Characterization of Error Correction IP Core for SDRAM Controller

[1]Lakshmi V V V Prasad K, [2]Pitcheswara Rao Nelapati, [3]Nandakumar. R

[1,2,3,4]National Institute of Electronics and Information Technology (NIELIT), Calicut, Kerala, India

Abstract

This paper describes an Error-Correcting Code (ECC) block for use with the DDR and DDR2 SDRAM controller Cores. The ECC block comprises an encoder and a decoder cum corrector, which can detect and correct single-bit errors and detect double-bit errors. It is a cost-efficient solution that is fast, has a low latency, no detrimental effect on system performance, and uses minimal system resources. The ECC block uses an 8-bit ECC for each 64-bit message. Fully parameterized Hamming code ECC block with 8-bit ECC for 64-bit message Configurable latency of 1 or 2 clock delay during writes and 2 or 3 clock delay during reads. Detect any single- and double bit error on the first clock and correct any detected single-bit errors on the second clock.

Keywords

Error Correction, Hamming Code, SEC-DED, Parity, Encoding, Syndrome, Mask, Decoding

I. Introduction

Error correction and detection schemes find use in implementations of reliable data transfer over noisy transmission links, data storage media (including dynamic RAM, compact disks), and other applications where the integrity of data is important. Error correction avoids retransmission of data, which can be degrading the system performance.

II. Single-Bit Error

Only one bit in the data unit has changed. Single-bit error: 0 ® 1 or 1 ® 0

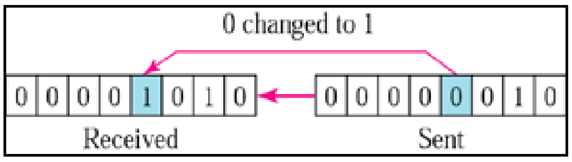


Fig. 1: Single Bit Error

III. Burst Error

Twp or more bits in the data unit have changed

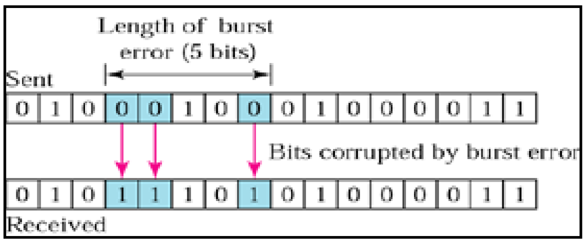


Fig. 2: Burst Error

IV. The Hamming Code

The ECC functions described in this paper are made possible by Hamming code [1], a relatively simple yet powerful ECC code. It involves transmitting data with multiple check bits (parity) [7] and decoding the associated check bits when receiving data to detect errors. The check bits are parallel parity bits generated from XORing certain bits in the original data word. If bit error(s) are introduced in the codeword, several check bits show parity errors after decoding the retrieved codeword. The combinations of these check bit errors display the nature of the error. In addition, the position of any single bit error is identified from the check bits. The Hamming codeword is a concatenation of the original data and the check bits (parity). It is described by an ordered set (d + p, d) where d is the width of the data and p is the width of the parity. The parity matrix [P] can be expressed as: [P] = [D] • [G] where [D] is the data matrix and [G] is the generator matrix. The [G] matrix consists of an identity matrix [I] and a creation matrix [C]. [G] = [I: C]

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

Hamming Codes are powerful ECC codes. For a 64-bit number there are 64 possible one bit errors. 64 different binary permutations can be recognized in a string of length 6 bits. However, another state is needed to represent the case when a detectable error has not occurred. The number of parity bits, m, needed to detect and correct a single bit error in a data string of length n is given by the following equation:

m = log2n +1

The ECC block uses the Hamming code with an additional parity bit, which can detect single and double-bit errors, and correct single-bit errors. The extra parity bit applies to all bits after the Hamming code check bits have been added. This extra parity bit represents the parity of the codeword. If one error occurs, the parity changes, if two errors occur, the parity stays the same. In general the number of parity bits, m, needed to detect a double-bit error or detect and correct a single-bit error in a data string of length n, is given by the following equation:

m = log2n +2

V. Traditional Hamming Codes

The ECC block uses the traditional Hamming code and places the parity bits at the end of the codeword. The extra parity bit is placed at the end of the codeword. Fig. 2 shows the location of the parity bits.

VI. Parity Bit Location

Extra Parity Bit	Parity Bits	Codeword
1	7	63:0

Fig. 3:

VII. Memory Error Correction With Hamming Codes

In automotive applications, software integrity level (ASIL) (memory with error correction capabilities) is one of the important issues in choosing embedded processors. Software- based Hamming codes can be used to improve the reliability of the most important sections of memory, thus improving the ASIL metric. Memory is used to store information various types. Some types of information require storage protection against errors and others do not. For example, application software code, data structures, parameters, and look-up tables are very sensitive and content alteration may end up with catastrophic errors. On the other hand, information such as data samples and image pixels is not as sensitive and may not require error protection.

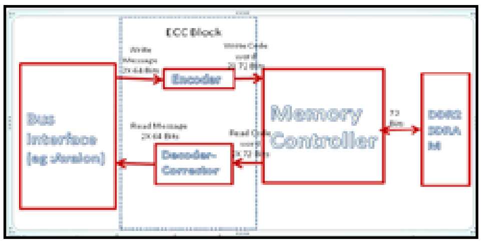


Fig. 4: software-Based Memory Error Correction

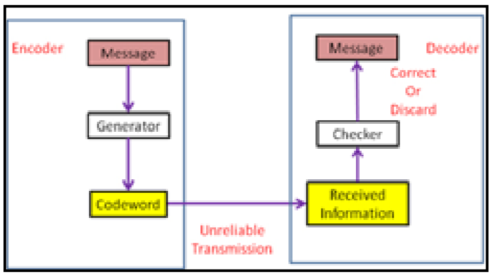


Fig. 5: Error Correction Process

VIII. Ecc Encoder Internal Layout

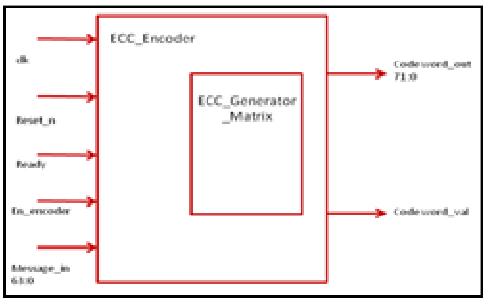


Fig. 6: Encoder Pin Outs

The Hamming encoder generates 8 bits of parity from 64 input data bits. To understand the hamming encoder parity bits generation [5], we arrange the data bits (input) and parity bits (output) in matrix fashion and give binary indexing to each row and column as shown in fig. We have eight columns and nine rows. Each bit in the matrix can be uniquely addressed with row and column index bits. For example the address of D20 is 0011010. Each parity bit P1 to P7 is placed at a special address that is a power of 2 (i.e., P1:0000001, P2:0000010, P3:0000100, P4:0001000, P5:00100000, P6:0100000, P7:1000000).

The parity bit P1 is generated by XORing the data bits that have "1" at position "k" in the address field xxxxxxxk. Similarly the parity bit P2 is generated by XORing the data bits with "1" at the position "k" in the address field xxxxxxkx and so on. The equations for generating parity bits P1 to P2 as follow.

$$p[0] = d[0] \oplus d[1] \oplus d[3] \oplus d[4] \oplus d[6] \oplus d[8] \oplus d[10] \oplus d[11] \oplus d[13] \oplus d[15] \oplus d[17] \oplus d[19] \oplus d[21] \oplus d[23] \oplus d[25] \oplus d[26] \oplus d[28] \oplus d[30] \oplus d[32] \oplus d[34] \oplus d[36] \oplus d[38] \oplus d[40] \oplus d[42] \oplus d[44] \oplus d[46] \oplus d[48] \oplus d[50] \oplus d[52] \oplus d[54] \oplus d[56] \oplus d[57] \oplus d[59] \oplus d[61] \oplus d[63]$$

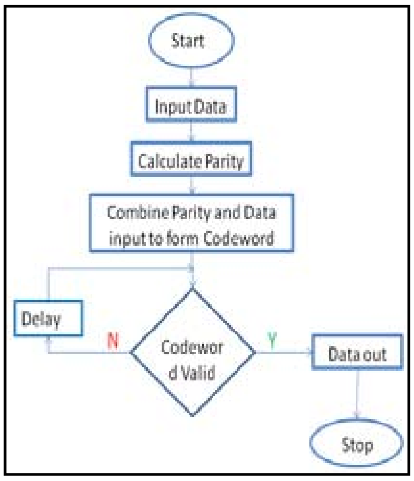


Fig. 7: Encoder Flow Chart

IX. ECC Decoder

The Hamming Decoder consists of two steps

1. Syndrome computation
2. Error Correction

In the syndrome computation step, the Hamming decoder computes eight syndromes using the 72 bits retrieved from memory. The syndromes S1 to S8 are computed by XORing the encoder parity bits P1 to P8 (which retrieved from memory and these parity bits may be different in value due to bit errors) with the decoder parity bits C1 to C8 (which we compute at decoder as follow)

Syndrome <= check $\oplus$ parity

X. Ecc Decoder Internal Architecture

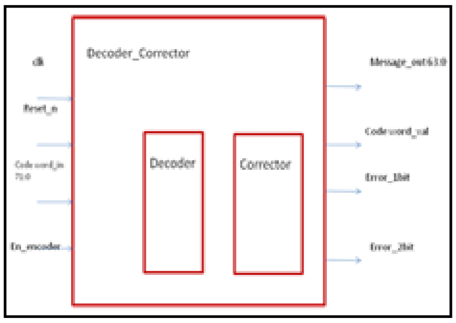


Fig. 8:

XI. Decoder Flowchart

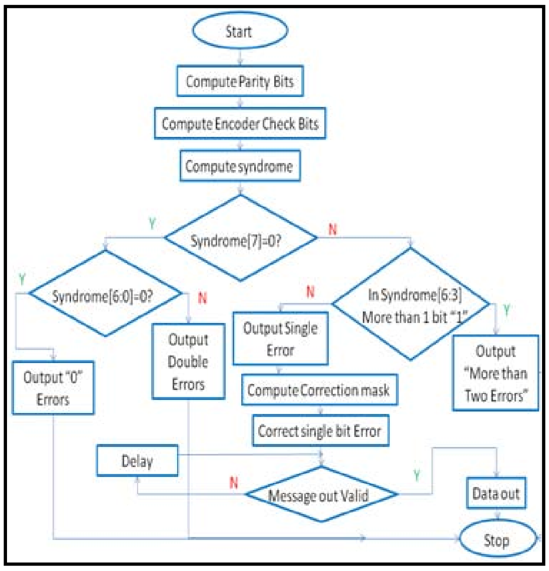


Fig. 9:

data_out_corr<=mask ⊕ decoded

XII. Syndrome Lut and Mask Generation

In order to correct a single bit error, a 64-bit correction mask is created. Each bit of this mask is generated based on the result of the syndrome from previous stage. When no error is detected, all bits of the mask become zero. When a single bit error is detected, the corresponding mask masks out the rest of the bits except for the error bit. The subsequent stage then XORs the mask with the original data. As a result, the error bit is reversed (or corrected) to the correct state. If a double bit error is detected, all mask bits become zero. The error type and corresponding correction mask are created during the same clock cycle.

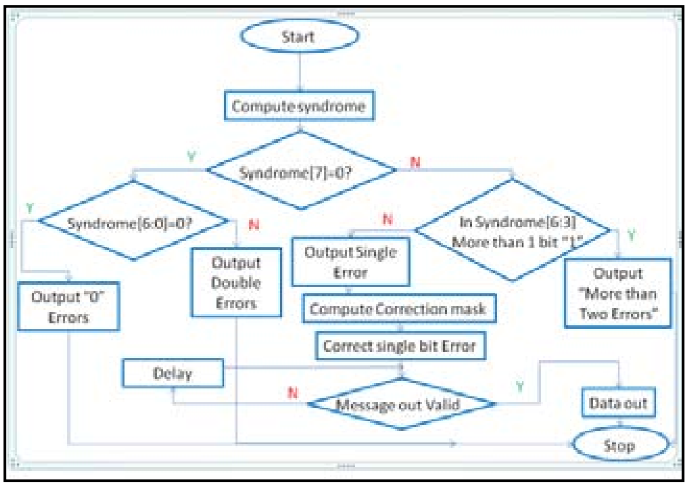


Fig. 10: Syndrome Computation Flowchart

XIII. Error Diagnosis

With respect to retrieved 72 bits from memory, there four possible cases of bit errors

1. No occurrence of bit error
2. Occurrence of 1-bit error
3. Occurrence of 2-bit errors
4. Occurrence of more than 2-bit errors

XIV. The Error Correction is Linked With The Error Detection

- For no errors—no corrections are made.
- For one error—if the error occurs in the parity bit of the codeword, no correction is performed on the message output; if the error occurs in the message, the bit is corrected on the message output.
- For two errors—the ECC block cannot perform an error correction on more than one error. No correction is applied on the output message. It is possible that one or both of the double-bit errors are on the parity bits of the codeword, so either no or one error appears on the output message.
- For more than two errors—the ECC block cannot detect when more than two errors occur in the codeword. So, the ECC block either diagnoses no, one, or two errors. If more than two errors are diagnosed as a single error, the ECC block applies a correction as if the error detected was a one error. This method can lead to corrupt data but is a limitation of this form of ECC.

XV. Simulation Results

A. Encoder

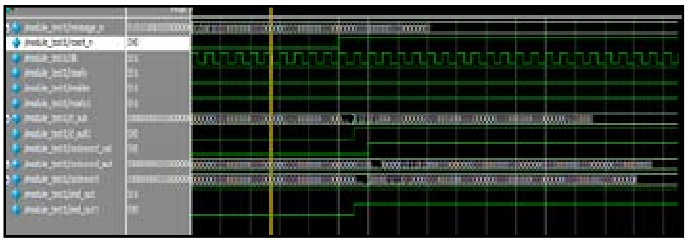


Fig. 11:

B. Decoder

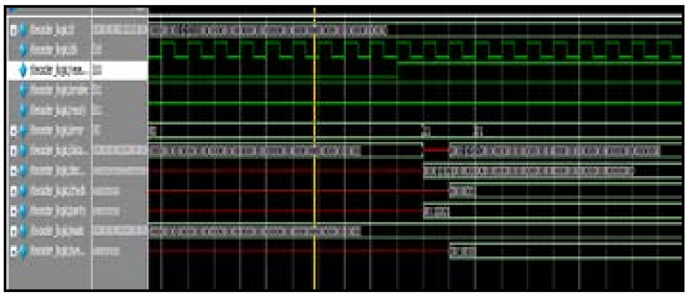


Fig. 12:

C. Power Analysis

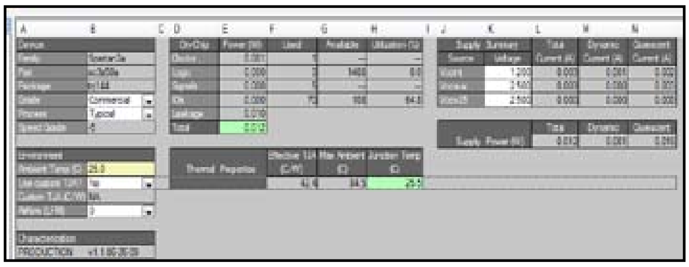


Fig. 13:

XVI. Synopsys Power Report

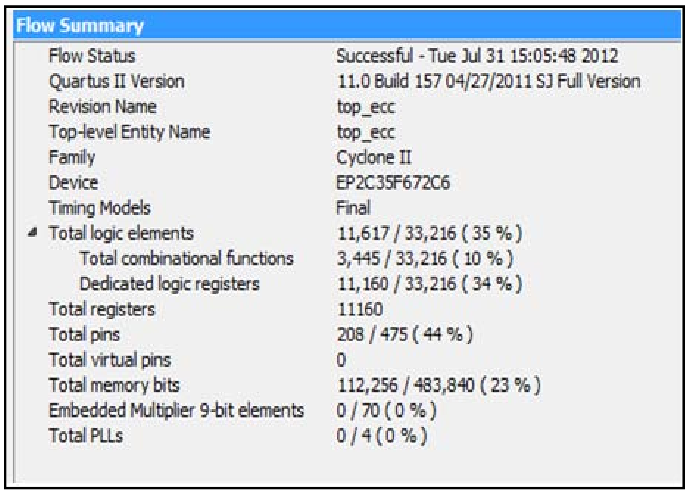
```
***********************************
Report : power
-analysis_effort low
Design : top_ecc
Version: E-2010.12-SP2
Date   : Thu Aug  2 11:58:35 2012
***********************************
```

Library(s) Used:lsi_10k (File: /Tools/Synopsys/syn_vE-2010.12-SP2/libraries/syn/lsi_10k.db)
Information: The cells in your design are not characterized for internal power. (PWR-229)

Operating Conditions:
Wire Load Model Mode: top
Global Operating Voltage = 5
Power-specific unit information :
Voltage Units = 1V
Capacitance Units = 0.100000ff
Time Units = 1ns
Dynamic Power Units = 100nW
(derived from V,C,T units)
Leakage Power Units = Unitless
Cell Internal Power =   0.0000 nW   (0%)
Net Switching Power = 11.7461 uW (100%)

```
Total Dynamic Power   = 11.7461 uW  (100%)
Cell Leakage Power    = 0.0000
***********************************
Report : area
Design : top_ecc
Version: E-2010.12-SP2
Date   : Thu Aug  2 11:56:17 2012
***********************************
Information: Updating design information... (UID-85)
Library(s) Used:
lsi_10k (File: /Tools/Synopsys/syn_vE-2010.12-SP2/libraries/
syn/lsi_10k.db)
Number of ports:   208
Number of nets:    216
Number of cells:   2
Number of combinational cells:  0
Number of sequential cells:     0
Number of macros:  0
Number of buf/inv: 0
Number of references:           2
Combinational area: 2489.000000
Noncombinational area:3205.000000
Net Interconnect area: undefined  (No wire load specified)
Total cell area:          5694.000000
Total area:               undefined
```

XVII. Altera Ecc Top Module Report



Flow Summary

Flow Status	Successful - Tue Jul 31 15:05:48 2012
Quartus II Version	11.0 Build 157 04/27/2011 SJ Full Version
Revision Name	top_ecc
Top-level Entity Name	top_ecc
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	11,617 / 33,216 (35 %)
Total combinational functions	3,445 / 33,216 (10 %)
Dedicated logic registers	11,160 / 33,216 (34 %)
Total registers	11160
Total pins	208 / 475 (44 %)
Total virtual pins	0
Total memory bits	112,256 / 483,840 (23 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Fig. 14:

XVIII. Signal Tap Analyzer Output

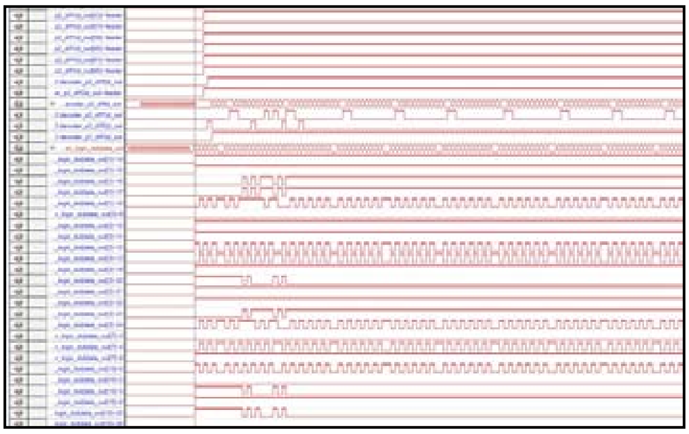


Fig. 15:

XIX. Synopsys Results

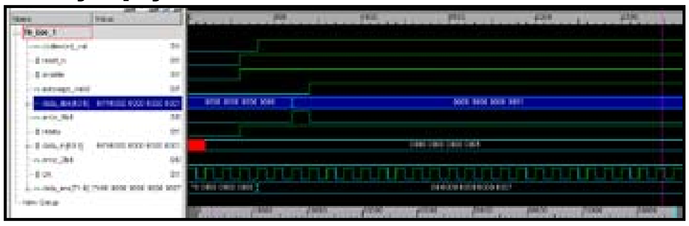


Fig. 16:

XX. Conclusion

The Hamming code concepts can be described in matrix form, where a generating matrix (G) creates valid codewords from information bits, and a check matrix (H) computes syndromes for error checking. When a valid codeword is multiplied by the check matrix, the result (syndrome) [3] is zero. Any non-zero syndrome indicates a bit error. It has been shown that this syndrome depends entirely on an error vector. Each column in the check matrix is different, meaning that all single bit errors of the form codeword produce a unique syndrome and no two-bit errors can produce a zero syndrome. This implies that all single bit errors can be corrected or that all two-bit errors can be detected.

References

[1] R.W. Hamming, “Error Correcting and Error Detecting Codes”, Bell Sys. Tech. Journal, Vol 29, pp. 147-160, April 1950.

[2] M.Y. Hsiao, “A Class of Optimal Minimum Odd-weight-column SECDED Codes”, IBM Journal of R & D Vol. 14, July 1970, pp. 395-401.

[3] S. Lin, D.J. Costello, “Error Control Coding: Fundamental and Applications”, Prentice-Hall, 1983.

[4] C. Berrou, A. Glavieux, P. Thitimajshima, “Near Shannon limit error-correcting coding and decoding,” in Proc. Int. Conf. Communication (ICC’93), Geneva, Switzerland, May 1993, pp. 1064–1070.

[5] R. E. Blahut, "Theory and Practice of Error Control Codes". Reading, MA: Addison-Wesley, 1983.

[6] P. Elias, “Coding for two noisy channels,” in Information Theory, 3rd London Symp., 1955, pp. 61–76.

[7] R. G. Gallager, Low Density Parity-Check Codes. Cambridge, MA: MIT Press, 1963.

Lakshmi VVK Prasad has MSc (Tech) in DSP & ESD and a Post Graduate Diploma in VLSI & Embedded Hardware Design from NIELIT, Calicut, India.

N. Pitcheswara Rao Nelapati, has Masters in Engineering with concentration on VLSI Design from Anna University and a Post Graduate Diploma in VLSI & Embedded Hardware Design from NIELIT,Calicut.