D

컴퓨터 공학 기초 실험2 보고서

실험제목: SimpleBus&Memory

실험일자: 2023년 11월 17일 (금)

제출일자: 2023년 11월 25일 (토)

학 과: 컴퓨터공학과

담당교수: 이준환 교수님

실습분반: 금요일 0, 1, 2

학 번: 2020202037

성 명: 염정호

1. 제목 및 목적

A. 제목

SimpleBus&Memory

B. 목적

어드레스 기반으로 데이터를 저장하는 하드웨어인 램과 여러 컴포넌트들 간의 데이터 전송을 할 수 있도록 연결해주는 버스 모듈을 구현해 보고 작독 방식을 이해한다.

2. 원리(배경지식)

RAM : Randon Access Memory의 약자로 컴퓨터 내에서 프로그램이나 데이터를 일시적으로 저장하고 빠르게 엑세스 할 수 있는 주 기억장치 중 하나이다.

일반적으로 DDR이라는 기술로 구현되며 컴퓨터 메인보드에 장착되는 메모리 모듈 형태로 사용된다. 램은 현재 컴퓨터가 실행중인 프로그램, 작업, 그래픽, 시스템프로세스 등을 처리하는데 이용된다. 중앙처리장치에서 작업을 처리할 때 램은 해당 작업에 필요한 데이터와 명령을 저장하여 중앙처리장치가 필요할 때 바로 가져다 쓸 수 있도록 한다.

램은 휘발성 메모리로 일반적으로 전원이 차단되면 내용이 지워진다. 렘의 성능 요소는 메모리 레이턴시, 메모리 쓰루풋 및 대역폭, 메모리 레벨 병렬처리 총 세가지로 분류된다.

메모리 레이턴시

메모리에 있는 데이터에 대한 요청부터 검색이 완료 될 때 까지의 접근 시간이다.

데이터를 요청한 뒤 실제 데이터를 받는데 까지 걸리는 시간이가도 하다.

메모리 쓰루풋 및 대역폭

단위시간동안 메모리가 데이터를 처리할 수 있는 용량을 말한다.

메모리 대역폭

중앙처리장치가 병렬처리 기능을 통해 얼마나 빠르게 데이터를 처리할 수 있는 가이다.

시스템 버스 :

컴퓨터 시스템 내에서 데이터와 명령어를 전송하는데 사용되는 핵심적인 통로이다. 즉 컴퓨터의 다양한 구성 요소 간에 정보를 주고 받을 수 있도록 하는 전기적인 통로이다.

시스템버스의 종류는 다음과 같다.

주변 구성 요소 연결 : 주변 장치들을 연결하는데 사용되는 버슬, 그래픽 카드, 네트워크, 어댑터, usb와 같은 장치들의 연결에 사용된다.

메모리 버스: CPU와 주 기억 장치 간에 데이터를 주고 받는데 사용된다. Cpu가 메모리에 있는 데이터를 읽거나 쓸 때 사용된다.

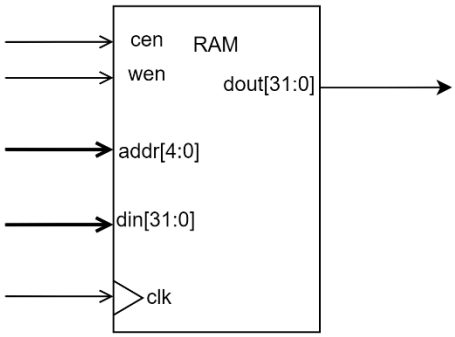
전원 버스: 전원 공급 장치와 각 부품을 연결하여 전기적인 전원 신호를 전달하는데 사용된다.

제어 버스 : 명령어 및 제어 신호 전송에 사용되며, 데이터와 함께 시스템의 동작을 제어

하는데 주요한 역할을 한다.

3. 설계 세부사항

RAM



Direction	Port name	Description
Input	clk	clock
	cen	Chip enable
	wen	Write enable
	addr[4:0]	Address
	din[31:0]	Data input
Output	dout[31:0]	Data out

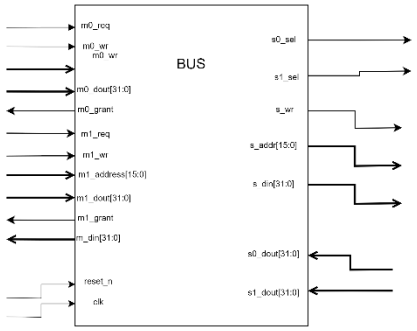
주소값의 크기는 5이며, data의 크기느 32bit이다.

cen과 wen이 모두 1인 경우 주소가 가리키는 메모리에 입력 받은 데이터를 저장한다. 이 때 dout값은 0을 출력한다.

Cen이 1이고 wen이 0이면 주소가 가리키는 메모리 값을 dout을 통해 출력한다.

Cen이 0 인경우 dout은 0을 출력한다.

bus



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

Pin Description

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 out
	s1_dout[31:0]	Slave 1 data out
Output	m0_grant	Master 0 grant
	m1_grant	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

두 개의 master와 slave를 가지고 있다.

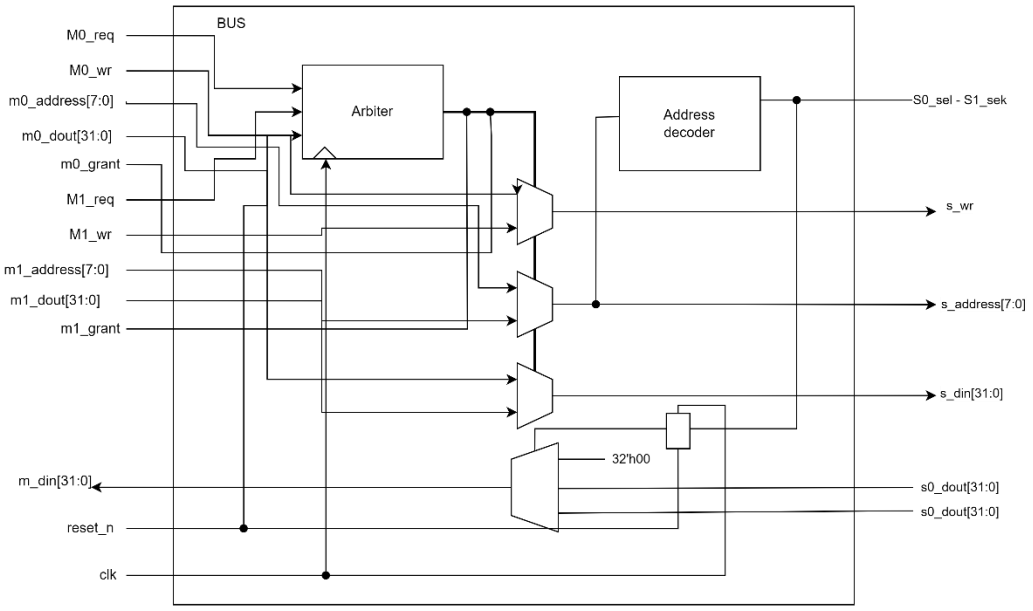
주소의 범위는 8 bit이다

데이터는 32bit데이터 이다.

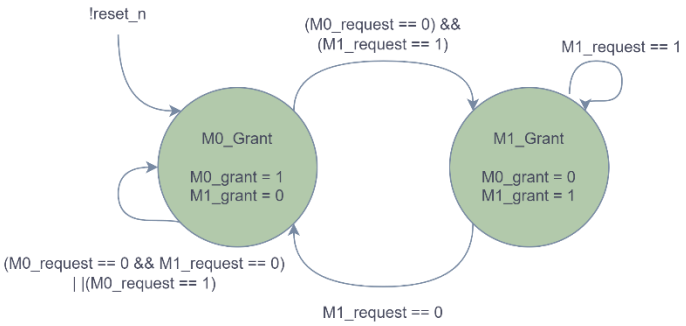
각 메모리 주소 범위는 위와 같다.

데이터 범위에 해당되지 않는 주소가 들어왔을 경우 어느 slave도 출력되지 않아야 한다.

Design



아비터 모듈에서 request에 따라 자신이 원하는 데이터를 전송할 수 있다. Request signal이 1인동안에 다른 요청에 의해 버스의 소유권을 뺐기지 않고 데이터 전송이 가능하다. 두 개의 마스터 모두 데이터를 요청하지 않는다면 초기 세팅된 주소의 데이터를 전송 받는다.

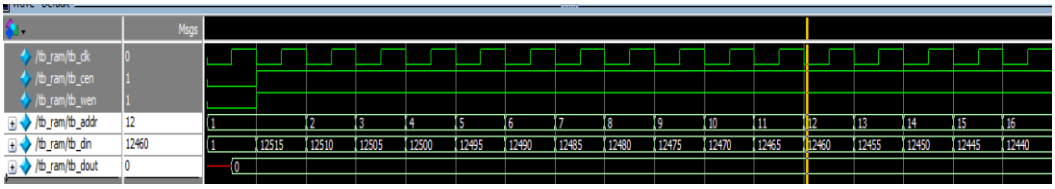


Request signal이 1인 동안에는 버스의 소유권을 뺏기지 않고 데이터 전송을 할 수 있도록 설계

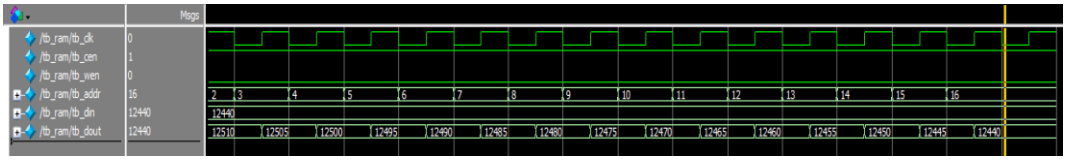
4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

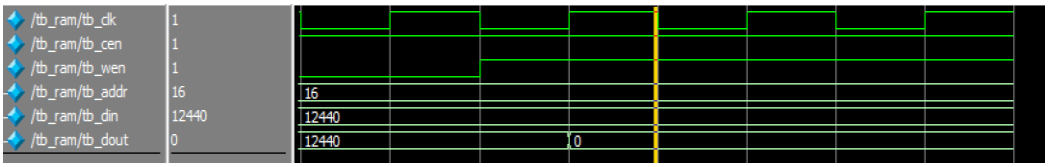
Ram - 데이터 입력 데이터를 입력하는 동안은 데이터 출력은 0으로 설정



Ram – 데이터 출력

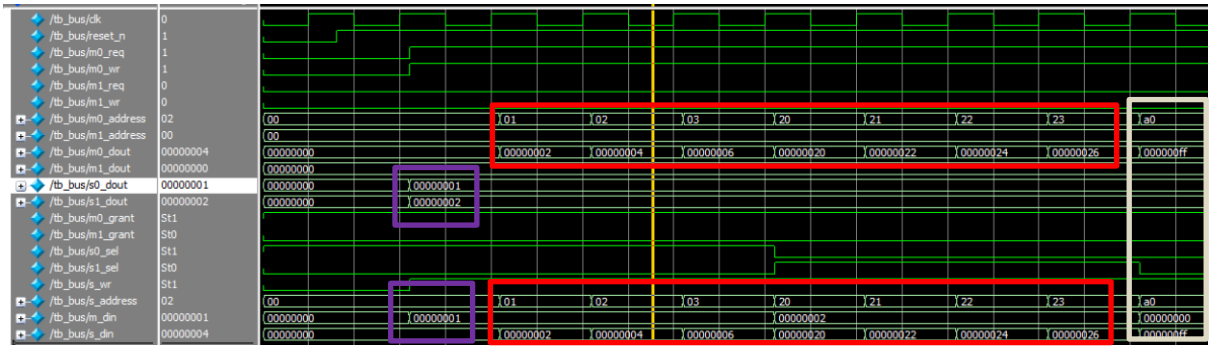


저장된 데이터 값이 차례데로 출력되는 것을 확인 할 수 있다.



다시 입력을 실행시키면 출력이 0이 되는 것을 확인 했다.

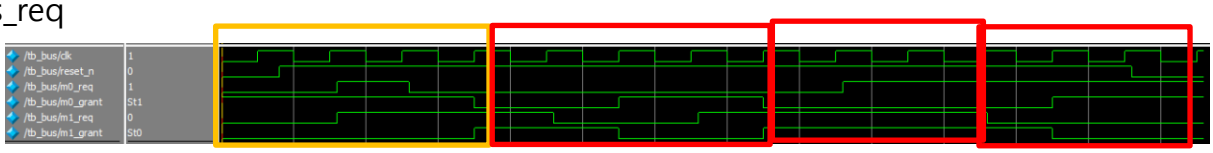
bus



req값에 따라 s0를 선택 해 출력한다.

Address로 선택한 주소 값 이외의 것이 입력될 경우 0을 출력한다.

Bus_req



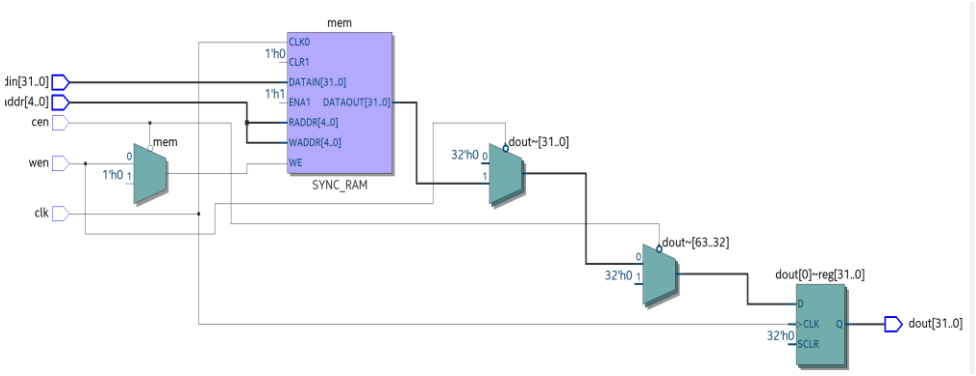
Reset이 0이고 req0 ,1 둘 다 0인 경우 m0선택 req동시에 1인 경우 이전 값 유지

Req 0 = 0 req1 =1일 때 m1 grant가 1로 변경 이후 req값이 둘 다 0으로 변할 경우 다시 grant 0 이 선택된다.

Grant1을 선택하고 있을 때 req 0 이1 로 변해도 값은 변화하지 않는다. 이후 리셋이 활성화 되면 다시 grant1 선택

Grant1이 선택 되었을 대 req 0 =1 ,req 1 = 0인 경우 grant 0로 값 변화를 확인했다.

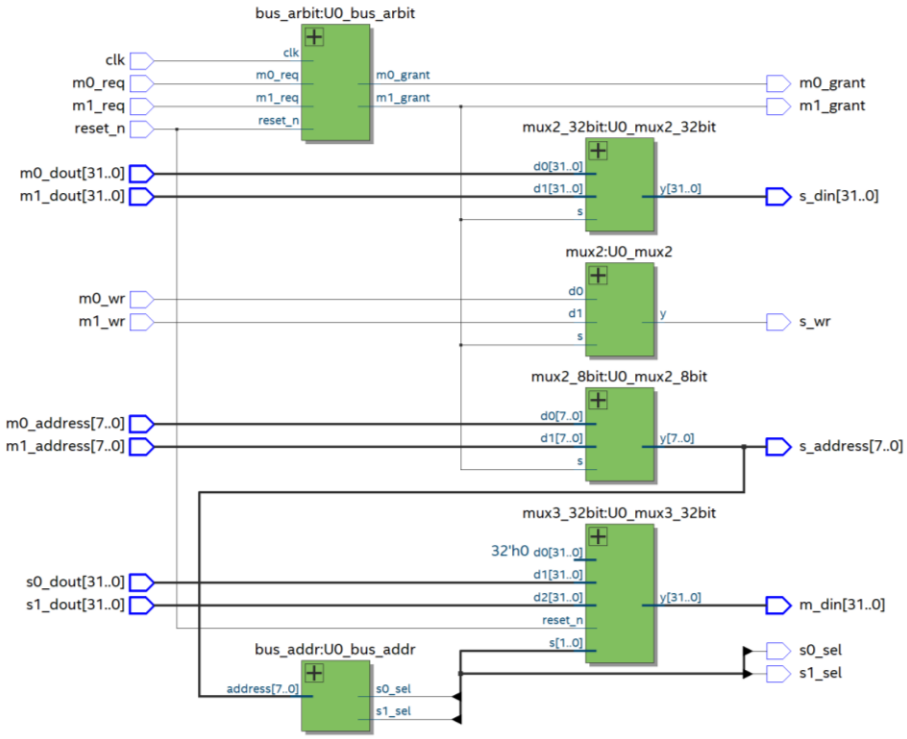
B. 합성(synthesis) 결과



Flow Summary

<<Filter>>

Flow Status	Successful - Sat Nov 25 10:06:24 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	ram
Top-level Entity Name	ram
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	1056
Total pins	72



Flow Status	Successful - Sat Nov 25 11:20:16 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	bus
Top-level Entity Name	bus
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	2
Total pins	227

5. 고찰 및 결론

A. 고찰

bus에서 아비터 로직을 구현 할 때 예외 처리 과정에서 문제가 생겼었다. 이전 값에 따라서 선택되는 grant값이 달라지는데 이전 값을 어떻게 입력받을지가 문제였다. 해당 문제를 해결하기 위해 아비터에 레지스터를 생성해 이전 grant값을 가지고 있도록 하였다. 이번 실험에서는 grant의 종류가 2개 이기 때문에 하나가 0인경우 다른 하나는 1로 자동으로 변화하기 때문에 하나의 grant를 이용한 예외 처리만 하면 문제 해결이 가능했다.
또한 각 멀티플라이어에서 값을 선택하는 부분에서 아비터에서 나오는 데이터를 이용하는 것으로 보았는데 아비터에서 2bit데이터를 어떻게 1bit로 넣을지 고민했다. M1 m0 grant중 m1을 선택함으로서 해결 할 수 있었는데 m1의 경우 req0가 1이면 0, req 1이 1이면 1로 변화하기 때문에 멀티플렉서에서 0값과 1갑중 선택할 때 이용 할 수 있었다.

B. 결론

이번 실험에서는 ram과 bus를 구현해 보았다. 램의 경우 여러 개의 데이터 셋을 만들어 놓고 해당 데이터에 값을 저장하고 불러오는 것이 c++의 배열의 형태와 유사하다는 생각이 들었다. Bus의 경우 마스터를 통해 여러 데이터 중에 하나를 선택 가능했다. 하지만 의문이 들었던 점은 여러 개여 요소에서 선택 가능했는데 wr의 경우 따로 값을 저장하는 것이 아닌데 이번 실험에서는 필요하지 않겠다는 생각이 들었다. 만약 데이터를 어딘가에 저장하고 해당 데이터를 읽을지 데이터에 값을 저장할지 정할 수 있는 로직에서 S_wr 요소가 필요할 것 같다.

6. 참고문헌

이준환/디지털논리회로/광운대학교/2023

이형근/컴퓨터공학기초실험/광운대학교/2023

렘/ https://namu.wiki/w/RAM

버스/https://velog.io/@fldfls/%EC%8B%9C%EC%8A%A4%ED%85%9C-

%EB%B2%84%EC%8A%A4-System-Bus

램