컴퓨터 공학 기초 실험2 보고서

실험제목: Register File

실험일자: 2020년 10월 16일 (금)

제출일자: 2020년 10월 23일 (월)

학 과: 컴퓨터공학과

담당교수: 이준환 교수님

실습분반: 금요일 5,6,7

학 번: 2019202009

성 명: 서여지

1. 제목 및 목적

A. 제목

Register File

B. 목적

여러 개의 register에 접근하여 값을 읽거나 쓰는 동작을 동시에 처리하는 것이 가능한 register file에 대해 이해하고, verilog를 사용하여 구현한다. register file에 사용하는 register는 behavioral instrument한 d flip-flop을 이용하여 구현한다. 이번 실습에서 구현하는 register file은 32bit register를 8개까지 이용할 수 있으며, read와 write를 수행할 수 있다.

2. 원리(배경지식)

1) register file

register file은 여러 개의 register의 집합으로 이루어진다. 각 register에 read와 wirte를 수행하는 회로를 포함한다. register file의 설계에 따라 한 번에 여러 개의 정보를 쓰거나 읽을 수 있다. 각 register들은 모두 다른 주소 값을 갖고, 읽거나 쓰는 동작을 수행할 주소를 입력받아 해당 register의 값을 변경하거나 읽어온다. 이번 실험에서 구현하는 register는 하나의 read와 write가 가능하며 8개의 32bit register를 이용한다.

2) stack, queue에 대해 조사하시오

1. stack

stack은 그 단어의 뜻이 의미하는 바와 같이 데이터를 바닥에서부터 쌓아 올리는 것과 유사하게 처리할 수 있는 구조이다. stack에 데이터가 들어올 때 아래에서 위로 탑을 쌓아 올리듯이 저장되었다면, 반대로 stack에서 데이터를 출력할 때는 위에서 아래방향으로 데이터를 하나씩 출력하게 된다. 결국 첫 번째로 stack에 삽입된 정보는 마지막으로 stack에서 출력되게 된다. 반대로 마지막으로 삽입된 정보가 가장 처음으로 출력되는 구조가 stack구조이다. stack에 가장 처음 삽입되는 정보의 주소를 head라 하고, 마지막에 삽입된 정보를 top이라 한다.

2. queue

queue는 stack과 달리 아래에서 위로 쌓인 데이터가 출력될 때도 아래에서 위 방향으로 출력되는 구조이다. 첫 번째로 queue에 삽입된 정보는 첫 번째로 출력되고, 마지막으로 삽입된 정보는 마지막으로 출력된다. 이러한 구조를 First In, First Out을 줄인 FIFO 혹은 선입선출이라 부르기도 한다.

3. 설계 세부사항

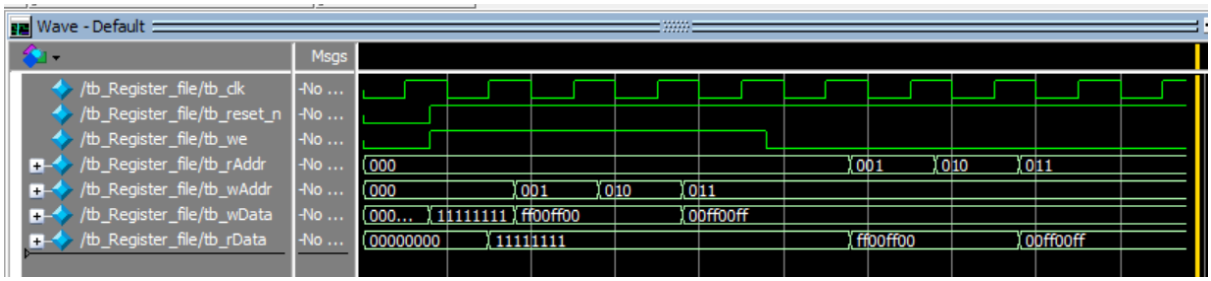
이번 실습에서 구현하는 Register File은 8개의 32bit register에 read, write기능을 할 수 있도록 한다. 입력은 wAddr, wData, we, rAddr, clk, reset_n으로, wAddr와 rAddr는 각각 쓰고 읽을 register의 주소값을 의미하고, wData는 wAddr에 해당하는 register에 저장할 정보이다. we는 write의 enable신호이다. register file은 clk에 동기화 되어있고, reset동작은 active-low이다. 출력은 read한 정보를 출력하는 rData 하나이다.

정보를 저장하는 register32_8 module과 write 동작을 하는 write_operation, read동작을 하는 read_operation module로 나누어서 설계한다. 8개의 register의 주소는 8가지 경우의 수를 나타낼 수 있는 3bit를 이용하여 binary encoding하면 000부터 111까지 주소가 생성된다. register에 write하는 과정은 wAddr로 입력된 주소를 decoder를 이용하여 해당 register를 찾고, wData의 정보를 register의 D 입력으로 전달한다. we이 1일 때만 write가 동작해야 하므로, en과 decoder의 결과를 and하여 register의 enable 입력으로 연결한다. register에서 값을 읽어오는 과정은 반대로 rAddr를 이용한 MUX로 구현한다. rAddr에 해당하는 register의 data만을 출력한다.

4. 설계 검증 및 실험 결과

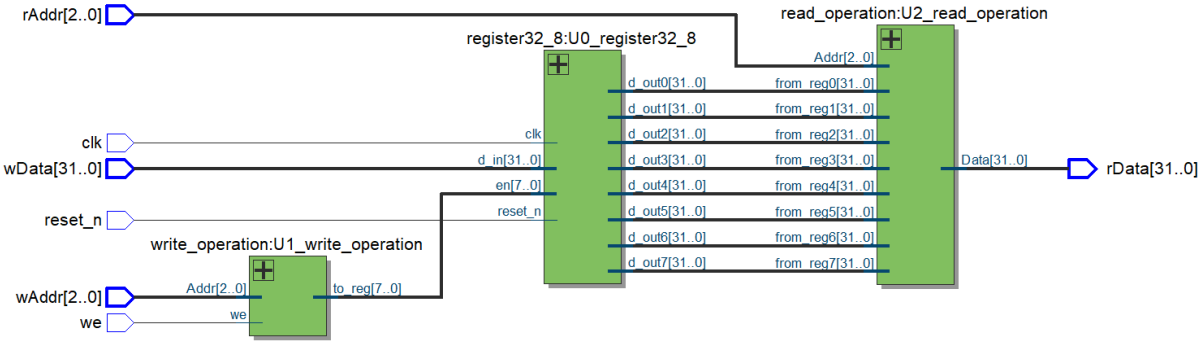
A. 시뮬레이션 결과

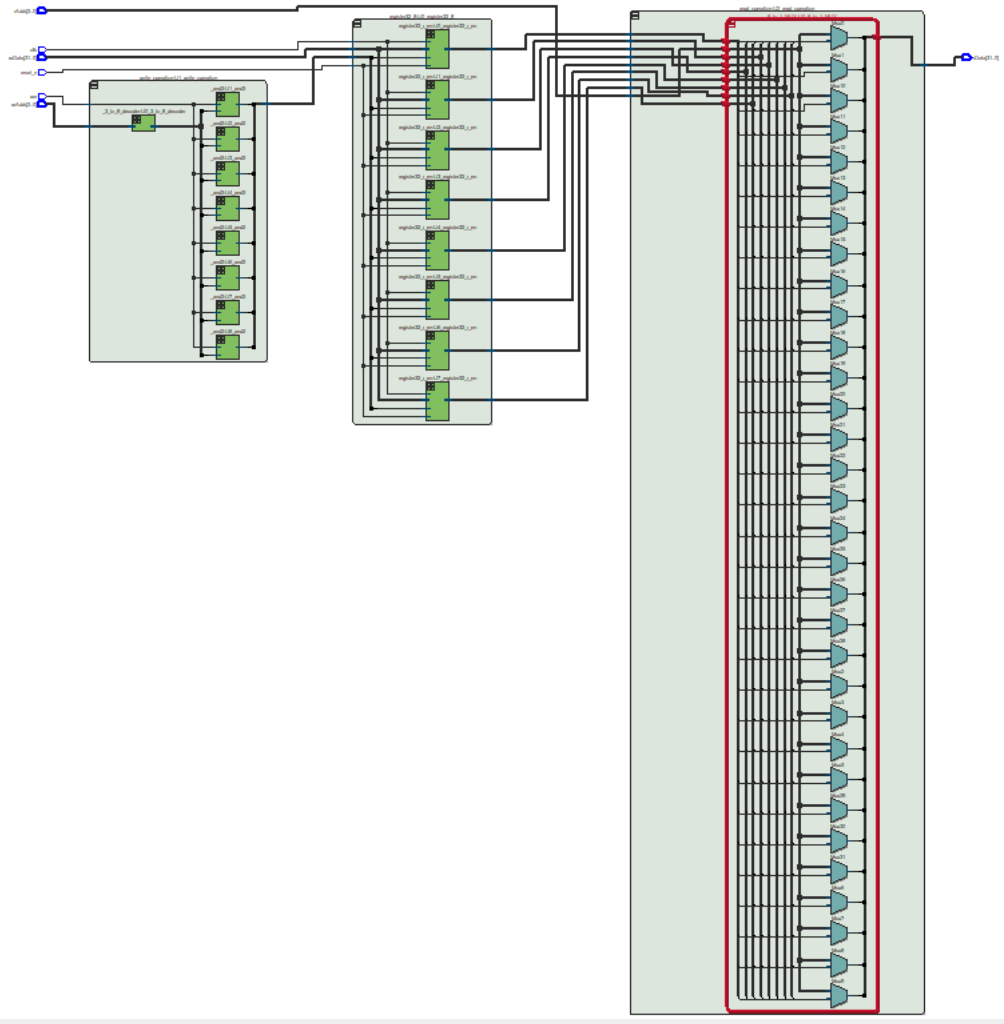
testbench는 강의자료에 제시된 것을 바탕으로 작성하였다. 최초 reset이 동작한 이후 we가 1인 동안 0,1, 2, 3번 register에 저장한 각각의 값을 출력한 결과이다.



B. 합성(synthesis) 결과

RTL viewer는 다음과 같이 나타났다. wAddr, we write_operation module에서 계산된 결과가 값을 저장할 register의 enable로 연결되고, register32_8 module에서 wData가 저장되는 구조임을 확인할 수 있다. read_operation의 경우 MUX를 이용하여 출력을 결정한다.





flow summary는 다음과 같이 나타났다. 73개의 pin을 이용했다. 사용된 73개의 pin중 64개는 각각 d_in, d_out에 사용되었고, wAddr와 rAddr가 6개, clk, reset_n, we가 나머지 3개의 pin을 이용한다. 32bit register가 모두 8개 이용되었으므로 총 256개의 register가 사용되었다.

Flow Summary	
Flow Status	Successful - Sun Oct 18 20:56:50 2020
Quartus Prime Version	15.1.0 Build 185 10/21/2015 SJ Lite Edition
Revision Name	Register_file
Top-level Entity Name	Register_file
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	129 / 41,910 (< 1 %)
Total registers	256
Total pins	73 / 499 (15 %)
Total virtual pins	0
Total block memory bits	0 / 5,662,720 (0 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

5. 고찰 및 결론

A. 고찰

testbench를 작성하는 과정에서 input으로 전달할 data의 진수표현을 잘못 입력하여 오류를 발생시켰다. input이나 output, wire, reg 등을 선언할 때 [31:0]과 같은 형태를 이용하였기 때문에 tb_wData의 값을 31'hff00ff00으로 입력하여 32bit인 data의 첫 번째 bit가 잘려 값이 7f00ff00로 출력되어 제대로 출력되지 않았다. 이후 32'hff00ff00으로 수정하여 값이 제대로 출력됨을 확인하였다.

B. 결론

이번 실험은 register file을 구현하고 동작을 확인하는 실험이었다. 이론시간에 배운 register file의 내용을 자세히 이해하는 것에 도움이 되었다. 특히 read와 write 동작이 동시에 진행된다는 부분이 정확하게 이해되지 않았는데 직접 Verilog code를 작성하여 확인할 수 있었다.

6. 참고문헌

이준환교수님, 디지털논리회로2 강의자료, 광운대학교 컴퓨터정보공학과,2020

이준환교수님, 컴퓨터공학기초실험2 강의자료, 광운대학교 컴퓨터정보공학과,2020

David Money Harris 외1인, Digital Design and Computer Architecture, Elesvier