컴퓨터 공학 기초 실험2 보고서

실험제목: Memory & Bus

실험일자: 2020년 11월 09일 (월)

제출일자: 2020년 11월 16일 (월)

학 과: 컴퓨터공학과

담당교수: 공진흥 교수님

실습분반: 월요일 0, 1, 2

학 번: 2019202052

성 명: 김 호 성

1. 제목 및 목적

A. 제목

Memory & Bus

B. 목적

Address에 기반하여 데이터를 저장하는 hardware인 memory를 ram 형식으로 구현해본다. 또 여러 component들 간에 데이터를 전송할 수 있도록 해주는 하드웨어인 bus를 구현해본다

2. 원리(배경지식)

-ROM

한번 기록한 데이터를 빠른 속도로 읽을 수 있지만, 다시 기록할 수 없는 메모리를 말한다. 그래서 Read Only Memory의 앞 글자를 따서 ROM이라 부른다. 컴퓨터의 전원이 꺼져도 데이터가 남아있기 때문에 비휘발성 메모리이다. 데이터가 남아있을 수 있는 이유는 정보가 입력되면 ROM 내부 소자의 구조로 저장하기 때문이다. 그래서 ROM은 시스템 초기화 프로그램이나 진단 프로그램, 자주 사용되는 데이터나 서브루틴에 주로 사용된다. ROM의 종류는 다음과 같다.

>Mask ROM

ROM의 제작사마다 고유한 데이터에 맞게 회로를 제작한 ROM이다. 어떠한 방법으로도 데이터를 수정할 수 없다. 흔히 부르는 CD-ROM이 이것이다.

>PROM(Programmable ROM)

PROM writer를 사용해서 사용자가 필요한 기능을 단 한 번 입력할 수 있는 ROM이다. 기록되고 나면 다시 수정할 수 없다.

>EPROM(Erasable ROM)

데이터를 수정하고 다시 입력할 수 있는 ROM이다. 데이터의 삭제는 ROM에 자외선을 20분 정도 쐬어주면 삭제된다. 횟수에 제한은 없다.

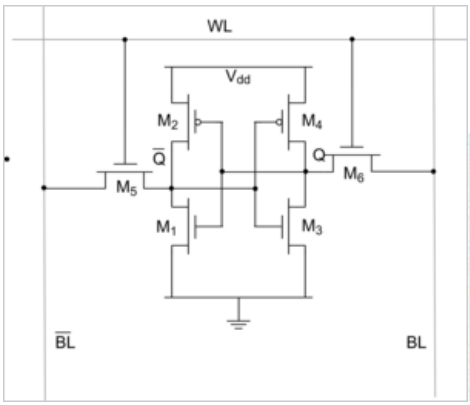
>EEPROM(Electronic Erasable ROM)

자외선 대신 전기를 통해 데이터를 수정하고 다시 입력할 수 있는 ROM이다. 데이터 삭제 및 입력이 EPROM에 비해 느리고 수만 번 정도의 횟수 제한이 있다.

-SRAM

Static Random Access Memory의 약자로 정적 ram이라고 한다. MOSFET 4개에서 6개로 구성된 플립플롭 방식의 기억소자를 사용하고 전원 공급이 계속되는 한 저장된 내

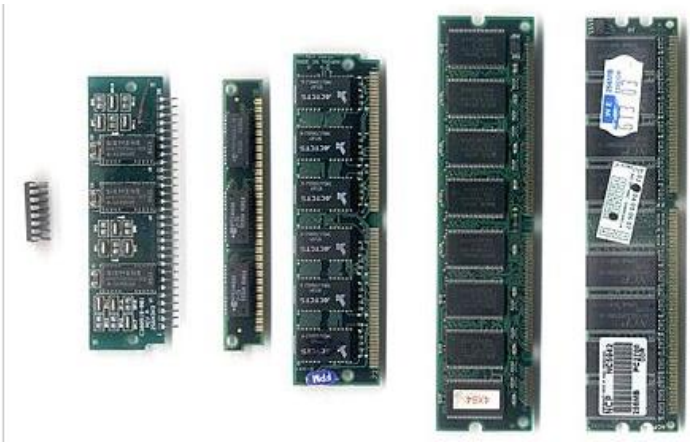
용을 기억한다. 복잡한 clk이 필요하지 않아서 소용량의 메모리나 캐시메모리에 주로 사용한다. DRAM보다 5배정도 빠르지만 가격도 비싸다. 외관 및 내부 구조는 아래와 같다.



왼쪽의 회로도에서 M은 트랜지스터를 뜻하며 6개의 트랜지스터로 구성된 CMOS SRAM 소자이다. SRAM의 종류로는 가장 흔한 CMOS SRAM, 속도가 매우 빠르지만 전력소모가 심한 양극 SRAM, 동기식 SRAM과 비동기식 SRAM 등이 있다.

-DRAM

Dynamic Random Access Memory의 약자로 동적 ram이라고 한다. 비트가 입력되면 집적회로 안에 분리된 축전기에 각각 다른 전하량이 들어오게 되고 이 전하량을 기반으로 데이터를 저장한다. 축전기는 시간이 지나면 방전되기 때문에 데이터를 잃어버리게 되는데 이를 방지하기 위해 기억장치를 재생(refresh)해야 한다. 다수의 트랜지스터가 필요한 SRAM에 비해 DRAM은 한 개의 트랜지스터와 한 개의 축전기로 구성되어 있기 때문에 가격이 싸다. 즉 전력 소모가 적고 가격이 싸서 대용량 기억장치에 많이 사용된다. 아래 그림은 DRAM의 외관이다.



3. 설계 세부사항

- Ram

```
integer i;

initial begin
   for(i = 0; i < 32; i = (i + 1))
      mem[i] <= 32'h0;
end
```

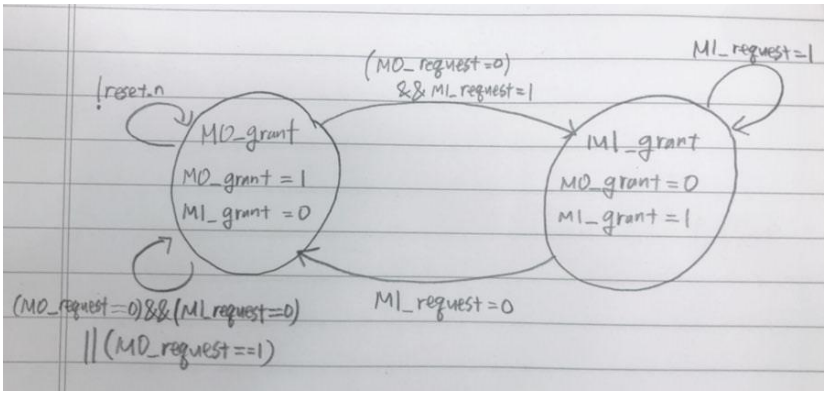
위 코드는 각 address의 주소를 초기화한 코드이다. 변수 i를 선언한 후 for문을 사용해서 총 32개의 32비트 메모리를 0으로 초기화했다. 메모리의 범위가 0부터 31까지 이므로 address는 5비트 2진수 값을 가진다. 메모리에 값을 입력하는 기능은 cen과 wen의 변화에 따른 case문을 사용했다. Cen과 wen이 모두 1이면 address가 가리키는 메모리에 din을 입력하고 dout은 0을 출력한다. cen이 1이고 wen이 0이면 address가 가리키는 메모리의 값을 dout에 입력한다. cen이 0이면 dout은 0이 된다. Cen과 wen이 디폴트 값이면 dout은 x를 출력한다. 데이터를 입력하는 din과 저장하는 메모리, 출력하는 dout이 있으므로 이 메모리는 ram과 기능이 같다고 할 수 있다.

- Bus

2개의 master와 2개의 slave 중 각각 하나씩 선택해 데이터를 원하는 방향으로 이동시켜 저장하거나 출력하는 기능이다.

>> arbiter의 역할

먼저 bus_arbit에서 어떤 master를 사용할지 결정한다. M0_req가 1이고 M1_req가 0이면 마스터0을, M0_req가 0이고 M1_req가 1이면 마스터1을 사용하도록 설계했다. 두 개의 req 신호가 모두 0이거나 모두 1이면 마스터0을 사용하도록 설계했다. 아래의 그림은 bus_arbit의 fsm 다이어그램이다.

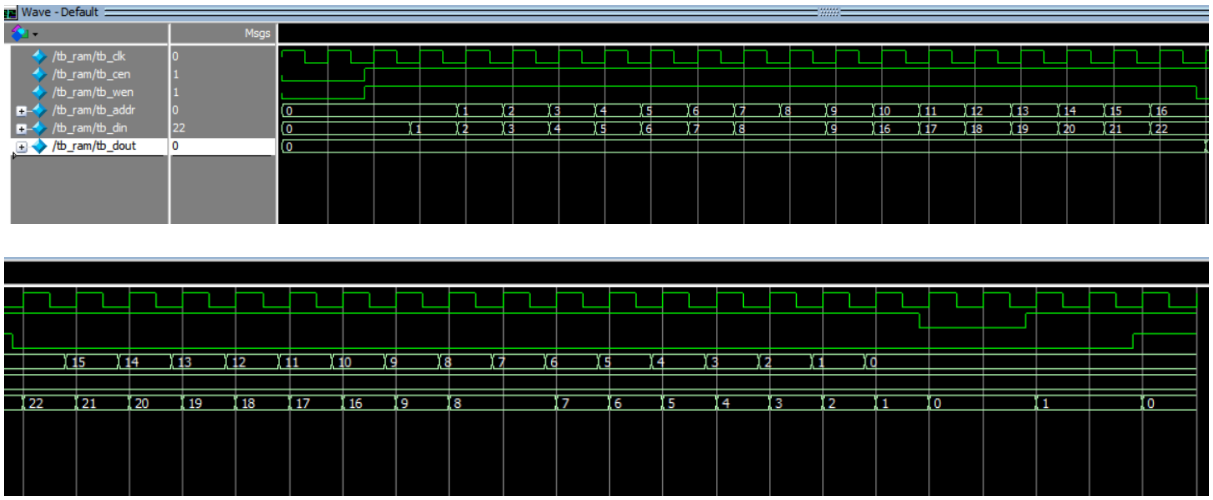


Bus_arbit에서 사용할 마스터를 정하고 나면 bus_addr에서 주소를 정해준다. 이 주소 값의 범위에 따라 slave0과 slave1 중 어떤걸 사용할지 결정하게 되는데, 주소 값의 MSB 3bit를 비교해서 000이면 slave 0을, 0011이면 slave 1을 사용하도록 설계했다. 각각이 32개의 주소를 가지고 있는 셈이다. 주소 값이 40을 넘어가면 어느 slave도 사용하지 않도록 했다.

4. 설계 검증 및 실험 결과

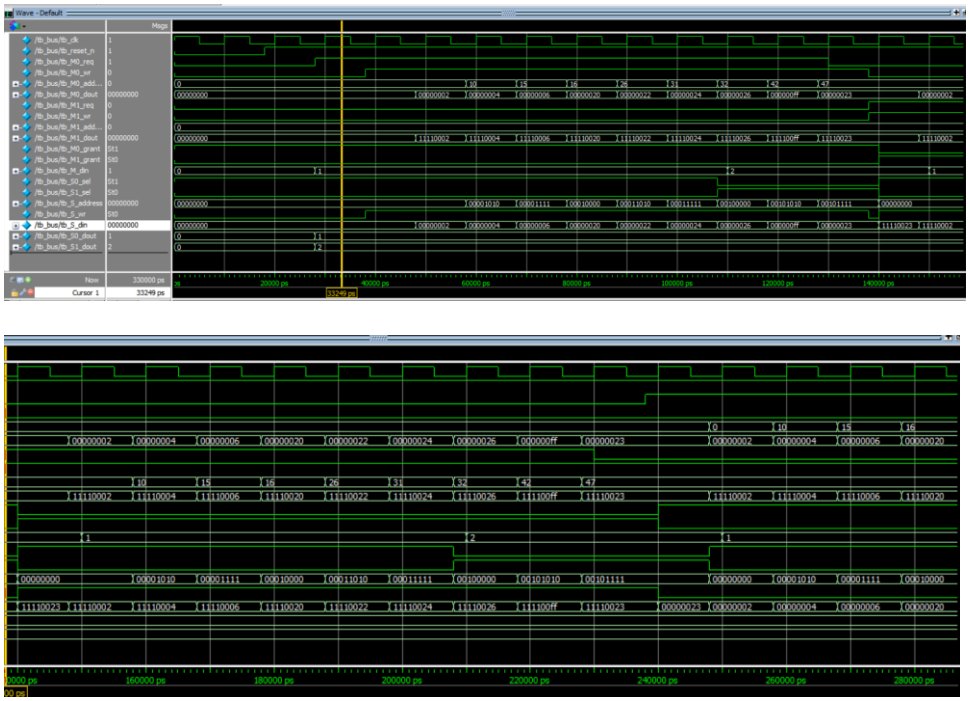
A. 시뮬레이션 결과

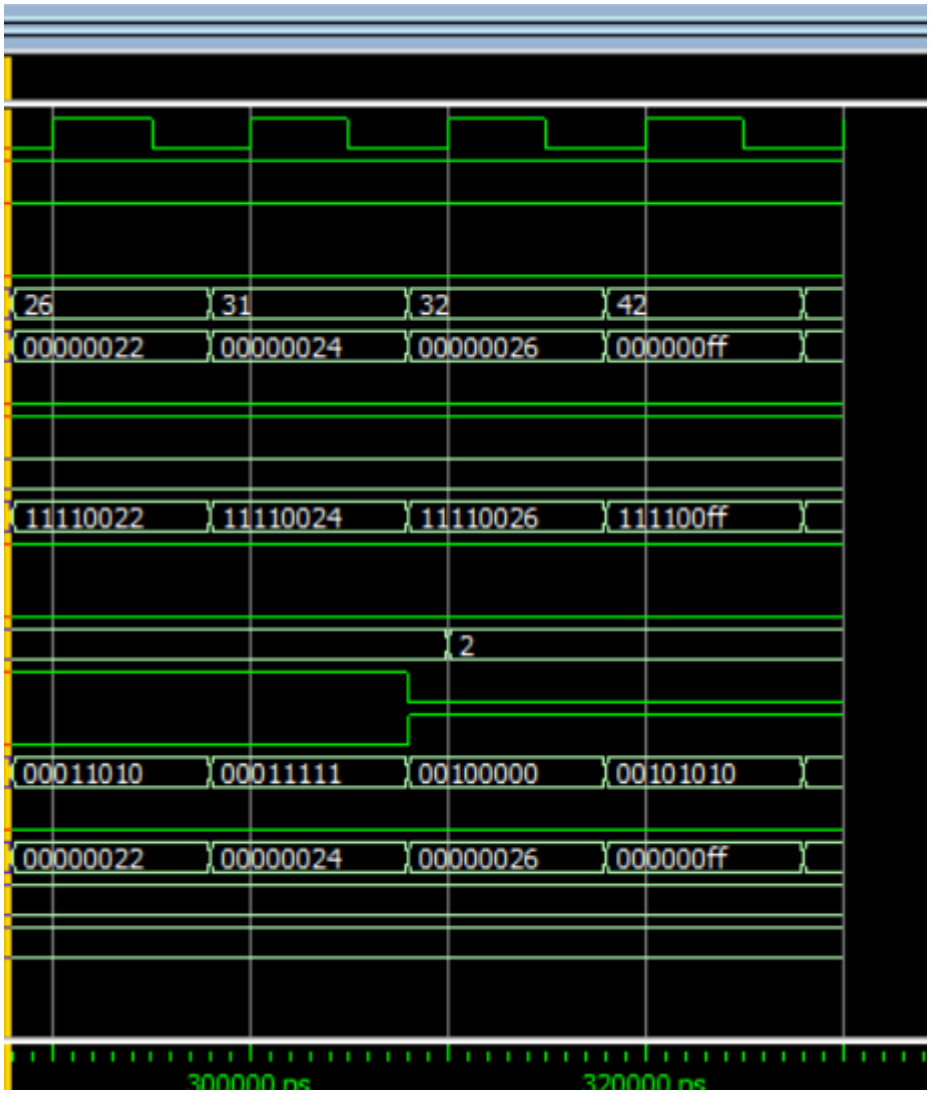
- Ram



각 주소 별 메모리 값은 0으로 초기화 되어 있으며, address 0 ~ 16번까지 임의의 값을 넣어준 후 wen의 값을 0으로 바꾼 후 address의 16번에서 0번까지 값을 출력했다. 이후 cen 값을 0으로 만들어 0을 출력해준 후 다시 cen 값을 1로 바꾸어 dout에 1을 출력하였다. 이후 wen 값을 1로 바꾸어 0x22값을 입력하고 dout을 0으로 출력하였다.

- Bus

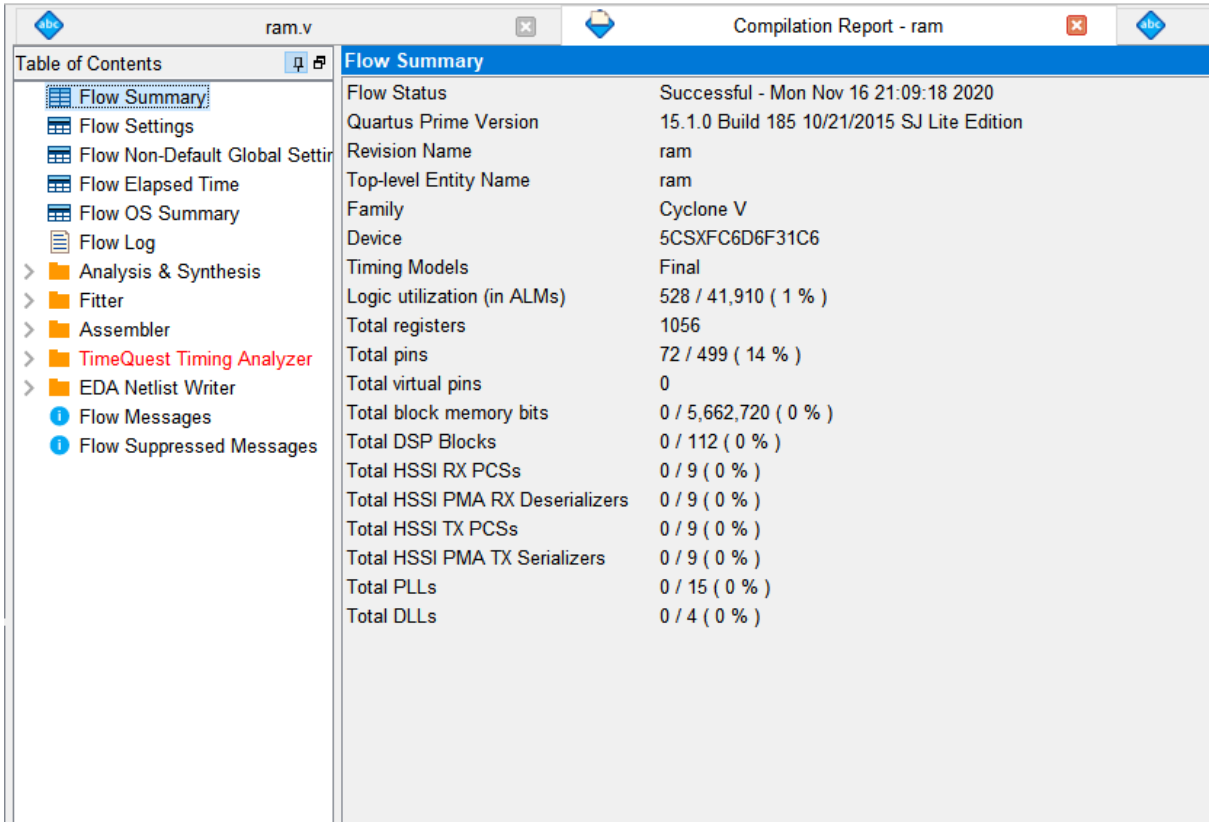


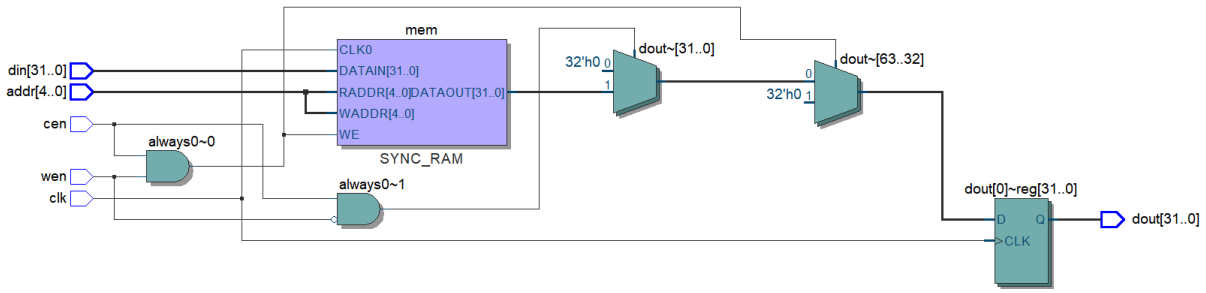


Master 0부터 활성화를 시킨 후 address값에 따라 값을 넣어주었다. 주소 값이 001....일 경우 에는 Salve 0을 쓰기 때문에 S0_dout 값이 M_din에 출력되고 주소 값이 01.....일 경우에는 S1_dout 값이 M_din에 출력된다. Test bench의 경우 Decimal 값으로 나타나져 있기 때문에 32일 때부터 M_din 값이 바뀌는 걸 확인할 수 있다. 이후 Master 1을 활성화 시켜 Master 0과 동일하게 값을 넣어주었고, M_din이 정상적으로 작동하는 걸 확인할 수 있다. 이후 Master 1을 사용하고 나서 M0_req 값을 1로 바꾸어 다시 Master 0에서 작동하는 것을 확인하였다.

B. 합성(synthesis) 결과

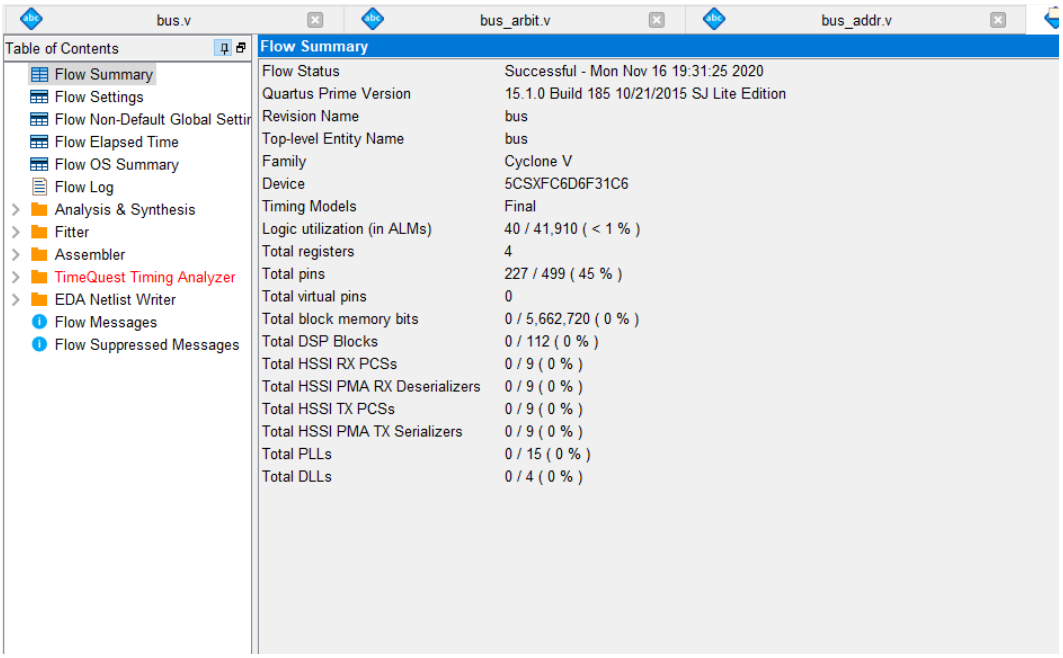
- Ram

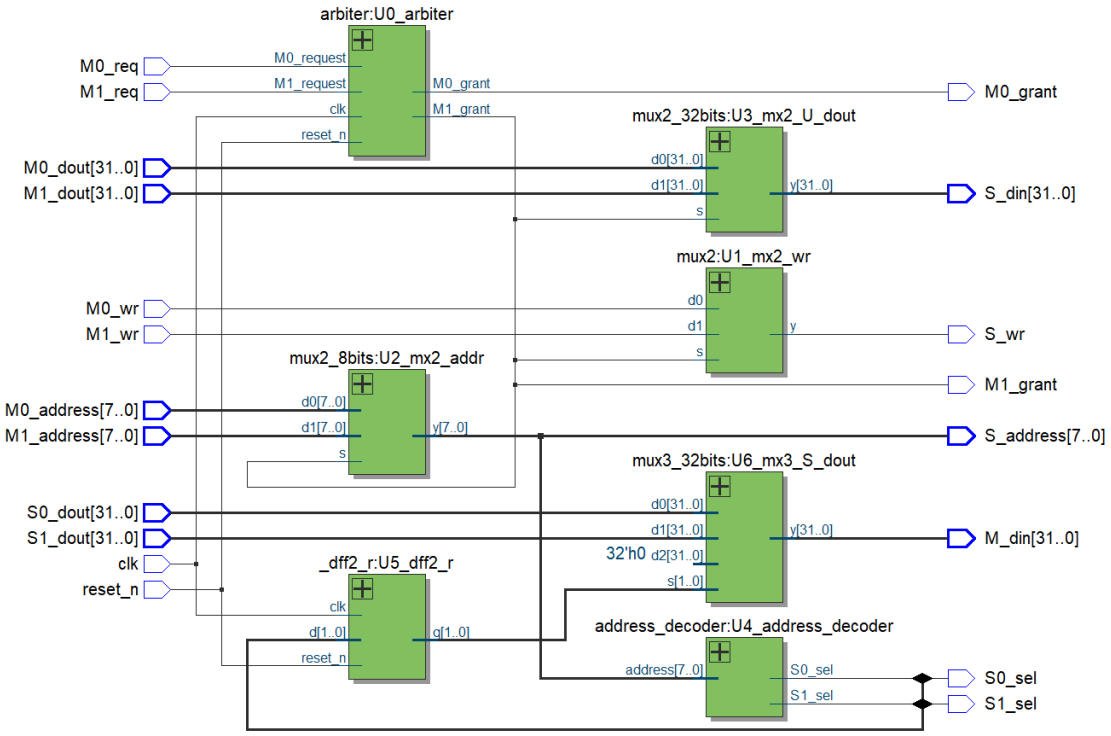




메모리를 초기화한 후, 원하는 메모리에 주소별로 값들을 저장할 수 있다. Cen과 wen 신호에 따라 어떤 주소에 어떤 값을 입력 또는 출력할지 정한 후 결과 값이 저장된다.

- Bus





합성 결과 및 RTL viewer는 위와 같다. BUS_arbit중 3input mux에서 입력 값 중 하나가 32'h00인 이유는 reset_n 기능을 하기 위함이며, 이 mux 이전에 붙어있는 레지스터를 이용해서 synchronous reset을 할 수 있기 때문이다. 이 레지스터로 인해 reset_n 신호를 받으면 M_din 값을 0으로 초기화할 수 있다.

5. 고찰 및 결론

A. 고찰

Ram의 경우 간단하게 짤 수 있었으며, data를 입력할지 출력할지 정하기에 용이하다고 느꼈다.

Bus에서는 bus_arbit fsm을 짜느라 시간이 조금 걸렸다. Always를 사용한 case 문을 여러 개 쓰는 방법이 아직 손에 제대로 익지 않아서 그런 것 같다. 또, 강의자료실의 decoder 코드를 보면, 중간에 중략이라는 의미를 가지는 ...이 들어가 있어 코드를 맞게 작성했지만, 내가 놓친 부분이 있을 거라 생각하여 시간을 허비했다. 때로는 자신의 코드에 자신감을 가지는 것도 좋을 것 같다.

B. 결론

메모리(ram)는 해당 주소에 값을 입력 및 출력할 수 있다. 단 데이터를 입력하기 위해서는 선언할 때 초기화를 해줘야 하는데 이 때 for문을 사용하면 코드 사이즈를 줄일 수 있다. 그리고 버스는 원하는 master와 slave를 선택해서 데이터를 입력할 수 있다. 버스의 instance 중 bus_addr에서 범위에 따라 select 신호를 정할 때 blocking과 non-blocking 둘 다 사용해봤는데 결과에 차이는 없었다.

6. 참고문헌

-기억장치 종류

https://m.blog.naver.com/PostView.nhn?blogId=ljh0326s&logNo=220851633811&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F

-SRAM

https://terms.naver.com/entry.nhn?docId=1180951&ref=y&cid=40942&categoryId=32832

https://ko.wikipedia.org/wiki/%EC%A0%95%EC%A0%81_%EB%9E%A8

-DRAM

https://ko.wikipedia.org/wiki/%EB%8F%99%EC%A0%81_%EB%9E%A8

https://terms.naver.com/entry.nhn?docId=1169752&cid=40942&categoryId=32384