Thermal-Awareness in a Soft Error Tolerant Architecture

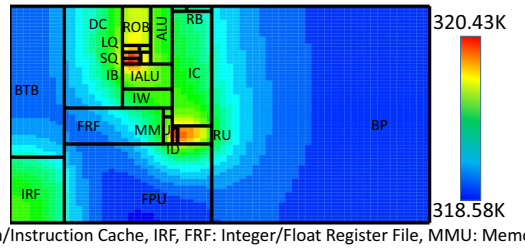
Sajjad Hussain*, Muhammad Shafique†, Jörg Henkel*
*Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany
{sajjad.hussain, henkel}@kit.edu
†Department of Computer Engineering Vienna University of Technology, Austria
{muhammad.shafique}@tuwien.ac.at

***Abstract*—It is crucial to provide soft error reliability in a power-efficient manner such that the maximum chip temperature remains within the safe operating limits. Different execution phases of an application have diverse performance, power, temperature and vulnerability behavior that can be leveraged to fulfill the resiliency requirements within the allowed thermal constraints. We propose a soft error tolerant architecture with fine-grained redundancy for different architectural components, such that their reliable operations can be activated selectively at fine-granularity to maximize the reliability under a given thermal constraint. When compared with state-of-the-art, our temperature-aware fine-grained reliability manager provides up to 30% reliability within the thermal budget.**

I. Introduction and Related Work

Soft errors are the most challenging reliability threat for current miniaturized embedded systems as the critical charge has continuously lowered due to heavy technology scaling over the past few years [1], [2]. Historically, soft error resiliency is provided using the redundant architectures that incur huge power and performance overhead and ultimately lead to an increase in thermal overheads that again retrospectively affect soft errors [3]–[5]. High performance battery-operated embedded systems are usually characterized in terms of performance-per-power due to high cooling cost and power consumption issues. Therefore, these systems need to have minimum possible energy and thermal consumption [6], [7]. Embedded systems are designed around a given thermal power called *thermal design power* (TDP) and the corresponding thermal heat needs to be dissipated within the critical safe temperature T_{safe} [16]. As, shown in the Fig. 1, different processor components have different thermal map during an application run time and Fig. 2 further shows that there are inter-component vulnerability variations for a processor executing a given applications. Therefore, traditional coarse-grained based redundancy (i.e., full-scale duplication or triplication) solutions to soft error resiliency, which are applied for entire execution of an application do not leverage above variations and may be thought as over-protected, for a period of execution where processor is less vulnerable. These variations can be intelligently harnessed at a fine-granularity (i.e., component-level reliability mechanism) to deliver maximum reliability that guarantees correct functionality while power consumption and ultimately the peak temperature is kept under the safe limits.

In Summary, *state-of-the-art has not yet harnessed above-mentioned fine-grained adaptation of different reliability features for different processor components to efficiently maximize the soft error resiliency, and still ensuring that the peak temperature remains within the safe limits.*

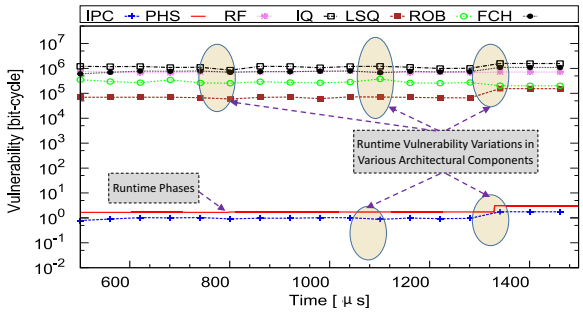


DC, IC: Data/Instruction Cache, IRF, FRF: Integer/Float Register File, MMU: Memory Management Unit, ID, IB, IW: Instruction Decoder/Buffer/Window, RU: Rename Unit, ROB: Reorder Buffer, IALU, FPU: Integer/Float Units, LQ, SQ: Load/Store Queue, BP: Branch Predictor, BTB: Buffer Translation Buffer,

Figure 1: Thermal map of a processor core, using [15], for SUSAN.

A. Concept Overview and Our Novel Contribution

In this paper, we address the above-discussed challenges and introduce a **novel temperature-aware fine-grained reliability manager** (Section IV) that exploits the temporal variations in vulnerability and temperature of different processor component during an execution phase and provide maximum soft error resiliency under a given thermal constraint. This is achieved through a dynamic manager that controls the available reliability features of different components at fine-granularity to intelligently trade-off between vulnerability and temperature during an execution phase. To envisage the above concept of fine-granularity, a comprehensive temperature and vulnerability analysis for different reliability features is performed (Section II).



IPC: Instruction-per-Cycle, PHS: Phase, RF: Register file, IQ: Instruction Queue, LSQ: Load-Store Queue, ROB: Re-order Buffer, FCH: Fetch Unit.

Figure 2: Vulnerability variations in different components for SUSAN.

978-3-9819263-2-3/DATE19/©2019 EDAA

II. Fine-grained Vulnerability Analysis

Fig.3 shows that different applications have distinct variations in their peak temperature, vulnerability and performance characteristics for various reliability mechanisms i.e., dual-modular redundancy- with execution (DMR-RE) and triple-modular redundancy (TMR), whereas Fig.1 and Fig. 2 shows that how temperature and vulnerability is also varied across different components during entire execution. The key observations from the analysis can be summarized as follows:

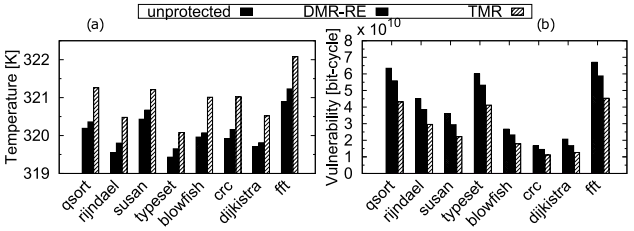


Figure 3: Variations in total (a) Peak Temperature and (b) Vulnerability of different benchmark applications.

Performance Phases: During the execution of an application, there is a variation in the instruction-per-cycle (IPC) of the processor which leads to have different phases of performance for that application, see Fig. 2.

Reliability-Temperature Trade-offs for Different Redundancy Mechanisms: Fig. 3, shows that different applications have variations in their total peak temperature and vulnerabilities for different reliability features. And during execution, different components have varying vulnerability and temperature profiles, see Fig. 2. *Therefore, this temporal diversity in temperature variations for various redundancy techniques during different run-time phases can be exploited to enable temperature-efficient reliability management under a constrained scenario.*

Component Level Vulnerability Variations: As shown in Fig. 4 different architectural components have different peak temperatures variations during the whole execution time, and Fig. 2 shows that vulnerability is also changing differently with time for different architectural components. Therefore, *Our analysis shows that based on the temperature and vulnerability variations in each phase, temperature-aware fine-grained reliability management can be achieved by leveraging different reliability features for different processor components executing an application.*

In this paper, we use DMR-RE and TMR as the underlying basic reliability features/knobs, though our proposed technique can easily be adopted to any other reliability mechanisms.

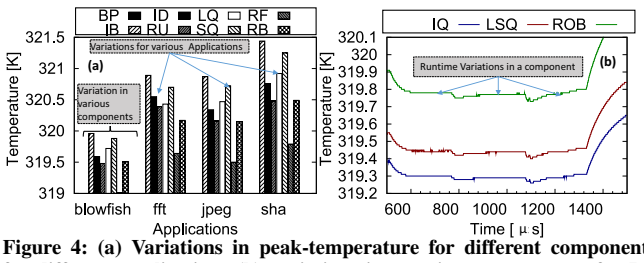


Figure 4: (a) Variations in peak-temperature for different components for different applications (b) variations in run-time temperature for RF

III. System Models

Processor Model: This paper considers an out-of-order processor having different n architectural components i.e., $C = c_1, c_2, .., c_n$ which are equipped with k different reliability features, i.e., $R = r_1, r_2, .., r_k$, and also has an in-built reliability manager, which can be used to control these features. Where r_1 is the least reliable mode and r_k is the most reliable mode.

Application Model: Each application is assumed to have m different execution phases, such that $F = f_1, f_2, .., f_m$, which are derived from performance (IPC) profiles. Also, the execution time budgets i.e., $E = e_1, e_2, .., e_m$ for each phase must be fulfilled to guarantee overall application's deadline constraint.

Temperature and Vulnerability Model: In this paper, HotSpot thermal model [15] is used for the temperature modeling and we use architectural vulnerability factor, AVF, [9] as soft error vulnerability model. A bit *"i"* in the processor is said to be vulnerable at time *"t"* if an error in this bit will be consumed by the processor at any time *"t+τ"*, with the potential to cause failures in the execution and/or program output. The vulnerability of that data bit is then *"τ"* bit-cycles. The system vulnerability (bit-cycles) is the sum of all bits in the system that are vulnerable during the execution. The total vulnerability is $V = \sum_{i=1}^{n} v_i$, Where v_i is the vulnerability of a component i in bit-cycles. From [14], the corresponding reliability of a component having vulnerability v_i is calculated using:

$$R_i(v_i, t, \lambda) = e^{(-\lambda * v_i * t)} \quad (1)$$

IV. Temperature-Aware Reliability Management

As discussed in Section II, different architecture components exhibit different vulnerabilities, performance and temperature properties over entire application execution. To increase the reliability of the component and ultimately of a system, one potential knob is to increase the reliability by selecting an appropriate reliability mechanism (e.g., unprotected, DMR-RE or TMR), where a more robust reliability mechanism also corresponds to higher power consumption and ultimately to higher temperature as shown in Fig. 3. Therefore in order to achieve temperature-efficient reliability we propose a temperature-aware run-time reliability manager that selects a particular reliability mechanism (unprotected, DMR-RE, or TMR) considering the variations in temperature and vulnerability for different phases of the applications. Fig. 5 shows the overview of our proposed technique, which performs the phase identification at design time and reliability management at run-time.

A. Architectural Support for Fine-Grained Management

Our processor model has different reliability features and a controller to control between these features. The different processor components are architecturally extended to have different redundancies (like DMR-RE and TMR) and an inbuilt controller to control these features. Thus, different component can be operated in any of available reliability modes.

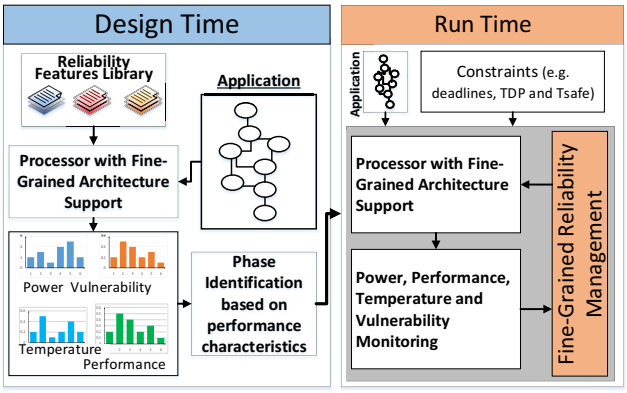


Figure 5: Overview of our proposed technique

B. Problem Formulation

Based on above models and notations (Section III) for representing temperature and vulnerability of *n* different processor components during an application phase, we can formulate the problem of mapping one of the reliability features *m* for various *n* components as multidimensional bounded knapsack problem (MBKP) as follows:

- Soft error vulnerability of different pipeline component at different application phases; $V_i(t)$ *s.t.* $t\epsilon[1,..m], i\epsilon[1,..n]$
- Power consumption of different pipeline component at different application phases; $P_i(t)$ *s.t.* $t\epsilon[1,..m], i\epsilon[1,..n]$
- Peak temperature of different pipeline component at different application phases; $T_i(t)$ *s.t.* $t\epsilon[1,..m], i\epsilon[1,..n]$
- Total power budget of the system is **P**
- Safe temperature limit of the system is **T**$= T_{safe}$

Then the goal is to optimally select a reliability feature *j* for a component *i*: $R_{ij}, \forall i\epsilon[1,..n], j\epsilon[1,..k]$. Such that the total vulnerability of the system is minimized and still the thermal constraint is also fulfilled. That is:

$$min \sum_{i=1}^{n} v_i(t) \qquad s.t. \quad \sum_{i=1}^{n} T_i(t) \leq \mathbf{T} \tag{2}$$

and also meeting the application deadline time for each phase: $e_i \leq deadline_i, \forall i\epsilon[1,..m]$. To optimize this problem, a heuristic algorithm is developed as shown in Algorithm 1. This algorithm heuristically finds the best possible reliability feature for each phase for different components under the constraints of thermal limit and execution time during that phase.

C. Phase Identification

This identifies the similar continuous set of performance values (IPC) to be in one phase which is used to distinguish various execution phases for a given applications.

D. Temperature-Aware Fine-Grained Reliability Manager

Based on our analysis, a temperature-efficient run-time reliability manager is proposed which maximizes the soft error reliability in a temperature constrained scenario. It is not necessary to provide full time reliability (like DMR-RE or TMR) for whole execution of a given application, instead the reliability manager analyzes the vulnerability and temperature consumption in the current phase of the application and can turn-on or off different reliability features of different components accordingly. The algorithm uses a reliability metric (ratio of reliability to temperature) to decide heuristically which component is to be operated in which reliability mode. Initially, all the components are in the highest reliability mode (Line.2).

Algorithm 1: Fine-Grained Reliability Manager

```
1  function ReliabilityManager
   (Phases, R_req, P_Budget, T_safe);
   Input  : Phases, R_req, P_Budget, T_safe
   Output: Redundancy feature for each component
2  put each component in highest reliable mode;
3  for Phases ∈ IPC do
4      make V_ij, R_ij, P_ij, T_ij matrices;
5      reliability metric RM_1 = reliability / power;
6      reliability metric RM_2 = reliability / temperature;
7      for i=Components ∈ n do
8          K_1 = all components power;
9          K_2 = all components temperature;
10         while cost K_2 > T_safe do
11             take first derivative of RM_1 and RM_2;
12             if switching modes does not change in Temp?
               then
13                 chose the mode with min(RM_1);
14             else
15                 chose the mode with min(RM_2);
16             end
17             update cost K_2 and K_1;
18         end
19     end
20 end
```

V. Experimental Setup

We evaluated our proposed system for various MiBench benchmark applications executing on a cycle accurate simulator GemV [10], which is the extension of Gem5 [11]. This simulator is also extended to support DMR-RE and TMR features of different architectural components like load-store queue, instruction queue, history buffer, re-order buffer. GemV is also used to perform vulnerability, performance and execution time analysis while McPAT and HotSpot [15] are used for power and temperature analysis respectively. The vulnerability statistics for different applications are obtained through GemV simulation. The statistics are dumped after every 1000µs and the first 500µs are ignored to allow the cache to be filled. Then the power trace for each dump is generated using McPAT, which is used by HotSpot to generate temperature traces. The reliability is calculated using Eq.1, where fault error rate is taken as $\lambda = 10^{-6}$ [12].

VI. Comparison with State-of-the-Art

We compared our proposed system with two state-of-the-art reliability management techniques: (i) dynamic DMR that

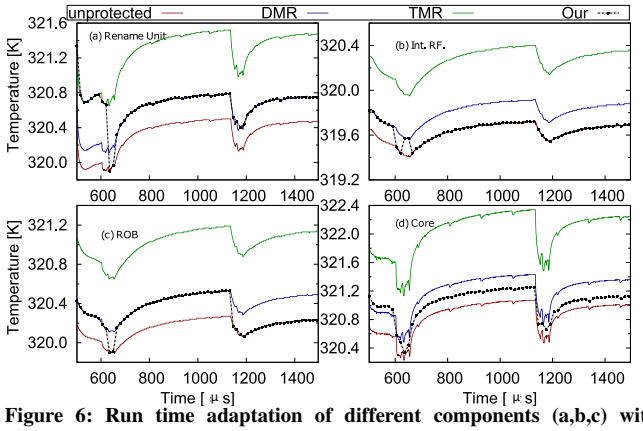


Figure 6: Run time adaptation of different components (a,b,c) with different modes and the peak core (d) temperature improvement

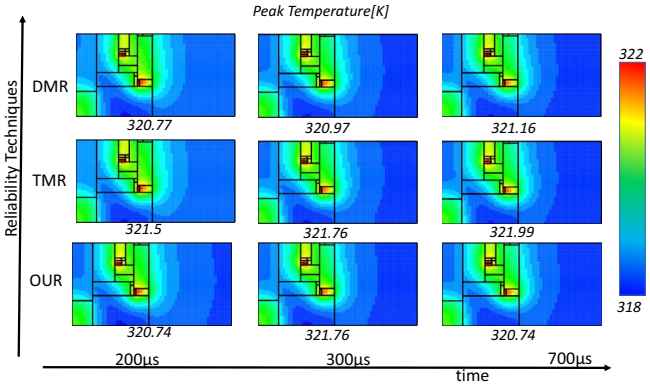


Figure 7: Thermal maps comparison for our technique with DMR-RE and TMR techniques

activate DMR feature for applications based on their vulnerability [3], (ii) full TMR where applications are fully executed in TMR mode throughout [8]. Fig. 6 shows, the time-wise thermal map comparison of our proposed technique to the state-of-the-art techniques and Fig. 7 shows individual components and cores run-time adaptation while our techniques is applied. The comparison of these techniques with ours is shown in Fig.8 which states that:

1) TMR is more temperature hungry and it violates the constraint while our reliability manager provides maximum reliability within the constraint.
2) our reliability manager decreases soft error vulnerability by 21%, in other words it increases the reliability by 30% as compared to DMR within the temperature constraint.

That is, TMR is more tolerant but always has high peak temperature as compared to DMR-RE and it also violates the thermal constraint. On the other hand, DMR-RE has low peak temperatures but it less tolerant than our techniques. Therefore, the results depict a comprehensive compromise in terms of peak temperature and reliability under thermal constraint.

VII. Conclusion

This paper presents a temperature-aware fine-grained reliability management system for soft error resiliency that minimizes vulnerability under a given temperature constraint by harnessing vulnerability and temperature variations during

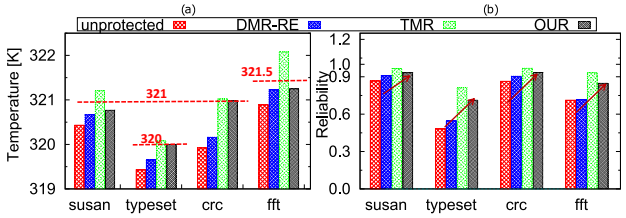


Figure 8: Comparison of our proposed technique with state-of-the-art techniques [3], [8]

execution phase of an application. This is achieved using a fine-granularity control of reliability features for different architectural components depending upon their vulnerability and temperature profiles during a phase. When compared to state-of-the-art, our proposed reliability manager achieves significant reliability improvements up to 30% while meeting the given thermal constraint. This suggests that vulnerability and temperature trade-offs can be harnessed at component-level with fine-granularity during different phases of the applications, and can provide great opportunities for temperature-efficient soft error reliability management.

Acknowledgement

This work is supported in parts by the German Research Foundation (DFG) as part of the priority program "Dependable Embedded Systems" (SPP 1500 - spp1500.itec.kit.edu).

References

[1] International technology roadmap for semiconductors,http://public.itrs.net/reports.html.
[2] J. Henkel et. al., "Design and architectures for dependable embedded systems," Proceedings of the seventh IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, 2011.
[3] R. Vadlamani et al.,"Multicore soft error rate stabilization using adaptive dual modular redundancy", DATE, pp. 27-32, 2010.
[4] D. Ernst et al., "Razor: circuit-level correction of timing errors for low-power operation," IEEE MICRO, vol. 24, no. 3, pp. 10-20, 2004.
[5] G. A. Reis et al., "SWIFT: Software Implemented Fault Tolerance", IEEE CGO, pp. 243-254, 2005.
[6] T. Ebi, D. Kramer, W. Karl, J. Henkel, "Economic learning for thermal-aware power budgeting in many-core architectures", CODES+ISSS, pp.189-196, 2011.
[7] Ebi, T, Al Faruque, M.A., and Henkel, J, "TAPE: Thermal-Aware Agent-Based Power Economy for Multi/Manycore Architectures," In Proc. of the International Conference on Computer-Aided Design (ICCAD), 2009
[8] S. S. Mukherjee, M. Kontz, S. K. Reinhardt. Detailed design and evaluation of redundant multithreading alternatives, In IEEE Int'l Symp. Comput. Arch. (ISCA), pp. 99110, 2002.
[9] S. S. Mukherjee et.al., "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in Micro. IEEE/ACM, 2003.
[10] Srinivas Karthik Tanikella et al. GemV: A Validated Toolset for the Early Exploration of System Reliability. ASAP, 2016.
[11] N. Binkert et al., "The gem5 simulator," ACM SIGARCH Computer Architecture News, vol. 39, no. 2, 2011.
[12] Huang et.al., "Low-Energy Standby Sparing for Hard Real-Time Systems," IEEE TCAD 2012.
[13] A. Ejlali, B.M. Al-Hashimi, and P. Eles, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," IEEE Transactions on VLSI Systems 2006.
[14] I. Koren, C.M. Krishna. 2007. Fault Tolerant Systems.
[15] W. Huang et al. "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," IEEE Trans. VLSI Syst., vol. 14, no. 5, May 2006.
[16] Intel Corporation. Dual-core intel xeon processor 5100 series datasheet, revision 003, August 2007.