Optimal Microarchitectural Design Configuration Selection for Processor Hard-Error Reliability

Ying Zhang[1], Lide Duan[1], Bin Li[2], Lu Peng[1]
[1]Department of Electrical and Computer Engineering
[2]Department of Experimental Statistics
Louisiana State University
Baton Rouge, LA, 70803
{yzhan29, lduan1, bli, lpeng}@lsu.edu

Abstract

Traditional design space exploration mainly focuses on performance and power consumption. However, as one of the first-class constraints for modern processor design, the relationship between hard-error reliability and processor configurations has not been well studied. In this paper, we investigate this relationship by exploring a large processor design space. We employ a rule search strategy, i.e. Patient Rule Induction Method, to generate a set of rules which choose optimal configurations for processor hard-error reliability and its tradeoff with performance and power consumption.

Keywords

Design space exploration, reliability, statistical model

1. Introduction

The increasing number of transistors with deep submicron size via modern fabrication technique enables computer architects to integrate complicated components on a single chip. In order to select optimal microarchitectural configurations from a vast design space, processor manufacturers need to know the features of each available design choice through cycle-accurate simulation, which usually makes an exhaustive exploration unfeasible due to the expensive time cost. Reducing the exploration time via predictive modeling has been widely explored in the past few years [8][9][14]. Those proposed approaches are able to significantly speed up the design space exploration, while achieving high accuracy in predicting specific metrics on unimplemented configurations.

As modern microprocessors continue scaling to smaller feature sizes, hard-error reliability has become a first-class constraint for high-performance processor design. Enduring high power density and temperature largely degrade the hard-error reliability of a processor. For instance, Negative Bias Temperature Instability (NBTI), whose occurrence is due to the continuous increase in processor threshold voltage, tends to be a key reliability issue when the manufactory technology reaches 90nm. The situation is further exacerbated when the on-die temperature increases. Although several techniques including power balancing and hotspot elimination have been proposed to improve the lifetime reliability for a given processor [3]; however, designing hard-error resilient processors at early design stage remains an open topic. Since processors with different microarchitectural configurations are prone to show distinct behaviors such as various power dissipations, we believe that microarchitectural configurations heavily impact the hard-error reliability of a processor. This indicates that selecting the most reliable configurations from a vast design space is of great importance in current industry.

In this paper, we investigate the relationship between hard-error reliability of a processor and the underlying microarchitectural configurations. We use the RAMP model [20] to track the reliability information. RAMP employs FIT (failures in 10^9 hours) as an agent to interpret a processor's life-time stability, where a smaller FIT value indicates a longer MTTF (mean time to failure). It simulates five important failure mechanisms [4][16][24] including electromigration (EM), stress migration (SM), thermal cycling (TC), time dependent dielectric breakdown (TDDB) and the aforementioned NBTI. Given the power consumption, temperature and frequency, RAMP is capable of automatically calculating the FIT of a processor. To correlate configuration parameters to the design objective, we employ an advanced statistical technique called Patient Rule Induction Method (PRIM) to facilitate our work. The PRIM model can extract a few simple "rules" or conditions by which the processor satisfies a preset design goal for a certain response. Specifically in our work, rules for improving hard-error reliability and its tradeoff with performance and power are collected from a group of representative benchmarks. Our validation results show that the proposed method can generate rules to effectively identify optimal processor design configurations for individual goals and their tradeoffs. In summary, the major contributions of this paper are as follows:

- **Optimal Design Configurations for Hard-Error Reliability**: we demonstrate that different design choices will result in distinct reliability behaviors. Based on the investigation, we identify the most reliable microarchitectural configurations.
- **Optimal Configurations for Other Metrics**: We show that different metrics including performance, power, temperature, and hard-error reliability usually favor different sets of design configurations. Therefore, we also investigate optimal configurations for other metrics.
- **Balancing Reliability, Power, and Performance**: We generate rules to filter out promising configurations that can yield optimal tradeoffs among multiple metrics.

The remainder of the paper is organized as follows: In section 2, we review the related work. Section 3 introduces

the statistical model engaged in this work and explicates the procedures of the proposed method. The experimental setup is demonstrated in section 4. We discuss the extracted rules along with their validation in section 5 and draw our conclusions in section 6.

2. Related Work

The hard-error defects that degrade processor reliability [4][16][24] are first noticed in the circuit design process. Srinivasan et al. propose a reliability-aware microprocessor (RAMP) model [20], which simulates five important failure mechanisms and aims at overcoming the unnecessary high cost in manufacturing an overestimated processor, as hard-error reliability is always tested under the worst conditions in the current industry. The authors further extend their study in [21] and demonstrate the impact of fabrication technology on hard-error reliability. Architecture wise, Bower et al. [1] introduce an online diagnosis technique to detect the hard faults. Tiwari et al. [22] provide a solution to mitigate Hot Carriers Injection (HCI) and NBTI problems. In [19], duplicated architectural components are used to hide the hard errors and restrict the performance degradation at an acceptable level. On the other hand, since the processor hard-error reliability is highly related to its runtime power and temperature, many techniques are proposed to improve the processor reliability by removing the on-die hotspot at runtime. Coskun et al. [3] discuss the impact of different job scheduling algorithms on processor lifetime. They demonstrate that the processor lifetime can be effectively prolonged when smart temperature-aware scheduling mechanisms are engaged. Wang et al. [23], however, demonstrates that simply balancing power consumption and decreasing maximal on-chip temperature is not sufficient to significantly improve the processor lifetime reliability. Instead, they propose that a processor core should be set to different frequency according to its power consumption. By doing this, the processor reliability can be greatly improved.

Substantial works have been done on design space exploration. Ozisikyilmaz [14] and Ipek [8] present design space exploration strategies using machine learning techniques to predict processor performance. In [10], Lee et al. consider power consumption as an additional target in prediction. The impact of pipeline depth, superscalar width and L2 cache size on temperature are investigated in [11][12]. Mochiero et al. [13] further study the impact of processor configuration on performance, power consumption and temperature, while identifying the important factors to leakage power and overall temperature distribution. Duan et al. [5] used the same statistical tool, PRIM, to extract the most resilient processor configurations for soft errors.

Our work deviates from the above studies in that we investigate the relationship between microarchitectural configurations and hard-error reliability and extract the promising configurations at early design stages. The configurations achieving the best tradeoff among performance, power and reliability are also identified.

3. Methodology

3.1. Patient Rule Induction Method (PRIM)

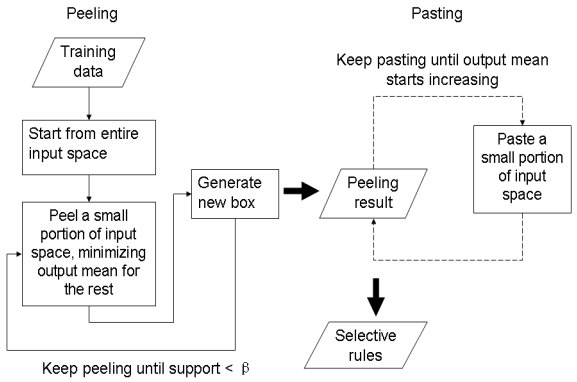


Figure 1. PRIM training procedure, including peeling and pasting

PRIM is an advanced statistical model [7] whose objective is to find a region in the input space (composed of configuration parameters in this work) that gives relatively low values for the output response, e.g. the FIT value. The selected region (or "box") is described in an interpretable form involving a set of "rules" depicted as $B = \bigcap_{j=1}^{p} \left(x_j \in s_j \right)$, where x_j represents the jth input variable and s_j is a subset of all possible values of the jth variable.

Figure 1 illustrates the construction of the "optimal" region, which is composed of two phases: (1) patient successive top-down peeling process; (2) bottom-up recursive pasting process. The top-down peeling starts from the entire space (box B) that covers all the data. At each iteration, a small subbox b within the current box B is removed, which yields the smallest output mean value in the result box $B - b$. We perform this operation iteratively and stop when the support of the current box B is below a chosen threshold β, which is actually the proportion of the points remaining in the selected area at the end of peeling.

The pasting algorithm works inversely from the peeling results and the final box can sometimes be improved by readjusting its boundaries. Starting with the peeling solution, the current box B is iteratively enlarged by pasting onto it a small subbox that minimizes the output mean in the new (larger) box. The bottom-up pasting is iteratively applied, successively enlarging the current box, until the addition of the next subbox causes the output mean to begin increasing.

An advantage of PRIM over greedy methods such as tree-based methods is its patience. For example, a binary tree rapidly fragments the data because of the binary splits in that tree, while the PRIM model only peels off a small proportion of data every time. Hence, the solution of PRIM (hyper-boxes) is usually much more stable than those obtained from tree models. In cases where the optimal subspace is not connected, PRIM can generate a sequence of hyper-boxes instead of only one. Namely, after getting the first hyper-box, the PRIM procedure is repeated on the remaining dataset. As a result, the disconnected subspace can also be covered. In this work, we found that the leading box often covers most of the points with the small response values. Finally, the threshold β indicates the percentage of data

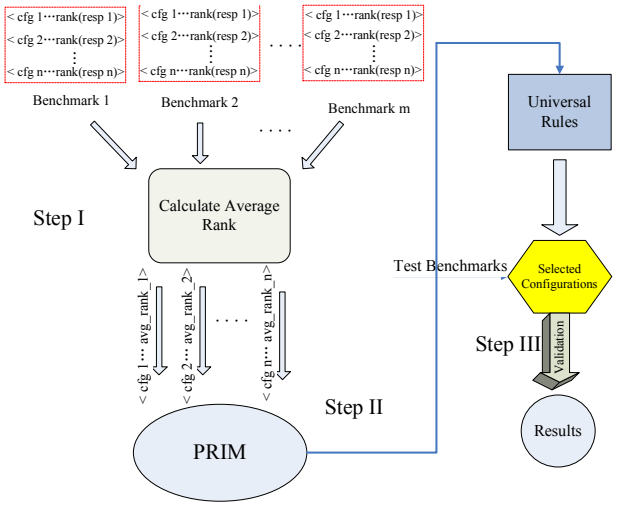


Figure 2. Procedure of the generation and validation of universal rules for a specific metric

points which will remain in the final hyper-box. We set β to 0.02 in this work. As can be seen from later sections, PRIM is very effective in identifying optimal points. The extracted 2% design space points are usually within top 8% - 12% optima of the entire design space. Note that the identified points are not necessarily the top 2% optima due to two reasons: (1) the optimal subspace may not be connected, and we only extract the leading one box in this work; (2) our model is evaluated across different programs which could demonstrate different behaviors for the metrics in analysis.

3.2. Overview of the Proposed Method

As shown in Figure 2, our modeling work consists of three major steps: First, a group of representative benchmarks (benchmark *1* through benchmark *m*) are selected. For each of them, we simulate *n* configurations (*cfg.1* through *cfg.n*) randomly and uniformly sampled from the entire design space. *Res.1* through *Res.n* are the measured responses corresponding to the sampled *n* configurations. Note that the absolute value ranges of the responses for one benchmark may be different from those of another benchmark. Therefore, it is not reasonable to directly use the configurations and their response values collected from different benchmarks to train a universal model. Taking this into consideration, we rank the configurations in each benchmark (For the *n* configurations in Figure 2, the one with the smallest output values, e.g., the shortest execution time or the lowest FIT, is ranked No. *1* while the one with largest output is ranked No. *n*). We then calculate the average rank for each configuration and use it in the model training. Second, we train PRIM models for optimizing different metrics. By doing so, we generate a set of rules which filter out a design subspace within which each configuration has the optimal output value (performance, power, temperature, or FIT) for different benchmarks. Third, we validate the effectiveness of the universal rules by applying them to benchmarks not used in the training set.

4. Experimental Setup

In this section, we present our experimental methodology. We integrate Wattch [2], Hotspot [18], and RAMP [20], for power, temperature and reliability (measured in FIT) computation respectively, into a cycle-accurate simulator SESC [15], and set the floorplan based on an Alpha 21264 processor. 2000 configurations randomly sampled from the design space are used in training the PRIM models. We choose 12 benchmarks covering all the 8 clusters [17] in SPEC CPU2000 Suite (*ammp*, *art*, *apsi*, *applu*, *crafty*, *gap*, *gcc*, *gzip*, *mcf*, *parser*, *twolf*, *wupwise*) as the training set and the other 7 benchmarks (*bzip2*, *equake*, *mesa*, *mgrid*, *swim*, *vortex*, *vpr*) whose binaries are provided by the website of SESC [15] for validation.

Table 1. Processor design space in our study

Parameters	Selected Values
Fetch/ Issue/Commit Width	2, 4, 6, 8
#ALU/FPU (dependent on width)	1/1, 2/1,associated with 2
	2/1, 4/2, associated with 4
	3/1, 6/3, associated with 6
	4/2, 8/4, associated with 8
Instruction Queue Size (IQ)	24, 32, 40, 48, 56, 64
L1D Cache Size	8, 16, 32, 64KB
L1D Cache Associativity (L1DA)	1, 2, 4
Reorder Buffer Size (ROB)	64 – 152, with a step of 8
Register File (RF) (dependent on ROB size)	48 – 144, with a step of 8
	RF = ROB Size – 8 (or 16)
Load Store Queue Size (LSQ)	16 – 64, with a step of 8
Load/Store Unit	1/1, 2/2
Branch Target Buffer Size (BTB)	1024, 2048
Max Branches	8, 16 ,32
Fixed Parameters	**Value**
L1 Instruction Cache Size	32KB
L1 Instruction Cache Associativity	4
L1 Block Size	64B
L2 Cache Size	2MB
L2 Cache Associativity	4
L2 Block Size	64B
Technology	65nm
Frequency	2GHz

Table 1 lists the parameters included in our design space. We choose design parameters for better investigation of hard-error reliability. Given the manufacture technique, the hard-error reliability of a processor varies with the dynamic power density and temperature on its functional blocks. Therefore in this work, we carefully study the impact of different components, each of which is changed within a wide range. Previous studies [12][13] show that an L2 cache is relatively cool compared to other components because it dominates more than half of the chip area and is thereby not likely to be an overheated and vulnerable part. For this reason, we fix the L2 cache parameters in this study. In total, our exploration space contains 1,161,216 points.

Modern processor manufacturers show that the melioration on one design goal is always implemented at the expense of degrading other metrics. Taking this into account, processor hard-error reliability is not the exclusive target in our study; we also aim at finding out configurations delivering the best tradeoff among multiple metrics. Therefore, we conduct a series of experiments to seek solutions that can yield the optimal balance among performance, power, and hard-error reliability.

5. Results Analysis

5.1. Optimizing hard-Error reliability

Table 2.Individual and universal rules for optimizing reliability

App.	Generated Rules	App.	Generated Rules
art	Width/ALU>=8/4/2 **&&** ROB/RF<=128/112	*gap*	Width/ALU>=8/4/2 **&&** L1DA>1
apsi	Width/ALU>=6/6/3 **&&** L1D>16KB **&&** ROB/RF <=136/120	*gzip*	Width/ALU>=8/4/2 **&&** L1D>32KB **&&** ROB/RF <=136/120
applu	Width/ALU>=8/4/2 **&&** ROB/RF <=136/120 **&&** LSQ<56	*gcc*	Width/ALU>=8/4/2 **&&** L1DA>1 **&&** ROB/RF <=144/128
crafty	Width/ALU>=8/4/2 **&&** ROB/RF<=128/112	*wupwise*	Width/ALU>=6/3/1 **&&** ROB/RF<=88/80
ammp	Width/ALU>=6/3/1 **&&** L1D>16KB **&&** L1DA>1	*twolf*	Width/ALU>=8/4/2 **&&** L1DA>1 **&&** ROB/RF<=128/120
mcf	Width/ALU>=6/6/3 **&&** L1D>8KB **&&** L1DA>1 **&&** ROB/RF<=128/112	*parser*	Width/ALU>=6/3/1 **&&** L1D>8KB **&&** L1DA>1 **&&** ROB/RF>=72/56
Universal rules	(Width/ALU>=8/4/2) **&&** (L1D>=32k) **&&** (L1DA>=2) **&&** (72/56<=ROB/RF<=104/96) **&&** (LSQ<=48)		

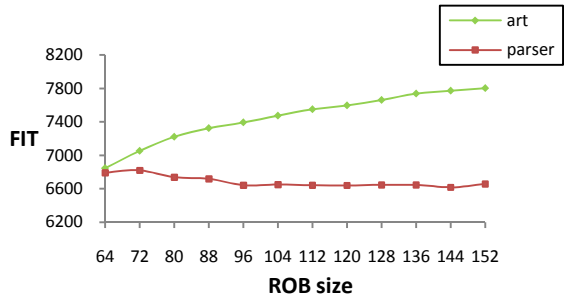


Figure 3. Variation of FIT with ROB size changing

In order to gain insights into the impact of microarchitectural configurations on processor hard-error reliability, we first generate a set of "individual" rules for each training benchmark by feeding the configurations and their corresponding responses into PRIM. Note that the absolute FIT values collected from a benchmark are used to train an individual model for this metric. Table 2 lists the rules for optimizing hard-error reliability for the 12 training benchmarks. The universal rule set which is summarized by following the steps described in section 3.2 is shown at the bottom. For example, the rule set of *art* suggests that a processor should be configured with fetch/issue/commit width of 8 and the number of ALU/FPU to be either 4/2 or 8/4, while the ROB/RF size not exceeding 128/112.

We observe that 11 out of 12 benchmarks do not select larger size ROBs, with an exception of *parser* that removes the extremely small ROBs from its optimal space. This implies that ROBs with intermediate sizes are effective to improve the reliability. To further understand this issue, we perform a sensitivity study to demonstrate the variation of FIT with changing of the ROB size while other parameters are fixed. This is illustrated in Figure 3. As can be seen, for *art*, the FIT is rising with the ROB size being increased (we also observe this from other benchmarks), while *parser* shows a reverse trend. Based on our observations, running *parser* with a small-size ROB tends to result in a relatively high occupancy on the reorder buffer. This can lead to a high power density on that component and consequently degrading the processor's lifetime reliability. For other benchmarks such as *art*, a large size reorder buffer, which is usually integrated in a modern out-of-order superscalar processor to fully explore the instruction-level parallelism, will consume substantial power and gradually create hot spots on die. This significantly increases the probability of failures. Therefore, the intermediate size ROBs are the most stable design. Other components such as branch predictor are not critical parameters for hard-error reliability and do not appear in the generated rules.

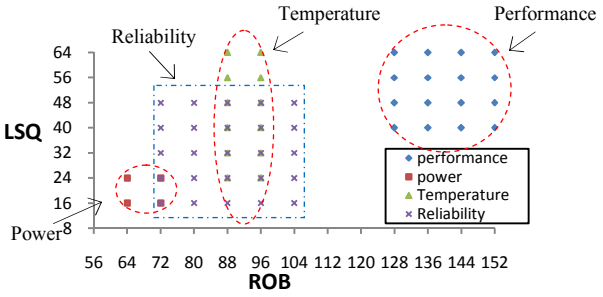
Table 3. Universal rules for optimizing other metrics

Metric	Rules
Execution time	(width/ALU>=6/6/3) **&&** (IQ >=32) **&&** (L1D>=32k) **&&** (L1DA>=2) **&&** (ROB/RF>=128/120) **&&** (LSQ>=40) **&&** (LSUnit = 2/2)
Power	(L1D>=16k) **&&** (ROB/RF=64/48 or 64/56 or 72/56) **&&** (LSQ<=24)
Temperature	(width/ALU=6/3/1) **&&** (32<=IQ<=56) **&&** (L1D>=32k) **&&** (L1DA>=2) **&&** (ROB/RF=88/80 or 96/80) **&&** (LSQ>=24)
Perf^0.5 * Rel^0.25 * Pow^0.25 (Tradeoff 1)	(width/ALU>=8/4/2) **&&** (L1D>=32k) **&&** (L1DA>=2) **&&** (72/64<=ROB/RF<=104/96) **&&** (24<=LSQ<=48)
Pow^0.5 * Rel^0.25 * Perf^0.25 (Tradeoff 2)	(width/ALU>=8/4/2) **&&** (L1D>=32k) **&&** (L1DA>=2) **&&** (64/56<=ROB/RF<=88/80) **&&** (LSQ<=48)
Rel^0.5 * Pow^0.25 * Perf^0.25 (Tradeoff 3)	(width/ALU>=8/4/2) **&&** (L1D>=32k) **&&** (L1DA=2) **&&** (72/56<=ROB/RF<=96/88) **&&** (LSQ<=48)

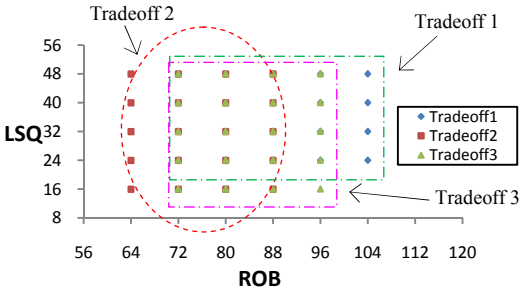
5.2. Optimizing other metrics

Table 3 lists the universal rules generated when performance, power consumption, temperature and the tradeoffs are optimized respectively.

The largest values of ROB/RF, cache and LSQ are selected if performance is the primary design goal. This is straightforward to understand. More resources will be available when a processor is configured in this manner, so the parallelism and overall performance is improved accordingly. When power saving becomes the most important constraint, smaller size ROB/RFs and LSQs are selected since the power reduction on these blocks significantly reduces the total core power consumption. However, L1 data cache demonstrates an opposite trend in which larger size caches are favored. This is because a cache miss consumes more power compared to a cache hit as cache misses generate substantial bus transactions and accesses to low-level memory. Therefore, larger L1 data caches are selected in the rule set for optimizing power because of the decreased number of cache misses. For the temperature constraint, in order to prevent a processor from being overheated, medium size ROB/RFs are selected to cool down the chip. This is due to a similar reason given in the previous subsection; high pow-



(a) Individual metrics



(b) Different tradeoffs

Figure 4. ROB & LSQ selections for individual metrics and different tradeoffs

er density in small-size ROB/RFs and high global power resulting from large-size structures both have the potential to raise the temperature on die. Figure 4(a) visualizes the optimal subspace for these metrics.

The tradeoff among performance, power dissipation and reliability is becoming increasingly attractive. Temperature is not included here because FIT is highly related to chip temperature and is able to reflect its variation. In this work, we utilize the following measurement for the tradeoff investigation, where a smaller value implies a better tradeoff.

(execution cycles^a) * (power^b) * (FIT^c), where a+b+c=1

In case of one variable dominating the above equation, one can further normalize each metric. However, this is not necessary in this work since our PRIM model is trained with the ranks of data points instead of real values (Section 3.2).

In the first study, we focus on the tradeoff by which high performance is emphasized. For simplicity, we fix a to 0.5 and set the values of both b and c to 0.25. As illustrated in Figure 4(b), larger ROBs and LSQs are suggested in this circumstance. However, these choices differ from those selected when performance is the exclusive design goal. In particular, moderate sized ROBs and LSQs are preferred by this tradeoff instead of extremely aggressive configurations.

In the second case, b is fixed to 0.5 while a and c both taking the values of 0.25. By doing this, we try to extract the optimal configurations when low power is more important. Similarly, a, b and c are respectively set to 0.25, 0.25 and 0.5 in the third study, which is conducted when high reliability is preferred. Figure 4(b) clearly demonstrates that the optimal subspace for the second and the third tradeoffs contain more conservative configurations compared to those for the first one. Again, this is because larger components incline to be more power-hungry and become vulnerable blocks due to their high temperature.

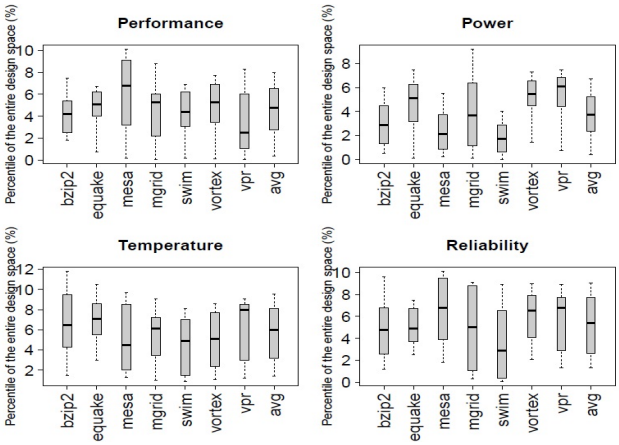


Figure 5. Validation results for rules of individual metrics

5.3. Rules validation

In this subsection, we respectively validate the universal rule sets for optimizing the four individual metrics and for the tradeoffs by running all the left 7 benchmarks provided by [15].

For each benchmark in the test set, we shall identify where the selected points are actually located in the entire design space (not just the 2000 points used for training). In other words, we intend to know what percentile (say p) of the design space that the values of these selected points are below. The p-percentile for the whole space indicates the value that is greater than p% of all the design points but less than the rest. In order to have an accurate estimate, we use the bootstrapping method [6]. Specifically, we first sample (with replacement) 1000 bootstrap samples for the 2000 configurations. Note that each bootstrap sample also contains 2000 design points. We then compute a confidence interval estimate of the p-percentile of the whole design space based on these samples. Specifically, for each bootstrap sample, we calculate its p-percentile. This gives us a total of 1000 values for p-percentiles (one for each bootstrap sample). Among these 1000 values, we further calculate their 20-percentile (say P). By doing so, we have 80% confidence that the p-percentile of the entire design space is larger than or equal to P. Finally, we adjust the p value (by repeating the above steps) to have the derived p-percentile slightly larger than the largest value of the selected points. Therefore, the final determined p value is the percentile that all the selected points are below. Note that this approach is conservative since the exact p-percentile of the entire design space could be much larger than P.

We use boxplot to demonstrate our validation results. In a boxplot, the lower and upper boundaries of the central gray box correspond to the 25% and 75% quantiles; the bold line within the box is at the median; the vertical dotted line drawn from the boundaries extend to the minimum and maximum. For instance, for *equake* in reliability validation in Figure 5 (right bottom), the maximum of points selected by the universal rules corresponds to a value of 7.3% in the vertical axis, meaning that for this benchmark all selected points are within the top 7.3% optima of the entire design space. As Figure 5 shows, the design points filtered by the

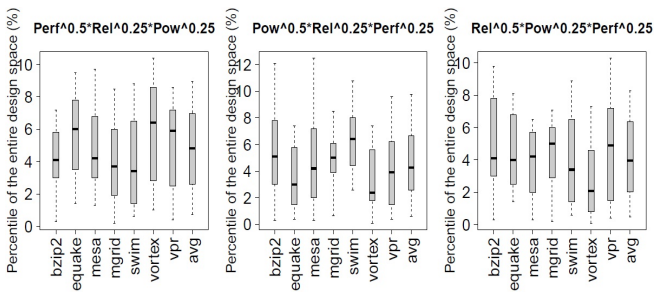


Figure 6. Validation results for optimal tradeoff rules

universal rules remain within the top 8% - 12% optima for the four individual metrics on average. For the three tradeoffs, Figure 6 illustrates that the approximately top 10% optima are extracted by the corresponding rule sets.

6. Conclusion

In this paper, we use an advanced statistical technique to bridge the gap between processor hard-error reliability and its microarchitectural configuration. We find that performance, power consumption, temperature and reliability favor different configurations. We also investigate the optimal balance among individual metrics. The evaluation results demonstrate that our strategy is effective for generating rules which can extract optimal configurations for processor hard-error reliability and different tradeoffs at early design stages.

Acknowledgement

This work is supported in part by an NSF grant CCF-1017961, the Louisiana Board of Regents grant NASA / LEQSF (2005-2010)-LaSPACE and NASA grant number NNG05GH22H, NASA(2011)-DART-46, LQESF(2011)-PFUND-238 and the Chevron Innovative Research Support (CIRS) Fund. Ying Zhang is holding a Flagship Graduate Fellowship from the LSU graduate school. We acknowledge the computing resources provided by the Louisiana Optical Network Initiative (LONI) HPC team. Finally, we appreciate invaluable comments from anonymous reviewers which help us finalize the paper.

References

[1] F. A. Bower, D. J. Sorin and S. Ozev. A mechanism for online diagnosis of hard faults in microprocessors. In MICRO, November 2005.

[2] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In ISCA, June 2000.

[3] A. K. Coskun, R. Strong, D. M. Tullsen, and T. S. Rosing. Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors. In SIGMETRICS, June 2009.

[4] A. Dasgupta and R. Karri. Electromigration reliability enhancement via bus activity distribution. In DAC, June 1996.

[5] L. Duan, Y. Zhang, B. Li, and L. Peng. Universal Rules Guided Design Parameter Selection for Soft Error Resilient Processors. In ISPASS, April 2011.

[6] B. Efron and R. Tibshirani. An introduction to the bootstrap. Chapman & Hall/CRC. 1994.

[7] J. Friedman and N. Fisher. Bump hunting in high-dimensional data. In Statistics and Computing, 9, 123-143, 1999.

[8] E. Ipek, S. A. Makee, B.R. de Supinski and R. Caruana. Efficiently predicting architectural design spaces via predictive modeling. In ASPLOS, October 2006.

[9] S. Khan, P. Xekalakis, J. Cavazos, and M. Cintra. Using predictive modeling for cross-program design space exploration in multicore Systems. In PACT, September 2007.

[10] B. C. Lee and D. M. Brooks. Accurate and efficient regression modeling for microarchitectural performance and power prediction. In ASPLOS, October 2006.

[11] Y. Li et al. Impact of thermal constraints on multi-core architectures. In ITHERM, May 2006.

[12] Y. Li et al. CMP design space exploration subject to physical constraints. In HPCA, February 2006.

[13] M. Mochiero, R. Canal, and A. Gonzalez. Design space exploration for multicore architectures: a power/performance/thermal view. In ICS, June 2006.

[14] B. Ozisikyilmaz, G. Memik, and A. Choudhary. Efficient system space exploration using machine learning techniques. In DAC, June 2008.

[15] J. Renau et al. SESC Simulator. http://sesc.sourceforge.net/, 2005.

[16] D. K. Schroder and J. A. Babcock. Negative bias temperature instability: road to cross in deep submicron silicon semiconductor manufacturing. In Journal of applied Physics, vol. 94, pp.1-18, July 2003.

[17] A.Phansalkar, A.Joshi, L.Eeckhout, and L.K.John. Measuring program similarity: experiments with SPEC CPU Benchmark Suites. In ISPASS, March 2005.

[18] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In ISCA, June 2003.

[19] J. Srinivasan, S. A. Adve, P. Bose and J. A. Rivers. Exploiting structural duplication for lifetime reliability enhancement. In ISCA, June 2005.

[20] J. Srinivasan, S. A. Adve, P. Bose and J. A. Rivers. The case for lifetime reliability-aware microprocessors. In ISCA, June 2004.

[21] J. Srinivasan, S. A. Adve, P. Bose and J. A. Rivers. The impact of technology scaling on lifetime reliability. In DSN, June 2004.

[22] A. Tiwari and J. Torrellas. Facelift: hiding and slowing down aging in multicores. In MICRO, November 2008.

[23] S. Wang, and J. Chen. Thermal-aware lifetime reliability in multicore systems. In ISQED, March 2010.

[24] E. Wu et al. Interplay of voltage and temperature acceleration of oxide breakdown for ultra-thin gate oxides. In Solid-state Electronics Journal, 2002.