B.M.S. College of Engineering

(Autonomous Institution affiliated to VTU, Belagavi)

Bengaluru – 19

Department of Computer Science and Engineering

Verilog Program List

19CSPC34

Laboratory Manual

(AUTONOMOUS SCHEME 2019)

B.M.S. College of Engineering

(Autonomous Institution affiliated to VTU, Belagavi)

Bengaluru - 19

Department of Computer Science and Engineering

Laboratory Certificate

This is to certify that Mr. _____ _____has satisfactorily completed the course of Experiments in Practical _____ prescribed by the Department during the year _____

Name of the Candidate: _____

USN No.: _____ Semester: _____

Marks	
Max. Marks	Obtained
10	

Marks in Words	

Signature of the staff in-charge **Head of the Department**

Date:

B.M.S. College of Engineering

(Autonomous Institution affiliated to VTU, Belagavi)

Bengaluru - 19

Department of Computer Science and Engineering

Laboratory Certificate

This is to certify that Mr. _____ _____has satisfactorily completed the course of Experiments in Practical _____ prescribed by the Department during the year _____

Name of the Candidate: _____

USN No.: _____ Semester: _____

Marks	
Max. Marks	**Obtained**
10	

Marks in Words	

Signature of the staff in-charge **Head of the Department**

Date:

B.M.S. College of Engineering

(Autonomous Institution affiliated to VTU, Belagavi)

Bengaluru - 19

Department of Computer Science and Engineering

Laboratory Certificate

This is to certify that Mr. _____ _____has satisfactorily completed the course of Experiments in Practical _____ prescribed by the Department during the year _____

Name of the Candidate: _____

USN No.: _____ Semester: _____

Marks	
Max. Marks	**Obtained**
10	

Marks in Words	

Signature of the staff in-charge **Head of the Department**

Date:

B.M.S. College of Engineering

(Autonomous Institution affiliated to VTU, Belagavi)

Bengaluru - 19

Department of Computer Science and Engineering

Laboratory Certificate

This is to certify that Mr. _____ _____has satisfactorily completed the course of Experiments in Practical _____ prescribed by the Department during the year _____

Name of the Candidate: _____

USN No.: _____ Semester: _____

Marks	
Max. Marks	**Obtained**
10	

Marks in Words	

Signature of the staff in-charge **Head of the Department**

Date:

B.M.S. College of Engineering

(Autonomous Institution affiliated to VTU, Belagavi)

Bengaluru - 19

Department of Computer Science and Engineering

Laboratory Certificate

This is to certify that Mr. _____ _____has satisfactorily completed the course of Experiments in Practical _____ prescribed by the Department during the year _____

Name of the Candidate: _____

USN No.: _____ Semester: _____

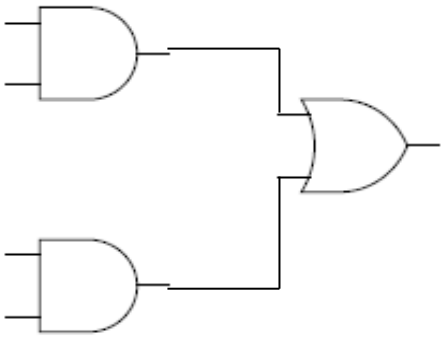
Marks	
Max. Marks	**Obtained**
10	

Marks in Words	

Signature of the staff in-charge **Head of the Department**

Date:

Verilog Program List
19CSPC34
Laboratory Manual

Serial No.	Title
	CYCLE I **Structural Modeling**
1.	Write HDL implementation for the following Logic a. AND/OR/NOT Simulate the same using structural model and depict the timing diagram for valid inputs.
2.	Write HDL implementation for the following Logic a. NAND/NOR Simulate the same using structural model and depict the timing diagram for valid inputs.
3.	Write HDL implementation for the following Logic  Simulate the same using structural model and depict the timing diagram for valid inputs.
4.	Write HDL implementation for a 4:1 Multiplexer. Simulate the same using structural model and depict the timing diagram for valid inputs.
5.	Write HDL implementation for a 2-to-4 decoder. Simulate the same using structural model and depict the timing diagram for valid inputs.
6.	Write HDL implementation for a 4-to-2 encoder. Simulate the same using structural model and depict the timing diagram for valid inputs.

	CYCLE II **Behavior Modeling**
7.	Write HDL implementation for a RS flip-flop using behavioral model. Simulate the same using Behavior model and depict the timing diagram for valid inputs.
8.	Write HDL implementation for a JK flip-flop using behavioral model. Simulate the same using Behavior model and depict the timing diagram for valid inputs.
9.	Write HDL implementation for a 4-bit right shift register using behavioral model. Simulate the same using Behavior model and depict the timing diagram for valid inputs.
10.	Write HDL implementation for a 3-bit up-counter using behavioral model. Simulate the same using Behavior model and depict the timing diagram for valid inputs.
	CYCLE III **Dataflow Modeling**
11.	Write HDL implementation for AND/OR/NOT gates using data flow model. Simulate the same using Dataflow model and depict the timing diagram for valid inputs.
12.	Write HDL implementation for a 3-bit full adder using data flow model. Simulate the same using Dataflow model and depict the timing diagram for valid inputs.

Verilog Program List-19CSPC34

SCHEME OF CONDUCT AND EVALUATION

CLASS: III SEMESTER
YEAR: 2019-20
EVALUATION SCHEME Tutorial Test: 1 hour

Expt. No.	TITLE	Max. Marks	Marks Obtained	Signature
1.	K-Map and Quine Mcclusky Method	2		
2.	AND/OR/NOT	3		
3.	NAND/NOR			
4.	Logic diagram			
5.	Multiplexer			
6.	Decoder			
7.	Encoder			
8.	RS			
9.	JK			
10.	Shift right			
11.	Counter			
12.	AND/OR/NOT – data flow			
13.	3-bit full adder			
	Test: Viva – 2 Marks + Writeup – 1 Mark + Execution – 2 Marks	5		
	TOTAL MARKS	10		

Verilog Program List-19CSPC34

Rubrics

Sl.No	Criteria	Excellent	Good	Average	Poor	Max Score
A	Design & specifications	1	0.5	0.25	**0**	**1**
B	Expected output	2	1	0.5	**0**	**2**
	Record					
C	Simulation/ Conduction of the experiment	3	2	1	**0**	**3**
D	K-Map and Quine Mcclusky Method	2	1	0.5	**0**	**2**
Viva						**2**
Total						**10**

STRUCTURAL MODELING

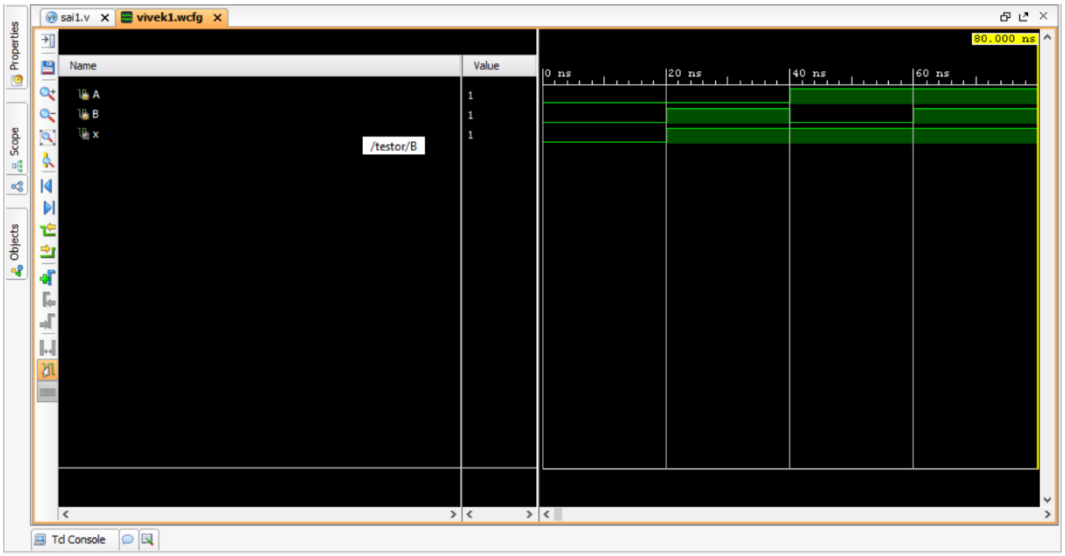
Experiment 1

1. Write HDL implementation for the following Logic
    a. AND/OR/NOT

Simulate the same using structural model and depict the timing diagram for valid inputs.

MAIN MODULE (OR GATE GIVEN)

```
module or_gate(A,B,Y);
input A,B;   // defines two input port
output Y;    // defines one output port
or g1(Y,A,B);

 /*Gate declaration with predefined keyword or representing logic   OR, g1 is optional user
defined gate identifier */

endmodule


module testor;
reg A,B;
wire x;
or_gate org(A,B,x);
initial
begin
A= 1'b0; B=1'b0;
#20
A= 1'b0; B=1'b1;
#20
A= 1'b1; B=1'b0;
#20
A= 1'b1; B=1'b1;
#20
$finish;
end

endmodule
```


sai1.v
vivek1.wcfg
Properties
Scope
Objects
Name
Value
A
B
x
1
1
1
/testor/B
80.000 ns
0 ns
20 ns
40 ns
60 ns
Tcl Console

Experiment 2

Write HDL implementation for the following Logic

a. NAND/NOR

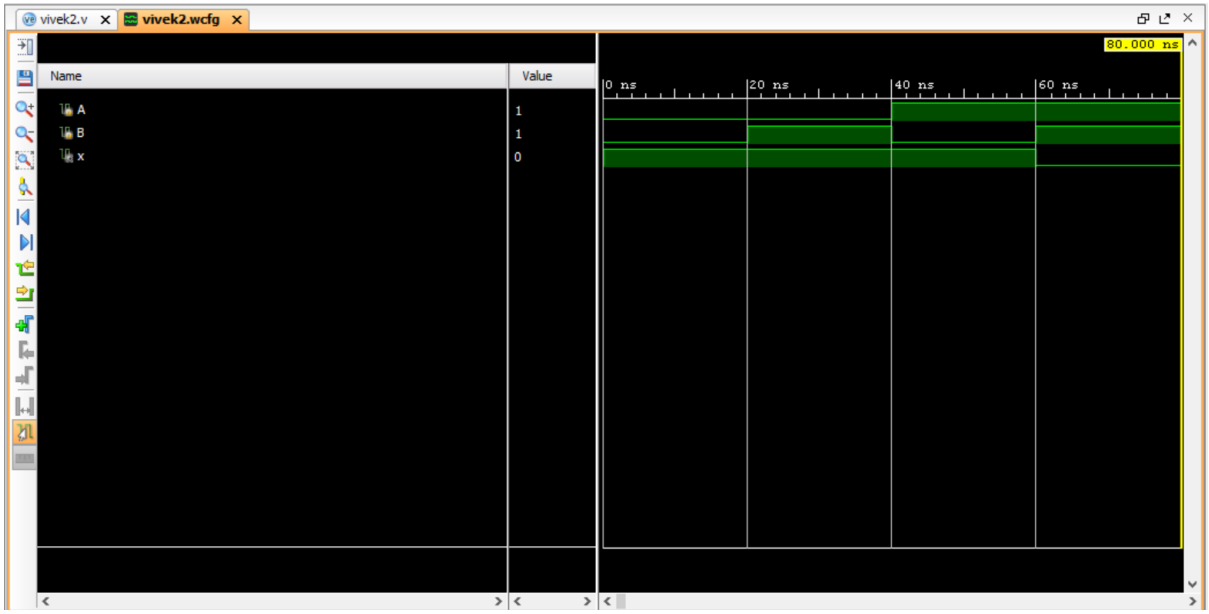
Simulate the same using structural model and depict the timing diagram for valid inputs.

MAIN MODULE(FOR NAND GATE)

```
module nand_gate(A,B,Y);
input A,B;

// defines two input port

output Y;

// defines one output port

nand g1(Y,A,B);

endmodule
```

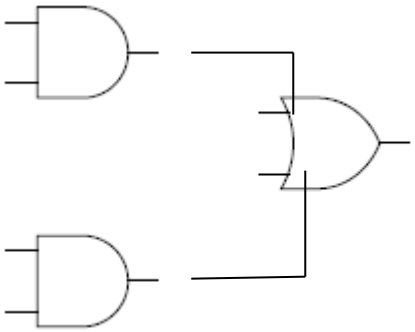
TEST MODULE

```
module testnand;
reg A,B;
wire x;
nand_gate nandg(A,B,x);
initial
begin
A= 1'b0; B=1'b0;
#20
A= 1'b0; B=1'b1;
#20
A= 1'b1; B=1'b0;
#20
A= 1'b1; B=1'b1;
#20
$finish;
end

endmodule
```


vivek2.v
vivek2.wcfg
Name
Value
A
1
B
1
x
0
80.000 ns
0 ns
20 ns
40 ns
60 ns

Experiment 3

Write HDL implementation for the following Logic



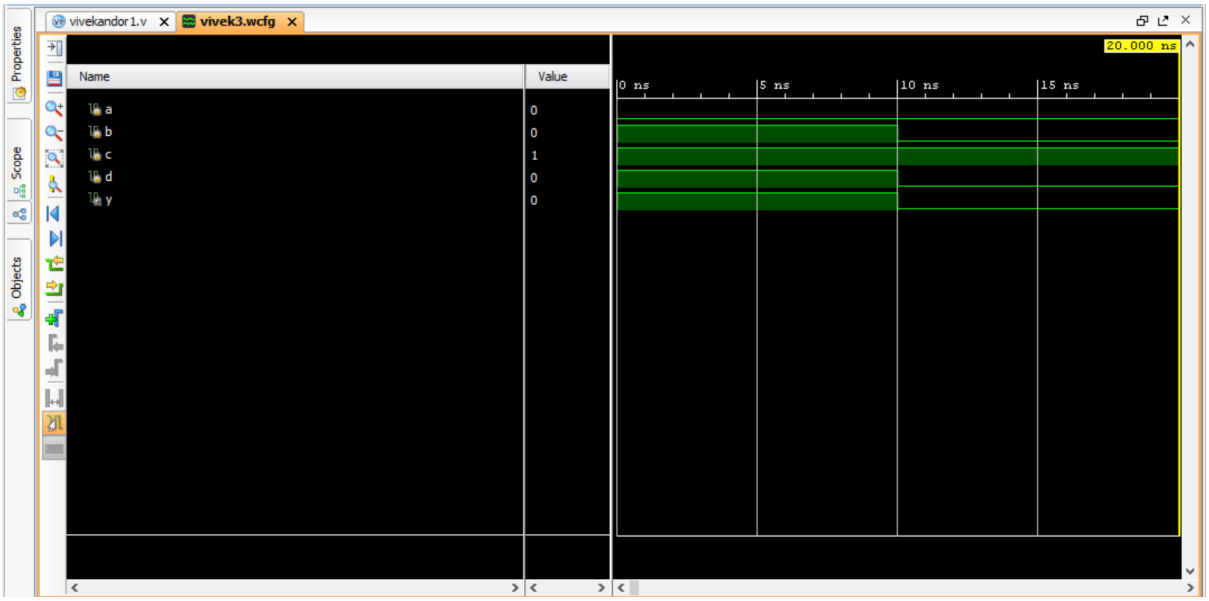
Simulate the same using structural model and depict the timing diagram for valid inputs.

MAIN MODULE

```
module addor(A,B,C,D,Y);
input A,B,C,D;
output Y;
wire and_opl, and_op2;
and gl(and_opl,A,B);
and g2(and_op2,C,D);

 // g2 represents lower A.ND

or g3(Y,and_opl,and_op2);

 // g3 represents the OR gate

endmodule
```

TEST MODULE

```
module test andor;
reg a,b,c,d;
wire y;
addor ao(a,b,c,d,y);
initial
begin
a=0; b=1; c=1; d=1; #10
a=0; b=0; c=1; d=0; #10
$finish;
end

endmodule
```


vivekandor1.v
vivek3.wcfg
Properties
Scope
Objects
Name
Value
a
b
c
d
y
0
0
1
0
0
20.000 ns
0 ns
5 ns
10 ns
15 ns

Experiment 4

Write HDL implementation for a 4:1 Multiplexer. Simulate the same using structural model and depict the timing diagram for valid inputs.

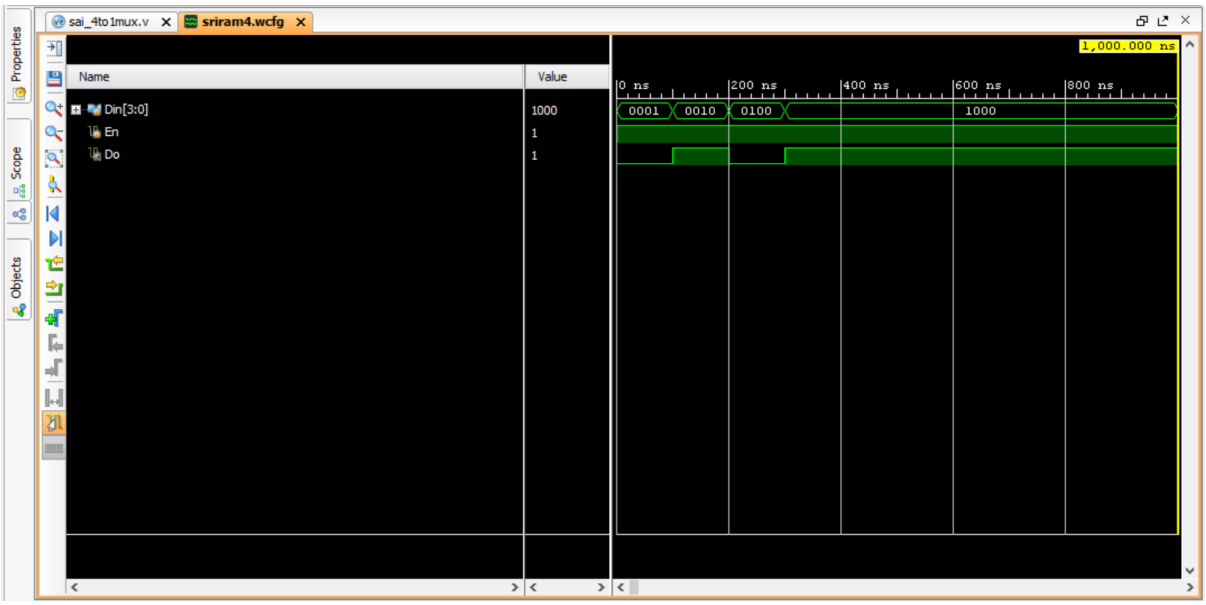
MAIN MODULE

```
module Multiplexer4to1(Do, Din, En);
input En;
input [3:0] Din;
output Do;

reg [1:0]Do;

always @ (En or Din)
begin
if (En)
begin
case (Din)
 4'b0001: Do = 2’b00;
4'b0010: Do = 2’b01;
4'b0100: Do = 2’b10;
4'b1000: Do = 2’b11;
default: Do=2'bzz;
endcase
end
end
endmodule
```

TESTBENCH MODULE

```
module multiplexer_tb;
reg [3:0] Din;
reg En;
wire Do;

    Multiplexer4to1 mux(
                        .Do(Do),
                        .Din(Din),
                        .En(En)
                    );
initial begin
// Initialize Inputs
En = 1;
Din = 4'b0001; #100;
Din = 4'b0010; #100;
Din = 4'b0100; #100;
Din = 4'b1000; #100;
end
```

endmodule


sai_4to1mux.v
sriram4.wcfg
Properties
Scope
Objects
Name
Value
Din[3:0]
1000
En
1
Do
1
1,000.000 ns
0 ns
200 ns
400 ns
600 ns
800 ns
0001
0010
0100
1000

Experiment 5

Write HDL implementation for a 2-to-4 decoder. Simulate the same using structural model and depict the timing diagram for valid inputs.

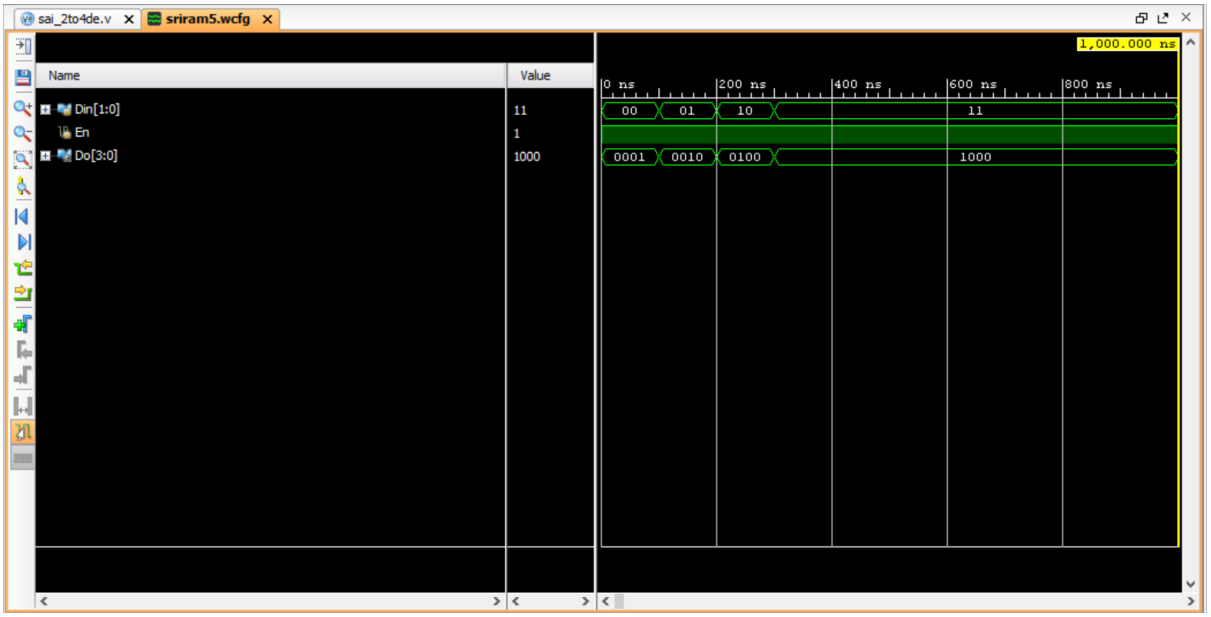
MAIN MODULE

```
module decoder_case(Do, Din, En);
input En;
input [1:0] Din;
output [3:0]Do;

reg [3:0]Do;

always @ (En or Din)
begin
if (En)
begin
case (Din)
                    2'b00: Do = 4'b0001;
                    2'b01: Do = 4'b0010;
                    2'b10: Do = 4'b0100;
                    2'b11: Do = 4'b1000;
default: Do=4'bzzzz;
endcase
end
end
endmodule
```

TEST BENCH MODULE

```
module decoder_tb_v;
reg [1:0] Din;
reg En;
wire [3:0] Do;

decoder_case uut(
      .Do(Do),
      .Din(Din),
      .En(En)
   );
initial begin
      // Initialize Inputs
      En = 1;
      Din =2'b00; #100;
      Din = 2'b01; #100;
       Din = 2'b10; #100;
```

```
        Din = 2'b11; #100;
    end
    endmodule
```



Experiment 6

Write HDL implementation for a 4-to-2 encoder. Simulate the same using structural model and depict the timing diagram for valid inputs.

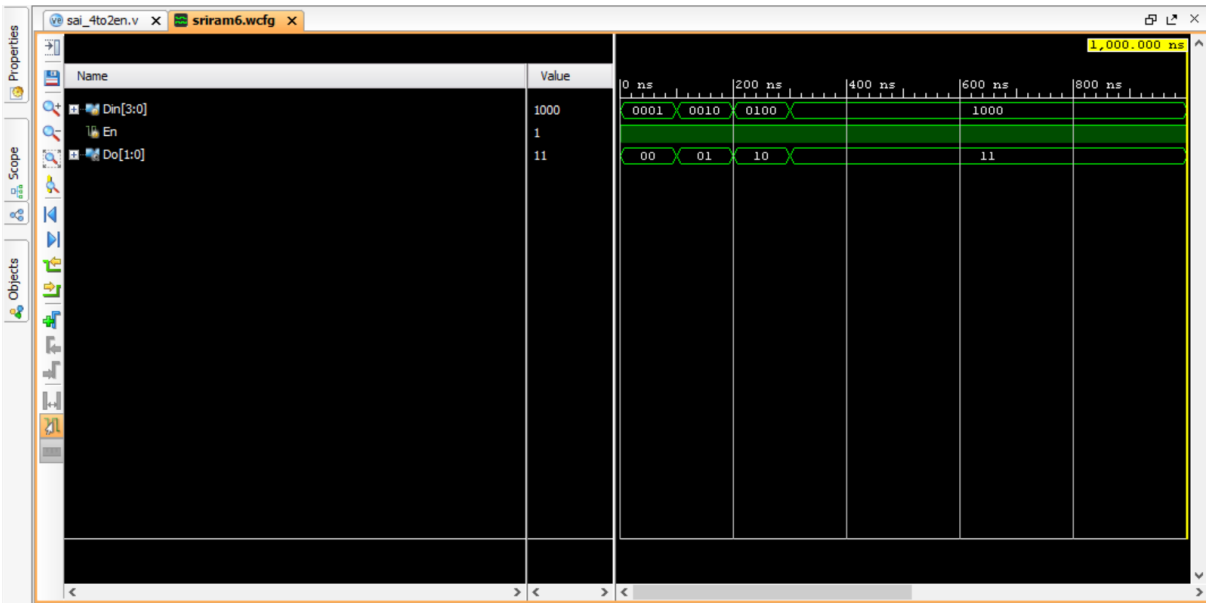
MAIN MODULE

```
module Encoder(Do, Din, En);
input En;
input [3:0] Din;
output [1:0]Do;

reg [1:0]Do;

always @ (En or Din)
begin
if (En)
begin
case (Din)
        4'b0001: Do = 2'b00;
        4'b0010: Do = 2'b01;
        4'b0100: Do = 2'b10;
        4'b1000: Do = 2'b11;
default: Do=2'bzz;
endcase
end
end
endmodule
```

TESTBENCH MODULE

```
module encoder_tb_v;
reg [3:0] Din;
reg En;
wire [1:0] Do;

  Encoder uut(
    .Do(Do),
    .Din(Din),
    .En(En)
  );
initial begin
    // Initialize Inputs
    En = 1;
    Din = 4'b0001; #100;
    Din = 4'b0010; #100;
    Din = 4'b0100; #100;
```

```
        Din = 4'b1000; #100;
    end
    endmodule
```



BEHAVIOR MODELING

Experiment 7

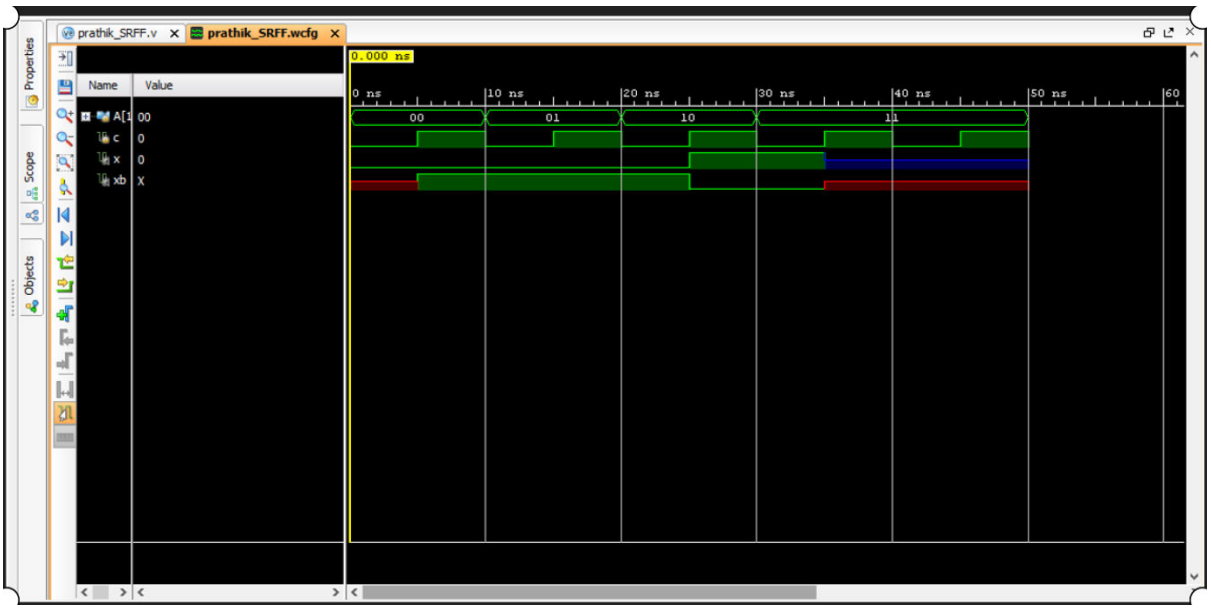
Write HDL implementation for a SR flip-flop using behavioral model. Simulate the same using behavioral model and depict the timing diagram for valid inputs.

MAIN MODULE

```
module SR_FF (sr, clk, q, qb);
input [1:0] sr;
input clk;
output reg q=1'b0;
output reg qb;
always @ (posedge clk)
begin
      case (sr)
                  2'b00 : q = q ;
                        2'b01 : q = 1'b0 ;
                        2'b10 : q = 1'b1 ;
                        2'b11 : q = 1'bz ;
      endcase
                  qb =~ q;
      end
endmodule
```

TEST MODULE

```
module testsrflipf;
   reg [1:0] A;
   reg c;
   wire x, xb;
  SR_FF srff(A,c,x,xb);
  initial c=1'b0;
  always #5 c=~c;
  initial
    begin

    A=2'b00; #10
    A=2'b01;#10
    A=2'b10;#10
    A=2'b11;
    #20 $finish;
    end
endmodule
```


prathik_SRFF.v
prathik_SRFF.wcfg
Properties
Scope
Objects
Name
Value
A[1
00
c
0
x
0
xb
X
0.000 ns
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60
00
01
10
11

Experiment 8

Write HDL implementation for a JK flip-flop using behavioral model. Simulate the same using behavioral model and depict the timing diagram for valid inputs.

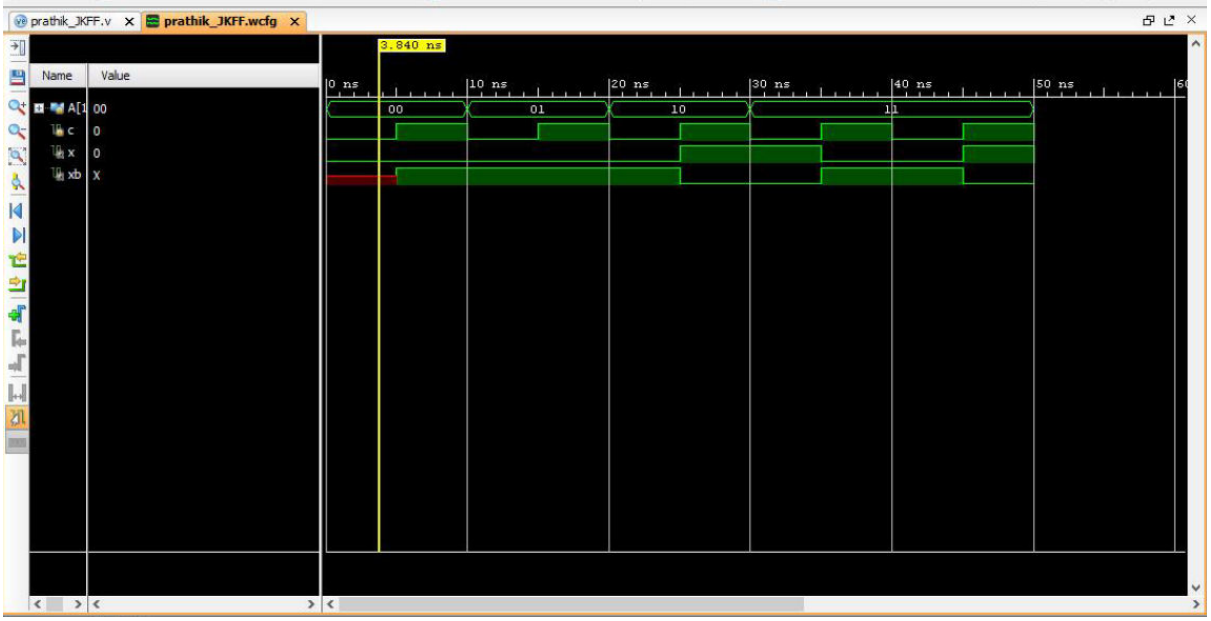
MAIN MODULE

```
module JK_FF (jk, clk, q, qb);
input [1:0] jk;
input clk;
output reg q=1'b0;
output reg qb;
always @ (posedge clk)
begin
      case (jk)
                        2'b00 : q = q ;
                       2'b01 : q = 1'b0 ;
                       2'b10 : q = 1'b1 ;
                       2'b11 : q = ~q ;
      endcase
               qb =~ q;
      end
endmodule
```

TEST MODULE

```
module testjkflipf;
   reg [1:0] A;
   reg c;
   wire x, xb;
  JK_FF jkff(A,c,x,xb);
  initial c=1'b0;
  always #5 c=~c;
  initial
    begin

    A=2'b00; #10
    A=2'b01;#10
    A=2'b10;#10
    A=2'b11;
    #20 $finish;
    end
```

endmodule


prathik_JKFF.v
prathik_JKFF.wcfg
3.840 ns
Name
Value
A[1
00
c
0
x
0
xb
X
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
00
01
10
11

Experiment 9

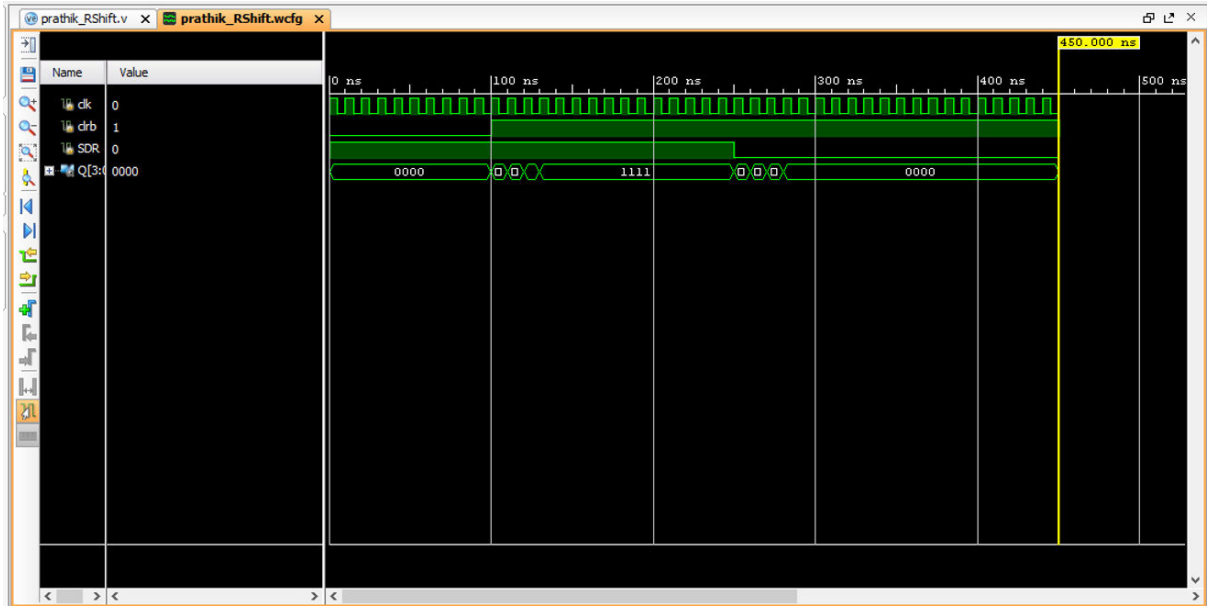
Write HDL implementation for a 4-bit right shift register using behavioral model. Simulate the same using behavioral model and depict the timing diagram for valid inputs.

MAIN MODULE

```
module Rshiftregister(  input clk, input clrb, input SDR, output reg [3:0] Q   );
//serial in, parallel out
   always @ ( posedge(clk), negedge(clrb))

   if (~clrb) Q<=4'b0000;
   else

       Q<={SDR,Q[3:1]};
endmodule
```

TEST MODULE

```
module testRshiftregister;
   reg clk,clrb,SDR;
   wire [3:0]Q;

   Rshiftregister RS(clk, clrb, SDR, Q );
   initial
   begin

   clk =1;
   clrb=0;

   SDR=1;


   #100
   clrb=1;
   SDR=1;
   #150
   SDR=0;
   #200 $finish;
end
//initial and always run in parallel and starts its execution at 0ns
always #5 clk=~clk;
endmodule
```


prathik_RShift.v
prathik_RShift.wcfg
Name
Value
clk
0
drb
1
SDR
0
Q[3:
0000
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
450.000 ns
0000
1111
0000

Experiment 10

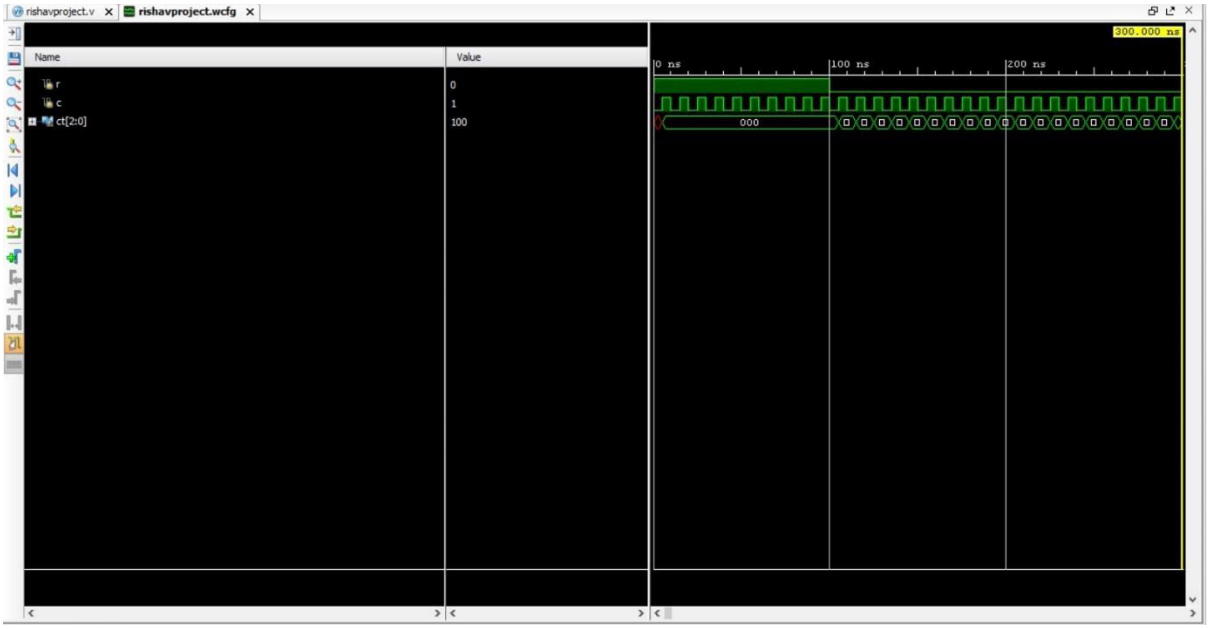
Write HDL implementation for a 3-bit up-counter using behavioral model. Simulate the same using behavioral model and depict the timing diagram for valid inputs.

Main Module

```
module counter_behav ( count,rst,clk);
input rst, clk;
output reg [2:0] count;
always @(posedge (clk))
if (rst)
count<= 3'b000;
else
count<= count + 1;
endmodule
```

TEST MODULE

```
module testmod;
reg r,c;
wire  [2:0] ct;

counter_behav countbeh (ct,r,c);
initial
begin

   r =1;

   c=0;
   #100 r=0;
   #200 $finish;
end
//initial and always run in parallel and starts its execution at 0ns
always #5 c=~c;
endmodule
```


rishavproject.v
rishavproject.wcfg
Name
Value
r
c
ct[2:0]
0
1
100
300.000 ns
0 ns
100 ns
200 ns
000

DATA FLOW MODELING

Experiment 11

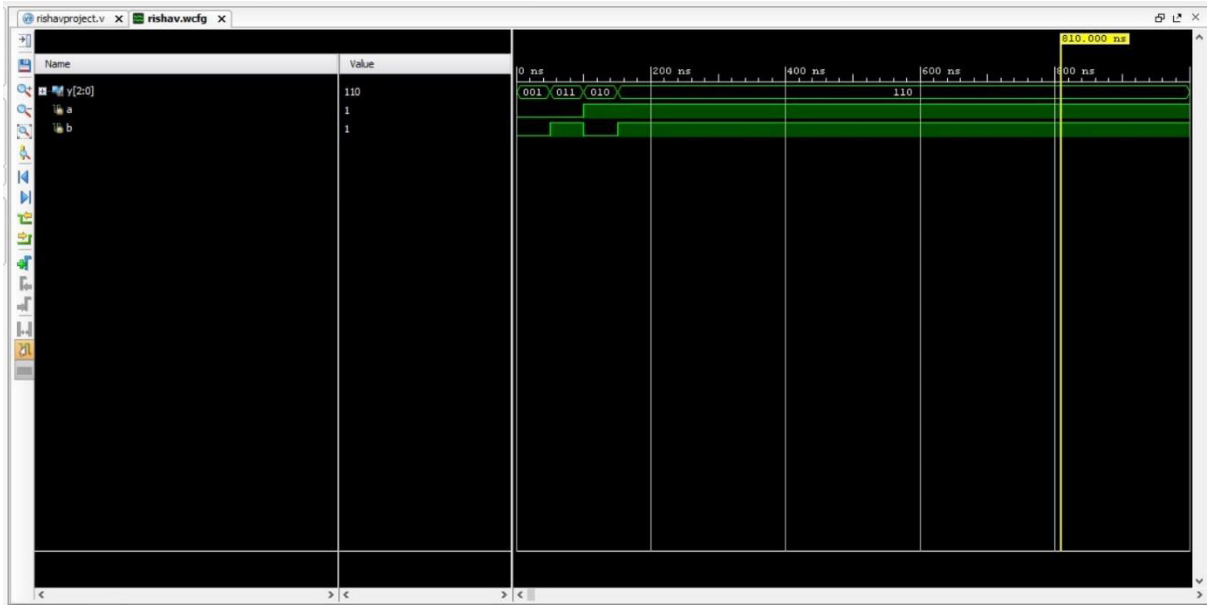
Write HDL implementation for AND/OR/NOT gates using data flow model. Simulate the same using data flow model and depict the timing diagram for valid inputs.

MAIN MODULE

```
module gates(input a, b, output [2:0]y);
assign y[2]= a & b; // AND gate
assign y[1]= a | b; // OR gate
assign y[0]= ~a; // NOT gate
endmodule
```

TESTBENCH MODULE

```
modulegates_tb;
wire [2:0]y;
reg a, b;
gatesdut(.y(y), .a(a), .b(b));
initial
begin
a = 1'b0;
b = 1'b0;
#50;
a = 1'b0;
b = 1'b1;
#50;
a = 1'b1;
b = 1'b0;
#50;
a = 1'b1;
b = 1'b1;
#50;
end
endmodule
```


rishavproject.v
rishav.wcfg
Name
Value
y[2:0]
110
a
1
b
1
810.000 ns
0 ns
200 ns
400 ns
600 ns
800 ns
001
011
010
110

Experiment 12

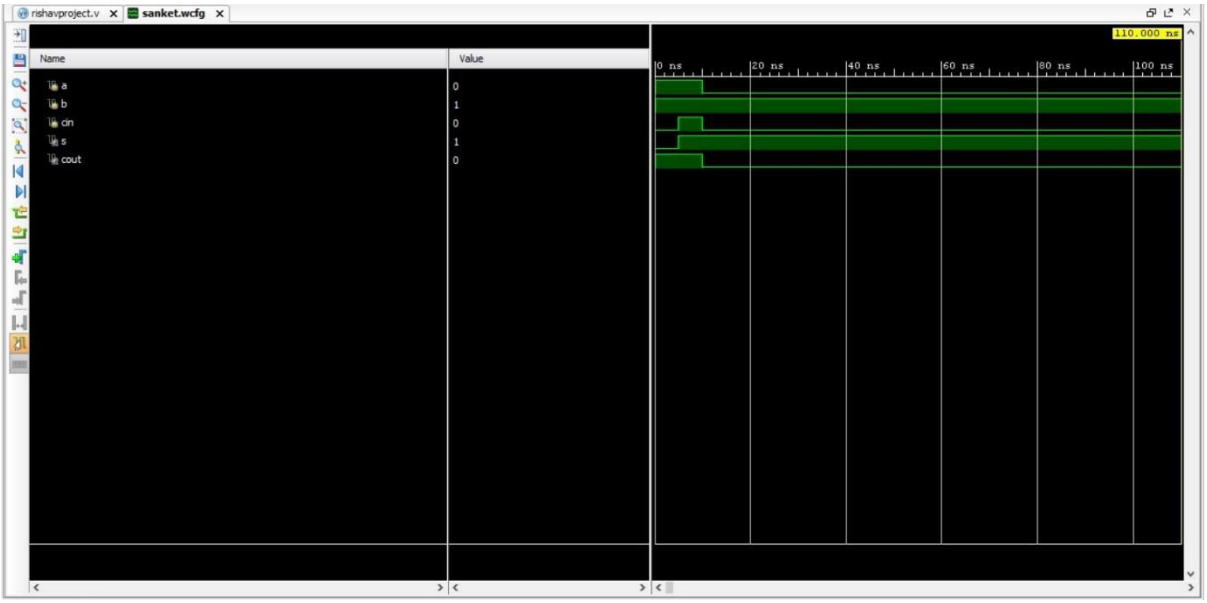
Write HDL implementation for a 3-bit full adder using data flow model. Simulate the same using data flow model and depict the timing diagram for valid inputs.

MAIN MODULE

```
module fa(a,b,cin,s,cout);
   input a,b,cin;
   output s,cout;
   assign s =a^b^cin;
   assign cout = (a&b) | (b&cin) | (cin&a);
endmodule
```

TEST MODULE

```
module fa_test;
   reg a,b,cin;
   wire s, cout;
   fa f1(a,b,cin,s,cout);
   initial
      begin
         a=1;     b=1; cin=0;
         #5
         a=1;     b=1; cin=1;
         #5
         a=0;     b=1; cin=0;
         #100 $finish;
      end
endmodule
```



Signature of the staff in-charge