컴퓨터 공학 기초 실험2 보고서

실험제목: Memory & Bus

실험일자: 2018년 11월 07일 (수)

제출일자: 2018년 11월 13일 (화)

학 과: 컴퓨터공학과

담당교수: 공진흥 교수님

실습분반: 수요일 0, 1, 2

학 번: 2015722025

성 명: 정 용 훈

1. 제목 및 목적

A. 제목

Memory & Bus

B. 목적

RAM으로 동작하는 간단한 Memory를 구현해보고 내부적으로 어떤 신호를 통해 어떤 값을 출력하도록 조정되는지를 이해하고 이것이 어떠한 방식으로 동작하는지 배운다. Bus가 무엇인지 알아보고, 이에 대한 개념을 적용하여 2개의 master와 2개의 slave로 구성된 간단한 bus를 구현한다.

2. 원리(배경지식)

A. Memory

Memory는 크게 RAM과 ROM으로 나뉜다. RAM이란 기억된 정보를 읽기도 하며 다른 정보를 기억시킬 수도 있는 메모리이다. 특징으로 RAM은 전원이 끊어지면 정보가 날아가는 휘발성 메모리이다. 그러한 이유로 실제로는 컴퓨터의 주기억장치, 응용 프로그램의 일시적 로딩, 데이터의 일시적 저장 등에 사용되고 있다. RAM은 SRAM, DRAM으로 나눌 수 있다 각각 Memory에 대한 설명은 아래와 같다.

ROM

약자로 READ Only Memory이다. 컴퓨터의 판독전용 기억장치를 뜻하며, 전원이 끊어져도 RAM과는 다르게 정보가 없어지지 않는 불 휘발성 기억장치이다. 문자 패턴 발생기나 코드 변환기처럼 행하는 처리가 일정하며 다량으로 사용되는 것은 기억할 정보를 소자의 제조와 동시에 설정하기 때문이다. 이러한 이유로 ROM은 정보를 소자의 구조로서 기억하고 있으므로 바꾸어 쓸 수가 없는 것이다.

SRAM

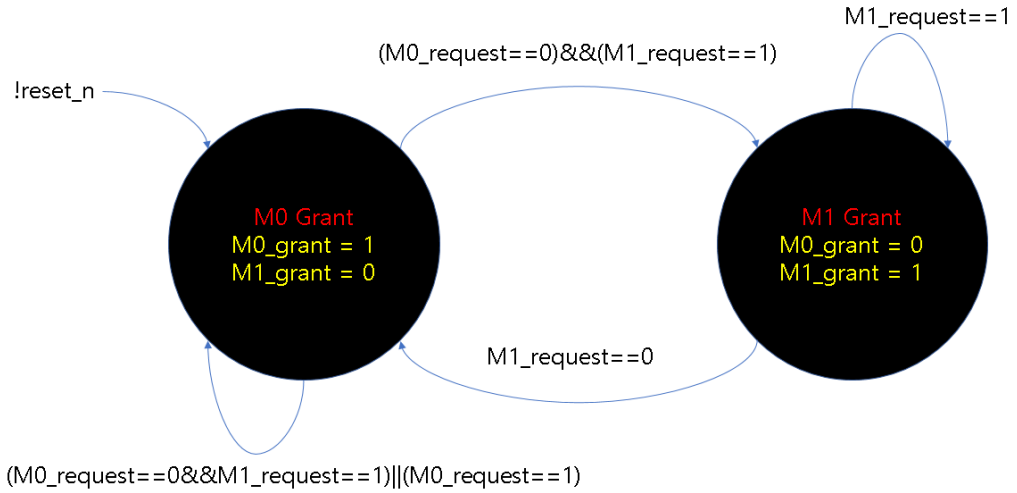
S(Static) RAM은 전원이 공급되는 한 기억된 데이터가 지워지지 않는다. 플립플롭 방식의 메모리 셀을 가진 임의 접근 기억장치로서 전원 공급이 계속되는 한 저장된 내용을 계속 기억하며, 복잡한 재생 클럭이 필요 없기 때문에 소용량의 메모리나 캐시 메모리에 주로 사용된다. 소비전력이 적고 D램에 비해 정보처리속도가 5배 정도 빨라 주로 PC의 수치계산 용도로 이용된다.

DRAM

D(Dynamic) RAM은 전원이 차단될 경우 저장되어 있는 자료가 소멸되는 특성이 있는 휘발성 기억 소자이며, 시간이 지나가면 축적된 전하가 감소되기 때문에 전원이 차단되지 않더라도 저장된 자료가 자연히 소멸되는 단점이 있다. 따라서 지속적으로 재 기록할 수 있고 저장용량이 커서 PC의 주요 메모리로 쓰인다.

B. Bus

Computer에서 Bus란 computer간에 데이터를 전송하는 통신 시스템을 말한다. 이러한 표현에는 관련된 모든 하드웨어 부품들(선, 광 파이버 등)및 통신 프로토콜을 포함한 소프트웨어를 아우른다. 쉽게 보충설명 하자면 데이터를 주고받는 component 사이에서 어떤 component가 데이터를 주며, 준 데이터를 어떤 component에서 처리하는지 결정하게 해주는 component이다. 또한 Bus의 장점으로 새로운 component를 추가하기 쉽고, 가격이 저렴한 특징을 갖는다. Bus내부에는 Arbiter라고하는 module이 존재하는데 이는 어원 그대로 중재라는 뜻으로 Bus내에서 데이터가 이동하는데 관여하는 module이다. Bus는 request신호를 받아 알맞은 master가 bus의 소유권을 갖게 되며 grant signal을 통해 확인할 수 있다. 다음 그림은 Arbiter의 역할을 diagram으로 나타낸 것이다.



그림에서 보는것과 같이 request신호에 따라 master가 결정되는 것을 확인할 수 있다. request신호 둘다 없다면 소유권은 M0, 즉 master0이 소유권을 갖게 된다. Bus의 Master와 Slave의 관계는 설계 세부사항에서 좀 더 자세하게 다루어 보겠다.

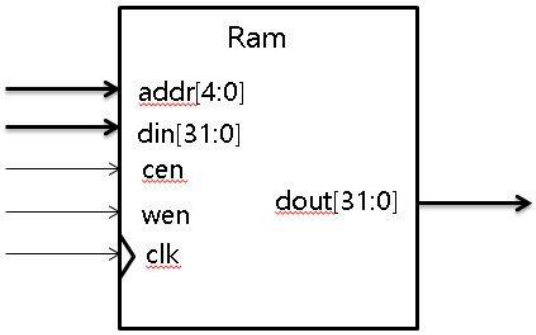
3. 설계 세부사항

A. Memory

-32bit 길이의 data를 32개까지 address에 기반하여 저장합니다.

-module을 실행하면 32개의 register를 초기화 합니다.

-아래와 같은 간단한 symbol로 표현할 수 있습니다.



구분	이름	비트 수	설명
Input	clk	1	Clock
	cen	1	Chip enable
	wen	1	Write enable
	addr	5	Address
	din	32	Data in
Output	dout	32	Data out

Input		Operation
cen	wen	
1	1	Write data into memory which address' pointing
1	0	Write memory data which address' pointing into 'dout'
0	x	Set 'dout' as 0

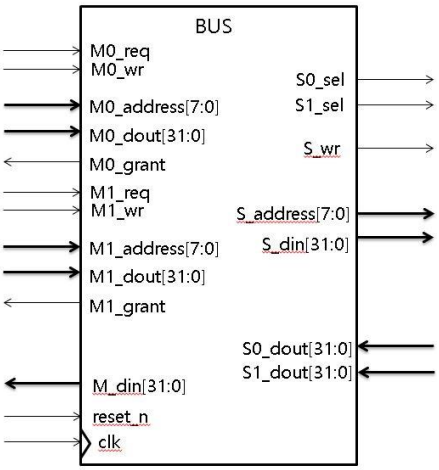
B. Bus

-여러 component간에 data를 전송할 수 있도록 연결해주는 component.

-2개의 slave와 master를 가지고 있습니다.

-slave는 아래와 같은 주소 범위를 갖게 되며, 이를 넘어가는 주소에 접근하게 되면 어떠한 slave도 선택하지 않습니다.

Memory map	
Slave 0	0x00 ~ 0x1F
Slave 1	0x20 ~ 0x3F

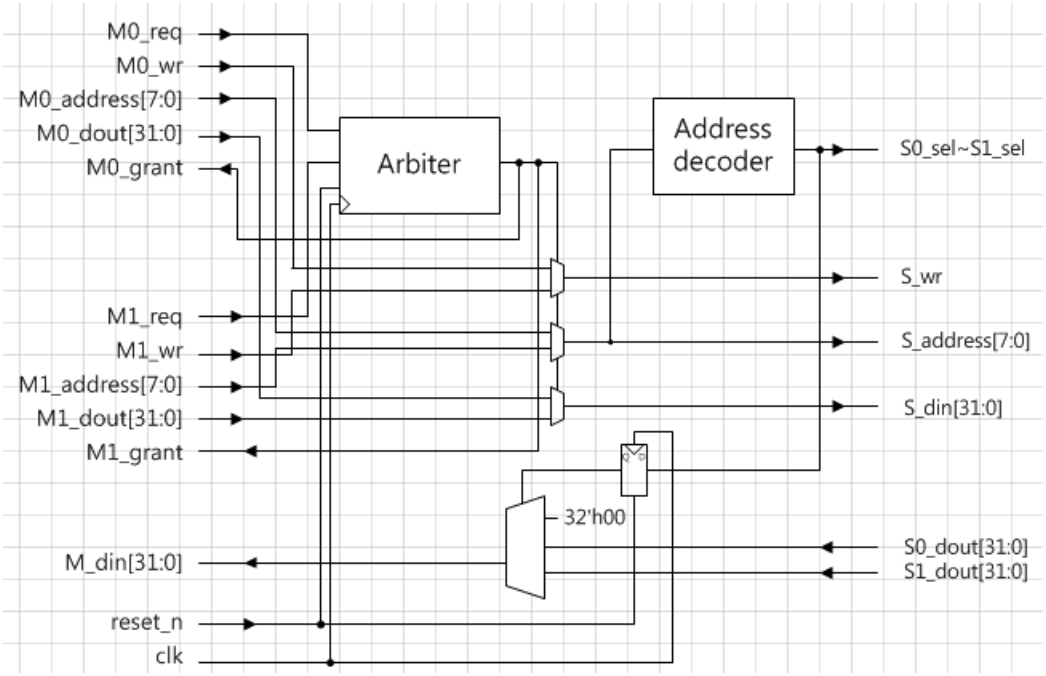


-master가 필요시에 request signal을 high상태로 유지하고, 이에 대한 확인으로 grant signal을 받은 후에 data를 전송할 수 있습니다.

-grant signal을 받은 master가 request signal을 high 상태로 유지하는 동안에는 bus의 소유권을 유지합니다.

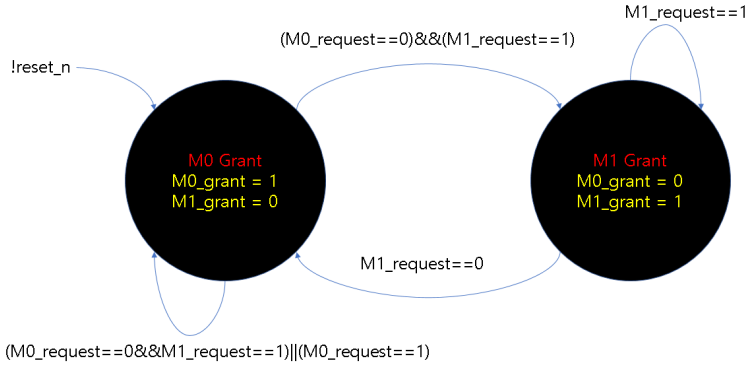
Direction	Port name	Description
Input	clk	Clock
	reset_n	Active low reset
	M0_req	Master 0 request
	M0_wr	Master 0 write/read
	M0_address[7:0]	Master 0 address
	M0_dout[31:0]	Master 0 data output
	M1_req	Master 1 request
	M1_wr	Master 1 write/read
	M1_address[7:0]	Master 1 address
	M1_dout[31:0]	Master 1 data output
	S0_dout[31:0]	Slave 0 data output
	S1_dout[31:0]	Slave 1 data output
Output	M0	Master 0 grant
	M1	Master 1 grant
	M_din[31:0]	Master data input
	S0_sel	Slave 0 select
	S1_sel	Slave 1 select
	S_address[7:0]	Slave address
	S_wr	Slave write/read
	S_din[31:0]	Slave data input

Bus의 전체적인 구조는 다음과 같다.



- Arbiter

  -master의 grant에 따른 state를 결정해주는 module로써 각 master의 request와 현재 state에 따라 출력 값이 바뀝니다.



- Address decoder

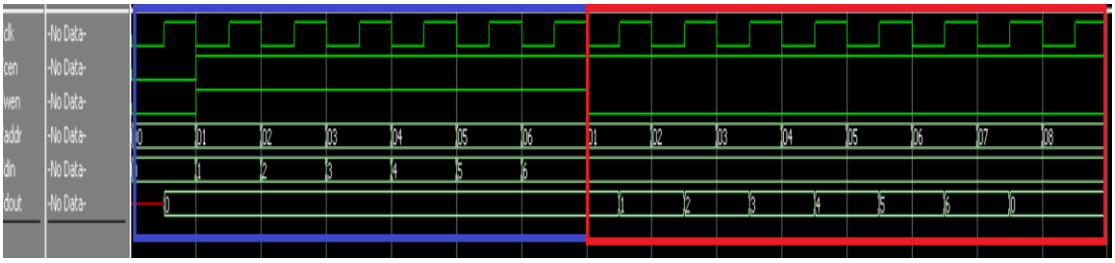
  -주소 값을 입력 받아 slave의 주소 범위에 따라서 selector역할을 할 output을 출력합니다.

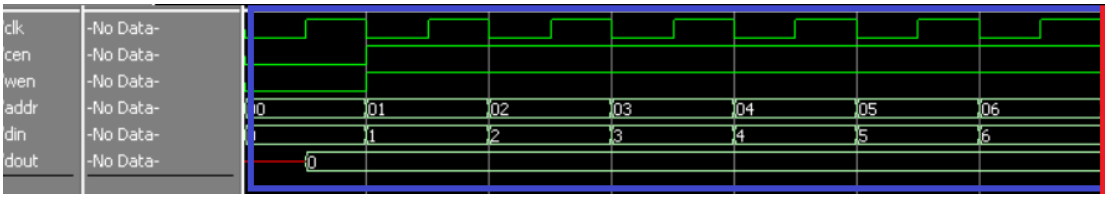
Input	Condition			Output	
	0x00	0x20	0x40	S0_sel	S1_sel
address	>=	<		1	0
		>=	<	0	1
	<		>=	0	0

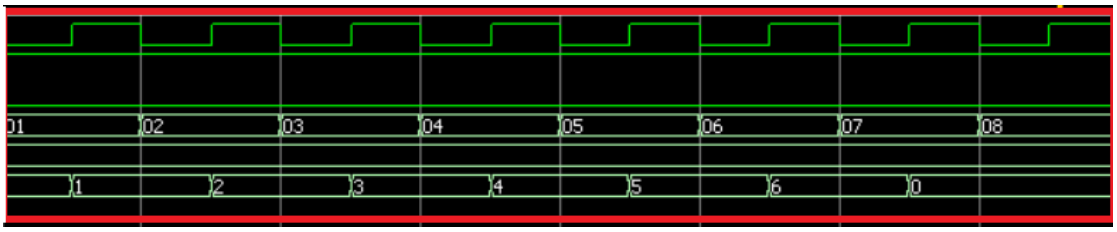
4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

Memory

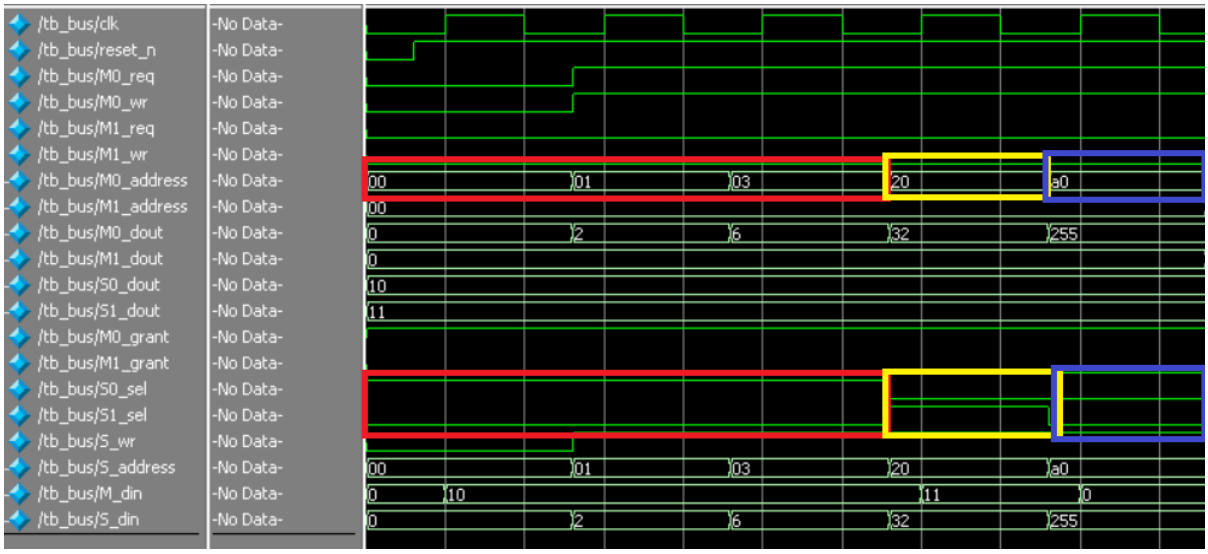


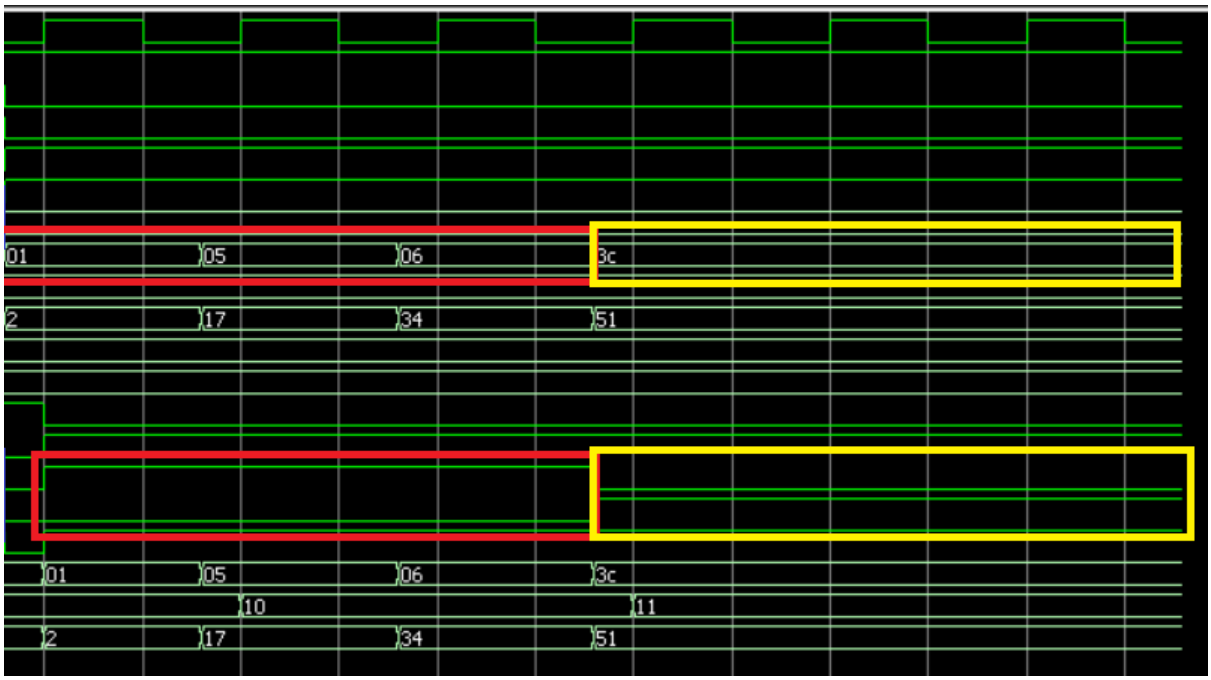




파란 부분은 data를 주소에 저장하는 과정임 wen signal이 1이되면 동작하는 결과이다. 빨간색 부분은 wen signa이 0이되고 test bench에서 받은 주소에 있는 data를 읽어오는 것을 확인할 수 있다. 또한 메모리 07과 08에는 데이터가 없기 때문에 기존에 Memory를 0으로 초기화 했기 때문에 0의 값이 나오는 것을 확인할 수 있다.

Bus



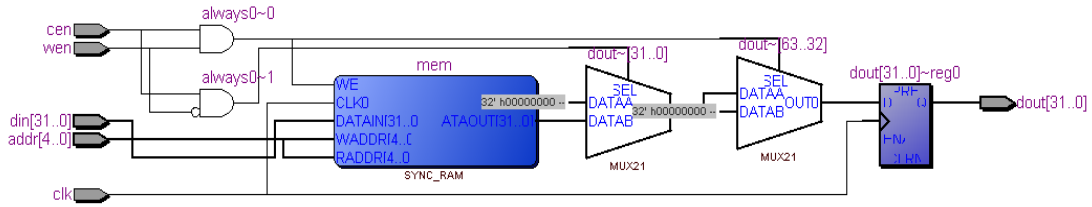


해당 wave form을 보면 request에 따라 정보를 주는 master가 설정되며 그에 따라grant signal도 변경되는 것을 확인할 수 있습니다. (첫번째 사진이 M0, 두번째 사진이 M1) 것을 확인할 수 있으며 Bus를 소유하고 있는 master가 주는 address에 따라 sel신호가 바뀌는 것을 확인할 수 있습니다. Address가 Slave들의 범위내에 있지 않으면 sel신호는 모두 0의 값을 가져야합니다.

B. 합성(synthesis) 결과

Memory

<RTL Viewer>



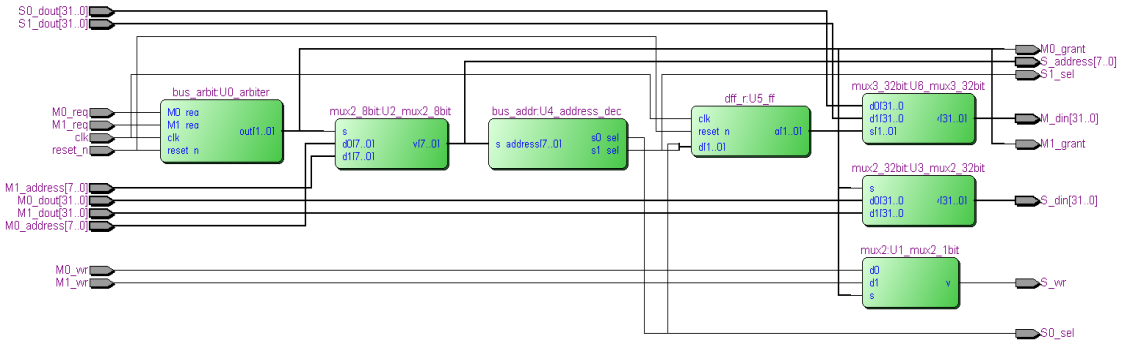
Memory의 RTL Viewer이다. cen신호와 wen신호가 mux로 들어가는 것을 확인할 수 있다.

<Flow summary>

Flow Summary	
Flow Status	Successful - Sat Nov 10 17:25:46 2018
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	memory
Top-level Entity Name	memory
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Total logic elements	1,237 / 68,416 (2 %)
Total combinational functions	722 / 68,416 (1 %)
Dedicated logic registers	1,056 / 68,416 (2 %)
Total registers	1056
Total pins	72 / 622 (12 %)
Total virtual pins	0
Total memory bits	0 / 1,152,000 (0 %)
Embedded Multiplier 9-bit elements	0 / 300 (0 %)
Total PLLs	0 / 4 (0 %)

Bus

<RTL Viewer>



Bus의 RTL Viewer이다. 결론에서 언급하게 될 flip flop이 sel의 신호를 받아 동작하는 것을 확인할 수 있으며 sel의 신호는 master의 address에 따라 바뀌게 된다.

<Flow summary>

Flow Summary	
Flow Status	Successful - Sat Nov 10 17:29:44 2018
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	bus
Top-level Entity Name	bus
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Total logic elements	117 / 68,416 (< 1 %)
Total combinational functions	117 / 68,416 (< 1 %)
Dedicated logic registers	4 / 68,416 (< 1 %)
Total registers	4
Total pins	227 / 622 (36 %)
Total virtual pins	0
Total memory bits	0 / 1,152,000 (0 %)
Embedded Multiplier 9-bit elements	0 / 300 (0 %)
Total PLLs	0 / 4 (0 %)

5. 고찰 및 결론

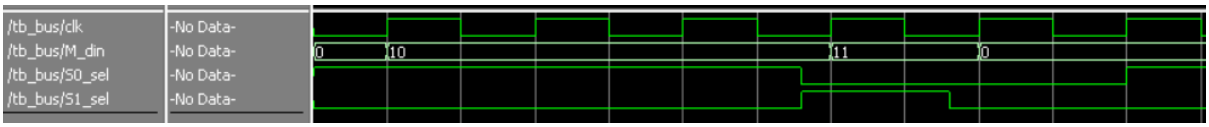
A. 고찰

이번 실습에서는 Project에 쓰일 Memory와 Bus를 구현해보았다. Memory구현은 for를 사용하여 메모리 내부를 0으로 모두 초기화 시킨 후 사용하였으며 구체적인 동작은 wen의 신호를 사용하여 데이터를 읽을지 쓸지 결정하게 된다. 이미 Register File과 FIFO에서 데이터를 읽고 쓰는 실습을 선행했기 때문에 어려운 실습은 아니었다. 이번 실습에서 Bus도 같이 구현하게 되었는데 처음 Bus를 구현할 때는 Master와 Slave의 관계를 명확하게 이해하지 못해서 구현하는데 어려움이 있었던 것 같다. Bus의 개념인 데이터의 오고가는 것을 다시 생각하며 Master와 Slave의 관계를 다시 적립하였으며 해당하는 request 신화 grant신호 주소의 범위를 나누어 sel신호를 나눈 것에 대해 이해할 수 있었다.

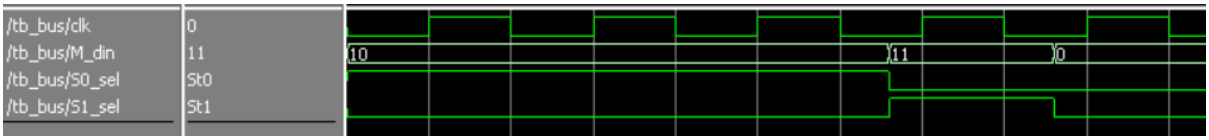
B. 결론

특히 Bus를 구현하면서 데이터의 이동에서 중간다리 역할을 해주어 전체적인 동작을 유연하고 효율성 있게 해줄 수 있다는 것을 배우게 되었다. 또한 생각해볼 문제로 Bus의 read operation을 수행할 때 sel신호를 flip flop을 거쳐 clock에 data값을 동기화 시켜 놓은 것을 확인할 수 있다. 그렇다면 왜 flip flop을 사용하였을까? 아래그림은 두 경우를 나타낸 wave form이다.

(Flip Flop을 거쳤을 때)



(Flip Flop을 거치지지 않았을 때)



두 wave form의 차이를 보면 sel신호에 따라 data in의 값이 바뀌는데 동기식과 비동기식의 차이만 나타나고 있다. 결론만 따지고 생각보면 Arbiter의 grant signal이 나오는 타이밍이 clock에 동기화되어 있기 때문에 Flip flop을 사용한 것이다. Bus구조를 봤을 때 grant signal에 따라 알맞은 master의 값이 선택되어 출력이 된다. 만약 Flip Flop을 사용하지 않는다면 grant signal이 바뀌는 경우 clock이 뛰기 전에 M_din의 값이 sel을 따라 바로 바뀌어 이전에 선택된 master의 grant signal을 따라가기 때문에 정확한 값이 나오지 않는 문제가 발생한다. 이런 원하지 않는 값이 나오는 것을 막기 위하여 Flip Flop을 사용한 것으로 보인다.

6. 참고문헌

김영민/Simple Bus/광운대학교/2018

김영민/Memory mapped IO/2018

RAM/https://terms.naver.com/entry.nhn?docId=1087343&cid=40942&categoryId=32832

ROM/https://terms.naver.com/entry.nhn?docId=1088735&cid=40942&categoryId=32832

BUS/ https://en.wikipedia.org/wiki/Bus_(computing)