TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 12, No. 10, October 2014, pp. 7253 ~ 7261
DOI: 10.11591/telkomnika.v12i8.5512

Design of ECC Controller and its Validation Based on FPGA

Cao Yan*, Du Lixia, Wang Zhengyu
Lanzhou Jiaotong University, Lanzhou, China
*Corresponding author, e-mail: 191720356@qq.com

Abstract

With the development of embedded systems and the mobile internet, embedded systems are equipped with more and more memory capacity. Memory reliability becomes a focus to us. Though parity checking has been applied in embedded systems, it only can detect errors, but cannot correct them. Therefore, researching better data protection becomes an important topic for the development of embedded systems. In developing a new algorithm and applying it to embedded systems, transplanting good data protection technology used in HPCs are possible solutions. Of these two options, the former requires a long development time, while the latter may be easier to realize and apply, saving time and money. In the HPC field, there are many data protection technologies such as ECC, chipkill, lockstep and so on. Taking the features of the embedded systems into consideration, ECC may be the best technology for transplantation to the embedded system. There are two reasons. First, because of the system's portability, it is hard to support 4 DIMMs. There is no possibility that chipkill or lockstep can be applied in the embedded system. Secondly, ECC is more general and can support all series of SDRAM, either 4-bit or 8-bit. Hence this paper focuses on ECC application on embedded systems. ECC is an advanced technology for memory error detection and correction, which is used to support high reliability of computers. It is widely used in sever memory while it can detect 2-bit error and correct 1-bit error. This paper introduces the ECC algorithm, and then discusses the more general correction coding mathematics, realizing it by programming with VHDL. After finishing these parts, a debug experiment is executed on the Altera Stratix IV family FPGA. Finally, the paper analyzes the simulation results and gives some suggestions for improving the performance of ECC controllers.

Keywords: *ECC, Hamming coding, FPGA, VHDL*

1. Introduction

Memory is an electronic storage device, and all electronic storage devices have the potential to incorrectly return information different from what was originally stored. Some technologies are more likely than others to do this. DRAM memory, because of its nature, is likely to return occasional memory errors. As an important part of the computer, memory errors can cause some unexpected results [1]. With the increase of the use of DRAM memory in modern servers and PCs, memory RAS features are encountered more and more by enterprise and common users [2].

There are two kinds of errors that can typically occur in a memory system. The first is called a repeatable or hard error which is caused by defects within the DRAM package among other things. It cannot be corrected. The second is called a transient or soft error which is mainly caused by charged particles from naturally occurring background radiation or cosmic rays. It can be technologically detected and corrected. Parity checking, as a basic form of error detection, has been adopted in modern PCs. It only can detect 1-bit errors but cannot correct them. This error detection may satisfy PC RAS demands, but it is far from satisfying the demands of server RAS. In modern servers, ECC memory provides a good level of reliability and has become the standard technology today on almost every server.

This thesis first introduces the ECC algorithm, then based on the ECC algorithm, shows a design for an ECC controller, which can be used for memory systems or some ECC related applications, and concludes with a validation of the algorithm on the Altera Stratix IV family FPGA [3-5].

Received December 27, 2013; Revised May 26, 2014; Accepted June 17, 2014

2. ECC Encoding Algorithm

ECC coding is an SEC-DED (Single-bit Error Correction, Double-bit Error Detection) coding which is for obtaining higher reliability. The origin of ECC coding is Hamming coding, which was first proposed by Hamming in 1950.

Hamming code is a code that permits correcting single bit errors. He assumes that the data to be transmitted consists of a certain number of information bits u, and he adds to these a number of check bits p such that if a block is received that has at most a bit error, then p can identify the error bit position (either the error occurred in the block of information bits or in the block of the check bits, it can be identified). Specially, in Hamming code p is interpreted as an integer which is 0 if there is no error, and otherwise is 1-origin index of the bit that is in error. Let k be the number of information bits and m the number of check bits. Because the m check bits must check not only the information bits but also themselves, the value of p must be interpreted as an integer, ranging from 0 to m+k, which has m+k+1 distinct values. Because m bits can distinguish 2m cases, we must have:

Error! Reference source not found.; (1)

This is known as the Hamming rule. It applies to any single bit error correcting (SEC) binary FEC (Forward Error Correcting, just a method that permit the receiver to correct a transmission error without asking the sender for more information about it or for a retransmission) block code in which all of the transmitted bits must be checked. According to Hamming rule, we can calculate that the number of check bits for 64-bit information bits is 7. Based on these added check bits, ECC coding adds another bit for parity check of all bits. Consequently, to a 64-bit information block, ECC coding needs 8 bit check block. The following are the steps to generate a check block p:

1) Calculate the number of check bits according to the Hamming rule;

2) Put the check bits in the power of 2 positions, and put the information bits in the other positions [6];

3) Calculate the check bits the SEC needs. The method is as follows:

a) Let the least significant bit of p be p0. The value of p0 is the parity check result of the bits in the positions 1,3,5,7….(in binary, the least significant of these positions number is 1);

b) Let the next from the least significant bit of p be p1. The value of p1 is the parity check result of the bits in the positions 2,3,6,7,10,11….(in binary, the next from the least significant of the position number is 1);

c) Similarly, let the third from least significant bit of p be p2. The value of p2 is made an even parity check on those positions that have a 1 in their third from least significant position number, namely positions 4, 5, 6, 7, 12, 13, 14, 15, 20, …

d) Continuing, calculate the other check bits in the same way;

4) Finally, add a parity check bit, the value of which is the parity check result of the information bits u and the check bits p.

Table 1 is the check table for 64-bit data ECC encoding.

Table 1. 64-bit Data ECC Encoding

	111	110	101	100	011	010	001	000
1000	D63	D62	D61	D60	D59	D58	D57	CB7
0111	D56	D55	D54	D53	D52	D51	D50	D49
0110	D48	D47	D46	D45	D44	D43	D42	D41
0101	D40	D39	D38	D37	D36	D35	D34	D33
0100	D32	D31	D30	D29	D28	D27	D25	CB6
0011	D25	D24	D23	D22	D21	D20	D19	D18
0010	D17	D16	D15	D14	D13	D12	D11	CB5
0001	D10	D9	D8	D7	D6	D5	D4	CB4
0000	D3	D2	D1	CB3	D0	CB2	CB1	No error

From table 1, we can calculate each check bit (CB1∼CB7), for example,

CB1=D0⊕D1⊕D3⊕D4⊕D6⊕D8⊕D10⊕D11⊕D13⊕D15⊕D17⊕D19⊕D21⊕D23⊕D25⊕D26⊕D28⊕⊕D30⊕D32⊕D34⊕D36⊕D38⊕D40⊕D42⊕D44⊕D46⊕D48⊕D50⊕D52⊕D54⊕D56⊕D57⊕D59⊕D61⊕D63;

After calculating CB1∼CB7 out, we do an overall parity check for D0∼D63 and CB1∼CB7, and the result is CB8.

2.1. ECC Decoding and Correcting Algorithm

As mentioned above, ECC encoding applies an SEC-DED code. So the receiver can detect errors by the check bit. Table 2 demonstrates how the receiver detects errors.As indicated in Table 2, if there are no errors, the overall parity (the parity of the entire n-bit received code word) will be even and the syndrome of the (n-1)-bit SEC portion of the block will be 0. If there is one-bit error, then the overall parity of the received block will be odd. If the error occurred in the overall parity bit, then the syndrome will be 0. If the error occurred in some other bit, then the syndrome will be nonzero and it will indicate which bit is in error. If there is a two-bit error, then the overall parity of the received block will be even. If one of the two errors is in the overall parity bit, then the other is in the SEC portion of the block. In this case the syndrome will be nonzero (and will indicate the bit in the SEC portion that is in error). If the errors are both in the SEC portion of the block, then the syndrome will also be nonzero, although the problem is hard to explain this two bits location [7-10].

Table 2. Receiver Conclusion of Error Detection

Errors	Possibility Overall Parity	Syndrome	Receiver Conclusion
0	even	0	No error
1	odd	=0	Overall parity bit is in error
		≠0	Syndrome indicates the error
2	even	≠0	Double error, cannot be corrected

If there is one-bit error, the correction will be executed when the error is detected. The syndrome will indicate the error location. The syndrome is calculated as follows. After the receiver receives the data, it will calculate the check bits again by the ECC encoding algorithm, if the calculated result is r, then do XOR operation with the check block p and the result r, and the result is syndrome [11].

2.2. More General Considerations of Error Correction.

In the last two sections, the ECC algorithm based on Hamming coding was discussed. This section will take levels of error correction and detection capability greater than SEC-DED into consideration.

The central concept in the theory of ECC is that of Hamming Distance, which is appropriate to call this a distance function because it satisfies the definition of a distance function used in linear algebra:s

$$d(x, y) = d(y, x),$$
$$d(x, y) \geq 0,$$
$$d(x, y) = 0 \quad \text{iff} \quad x = y, \quad \text{and}$$
$$d(x, y) + d(y, z) \geq d(x, z) \quad \text{(triangle inequality)}.$$

Here d(x, y) denotes the Hamming distance between code word x and y, which for brevity we will call simply the distance between x and y.

Based on the above concept, we will now turn this question around and ask, “For a given code length n and minimum distance d, how many code words are possible?”

For minimum distance 1, suppose the code words according to the following forum:

A (n, 1) =2n

For minimum distance 2, we know from the single parity bit example that but A (n, 2) cannot exceed for the following reason. Suppose there is a code of length n and minimum distance 2 that has more than code words. Delete any one column from the code words. This produces a code of length and minimum distance at least 1 (deleting a column can reduce the

minimum distance by at most 1), and of size exceeding. Thus it has contradicting equation (6). Hence

A(n-2)=2n-1

What about the distance 3? That is an unsolved problem, in the sense that no formula or reasonably easy means of calculating it is known. Of course many specific values of are known, and some bounds are known, but the exact value is unknown in most cases. When equality holds in Equation (1), it represents the solution to this problem for the case d=3. Let Equation (1) be rewritten:

$$2^k \le \frac{2^n}{n+1}.$$

Here k is the number of information bits, so is the number of code words. Hence we have:

$$A(n,3) \le \frac{2^n}{n+1}$$

Similarly, for the distance n=7, and so on. An interesting relation is that for $n \ge 2$,

$$A(n,d) \le 2A(n-1,d).$$

This is known as the sphere-packing bound, which can achieve a higher level of error correction and detection capability than SEC-DED [12].

3. ECC Controller Design

Based on the research of ECC the error detection and correction algorithm, with the top-down and modular design methodology, an ECC controller has been designed. Figure 1 is the block diagram of a 64-bit ECC controller and its application in the memory system.

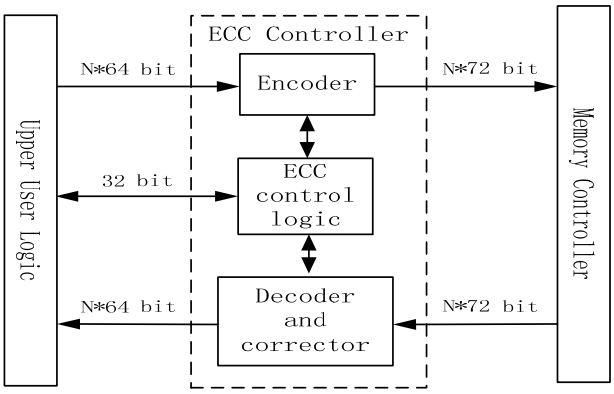

Figure 1. ECC Controller Block Diagram and its Application

From the Figure 1, we can see, the ECC controller consists of an ECC encoder, ECC control logic and ECC decoder and corrector. The ECC encoder is in charge of encoding the information data with check bits, while the ECC decoder and corrector is in charge of decoding the received block and doing error detection and correction. If a two-bit error detected and uncorrectable, the ECC control logic will send an interrupt to the upper user logic, which indicates that an uncorrectable error occurred. ECC control logic consists of many registers which can be divided into three types as follows:

Configuration Register (CR), which is used for recording the threshold of the occurring 1-bit or 2-bit errors. Once over the threshold, the control logic will send an interrupt.

Status Register (SR), which is used for recording the type of error which occurred, error location and ECC syndrome.

Counter, used for recording the number of 1-bit and 2-bit errors.

After the design of ECC controller architecture, each module will be realized by programming with the VHDL language [13-15].

4. FPGA Validation

Validation and simulation is an important step in the EDA design. The purpose of validation is to ensure the design accuracy. After finishing the ECC controller design, a FPGA validation has been executed on the Altera Straix family FGPA. According to the Altera EDA design solution, two development kits (Quartus II 10.1 and ModelSim 6.6c) have been chosen for this validation. At first, a project was created in the Quartus II 10.1, then source files were added to the project and a full compilation made. Figure 2 is the full compilation result [16]. Figures 3 and 4 show the analysis & synthesis result and the fitter result individually.

Flow Status	Successful - Sat May 12 15:17:21 2012
Quartus II Version	10.1 Build 153 11/29/2010 SJ Full Version
Revision Name	ecc
Top-level Entity Name	ecc
Family	Stratix IV
Device	EP4SE530H35C2
Timing Models	Final
◢ Logic utilization	< 1 %
Combinational ALUTs	608 / 424,960 (< 1 %)
Memory ALUTs	0 / 212,480 (0 %)
Dedicated logic registers	64 / 424,960 (< 1 %)
Total registers	64
Total pins	176 / 744 (24 %)
Total virtual pins	0
Total block memory bits	0 / 21,233,664 (0 %)
DSP block 18-bit elements	0 / 1,024 (0 %)
Total GXB Receiver Channel PCS	0
Total GXB Receiver Channel PMA	0
Total GXB Transmitter Channel PCS	0
Total GXB Transmitter Channel PMA	0
Total PLLs	0 / 8 (0 %)
Total DLLs	0 / 4 (0 %)

Figure 2. ECC Controller Full Compilation Result

Analysis & Synthesis Status	Successful - Sat May 12 15:10:51 2012
Quartus II Version	10.1 Build 153 11/29/2010 SJ Full Version
Revision Name	ecc
Top-level Entity Name	ecc
Family	Stratix IV
◢ Logic utilization	N/A
Combinational ALUTs	608
Memory ALUTs	0
Dedicated logic registers	64
Total registers	64
Total pins	176
Total virtual pins	0
Total block memory bits	0
DSP block 18-bit elements	0
Total GXB Receiver Channel PCS	0
Total GXB Receiver Channel PMA	0
Total GXB Transmitter Channel PCS	0
Total GXB Transmitter Channel PMA	0
Total PLLs	0
Total DLLs	0

Figure 3. Analysis & Synthesis Compilation Result

After finishing a full compilation, an RTL simulation is executed in the ModelSim 6.6c. The purpose of the RTL simulation is to find some unstrained trace in the design. Before an RTL simulation, you should design a test bench, using a PCB equipped with your device. Here, the device is just the ECC controller. In addition, a test bench should contain some signal stimulator to drive the circuit to work, which is also designed by VHDL programming [17].

The ECC controller RTL simulation consists of an encoder RTL simulation and decoder RTL simulation. To a memory system, the ECC encoder will work if a processor executes a write memory command. Figure 5 is the simulation result of ECC encoding.

Fitter Status	Successful - Sat May 12 15:13:56 2012
Quartus II Version	10.1 Build 153 11/29/2010 SJ Full Version
Revision Name	ecc
Top-level Entity Name	ecc
Family	Stratix IV
Device	EP4SE530H35C2
Timing Models	Final
Logic utilization	< 1 %
Combinational ALUTs	608 / 424,960 (< 1 %)
Memory ALUTs	0 / 212,480 (0 %)
Dedicated logic registers	64 / 424,960 (< 1 %)
Total registers	64
Total pins	176 / 744 (24 %)
Total virtual pins	0
Total block memory bits	0 / 21,233,664 (0 %)
DSP block 18-bit elements	0 / 1,024 (0 %)
Total GXB Receiver Channel PCS	0
Total GXB Receiver Channel PMA	0
Total GXB Transmitter Channel PCS	0
Total GXB Transmitter Channel PMA	0
Total PLLs	0 / 8 (0 %)
Total DLLs	0 / 4 (0 %)

Figure 4. Fitter Compilation Results

In Figure 3, dn is the input data while dq is the output data, and cb is the check bits calculated by the ECC encoder. Seeing from Figure 5, we can find the value of dq is the same as dn. In a memory system, the ECC decoder and corrector will work if a processor reads data from the memory. Due to memory errors occurring randomly, to better observe the decoder and corrector RTL simulation; some errors were injected into the signal stimulator. Table 3 and table 4 are the VHDL source codes of error injection [18].

Table 3 and 4 individually used a state machine to stimulate the decoder and corrector function. From the source codes, we know the stimulator contained all types of 1-bit and 2-bit errors. After importing the test bench and source files to the Modelsim and doing an RTL simulation, the result is shown as the Figure 6

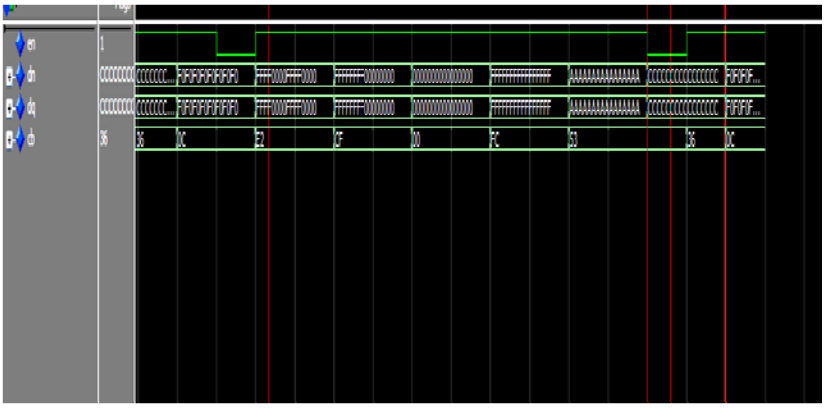

Figure 5. ECC Encoding Simulation Result

Table 3. Errors Injected to the Information Bits

```
data:process
begin
  wait for 40 ns;
   Dp<=X"0000000000000001";  --inject single bit error, Dp(0) flip
  wait for 40 ns;
  Dp<=X"FFFFFFFFFFFFFFEF";  --inject single bit error, Dp(4) flip
  wait for 40 ns;
  Dp<=X"AAAAAAAAAAAAAAAF"; --inject double bit error, both occurred at the data bits. Dp(0),Dp(2) both flip
  wait for 40 ns;
  Dp<=X"CCCCCCCCCCCCCCCD";--inject double bit error; one occurred at the data bit, the other occurred at the
check bit. Dp(0), cb(0) both flip.
  wait for 40 ns;
  Dp<=X"F0F0F0F0F0F0F0F0";  --inject double bit error, both occurred at the check bits, cb(1), cb(2) both flip
  wait for 40 ns;
  Dp<=X"FFFF0000FFFF0000";  --inject single bit error, cb(3) flip
  wait for 40 ns;
  Dp<=X"FFFFFFFF00000000"; -- no error
end process;
```

Table 4. Errors Injected to the Check Bits

```
check_bit: process
begin
--inject single bit error,Dp(0) flip
  wait for 40 ns;
   cb<=X"08";
--inject single bit error,Dp(4) flip
  wait for  40 ns;
  cb<=X"EF";
--inject double bit error; both occurred at the data bits. Dp(0), Dp(2) both flip
  wait for 40 ns;
  cb<=X"58";
--inject double bit error, one occurred at the data bit, the other occurred at the check --bit. Dp(0),cb(0) both flip
  wait for 40 ns;
  cb<=X"30";
--inject double bit error, both occurred at the check bits,cb(1),cb(2) both flip
  wait for 40 ns;
   cb<=X"0A";
--inject single bit error, cb(3) flip
  wait for 40 ns;
   cb<=X"E6";
-- no error
  wait for 40 ns;
  cb<=X"CE";
end process;
```

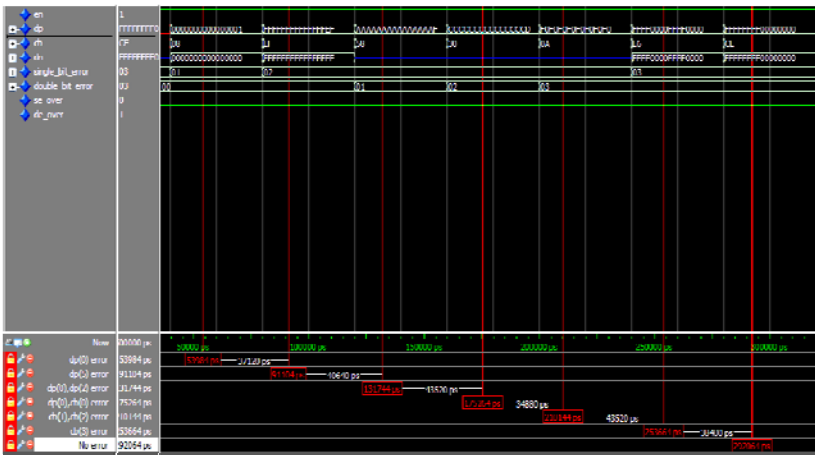

Figure 6. ECC Decoder and Corrector RTL Simulation Result

From Figure 6 we see, all data is decoded to 64-bit original data. All 1-bit errors are corrected, in addition, when it occurred, the single_bit_error register is incremented by 1 automatically. When 2-bit errors occurred, the decoder cannot decode the right data, so the state of output data (dn) is high impedance, which is shown in the period from 131744ps to 210144ps, because 2-bit error occurred 3 times in the period, the double_bit_error count to "0x03" as shown in Figure 6.

In the Figure 6, each signal definition is shown in the Table 5.

Table 5. Signal Definition of ECC Decoder and Corrector

Symbol	Type	Definition
en	Input	Enable the decoder and corrector
dp	Input	information bits of input data，64bit
cb	Input	check bits of input data，8bit
dn	Output	output data，64bit
single_bit_error	Output	1-bit error register, the threshold of which is 3F
double_bit_error	Output	2-bit error register, the threshold of which is 1F
se_over	Output	interrupt when over the threshold of 1-bit error
de_over	Output	interrupt when over the threshold of 2-bit error

5. Improvement Suggestions

From the FPGA validation results of the ECC controller design in this thesis, for better performance, two suggestions are given as follows:

The ECC controller was only validated in 64-bit mode in this thesis, for some application needing support for 128-bit or more, the timing is preliminary and not validated. Therefore, a validation of a 128-bit or more mode ECC controller should be executed after this thesis.

On the flexibility aspect, dynamic reconfigurable architecture may be a good choice for designing an ECC controller which supports 128-bit or more mode. This is an area for improvement.

The ECC controller only can correct 1-bit errors, for some applications which need higher reliability, the ECC controller is inadequate. Therefore, a new data protection technology maybe a good choice for solving such problems. How to design a new data protection technology supporting multiply bits correction will be the next job after this thesis.

6. Conclusion

ECC coding is important for ensuring data reliability. On the aspect of ECC coding algorithm research, this thesis realized a 64-bit ECC controller by VHDL programming and its validation on the Altera Stratix IV Family FPGA. The controller can apply to a memory controller and some other ECC related applications perfectly because it is smart and easy to transplant in an EDA application.

References

[1] Slayman C, Ops A La carte. *Soft error trends and mitigation techniques in memory devices.* The Conference of Reliability and Maintain ability Symposium. 2011.

[2] Li Zhanghui, Ni Xiaoqiang, Wang Yongwen. *Implementation and Desgin Of Error-correcting Codes In High Performance Processor*. The 15th Computer Engineering and Industrial Meeting (Conference). 2011; A: 3-4.

[3] HUA Bin, HUANG Jie-wen, ZHOU Zhang-lun, SUN Jian-tao, ZHANG Ping. Design and Implementation of the Data ECC in High Speed and Large Capacity Solid State Storage System Based on FPGA. *Journal of science of technology and engineering (Publication_Name)*. 2010; 18: 2-3.

[4] Chin-Lung Su, Yi-Ting Yeh, Cheng-wen Wu. *An integrated ECC and redundancy repair scheme for memory reliability enhancement. The Defect and Fault Tolerance in VLSI Systems.* DFT 2005. 20th IEEE International Symposium (Conference). 2005; 81-89.

[5] RW Hamming. Error Detecting and Error Correcting Codes. *Journal of the Bell System Technical (Pulication_Name)*. 1950; XXIX(2): 1-14.

[6] Nikolov H, Stefanov. Efficient External Memory Interface for Multi-Processor Platforms Realized on FPGA Chips. *Journal of IEEE Computer Society (Pulication_Name)*. 2005; 8(3): 23-27.

[7] Feng Qin, Shan Lu, Yuanyuan Zhou. *Exploiting ECC-Memory for Detecting Memory Leaks and Memory Corrution During Production Runs.* High-Performance Computer Architecture. HPCA-11,11th International Symposium (Conference). 2005; 291-302.
[8] Xingjun Zhang, Endong Wang, Dong Zhang etc. *Software-Based Detecting and Recovering from ECC-Memory Faults", In Intelligent Networking and Collaborative System (INcOS).* Third International Conference (Conference). 2011; 715-719.
[9] Doe Hyun Yoon, Mattan Erez. *Memory Mapped ECC: Low-Cost Error Protection for Last Level Cache.* Proceeding ISCA 09 Proceedings of the 36th annual International Symposium (Conference) on Computer Architecture. 2009; 116-127.
[10] Doe Hyun Yoon, Mattan Erez. *Virtualized and Flexible ECC for Main Memory.* Proceeding ASPLOS'10 Proceedings of the 15th Edition of ASPLOS on Architecture Support for Programming Languages and operating Systems. 2010; 97-408.
[11] K RajaSekhar, BKV Prasad, T Madnu, P Satish Kumar. Implementation of Fault Secure Encoder and Decoder for Memory Application. *Journal of Trends in Network and Communications.* 197(1): 33-43.
[12] CL Chen, MY Hsiao. Error-correcting codes for Seniconductor Memory Application:A State-of-the-art Review. IBM *Journal of Research and Development*. 1984; 28(2): 124–134.
[13] RW Hamming. Errror Correcting and Error Detecing Codes. *Bell System Technical Journal.* 1950; 29(1): 147–160.
[14] MY Hsiao. A Class of Optimal Minimum Odd-weight-column SEC-DED codes. IBM Journal of Reserach and Development. 1970; 14(3): 395–301.
[15] C Slayman. Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations. *IEEE Transactions on Device and Materials Reliability.* 2005; 5: 397–404.
[16] D Wang, B Ganesh, N Tuaycharoen, K Baynes, A Jaleel, B Jacob. DRAMsim: A memory-system simulator. *SIGARCH Computer Architecture News (CAN).* 2005; 33: 100–107.
[17] Xinquan Jiao, Peijiao Ma, Yuguang Zhang*. ECC algorithm and the realization of the VHDL for high-capacity solid-state storage.* The Electronics and Optoelectronics (IOEOE) 2011 International Conference. 2011; 239-242.
[18] Aymen F, Belgacem H, Chiraz K. *A new efficient self-checking Hsiao SEC-DED memory Error Correcting Code*. The Microelectronics(ICM). International Conference. 2011; 1-5.