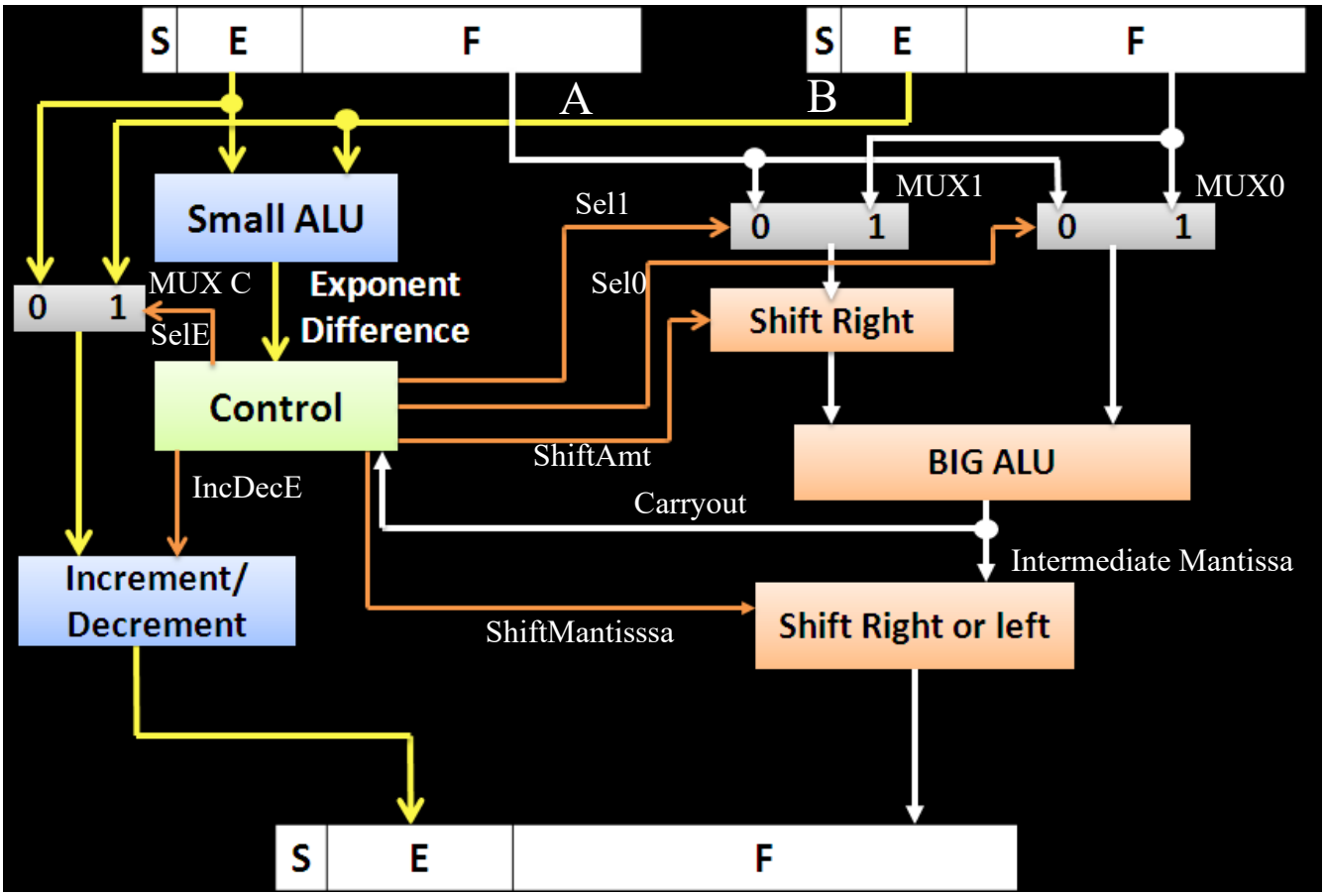
Experiment No 9: Implementation of Floating point Adder

The aim of this experiment is to implement a floating point Adder circuit and synthesize to check LUT usage and delay of the circuit.



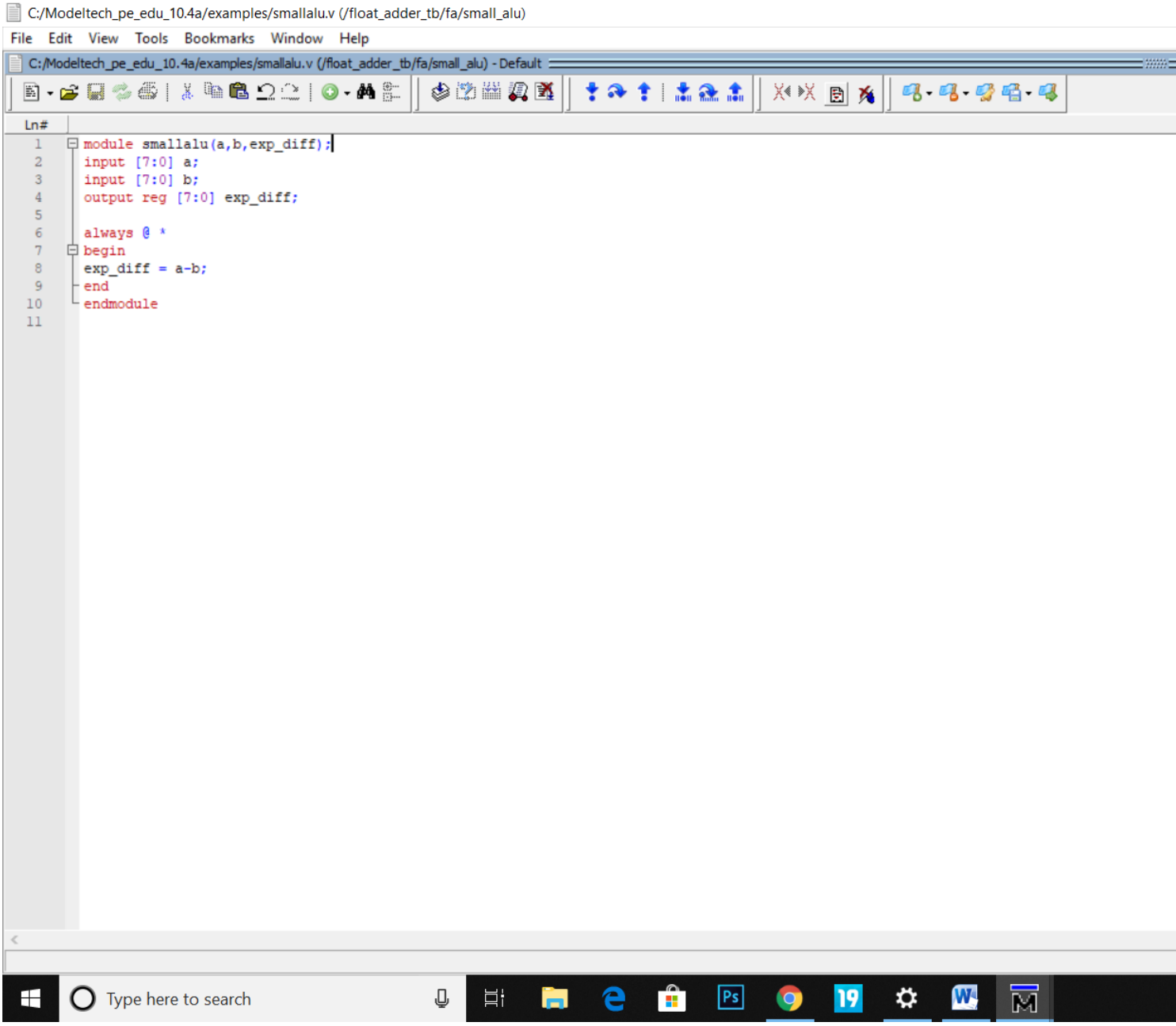
<u>Exercise 9.1 Implement a floating point adder. Assume the floating point numbers are in IEEE 754 single precision format. Assume both the inputs will have same sign.</u>

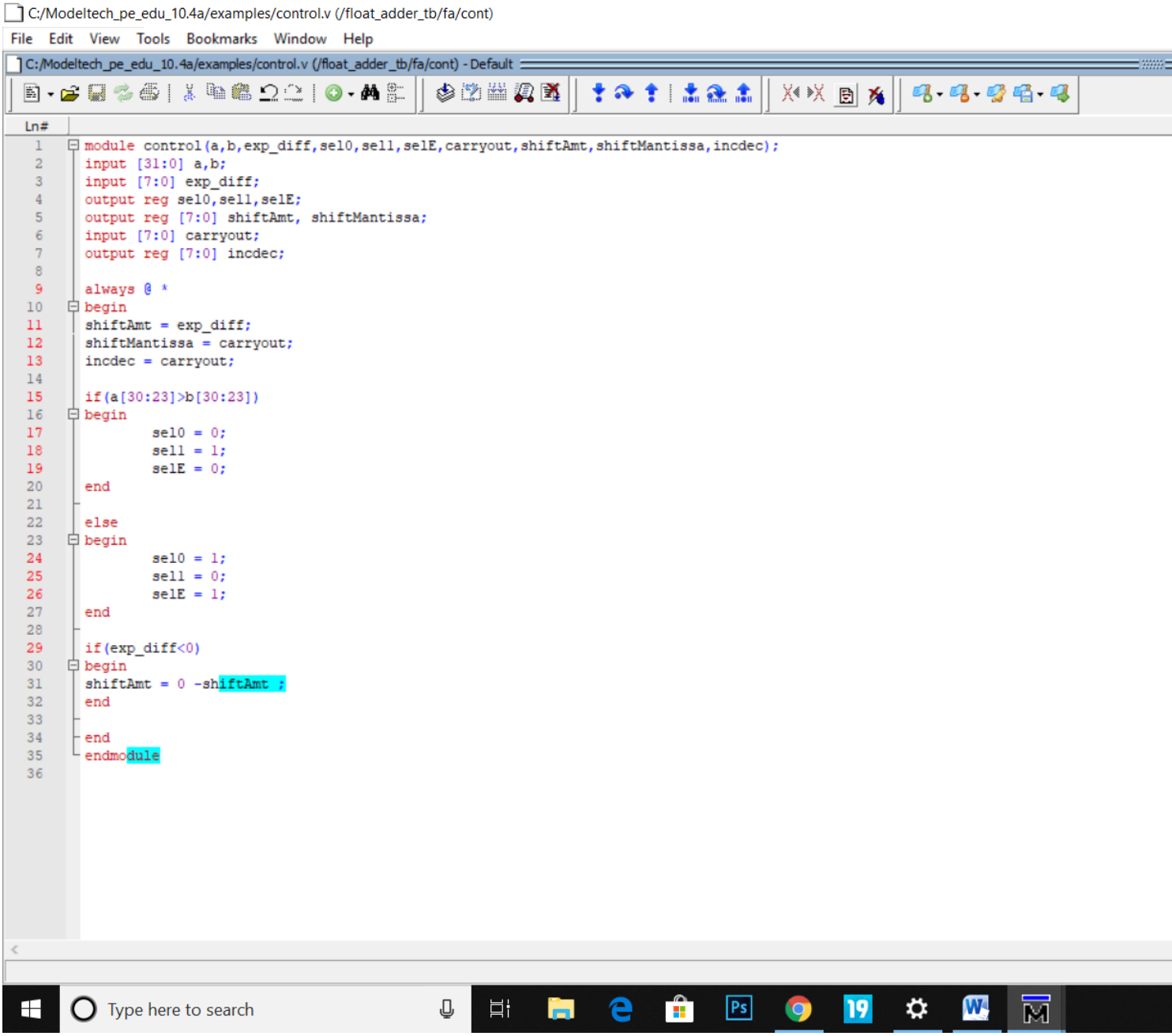
Any other assumptions made should be clearly specified here

Assumptions: Sign of both the numbers is same i.e. both are positive or both are negative. There is no overflow or underflow.

Q9.1. Implement the submodules which are necessary for implementation of floating point addition and copy the <u>images</u> of those Verilog codes here.

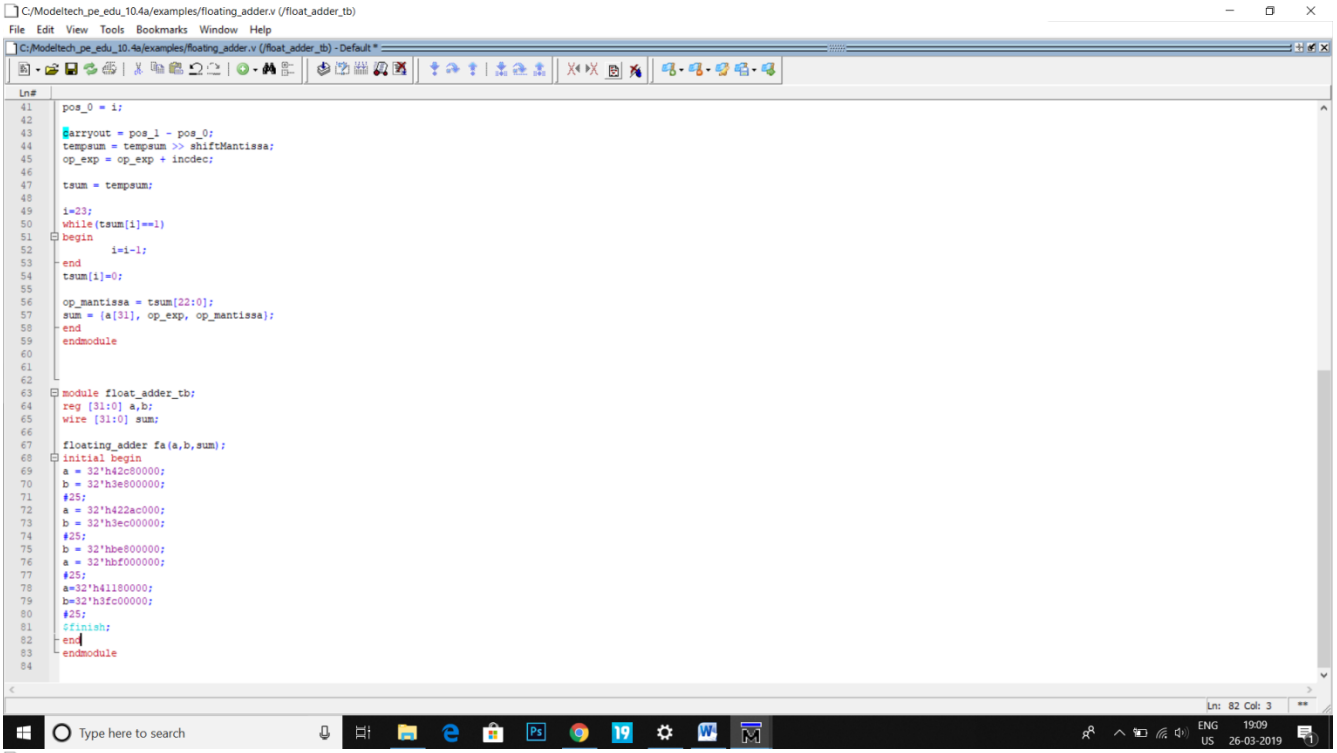
Answer:


C:/Modeltech_pe_edu_10.4a/examples/smallalu.v (/float_adder_tb/fa/small_alu)
File Edit View Tools Bookmarks Window Help
C:/Modeltech_pe_edu_10.4a/examples/smallalu.v (/float_adder_tb/fa/small_alu) - Default
Ln#
1 module smallalu(a,b,exp_diff);
2 input [7:0] a;
3 input [7:0] b;
4 output reg [7:0] exp_diff;
5
6 always @ *
7 begin
8 exp_diff = a-b;
9 end
10 endmodule
11
Type here to search

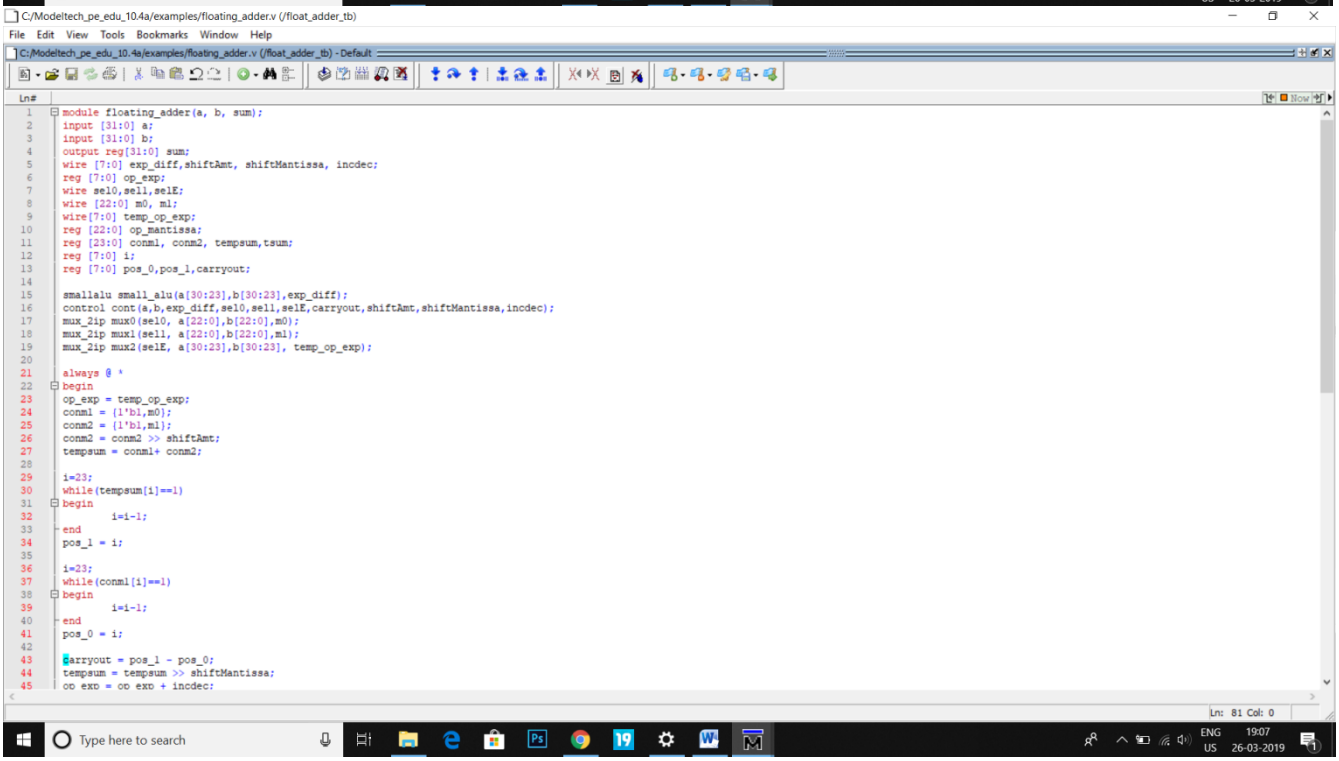


Q9.2. Implement complete floating point adder in Verilog (by instantiating all the submodules). Copy the image of Verilog code of the floating point adder here.

Answer:



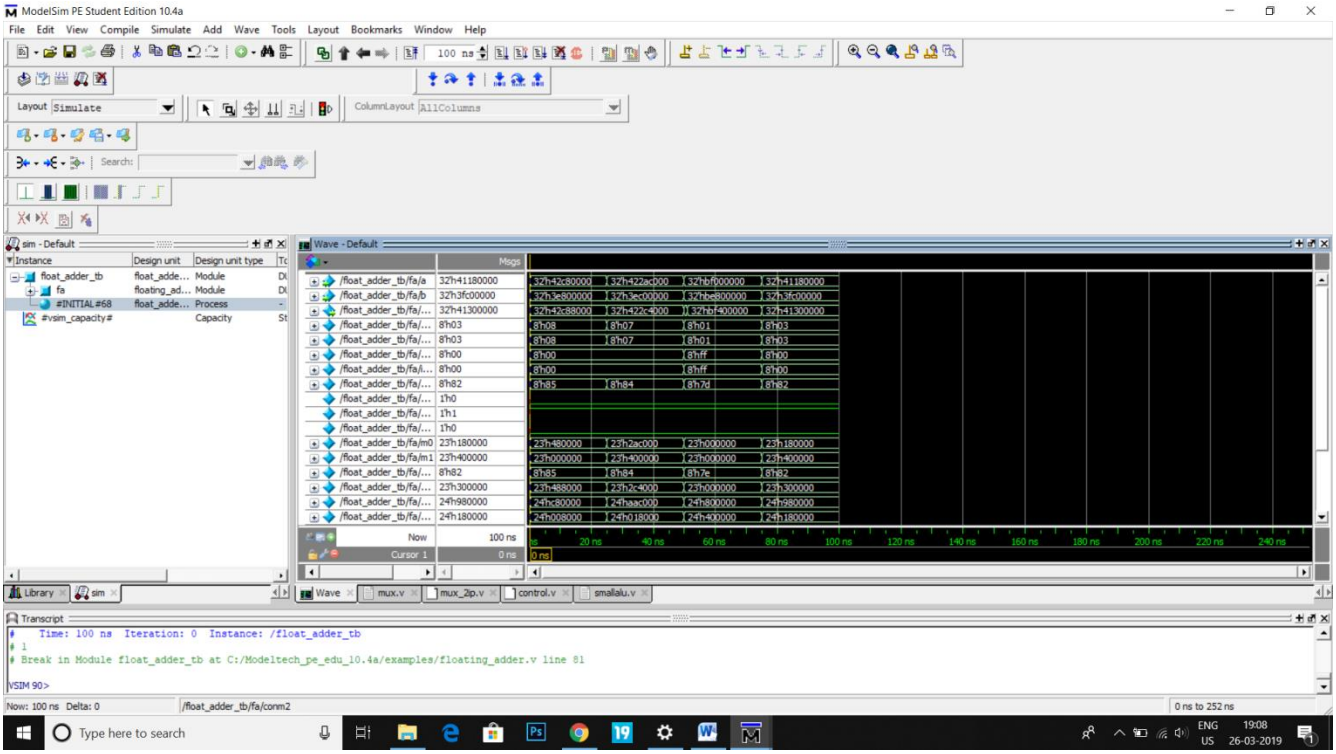
```
pos_0 = i;

carryout = pos_1 - pos_0;
tempsum = tempsum >> shiftMantissa;
op_exp = op_exp + incdec;

tsum = tempsum;

i=23;
while(tsum[i]==1)
begin
        i=i-1;
end
tsum[i]=0;

op_mantissa = tsum[22:0];
sum = {a[31], op_exp, op_mantissa};
end
endmodule


module float_adder_tb;
reg [31:0] a,b;
wire [31:0] sum;

floating_adder fa(a,b,sum);
initial begin
a = 32'h42c80000;
b = 32'h3e800000;
#25;
a = 32'h422ac000;
b = 32'h3ec00000;
#25;
b = 32'hbe800000;
a = 32'hbf000000;
#25;
a=32'h41180000;
b=32'h3fc00000;
#25;
$finish;
end
endmodule
```



```
module floating_adder(a, b, sum);
input [31:0] a;
input [31:0] b;
output reg[31:0] sum;
wire [7:0] exp_diff,shiftAmt, shiftMantissa, incdec;
reg [7:0] op_exp;
wire sel0,sel1,selE;
wire [22:0] m0, m1;
wire[7:0] temp_op_exp;
reg [22:0] op_mantissa;
reg [23:0] conm1, conm2, tempsum,tsum;
reg [7:0] i;
reg [7:0] pos_0,pos_1,carryout;

smallalu small_alu(a[30:23],b[30:23],exp_diff);
control cont(a,b,exp_diff,sel0,sel1,selE,carryout,shiftAmt,shiftMantissa,incdec);
mux_2ip mux0(sel0, a[22:0],b[22:0],m0);
mux_2ip mux1(sel1, a[22:0],b[22:0],m1);
mux_2ip mux2(selE, a[30:23],b[30:23], temp_op_exp);

always @ *
begin
op_exp = temp_op_exp;
conm1 = {1'b1,m0};
conm2 = {1'b1,m1};
conm2 = conm2 >> shiftAmt;
tempsum = conm1+ conm2;

i=23;
while(tempsum[i]==1)
begin
        i=i-1;
end
pos_1 = i;

i=23;
while(conm1[i]==1)
begin
        i=i-1;
end
pos_0 = i;

carryout = pos_1 - pos_0;
tempsum = tempsum >> shiftMantissa;
op_exp = op_exp + incdec;
```

Q9.3. Copy the image of waveform window that is generated for your Testbench?

Answer:



Q9.4. From the synthesis report find the Delay and Resource information your floating point adder on Spartan 3E XC3S1600E.

Answer:

Q9.5. Once design, test and verification are complete call one of the instructors and get your design/output verified.

Q9.6. List the concepts you learnt from this experiment (Conclusions/Observations)

Answer: We learnt the implementation of a floating point adder circuit and the methodology of adding and storing floating point numbers

Sumbitted by
Neil Thanawala 2015A8PS0517G