디지털 논리회로2

Term project 제안서

제출일자: 2018년 11월 11일 (일)

학 과: 컴퓨터정보공학부

담당교수: 이준환 교수님

학 번: 2017202087

성 명: 홍 세 정

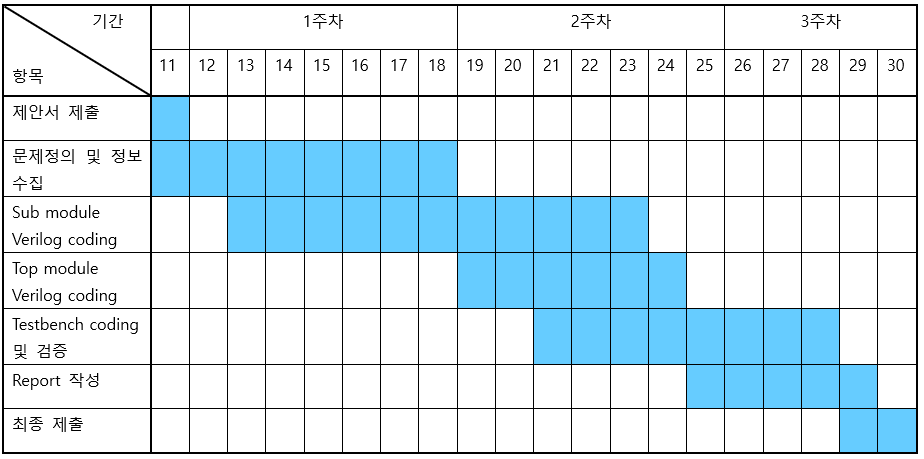
1. 과제제목

Multiplier, Adder, FIFO(first in first out), RF(register file)를 이용하여 Matrix를 설계하고, bus를 통해 Memory와 연결하여 검증.

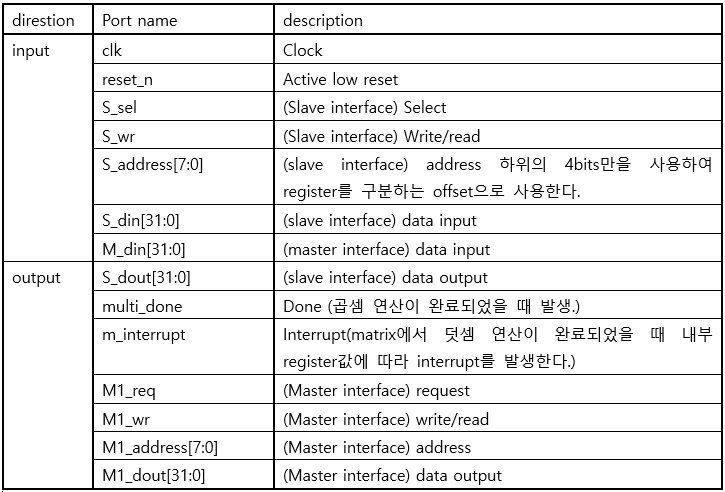
1. 과제목표

Multiplier, Adder, FIFO, RF를 이용하여 Matrix를 설계하여 연속적인 연산들을 진행할 수 있다. 또한 testbench를 master로 하여 입력 값들을 저장하고 연산을 출력할 수 있다.

Matrix, Memory 그리고 Bus를 이용하여 Top module을 구현할 수 있다.

1. 일정
2. 각 module 별 구현 방법(state transition diagram 또는 내부 register map)

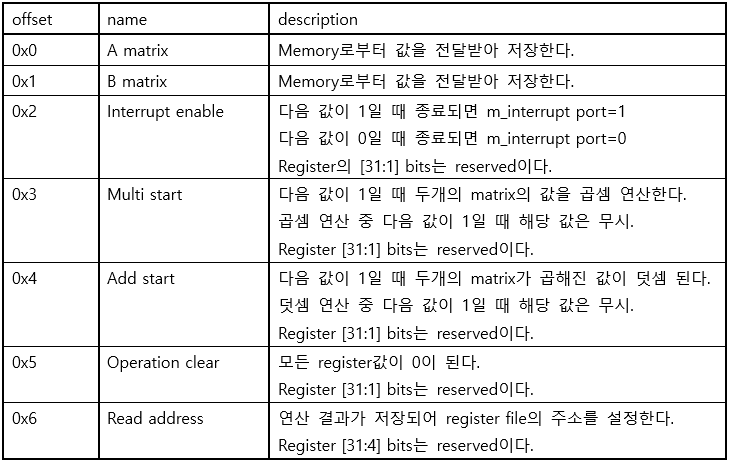
Matrix, Memory, Register File, FIFO, BUS module이 있다.

* matrix (= A Matrix와 B Matrix를 곱하여 결과값을 도출하는 hardware이다.)

matrix module의 input/output을 나타낸 것이다.



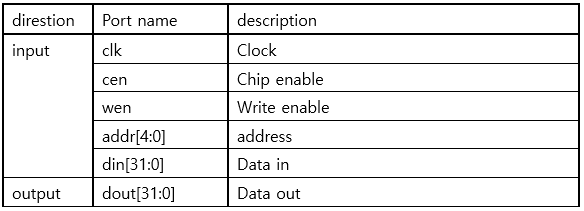
FIFO, Multiplier, Adder, Register File을 이용하여 Matrix를 구현할 수 있다.

matrix의 register이다.



다음과 같이 state를 구성할 수 있다.

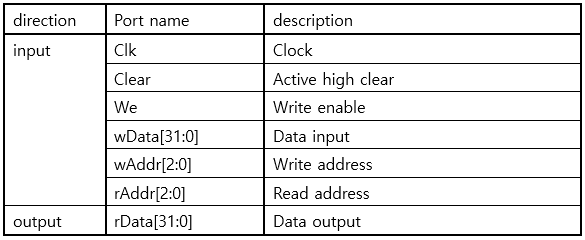
* Memory (= address에 기반하여 data를 저장하는 hardware로, 해당 프로젝트에서는 random access memory(ram)을 구현하도록 한다.)



memory module의 input/output을 나타낸 것이다.

Cen과 wen이 모두 1이면 address가 가리키는 memory에 din을 write한다. 이때는 dout은 0을 출력한다.

Cen이 1, wen이 0이면 address가 가리키는 memory의 값을 dout에 write한다.

* Register File (= 읽고 쓸 수 있는 register들로 구성되며 address에 따라 접근할 수 있는 저장 장치이다.

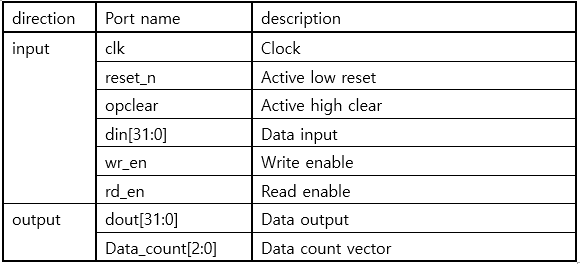
Register file module의 input/output을 나타낸 것이다.

A register file in composed of a set of registers that can be read and written by supplying a register number(address) to be accessed.

Write operation, register, read operation으로 구성할 수 있다.

Register file은 동시에 read와 write를 여러 요청을 처리할 수 있다.

* FIFO (= 시간과 우선 순위와 관련된 데이터를 정리하고 이용하는 방식으로 Matrix 내부로 들어가게 된다.)



나중에 집어넣은 데이터가 먼저 나오는 스택과 반대되는 개념이다. 선입선출의 자료 구조 데이터가 들어오는 위치는 가장 뒤에 있고, 데이터 나가는 위치는 가장 앞에 있어서, 먼저 들어오는 데이터가 먼저 나가고 나중에 들어온 데이터는 가장 나중에 나간다.

여기서 queue의 구조를 이용하여 module FIFO를 구현한다.



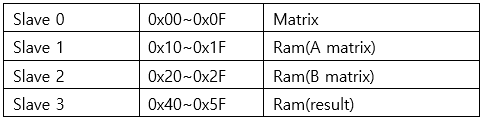
INIT-초기값이다. 초기값은 head=0, tail=0, empty=1이다. 초기값 이외에는 next\_state로 안 들어온다.

READ: (=dequeue)말그대로 input을 read하는 state이다.

WRITE : (enqueue)data 값을 써주는 작업을 하는 state이다.

NO\_OP: 실행할 수 없는 값을 받았을 때의 state이다. Rd\_en, wr\_en은 동시에 같은 값을 받을 수 없다. 반드시 실행할 때 rd\_en, wr\_en 중 하나가 1이 들어오게 되어 read를 하거나 write하거나 둘 중 하나의 작업을 해야 한다.

* BUS (= 여러 component들 간에 data를 전송할 수 있도록 연결해주는 component이다, 새로운 component들을 추가하기 쉽고, 가격이 저렴한 특징을 가지고 있다.)

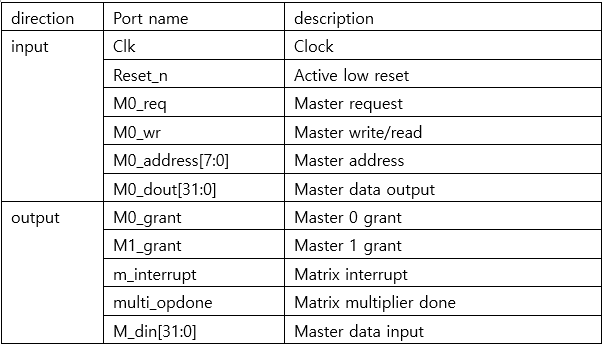
다음 표와 같은 memory를 가진다.

BUS는 grant에 따라 master를 결정해주며, grant에 따른 master만 bus를 사용할 수 있다.

Master는 bus를 통해 data를 transfer하고자 할 때, 자신에게 해당하는 request signal을 1로 한 이후에, 그에 대한 확인으로 grant signal을 받은 후 data transfer를 올바르게 할 수 있다.

Master가 grant signal을 받은 후 request signal이 1인 동안에는 bus의 소유권을 빼앗기지 않고 data transfer를 계속 할 수 있다.



* TOP (= Matrix, RAM 그리고 BUS를 instance하여 이들을 연결한 component이다.)



1. 예상되는 문제점

Testbench로부터 memory에 2행 2열의 행렬 값을 serial하게 저장할 수 있어야한다.

A Matrix와 B Matrix를 곱할 때 행렬을 잘 이해하고 곱할 수 있어야 한다.

각 input을 이해하여 해당 input이 들어올 때 어떤 value를 출력해야 하는지 이해할 수 있어야한다.

과제에서 수행한 FIFO와 비트수가 다르므로 알맞게 비트 수를 조정해야 한다.

각 module을 연결할 때 bit 수를 알맞게 맞춰주어 오류가 없게 해야 한다.

너무 많은 pin에 의한 write와 register 사용에 불필요한 것이 없게 해야 한다.

Clk의 주기를 잘 맞춰주어 over되는 경우가 없게 한다.

Top module에서 전체 시스템의 memory map이 정확하게 나오는지 확인해야 한다.

1. 검증전략

Matrix에서 A Matrix와 B Matrix가 FIFO를 통해 먼저 들어오는 데이터가 먼저 나가고 나중에 들어온 데이터가 나중에 나가는지 확인할 수 있다.

A Matrix와 B Matrix가 multiplier을 통해 곱해진 값이 정확히 나왔는지 검증할 수 있어야한다.

각 STATE일 때 NEXT\_STATE가 어디인지 확인하고 해당 register에 저장하는지 확인할 수 있다.

Input이 알맞게 들어왔을 때 ADD연산과 multiplier연산을 하는지 확인할 수 있다.

BUS master에 대한 요청이 slave중 한 개를 선택하여 신호가 정상적으로 전달되는지 확인한다.

Memory에서 cen과 wen에 값이 들어가면 write / read 하는 것을 확인한다.

Top에서 최종 result가 알맞게 나오는 것을 확인한다.