ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A Lossless Implementation of Fault Detection Design for High Speed Memory Applications

M.Lakshmi Sarada*, K.Jhansi Rani

* PG student,M.Tech-instrumentation and control systems,JNTUK,Kakinada,AP,India.
Assistant Professor, Department of ECE, JNTUK, Kakinada,A. P, India.

Abstract

The technology advancement scaling to Reliability,Availability and Serviceability are the three important parameters to be satisfied by any application. With further reduction in transistor size that leads to smaller dimensions, higher integration densities, and lower operating voltages, the reliability of memories is put into jeopardy. Single event upsets (SEUs) altering digital circuits are becoming a bigger concern for memory applications,and also to prevent errors from causing data corruption, memories are typically protected with error correction codes(ECC). An advanced error correction codes are used when an additional protection is needed. The majority logic decoder /detector codes are used for memory application because of correcting large number of errors, less decoding time, area consumption. The proposed fault-detection method significantly reduces memory access time when there is no error in the data read. The technique uses the majority logic decoder itself to detect failures, which makes the area overhead minimal and keeps the extra power consumption low. This technique will tend to correct burst errors of any length

Keywords:Single event upsets(SEUs),Memory,eError correction codes(ECC),Majority logic detector/decoder.

Introduction

Memory system are protected against transient upsets of data bits using ECCs. Cache memory is designed using static random access memory (SRAM) because of its superior access speed.Memory applications should be protected from all kinds of faults for reliable performance. Those faults can be detected using Error Correction Codes (ECC). The reliability, availability, and serviceability (RAS) of the system to perform to customer expectations are a strong function of how the system is designed to respond to hard and soft failures. Soft errors or failures are defined as any upset of a semiconductor device that does not lead to permanent damage.

Existent majority logic decoding (MLD) solutions

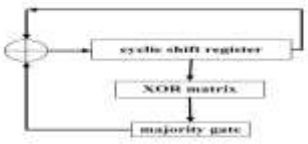
1) Plain ML Decoder

As described before, the ML decoder is a simple and powerful decoder, capable of correcting multiple random bit-flips depending on the number of parity check equations.

It consists of four parts:

i. a cyclic shift register;
ii. An XOR matrix;
iii. A majority gate; and
iv. An XOR for correcting the codeword bit under decoding

majority logic detector



The input signal is initially stored into the cyclic shift register and shifted through all the taps. The intermediate values in each tap are then used to calculate the results of the check sum equations from the XOR matrix. In the cycle, the result has reached the final tap, producing the output signal (which is the decoded version of input). As stated before, input might correspond to wrong data corrupted by a soft error. To handle this situation, the decoder would behave as follows. After the initial step, in which the codeword is loaded into the cyclic shift register, the decoding starts by calculating the parity check equations hardwired in the XOR matrix. The

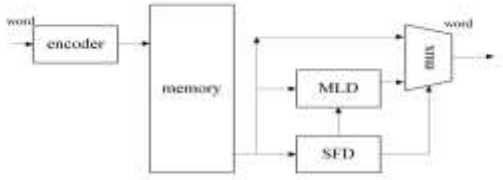
ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

resulting sums are then forwarded to the majority gate for evaluating its correctness. If the number of 1's received in is greater than the number of 0's, that would mean that the current bit under decoding is wrong, and a signal to correct it would be triggered. Otherwise, the bit under decoding would be correct and no extra operations would be needed on it In the next step, the content of the registers are rotated and the above procedure is repeated until all codeword bits have been processed. Finally, the parity check sums should be zero if the codeword has been correctly decoded.

Plain MLD with Syndrome Fault Detector (SFD)

In order to improve the decoder performance, alternative designs may be used. One possibility is to add a fault detector by calculating t he syndrome, so t hat only faulty code words are decoded [11]. Since most of the code words will be error-free, no further correction will be needed, and therefore performance will not be affected. Although the implementation of an SFD reduces the average latency of the decoding process, it also adds complexity to the design (see Fig. 4). T he SFD is an XOR matrix that calculates the syndrome based on the parity check matrix. Each parity bit results in a syndrome equation. Therefore, the complexity of the syndrome calculator increases with the size of the code. A faulty code word is detected when at least one of the syndrome bits is "1." This triggers the MLD to start the decoding, as explained before. On the other hand, if the codeword is error-free, it is forwarded directly to the output, thus saving the correction cycles. In this way, the performance is improved in exchange of an additional module in the memory system: a matrix of XOR gates to resolve the parity check matrix, where each check bit results into a syndrome equation. This finally results in a quiet complex module, with a large amount of additional hardware and power consumption in the system

Memory system schematics for ML decoder with SFD



Proposed ML Detector / Decoder

The existing methodology is based on using Difference-set Cyclic Codes (DSCCs). This code is part of the LDPC codes, and, based on their attributes, they have the following properties:

- ability to correct large number of errors;
- sparse encoding, decoding and checking circuits synthesizable into simple hardware;
- modular encoder and decoder blocks that allow an efficient hardware implementation;
- Systematic code structure for clean partition of information and code bits in the memory.

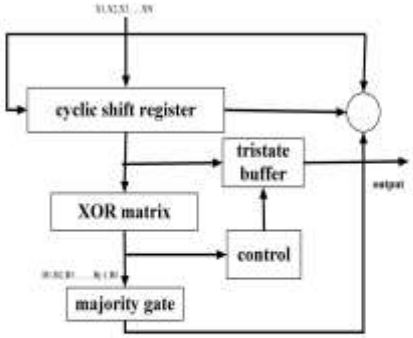
An important thing about the DSCC is that its systematical distribution allows the ML decoder to perform error detection in a simple way, using parity check sums However, when multiple errors accumulate in a single word, this mechanism may misbehave, as explained in the following.

In general, the decoding algorithm is still the same as the one in the plain ML decoder version. The difference is that, instead of decoding all codeword bits by processing the ML decoding during cycles, the proposed method stops intermediately in the third cycle.

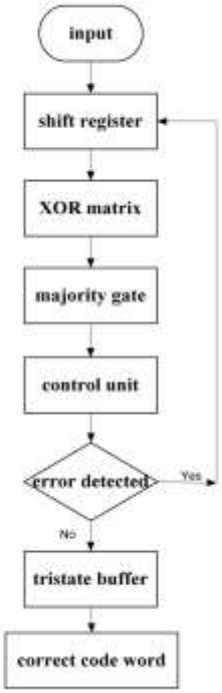
If in the first three cycles of the decoding process, the evaluation of the XOR matrix for all {Bj} is" 0," the codeword is determined to be error-free and forwarded directly to the output. If the {Bj} contain in any of the three cycles at least a "1," this existing method would continue the whole decoding process in order to eliminate the errors

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

MLDD DESIGN



The additional hardware to perform the error detection is illustrated in above figure as:
i) The control unit which triggers a finish flag when no errors are detected after the third cycle and
ii) The output tristate buffers.
The output tristate buffers are always in high impedance unless the control unit sends the finish signal so that the current values of the shift register are forwarded to the output. The control unit manages the detection process. It uses a counter that counts up to three, which distinguishes the first three iterations of the ML decoding. In these first three iterations, the control unit evaluates the by combining them with the OR1 function. This value is fed into a three-stage shift register, which holds the results of the last three cycles. In the third cycle, the OR2 gate evaluates the content of the detection register. When the result is "0," the FSM sends out the finish signal indicating that the processed word is error-free. In the other case, if the result is "1," the ML decoding process runs until the end.



Flow chart of MLDD algorithm

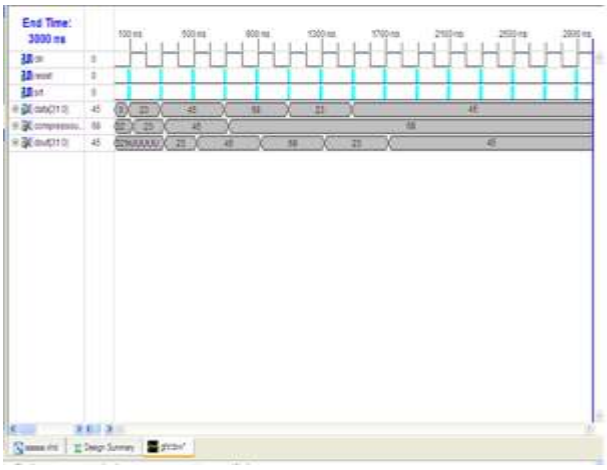
Initially the input is stored into the cyclic shift register and it shifted through all the taps. The intermediate values in each tap are given to the XOR matrix which is used to perform the checksum equations. The resulting sums are then forwarded to the majority gate for evaluating its correctness. If the number of 1's received is greater than the number of 0's which would mean that the current bit under decoding is wrong, so it move on the decoding process of t It is used to produce the accurate result of the MLDD. Otherwise, the bit under decoding would be correct and no extra operations would be needed on it. Decoding process involving the operation of the content of the registers is rotated and the above procedure is repeated and it stops intermediately in the third cycle. If in the first three cycles of the decoding process, the evaluation of the XOR matrix for all is "0," the code word is determined to be error-free and forwarded directly to the output. If the error contains in any of the three cycles at least a "1," it would continue the whole decoding process in order to eliminate the errors.Finally, the parity check sums should be zero if the code word has been correctly decoded.

.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Simulation results of Xilinx

The behavioural simulation for fault secure encode /decoder,input is theinformation vector and output is the dectector which detects error in encoder.first information is given to the encoder and the error are retrived for n-bit length .



Conclusion

In this paper fault detection mechanism ,mldd has been presented based on ml decoding ,exhaustive simulation results shows that the proposed techniques is able to detect any pattern of n-bit length.This improves the of the design with respect to the traditional MLD approach,MLDD error detector modulehas beendesigned an a way that it is independent of code size

Acknowledgement

I am greatful to **K.Jhansi Rani, Associate Professor,JNTUK** for her constantly encouragement and I whole heartlythank himfor her supportin completion of this paper successfully.

References

1. P. Ankolekar, S. Rosner, R. Isaac, and J. Bredow, “Multi-bit error correction methods for latency-contrained flash memory systems,” IEEETrans. Device Mater.Reliabil., vol. 10, no. 1, pp. 33–39, Mar. 2010.
2. M. A. Bajuraet al., “Models and algorithmic limits for an ECC-basedapproach to hardening sub-100-nm SRAMs,” IEEE Trans. Nucl. Sci., vol. 54, no. 4, pp. 935–945, Aug. 2007.
3. R. C. Baumann, “Radiation-induced soft errors in advanced semiconductor technologies,” IEEE Trans. Device Mater. Reliabil., vol. 5, no. 3, pp. 301–316, Sep. 2005
4. S. Ghosh and P. D. Lincoln, “Low-density parity check codes for error correction in nanoscale memory,” SRI Comput. Sci. Lab. Tech. Rep. CSL- 0703, 2007.
5. Y. Kato and T. Morita, “Error correction circuit usingdifference-set cyclic code,” in Proc. ASP-DAC, 2003, pp. 585–586.
6. T. Kuroda, M. Takada, T. Isobe, and O. Yamada, “Transmission scheme of high-capacity FM multiplex broadcasting system,” IEEETrans. Broadcasting, vol. 42, no. 3, pp. 245–250, Sep. 1996.
7. S. Lin and D. J. Costello, Error Control Coding, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2004
8. J. L. Massey, Threshold Decoding. Cambridge, MA: MIT Press,1963.
9. H. Naeimi and A. DeHon, “Fault secure encoder and decoder for NanoMemory applications,” IEEE Trans. Very LargeScale Integr. (VLSI) Syst., vol. 17, no. 4, pp. 473–486, Apr. 2009.
10. R. Naseer and J. Draper, “DEC ECC design to improve memory reliability in sub-100 nm technologies,” in Proc. IEEE ICECS, 2008, pp. 586– 589.
11. I. S. Reed, “A class of multiple-error-correcting codes and the decoding scheme,” IRE Trans. Inf. Theory, vol. IT-4, pp. 38–49, 1954.
12. Shih-fu Liu ,” Efficient Majority Logic Fault Detection With Difference- Set Codes for Memory Applications”, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 20, NO. 1, JANUARY 2012.