컴퓨터 공학 기초 실험2 보고서

실험제목: Register file

실험일자: 2022년 10월 19일 (월)

제출일자: 2022년 11월 01일 (일)

학 과:컴퓨터정보공학부

담당교수: 공진흥 교수님

실습분반: 수요일 0, 1, 2

학 번: 2019202021

성 명: 정 성 엽

1. 제목 및 목적

A. 제목

Register file

B. 목적

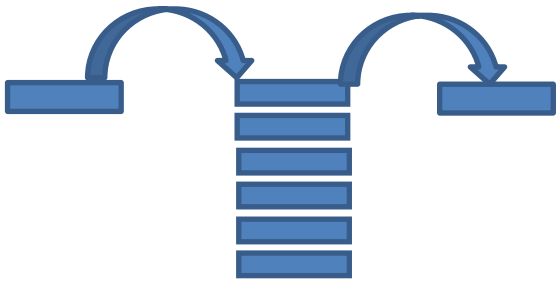
Register 여러 개를 묶어 특정 주소에 bit를 저장하거나 불러올수 있는 register file의 원리를 이해하고 Verilog를 통해 제작하여 구현한다. 추가적으로 새로운 모듈을 모두 testbench로 검증한다.

2. 원리(배경지식)

A. Stack

stack이란 데이터를 Last in, first out의 형식으로 저장하는 자료구조의 일종으로 먼저 들어간 데이터는 아래부터 블록처럼 쌓이고, 나중에 들어간 데이터는 먼저 들어간 데이터 위에 블록처럼 쌓는다. 이 과정을 Push라고 명칭하며, 해당 구조를 그림으로 그리면 아래와 같다. 후에 이 데이터를 불러올 때는 제일 위에 있는 데이터부터 차례로 하나씩 꺼낼 수 있으며 이 과정을 Pop이라고 명칭한다.

크기에 제한이 되어있는 스택은 배열을 통해 쉽게 구현 가능한데 이 때 기존 배열처럼 0부터 데이터를 저장하였다가 0부터 꺼내고 그 다음 1, 2, 3, 순서로 이동하여 데이터를 출력하면 된다.



LAST IN, FIRST OUT

B. Queue

queue는 데이터를 first in, first out 형식으로 저장하는 자료구조의 일종으로 먼저 들어간 데이터가 가자 먼저 출력된다. 데이터를 순서대로 queue에 저장하고 가장 먼저 삽입한 데이터를 head, 가장 나중에 삽입한 데이터를 tail로 잡아 가장 먼저 저장된 데이터를 불러와 삭제하거나, 제일 처음 데이터를 삽입하는 명령어가 있다. 이는 다음주 FIFO에서 다룰 예정이다. 해당 구조를 그림으로 표현하면 아래와 같다.

스택과 마찬가지로 크기에 제한이 있다면 배열을 통해 쉽게 구현이 가능하다. 여기

서 배열의 구성은 register과 동일하다고 볼 수 있으며 FSM으로 state를 구현한 controller를 통해 stack/queue의 동장을 논리회로를 통해서 구현할 수 있다.

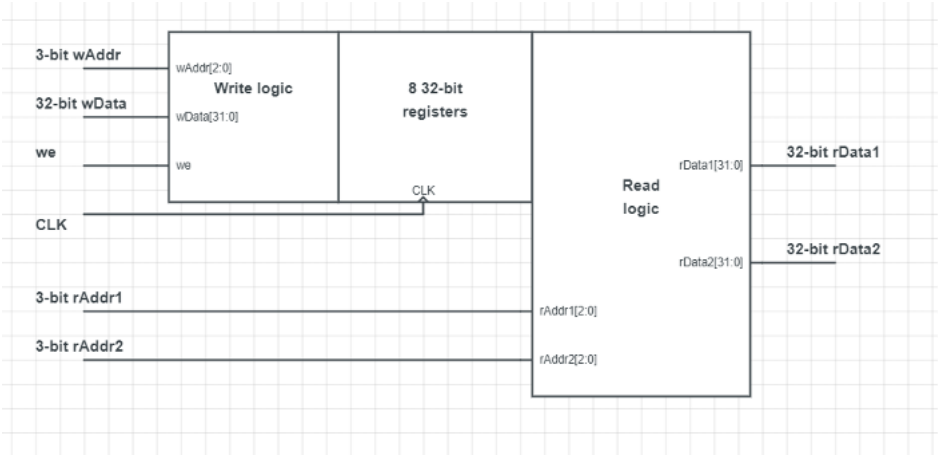
FIRST IN, FISRT OUT

C. Register File

Register file이란 register 여러 개를 그룹으로 묶은 것으로 여러 개의 레지스터에서 특정 레지스터에 접근하여 원활한 읽고 쓰기를 수행하기 위해 특정 주소값을 부여하여 주소의 bit에 따라 접근할 수 있는 register의 개수가 정해진다. Register file은 read logic, write logic, register 모음 단위로 쪼개서 구현한다. 이는 이번 과제에서도 사용하는 구성이다.

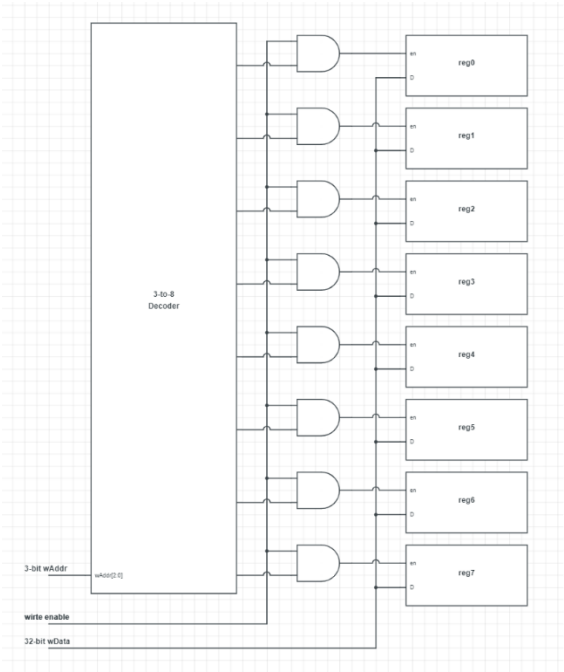
3. 설계 세부사항

A. Register File



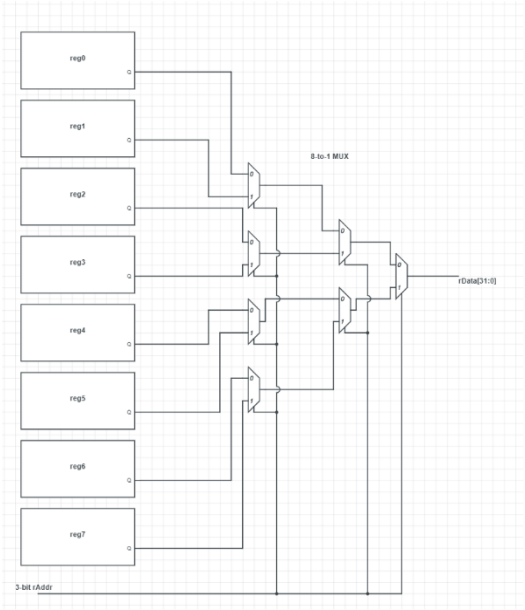
위 그림과 같이 데이터를 두개를 한 번에 읽는 것이 아니라 한 개만 읽을 수 있으며 각각 wAddr, wData 그리고 rAddr, rAddr로 나눠서 input이 된다.

B. Write Operation



3bit의 주소, 입력할 32bit의 data, 그리고 enable 값을 받아 해당 register에 값을 쓰고, Binary Encoding된 3_bit wAddr을 one_hot encoding으로 바꿔 해당 주소에 맞는 register에 값을 삽입하기 위해 3-to-8 decoder를 사용한다. 그리고 enable의 값을 통해 삽입 유무를 정한다.

C. Read Operation



3bit 주소를 32bit의 8-to-1 mux에 입력하면 그 주소에 따른 register의 값을 선택하여 출력한다.

D. Module 정리

구분	이름	설명
Top module	Register_file	top_module

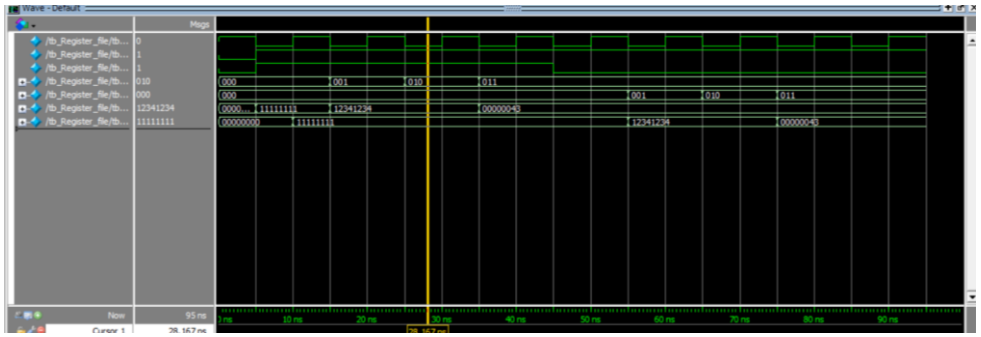
Sub module	register32_8	8개의 32bit register module
	write_operation	write address decode module
	read_operation	select register using read address

module	구분	이름	비트	설명
register_file	input	clk	1	clock
		reset_n	1	active low에 동작하는 reset신호가 인가되면 register 값을 0으로 초기화
		we	1	write enable
		wAddr	3	write address
		rAddr	3	read address
		wData	32	write data
	output	rData	32	read data
register32_8	input	clk	1	clk
		reset_n	1	active low에 동작하는 reset 신호가 인가되면 register값을 0으로 초기화
		en	1	register enable
		d_in	32	data in
	output	d_out0~ d_out7	32	register data out
write_operation	input	we	1	write enable
		Addr	3	Write address
	output	wEn	8	Selected register enable signal
read_operation	input	from_reg0~ from_reg7	32	8 registers data
		Addr	3	Read address
	output	data	32	data out

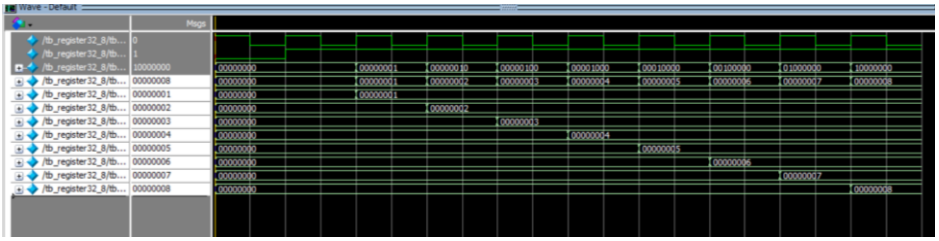
4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

i. register file



ii. register32_8

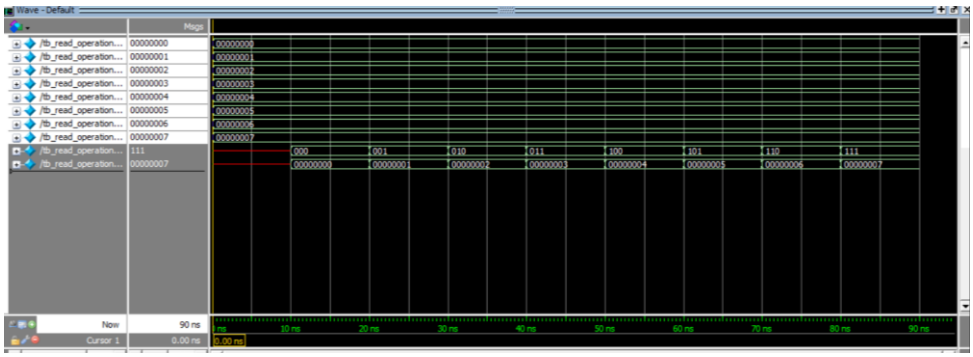


one-hot encoding에 대해 검증을 위해 1bit 씩 저장하고 있는 레지스터의 경우를 확인하였고 각 en 마다 값이 바뀜을 testbench를 통해 확인할 수 있다.

iii. write_operation

주소값이 바뀜에 따라 저장할 위치가 변화함을 testbench를 통해 확인할 수 있다.

iv. read_operation



각 주소별 값을 읽어 오는 것을 확인 할 수 있다.

v. 3_to_8_decoder

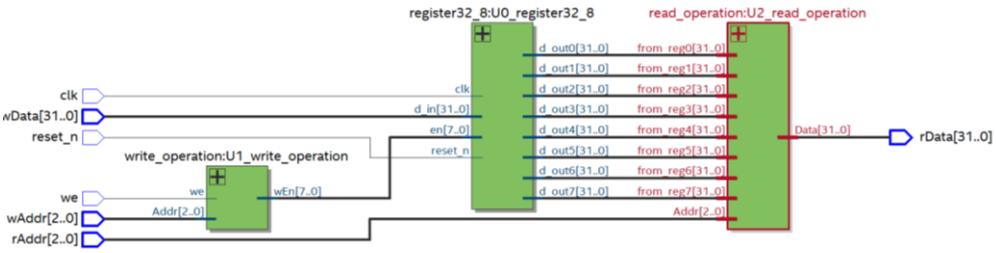
d의 값에 따라 decoding 되어 q의 값이 출력됨을 test bench를 통해 확인할 수 있다.

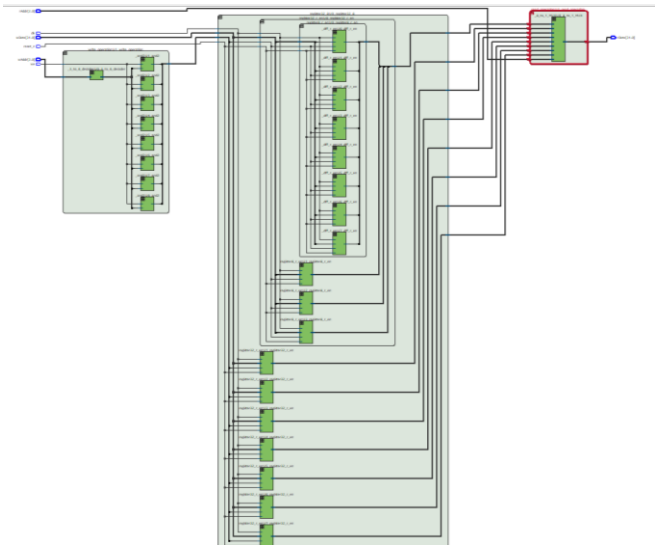
B. 합성(synthesis) 결과

<flow summary>

Flow Summary	
Flow Status	Successful - Tue Nov 01 20:25:36 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	RF
Top-level Entity Name	Register_file
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	123 / 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 %)

<RTL Viewer>





register32_8을 구현하기 위해 dff 8개, register8_r_en 4개를 사용하여 구현하였다. 나머지 RTL viewer는 pin 개수 초과 오류로 인해서 top module 단에서 RTL viewer를 세부사항을 확대하여 확인할 수 있다.

5. 고찰 및 결론

A. 고찰

1개의 쓰기와 1개의 읽기가 가능한 register file을 verilog를 통해 직접 구현하였지만, 보통 register file을 구현할 때에는 기본 1개 쓰기와 2개 읽기가 가능하게 구현하였다 왜냐하면 2개의 인자를 가지고 연산을 수행하는 경우가 더 많기 때문이다. 예를 들어 ALU에서 덧셈 연산을 위해 값을 불러온다면 2개의 값을 읽어 더한 후 1개의 값에 써야 된다. 따라서 동시에 2개의 읽기와 1개의 쓰기가 가능해야 하는 것이다. 이번 실험에서는 이러한 2개의 읽기 연산이 필요한 경우가 아니고, Register file이 1개 읽고 1개 쓸 수 있도록 구현하였다.

write operation에서는 we 명령어와 register의 주소 그리고 값을 입력 받아 저장한다. we는 단순히 쓰기 가능과 불가능 여부를 정해준다. 또한 입력받은 주소를 변환하여 해당 레지스터에 저장할 수 있도록 해야하는데 이는 3-to-8 decoder를 이용하여 one-hot encoding과 비슷하게 각 레지스터를 이를 각 1개씩만 반응할 수 있게 한다.

Read operation은 8개의 register를 가지므로 8-to-1 muxfmf 이용하여 원하는 주소의 register값을 읽을 수 있다.

우연하게도 어셈블리프로그래밍 설계 및 실습에서도 해당 register을 push pop하여 가져오는 실습을 하여 stack, queue 이론 자체는 이해하기 정말 쉬웠다.

B. 결론

이번 실험을 통해 register file에 대한 구조와 설계 방법을 직접 구현해보면서 익힐 수 있

었으며 register file은 단순히 혼자 사용되는 것이 아니라 여러 논리 회로에 응용되어 사용된다. 특히 다수의 값을 저장하는 FIFO 모듈이나 ALU 계산에서는 사용될 값을 저장할 때 사용하는 것을 예로 들 수 있다. 이러한 모듈에서 사용할 Register file을 구현할 때는 연산에 필요한 read write 개수를 정확히 고려해야할 필요가 있다.

6. 참고문헌

1) 공진흥 교수님/컴퓨터공학기초실험2/광운대학교 컴퓨터정보공학부/2022
2) 공영호 교수님/디지털논리회로2/광운대학교 컴퓨터정보공학부/2022