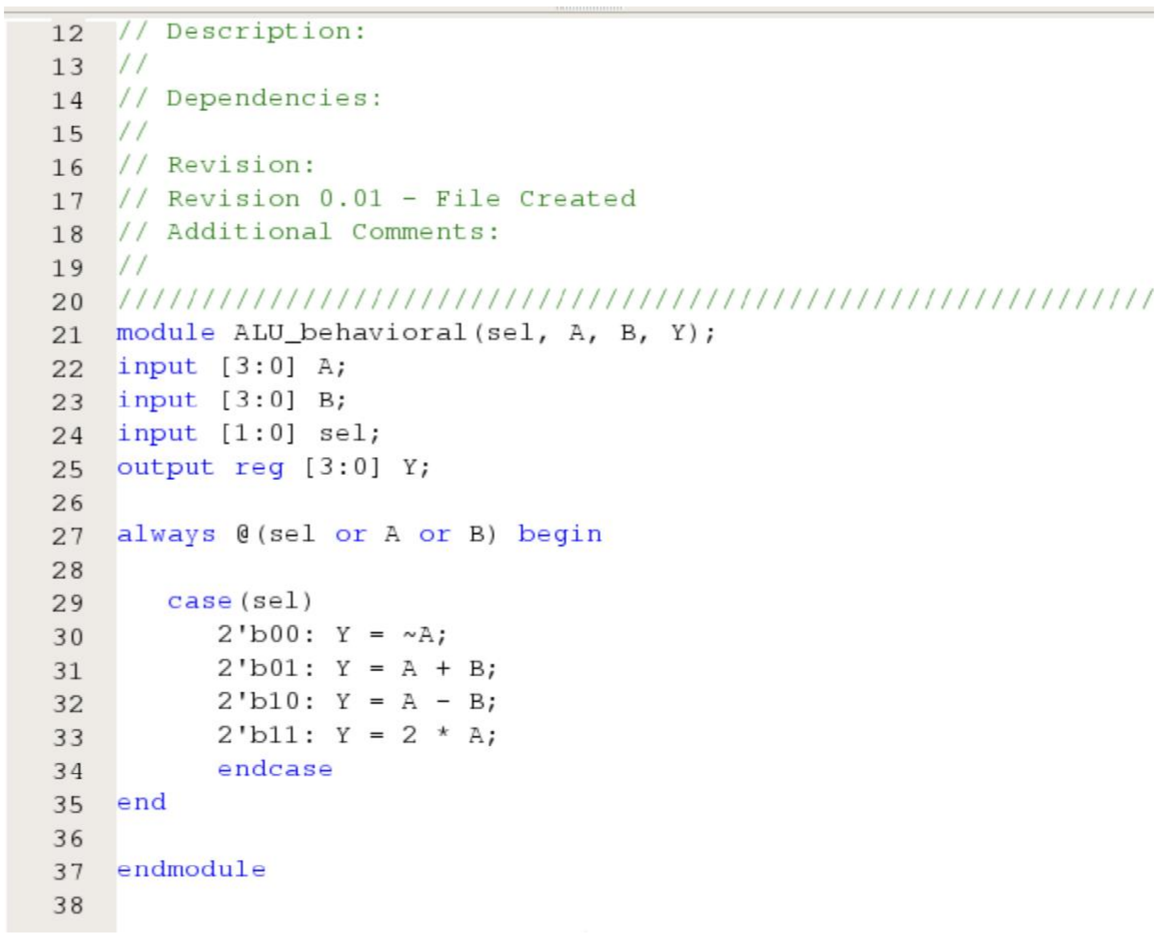
CDA 4203L
Computer System Design Lab

Lab 2 Report
Verilog Based ALU Design

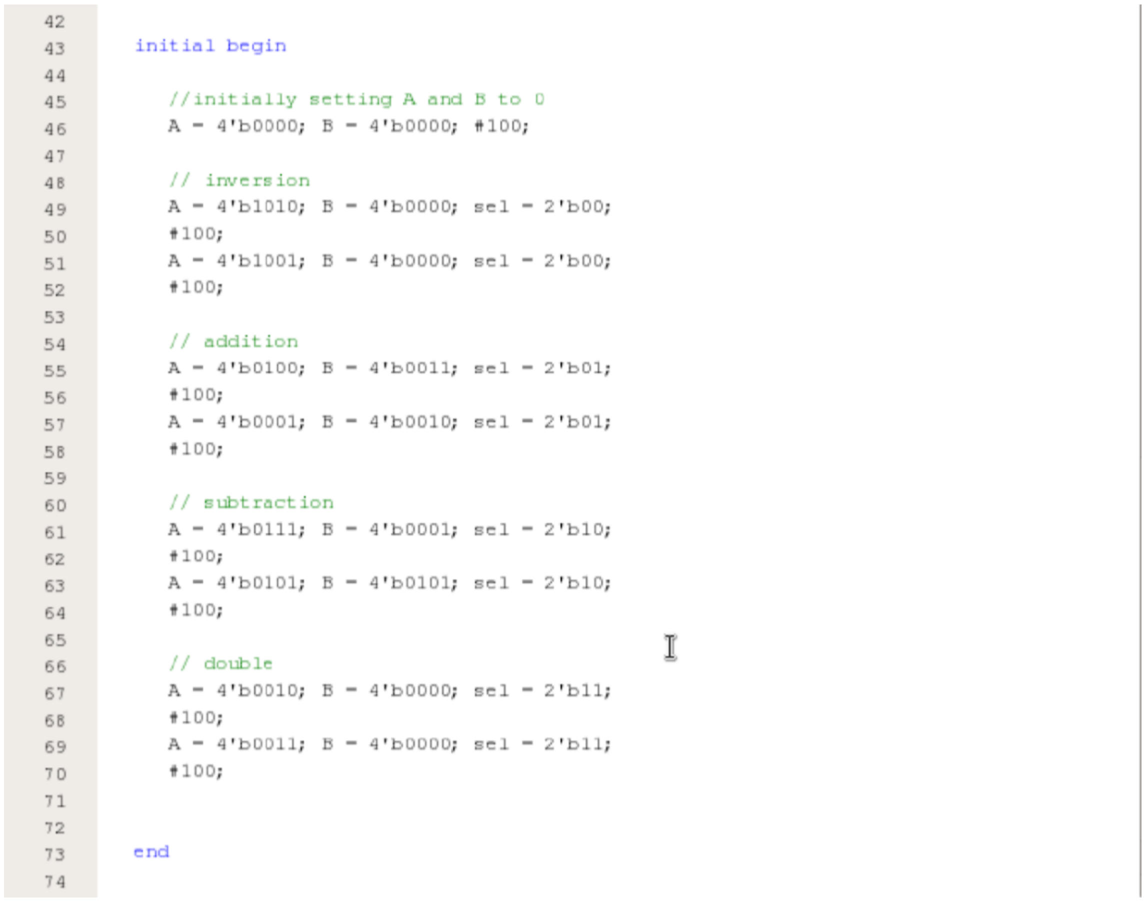
Today's Date:	Feb 12, 2025
Your Name:	Claude Watson
Your U Number:	U72087839
No. of Hours Spent:	10
Exercise Difficulty: (Easy, Average, Hard)	Average
Any Other Feedback:	N/A

Problem 1: ALU Behavioral Verilog



```
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
/////////////////////////////////////////////////////////////////
module ALU_behavioral(sel, A, B, Y);
input [3:0] A;
input [3:0] B;
input [1:0] sel;
output reg [3:0] Y;

always @(sel or A or B) begin

   case(sel)
      2'b00: Y = ~A;
      2'b01: Y = A + B;
      2'b10: Y = A - B;
      2'b11: Y = 2 * A;
      endcase
end

endmodule

```

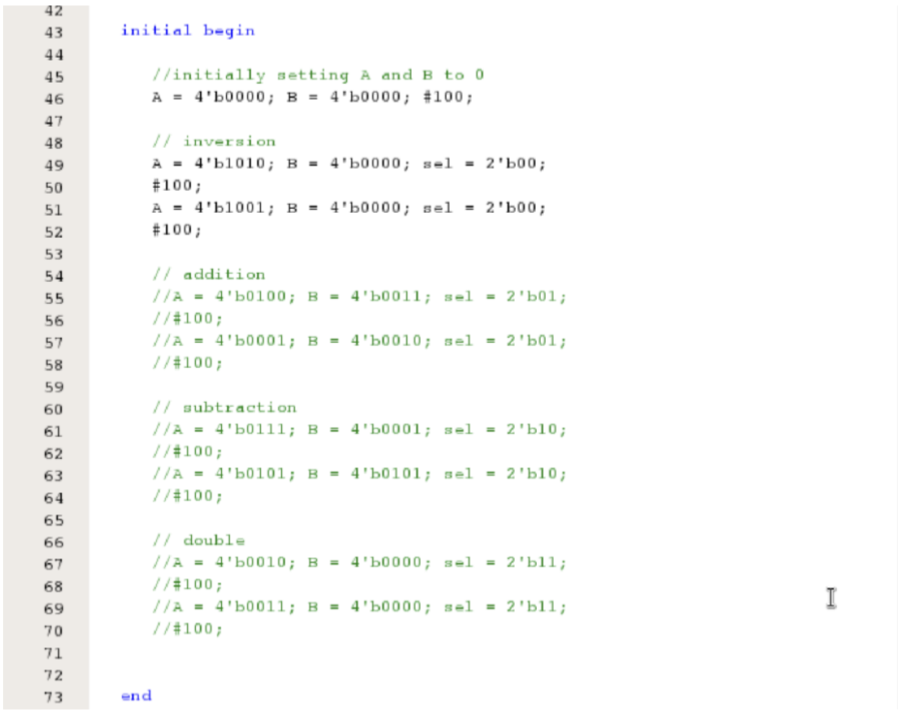
Problem 1: Verilog Test Bench



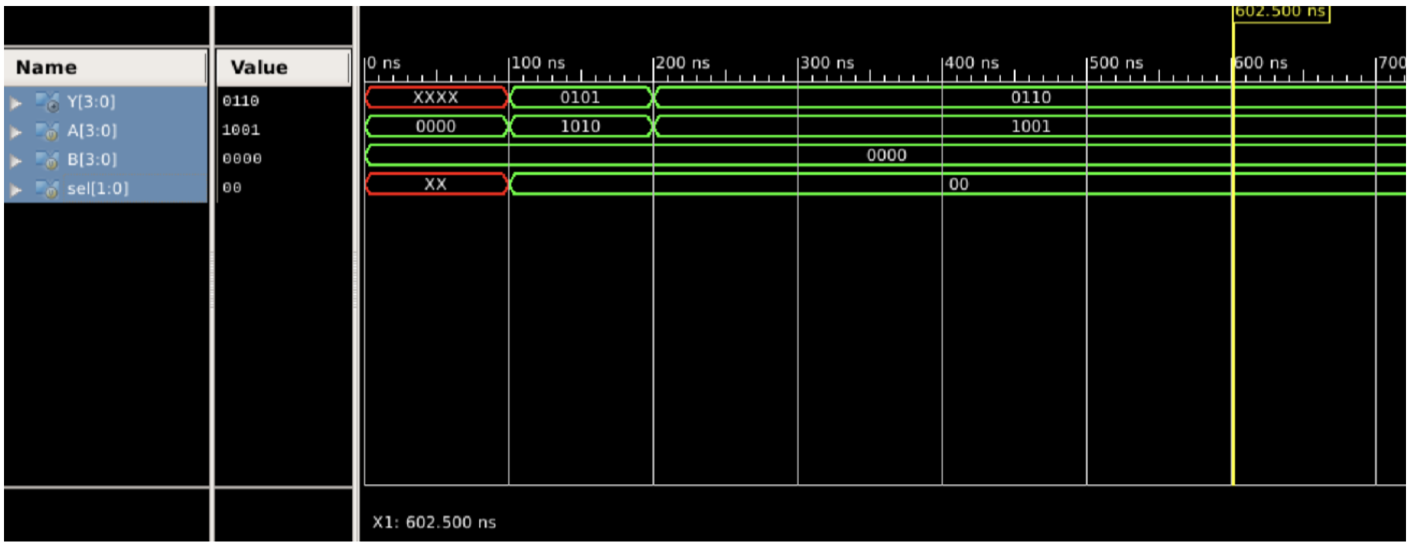
```verilog

initial begin

   //initially setting A and B to 0
   A = 4'b0000; B = 4'b0000; #100;

   // inversion
   A = 4'b1010; B = 4'b0000; sel = 2'b00;
   #100;
   A = 4'b1001; B = 4'b0000; sel = 2'b00;
   #100;

   // addition
   A = 4'b0100; B = 4'b0011; sel = 2'b01;
   #100;
   A = 4'b0001; B = 4'b0010; sel = 2'b01;
   #100;

   // subtraction
   A = 4'b0111; B = 4'b0001; sel = 2'b10;
   #100;
   A = 4'b0101; B = 4'b0101; sel = 2'b10;
   #100;

   // double
   A = 4'b0010; B = 4'b0000; sel = 2'b11;
   #100;
   A = 4'b0011; B = 4'b0000; sel = 2'b11;
   #100;


end

```

Problem 1: Simulation Waveforms (add as many pages as you need).

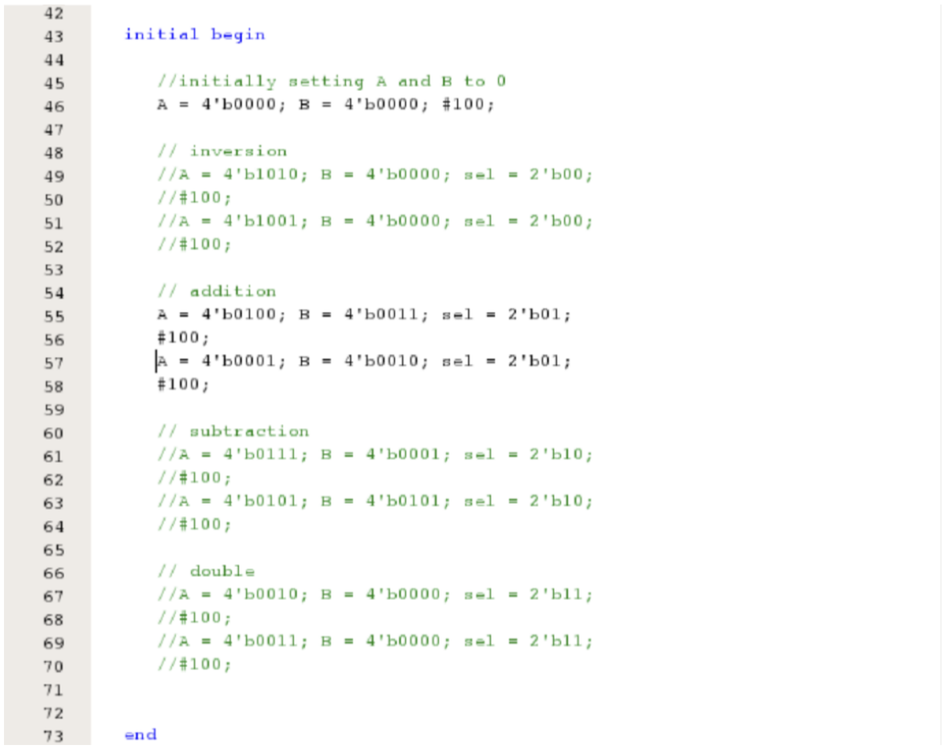
Inversion:



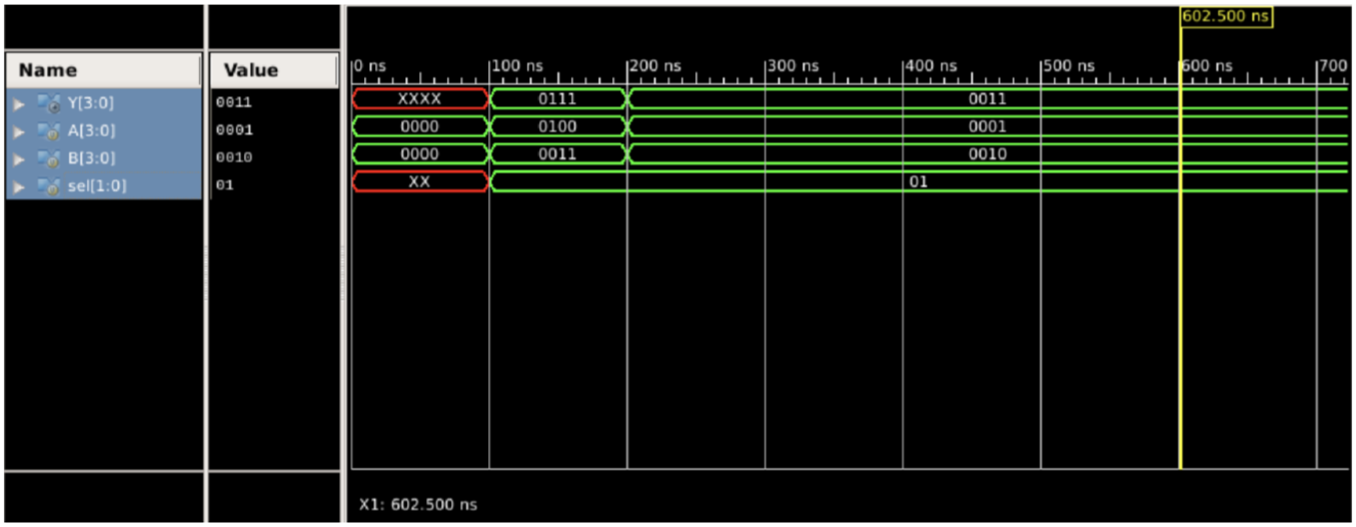
```
initial begin

   //initially setting A and B to 0
   A = 4'b0000; B = 4'b0000; #100;

   // inversion
   A = 4'b1010; B = 4'b0000; sel = 2'b00;
   #100;
   A = 4'b1001; B = 4'b0000; sel = 2'b00;
   #100;

   // addition
   //A = 4'b0100; B = 4'b0011; sel = 2'b01;
   //#100;
   //A = 4'b0001; B = 4'b0010; sel = 2'b01;
   //#100;

   // subtraction
   //A = 4'b0111; B = 4'b0001; sel = 2'b10;
   //#100;
   //A = 4'b0101; B = 4'b0101; sel = 2'b10;
   //#100;

   // double
   //A = 4'b0010; B = 4'b0000; sel = 2'b11;
   //#100;
   //A = 4'b0011; B = 4'b0000; sel = 2'b11;
   //#100;


end
```



Addition:

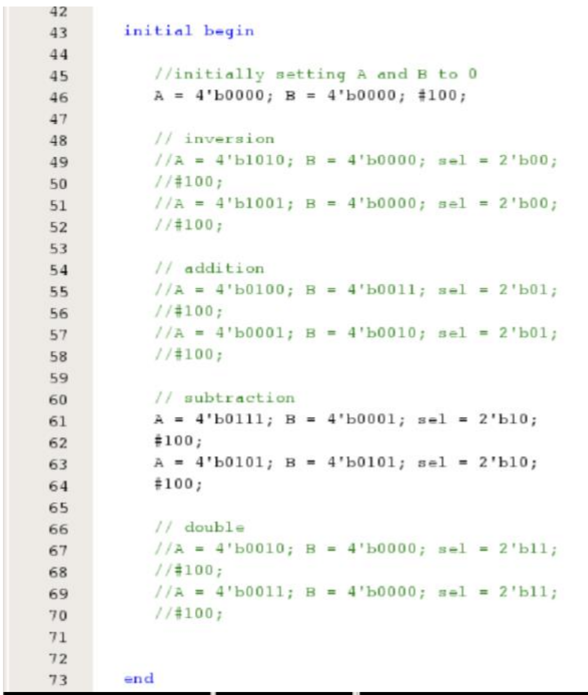


```
initial begin

    //initially setting A and B to 0
    A = 4'b0000; B = 4'b0000; #100;

    // inversion
    //A = 4'b1010; B = 4'b0000; sel = 2'b00;
    //#100;
    //A = 4'b1001; B = 4'b0000; sel = 2'b00;
    //#100;

    // addition
    A = 4'b0100; B = 4'b0011; sel = 2'b01;
    #100;
    A = 4'b0001; B = 4'b0010; sel = 2'b01;
    #100;

    // subtraction
    //A = 4'b0111; B = 4'b0001; sel = 2'b10;
    //#100;
    //A = 4'b0101; B = 4'b0101; sel = 2'b10;
    //#100;

    // double
    //A = 4'b0010; B = 4'b0000; sel = 2'b11;
    //#100;
    //A = 4'b0011; B = 4'b0000; sel = 2'b11;
    //#100;


end
```

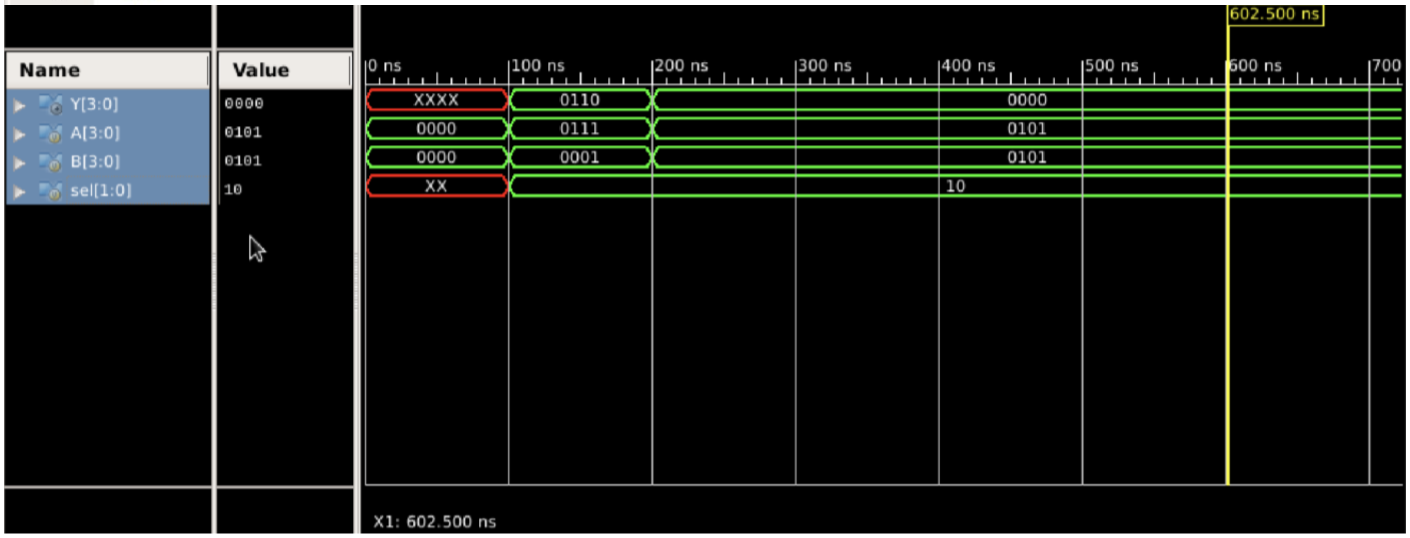


Name	Value	0 ns	100 ns	200 ns – 700
Y[3:0]	0011	XXXX	0111	0011
A[3:0]	0001	0000	0100	0001
B[3:0]	0010	0000	0011	0010
sel[1:0]	01	XX	01	01

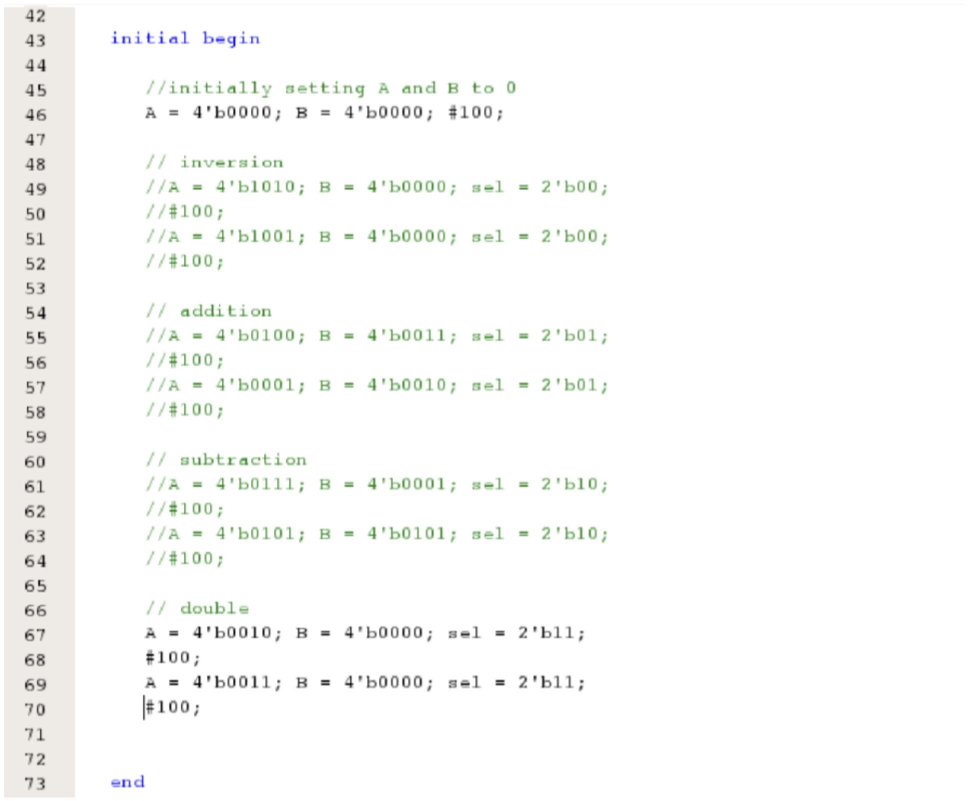
Subtraction:



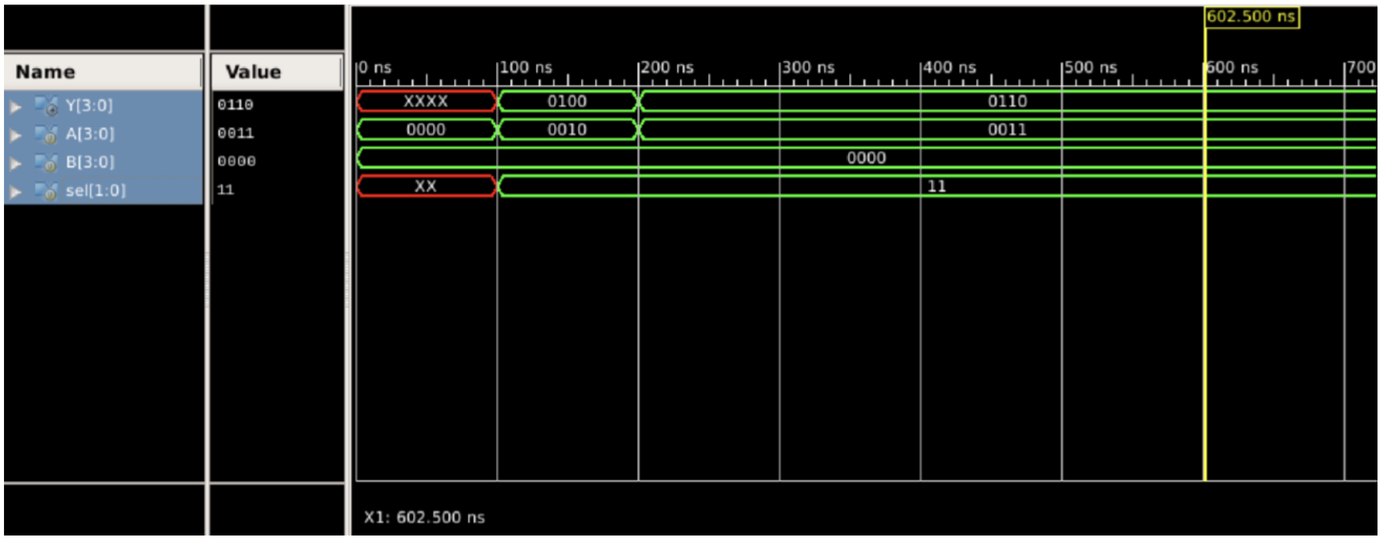
```
initial begin

   //initially setting A and B to 0
   A = 4'b0000; B = 4'b0000; #100;

   // inversion
   //A = 4'b1010; B = 4'b0000; sel = 2'b00;
   //#100;
   //A = 4'b1001; B = 4'b0000; sel = 2'b00;
   //#100;

   // addition
   //A = 4'b0100; B = 4'b0011; sel = 2'b01;
   //#100;
   //A = 4'b0001; B = 4'b0010; sel = 2'b01;
   //#100;

   // subtraction
   A = 4'b0111; B = 4'b0001; sel = 2'b10;
   #100;
   A = 4'b0101; B = 4'b0101; sel = 2'b10;
   #100;

   // double
   //A = 4'b0010; B = 4'b0000; sel = 2'b11;
   //#100;
   //A = 4'b0011; B = 4'b0000; sel = 2'b11;
   //#100;


end
```



Double:

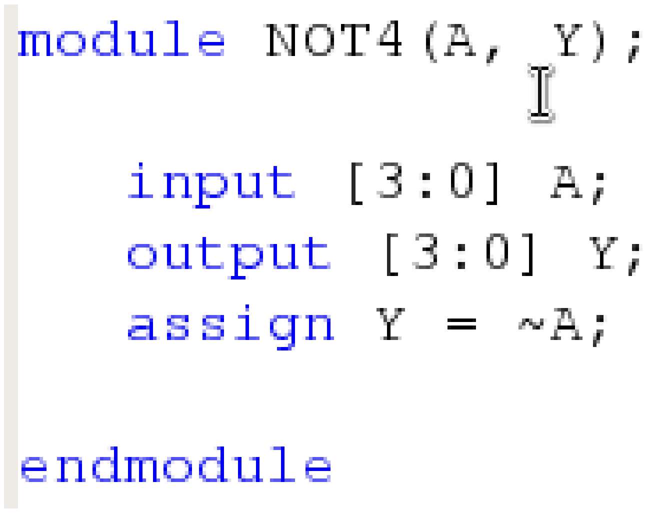


```
initial begin

   //initially setting A and B to 0
   A = 4'b0000; B = 4'b0000; #100;

   // inversion
   //A = 4'b1010; B = 4'b0000; sel = 2'b00;
   //#100;
   //A = 4'b1001; B = 4'b0000; sel = 2'b00;
   //#100;

   // addition
   //A = 4'b0100; B = 4'b0011; sel = 2'b01;
   //#100;
   //A = 4'b0001; B = 4'b0010; sel = 2'b01;
   //#100;

   // subtraction
   //A = 4'b0111; B = 4'b0001; sel = 2'b10;
   //#100;
   //A = 4'b0101; B = 4'b0101; sel = 2'b10;
   //#100;

   // double
   A = 4'b0010; B = 4'b0000; sel = 2'b11;
   #100;
   A = 4'b0011; B = 4'b0000; sel = 2'b11;
   #100;


end
```



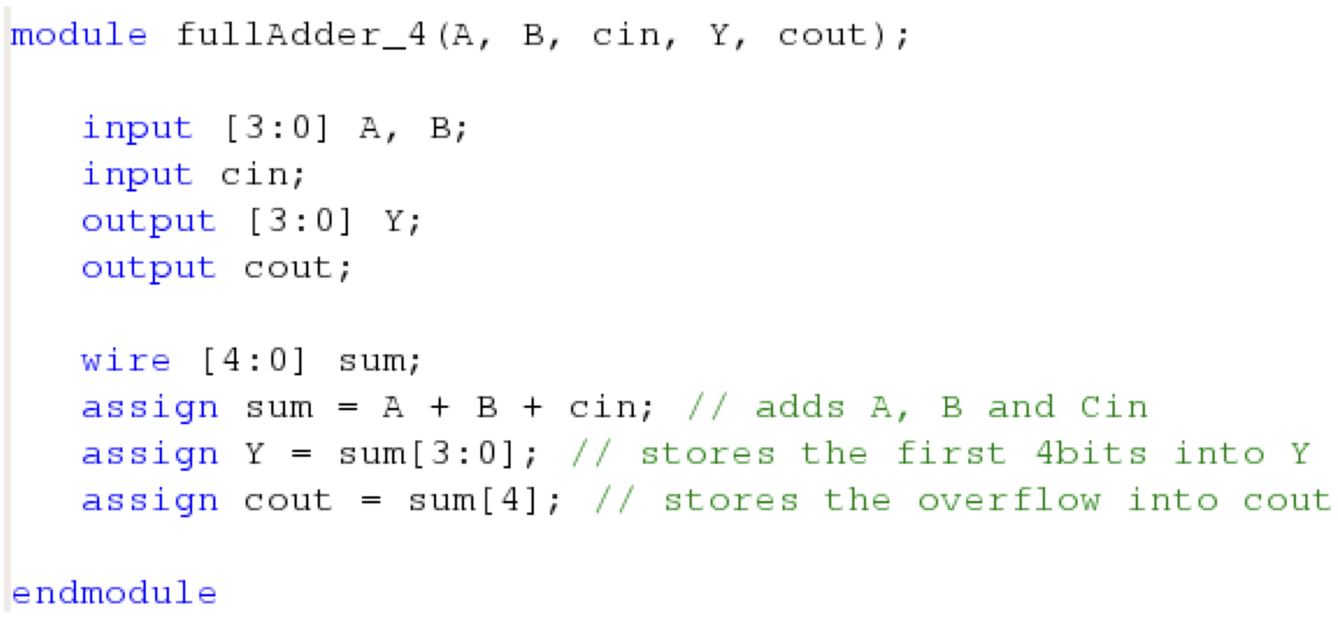
Problem 2: Behavioral Components (insert 1 page per component)

4bit inverter:



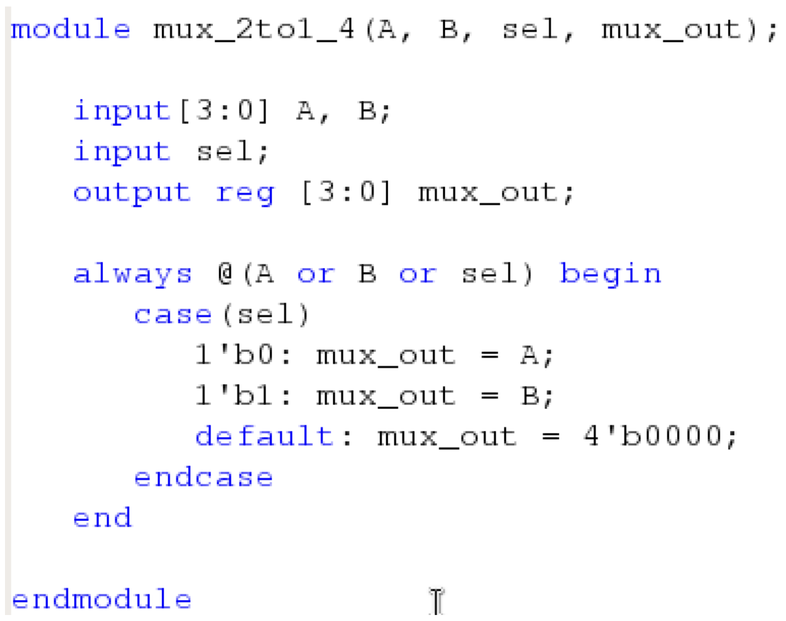
```
module NOT4(A, Y);

   input [3:0] A;
   output [3:0] Y;
   assign Y = ~A;

endmodule
```

4bit Fulladder:



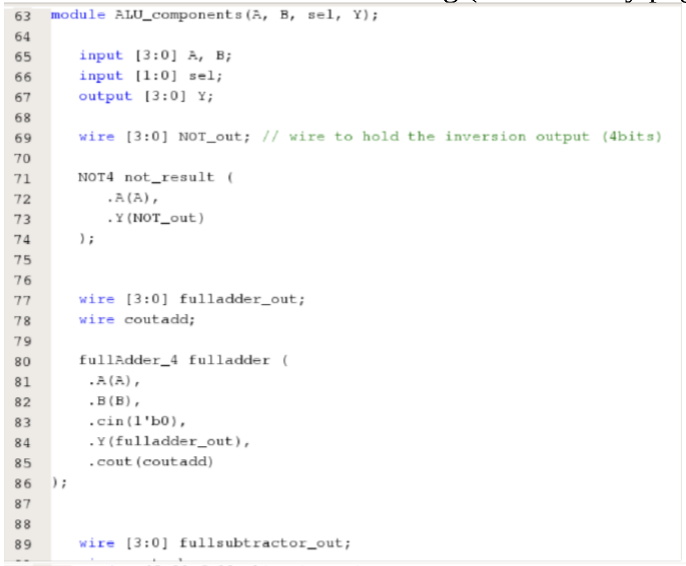
```
module fullAdder_4(A, B, cin, Y, cout);

   input [3:0] A, B;
   input cin;
   output [3:0] Y;
   output cout;

   wire [4:0] sum;
   assign sum = A + B + cin; // adds A, B and Cin
   assign Y = sum[3:0]; // stores the first 4bits into Y
   assign cout = sum[4]; // stores the overflow into cout

endmodule
```

4bit 2to1 MUX:

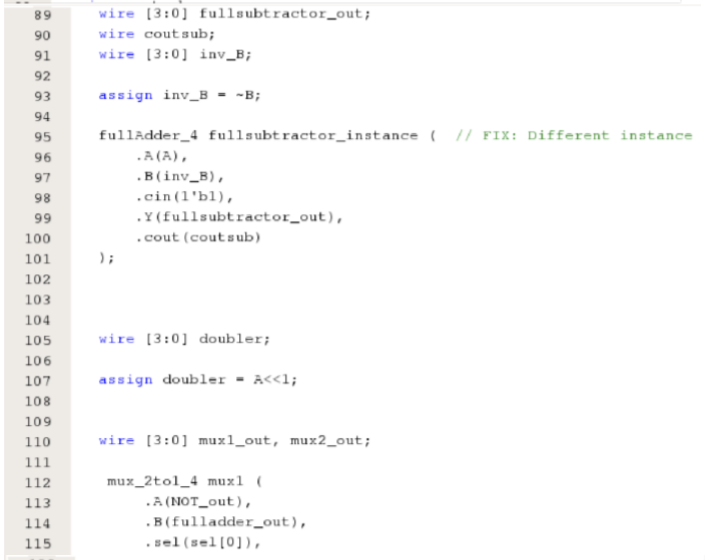


```
module mux_2to1_4(A, B, sel, mux_out);

   input[3:0] A, B;
   input sel;
   output reg [3:0] mux_out;

   always @(A or B or sel) begin
      case(sel)
         1'b0: mux_out = A;
         1'b1: mux_out = B;
         default: mux_out = 4'b0000;
      endcase
   end

endmodule
```

Problem 2: ALU Structural Verilog (Use as many pages as needed.)

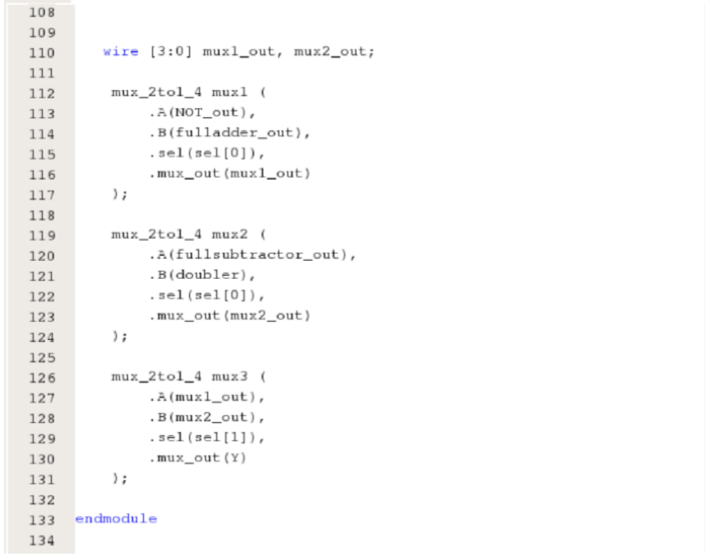


```
module ALU_components(A, B, sel, Y);

   input [3:0] A, B;
   input [1:0] sel;
   output [3:0] Y;

   wire [3:0] NOT_out; // wire to hold the inversion output (4bits)

   NOT4 not_result (
      .A(A),
      .Y(NOT_out)
   );


   wire [3:0] fulladder_out;
   wire coutadd;

   fullAdder_4 fulladder (
    .A(A),
    .B(B),
    .cin(1'b0),
    .Y(fulladder_out),
    .cout(coutadd)
);


   wire [3:0] fullsubtractor_out;
```



```
   wire [3:0] fullsubtractor_out;
```

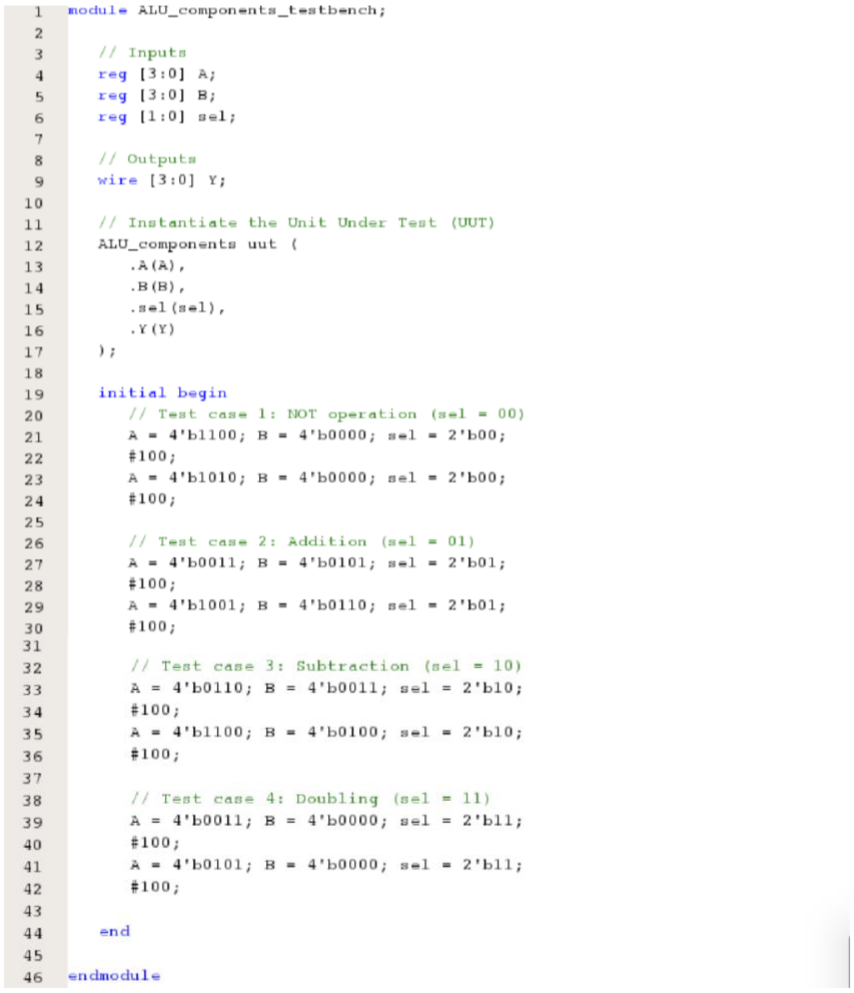
```
   wire coutsub;
   wire [3:0] inv_B;

   assign inv_B = ~B;

   fullAdder_4 fullsubtractor_instance (  // FIX: Different instance
       .A(A),
       .B(inv_B),
       .cin(1'b1),
       .Y(fullsubtractor_out),
       .cout(coutsub)
   );



   wire [3:0] doubler;

   assign doubler = A<<1;


   wire [3:0] mux1_out, mux2_out;

    mux_2to1_4 mux1 (
        .A(NOT_out),
        .B(fulladder_out),
        .sel(sel[0]),
```



```


   wire [3:0] mux1_out, mux2_out;

    mux_2to1_4 mux1 (
        .A(NOT_out),
        .B(fulladder_out),
        .sel(sel[0]),
```

```
        .mux_out(mux1_out)
    );

    mux_2to1_4 mux2 (
        .A(fullsubtractor_out),
        .B(doubler),
        .sel(sel[0]),
        .mux_out(mux2_out)
    );

    mux_2to1_4 mux3 (
        .A(mux1_out),
        .B(mux2_out),
        .sel(sel[1]),
        .mux_out(Y)
    );

endmodule
```

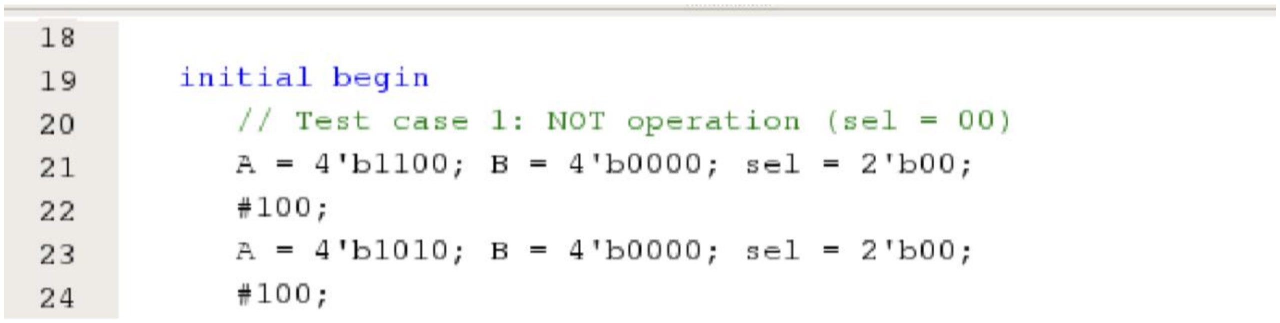
Problem 2: Verilog Test Bench



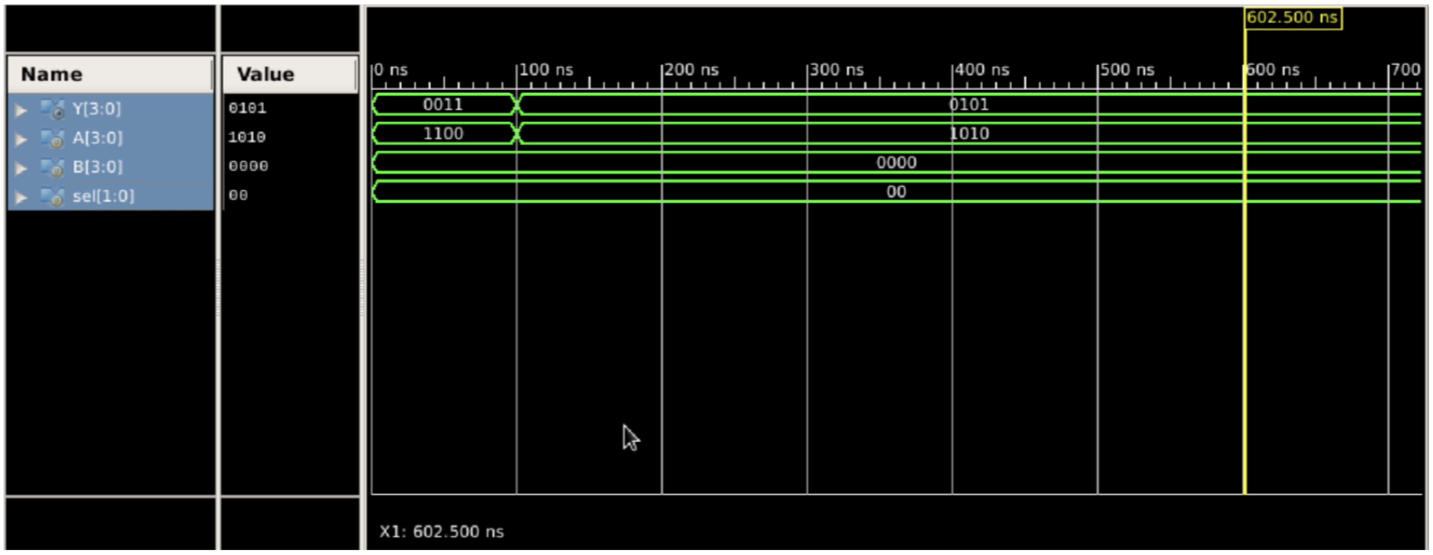
```
module ALU_components_testbench;

   // Inputs
   reg [3:0] A;
   reg [3:0] B;
   reg [1:0] sel;

   // Outputs
   wire [3:0] Y;

   // Instantiate the Unit Under Test (UUT)
   ALU_components uut (
      .A(A),
      .B(B),
      .sel(sel),
      .Y(Y)
   );

   initial begin
      // Test case 1: NOT operation (sel = 00)
      A = 4'b1100; B = 4'b0000; sel = 2'b00;
      #100;
      A = 4'b1010; B = 4'b0000; sel = 2'b00;
      #100;

      // Test case 2: Addition (sel = 01)
      A = 4'b0011; B = 4'b0101; sel = 2'b01;
      #100;
      A = 4'b1001; B = 4'b0110; sel = 2'b01;
      #100;

      // Test case 3: Subtraction (sel = 10)
      A = 4'b0110; B = 4'b0011; sel = 2'b10;
      #100;
      A = 4'b1100; B = 4'b0100; sel = 2'b10;
      #100;

      // Test case 4: Doubling (sel = 11)
      A = 4'b0011; B = 4'b0000; sel = 2'b11;
      #100;
      A = 4'b0101; B = 4'b0000; sel = 2'b11;
      #100;

   end

endmodule
```

Problem 2: Simulation Results

Invert:

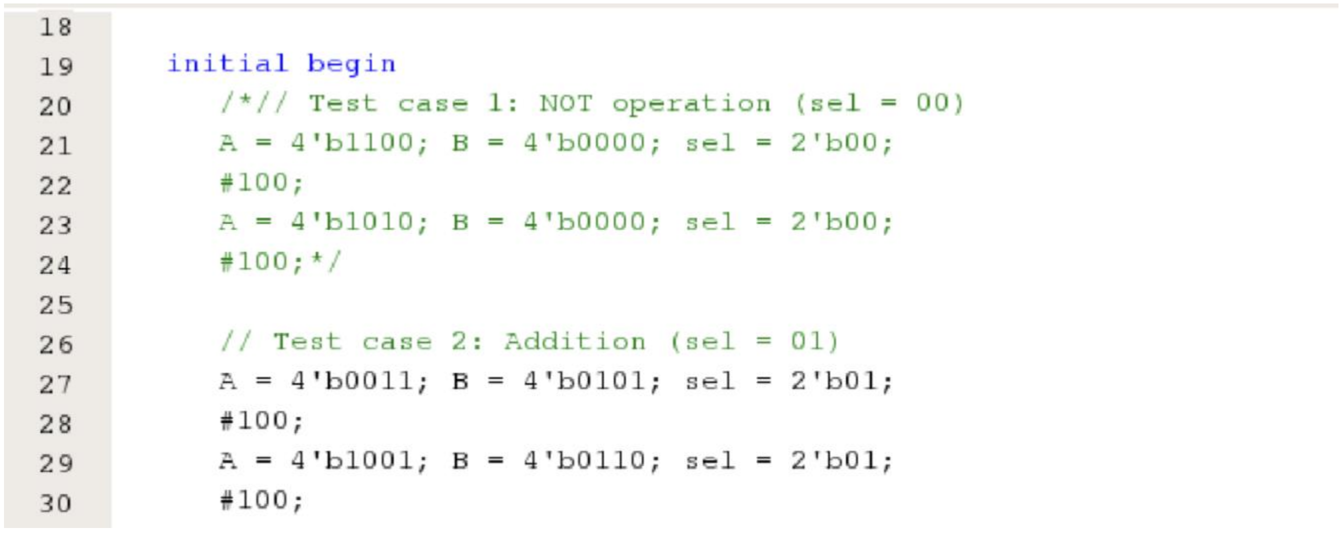


```

initial begin
   // Test case 1: NOT operation (sel = 00)
   A = 4'b1100; B = 4'b0000; sel = 2'b00;
   #100;
   A = 4'b1010; B = 4'b0000; sel = 2'b00;
   #100;
```

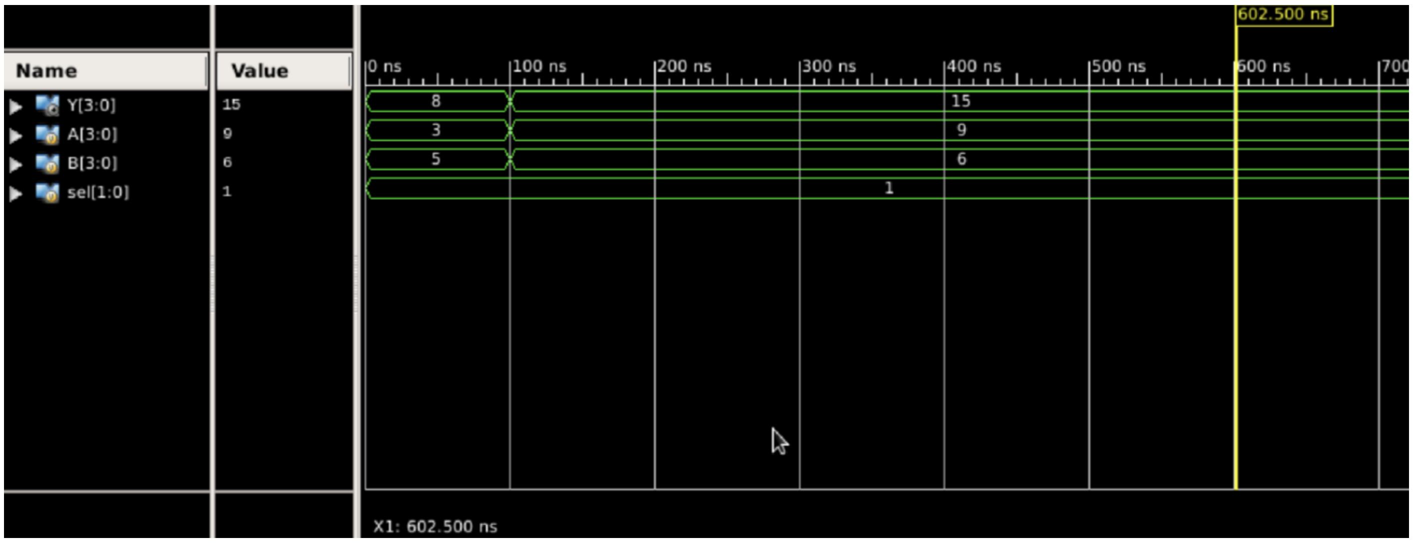


Name	Value
Y[3:0]	0101
A[3:0]	1010
B[3:0]	0000
sel[1:0]	00

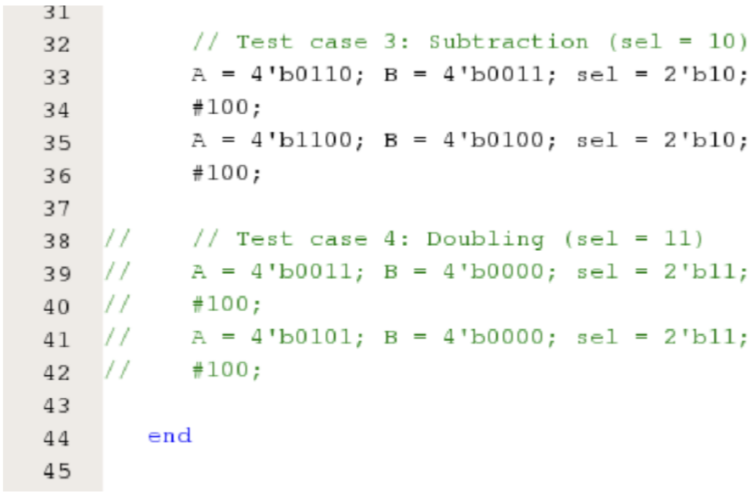
Addition:



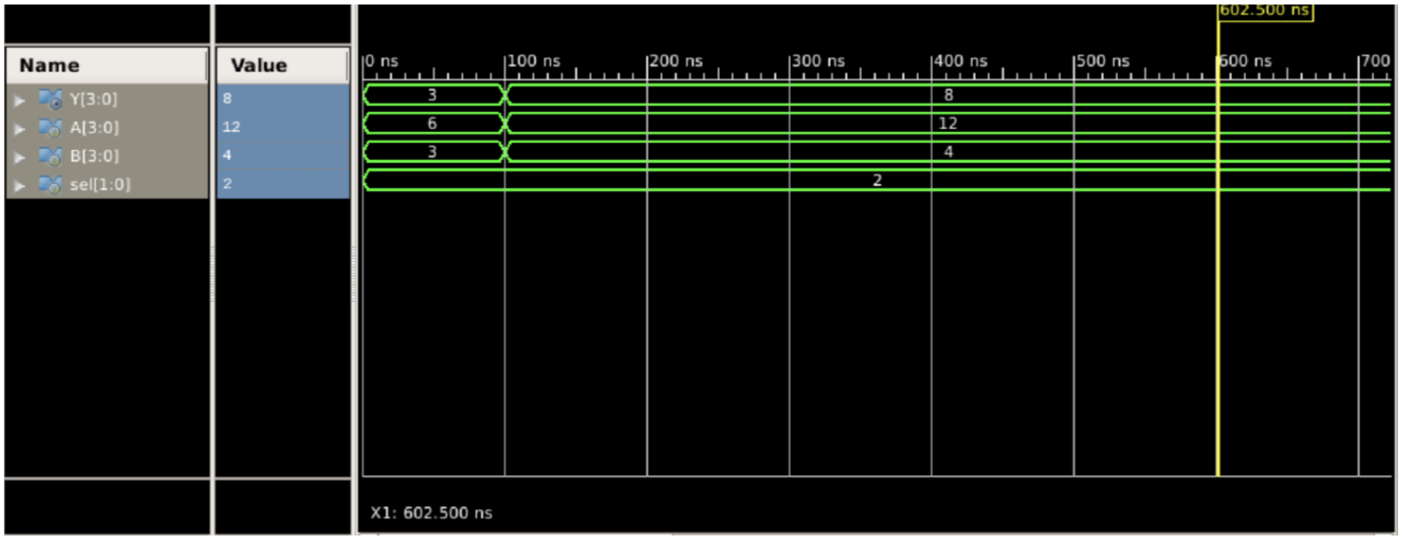
```

initial begin
   /*// Test case 1: NOT operation (sel = 00)
   A = 4'b1100; B = 4'b0000; sel = 2'b00;
   #100;
   A = 4'b1010; B = 4'b0000; sel = 2'b00;
   #100;*/

   // Test case 2: Addition (sel = 01)
   A = 4'b0011; B = 4'b0101; sel = 2'b01;
   #100;
   A = 4'b1001; B = 4'b0110; sel = 2'b01;
   #100;
```



Subtraction:

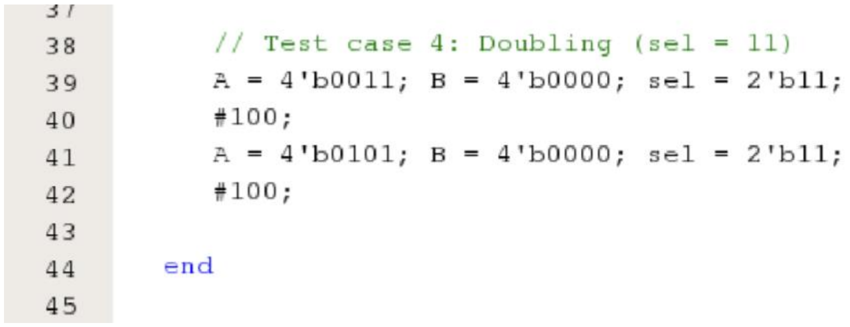


```
        // Test case 3: Subtraction (sel = 10)
        A = 4'b0110; B = 4'b0011; sel = 2'b10;
        #100;
        A = 4'b1100; B = 4'b0100; sel = 2'b10;
        #100;

//      // Test case 4: Doubling (sel = 11)
//      A = 4'b0011; B = 4'b0000; sel = 2'b11;
//      #100;
//      A = 4'b0101; B = 4'b0000; sel = 2'b11;
//      #100;

    end
```

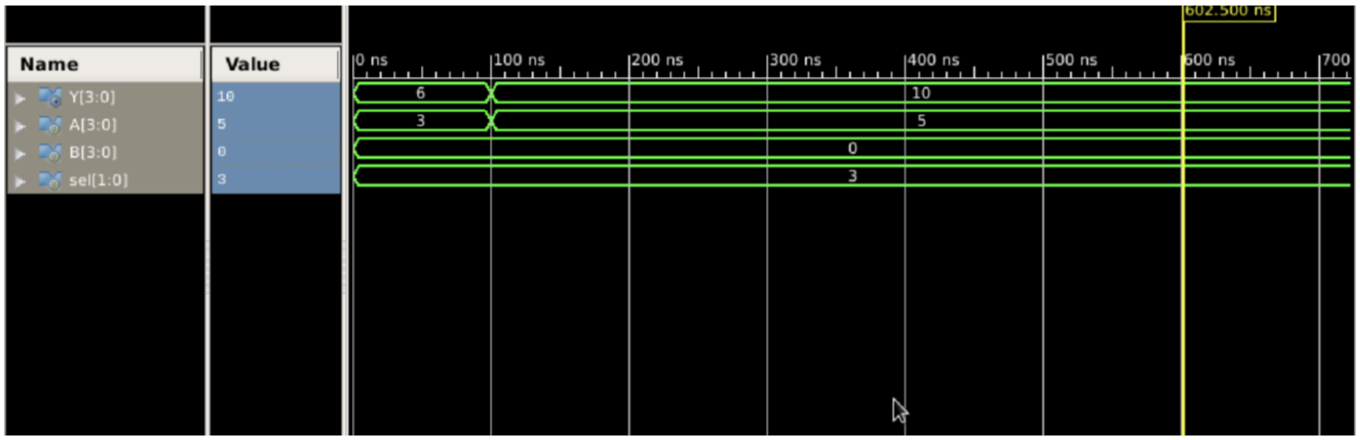


Name	Value
Y[3:0]	8
A[3:0]	12
B[3:0]	4
sel[1:0]	2

Double:



```
    // Test case 4: Doubling (sel = 11)
    A = 4'b0011; B = 4'b0000; sel = 2'b11;
    #100;
    A = 4'b0101; B = 4'b0000; sel = 2'b11;
    #100;

end

```



602.500 ns
Name
Value
Y[3:0] 10
A[3:0] 5
B[3:0] 0
sel[1:0] 3
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700
6
10
3
5
0
3