Worksheet of Experiment

Date of Experiment	1 oct, 2024
Name	Akshada Jotiram Ghorpade.
Registration Number	12302956.

Title of Experiment: Develop RTL designs in verilog to implement 8 bit ALU and verify their functionality using cadence NCSIM.

Name of Language: Verilog.

Name of Software: Cadence NCSIM, simvision.

Theory of Experiment:

Arithmetic and Logical Unit:

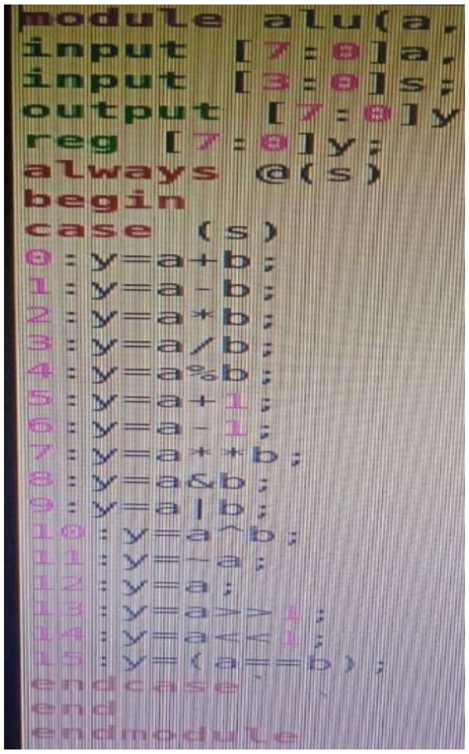
- An 8-bit Arithmetic Logic Unit (ALU) is a digital circuit that performs arithmetic and logical operations on 8-bit binary numbers.
- It is a core component in many microprocessors and microcontroller.
- It is a major component of the CPU in a computer system.
- An integer unit (IU) is just an integrated circuit within a GPU or GPU that performs the last calculations in the processor.
- It can execute all arithmetic and logic operations, including Boolean comparisons, such as subtraction, addition, and shifting (XOR, OR, AND, and NOT operations).
- Binary numbers can also perform bitwise and mathematical operations.
- AU (arithmetic unit) and LU (logic unit) are two types of arithmetic logic units.
- The ALU’s operands and code instruct it on which operations to perform based on the incoming data.
- When the ALU has finished processing the data, it sends the result to the computer memory.

▪ The following are some of ALU’s drawbacks:

- Floating variables have higher delays with the ALU, and the intended controller is difficult to grasp.
- If memory space were fixed, bugs would appear in the results.
- Amateurs are tough to understand since their circuits are complex, and the principle of pipelining is also difficult to grasp.
- The inconsistencies in latencies are a known drawback of ALU.

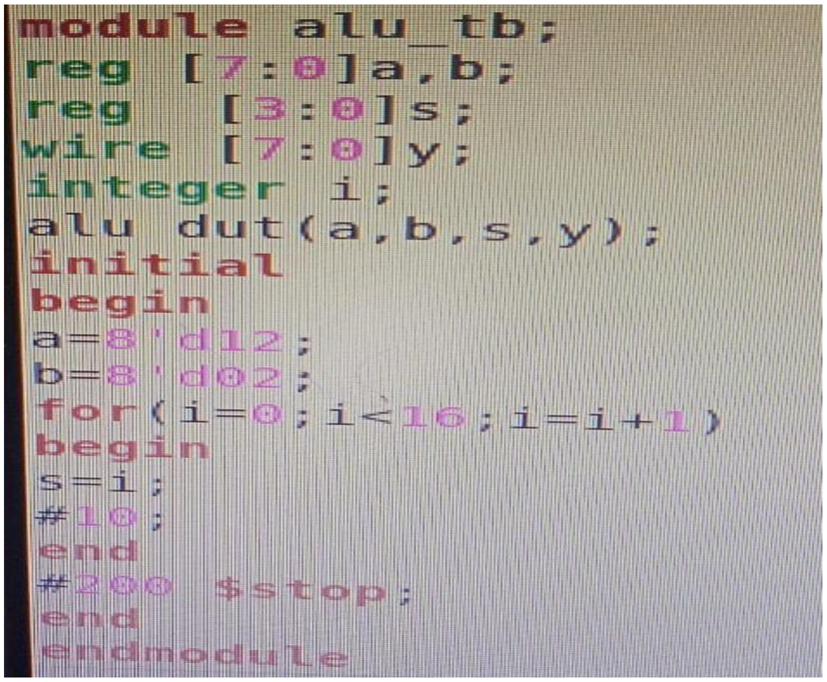
Program and Stimulus (Testbench) program:

PROGRAM:



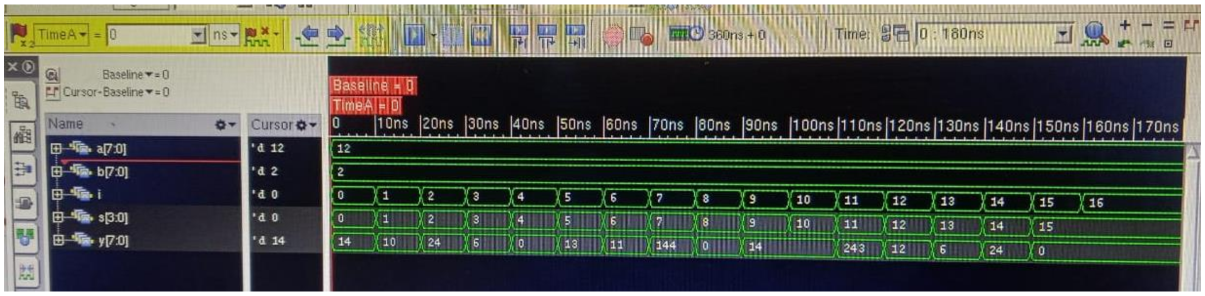
```
module alu(a,
input [7:0]a,
input [3:0]s;
output [7:0]y
reg [7:0]y;
always @(s)
begin
case (s)
0:y=a+b;
1:y=a-b;
2:y=a*b;
3:y=a/b;
4:y=a%b;
5:y=a+1;
6:y=a-1;
7:y=a**b;
8:y=a&b;
9:y=a|b;
10:y=a^b;
11:y=~a;
12:y=a;
13:y=a>>1;
14:y=a<<1;
15:y=(a==b);
endcase
end
endmodule
```

Testbench:

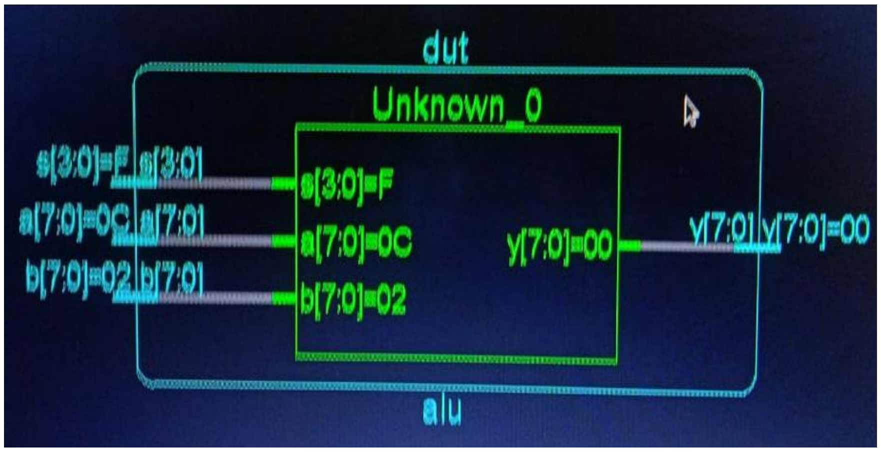


```
module alu_tb;
reg [7:0]a,b;
reg  [3:0]s;
wire [7:0]y;
integer i;
alu dut(a,b,s,y);
initial
begin
a=8'd12;
b=8'd02;
for(i=0;i<16;i=i+1)
begin
s=i;
#10;
end
#200 $stop;
end
endmodule
```

Simulation waveform:



RTL Schematic:



Learning Outcome:

1. Learn to write syntactically correct Verilog code for ALU.

2. Create testbenches to simulate and verify the functionality of ALU.

3. Apply techniques to optimize logic circuits for speed, area, and power efficiency.

4. Perform timing analysis to ensure circuits meet required timing constraints.

5. Analyze the performance of ALU and circuits in terms of timing, power, and area using simulation results.