컴퓨터 공학 기초 실험2 보고서

실험제목: Ram & Bus

실험일자: 2021년 11월 15일 (월)

제출일자: 2021년 11월 19일 (금)

학 과: 컴퓨터정보공학부

담당교수: 공진흥 교수님

실습분반: 월요일 0, 1, 2

학 번: 2018202046

성 명: 이준휘

1. 제목 및 목적

A. 제목

Ram & bus

B. 목적

해당 수업을 통해 Ram에 대해 이해할 수 있다. verilog에서 반복문을 사용하여 이를 구현할 수 있다. bus를 이해하고 이를 바탕으로 프로세서의 모듈간이 어떻게 연결되어 있는지 이해한다.

2. 원리(배경지식)

i. ram

ram이란 Address를 기반으로 하여 데이터를 저장하는 hardware를 말한다.

해당 ram에서는 Chip이 값을 읽거나 쓰도록 하는 신호인 cen 신호와 쓰거나 읽기를 구분하는 신호인 wen신호를 갖는다. 이 때 cen 신호가 0이면 해당 회로는 32bits output을 0으로 출력하게 된다. cen이 1일 경우 wen이 1이면 쓰기, 0일 경우에는 해당 주소의 값 읽기가 작동된다.

ii. bus

Bus란 여러 모듈 간의 data를 주고받을 수 있도록 연결해주는 회로를 의미한다.

bus는 각 slave마다 가지는 주소의 범위를 나누며, 주소값에 따라 slave를 선택해서 해당 회로에 신호를 보낼 수 있다. 또한 master의 신호들 중 하나만을 선택하여 입력을 받을 수 있다.

bus에서 arbiter는 Master의 우선순위를 정하는 모듈이다. 부분에서는 이전에 보낸 master의 신호가 꺼지지 않는 한 현재 master의 출력을 유지하고, 만약 현재 신호가 없고 다른 신호가 존재할 경우 해당 신호로 grant를 바꾸는 회로다.

address decoder는 현재 사용하는 master의 address에서 주소의 위치의 값을 토대로 어떤 slave에 들어가는지 알려주는 회로다. 해당 값을 토대로 다시 값을 bus로 반환받을 때 해당 신호를 다시 사용하여 이를 mux로 걸러낸다.

bus는 component들을 추가하기 쉽고, 저렴하게 제작이 가능한 특징이 있다.

3. 설계 세부사항

i. ram

해당 모듈은 ram이 구현되는 모듈이다. 해당 모듈에서는 initial문과 for문을 사용하여 각 메모리의 주소에 들어가 해당 값을 0으로 초기화시켜준다. 그 후 always문을 사용하여 clk의 posedge마다 cen과 wen의 값을 판별하여 적절한 값을 출력하도록 조정한다.

ii. gate

해당 모듈은 gate들이 저장된 모듈로 이전에 제작되었던 모듈을 재사용하였다.

iii. mx2

해당 모듈은 2-to-1 MUX로 이전에 제작되었던 것을 재활용하였다.

iv. mx2_32bits

해당 모듈은 2-to-1 32-bits MUX로 이전에 제작되었던 것을 재활용하였다.

v. mx2_8bits

해당 모듈은 2-to-1 8-bits MUX로 위의 mx2_32bits 모듈을 변형시켜 제작하였다.

vi. mx3_32bits

해당 모듈은 3-to-1 32-bits MUX로 위의 mx2_32bits 모듈을 변형시켜 제작하였다.

vii. register_r

해당 모듈은 1개의 register로 이전에 제작한 모듈을 다시 사용하였다..

viii. arbiter

해당 모듈은 두 개의 request 중 하나의 신호를 선택하여 grant를 보내주는 회로다. 해당 모듈에서는 2개의 req신호와 이전 grant를 인자로 입력받는다. 그리고 always문을 통해 위의 원리에서 설명한 상태에 따라 grant1을 끄거나 킨다.

ix. address_decoder

해당 모듈은 주소를 확인하여 sel신호를 알맞게 출력하는 모듈이다. 해당 모듈에서는 앞쪽 3비트의 수를 확인한다. 해당 값이 000일 경우 s0_sel을, 해당 값이

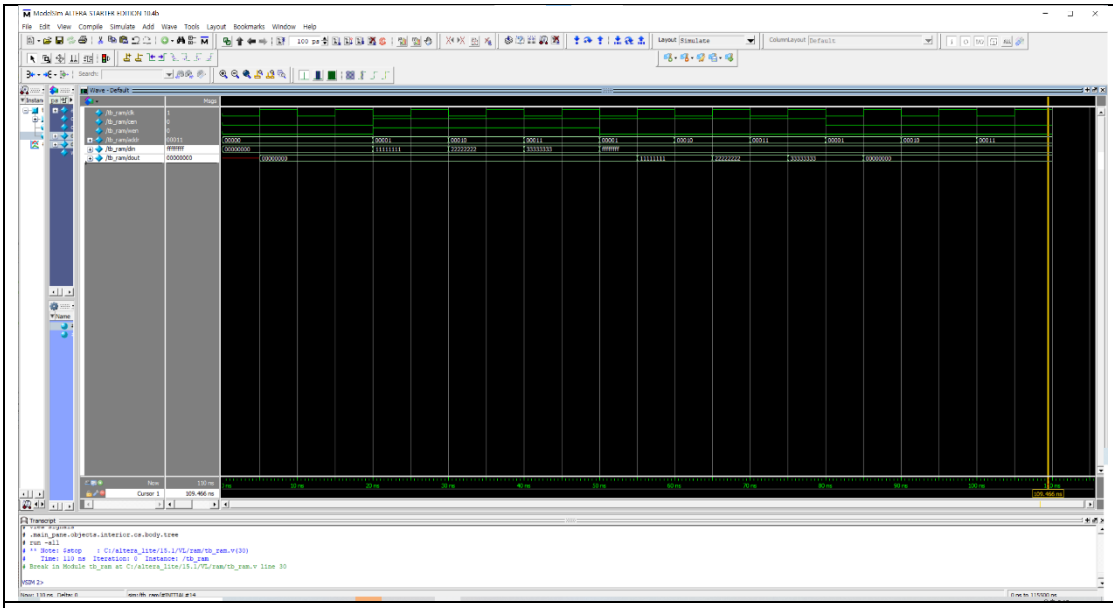
001일 경우 s1_sel을 1로 한다.

x. bus

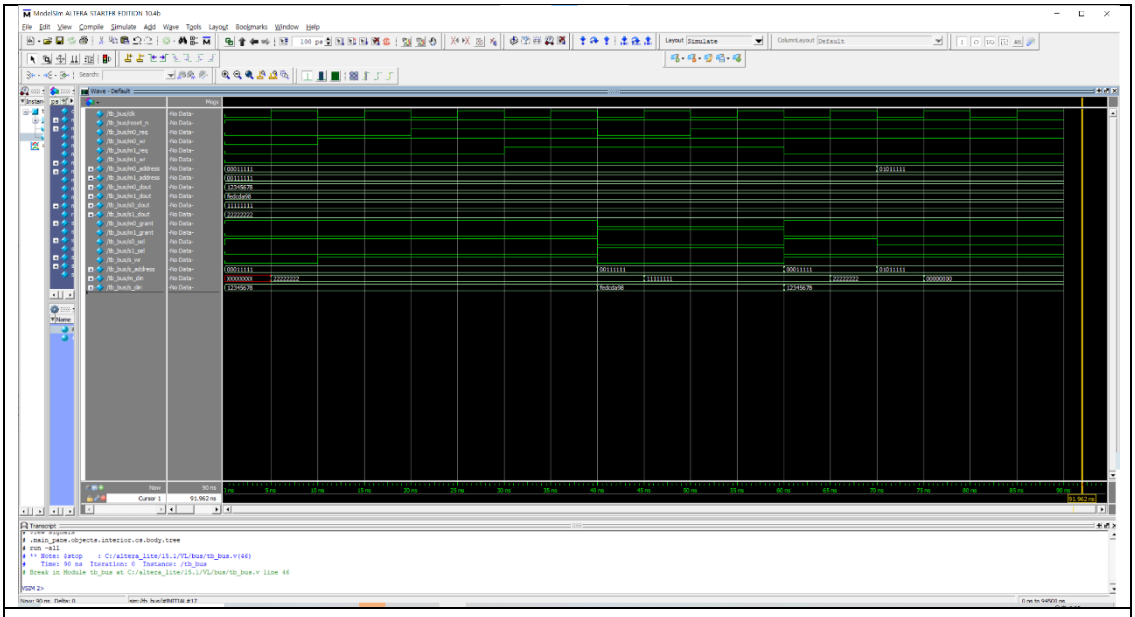
해당 모듈은 bus를 최종적으로 구현하는 모듈이다. 해당 모듈에서 이전의 만든 register_r을 통해 m1_grant의 상태를 기록하고 arbiter 모듈을 이용해 현재 사용할 master를 선택한다. 해당 신호를 mux들과 연결하여 신호를 최종적으로 선택한다. 선택된 신호 중 address의 경우 decoder로 이동하여 sel 신호로 바뀌고 해당 신호는 output을 1사이클 후에 받음으로 register에 연결한 후에 이를 mux3에 연결한다.

4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과



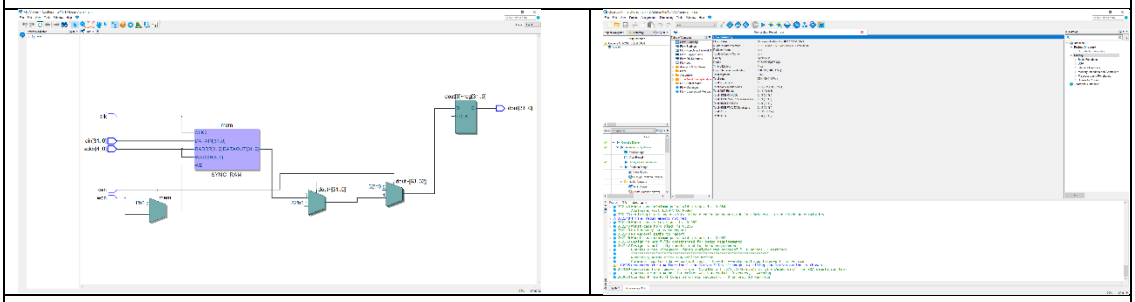
해당 testbench는 ram을 검증하는 testbench이다. 해당 결과에서 볼 수 있듯 쓰기와 읽기가 적절하게 출력되는 모습을 볼 수 있다.



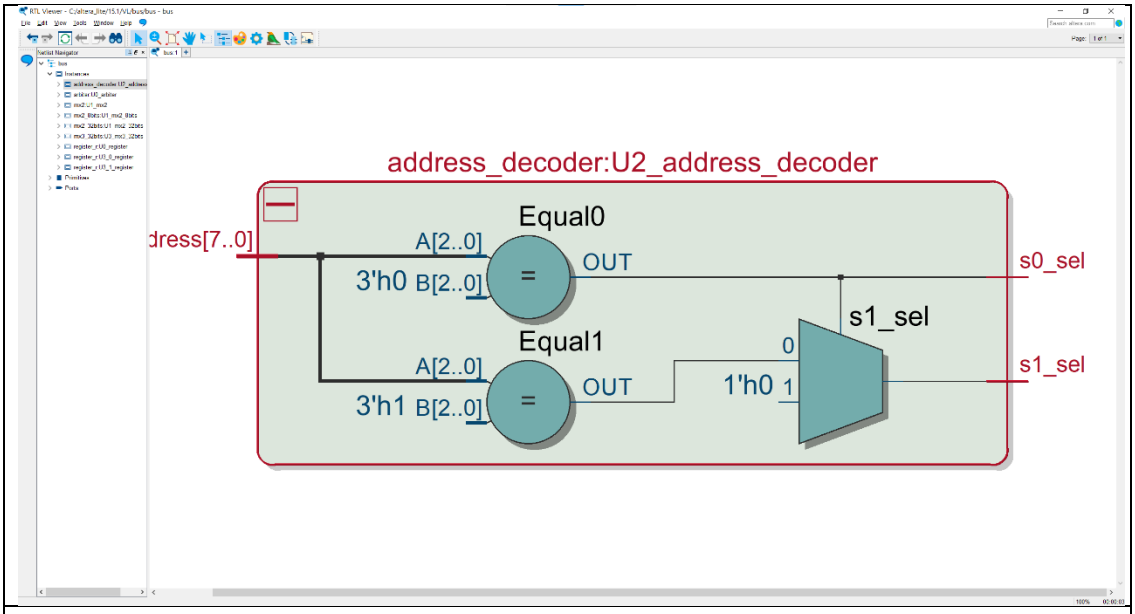
해당 testbench는 bus를 검증한다. 해당 결과에서 볼 수 있듯 grant 신호의 경우 req가 꺼질 때까지 해당 신호를 바꾸지 않는 모습을 볼 수 있고, address, wr, data 또한 grant에 따라 선택되는 것을 확인할 수 있다. 그리고 address의 주소에 따라 slave가 바뀌는 모습 또한 확인할 수 있고, 해당 신호는 1사이클 후에 들어오는 것을 확인할 수 있다.

B. 합성(synthesis) 결과

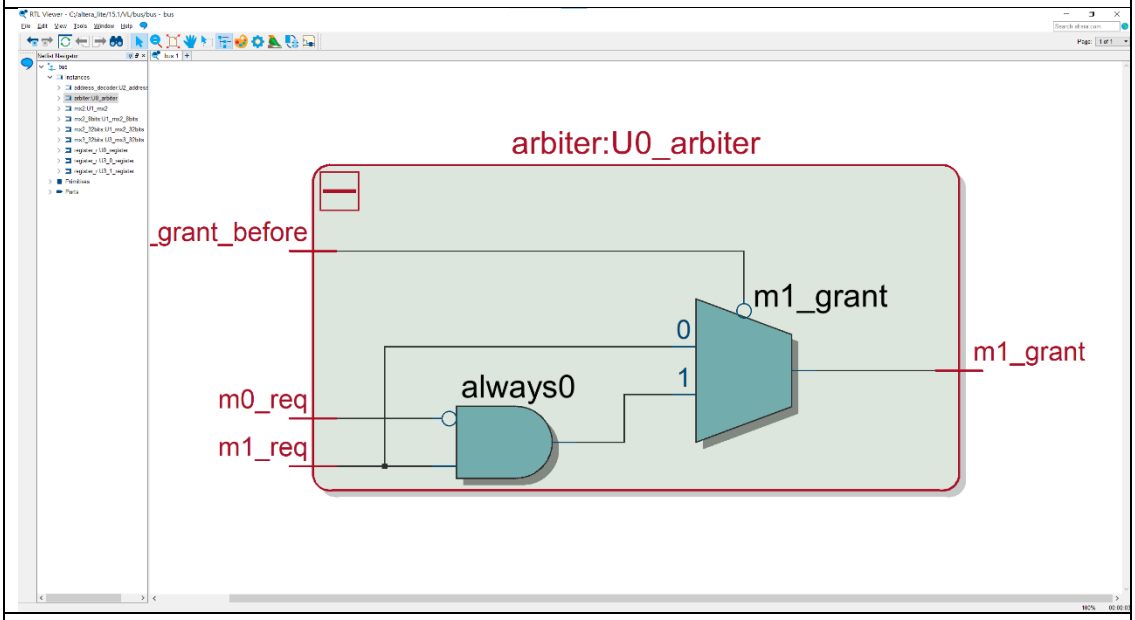
ram



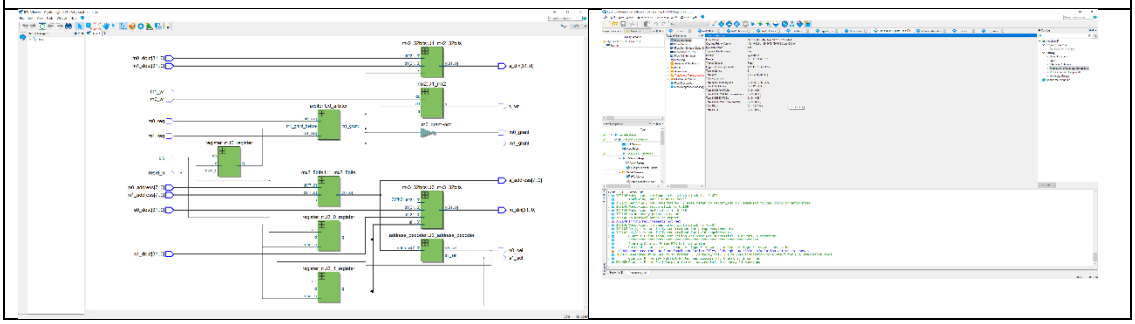
address_decoder



arbiter



bus



해당 모듈의 설계를 확인하면 각각의 모듈이 정상적으로 설계되어 연결되었음을 알 수 있다.

C. FPGA board targeting 결과

5. 고찰 및 결론

A. 고찰

이번 주차의 과제는 저번 multiplier에 비해 비교적 틀이 제공된 과제여서 구현하는데 많은 어려움은 없었다. 이전의 작업물을 활용하는 것이 많아 비교적 시간도 적게 들었다. 이번 과제에서 ram을 만들 때 for 문을 처음 사용해보면서 이렇게 초기화할 수도 있다는 것을 알게 되었다.

B. 결론

ram을 제작함으로써 반복문을 사용하여 해당 회로를 초기화할 수 있다. bus는 단순히 와이어가 아니라 master와 slave를 받아 이를 분배시키는 module이다. 각 slave에 고유의 주소 범위가 있다.

6. 참고문헌

이준환 교수님/디지털논리회로1/광운대학교(컴퓨터정보공학부)/2021
이준환 교수님/디지털논리회로2/광운대학교(컴퓨터정보공학부)/2021