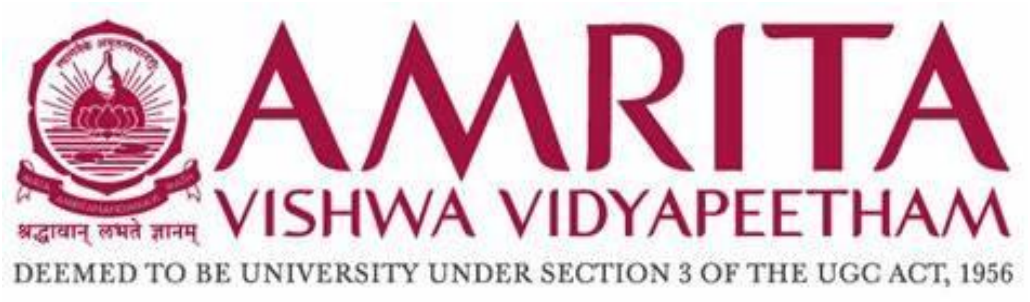


DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

SYSTEM LEVEL PROJECT

19ECE383

VLSI DESIGN LAB

Submitted by

1. Prajwal J G (BL.EN.U4ECE22237)
2. Shivani S (BL.EN.U4ECE22246)
3. Smrithi B (BL.EN.U4ECE22251)
4. Abishai G (BL.EN.U4ECE22263)

Faculty In-charge

Mr. Vignesh V

Aim

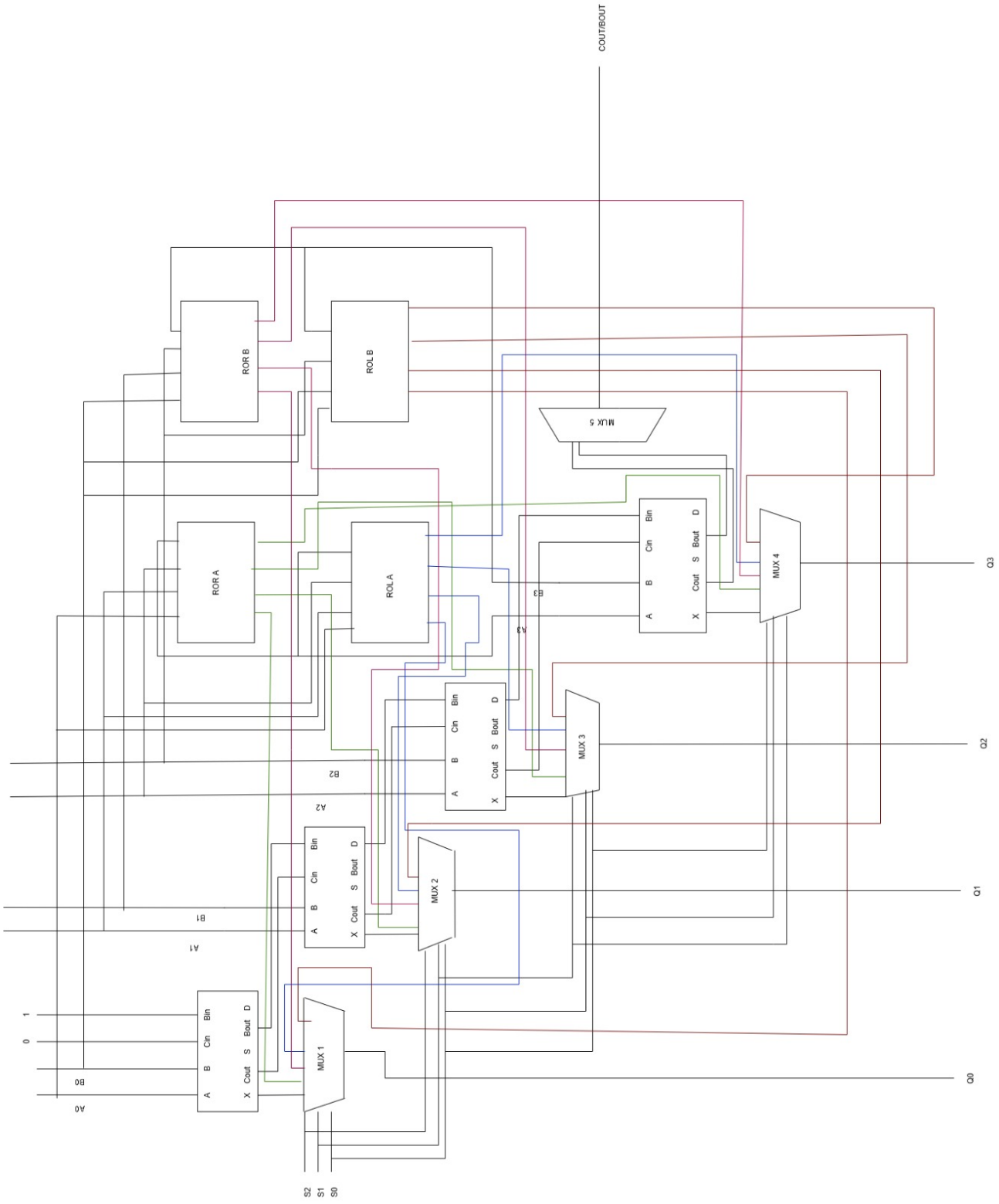
a) Design a 4-bit ALU with ADD, SUB, XNOR, Rotate Left/Right using control MUX
b) Design and synthesize the same using Xilinx Vivado and dump it on the FPGA board (Basys 3)

Tools Required

Cadence Virtuoso and Xilinx Vivado

a) Solution on Virtuoso

4-bit ALU block schematics:



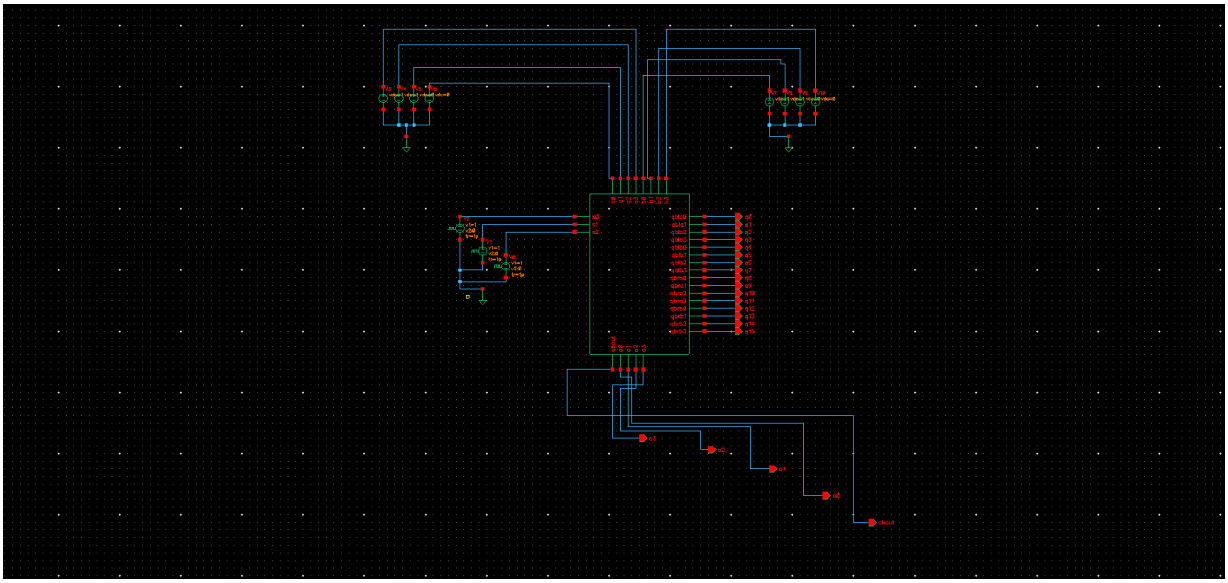


Fig 1. 4-bit ALU final symbol

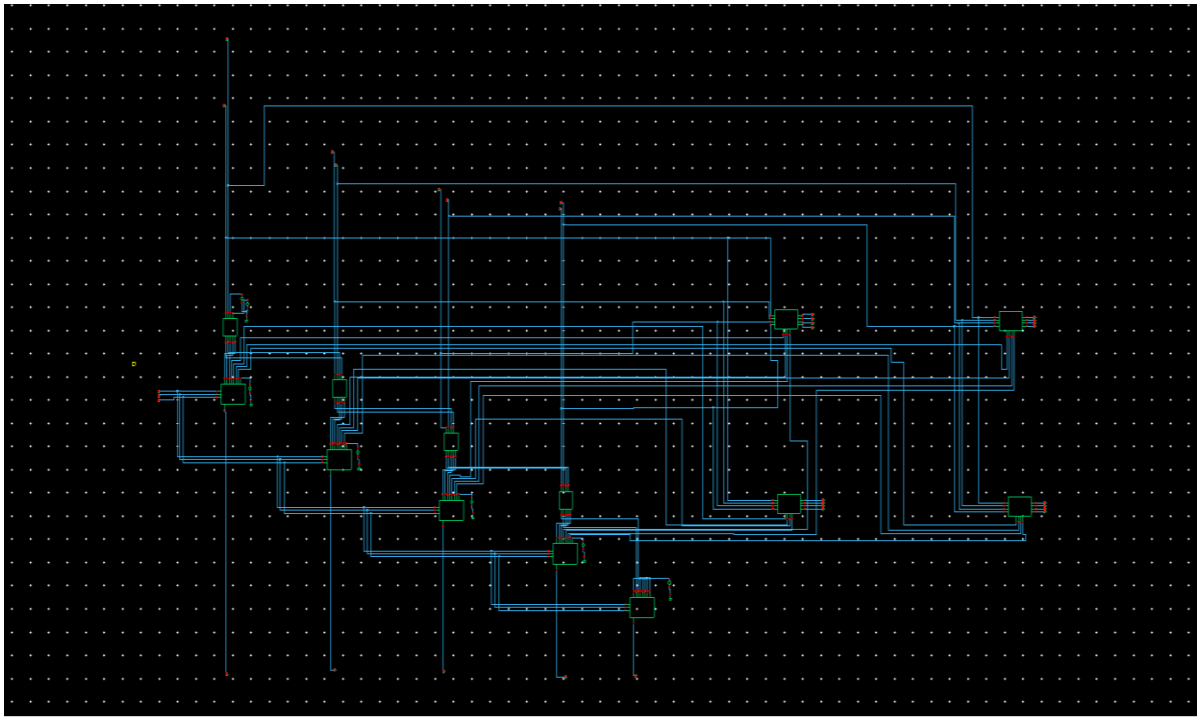


Fig 2. 4-bit ALU final circuit

SELECT INPUTS	OPERATION PERFORMED
000	XNOR
001	Full adder
010	Full subtractor
011	Rotate Right A
100	Rotate Left A
101	Rotate Right B
110	Rotate Left B
111	No operation

Truth Table for the 4-bit ALU

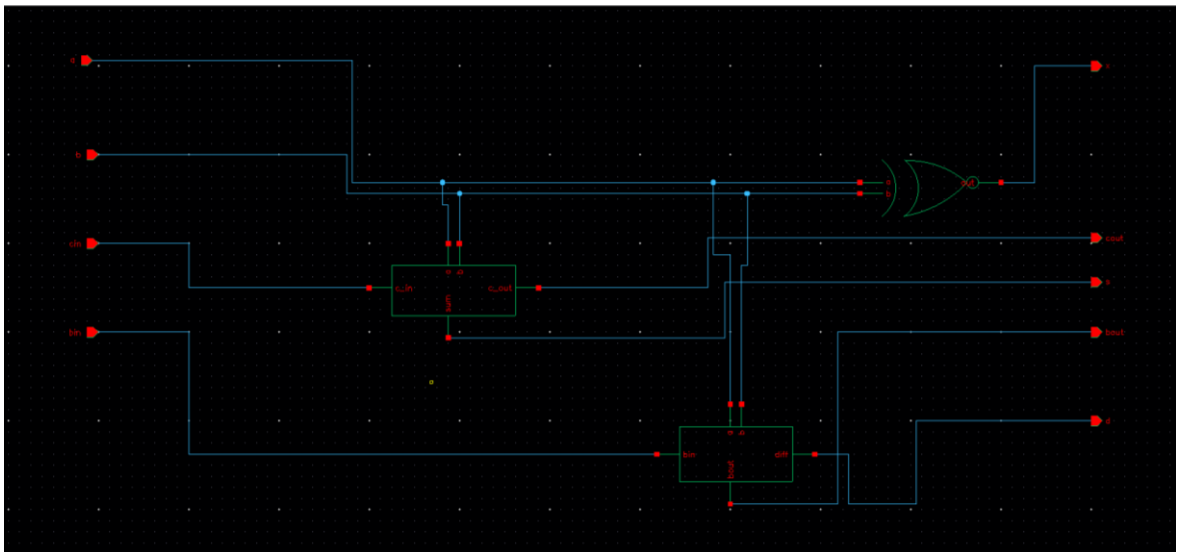


Fig3. 1-bit ALU using full adder and full subtractor blocks and an XNOR gate

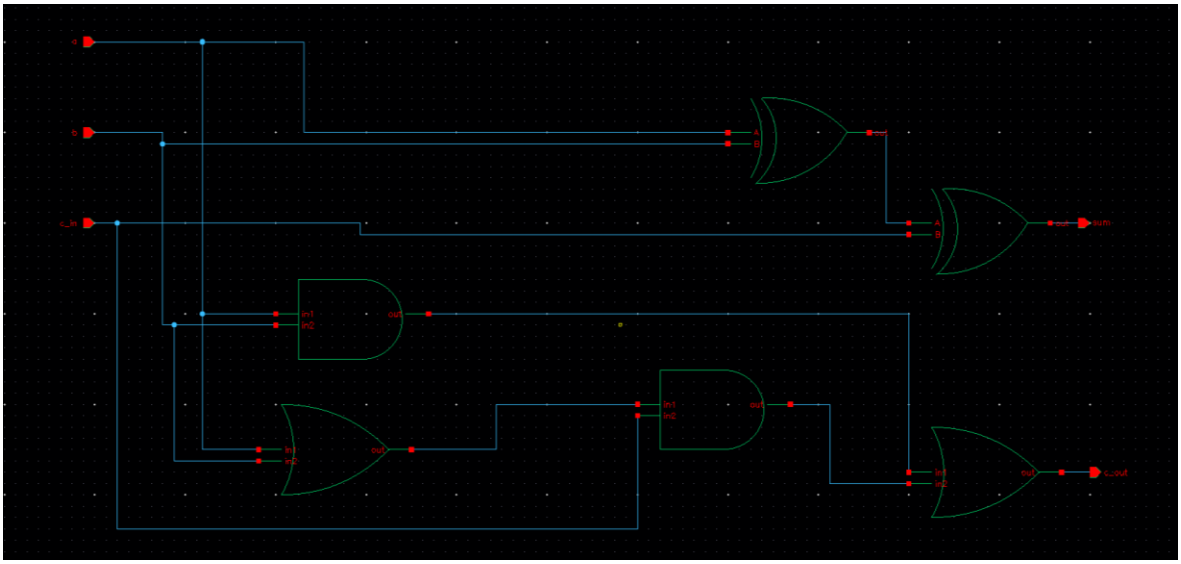


Fig3. Full adder circuit

A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Full adder truth table

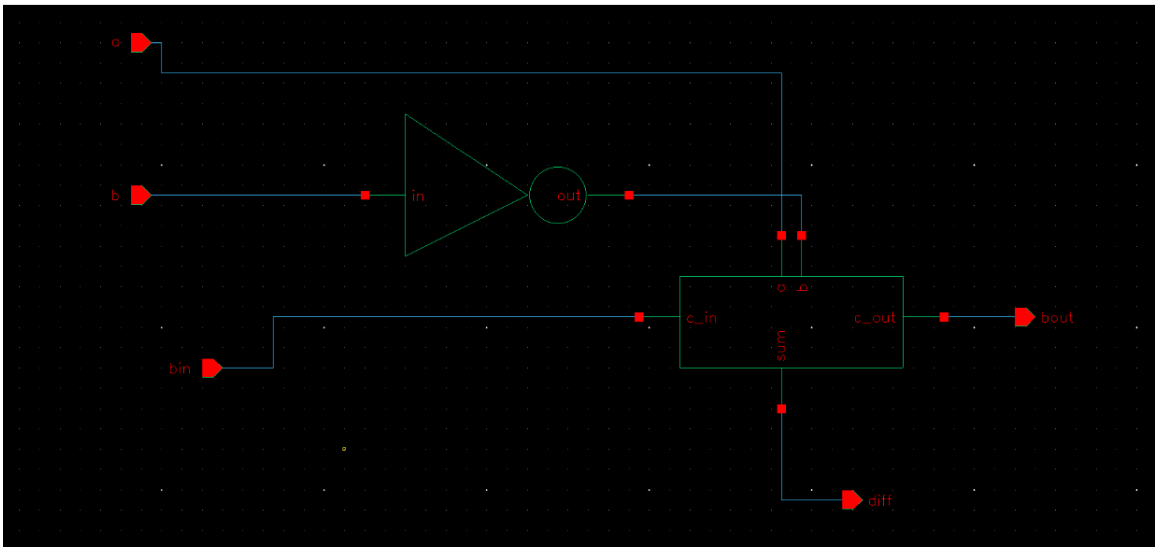


Fig5. Full subtractor circuit using Full adder

Input			Output	
A	B	C	Difference = D	Borrow = B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Full Subtractor Truth Table

Full subtractor truth table

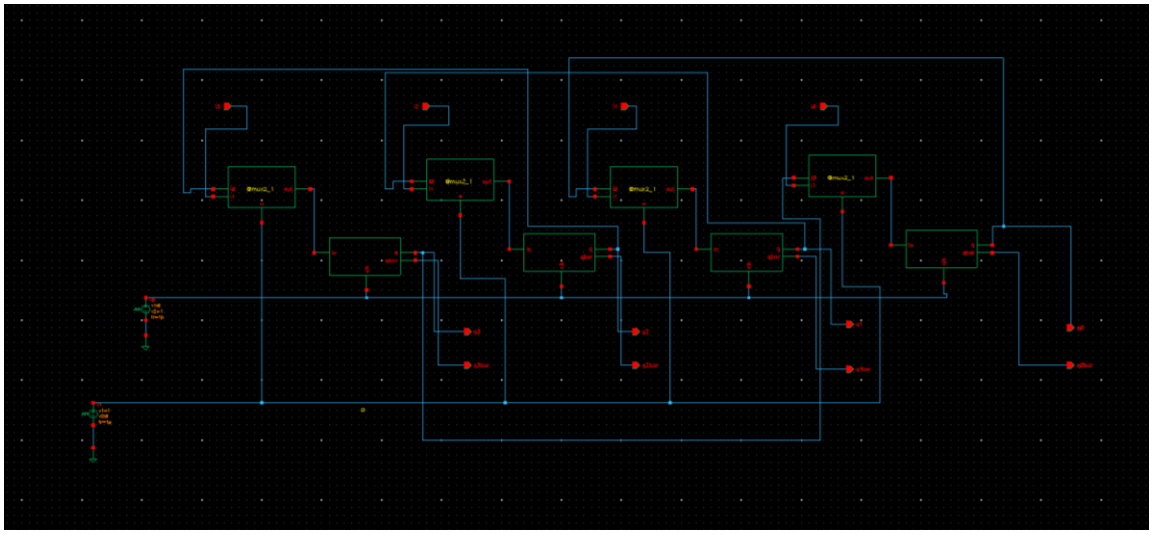


Fig6. Rotate Left Circuit using 2:1 MUX and D-flip flop

INPUT	ROTATE LEFT
i3 i2 i1 i0	i2 i1 i0 i3
i2 i1 i0 i3	i1 i0 i3 i2
i1 i0 i3 i2	i0 i3 i2 i1
i0 i3 i2 i1	i3 i2 i1 i0

Rotate Left Truth Table

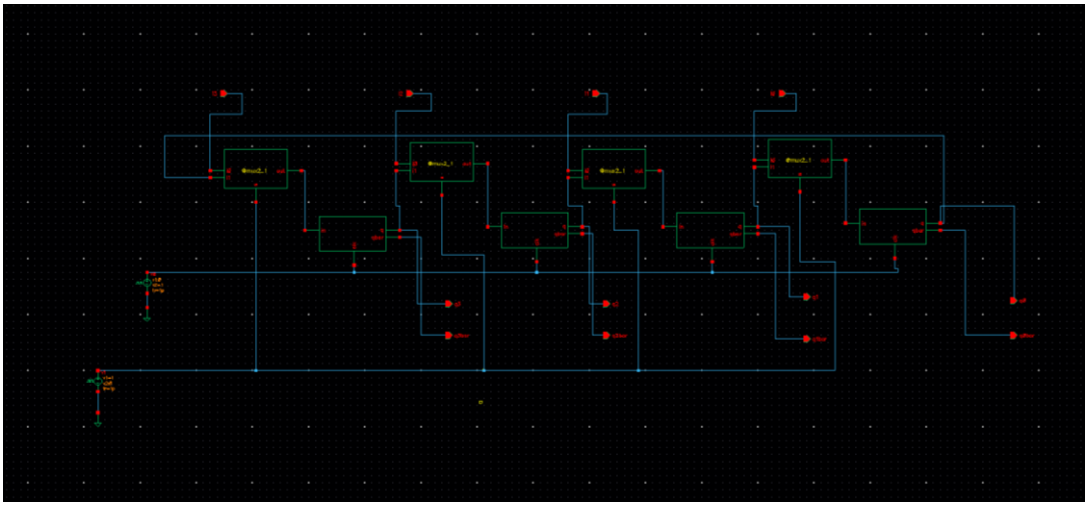
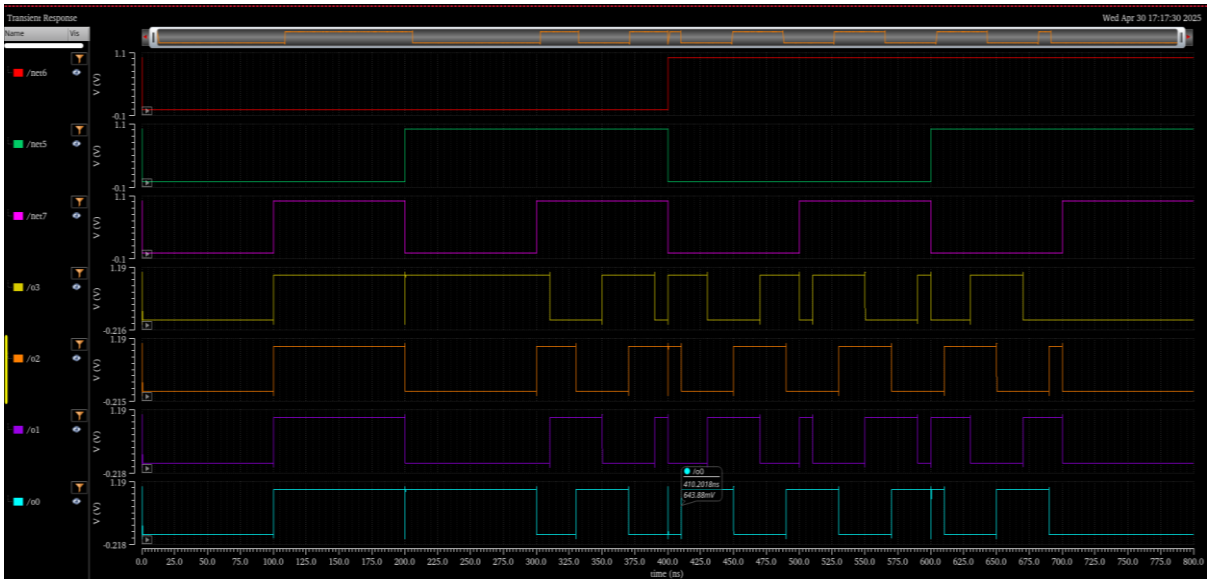


Fig7. Rotate Right circuit using 2:1 MUX and D-flip flop

INPUTS	ROTATE RIGHT
i3 i2 i1 i0	i0 i3 i2 i1
i0 i3 i2 i1	i1 i0 i3 i2
i1 i0 i3 i2	i2 i1 i0 i3
i2 i1 i0 i3	i3 i2 i1 i0

Rotate Right Truth Table

OUTPUT WAVEFORM



Inference

Observed output from the above given waveform for each of the above-mentioned operations.

b) Solution on Vivado

Tools used

Xilinx Vivado, Basys 3 FPGA board

Verilog code:

```
`timescale 1ns/1ps


module fa(s, cout, a, b, cin);
output s, cout;
input a, b, cin;


assign s = (a ^ (b ^ cin));
assign cout = ((a & b) | cin & (a ^ b));


endmodule


module fs(d, bout, x, y, bin);
output d, bout;
input x, y, bin;


assign d = (x ^ (y ^ bin));
assign bout = ((~x & bin) | (~x & y) | (y & bin));


endmodule


module mux4to1(y, s0, s1, i3, i2, i1, i0);
```

```
output reg y;
input wire s0, s1;
input wire i0, i1, i2, i3;

always@(*)
case({s1,s0})
2'b00:y=i3;
2'b01:y=i2;
2'b10:y=i1;
2'b11:y=i0;
default:$display("invalid");
endcase
endmodule

module right_rotate_a(y,w,s0,s1);
input [3:0]w;
input s0,s1;
output [3:0] y;

mux4to1 m1(y[3],s0,s1,w[0],w[1],w[2],w[3]);
mux4to1 m2(y[2],s0,s1,w[3],w[0],w[1],w[2]);
mux4to1 m3(y[1],s0,s1,w[2],w[3],w[0],w[1]);
mux4to1 m4(y[0],s0,s1,w[1],w[2],w[3],w[0]);

endmodule

module left_rotate_a(y,w,s0,s1);
input [3:0]w;
```

```
input s0,s1;
output [3:0] y;

mux4to1 m5(y[3],s0,s1,w[2],w[1],w[0],w[3]);
mux4to1 m6(y[2],s0,s1,w[1],w[0],w[3],w[2]);
mux4to1 m7(y[1],s0,s1,w[0],w[3],w[2],w[1]);
mux4to1 m8(y[0],s0,s1,w[3],w[2],w[1],w[0]);

endmodule

module right_rotate_b(y,w,s0,s1);
input [3:0]w;
input s0,s1;
output [3:0] y;

mux4to1 m9(y[3],s0,s1,w[0],w[1],w[2],w[3]);
mux4to1 m10(y[2],s0,s1,w[3],w[0],w[1],w[2]);
mux4to1 m11(y[1],s0,s1,w[2],w[3],w[0],w[1]);
mux4to1 m12(y[0],s0,s1,w[1],w[2],w[3],w[0]);

endmodule

module left_rotate_b(y,w,s0,s1);
input [3:0]w;
input s0,s1;
output [3:0] y;

mux4to1 m13(y[3],s0,s1,w[2],w[1],w[0],w[3]);
```

```
mux4to1 m14(y[2],s0,s1,w[1],w[0],w[3],w[2]);
mux4to1 m15(y[1],s0,s1,w[0],w[3],w[2],w[1]);
mux4to1 m16(y[0],s0,s1,w[3],w[2],w[1],w[0]);

endmodule



module mux8to1(y, s0, s1, s2, i0, i1, i2, i3, i4, i5, i6, i7);
output reg [3:0] y;
input wire s0, s1, s2;
input wire [3:0]i0, i1, i2, i3, i4, i5, i6, i7;

always@(*)
case({s2,s1,s0})
3'b000:y=i0;
3'b001:y=i1;
3'b010:y=i2;
3'b011:y=i3;
3'b100:y=i4;
3'b101:y=i5;
3'b110:y=i6;
3'b111:y=i7;
default:$display("invalid");
endcase
endmodule

module one_bit_alu(x_out, s_out, c_out, d_out, b_out, a, b, cin, bin);
output x_out, s_out, c_out, d_out, b_out;
```

```
input a, b, cin, bin;

assign x_out = ~(a^b);

fa fa1(s_out, c_out, a, b, cin);

fs fs1(d_out, b_out, a, b, bin);

endmodule

module four_bit_alu(Y,Y2, A, B, S, S_rotate, COUT, BOUT);
output [3:0]Y;
output Y2;
output COUT, BOUT;
input [3:0]A, B;
input [2:0]S;
input [1:0]S_rotate;

wire [3:0]XNOR,SUM, DIFF, right_a, left_a, right_b, left_b;
wire co1, co2, co3, bo1, bo2, bo3;

one_bit_alu aloo1(XNOR[0], SUM[0], co1, DIFF[0], bo1, A[0], B[0], 1'b0, 1'b0);
one_bit_alu aloo2(XNOR[1], SUM[1], co2, DIFF[1], bo2, A[1], B[1], co1, bo1);
one_bit_alu aloo3(XNOR[2], SUM[2], co3, DIFF[2], bo3, A[2], B[2], co2, bo2);
one_bit_alu aloo4(XNOR[3], SUM[3], COUT, DIFF[3], BOUT, A[3], B[3], co3, bo3);

right_rotate_a ra(right_a,A,S_rotate[0],S_rotate[1]);
right_rotate_b rb(right_b,B,S_rotate[0],S_rotate[1]);
```

```
left_rotate_a la(left_a,A,S_rotate[0],S_rotate[1]);
left_rotate_b lb(left_b,B,S_rotate[0],S_rotate[1]);

mux8to1 m81(Y, S[0], S[1], S[2], XNOR, SUM, DIFF, right_a, left_a, right_b, left_b, 4'b0);
wire w1, w2;
and a1(w1, S[0], ~S[1], ~S[2]);
and a2(w2, ~S[0], S[1], ~S[2]);
or a3(Y2, w1, w2);

endmodule
```

Testbench:

```
module testbench;
reg [3:0]A, B;
reg [2:0]S;
reg [1:0]S_rotate;
wire COUT, BOUT;
wire [3:0]Y;
wire Y2;
four_bit_alu alu(.Y(Y), .Y2(Y2), .A(A), .B(B), .S(S), .S_rotate(S_rotate), .COUT(COUT),
.BOUT(BOUT));

initial
begin

$monitor("Time = %0t A=%d, B=%d, S=%d, Sr=%d, Y=%d",$time, A, B, S, S_rotate, Y);
A = 4'b1100;
```

```
B = 4'b0011;
S = 3'b000;
S_rotate = 2'b00;

#5 S = 3'b001;
#5 S = 3'b010;

#5 S = 3'b011;
#5 S_rotate = 2'b01;
#5 S_rotate = 2'b10;
#5 S_rotate = 2'b11;

#5 S = 3'b100;
S_rotate = 2'b00;
#5 S_rotate = 2'b01;
#5 S_rotate = 2'b10;
#5 S_rotate = 2'b11;

#5 S = 3'b101;
S_rotate = 2'b00;
#5 S_rotate = 2'b01;
#5 S_rotate = 2'b10;
#5 S_rotate = 2'b11;

#5 S = 3'b110;
S_rotate = 2'b00;
#5 S_rotate = 2'b01;
#5 S_rotate = 2'b10;
```

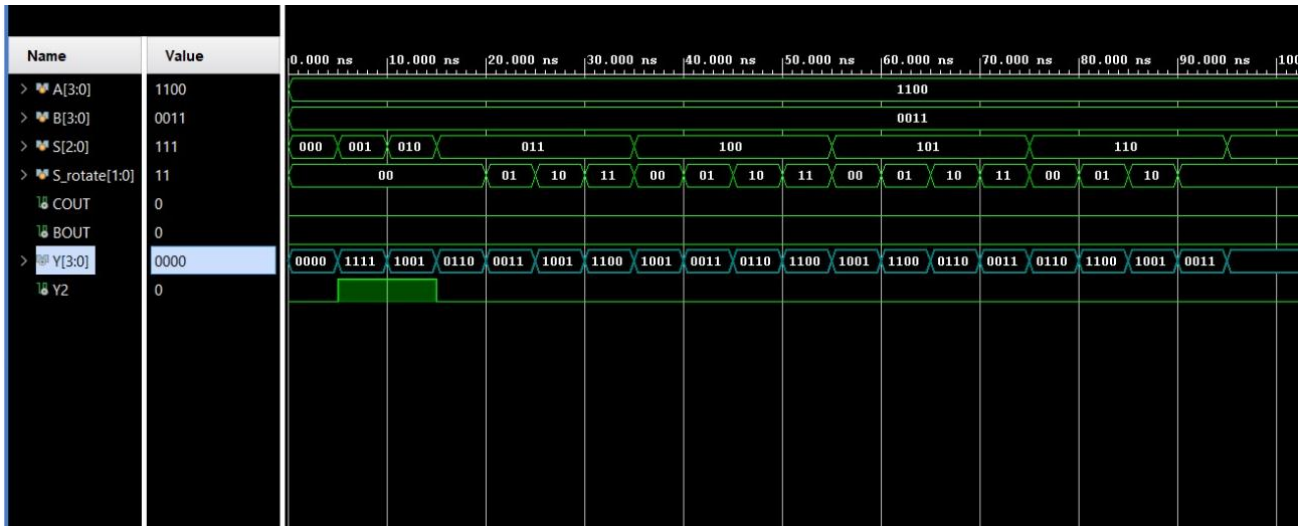
#5 S_rotate = 2'b11;

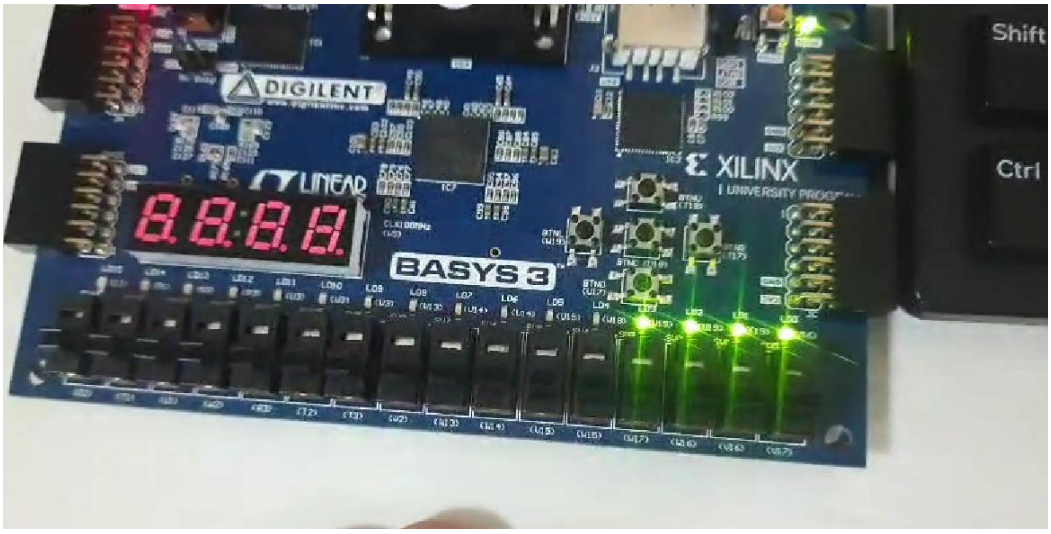
#5 S = 3'b111;

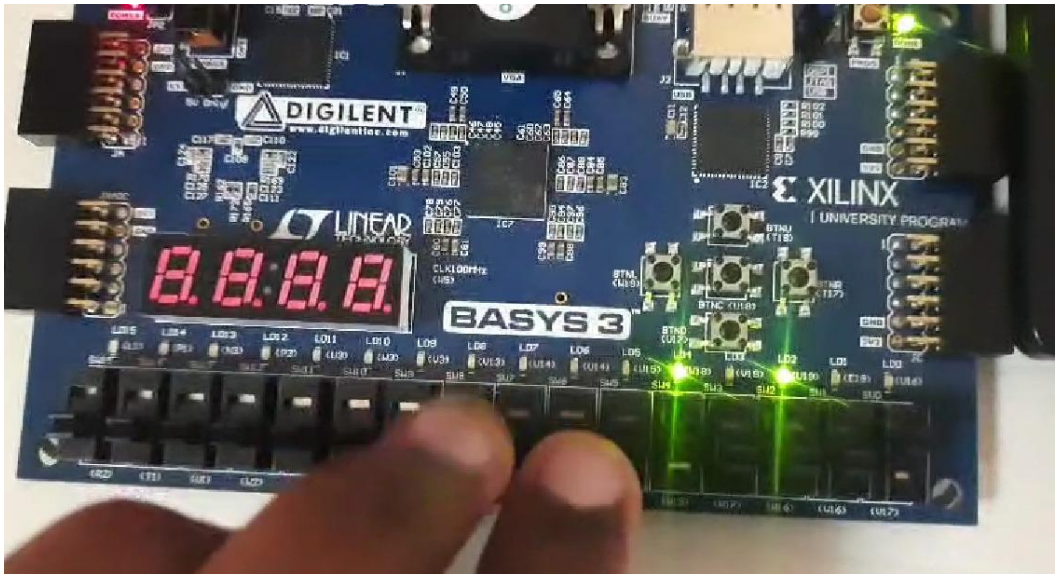
end

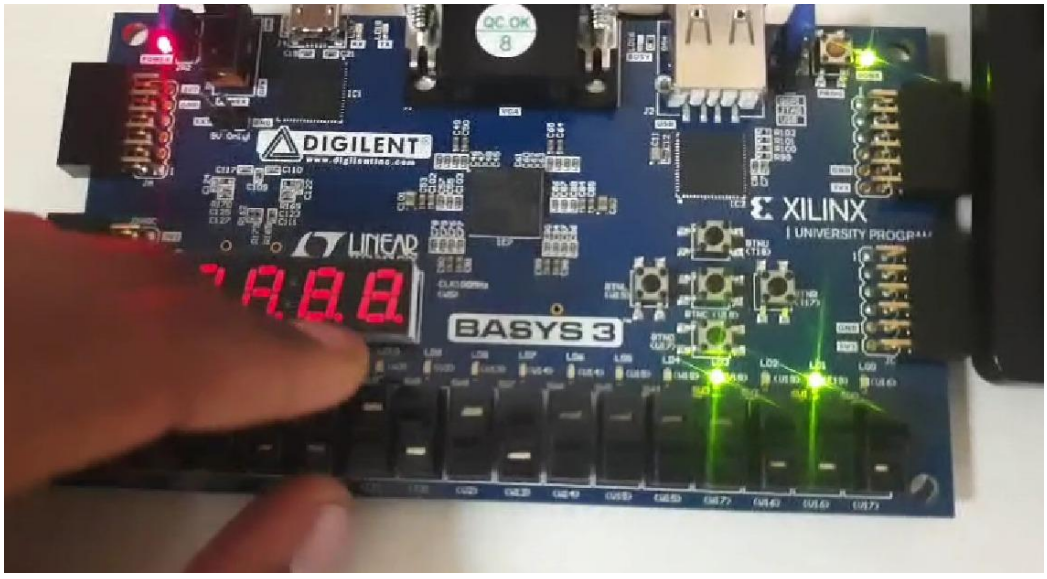
endmodule

Outputs:






DIGILENT
XILINX
BASYS 3


QC OK
DIGILENT
XILINX
BASYS 3