컴퓨터 공학 기초 실험2 보고서

실험제목: ALU with Multiplier

실험일자: 2022년 11월 11일 (월)

제출일자: 2022년 12월 03일 (토)

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

ALU with Multiplier

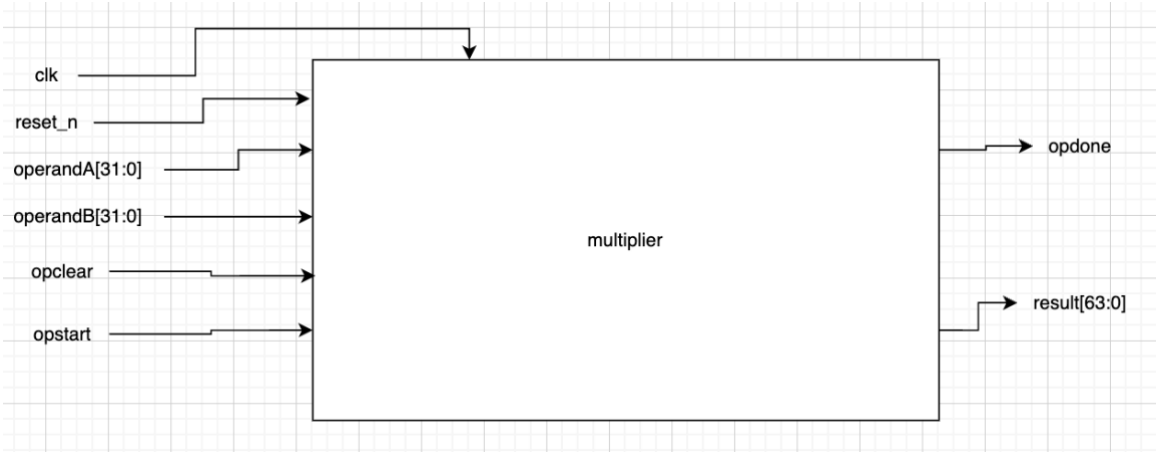
B. 목적

이전까지 해왔던 Ram, Bus, multiplier, alu, shift 등을 조합하여 하나의 장치로 제작한다. Bus와 ALUwMUl과 memory는 top 모듈 안에서 연결되고 testbench를 통해 값을 입력해 읽고 써서 검증한다.

2. 원리(배경지식)

A. ALU with Multiplier

Multiplier



x_i	x_{i-1}	Operation	Description
0	0	Shift only	String of zeros
0	1	Add and shift	End of a string of ones
1	0	Subtract and shift	Beginning of a string of ones
1	1	Shift only	String of ones

Booth Multiplication

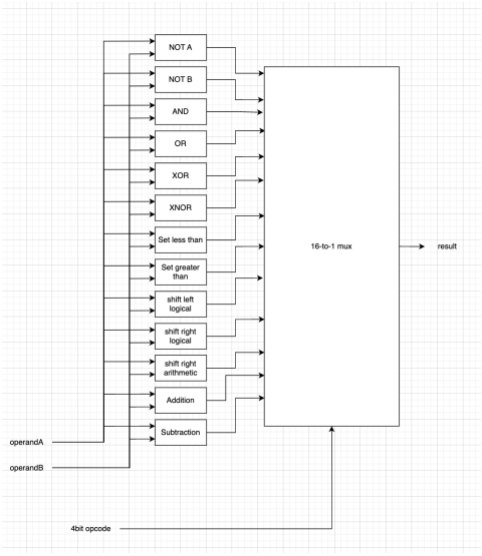
i. Booth Multiplication은 승수의 bit x_i, x_{i-1}의 패턴을 읽고 연산 수행을 반복하여 곱셈을 수행한다. 이때 승수의 패턴에 따른 연산은 아래의 표와 같다. 이 표는 Radix-2를 나타낸 것이다.

x_i	x_{i-1}	Operation	Description
0	0	Shift only	String of zeros
0	1	Add and shift	End of a string of ones
1	0	Subtract and shift	Beginning of a string of ones
1	1	Shift only	String of ones

x_i와 x_{i-1}이 연속 된다면 shift만 진행하며 만약 순서가 달라진다면 01인 경우 add 연산 10인 경우 sub 연산을 진행한다.

ii. 위 표와 다르게 패턴을 더 추가하여 Radix-4 또한 만들 수 있다. 이 패턴을 추가할 수록 더욱 적은 횟수로 연산이 가능하지만 그만큼 더 많은 공간을 차지한다.

ALU



Arithmetic logic unit은 주어진 opcode에 따라 다른 연산 값을 출력하는 모듈로 이전 과제와 다르게 8가지의 선택지가 아닌 13가지의 선택지를 가진다. 이를 위해 16 to 1 mux를 구현하여 원하는 선택지를 가져오도록 설계한다.

B. BUS

Bus는 하드웨어 간의 데이터 이동을 위해 연결해주는 연결 통로의 역할을 하는 것으로 이때 Master은 어떠한 기능을 수행할지 그리고 Slave는 명령

수행 역할을 하여 서로 연결하며 1개의 통로를 이용하여 여러 레지스터를 연결 가능하다. 또한 Master는 request에 따라 구분할 수 있고 Slave 또한 지정된 Address의 범위에 다라 각각 구분된다.

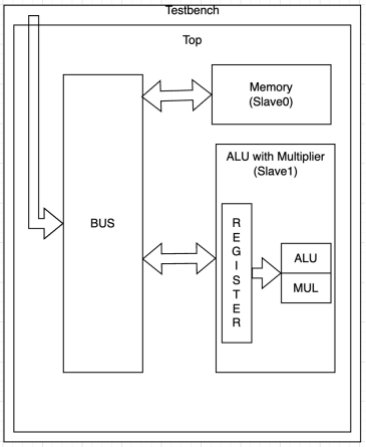
Bus는 레지스터가 포함된 시스템에서 효율적인데 이는 레지스터에서 다른 레지스터로 이동할 때 연결되는 경로를 무한정으로 만들기보다 Bus를 통해 각 레지스터와연결되면 이 Bus를 공용 통로로 이용하는 것이 더 효율적이기 때문이다. 이를 통해데이터의 이동을 할 때마다 여러 연결을 만들 필요없이 Bus만 설계하여 구현한 후 서로를 연결하는 효율적인 설계가 가능해진다.

C. Memory

Memory(Ram)은 register와 함께 2진 정보를 저장한다. 이 Memory와 register의 차이점은 Register은 CPU 내부에 존재하지만 Memory는 외부에 존재하여 Register보다 더 많은 정보를 읽고 쓸 수 있다. 다만 외부에 존재하는 만큼 속도는 느리다. 다른 비교군으로 ROM을 비교할 수 있다. ROM은 Read Only Memory로 영구적으로 데이터를 저장하기만 가능하고 쓰기는 불가능하다. 이에 반해 RAM은 Random Access Memory로 전원이 나가면 데이터가 손실되지만 속도는 빠르고 읽고 쓰기 가능하다. 최근에 ROM은 NAND Flash Memory로 제작되어서 읽고 쓰기 모두 가능하다.

3. 설계 세부사항

A. Top module

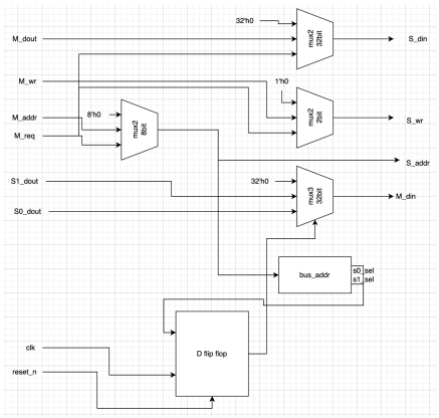


Address region(in Hexadecimal)	Component
0x00~0x1F	Memory
0x30~0x3F	ALU with Multiplier

Top모듈은 위의 그림과 같으며 testbench가 master로 값을 입력한다. 이 모듈은 안의 bus, memory, ALU with Multiplier를 모두 instance하여 연결되어 있고 input port를 이용해 master portdp 접근하도록 한다. Input과 output을 설정하고 후에 testbench가 설정하는 주소를 bus로 넣어 원하는 slave에 접근하도록 한다. 해당 내용은 아래 bus에서 다룬다.

구분	이름	비트	설명
Input	Clk	1	Clock
	Reset_n	1	Active low reset
	M_req	1	Master request
	M_wr	1	Master write /read
	M_addr	8	Master address
	M_dout	32	Master data output
output	M_grant	1	Master grant
	M_din	32	Master data in

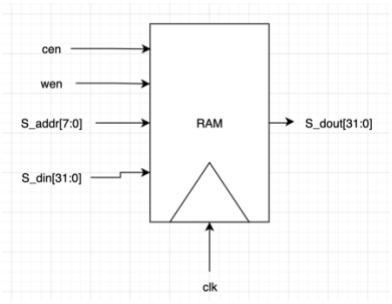
B. BUS



이번 과제의 bus는 master가 1개 slave가 2개인 구조로 입력 받은 address가 0x00부터 0x1F인 경우 slave0인 memory로 선택하도록 하고 0x30~0x3F인 경우 slave1인 ALU with Multiplier를 선택하도록 한다. 이는 S0_sel과 S1_sel로 구분하게 한다.

구분	이름	비트	설명
Input	Clk	1	Clock
	Reset_n	1	Active low reset
	M_req	1	Master request
	M_wr	1	Master write /read
	M_addr	8	Master address
	M_dout	32	Master data output
	S0_dout	32	Slave 0 data out
	S1_dout	32	Slave 1 data out
output	M_grant	1	Master grant
	M_din	32	Master data input
	S0_sel	1	Slave 0 select
	S1_sel	1	Slave 1 select
	S_addr	8	Slave address
	S_wr	1	Slave write /read
	S_din	32	Slave data input

C. Memory



이전 과제와 크게 다를 것 없지만 addr의 크기만 변경한다. 8bit의 address에 32bit din data를 저장하도록 하여 verilog로 메모리 변수를 직접 선언하여 RAM을 구현하여 데이터를 저장한다. Cen은 Ram을 사용할 때 1로 set하고 사용하지 않으면 0으로 set한다. Wen은 0인 경우 지정된 address에 dout으로 출력하고 1로 set 되면 지정된 address로 din 값을 저정한다.

구분	이름	비트	설명
Input	Clk	1	Clock
	Cen	1	Chip enable
	Wen	1	Write enable
	S_addr	5	Address
	S_din	32	Data in
output	S_dout	32	Data out

D. ALU with Multiplier

설계를 완료하지 못하였다.

4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

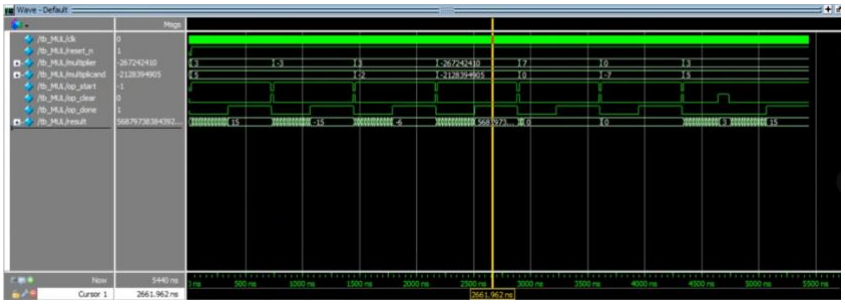
i. Shifter

LSL, LSR, ASR을 32bit로 변경한 후 shift 연산이 문제 없이 작동하는 것을 확인했다. 이전 과제의 testbench를 수정해서 가져오는 과정에 shamt 값을 고정시켰지만 값을 입력하도록 되었다.

ii. ALU

ALU의 13가지를 확인하였다. 값은 임의로 3과 5로 진행하였으면 not A부터 set less than, shift연산, add 연산까지 정상적으로 작동하는 것을 확인하였다.

iii. MUL



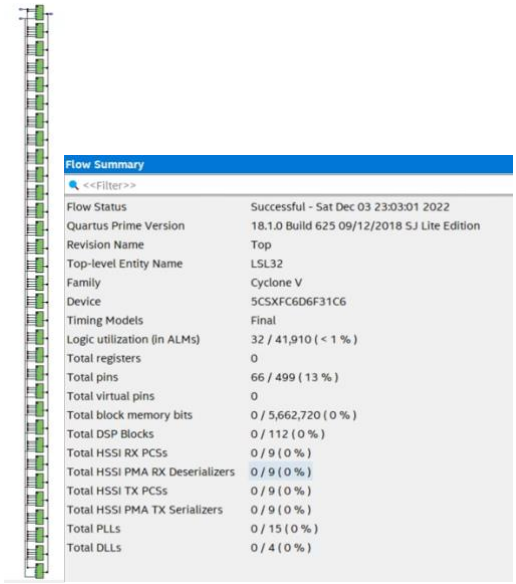
MUL의 testbench도 이전 과제를 참고하여 진행하였으며 32bit로 줄여서 계산하였지만 문제없이 잘 작동하는 것을 확인했다.

iv. Top

ALU with MUL의 모듈을 완성하지 못하여 top 모듈에서는 bus와 연결된 memory만을 확인하여 testbench를 하였다. 기존에 0x30주소라면 ALU with MUL에 연결되어야하지만 해당 모듈을 비워뒀기 때문에 파란선이 나왔고 나머지는 각 주소에 잘 저장했고 출력도 문제 없음을 확인했다.

B. 합성(synthesis) 결과

i. Shifter



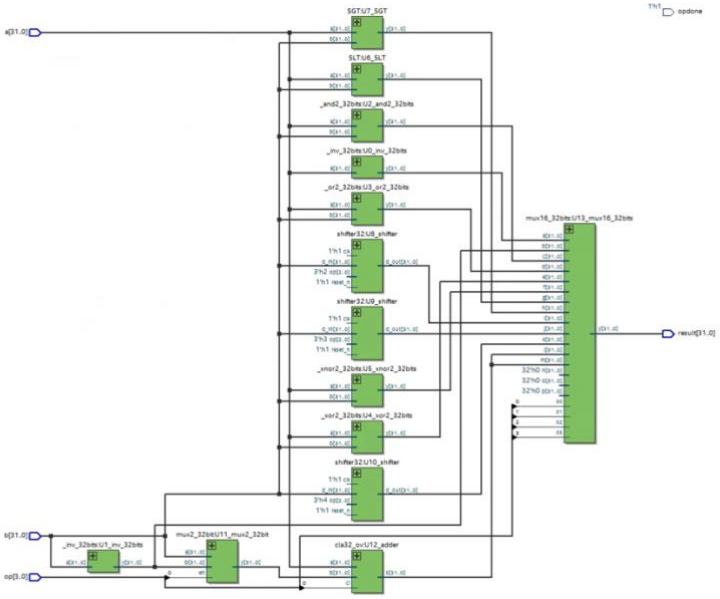
Flow Summary

<<Filter>>

Flow Status	Successful - Sat Dec 03 23:03:01 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	Top
Top-level Entity Name	LSL32
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	32 / 41,910 (< 1 %)
Total registers	0
Total pins	66 / 499 (13 %)
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 %)

LSL 모듈의 예시이고 ASR, LSR 모듈 모두 이와 같은 형식으로 연결되어 있는 RTL viewer 모습을 보인다. 32bit이다 보니 모듈 개수 또한 상당히 증가한 것을 볼 수 있다.

ii. ALU



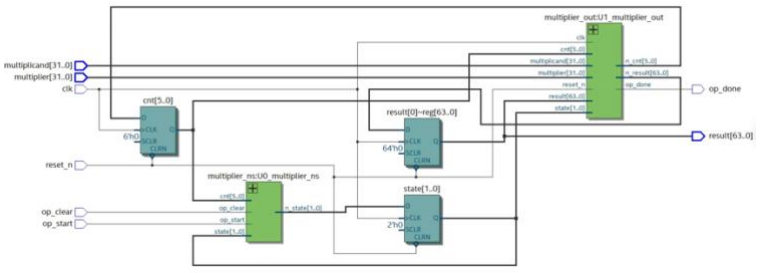
Flow Summary

<<Filter>>

Flow Status	Successful - Sat Dec 03 22:58:22 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	Top
Top-level Entity Name	ALU
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	164 / 41,910 (< 1 %)
Total registers	0
Total pins	132 / 499 (26 %)
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 %)

이번 과제를 위한 13가지 선택지의 alu 모듈의 rtl view이다. 비교를 위한 모듈도 따로 제작하여 gate 모듈에 두었고 shift 모듈도 가져와 적용하였다. 이 연산 값이 16-to-1 mux를 통해 원하는 것을 출력한다.

iii. MUL



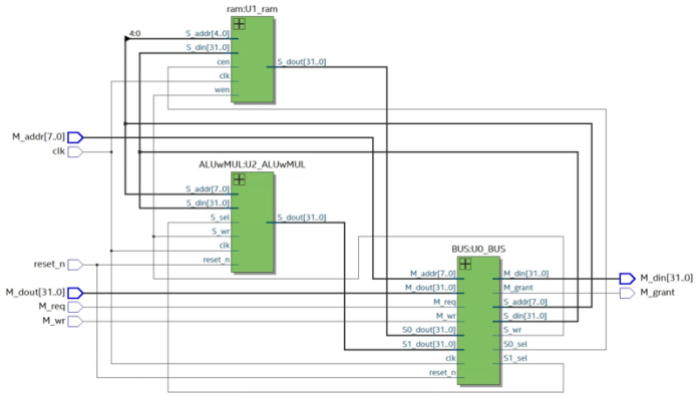
Flow Summary

<<Filter>>

Flow Status	Successful - Sat Dec 03 22:49:24 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	Top
Top-level Entity Name	MUL
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	170 / 41,910 (< 1 %)
Total registers	73
Total pins	133 / 499 (27 %)
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 %)

Multiplier 모듈의 rtl viewer로 값을 입력 받고 next state logic에 따라 state를 이동하여 clk에 따라 연산하도록 설계되어 있다.

iv. TOP



Flow Summary

<<Filter>>

Flow Status	Successful - Sat Dec 03 23:10:37 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	Top
Top-level Entity Name	Top
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	513 / 41,910 (1 %)
Total registers	1058
Total pins	77 / 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 %)

Top module의 rtl viewer로 모듈상에는 연결되어있지만 ALUwMUL을 완성하지 못했기 때문에 따로 작동하지는 않고 memory부분만 작동한다.

5. 고찰 및 결론

A. 고찰

이번 과제에서는 bus와 ALU with Multiplier 그리고 memory를 설계하여 top moduled을 testbench에 올려 연산을 진행하는 것으로 프로젝트를 진행하면서 기존의 것만 사용해서 잘 조합하면 되겠다는 안일한 생각은 ALU with Multiplier에서 막히게 하였다. 당장 이 모듈을 돌리기 위한 FSM구조도 생각나지 않고 직결 연결하려고 하니 구현이 되지 않았다. 특히 alu는 한 사이클 안에 계산이 가능한 방면 MUL은 여러 사이클을 돌려야하기 때문에 이 FSM을 짜는 것이 매우 힘들었고 성공하지 못했다.

BUS모듈은 1개의 master만 있기 때문에 bus_arbit을 따로 두지 않아도 되겠다고 생각하여 따로 두지 않고 2개의 slave를 구분하기 위한 bus_addr을 설계하여 가져왔다. 특히 제안서 초반에 적힌 계산된 결과를 다시 memory로 저장해야한다는 말은 더 헷갈리가 하기 충분했다.

결국 완성을 못하였고 제작한 bus, memory 그리고 alu with multiplier 전용으로 필요한 ALU와 MUL을 제작하여 testbench를 돌렸다. 주어진 FSM은 잘 만들 수 있으나 처음부터 제작하려고 하니 매우 막혔고 답답한 부분들이 많았다.

B. 결론

이번 프로젝트를 통해 BUS와 memory그리고 alu with multiplier를 연결하는 top을 제작하여 testbench를 통해 검증하려고 했으나 완성하지 못했고 이번 프로젝트 제안서에는 각 모듈 별 input/output port 그리고 동작 상태만 주어지고 구체적 구조는 직접 디자인 설계하여야 하여 힘든 점이 많았다.

그리고 다른 학생의 발표를 보면서 사람마다 다른 설계 디자인이 가능하고 변경 될 일이 없다면 해당 과제를 위한 최적화 작업도 가능함을 볼 수 있었다. 물론 변경 사항이 많은 모듈인 경우 최적화 작업을 하면 후에 수정이 불편할 수 있다는 점을 교수님께서 상기시켜 주셨다.

6. 참고문헌

1) 공진흥 교수님/ 컴퓨터공학기초실험2/광운대학교 컴퓨터정보공학부/2022

2) 공영호 교수님/ 디지털논리회로2/광운대학교 컴퓨터정보공학부 /2022