Rohan Shaveri
830962238

Lab Report-2

Goal: To design a 3 bit ALU that performs the functions of addition, subtraction, ex-or, left shift using Verilog Hardware description language.

Steps:

- Prelab
- Code for 3 bit ALU
- Simulation on Quartus
- Implementing the design on 3 bit ALU on Altera FPGA board and displaying the output on 7 segment display and LEDs.

Tools:

- Quartus
- Altera

Code: Demo pending

```
module ALU_3bit(out, HEX0, HEX1, a, b, sel);
input [2:0] a,b;
input [1:0] sel;
output [3:0] out;
reg [3:0] out;
output [6:0] HEX0;
output [6:0] HEX1;
reg [6:0] HEX0;
reg [6:0] HEX1;
always @(*)
begin
case(sel)
2'b00 : out = a+b;
2'b01 : out = a-b;
2'b10 : out = a^b;
2'b11 : out = a<<1;
endcase
case(out)
```

```
4'b0000: begin//0
HEX0 = 7'b1000000;//active low for pins of leds
HEX1 = 7'b1000000;
end
4'b0001: begin//1
HEX0 = 7'b1111001;
HEX1 = 7'b1000000;
end
4'b0010: begin//2
HEX0 = 7'b0100100;
HEX1 = 7'b1000000;
end
4'b0011: begin//3
HEX0 = 7'b0110000;
HEX1 = 7'b1000000;
end
4'b0100: begin//4
HEX0 = 7'b0011001;
HEX1 = 7'b1000000;
end
4'b0101: begin//5
HEX0 = 7'b0010010;
HEX1 = 7'b1000000;
end
4'b0110: begin//6
HEX0 = 7'b0000010;
HEX1 = 7'b1000000;
end
4'b0111: begin//7
HEX0 = 7'b1111000;
```

```
HEX1 = 7'b1000000;
end
4'b1000: begin//8
HEX0 = 7'b0000000;
HEX1 = 7'b1000000;
end
4'b1001: begin//9
HEX0 = 7'b0010000;
HEX1 = 7'b1000000;
end
4'b1010: begin//10
HEX0 = 7'b1000000;
HEX1 = 7'b1111001;
end
4'b1011: begin//11
HEX0 = 7'b1111001;
HEX1 = 7'b1111001;
end
4'b1100: begin//12
HEX0 = 7'b0100100;
HEX1 = 7'b1111001;
end
4'b1101: begin//13
HEX0 = 7'b0110000;
HEX1 = 7'b1111001;
end
4'b1110: begin//14
HEX0 = 7'b0011001;
HEX1 = 7'b1111001;
end
```

```
4'b1111: begin//15
HEX0 = 7'b0010010;
HEX1 = 7'b1111001;
end
endcase
end
endmodule
```

Results:

The simulation graph is attached below along with the prelab. It provides the same simulation results as in case of cadence but saves lot of time.

Conclusion:

This lab taught us how to design a circuit using Verilog HDL. It taught how to use FPGA to design customized circuits. Using Verilog, we can save time by using codes instead of manually implementing the circuits on cadence.

Advantages of Hardware Description Language (HDL):

HDL's like VHDL, Verilog are used to capture the features of hardware and describe them in a code in order to simulate a circuit. Then burning this code into programmable devices like FPGA's and implementing the customized circuits. Using these HDL's saves significant amount of time rather than implementing circuits manually in cadence and then simulating them. Also debugging becomes an easy task when it comes to designing in Verilog.

D=1							
Inputs		Outputs					
Ai	Bi	P	G	Y	R	Ao	Bo
0	0	0	1	0	0	0	0
0	0	1	1	0	0	0	1
0	1	0	0	0	1	1	0
0	1	1	0	0	1	1	1
1	0	0	0	1	0	0	0
1	0	1	0	1	0	0	1
1	1	0	0	0	1	1	0
1	1	1	0	0	1	1	1

- Boolean Equations for all outputs

For D = 0

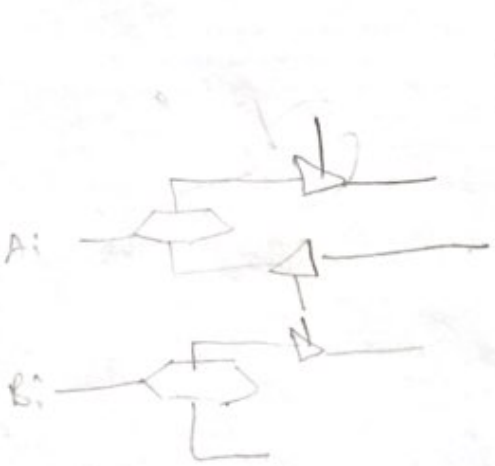
G = Ao'.Bo'

Y = Ao.Bo'

R = Bo

Ai = Bo

Bi = P



For D = 1

G = Ai'.Bi'

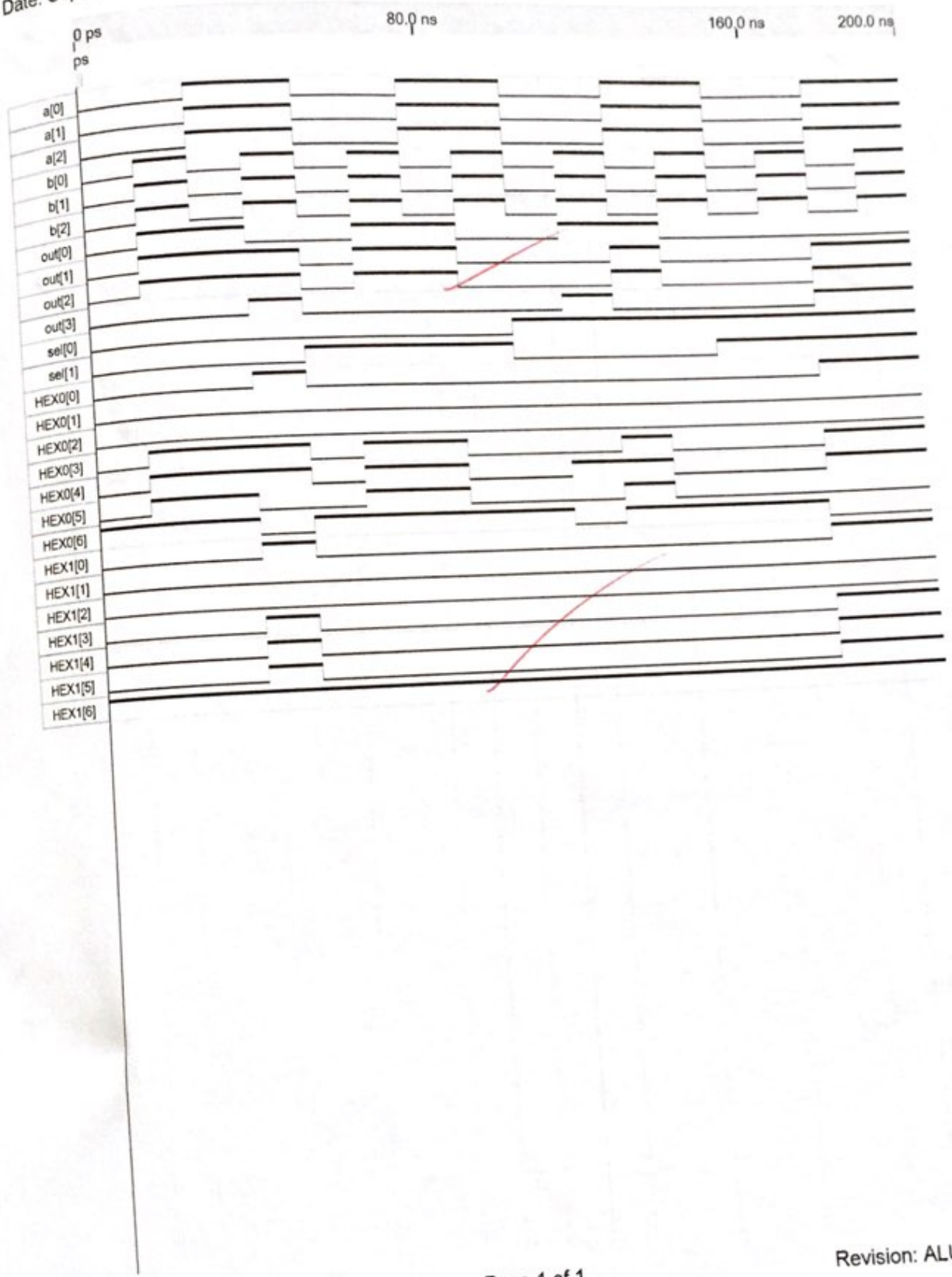
Y = Ai.Bi'

R = Bi

Ao = Bi

Bo = P

- Paper Schematic layout using tri-state buffers and bi-directional ports for 1 controller station. (Handwritten copy attached along with.)


0 ps
80.0 ns
160.0 ns
200.0 ns
ps
a[0]
a[1]
a[2]
b[0]
b[1]
b[2]
out[0]
out[1]
out[2]
out[3]
sel[0]
sel[1]
HEX0[0]
HEX0[1]
HEX0[2]
HEX0[3]
HEX0[4]
HEX0[5]
HEX0[6]
HEX1[0]
HEX1[1]
HEX1[2]
HEX1[3]
HEX1[4]
HEX1[5]
HEX1[6]

Rohan Thaver
8309623

DATE:

ECE 450 - Prelab 2

```
module adder(a,b,c, S, Cout);
input [2:0] a, b, c;
output S, Cout
wire a, b, c, S, Cout;
assign S = a^b^c;
assign Cout = ((a&b) | (b&c)|(a&c));
end module.

module subtractor (a, b, c, S, Cout);
input a, b, c;
output S, Cout;
wire a, b, c, S, Cout;
assign S = a^b^c;
assign Cout = ((a&b') | ((a^b)'&c));
end module

module Xor(a, b, S);
input a, b;
output S;
wire a, b, S;
assign S = a^b
end module.

module Left_Shift (a, S);
input[2:0]a;
output S;
wire a, S;
assign S = a<<1;
end module.
```

DATE:

```
module ALU_3bit(a, b, c, S, Cout, fun_sel, result);
input [2:0] a, b, c;
input [2:0] fun_sel;
output S, Cout;
wire fun_sel, a, b, c, S, Cout;

always @(fun_sel);
begin
 if(fun_sel == 2'b00)
assign result = adder(a, b, c, S, Cout);
 else if(fun_sel == 2'b01)
 assign result = subtractor(a, b, c, S, Cout);
 else if (fun_sel == 2'b10)
 assign result = Xor(a, b, S);
 else if (fun_sel == 2'b11)
 assign result = Left_shift(a, S);
 else
   $display("Invalid fun_sel");
 end
end module.
```

9/15/16