3-BIT ALU IN VERILOG

ECE 451- LAB 02

Ishani Gowaikar
CSU ID- 831702439

3-bit ALU Design - Verilog

Objective: The objective of this lab is to learn the basic logic synthesis steps by synthesizing a 3-bit signed Arithmetic Logic Unit (ALU) using Verilog.

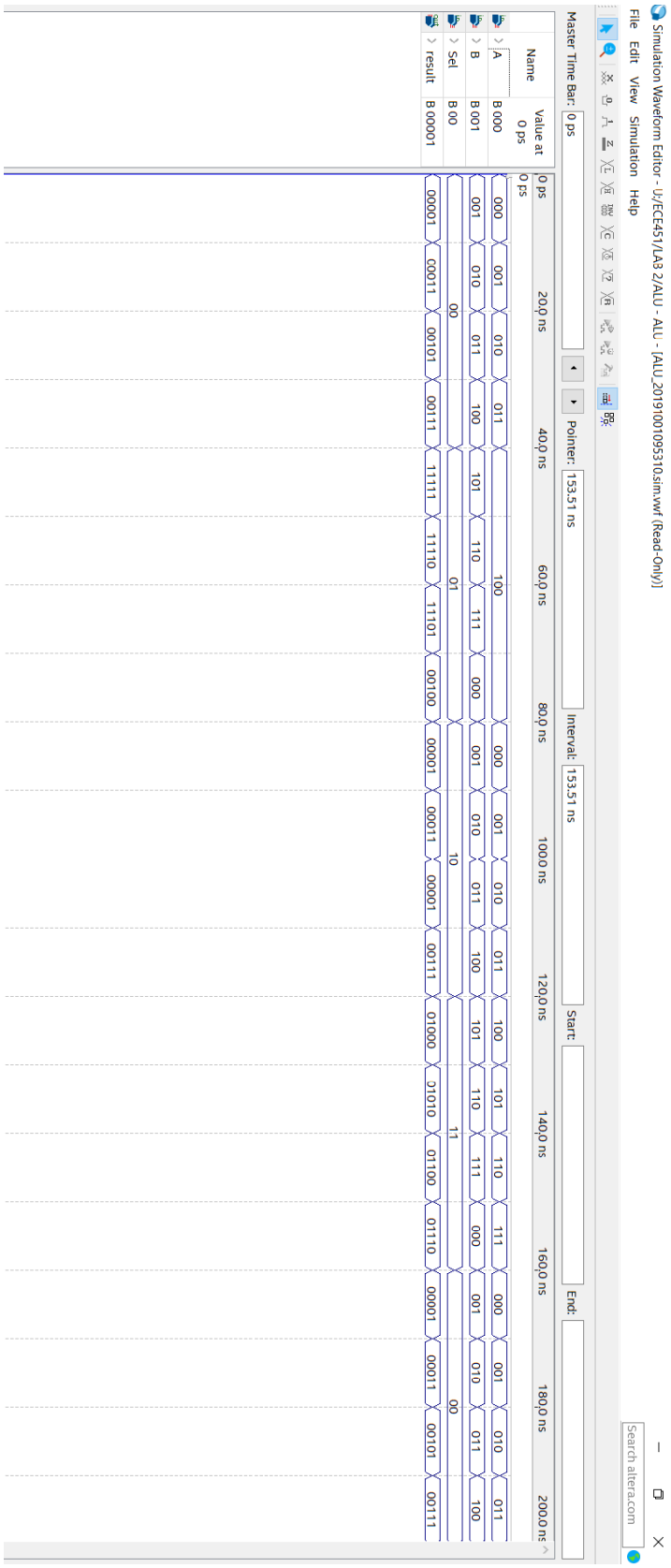
Introduction: The 3-bit ALU designed in lab 1 had 4 functions: ADD, SUB, XOR and Left-shift. Here, we have implemented an ALU module in Verilog to provide same functionality and a 4-bit output (3-bit result and Carry Out). For the implementation on the ALTERA kit, we used SSD Hex0 and Hex1 to display result in decimal form.

Code:

```
module ALU(
          input [2:0]           A,B,
          input [1:0]           Sel,
          output[4:0]           Out,
          output reg [4:0]      result,
          output reg [13:0]     z
          );

    wire[4:0] temp;
    assign Out = result;

    always
    begin
       case (Sel)
       2'b00: result = A+B;
       2'b01: result = A-B;
       2'b10: result = A^B;
       2'b11: result = A<<1;

       endcase


       case (result[4:0])

       5'b00000 :
       z = 14'b10000001000000;              //Hex 0

       5'b00001 :
       z = 14'b10000001111001;              //Hex 1

       5'b00010 :
       z = 14'b10000000100100;              //Hex 2

       5'b00011 :
       z = 14'b10000000110000;              //Hex 3

       5'b00100 :
       z = 14'b10000000011001;              //Hex 4

       5'b00101 :
       z = 14'b10000000010010;              //Hex 5

       5'b00110 :
       z = 14'b10000000000010;              //Hex 6

       5'b00111 :
       z = 14'b10000001111000;              //Hex 7

       5'b01000 :
       z = 14'b10000000000000;              //Hex 8

       5'b01001 :
       z = 14'b10000010010000;              //Hex 9

```

```
55
56          5'b01010 :
57          z = 14'b11110011000000;             //Hex 10
58
59          5'b01011 :
60          z = 14'b11110011111001;             //Hex 11
61
62          5'b01100 :
63          z = 14'b11110010100100;             //Hex 12
64
65          5'b01101 :
66          z = 14'b11110010110000;             //Hex 13
67
68          5'b01110 :
69          z = 14'b11110010011001;             //Hex 14
70
71          5'b01111 :
72          z = 14'b11110010010010;             //Hex 15
73
74          //subtraction
75
76          5'b11111 :
77          z = 14'b01111111111001;             //dec -1
78
79          5'b11110 :
80          z = 14'b01111110100100;             //dec -2
81
82          5'b11101 :
83          z = 14'b01111110110000;             //dec -3
84
85          5'b11100 :
86          z = 14'b01111110011001;             //dec -4
87
88          5'b11011 :
89          z = 14'b01111110010010;             //dec -5
90
91          5'b11010 :
92          z = 14'b01111110000010;             //dec -6
93
94          5'b11001:
95          z = 14'b01111111111000;             //dec -7
96
97          5'b11000 :
98          z = 14'b01111110000000;             //dec -8
99
100         endcase
101
102     end
103
104 endmodule
```

Simulation waveform:



****The result is taken as 5 bit, 1 bit for the over-low flag, to indicate it as a negative signed decimal.**

Advantages of HDL:

Hardware description languages (HDL) are a special class of languages that provide functionalities to capture true hardware behavior in software. Most high-level programming languages do not take hardware parameters like timing, delay etc. into consideration. Thus, such functionalities can be captured using HDLs.

- HDLs can be used to model hardware and related timing functionalities.
- Can be used to synthesize simulations that capture true nature of hardware.