Training Project

32 bit ALU using Verilog Code

Name :- Sarthak Sharma

College :- Chaudhary Charan Singh University Meerut

Title: Design and Implementation of a 32-bit ALU using Verilog HDL

Abstract: This report presents the design and implementation of a 32-bit ALU using Verilog HDL. The ALU is a fundamental component in digital circuits that performs arithmetic and logical operations on binary numbers. The objective of this project is to design a versatile ALU capable of performing a wide range of operations efficiently and accurately.

Introduction:

The Arithmetic Logic Unit (ALU) is a crucial component in digital systems, responsible for executing arithmetic and logical operations. The ALU takes in two 32-bit binary numbers as inputs and performs operations such as addition, subtraction, bitwise logical operations (AND, OR, XOR), and comparison operations. The output of the ALU is the result of the specified operation.

Design Methodology:

The ALU performs the different operations including adders, subtractors, bitwise logical operators, and comparators. The main ALU module takes control signals as inputs to determine the operation to be performed and selects the appropriate sub-module accordingly.

The adder module is responsible for performing addition operations. It takes two 32-bit inputs, performs binary addition bit by bit, and generates a 32-bit sum output along with a carry output. The subtractor module is similar to the adder but performs subtraction using two's complement representation.

The bitwise logical operator modules, such as AND, OR, and XOR, take two 32-bit inputs and perform the corresponding logical operation on each bit to generate a 32-bit output.

The comparator module compares the two 32-bit inputs and produces control signals such as equal, greater than, less than, etc., which are used for decision making within the ALU.

Implementation:

The design is implemented in Verilog HDL using behavioral modeling. ports. The main ALU module instantiates the required sub-modules and connects them based on the control signals.

The Verilog code includes appropriate data types and signal declarations, combinational logic for each operation, and sequential logic for carry propagation and comparison operations. The code is synthesized using a suitable synthesis tool to generate the corresponding gate-level representation.

Verilog code:

```
module alu(result,a,b,opcode,en);

output [31:0]result;

input [2:0]opcode;

input [31:0]a,b;

input en;

reg[31:0]result;

always @(a,b,opcode,en)

begin

if(en==1)

begin

case(opcode)

3'b000 : result = a + b;

3'b001 : result = a - b;

3'b010 : result = a + 1;
```

3'b011 : result = a - 1;

3'b100 : result = a;

3'b101 : result = ~a;

3'b110 : result = a & b;

3'b111 : result = a | b;

default: result = 32'b0;

endcase

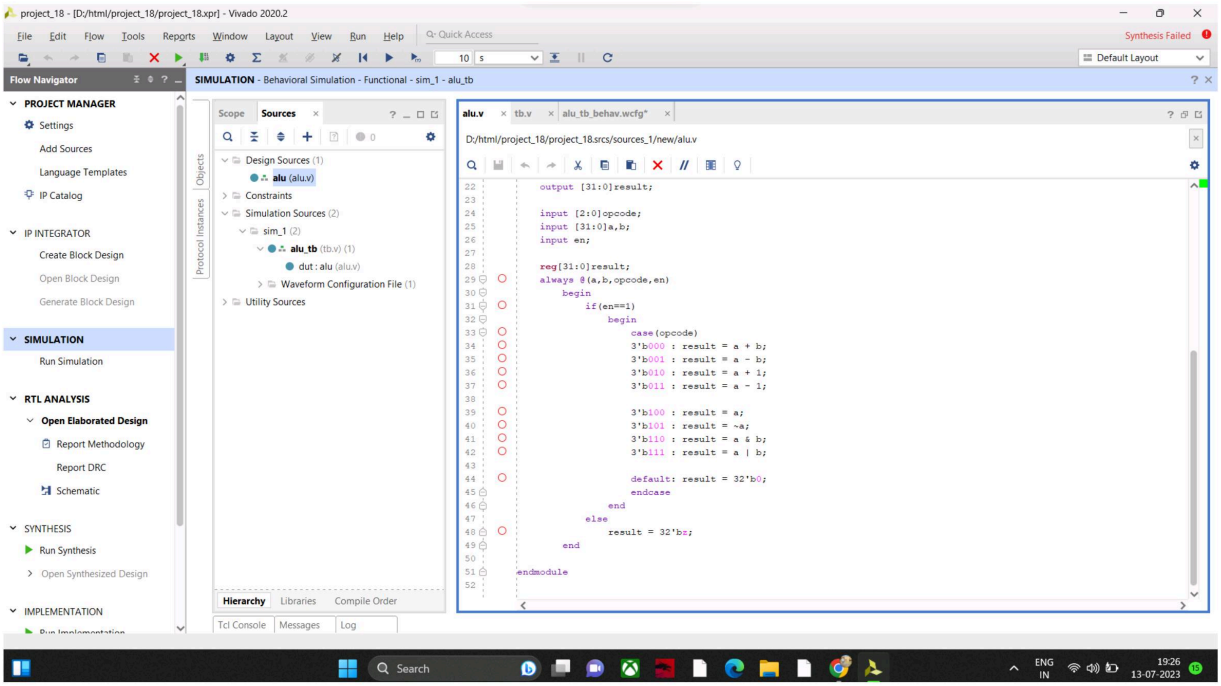
end

else

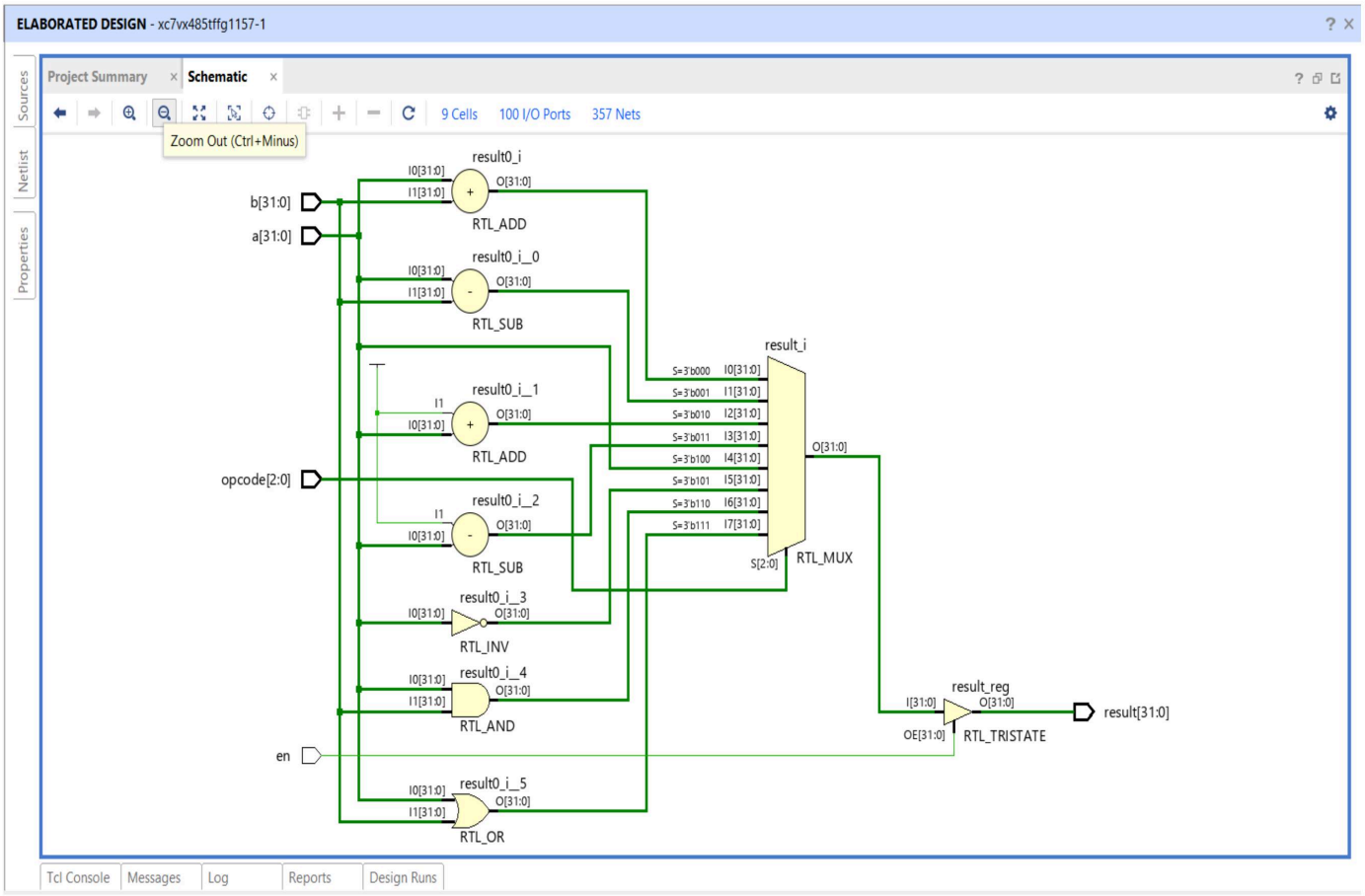
result = 32'bz;

end

endmodule



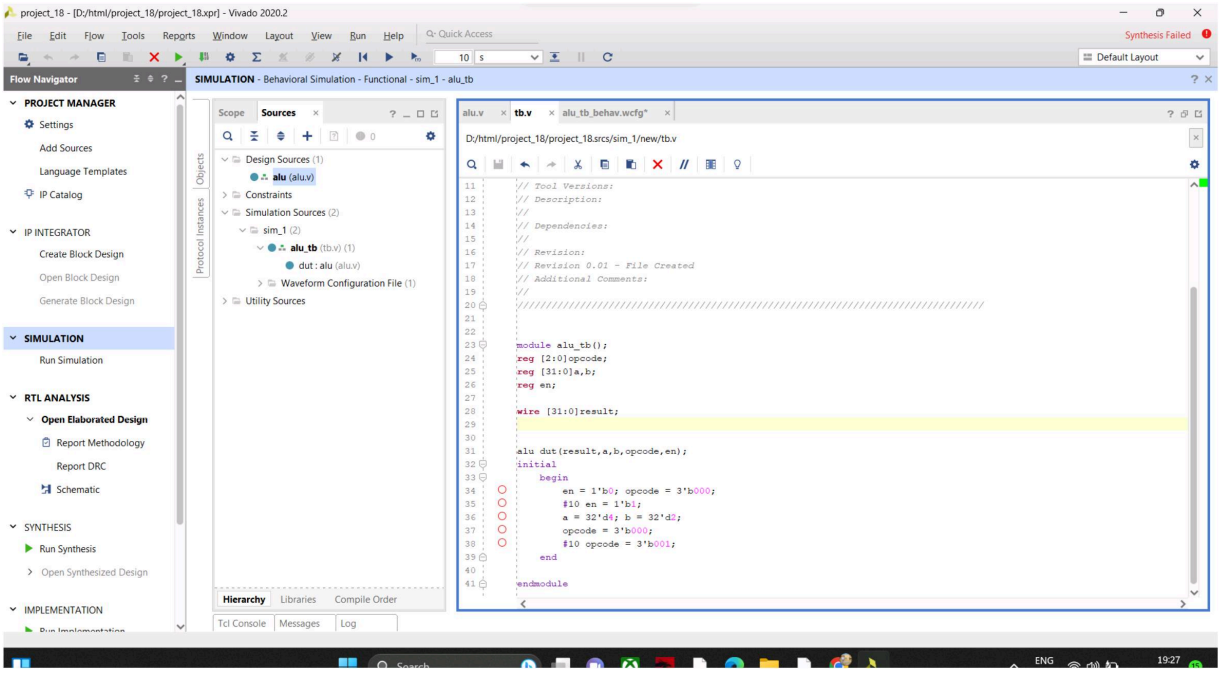
Schematic:


ELABORATED DESIGN - xc7vx485tffg1157-1
Project Summary
Schematic
Sources
Netlist
Properties
Zoom Out (Ctrl+Minus)
9 Cells
100 I/O Ports
357 Nets
b[31:0]
a[31:0]
opcode[2:0]
en
result0_i
RTL_ADD
result0_i__0
RTL_SUB
result0_i__1
RTL_ADD
result0_i__2
RTL_SUB
result0_i__3
RTL_INV
result0_i__4
RTL_AND
result0_i__5
RTL_OR
result_i
RTL_MUX
result_reg
RTL_TRISTATE
result[31:0]
Tcl Console
Messages
Log
Reports
Design Runs

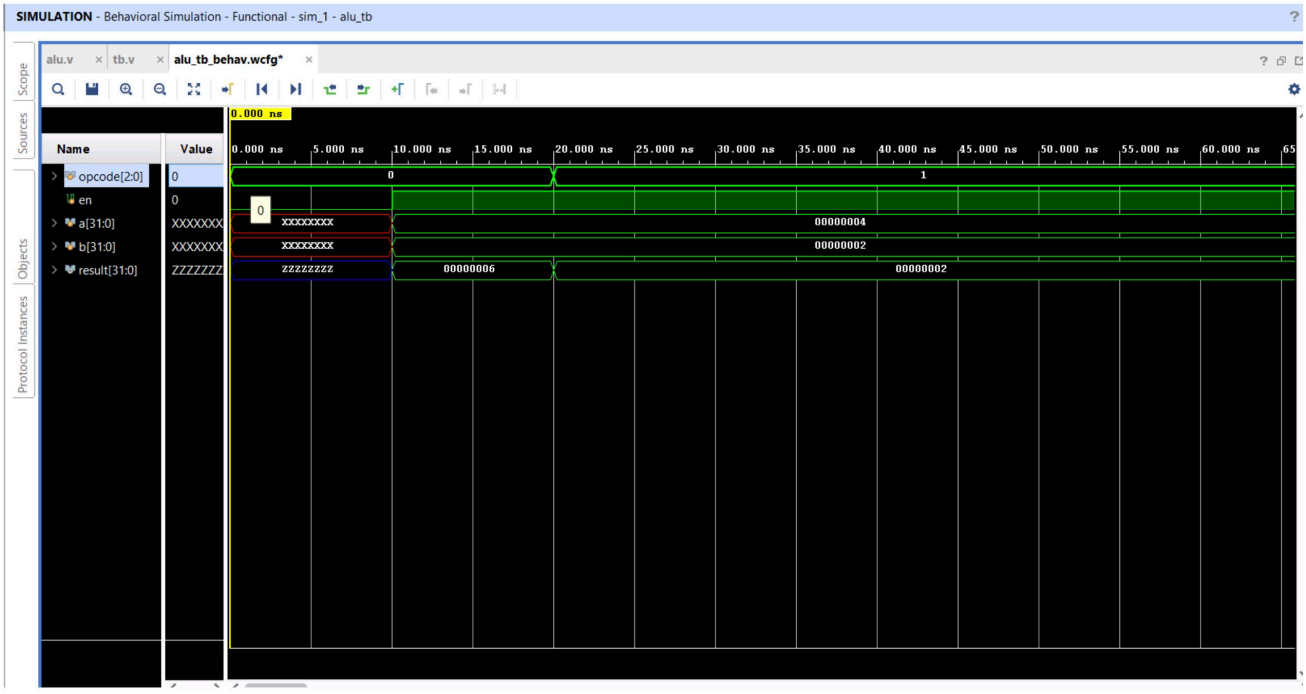
Testbench:

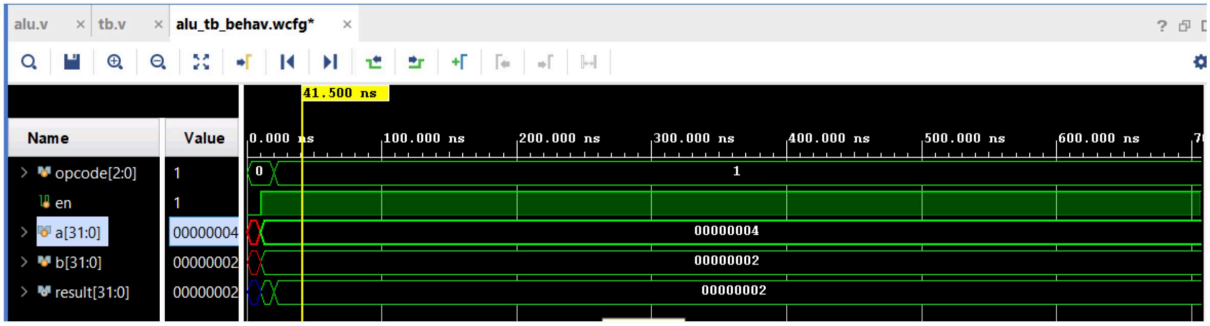
```
module alu_tb();

reg [2:0]opcode;

reg [31:0]a,b;

reg en;

wire [31:0]result;

alu uut(result,a,b,opcode,en);

initial

        begin

                en = 1'b0; opcode = 3'b000;

                #10 en = 1'b1;

                a = 32'd4; b = 32'd2;

                opcode = 3'b000;

                #10 opcode = 3'b001;

        end

Endmodule
```

In this testbench, we instantiate the `ALU 32 bit` module (`uut`) and connect its input and output ports to signals declared within the testbench.


project_18 - [D:/html/project_18/project_18.xpr] - Vivado 2020.2
File Edit Flow Tools Reports Window Layout View Run Help
Quick Access
Synthesis Failed
Default Layout
Flow Navigator
SIMULATION - Behavioral Simulation - Functional - sim_1 - alu_tb
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
Report Methodology
Report DRC
Schematic
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Scope
Sources
Design Sources (1)
alu (alu.v)
Constraints
Simulation Sources (2)
sim_1 (2)
alu_tb (tb.v) (1)
dut : alu (alu.v)
Waveform Configuration File (1)
Utility Sources
Hierarchy Libraries Compile Order
Tcl Console Messages Log
alu.v
tb.v
alu_tb_behav.wcfg*
D:/html/project_18/project_18.srcs/sim_1/new/tb.v
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
module alu_tb();
reg [2:0]opcode;
reg [31:0]a,b;
reg en;
wire [31:0]result;
alu dut(result,a,b,opcode,en);
initial
begin
en = 1'b0; opcode = 3'b000;
#10 en = 1'b1;
a = 32'd4; b = 32'd2;
opcode = 3'b000;
#10 opcode = 3'b001;
end
endmodule
ENG
19:27

Simulation:


SIMULATION - Behavioral Simulation - Functional - sim_1 - alu_tb
alu.v
tb.v
alu_tb_behav.wcfg*
Scope
Sources
Objects
Protocol Instances
0.000 ns
Name
Value
opcode[2:0]
en
a[31:0]
b[31:0]
result[31:0]
0
0
XXXXXXX
XXXXXXX
ZZZZZZZ
0.000 ns
5.000 ns
10.000 ns
15.000 ns
20.000 ns
25.000 ns
30.000 ns
35.000 ns
40.000 ns
45.000 ns
50.000 ns
55.000 ns
60.000 ns
00000004
00000002
00000006
00000002



Here, in this simulation waveform defined,

EN: Enable input

Enable input is low then the output result is not defined

When enable is high then the ALU starts working and performs different operations.

Conclusion:

In conclusion, the design and implementation of a 32-bit ALU using Verilog HDL have been successfully achieved. The ALU performs various arithmetic and logical operations efficiently and accurately. The modular design allows for flexibility and easy integration into larger digital systems.