

Analysis of Soft Error Mitigation Techniques for Register Files in IBM Cu-08 90nm Technology

Riaz Naseer, Rashed Zafar Bhatti, Jeff Draper

Information Sciences Institute
University of Southern California
Marina Del Rey, CA 90292 USA

{naseer@ISI.EDU, bhatti@ISI.EDU, draper@ISI.EDU}

Abstract— Soft errors are a major reliability concern for today’s nanometer technologies. The errors in register files in Application Specific Integrated Circuits (ASIC) can quickly spread to various parts of the system and result in data corruption which may go unnoticed. Single Error Correction (SEC) Hamming code and Triple Modular Redundancy (TMR) provide a high-level mitigation solution for soft errors. The experimental results presented in this work for a 64-bit wide, 32-word entry register file, synthesized for IBM Cu-08 (90nm) standard cell ASIC technology, show that TMR is a better solution for latency sensitive ASIC applications. This technique increases the read access time by only 17% but incurs 204% area penalty. Whereas SEC applied on the 64-bit word size incurs 129% increase in read access time with area penalty only about 22%. This large read access penalty for applying SEC on wide words makes it necessary to partition the word into smaller block sizes before applying SEC. The design space explored in this work, for various combinations of word and block sizes, shows that Hamming code provides a reasonable spectrum of choice to trade-off between area and latency of fault-tolerant register files.

I. INTRODUCTION

Technology scaling has made today’s designs much more susceptible to soft errors. The ever decreasing supply voltages and nodal capacitances (required for constraining the power and making circuit transition faster) results in reduced critical charge (Q_{crit}) required to upset a node in digital circuits. The problem becomes more acute for aircraft and space electronics where high-energy neutrons at higher altitudes and heavy ions in space are more abundant. Because of the prevailing predictions that soft error rate is increasing exponentially [1], there is a growing trend in the community to adopt soft error rate as a design parameter along with speed, area and power requirements. Furthermore, for the high-end server market, networking switches and database applications, the reliability of the computing machine is even more important along with its performance.

Single Event Upsets (SEU) have long been a reliability concern for space electronics. There are a number of solutions employed for mitigating radiation effects ranging from specialized processes to architecture and circuit-level techniques. The micro-architectural solutions to mitigate SEU employ fault tolerant techniques such as Triple Modular Redundancy (TMR) and Error Detection and Correction (EDAC) codes [2] for regular memory structures. Specialized storage structures have been presented as an alternative to mitigate SEU as well as Single Event Transients (SET) [3]-[8].

Register files are an integral part of any microprocessor architecture. Although the overall area of the register files is small, they are the most frequently accessed part of any microprocessor. Any error in the register file can quickly spread throughout the system and result in silent data corruption. Although research in the computer engineering community has focused on mitigating soft errors from main memory [9], regular cache structures [10]-[11] and storage elements in the data path [1][8], there is little published data available about SEU protection of register files. Recently a high level architectural study in [12] proposed a solution to mitigate SEU from the register files by using register duplication from the unused registers in combination with Error Correcting Codes (ECC). Even though the proposed solution will require some form of error detection and correction scheme, no actual circuit level performance data was presented. To the best of our knowledge the work in [13] is the first available data that compares the area and performance for SEU mitigation in register files. Yet this work has been done on Field Programmable Gate Array (FPGA) technology and does not directly relate to the standard cell ASIC design methodology.

The work described in this paper explores trade-offs for micro-architecture level soft error mitigation solutions, e.g., TMR and Hamming codes, for register files in Application Specific Integrated Circuits (ASIC). We have implemented register files with 2 read ports and 1 write port such as used in the reduced instruction set (RISC) processor of the DIVA

architecture [14]. The register files and Single Error Correction (SEC) Hamming encoder and decoder have been synthesized using the IBM Cu-08 90nm standard cell library. We have implemented various possible block size configurations of the SEC code for the 32-bit and 64-bit wide register files. The results show that TMR and (7, 4) Hamming code are better solutions for performance-hungry designs. On the other hand, Hamming code applied to various block lengths such as (71, 64), (38, 32), (21, 16) and (12, 8) offer area and performance trade-offs depending on the need of the ASIC design.

The rest of the paper is organized as follows. Section II summarizes previous work. Section III presents a probability analysis for TMR and Hamming code. In section IV we describe implementation scenarios for TMR and Hamming code. Section V presents and discusses implementation results, and in Section VI we conclude the paper.

II. PREVIOUS WORK

Soft error mitigation techniques involve redundancy mechanisms to obtain fault tolerance. The simplest redundancy method, Triple Modular Redundancy (TMR), replicates the hardware three times and then a majority voting logic is used to ignore any corrupt value [2]. Most modern microprocessors employ some form of Error Detection and Correction (EDAC) technique for protecting system memory and caches from soft errors [9]-[11].

The proposed solution in [12] for increasing register file resilience against transition errors in superscalar processor architectures makes use of the available physical registers which are not currently mapped to logical/architectural registers by the reservation station. (In superscalar architectures the number of physical registers is much larger than the architectural registers. Not all of the physical registers are used during a normal execution of an application.) This solution requires the use of ECC such as parity or Single Error Correction – Double Error Detection (SEC-DED) in addition to the replication of data in unused registers to maintain redundant copies (named as copy registers). This scheme requires modification not only to the register file interface, but also to the register renaming circuit (RRC) and the reservation station (RS). In case an error is detected in a register being read, the uncorrupted data can be forwarded from the copy register if that is available. In case an error cannot be corrected then the instruction is squashed and the Reservation Station is used to rollback the registers involved (data register being read and the copy register).

The work in [13] compares the area and performance penalties for Hamming code and TMR schemes for reconfigurable logic. In a case study, a group of 5 registers with 8-bit and 16-bit data widths has been considered as a register file. The synthesis results for Xilinx and Altera FPGAs does not show significant differences between the Hamming code and TMR implementations. These results can be attributed to the inherent large delay values in reconfigurable logic technologies and due to the fact that a

very small number of registers with smaller word-widths has been treated as a register file.

In this work, we consider state-of-the-art IBM Cu-08 monolithic register files which are used in practical ASIC systems such as DIVA [14]. The considered register files are 32 words deep which is typical for RISC architectures. In addition to the 32-bit data width, we have also considered a 64-bit data width so that trends can be drawn for future technologies.

III. ERROR PROBABILITY ANALYSIS OF SOFT ERROR MITIGATION TECHNIQUES

For standard cell based ASIC designs, micro-architectural techniques such as TMR and EDAC codes are a possible solution for soft error mitigation. Register files are usually a part of the high-speed pipeline so EDAC codes which require complex encoding and decoding techniques may not be suitable for these applications. Furthermore, the work in [15] indicates that there is very little gain (2%) in fault tolerance for on-chip caches if we increase the protection from SEC-DED to Double Error Correct-Triple Error Detect (DEC-TED) while the overhead for that extra 2% gain is large. Comparing the register file area to overall chip area, this gain will be even less likely. In addition, double error detection will require architectural changes to rollback and replay an instruction in the case of a double error. A separate analysis needs to be performed to assess the improvement in error probability and the cost of this complex control for the double error detection case. We have considered only the SEC Hamming code in this study. This code will correct all single bit errors but will pass the double error as it is and will result in a system error. It is important to note that the probability of double errors is very low as compared to single bit errors. We have also performed a probability analysis in the following section which describes the overall improvement in bit-error rate for Hamming codes and TMR. The analysis of SEC-DED modified Hamming code can be performed in a similar manner and it is expected that results will not be significantly different for that case. Therefore, we will be focusing only on TMR and SEC Hamming code for register files.

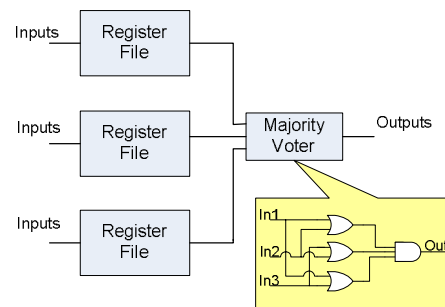


Figure 1. Triple Modular Redundancy

A. Error Probability Analysis for TMR

Triple Modular Redundancy (TMR) is a commonly applied radiation hardening technique [2]. In this technique, the hardware that is to be made radiation tolerant is tripled and a majority vote is used at the output to forward the correct results. The scheme can be applied from the system level down to the component level. Fig. 1 shows the general architecture for a TMR implementation of the register file. The majority voter consists of a simple two level OR-AND combinational logic.

Before looking into the application of this technique at the system level it is important to quantify its fault tolerance benefits as applied to w -bit wide storage words. Abstracting all parameters causing SEU to a probability ϵ_e of a single bit being upset, the probability of error occurrence in w -bit wide word is given by an independently, identically distributed (iid) w long sequence of Bernoulli random variables:

$$P_{\text{word-unprotected}} = 1 - (1 - \epsilon_e)^w \quad (1)$$

When TMR is applied for a single bit, it will reduce the bit error probability. The probability analysis for this shows that the resulting bit error probability ϵ_{TMR} reduces to:

$$\epsilon_{\text{TMR}} = 3\epsilon_e^2(1 - \epsilon_e) + \epsilon_e^3 \quad (2)$$

Therefore for a w -bit word the probability of error for TMR reduces to:

$$P_{\text{word-TMR}} = 1 - (1 - 3\epsilon_e^2(1 - \epsilon_e) + \epsilon_e^3)^w \quad (3)$$

B. Error Probability Analysis for Hamming code

The second soft error mitigation technique considered for fault tolerant register files in this paper is Hamming code. Hamming codes can be applied to various block sizes. Larger block sizes require less redundant bits and save area for required additional memory cells at the cost of reduction in performance and fault tolerance. Generally error correcting linear block codes are defined by three parameters: (1) k , the number of information bits in the block to which the code is applied, (2) n , the number of bits in the coded word and (3) d , the minimum Hamming distance between any two valid code words. Hamming code can be represented in two notations (1) $Ham(n, k)$, where the value of d is not specifically mentioned, and (2) $Ham(n, k, d)$ where the value of d is specifically mentioned. In this paper, the 1st notation is used where $d = 3$ is assumed for all SEC Hamming codes considered. Similar to the TMR technique, the achievable fault tolerance with Hamming code can also be quantified in terms of the probability ϵ_e of a single bit being upset. For this, consider an SEC code applied to k -bit block that produces a code word of $n = (k + \log_2(k+1))$ bits. Since Hamming codes can recover a single bit error in an n -bit block, the probability of having 2 or more errors in a code word of size n -bits is given by:

$$P_{\text{cw}} = 1 - \{(1 - \epsilon_e)^n + n\epsilon_e(1 - \epsilon_e)^{(n-1)}\} \quad (4)$$

Given this the overall error probability of a w -bit word, i.e., error probability in w/k blocks with SEC Hamming code can be found as:

$$P_{\text{word-HAM}} = 1 - \{(1 - \epsilon_e)^n + n\epsilon_e(1 - \epsilon_e)^{(n-1)}\}^{(w/k)} \quad (5)$$

Equation (3) and (5) can be used for comparison purposes of achievable fault tolerance using TMR and Hamming codes. Word level error probability achieved for various bit-error probability values are shown in Fig. 2 for quick reference. Notice that there is a huge differential as we go from unprotected to any protection scheme, where as there is very small difference for word error probabilities between TMR and Ham (7, 4). Note also that the difference between unprotected and protected word error probabilities decreases as the bit error probability increases.

IV. IMPLEMENTATION SCENARIOS

For a triple modular redundancy implementation of a register file, there are two possible scenarios. A direct implementation of TMR will involve three identical register files as shown in Fig. 1 along with the voter placed at their output before passing the output to the next stage. In a second scenario, only the storage cells can be tripled instead of tripling the whole register files. This implementation exploits the fact that the address decoding is common among all three register files. This results in saving part of the area for TMR while slightly increasing the latency because of the large word-line load for the address decoding circuitry.

Register arrays are an integral part of today's ASIC design flow. Most modern ASIC libraries provide highly condensed and efficient compile-able monolithic register arrays. Therefore register files can be synthesized from these compile-able register arrays instead of synthesizing using basic standard cells. In the available IBM Cu-08 (90nm) cell library, the width of the compile-able register arrays is limited, and hence the word width cannot be made 96-bit wide for the second scenario considered for TMR implementation of 32-bit registers. Therefore, three identical register files are implemented in this experiment.

The implementation of Hamming code requires placement of the encoder circuit at the input of the register file so that every write occurs through the encoder. Additionally, a decoder circuit is placed at the output of the

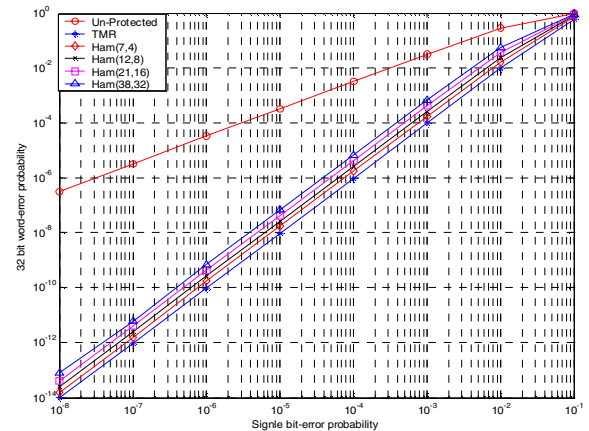


Figure 2. 32 bit word-error probability comparison curves

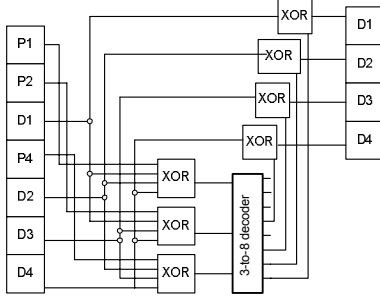


Figure 3. SEC Decoder

register file so that every read passes through the decoder circuit. The SEC code uses different parity check bits for various combinations of data bits. The parity check bits are generated using modulo-2 addition. For example, if four data bits are D_1 through D_4 , then the required three parity check bits for a (7, 4) Hamming code can be generated as follows:

$$P_1 = D_1 + D_2 + D_4 \quad (6)$$

$$P_2 = D_1 + D_3 + D_4 \quad (7)$$

$$P_4 = D_2 + D_3 + D_4 \quad (8)$$

The parity bits are labeled 1, 2 and 4 because they are placed at the power of 2 locations in the output word while data bits are spread at the remaining locations. The implementation for this encoder is simple and can be accomplished by using XOR gates for the parity check equations. The decoder circuit, on the other hand, is more complex. The decoder circuit for this example is shown in Fig. 3. The first part of the decoder circuit is similar to the encoder where a syndrome is calculated, which is a modulo-2 sum of the received parity bits and computed parity bits. If this syndrome is zero then it means that no error has occurred. If the syndrome is not zero, then it tells the location of the error bit. A syndrome decoder is needed to decode the location of error; the error bit can be corrected by performing a mod-2 sum of every bit with the output of the syndrome decoder. In this way the correction circuit can be made from 2-input XOR gates; one input of these XOR gates is the data read bit while the other bit is the output of the error location (syndrome) decoder. Note that the parity bits are not forwarded to the output and there is no need to correct if the parity bits are wrong, so not all outputs of the syndrome decoder are used and it can be optimized. For an SEC Hamming code implementation, in addition to the encoder and decoder circuits, the register file needs to be expanded to accommodate the parity check bits.

V. RESULTS AND DISCUSSION

The synthesis has been performed using Synopsys Design Compiler targeted to IBM Cu-08 (90nm) standard cell library. The register files developed consist of one write port and two read ports. The register file synthesis has been obtained by using highly dense and efficient compile-able register arrays provided with the IBM library. The encoder

and decoder for Hamming code have also been targeted to the standard cell library of the same technology.

The post synthesis area and timing results are shown in Table I, Table II and Table III. All results are normalized to protect the confidentiality of the actual data. Table I shows area for a 32-bit wide register file with no protection as a base case, and area of register files with TMR and SEC Hamming code of various block sizes. It is apparent from these results that the minimum area solution will be to implement the Hamming code on the entire word as a single block. Table II shows the results for read access latency for all the above cases. According to these results, TMR is the best solution if the latency of the register file is more important. Table III shows the increase in write setup time. Although the write path is not timing critical, yet we observe

TABLE I. AREA COMPARISON

Register File	Area (unit cells)	Encoder Area (unit cells)	Decoder/Voter Area (unit cells)	Total Area (unit cells)
32x32	1.0000	-	-	1.0000
TMR	3.0000	-	0.0362	3.0362
(7, 4)	1.6440	0.0032	0.0126	1.6599
(12, 8)	1.4352	0.0049	0.0278	1.4680
(21, 16)	1.2654	0.0145	0.0510	1.3309
(38, 32)	1.1522	0.0339	0.0921	1.2783

TABLE II. READ ACCESS LATENCY

Register File	Read Access (ns)	Decoder/Voter Delay (ns)	Net Read Latency (ns)
32x32	1.0000	-	1.0000
TMR	1.0000	0.1860	1.1860
(7, 4)	1.0698	0.5271	1.5969
(12, 8)	1.0465	0.7132	1.7597
(21, 16)	1.0310	0.8915	1.9225
(38, 32)	1.0155	1.1163	2.1318

TABLE III. WRITE SETUP TIME

Register File	Write Setup Time (ns)	Encoder Delay (ns)	Net Write Latency (ns)
32x32	1.0000	-	1.0000
TMR	1.0000	-	1.0000
(7, 4)	1.1579	0.7105	1.8684
(12, 8)	1.0526	1.1316	2.1842
(21, 16)	1.0263	1.5789	2.6053
(38, 32)	1.0263	2.1842	3.2105

TABLE IV. READ ACCESS FOR 64-BIT REG FILE

Register File	Read Access (ns)	Decoder/Voter Delay (ns)	Net Read Latency (ns)	%age increase in Area
32x64	1.0000	-	1.0000	-
TMR	1.0000	0.1702	1.1702	203.90
(38, 32)	1.0355	1.0213	2.0567	111.15
(71, 64)	1.0213	1.2695	2.2908	21.78

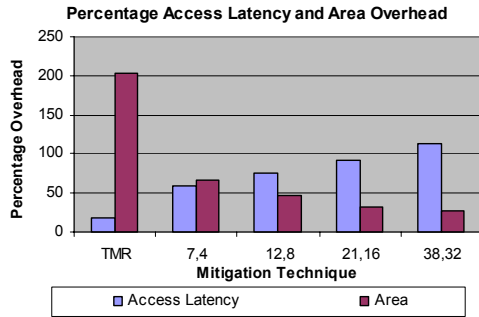


Figure 4. Area and Latency Comparison for TMR and SEC for various block sizes

from Table III that even the (12, 8) Hamming encoder doubles the setup time requirement. Table IV shows the increase in read access latency and percentage area increase for TMR and SEC Hamming codes applied to a 64-bit wide register file. From the results it is clear that applying Hamming code on larger data widths such as 64-bit and 32-bit words as a single block have much larger latency which cannot be tolerated in register files used in extremely pipelined high-speed microprocessors. The chart in Fig. 4 provides a side-by-side area and read access latency comparison for a 32-bit wide register file with TMR and Hamming code implementation of different block sizes. From the chart various trade-off points can clearly be seen. A compromise between area and latency can be achieved if SEC code is applied to 4-bit blocks of the data word. An SEC Hamming code applied on a block size of 4-bits saves around 45% area against TMR, incurring around 34% degradation in speed compared to TMR. On the other hand, the large latency for applying SEC on larger block sizes suggests that the decoding should be done in a pipelined fashion. This pipeline implementation will require architectural changes so that the data is read in the first cycle and the decoding is done during the second cycle. An important point to note is that Hamming code provides only single bit error correction while TMR can tolerate any w-bit error confined to a single word. The overall bit error rate analysis done in the previous section captures this effect and it should be part of the overall reliability computation for any system. Notice that the probability of multi-bit errors is extremely small. Therefore a choice of block size for Hamming code may still provide area- and latency-efficient error protection for register files.

VI. CONCLUSION

We have synthesized 32-bit and 64-bit wide, 32-entry, dual read port, single write port RISC style register files in IBM Cu-08 90nm technology. We have implemented SEC Hamming code for various block lengths and TMR soft error mitigation techniques for these register files. A bit-error probability analysis has been performed to make the overall reliability comparison fair between TMR and Hamming codes. It is noted that the overall bit error rate probability analysis should be carefully analyzed for a system before selecting various soft error mitigation techniques. The area

and read access and write setup latency results have been analyzed for TMR and various block size Hamming codes. These results propose that TMR is a better solution for latency sensitive applications incurring only 17% overhead in read access for a 64-bit wide register file but have 204% area overhead. Whereas SEC applied on the 64-bit word size incurs 129% increase in read access time with area penalty only about 22%. This large read access penalty for applying SEC on wide words makes it necessary to partition the word into smaller block sizes before applying SEC. Latency and area figures from synthesis results for applying SEC on smaller block sizes show that a Hamming code provides various trade-offs between area and latency for register files. An SEC Hamming code applied on a block size of 4-bits can be a good trade-off for area against TMR with some degradation in speed compared to TMR.

REFERENCES

- [1] P. Shivakumar, M. Kistler, S.W. Keckler, D. Burger, L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic". Dependable Systems and Networks, International Conference on 23-26 June 2002 Page(s):389 – 398
- [2] W. Peterson, "Error-correcting codes", 2nd ed., Cambridge: The MIT Press, 1980
- [3] D. Bessot, R. Velazco, "Design of SEU-hardened CMOS memory cells: the HIT cell". Radiation and its Effects on Components and Systems, Second European Conference on, RADECS 93, 13-16 Sept. 1993 Pages: 563 – 570
- [4] T. Calin, M. Nicolaidis, R. Velazco, "Upset hardened memory design for submicron CMOS technology". Nuclear Science, IEEE Transactions on, Volume 43, Issue 6, Dec. 1996 Page(s):2874 – 2878
- [5] Q. Shi and G. Maki, "New design techniques for SEU immune circuits," NASA Symposium on VLSI Design, Nov. 2000.
- [6] K.J. Hass, J.W. Gambles, B. Walker, M. Zampaglione, "Mitigating single svent upsets from combinational logic". 7th NASA Symposium on VLSI Design 1998
- [7] D.G. Mavis, P.H. Eaton, "Soft error rate mitigation techniques for modern microcircuits". Reliability Physics Symposium Proceedings, 40th Annual, 7-11 April 2002 Page(s):216 – 225
- [8] R. Naseer, J. Draper, "The DF-DICE storage element for immunity to soft errors ". Circuits and Systems, 2005. 48th Midwest Symposium on , August 7-10, 2005 Page(s):303 - 306
- [9] T.J. Dell, "A whitepaper on the benefits of Chippkill-Correct ECC for PC server main memory", IBM Microelectronics division Nov 1997
- [10] C. Chen, A.K. Somani, "Fault Containment in Cache Memories for TMR redundant processor systems", IEEE Transactions on computers, March 1999 Pages:609-623
- [11] S. Kim, A.K. Somani, "An adaptive write error detection technique in on-chip caches of multi-level cache systems". Journal of microprocessors and microsystems, March 1999 Pages:561-570
- [12] Memik, G.; Kandemir, M.T.; Ozturk, O.; "Increasing register file immunity to transient errors". Design, Automation and Test in Europe, 2005. Proceedings 2005 Page(s):586 - 591 Vol. 1
- [13] R. Hentschke, et al. "Analyzing Area and Performance penalty of protecting differential modules with hamming code and triple modular redundancy". SBCCI'02
- [14] Jeffrey Draper, et al, A Prototype Processing-in-Memory (PIM) Chip for the Data-Intensive Architecture (DIVA) System, Journal of VLSI Signal Processing , Vol 40, Number 1, May 2005, pp. 73 - 84
- [15] M. Spica, T.M. Mak, "Do we need anything more than single bit error correction (ECC)?" Memory Technology, Design and Testing, Records of the 2004 International Workshop on 9-10 Aug. 2004 Page(s):111 - 116