디지털 논리회로2 프로젝트 제안서

ALU with Multiplier

학　　과:컴퓨터정보공학부

담당교수: 공진흥 교수님

실습분반: 수요일 0, 1, 2

학　　번: 2019202021

성　　명: 정 성 엽

1. Title & Object

A. Title

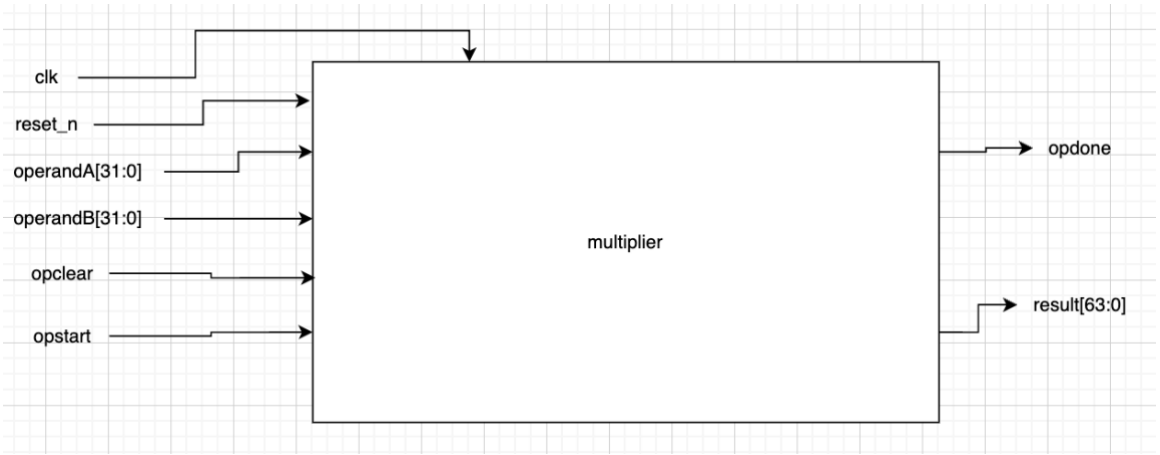
ALU with Multiplier

B. Object

이전까지 해왔던 Ram, Bus, multiplier, ALU, shift 등을 조합하여 하나의 장치로 제작한다. Bus와 ALUwMUL과 memory는 TOP모듈 안에서 연결되고 testbench를 통해 값을 입력해 읽고 써서 검증한다.

2. Component concept

A. ALU with 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

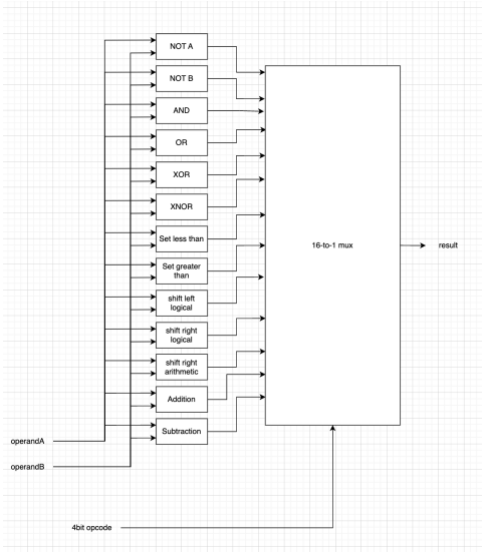
i. Booth Multiplication은 승수의 bit x_i, x_{i-1}의 패턴을 읽고 연산 수행을 반복하여 곱셈을 수행한다. 이때 승수의 패턴에 따른 연산은 아래의 표와 같다. 이 표는 Radix-2를 나타낸 것이다.

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.

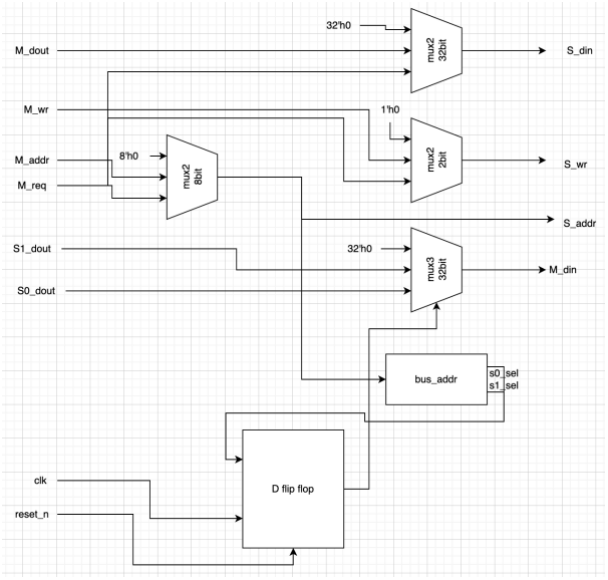
ii. 위 표와 다르게 패턴을 더 추가하여 Radix-4 또한 만들 수 있다. 이 패턴을 추가할 수록 더욱 적은 횟수로 연산이 가능하지만 그만큼 더 많은 공간을 차지한다.

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.

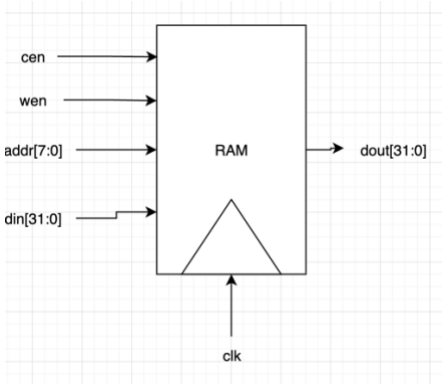
B. Bus



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 레지스터와 연결되면 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.

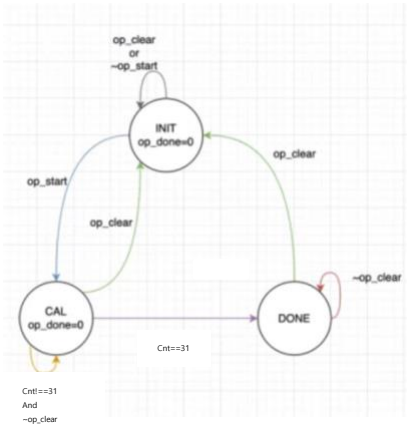
3. Schedule

	10	11	12	13	14
제안서					
코드작성					
코드검증					
결과보고서					

4. State transition diagram

A. ALU with Multiplier

Multiplier state diagram



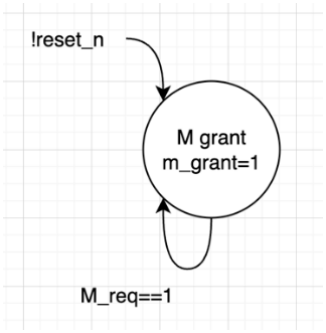
Multiplier를 구현하기 위해 위와 같은 3단계의 FSM을 설계하였다. Op_start는 연산의 시작을, op_clear는 연산 초기화로 새로 연산을 시작할 때, op_done은 연산의 완료를 뜻한다. 이 때 op_clear는 연산을 진행하건 안하든 모든 값을 초기화

하는 것으로 한다.

위의 state diagram처럼 INIT state에서는 시작점을 뜻하고 만약 op_start가 1로 set이된다면 Binary Multiplication 연산을 하는 CAL state로 이동하고 이 state에서는 cnt가 0부터 시작해 31이 될 때까지 연산을 한다. 31번 연산이 끝나면 DONE state로 이동하는데 이는 연산이 종료됨을 뜻하고 op_done을 1로 set한다. Op_clear가 1로 set 된다면 모든 값을 초기화한다.

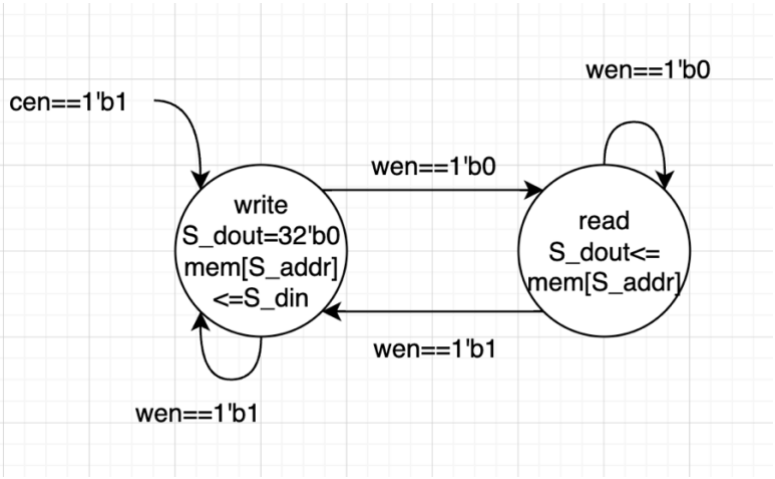
ALU 같은 경우 state diagram이 필요로 하지 않으며 ALUwM의 state diagram은 설계를 완료하지 못하였다.

B. Bus



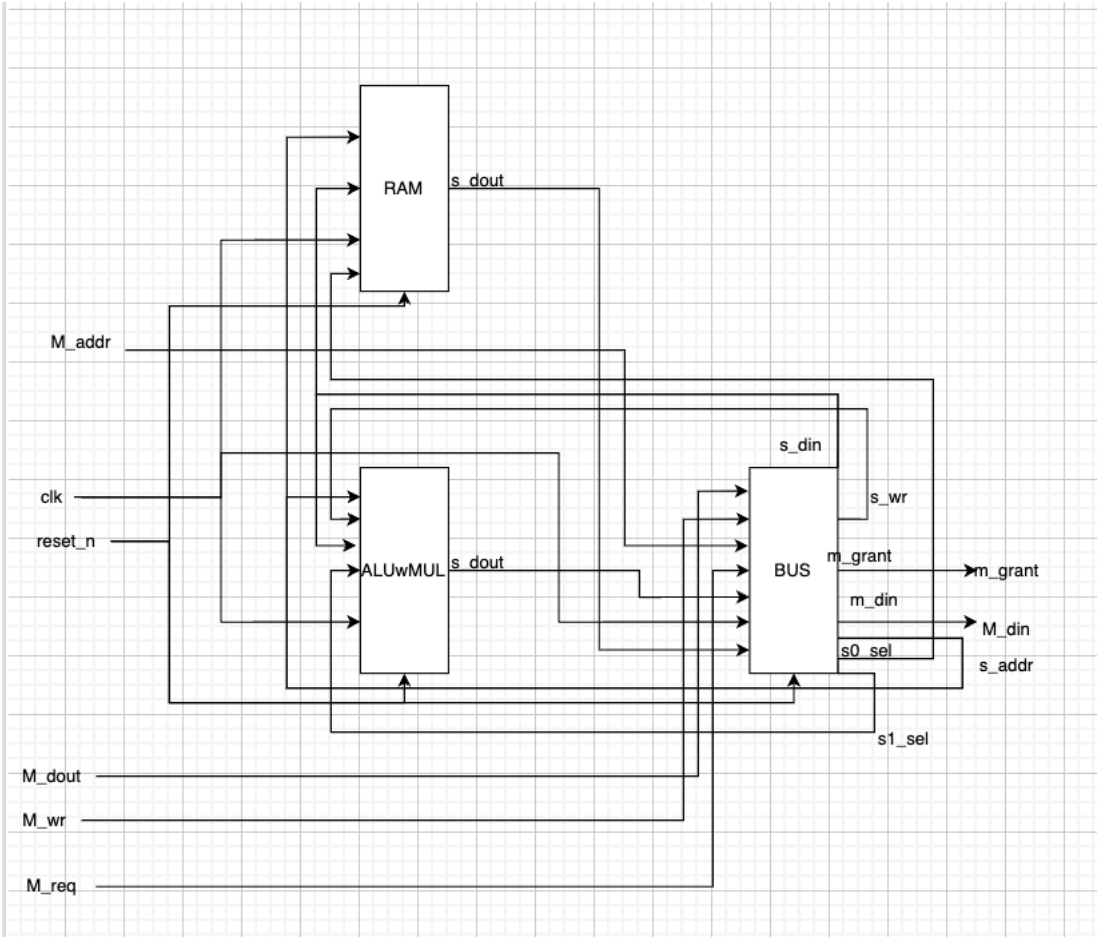
Master가 하나이기 때문에 BUS에서의 state diagram은 크게 유의미하지 않다.

C. Memory



Memory에서는 따로 state logic을 사용하지 않았지만 간단히 표현하면 위 그림과 같고 cen이 1일 때 memory가 시작하고 wen이 1인경우 write를 0인경우 read를 하도록 한다.

5. Module instance design



6. Design verification strategy

기존에 과제에서 사용한 모듈과 다른 점이 매우 많다. 대표적으로 bit 수가 많이 달라졌으며 이는 숫자만 바꿀 것이 아닌 모듈 개수도 달리질 수 있다. 그래서 새로이 만든 MUL, ALU, 등을 각각 testbench를 돌린 후 작동 확인 후에 ALUwMUL을 설계한다.

또한 이번 과제의 ALU의 경우 기존의 8개의 선택지에서 13개의 선택지가 되었으므로 이 또한 고려하여 16-to-1 mux와 ALU의 RTL module단부터 연결을 확인하고 특히 SHIFT 연산을 testbench를 통해 검증한다.