

Impact of Dynamic Voltage Frequency Scaling on the Architectural Vulnerability of GALS Architectures

Niranjan Soundararajan N. Vijaykrishnan Anand Sivasubramaniam
Dept. of CSE, The Pennsylvania State University
University Park, PA, USA
soundara@cse.psu.edu, vijay@cse.psu.edu, anand@cse.psu.edu

ABSTRACT

Aggressive technology scaling is increasing the impact of soft errors on microprocessor reliability. Dynamic Voltage Frequency Scaling (DVFS) algorithms are conventionally studied from a performance per watt basis. But applying DVFS impacts reliability as well. Since DVFS affects the occupancy of different pipeline structures, they impact the soft error masking seen at the architectural level. Architectural Vulnerability Factors (AVF) captures this masking and in this work we study the impact of DVFS on AVF in a GALS environment. We show that the AVF of pipeline structures could vary by as much as 80% between different DVFS algorithms. Since AVF has a significant impact on the Mean Time To Failure (MTTF) of a system, these results indicate that when choosing a particular DVFS algorithm their reliability impact cannot be ignored. Hence we provide the *Vulnerability Efficiency* for the DVFS algorithms which captures their ability to optimize performance, power and reliability. Our results show that a Non-DVFS environment optimizes vulnerability efficiency better than any of the DVFS algorithms.

Categories and Subject Descriptors

C.1.0 [Processor Architectures]: General

General Terms

Experimentation, Performance, Reliability

Keywords

Soft Errors, Multi-clocked Domains, Microarchitecture

1. INTRODUCTION

Ever-increasing power consumption is driving microarchitects to consider alternate design solutions. One popular solution is the Globally Asynchronous Locally Synchronous (GALS) design that incorporates multiple independent clocks driving different parts of the processor independently. A Multi-Clock Domain (MCD) design allows the different domains to be run at different voltage and frequency levels providing an ideal platform for applying Dynamic Voltage-Frequency Scaling (DVFS). Several runtime DVFS algorithms

have been proposed in literature to increase the power efficiency of a MCD platform [12].

In addition to power constraints, reliability concerns due to transient errors have emerged as an important design constraint. Transient errors that occur due to high energy particle strikes resulting from atmospheric radiation and packaging effects have gained increasing attention. With aggressive technology scaling, the impact of these transient errors, also called soft errors, is increasing significantly [7]. Past works [13] have explored the impact of voltage scaling techniques on raw soft error rates (SER). In addition to the raw SER, the overall system SER also depends on architectural masking effects captured by a metric called Architectural Vulnerability Factor (AVF) [7]. The AVF of any structure in the pipeline depends on the utilization of a structure. Since DVFS causes instruction flow changes through the pipeline, it affects the utilization of the structures and thereby has an impact on AVF. This work studies the impact of applying different DVFS algorithms on AVF in a GALS environment. This would in turn help designers in choosing the optimal algorithm that meets specific reliability goals. Our current work explores AVF of the issue queue due to its importance in a superscalar design and because prior results show that issue queues have significant AVF [3]. To our knowledge, this is the first effort that explores the architectural vulnerability analysis of DVFS algorithms.

Our results show that except for one DVFS algorithm (Greedy Search [6]), the overall AVF increases when applying DVFS when compared with a Non-DVFS environment. Considering the fact that AVF is inversely proportional to the Mean Time To Failure (MTTF) of a system [7], applying DVFS could increase the failure rate of a system. Since the DVFS algorithms have varying performance-power tradeoffs, we characterize the AVF variations with respect to these parameters as well. Our study shows that the Non-DVFS case does better than any of the DVFS algorithms in terms of reducing AVF per IPC (AVF/IPC). In terms of reducing both total power and AVF ($TotalPower * AVF$), the Greedy search and PI [14] DVFS algorithms provide the best option. Combining all three parameters we look at characterizing an algorithm's *Vulnerability efficiency*, given in terms of $AVF * Watts/IPC$, which gives a DVFS algorithm's capability to reduce AVF and power without degrading performance significantly. We find that voltage-frequency scaling using DVFS does not provide good vulnerability efficiency as compared with a Non-DVFS environment. Amongst the DVFS algorithms, Attack-Decay [8] and PI provide good vulnerability efficiency.

Section 2 provides an introduction to Architectural Vulnerability Factor and how it impacts the overall MTTF. Sec-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISLPED'08, August 11–13, 2008, Bangalore, India..

Copyright 2008 ACM 978-1-60558-109-5/08/08 ...\$5.00.

tion 3 discusses the different DVFS algorithms we analyze in our work while section 4 provides the different configuration parameters used in our study. Section 5 looks at the AVF characteristics of the DVFS algorithms. The section also analyzes the vulnerability efficiency of the DVFS algorithms and finally in section 6 we conclude our work.

2. BACKGROUND

This section provides a background on Architectural Vulnerability Factor (AVF) and how it is important to determine the overall SER of a system. Further this section also provides an introduction to the GALS platform used in this study.

2.1 Architectural Vulnerability Factor (AVF)

Chip reliability can be captured in terms of Failures in Time (FIT) rates [7] which gives the number of failures occurring in a billion hours of operation. FIT rates are inversely proportional to the Mean Time To Failure (MTTF) of the system. The FIT rates of entire chip is the additive sum of the FIT rates of the individual sub-components. The system SER can be obtained by derating the raw SER based on the observation that not all soft errors affect the architectural state. This occurs due to several pipeline phenomena such as idle entries, mis-speculated instructions, dead code and prefetching. These phenomena mask a soft error from appearing in user-visible output. The probability of this deration is given by Architectural Vulnerability Factors (AVF) [5]. Consequently, the overall SER of a system is given by $FIT_{Overall} = FIT_{Raw} * AVF$. Given the importance of AVF in determining the overall system SER, it is important to study how applying different DVFS algorithms affect the overall AVF in a MCD environment.

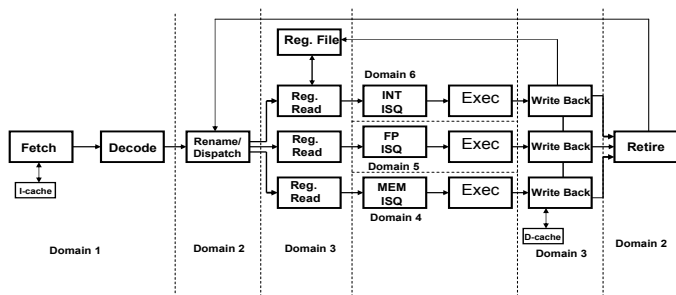


Figure 1: GALS pipeline showing the different asynchronous domains

By knowing the average AVF of different pipeline structures early in the design cycle, an architect can decide the structures that need error protection. AVF estimation using ACE analysis [1, 7] provides a simple and accurate technique to compute a structure’s AVF. Bits, that constitute every instruction, flowing through the pipeline can be classified into those that affect output-correctness, called Architecturally Correct Execution (ACE) bits, and those that do not, called unACE bits. The unACE bits belong to mis-speculated, prefetched or instructions that result in dead code [7]. These bits even if they are affected by a soft error, do not affect final outcome. Hence AVF can be computed as

$$AVF = \frac{\sum_{i=1}^N ACE \text{ Bit} - \text{cycles}}{\left(\sum_{i=1}^N ACE \text{ Bit} - \text{cycles} + \sum_{i=1}^K UnACE \text{ Bit} - \text{cycles}\right)}$$

where *ACE Bit-cycles* refers to occupancy time of each ACE bit that went through the structure, *unACE Bit-cycles* refers to the corresponding value for unACE bits, N is the number of ACE bits that went through the structure and K the number of unACE bits.

2.2 GALS Microarchitecture

The baseline microarchitecture used in this work is shown in Figure 1 [11, 12]. It includes 6 asynchronous domains which are the Fetch, Rename, Register Read, Integer (Int), Floating-Point (FP) and Memory (Mem) domains. Each of the domains operating at different frequencies interface with each other using asynchronous FIFOs [12]. The Fetch stage includes the Fetch queue, branch prediction, I-cache and the decode stage of pipeline. The Rename state includes register-renaming and the retire queue since every instruction retiring needs to update the register rename table. Register reading of source operands from the Int and FP register files occur in the Register Read stage after which the instruction move into the Int, FP or Mem domains based on their type. The Int, FP and Mem domains have separate issue queues besides execution logic. Since the address-generation to actual load-store instruction execution has a significant impact on performance [15], the Mem domain has a separate issue queue for issuing the effective address instructions. We perform sensitivity analysis of our evaluation by varying our baseline architecture later in the paper.

3. DVFS ALGORITHMS

As the importance and need for asynchronous designs have grown, so have the number of algorithms that optimize their power and performance. We analyze the following representative set of DVFS algorithms in this work.

Threshold algorithm: Threshold is a simple algorithm that does the voltage-frequency (VF) switching based on pre-defined utilization thresholds for different structures across different domains [12]. For each interval, the utilization is evaluated as the number of instructions passing through the structure divided by the interval size. If it lies above a threshold, the voltage-level of that domain is increased while if it lies below a threshold, it is decreased. Since AVF is related to utilization of structures, lowering VF only when utilization is low could have minimal impact on AVF. The constraint of a fixed threshold affects the efficiency of the algorithm. A benchmark whose structure utilization is just below the threshold for decreasing VF could have a much higher AVF than a benchmark whose utilization is slightly above the threshold but runs at a higher VF level.

Attack-Decay (AD) algorithm: This algorithm [8] tracks the issue queue occupancy changes over two neighboring periods and whenever a rapid change is detected, it triggers a *attack* phase which corresponds to a rapid scaling in VF level by a fixed number of VF levels. In other cases, the VF level is decreased slowly but continuously as long as performance drop does not exceed pre-defined thresholds. This algorithm is applied only across domains 4, 5 and 6 (see Figure 1). By taking benchmark characteristics into account, AD does not suffer the problem of fixed thresholds. But taking decisions

based on information from only two neighboring periods and allowing continuous decay (within certain bounds) might not have an optimal impact on overall AVF.

Modified Attack-Decay (ModAD) algorithm: The Modified Attack-Decay algorithm [15] was proposed to optimize on the attack phase of the original AD algorithm. In original AD, the attack phase involved changing the VF level by a fixed number of VF levels. But the fixed change leads to mismatch between the magnitude of queue occupancy change and VF change and could result in sub-optimal energy savings or performance. In ModAD, the attack phase is changed by making the number of VF levels to linearly change with respect to queue occupancy changes. Since AVF corresponds to queue occupancy, scaling VF level linearly with respect to occupancy changes should enable ModAD to optimize a structure’s AVF better than AD. But it faces the same problems as AD with respect to the performance degradation in the decay phase. Also sub-optimal VF changes due to intermittent pipeline phenomena (like L2 cache miss) could affect the AVF.

Greedy Search (Greedy) algorithm: Energy efficiency can be expressed in terms of Energy-Delay square product (ED^2). Greedy search [6] approximates the optimal ED^2 value based on the history of last two intervals. If the ED^2 was reduced over the two intervals, the VF change was correct and we optimize in same direction. Otherwise, if ED^2 was increased, the optimization direction is reversed. Each domain has a sample phase and hold phase. The sample phase is when a domain optimizes its VF level. In the hold phase, it operates in the same VF level and another domain does the sampling then. The sample phase for the domains goes in a round-robin fashion. Optimizing ED^2 which is a function of occupancy and time both of which affect AVF as well indicates that greedy search algorithm might probably optimize AVF the best. But the hold phase of each domain could lead to sub-optimal impact on AVF, since a domain operates in the same VF during the hold phase.

PI algorithm: PI [14] is an analytic online DVFS scheme based on modeling the queue-domain network. Issue queue occupancies are used as feedback signals to control the VF setting. The service rate over a specific interval is computed as

$$\mu_k = \mu_{k-1} + K_I(\bar{q}_k - q_{ref}) + K_P(\bar{q}_k - \bar{q}_{k-1})$$

where μ refers to the service rate, \bar{q}_k refers to the queue occupancy over interval k , \bar{q}_{k-1} is the queue occupancy over interval $k-1$, q_{ref} is the threshold queue occupancy which is pre-determined. K_I and K_P are control parameters (for further details, refer [14]). The new frequency f_k is computed using the relation $\mu = IPC * f$. If fine-grained VF transitions are not possible, the VF setting is approximated to the closest available level. Since PI computes the service rate independent of IPC and since the algorithm depends on the q_{ref} parameter, it could affect the AVF of structures similar to the threshold algorithm. But PI tracks benchmark characteristics better than threshold since the queue occupancy over successive intervals is tracked.

Baseline Parameters

Parameter	Value
Pipeline Width	4
Branch-Predictor	2-level
Fetch Queue/RUU/LSQ	16/128/64
ISQ Int / FP / Mem	16 / 8 / 8
Int ALUs / Mult. / Div.	4(1-cycle latency) / 2(3) / 2(20)
FP ALUs / Mult. / Div.	2(2) / 2(4) / 2(12)
L1 D-Cache	64KB, 2-way 32B block, 2 ports (2)
L1 I-Cache	32KB, 2-way 32B block (2)
L2 Unified	512 KB, 4-way 32B line-size (12)
I-TLB / D-TLB	512-entries 4-way/ 1K-entries 4-way
Mem. Lat / Baseline Freq.	200 cycles/ 4.0 GHz
Technology / Voltages	45 nm / $V_{DD} = 1.0$ V, $V_{TH} = 0.151$ V
DVFS Interval Period	20,000 cycles (at baseline frequency)

DVFS Algorithm Parameters

Parameter	Value
Threshold	Int ISQ - [4,8] FP ISQ - [3,5] Mem ISQ - [3,5]
AD / ModAD	Attack Phase - 10% change in queue occupancy
Greedy Search	Sample Phase is 4 intervals
PI	$K_I=0.6$ $K_P = 0.2$, Int ISQ $q_{ref}=6$ entries, FP and Mem ISQ $q_{ref} = 4$ entries

Table 1: Simulation parameters.

4. SIMULATION SETUP

Table 1 gives the simulation configuration and DVFS algorithm parameters used in our experiments. The MCD environment was based on the simulation model developed in [12]. The model uses Wattch [2] to compute both static and dynamic power. The framework was modified to compute the AVF of the different structures. The default configuration has the 6 asynchronous domains discussed in the pipeline model in section 2.2. Since we apply DVFS, arbiter-based asynchronous FIFOs are used for the asynchronous domains to interface [12]. All our experiments were done on 16 SPEC2000 benchmarks, 7 Int and 9 FP benchmarks, each run for 100 million instructions after fast-forwarding to an early SimPoint [9]. Parameters for the different algorithms were chosen after careful consideration based on empirical results.

Besides the DVFS algorithms, we also consider a Non-DVFS case, where all the domains are run at same frequency (4.0 GHz) through the entire simulation run. Note that we do not include a case of a single synchronous domain since the microarchitecture in that case would be different.

5. EXPERIMENTAL RESULTS

In this section, we study the AVF characteristics of Int, FP and Memory issue queues with respect to the different DVFS algorithms. While previous works [4] have looked into the performance-power tradeoffs, we show that their AVF impact cannot be ignored when studying these algorithms. Since different DVFS algorithms have varying behavior, the performance-AVF and power-AVF tradeoffs involved with the different algorithms is analyzed as well. Since the raw error rate changes with voltage-scaling, we study the DVFS impact on the overall FIT rate. All the results presented in this section are normalized to the Non-DVFS case. Also some of the SPECInt benchmarks did not use the FP ISQ and hence had zero AVF. Hence these benchmarks appear with zero value in the FP ISQ graphs.

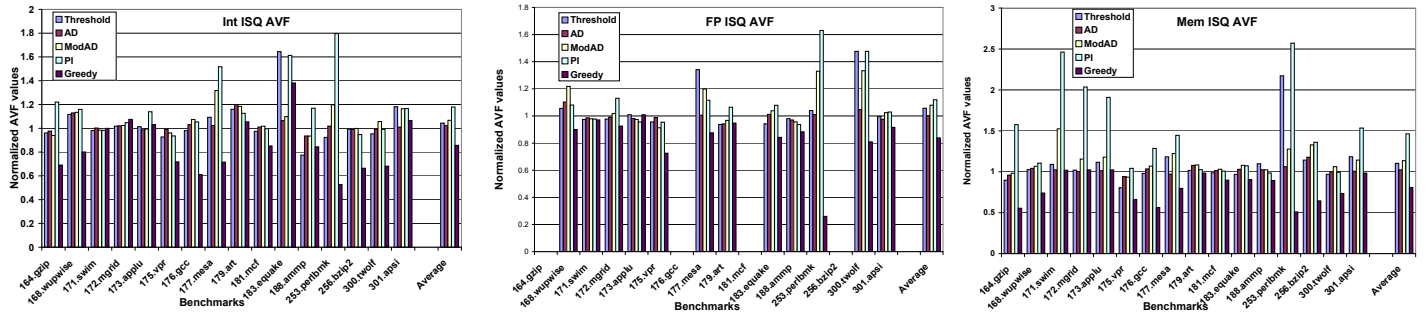


Figure 2: Impact of different DVFS algorithms on issue queue's AVF.

5.1 AVF Variation across DVFS algorithms

Figure 2 shows the percentage AVF variation across the different DVFS algorithms for the Int, FP and Mem issue queues. Across the different DVFS algorithms, the AVF of structures could vary by as much as 80% (seen in Mem ISQ AVFs when applying Greedy and PI algorithms) showing that choosing the right DVFS algorithm can have significant impact on the MTTF. Further we notice that, on an average, only the Greedy algorithm does better than a Non-DVFS case. The Greedy criterion of optimizing on energy and delay have significant correlation to issue queue occupancies and thereby reduces the AVF of the structures. PI causes the biggest increase in AVF of the issue queues because inter-domain dependencies cause occupancy increase in a dependent domain while the VF level in the source domain does not scale correspondingly. Interestingly ModAD increases AVF of issue queues more than AD. This is because inter-domain dependencies cause the fixed VF level change in the attack phase of AD to recover better than ModAD. For example, in 177.mesa the decay phase causes the memory domain (incremental increase in entries and bursty issue) to operate at lowest VF-level while the Int and FP domain are dependent on its execution.

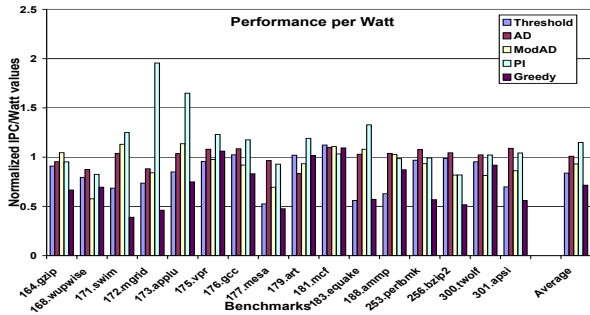


Figure 4: Performance per Watt of the DVFS algorithms.

5.2 Performance-Power tradeoffs

Previous works have studied the performance-power tradeoffs when applying DVFS. We summarize the results here and use it to contrast against those which incorporate AVF variations as well. Figure 4 shows the results. We see that the PI and AD algorithms do well in terms of increasing the performance/watt compared to the Non-DVFS case.

5.3 Performance-Vulnerability tradeoffs

A DVFS algorithm could do well in terms of reducing AVF of a structure but it might have a big performance impact as well. This would mean that entries occupy the structure longer and hence are more vulnerable to soft errors (refer section 2.1) compared to another algorithm which has high AVF but negligible performance impact. Hence we look at Vulnerability per IPC (AVF/IPC) in Figure 3. The results show that all the DVFS algorithms have worse AVF/IPC than Non-DVFS. The Greedy algorithm which had the smallest AVF now has pretty high AVF/IPC values due to the larger performance degradation. For algorithms like AD and ModAD, even though IPC is a VF scaling criterion, the performance degradation that is allowed between intervals (during the *Decay* phase) causes higher AVF/IPC .

5.4 Power-Vulnerability optimization

Though high performance increase is a primary goal, power and reliability (in terms of AVF) can no longer be ignored in current day microprocessors. Figure 5 shows the ability of DVFS algorithms to reduce both power and AVF ($Total Power * AVF$). Both the Greedy and PI algorithms do well in terms of reducing power and AVF mainly due to their ability to reduce power consumption.

5.5 Vulnerability Efficiency characterization

Vulnerability Efficiency, given by $AVF * Total Power / IPC$, characterizes the ability of a DVFS algorithm to reduce AVF and power and also have a minimal impact on performance degradation. Given the importance of reliable operation in system design, merely looking at performance per watt goals for choosing DVFS algorithms need not provide an optimal operating point. Figure 6 shows the vulnerability efficiency of the different DVFS algorithms is worse than Non-DVFS for Int, FP and Mem issue queues. For the DVFS algorithms, we clearly see that no single algorithm does consistently better than others though AD and PI provide the best average case vulnerability efficiency.

5.6 Impact of voltage scaling on overall FIT rate

Works like [13, 10] have pointed out that the raw FIT rate increases with decreasing voltages. Since DVFS algorithms scale voltages based on different criteria, they affect the raw FIT rates during those periods. Raw FIT rate variation with voltage is given by $FIT_{raw} = K.exp(-Q_{crit}/Q_s)$ where K is a constant for a particular technology generation, Q_{crit} is the critical charge for a SRAM cell and Q_s is charge collec-

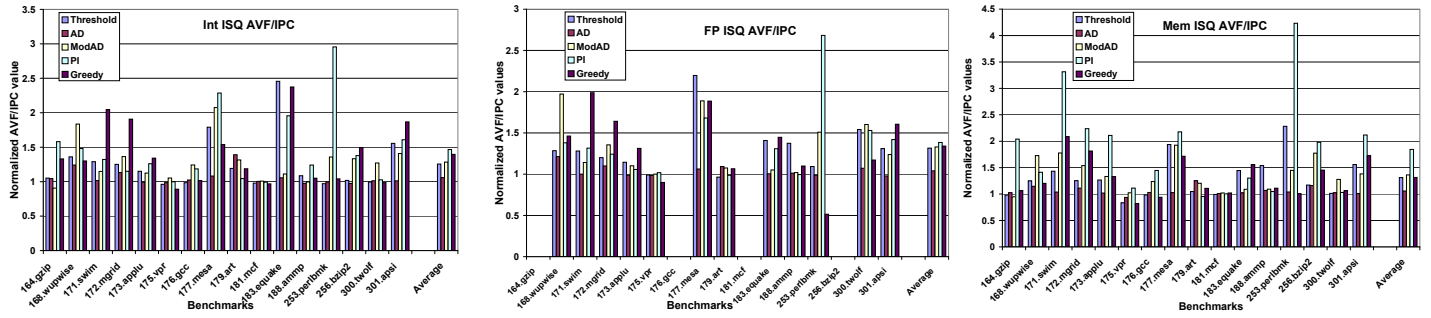


Figure 3: Performance Vulnerability tradeoffs of the DVFS algorithms.

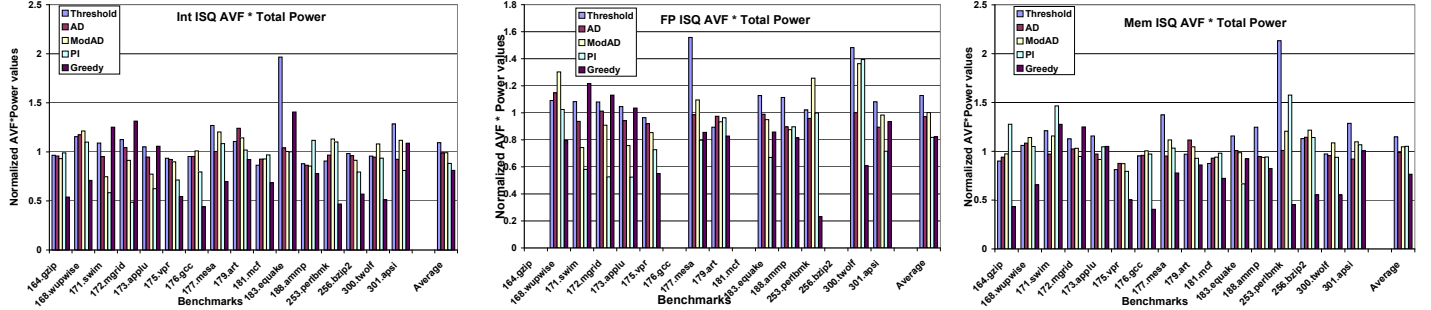


Figure 5: Power Vulnerability tradeoffs of the different DVFS algorithms.

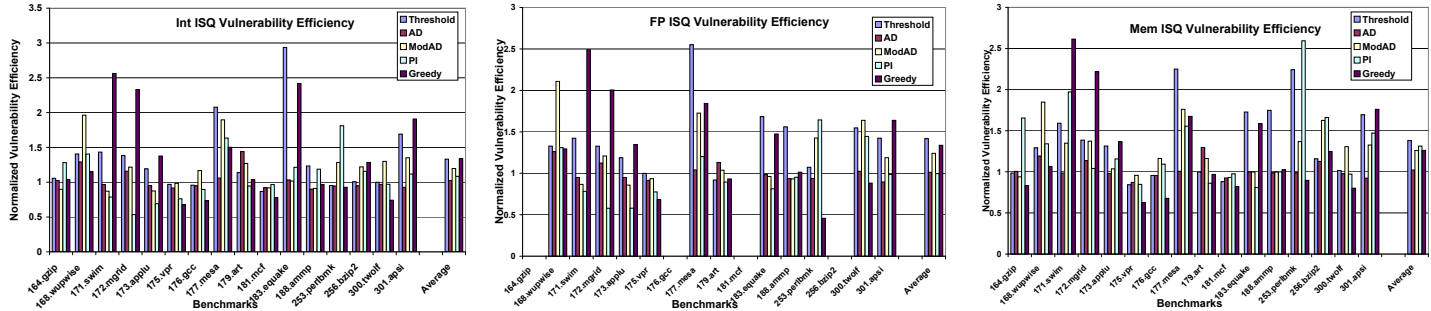


Figure 6: Vulnerability Efficiency of the different DVFS algorithms.

tion efficiency. Since both Q_{crit} and Q_s change when the supply voltage is scaled, the relationship between raw error rate and voltage is not straight-forward. [13] shows empirical results which indicates that there exists a super-linear relationship between voltage decrease and SER increase. Here we assume both a linear and exponential relation between voltage change and raw SER and compute the overall FIT rate. Figure 8 shows the overall FIT rate averaged over all benchmarks. We see that the Greedy algorithm has the lowest overall SER followed by the Non-DVFS case. Looking at Figure 8(A) which gives the time fraction spent in different voltage levels some interesting observations can be made. Since DVFS algorithms typically reduce the voltage level when occupancy reduces, we anticipate the AVF to be lower. However, lower voltages increase the raw SER. This trend where the raw SER dominates AVF in determining the overall FIT rate is seen when the threshold algorithm is run for the FP issue queues. But consider the Greedy algorithm and Non-DVFS case for Int issue queues. In this case,

even though the Non-DVFS case spends its entire execution time in the highest VF level (reduced raw error rates), it has higher overall SER due to the AVF factor dominating raw error rates.

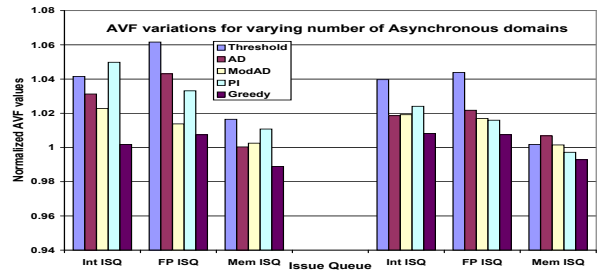


Figure 7: Average AVF variation for GALS pipeline with 4 and 5 asynchronous domains

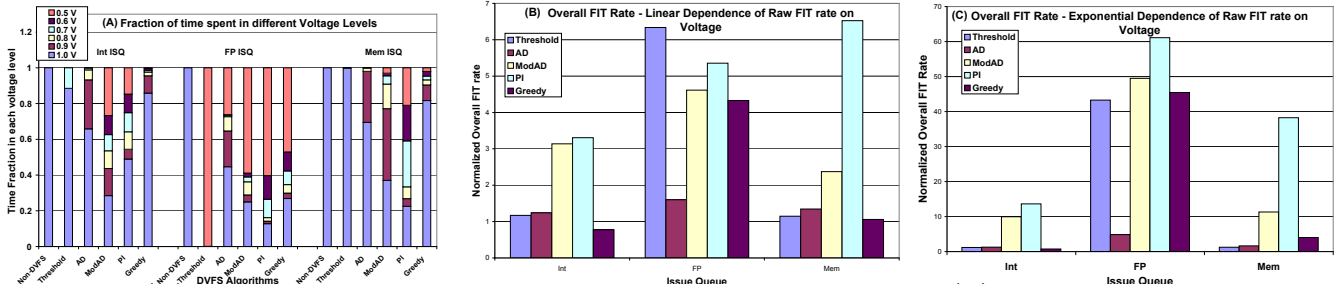


Figure 8: (A) Fraction of time each DVFS algorithm spends in a VF level. (B) Linear raw error rate voltage relationship. (C) Exponential raw error rate voltage relationship.

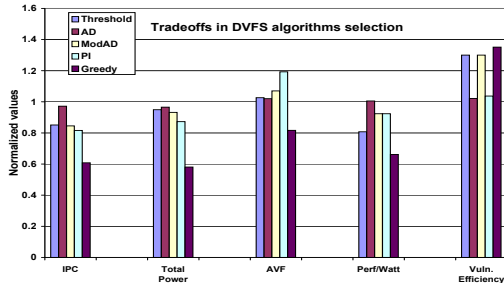


Figure 9: Performance, Power and Reliability tradeoffs provided by different DVFS algorithms

5.7 Sensitivity Analysis

Here we look at the impact of varying the number of asynchronous domains (4 and 5 asynchronous domains) on the AVF of the issue queues when applying the different DVFS algorithms. Figure 7 shows the average percentage increase in AVF across all benchmarks for the three issue queues. We find that the Non-DVFS case has the lowest AVF for Int and FP issue queues while for Mem issue queues the Greedy Search algorithm optimizes best. In general, the AVF variation between algorithms decreases with decrease in number of domains. But this also reduces the efficiency of the algorithms in applying DVFS as more domains are brought together.

6. CONCLUSION

Conventionally DVFS algorithms are studied based on their performance per watt, without considering their reliability impact. Growing concern for soft errors and the impact of DVFS algorithms on instruction occupancy in pipeline structures necessitated us to look at their AVF characteristics. Figure 9 shows the IPC, total power and AVF variations of the DVFS algorithms along with their performance per watt and vulnerability efficiency. We see that the DVFS algorithms have significantly different AVFs, thereby impacting the overall MTTF of the system. Classifying DVFS algorithms based on their vulnerability efficiency is important to meet reliability goals and such a classification leads to very different algorithm choices compared to ones based only on performance and power. We plan to extend our AVF infrastructure to look at AVF impact of DVFS on the entire system.

Acknowledgements

We would like to thank the reviewers whose comments helped improve the quality of the paper. This research was funded by NSF grants 0615097, 0621429, 0454123 and 0702617.

7. REFERENCES

- [1] A. Biswas, P. Racunas, R. Cheveresan, J. Emer, S. S. Mukherjee, and R. Rangan. Computing architectural vulnerability factors for address-based structures. In *ISCA-32*, pages 532–543, 2005.
- [2] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. *SIGARCH Comput. Archit. News*, 28(2):83–94, 2000.
- [3] X. Fu, T. Li, and J. Fortes. Sim-soda: A framework for microarchitecture reliability analysis. In *Workshop on Modeling, Benchmarking and Simulation (Held in conjunction with ISCA-33)*, 2006.
- [4] A. Iyer and D. Marculescu. Power efficiency of voltage scaling in multiple clock, multiple voltage cores. In *ICCAD '02*, pages 379–386, New York, NY, USA, 2002. ACM.
- [5] X. Li, S. V. Adve, P. Bose, and J. A. Rivers. Softarch: An architecture level tool for modeling and analyzing soft errors. In *IEEE DSN-35*, pages 496–505, 2005.
- [6] G. Magklis, P. Chaparro, J. González, and A. González. Independent front-end and back-end dynamic voltage scaling for a gals microarchitecture. In *ISLPED '06*, pages 49–54, New York, NY, USA, 2006. ACM.
- [7] S. Mukherjee, C. Weaver, J. Emer, S. Reinhardt, and T. Austin. A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor. In *MICRO-36*, pages 29–40, December 2003.
- [8] G. Semeraro, D. H. Albonesi, S. G. Dropsho, G. Magklis, S. Dwarkadas, and M. L. Scott. Dynamic frequency and voltage control for a multiple clock domain microarchitecture. In *MICRO 35*, pages 356–367, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.
- [9] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically Characterizing Large Scale Program Behavior. In *ASPLOS-10*, October 2002.
- [10] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi. Modeling the Effect of Technology Trends on Soft Error Rate of Combinational Logic. In *DSN-32*, June 2002.
- [11] E. Talpes and D. Marculescu. A critical analysis of application-adaptive multiple clock processors. In *ISLPED '03*, pages 278–281, New York, NY, USA, 2003. ACM.
- [12] E. Talpes and D. Marculescu. Toward a multiple clock/voltage island design style for power-aware processors. *IEEE Trans. Very Large Scale Integr. Syst.*, 13(5):591–603, 2005.
- [13] K. Ünlü, V. Narayanan, S. M. Cetiner, V. Degalahal, and M. J. Irwin. Neutron-induced soft error rate measurements in semiconductor memories. *Nuclear Instruments and Methods in Physics Research A*, 579:252–255, 2007.
- [14] Q. Wu, P. Juang, M. Martonosi, and D. W. Clark. Formal online methods for voltage/frequency control in multiple clock domain microprocessors. *SIGARCH Comput. Archit. News*, 32(5):248–259, 2004.
- [15] Y. Zhu, D. H. Albonesi, and A. Buyuktosunoglu. A high performance, energy efficient gals processor microarchitecture with reduced implementation complexity. In *ISPASS '05*, pages 42–53, Washington, DC, USA, 2005. IEEE Computer Society.