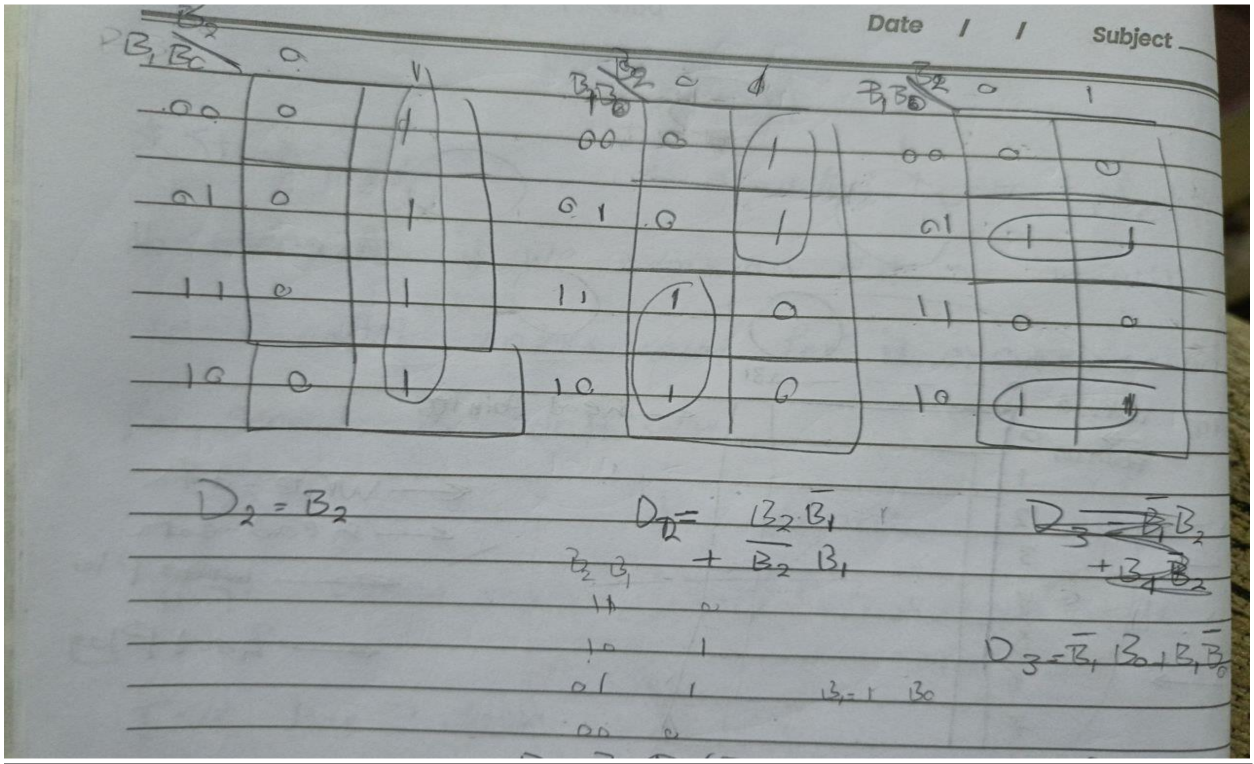
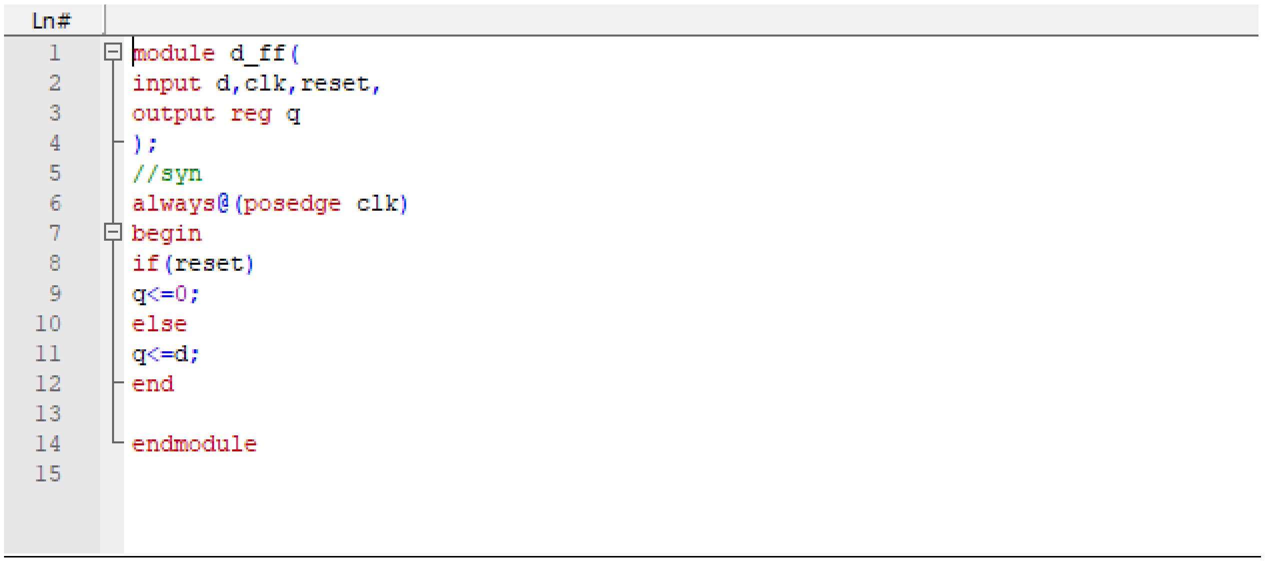
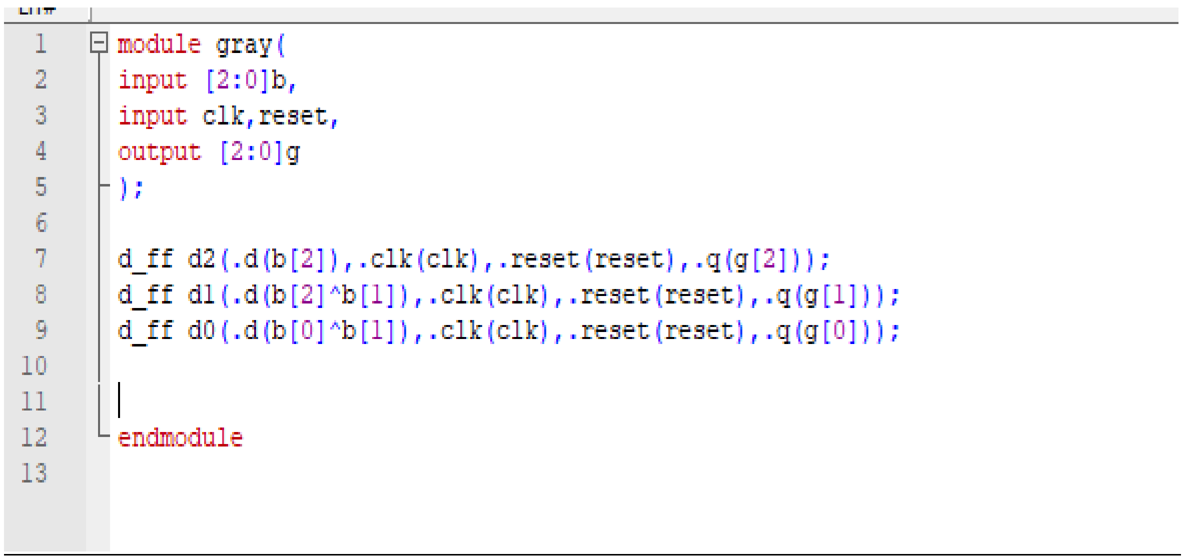
Lap2

Gray to binary

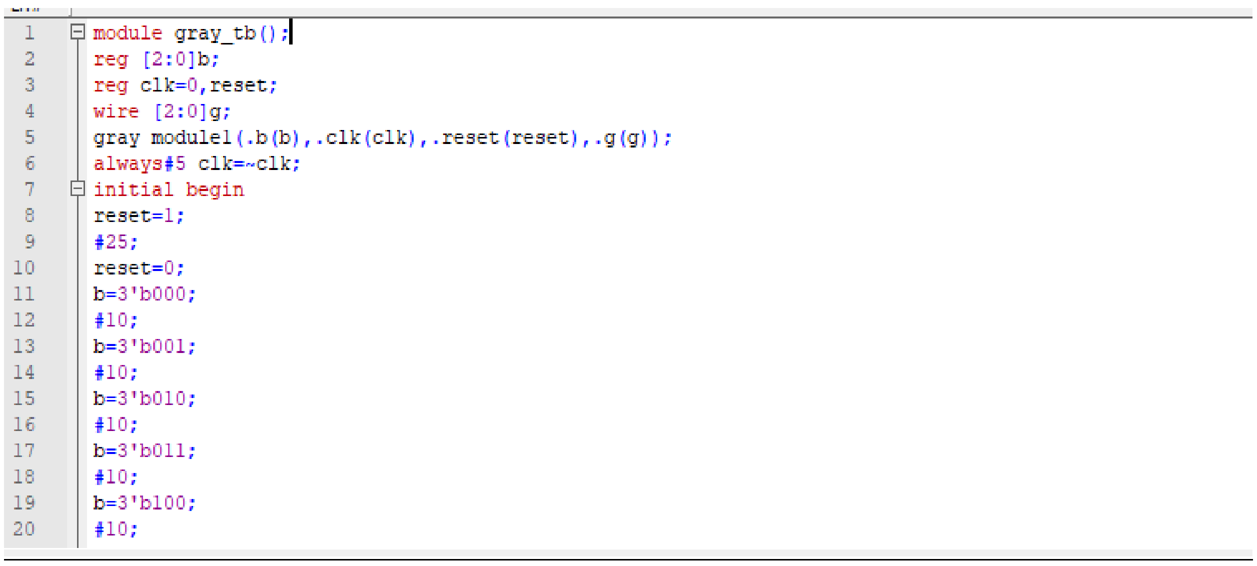

Date / / Subject
$D_2 = B_2$
$D_1 = B_2 \bar{B}_1 + \bar{B}_2 B_1$
$D_0 = \bar{B}_1 B_0 + B_1 \bar{B}_0$



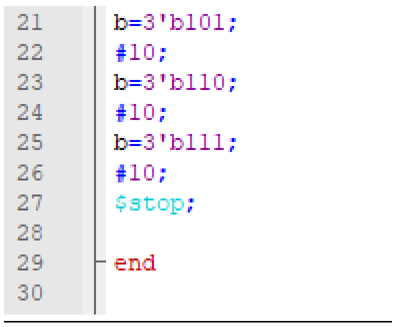
```
module d_ff(
input d,clk,reset,
output reg q
);
//syn
always@(posedge clk)
begin
if(reset)
q<=0;
else
q<=d;
end

endmodule
```



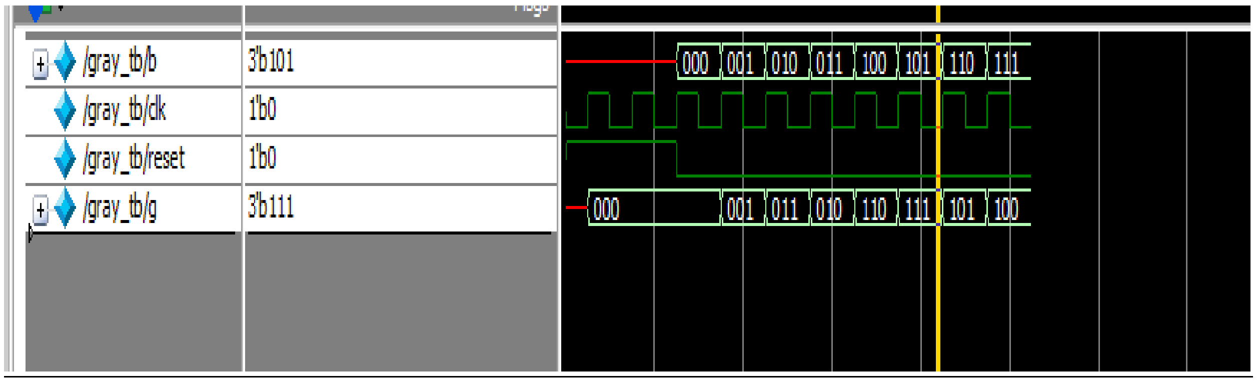
```verilog
module gray(
input [2:0]b,
input clk,reset,
output [2:0]g
);

d_ff d2(.d(b[2]),.clk(clk),.reset(reset),.q(g[2]));
d_ff d1(.d(b[2]^b[1]),.clk(clk),.reset(reset),.q(g[1]));
d_ff d0(.d(b[0]^b[1]),.clk(clk),.reset(reset),.q(g[0]));


endmodule
```



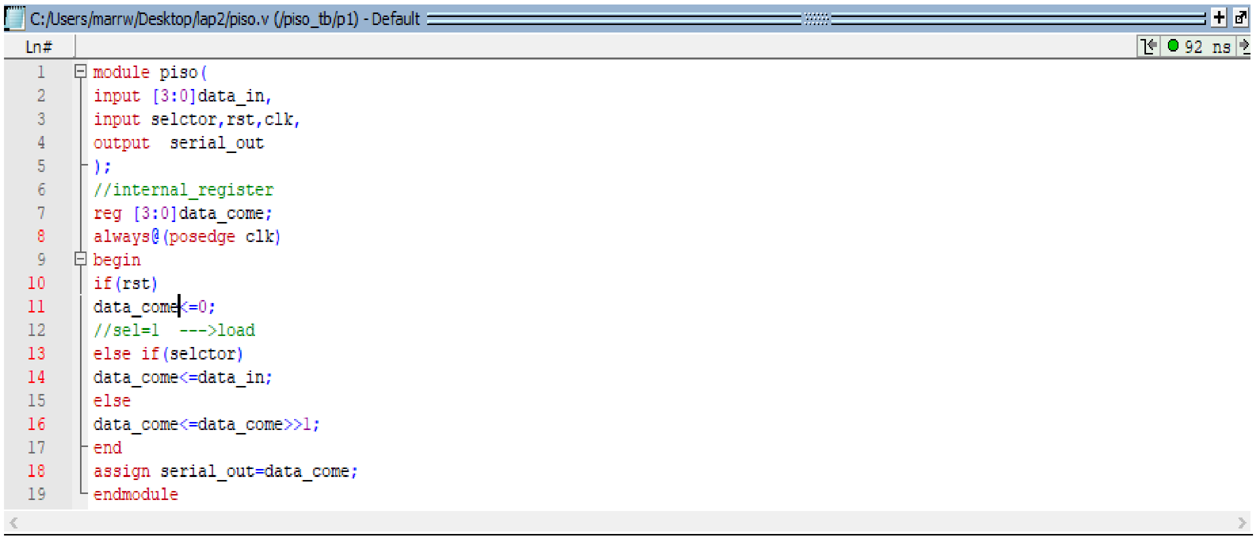
```verilog
module gray_tb();
reg [2:0]b;
reg clk=0,reset;
wire [2:0]g;
gray module1(.b(b),.clk(clk),.reset(reset),.g(g));
always#5 clk=~clk;
initial begin
reset=1;
#25;
reset=0;
b=3'b000;
#10;
b=3'b001;
#10;
b=3'b010;
#10;
b=3'b011;
#10;
b=3'b100;
#10;
```



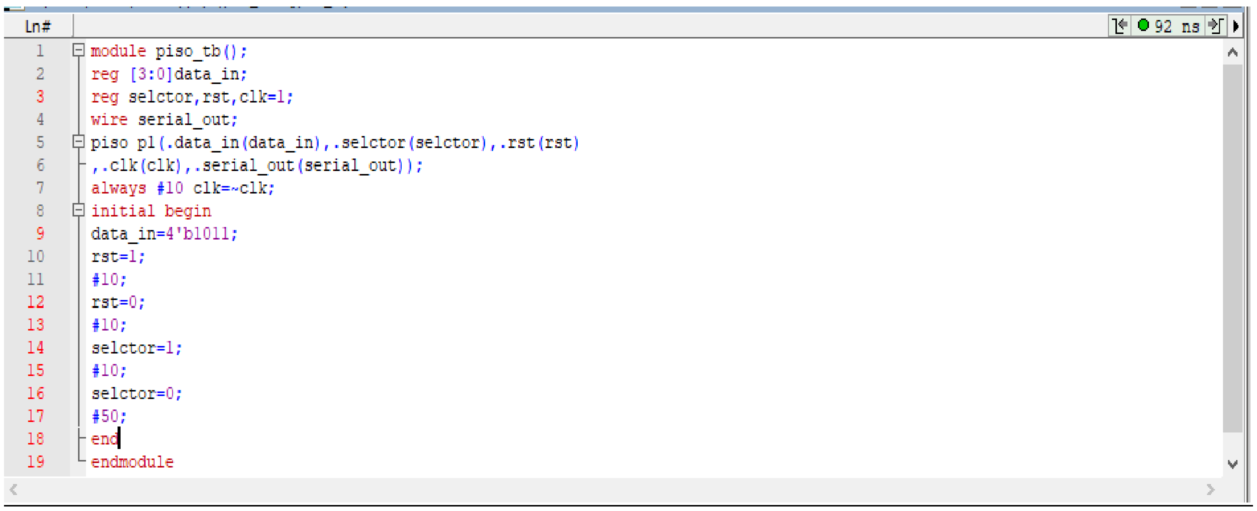
```verilog
b=3'b101;
#10;
b=3'b110;
#10;
b=3'b111;
#10;
$stop;

end
```



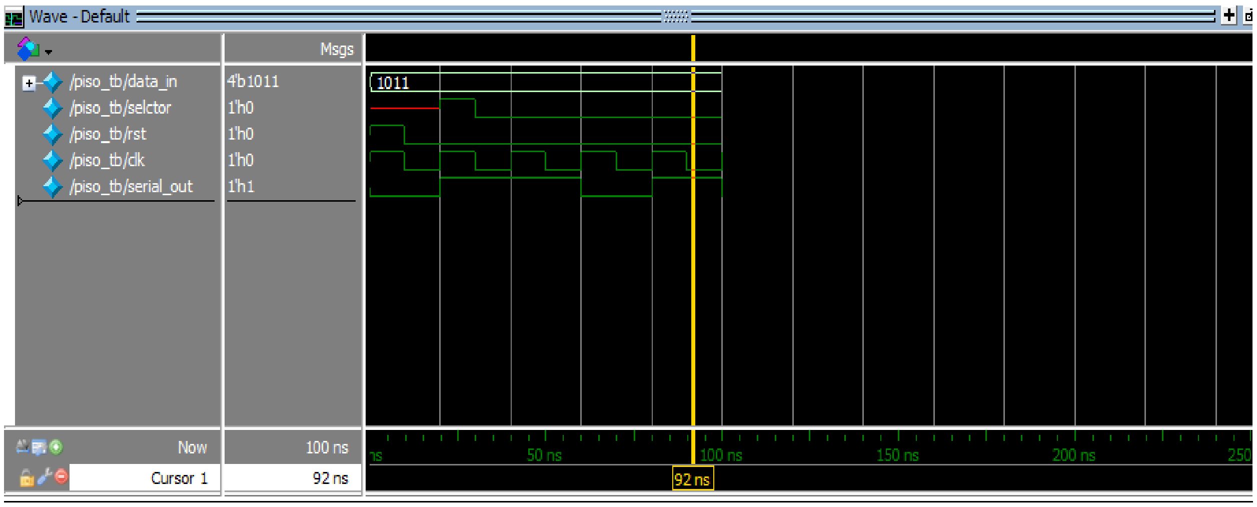
PISO



```
module piso(
input [3:0]data_in,
input selctor,rst,clk,
output  serial_out
);
//internal_register
reg [3:0]data_come;
always@(posedge clk)
begin
if(rst)
data_come<=0;
//sel=1  --->load
else if(selctor)
data_come<=data_in;
else
data_come<=data_come>>1;
end
assign serial_out=data_come;
endmodule
```

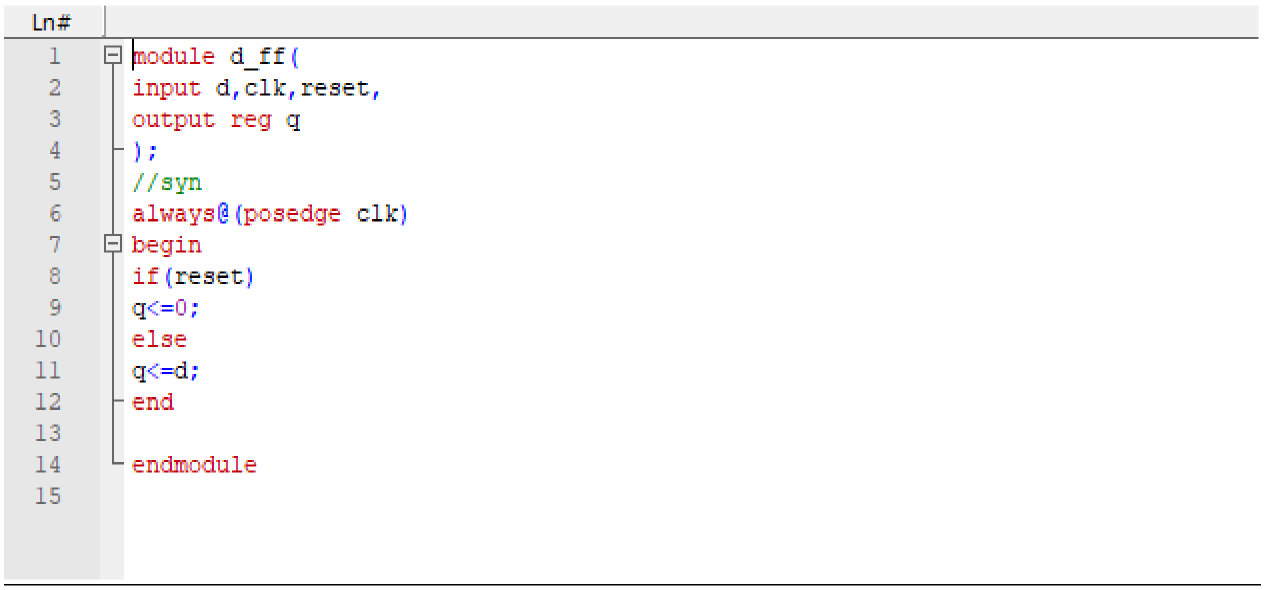


```
module piso_tb();
reg [3:0]data_in;
reg selctor,rst,clk=1;
wire serial_out;
piso p1(.data_in(data_in),.selctor(selctor),.rst(rst)
,.clk(clk),.serial_out(serial_out));
always #10 clk=~clk;
initial begin
data_in=4'b1011;
rst=1;
#10;
rst=0;
#10;
selctor=1;
#10;
selctor=0;
#50;
end
endmodule
```

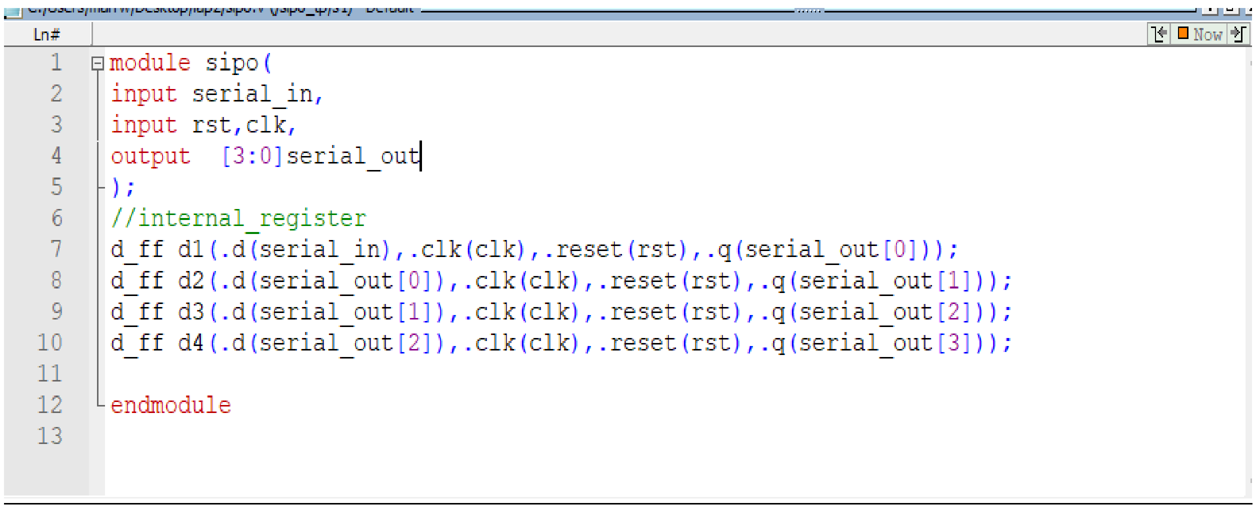


```
/piso_tb/data_in     4'b1011
/piso_tb/selctor     1'h0
/piso_tb/rst         1'h0
/piso_tb/clk         1'h0
/piso_tb/serial_out  1'h1
Now      100 ns
Cursor 1  92 ns
```

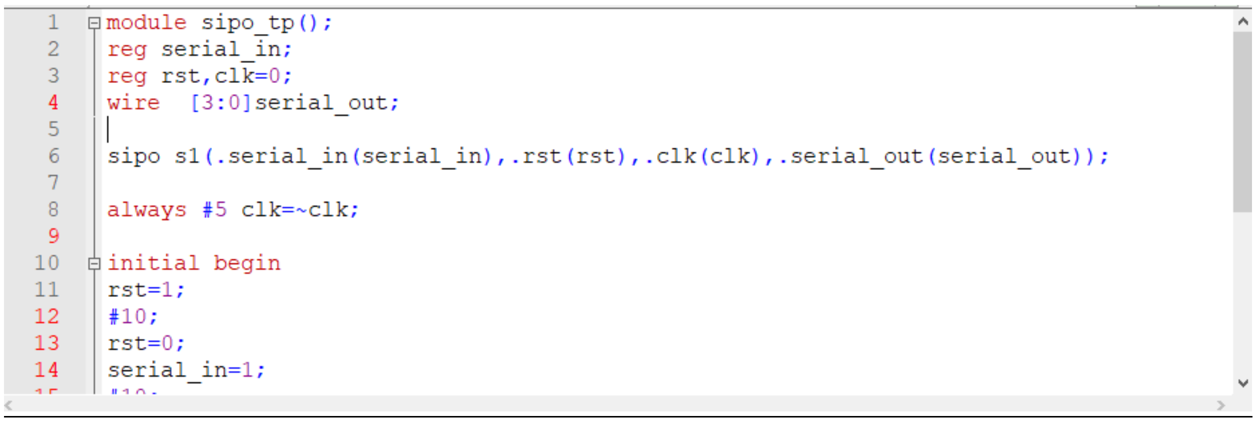
<u>SIPO</u>



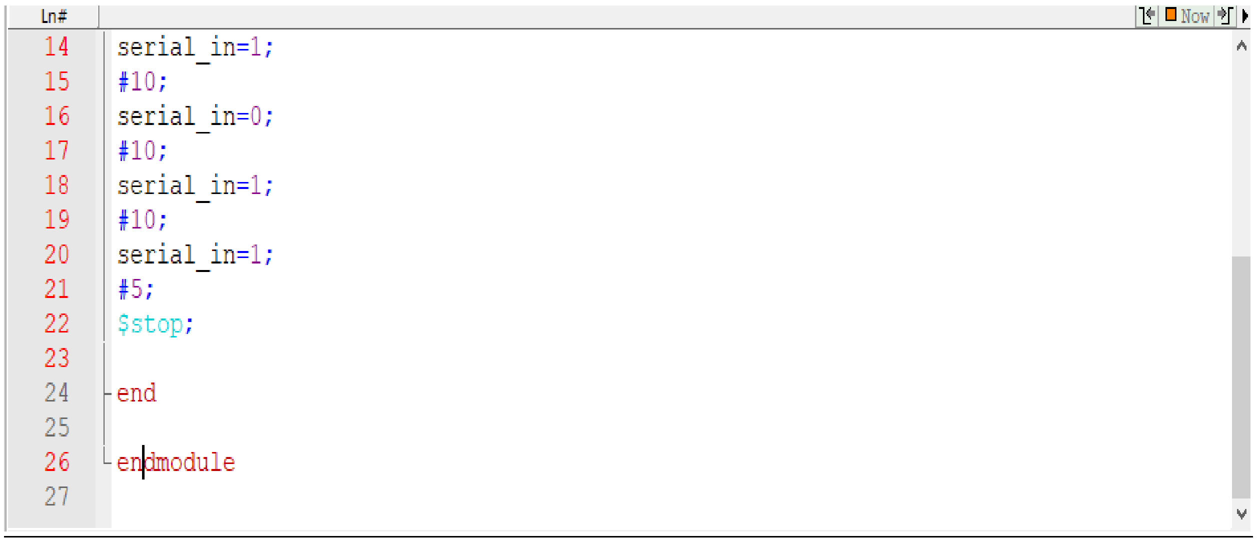
```
module d_ff(
input d,clk,reset,
output reg q
);
//syn
always@(posedge clk)
begin
if(reset)
q<=0;
else
q<=d;
end

endmodule
```



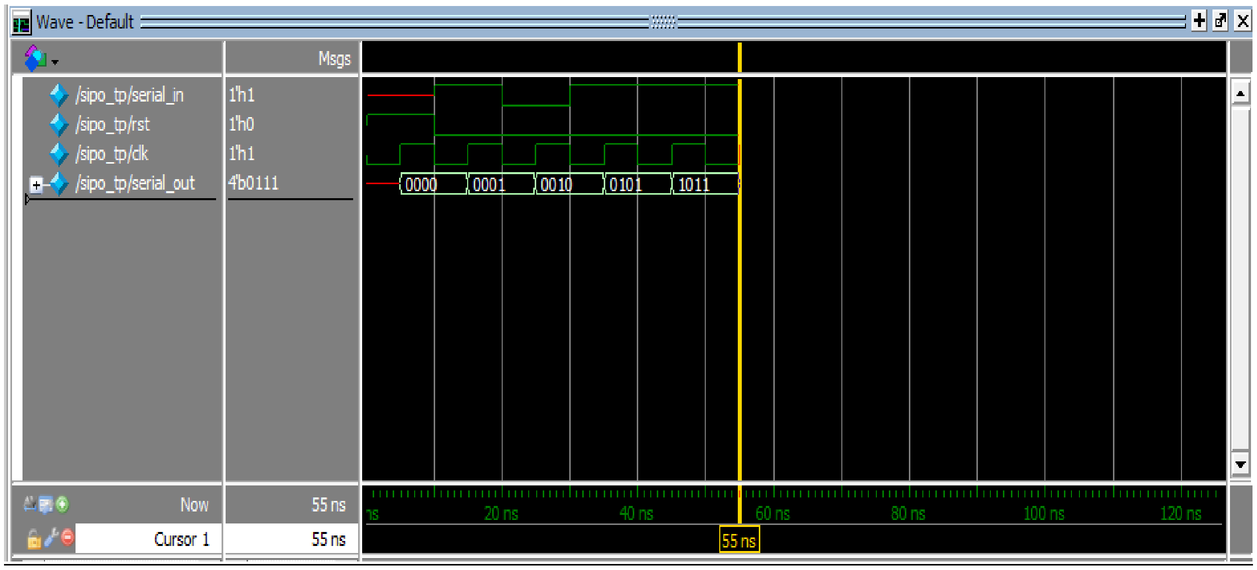
```
module sipo(
input serial_in,
input rst,clk,
output  [3:0]serial_out
);
//internal_register
d_ff d1(.d(serial_in),.clk(clk),.reset(rst),.q(serial_out[0]));
d_ff d2(.d(serial_out[0]),.clk(clk),.reset(rst),.q(serial_out[1]));
d_ff d3(.d(serial_out[1]),.clk(clk),.reset(rst),.q(serial_out[2]));
d_ff d4(.d(serial_out[2]),.clk(clk),.reset(rst),.q(serial_out[3]));

endmodule
```



```
module sipo_tp();
reg serial_in;
reg rst,clk=0;
wire  [3:0]serial_out;

sipo s1(.serial_in(serial_in),.rst(rst),.clk(clk),.serial_out(serial_out));

always #5 clk=~clk;

initial begin
rst=1;
#10;
rst=0;
serial_in=1;
```

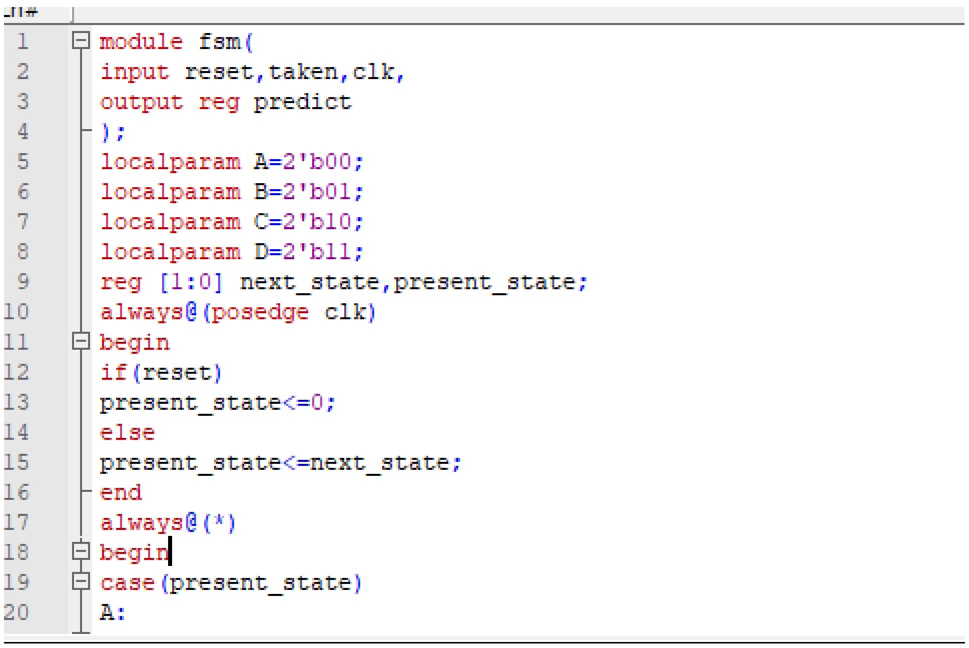


```
serial_in=1;
#10;
serial_in=0;
#10;
serial_in=1;
#10;
serial_in=1;
#5;
$stop;

end

endmodule
```

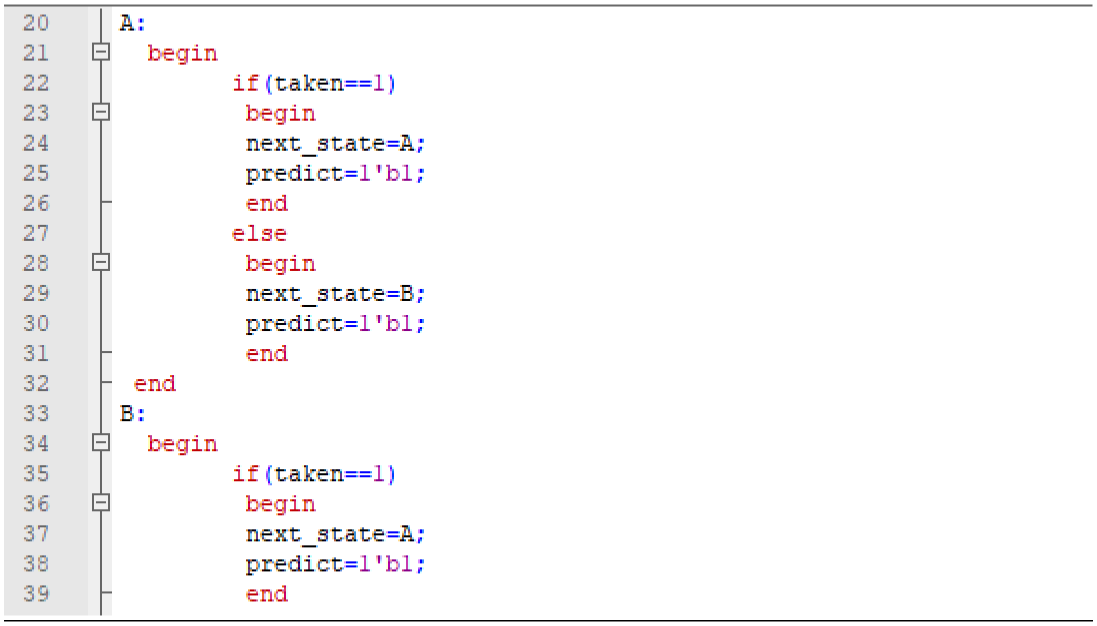


```
Wave - Default
                          Msgs
/sipo_tp/serial_in        1'h1
/sipo_tp/rst              1'h0
/sipo_tp/clk              1'h1
/sipo_tp/serial_out       4'b0111     0000  0001  0010  0101  1011

Now                       55 ns
Cursor 1                  55 ns       55 ns
```

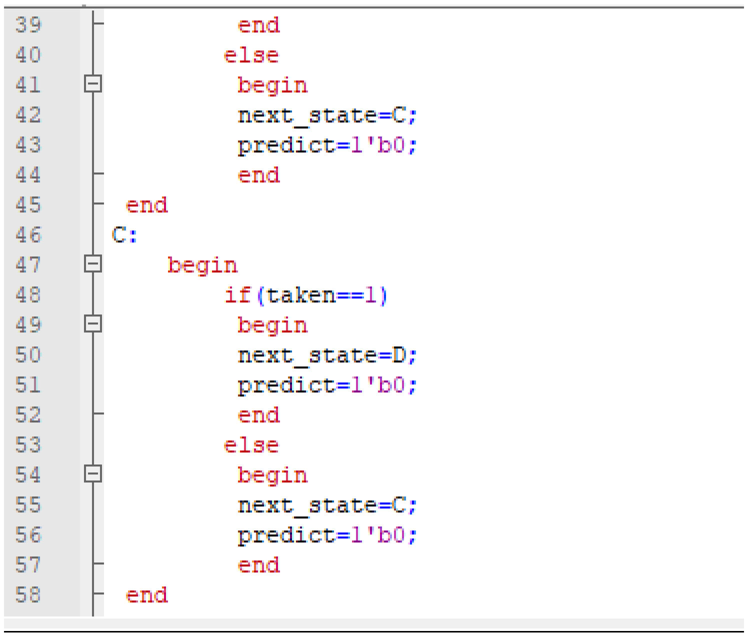
FSM



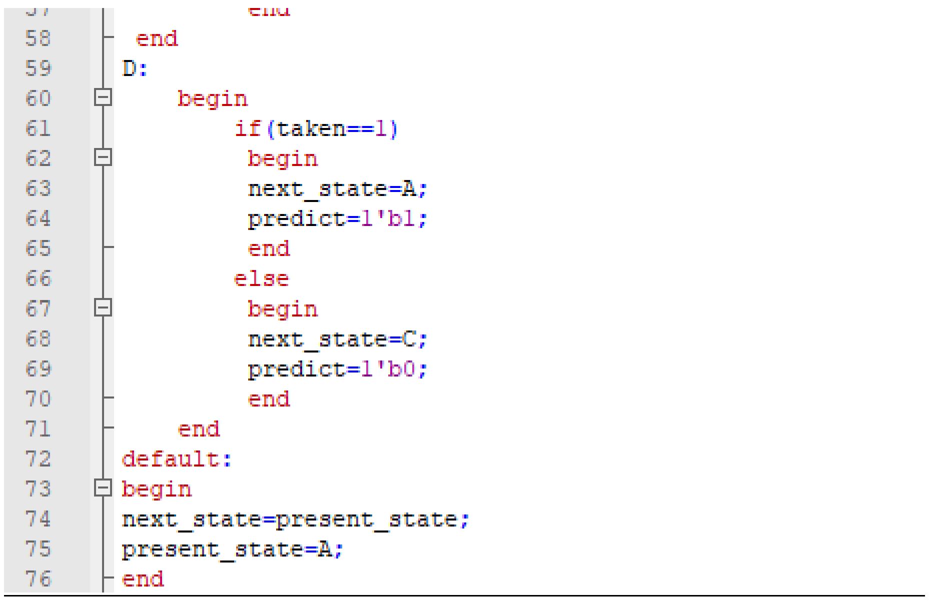
```
module fsm(
input reset,taken,clk,
output reg predict
);
localparam A=2'b00;
localparam B=2'b01;
localparam C=2'b10;
localparam D=2'b11;
reg [1:0] next_state,present_state;
always@(posedge clk)
begin
if(reset)
present_state<=0;
else
present_state<=next_state;
end
always@(*)
begin
case(present_state)
A:
```



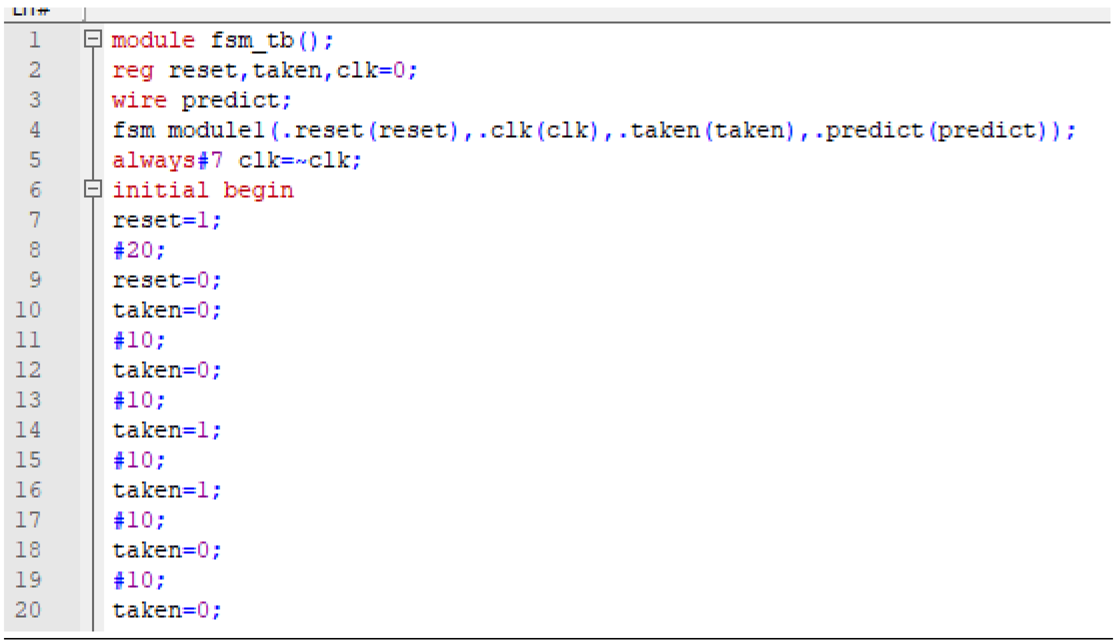
```
A:
  begin
        if(taken==1)
         begin
         next_state=A;
         predict=1'b1;
         end
        else
         begin
         next_state=B;
         predict=1'b1;
         end
 end
B:
  begin
        if(taken==1)
         begin
         next_state=A;
         predict=1'b1;
         end
```



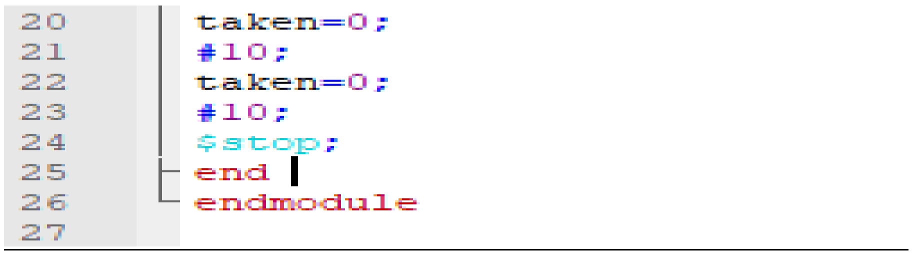
```
         end
        else
         begin
         next_state=C;
         predict=1'b0;
         end
 end
C:
    begin
        if(taken==1)
         begin
         next_state=D;
         predict=1'b0;
         end
        else
         begin
         next_state=C;
         predict=1'b0;
         end
 end
```



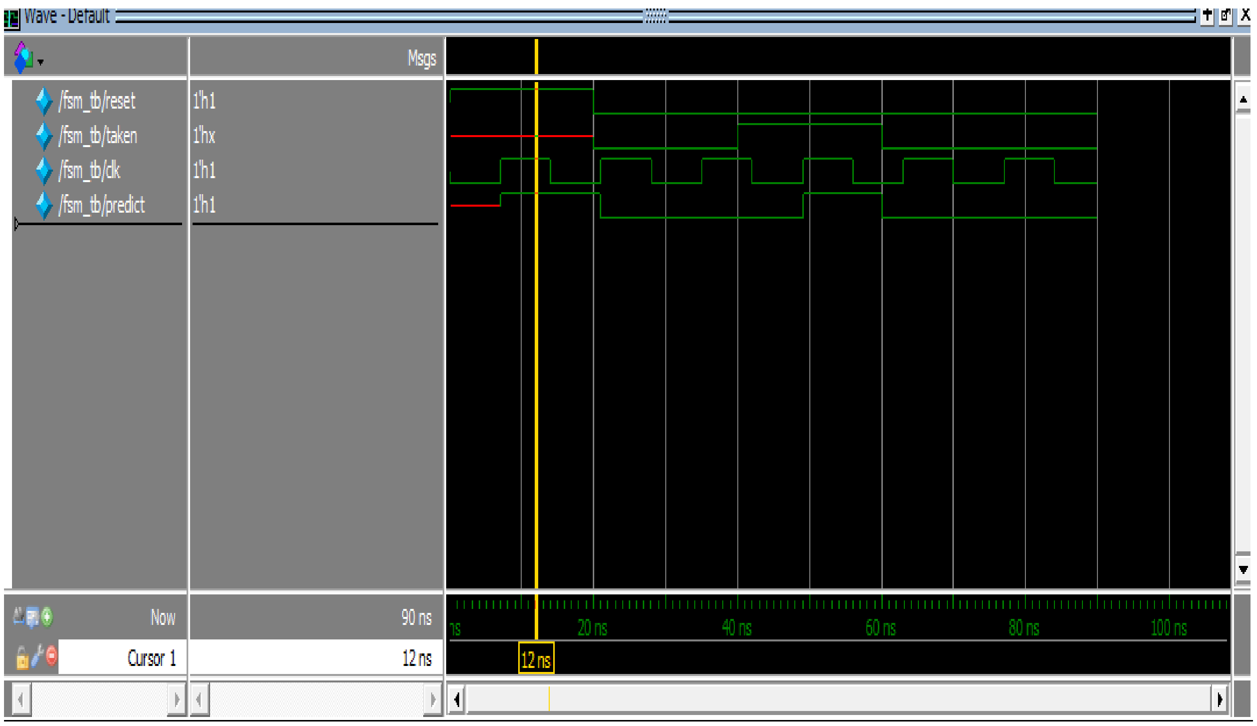
```
 end
D:
    begin
        if(taken==1)
         begin
         next_state=A;
         predict=1'b1;
         end
        else
         begin
         next_state=C;
         predict=1'b0;
         end
    end
default:
begin
next_state=present_state;
present_state=A;
end
```



```
module fsm_tb();
reg reset,taken,clk=0;
wire predict;
fsm module1(.reset(reset),.clk(clk),.taken(taken),.predict(predict));
always#7 clk=~clk;
initial begin
reset=1;
#20;
reset=0;
taken=0;
#10;
taken=0;
#10;
taken=1;
#10;
taken=1;
#10;
taken=0;
#10;
taken=0;
```

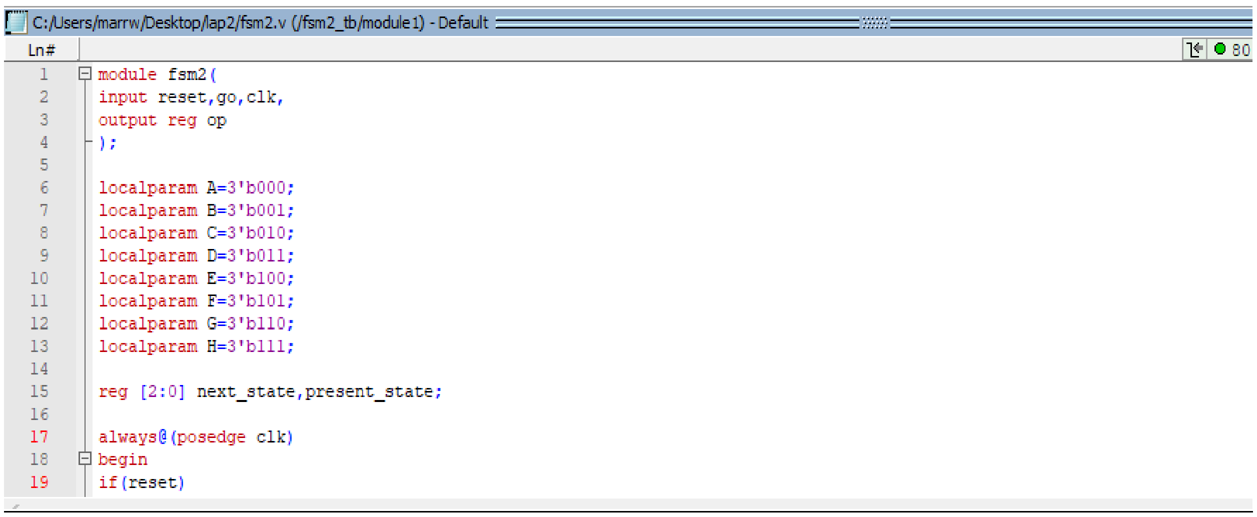


```
taken=0;
#10;
taken=0;
#10;
$stop;
end
endmodule
```

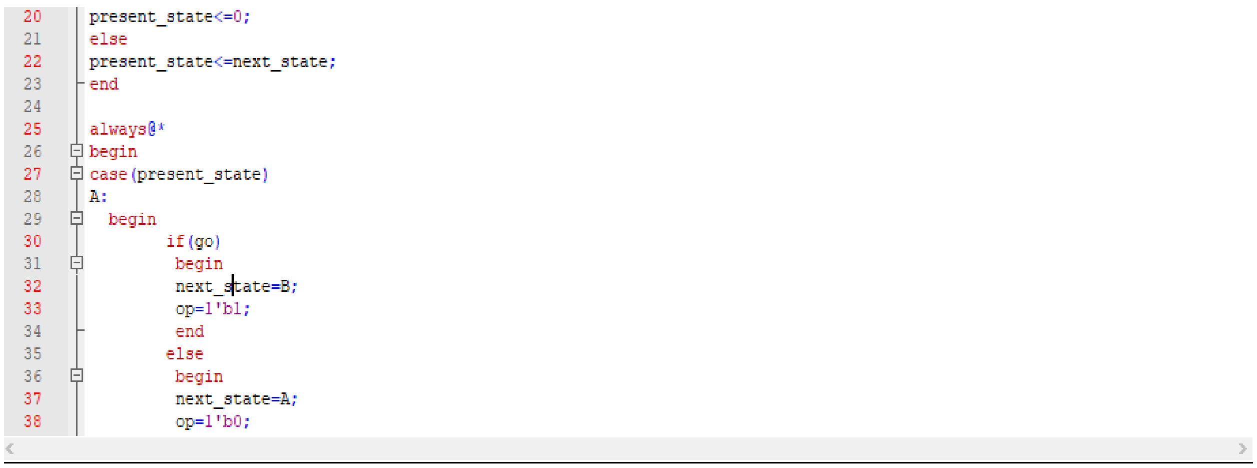


```
Wave - Default
                         Msgs
/fsm_tb/reset            1'h1
/fsm_tb/taken            1'hx
/fsm_tb/clk              1'h1
/fsm_tb/predict          1'h1

Now                      90 ns
Cursor 1                 12 ns
```

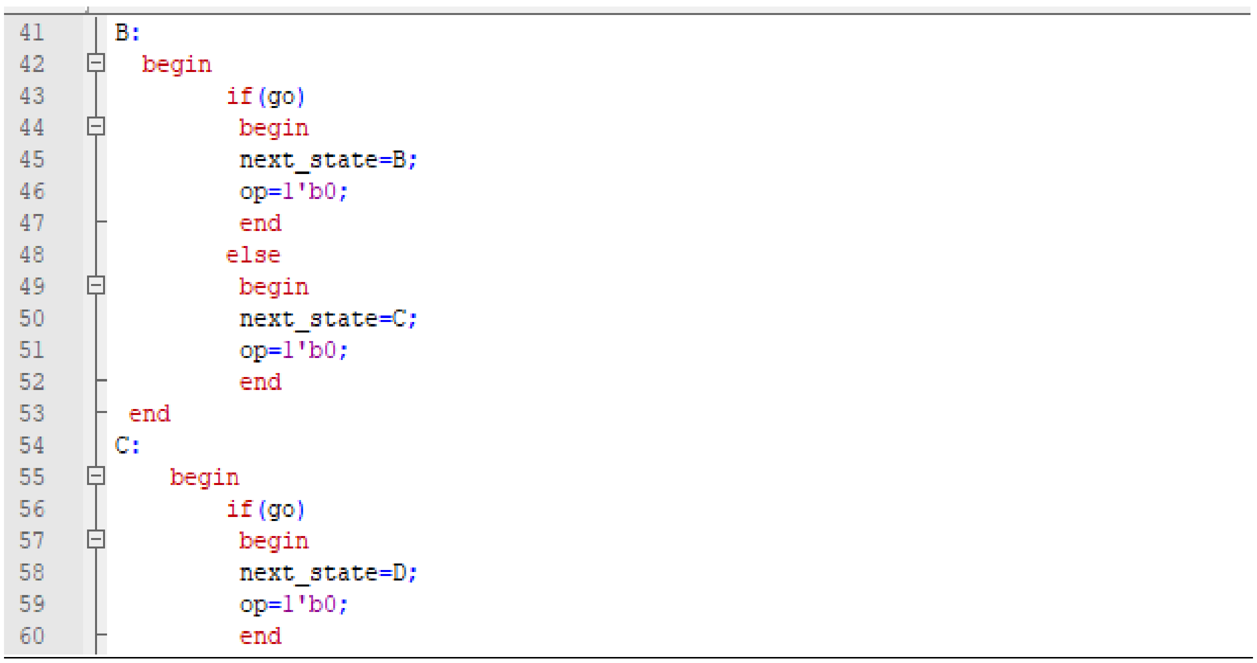
FSM2


C:/Users/marrw/Desktop/lap2/fsm2.v (/fsm2_tb/module1) - Default

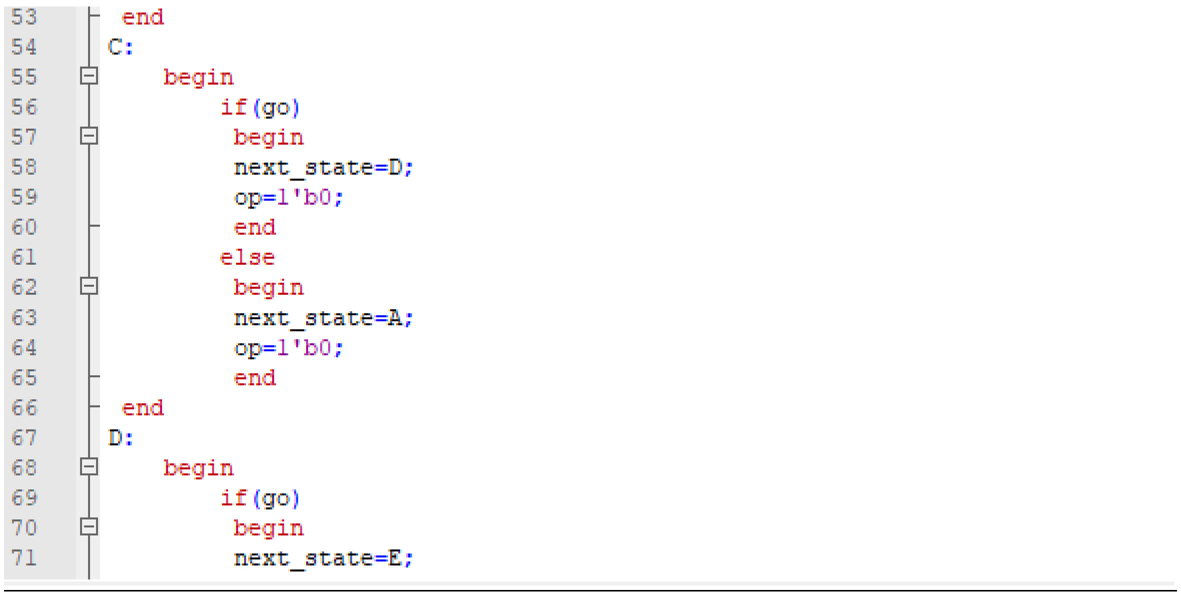
```
module fsm2(
input reset,go,clk,
output reg op
);

localparam A=3'b000;
localparam B=3'b001;
localparam C=3'b010;
localparam D=3'b011;
localparam E=3'b100;
localparam F=3'b101;
localparam G=3'b110;
localparam H=3'b111;

reg [2:0] next_state,present_state;

always@(posedge clk)
begin
if(reset)
```



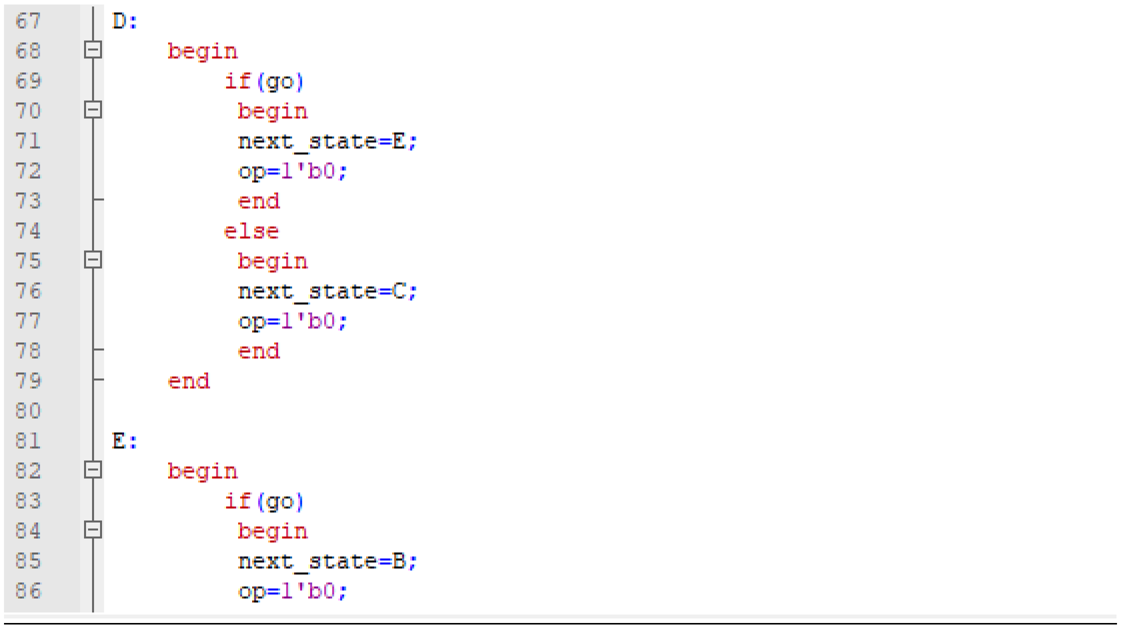
```
present_state<=0;
else
present_state<=next_state;
end

always@*
begin
case(present_state)
A:
  begin
        if(go)
         begin
         next_state=B;
         op=1'b1;
         end
        else
         begin
         next_state=A;
         op=1'b0;
```



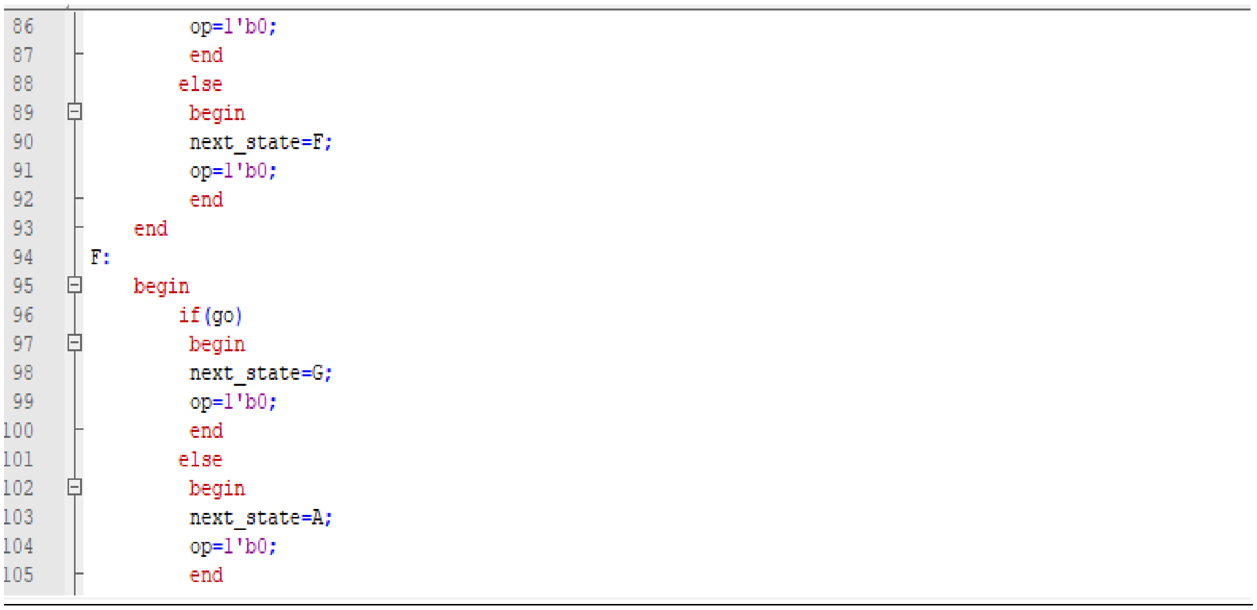
```
B:
  begin
        if(go)
         begin
         next_state=B;
         op=1'b0;
         end
        else
         begin
         next_state=C;
         op=1'b0;
         end
 end
C:
    begin
        if(go)
         begin
         next_state=D;
         op=1'b0;
         end
```



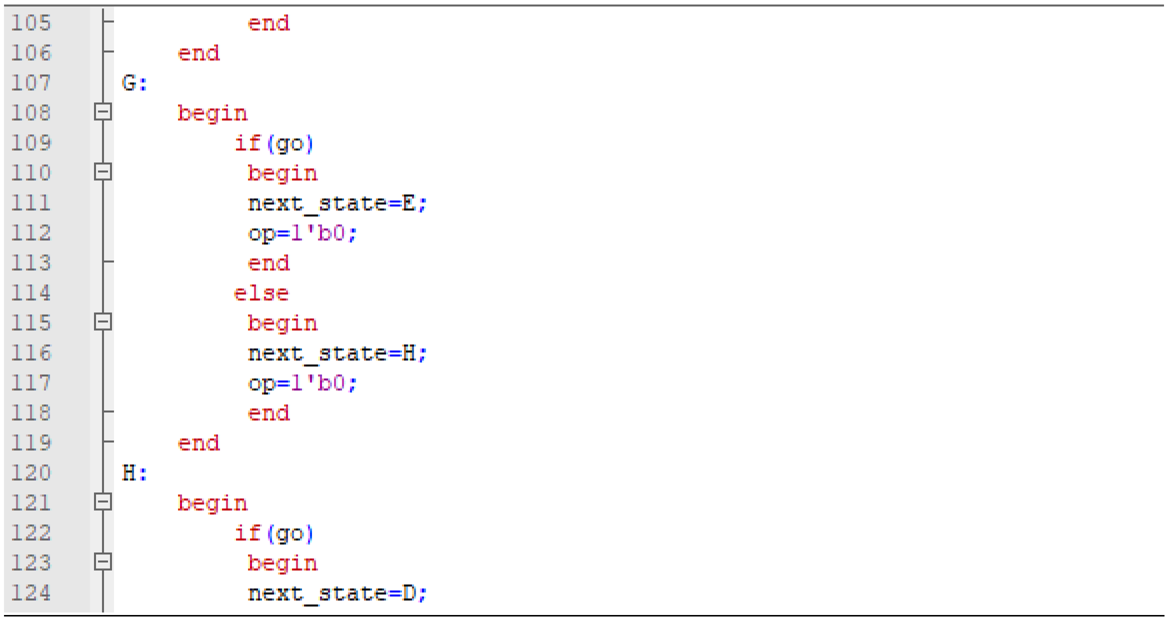
```
 end
C:
    begin
        if(go)
         begin
         next_state=D;
         op=1'b0;
         end
        else
         begin
         next_state=A;
         op=1'b0;
         end
 end
D:
    begin
        if(go)
         begin
         next_state=E;
```



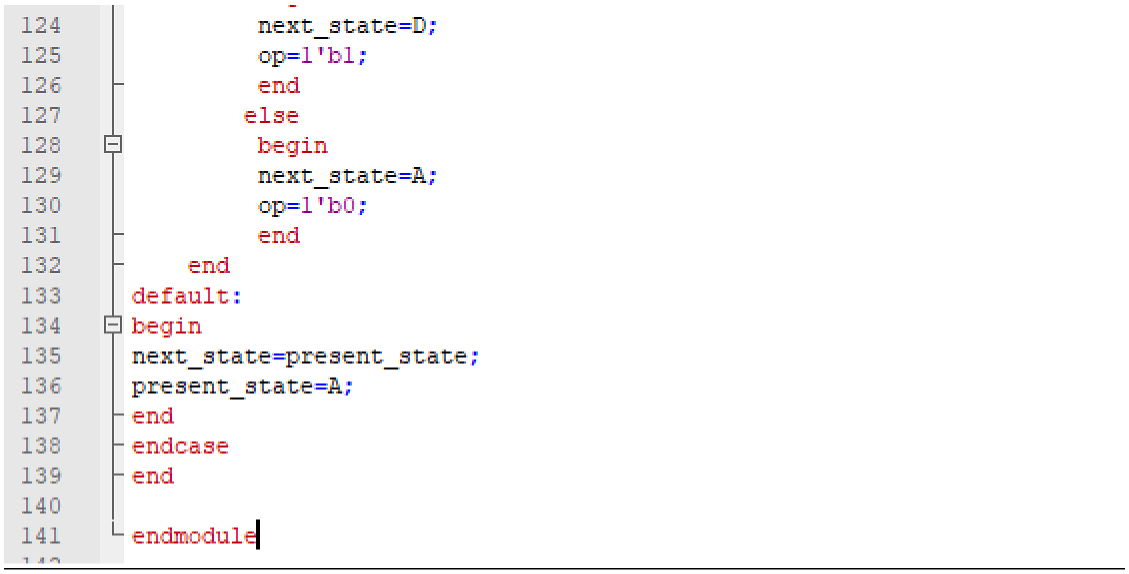
```
D:
    begin
        if(go)
         begin
         next_state=E;
         op=1'b0;
         end
        else
         begin
         next_state=C;
         op=1'b0;
         end
    end

E:
    begin
        if(go)
         begin
         next_state=B;
         op=1'b0;
```



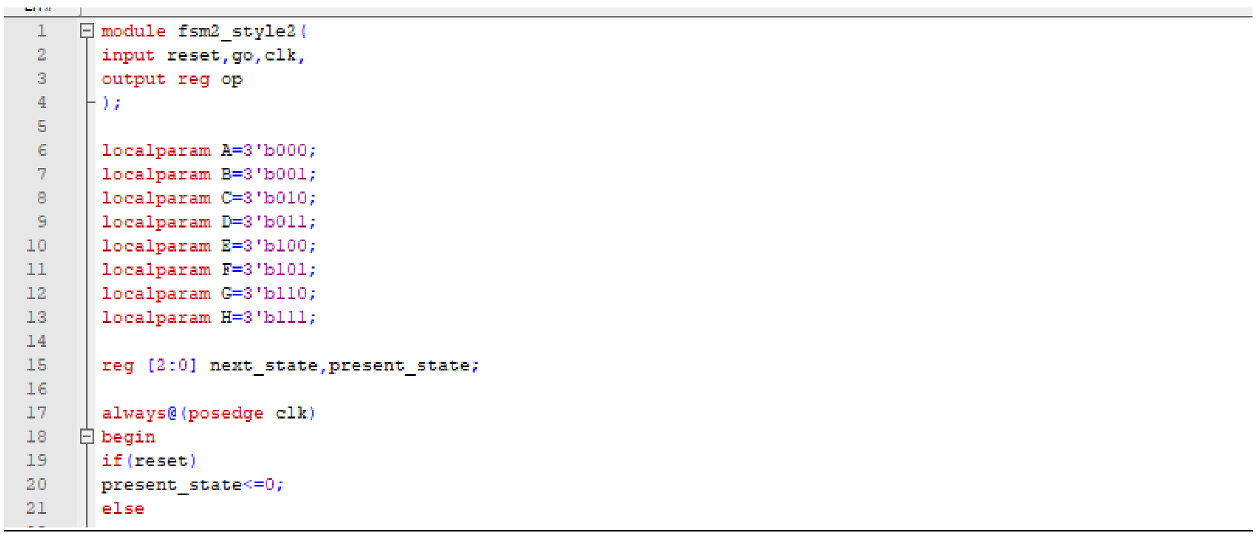
```
         op=1'b0;
         end
        else
         begin
         next_state=F;
         op=1'b0;
         end
    end
F:
    begin
        if(go)
         begin
         next_state=G;
         op=1'b0;
         end
        else
         begin
         next_state=A;
         op=1'b0;
         end
```



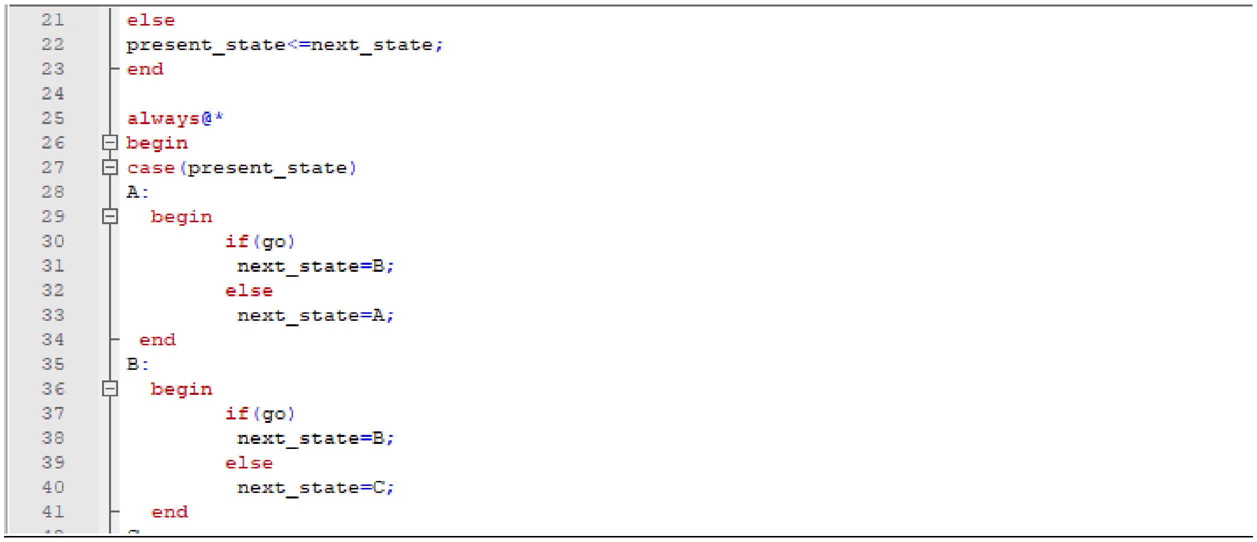
```verilog
         end
    end
G:
    begin
        if(go)
         begin
         next_state=E;
         op=1'b0;
         end
        else
         begin
         next_state=H;
         op=1'b0;
         end
    end
H:
    begin
        if(go)
         begin
         next_state=D;
```



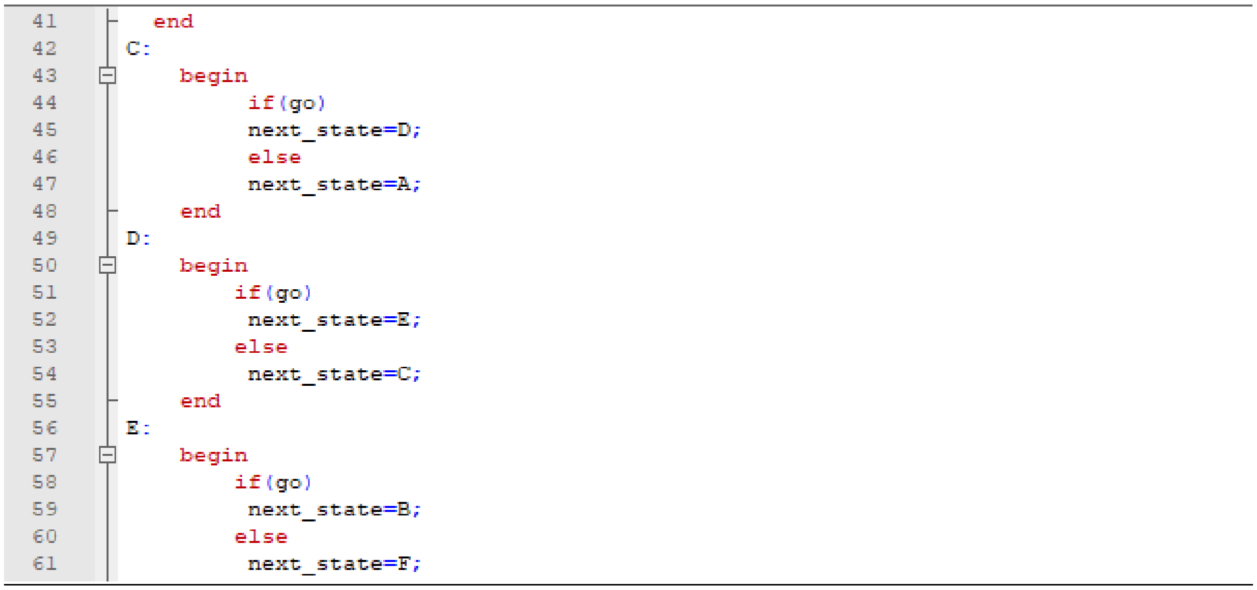
```verilog
         next_state=D;
         op=1'b1;
         end
        else
         begin
         next_state=A;
         op=1'b0;
         end
    end
default:
begin
next_state=present_state;
present_state=A;
end
endcase
end

endmodule
```



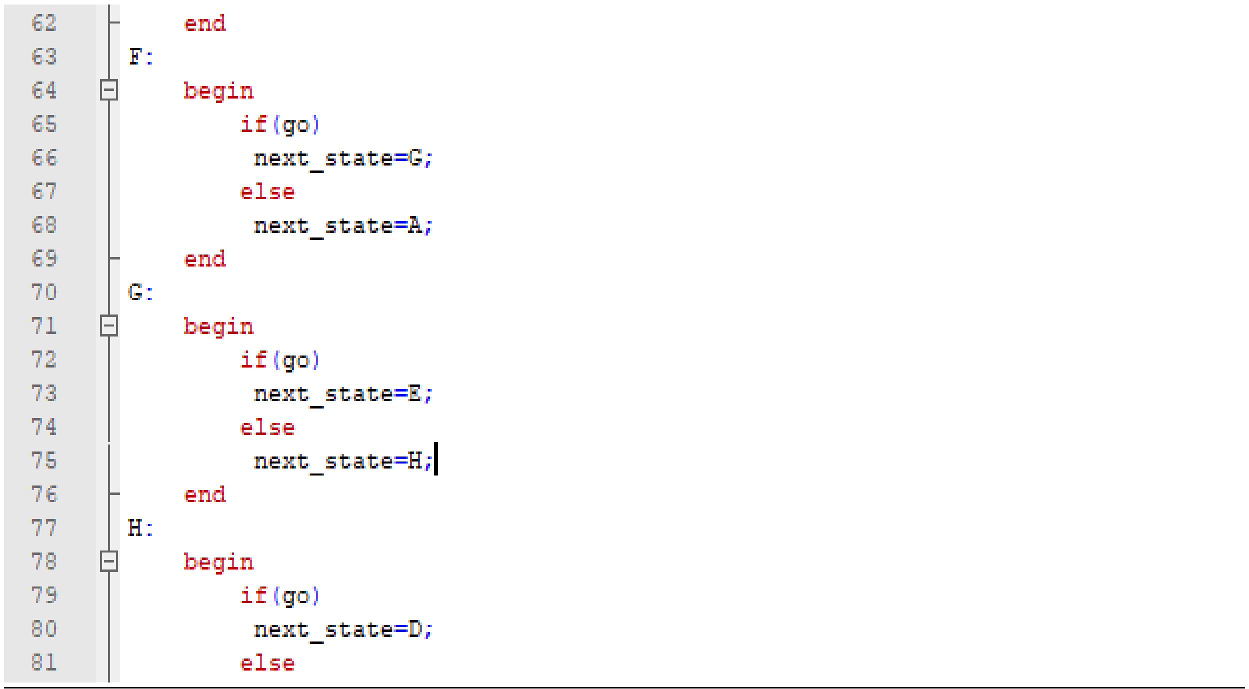
```
module fsm2_style2(
input reset,go,clk,
output reg op
);

localparam A=3'b000;
localparam B=3'b001;
localparam C=3'b010;
localparam D=3'b011;
localparam E=3'b100;
localparam F=3'b101;
localparam G=3'b110;
localparam H=3'b111;

reg [2:0] next_state,present_state;

always@(posedge clk)
begin
if(reset)
present_state<=0;
else
```



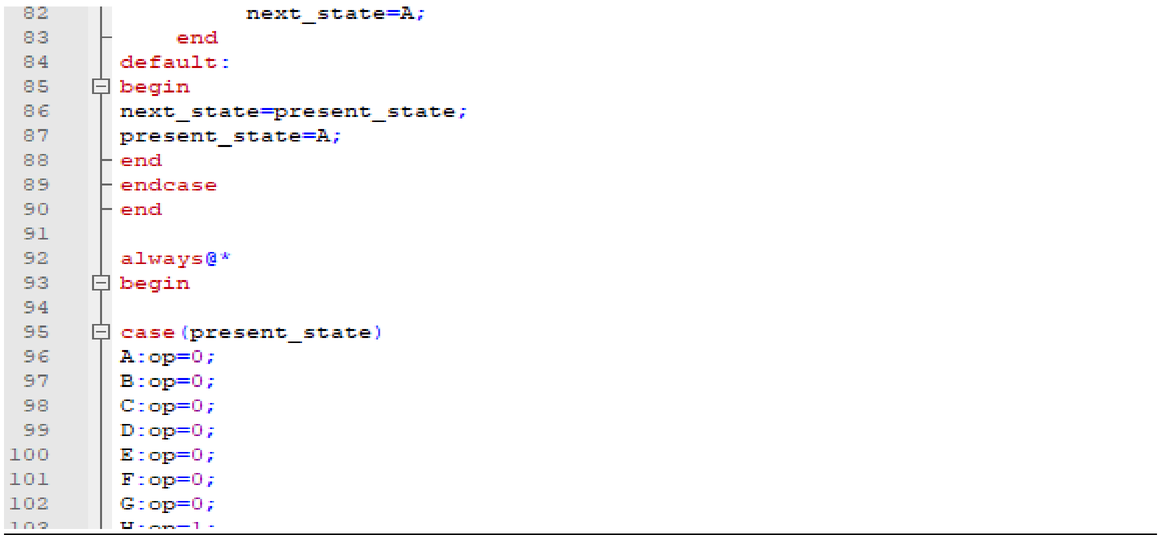
```
else
present_state<=next_state;
end

always@*
begin
case(present_state)
A:
  begin
        if(go)
         next_state=B;
        else
         next_state=A;
 end
B:
  begin
        if(go)
         next_state=B;
        else
         next_state=C;
  end
```



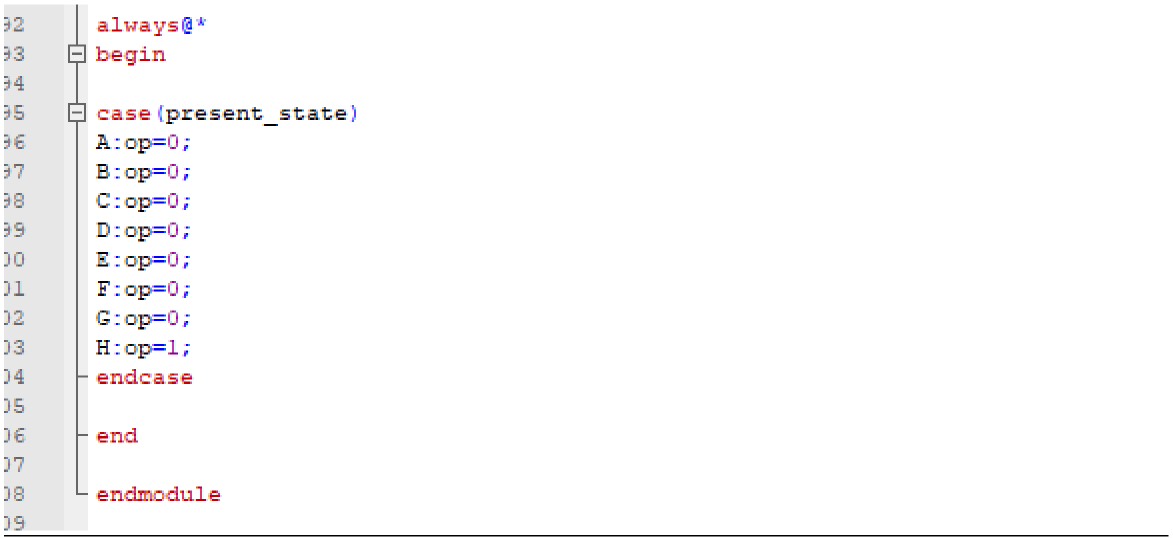
```
   end
C:
    begin
         if(go)
         next_state=D;
         else
         next_state=A;
    end
D:
    begin
        if(go)
         next_state=E;
        else
         next_state=C;
    end
E:
    begin
        if(go)
         next_state=B;
        else
         next_state=F;
```



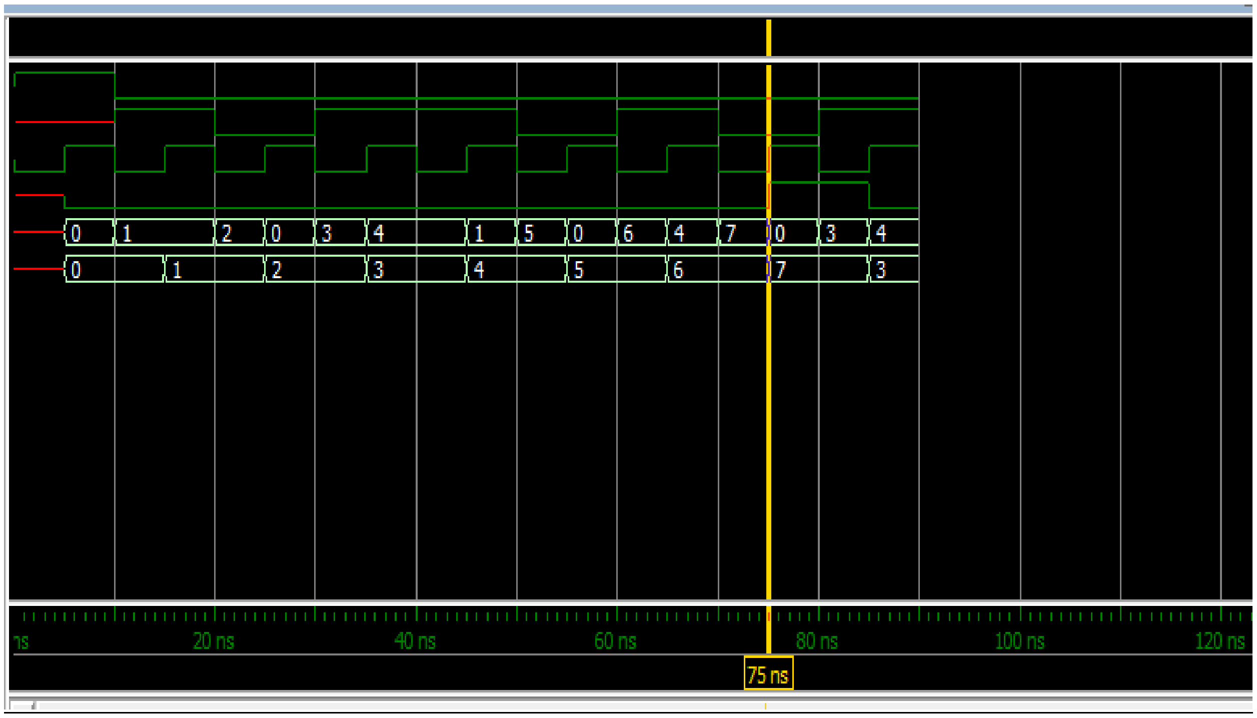
```
    end
F:
    begin
        if(go)
         next_state=G;
        else
         next_state=A;
    end
G:
    begin
        if(go)
         next_state=E;
        else
         next_state=H;
    end
H:
    begin
        if(go)
         next_state=D;
        else
```



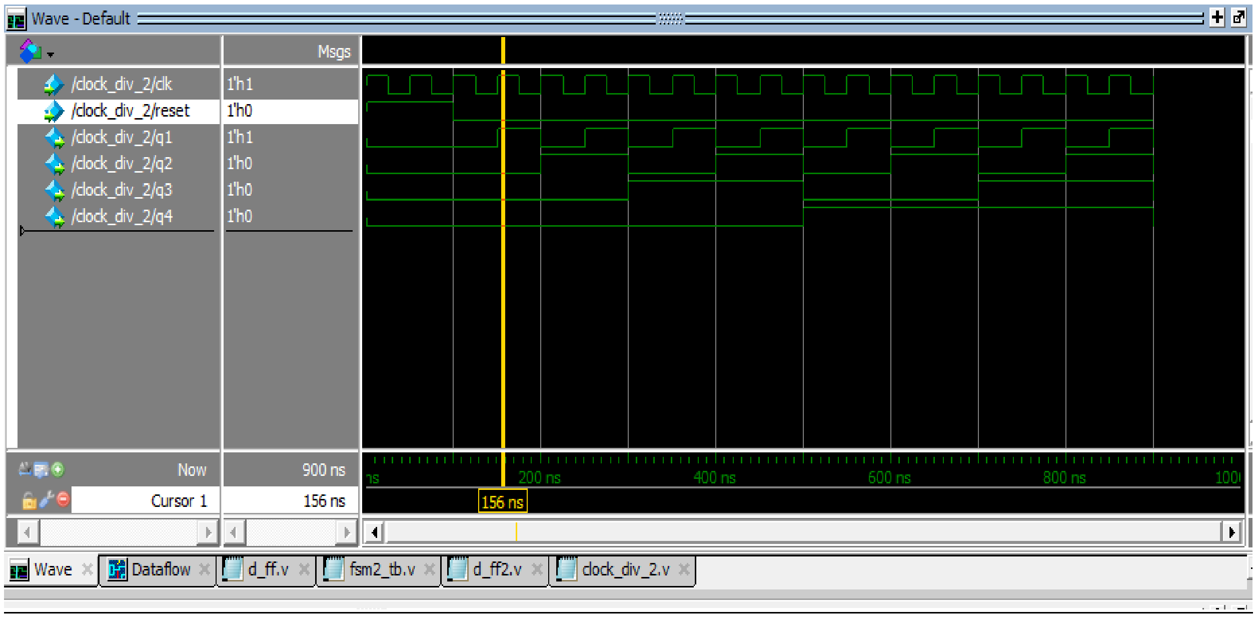
```
         next_state=A;
    end
default:
begin
next_state=present_state;
present_state=A;
end
endcase
end

always@*
begin

case(present_state)
A:op=0;
B:op=0;
C:op=0;
D:op=0;
E:op=0;
F:op=0;
G:op=0;
H:op=1;
```

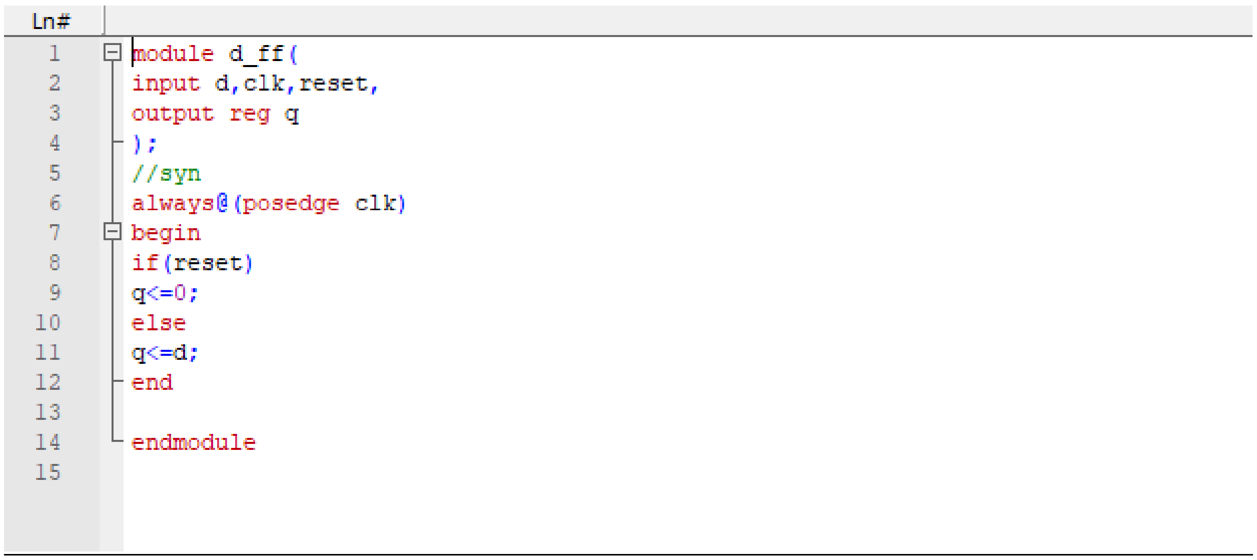


```
always@*
begin

case(present_state)
A:op=0;
B:op=0;
C:op=0;
D:op=0;
E:op=0;
F:op=0;
G:op=0;
H:op=1;
endcase

end

endmodule
```

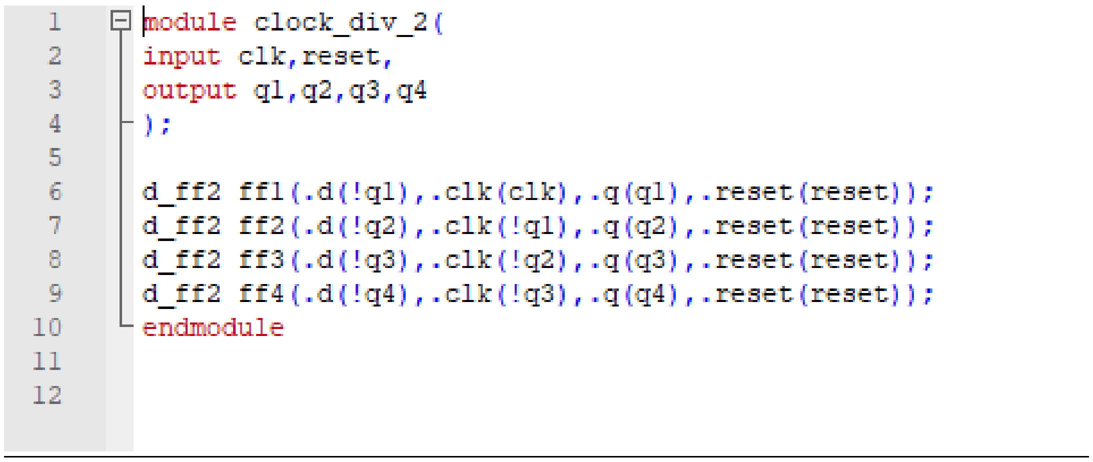


CLOCK DIV BY 2,4,8,16



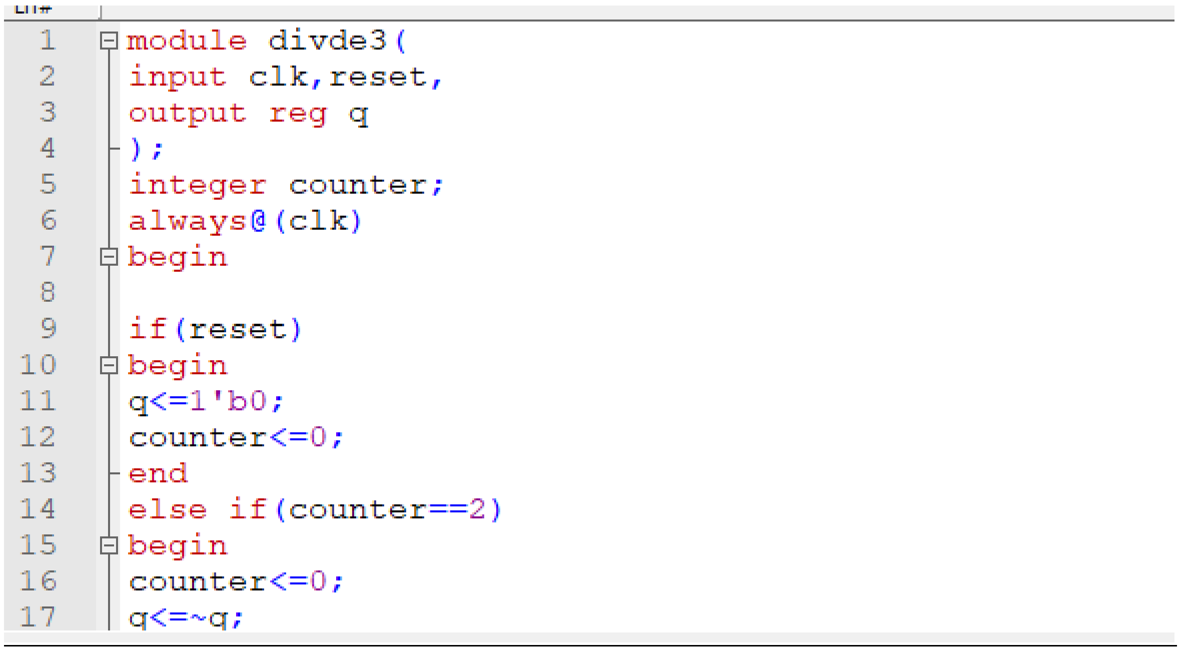


```
module d_ff(
input d,clk,reset,
output reg q
);
//syn
always@(posedge clk)
begin
if(reset)
q<=0;
else
q<=d;
end

endmodule
```

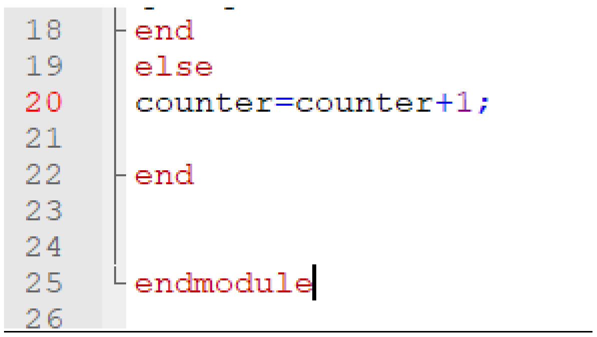


```
module clock_div_2(
input clk,reset,
output q1,q2,q3,q4
);

d_ff2 ff1(.d(!q1),.clk(clk),.q(q1),.reset(reset));
d_ff2 ff2(.d(!q2),.clk(!q1),.q(q2),.reset(reset));
d_ff2 ff3(.d(!q3),.clk(!q2),.q(q3),.reset(reset));
d_ff2 ff4(.d(!q4),.clk(!q3),.q(q4),.reset(reset));
endmodule
```

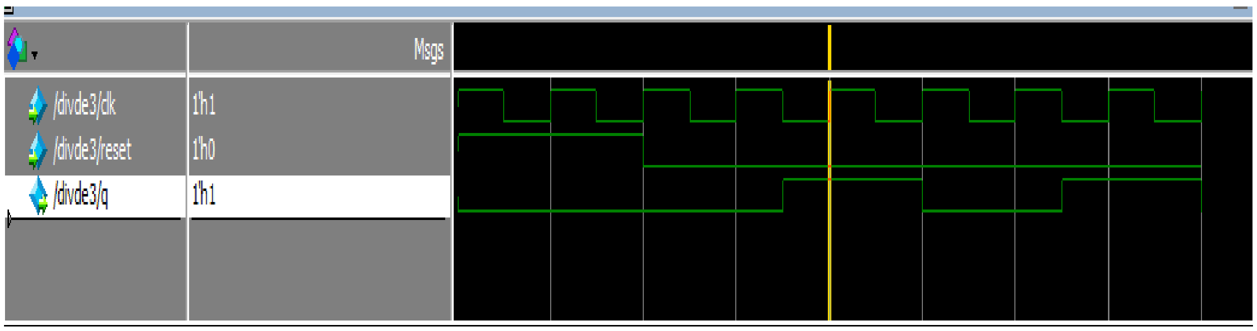
Div by 3



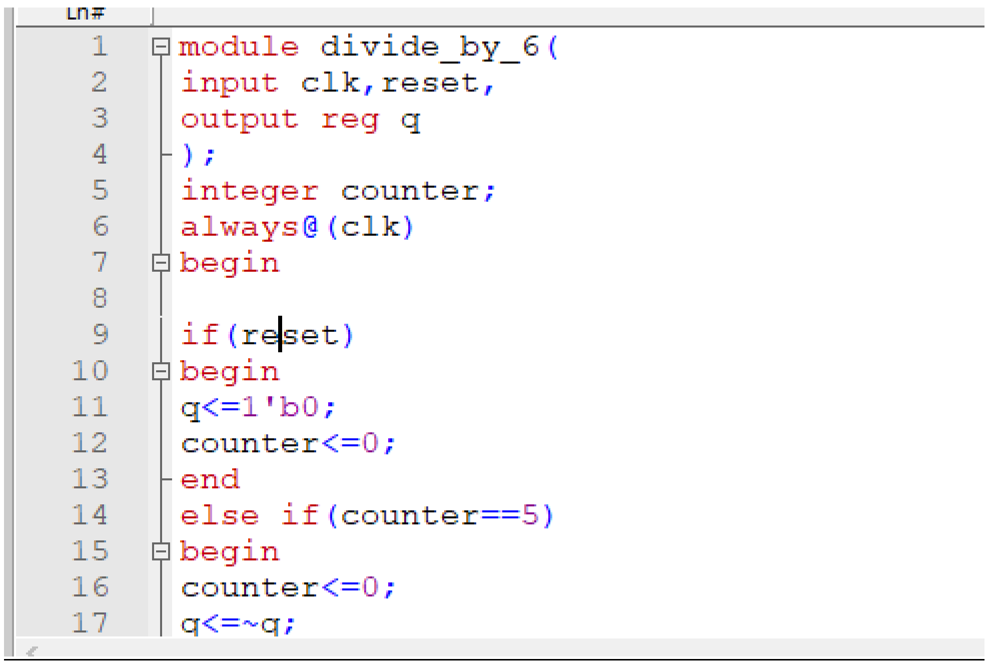
```
module divde3(
input clk,reset,
output reg q
);
integer counter;
always@(clk)
begin

if(reset)
begin
q<=1'b0;
counter<=0;
end
else if(counter==2)
begin
counter<=0;
q<=~q;
```



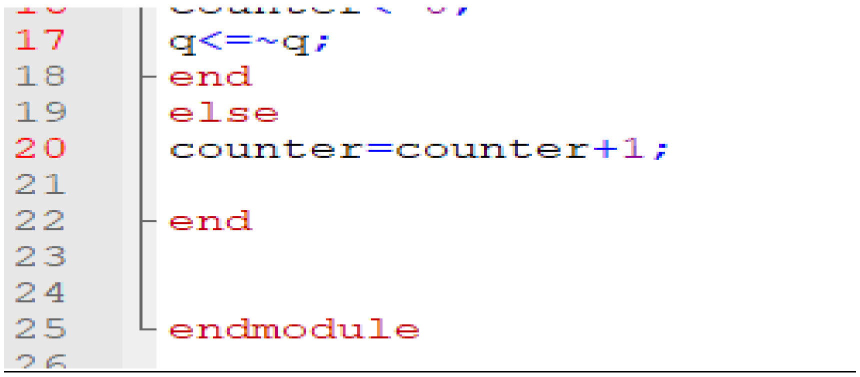
```
end
else
counter=counter+1;

end


endmodule
```



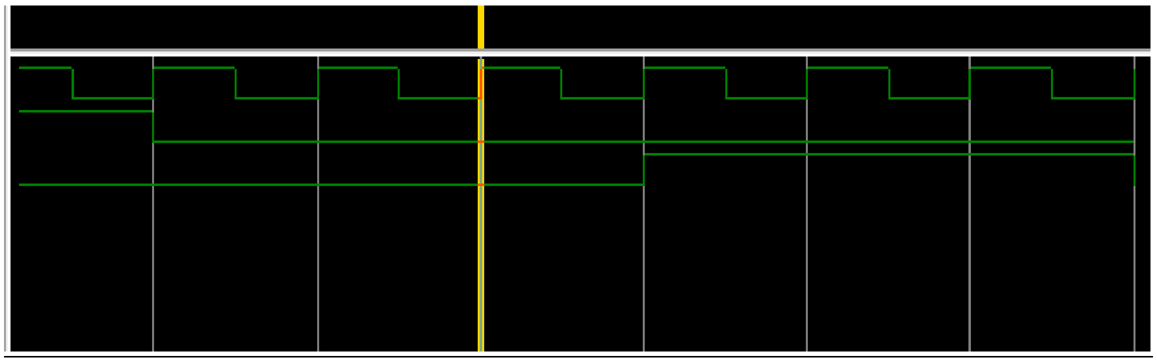
Div by 6



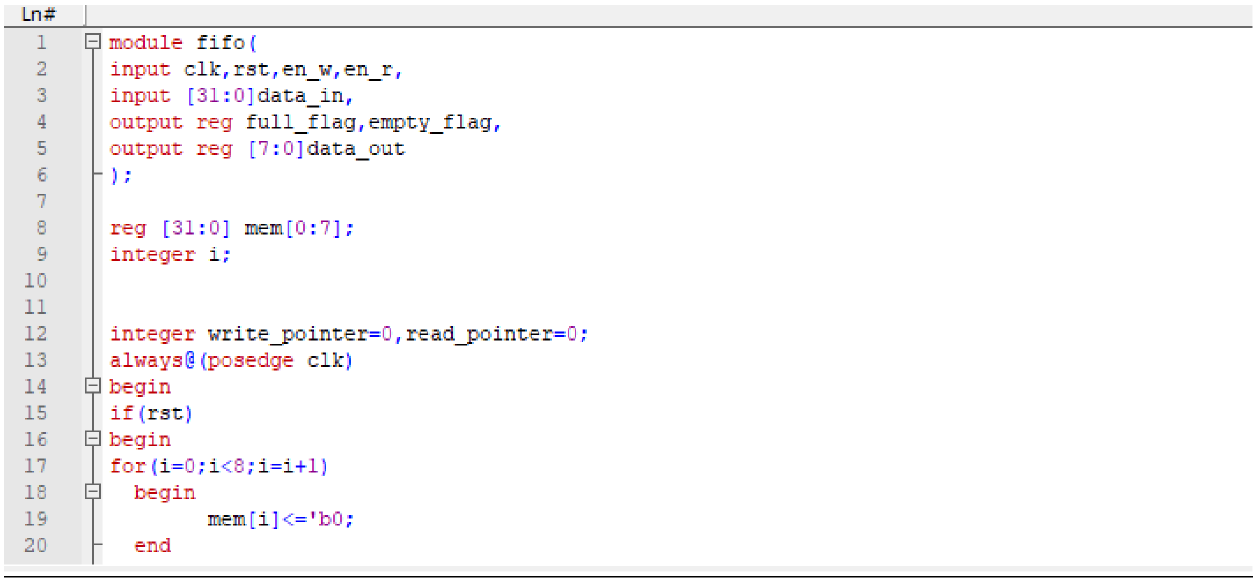
```
module divide_by_6(
input clk,reset,
output reg q
);
integer counter;
always@(clk)
begin

if(reset)
begin
q<=1'b0;
counter<=0;
end
else if(counter==5)
begin
counter<=0;
q<=~q;
```



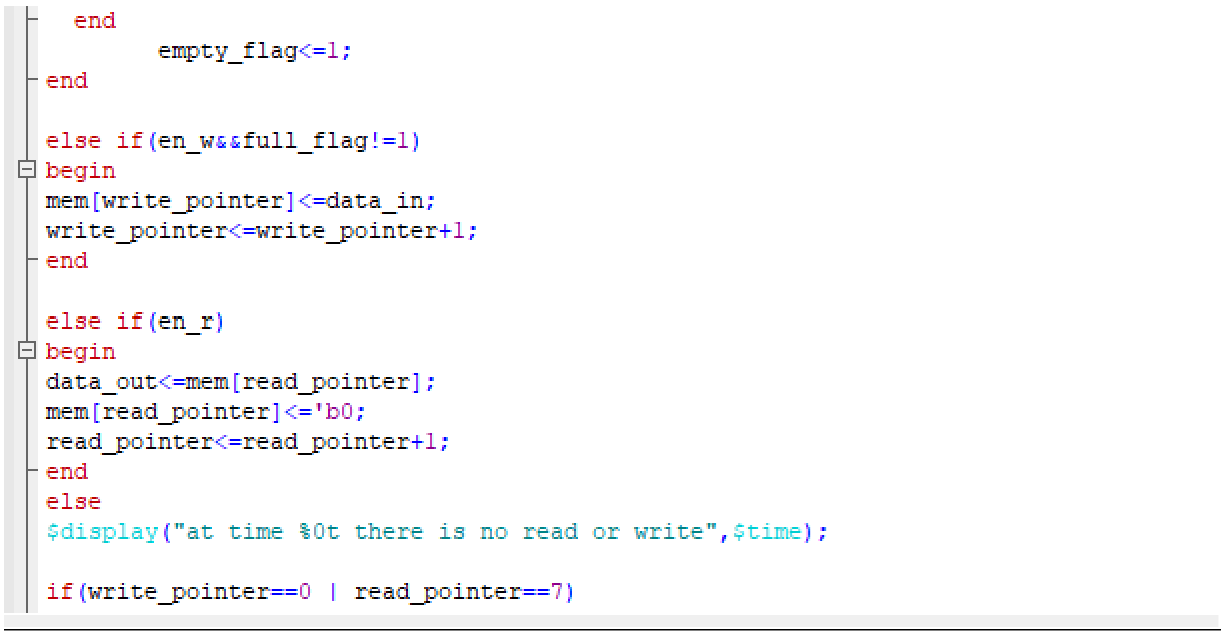
```
q<=~q;
end
else
counter=counter+1;

end


endmodule
```



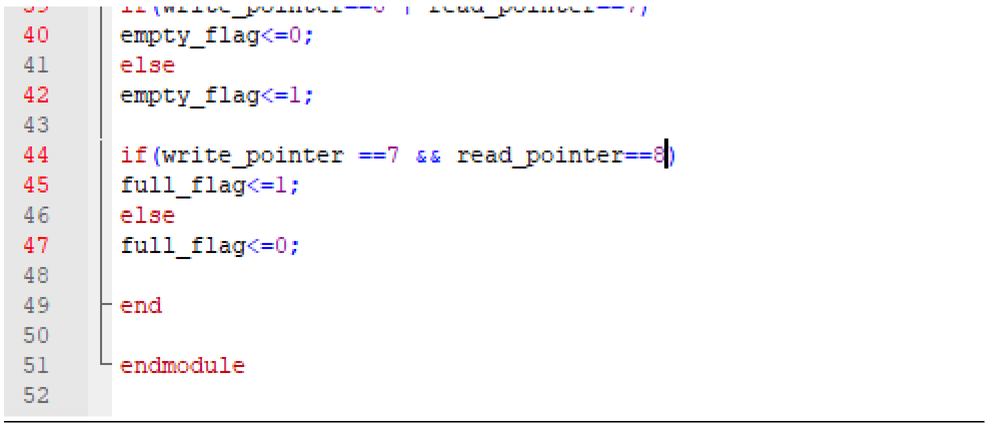
Fifo



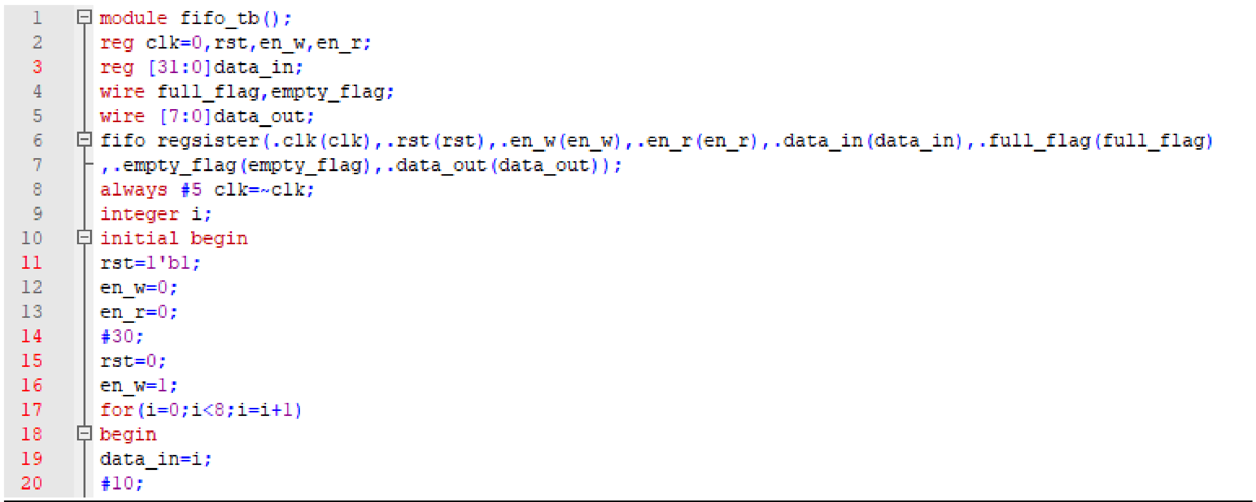
```
module fifo(
input clk,rst,en_w,en_r,
input [31:0]data_in,
output reg full_flag,empty_flag,
output reg [7:0]data_out
);

reg [31:0] mem[0:7];
integer i;


integer write_pointer=0,read_pointer=0;
always@(posedge clk)
begin
if(rst)
begin
for(i=0;i<8;i=i+1)
  begin
        mem[i]<='b0;
  end
```



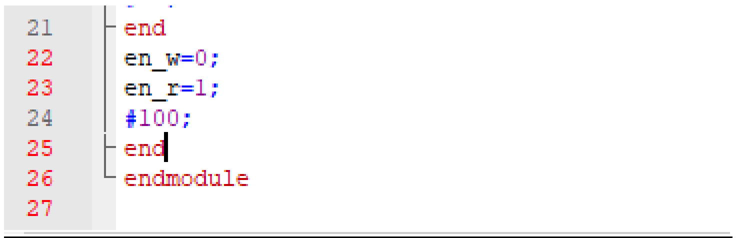
```
  end
        empty_flag<=1;
end

else if(en_w&&full_flag!=1)
begin
mem[write_pointer]<=data_in;
write_pointer<=write_pointer+1;
end

else if(en_r)
begin
data_out<=mem[read_pointer];
mem[read_pointer]<='b0;
read_pointer<=read_pointer+1;
end
else
$display("at time %0t there is no read or write",$time);

if(write_pointer==0 | read_pointer==7)
```



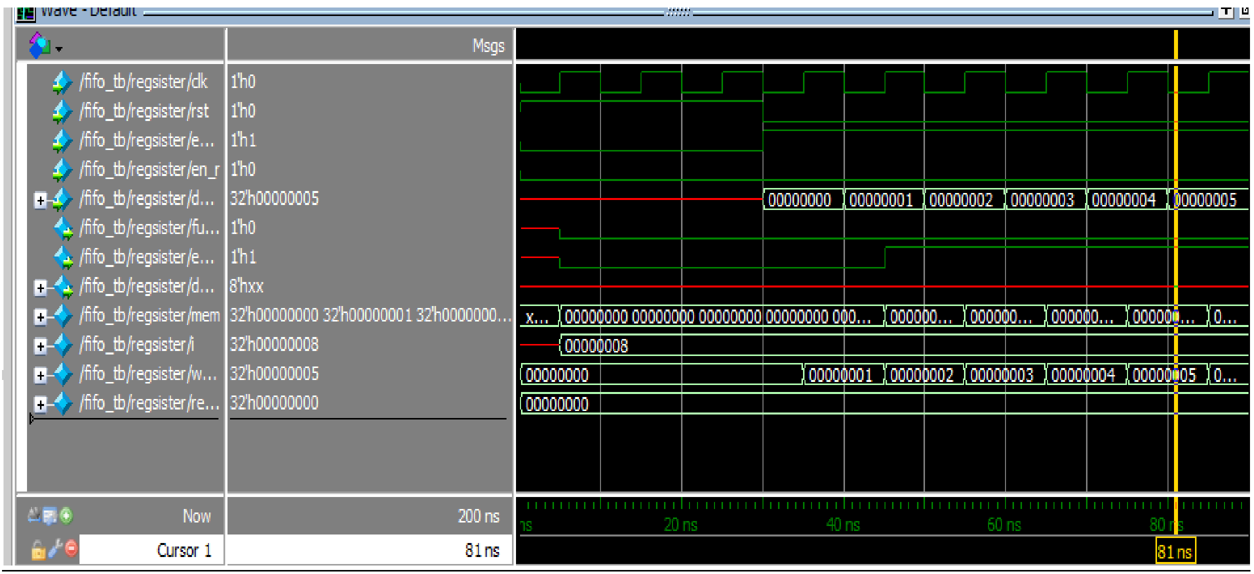
```
empty_flag<=0;
else
empty_flag<=1;

if(write_pointer ==7 && read_pointer==8)
full_flag<=1;
else
full_flag<=0;

end

endmodule
```



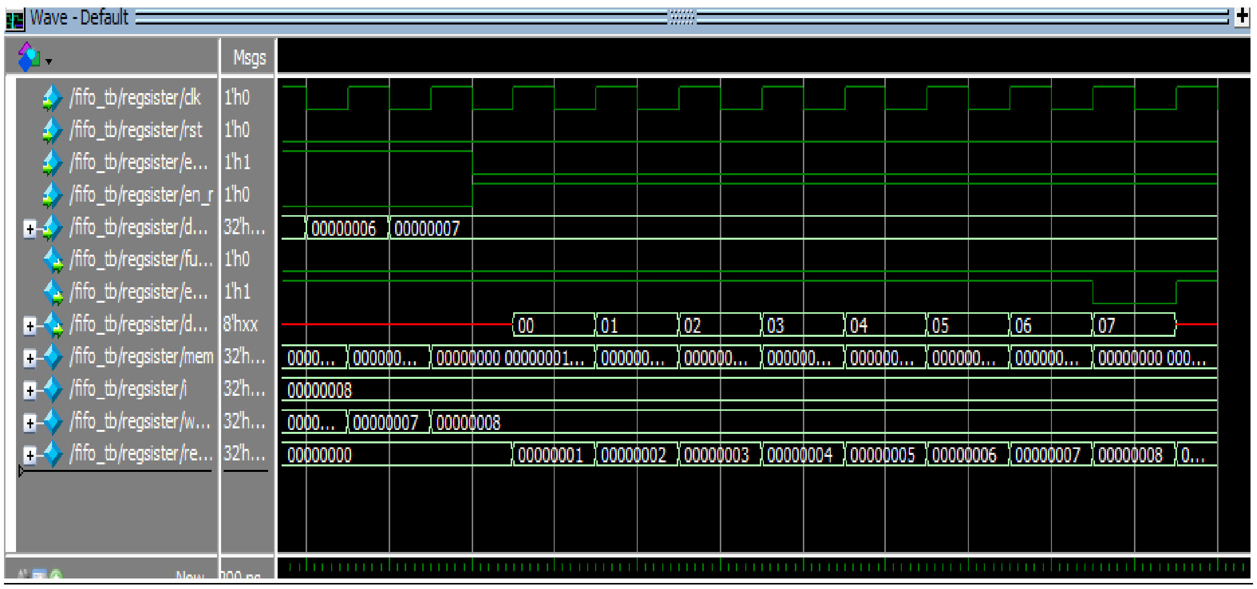
```
module fifo_tb();
reg clk=0,rst,en_w,en_r;
reg [31:0]data_in;
wire full_flag,empty_flag;
wire [7:0]data_out;
fifo regsister(.clk(clk),.rst(rst),.en_w(en_w),.en_r(en_r),.data_in(data_in),.full_flag(full_flag)
,.empty_flag(empty_flag),.data_out(data_out));
always #5 clk=~clk;
integer i;
initial begin
rst=1'b1;
en_w=0;
en_r=0;
#30;
rst=0;
en_w=1;
for(i=0;i<8;i=i+1)
begin
data_in=i;
#10;
```



```
end
en_w=0;
en_r=1;
#100;
end
endmodule
```



```
Wave - Default
                         Msgs
/fifo_tb/regsister/clk   1'h0
/fifo_tb/regsister/rst   1'h0
/fifo_tb/regsister/e...  1'h1
/fifo_tb/regsister/en_r  1'h0
/fifo_tb/regsister/d...  32'h00000005
/fifo_tb/regsister/fu... 1'h0
/fifo_tb/regsister/e...  1'h1
/fifo_tb/regsister/d...  8'hxx
/fifo_tb/regsister/mem   32'h00000000 32'h00000001 32'h0000000...
/fifo_tb/regsister/i     32'h00000008
/fifo_tb/regsister/w...  32'h00000005
/fifo_tb/regsister/re... 32'h00000000

Now       200 ns
Cursor 1  81 ns
```



```
Wave - Default
                         Msgs
/fifo_tb/regsister/clk   1'h0
/fifo_tb/regsister/rst   1'h0
/fifo_tb/regsister/e...  1'h1
/fifo_tb/regsister/en_r  1'h0
/fifo_tb/regsister/d...  32'h...
/fifo_tb/regsister/fu... 1'h0
/fifo_tb/regsister/e...  1'h1
/fifo_tb/regsister/d...  8'hxx
/fifo_tb/regsister/mem   32'h...
/fifo_tb/regsister/i     32'h...
/fifo_tb/regsister/w...  32'h...
/fifo_tb/regsister/re... 32'h...

Now       200 ns
```