## VERILOG CODES

1. **RCA**

module FA(A,B,C,S,Cout);

input A,B,C;

output reg S,Cout;

always @(A or B or C)

begin

S=A^B^C;

Cout=(A&B)|(B&C)|(A&C);

end

endmodule

module rca(A,B,C,S,Co);

input [3:0]A,B;

input C;

output [3:0]S;

output Co;

wire [2:0]W;

FA fa1(A[0],B[0],C,S[0],W[0]);

FA fa2(A[1],B[1],W[0],S[1],W[1]);

FA fa3(A[2],B[2],W[1],S[2],W[2]);

FA fa4(A[3],B[3],W[2],S[3],Co);

endmodule

module rca\_tb();

reg [3:0]a,b;

reg c;

wire [3:0]s;

wire co;

integer i;

rca dut(a,b,c,s,co);

initial

begin

{a,b,c}=0;

for(i=501;i<511;i=i+1)

begin

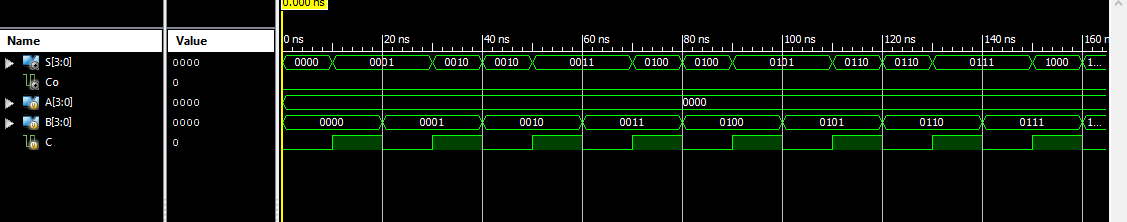
{a,b,c}=i;

#10;

end

end

endmodule



1. **MUX 8by1**

module mux8by1(i,s,Y);

input [7:0]i;

input [2:0]s;

output reg Y;

always @(i or s)

begin

case(s)

3'b000:Y=i[0];

3'b001:Y=i[1];

3'b010:Y=i[2];

3'b011:Y=i[3];

3'b100:Y=i[4];

3'b101:Y=i[5];

3'b110:Y=i[6];

3'b111:Y=i[7];

endcase

end

endmodule

module mux8by1\_tb();

reg [7:0]I;

reg [2:0]S;

wire y;

integer j;

mux8by1 DUT(I,S,y);

initial

begin

{I,S}=0;

for(j=0;j<8;j=j+1)

begin

S=j;

I[j]=1'b1;

#10;

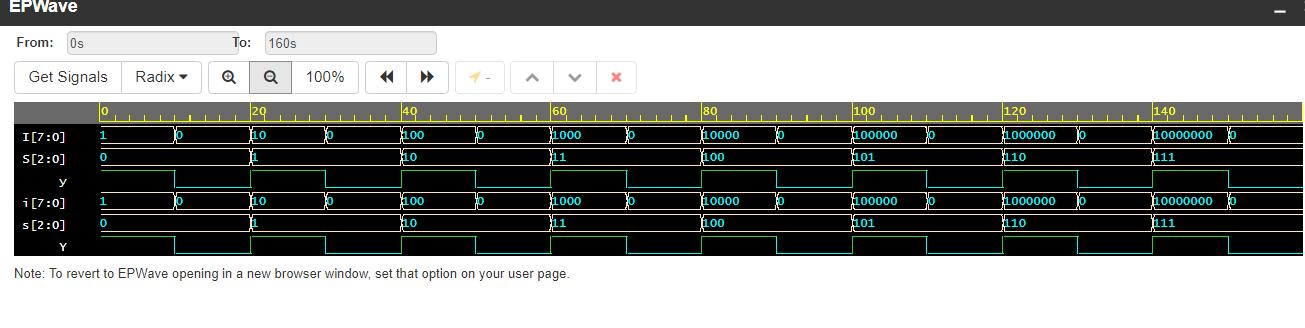
I[j]=1'b0;

#10;

end

end

endmodule



1. **BI DIRECTIONAL BUFFER**

module bibuffer(A,B,C);

inout A,B;

input C;

assign B=(C==1'b1)?A:1'bz;

assign A=(C==1'b0)?B:1'bz;

endmodule

module bibuffer\_tb();

reg c;

wire a,b;

integer i;

reg a\_tmp,b\_tmp;

bibuffer dut(a,b,c);

assign a=(c==1)?a\_tmp:1'bz;

assign b=(c==0)?b\_tmp:1'bz;

initial

begin

{c,a\_tmp,b\_tmp}=0;

for(i=0;i<8;i=i+1)

begin

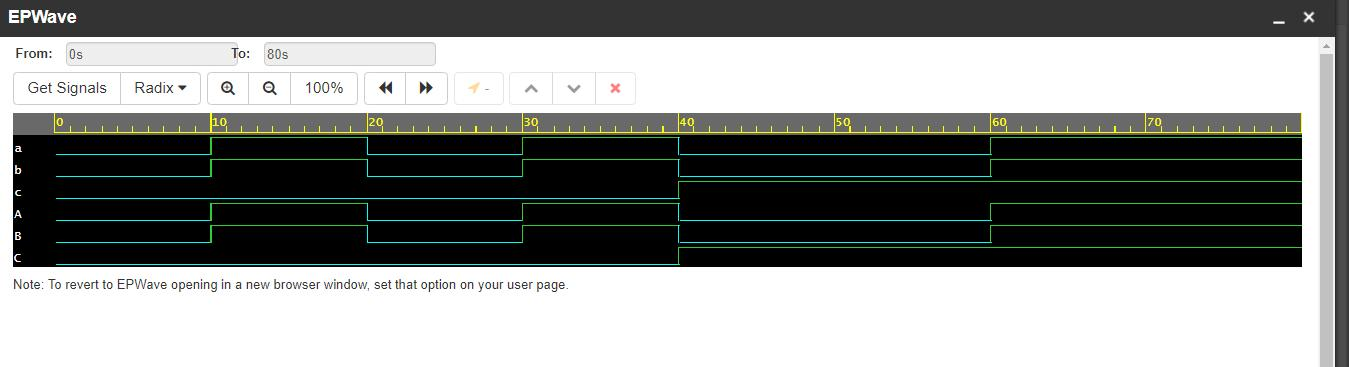
{c,a\_tmp,b\_tmp}=i;

#10;

end

end

endmodule



1. **PRIORITY ENCODER**

module pr\_enc(D,Y);

input [7:0]D;

output reg[2:0]Y;

always @(D)

begin

if(D[7])

Y=3'b111;

else if(D[6])

Y=3'b110;

else if(D[5])

Y=3'b101;

else if(D[4])

Y=3'b100;

else if(D[3])

Y=3'b011;

else if(D[2])

Y=3'b010;

else if(D[1])

Y=3'b001;

else if(D[0])

Y=3'b000;

else

Y=3'bzz;

end

endmodule

module pr\_enc\_tb();

reg [7:0]d;

wire [2:0]y;

integer j;

pr\_enc DUT(d,y);

initial

begin

{d}=0;

for(j=0;j<256;j=j+1)

begin

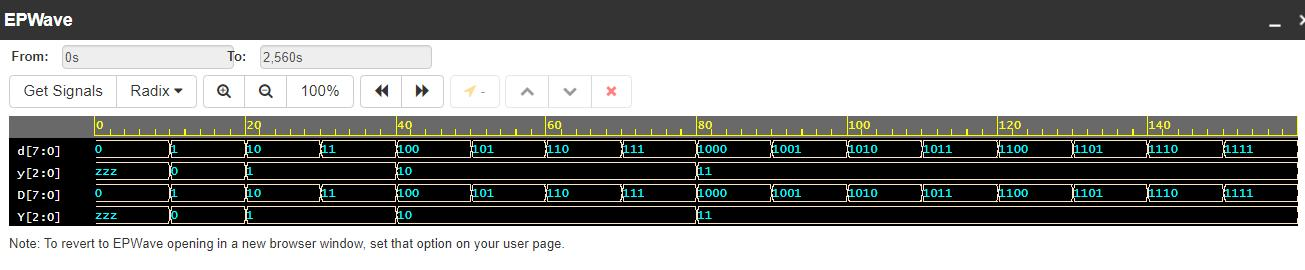
d=j;

#10;

end

end

endmodule



1. **T-FF**

module T\_FF(clk,T,rst,q,qb);

input clk,T,rst;

output reg q,qb;

always @(posedge clk)

begin

qb<=~q;

if(rst)

q<=0;

else if(T==1'b1)

q<=~q;

else

q<=q;

end

endmodule

module T\_FF\_tb();

reg clk,T,rst;

wire q;

integer j;

T\_FF uut(.clk(clk),.T(T),.rst(rst),.q(q));

always #5 clk=~clk;

task initialise;

begin

rst<=1'b0;

clk<=1'b0;

end

endtask

task rst1;

begin

@(negedge clk)

rst<=1'b1;

@(negedge clk)

rst<=1'b0;

end

endtask

task t\_tsk(input i);

begin

@(negedge clk)

T<=i;

end

endtask

initial

begin

initialise;

rst1;

t\_tsk(1);

t\_tsk(0);

t\_tsk(1);

rst1;

t\_tsk(0);

t\_tsk(1);

rst1;

$finish;

end

endmodule



1. **COUNTER (WITH LOAD)**

module counter4b(clk,din,ld,rst,d);

input clk,rst,ld;

input [3:0]din;

output reg [3:0]d;

always @(negedge clk)

begin

if(rst==1'b1)

d<=4'b0000;

else if(ld)

d<=din;

else

begin

if(d!=4'b1111)

d<=d+1;

else

d<=4'b0000;

end

end

endmodule

module counter4b\_tb();

reg clk,ld,rst;

reg [3:0] din;

wire [3:0] d;

counter4b uut (.clk(clk), .din(din), .ld(ld), .rst(rst), .d(d));

always #5 clk=~clk;

task initialise;

begin

rst<=0;

ld<=0;

clk<=0;

end

endtask

task load\_in(input [3:0]a);

begin

@(posedge clk)

ld<=1'b1;

din<=a;

@(posedge clk)

ld<=1'b0;

end

endtask

task rst\_tsk;

begin

@(posedge clk)

rst<=1;

@(posedge clk)

rst<=0;

end

endtask

initial

begin

initialise;

rst\_tsk;

din<=3'b000;

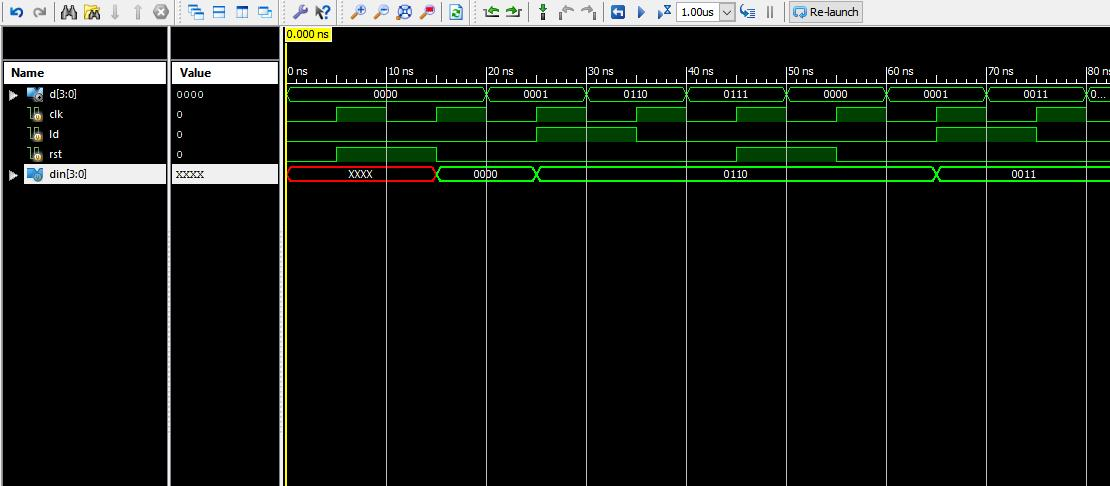
load\_in(3'b110);

rst\_tsk;

load\_in(3'b011);

end

endmodule



1. **ASYNCHRONOUS COUNTER (4-BIT)**

module T\_FF(clk,rst,T,q,qb);

input clk,T,rst;

output reg q;

output qb;

assign qb=~q;

always @(posedge clk)

begin

if(rst)

q<=0;

else if(T==1'b1)

q<=~q;

else

q<=q;

end

endmodule

module async\_count(clk,rst,q,qb);

input clk,rst;

output [3:0]q;

wire [3:0]w;

T\_FF t1(clk,rst,1'b1,q[0],w[0]);

T\_FF t2(w[0],rst,1'b1,q[1],w[1]);

T\_FF t3(w[1],rst,1'b1,q[2],w[2]);

T\_FF t4(w[2],rst,1'b1,q[3],w[3]);

endmodule

module Asynch\_counter\_tb();

reg clk,rst,T;

wire [3:0]q;

Asynch\_counter uut(.clk(clk),.rst(rst),.T(T),.q(q));

always #5 clk=~clk;

initial

begin

T=0;

clk=0;

rst=0;

end

task rst\_tsk;

begin

@(negedge clk)

rst<=1;

@(negedge clk)

rst<=0;

end

endtask

initial

begin

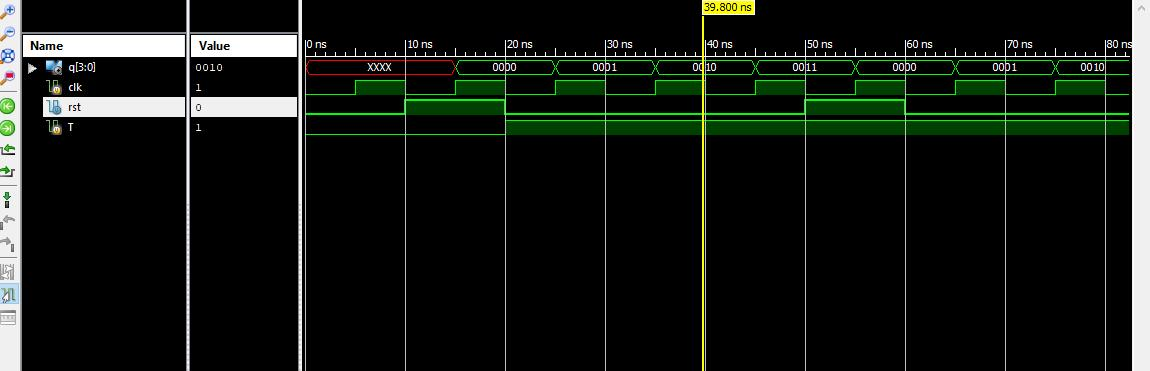
rst\_tsk;

T=1'b1;

#30 rst\_tsk;

end

endmodule



1. **SINGLE PORT RAM**

module ram\_1(clk,data,addr,we,re);

input clk,re,we;

input [3:0]addr;

inout [7:0]data;

reg [7:0]temp;

reg [7:0] MEM [15:0];

assign data=(re && !we)?temp:8'hzz;

always @(posedge clk)

begin

if(we && !re)

MEM[addr]<=data;

else if(re && !we)

temp<=MEM[addr];

end

endmodule

module ram\_tb();

reg clk,we,re;

reg [3:0]addr;

wire [7:0]data;

reg [7:0]temp;

ram\_1 ram(clk,data,addr,we,re);

integer i,j;

always

begin

clk=0;

#20;

clk=1;

#20;

end

assign data=(we && !re)?temp:8'hzz;

task initialise;

begin

temp<=8'h00;

clk<=0;

end

endtask

task write(input [7:0]A,input [3:0]B);

begin

@(negedge clk)

we<=1'b1;

re<=1'b0;

temp<=A;

addr<=B;

end

endtask

task read(input [3:0]C);

begin

@(negedge clk)

we<=1'b0;

re<=1'b1;

addr<=C;

end

endtask

initial

begin

initialise();

for(i=0;i<=15;i=i+1)

begin

write(i,i);

#10;

end

for(j=0;j<=15;j=j+1)

begin

read(j);

initialise();

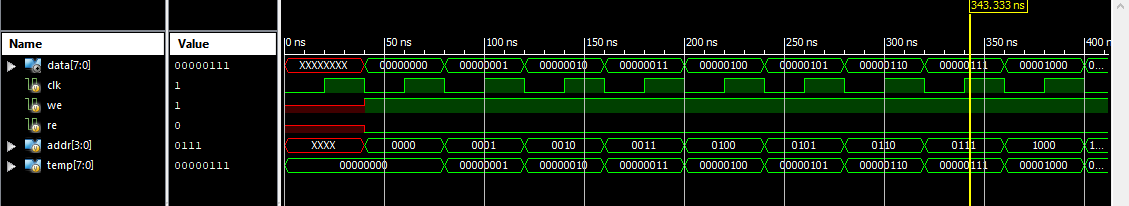
#10;

end

$finish;

end

endmodule

****