WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

SOFT ERROR DETECTION AND CORRECTION

Akshay K J, Anoki S, Anusha R Sajjanshettar, Basavaraj R M, Prof. Vibha T G

Department of ECE, Dayananda Sagar College of Engineering, Bangalore, India

ABSTRACT:

Error Correcting Codes are used to correct errors when data is stored inside a memory or when data is transmitted over a noisy channel. Use of ECC in memory increases the reliability of memory and hence can be used in harsh environment prone to alpha and neuron particles. Extended radiation creates soft errors which are turning into an undeniable critical issue. Single error correction codes which can detect and correct one bit error per memory are currently in use at present day. But as the technology scale and cell interval distance decreases, the number of affected bits can easily extend to two bits or more. The previous methods are not enough to satisfy the reliability requirement of application of application in harsh environment.

Keywords: Weather Prediction, forecast accuracy

Introduction:

As semiconductor size decreases from submicron to ultra-submicron, the critical charge keeps decreasing and the area of the memory cell scales down for each successive technology node. This makes more memory cells to get affected by a particle hit (atmospheric neutron and alpha particles). Hence there is a need for developing more efficient algorithm with capabilities to detect and correct multiple bit upset (MBU). Error correcting codes are used for detection and correction of errors when stored data is retrieved from the memory or messages are transmitted over a noisy channels. Error in memory can be classified as- soft errors: can be caused due to ion strike, due to cosmic events, power issues, running out of specifications etc. this causes the data stored to flip. Hard errors: results from permanent physical flaw in the module caused by hardware failure. External radiation creates soft errors which degrades the reliability of the memory. The use of error correcting codes (ECC) with advance correction capabilities is a common system-level strategy to harden the memory against bit flips. Much progress in scaling of process technologies down to the deep submicron domain has paved way for significant increase in the level of integration and performance of modern VLSI chips but this has led to the rise of multiple bit upsets and multiple event transient.so in order to tackle multiple bit upsets, our project aims to incorporate a novel HVPDH(horizontal and vertical parity diagonal hamming) coding method which will be effective in detecting and correcting all combinations of 3,4 and 5 bit errors. Verilog implementation of 32 and 64bit memory word were designed and compared in terms of various parameters like Bit overhead, Code rate, power and Area.

LITERATURE SURVEY

Error Detecting Codes (EDC) and Error-Correcting Codes (ECC) are two common methods for preserving the integrity of memory blocks. The Single-Error-Correction Double Error Detection (SECDED) coding scheme is used in several conventional memory. In most cases, ECC is applied to the data word for error detection and correction at the expense of a few extra bits. Because of the high computation complexity, the amount of storage space and energy required to enable multi-bit error detection and correction rapidly increases. There is a possibility that one or more bits will flip or shift to an incorrect value, at any moment during the storage or transmission of data. Errors are inaccurate values that occur as a result of a changeless flaw (broken hardware) or a temporary circumstance. Error-correcting codes (ECC)[1] are used to solve this problem and ensure reliable operation. To provide redundant data, additional bits are supplied or kept near the data bits. Hamming code is commonly used for transmitting data of short lengths. Because the redundancy bits are inserted and then cleared subsequently, scaling it for longer data lengths adds a lot of overhead. Improved hamming code strategy is exceptionally adaptable without much overhead, as a result, it's ideal for transmitting large data bitstreams with low overhead bits per data bit ratios.

For detection of up to 8-bit faults and correction of a 1-bit error, all combinations of 2-bit errors, and most combinations of 3, 4, and 5-bit errors in memory, the Horizontal-Vertical Parity and Diagonal Hamming (HVPDH) method is given[2] . The goal is to include an encoder and decoder that is capable of identifying and fixing faults. The encoder and decoder in this design use horizontal, vertical and grouped diagonal hamming parities. When compared to existing approaches, the HVPDH method achieves a higher coding rate, according to the research. A multi-bit error detection and correction technique[3] based on diagonal Hamming is presented to detect errors up to the 8-bit level. It is possible to correct 1, 2, 3, 4, and 5-bit errors. Only a few combinations of 6 and 7 random bit errors, as well as 8-bit burst errors, can be corrected. This technology achieves a high coding rate with less area and latency than other techniques.

III. METHODOLOGY AND IMPLEMENTATION

A. Generation of parity bits

The even parity bits technique is used to generate horizontal and vertical parity bits.

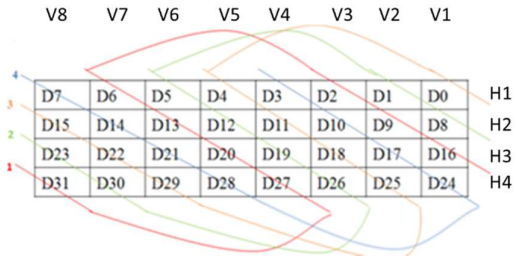

Fig 1. generation of horizontal and vertical parity bits for 32-bit data.

The bit wise XOR operation of each bits in each columns forms vertical parity bits Vn(n=1,2….8) and the bit wise XOR operation of each bits in each rows forms horizontal parity bits Hn(n=1,2,3,4). Hamming parity for each of the groups (with respect to group position) are calculated as below:

Group 1 hamming parity bits
Hm11 = D16 ⊕ D9 ⊕ D6 ⊕ D13 ⊕ D27
Hm21 = D16 ⊕ D2 ⊕ D6 ⊕ D20 ⊕ D27
Hm31 = D9 ⊕ D2 ⊕ D6 ⊕ D31
Hm41 = D13 ⊕ D20 ⊕ D27 ⊕ D31

Group 2 hamming parity bits
Hm12 = D8 ⊕ D1 ⊕ D12 ⊕ D19 ⊕ D30
Hm22 = D8 ⊕ D5 ⊕ D12 ⊕ D26 ⊕ D30
Hm32 = D1 ⊕ D5 ⊕ D12 ⊕ D23
Hm42 = D19 ⊕ D26 ⊕ D30 ⊕ D23

Group 3 hamming parity bits
Hm13 = D0 ⊕ D4 ⊕ D18 ⊕ D25 ⊕ D22
Hm23 = D0 ⊕ D11 ⊕ D18 ⊕ D29 ⊕ D22
Hm33 = D4 ⊕ D11 ⊕ D18 ⊕ D15
Hm43 = D25 ⊕ D29 ⊕ D22 ⊕ D15

Group 4 hamming parity bits
Hm14 = D3 ⊕ D10 ⊕ D24 ⊕ D28 ⊕ D14
Hm24 = D3 ⊕ D17 ⊕ D24 ⊕ D21 ⊕ D14
Hm34 = D10 ⊕ D17 ⊕ D24 ⊕ D7
Hm44 = D28 ⊕ D21 ⊕ D14 ⊕ D7

Same way horizontal and vertical parity along with diagonal hamming bits are calculated for 64 bits data block.

B. Encoding unit

Encoding unit takes m bits of user data as a input and produces m/8 horizontal parity bits, 8 vertical parity bits and (m/8) *4 diagonal parity bits. where m=32,64.
Horizontal (h1), vertical (v1) and diagonal hamming (d) parity bits generated from encoding unit is stored along with the user data(m) in the memory.

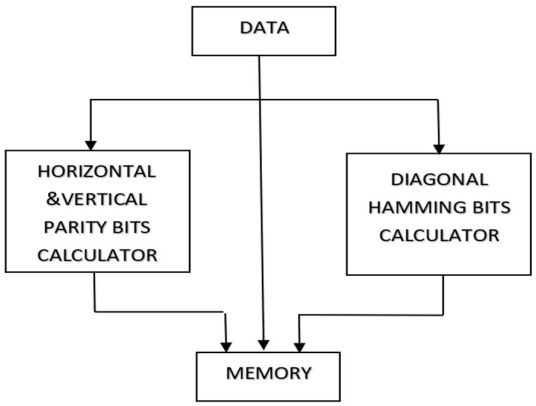

Fig 2. Encoding unit flowchart

C. (a) Decoding Unit

Decoding unit takes corrupted data(m1) and the diagonal hamming bits(d) as an input from the memory.

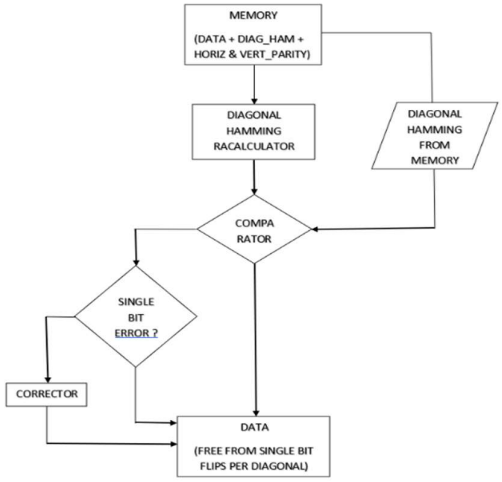

Fig 3. Decoding unit flowchart

It recalculates diagonal hamming (d1) from the corrupted data m1. By comparing (d) and (d1) it rectifies only single bit errors in each grouped diagonal bits and produces rectified data (m2) as an output.

(b) Error detector

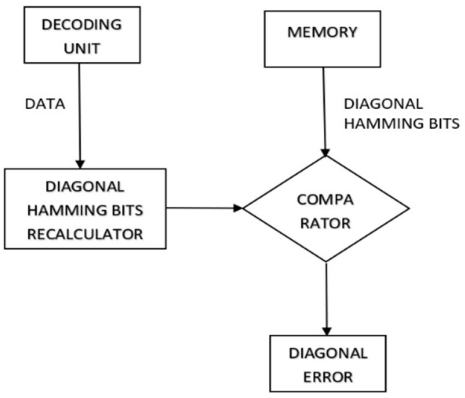

Fig 4. Error detector flowchart

Error detector takes rectified data (m2) from decoding unit and diagonal hamming (d) from memory as an input.

Error detector recalculates the diagonal hamming bits (d2) and compares it with (d) to find the position of errors in the grouped diagonal bit data and produces Diag_Error as an output.

(c) Syndrome calculator

Syndrome calculator takes horizontal parity (h1) bits, vertical parity (v1) bits from the memory and rectified data(m2) from the decoding unit.

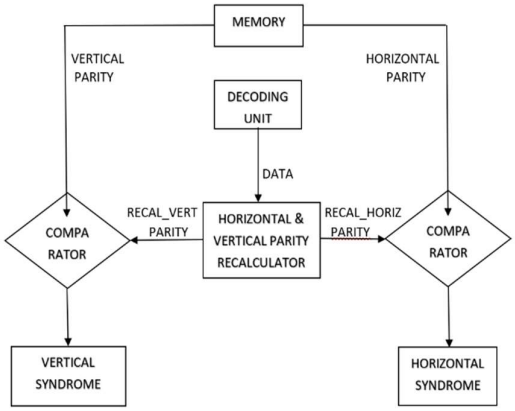

Fig 5. Syndrome calculator flowchart

Syndrome calculator recalculates horizontal parity (h2), vertical parity (v2) and compares it with (h1), (v1) to find error position in each row and columns data bit. And produces Horiz_Syndrome and Verti_Syndrome as an output.

(d) Error corrector

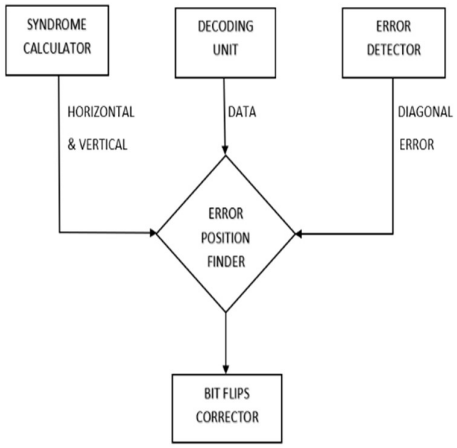

Fig 6. Error corrector flowchart

Error detector takes Diag_Error from Error detector and Horiz_Syndrome, Verti_Syndrome from Syndrome calculator as an input. Error detector detects error position using Diag_Error, Horiz_Syndrome and Verti_Syndrome. It then flips the bit present in the error position. And produces corrected data(m) as an output.

IV. RESULT AND DISCUSSION

The proposed HVPDH method is coded in Verilog HDL and simulation is performed using Xilinx ISE and the output is verified for various random test cases. This paper compares the error correction capabilities of the memory word data using HVPDH codes with various error control coding techniques, and it also compares the code rate and bit overhead of 32-bit and 64-bit words. Reliability is measured in terms of the number of errors that can be rectified, the code rate, and the percentage of bit overhead. The HVPDH method can detect and correct up to four-bit errors, as well as any combinations of two-bit errors and maximum combinations of three and four-bit errors.

Data bits (k)	**Parity bits (r)**	**Codeword N=k+r**	**Code rate (%)**	**Bit Overhead (%)**
32	28	60	53.3	87.2
64	48	112	57.1	75

Table 1. Analysis of HVPDH method with increase in Data word

V. CONCLUSION

HVPDH method can detect up to four-bit errors and correct 1-bit errors as well as any combinations of two-bit errors and maximum combinations of three and four-bit errors. The Horizontal Vertical Parity and Diagonal Hamming (HVPDH) method for 32-bit give bit overhead and code rate of 87.3% and 53.3% respectively. This method is comparative more efficient than the Decimal Matrix code (DMC) and Modified Decimal Matrix code (MDMC) which have bit overhead and code rates of 112% and 47% for DMC,100% and 50% for MDMC respectively. The 64-bit HVPDH design gives a bit overhead rate of 75% and a code rate of 57.1%. In this HVPDH method, the same reliability is achieved by reducing the number of parity bits. The proposed scheme can be improved for reliability while preserving the same bit overhead in future work.

REFERENCES:

1. Raymond Irudayaraj I , Abdul Lateef Haroon P.S, Ulaganathan J, Shridhar S. Bilagi, "Design And Verification of Improved Hamming Code (Ecc) Using Verilog", International Journal Of Electrical, Electronics And Data Communication, Aprl.-2017.
2. Paromita Raha, M Vinodhini, N. S. Murty," Horizontal-Vertical Parity and Diagonal Hamming Based Soft Error Detection and Correction for Memories", International Conference on Computer Communication and Informatics (ICCCI -2017), January 2017.
3. G. Manoj Sai, K. Mohan Avinash, L. Sri Ganesh Naidu, M. Shiva Rohith and M.Vinodhini," Diagonal Hamming Based Multi-Bit Error Detection and Correction Technique for Memories", International Conference on Communication and Signal Processing,July 2022.
4. Jiaqiang Li,Pedro Reviriego,Liyi Xiao,"Low Delay 3-Bit Burst Error Correction Codes", Journal of Electronic Testing, Springer, March 2019.
5. Vishal Badole and Amit Udawat, "Implementation of Multidirectional Parity Check Code Using Hamming Code for Error Detection and Correction", International Journal of Research in Advent Technology, , May 2014.
6. Black, P. E. Dodd, and K. M. Warren, "Physics of multiplenode charge collection and impacts on single-event characterization and soft error rate prediction," IEEE Trans. Nucl. Sci., vol. 60, no. 3, pp. 1836–1851, Jun. 2013.
7. Ahilan A. and Deepa P., "Modified Decimal Matrix Codes in FPGA Configuration Memory for Multiple Bit Upsets", International Conference on Computer Communication and Informatics (ICCCI), pp. 1-5, Jan. 2015.
8. Jing Guo, Liyi Xiao, Zhigang Mao and Qiang Zhao, "Enhanced Memory Reliability against Multiple Cell Upsets Using Decimal Matrix Code", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 22, Issue:1, pp. 127-135, Jan. 2014.