uc3m | Universidad Carlos III de Madrid

e-Archivo

This is a postprint version of the following published document:

Liu, He; Reviriego, Pedro; Argyrides, Costas; Xiao, Liyi. (2022). Correction Masking: a Technique to Implement Efficient SET Tolerant Error Correction Decoders. *IEEE Transactions on Device and Materials Reliability*, 11(1), pp.: 36-41.

DOI: https://doi.org/10.1109/TDMR.2021.3132045

©2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
See https://www.ieee.org/publications/rights/index.html for more information.

Correction Masking: a Technique to Implement Efficient SET Tolerant Error Correction Decoders

He Liu, Pedro Reviriego, *Senior Member, IEEE*, Costas Argyrides, *Senior Member, IEEE*,
Liyi Xiao, *Member, IEEE*

***Abstract*—Single Event Transients (SETs) can be a major concern for combinational circuits. Its importance grows as technology scales because a small charge can create a large disturbance on a circuit node. One example of circuits that can suffer from SETs is the decoders of the Error Correction Codes (ECCs) that are used to protect memories from errors. This paper presents Correction Masking (CM), a technique to implement SET tolerant syndrome decoders. The proposed technique is presented and evaluated both in terms of protection effectiveness and circuit overhead. The results show that it can provide an effective protection while reducing the circuit area and power significantly compared to a Triple Modular Redundancy (TMR) protection. An interesting result is that Correction Masking also reduces the delay as it adds less logic in the critical path than TMR. Finally, the proposed technique can be used for any syndrome decoder. This means that it is applicable to many of the ECCs used to protect memories such as Single Error Correction (SEC), Single Error Correction Double Error Detection (SEC-DED), Single Error Correction Double Adjacent Error Correction (SEC-DAEC), and 3-bit burst codes.**

***Index Terms*—Error Correction Codes, Memories, Single Event Transients.**

I. Introduction

Memories are an important element in modern computing systems and store data that is critical for correct system operation. An important issue for memories is that as most modern electronic devices, they are prone to experience errors due to noise, interference, aging or radiation induced soft errors [1]. To avoid data corruption, Error Correction Codes (ECCs) are widely used to protect memories [2]. For example, Single Error Correction (SEC) [3] and Single Error Correction Double Error Detection (SEC-DED) [4] codes are commonly used to protect against single bit errors. However, as technology scales, Multiple Cell Upsets (MCUs) that affect nearby memory cells become more prominent [1] and codes that can correct double adjacent errors [5], [6] or larger burst of errors [7], [8], [9], [10] are also becoming popular.

In all cases, an Error Correction Code introduces a number of overheads to the memory. For example, parity bits have to be added to each memory word thus increasing the number of memory cells needed to store the same amount of data. An encoder is needed in write operations to compute those parity check bits and a decoder to use them in read operations to detect and when possible correct errors. The encoder and decoder introduce additional circuitry and also delay for read/write operations.

An important consideration when using ECCs to protect memories is that the encoder and decoder combinational circuitry of the ECC may also be affected by errors itself. In fact, Single Event Transients (SETs) that cause errors on combinational logic become also more relevant as technology scales. Therefore, a number of techniques have been proposed to protect the encoders, for example of SEC [11] and SEC-DAEC [12] codes. Concurrent error detection has also been considered for Orthogonal Latin Square Codes [13], [14] or Matrix codes [15]. However, all those solutions are specifically designed for a type of codes and not generally applicable. In summary, there is no general protection technique to protect ECC decoders except for the traditional Triple Modular Redundancy (TMR) that triplicates the decoder and votes among the three replicas. However, TMR introduces a large penalty in circuit area and power that may not be acceptable in some applications and also impacts delay due to the voting logic.

In this paper, Correction Masking (CM), an efficient technique to implement SET tolerant decoders is presented and evaluated. The results show that CM can significantly reduce the protection cost compared to TMR while providing an effective protection. The proposed CM technique is applicable to any syndrome based decoder and thus includes most of the ECCs used to protect memories such as SEC, SEC-DED, SEC-DAEC, and 3-4 bit burst error correction codes. The rest of the paper is organized as follows, in Section II the general structure of a syndrome based decoder is presented to provide the background needed to present the new technique. The proposed Correction Masking technique is presented in Section III and evaluated in Section IV. The paper ends in Section V with the conclusions and ideas for future work.

II. Syndrome based Decoder

A common decoder implementation for SEC-DED [4], SEC-DAEC [5] or 3-bit burst [8], [9] codes is to recompute the parity check equations to obtain a vector called syndrome

H. Liu and L. Xiao are with the Microelectronics Center, Harbin Institute of Technology, Harbin 150001, China. (e-mail: liuhe_hitsa@163.com; xiaoly@hit.edu.cn).

P. Reviriego is with Universidad Carlos III de Madrid, Leganés 28911, Madrid, Spain. (e-mail: revirieg@it.uc3m.es).

C. Argyrides is with the Radeon Technologies Group, Advanced Micro Devices, Inc., Santa Clara, CA, 95054 USA. (e-mail: caa@ieee.org).

Corresponding author: L. Xiao (xiaoly@hit.edu.cn).

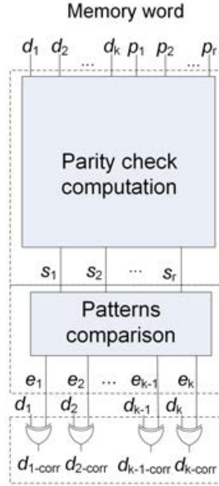

Fig. 1. Block diagram of a syndrome decoder.

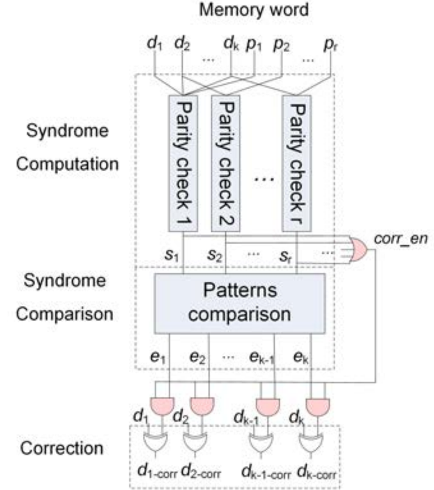

Fig. 2. Block diagram of the decoder protected with the proposed technique.

that is compared to the syndrome of each correctable pattern to identify and correct the bits in error. In more detail, the syndrome is calculated by multiplying the codeword with the parity check matrix, as shown in equation (1). So when in the absence of errors, the syndrome should be an all zeros vector as the parity bits were computed using the original data. If there is a single bit error, then the syndrome will match the column of the parity check matrix that corresponds to that bit. If a group of bits is in error, then the syndrome will match the xor of the columns that correspond to those bits. Therefore, after computing the syndrome, it can be compared to the correctable error patterns to determine the bits in error. Finally, once the erroneous bits have been identified, they can be corrected.

$$s = r \cdot H^T = e \cdot H^T \tag{1}$$

$$d_{i_corr} = d_i \oplus e_i \tag{2}$$

The block diagram of this decoder is illustrated in Fig. 1 that shows the parity computation, pattern comparison, and correction blocks. The parity computation produces the syndrome vector s_i that is then used to generate a correction vector e_i in the pattern comparison block. Then, based on that vector the data is corrected to obtain the modified output of the decoder. Obviously, as the number of correctable error patterns increases, so does the complexity of the pattern comparison logic. This makes decoders for SEC-DAEC codes more complex than for SEC codes, 3-bit burst decoders more complex than SEC-DAEC codes, and so on.

III. Correction Masking Protection

Since soft errors are rare events, a commonly used assumption is that within a full clock cycle, there is only one event in the system. This is for example needed for TMR or SEC codes to be effective. Therefore, the same assumption is made when designing the proposed fault tolerant decoders. In our case, this means that there is either an error in the input to the decoder (that would correspond to an SEU in the memory) or an SET in the decoder circuitry but not both. This leads to a subtle situation that will enable an efficient protection. Under the single error event assumption, if the decoder suffers an SET, its input has to be a correct word.

Based on that situation, the decoder can be protected against SETs by enabling the correction of the data bits only when an SEU has been detected. This can be done using the syndrome that is computed in the first stage of decoding. If there is an error on the word being decoded, then the syndrome will not be zero, and when the word is error free, the syndrome will be zero. Therefore, checking if the syndrome is different from zero could be used to enable correction of the data bits. However, there is, the possibility that the SET affects the syndrome computation circuitry so that for a correct word, the syndrome is no longer zero. This may lead to miss correction. To avoid this issue, the parity check equations can be implemented as independent blocks with no logic sharing so that an SET can at most corrupt one syndrome bit. Syndrome bits can be calculated with parity check equations and received codewords. The same equation is used to find the error. Expression (3), derived from equation (1) shows the calculation. This has been proposed to implement fault tolerant encoders for SEC-DED codes in [11]. A syndrome with only a single non zero bit would correspond to an error on a parity bit and will therefore not impact the correctness of the data at the output of the decoder. This combination of non logic sharing in the syndrome computation with the enabling of the correction only when syndrome is non zero can provide an efficient protection for the decoder with a much lower cost than traditional solutions such as TMR.

$$s_i = r \cdot H^T_{(i)} = e \cdot H^T_{(i)} \tag{3}$$

Let us now discuss the implementation of the proposed

technique in more detail. This is better done by referring to Figure 2 that shows a block diagram of the proposed decoder. The added logic is marked in red. Firstly, in the syndrome computation block, the parity check equations are computed using the word read from the memory that includes the data bits d_i and the parity bits p_i to obtain the syndrome bits s_i. This is done using independent circuitry for each parity check equation so that an SET would corrupt only one of the s_i bits. The syndrome is then fed to the pattern comparison block to generate the error signals e_i and also to an r input OR gate to generate the $corr_{en}$ signal as shown in equation (4). The second block compares the syndrome with the correctable error patterns to determine which (if any) bits need to be corrected and outputs the error pattern e_i. Then correction is done. In our design, the correction would only take place if both e_i and $corr_{en}$ are both one. The data are corrected with the $corr_{en}$ signal, in contrast to the data corrected with general syndromes based on (2). This means that when the syndrome is zero, no correction will take place regardless of the values of e_i. Therefore, SETs in the pattern comparison block will not corrupt the output data.

$$corr_{en} = s_1 + s_2 + \cdots + s_r \tag{4}$$

$$d_{i_corr_cm} = d_i \oplus (e_i \cdot corr_{en}) \tag{5}$$

The proposed technique can be used to protect any block binary error correction code that uses a syndrome comparison based decoder. Therefore, it can be used for example for SEC [3], SEC-DED [4], SEC-DAEC [5], [6] or three bit burst [9], [10] codes. This means that the proposed technique would be applicable to most error correction codes currently used to protect memories.

Let us now compare the proposed technique with the traditional TMR protection shown in Figure 3. It can be observed that TMR protection introduces a much larger overhead in terms of circuitry and also in terms of delay as TMR voting is more complex than the AND with $corr_{en}$ added to the critical path in our design. This will be corroborated in the following section when both options are implemented and compared.

IV. Evaluation

To evaluate the proposed technique, it has been implemented for words of 16,32 and 64 data bits and compared to both an unprotected decoder and a TMR protected decoder. The reason to use TMR for comparison is that the proposed scheme is applicable to any syndrome based decoder and thus it makes sense to compare it with a general technique like TMR rather that with schemes that are only applicable to a specific type of codes [13], [14]. This has been done for the following codes that are representative of the types of codes commonly used to protect memories:

- The Odd weight Single Error Correction Double Error Detection (SEC-DED) codes in [4].

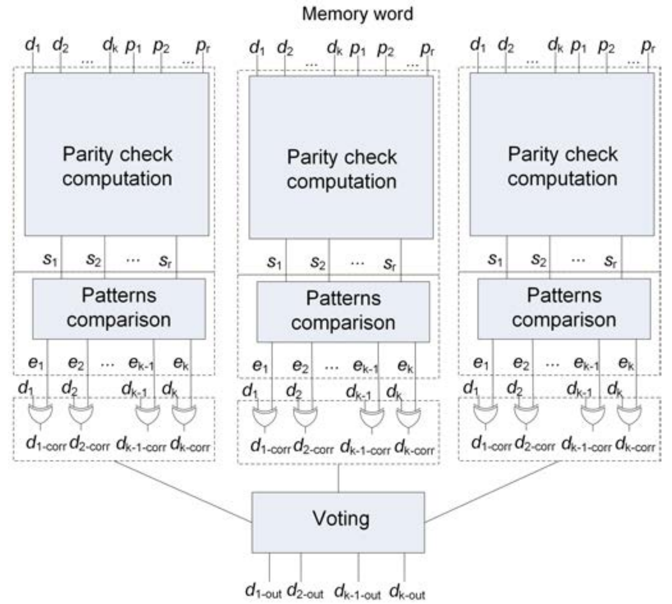

Fig. 3. Block diagram of a syndrome decoder protected with TMR.

- The Single Error Correction Double Adjacent Error Correction (SEC-DAEC) codes in [5].
- The three bit burst error correction codes in [9].
- The four bit burst error correction codes in [10].
- A Double Error Correction (DEC) BCH code.

The comparison is made both in terms of circuit complexity and fault tolerance. The following subsections cover each of those aspects.

A. Circuit Implementation

The decoders have been implemented in HDL and mapped to a TSMC 65nm library. The synthesis has been run twice, first setting maximum effort in optimizing area and then in optimizing delay. The data word lengths considered are k = 16, 32, and 64. For each of those configurations, three designs are compared: the unprotected decoder, the decoder protected with TMR and the proposed decoder. The results obtained are summarized in Table I. It can be observed that the proposed Correction Masking technique is able to reduce significantly the area, power, and delay in all configurations compared to TMR. This was expected as CM adds very little additional logic and most of it is not on the critical path. TMR instead triples the decoder and adds voters on the outputs that introduce additional delay. More specifically, for an area optimized synthesis, TMR requires an overhead in the range 206% to 253% compared to 24% to 43% for the proposed correction masking technique. For a delay optimized synthesis, TMR increases the delay by 14% to 26% while the proposed technique introduces a delay of 1-3% in most cases and 10% in the worst case. The savings in power compared to TMR also large for both synthesis options. In some cases, the proposed decoder is actually faster than the unprotected decoder. This may be due to the use of independent blocks

TABLE I
DECODER: AREA (μm^2), DELAY (ns), AND POWER (mW).

k	Code	Protection	Area Optimization				Delay Optimization			
			Area	Overhead	Delay	Power	Area	Delay	Overhead	Power
16	SEC-DED [4]	Unprotected	322.80	0%	1.86	0.47	1340.7	0.67	0%	1.13
		Proposed	402.80	24.78%	2.15	0.52	1548.3	0.68	1.49%	1.13
		TMR	1128.80	249.70%	2.44	1.61	4374.3	0.82	22.39%	3.76
	SEC-DAEC [5]	Unprotected	421.20	0%	2.67	0.72	2343.99	0.75	0%	2.40
		Proposed	543.20	28.96%	2.54	0.84	2537.59	0.79	5.33%	2.24
		TMR	1422.40	237.70%	3.27	2.37	5901.99	0.95	26.67%	5.67
	3-Burst [9]	Unprotected	503.20	0%	2.65	0.78	2519.1	0.80	0%	2.70
		Proposed	658.00	30.76%	3.20	0.87	3426.3	0.81	1.25%	2.30
		TMR	1663.20	230.52%	3.24	3.33	7277.1	0.93	16.25%	7.74
	4-Burst [10]	Unprotected	593.60	0%	2.85	1.16	2431.19	0.84	0%	2.57
		Proposed	1018.40	71.56%	3.91	1.24	4666.79	0.90	7.14%	3.08
		TMR	1932.40	225.54%	3.45	3.66	7798.39	0.99	17.86%	7.92
	BCH-DEC	Unprotected	1578.80	0%	4.52	3.99	7002.78	1.12	0%	8.92
		Proposed	1940.80	22.93%	4.49	3.02	7258.79	1.09	-2.68%	9.14
		TMR	4906.80	210.79%	5.05	12.16	20517.20	1.28	14.29%	26.62
32	SEC-DED [4]	Unprotected	584.80	0%	1.96	0.85	2390.7	0.78	0%	2.17
		Proposed	816.00	39.53%	2.22	0.97	2871.5	0.80	2.56%	2.45
		TMR	2065.20	253.15%	2.56	2.90	7598.3	0.93	19.23%	6.76
	SEC-DAEC [5]	Unprotected	774.00	0%	3.45	1.38	3598.79	0.88	0%	3.68
		Proposed	1057.20	36.59%	3.40	1.80	4129.19	0.88	0%	4.23
		TMR	2641.20	241.24%	4.05	4.50	11496.79	1.04	18.18%	11.40
	3-Burst [9]	Unprotected	1008.40	0%	3.89	1.79	6605.5	0.86	0%	8.05
		Proposed	1314.40	30.35%	4.15	1.91	7117.5	0.95	10.47%	6.99
		TMR	3317.20	228.96%	4.50	6.95	14296.3	1.03	19.77%	15.92
	4-Burst [10]	Unprotected	1448.00	0%	3.99	3.65	6535.99	1.01	0%	8.29
		Proposed	1927.60	33.12%	4.88	2.78	9209.99	1.03	1.98%	7.95
		TMR	4631.20	219.83%	4.67	11.32	17772.39	1.17	15.84%	23.33
	BCH-DEC	Unprotected	4876.00	0%	5.76	13.59	24971.19	1.17	0%	35.92
		Proposed	4791.20	-1.74%	5.57	6.12	22262.79	1.20	2.56%	18.65
		TMR	14966.40	206.94%	6.30	40.91	56064.39	1.43	22.22%	81.20
64	SEC-DED [4]	Unprotected	1163.6	0%	3.21	1.88	5128.7	0.84	0%	5.86
		Proposed	1671.2	43.62%	3.03	2.10	6405.9	0.86	2.38%	5.70
		TMR	4080.8	250.70%	4.03	6.28	16315.1	1.01	20.24%	17.08
	SEC-DAEC [5]	Unprotected	1497.60	0%	3.93	2.64	6408.79	1.05	0%	6.83
		Proposed	2139.20	42.84%	4.48	3.49	9068.39	0.95	-9.52%	9.76
		TMR	5098.80	240.46%	4.52	8.59	19009.19	1.19	13.33%	19.76
	3-Burst [9]	Unprotected	1945.20	0%	4.05	3.06	8717.5	1.05	0%	10.47
		Proposed	2474.40	27.21%	3.75	3.37	11087.5	1.04	-0.95%	10.17
		TMR	6372.00	227.58%	4.62	13.72	23695.5	1.20	14.23%	29.38
	4-Burst [10]	Unprotected	3113.20	0%	5.01	9.29	15345.59	1.06	0%	22.45
		Proposed	3894.40	25.09%	4.97	5.48	18361.19	1.13	6.60%	17.89
		TMR	9940.00	219.29%	5.68	28.50	38918.78	1.28	20.75%	59.53
	BCH-DEC	Unprotected	15470.40	0%	8.03	39.61	70029.79	1.41	0%	94.79
		Proposed	18162.80	17.40%	8.62	22.64	62635.58	1.45	2.84%	47.48
		TMR	47759.60	208.72%	8.33	120.67	163594.79	1.72	21.99%	224.48

for the computation of the parity check equations in the proposed technique that may enable the synthesizer to obtain a better result. Those cases were double checked to ensure that the same results were consistently obtained on different runs. Overall, the benefits of correction masking over TMR are observed consistently for all the codes considered. These results clearly illustrate the benefits of the Correction Masking in terms of circuit complexity compared to TMR.

B. Fault Tolerance

To evaluate the effectiveness of the proposed technique in protecting against Single Event Transients (SETs), FITBS, a RTL-level fault injection tool based on the simulation has been used [16]. FITBS analyzes the circuit structure with the gate-level netlists to obtain the information of the nodes in circuit and performs fault injection into the circuit. In the tool, the widely used Modelsim HDL simulator is invoked, so the results from the simulator can be reproduced. The tool invokes the simulator to get a golden model without faults at first. Then faults are injected to the circuits. Simulator generates

TABLE II
FAULT INJECTION.

Data Bits	Code	Protection	N_{node}	Total SETs	SET Pulse (500 to 1000[ps])		SET Pulse (50 to 200[ps])	
					SET Failure	α_{system}	SET Failure	α_{system}
16	SEC-DED [4]	Unprotected	63	31500	3356	10.65%	479	1.52%
		Proposed	63	31500	0	0%	0	0%
		TMR	237	118500	0	0%	0	0%
	SEC-DAEC [5]	Unprotected	98	49000	8100	16.53%	1178	2.40%
		Proposed	104	52000	0	0%	0	0%
		TMR	357	178500	0	0%	0	0%
	3-Burst [9]	Unprotected	145	72500	9139	12.61%	911	1.26%
		Proposed	174	87000	0	0%	0	0%
		TMR	515	257500	0	0%	0	0%
	4-Burst [10]	Unprotected	170	85000	7243	8.52%	900	1.06%
		Proposed	305	152500	0	0%	0	0%
		TMR	581	290500	0	0%	0	0%
	BCH-DEC	Unprotected	585	292500	34769	11.89%	2956	1.01%
		Proposed	631	315500	0	0%	0	0%
		TMR	1732	866000	0	0%	0	0%
32	SEC-DED [4]	Unprotected	119	59500	6258	10.52%	967	1.63%
		Proposed	126	63000	0	0%	0	0%
		TMR	462	231000	0	0%	0	0%
	SEC-DAEC [5]	Unprotected	200	100000	16128	16.13%	2408	2.41%
		Proposed	217	108500	0	0%	0	0%
		TMR	705	352500	0	0%	0	0%
	3-Burst [9]	Unprotected	325	162500	22175	13.65%	3292	2.02%
		Proposed	380	190000	0	0%	0	0%
		TMR	1019	509500	0	0%	0	0%
	4-Burst [10]	Unprotected	479	239500	29024	12.12%	3967	1.66%
		Proposed	604	302000	0	0%	0	0%
		TMR	1494	747000	0	0%	0	0%
	BCH-DEC	Unprotected	1785	892500	95951	10.75%	11826	1.33%
		Proposed	1789	894500	0	0%	0	0%
		TMR	5449	2724500	0	0%	0	0%
64	SEC-DED [4]	Unprotected	247	123500	13156	10.65%	2126	1.72%
		Proposed	276	138000	0	0%	0	0%
		TMR	969	484500	0	0%	0	0%
	SEC-DAEC [5]	Unprotected	401	200500	31853	15.89%	3948	1.97%
		Proposed	431	215500	0	0%	0	0%
		TMR	1360	680000	0	0%	0	0%
	3-Burst [9]	Unprotected	586	293000	44713	15.26%	5874	2.00%
		Proposed	706	353000	0	0%	0	0%
		TMR	1890	945000	0	0%	0	0%
	4-Burst [10]	Unprotected	1043	521500	64281	12.33%	8282	1.59%
		Proposed	1319	659500	0	0%	0	0%
		TMR	3294	1647000	0	0%	0	0%
	BCH-DEC	Unprotected	6028	3014000	345585	11.47%	40369	1.34%
		Proposed	6861	3430500	0	0%	0	0%
		TMR	17620	8810000	0	0%	0	0%

the fault-injected results. The FITBS tool can compare the results from fault-injected and golden model to simulate the effects of SETs.

The injection was done on the synthesized netlist for a 65nm technology library. Two SET pulse width ranges are defined and set according to the general SET pulse distribution [17]. The typical SET pulse distribution range is 50 to 200 ps and the worst case is 500 to 1000 ps. For each design, all the injection nodes are analyzed from the netlist file. The correction gates and TMR voters in all designs were not considered for SET injection insertion as they are assumed to be implemented with rad-hard cells. Then 500 injections are performed on each node at different random time instants.The failure rate of each injection node is defined as the ratio of the number of failure occurrence and the number of total injection times. The final system failure rate is the average of all the injection node failure rates. These two evaluation parameters are defined in the following formula:

$$\alpha_{node} = \frac{n_{occurrence}}{n_{total_injection_times}} \quad (6)$$

$$\alpha_{system} = \frac{\sum_{i=1}^{i=N_{node}} (\alpha_{node})_i}{N_{node}} \quad (7)$$

where N_{node} is the number of the total injection nodes; α_{node} is the failure rate of each injection node;α_{system} is the failure rate of the whole system.

The results are shown in Table II. It can be seen that both TMR and correction masking tolerate all the injected SETs while the unprotected designs have failure rates of 1-2.5% for the short pulses and of 8-16% for the long pulses. Therefore, these experiments suggest that the proposed correction masking technique will be effective in protecting the decoders against SETs.

V. Conclusion and Future Work

This paper has presented Correction Masking (CM) an efficient technique to protect syndrome decoders against Single Event Transients (SETs). Correction Masking exploits the single error assumption to mask corrections when the syndrome is zero thus ensuring that SETs that affect the decoder when its input is a valid word will not affect the output data. This provides an effective protection while requiring a much simpler logic than Triple Modular Redundancy (TMR).

The proposed technique has been implemented and evaluated for different codes that include SEC-DED, SEC-DAEC, 3-bit burst, 4-bit burst and DEC codes. The results confirm that it can provide an effective protection with a significant reduction in area, power an delay compared to TMR. Therefore, Correction Masking seems to be an attractive alternative to TMR to protect syndrome decoders. It can be used for example to protect the decoders of SEC-DED, SEC-DAEC or three bit burst codes that are frequently used codes for memory protection.

Future work will consider the implementation of a physical prototype of correction masking to validate it with radiation testing.

Acknowledgment

Pedro Reviriego would like to acknowledge the support of the ACHILLES project PID2019-104207RB-I00 and the Go2Edge network RED2018-102585-T funded by the Spanish Agencia Estatal de Investigación (AEI) 10.13039/501100011033 and of the Madrid Community research project TAPIR-CM grant no. P2018/TCS-4496.

AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies. ©2021 Advanced Micro Devices, Inc. All rights reserved.

References

[1] N. Kanekawa, E. H. Ibe, T. Suga Y. and Uematsu, "Dependability in Electronic Systems," Springer, New York, 2011.

[2] C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: a state-of-the-art review," IBM J. Res. Develop., vol. 28, no. 2, pp. 124-134, Mar. 1984.

[3] R. W. Hamming, "Error detecting and error correcting codes," The Bell Syst. Tech. J., vol. 26, no. 2, pp. 147-160, 1950.

[4] M. Y. Hsiao, "A class of optimal minimum odd-weight-column SEC-DED codes", IBM J. Res. Develop., vol. 14, no. 4, pp. 395-401, Jul. 1970.

[5] A. Dutta and N. A. Touba, "Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code," in Proc. IEEE VLSI Test Symp., May 2007, pp. 349-354.

[6] Z. Ming, X. L. Yi, and L. H. Wei, "New SEC-DED-DAEC codes for multiple bit upsets mitigation in memory," in Proc. IEEE/IFIP 20th Int. Conf. VLSI Syst.-Chip, Oct. 2011, pp. 254-259.

[7] D. Abhishek and N. A. Touba, "Low Complexity Burst Error Correcting Codes to Correct MBUs in SRAMs," in Proc. ACM Great Lakes Symposium on VLSI, May 2018, pp. 219-224.

[8] A. Klockmann, G. Georgakos and M. Goessel, "Systematic Design of a New 3-Bit-Burst-Error Correction Code with Minimal Number of Check Bits," Reliability by Design; 9. ITG/GMM/GI-Symposium, Cottbus, Germany, 2017.

[9] L. J. Saiz-Adalid, P. Reviriego, P. Gil, S. Pontarelli, and J. A. Maestro,"MCU tolerance in SRAMs through low-redundancy triple adjacent error correction," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 10, pp. 2332-2336, Oct. 2015.

[10] J.Q. Li, L.Y. Xiao, P. Reviriego abd R.S. Zhang, "Efficient Implementations of 4-Bit Burst Error Correction for Memories," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 65, no. 12, pp. 2037-2041, March. 2018.

[11] J.A. Maestro, P. Reviriego, C. Argyrides and D.K. Pradhan, "Fault Tolerant Single Error Correction Encoders," Journal of Electronic Testing: Theory and Applications, vol. 27, no. 2, pp. 215-218 , Apr. 2011.

[12] S. Liu, P. Reviriego, L. Xiao and J.A. Maestro, "Fault Tolerant Encoders for Single Error Correction and Double Adjacent Error Correction Codes," Microelectronics Reliability, Elsevier, 2018, 81:167-173.

[13] K. Namba and F. Lombardi, "Concurrent error detection of binary and nonbinary OLS parallel decoders," IEEE Trans. Device Mater. Rel., vol. 14, no. 1, pp. 112-120, Mar. 2014.

[14] P. Reviriego, S. Pontarelli and J.A. Maestro, "Concurrent Error Detection for Orthogonal Latin Squares Encoders and Syndrome Computation," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no. 12, pp. 2334-2338, Dec. 2013.

[15] S. Liu, L. Xiao, J. Guo and Z. Mao, "Fault Secure Encoder and Decoder Designs for Matrix Codes," IEEE 14th International Conference on Computer-Aided Design and Computer Graphics, Aug. 2015, pp.181-185.

[16] L. Xiao, A. Li, X. Cao, H. Li, R Zhang, J, Li, T. Wang, "A method to estimate cross-section of circuits at RTL levels,"IEEE 12th International Conference on ASIC (ASICON), Oct. 2017, pp.363-366.

[17] X. Cao, L. Xiao, J. Li, R. Zhang, S. Liu, J. Wang, "A Layout-Based Soft Error Vulnerability Estimation Approach for Combinational Circuits Considering Single Event Multiple Transients (SEMTs)," IEEE Trans. Computer-Aided Design Integr. Circuits Syst., vol. 38, no. 6, pp. 1109-1122, June 2019.