Open

dp_ram.v
~/verilog/ram

```
module dp_ram (clk,wr_addr,rd_addr,data_in,data_out,cs,we,re,oe);

parameter ADDR_WIDTH =4;
parameter DATA_WIDTH =8;
parameter DEPTH =16;

input clk,cs,we,re,oe;
input [ADDR_WIDTH-1:0] rd_addr;
input [ADDR_WIDTH-1:0] wr_addr;

input [DATA_WIDTH-1:0] data_in;
input [DATA_WIDTH-1:0] data_out;

reg [DATA_WIDTH-1:0] tmp_data;
reg [DATA_WIDTH-1:0] mem[DEPTH];

always @(posedge clk) begin
if(cs & we)
   mem[wr_addr] <= data_in;
end

always @(posedge clk) begin
if(cs & re)
   tmp_data <=mem[rd_addr];
end

assign data_out = cs & oe & re ? tmp_data : 'hz;

endmodule
```

Open

dp_ram_test.v
~/verilog/ram

```
include "dp_ram.v"
module dp_ram_test;

parameter ADDR_WIDTH =4;
parameter DATA_WIDTH =8;
parameter DEPTH =16;

reg clk,cs,we,re,oe;
reg [ADDR_WIDTH-1:0] wr_addr;
reg [ADDR_WIDTH-1:0] rd_addr;
wire [DATA_WIDTH-1:0] data_out;
wire [DATA_WIDTH-1:0] data_in;
reg [DATA_WIDTH-1:0] tb_data;

dp_ram inst(clk,wr_addr,rd_addr,data_in,data_out,cs,we,re,oe);

always #10 clk = ~clk;
assign data_in= we ? tb_data : 'hz;

initial begin
$monitor("Time=%0t data_in=%0d mem=[%0d] <==> data_out=%0d mem=[%0d]",$time,data_in,wr_addr,data_out,rd_addr);
end

initial begin
 {clk,cs,we,re,wr_addr,rd_addr,tb_data,oe} <=0;

#10
#20 we =1; cs=1; wr_addr =1; tb_data=10;
#20 we =1; cs=1; wr_addr =2; tb_data=20;
#20 we =1; cs=1; wr_addr =3; tb_data=30;
#20 we =1; cs=1; wr_addr =4; tb_data=40;
#20 we =1; cs=1; wr_addr =5; tb_data=50;
#20 we =1; cs=1; wr_addr =6; tb_data=60;
```

```
#20 we =1; cs=1; wr_addr =3; tb_data=30;
#20 we =1; cs=1; wr_addr =4; tb_data=40;
#20 we =1; cs=1; wr_addr =5; tb_data=50;
#20 we =1; cs=1; wr_addr =6; tb_data=60;
#20 we =1; cs=1; wr_addr =7; tb_data=70;
#20 we =1; cs=1; wr_addr =8; tb_data=80;
#20 we =1; cs=1; wr_addr =9; tb_data=90;
#20 we =1; cs=1; wr_addr =10; tb_data=100;
#20 we =1; cs=1; wr_addr =11; tb_data=110;
#20 we =1; cs=1; wr_addr =12; tb_data=120;
#20 we =1; cs=1; wr_addr =13; tb_data=130;
#20 we =1; cs=1; wr_addr =14; tb_data=140;
#20 we =1; cs=1; wr_addr =15; tb_data=150;
#20 we =1; cs=1; wr_addr =5; tb_data=5;
#20
#200;
end

initial begin
#50
#20 re=1; cs=1; oe=1;

#20 rd_addr=1;
#20 rd_addr=2;
#20 rd_addr=3;
#20 rd_addr=4;
#20 rd_addr=5;
#20 rd_addr=6;
#20 rd_addr=7;
#20 rd_addr=8;
#20 rd_addr=9;
#20 rd_addr=10;
```

Open

dp_ram_test.v
~/verilog/ram

```
#20 we =1; cs=1; wr_addr =14; tb_data=140;
#20 we =1; cs=1; wr_addr =15; tb_data=150;
#20 we =1; cs=1; wr_addr =5; tb_data=5;
#20
#200;
end

initial begin
#50
#20 re=1; cs=1; oe=1;

#20 rd_addr=1;
#20 rd_addr=2;
#20 rd_addr=3;
#20 rd_addr=4;
#20 rd_addr=5;
#20 rd_addr=6;
#20 rd_addr=7;
#20 rd_addr=8;
#20 rd_addr=9;
#20 rd_addr=10;
#20 rd_addr=11;
#20 rd_addr=12;
#20 rd_addr=13;
#20 rd_addr=14;
#20 rd_addr=15;
#40 rd_addr=0;
#40

$finish;

end
endmodule
```

Open

sp_ram.v
~/verilog/ram

```
module sp_ram (clk,addr,data,cs,we,oe);

parameter ADDR_WIDTH =4;
parameter DATA_WIDTH =8;
parameter DEPTH =16;

input clk,cs,we,oe;
input [ADDR_WIDTH-1:0] addr;
input [DATA_WIDTH-1:0] data;

reg [DATA_WIDTH-1:0] tmp_data;
reg [DATA_WIDTH-1:0] mem[DEPTH];

always @(posedge clk) begin
if(cs & we)
   mem[addr] <= data;
end

always @(posedge clk) begin
if(cs & !we)
   tmp_data <=mem[addr];
end

assign data = cs & oe & !we ? tmp_data : 'hz;

endmodule
```

Open

sp_ram_test.v
~/verilog/ram

```
`include "sp_ram.v"
module sp_ram_test;

parameter ADDR_WIDTH =4;
parameter DATA_WIDTH =8;
parameter DEPTH =16;

reg clk,cs,we,oe;
reg [ADDR_WIDTH-1:0] addr;
wire [DATA_WIDTH-1:0] data;

reg [DATA_WIDTH-1:0] tb_data;

sp_ram inst(clk,addr,data,cs,we,oe);

always #10 clk = ~clk;

assign data= !oe ? tb_data : 'hz;

initial begin
$monitor("Time=%0t data=%0d mem=[%0d]",$time,data,addr);
 {clk,cs,we,addr,tb_data,oe} <=0;

#10
$display("Time=>Writing in Memory");
#20 we =1; cs=1; addr =1; tb_data=10;
#20 we =1; cs=1; addr =2; tb_data=20;
#20 we =1; cs=1; addr =3; tb_data=30;
#20 we =1; cs=1; addr =4; tb_data=40;
#20 we =1; cs=1; addr =5; tb_data=50;
#20 we =0; cs=1; oe=1;

$display("Time=>**************************");
```

Applications Places Text Editor

Open

sp_ram_test.v
~/verilog/ram

```
sp_ram inst(clk,addr,data,cs,we,oe);

always #10 clk = ~clk;

assign data= !oe ? tb_data : 'hz;

initial begin
$monitor("Time=%0t data=%0d mem=[%0d]",$time,data,addr);
 {clk,cs,we,addr,tb_data,oe} <=0;

#10
$display("Time=>Writing in Memory");
#20 we =1; cs=1; addr =1; tb_data=10;
#20 we =1; cs=1; addr =2; tb_data=20;
#20 we =1; cs=1; addr =3; tb_data=30;
#20 we =1; cs=1; addr =4; tb_data=40;
#20 we =1; cs=1; addr =5; tb_data=50;
#20 we =0; cs=1; oe=1;

$display("Time=>*****************************");
$display("Time=>Reading from Memory");

#20 addr=1;
#20 addr=2;
#20 addr=3;
#20 addr=4;
#20 addr=5;
#40

$finish;

end
endmodule
```

Open

mealy_practice.v
~/verilog/m_test

```
module mealy_practice (input clk,reset,in, output reg out);
parameter A =0, B=1, C=2, D=3;
bit [1:0] state;
always @(posedge clk or posedge reset) begin
if(reset) begin state <=A; out=0; end
else begin

case(state)
 A: begin
 if(in==1) begin state =B ; out=0; end
else begin state =A ; out=0; end
 end
B: begin
 if(in==1) begin state =C ; out=0; end
else begin state =A ; out=0; end
end

C: begin
 if(in==1) begin state =D ; out=0; end
 else begin state =A ; out=0; end
 end

D: begin
 if(in==0) begin state =A ; out=1; end
 else begin state =D ; out=0;end
 end
default: begin state =A ; out=0; end

endcase
end
end
endmodule
```

Open

*mealy_practice_test.v
~/verilog/m_test

```
`include "mealy_practice.v"
module mealy_practice_test;
reg clk,reset,in;
wire out;
always #10 clk = ~ clk;
 mealy_practice inst(.clk(clk), .reset(reset), .in(in), .out(out));
initial begin
$monitor("Time=%0t in=%0b out=%0b",$time,in,out);
clk=0;
#10 reset =1;
#20 reset =0;
#20 in=1;
#20 in=1;
#20 in=1;
#20 in=1;
#20 in=0;
#20 in=0;
#20 in=0;
#20 in=1;
#20 in=1;
#20 in=1;
#20 in=1;
#20 in=0;
#20 in=1;
#20 in=1;
#20 in=1;
#20 in=1;
#20 in=0;
#20
$finish;
end
endmodule
```

Open

*pipo_test.v
~/verilog/register/pipo

```
`include "pipo.v"

module pipo_test;

 reg clk;
 reg [3:0] d_in;
 wire [3:0] d_out;

 pipo inst(.clk(clk), .d_in(d_in), .d_out(d_out));

  always #10 clk = ~clk;

 initial begin

  $monitor("Time=%0t, d_in=%b, d_out=%b ",$time,d_in,d_out);


  clk=0;
  #10 d_in=4'b1010;
  #20
  #20 d_in=4'b1111;
  #20
  #20 d_in=4'b0000;
  #20
  #20 d_in=4'b1111;
  #20

 #20
 $finish;
end
endmodule
```

Open

pipo.v
~/verilog/register/pipo

```
module pipo(output reg [3:0] d_out, input wire clk,[3:0] d_in);


  always @(posedge clk) begin
  d_out=d_in;

  end


endmodule
```

Open

piso.v
~/verilog/register/piso

```
module piso(output reg d_out, input wire clk,load,[3:0] d_in);

reg [3:0] tmp;


  always @(posedge clk) begin
    if(load)
     tmp <=d_in;
    else
      tmp <= {1'b0,tmp[3:1]};


  end
   assign d_out=tmp[0];


endmodule
```

Applications Places Text Editor

Open

piso_test.v

~/verilog/register/piso

```
`include "piso.v"
module piso_test;

 reg clk;
 reg [3:0] d_in,load;
 wire d_out;

 piso inst(.clk(clk), .d_in(d_in), .d_out(d_out), .load(load));

  always #10 clk = ~clk;

 initial begin

  $monitor("Time=%0t, d_in=%b, d_out=%b ",$time,d_in,d_out);


  clk=0;
  #10 load=1'b1;d_in=4'b1010;
  #10 load=1'b0;

 #90 load=1'b1;d_in=4'b1111;
 #10 load=1'b0;

 #70 load=1'b1;d_in=04'b0000;
 #10 load=1'b0;
#70 load=1'b1;d_in=4'b1111;
 #10 load=1'b0;

 #80
 $finish;
end
endmodule
```

Open

```
module sipo(output reg [3:0] d_out, input wire clk,d_in);

  always @(posedge clk) begin
  // d_out= {d_out[2:0],d_in};  //left shift

  d_out= {d_in,d_out[3:1]}; //right shift
 end
endmodule
```

```
module sipo(output reg [3:0] d_out, input wire clk,d_in);

  always @(posedge clk) begin
  // d_out= {d_out[2:0],d_in};  //left shift

  d_out= {d_in,d_out[3:1]}; //right shift
 end
endmodule
```

Open

sipo_test.v
~/verilog/register/sipo

```
`include "sipo.v"
module sipo_test;


 reg clk,d_in;
 wire [3:0]d_out;


 sipo inst(.clk(clk), .d_in(d_in), .d_out(d_out));

 always #10 clk = ~clk;

 initial begin


 $monitor("Time=%0t, d_in=%b d_out=%b" ,$time,d_in,d_out,);
  clk=0;
  #10 d_in=1'b1;
  #20 d_in=1'b0;
  #20 d_in=1'b1;
  #20 d_in=1'b0;


  #20


 $finish;

end

endmodule
```

```
module siso(output reg d_out, input wire clk,d_in);
   reg [3:0] tmp;
  always @(posedge clk) begin

   tmp= {tmp[2:0],d_in}; //right shift
  // tmp= {d_in,tmp[3:1]}; //left shift
 d_out=tmp[3];


 end
endmodule
```

Open

siso_test.v

~/verilog/register/siso

```
`include "siso.v"
module siso_test;


 reg clk,d_in;
 wire d_out;


 siso inst(.clk(clk), .d_in(d_in), .d_out(d_out));

 always #10 clk = ~clk;

 initial begin


 $monitor("Time=%0t, d_in=%b d_out=%b" ,$time,d_in,d_out,);
  clk=0;
  #10 d_in=1'b1;
  #20 d_in=1'b0;
  #20 d_in=1'b1;
  #20 d_in=1'b0;
  #20 d_in=1'b1;
  #20 d_in=1'b1;
  #20 d_in=1'b0;

  #20


 $finish;
end

endmodule
```

Open

comparator.v
~/verilog/comparator

```
module comparator(input[3:0]a,b, output lt,gt,eq);


always @* begin

if(a>b) begin
 gt=1; lt=
end else if (a<b) begin
 a_lt_b = 1;
end else begin
a_eq_b = 1;

end
end


endmodule
```

Applications Places Text Editor

Open

t_ff.v
~/verilog/flipflop/t_ff

```
module t_ff(clk,t,q);

 output reg q=1;
 input wire clk,t;


 always @(posedge clk)begin
 if(t)
   q<= ~q;
  else
   q<=q;


 end

endmodule
```

```
`include "t_ff.v"
module t_ff_test;


 reg clk,t;
 wire q;


 t_ff inst(.clk(clk), .t(t) , .q(q));

 always #10 clk = ~clk;

 initial begin


 $monitor("Time=%0t, t=%b q=%b" ,$time,t,q);
  clk=0;
  #10 t=0;
  #20 t=1;
  #20 t=0;
  #20 t=1;
  #20 t=0;

  #20


 $finish;

end

endmodule
```

Applications Places Text Editor

Open

d_ff.v
~/verilog/flipflop/d_ff

```
module d_ff(clk,d,q);

  output reg q;
  input wire clk,d;


  always @(posedge clk)begin

  q<=d;


  end

endmodule
```

Open

d_ff_test.v

~/verilog/flipflop/d_ff

```
`include "d_ff.v"
module d_ff_test;


 reg clk,d;
 wire q;


 d_ff inst(.clk(clk), .d(d) , .q(q));

 always #10 clk = ~clk;

 initial begin


 $monitor("Time=%0t, d=%b q=%b" ,$time,d,q);
  clk=0;
  #10 d=0;
  #20 d=1;
  #20 d=0;
  #20 d=1;
  #20 d=0;

  #20


 $finish;

end

endmodule
```

Open

jk_ff.v
~/verilog/flipflop/jk_ff

```
module jk_ff(clk,j,k,q);

  output reg q;
  input wire clk,j,k;


  always @(posedge clk)begin



  case({j,k})
    2'b00: q<=q;
    2'b01: q<=1'b0;
    2'b10: q<=1'b1;
    2'b11: q<= ~q;
    default : q<=q;

  endcase


  end

endmodule
```

Open

jk_ff_test.v
~/verilog/flipflop/jk_ff

```
include "jk_ff.v"
module jk_ff_test;


reg clk,j,k;
wire q;


jk_ff inst(.clk(clk), .j(j), .k(k), .q(q));

always #10 clk = ~clk;

initial begin

  $monitor("Time=%0t, j=%b k=%b q=%b" ,$time,j,k,q);
 clk=0;

  #10 j=0; k=0;
 #20 j=0; k=1;


 #20 j=1; k=0;
 #20 j=1; k=1;
 #20 j=0; k=0;
 #20
$finish;

end

endmodule
```