International Journal of Control and Automation
Vol. 9, No. 4 (2016), pp. 11-18
http://dx.doi.org/10.14257/ijca.2016.9.4.02

Data Transmission Error Detect Scheme for High Speed Semiconductor Memory

Joong-Ho Lee

Yongin University, Computer Science Dept.
Joongho65@yongin.ac.kr

Abstract

This paper presents a code scheme for data bit error detection in the semiconductor memory devices. Conventional error detecting method by using the ATM-8 HEC code has a significant amounts of area-overhead (~700 XOR gates), and long processing time(XOR 6 stage). Therefore it leads to a considerable burden on the timing margin at the time of reading and writing of the low power memory devices for CRC(Cyclic Redundancy Check) calculations. The proposed error detecting scheme which is based on the parity check is improved area-overhead and decreased error detection delay time. The double bit error detection coverage has improved up to 77% compared with conventional method.

Keywords*: error detection scheme, ATM-8 HEC code, area-overhead, processing time, low power memory devices, CRC*

1. Introduction

As the computer system process speed becomes faster, valid data window (tDV:Data Valid Window) is a crucial part for reliable data transmission and it is the main reason that cause data transmission error between semiconductor memory device and computer system.[1] Especially, as the device uses lower electrical power, its importance correspondingly increases. As the semiconductor such as DDR4 SDRAM (Double Data Rate 4 Synchronous Dynamic Random Access Memory) and GDDR5(Graphic DDR5) required more high speed data rate and less power, CRC(Cyclic Redundancy Check) and DBI(Data Bus Inversion) functions have been implemented.[2,3,5,6] Usually, ATM-8 HEC code is adopted in the CRC, which is error detecting function when the data transmit between system and memory. This code scheme check the data bit error for the 8 DQ, 64 bit data during 4 clock cycle(8 UI : Unit Interval). The following Table 1 shows detailed bit mapping for a DDR4 x8 device that includes CRC bits.

Table 1. Data Bit Mapping of 8UI DQ Data and CRC for X8 DDR4 Devices

Pin	x8									
	UI0	UI1	UI2	UI3	UI4	UI5	UI6	UI7	UI8	UI9
DQ0	d0	d1	d2	d3	d4	d5	d6	d7	CRC0	1
DQ1	d8	d9	d10	d11	d12	d13	d14	d15	CRC1	1
DQ2	d16	d17	d18	d19	d20	d21	d22	d23	CRC2	1
DQ3	d24	d25	d26	d27	d28	d29	d30	d31	CRC3	1
DQ4	d32	d33	d34	d35	d36	d37	d38	d39	CRC4	1
DQ5	d40	d41	d42	d43	d44	d45	d46	d47	CRC5	1
DQ6	d48	d49	d50	d51	d52	d53	d54	d55	CRC6	1
DQ7	d56	d57	d58	d59	d60	d61	d62	d63	CRC7	1
DBI	dbi0	dbi1	dbi2	dbi3	dbi4	dbi5	dbi6	dbi7	1	1

In the Table 1, CRC0~CRC7 bits are added to detect the data bit corruption and ATM-8 HEC code generate the error correcting code from the polynomial

ISSN: 2005-4297 IJCA

x^8+x^2+x+1. The syndrome polynomial for error checks are consisted of S0 ~ S7. [9] Unfortunately, many conventional CRC polynomials provide eventually less error detection capability than they might. So, we need to focus on the implementation for proper CRC for the error detection system [7].

$$\begin{aligned}
CRC0 = {} & d69 \oplus d68 \oplus d67 \oplus d66 \oplus d64 \oplus d63 \oplus d60 \oplus d56 \oplus d54 \oplus d53 \oplus d52 \oplus d50 \oplus \\
& d49 \oplus d48 \oplus d45 \oplus d43 \oplus d40 \oplus d39 \oplus d35 \oplus d34 \oplus d31 \oplus d30 \oplus d28 \oplus d23 \oplus \\
& d21 \oplus d19 \oplus d18 \oplus d16 \oplus d14 \oplus d12 \oplus d8 \oplus d7 \oplus d6 \oplus d0 \\
S0 = {} & d69 \oplus d68 \oplus d67 \oplus d66 \oplus d64 \oplus d63 \oplus d60 \oplus d56 \oplus d54 \oplus d53 \oplus d52 \oplus d50 \oplus d49 \\
& \oplus d48 \oplus d45 \oplus d43 \oplus d40 \oplus d39 \oplus d35 \oplus d34 \oplus d31 \oplus d30 \oplus d28 \oplus d23 \oplus d21 \\
& \oplus d19 \oplus d18 \oplus d16 \oplus d14 \oplus d12 \oplus d8 \oplus d7 \oplus d6 \oplus d0 \oplus CRC0
\end{aligned} \quad (1)$$

The detection circuit produced from the polynomial is made of 6 layers of XOR gate logic, and in total it has the overhead of over 700 XOR gate. The CRC calculation gives a lot of burden to the internal timing margin when the XOR gate is consisted of 6 layers. When DDR4 SDRAM transfer the seamless data during Burst Length(BL) 8, it should not exceed the tCCD(CAS to CAS Delay time = 5nCK). For these reasons, the designer should make sure that the circuit will not exceed tCCD. However, DRAM allows the CRC calculation for the time amount of 4nCK-α . α is the flight time for CRC calculation, when it pass through GIO(Global IO) from DRAM core and it reach to DQ buffer. This time factor increase tCL(CAS Latency). Considering that a XOR gate delay time is 120ps in 3.2Gbps (tCL=1.25ns), which seems difficult to meet in slow PVT conditions in a conventional technology. Furthermore, the difficulty becomes worse in the low power memory device [2, 4].

In this paper, a new CRC methodology which is based on the DBI is proposed to reduce the CRC calculation delay time and area overhead for high-speed memory devices. The proposed method was derived out of the idea that DBI data can be utilized to detect data bit errors. Table 1 also shows the data bit mapping for the DBI as well. With the proposed method, this paper presents an efficient CRC design method for high-speed memory devices and quantitatively analyzes the improvements made and the errors detected.

2. Previous Work

2.1. ATM-8 HEC Code

There are two way CRC application, that is serial and parallel implementation. The serial implementation is shown in Figure 1 for the case where the polynomial divisor is $g(x)=x^8+x^2+x+1$. Figure 2 shows the case of parallel scheme for the same polynomial divisor [10].

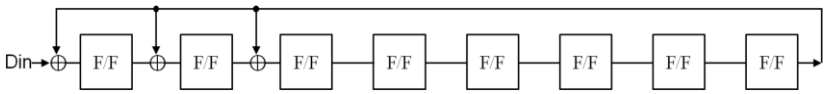

Figure 1. Serial Polynomial for $g(x)=x^8+x^2+x+1$

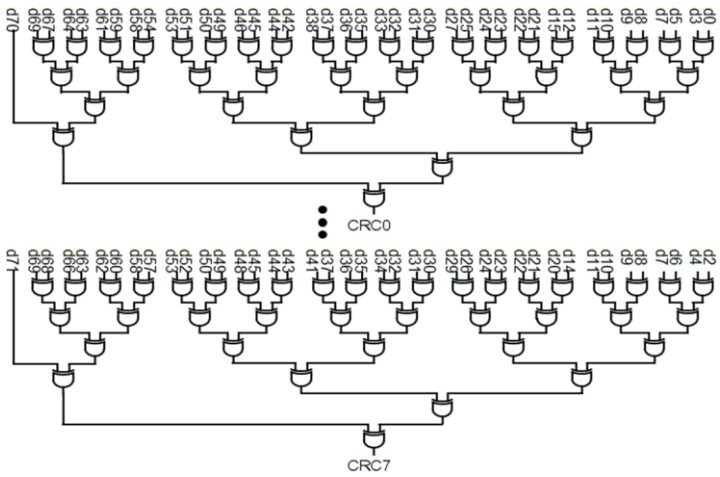

Figure 2. Parallel Implementation for $g(x)=x^8+x^2+x+1$

Parallel implementation is suitable for the high-speed computer systems even though parallel scheme is expensive solutions. But it is not enough for the system, which need more than 3.2Gbps clock speed like DDR4 SDRAM, GDDR5. For example, in the case of ATM-8 HEC code need 6 stages of 2-input XOR gates. To maintain seamless data transfer in DDR4 SDRAM, CRC operation time should not exceed CAS to CAS delay time(tCCD). So, the CRC calculation should be completed within a CRC time < tCCD = 5nCK. From the CRC time constraints, each XOR gate delay time to be allocated less than 120ps. In a convention DRAM process, this time constraints seems hard to meet in slow PVT conditions. To solve this problem, CAS Latency (CL) should be increased to guarantee proper timing slots. In the end, it results in a loss of effective bandwidth for a system [4,8].

2.2. CRC by Using the DBI (Data Bus Inversion)

In the high speed semiconductor memory devices, a shortage of internal timing increases for the CRC calculation. Because the conventional parallel CRC scheme using ATM-8 HEC code requires many additional circuit (more than 2-input XOR 700 gates) and long processing delay time (6 stage XOR gate). To improve the circuit overhead and gate delay time, CRC scheme by the DBI function mixed with parity checksum has been proposed. DBI function has been adopted to improve of data transmission speed for high speed semiconductor memory. But this CRC scheme has low error detection coverage for the double bit error compare to conventional CRC. Figure 3 shows the CRC decoder by using the DBI.

This scheme has established error check code based on the parity check bit, which was mixed with DBI data. Basically, an odd-parity check system can detect only the number of 1's is odd. Because odd-parity check system make the parity bit '1' when an odd number of '1' bits are in the data. Therefore, even number of '1' bits have to be check to compensate the missing error bits from odd-parity check system. DBI include information on the number of '1' of the total data bits. So, an even number of bit errors can be detected from the DBI characteristic. If DBI is enabled, then when the driver is sending out data on a lane, it counts the number of '1' bits. It can classify '1' bits of data as if the number is even or odd.

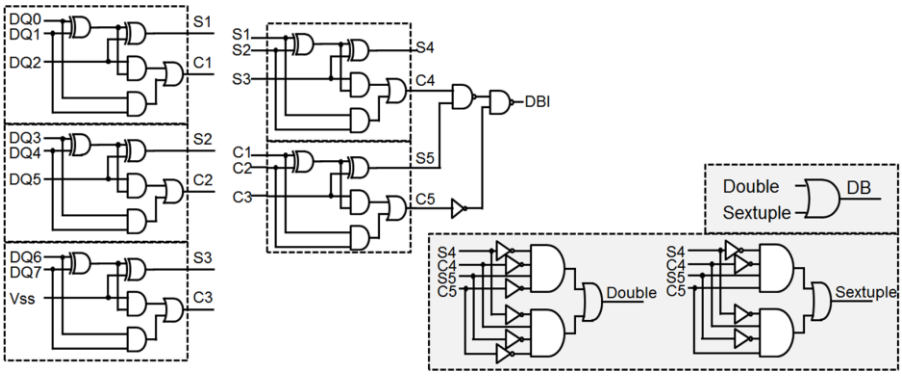

Figure 3. CRC Decoder by Using the DBI

3. CRC Scheme by Using the DBI

This paper presents a CRC scheme which is improve previous CRC by using the DBI scheme. DBI is consists of full adder (F/A) and from this we can extract the information of even number of data. Actually, DBI feature has been applied to reduce the SSO (Simultaneous Switching Output) noise and improve the signal integrity for the high speed data rate communication system. If more than four bits of a byte lane are Low then DBI invert output data. Hence, DBI enables fewer bits switching, which results in less noise and a better data eye. DBI feature was already shown in the Figure 3.

Furthermore, we can extract the number of double and sextuple of '1' bits from the DBI feature. And then, it can extract the position of double 1's and sextuple 1's. This information will apply to the proposed CRC to improve double bits error detection coverage. New CRC scheme is shown in the Figure 4.

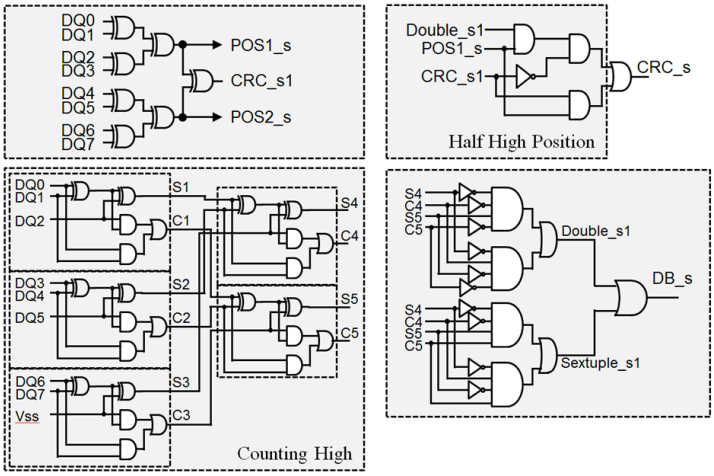

Figure 4. Proposed CRC Scheme

The proposed CRC's syndrome polynomial is shown below.

$$
\begin{aligned}
S0 &= d0 \oplus d8 \oplus d16 \oplus d24 \oplus d32 \oplus d40 \oplus d48 \oplus d56 \oplus CRC0 \\
S1 &= d1 \oplus d9 \oplus d17 \oplus d25 \oplus d33 \oplus d41 \oplus d49 \oplus d57 \oplus CRC1 \\
S2 &= d2 \oplus d10 \oplus d18 \oplus d26 \oplus d34 \oplus d42 \oplus d50 \oplus d58 \oplus CRC2 \\
S3 &= d3 \oplus d11 \oplus d19 \oplus d27 \oplus d35 \oplus d43 \oplus d51 \oplus d59 \oplus CRC3 \\
S4 &= d4 \oplus d12 \oplus d20 \oplus d28 \oplus d36 \oplus d44 \oplus d52 \oplus d60 \oplus CRC4 \\
S5 &= d5 \oplus d13 \oplus d21 \oplus d29 \oplus d37 \oplus d45 \oplus d53 \oplus d61 \oplus CRC5 \\
S6 &= d6 \oplus d14 \oplus d22 \oplus d30 \oplus d38 \oplus d46 \oplus d54 \oplus d62 \oplus CRC6 \\
S7 &= d7 \oplus d15 \oplus d23 \oplus d31 \oplus d39 \oplus d47 \oplus d55 \oplus d63 \oplus CRC7
\end{aligned} \tag{2}
$$

When data write to the memory, DB_s signal will send from system to memory and

then check the data error between DB_s and DB_m. Simultaneously, calculate the CRC syndrome polynomial S0~S7 in the memory. Finally, the odd data error can be detected by comparing the system data CRC_s and CRC_m. CRC_s is the system data, which is send to the memory and CRC_m is the generated data in the memory which is based on the DQ data in the memory. Also, the partial even data error can be detected by comparing the system data DB_s/DBI_s and DB_m/DBI_m. Figure 5 shows the proposed CRC configuration. The expression (3) shows the whole error detection components.

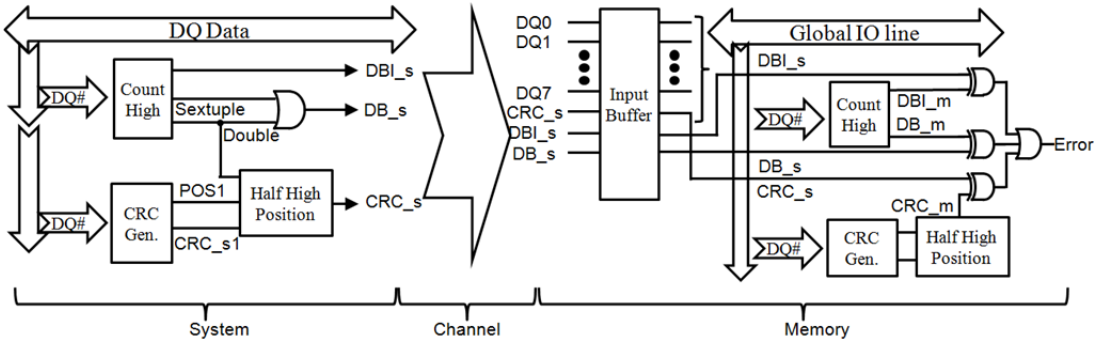

Figure 5. Proposed CRC System Configuration

$$E1 = DBI\_s \oplus DBI\_m,\ E2 = DB\_s \oplus DB\_m,\ E3 = CRC\_s \oplus CRC\_m$$

$$Error = E1 + E2 + E3 \quad (3)$$

Proposed CRC bit mapping configuration needs to change from the Table 1 for DDR4 data configuration. UI9 is fixed to High in the Table 1 which means that UI9 is wasted. So proposed scheme use the UI9 for mapping the DB_s data. Therefore UI9 of DQ0 is replaced to the DB_s of the UI0's all DQ data. In the same way, UI9 of DQ1 is replaced to the DB_s of the UI1's all DQ data. All the DB_s of UI9 is consists of the same way as before.

4. Error Detection and Coverage

Proposed CRC has odd parity check bit. Therefore all odd bit error can be detected like single, triple, quintuple and septuple bit error in the all DQ for each UI.

4.1. Double bit Error Detection

From the DB_s and DB_m of expression (3), by counting the number of '1' among the DQ data, double bit error can be detected. Furthermore, by extracting the position of the double bit errors, it can improve the error detection coverage. For example, when the data "1100 0000" was received from the original data "00000000", double error can be detected by comparing the DB_s and DB_m. Because DB include the information of the double data of "1". The other example, for the original data "11110000" when the double error occurred, the basic possible error patterns are "11000000", "11101000" and "11111100". The possible double error patterns for the "11000000" could be extended to the "01100000" and "0011 0000". Therefore these error patterns could be detected the same way as before. In the case of pattern "1110100", it could be detected double error by comparing the signal POS1_s and POS1_m. Table 2 shows the double error patterns for each DQ which is classified the original pattern of the number of "1". From this classified 9 patterns, we can extract all of the possible 28 error patterns for each classified patterns. Table 2 shows all of the extended double error patterns for the classified original pattern "11111110". For the pattern "11111110" of the #8, there are 28ea of the

extended error patterns. And 16ea error patterns among the whole 28ea errors could detected. All of the number of combinations for each UI is 252(=C(8,2)*9). Proposed CRC scheme has 77% double error detecting coverage.

Table 2. Error Patterns for the Classified Original Pattern of the Number of "1"

DQ	1	Error	2	Error	3	Error PT#	4	Error PT#	5	Error PT#	6	Error PT#	7	Error PT#	8	Error	9	Error
	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#
0	0	1	1	0 1	1	0 1 1	1	1 1 1	1	1 1 1	1	1 1 1	1	1 1 1	1	1 1 1	1	1
1	0	1	0	1 1	1	0 0 1	1	0 1 1	1	1 1 1	1	1 1 1	1	1 1 1	1	1 1 1	1	1
2	0	0	0	0 1	0	0 1 1	1	0 0 1	1	0 1 1	1	1 1 1	1	1 1 1	1	1 1 1	1	1
3	0	0	0	0 0	0	0 0 1	0	0 1 1	1	0 0 1	1	0 1 1	1	1 1 1	1	1 1 1	1	1
4	0	0	0	0 0	0	0 0 0	0	0 0 1	0	0 1 1	1	0 0 1	1	0 1 1	1	1 1 1	1	1
5	0	0	0	0 0	0	0 0 0	0	0 0 0	0	0 0 1	0	0 1 1	1	0 0 1	1	0 1 1	1	1
6	0	0	0	0 0	0	0 0 0	0	0 0 0	0	0 0 0	0	0 0 1	0	0 1 1	1	0 0 1	1	0
7	0	0	0	0 0	0	0 0 0	0	0 0 0	0	0 0 0	0	0 0 0	0	0 0 1	0	0 1 1	1	0
DBI	0	0	0	0 0	0	0 0 1	0	0 0 1	1	0 1 1	1	0 1 1	1	1 1 1	1	1 1 1	1	1
DB	0	1	0	0 0	1	0 1 0	0	0 0 0	0	1 0 1	0	0 0 0	1	0 1 1	0	0 0 0	0	1
crc	0	0	1	1 1	0	0 0 0	1	1 1 0	0	0 0 0	1	1 1 1	0	0 0 0	0	1 1 1	0	0
POS1	0	0	1	1 1	0	0 0 0	1	1 1 0	0	0 1 0	0	1 0 0	0	0 0 0	0	0 0 0	0	0
PT#		28		7 21		1 12 15		3 15 10		6 16 6		10 15 3		15 12 1		7 21		28

	Org.	All Error patterns for each UI from the original #8 pattern
0	1	0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
1	0	1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1
2	1	1 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 0 1 1 1
3	1	1 1 0 1 1 1 1 0 1 1 1 0 1 1 1 0 0 0 1 1 1 1 1 0 1 1
4	1	1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 1 0 1
5	1	0 1 1 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1 0 1 0 1 1 1 1 0
6	1	1 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 1
7	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1
POS1	0	0 0 0 1 1 1 0 0 1 1 1 0 1 1 1 1 1 1 0 0 0 1 1 1 0 0 0
Detect	-	X X X O O O X X O O O X O O O O O O X X X O O O X X X

4.2. Other Even Bit Error Detection

For the quadruple bit error patterns can be categorized the same way as before. There are 630ea possible combinations for the error pattern, and total error detection coverage has 73%. Also, the error detection coverage is 96.5% for the sextuple bit error patterns which has 252ea possible combinations for the error pattern. In the case of octuple bit error, its error detection coverage is 88.9%.

4.3. CRC Implementation

Proposed CRC scheme is consisted of even parity checks syndrome (S0~S7) which has 4 stage XOR gate. The even parity scheme can't detect double bit error. To resolve this, additional error detecting scheme need for double bit error. Therefore, this paper show how can detect the double bit error by extracting the number of "1" data from DBI, and improved error detection coverage. Figure 6 shows the proposed CRC circuits block.

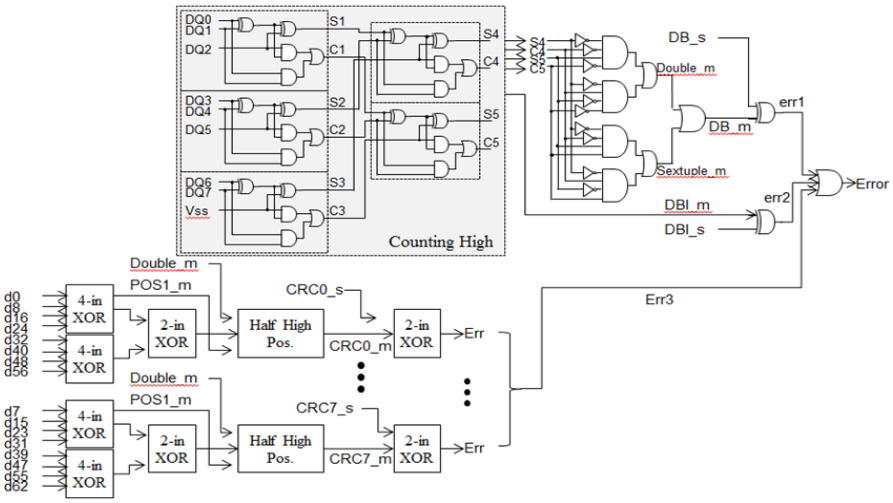

Figure 6. Proposed CRC Circuits

Table 3 shows the overhead comparison data between conventional CRC and proposed CRC method.

Table 3. Overhead Comparison Data

CRC		Area Overhead
scheme	XOR	
ATM-8 HEC	6	720 gates
Moon[4]	6	700 gates
Proposed CRC	4	56 gates

5. Conclusion

Proposed CRC scheme has implemented low cost and high speed CRC system. Conventional CRC function in the system has implemented by using ATM-8 HEC code which has 6 stage XOR gates. On the other hand, proposed CRC scheme has reduced to 4 stage XOR gates by using counting the number of "1" from the DBI and parity checks. The comparing data for the area overhead has shown in the Table 3. Finally, double bit error coverage improved to 77% from the previous CRC scheme, and error detection coverage is better than conventional CRC for the other bit errors. Table 4 shows the comparison data for the error detection coverage.

Table 4. Error Coverage Comparison Data

CRC Size (bit)	CRC	Error Coverage for Bit Errors							
		1bits	2bits	3bits	4bits	5bits	6bits	7bits	8bits
12	CRC-12	100%	100%	100%	99.96%	100%	99.99%	-	-
8	DARC-8	100%	97.9%	100%	99.87%	99.95%	99.95%	-	-
8	CRC-8	100%	100%	100%	99.81%	100%	99.91%	-	-
8	Proposed CRC	100%	77%	100%	73%	100%	95.6%	100%	88.9%
7	CRC-7	100%	100%	99.7%	99.83%	99.89%	99.92%	-	-

Acknowledgments

This work was supported by YongIn University.

References

[1] K. Koo, S. Ok, Y. Kang, S. Kim, C. Song, H. Lee, H. Kim, Y. Kim, J. Lee, S. Oak, Yoon-Jik M. Lee, J. Jang, J. C. Jung, B. Choi, Y. Hur and B.T. Chung, "A 1.2V 38nm 2.4Gb/s/pin 2Gb DDR4 SDRAM with Bank Group and x4 Half-Page Architecture", IEEE International Solid State Circuits Conference, San Francisco, California**,** USA, **(2012)**, pp. 40-41.

[2] S. Yoon, B. Kim, Y. Kim and B. Chung, "A Fast GDDR5 Read CRC Calculation Circuit with Read DBI Operation", IEEE Asian Solid-State Circuits Conference, Fukuoka, **(2008)**, pp. 249-252.

[3] S. J. Bae and K.I. Park, "An 80 nm 4 Gb/s/pin 32 bit 512 Mb GDDR4 Graphics DRAM With Low Power and Low Noise Data Bus Inversion", IEEE Journal of Solid-State Circuits, vol. 43, no. 1, **(2008)**, pp. 121-131.

[4] J. Moon, "Fast Parallel CRC & DBI Calculation for High-speed Memories:GDDR5 and DDR4", IEEE International symposium, Circuits and Systems (ISCAS), Rio de Janeiro, Brazil, **(2011)**, pp. 317-320.

[5] H. Sheidaeian and B. Zolfaghari, "Parallel Computation of CRC Using Special Generator Polynomials", International Journal of Computer Networks & Communications (IJCNC) ,vol. 4, no. 1, **(2012)**, pp. 39-47.

[6] K. Sohn, T. Na, I. Song, Y. Shim, W. Bae, S. Kang, D. Lee, H. Jung, S. Hyun, H. Jeoung, K. W. Lee, J. S. Park, J. Lee, B. Lee, I. Jun, J. Park, J. Park, H. Choi, S. Kim, H. Chung, Y. Choi, D. H. Jung, B. Kim, J. H. Choi, S. J. Jang, C. W. Kim, J. B. Lee and J. S. Choi, "A 1.2V 30nm 3.2Gb/s/pin 4Gb DDR4 SDRAM With Dual-Error Detection and PVT-Tolerant Data-Fetch Scheme", IEEE Journal of Solid-State Circuits(ISSCC), vol. 48, no. 1, **(2013)**, pp. 168-177.

[7] P. Koopman and T. Chakravarty "Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks", International Conference on Dependable Systems and Networks, Washington.

DC, USA. **(2004)** June-July, pp. 145-154.
[8] L. Joongho, "Matrix type CRC and XOR/XNOR for high-speed operation in DDR4 and GDDR5", Journal of IEEK (Institute of Electronics Engineers of Korea), vol. 50, no. 8, **(2013)**, pp. 136-142.
[9] JEDEC STANDARD, "DDR4 SDRAM", JEDEC Solid State Technology Association, JESD79-4, **(2012)** September.
[10] F. Monteiro, A. Dandache, A. M'sir, and B. Lepley, "A Fast CRC Implementation on FPGA Using a Pipelined Architecture for the Polynomial Division", The 8th IEEE International Conference on Electronics, Circuits and Systems, ICECS, St Julian, Malta, **(2001)** September, pp. 1231-1234.

Author

Joong-Ho Lee is currently a professor of Computer Science of the University of Yongin. He received his BS degree in Electronics & Computer Engineering from University of Ulsan in 1988 and received MS degree in Electronics & Computer Engineering from University of Ulsan in 1990. He received PhD degree in Electronics & Computer Engineering from University of Ulsan in 1994. He worked as research engineer in SK-Hynix from 1994 to 2012. He was involved in DDR1, DDR2, and DDR4 SDRAM design.

