# Lab 9 ALU and Seven-Segment Display

**Objective**

The objectives of this lab are to be familiar with the behavioral modeling of circuits and to design a 4-bit ALU capable of performing various arithmetic and logic operations and a seven-segment display decoder to display results.

1. **ALU Design**

An arithmetic logic unit (ALU) is a combinational circuit used to perform arithmetic and logic operations. It represents the fundamental building block of the central processing unit (CPU) of a computer. The block diagram of the ALU to design in this lab is shown below and can perform eight different operations as shown in the table.

|  |  |
| --- | --- |
| operation | result = |
| 0 0 0 | A and B |
| 0 0 1 | A or B |
| 0 1 0 | Not B |
| 0 1 1 | A << B |
| 1 0 0 | A + B |
| 1 0 1 | A – B |
| 1 1 0 | A\* B |
| 1 1 1 | A ÷ B |

4

result

A

4-bit

ALU

4

B

4

3

operation

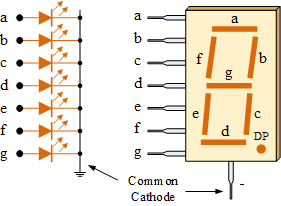
Write a Verilog code to implement the ALU in behavioral level

|  |
| --- |
| module Alu(A,B,operation,result);  input [3:0] A,B;  input [2:0] operation;  output [3:0] result;  reg [3:0] result; // redeclare the signals that appear on the left-hand side of the  // assignment statements inside the always block  always @(A,B,operation)  begin  case(operation)  3’b000: result = A & B;  3'b001: result = A | B;  3'b010: result = ~B; // bit-wise operator and returns the invert of the argument  // use ! for if true or false of single bit  3'b011: result = A << B;  3'b100: result = A + B;  3'b101: result = A - B;  3'b110: result = A \* B;  3'b111: result = A / B;  endcase  end  endmodule |

1. **Seven-Segment Display Decoder:** design a 7-segment display driver to display a 4-bit binary number.

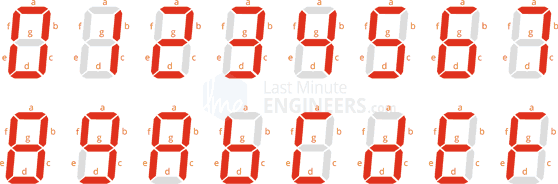
Seven segment display uses seven different and individual LEDs to display a hexadecimal symbol. It has 7 wires to control the individual LED, one wire to control the decimal point and one enable wire.

1. Derive the truth-table for a 7 segment display decoder. This circuit inputs a 4-bit binary number x and provides **active low outputs** seg[6:0] for a 7-segment decoder.

**If you put a 0 on A it will light up segment A**

**If you put a 1 on A it will NOT light up segment A**

|  |  |  |
| --- | --- | --- |
|  | x | a b c d e f g (seg) |
| 0 | 0000 | 0 0 0 0 0 0 1 |
| 1 | 0001 | 1 0 0 1 1 1 1 |
| 2 | 0010 | 0 0 1 0 0 1 0 |
| 3 | 0011 | 0 0 0 0 1 1 0 |
| 4 | 0100 | 1 0 0 1 1 0 0 |
| 5 | 0101 | 0 1 0 0 1 0 0 |
| 6 | 0110 | 0 1 0 0 0 0 0 |
| 7 | 0111 | 0 0 0 1 1 1 1 |
| 8 | 1000 | 0 0 0 0 0 0 0 |
| 9 | 1001 | 0 0 0 0 1 0 0 |
| 10 | 1010 | 0 0 0 1 0 0 0 |
| 11 | 1011 | 1 1 0 0 0 0 0 |
| 12 | 1100 | 0 1 1 0 0 0 1 |
| 13 | 1101 | 1 0 0 0 0 1 0 |
| 14 | 1110 | 0 1 1 0 0 0 0 |
| 15 | 1111 | 0 1 1 1 0 0 0 |



1. Implement the binary to 7 segment decoder/driver behaviorally in Verilog

**module bin7seg (x, seg, dp);**

input [3:0] x ; //4-bit input to display

output [6:0] seg; // segments from a to g

output dp; // decimal point

**reg [6:0] seg; // re-declare as the type of reg**

**always** @(x)

**case** (x)

0: seg = 7'b0000001;

1: seg = 7'b1001111;

2: seg = 7'b0010010;

3: seg = 7'b0000110;

4: seg = 7'b1001100;

5: seg = 7'b0100100;

6: seg = 7'b0100000;

7: seg = 7'b0001111;

8: seg = 7'b0000000;

9: seg = 7'b0000100;

10: seg = 7'b0001000;

11: seg = 7'b1100000;

12: seg = 7'b0110001;

13: seg = 7'b1000010;

14: seg = 7'b0110000;

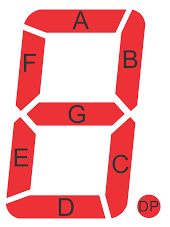
15: seg = 7'b0111000;

**default**: seg = 7'b1111110;

**endcase**

**endmodule**

1. **Write a Verilog module to display the ALU result on a 7- segment display and turn off the decimal point.**

****

7

x

7-segment decoder

result

A

4-bit

ALU

seg

4

4

B

dp

4

3

operation

|  |
| --- |
| module toplevelmodule(A,B,operation, seg, dp);  // result and x get tied together internally , make it a wire  input [3:0] A, B;  input [2:0] operation;  wire result;  output [6:0] seg;  output dp;    // turn off decimal point  // active low -> 1 not 0  assign dp = 1;    // instantiate the ALU  // module Alu(A,B,operation,result);  ALU ALU(A, B, operation, result);    // instantiate the 7-seg display decoder  // module bin7seg(x,seg,dp);  bin7seg bin7seg(result,seg,dp);  endmodule |

1. **Testbench: Write a testbench to test both ALU and the toplevelmodule**

On the EDAplayground.com, create a Verilog testbench to test your ALU and toplevelmodule, and perform the simulation to check if the results are correct. Test all 8 functions of the ALU by setting A = 3, and B = 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| operation | A = 3 | B = 2 | Result = | seg |
| 0 0 0 | 0011 | 0010 | 2 | 0010010 |
| 0 0 1 | 0011 | 0010 | 3 | 0000110 |
| 0 1 0 | 0011 | 0010 | 13 | 1000010 |
| 0 1 1 | 0011 | 0010 | 12 | 0110001 |
| 1 0 0 | 0011 | 0010 | 5 | 0100100 |
| 1 0 1 | 0011 | 0010 | 1 | 1001111 |
| 1 1 0 | 0011 | 0010 | 6 | 0100000 |
| 1 1 1 | 0011 | 0010 | 1 | 1001111 |

|  |
| --- |
| // Code your testbench here  // or browse Examples  module test;  // inputs  reg [3:0] A,B;  reg [2:0] operation;  // outputs  wire [3:0] result;  wire [6:0] seg;  wire dp;    // instantiate the ALU  Alu uut0(A, B, operation, result);    // instantiate toplevelmodule  toplevelmodule uut1(A,B,operation, seg, dp);    initial  begin  $dumpfile("dump.vcd");  $dumpvars(1,test);    // display the inputs and outputs  $monitor("%b %d %d %d %b", operation, A,B, result, seg);    // initialize inputs  A = 3; B = 2;  for(int i = 0; i < 8; i = i + 1) begin  #10 operation = i; end  #10 $finish;  end  endmodule |

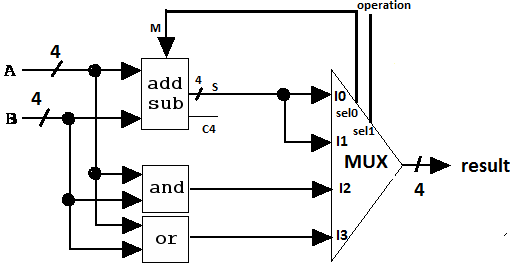
**Graphical user interface, text

Description automatically generated with medium confidenceCHECKING OUTPUT :**

https://www.edaplayground.com/x/BsXs

1. **Homework**: Write a Verilog code and testbench to implement another 4-bit ALU capable of performing four operations (AND, OR, ADD, SUB) based on the diagram shown below. In your code, you use the **adder\_subtractor** module from Lab 8 to perform the addition and subtraction, and the given **mux4x1** to selection the operation.

**module** Alu(A, B, operation, result);

//inputs and outputs

input [1:0] operation;

……

// Instantiate AND gate and OR gate

……

// connect M to operation[0]

……

// Instantiate add\_subtractor

……

// Instantiate mux4x1

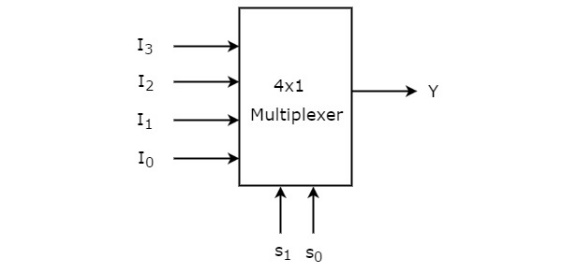
…….

**endmodule**

**module** mux4x1(i0, i1, i2, i3, select, y);

input [3:0] i0,i1,i2,i3;

input [1:0] select;

 output [3:0] y;

reg [3:0] y;

always @ (i0 or i1 or i2 or i3 or select)

case (select)

2'b00: y = i0;

2'b01: y = i1;

2'b10: y = i2;

2'b11: y = i3;

endcase

**endmodule**

**Submission Instructions:**

Lab work submission

1. Take a screenshot of your wavefroms.
2. Add the following information as comments to the beginning of your code. Make sure to click the “Save” button to save your project, then take a screenshot of your code.

// Author: Name

// Lab 9: put the title here

// Link to your project

1. Copy the link of your design from the address bar of the browser.
2. On the Blackboard, click on Lab 9. Attach the screenshot from the first two steps and paste the link from Step 3 into the Comments area, then click the “Submit” button.

Lab report submission

Lab report is needed for this lab. Please follow the guidelines and sample report on the Blackboard when you are writing your lab reports. Click on **Lab 9 Report Submission** to submit your report. It is due one week after the lab is done.