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

```
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
21 module ALU_behavioral(sel, A, B, Y);
22 input [3:0] A;
23 input [3:0] B;
24 input [1:0] sel;
25 output reg [3:0] Y;
26
27 always @(sel or A or B) begin
28
29
     case (sel)
        2'b00: Y = \sim A;
30
        2'b01: Y = A + B;
31
        2'b10: Y = A - B;
32
        2'b11: Y = 2 * A;
33
        endcase
34
35 end
36
37 endmodule
38
```

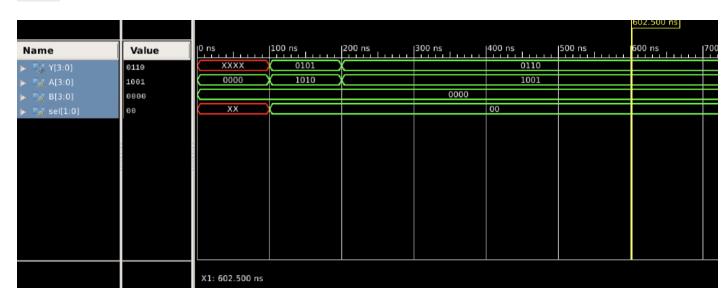
Problem 1: Verilog Test Bench

```
42
        initial begin
 43
 44
           //initially setting A and B to 0
 45
           A - 4'b0000; B - 4'b0000; #100;
 46
 47
           // inversion
 48
           A = 4'b1010; B = 4'b0000; sel = 2'b00;
 49
           #100;
 50
           A - 4'b1001; B - 4'b0000; sel - 2'b00;
 51
 52
           #100;
 53
           // addition
 54
           A = 4'b0100; B = 4'b0011; sel = 2'b01;
 55
 56
           #100;
           A - 4'b0001; B - 4'b0010; sel - 2'b01;
 57
           #100;
 58
 59
           // subtraction
 60
           A - 4'b0111; B - 4'b0001; sel - 2'b10;
 61
 62
           #100;
           A - 4'b0101; B - 4'b0101; sel - 2'b10;
 63
           #100;
 64
 65
                                                    I
           // double
 66
           A = 4'b0010; B = 4'b0000; sel = 2'b11;
 67
 68
           #100;
           A = 4'b0011; B = 4'b0000; sel = 2'b11;
 69
           #100;
 70
 71
 72
 73
        end
7.4
```

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

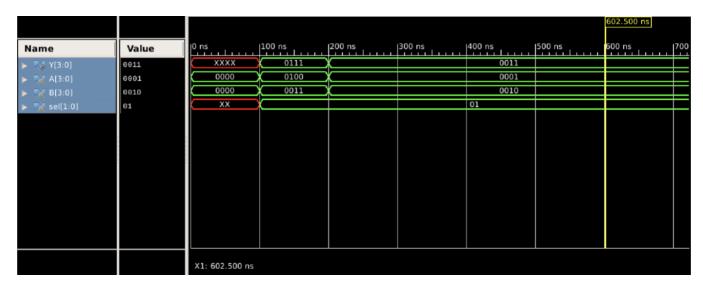
Inversion:

```
initial begin
43
44
45
           //initially setting A and B to 0
          A = 4'b0000; B = 4'b0000; #100;
46
47
           // inversion
48
          A = 4'b1010; B = 4'b00000; sel = 2'b00;
49
          #100;
50
          A = 4'b1001; B = 4'b0000; sel = 2'b00;
51
           #100;
52
53
          // addition
54
          //A = 4'b0100; B = 4'b0011; sel = 2'b01;
55
          //#100;
56
          //A = 4'b0001; B = 4'b0010; sel = 2'b01;
57
          //#100;
58
59
           // subtraction
60
          //A = 4'b0111; B = 4'b0001; sel = 2'b10;
61
          //#100;
62
          //A = 4'b0101; B = 4'b0101; sel = 2'b10;
63
          //#100;
64
65
66
          //A = 4'b0010; B = 4'b0000; sel = 2'b11;
67
          //#100;
68
                                                                             \mathbb{I}
          //A = 4'b0011; B = 4'b0000; sel = 2'b11;
69
70
          //#100;
71
72
        end
73
```



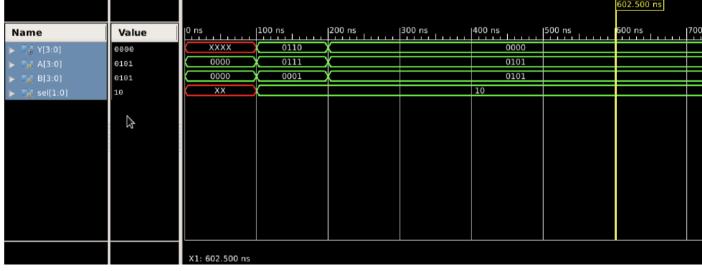
Addition:

```
42
        initial begin
43
44
           //initially setting A and B to 0 A = 4'b0000; B = 4'b0000; #100;
45
46
47
           // inversion
48
           //A = 4'b1010; B = 4'b00000; sel = 2'b00;
49
50
           //#100;
           //A = 4'b1001; B = 4'b00000; sel = 2'b00;
51
           //#100;
52
53
           // addition
54
           A = 4'b0100; B = 4'b0011; sel = 2'b01;
55
           #100;
56
           A = 4'b0001; B = 4'b0010; sel = 2'b01;
57
           #100;
58
59
           // subtraction
60
           //A = 4'b0111; B = 4'b0001; sel = 2'b10;
61
62
           //#100;
           //A = 4'b0101; B = 4'b0101; sel = 2'b10;
63
           //#100;
64
65
66
           // double
           //A = 4'b0010; B = 4'b0000; sel = 2'b11;
67
           //#100;
68
           //A = 4'b0011; B = 4'b00000; sel = 2'b11;
69
           //#100;
70
71
72
        end
73
```



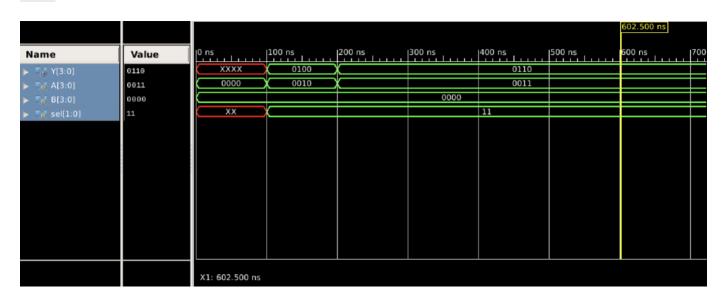
Subtraction:

```
43
        initial begin
44
           //initially setting A and B to 0 A = 4'b0000; B = 4'b0000; #100;
45
46
47
           // inversion
48
           //A = 4'b1010; B = 4'b0000; sel = 2'b00;
49
           //#100;
50
           //A = 4'b1001; B = 4'b0000; sel = 2'b00;
51
           //#100;
52
53
54
           // addition
55
           //A = 4'b0100; B = 4'b0011; sel = 2'b01;
56
           //#100;
57
           //A = 4'b0001; B = 4'b0010; sel = 2'b01;
           //#100;
58
59
           // subtraction
60
           A = 4'b0111; B = 4'b0001; sel = 2'b10;
61
           #100;
62
           A = 4'b0101; B = 4'b0101; sel = 2'b10;
63
           #100;
64
65
66
           // double
           //A = 4'b0010; B = 4'b0000; sel = 2'b11;
67
           //#100;
68
           //A = 4'b0011; B = 4'b0000; sel = 2'b11;
69
70
           //#100;
71
72
        end
73
```



Double:

```
42
        initial begin
 43
 44
 45
           //initially setting A and B to 0
           A = 4'b0000; B = 4'b0000; #100;
 46
 47
           // inversion
 48
           //A = 4'b1010; B = 4'b00000; sel = 2'b00;
 49
           //#100;
 50
           //A = 4'b1001; B = 4'b0000; sel = 2'b00;
 51
           //#100;
 52
 53
           // addition
 54
           //A = 4'b0100; B = 4'b0011; sel = 2'b01;
 55
           //#100;
 56
           //A = 4'b0001; B = 4'b0010; sel = 2'b01;
 57
           //#100;
 58
 59
           // subtraction
 60
           //A = 4'b0111; B = 4'b0001; sel = 2'b10;
 61
           //#100;
 62
           //A = 4'b0101; B = 4'b0101; sel = 2'b10;
 63
 64
           //#100;
 65
           // double
 66
           A = 4'b0010; B = 4'b00000; sel = 2'b11;
 67
           #100;
 68
           A = 4'b0011; B = 4'b0000; sel = 2'b11;
 69
           #100;
 70
 71
 72
        end
73
```



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'b00000;
    endcase
  end

endmodule
```

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

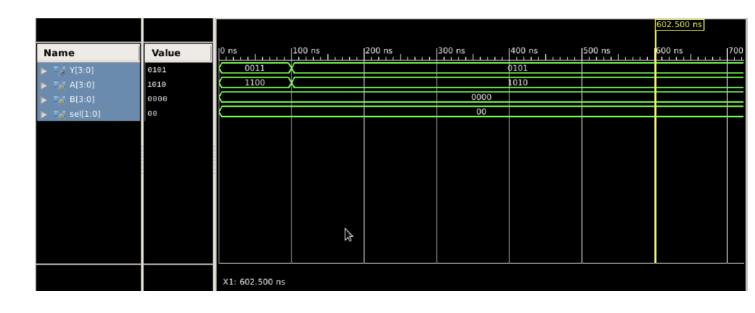
```
module ALU_components(A, B, sel, Y);
65
       input [3:0] A, B;
66
       input [1:0] sel;
       output [3:0] Y;
67
68
       wire [3:0] NOT_out; // wire to hold the inversion output (4bits)
69
70
71
       NOT4 not_result (
          .A(A),
72
73
         .Y(NOT_out)
74
75
76
      wire [3:0] fulladder_out;
77
78
       wire coutadd;
79
       fullAdder_4 fulladder (
80
       .A(A),
81
82
       .B(B),
83
        .cin(1'b0),
        .Y(fulladder_out),
85
        .cout(coutadd)
86);
87
88
       wire [3:0] fullsubtractor_out;
89
        wire [3:0] fullsubtractor_out;
 89
         wire coutsub;
  90
  91
         wire [3:0] inv_B;
  93
         assign inv_B = ~B;
  94
         fullAdder_4 fullsubtractor_instance ( // FIX: Different instance
  95
             .A(A),
  96
             .B(inv_B),
  97
             .cin(1'b1),
  98
             .Y(fullsubtractor_out),
  99
             .cout(coutsub)
 100
 101
 102
 103
 104
         wire [3:0] doubler;
 105
 106
         assign doubler = A<<1;
 107
 108
 109
 110
         wire [3:0] mux1_out, mux2_out;
          mux_2to1_4 mux1 (
 112
 113
              .A(NOT_out),
              .B(fulladder_out),
 114
              .sel(sel[0]),
 115
 108
 109
 110
         wire [3:0] mux1_out, mux2_out;
 112
          mux_2to1_4 mux1 (
 113
              .A(NOT out),
               .B(fulladder out),
 114
               .sel(sel[0]),
 115
               .mux_out(mux1_out)
 116
 117
 118
           mux_2to1_4 mux2 (
 120
              .A(fullsubtractor_out),
 121
               .B(doubler),
               .sel(sel[0]),
 122
 123
              .mux_out(mux2_out)
 124
 125
          mux_2to1_4 mux3 (
 126
              .A(mux1_out),
 127
 128
               .B(mux2_out),
 129
               .sel(sel[1]),
 130
               .mux_out(Y)
 131
 132
 133 endmodule
 134
```

Problem 2: Verilog Test Bench

```
1 module ALU_components_testbench;
 2
         // Inputs
 3
        reg [3:0] A;
  4
        reg [3:0] B;
  5
        reg [1:0] sel;
  6
        // Outputs
  8
        wire [3:0] Y;
 9
 10
        \slash\hspace{-0.05cm}// Instantiate the Unit Under Test (UUT) ALU_components uut (
 11
 12
           .A(A),
 13
           .B(B),
 14
 15
           .sel(sel),
           .Y(Y)
 16
 17
 18
        initial begin
 19
           // Test case 1: NOT operation (sel = 00)
 20
            A = 4'b1100; B = 4'b0000; sel = 2'b00;
 21
           #100;
 22
 23
           A = 4'b1010; B = 4'b0000; sel = 2'b00;
 24
           #100;
 26
           // Test case 2: Addition (sel = 01)
           A = 4'b0011; B = 4'b0101; sel = 2'b01;
 28
           #100;
            A = 4'b1001; B = 4'b0110; sel = 2'b01;
 29
 30
31
            #100;
            // Test case 3: Subtraction (sel = 10)
 32
           A = 4'b0110; B = 4'b0011; sel = 2'b10;
 33
            #100;
 34
            A = 4'b1100; B = 4'b0100; sel = 2'b10;
 35
 36
 37
            // Test case 4: Doubling (sel = 11)
 38
            A = 4'b0011; B = 4'b0000; sel = 2'b11;
 39
           #100;
A = 4'b0101; B = 4'b0000; s=1 = 2'b11;
 40
 41
            #100;
 42
 43
 44
 45
46 endmodule
```

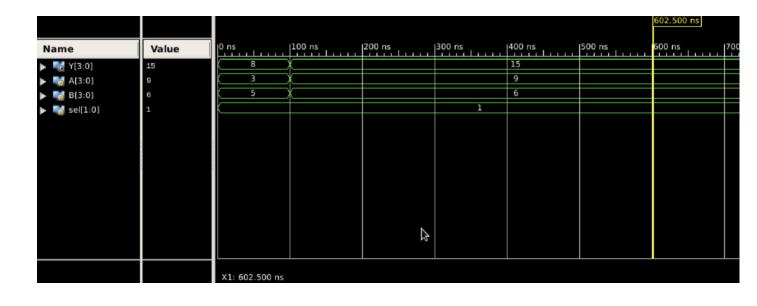
Problem 2: Simulation Results

Invert:



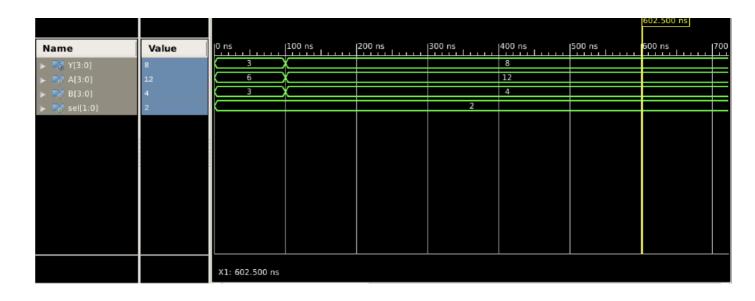
Addition:

```
18
       initial begin
19
          /*// Test case 1: NOT operation (sel = 00)
20
          A = 4'b1100; B = 4'b00000; sel = 2'b00;
21
22
          A = 4'b1010; B = 4'b00000; sel = 2'b000;
23
          #100; */
24
25
          // Test case 2: Addition (sel = 01)
26
          A = 4'b0011; B = 4'b0101; sel = 2'b01;
27
          #100;
28
          A = 4'b1001; B = 4'b0110; sel = 2'b01;
29
          #100;
30
```



Subtraction:

```
31
          // Test case 3: Subtraction (sel = 10)
32
          A = 4'b0110; B = 4'b0011; sel = 2'b10;
33
34
          A = 4'b1100; B = 4'b0100; sel = 2'b10;
35
          #100;
36
37
         // Test case 4: Doubling (sel = 11)
38
         A = 4'b0011; B = 4'b0000; sel = 2'b11;
39
40
         A = 4'b0101; B = 4'b0000; sel = 2'b11;
   1/
41
42 //
          #100;
43
                                                          1
       end
44
45
```



Double:

