J Electron Test (2011) 27:215–218
DOI 10.1007/s10836-011-5208-9

Fault Tolerant Single Error Correction Encoders

Juan Antonio Maestro · Pedro Reviriego · Costas Argyrides · Dhiraj K. Pradhan

Received: 5 September 2009 / Accepted: 21 February 2011 / Published online: 30 March 2011

Abstract Soft errors are an important issue for circuit reliability. To mitigate their effects on the system functionality, different techniques are used. In many cases Error Correcting Codes (ECC) are used to protect circuits. Single Error Correction (SEC) codes are commonly used in memories and can effectively remove errors as long as there is only one error per word. Soft errors however may also affect the circuits that implement the Error Correcting Codes: the encoder and the decoder. In this paper, the protection against soft errors in the ECC encoder is studied and an efficient fault tolerant implementation is proposed.

Keywords Reliability · ECC · Fault tolerance

Responsible Editor: K. K. Saluja

J. A. Maestro · P. Reviriego
Departmento de Ingenieria Informatica,
Universidad Antonio de Nebrija, Madrid, Spain

J. A. Maestro
e-mail: jmaestro@nebrija.es

P. Reviriego
e-mail: previrie@nebrija.es

C. Argyrides (✉)
Research, C.A. EVOLVIT LTD,
8 Josep Broz Tito, 3010, Limassol, Cyprus
e-mail: costas@computer.org

D. K. Pradhan
Computer Science Department, University of Bristol,
Bristol, UK
e-mail: pradhan@cs.bris.ac.uk

1 Introduction

Soft errors are an important issue for electronic circuit reliability, as highlighted in the International Technology Roadmap for Semiconductors [5]. They can be caused for example by radiation particles hitting the circuit and altering the state of a logic element [10]. Several techniques have been proposed to mitigate the effects of radiation on electronic circuits ranging from modifications in the manufacturing process to system level approaches [8]. One of those techniques is the use of Error Correcting Codes (ECC) to protect data stored in memories or registers [3]. Single Error Correcting (SEC) codes for example are widely used to increase memory reliability [1].

ECCs add redundant bits to the memory word or register bits. Those bits provide error detection and correction capabilities. An ECC is typically implemented in two blocks, an encoder and a decoder as shown in Fig. 1. The encoder takes the information bits and produces the corresponding redundant bits. This is typically done when the data is stored. On the other hand, the decoder takes the stored data and redundant bits and performs error detection and correction.

There are many types of ECCs that provide different error correction capabilities [6]. In general, codes that can correct more errors imply a larger number of redundant bits and more complex encoders and decoders. The most common type of codes used to protect electronic circuits from soft errors is Single Error Correction codes, which can correct only one error [1]. Double Error Correction (DEC) codes are also used [7]. SEC codes are preferred as they result in a low number of additional bits and in simple and fast encoders and decoders. If the required reliability is not met, scrubbing

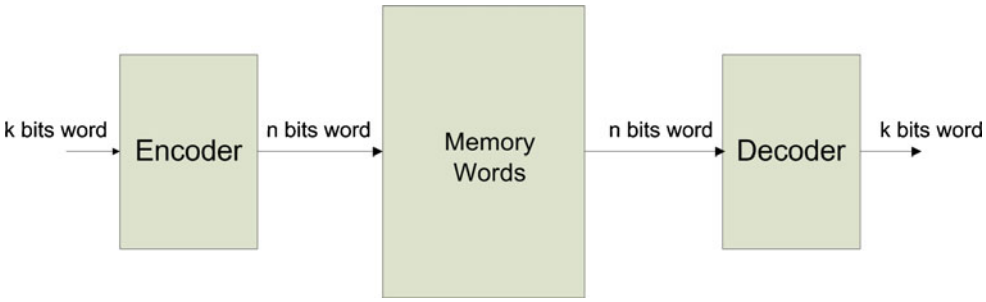

Fig. 1 Illustration of an ECC applied for memory protection

can be used in addition to SEC to achieve the requirements [9].

An important observation is that soft errors may also affect the protection logic, in our case the encoder and decoder circuits. Therefore those circuits should also be protected. A straightforward solution is to triplicate those circuits and use majority logic to select the correct value in case of error. This is known as Triple Modular Redundancy (TMR) [8]. However this may not lead to an efficient implementation.

In this paper, the protection of the encoder circuit for a SEC code is studied. The analysis first shows that TMR is not an efficient protection technique to then propose a more efficient approach.

2 Analysis of Single Error Correction Encoders

Single Error Correction encoders are simple circuits that compute a number of parity check bits for the input data. The data bits involved in each parity check bit is given by the code generator matrix G that defines the code. Several SEC codes can be used, like for example the ones proposed by Hamming [2] or the optimizations proposed in [4]. To illustrate the encoding process let us consider the (22,16) SEC code described in [4]. In this case, the code takes $k = 16$ data bits and computes six redundant bits to form a 22 bit word. The parity check bits are generated by the G matrix as shown in Fig. 2, where the first 16 column just map the input bit to the corresponding output bit and the last 6 compute the parity check bits. In other words, having an input vector x, the corresponding codeword is obtained by computing $c = x \cdot G$, where additions are performed modulo-2. This code can correct single errors and detect all double errors. The codes that have this capability are also known as SEC-DED codes.

$$G = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&1&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&1&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&1&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&1&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&0&1&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&0&1&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&1&0&0&1&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&1&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&1&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&1&0&0&1
\end{bmatrix}$$

Fig. 2 G-Matrix for an (22,16) code

The encoder basically performs multiple xor operations between the input data bits. For the code considered before, this can be implemented with 19 xor gates, as shown in Fig. 3 resulting in an efficient implementation. Let us consider the protection of this circuit. One option, as mentioned before, is to use

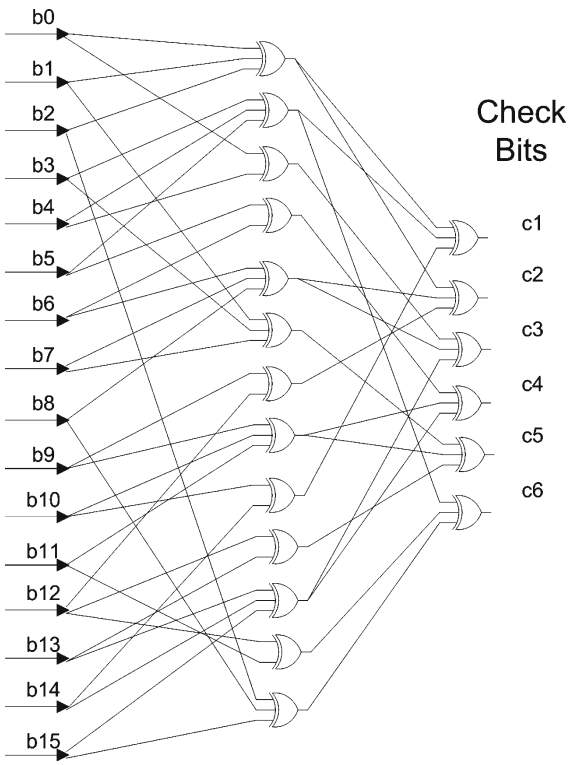

Fig. 3 Illustration of the encoder for the (22,16) SEC-DED code proposed in [4]

TMR. This would approximately triple the complexity. Another option is to consider the effects of an error on the encoder. Since the output of the encoder is a codeword that is stored in the memory, any soft error that corrupts a single bit can be corrected by the decoder and therefore would not cause a failure. In this regard, such error would be equivalent to a soft error in a memory bit.

This observation leads us to the following idea. If we are able to protect the encoder in such a way that soft errors only corrupt one bit of the encoder output, then the encoder would have the same protection level as the memory words or registers. This would minimize the cost with a negligible impact on reliability as a system is at most as reliable as it weakest element. Therefore there is little to be gained from providing extra protection to the encoders by using TMR.

If we consider the implementation in Fig. 3, it can be observed that errors on many of the first stage gates would corrupt more than one of the redundant bits. For example, an error at the output of the gate at the top of the circuit would corrupt the first two parity check bits. This error pattern could also be due to a double error on bits 9 and 10 or on bits 6 and 14 and it is therefore uncorrectable.

To avoid a soft error causing multiple errors on the check bits, the implementation could be modified as shown in Fig. 4. Basically, the computation of each parity bit has been made completely independent from the others, so that soft errors affect only one check bit. This implementation requires 5 additional xor gates bringing the total cost to 24 gates with a 26% overhead over the original implementation.

In a general case automatic synthesis tools can be used to produce a circuit implementation in which each parity check is independent of the rest by using the appropriate directives that disable logic sharing in the computation of the check bits.

With this strategy, an error on the encoder is in the worst case equivalent to an error on a memory bit. This is so because an error in the encoder will affect the memory only if it occurs during a write operation. This is an adequate protection level as will be shown in the next section, where the proposed approach is compared with TMR in terms of reliability and cost.

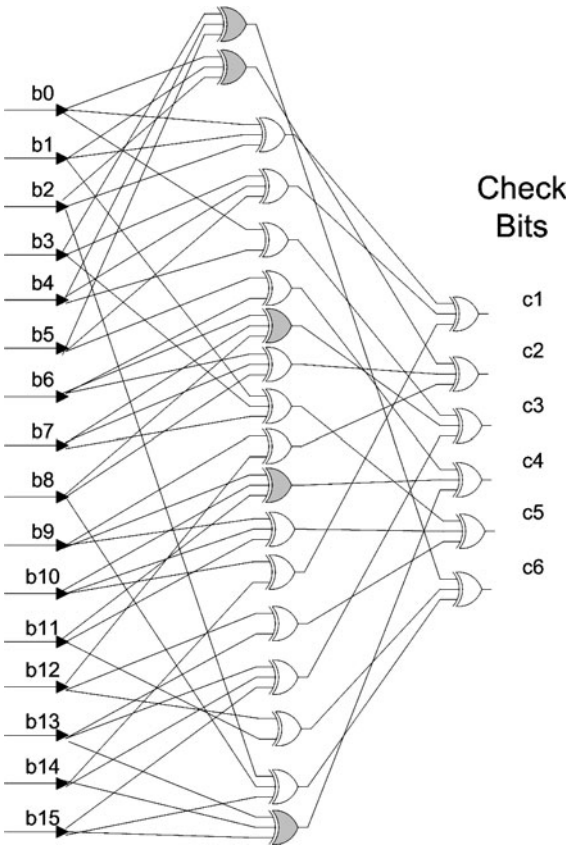

Fig. 4 Illustration of the proposed modified encoder for the (22,16) SEC-DED code (additional gates in *grey*)

3 Comparison with TMR

To compare the reliability of a memory in which the encoder is protected with TMR and another one in which the encoder uses the proposed technique, we assume that the decoder is protected, and therefore it is fault free. This allows us to focus on the reliability of the encoder and memory cells.

For a memory protected with TMR, soft errors on the encoder would have no effect unless at least two occur on the same clock cycle. If that is the case, the memory would probably have failed before, as errors that have occurred in different clock cycles could have accumulated in the same word. Therefore, the reliability of the system can be approximated by that of the memory cells. Assuming a memory with M words, the Mean number of Events to Failure (METF) can be approximated when M is large [9] by

$$METF|_{\mathrm{TMR}} \cong \sqrt{\frac{\pi \cdot M}{2}} \tag{1}$$

For the proposed technique, a soft error in the encoder will have at most the same effect as a soft error in a memory word, that is one bit of a word will be corrupted. The rate of soft errors in the encoder can be larger than the rate of soft errors in a memory word due to the larger area of the encoder that makes it

more likely to be hit by a radiation particle. However, this is compensated by the fact that storage cells are more sensitive to radiation effects than combinational logic and also because a soft error on the encoder will only corrupt a memory bit if it occurs when the memory is being written. Therefore, the reliability can be estimated by that of a memory with M+1 words (where the additional word models the encoder), which can be approximated by

$$METF|_{\text{proposed}} \cong \sqrt{\frac{\pi \cdot (M+1)}{2}} \tag{2}$$

In fact this is a conservative estimate in that the encoder can produce only a single error, that is if we write to a word that already has an error the result is a word with only an error (the one introduced by the soft error in the encoder as the previous one is overwritten). For large values of M, it becomes clear that both approaches provide a similar reliability level. The same conclusion is valid when scrubbing is used to increase the METF [9]. It is also interesting to consider the impact on the METF of leaving the encoder unprotected. In that case a soft error in the encoder can cause an uncorrectable error and therefore a failure. If the probability of a soft error occurring in the encoder is similar to that of a soft error occurring in a memory word, the METF due to failures in the encoder would be M+1. This value is larger than that of the METF of the memory when scrubbing is not used (see Eq. 1) so that in that case the impact would be limited. However when scrubbing is used the METF can be much larger than M such that an unprotected encoder can limit the METF of the whole memory. Therefore the encoder should be protected when scrubbing is used.

Table 1 Number of equivalent gates for the different encoder implementations code

Code	TMR			Proposed		
	Area mm^2	Power mW	Delay ns	Area mm^2	Power mW	Delay ns
(22,16)	2,593.374	3.445	0.89	1,006.386	1.367	0.46
(39,32)	5,412.549	7.742	1.06	2,167.599	2.967	0.71
(72,64)	10,786.37	15.696	1.38	4,748.072	6.512	0.73

To analyze the cost of the encoder circuit in terms of area and power consumption, the different codes proposed in [4] have been considered. The encoders have been implemented in VHDL for TMR and the proposed approach. The circuits have then been synthesized for a 180 nm technology, and the results are shown in Table 1. It can be observed that in all cases the proposed approach requires a much smaller area and power than TMR. The proposed technique is also faster, as it does not require the majority logic voting step.

Combining the reliability and cost comparisons, it can be concluded that the proposed approach offers a similar reliability with a significant reduction in cost.

4 Conclusion

In this paper, the effects of soft errors on encoders for Single Error Correction codes have been analyzed. The results show that applying TMR to the encoders is not an efficient approach and that a careful encoder design can provide a similar reliability with a reduced cost in terms of circuit area and power consumption.

References

1. Blaum M, Goodman R, McEliece R (1988) The reliability of single error protected computer memories. IEEE Trans Comput 37(1):114–119
2. Hamming RW (1950) Error detecting and error correcting codes. Bell System Tech J 29:147–160
3. Haraszti TP (2000) CMOS memory circuits. Kluwer Academic Publishers
4. Hsiao MY (1970) A Class of Optimal Minimum Odd-weight-column SEC-DED codes. IBM J Res Develop 14:395–301
5. International technology roadmap for semiconductors (ITRS), edition (2007)
6. Lin S, Costello DJ (2004) Error control coding, 2nd edn. Prentice-Hall, Inc., Upper Saddle River, NJ
7. Naseer R, Draper J (2008) DEC ECC Design to Improve Memory Reliability in Sub-100nm Technologies. Proceedings of the IEEE International Conference on Electronics, Circuits and Systems, September, pp 586–589
8. Nicolaidis M (2005) Design for Soft Error Mitigation. IEEE Trans Device Mater Reliab 5(3):405–418
9. Saleh M, Serrano JJ, Patel JH (1990) Reliability of scrubbing recovery techniques for memory systems. IEEE Trans Rel 39(1):114–122
10. Schrimpf RD, Fleetwood DM (2004) Radiation effects and soft errors in integrated circuits and electronic devices. World Scientific Publishing