

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



VERILOG LABORATORY REPORT

Submitted by

HARSHITHA R(1BM21CS075)

KANJIKA SINGH(1BM21CS086)

NEHA BHASKAR KAMATH(1BM21CS113)

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

Oct 2022-Feb 2023

**B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering**



CERTIFICATE

This is to certify that the Lab work entitled “**LOGIC DESIGN**” carried out by, **HARSHITHA R(1BM21CS075)**, **KANJIKA SINGH(1BM21CS086)** and **NEHA BHASKAR KAMATH(1BM21CS113)**, who are bonafide students of B. M. S. College of Engineering. It is in partial fulfilment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belgaum during the year 2022-23.

The Lab report has been approved as it satisfies the academic requirements in respect of **LOGIC DESIGN- (22CS3PCLOD)** work prescribed for the said degree.

| Marks | |
|----------------|----------|
| Max. Marks | Obtained |
| 10 | |
| Marks in Words | |
| | |

Signature of the staff in-charge

Head of the Department

Date:

TABLE OF CONTENTS

| Sl.No | Experiment title | Pg.No |
|-------|--|-------|
| | CYCLE 1: STRUCTURAL MODELLING | |
| 1 | Write HDL implementation for the following Logic AND/OR/NOT Simulate the same using structural model and depict the timing diagram for valid inputs. | 5-6 |
| 2 | Write HDL implementation for the following Logic NAND/NOR Simulate the same using structural model and depict the timing diagram for valid inputs. | 7 |
| 3 | Write HDL implementation for the following Boolean expression: $A'C' + A'C + AB'$ Simulate the same using structural model and depict the timing diagram for valid inputs. | 8 |
| 4 | Write HDL implementation for a 8:1 Multiplexer. Simulate the same using structural model and depict the timing diagram for valid inputs. | 9-10 |
| 5 | Write HDL implementation for a 2-to-4 decoder. Simulate the same using structural model and depict the timing diagram for valid inputs. | 11-12 |
| 6 | Write HDL implementation for a 4-to-2 encoder. Simulate the same using structural model and depict the timing diagram for valid inputs. | 13-14 |
| | CYCLE 2: BEHAVIOR MODELLING | |
| 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. | 15-16 |
| 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. | 17-18 |
| 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. | 19 |

| | | |
|----|--|-------|
| 10 | Write HDL implementation for a 3-bit down-counter using behavioral model. Simulate the same using Behavior model and depict the timing diagram for valid inputs. | 20 |
| | CYCLE 3: DATAFLOW MODELLING | |
| 11 | Write HDL implementation for NAND/NOR/EXOR gates using data flow model. Simulate the same using Dataflow model and depict the timing diagram for valid inputs. | 21-22 |
| 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. | 23-24 |

CYCLE 1: STRUCTURAL MODELLING

EXPERIMENT 1

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

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

PROGRAM

```
module AND_OR_NOT( output and1,or1,not1,input A, B);  
and(and1, A, B);  
or(or1,A,B);  
not(not1,B);  
endmodule
```

```
module testbench;  
reg A,B;  
wire and1,or1,not1;  
AND_OR_NOT g(and1,or1,not1,A,B);  
initial  
begin  
$dumpfile("gates.vcd");  
$dumpvars(0,testbench);  
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

COMPILATION, EXECUTION AND RESULT OF SIMULATION

```
Command Prompt - gtkwave
Microsoft Windows [Version 10.0.22621.1105]
(c) Microsoft Corporation. All rights reserved.

C:\Users\harsh>cd C:\iverilog\bin

C:\iverilog\bin>iverilog AND_OR_NOT.V

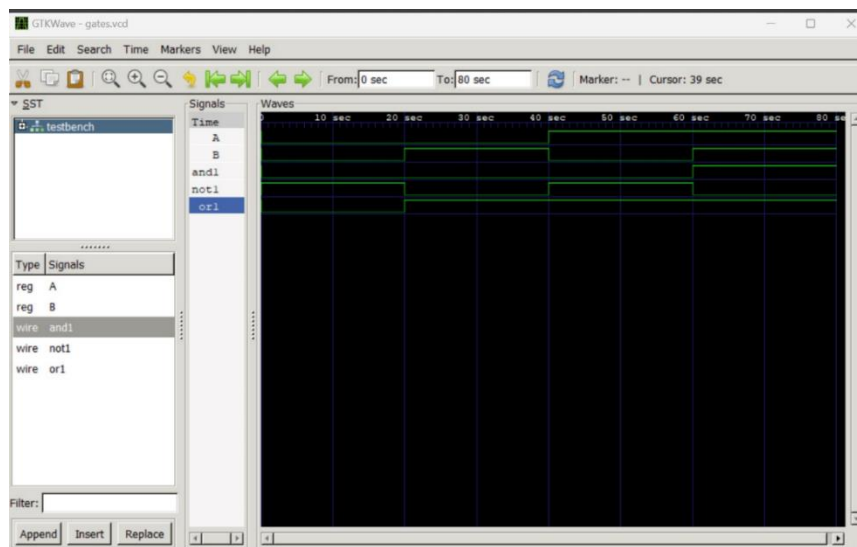
C:\iverilog\bin>iverilog -o AND_OR_NOT AND_OR_NOT.v

C:\iverilog\bin>vvp AND_OR_NOT
VCD info: dumpfile gates.vcd opened for output.

C:\iverilog\bin>gtkwave gates.vcd

GTKWave Analyzer v3.3.48 (w)1999-2013 BSI

[0] start time.
[80] end time.
```



EXPERIMENT 2

Write HDL implementation for the following Logic NAND/NOR.

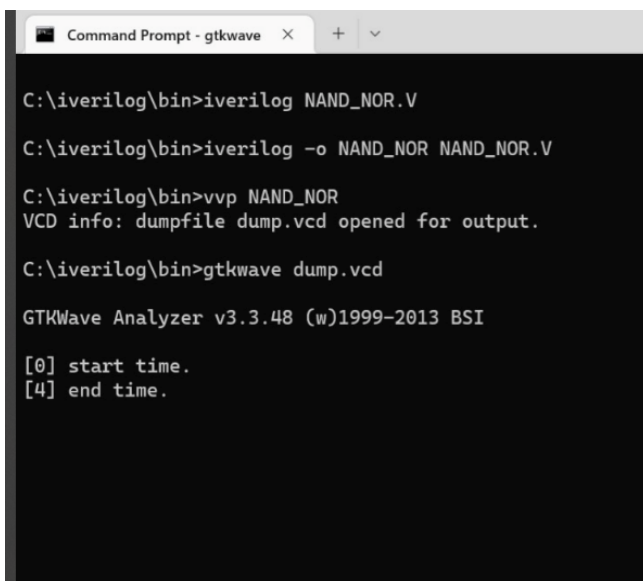
Simulate the same using structural model and depict the timing diagram for valid inputs.

PROGRAM

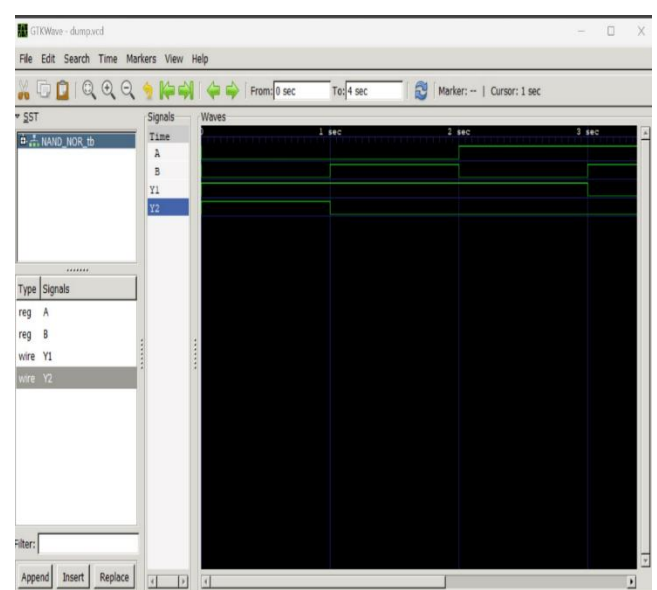
```
module NAND_NOR(output Y1,Y2, input A, B); wire Yd1,Yd2;
and(Yd1, A, B);
not(Y1, Yd1);
or(Yd2,A,B);
not(Y2,Yd2); endmodule

module NAND_NOR_tb; reg A, B;
wire Y;
NAND_NOR Indtance0 (Y1,Y2, A, B);
initial begin
A = 1'b0; B = 1'b0;
#1 A = 1'b0; B = 1'b1; #1 A = 1'b1; B = 1'b0; #1 A = 1'b1; B = 1'b1;
#1 $finish; end
initial begin
$dumpfile("dump.vcd");
$dumpvars(); end endmodule
```

COMPILATION, EXECUTION AND RESULT OF SIMULATION



```
Command Prompt - gtkwave
C:\iverilog\bin>iverilog NAND_NOR.V
C:\iverilog\bin>iverilog -o NAND_NOR NAND_NOR.V
C:\iverilog\bin>vvp NAND_NOR
VCD info: dumpfile dump.vcd opened for output.
C:\iverilog\bin>gtkwave dump.vcd
GTKWave Analyzer v3.3.48 (w)1999-2013 BSI
[0] start time.
[4] end time.
```



EXPERIMENT 3

Write HDL implementation for the following Boolean expression:

$$A'C' + A'C + AB'$$

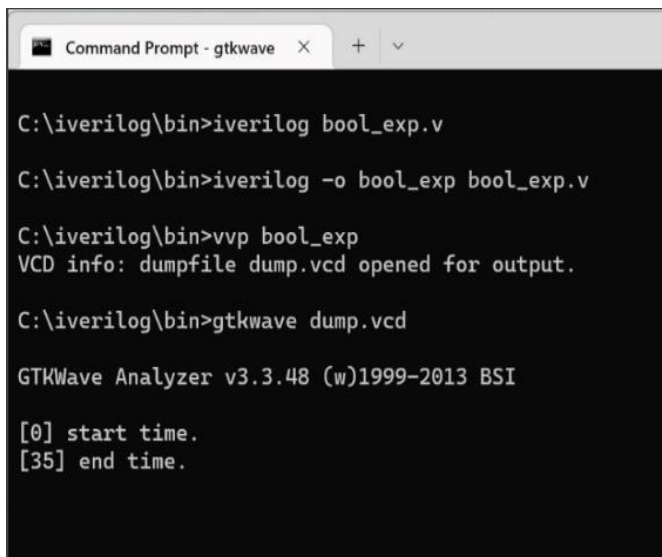
Simulate the same using structural model and depict the timing diagram for valid inputs.

PROGRAM

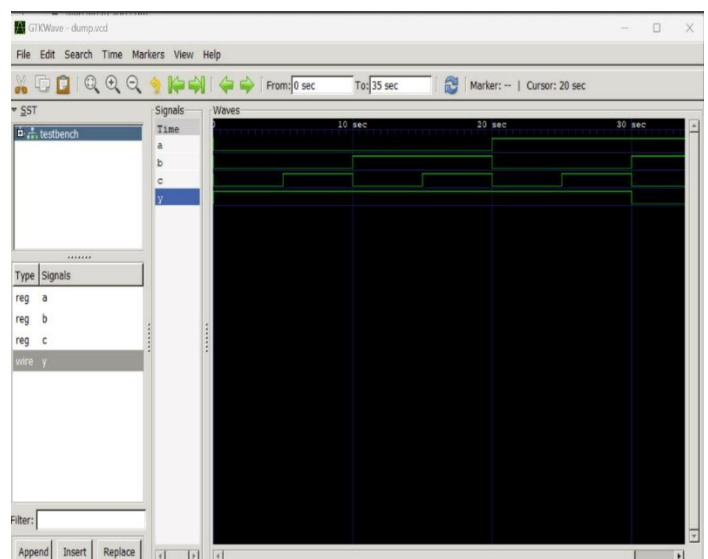
```
module bool_exp(a,b,c,y); input a,b,c;
output y;
wire and_op1, and_op2, and_op3, and_op4, and_op5; and g1(and_op1,~a,~c);
and g2(and_op2,~a,c); and g3(and_op3,a,~b);
or g4(and_op4, and_op1, and_op2); or g5(and_op5, and_op3, and_op4); or g6(y, and_op5,
and_op4);
endmodule

module testbench; reg a,b,c,d; wire y;
bool_exp ao(a,b,c,y); initial begin
$dumpfile("dump.vcd");
$dumppvars(0,testbench);
a=0;b=0;c=0; #5 a=0;b=0;c=1; #5 a=0;b=1;c=0; #5 a=0;b=1;c=1; #5 a=1;b=0;c=0; #5
a=1;b=0;c=1; #5 a=1;b=1;c=0; #5 a=1;b=1;c=1;
$finish;
end endmodule
```

COMPILATION, EXECUTION AND RESULT OF SIMULATION



```
Command Prompt - gtkwave
C:\iverilog\bin>iverilog bool_exp.v
C:\iverilog\bin>iverilog -o bool_exp bool_exp.v
C:\iverilog\bin>vvp bool_exp
VCD info: dumpfile dump.vcd opened for output.
C:\iverilog\bin>gtkwave dump.vcd
GTKWave Analyzer v3.3.48 (w)1999-2013 BSI
[0] start time.
[35] end time.
```



EXPERIMENT 4

Write HDL implementation for a 8:1 Multiplexer.

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

PROGRAM

```
module and_gate(output a, input b, c, d, e); assign a = b & c & d & e;
endmodule
module not_gate(output f, input g); assign f = ~ g;
endmodule
module or_gate(output l, input m, n, o, p, q, r, s, t); assign l = m | n | o | p | q | r | s | t;
endmodule
module m81(out, D0, D1, D2, D3, D4, D5, D6, D7, S0, S1, S2);
output out;
input D0, D1, D2, D3, D4, D5, D6, D7, S0, S1, S2; wire s0bar, s1bar, T1, T2, T3, T4, T5, T6,
T7, T8;
not_gate u1(s1bar, S1); not_gate u2(s0bar, S0); not_gate u3(s2bar, S2);
and_gate u4(T1, D0, s0bar, s1bar, s2bar); and_gate u5(T2, D1, S0, s1bar, s2bar); and_gate
u6(T3, D2, s0bar, S1, s2bar); and_gate u7(T4, D3, S0, S1, s2bar); and_gate u8(T5, D4, s0bar,
s1bar, S2); and_gate u9(T6, D5, S0, s1bar, S2); and_gate u10(T7, D6, s0bar, S1, S2); and_gate
u11(T8, D7, S0, S1, S2);
or_gate u12(out, T1, T2, T3, T4, T5, T6, T7, T8);
endmodule

//timescale 1ns/1ps
module top;
wire out;
reg D0, D1, D2, D3, D4, D5, D6, D7, D8, S0, S1, S2;
m81 name(.D0(D0), .D1(D1), .D2(D2), .D3(D3), .D4(D4), .D5(D5), .D6(D6), .D7(D7),
.S0(S0), .S1(S1), .S2(S2), .out(out));
initial begin
$dumpfile("dump.vcd");
$dumpvars(0,top);
D0=1'b0; D1=1'b0; D2=1'b0; D3=1'b0; D4=1'b0; D5=1'b0; D6=1'b0; D7=1'b0;S0=1'b0;
S1=1'b0; S2=1'b0;
#500 $finish; end
always #1 D0=~D0;
always #2 D1=~D1;
always #3 D2=~D2;
always #4 D3=~D3;
always #5 D4=~D4;
always #6 D5=~D5;
always #7 D6=~D6;
always #8 D7=~D7;
always #9 S0=~S0;
always #10 S1=~S1;
always #11 S2=~S2;
always@(D0 or D1 or D2 or D3 or D4 or D5 or D6 or D7 or S0 or S1 or S2);
```

endmodule

COMPILATION, EXECUTION AND RESULT OF SIMULATION

```
Command Prompt - gtkwave X + v

C:\iverilog\bin>iverilog and_gate.v

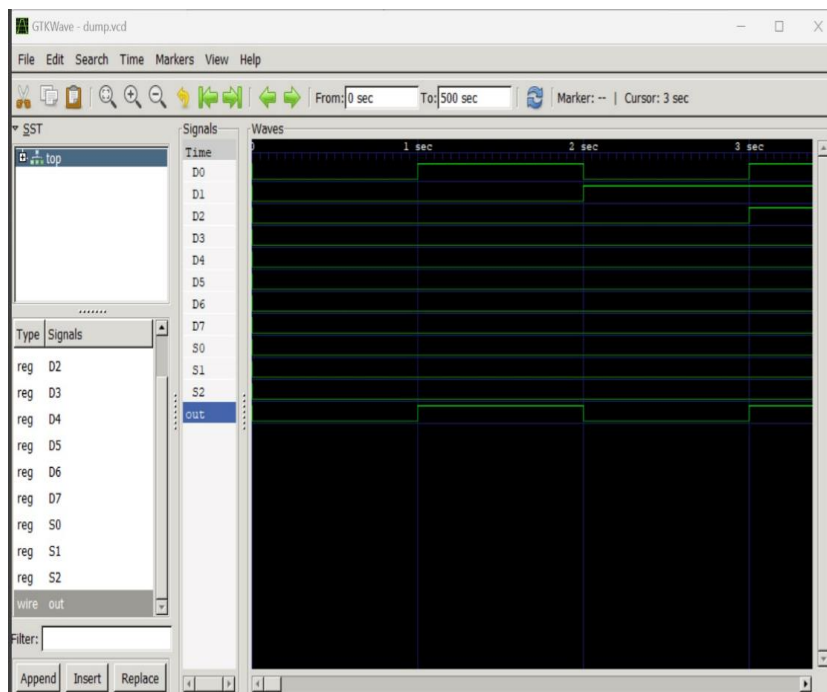
C:\iverilog\bin>iverilog -o and_gate and_gate.v

C:\iverilog\bin>vvp and_gate
VCD info: dumpfile dump.vcd opened for output.

C:\iverilog\bin>gtkwave dump.vcd

GTKWave Analyzer v3.3.48 (w)1999-2013 BSI

[0] start time.
[500] end time.
```



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.

PROGRAM

```
module bcddecoder2to4(b0, b1, d0, d1, d2, d3); input b0, b1;
output d0, d1, d2, d3; wire t0, t1;
not n1(t0, b0);
not n2(t1, b1);
and a0(d0, t0, t1);
and a1(d1, t0, b1);
and a2(d2, b0, t1);
and a3(d3, b0, b1);
endmodule
```

```
module test;
reg b0, b1; wire d0, d1, d2, d3;
bcddecoder2to4 bcdg(b0, b1, d0, d1, d2, d3); initial
begin
$dumpfile("bcd.vcd");
$dumpvars(0, test); b0 = 0; b1 = 0;
#40
b0 = 0; b1 = 1;
#40
b0 = 1; b1 = 0;
#40
b0 = 1; b1 = 1;
#40
$finish;
end endmodule
```

COMPILATION, EXECUTION AND RESULT OF SIMULATION

```
Command Prompt - gtkwave  x + v

C:\iverilog\bin>iverilog bcddecoder2to4.v

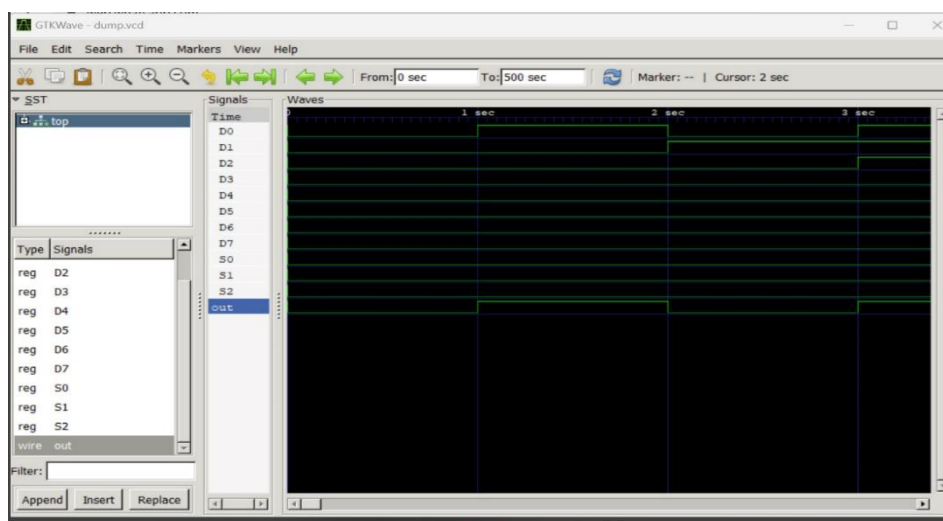
C:\iverilog\bin>iverilog -o bcddecoder2to4 bcddecoder2to4.v

C:\iverilog\bin>vvp bcddecoder2to4
VCD info: dumpfile bcd.vcd opened for output.

C:\iverilog\bin>gtkwave dump.vcd

GTKWave Analyzer v3.3.48 (w)1999-2013 BSI

[0] start time.
[500] end time.
```



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.

PROGRAM

```
module encoder4to2(b0, b1, d0, d1, d2, d3); output b0,b1;
input d0, d1, d2, d3;
wire t0, t1, t2, t3, t4, t5, t6, t7; not n0(t0, d0);
not n1(t1, d1);
not n2(t2, d2);
not n3(t3, d3);
and a0(t4, t0, t1, d2, t3);
and a1(t5, t0, t1, t2, d3);
and a2(t6, t0, d1, t2, t3);
and a3(t7, t0, t1, t2, d3);
or o0(b0, t4, t5);
or o1(b1, t6, t7);
endmodule
```

```
module test; wire b0, b1;
reg d0, d1, d2, d3;
encoder4to2 encg(b0, b1, d0, d1, d2, d3); initial
begin
$dumpfile("enc4to2.vcd");
$dumpvars(0, test);
d0 = 1; d1 = 0; d2 = 0; d3 = 0;
#40
d0 = 0; d1 = 1; d2 = 0; d3 = 0;
#40
d0 = 0; d1 = 0; d2 = 1; d3 = 0;
#40
d0 = 0; d1 = 0; d2 = 0; d3 = 1;
#40
$finish; end
Endmodule
```

COMPILATION, EXECUTION AND RESULT OF SIMULATION

```
Command Prompt - gtkwave  X + v

C:\iverilog\bin>iverilog encoder4to2.v

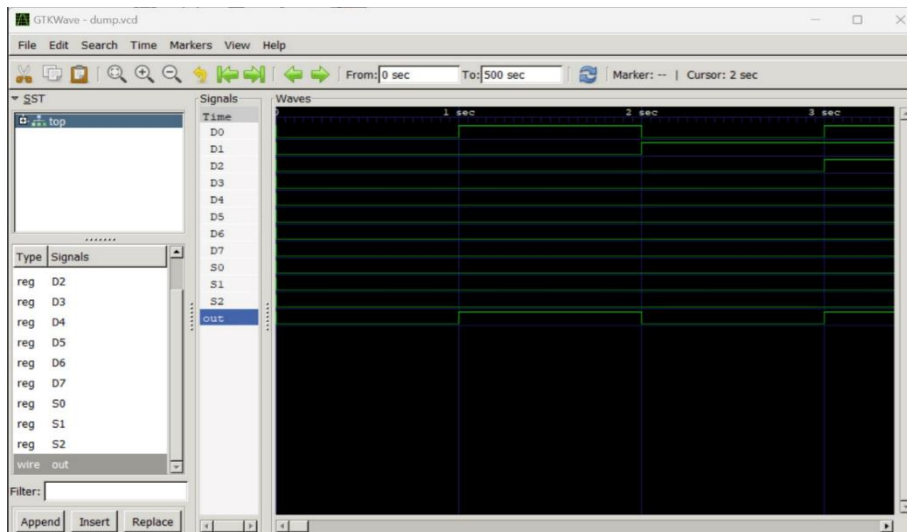
C:\iverilog\bin>iverilog -o encoder4to2 encoder4to2.v

C:\iverilog\bin>vvp encoder4to2
VCD info: dumpfile enc4to2.vcd opened for output.

C:\iverilog\bin>gtkwave dump.vcd

GTKWave Analyzer v3.3.48 (w)1999-2013 BSI

[0] start time.
[500] end time.
```



CYCLE 2: BEHAVIOR MODELLING

EXPERIMENT 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

PROGRAM

```
module SRFF(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

module test_srflipf;
reg[1:0]A; reg c; wire x, xb;
SRFF srff(A, c, x, xb);
initial c=1'b0; always #5 c=~c; initial
begin
    $dumpfile("sr.vcd");
    $dumpvars(0, test_srflipf);
    A=2'b00;#10;
    A=2'b01;#10;
    A=2'b10;#10;
    A=2'b11;
    #20 $finish;
end endmodule
```

COMPILATION, EXECUTION AND RESULT OF SIMULATION

```
Command Prompt - gtkwave X + v

C:\iverilog\bin>iverilog SRFF.v

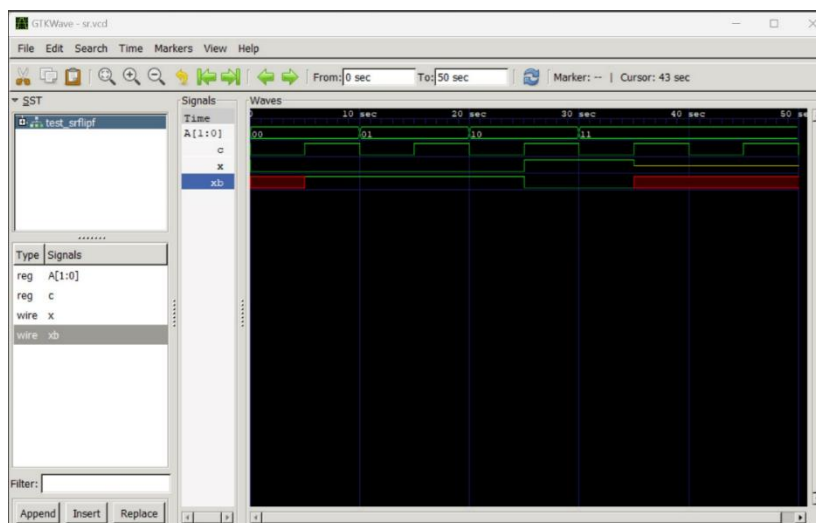
C:\iverilog\bin>iverilog -o SRFF SRFF.v

C:\iverilog\bin>vvp SRFF
VCD info: dumpfile sr.vcd opened for output.

C:\iverilog\bin>gtkwave sr.vcd

GTKWave Analyzer v3.3.48 (w)1999-2013 BSI

[0] start time.
[50] end time.
```



EXPERIMENT 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.

PROGRAM

```
module JKFF(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
```

```
module test_jkflipf; reg[1:0]A;
reg c; wire x, xb;
JKFF jkff(A, c, x, xb); initial c=1'b0; always #5 c=~c; initial
begin
    $dumpfile("jk.vcd");
    $dumpvars(0,test_jkflipf);
    A=2'b00;#10;
    A=2'b01;#10;
    A=2'b10;#10;
    A=2'b11;
    #20 $finish;
end endmodule
```

COMPILATION, EXECUTION AND RESULT OF SIMULATION

```
Command Prompt - gtkwave X + v

C:\iverilog\bin>iverilog JKFF.v

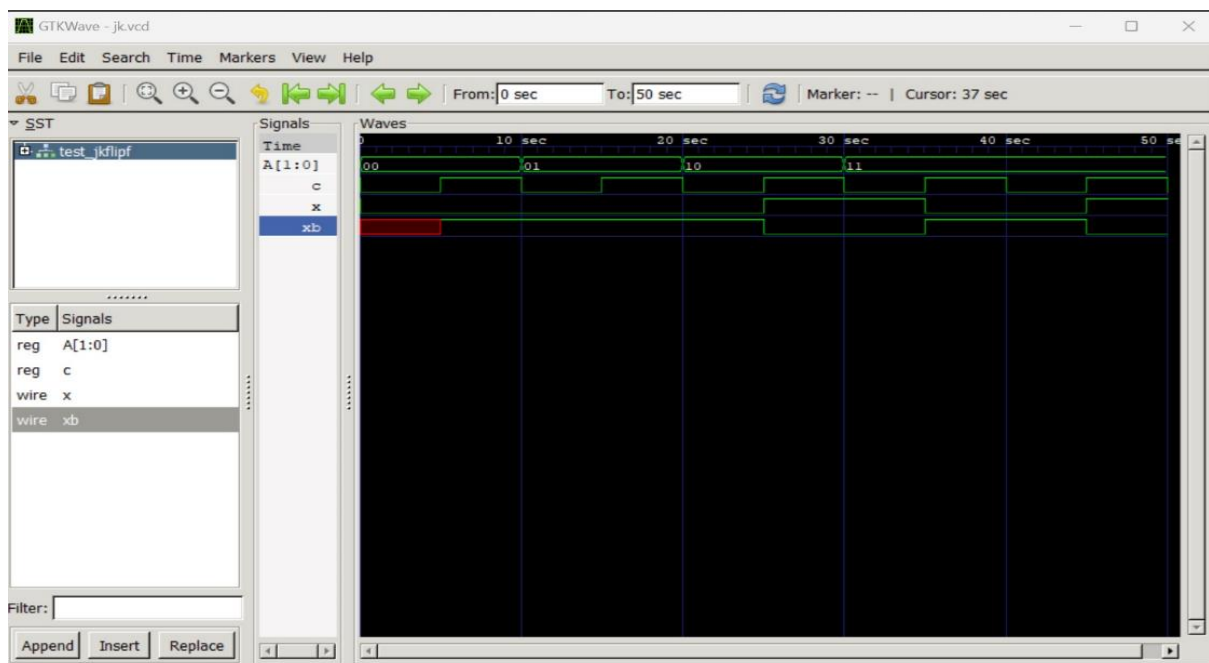
C:\iverilog\bin>iverilog -o JKFF JKFF.v

C:\iverilog\bin>vvp JKFF
VCD info: dumpfile jk.vcd opened for output.

C:\iverilog\bin>gtkwave jk.vcd

GTKWave Analyzer v3.3.48 (w)1999-2013 BSI

[0] start time.
[50] end time.
```



EXPERIMENT 9

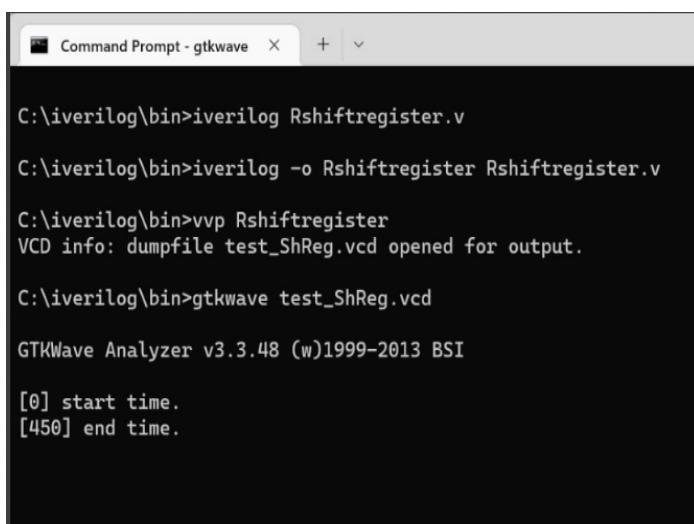
Write HDL implementation for a 4-bit right shift register using behavioral model.
Simulate the same using Behavior model and depict the timing diagram for valid inputs.

PROGRAM

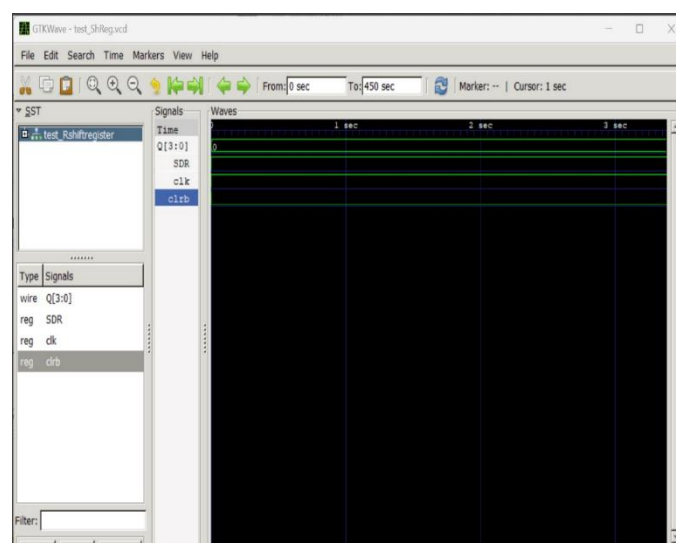
```
module Rshiftregister(input clk, input clrb, input SDR, output reg[3:0]Q);
always@(posedge(clk), negedge(clrb))
if(~clrb) Q<=4'b0000;
else
Q<={SDR, Q[3:1]};
endmodule
```

```
module test_Rshiftregister; reg clk, clrb, SDR; wire[3:0]Q;
Rshiftregister RS(clk, clrb, SDR, Q); initial clk = 0;
always #50 clk= ~clk; initial
begin
$dumpfile("test_ShReg.vcd");
$dumpvars(0,test_Rshiftregister); clk=1;
clrb=0; SDR=1; #100;
clrb=1; SDR=1; #150; SDR=0;
#200 $finish;
end endmodule
```

COMPILATION, EXECUTION AND RESULT OF SIMULATION



```
Command Prompt - gtwave
C:\iverilog\bin>iverilog Rshiftregister.v
C:\iverilog\bin>iverilog -o Rshiftregister Rshiftregister.v
C:\iverilog\bin>vvp Rshiftregister
VCD info: dumpfile test_ShReg.vcd opened for output.
C:\iverilog\bin>gtkwave test_ShReg.vcd
GTKWave Analyzer v3.3.48 (w)1999-2013 BSI
[0] start time.
[450] end time.
```



EXPERIMENT 10

Write HDL implementation for a 3-bit down-counter using behavioral model.

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

PROGRAM

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

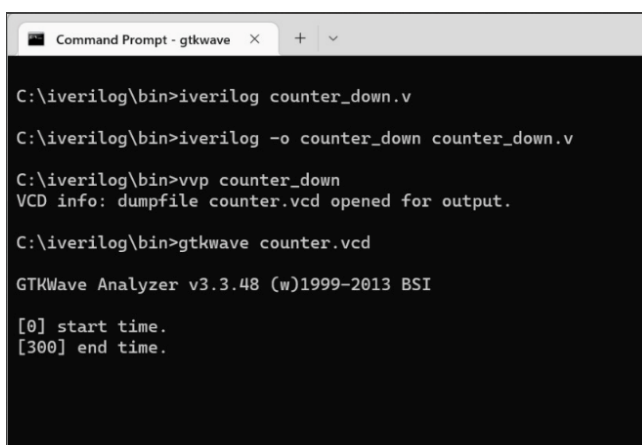
```
else
```

```
count<=count-1;
endmodule
```

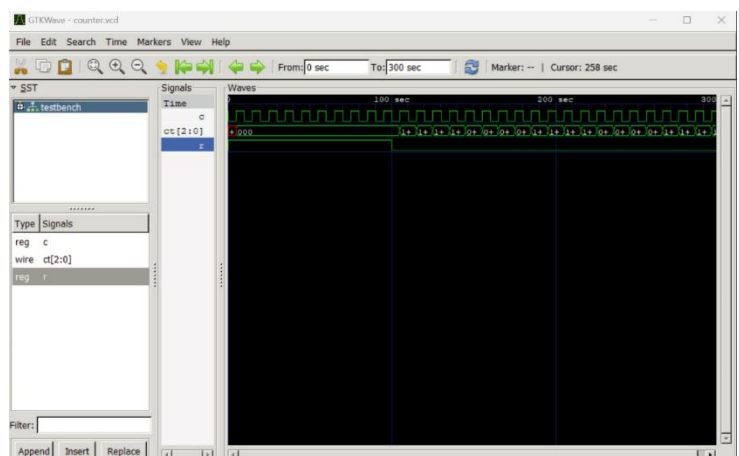
```
module testbench; reg r, c;
```

```
wire [2:0] ct;
counter_down countd(ct,r,c);
initial begin
$dumpfile("counter.vcd");
$dumpvars(0,testbench); r=1;
c=8; #100; r=0; #200
$finish;
end
always #5 c=~c;
endmodule
```

COMPILATION, EXECUTION AND RESULT OF SIMULATION



```
Command Prompt - gtkwave
C:\iverilog\bin>iverilog counter_down.v
C:\iverilog\bin>iverilog -o counter_down counter_down.v
C:\iverilog\bin>vvp counter_down
VCD info: dumpfile counter.vcd opened for output.
C:\iverilog\bin>gtkwave counter.vcd
GTKWave Analyzer v3.3.48 (w)1999-2013 BSI
[0] start time.
[300] end time.
```



CYCLE 3: DATAFLOW MODELLING

EXPERIMENT 11

Write HDL implementation for NAND/NOR/EXOR gates using data flow model.

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

PROGRAM

```
module NAND_NOR_EXOR (output nand1,nor1,exor1, input A, B);

assign nand1= ~(A & B);
assign nor1=~(A+B);
assign exor1 = A ^ B;
endmodule

module NAND_NOR_EXOR_tb;

reg A, B;

wire nand1,nor1,exor1;

NAND_NOR_EXOR Indtance (nand1,nor1,exor1, A, B);
initial begin
A = 1'b0; B = 1'b0; #1 A = 1'b0; B = 1'b1; #1 A = 1'b1; B = 1'b0; #1 A = 1'b1; B = 1'b1;
#1 $finish; end
initial begin
$dumpfile("dump.vcd");
$dumpvars();
end
endmodule
```

COMPILATION, EXECUTION AND RESULT OF SIMULATION

```
Command Prompt - gtkwave  x  +  v

C:\iverilog\bin>iverilog NAND_NOR_EXOR.v

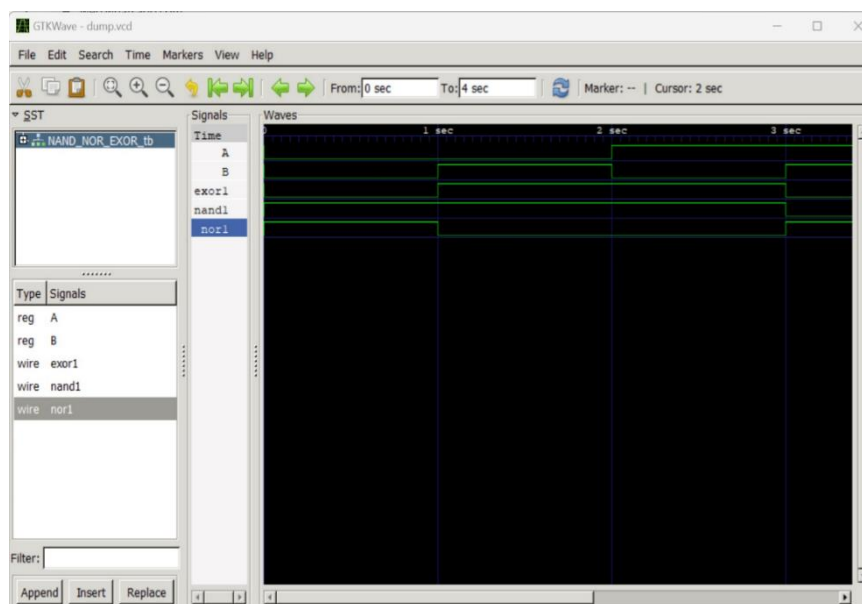
C:\iverilog\bin>iverilog -o NAND_NOR_EXOR NAND_NOR_EXOR.v

C:\iverilog\bin>vvp NAND_NOR_EXOR
VCD info: dumpfile dump.vcd opened for output.

C:\iverilog\bin>gtkwave dump.vcd

GTKWave Analyzer v3.3.48 (w)1999-2013 BSI

[0] start time.
[4] end time.
```



EXPERIMENT 12

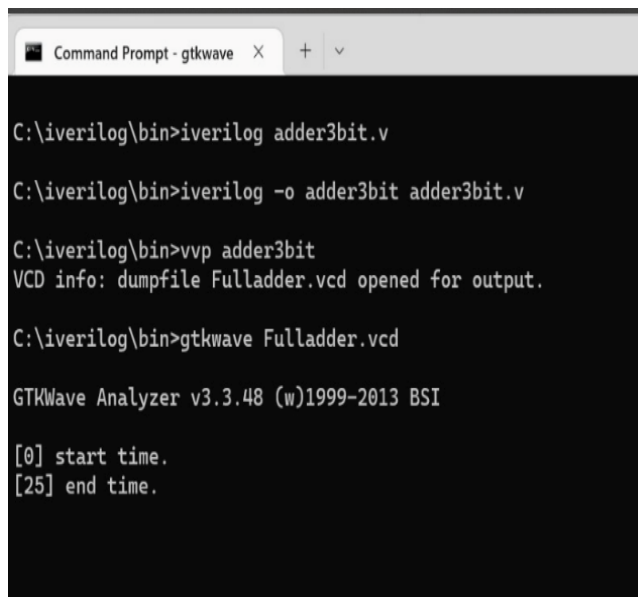
Write HDL implementation for a 3-bit full adder using data flow model.
Simulate the same using Dataflow model and depict the timing diagram for valid inputs.

PROGRAM

```
module adder3bit(input a, input b, input cin, output s, output cout);
assign s = a^b^cin;
assign cout = (a & b) | (a & cin) | (b & cin);
endmodule

module test;
reg a, b, cin; wire s, cout;
adder3bit Full(a,b,cin,s,cout);
initial
begin
$dumpfile("Fulladder.vcd");
$dumpvars(0,test); a=1;
b=1;
cin =0; #5 a=1;
b=1;
cin = 1; #5 a=0;
b=1;
cin = 0;#5
#10 $finish;
end
endmodule
```

COMPILATION, EXECUTION AND RESULT OF SIMULATION



```
Command Prompt - gtkwave X + v

C:\iverilog\bin>iverilog adder3bit.v

C:\iverilog\bin>iverilog -o adder3bit adder3bit.v

C:\iverilog\bin>vvp adder3bit
VCD info: dumpfile Fulladder.vcd opened for output.

C:\iverilog\bin>gtkwave Fulladder.vcd

GTKWave Analyzer v3.3.48 (w)1999-2013 BSI

[0] start time.
[25] end time.
```

