

PROJECT 2

CE 6325.001 VLSI DESIGN

**PROJECT 2:
CELL REPORT FOR
32-BIT ARITHMETIC LOGIC UNIT (ALU)**

TEAM MEMBERS:

- 1) ALEXANDRA EDWINRAJ (NET ID: AXE210023)**
- 2) HEENISHA REDDY BACHUGUDEM (NET ID: HXR210022)**
- 3) KANUPRIYA SHARMA (NET ID: KXS220016)**

Introduction:

In this Project, we have 2 16-bit Inputs which perform functions depending on the selected line. We have found the cell count for the ALU Module using Synopsys Design Vision (CAD Tool). The Design Vision software read our Behavioral Verilog code and then used the library to generate a mapped netlist file. This mapped the netlist file along with the concatenated header.v code was simulated again in ModelSim using the same testbench code. The waveform obtained for the mapped netlist code is compared with the waveforms from the behavioral code.

Block diagram

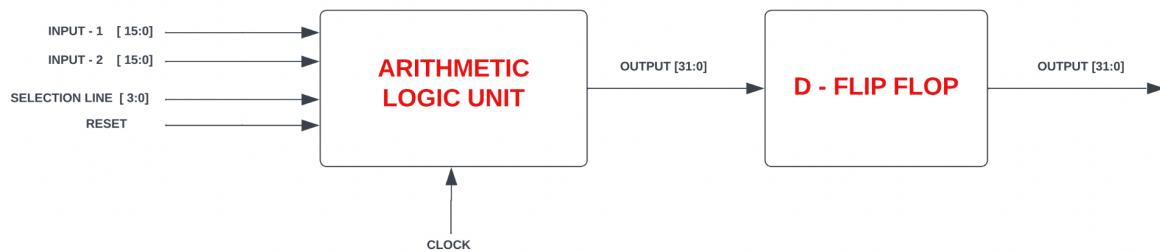


Figure 1 - Block diagram of the ALU with D-flip flop

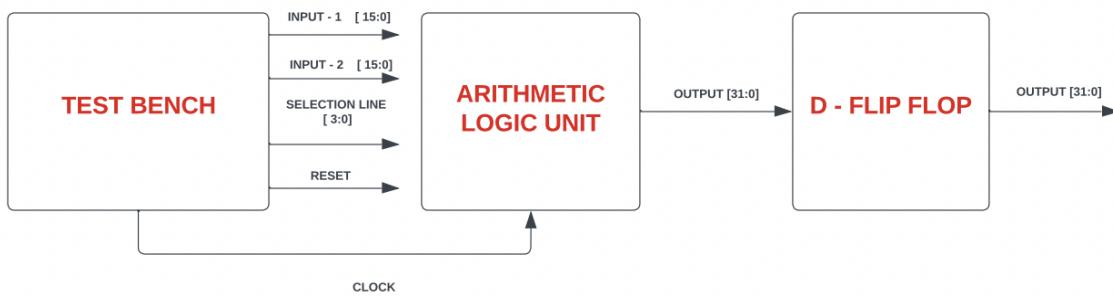


Figure 2 - Block diagram of the test bench connection with ALU

Behavioural Code and Testbench for 32-bit Arithmetic Logic Unit (ALU)

Behavioural Code for 32-bit Arithmetic Logic Unit (ALU):

// Arithmetic Logic Unit

module alu(c, s, r, X, Y, Z);

input c,r; // inputs- clock and reset

input [3:0] s; // select line for operations

input [15:0] X,Y; // 16-bit inputs

output reg [31:0] Z; // 32-bit output

reg [31:0] out; //output register

wire [31:0] w; //output wire

parameter Add= 4'd0, Subtract=4'd1, Multiply= 4'd2, Divide=4'd3, //Arthematic Operations

Shift_left =4'd4, Shift_right= 4'd5, Rotate_left=4'd6, Rotate_right = 4'd7, //Shift & Rotate

AND = 4'd8, OR = 4'd9, NOR = 4'd10, NAND = 4'd11, XOR = 4'd12, XNOR = 4'd13, //Logic Operations

Set_less_than = 4'd14, set_if_equal=4'd15; //less than & equal to

always @ (negedge c) // Triggers the block on the negative edge of a clock signal only

Begin

if (r==1'b0)

out<=32'b0; // active low reset

else

case (s) //Based on the select line

Add : out <= X + Y ; // Addition between two inputs

Subtract : out <= X - Y ; // Subtraction between two inputs

```

Multiply : out <= X * Y ; // Multiplication between two inputs

Divide : out <= X / Y ; // Division between two inputs

Shift_left : out <= X << 1; // Input is shifted to the left by 1 bit

Shift_right : out <= X >> 1; // Input is shifted to left by 1 bit

Rotate_left : out <= {X [14:0], X[15]}; // Input is rotated to the left by 1 bit

Rotate_right : out <= {X [0], X[15:1]}; // Input is rotated to the right by 1 bit

AND : out <= X & Y; // Logical AND function is performed between two inputs

OR : out <= X | Y; // Logical OR function is performed between two inputs

NOR : out <= ~ (X | Y); // Logical NOR function is performed between two inputs

NAND : out <= ~ (X & Y); // Logical NAND function is performed between two inputs

XOR : out <= X ^ Y; // Logical XOR function is performed between two inputs

XNOR : out <= ~ (X ^ Y) ; // Logical XNOR function is performed between two inputs

Set_less_than : out <= (X < Y)? 32'b1 : 32'b0; // Output is set to 1 if first input is less than
second input

set_if_equal : out <= (X == Y)? 32'b1 : 32'b0; // Output is set to 1 if first input is equal to second
input

default : out <= 32'd0; // For any other values, the output is reset

Endcase

end

//D Flip-flop

d_ff d1(out,c,r,w); //D Flip-flop inputs and outputs

always@(negedge c) // Triggers the block on the negative edge of a clock signal only

begin

Z<=w;

```

```

end

endmodule

module d_ff(d,c,r,w);
input c,r; // inputs- clock & reset
input [31:0]d; // 32-bit input
output reg[31:0]w; // 32-bit output register
always @( negedge c)
begin
if (r == 1'b0)
w<=64'b0; // active low reset
else
w<=d;
end
endmodule

```

Testbench for 32-bit ALU:

```

//Test Bench Module for Arithmetic Logic Unit
module alu_tb;
reg c; //Clock
reg r; //Reset
reg [3:0] s; //Select
reg [15:0] X; //Input
reg [15:0] Y; //Input
wire [31:0] Z; //Output

// Instantiating the unit under test
alu uut (
.c(c),
.r(r),
.s(s),

```

```

.X(X),
.Y(Y),
.Z(Z)
);

initial begin
c=0; r=0; s=4'h0; X=16'h0; Y=16'h0; #100; // Reset and wait for 100ns
r=1; s=4'h0; X=16'h28; Y=16'h14; #100; // Performing addition and waiting for 100ns
r=1; s=4'h1; X=16'h28; Y=16'h14; #100; // Performing subtraction and waiting for 100ns
r=1; s=4'h2; X=16'h28; Y=16'h14; #100; // Performing multiplication and waiting for 100ns
r=1; s=4'h3; X=16'h28; Y=16'h14; #100; // Performing division and waiting for 100ns
r=1; s=4'h4; X=16'h28; Y=16'h14; #100; // Shifting to the left by one bit and waiting for 100ns
r=1; s=4'h5; X=16'h28; Y=16'h14; #100; // Shifting to the right by one bit and waiting for 100ns
r=1; s=4'h6; X=16'h28; Y=16'h14; #100; // Rotating to the left by one bit and waiting for 100ns
r=1; s=4'h7; X=16'h28; Y=16'h14; #100; // Rotating to the right by one bit and waiting for 100ns
r=1; s=4'h8; X=16'h28; Y=16'h14; #100; // Performing logical AND operation and waiting for
100ns
r=1; s=4'h9; X=16'h28; Y=16'h14; #100; // Performing logical OR operation and waiting for
100ns
r=1; s=4'ha; X=16'h28; Y=16'h14; #100; // Performing logical NOR operation and waiting for
100ns
r=1; s=4'hb; X=16'h28; Y=16'h14; #100; // Performing logical NAND operation and waiting for
100ns
r=1; s=4'hc; X=16'h28; Y=16'h14; #100; // Performing logical XOR operation and waiting for
100ns
r=1; s=4'hd; X=16'h28; Y=16'h14; #100; // Performing logical XNOR operation and waiting for
100ns
r=1; s=4'he; X=16'h28; Y=16'h14; #100; // Setting the output to 1 if "X" is less than "Y", or
setting the output to 0 and waiting for 100ns
r=1; s=4'hf; X=16'h28; Y=16'h14; #100; // Setting the output to 1 if "X" is equal to "Y", or
setting the output to 0 and waiting for 100ns
#100; // delay
$finish; // end of simulation
end
always #50 c = ~c; // generating the clock signal every 50ns by toggling
endmodule

```

Cell Report by Synopsys Design Vision:

The Cell Report for ALU has two 16-bit Inputs and 32-bit output; We have found the cell count for the ALU Module using Synopsys Design Vision - CAD tool and the report was generated with zero errors. The output pictures show the cell reports from a compiled result by checking the **ungroup** option.

Number of Cells: 3536

A screenshot of the Synopsys Design Vision software interface. The window title is "NoMachine - VLSI_Design". The main pane displays a cell report with the following details:

```
*****  
Report : cell  
Design : alu  
Version: 0-2018.06-SP1  
Date   : Sun Sep 18 01:22:33 2022  
*****  
Attributes:  
b - black box (unknown)  
h - hierarchical  
n - noncombinational  
r - removable  
u - contains unmapped logic  
Cell      Reference    Library     Area  Attributes  
C62       nand2       library    1.000000  
C63       nand2       library    1.000000  
C66       nand2       library    1.000000  
C82       nand2       library    1.000000  
C83       nand2       library    1.000000  
C89       nand2       library    1.000000  
C95       nand2       library    1.000000  
C102      nand2       library    1.000000  
C106      nand2       library    1.000000  
C131      nand2       library    1.000000  
C149      nand2       library    1.000000  
C150      nand2       library    1.000000  
C315      nand2       library    1.000000  
C316      nand2       library    1.000000  
.....  
-----  
Total 3536 cells
```

Below the report, there is a command window containing Verilog code:

```
1 gui_start  
2 analyze -library WORK -format verilog {{/home/011/a/ax/axe210023/project 1/ALU_File_Verilog.v} {{/home/011/a/ax/axe210023/project 1/TestBench_ALU_File.v}}  
3 elaborate alu -architecture verilog -library DEFAULT  
4 compile -exact map -ungroup_all -auto_ungroup area -only_design_rule  
5 uplevel #0 { report cell }
```

At the bottom of the window are buttons for "Edit", "Execute", and "Save Contents As ...".

A screenshot of the Synopsys Design Vision software interface, similar to the previous one but showing the results of the "ungroup" option. The window title is "NoMachine - VLSI_Design". The main pane displays a cell report with the following details:

```
*****  
Report : cell  
Design : alu  
Version: 0-2018.06-SP1  
Date   : Sun Sep 18 01:22:33 2022  
*****  
Attributes:  
b - black box (unknown)  
h - hierarchical  
n - noncombinational  
r - removable  
u - contains unmapped logic  
Cell      Reference    Library     Area  Attributes  
U2126     xor2        library    3.000000  
U2127     xor2        library    3.000000  
U2128     xor2        library    3.000000  
U2129     xor2        library    3.000000  
U2130     xor2        library    3.000000  
U2131     xor2        library    3.000000  
U2132     xor2        library    3.000000  
U2133     xor2        library    3.000000  
U2134     xor2        library    3.000000  
U2135     xor2        library    3.000000  
U2136     xor2        library    3.000000  
U2137     xor2        library    3.000000  
U2138     xor2        library    3.000000  
U2139     xor2        library    3.000000  
U2140     xor2        library    3.000000  
U2141     xor2        library    3.000000  
U2142     xor2        library    3.000000  
U2143     xor2        library    3.000000  
U2144     xor2        library    3.000000  
U2145     xor2        library    3.000000  
U2146     xor2        library    3.000000  
U2147     xor2        library    3.000000  
U2148     xor2        library    3.000000  
U2149     xor2        library    3.000000  
U2150     xor2        library    3.000000  
U2151     xor2        library    3.000000  
U2152     xor2        library    3.000000  
U2153     xor2        library    3.000000  
U2154     xor2        library    3.000000  
U2155     xor2        library    3.000000  
.....  
-----  
Total 3536 cells
```

At the bottom of the window is a message: "design_vision>".

NoMachine - VLSI_Design

```

mult_25[FS_1/U6_0_6_2] oai12 library 2.000000
mult_25[FS_1/U6_0_6_3] oai12 library 2.000000
mult_25[FS_1/U6_0_7_1] oai12 library 2.000000
mult_25[FS_1/U6_1_1_1] oai12 library 2.000000
mult_25[FS_1/U6_1_1_2] oai12 library 2.000000
mult_25[FS_1/U6_1_1_3] oai12 library 2.000000
out_reg[8] dff library 7.000000 n
out_reg[1] dff library 7.000000 n
out_reg[2] dff library 7.000000 n
out_reg[3] dff library 7.000000 n
out_reg[4] dff library 7.000000 n
out_reg[5] dff library 7.000000 n
out_reg[6] dff library 7.000000 n
out_reg[7] dff library 7.000000 n
out_reg[8] dff library 7.000000 n
out_reg[9] dff library 7.000000 n
out_reg[10] dff library 7.000000 n
out_reg[11] dff library 7.000000 n
out_reg[12] dff library 7.000000 n
out_reg[13] dff library 7.000000 n
out_reg[14] dff library 7.000000 n
out_reg[15] dff library 7.000000 n
out_reg[16] dff library 7.000000 n
out_reg[17] dff library 7.000000 n
out_reg[18] dff library 7.000000 n
out_reg[19] dff library 7.000000 n
out_reg[20] dff library 7.000000 n
out_reg[21] dff library 7.000000 n
out_reg[22] dff library 7.000000 n
out_reg[23] dff library 7.000000 n
out_reg[24] dff library 7.000000 n
out_reg[25] dff library 7.000000 n
out_reg[26] dff library 7.000000 n
out_reg[27] dff library 7.000000 n
out_ran[29] nff library 7.000000 n

Hier.1 Report.1
r79/ULTI2_5 nand2 library 1.000000
r79/ULTI2_6 nand2 library 1.000000
r79/ULTI2_7 nand2 library 1.000000

Log History
design_vision>

```

NoMachine - VLSI_Design

```

r79/ULTI1_3 nand2 library 1.000000
r79/ULTI1_4 nand2 library 1.000000
r79/ULTI1_5 nand2 library 1.000000
r79/ULTI1_6 nand2 library 1.000000
r79/ULTI1_7 nand2 library 1.000000
r79/ULTI1_8 nand2 library 1.000000
r79/ULTI1_9 nand2 library 1.000000
r79/ULTI1_10 nand2 library 1.000000
r79/ULTI1_11 nand2 library 1.000000
r79/ULTI1_12 nand2 library 1.000000
r79/ULTI1_13 nand2 library 1.000000
r79/ULTI1_14 nand2 library 1.000000
r79/ULTI12 nand2 library 1.000000
r79/ULTI2_1 nand2 library 1.000000
r79/ULTI2_2 nand2 library 1.000000
r79/ULTI2_3 nand2 library 1.000000
r79/ULTI2_4 nand2 library 1.000000
r79/ULTI2_5 nand2 library 1.000000
r79/ULTI2_6 nand2 library 1.000000
r79/ULTI2_7 nand2 library 1.000000
r79/ULTI2_8 nand2 library 1.000000
r79/ULTI2_9 nand2 library 1.000000
r79/ULTI2_10 nand2 library 1.000000
r79/ULTI2_11 nand2 library 1.000000
r79/ULTI2_12 nand2 library 1.000000
r79/ULTI2_13 nand2 library 1.000000
r79/ULTI2_14 nand2 library 1.000000
r79/UNGT0 nand2 library 1.000000
r79/UNL10 nand2 library 1.000000
----- Total 3536 cells 6077.000000
***** End Of Report *****

Hier.1 Report.1
r79/ULTI2_5 nand2 library 1.000000
r79/ULTI2_6 nand2 library 1.000000
r79/ULTI2_7 nand2 library 1.000000

Log History
design_vision>

```

The Cell report has the D - Flip flop and the Logic Gates

Mapped Net List:

NoMachine - VLSI_Design
ALU_PROJECT_2..CELLS..syn.v

```
ALU_File_Verilog.v ALU_PROJECT_2..CELLS..syn.v

// Generated by: Synopsys DC Expert(TM) in wire load mode
// Version : 0-2018.06-SP1
// Date   : Wed Sep 21 12:34:21 2022
///////////////////////////////////////////////////////////////////




`e alu ( c, s, r, X, Y, Z );
`ut [3:0] s;
`ut [15:0] X;
`ut [15:0] Y;
`put [31:0] Z;
`ut c, r;
`e N22, N23, N24, N25, N26, N27, N28, N29, N30, N31, N32, N33, N34, N35,
N36, N37, N38, N39, N40, N41, N42, N43, N44, N45, N46, N47, N48, N49,
N50, N51, N52, N53, N54, N55, N56, N57, N58, N59, N60, N61, N62, N63,
N64, N65, N66, N67, N70, N72, N73, N74, N75, N76, N77, N78, N79, N80,
N81, N82, N83, N84, N85, N86, N87, N88, N89, N90, N91, N92, N93, N94,
N95, N96, N97, N98, N99, N100, N101, N102, N103, N104, N105, N106,
N107, N108, N109, N110, N111, N116, N117, N118, N119, N120, N121,
N122, N123, N124, N125, N126, N127, N128, N129, N130, N131, N132,
N133, N134, N135, N136, N137, N138, N139, N140, N141, N142, N143,
N144, N145, N146, N147, N148, N149, N150, N151, N152, N153, N154,
N155, N156, N157, N158, N159, N160, N161, N162, N163, N164, N165,
N166, N167, N168, N169, N170, N171, N172, N173, N174, N175, N176,
N177, N178, N179, N180, N181, N182, N183, N184, N185, N186, N187,
N188, N189, N190, N191, N192, N193, N194, N195, N196, N213, N214,
N215, N216, N217, N218, N219, N220, N221, N222, N223, N224, N225,
N226, N227, N228, N245, N246, N279, N280, N281, N282, N283, N284,
N285, N286, N287, N288, N289, N290, N291, N292, N293, N294, N295,
N296, N297, N298, N299, N300, N301, N302, N303, N304, N305, N306,
N307, N308, N309, N310, n74, n75, n76, n77, n78, n79, n80, n81, n82,
n83, n84, n85, n86, n87, n88, n89, n90, n91, n92, n93, n94, n95, n96,
n97, n98, n99, n100, n101, n102, n103, n104, n105, n106, n107, n108,
n109, n110, n111, n112, n113, n114, n115, n116, n117, n118, n119,
n120, n121, n122, n123, n124, n125, n126, n127, n128, n129, n130,
n131, n132, n133, n134, n135, n136, n137, n138, n139, n140, n141,
n142, n143, n144, n145, n146, n147, n148, n149, n150, n151, n152,
n153, n154, n155, n156, n157, n158, n159, n160, n161, n162, n163,
n164, n165, n166, n167, n168, n169, n170, n171, n172, n173, n174,
```

NoMachine - VLSI_Design
ALU_PROJECT_2..CELLS..syn.v

```
ALU_File_Verilog.v ALU_PROJECT_2..CELLS..syn.v

// Generated by: Synopsys DC Expert(TM) in wire load mode
// Version : 0-2018.06-SP1
// Date   : Wed Sep 21 12:34:21 2022
///////////////////////////////////////////////////////////////////




nand2 \r79/ULTI2_9 ( .a(\r79/AEQB [9]), .b(\r79/LTV [9]), .out(
  \r79/LTV2 [9]) );
nand2 \r79/ULTI2_9 ( .a(\r79/LTV1 [9]), .b(\r79/LTV2 [9]), .out(
  \r79/LTV [10]) );
xor2 \r79/UEQI_10 ( .a(X[10]), .b(Y[10]), .out(n2476) );
nand2 \r79/UGTI0_10 ( .a(n2475), .b(X[10]), .out(\r79/GTV1 [10]) );
nand2 \r79/UGTI1_10 ( .a(\r79/AEQB [10]), .b(\r79/GTV [10]), .out(
  \r79/GTV2 [10]) );
nand2 \r79/UGTI2_10 ( .a(\r79/GTV1 [10]), .b(\r79/GTV2 [10]), .out(
  \r79/GTV [11]) );
nand2 \r79/UGTI2_10 ( .a(n2474), .b(Y[10]), .out(\r79/LTV1 [10]) );
nand2 \r79/ULTI1_10 ( .a(\r79/AEQB [10]), .b(\r79/LTV [10]), .out(
  \r79/LTV2 [10]) );
nand2 \r79/ULTI2_10 ( .a(\r79/LTV1 [10]), .b(\r79/LTV2 [10]), .out(
  \r79/LTV [11]) );
xor2 \r79/UEQI_11 ( .a(X[11]), .b(Y[11]), .out(n2473) );
nand2 \r79/UGTI0_11 ( .a(n2472), .b(X[11]), .out(\r79/GTV1 [11]) );
nand2 \r79/UGTI1_11 ( .a(\r79/AEQB [11]), .b(\r79/GTV [11]), .out(
  \r79/GTV2 [11]) );
nand2 \r79/UGTI2_11 ( .a(\r79/GTV1 [11]), .b(\r79/GTV2 [11]), .out(
  \r79/GTV [12]) );
nand2 \r79/ULTI1_11 ( .a(n2471), .b(Y[11]), .out(\r79/LTV1 [11]) );
nand2 \r79/UGTI1_11 ( .a(\r79/AEQB [11]), .b(\r79/LTV [11]), .out(
  \r79/LTV2 [11]) );
nand2 \r79/UGTI2_11 ( .a(\r79/LTV1 [11]), .b(\r79/LTV2 [11]), .out(
  \r79/LTV [12]) );
xor2 \r79/UEQI_12 ( .a(X[12]), .b(Y[12]), .out(n2470) );
nand2 \r79/UGTI0_12 ( .a(n2469), .b(X[12]), .out(\r79/GTV1 [12]) );
nand2 \r79/UGTI1_12 ( .a(\r79/AEQB [12]), .b(\r79/GTV [12]), .out(
  \r79/GTV2 [12]) );
nand2 \r79/UGTI2_12 ( .a(\r79/GTV1 [12]), .b(\r79/GTV2 [12]), .out(
  \r79/GTV [13]) );
nand2 \r79/ULTI0_12 ( .a(n2468), .b(Y[12]), .out(\r79/LTV1 [12]) );
nand2 \r79/UGTI1_12 ( .a(\r79/AEQB [12]), .b(\r79/LTV [12]), .out(
  \r79/LTV2 [12]) );
nand2 \r79/UGTI2_12 ( .a(\r79/LTV1 [12]), .b(\r79/LTV2 [12]), .out(
  \r79/LTV [13]) );
xor2 \r79/UEQI_13 ( .a(X[13]), .b(Y[13]), .out(n2467) );
nand2 \r79/UGTI0_13 ( .a(n2466), .b(X[13]), .out(\r79/GTV1 [13]) );
```

NoMachine - VLSI_Design

```
\mult 25/FS 1/G[1][0][3] , \mult 25/FS 1/G[1][1][0] ,
\mult 25/FS 1/G[1][1][1] , \mult 25/FS 1/G[1][1][2] ,
\mult 25/FS 1/G[2][0][0] , \mult 25/FS 1/TEMP_G[0][4][1]
\mult 25/FS 1/TEMP_G[0][4][2] , \mult 25/FS 1/TEMP_G[0][5][1] ,
\mult 25/FS 1/TEMP_G[0][5][2] , \mult 25/FS 1/TEMP_G[0][6][1] ,
\mult 25/FS 1/TEMP_G[0][6][2] , \mult 25/FS 1/G_n_int[0][3][3] ,
\mult 25/FS 1/G_n_int[0][4][0] , \mult 25/FS 1/G_n_int[0][4][1] ,
\mult 25/FS 1/G_n_int[0][4][2] , \mult 25/FS 1/G_n_int[0][4][3] ,
\mult 25/FS 1/G_n_int[0][5][0] , \mult 25/FS 1/G_n_int[0][5][1] ,
\mult 25/FS 1/G_n_int[0][5][2] , \mult 25/FS 1/G_n_int[0][5][3] ,
\mult 25/FS 1/G_n_int[0][6][0] , \mult 25/FS 1/G_n_int[0][6][1] ,
\mult 25/FS 1/G_n_int[0][6][2] , \mult 25/FS 1/G_n_int[0][6][3] ,
\mult 25/FS 1/G_n_int[0][7][0] , \mult 25/FS 1/Pg_int[0][4][0] ,
\mult 25/FS 1/Pg_int[0][4][1] , \mult 25/FS 1/Pg_int[0][4][2]
\mult 25/FS 1/Pg_int[0][4][3] , \mult 25/FS 1/Pg_int[0][5][0] ,
\mult 25/FS 1/Pg_int[0][5][1] , \mult 25/FS 1/Pg_int[0][5][2] ,
\mult 25/FS 1/Pg_int[0][5][3] , \mult 25/FS 1/Pg_int[0][6][0] ,
\mult 25/FS 1/Pg_int[0][6][1] , \mult 25/FS 1/Pg_int[0][6][2] ,
\mult 25/FS 1/Pg_int[0][6][3] , \mult 25/FS 1/Pg_int[0][7][0] ,
\mult 25/A2[15] , \mult 25/A2[16] , \mult 25/A2[17] , \mult 25/A2[18] ,
\mult 25/A2[19] , \mult 25/A2[20] , \mult 25/A2[21] , \mult 25/A2[22] ,
\mult 25/A2[23] , \mult 25/A2[24] , \mult 25/A2[25] ,
\mult 25/A2[26] , \mult 25/A2[27] , \mult 25/A2[28] ,
\mult 25/A2[29] , \mult 25/A1[0] , \mult 25/A1[1] , \mult 25/A1[2] ,
\mult 25/A1[3] , \mult 25/A1[4] , \mult 25/A1[5] , \mult 25/A1[6] ,
\mult 25/A1[7] , \mult 25/A1[8] , \mult 25/A1[9] , \mult 25/A1[10] ,
\mult 25/A1[11] , \mult 25/A1[12] , \mult 25/A1[13] ,
\mult 25/A1[14] , \mult 25/A1[15] , \mult 25/A1[16] ,
\mult 25/A1[17] , \mult 25/A1[18] , \mult 25/A1[19] ,
\mult 25/A1[20] , \mult 25/A1[21] , \mult 25/A1[22] ,
\mult 25/A1[23] , \mult 25/A1[24] , \mult 25/A1[25] ,
\mult 25/A1[26] , \mult 25/A1[27] , \mult 25/A1[28] ,
\mult 25/ab[0][1] , \mult 25/ab[0][2] , \mult 25/ab[0][3] ,
\mult 25/ab[0][4] , \mult 25/ab[0][5] , \mult 25/ab[0][6] ,
\mult 25/ab[0][7] , \mult 25/ab[0][8] , \mult 25/ab[0][9] ,
\mult 25/ab[0][10] , \mult 25/ab[0][11] , \mult 25/ab[0][12] ,
\mult 25/ab[0][13] , \mult 25/ab[0][14] , \mult 25/ab[0][15] ,
\mult 25/ab[1][0] , \mult 25/ab[1][1] , \mult 25/ab[1][2] ,
\mult 25/ab[1][3] , \mult 25/ab[1][4] , \mult 25/ab[1][5] ,
\mult 25/ab[1][6] , \mult 25/ab[1][7] , \mult 25/ab[1][8] ,
\mult 25/ab[1][9] , \mult 25/ab[1][10] , \mult 25/ab[1][11]
```

Behavioural Code Output:

Behavioural code result output for all operations of the ALU

S.NO	FUNCTION	INPUT - X	INPUT - Y	OUTPUT - Z
1.	Reset	0000	0000	00000000
2.	Addition	28	14	0000003c
3.	Subtraction	28	14	00000014
4.	Multiplication	28	14	00000320
5.	Division	28	14	00000002
6.	Logical shift left	28	14	00000050
7.	Logical shift right	28	14	00000014
8.	Rotate left	28	14	00000050
9.	Rotate right	28	14	00000014
10.	AND	28	14	00000000
11.	OR	28	14	0000003c
12.	NOR	28	14	fffffc3
13.	NAND	28	14	fffffff
14.	XOR	28	14	0000003c
15.	XNOR	28	14	fffffc3
16.	Less than	28	14	0000

Comparison of Mapped code and Behavioural code Waveform:

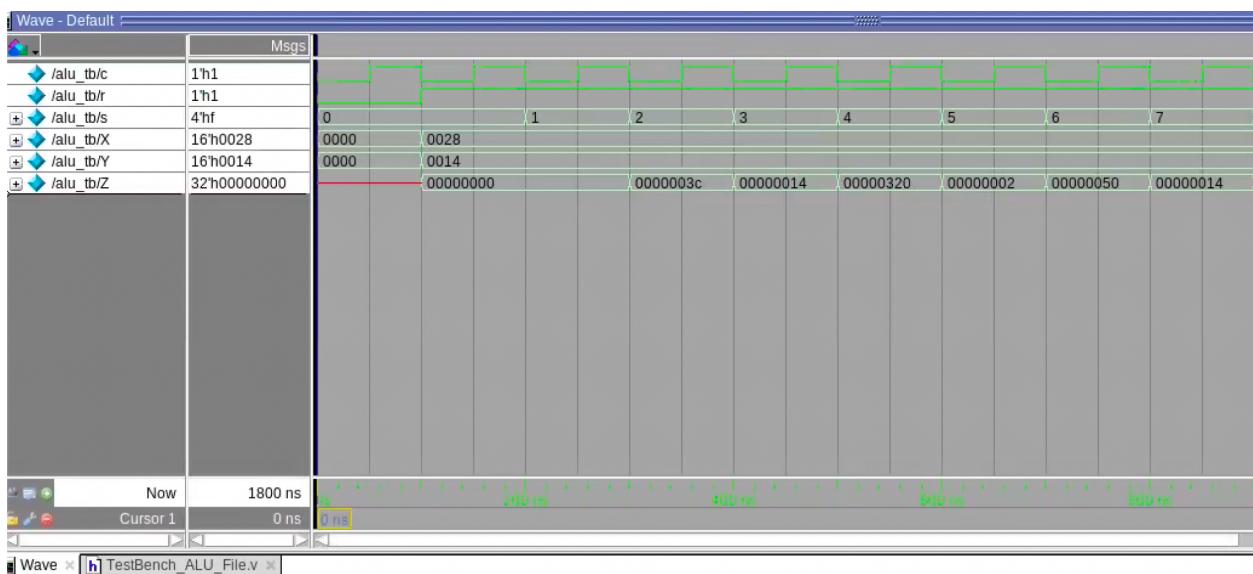
The following images consist of the resultant Mapped code waveforms and Behavioural code waveforms that have been simulated in ModelSim. The waveform shows the outputs for 16 different ALU operations. The first two set of waveforms consists of 7 ALU operations and the second set of waveforms consists of 9 ALU operations.

Waveform - Output for the following operations:

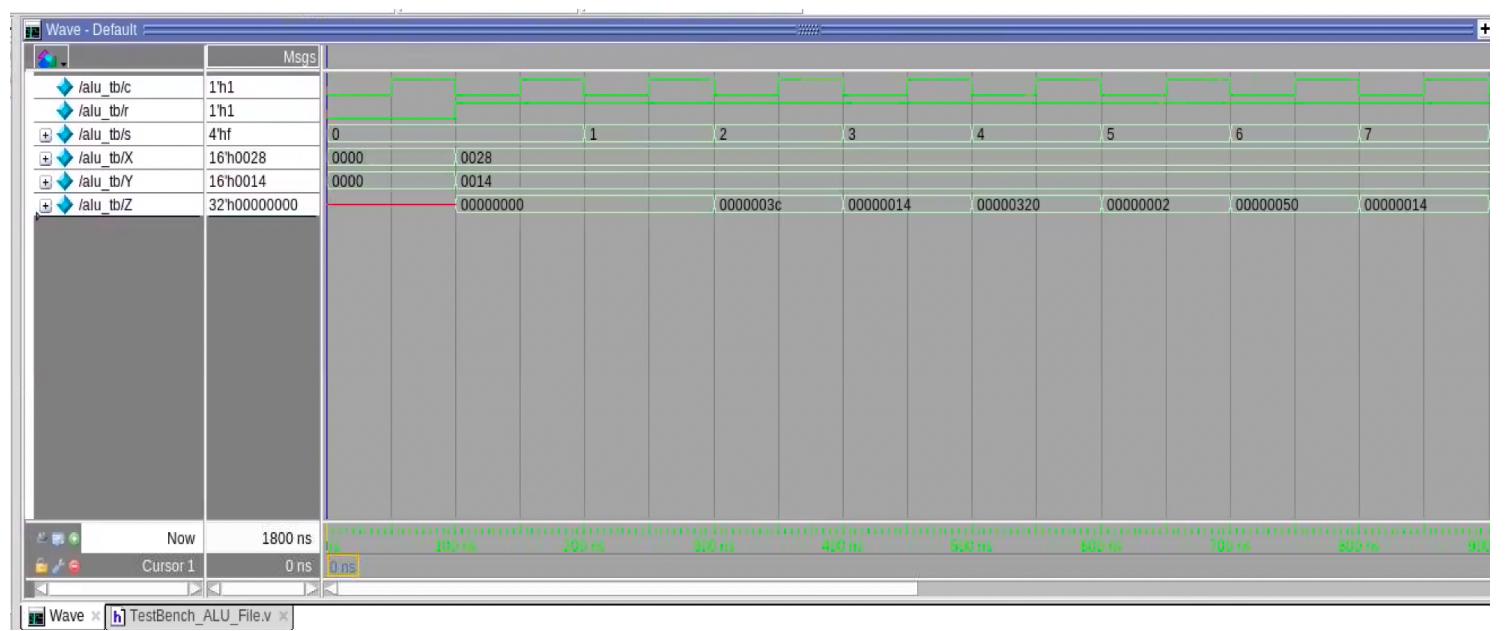
- **Operations:**

1. ALU - RESET
2. ALU - ADDITION OPERATION
3. ALU - SUBTRACTION OPERATION
4. ALU - MULTIPLICATION OPERATION
5. ALU - DIVISION OPERATION
6. ALU - Logical Shift Left OPERATION
7. ALU - Logical Shift Right OPERATION

Mapped Code Waveform:



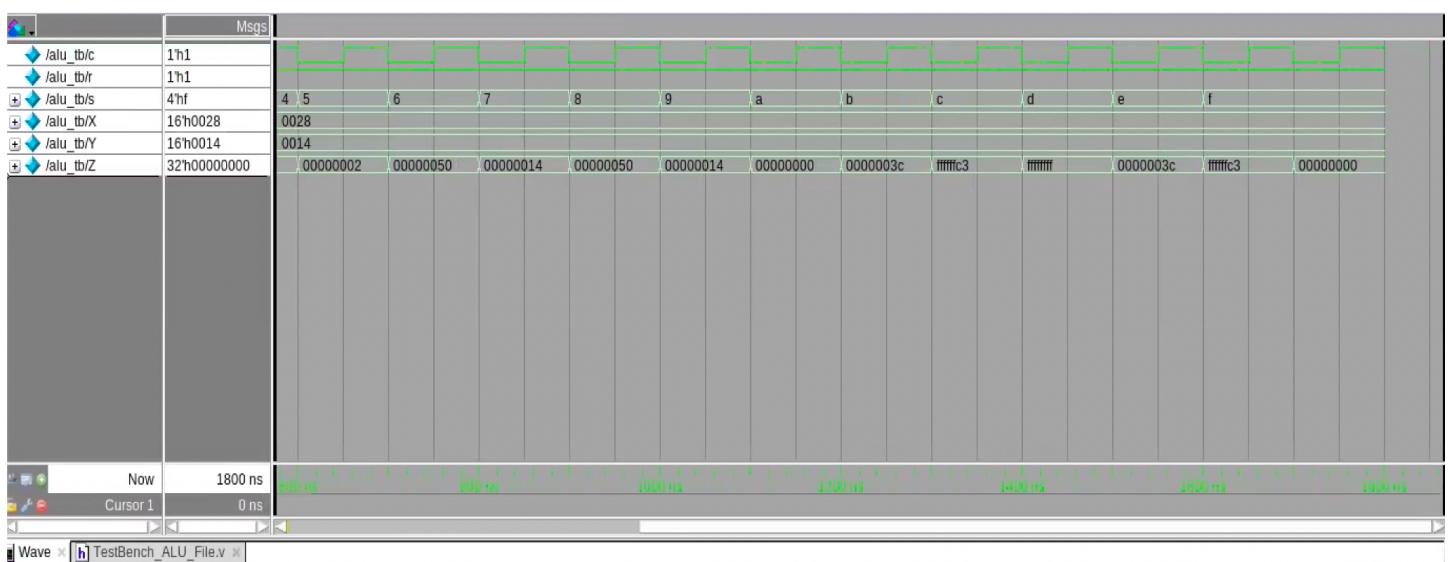
Behavioral Code Waveform



- **Operations:**

8. ALU - Rotate left OPERATION
9. ALU - Rotate right OPERATION
10. ALU - AND OPERATION
11. ALU - OR OPERATION
12. ALU - NOR OPERATION
13. ALU - NAND OPERATION
14. ALU - XOR OPERATION
15. ALU - XNOR OPERATION
16. ALU - Less than OPERATION

Mapped Code Waveform:



Behavioral Code Waveform:

