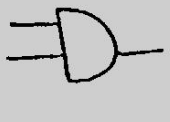
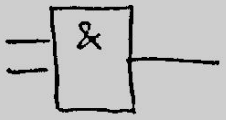


traditional symbol

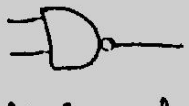


IEC symbol

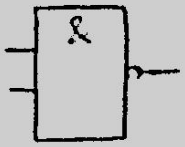
input x_1	input x_2	o/p Y
0	0	0
0	1	0
1	0	0
1	1	1

=> NAND Gate : (NAND = Not AND)

This is an AND gate with the output inverted, as shown by 'o' on the output. The output is true if input x_1 and input x_2 are not both true : Y = NOT (x_1 AND x_2). A NAND Gate can have two or more inputs, its output is true if NOT all inputs are true.



traditional symbol



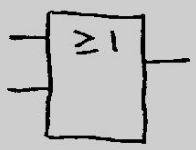
IEC symbol

input x_1	input x_2	o/p Y
0	0	1
0	1	1
1	0	1
1	1	0

=> OR Gate :

The output Y is true if input x_1 OR input x_2 is true (or both of them are true) : Y = x_1 OR x_2 An OR gate can have two or more inputs, its output is true if at least one input is true.

traditional symbol

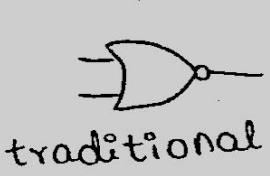


IEC symbol

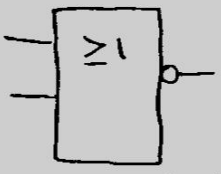
input x_1	input x_2	o/p Y
0	0	0
0	1	1
1	0	1
1	1	1

=> NOR Gate: (NOR = Not OR):

This is an OR gate with the output inverted, as shown by the 'o' on the output. The output Y is true if NOT inputs X_1 OR X_2 are true: Y = NOT (X_1 OR X_2). A NOR Gate can have two or more inputs. its output if no inputs are true.



traditional symbol



IEC symbol

input X_1	input X_2	o/p Y
0	0	1
0	1	0
1	0	0
1	1	0

=> EX - OR : (Exclusive - OR) gate :

The output Y is true if either input X_1 is true OR input X_2 is true, but not when both of them are true : Y = (X_1 AND NOT B) OR (B AND NOT A). This is like an OR Gate but Excluding both inputs being true. The output is true if inputs X_1 and X_2 are Different. Ex-OR gates can only have 2 inputs.

traditional symbol

IEC symbol

input X_1	input X_2	o/p Y
0	0	0
0	1	1
1	0	1
1	1	0

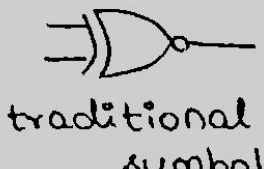
=> EX - NOR (Exclusive - NOR) gate:

This is an EX-OR gate with the output inverted, as shown by the 'o' on the output. The output Y is true if inputs X_1 and

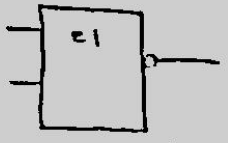
x_2 are the same (both true or both false.)

$Y = (X_1 \text{ AND } X_2) \text{ OR } (\text{NOT } x_1 \text{ AND NOT } x_2)$

EX-NOR gates can only have 2 inputs



traditional symbol

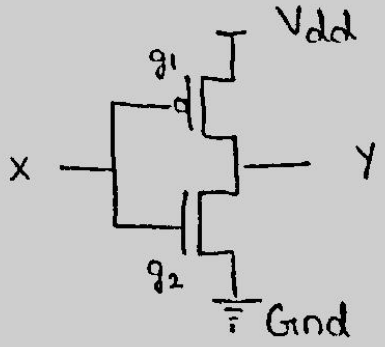


IEC symbol

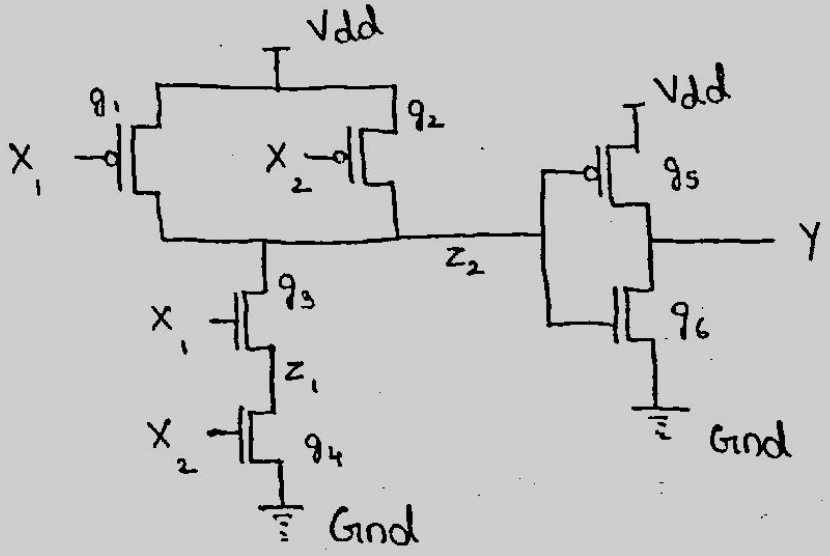
input X_1	input X_2	o/p Y
0	0	1
0	1	0
1	0	0
1	1	1

Circuit Diagram in Switch Level Model:

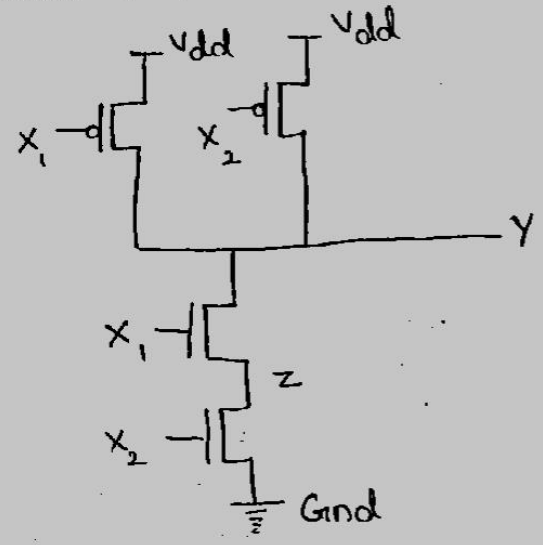
=> Not Gate:



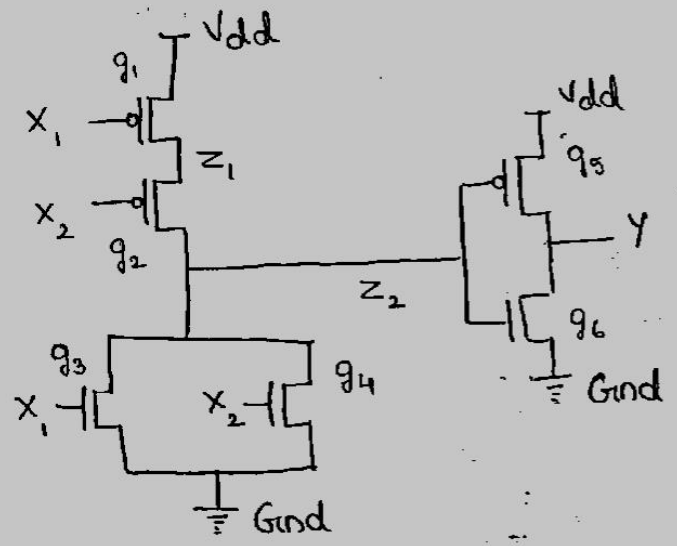
=> And Gate



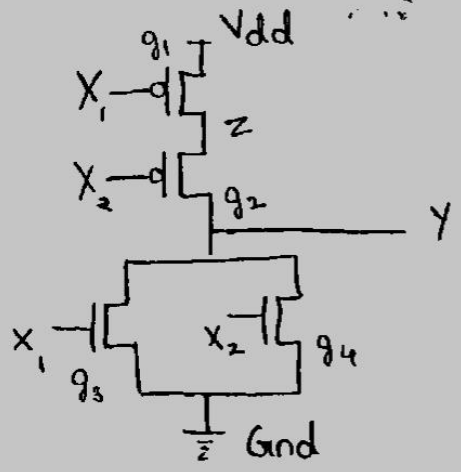
=> Nand Gate:



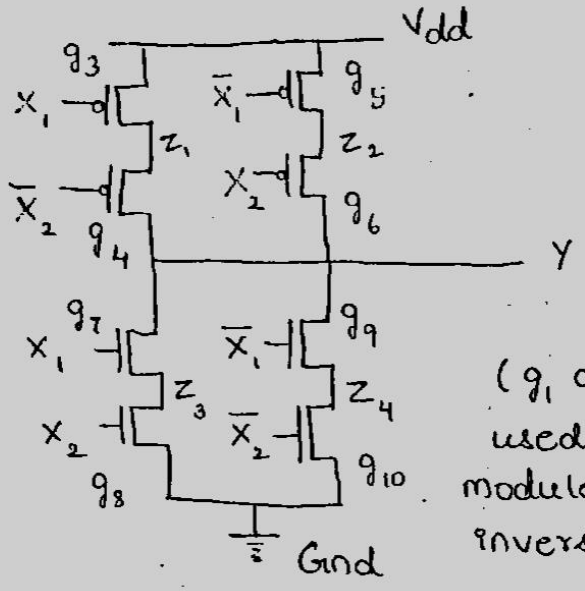
=> OR Gate:



=> NOR Gate:



=>EXOR Gate:

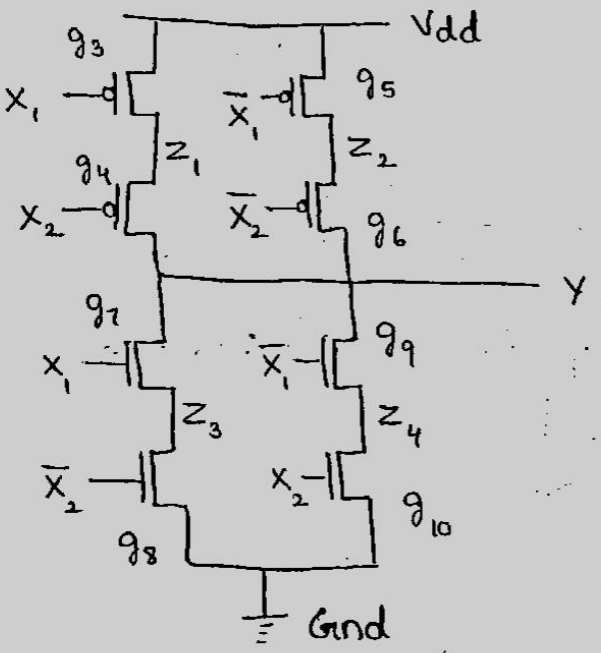


(g_1 and g_2 are used by inverter module to get inverse of X_1 & X_2)

↓

[for both EXOR & EX-NOR Gate]

=> EX-NOR Gate:



AND(SW) EDA

DESIGN CODE	TEST BENCH

module andgate_usn94(y,x1,x2);
output y;
input x1, x2;
supply1 vdd;
supply0 gnd;
wire z1,z2;
 	
pmos g1(z2,vdd,x1);
pmos g2(z2,vdd,x2);
nmos g3(z2,z1,x1);
nmos g4(z1,gnd,x2);
pmos g5(y,vdd,z2);
nmos g6(y,gnd,z2);
endmodule
	module andgate_usn94test;
	reg x1, x2;
	
	andgate_usn94 g7(y,x1,x2);
	
	initial
	begin
	#0 x1 =0; x2 =0;
	#2 x1 =0; x2 =1;
	#3 x1 =1; x2 =0;
	#2 x1 =1; x2 =1;
	end
	initial
	$monitor($time,"x1=%d,x2=%d,y=%d",x1,x2,y);
	initial
	#15 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, andgate_usn94test);
	end
	endmodule

NAND(SW) EDA

DESIGN CODE	TEST BENCH

module nandgate_usn94(y,x1,x2);
output y;
input x1, x2;
supply1 vdd;
supply0 gnd;
wire z;
pmos g1(y,vdd,x1);
pmos g2(y,vdd,x2);
nmos g3(y,z,x1);
nmos g4(z,gnd,x2);
endmodule
	module nandgate_usn94test;
	reg x1, x2;
	
	nandgate_usn94 g5(y,x1,x2);
	
	initial
	begin
	#0 x1 =0; x2 =0;
	#2 x1 =0; x2 =1;
	#3 x1 =1; x2 =0;
	#2 x1 =1; x2 =1;
	end
	initial
	$monitor($time,"x1=%d,x2=%d,y=%d",x1,x2,y);
	initial
	#15 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, nandgate_usn94test);
	end
	endmodule

NOR(SW) EDA

DESIGN CODE	TEST BENCH

module norgate_usn94(y,x1,x2);
output y;
input x1, x2;
supply1 vdd;
supply0 gnd;
wire z;
 	
pmos g1(z,vdd,x1);
pmos g2(y,z,x2);
nmos g3(y,gnd,x1);
nmos g4(y,gnd,x2);
endmodule
	module norgate_usn94test;
	reg x1, x2;
	
	norgate_usn94 g5(y,x1,x2);
	
	initial
	begin
	#0 x1 =0; x2 =0;
	#2 x1 =0; x2 =1;
	#3 x1 =1; x2 =0;
	#2 x1 =1; x2 =1;
	end
	initial
	$monitor($time,"x1=%d,x2=%d,y=%d",x1,x2,y);
	initial
	#15 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, norgate_usn94test);
	end
	endmodule

NOT(SW) EDA

DESIGN CODE	TEST BENCH

module invgate_usn94(y,x);
  input x;
  output y;
  wire y;
  supply1 vdd;
  supply0 gnd;
  pmos g1(y,vdd,x);
  nmos g2(y,gnd,x);
endmodule
	module invgate_usn94test;
	  reg x;
	  wire y;
	
	  invgate_usn94 g3(y,x);
	
	  initial
	  begin
	  #0 x=0;
	  #2 x=1;
	  #3 x=0;
	  #2 x=1;
	  #3 x=0;
	
	  end
	  initial
	  $monitor($time,"x=%d,y=%d",x,y);
	  initial
	  #15 $stop;
	 initial
	  begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1, invgate_usn94test);
	  end
	endmodule

OR(SW) EDA

DESIGN CODE	TEST BENCH

module orgate_usn94(y,x1,x2);
output y;
input x1, x2;
supply1 vdd;
supply0 gnd;
wire z1,z2;
 	
pmos g1(z1,vdd,x1);
pmos g2(z2,z1,x2);
nmos g3(z2,gnd,x1);
nmos g4(z2,gnd,x2);
pmos g5(y,vdd,z2);
nmos g6(y,gnd,z2);
endmodule
	module orgate_usn94 test;
	reg x1, x2;
	
	orgate_usn94 g7(y,x1,x2);
	
	initial
	begin
	#0 x1 =0; x2 =0;
	#2 x1 =0; x2 =1;
	#3 x1 =1; x2 =0;
	#2 x1 =1; x2 =1;
	end
	initial
	$monitor($time,"x1=%d,x2=%d,y=%d",x1,x2,y);
	initial
	#15 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, orgate_usn94test);
	end
	endmodule

XNOR(SW) EDA

DESIGN CODE	TEST BENCH

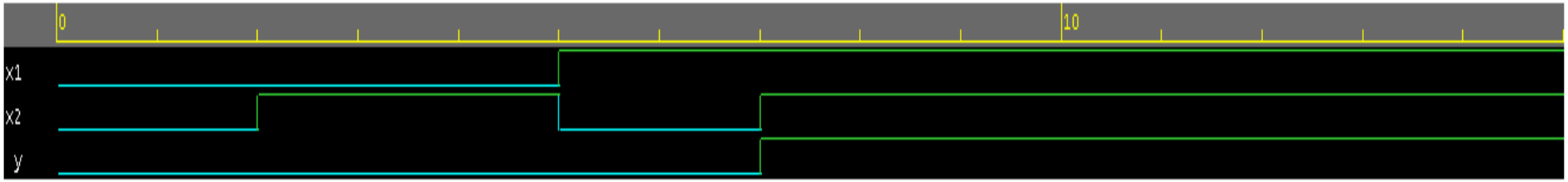
module xnorgate_usn94(y,x1,x2);
output y;
input x1, x2;
wire z1,z2,z3,z4;
supply1 vdd;
supply0 gnd;
inv g1(x1inv,x1);
inv g2(x2inv,x2);
pmos g3(z1,vdd,x1);
pmos g4(y,z1,x2);
pmos g5(z2,vdd,x1inv);
pmos g6(y,z2,x2inv);
nmos g7(y,z3,x1);
nmos g8(z3,gnd,x2inv);
nmos g9(y,z4,x1inv);
nmos g10(z4,gnd,x2);
endmodule
module inv(y,x);
output y;
input x;
supply1 vdd;
supply0 gnd;
pmos(y,vdd,x);
nmos(y,gnd,x);
endmodule
	module xnorgate_usn94test;
	reg x1, x2;
	
	xnorgate_usn94 g11(y,x1,x2);
	
	initial
	begin
	#0 x1 =0; x2 =0;
	#2 x1 =0; x2 =1;
	#3 x1 =1; x2 =0;
	#2 x1 =1; x2 =1;
	end
	initial
	$monitor($time,"x1=%d,x2=%d,y=%d",x1,x2,y);
	initial
	#15 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, xnorgate_usn94test);
	end
	endmodule

XOR(SW) EDA

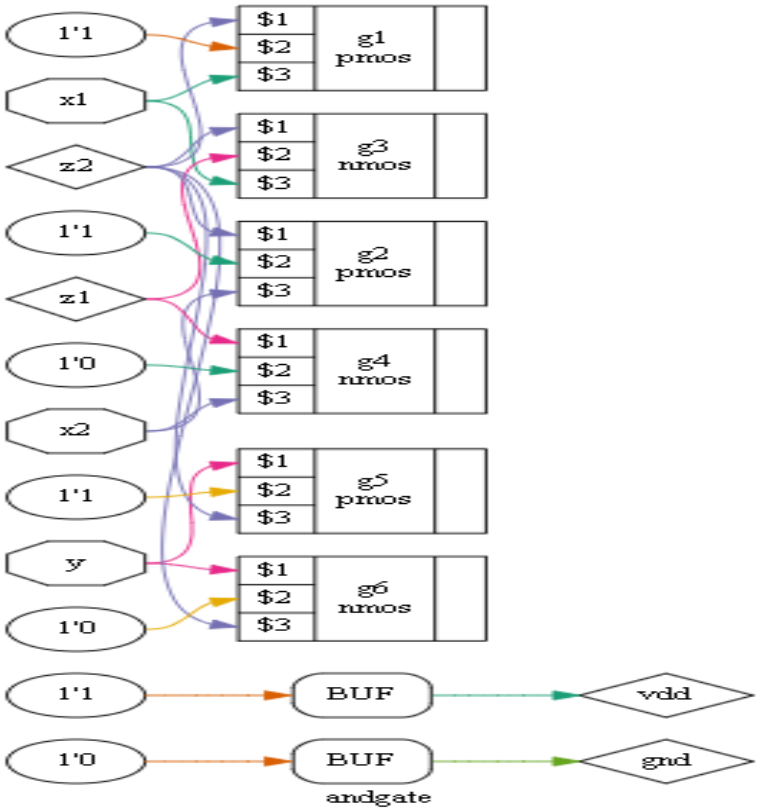
DESIGN CODE	TEST BENCH

module xorgate_usn94(y,x1,x2);
output y;
input x1, x2;
wire z1,z2,z3,z4;
supply1 vdd;
supply0 gnd;
inv g1(x1inv,x1);
inv g2(x2inv,x2);
pmos g3(z1,vdd,x1);
pmos g4(y,z1,x2inv);
pmos g5(z2,vdd,x1inv);
pmos g6(y,z2,x2);
nmos g7(y,z3,x1);
nmos g8(z3,gnd,x2);
nmos g9(y,z4,x1inv);
nmos g10(z4,gnd,x2inv);
endmodule
module inv(y,x);
output y;
input x;
supply1 vdd;
supply0 gnd;
pmos(y,vdd,x);
nmos(y,gnd,x);
endmodule
	module xorgate_usn94test;
	reg x1, x2;
	
	xorgate_usn94 g11(y,x1,x2);
	
	initial
	begin
	#0 x1 =0; x2 =0;
	#2 x1 =0; x2 =1;
	#3 x1 =1; x2 =0;
	#2 x1 =1; x2 =1;
	end
	initial
	$monitor($time,"x1=%d,x2=%d,y=%d",x1,x2,y);
	initial
	#15 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, xorgate_usn94test);
	end
	endmodule

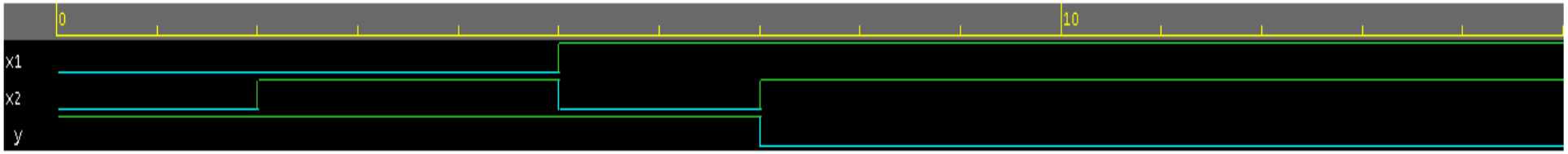
AND(SW) EDA OUTPUT



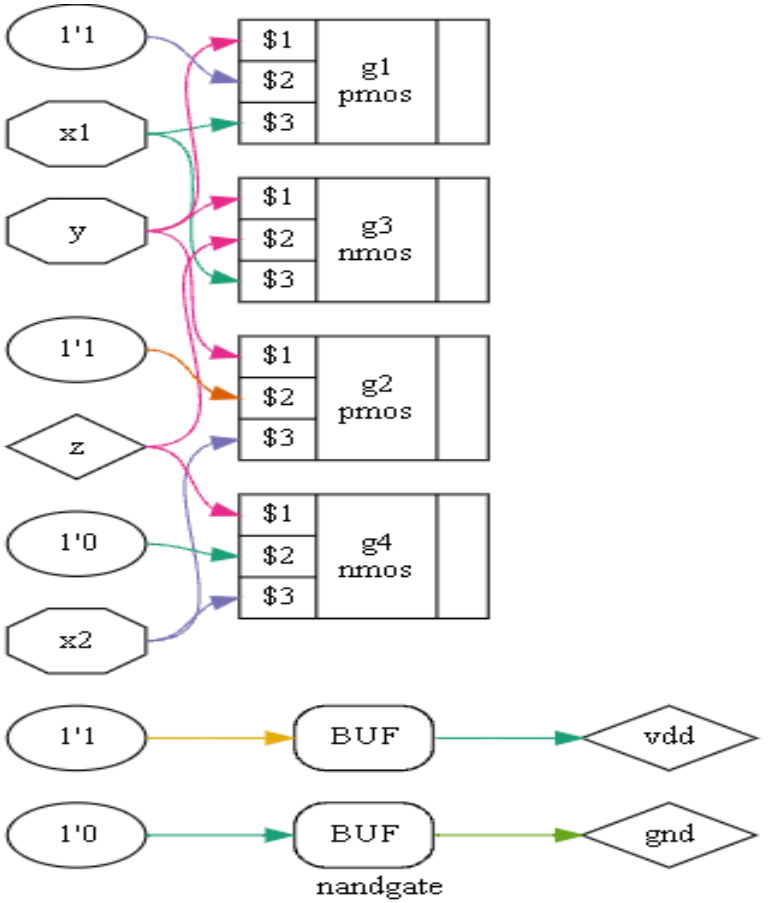

```
//
// Verilog description for cell andgate,
// Mon Oct 18 08:46:19 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module andgate ( y, x1, x2 ) ;

output y ;
input x1 ;
input x2 ;

wire x1_int, x2_int, nx58736z1;



OBUF y_obuf (.O (y), .I (nx58736z1)) ;
IBUF x2_ibuf (.O (x2_int), .I (x2)) ;
IBUF x1_ibuf (.O (x1_int), .I (x1)) ;
LUT2 ix58736z1322 (.O (nx58736z1), .I0 (x1_int), .I1 (x2_int)) ;
defparam ix58736z1322.INIT = 4'h8;
endmodule
```



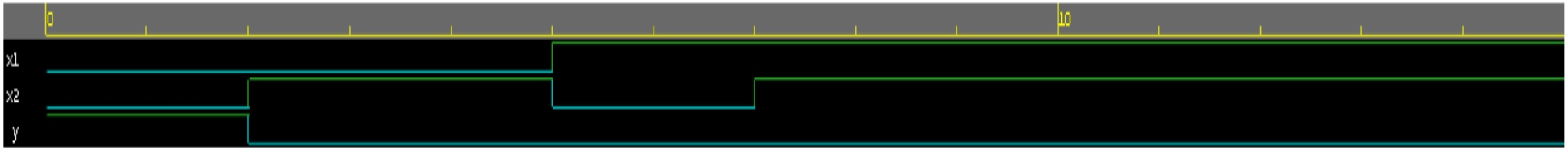
NAND(SW) EDA OUTPUT



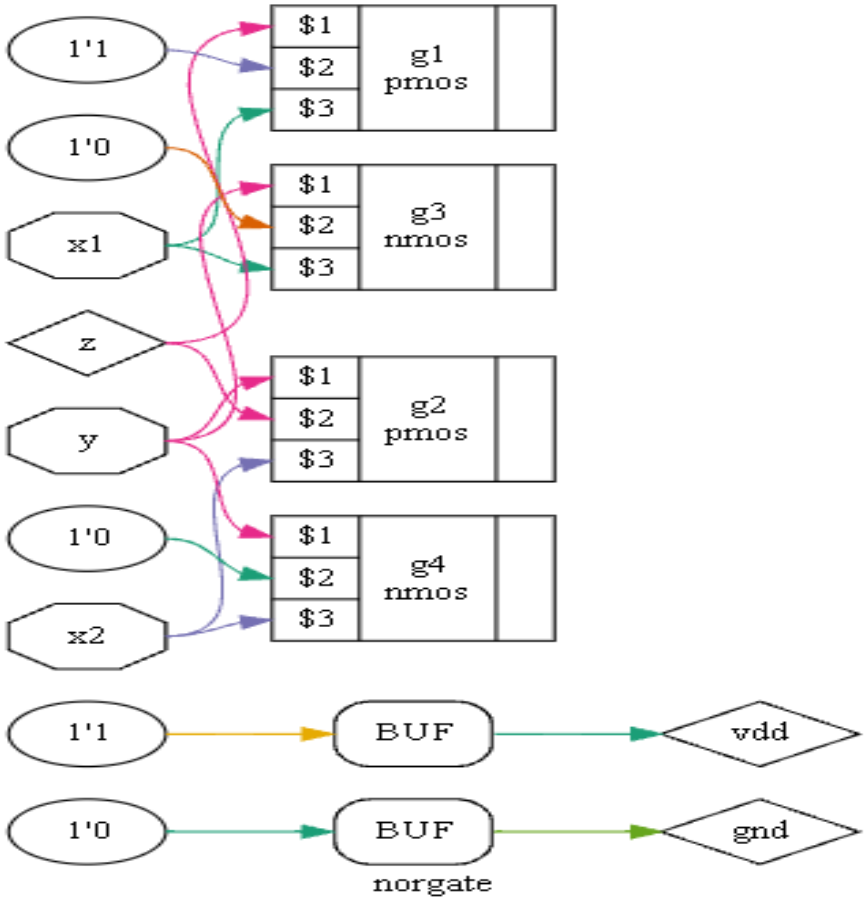

```
//
// Verilog description for cell nandgate,
// Mon Oct 18 09:18:49 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module nandgate ( y, x1, x2 ) ;

output y ;
input x1 ;
input x2 ;

wire x1_int, x2_int, nx58736z1;



OBUF y_obuf (.O (y), .I (nx58736z1)) ;
IBUF x2_ibuf (.O (x2_int), .I (x2)) ;
IBUF x1_ibuf (.O (x1_int), .I (x1)) ;
LUT2 ix58736z1321 (.O (nx58736z1), .I0 (x1_int), .I1 (x2_int)) ;
defparam ix58736z1321.INIT = 4'h7;
endmodule
```



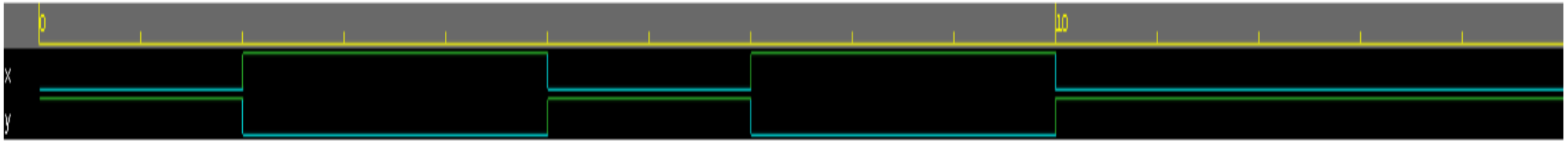
NOR(SW) EDA OUTPUT



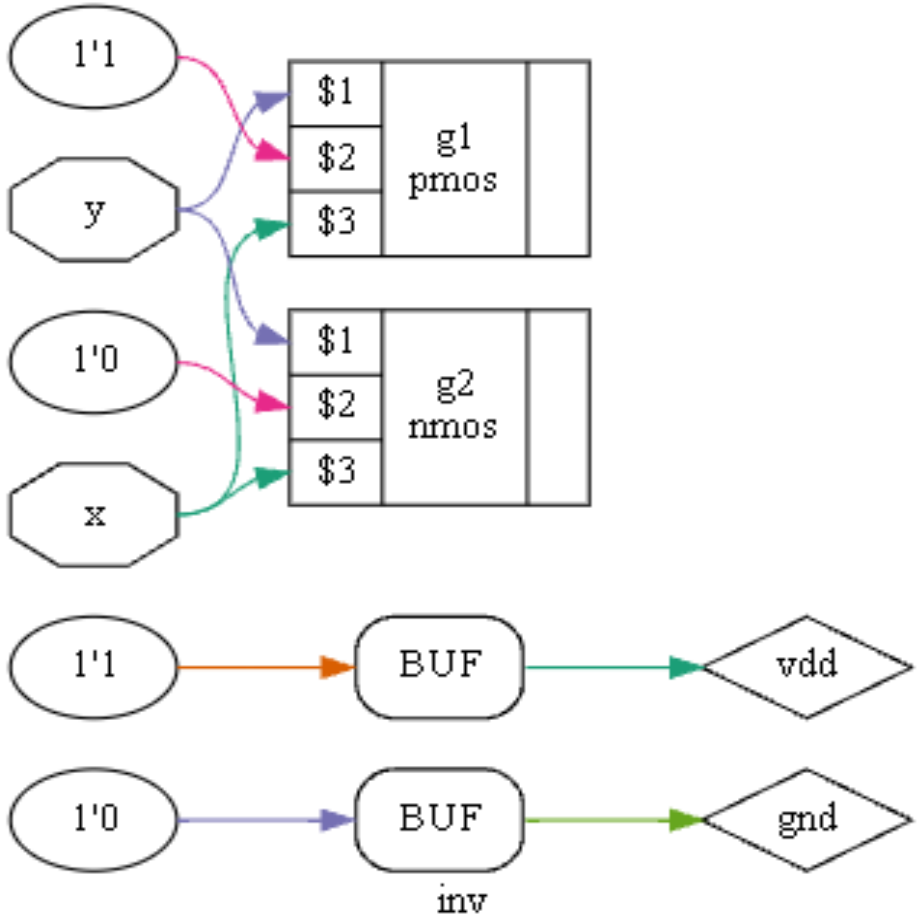

```
//
// Verilog description for cell norgate,
// Mon Oct 18 09:27:37 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module norgate ( y, x1, x2 ) ;

output y ;
input x1 ;
input x2 ;

wire x2_int, y_1_0;



OBUF y_obuf (.O (y), .I (y_1_0)) ;
IBUF x2_ibuf (.O (x2_int), .I (x2)) ;
INV ix58736z1315 (.O (y_1_0), .I (x2_int)) ;
endmodule
```



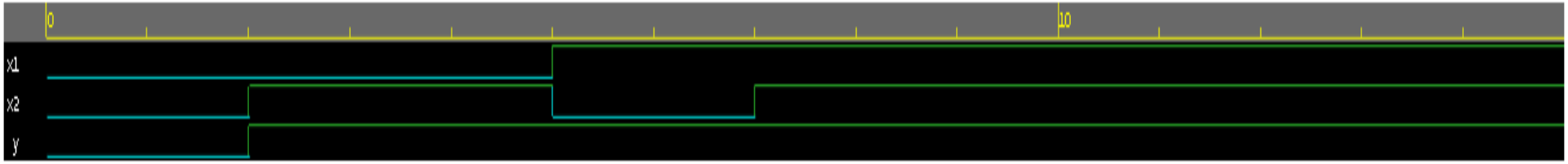
NOT(SW) EDA OUTPUT



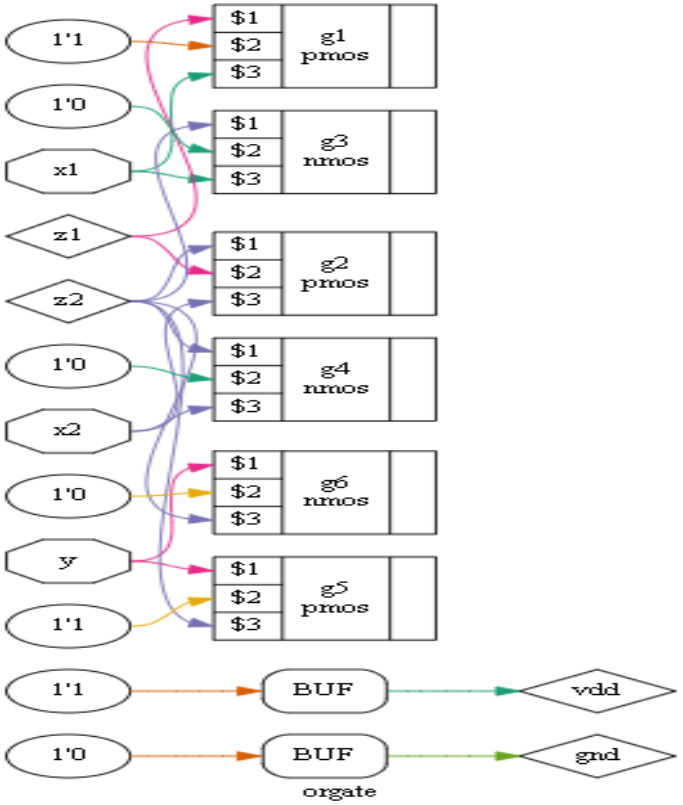

```
//
// Verilog description for cell inv,
// Mon Oct 18 09:35:10 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module inv ( y, x ) ;

output y ;
input x ;

wire x_int, y_1_0;



OBUF y_obuf (.O (y), .I (y_1_0)) ;
IBUF x_ibuf (.O (x_int), .I (x)) ;
INV ix58736z1315 (.O (y_1_0), .I (x_int)) ;
endmodule
```



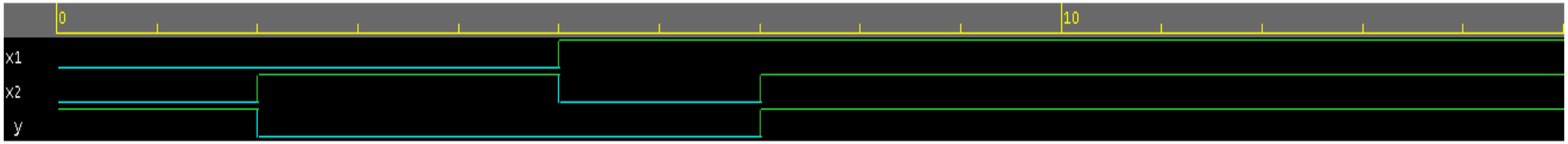
OR(SW) EDA OUTPUT




```
//
// Verilog description for cell orgate,
// Mon Oct 18 09:39:23 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module orgate ( y, x1, x2 ) ;

output y ;
input x1 ;
input x2 ;

wire x2_int;



OBUF y_obuf (.O (y), .I (x2_int)) ;
IBUF x2_ibuf (.O (x2_int), .I (x2)) ;
endmodule
```

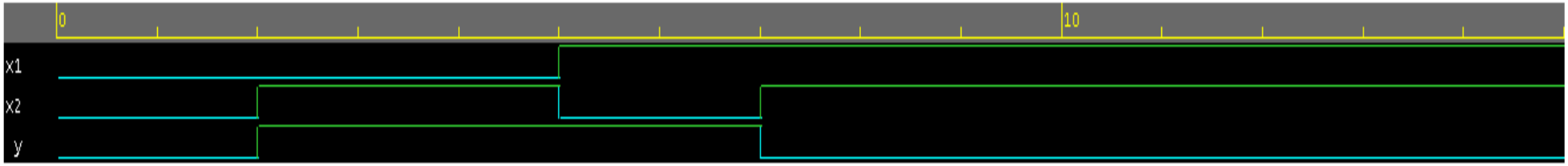


XNOR(SW) EDA OUTPUT




```
//
// Verilog description for cell xnorgate,
// Mon Oct 18 09:46:36 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module xnorgate ( y, x1, x2 ) ;

output y ;
input x1 ;
input x2 ;

wire nx21046z1;



OBUF \[96]_obuf (.O (y), .I (nx21046z1)) ;
VCC ps_vcc (.P (nx21046z1)) ;
endmodule
```

XOR(SW) EDA OUTPUT




```
//
// Verilog description for cell xorgate,
// Mon Oct 18 09:54:31 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module xorgate ( y, x1, x2 ) ;

output y ;
input x1 ;
input x2 ;

wire nx21046z1;



OBUF \[96]_obuf (.O (y), .I (nx21046z1)) ;
VCC ps_vcc (.P (nx21046z1)) ;
endmodule
```

Date :
12-08-2021

Experiment No. 2

Half Adder, Full Adder & n-bit parallel adder

Aim : To implement half-adder, full-adder, and n-bit parallel adder using verilog, verify the design using Test bench and perform functional simulation and to synthesize the design

Requirements : EDA playground Tool.

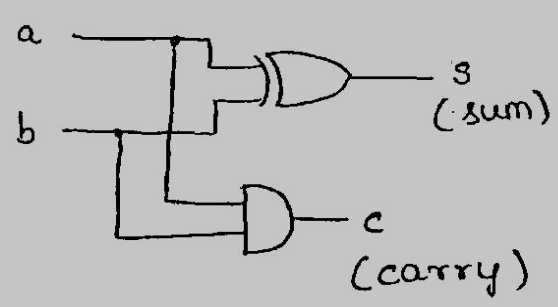
Theory :

=> Half-Adder :

Half-adder is a combinational logic circuit which is designed by connecting one EX-OR gate. The Half adder circuit has two inputs : a and b, which add two input digits & generates a carry and a sum.

S = 1 either if (A=0 and B=1) or if (A=1 and B=0) = A XOR B

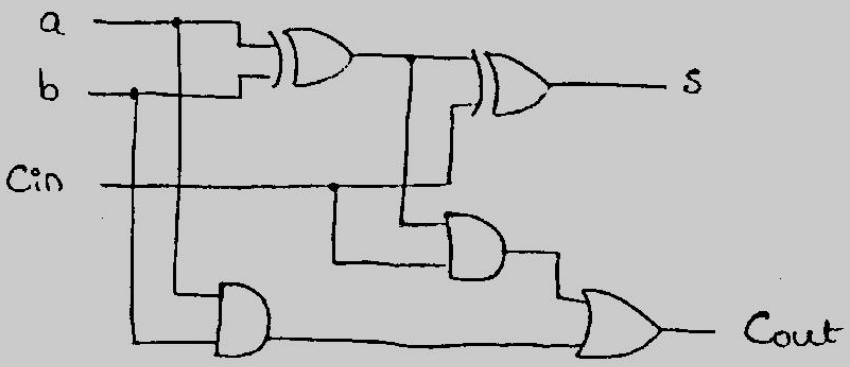
C=1 only if (A=1 and B=1) = A AND B



a	b	c	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

=> Full Adder:

Full adder is the circuit which consists of two EX-OR gates and two AND gates and one OR gate. Full adder is the adder which adds three inputs and produces two-outputs, the first two inputs are a and b and input 3rd carry as C-in. The output is designated as C-out and the normal output is designated as S which is Sum.



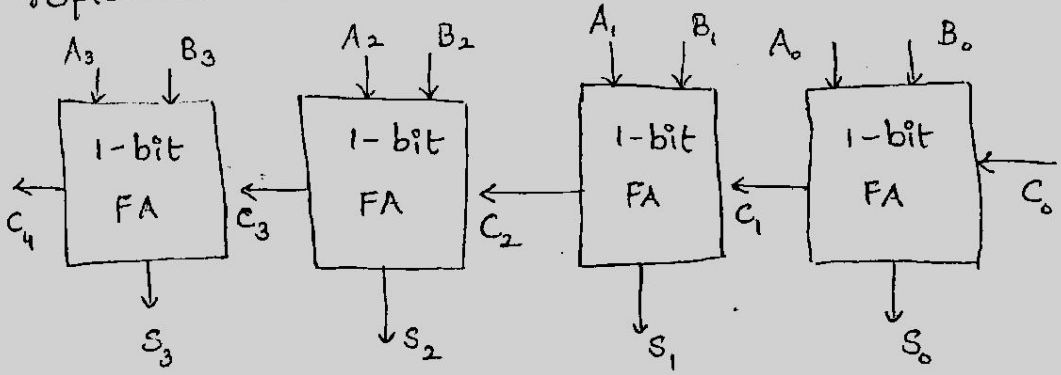
S = A XOR B XOR Cin

Cout = (A AND B) OR (Cin AND (A XOR B))

A	B	Cin	Cout	S
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

=> 4 - bit ripple carry Adder: (19-08-2021)

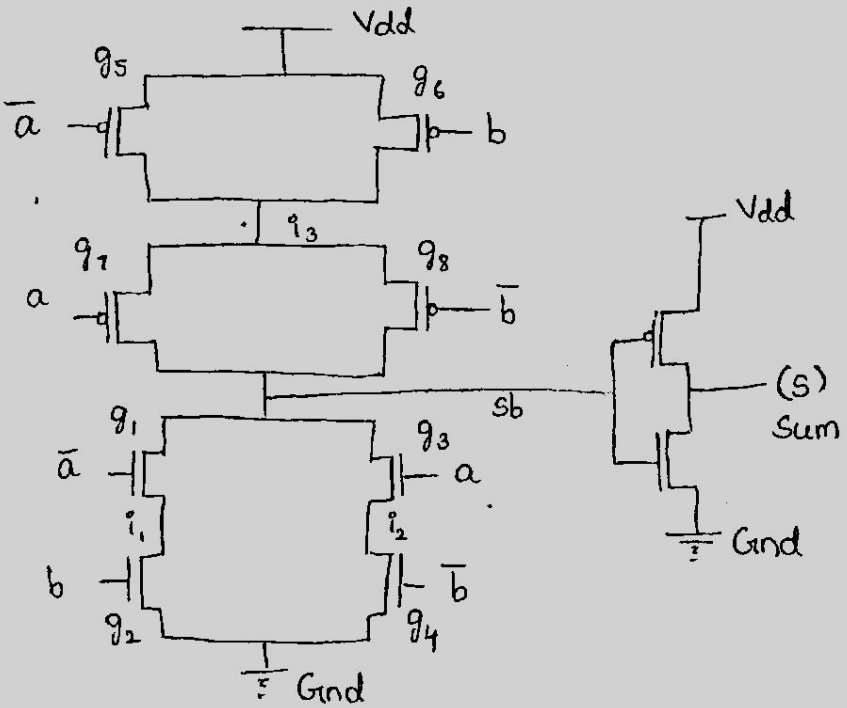
Multiple full adders can be used to create adders of greater bit lengths. Each full adder uses the Cout of the previous adder as its Cin. This kind of adder is a ripple carry adder, since the carry bits "ripple" through the full adder stages. Note tha the first (and only the first) full adder may be replaced by a half adder.



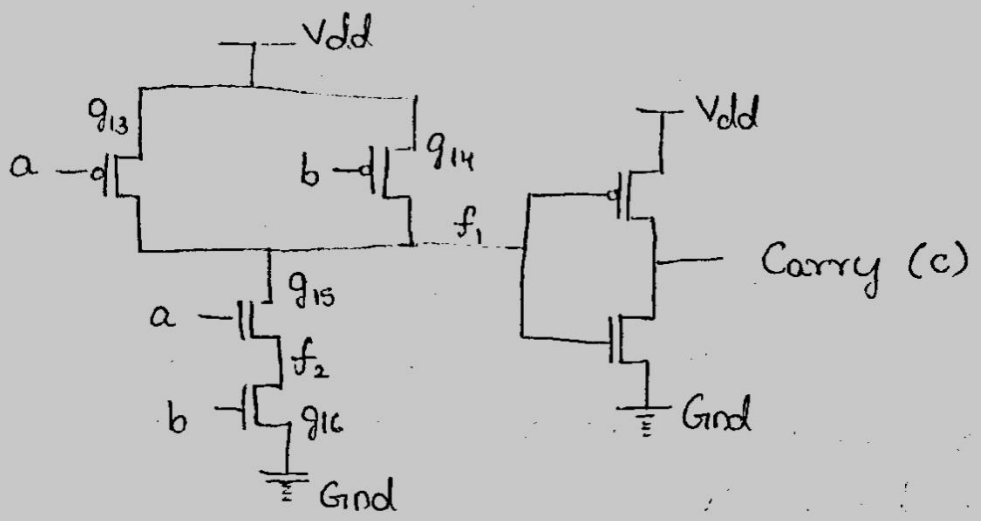
Circuit diagrams:

=> Half Adder (switch level):

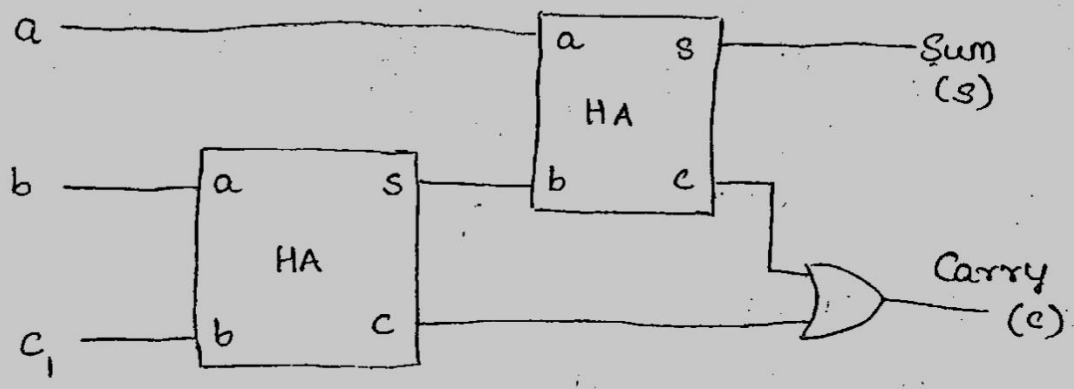
Sum:



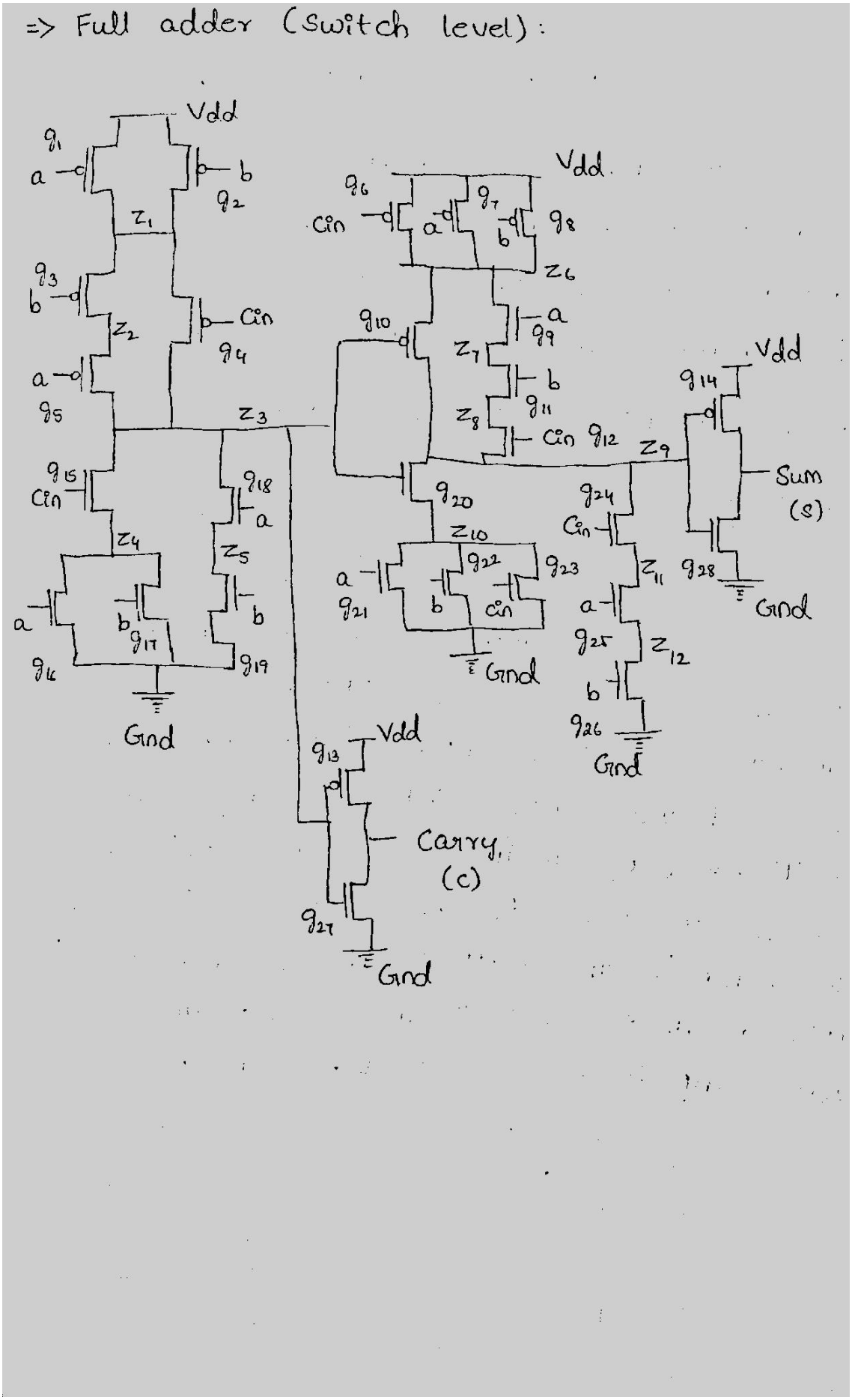
Carry:



=> Full adder using 2 Half adders



=> Full adder (switch level):



FA USING 2 HA (GL) EDA

DESIGN CODE	TEST BENCH

module fa_usn94(s,c,a,b,c1);
input a,b,c1;
output s,c;
wire s1,c2,c3;
xor(s,a,s1);
and(c2,a,s1);
xor(s1,b,c1);
and(c3,b,c1);
or(c,c2,c3);
endmodule
	module fa_test94;
	reg a,b,c1;
	wire s,c;
	fa_usn94 g1(s,c,a,b,c1);
	initial
	begin
	#0 a=0;b=0;c1=0;
	#2 a=0;b=0;c1=1;
	#3 a=0;b=1;c1=0;
	#2 a=0;b=1;c1=1;
	#3 a=1;b=0;c1=0;
	#2 a=1;b=0;c1=1;
	#3 a=1;b=1;c1=0;
	#2 a=1;b=1;c1=1;
	end
	initial
	$monitor ($time, " a=%d b=%d c1=%d s=%d c=%d",a,b,c1,s,c);
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, fa_test94);
	end
	endmodule

FA USING 2 HA (SW) EDA

DESIGN CODE	TEST BENCH

module fa_usn94(s,c,a,b,c1);
input a,b,c1;
output s,c;
wire s1,c2,c3;
ha_usn94 g1(s1,c2,a,b);
ha_usn94 g2(s,c3,s1,c1);
org_usn94 g3(c,c2,c3);
endmodule
module org_usn94(y,a,b);
input b,a;
output y;
supply0 gnd;
supply1 pwr;
wire w1,w2;
pmos g1(w1,pwr,a);
pmos g2(y1,w1,b);
nmos g3(y1,gnd,a);
nmos g4(y1,gnd,b);
inv_usn94 g5(y,y1);
endmodule
module ha_usn94(s,c,a,b);
input a,b;
output s,c;
wire i1,i2,i3,ab,bb,sb;
supply1 vdd;
supply0 gnd;
nmos g1(sb,i1,ab);
nmos g2(i1,gnd,b);
nmos g3(sb,i2,a);
nmos g4(i2,gnd,bb);
pmos g9(i3,vdd,ab);
pmos g10(i3,vdd,b);
pmos g11(sb,i3,a);
pmos g12(sb,i3,bb);
andg_usn94 g13(c,a,b);
inv_usn94 g5(ab,a);
inv_usn94 g6(bb,b);
inv_usn94 g7(s,sb);
endmodule
module andg_usn94(z,x,y);
input x,y;
output z;
wire f,a;
supply1 vdd;
supply0 gnd;
pmos g1(f,vdd,x);
pmos g2(f,vdd,y);
nmos g3(f,a,x);
nmos g4(a,gnd,y);
pmos g5(z,vdd,f);
	module fa_test94;
	reg a,b,c1;
	wire s,c;
	fa_usn94 g1(s,c,a,b,c1);
	initial
	begin
	#0 a=0;b=0;c1=0;
	#2 a=0;b=0;c1=1;
	#3 a=0;b=1;c1=0;
	#2 a=0;b=1;c1=1;
	#3 a=1;b=0;c1=0;
	#2 a=1;b=0;c1=1;
	#3 a=1;b=1;c1=0;
	#2 a=1;b=1;c1=1;
	end
	initial
	$monitor ($time, " a=%d b=%d c1=%d s=%d c=%d",a,b,c1,s,c);
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, fa_test94);
	end
	endmodule

```
  nmos g6(z,gnd,f);
endmodule
module inv_usn94(y,a);
  input a; output y;
  supply1 vdd;
  supply0 gnd;
  pmos g13(y,vdd,a);
  nmos g14(y,gnd,a);
endmodule
```

FA(BEHAVIOURAL) EDA

DESIGN CODE

```
module fa_usn94(s,c,a,b,cin);
  input a,b,cin;
  output reg s,c;
  always@(*)
    begin
      s=a ^ b ^ cin;
      c=(a & b)|(cin & (a ^ b));
    end
endmodule
```

TEST BENCH

```
module fa_test94;
reg a,b,c1;
wire s,c;
  fa_usn94 g1(s,c,a,b,c1);
    initial
    begin
    #0 a=0;b=0;c1=0;
    #2 a=0;b=0;c1=1;
    #3 a=0;b=1;c1=0;
    #2 a=0;b=1;c1=1;
    #3 a=1;b=0;c1=0;
    #2 a=1;b=0;c1=1;
    #3 a=1;b=1;c1=0;
    #2 a=1;b=1;c1=1;
    end
initial
  $monitor ($time,  " a=%d b=%d c1=%d s=%d
c=%d",a,b,c1,s,c);
            initial
                #25 $stop;
initial
begin
   $dumpfile("dump.vcd");
  $dumpvars(1, fa_test94);
  end
endmodule
```

FA(DF) EDA

DESIGN CODE	TEST BENCH

module fa_usn94(s,c,a,b,cin);
input a,b,cin;
output s,c;
assign s = a ^ b ^ cin;
assign c = (a & b)|(cin & (a ^ b));
endmodule
	module fa_test94;
	reg a,b,cin;
	wire s,c;
	fa_usn94 dut1(s,c,a,b,cin);
	initial
	begin
	#0 a=0; b=0; cin=0;
	#2 a=0; b=0; cin=1;
	#3 a=0; b=1; cin=0;
	#2 a=0; b=1; cin=1;
	#3 a=1; b=0; cin=0;
	#2 a=1; b=0; cin=1;
	#3 a=1; b=1; cin=0;
	#2 a=1; b=1; cin=1;
	end
	initial
	$monitor($time,"a=%d, b=%d, cin=%d, s=%d, c=%d",a,b,cin,s,c);
	initial
	#20 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, fa_test94);
	end
	endmodule

FA(FUNC) EDA

DESIGN CODE	TEST BENCH

module fa_usn94(s,c,a,b,cin);
input a,b,cin;
output reg s,c;
always @ *
begin
{s,c}=fa(a,b,cin);
end
function[1:0]fa;
input a,b,cin;
begin
assign s = a ^ b ^ cin;
assign c = (a & b)|(cin & (a ^ b));
end
endfunction
endmodule
	module fa_test94;
	reg a,b,cin;
	wire s,c;
	fa_usn94 dut1(s,c,a,b,cin);
	initial
	begin
	#0 a=0; b=0; cin=0;
	#2 a=0; b=0; cin=1;
	#3 a=0; b=1; cin=0;
	#2 a=0; b=1; cin=1;
	#3 a=1; b=0; cin=0;
	#2 a=1; b=0; cin=1;
	#3 a=1; b=1; cin=0;
	#2 a=1; b=1; cin=1;
	end
	initial
	$monitor($time,"a=%d, b=%d, cin=%d, s=%d, c=%d",a,b,cin,s,c);
	initial
	#20 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, fa_test94);
	end
	endmodule

FA(GL) EDA

DESIGN CODE	TEST BENCH

module fa_usn94(s,c,a,b,cin);
input a,b,cin;
output s,c;
wire z1,z2,z3;
xor g1 (s,a,b,cin);
and(z1,a,b);
xor(z2,a,b);
and(z3,cin,z2);
or(c,z1,z3);
endmodule
	module fa_test94;
	reg a,b,cin;
	wire s,c;
	fa_usn94 dut1(s,c,a,b,cin);
	initial
	begin
	#0 a=0; b=0; cin=0;
	#2 a=0; b=0; cin=1;
	#3 a=0; b=1; cin=0;
	#2 a=0; b=1; cin=1;
	#3 a=1; b=0; cin=0;
	#2 a=1; b=0; cin=1;
	#3 a=1; b=1; cin=0;
	#2 a=1; b=1; cin=1;
	end
	initial
	$monitor($time,"a=%d, b=%d, cin=%d, s=%d, c=%d",a,b,cin,s,c);
	initial
	#20 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, fa_test94);
	end
	endmodule

FA(SW) EDA

DESIGN CODE	TEST BENCH

module fa_usn94(s,c,a,b,cin);
  input a,b,cin;
  output s,c;
  supply1 vdd;
  supply0 gnd;
  wire [12:1]z;
  pmos g1(z[1],vdd,a);
  pmos g2(z[1],vdd,b);
  pmos g3(z[2],z[1],b);
  pmos g4(z[3],z[1],cin);
  pmos g5(z[3],z[2],a);
  pmos g6(z[6],vdd,cin);
  pmos g7(z[6],vdd,a);
  pmos g8(z[6],vdd,b);
  pmos g9(z[7],z[6],a);
  pmos g10(z[9],z[6],z[3]);
  pmos g11(z[8],z[7],b);
  pmos g12(z[9],z[8],cin);
  pmos g13(c,vdd,z[3]);
  pmos g14(s,vdd,z[9]);
  nmos g15(z[3],z[4],cin);
  nmos g16(z[4],gnd,a);
  nmos g17(z[4],gnd,b);
  nmos g18(z[3],z[5],a);
  nmos g19(z[5],gnd,b);
  nmos g20(z[9],z[10],z[3]);
  nmos g21(z[10],gnd,a);
  nmos g22(z[10],gnd,b);
  nmos g23(z[10],gnd,cin);
  nmos g24(z[9],z[11],cin);
  nmos g25(z[11],z[12],a);
  nmos g26(z[12],gnd,b);
  nmos g27(c,gnd,z[3]);
  nmos g28(s,gnd,z[9]);
endmodule
	module fa_test94;
	 reg a,b,cin;
	 wire s,c;
	
	  fa_usn94 dut1(s,c,a,b,cin);
	
	  initial
	  begin
	  #0 a=0; b=0; cin=0;
	  #2 a=0; b=0; cin=1;
	  #3 a=0; b=1; cin=0;
	  #2 a=0; b=1; cin=1;
	  #3 a=1; b=0; cin=0;
	  #2 a=1; b=0; cin=1;
	  #3 a=1; b=1; cin=0;
	  #2 a=1; b=1; cin=1;
	  end
	  initial
	  $monitor($time,"a=%d, b=%d, cin=%d, s=%d, c=%d",a,b,cin,s,c);
	  initial
	  #20 $stop;
	
	 initial
	  begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1, fa_test94);
	  end
	endmodule

FA(TASK) EDA

DESIGN CODE	TEST BENCH

module fa_usn94(s,c,a,b,cin);
input a,b,cin;
output reg s,c;
always @ *
begin
FAtsk_94(a,b,cin);
end
task FAtsk_94;
input a,b,cin;
begin
assign s = a ^ b ^ cin;
assign c = (a & b)|(cin & (a ^ b));
end
endtask
endmodule
	module fa_test94;
	reg a,b,cin;
	wire s,c;
	
	fa_usn94 dut1(s,c,a,b,cin);
	
	initial
	begin
	#0 a=0; b=0; cin=0;
	#2 a=0; b=0; cin=1;
	#3 a=0; b=1; cin=0;
	#2 a=0; b=1; cin=1;
	#3 a=1; b=0; cin=0;
	#2 a=1; b=0; cin=1;
	#3 a=1; b=1; cin=0;
	#2 a=1; b=1; cin=1;
	end
	initial
	$monitor($time,"a=%d, b=%d, cin=%d, s=%d, c=%d",a,b,cin,s,c);
	initial
	#20 $stop;
	
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, fa_test94);
	end
	endmodule

HA(BEHAVIOURAL) EDA

DESIGN CODE	TEST BENCH

module ha_usn94(s,c,a,b);
  input a,b;
  output reg s,c;
  always@(*)
  begin
  s=a^b;
  c=a&b;
  end
endmodule
	module ha_test94;
	 reg a,b;
	 wire s,c;
	
	 ha_usn94 dut1(s,c,a,b);
	
	 initial
	  begin
	  #0 a=0; b=0;
	  #2 a=0; b=1;
	  #3 a=1; b=0;
	  #2 a=1; b=1;
	  end
	 initial
	  $monitor($time,"a=%d, b=%d, s=%d, c=%d",a,b,s,c);
	 initial
	  #15 $stop;
	
	 initial
	 begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1, ha_test94);
	 end
	endmodule

HA(DF) EDA

DESIGN CODE	TEST BENCH

module ha_usn94(s,c,a,b);
  input a,b;
  output s,c;
assign s = a ^ b;
assign c = a & b;
endmodule
	module ha_test94;
	 reg a,b;
	 wire s,c;
	
	  ha_usn94 dut1(s,c,a,b);
	
	  initial
	  begin
	  #0 a=0; b=0;
	  #2 a=0; b=1;
	  #3 a=1; b=0;
	  #2 a=1; b=1;
	  end
	 initial
	  $monitor($time,"a=%d, b=%d, s=%d, c=%d",a,b,s,c);
	 initial
	  #15 $stop;
	
	 initial
	 begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1, ha_test94);
	 end
	endmodule

HA(GL) EDA

DESIGN CODE	TEST BENCH

module ha_usn94(s,c,a,b);
  input a,b;
  output s,c;
  xor g1(s,a,b);
  and g2(c,a,b);
endmodule
	module ha_test94;
	 reg a,b;
	 wire s,c;
	
	  ha_usn94 dut1(s,c,a,b);
	
	  initial
	  begin
	  #0 a=0; b=0;
	  #2 a=0; b=1;
	  #3 a=1; b=0;
	  #2 a=1; b=1;
	  end
	 initial
	  $monitor($time,"a=%d, b=%d, s=%d, c=%d",a,b,s,c);
	 initial
	  #15 $stop;
	
	 initial
	 begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1, ha_test94);
	 end
	endmodule

HA(SW) EDA

DESIGN CODE	TEST BENCH

module ha_usn94(s,c,a,b);
  input a,b;
  output s,c;
  wire i1,i2,i3,ab,bb,sb;
  supply1 vdd;
  supply0 gnd;
 	
  nmos g1(sb,i1,ab);
  nmos g2(i1,gnd,b);
  nmos g3(sb,i2,a);
  nmos g4(i2,gnd,bb);
  pmos g5(i3,vdd,ab);
  pmos g6(i3,vdd,b);
  pmos g7(sb,i3,a);
  pmos g8(sb,i3,bb);
  andg_usn94 g9(c,a,b);
  inv_usn94 g10(ab,a);
  inv_usn94 g11(bb,b);
  inv_usn94 g12(s,sb);
endmodule
 	
module andg_usn94(z,x,y);
  input x,y;
  output z;
  wire f1,f2;
  supply1 vdd;
  supply0 gnd;
 	
  pmos g13(f1,vdd,x);
  pmos g14(f1,vdd,y);
  nmos g15(f1,f2,x);
  nmos g16(f2,gnd,y);
  inv_usn94 g17(z,f1);
endmodule
 	
module inv_usn94(y,a);
  input a;
  output y;
  supply1 vdd;
  supply0 gnd;
 	
  pmos g18(y,vdd,a);
  nmos g19(y,gnd,a);
endmodule
	module ha_test94;
	 reg a,b;
	 wire s,c;
	
	  ha_usn94 dut1(s,c,a,b);
	
	  initial
	  begin
	  #0 a=0; b=0;
	  #2 a=0; b=1;
	  #3 a=1; b=0;
	  #2 a=1; b=1;
	  end
	 initial
	  $monitor($time,"a=%d, b=%d, s=%d, c=%d",a,b,s,c);
	 initial
	  #15 $stop;
	
	 initial
	 begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1, ha_test94);
	 end
	endmodule

RCA(FUNC) EDA

DESIGN CODE	TEST BENCH

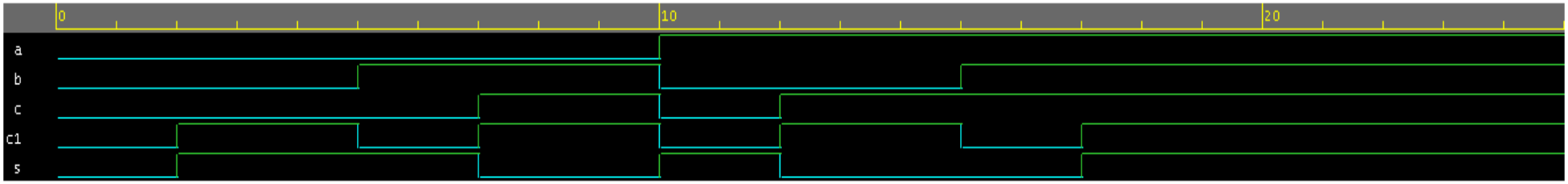
module rca_94(s,c,a,b,cin);
output reg [3:0] s;
output reg [3:0]c;
input [3:0] a;
input [3:0] b;
input cin;
fa_usn94 g1(s[0],c[0],a[0],b[0],cin);
fa_usn94 g2(s[1],c[1],a[1],b[1],c[0]);
fa_usn94 g3(s[2],c[2],a[2],b[2],c[1]);
fa_usn94 g4(s[3],c[3],a[3],b[3],c[2]);
endmodule
module fa_usn94(s,c,a,b,cin);
input a,b,cin;
output reg s,c;
always @ *
begin
{s,c}=fa(a,b,cin);
end
function[1:0]fa;
input a,b,cin;
begin
assign s = a ^ b ^ cin;
assign c = (a & b)|(cin & (a ^ b));
end
endfunction
endmodule
	module rca_test94;
	wire [3:0] s;
	wire [3:0] c;
	reg [3:0] a;
	reg [3:0] b;
	reg cin;
	
	rca_94 dut1(s,c,a,b,cin);
	
	initial
	begin
	#0 a=4'b0000; b=4'b0000; cin=0;
	#2 a=4'b0100; b=4'b0100;
	#3 a=4'b0101; b=4'b0101;
	#2 a=4'b1000; b=4'b1000;
	#3 a=4'b0110; b=4'b0110;
	#2 a=4'b1001; b=4'b1001;
	#3 a=4'b1010; b=4'b1010;
	#2 a=4'b1111; b=4'b1111;
	end
	initial
	$monitor($time,"a=%d, b=%d, cin=%d, s=%d, c=%d",a,b,cin,s,c);
	initial
	#20 $stop;
	
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, rca_test94);
	end
	endmodule

RCA(TASK) EDA

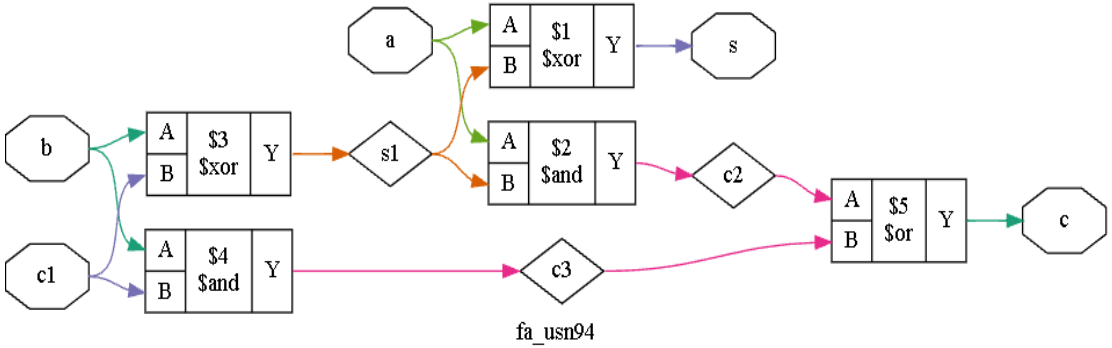
DESIGN CODE	TEST BENCH

module rca_94(s,c,a,b,cin);
  output reg [3:0] s;
  output reg [3:0]c;
  input [3:0] a;
  input [3:0] b;
  input cin;
  fa_usn94 g1(s[0],c[0],a[0],b[0],cin);
  fa_usn94 g2(s[1],c[1],a[1],b[1],c[0]);
  fa_usn94 g3(s[2],c[2],a[2],b[2],c[1]);
  fa_usn94 g4(s[3],c[3],a[3],b[3],c[2]);
endmodule
  module fa_usn94(s,c,a,b,cin);
  input a,b,cin;
  output reg s,c;
  always @ *
    begin
      FAtsk_94(a,b,cin);
    end
  task FAtsk_94;
    input a,b,cin;
    begin
  assign s = a ^ b ^ cin;
assign c = (a & b)|(cin & (a ^ b));
    end
  endtask
endmodule
	module rca_test94;
	  wire [3:0] s;
	  wire [3:0] c;
	  reg [3:0] a;
	  reg [3:0] b;
	  reg cin;
	
	  rca_94 dut1(s,c,a,b,cin);
	
	  initial
	    begin
	      #0 a=4'b0000; b=4'b0000; cin=0;
	      #2 a=4'b0100; b=4'b0100;
	      #3 a=4'b0101; b=4'b0101;
	      #2 a=4'b1000; b=4'b1000;
	      #3 a=4'b0110; b=4'b0110;
	      #2 a=4'b1001; b=4'b1001;
	      #3 a=4'b1010; b=4'b1010;
	      #2 a=4'b1111; b=4'b1111;
	    end
	  initial
	    $monitor($time,"a=%d, b=%d, cin=%d, s=%d, c=%d",a,b,cin,s,c);
	  initial
	    #20 $stop;
	
	 initial
	  begin
	    $dumpfile("dump.vcd");
	    $dumpvars(1, rca_test94);
	  end
	endmodule

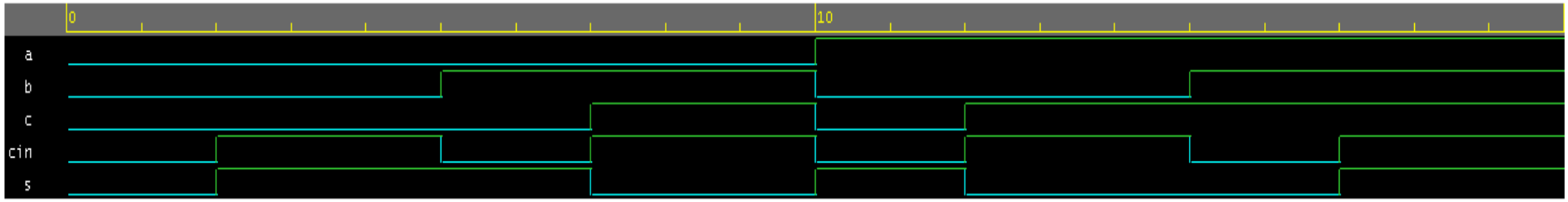
FA USING 2 HA EDA OUTPUT



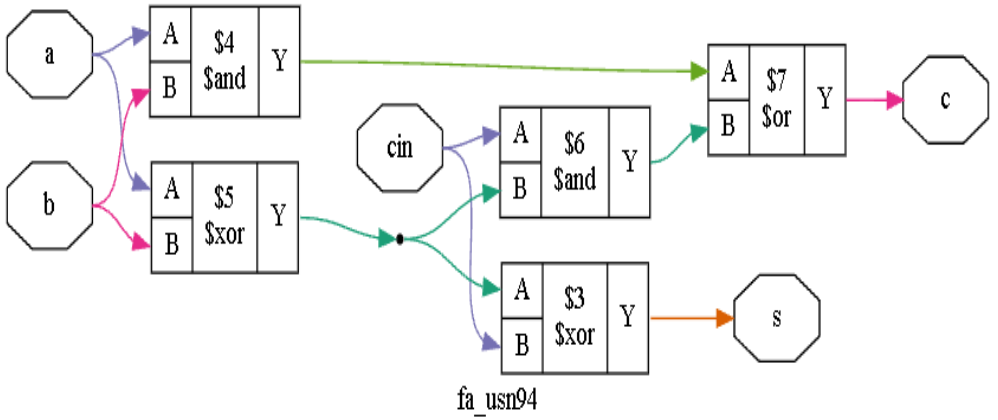

```
//
// Verilog description for cell fa_usn94,
// Mon Oct 18 09:59:15 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module fa_usn94 ( s, c, a, b, c1 ) ;

output s ;
output c ;
input a ;
input b ;
input c1 ;

wire a_int, b_int, c1_int, nx53522z1, nx18494z1;


OBUF c_obuf (.O (c), .I (nx18494z1)) ;
OBUF s_obuf (.O (s), .I (nx53522z1)) ;
IBUF c1_ibuf (.O (c1_int), .I (c1)) ;
IBUF b_ibuf (.O (b_int), .I (b)) ;
IBUF a_ibuf (.O (a_int), .I (a)) ;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix53522z1464 (.O (nx53522z1), .I0 (a_int), .I1 (b_int), .I2 (c1_int)) ;
defparam ix53522z1464.INIT = 8'h96;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix18494z1546 (.O (nx18494z1), .I0 (a_int), .I1 (b_int), .I2 (c1_int)) ;
defparam ix18494z1546.INIT = 8'hE8;
endmodule
```



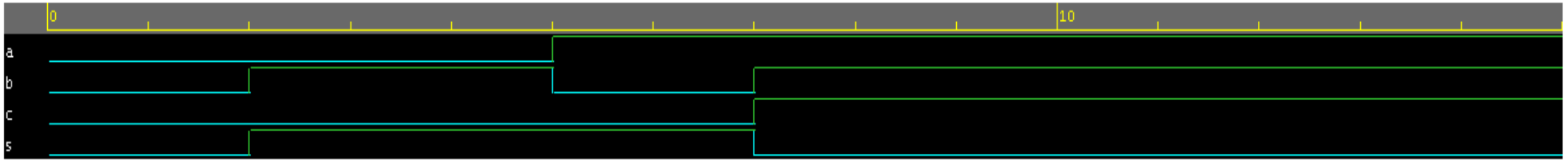
FA EDA OUTPUT



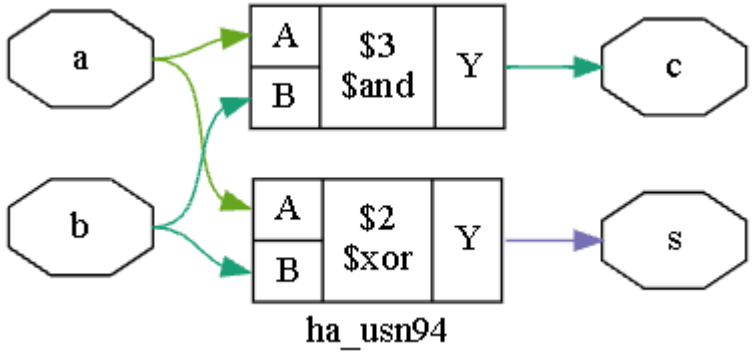

```
//
// Verilog description for cell fa_usn94,
// Mon Oct 18 10:23:55 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module fa_usn94 ( s, c, a, b, cin ) ;

output s ;
output c ;
input a ;
input b ;
input cin ;

wire a_int, b_int, cin_int, nx53522z1, nx18494z1;



OBUF c_obuf (.O (c), .I (nx18494z1)) ;
OBUF s_obuf (.O (s), .I (nx53522z1)) ;
IBUF cin_ibuf (.O (cin_int), .I (cin)) ;
IBUF b_ibuf (.O (b_int), .I (b)) ;
IBUF a_ibuf (.O (a_int), .I (a)) ;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix53522z1464 (.O (nx53522z1), .I0 (a_int), .I1 (b_int), .I2 (cin_int)
) ;
defparam ix53522z1464.INIT = 8'h96;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix18494z1546 (.O (nx18494z1), .I0 (a_int), .I1 (b_int), .I2 (cin_int)
) ;
defparam ix18494z1546.INIT = 8'hE8;
endmodule
```



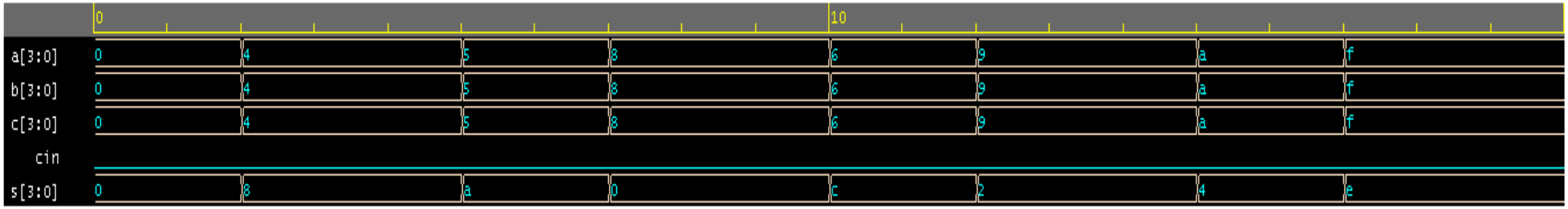
HA EDA OUTPUT




```
//
// Verilog description for cell ha_usn94,
// Mon Oct 18 10:32:04 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module ha_usn94 ( s, c, a, b ) ;

output s ;
output c ;
input a ;
input b ;

wire a_int, b_int, nx53522z1, nx18494z1;


OBUF c_obuf (.O (c), .I (nx18494z1)) ;
OBUF s_obuf (.O (s), .I (nx53522z1)) ;
IBUF b_ibuf (.O (b_int), .I (b)) ;
IBUF a_ibuf (.O (a_int), .I (a)) ;
(* HLUTNM = "LUT62_1_1" *)
LUT2 ix53522z1320 (.O (nx53522z1), .I0 (a_int), .I1 (b_int)) ;
defparam ix53522z1320.INIT = 4'h6;
(* HLUTNM = "LUT62_1_1" *)
LUT2 ix18494z1322 (.O (nx18494z1), .I0 (a_int), .I1 (b_int)) ;
defparam ix18494z1322.INIT = 4'h8;
endmodule
```



RCA(FUNC) EDA OUTPUT



Date: 26-08-2021

Experiment No. 3

Multiplexer and Demultiplexer

Aim: To implement multiplexer and demultiplexer using verilog, verify the design using Testbench and perform functional simulation and to synthesize.

Requirements: EDA playground Tool

Theory:

=> Multiplexer: (8:1)

A multiplexer or a mux is a device that selects one of several analog or digital input signals and forwards the selected input into a single line. A multiplexer of 2n inputs has n select lines, which are used to select which input line to send to the output. The digital code applied at the select inputs determines which data inputs will be switched to output.

A_2	A_1	A_0	x
0	0	0	Y_0
0	0	1	Y_1
0	1	0	Y_2
0	1	1	Y_3

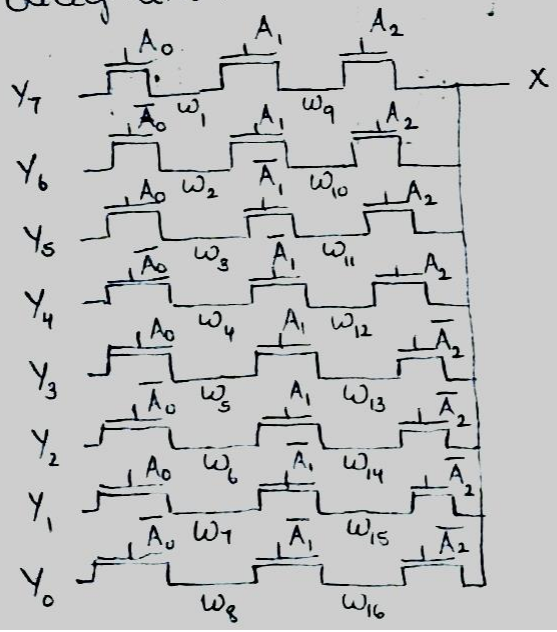
A_2	A_1	A_0	x
1	0	0	Y_4
1	0	1	Y_5
1	1	0	Y_6
1	1	1	Y_7

=> Demultiplexer : (1:8)

A demultiplexer is a device taking a single input signal and selecting one of many data output lines, which is connected to the single input. Demultiplexers take one data input and a no of selection inputs, they have several outputs. They forward the data input to one of the outputs depending on the values of selection input.

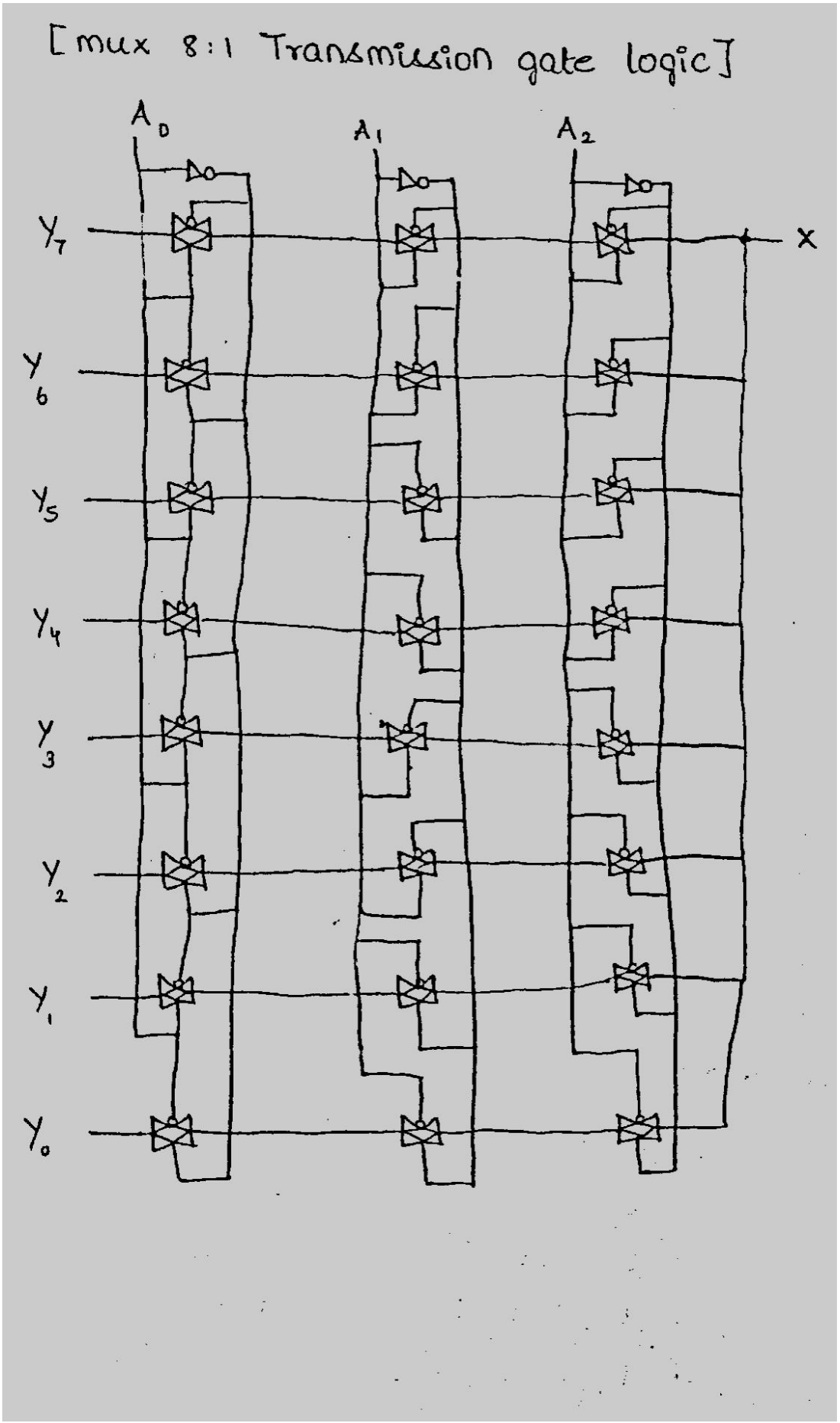
A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Circuit diagram :

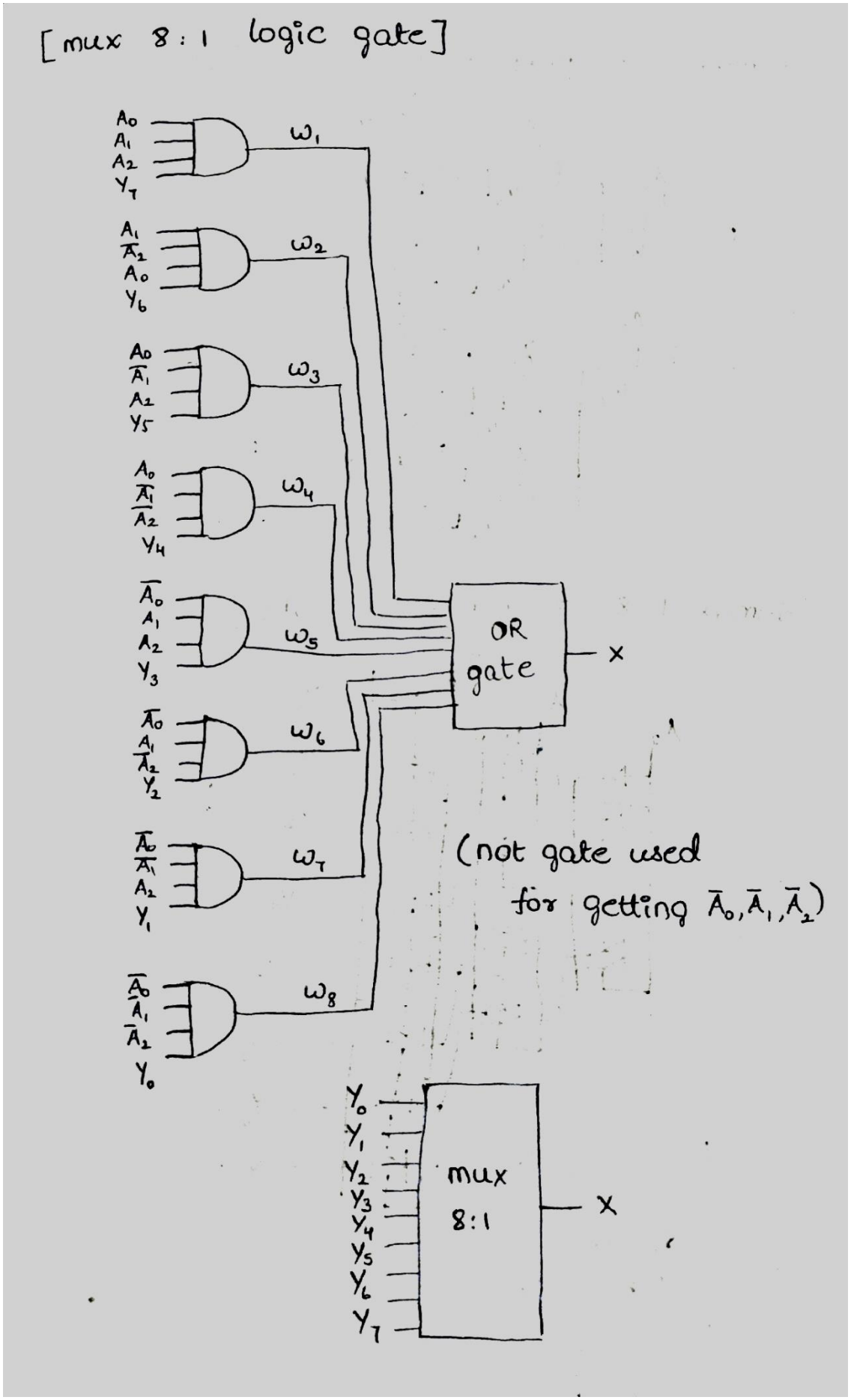


[mux 8:1 - pass transistor logic]

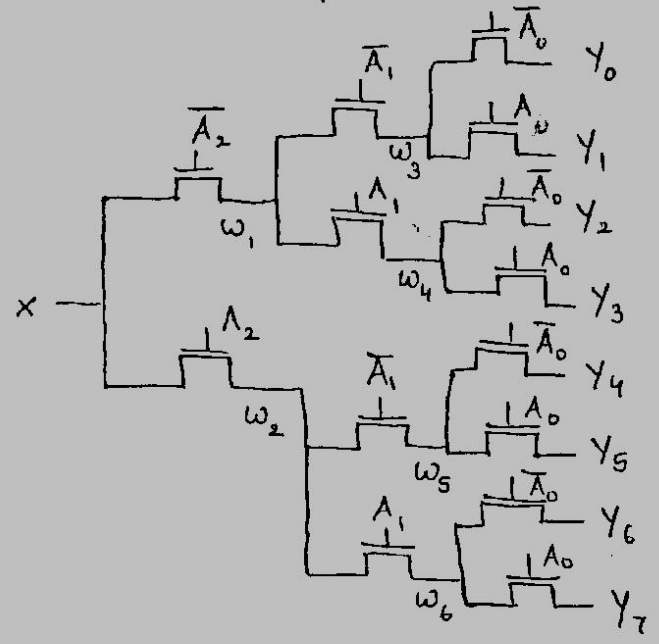
[mux 8:1 Transmission gate logic]



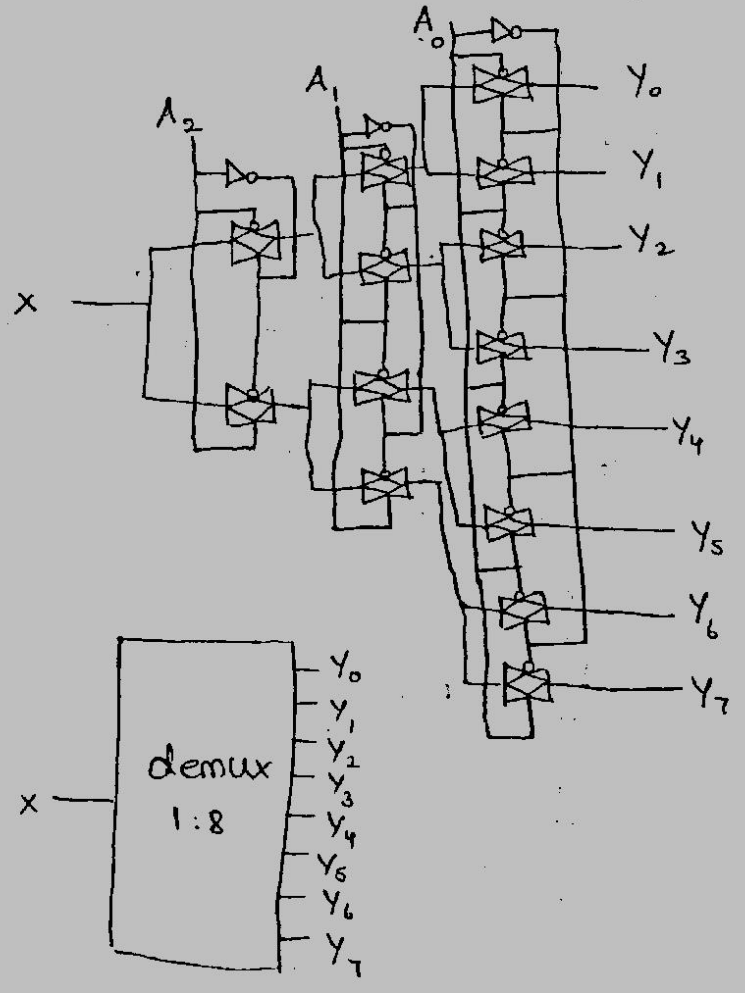
[mux 8:1 logic gate]



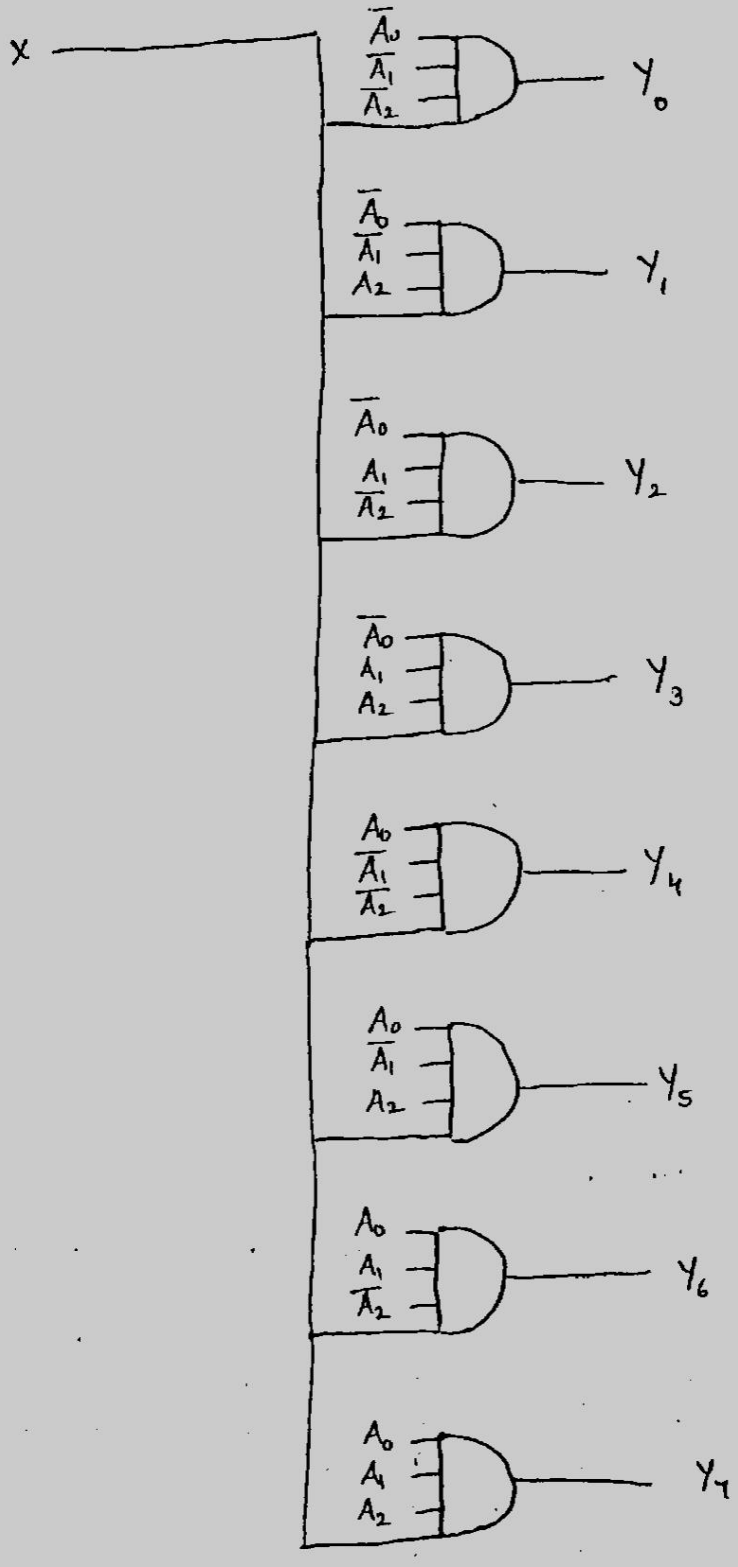
[demux 1:8 pass transistor logic]



[demux 1:8 transmission gate]



[demux 1:8 logic gate]



DMUX(BEHAVIOURAL) 1_8 EDA

DESIGN CODE	TEST BENCH
module Demux_94 (output reg [7:0] Y, input [2:0] A, input x); always @(Y, A) begin case (A) 3'b000 : begin Y[0] = x; Y[7:1] = 0; end 3'b001 : begin Y[1] = x; Y[7:2]=0;Y[0] = 0; end 3'b010 : begin Y[2] = x; Y[7:3] = 0;Y[1:0]=0; end 3'b011 : begin Y[3] = x; Y[7:4] = 0;Y[2:0]=0; end 3'b100 : begin Y[4] = x; Y[7:5] = 0;Y[3:0]=0; end 3'b101 : begin Y[5] = x; Y[7:6] = 0;Y[4:0]=0; end 3'b110 : begin Y[6] = x; Y[7] = 0;Y[5:0]=0; end 3'b111 : begin Y[7] = x; Y[6:0] = 0; end endcase end endmodule	module Demux_94test; wire [7:0] Y; reg [2:0] A; reg x; Demux_94 u1(Y, A, x); initial begin #3 x = 1;A = 3'b000; #2 A = 3'b001; #3 A = 3'b010; #2 A = 3'b011; #3 A = 3'b100; #2 A = 3'b101; #3 A = 3'b110; #2 A = 3'b111; end initial begin $monitor($time,"x = %d,A[2] = %d,A[1] = %d,A[0] = %d,Y[0] = %d,Y[1] = %d,Y[2] = %d,Y[3] = %d,Y[4] = %d,Y[5] = %d,Y[6] = %d,Y[7] = %d",x,A[2], A[1], A[0], Y[0], Y[1], Y[2], Y[3], Y[4], Y[5], Y[6], Y[7]); end initial #25 $stop; initial begin $dumpfile("dump.vcd"); $dumpvars(1, Demux_94test); end endmodule

DMUX(DF) 1_8 EDA

DESIGN CODE	TEST BENCH

module Demux_94(output reg [7:0] Y, input [2:0] A, input x);
wire A0b,A1b,A2b;
assign A0b=~A[0];
assign A1b=~A[1];
assign A2b=~A[2];
assign Y[0]=x&A2b&A1b&A0b;
assign Y[1]=x&A2b&A1b&A[0];
assign Y[2]=x&A2b&A[1]&A0b;
assign Y[3]=x&A2b&A[1]&A[0];
assign Y[4]=x&A[2]&A1b&A0b;
assign Y[5]=x&A[2]&A1b&A[0];
assign Y[6]=x&A[2]&A[1]&A0b;
assign Y[7]=x&A[2]&A[1]&A[0];
endmodule
	module Demux_94test;
	wire [7:0] Y;
	reg [2:0] A;
	reg x;
	Demux_94 u1(Y, A, x);
	initial
	begin
	#3 x = 1;A = 3'b000;
	#2 A = 3'b001;
	#3 A = 3'b010;
	#2 A = 3'b011;
	#3 A = 3'b100;
	#2 A = 3'b101;
	#3 A = 3'b110;
	#2 A = 3'b111;
	end
	initial
	begin
	$monitor($time,"x = %d,A[2] = %d,A[1] = %d,A[0] = %d,Y[0] = %d,Y[1] = %d,Y[2] = %d,Y[3] = %d,Y[4] = %d,Y[5] = %d,Y[6] = %d,Y[7] = %d",x,A[2], A[1], A[0], Y[0], Y[1], Y[2], Y[3], Y[4], Y[5], Y[6], Y[7]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, Demux_94test);
	end
	endmodule

DMUX(FUNC) 1_8 EDA

DESIGN CODE	TEST BENCH

module
Demux_94(y0,y1,y2,y3,y4,y5,y6,y7,A,x);
input [2:0] A;
input x;
output reg y0,y1,y2,y3,y4,y5,y6,y7;
always@(*)
begin
 	
{y0,y1,y2,y3,y4,y5,y6,y7}=demuxfunc_94(A,x);
end
function [7:0]demuxfunc_94;
input [2:0]A,x;
begin
assign y0=x&~A[2]&~A[1]&~A[0];
assign y1=x&~A[2]&~A[1]&A[0];
assign y2=x&~A[2]&A[1]&~A[0];
assign y3=x&~A[2]&A[1]&A[0];
assign y4=x&A[2]&~A[1]&~A[0];
assign y5=x&A[2]&~A[1]&A[0];
assign y6=x&A[2]&A[1]&~A[0];
assign y7=x&A[2]&A[1]&A[0];
end
endfunction
endmodule
	module Demux_94test;
	wire y0,y1,y2,y3,y4,y5,y6,y7;
	reg [2:0] A;
	reg x;
	Demux_94 u1(y0,y1,y2,y3,y4,y5,y6,y7, A, x);
	initial
	begin
	#3 x = 1;A = 3'b000;
	#2 A = 3'b001;
	#3 A = 3'b010;
	#2 A = 3'b011;
	#3 A = 3'b100;
	#2 A = 3'b101;
	#3 A = 3'b110;
	#2 A = 3'b111;
	end
	initial
	begin
	$monitor($time,"x = %d,A[2] = %d,A[1] = %d,A[0] = %d,y0 = %d,y1 = %d,y2 = %d,y3 = %d,y4 = %d,y5 = %d,y6 = %d,y7 = %d",x,A[2], A[1], A[0],y0,y1,y2,y3,y4,y5,y6,y7);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, Demux_94test);
	end
	endmodule

DMUX(GL) 1_8 EDA

DESIGN CODE	TEST BENCH
module Demux_94(output reg [7:0] Y, input [2:0] A, input x); wire A0b,A1b,A2b; not (A0b,A[0]), (A1b,A[1]),(A2b,A[2]); and (Y[0],x,A2b,A1b,A0b), (Y[1],x,A2b,A1b,A[0]),(Y[2],x,A2b,A[1],A0b), (Y[3],x,A2b,A[1],A[0]),(Y[4],x,A[2],A1b,A0b), (Y[5],x,A[2],A1b,A[0]),(Y[6], x,A[2],A[1],A0b),(Y[7],x,A[2],A[1],A[0]); endmodule	module Demux_94test; wire [7:0] Y; reg [2:0] A; reg x; Demux_94 u1(Y, A, x); initial begin #3 x = 1;A = 3'b000; #2 A = 3'b001; #3 A = 3'b010; #2 A = 3'b011; #3 A = 3'b100; #2 A = 3'b101; #3 A = 3'b110; #2 A = 3'b111; end initial begin $monitor($time,"x = %d,A[2] = %d,A[1] = %d,A[0] = %d,Y[0] = %d,Y[1] = %d,Y[2] = %d,Y[3] = %d,Y[4] = %d,Y[5] = %d,Y[6] = %d,Y[7] = %d",x,A[2], A[1], A[0], Y[0], Y[1], Y[2], Y[3], Y[4], Y[5], Y[6], Y[7]); end initial #25 $stop; initial begin $dumpfile("dump.vcd"); $dumpvars(1, Demux_94test); end endmodule

DMUX(PASS TRANS-NMOS) 1_8 EDA

DESIGN CODE	TEST BENCH

module Demux_94(output reg [7:0] Y, input [2:0] A, input x);
wire A0b,A1b,w1,w2,w3,w4,w5,w6;
inv g1 (A0b,A[0]);
inv g2 (A1b,A[1]);
inv g3 (A2b,A[2]);
nmos g4(w1,x,A2b);
nmos g5(w2,x,A[2]);
nmos g6(w3,w1,A1b);
nmos g7(w4,w1,A[1]);
nmos g8(w5,w2,A1b);
nmos g9(w6,w2,A[1]);
nmos g10(Y[0],w3,A0b);
nmos g11(Y[1],w3,A[0]);
nmos g12(Y[2],w4,A0b);
nmos g13(Y[3],w4,A[0]);
nmos g14(Y[4],w5,A0b);
nmos g15(Y[5],w5,A[0]);
nmos g16(Y[6],w6,A0b);
nmos g17(Y[7],w6,A[0]);
endmodule
module inv(y,x);
output y;
input x;
supply1 vdd;
supply0 gnd;
pmos(y,vdd,x);
nmos(y,gnd,x);
endmodule
	module Demux_94test;
	wire [7:0] Y;
	reg [2:0] A;
	reg x;
	Demux_94 u1(Y, A, x);
	initial
	begin
	#3 x = 1;A = 3'b000;
	#2 A = 3'b001;
	#3 A = 3'b010;
	#2 A = 3'b011;
	#3 A = 3'b100;
	#2 A = 3'b101;
	#3 A = 3'b110;
	#2 A = 3'b111;
	end
	initial
	begin
	$monitor($time,"x = %d,A[2] = %d,A[1] = %d,A[0] = %d,Y[0] = %d,Y[1] = %d,Y[2] = %d,Y[3] = %d,Y[4] = %d,Y[5] = %d,Y[6] = %d,Y[7] = %d",x,A[2], A[1], A[0], Y[0], Y[1], Y[2], Y[3], Y[4], Y[5], Y[6], Y[7]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, Demux_94test);
	end
	endmodule

DMUX(TASK) 1_8 EDA

DESIGN CODE	TEST BENCH

module Demux_94(y0,y1,y2,y3,y4,y5,y6,y7,A,x);
input [2:0] A;
input x;
output reg y0,y1,y2,y3,y4,y5,y6,y7;
always@(*)
begin
demuxtsk_94(A,x);
end
task demuxtsk_94;
input [2:0]A,x;
begin
assign y0=x&~A[2]&~A[1]&~A[0];
assign y1=x&~A[2]&~A[1]&A[0];
assign y2=x&~A[2]&A[1]&~A[0];
assign y3=x&~A[2]&A[1]&A[0];
assign y4=x&A[2]&~A[1]&~A[0];
assign y5=x&A[2]&~A[1]&A[0];
assign y6=x&A[2]&A[1]&~A[0];
assign y7=x&A[2]&A[1]&A[0];
end
endtask
endmodule
	module Demux_94test;
	wire y0,y1,y2,y3,y4,y5,y6,y7;
	reg [2:0] A;
	reg x;
	Demux_94 u1(y0,y1,y2,y3,y4,y5,y6,y7, A, x);
	initial
	begin
	#3 x = 1;A = 3'b000;
	#2 A = 3'b001;
	#3 A = 3'b010;
	#2 A = 3'b011;
	#3 A = 3'b100;
	#2 A = 3'b101;
	#3 A = 3'b110;
	#2 A = 3'b111;
	end
	initial
	begin
	$monitor($time,"x = %d,A[2] = %d,A[1] = %d,A[0] = %d,y0 = %d,y1 = %d,y2 = %d,y3 = %d,y4 = %d,y5 = %d,y6 = %d,y7 = %d",x,A[2], A[1], A[0],y0,y1,y2,y3,y4,y5,y6,y7);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, Demux_94test);
	end
	endmodule

DMUX(TG) 1_8 EDA

DESIGN CODE	TEST BENCH

module Demux_94(output reg [7:0] Y, input [2:0] A, input x);
wire w1,w2,w3,w4,w5,w6;
cmos g1(w1,x,~A[2],A[2]);
cmos g2(w2,x,A[2],~A[2]);
cmos g3(w3,w1,~A[1],A[1]);
cmos g4(w4,w1,A[1],~A[1]);
cmos g5(w5,w2,~A[1],A[1]);
cmos g6(w6,w2,A[1],~A[1]);
cmos g7(Y[0],w3,~A[0],A[0]);
cmos g8(Y[1],w3,A[0],~A[0]);
cmos g9(Y[2],w4,~A[0],A[0]);
cmos g10(Y[3],w4,A[0],~A[0]);
cmos g11(Y[4],w5,~A[0],A[0]);
cmos g12(Y[5],w5,A[0],~A[0]);
cmos g13(Y[6],w6,~A[0],A[0]);
cmos g14(Y[7],w6,A[0],~A[0]);
endmodule
	module Demux_94test;
	wire [7:0] Y;
	reg [2:0] A;
	reg x;
	Demux_94 u1(Y, A, x);
	initial
	begin
	#3 x = 1;A = 3'b000;
	#2 A = 3'b001;
	#3 A = 3'b010;
	#2 A = 3'b011;
	#3 A = 3'b100;
	#2 A = 3'b101;
	#3 A = 3'b110;
	#2 A = 3'b111;
	end
	initial
	begin
	$monitor($time,"x = %d,A[2] = %d,A[1] = %d,A[0] = %d,Y[0] = %d,Y[1] = %d,Y[2] = %d,Y[3] = %d,Y[4] = %d,Y[5] = %d,Y[6] = %d,Y[7] = %d",x,A[2], A[1], A[0], Y[0], Y[1], Y[2], Y[3], Y[4], Y[5], Y[6], Y[7]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, Demux_94test);
	end
	endmodule

MUX(BEHAVIOURAL) 8_1 EDA

DESIGN CODE	TEST BENCH

module mux_94 (input [7:0] Y, input [2:0] A,
output reg x);
always @(Y,A) begin
case (A)
3'b000 : begin x=Y[0]; end
3'b001 : begin x=Y[1]; end
3'b010 : begin x=Y[2]; end
3'b011 : begin x=Y[3]; end
3'b100 : begin x=Y[4]; end
3'b101 : begin x=Y[5]; end
3'b110 : begin x=Y[6]; end
3'b111 : begin x=Y[7]; end
endcase
end
endmodule
	module mux_94test;
	reg [7:0] Y;
	reg [2:0] A;
	wire x;
	mux_94 u1(Y, A, x);
	initial
	begin
	#0 A = 3'bxxx;
	#3 A = 3'b000;Y[0]=1;Y[7:1]=0;
	#2 A = 3'b001;Y[1]=1;Y[7:2]=0;Y[0]=0;
	#3 A = 3'b010;Y[2]=1;Y[7:3]=0;Y[1:0]=0;
	#2 A = 3'b011;Y[3]=1;Y[7:4]=0;Y[2:0]=0;
	#3 A = 3'b100;Y[4]=1;Y[7:5]=0;Y[3:0]=0;
	#2 A = 3'b101;Y[5]=1;Y[7:6]=0;Y[4:0]=0;
	#3 A = 3'b110;Y[6]=1;Y[5:0]=0;Y[7]=0;
	#2 A = 3'b111;Y[7]=1;Y[6:0]=0;
	end
	initial
	begin
	$monitor($time,"x = %d,A[2] = %d,A[1] = %d,A[0] = %d,Y[0] = %d,Y[1] = %d,Y[2] = %d,Y[3] = %d,Y[4] = %d,Y[5] = %d,Y[6] = %d,Y[7] = %d",x,A[2], A[1], A[0], Y[0], Y[1], Y[2], Y[3], Y[4], Y[5], Y[6], Y[7]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, mux_94test);
	end
	endmodule

MUX(DF) 8_1 EDA

DESIGN CODE	TEST BENCH

module mux_94(input [3:0]Y , input [1:0] A,output reg x);
assign
x=A[2]?(A[1]?(A[0]?Y[7]:Y[6]):(A[0]?Y[5]:Y[4])):
(A[1]?(A[0]?Y[3]:Y[2]):(A[0]?Y[1]:Y[0]));
endmodule
	module mux_94test;
	reg [7:0] Y;
	reg [2:0] A;
	wire x;
	mux_94 u1(Y, A, x);
	initial
	begin
	#0 A = 3'bxxx;
	#3 A = 3'b000;Y[0]=1;Y[7:1]=0;
	#2 A = 3'b001;Y[1]=1;Y[7:2]=0;Y[0]=0;
	#3 A = 3'b010;Y[2]=1;Y[7:3]=0;Y[1:0]=0;
	#2 A = 3'b011;Y[3]=1;Y[7:4]=0;Y[2:0]=0;
	#3 A = 3'b100;Y[4]=1;Y[7:5]=0;Y[3:0]=0;
	#2 A = 3'b101;Y[5]=1;Y[7:6]=0;Y[4:0]=0;
	#3 A = 3'b110;Y[6]=1;Y[5:0]=0;Y[7]=0;
	#2 A = 3'b111;Y[7]=1;Y[6:0]=0;
	end
	initial
	begin
	$monitor($time,"x = %d,A[2] = %d,A[1] = %d,
	A[0] = %d,Y[0] = %d,Y[1] = %d,Y[2] = %d,
	Y[3] = %d,Y[4] = %d,Y[5] = %d,Y[6] = %d,Y[7] =
	%d",
	x,A[2], A[1], A[0], Y[0], Y[1], Y[2], Y[3], Y[4],
	Y[5], Y[6], Y[7]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, mux_94test);
	end
	endmodule

MUX(FUNC) 8_1 EDA

DESIGN CODE	TEST BENCH

module mux_94(input [7:0]Y , input [2:0] A,output reg x);
always@(*)
begin
{x}=muxfunc_94(Y,A);
end
function muxfunc_94;
input [7:0]Y;
input [2:0] A;
begin
assign
x=A[2]?(A[1]?(A[0]?Y[7]:Y[6]):(A[0]?Y[5]:Y[4])):(A[1]?(A[0]?Y[3]:Y[2]):(A[0]?Y[1]:Y[0]));
end
endfunction
endmodule
	module mux_94test;
	reg [7:0] Y;
	reg [2:0] A;
	wire x;
	mux_94 u1(Y, A, x);
	initial
	begin
	#0 A = 3'bxxx;
	#3 A = 3'b000;Y[0]=1;Y[7:1]=0;
	#2 A = 3'b001;Y[1]=1;Y[7:2]=0;Y[0]=0;
	#3 A = 3'b010;Y[2]=1;Y[7:3]=0;Y[1:0]=0;
	#2 A = 3'b011;Y[3]=1;Y[7:4]=0;Y[2:0]=0;
	#3 A = 3'b100;Y[4]=1;Y[7:5]=0;Y[3:0]=0;
	#2 A = 3'b101;Y[5]=1;Y[7:6]=0;Y[4:0]=0;
	#3 A = 3'b110;Y[6]=1;Y[5:0]=0;Y[7]=0;
	#2 A = 3'b111;Y[7]=1;Y[6:0]=0;
	end
	initial
	begin
	$monitor($time,"x = %d,A[2] = %d,A[1] = %d,A[0] = %d,Y[0] = %d,Y[1] = %d,Y[2] = %d,Y[3] = %d,Y[4] = %d,Y[5] = %d,Y[6] = %d,Y[7] = %d",x,A[2], A[1], A[0], Y[0], Y[1], Y[2], Y[3], Y[4], Y[5], Y[6], Y[7]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, mux_94test);
	end
	endmodule

MUX(GL) 8_1 EDA

DESIGN CODE	TEST BENCH

module mux_94(input [7:0]Y , input [2:0] A,output reg x);
wire A0b, A1b, A2b, w1, w2, w3, w4, w5, w6, w7, w8;
not (A0b, A[0]), (A1b, A[1]), (A2b,A[2]);
and (w1, Y[0], A2b, A1b, A0b), (w2, Y[1], A2b, A1b, A[0]),(w3, Y[2], A[1], A0b, A2b), (w4, Y[3], A2b, A[1], A[0]), (w5, Y[4], A0b, A1b, A[2]), (w6, Y[5], A[0], A1b, A[2]), (w7, Y[6], A0b, A[1], A[2]), (w8, Y[7], A[2], A[1], A[0]);
or(x, w1, w2, w3, w4, w5, w6, w7, w8);
endmodule
	module mux_94test;
	reg [7:0] Y;
	reg [2:0] A;
	wire x;
	mux_94 u1(Y, A, x);
	initial
	begin
	#0 A = 3'bxxx;
	#3 A = 3'b000;Y[0]=1;Y[7:1]=0;
	#2 A = 3'b001;Y[1]=1;Y[7:2]=0;Y[0]=0;
	#3 A = 3'b010;Y[2]=1;Y[7:3]=0;Y[1:0]=0;
	#2 A = 3'b011;Y[3]=1;Y[7:4]=0;Y[2:0]=0;
	#3 A = 3'b100;Y[4]=1;Y[7:5]=0;Y[3:0]=0;
	#2 A = 3'b101;Y[5]=1;Y[7:6]=0;Y[4:0]=0;
	#3 A = 3'b110;Y[6]=1;Y[5:0]=0;Y[7]=0;
	#2 A = 3'b111;Y[7]=1;Y[6:0]=0;
	end
	initial
	begin
	$monitor($time,"x = %d,A[2] = %d,A[1] = %d,A[0] = %d,Y[0] = %d,Y[1] = %d,Y[2] = %d,Y[3] = %d,Y[4] = %d,Y[5] = %d,Y[6] = %d,Y[7] = %d",x,A[2], A[1], A[0], Y[0], Y[1], Y[2], Y[3], Y[4], Y[5], Y[6], Y[7]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, mux_94test);
	end
	endmodule

MUX(PASS TRANS-NMOS) 8_1 EDA

DESIGN CODE	TEST BENCH

module mux_94(Y,A,x);
input[7:0]Y;
input[2:0]A;
output x;
wire A0b,A1b;
wire [16:1]w;
assign A0b=~A[0];
assign A1b=~A[1];
assign A2b=~A[2];
nmos g1(w[1],Y[7],A[0]);
nmos g2(w[9],w[1],A[1]);
nmos g3(x,w[9],A[2]);
nmos g4(w[2],Y[6],A0b);
nmos g5(w[10],w[2],A[1]);
nmos g6(x,w[10],A[2]);
nmos g7(w[3],Y[5],A[0]);
nmos g8(w[11],w[3],A1b);
nmos g9(x,w[11],A[2]);
nmos g10(w[4],Y[4],A0b);
nmos g11(w[12],w[4],A1b);
nmos g12(x,w[12],A[2]);
nmos g13(w[5],Y[3],A[0]);
nmos g14(w[13],w[5],A[1]);
nmos g15(x,w[13],A2b);
nmos g16(w[6],Y[2],A0b);
nmos g17(w[14],w[6],A[1]);
nmos g18(x,w[14],A2b);
nmos g19(w[7],Y[1],A[0]);
nmos g20(w[15],w[7],A1b);
nmos g21(x,w[15],A2b);
nmos g22(w[8],Y[0],A0b);
nmos g23(w[16],w[8],A1b);
nmos g24(x,w[16],A2b);
endmodule
	module mux_94test;
	reg [7:0] Y;
	reg [2:0] A;
	wire x;
	mux_94 u1(Y, A, x);
	initial
	begin
	#0 A = 3'bxxx;
	#3 A = 3'b000;Y[0]=1;Y[7:1]=0;
	#2 A = 3'b001;Y[1]=1;Y[7:2]=0;Y[0]=0;
	#3 A = 3'b010;Y[2]=1;Y[7:3]=0;Y[1:0]=0;
	#2 A = 3'b011;Y[3]=1;Y[7:4]=0;Y[2:0]=0;
	#3 A = 3'b100;Y[4]=1;Y[7:5]=0;Y[3:0]=0;
	#2 A = 3'b101;Y[5]=1;Y[7:6]=0;Y[4:0]=0;
	#3 A = 3'b110;Y[6]=1;Y[5:0]=0;Y[7]=0;
	#2 A = 3'b111;Y[7]=1;Y[6:0]=0;
	end
	initial
	begin
	$monitor($time,"x = %d,A[2] = %d,A[1] = %d,A[0] = %d,Y[0] = %d,Y[1] = %d,Y[2] = %d,Y[3] = %d,Y[4] = %d,Y[5] = %d,Y[6] = %d,Y[7] = %d",x,A[2], A[1], A[0], Y[0], Y[1], Y[2], Y[3], Y[4], Y[5], Y[6], Y[7]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, mux_94test);
	end
	endmodule

MUX(TASK) 8_1 EDA

DESIGN CODE	TEST BENCH

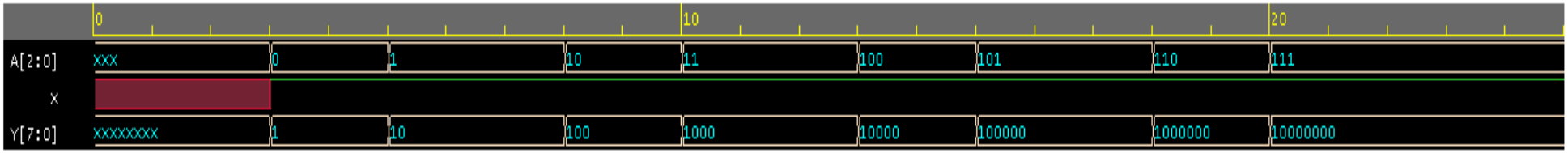
module mux_94(input [7:0]Y , input [2:0] A,output reg x);
always@(*)
begin
muxtsk_94(Y,A,x);
end
task muxtsk_94;
input [7:0]Y;
input [2:0] A;
output reg x;
begin
assign
x=A[2]?(A[1]?(A[0]?Y[7]:Y[6]):(A[0]?Y[5]:Y[4])):(A[1]?(A[0]?Y[3]:Y[2]):(A[0]?Y[1]:Y[0]));
end
endtask
endmodule
	module mux_94test;
	reg [7:0] Y;
	reg [2:0] A;
	wire x;
	mux_94 u1(Y, A, x);
	initial
	begin
	#0 A = 3'bxxx;
	#3 A = 3'b000;Y[0]=1;Y[7:1]=0;
	#2 A = 3'b001;Y[1]=1;Y[7:2]=0;Y[0]=0;
	#3 A = 3'b010;Y[2]=1;Y[7:3]=0;Y[1:0]=0;
	#2 A = 3'b011;Y[3]=1;Y[7:4]=0;Y[2:0]=0;
	#3 A = 3'b100;Y[4]=1;Y[7:5]=0;Y[3:0]=0;
	#2 A = 3'b101;Y[5]=1;Y[7:6]=0;Y[4:0]=0;
	#3 A = 3'b110;Y[6]=1;Y[5:0]=0;Y[7]=0;
	#2 A = 3'b111;Y[7]=1;Y[6:0]=0;
	end
	initial
	begin
	$monitor($time,"x = %d,A[2] = %d,A[1] = %d,A[0] = %d,Y[0] = %d,Y[1] = %d,Y[2] = %d,Y[3] = %d,Y[4] = %d,Y[5] = %d,Y[6] = %d,Y[7] = %d",x,A[2], A[1], A[0], Y[0], Y[1], Y[2], Y[3], Y[4], Y[5], Y[6], Y[7]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, mux_94test);
	end
	endmodule

MUX(TG) 8_1 EDA

DESIGN CODE	TEST BENCH

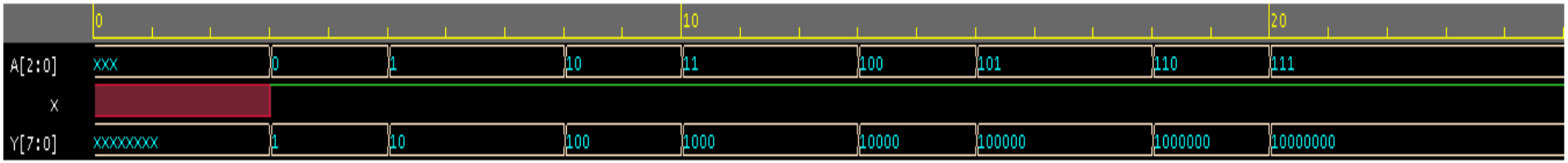
module mux_94(Y,A,x);
input[7:0]Y;
input[2:0]A;
output x;
wire A0b,A1b;
wire [16:1]w;
assign A0b=~A[0];
assign A1b=~A[1];
assign A2b=~A[2];
cmos g1(w[1],Y[0],~A[0],A[0]);
cmos g2(w[9],w[1],~A[1],A[1]);
cmos g3(x,w[9],~A[2],A[2]);
cmos g4(w[2],Y[1],A[0],~A[0]);
cmos g5(w[10],w[2],~A[1],A[1]);
cmos g6(x,w[10],~A[2],A[2]);
cmos g7(w[3],Y[2],~A[0],A[0]);
cmos g8(w[11],w[3],A[1],~A[1]);
cmos g9(x,w[11],~A[2],A[2]);
cmos g10(w[4],Y[3],A[0],~A[0]);
cmos g11(w[12],w[4],A[1],~A[1]);
cmos g12(x,w[12],~A[2],A[2]);
cmos g13(w[5],Y[4],~A[0],A[0]);
cmos g14(w[13],w[5],~A[1],A[1]);
cmos g15(x,w[13],A[2],~A[2]);
cmos g16(w[6],Y[5],A[0],~A[0]);
cmos g17(w[14],w[6],~A[1],A[1]);
cmos g18(x,w[14],A[2],~A[2]);
cmos g19(w[7],Y[6],~A[0],A[0]);
cmos g20(w[15],w[7],A[1],~A[1]);
cmos g21(x,w[15],A[2],~A[2]);
cmos g22(w[8],Y[7],A[0],~A[0]);
cmos g23(w[16],w[8],A[1],~A[1]);
cmos g24(x,w[16],A[2],~A[2]);
endmodule
	module mux_94test;
	reg [7:0] Y;
	reg [2:0] A;
	wire x;
	mux_94 u1(Y, A, x);
	initial
	begin
	#0 A = 3'bxxx;
	#3 A = 3'b000;Y[0]=1;Y[7:1]=0;
	#2 A = 3'b001;Y[1]=1;Y[7:2]=0;Y[0]=0;
	#3 A = 3'b010;Y[2]=1;Y[7:3]=0;Y[1:0]=0;
	#2 A = 3'b011;Y[3]=1;Y[7:4]=0;Y[2:0]=0;
	#3 A = 3'b100;Y[4]=1;Y[7:5]=0;Y[3:0]=0;
	#2 A = 3'b101;Y[5]=1;Y[7:6]=0;Y[4:0]=0;
	#3 A = 3'b110;Y[6]=1;Y[5:0]=0;Y[7]=0;
	#2 A = 3'b111;Y[7]=1;Y[6:0]=0;
	end
	initial
	begin
	$monitor($time,"x = %d,A[2] = %d,A[1] = %d,A[0] = %d,Y[0] = %d,Y[1] = %d,Y[2] = %d,Y[3] = %d,Y[4] = %d,Y[5] = %d,Y[6] = %d,Y[7] = %d",x,A[2], A[1], A[0], Y[0], Y[1], Y[2], Y[3], Y[4], Y[5], Y[6], Y[7]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, mux_94test);
	end
	endmodule

DMUX 1_8 EDA OUTPUT

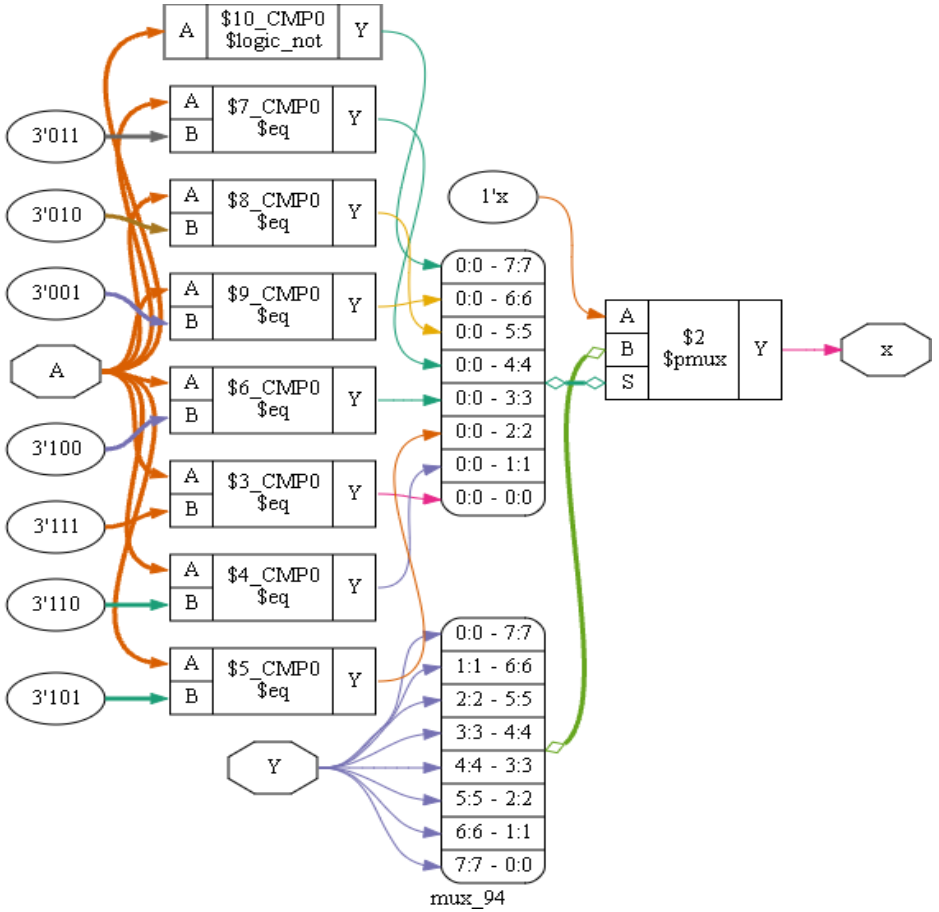



```
//
// Verilog description for cell Demux_94,
// Mon Oct 18 10:48:42 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module Demux_94 ( Y, A, x ) ;

output [7:0]Y ;
input [2:0]A ;
input x ;

wire [2:0]A_int;
wire x_int, nx8588z1, nx9585z1, nx10582z1, nx11579z1, nx12576z1, nx13573z1,
nx14570z1, nx15567z1;



OBUF \Y_obuf(0) (.O (Y[0]), .I (nx15567z1)) ;
OBUF \Y_obuf(1) (.O (Y[1]), .I (nx14570z1)) ;
OBUF \Y_obuf(2) (.O (Y[2]), .I (nx13573z1)) ;
OBUF \Y_obuf(3) (.O (Y[3]), .I (nx12576z1)) ;
OBUF \Y_obuf(4) (.O (Y[4]), .I (nx11579z1)) ;
OBUF \Y_obuf(5) (.O (Y[5]), .I (nx10582z1)) ;
OBUF \Y_obuf(6) (.O (Y[6]), .I (nx9585z1)) ;
OBUF \Y_obuf(7) (.O (Y[7]), .I (nx8588z1)) ;
IBUF x_ibuf (.O (x_int), .I (x)) ;
IBUF \A_ibuf(0) (.O (A_int[0]), .I (A[0])) ;
IBUF \A_ibuf(1) (.O (A_int[1]), .I (A[1])) ;
IBUF \A_ibuf(2) (.O (A_int[2]), .I (A[2])) ;
(* HLUTNM = "LUT62_1_4" *)
LUT4 ix8588z34082 (.O (nx8588z1), .I0 (A_int[2]), .I1 (A_int[1]), .I2 (
A_int[0]), .I3 (x_int)) ;
defparam ix8588z34082.INIT = 16'h8000;
(* HLUTNM = "LUT62_1_4" *)
LUT4 ix9585z3362 (.O (nx9585z1), .I0 (A_int[2]), .I1 (A_int[1]), .I2 (
A_int[0]), .I3 (x_int)) ;
defparam ix9585z3362.INIT = 16'h0800;
(* HLUTNM = "LUT62_1_1" *)
LUT4 ix10582z9506 (.O (nx10582z1), .I0 (A_int[2]), .I1 (A_int[1]), .I2 (
A_int[0]), .I3 (x_int)) ;
defparam ix10582z9506.INIT = 16'h2000;
(* HLUTNM = "LUT62_1_1" *)
LUT4 ix11579z1826 (.O (nx11579z1), .I0 (A_int[2]), .I1 (A_int[1]), .I2 (
A_int[0]), .I3 (x_int)) ;
defparam ix11579z1826.INIT = 16'h0200;
(* HLUTNM = "LUT62_1_2" *)
LUT4 ix12576z17698 (.O (nx12576z1), .I0 (A_int[2]), .I1 (A_int[1]), .I2 (
A_int[0]), .I3 (x_int)) ;
defparam ix12576z17698.INIT = 16'h4000;
(* HLUTNM = "LUT62_1_2" *)
LUT4 ix13573z2338 (.O (nx13573z1), .I0 (A_int[2]), .I1 (A_int[1]), .I2 (
A_int[0]), .I3 (x_int)) ;
defparam ix13573z2338.INIT = 16'h0400;
(* HLUTNM = "LUT62_1_3" *)
LUT4 ix14570z5410 (.O (nx14570z1), .I0 (A_int[2]), .I1 (A_int[1]), .I2 (
A_int[0]), .I3 (x_int)) ;
defparam ix14570z5410.INIT = 16'h1000;
(* HLUTNM = "LUT62_1_3" *)
LUT4 ix15567z1570 (.O (nx15567z1), .I0 (A_int[2]), .I1 (A_int[1]), .I2 (
A_int[0]), .I3 (x_int)) ;
defparam ix15567z1570.INIT = 16'h0100;
endmodule
```

MUX 8_1 EDA OUTPUT




```
//
// Verilog description for cell mux_94,
// Mon Oct 18 11:00:11 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module mux_94 ( Y, A, x ) ;

input [7:0]Y ;
input [2:0]A ;
output x ;

wire x_1_0;
wire [7:0]Y_int;
wire [2:0]A_int;
wire nx14587z1, nx14587z2;



MUXF7 x_mux_0_ix14587z55180 (.O (x_1_0), .I0 (nx14587z1), .I1 (nx14587z2), .S (
A_int[2])) ;
OBUF x_obuf (.O (x), .I (x_1_0)) ;
IBUF \A_ibuf(0) (.O (A_int[0]), .I (A[0])) ;
IBUF \A_ibuf(1) (.O (A_int[1]), .I (A[1])) ;
IBUF \A_ibuf(2) (.O (A_int[2]), .I (A[2])) ;
IBUF \Y_ibuf(0) (.O (Y_int[0]), .I (Y[0])) ;
IBUF \Y_ibuf(1) (.O (Y_int[1]), .I (Y[1])) ;
IBUF \Y_ibuf(2) (.O (Y_int[2]), .I (Y[2])) ;
IBUF \Y_ibuf(3) (.O (Y_int[3]), .I (Y[3])) ;
IBUF \Y_ibuf(4) (.O (Y_int[4]), .I (Y[4])) ;
IBUF \Y_ibuf(5) (.O (Y_int[5]), .I (Y[5])) ;
IBUF \Y_ibuf(6) (.O (Y_int[6]), .I (Y[6])) ;
IBUF \Y_ibuf(7) (.O (Y_int[7]), .I (Y[7])) ;
LUT6 ix14587z20323 (.O (nx14587z1), .I0 (A_int[0]), .I1 (Y_int[2]), .I2 (
A_int[1]), .I3 (Y_int[1]), .I4 (Y_int[0]), .I5 (Y_int[3])) ;
defparam ix14587z20323.INIT = 64'hEFE5EAE04F454A40;
LUT6 ix14587z20324 (.O (nx14587z2), .I0 (A_int[0]), .I1 (Y_int[6]), .I2 (
A_int[1]), .I3 (Y_int[5]), .I4 (Y_int[4]), .I5 (Y_int[7])) ;
defparam ix14587z20324.INIT = 64'hEFE5EAE04F454A40;
endmodule
```



Date: 02-09-2021

Experiment No. 4

Encoder and decoder

Aim: To implement encoder and decoder using verilog, verify the design using Testbench and perform functional simulation and to synthesize

Requirements: EDA playground Tool

Theory:

=> Encoder: (8:3)

An encoder is a combinational circuit that performs the reverse operation of decoder. It has maximum of 2^n input lines and 'n' output lines. It produces a binary code equivalent to the input which is active high.

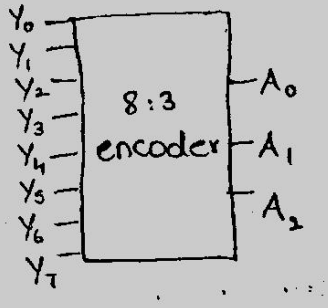
Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Input | Output

$$A_2 = Y_7 + Y_6 + Y_5 + Y_4$$

$$A_1 = Y_7 + Y_6 + Y_3 + Y_2$$

$$A_0 = Y_7 + Y_5 + Y_3 + Y_1$$

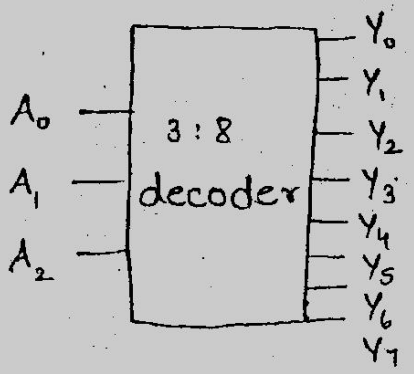


=> Decoder (3:8)

Decoder is a combinational circuit that has 'n' input lines and maximum of 2^n output lines. One of these outputs will be active high based on combination of inputs present, when the decoder is min terms of 'n' input variables, when it is enabled

A_2	A_1	A_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

input output



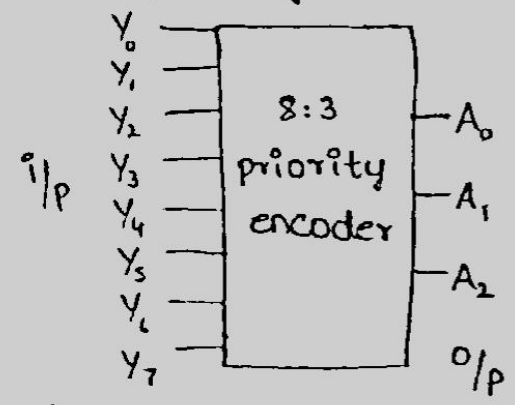
⇒ Priority Encoder:

It has the output which correspond to the currently active input which has the highest priority. So when an input with a higher priority is present, all other input with a lower priority will be ignored.

Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

output — input

lowest priority



highest priority

DECODER(BEHAVIOURAL CASE) 3_8 EDA

DESIGN CODE	TEST BENCH

module decode_94 (output reg [7:0] Y, input [2:0] A);
always @(Y,A) begin
case (A)
3'b000 : begin
Y[0]=(~A[2]&~A[1]&~A[0]);Y[7:1]=0;
end
3'b001 : begin
Y[1]=(~A[2]&~A[1]&A[0]);Y[7:2]=0;Y[0]=0;
end
3'b010 : begin
Y[2]=(~A[2]&A[1]&~A[0]);Y[7:3]=0;Y[1:0]=0;
end
3'b011 : begin
Y[3]=(~A[2]&A[1]&A[0]);Y[7:4]=0;Y[2:0]=0;
end
3'b100 : begin
Y[4]=(A[2]&~A[1]&~A[0]);Y[7:5]=0;Y[3:0]=0;
end
3'b101 : begin
Y[5]=(A[2]&~A[1]&A[0]);Y[7:6]=0;Y[4:0]=0;
end
3'b110 : begin
Y[6]=(A[2]&A[1]&~A[0]);Y[7]=0;Y[5:0]=0;
end
3'b111 : begin
Y[7]=(A[2]&A[1]&A[0]);Y[6:0]=0; end
endcase
end
endmodule
	module decode_94test;
	wire [7:0] Y;
	reg [2:0] A;
	
	decode_94 g1(Y,A);
	initial
	begin
	#0 A = 3'b000;
	#2 A = 3'b001;
	#3 A = 3'b010;
	#2 A = 3'b011;
	#3 A = 3'b100;
	#2 A = 3'b101;
	#3 A = 3'b110;
	#2 A = 3'b111;
	end
	initial
	begin
	$monitor($time,"A[2] = %d,A[1] = %d,A[0]=%d,Y[7] = %d,Y[6] = %d,Y[5] = %d,Y[4] = %d, Y[3] = %d,Y[2] = %d,Y[1] = %d,Y[0] = %d", A[2], A[1], A[0], Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1],Y[0]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, decode_94test);
	end
	endmodule

DECODER(BEHAVIOURAL IF) 3_8 EDA

DESIGN CODE	TEST BENCH

module decode_94 (output reg [7:0] Y,input [2:0] A);
always @(Y,A) begin
if(A==3'b000)
begin
Y[0]=(~A[2]&~A[1]&~A[0]);Y[7:1]=0; end
else if(A==3'b001)
begin
Y[1]=(~A[2]&~A[1]&A[0]);Y[7:2]=0;Y[0]=0;
end
else if(A==3'b010)
begin
Y[2]=(~A[2]&A[1]&~A[0]);Y[7:3]=0;Y[1:0]=0;
end
else if(A==3'b011)
begin
Y[3]=(~A[2]&A[1]&A[0]);Y[7:4]=0;Y[2:0]=0;
end
else if(A==3'b100)
begin
Y[4]=(A[2]&~A[1]&~A[0]);Y[7:5]=0;Y[3:0]=0;
end
else if(A==3'b101)
begin
Y[5]=(A[2]&~A[1]&A[0]);Y[7:6]=0;Y[4:0]=0;
end
else if(A==3'b110)
begin
Y[6]=(A[2]&A[1]&~A[0]);Y[7]=0;Y[5:0]=0;
end
else if(A==3'b111)
begin Y[7]=(A[2]&A[1]&A[0]);Y[6:0]=0;
end
else
begin Y=3'bxxx; end
end
endmodule
	module decode_94test;
	wire [7:0] Y;
	reg [2:0] A;
	
	decode_94 g1(Y,A);
	initial
	begin
	#0 A = 3'b000;
	#2 A = 3'b001;
	#3 A = 3'b010;
	#2 A = 3'b011;
	#3 A = 3'b100;
	#2 A = 3'b101;
	#3 A = 3'b110;
	#2 A = 3'b111;
	end
	initial
	begin
	$monitor($time,"A[2] = %d,A[1] = %d,A[0]=%d,Y[7] = %d,Y[6] = %d,Y[5] = %d,Y[4] = %d, Y[3] = %d,Y[2] = %d,Y[1] = %d,Y[0] = %d", A[2], A[1], A[0], Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1],Y[0]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, decode_94test);
	end
	endmodule

DECODER(DF) 3_8 EDA

DESIGN CODE	TEST BENCH

module decode_94 (Y,A);
  output [7:0] Y;
  input [2:0] A;
  assign Y[0]=(~A[2]&~A[1]&~A[0]);
  assign Y[1]=(~A[2]&~A[1]&A[0]);
  assign Y[2]=(~A[2]&A[1]&~A[0]);
  assign Y[3]=(~A[2]&A[1]&A[0]);
  assign Y[4]=(A[2]&~A[1]&~A[0]);
  assign Y[5]=(A[2]&~A[1]&A[0]);
  assign Y[6]=(A[2]&A[1]&~A[0]);
  assign Y[7]=(A[2]&A[1]&A[0]);
endmodule
	module decode_94test;
	  wire [7:0] Y;
	  reg [2:0] A;
	
	  decode_94 g1(Y,A);
	initial
	  begin
	#0 A = 3'b000;
	#2 A = 3'b001;
	#3 A = 3'b010;
	#2 A = 3'b011;
	#3 A = 3'b100;
	#2 A = 3'b101;
	#3 A = 3'b110;
	#2 A = 3'b111;
	  end
	initial
	  begin
	  $monitor($time,"A[2] = %d,A[1] = %d,A[0]=%d,Y[7] = %d,Y[6] = %d,Y[5] = %d,Y[4] = %d, Y[3] = %d,Y[2] = %d,Y[1] = %d,Y[0] = %d", A[2], A[1], A[0], Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1],Y[0]);
	  end
	  initial
	  #25 $stop;
	 initial
	  begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1, decode_94test);
	  end
	endmodule

DECODER(FUNC) 3_8 EDA

DESIGN CODE	TEST BENCH

module decode_94 (y0,y1,y2,y3,y4,y5,y6,y7,A);
output reg y0,y1,y2,y3,y4,y5,y6,y7;
input [2:0] A;
always @ *
begin
{y0,y1,y2,y3,y4,y5,y6,y7}=decoderfunc_94(A);
end
function[7:0] decoderfunc_94;
input [2:0] A;
begin
assign y0=(~A[2]&~A[1]&~A[0]);
assign y1=(~A[2]&~A[1]&A[0]);
assign y2=(~A[2]&A[1]&~A[0]);
assign y3=(~A[2]&A[1]&A[0]);
assign y4=(A[2]&~A[1]&~A[0]);
assign y5=(A[2]&~A[1]&A[0]);
assign y6=(A[2]&A[1]&~A[0]);
assign y7=(A[2]&A[1]&A[0]);
end
endfunction
endmodule
	module decode_94test;
	wire y0,y1,y2,y3,y4,y5,y6,y7;
	reg [2:0] A;
	
	decode_94 g1(y0,y1,y2,y3,y4,y5,y6,y7,A);
	initial
	begin
	#0 A = 3'b000;
	#2 A = 3'b001;
	#3 A = 3'b010;
	#2 A = 3'b011;
	#3 A = 3'b100;
	#2 A = 3'b101;
	#3 A = 3'b110;
	#2 A = 3'b111;
	end
	initial
	begin
	$monitor($time,"A[2] = %d,A[1] = %d,A[0]=%d,y7 = %d,y6 = %d,y5 = %d,y4 = %d, y3 = %d,y2 = %d,y1 = %d,y0 = %d", A[2], A[1], A[0], y7, y6, y5, y4, y3, y2, y1, y0);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, decode_94test);
	end
	endmodule

DECODER(GL) 3_8 EDA

DESIGN CODE	TEST BENCH

module decode_94 (Y,A);
  output [7:0] Y;
  input [2:0] A;
  and (Y[0],~A[2],~A[1],~A[0]);
  and (Y[1],~A[2],~A[1],A[0]);
  and (Y[2],~A[2],A[1],~A[0]);
  and (Y[3],~A[2],A[1],A[0]);
  and (Y[4],A[2],~A[1],~A[0]);
  and (Y[5],A[2],~A[1],A[0]);
  and (Y[6],A[2],A[1],~A[0]);
  and (Y[7],A[2],A[1],A[0]);
endmodule
	module decode_94test;
	  wire [7:0] Y;
	  reg [2:0] A;
	
	  decode_94 g1(Y,A);
	initial
	  begin
	#0 A = 3'b000;
	#2 A = 3'b001;
	#3 A = 3'b010;
	#2 A = 3'b011;
	#3 A = 3'b100;
	#2 A = 3'b101;
	#3 A = 3'b110;
	#2 A = 3'b111;
	  end
	initial
	  begin
	  $monitor($time,"A[2] = %d,A[1] = %d,A[0]=%d,Y[7] = %d,Y[6] = %d,Y[5] = %d,Y[4] = %d, Y[3] = %d,Y[2] = %d,Y[1] = %d,Y[0] = %d", A[2], A[1], A[0], Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1],Y[0]);
	  end
	  initial
	  #25 $stop;
	 initial
	  begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1, decode_94test);
	  end
	endmodule

DECODER(TASK) 3_8 EDA

DESIGN CODE	TEST BENCH

module decode_94 (Y,A);
output reg[7:0] Y;
input [2:0] A;
always @ (*)
begin
decodertsk_94(A,Y);
end
task decodertsk_94;
input [2:0] A;
output reg[7:0] Y;
begin
Y[0]=(~A[2]&~A[1]&~A[0]);
Y[1]=(~A[2]&~A[1]&A[0]);
Y[2]=(~A[2]&A[1]&~A[0]);
Y[3]=(~A[2]&A[1]&A[0]);
Y[4]=(A[2]&~A[1]&~A[0]);
Y[5]=(A[2]&~A[1]&A[0]);
Y[6]=(A[2]&A[1]&~A[0]);
Y[7]=(A[2]&A[1]&A[0]);
end
endtask
endmodule
	module decode_94test;
	wire [7:0] Y;
	reg [2:0] A;
	
	decode_94 g1(Y,A);
	initial
	begin
	#0 A = 3'b000;
	#2 A = 3'b001;
	#3 A = 3'b010;
	#2 A = 3'b011;
	#3 A = 3'b100;
	#2 A = 3'b101;
	#3 A = 3'b110;
	#2 A = 3'b111;
	end
	initial
	begin
	$monitor($time,"A[2] = %d,A[1] = %d,A[0]=%d,Y[7] = %d,Y[6] = %d,Y[5] = %d,Y[4] = %d, Y[3] = %d,Y[2] = %d,Y[1] = %d,Y[0] = %d", A[2], A[1], A[0], Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1],Y[0]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, decode_94test);
	end
	endmodule

ENCODER(BEHAVIOURAL CASE) 8_3 EDA

DESIGN CODE	TEST BENCH

module encode_94 (output reg [2:0] A,input [7:0] Y);
always @(A,Y) begin
case (Y)
8'b10000000 : begin A[2] = ((Y[7]|Y[6]|Y[5]|Y[4])); A[1] = ((Y[7]|Y[6]|Y[3]|Y[2])); A[0] =((Y[7]|Y[5]|Y[3]|Y[1])); end
8'b01000000 : begin A[2] = ((Y[7]|Y[6]|Y[5]|Y[4])); A[1] = ((Y[7]|Y[6]|Y[3]|Y[2])); A[0] =((Y[7]|Y[5]|Y[3]|Y[1])); end
8'b00100000 : begin A[2] = ((Y[7]|Y[6]|Y[5]|Y[4])); A[1] = ((Y[7]|Y[6]|Y[3]|Y[2])); A[0] =((Y[7]|Y[5]|Y[3]|Y[1])); end
8'b00010000 : begin A[2] = ((Y[7]|Y[6]|Y[5]|Y[4])); A[1] = ((Y[7]|Y[6]|Y[3]|Y[2])); A[0] =((Y[7]|Y[5]|Y[3]|Y[1])); end
8'b00001000 : begin A[2] = ((Y[7]|Y[6]|Y[5]|Y[4])); A[1] = ((Y[7]|Y[6]|Y[3]|Y[2])); A[0] =((Y[7]|Y[5]|Y[3]|Y[1])); end
8'b00000100 : begin A[2] = ((Y[7]|Y[6]|Y[5]|Y[4])); A[1] = ((Y[7]|Y[6]|Y[3]|Y[2])); A[0] =((Y[7]|Y[5]|Y[3]|Y[1])); end
8'b00000010 : begin A[2] = ((Y[7]|Y[6]|Y[5]|Y[4])); A[1] = ((Y[7]|Y[6]|Y[3]|Y[2])); A[0] =((Y[7]|Y[5]|Y[3]|Y[1])); end
8'b00000001 : begin A[2] = ((Y[7]|Y[6]|Y[5]|Y[4])); A[1] = ((Y[7]|Y[6]|Y[3]|Y[2])); A[0] =((Y[7]|Y[5]|Y[3]|Y[1])); end
endcase
end
endmodule
	module encode_94test;
	wire [2:0] A;
	reg [7:0] Y;
	
	encode_94 g1(A,Y);
	initial
	begin
	#0 Y = 8'b00000001;
	#2 Y = 8'b00000010;
	#3 Y = 8'b00000100;
	#2 Y = 8'b00001000;
	#3 Y = 8'b00010000;
	#2 Y = 8'b00100000;
	#3 Y = 8'b01000000;
	#2 Y = 8'b10000000;
	end
	initial
	begin
	$monitor($time,"Y[7] = %d,Y[6] = %d,Y[5] = %d,Y[4] = %d, Y[3] = %d,Y[2] = %d,Y[1] = %d,Y[0] = %d,A[0] = %d,A[1] = %d,A[2]=%d", Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1], Y[0], A[0], A[1], A[2]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, encode_94test);
	end
	endmodule

ENCODER(BEHAVIOURAL IF) 8_3 EDA

DESIGN CODE	TEST BENCH

module encode_94 (output reg [2:0] A,input [7:0] Y);
always @(A,Y) begin
if(Y==8'b10000000)
begin A[2] = ((Y[7]|Y[6]|Y[5]|Y[4])); A[1] = ((Y[7]|Y[6]|Y[3]|Y[2])); A[0] =((Y[7]|Y[5]|Y[3]|Y[1])); end
else if(Y==8'b01000000)
begin A[2] = ((Y[7]|Y[6]|Y[5]|Y[4])); A[1] = ((Y[7]|Y[6]|Y[3]|Y[2])); A[0] =((Y[7]|Y[5]|Y[3]|Y[1])); end
else if(Y==8'b00100000)
begin A[2] = ((Y[7]|Y[6]|Y[5]|Y[4])); A[1] = ((Y[7]|Y[6]|Y[3]|Y[2])); A[0] =((Y[7]|Y[5]|Y[3]|Y[1])); end
else if(Y==8'b00010000)
begin A[2] = ((Y[7]|Y[6]|Y[5]|Y[4])); A[1] = ((Y[7]|Y[6]|Y[3]|Y[2])); A[0] =((Y[7]|Y[5]|Y[3]|Y[1])); end
else if(Y==8'b00001000)
begin A[2] = ((Y[7]|Y[6]|Y[5]|Y[4])); A[1] = ((Y[7]|Y[6]|Y[3]|Y[2])); A[0] =((Y[7]|Y[5]|Y[3]|Y[1])); end
else if(Y==8'b00000100)
begin A[2] = ((Y[7]|Y[6]|Y[5]|Y[4])); A[1] = ((Y[7]|Y[6]|Y[3]|Y[2])); A[0] =((Y[7]|Y[5]|Y[3]|Y[1])); end
else if(Y==8'b00000010)
begin A[2] = ((Y[7]|Y[6]|Y[5]|Y[4])); A[1] = ((Y[7]|Y[6]|Y[3]|Y[2])); A[0] =((Y[7]|Y[5]|Y[3]|Y[1])); end
else if(Y==8'b00000001)
begin A[2] = ((Y[7]|Y[6]|Y[5]|Y[4])); A[1] = ((Y[7]|Y[6]|Y[3]|Y[2])); A[0] =((Y[7]|Y[5]|Y[3]|Y[1])); end
else
begin A=3'bxxx; end
end
endmodule
	module encode_94test;
	wire [2:0] A;
	reg [7:0] Y;
	
	encode_94 g1(A,Y);
	initial
	begin
	#0 Y = 8'b00000001;
	#2 Y = 8'b00000010;
	#3 Y = 8'b00000100;
	#2 Y = 8'b00001000;
	#3 Y = 8'b00010000;
	#2 Y = 8'b00100000;
	#3 Y = 8'b01000000;
	#2 Y = 8'b10000000;
	end
	initial
	begin
	$monitor($time,"Y[7] = %d,Y[6] = %d,Y[5] = %d,Y[4] = %d, Y[3] = %d,Y[2] = %d,Y[1] = %d,Y[0] = %d,A[0] = %d,A[1] = %d,A[2]=%d", Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1], Y[0], A[0], A[1], A[2]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, encode_94test);
	end
	endmodule

ENCODER(DF) 8_3 EDA

DESIGN CODE	TEST BENCH

module encode_94 (A,Y);
  output [2:0] A;
  input [7:0] Y;
  assign A[2] = ((Y[7]|Y[6]|Y[5]|Y[4]));
  assign A[1] = ((Y[7]|Y[6]|Y[3]|Y[2]));
  assign A[0] =((Y[7]|Y[5]|Y[3]|Y[1]));
endmodule
	module encode_94test;
	  wire [2:0] A;
	  reg [7:0] Y;
	
	  encode_94 g1(A,Y);
	initial
	  begin
	#0 Y = 8'b00000001;
	#2 Y = 8'b00000010;
	#3 Y = 8'b00000100;
	#2 Y = 8'b00001000;
	#3 Y = 8'b00010000;
	#2 Y = 8'b00100000;
	#3 Y = 8'b01000000;
	#2 Y = 8'b10000000;
	  end
	initial
	  begin
	  $monitor($time,"Y[7] = %d,Y[6] = %d,Y[5] = %d,Y[4] = %d, Y[3] = %d,Y[2] = %d,Y[1] = %d,Y[0] = %d,A[0] = %d,A[1] = %d,A[2]=%d", Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1], Y[0], A[0], A[1], A[2]);
	  end
	  initial
	  #25 $stop;
	 initial
	  begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1, encode_94test);
	  end
	endmodule

ENCODER(FUNC) 8_3 EDA

DESIGN CODE	TEST BENCH

module encode_94 (a0,a1,a2,Y);
output reg a0,a1,a2;
input [7:0] Y;
always @ *
begin
{a0,a1,a2}=encoderfunc_94(Y);
end
function[2:0] encoderfunc_94;
input [7:0] Y;
begin
assign a2 = Y[7]|Y[6]|Y[5]|Y[4];
assign a1 = Y[7]|Y[6]|Y[3]|Y[2];
assign a0 = Y[7]|Y[5]|Y[3]|Y[1];
end
endfunction
endmodule
	module encode_94test;
	wire a0,a1,a2;
	reg [7:0] Y;
	
	encode_94 g1(a0,a1,a2,Y);
	initial
	begin
	#0 Y = 8'b00000001;
	#2 Y = 8'b00000010;
	#3 Y = 8'b00000100;
	#2 Y = 8'b00001000;
	#3 Y = 8'b00010000;
	#2 Y = 8'b00100000;
	#3 Y = 8'b01000000;
	#2 Y = 8'b10000000;
	end
	initial
	begin
	$monitor($time,"Y[7] = %d,Y[6] = %d,Y[5] = %d,Y[4] = %d, Y[3] = %d,Y[2] = %d,Y[1] = %d,Y[0] = %d,a0 = %d,a1 = %d,a2=%d", Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1], Y[0], a0, a1, a2);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, encode_94test);
	end
	endmodule

ENCODER(GL) 8_3 EDA

DESIGN CODE	TEST BENCH

module encode_94 (A,Y);
  output [2:0] A;
  input [7:0] Y;
  or g1 (A[2],Y[7],Y[6],Y[5],Y[4]);
  or g2 (A[1],Y[7],Y[6],Y[3],Y[2]);
  or g3 (A[0],Y[7],Y[5],Y[3],Y[1]);
endmodule
	module encode_94test;
	  wire [2:0] A;
	  reg [7:0] Y;
	
	  encode_94 g1(A,Y);
	initial
	  begin
	#0 Y = 8'b00000001;
	#2 Y = 8'b00000010;
	#3 Y = 8'b00000100;
	#2 Y = 8'b00001000;
	#3 Y = 8'b00010000;
	#2 Y = 8'b00100000;
	#3 Y = 8'b01000000;
	#2 Y = 8'b10000000;
	  end
	initial
	  begin
	  $monitor($time,"Y[7] = %d,Y[6] = %d,Y[5] = %d,Y[4] = %d, Y[3] = %d,Y[2] = %d,Y[1] = %d,Y[0] = %d,A[0] = %d,A[1] = %d,A[2]=%d", Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1], Y[0], A[0], A[1], A[2]);
	  end
	  initial
	  #25 $stop;
	 initial
	  begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1, encode_94test);
	  end
	endmodule

ENCODER(TASK) 8_3 EDA

DESIGN CODE	TEST BENCH

module encode_94 (A,Y);
output reg[2:0] A;
input [7:0] Y;
always @ (*)
begin
encodertsk_94(Y,A);
end
task encodertsk_94;
input [7:0] Y;
output reg[2:0] A;
begin
A[2] = ((Y[7]|Y[6]|Y[5]|Y[4]));
A[1] = ((Y[7]|Y[6]|Y[3]|Y[2]));
A[0] =((Y[7]|Y[5]|Y[3]|Y[1]));
end
endtask
endmodule
	module encode_94test;
	wire [2:0] A;
	reg [7:0] Y;
	
	encode_94 g1(A,Y);
	initial
	begin
	#0 Y = 8'b00000001;
	#2 Y = 8'b00000010;
	#3 Y = 8'b00000100;
	#2 Y = 8'b00001000;
	#3 Y = 8'b00010000;
	#2 Y = 8'b00100000;
	#3 Y = 8'b01000000;
	#2 Y = 8'b10000000;
	end
	initial
	begin
	$monitor($time,"Y[7] = %d,Y[6] = %d,Y[5] = %d,Y[4] = %d, Y[3] = %d,Y[2] = %d,Y[1] = %d,Y[0] = %d,A[0] = %d,A[1] = %d,A[2]=%d", Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1], Y[0], A[0], A[1], A[2]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, encode_94test);
	end
	endmodule

PR-ENCODER(BEHAVIOURAL CASE) 8_3 EDA

DESIGN CODE	TEST BENCH

module prencoder_94(Y,A);
input [7:0] Y;
output [2:0] A;
reg [7:0] Y;
reg [2:0] A;
always @ (Y)
begin
case(1'b1)
Y[7] : A=3'b111;
Y[6] : A=3'b110;
Y[5] : A=3'b101;
Y[4] : A=3'b100;
Y[3] : A=3'b011;
Y[2] : A=3'b010;
Y[1] : A=3'b001;
Y[0] : A=3'b000;
endcase
end
endmodule
	module prencode_94test;
	wire [2:0] A;
	reg [7:0] Y;
	
	prencoder_94 g1(Y,A);
	initial
	begin
	#0 Y = 8'b00000001;
	#2 Y = 8'b0000001x;
	#3 Y = 8'b000001xx;
	#2 Y = 8'b00001xxx;
	#3 Y = 8'b0001xxxx;
	#2 Y = 8'b001xxxxx;
	#3 Y = 8'b01xxxxxx;
	#2 Y = 8'b1xxxxxxx;
	end
	initial
	begin
	$monitor($time,"Y[7] = %d,Y[6] = %d,Y[5] = %d,Y[4] = %d, Y[3] = %d,Y[2] = %d,Y[1] = %d,Y[0] = %d,A[0] = %d,A[1] = %d,A[2]=%d", Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1], Y[0], A[0], A[1], A[2]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, prencode_94test);
	end
	endmodule

PR-ENCODER(BEHAVIOURAL IF) 8_3 EDA

DESIGN CODE	TEST BENCH

module prencoder_94(Y,A);
input [7:0] Y;
output [2:0] A;
reg [7:0] Y;
reg [2:0] A;
always @ (Y)
begin
if(Y[7]==1) A=3'b111;
else if(Y[6]==1) A=3'b110;
else if(Y[5]==1) A=3'b101;
else if(Y[4]==1) A=3'b100;
else if(Y[3]==1) A=3'b011;
else if(Y[2]==1) A=3'b010;
else if(Y[1]==1) A=3'b001;
else if(Y[0]==1) A=3'b000;
else A=3'bxxx;
end
endmodule
	module prencode_94test;
	wire [2:0] A;
	reg [7:0] Y;
	
	prencoder_94 g1(Y,A);
	initial
	begin
	#0 Y = 8'b00000001;
	#2 Y = 8'b0000001x;
	#3 Y = 8'b000001xx;
	#2 Y = 8'b00001xxx;
	#3 Y = 8'b0001xxxx;
	#2 Y = 8'b001xxxxx;
	#3 Y = 8'b01xxxxxx;
	#2 Y = 8'b1xxxxxxx;
	end
	initial
	begin
	$monitor($time,"Y[7] = %d,Y[6] = %d,Y[5] = %d,Y[4] = %d, Y[3] = %d,Y[2] = %d,Y[1] = %d,Y[0] = %d,A[0] = %d,A[1] = %d,A[2]=%d", Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1], Y[0], A[0], A[1], A[2]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, prencode_94test);
	end
	endmodule

PR-ENCODER(FUNC) 8_3 EDA

DESIGN CODE	TEST BENCH

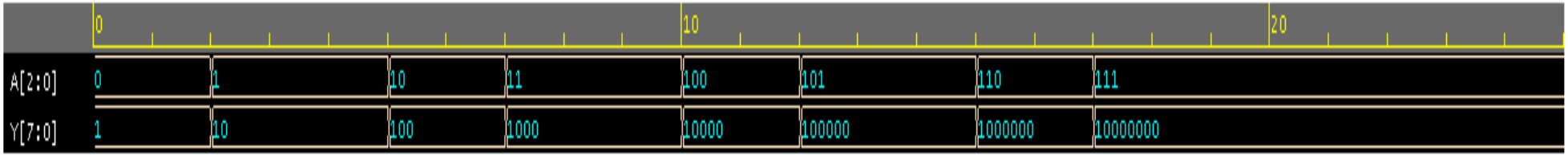
module prencoder_94 (a0,a1,a2,Y);
output reg a0,a1,a2;
input [7:0] Y;
always @ *
begin
{a0,a1,a2}=prencoderfunc_94(Y);
end
function[2:0] prencoderfunc_94;
input [7:0] Y;
begin
assign a2= (Y[4]|Y[5]|Y[6]|Y[7]);
assign a1=((~Y[5]&~Y[4])&(Y[2]|Y[3])|(Y[6]|Y[7]));
assign
a0=((~Y[6]&((~Y[4]&~Y[2]&Y[1])|(~Y[4]&Y[3]|Y[5]))|Y[7]));
end
endfunction
endmodule
	module prencode_94test;
	wire a0,a1,a2;
	reg [7:0] Y;
	
	prencoder_94 g1(a0,a1,a2,Y);
	initial
	begin
	#0 Y = 8'b00000001;
	#2 Y = 8'b0000001x;
	#3 Y = 8'b000001xx;
	#2 Y = 8'b00001xxx;
	#3 Y = 8'b0001xxxx;
	#2 Y = 8'b001xxxxx;
	#3 Y = 8'b01xxxxxx;
	#2 Y = 8'b1xxxxxxx;
	end
	initial
	begin
	$monitor($time,"Y[7] = %d,Y[6] = %d,Y[5] = %d,Y[4] = %d, Y[3] = %d,Y[2] = %d,Y[1] = %d,Y[0] = %d,a0 = %d,a1 = %d,a2=%d", Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1], Y[0], a0, a1, a2);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, prencode_94test);
	end
	endmodule

PR-ENCODER(TASK) 8_3 EDA

DESIGN CODE	TEST BENCH

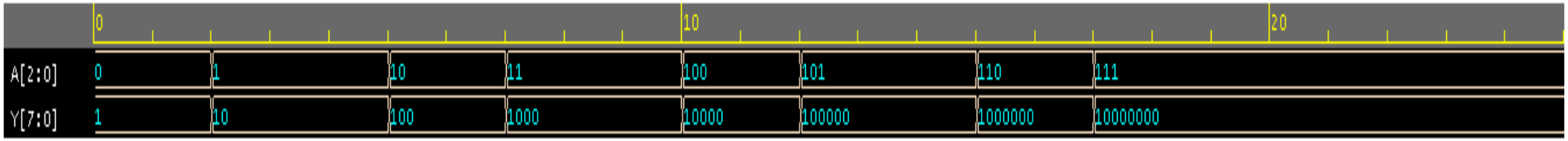
module prencoder_94 (Y,A);
output reg[2:0] A;
input [7:0] Y;
always @ (*)
begin
prencodertsk_94(Y,A);
end
task prencodertsk_94;
input [7:0] Y;
output reg[2:0] A;
begin
case(1'b1)
Y[7] : A=3'b111;
Y[6] : A=3'b110;
Y[5] : A=3'b101;
Y[4] : A=3'b100;
Y[3] : A=3'b011;
Y[2] : A=3'b010;
Y[1] : A=3'b001;
Y[0] : A=3'b000;
endcase
end
endtask
endmodule
	module prencode_94test;
	wire [2:0] A;
	reg [7:0] Y;
	
	prencoder_94 g1(Y,A);
	initial
	begin
	#0 Y = 8'b00000001;
	#2 Y = 8'b0000001x;
	#3 Y = 8'b000001xx;
	#2 Y = 8'b00001xxx;
	#3 Y = 8'b0001xxxx;
	#2 Y = 8'b001xxxxx;
	#3 Y = 8'b01xxxxxx;
	#2 Y = 8'b1xxxxxxx;
	end
	initial
	begin
	$monitor($time,"Y[7] = %d,Y[6] = %d,Y[5] = %d,Y[4] = %d, Y[3] = %d,Y[2] = %d,Y[1] = %d,Y[0] = %d,A[0] = %d,A[1] = %d,A[2]=%d", Y[7], Y[6], Y[5], Y[4], Y[3], Y[2], Y[1], Y[0], A[0], A[1], A[2]);
	end
	initial
	#25 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, prencode_94test);
	end
	endmodule

DECODER 3_8 EDA OUTPUT

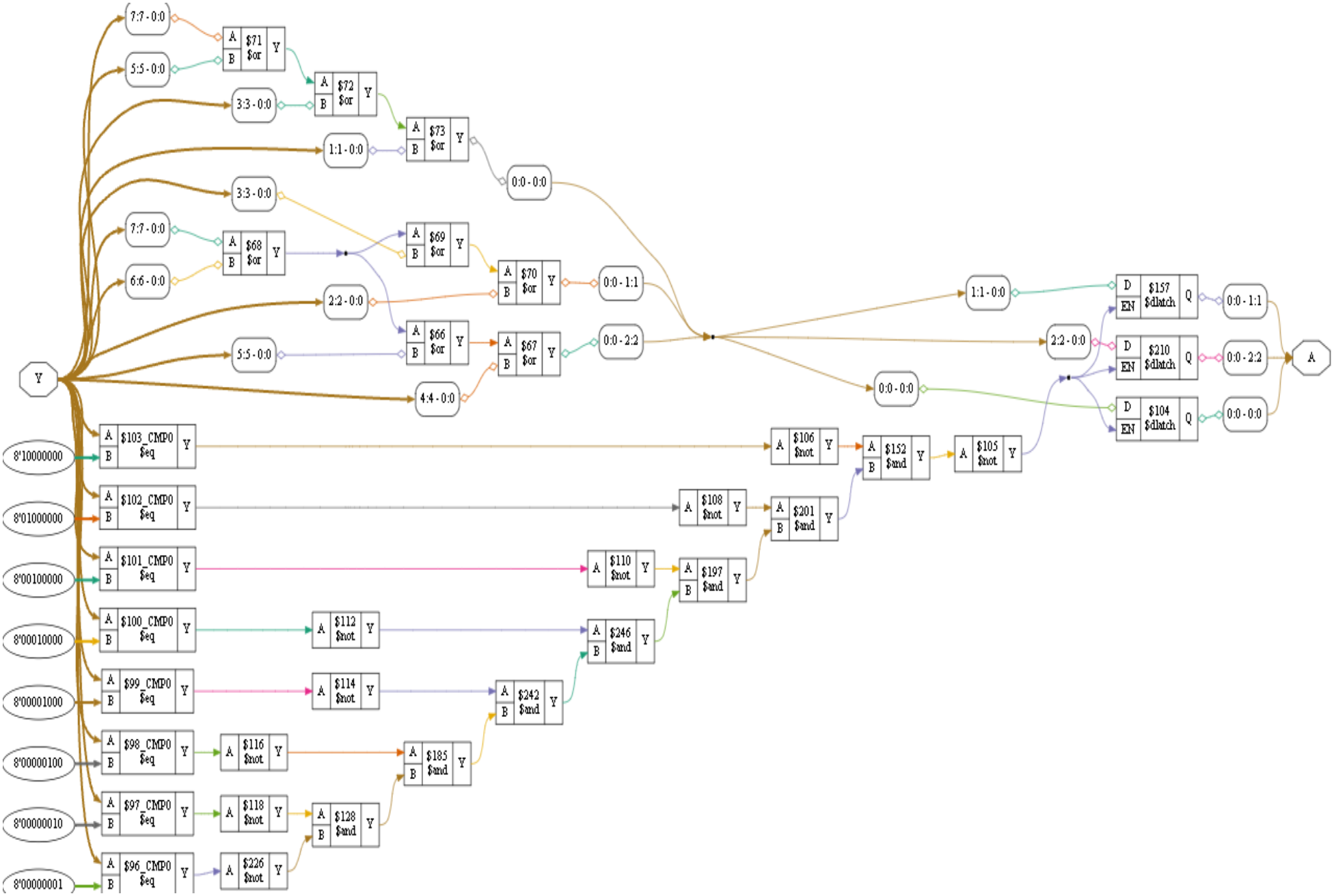



```
//
// Verilog description for cell decode_94,
// Mon Oct 18 11:08:58 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module decode_94 ( Y, A ) ;

output [7:0]Y ;
input [2:0]A ;

wire [2:0]A_int;
wire nx8588z1, nx9585z1, nx10582z1, nx11579z1, nx12576z1, nx13573z1,
nx14570z1, nx15567z1;



OBUF \Y_obuf(0) (.O (Y[0]), .I (nx15567z1)) ;
OBUF \Y_obuf(1) (.O (Y[1]), .I (nx14570z1)) ;
OBUF \Y_obuf(2) (.O (Y[2]), .I (nx13573z1)) ;
OBUF \Y_obuf(3) (.O (Y[3]), .I (nx12576z1)) ;
OBUF \Y_obuf(4) (.O (Y[4]), .I (nx11579z1)) ;
OBUF \Y_obuf(5) (.O (Y[5]), .I (nx10582z1)) ;
OBUF \Y_obuf(6) (.O (Y[6]), .I (nx9585z1)) ;
OBUF \Y_obuf(7) (.O (Y[7]), .I (nx8588z1)) ;
IBUF \A_ibuf(0) (.O (A_int[0]), .I (A[0])) ;
IBUF \A_ibuf(1) (.O (A_int[1]), .I (A[1])) ;
IBUF \A_ibuf(2) (.O (A_int[2]), .I (A[2])) ;
(* HLUTNM = "LUT62_1_4" *)
LUT3 ix8588z1442 (.O (nx8588z1), .I0 (A_int[2]), .I1 (A_int[1]), .I2 (
A_int[0])) ;
defparam ix8588z1442.INIT = 8'h80;
(* HLUTNM = "LUT62_1_4" *)
LUT3 ix9585z1322 (.O (nx9585z1), .I0 (A_int[2]), .I1 (A_int[1]), .I2 (
A_int[0])) ;
defparam ix9585z1322.INIT = 8'h08;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix10582z1346 (.O (nx10582z1), .I0 (A_int[2]), .I1 (A_int[1]), .I2 (
A_int[0])) ;
defparam ix10582z1346.INIT = 8'h20;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix11579z1316 (.O (nx11579z1), .I0 (A_int[2]), .I1 (A_int[1]), .I2 (
A_int[0])) ;
defparam ix11579z1316.INIT = 8'h02;
(* HLUTNM = "LUT62_1_2" *)
LUT3 ix12576z1378 (.O (nx12576z1), .I0 (A_int[2]), .I1 (A_int[1]), .I2 (
A_int[0])) ;
defparam ix12576z1378.INIT = 8'h40;
(* HLUTNM = "LUT62_1_2" *)
LUT3 ix13573z1318 (.O (nx13573z1), .I0 (A_int[2]), .I1 (A_int[1]), .I2 (
A_int[0])) ;
defparam ix13573z1318.INIT = 8'h04;
(* HLUTNM = "LUT62_1_3" *)
LUT3 ix14570z1330 (.O (nx14570z1), .I0 (A_int[2]), .I1 (A_int[1]), .I2 (
A_int[0])) ;
defparam ix14570z1330.INIT = 8'h10;
(* HLUTNM = "LUT62_1_3" *)
LUT3 ix15567z1315 (.O (nx15567z1), .I0 (A_int[2]), .I1 (A_int[1]), .I2 (
A_int[0])) ;
defparam ix15567z1315.INIT = 8'h01;
endmodule
```

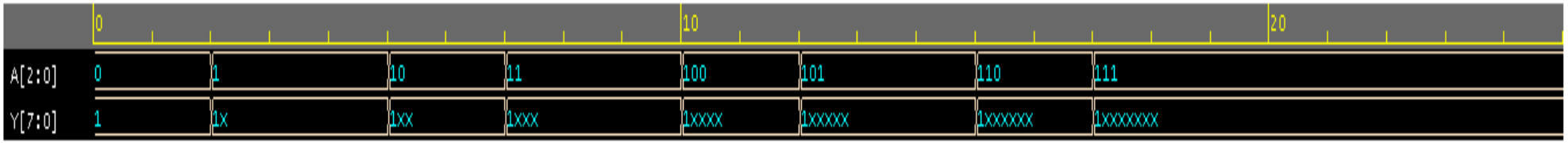
ENCODER 8_3 EDA OUTPUT




```
//
// Verilog description for cell encode_94,
// Mon Oct 18 11:17:39 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module encode_94 ( A, Y ) ;

output [2:0]A ;
input [7:0]Y ;

wire [2:0]A_1_0;
wire [7:0]Y_int;
wire nx44480z2, nx46474z1, nx44480z4, nx44480z3, nx44480z1, nx3, nx4;



OBUF \A_obuf(0) (.O (A[0]), .I (A_1_0[0])) ;
OBUF \A_obuf(1) (.O (A[1]), .I (A_1_0[1])) ;
OBUF \A_obuf(2) (.O (A[2]), .I (A_1_0[2])) ;
IBUF \Y_ibuf(0) (.O (Y_int[0]), .I (Y[0])) ;
IBUF \Y_ibuf(1) (.O (Y_int[1]), .I (Y[1])) ;
IBUF \Y_ibuf(2) (.O (Y_int[2]), .I (Y[2])) ;
IBUF \Y_ibuf(3) (.O (Y_int[3]), .I (Y[3])) ;
IBUF \Y_ibuf(4) (.O (Y_int[4]), .I (Y[4])) ;
IBUF \Y_ibuf(5) (.O (Y_int[5]), .I (Y[5])) ;
IBUF \Y_ibuf(6) (.O (Y_int[6]), .I (Y[6])) ;
IBUF \Y_ibuf(7) (.O (Y_int[7]), .I (Y[7])) ;
LUT5 ix44480z3496 (.O (nx44480z2), .I0 (nx44480z3), .I1 (nx44480z4), .I2 (
Y_int[7]), .I3 (Y_int[6]), .I4 (Y_int[2])) ;
defparam ix44480z3496.INIT = 32'h00050885;
LUT4 ix46474z1315 (.O (nx46474z1), .I0 (Y_int[6]), .I1 (Y_int[4]), .I2 (
Y_int[2]), .I3 (Y_int[0])) ;
defparam ix46474z1315.INIT = 16'h0001;
LUT4 ix44480z1318 (.O (nx44480z4), .I0 (Y_int[5]), .I1 (Y_int[4]), .I2 (
Y_int[1]), .I3 (Y_int[0])) ;
defparam ix44480z1318.INIT = 16'h0001;
LUT6 ix44480z1037 (.O (nx44480z3), .I0 (Y_int[5]), .I1 (Y_int[4]), .I2 (
Y_int[3]), .I3 (Y_int[2]), .I4 (Y_int[1]), .I5 (Y_int[0])) ;
defparam ix44480z1037.INIT = 64'hFFFFFFFFEFFFFEE9;
LUT4 ix44480z1315 (.O (nx44480z1), .I0 (Y_int[3]), .I1 (Y_int[2]), .I2 (
Y_int[1]), .I3 (Y_int[0])) ;
defparam ix44480z1315.INIT = 16'h0001;
VCC \lat_A(2)_I14_LD_PWR (.P (nx3)) ;
GND \lat_A(2)_I14_LD_GND (.G (nx4)) ;
LDCE \lat_A(2) (.Q (A_1_0[2]), .CLR (nx4), .D (nx44480z1), .G (nx44480z2),
.GE (nx3)) ;
LDCE \lat_A(1) (.Q (A_1_0[1]), .CLR (nx4), .D (nx44480z4), .G (nx44480z2),
.GE (nx3)) ;
LDCE \lat_A(0) (.Q (A_1_0[0]), .CLR (nx4), .D (nx46474z1), .G (nx44480z2),
.GE (nx3)) ;
endmodule
```


Y
7:7 - 0:0
5:5 - 0:0
3:3 - 0:0
1:1 - 0:0
6:6 - 0:0
2:2 - 0:0
4:4 - 0:0
$71 $or
$72 $or
$73 $or
$68 $or
$69 $or
$70 $or
$66 $or
$67 $or
0:0 - 0:0
0:0 - 1:1
0:0 - 2:2
$103_CMP0 $eq
$102_CMP0 $eq
$101_CMP0 $eq
$100_CMP0 $eq
$99_CMP0 $eq
$98_CMP0 $eq
$97_CMP0 $eq
$96_CMP0 $eq
8'10000000
8'01000000
8'00100000
8'00010000
8'00001000
8'00000100
8'00000010
8'00000001
$106 $not
$108 $not
$110 $not
$112 $not
$114 $not
$116 $not
$118 $not
$226 $not
$152 $and
$201 $and
$197 $and
$246 $and
$242 $and
$185 $and
$128 $and
$105 $not
1:1 - 0:0
2:2 - 0:0
0:0 - 0:0
$157 $dlatch
$210 $dlatch
$104 $dlatch
0:0 - 1:1
0:0 - 2:2
0:0 - 0:0
A

PR-ENCODER 8_3 EDA OUTPUT




```
//
// Verilog description for cell prencoder_94,
// Mon Oct 18 11:26:05 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module prencoder_94 ( Y, A ) ;

input [7:0]Y ;
output [2:0]A ;

wire [2:0]A_1_0;
wire [7:0]Y_int;
wire nx46474z1, nx45477z1, nx44480z2, nx44480z1, nx46474z2, nx44480z3, nx3,
nx4;



OBUF \A_obuf(0) (.O (A[0]), .I (A_1_0[0])) ;
OBUF \A_obuf(1) (.O (A[1]), .I (A_1_0[1])) ;
OBUF \A_obuf(2) (.O (A[2]), .I (A_1_0[2])) ;
IBUF \Y_ibuf(0) (.O (Y_int[0]), .I (Y[0])) ;
IBUF \Y_ibuf(1) (.O (Y_int[1]), .I (Y[1])) ;
IBUF \Y_ibuf(2) (.O (Y_int[2]), .I (Y[2])) ;
IBUF \Y_ibuf(3) (.O (Y_int[3]), .I (Y[3])) ;
IBUF \Y_ibuf(4) (.O (Y_int[4]), .I (Y[4])) ;
IBUF \Y_ibuf(5) (.O (Y_int[5]), .I (Y[5])) ;
IBUF \Y_ibuf(6) (.O (Y_int[6]), .I (Y[6])) ;
IBUF \Y_ibuf(7) (.O (Y_int[7]), .I (Y[7])) ;
LUT4 ix46474z54512 (.O (nx46474z1), .I0 (nx46474z2), .I1 (Y_int[7]), .I2 (
Y_int[6]), .I3 (Y_int[5])) ;
defparam ix46474z54512.INIT = 16'hCFCE;
LUT6 ix45477z62480 (.O (nx45477z1), .I0 (Y_int[7]), .I1 (Y_int[6]), .I2 (
Y_int[5]), .I3 (Y_int[4]), .I4 (Y_int[3]), .I5 (Y_int[2])) ;
defparam ix45477z62480.INIT = 64'hEEEFEEEFEEEFEEEE;
LUT5 ix44480z1316 (.O (nx44480z2), .I0 (nx44480z3), .I1 (Y_int[3]), .I2 (
Y_int[2]), .I3 (Y_int[1]), .I4 (Y_int[0])) ;
defparam ix44480z1316.INIT = 32'hFFFFFFFD;
(* HLUTNM = "LUT62_1_1" *)
LUT4 ix44480z1312 (.O (nx44480z1), .I0 (Y_int[7]), .I1 (Y_int[6]), .I2 (
Y_int[5]), .I3 (Y_int[4])) ;
defparam ix44480z1312.INIT = 16'hFFFE;
LUT4 ix46474z19047 (.O (nx46474z2), .I0 (Y_int[4]), .I1 (Y_int[3]), .I2 (
Y_int[2]), .I3 (Y_int[1])) ;
defparam ix46474z19047.INIT = 16'h4544;
(* HLUTNM = "LUT62_1_1" *)
LUT4 ix44480z1318 (.O (nx44480z3), .I0 (Y_int[7]), .I1 (Y_int[6]), .I2 (
Y_int[5]), .I3 (Y_int[4])) ;
defparam ix44480z1318.INIT = 16'h0001;
VCC \lat_A(2)_I14_LD_PWR (.P (nx3)) ;
GND \lat_A(2)_I14_LD_GND (.G (nx4)) ;
LDCE \lat_A(2) (.Q (A_1_0[2]), .CLR (nx4), .D (nx44480z1), .G (nx44480z2),
.GE (nx3)) ;
LDCE \lat_A(1) (.Q (A_1_0[1]), .CLR (nx4), .D (nx45477z1), .G (nx44480z2),
.GE (nx3)) ;
LDCE \lat_A(0) (.Q (A_1_0[0]), .CLR (nx4), .D (nx46474z1), .G (nx44480z2),
.GE (nx3)) ;
endmodule
```

Date:
09-09-2021

Experiment No. 5

Code converters and comparator

Aim: To implement code converters (gray to Binary, Binary to gray) and comparator using verilog, verify the design using Testbench and perform functional simulation and to synthesize.

Requirement: EDA playground Tool

Theory:

=> Code converters:

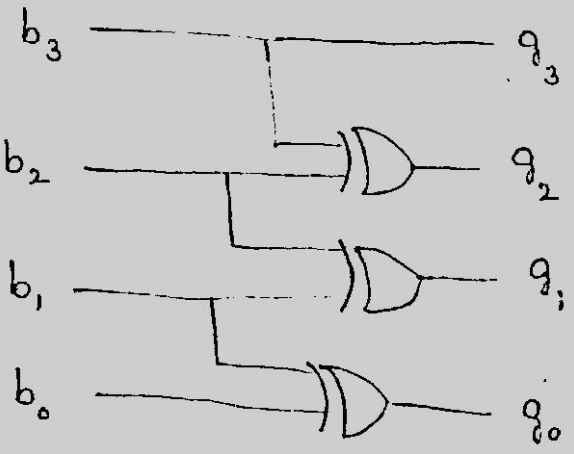
* Binary to Gray code/ Gray to Binary:

Binary code				Gray code			
b_3	b_2	b_1	b_0	g_0	g_1	g_2	g_3
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0

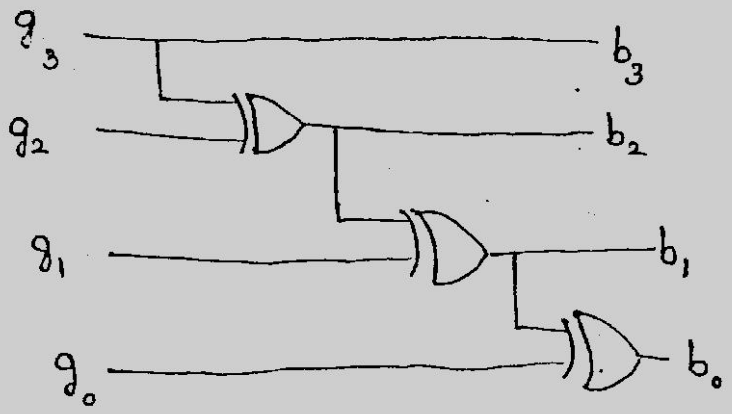
b_3	b_2	b_1	b_0	g_3	g_2	g_1	g_0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Circuit Diagram:

* Binary to Gray code:

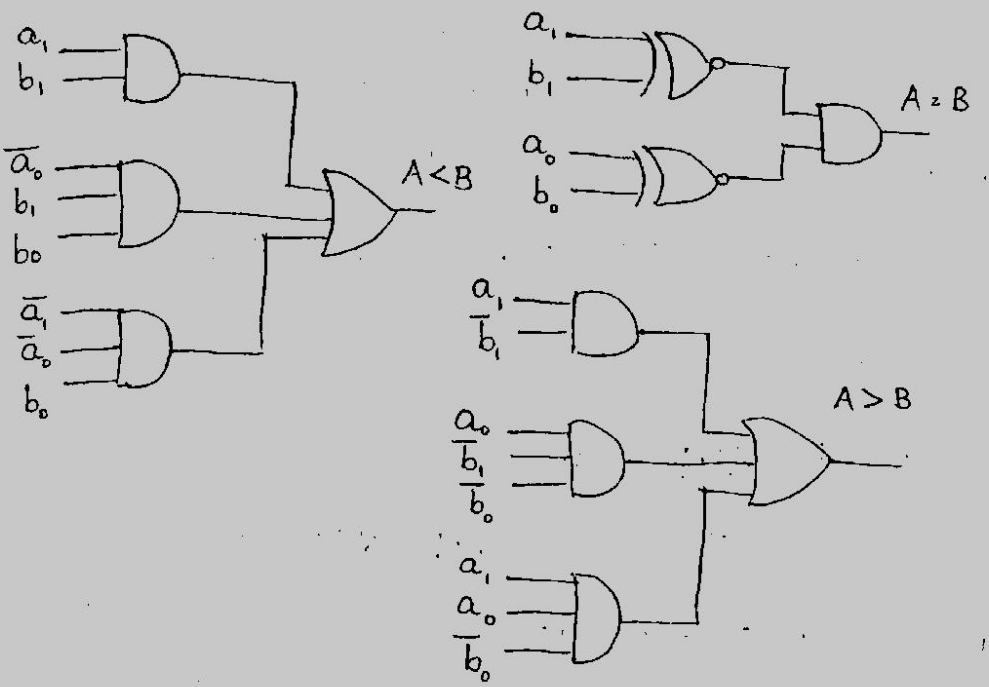


* Gray to Binary Code:



=> 2 - bit comparator:

A comparator used to compare two binary numbers each of two bits is called a 2 - bit magnitude comparator.



a_1	a_0	b_1	b_0	$a < b$	$a = b$	$a > b$
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

B-G(BEHAVIOURAL) EDA

DESIGN CODE	TEST BENCH

module bg_94(output reg[3:0]g,input[3:0]b);
always@(*)
begin
case(b)
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;
4'b1111:g=4'b1000;
endcase
end
endmodule
	module bg_94test;
	reg [3:0]b;
	wire [3:0]g;
	bg_94 u1(g,b);
	initial
	begin
	#0 b[3]=0; b[2]=0; b[1]=0; b[0]=0;
	#1 b[3]=0; b[2]=0; b[1]=0; b[0]=1;
	#1 b[3]=0; b[2]=0; b[1]=1; b[0]=0;
	#1 b[3]=0; b[2]=0; b[1]=1; b[0]=1;
	#1 b[3]=0; b[2]=1; b[1]=0; b[0]=0;
	#1 b[3]=0; b[2]=1; b[1]=0; b[0]=1;
	#1 b[3]=0; b[2]=1; b[1]=1; b[0]=0;
	#1 b[3]=0; b[2]=1; b[1]=1; b[0]=1;
	#1 b[3]=1; b[2]=0; b[1]=0; b[0]=0;
	#1 b[3]=1; b[2]=0; b[1]=0; b[0]=1;
	#1 b[3]=1; b[2]=0; b[1]=1; b[0]=0;
	#1 b[3]=1; b[2]=0; b[1]=1; b[0]=1;
	#1 b[3]=1; b[2]=1; b[1]=0; b[0]=0;
	#1 b[3]=1; b[2]=1; b[1]=0; b[0]=1;
	#1 b[3]=1; b[2]=1; b[1]=1; b[0]=0;
	#1 b[3]=1; b[2]=1; b[1]=1; b[0]=1;
	end
	initial
	$monitor($time, "b[3] = %d, b[2] = %d, b[1] = %d, b[0] = %d, g[3] = %d, g[2] = %d, g[1] = %d, g[0] = %d",b[3],b[2],b[1],b[0],g[3],g[2],g[1],g[0]);
	initial
	#30 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1,bg_94test);
	end
	endmodule

B-G(DF) EDA

DESIGN CODE	TEST BENCH

module bg_94(output reg[3:0]g,input[3:0]b);
assign g[3] = b[3];
assign g[2] = b[3]^b[2];
assign g[1] = b[2]^b[1];
assign g[0] = b[1]^b[0];
endmodule
	module bg_94test;
	reg [3:0]b;
	wire [3:0]g;
	bg_94 u1(g,b);
	initial
	begin
	#0 b[3]=0; b[2]=0; b[1]=0; b[0]=0;
	#1 b[3]=0; b[2]=0; b[1]=0; b[0]=1;
	#1 b[3]=0; b[2]=0; b[1]=1; b[0]=0;
	#1 b[3]=0; b[2]=0; b[1]=1; b[0]=1;
	#1 b[3]=0; b[2]=1; b[1]=0; b[0]=0;
	#1 b[3]=0; b[2]=1; b[1]=0; b[0]=1;
	#1 b[3]=0; b[2]=1; b[1]=1; b[0]=0;
	#1 b[3]=0; b[2]=1; b[1]=1; b[0]=1;
	#1 b[3]=1; b[2]=0; b[1]=0; b[0]=0;
	#1 b[3]=1; b[2]=0; b[1]=0; b[0]=1;
	#1 b[3]=1; b[2]=0; b[1]=1; b[0]=0;
	#1 b[3]=1; b[2]=0; b[1]=1; b[0]=1;
	#1 b[3]=1; b[2]=1; b[1]=0; b[0]=0;
	#1 b[3]=1; b[2]=1; b[1]=0; b[0]=1;
	#1 b[3]=1; b[2]=1; b[1]=1; b[0]=0;
	#1 b[3]=1; b[2]=1; b[1]=1; b[0]=1;
	end
	initial
	$monitor($time, "b[3] = %d, b[2] = %d, b[1] = %d, b[0] = %d, g[3] = %d, g[2] = %d, g[1] = %d, g[0] = %d",b[3],b[2],b[1],b[0],g[3],g[2],g[1],g[0]);
	initial
	#30 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1,bg_94test);
	end
	endmodule

B-G(FUNC) EDA

DESIGN CODE	TEST BENCH

module bg_94(g0,g1,g2,g3,b);
input[3:0]b;
output reg g0,g1,g2,g3;
always@(*)
begin
{g0,g1,g2,g3}=bgfunc_94(b);
end
function [3:0] bgfunc_94;
input[3:0]b;
begin
assign g3 = b[3];
assign g2 = b[3]^b[2];
assign g1 = b[2]^b[1];
assign g0 = b[1]^b[0];
end
endfunction
endmodule
	module bg_94test;
	reg [3:0]b;
	wire g0,g1,g2,g3;
	bg_94 u1(g0,g1,g2,g3,b);
	initial
	begin
	#0 b[3]=0; b[2]=0; b[1]=0; b[0]=0;
	#1 b[3]=0; b[2]=0; b[1]=0; b[0]=1;
	#1 b[3]=0; b[2]=0; b[1]=1; b[0]=0;
	#1 b[3]=0; b[2]=0; b[1]=1; b[0]=1;
	#1 b[3]=0; b[2]=1; b[1]=0; b[0]=0;
	#1 b[3]=0; b[2]=1; b[1]=0; b[0]=1;
	#1 b[3]=0; b[2]=1; b[1]=1; b[0]=0;
	#1 b[3]=0; b[2]=1; b[1]=1; b[0]=1;
	#1 b[3]=1; b[2]=0; b[1]=0; b[0]=0;
	#1 b[3]=1; b[2]=0; b[1]=0; b[0]=1;
	#1 b[3]=1; b[2]=0; b[1]=1; b[0]=0;
	#1 b[3]=1; b[2]=0; b[1]=1; b[0]=1;
	#1 b[3]=1; b[2]=1; b[1]=0; b[0]=0;
	#1 b[3]=1; b[2]=1; b[1]=0; b[0]=1;
	#1 b[3]=1; b[2]=1; b[1]=1; b[0]=0;
	#1 b[3]=1; b[2]=1; b[1]=1; b[0]=1;
	end
	initial
	$monitor($time, "b[3] = %d, b[2] = %d, b[1] = %d, b[0] = %d, g3 = %d, g2 = %d, g1 = %d, g0 = %d",b[3],b[2],b[1],b[0],g3,g2,g1,g0);
	initial
	#30 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1,bg_94test);
	end
	endmodule

B-G(GL) EDA

DESIGN CODE	TEST BENCH

module bg_94(output reg[3:0]g,input[3:0]b);
  buf g1(g[3],b[3]);
  xor g2(g[2],b[2],b[3]);
  xor g3(g[1],b[1],b[2]);
  xor g4(g[0],b[0],b[1]);
endmodule
	module bg_94test;
	  reg [3:0]b;
	  wire [3:0]g;
	  bg_94 u1(g,b);
	  initial
	  begin
	  #0 b[3]=0; b[2]=0; b[1]=0; b[0]=0;
	  #1 b[3]=0; b[2]=0; b[1]=0; b[0]=1;
	  #1 b[3]=0; b[2]=0; b[1]=1; b[0]=0;
	  #1 b[3]=0; b[2]=0; b[1]=1; b[0]=1;
	  #1 b[3]=0; b[2]=1; b[1]=0; b[0]=0;
	  #1 b[3]=0; b[2]=1; b[1]=0; b[0]=1;
	  #1 b[3]=0; b[2]=1; b[1]=1; b[0]=0;
	  #1 b[3]=0; b[2]=1; b[1]=1; b[0]=1;
	  #1 b[3]=1; b[2]=0; b[1]=0; b[0]=0;
	  #1 b[3]=1; b[2]=0; b[1]=0; b[0]=1;
	  #1 b[3]=1; b[2]=0; b[1]=1; b[0]=0;
	  #1 b[3]=1; b[2]=0; b[1]=1; b[0]=1;
	  #1 b[3]=1; b[2]=1; b[1]=0; b[0]=0;
	  #1 b[3]=1; b[2]=1; b[1]=0; b[0]=1;
	  #1 b[3]=1; b[2]=1; b[1]=1; b[0]=0;
	  #1 b[3]=1; b[2]=1; b[1]=1; b[0]=1;
	  end
	  initial
	  $monitor($time, "b[3] = %d, b[2] = %d, b[1] = %d, b[0] = %d, g[3] = %d, g[2] = %d, g[1] = %d, g[0] = %d",b[3],b[2],b[1],b[0],g[3],g[2],g[1],g[0]);
	  initial
	  #30 $stop;
	  initial
	  begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1,bg_94test);
	  end
	endmodule

B-G(TASK) EDA

DESIGN CODE	TEST BENCH

module bg_94(output reg[3:0]g,input[3:0]b);
always@(*)
begin
bgtsk_94(b,g);
end
task bgtsk_94;
input[3:0]b;
output reg[3:0]g;
begin
g[3] = b[3];
g[2] = b[3]^b[2];
g[1] = b[2]^b[1];
g[0] = b[1]^b[0];
end
endtask
endmodule
	module bg_94test;
	reg [3:0]b;
	wire [3:0]g;
	bg_94 u1(g,b);
	initial
	begin
	#0 b[3]=0; b[2]=0; b[1]=0; b[0]=0;
	#1 b[3]=0; b[2]=0; b[1]=0; b[0]=1;
	#1 b[3]=0; b[2]=0; b[1]=1; b[0]=0;
	#1 b[3]=0; b[2]=0; b[1]=1; b[0]=1;
	#1 b[3]=0; b[2]=1; b[1]=0; b[0]=0;
	#1 b[3]=0; b[2]=1; b[1]=0; b[0]=1;
	#1 b[3]=0; b[2]=1; b[1]=1; b[0]=0;
	#1 b[3]=0; b[2]=1; b[1]=1; b[0]=1;
	#1 b[3]=1; b[2]=0; b[1]=0; b[0]=0;
	#1 b[3]=1; b[2]=0; b[1]=0; b[0]=1;
	#1 b[3]=1; b[2]=0; b[1]=1; b[0]=0;
	#1 b[3]=1; b[2]=0; b[1]=1; b[0]=1;
	#1 b[3]=1; b[2]=1; b[1]=0; b[0]=0;
	#1 b[3]=1; b[2]=1; b[1]=0; b[0]=1;
	#1 b[3]=1; b[2]=1; b[1]=1; b[0]=0;
	#1 b[3]=1; b[2]=1; b[1]=1; b[0]=1;
	end
	initial
	$monitor($time, "b[3] = %d, b[2] = %d, b[1] = %d, b[0] = %d, g[3] = %d, g[2] = %d, g[1] = %d, g[0] = %d",b[3],b[2],b[1],b[0],g[3],g[2],g[1],g[0]);
	initial
	#30 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1,bg_94test);
	end
	endmodule

COMPARATOR_2B(BEHAVIOURAL) EDA

DESIGN CODE	TEST BENCH

module compare2_94(l,e,g,a,b);
input [1:0]a;
input [1:0]b;
output reg l,e,g;
always@(*)
begin
if(a>b)
begin e=0;l=0;g=1;end
else if(a<b)
begin e=0;l=1;g=0;end
else
begin e=1;l=0;g=0;end
end
endmodule
	module compare2_94test;
	reg [1:0]a;
	reg [1:0]b;
	wire l,e,g;
	compare2_94 g1(l,e,g,a,b);
	initial
	begin
	#0 a[1]=0;a[0]=0;b[1]=0;b[0]=0;
	#1 a[1]=0;a[0]=0;b[1]=0;b[0]=1;
	#1 a[1]=0;a[0]=0;b[1]=1;b[0]=0;
	#1 a[1]=0;a[0]=0;b[1]=1;b[0]=1;
	#1 a[1]=0;a[0]=1;b[1]=0;b[0]=0;
	#1 a[1]=0;a[0]=1;b[1]=0;b[0]=1;
	#1 a[1]=0;a[0]=1;b[1]=1;b[0]=0;
	#1 a[1]=0;a[0]=1;b[1]=1;b[0]=1;
	#1 a[1]=1;a[0]=0;b[1]=0;b[0]=0;
	#1 a[1]=1;a[0]=0;b[1]=0;b[0]=1;
	#1 a[1]=1;a[0]=0;b[1]=1;b[0]=0;
	#1 a[1]=1;a[0]=0;b[1]=1;b[0]=1;
	#1 a[1]=1;a[0]=1;b[1]=0;b[0]=0;
	#1 a[1]=1;a[0]=1;b[1]=0;b[0]=1;
	#1 a[1]=1;a[0]=1;b[1]=1;b[0]=0;
	#1 a[1]=1;a[0]=1;b[1]=1;b[0]=1;
	end
	initial
	
	$monitor($time,"l=%b,e=%b,g=%b,a[1]=%b,a[0]=%b,b[1]=%b,b[0]=%b",l,e,g,a[1],a[0],b[1],b[0]);
	initial
	#25 $stop;
	
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1,compare2_94test);
	end
	endmodule

COMPARATOR_2B(DF) EDA

DESIGN CODE	TEST BENCH

module compare2_94(l,e,g,a1,a0,b1,b0);
input a1,a0,b1,b0;
output l,e,g;
assign e = (a0~^b0)&(a1~^b1);
assign g = ((a0)&(~b1)&(~b0))
	((a1)&(~b1))	((a1)&(a0)&(~b0));
		assign l = ((~a1)&(b1))
			((~a0)&(b1)&(b0))	((~a1)&(~a0)&(b0));
				endmodule
					module compare2_94test;
					reg a1,a0,b1,b0;
					wire l,e,g;
					compare2_94 g1(l,e,g,a1,a0,b1,b0);
					initial
					begin
					#0 a1=0;a0=0;b1=0;b0=0;
					#1 a1=0;a0=0;b1=0;b0=1;
					#1 a1=0;a0=0;b1=1;b0=0;
					#1 a1=0;a0=0;b1=1;b0=1;
					#1 a1=0;a0=1;b1=0;b0=0;
					#1 a1=0;a0=1;b1=0;b0=1;
					#1 a1=0;a0=1;b1=1;b0=0;
					#1 a1=0;a0=1;b1=1;b0=1;
					#1 a1=1;a0=0;b1=0;b0=0;
					#1 a1=1;a0=0;b1=0;b0=1;
					#1 a1=1;a0=0;b1=1;b0=0;
					#1 a1=1;a0=0;b1=1;b0=1;
					#1 a1=1;a0=1;b1=0;b0=0;
					#1 a1=1;a0=1;b1=0;b0=1;
					#1 a1=1;a0=1;b1=1;b0=0;
					#1 a1=1;a0=1;b1=1;b0=1;
					end
					initial
					$monitor($time,"l=%b,e=%b,g=%b,a1=%b,a0=%b,
					b1=%b,b0=%b",l,e,g,a1,a0,b1,b0);
					initial
					#25 $stop;
					
					initial
					begin
					$dumpfile("dump.vcd");
					$dumpvars(1,compare2_94test);
					end
					endmodule

COMPARATOR_2B(FUNC)

DESIGN CODE	TEST BENCH

module compare2_94(l,e,g,a,b);
input [1:0]a;
input [1:0]b;
output reg l,e,g;
always@(*)
begin
{e,g,l}=compare2func_94(a,b);
end
function [2:0]compare2func_94;
input [1:0]a;
input [1:0]b;
begin
assign e = (a[0]~^b[0])&(a[1]~^b[1]);
assign g = ((a[0])&(~b[1])&(~b[0])) |
((a[1])&(~b[1])) | ((a[1])&(a[0])&(~b[0]));
assign l = ((~a[1])&(b[1])) |
((~a[0])&(b[1])&(b[0])) |
((~a[1])&(~a[0])&(b[0]));
end
endfunction
endmodule
	module compare2_94test;
	reg [1:0]a;
	reg [1:0]b;
	wire l,e,g;
	compare2_94 g1(l,e,g,a,b);
	initial
	begin
	#0 a[1]=0;a[0]=0;b[1]=0;b[0]=0;
	#1 a[1]=0;a[0]=0;b[1]=0;b[0]=1;
	#1 a[1]=0;a[0]=0;b[1]=1;b[0]=0;
	#1 a[1]=0;a[0]=0;b[1]=1;b[0]=1;
	#1 a[1]=0;a[0]=1;b[1]=0;b[0]=0;
	#1 a[1]=0;a[0]=1;b[1]=0;b[0]=1;
	#1 a[1]=0;a[0]=1;b[1]=1;b[0]=0;
	#1 a[1]=0;a[0]=1;b[1]=1;b[0]=1;
	#1 a[1]=1;a[0]=0;b[1]=0;b[0]=0;
	#1 a[1]=1;a[0]=0;b[1]=0;b[0]=1;
	#1 a[1]=1;a[0]=0;b[1]=1;b[0]=0;
	#1 a[1]=1;a[0]=0;b[1]=1;b[0]=1;
	#1 a[1]=1;a[0]=1;b[1]=0;b[0]=0;
	#1 a[1]=1;a[0]=1;b[1]=0;b[0]=1;
	#1 a[1]=1;a[0]=1;b[1]=1;b[0]=0;
	#1 a[1]=1;a[0]=1;b[1]=1;b[0]=1;
	end
	initial
	$monitor($time,"l=%b,e=%b,g=%b,
	a[1]=%b,a[0]=%b,b[1]=%b,b[0]=%b",l,e,g,a[1],
	a[0],b[1],b[0]);
	initial
	#25 $stop;
	
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1,compare2_94test);
	end
	endmodule

COMPARATOR_2B(GL) EDA

DESIGN CODE	TEST BENCH

module compare2_94(l,e,g,a1,a0,b1,b0);
input a1,a0,b1,b0;
output l,e,g;
wire w1,w2,w3,w4,w5,w6,w7,w8;
xnor (w1,a0,b0),(w2,a1,b1);
and (e,w1,w2),
(w3,a0,~b1,~b0),(w4,a1,~b1),(w5,a1,a0,~b0),(w6,~a1,b1),(w7,~a0,b1,b0),(w8,~a1,~a0,b0);
or (g,w3,w4,w5),(l,w6,w7,w8);
endmodule
	module compare2_94test;
	reg a1,a0,b1,b0;
	wire l,e,g;
	compare2_94 g1(l,e,g,a1,a0,b1,b0);
	initial
	begin
	#0 a1=0;a0=0;b1=0;b0=0;
	#1 a1=0;a0=0;b1=0;b0=1;
	#1 a1=0;a0=0;b1=1;b0=0;
	#1 a1=0;a0=0;b1=1;b0=1;
	#1 a1=0;a0=1;b1=0;b0=0;
	#1 a1=0;a0=1;b1=0;b0=1;
	#1 a1=0;a0=1;b1=1;b0=0;
	#1 a1=0;a0=1;b1=1;b0=1;
	#1 a1=1;a0=0;b1=0;b0=0;
	#1 a1=1;a0=0;b1=0;b0=1;
	#1 a1=1;a0=0;b1=1;b0=0;
	#1 a1=1;a0=0;b1=1;b0=1;
	#1 a1=1;a0=1;b1=0;b0=0;
	#1 a1=1;a0=1;b1=0;b0=1;
	#1 a1=1;a0=1;b1=1;b0=0;
	#1 a1=1;a0=1;b1=1;b0=1;
	end
	initial
	
	$monitor($time,"l=%b,e=%b,g=%b,a1=%b,a0=%b,b1=%b,b0=%b",l,e,g,a1,a0,b1,b0);
	initial
	#25 $stop;
	
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1,compare2_94test);
	end
	endmodule

COMPARATOR_2B(TASK) EDA

DESIGN CODE	TEST BENCH

module compare2_94(l,e,g,a,b);
input [1:0]a;
input [1:0]b;
output reg l,e,g;
always@(*)
begin
compare2tsk_94(a,b);
end
task compare2tsk_94;
input [1:0]a;
input [1:0]b;
begin
assign e = (a[0]~^b[0])&(a[1]~^b[1]);
assign g = ((a[0])&(~b[1])&(~b[0])) |
((a[1])&(~b[1])) | ((a[1])&(a[0])&(~b[0]));
assign l = ((~a[1])&(b[1])) | ((~a[0])&(b[1])&(b[0]))
| ((~a[1])&(~a[0])&(b[0]));
end
endtask
endmodule
	module compare2_94test;
	reg [1:0]a;
	reg [1:0]b;
	wire l,e,g;
	compare2_94 g1(l,e,g,a,b);
	initial
	begin
	#0 a[1]=0;a[0]=0;b[1]=0;b[0]=0;
	#1 a[1]=0;a[0]=0;b[1]=0;b[0]=1;
	#1 a[1]=0;a[0]=0;b[1]=1;b[0]=0;
	#1 a[1]=0;a[0]=0;b[1]=1;b[0]=1;
	#1 a[1]=0;a[0]=1;b[1]=0;b[0]=0;
	#1 a[1]=0;a[0]=1;b[1]=0;b[0]=1;
	#1 a[1]=0;a[0]=1;b[1]=1;b[0]=0;
	#1 a[1]=0;a[0]=1;b[1]=1;b[0]=1;
	#1 a[1]=1;a[0]=0;b[1]=0;b[0]=0;
	#1 a[1]=1;a[0]=0;b[1]=0;b[0]=1;
	#1 a[1]=1;a[0]=0;b[1]=1;b[0]=0;
	#1 a[1]=1;a[0]=0;b[1]=1;b[0]=1;
	#1 a[1]=1;a[0]=1;b[1]=0;b[0]=0;
	#1 a[1]=1;a[0]=1;b[1]=0;b[0]=1;
	#1 a[1]=1;a[0]=1;b[1]=1;b[0]=0;
	#1 a[1]=1;a[0]=1;b[1]=1;b[0]=1;
	end
	initial
	$monitor($time,"l=%b,e=%b,g=%b,a[1]=%b,
	a[0]=%b,b[1]=%b,b[0]=%b",l,e,g,a[1],a[0],b[1],b[0]);
	initial
	#25 $stop;
	
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1,compare2_94test);
	end
	endmodule

G-B(BEHAVIOURAL) EDA

DESIGN CODE	TEST BENCH

module gb_94(output reg[3:0]b,input[3:0]g);
always@(*)
begin
case(g)
4'b0000:b=4'b0000;
4'b0001:b=4'b0001;
4'b0011:b=4'b0010;
4'b0010:b=4'b0011;
4'b0110:b=4'b0100;
4'b0111:b=4'b0101;
4'b0101:b=4'b0110;
4'b0100:b=4'b0111;
4'b1100:b=4'b1000;
4'b1101:b=4'b1001;
4'b1111:b=4'b1010;
4'b1110:b=4'b1011;
4'b1010:b=4'b1100;
4'b1011:b=4'b1101;
4'b1001:b=4'b1110;
4'b1000:b=4'b1111;
endcase
end
endmodule
	module gb_94test;
	reg [3:0]g;
	wire [3:0]b;
	gb_94 u1(b,g);
	initial
	begin
	#0 g[3]=0; g[2]=0; g[1]=0; g[0]=0;
	#1 g[3]=0; g[2]=0; g[1]=0; g[0]=1;
	#1 g[3]=0; g[2]=0; g[1]=1; g[0]=1;
	#1 g[3]=0; g[2]=0; g[1]=1; g[0]=0;
	#1 g[3]=0; g[2]=1; g[1]=1; g[0]=0;
	#1 g[3]=0; g[2]=1; g[1]=1; g[0]=1;
	#1 g[3]=0; g[2]=1; g[1]=0; g[0]=1;
	#1 g[3]=0; g[2]=1; g[1]=0; g[0]=0;
	#1 g[3]=1; g[2]=1; g[1]=0; g[0]=0;
	#1 g[3]=1; g[2]=1; g[1]=0; g[0]=1;
	#1 g[3]=1; g[2]=1; g[1]=1; g[0]=1;
	#1 g[3]=1; g[2]=1; g[1]=1; g[0]=0;
	#1 g[3]=1; g[2]=0; g[1]=1; g[0]=0;
	#1 g[3]=1; g[2]=0; g[1]=1; g[0]=1;
	#1 g[3]=1; g[2]=0; g[1]=0; g[0]=1;
	#1 g[3]=1; g[2]=0; g[1]=0; g[0]=0;
	end
	initial
	$monitor($time, "b[3] = %d, b[2] = %d, b[1] = %d, b[0] = %d, g[3] = %d, g[2] = %d, g[1] = %d, g[0] = %d",b[3],b[2],b[1],b[0],g[3],g[2],g[1],g[0]);
	initial
	#30 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1,gb_94test);
	end
	endmodule

G-B(DF) EDA

DESIGN CODE	TEST BENCH

module gb_94(output reg[3:0]b,input[3:0]g);
  assign b[3]=g[3];
  assign b[2]=g[3]^g[2];
  assign b[1]=g[3]^g[2]^g[1];
  assign b[0]=g[3]^g[2]^g[1]^g[0];
endmodule
	module gb_94test;
	  reg [3:0]g;
	  wire [3:0]b;
	  gb_94 u1(b,g);
	  initial
	  begin
	  #0 g[3]=0; g[2]=0; g[1]=0; g[0]=0;
	  #1 g[3]=0; g[2]=0; g[1]=0; g[0]=1;
	  #1 g[3]=0; g[2]=0; g[1]=1; g[0]=1;
	  #1 g[3]=0; g[2]=0; g[1]=1; g[0]=0;
	  #1 g[3]=0; g[2]=1; g[1]=1; g[0]=0;
	  #1 g[3]=0; g[2]=1; g[1]=1; g[0]=1;
	  #1 g[3]=0; g[2]=1; g[1]=0; g[0]=1;
	  #1 g[3]=0; g[2]=1; g[1]=0; g[0]=0;
	  #1 g[3]=1; g[2]=1; g[1]=0; g[0]=0;
	  #1 g[3]=1; g[2]=1; g[1]=0; g[0]=1;
	  #1 g[3]=1; g[2]=1; g[1]=1; g[0]=1;
	  #1 g[3]=1; g[2]=1; g[1]=1; g[0]=0;
	  #1 g[3]=1; g[2]=0; g[1]=1; g[0]=0;
	  #1 g[3]=1; g[2]=0; g[1]=1; g[0]=1;
	  #1 g[3]=1; g[2]=0; g[1]=0; g[0]=1;
	  #1 g[3]=1; g[2]=0; g[1]=0; g[0]=0;
	  end
	  initial
	  $monitor($time, "b[3] = %d, b[2] = %d, b[1] = %d, b[0] = %d, g[3] = %d, g[2] = %d, g[1] = %d, g[0] = %d",b[3],b[2],b[1],b[0],g[3],g[2],g[1],g[0]);
	  initial
	  #30 $stop;
	  initial
	  begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1,gb_94test);
	  end
	endmodule

G-B(FUNC) EDA

DESIGN CODE	TEST BENCH

module gb_94(g,b0,b1,b2,b3);
input[3:0]g;
output reg b0,b1,b2,b3;
always@(*)
begin
{b0,b1,b2,b3}=gbfunc_94(g);
end
function [3:0]gbfunc_94;
input[3:0]g;
begin
assign b3=g[3];
assign b2=g[3]^g[2];
assign b1=g[3]^g[2]^g[1];
assign b0=g[3]^g[2]^g[1]^g[0];
end
endfunction
endmodule
	module gb_94test;
	reg [3:0]g;
	wire b0,b1,b2,b3;
	gb_94 u1(g,b0,b1,b2,b3);
	initial
	begin
	#0 g[3]=0; g[2]=0; g[1]=0; g[0]=0;
	#1 g[3]=0; g[2]=0; g[1]=0; g[0]=1;
	#1 g[3]=0; g[2]=0; g[1]=1; g[0]=1;
	#1 g[3]=0; g[2]=0; g[1]=1; g[0]=0;
	#1 g[3]=0; g[2]=1; g[1]=1; g[0]=0;
	#1 g[3]=0; g[2]=1; g[1]=1; g[0]=1;
	#1 g[3]=0; g[2]=1; g[1]=0; g[0]=1;
	#1 g[3]=0; g[2]=1; g[1]=0; g[0]=0;
	#1 g[3]=1; g[2]=1; g[1]=0; g[0]=0;
	#1 g[3]=1; g[2]=1; g[1]=0; g[0]=1;
	#1 g[3]=1; g[2]=1; g[1]=1; g[0]=1;
	#1 g[3]=1; g[2]=1; g[1]=1; g[0]=0;
	#1 g[3]=1; g[2]=0; g[1]=1; g[0]=0;
	#1 g[3]=1; g[2]=0; g[1]=1; g[0]=1;
	#1 g[3]=1; g[2]=0; g[1]=0; g[0]=1;
	#1 g[3]=1; g[2]=0; g[1]=0; g[0]=0;
	end
	initial
	$monitor($time, "b3 = %d, b2 = %d, b1 = %d, b0 = %d, g[3] = %d, g[2] = %d, g[1] = %d, g[0] = %d",b3,b2,b1,b0,g[3],g[2],g[1],g[0]);
	initial
	#30 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1,gb_94test);
	end
	endmodule

G-B(GL) EDA

DESIGN CODE	TEST BENCH

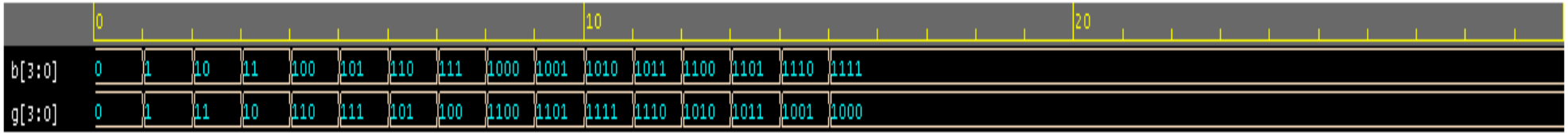
module gb_94(output reg[3:0]b,input[3:0]g);
  buf g1(b[3],g[3]);
  xor g2(b[2],g[3],g[2]);
  xor g3(b[1],g[3],g[2],g[1]);
  xor g4(b[0],g[3],g[2],g[1],g[0]);
endmodule
	module gb_94test;
	  reg [3:0]g;
	  wire [3:0]b;
	  gb_94 u1(b,g);
	  initial
	  begin
	  #0 g[3]=0; g[2]=0; g[1]=0; g[0]=0;
	  #1 g[3]=0; g[2]=0; g[1]=0; g[0]=1;
	  #1 g[3]=0; g[2]=0; g[1]=1; g[0]=1;
	  #1 g[3]=0; g[2]=0; g[1]=1; g[0]=0;
	  #1 g[3]=0; g[2]=1; g[1]=1; g[0]=0;
	  #1 g[3]=0; g[2]=1; g[1]=1; g[0]=1;
	  #1 g[3]=0; g[2]=1; g[1]=0; g[0]=1;
	  #1 g[3]=0; g[2]=1; g[1]=0; g[0]=0;
	  #1 g[3]=1; g[2]=1; g[1]=0; g[0]=0;
	  #1 g[3]=1; g[2]=1; g[1]=0; g[0]=1;
	  #1 g[3]=1; g[2]=1; g[1]=1; g[0]=1;
	  #1 g[3]=1; g[2]=1; g[1]=1; g[0]=0;
	  #1 g[3]=1; g[2]=0; g[1]=1; g[0]=0;
	  #1 g[3]=1; g[2]=0; g[1]=1; g[0]=1;
	  #1 g[3]=1; g[2]=0; g[1]=0; g[0]=1;
	  #1 g[3]=1; g[2]=0; g[1]=0; g[0]=0;
	  end
	  initial
	  $monitor($time, "b[3] = %d, b[2] = %d, b[1] = %d, b[0] = %d, g[3] = %d, g[2] = %d, g[1] = %d, g[0] = %d",b[3],b[2],b[1],b[0],g[3],g[2],g[1],g[0]);
	  initial
	  #30 $stop;
	  initial
	  begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1,gb_94test);
	  end
	endmodule

G-B(TASK) EDA

DESIGN CODE	TEST BENCH

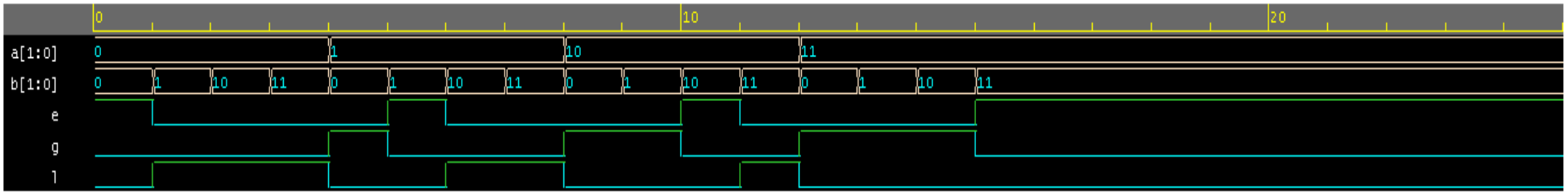
module gb_94(output reg[3:0]b,input[3:0]g);
always@(*)
begin
gbtsk_94(g,b);
end
task gbtsk_94;
input[3:0]g;
output reg[3:0]b;
begin
b[3]=g[3];
b[2]=g[3]^g[2];
b[1]=g[3]^g[2]^g[1];
b[0]=g[3]^g[2]^g[1]^g[0];
end
endtask
endmodule
	module gb_94test;
	reg [3:0]g;
	wire [3:0]b;
	gb_94 u1(b,g);
	initial
	begin
	#0 g[3]=0; g[2]=0; g[1]=0; g[0]=0;
	#1 g[3]=0; g[2]=0; g[1]=0; g[0]=1;
	#1 g[3]=0; g[2]=0; g[1]=1; g[0]=1;
	#1 g[3]=0; g[2]=0; g[1]=1; g[0]=0;
	#1 g[3]=0; g[2]=1; g[1]=1; g[0]=0;
	#1 g[3]=0; g[2]=1; g[1]=1; g[0]=1;
	#1 g[3]=0; g[2]=1; g[1]=0; g[0]=1;
	#1 g[3]=0; g[2]=1; g[1]=0; g[0]=0;
	#1 g[3]=1; g[2]=1; g[1]=0; g[0]=0;
	#1 g[3]=1; g[2]=1; g[1]=0; g[0]=1;
	#1 g[3]=1; g[2]=1; g[1]=1; g[0]=1;
	#1 g[3]=1; g[2]=1; g[1]=1; g[0]=0;
	#1 g[3]=1; g[2]=0; g[1]=1; g[0]=0;
	#1 g[3]=1; g[2]=0; g[1]=1; g[0]=1;
	#1 g[3]=1; g[2]=0; g[1]=0; g[0]=1;
	#1 g[3]=1; g[2]=0; g[1]=0; g[0]=0;
	end
	initial
	$monitor($time, "b[3] = %d, b[2] = %d, b[1] = %d, b[0] = %d, g[3] = %d, g[2] = %d, g[1] = %d, g[0] = %d",b[3],b[2],b[1],b[0],g[3],g[2],g[1],g[0]);
	initial
	#30 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1,gb_94test);
	end
	endmodule

B-G EDA OUTPUT

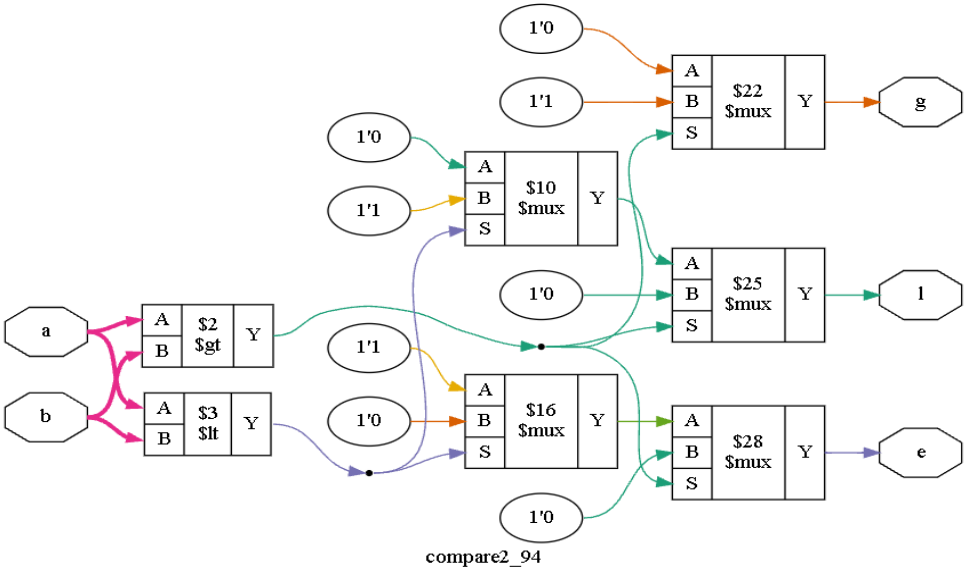



```
//
// Verilog description for cell bg_94,
// Mon Oct 18 11:41:10 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module bg_94 ( g, b ) ;

output [3:0]g ;
input [3:0]b ;

wire \b_int(3) ;
wire [3:1]b_int;
wire nx42825z1, nx41828z1, nx40831z1;



OBUF \g_obuf(0) (.O (g[0]), .I (nx40831z1)) ;
OBUF \g_obuf(1) (.O (g[1]), .I (nx41828z1)) ;
OBUF \g_obuf(2) (.O (g[2]), .I (nx42825z1)) ;
OBUF \g_obuf(3) (.O (g[3]), .I (\b_int(3) )) ;
IBUF \b_ibuf(0) (.O (b_int[1]), .I (b[0])) ;
IBUF \b_ibuf(1) (.O (b_int[2]), .I (b[1])) ;
IBUF \b_ibuf(2) (.O (b_int[3]), .I (b[2])) ;
IBUF \b_ibuf(3) (.O (\b_int(3) ), .I (b[3])) ;
LUT2 ix42825z1320 (.O (nx42825z1), .I0 (\b_int(3) ), .I1 (b_int[3])) ;
defparam ix42825z1320.INIT = 4'h6;
(* HLUTNM = "LUT62_1_1" *)
LUT2 ix41828z1320 (.O (nx41828z1), .I0 (b_int[3]), .I1 (b_int[2])) ;
defparam ix41828z1320.INIT = 4'h6;
(* HLUTNM = "LUT62_1_1" *)
LUT2 ix40831z1320 (.O (nx40831z1), .I0 (b_int[2]), .I1 (b_int[1])) ;
defparam ix40831z1320.INIT = 4'h6;
endmodule
```

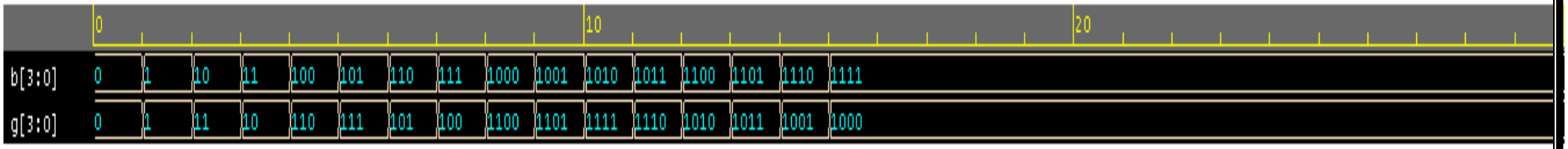
COMPARATOR_2B EDA OUTPUT




```
//
// Verilog description for cell compare2_94,
// Mon Oct 18 11:44:55 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module compare2_94 ( l, e, g, a, b ) ;

output l ;
output e ;
output g ;
input [1:0]a ;
input [1:0]b ;

wire [1:0]a_int;
wire [1:0]b_int;
wire nx23041z1, nx34068z1, nx15894z1;



OBUF g_obuf (.O (g), .I (nx15894z1)) ;
OBUF e_obuf (.O (e), .I (nx34068z1)) ;
OBUF l_obuf (.O (l), .I (nx23041z1)) ;
IBUF \b_ibuf(0) (.O (b_int[0]), .I (b[0])) ;
IBUF \b_ibuf(1) (.O (b_int[1]), .I (b[1])) ;
IBUF \a_ibuf(0) (.O (a_int[0]), .I (a[0])) ;
IBUF \a_ibuf(1) (.O (a_int[1]), .I (a[1])) ;
(* HLUTNM = "LUT62_1_1" *)
LUT4 ix23041z30322 (.O (nx23041z1), .I0 (a_int[1]), .I1 (a_int[0]), .I2 (
b_int[1]), .I3 (b_int[0])) ;
defparam ix23041z30322.INIT = 16'h7150;
LUT4 ix34068z35139 (.O (nx34068z1), .I0 (a_int[1]), .I1 (a_int[0]), .I2 (
b_int[1]), .I3 (b_int[0])) ;
defparam ix34068z35139.INIT = 16'h8421;
(* HLUTNM = "LUT62_1_1" *)
LUT4 ix15894z4016 (.O (nx15894z1), .I0 (a_int[1]), .I1 (a_int[0]), .I2 (
b_int[1]), .I3 (b_int[0])) ;
defparam ix15894z4016.INIT = 16'h0A8E;
endmodule
```



G-B EDA




```
//
// Verilog description for cell gb_94,
// Mon Oct 18 11:49:38 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module gb_94 ( b, g ) ;

output [3:0]b ;
input [3:0]g ;

wire \b_1_0(1) , \g_int(3) ;
wire [3:1]g_int;
wire nx60132z1, nx52153z1, nx58138z1;



(* IS_INTERNAL_DONT_TOUCH = "true" *)
INV \ix10_b(1) (.O (\b_1_0(1) ), .I (nx52153z1)) ;
OBUF \b_obuf(0) (.O (b[0]), .I (nx58138z1)) ;
OBUF \b_obuf(1) (.O (b[1]), .I (\b_1_0(1) )) ;
OBUF \b_obuf(2) (.O (b[2]), .I (nx60132z1)) ;
OBUF \b_obuf(3) (.O (b[3]), .I (\g_int(3) )) ;
IBUF \g_ibuf(0) (.O (g_int[1]), .I (g[0])) ;
IBUF \g_ibuf(1) (.O (g_int[2]), .I (g[1])) ;
IBUF \g_ibuf(2) (.O (g_int[3]), .I (g[2])) ;
IBUF \g_ibuf(3) (.O (\g_int(3) ), .I (g[3])) ;
LUT2 ix60132z1320 (.O (nx60132z1), .I0 (\g_int(3) ), .I1 (g_int[3])) ;
defparam ix60132z1320.INIT = 4'h6;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix52153z1419 (.O (nx52153z1), .I0 (\g_int(3) ), .I1 (g_int[3]), .I2 (
g_int[2])) ;
defparam ix52153z1419.INIT = 8'h69;
(* HLUTNM = "LUT62_1_1" *)
LUT4 ix58138z28344 (.O (nx58138z1), .I0 (\g_int(3) ), .I1 (g_int[3]), .I2 (
g_int[2]), .I3 (g_int[1])) ;
defparam ix58138z28344.INIT = 16'h6996;
endmodule
```

Date:
16-09-2021

Experiment No. 6

Flip flops

Aim: To implement flip flops (SR, D, JK, T, MS-JK) using verilog, verify the design using Testbench and perform functional simulation and to synthesize

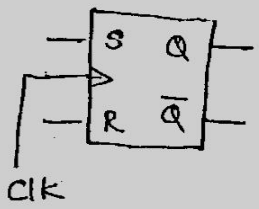
Requirement: EDA playground Tool

Theory:

* SR flip flop:

The SR flip flop is a 1-bit memory bistable device having two inputs, i.e., SET and RESET. The reset input is used to get the flip flop back to its original state from the current state with an output Q.

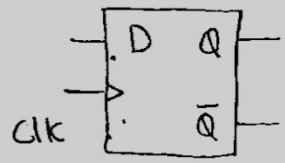
S	R	Q
0	0	No change
0	1	Set to 0
1	0	Set to 1
1	1	Invalid



* D flip flop:

In D flip flop, the single input 'D' is referred to as the data input. when the data input is set to 1, the flip flop would be set, and when it is set to 0 the flip flop would change and become reset.

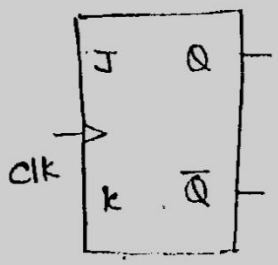
D	Q
0	Set to 0
1	Set to 1



* Jk flip flop:

The Jk flip flop has 'J' and 'k' flip flop inputs, it works in the same way as SR, the only difference is that, when SR is set to 1, the o/p is invalid, but that is not the case of Jk

J	k	Q
0	0	No change
0	1	Set to 0
1	0	Set to 1
1	1	Toggle

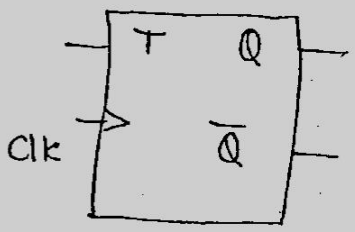


* T flip flop:

In 'T' flip flop, the term 'T' defines toggle. This flip flop works as a Toggle switch. The next output state is changed with the complement of present state output.

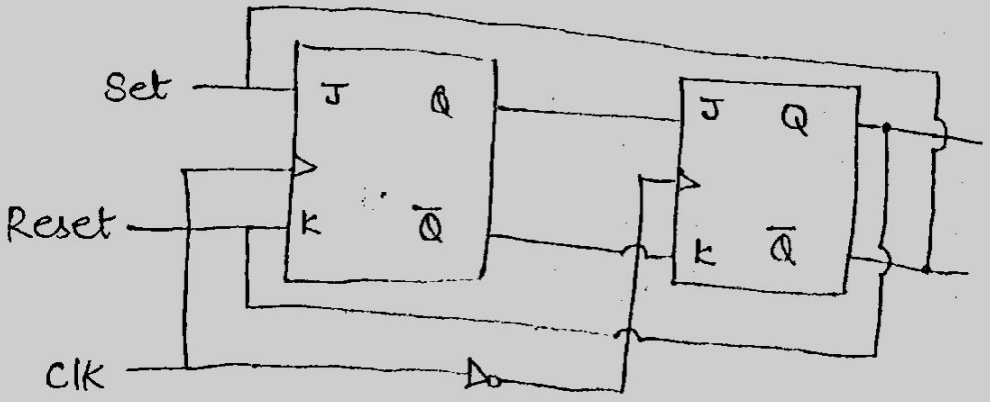
T	Q
0	1
1	0

toggle



* MS - Jk flip flop :

In this flip flop; two Jk flip flops are connected to each other, where one flip flop works as the master and the other as the slave.



D EDA

DESIGN CODE	TEST BENCH

module d_94(q,qbar,clk,d);
input clk,d;
output reg q,qbar;
 	
initial
begin
q=0; qbar=0;
end
 	
always@ (posedge clk)
begin
case(d)
1'b0:q=0;
1'b1:q=1;
default: q=0;
endcase
qbar=~q;
end
endmodule
	module d94_test;
	reg clk,d;
	wire q,qbar;
	d_94 g1(q,qbar,clk,d);
	initial
	begin
	#0 d=0;clk=0;
	#2 d=1;
	#3 d=0;
	#2 d=1;
	end
	initial
	$monitor ($time, " clk=%d, d=%d, q=%d qbar=%d",clk,d,q,qbar);
	initial
	#20 $stop;
	always
	begin
	#1 clk=~clk;
	end
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, d94_test);
	end
	endmodule

JK EDA

DESIGN CODE	TEST BENCH

module jk_94(q,qbar,clk,j,k);
input clk,j,k;
output reg q,qbar;
 	
initial
begin
q=0; qbar=0;
end
 	
always@ (posedge clk)
begin
case({j,k})
2'b00:q=q;
2'b01:q=0;
2'b10:q=1;
2'b11:q=~q;
default: q=0;
endcase
qbar=~q;
end
endmodule
	module jk94_test;
	reg clk,j,k;
	wire q,qbar;
	jk_94 g1(q,qbar,clk,j,k);
	initial
	begin
	#0 {j,k}=2'b00;clk=0;
	#2 {j,k}=2'b01;
	#3 {j,k}=2'b10;
	#2 {j,k}=2'b11;
	end
	initial
	$monitor ($time, " clk=%d, j=%d, k=%d, q=%d qbar=%d",clk,j,k,q,qbar);
	initial
	#20 $stop;
	always
	begin
	#1 clk=~clk;
	end
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, jk94_test);
	end
	endmodule

MS_JK EDA

DESIGN CODE	TEST BENCH

module msjk_94(q,qb,j,k,clk);
  input j,k,clk;
  output q,qb;
 	
  wire qi,qbi,clkb;
  assign clkb=~clk;
 	
  jk_94 g1(qi,qbi,j,k,clk);
  jk_94 g2(q,qb,qi,qbi,clkb);
endmodule
 	
module jk_94(qb,q,j,k,clk);
  input j,k,clk;
  output reg q,qb;
  initial
  begin
  q=0;
  qb=0;
  end
  always @(negedge clk)
  begin
  case({j,k})
  2'b00:begin q=q; qb=qb; end
  2'b01:begin q=0;qb=1; end
  2'b10:begin q=1;qb=0; end
  2'b11:begin q=~q;qb=~q; end
  endcase
  end
endmodule
	module msjk_94test;
	reg j,k, clk;
	wire q, qb;
	
	  msjk_94 g1(q,qb,j,k,clk);
	initial
	  begin
	  #0 {j,k}=2'b00;clk=0;
	  #2 {j,k}=2'b01;
	  #4 {j,k}=2'b10;
	  #8 {j,k}=2'b11;
	  end
	
	  initial
	  $monitor($time,"clk = %b, j = %b, k = %b, q = %b, qb = %b", clk, j, k, q, qb);
	
	initial
	  #30 $stop;
	
	initial begin
	 $dumpfile("dump.vcd");
	  $dumpvars(1, msjk_94test);
	end
	  always
	  #1 clk=~clk;
	endmodule

SR EDA

DESIGN CODE	TEST BENCH

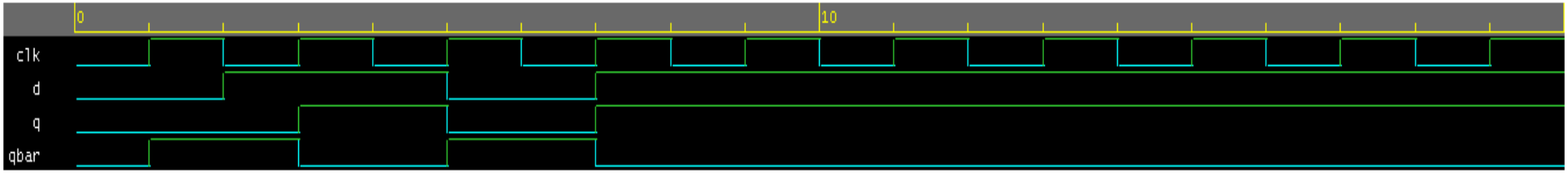
module sr_94(q,qbar,clk,s,r);
input clk,s,r;
output reg q,qbar;
 	
initial
begin
q=0; qbar=0;
end
 	
always@ (posedge clk)
begin
case({s,r})
2'b00:q=q;
2'b01:q=0;
2'b10:q=1;
2'b11:q=1'bx;
default: q=0;
endcase
qbar=~q;
end
endmodule
	module sr94_test;
	reg clk,s,r;
	wire q,qbar;
	sr_94 g1(q,qbar,clk,s,r);
	initial
	begin
	#0 {s,r}=2'b00;clk=0;
	#2 {s,r}=2'b01;
	#3 {s,r}=2'b10;
	#2 {s,r}=2'b11;
	end
	initial
	$monitor ($time, " clk=%d, s=%d, r=%d, q=%d qbar=%d",clk,s,r,q,qbar);
	initial
	#15 $stop;
	always
	begin
	#1 clk=~clk;
	end
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, sr94_test);
	end
	endmodule

T EDA

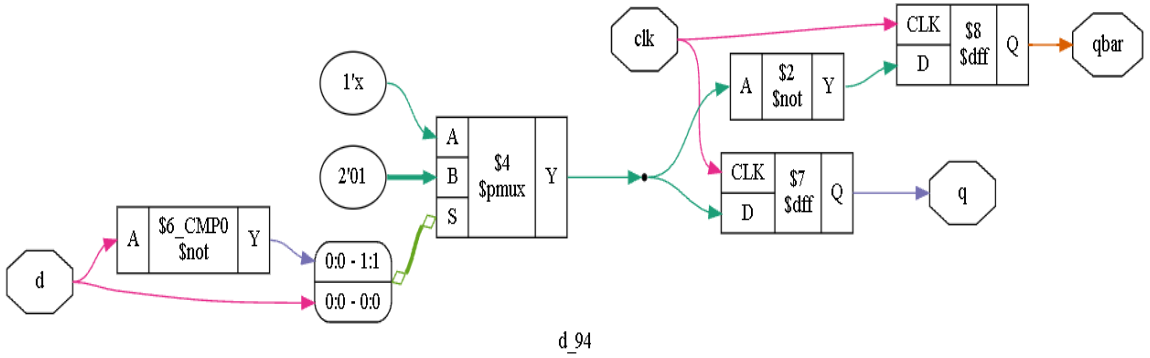
DESIGN CODE	TEST BENCH

module t_94(q,qbar,clk,t);
input clk,t;
output reg q,qbar;
 	
initial
begin
q=0; qbar=0;
end
 	
always@ (posedge clk)
begin
case(t)
1'b0:q=~q;
1'b1:q=~q;
default:q=0 ;
endcase
qbar=~q;
end
endmodule
	module t94_test;
	reg clk,t;
	wire q,qbar;
	t_94 g1(q,qbar,clk,t);
	initial
	begin
	#0 t=0;clk=0;
	#2 t=1;
	#3 t=0;
	#2 t=1;
	end
	initial
	$monitor ($time, " clk=%d, t=%d, q=%d qbar=%d",clk,t,q,qbar);
	initial
	#20 $stop;
	always
	begin
	#1 clk=~clk;
	end
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, t94_test);
	end
	endmodule

D EDA OUTPUT

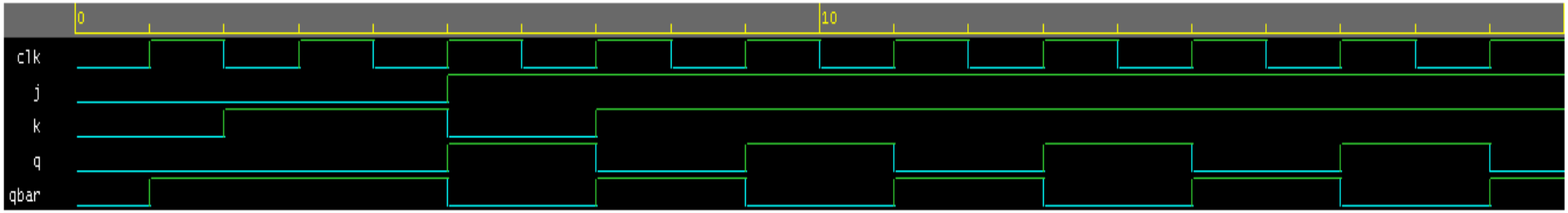



```
//
// Verilog description for cell d_94,
// Mon Oct 18 12:41:06 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module d_94 ( q, qbar, clk, d ) ;

output q ;
output qbar ;
input clk ;
input d ;

wire clk_int;
wire d_int, not_d, nx14466z1, q_1_0, nx86, qbar_1_0;



OBUF qbar_obuf (.O (qbar), .I (qbar_1_0)) ;
OBUF q_obuf (.O (q), .I (q_1_0)) ;
IBUF d_ibuf (.O (d_int), .I (d)) ;
INV ix14466z1315 (.O (not_d), .I (d_int)) ;
VCC ps_vcc (.P (nx14466z1)) ;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
FDRE reg_q (.Q (q_1_0), .C (clk_int), .CE (nx14466z1), .D (nx14466z1), .R (
not_d)) ;
GND ps_gnd (.G (nx86)) ;
FDRE reg_qbar (.Q (qbar_1_0), .C (clk_int), .CE (nx14466z1), .D (not_d), .R (
nx86)) ;
endmodule
```

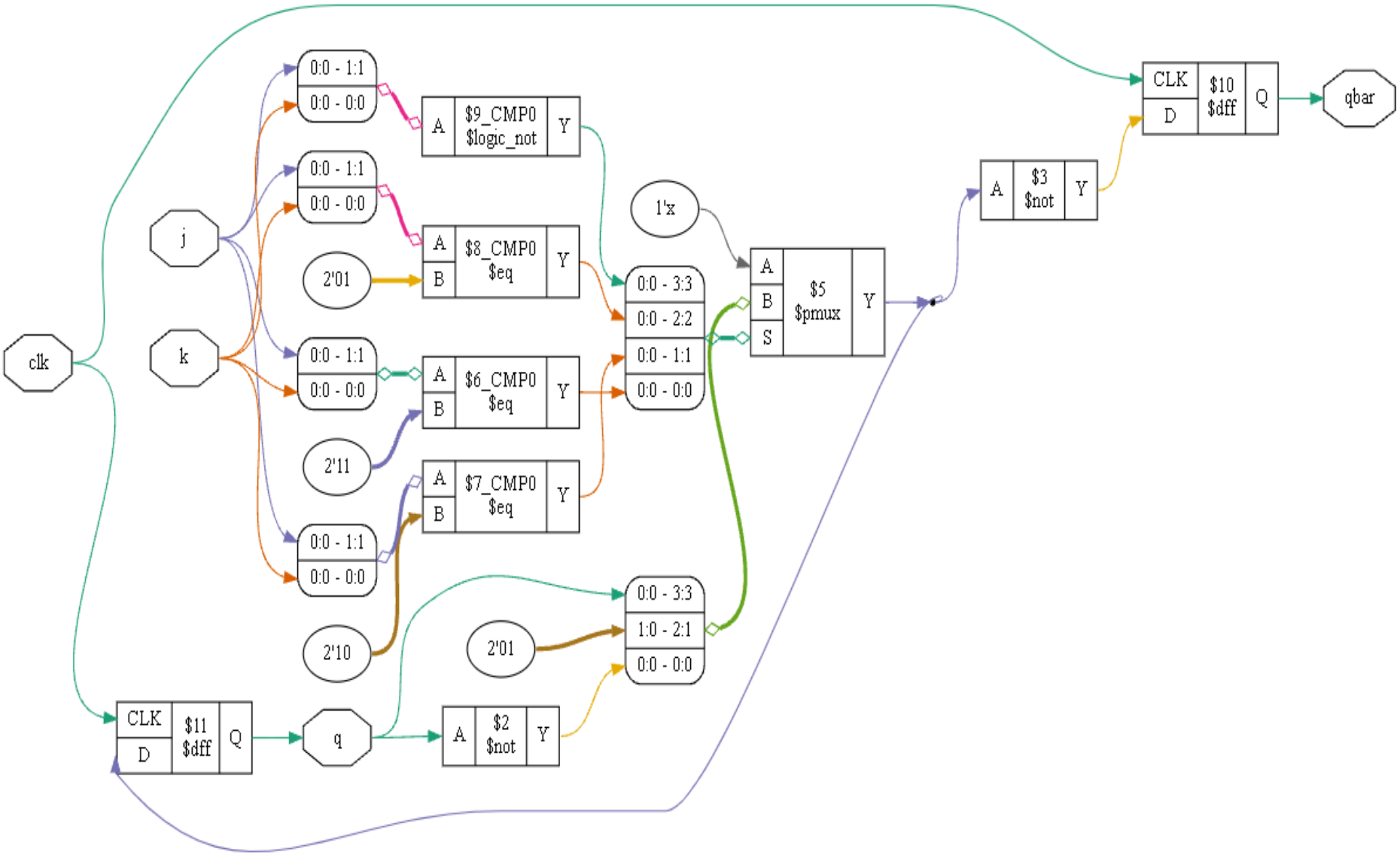


d_94

JK EDA OUTPUT

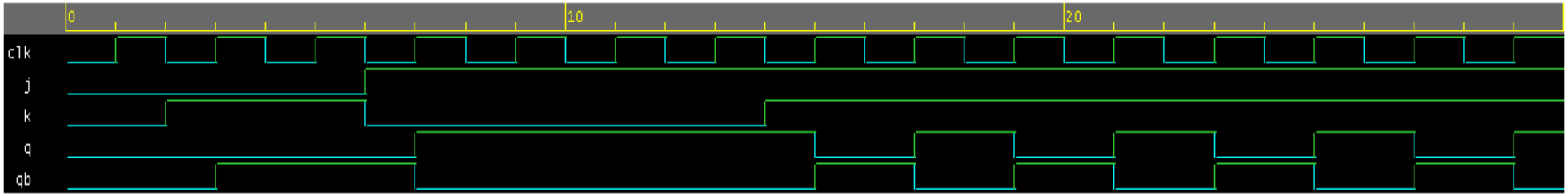



```
//
// Verilog description for cell jk_94,
// Mon Oct 18 12:49:03 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module jk_94 ( q, qbar, clk, j, k ) ;

output q ;
output qbar ;
input clk ;
input j ;
input k ;

wire clk_int;
wire j_int, k_int, nx14466z2, nx14466z1, nx15503z1, nx117, nx119, qbar_1_0,
q_1_0;



OBUF qbar_obuf (.O (qbar), .I (qbar_1_0)) ;
OBUF q_obuf (.O (q), .I (q_1_0)) ;
IBUF k_ibuf (.O (k_int), .I (k)) ;
IBUF j_ibuf (.O (j_int), .I (j)) ;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix14466z1394 (.O (nx14466z2), .I0 (q_1_0), .I1 (j_int), .I2 (k_int)) ;
defparam ix14466z1394.INIT = 8'h4F;
LUT2 ix14466z1328 (.O (nx14466z1), .I0 (j_int), .I1 (k_int)) ;
defparam ix14466z1328.INIT = 4'hE;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix15503z1491 (.O (nx15503z1), .I0 (q_1_0), .I1 (j_int), .I2 (k_int)) ;
defparam ix15503z1491.INIT = 8'hB1;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
VCC ps_vcc (.P (nx117)) ;
GND ps_gnd (.G (nx119)) ;
FDRE reg_qbar (.Q (qbar_1_0), .C (clk_int), .CE (nx117), .D (nx15503z1), .R (
nx119)) ;
FDRE reg_q (.Q (q_1_0), .C (clk_int), .CE (nx14466z1), .D (nx14466z2), .R (
nx119)) ;
endmodule
```


clk
j
k
0:0 - 1:1
0:0 - 0:0
A
$9_CMP0
$logic_not
Y
2'01
$8_CMP0
$eq
2'11
$6_CMP0
2'10
$7_CMP0
B
S
1'x
$5
$pmux
0:0 - 3:3
0:0 - 2:2
0:0 - 1:1
1:0 - 2:1
$3
$not
CLK
D
$10
$dff
Q
qbar
$11
q
$2

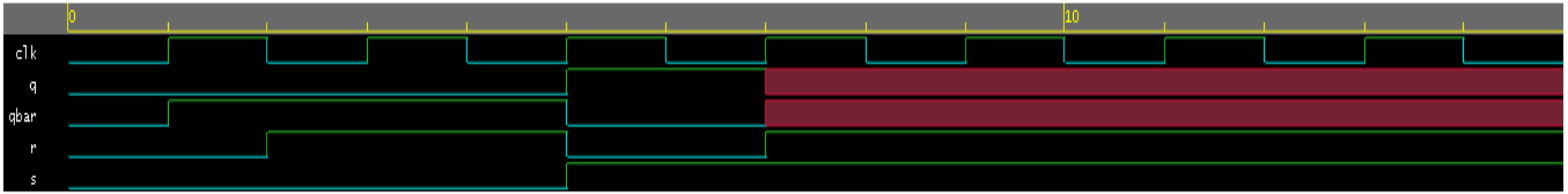
jk_94

MS_JK EDA OUTPUT

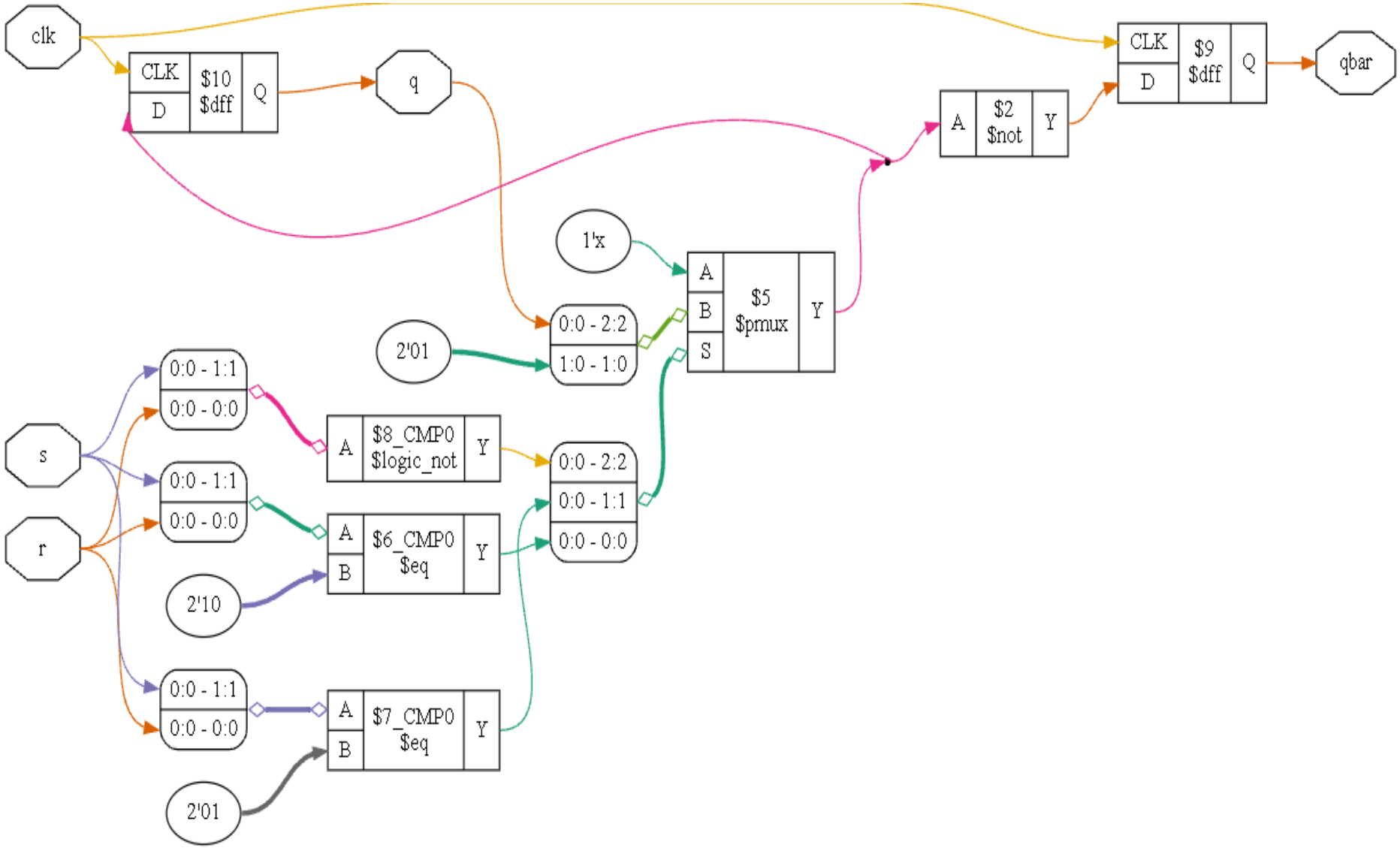



```
//
// Verilog description for cell msjk_94,
// Mon Oct 18 12:53:20 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module msjk_94 ( q, qb, j, k, clk ) ;

output q ;
output qb ;
input j ;
input k ;
input clk ;

wire qi, qbi, j_int, k_int;
wire clk_int;
wire nx27654z1, nx19520z2, nx19520z1, nx58045z1, nx62861z2, nx62861z1, nx209,
q_1_0, qb_1_0;



OBUF qb_obuf (.O (qb), .I (qb_1_0)) ;
OBUF q_obuf (.O (q), .I (q_1_0)) ;
IBUF k_ibuf (.O (k_int), .I (k)) ;
IBUF j_ibuf (.O (j_int), .I (j)) ;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix27654z1393 (.O (nx27654z1), .I0 (qb_1_0), .I1 (qi), .I2 (qbi)) ;
defparam ix27654z1393.INIT = 8'h4F;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix19520z1494 (.O (nx19520z2), .I0 (qb_1_0), .I1 (qi), .I2 (qbi)) ;
defparam ix19520z1494.INIT = 8'hB3;
(* HLUTNM = "LUT62_1_3" *)
LUT2 ix19520z1328 (.O (nx19520z1), .I0 (qi), .I1 (qbi)) ;
defparam ix19520z1328.INIT = 4'hE;
(* HLUTNM = "LUT62_1_2" *)
LUT3 ix58045z1393 (.O (nx58045z1), .I0 (qbi), .I1 (j_int), .I2 (k_int)) ;
defparam ix58045z1393.INIT = 8'h4F;
(* HLUTNM = "LUT62_1_2" *)
LUT3 ix62861z1494 (.O (nx62861z2), .I0 (qbi), .I1 (j_int), .I2 (k_int)) ;
defparam ix62861z1494.INIT = 8'hB3;
(* HLUTNM = "LUT62_1_3" *)
LUT2 ix62861z1328 (.O (nx62861z1), .I0 (j_int), .I1 (k_int)) ;
defparam ix62861z1328.INIT = 4'hE;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
GND ps_gnd (.G (nx209)) ;
FDRE g2_reg_qb (.Q (q_1_0), .C (clk_int), .CE (nx19520z1), .D (nx19520z2), .R (
nx209)) ;
FDRE g2_reg_q (.Q (qb_1_0), .C (clk_int), .CE (nx19520z1), .D (nx27654z1), .R (
nx209)) ;
FDRE g1_reg_qb (.Q (qi), .C (clk_int), .CE (nx62861z1), .D (nx62861z2), .R (
nx209)) ;
defparam g1_reg_qb.IS_C_INVERTED = 1'b1;
FDRE g1_reg_q (.Q (qbi), .C (clk_int), .CE (nx62861z1), .D (nx58045z1), .R (
nx209)) ;
defparam g1_reg_q.IS_C_INVERTED = 1'b1;
endmodule
```

SR EDA OUTPUT

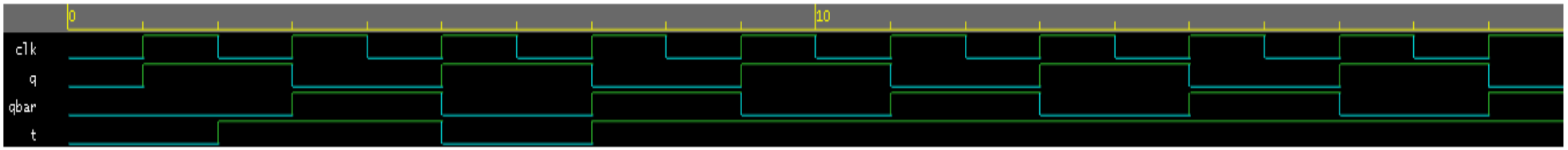



```
//
// Verilog description for cell sr_94,
// Mon Oct 18 12:57:25 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module sr_94 ( q, qbar, clk, s, r ) ;

output q ;
output qbar ;
input clk ;
input s ;
input r ;

wire q_1_0;
wire clk_int;
wire s_int, r_int, nx14466z2, nx14466z1, nx15503z1, nx14466z3, nx106,
qbar_1_0;



FDRE reg_q (.Q (q_1_0), .C (clk_int), .CE (nx14466z1), .D (nx14466z2), .R (
nx14466z3)) ;
OBUF qbar_obuf (.O (qbar), .I (qbar_1_0)) ;
OBUF q_obuf (.O (q), .I (q_1_0)) ;
IBUF r_ibuf (.O (r_int), .I (r)) ;
IBUF s_ibuf (.O (s_int), .I (s)) ;
VCC ps_vcc (.P (nx14466z2)) ;
LUT2 ix14466z1328 (.O (nx14466z1), .I0 (s_int), .I1 (r_int)) ;
defparam ix14466z1328.INIT = 4'hE;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix15503z1363 (.O (nx15503z1), .I0 (q_1_0), .I1 (s_int), .I2 (r_int)) ;
defparam ix15503z1363.INIT = 8'h31;
(* HLUTNM = "LUT62_1_1" *)
LUT2 ix14466z1319 (.O (nx14466z3), .I0 (s_int), .I1 (r_int)) ;
defparam ix14466z1319.INIT = 4'h4;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
GND ps_gnd (.G (nx106)) ;
FDRE reg_qbar (.Q (qbar_1_0), .C (clk_int), .CE (nx14466z2), .D (nx15503z1)
, .R (nx106)) ;
endmodule
```

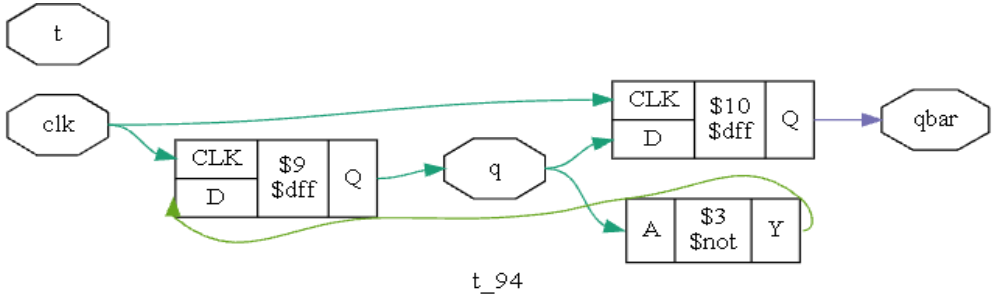

clk
CLK
D
$10
$dff
Q
q
CLK
D
$9
$dff
Q
qbar
A
$2
$not
Y
1'x
A
B
S
$5
$pmux
Y
0:0 - 2:2
1:0 - 1:0
2'01
0:0 - 1:1
0:0 - 0:0
s
r
A
$8_CMP0
$logic_not
Y
0:0 - 2:2
0:0 - 1:1
0:0 - 0:0
A
B
$6_CMP0
$eq
Y
2'10
0:0 - 1:1
0:0 - 0:0
A
B
$7_CMP0
$eq
Y
2'01

sr_94

T EDA OUTPUT




```
//
// Verilog description for cell t_94,
// Mon Oct 18 13:01:04 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module t_94 ( q, qbar, clk, t ) ;

output q ;
output qbar ;
input clk ;
input t ;

wire clk_int;
wire not_q, nx153, nx155, qbar_1_0, q_1_0;



OBUF qbar_obuf (.O (qbar), .I (qbar_1_0)) ;
OBUF q_obuf (.O (q), .I (q_1_0)) ;
INV ix14466z1315 (.O (not_q), .I (q_1_0)) ;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
VCC ps_vcc (.P (nx153)) ;
GND ps_gnd (.G (nx155)) ;
FDRE reg_qbar (.Q (qbar_1_0), .C (clk_int), .CE (nx153), .D (q_1_0), .R (
nx155)) ;
FDRE reg_q (.Q (q_1_0), .C (clk_int), .CE (nx153), .D (not_q), .R (nx155)) ;
endmodule
```



Date:
23-09-2021

Experiment No.7

Counters

Aim: To implement counters (Binary, BCD, Synchronous & Asynchronous) using verilog. verify the design using Testbench and perform functional simulation and to synthesize

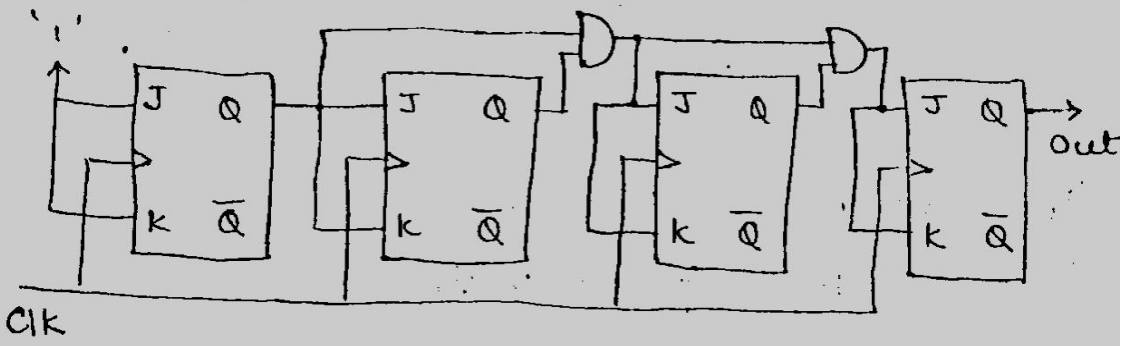
Requirement: EDA playground Tool.

Theory:

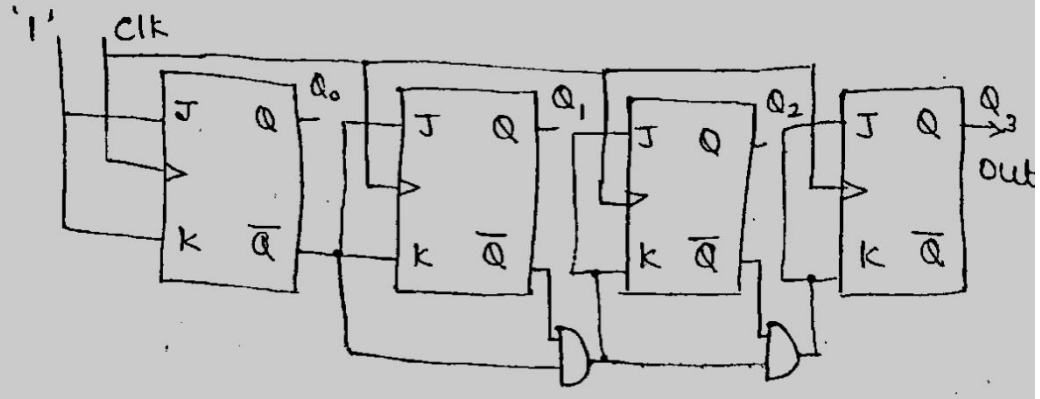
* Binary and BCD counters:

A Binary counter and BCD counter are very much the same in their working. the only difference is that a BCD is a decade counter, in the sense, it is designed to count only ten digits.

* Synchronous counter: (up counter)

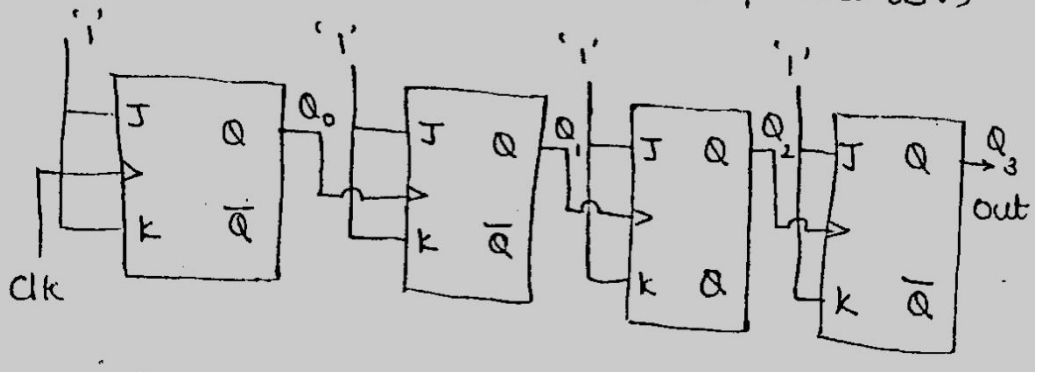


* Synchronous counter : (down counter)

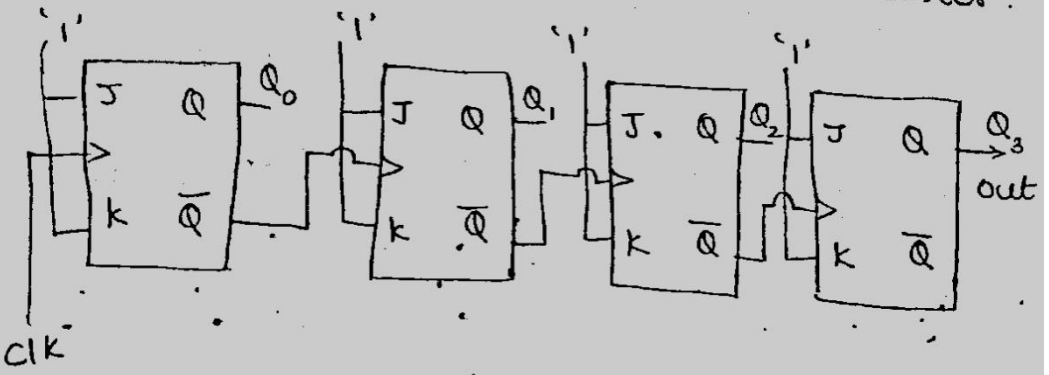


In synchronous counter, same clock is given to all flip flops

* Asynchronous counter : (up counter)



* Asynchronous counter : (down counter



In asynchronous counter, different clock is given to each flip flop

ASYNCOUNT(DOWN) EDA

DESIGN CODE	TEST BENCH

module Asyncount_94(q,qb,clk);
  input clk;
  output [3:0]q,qb;
  jkff g1(q[0],qb[0],1,1,clk);
  jkff g2(q[1],qb[1],1,1,q[0]);
  jkff g3(q[2],qb[2],1,1,q[1]);
  jkff g4(q[3],qb[3],1,1,q[2]);
endmodule
 	
module jkff(q,qb,j,k,clk);
  input j,k,clk;
  output reg q,qb;
 	
  initial
  begin
  q=0; qb=1;
  end
 	
  always@ (posedge clk)
  begin
  qb=~q;
  case({j,k})
  2'b00:begin q=q;qb=qb;end
  2'b01:begin q=0;qb=1;end
  2'b10:begin q=1;qb=0;end
  2'b11:begin q=~q;qb=~qb;end
  endcase
  end
endmodule
	module Asyncount_94test;
	  reg clk;
	  wire[3:0]q,qb;
	  Asyncount_94 g5(q,qb,clk);
	  initial
	  begin
	  #0 clk=1;
	  end
	  always
	  #1 clk=~clk;
	  initial
	  $monitor ($time, " clk=%b, q=%b, qb=%b",clk,q,qb);
	  initial
	  #32 $stop;
	
	  initial
	begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1, Asyncount_94test);
	  end
	endmodule

ASYNCOUNT(UP) EDA

DESIGN CODE	TEST BENCH

module Asyncount_94(q,qb,clk);
  input clk;
  output [3:0]q,qb;
  jkff g1(q[0],qb[0],1,1,clk);
  jkff g2(q[1],qb[1],1,1,qb[0]);
  jkff g3(q[2],qb[2],1,1,qb[1]);
  jkff g4(q[3],qb[3],1,1,qb[2]);
endmodule
 	
module jkff(q,qb,j,k,clk);
  input j,k,clk;
  output reg q,qb;
 	
  initial
  begin
  q=0; qb=1;
  end
 	
  always@ (posedge clk)
  begin
  qb=~q;
  case({j,k})
  2'b00:begin q=q;qb=qb;end
  2'b01:begin q=0;qb=1;end
  2'b10:begin q=1;qb=0;end
  2'b11:begin q=~q;qb=~qb;end
  endcase
  end
endmodule
	module Asyncount_94test;
	  reg clk;
	  wire[3:0]q,qb;
	  Asyncount_94 g5(q,qb,clk);
	  initial
	  begin
	  #0 clk=1;
	  end
	  always
	  #1 clk=~clk;
	  initial
	  $monitor ($time, " clk=%b, q=%b, qb=%b",clk,q,qb);
	  initial
	  #32 $stop;
	
	  initial
	begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1, Asyncount_94test);
	  end
	endmodule

BCDCOUNT(DOWN) EDA

DESIGN CODE	TEST BENCH

module counter_94(clk,count);
input clk;
output reg [3:0] count;
initial count = 4'b1001;
always@(posedge clk)
begin
count=count-1;
end
endmodule
	module counter_test94;
	reg clk;
	wire [3:0] count;
	initial
	begin
	clk=0;
	end
	counter_94 g1(clk,count);
	always #5 clk=~clk;
	initial
	#95 $finish;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, counter_test94);
	end
	endmodule

BCDCOUNT(UP) EDA

DESIGN CODE	TEST BENCH

module counter_94(clk,count);
input clk;
output reg [3:0] count;
initial count = 0;
always@(posedge clk)
begin
if(count<4'b1001)
count=count+1;
end
endmodule
	module counter_test94;
	reg clk;
	wire [3:0] count;
	initial
	begin
	clk=0;
	end
	counter_94 g1(clk,count);
	always #5 clk=~clk;
	initial
	#95 $finish;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, counter_test94);
	end
	endmodule

BCDCOUNT(UP_DOWN) EDA

DESIGN CODE	TEST BENCH

module counter_94(Clk,reset,m,Count);
input Clk,reset,m;
output [3 : 0] Count;
reg [3 : 0] Count = 0;
always @(posedge(Clk) or posedge(reset))
begin
if(reset == 1)
Count <= 0;
else
if(m == 1)
if(Count == 9)
Count <= 0;
else
Count <= Count + 1;
else
if(Count == 0)
Count <= 9;
else
Count <= Count - 1;
end
 	
endmodule
	module counter_94test;
	reg Clk;
	reg reset;
	reg m;
	wire [3:0] Count;
	
	counter_94 u1(Clk,reset,m,Count);
	initial Clk = 0;
	always #5 Clk = ~Clk;
	
	initial begin
	reset = 0;
	m = 1;
	#95;
	m = 0;
	end
	initial
	begin #185 $stop; end
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, counter_94test);
	end
	
	endmodule

BINARYCOUNT(DOWN) EDA

DESIGN CODE	TEST BENCH

module counter_94(clk,count);
input clk;
output reg [3:0] count;
initial count =4'b1111;
always@(posedge clk)
begin
count=count-1;
end
endmodule
	module counter_test94;
	reg clk;
	wire [3:0] count;
	initial
	begin
	clk=0;
	end
	counter_94 g1(clk,count);
	always #5 clk=~clk;
	initial
	#155 $finish;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, counter_test94);
	end
	endmodule

BINARYCOUNT(UP) EDA

DESIGN CODE	TEST BENCH

module counter_94(clk,count);
input clk;
output reg [3:0] count;
initial count = 0;
always@(posedge clk)
begin
count=count+1;
end
endmodule
	module counter_test94;
	reg clk;
	wire [3:0] count;
	initial
	begin
	clk=0;
	end
	counter_94 g1(clk,count);
	always #5 clk=~clk;
	initial
	#155 $finish;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, counter_test94);
	end
	endmodule

BINARYCOUNT(UP_DOWN) EDA

DESIGN CODE	TEST BENCH

module counter_94(clk,m,rst,count);
input clk,m,rst;
output reg [3:0] count;
always@(posedge clk or negedge rst)
begin
if(!rst)
count=0;
else if(m)
count=count+1;
else
count=count-1;
end
endmodule
	module counter_test94;
	reg clk, rst,m;
	wire [3:0] count;
	initial
	begin
	clk=1;
	rst=0;#10;
	rst=1;
	end
	initial
	begin
	m=1;
	#165 m=0;
	rst=1;
	#50 m=0;
	end
	counter_94 g1(clk,m,rst, count);
	always #5 clk=~clk;
	initial
	#310 $finish;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, counter_test94);
	end
	endmodule

RANCOUNT EDA

DESIGN CODE	TEST BENCH

module rancount_94(clk,q);
  input clk;
  output reg [2:0]q;
  initial
  begin
  q=3'b000;
  end
  always@(posedge clk)
  begin
  if (q==3'b000)
  q=3'b010;
  else if (q==3'b001)
  q=3'b011;
  else if (q==3'b010)
  q=3'b100;
  else if (q==3'b011)
  q=3'b101;
  else if (q==3'b100)
  q=3'b110;
  else if (q==3'b101)
  q=3'b111;
  else if (q==3'b110)
  q=3'b001;
  else if (q==3'b111)
  q=3'b000;
  end
endmodule
	module rancount_94test;
	  reg clk;
	  wire[2:0]q;
	  rancount_94 g5(clk,q);
	  initial
	  begin
	  #0 clk=0;
	  end
	  always
	  #1 clk=~clk;
	  initial
	  $monitor ($time, " clk=%b, q=%b",clk,q);
	  initial
	  #32 $stop;
	
	  initial
	begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1, rancount_94test);
	  end
	endmodule

SYNCOUNT(DOWN) EDA

DESIGN CODE	TEST BENCH

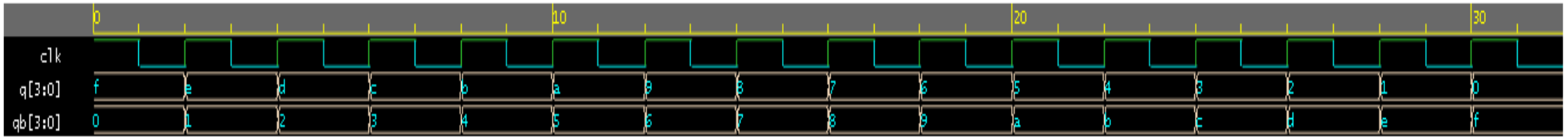
module syncount_94(q,qb,clk);
  input clk;
  output [3:0]q,qb;
  wire w1,w2;
  jkff g1(q[0],qb[0],1,1,clk);
  jkff g2(q[1],qb[1],qb[0],qb[0],clk);
  jkff g3(q[2],qb[2],w1,w1,clk);
  jkff g4(q[3],qb[3],w2,w2,clk);
  and g5(w1,qb[1],qb[0]);
  and g6(w2,w1,qb[2]);
endmodule
 	
module jkff(q,qb,j,k,clk);
  input j,k,clk;
  output reg q,qb;
 	
  initial
  begin
  q=0; qb=1;
  end
 	
  always@ (posedge clk)
  begin
  qb=~q;
  case({j,k})
  2'b00:begin q=q;qb=qb;end
  2'b01:begin q=0;qb=1;end
  2'b10:begin q=1;qb=0;end
  2'b11:begin q=~q;qb=~qb;end
  endcase
  end
endmodule
	module syncount_94test;
	  reg clk;
	  wire[3:0]q,qb;
	  syncount_94 g5(q,qb,clk);
	  initial
	  begin
	  #0 clk=1;
	  end
	  always
	  #1 clk=~clk;
	  initial
	  $monitor ($time, " clk=%b, q=%b, qb=%b",clk,q,qb);
	  initial
	  #32 $stop;
	
	  initial
	begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1, syncount_94test);
	  end
	endmodule

SYNCOUNT(UP) EDA

DESIGN CODE	TEST BENCH

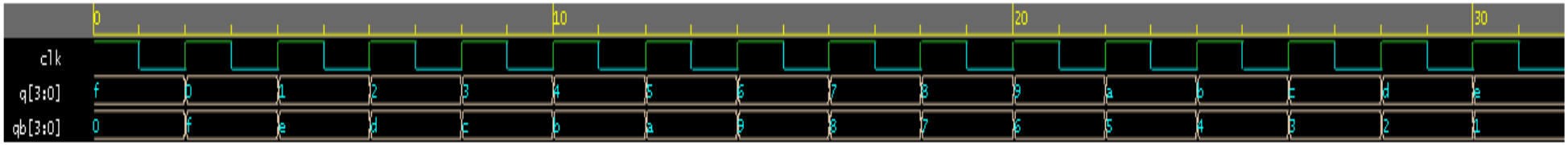
module syncount_94(q,qb,clk);
  input clk;
  output [3:0]q,qb;
  wire w1,w2;
  jkff g1(q[0],qb[0],1,1,clk);
  jkff g2(q[1],qb[1],q[0],q[0],clk);
  jkff g3(q[2],qb[2],w1,w1,clk);
  jkff g4(q[3],qb[3],w2,w2,clk);
  and g5(w1,q[1],q[0]);
  and g6(w2,w1,q[2]);
endmodule
 	
module jkff(q,qb,j,k,clk);
  input j,k,clk;
  output reg q,qb;
 	
  initial
  begin
  q=0; qb=1;
  end
 	
  always@ (posedge clk)
  begin
  qb=~q;
  case({j,k})
  2'b00:begin q=q;qb=qb;end
  2'b01:begin q=0;qb=1;end
  2'b10:begin q=1;qb=0;end
  2'b11:begin q=~q;qb=~qb;end
  endcase
  end
endmodule
	module syncount_94test;
	  reg clk;
	  wire[3:0]q,qb;
	  syncount_94 g5(q,qb,clk);
	  initial
	  begin
	  #0 clk=0;
	  end
	  always
	  #1 clk=~clk;
	  initial
	  $monitor ($time, " clk=%b, q=%b, qb=%b",clk,q,qb);
	  initial
	  #30 $stop;
	
	  initial
	begin
	  $dumpfile("dump.vcd");
	  $dumpvars(1, syncount_94test);
	  end
	endmodule

ASYNCOUNT(DOWN) EDA OUTPUT



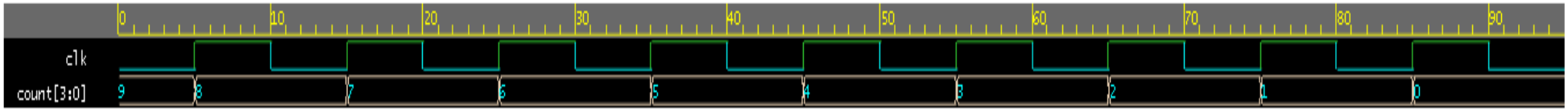

```
//
// Verilog description for cell Asyncount_94,
// Mon Oct 18 13:08:41 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module Asyncount_94 ( q, qb, clk ) ;

output [3:0]q ;
output [3:0]qb ;
input clk ;

wire clk_int;
wire g1_not_q, g2_not_q, g3_not_q, g4_not_q, \ripple_ena_q_1_0(2) ,
\ripple_ena_q_1_0(1) , \ripple_ena_q_1_0(0) ,
\ripple_ena_q_1_0(1)_and_ripple_ena_q_1_0(0)_out ,
\ripple_ena_q_1_0(2)_and_ripple_ena_q_1_0(1)_out_and_ripple_ena_q_1_0(0)_out ,
nx232, nx234;
wire [3:0]qb_1_0;
wire [3:0]q_1_0;



OBUF \qb_obuf(0) (.O (qb[0]), .I (qb_1_0[0])) ;
OBUF \qb_obuf(1) (.O (qb[1]), .I (qb_1_0[1])) ;
OBUF \qb_obuf(2) (.O (qb[2]), .I (qb_1_0[2])) ;
OBUF \qb_obuf(3) (.O (qb[3]), .I (qb_1_0[3])) ;
OBUF \q_obuf(0) (.O (q[0]), .I (q_1_0[0])) ;
OBUF \q_obuf(1) (.O (q[1]), .I (q_1_0[1])) ;
OBUF \q_obuf(2) (.O (q[2]), .I (q_1_0[2])) ;
OBUF \q_obuf(3) (.O (q[3]), .I (q_1_0[3])) ;
INV ix58045z1315 (.O (g1_not_q), .I (q_1_0[0])) ;
INV ix27654z1316 (.O (g2_not_q), .I (q_1_0[1])) ;
INV ix2737z1320 (.O (g3_not_q), .I (q_1_0[2])) ;
INV ix33128z1318 (.O (g4_not_q), .I (q_1_0[3])) ;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
INV ix33128z1317 (.O (\ripple_ena_q_1_0(2) ), .I (q_1_0[2])) ;
INV ix33128z1316 (.O (\ripple_ena_q_1_0(1) ), .I (q_1_0[1])) ;
INV ix27654z1315 (.O (\ripple_ena_q_1_0(0) ), .I (q_1_0[0])) ;
(* HLUTNM = "LUT62_1_1" *)
LUT2 ix2737z1316 (.O (\ripple_ena_q_1_0(1)_and_ripple_ena_q_1_0(0)_out ), .I0 (
\ripple_ena_q_1_0(1) ), .I1 (q_1_0[0])) ;
defparam ix2737z1316.INIT = 4'h2;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix33128z1378 (.O (
\ripple_ena_q_1_0(2)_and_ripple_ena_q_1_0(1)_out_and_ripple_ena_q_1_0(0)_out
), .I0 (q_1_0[0]), .I1 (\ripple_ena_q_1_0(1) ), .I2 (
\ripple_ena_q_1_0(2) )) ;
defparam ix33128z1378.INIT = 8'h40;
VCC ps_vcc (.P (nx232)) ;
GND ps_gnd (.G (nx234)) ;
FDRE g1_reg_qb (.Q (qb_1_0[0]), .C (clk_int), .CE (nx232), .D (q_1_0[0]), .R (
nx234)) ;
FDRE g1_reg_q (.Q (q_1_0[0]), .C (clk_int), .CE (nx232), .D (g1_not_q), .R (
nx234)) ;
FDRE g2_reg_qb (.Q (qb_1_0[1]), .C (clk_int), .CE (\ripple_ena_q_1_0(0) ),
.D (q_1_0[1]), .R (nx234)) ;
FDRE g2_reg_q (.Q (q_1_0[1]), .C (clk_int), .CE (\ripple_ena_q_1_0(0) ), .D (
g2_not_q), .R (nx234)) ;
FDRE g3_reg_qb (.Q (qb_1_0[2]), .C (clk_int), .CE (
\ripple_ena_q_1_0(1)_and_ripple_ena_q_1_0(0)_out ), .D (q_1_0[2]), .R (
nx234)) ;
FDRE g3_reg_q (.Q (q_1_0[2]), .C (clk_int), .CE (
\ripple_ena_q_1_0(1)_and_ripple_ena_q_1_0(0)_out ), .D (g3_not_q), .R (
nx234)) ;
FDRE g4_reg_qb (.Q (qb_1_0[3]), .C (clk_int), .CE (
\ripple_ena_q_1_0(2)_and_ripple_ena_q_1_0(1)_out_and_ripple_ena_q_1_0(0)_out
), .D (q_1_0[3]), .R (nx234)) ;
FDRE g4_reg_q (.Q (q_1_0[3]), .C (clk_int), .CE (
\ripple_ena_q_1_0(2)_and_ripple_ena_q_1_0(1)_out_and_ripple_ena_q_1_0(0)_out
), .D (g4_not_q), .R (nx234)) ;
endmodule
```

ASYNCOUNT(UP) EDA OUTPUT

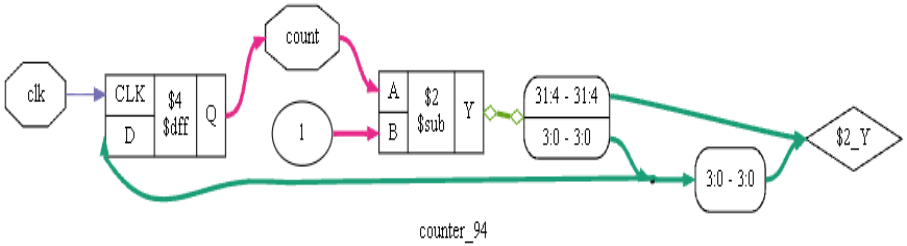



```
//
// Verilog description for cell Asyncount_94,
// Mon Oct 18 13:17:44 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module Asyncount_94 ( q, qb, clk ) ;

output [3:0]q ;
output [3:0]qb ;
input clk ;

wire clk_int;
wire g1_not_q, g2_not_q, g3_not_q, g4_not_q, \ripple_ena_qb_1_0(0) ,
\ripple_ena_qb_1_0(1)_and_ripple_ena_qb_1_0(0)_out ,
\ripple_ena_qb_1_0(2)_and_ripple_ena_qb_1_0(1)_out_and_ripple_ena_qb_1_0(0)_out ,
nx227, nx229;
wire [3:0]qb_1_0;
wire [3:0]q_1_0;



OBUF \qb_obuf(0) (.O (qb[0]), .I (qb_1_0[0])) ;
OBUF \qb_obuf(1) (.O (qb[1]), .I (qb_1_0[1])) ;
OBUF \qb_obuf(2) (.O (qb[2]), .I (qb_1_0[2])) ;
OBUF \qb_obuf(3) (.O (qb[3]), .I (qb_1_0[3])) ;
OBUF \q_obuf(0) (.O (q[0]), .I (q_1_0[0])) ;
OBUF \q_obuf(1) (.O (q[1]), .I (q_1_0[1])) ;
OBUF \q_obuf(2) (.O (q[2]), .I (q_1_0[2])) ;
OBUF \q_obuf(3) (.O (q[3]), .I (q_1_0[3])) ;
INV ix58045z1315 (.O (g1_not_q), .I (q_1_0[0])) ;
INV ix27654z1315 (.O (g2_not_q), .I (q_1_0[1])) ;
INV ix2737z1315 (.O (g3_not_q), .I (q_1_0[2])) ;
INV ix33128z1316 (.O (g4_not_q), .I (q_1_0[3])) ;
(* HLUTNM = "LUT62_1_1" *)
LUT2 i_enable_gate_1_1 (.O (\ripple_ena_qb_1_0(0) ), .I0 (q_1_0[0]), .I1 (
qb_1_0[0])) ;
defparam i_enable_gate_1_1.INIT = 4'h2;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
(* HLUTNM = "LUT62_1_1" *)
LUT4 ix23821z1826 (.O (\ripple_ena_qb_1_0(1)_and_ripple_ena_qb_1_0(0)_out )
, .I0 (q_1_0[0]), .I1 (qb_1_0[0]), .I2 (qb_1_0[1]), .I3 (q_1_0[1])) ;
defparam ix23821z1826.INIT = 16'h0200;
LUT6 ix33128z1314 (.O (
\ripple_ena_qb_1_0(2)_and_ripple_ena_qb_1_0(1)_out_and_ripple_ena_qb_1_0(0)_out
), .I0 (q_1_0[1]), .I1 (qb_1_0[1]), .I2 (q_1_0[0]), .I3 (qb_1_0[0]), .I4 (
qb_1_0[2]), .I5 (q_1_0[2])) ;
defparam ix33128z1314.INIT = 64'h0000002000000000;
VCC ps_vcc (.P (nx227)) ;
GND ps_gnd (.G (nx229)) ;
FDRE g1_reg_qb (.Q (qb_1_0[0]), .C (clk_int), .CE (nx227), .D (q_1_0[0]), .R (
nx229)) ;
FDRE g1_reg_q (.Q (q_1_0[0]), .C (clk_int), .CE (nx227), .D (g1_not_q), .R (
nx229)) ;
FDRE g2_reg_qb (.Q (qb_1_0[1]), .C (clk_int), .CE (\ripple_ena_qb_1_0(0) )
, .D (q_1_0[1]), .R (nx229)) ;
FDRE g2_reg_q (.Q (q_1_0[1]), .C (clk_int), .CE (\ripple_ena_qb_1_0(0) ), .D (
g2_not_q), .R (nx229)) ;
FDRE g3_reg_qb (.Q (qb_1_0[2]), .C (clk_int), .CE (
\ripple_ena_qb_1_0(1)_and_ripple_ena_qb_1_0(0)_out ), .D (q_1_0[2]), .R (
nx229)) ;
FDRE g3_reg_q (.Q (q_1_0[2]), .C (clk_int), .CE (
\ripple_ena_qb_1_0(1)_and_ripple_ena_qb_1_0(0)_out ), .D (g3_not_q), .R (
nx229)) ;
FDRE g4_reg_qb (.Q (qb_1_0[3]), .C (clk_int), .CE (
\ripple_ena_qb_1_0(2)_and_ripple_ena_qb_1_0(1)_out_and_ripple_ena_qb_1_0(0)_out
), .D (q_1_0[3]), .R (nx229)) ;
FDRE g4_reg_q (.Q (q_1_0[3]), .C (clk_int), .CE (
\ripple_ena_qb_1_0(2)_and_ripple_ena_qb_1_0(1)_out_and_ripple_ena_qb_1_0(0)_out
), .D (g4_not_q), .R (nx229)) ;
endmodule
```

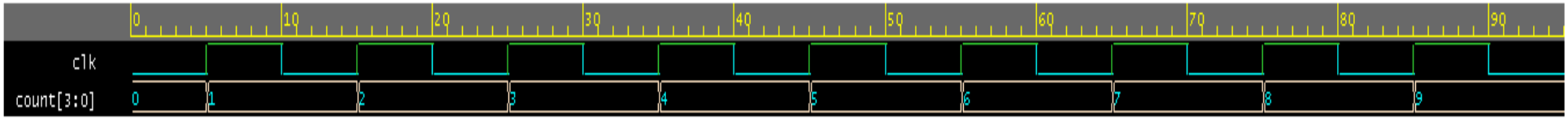
BCDCOUNT(DOWN) EDA OUTPUT



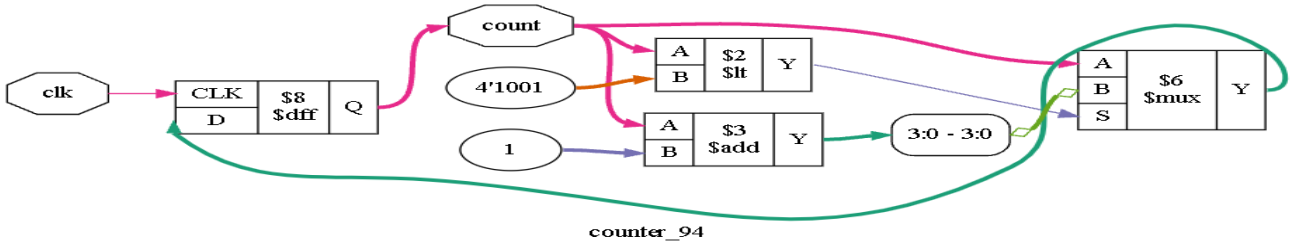

```
// Verilog description for cell counter_94,
// Mon Oct 18 13:24:03 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module counter_94 ( clk, count ) ;

input clk ;
output [3:0]count ;

wire clk_int;
wire n16265z1, n14271z1, n13274z1, n15268z1, n104, n106;
wire [3:0]count_1_0;



OBUF \count_obuf(0) (.O (count[0]), .I (count_1_0[0])) ;
OBUF \count_obuf(1) (.O (count[1]), .I (count_1_0[1])) ;
OBUF \count_obuf(2) (.O (count[2]), .I (count_1_0[2])) ;
OBUF \count_obuf(3) (.O (count[3]), .I (count_1_0[3])) ;
INV i16265z1315 (.O (n16265z1), .I (count_1_0[0])) ;
(* HLUTNM = "LUT62_1_1" *)
LUT3 i14271z1483 (.O (n14271z1), .I0 (count_1_0[2]), .I1 (count_1_0[1]), .I2 (
count_1_0[0])) ;
defparam i14271z1483.INIT = 8'hA9;
(* HLUTNM = "LUT62_1_1" *)
LUT4 i13274z45003 (.O (n13274z1), .I0 (count_1_0[3]), .I1 (count_1_0[2]),
.I2 (count_1_0[1]), .I3 (count_1_0[0])) ;
defparam i13274z45003.INIT = 16'hAAA9;
LUT2 i15268z1323 (.O (n15268z1), .I0 (count_1_0[1]), .I1 (count_1_0[0])) ;
defparam i15268z1323.INIT = 4'h9;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
VCC ps_vcc (.P (n104)) ;
GND ps_gnd (.G (n106)) ;
(* no_enable_dff = "TRUE" *)
FDRE \modgen_counter_count_reg_q(3) (.Q (count_1_0[3]), .C (clk_int), .CE (
n104), .D (n13274z1), .R (n106)) ;
(* no_enable_dff = "TRUE" *)
FDRE \modgen_counter_count_reg_q(2) (.Q (count_1_0[2]), .C (clk_int), .CE (
n104), .D (n14271z1), .R (n106)) ;
(* no_enable_dff = "TRUE" *)
FDRE \modgen_counter_count_reg_q(1) (.Q (count_1_0[1]), .C (clk_int), .CE (
n104), .D (n15268z1), .R (n106)) ;
(* no_enable_dff = "TRUE" *)
FDRE \modgen_counter_count_reg_q(0) (.Q (count_1_0[0]), .C (clk_int), .CE (
n104), .D (n16265z1), .R (n106)) ;
endmodule
```



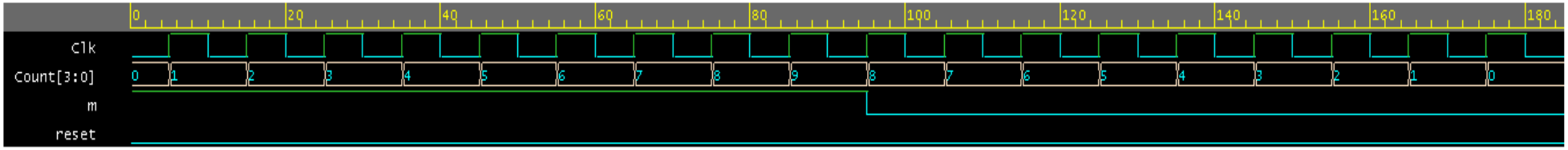
BCDCOUNT(UP) EDA OUTPUT



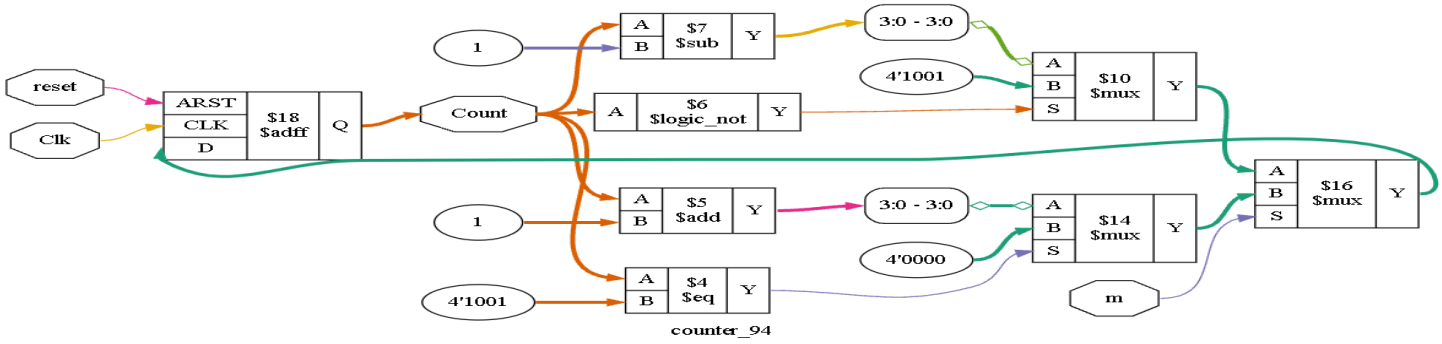

```
//
// Verilog description for cell counter_94,
// Mon Oct 18 13:30:02 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module counter_94 ( clk, count ) ;

input clk ;
output [3:0]count ;

wire clk_int;
wire nx16265z1, nx15268z1, nx14271z1, nx13274z2, nx13274z1, nx119;
wire [3:0]count_1_0;



OBUF \count_obuf(0) (.O (count[0]), .I (count_1_0[0])) ;
OBUF \count_obuf(1) (.O (count[1]), .I (count_1_0[1])) ;
OBUF \count_obuf(2) (.O (count[2]), .I (count_1_0[2])) ;
OBUF \count_obuf(3) (.O (count[3]), .I (count_1_0[3])) ;
INV ix16265z1315 (.O (nx16265z1), .I (count_1_0[0])) ;
(* HLUTNM = "LUT62_1_2" *)
LUT2 ix15268z1320 (.O (nx15268z1), .I0 (count_1_0[1]), .I1 (count_1_0[0])) ;
defparam ix15268z1320.INIT = 4'h6;
(* HLUTNM = "LUT62_1_2" *)
LUT3 ix14271z1420 (.O (nx14271z1), .I0 (count_1_0[2]), .I1 (count_1_0[1]), .I2 (
count_1_0[0])) ;
defparam ix14271z1420.INIT = 8'h6A;
(* HLUTNM = "LUT62_1_1" *)
LUT4 ix13274z28621 (.O (nx13274z2), .I0 (count_1_0[3]), .I1 (count_1_0[2]),
.I2 (count_1_0[1]), .I3 (count_1_0[0])) ;
defparam ix13274z28621.INIT = 16'h6AAA;
(* HLUTNM = "LUT62_1_1" *)
LUT4 ix13274z23161 (.O (nx13274z1), .I0 (count_1_0[3]), .I1 (count_1_0[2]),
.I2 (count_1_0[1]), .I3 (count_1_0[0])) ;
defparam ix13274z23161.INIT = 16'h5557;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
GND ps_gnd (.G (nx119)) ;
(* no_enable_dff = "TRUE" *)
FDRE \modgen_counter_count_reg_q(3) (.Q (count_1_0[3]), .C (clk_int), .CE (
nx13274z1), .D (nx13274z2), .R (nx119)) ;
(* no_enable_dff = "TRUE" *)
FDRE \modgen_counter_count_reg_q(2) (.Q (count_1_0[2]), .C (clk_int), .CE (
nx13274z1), .D (nx14271z1), .R (nx119)) ;
(* no_enable_dff = "TRUE" *)
FDRE \modgen_counter_count_reg_q(1) (.Q (count_1_0[1]), .C (clk_int), .CE (
nx13274z1), .D (nx15268z1), .R (nx119)) ;
(* no_enable_dff = "TRUE" *)
FDRE \modgen_counter_count_reg_q(0) (.Q (count_1_0[0]), .C (clk_int), .CE (
nx13274z1), .D (nx16265z1), .R (nx119)) ;
endmodule
```



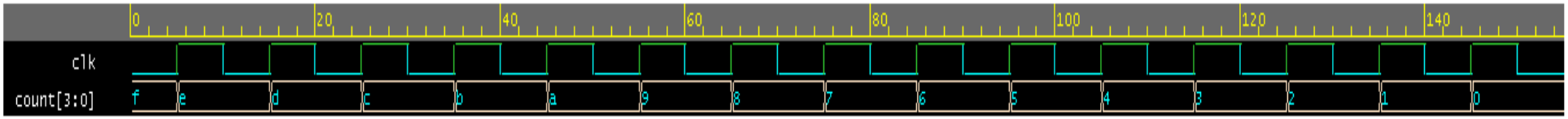
BCDCOUNT(UP_DOWN) EDA OUTPUT



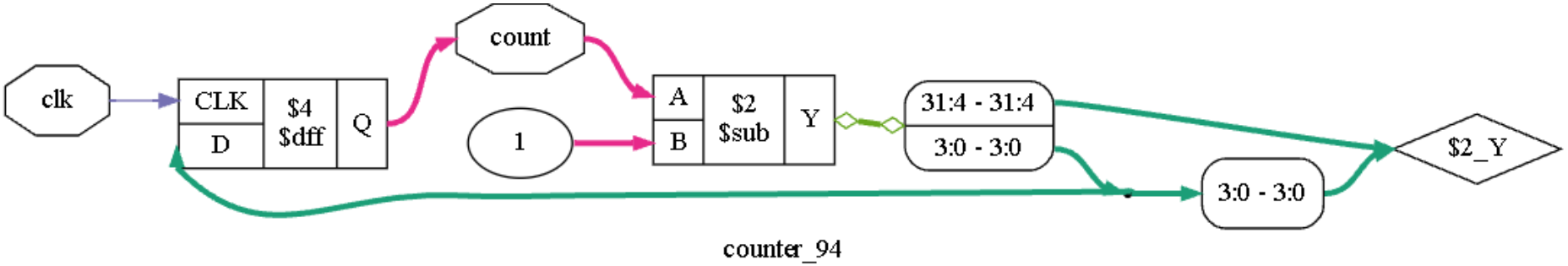

```
//
// Verilog description for cell counter_94,
// Mon Oct 18 13:34:38 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module counter_94 ( Clk, reset, m, Count ) ;

input Clk ;
input reset ;
input m ;
output [3:0]Count ;

wire Clk_int;
wire reset_int, m_int, nx48681z1, nx47684z1, nx46687z1, nx45690z1, nx146;
wire [3:0]Count_1_0;



OBUF \Count_obuf(0) (.O (Count[0]), .I (Count_1_0[0])) ;
OBUF \Count_obuf(1) (.O (Count[1]), .I (Count_1_0[1])) ;
OBUF \Count_obuf(2) (.O (Count[2]), .I (Count_1_0[2])) ;
OBUF \Count_obuf(3) (.O (Count[3]), .I (Count_1_0[3])) ;
IBUF m_ibuf (.O (m_int), .I (m)) ;
IBUF reset_ibuf (.O (reset_int), .I (reset)) ;
INV ix48681z1315 (.O (nx48681z1), .I (Count_1_0[0])) ;
LUT5 ix47684z62768 (.O (nx47684z1), .I0 (Count_1_0[3]), .I1 (Count_1_0[2]),
.I2 (Count_1_0[1]), .I3 (Count_1_0[0]), .I4 (m_int)) ;
defparam ix47684z62768.INIT = 32'h0DF0F00E;
(* HLUTNM = "LUT62_1_1" *)
LUT5 ix46687z53732 (.O (nx46687z1), .I0 (Count_1_0[3]), .I1 (Count_1_0[2]),
.I2 (Count_1_0[1]), .I3 (Count_1_0[0]), .I4 (m_int)) ;
defparam ix46687z53732.INIT = 32'h3CCCCCC2;
(* HLUTNM = "LUT62_1_1" *)
LUT5 ix45690z45003 (.O (nx45690z1), .I0 (Count_1_0[3]), .I1 (Count_1_0[2]),
.I2 (Count_1_0[1]), .I3 (Count_1_0[0]), .I4 (m_int)) ;
defparam ix45690z45003.INIT = 32'h68AAAAA9;
BUFGP Clk_ibuf (.O (Clk_int), .I (Clk)) ;
VCC ps_vcc (.P (nx146)) ;
(* no_enable_dff = "TRUE" *)
FDCE \modgen_counter_Count_reg_q(3) (.Q (Count_1_0[3]), .C (Clk_int), .CE (
nx146), .CLR (reset_int), .D (nx45690z1)) ;
(* no_enable_dff = "TRUE" *)
FDCE \modgen_counter_Count_reg_q(2) (.Q (Count_1_0[2]), .C (Clk_int), .CE (
nx146), .CLR (reset_int), .D (nx46687z1)) ;
(* no_enable_dff = "TRUE" *)
FDCE \modgen_counter_Count_reg_q(1) (.Q (Count_1_0[1]), .C (Clk_int), .CE (
nx146), .CLR (reset_int), .D (nx47684z1)) ;
(* no_enable_dff = "TRUE" *)
FDCE \modgen_counter_Count_reg_q(0) (.Q (Count_1_0[0]), .C (Clk_int), .CE (
nx146), .CLR (reset_int), .D (nx48681z1)) ;
endmodule
```



BINARYCOUNT(DOWN) EDA OUTPUT

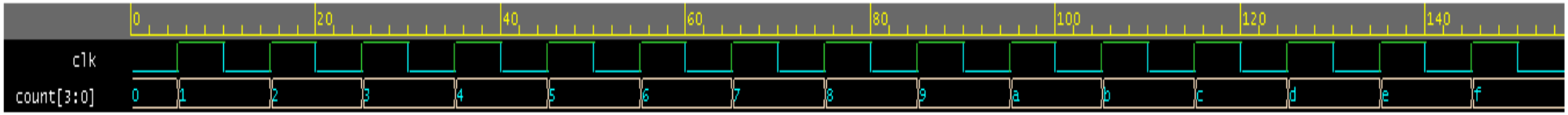



```
//
// Verilog description for cell counter_94,
// Mon Oct 18 13:38:19 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module counter_94 ( clk, count ) ;

input clk ;
output [3:0]count ;

wire clk_int;
wire nx16265z1, nx14271z1, nx13274z1, nx15268z1, nx104, nx106;
wire [3:0]count_1_0;



OBUF \count_obuf(0) (.O (count[0]), .I (count_1_0[0])) ;
OBUF \count_obuf(1) (.O (count[1]), .I (count_1_0[1])) ;
OBUF \count_obuf(2) (.O (count[2]), .I (count_1_0[2])) ;
OBUF \count_obuf(3) (.O (count[3]), .I (count_1_0[3])) ;
INV ix16265z1315 (.O (nx16265z1), .I (count_1_0[0])) ;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix14271z1483 (.O (nx14271z1), .I0 (count_1_0[2]), .I1 (count_1_0[1]), .I2 (
count_1_0[0])) ;
defparam ix14271z1483.INIT = 8'hA9;
(* HLUTNM = "LUT62_1_1" *)
LUT4 ix13274z45003 (.O (nx13274z1), .I0 (count_1_0[3]), .I1 (count_1_0[2]),
.I2 (count_1_0[1]), .I3 (count_1_0[0])) ;
defparam ix13274z45003.INIT = 16'hAAA9;
LUT2 ix15268z1323 (.O (nx15268z1), .I0 (count_1_0[1]), .I1 (count_1_0[0])) ;
defparam ix15268z1323.INIT = 4'h9;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
VCC ps_vcc (.P (nx104)) ;
GND ps_gnd (.G (nx106)) ;
(* no_enable_dff = "TRUE" *)
FDRE \modgen_counter_count_reg_q(3) (.Q (count_1_0[3]), .C (clk_int), .CE (
nx104), .D (nx13274z1), .R (nx106)) ;
(* no_enable_dff = "TRUE" *)
FDRE \modgen_counter_count_reg_q(2) (.Q (count_1_0[2]), .C (clk_int), .CE (
nx104), .D (nx14271z1), .R (nx106)) ;
(* no_enable_dff = "TRUE" *)
FDRE \modgen_counter_count_reg_q(1) (.Q (count_1_0[1]), .C (clk_int), .CE (
nx104), .D (nx15268z1), .R (nx106)) ;
(* no_enable_dff = "TRUE" *)
FDRE \modgen_counter_count_reg_q(0) (.Q (count_1_0[0]), .C (clk_int), .CE (
nx104), .D (nx16265z1), .R (nx106)) ;
endmodule
```

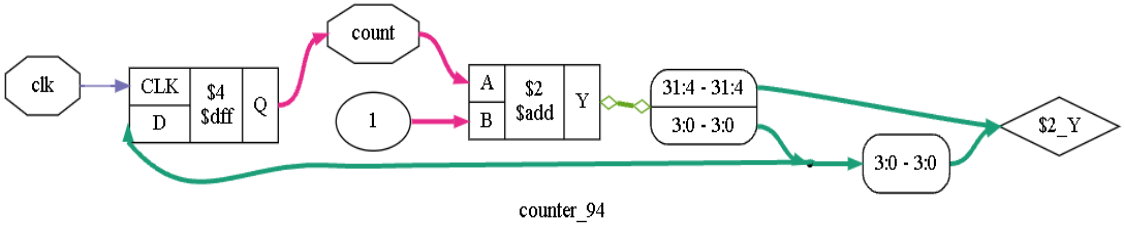


counter_94

BINARYCOUNT(UP) EDA OUTPUT

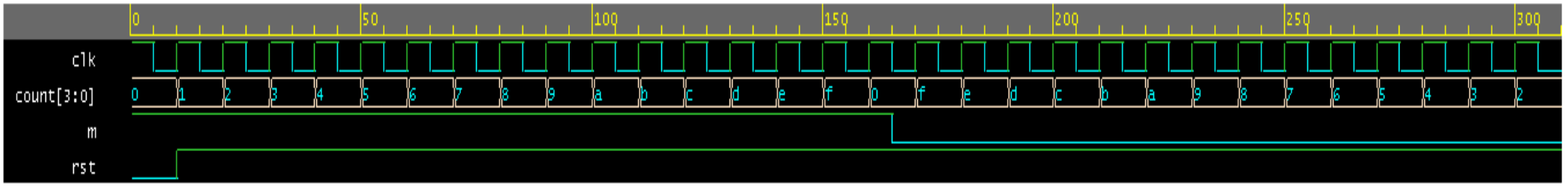



```
//
// Verilog description for cell counter_94,
// Mon Oct 18 13:42:49 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module counter_94 ( clk, count ) ;

input clk ;
output [3:0]count ;

wire clk_int;
wire n16265z1, n15268z1, n14271z1, n13274z1, n104, n106;
wire [3:0]count_1_0;



OBUF \count_obuf(0) (.O (count[0]), .I (count_1_0[0])) ;
OBUF \count_obuf(1) (.O (count[1]), .I (count_1_0[1])) ;
OBUF \count_obuf(2) (.O (count[2]), .I (count_1_0[2])) ;
OBUF \count_obuf(3) (.O (count[3]), .I (count_1_0[3])) ;
INV ix16265z1315 (.O (n16265z1), .I (count_1_0[0])) ;
LUT2 ix15268z1320 (.O (n15268z1), .I0 (count_1_0[1]), .I1 (count_1_0[0])) ;
defparam ix15268z1320.INIT = 4'h6;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix14271z1420 (.O (n14271z1), .I0 (count_1_0[2]), .I1 (count_1_0[1]), .I2 (
count_1_0[0])) ;
defparam ix14271z1420.INIT = 8'h6A;
(* HLUTNM = "LUT62_1_1" *)
LUT4 ix13274z28620 (.O (n13274z1), .I0 (count_1_0[3]), .I1 (count_1_0[2]),
.I2 (count_1_0[1]), .I3 (count_1_0[0])) ;
defparam ix13274z28620.INIT = 16'h6AAA;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
VCC ps_vcc (.P (n104)) ;
GND ps_gnd (.G (n106)) ;
(* no_enable_dff = "TRUE" *)
FDRE \modgen_counter_count_reg_q(3) (.Q (count_1_0[3]), .C (clk_int), .CE (
n104), .D (n13274z1), .R (n106)) ;
(* no_enable_dff = "TRUE" *)
FDRE \modgen_counter_count_reg_q(2) (.Q (count_1_0[2]), .C (clk_int), .CE (
n104), .D (n14271z1), .R (n106)) ;
(* no_enable_dff = "TRUE" *)
FDRE \modgen_counter_count_reg_q(1) (.Q (count_1_0[1]), .C (clk_int), .CE (
n104), .D (n15268z1), .R (n106)) ;
(* no_enable_dff = "TRUE" *)
FDRE \modgen_counter_count_reg_q(0) (.Q (count_1_0[0]), .C (clk_int), .CE (
n104), .D (n16265z1), .R (n106)) ;
endmodule
```

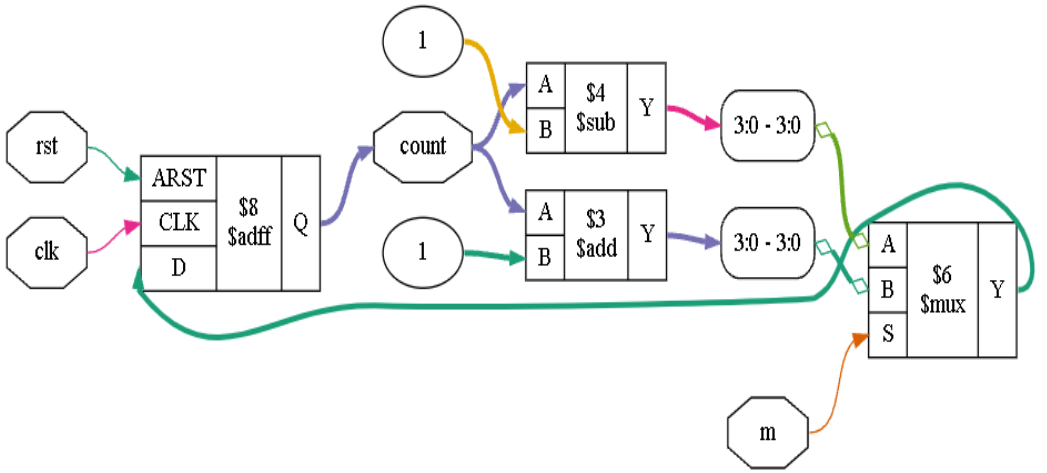


counter_94

BINARYCOUNT(UP_DOWN) EDA OUTPUT

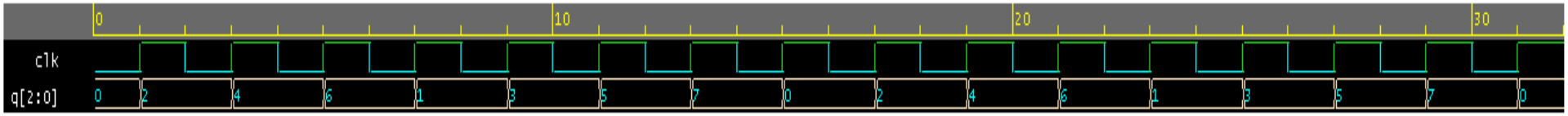



```
//
// Verilog description for cell counter_94,
// Mon Oct 18 13:46:21 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module counter_94 ( clk, m, rst, count ) ;

input clk ;
input m ;
input rst ;
output [3:0]count ;

wire clk_int;
wire m_int, rst_int, not_rst, n16265z1, n14271z1, n13274z1, n15268z1,
n106;
wire [3:0]count_1_0;


OBUF \count_obuf(0) (.O (count[0]), .I (count_1_0[0])) ;
OBUF \count_obuf(1) (.O (count[1]), .I (count_1_0[1])) ;
OBUF \count_obuf(2) (.O (count[2]), .I (count_1_0[2])) ;
OBUF \count_obuf(3) (.O (count[3]), .I (count_1_0[3])) ;
IBUF rst_ibuf (.O (rst_int), .I (rst)) ;
IBUF m_ibuf (.O (m_int), .I (m)) ;
INV ix13274z1315 (.O (not_rst), .I (rst_int)) ;
INV ix16265z1315 (.O (n16265z1), .I (count_1_0[0])) ;
(* HLUTNM = "LUT62_1_1" *)
LUT4 ix14271z28619 (.O (n14271z1), .I0 (count_1_0[2]), .I1 (count_1_0[1]),
.I2 (count_1_0[0]), .I3 (m_int)) ;
defparam ix14271z28619.INIT = 16'h6AA9;
(* HLUTNM = "LUT62_1_1" *)
LUT5 ix13274z45004 (.O (n13274z1), .I0 (count_1_0[3]), .I1 (count_1_0[2]),
.I2 (count_1_0[1]), .I3 (count_1_0[0]), .I4 (m_int)) ;
defparam ix13274z45004.INIT = 32'h6AAAAAA9;
LUT3 ix15268z1419 (.O (n15268z1), .I0 (count_1_0[1]), .I1 (count_1_0[0]), .I2 (
m_int)) ;
defparam ix15268z1419.INIT = 8'h69;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
\CC ps_vcc (.P (n106)) ;
(* no_enable_dff = "TRUE" *)
FDCE \modgen_counter_count_reg_q(3) (.Q (count_1_0[3]), .C (clk_int), .CE (
n106), .CLR (not_rst), .D (n13274z1)) ;
(* no_enable_dff = "TRUE" *)
FDCE \modgen_counter_count_reg_q(2) (.Q (count_1_0[2]), .C (clk_int), .CE (
n106), .CLR (not_rst), .D (n14271z1)) ;
(* no_enable_dff = "TRUE" *)
FDCE \modgen_counter_count_reg_q(1) (.Q (count_1_0[1]), .C (clk_int), .CE (
n106), .CLR (not_rst), .D (n15268z1)) ;
(* no_enable_dff = "TRUE" *)
FDCE \modgen_counter_count_reg_q(0) (.Q (count_1_0[0]), .C (clk_int), .CE (
n106), .CLR (not_rst), .D (n16265z1)) ;
endmodule
```

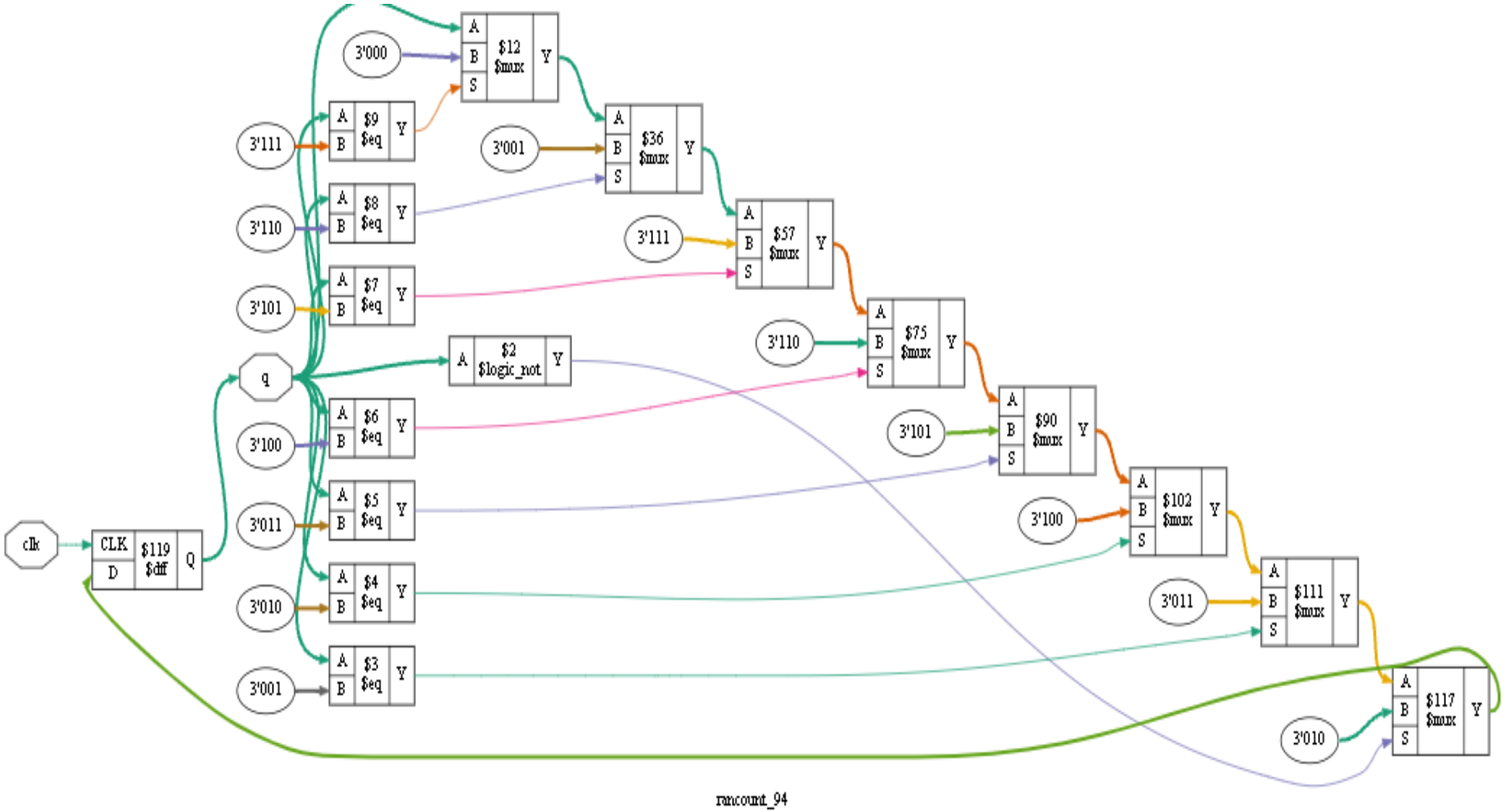


counter_94

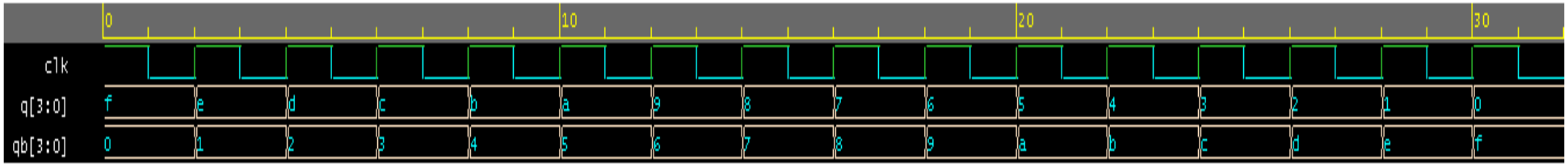
RANCOUNT EDA OUTPUT




```
// Verilog description for cell rancount_94,
// Mon Oct 18 13:51:12 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module rancount_94 ( clk, q ) ;

input clk ;
output [2:0]q ;

wire clk_int;
wire not_q_1, nx51271z1, nx53265z1, nx193, nx195;
wire [2:0]q_1_0;



OBUF \q_obuf(0) (.O (q[0]), .I (q_1_0[0])) ;
OBUF \q_obuf(1) (.O (q[1]), .I (q_1_0[1])) ;
OBUF \q_obuf(2) (.O (q[2]), .I (q_1_0[2])) ;
INV ix52268z1315 (.O (not_q_1), .I (q_1_0[1])) ;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix51271z1434 (.O (nx51271z1), .I0 (q_1_0[2]), .I1 (q_1_0[1]), .I2 (
q_1_0[0])) ;
defparam ix51271z1434.INIT = 8'h78;
(* HLUTNM = "LUT62_1_1" *)
LUT2 ix53265z1320 (.O (nx53265z1), .I0 (q_1_0[2]), .I1 (q_1_0[1])) ;
defparam ix53265z1320.INIT = 4'h6;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
VCC ps_vcc (.P (nx193)) ;
GND ps_gnd (.G (nx195)) ;
FDRE \reg_q(2) (.Q (q_1_0[2]), .C (clk_int), .CE (nx193), .D (nx53265z1), .R (
nx195)) ;
defparam \reg_q(2) .INIT = 1'b0;
FDRE \reg_q(1) (.Q (q_1_0[1]), .C (clk_int), .CE (nx193), .D (not_q_1), .R (
nx195)) ;
defparam \reg_q(1) .INIT = 1'b0;
FDRE \reg_q(0) (.Q (q_1_0[0]), .C (clk_int), .CE (nx193), .D (nx51271z1), .R (
nx195)) ;
defparam \reg_q(0) .INIT = 1'b0;
endmodule
```

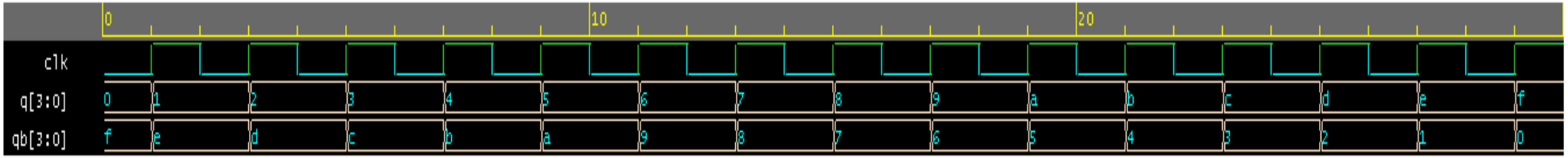

clk
CLK
D
$119
$dff
Q
q
3'000
3'111
3'110
3'101
3'100
3'011
3'010
3'001
A
B
S
$12
$mux
Y
$9
$eq
$8
$7
$6
$5
$4
$3
$2
$logic_not
3'001
3'111
3'110
3'101
3'100
3'011
3'010
$36
$57
$75
$90
$102
$111
$117
rancount_94

SYNCOUNT(DOWN) EDA OUTPUT




```
//
// Verilog description for cell syncount_94,
// Mon Oct 18 13:52:54 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module syncount_94 ( q, qb, clk ) ;

output [3:0]q ;
output [3:0]qb ;
input clk ;

wire clk_int;
wire g1_not_q, nx2737z2, nx33128z2, nx63910z1, nx33128z1, nx23821z1,
nx2737z1, nx19520z1, nx27654z1, nx131, nx133;
wire [3:0]q_1_0;
wire [3:0]qb_1_0;



OBUF \qb_obuf(0) (.O (qb[0]), .I (qb_1_0[0])) ;
OBUF \qb_obuf(1) (.O (qb[1]), .I (qb_1_0[1])) ;
OBUF \qb_obuf(2) (.O (qb[2]), .I (qb_1_0[2])) ;
OBUF \qb_obuf(3) (.O (qb[3]), .I (qb_1_0[3])) ;
OBUF \q_obuf(0) (.O (q[0]), .I (q_1_0[0])) ;
OBUF \q_obuf(1) (.O (q[1]), .I (q_1_0[1])) ;
OBUF \q_obuf(2) (.O (q[2]), .I (q_1_0[2])) ;
OBUF \q_obuf(3) (.O (q[3]), .I (q_1_0[3])) ;
INV ix58045z1315 (.O (g1_not_q), .I (q_1_0[0])) ;
INV ix2737z1316 (.O (nx2737z2), .I (q_1_0[2])) ;
INV ix33128z1316 (.O (nx33128z2), .I (q_1_0[3])) ;
(* HLUTNM = "LUT62_1_1" *)
LUT4 ix63910z39543 (.O (nx63910z1), .I0 (q_1_0[3]), .I1 (qb_1_0[2]), .I2 (
qb_1_0[1]), .I3 (qb_1_0[0])) ;
defparam ix63910z39543.INIT = 16'h9555;
(* HLUTNM = "LUT62_1_2" *)
LUT3 ix33128z1442 (.O (nx33128z1), .I0 (qb_1_0[2]), .I1 (qb_1_0[1]), .I2 (
qb_1_0[0])) ;
defparam ix33128z1442.INIT = 8'h80;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix23821z1463 (.O (nx23821z1), .I0 (q_1_0[2]), .I1 (qb_1_0[1]), .I2 (
qb_1_0[0])) ;
defparam ix23821z1463.INIT = 8'h95;
(* HLUTNM = "LUT62_1_3" *)
LUT2 ix2737z1322 (.O (nx2737z1), .I0 (qb_1_0[1]), .I1 (qb_1_0[0])) ;
defparam ix2737z1322.INIT = 4'h8;
(* HLUTNM = "LUT62_1_2" *)
LUT2 ix19520z1323 (.O (nx19520z1), .I0 (q_1_0[1]), .I1 (qb_1_0[0])) ;
defparam ix19520z1323.INIT = 4'h9;
(* HLUTNM = "LUT62_1_3" *)
LUT2 ix27654z1318 (.O (nx27654z1), .I0 (q_1_0[1]), .I1 (qb_1_0[0])) ;
defparam ix27654z1318.INIT = 4'h4;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
VCC ps_vcc (.P (nx131)) ;
GND ps_gnd (.G (nx133)) ;
FDRE g1_reg_q (.Q (q_1_0[0]), .C (clk_int), .CE (nx131), .D (g1_not_q), .R (
nx133)) ;
FDRE g1_reg_qb (.Q (qb_1_0[0]), .C (clk_int), .CE (nx131), .D (q_1_0[0]), .R (
nx133)) ;
FDRE g2_reg_q (.Q (q_1_0[1]), .C (clk_int), .CE (qb_1_0[0]), .D (nx27654z1)
, .R (nx133)) ;
FDRE g2_reg_qb (.Q (qb_1_0[1]), .C (clk_int), .CE (nx131), .D (nx19520z1), .R (
nx133)) ;
FDRE g3_reg_q (.Q (q_1_0[2]), .C (clk_int), .CE (nx2737z1), .D (nx2737z2), .R (
nx133)) ;
FDRE g3_reg_qb (.Q (qb_1_0[2]), .C (clk_int), .CE (nx131), .D (nx23821z1), .R (
nx133)) ;
FDRE g4_reg_q (.Q (q_1_0[3]), .C (clk_int), .CE (nx33128z1), .D (nx33128z2)
, .R (nx133)) ;
FDRE g4_reg_qb (.Q (qb_1_0[3]), .C (clk_int), .CE (nx131), .D (nx63910z1), .R (
nx133)) ;
endmodule
```

SYNCOUNT(UP) EDA OUTPUT




```
//
// Verilog description for cell syncount_94,
// Mon Oct 18 13:57:30 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module syncount_94 ( q, qb, clk ) ;

output [3:0]q ;
output [3:0]qb ;
input clk ;

wire clk_int;
wire g1_not_q, nx2737z2, nx33128z2, nx63910z1, nx23821z1, nx33128z1,
nx19520z1, nx2737z1, nx27654z1, nx131, nx133;
wire [3:0]q_1_0;
wire [3:0]qb_1_0;



OBUF \qb_obuf(0) (.O (qb[0]), .I (qb_1_0[0])) ;
OBUF \qb_obuf(1) (.O (qb[1]), .I (qb_1_0[1])) ;
OBUF \qb_obuf(2) (.O (qb[2]), .I (qb_1_0[2])) ;
OBUF \qb_obuf(3) (.O (qb[3]), .I (qb_1_0[3])) ;
OBUF \q_obuf(0) (.O (q[0]), .I (q_1_0[0])) ;
OBUF \q_obuf(1) (.O (q[1]), .I (q_1_0[1])) ;
OBUF \q_obuf(2) (.O (q[2]), .I (q_1_0[2])) ;
OBUF \q_obuf(3) (.O (q[3]), .I (q_1_0[3])) ;
INV ix58045z1315 (.O (g1_not_q), .I (q_1_0[0])) ;
INV ix2737z1316 (.O (nx2737z2), .I (q_1_0[2])) ;
INV ix33128z1316 (.O (nx33128z2), .I (q_1_0[3])) ;
(* HLUTNM = "LUT62_1_1" *)
LUT4 ix63910z39543 (.O (nx63910z1), .I0 (q_1_0[3]), .I1 (q_1_0[2]), .I2 (
q_1_0[1]), .I3 (q_1_0[0])) ;
defparam ix63910z39543.INIT = 16'h9555;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix23821z1463 (.O (nx23821z1), .I0 (q_1_0[2]), .I1 (q_1_0[1]), .I2 (
q_1_0[0])) ;
defparam ix23821z1463.INIT = 8'h95;
(* HLUTNM = "LUT62_1_2" *)
LUT3 ix33128z1442 (.O (nx33128z1), .I0 (q_1_0[2]), .I1 (q_1_0[1]), .I2 (
q_1_0[0])) ;
defparam ix33128z1442.INIT = 8'h80;
(* HLUTNM = "LUT62_1_2" *)
LUT2 ix19520z1323 (.O (nx19520z1), .I0 (q_1_0[1]), .I1 (q_1_0[0])) ;
defparam ix19520z1323.INIT = 4'h9;
(* HLUTNM = "LUT62_1_3" *)
LUT2 ix2737z1322 (.O (nx2737z1), .I0 (q_1_0[1]), .I1 (q_1_0[0])) ;
defparam ix2737z1322.INIT = 4'h8;
(* HLUTNM = "LUT62_1_3" *)
LUT2 ix27654z1318 (.O (nx27654z1), .I0 (q_1_0[1]), .I1 (q_1_0[0])) ;
defparam ix27654z1318.INIT = 4'h4;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
VCC ps_vcc (.P (nx131)) ;
GND ps_gnd (.G (nx133)) ;
FDRE g1_reg_q (.Q (q_1_0[0]), .C (clk_int), .CE (nx131), .D (g1_not_q), .R (
nx133)) ;
FDRE g1_reg_qb (.Q (qb_1_0[0]), .C (clk_int), .CE (nx131), .D (q_1_0[0]), .R (
nx133)) ;
FDRE g2_reg_q (.Q (q_1_0[1]), .C (clk_int), .CE (q_1_0[0]), .D (nx27654z1),
.R (nx133)) ;
FDRE g2_reg_qb (.Q (qb_1_0[1]), .C (clk_int), .CE (nx131), .D (nx19520z1), .R (
nx133)) ;
FDRE g3_reg_q (.Q (q_1_0[2]), .C (clk_int), .CE (nx2737z1), .D (nx2737z2), .R (
nx133)) ;
FDRE g3_reg_qb (.Q (qb_1_0[2]), .C (clk_int), .CE (nx131), .D (nx23821z1), .R (
nx133)) ;
FDRE g4_reg_q (.Q (q_1_0[3]), .C (clk_int), .CE (nx33128z1), .D (nx33128z2)
, .R (nx133)) ;
FDRE g4_reg_qb (.Q (qb_1_0[3]), .C (clk_int), .CE (nx131), .D (nx63910z1), .R (
nx133)) ;
endmodule
```

Date :
30-9-2021

Experiment No. 8

Shift Registers

Aim: To implement Shift registers (SISO, SIPO, PISO, PIPO) using verilog, verify the design using testbench and perform functional simulation and to synthesize.

Requirement: EDA playground Tool.

Theory :

Shift registers are sequential logic circuits, capable of storage and transfer of data. They are of both parallel and serial inputs and outputs. SISO, SIPO, PISO and PIPO are the types of shift registers and they are bidirectional, allowing to shift in both directions (left and right)

* SISO - Serial-in, Serial-out
* SIPO - Serial-in, Parallel-out
* PISO - Parallel-in, Serial-out
* PIPO - Parallel-in, Parallel-out

PIPO EDA

DESIGN CODE	TEST BENCH

module pipo_94(din,clk,reset,dout);
output[3:0]dout;
wire[3:0]dout;
input[3:0]din;
wire[3:0]din;
input clk;
wire clk;
input reset;
wire reset;
reg[3:0]s;
 	
always @(*)
begin
if (reset)
s<=0;
else
begin
s <= din;
end
end
assign dout=s;
endmodule
	module pipo_94test;
	reg clk,reset;
	reg[3:0]din;
	wire[3:0]dout;
	pipo_94 g1(din,clk,reset,dout);
	initial
	begin
	#0 clk=0;reset=1;
	#3 din=4'b1110;reset=0;
	#3 din=4'b1111;
	end
	initial
	#35 $stop;
	always
	begin
	#1 clk=~clk;
	end
	
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, pipo_94test);
	end
	endmodule

PISO EDA

DESIGN CODE	TEST BENCH

module piso_94(clk,reset,din,dout);
input clk,reset;
input [3:0]din;
output dout;
reg dout;
reg [3:0]s;
always@(posedge clk)
begin
if(reset==1'b1)
begin
dout<=0;
s<=din;
end
else
begin
dout<=s[0];
s <= s>>1;
end
end
endmodule
	module piso_94test;
	reg clk, reset;
	reg [3:0]din;
	wire dout;
	piso_94 g1(clk, reset, din, dout);
	initial
	clk=1;
	always
	#1 clk=~clk;
	initial
	begin
	#0 reset=1; din=4'b1001;
	#4 reset=0;
	#8 reset=1; din=4'b1111;
	#4 reset=0;
	end
	initial
	#24 $stop;
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, piso_94test);
	end
	endmodule

SIPO EDA

DESIGN CODE	TEST BENCH

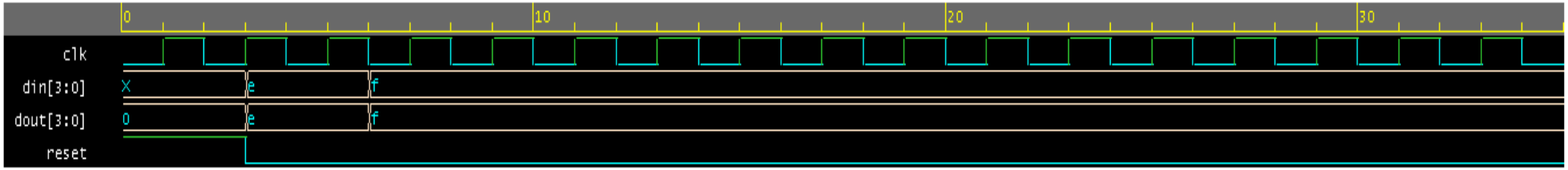
module sipo_94(din,clk,reset,dout);
output [3:0]dout;
wire [3:0]dout;
input din;
wire din;
input clk;
wire clk;
input reset;
wire reset;
reg[3:0]s;
 	
always@(posedge clk)
begin
if(reset)
s<=0;
else
begin
s[3]<=din;
s[2]<=s[3];
s[1]<=s[2];
s[0]<=s[1];
end
end
assign dout=s;
endmodule
	module sipo_94test;
	reg clk,din,reset;
	wire[3:0]dout;
	sipo_94 g1(din,clk,reset,dout);
	initial
	begin
	#0 clk=0;din=1;reset=1;
	#3 din=1;reset=0;
	end
	initial
	#35 $stop;
	always
	begin
	#1 clk=~clk;
	end
	
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, sipo_94test);
	end
	endmodule

SISO EDA

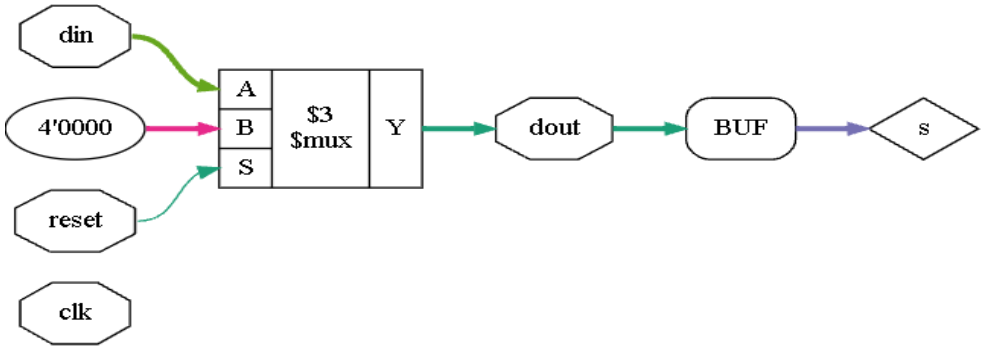
DESIGN CODE	TEST BENCH

module siso_94(din,clk,reset,dout);
output [3:0]dout;
wire [3:0]dout;
input din;
wire din;
input clk;
wire clk;
input reset;
wire reset;
reg[3:0]s;
always@(posedge clk)
begin
if(reset)
s<=0;
else
begin
s[3]<=din;
@(posedge clk)
s[2]<=s[3];
@(posedge clk)
s[1]<=s[2];
@(posedge clk)
s[0]<=s[1];
end
end
assign dout=s;
endmodule
	module siso_94test;
	reg clk,din,reset;
	wire[3:0]dout;
	siso_94 g1(din,clk,reset,dout);
	initial
	begin
	#0 clk=0;din=1;reset=1;
	#2 din=1;reset=0;
	#3 din=1;
	#2 din=0;
	end
	initial
	#35 $stop;
	always
	begin
	#1 clk=~clk;
	end
	
	initial
	begin
	$dumpfile("dump.vcd");
	$dumpvars(1, siso_94test);
	end
	endmodule

PIPO EDA OUTPUT

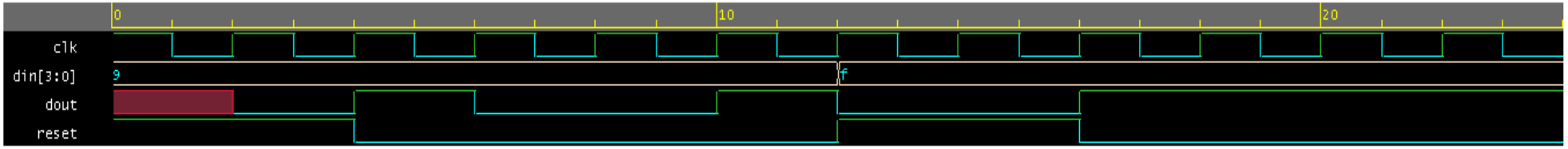



```
//
// Verilog description for cell pipo_94,
// Mon Oct 18 15:14:23 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module pipo_94 ( din, clk, reset, dout ) ;

input [3:0]din ;
input clk ;
input reset ;
output [3:0]dout ;

wire [3:0]din_int;
wire reset_int, nx17199z1, nx16202z1, nx15205z1, nx14208z1;



OBUF \dout_obuf(0) (.O (dout[0]), .I (nx14208z1)) ;
OBUF \dout_obuf(1) (.O (dout[1]), .I (nx15205z1)) ;
OBUF \dout_obuf(2) (.O (dout[2]), .I (nx16202z1)) ;
OBUF \dout_obuf(3) (.O (dout[3]), .I (nx17199z1)) ;
IBUF reset_ibuf (.O (reset_int), .I (reset)) ;
IBUF \din_ibuf(0) (.O (din_int[0]), .I (din[0])) ;
IBUF \din_ibuf(1) (.O (din_int[1]), .I (din[1])) ;
IBUF \din_ibuf(2) (.O (din_int[2]), .I (din[2])) ;
IBUF \din_ibuf(3) (.O (din_int[3]), .I (din[3])) ;
(* HLUTNM = "LUT62_1_2" *)
LUT2 ix17199z1316 (.O (nx17199z1), .I0 (din_int[3]), .I1 (reset_int)) ;
defparam ix17199z1316.INIT = 4'h2;
(* HLUTNM = "LUT62_1_2" *)
LUT2 ix16202z1316 (.O (nx16202z1), .I0 (din_int[2]), .I1 (reset_int)) ;
defparam ix16202z1316.INIT = 4'h2;
(* HLUTNM = "LUT62_1_1" *)
LUT2 ix15205z1316 (.O (nx15205z1), .I0 (din_int[1]), .I1 (reset_int)) ;
defparam ix15205z1316.INIT = 4'h2;
(* HLUTNM = "LUT62_1_1" *)
LUT2 ix14208z1316 (.O (nx14208z1), .I0 (din_int[0]), .I1 (reset_int)) ;
defparam ix14208z1316.INIT = 4'h2;
endmodule
```

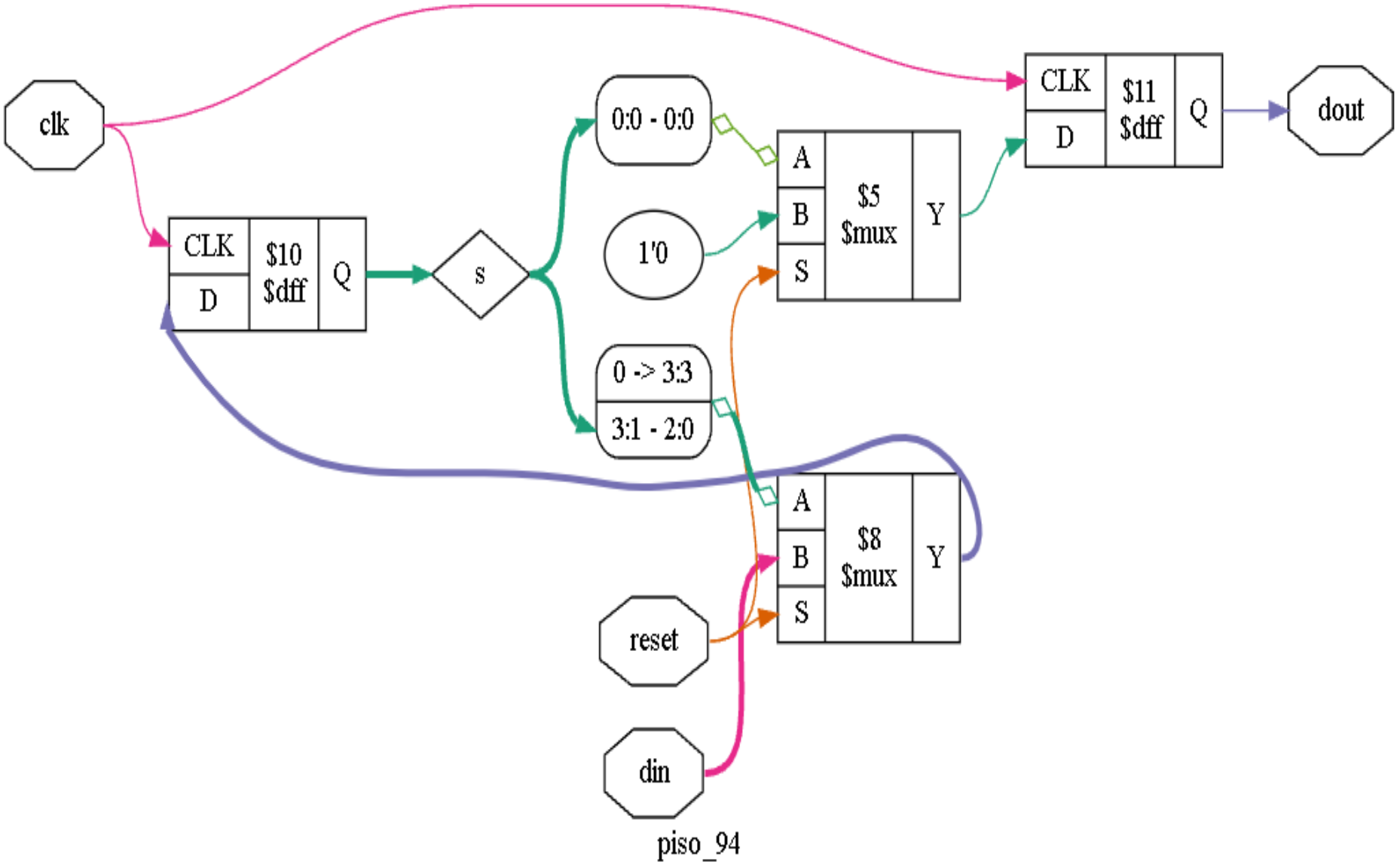


pipo_94

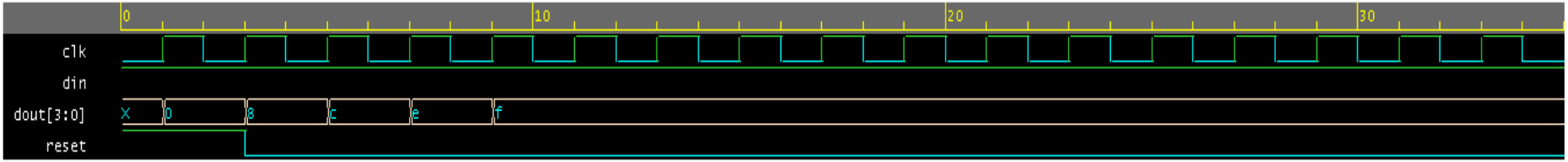
PISO EDA OUTPUT



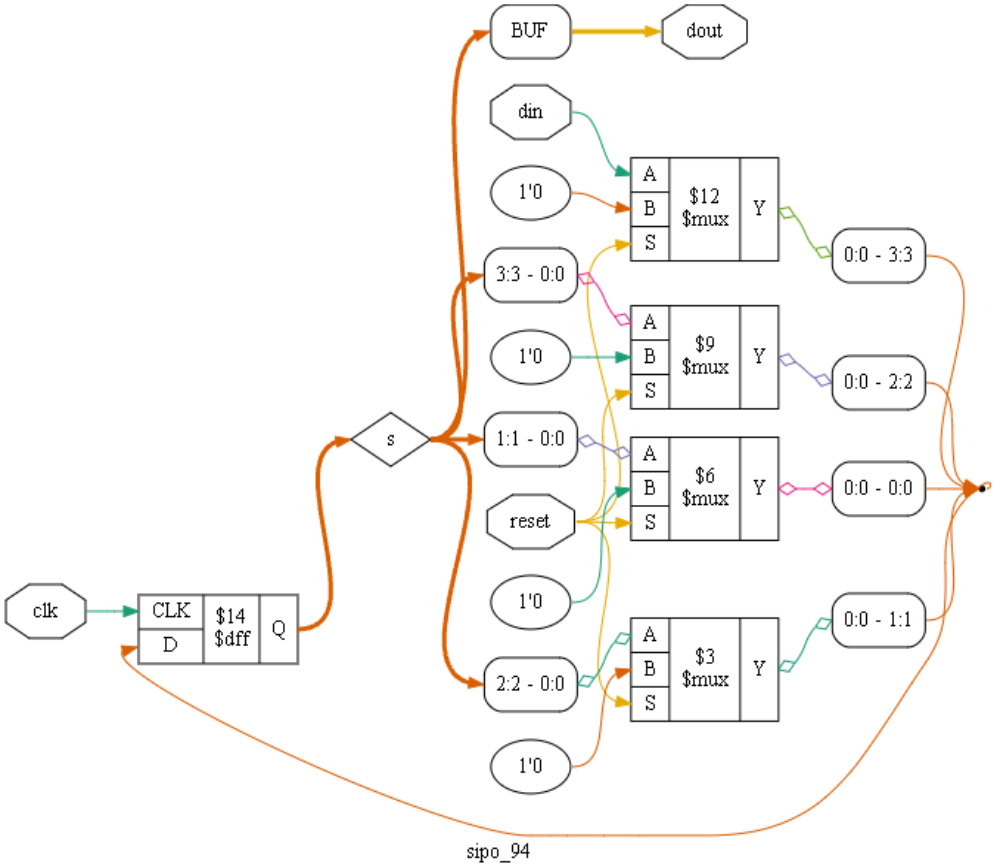

```
//
// Verilog description for cell piso_94,
// Mon Oct 18 15:17:00 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module piso_94 ( clk, reset, din, dout ) ;

input clk ;
input reset ;
input [3:0]din ;
output dout ;

wire clk_int;
wire reset_int;
wire [3:0]din_int;
wire not_reset, nx31103z1, nx30106z1, nx29109z1, nx111;
wire [3:0]s;
wire dout_1_0, nx117;



OBUF dout_obuf (.O (dout), .I (dout_1_0)) ;
IBUF \din_ibuf(0) (.O (din_int[0]), .I (din[0])) ;
IBUF \din_ibuf(1) (.O (din_int[1]), .I (din[1])) ;
IBUF \din_ibuf(2) (.O (din_int[2]), .I (din[2])) ;
IBUF \din_ibuf(3) (.O (din_int[3]), .I (din[3])) ;
IBUF reset_ibuf (.O (reset_int), .I (reset)) ;
INV ix28112z1315 (.O (not_reset), .I (reset_int)) ;
LUT3 ix31103z1540 (.O (nx31103z1), .I0 (s[1]), .I1 (reset_int), .I2 (
din_int[0])) ;
defparam ix31103z1540.INIT = 8'hE2;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix30106z1540 (.O (nx30106z1), .I0 (s[2]), .I1 (reset_int), .I2 (
din_int[1])) ;
defparam ix30106z1540.INIT = 8'hE2;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix29109z1540 (.O (nx29109z1), .I0 (s[3]), .I1 (reset_int), .I2 (
din_int[2])) ;
defparam ix29109z1540.INIT = 8'hE2;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
VCC ps_vcc (.P (nx111)) ;
FDRE \reg_s(3) (.Q (s[3]), .C (clk_int), .CE (nx111), .D (din_int[3]), .R (
not_reset)) ;
FDRE reg_dout (.Q (dout_1_0), .C (clk_int), .CE (nx111), .D (s[0]), .R (
reset_int)) ;
GND ps_gnd (.G (nx117)) ;
FDRE \reg_s(2) (.Q (s[2]), .C (clk_int), .CE (nx111), .D (nx29109z1), .R (
nx117)) ;
FDRE \reg_s(1) (.Q (s[1]), .C (clk_int), .CE (nx111), .D (nx30106z1), .R (
nx117)) ;
FDRE \reg_s(0) (.Q (s[0]), .C (clk_int), .CE (nx111), .D (nx31103z1), .R (
nx117)) ;
endmodule
```


clk
CLK
D
$10
$dff
Q
s
0:0 - 0:0
1'0
0 -> 3:3
3:1 - 2:0
A
B
S
$5
$mux
Y
A
B
S
$8
$mux
Y
reset
din
CLK
D
$11
$dff
Q
dout
piso_94

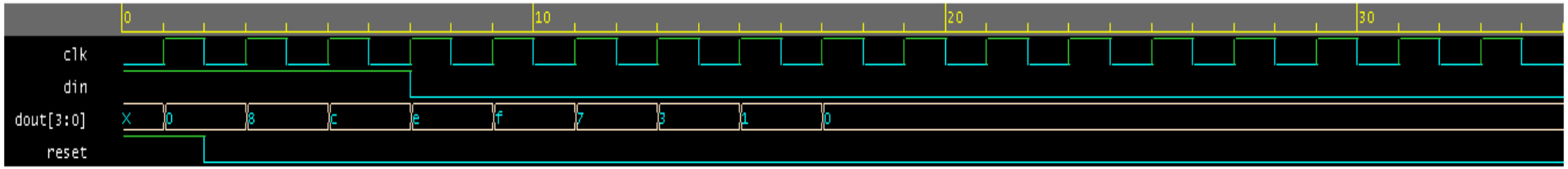
SIPO EDA OUTPUT




```
//
// Verilog description for cell sipo_94,
// Mon Oct 18 15:21:01 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module sipo_94 ( din, clk, reset, dout ) ;

input din ;
input clk ;
input reset ;
output [3:0]dout ;

wire din_int;
wire clk_int;
wire reset_int, nx87;
wire [3:0]dout_1_0;



OBUF \dout_obuf(0) (.O (dout[0]), .I (dout_1_0[0])) ;
OBUF \dout_obuf(1) (.O (dout[1]), .I (dout_1_0[1])) ;
OBUF \dout_obuf(2) (.O (dout[2]), .I (dout_1_0[2])) ;
OBUF \dout_obuf(3) (.O (dout[3]), .I (dout_1_0[3])) ;
IBUF reset_ibuf (.O (reset_int), .I (reset)) ;
IBUF din_ibuf (.O (din_int), .I (din)) ;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
VCC ps_vcc (.P (nx87)) ;
FDRE \reg_s(3) (.Q (dout_1_0[3]), .C (clk_int), .CE (nx87), .D (din_int), .R (
reset_int)) ;
FDRE \reg_s(2) (.Q (dout_1_0[2]), .C (clk_int), .CE (nx87), .D (dout_1_0[3]
), .R (reset_int)) ;
FDRE \reg_s(1) (.Q (dout_1_0[1]), .C (clk_int), .CE (nx87), .D (dout_1_0[2]
), .R (reset_int)) ;
FDRE \reg_s(0) (.Q (dout_1_0[0]), .C (clk_int), .CE (nx87), .D (dout_1_0[1]
), .R (reset_int)) ;
endmodule
```



SISO EDA OUTPUT




```
//
// Verilog description for cell siso_94,
// Mon Oct 18 15:23:57 2021
//
// Precision RTL Synthesis, 64-bit 2021.1.0.4//


module siso_94 ( din, clk, reset, dout ) ;

input din ;
input clk ;
input reset ;
output [3:0]dout ;

wire din_int;
wire clk_int;
wire reset_int, nx16430z1, nx31103z1, nx15433z1, nx31103z2, nx30106z2,
nx30106z1, nx29109z2, nx29109z1, nx28112z2, nx28112z1, nx261;
wire [3:0]dout_1_0;
wire nx271;
wire [1:0]_STATE_VAR_;



OBUF \dout_obuf(0) (.O (dout[0]), .I (dout_1_0[0])) ;
OBUF \dout_obuf(1) (.O (dout[1]), .I (dout_1_0[1])) ;
OBUF \dout_obuf(2) (.O (dout[2]), .I (dout_1_0[2])) ;
OBUF \dout_obuf(3) (.O (dout[3]), .I (dout_1_0[3])) ;
IBUF reset_ibuf (.O (reset_int), .I (reset)) ;
IBUF din_ibuf (.O (din_int), .I (din)) ;
(* HLUTNM = "LUT62_1_3" *)
LUT2 ix16430z1320 (.O (nx16430z1), .I0 (_STATE_VAR_[1]), .I1 (_STATE_VAR_[0]
)) ;
defparam ix16430z1320.INIT = 4'h6;
(* HLUTNM = "LUT62_1_2" *)
LUT3 ix31103z1466 (.O (nx31103z1), .I0 (_STATE_VAR_[1]), .I1 (_STATE_VAR_[0]
), .I2 (reset_int)) ;
defparam ix31103z1466.INIT = 8'h98;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix15433z1349 (.O (nx15433z1), .I0 (_STATE_VAR_[1]), .I1 (_STATE_VAR_[0]
), .I2 (reset_int)) ;
defparam ix15433z1349.INIT = 8'h23;
(* HLUTNM = "LUT62_1_5" *)
LUT2 ix31103z1323 (.O (nx31103z2), .I0 (dout_1_0[1]), .I1 (_STATE_VAR_[0])
) ;
defparam ix31103z1323.INIT = 4'h8;
(* HLUTNM = "LUT62_1_5" *)
LUT2 ix30106z1323 (.O (nx30106z2), .I0 (dout_1_0[2]), .I1 (_STATE_VAR_[1])
) ;
defparam ix30106z1323.INIT = 4'h8;
(* HLUTNM = "LUT62_1_2" *)
LUT3 ix30106z1364 (.O (nx30106z1), .I0 (_STATE_VAR_[1]), .I1 (_STATE_VAR_[0]
), .I2 (reset_int)) ;
defparam ix30106z1364.INIT = 8'h32;
(* HLUTNM = "LUT62_1_4" *)
LUT2 ix29109z1323 (.O (nx29109z2), .I0 (dout_1_0[3]), .I1 (_STATE_VAR_[0])
) ;
defparam ix29109z1323.INIT = 4'h8;
(* HLUTNM = "LUT62_1_1" *)
LUT3 ix29109z1398 (.O (nx29109z1), .I0 (_STATE_VAR_[1]), .I1 (_STATE_VAR_[0]
), .I2 (reset_int)) ;
defparam ix29109z1398.INIT = 8'h54;
(* HLUTNM = "LUT62_1_4" *)
LUT2 ix28112z1317 (.O (nx28112z2), .I0 (din_int), .I1 (reset_int)) ;
defparam ix28112z1317.INIT = 4'h2;
(* HLUTNM = "LUT62_1_3" *)
LUT2 ix28112z1315 (.O (nx28112z1), .I0 (_STATE_VAR_[1]), .I1 (_STATE_VAR_[0]
)) ;
defparam ix28112z1315.INIT = 4'h1;
BUFGP clk_ibuf (.O (clk_int), .I (clk)) ;
GND ps_gnd (.G (nx261)) ;
FDRE \reg_s(3) (.Q (dout_1_0[3]), .C (clk_int), .CE (nx28112z1), .D (
nx28112z2), .R (nx261)) ;
FDRE \reg_s(2) (.Q (dout_1_0[2]), .C (clk_int), .CE (nx29109z1), .D (
nx29109z2), .R (nx261)) ;
FDRE \reg_s(1) (.Q (dout_1_0[1]), .C (clk_int), .CE (nx30106z1), .D (
nx30106z2), .R (nx261)) ;
FDRE \reg_s(0) (.Q (dout_1_0[0]), .C (clk_int), .CE (nx31103z1), .D (
nx31103z2), .R (nx261)) ;
VCC ps_vcc (.P (nx271)) ;
FDRE \reg__STATE_VAR_(1) (.Q (_STATE_VAR_[1]), .C (clk_int), .CE (nx271), .D (
nx16430z1), .R (nx261)) ;
defparam \reg__STATE_VAR_(1) .INIT = 1'b0;
FDRE \reg__STATE_VAR_(0) (.Q (_STATE_VAR_[0]), .C (clk_int), .CE (nx271), .D (
nx15433z1), .R (nx261)) ;
defparam \reg__STATE_VAR_(0) .INIT = 1'b0;
endmodule
```

Date:
04-10-2021

Experiment No. 9
Memory Design

Aim: To implement Memory design (RAM cell) using verilog, verify the design using Testbench and perform functional simulation and to synthesize.

Requirement: EDA playground Tool

Theory:

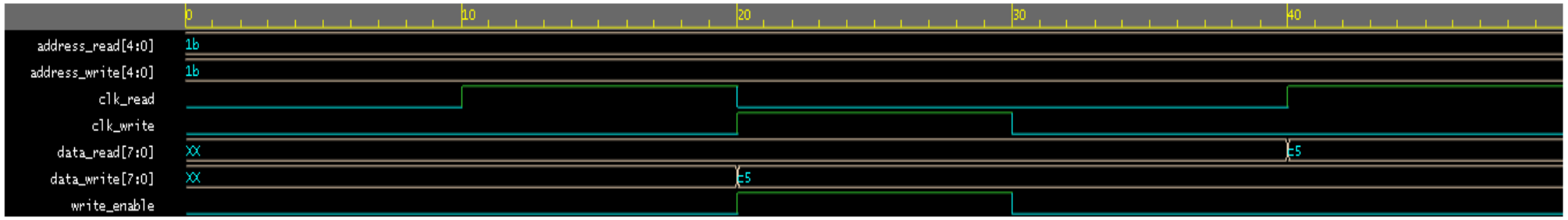
There are many types RAM There differ in terms of the no. of ports, synchronous or asynchronous modes of operation etc., Here we have taken a single port RAM. The reading is done from the port synchronously The RAM specified is a 8-bit data (i/p or o/p) and 5-bit address line making it a 32-bit x 8-bit RAM design with common read and write functions.

MEMORY_RAM EDA

DESIGN CODE	TEST BENCH

module ram_94 (clk_write,
address_write,data_write,
write_enable,clk_read, address_read,
data_read);
parameter D_WIDTH = 16;
parameter A_WIDTH = 4;
parameter A_MAX = 16;
input clk_write;
input [A_WIDTH-1:0] address_write;
input [D_WIDTH-1:0] data_write;
input write_enable;
input clk_read;
input [A_WIDTH-1:0] address_read;
output [D_WIDTH-1:0] data_read;
reg [D_WIDTH-1:0] data_read;
reg [D_WIDTH-1:0] memory [A_MAX-1:0];
always @(posedge clk_write) begin
if (write_enable) begin
memory[address_write] <= data_write;
end
end
always @(posedge clk_read) begin
data_read <= memory[address_read];
end
endmodule
	module ram_94test;
	reg clk_write;
	reg [4:0] address_write;
	reg [7:0] data_write;
	reg write_enable;
	reg clk_read;
	reg [4:0] address_read;
	wire [7:0] data_read;
	ram_94 #(8, 5, 32) RAM
	(.clk_write(clk_write),.address_write(address_write),
	.data_write(data_write),.write_enable(write_enable)
	,.clk_read(clk_read),.address_read(address_read)
	,.data_read(data_read));
	initial begin
	$dumpfile("dump.vcd");
	$dumpvars(1, ram_94test);
	clk_write = 0; clk_read = 0;
	write_enable = 0;
	address_read = 5'h1B;
	address_write = address_read;
	$display("Read initial data.");
	toggle_clk_read;
	$display("data[%0h]: %0h",
	address_read, data_read);
	$display("Write new data.");
	write_enable = 1;
	data_write = 8'hC5;
	toggle_clk_write;
	write_enable = 0;
	$display("Read new data.");
	toggle_clk_read;
	$display("data[%0h]: %0h",
	address_read, data_read);
	end
	task toggle_clk_write;
	begin
	#0 clk_write = ~clk_write;
	#10 clk_write = ~clk_write;
	end
	endtask
	task toggle_clk_read;
	begin
	#10 clk_read = ~clk_read;
	#10 clk_read = ~clk_read;
	end
	endtask
	
	endmodule

MEMORY_RAM EDA OUTPUT



Date:
11-10-2021

Experiment No. 10

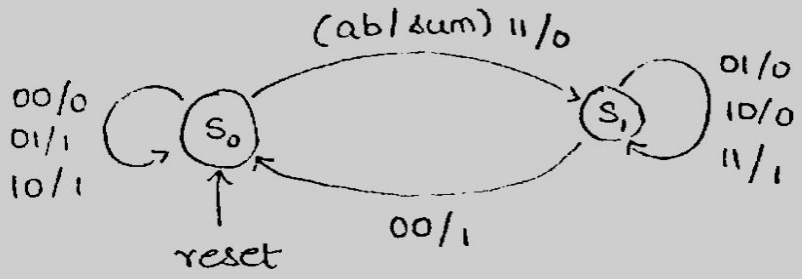
FSM Serial Adder

Aim: To implement FSM serial adder using verilog, verify the design using Testbench and perform functional simulation and to synthesize.

Requirement: EDA playground Tool.

Theory:

FSM serial adder is a simple, circuit that can be used to add numbers of any length.



State { S_0 presents carry in = 0
S_1 presents carry in = 1

input valuation	output (s)	State
00	0	remain in state S_0
01, 10	1	remain in state S_0
11	0	moves to S_1
01, 10	0	remain in state S_1
11	1	remain in state S_1
00	1	moves to S_0

FSM EDA

DESIGN CODE	TEST BENCH

module FSM_sa_94 (z, x1, x2, clk, reset) ;
output reg z ;
input x1, x2, reset;
input clk ;
parameter A=0, B=1;
reg state ;
reg q,qb;
initial q = 0;
initial qb = 0 ;
 	
always @ (posedge clk or reset)
begin
if (reset)
begin
state=A;z=0 ;
end
else
case (state)
A : if (x1 && x2)
state <= B ;
B : if (~x1 && ~x2)
state <= A ;
endcase
end
 	
always @ (*)
begin
if(state==0)
case ({x1,x2})
2'b00 : begin q=~q;z=~q; end
2'b01 : z = 1 ;
2'b10 : z = 1 ;
2'b11 : begin q=~q;z=~q; end
endcase
else
case ({x1,x2})
2'b00 : begin q=~q;z=~q; end
2'b01 : z = 0 ;
2'b10 : z = 0 ;
2'b11 : begin qb=~qb;z=~qb; end
endcase
end
endmodule
	module FSM_sa_94test ;
	
	reg x1, x2, clk, reset ;
	wire z ;
	
	FSM_sa_94 g1 (z, x1, x2, clk, reset) ;
	
	initial
	begin
	$dumpfile ("design.vcd") ;
	$dumpvars (0,FSM_sa_94test) ;
	clk = 1'b1 ;
	end
	
	always
	#5 clk = ~clk ;
	
	initial
	begin
	#0 reset = 1'b1 ;
	#10 reset = 1'b0 ;x1=0;x2=0;
	#10 x1 = 1'b0 ; x2 = 1'b1;
	#10 x1 = 1'b1 ; x2 = 1'b0;
	#10 x1 = 1'b1 ; x2 = 1'b1;
	#10 x1 = 1'b0 ; x2 = 1'b1;
	#10 x1 = 1'b1 ; x2 = 1'b0;
	#10 x1 = 1'b1 ; x2 = 1'b1;
	#10 x1 = 1'b0 ; x2 = 1'b0;
	end
	initial
	#100 $stop;
	endmodule

FSM EDA OUTPUT