컴퓨터 공학 기초 실험2 보고서

실험제목: Simple Memory & Bus

실험일자: 2022년 11월 22일 (화)

제출일자: 2022년 11월 22일 (화)

학 과: 컴퓨터정보공학부

담당교수: 공영호 교수님

실습분반: 화요일 0,1,2

학 번: 2019202050

성 명: 이강현

1. 제목 및 목적
   A. 제목

      Memory & Bus

   B. 목적

      Memory와 Bus의 개념에 대해 이해하고 이를 verilog로 구현하여 본다. ram에서 ram이 어떻게 데이터를 저장하는지 알고 많은 요청자와 많은 데이터 사이에서 bus가 어떻게 데이터를 주고 받는지에 대해 이해하고 구현하여 본다.

2. 원리(배경지식)

   <Memory, RAM>

   컴퓨터에서 메모리란 기억장치의 역할을 하며 데이터를 저장하고 읽는 작업을 처리한다. 메모리는 컴퓨터의 전원이 꺼지면 데이터가 지워지는 휘발성이라는 특징을 가진다. 램은 컴퓨터의 주 기억장치, 데이터의 일시적인 저장등에 사용된다.
   메모리는 가격과 용량 속도와 같은 범주로 나누면 계층구조를 가지게 되는데 이는 피라미드형태를 띄며 위에서부터 레지스터,캐시,주기억장치,버퍼,보조기억장치로 나눌 수 있다. Ram은 임의 접근 메모리로 임의의 data와 address를 가진다면 이를 통해 순차적이지 않더라도 임의적으로 접근할 수 있는 기억장치를 말한다. 따라서 읽기 쓰기 속도가 빠르다는 장점이 있다.

   <Bus>

   Bus란 여러 구성요소간 data를 전송할 수 있도록 사이에서 연결을 맡는 구조이다. Master와 Slave가 존재하며 Master가 request를 보내면 이를 Arbiter에서 grant신호로써 회답을 하며 이러한 신호를 받게되면 해당 Master의 address와 data가 메모리에 접근할 수 있는 권한을 갖게 된다. 이를 mux를 통해 현재 접근하려는 master가 누군지 그리고 어떤 slave에 접근하려고 하는지 접근하여 어떤 작업을 하려는지등이 신호로서 알 수 있으며 master와 slave를 서로 꼬이지않게 1대1로 연결해주는 시스템이 바로 Bus이다.
   Bus의 장점으로는 새로운 component를 추가하기가 쉽다는 것이다.

3. 설계 세부사항

<RAM>

Direction	Port name	Bits	Description
Input	clk	1bit	Clock
	cen	1bit	Chip enable
	wen	1bit	Write enable
	addr[4:0]	5bit	Address
	din[31:0]	32bit	Data in
Output	dout[31:0]	32bit	Data out

Input 값으로 cen과 wen을 받고 데이터를 읽을지 쓸지를 결정하고 해당 작업에 맞게 addr과 din을 사용한다. cen은 RAM을 사용할지 말지를 결정하고 wen은 RAM이 write할지 read할지를 결정한다. read라면 dout에 addr에 저장된 메모리의 데이터를 보여주고 write라면 메모리의 addr부분에 din을 저장하는 작업을 수행한다.

<BUS>

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_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

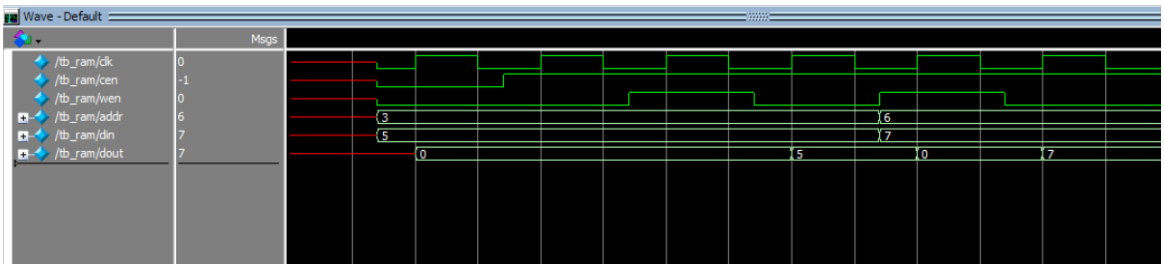
	s_din [31:0]	Slave data input

위와 같은 입출력 변수를 사용하였고 bus에서 어떤 master의 요청을 받을지 결정하는 Aribiter와 요청받은 master의 데이터만을 걸러내기 위한 mux, 그리고 그 데이터중 어떤 slave를 사용할 것인지에 대한 정보를 가진 address값을 통해 slave에 접근하고 수정하는 작업을 하게 끔 구현하였다. Master의 req에 따라 state로 어떤곳에 권한을 부여할지를 구현하였고 이를 grant변수로 명시하였다. 이렇게 grant를 가진 master의 address값에 따라 출력값이 결정되고 slave또한 결정된다. 그 후 어떤 slave가 선택되었는지 알 수 있는 s_sel신호들에 따라 mux를 통해 slave의 데이터를 선택하여 가져올 수 있다.

4. 설계 검증 및 실험 결과

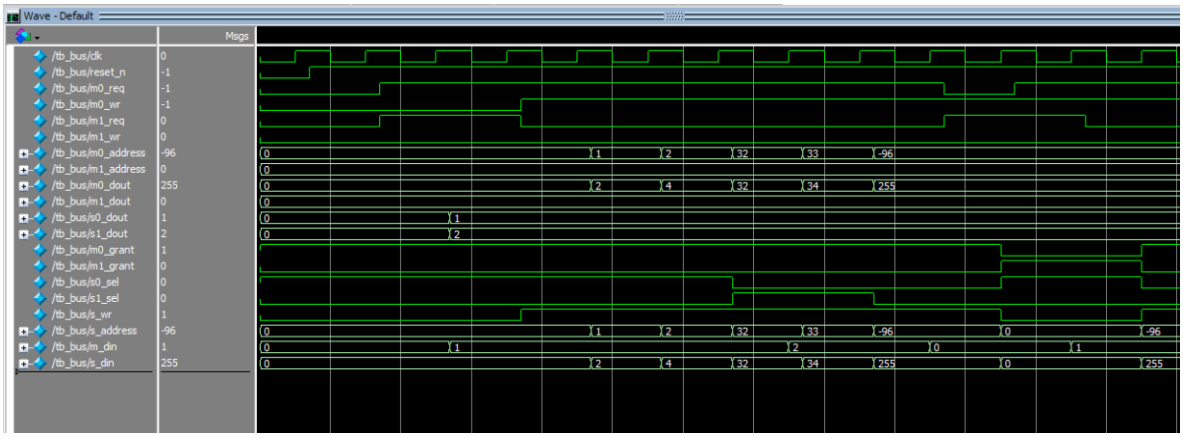
A. 시뮬레이션 결과

<ram>(top module)



Ram의 시뮬레이션 결과이다. 주소값을 3, 데이터를 5를 주었을 때 cen이 0이기 때문에 아무런 값이 dout에 나타나지 않는다. cen이 1이 되었을 때 3의 주소값에는 아무런 값도 write되어있지 않은 초기상태이므로 동일하게 0이 나타난다. wen또한 1로 바꾸어 write해준 후 다시 wen을 0으로 바꾸어 read를 수행하면 5라는 값이 나타나는 것을 확인할 수 있다. 두번째로 6이라는 주소에 7이라는 값을 넣기 위해 입력값을 수정하고 write를 실행한 결과 클록의 상승부분에서 write를 수행중이기 때문에 dout이 0이 나오고 다시 read를 수행중일때는 write했던 7이 나오는 것을 확인할 수 있다.

<bus>(top module)

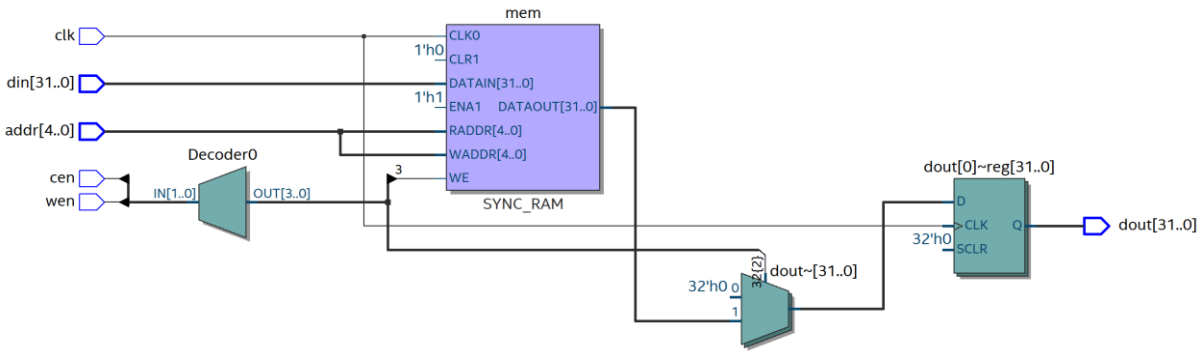


처음에 m0와 m1의 req가 모두 0이므로 기본적으로 m0가 선택되어 grant가 1인 것을 통해 알 수 있고 s0_dout과 s1_dout을 입력값으로 준 결과 m0의 주소값이 초기에 0이므로 s0가 선택되어야 하기 때문에 s0_sel값이 1이고 그에 맞게 s0_dout값이 m_din으로 나오는 것을 확인할 수 있다. 그 이후도 grant는 현재 state와 입력값을 고려하여 master들에게 주어지고 slave는 master가 결정된 후 master의 address에 따라 그 범위에 맞게 선택되어 값이 반영됨을 알 수 있다. m_din은 처음에는 s0_dout과 s1_dout에 따라 바로 바뀌어 클록의 영향이 없는 것처럼 보여지지만 이전 클록의 상승에서 이미 mux에 selecter로서의 값이 기다리고 있기 때문에 dout들이 반영되자마자 값 값 나온 것을 알 수 있다. 그 이유는 m_din의 값이 2로 바뀌는 부분에 dout 값들인 1과 2는 이미 입력되어있으나 2로 변경되는 부분은 s1_sel가 1이 되고 클록의 상승에서 변경됨을 알 수 있기 때문이다. Arbiter의 결과값이 mux에 연결된 부분의 값 s_wr,s_address,s_din의 값 또한 마찬가지로 arbiter만이 clk의 영향을 받기 때문에 clk의 영향을 받지 않는 입력값들은 바뀌면 바로 출력에 반영되는 모습을 보인다. 마지막으로는 state가 어떻게 바뀌는지 보여지고 주소값이 모든 slave들의 범위를 벗어났을 경우 어떤것도 sel하지 않는 모습을 보여준다.

B. 합성(synthesis) 결과

<ram>(top module)

<RTL viewer>



Memory가 크게 형성된 것을 확인 할 수 있고 cen과 wen을 통해 메모리의 접근 권한을 설정하는 것을 그림을 통해 확인할 수 있다.

<flow summary>

Flow Summary

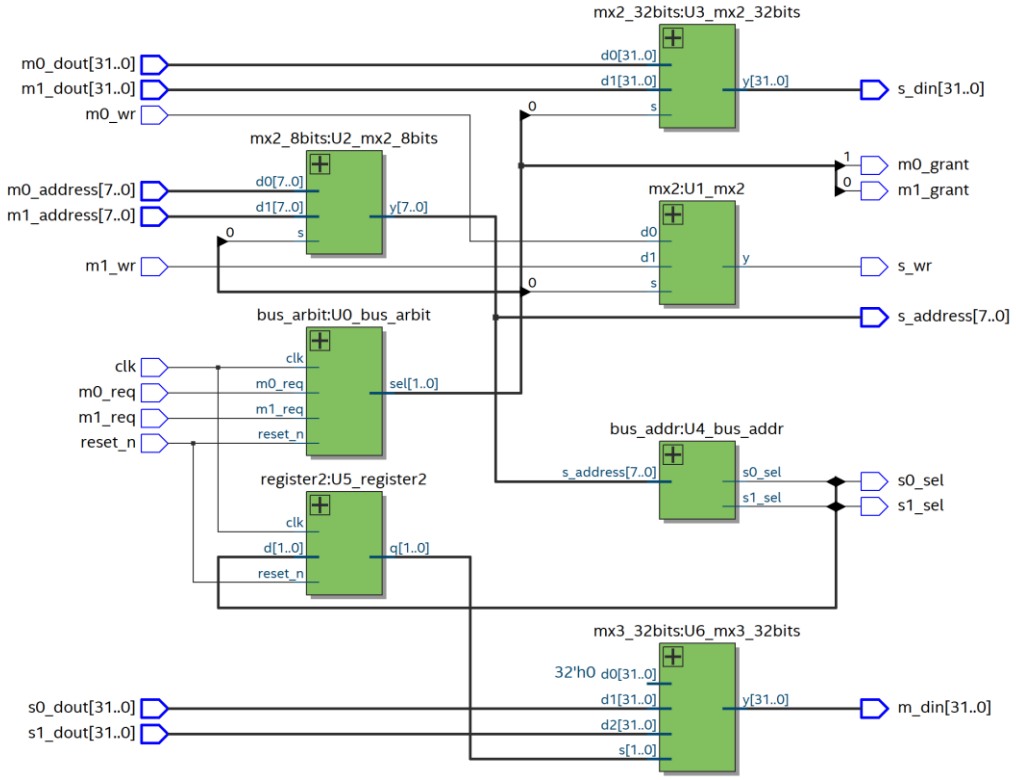
<<Filter>>

Flow Status	Successful - Tue Nov 22 18:55:32 2022
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
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

컴파일이 성공적으로 완료되었음을 확인한다.

<bus>(top module)

<RTL viewer>



Bus의 RTL viewer이다. 모든 모듈들이 잘 연결된 것을 확인할 수 있다.

<flow summary>

Flow Summary

<<Filter>>

Flow Status	Successful - Tue Nov 22 17:15:06 2022
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	3
Total pins	227
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

컴파일이 성공적으로 잘 되었다.

5. 고찰 및 결론

A. 고찰

Ram은 단순히 주소값으로 접근가능한 memory였기 때문에 모듈하나로 쉽게 구현할 수 있었다. enable신호에 맞게 read,write작업을 수행하면 되었기 때문이다. 또한 메모리를 초기화하는 단계에서 c언어와 동일하게 for문을 사용하여 반복문을 수행할 수 있다는 것을 배웠다. Bus는 모듈들이 많았고 개념적인 것을 이해하는데 시간이 많이 걸렸다. Mux를 상황에 맞게 코드를 수정하여 사용하여야했고 특히 모두 다 구현을 하고 나서 testbench를 돌리던중 클록과 동기화되지 않아서 문제점을 찾고 있었는데 입력값들은 클록의 영향을 받지 않는다는 점을 알고 문제가 없음을 확인하였다. Slave의 output이 master의 input이고 master의 output이 slave의 input으로 작용한다는 점 또한 헷갈렸던 것 같다.

B. 결론

메모리 배열을 사용해보고 for문또한 사용해보면서 새로운 코드적인 개념을 배웠다. State 사용하여 접근 권한을 주는 우선순위를 구현해보면서 fsm의 또다른 사용법을 학습한 것 같다. 연결구조로 사용되는 bus의 구조와 역할을 실습을 통해 더 잘 이해하게 된 것 같다.

6. 참고문헌

공영호 교수님/컴퓨터공학기초실험2/2022

기억장치/https://namu.wiki/w/%EA%B8%B0%EC%96%B5%EC%9E%A5%EC%B9%98