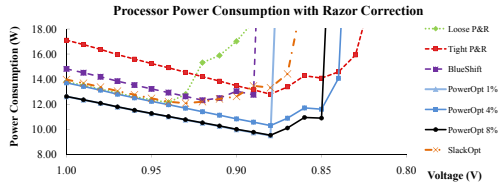

Fig. 7. The benefit of designing a processor to produce errors then correcting them with an error tolerance mechanism over designing for correctness and then relaxing the correctness guarantee can be significant. Results are shown for processors with Razor.

Figure 7 demonstrates that the minimum processor power is indeed achieved for processors that are designed to produce errors that can be gainfully tolerated by Razor. Designing the processor for the error rate target at which Razor operates most efficiently allowed us to extend the range of voltage scaling from 0.94V for the best "designed for correct operation" processor to 0.88V for the processor designed for an error rate of 1%, affording an additional 22% power savings.

2) Application Noise Tolerance: To demonstrate the benefits of recovery-driven design targeted at application-level noise tolerance, we use a face detection algorithm [14] as the target application. Face detection is naturally robust to errors in several arithmetic processor modules and does not require strict computational correctness. Rather than causing program failure, errors may result in reduced output quality (false positive or negative detections) [12].

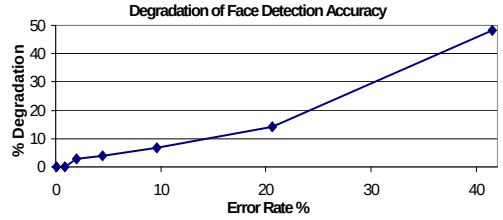

Fig. 8. The figure above shows how the detection accuracy for the face detection application degrades as the error rate of the processor is increased. Even at some non-zero error rates, maximum output quality can be achieved.

Figure 8 shows how detection accuracy degrades as the error rate of the processor increases. Notice that due to the robustness of the face detection algorithm, maximum output quality is observed

even for non-zero error rates of up to 1% and remains within 10% of the maximum value for error rates of up to 15%.

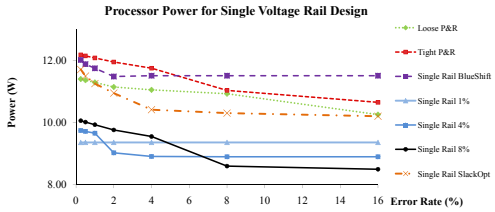

Fig. 9. This figure demonstrates the power benefit of a processor that is designed to allow errors in the arithmetic units over a processor that is designed for correctness. All modules in the processor operate at the same voltage. Razor is used to correct errors in non-arithmetic units.

Figure 9 compares the power consumption of processors designed for software-based error-resilience. In the figure, all processors achieve the same output quality at a given error rate, but processors designed to allow errors consume less power, and power is minimized for these designs at their respective error rate targets. For example, at an error rate of 1%, where output quality is still maximized for the face detection application, the processor designed for an error rate target of 1% consumes 25% less power than the best processor designed for correctness. Power benefits over the next best alternative are 17%. Benefits are even higher for larger error rates if some output degradation is permissible.

C. Heterogeneously Reliable Multi-core Processors

To demonstrate the power benefits of heterogeneously reliable multi-core processors over homogeneous multi-core processors, we considered a heterogeneously reliable dual-core CMP where one core is designed for hardware-based error tolerance with Razor, and the second core relies on application-level error tolerance and is designed to allow errors in arithmetic units under nominal conditions. We consider three homogeneous configurations – one with baseline cores designed for correctness, one with cores that are optimized for Razor-based correction, and one with cores that allow errors in arithmetic units and rely on software error tolerance. We also consider three types of workloads – one includes only applications that do not tolerate errors (SPEC), one includes only applications that tolerate errors (face detection, CG, FIR, least squares), and one includes a mixture of error-tolerant and error-intolerant applications. The Razor core and the application noise-tolerant core in have been designed for an error rate of 1%. The operating frequency for all designs is 0.8GHz as above.

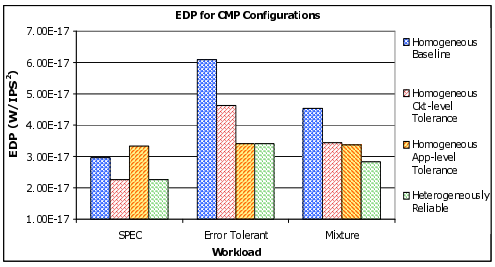

Fig. 10. The heterogeneously reliable CMP is able to adapt to the needs of the applications to provide power efficiency for a diverse set of applications.

Figure 10 compares the energy delay product (EDP) for various homogeneous CMP configurations against that of a heterogeneously reliable CMP for different workload types. The homogeneous CMP that relies on applications to tolerate errors in the arithmetic units performs poorly for workloads with applications that have no error tolerance, since the frequency on the cores must be scaled down considerably to guarantee that no paths in the processor suffer from timing errors. In this case, the heterogeneously reliable CMP has 32% lower EDP. For workloads with exclusively error tolerant applications, the homogeneous CMPs that guarantee correctness (baseline and Razor-based error tolerance) suffer, since these configurations have over-designed arithmetic units. I.e., these configurations use additional power and area (due to guardbanding and Razor overhead respectively) to ensure that no errors occur in these units, even though the error-tolerant applications can gainfully tolerate errors in these units. These overheads result in 44% lower EDP for the heterogeneously reliable CMP with respect to the baseline CMP and 26% lower EDP with respect to the CMP with Razor-based error tolerance. For the workload with a mixture of applications with different reliability requirements, each homogeneous CMP suffers from the sub-optimality described above, and the heterogeneously reliable CMP stands out uniquely as the most efficient design point, achieving EDP benefits of 37%, 17%, and 16% over the baseline CMP, and CMPs with Razor-based and software-based error tolerance, respectively.

Another example heterogeneously reliable multi-core processor that exploits diversity in the error resilience of applications consists of cores customized for different reliability targets. Applications are mapped to cores based on the error rates they can gainfully tolerate. Figure 9 demonstrates the power benefits of matching the reliability requirement of a task to the reliability design target of a core. Additional power savings become available as the tolerable error rate increases. Note that portions of a core may still be protected using a hardware-based error resilience mechanism.

VII. Conclusion

This paper presents a novel approach to reducing processor power by designing a processor from the ground up to deliberately allow voltage overscaling-based timing errors([4],[2]) during nominal operation, while using an error recovery technique to tolerate these errors. In this *error rate-optimal* approach to reducing processor power, the processor is optimized for a target timing error rate instead of correct operation during nominal conditions.

We showed that optimizing power for a target error rate allows significantly lower operating voltages for the same frequency of operation, resulting in significant processor power savings for similar levels of performance. Power improvements were up to 39% for an error rate target of 0.5% over the traditional design methodology. We also showed that *recovery-driven processors* and *heterogeneously reliable multi-core processors* have substantial power and EDP benefits over their conventional counterparts. Benefits were up to 29% and 32%, respectively.

As the need for low-power processing increases and as applications show increased diversity in error tolerance, the benefits of the proposed design philosophy will continue to increase.

References

[1] T. Austin, V. Bertacco, D. Blaauw and T. Mudge, "Opportunities and Challenges for Better Than Worst-Case Design", *Proc. Asia South Pacific Design Automation*, 2005, pp. 2–7.

[2] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner and T. Mudge, "Razor: A Lowpower Pipeline Based on Circuit-Level Timing Speculation", *IEEE/ACM Proc. International Symposium on Microarchitecture*, 2003, pp. 7–18.

[3] B. Greskamp, L. Wan, W. R. Karpuzcu, J. J. Cook, J. Torrellas, D. Chen and C. Zilles, "Blueshift: Designing Processors for Timing Speculation from the Ground Up", *International Symposium on High Performance Computer Architecture*, 2009, pp. 213–224.

[4] R. Hegde and N. R. Shanbhag, "Energy-Efficient Signal Processing via Algorithmic Noise-Tolerance", *Proc. International Symposium on Low Power Electronics and Design*, 1999, pp. 30–35.

[5] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose and M. Martonosi, "An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget", *Proc. IEEE/ACM International Symposium on Microarchitecture*, 2006, pp. 347–358

[6] A. B. Kahng, S. Kang, R. Kumar and J. Sartori, "Designing a Processor From the Ground Up to Allow Voltage/Reliability Tradeoffs", *IEEE International Symposimum on High-Performance Computer Architecture*, 2010, pp. 119–129

[7] A. B. Kahng, S. Kang, R. Kumar and J. Sartori, "Slack Redistribution for Graceful Degradation Under Voltage Overscaling", *Proc. Asia and South Pacific Design Automation Conf.*, 2010, pp. 825–831

[8] A. B. Kahng, S. Kang, R. Kumar and J. Sartori, "Recovery-driven Design: A Methodology for Power Minimization for Error Tolerant Processor Modules", *Proc. Design Automation Conf.*, 2010.

[9] N.R. Shanbhag, R. Abdallah, R. Kumar and D. Jones, "Stochastic Computation", *Proc. Design Automation Conf.*, 2010.

[10] R. Kumar, K.I. Farkas, N.P. Jouppi, P. Ranganathan and D.M. Tullsen, "Single-ISA Heterogeneous Multi-core Architectures: The Potential for Processor Power Reduction", *IEEE/ACM International Symposium on Microarchitecture*, 2003, pp. 81

[11] J. Sartori and R. Kumar, "Three Scalable Approaches to Improving Many-core Throughput for a Given Peak Power Budget", *International Conference on High Performance Computing*, 2009.

[12] J. Satori, J. Sloan and R. Kumar, "Fluid NMR - performing power/reliability tradeoffs for applications with error tolerance", *Workshop on Power Aware Computing and Systems*, 2009.

[13] J. W. Tschanz, K. Bowman, S-L. Lu, P. Aseron, M. Khellah, A. Raychowdhury, B. Geuskens, C. Tokunaga, C. Wilkerson and T. Karnik, V. De, "A 45nm Resilient and Adaptive Microprocessor Core for Dynamic Variation Tolerance", *International Solid-State Circuits Conference*, 2010.

[14] P. Viola and M. J. Jones, "Robust real-time face detection", *International Journal of Computer Vision*, 52(2), 2004, pp. 137–154.

[15] T. Yamada, S. Kataoka and K. Watanabe, "Heuristic and Exact Algorithms for the Disjunctively Constrained Knapsack Problem", *Information Processing Society of Japan Journal*, 43(9), 2002, pp. 2864–2870.

[16] *International Technology Roadmap for Semiconductors 2008 Update.*, http://www.itrs.net/.