ECE451Lab2

Huanjia Liu (ID:831993149)

Sep 25, 2020

Contents

1 Objective

During this lab, we are asked to learn the basic logic synthesis steps by synthesizing a 3-bit signed Arithmetic Logic Unit (ALU) using Verilog. on PC with a Linux OS for Quartus and ModelSim, FPGA Board.

2 Introduction

Verilog, standardized as IEEE 1364, is a hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the register-transfer level of abstraction.

In Lab2, we should design a 3 bits ALU again like Lab1, but this time, we use more convenient way: Verilog.

There are 4 kinds of input: fun_sel0 and fun_sel1: Selects one of the four functions. ain[2:0] and bin[2:0]: Inputs for the two 3-bit numbers or one 3-bit number. Also there is one output: out[2:0]: The three-bit number that is the result of the ALU operation. Also a little different with Lab1, we need 2 digits 7-segment display control output to show the final result on 7-segment display.

As for our 3 bits ALU, four functions are combined: three-bit addition, subtraction, XOR (bitwise between A and B), and 3-bit shift-left (on A; shift in zero at the least significant bit).

3 Procedure

3.1 Verilog Code

We can divide Verilog Code to three part. Declare part, function part, segment display part.

3.1.1 Declare part

Here we declare three input and three output. Sel means select, which is used to select mode of ALU, the LED_out and LED_outs are the variable to control first and second7-segment.

```
module lab2(A, B, Sel, V_out, LED_out, LED_outs);
```

```
input[2:0] A,B;

input[1:0] Sel;

//output reg C_out;

output reg [3:0] V_out;

output reg [6:0] LED_out;

output reg [6:0] LED_outs;
```

3.1.2 Function part

Here we use Verilog code to achieve four function (addition, subtraction, XOR, and 3-bit shift-left) in different Sel mode. We are asked to display two 7-segment, thus, 4 bit V_out is required.

```
reg [3:0] temp;



always @ (A, B, Sel)
begin
   case(Sel)
      2'b00:
      begin
         V_out = {1'b0, A} + {1'b0, B};
         //C_out = temp[3];
         //V_out = temp[2:0];
      end


      2'b01:
      begin
         V_out = {1'b0, A} - {1'b0, B};
         //C_out = temp [3];
```

```
        //V_out = temp[2:0];
      end
      2'b10: V_out = {1'b0,A^B};
      2'b11:
      begin
         V_out = {1'b0, A} << 1;
         //C_out = temp[3];
         //V_out = temp[2:0];
      end
   endcase
end
```

3.1.3 Display part

Now, we can use the variable V_out come from function part to display number on two 7-segment. 0 means led light on, while 1 means led light off.

```
always @(*)
begin
  case(V_out)
  4'b0000:
  begin
   LED_out = 7'b1000000; // "0"
   LED_outs = 7'b1000000;
  end
   4'b0001:
  begin
   LED_out = 7'b1111001; // "1"
   LED_outs = 7'b1000000;
  end
```

```
 4'b0010:
begin
 LED_out = 7'b0100100; // "2"
 LED_outs = 7'b1000000;
end

 4'b0011:
begin
 LED_out = 7'b0110000; // "3"
 LED_outs = 7'b1000000;
end

 4'b0100:
begin
 LED_out = 7'b0011001; // "4"
 LED_outs = 7'b1000000;
end

 4'b0101:
begin
 LED_out = 7'b0010010; // "5"
 LED_outs = 7'b1000000;
end

 4'b0110:
begin
 LED_out = 7'b0000010; // "6"
 LED_outs = 7'b1000000;
```

```
end

 4'b0111:
begin
 LED_out = 7'b1111000; // "7"
 LED_outs = 7'b1000000;
end

 4'b1000:
begin
 LED_out = 7'b0000000; // "8"
 LED_outs = 7'b1000000;
end

 4'b1001:
begin
 LED_out = 7'b0010000; // "9"
 LED_outs = 7'b1000000;
end

 4'b1010:
begin
 LED_out = 7'b1000000; // "10"
 LED_outs = 7'b1111001;
end

 4'b1011:
begin
 LED_out = 7'b1111001; // "11"
```

```
 LED_outs = 7'b1111001;
end

 4'b1100:
begin
 LED_out = 7'b0100100; // "12"
 LED_outs = 7'b1111001;
end

 4'b1101:
begin
 LED_out = 7'b0110000; // "13"
 LED_outs = 7'b1111001;
end

 4'b1110:
begin
 LED_out = 7'b0011001; // "14"
 LED_outs = 7'b1111001;
end

 4'b1111:
begin
 LED_out = 7'b0010010; // "15"
 LED_outs = 7'b1111001;
end
```

```
  default: LED_out = 7'b1000000; // "0"

  endcase

end
```

3.2 Simulation

Before compiling and downloading code into FPGA board, I create a VWF file in the Quartus to simulate my design.

At first, I simulate function part with 7-segment display. Here are four function test result.

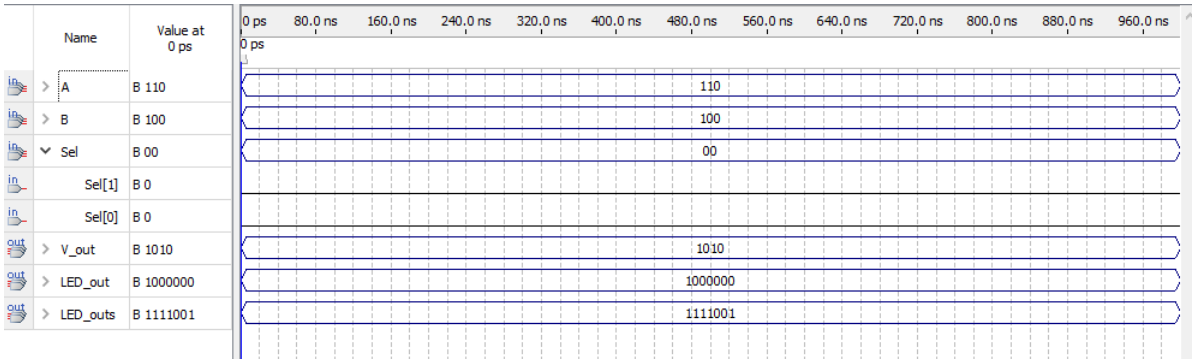


Figure 1: 6(110) plus 4(100)

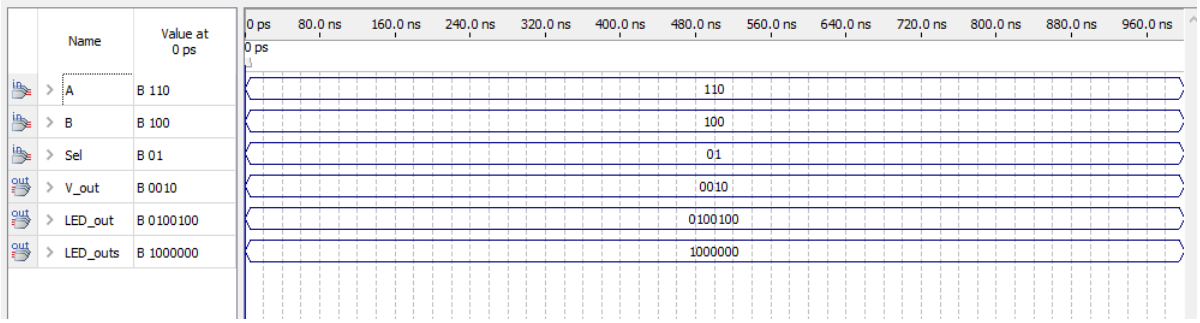


Figure 2: 6(110) plus 4(100)

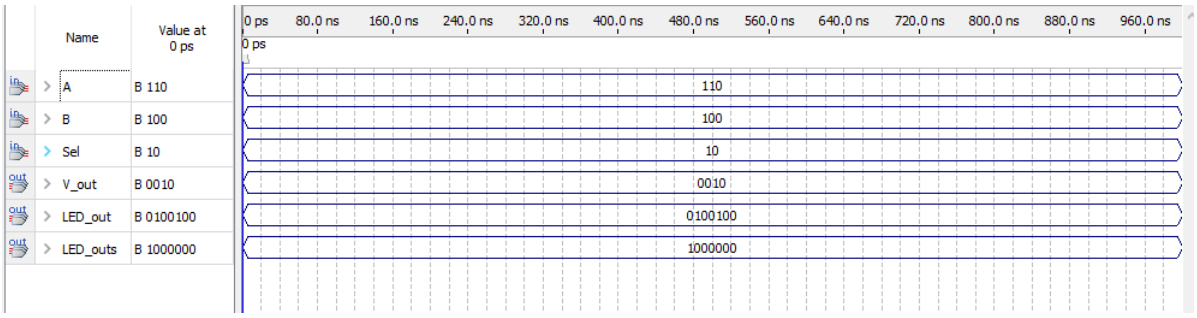


Figure 3: 6(110) xor 4(100)

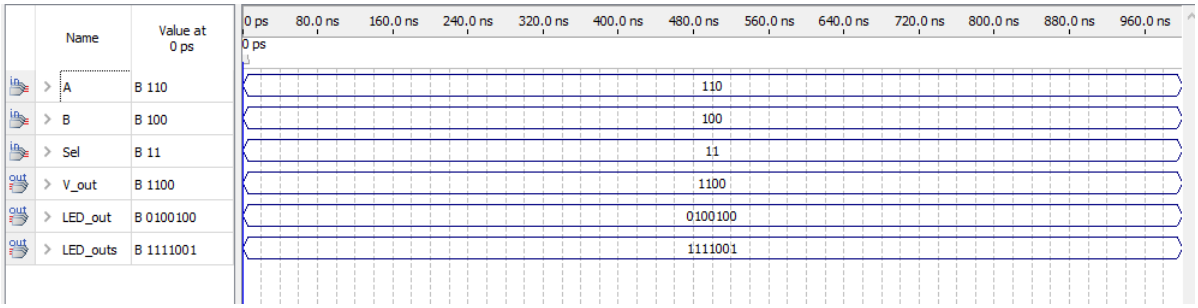
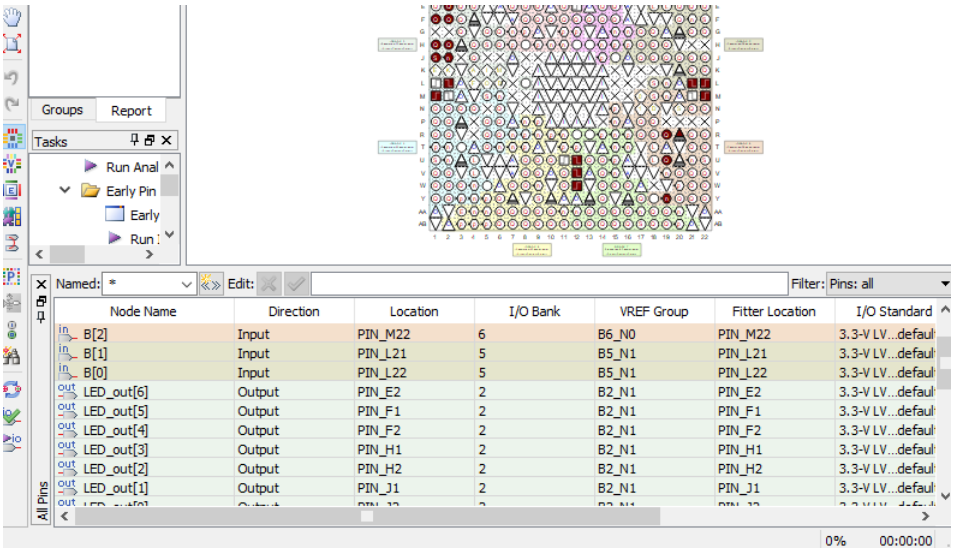


Figure 4: 6(110) left shift 1 bit

3.3 FPGA running

As simulation works pretty well, I edit pin planner and download the code into FPGA, and show the result to TA.



Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard
B[2]	Input	PIN_M22	6	B6_N0	PIN_M22	3.3-V LV...defaul
B[1]	Input	PIN_L21	5	B5_N1	PIN_L21	3.3-V LV...defaul
B[0]	Input	PIN_L22	5	B5_N1	PIN_L22	3.3-V LV...defaul
LED_out[6]	Output	PIN_E2	2	B2_N1	PIN_E2	3.3-V LV...defaul
LED_out[5]	Output	PIN_F1	2	B2_N1	PIN_F1	3.3-V LV...defaul
LED_out[4]	Output	PIN_F2	2	B2_N1	PIN_F2	3.3-V LV...defaul
LED_out[3]	Output	PIN_H1	2	B2_N1	PIN_H1	3.3-V LV...defaul
LED_out[2]	Output	PIN_H2	2	B2_N1	PIN_H2	3.3-V LV...defaul
LED_out[1]	Output	PIN_J1	2	B2_N1	PIN_J1	3.3-V LV...defaul

Figure 5: Pin Planner

4 Question

4.1 Discuss the advantages of a Hardware Descriptive Language (HDL).

Process-independence and significantly shorter design cycles. This allows the engineer to design the actual circuit in the functional design, logic verification stage without having to consider too much about the gate level and the specific details of the process implementation, and only need to use the requirements of the chip in the system design and impose different constraints.

5 Conclusion

Now we can get the conclusion, I finish this lab successfully, and achieve all the objective.