디지털 논리회로2 프로젝트 제안서

BUS & ALU with Multiplier & Memory

학 과: 컴퓨터정보공학부

담당교수: 공영호 교수님

실습분반: 화 1,2,3

학 번: 2019202050

성 명: 이강현

1. Title & Object
   A. Title

   BUS & ALU with Multiplier & Memory

   B. Object

   이번 프로젝트는 testbench가 BUS를 통해 ALU와 multiplier에 접근하여 논리연산을 진행하고 그 결과를 memory에 저장하는 흐름으로 진행된다.. 따라서 이러한 시스템을 설계하고 이를 검증하는 것이 프로젝트의 목적이다.

2. Component concept
   A. ALU with Multiplier

      ALU with Multiplier는 ALU, Multiplier, Register로 구성되어 있고
      ALU는 Arithmetic Logic Unit으로 CPU에서 덧셈이나 뺄셈과 같은 산술연산이나 AND, OR등의 논리연산을 진행하는 곳이다. Opcode를 통해 operator를 고른 후 operand의 연산에서 사용한다. 총 14가지의 연산을 진행하며 연산 후 Register에 저장하는 역할을 한다.

      Multiplier는 곱셈기로 Register에 저장되어 있는 operandA와 operandB를 곱하는 역할을 하며 2-radix booth algorithm을 사용하여 연산에 32clock을 사용하도록 구성된다.

      Register는 operandA, operandB, opcode, opstart, opdone, opclear, result1,result2에 대한 데이터들을 저장하도록 총 8개의 32bit register로 구성되어있다.
      이번 프로젝트에서는 이를 ALU와 Multiplier가 필요한 작업에 따라 읽기, 쓰기 작업을 하도록 구성된다.
      아래와 같은 핀으로 구성되어있다.

direction	Port name	Bit width	Description
Input	clk	1	Clock
	reset_n	1	Active low reset
	S_sel	1	Select
	S_wr	1	Write/read
	S_addr	8	Address
	S_din	32	Data input
output	S_dout	32	Data output

B. Bus

BUS는 여러 component들간의 data를 전송할 수 있도록 연결해주는 component이다. 1개의 Master(testbench)와 2개의 slave(ALU with Multiplier와 Memory)가 있다. Master가 req(사용요청)을 1로 만들면 grant(허가)를 통해 가능여부를 알려주고 master의 데이터에 따라 slave에 접근하여 데이터를 주고받게 된다.

direction	Port name	Bit width	Description
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	Slave0 select
	S1_sel	1	Slave1 select
	S_wr	1	Slave write/read
	S_addr	8	Slave address

	S_din	32	Slave data input

C. Memory

Address에 기반하여 data를 저장하는 hardware이고 프로젝트에서는 ram(random access memory)을 구현하여 구성한다. Ram은 32비트이고 아래와 같이 구성되어있다.

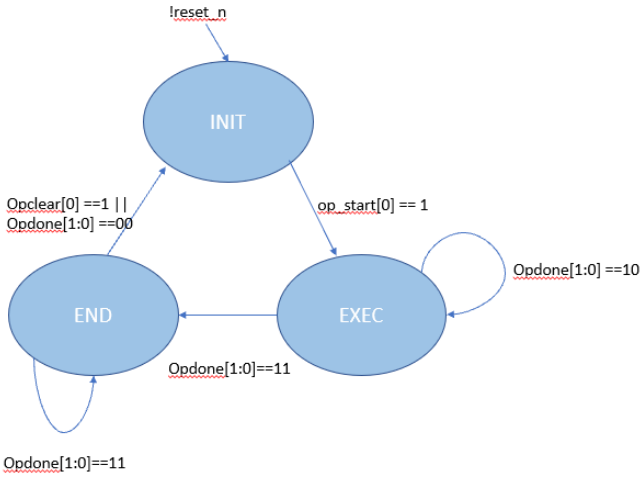
direction	Port name	Bit width	Description
Input	clk	1	Clock
	cen	1	Chip enable
	wen	1	Write enable
	S_addr	5	Address
	S_din	8	Data in
output	S_dout	32	Data output

3. Schedule

	11주차	12주차	13주차	14주차
제안서				
코드 작성				
코드 검증				
결과 보고서				

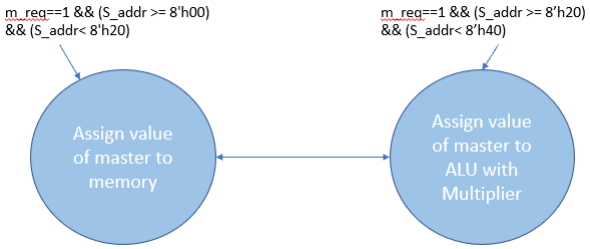
4. State transition diagram

A. ALU with Multiplier



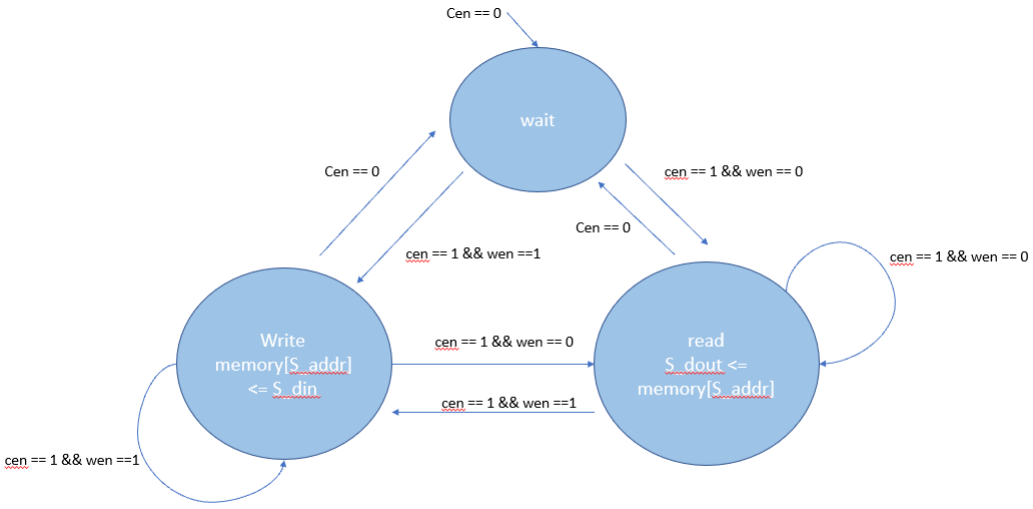
ALU wit Multiplier의 상태 흐름표는 위와같다. 처음에 init 상태를 유지하다가 opstart의 값이 변하면 opdone으로 현재상태를 나타내며 opdone이 11이 되면 연산을 마무리하고 결과를 유지한다. Opclear나 opdone의 초기화가 이루어지면 다시 초기상태로 이동하여 연산을 준비한다.

B. Bus



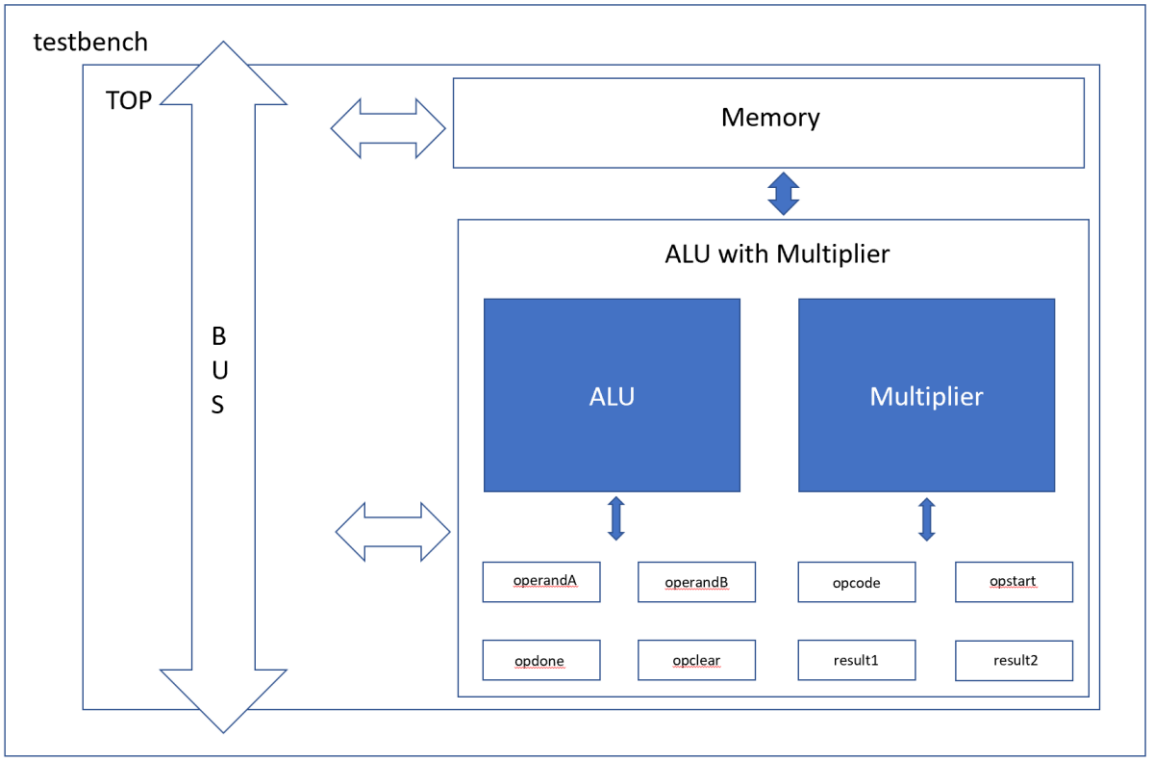
BUS는 입력받은 주소의 범위와 req의 값에 따라 어디와 값을 주고받을지 결정하고 조건이 만족되면 입력값을 할당하게끔 구현한다.

C. Memory



Memory는 ram으로 구현하고 위처럼 chip enable과 write enable에 따라 write할지 read 할지를 결정한다.

5. Module instance design



프로젝트의 전체 모듈들이 어떻게 인스턴스 되어있는지를 도식화 한것이다. Testbench와 데이터를 주고받을 수 있게끔 BUS를 이용하고 그 안에서 ALU와 Multiplier가 8개의 레지스터와 데이터를 주고받고 저장된 데이터를

bus를 통해 testbench는 확인한다. 또한 이러한 결과를 memory에 저장할 수 있다.

6. Design verification strategy

   우선 ALU를 검증한다. ALU의 각각 연산들이 정의되어있는 gates에서 and,or,not등의 논리연산이 잘 연산되고 출력되는지 확인한다. 그 이후 adder와 substractor를 담당하는 cla가 제대로 구현되어있는지 확인해야한다. 이를 모두 검증한 후 ALU에서 주소에 맞게 결과를 얻을 수 있는지 검증한다. 두번째로 Multiplier를 검증한다. Multiplier는 우선 덧셈과 쉬프트연산을 사용하므로 shifter모듈들과 cla를 검증하여야한다. Shifter가 제대로 비트이동을 하는지 검증한다. 그 후 연산을 마무리하는데 16번의 clock cycle을 사용하는지 결과가 잘 나오는지를 검증한다.

   세번째로 ALU와 Multiplier를 함께 ALU with Multiplier모듈에 인스턴스 한 후 register 8개와 데이터를 서로 잘 주고받는지 연산결과가 잘 저장되는지 검증한다.

   네번째로 ram을 검증한다. Ram은 enable신호에 따라 데이터를 잘 읽고 쓰는지 검증한다.

   다섯번째로 BUS를 검증한다. BUS는 req신호에 맞게 데이터를 master에서 잘 가져오는지 우선 검증하고 그 후 주소범위에 맞게 slave를 잘 선택하여 데이터 이동이 이루어지는지 검증한다.

   마지막으로 이 모두를 인스턴스하고 testbench의 입장에서 출력값이 정확히 출력되는지 올바른 타이밍에 나오는지를 생각하며 검증을 완료한다.

7. 예상되는 문제점

   각각의 모듈들은 실습시간에 미리 구현해본 모듈들이고 testbench를 통해 검증을 완료했던 모듈들이기 때문에 결국 프로젝트의 data type과 instance만 적절하게 구성하고 수정한다면 무리없이 작동할 것이다. 하지만 state에 따라 적절하게 구현하지 못하거나 state를 사용하지 않고 behavior하게 구현한다면 적절한 조건문에 분기에 따라 값을 할당하지 못한다거나 clock을

고려한 상태변화를 제대로 이해하지 못한채 구현한다면 올바른 결과값이나 적절한 타이밍에 결과값을 산출해내지 못할 것 같다.
결국 주어진 pin들을 정확하게 연결하고 모듈 수정 혹은 연결시 검증이 마지막 top모듈 검증까지 지속적으로 이루어져야 오류없이 시스템을 구현해낼 수 있을 것 같다.