ISSN: 1226-7244 (Print)
ISSN: 2288-243X (Online)
논문번호 15-03-02

j.inst.Korean.electr.electron.eng.Vol.19,No.3,288~294,September 2015
http://dx.doi.org/10.7471/ikeee.2015.19.3.288

고속반도체 메모리를 위한 DBI(Data Bus Inversion)를 이용한 저비용 CRC(Cyclic Redundancy Check)방식

Low-Cost CRC Scheme by Using DBI(Data Bus Inversion) for High Speed Semiconductor Memory

이 중 호*★

Joong-Ho Lee*★

Abstract

CRC function has been built into the high-speed semiconductor memory device in order to increase the reliability of data for high-speed operation. Also, DBI function is adopted to improve of data transmission speed. Conventional CRC(ATM-8 HEC code) method has a significant amounts of area-overhead(~XOR 700 gates), and processing time(6 stage XOR) is large. 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 calculations. In this paper, we propose a CRC method for low cost and high speed memory, which was improved 92% for area-overhead. For low-cost implementation of the CRC scheme by the DBI function it was supplemented by data bit error detection rate. And analyzing the error detection rate were compared with conventional CRC method.

요 약

고속동작을 위한 반도체 메모리 제품에서 데이터의 신뢰도를 개선하기 위해 CRC(Cyclic Redundancy Check) 기능이 내장되었으며, 데이터전송 속도 개선을 위해 DBI(Data Bus Inversion) 기능이 내장되었다. DDR4, GDDR4 등의 제품에 추가된 기존의 ATM-8 HEC 코드 방식은 부가회로 면적이 크고(~XOR 700 gates) CRC 처리 시간이 길어서(XOR 6단), 저전력 메모리 제품의 데이터 읽기, 쓰기시 내부 동작 마진(margin)에 적지 않은 부담을 초래한다. 본 논문에서는 저비용, 고속 반도체 메모리에 적합한 CRC방식을 제안하였으며 92%의 부가회로가 개선되었다. 제안한 CRC방식의 저비용 구현을 위해 DBI 기능을 이용하여 데이터 비트 오류 검출율을 보완하였으며, 오류 검출율을 분석하여 기존의 CRC방식과 비교하였다.

Keywords : DBI, CRC, High-speed memory, area overhead, error coverage

*★ Dept. of Computer Science, Yongin University
joongho65@yongin.ac.kr TEL: 031-8020-2768
Manuscript received June. 29, 2015; revised Aug. 4, 2015 ; accepted Aug. 5. 2015.

This work was supported by IDEC(EDA tool)

Ⅰ. 서론

반도체 메모리 제품은 컴퓨터 시스템의 저전력화, 고속화에 따라 지속적인 성능 개선이 요구된다. 컴퓨터 시스템의 고속, 저전력화를 위해 안전한 데이터 유효폭(tDV : Data Vaild window)의 확보는 필수이다.[1,2] 고속 반도체 메모리로 DDR4 SDRAM(Dual Data Rate 4 Synchronous Dynamic Random Access Memory), GDDR5(Graphic DDR 5) 등을 들 수 있으

며, 이러한 제품군의 고속화, 저전력화를 위해 CRC (Cyclic Redundancy Check)와 DBI(Data Bus Inversion) 기능이 채택되었다.[3~4, 6~7] CRC에 기존 ATM-8 HEC 코드 방식이 사용되는데, 다항식 x^8+x^2+x+1을 사용한다. 이 코드는 64비트 데이터(DQ0~DQ7에 대한 0UI~9UI 클럭 사이클의 데이타)에 대한 오류검출을 실시하기 때문에 전체 다항식에서 6단의 XOR 게이트를 사용하여 약 700 XOR 게이트 이상의 부가회로 면적을 가진다.[5] DDR4 SDRAM에서 Burst Length 8(BL8) 동안 연속적인 데이터 전송을 위해 tCCD(CAS to CAS delay time=5nCK)를 초과하지 않아야 한다. 이러한 이유로 인해 CRC를 위한 구현 시 설계자는 tCCD를 초과하지 않도록 설계해야하는데, DRAM 내부에서 CRC계산시 허용되는 시간은 4nCK-α 이며 그림 1에 나타내었다. α(=α1+α2)는 CRC 계산을 위해 DRAM core로 부터 GIO(Global IO)를 거쳐서 도달하는 시간과 CRC계산 이후 DQ 버퍼(buffer)로 도달하는데 소요되는 시간이다.[5] 이러한 시간요소는 tCL(CAS Latency)을 증가시키는 요인이며, 1600MHz의 3.2Gbps에서 1.25ns인데 CRC를 위해 6단의 XOR 게이트가 요구되는 상황에서 XOR 게이트 지연시간이 약 120ps로 설정하면 기존 tech.에서 PVT slow 조건에서는 취약한 마진을 가지며 저전력 반도체 메모리 제품에 더욱 열악하다.[1]

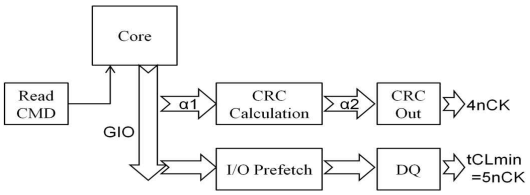

Fig. 1. CRC timing conceptual for DRAM
그림 1. DRAM에서 CRC 타이밍 개념도

전송라인 상에서 데이터 비트의 신뢰도 확보를 위한 CRC는 반도체 메모리제품 내부 타이밍 마진에 직접적인 영향을 미친다. CRC를 사용하는 많은 어플리케이션에서 주어진 CRC 비트에 비해 훨씬 적은 오류검출기능을 제공하는 CRC방식을 채택하고 있다. Koopman의 논문[8]에서 그 원인은 공식화된 지침이나 정량화된 데이터가 부재하기 때문으로 분석했다. 동일한 이유로 DDR4/GDDR5에 적용된 CRC는 오류검출 효율대비 비싼 비용의 코드방식을 채용하고 있다고 판단한다. 기존의 ATM-8 HEC 코드 방식을 개선하여 동일한 CRC비트 대비 개선된 방식을 제시하였다.[9]

본 논문에서는 기존의 CRC 구현 시 부가회로 면적을 줄일 수 있도록 고속메모리 제품에 적용되는 DBI (Data Bus Inversion)를 이용하여 CRC에 적용하였다. 제안한 방식은 데이터 비트의 오류 검출시 DBI 정보를 활용할 수 있다는데 기초하였다. DDR4 x8 제품에서 burst length 8 에 대한 DBI와 데이터 프레임 구성은 표 1과 같다. 표 1에서 매 클럭 당 2개의 Unit Interval(UI)로 구성되며, 매 UI당 각 DQ 핀에 대해 하나의 데이터가 할당된다. DBI동작은 CRC 계산에 선행해야 하며, 전체 데이터 프레임 중 CRC0~CRC7은 오류검출 다항식에 의해 64비트 데이터를 대상으로 CRC 비트를 구성한다. GDDR5의 경우 DBI와 CRC 구성이 달라서 데이터 프레임의 구성이 다르다.

Table 1. configuration of 8UI DQ data and CRC for DDR4
표 1. DDR4의 8UI DQ 데이터와 CRC의 구성

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

ATM-8 HEC 코드방식에서 CRC 회로는 9UI, 8 burst에 대해 64 비트의 데이터에 기초하여 구성된다. 아래에 CRC0의 경우 전체 데이터 64 비트에 대한 CRC0이 검사하는 43개 데이터 비트의 다항식 구성을 나타내었다. CRC1~CRC7도 64비트에 대한 검사하는 비트만 다르고 다항식에 기초하여 구성된다.[10]

$$CRC0 = d69 \oplus d68 \oplus d67 \oplus d66 \oplus d64 \oplus d63 \oplus d60 \oplus d56 \oplus d54 \oplus d53 \oplus d52 \oplus d50 \oplus d49 \oplus d48 \oplus d45 \oplus d43 \oplus d40 \oplus d39 \oplus d35 \oplus d34 \oplus d31 \oplus d30 \oplus d28 \oplus d23 \oplus d21 \oplus d19 \oplus d18 \oplus d16 \oplus d14 \oplus d12 \oplus d8 \oplus d7 \oplus d6 \oplus d0$$

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

본 연구에서는 고속 반도체 메모리에서 저렴한 CRC 설계 방식을 제시하여 부가회로의 개선정도와 오류검출 정도를 정량적으로 분석하였다.

II. DBI & CRC 구성

저렴한 비용의 CRC 구성을 위해 기존 고속 메모리에 적용되고 있는 DBI를 CRC 구성에 활용한다. DBI는 데이터의 개수에 대한 정보를 가지고 있으므로 이를 데이타 오류 검출에 활용할 수 있다. 그림 2에 전가산기(F/A)로 구성된 DBI 회로 구성을 나타내었다. DDR4에서는 termination 전압으로 VDD를 사용하기 때문에 low 데이터를 구동할 때 pull-down 트랜지스터로 관통하는 전류에 의해 많은 전력소모가 발생된다. DBI 기능은 데이타 출력단에서 발생하는 관통전류로 인한 SSO(Simultaneous Switching Output) 노이즈를 감소시켜 IO 전력감소와 signal integrity를 개선하는데 사용된다.[7] IO 전력소모 감소를 위해서는 DQ 의 high 데이터의 개수를 줄여 low에서 high로 천이를 줄임으로서 가능하다.

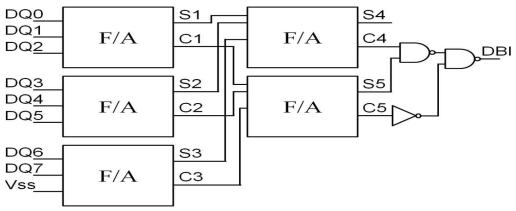

Fig. 2. DBI circuit
그림 2. DBI 회로

DQ 데이타에 대한 DBI 예를 표 2에 나타내었다. 시스템에서 원래 데이터에 대한 DBI를 생성해서 채널에 전송하고 메모리에서 받은 데이터는 DBI 값에 기초하여 원래 데이타를 저장한다. x8 IO구성에서 DQ high 데이터가 DBI에 의해 4개 이상 발생하지 않는다. 메모리에서 시스템에 데이터를 전송시에도 동일한 방법으로 DBI를 생성하여 전송하고 시스템에서 DBI 정보에 의해 원래 데이터를 복원한다.

Table 2. received data with DBI
표 2. DBI 동작과 수신된 데이터

	System							
	#1	#2	#3	#4	#5	#6	#7	#8
DQ0	0	0	0	0	0	0	0	1
DQ1	0	0	0	0	0	0	1	1
DQ2	0	0	0	0	0	1	1	1
DQ3	0	0	0	0	1	1	1	1
DQ4	0	0	0	1	1	1	1	1
DQ5	0	0	1	1	1	1	1	1
DQ6	0	1	1	1	1	1	1	1
DQ7	1	1	1	1	1	1	1	1
no. of "1"	1	2	3	4	5	6	7	8

	Channel							
	#1	#2	#3	#4	#5	#6	#7	#8
DQ0	0	0	0	1	1	1	1	0
DQ1	0	0	0	1	1	1	0	0
DQ2	0	0	0	1	1	0	0	0
DQ3	0	0	0	1	0	0	0	0
DQ4	0	0	0	0	0	0	0	0
DQ5	0	0	1	0	0	0	0	0
DQ6	0	1	1	0	0	0	0	0
DQ7	1	1	1	0	0	0	0	0
DBI	0	0	0	1	1	1	1	1
no. of "1"	1	2	3	5	4	3	2	1

	Memory							
	#1	#2	#3	#4	#5	#6	#7	#8
DQ0	0	0	0	0	0	0	0	1
DQ1	0	0	0	0	0	0	1	1
DQ2	0	0	0	0	0	1	1	1
DQ3	0	0	0	0	1	1	1	1
DQ4	0	0	0	1	1	1	1	1
DQ5	0	0	1	1	1	1	1	1
DQ6	0	1	1	1	1	1	1	1
DQ7	1	1	1	1	1	1	1	1

1. 오류 검출(error detect) 시스템 구성

기존의 ATM-8코드를 적용한 CRC를 개선하여 저비용의 오류검출 기능을 갖는 시스템을 제안하였다. 데이터 오류검출을 위한 기초적인 방법은 짝수 패리티 검사방법이 있으며 메모리 제품에 대한 오류 검출시 열(column) 방향이나 행(row)방향에 대한 패리티 검사를 실시함으로서 저비용 오류검출이 가능하다. 그러나 열이나 행 방향에 대한 패리티 검사는 이중고장 검출 및 짝수개의 비트 오류검출이 불가능하다. 표 3에 2차원 구조를 갖는 메모리 제품에 대한 패리티검사 방법에 의한 패리티비트 구성 예를 나타내었다.

Table 3. parity check for row and column data
표 3. 행과 열 데이타의 패리티 검사

	Column parity check							
	#1	#2	#3	#4	#5	#6	#7	#8
DQ0	0	0	0	0	0	0	0	1
DQ1	0	0	0	0	0	0	1	1
DQ2	0	0	0	0	0	1	1	1
DQ3	0	0	0	0	1	1	1	1
DQ4	0	0	0	1	1	1	1	1
DQ5	0	0	1	1	1	1	1	1
DQ6	0	1	1	1	1	1	1	1
DQ7	1	1	1	1	1	1	1	1
parity	1	0	1	0	1	0	1	0

	Row parity check								
	#1	#2	#3	#4	#5	#6	#7	#8	parity
DQ0	0	0	0	0	0	0	0	1	1
DQ1	0	0	0	0	0	0	1	1	0
DQ2	0	0	0	0	0	1	1	1	1
DQ3	0	0	0	0	1	1	1	1	0
DQ4	0	0	0	1	1	1	1	1	1
DQ5	0	0	1	1	1	1	1	1	0
DQ6	0	1	1	1	1	1	1	1	1
DQ7	1	1	1	1	1	1	1	1	0

DBI로 부터 데이터의 개수 정보를 추출할 수 있으므로 이를 표 2와 같은 패리티 검사 방법에 활용하여 CRC 기능을 구현할 수 있다. 즉 짝수 패리티 검사 방식에서 이중고장이 발생할 경우 데이터 송신측에서 데이타 개수 정보를 같이 보내면 이중고장이 발생했을때 검출 가능하다. 오류 검출예는 뒤에서 설명하겠다. DBI에서 데이터의 개수 정보를 받기위해 전체 데이터 프레임을 표 1의 기존 프레임으로부터 새로이 구성한다. DDR4의 기존 데이터 프레임에서 UI9는 사용하지 않고 high fix인 상태를 유지한다. 본 제안에서는 기존에 사용하지 않는 UI9를 활용하여 DBI로부터 데이터의 double과 sextuple값을 입력 받는다. 표 4에 제안하는 데이터 프레임을 나타내었다. DB는 데이터의 "1"의 개수가 두 개(double)나 여섯 개(sextuple) 이면 "1"값을 가지도록 구성하였다.

Table 4. configuration data frame of the proposed method
표 4. 제안한 방식의 데이터 프레임 구성

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

DBI에서 데이터의 개수정보를 추출하는 회로를 그림 3에 나타내었다. 기존 DBI 회로에서 S4, C4, S5 및 C5로부터 1의 개수가 2개이면 double 신호를, 1의 개수가 6개이면 sextuple신호를 발생한다. double과 sextuple신호를 표 4의 DB정보로 사용한다.

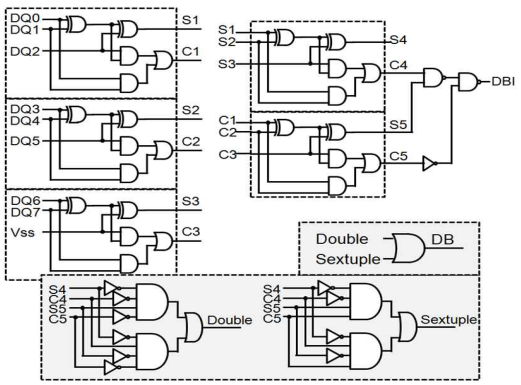

Fig. 3. DBI circuit configuration of the proposed CRC
그림 3. 제안한 CRC방식의 DBI 회로구성

본 논문에서 제안하는 CRC 방식의 시스템 구성을 위한 CRC syndrome 다항식을 식 (1)에 나타내었다.

$$
\begin{aligned}
S0 &= d0 \oplus d8 \oplus d16 \oplus d24 \oplus d32 \oplus d40 \oplus d48 \oplus d56 \oplus CRC0\\
S1 &= d1 \oplus d9 \oplus d17 \oplus d25 \oplus d33 \oplus d41 \oplus d49 \oplus d57 \oplus CRC1\\
S2 &= d2 \oplus d10 \oplus d18 \oplus d26 \oplus d34 \oplus d42 \oplus d50 \oplus d58 \oplus CRC2\\
S3 &= d3 \oplus d11 \oplus d19 \oplus d27 \oplus d35 \oplus d43 \oplus d51 \oplus d59 \oplus CRC3\\
S4 &= d4 \oplus d12 \oplus d20 \oplus d28 \oplus d36 \oplus d44 \oplus d52 \oplus d60 \oplus CRC4\\
S5 &= d5 \oplus d13 \oplus d21 \oplus d29 \oplus d37 \oplus d45 \oplus d53 \oplus d60 \oplus CRC5\\
S6 &= d6 \oplus d14 \oplus d22 \oplus d30 \oplus d38 \oplus d46 \oplus d54 \oplus d61 \oplus CRC6\\
S7 &= d7 \oplus d15 \oplus d23 \oplus d31 \oplus d39 \oplus d47 \oplus d55 \oplus d62 \oplus CRC7
\end{aligned}
\tag{1}
$$

메모리에 데이터 쓰기동작 시 입력받은 DBI, DB정보에 대해 시스템에서 보낸 정보(DBI_sys, DB_sys)와 DRAM에서 발생한 DBI, DB정보를 각각 비교하여 데이터 오류를 검출한다. 이와 동시에 열 방향 CRC의 다항식을 계산하여 syndrome S0~S7로부터 데이터 오류 여부를 검출한다. 오류검출 식을 식(2)에 나타내었다.

$$
\begin{aligned}
E1 &= DBI\_sys \oplus DBI\\
E2 &= DB\_sys \oplus DB\\
syndrome &= S0 + S1 + S2 + S3 + S4 + S5 + S6 + S7\\
Error &= E1 + E2 + syndrome
\end{aligned}
\tag{2}
$$

쓰기동작 시에는 DBI 동작과 CRC 계산이 독립적으로 이루어 질 수 있으나, 읽기동작 시에는 DBI 동작이 CRC보다 먼저 완료되어야 한다. CRC 회로는 식(1)로부터 9UI, 8 burst에 대해 72비트의 데이터에 기초하여 구성되는데, 그림 4는 쓰기동작 시 DDR4에 적용되는 DBI와 CRC 블럭도를 나타내고 있으며, 그림 5는 읽기 동작시 DBI와 CRC 블럭도의 관계를 나타내었다.

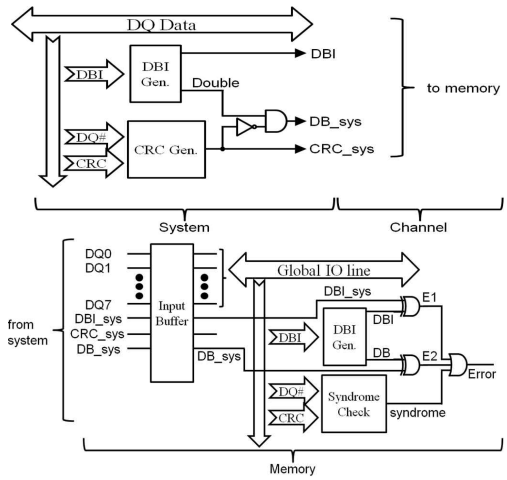

Fig. 4. CRC configuration during a write operation
그림 4. 쓰기 동작시 CRC의 구성

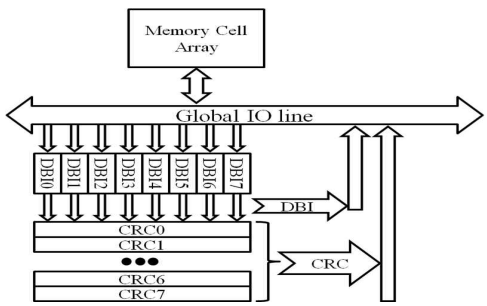

Fig. 5. CRC configuration during a read operation
그림 5. 읽기 동작시 CRC의 구성

2. 오류 검출(error detect) 방법 및 범위

2.1. 홀수 비트 오류(odd bit error)

본 논문에서 제안한 CRC 구성방식의 오류검출 범위에 대해 정량적으로 분석하여 아래에 나타내었다. 제안한 CRC는 데이터 프레임(표 1)의 열 방향에 대해 식 (1)로 부터 짝수 패리티(even parity)로 구성되기 때문에 각각의 UI에서 발생하는 DQ 데이터의 모든 홀수 비트 오류에 대해 검출 가능하다. 즉 각 UI에서 발생되는 단일 비트(single bit), 삼중 비트(triple bit), 5중 비트(quintuple bit), 7중 비트(septuple bit) 오류를 모두 검출 가능하다. 채널에서 발생된 이중 고장 이상의 짝수비트 고장은 식(1)과 같은 짝수 패리티 검사방식으로는 검출 불가능하다. 따라서 이중고장 검출을 위해 그림 3의 DBI 기능을 적용하여, 식(2)의 E2 항으로부터 DBI 에서 로직 “1” 데이터의 개수를 판독한 DB 값을 적용하였다. 표 5에 데이터의 패턴에 따른 이중고장 발생 패턴을 나타내었으며, 이때 CRC, DBI 및 DB값으로부터 오류 검출 가능여부를 나타내었다. 표 5의 데이터 패턴은 데이터에 포함된 ″1″의 개수에 따라 1~9까지 경우로 분류하였다. 표의 각 패턴으로부터 발생할 수 있는 경우의 수 패턴조합에 의해 확장된 패턴이 만들어 질 수 있으나, 확장된 패턴일지라도 표에 나타낸 오류검출 결과에 수렴한다. 확장된 패턴이라 함은 2의 데이터의 경우 원래 데이터(“10000000”)에서 “01000000”, “00100000”, “00010000”, “00001000”, “00000100”, “00000010”, “00000001”과 같은 7개의 패턴을 말한다. 표의 데이터 1번의 경우 시스템으로부터 수신된 원래 데이터(00000000)에서 이중고장이 발생하여 “1100 0000”이 수신되었을 경우, 이 때 수신된 DB값이 “0”이고, DRAM내부에서 발생한 DB값은 “1”이므로 오류를 검출할 수 있다. 또 다른 원래 데이터 5번(“11110000”)의 예에서 이중고장이 발생할 경우는 “11000000”, “11101000” 및 “11111100” 3개이며, 원래 데이터 패턴의 확장된 패턴에 따른 이중고장 발생 경우의 수는 동일하다. 이때 이중고장 “11000000”패턴의 경우, “1”이 2개이므로 DRAM내부에서 발생한 DB값은 “0”이고 원래 데이터의 DB값은 ″1″ 되어 오류를 검출할 수 있다. 그러나 “11101000”이 수신되었다면 DRAM내부에서 발생한 DB값과 원래 데이터의 DB값이 동일하여 오류검출이 불가능하다.

그림 6에 DQ0과 DQ1에 “low fix″의 이중고장이 발생할 경우에 대한 시뮬레이션 결과를 나타내었다. 예

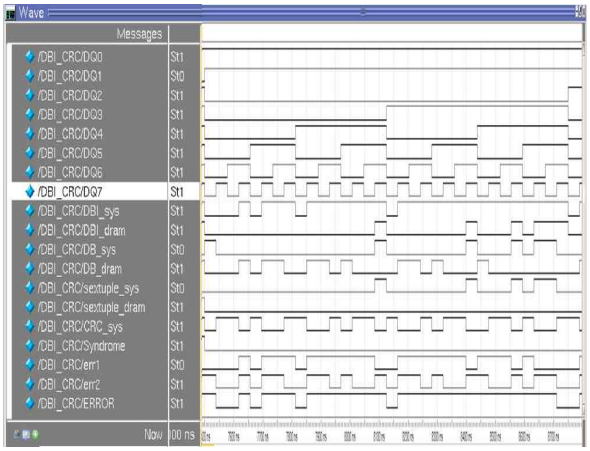

Fig. 6. simulation results of the double error
그림 6. 이중고장의 검출 시뮬레이션 결과

를 들어 원래 데이타 ”10000011“에 대해 ”10000000“ 데이타가 수신되어 오류 검출신호인 Error값이 ”1“이 되어 오류가 발생했음을 나타내고 있다. 표 5에서 각 패턴의 조합의 수는 총 252가지(패턴 1에서 C(8,2)=28)이며, 이때 하나의 UI에 대한 오류검출 가능한 경우는 119이며, 총 9UI에 대해서는 1071가지이다. 기존의 홀수 패리티 방식에서는 검출 불가한 2중 고장이지만, 제안한 방식에서는 홀수 패리티 방식을 사용하면서 DBI 기능을 추가 보완하여 47.6%까지 검출할 수 있도록 개선하였다. 표 5를 기초하여 확장 패턴을 포함한 경우도 동일한 검출율을 가진다.

Table 5. Table of double error coverage
표 5. 이중고장의 검출표

DQ	1	Error	2	Error	3	Error	4	Error	5	Error	6	Error	7	Error	8	Error	9	Error
UI#	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	0	1	0	1	1	1	0	0	1	1	0	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
3	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	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	1	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1	1	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0
DBI	0	0	0	0	0	0	0	1	0	0	0	1	1	0	1	1	1	0	1	1	1	1	1	1
DB	0	1	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	1
crc	0	0	1	1	1	0	1	0	0	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
검출	-	O	-	X	X	-	O	X	O	-	X	X	O	-	O	X	O	-	O	X	X	-	O	X
No.		28		7	21		1	12	15		3	15	10		6	16	6		10	15	3		15	12

2.3. 4중 고장(quadruple error)

4중 고장의 경우 이중고장에서와 같은 방법으로 데이터에 포함된 ″1″의 개수에 따라 1~9까지 경우로 분류하여 가능한 패턴에 대한 검출 여부를 표 6에 나타

내었다. 표 6의 1UI에 대한 전체 패턴 조합은 총 630가지이며, 오류검출 가능한 경우는 368(9UI에 대해 3312)가지로 58.4%의 검출율을 가진다.

Table 6. Table of quadruple error coverage
표 6. 4중 고장의 검출표

DQ	1 Error	2 Error	3 Error	4 Error	5 Error	6 Error	7 Error	8 Error	9 Error
UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#
0	0 1	1 0 1	1 0 1 1	1 0 1 1 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	1 1 1 1 1 1 1 1 1
1	0 1	0 1 1	1 0 0 1	1 0 0 1 1	1 0 0 1 1 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 1 1 1
2	0 1	0 1 1	0 1 1 1	1 0 0 0 1	1 1 0 0 1 1	1 1 0 1 1 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
3	0 1	0 1 1	1 0 1 1	0 1 1 1 1	1 0 0 0 0 1	1 0 0 0 0 1	1 0 0 1 1 1	1 0 1 1 0 1 1	1 1 1 1 1 1 1 1
4	0 0	0 0 1	0 1 1 0	0 1 1 1 1	0 1 1 1 1 1	1 0 0 0 0 1	1 0 0 0 1 1	1 0 0 1 1 0 1	1 0 1 1 0 1 0
5	0 0	0 0 0	0 0 0 1	0 0 0 1 1	0 0 1 1 1 0	0 1 1 1 1 1	1 0 0 0 1 1	1 0 0 1 0 1 0	1 0 1 1 0 1 0
6	0 0	0 0 0	0 0 0 0	0 0 0 0 1	0 0 0 1 1 0	0 0 0 1 1 1	0 1 1 1 1 0	1 0 1 1 0 1 0	1 0 1 1 0 1 0
7	0 0	0 0 0	0 0 0 0	0 0 0 0 0	0 0 0 0 0 1	0 0 0 0 1 1	0 0 1 1 1 1	1 0 1 1 0 1 1	1 0 1 1 0 1 1
DBI	0 1	0 0 1	0 1 0 1	0 0 1 1 1	0 0 1 1 1 1	1 0 0 1 1 1	1 1 0 1 1 1	1 0 1 1 1 0 1	1 1 1 1 1 1 1
DB	0 0	0 0 0	0 1 1 0	1 0 0 0 0	1 0 0 0 0 0	1 0 0 0 0 0	1 0 0 0 0 0	1 0 0 1 0 0 0	0 0 0 0 0 0 0
crc	0 0	1 1 1	0 0 0 0	1 1 1 1 1	0 0 0 0 0 0	1 1 1 1 1 0	0 0 0 0 1 1	1 1 1 1 0 0 0	1 1 1 1 0 0 0
검출	- O	- X O	- X X O	- X X O O	- O O X O X	- O O X X	- O O X	- O X	- X
No.	70	35 35	15 40 15	5 30 30 5	1 16 36 16 1	5 30 30 5	15 40 15	35 35	70

2.4. 6중 고장(sextuple error)
6중 고장의 경우 식 E1과 E2항으로 부터 표 7에 오류검출 여부를 나타내었으며 총 252가지의 패턴중에서 216가지를 검출가능하며, 총 검출율은 85.7%이다.

Table 7. Table of sextuple error coverage
표 7. 6중 고장의 검출표

DQ	1 Error	2 Error	3 Error	4 Error	5 Error	6 Error	7 Error	8 Error	9 Error
UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#
0	0 1	1 0 1	1 0 1 1	1 0 1 1 1	1 0 1 1 1 1	1 0 1 1 1 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	0 1	0 1 1	1 0 0 1	1 0 0 1 1	1 0 0 1 1 1	1 0 0 1 1 1	1 0 0 1 1 1 1	1 0 1 1 0 1 1	1 1 1 1 1 1 1 1 1
2	0 1	0 1 1	0 1 1 1	1 0 0 0 1	1 1 0 0 1 1	1 1 0 0 1 1	1 1 0 0 1 1 1	1 0 0 1 1 0 1	1 0 1 1 0 1 0
3	0 1	0 1 1	1 0 1 1	0 1 1 1 1	1 0 0 0 0 1	1 0 0 0 0 1	1 0 0 0 1 1	1 0 0 1 1 0 1	1 0 1 1 0 1 0
4	0 1	0 1 1	1 0 1 1	0 1 1 1 1	0 1 1 1 1 1	1 0 0 0 0 1	1 0 0 0 1 1	1 0 0 1 1 0 1	1 0 1 1 0 1 0
5	0 1	0 1 1	0 1 1 0	0 1 1 1 1	0 1 1 1 1 1	1 0 0 0 0 1	1 0 0 0 1 1	1 0 0 1 0 1 0	1 0 1 1 0 1 0
6	0 0	0 0 0	0 0 0 1	0 0 0 1 1	0 0 1 1 1 0	0 1 1 1 1 1	1 0 0 1 1 1	1 0 0 1 0 1 0	1 0 1 1 0 1 0
7	0 0	0 0 0	0 0 0 0	0 0 0 0 1	0 0 0 1 1 0	0 0 0 1 1 1	0 1 1 1 1 0	1 0 1 1 0 1 0	1 0 1 1 0 1 0
DBI	0 1	0 1 1	0 1 0 1	0 0 1 1 1	0 0 1 1 1 1	1 0 0 1 1 1	1 1 0 1 1 1	1 0 1 1 1 0 1	1 1 1 1 1 1 1
DB	0 1	0 0 0	1 0 0 0	0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0
crc	0 0	1 1 1	0 0 0 0	1 1 1 1 1	0 0 0 0 0 0	1 1 1 1 1 0	0 0 0 0 1 1	1 1 1 1 0 0 0	1 1 1 1 0 0 0
검출	- O	- O O	- O O O	- X O O -	O O X O -	O O X -	O O X -	O O O -	O O - O
No.	28	21 7	15 12 1	10 15 3	6 16 6	3 15 10	1 12 15	7 21	28

2.5. 8중 고장(octuple error)
8중 고장의 경우 식 E1과 E2항으로부터 88.9%의 검출율을 가진다.

Table 8. Table of octuple error coverage
표 8. 8중 고장의 검출

DQ	1 Error	2 Error	3 Error	4 Error	5 Error	6 Error	7 Error	8 Error	9 Error
UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#	UI#
0	0 1	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0
1	0 1	0 1	1 0	1 0	1 0	1 0	1 0	1 0	1 0
2	0 1	0 1	0 1	1 0	1 0	1 0	1 0	1 0	1 0
3	0 1	0 1	0 1	0 1	1 0	1 0	1 0	1 0	1 0
4	0 1	0 1	0 1	0 1	0 1	1 0	1 0	1 0	1 0
5	0 1	0 1	0 1	0 1	0 1	0 1	1 0	1 0	1 0
6	0 1	0 1	0 1	0 1	0 1	0 1	0 1	1 0	1 0
7	0 1	0 1	0 1	0 1	0 1	0 1	0 1	1 0	1 0
DBI	0 1	0 1	0 1	0 1	1 1	1 0	1 0	1 0	1 0
DB	0 1	0 0	1 1	0 0	0 0	0 0	0 0	0 0	0 0
crc	0 0	1 1	0 0	1 1	0 0	1 1	1 1	1 1	1 1
검출	- O	- O	- O	- O	- X	- O	- O	- O	- O

3. CRC 회로 구성

본 연구의 CRC방식을 그림 7에 나타내었다.

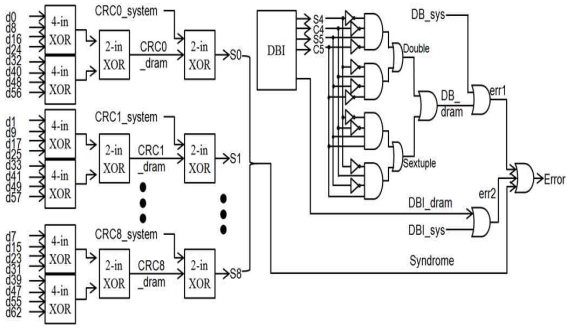

Fig. 7. proposed CRC circuit diagram
그림 7. 제안한 CRC회로도

제안한 CRC 방식은 4단으로 구성한 짝수패리티 검사가 가능하도록 syndrome(S0~S8)을 구성하였다. 짝수패리티가 가지는 문제를 보완하기위해 DBI 방식으로부터 데이터 "1"의 개수를 추출하여 짝수 고장시 검출율을 개선하도록 하였다. 이때 "1"의 개수중 double과 sextuple을 추출하여 데이터 비트로 사용하지 않는 UI9에 DB신호로 추가하였다. 기존의 방식은 x^8+x^2+x+1 다항식을 사용한 ATM-8 HEC 코드 방식으로, 표 1의 64개 데이터(d0~d63)를 다항식에 기초한 인코딩으로 부터 6단의 XOR조합으로 구성된다.[9] 표 9에 기존방식과 제안한 방식의 부가회로와 게이트단을 비교하였다.

Table 9. comparison of area-overhead and XOR stage
표 9. 부가회로와 XOR단 비교

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

III. 결 론

제안한 방식은 저비용 고속 CRC검사가 가능한 시스템 설계가 가능토록 하였다. 기존의 CRC를 위한 ATM-8 HEC 코드방식은 8단의 XOR게이트를 사용하여 CRC를 구성하지만, 제안한 방식은 4단으로 CRC를 구성한다. 오류 검출 능력을 보완하기 위해

별도의 부가회로를 추가하지 않고 DBI 기능을 적용하여 DB값을 UI9에 추가함으로서, 홀수 패리티 방식임에도 불구하고 2중 고장의 검출율을 47.6%까지 검출 가능하며, 다른 비트 오류에서는 우월한 성능을 가지도록 하였다. 표 10에 기존 방식과 CRC 오류 검출능력에 대해 비교하였다.

Table 10. error detection capability of proposed CRC scheme
표 10. 제안한 CRC scheme의 오류검출능력

CRC Size (bit)	CRC	Number of undetectable Bits Error							
		1bits	2bits	3bits	4bits	5bits	6bits	7bits	8bits
12	CRC-12	0	0	0	575	0	28809	-	-
8	DARC-8	0	66	0	2039	13122	124248	-	-
8	CRC-8	0	0	0	2984	0	253084	-	-
8	Proposed CRC	0	1197	0	2358	0	306	0	9
7	CRC-7	0	0	216	2690	27051	226856	-	-

기존 방식은 총 700게이트 이상의 부가회로가 소요된 반면 제안한 방식은 56게이트와 약간의 부가로직으로 구성할 수 있다. 따라서 제안하는 방식은 부가회로 면적을 92% 개선하여 저비용 CRC구현이 가능토록 하였다. 또한 게이트 지연시간을 33% 줄임으로서 고속 반도체 메모리제품에서 공정 PVT로 인한 tCK와 관련된 파라미터의 마진을 개선할 수 있도록 하였다.

References

[1] D. Graham-Smith, “IDF: DDR3 won’t catch up with DDR2 during 2009,” in PC Pro, Aug. 2008.
[2] Kibong Koo, et al., “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, pp. 40 - 41, Feb. 2012.
[3] S. Yoon, B. Kim, Y. Kim, B. Chung, “A Fast GDDR5 Read CRC Calculation Circuit with Read DBI Operation,” IEEE Asian Solid-Sate Circuits Conference, pp. 249-252, November, 2008.
[4] Seung-Jun Bae, Kwang-Il 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, pp. 121-131, January, 2008.
[5] J. Moon. “Fast Parallel CRC & DBI Calculation for High-speed Memories:GDDR5 and DDR4” Circuits and Systems (ISCAS), 2011 IEEE International symposium, pp. 317 - 320, May. 2011.
[6] J. Moon. “Fast Parallel CRC & DBI Calculation for High-speed Memories:GDDR5 and DDR4” Circuits and Systems (ISCAS), 2011 IEEE International symposium, pp. 317 - 320, May. 2011.
[7] Kyomin Sohn, et al., “A 1.2V 30nm 3.2Gb/s/pin 4Gb DDR4 SDRAM With Dual-Error Detection and PVT-Tolerant Data-Fetch Scheme” Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International Conference Vol.48, pp. 168-177, Jan. 2013.
[8] Koopman, P. and Chakravarty T. “Cyclic Redundancy Code(CRC) Polynomial Selection For Embedded Networks” 2004 International Conference on Dependable Systems and Networks, pp. 145-154 28 June-1 July 2004.
[9] Joongho, Lee, "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, pp. 136-142, Aug. 2013.
[10] JEDEC STANDARD, "DDR4 SDRAM", JEDEC Solid State Technology Association, JESD79-4, September 2012.

BIOGRAPHY

Joong-Ho Lee (Member)

1988 : BS degree in Electronics & Computer Engineering, Ulsan University.
1990 : MS degree in Electronics & Computer Engineering, Ulsan University.
1994 : PhD degree in Electronics & Computer Engineering, Ulsan University.
1994~2012 : SK-Hynix Semiconductor Inc. Senior research engineer.
2012~Present : Professor, Computer Science, Yongin University