다중 마스터를 위한 고성능의 범용 메모리 제어기의 구조

최현준* · 서영호**· 김동욱***

VLSI Architecture of General-purpose Memory Controller with High-Performance for Multiple Master

Hyun-Jun Choi* · Young-Ho Seo** · Dong-Wook Kim***

이 논문은 2010년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2007-331-D00405).

요 약

본 논문은 비디오 처리를 위한 SoC 내에서 다수 개의 프로세싱 블록(마스터)들을 처리할 수 있는 고성능의 메모리 제어기를 설계하였다. 메모리 제어기는 마스터 중재기에 의해 중재되며 이것은 메모리 접근을 요구하는 마스터들의 요구 신호를 받아 데이터를 전송하는 역할을 해주게 된다. 구현된 메모리 제어기는 마스터 선택기, 마스터 중재기, 메모리 신호 생성기, 명령어 디코더, 데이터 버스, 그리고 메모리 신호 생성기로 구성된다. 제안한 메모리 제어기는 VHDL을 이용하여 설계하였고, 삼성의 메모리 모델을 이용하여 동작을 검증하였다. FPGA 합성 및 검증을 위해서는 ATERA사의 Quartus Ⅱ를 이용하였고, 구현된 하드웨어는 Cyclone II 칩을 사용하였다. 시뮬레이션을 위해서는 Cadence사의 ModelSim을 이용하였고, FPGA 환경에서 174.28MHz의 주파수로 동작하여, SDRAM의 규격을 모두 만족할 수 있었다.

ABSTRACT

In this paper, we implemented a high-performence memory controller which can accommodate processing blocks(multiple masters) in SoC for video signal processing. The memory controller is arbitrated by the internal arbiter which receives request signals from masters and sends grant and data signals to masters. The designed memory controller consists of Master Selector, Mster Arbiter, Memory Signal Generator, Command Decoder, and memory Signal Generator. It was designed using VHDL, and verified using the memory model of SAMSING Inc. For FPGA synthesis and verification, Quartus II of ATERA Inc. was used. The target device is Cyclone II. For simulation, ModelSim of Cadence Inc was used. Since the designed H/W can be stably operated in 174.28MHz, it satisfies the specification of SDRAM technology.

키워드

메모리 제어기, 다중마스터, SDRAM, 설계, FPGA

Key word

memory controller, multiple master, SDRAM, design, FPGA

* 정회원 : 안양대학교
** 종신회원 : 광운대학교 (교신저자, yhseo@kw.ac.kr)
*** 정회원 : 광운대학교

접수일자 : 2010. 08. 14
심사완료일자 : 2010. 09. 16

Ⅰ. 서 론

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[1][2][3].

비동기 DRAM에서는 메모리 액세스 동작 시 메모리로부터 읽혀지거나 쓰여질 수 있는 데이터의 비트 양을 증가시키기 위해 FPM(Fast Page Mode), EDO(Extended Data Out)와 같은 기술들이 이용되었다[4].

SDRAM[5]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[6][7].

여기서 SOC(System On a Chip)와 같이 시스템 메모리로 단일 메모리를 사용하고 있는 일반적인 데이터 처리 시스템에 있어서, 메모리 중재기가 시스템 메모리를 사용하는 모든 메모리 액세스 접근(Memory Access Unit)들의 메모리 버스 요청을 실시간로 감시하여, 각 메모리 접근 유닛의 진행 중 인 현재의 메모리 접근 대기시간(Current Memory Access Latency)과 일정 주기 동안의 최대 메모리 접근 대기시간(Periodic Maximum Memory Access Latency) 및 최대 메모리 접근 대기시간(Maximum Memory Access Latency) 등을 각각 측정하여 실시간으로 각 메모리 접근 유닛에 대한 메모리 접근 대기시간을 파악하고, 이를 미리 설정한 각 메모리 접근 유닛마다 허용되는 메모리 접근 대기지연시간인 허용대기시간(Required Memory Access Latency)과 비교하여 실시간으로 각 메모리액세스유닛의 중재 우선순위를 조정하여 메모리 버스를 중재함으로써, 시스템의 성능과 안정성을 향상시킬 수 있다[8].

본 논문은 다음과 같이 구성된다. 2장에서는 SDRAM에 대한 이론에 대해서 간략히 설명하고, 3장에서는 제안한 메모리의 구조와 동작에 대해서 설명한다. 4장에서는 구현 결과 및 시뮬레이션 결과를 보이고, 5장에서 결론을 맺는다.

Ⅱ. SDRAM 개요 및 동작

2.1 SDRAM 개요

본 절에서는 SDRAM에 대해 간략히 소개하고자 한다. 본 절의 내용은 다른 SDRAM 문서들과의 비교 및 참조를 용이하게 하기 위해서 가능한한 원래의 영문 용어를 그대로 사용하고자 한다.

SDRAM은 데이터를 저장할 수 있는 단위인 셀(cell)이 행렬 형태로 모여 있는 구조로 이루어져 있다. SDRAM의 기초 동작은 정보의 저장 및 인출 동작이다. 셀에 정보를 저장 하는데 있어서 BITLINE에 'HIGH' 혹은 'LOW' 전압을 인가하여 1과 0을 기록한다. 셀에서 정보를 읽어내는 데에는 BITLINE에 '1/2 HIGH' 전압을 인가하여 셀 캐패시터 내의 전하의 유무에 따라 '1' 또는 '0'으로 해석한다. 또한 전하 셀은 가만히 놔두어도 방전된다. 그렇기 때문에 정보가 모두 소실되기 전에 REFRESH를 통해 주기적으로 데이터를 재충전 해 주어야 한다. 그리고 주소 비트의 일부는 행(column) 주소를 가리키고, 나머지 일부는 열(row) 주소를 지정하도록 할당되어 있다[4].

이런 SDRAM의 사용을 위해서는 필히 SDRAM 제어기가 있어야 한다. 기본적으로 메모리 시스템의 구조는 MCU 내부에 CPU 코어가 있고, 메모리 제어기를 통해서 외부의 메모리를 접근하고 읽고/쓰기할 수 있다[4].

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 주소에 있는 값이 열 주소임을 뜻한다. /CAS신호는 SDRAM에 행 주소를 인가했음을 알려주는 신호이다. /WE신호는 SDRAM에 데이터를 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 뱅크(bank)의 특정 열을 활성화시키는 PRECHARGE동작이 있다. SDRAM 제어기는 실제 물리 메모리 주소를 열과 행 주소로 변환한다. 많은 SDRAM 컨트롤러들이 열과 행의 너비를 따로 설정할 수 있도록 되어 있다.

2.2 SDRAM의 동작

메모리 제어기의 출력값들로 메모리의 동작이 결정된다. 메모리는 연속동작(BURST operation)기능을 갖고 있다. 이 기능으로 읽거나 쓸 때의 연속적인 데이터의 길이를 BURST 길이라 한다. 이 BURST 길이는 1,2,4,8, FULL PAGE 모드가 있는 것이 일반적이며, 이들은 열방향으로만 가능하도록 되어있다. BURST 모드를 쓰지 않은 단일 동작과 BURST 동작 그리고 FULL PAGE 모드를 이용해서 영상을 읽고/쓰기를 수행하는 동작은 다음과 같다

2.2.1 초기화

우선 SDRAM이 동작하기 위해서는 초기화과정을 거쳐야한다. 우선 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 읽고/쓰기를 할 수 있게 된다. 읽고/쓰기를 하기 위해 해당 열에 ACTIVE를 시켜준다. 그 다음 읽고/쓰기 명령을 통해 해당 주소의 값을 처리한다. 초기화 과정에서 32클럭이 소요된다.

2.2.2 단일동작

단일 동작에서 ACTIVE 과정 뒤에 쓰기 동작이라면 2 클럭 이후부터 읽기 명령을 내리게 되면 5클럭 이후부터 데이터가 릴리스 된다. 그 이유는 쓰기동작의 경우 레이턴시가 항상 '0'이기 때문인데, 즉 데이터를 I/O버퍼까지 넣어주는 것이 우리의 책임이라서 SDRAM 내부에서 일어나는 동작까지 고려하지 않아도 되기 때문이다. 다음 읽고/쓰기 명령을 이행할 때까지 같은 클럭 사이클이 걸린다.

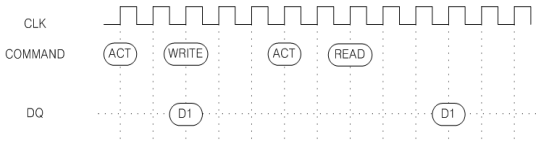

그림 1. 읽고/쓰기 모드(단일동작)
Fig. 1. Read/write mode(single operation)

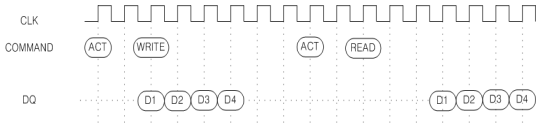

그림 2. 읽고/쓰기 모드 (BURST 4)
Fig. 2. Read/write mode(BURST 4)

2.2.2 Burst 동작

BURST 동작의 경우 초기 어드레스를 주면 순차적으로 어드레스를 증가시키면서 읽고/쓰기 동작을 한다. BURST 길이에 따라서 한 번에 릴리스 되는 데이터의 양이 조절된다. BURST 길이 4인 경우 읽기동작은 9 클럭만에 4개의 데이터를 읽는 것을 볼수 있다. 단일 동작으로 4개의 데이터를 읽기동작을 시키는 경우를 계산해 보면 1개의 데이터를 6클럭만에 읽기 때문에 4개의 데이터는 그 4배인 24클럭이 된다. BURST 동작이 단일 동작보다 약 2.7배 빠르다는 것을 알 수 있다. 그러나 BURST 동작의 경우 하나의 열 내에서만 가능한 동작이다.

2.2.3 FULL PAGE 모드

BURST동작에서는 BURST 길이를 조절하여 1/2/4/8의 길이를 쓸 수도 있지만 한 열 전체에 대해서 동작하는 FULL PAGE 모드도 있다. FULL PAGE 모드 동작을 하게 되면 256개의 데이터가 한 클럭에 하나씩 릴리스 된다. 한 열만 단일동작과 BURST 동작을 비교해 보아도 10배 정도 빠르다는 것을 확인 할 수 있다.

Ⅲ. 제안한 메모리 제어기의 구조 및 동작

3.1 메모리 제어기의 구조

실제로 멀티미디어용 SoC 내에서는 하나의 메모리가 다양한 동작 유닛들의 데이터 처리를 담당해야 한다. 다수 개의 메모리 제어기와 다수 개의 외부 SDRAM을 장착할 수도 있겠지만, SDRAM의 가격보다 멀티미디어용 SoC의 가격이 훨씬 저가라는 점을 고려한다면 현실적으로 불가능하다. 따라서 그림 3과 같이 하나의 메모리 제어기가 다수 개의 마스터들을 처리해야만 한다. 각 마스터들의 우선순위는 SoC 설계자에 의해서 이미 알려져 있을 것으로 그에 따라서 메모리 제어기와의 허가 및 요청 신호의 연결 기준이 정해진다.

그림 4는 메모리 제어기의 최상위 블록을 나타낸 것이다. 메모리 제어기는 각 모듈의 요구 신호에 의해 동작하며 각 모듈에서의 요구가 있을때 구 우선순위에 따른 메모리 제어신호를 발생시킨다. SDRAM 메모리 제어기는 다음과 같이 각 모듈간의 데이터 전송을 제어한다. 그림 4에서 OPCODE는 각 마스터별로 출력되고, SDRAM에 데이터를 접근하기 위한 정보(request 신호, 각종 주소 정보, 읽고/쓰기 모드)를 갖고 있다. GRANT 신호는 SDRAM에 접근이 허가된 마스터에게 주어지는 허가 신호된다. 마스터에게 쓰기 허가가 주어지면 WRREQ 신호가 동시에 출력되면서 쓰기를 요구하고 마스터는 WRDATA를 출력하여 SRAM에 데이터를 보낸다. 마스터에서 읽기 허가가 주어지면 읽기 데이터 유효 신호인 RDVAL와 함께 RDDATA가 메모리로부터 입력된다.

제안한 메모리의 세부적인 구조는 그림 5와 같다. 제안한 메모리 제어기는 버스를 사용하기 위한 승인을 받기 위해서 마스터와 신호를 주고 받는 마스터 선택기(MS, Master Selector), 허가 신호를 디코딩하고 컨트롤 신호의 상태를 정의한 마스터 중재기(MA, MAster Arbiter), SDRAM의 파라미터를 저장하고 뱅크의 준비 여부, 읽고/쓰기 가능 여부, PRECHARGE와 REFRESH의 가능 여부를 확인하여 system과 읽고/쓰기가 준비되었다는 신호를 출력, SDRAM의 실질적인 입력신호를 생성하는 메모리 신호 생성기(MSG, Memory Signal Generator), 마스터로부터 명령어 코드를 분석하는 명령어 디코더(CD, Command Decoder), 생성된 입력신호를 저장하고 마스터에서 직접 쓰기 데이터를 입력 받는 데이터 버스(DB, Data Bus)로 구성된다.

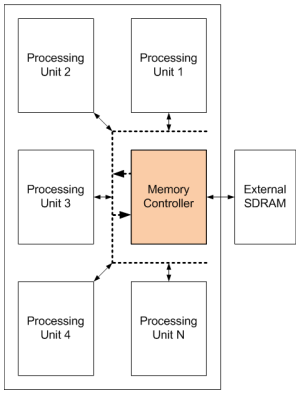

그림 3. SoC 내에서 메모리 제어기의 구성
Fig. 3. Memory controller configuration in SoC

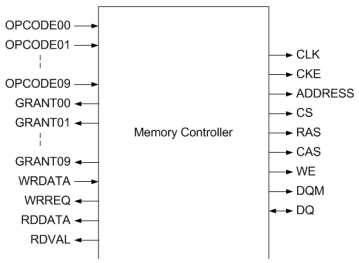

그림 4. 메모리 제어기의 최상위 구조
Fig. 4. Top architecture of memory controller

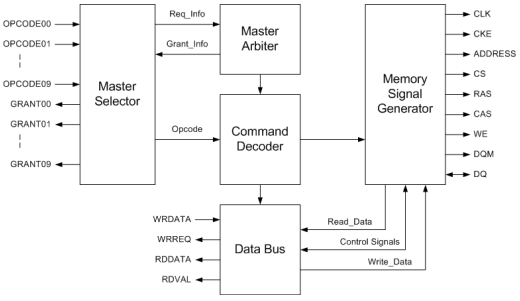

그림 5. 메모리 제어기의 세부 구조
Fig. 5. Detail architecture of memory controller

MS 블록은 버스를 사용하기 위한 승인을 받기 위해서 프로세서 유닛과 신호를 주고 받는다. 그 절차는

OPCODE와 GRANT 신호로 결정된다. MS 블록은 입력 신호로 OPCODE 신호를 받게 된다. OPCODE는 요청 신호를 포함하는데 마스터가 메모리의 사용을 요청하면 이 신호의 상태가 변화한다. 이 신호의 변화에 MS는 어떤 주변장치가 메모리를 요청하는지 확인 후 GRANT 신호로 메모리의 사용여부를 출력한다. OPCODE 신호는 주소, 뱅크 주소, R/W 등으로 구성되어 있으며 30 비트이다. OPCODE 신호에 의해 메모리를 사용하는 마스터가 정해지면서 그에 맞는 OPCODE 신호에 대한 GRANT 신호에 같이 출력한다.

MA 블록은 GRANT 신호를 디코딩하고 컨트롤 신호의 상태를 정의한 블록이다. GRANT 신호를 디코딩해서 읽고/쓰기 동작의 여부를 확인한다. MSG 블록에서 출력된 동작 요청 신호에 따라서 뱅크와 열을 선택해 정의된 상태값들을 출력한다. 제어신호가 idle상태 일 경우에 동작대기 신호를 출력한다. MA는 실질적인 메모리 제어를 하는 블록이다. MA 블록에서 시스템의 준비여부와 읽고/쓰기에 대한 요청 신호를 입력 받는다. SDRAM의 파라미터를 미리 저장해 두는 버퍼 기능과 FSM으로 설계되어 직접적으로 SDRAM의 컨트롤 신호에 맞게 출력하고, 디코딩 블록으로 ready 신호를 출력하는 기능이 있다. 또한 SDRAM에 직접적인 입력신호를 주는 기능도 담당한다. 쓰기 데이터를 제외한 각 입력신호들은 디코딩 되어서 출력되고, 입력된 신호들은 한 클럭 버퍼링되어서 SDRAM으로 출력된다. 쓰기 데이터는 쓰기 동작신호가 되면 SDRAM으로 출력된다. 주소가 시분할 다중화 방식으로 입력되며 /RAS와 /CAS제어 신호의 시분할 입력이 필요하며 모든 제어신호를 클럭에 동기시켜 구동한다. 디바이스의 초기화 과정에서 BURST 길이, 그리고 cas latency등의 파이프 라인 셋팅을 프로그램 입력을 통해 설정이 가능하며 데이터의 손실을 막기 위한 리프레쉬(REFRESH)기능을 수행한다. 만일 단일 마스터만 존재하는 칩을 설계하고자 한다면 MSG만 사용하여도 무방하다.

DB 블록은 GRANT 신호에 따라서 해당 마스터의 OPCODE를 열 주소, 행 주소, 뱅크 주소 등으로 디코딩하는 기능을 수행한다.

3.2 메모리 제어기의 동작

MS 블록은 메모리 제어기를 사용하려는 마스터들의 우선순위를 정한다. 여기서 서로 다른 두 개 이상의 마스터가 요청신호를 보내서 제어기를 사용하려 할 때는 첫 번째 OPCODE에 우선순위가 돌아간다. 여기서 요청신호를 통해 선택된 마스터의 OPCODE 신호는 4 비트를 더 붙여져 34 비트로 허가정보로 출력된다. 여기서 4 비트는 MA 블록에서 허가에 대한 준비 신호와 마스터들에게서 요청 OPCODE 신호의 여부에 대한 정보이다. 또한 마스터들에 대한 5 비트의 ID를 요청 마스터의 정보 신호로 출력한다.

MA 블록이 읽기 데이터를 마스터로 출력하는 것은 MSG 블록에서 출력되는 start read 데이터 신호에 의해서 결정된다. start read 데이터 신호를 입력 받으면 MA 블록은 start read data 신호와 valid read data 신호를 마스터로 출력한다. 쓰기 데이터의 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. MA 읽고/쓰기 데이터의 경우 허가신호에 의해서 단일 동작과 시퀀스 동작이 변화한다.

MA 블록은 FSM(finite state machine)으로 설계되어 있어 memory accelerator 블록에서 입력받은 PRECHARGE, REFRESH 등의 신호에 따라 MSG가 각 상태로 활성화된다. 대기신호를 입력 받게 되면 시작하라는 신호를 MSG 블록으로 출력함과 동시에 마스터로 대기신호를 출력한다. MA 블록으로부터 허가신호와 허가된 마스터에 대한 신호를 입력받는다. 허가신호로부터 읽고/쓰기의 구분을 하고, 뱅크 주소, 열 주소와 함께 시스템의 신호로 출력한다. 디코딩한 허가신호, 허가된 마스터, 시스템, 마스터의 요청정보와 허가대기 여부정보를 가진 신호를 MSG 블록으로 출력한다.

MSG 블록은 우선 SDRAM의 파라미터들을 저장해둔다. MA로부터 4가지의 신호를 받아서 마스터의 ID, 읽고/쓰기의 구분, 뱅크 주소, 열 주소, 행 주소, BURST 모드의 신호로 디코딩한다. SDRAM으로 출력되는 직접적인 신호를 출력하는 블록은 FSM으로 설계되어 있다. 이 블록은 파마미터를 로드하여 각 뱅크의 준비 여부, 읽고/쓰기 가능 여부, PRECHARGE와 REFRESH의 가능 여부를 확인하고, 디코딩 블록으로 보내 시스템과 읽고/쓰기가 준비되었다는 신호를 출력한다.

그림 6에는 메모리 제어기와 마스터 간의 간단한 데이터 입출력 예를 보여주고 있다. OPCODE에 의한 마스터의 요청에 따라 메모리 제어기가 허가신호를 내보낸

다. OPCODE가 쓰기 모드이면 마스터는 메모리 제어기의 쓰기요청신호(WRREQ)에 맞추어 데이터(WRDATA)를 출력한다. OPCODE가 읽기 모드이면 마스터는 데이터유효신호(RDVAL)에 맞추어서 SDRAM으로부터 읽혀진 데이터(RDDATA)를 읽으면 된다.

SDRAM의 제어신호는 매우 복잡하다. 그러나 그림 6에서 보여지는 것과 같이 본 논문에서 제안하고 있는 메모리 제어기를 사용하면 단순히 플래그 신호와 데이터 라인만을 이용하여 간단하게 SDRAM을 사용할 수 있다. 또한 AMBA 버스 혹은 AXI 버스와도 연동 및 인터페이스가 쉽게 이루어질 수 있음을 확인할 수 있다[21].

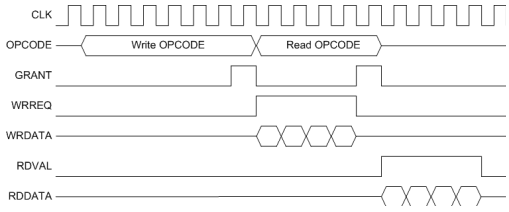

그림 6. 메모리 제어기와 마스터간의 데이터 입출력
Fig. 6. Data input/output between memory controller and master

Ⅳ. 실험결과

4.1 하드웨어 구현 결과

앞 장까지 설명한 메모리 제어기는 VHDL[22]로 설계하였으며, Altera의 Quartus Ⅱ 환경을 사용하였다. VHDL 코딩에 의한 함수적 검증을 거쳐 합성단계를 수행한 후 설계사양에 맞는 지연시간을 각 동작에 따라 검증하였다. 메모리 제어기의 자원 사용율은 표 1에 나타내었고, FPGA기반의 RTL 합성도는 그림 7에 나타냈다. 구현한 메모리 제어기는 FPGA 환경에서 최대 174.28MHz로 동작할 수 있었다. 이것은 FPGA 환경에서도 모든 SDRAM에 대해 동작이 가능하다는 것을 보여준다. 타겟 FPGA는 Cyclone II를 사용하였다. 총 943개의 LUT를 사용하고 566개의 레지스터를 사용하여 구현되었다.

표 1. 자원 사용율
Table 1. Resource utilization

Logic utilization	Used
Combinational ALUTs	943
Dedicated logic registers	566

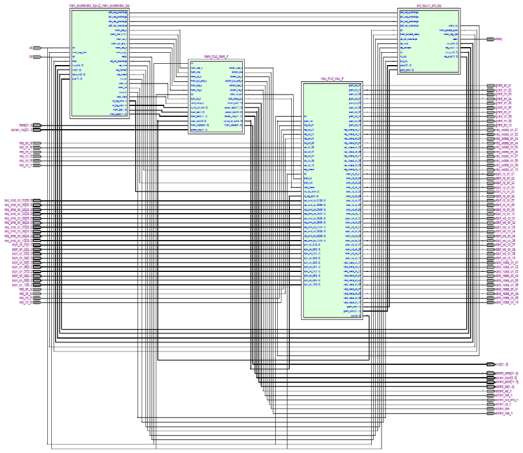

그림 7. 메모리 제어기의 FPGA 기반 RTL 합성도
Fig. 7. FPGA-based RTL synthesis of memory controller

4.2 시뮬레이션 결과

시뮬레이션은 Cadence사의 Modelsim으로 수행하였고, 삼성의 메모리 모델을 이용하였다. 그림 8은 BURST를 사용하지 않는 BURST 길이가 1(default)일 경우에 쓰기와 읽기를 한 결과이다. 초기화할 때 PRECHARGE와 REFRESH, register mode selection, active 과정을 거친 뒤 쓰기 신호를 주었을 때 쓰기가 되고 읽기 신호를 주면 2 clock 뒤에 읽기가 된다. 그림 9는 BURST 길이가 4일 때 BURST동작의 결과인데, 쓰기 동작 신호를 주었을 때 4 clock 동안 데이터가 4개 들어가는 것을 볼 수 있다. 다음 읽기 동작 신호를 주었을 때 마찬가지로 4개의 데이터가 4 clock 동안 읽히는 것을 볼 수 있다. 그림 10과 11은 FULL PAGE 모드를 선택했을 때 영상의 한 라인이 쓰기/읽기되는 결과이다. 마지막으로 그림 12는 영상의 한 프레임을 메모리에 읽고 쓴 결과를 보이고 있다.

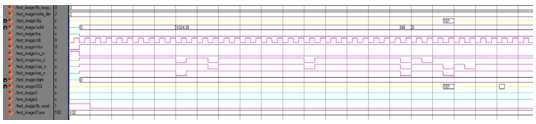

그림 8. 읽고 쓰기(기본 모드)
Fig. 8. WRITE & READ(default)

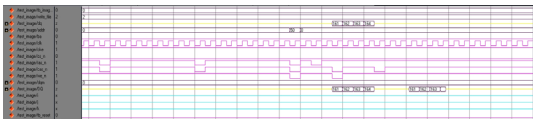

그림 9. 읽고 쓰기(Burst 길이 = 4)
Fig. 9. WRITE & READ(Burst Length = 4)

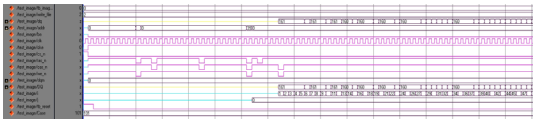

그림 10. 쓰기(FULL PAGE 모드)
Fig. 10. WRITE(full page mode)

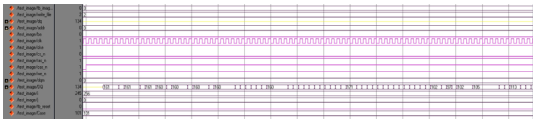

그림 11. 읽기(FULL PAGE 모드)
Fig. 11. READ(full page mode)

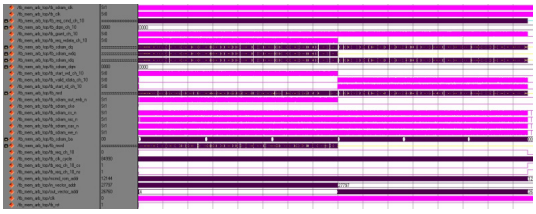

그림 12. 한 프레임의 영상을 읽고 쓰기
Fig. 12. One frame image read/write

V. 결 론

본 논문에서는 비디오 처리를 위하여 SDRAM을 제어할 수 있는 범용적인 메모리 제어기를 설계하였다. 구현된 메모리 제어기는 버스를 사용하기 위한 승인을 받기 위해서 마스터와 신호를 주고 받는 마스터 선택기, 허가 신호를 디코딩하고 컨트롤 신호의 상태를 정의한 마스터 중재기, SDRAM의 파라미터를 저장하고 뱅크의 준비 여부, 읽고/쓰기 가능 여부, PRECHARGE와 REFRESH의 가능 여부를 확인하여 마스터와 읽고/쓰기가 준비되었다는 신호를 출력, SDRAM의 실질적인 입력신호를 생성하는 메모리 신호 생성기, 마스터로부터 명령어 코드를 분석하는 명령어 디코더, 생성된 입력신호를 저장하고 마스터에서 직접 쓰기 데이터를 입력 받는 데이터 버스로 구성된다. 이 메모리 제어기는 총 943개의 LUT를 사용하고 566개의 레지스터를 사용하여 구현되었고, 174.28MHz의 주파수로 동작하였다.

참고문헌

[1] B. Furht, “”Multimedia systems: An overview,”” IEEE Multimedia, vol. 1, no. 1, pp. 47-59, Spring 1994.

[2] F. Catthoor et al., Custom Memory Management Methodology: Exploration of Memory Organization for Embedded Multimedia System Design. Norwell, MA: Kluwer, 1998.

[3] John G. Ackenhusen, Real-time signal processing: Design and Implenentation of signal processing systems, Prentice Hall , pp. 290-319, 1999

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

[5] Micron Technology, Inc. MT48LC8M32B2 SDRAM

[6] S. Hosseini-Khayat and A. D. Bovopoulos, “”A simple and efficient bus management scheme that supports continuous streams,”” ACM Trans. Comput. Syst., vol. 13, no. 2, pp. 122 – 140, 1995.

[7] J. Carter et al., “”Impulse: Building a smarter memory controller,”” in Proc. HPCA, Jan. 1999, pp. 70-79.

[8] S. Rixner et al., “”Memory access scheduling,”” in Proc. ISCA, Vancouver, BC, Canada, Jun. 2000, pp. 128 – 138.

저자소개

최현준(Hyun-Jun Choi)

2003년 2월 : 광운대학교 전자재료 공학과 졸업(공학사)
2005년 2월 : 광운대학교 일반대학원 졸업(공학석사)
2009년 2월 : 광운대학교 일반대학원 졸업(공학박사)
2009년 3월~2010년 2월 : 광운대학교 연구교수
2010년 3월~현재 : 안양대학교 정보통신공학과 조교수
※관심분야 : 영상압축, 워터마킹, 암호학, FPGA/ASIC 설계, Design Methodology

서영호(Young-Ho Seo)

1999년 2월 : 광운대학교 전자재료 공학과 졸업(공학사)
2001년 2월 : 광운대학교 일반대학원 졸업(공학석사)
2004년 8월 : 광운대학교 일반대학원 졸업(공학박사)
2003년 6월~2004년 6월 : 한국전기연구원 연구원
2005년 9월~2008년 2월 : 한성대학교 조교수
2008년 3월~현재 : 광운대학교 교양학부 조교수
※관심분야 : 2D/3D 영상 및 비디오 처리, 디지털 홀로그램, SoC 설계, 워터마킹/암호화

김동욱(Dong-Wook Kim)

1983년 2월 : 한양대학교 전자공학과 졸업(공학사)
1985년 2월 : 한양대학교 대학원 졸업(공학석사)
1991년 9월 : Georgia 공과대학 전기공학과 졸업 (공학박사)
1992년 3월~현재 : 광운대학교 전자재료공학과 정교수
2000년 3월~2001년 12월 : 인티스닷컴(주) 연구원
2007년 3월~현재 : (사)실감미디어산업협회 이사
2009년 3월~현재 : 실감미디어연구소 소장
※관심분야 : 디지털 VLSI Testability, VLSI, CAD, DSP 설계, Wireless Communication