Enhanced Duplication: a Technique to Correct Soft Errors in Narrow Values

İ. Burak Karsli[1], Pedro Reviriego[2], M. Fatih Balli[1], Oğuz Ergin[1] and J. A. Maestro[2]

[1]TOBB University of Economics and Technology, Ankara, Turkey
[2]Universidad Antonio de Nebrija, Madrid, Spain

Abstract— Soft errors are transient errors that can alter the logic value of a register bit causing data corruption. They can be caused by radiation particles such as neutrons or alpha particles. Narrow values are commonly found in the data consumed or produced by processors. Several techniques have recently been proposed to exploit the unused bits in narrow values to protect them against soft errors. These techniques replicate the narrow value over the unused register bits such that errors can be detected when the value is duplicated and corrected when the value is tripled. In this letter, a technique that can correct errors when the narrow value is only duplicated is presented. The proposed approach stores a modified duplicate of the narrow value such that errors on the original value and the duplicate can be distinguished and therefore corrected. The scheme has been implemented at the circuit level to evaluate its speed and also at the architectural level to assess the benefits in correcting soft errors. The results show that the scheme is significantly faster than a parity check and can improve substantially the number of soft errors that are corrected compared to existing techniques.

Index Terms— Error Correction, Soft Errors, Narrow Values, Data Cache.

1 Introduction

SOFT errors are an important issue in modern computing systems [1]. They are caused when radiation affects the electronic components such as memories and processors. For terrestrial systems soft errors are mostly caused by neutrons from outer space or solar flare and alpha particles from radioactive impurities in the integrated circuits [9]. A soft error causes a transient error in a circuit node that changes its logical value. When the soft error affects a storage element such as a memory or a register file, its data is corrupted and the execution of the computing system from that point on may be incorrect [8]. A number of techniques have been developed over the years to mitigate soft errors [7]. These range from changes in the manufacturing process that prevent soft errors from occurring to system level redundancy schemes that detect and correct soft errors.

It has been observed that a significant percentage of the values produced and consumed in modern processors are narrow and use only a fraction of the register bit-width [2]. This narrowness can be used to detect and correct soft errors as proposed in [3],[4],[5]. In those works duplication of the narrow value over the unused bits is proposed for error detection and replication three or more times for error correction. Also a bit was used to mark narrow values such that the processor can identify them and perform error detection or correction.

The main motivation to use replication and comparison is that it can be implemented with low delay and power consumption. More complex schemes such as parity check or Error Correction Codes (ECCs) require a substantially larger delay and power [4]. Low delay and power are critical in modern processors and therefore any soft error mitigation technique should have a small overhead in terms of delay and power.

In this letter, Enhanced Duplication (ED), a technique that can correct single bit soft errors on narrow values that only use bits in the lower half of the registers is presented. This is a clear improvement over previous techniques that for duplication only provide error detection [4],[5]. The proposed scheme is based on duplication but using a modified copy for the duplicate. This modification is similar to a convolution encoding [6] and ensures that single errors on the original copy and the duplicate can be distinguished and therefore correction can be performed by selecting the copy that is error free. The results show that Enhanced Duplication can be implemented with lower delay than parity check. The low delay combined with the error correction capability, make Enhanced Duplication an interesting option to protect register files in modern processors.

The rest of the letter is organized as follows, on section 2 the proposed technique is described and analyzed. Then in section 3 the scheme is evaluated at the circuit level in terms of delay and at the architecture level in terms of the percentage of soft errors that are corrected using a software benchmark running on a processor simulator. Finally the conclusions are discussed in section 4.

Manuscript submitted: 13-Mar-2012. Manuscript accepted: 15-Apr-2012. Final manuscript received 20-Apr-2012.

Posted to the IEEE & CSDL on 4/27/2012
DOI 10.1109/L-CA.2012.6 1556-6056/12/$31.00

2 Enhanced Duplication (ED)

The proposed technique is based on duplication such that the registers are organized in two halves and a bit that identifies narrow values is added. The proposed register organization is similar to that used in [4] and is illustrated in Figure 1. When a narrow value is stored in the register, the Narrow Value Identifier Bit (NVIB) is set to one and the upper half bits are a modified version of the lower half. The modification is done by computing the xor of each bit and its adjacent bit to the left except for the upper most bit. For that bit the xor is done with the rightmost bit. This processing is illustrated in Figure 2. The computation of the modified copy to be stored in the upper half requires only one level of xor gates and therefore can be implemented with low delay. When a value is read, if the NVIB is set to one error detection is performed. This is done by recomputing the upper bits based on the bits stored in the lower half and comparing with the bits the stored in the upper bits of the register. The process is illustrated in Figure 3. It can be observed that in this case it involves one level of xor gates plus a comparison. Therefore the additional delay compared with normal duplication and comparison is only one xor gate.

Once an error is detected, the error correction process is triggered. Assuming a single bit error, the half that has suffered the error can be identified by checking the number of differences between the stored upper half and the recomputed one. If there is only one difference, then the error occurred in the upper half, if there are two differences, then the error occurred in the lower half. If the error occurred in the upper half, the narrow value can be safely recovered from the lower half. However, if the error occurred in the lower part, the original data has to be recovered from the modified copy stored in the upper half. This can be done as follows, set the recovered L_1 to 0 and then compute the rest of the bits iteratively: $L_2 = (L_1)$ xor (U_1); $L_3 = (L_2)$ xor (U_2) and so on. If the value obtained for L differs with that stored in the register by one bit then the recovered L is the original data. If there are more differences, then set the recovered L_1 to 1 and apply the same procedure to recover the rest of the bits. If the value obtained for L differs with that stored in the register by one bit then the recovered L is the original data. The process described ensures that all single bit errors are corrected in the Enhanced Duplication technique. Obviously, the added delay for correction is much larger than for detection. However, this is not a major problem as soft errors are rare events and therefore in most cases the registers will be error-free. This means that the impact on the average delay will be low as noted in [11] where fast error detection followed by a slower error correction was proposed to protect caches.

Soft errors can also affect the NVIB. When that bit had a value of one and the error flips it to zero the error will not be detected. When it had a value of zero and the error flips it to one, in most cases there will be many differences in the error detection process. Therefore the error can be identified as having affected the NVIB. One option to ensure error detection in all cases is to duplicate this bit.

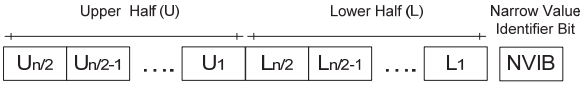

Figure 1. Register organization in the proposed technique.

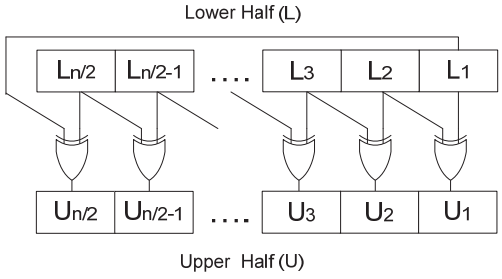

Figure 2. Computation of the upper bits in the proposed technique.

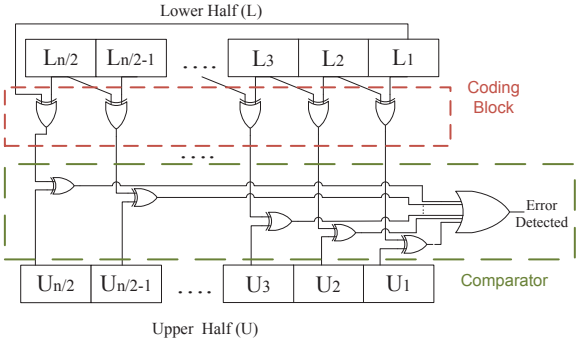

Figure 3. Error detection process in the proposed technique.

The use of the proposed technique is illustrated in Figure 4 for the case of an 8 bit register storing a 4 bit narrow value. In a) an example of a narrow value stored with the proposed technique is shown. Then in b) the case of an error in the upper half of the register is considered showing that it causes a single difference between the stored and recomputed versions of the modified duplicate. Therefore the error has occurred in the upper half and the narrow value is recovered from the lower half. In c) an error in the lower half is considered showing that it causes two differences between the stored and recomputed versions of the modified duplicate. Then the narrow value is recovered using the procedure described previously.

From the analysis presented, it can be concluded that the proposed technique is able to correct all single errors affecting the register bits except the NVIB. The added latency is only one xor gate more than duplicate and compare. The area of the circuitry needed for implementation will be larger than for duplicate and compare or parity check mostly due to the error correction mechanism. However it will be small compared to the area of the register file having therefore a low overall impact on circuit area.

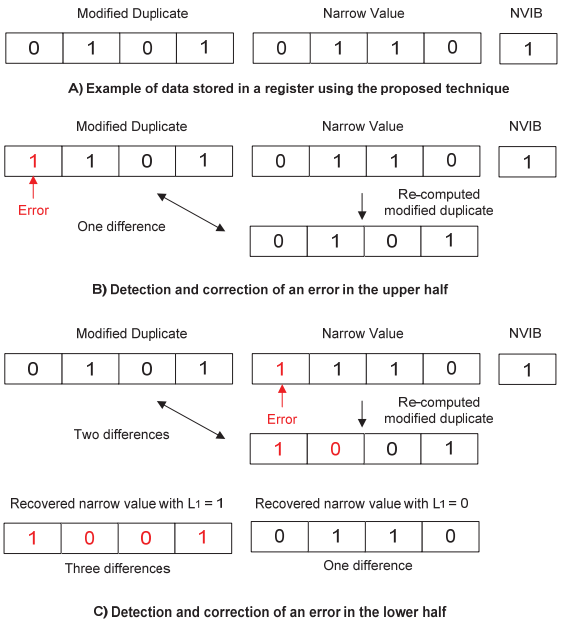

Figure 4. Example of the use of the proposed technique.

3 Evaluation

To evaluate the proposed Enhanced Duplication technique, it has been implemented. To minimize the impact on the processor performance, error detection is performed in parallel with the reading operation. This means that when an error is detected and corrected, the consumer has to access the corrected value again. To this end, the same mechanism as the one used for branch misprediction is used. Therefore no additional hardware is added to the pipeline for the removal of instructions. So, as the faulty value is read and used for execution, the error detection and correction circuitry performs error detection and signals the error if there is one. Meanwhile the dependent proceeds with the faulty value and is informed about the error when it concludes its execution. At this time the dependent that read the faulty value writes the error code inside the reorder buffer and all of the instructions that follow, including the dependent instruction that read the faulty value, are removed from the pipeline and refetched. This ensures that the correct value is used in the subsequent execution.

Figure 5 shows the reading circuit used for enhanced duplication. The computation on the upper order bits (that hold the calculated information) and the lower order bits are done in parallel and are compared against each other. According to the comparison results, the control logic decides if there is an error on the stored values and selects the correct version of the value to be written back to the storage space through a multiplexer activated by a generated valid signal.

The writing logic of the enhanced duplication is as simple as one level of xor gates connected in parallel. Therefore the delay overhead of the proposed scheme is as low as the delay of a single XOR gate. This delay is critical as the additional information has to be written inside the storage space at the same time with the actual data.

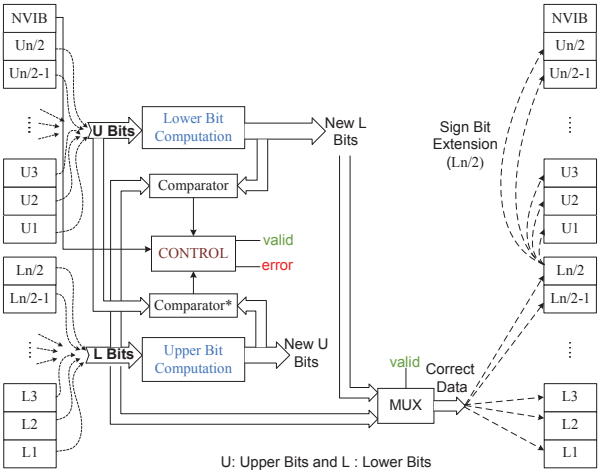

Figure 5. Reading circuit for Enhanced Duplication.

The error detection and correction logic used for enhanced duplication is more complicated than the information encoding logic. It consists of serially connected XOR gates as shown on the left side of Figure 6. For 32-bit data, the computation logic has the delay of 32 XOR gates which is a large delay overhead. In order to overcome this issue, we propose to partition the stored data into 4-bit elements. Each 4-bit partition is calculated separately upon the writing of the data and is also computed separately with a delay of 4 XOR gates connected in series. This way the number of control logic and the comparators is increased but the number of input bits to these blocks is decreased. With this partitioning method it is also possible to correct multiple bit errors if the faults occur on different partitions. This means that for a 64 bit register, 8 partitions of 4 bits are used to compute the modified copy and therefore up to 8 errors can be corrected if they occur one in each partition.

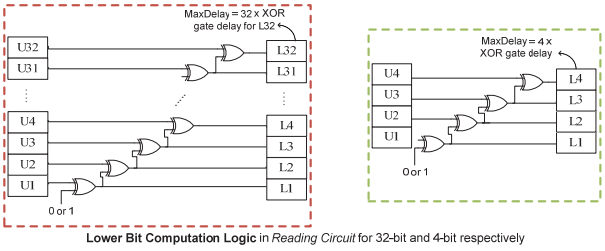

Figure 6. Computation circuit for enhanced duplication.

In order to estimate the delay overhead of the proposed enhanced duplication scheme we used the Cadence design tools with the 90nm UMC manufacturing technology files. The delay results are shown in Table 1. As it is shown in the table, the generation of the parity bit information takes significantly longer time than the generation of the redundant information required for enhanced duplication. The correction delay for the enhanced duplication is the required time for the circuit to generate the

correct value at its output while the detection delay is the required time for the circuit to generate the error signal in Figure 5. The circuit area required to implement Enhanced Duplication was 22% larger than that of an ECC that can also correct one error. However as, mentioned before the impact on the overall area of the register file will be much small smaller.

Table 1: Delay for the different schemes for 32 bits register

	Parity	Enhanced Duplicaton
Coverage	*All Values*	*32 bit narrow Values*
Redundant Information	*1bit*	*32bit*
Recovery/Detection	*Only Detection*	*Detection and Recovery*
Writing Latency	*355ps*	*56ps*
Error detection when reading	*510ps*	*318ps*
Correction	*X*	*508ps*

Enhanced Duplication requires the presence of narrow values to be effective. In order to get an accurate idea of the value widths inside contemporary applications we used the MSIM [10] processor simulator along with the SPEC 2006 benchmark suite programs. Figure 6 shows the cumulative percentage of value widths stored inside the register file for each SPEC 2006 benchmark program. As the graph figure reveals most of the values stored inside the register file can be represented with 34 or fewer bits.
If the enhanced duplication is used with a 32-bit narrow value definition, it is possible to fit all the redundant information inside the available storage space for the register file of a 64-bit processor (except for the NVIB bit). If the enhanced duplication is implemented for 34-bits, a total of 68 bits of storage space is needed to hold addional information. In this case an addional storage space of 4-bits is required to implement enhanced duplication.

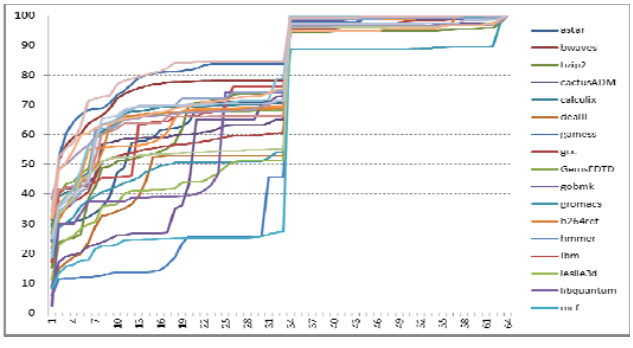

Figure 7. Percentage of Narrow Values for SPEC2006 Benchmarks.

Figure 8 shows the percent reduction of soft error Architectural Vulnerability Factor (AVF) when enhanced duplication is used with 34-bit and 32-bit narrow value definitons. As the figure reveals, on the average across all benchmarks, it is possible to reduce the vulnerability of the register file by more than 65% for 32 bit narrow values. The reduction goes up to more than 95% when a 4-bit information overhead is used so that 34 narrow values can be supported.

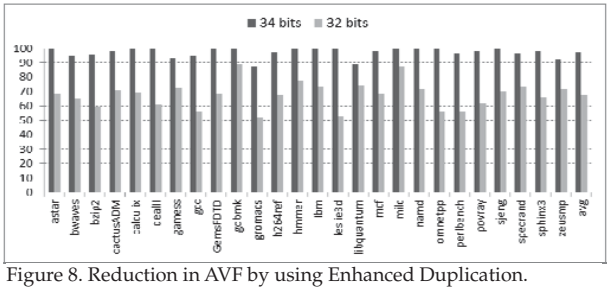

Figure 8. Reduction in AVF by using Enhanced Duplication.

4 Conclusions

In this letter, Enhanced Duplication, a new technique to correct soft errors in narrow values has been presented and evaluated both at the circuit and at the architectural level. At the circuit level, the results show that Enhanced Duplication can be implemented with a lower latency than a parity check and enables the correction of soft errors that affect a single bit. At the architectural level, it has been shown that due to the large percentage of narrow values in common applications, Enhanced Duplication can provide significant reductions in the Architectural Vulnerability Factor of the processor register file.

Acknowledgements

This work was supported in part by the Spanish Ministry of Science and Education under Grant AYA2009-13300-C03 and by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant 112E004. The work is a collaboration in the framework of COST ICT Action 1103 “Manufacturable and Dependable Multicore Architectures at Nanoscale”.

References

[1] R. C. Baumann, “Soft errors in advanced computer systems”, IEEE Des. Test. Comput., vol. 22, no. 3, pp. 258–266, May/Jun. 2005.
[2] D. Brooks and M. Martonosi, “Dynamically Exploiting Narrow Width Operands to Improve Processor Power and Performance”, Proc. Int’l Symp. High-Performance Computer Architecture (HPCA), 1999.
[3] O. Ergin, O. Unsal, X. Vera, and A. González, “Exploiting Narrow Values for Soft Error Tolerance”, IEEE Computer Architecture Letters (CAL ’06), vol. 5, 2006.
[4] O. Ergin, O. Unsal, X. Vera and A. Gonzalez, “Reducing Soft Errors through Operand Width Aware Policies”, IEEE Transactions on Dependable and Secure Computing, vol. 6, issue, 3, July-Sept. 2009.
[5] J. Hu, S. Wang, and S.G. Ziavras, “In-Register Duplication: Exploiting Narrow-Width Value for Improving Register File Reliability”, Proc. Int’l Conf. Dependable Systems and Networks (DSN), 2006.
[6] J. J. Metzner, “Convolutionally Encoded Memory Protection”, IEEE Transactions on Computers, vol 31, no 6, pp 547-551, 1983.
[7] M. Nicolaidis, “Design for Soft Error Mitigation”, IEEE Transactions on Device and Materials Reliability, Vol. 5, No. 3, September 2005.
[8] S. K. Reinhardt, S. S. Mukherjee, and J. Emer, “The Soft Error Problem: An Architectural Perspective”, 11th International Symposium on High-Performance Computer Architecture (HPCA), 2005.
[9] R.D. Schrimpf and D. M. Fleetwood “Radiation effects in integrated circuits and electronic devices”, World Scientific Publishing, 2004.
[10] J. Sharkey, “M-Sim: A Flexible, Multithreaded Architectural Simulation Environment”, Technical Report CS-TR-05-DP01, Dept. of CS, SUNY - Binghamton, October 2005.
[11] C. Wilkerson, A. R. Alameldeen, Z. Chishti, W. Wu, D. Somasekhar and S. Lu, “Reducing Cache Power with Low Cost, Multi-bit Error-Correcting Codes”, International Symposium on Computer Architecture, pp. 83-93, Jun. 2010.