Implementation of the Datapath
IIT3004 Digital System
2019193015 박준하

I. Goal

Write Verilog source code for Datapath and test it

II. Implementation

A. Datapath module

실제 구조와 같이 Register file module 과 Function unit module 을 sub module 로 구성하고 그 사이 wire, bus 들만 Datapath module 에서 작성하였다.
Control word 는 다음과 같이 wire 를 assign 하였다.

```
assign DA = CTRWRD[15:13];
assign AA = CTRWRD[12:10];
assign BA = CTRWRD[9:7];
assign MB = CTRWRD[6];
assign FS = CTRWRD[5:2];
assign MD = CTRWRD[1];
assign RW = CTRWRD[0];
```

i. Register file module

DAtemp 와 RWtemp Register 를 두어 다음 Clock 에서 실행될 때에도 원하는 register 에 작성할 수 있도록 하였다.

ii. Function unit module

Arithmetic module 과 Logic module 을 하나의 case statement 로 구현하였다.

B. Testbench module

i. Target Instruction

	Micro-operation	DA	AA	BA	MB	FS	MD	RW
1	$R1 \leftarrow R2 - R3$	$R1$	$R2$	$R3$	Register	$F = A + \overline{B} + 1$	Function	Write
2	$R4 \leftarrow$ sl R6	$R4$	—	$R6$	Register	$F =$ sl B	Function	Write
3	$R7 \leftarrow R7 + 1$	$R7$	$R7$	—	Register	$F = A + 1$	Function	Write
4	$R1 \leftarrow R0 + 2$	$R1$	$R0$	—	Constant	$F = A + B$	Function	Write
5	Data out $\leftarrow R3$	——		$R3$	Register	—	—	No Write
6	$R4 \leftarrow$ Data in	$R4$	——	—	—	—	Data in	Write
7	$R5 \leftarrow 0$	$R5$	$R0$	$R0$	Register	$F = A \oplus B$	Function	Write

1. Corresponding stimulus

```
#PERIOD CTRWRD = 16'b0010100110010101;
#PERIOD CTRWRD = 16'b1000001100111001;
#PERIOD CTRWRD = 16'b1111110000000101;
#PERIOD CTRWRD = 16'b0010000001001001;
#PERIOD CTRWRD = 16'b0000000110000000;
#PERIOD CTRWRD = 16'b1000000000000011;
#PERIOD CTRWRD = 16'b1010000000101001;
```

ii. Additional instruction

Overflow detection 을 확인하기 위해 다음과 같은 instruction 을 구성하였다.

R0 <- Data in, Data in: 0xefff

R1 <- Data in, Data in: 0xeffe

R2 <- R0+R1

1. Corresponding stimulus

```
#PERIOD Din = 16'hefff;
CTRWRD = 16'b0000000000000011;
#PERIOD Din = 16'heffe;
CTRWRD = 16'b0010000000000011;
#PERIOD CTRWRD = 16'b0100000010001001;
```

C. Overall process (Source code 는 따로 첨부하였습니다.)

1. @posedge CLK (Register File unit)
   DAtemp 값을(이전 Clock 의 DA)이용하여 bus D 의 값을 register 에 쓴다.(RW 에 따라 쓸지 말지 결정)
   이후 조건을 이용하여 이번 Clock 의 DA 와 RW 를 temporary register 에 불러들인다.
   그리고 AA, BA 로 Register 로부터 값을 읽어오도록 하였다.

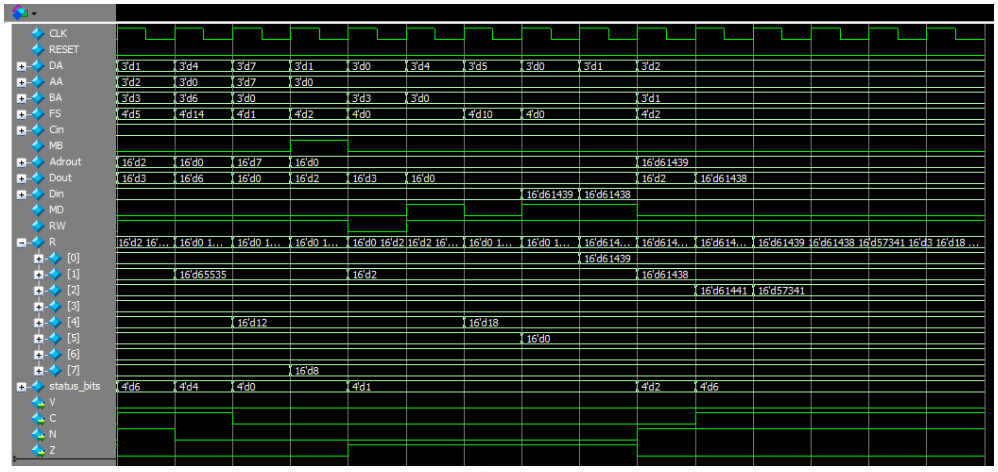
2. MB 값을 보고 Constant input 을 받을지 선택한다.(Datapath module)

3. busA, busB 의 값으로 Functional process 를 수행한다. 그 값을 output register 에 쓴다.(Function Unit module)

4. output register 의 값과 Carrier 를 이용하여 V, C, N, Z 를 계산한다(Function Unit module)

5. MD 값을 보고 output register 의 값과 Data in 중 어느 값을 받을지 선택하고 busD 에 불러들인다.(Datapath module)

III. Result



편의를 위해 Register 를 reset 하고 초기값을 대입하는 부분은 잘라냈다.
(각 Register 의 초기값 R0 = 0, R1 = 1, … R7 = 7)
기대한 바와 같이 모두 잘 동작하는 모습을 볼 수 있다.