Assignment 4 : MIPS implementation by VHDL

By **IP Catalog**, we generate a block memory with -
1. Memory Type = Single Port RAM
2. Native Interface
3. Write Width = 32
4. Read Width = 32
5. Write Depth = 4096
6. Read Depth = 4096
7. Operating Mode = Write First
8. Enable Port Type = Always Enable
9. Total Port A read latency : 1 Clock Cycle(s)
10. Address Width A: 12

reg1 array of std_logic_vector(32 downto 0) of size 32 is used to store the values of all 32 registers according to their corresponding indices.

We defined 7 states that change in each other on every rising edge of the clock.

Components:
blk_mem_gen_0 : component that connects the memory with the instructions of coe file. It has 4 input ports (clka, wea, addra, dina) and one output port (dout).

Segment: component to light up the 4-digit seven segment.

It has 6 input ports:

1. clk : clock with fundamental frequency.

2. en1 : single bit port to determine whether PC should be printed or the last 16 bits of output.

3. an0, an1, an2, an3 : anode ports for all anodes.

4. bitset1, bitset2, bitset3, bitset4 : each 4-digit bit vector corresponding to the segments to be printed.

Two Port Maps are used :

Initially system is in idle state,

Idle State : change en1, itype, rtype to 0, addra to 16-bit-vector of count(PC), and change the state to read.

Read state :

here we check if douta is a zero vector with 32 bit, change state to endstate. Else if douta (31downto 26) is “000000”, then it is R type, else if it is “100011” then it is I type, else change state to Idle.

If it is R type, assign values to irsc1, irsc2, rdes, shamt, funct and rtype. If the value of shamt is non zero than jump state to shiftoper.

IF it is I type, assign values to ifist, irs, ird, iaddress1 and change state to updatir.

Updateir : assign value of address to irs1 and change state to **change.**

Shiftoper: if funct = 0 then change state to **shiftrighthelper.** Else if funct = 2 then change state to **shiftlefthelper.**

Shiftrighthelper: it works recursively. Shift reg1(rsrc2) to right by 1 bit and assign rsrc2 to rdest.
If shamt is greater than 1, than decrease it by one and call the same state again.
Else update sevensegment1 by reg1(rdest) and change state to **idle.**

change: if douta(5 downto 0) is “100000” then it is “add” instruction. Add reg1(rsrc1) and reg1(rsrc2) and assign the result to reg1(rdest). Also update sevensegment1. Change state to **idle.**
If douta (5 downto 0) is “100010” then it is sub instruction. Subtract reg1(rsrc1) and reg1(rsrc2) and assign the result to reg1(rdest). Also update sevensegment1. Change state to **idle.**

Else if itype is 1 then we check, ifirst is whether 35 or 43.
if it is 35, then it is lw instruction. Else sw instruction. Assign addra by 12-bit vector of irs1 and in case of lw, change state to s1 or in case of sw, update dina by reg1(ird) and update sevensegment1 and also change state to idle.

S1: update reg1(ird) by douta and update sevensegment1 and change state to **idle.**

Endstate: assign en1 to 1. now if input en2 is 0 then pass 16-bit logic vector of count to sevensegment2 else pass sevensegment1 to sevensegment2.

Binary input for MIPS instructions:
for shift left by 4 bit:

op	rs	rt	rd	shamt	funct
0	0	16	10	4	0

for shift right by 4 bit:

op	rs	rt	rd	shamt	funct
0	0	16	10	4	2

MIPS machine language

Name	Format	Example						Comments
add	R	0	18	19	17	0	32	add $s1,$s2,$s3
sub	R	0	18	19	17	0	34	sub $s1,$s2,$s3
addi	I	8	18	17	100			addi $s1,$s2,100
lw	I	35	18	17	100			lw $s1,100($s2)
sw	I	43	18	17	100			sw $s1,100($s2)
Field size		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions are 32 bits long
R-format	R	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	I	op	rs	rt	address			Data transfer format

.coe file :

memory_initialization_radix=2;

memory_initialization_vector=

00000010010010001000100000100000 ; ADD $s1 $s2 $t0

00000000000100000101000100000000 ; SLL $t2 $s0 4

00000000000100000101000100000010 ; SRL $t2 $s0 4

00000010010010001000100000100010 ; SUB $s1 $s2 $t0

10101110111011110000000000001100 ; SW $t7 12($s7)

10001110111101010000000000001100; LW $s5 12($s7)

Manoj Kumar 2018CS50411

Nikita Bhamu 2018CS50413