Lab no 05: Arithmetic and Logic Operations (ALU)

The purpose of this Lab is to learn to:

1) Implement a 2-bits Arithmetic and Logic Operations (ALU) on FPGA. You will write the behavioral description for the ALU using Verilog.
2) Connect sub-modules on a top-level module. You will connect the ALU and the seven-segment decoder.

   Be aware: In this lab, the seven-segment decoder is written in behavioral description using a case statement, not logical equations like Lab 2.
3) Use the switches and seven-segment on the FPGA board to display the output of the ALU.

Parts: -

1. Code the behavioral description of the 2-bit ALU.
2. Code the behavioral description of the seven-segment decoder.
3. Connect the ALU and the seven-segment decoder on the top-level module and run it on FPGA.

Part 1. Code the behavioral description of the 2-bit ALU

One of the core components of a CPU is the Arithmetic and Logic Unit (ALU). This component is responsible for performing the calculations (such as addition) on data when executing a machine code instruction. It is like a simple calculator where you set the inputs and the operation, giving you the result.

In this lab, we will design and build a basic ALU that is capable of eight operations, as follows:

ALU Selection	ALU Operation	ALU Output
000	ALU_Out = A and B;	On 2 Leds
001	ALU_Out = A or B;	On 2 Leds
010	ALU_Out = A + B;	On 1 Seven Segment
011	ALU_Out = A - B;	On 1 Seven Segment
100	ALU_Out = A * B;	On 1 Seven Segment
101	ALU_Out = A > B;	Draw A on Seven Segment
110	ALU_Out = A < B;	Draw B on Seven Segment
111	ALU_Out = A == B;	Draw = on Seven Segment

Behavioral description Verilog code for ALU

```
module ALU ( A, B, ALU_sel, ALU_result);
input [1:0] A,B; // ALU 2-bit Inputs
input [2:0] ALU_sel; // ALU 3-bit Selector
output [3:0] ALU_result; // ALU 4-bit Output
reg [3:0] ALU_result;
always @(*)
    begin
        case(ALU_sel)
        3'b000: // Logical and
           ALU_result = A & B;
        3'b001: // Logical or
           ALU_result = A | B;
        3'b010: // Addition
           ALU_result = A + B;
        3'b011: // Subtraction
           ALU_result = A - B;
        3'b100: // Multiplication
           ALU_result = A * B;
        3'b101: // Greater comparison
           ALU_result = A > B? 4'b1010:4'b1011;
        3'b110: // lower comparison
           ALU_result = A < B? 4'b1011:4'b1010;
        3'b111: // Equal comparison
           ALU_result = A == B? 4'b1100:4'b0000;
        default: ALU_result = 4'b0 ;
        endcase
    end
endmodule
```

Part 2. Code the behavioral description of the seven-segment decoder

Seven segments decoder using the **Case statement**. A seven-segment display decoder takes a 4-bit data input BCD and produces seven outputs to control light-emitting diodes to display a digit from 0 to 9 and comparison outputs.

Behavioral description Verilog code for seven segments decoder.

```
module sevenSegments (bcd,dec);
input [3:0]bcd;
output reg [6:0] dec;
always@(bcd)
          begin
               case(bcd)
                    4'b0000 : dec = ~7'b1111110; // 0
                    4'b0001 : dec = ~7'b0110000; // 1
                    4'b0010 : dec = ~7'b1101101; // 2
                    4'b0011 : dec = ~7'b1111001; // 3
                    4'b0100 : dec = ~7'b0110011; // 4
                    4'b0101 : dec = ~7'b1011011; // 5
                    4'b0110 : dec = ~7'b1011111; // 6
                    4'b0111 : dec = ~7'b1110000; // 7
                    4'b1000 : dec = ~7'b1111111; // 8
                    4'b1001 : dec = ~7'b1111011; // 9
                    4'b1010 : dec = ~7'b1110111; // A
                    4'b1011 : dec = ~7'b1111111; // B
                    4'b1100 : dec = ~7'b1001000; // =
               default : dec = ~7'b1111110; //0 default
               endcase
          end
endmodule
```

Part 3. Connect the ALU and the seven-segment decoder on the top-level module and run it on FPGA

```
module ALU_to_decoder (A, B, ALU_sel,Segmentleds,leds);
input [1:0] A,B; // ALU 2-bit Inputs
input [2:0] ALU_sel; // ALU 3-bit Selector
output [6:0] Segmentleds; //Seven segment leds
output [1:0] leds; // Leds
wire [3:0] ALU_result; // ALU 4-bit Output
// Instantiation of the  ALU
ALU ALU_dut(A, B, ALU_sel, ALU_result);
// Instantiation of the  Seven segment //Display Output on Seven segments
sevenSegments sevenSegments_dut (ALU_result,Segmentleds);
assign leds = ALU_result [1:0]; //Diplay Output on Leds
endmodule
```

Run the integrated design (ALU + Decoder) on FPGA.

- Refer to Lab 2 to program the FPGA by **Quartus**
- Use DE-10lite kit, Altera MAX 10 based FPGA board
- Check DE10-lite user manual (Here) for pin assignment.

Note: Inputs/Outputs Assignment, Open the **Assignment** tab, click on **Pin Planner**, and assign pins as shown in the figure below.

All Pins

Named: * Edit:

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard
in A[1]	Input	PIN_C11	7	B7_N0	PIN_C11	2.5 V
in A[0]	Input	PIN_C10	7	B7_N0	PIN_C10	2.5 V
in ALU_Sel[2]	Input	PIN_A13	7	B7_N0	PIN_A13	2.5 V
in ALU_Sel[1]	Input	PIN_B12	7	B7_N0	PIN_B12	2.5 V
in ALU_Sel[0]	Input	PIN_A12	7	B7_N0	PIN_A12	2.5 V
in B[1]	Input	PIN_C12	7	B7_N0	PIN_C12	2.5 V
in B[0]	Input	PIN_D12	7	B7_N0	PIN_D12	2.5 V
out Segmentleds[6]	Output	PIN_C14	7	B7_N0	PIN_C14	2.5 V
out Segmentleds[5]	Output	PIN_E15	7	B7_N0	PIN_E15	2.5 V
out Segmentleds[4]	Output	PIN_C15	7	B7_N0	PIN_C15	2.5 V
out Segmentleds[3]	Output	PIN_C16	7	B7_N0	PIN_C16	2.5 V
out Segmentleds[2]	Output	PIN_E16	7	B7_N0	PIN_E16	2.5 V
out Segmentleds[1]	Output	PIN_D17	7	B7_N0	PIN_D17	2.5 V
out Segmentleds[0]	Output	PIN_C17	7	B7_N0	PIN_C17	2.5 V
out leds[1]	Output	PIN_A9	7	B7_N0	PIN_A9	2.5 V
out leds[0]	Output	PIN_A8	7	B7_N0	PIN_A8	2.5 V
<<new node>>						