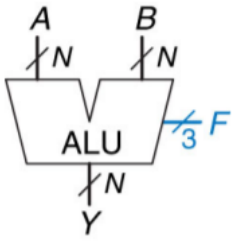
ECE 469 HDL
Project 1, Part 1
Tamanna Ravi Rupani
UIN: 665679988
32-bit Arithmetic Logic Unit (ALU)

Following is the design for an n-bit ALU with two inputs and 1 output. And given is the table for ALU operations based on the F input.



$F_{2:0}$	Function
000	A AND B
001	A OR B
010	A + B
011	not used
100	A AND $\overline{B}$
101	A OR $\overline{B}$
110	A − B
111	SLT

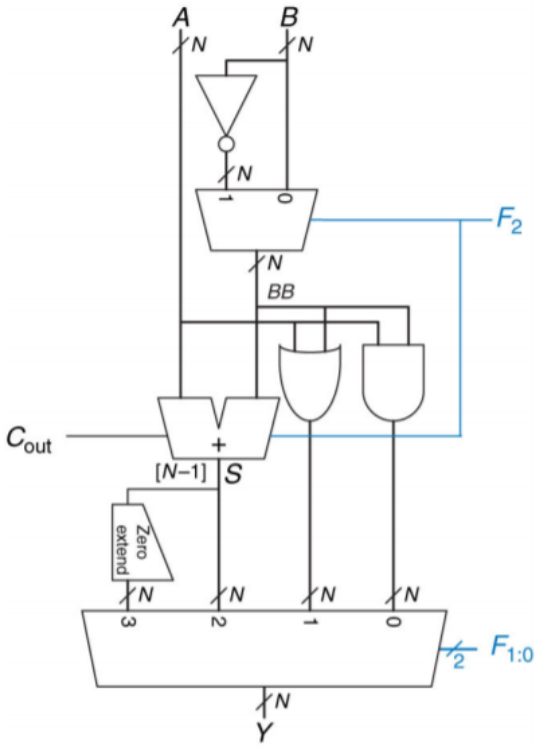


Table of test vectors

Test	F [2:0]	A	B	Y	Zero
ADD 0+0	2	00000000	00000000	00000000	1
ADD 0+(-1)	2	00000000	FFFFFFFF	FFFFFFFF	0
ADD 1+(-1)	2	00000001	FFFFFFFF	00000000	1
ADD FF+1	2	000000FF	00000001	00000100	0
SUB 0-0	6	00000000	00000000	00000000	1
SUB 0-(-1)	6	00000000	FFFFFFFF	00000001	0
SUB 1-1	6	00000001	00000001	00000000	1
SUB 100-10	6	00000100	00000010	000000F0	0
SLT 0,0	7	00000000	00000000	00000000	1
SLT 0,1	7	00000000	00000001	00000001	0
SLT 0,-1	7	00000000	FFFFFFFF	00000000	1
SLT 1,0	7	00000001	00000000	00000000	1
SLT -1,0	7	FFFFFFFF	00000000	00000001	0
AND FFFFFFFF, FFFFFFFF	0	FFFFFFFF	FFFFFFFF	FFFFFFFF	0
AND FFFFFFFF, 12345678	0	FFFFFFFF	12345678	12345678	0
AND 12345678, 87654321	0	12345678	87654321	02244220	0
AND 00000000, FFFFFFFF	0	00000000	FFFFFFFF	00000000	1
OR FFFFFFFF, FFFFFFFF	1	FFFFFFFF	FFFFFFFF	FFFFFFFF	0
OR 12345678, 87654321	1	12345678	87654321	97755779	0
OR 00000000, FFFFFFFF	1	00000000	FFFFFFFF	FFFFFFFF	0
OR 00000000, 00000000	1	00000000	00000000	00000000	1
AND FFFFFFFF,00000000	4	FFFFFFFF	00000000	FFFFFFFF	0
AND FFFFFFFF,FFFFFFFF	4	FFFFFFFF	FFFFFFFF	00000000	1
OR FFFFFFFF,FFFFFFFF	5	FFFFFFFF	FFFFFFFF	FFFFFFFF	0
OR 00000000,00000000	5	00000000	00000000	FFFFFFFF	0

ALU Code

```
module alu(input logic [31:0]A,
        input logic [31:0]B,
        input logic [2:0]F,
        output logic [31:0]Y,
        output logic zero);
logic [31:0]Sum;
logic [31:0]BB_Mux;

always_comb
```

```
case(F[2])
1'b0: BB_Mux = B;
1'b1: BB_Mux = ~B;
endcase
assign Sum = A + BB_Mux + F[2];
always_comb
case (F[1:0])
2'b00: Y = A & BB_Mux;
2'b01: Y = A | BB_Mux;
2'b10: Y = Sum;
2'b11: Y = {31'b0000000000000000000000000000000, Sum[31]};
endcase
assign zero = (Y==32'b0);
endmodule
```

ALU Test Vector File

```
010_00000000000000000000000000000000_00000000000000000000000000000000_00000000000000000000000000000000_1
010_00000000000000000000000000000000_11111111111111111111111111111111_11111111111111111111111111111111_0
010_00000000000000000000000000000001_11111111111111111111111111111111_00000000000000000000000000000000_1
010_00000000000000000000000011111111_00000000000000000000000000000001_00000000000000000100000000_0
110_00000000000000000000000000000000_00000000000000000000000000000000_00000000000000000000000000000000_1
110_00000000000000000000000000000000_11111111111111111111111111111111_00000000000000000000000000000001_0
110_00000000000000000000000000000001_00000000000000000000000000000001_00000000000000000000000000000000_1
110_00000000000000000000000100000000_00000000000000000000000000010000_00000000000000011110000_0
111_00000000000000000000000000000000_00000000000000000000000000000000_00000000000000000000000000000000_1
111_00000000000000000000000000000000_00000000000000000000000000000001_00000000000000000000000000000001_0
111_00000000000000000000000000000000_11111111111111111111111111111111_00000000000000000000000000000000_1
111_00000000000000000000000000000001_00000000000000000000000000000000_00000000000000000000000000000000_1
```

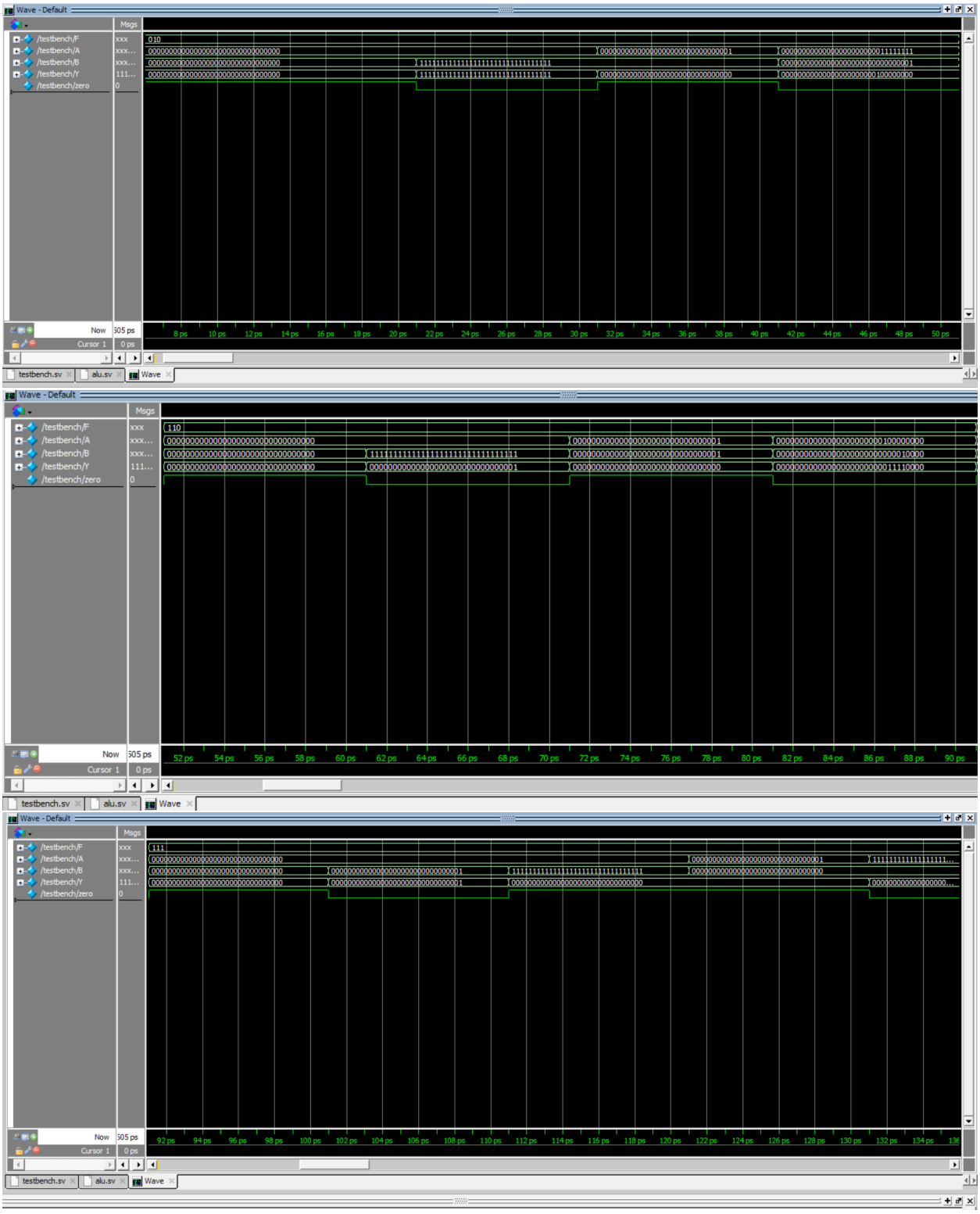
```
111_11111111111111111111111111111111_00000000000000000000000000000000_00000000
00000000000000000000001_0
000_11111111111111111111111111111111_11111111111111111111111111111111_11111111
11111111111111111111111_0
000_11111111111111111111111111111111_00010010001101000101011001111000_00010010
00110100010101100111000_0
000_00010010001101000101011001111000_10000111011001010100001100100001_00000010
00100100010000100010000_0
000_00000000000000000000000000000000_11111111111111111111111111111111_00000000
00000000000000000000000_1
001_11111111111111111111111111111111_11111111111111111111111111111111_11111111
11111111111111111111111_0
001_00010010001101000101011001111000_10000111011001010100001100100001_10011001
10011001100110011001001_0
001_00000000000000000000000000000000_11111111111111111111111111111111_11111111
11111111111111111111111_0
001_00000000000000000000000000000000_00000000000000000000000000000000_00000000
00000000000000000000000_1
100_11111111111111111111111111111111_00000000000000000000000000000000_11111111
11111111111111111111111_0
100_11111111111111111111111111111111_11111111111111111111111111111111_00000000
00000000000000000000000_1
101_11111111111111111111111111111111_11111111111111111111111111111111_11111111
11111111111111111111111_0
101_00000000000000000000000000000000_00000000000000000000000000000000_11111111
11111111111111111111111_0
```

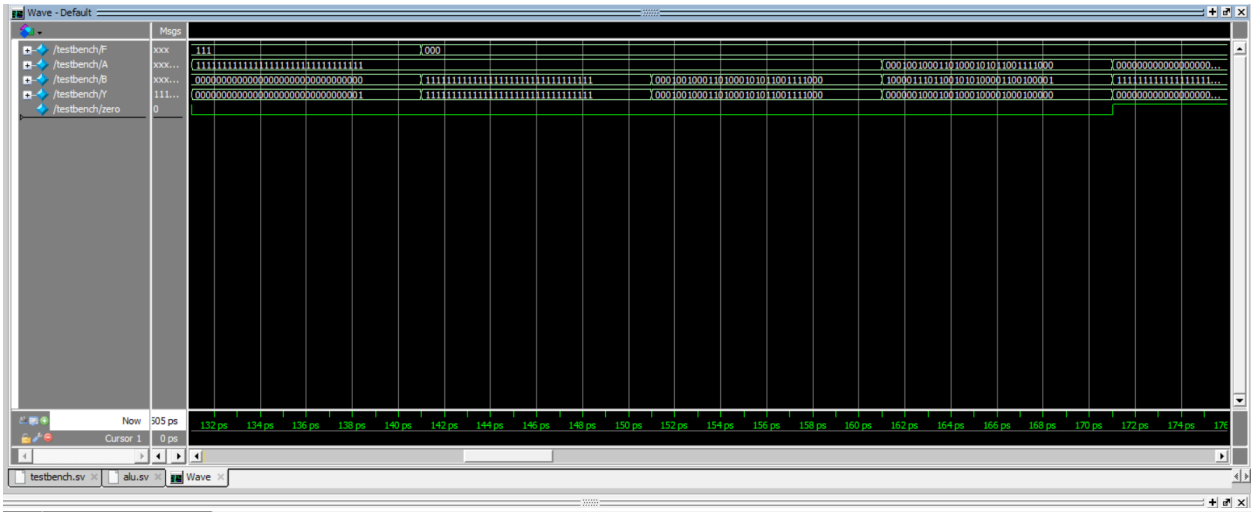
ALU Testbench code

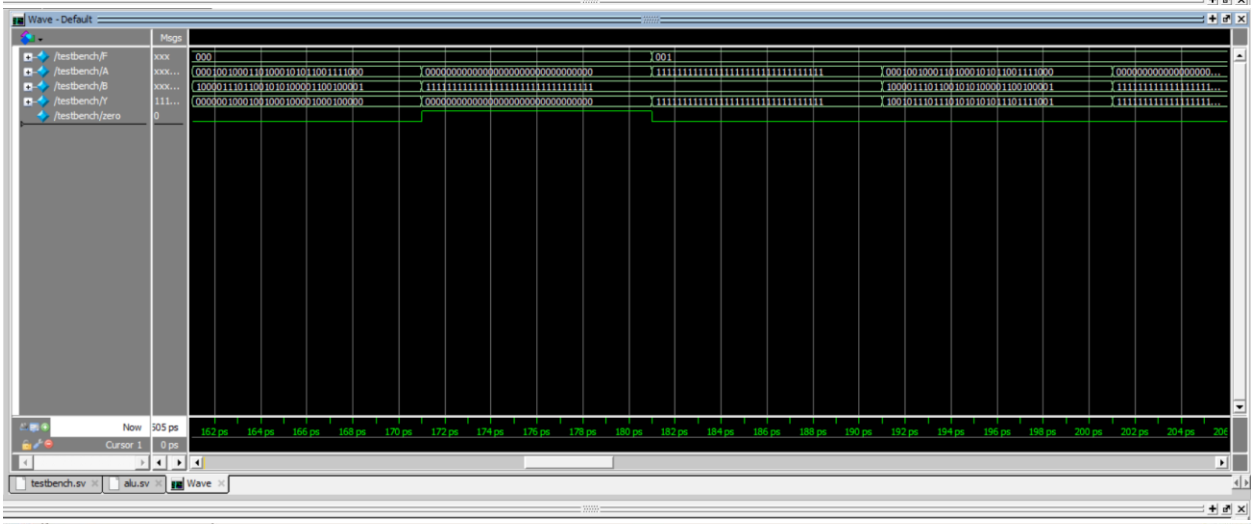
```
module testbench();
logic clk, reset; // Clock and Reset are internal
logic [31:0]A; // Input 1
logic [31:0]B; // Input 2
logic [2:0]F; // Control element
logic [31:0]yexpected; // Expected value of the output
logic zeroexp; // Expected zero value
logic zero; // Values from testvectors
logic [31:0]Y; // Output of circuit
reg [31:0]vectornum, errors; // Bookkeeping variables
reg [99:0]testvectors[0:10000];// Array of testvectors
alu dut(.F(F), .A(A), .B(B), .Y(Y), .zero(zero));  // Instantiate device under test
// Generate clock
always // No sensitivity list, so it always executes
```

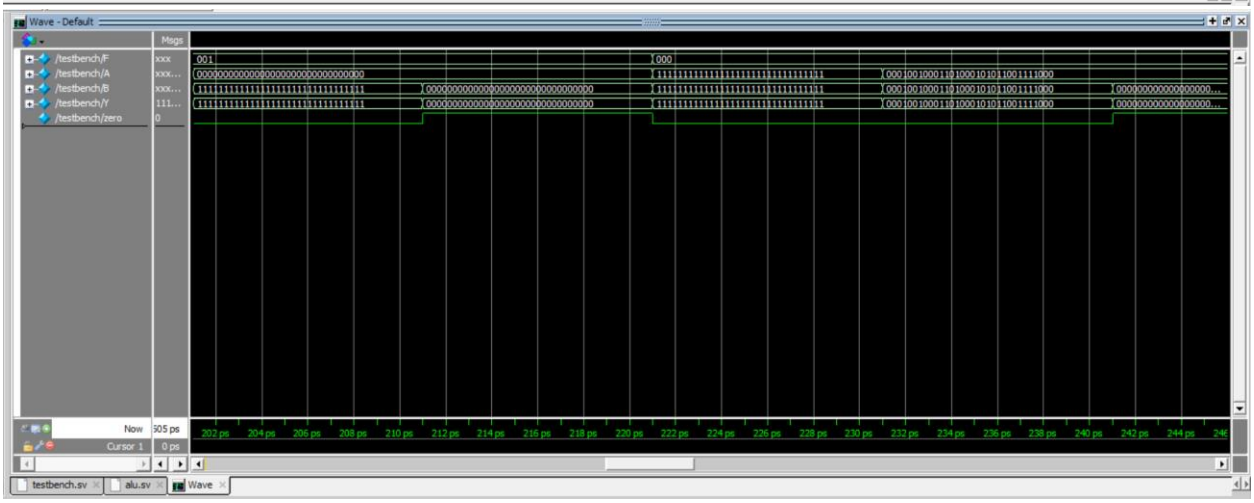
```
begin
clk = 1; #5; clk = 0; #5; // 10ns period
end
// At start of test, load vectors
// And pulse reset
initial // Will execute at the beginning once
begin
$readmemb("D:/MS Sem 2/ECE 469 HDL/Project 1/Part 1/alutv.txt", testvectors); // Read vectors
vectornum = 0; errors = 0; // Initialize
reset = 1; #10; reset = 0; // Apply reset wait
end
// Apply test vectors on rising edge of clk
always @(posedge clk)
begin
#1; {F, A, B, yexpected, zeroexp} = testvectors[vectornum];
end
// Check results on falling edge of clk
always @(negedge clk)
if (~reset) // Skip during reset
begin
if (Y !== yexpected)
begin
$display("Error: inputs = %b", {F, A, B});
$display(" outputs = %b (%b expected)",Y,yexpected);
errors = errors + 1;
end
// Increment array index and read next testvector
vectornum = vectornum + 1;
if ( vectornum == 50)
begin
$display("%d tests completed with %d errors",
vectornum, errors);
$finish; // End simulation
end
end
endmodule
```

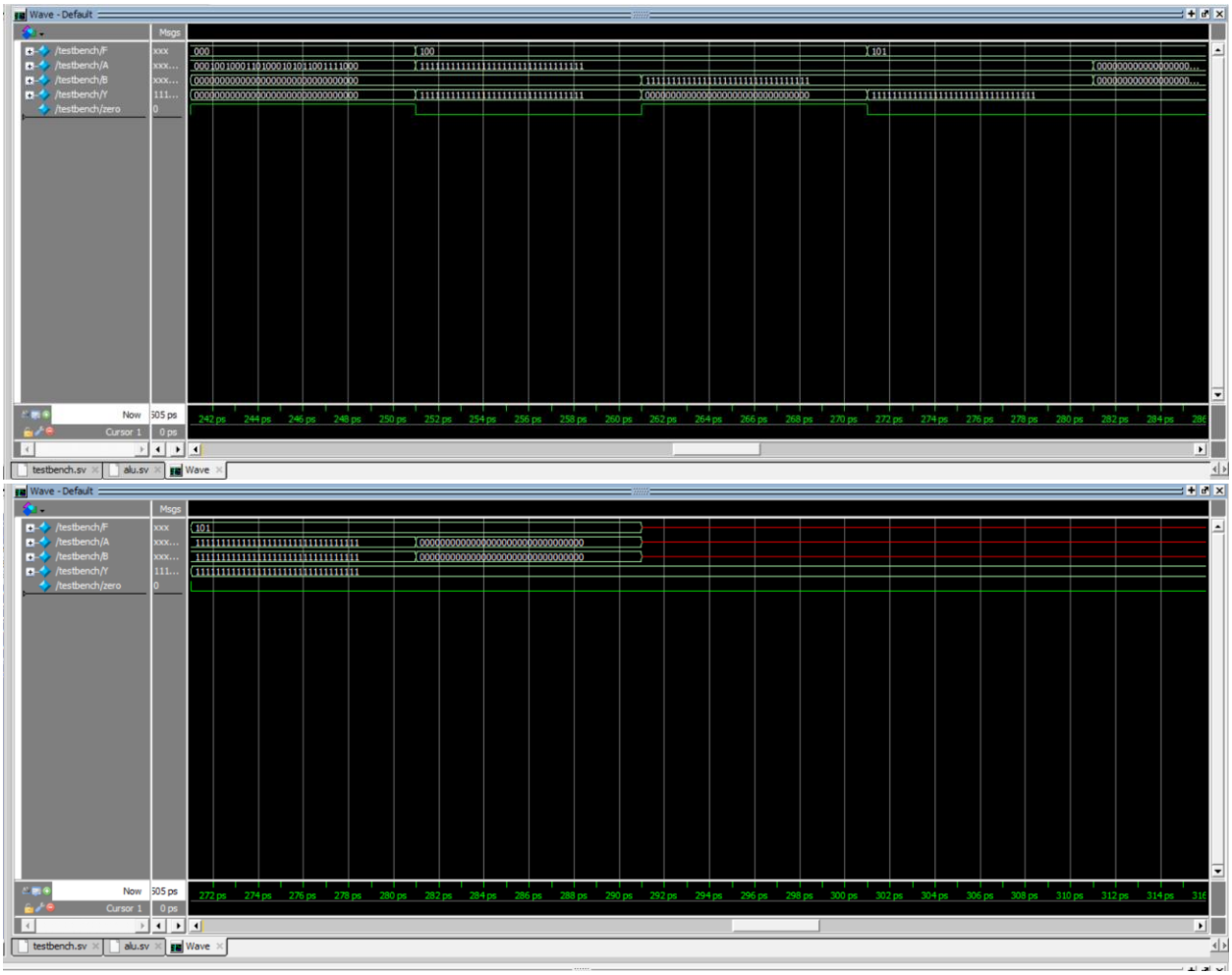
Waveforms


Wave - Default
Msgs
/testbench/F
/testbench/A
/testbench/B
/testbench/Y
/testbench/zero
Now
Cursor 1
testbench.sv
alu.sv
Wave


Wave - Default
Msgs
/testbench/F
/testbench/A
/testbench/B
/testbench/Y
/testbench/zero
Now
Cursor 1
testbench.sv
alu.sv
Wave


Wave - Default
Msgs
/testbench/F
/testbench/A
/testbench/B
/testbench/Y
/testbench/zero
Now
Cursor 1
testbench.sv
alu.sv
Wave


Wave - Default
Msgs
/testbench/F
/testbench/A
/testbench/B
/testbench/Y
/testbench/zero
Now
Cursor 1
testbench.sv
alu.sv
Wave



Workload report:
I nearly spent one entire day reading the concepts from the books and understanding it.
The code work on ModelSim took me approximately 4-5 hours as I encountered some bugs in the code.
The activity to make the testbench read the test vectors takes the most amount of time because some inputs for me didn’t get read properly.
All in all, the project was completed in a approximate duration of 2 days.