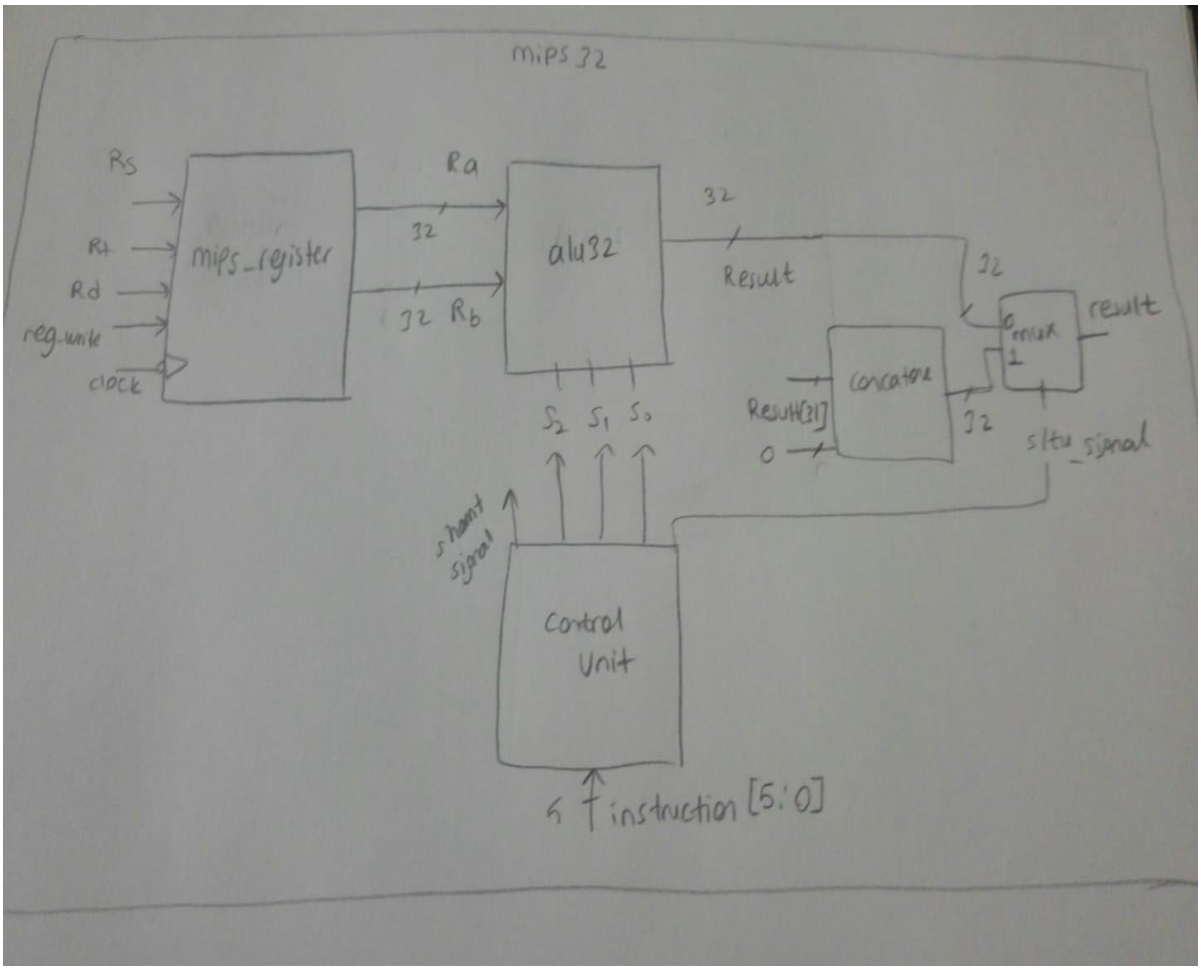
CSE 331 Computer Organization Project 3
R-Type Single Cycle Mips with Structural Verilog

1. module mips32(instruction,clock,result);



Mips32 is the top level entity.Mips32 module takes two inputs and one output paramaters.Firstly mips_register module is called so register module convert 32 bit rA and rB then these are sent to alu32 module to process right operation and get the result.Alu32 uses three select bits for 8x1 mux.It must take ALUOP bits. Control Unit module is wrote for that need.It takes 5 bit function code from the instruction then it gives an output 3 bits for select bits of alu32.

After the result output from the alu32, for controlling the sltu signal , there is concatane module.It takes a most significant bit of the result and 0 for input and then that module concatane them.After that operation , 2x1 mux is used for set less than operation.It takes two input ,for 1 input is concatane module's output , for 0 input is result output from the alu32 module.Mux that is talked about, uses

a selection bit for sltu instruction.There is a control signal for that from the control unit.

2.module mips_registers(read_data_1, read_data_2, write_data, read_reg_1, read_reg_2, write_reg, signal_reg_write, clk);

mips_register module is wrote for taking an adress for the rs,rt and rd than assign them their values from the register.mem file.
read_data_1 and read_data_2 is output for rs and rt.
Write_data is output for rd.
read_reg_1, read_reg_2, write_reg are adresses of rs ,rt and rd.
Signal_reg_write is a signal for writing the content of the result to rd register's adress.If the signal is 1 and write adress is not $zero(00000),data can write there.

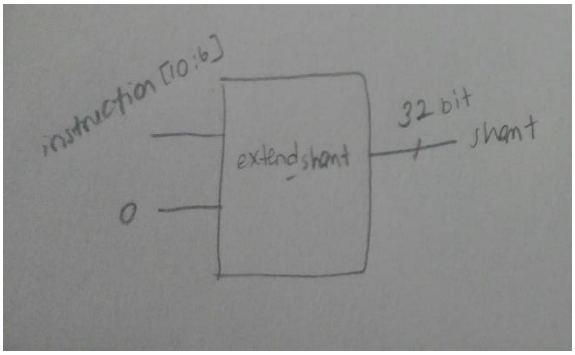
3.module control_unit(select_bits_ALU, function_code,Shamt_signal,sltu_signal,selectbit_sr);

Control unit module is designed to generate 3 bit select for alu32.It takes 5 bit function code from the instruction (instructin[0:5]) then it convertes them to 3 bit with using and or gates.Also it generates a shamt_signal and sltu_signal and selectbit_sr.These are control signals.
shamt_signal : it checks the function code belongs to srl instruction or sll.If it is srl or sll shamt_signal is 1 otherwise it is 0.
sltu_signal : it checks the function code belongs to sltu instruction.If it is sltu sltu_signal is 1, otherwise it is 0.
selectbit_sr : This signal is always 0.

4. module concatane_32_bit (A,B,S);

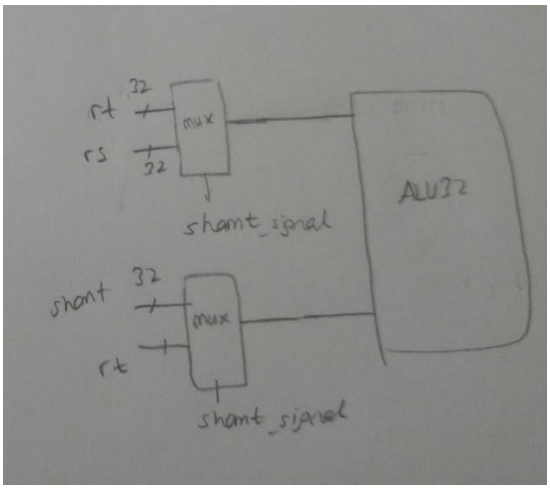
Concatane module is takes a zero bit and one bit to concatane.Result is 32 bit. For that used and gates.

5. module extend_shamt(shamt,extend_bit,S);



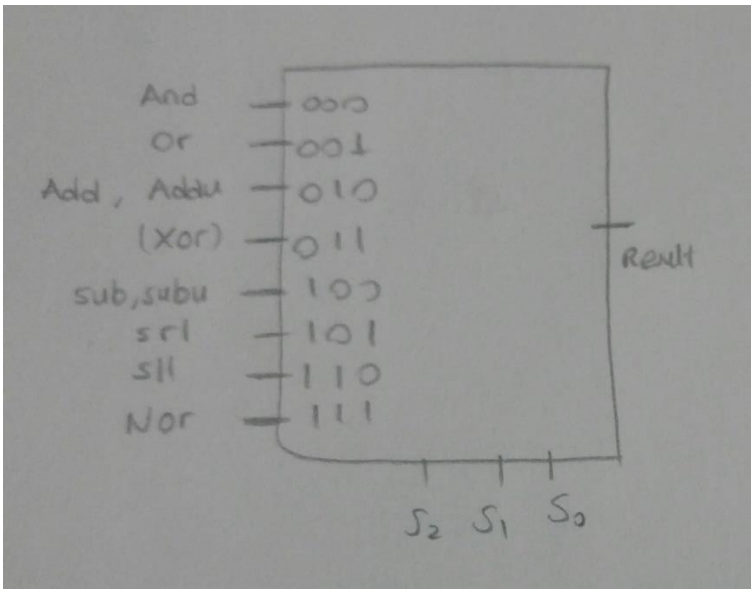
Module is designed for extend the 5 bit shamt with zero bit.S is the 32 bit output.

6. module alu32(ALUOP,A,B,C,V,Z,Y,shiftSig);



Alu32 module is designed for the assignment2. There is one difference between new alu32 is shiftSig input for controlling the shift amount for srl. Because before design there was a sra operation. Now instruction is srl and change is needed.There is a 2x1 mux for that.mux is select a select bit for shift right logic module.If the shiftSig is 1 select bit is 1 , if it is 0 it is 0.

Also alu32 module's inputs are selected with the shamt signal.If the signal is 1 First mux is select rt instead of rs.Second mux is select 32 shamt if the signal is one.



Mux is shown above was design for the assignment 2 ,now the same 8x1 mux is used.Additionally,there are 10 R type instruction's results for input.There is no change.

MODELSIM SIMULATION RESULT

Transcript

File Edit View Bookmarks Window Help

Transcript

```
# opcode = 000000, rs = 10000, rt = 10001, rd = 10001, shamt = 00000, funct = 100000 ,shmat== 0,sltu=0 ,index==  1
# $rs = 00000000000000000000000000010000
# $rt =00000000000000000000000000110001
# result = 00000000000000000000000001000001
# opcode = 000000, rs = 10100, rt = 10101, rd = 10011, shamt = 00000, funct = 100001 ,shmat== 0,sltu=0 ,index==  2
# $rs = 00000000000000000000000000010100
# $rt =00000000000000000000000000010101
# result = 00000000000000000000000000101001
# opcode = 000000, rs = 01001, rt = 01010, rd = 01101, shamt = 00000, funct = 100111 ,shmat== 0,sltu=0 ,index==  3
# $rs = 00000000000000000000000011110000
# $rt =00000000000000000000111100000000
# result = 11111111111111111111000000001111
# opcode = 000000, rs = 01000, rt = 01001, rd = 01010, shamt = 00000, funct = 100101 ,shmat== 0,sltu=0 ,index==  4
# $rs = 00000000000000000000000000001111
# $rt =00000000000000000000000011110000
# result = 00000000000000000000000011111111
# opcode = 000000, rs = 10110, rt = 10111, rd = 01111, shamt = 00000, funct = 101011 ,shmat== 0,sltu=1 ,index==  5
# $rs = 00000000000000000000000000010110
# $rt =00000000000000000000000000010111
# result = 00000000000000000000000000000001
# opcode = 000000, rs = 00000, rt = 10100, rd = 10001, shamt = 00010, funct = 000000 ,shmat== 1,sltu=0 ,index==  6
# $rs = 00000000000000000000000000010100
# $rt =00000000000000000000000000000010
# result = 00000000000000000000000001010000
# opcode = 000000, rs = 00000, rt = 10000, rd = 10000, shamt = 00011, funct = 000010 ,shmat== 1,sltu=0 ,index==  7
# $rs = 00000000000000000000000000000000
# $rt =00000000000000000000000000000011
# result = 00000000000000000000000000000000
# opcode = 000000, rs = 10000, rt = 10001, rd = 11000, shamt = 00000, funct = 100010 ,shmat== 0,sltu=0 ,index==  8
# $rs = 00000000000000000000000000000000
# $rt =00000000000000000000000001010000
# result = 11111111111111111111111110110000
# opcode = 000000, rs = 10110, rt = 01000, rd = 11001, shamt = 00000, funct = 100011 ,shmat== 0,sltu=0 ,index==  9
# $rs = 00000000000000000000000000010110
# $rt =00000000000000000000000000001111
# result = 00000000000000000000000000000111
# opcode = 000000, rs = 10101, rt = 11001, rd = 01000, shamt = 00000, funct = 100100 ,shmat== 0,sltu=0 ,index== 10
# $rs = 00000000000000000000000000010101
# $rt =00000000000000000000000000000111
# result = 00000000000000000000000000000101
#   10 tests completed.
```