### Verilog Codes

Name: Thota Bhagyasri Annapurna

**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

**Test bench:**

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();

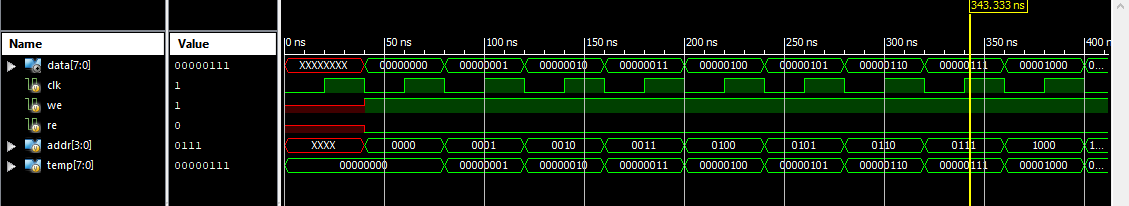
#10;

end

$finish;

end

endmodule

****

**2.DUAL PORT RAM**

module ram\_2(clk,din,w\_addr,dout,r\_addr,we,re);

input clk,re,we;

input [3:0]w\_addr,r\_addr;

input [7:0]din;

output reg [7:0]dout;

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

always @(posedge clk)

begin

if(we)

MEM[w\_addr]<=din;

if(re)

dout<=MEM[r\_addr];

end

endmodule

**Test bench:**

module ram\_2\_tb;

reg clk,we,re;

reg [7:0] din;

reg [3:0] w\_addr;

reg [3:0] r\_addr;

wire [7:0] dout;

ram\_2 uut (.clk(clk), .din(din), .w\_addr(w\_addr), .dout(dout), .r\_addr(r\_addr), .we(we), .re(re));

integer i,j;

always

begin

clk=0;

#10;

clk=1;

#10;

end

task initialise;

begin

clk<=0;

w\_addr<=0;

r\_addr<=0;

we<=0;

re<=0;

end

endtask

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

begin

@(negedge clk)

we<=1'b1;

re<=1'b0;

din<=A;

w\_addr<=B;

end

endtask

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

begin

@(negedge clk)

we<=1'b0;

re<=1'b1;

r\_addr<=C;

end

endtask

initial

begin

initialise;

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

begin

write(i,i);

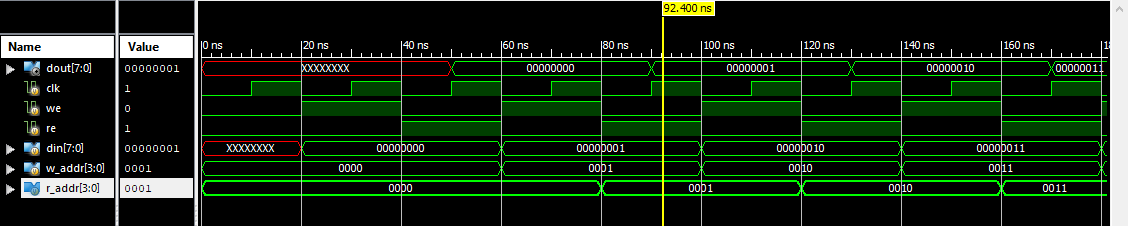
read(i);

end

$finish;

end

endmodule



**3.TRUE DUAL PORT RAM**

module truedp\_ram(clk,data0,data1,ad0,ad1,we0,we1,re0,re1);

input clk,we0,we1,re0,re1;

input [3:0]ad0,ad1;

inout [7:0]data0,data1;

reg [7:0]temp0,temp1;

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

assign data0=(re0 && !we0)?temp0:8'hzz;

assign data1=(re1 && !we1)?temp1:8'hzz;

always @(posedge clk)

begin

if(we0 && !re0)

MEM(ad0)<=data0;

else if(re0 && !we0)

temp0<=MEM(ad0);

end

always @(negedge clk)

begin

if(we1 && !re1)

MEM(ad1)<=data1;

else if(re1 && !we1)

temp1<=MEM(ad1);

end

endmodule

**Test bench:**

module truedp\_ram\_tb;

reg clk;

reg [3:0]ad\_0;

reg [3:0]ad\_1;

reg we\_0,we\_1,re\_0,re\_1;

wire [7:0]data\_0;

wire [7:0]data\_1;

truedp\_ram uut(.clk(clk), .data\_0(data\_0), .data\_1(data\_1), .ad\_0(ad\_0), .ad\_1(ad\_1), .we\_0(we\_0), .we\_1(we\_1), .re\_0(re\_0), .re\_1(re\_1));

reg [7:0]temp0,temp1;

assign data\_0=(re\_0 && !we\_0)?temp0:8'hzz;

assign data\_1=(re\_1 && !we\_1)?temp1:8'hzz;

integer i,j;

always

begin

clk=0;

#5;

clk=1;

#5;

end

task initialise;

begin

{we\_0,we\_1,re\_0,re\_1}<=0;

temp0<=0;

temp1<=0;

ad\_0<=0;

ad\_1<=0;

end

endtask

task write;

begin

@(negedge clk)

we\_0<=1'b1;

re\_0<=1'b0;

@(posedge clk)

we\_1<=1'b1;

re\_1<=1'b0;

end

endtask

task read;

begin

@(negedge clk)

we\_0<=1'b0;

re\_0<=1'b1;

@(posedge clk)

we\_1<=1'b0;

re\_1<=1'b1;

end

endtask

task stimulus\_0(input[3:0]a0,input[7:0]d0);

begin

@(negedge clk)

ad\_0<=a0;

temp0<=d0;

end

endtask

task stimulus\_1(input[3:0]a1,input[7:0]d1);

begin

@(posedge clk)

ad\_1<=a1;

temp1<=d1;

end

endtask

initial

begin

initialise;

write;

end

initial

fork

#20 for(i=0;i<16;i=i+2)

begin

stimulus\_0(i,$random);

end

#20 for(j=1;j<16;j=j+2)

begin

stimulus\_1(j,$random);

end

join

initial

begin

#170 read;

end

initial

fork

#200 for(i=0;i<16;i=i+2)

begin

stimulus\_0(i,i);

end

#200 for(j=1;j<16;j=j+2)

begin

stimulus\_1(j,j);

end

join

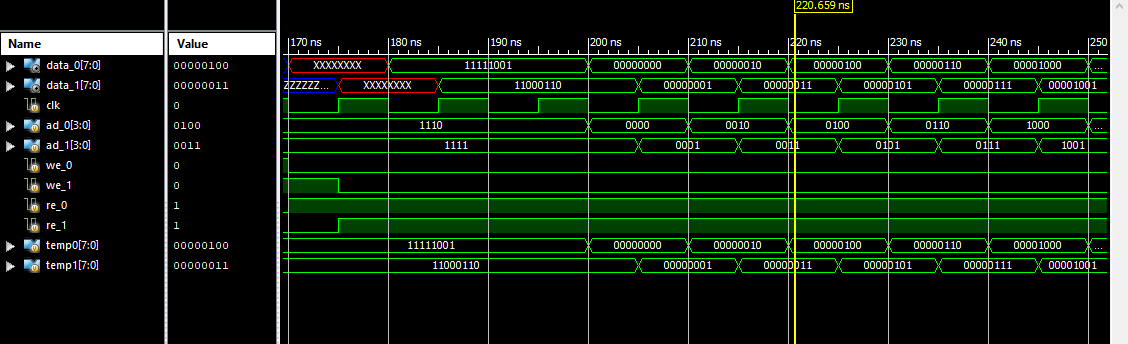
initial

begin

#400 $finish;

end

endmodule



1. **FIFO**

module fifo(dout,din,full,empty,clk,rst,we,re);

parameter dw=8; //width

parameter aw=4;

parameter rd=16; //depth

output reg [dw-1:0]dout;

output full,empty;

input [dw-1:0]din;

input we,re,clk,rst;

reg [aw-1:0]wr\_pointer;

reg [aw-1:0]rd\_pointer;

reg [aw-1:0]status\_counter;

reg [dw-1:0]dram;

always @(posedge clk,posedge rst)

begin

if(rst)

wr\_pointer=0;

else

wr\_pointer=wr\_pointer+1;

end

always @(posedge clk,posedge rst)

begin

if(rst)

rd\_pointer=0;

else

rd\_pointer=rd\_pointer+1;

end

always @(posedge clk,posedge rst)

begin

if(rst)

dout<=0;

else

begin

if(re && !we)

dout<=dram;

end

end

always @(posedge clk,posedge rst)

begin

if(rst)

status\_counter=0;

else if((we && !re) && (status\_counter != rd))

status\_counter=status\_counter+1;

else if((re && !we) && (status\_counter !=rd))

status\_counter=status\_counter-1;

end

assign full=(status\_counter==(rd));

assign empty=(status\_counter==0);

dualport\_ram one(clk,wr\_pointer,rd\_pointer,we,re,din,dout);

endmodule

module dualport\_ram(clk,w\_add,r\_add,we,re,din,dout);

input clk,re,we;

input [3:0]w\_add,r\_add;

input [7:0]din;

output reg [7:0]dout;

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

always @(posedge clk)

begin

mem[w\_add]<=we?din:8'bz;

dout<=re?mem[r\_add]:8'bz;

end

endmodule

**Test Bench:**

module fifo\_tb;

reg [7:0]din;

reg clk;

reg rst;

reg we;

reg re;

wire [7:0]dout;

wire full;

wire empty;

fifo uut (.dout(dout), .din(din), .full(full), .empty(empty), .clk(clk), .rst(rst), .we(we), .re(re));

always

begin

clk=0;

#10;

clk=1;

#10;

end

task initialise;

begin

rst<=0;

we<=0;

re<=0;

din<=0;

end

endtask

task data(input[7:0]a)

begin

@(negedge clk)

din<=a;

end

endtask

initial

begin

initialise;

#10;

rst<=1;

#10;

rst<=0;

#10;

we<=1;

repeat(16)

begin

data($random)

end

we<=0;

#50 re<=1;

#150 re<=0;

$finish;

end

endmodule

1. **FSM (Moore 1110 )**

module fsm1(din,dout,rst,clk);

input din,rst,clk;

output dout;

parameter idle=3'b000,s1=3'b001,s11=3'b010,s111=3'b011,s1110=3'b100;

reg [2:0] cst,nst;

always @(posedge clk, posedge rst)

begin

if(rst)

cst<=idle;

else

cst<=nst;

end

always @(cst,din)

begin

case(cst)

idle:

if(din==0)

nst<=idle;

else

nst<=s1;

s1:

if(din==0)

nst<=idle;

else

nst<=s11;

s11:

if(din==0)

nst<=idle;

else

nst<=s111;

s111:

if(din==0)

nst<=s1110;

else

nst<=s111;

s1110:

if(din==0)

nst<=idle;

else

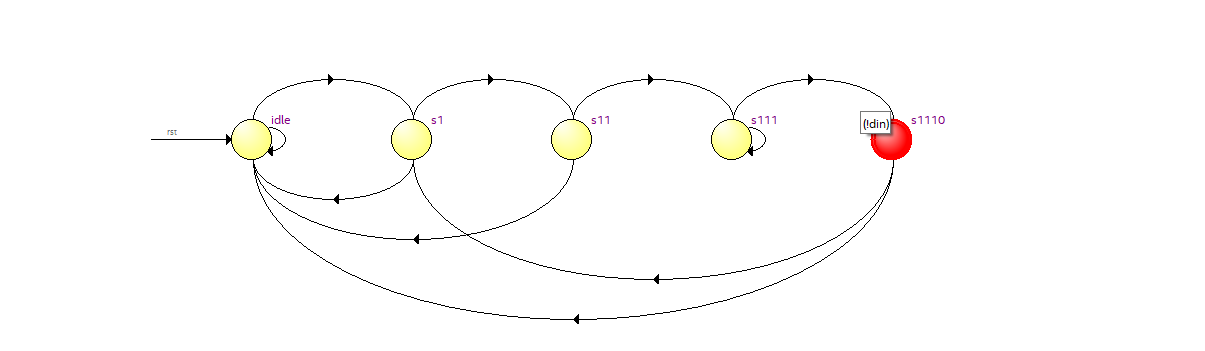
nst<=s1;

endcase

end

assign dout=(cst==s1110)?1:0;

endmodule



**Test Bench:**

module fsm1\_tb;

reg clk,rst,din;

wire dout;

fsm1 uut(din,dout,rst,clk);

initial

begin

$dumpfile("fsm1\_tb.vcd");

$dumpvars(0);

end

always

begin

clk=0;

#10;

clk=1;

#10;

end

task initialise;

begin

{clk,rst,din}=0;

end

endtask

task d\_in(input a);

begin

@(posedge clk)

din<=a;

end

endtask

initial

begin

initialise;

rst<=1'b1;

#10 rst<=1'b0;

d\_in(0);

d\_in(1);

d\_in(1);

d\_in(1);

d\_in(1);

d\_in(0);

d\_in(0);

d\_in(1);

d\_in(1);

d\_in(1);

d\_in(1);

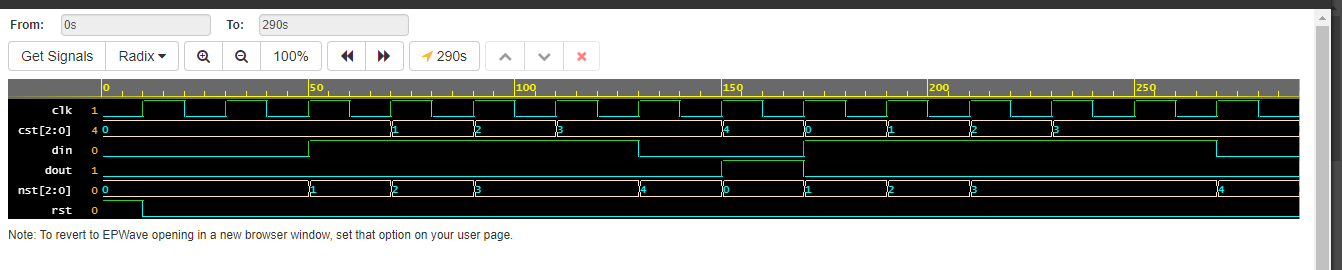
d\_in(1);

d\_in(0);

#20 $finish;

end

endmodule



1. **if the input is 0 output is 0 until the next state is greater than present state .if the state has the input 1 output is changed to 1.The output is same as input until the present state is greater than previous state the inputs available only from 0…3.**

Input: 0010120112130123

Output: 0011122222233333

module fsm2ip(din,dout,clk,rst);

input clk,rst;

input [1:0] din;

output [1:0] dout;

reg [2:0] cst,nst;

parameter idle=3'b000,s0=3'b001,s1=3'b010,s2=3'b011,s3=3'b100;

always @(posedge clk,posedge rst)

begin

if(rst)

cst<=idle;

else

cst<=nst;

end

always

begin

case(cst)

idle:

if(din==2'b00)

nst<=s0;

else if(din==2'b01)

nst<=s1;

else if(din==2'b10)

nst<=s2;

else

nst<=s3;

s0:

if(din==0)

nst<=s0;

else if(din==2'b01)

nst<=s1;

else if(din==2'b10)

nst<=s2;

else

nst<=s3;

s1:

if(din==0)

nst<=s1;

else if(din==2'b01)

nst<=s1;

else if(din==2'b10)

nst<=s2;

else

nst<=s3;

s2:

if(din==0)

nst<=s2;

else if(din==2'b01)

nst<=s2;

else if(din==2'b10)

nst<=s2;

else

nst<=s3;

s3:

if(din==0)

nst<=s3;

else if(din==2'b01)

nst<=s3;

else if(din==2'b10)

nst<=s3;

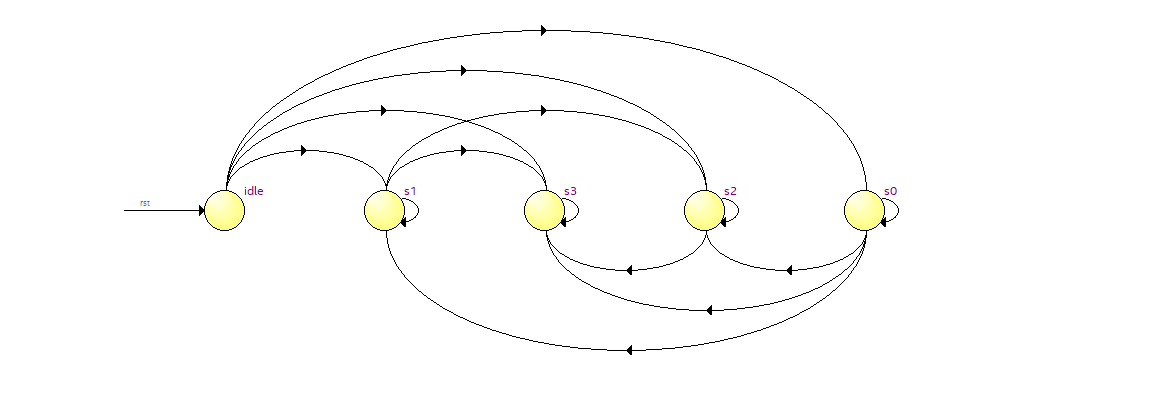
else

nst<=s3;

endcase

end

endmodule



1. **Moore(0101)**

module fsm2(din,dout,rst,clk);

input rst,clk,din;

output reg dout;

parameter idle=3'b000,s0=3'b001,s01=3'b010,s010=3'b011,s0101=3'b100;

reg [2:0]cst,nst;

always @(posedge clk , posedge rst)

begin

if(rst==1)

cst<=idle;

else

cst<=nst;

end

always @(cst,din)

begin

case(cst)

begin

idle:

if(din==0)

nst<=s0;

else

nst<=idle;

s0:

if(din==0)

nst<=s0;

else

nst<=s01;

s01:

if(din==0)

nst<=s010;

else

nst<=idle;

s010:

if(din==0)

nst<=s0;

else

nst<=s0101;

s0101:

if(din==0)

nst<=s010;

else

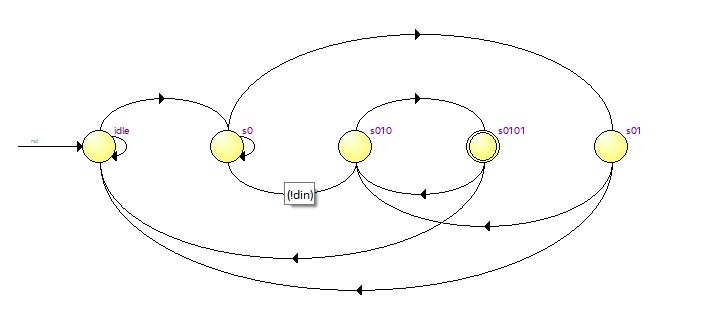
nst<=idle;

endcase

end

assign dout=(cst==s0101)?1:0;

endmodule



**Test bench:**

module fsm2\_tb;

reg clk,rst,din;

wire dout;

fsm2 uut(din,dout,rst,clk);

initial

begin

$dumpfile("fsm2\_tb.vcd");

$dumpvars(0);

end

always

begin

clk=0;

#10;

clk=1;

#10;

end

task initialise;

begin

{clk,rst,din}=0;

end

endtask

task d\_in(input a);

begin

@(posedge clk)

din<=a;

end

endtask

initial

begin

initialise;

rst<=1'b1;

#10 rst<=1'b0;

d\_in(0);

d\_in(1);

d\_in(0);

d\_in(1);

d\_in(1);

d\_in(0);

d\_in(1);

d\_in(1);

d\_in(0);

d\_in(1);

d\_in(0);

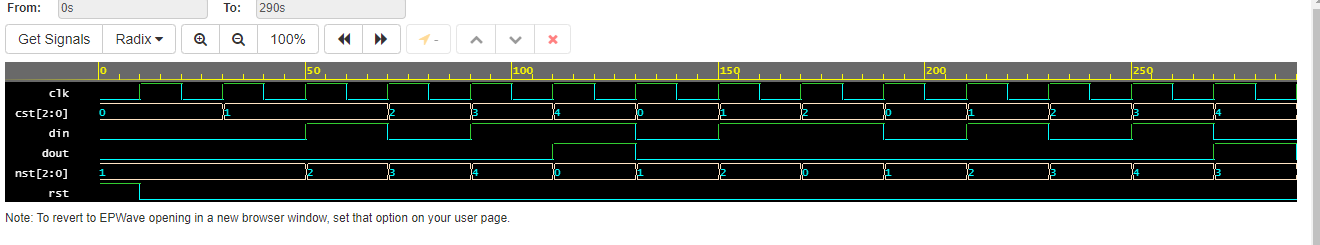
d\_in(1);

d\_in(0);

#20 $finish;

end

endmodule



1. **MELAY**

module fsmmelay(din,dout,rst,clk);

input clk,rst,din;

output reg dout;

reg [1:0] cst,nst;

parameter s0=2'b00,s1=2'b01,s2=2'b10,s3=2'b11;

always @(posedge clk,posedge rst)

begin

if(rst)

cst<=s0;

else

cst<=nst;

end

always @(cst,din)

begin

case(cst)

s0:

if(din==0)

nst<=s1;

else

nst<=s0;

s1:

if(din==0)

nst<=s1;

else

nst<=s2;

s2:

if(din==0)

nst<=s3;

else

nst<=s0;

s3:

if(din==0)

nst<=s1;

else

nst<=s2;

endcase

end

always @(posedge clk)

begin

case(cst)

s0: if(din==0 || din==1)

dout<=0;

s1: if(din==0 || din==1)

dout<=0;

s2: if(din==0 || din==1)

dout<=0;

s3: if(din==0)

dout<=1;

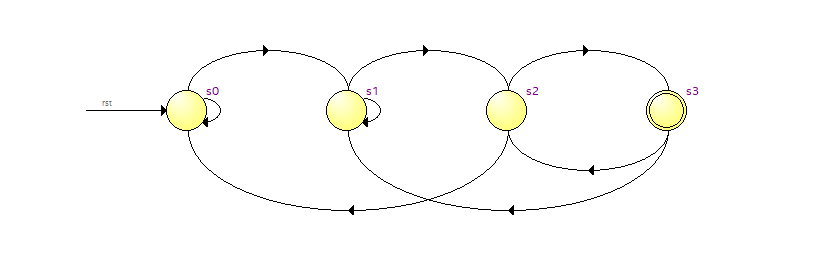
else

dout<=0;

endcase

end

endmodule



**Test bench:**

module fsmmelay\_tb;

reg clk,rst,din;

wire dout;

fsmmelay uut(din,dout,rst,clk);

initial

begin

$dumpfile("fsmmelay\_tb.vcd");

$dumpvars(0);

end

always

begin

clk=0;

#10;

clk=1;

#10;

end

task initialise;

begin

{clk,rst,din}=0;

end

endtask

task d\_in(input a);

begin

@(posedge clk)

din<=a;

end

endtask

initial

begin

initialise;

rst<=1'b1;

#10 rst<=1'b0;

d\_in(0);

d\_in(1);

d\_in(0);

d\_in(0);

d\_in(1);

d\_in(0);

d\_in(0);

d\_in(1);

d\_in(1);

d\_in(0);

d\_in(1);

d\_in(0);

d\_in(0);

#20 $finish;

end

endmodule

