CSE 331 – COMPUTER ORGANIZATION HOMEWORK 4 REPORT

Name Surname : Emirhan UZUN
Number : 171044019

REQUESTED

In this project, you will use Altera Quartus II with Verilog. You will implement a different version of 32-bit MIPS processor. The block that you will design will get no inputs from outside. You will have two memories: Data Memory and Instruction Memory. The instructions must be loaded to the instruction memory and the data must be put in data memory. You will support **lw, sw, j, jal, jr, beq**, **bne**, **addn, subn, xorn, andn, orn, ori** and **lui** instructions**.**

I just could some of these and I'll show below one by one :

1 – ALU Testbench

```
ALU actb (out2,out1,out0,functionfield,op1,op0);

initial begin
functionfield = 6'b100100; op1 = 1'b1; op0 = 1'b0;

end
```

It takes the 6 bit code and 2 bit opcode. After that it gives us to output 3 bit.

The results are :

```
VSIM 6> step -current
# time =  0, f.field =100100,opcode=10 aluctr=011
```

2 – Control Unit Testbench

This is my control unit for control unit.

	opcode5	opcode4	opcode3	opcode2	opcode1	opcode0	RegDst	Branch	MemRead	MemtoReg	MemWrite	ALUSrc	RegWrite	Jump	JAL	JR	ALUOp1	ALUOp0
lw	1	0	0	0	1	1	0	0	1	1	0	1	1	0	0	0	0	0
sw	1	0	1	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0
j	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	x	x
jal	0	0	0	0	1	1	0	0	0	0	0	0	1	0	1	0	x	x
jr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	x	x
beq	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1
bne	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1
addn	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0
subn	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0
xorn	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0
andn	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0
orn	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0
ori	0	0	1	1	0	1	0	0	0	0	0	1	1	0	0	0	1	1
lui	0	0	1	1	1	1	0	0	0	0	0	1	1	0	0	0	0	0

And these are the booleans expressions :

RegDst = p2'p1'p0'
Branch = p3'p2
MemRead = p5p3'
MemtoReg = MemRead
MemWrite = p5p3
ALUSrc = p5 + p3p2
RegWrite = p3'p2'p1p0 + RegDst + p3p2
Jump = p2'p1p0'
JAL = p5'p2'p1p0
ALUOp1 = RegDst + p3p1'
ALUOp0 = p2p1'

Here is my test brench :

```
initial begin
p = 6'b000101;
#`DELAY;
end
```

The results are :

```
# time =  0, opcode =000101,RegDst=0,Branch=1,MemRead=0,MemtoReg=0,MemWrite=0,ALUSrc=0,RegWrite=0,Jump=0,JAL=0,ALUOp=01
```

3 – Full Adder (32 bit) Testbench

```
`define DELAY 20
module full_adder_testbench();
reg [31:0] a, b;
reg  carry_in;
wire [31:0]sum;
wire carry_out;

_32bit_adder adder(sum, carry_out, a, b, carry_in);

initial begin
a = 32'd15; b = 32'd5; carry_in = 1'b0;

end


initial
begin
$monitor("time = %2d, a =%d, b=%d, carry_in=%1b, sum=%d, carry_out=%1b", $time, a, b, carry_in, sum, carry_out);
end

endmodule
```

This is the test case which is sum of 15 + 5 in decimal. Carry in is zero.

This is the result of full adder testbench.

```
add wave -position insertpoint  \
sim:/full_adder_testbench/a \
sim:/full_adder_testbench/b \
sim:/full_adder_testbench/carry_in \
sim:/full_adder_testbench/carry_out \
sim:/full_adder_testbench/sum
VSIM 12> step -current
# time =  0, a =        15, b=         5, carry_in=0, sum=        20, carry_out=0
```

4 – Or 32 bit testbench

```
initial begin
a = 32'b00100101011010101010110010111111; b = 32'b01100110001001010100100011111000;
#`DELAY;
a = 32'b10101110100101010101111010101010; b = 32'b10101010001101011100101010101010;
```

This takes 2 32 bit number and find the result which is (A or B). The result is

```
# time =  0, a =00100101011010101010110010111111, b=01100110001001010100100011111000,result=01100111011011111111011001111111
# time = 20, a =10101110100101010101111010101010, b=10101010001101011100101010101010,result=10101110101101011101111010101010
```

5 – XOR 32 bit testbench

```
a = 32'b00101011010010101010000010111111; b = 32'b01100100101110101000001011111000;
#`DELAY;
a = 32'b10101000101011101010101010101010; b = 32'b10101101011010110101001010101010;
```

This takes 2 32 bit number and find the result which is (A xor B). The result is

```
# time =  0, a =00101011010010101010000010111111, b=01100100101110101000001011111000,result=01001111111100000010001001000111
# time = 20, a =10101000101011101010101010101010, b=10101101011010110101001010101010,result=00000101110001011111100000000000
```

6 – Is Equal 32 bit testbench

```
initial begin
a = 32'b00101001110101110101010010111111; b = 32'b01100000101011010101000011111000;
#`DELAY;
a = 32'b11111111111111111111111111111111; b = 32'b11111111111111111111111111111111;
```

This takes 2 32 bit number and find the result which is (A equals B). The result is

```
# time =  0, a =00101001110101110101010010111111, b=01100000101011010101000011111000,result=0
# time = 20, a =11111111111111111111111111111111, b=11111111111111111111111111111111,result=1
```

7– And 32 bit testbench

```
initial begin
a = 32'b00101000101001011011001000111111; b = 32'b01100110000110101000000011111000;
#`DELAY;
a = 32'b11110101100010001010101010101010; b = 32'b10101011101010010001101010101010;
```

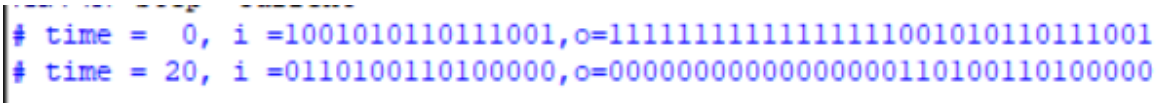
This takes 2 32 bit number and find the result which is (A and B). The result is

```
VSIM 28> step -current
# time =  0, a =00101000101001011011001000111111, b=01100110000110101000000011111000,result=00100000000000001000000000111000
# time = 20, a =11110101100010001010101010101010, b=10101011101010010001101010101010,result=10100001100010000000101010101010
```

8 – Sign Extender 32 Bit Testbench

```
in = 16'b1001010110111001;
#`DELAY;
in = 16'b0110100110100000;
end
```

It takes a 16 bit number and extend to 32 bit according to most significant bit of number. If msb is 0, then extend with 0, otherwise extend with 1. The results of above tests are :



I have 8 different testbench and all of them work correctly. I couldn't complete the entire homework because there were too much exams and assignments. But our teacher and assistant said, "Even if you cannot do the entire homework, send the places you have done.". So I have sent all the instructions I have done by attaching them to my homework file and report. The list of my instructions are :

1 – Full Adder

2 – And

3 – Or

4 – Xor

5 – Is equal

6 – Sign Extender

7 – ALU

8 – Control Unit