DIGITAL CIRCUITS DESIGN USING VERILOG HDL

Table of Contents:

	❖ 4x1 Multiplexer • Behavioral model • Dataflow model • Structural model • 4x1 using 2x1 mux ❖ 8x1 Multiplexer • Behavioral model • Dataflow model • Structural model ❖ 1x2 Demultiplexer • Behavioral model • Dataflow model • Structural model ❖ 1x4 Demultiplexer • Behavioral model • Dataflow model • Structural model ❖ 1x8 Demultiplexer • Behavioral model • Dataflow model • Structural model	
5	**Encoder and Decoder** ❖ 2x4 Encoder • Behavioral model • Dataflow model • Structural model ❖ 2x4 Priority Encoder • Behavioral model • Dataflow model • Structural model ❖ 3x8 Encoder • Behavioral model • Dataflow model • Structural model ❖ 4x2 Decoder • Behavioral model • Dataflow model • Structural model ❖ 8x3 Decoder • Behavioral model • Dataflow model • Structural model	39 to 53

EXPERIMENT-1
REALISATION OF LOGIC GATES
(AND, OR, NOT, NAND, NOR, XOR, XNOR)

Aim: Design, simulate and implement all logic gates using Xilinx ISE.
Software Used: Xilinx **ISE-14.2**
Simulator Used: iSim
Synthesizer Used: XST
Procedure:

- Check the syntax of the program for any errors if any correct and verify again.
- In the process window, put the simulation mode in behavioral model, take a Verilog text fixture and give the input combinations.
- Perform simulation to verify the functionality and logic of the code.
- In the process window, change it to implement design highlight program in the source window, performs the run operations to implement the design.
- To view RTL and Technology schematic.

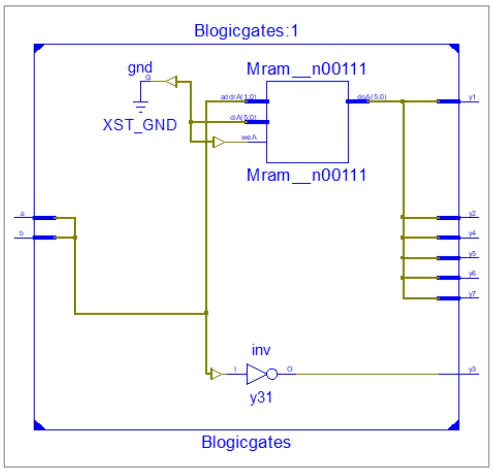
Verilog reports:

Behavioral Modelling

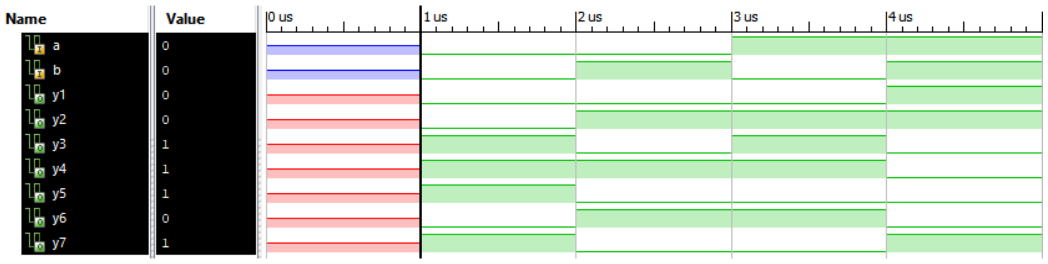
Verilog Program:

```
module Blogicgates(y1,y2,y3,y4,y5,y6,y7,a,b
    );
input a,b;
output y1,y2,y3,y4,y5,y6,y7;
reg y1,y2,y3,y4,y5,y6,y7;
always@(a,b)
begin
case({a,b})
2'b00: {y1,y2,y3,y4,y5,y6,y7}=7'b0011101;
2'b01: {y1,y2,y3,y4,y5,y6,y7}=7'b0101010;
2'b10: {y1,y2,y3,y4,y5,y6,y7}=7'b0111010;
2'b11: {y1,y2,y3,y4,y5,y6,y7}=7'b1100001;
default:{y1,y2,y3,y4,y5,y6,y7}=7'bxxxxxxx;
endcase
end
endmodule
```

RTL Schematic:



Simulation:

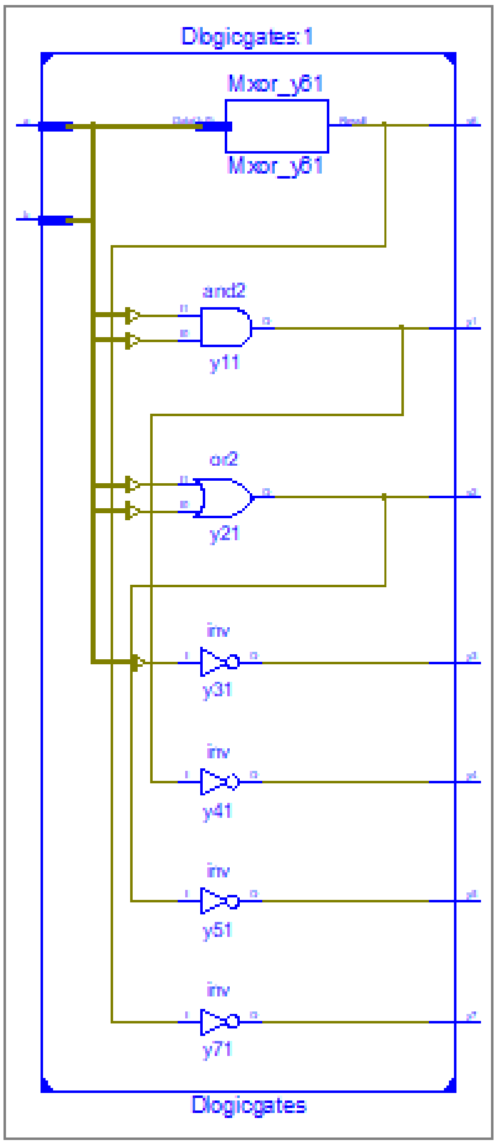


Dataflow Modelling

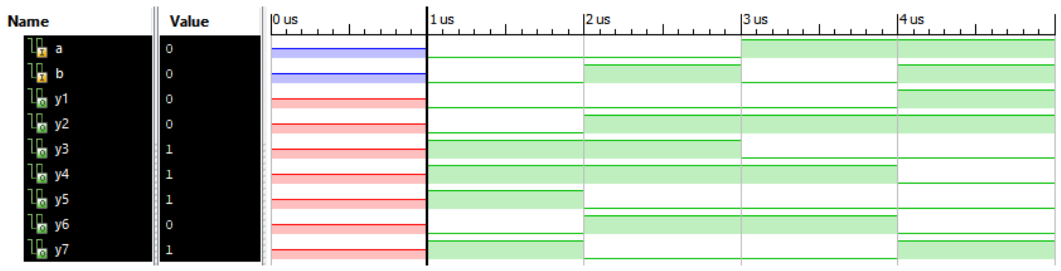
Verilog Program:

```
module
Dlogicgates(y1,y2,y3,y4,y5,y6,y7,a,b
    );
input a,b;
output y1,y2,y3,y4,y5,y6,y7;
wire y1,y2,y3,y4,y5,y6,y7;
assign y1=a&b;
assign y2=a|b;
assign y3=~a;
assign y4=~(a&b);
assign y5=~(a|b);
assign y6=a^b;
assign y7=~(a^b);
endmodule
```

RTL Schematic:



Simulation:



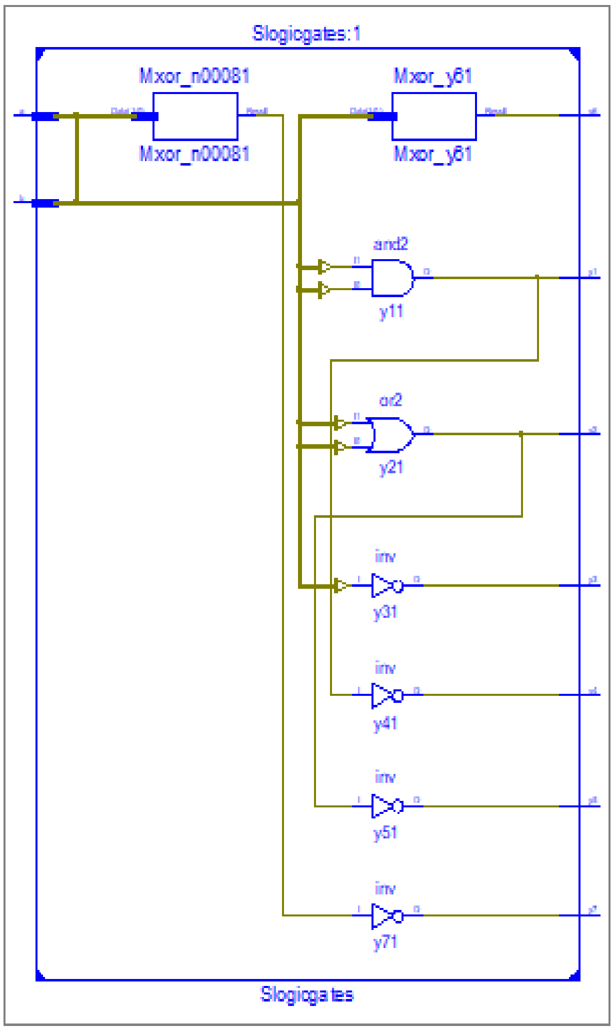
```
Name  Value
a     0
b     0
y1    0
y2    0
y3    1
y4    1
y5    1
y6    0
y7    1
```

Structural Modelling

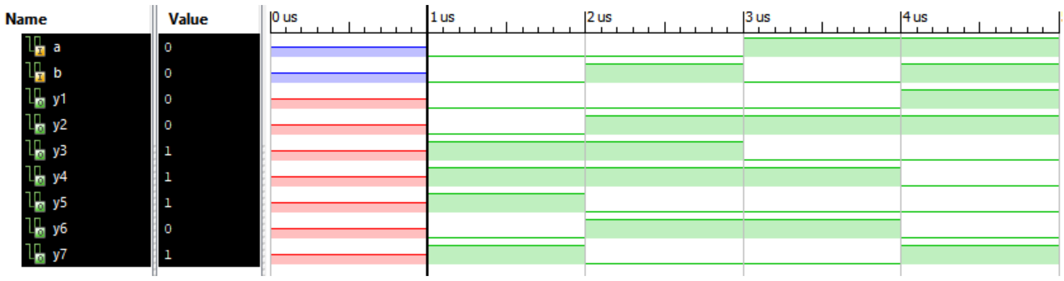
Verilog Program:

```
module
Slogicgates(y1,y2,y3,y4,y5,y6,y7,a,b
    );
input a,b;
output y1,y2,y3,y4,y5,y6,y7;
wire y1,y2,y3,y4,y5,y6,y7;
and  g1(y1,a,b);
or   g2(y2,a,b);
not  g3(y3,a);
nand g4(y4,a,b);
nor  g5(y5,a,b);
xor  g6(y6,a,b);
xnor g7(y7,a,b);
endmodule
```

RTL Schematic:



Simulation:



Result:

Designed and implemented all logic gates using Behavioral, dataflow, structural modelling using **Xilinx ISE 14.2**

EXPERIMENT-2
ADDER AND SUBTRACTOR

Aim: Design, simulate and implement adder and subtractor using Xilinx ISE.
Software Used: Xilinx **ISE-14.2**
Simulator Used: iSim
Synthesizer Used: XST
Procedure:

- Check the syntax of the program for any errors if any correct and verify again.
- In the process window, put the simulation mode in behavioral model, take a Verilog text fixture and give the input combinations.
- Perform simulation to verify the functionality and logic of the code.
- In the process window, change it to implement design highlight program in the source window, performs the run operations to implement the design.
- To view RTL and Technology schematic.

Verilog reports:

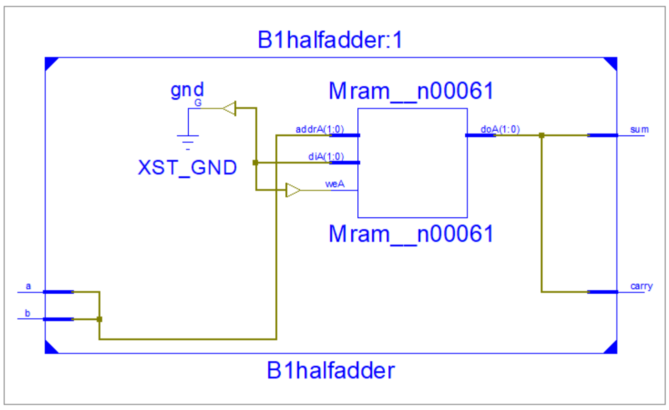
HALF ADDER

Behavioral Modelling

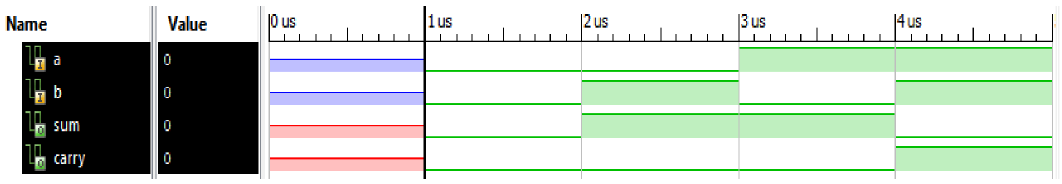
Verilog Program:

```
module
B1halfadder(a,b,sum,carry
   );
input a,b;
output sum,carry;
reg sum,carry;
always@(a,b)
begin
case({a,b})
2'b00:  {sum,carry}=2’b00;
2'b01:  {sum,carry}=2’b10;
2'b10:  {sum,carry}=2’b10;
2'b11:  {sum,carry}=2’b01;
default:{sum,carry}=2’bxx;
endcase
end
endmodule
```

RTL Schematic:



Simulation:

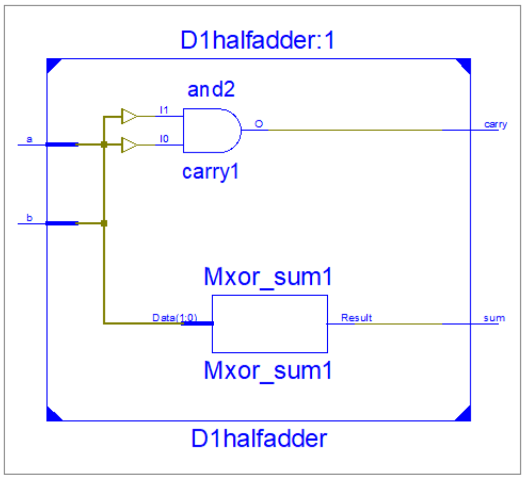


Dataflow Modelling

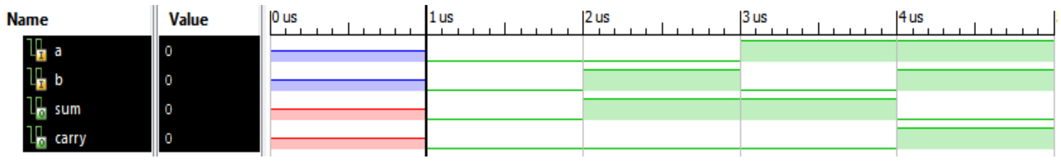
Verilog Program:

```
module D1halfadder(a,b,sum,carry
    );
input a,b;
output sum,carry;
wire sum,carry;
assign sum=a^b;
assign carry=a&b;
endmodule
```

RTL Schematic:



Simulation:

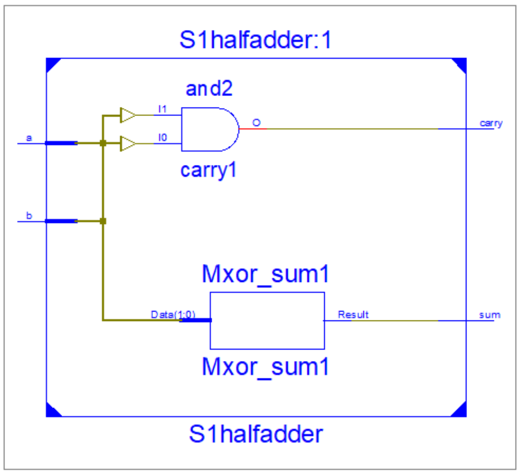


Structural Modelling

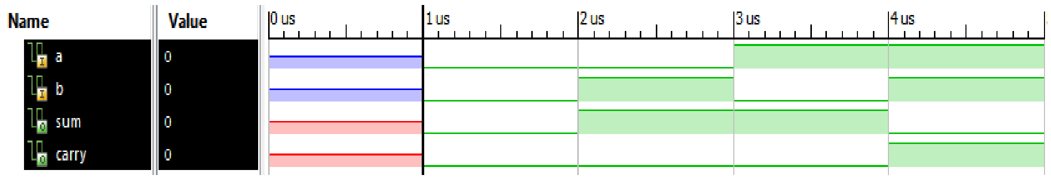
Verilog Program:

```
module S1halfadder(a,b,sum,carry
    );
input a,b;
output sum,carry;
wire sum,carry;
xor x1(sum,a,b);
and a1(carry,a,b);
endmodule
```

RTL Schematic:



Simulation:



Verilog reports:

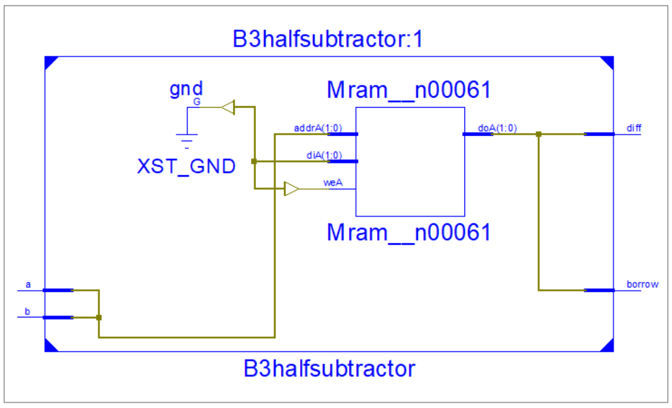
HALF SUBTRACTOR

Behavioral Modelling

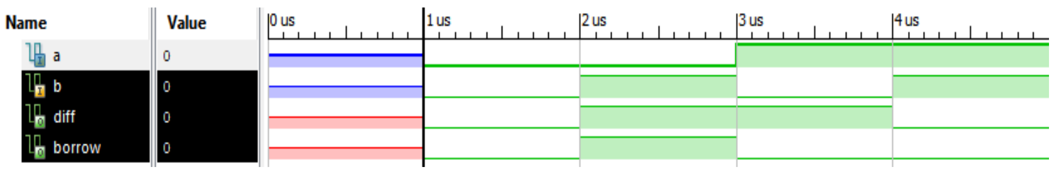
Verilog Program:

```
module
B3halfsubtractor(a,b,diff,borrow
    );
input a,b;
output diff,borrow;
reg diff,borrow;
always@(a,b)
begin
case({a,b})
2'b00:
 begin
diff=0;
borrow=0;
end
2'b01:
begin
diff=1;
borrow=1;
end
2'b10:
begin
diff=1;
borrow=0;
 end
2'b11:
begin
diff=0;
borrow=0;
end
default:
begin
diff=0;
borrow=0;
end
endcase
end
endmodule
```

RTL Schematic:



Simulation:

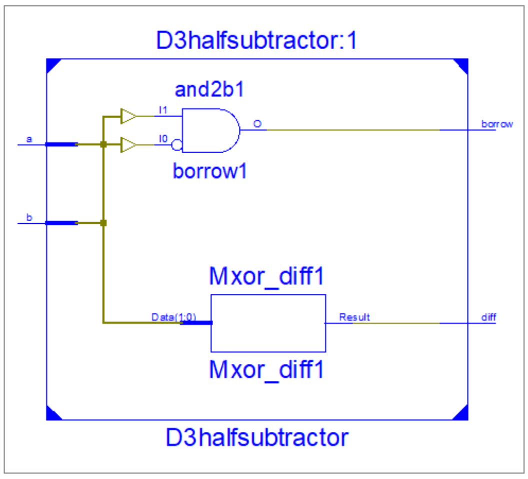


Dataflow Modelling

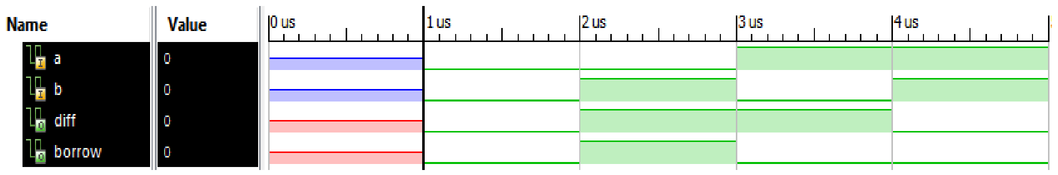
Verilog Program:

```
module D3halfsubtractor(a,b,diff,borrow
    );
input a,b;
output diff,borrow;
wire diff,borrow;
assign diff=a^b;
assign borrow=~a&b;
endmodule
```

RTL Schematic:



Simulation:

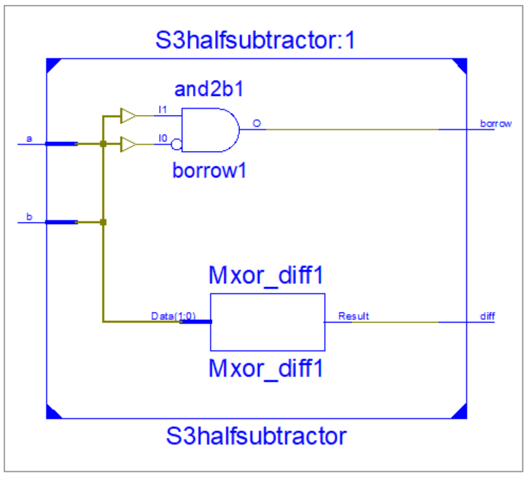


Structural Modelling

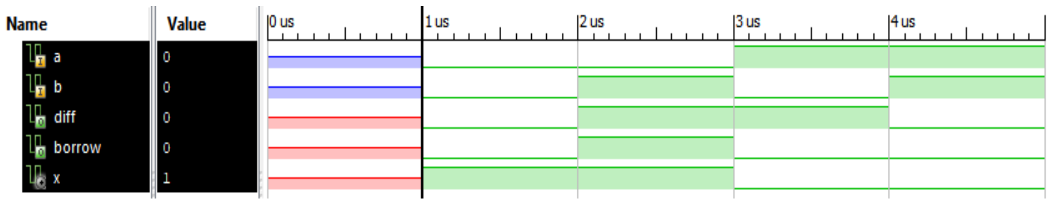
Verilog Program:

```
module S3halfsubtractor(a,b,diff,borrow
    );
input a,b;
output diff,borrow;
wire diff,borrow,x;
xor x1(diff,a,b);
not n1(x,a);
and a1(borrow,x,b);
endmodule
```

RTL Schematic:



Simulation:



Verilog reports:

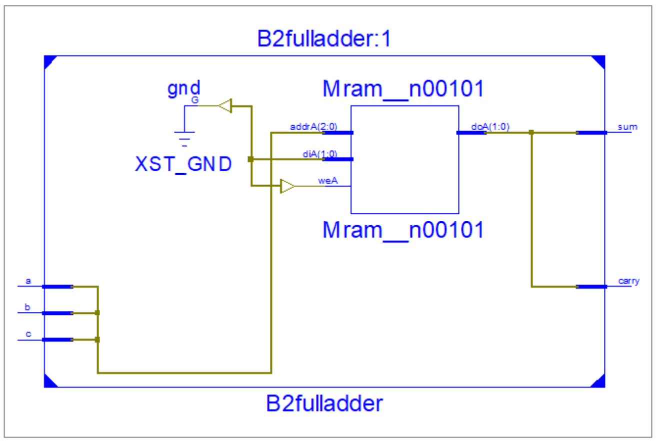
FULL ADDER

Behavioral Modelling

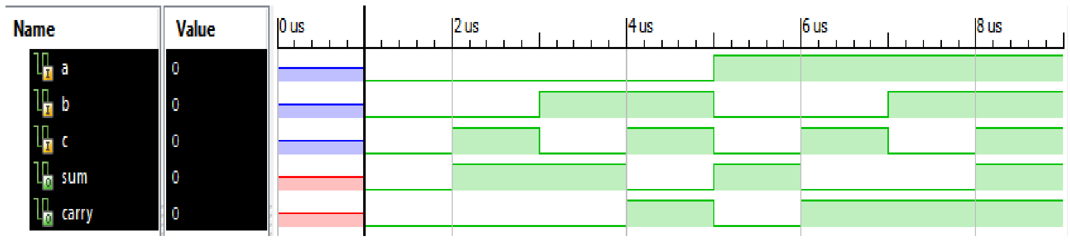
Verilog Program:

```
module
B2fulladder(a,b,c,sum,carry
    );
input a,b,c;
output sum,carry;
reg sum,carry;
always@(a,b,c)
begin
case({a,b,c})
3'b000:
begin sum=1'b0; carry=1'b0; end
3'b001:
 begin sum=1'b1; carry=1'b0;end
3'b010:
 begin sum=1'b1; carry=1'b0; end
3'b011:
 begin sum=1'b0; carry=1'b1;end
3'b100:
 begin sum=1'b1; carry=1'b0; end
3'b101:
 begin sum=1'b0; carry=1'b1; end
3'b110:
 begin sum=1'b0; carry=1'b1; end
3'b111:
 begin sum=1'b1; carry=1'b1; end
default:
begin sum=1'bx; carry=1'bx; end
endcase
end
endmodule
```

RTL Schematic:



Simulation:

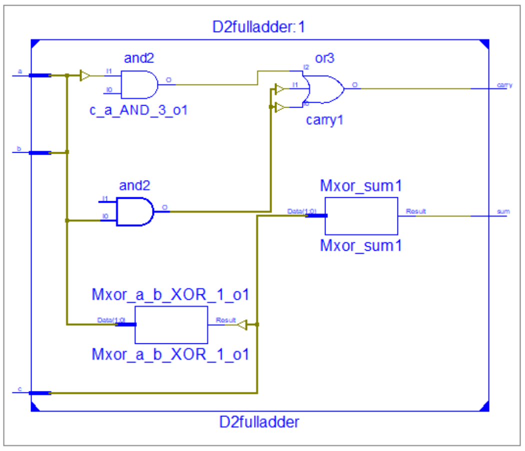


Dataflow Modelling

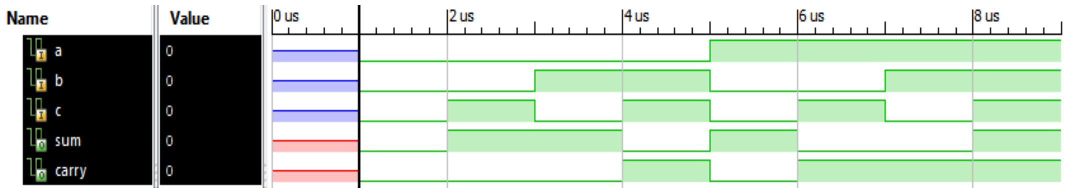
Verilog Program:

```
module D2fulladder(a,b,c,sum,carry
    );
input a,b,c;
output sum,carry;
wire sum,carry;
assign sum=a^b^c;
assign carry=((a&b)|(b&c)|(c&a));
endmodule
```

RTL Schematic:



Simulation:

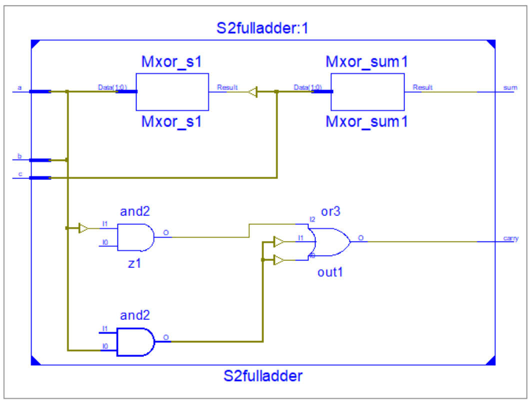


Structural Modelling

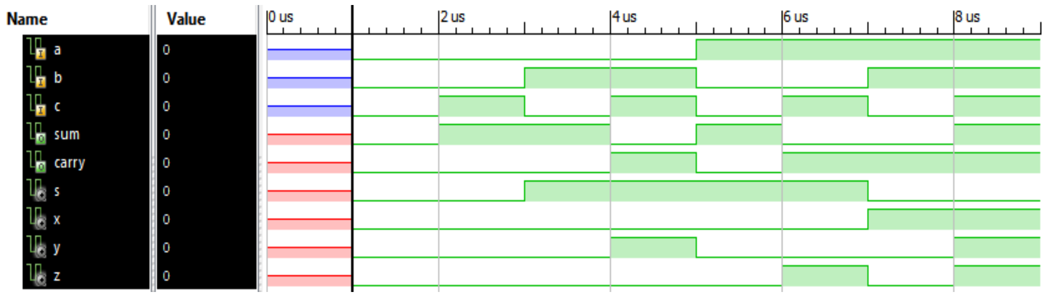
Verilog Program:

```
module S2fulladder(a,b,c,sum,carry
    );
input a,b,c;
output sum,carry;
wire sum,carry,s,x,y,z;
xor x1(s,a,b);
xor x2(sum,c,s);
and a1(x,a,b);
and a2(y,b,c);
and a3(z,c,a);
or o1(carry,x,y,z);
endmodule
```

RTL Schematic:



Simulation:

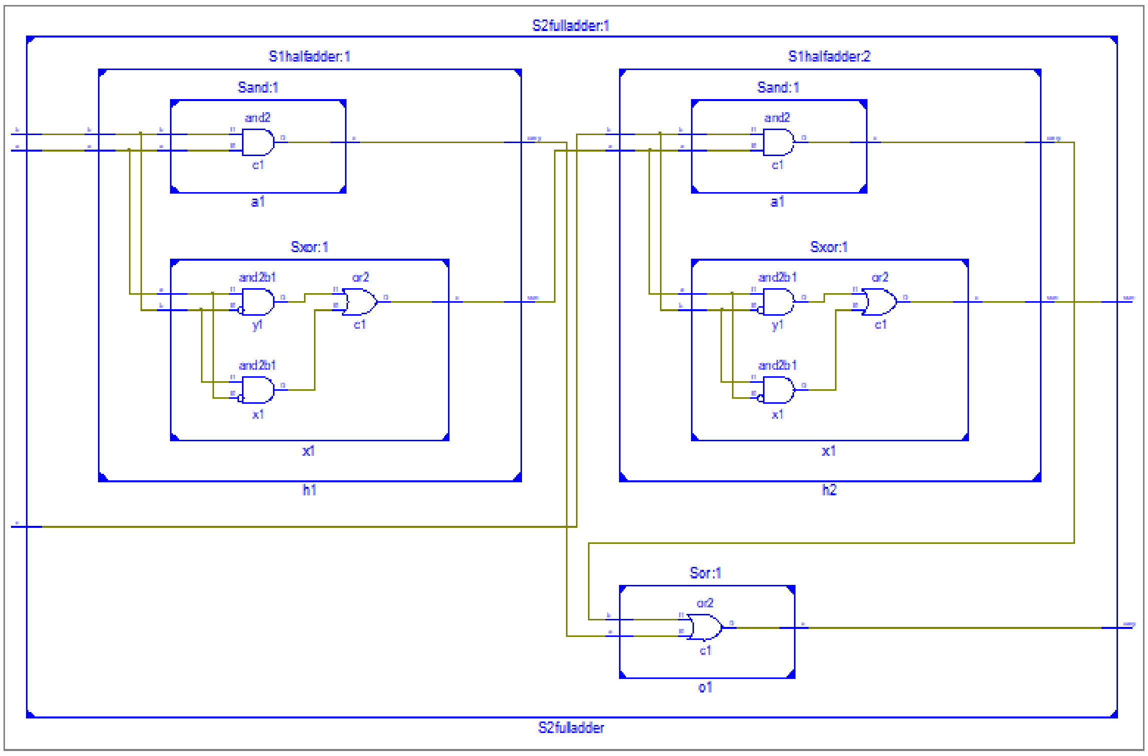


Full adder using half adder

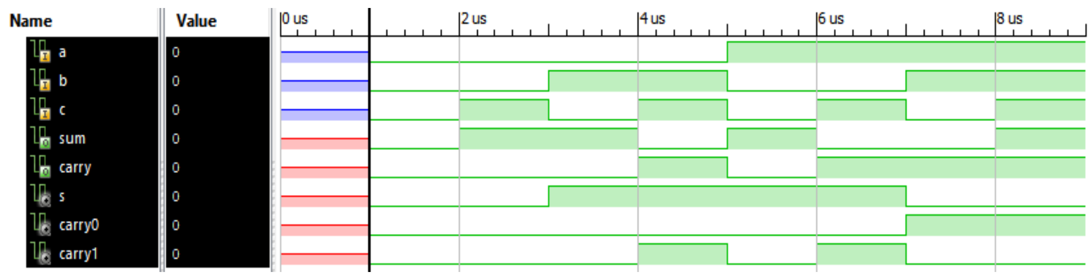
Verilog Program:

```
module S2fulladder(a,b,c,sum,carry
    );
input a,b,c;
output sum,carry;
wire sum,s,carry0,carry1,carry;
S1halfadder h1(a,b,s,carry0);
S1halfadder h2(s,c,sum,carry1);
Sor o1(carry0,carry1,carry);
endmodule
```

RTL Schematic:



Simulation:



Verilog reports:

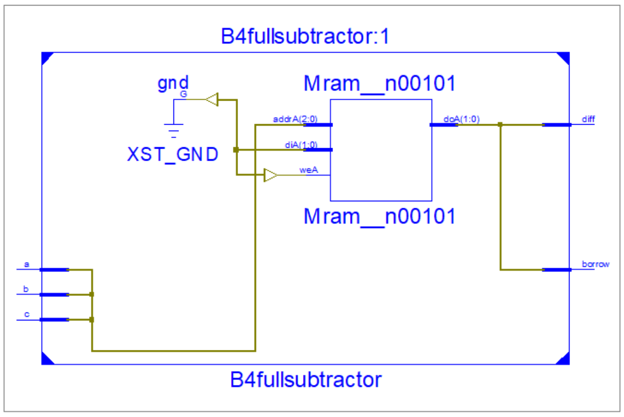
FULL SUBTRACTOR

Behavioral Modelling

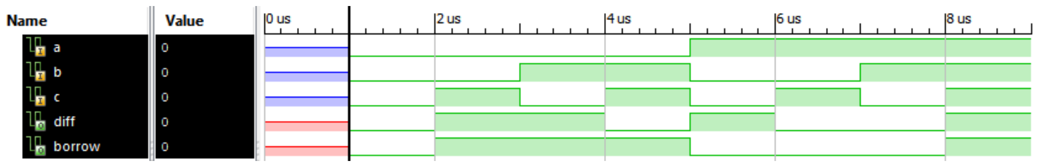
Verilog Program:

```
module
B4fullsubtractor(a,b,c,diff,borrow
    );
input a,b,c;
output diff,borrow;
reg diff,borrow;
always@(a,b,c)
begin
case({a,b,c})
3'b000:
begin diff=1'b0; borrow=1'b0; end
3'b001:
begin diff=1'b1; borrow=1'b1; end
3'b010:
begin diff=1'b1; borrow=1'b1; end
3'b011:
 begin diff=1'b0; borrow=1'b1; end
3'b100:
begin diff=1'b1; borrow=1'b0; end
3'b101:
begin diff=1'b0; borrow=1'b0; end
3'b110:
begin diff=1'b0; borrow=1'b0; end
3'b111:
begin diff=1'b1; borrow=1'b1; end
default:
begin diff=1'bx; borrow=1'bx; end
endcase
end
endmodule
```

RTL Schematic:



Simulation:

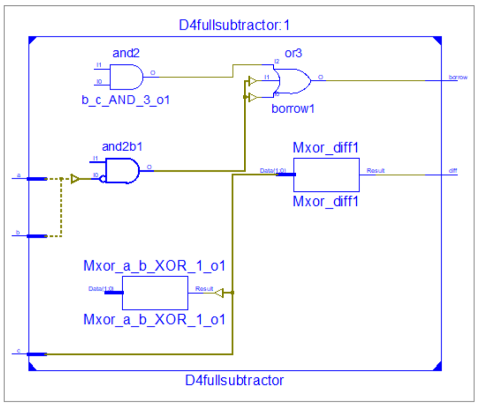


Dataflow Modelling

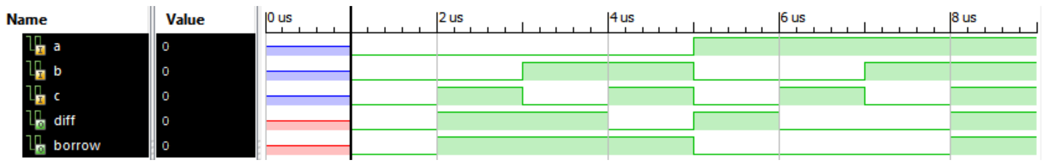
Verilog Program:

```
module D4fullsubtractor(a,b,c,diff,borrow
    );
input a,b,c;
output diff,borrow;
wire diff,borrow;
assign diff=a^b^c;
assign borrow=((~a&b)|(~a&c)|(b&c));
endmodule
```

RTL Schematic:



Simulation:

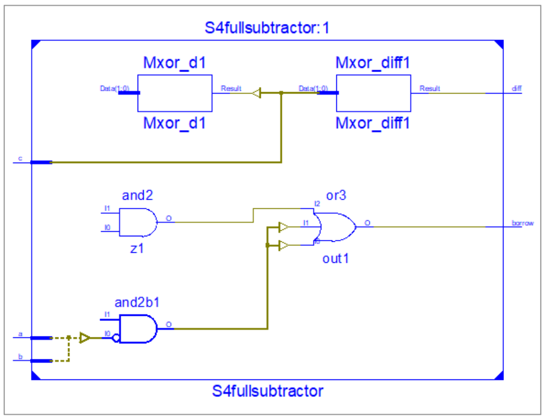


Structural Modelling

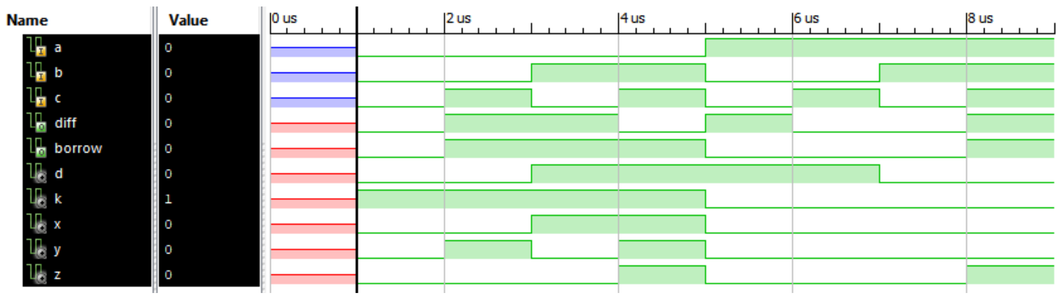
Verilog Program:

```
module S4fullsubtractor(a,b,c,diff,borrow
    );
input a,b,c;
output diff,borrow;
wire diff,borrow,d,k,x,y,z;
xor x1(d,a,b);
xor x2(diff,c,d);
not n1(k,a);
and a1(x,k,b);
and a2(y,k,c);
and a3(z,b,c);
or o1(borrow,x,y,z);
endmodule
```

RTL Schematic:



Simulation:

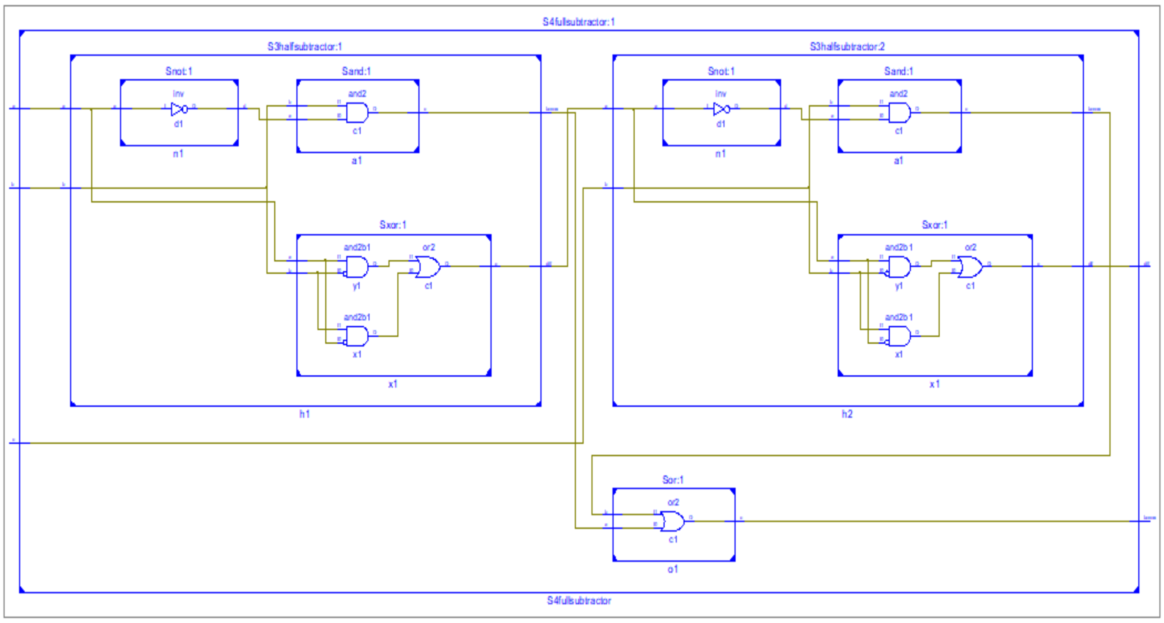


Full subtractor using Half subtractor

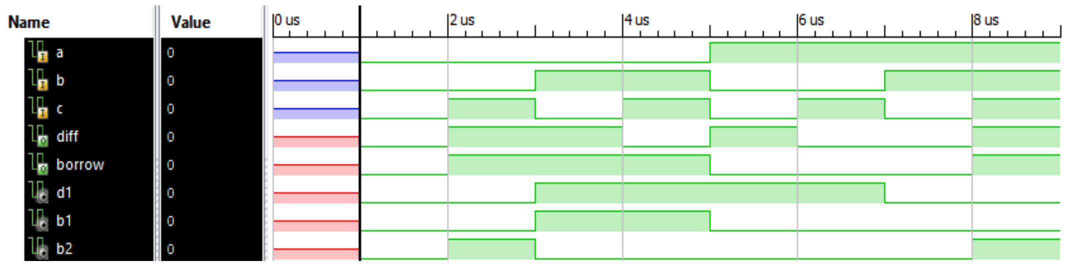
Verilog Program:

```
module S4fullsubtractor(a,b,c,diff,borrow
    );
input a,b,c;
output diff,borrow;
wire diff,d1,borrow,b1,b2;
S3halfsubtractor h1(a,b,d1,b1);
S3halfsubtractor h2(d1,c,diff,b2);
Sor o1(b1,b2,borrow);
endmodule
```

RTL Schematic:



Simulation:



Result:

Designed and implemented adders and subtractors using Behavioral ,dataflow ,structural modelling using **Xilinx ISE 14.2**

EXPERIMENT-3
CODE CONVERTERS

Aim: Design, simulate and implement code converters using Xilinx ISE.
Software Used: Xilinx **ISE-14.2**
Simulator Used: iSim
Synthesizer Used: XST
Procedure:

- Check the syntax of the program for any errors if any correct and verify again.
- In the process window, put the simulation mode in behavioral model, take a Verilog text fixture and give the input combinations.
- Perform simulation to verify the functionality and logic of the code.
- In the process window, change it to implement design highlight program in the source window, performs the run operations to implement the design.
- To view RTL and Technology schematic.

Verilog reports:

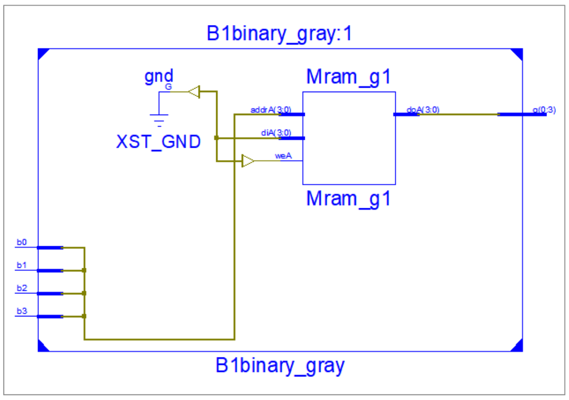
BINARY TO GRAY CONVERTER

Behavioral Modelling

Verilog Program:

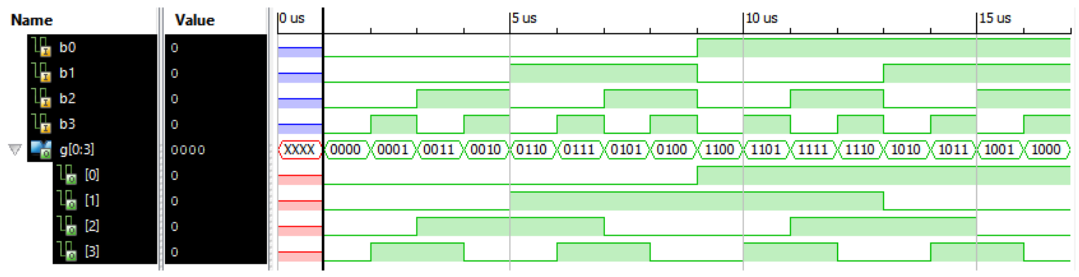
```
module B1binary_gray(b0,b1,b2,b3,g
    );
input b0,b1,b2,b3;
output [0:3]g;
reg [0:3]g;
always@(b0,b1,b2,b3)
begin
case({b0,b1,b2,b3})
4'b0000: g=4'b0000;
4'b0001: g=4'b0001;
4'b0010: g=4'b0011;
4'b0011: g=4'b0010;
4'b0100: g=4'b0110;
4'b0101: g=4'b0111;
4'b0110: g=4'b0101;
4'b0111: g=4'b0100;
4'b1000: g=4'b1100;
4'b1001: g=4'b1101;
4'b1010: g=4'b1111;
4'b1011: g=4'b1110;
4'b1100: g=4'b1010;
4'b1101: g=4'b1011;
4'b1110: g=4'b1001;
```

RTL Schematic:



```
4'b1111: g=4'b1000;
default: g=4'bxxxx;
endcase
end
endmodule
```

Simulation:

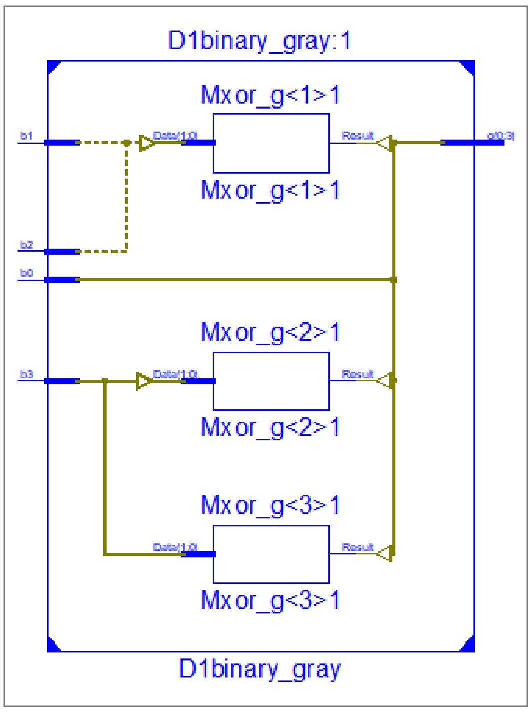


Dataflow Modelling

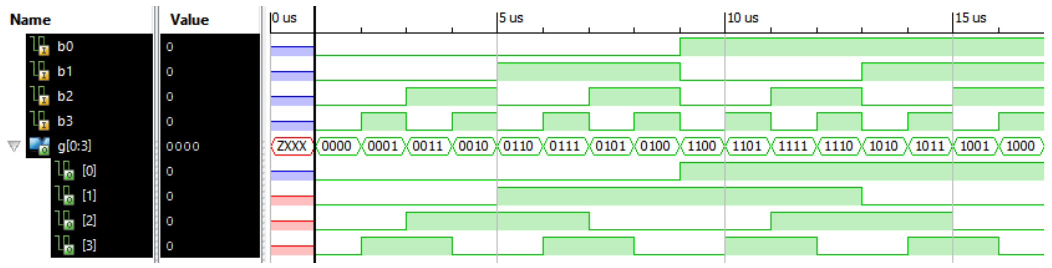
Verilog Program:

```
module D1binary_gray(b0,b1,b2,b3,g
    );
input b0,b1,b2,b3;
output [0:3]g;
wire [0:3]g;
assign g[0]=b0;
assign g[1]=b0^b1;
assign g[2]=b1^b2;
assign g[3]=b2^b3;
endmodule
```

RTL Schematic:



Simulation:

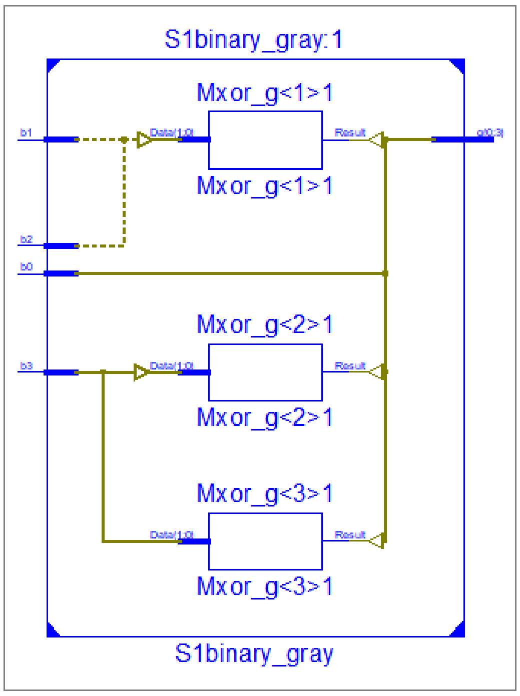


Structural Modelling

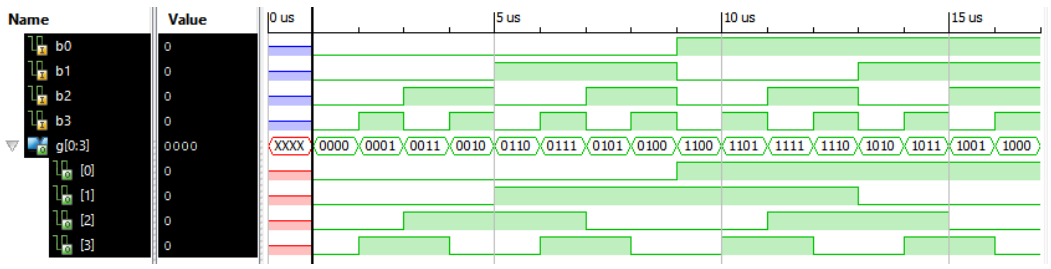
Verilog Program:

```
module S1binary_gray(b0,b1,b2,b3,g
    );
input b0,b1,b2,b3;
output [0:3]g;
wire [0:3]g;
buf g1(g[0],b0);
xor g2(g[1],b0,b1);
xor g3(g[2],b1,b2);
xor g4(g[3],b2,b3);
endmodule
```

RTL Schematic:



Simulation:


Name
Value
0 us
5 us
10 us
15 us
b0 0
b1 0
b2 0
b3 0
g[0:3] 0000
[0] 0
[1] 0
[2] 0
[3] 0
XXXX 0000 0001 0011 0010 0110 0111 0101 0100 1100 1101 1111 1110 1010 1011 1001 1000

Verilog reports:

GRAY TO BINARY CONVERTER

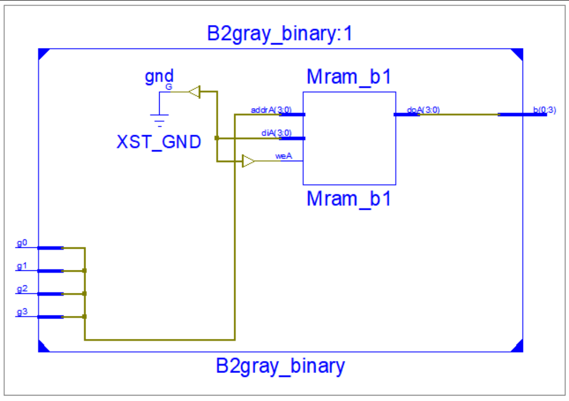
Behavioral Modelling

Verilog Program:

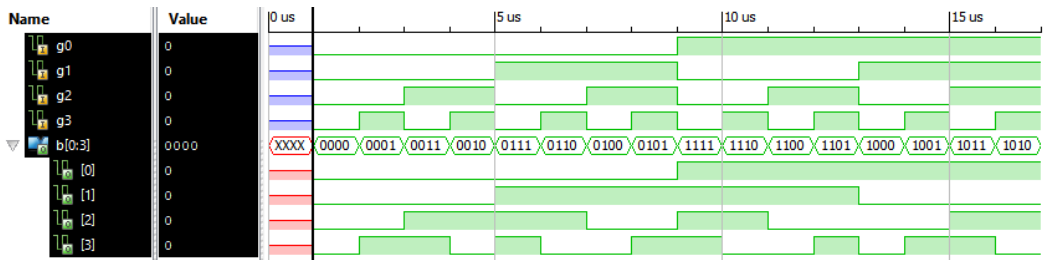
```
module B2gray_binary(g0,g1,g2,g3,b
    );
input g0,g1,g2,g3;
output [0:3]b;
reg [0:3]b;
always@(g0,g1,g2,g3)
begin
case({g0,g1,g2,g3})
4'b0000: b=4'b0000;
4'b0001: b=4'b0001;
```

RTL Schematic:

```
4'b0010: b=4'b0011;
4'b0011: b=4'b0010;
4'b0100: b=4'b0111;
4'b0101: b=4'b0110;
4'b0110: b=4'b0100;
4'b0111: b=4'b0101;
4'b1000: b=4'b1111;
4'b1001: b=4'b1110;
4'b1010: b=4'b1100;
4'b1011: b=4'b1101;
4'b1100: b=4'b1000;
4'b1101: b=4'b1001;
4'b1110: b=4'b1011;
4'b1111: b=4'b1010;
default: b=4'bxxxx;
endcase
end
endmodule
```



Simulation:

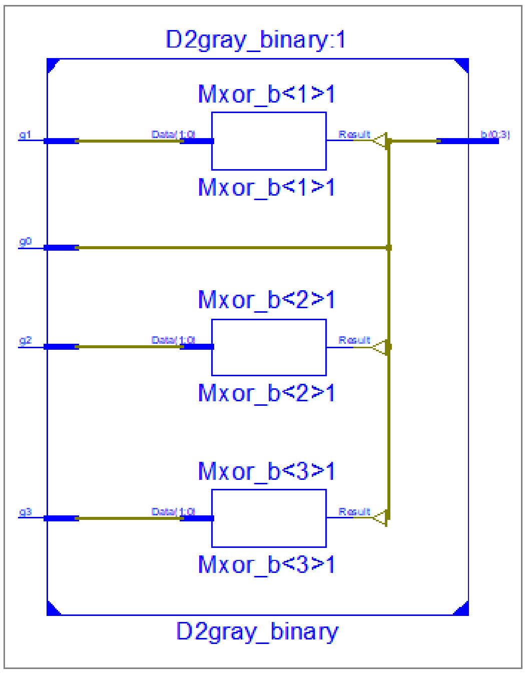


Dataflow Modelling

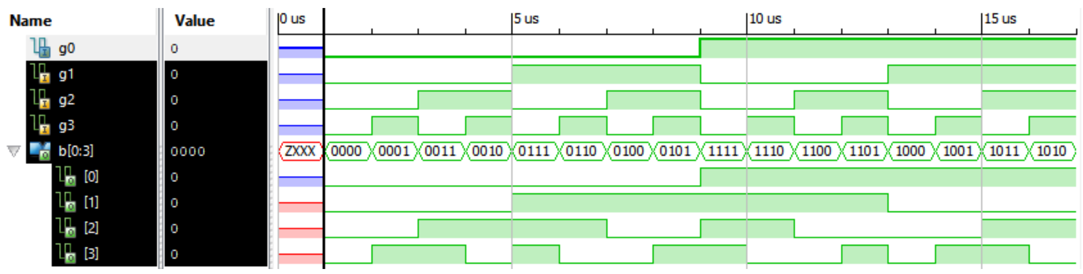
Verilog Program:

```
module D2gray_binary(g0,g1,g2,g3,b
    );
input g0,g1,g2,g3;
output [0:3]b;
wire [0:3]b;
assign b[0]=g0;
assign b[1]=b[0]^g1;
assign b[2]=b[1]^g2;
assign b[3]=b[2]^g3;
endmodule
```

RTL Schematic:



Simulation:

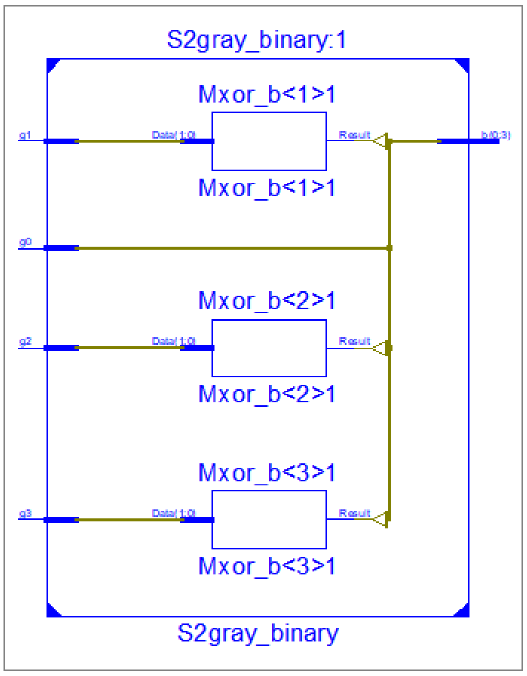


Structural Modelling

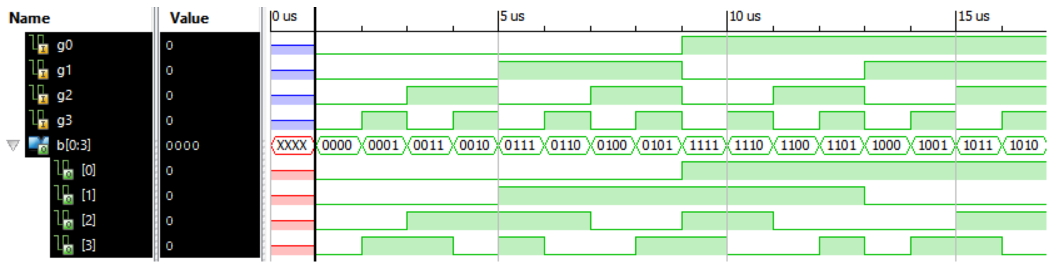
Verilog Program:

```
module S2gray_binary(g0,g1,g2,g3,b
    );
input g0,g1,g2,g3;
output [0:3]b;
wire [0:3]b;
buf b1(b[0],g0);
xor b2(b[1],b[0],g1);
xor b3(b[2],b[1],g2);
xor b4(b[3],b[2],g3);
endmodule
```

RTL Schematic:



Simulation:



Result:

Designed and implemented code converters using Behavioral,dataflow,structural modelling using **Xilinx ISE 14.2**

EXPERIMENT-4
MULTIPLEXERS AND DEMULTIPLEXERS

Aim: Design, simulate and implement multiplexers and demultiplexers using Xilinx ISE.
Software Used: Xilinx **ISE-14.2**
Simulator Used: iSim
Synthesizer Used: XST
Procedure:

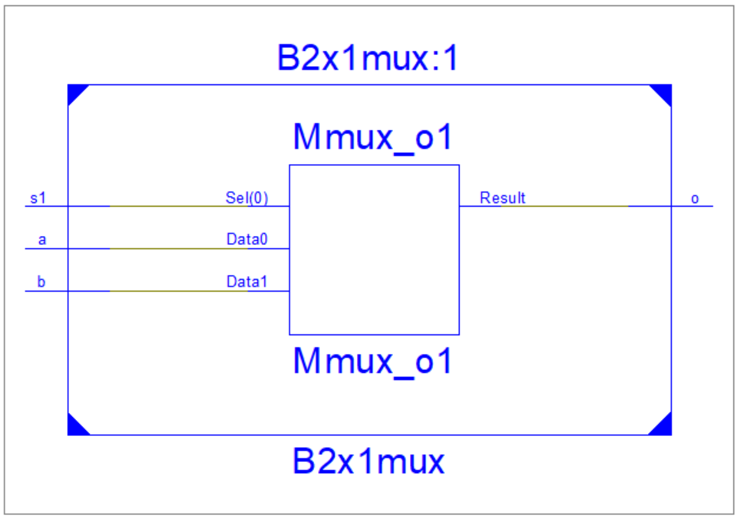
- Check the syntax of the program for any errors if any correct and verify again.
- In the process window, put the simulation mode in Behavioral model, take a Verilog text fixture and give the input combinations.
- Perform simulation to verify the functionality and logic of the code.
- In the process window, change it to implement design highlight program in the source window, performs the run operations to implement the design.
- To view RTL and Technology schematic.

Verilog reports:

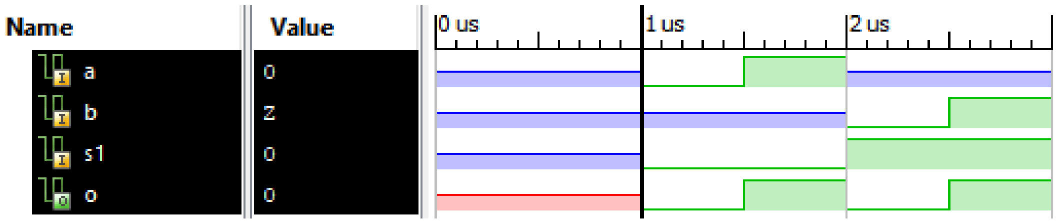
2X1 MULTIPLEXER

Behavioral Modelling

Verilog Program:	RTL Schematic:

```
module B2x1mux(o,a,b,s1
    );
input a,b,s1;
output o;
reg o;
always@(a,b,s1)
begin
case({s1})
1'b0:o=a;
1'b1:o=b;
default:o=1'bx;
endcase
end
endmodule
```
	 B2x1mux:1
	Mmux_o1
	s1
	Sel(0)
	a
	Data0
	b
	Data1
	Result
	o
	Mmux_o1
	B2x1mux

Simulation:

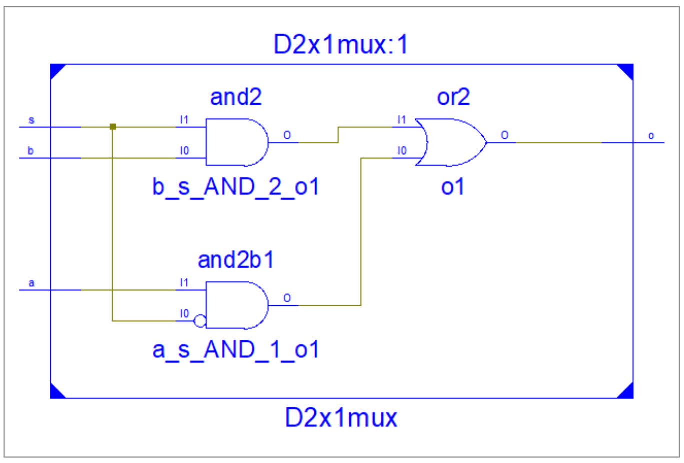


Dataflow Modelling

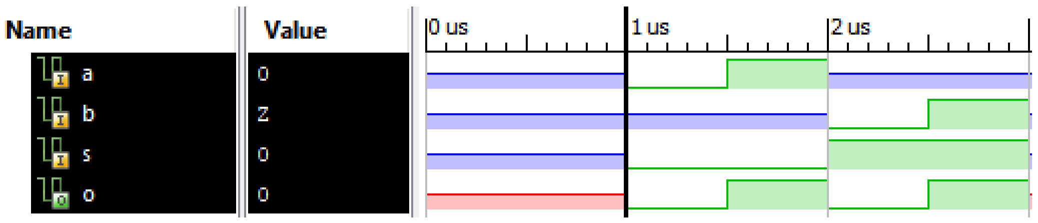
Verilog Program:

```
module D2x1mux(o,a,b,s
    );
input a,b,s;
output o;
wire o;
assign o=(a&s)|(b&~s);
endmodule
```

RTL Schematic:



Simulation:

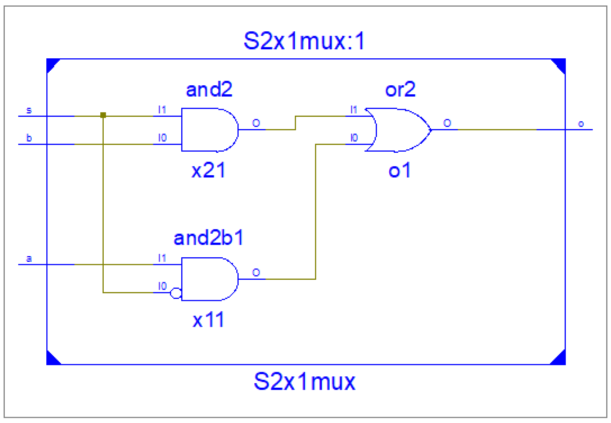


Structural Modelling

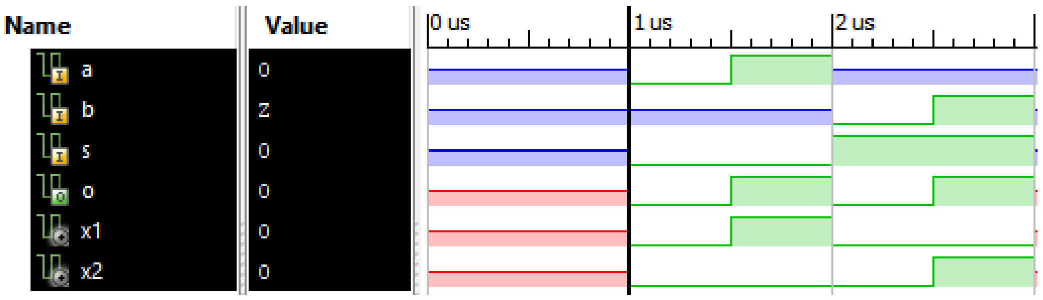
Verilog Program:

```
module S2x1mux(o,a,b,s
    );
input a,b,s;
output o;
wire o,x1,x2;
and a1(x1,a,~s);
and a2(x2,b,s);
or  o1(o,x1,x2);
endmodule
```

RTL Schematic:



Simulation:



Verilog reports:

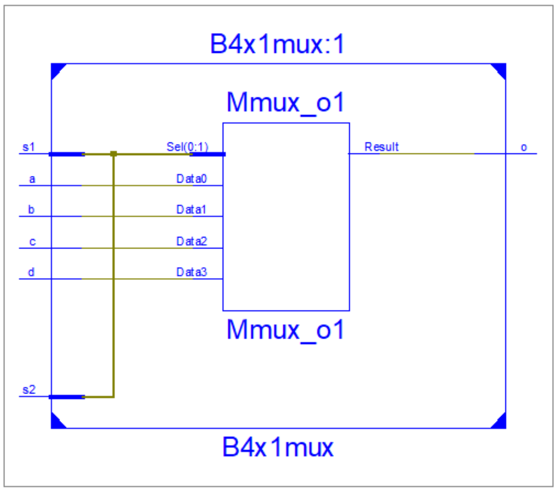
4X1 MULTIPLEXER

Behavioral Modelling

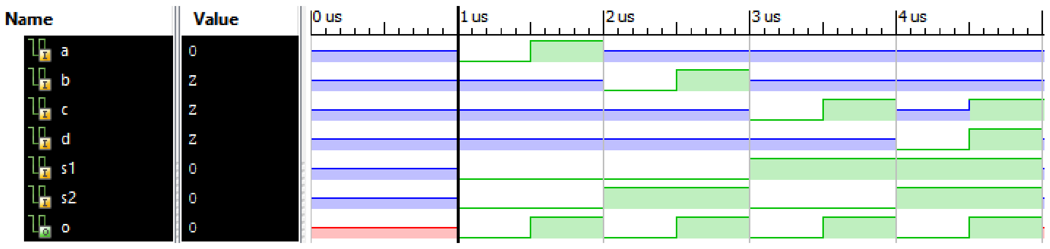
Verilog Program:

```
module B4x1mux(o,a,b,c,d,s1,s2
   );
input a,b,c,d,s1,s2;
output o;
reg o;
always@(a,b,c,d,s1,s2)
begin
case({s1,s2})
2'b00:o=a;
2'b01:o=b;
2'b10:o=c;
2'b11:o=d;
default:o=1'bx;
endcase
end
endmodule
```

RTL Schematic:



Simulation:



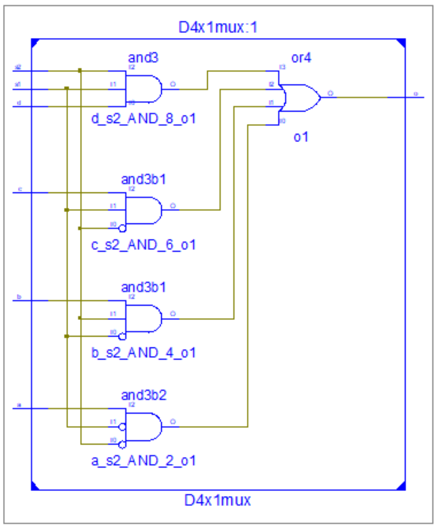
Name	Value
a	0
b	z
c	z
d	z
s1	0
s2	0
o	0

Dataflow Modelling

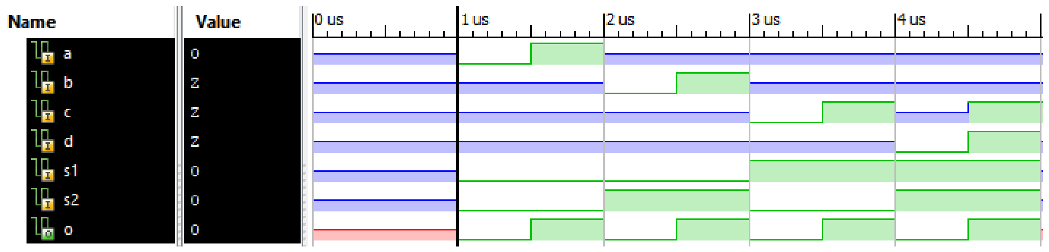
Verilog Program:

```
module D4x1mux(o,a,b,c,d,s1,s2
   );
input a,b,c,d,s1,s2;
output o;
wire o;
assign
o=(a&~s1&~s2)|(b&~s1&s2)|(c&s1&~s2)|(d&s1&s
2);
endmodule
```

RTL Schematic:



Simulation:

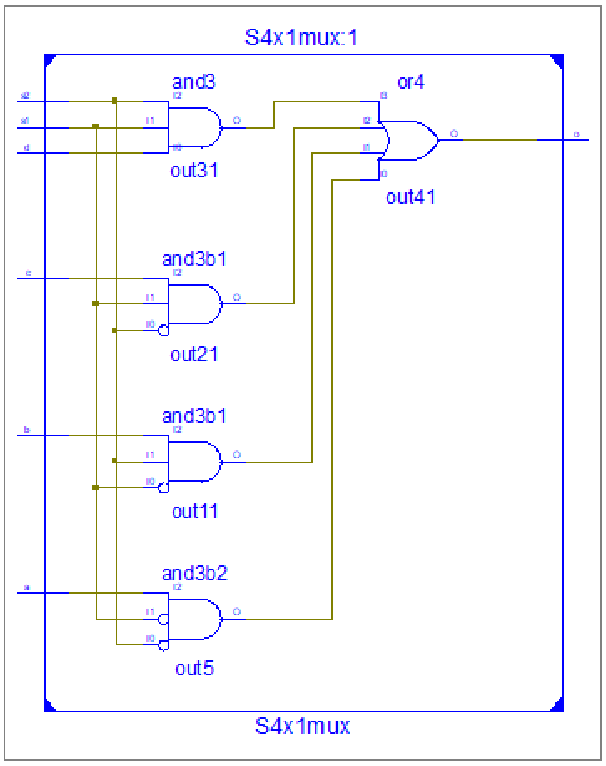


Structural Modelling

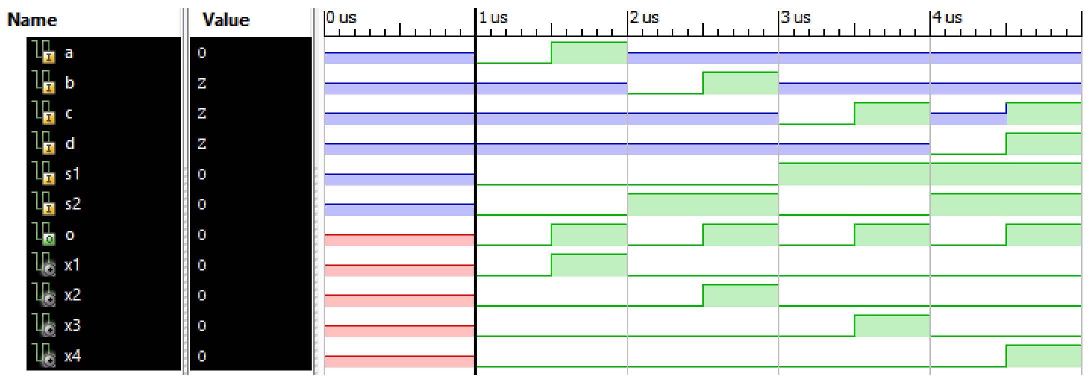
Verilog Program:

```
module S4x1mux(o,a,b,c,d,s1,s2
    );
input a,b,c,d,s1,s2;
output o;
wire o,x1,x2,x3,x4;
and a1(x1,a,~s1,~s2);
and a2(x2,b,~s1,s2);
and a3(x3,c,s1,~s2);
and a4(x4,d,s1,s2);
or o1(o,x1,x2,x3,x4);
endmodule
```

RTL Schematic:



Simulation:



Verilog reports:

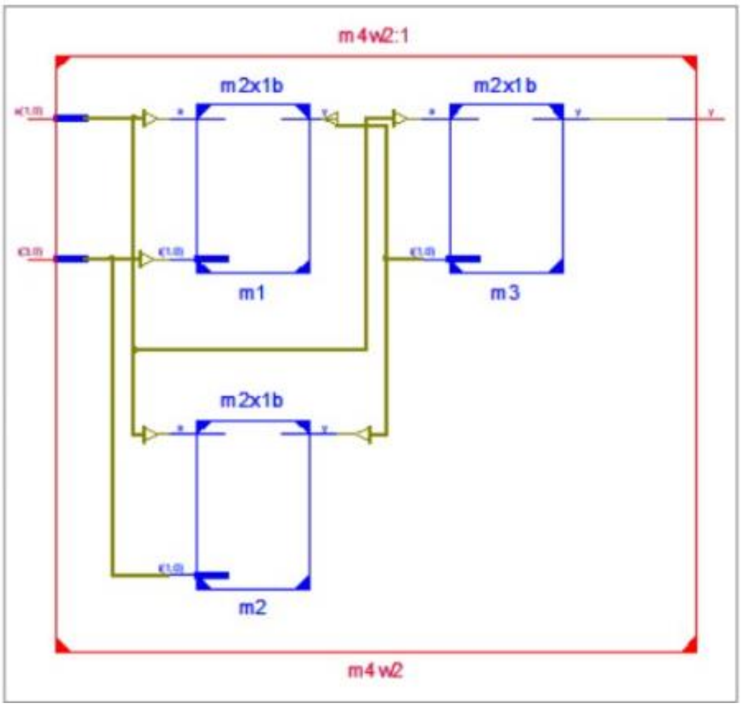
4X1 USING 2X1 MULTIPLEXER

Structural Modelling

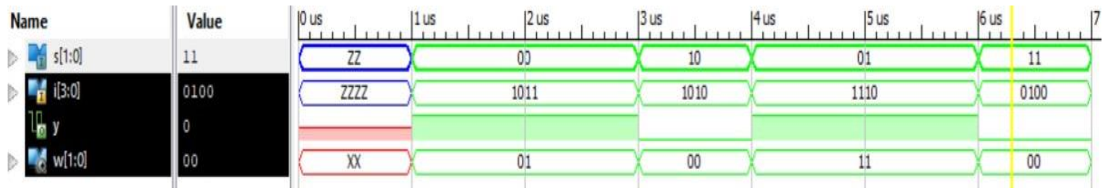
Verilog Program

```
module m4w2(y,s,i);
input [1:0]s;
input [3:0]i;
output y;
wire [1:0]w;
m2x1b m1(w[0],s[0],i[1:0]);
m2x1b m2(w[1],s[0],i[3:2]);
m2x1b m3(y,s[1],w[1:0]);
endmodule
```

RTL Schematic:



Simulation Result:



Verilog reports:

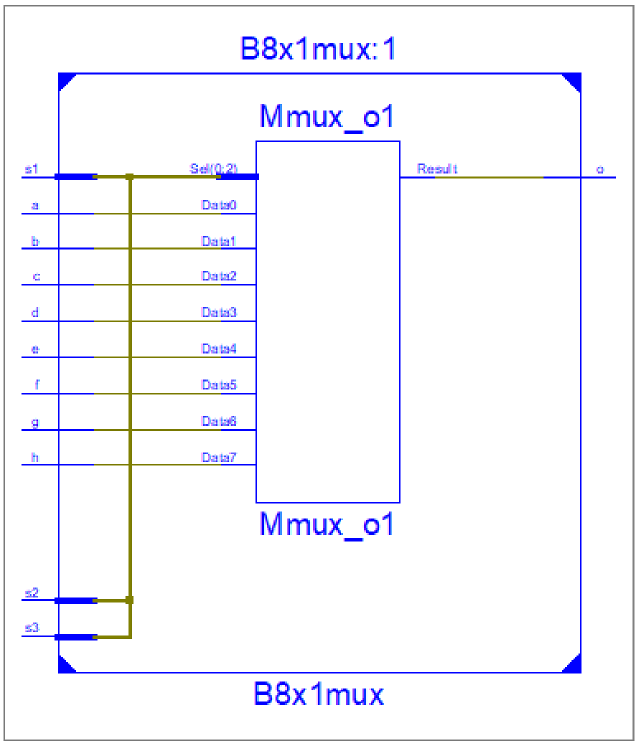
8X1 MULTIPLEXER

Behavioral Modelling

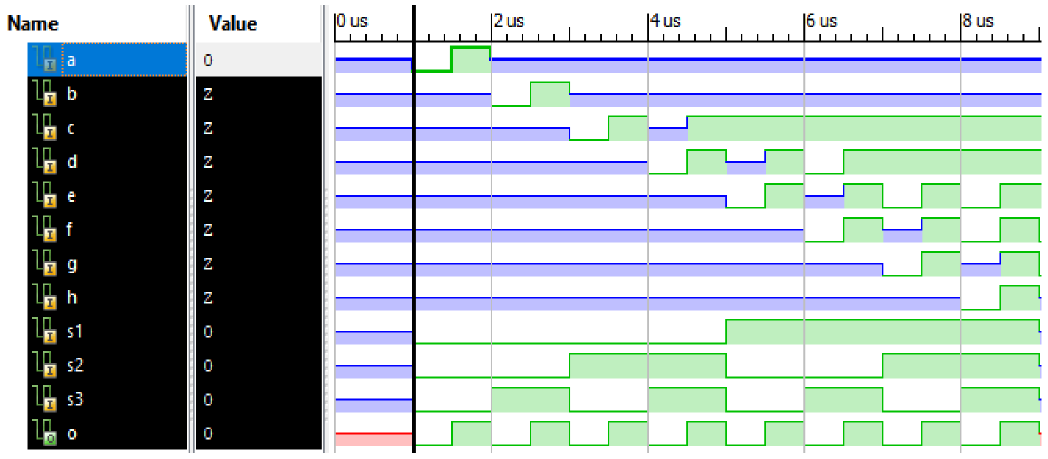
Verilog Program:

```
module
B8x1mux(o,a,b,c,d,e,f,g,h,s1,s2,s3
);
input a,b,c,d,e,f,g,h,s1,s2,s3;
output o;
reg o;
always@(a,b,c,d,e,f,g,h,s1,s2,s3)
begin
case({s1,s2,s3})
4'b000:o=a;
4'b001:o=b;
4'b010:o=c;
4'b011:o=d;
4'b100:o=e;
4'b101:o=f;
4'b110:o=g;
4'b111:o=h;
default:o=1'bx;
endcase
end
endmodule
```

RTL Schematic:



Simulation:

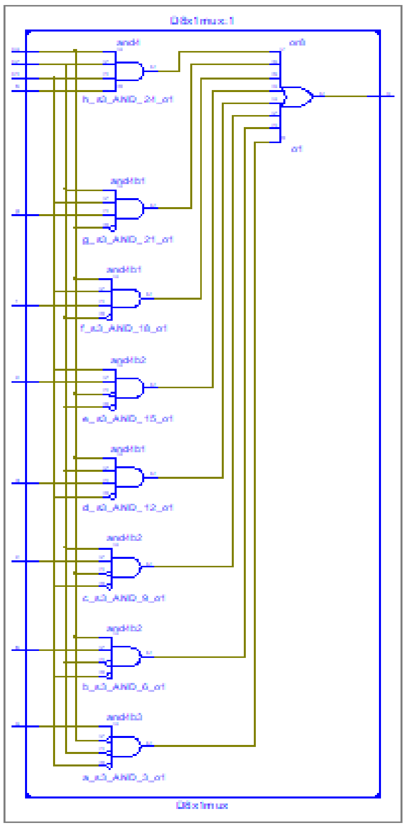


Dataflow Modelling

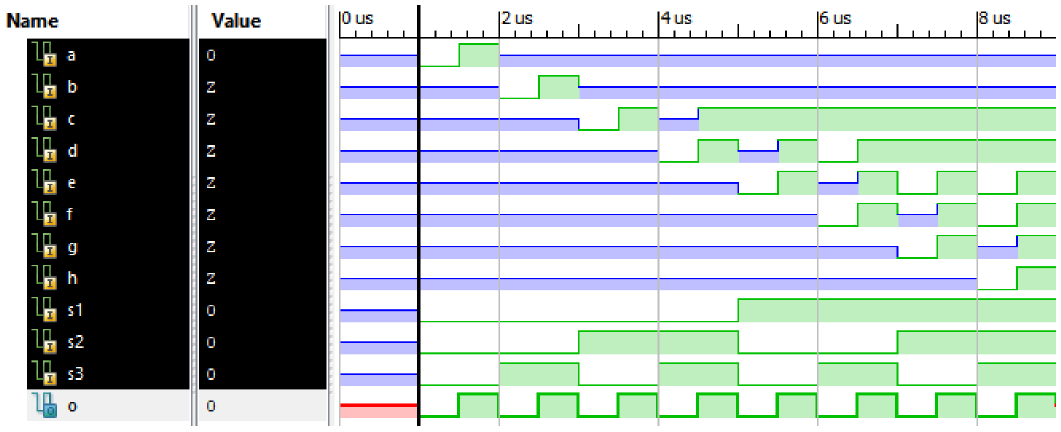
Verilog Program:

```
module D8x1mux(o,a,b,c,d,e,f,g,h,s1,s2,s3
    );
input a,b,c,d,e,f,g,h,s1,s2,s3;
output o;
wire o;
assign
o=(a&~s1&~s2&~s3)|(b&~s1&~s2&s3)|(c&~s1&s2&~s3)|(d&~s1&s2&s3)|(e&s1&~s2&
~s3)|(f&s1&~s2&s3)|(g&s1&s2&~s3)|(h&s1&s2&s3);
endmodule
```

RTL Schematic:



Simulation:



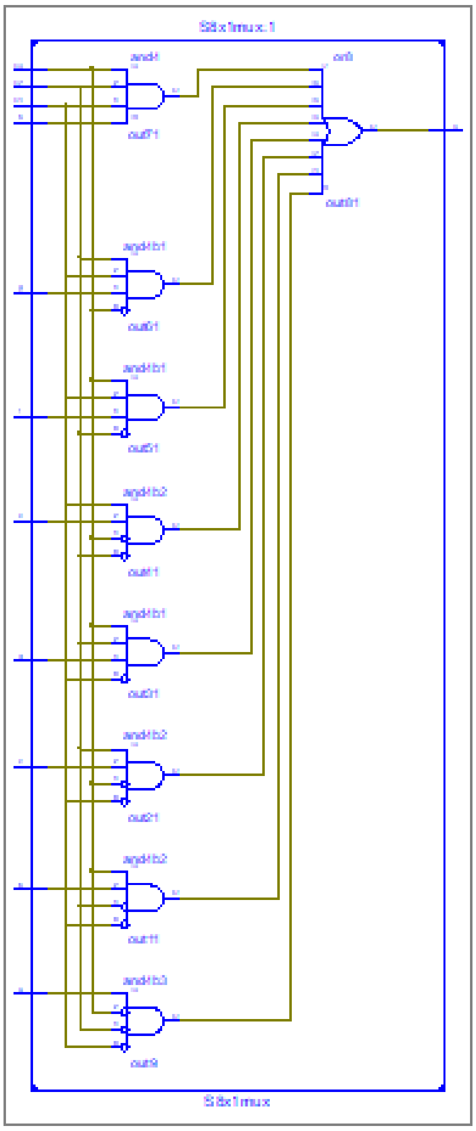
Name	Value
a	0
b	Z
c	Z
d	Z
e	Z
f	Z
g	Z
h	Z
s1	0
s2	0
s3	0
o	0

Structural Modelling

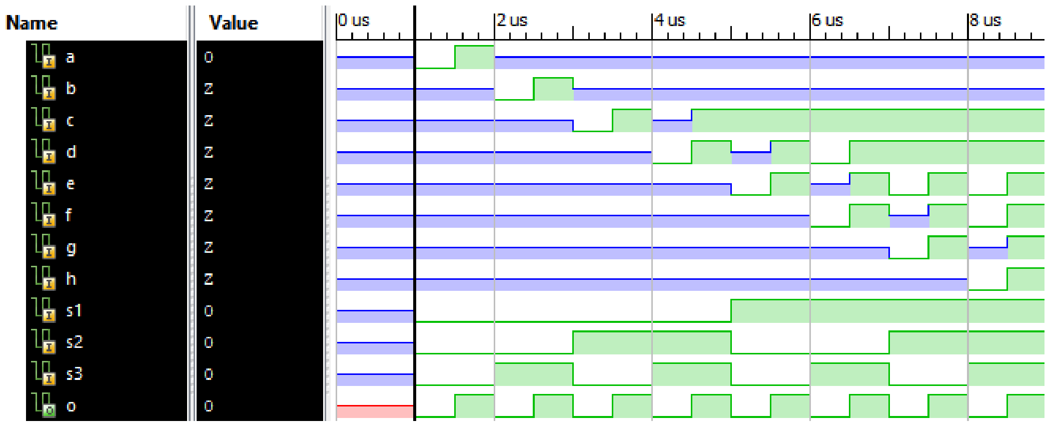
Verilog Program:

```
module S8x1mux(o,a,b,c,d,e,f,g,h,s1,s2,s3
    );
input a,b,c,d,e,f,g,h,s1,s2,s3;
output o;
wire o,x1,x2,x3,x4,x5,x6,x7,x8;
and a1(x1,a,~s1,~s2,~s3);
and a2(x2,b,~s1,~s2,s3);
and a3(x3,c,~s1,s2,~s3);
and a4(x4,d,~s1,s2,s3);
and a5(x5,e,s1,~s2,~s3);
and a6(x6,f,s1,~s2,s3);
and a7(x7,g,s1,s2,~s3);
and a8(x8,h,s1,s2,s3);
or o1(o,x1,x2,x3,x4,x5,x6,x7,x8);
endmodule
```

RTL Schematic:



Simulation:



Verilog reports:

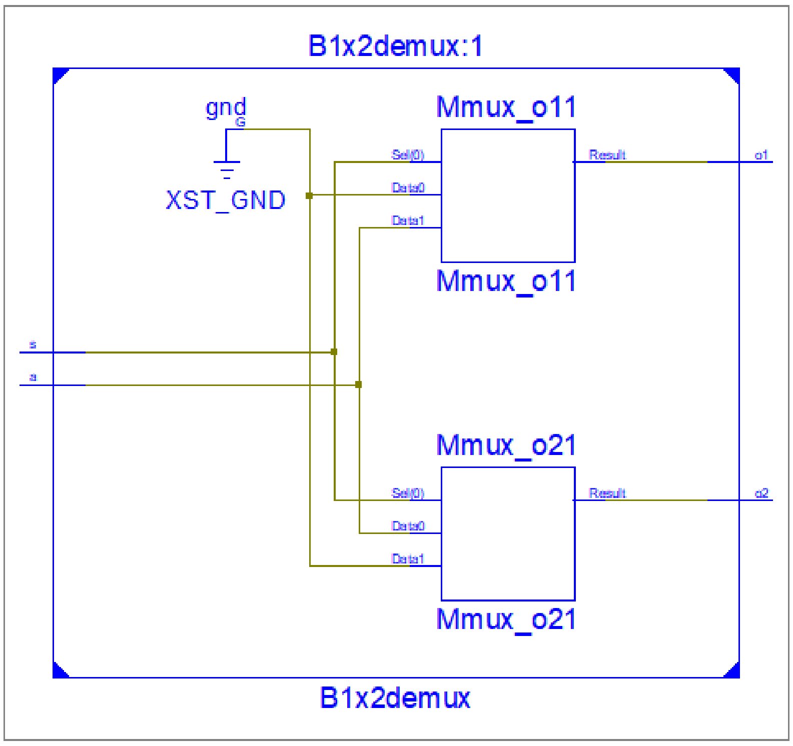
1X2 DEMULTIPLEXER

Behavioral Modelling

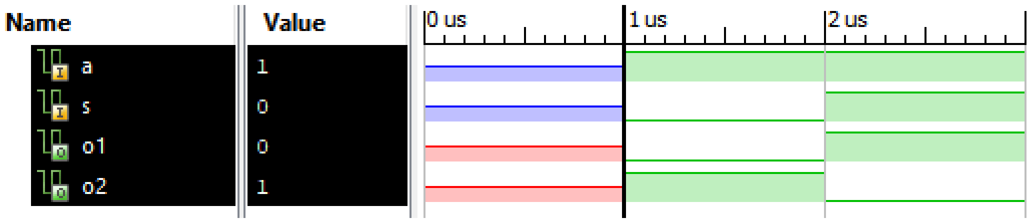
Verilog Program:

```
odule B1x2demux(o1,o2,a,s
    );
input a,s;
output o1,o2;
reg o1,o2;
always@(s)
begin
case(s)
1'b1:begin o1=a;o2=1'b0; end
1'b0:begin o1=1'b0;o2=a; end
endcase
end
endmodule
```

RTL Schematic:



Simulation:

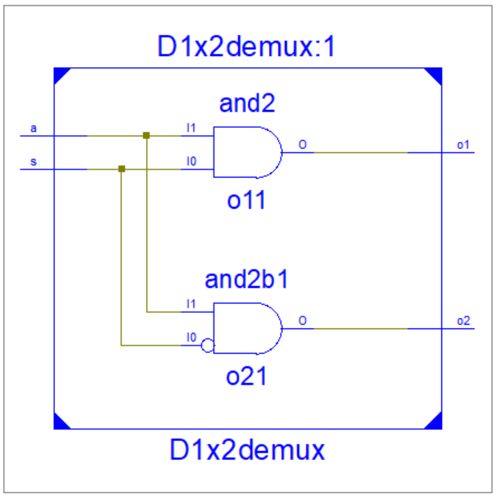


Dataflow Modelling

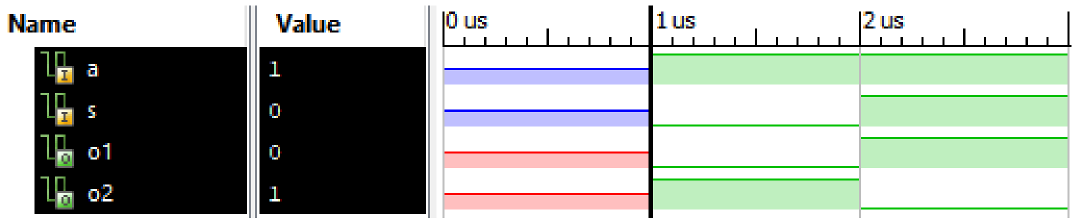
Verilog Program:

```
module D1x2demux(o1,o2,a,s
    );
input a,s;
output o1,o2;
wire o1,o2;
assign o1=s&a;
assign o2=~s&a;
endmodule
```

RTL Schematic:



Simulation:



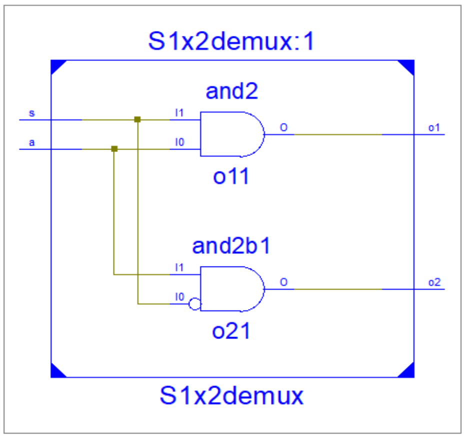
```
Name  Value
a     1
s     0
o1    0
o2    1
```

Structural Modelling

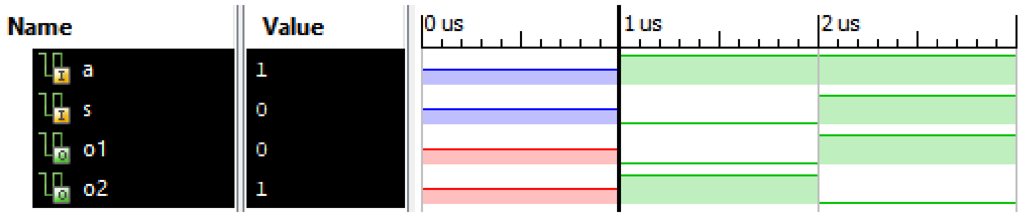
Verilog Program:

```
module S1x2demux(o1,o2,a,s
    );
input a,s;
output o1,o2;
wire o1,o2;
and a1(o1,a,s);
and a2(o2,a,~s);
endmodule
```

RTL Schematic:



Simulation:



```
Name  Value
a     1
s     0
o1    0
o2    1
```

Verilog reports:

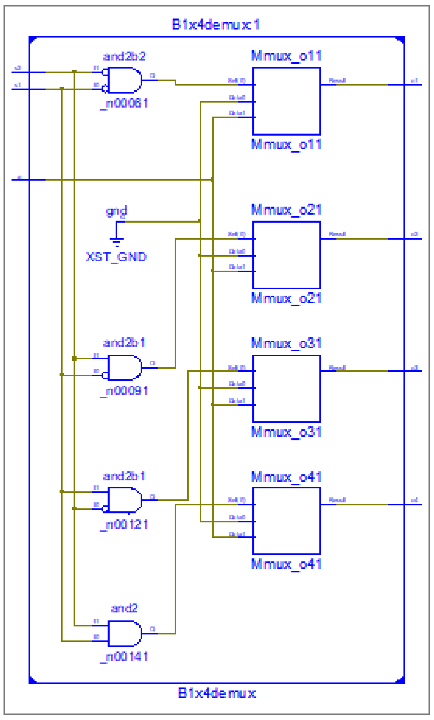
1X4 DEMULTIPLEXER

Behavioral Modelling

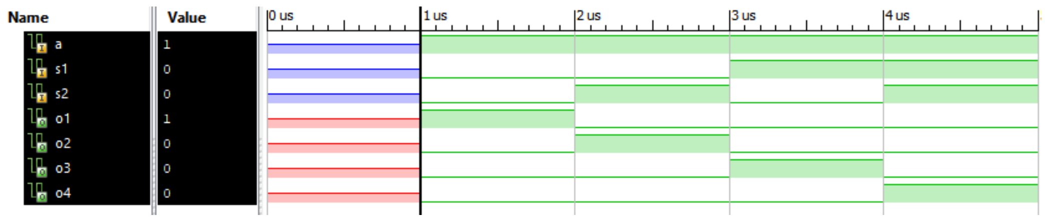
Verilog Program:

```
module B1x4demux(o1,o2,o3,o4,a,s1,s2
    );
input a,s1,s2;
output o1,o2,o3,o4;
reg o1,o2,o3,o4;
always@(s1,s2)
begin
case({s1,s2})
2'b00:begin o1=a;o2=1'b0;o3=1'b0;o4=1'b0; end
2'b01:begin o1=1'b0;o2=a;o3=1'b0;o4=1'b0; end
2'b10:begin o1=1'b0;o2=1'b0;o3=a;o4=1'b0; end
2'b11:begin o1=1'b0;o2=1'b0;o3=1'b0;o4=a; end
default:begin
o1=1'bx;o2=1'bx;o3=1'bx;o4=1'bx;end
endcase
end
endmodule
```

RTL Schematic:



Simulation:

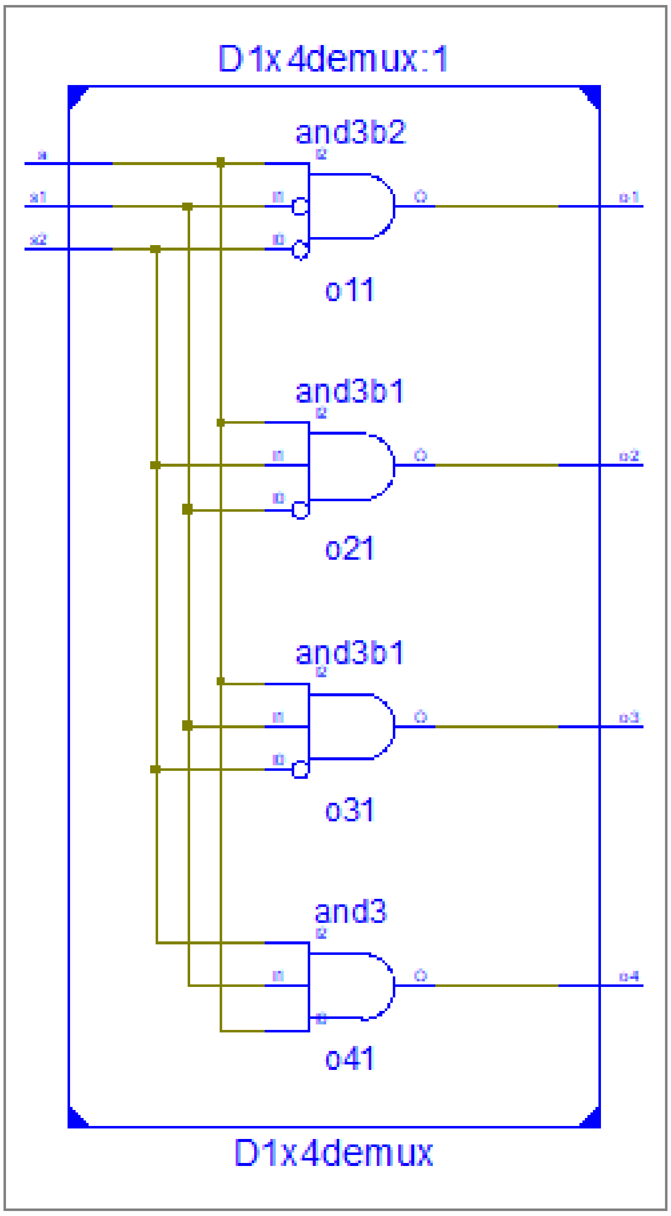


Dataflow Modelling

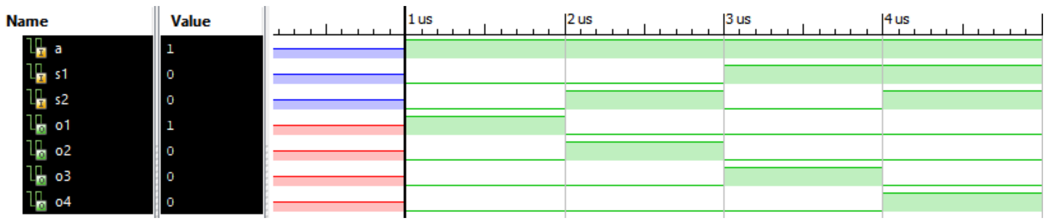
Verilog Program:

```
module D1x4demux(o1,o2,o3,o4,a,s1,s2
    );
input a,s1,s2;
output o1,o2,o3,o4;
wire o1,o2,o3,o4;
assign o1=~s1&~s2&a;
assign o2=~s1&s2&a;
assign o3=s1&~s2&a;
assign o4=s1&s2&a;
endmodule
```

RTL Schematic:



Simulation:



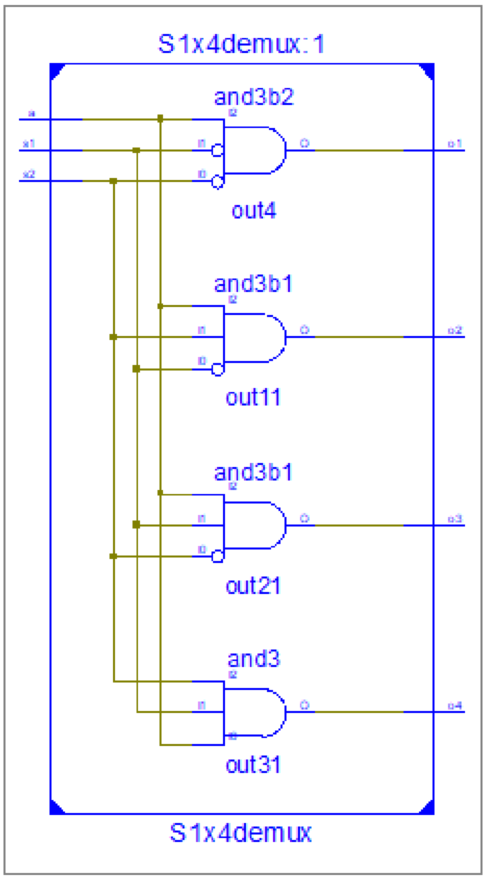
Name	Value
a	1
s1	0
s2	0
o1	1
o2	0
o3	0
o4	0

Structural Modelling

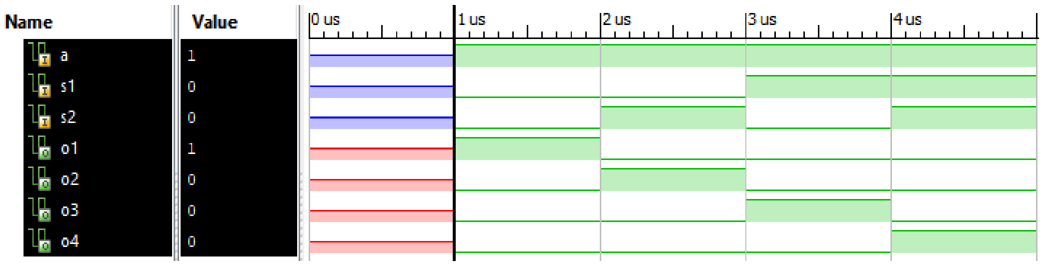
Verilog Program:

```
module S1x4demux(o1,o2,o3,o4,a,s1,s2
   );
input a,s1,s2;
output o1,o2,o3,o4;
wire o1,o2,o3,o4;
and a1(o1,a,~s1,~s2);
and a2(o2,a,~s1,s2);
and a3(o3,a,s1,~s2);
and a4(o4,a,s1,s2);
endmodule
```

RTL Schematic:



Simulation:



Verilog reports:

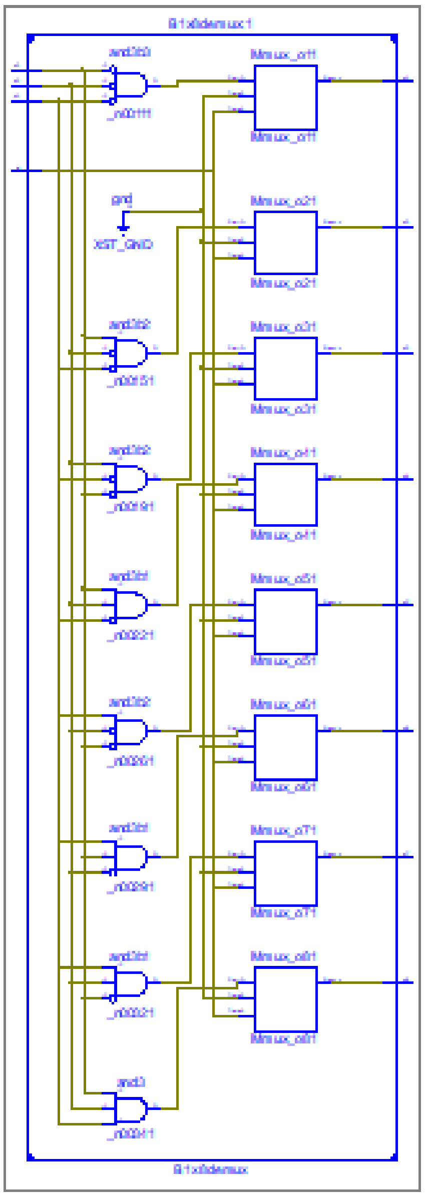
1X8 DEMULTIPLEXER

Behavioral Modelling

Verilog Program:

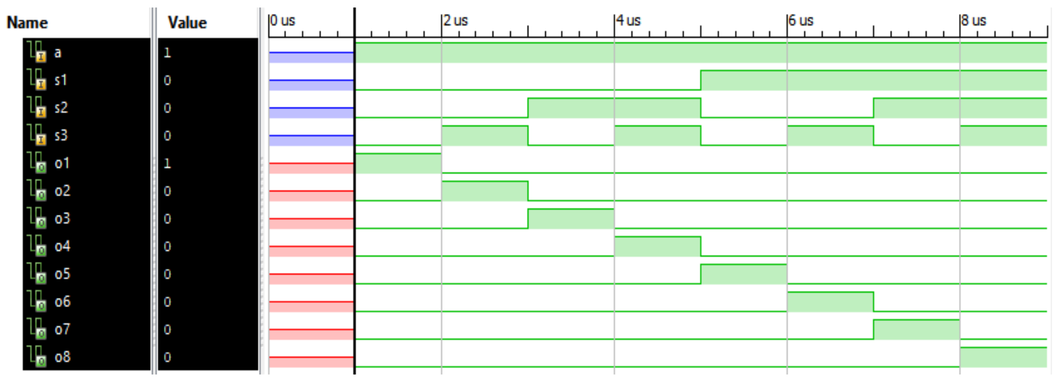
```
module
B1x8demux(o1,o2,o3,o4,o5,o6,o7,o8,a,s1,s2,s3
    );
input a,s1,s2,s3;
output o1,o2,o3,o4,o5,o6,o7,o8;
reg o1,o2,o3,o4,o5,o6,o7,o8;
always@(a,s1,s2,s3)
begin
case({s1,s2,s3})
3'b000:
begin
o1=a;o2=1'b0;
o3=1'b0;o4=1'b0;
o5=1'b0;o6=1'b0;
o7=1'b0;o8=1'b0;
 end
3'b001:
begin
 o1=1'b0;o2=a;
o3=1'b0;o4=1'b0;
o5=1'b0;o6=1'b0;
o7=1'b0;o8=1'b0;
 end
3'b010:
begin
o1=1'b0;o2=1'b0;
o3=a;o4=1'b0;
o5=1'b0;o6=1'b0;
o7=1'b0;o8=1'b0;
end
3'b011:
begin
o1=1'b0;o2=1'b0;
o3=1'b0;o4=a;
o5=1'b0;o6=1'b0;
o7=1'b0;o8=1'b0;
end
3'b100:
begin
o1=1'b0;o2=1'b0;
o3=1'b0;o4=1'b0;
o5=a;o6=1'b0;
o7=1'b0;o8=1'b0;
end
```

RTL Schematic:



```
3'b101:
begin
o1=1'b0;o2=1'b0;
o3=1'b0;o4=1'b0;
o5=1'b0;o6=a;
o7=1'b0;o8=1'b0;
end
3'b110:
begin
 o1=1'b0;o2=1'b0;
o3=1'b0;o4=1'b0;
o5=1'b0;o6=1'b0;
o7=a;o8=1'b0;
 end
3'b111:
begin
o1=1'b0;o2=1'b0;
o3=1'b0;o4=1'b0;
o5=1'b0;o6=1'b0;
o7=1'b0;o8=a;
end
default:
begin
o1=1'bx;o2=1'bx;
o3=1'bx;o4=1'bx;
o5=1'bx;o6=1'bx;
o7=1'bx;o8=1'bx;
end
endcase
end
endmodule
```

Simulation:

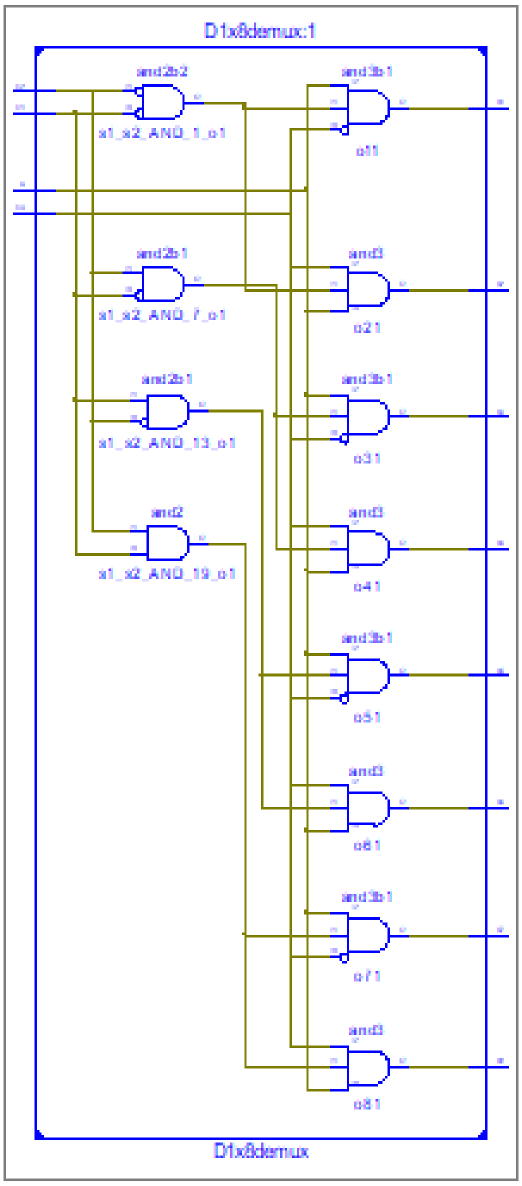


Dataflow Modelling

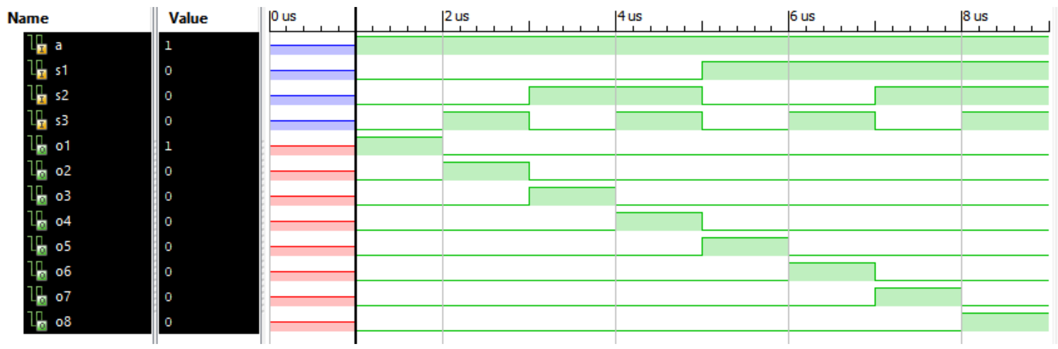
Verilog Program:

```
module
D1x8demux(o1,o2,o3,o4,o5,o6,o7,o8,a,s1,s2
,s3
    );
input a,s1,s2,s3;
output o1,o2,o3,o4,o5,o6,o7,o8;
wire o1,o2,o3,o4,o5,o6,o7,o8;
assign o1=~s1&~s2&~s3&a;
assign o2=~s1&~s2&s3&a;
assign o3=~s1&s2&~s3&a;
assign o4=~s1&s2&s3&a;
assign o5=s1&~s2&~s3&a;
assign o6=s1&~s2&s3&a;
assign o7=s1&s2&~s3&a;
assign o8=s1&s2&s3&a;
endmodule
```

RTL Schematic:



Simulation:



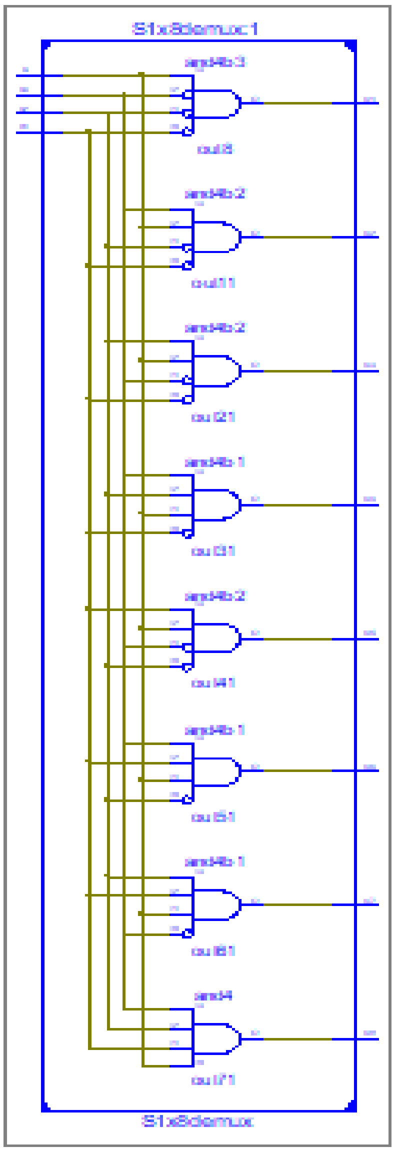
```
Name  Value
a     1
s1    0
s2    0
s3    0
o1    1
o2    0
o3    0
o4    0
o5    0
o6    0
o7    0
o8    0
```

Structural Modelling

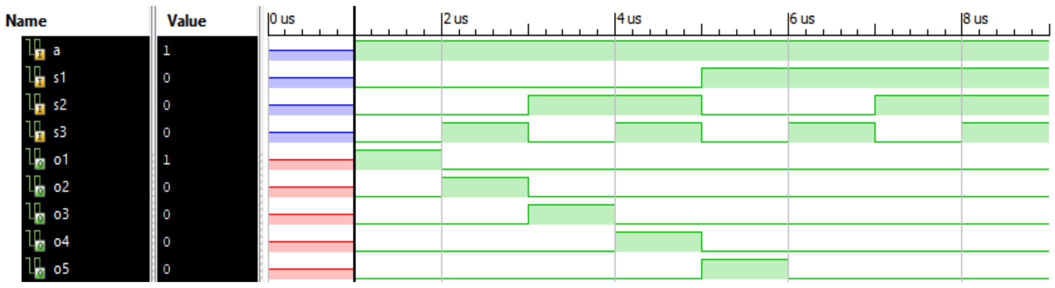
Verilog Program:

```
module
S1x8demux(o1,o2,o3,o4,o5,o6,o7,o8,a,s1,s2,s3
    );
input a,s1,s2,s3;
output o1,o2,o3,o4,o5,o6,o7,o8;
wire o1,o2,o3,o4,o5,o6,o7,o8;
and a1(o1,a,~s1,~s2,~s3);
and a2(o2,a,~s1,~s2,s3);
and a3(o3,a,~s1,s2,~s3);
and a4(o4,a,~s1,s2,s3);
and a5(o5,a,s1,~s2,~s3);
and a6(o6,a,s1,~s2,s3);
and a7(o7,a,s1,s2,~s3);
and a8(o8,a,s1,s2,s3);
endmodule
```

RTL Schematic:



Simulation:



Result:

Designed and implemented multiplexers and demultiplexers using Behavioral, dataflow, structural modelling using **Xilinx ISE 14.2**

EXPERIMENT-5
ENCODERS AND DECODERS

Aim: Design, simulate and implement encoders and decoders using Xilinx ISE.
Software Used: Xilinx **ISE-14.2**
Simulator Used: iSim
Synthesizer Used: XST
Procedure:

- Check the syntax of the program for any errors if any correct and verify again.
- In the process window, put the simulation mode in behavioral model, take a Verilog text fixture and give the input combinations.
- Perform simulation to verify the functionality and logic of the code.
- In the process window, change it to implement design highlight program in the source window, performs the run operations to implement the design.
- To view RTL and Technology schematic.

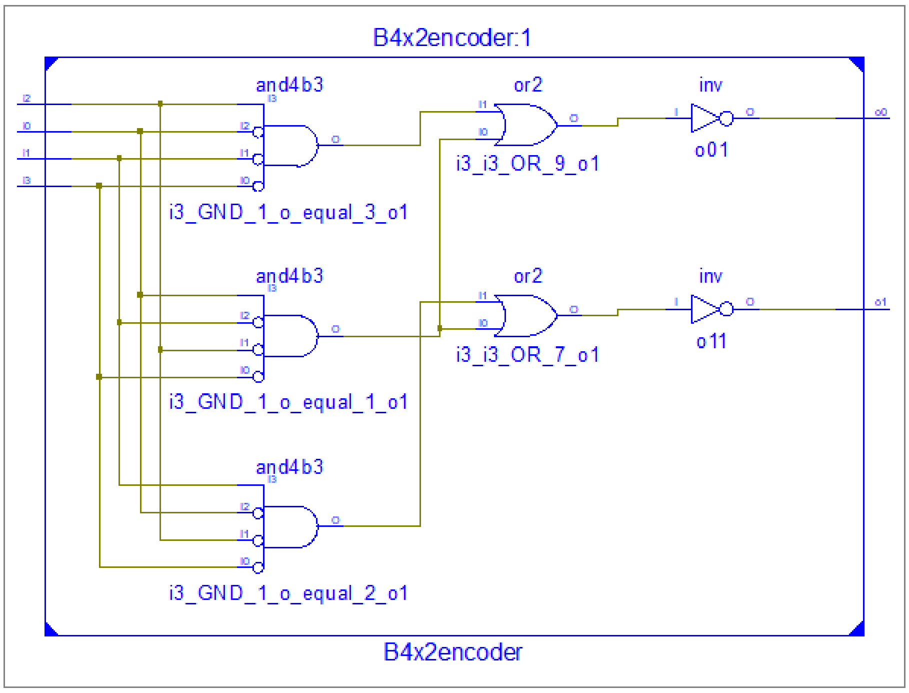
Verilog reports:

<u>4x2 ENCODER</u>
<u>Behavioral Modelling</u>

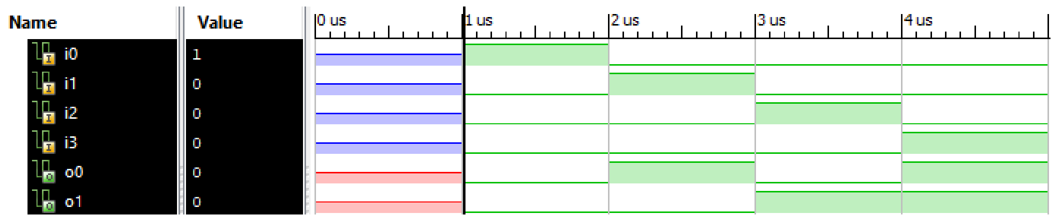
Verilog Program:

```
module B4x2encoder(o0,o1,i3,i2,i1,i0
);
input i3,i2,i1,i0;
output o0,o1;
reg o0,o1;
always@(i3,i2,i1,i0)
begin
case({i3,i2,i1,i0})
4'b0001: {o0,o1}=2'b00;
4'b0010: {o0,o1}=2'b10;
4'b0100: {o0,o1}=2'b01;
4'b1000: {o0,o1}=2'b11;
default: {o0,o1}=2'bxx;
endcase
end
endmodule
```

RTL Schematic:



Simulation:



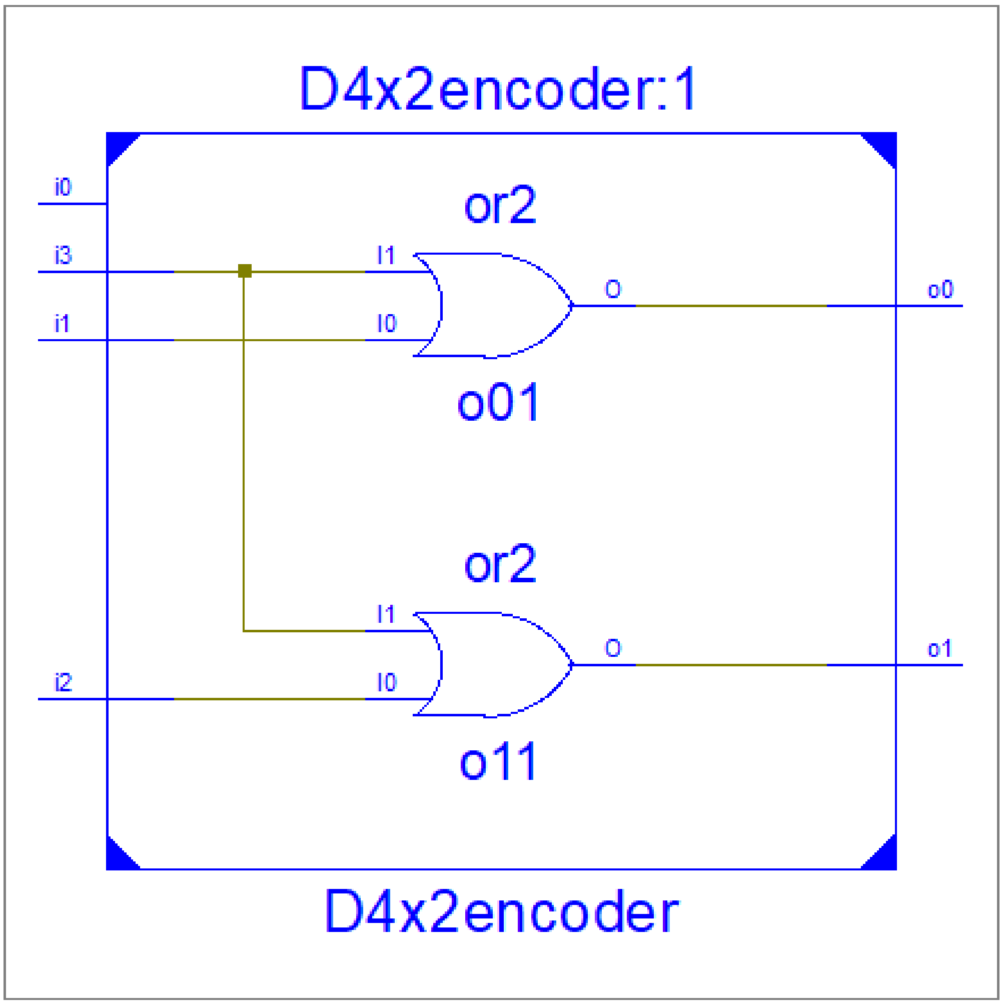
Name	Value
i0	1
i1	0
i2	0
i3	0
o0	0
o1	0

Data flow Modelling

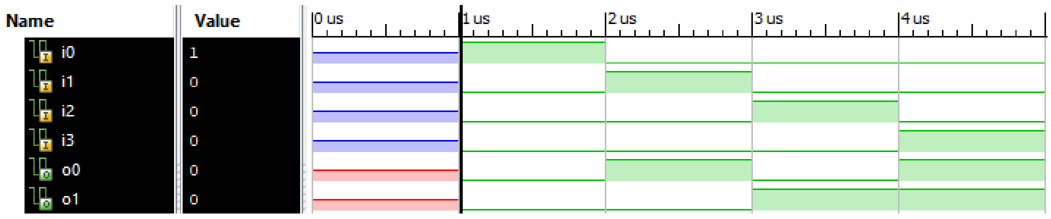
Verilog Program:

```
module D4x2encoder(o0,o1,i0,i1,i2,i3
    );
input i0,i1,i2,i3;
output o0,o1;
wire o0,o1;
assign o0=i1|i3;
assign o1=i2|i3;
endmodule
```

RTL Schematic:



Simulation:

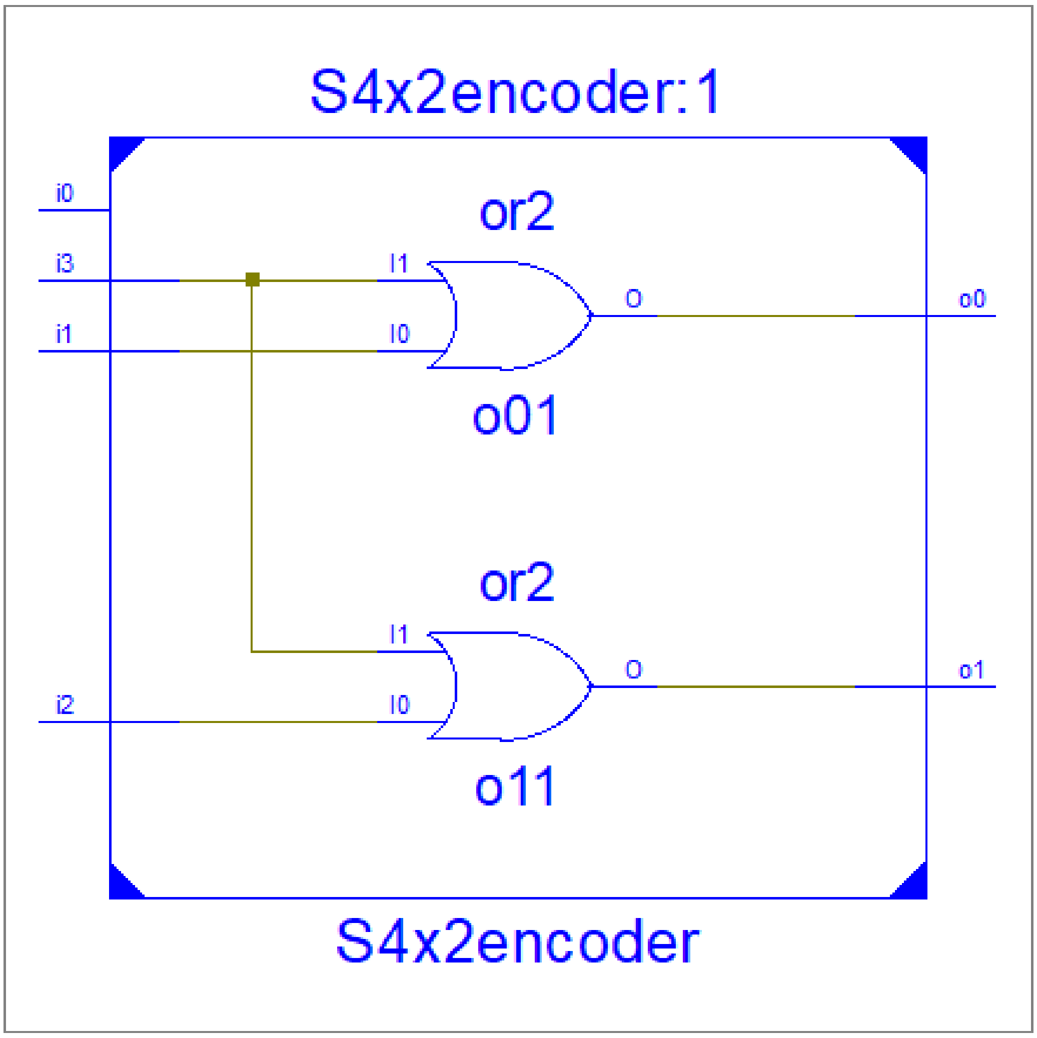


Structural Modelling

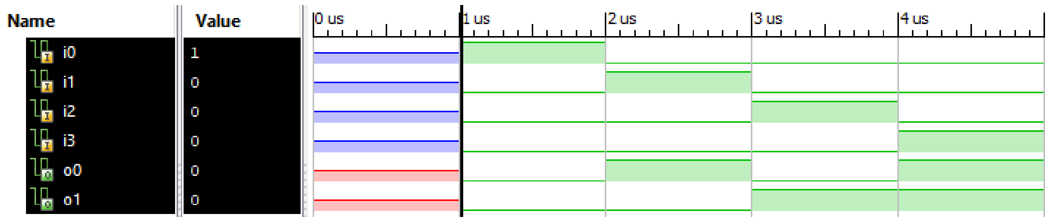
Verilog Program:

```
module S4x2encoder(o0,o1,i0,i1,i2,i3
    );
input i0,i1,i2,i3;
output o0,o1;
wire o0,o1;
or (o0,i1,i3);
or (o1,i2,i3);
endmodule
```

RTL Schematic:



Simulation:



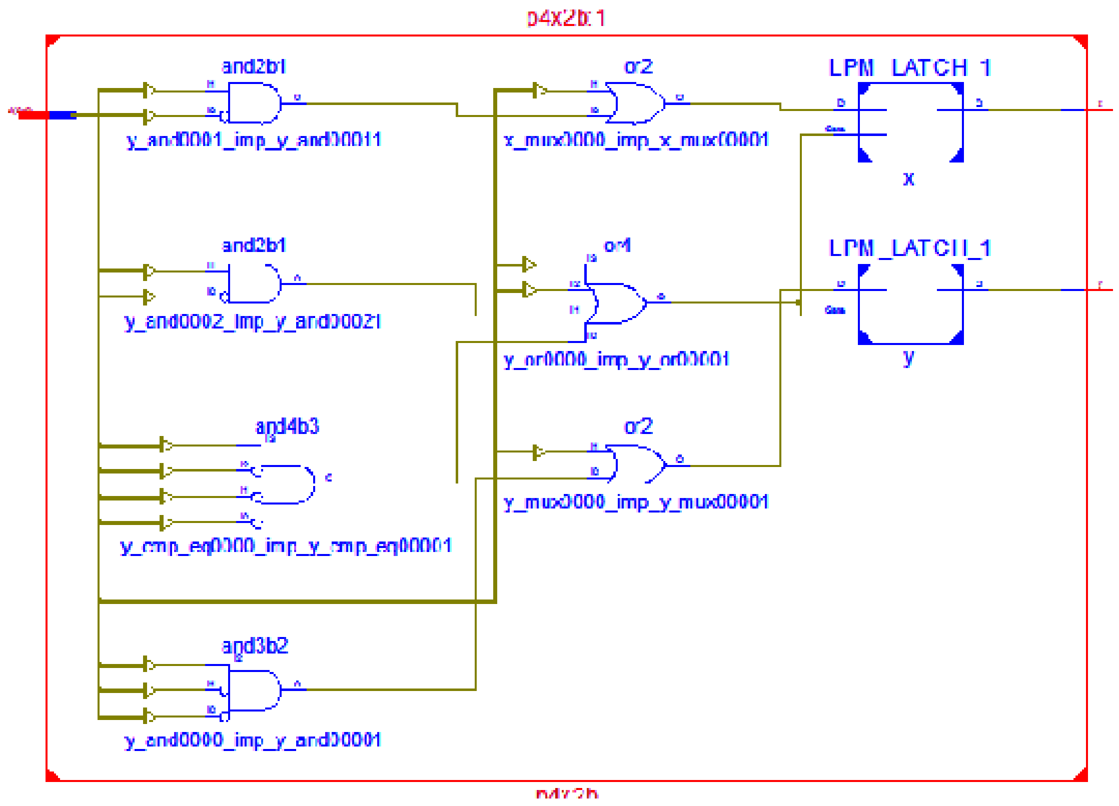
Verilog reports:

4X2 PRIORITY ENCODER
Behavioral Modelling

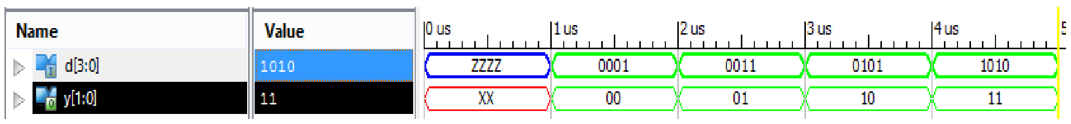
Verilog Program:

```
Module p4x2b(y,d);
input [3:0]d;
output reg [1:0]y;
always @(d)
begin
casex (d)
4'b0001:y=2'b00;
4'b001x:y=2'b01;
4'b01xx:y=2'b10;
4'b1xxx:y=2'b11;
endcase
end
endmodule
```

RTL Schematic:



Simulation:

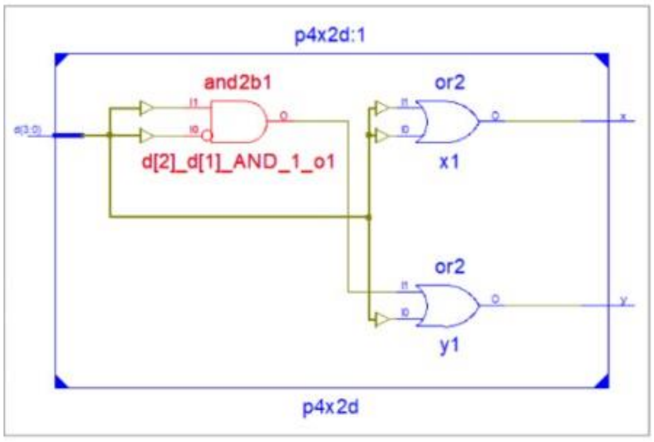


4X2 PRIORITY ENCODER
Data flow Modelling

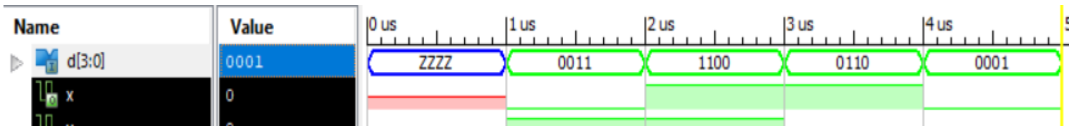
Verilog Program:

```
module p4x2d(x,y,d);
input [3:0]d;
output x,y;
assign x=d[3]|d[2];
assign y=d[3]|(~d[2]&d[1]);
endmodule
```

RTL Schematic:



Simulation:

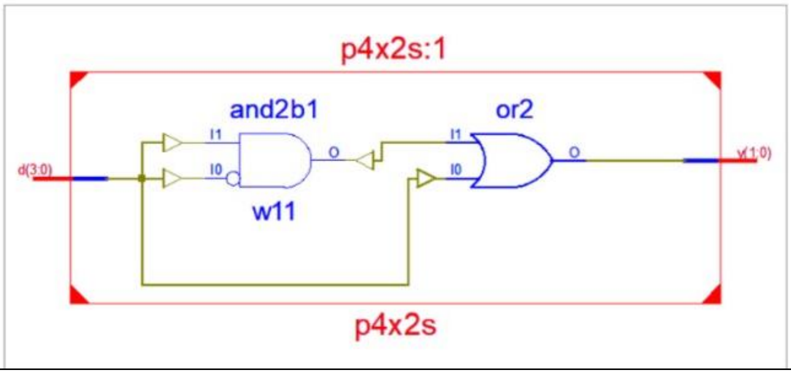


4X2 PRIORITY ENCODER
Structural Modelling

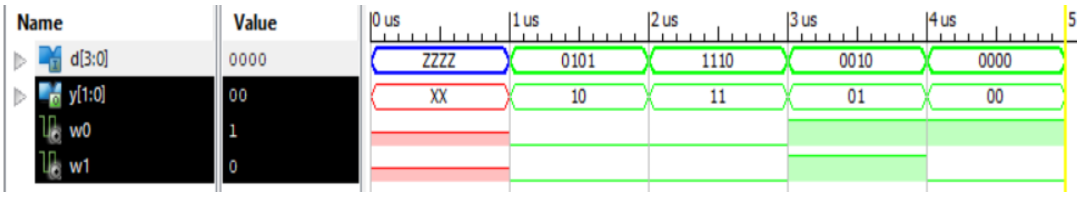
Verilog Program:

```
module p4x2s(y,d);
input [3:0]d;
output [1:0]y;
wire  w0,w1;
not a1(w0,d[2]);
or a2(y[1],d[2],d[3]);
and a3(w1,w0,d[1]);
or a4(y[0],d[3],w1);
endmodule
```

RTL Schematic:



Simulation:



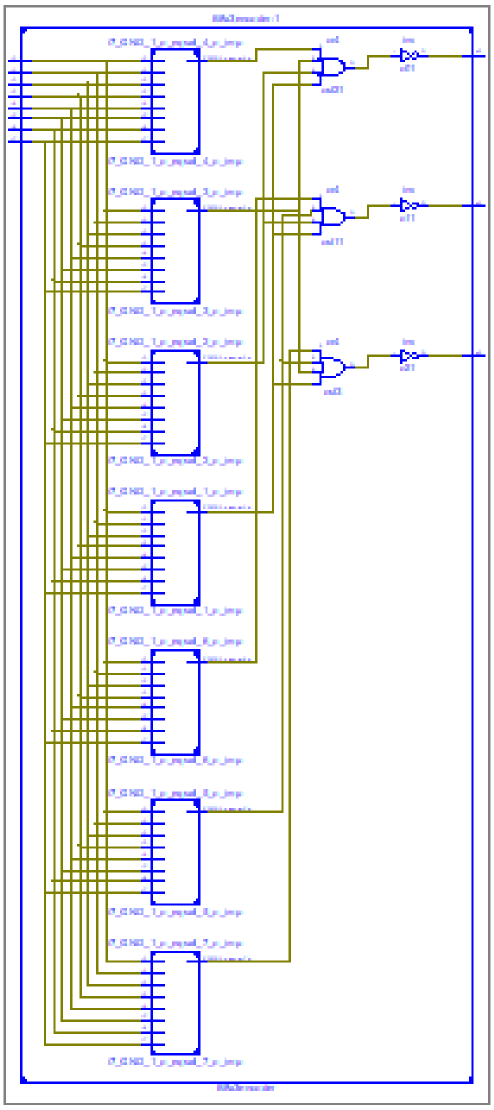
Verilog reports:

8x3 ENCODER
Behavioral Modelling

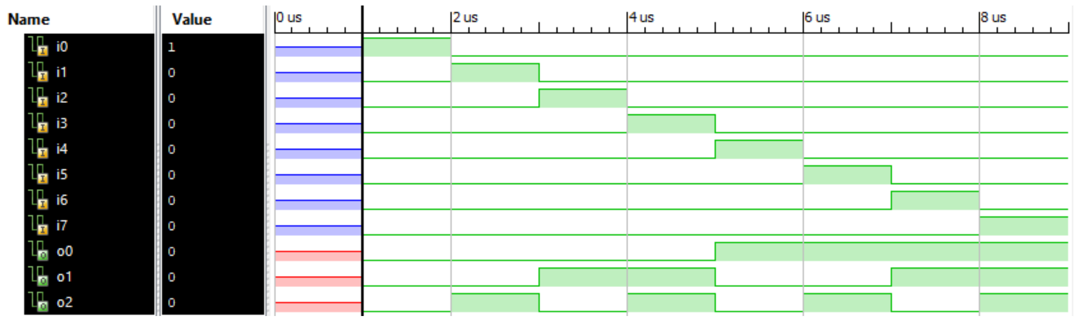
Verilog Program:

```
module
B8x3encoder(o0,o1,o2,i7,i6,i5,i4,i3,i2,i1,i0
);
input i7,i6,i5,i4,i3,i2,i1,i0;
output o0,o1,o2;
reg o0,o1,o2;
always@(i7,i6,i5,i4,i3,i2,i1,i0)
begin
case({i7,i6,i5,i4,i3,i2,i1,i0})
8'b00000001: {o0,o1,o2}=3'b000;
8'b00000010: {o0,o1,o2}=3'b001;
8'b00000100: {o0,o1,o2}=3'b010;
8'b00001000: {o0,o1,o2}=3'b011;
8'b00010000: {o0,o1,o2}=3'b100;
8'b00100000: {o0,o1,o2}=3'b101;
8'b01000000: {o0,o1,o2}=3'b110;
8'b10000000: {o0,o1,o2}=3'b111;
default:     {o0,o1,o2}=3'bxxx;
endcase
end
endmodule
```

RTL Schematic:



Simulation:

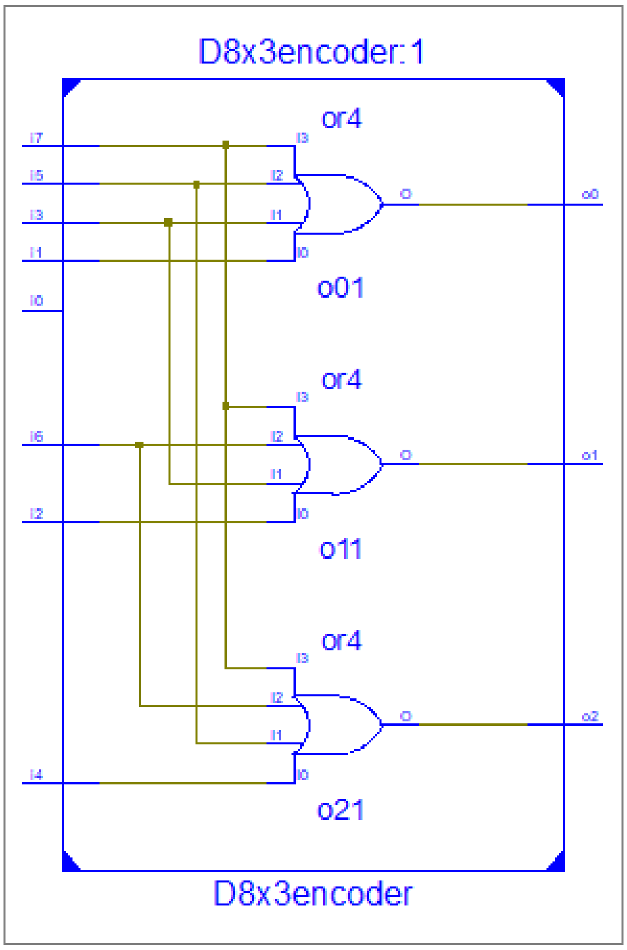


Data flow Modelling

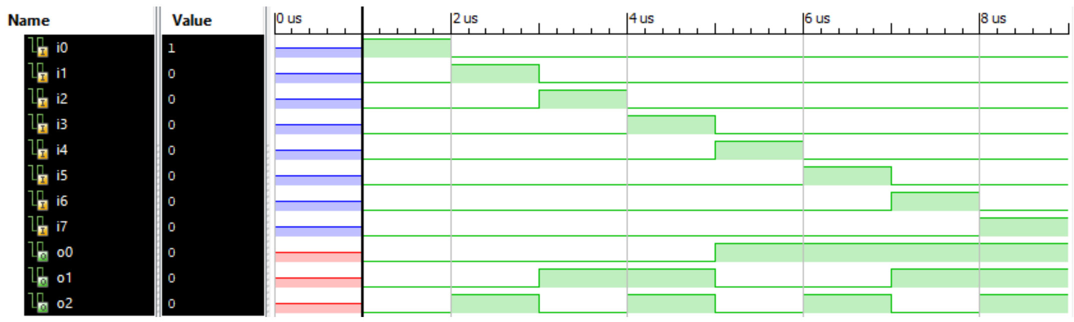
Verilog Program:

```
module D8x3encoder(o0,o1,o2,i0,i1,i2,i3,i4,i5,i6,i7
    );
input i0,i1,i2,i3,i4,i5,i6,i7;
output o0,o1,o2;
wire o0,o1,o2;
assign o0=i1|i3|i5|i7;
assign o1=i2|i3|i6|i7;
assign o2=i4|i5|i6|i7;
endmodule
```

RTL Schematic:



Simulation:

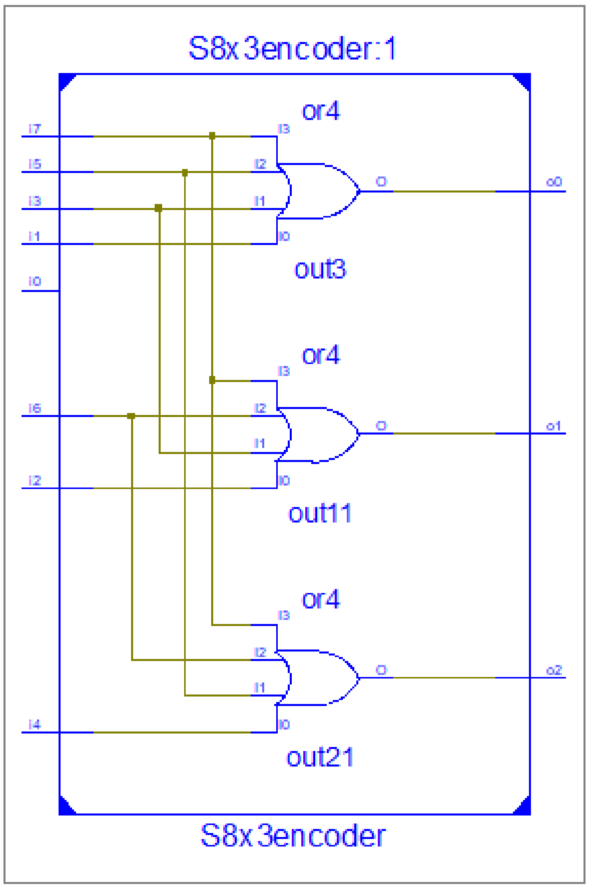


Structural Modelling

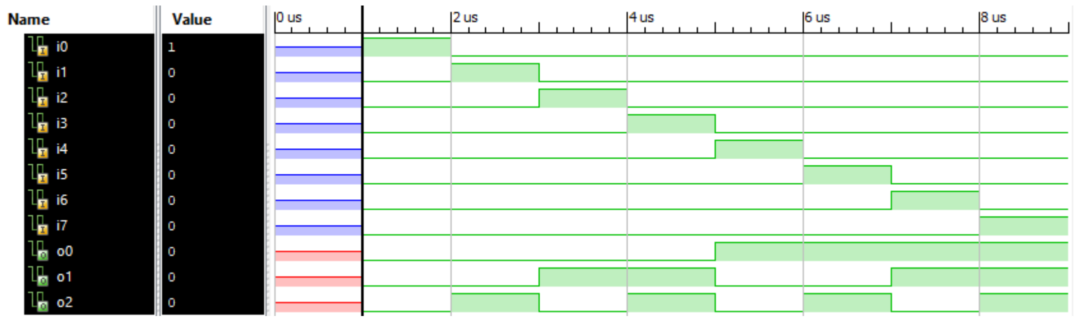
Verilog Program:

```
module S8x3encoder(o0,o1,o2,i0,i1,i2,i3,i4,i5,i6,i7
    );
input i0,i1,i2,i3,i4,i5,i6,i7;
output o0,o1,o2;
wire o0,o1,o2;
or (o0,i1,i3,i5,i7);
or (o1,i2,i3,i6,i7);
or (o2,i4,i5,i6,i7);
endmodule
```

RTL Schematic:



Simulation:



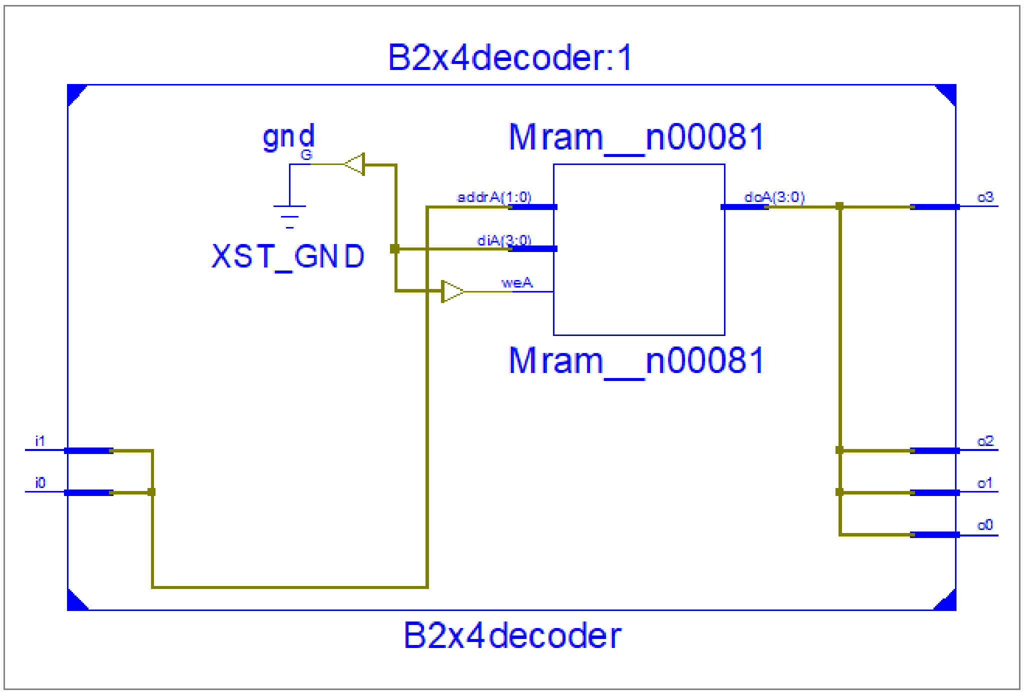
Verilog reports:

2x4 DECODER
Behavioral Modelling

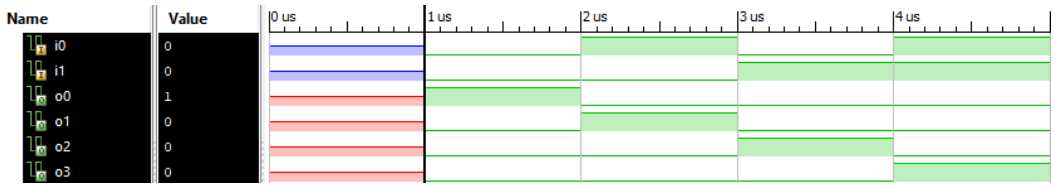
Verilog Program:

```
module B2x4decoder(o3,o2,o1,o0,i1,i0
);
input  i1,i0;
output o3,o2,o1,o0;
reg  o3,o2,o1,o0;
always@(i1,i0)
begin
case({i1,i0})
2'b00: {o3,o2,o1,o0}=4'b0001;
2'b01: {o3,o2,o1,o0}=4'b0010;
2'b10: {o3,o2,o1,o0}=4'b0100;
2'b11: {o3,o2,o1,o0}=4'b1000;
default:{o3,o2,o1,o0}=4'bxxxx;
endcase
end
endmodule
```

RTL Schematic:



Simulation:

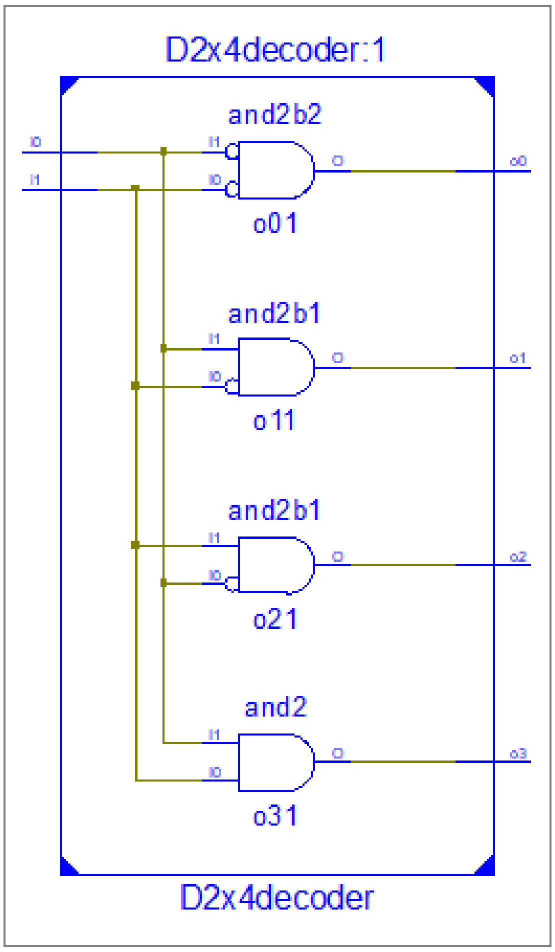


Data flow Modelling

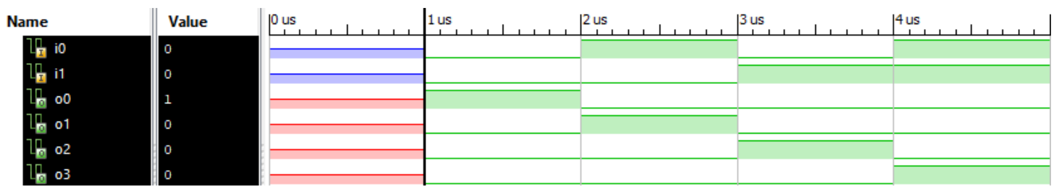
Verilog Program:

```
module D2x4decoder(o0,o1,o2,o3,i0,i1
    );
input i0,i1;
output o0,o1,o2,o3;
wire o0,o1,o2,o3;
assign o0=(~i1)&(~i0);
assign o1=(~i1)&i0;
assign o2=i1&(~i0);
assign o3=i1&i0;
endmodule
```

RTL Schematic:



Simulation:

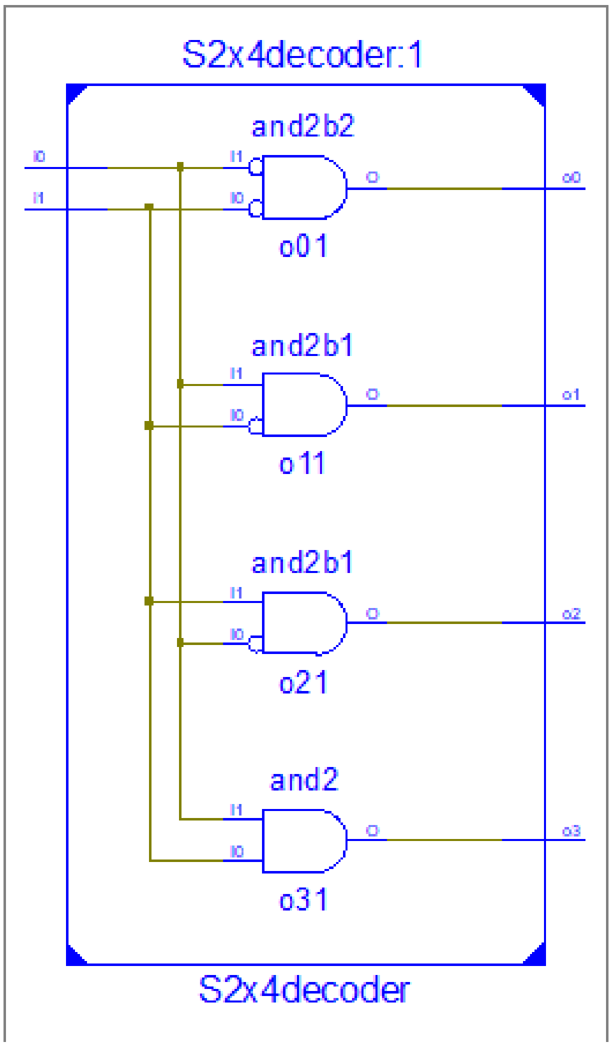


Structural Modelling

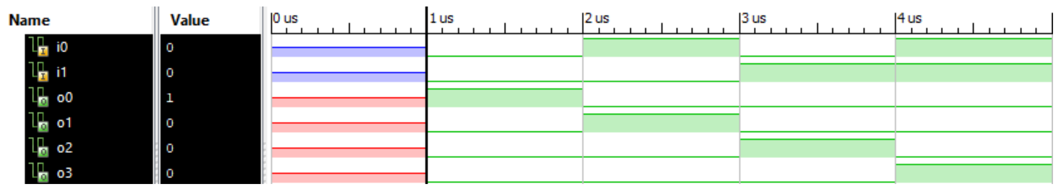
Verilog Program:

```
module S2x4decoder(o0,o1,o2,o3,i0,i1
    );
input i0,i1;
output o0,o1,o2,o3;
wire o0,o1,o2,o3;
and (o0,~i1,~i0);
and (o1,~i1,i0);
and (o2,i1,~i0);
and (o3,i1,i0);
endmodule
```

RTL Schematic:



Simulation:



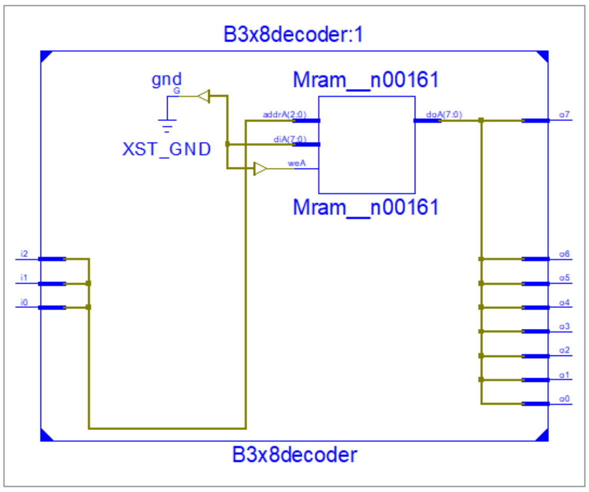
Verilog reports:

3x8 DECODER
Behavioral Modelling

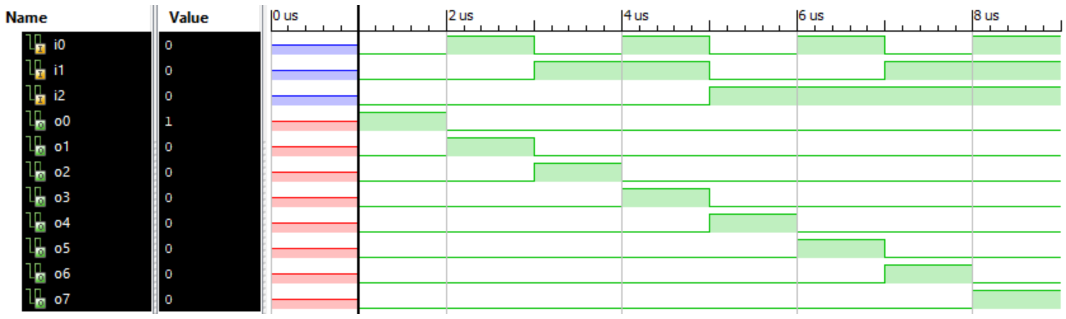
Verilog Program:

```
module B3x8decoder(o7,o6,o5,o4,o3,o2,o1,o0,i2,i1,i0);
input i2,i1,i0;
output o7,o6,o5,o4,o3,o2,o1,o0;
reg  o7,o6,o5,o4,o3,o2,o1,o0;
always@(i2,i1,i0)
begin
case({i2,i1,i0})
3'b000: {o7,o6,o5,o4,o3,o2,o1,o0}=8'b00000001;
3'b001: {o7,o6,o5,o4,o3,o2,o1,o0}=8'b00000010;
3'b010: {o7,o6,o5,o4,o3,o2,o1,o0}=8'b00000100;
3'b011: {o7,o6,o5,o4,o3,o2,o1,o0}=8'b00001000;
3'b100: {o7,o6,o5,o4,o3,o2,o1,o0}=8'b00010000;
3'b101: {o7,o6,o5,o4,o3,o2,o1,o0}=8'b00100000;
3'b110: {o7,o6,o5,o4,o3,o2,o1,o0}=8'b01000000;
3'b111: {o7,o6,o5,o4,o3,o2,o1,o0}=8'b10000000;
default: {o7,o6,o5,o4,o3,o2,o1,o0}=8'bxxxxxxxx;
endcase
end
endmodule
```

RTL Schematic:



Simulation:

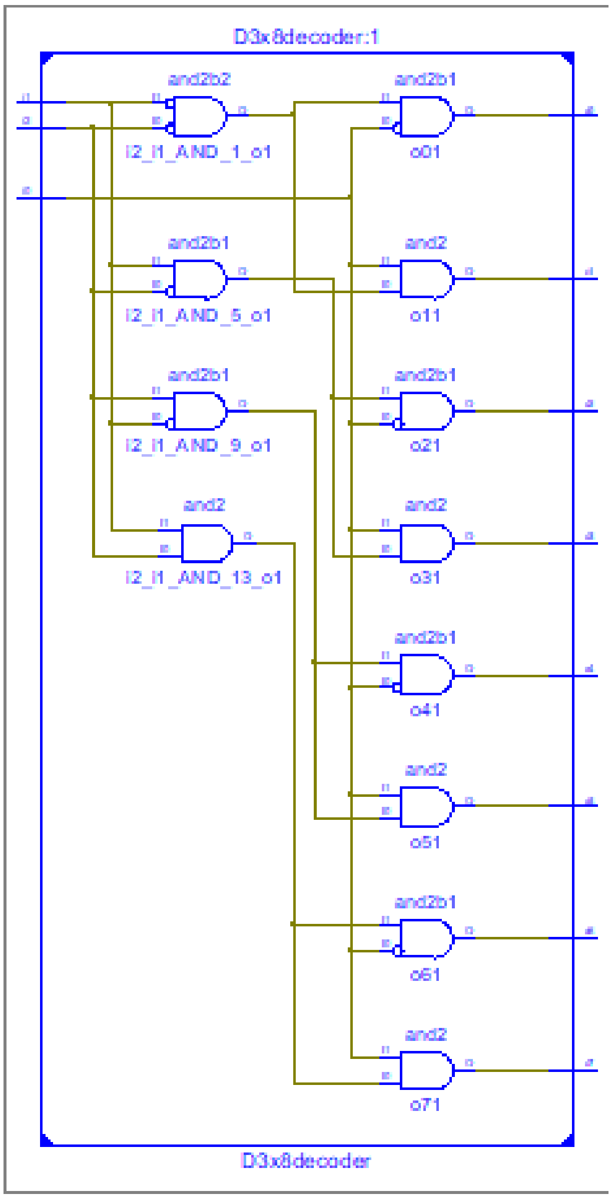


Data flow Modelling

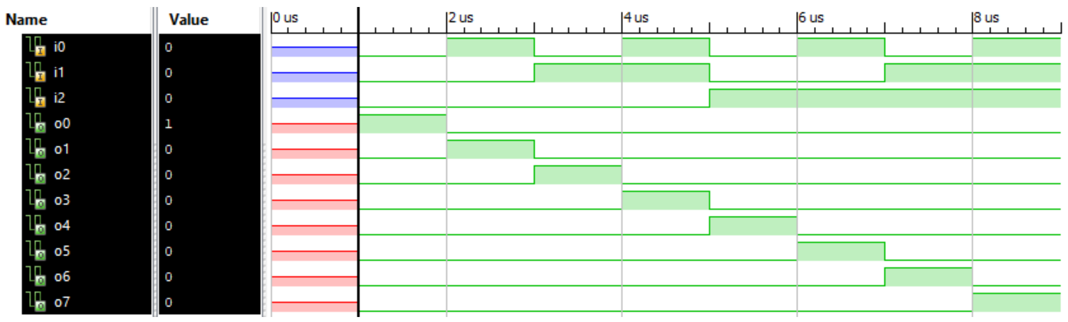
Verilog Program:

```
module
D3x8decoder(o0,o1,o2,o3,o4,o5,
o6,o7,i0,i1,i2);
input  i0,i1,i2;
output o0,o1,o2,o3,o4,o5,o6,o7;
wire   o0,o1,o2,o3,o4,o5,o6,o7;
assign o0=(~i2)&(~i1)&(~i0);
assign o1=(~i2)&(~i1)&(i0);
assign o2=(~i2)&(i1)&(~i0);
assign o3=(~i2)&(i1)&(i0);
assign o4=(i2)&(~i1)&(~i0);
assign o5=(i2)&(~i1)&(i0);
assign o6=(i2)&(i1)&(~i0);
assign o7=(i2)&(i1)&(i0);
endmodule
```

RTL Schematic:



Simulation:

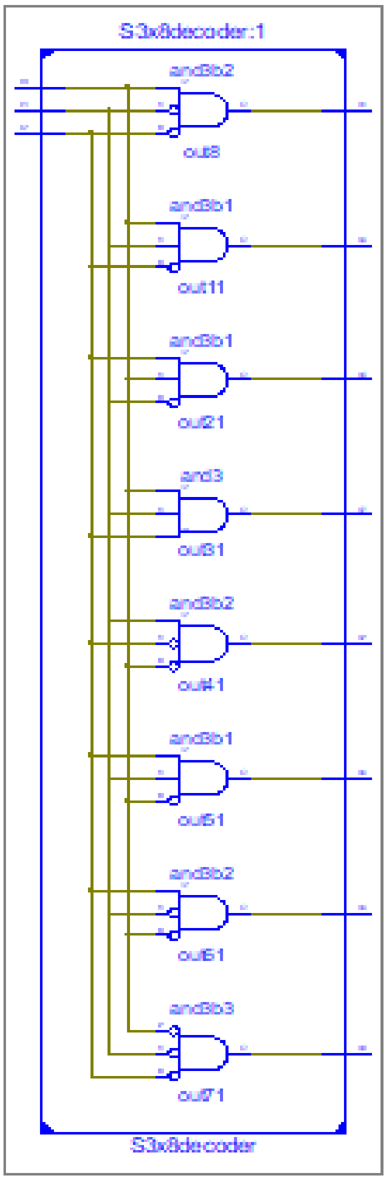


Structural Modelling

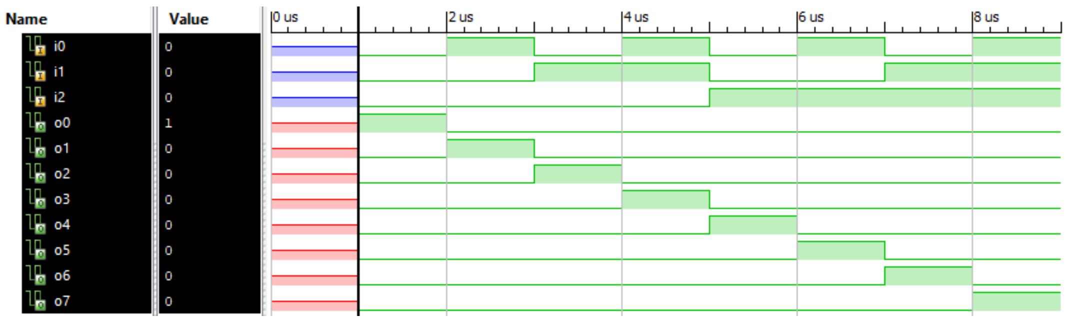
Verilog Program:

```
module S3x8decoder(o0,o1,o2,o3,o4,o5,o6,
o7,i0,i1,i2 );
input i0,i1,i2;
output o0,o1,o2,o3,o4,o5,o6,o7;
wire o0,o1,o2,o3,o4,o5,o6,o7;
and (o0,~i2,~i1,~i0);
and (o1,~i2,~i1,i0);
and (o2,~i2,i1,~i0);
and (o3,~i2,i1,i0);
and (o4,i2,~i1,~i0);
and (o5,i2,~i1,i0);
and (o6,i2,i1,~i0);
and (o7,i2,i1,i0);
endmodule
```

RTL Schematic:



Simulation:



Result:

Designed and implemented encoders and decoders using Behavioral, dataflow, structural modelling using **Xilinx ISE 14.2**

EXPERIMENT-6
COMPARATORS

Aim: Design, simulate and implement comparators using Xilinx ISE.
Software Used: Xilinx **ISE-14.2**
Simulator Used: iSim
Synthesizer Used: XST
Procedure:

- Check the syntax of the program for any errors if any correct and verify again.
- In the process window, put the simulation mode in behavioral model, take a Verilog text fixture and give the input combinations.
- Perform simulation to verify the functionality and logic of the code.
- In the process window, change it to implement design highlight program in the source window, performs the run operations to implement the design.
- To view RTL and Technology schematic.

Verilog reports:

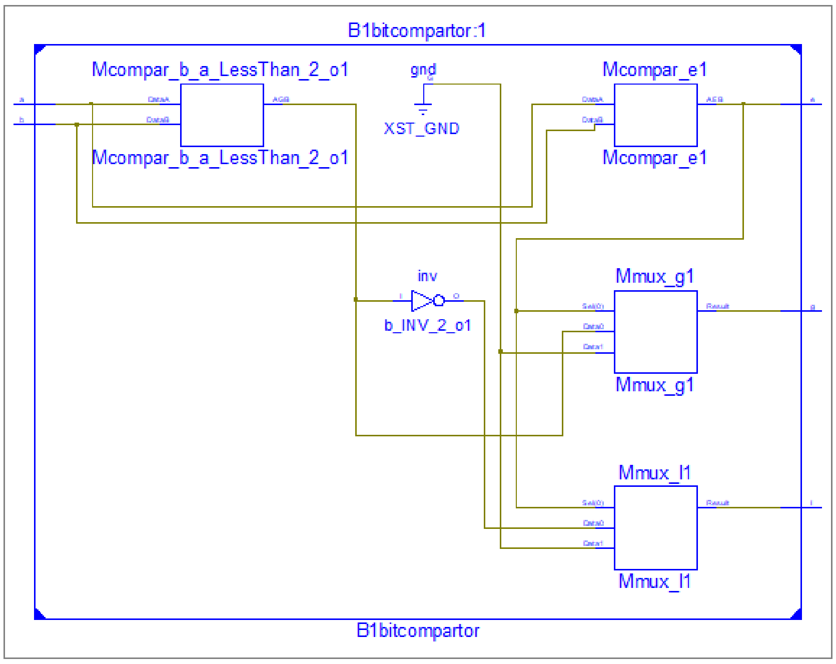
1 BIT COMPARATOR

Behavioral Modelling

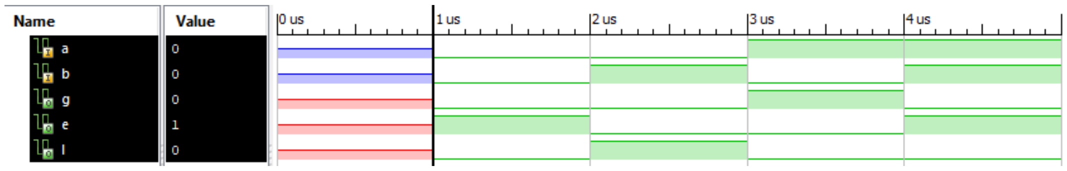
Verilog Program:

```
module B1bitcompartor(g,e,l,a,b);
input a,b;
output g,e,l;
reg g,e,l;
always@(a,b)
begin
if(a==b)
{g,e,l}=3'b010;
else if(a>b)
{g,e,l}=3'b100;
else if(a<b)
{g,e,l}=3'b001;
else
{g,e,l}=3'bxxx;
end
endmodule
```

RTL Schematic:



Simulation:

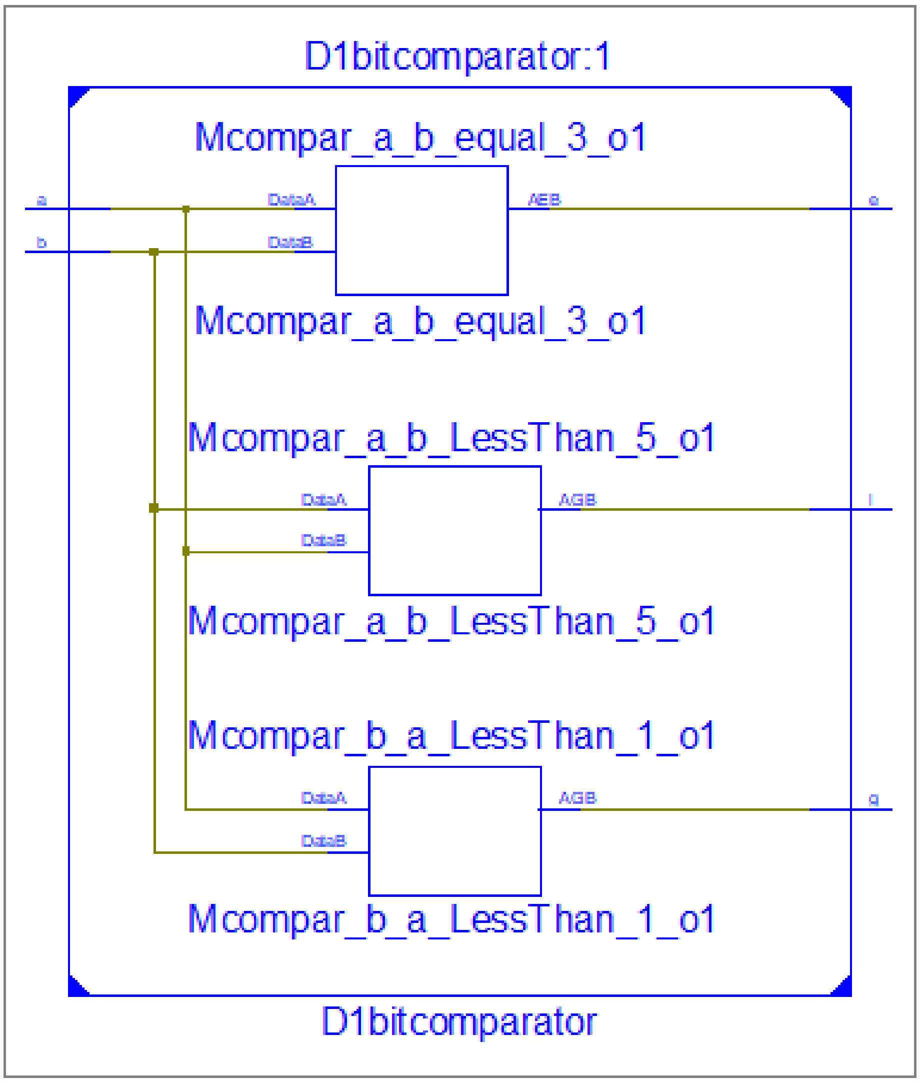


Data flow Modelling

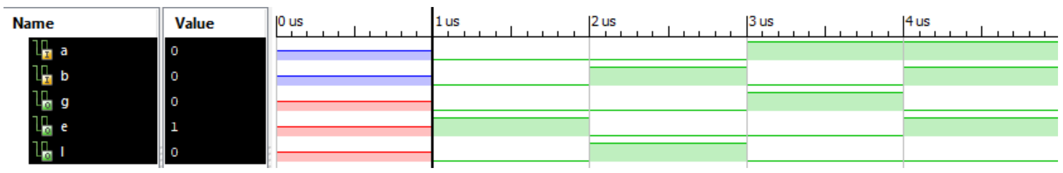
Verilog Program:

```
module D1bitcomparator(g,e,l,a,b
    );
input a,b;
output g,e,l;
wire g,e,l;
assign g=a>b?1:0;
assign e=a==b?1:0;
assign l=a<b?1:0;
endmodule
```

RTL Schematic:



Simulation:

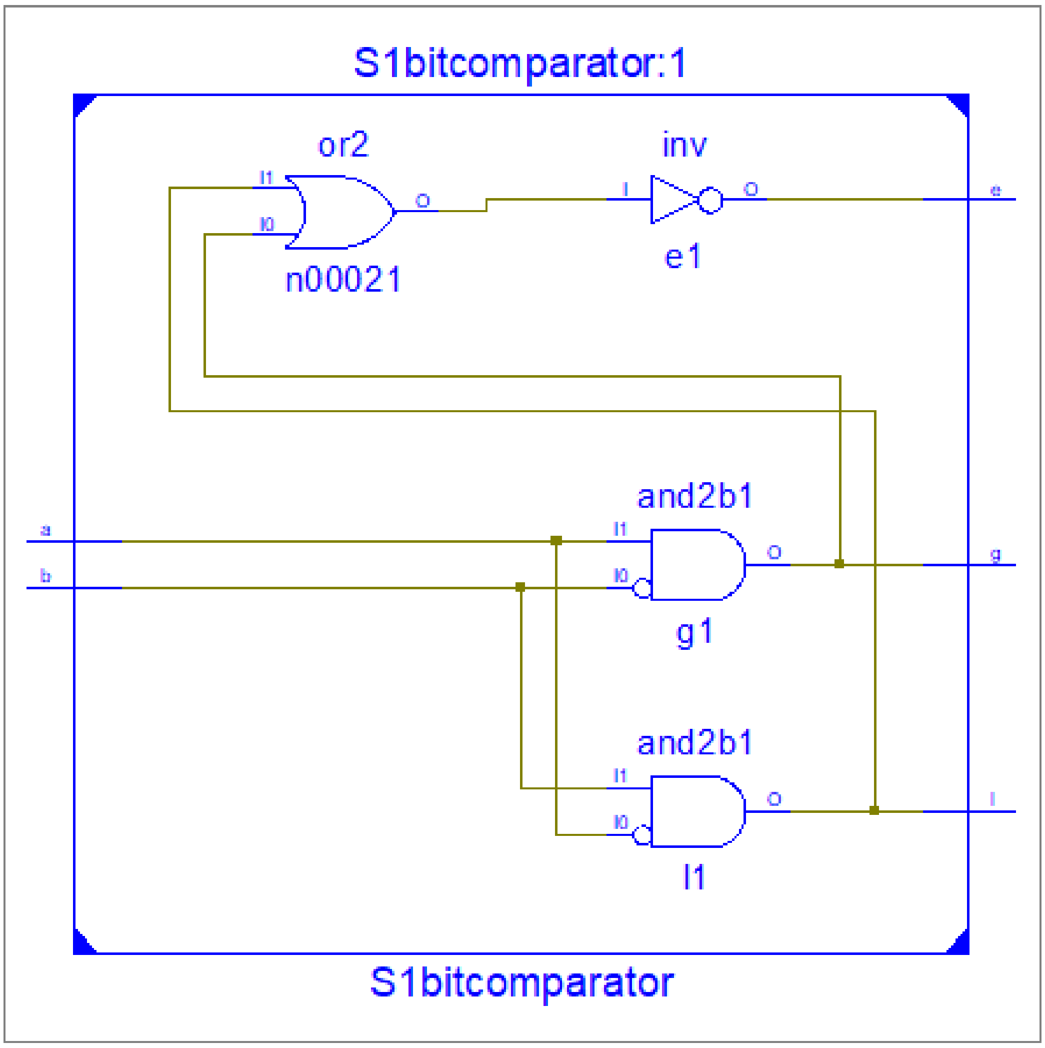


Structural Modelling

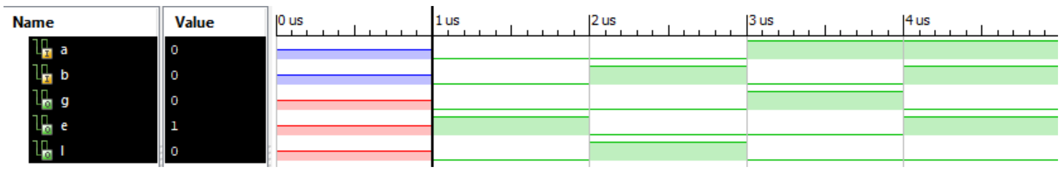
Verilog Program:

```
module S1bitcomparator(g,e,l,a,b
    );
input a,b;
output g,e,l;
wire g,e,l;
and a1(g,a,~b);
nor a3(e,g,l);
and a2(l,~a,b);
endmodule
```

RTL Schematic:



Simulation:



Verilog reports:

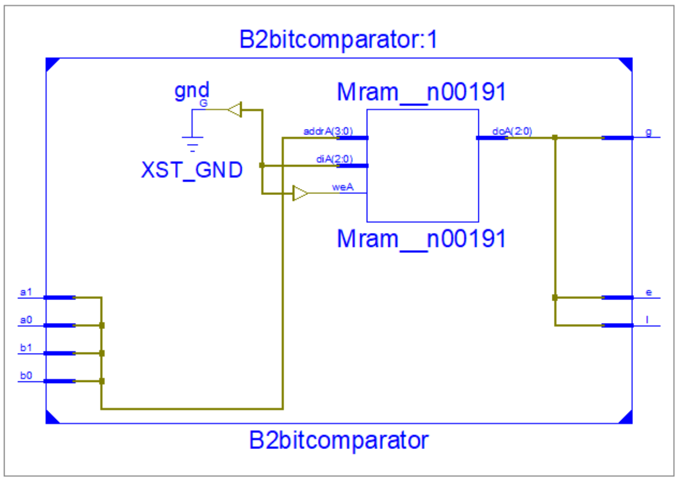
2 BIT COMPARATOR

Behavioral Modelling

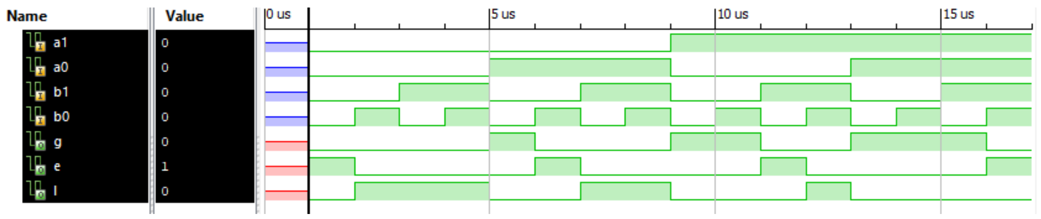
Verilog Program:

```
module
B2bitcomparator(g,e,l,a1,a0,b1,
b0
    );
input a1,a0,b1,b0;
output g,e,l;
reg g,e,l;
always@(a1,a0,b1,b0)
begin
case({a1,a0,b1,b0})
4'b0000: {g,e,l}=3'b010;
4'b0001: {g,e,l}=3'b001;
4'b0010: {g,e,l}=3'b001;
4'b0011: {g,e,l}=3'b001;
4'b0100: {g,e,l}=3'b100;
4'b0101: {g,e,l}=3'b010;
4'b0110: {g,e,l}=3'b001;
4'b0111: {g,e,l}=3'b001;
4'b1000: {g,e,l}=3'b100;
4'b1001: {g,e,l}=3'b100;
4'b1010: {g,e,l}=3'b010;
4'b1011: {g,e,l}=3'b001;
4'b1100: {g,e,l}=3'b100;
4'b1101: {g,e,l}=3'b100;
4'b1110: {g,e,l}=3'b100;
4'b1111: {g,e,l}=3'b010;
default: {g,e,l}=3'bxxx;
endcase
end
endmodule
```

RTL Schematic:



Simulation:

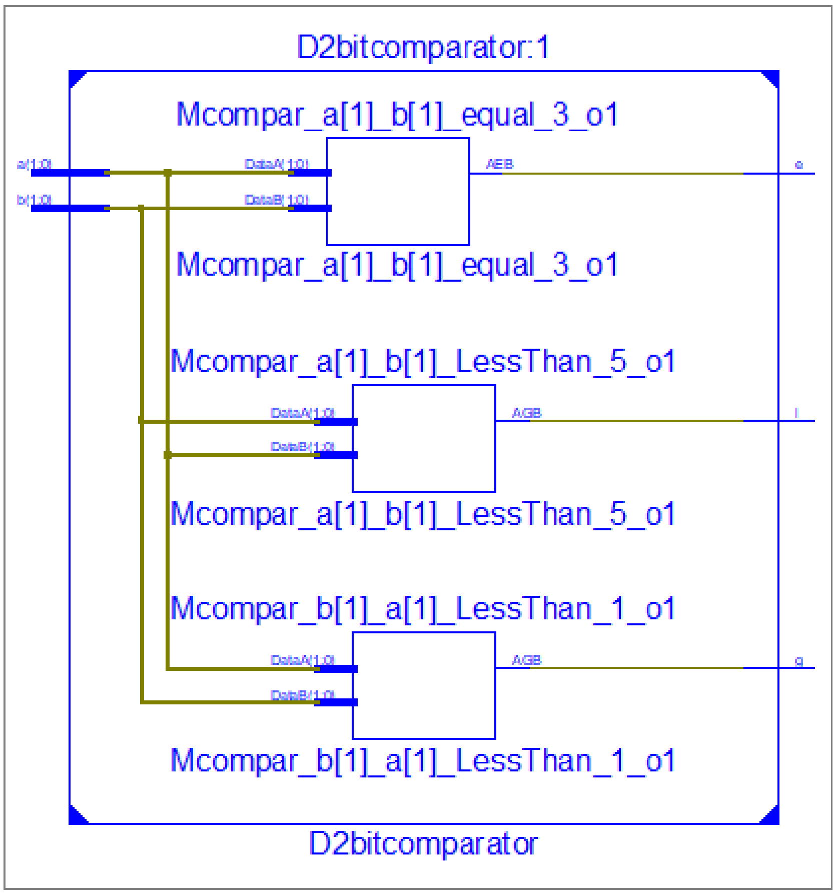


Data flow Modelling

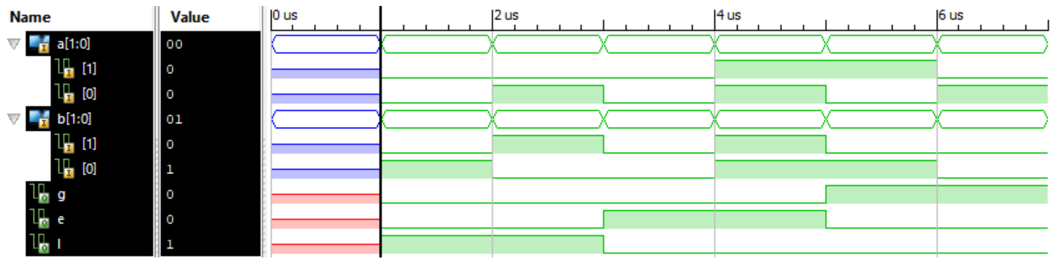
Verilog Program:

```
module D2bitcomparator(g,e,l,a,b
    );
input [1:0]a,b;
output g,e,l;
wire g,e,l;
assign g=a[1:0]>b[1:0]?1:0;
assign e=a[1:0]==b[1:0]?1:0;
assign l=a[1:0]<b[1:0]?1:0;
endmodule
```

RTL Schematic:


D2bitcomparator:1
Mcompar_a[1]_b[1]_equal_3_o1
a[1:0]
b[1:0]
DataA[1:0]
DataB[1:0]
AEB
e
Mcompar_a[1]_b[1]_equal_3_o1
Mcompar_a[1]_b[1]_LessThan_5_o1
DataA[1:0]
DataB[1:0]
AGB
l
Mcompar_a[1]_b[1]_LessThan_5_o1
Mcompar_b[1]_a[1]_LessThan_1_o1
DataA[1:0]
DataB[1:0]
AGB
g
Mcompar_b[1]_a[1]_LessThan_1_o1
D2bitcomparator

Simulation:

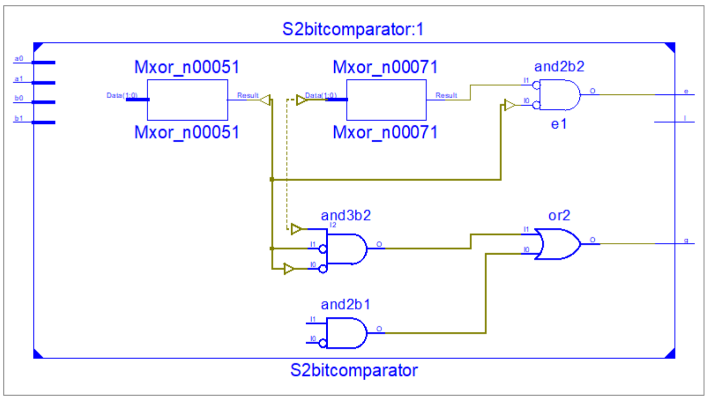


Structural Modelling

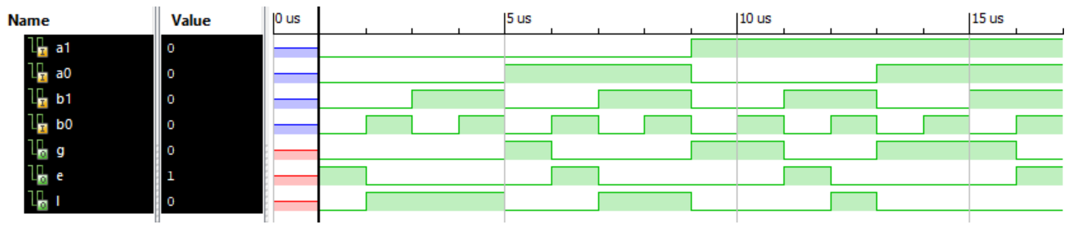
Verilog Program:

```
module S2bitcomparator(g,e,l,a1,a0,b1,b0
    );
input a1,a0,b1,b0;
output g,e,l;
wire g,g0,g1,g2,e,e0,e1,l,l0,l1,l2;

and (g0,a1,~b1);
and (g1,e1,a0,~b0);
or  (g,g0,g1);

xnor(e1,a1,b1);
xnor(e0,a0,b0);
and (e,e1,e0);

and (l0,~a1,b1);
and (l1,e1,~a0,b0);
or  (l,l0,l1);

endmodule
```

RTL Schematic:



Simulation:



Result:

Designed and implemented comparators using Behavioral, dataflow, structural modelling using **Xilinx ISE 14.2**

EXPERIMENT-7
PARITY GENERATOR AND CHECKER

Aim: Design, simulate and implement parity generator and checker using Xilinx ISE.
Software Used: Xilinx **ISE-14.2**
Simulator Used: iSim
Synthesizer Used: XST
Procedure:

- Check the syntax of the program for any errors if any correct and verify again.
- In the process window, put the simulation mode in behavioral model, take a Verilog text fixture and give the input combinations.
- Perform simulation to verify the functionality and logic of the code.
- In the process window, change it to implement design highlight program in the source window, performs the run operations to implement the design.
- To view RTL and Technology schematic.

Verilog reports:

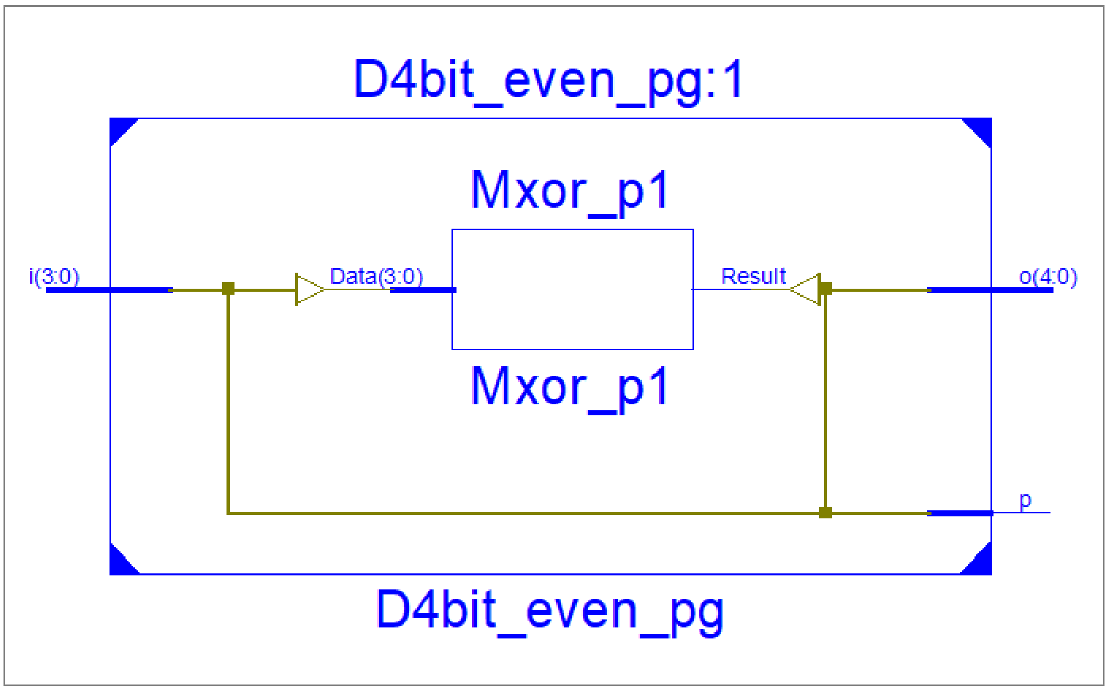
4 BIT EVEN PARITY GENERATOR

Data flow Modelling

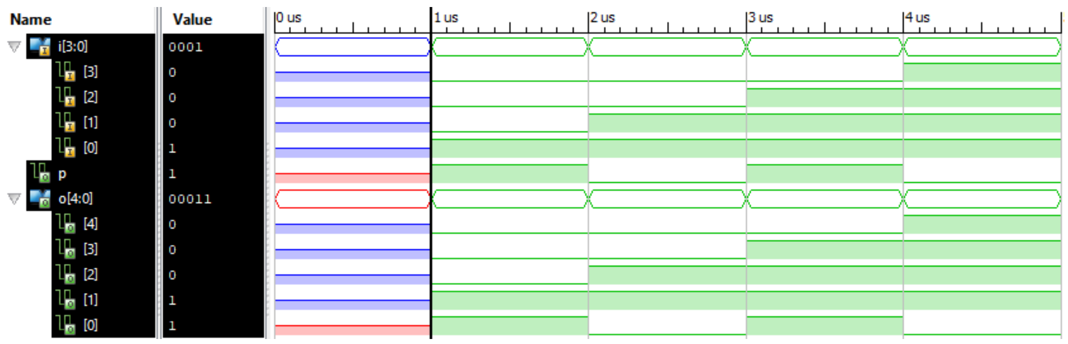
Verilog Program:

```
module D4bit_even_pg(o,p,i);
input [3:0]i;
output p;
output [4:0]o;
assign p=i[0]^i[1]^i[2]^i[3];
assign o={i,p};
endmodule
```

RTL Schematic:



Simulation:



Verilog reports:

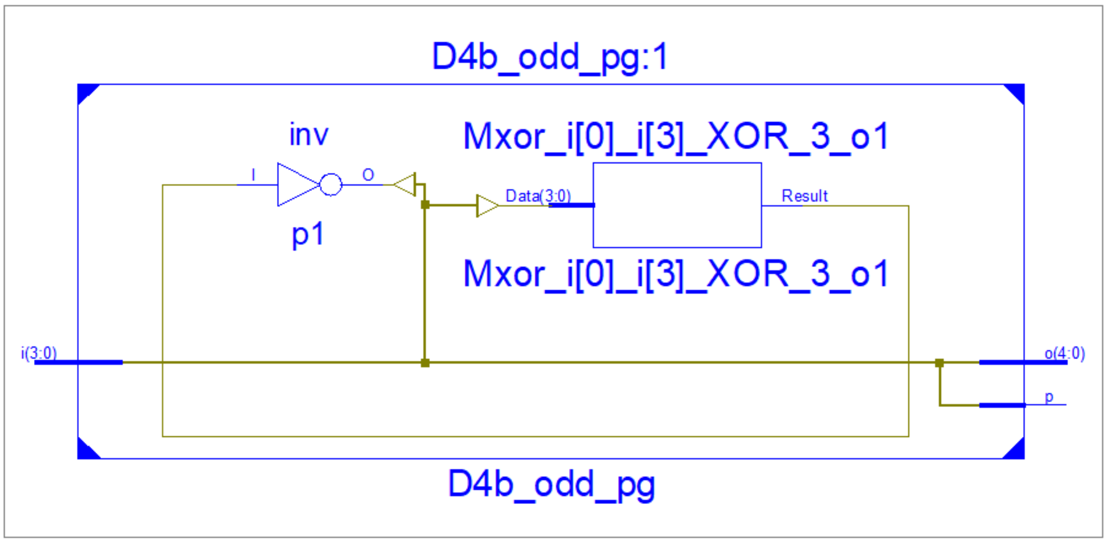
4 BIT ODD PARITY GENERATOR

Data flow Modelling

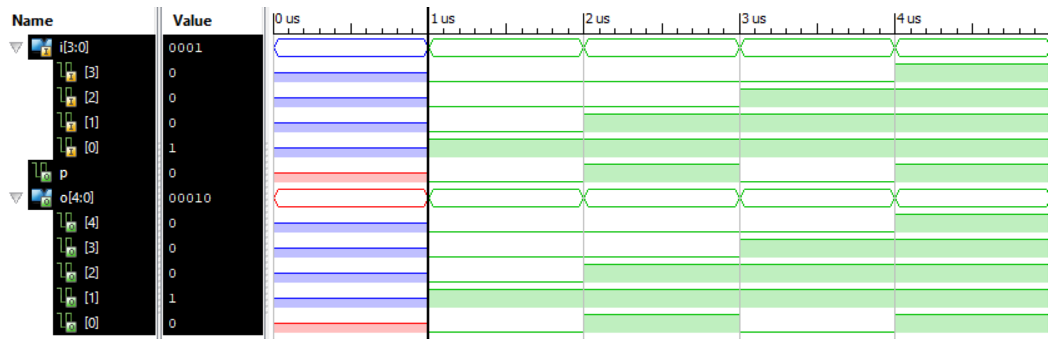
Verilog Program:

```
module D4b_odd_pg(o,p,i);
input [3:0]i;
output p;
output [4:0]o;
assign p=~(i[0]^i[1]^i[2]^i[3]);
assign o={i,p};
endmodule
```

RTL Schematic:



Simulation:



Verilog reports:

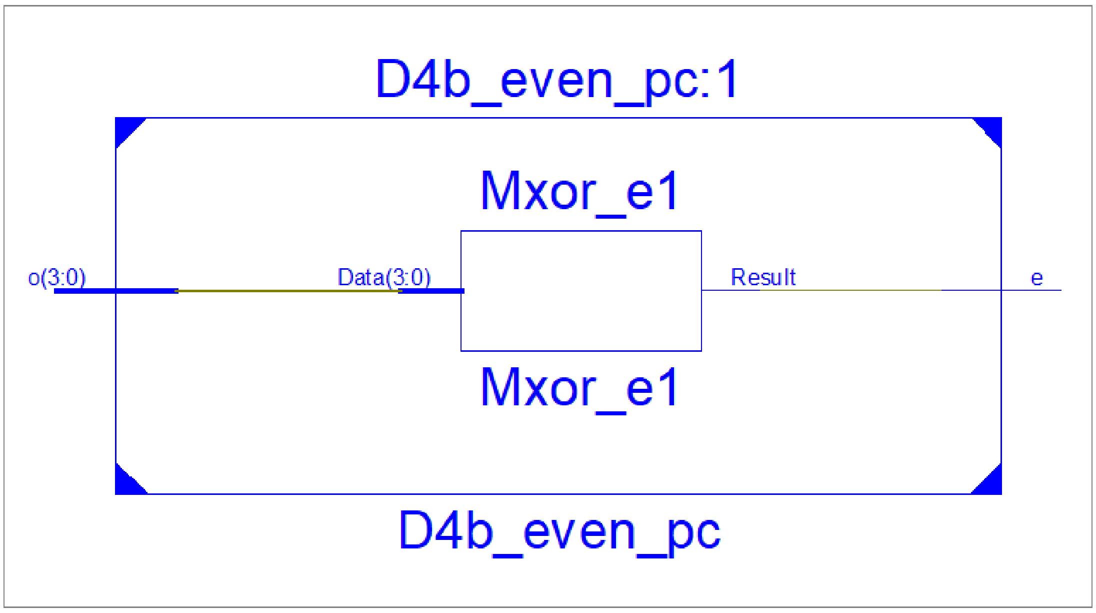
4 BIT EVEN PARITY CHECKER

Data flow Modelling

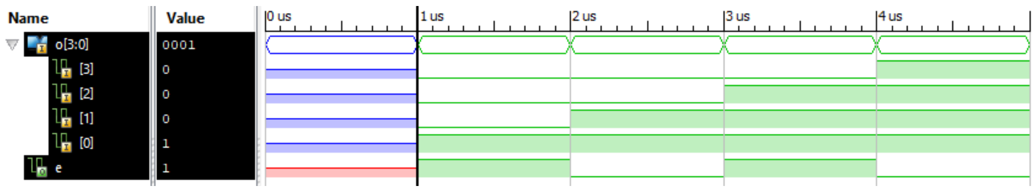
Verilog Program:

```
module D4b_even_pc(e,o);
input [3:0]o;
output wire e;
assign e=o[0]^o[1]^o[2]^o[3];
endmodule
```

RTL Schematic:



Simulation:



Verilog reports:

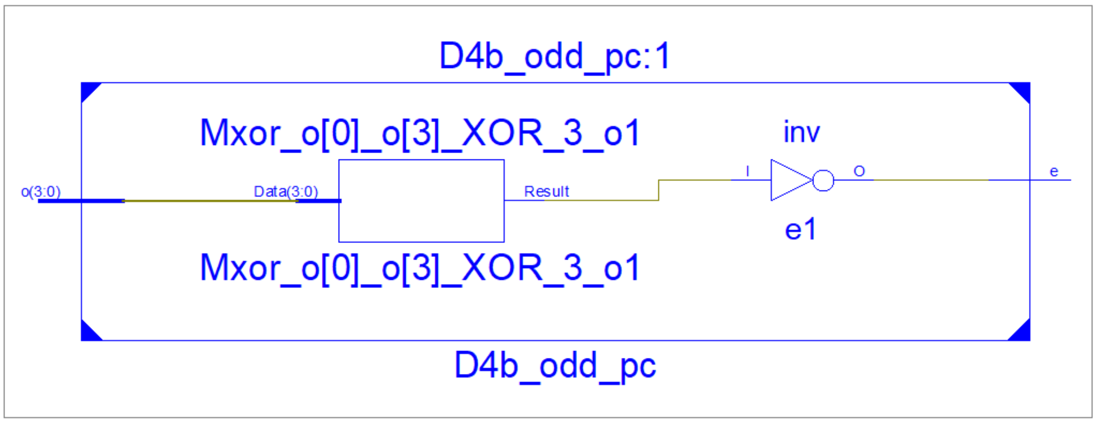
4 BIT ODD PARITY CHECKER

Data flow Modelling

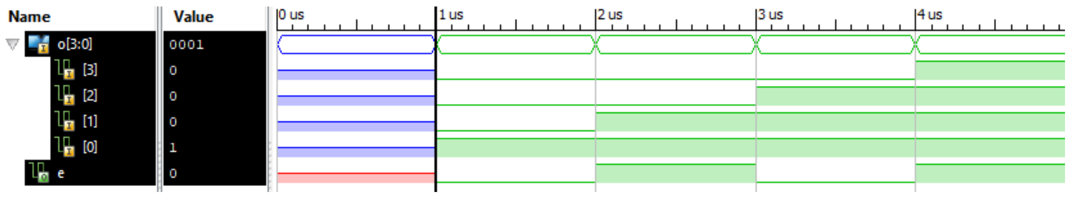
Verilog Program:

```
module D4b_odd_pc(e,o);
input [3:0]o;
output wire e;
assign e=~(o[0]^o[1]^o[2]^o[3]);
endmodule
```

RTL Schematic:



Simulation:



Result:

Designed and implemented odd and even parity generator and checker using Behavioral,dataflow,structural modelling using **Xilinx ISE 14.2**

EXPERIMENT-8
FLIP FLOPS

Aim: Design, simulate and implement flipflops using Xilinx ISE.
Software Used: Xilinx **ISE-14.2**
Simulator Used: iSim
Synthesizer Used: XST
Procedure:

- Check the syntax of the program for any errors if any correct and verify again.
- In the process window, put the simulation mode in behavioral model, take a Verilog text fixture and give the input combinations.
- Perform simulation to verify the functionality and logic of the code.
- In the process window, change it to implement design highlight program in the source window, performs the run operations to implement the design.
- To view RTL and Technology schematic.

Verilog reports:

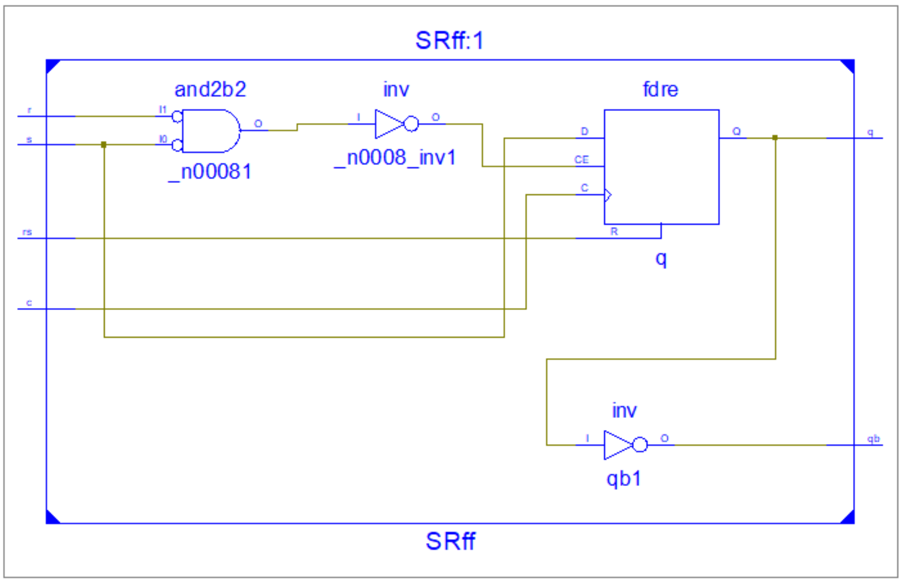
SR FLIP FLOP

Behavioral Modelling

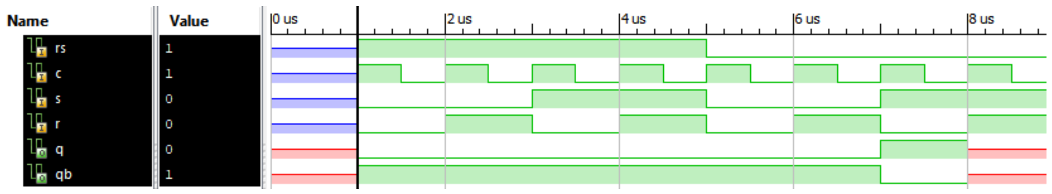
Verilog Program:

```
module SRff(q,qb,s,r,c,rs
    );
input s,r,c,rs;
output qb,q;
reg q;
assign qb=~q;
always@(posedge c)
begin
if(rs)
q=0;
else
casex({s,r})
2'b00:q=q;
2'b01:q=0;
2'b10:q=1;
2'b11:q=1'bx;
endcase
end
endmodule
```

RTL Schematic:



Simulation:



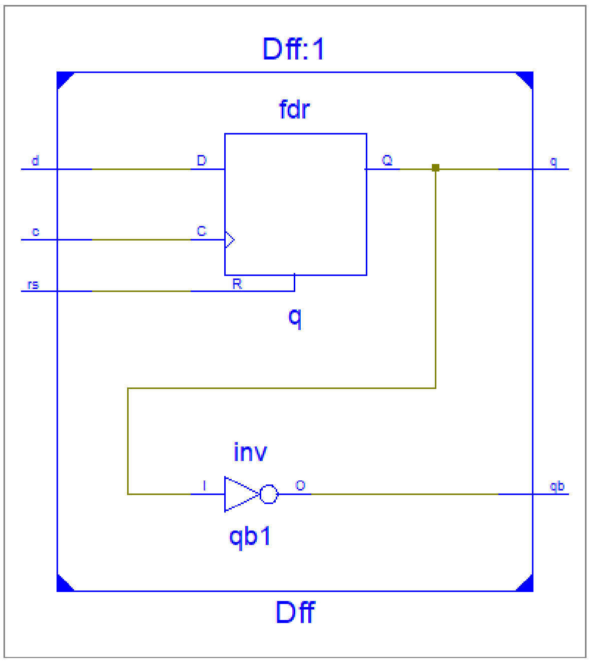
D FLIP FLOP

Behavioral Modelling

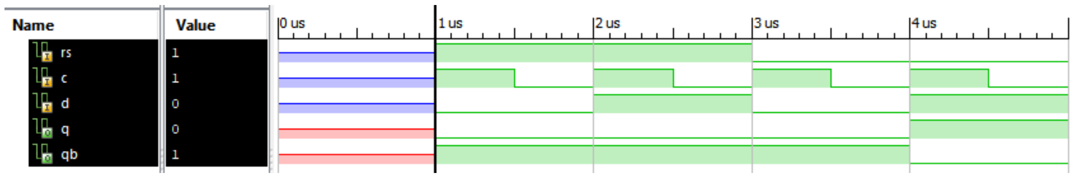
Verilog Program:

```
module Dff(q,qb,d,c,rs
    );
input d,c,rs;
output qb,q;
reg q;
assign qb=~q;
always@(posedge c)
begin
if(rs)
q=0;
else
casex(d)
1'b0:q=1'b0;
1'b1:q=1'b1;
endcase
end
endmodule
```

RTL Schematic:



Simulation:



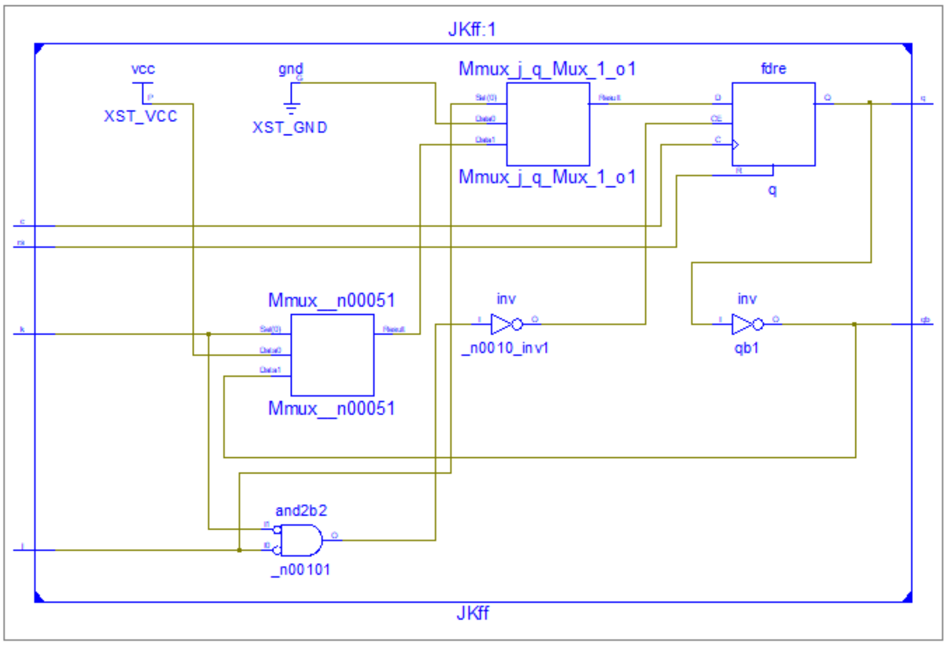
JK FLIP FLOP

Behavioral Modelling

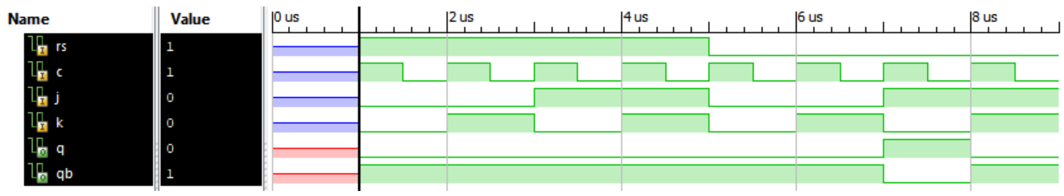
Verilog Program:

```
module JKff(q,qb,j,k,c,rs
    );
input j,k,c,rs;
output qb,q;
reg q;
assign qb=~q;
always@(posedge c)
begin
if(rs)
q=0;
else
casex({j,k})
2'b00:q=q;
2'b01:q=0;
2'b10:q=1;
2'b11:q=~q;
endcase
end
endmodule
```

RTL Schematic:



Simulation:



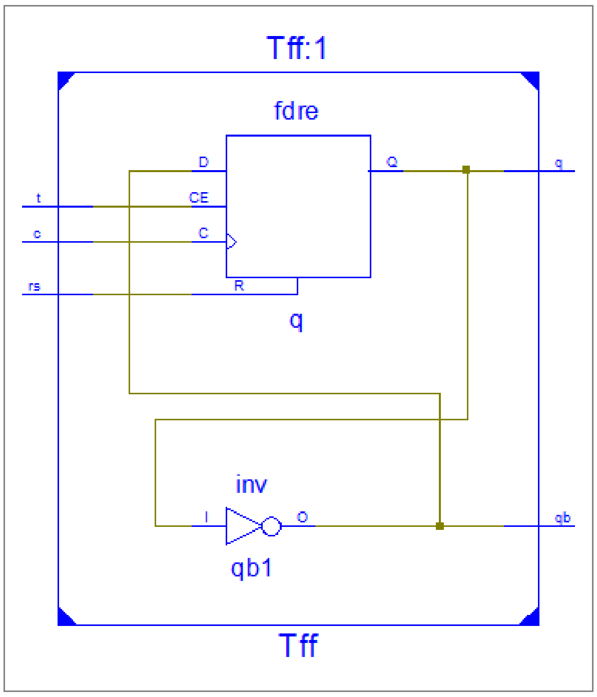
T FLIP FLOP

Behavioral Modelling

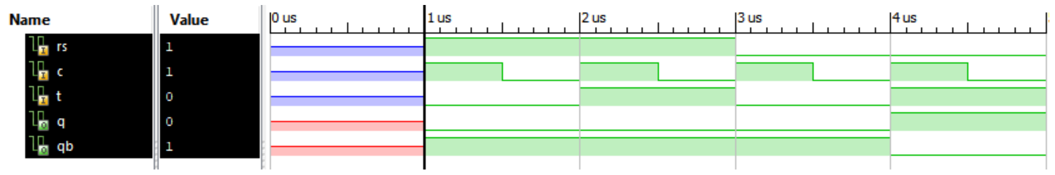
Verilog Program:

```
module Tff(q,qb,t,c,rs
    );
input t,c,rs;
output qb,q;
reg q;
assign qb=~q;
always@(posedge c)
begin
if(rs)
q=0;
else
casex(t)
1'b0:q=q;
1'b1:q=~q;
endcase
end
endmodule
```

RTL Schematic:



Simulation:



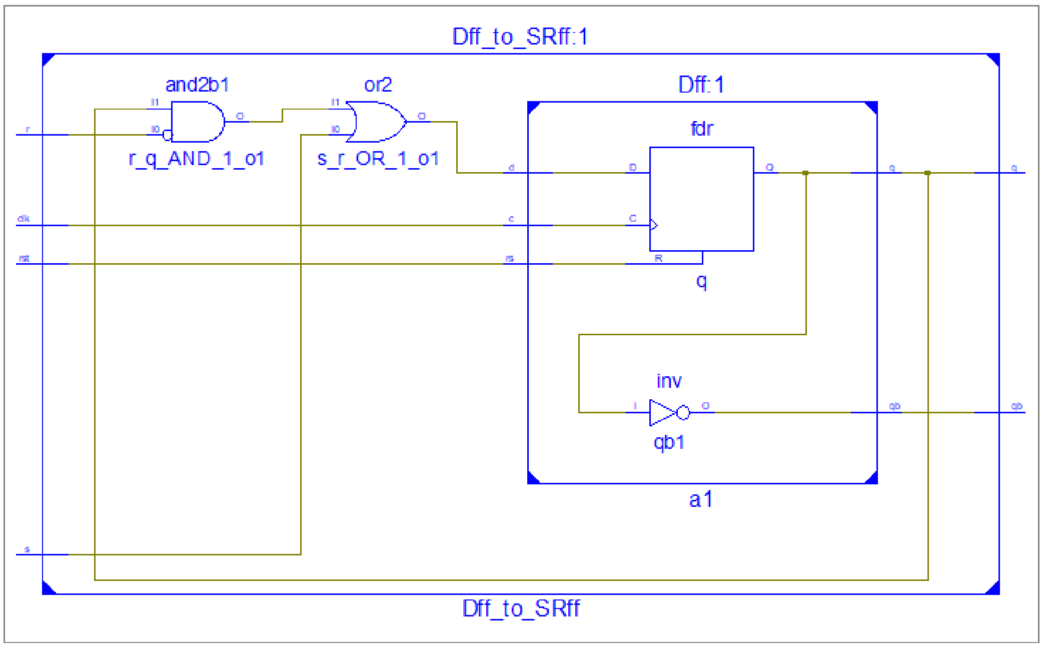
D to SR FLIP FLOP

Structural Modelling

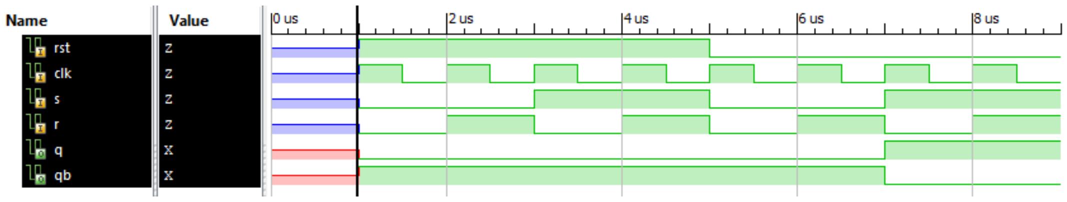
Verilog Program:

```
module Dff_to_SRff(q,qb,s,r,clk,rst);
input s,r;
input clk,rst;
output wire q,qb;
Dff a1(q,qb,(s|(~r&q)),clk,rst);
Endmodule
```

RTL Schematic:



Simulation:



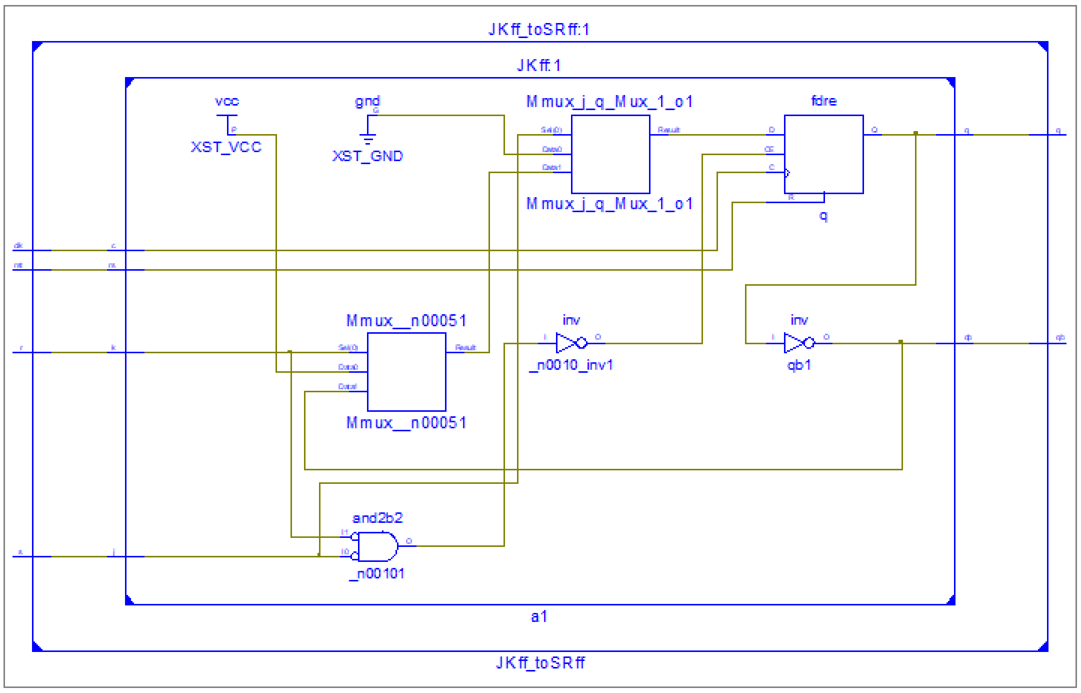
JK to SR FLIP FLOP

Structural Modelling

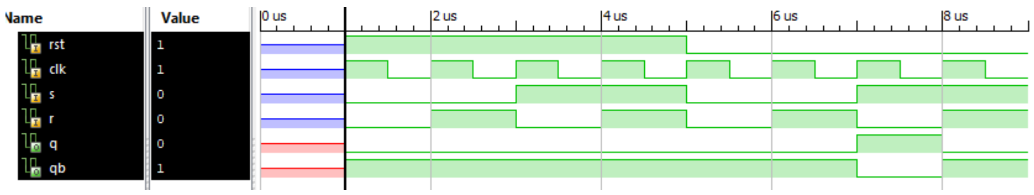
Verilog Program:

```
module JKff_to_SRff(q,qb,s,r,clk,rst);
input s,r;
input clk,rst;
output wire q,qb;
JKff a1(q,qb,s,r,clk,rst);
Endmodule
```

RTL Schematic:



Simulation:



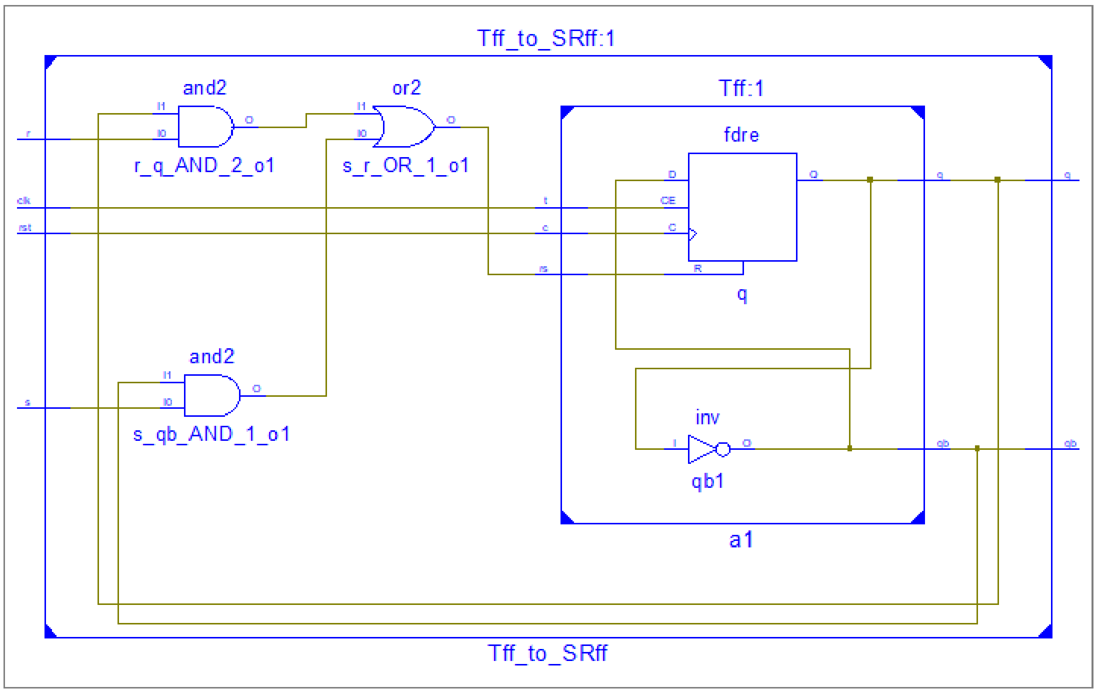
T to SR FLIP FLOP

Structural Modelling

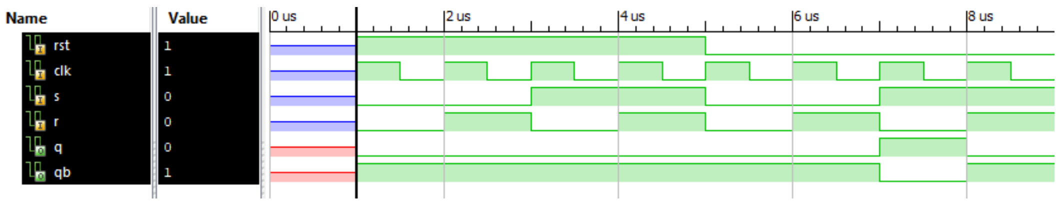
Verilog Program:

```
module Tff_to_SRff(q,qb,s,r,clk,rst);
input s,r;
input clk,rst;
output wire q,qb;
Tff a1(q,qb,((s&qb)|(r&q)),clk,rst);
Endmodule
```

RTL Schematic:



Simulation:



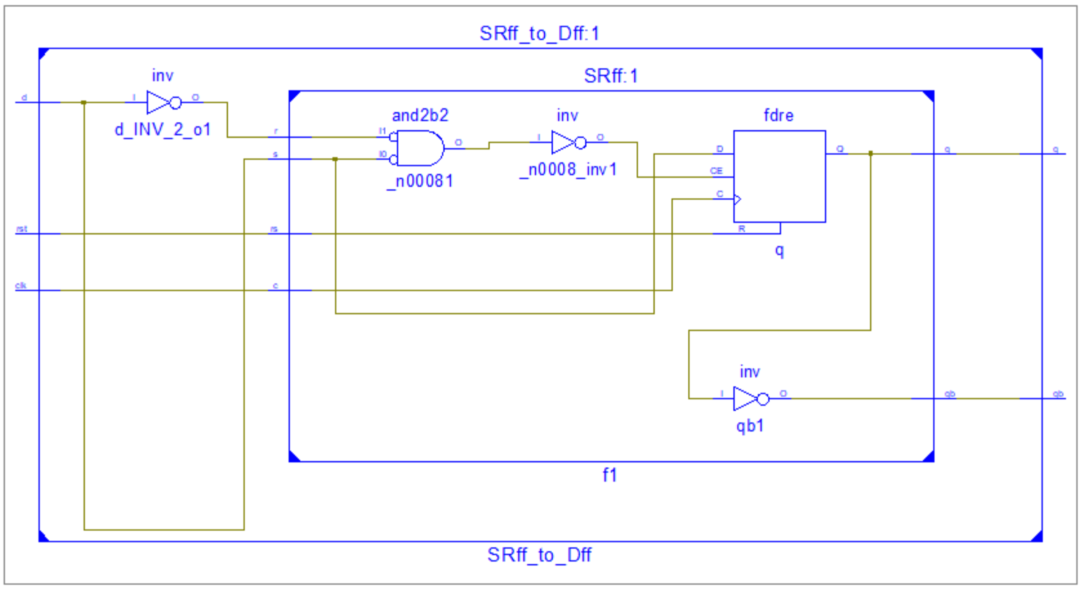
SR to D FLIP FLOP

Structural Modelling

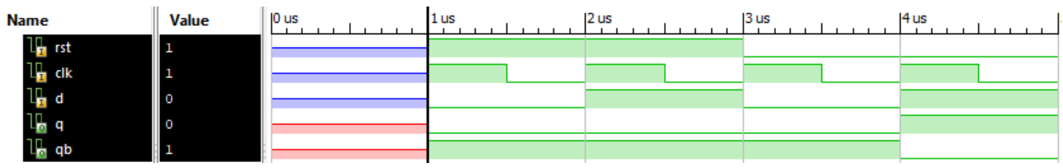
Verilog Program:

```
module SRff_to_Dff(q,qb,d,clk,rst);
input d;
input clk,rst;
output wire q,qb;
SRff f1(q,qb,d,~d,clk,rst);
Endmodule
```

RTL Schematic:



Simulation:



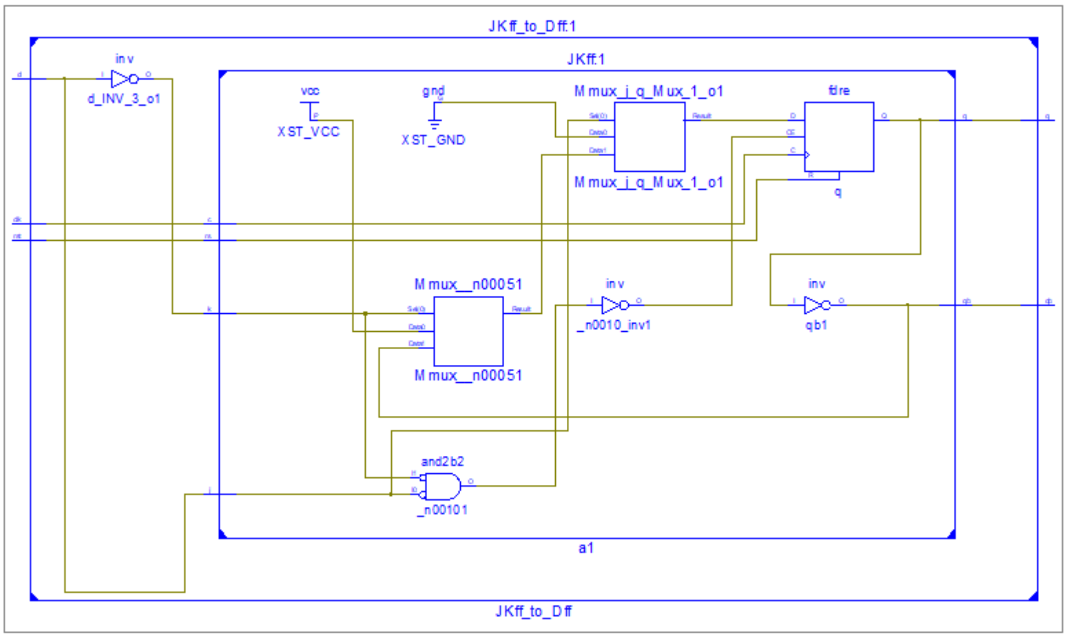
JK to D FLIP FLOP

Structural Modelling

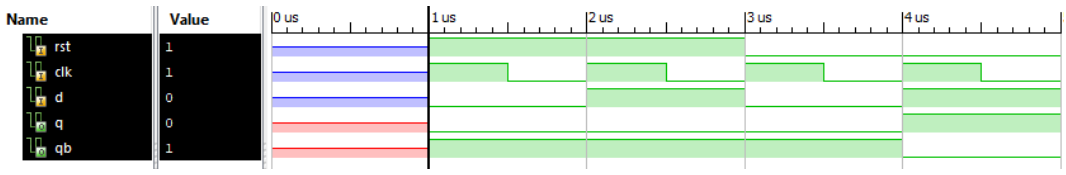
Verilog Program:

```
module JKff_to_Dff(q,qb,d,clk,rst);
input d;
input clk,rst;
output wire q,qb;
JKff a1(q,qb,d,~d,clk,rst);
Endmodule
```

RTL Schematic:



Simulation:



Name	Value
rst	1
clk	1
d	0
q	0
qb	1

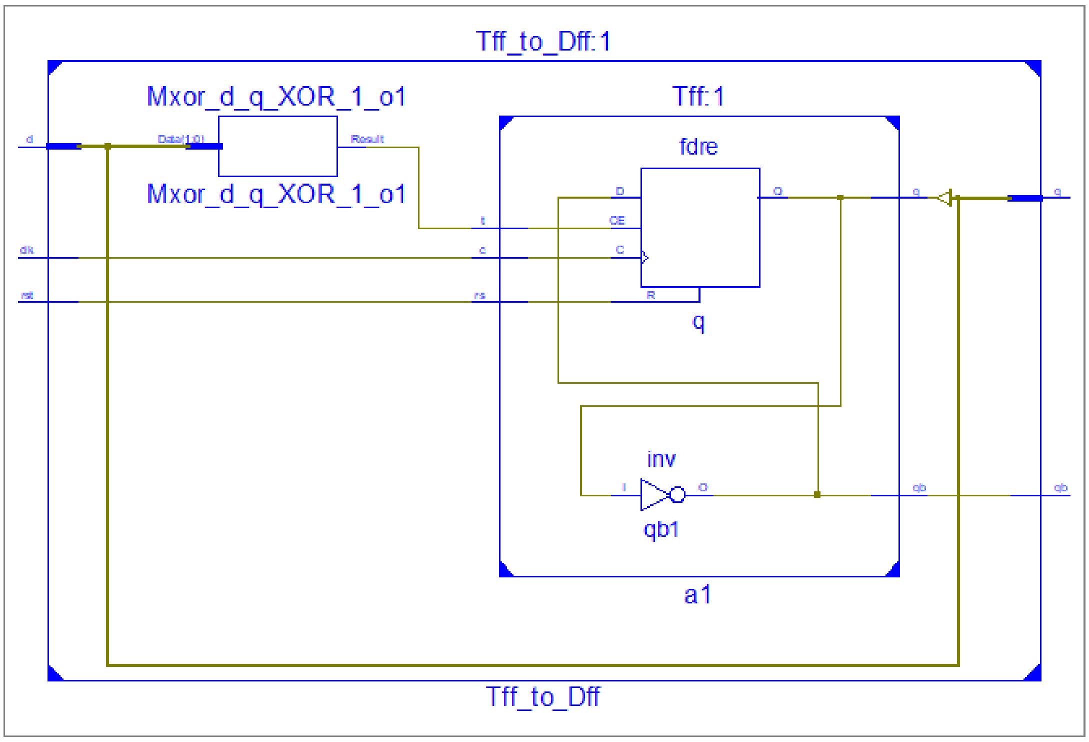
T to D FLIP FLOP

Structural Modelling

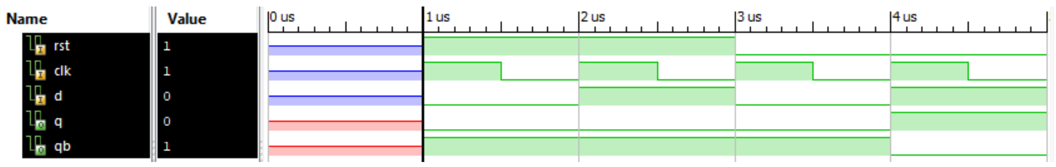
Verilog Program:

```
module Tff_to_Dff(q,qb,d,clk,rst);
input d;
input clk,rst;
output q,qb;
Tff a1(q,qb,d^q,clk,rst);
Endmodule
```

RTL Schematic:



Simulation:



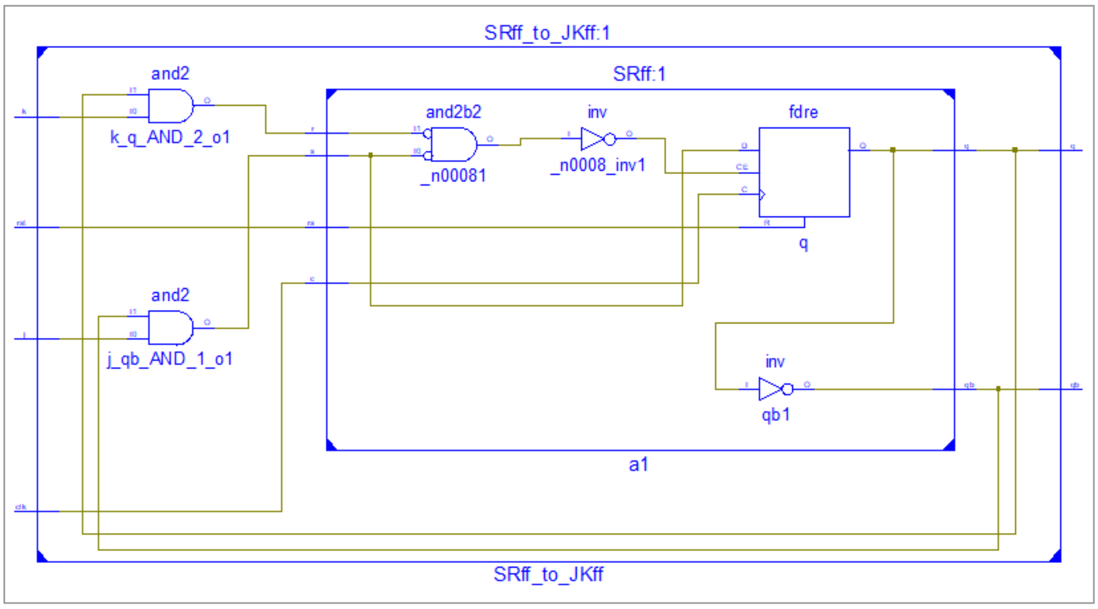
SR to JK FLIP FLOP

Structural Modelling

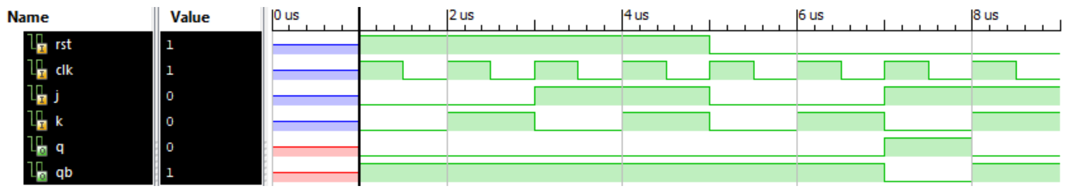
Verilog Program:

```
module SRff_to_JKff(q,qb,clk,j,k,clk,rst);
input j,k;
input clk,rst;
output wire q,qb;
SRff a1(q,qb,j&qb,k&q,clk,rst);
Endmodule
```

RTL Schematic:



Simulation:



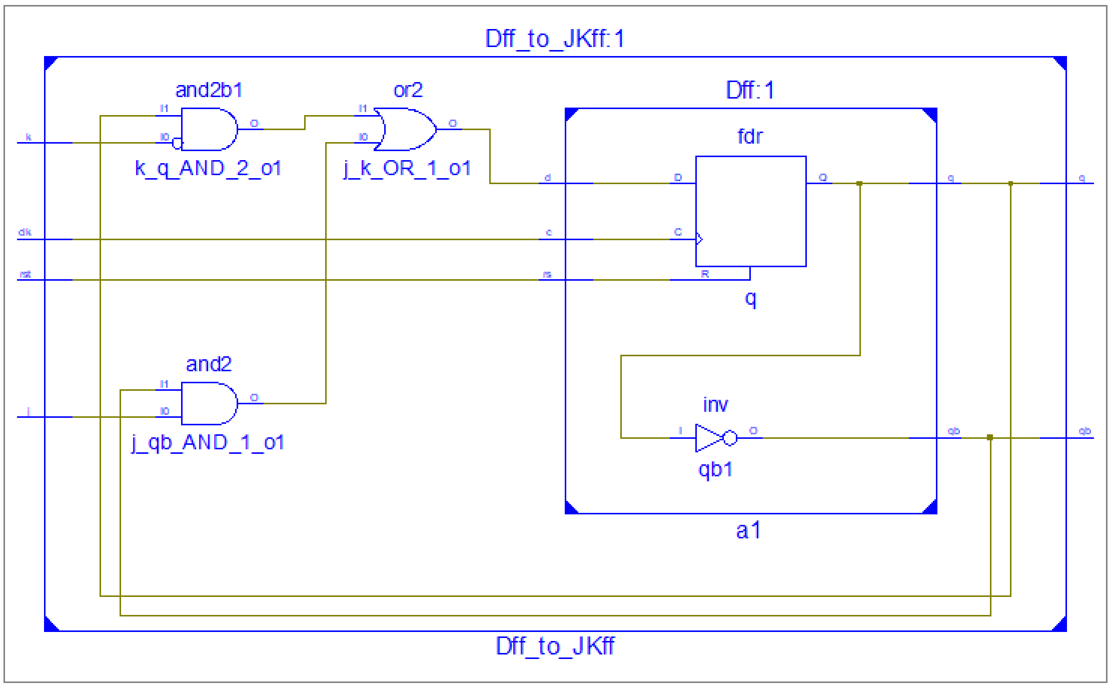
D to JK FLIP FLOP

Structural Modelling

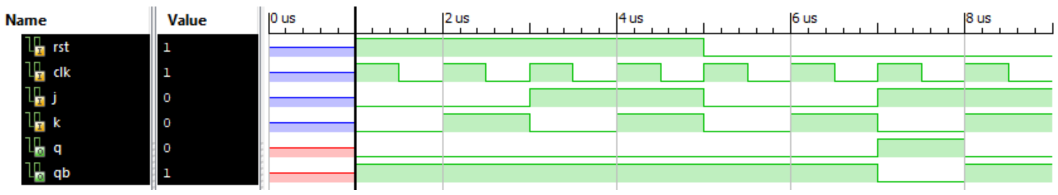
Verilog Program:

```
module Dff_to_JKff(q,qb,j,k,clk,rst);
input j,k;
input clk,rst;
output wire q,qb;
Dff a1(q,qb,((j&qb)|(~k&q)),clk,rst);
Endmodule
```

RTL Schematic:



Simulation:



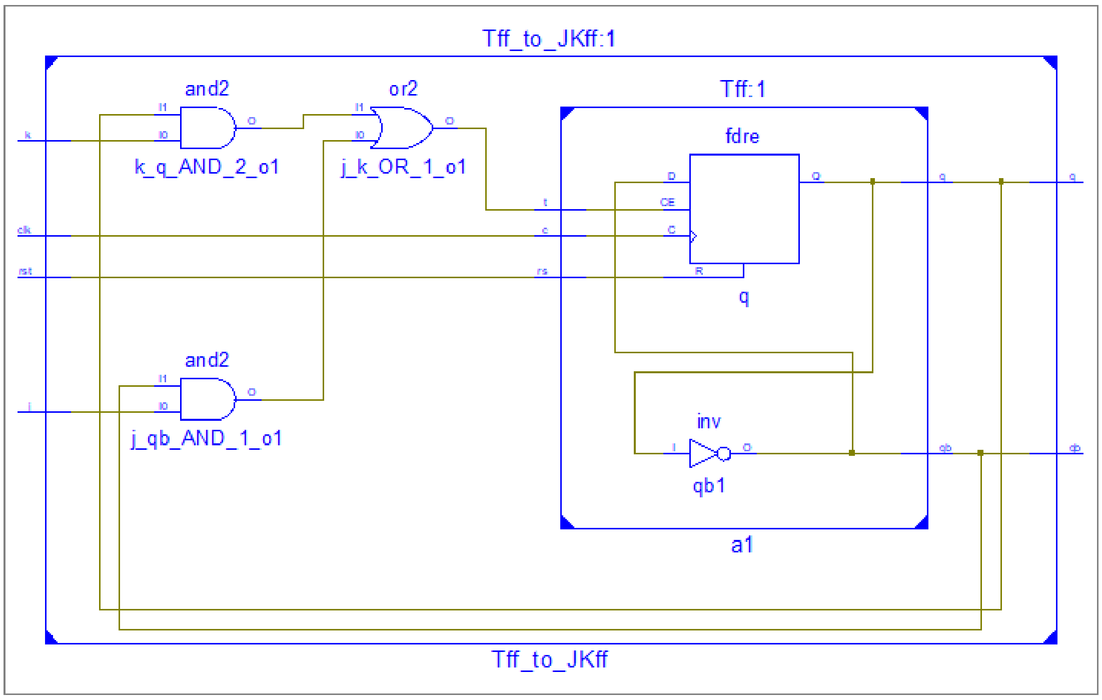
T to JK FLIP FLOP

Structural Modelling

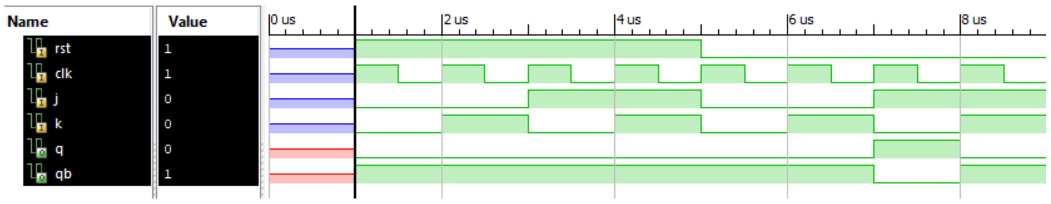
Verilog Program:

```
module Tff_to_JKff(q,qb,j,k,clk,rst);
input j,k;
input clk,rst;
output wire q,qb;
Tff a1(q,qb,((j&qb)|(k&q)),clk,rst);
Endmodule
```

RTL Schematic:



Simulation:



Name	Value
rst	1
clk	1
j	0
k	0
q	0
qb	1

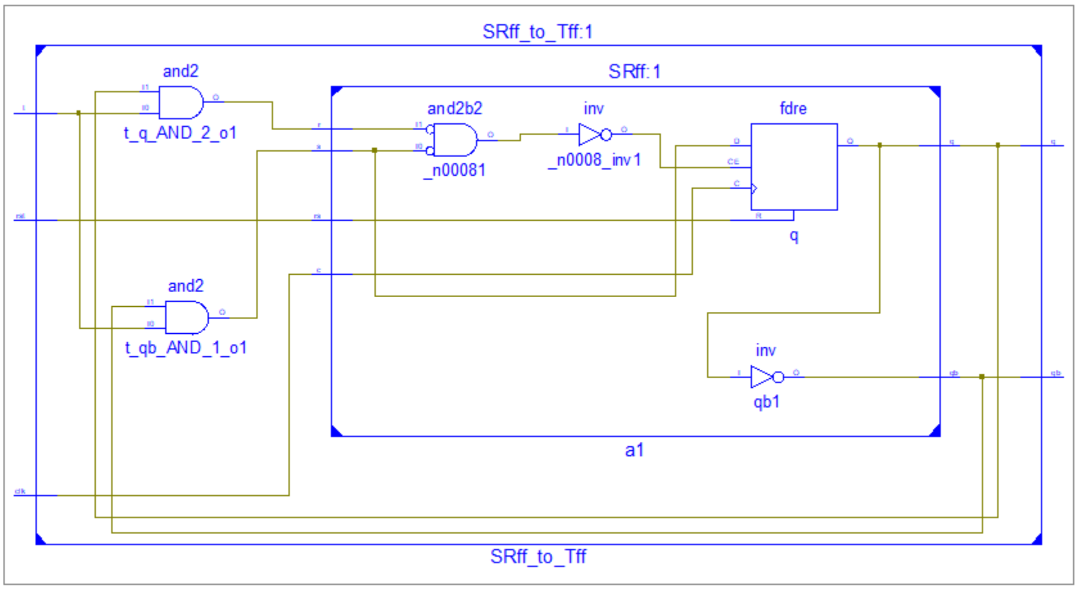
SR to T FLIP FLOP

Structural Modelling

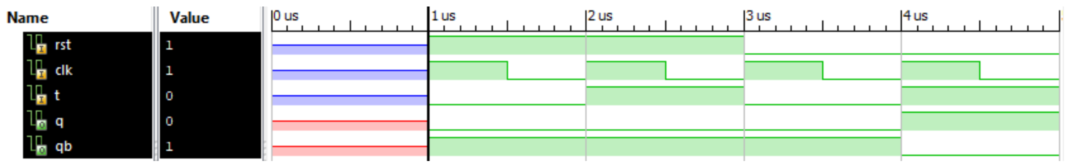
Verilog Program:

```
module SRff_to_Tff(q,qb,t,clk,rst);
input t;
input clk,rst;
output wire q,qb;
SRff a1(q,qb,t&qb,t&q,clk,rst);
Endmodule
```

RTL Schematic:



Simulation:



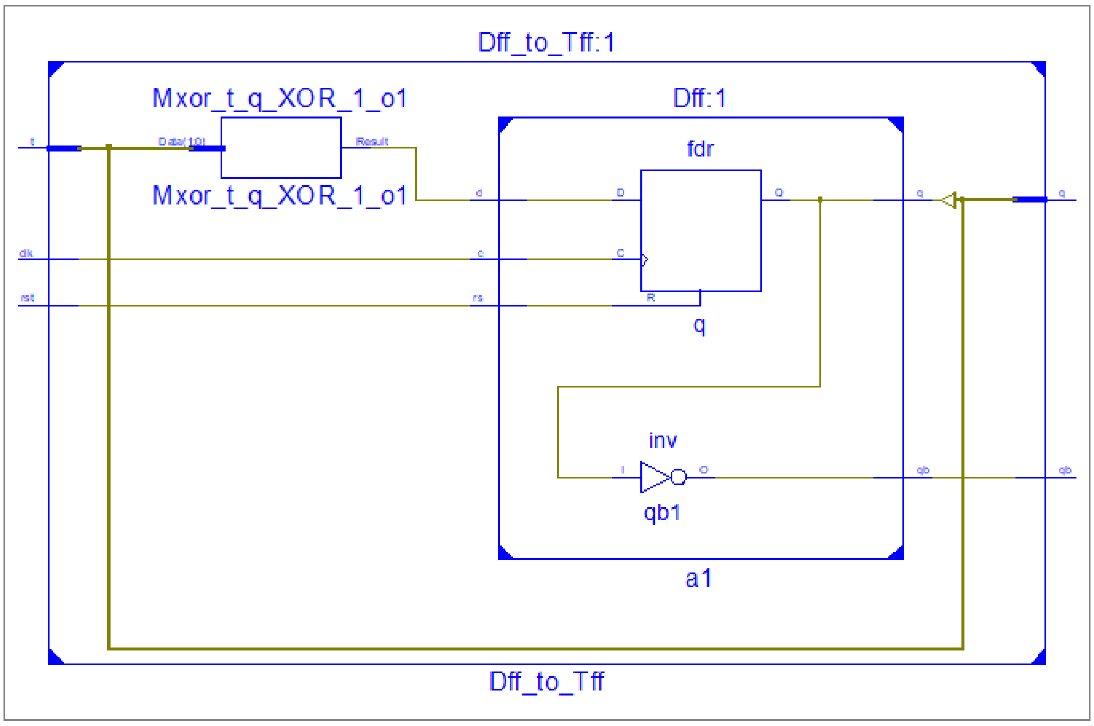
D to T FLIP FLOP

Structural Modelling

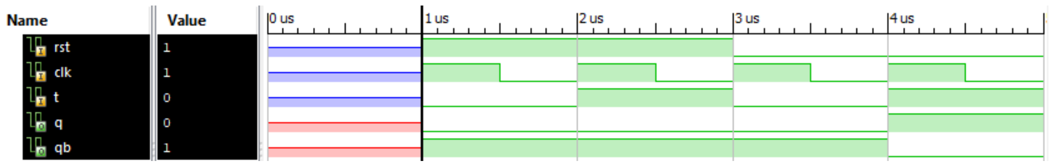
Verilog Program:

```
module Dff_to_Tff(q,qb,t,clk,rst);
input t;
input clk,rst;
output wire q,qb;
Dff a1(q,qb,t^q,clk,rst);
Endmodule
```

RTL Schematic:



Simulation:



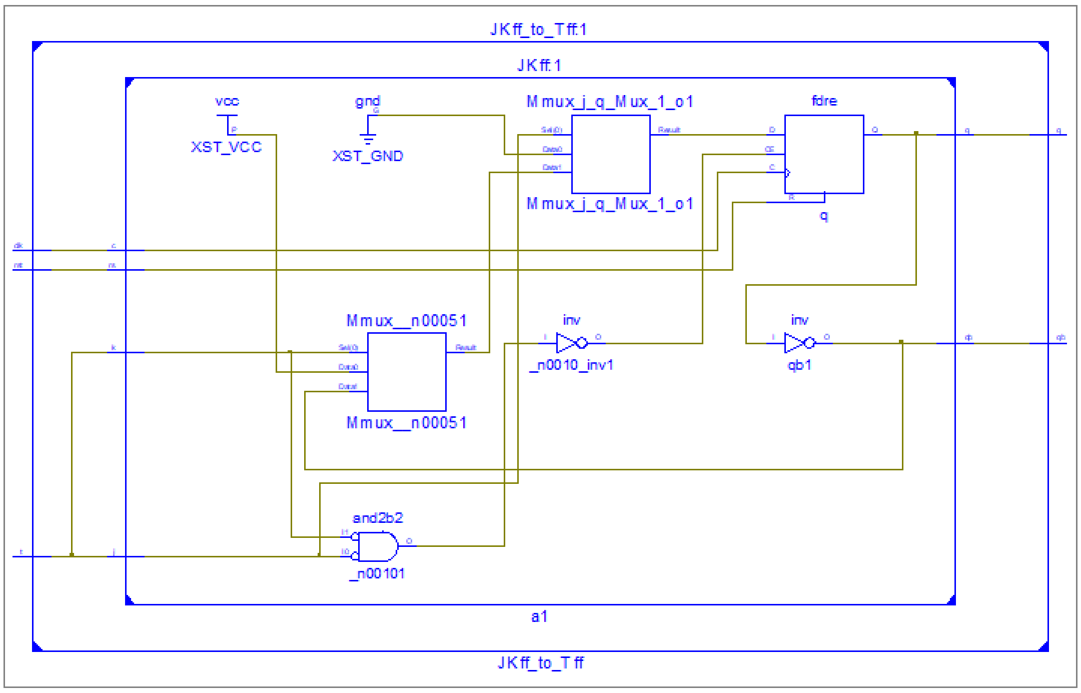
JK to T FLIP FLOP

Structural Modelling

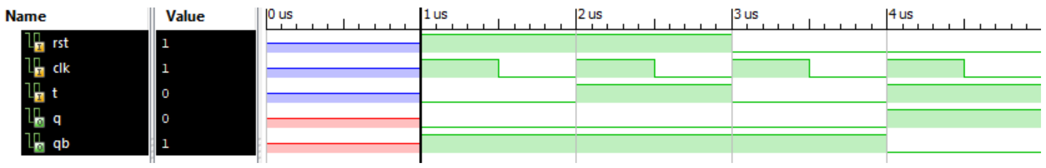
Verilog Program:

```
module JKff_to_Tff(q,qb,t,clk,rst);
input t;
input clk,rst;
output wire q,qb;
JKff a1(q,qb,t,t,clk,rst);
Endmodule
```

RTL Schematic:



Simulation:



Result:

Designed and implemented all flipflops and conversions using behavioral modelling using **Xilinx ISE 14.2**

EXPERIMENT-9
SYNCHRONOUS COUNTERS

Aim: Design, simulate and implement synchronous counters using Xilinx ISE.
Software Used: Xilinx **ISE-14.2**
Simulator Used: iSim
Synthesizer Used: XST
Procedure:

- Check the syntax of the program for any errors if any correct and verify again.
- In the process window, put the simulation mode in behavioral model, take a Verilog text fixture and give the input combinations.
- Perform simulation to verify the functionality and logic of the code.
- In the process window, change it to implement design highlight program in the source window, performs the run operations to implement the design.
- To view RTL and Technology schematic.

Verilog reports:

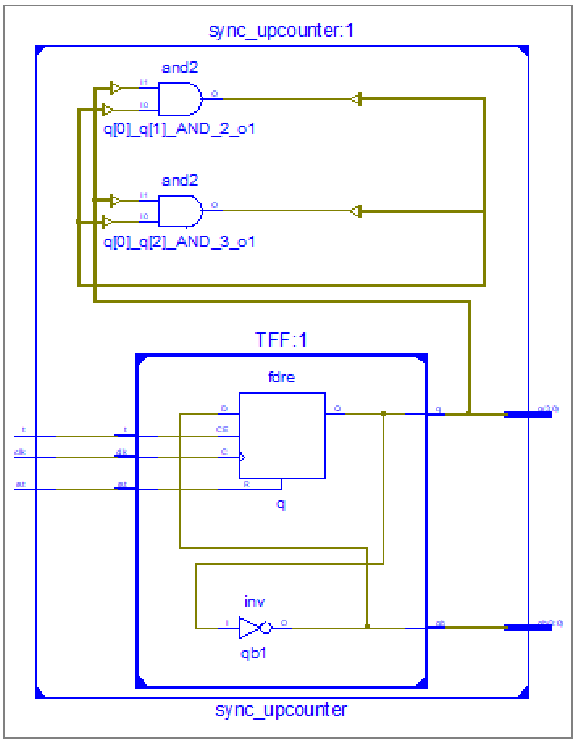
SYNCHRONUS UP COUNTER

Structural Modelling

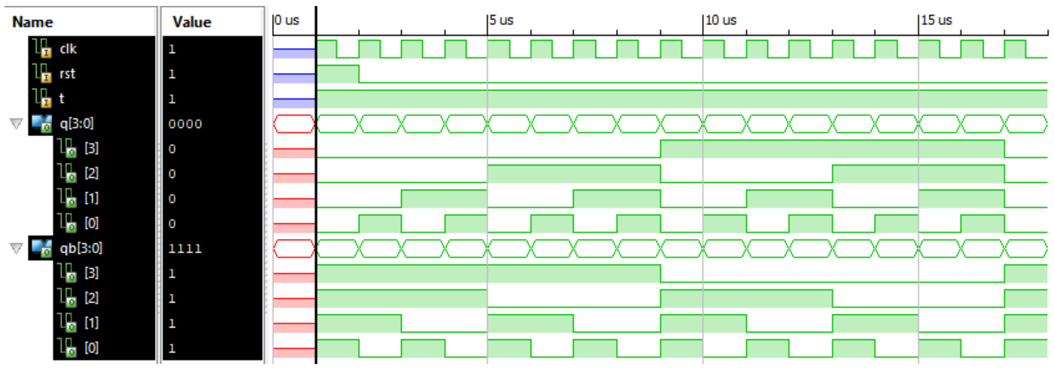
Verilog Program:

```
module sync_upcounter(q,qb,clk,rst,t);
input clk,rst;
input t;
output [3:0]q,qb;
TFF t0(q[0],qb[0],clk,rst,t);
TFF t1(q[1],qb[1],clk,rst,q[0]);
TFF t2(q[2],qb[2],clk,rst,q[0]&q[1]);
TFF t3(q[3],qb[3],clk,rst,(q[0]&q[1]&q[2]));
endmodule
```

RTL Schematic:



Simulation:



Verilog reports:

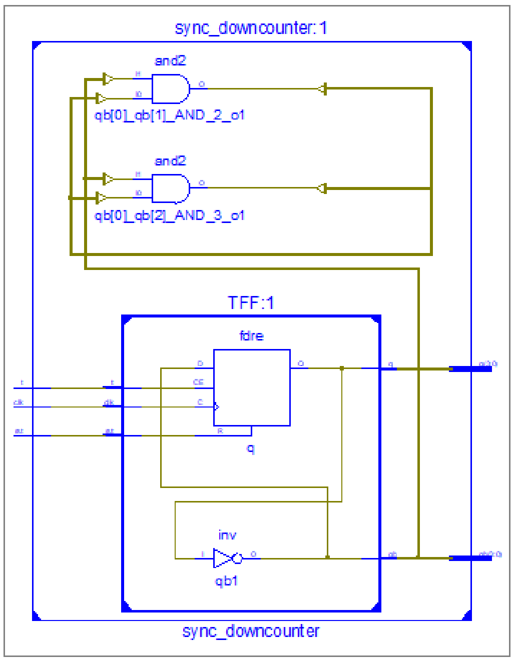
SYNCHRONUS DOWN COUNTER

Structural Modelling

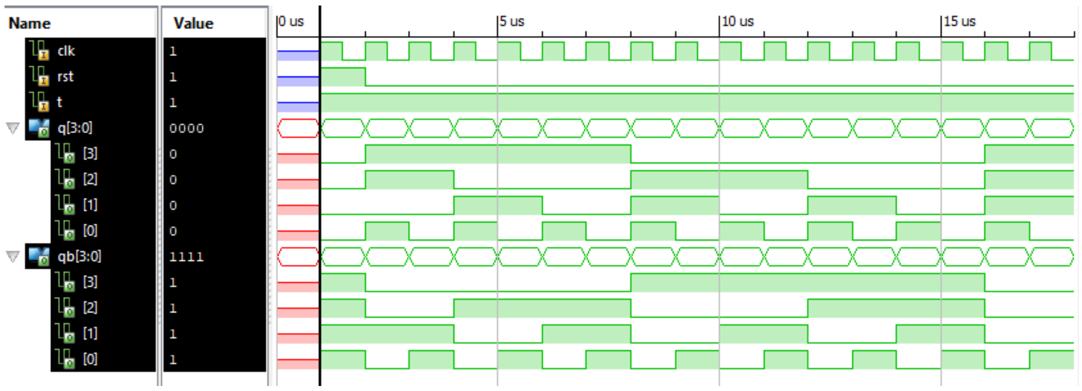
Verilog Program:

```
module sync_downcounter(q,qb,clk,rst,t);
input clk,rst;
input t;
output [3:0]q,qb;
TFF t0(q[0],qb[0],clk,rst,t);
TFF t1(q[1],qb[1],clk,rst,qb[0]);
TFF t2(q[2],qb[2],clk,rst,qb[0]&qb[1]);
TFF t3(q[3],qb[3],clk,rst,(qb[0]&qb[1]&qb[2]));
endmodule
```

RTL Schematic:



Simulation:



Verilog reports:

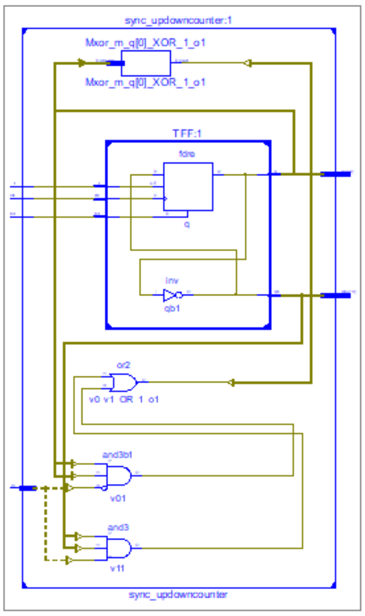
SYNCHRONOUS UP/DOWN COUNTER

Structural Modelling

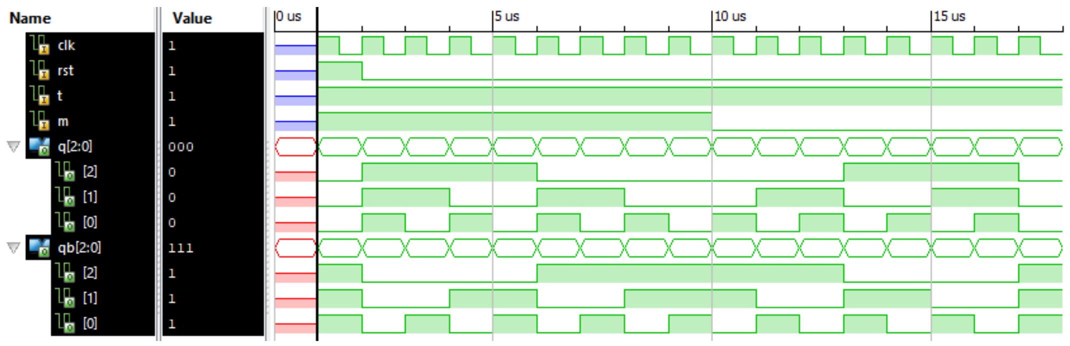
Verilog Program:

```
module sync_updowncounter(q,qb,clk,rst,t,m);
input clk,rst;
input t,m;
output [2:0]q,qb;
wire v0,v1;
and ao(v0, ~m&q[1]&q[0]);
and a1(v1, m&qb[1]&qb[0]);
TFF a2(q[0],qb[0],clk,rst,t);
TFF a3(q[1],qb[1],clk,rst,m^q[0]);
TFF a4(q[2],qb[2],clk,rst,v0|v1);
Endmodule
```

RTL Schematic:



Simulation:



Name	Value
clk	1
rst	1
t	1
m	1
q[2:0]	000
[2]	0
[1]	0
[0]	0
qb[2:0]	111
[2]	1
[1]	1
[0]	1

Result:

Designed and implemented synchronous counters using Behavioral modelling using **Xilinx ISE 14.2**

EXPERIMENT-9
ASYNCHRONOUS COUNTERS

Aim: Design, simulate and implement asynchronous counters using Xilinx ISE.
Software Used: Xilinx **ISE-14.2**
Simulator Used: iSim
Synthesizer Used: XST
Procedure:

- Check the syntax of the program for any errors if any correct and verify again.
- In the process window, put the simulation mode in behavioral model, take a Verilog text fixture and give the input combinations.
- Perform simulation to verify the functionality and logic of the code.
- In the process window, change it to implement design highlight program in the source window, performs the run operations to implement the design.
- To view RTL and Technology schematic.

Verilog reports:

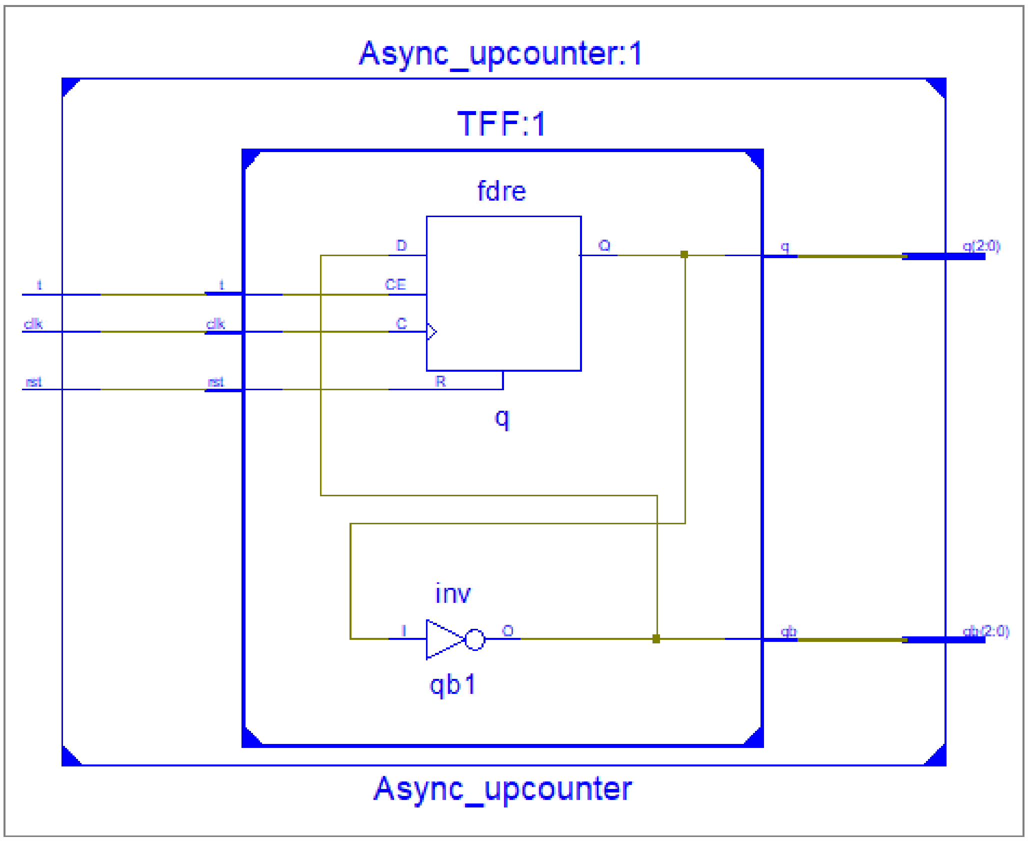
ASYNCHRONUS UP COUNTER

Structural Modelling

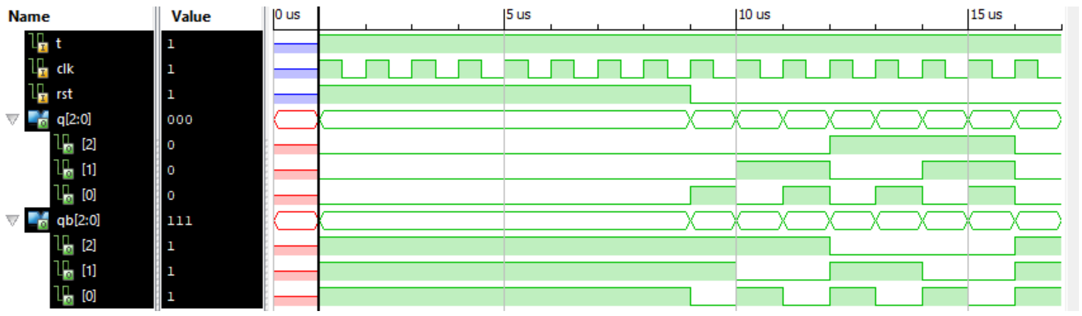
Verilog Program:

```
module Async_upcounter(q,qb,clk,rst,t);
input t;
input clk,rst;
output [2:0]q,qb;
TFF t0(q[0],qb[0],clk,rst,t);
TFF t1(q[1],qb[1],qb[0],rst,t);
TFF t2(q[2],qb[2],qb[1],rst,t);
Endmodule
```

RTL Schematic:



Simulation:



Name	Value
t	1
clk	1
rst	1
q[2:0]	000
[2]	0
[1]	0
[0]	0
qb[2:0]	111
[2]	1
[1]	1
[0]	1

Verilog reports:

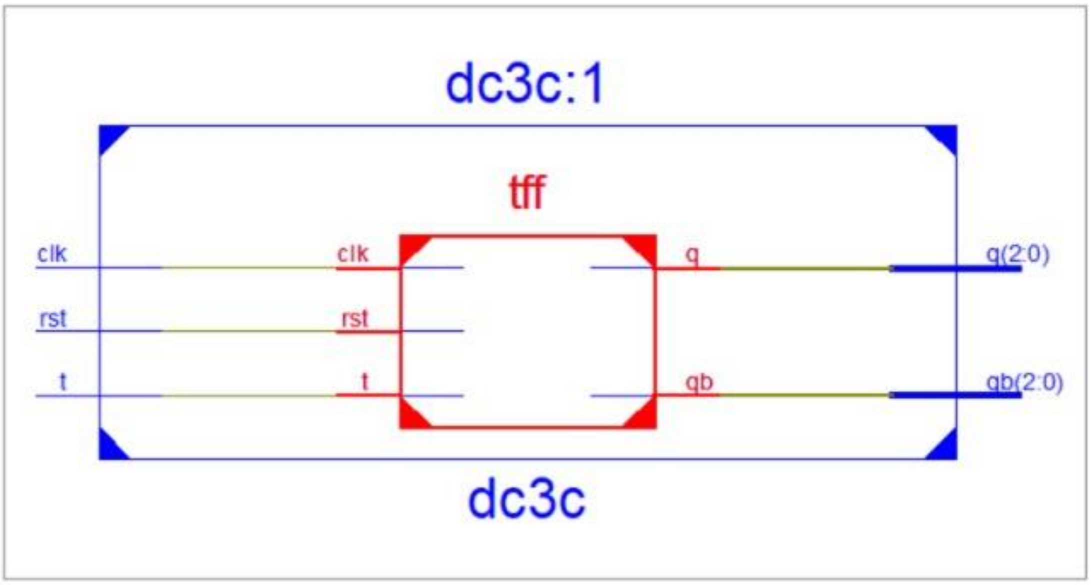
ASYNCHRONUS DOWN COUNTER

Structural Modelling

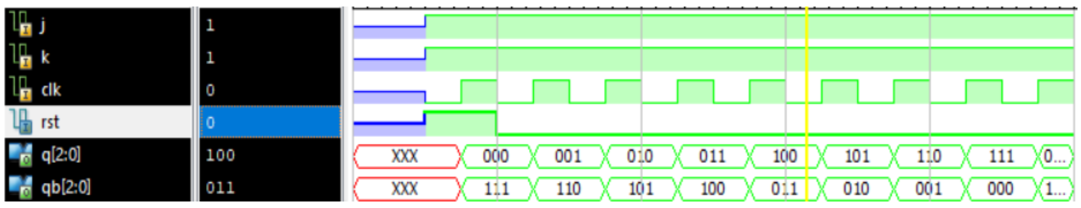
Verilog Program:

```
module up3c(q,qb,j,k,clk,rst);
input j,k;
input clk,rst;
output [2:0]q,qb;
jkff a0(q[0],qb[0],j,k,clk,rst);
jkff a1(q[1],qb[1],j,k,qb[0],rst);
jkff a2(q[2],qb[2],j,k,qb[1],rst);
endmodule
```

RTL Schematic:



Simulation:


j 1
k 1
clk 0
rst 0
q[2:0] 100
qb[2:0] 011
XXX 000 001 010 011 100 101 110 111 0...
XXX 111 110 101 100 011 010 001 000 1...

Verilog reports:

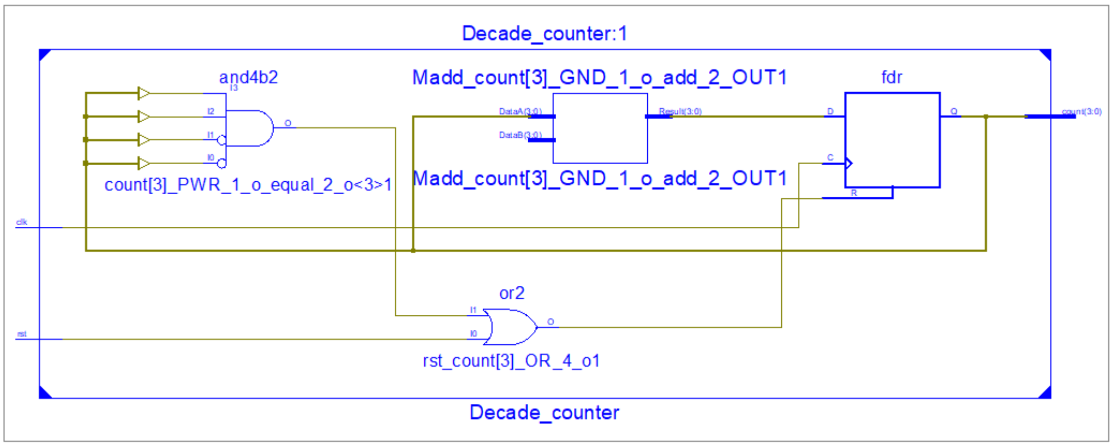
DECADE COUNTER

Behavioral Modelling

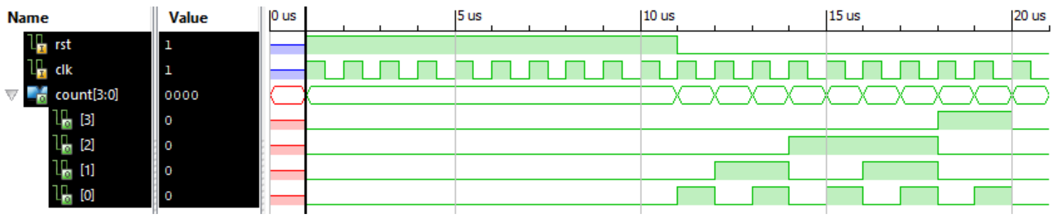
Verilog Program:

```
module Decade_counter(count,clk,rst);
input clk,rst;
output reg [3:0]count;
always @(posedge clk)
begin
if(rst||count==4'b1001)
count=4'b0000;
else
count=count+1;
end
endmodule
```

RTL Schematic:



Simulation:



Result:

Designed and implemented asynchronous counters using Behavioral modelling using **Xilinx ISE 14.2**

EXPERIMENT-11
SHIFT REGISTERS

Aim: Design, simulate and implement shift registers using Xilinx ISE.
Software Used: Xilinx **ISE-14.2**
Simulator Used: iSim
Synthesizer Used: XST
Procedure:

- Check the syntax of the program for any errors if any correct and verify again.
- In the process window, put the simulation mode in behavioral model, take a Verilog text fixture and give the input combinations.
- Perform simulation to verify the functionality and logic of the code.
- In the process window, change it to implement design highlight program in the source window, performs the run operations to implement the design.
- To view RTL and Technology schematic.

Verilog reports:

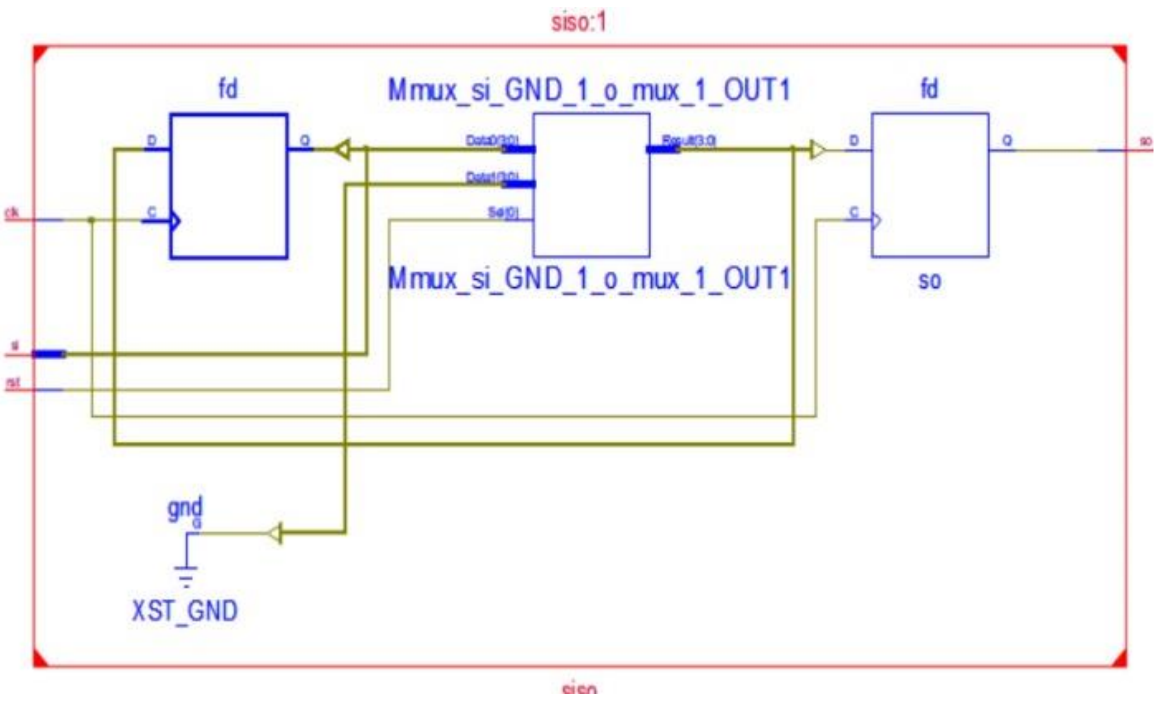
SERIAL IN SERIAL OUT

Behavioral Modelling

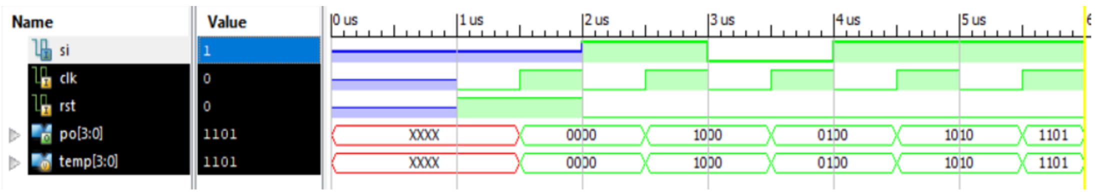
Verilog Program:

```
module siso(so,si,clk,rst);
input si;
input clk,rst;
output reg so;
reg [3:0]temp;
always @(posedge clk)
begin
if(rst)
temp=4'b0000;
else
begin temp=temp>>1;
temp[3]=si;
end
so=temp[0];
end
endmodule
```

RTL Schematic:



Simulation:



Verilog reports:

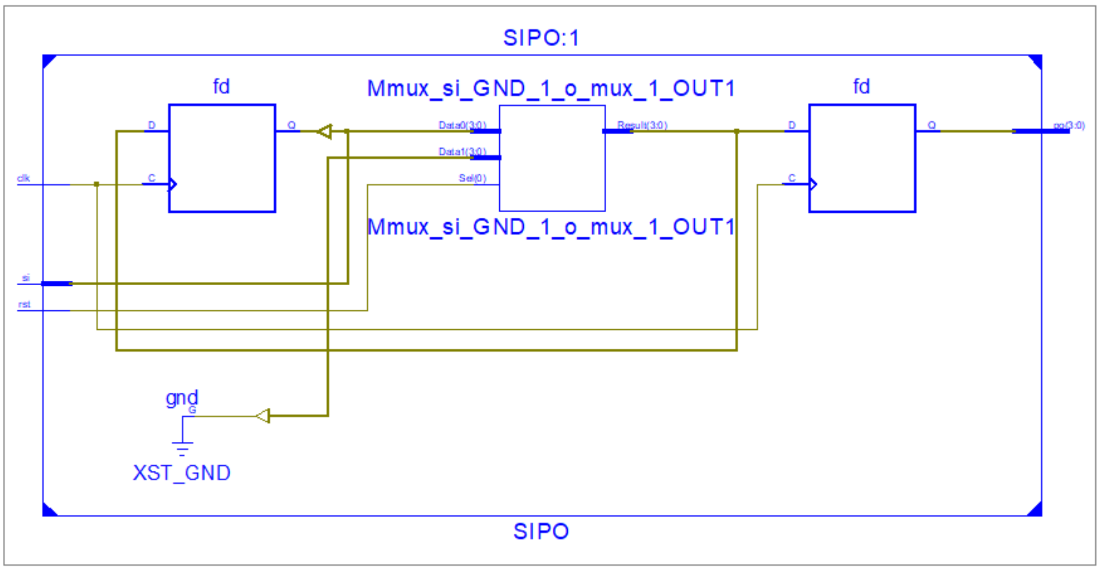
SERIAL IN PARALLEL OUT

Behavioral Modelling

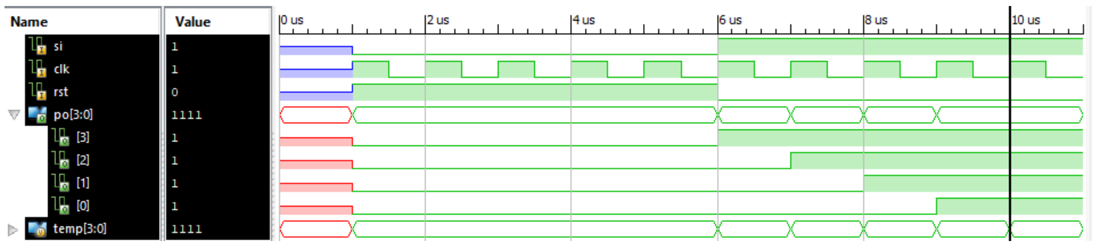
Verilog Program:

```
module SIPO(po,si,clk,rst);
input si;
input clk,rst;
output reg [3:0]po;
reg [3:0]temp;
always @(posedge clk)
begin
if(rst)
temp=4'b0000;
else
begin
temp=temp>>1;
temp[3]=si;
end
po=temp;
end
endmodule
```

RTL Schematic:



Simulation:



Name	Value
si	1
clk	1
rst	0
po[3:0]	1111
[3]	1
[2]	1
[1]	1
[0]	1
temp[3:0]	1111

Verilog reports:

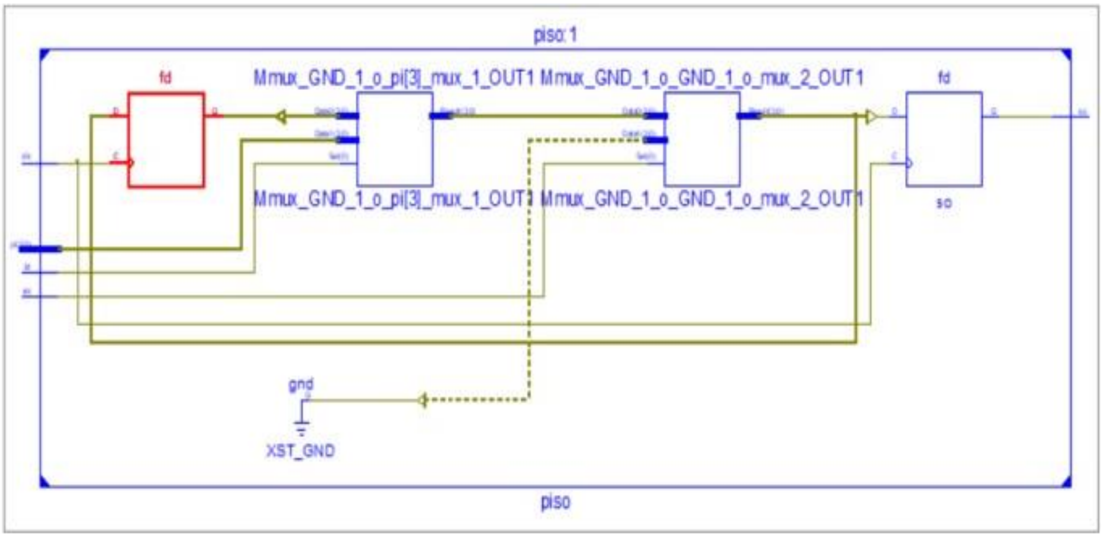
PARALLEL IN SERIAL OUT

Behavioral Modelling

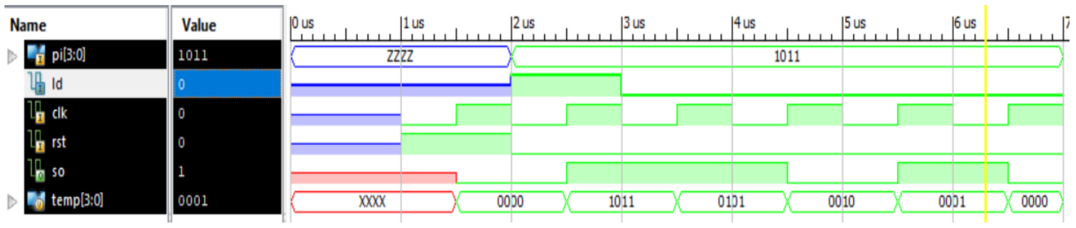
Verilog Program:

```
module piso(so,clk,rst,pi,ld);
input [3:0]pi;
input ld,clk,rst;
output reg so;
reg [3:0]temp;
always @(posedge clk)
begin if(rst)
temp=4'b0000; else
if(ld) temp=pi;
else
temp=temp>>1;
so=temp[0];
end
endmodule
```

RTL Schematic:


piso:1
fd
Mmux_GND_1_o_pi[3]_mux_1_OUT1
Mmux_GND_1_o_GND_1_o_mux_2_OUT1
fd
so
Mmux_GND_1_o_pi[3]_mux_1_OUT1
Mmux_GND_1_o_GND_1_o_mux_2_OUT1
gnd
XST_GND
piso

Simulation:



Verilog reports:

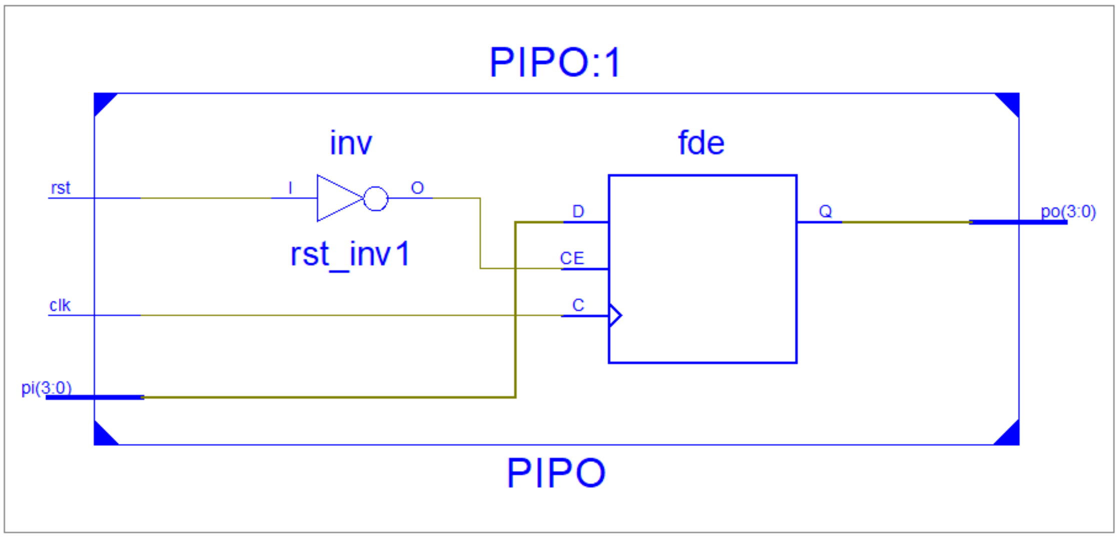
PARALLEL IN PARALLEL OUT

Behavioral Modelling

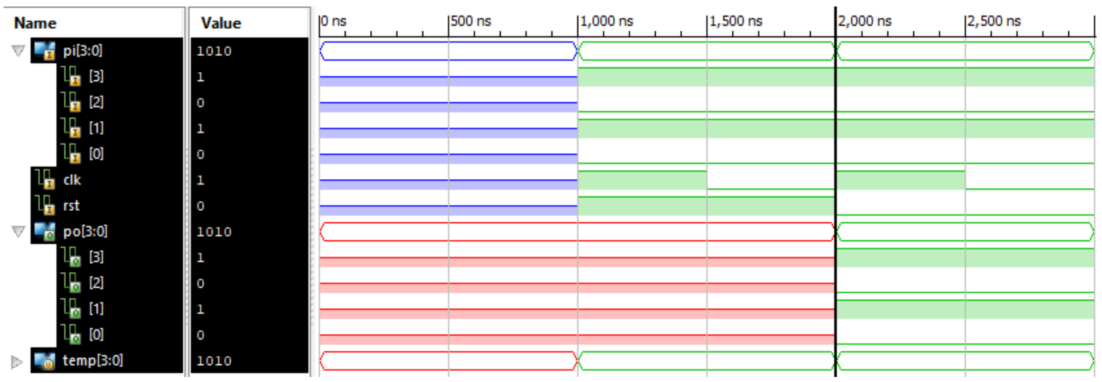
Verilog Program:

```
module pipo(po,pi,clk,rst);
input [3:0]pi;
input clk,rst;
output reg [3:0]po;
reg [3:0]temp;
always @(posedge clk)
begin
if(rst)
temp=4'b0000;
else
temp=pi;
po=temp;
end
endmodule
```

RTL Schematic:



Simulation:



```
pi[3:0]    1010
  [3]      1
  [2]      0
  [1]      1
  [0]      0
clk        1
rst        0
po[3:0]    1010
  [3]      1
  [2]      0
  [1]      1
  [0]      0
temp[3:0]  1010
```

Verilog reports:

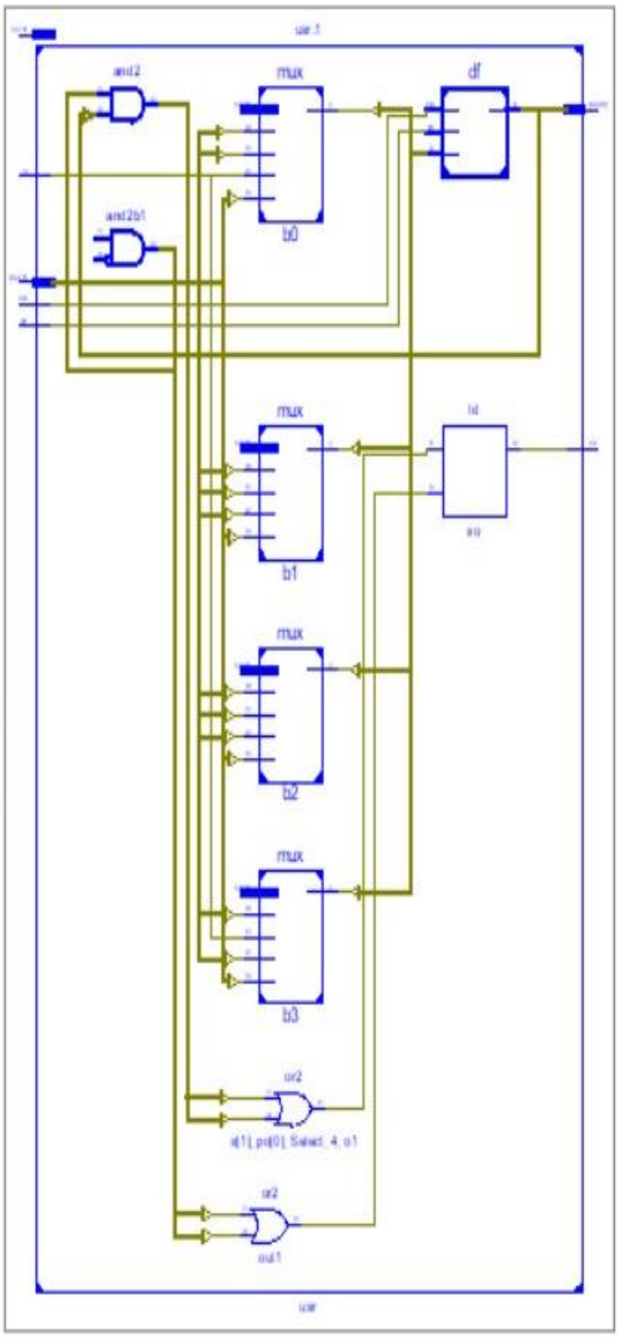
UNIVERSAL SHIFT REGISTER

Behavioral Modelling

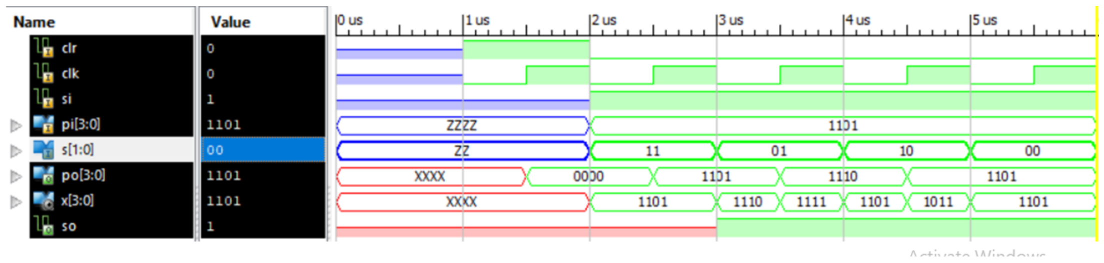
Verilog Program:

```
module usr(po,so,pi,si,s,clk,clr);
input clr,clk,si;
input [3:0]pi;
input [1:0]s;
output reg so;
output [3:0]po;
wire [3:0]x;

df  a0(po[0],x[0],clk,clr);
df  a1(po[1],x[1],clk,clr);
df  a2(po[2],x[2],clk,clr);
df  a3(po[3],x[3],clk,clr);
mux  b0(x[0],s[1:0],pi[0],si,po[1],po[0]);
mux  b1(x[1],s[1:0],pi[1],po[0],po[2],po[1]);
mux  b2(x[2],s[1:0],pi[2],po[1],po[3],po[2]);
mux  b3(x[3],s[1:0],pi[3],po[2],si,po[3]);
always @(s)
begin
case(s)
2'b01:so=po[0];
2'b10:so=po[3];
endcase
end
endmodule

module mux(y,s,i3,i2,i1,i0);
input [1:0]s;
input i3,i2,i1,i0;
output y;
assign y=(~s[1]&~s[0]&i0)|(~s[1]&s[0]&i1)|(s[1] &~s[0]&i2)|(s[1]&s[0]&i3);
endmodule
module df(q,d,clk,clr);
input clr,clk,d;
output reg q;
always @(posedge clk)
begin
if(clr) q=0;
else
casex(d)
0:q=0;
1:q=1;
endcase
end
endmodule
```

RTL Schematic:



Simulation:



Name	Value
clr	0
clk	0
si	1
pi[3:0]	1101
s[1:0]	00
po[3:0]	1101
x[3:0]	1101
so	1

Result:

Designed and implemented shift registers using Behavioral modelling using **Xilinx ISE 14.2**

EXPERIMENT-12
STATE MACHINES

Aim: Design, simulate and implement melay and more state machines using Xilinx ISE.
Software Used: Xilinx **ISE-14.2**
Simulator Used: iSim
Synthesizer Used: XST
Procedure:

- Check the syntax of the program for any errors if any correct and verify again.
- In the process window, put the simulation mode in behavioral model, take a Verilog text fixture and give the input combinations.
- Perform simulation to verify the functionality and logic of the code.
- In the process window, change it to implement design highlight program in the source window, performs the run operations to implement the design.
- To view RTL and Technology schematic.

Verilog reports:

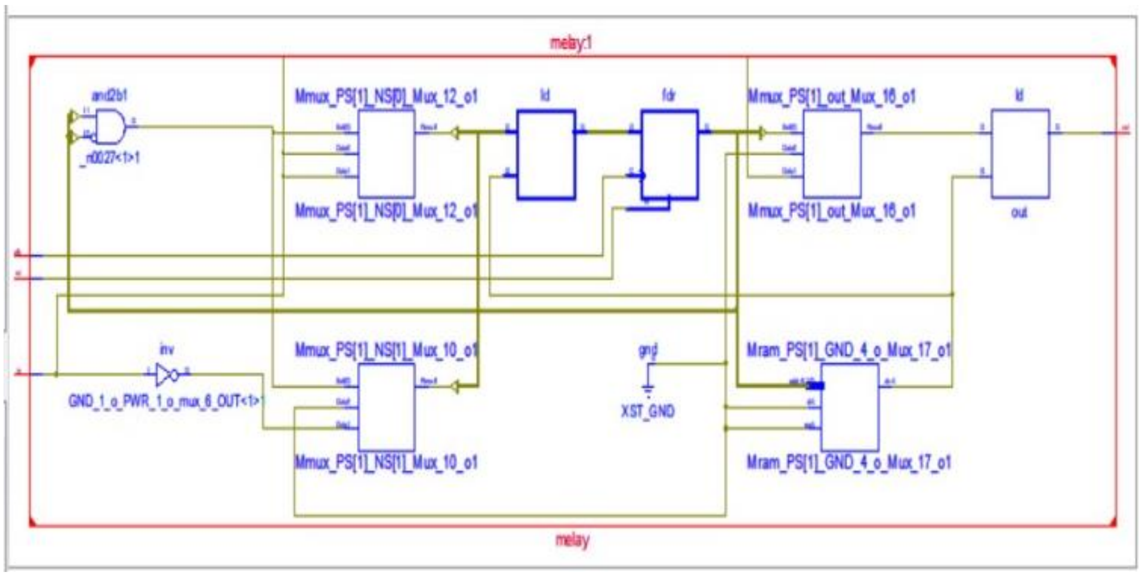
MELAY model 101 sequence detector

Behavioral Modelling

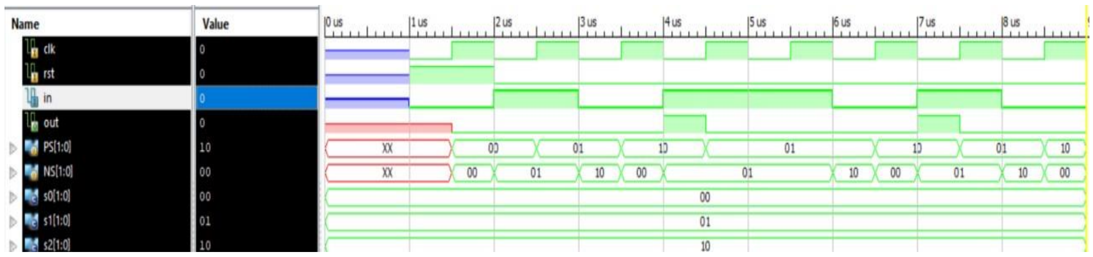
Verilog Program:

```
module melay(out,clk,rst,in);
input clk,rst,in;
output reg out;
parameter
s0=2'b00,s1=2'b01,s2=2'b10;
reg [1:0]PS,NS;
always@(posedge clk)
if(rst) PS=s0;
else   PS=NS;
always @(in,PS)
begin
case(PS)
s0:if(in==1)NS=s1; else NS=s0;
s1:if(in==0)NS=s2; else NS=s1;
s2:if(in==1)NS=s1; else NS=s0;
endcase
end
always @(in,PS)
begin
case(PS)
s0: out=0;
s1: out=0;
s2: if(in==1)out=1;else out=0;
endcase
end
endmodule
```

RTL Schematic:


melay:1
and2b1
_n0027<1>1
Mmux_PS[1]_NS[0]_Mux_12_o1
ld
fdr
Mmux_PS[1]_out_Mux_16_o1
ld
out
inv
GND_1_o_PWR_1_o_mux_6_OUT<1>1
Mmux_PS[1]_NS[1]_Mux_10_o1
gnd
XST_GND
Mram_PS[1]_GND_4_o_Mux_17_o1
melay

Simulation:



Verilog reports:

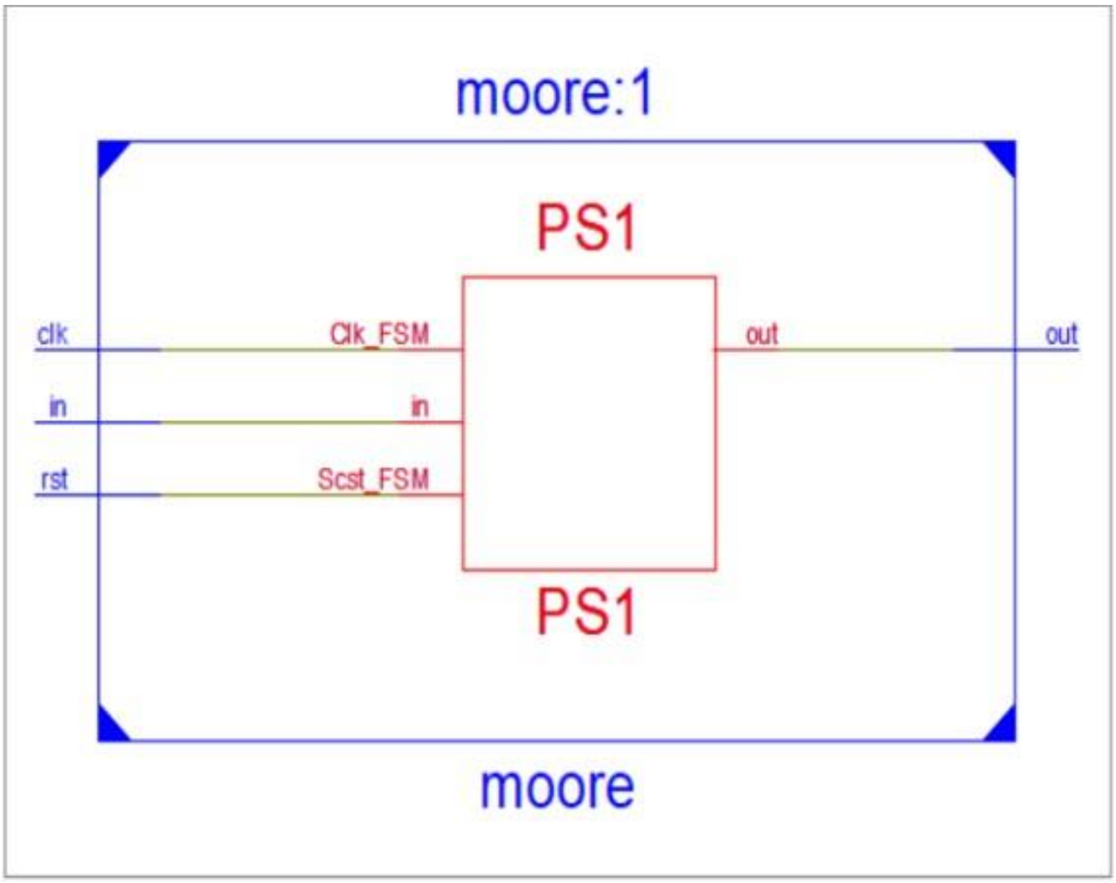
MOORE model 010 sequence detector

Behavioral Modelling

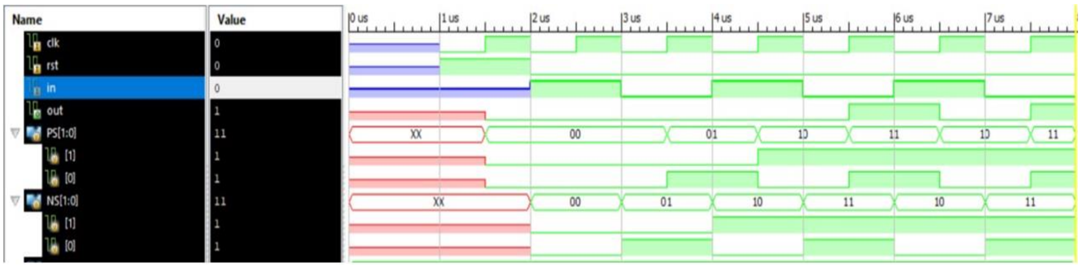
Verilog Program:

```
module moore(out,clk,rst, in);
input clk,rst,in;
output reg out;
parameter
s0=2'b00,s1=2'b01,s2=2'b10,s3=2'b11;
reg [1:0]PS,NS;
always@(posedge clk)
if(rst)
PS=s0;
else
PS=NS;
always @(in)
begin
case(PS)
s0:if(in==0)NS=s1; else NS=s0;
s1:if(in==1)NS=s2; else NS=s1;
s2:if(in==0)NS=s3; else NS=s0;
s3:if(in==1)NS=s2; else NS=s0;
endcase
end
always @(PS)
begin case(PS)
s0: out=0;
s1: out=0;
s2: out=0;
s3: out=1;
endcase
end
endmodule
```

RTL Schematic:



Simulation:



Name	Value
clk	0
rst	0
in	0
out	1
PS[1:0]	11
[1]	1
[0]	1
NS[1:0]	11
[1]	1
[0]	1

Result:

Designed and implemented melay and moore state machines using **Xilinx ISE 14.2**