RAM ERROR DETECTION & CORRECTION USING HVD IMPLEMENTATION

Narinder Pal Singh, M.Tech (Final Year)
Sukhjit Singh, Assistant Prof.
ECE Department, GTBKIET, Malout, India
Vikrant Sharma, Assistant Prof.
Amandeep Sehmby, M.Tech (Final Year)
ECE Department, CTIEMT, Jalandhar, India

Abstract

Data that is either transmitted over communication channel (e.g. bus) or stored in memory is not completely error free. RAM memory cell contents can change spuriously due to some electromagnetic interference. In magnetic storage devices such as disks, magnetic flux density increases could cause one or more bits to flip (change that value). Exposure to high speed α ray particles is a prominent problem in all the semiconductor memories used for various communication applications. So, in this paper, an error detection and correction method to protect the RAM against the errors is proposed. This method is based on 2-d parities. The parity bits are calculated at the transmitter end for each row, column and diagonal in slash and backslash directions in a memory array. The parities are regenerated at the receiver end. The comparison of transmitted and received parity bits detects the error. As soon as the error is detected, the code corrects the detected error. This method is a promising technique to detect and correct errors in semiconductor memories in presence of large electromagnetic interference with less computational complexity.

Keywords: HVD (Horizontal Vertical Diagonal), EDAC (Error Detection and Correction), Xilinx, Modelsim

1. Introduction

RAM memory may provide increased protection against soft errors by relying on error correcting codes. Such error-correcting memory, known as *ECC* or *EDAC-protected* memory, is particularly desirable for high fault-tolerant applications, such as servers, as well as deep-space applications due to increased radiation. Error-correcting memory controllers traditionally use

Hamming codes, although some use triple modular redundancy. Interleaving allows distributing the effect of a single cosmic ray potentially upsetting multiple physically neighboring bits across multiple words by associating neighboring bits to different words. As long as a single event upset (SEU) does not exceed the error threshold (e.g., a single error) in any particular word between accesses, it can be corrected (e.g., by a single-bit error correcting code), and the illusion of an error-free memory system may be maintained.

Transient errors and permanent faults in memory chips are well known reliability issues in computer systems. Error detection and correction (EDAC) codes, also called error-correcting codes (ECCs) are the prevailing solution to this problem. Typically, to accommodate extra bits the memory bus architecture is extended and to detect and correct errors the coding and checking circuitry is added. But due to cost considerations this additional hardware can be sometimes removed. Hardware redundancy techniques, such as duplication or triple modular redundancy (TMR), can be one solution, but they are very expensive.

Bit-flips caused by single event upsets (SEUs) is a major problem in memory chips and use of EDAC codes remained an effective solution to this problem. To implement these codes on hardware, extra memory chips and encoding/decoding circuitry is required. In systems where EDAC hardware is not available, but when the hardware support is not available then the protection is provided through software. Software implemented EDAC could be a better choice than hardware EDAC, because it can be used with a simple memory system and it provides the flexibility of implementing more complex coding schemes.

Single event upsets (SEUs), power fluctuations or electromagnetic interference can be the reason for soft errors. As process technology scales down to small nanometres, high-density, low cost, high performance integrated circuits, characterized by high operating frequencies, low voltage levels and small noise margins will be increasingly susceptible to temporary faults. Moreover, single-event upsets (SEUs) and single-event transients (SETs) generated by atmospheric neutrons and alpha particles severely impact field-level product reliability, not only for memories, but also for logics in very deep sub-micron technologies. When these particles hit the silicon bulk, they create minority carriers which if collected by the source/drain diffusions, could change the voltage level of a node.

SEUs undoubtedly are the major concern in space and terrestrial applications but multiple bit upsets (MBU) and multiple event transient (MET) became important problem in designing memories because of these points:

- Scaling down technology to increase the error rate. Therefore the probability of having multiple errors increases.
- After passing a high-energy ion, MBUs, METs can be induced by direct ionization or nuclear recoil.
- The experiments in memories under proton and heavy ions fluxes show the probability of having multiple errors is increased when the size of memory is increased.

Therefore, achieving acceptable reliability levels for modern VLSI chips is a key issue. Unfortunately, packaging and shielding cannot effectively be used to shield against soft errors since they may be caused by neutrons which can be easily penetrate through packages. In order to maintain good level of reliability, it is necessary to protect memory cells using protection codes. Error detection and correction codes play a key role in memory protection and reliability improvement. These codes are usually implemented in hardware, but require extended memory bus architecture to accommodate parity bits and additional encoding/decoding circuitry.

The reliability issue can be solved using other forms of redundancy than hardware redundancy because hardware redundancy schemes like duplication or triple modular redundancies are expensive. In low-cost satellite projects, the reliability of a system can be improved by cost effective software error detection and correction code schemes. An important advantage of using software-based code scheme is that, one can change it dynamically to meet the observed response of the memory devices. This is particularly beneficial when, as is often the case in low-cost satellite design, there is little chance to test the devices prior to flight. Various types of error detection and correction codes are used in computers, communication systems based on constrains exposed by applications. For example, for satellite applications, Hamming code and different types of parity codes are used to protect memory devices and complex codes are not applied due to time constraints and limited resources on board.

2. Proposed Method

The proposed error detection and correction method is called HVD code as the parity bits are applied on the row, column and two diagonals on a data part. In addition to horizontal (H) and vertical (V) parity bits, the diagonal (D) parity bits in two directions can be used as shown in Fig 1.

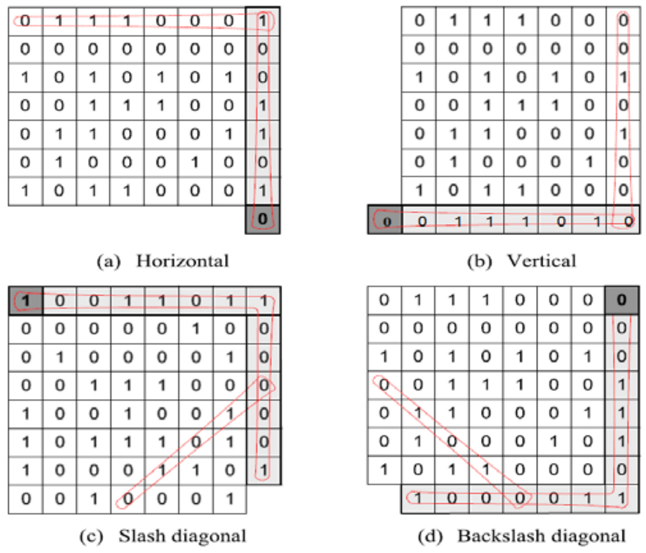

Fig. 1 The parity scheme in HVD

To increase the ability of detection, an additional parity bit is computed on the basis of calculated parity bits of each dimension. In our proposed HVD code implementation, h, v, d and d' represent the number of errors in the horizontal, vertical and slash and backslash lines respectively. Whereas h1, v1, f1 and b1 represent the position of first error in horizontal, vertical and the both diagonal parity lines, respectively.

The proposed method HVD (Horizontal-Vertical-Diagonal) is based on 2-d parities. Parities are generated in the 4 directions that are horizontal, vertical, forward slash and back slash diagonals.

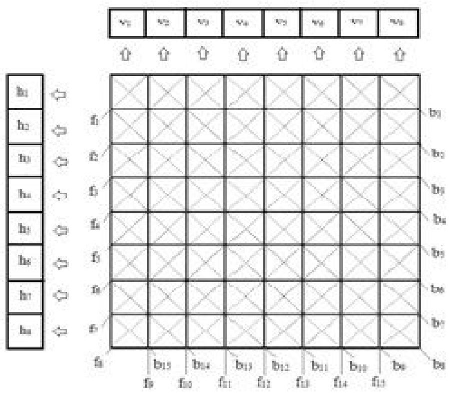

Fig. 2 Two-d coded array

The 8 x 8 array is given in Fig 2 where the symbols h, v, f and b denote the parity bits in horizontal, vertical, forward slash and backward slash respectively and the subscripts indicate the position of parity.

2.1 Detection algorithm

At the receiver end the parities are calculated in all the directions for whole array (for example, from h_1 to h_8 in horizontal direction in the above figure).After the calculation of parities at the receiver the comparison of received parities and the calculated parities is performed. If the result of comparison shows any difference, it means the received data at the receiver is incorrect and correction is required. So the erroneous parity lines are identified and then the correction process starts but if the comparison shows no difference, it means data is correct and no correction is needed.

In order to detect bit upsets in the codeword, at receiver end, all mentioned types of parity bits on the data part of the codeword are required to be calculated again. As the different types of parity bits can be calculated independently in parallel i.e., while computing the horizontal parity bits, vertical parity bits are calculated simultaneously; encoding and consequently detection procedure can be done by means of parallel computation. In real time and high speed applications the above mentioned property becomes very useful. On the other hand, when the first difference between the received parity bits and the calculated ones is detected, the detection algorithm is then stopped and there is no need of further comparison between other calculated parity bits with the received ones.

2.2 Correction algorithm

In this section, we introduce how the data array can be corrected after detecting a probable error. First, it is supposed that three upsets are present only on data part that we need to correct. After receiving a codeword in the receiver side, the parity bits must be computed again. When the differences in calculated and received parity in the vertical, horizontal and both of the diameters dimensions is found and it sets the corresponding syndrome bit to one. As a result, four arrays are produced such as example; the elements of the horizontal array include the number of the rows that has bit upsets. After that, all ordered pairs of them are produced by choosing two different arrays.

Through every ordered pair, there may be maximum one candidate bit present. Note that no candidate bit may be determined from some case of the ordered pairs. These situations are shown in Fig. 3 In the case of Fig 3.3(b) the two parity bits are faulty.

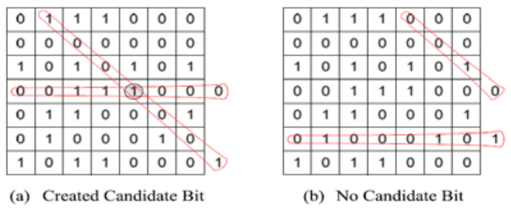

Fig. 3 The created candidate bit and no candidate bit of the intersection

So the backslash and horizontal bits are not connected and there is no candidate bit.

The program follows the following flow given in the Fig 4.

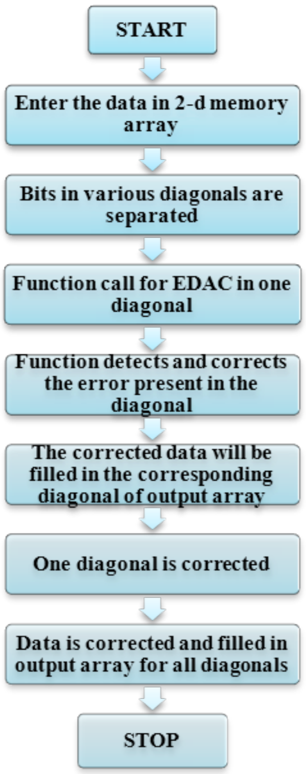
START
Enter the data in 2-d memory array
Bits in various diagonals are separated
Function call for EDAC in one diagonal
Function detects and corrects the error present in the diagonal
The corrected data will be filled in the corresponding diagonal of output array
One diagonal is corrected
Data is corrected and filled in output array for all diagonals
STOP

Fig 4 Flow of EDAC code

Fig. 5 shows the general structure of the hardware used for calculating and updating the parity bits.

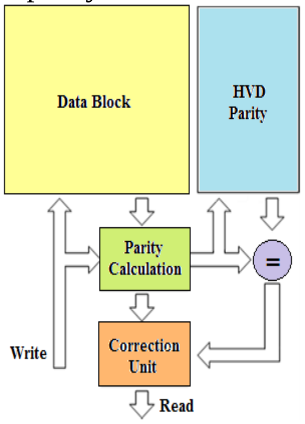

Fig. 5 General hardware structure for the proposed method

Parity bits required for block read and write operations just in one cycle can be calculated by the proposed hardware. Therefore, the performance overhead of the method is significantly low. As this hardware is shared for all of the memory blocks, its area overhead is negligible compared to the other hardware structure used for error detection and correction code.

All parity bits in horizontal, vertical slash and backslash directions are checked in read operation. We assume that the granularity of the memory access is a block. So the numbers of the parity bits to be calculated in the block read and write operations are the same. The number of XOR gates needed is shown in Table 1.

Table 1 The number of the XOR gates used for a given block memory

Parity direction	**Symbol**	**Number of XOR gates**
Horizontal parity	h	7×8+ 7 = 64
Vertical parity	v	7×8+ 7 = 64
Slash diagonal parity	d	(7×6)/2+(6×5)/2+14 =50
Backslash diagonal parity	d’	(7×6)/2+(6×5)/2+14 =50

It is assumed that a dedicated hardware is used to calculate Horizontal, Vertical, Slash and Backslash diagonal parities simultaneously in just one cycle. Meanwhile, in the cases that parity calculation delay can be tolerated, we can use a simpler hardware (i.e. use a hardware having a few number of XOR gates, but with delay penalty) to reduce the leakage power overhead.

While reading a block, all the parity bits in horizontal, vertical and diagonal directions are calculated and compared with the previous parity bits. If in case an incoherence is detected, the correction mechanism will correct the erroneous block and then the correct block will be read. While writing a block, the parity bits in all directions are calculated and written in the appropriate place.

As seen, that the computation process is done using combinational XOR structures in one cycle and thus the performance overhead of the method is reduced in expense of some hardware costs.

3. Results

In this paper an easy method with fewer computations is presented. This method can detect and correct the errors in data bits as well as in the parity bits without any extra calculations. As it is a software EDAC method so no extra hardware circuitry is required and also the transient faults and bit flips in the storage cells in the combinational logic will be immediately corrected, before it can be stored in the storage cells. The simulation tool used here is VHDL ModelSim- XiLinx 8.1. Using Xilinx ISE 8.1, various parameters are analyzed that are discussed as follows.

3.1 Granularity parameter

To show tradeoffs between implementation parameters, a 256 x 256 block of data is considered. HVD method is used to correct errors on this block. Several possible configurations are there to implement the proposed method for a given data block. In the first configuration, HVD is applied to the entire block. In the other one, the block is divided into four separate blocks, called data segments, and then the proposed method is applied on it. In other configurations, the number of separated block increases. As Table 2 shows, the bit overhead rate increases with the number of segments.

Configuration of Segments	**Bit overhead (parity bits / data bits)**
1	0.006
4	0.012
16	0.023
64	0.047
256	0.095

Table 2 Overhead of each block configuration

As shown in above table, overhead increases as we increase the number of blocks increases, that is not recommended but the increment is not linear but it is decreases as we increase the number of blocks.

3.2 Percentage of Redundancy

The simulation results shows that a small increase in the number of parity bits can cause an increase in number of data bits that can be transmitted as a whole codeword as shown in Table 3 and variation is given in Fig 6.

No. of parity bits	**No. of data bits**
3	4
4	8
5	16
6	32
7	64
8	128

Table 3 No. of data bits according to the no. of parity bits

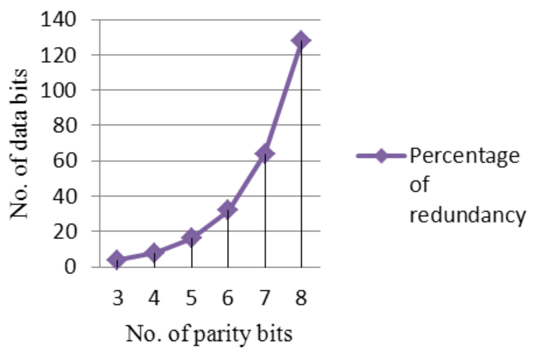

Fig 6 Variation in redundant bits

From the above figure, it is seen that the number of data bits that can be transferred, increases with a small increase as compared to number of parity bits to form code word for transmission.

3.3 Bit overhead

The number of parity bits required to frame a codeword for a number of data bits is called as the Bit overhead. A good error correction and detection method should have lesser bit overhead. With the proposed method, we can get the reduced bit overhead as shown with red line in the Fig 7.

Bit Overhead= number of parity bits/ number of data bits

3.4 Code Rate

The number of data bits that can be transmitted with a number of codeword is termed as the code word. A good error detection and correction method should have higher code rate. In The variation in code rate for different bits is shown with blue line in Fig 7.

Code Rate= number of data bits/ number of bits in code word

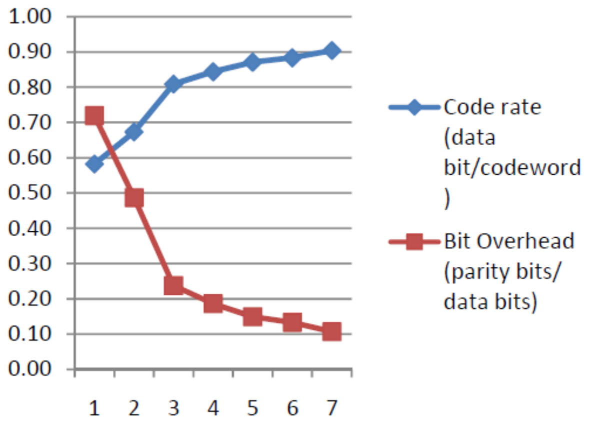

Fig 7 Bit overhead and Code rate for various bits

In the above figure it is clearly visible that with the increase in number of parity bits, bit overhead increases and the code rate decreases which should be a basic requirement for a good error detection and correction scheme.

3.5 Comparison of triple bit error correcting codes

HVD code is compared with other triple error correcting codes such as Golay code, BCH code and Reed-Solomon code. This comparison is based on bit overhead and code rate. It is presented in Fig 8.

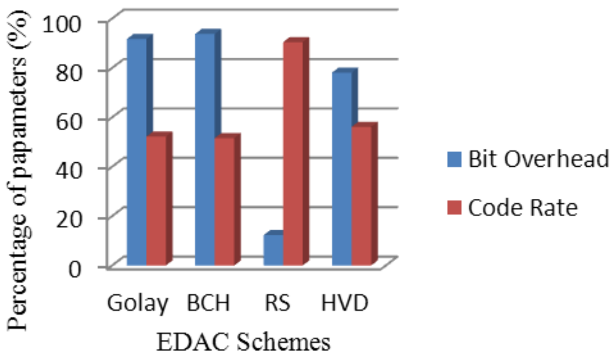

Fig 8 Comparison of triple bit error correcting codes

As seen, that the proposed HVD code has the moderate bit overhead and code rate as compared to the other code schemes.

3.6 Time Analysis

The CPU time are analysed on the same platform for the various word length of 4, 8, 16 and 32 bit. The CPU time for different length of code is given in the Fig 9.

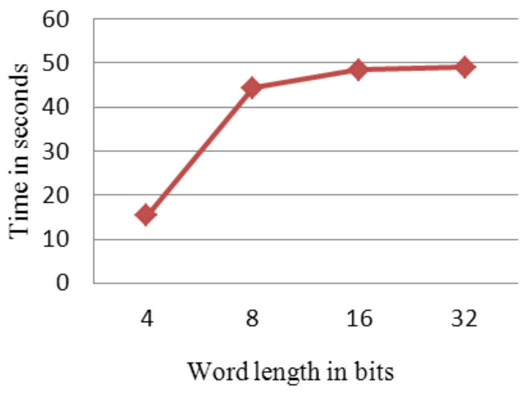

Fig 9 CPU time analysis

The analysis shows as the length of code is increases, the time required to correct the error increases non-linearly.

3.7 EDAC Code

The output for the 8 bit data transmission follows. Error detection and correction is done in all the four directions, and the corrected output is filled in the corresponding position. The output waveform for a sample input is given in the following Fig 10.

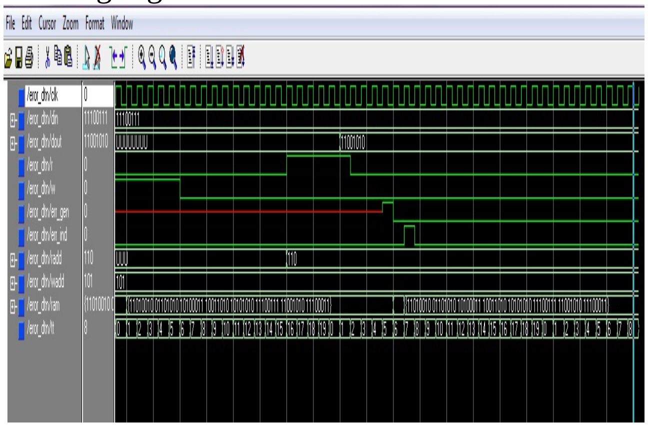

Fig 10 Output waveform for EDAC code

The error detection and correction is done here by using a function discussed in previous section. This function is used for error detection and correction in all the diagonals in all the four direction. This code detects and corrects the errors in all diagonals in the whole memory array simultaneously.

4. Conclusion

In this paper, an easy method of error detection and correction with lesser computations is presented. It is a more simple method to find the errors by removal of candidate bits, so it reduces the complexity. This method can detect and correct the errors in data bits as well as in the parity bits without any extra calculations. As it is a software EDAC method so no extra hardware circuitry is required and also the transient faults and bit flips in the storage cells in the combinational logic will be immediately corrected, before it can be stored in the storage cells. A large combination of multiple faults can be corrected with this method. The number of errors that can be corrected varies with the length of the coded word array. The number increases with the length of the array. In applications for which high error detection coverage is very important, HVD code can be useful. This method can detect and correct a single error in a particular line of code of the coded word array as it uses hamming codes for error detection and correction and it is an only limitation of this method. Another advantage of this code is that it can be used in combination with the other code schemes that has high correction coverage and low detection coverage in comparison with HVD code, so that the number of errors can be increased that can be corrected in a single line.

References:

Fernanda Lima, Luigi Carro, Ricardo Reis “Designing Fault Tolerant Systems into SRAM-based FPGAs” Anaheim, California, USA, DAC’03, June 2-6, 2003**.**

Gaurav Chandil, Priyanka Mishra “Study and Performance Evaluation of Xilinx HDLC Controller and FCS Calculator”IOSR Journal of Engineering (IOSRJEN) e-ISSN: 2250-3021, p-ISSN: 2278-8719 Volume 2, Issue 10 (October 2012), PP 41-50

Fernanda Lima, Luigi Carro, Ricardo Reis “Designing Fault Tolerant Systems into SRAM-based FPGAs” Anaheim, California, USA, DAC’03, June 2-6, 2003**.**

Argyrides C, Zarandi HR, Pradhan DK, “Multiple upsets tolerance in SRAM memory” International symposium on circuits an system, New Orleans, LA, May 2007.

Y. Bentoutou, “Program Memories Error Detection and Correction On-Board Earth Observation Satellites”, World Academy of science, Engineering and Technology 66 2010.

Heesung Lee, Joonkyung Sung, and Euntai Kim, “Reducing Power in Error Correcting Code using Genetic Algorithm”, World Academy of Science, Engineering and Technology 25 2007.

Mostafa Kishani, Hamid R. Zarandi, Hossein Pedram, Alireza Tajary, Mohsen Raji, Behnam Ghavami, “HVD: horizontalvertical-diagonal error detecting and correcting code to protect against with soft errors", 11 April 2011.

Design and VLSI Implementation of HDLC Controller” International Journal of Engineering and Technology Volume 2 No. 10, October, 2012 ISSN: 2049-3444.

Zainalabedin Navabi, ”VHDL Modular Design and synthesis of cores and systems” 3rd edition ,Tata McGRAW-Hill private limited, New Delhi, 2011.