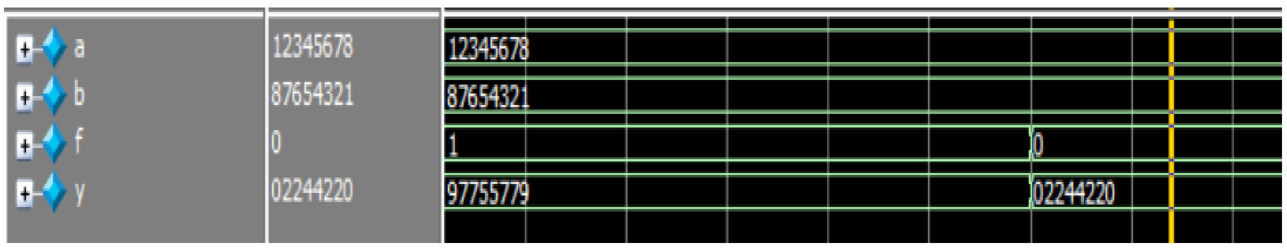
1 more than 10 hours

2

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 100-1	6	00000100	00000001	000000FF	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

6.



3.

```
module alu(input logic [31:0] a, b,
input logic [2:0] f,
output logic [31:0] y,
output logic zero);

wire logic[31:0] B_par;
wire logic[31:0] or_product;
wire logic[31:0] and_product;
wire logic[31:0] S;
wire logic[31:0] S_extended;
 logic[31:0] BB;
wire logic C_out;

assign B_par= ~ b;

always_comb begin

case (f[2])
0 : BB[31:0]=b[31:0];
1 : BB[31:0]=B_par[31:0];
endcase
end

assign and_product= a & BB;
assign or_product = a | BB;
assign {C_out, S} = a + BB + f[2];
assign S_extended = {28'h0000000,3'b000,S[31]};




always_comb begin
case (f[1:0])
2'b00 : y<= and_product;
2'b01 : y<= or_product;
2'b10 : y<= S;
2'b11 : y<= S_extended;
endcase

zero <= (y == 32'h00000000) ? 1 : 0;

end
endmodule
```

4.

```
2_00000000_00000000_00000000_1
2_00000000_FFFFFFFF_FFFFFFFF_0
2_00000001_FFFFFFFF_00000000_1
2_000000FF_00000001_00000100_0
6_00000000_00000000_00000000_1
6_00000000_FFFFFFFF_00000001_0
6_00000001_00000001_00000000_1
6_00000100_00000001_000000FF_0
7_00000000_00000000_00000000_1
7_00000000_00000001_00000001_0
7_00000000_FFFFFFFF_00000000_1
7_00000001_00000000_00000000_1
7_FFFFFFFF_00000000_00000001_0
0_FFFFFFFF_FFFFFFFF_FFFFFFFF_0
0_FFFFFFFF_12345678_12345678_0
0_12345678_87654321_02244220_0
0_00000000_FFFFFFFF_00000000_1
1_FFFFFFFF_FFFFFFFF_FFFFFFFF_0
1_12345678_87654321_97755779_0
1_00000000_FFFFFFFF_FFFFFFFF_0
1_00000000_00000000_00000000_1
```

5.

```
module testbench;
logic [31:0] a,b;
logic [2:0]    f;
logic [31:0]  y,yexpected;
logic zero,zeroexpected;

logic [31:0] vectornum;
logic [25:0] testvectors[20:0];
logic clk, reset;

alu DUT(
.a(a),
.b(b),
.f(f),
.y(y),
.zero(zero)
);

initial
begin
$readmemh("testbench.tv", testvectors);

{f, a, b, yexpected, zeroexpected} = testvectors[0];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[1];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[2];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;
 {f, a, b, yexpected, zeroexpected} = testvectors[3];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[4];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[5];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[6];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[7];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[8];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[9];
if(y == yexpected)
        $display("Correct");
```

```
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[10];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[11];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[12];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[13];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[14];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[15];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;
 {f, a, b, yexpected, zeroexpected} = testvectors[16];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[17];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[18];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[19];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");
 #5;

 {f, a, b, yexpected, zeroexpected} = testvectors[20];
if(y == yexpected)
        $display("Correct");
else
        $display("Error");

$finish;
end
endmodule
```