디지털 논리회로2 프로젝트 제안서

Factorial computation system

학 과: 컴퓨터정보공학부

담당교수: 이준환 교수님

실습 분반: 월요일 0, 1, 2

학 번: 2020202031

성 명: 김재현

1. Title & Object

A. Title

Factorial computation system

B. Object

이번 프로젝트는 디지털논리회로2 배운 배경지식을 통해 컴퓨터공학기초실험2에서 구현해왔던 여러 논리회로 모듈들을 이용하여 Factorial 연산 처리가 가능한 Top Module을 구현해봅니다. 또한 Memory, Bus를 조합하여 구현해보고 동작을 확인해 봅니다.

2. Component concept

A. Factorial core

Factorial core는 Booth Multiplier를 통해 operand를 곱셈한 후 Factorial controller를 통해 operand를 1씩 줄여주고는 모듈입니다. opstart, opclear 등으로 연산, 초기화 등을 시작하는 시기를 판단할 수 있고, opdone, result_h, result_l 등을 통해 연산이 끝났는지와, 연산결과를 알 수 있습니다.

B. Bus

request 신호를 통해 Master의 명령을 전달할 것인지를 결정하고, Master에서 전달하는 주소 값의 Base Address를 통해 여러 Slave 중 하나를 선택합니다.

Bus는 컴퓨터 시스템 아키텍처에서 중요한 개념 중 하나로, 여러 컴포넌트 간에 데이터, 주소, 제어 신호 등을 전송하기 위한 통신 경로로 사용됩니다. Bus는 시스템 내에서 정보를 효과적으로 교환하고 각각의 하드웨어 구성 요소 간의 동작을 용이하게 해줍니다.

다양한 종류의 버스가 있으며, 이들은 주로 데이터, 주소, 제어 버스로 구성됩니다.

Data Bus: 시스템 모듈 간의 데이터 이동 경로를 제공해주며, 각 component 간에 데이터를 전송하는 데 사용됩니다.
Address Bus: CPU가 메모리나 입출력 장치에 특정 위치에 접근하기 위해 사용하는데, 주소 정보를 전송하는 데 활용되며, CPU가 어떤 메모리 주소로 데이터를 읽거나 쓸지를 결정하는데 관여합니다.
제어 버스 (Control Bus): 제어 버스는 각각의 구성 요소 간에 제어 신호를 전달하는 데 사용됩니다. 제어 버스는 시스템의 동작을 제어하기 위해 제어 신호를 전송하며,

이는 데이터의 읽기, 쓰기, 버스의 활성화 또는 비활성화 등을 포함합니다.

C. Memory(RAM)

RAM은 주로 컴퓨터가 현재 작업 중인 데이터 및 프로그램을 일시적으로 저장하는 데 사용됩니다. RAM은 휘발성 메모리로, 전원이 꺼지면 저장된 데이터가 소멸하는데, 주로 컴퓨터의 작업 메모리로 사용되어 빠른 읽기 및 쓰기 속도가 필요한 임시 데이터를 보관하는데 사용됩니다.

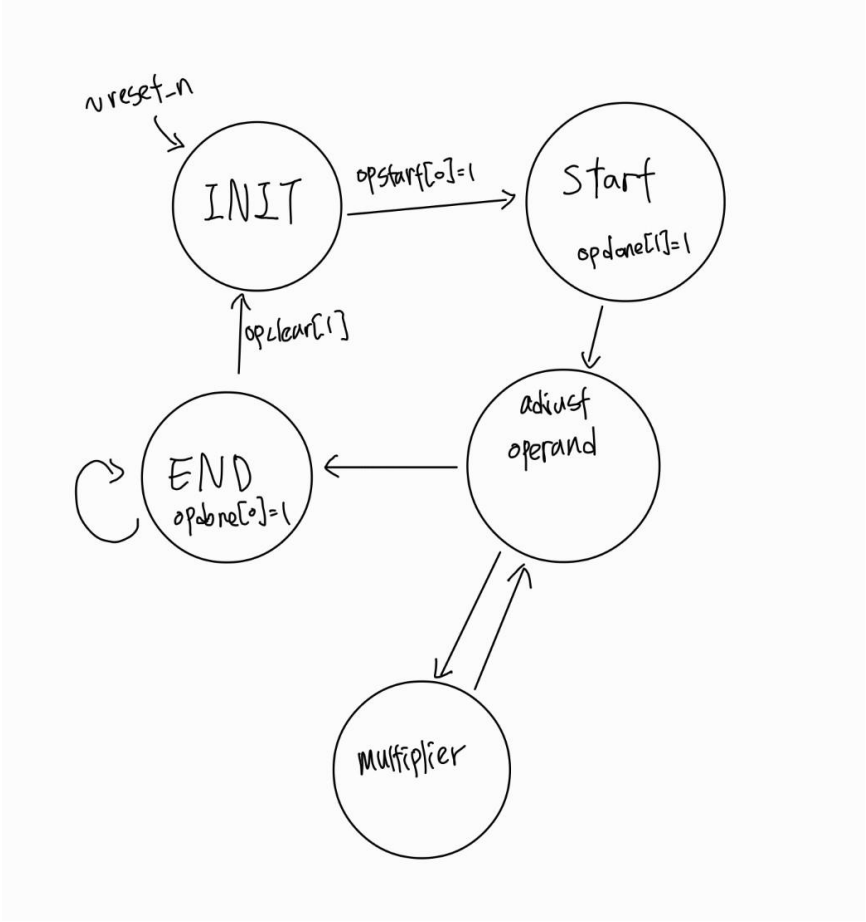
3. Schedule

해당 project를 진행할 계획을 표 또는 그림으로 작성

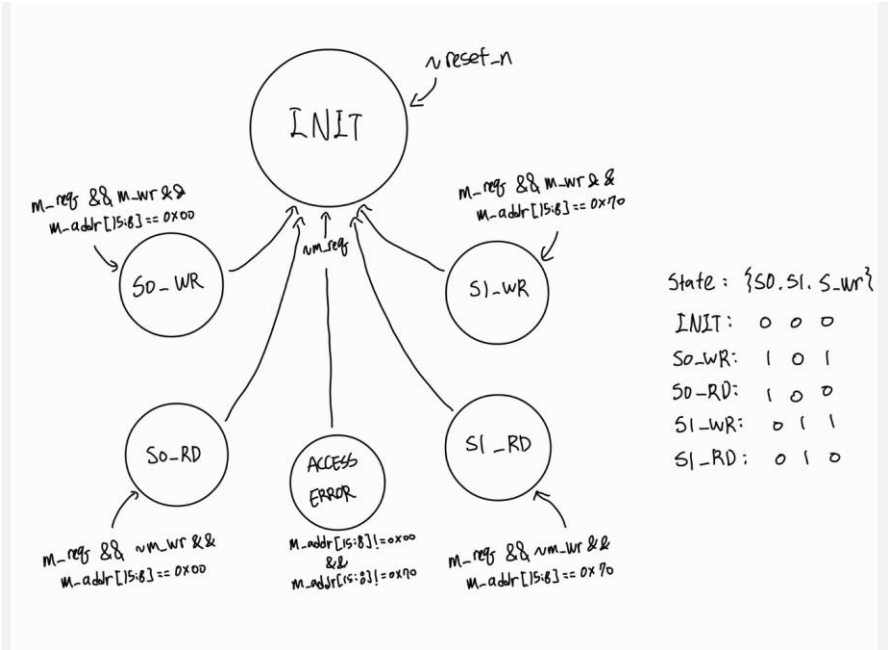
	제안서	코드 작성	코드 검증	결과 보고서
11월 3째주	~11/16			
11월 4째주		~11/25		
11월 5째주		~11/30		
12월 초				~12/6

4. State transition diagram

A. Factorial core



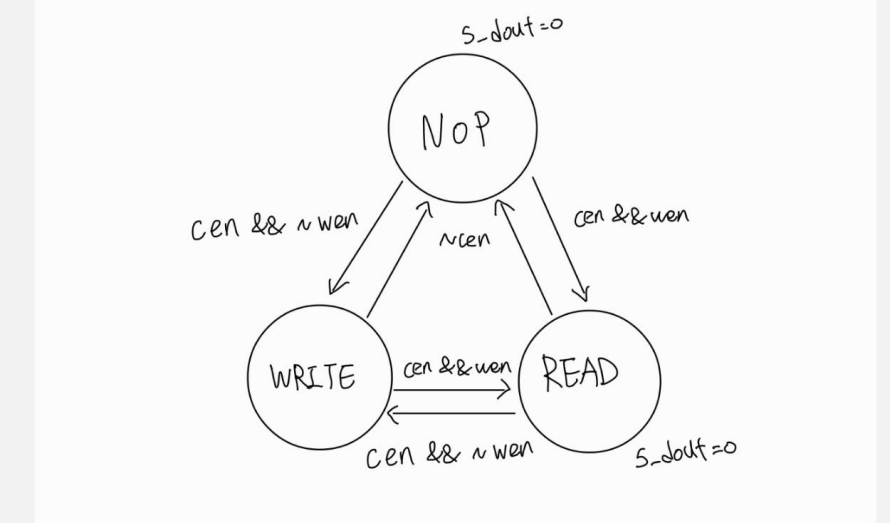
B. Bus



m_req를 통해 master의 request를 slave에게 전달할지 여부를, m_wr를 통해 master

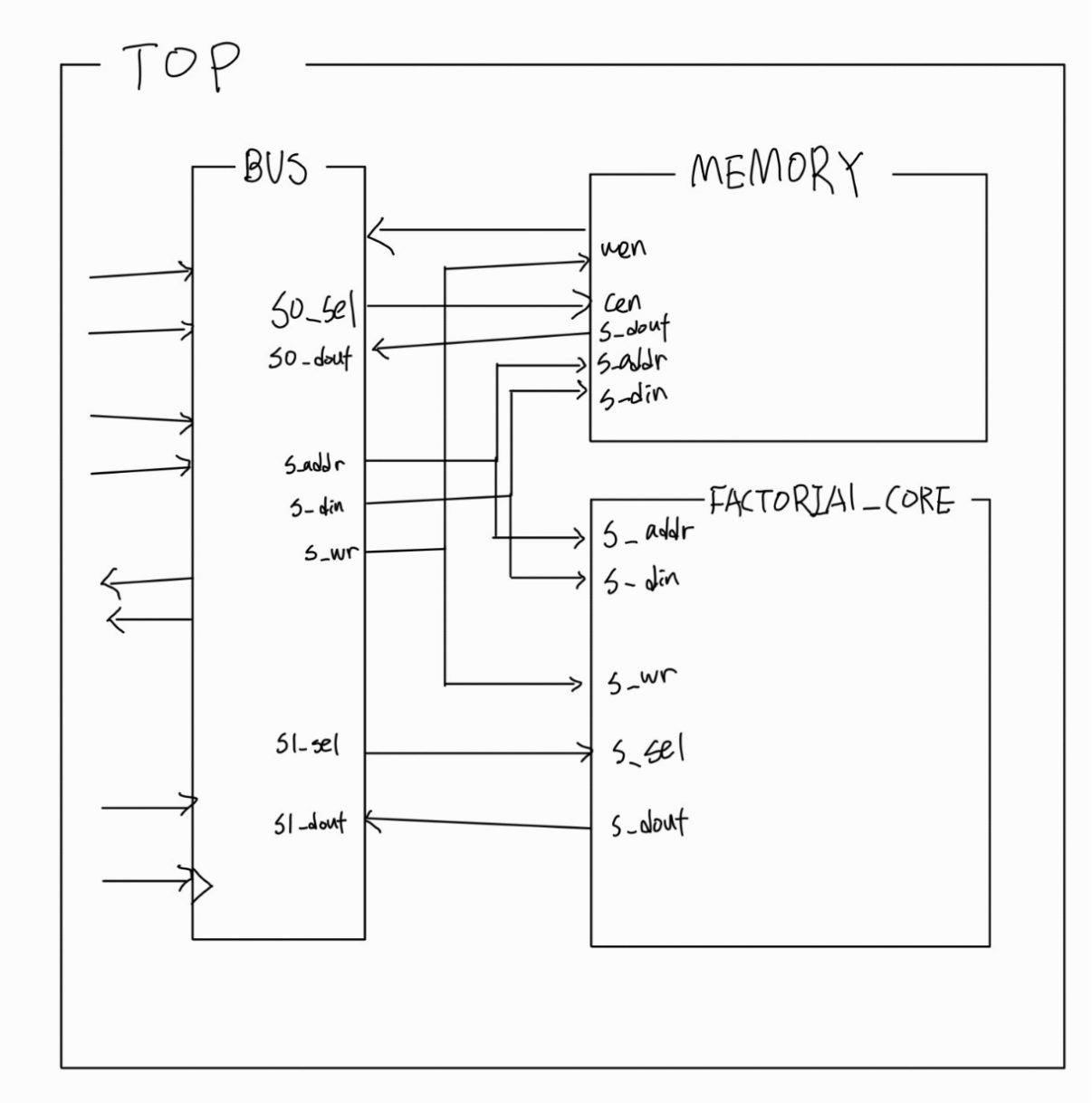
에서 slave에 WRITE or READ를 요청할지 여부를, m_addr[15:8] (base address)를 통해 slave0, slave1 중 어느 것을 택할지 여부를 결정하게 됩니다.

C. Memory



cen이 0이면 master가 factorial core와 연결이 된 상태이므로 아무런 동작을 하지 않고 0을 출력합니다. cen이 1이면, wen이 1일 때 s_addr에 해당하는 메모리에 s_din값을 write하고, wen이 0일 때 s_addr에 해당하는 메모리에 저장된 값을 s_dout에 출력합니다.

5. Module instance design



6. Design verification strategy
    1. operand가 0인 경우
    2. operand의 모든 비트가 1인 경우
    3. operand의 비트에 연속되는 같은 수가 없는 경우
       = booth multiplier에서 operand를 변환시킨 수의 비트가 0이 없는 경우
    4. operand의 마지막 비트가 0인 경우