컴퓨터 공학 기초 실험2 보고서

실험제목: Register File

실험일자: 2022년 10월 18일 (화)

제출일자: 2022년 10월 30일 (일)

학 과: 컴퓨터정보공학부

담당교수: 공영호 교수님

실습분반: 화요일 0,1,2

학 번: 2019202050

성 명: 이강현

1. 제목 및 목적

A. 제목

Register File

B. 목적

Register를 통해 읽기와 쓰기가 가능한 register file에 대한 구조를 익히고 이를 verilog를 통해 코드를 통해 구현해본다. 주소에 따른 데이터 접근과 수정의 원리를 알아본다.

2. 원리(배경지식)

<Stack>

Stack은 LIFO(Last Input First Output)로 후입선출 자료구조를 말한다. 스택에서 데이터의 추가시에는 스택의 맨 아래서부터 순차적으로 쌓이고 이러한 과정을 push라고 칭한다. 또한 스택에서 데이터가 삭제될 시에는 스택의 맨 위에서부터 순차적으로 나가며 이를 pop이라 칭한다. Top은 맨 위 즉 pop을 할 때 나가는 데이터를 가리키는 역할을 한다.

<Queue>

Queue는 FIFO(First Input First Output)로 선입선출 자료구조를 말한다. 따라서 스택과는 달리 가장 먼저 들어온 데이터가 가장 먼저 나가게되는 구조이다. 스택과 다른 점은 pop을 할 때 맨 아래데이터부터 시행되므로 가장 먼저 들어온 데이터가 가장 먼저 나갈 수 있다.

<Register File>

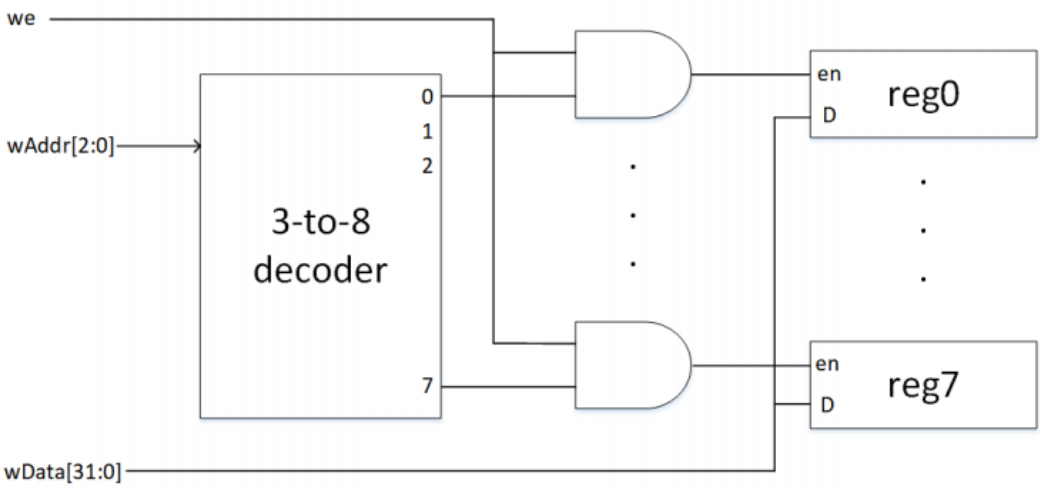
Register File은 register로 이루어져 있으며 주소를 통해 데이터의 접근과 수정이 가능하다. Write operation, read operation, 8_32bits register들로 이루어져 있다. Write operation logic에서 디코더를 통해 8개의 register중 하나의 enable 신호를 1로 만들어주면 이를 read operation logic에서 mux를 통하여 8개의 register 중 한 개를 선택하여 선택된 레지스터의 값을 출력하는 방식으로 작동한다.

3. 설계 세부사항

Register file은 write operation logic과 32bit register 8개, read operation logic으로 크게 구성되어 있다. Write operation logic의 가장 핵심이 되는 요소는 3-to-8 decoder이다. 이는 3bit로 8개의 값을 만들어내는 회로이다. 따라서 아래와 같은 진리표를 가진다.

Input			Output							
I[2]	I[1]	I[0]	O[7]	O[6]	O[5]	O[4]	O[3]	O[2]	O[1]	O[0]
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

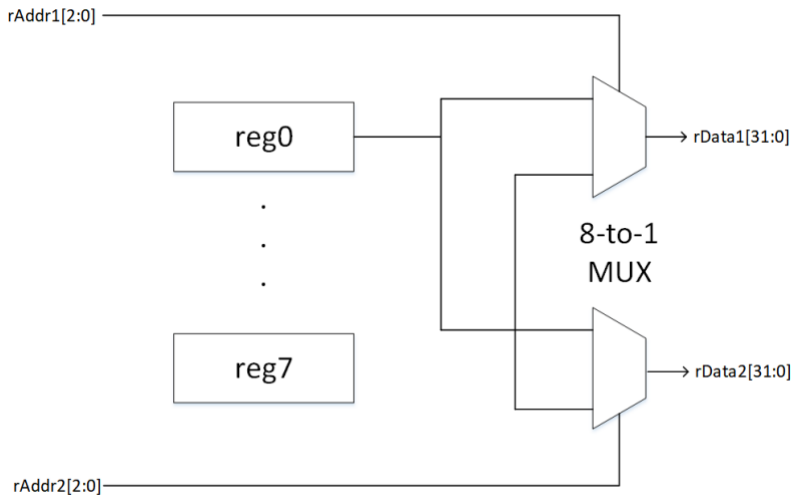
아래는 올바른 write operation을 위해 구현된 회로의 모습이다. wAddr 입력값을 통해 어떤 register를 선택할지를 write enable 입력값과 decoder의 출력값을 함께 and gate로 묶어 register 각각에 enable단자에 연결해주었다. 이렇게 되면 write enable이 활성화되고 해당 주소가 일치하면 해당 레지스터에 데이터를 넣어줄 수 있다.



32bit register 8개의 구현은 32bit register를 always 구문을 통해 구현하고 이를 8개를 instance하여 구현하였다.

마지막으로 register에 저장된 값을 읽어 출력해주는 read operation logic은 8개중

1개을 내보내는 역할을 하는 mux를 활용하여 구현한다. 주소값은 select의 역할을 하며 case구문을 통해 해당주소에서 해당 레지스터값을 가리키게 구현하였다. 구현방법은 아래와 같다.

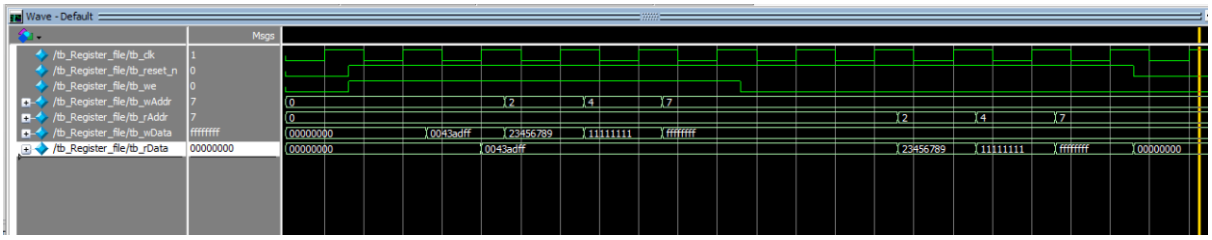


주소값 rAddr이 mux의 select가 되며 8개의 mux가 병렬적으로 연결되어 있음을 확인할 수 있다.

4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

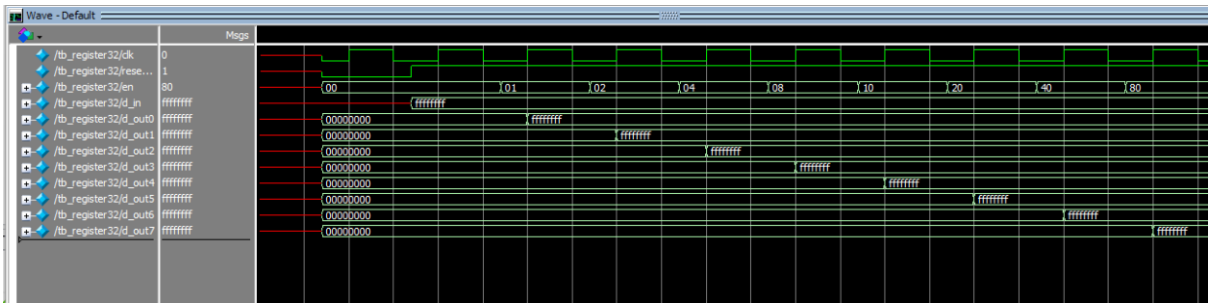
<Register_file>(top module)



Top module인 Register file의 시뮬레이션 결과이다. Testbench 흐름은 무작위로 wAddr에 값을 주고 이에 wData를 각각 다르게 입력한 뒤 rAddr값을 wAddr과 같은 값을 주어 rData를 통해 register에 올바른 값이 저장되어있는지를 확인하는 흐름이다. 위의 결과를 보면 wData에 넣어준 값이 read할때에 해당주소에 올바르게 저장되어 출력시 나타난 것을 확인할 수 있다.

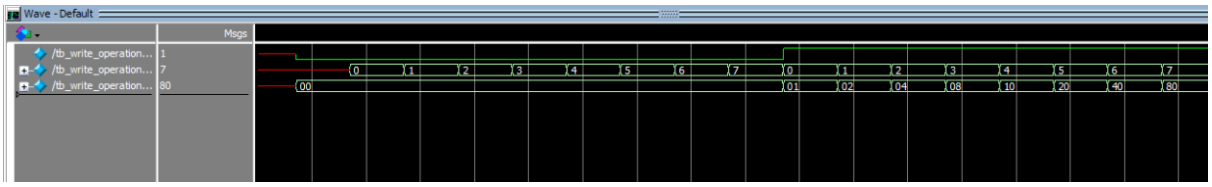
Exhaustive verification을 하기엔 입력케이스가 너무 많아 임의적으로 3개의 주소에 3개의 값을 정하여 direct verification방법을 사용하였다.

<register32_8>(sub module)



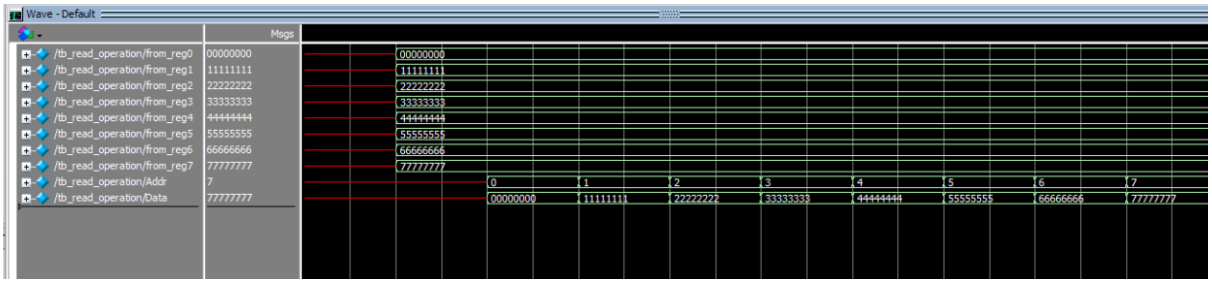
Register32_8파일의 시뮬레이션 결과이다. 해당 레지스터주소에 값이 잘 저장되는지를 확인하기 위해 ffffffff라는 데이터가 en값이 00000001,00000010,00000100....10000000까지 일 때 각각 해당 레지스터값이 저장되는 것을 흐름적으로 확인할 수 있다. 따라서 올바른 결과를 얻었다.

<write_operation>(sub module)



Write_operation의 시뮬레이션 결과이다. Write enable이 0일때는 작동하지 않고 1일 때 주소에 맞게 각각 00000001,00000010,....10000000까지 디코더를 통과한 값들이 나온 것을 확인할 수 있다.

<read_operation>(sub module)

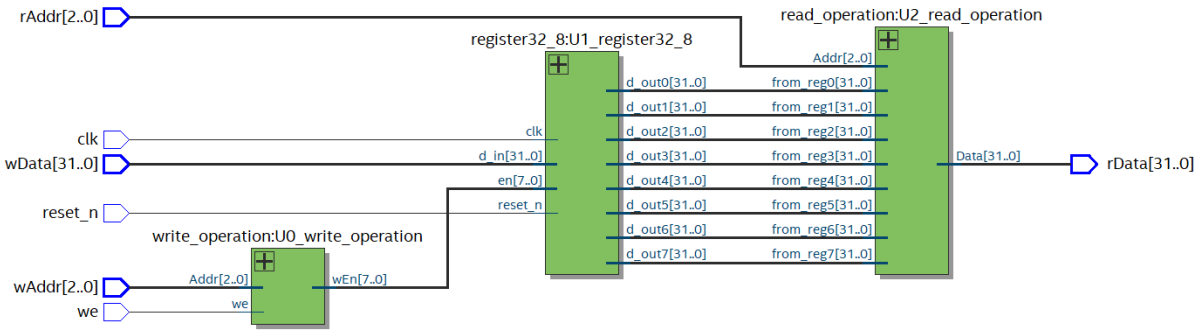


Read_operation의 시뮬레이션 결과이다. 각각의 레지스터 값들이 Addr값에 따라 Data에 보여지는 것을 확인할 수 있다. Read_operation에는 mux가 핵심인데 잘 구현되었다는 것을 알 수 있다.

B. 합성(synthesis) 결과

<Register_file>(top module)

<RTL viewer>



Register_file의 RTL viewer이다. 소자들이 서로 잘 연결되어 있음을 확인할 수 있다.

<flow summary>

Flow Summary

<<Filter>>

Flow Status	Successful - Sun Oct 30 19:02:29 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 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)	N/A
Total registers	256
Total pins	73
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

Register가 256개 사용되었다. 32*8 = 256이기에 32비트 레지스터가 8개 사용됨을 알 수 있고 pin은 73개가 사용되었다.

5. 고찰 및 결론

A. 고찰

quartus라는 툴을 통해 여러 번 실습을 해보았는데 d latch를 instance하여 d flip flop을 만들고 이를 instance하여 8bit register를 만들고 또 이를 instance하여 32bit register를 만들었는데 이론적으로는 이런식으로 구현하나 쿼터스에서 한번에 32비트 변수를 선언하고 always구문과 조건문을 통해 구현할 수 있어 배우면 배울수록 편리한 툴이라는 생각을 하였다.

B. 결론

Register file이라는 개념이 이론적으로 학습하였을 때는 레지스터가 여러 개 있구나라는 생각만 있었으나 실습을 통해 확인해보니 더욱 이해가 잘 되었다. register라는 것을 통해 값을 저장하는 것은 알고 있었으나 write enable과 같은 단자를 통해 데이터를 읽을지 말지를 결정하고 주소값을 받아 해당 주소에 맞는 데이터를 뽑아낸다는 것이 우리가 사용하고 있는 컴퓨터에서 데이터를 어떻게 관리하고 사용자가 원하는 데이터를 어떻게 화면에 보여주는지에 대해 감을 잡을 수 있었던 실험이었다. 그동안 배운 mux나 decorder와 같은 것들이 실제로 어떻게 사용되는지 확실하게 체감되는 실험이었다.

6. 참고문헌

공영호 교수님/컴퓨터공학기초실험2/2022