Error Analysis for the Support of Robust Voltage Scaling

David Roberts, Todd Austin,
David Blauww and Trevor Mudge
Advanced Computer Architecture Lab
University of Michigan, USA

{daverobe, austin, blauww, tnm}@eecs.umich.edu

Krisztián Flautner
Research and Development
ARM Ltd., 110 Fulbourn Road, Cambridge, UK

krisztian.flautner@arm.com

Abstract

Recently, a new Dynamic Voltage Scaling (DVS) scheme has been proposed that increases energy efficiency significantly by allowing the processor to operate at or slightly below the minimum supply voltage even if occasional errors result. To determine which technique can reliably and efficiently detect such failures, it is necessary to understand the manner in which digital designs fail at critical voltages. In this paper, we report hardware measurements of the failure modes of a multiplier circuit under voltage scaling. We show that even at small error rates, it is necessary to deal with multiple errors where bits are flipped from both 0 to 1 and 1 to 0. Intra- and inter-die variations make the exact nature of these flips unpredictable. This suggests that conventional single and unidirectional error detectors will not work. We conclude that the most suitable solution is a simple delay-error tolerant flip-flop that detects and corrects errors by double sampling signals.

1. Introduction

Recent work on low-power computer processor pipelines has identified that operation at voltages below the point of timing failure can yield significant energy savings. Razor [1] is one such example. By speculating on circuit propagation delays, aggressive voltage scaling below conventional safety margins may cause circuit-timing errors, which are recovered using the proposed pipeline design of [1] that guarantees forward progress.

The trends are towards increasing intra- and inter-die process variation, increasing noise susceptibility and lower reliability. Many of these effects are counteracted with more padding in the operating voltage, which in turn increases the energy consumption of designs.

Since operating margins are used to make the circuits' operation safe under a wide variety of conditions, reducing the margins will cause failures under some circumstances. Fault detection and correction techniques can be used to find the point where some or all operating margins are eliminated at run-time. However, not all error-correction codes can detect the resulting class of errors. To understand the error correction requirements better, we report on a set of experiments to measure the type and frequency of errors when scaling the voltage of a set of FPGA multipliers. Although real microprocessors often employ pipelined structures to increase throughput, our experiments, which model a single stage in a pipeline whose delay is data dependent, are sufficient to enable a comparison between candidate fault-tolerance techniques.

As we will show the experiments illustrate the effects of intra- and inter-die variation and motivate the needs for alternative approaches to error correction than is cost-effectively attainable using conventional error correcting codes. In particular, our work shows that faults induced by low-voltage operation involve multiple bidirectional output faults even at low error rates. Conventional single error and unidirectional fault detection techniques are therefore ineffective under such conditions. More complex codes that can deal with multiple bidirectional errors are not practical.

This paper is organized as follows. Section 2 discusses related background material. In Section 3, the experimental set-up is explained along with details of the test procedures used. Results are presented in Section 4 and analyzed in Section 5. Section 6 concludes this paper and discusses implications of the error results.

2. Background and related work

In this section we will survey some common proposals for error detection and comment on the kind of coverage that they offer.

In [2], three fault-secure multiplier designs are presented and compared. These are representative of conventional fault-tolerant design and can detect single faults, but they do not provide the multiple bidirectional fault detection we require.

Proceedings of the Sixth International Symposium on Quality Electronic Design (ISQED'05)
0-7695-2301-3/05 $ 20.00 IEEE

A ripple-carry adder with fine-grained adaptive voltage scaling is presented in [3]. Although effective for energy saving, the design cannot tolerate errors, and other adder architectures (e.g., Kogge-Stone) provide higher-performing alternatives.

In [4] Lo presents floating-point arithmetic algorithms for residue and Berger encoded operands. Residue codes can detect all single-bit errors, and Berger codes can detect all unidirectional errors [5]. The former provide a check using modulo arithmetic. Berger code-words include a count of the number of 0's in a word, adding ceil($\log_2$ n) extra bits. Circuits were presented for these techniques, whose area overheads were estimated at around 8% and 46% respectively for a single-precision floating-point multiplier—the large overhead for Berger codes is due to their being unsuitable for modeling arithmetic operations. For voltage scaling applications, the checker circuits must be guaranteed to fail at lower voltages than the main logic. In addition, they do not provide the multiple bidirectional fault detection we require.

Mitra and McCluskey [6] analyze concurrent error detection techniques with regard to common-mode failure vulnerability. Their results indicate that diverse duplex systems provide the best protection against multiple failures. Parity prediction, Berger and Bose-Lin codes have a greater area overhead and less effective protection than this full-redundancy technique. Bose-Lin codes are similar to Berger codes except that they detect t-bit unidirectional errors in the code-word. However, redundancy provides no guarantee against multiple failures at critical voltages.

The NanoBox [7] implements logic functions as truth tables containing error correcting codes. Faulty logic is corrected at the function output. Using triple-mode redundancy allows high fault tolerance for a 1.9x area overhead. Again, redundancy provides no guarantee against multiple failures at critical voltages.

On-chip critical path emulation [8] is a DVS technique whereby logic and interconnect delays for the critical path are estimated. While accurately modelling the circuit critical path, this scheme cannot take advantage of data-dependent variation of the actual critical path at the point of execution. Critical path emulation is a viable technique for removing some of the headroom in voltage-setting but it cannot eliminate all of it. Replicating a critical path also brings up the question of how closely a replica can track the original when on-chip silicon variation is high.

Figure 1 shows a simple delay-error tolerant Razor flip-flop that detects and corrects errors by double sampling signals. The standard positive edge triggered flip-flop is augmented with a shadow latch which samples at the negative clock edge. Timing errors are detected by comparing the main flip-flop data with that of the shadow latch. Clearly it detects multiple bidirectional errors provided the timing for the shadow latch is guaranteed to be correct. Because it is clocked much later than the main flip-flop this requirement is usually straightforward to satisfy. The Razor flip-flop has a relatively low overhead—a few percent of total area in a pipelined prototype [1].

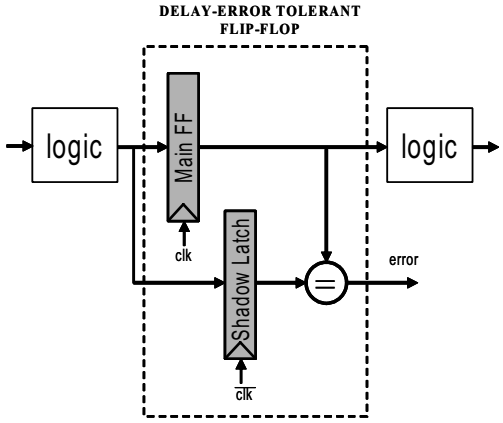

Figure 1: Delay-error tolerant or Razor flip-flop

The delay-error tolerant flip-flop is not universally better than conventional error detection. It is not a solution to single event upsets caused by energetic particle strikes, and it is not a solution to errors resulting from component failures. Traditional checkers are more suitable. The delay-error tolerant flip-flop is most suited to cases that result in delay errors. Examples arise from voltage scaling (the path to the main flip-flop fails), temperature and environmental effects, and transient noise. They are characterized by multiple bidirectional errors.

3. Experimental methodology

Analysis in [1] recorded error rates for a multiplier block in a Xilinx XC2V250 FPGA [9][10]. Two half-speed multipliers were used to check the result from a fast multiplier when pseudo-random input vectors were supplied by a Linear Feedback Shift Register (LFSR). The error rates were recorded and energy savings were measured relative to estimated voltage margins. At a 1.3% error rate, an energy saving of 33% was obtained. The point to note is that the 1.3% error rate is considerably past the point of first error, thus detection will require a technique that identifies multiple bidirectional errors. The experiment is explored further in this paper. The emphasis here is on the failure modes as the voltage is reduced.

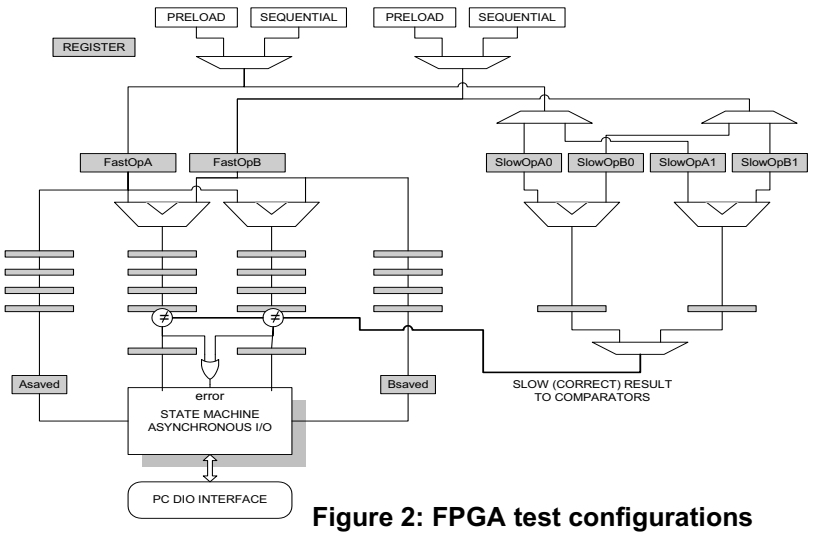

Figure 2: FPGA test configurations

Figure 2 illustrates the FPGA configuration used for the experiment. Two 'fast' 17x17 bit multipliers running at the full clock speed (90 MHz) are provided with the same pair of operands. The operands themselves alternate between a pseudo-random 'preload' value from an LFSR and the next pair of 'sequential' test vectors. The sequential values represent all possible combinations of operands, as illustrated in Table 1. The total number of test vectors is given by:

$$n_{MAX} = \left(\frac{2^{17}+1}{2}\right) \cdot 2^{17} = 8.59 \cdot 10^{9}$$

Their results are fed into a sequence of four latches. As the supply voltage is reduced errors will appear and be recorded in the latches. In parallel with these units, two 'slow' multipliers are alternately selected to calculate the product of each pair of input operands. The slow multipliers should continue to give correct results after the fast ones fail. Preload values are not checked so the slow multipliers have four times as long as the fast ones for the result to propagate to the output pins. The purpose of the preload is to mitigate the effects of previous state dependency on error rate statistics. Whenever there is a difference between a fast and slow multiplier output after the four-cycle delay, an error signal is asserted and FPGA state is logged on a PC. Operands are also latched so that they may be recorded during logging.

Table 1: Sequence of multiplier operands

Sequence	Operand A	Operand B
1	0	0
2	1	0
3	1	1
4	2	0
5	2	1
6	2	2
7	3	0
8	3	1
9	3	2
10	3	3
...		
n_{MAX}	$2^{17}-1$	$2^{17}-1$

A finite state machine is employed to perform asynchronous transfer of data to a PC-based data logging unit when an error occurs. The data recorded when an error occurs includes the input operands producing the incorrect result, the number of errors observed so far, number of elapsed clock cycles and the outputs from all fast multipliers. In addition, the 'correct' result from the selected slow multiplier is recorded and compared with the product calculated on the PC. This ensures that the slow product is not the source of error.

Care was taken during synthesis to ensure that circuits involved in error checking were not on the critical path for any part of the experiment. This was achieved through the use of timing constraints as well as retiming within the design. Table 2 below shows the synthesized static timings used for the tests (Mn indicates which 'fast' multiplier is involved). The comparator, slow multiplier and error-

counter paths were sufficiently short to cover a useful voltage range. By using two operating modes it was possible to measure error rates beyond the point at which the data-logging state machine failed. In full-logging mode, the state machine was used to record all errors at low voltage. In statistics-only mode which fails at 1.02V, the values of the error counters for each multiplier were logged at the end of each test at nominal voltage. In the results section, error rates recorded in each mode are shown to be consistent.

Path	Max. delay
Fast multiplier M0 operands to output	7.690 ns
Fast multiplier M1 operands to output	7.385 ns
Error comparator M0 to state machine	5.192 ns
Error comparator M1 to state machine	5.973 ns
Error comparator M0 to error counter	3.948 ns
Error comparator M1 to error counter	3.670 ns

Table 2: Static timing analysis

FPGA routing fabric delay was minimized relative to multiplier delay to ensure that most timing errors were occurring within the multiplier blocks. Approximately 70% of total delay was due to the multiplier blocks. The experiment was reduced to only 2 fast multipliers to achieve this goal since routing delay became excessive with more multipliers.

Results were gathered for a set of different FPGA core voltages below the nominal 1.5V operating point. A test at a given voltage recorded errors for all input combinations. Each test was re-run in statistics mode and compared for consistency. The tests were then repeated on three different FPGA chips.

4. Circuit Timing Error Analysis

4.1 Intra-Die Variation

Figure 3, Figure 4 and Figure 5 show the relationship between voltage and error rate for each FPGA at a range of voltages. MUL0 and MUL1 were placed in physically adjacent multipliers on the FPGA by the Xilinx synthesis tool.

Data points represent error rates for the full set of test inputs at each voltage level. Mul0 and Mul1 are the two fast multipliers. Lines Mul0F and Mul1F (F = full) represent results from a separate experiment where all the error data is logged. Note that full error logging is stopped at around 1.1V since the data transfers were time consuming and we only require error analysis at low rates.

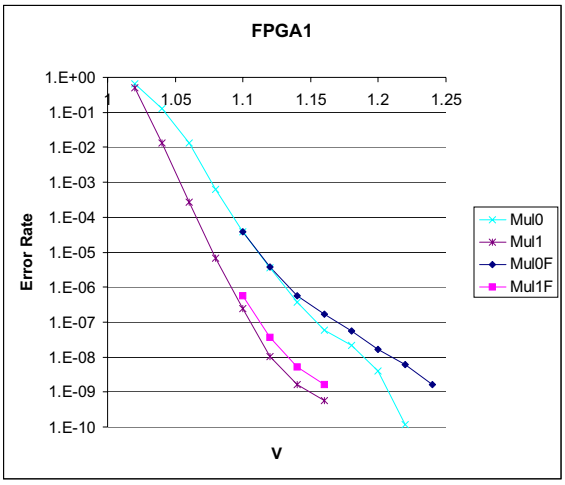

Figure 3: Error rates for FPGA 1

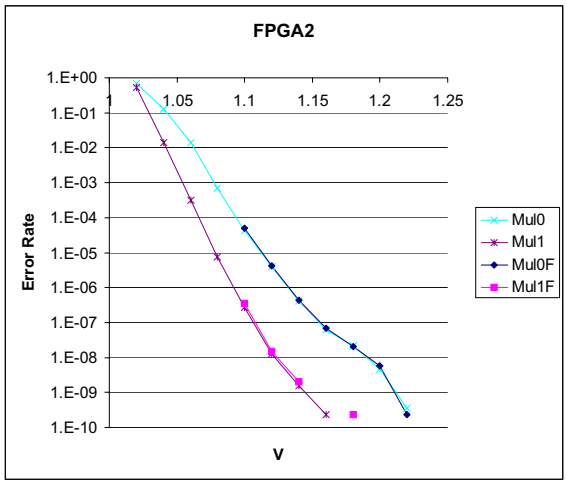

Figure 4 Error rates for FPGA 2

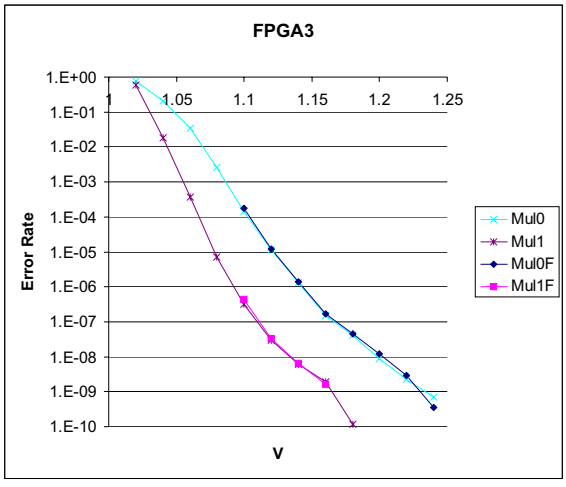

Figure 5 Error rates for FPGA 3

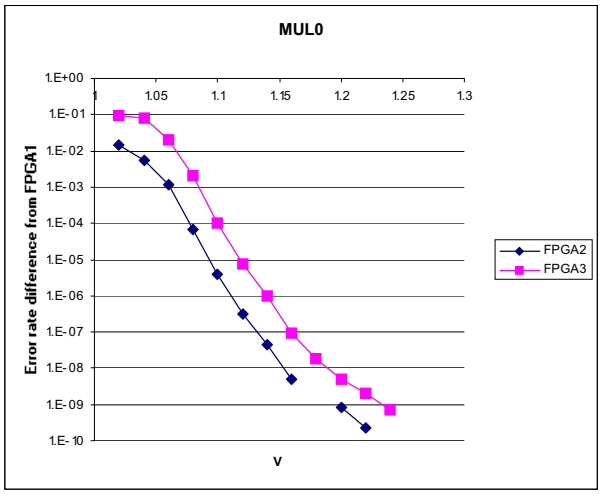

Figure 6: Error rate relative to FPGA1 for MUL0

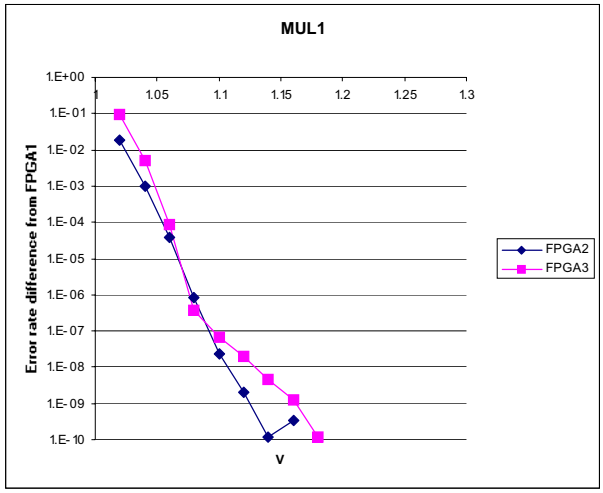

Figure 7: Error rate relative to FPGA1 for MUL1

4.2 Inter-Die Variation

Figure 6 and Figure 7 show the absolute differences in error rate between FPGAs 2, 3 and FPGA1, which has the lowest error rate. Figure 6 shows error rate differences for MUL0 and Figure 7 for MUL1.

4.3 Error Characteristics

Figure 8 categorizes bit flips by direction (0 to 1 indicated by 01 or 1 to 0 indicated by 10) for all FPGAs and multipliers. F denotes the FPGA and M denotes the multiplier. Each point on Figure 8 represents the total number of incorrect bits observed on the multiplier output for the full

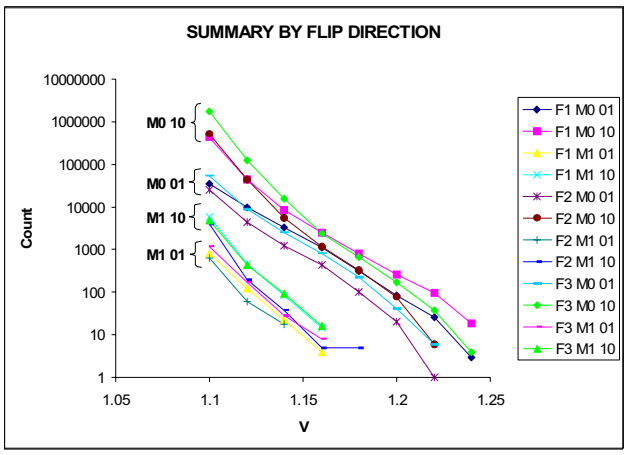

Figure 8: Incorrect bit counts

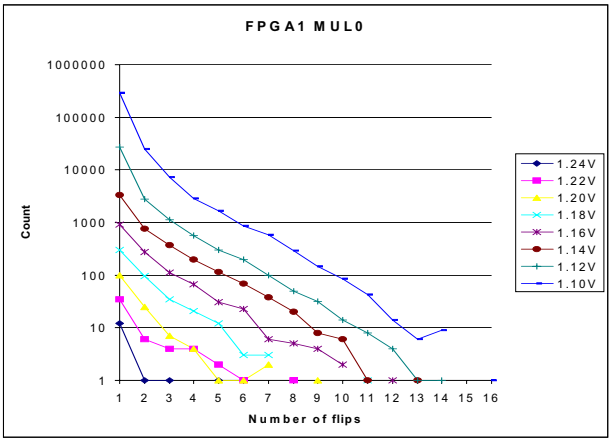

Figure 9: Count of vectors with N incorrect bits

set of input operands tested at that voltage. Figure 9 and Figure 10 show the number of input vectors which caused N incorrect bits in the multiplier output. Due to the page limit, only FPGA1 MUL0 is shown.

5. Analysis

The intra-die variation shown in Figure 3, Figure 4 and Figure 5 show both the final error counts generated in statistics mode, and error counts obtained from full logging mode. There is an exponential increase in error rate as voltage is reduced. This characteristic was also seen in [1]. MUL0 fails earliest at around 1.24V in all cases and also has the highest error rates. MUL1 fails at around 1.15V. This difference is due to a 0.305 ns variation in maximum routing delay (see Table 2).

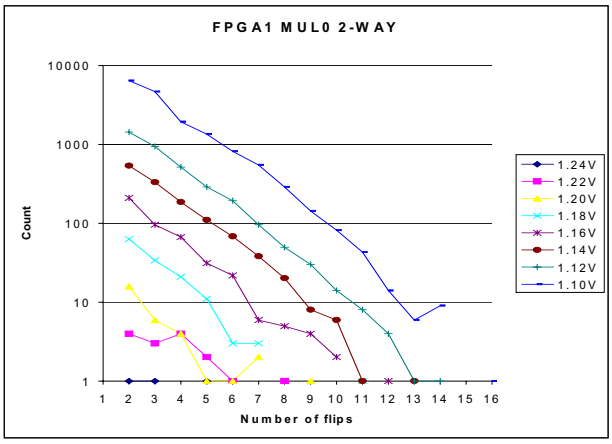

Figure 10: Count of vectors containing at least 2 incorrect bits, one flipped from 0->1 and the other from 1->0

Differences in error rates between FPGAs provide a more accurate picture of process variation since the same statically characterized routing delays are present on each chip. There is an order of 10 magnitude difference between FPGA2 and FPGA3 for MUL0. There is an exponentially increasing difference between these multipliers and MUL0 as voltage is reduced. For MUL1 however, there is no smooth exponential relationship. This shows that the voltage corresponding to a given error rate is not deterministic, even if the input data set is consistent. Therefore, for low error coverage techniques, process variation margins must still be added to the supply voltage. High error coverage techniques such as [1] allow the complete removal of these margins.

Regarding the 2-way bit flip occurrences, it should be noted that at 1.1V the error-rate is less than 0.004% (from Figure 10). Ten input vector combinations cause errors where a total of 14 output bits require correction. Some of these bits flip from 0 to 1 and others from 1 to 0. This is significant because it rules out the use of many single error detection codes that also correct unidirectional errors. Hamming and Berger codes are examples. To correct multiple bidirectional errors using traditional error correction techniques requires logic that is significantly more complex and expensive. The error rate that corresponds to the minimum power dissipation occurs below 1.1V, where the error rates are between 0.01% and 2.67% [1].

6. Conclusions

Our experiments show that when pushing the limits of voltage scaling, multiple-bit errors often occur. The delay-error tolerant flip-flop technique is one method for successfully dealing with this situation. This was successfully employed it in the Razor pipeline described in [1]. Moreover, it can take advantage of data-dependent changes in the critical path, work in the presence of high on-chip silicon variation, and may reduce the need for overly accurate voltage regulator by being able to tolerate a sustained non-zero error rate. Our studies in this paper show that it is a cost effective solution when compared to other error detection techniques.

Acknowledgements: This work was supported by grants from NSF, ARM Ltd., and GSRC.

7. References

[1] D. Ernst, N. S. Kim, S. Das, S. Pant, T. Pham, R. Rao, C. Ziesler, D. Blaauw, T. Austin, T. Mudge, and K. Flautner. "Razor: A low-power pipeline based on circuit-level timing speculation." In *Proc. of the 36th Ann. Int'l Symp. on Microarchitecture (MICRO-36)*, Dec. 2003.

[2] K. Papadomanolakis, A. Kakarountas, V. Kokkinos, N. Sklavos, and C. Goutis, "The Effect of Fault Secureness in Low Power Multiplier Designs." In *Proc. of 2001 IEEE International Workshop on Power And Timing Modeling, Optimization and Simulation (PATMOS'01)*, Yverdon-Les-Bains, Switzerland, pp. 10.3, Sept. 2001.

[3] H. Suzuki, W. Jeong, and K. Roy, "Low Power Adder with Adaptive Supply Voltage." In *Proc. of the 21st Int'l Conf. on Computer Design (ICCD)*, 2003.

[4] J.-C. Lo, "Reliable Floating-Point Arithmetic Algorithms for Error-Coded Operands." In *IEEE Trans. on Computers,* vol. 43, no. 4, April 1994.

[5] T. Rao, "Error Coding for Arithmetic Processors." *Academic Press*, 1974.

[6] S. Mitra, and E. McCluskey, "Which concurrent error detection scheme to choose?" In *Proceedings of the 2000 IEEE International Test Conference*

[7] A. Kleinosowski et al., "The NanoBox: A Self-Correcting Logic Block for Emerging Process Technologies with High Defect Rates," In *Laboratory for Advanced Research in Computing Technology and Compilers Technical Report No. ARCTiC 03-02*, June, 2003.

[8] M. Elgebaly, M. Sachdev, "Efficient Adaptive Voltage Scaling System Through On-Chip Critical Path Emulation," in *Proc. of the Int'l Symp. on Low Power Electronics and Design (ISLPED), 2004*.

[9] Xilinx DS031: "Virtex-II Platform FPGAs: Complete Data Sheet." March, 2004.

[10] Xilinx UG002: "Virtex-II Platform FPGA User Guide," April 2004.