

Lap2

Gray to binary

B_2

B_1, B_0

00	0	1
01	0	1
11	0	1
10	0	1

B_2

B_1, B_0

00	0	1
01	0	1
11	1	0
10	1	0

B_2

B_1, B_0

00	0	0
01	0	1
11	0	0
10	1	1

$D_2 = B_2$

$D_2 = B_2 \bar{B}_1 + \bar{B}_2 B_1$

$D_3 = \bar{B}_1 \bar{B}_2 + B_1 B_2$

$D_3 = \bar{B}_1 B_0 + B_1 \bar{B}_0$

$D_3 = \bar{B}_1 B_0 + B_1 \bar{B}_0$

$D_3 = \bar{B}_1 B_0 + B_1 \bar{B}_0$

```
Ln# |
1  | module d_ff(
2  |     input d,clk,reset,
3  |     output reg q
4  | );
5  | //syn
6  | always@(posedge clk)
7  | begin
8  |     if(reset)
9  |         q<=0;
10 |     else
11 |         q<=d;
12 |     end
13 |
14 | endmodule
15 |
```

```

1  module gray(
2      input [2:0]b,
3      input clk,reset,
4      output [2:0]g
5  );
6
7      d_ff d2(.d(b[2]),.clk(clk),.reset(reset),.q(g[2]));
8      d_ff d1(.d(b[2]^b[1]),.clk(clk),.reset(reset),.q(g[1]));
9      d_ff d0(.d(b[0]^b[1]),.clk(clk),.reset(reset),.q(g[0]));
10
11
12  endmodule
13

```

```

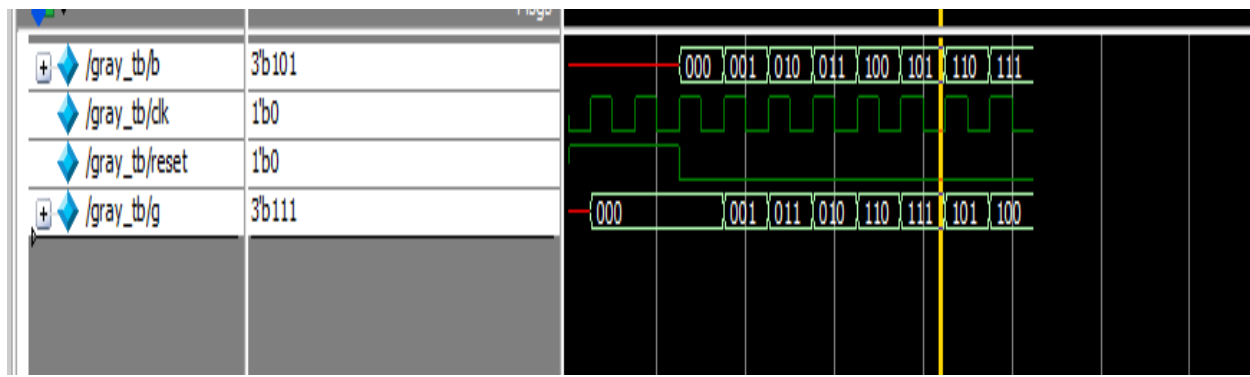
1  module gray_tb();
2      reg [2:0]b;
3      reg clk=0,reset;
4      wire [2:0]g;
5      gray module1(.b(b),.clk(clk),.reset(reset),.g(g));
6      always#5 clk=~clk;
7      initial begin
8          reset=1;
9          #25;
10         reset=0;
11         b=3'b000;
12         #10;
13         b=3'b001;
14         #10;
15         b=3'b010;
16         #10;
17         b=3'b011;
18         #10;
19         b=3'b100;
20         #10;

```

```

21         b=3'b101;
22         #10;
23         b=3'b110;
24         #10;
25         b=3'b111;
26         #10;
27         $stop;
28
29     end
30

```



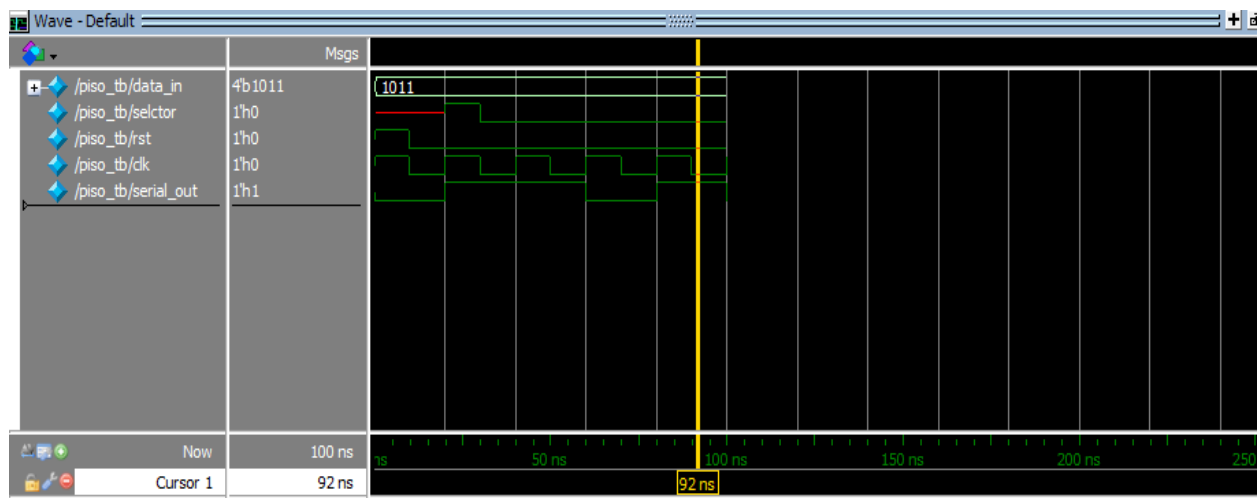
PISO

```

C:/Users/marw/Desktop/ap2/piso.v (/piso_tb/p1) - Default
Ln# 76 92 ns
1  module piso(
2      input [3:0]data_in,
3      input selector,rst,clk,
4      output serial_out
5  );
6      //internal_register
7      reg [3:0]data_come;
8      always@(posedge clk)
9      begin
10         if(rst)
11             data_come<=0;
12         //sel=1 --->load
13         else if(selector)
14             data_come<=data_in;
15         else
16             data_come<=data_come>>1;
17         end
18         assign serial_out=data_come;
19     endmodule

```

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



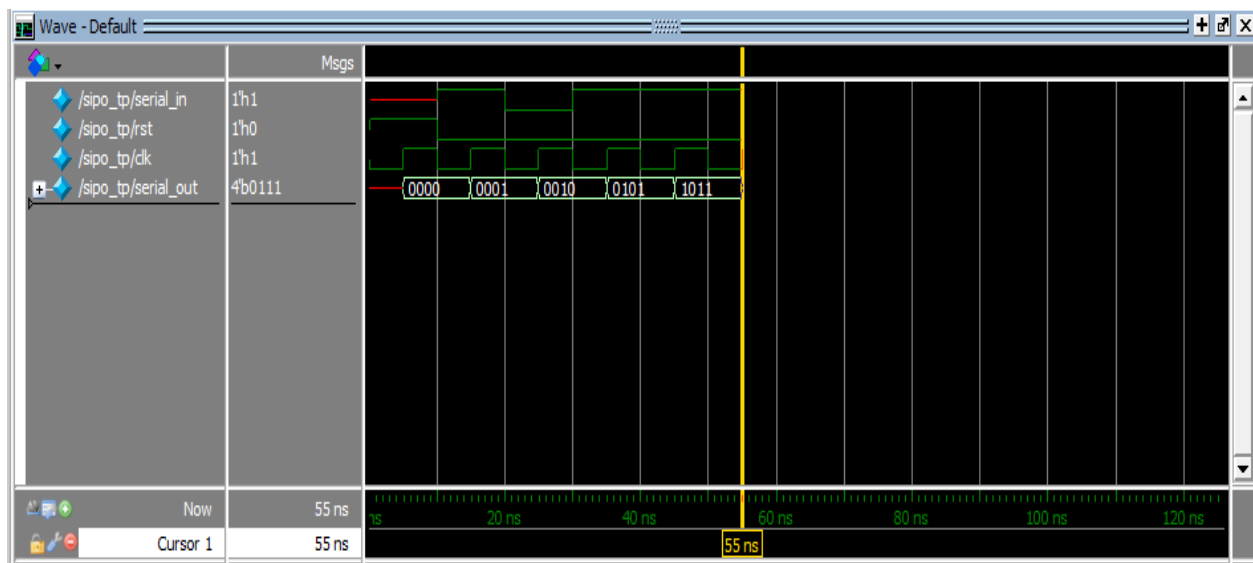
SIPO

Ln#	
1	module d_ff(
2	input d,clk,reset,
3	output reg q
4);
5	//syn
6	always@(posedge clk)
7	begin
8	if(reset)
9	q<=0;
10	else
11	q<=d;
12	end
13	
14	endmodule
15	

Ln#	
1	module sipo(
2	input serial_in,
3	input rst,clk,
4	output [3:0]serial_out
5);
6	//internal_register
7	d_ff d1(.d(serial_in),.clk(clk),.reset(rst),.q(serial_out[0]));
8	d_ff d2(.d(serial_out[0]),.clk(clk),.reset(rst),.q(serial_out[1]));
9	d_ff d3(.d(serial_out[1]),.clk(clk),.reset(rst),.q(serial_out[2]));
10	d_ff d4(.d(serial_out[2]),.clk(clk),.reset(rst),.q(serial_out[3]));
11	
12	endmodule
13	

1	module sipo_tp();
2	reg serial_in;
3	reg rst,clk=0;
4	wire [3:0]serial_out;
5	
6	sipo s1(.serial_in(serial_in),.rst(rst),.clk(clk),.serial_out(serial_out));
7	
8	always #5 clk=~clk;
9	
10	initial begin
11	rst=1;
12	#10;
13	rst=0;
14	serial_in=1;
15	#10;

```
Ln# |
14 | serial_in=1;
15 | #10;
16 | serial_in=0;
17 | #10;
18 | serial_in=1;
19 | #10;
20 | serial_in=1;
21 | #5;
22 | $stop;
23 |
24 | end
25 |
26 | endmodule
27 |
```



FSM

```
11#
1  module fsm(
2      input reset,taken,clk,
3      output reg predict
4  );
5      localparam A=2'b00;
6      localparam B=2'b01;
7      localparam C=2'b10;
8      localparam D=2'b11;
9      reg [1:0] next_state,present_state;
10     always@(posedge clk)
11     begin
12         if(reset)
13             present_state<=0;
14         else
15             present_state<=next_state;
16         end
17     always@(*)
18     begin
19         case(present_state)
20             A:
```

```
20         A:
21             begin
22                 if(taken==1)
23                     begin
24                         next_state=A;
25                         predict=1'b1;
26                     end
27                 else
28                     begin
29                         next_state=B;
30                         predict=1'b1;
31                     end
32             end
33         B:
34             begin
35                 if(taken==1)
36                     begin
37                         next_state=A;
38                         predict=1'b1;
39                     end
```

```
39         end
40     else
41     begin
42         next_state=C;
43         predict=1'b0;
44     end
45 end
46 C:
47 begin
48     if(taken==1)
49     begin
50         next_state=D;
51         predict=1'b0;
52     end
53     else
54     begin
55         next_state=C;
56         predict=1'b0;
57     end
58 end
```

```
57         end
58     end
59 D:
60 begin
61     if(taken==1)
62     begin
63         next_state=A;
64         predict=1'b1;
65     end
66     else
67     begin
68         next_state=C;
69         predict=1'b0;
70     end
71 end
72 default:
73 begin
74     next_state=present_state;
75     present_state=A;
76 end
```


FSM2

```
C:/Users/marw/Desktop/lap2/fsm2.v (/fsm2_tb/module1) - Default
Ln#
1  module fsm2(
2      input reset,go,clk,
3      output reg op
4  );
5
6      localparam A=3'b000;
7      localparam B=3'b001;
8      localparam C=3'b010;
9      localparam D=3'b011;
10     localparam E=3'b100;
11     localparam F=3'b101;
12     localparam G=3'b110;
13     localparam H=3'b111;
14
15     reg [2:0] next_state,present_state;
16
17     always@(posedge clk)
18     begin
19         if(reset)
20
21             present_state<=0;
22         else
23             present_state<=next_state;
24         end
25     always@*
26     begin
27         case(present_state)
28         A:
29             begin
30                 if(go)
31                 begin
32                     next_state=B;
33                     op=1'b1;
34                 end
35             else
36             begin
37                 next_state=A;
38                 op=1'b0;
39             end
40         endcase
41     end
42 end
```

```
41 B:
42   begin
43       if(go)
44           begin
45               next_state=B;
46               op=1'b0;
47           end
48       else
49           begin
50               next_state=C;
51               op=1'b0;
52           end
53       end
54   C:
55       begin
56           if(go)
57               begin
58                   next_state=D;
59                   op=1'b0;
60               end
```

```
53   end
54   C:
55       begin
56           if(go)
57               begin
58                   next_state=D;
59                   op=1'b0;
60               end
61           else
62               begin
63                   next_state=A;
64                   op=1'b0;
65               end
66       end
67   D:
68       begin
69           if(go)
70               begin
71                   next_state=E;
```

```
67 D:
68   begin
69     if(go)
70     begin
71       next_state=E;
72       op=1'b0;
73     end
74   else
75   begin
76     next_state=C;
77     op=1'b0;
78   end
79 end
80
81 E:
82   begin
83     if(go)
84     begin
85       next_state=B;
86       op=1'b0;
```

```
86       op=1'b0;
87     end
88   else
89   begin
90     next_state=F;
91     op=1'b0;
92   end
93 end
94 F:
95   begin
96     if(go)
97     begin
98       next_state=G;
99       op=1'b0;
100    end
101   else
102   begin
103     next_state=A;
104     op=1'b0;
105   end
```

```
105         end
106     end
107     G:
108     begin
109         if (go)
110         begin
111             next_state=E;
112             op=1'b0;
113         end
114         else
115         begin
116             next_state=H;
117             op=1'b0;
118         end
119     end
120     H:
121     begin
122         if (go)
123         begin
124             next_state=D;
```

```
124             next_state=D;
125             op=1'b1;
126         end
127         else
128         begin
129             next_state=A;
130             op=1'b0;
131         end
132     end
133     default:
134     begin
135         next_state=present_state;
136         present_state=A;
137     end
138 endcase
139 end
140
141 endmodule
```

```

1 module fsm2_style2(
2     input reset,go,clk,
3     output reg op
4 );
5
6     localparam A=3'b000;
7     localparam B=3'b001;
8     localparam C=3'b010;
9     localparam D=3'b011;
10    localparam E=3'b100;
11    localparam F=3'b101;
12    localparam G=3'b110;
13    localparam H=3'b111;
14
15    reg [2:0] next_state,present_state;
16
17    always@(posedge clk)
18    begin
19        if(reset)
20            present_state<=0;
21        else

```

```

21        else
22            present_state<=next_state;
23        end
24
25        always@*
26        begin
27            case (present_state)
28            A:
29                begin
30                    if(go)
31                        next_state=B;
32                    else
33                        next_state=A;
34                end
35            B:
36                begin
37                    if(go)
38                        next_state=B;
39                    else
40                        next_state=C;
41                end

```

```

41            end
42            C:
43                begin
44                    if(go)
45                        next_state=D;
46                    else
47                        next_state=A;
48                end
49            D:
50                begin
51                    if(go)
52                        next_state=E;
53                    else
54                        next_state=C;
55                end
56            E:
57                begin
58                    if(go)
59                        next_state=B;
60                    else
61                        next_state=F;

```

```

62     end
63   F:
64   begin
65     if(go)
66       next_state=G;
67     else
68       next_state=A;
69     end
70   G:
71   begin
72     if(go)
73       next_state=E;
74     else
75       next_state=H;
76     end
77   H:
78   begin
79     if(go)
80       next_state=D;
81     else

```

```

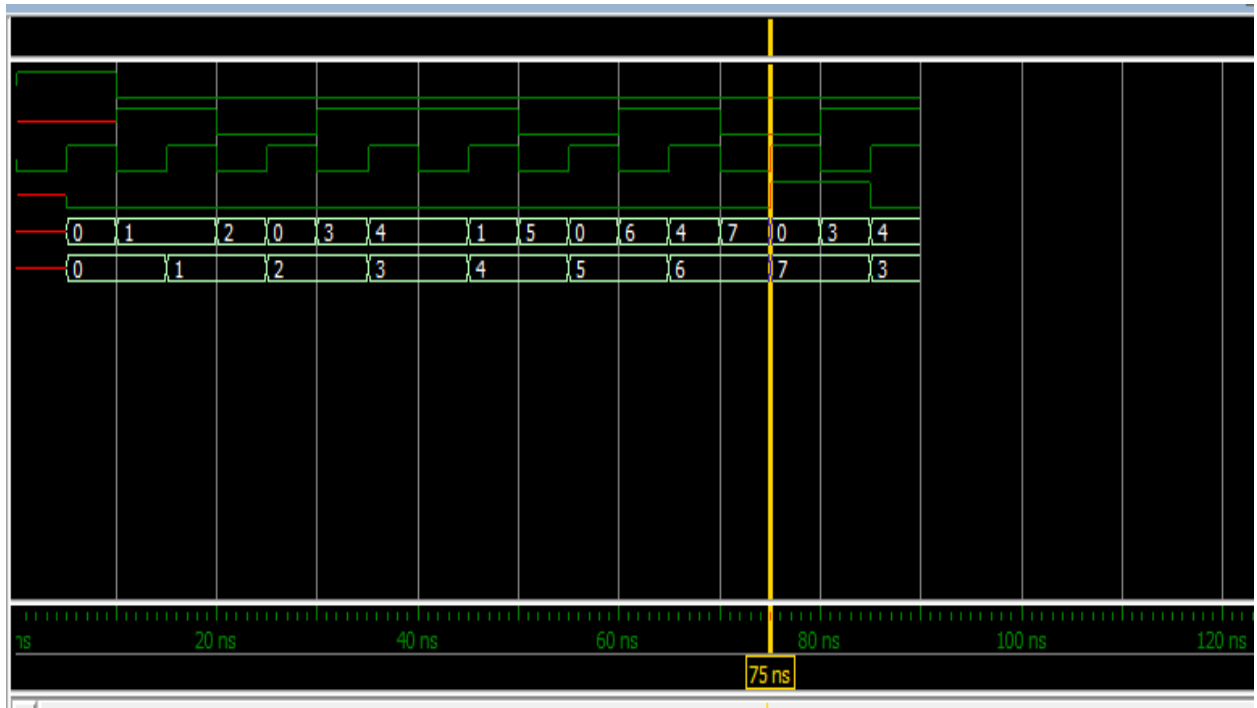
82       next_state=A;
83     end
84   default:
85   begin
86     next_state=present_state;
87     present_state=A;
88   end
89   endcase
90   end
91
92   always@*
93   begin
94
95     case (present_state)
96     A:op=0;
97     B:op=0;
98     C:op=0;
99     D:op=0;
100    E:op=0;
101    F:op=0;
102    G:op=0;
103    H:op=1;

```

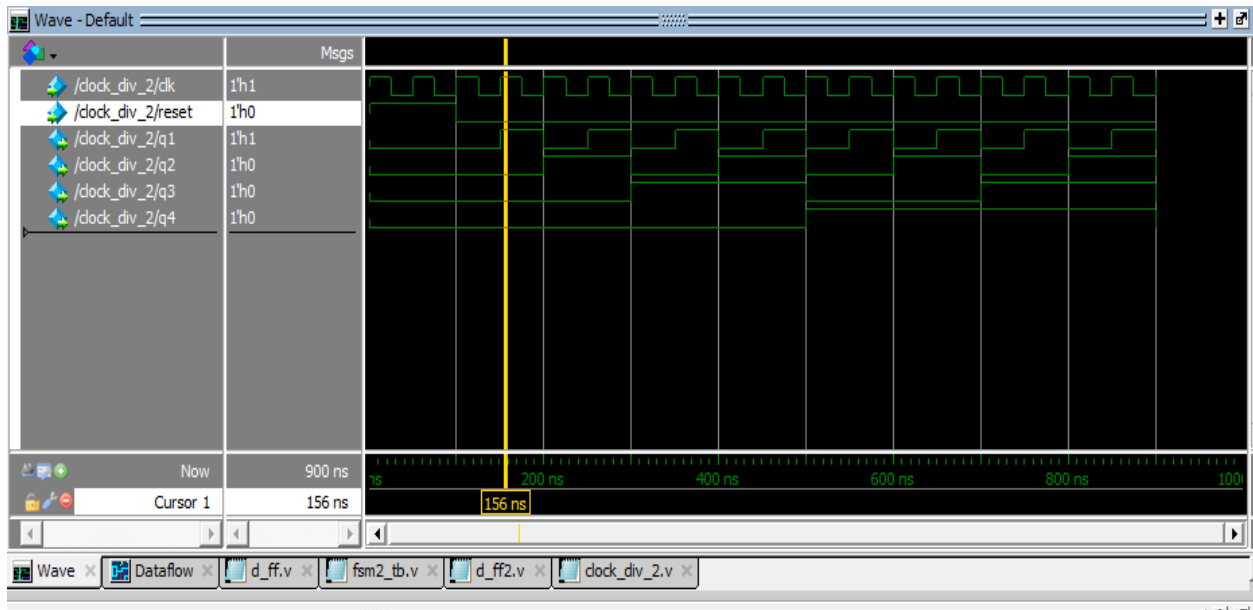
```

92   always@*
93   begin
94
95   case (present_state)
96   A:op=0;
97   B:op=0;
98   C:op=0;
99   D:op=0;
100  E:op=0;
101  F:op=0;
102  G:op=0;
103  H:op=1;
104  endcase
105
106  end
107
108  endmodule
109

```



CLOCK DIV BY 2,4,8,16



```
Ln#
1  module d_ff(
2      input d,clk,reset,
3      output reg q
4  );
5      //syn
6      always@(posedge clk)
7      begin
8          if(reset)
9              q<=0;
10         else
11             q<=d;
12         end
13     end
14     endmodule
15
```

```

1  module clock_div_2(
2      input clk,reset,
3      output q1,q2,q3,q4
4  );
5
6      d_ff2 ff1(.d(!q1),.clk(clk),.q(q1),.reset(reset));
7      d_ff2 ff2(.d(!q2),.clk(!q1),.q(q2),.reset(reset));
8      d_ff2 ff3(.d(!q3),.clk(!q2),.q(q3),.reset(reset));
9      d_ff2 ff4(.d(!q4),.clk(!q3),.q(q4),.reset(reset));
10  endmodule
11
12

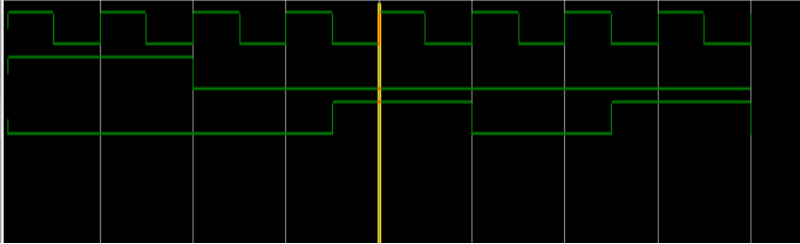
```

Div by 3

```

1  module divde3(
2      input clk,reset,
3      output reg q
4  );
5      integer counter;
6      always@(clk)
7      begin
8
9          if(reset)
10         begin
11             q<=1'b0;
12             counter<=0;
13         end
14         else if(counter==2)
15         begin
16             counter<=0;
17             q<=~q;
18
19         end
20         else
21             counter=counter+1;
22     end
23
24
25  endmodule
26

```

	Msgs	
/divide3/clk	1'h1	
/divide3/reset	1'h0	
/divide3/q	1'h1	

Div by 6

```

Ln#
1  module divide_by_6(
2      input clk,reset,
3      output reg q
4  );
5      integer counter;
6      always@(clk)
7      begin
8
9          if(reset)
10         begin
11             q<=1'b0;
12             counter<=0;
13         end
14         else if(counter==5)
15         begin
16             counter<=0;
17             q<=~q;

```

```

18         end
19         else
20             counter=counter+1;
21
22         end
23
24
25     endmodule
26

```


Fifo

Ln#	
1	module fifo(
2	input clk,rst,en_w,en_r,
3	input [31:0]data_in,
4	output reg full_flag,empty_flag,
5	output reg [7:0]data_out
6);
7	
8	reg [31:0] mem[0:7];
9	integer i;
10	
11	
12	integer write_pointer=0,read_pointer=0;
13	always@(posedge clk)
14	begin
15	if(rst)
16	begin
17	for(i=0;i<8;i=i+1)
18	begin
19	mem[i]<='b0;
20	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)

```

39 | if(write_pointer==0 || read_pointer==7)
40 |     empty_flag<=0;
41 | else
42 |     empty_flag<=1;
43 |
44 | if(write_pointer ==7 && read_pointer==8)
45 |     full_flag<=1;
46 | else
47 |     full_flag<=0;
48 |
49 | end
50 |
51 | endmodule
52 |

```

```

1 | module fifo_tb();
2 |     reg clk=0,rst,en_w,en_r;
3 |     reg [31:0]data_in;
4 |     wire full_flag,empty_flag;
5 |     wire [7:0]data_out;
6 |     fifo regfister(.clk(clk),.rst(rst),.en_w(en_w),.en_r(en_r),.data_in(data_in),.full_flag(full_flag)
7 |     ,.empty_flag(empty_flag),.data_out(data_out));
8 |     always #5 clk=~clk;
9 |     integer i;
10 |    initial begin
11 |        rst=1'b1;
12 |        en_w=0;
13 |        en_r=0;
14 |        #30;
15 |        rst=0;
16 |        en_w=1;
17 |        for(i=0;i<8;i=i+1)
18 |        begin
19 |            data_in=i;
20 |            #10;

```

```

21 |        end
22 |        en_w=0;
23 |        en_r=1;
24 |        #100;
25 |    end
26 | endmodule
27 |

```

