Multicore Soft Error Rate Stabilization Using Adaptive Dual Modular Redundancy

Ramakrishna Vadlamani, Jia Zhao, Wayne Burleson and Russell Tessier
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst MA, United States

***Abstract*— The use of dynamic voltage and frequency scaling (DVFS) in contemporary multicores provides significant protection from unpredictable thermal events. A side effect of DVFS can be an increased processor exposure to soft errors. To address this issue, a flexible fault prevention mechanism has been developed to selectively enable a small amount of per-core dual modular redundancy (DMR) in response to increased vulnerability, as measured by the processor architectural vulnerability factor (AVF). Our new algorithm for DMR deployment aims to provide a stable effective soft error rate (SER) by using DMR in response to DVFS caused by thermal events. The algorithm is implemented in real-time on the multicore using a dedicated monitor network-on-chip and controller which evaluates thermal information and multicore performance statistics. Experiments with a multicore simulator using standard benchmarks show an average 6% improvement in overall power consumption and a stable SER by using selective DMR versus continuous DMR deployment.**

Keywords-architectural vulnerability, DVFS, monitor network

I. INTRODUCTION

The reliability and performance of multicore processors has become a significant concern as process technologies have scaled in recent years. Embedded on-chip sensors, such as thermal and error monitors, allow for the run-time assessment of multicore behavior in an effort to positively influence system behavior. For example, to mitigate the impact of soft errors, current multicores implement redundancy-based error detection and recovery schemes including component dual modular redundancy (DMR) and redundant multithreading [1]. However, these approaches may not be appropriate in all cases as they incur significant performance and power overhead and often require significant operating system support. A localized, low overhead error reduction approach which can be selectively enabled provides a possible alternative.

In general, memory-based components in processor cores are vulnerable to single event upsets due to radiation. Although large memory structures are often protected by error checking and correcting circuits, smaller components, such as instruction queues and retirement order buffers, have less protection. Fortunately, not every bit flip in these components leads to an observable system error. A component's architectural vulnerability factor (AVF) states the probability that a fault generated in a processor structure will result in an error in the program output [2]. The AVF for various processor structures has been shown to vary widely both across and within applications [3]. Previous studies [3][4][5] have described the efficient run-time estimation and use of AVF for single core processors in an effort to promote stable processor failure in time (FIT) rates. However, the growth of multicore use and frequent per-core voltage and frequency scaling necessitates the reexamination of AVF calculation and use.

Dynamic voltage and frequency scaling (DVFS) is commonly used in multicores to reduce hotspot temperatures and system power consumption. Unfortunately, voltage decreases and frequency increases can adversely affect system reliability [6][7], necessitating a fast system response to maintain a stable multicore soft error rate. One approach to maintaining system reliability is to enable a small amount of redundant resources for critical system components in the presence of increased soft error risk. This risk is determined by comparing the instantaneous AVF for the components following DVFS against a predetermined threshold. If the threshold is passed, redundant components are enabled to facilitate DMR actions.

In this paper the power effects of using AVF-enabled DMR in a multicore environment are explored. AVF values for critical resources are continually assessed throughout processing but special consideration is given following thermally-induced voltage and frequency scaling. Thermal and AVF monitor data are transported to a centralized controller via a specialized monitor data interconnect. The controller collaboratively uses the data to perform DVFS on affected cores and to enable/disable redundant resources. Our approach is designed to scale to tens of cores, enabling flexible fault coverage and performance and power control enhancement. A multicore architectural simulator and an interconnect simulator are used to assess the power and performance benefit of our approach for 8 and 16 processor multicores. An overall power benefit of 6% on average is achieved for 16 cores versus the continual use of redundant resources.

The remainder of the paper is organized as follows. Section II presents brief background on AVF, DVFS, and soft error protection. Section III describes AVF estimation for multicores and the impact of thermal information on its calculation. Section IV discusses our experimental approach and experimental

results are presented in Section V. Section VI concludes the paper and offers directions for future work.

II. Background

Measurements of a processor component's architectural vulnerability factor determine the probability that a soft error will lead to a user visible error [2]. If a data bit is necessary for architecturally-correct execution (ACE) [2], an output error will be produced whenever it flips due to a transient fault. Effectively, the soft error rate (SER) of a component is the product of the raw SER (overall bit flip rate) and the component's AVF. Per-component AVF is a time-varying quantity that varies both across and within applications based on a component's utilization [2][3][4]. For example, AVF metrics for the instruction queue (IQ), retirement-order buffer (ROB) and the load/store queue (LSQ) vary by 40%, 30% and 20% respectively during application run time [4]. These metrics suggest that affected processor units can potentially benefit from an AVF-aware redundancy scheme that disables redundant units during periods of low AVF, thus saving power [2][3][4].

Accurate run-time AVF evaluation has recently been shown to be computationally feasible [2][3]. Walcott, et al. [3] and Biswas, et al. [4] demonstrated that the aggregated AVFs of uniprocessor pipeline components can be estimated with up to a 90% accuracy using a small set of periodically-sampled microarchitectural parameters. This quantized-AVF (Q-AVF) approach is lightweight since the amount of processed data is restricted to a small quantum over a restricted sampling interval.

AVF varies with the operating frequency and voltage of a component since it impacts the utilization of the component [8]. In Soundararajan, et al. [8], this variation was quantified for DVFS applied to a uniprocessor. More recently, Siddiqua and Gurumurthi [9] used AVF variation to support redundant multithreading (RMT) in an effort to reduce soft errors. In the latter two cases, SER levels are considered static and unaffected by per-core variations in voltage and frequency.

Error detection for storage components in processor-based systems is often performed using dual modular redundancy, in which outputs of duplicate copies of a component are compared before memory commits are performed [1][10]. DMR incurs a power consumption penalty and should only be used if a processor component is likely to incur soft errors. Many storage-based processor pipeline components are protected without the need for DMR. Register files and caches are generally protected by ECC/parity-check circuitry. Pipeline latches can use low-overhead error self detection and correction (i.e. Razor) [11]. Additionally, the AVF of a branch predictor is always 0% since a misprediction due to a predictor soft error strike will not lead to an output error [1]. As a result, this paper focuses on the DMR protection of specific components (instruction queue, retirement order buffer, and load store queue) which would otherwise be unprotected. The detection and rollback circuitry required to restore processor state following an error detection have been documented in numerous previous publications [10][12] and are not discussed here.

III. Adaptive AVF Calculation and Use for DMR

Our adaptive DMR approach requires real-time AVF computation and the use of an interconnect architecture for thermal monitor and system parameter data collection and processing. Three specific operating scenarios are considered in which real-time AVF information is used to enable/disable component-based DMR for the instruction queue (IQ), retirement order buffer (ROB), and load-store queue (LSQ):

1) AVF information is used to enable/disable DMR for the components which exhibit an AVF below a *predetermined, fixed* threshold.
2) AVF information is used to enable/disable DMR for the components which exhibit an AVF below a *dynamically-determined, variable* threshold which changes with voltage and frequency updates.
3) AVF information is ignored and DMR is always enabled for the components.

Each of these cases is considered in the context of multicore DVFS performed in response to thermal events.

A. AVF Computation in a Multicore Environment

AVF calculation for IQ, ROB, and LSQ components must occur periodically since AVF values typically show significant run time variation [1][3]. The AVF of each component is determined using microarchitectural parameters obtained from the processor. A linear combination of eight parameters can be combined [4] to describe the AVF for each component at an accuracy level approaching 90%. These parameters include (1) the stores flushed before data translation lookaside buffer response, (2) store buffer utilization, (3) retirement order buffer empty cycles, (4) retirement order buffer utilization, (5) branch misprediction count, (6) reservation station utilization, (7) instruction queue utilization, and (8) total front-end instruction kill latency.

Each parameter is scaled and linearly combined to form the AVF estimates for the three components (IQ, ROB, LSQ) in each processor. Since our processor model is slightly less complex than the one used in [4], the coefficients for parameters (1), (6), and (8) are set to zero. The five remaining parameters related to AVF calculation are commonly monitored in hardware in microprocessors. The required performance monitoring hardware often consists of two parts [13], an event detector and an event counter.

To evaluate AVF in our system, fixed event detectors and counters are needed to collect desired performance information. For example, store buffer (STB) utilization can be determined by monitoring store buffer write and read events. A corresponding counter for STB utilization increases by 1 when an STB write occurs and decreases by 1 when an STB read occurs. Since the size of the STB is known, it is straightforward to calculate STB utilization from the counter. The same method works for the utilization calculation of an ROB, a reservation station and an instruction decode queue. Event detectors are connected to both the read and write signals of the target structure. A write event increments the counter and a read event decreases the counter

for the target structure. To determine ROB empty cycles, an event detector is connected to the ROB empty signal. A count is incremented for each cycle the signal indicates an ROB empty. The similar method works for the branch misprediction counter. The event detector is connected to a misprediction signal. Whenever a misprediction happens, the counter increases by 1.

Fig. 1 shows the structure of a processor pipeline and the associated AVF monitoring circuitry. Event detectors are connected to the IQ, LSQ, ROB and branch predictor to probe operations in these units. Some detectors have been omitted from Fig. 1 for clarity. Five counters are connected to the corresponding detectors to obtain utilization information for the five parameters. The hardware cost of the performance counters and detectors is modest. AVF calculation is performed every 1024 cycles [4] leading to a counter requirement of 5 * 10 bits = 50 bits. Each detector can be implemented in a small number of logic gates.

The hardware overhead required to duplicate the IQ, LSQ, and ROB is also modest. For our architecture, based on an Alpha264, a total of 16, 112 and 176 thirty-two bit values are needed (see Table 1 on page 5). This analysis indicates a total of 9728 storage bits. As a result, the total hardware overhead for per-core AVF-enabled DMR is about 200K transistors, a small percentage of the total processor transistor count, including cache.

As shown in Fig. 1, monitored information is transferred to a centralized processor using an interconnect network (MNoC), which is discussed in Section IV. The centralized processor (MEP) calculates the AVF of each core based on the obtained counter values.

B. Reliability-aware AVF threshold computation

Processing components require a stable SER to operate properly. Due to the masking capability of the AVF, the effective SER of a processor core [2] is defined as:

$$\text{Effective\_SER} = \text{AVF} * \text{Raw\_SER} \qquad (1)$$

The *Raw_SER* (total expected bit flip rate) is reduced to an *Effective_SER* since not all soft errors eventually affect the visible program output. Processors generally have a target *effective_SER* threshold (*Target_SER*) which is predefined for the architecture. To ensure proper operation, it is desirable to keep the instantaneous SER of the processor core components below the target SER. If the rate rises above the target SER threshold, resource redundancy can be used to mitigate errors. Equation 1 can be rewritten as:

$$\text{Target\_SER} = \text{AVF\_threshold} * \text{Raw\_SER} \qquad (2)$$

which indicates that the target SER threshold is directly related to the *AVF_threshold*. If the *Raw_SER* is constant, the need for component redundancy can be directly determined from the measured AVF. A measured value for a component which is above the AVF threshold indicates the need for DMR. Otherwise, DMR can be deactivated.

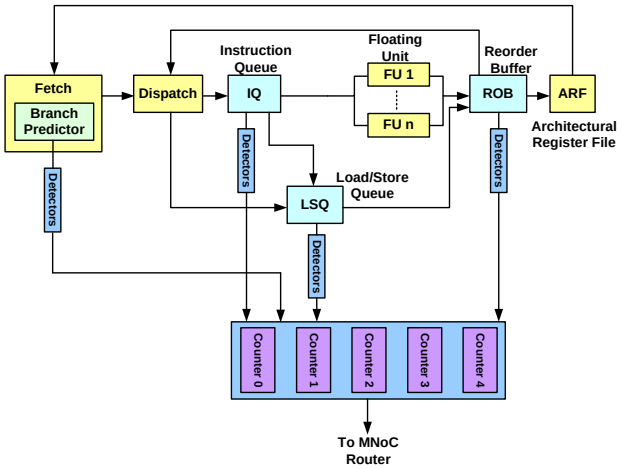

Fig. 1. AVF monitor for one pipeline

In most cases, however, *Raw_SER* is not constant. For example, the SER fault model in Equation 3 [6] [14] assumes an exponential relationship for SER with respect to the frequency and supply voltage. In the equation, $\mathbf{f_{min}}$ corresponds to a normalized minimum-energy frequency [15] (e.g. an $\mathbf{f_{min}}$ of 0.2 indicates the minimum frequency is 20% of the maximum) and **x** indicates the relative scaling of **f** and **V** between their min and max values.

$$SER(V, f) = SER_0 \cdot 10^{\frac{d(1-x)}{1-f_{\min}}} \qquad (3)$$

Frequencies below $\mathbf{f_{min}}$ (typically 5% of $\mathbf{f_{max}}$ [15]) consume additional energy due to increased memory latency. Parameter **d = 2** is based on the expected fault injection source [6]. SER_0 is the raw SER corresponding to the maximum voltage and frequency used by the multicore. This model indicates that both *AVF_threshold* and instantaneous *SER* must be considered in determining the need for DMR. If raw *SER* increases, the AVF threshold used to enable redundancy must be reduced so that target SER levels are not crossed. In summary, the relationship between *AVF_threshold* and *Target_SER* can be expressed as:

$$\text{AVF\_Threshold} = \text{Target\_SER}/\text{SER}(V,f) \qquad (4)$$

where *SER(V, f)* can be calculated using Equation (3).

C. Example AVF threshold and overhead computation

The AVF thresholds used for experimentation in this work were determined as follows. Mukherjee et al. [2] determined a FIT of approximately 200 to 2000 for the 200,000 vulnerable bits in the SPARC64 microprocessor. Since each core (based on an Alpha264) in our example multicore has roughly 10% the number of vulnerable bits of a SPARC64, *raw_SER* in Equation 2 is set to 28 FIT. Additionally, Mukherjee et al. determined a 1000 year mean time between failures (MTBF) for a SPARC64 and suggested that MTBF should be increased proportionally for each core when a multicore system is considered. For our 8 core system, an MTBF of 8000 years is used to achieve an

overall 1000 year MTBF. This MTBF corresponds to a 14 FIT per core (10^9/8000*365*24), which is our *target_SER*. Using Equation 3, an AVF_threshold of 50% is determined.

Our system dynamically computes AVF thresholds for each core in the system based on current frequency and voltage values. A two-level DVFS system is implemented which switches the voltage and frequency between more aggressive (2GHz, 1.2V) and less aggressive (1GHz, 0.84V) parameters. Since the minimum frequency for our system is equal to 0.05 * f_{max}, f_{min} is set to 100MHz/2GHz = 1/20 after normalization. When this value of f_{min} and the *raw_SER* (i.e. SER_0) of 28 FIT are used in Equation 3, a new *raw_SER* at 0.84V, 1GHz is set to 10 times SER_0. Using Equation 4, this value leads to an adjusted *AVF_threshold* of 25% for low voltage (0.84V). This adjustment can be applied since AVF values are influenced by frequency changes [8].

IV. Experimental Approach

Our AVF-enabled DMR approach benefits from the use of a dedicated interconnect for monitor traffic. The five microarchitectural parameters listed in Section III.A are collected by an AVF monitor located in each processing core (Fig. 1). The monitor interfaces to lightweight monitor network on chip (MNoC) routers which transport the information to a centralized monitor executive processor (MEP) [16]. The irregular topology support provided by MNoC is suited to the distributed placement of an AVF monitor and one MNoC router in each core (Fig. 2). In addition to AVF information, MNoC also transports thermal monitor information and control information which modulates per-core voltage and frequency. A total of 8 thermal monitors are allocated per core to achieve 0.1 degC accuracy [16].

Shared memory multiprocessor systems consisting of 8 and 16 cores are used for experimentation. Each processor contains the following duplicated pipeline structures: instruction queue, retirement-order buffer/reservation stations and load/store buffers. When DMR is enabled for a pipeline component, a per-core error detection system flags an error if a component output does not match the data from its counterpart. An AVF monitor, the 8 thermal monitors, and 3 error monitors per core are connected to an MNoC router via a multiplexer. The error monitor's low bandwidth limits their impact on monitor data and processing.

The MEP executes the following DMR throttling algorithm for each redundant component in each core. Initially, all redundancy is enabled. Each AVF monitor in the multiprocessor system is then sampled in a round-robin fashion. When the AVF values for a processor component falls below a predefined lower threshold, the MEP sends a disable signal for the replicated resource. If the AVF is greater than the threshold, the redundant resource is re-enabled. Sequentially, the DVFS algorithm on the MEP proceeds as follows:

- Measure the instantaneous values of temperature (T) and AVFs for each core in each sampling period
- If *T* > *threshold*, reduce *V*, *f* (perform DVFS) for affected core. *Update AVF_Threshold* values.

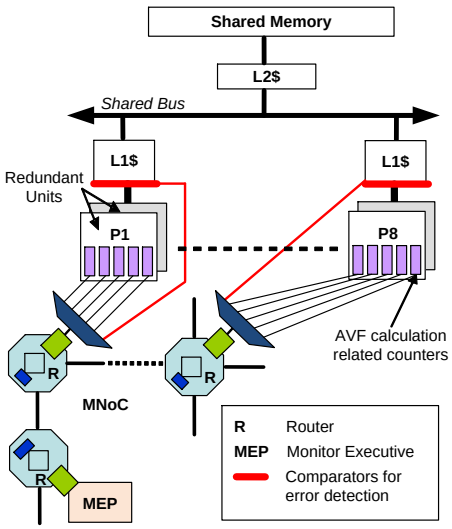

Fig. 2. An 8-core system for AVF-aware DMR throttling

- If *AVF* > *AVF_Threshold* for a processor component, enable DMR for the component. Otherwise disable DMR.

The *Update_AVF_Threshold* routine required in the algorithm uses Equation (4).

Based on previous work [17], the sampling rate of a thermal monitor is set to 1 per 800 clock cycles at 500 MHz. As mentioned in Section III, the five parameters needed for AVF calculation are transferred to the MEP once every 1024 system cycles. Since one MNoC router is used per core, the injection rate per MNoC router is 8 * (1 per 800 cycles) + 5 * (1 per 1024) = 1 injection per 67 clock cycles.

To assess the expected MNoC monitor data latency in MNoC, a latency experiment was performed for a 16 processor multicore system. As shown in Fig. 3, as long as the injection rate per router is less than 1 per 17 cycles, MNoC latency remains consistently low. In our system, the injection rate per MNoC router is low enough to provide an MNoC latency of less than 15 clock cycles. For 8 core multiprocessors, our experiments show a similar 15 cycle latency.

V. Experimental Procedure and Results

The simulation setup including the configuration of the

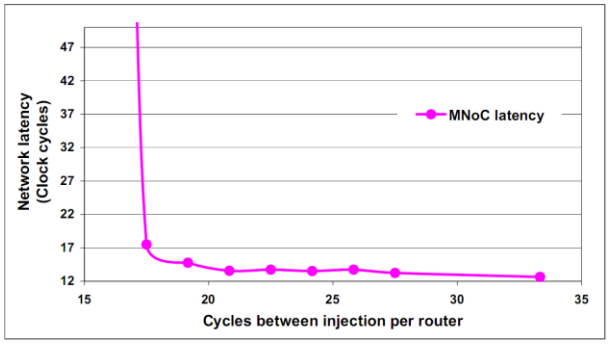

Fig. 3: Latency vs. injection rate per router for a 16 router system. Results were generated using a modified Popnet simulator

Table 1. Experimental Setup

Simulator	SESC multiprocessor simulator [18]
Technology	90 nm
Num of processors	8, 16
DVFS V, f levels	f(high)=2GHz, V(high)=1.2V f(low)=1GHz, V(low)=0.84V
Benchmarks	SPLASH2 (400M instructions each)
Processor configuration	
Instruction Issue	4 out-of-order
I-cache	64KB, 4-way
D-cache	64KB, 8-way, 2 cycles
Branch Predictor	Hybrid
Branch Target Buffer	4K entries, 16-way
Instruction Queue	16 entries
Retirement Order Buffer	176 entries
Load/Store Buffers	56/56 entries
L2 Cache	1MB, 8-way, 10 cycles

simulated shared memory multiprocessor system is summarized in Table 1. A modified SESC [18] multiprocessor architectural simulator is used to evaluate the run-time effects of DVFS on a series of applications and the collection of information from one AVF (Fig. 1) and 8 thermal monitors in each processor core. The MEP functionality is assigned to one of the cores in the simulated multicore system.

The processor power model used by SESC is based on Wattch [19]. The cache power model is based on CACTI [20] and the temperature model for both (called SESCSpot) is based on HotSpot [16]. SESCSpot calculates the temperature of processor subblocks based on the power trace of the architecture in a post processing fashion. The processor architecture is modeled on an Alpha264 with a MIPS ISA and the floorplan of each processor core used for thermal modeling is based on prior work [21]. For our DVFS implementation we integrated SESCSpot into the core of the SESC simulator to obtain temperature readings at run-time. This approach enabled the MEP to sample the temperature readings at run-time and execute the DVFS algorithm.

In order to assess the benefits of our AVF-based dual modular redundancy approach, the three specific operating scenarios described in Section III are considered:

1) *AVF threshold fixed* – DMR enabled when a component AVF passes a fixed threshold
2) *AVF variable threshold* – DMR enabled when a component AVF passes a threshold which varies with DVFS based on Equation 4.
3) *Full DMR*: DMR is always enabled for all three components (IQ, ROB, and LSQ).

All three of these cases are considered in the context of DVFS. The third case is the worst case scenario and it is used as a baseline for the other two. The first case considers the AVF threshold for a component to be fixed regardless of voltage and frequency. As a result, the AVF threshold must be set to a reduced value of 0.25 (25% of bits are important) which is used

Table 2. Power benefit and overhead results for 8 and 16 core system

Test bench name	Case	8 core		16 core	
		Power per core (W)	Power benefit (%)	Power per core (W)	Power benefit (%)
LU	Full DMR	11.50		11.75	
	Fixed threshold	10.88	**5.39**	11.19	**4.77**
	Variable threshold	10.80	**6.09**	11.10	**5.53**
Ocean	Full DMR	9.83		10.04	
	Fixed threshold	9.63	**2.03**	9.63	**4.08**
	Variable threshold	9.13	**7.12**	9.29	**7.47**
FMM	Full DMR	14.28		10.28	
	Fixed threshold	14.13	**1.05**	9.75	**5.16**
	Variable threshold	12.28	**14.01**	9.69	**5.74**
Radix	Full DMR	4.48		4.25	
	Fixed threshold	4.38	**2.23**	4.13	**2.82**
	Variable threshold	4.12	**8.04**	3.94	**7.29**

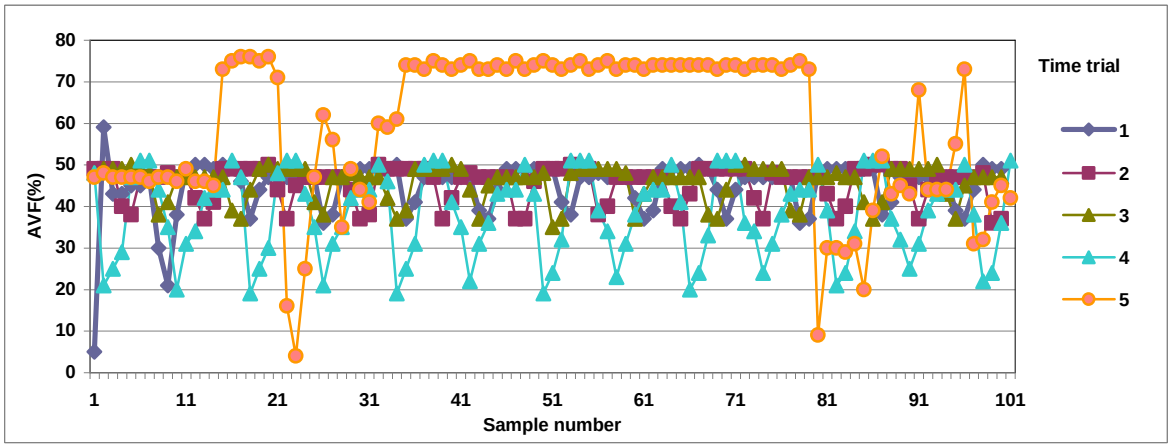

Fig. 4. Five 100-sample instruction queue AVF traces for the LU benchmark

during both high voltage and low voltage usage. The second case considers the AVF threshold as dynamically varying as DVFS changes voltage and frequency levels. AVF thresholds of between 25% and 50% are determined by the MEP for each processor component.

The power benefits of a variable AVF threshold in enabling DMR are shown in Table 2 for four SPLASH2 benchmarks mapped to 8 and 16 cores. Portions of each benchmark are distributed across the cores. DMR is only performed on the specific processor components which have an AVF greater than the target threshold. On average, the variable AVF threshold approach (case 1) reduces core power (without cache) versus full DMR (case 3) by about 8% and 6%, respectively, for 8 and 16 core processors. An average power improvement of 6% and 2% is seen for the variable AVF threshold approach versus the fixed AVF threshold approach. In general, the cost of providing a stable SER through DMR is low. The power cost of including DMR is about 5% for 8 cores and 6% for 16 versus unprotected scenarios. The power consumption of MNoC (~250 mW) is considered in these calculations. The 8-core FMM application shows a particular savings with a variable versus fixed threshold (14% vs. 1%) since most AVF values are above the fixed threshold.

The variability of AVF is apparent from Fig. 4, which shows AVF variation across LU benchmark run time for an instruction queue for five traces of 100 samples. AVF values are measured over several time trials. In general, calculated AVF is mostly at or below 50% with frequent deviations over a wide range. Wattch has previously been shown to have an accuracy of about 10%, although there has been no explicit discussion on its precision [19]. Since we measure relative benefit using Wattch, the absolute precision of Wattch is a less important factor.

VI. Conclusion and Future Work

The flexible deployment of dual modular redundancy can provide SER stability in DVFS-enabled multicores. In this paper, real-time architectural vulnerability metrics for processor components are used to evaluate if DMR is needed based on a dynamic threshold. Our system uses a novel monitor data collection and processing system to determine AVF and enable redundancy if it is needed. A 6% reduction in power is achieved versus always-active DMR for a 16 processor multicore while a stable multicore SER is maintained. Since we focus on IQ, ROB and LSQ processor memory structures, an error correction code (ECC) based error correction technique could potentially result in lower overall power, performance and area overhead compared to DMR. In the future, the benefits of such an approach will be assessed.

Acknowledgements

This work was funded by the Semiconductor Research Corporation under Task 1595.001. The authors would like to acknowledge our SRC liaisons at Intel, AMD, and Freescale.

References

[1] S. Mukherjee, J. Emer, and S. Reinhardt, "The soft error problem: an architectural perspective", in the Proc. of Int'l Symp. High-Performance Computer Architecture, pp. 243-247, 2005.

[2] S. Mukherjee, et al., "A systematic methodology to compute the architectural vulnerability factors of a high-performance microprocessor", in the Proc. of 36th Ann. Int'l Symp. Microarchitecture, pp 29-40, 2003.

[3] K. Walcott, et al., "Dynamic Prediction of Architectural Vulnerability From Microarchitectural State", in the Proc. of the Int'l Symp. Computer Architecture, pp. 516-527, 2007.

[4] A. Biswas, et al., "Quantized AVF: A means of capturing vulnerability variations over small windows of time", IEEE Workshop on Silicon Errors in Logic - System Effects, 2009.

[5] X. Li, S. V. Adve, P. Bose, and J. A. Rivers, "Online estimation of architectural vulnerability factor for soft errors", in the Proc. of Int'l Symp. Computer Architecture, pp.341-352, 2008.

[6] D. Zhu, et al., "The effects of energy management on reliability in real-time embedded systems", in the Proc. of the IEEE/ACM Int'l Conf. Computer Aided Design, pp. 35-40, 2004.

[7] Z. Baoxian, H. Aydin, and D. Zhu, "Reliability-aware dynamic voltage scaling for energy-constrained real-time embedded systems", in the Proc. of the IEEE Conf. Computer Design, pp. 633-639, 2008.

[8] N. Soundararajan, et al., "Impact of DVFS on the architectural vulnerability of GALS architectures", in the Proc. of the Int'l Symp. Low Power Electronics and Design, pp. 351-356, 2008.

[9] T. Siddiqua, and S. Gurumurthi, "Balancing soft error coverage with lifetime reliability in redundantly multithreaded processors", in the Proc. of Int'l Symp. Modelling, Analysis and Simulation of Computer and Telecommunication Systems, 2009.

[10] A. Golander, S. Weiss, and R. Ronen, "DDMR: Dynamic and scalable dual modular redundancy with short validation intervals", in Computer Architecture Letters, vol. 7, issue 2, pp. 65-68, 2008.

[11] D. Ernst, et al., "Razor: circuit-level correction of timing errors for low-power operation", IEEE Micro, vol. 24, no. 6, pp.10-20, 2004.

[12] K.-L. Wu, et al., "Error recovery in shared memory multiprocessors using private caches", IEEE Transactions on Parallel and Distributed Systems, Volume 1, Issue 2, pp 231 – 240, 1990.

[13] B. Sprunt, "Pentium 4 performance-monitoring features", IEEE Micro, vol. 22, no. 4, pp. 72-82, 2002.

[14] P. Hazucha, and C. Svensson, "Impact of CMOS technology scaling on the atmospheric neutron soft error rate", in IEEE Transactions on Nuclear Science, pp. 2586-2594, 2000.

[15] X. Fan, C. Ellis, and A. Lebeck, "The synergy between power-aware memory systems and processor voltage", in the Proc. of the Workshop on Power-Aware Computing Systems, 2003.

[16] S. Madduri, et al., "A monitor interconnect and support subsystem for multicore processors," in the Proc. of the IEEE/ACM Design Automation and Test in Europe Conference, Nice France, pp. 761-766, 2009.

[17] K. Skadron, et al., "Temperature-aware microarchitecture: Modeling and implementation", in ACM Transactions on Architecture and Code Optimization, vol. 1 no. 1, pp. 94-125, 2004.

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

[19] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in the Proc. of the Int'l Symp. Computer Architecture, pp. 83-94, 2000.

[20] P. Shivakumar, and N. Jouppi, "CACTI 3.0: An integrated cache timing, power and area model," in Technical Report 2001/2, Compaq Computer Corporation, 2001.

[21] G. Link, and N. Vijaykrishnan, "Thermal trends in emerging technologies," in Proc. of the Int'l Symposium on Quality Electronic Design, pp. 625-632, 2006.