비디오 처리를 위한 고성능 메모리 제어기의 FPGA 설계

*노혁래, *서영호, **최현준, *김동욱
*광운대학교, **안양대학교
*nhr@kw.ac.kr

FPGA Design of High-Performance Memory Controller for Video Processing

*Hyuk-Rae Noh, *Young-Ho Seo, **Hyun-Jun Choi, *Dong-Wook Kim
*Kwangwoon University, **Anyang University

요약

본 논문은 비디오 처리를 위한 고성능의 메모리 제어기를 설계하였다. 메모리 제어기는 arbiter에 의해 제어되며 이것은 메모리 억세스를 요구하는 모듈들의 요구 신호를 받아 데이터를 전송하는 역할을 해주게 된다. 구현된 메모리 제어기는 버스를 사용하기 위한 승인을 받기 위해서 마스터와 신호를 주고 받는 MAU블록, grant 신호를 디코딩하고 컨트롤 신호의 상태를 정의한 arbiter 블록, SDRAM의 ac parameter를 저장하고 bank의 준비 여부, read/write 가능 여부, precharge와 refresh의 가능 여부를 확인하여 system과 read/write가 준비되었다는 신호를 출력, SDRAM의 실질적인 입력신호를 생성하는 memory accelerator 블록, 생성된 입력신호를 저장하고 마스터에서 직접 write data를 입력 받는 memory I/F 블록으로 구성된다. 이 메모리 제어기는 174.28MHz의 주파수로 동작하였다. 본 설계는 VHDL을 이용하여 설계되었고, ALTERA의 Quartus Ⅱ를 이용하여 합성하였다. 또한 ModelSim을 이용하여 설계된 회로를 검증하였다. 구현된 하드웨어는 StatixⅢ EP3SE80F1152C2 칩을 사용하였다.

1. 서론

향후 PC 메모리 시장에서는 DRAM(Dynamic Random-Access Memory) 기술 부문의 경쟁이 더욱 치열해질 전망이다. 1990년 초에는 단지 몇 MHz에 달하던 마이크로프로세서 클럭 속도가 1990년대 말에는 수 백 MHz에 이르러 지난 10여 년 동안 폭발적으로 증가해 왔으며, 더 나아가 GHz급의 클럭 속도를 가지는 마이크로프로세서가 보편화 될 것으로 기대되고 있다. 이러한 경향은 PC 산업의 성장과 더불어 반도체 설계, 제조 부문의 발전에 힘입어 가능해진 것으로 분석된다.

비동기 DRAM에서는 메모리 액세스 동작시 메모리로부터 읽혀지거나 쓰여질 수 있는 데이터의 비트 양을 증가시키기 위해 FPM(Fast Page Mode), EDO(Extended Data Out)와 같은 기술들이 이용되었다.

SDRAM 은 DRAM 의 발전된 형태이며 보통 DRAM 과는 달리 제어 장치 입력을 클록펄스(Clock Pulse)와 동시에 일어나도록 하는 동기식 DRAM이다. SDRAM의 도입은 메모리 액세스를 시스템 버스 데이터 전송을 동기시킴으로써 시스템 성능에 향상을 가져왔다.

SOC(System On a Chip)와 같이 시스템 메모리로 단일 메모리(Unified Memory)를 사용하고 있는 일반적인 데이터 처리 시스템에 있어서, 메모리 중재기가 시스템 메모리를 사용하는 모든 메모리액세스유닛(Memory Access Unit)들의 메모리 버스 요청을 실시간으로 감시하여, 각 메모리액세스유닛의 진행중인 현재의 메모리 액세스 대기시간(Current Memory Access Latency)과 일정주기 동안의 최대 메모리 액세스 대기시간(Periodic Maximum Memory Access Latency) 및 최대 메모리 억세스 대기시간(Maximum Memory Access Latency) 등을 각각 측정하여 실시간으로 각 메모리액세스유닛에 대한 메모리 액세스 대기시간을 파악하고, 이를 미리 설정한 각 메모리액세스유닛마다 허용되는 메모리 액세스 레이턴시인 허용대기시간(Required Memory Access Latency)과 비교하여 실시간으로 각 메모리액세스유닛의 중재 우선순위를 조정하여 메모리 버스를 중재함으로써, 시스템의 성능과 안정성을 향상시킬 수 있다.
이 논문에서는 입력되는 영상을 저장하고 필요한 영상 정보를 읽어들이기 위해 Memory Controller를 통한 SDRAM을 사용한다.

따라서 본 연구는 최종 목표인 SoC 구현에 맞추어 Memory Controller를 최적화하여 구현하고자 한다.[1]

2. SDRAM 개요 및 동작

가. SDRAM 개요

SDRAM은 데이터를 저장할 수 있는 단위인 셀(cell)이 행렬 형태로 모여 있는 구조로 이루어져 있다. SDRAM의 기초동작은 정보의 저장 및 인출 동작이다. CELL에 정보를 저장 하는데 있어서 BITLINE에 'HIGH','LOW' 전압을 걸어서 1과 0을 기록한다. CELL에서 정보를 읽어내는데에는 BITLINE에 '1/2 HIGH' 전압을 걸어 셀 캐패시터 내의 전하의 유무에 따라 '1' 또는 '0'으로 해석한다. 또한 전하셀은 가만히 놔두어도 방전된다. 때문에 정보가 모두 소실되기 전에 REFRESH를 통해 주기적으로 DATA를 재충전 해 주어야 한다. 그리고 주소 비트의 일부는 행(column) 주소를 가리키고, 나머지 일부는 열(row) 주소를 지정하도록 할당되어 있다.

이런 SDRAM의 사용을 위해서는 필히 SDRAM 제어기(controller)가 있어야 한다. 기본적으로 메모리 시스템의 구조는 MCU 내부에 CPU core가 있고, Memory Controller를 통해서 외부의 Memory를 Access하고 Read/Write할 수 있다.

SDRAM 동작을 제어하는 입력 신호는 /RAS, /CAS, /WE 등이 있다. /RAS신호는 SDRAM 전체를 제어하는 Chip Enable과 같은 역할을 하며 /RAS신호가 입력된 후에야 DRAM이 동작을 시작한다. 이 신호가 Low로 변화할 때 Address에 있는 값이 Row Address임을 뜻한다. /CAS신호는 SDRAM에 Column Address를 인가했음을 알려주는 신호이다. /WE신호는 SDRAM에 Data를 써넣을 것인지 읽어낼 것인지를 결정하는 신호로 Low이면 쓰고 High이면 읽어낸다. 프로세서는 열 주소를 SDRAM의 주소 신호 선에 내보낼 때 /RAS(열 주소 선택) 신호를 활성화(assert)시킨다. 그리고 SDRAM 회로가 열 주소를 인식할 수 있도록 미리 프로그램 된 지연 시간이 흐르고 난 뒤에 프로세서는 행 주소를 내보내고 /CAS(행 주소 선택) 신호를 활성화시킨다. 활성화된 열을 비활성화 시키거나 모든 bank의 특정 열을 활성화시키는 precharge동작이 있다. SDRAM 컨트롤러는 실제 물리 메모리 주소를 열과 행 주소로 변환한다. 많은 SDRAM 컨트롤러들이 열과 행의 너비를 따로 설정할 수 있도록 되어 있다.

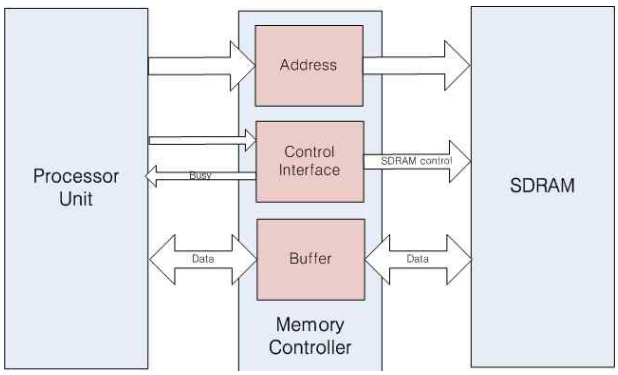

그림 1. 일반적인 Memory Controller의 구조

나. SDRAM 동작

메모리 제어기의 출력값들로 메모리의 동작이 결정된다. SDRAM이 동작을 하기 위해서는 우선 초기화 과정을 거쳐야한다.

초기화과정은 CKE를 Low로 유지하면서 SDRAM의 전원을 켜고 전원이 안정될 때까지 기다린 후 안정되면 CKE신호를 High로 만들고 CLK 신호를 시작한다. 파워와 클럭신호가 안정될 때까지 300us를 기다린다. 모든 뱅크에 대해 Precharge를 한다. Precharge는 정해진 명령어에 따라 /CAS가 High 상태이고 /RAS, /CS, /WE가 Low 상태일 때 동작한다. 규정된 /RAS Precharge Time의 지연 후에 8 cycle이나 그 이상을 auto refresh를 한다. auto refresh는 WE가 High 상태이고 /RAS, /CAS, /CS가 Low 상태일 때 동작한다. 모드 레지스터 셋 명령을 통해서 모드 레지스터를 초기화 시킨다.

이 초기화 과정이 끝나면 READ/WRITE를 할 수 있게 된다. READ/WRITE를 할 때 메모리는 연속동작(burst operation)기능을 갖고 있다. 이 기능으로 읽거나 쓸때의 연속적인 data의 길이를 burst length라 하고 이 burst length는 1,2,4,8,full page mode가 있는 것이 일반적이며, 이들은 열방향으로만 가능하도록 되어있다.

burst 모드를 쓰지 않은 단일 동작에서 우선 READ/WRITE를 하기 위해 해당 ROW에 ACTIVE를 시켜준다. 그 다음 READ/WRITE 명령을 통해 해당 주소의 값을 처리하게 된다. WRITE 동작이라면 바로 data가 쓰여지고, READ 동작의 경우는 딜레이가 되어 data가 읽히는 것을 확인되는데, 그 이유는 WRITE 동작의 경우 레이턴시가 항상 '0'이기 때문, 즉 데이터를 I/O버퍼까지 넣어주는 것이 우리의 책임이라서 SDRAM 내부에서 일어나는 동작까지 고려하지 않아도 되기 때문이다. 여기서 READ 동작의 경우 data가 바로 읽히지 않고 3클럭 뒤에 나오는 것을 그림2에서 확인 할 수 있다. 바로 이런 레이턴시가 있는 이유는 물리적으로 캐패시터로 이루어져 있으며 내부 버퍼가 존재하기 때문이다.

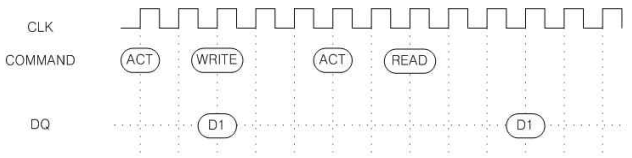

그림 2. 단일동작의 READ/WRITE Cycle

burst 동작의 경우 초기 어드레스를 주면 순차적으로 어드레스를 증가시키면서 READ/WRITE 동작을 한다. burst length에 따라서 한번에 릴리스 되는 DATA의 양이 조절된다. burst length 4인 경우 READ동작은 9클럭만에 4개의 DATA를 읽는 것을 볼수 있다. 단일동작으로 4개의 DATA를 READ동작을 시키는 경우를 계산해 보면 1개의 DATA를 6클럭만에 읽기 때문에 4개의 DATA는 그 4배인 24클럭이 된다. burst 동작이 단일 동작보다 약 2.7배 빠르다는 것을 알 수 있다. 그러나 burst 동작의 경우 하나의 ROW 내에서만 가능한 동작이다. Burst동작에서는 Burst length를 조절하여 1/2/4/8의 length를 쓸수도 있지만 한 ROW 전체에 대해서 동작하는 full page mode도 있다. full page mode 동작을 하게 되면 256개의 data가 한 클럭에 하나씩 릴리스 된다. 한 ROW만 단일동작과 burst 동작을 비교해 보아도 10배 정도 빠르다는 것을 확인 할 수 있다.

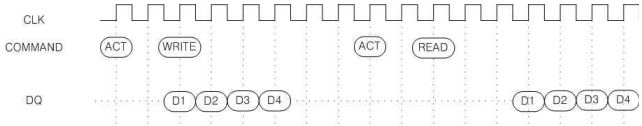

그림 3. Burst length = 4 인 Burst 동작의 READ/WRITE Cycle

3. 제안한 메모리 제어기

가. 메모리 제어기의 구조

영상처리분야에서의 제약은 동작을 진행하면서 데이터를 임시 저장하기 위한 메모리의 사용에 있다. 각 모듈들은 SDRAM과의 데이터 전송이 필요하고 이를 제어하는 것이 메모리 제어기의 주요 기능이다. 그림 1은 메모리 제어기의 전체 구조를 나타낸 것이다. 메모리 제어기는 각 모듈의 요구 신호에 의해 동작하며 각 모듈에서의 요구가 있을 때 구 우선순위에 따른 메모리 제어신호를 발생시킨다. 메모리 제어기는 다음과 같이 각 모듈간의 데이터 전송을 제어하고, SDRAM의 컨트

롤 신호를 디코딩한다[2].

본 논문에서의 메모리 제어기는 그림 5에서 나타낸 바와 같이 MAU 블록, arbiter 블록, memory accelerator블록, memory I/F 블록으로 구성하였다. 메모리 제어기는 arbiter에 의해 제어되며 이것은 메모리 억세스를 요구하는 모듈들의 요구 신호를 받아 데이터를 전송하는 역할을 해주게 된다.

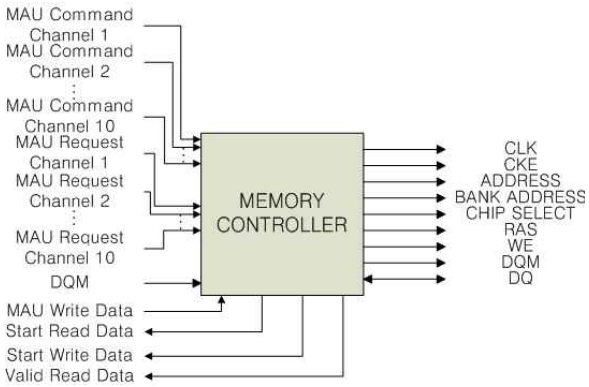

그림 4. 메모리 제어기의 전체 구조

MAU블록은 버스를 사용하기 위한 승인을 받기 위해서 프로세서 유닛과 신호를 주고 받는다. 그 절차는 request, command, grant 신호로 결정된다. MAU 블록은 입력신호로 request 신호와 command 신호를 받게 된다. request 신호는 주변장치가 메모리의 사용을 요청하면 이 신호의 상태가 변화한다. 이 신호의 변화에 MAU블록은 어떤 마스터가 메모리를 요청하는지 확인 후 grant 신호로 메모리의 사용여부를 출력한다. request 신호에 의해 메모리를 사용하는 마스터가 정해지면서 그에 맞는 command 신호를 grant 신호에 같이 출력한다. 여기서 command 신호에는 read/write, address, bank address 등의 신호로 구성되어 있다. 또한 channel에 대한 정보를 출력한다. Arbiter 블록은 Grant 신호를 디코딩하고 컨트롤 신호의 상태를 정의한 블록이다. Grant 신호를 디코딩해서 read/write 동작의 여부를 확인한다. Memory accelerator 블록에서 출력된 request 신호에 따라서 bank 와 row를 선택해 정의된 상태값 들을 출력한다. 컨트롤 신호가 idle상태일 경우 ready 신호를 Memory accelerator 블록과 마스터에 각각 출력한다. MAU블록으로부터 grant 신호와 채널에 대한 정보를 입력받은 뒤 read/write의 구분, bank addres, row address에 대한 정보와 마스터의 채널 정보를 Memory accelerator 블록으로 출력한다. Memory accelerator 블록은 실질적인 메모리 제어를 하는 블록이다. 이 블록은 크게 3가지의 세부 블록으로 나눌 수 있다. Arbiter 블록에서 system의 준비여부와 read/write에 대한 요청 신호를 입력 받는다. SDRAM의 ac parameter를 미리 저장해 두는 버퍼 블록. Grant 신호를 row address, column address, bank address 등으로 디코딩 하는 블록과 FSM으로 설계되어 직접적으로 SDRAM의 컨트롤 신호에 맞게 출력하고, 디코딩 블록으로 ready 신호를 출력하는 블록이 있다. Memory accelerator 블록은 ac parameter를 load하여 각 bank의 준비 여부, read/write 가능 여부, precharge와 refresh의 가능 여부를 확인하고, 디코딩 블록으로 보내 system과 read/write가 준비되었다는 신호를 출력한다. Memory I/F 블록은 SDRAM에 직접적인 입력신호를 주는 모듈이다. 마스터에서 직접적으로 MAU write data에 대한 입력을 받고 write data를 제외한 각 입력신호들은 memory accelerator 블록에서 디코딩 되어서 출력된 신호들이며, 이 입력된 신호들은 한 클럭 버퍼링되어서 SDRAM으로 출력된다. 또한 마스터의 channel 정보를 저장해 두었다가 MAU 블록으로 보내 read data에 대한 channel 정보를 알려주고 read data는 마스터로 직접 출력한다.

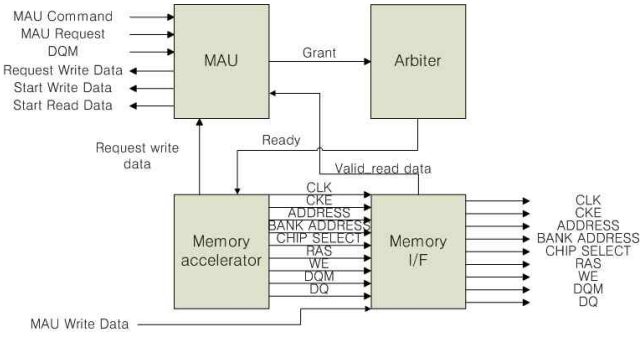

그림 5. 메모리 제어기의 세부 구조

나. 메모리 제어기의 동작

MAU 블록이 read data를 마스터로 출력하는 것은 memory I/F 블록에서 출력되는 start read data 신호에 의해서 결정된다. start read data 신호를 입력 받으면 MAU 블록은 start read data 신호와 valid read data 신호를 마스터로 출력한다. write data의 경우 memory accelerator 블록에서 디코딩된 start write data 신호가 입력되면 request 신호를 통해 선택된 마스터로 start write data 신호와 request write data 신호를 보내게 된다. MAU read/write data의 경우 grant 신호에 의해서 단일 동작과 시퀀스 동작이 변화한다.

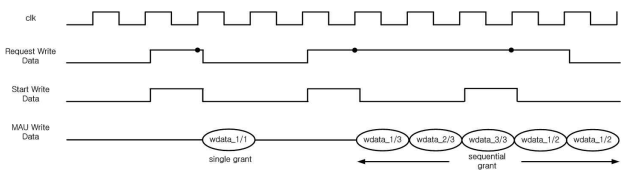

그림 6. write 동작

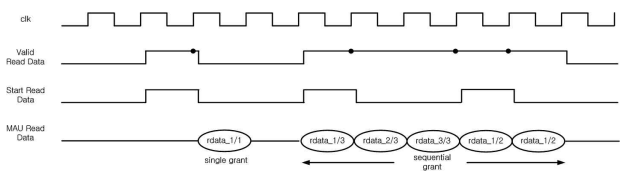

그림 7. read 동작

4. 구현결과

가. 하드웨어 구현 결과

제안한 메모리 제어기는 VHDL을 이용하여 설계하였으며, Altera의 Quartus II 환경을 사용하였다. VHDL 코딩에 의한 함수적 검증을 거쳐 합성단계를 수행한 후 설계사양에 맞는 지연시간을 각 동작에 따라 검증하였다. 메모리 제어기의 하드웨어 자원 사용율은 표 1에 나타내었고, 메모리 제어기의 FPGA 합성도는 그림 8에 나타내었다.

SDRAM의 동작 주파수가 FPGA 환경에서 174.28MHz인 것을 확인하였다. 타겟 플랫폼은 Altera의 StatixⅢ EP3SE80F1152C2칩이었다.

표 1. FPGA 자원 사용율

Logic utilization	Used	Available	Utilization
Combinational ALUTs	943	64000	1%
Dedicated logic registers	566	64000	1%
Total registers	566		
Total pins	545	744	73%

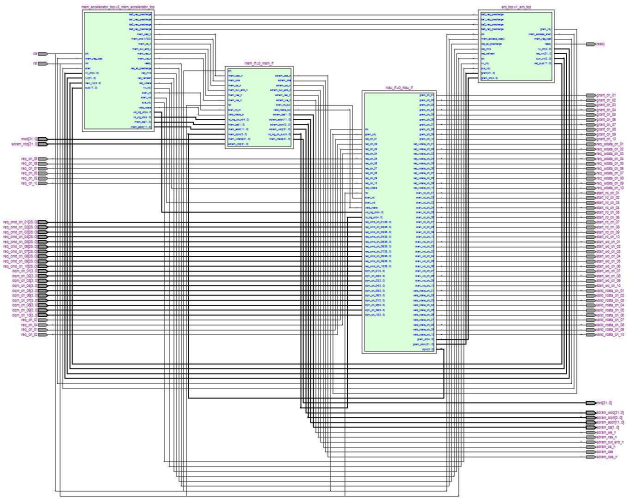

그림 8. 메모리 제어기의 FPGA 합성도

나. 시뮬레이션 결과

시뮬레이션은 Cadence사의 ModelSim으로 수행하였다. 그림 9은 burst를 사용하지 않는 burst length가 1(default)일 경우에 write와 read를 한 결과이다.

recharge와 refresh, register mode selection, active동작인 초기화 과정을 거친 뒤 write 동작 신호를 주었을 때 레이턴시 없이 바로 write 동작을 하는 것을 확인 할 수 있었고 레이턴시가 3인 경우에서 read 동작의 경우는 3클럭 뒤에 나오는 것을 확인할 수가 있었다.

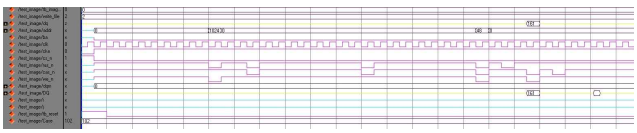

그림 9. WRITE & READ(default)

그림 10는 burst length가 4일 때 burst동작의 결과이다. write 동작의 경우 4 clock 동안 4개의 data가 입력되는 것을 볼 수 있다. 다음 read 동작 신호를 주었을 때 마찬가지로 3클럭의 레이턴시 후에 4개의 data가 4 clock 동안 읽히는 것을 볼 수 있다. 단일동작과 비교 했을 때 두 번째 data부터는 latency없이 초기 신호를 주면 address를 증가시키면서 read/write 동작을 하는 것을 볼 수 있다.

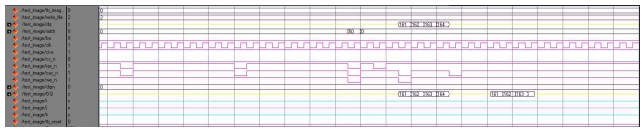

그림 10. WRITE & READ(Burst Length = 4)

그림 11과 12은 full page mode를 선택했을 때 영상의 한 line이 write, read 되는 결과이다. 여기서 full page mode의 한 row는 256개의 데이터까지 한번에 read/write 동작을 하는 것을 볼 수 있다.

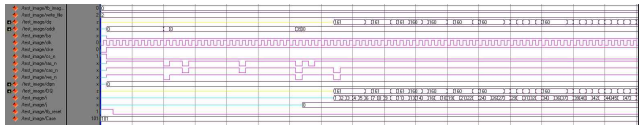

그림 11. WRITE(full page mode)

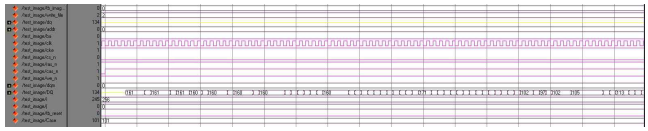

그림 12. READ(full page mode)

5. 결론

본 논문에서는 비디오 처리를 위하여 SDRAM을 제어할 수 있는 범용적인 메모리 제어기를 설계하였다. 구현한 메모리 제어기는 SDRAM을 동작시키기 위한 모든 모드를 지원하고 있고, 다수 개의 마스터를 수용할 수 있는 입력 구조를 가지고 있다. 또한 SDRAM이 수용할 수 있는 최대 주파수에서 동작이 가능하였다.

감사의 글

본 연구는 한국산업기술평가관리원(KEIT)의 IT산업원천기술개발사업의 일환으로 수행하였음. [KI002058, 대화형 디지털 홀로그램 통합서비스 시스템의 구현을 위한 신호처리 요소 기술 및 SoC 개발]

참 고 문 헌

[1] J. Kim et al., “A 512 Mb Two-Channel Mobile DRAM (OneDRAM) with Shared Memory Array,” IEEE J. Solid-State Circuits, vol. 43, no.11, Nov. 2008, pp. 2381-2389.

[2] 유회준, "DRAM DESIGN," 홍릉과학출판사, 1996.

[3] K. C. Chang, “Digital systems design with VHDL and synthesis”, IEEE Computer Society Press, California, 1999.