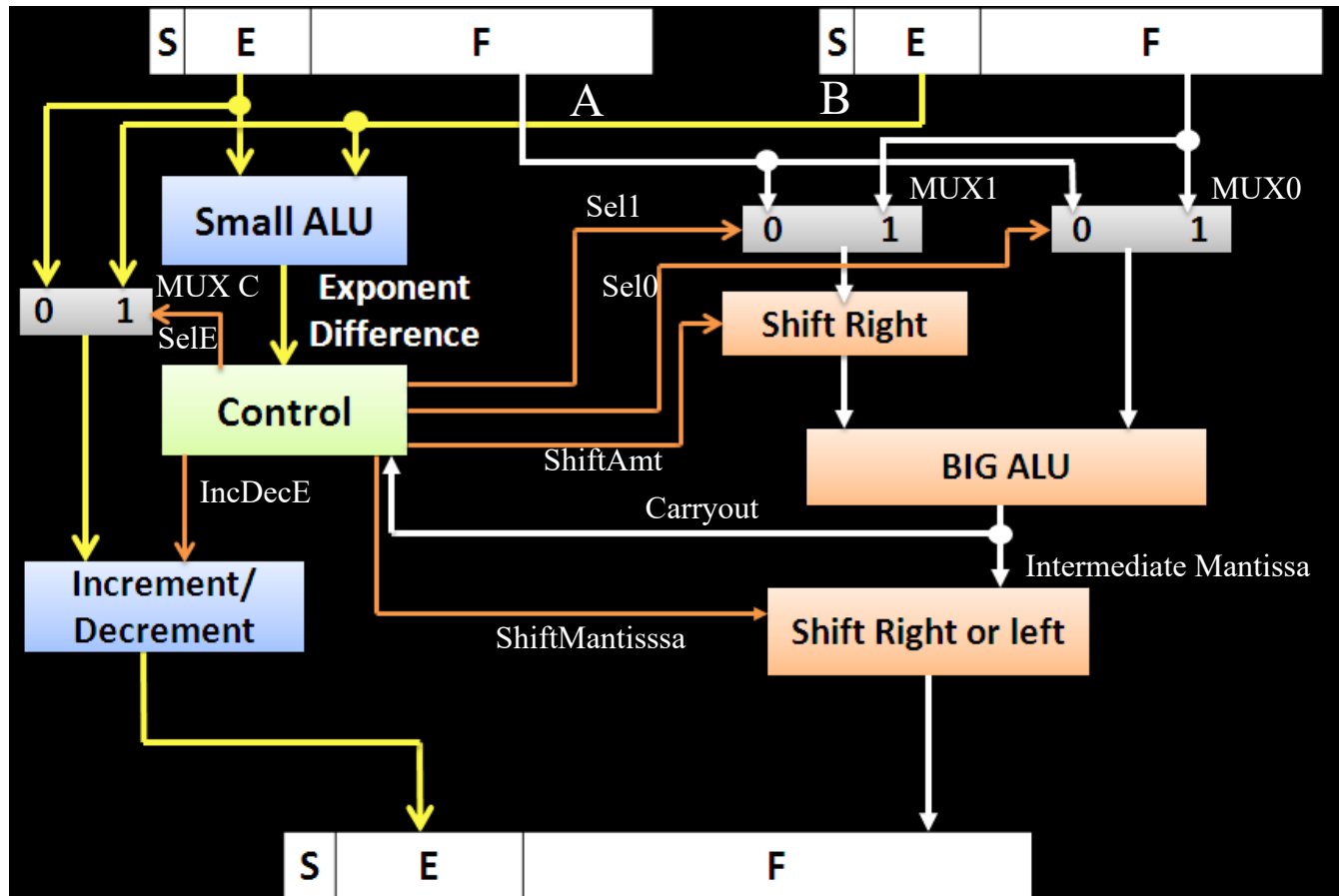


## Experiment No 9: Implementation of Floating point Adder

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



**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.**

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 images 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

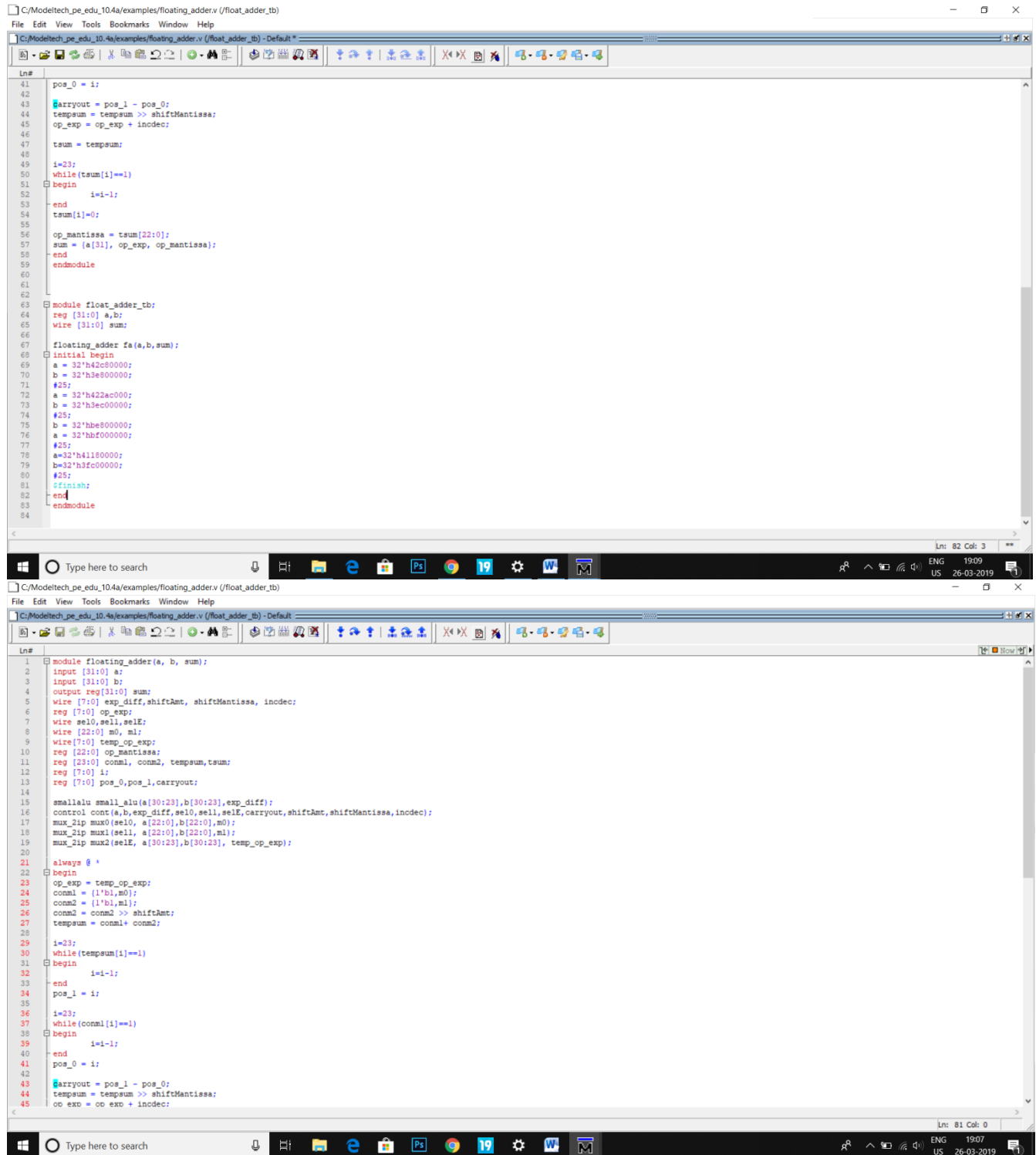


```
C:/Modeltech_pe_edu_10.4a/examples/control.v (/float_adder_tb/fa/cont)
File Edit View Tools Bookmarks Window Help
C:/Modeltech_pe_edu_10.4a/examples/control.v (/float_adder_tb/fa/cont) - Default

Ln#
1 module control(a,b,exp_diff,sel0,sell,selE,carryout,shiftAmt,shiftMantissa,indec);
2   input [31:0] a,b;
3   input [7:0] exp_diff;
4   output reg sel0,sell,selE;
5   output reg [7:0] shiftAmt, shiftMantissa;
6   input [7:0] carryout;
7   output reg [7:0] indec;
8
9   always @ *
10  begin
11    shiftAmt = exp_diff;
12    shiftMantissa = carryout;
13    indec = carryout;
14
15    if(a[30:23]>b[30:23])
16    begin
17      sel0 = 0;
18      sell = 1;
19      selE = 0;
20    end
21
22    else
23    begin
24      sel0 = 1;
25      sell = 0;
26      selE = 1;
27    end
28
29    if(exp_diff<0)
30    begin
31      shiftAmt = 0 -shiftAmt ;
32    end
33
34  end
35 endmodule
36
```

**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:

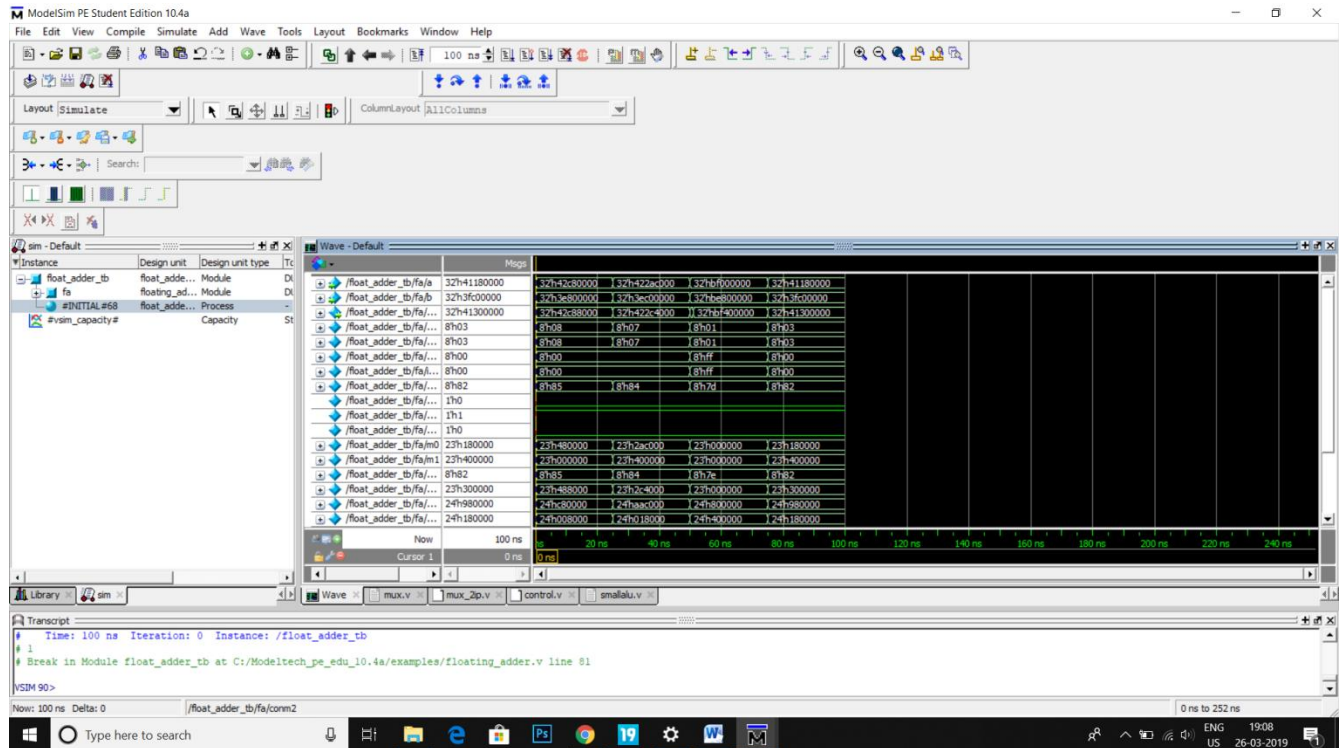


```
41 pos_0 = 1;
42
43 carryout = pos_1 - pos_0;
44 tempsum = tempsum >> shiftMantissa;
45 op_exp = op_exp + inoddec;
46
47 tsum = tempsum;
48
49 i=23;
50 while(tsum[i]==1)
51 begin
52     i=i-1;
53 end
54 tsum[i]=0;
55
56 op_mantissa = tsum[22:0];
57 sum = {a[31], op_exp, op_mantissa};
58 end
59 endmodule
60
61
62
63 module float_adder_tb;
64 reg [31:0] a,b;
65 wire [31:0] sum;
66
67 floating_adder fa(a,b,sum);
68
69 initial begin
70     a = 32'h42c80000;
71     b = 32'h3ec00000;
72     #25;
73     a = 32'h422ac000;
74     b = 32'h3ec00000;
75     #25;
76     a = 32'hbe800000;
77     b = 32'hbf000000;
78     #25;
79     a=32'h41180000;
80     b=32'h3fc00000;
81     #25;
82     $finish;
83 end
84 endmodule
85
```

```
1 module floating_adder(a, b, sum);
2 input [31:0] a;
3 input [31:0] b;
4 output reg[31:0] sum;
5 wire [7:0] exp_diff,shiftAmt, shiftMantissa, inoddec;
6 reg [7:0] op_exp;
7 wire sel0,sel1,selE;
8 wire [22:0] m0, m1;
9 wire[7:0] temp_op_exp;
10 reg [22:0] op_mantissa;
11 reg [23:0] conml, conml2, tempsum,tsum;
12 reg [7:0] i;
13 reg [7:0] pos_0,pos_1,carryout;
14
15 smallalu small_alu(a[30:23],b[30:23],exp_diff);
16 control cont(a,b,exp_diff,sel0,sel1,selE,carryout,shiftAmt,shiftMantissa,inoddec);
17 mux_2ip mux0(sel0, a[22:0],b[22:0],m0);
18 mux_2ip mux1(sel1, a[22:0],b[22:0],m1);
19 mux_2ip mux2(selE, a[30:23],b[30:23], temp_op_exp);
20
21 always @ *
22 begin
23     op_exp = temp_op_exp;
24     conml = {1'b1,m0};
25     conml2 = {1'b1,m1};
26     conml2 = conml2 >> shiftAmt;
27     tempsum = conml+ conml2;
28
29     i=23;
30     while(tempsum[i]==1)
31     begin
32         i=i-1;
33     end
34     pos_1 = i;
35
36     i=23;
37     while(conml[i]==1)
38     begin
39         i=i-1;
40     end
41     pos_0 = i;
42
43     carryout = pos_1 - pos_0;
44     tempsum = tempsum >> shiftMantissa;
45     op_exp = op_exp + inoddec;
46 end
```

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