Lab6
Submitted By :
Snehal Gupta
2016201

Moore Circuit :

Clock divider :
//clk_div.v

```
`timescale 1ns / 1ps
module clk_div(
input wire mclk,
output wire clk190);
reg [18:0] q;
always @(posedge mclk)
begin
q <= q+1;
end
assign clk190 = q[18];
endmodule
```

Debouncer :
//debounce.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04.10.2017 16:46:57
// Design Name:
// Module Name: debounce
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
```

```
module debounce(
    input clk_in,
    input clr_in,
    output clr_out
    );
    reg D1,D2,D3;
    always@(posedge clk_in)
    begin
    D1 <= clr_in;
    D2 <= D1;
    D3 <= D2;
    end
    assign clr_out = D1 && D2 && D3;
endmodule
```

Clock_pulse generator:
//clock_pulse.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 26.09.2017 18:04:34
// Design Name:
// Module Name: clock_pulse
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module clock_pulse(
input wire inp,
input wire cclk,
input wire clr,
output wire outp);
reg delay1;
reg delay2;
reg delay3;
always@(posedge cclk or posedge clr)
begin
if(clr == 1)
begin
delay1 <= 1'b0;
delay2 <= 1'b0;
delay3 <= 1'b0;
end
else
begin
delay1 <= inp;
delay2 <= delay1;
delay3 <= delay2;
end
end
assign outp = delay1 & delay2 & delay3;
endmodule
```

Moore circuit:
//moore.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04.10.2017 16:54:21
// Design Name:
// Module Name: moore
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
```

```
///////////////////////////////////////////////////////////////////////////////////


module moore(
    input clk,
    input clr,
    input din,
    output reg dout,
    output reg [2:0] present_state
    );
    reg[2:0] next_state;
    parameter S0=3'b000 , S1=3'b001 , S2=3'b010 , S3=3'b011 , S4=3'b100 , S5=3'b101 ;
    always@(posedge clk or posedge clr)
    begin
    if(clr == 1)
    present_state <= S0;
    else
    present_state <= next_state;
    end
    always@(*)
    begin
    case(present_state)
    S0 : if(din == 1)
            next_state <= S1;
         else
            next_state <= S0;
    S1 : if(din == 0)
            next_state <= S2;
         else
            next_state <= S1;
    S2 : if(din == 1)
            next_state <= S3;
         else
            next_state <= S0;
    S3 : if(din == 0)
            next_state <= S4;
         else
            next_state <= S1;
    S4 : if(din == 1)
            next_state <= S5;
         else
            next_state <= S0;
    S5 : if(din == 0)
```

```
            next_state <= S0;
         else
            next_state <= S1;
    default next_state <= S0;
    endcase
    end
    always@(*)
    begin
    if(present_state == S5)
        dout <= 1;
    else
        dout <= 0;
    end

endmodule
```

Top module:

//top.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04.10.2017 17:15:19
// Design Name:
// Module Name: top
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module top(
    input clk,
    input clr,
    input [1:0] btn,
    output [1:0] led,
    output [2:0] present_state
    );
    wire inp;
    assign inp= btn[0] || btn[1];
    clk_div CD1(.mclk(clk),.clk190(clk_190));
    debounce D1(.clk_in(clk_190),.clr_in(clr),.clr_out(clr_de));
    clock_pulse CP(.inp(inp),.cclk(clk_190),.clr(clr_de),.outp(out_pulse));
    moore M1(.clk(out_pulse),.clr(clr_de),.din(btn[1]),.dout(led[0]),.present_state(present_state));
    assign led[1] = out_pulse;
endmodule
```

Mealy circuit :

//Clock_divider,clock_pulse and debouncer remains the same.

//top.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04.10.2017 17:15:19
// Design Name:
// Module Name: top
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module top(
    input clk,
    input clr,
    input [1:0] btn,
    output [1:0] led,
    output [2:0] present_state
    );
    wire inp;
    assign inp= btn[0] || btn[1];
    clk_div CD1(.mclk(clk),.clk190(clk_190));
    debounce D1(.clk_in(clk_190),.clr_in(clr),.clr_out(clr_de));
    clock_pulse CP(.inp(inp),.cclk(clk_190),.clr(clr_de),.outp(out_pulse));
    mealy M1(.clk(out_pulse),.clr(clr_de),.din(btn[1]),.dout(led[0]),.present_state(present_state));
    assign led[1] = out_pulse;
endmodule
```

//mealy.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04.10.2017 16:54:21
// Design Name:
// Module Name: moore
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module mealy(
    input clk,
    input clr,
    input din,
    output reg dout,
    output [2:0] present_state
    );
    reg[2:0] next_state;
    parameter S0=3'b000 , S1=3'b001 , S2=3'b010 , S3=3'b011 , S4=3'b100;
    always@(posedge clk or posedge clr)
    begin
    if(clr == 1)
```

```
begin
next_state <= S0;
dout <= 0;
end
else
case(present_state)
S0 : if(din == 1)
        begin
        next_state <= S1;
        dout <= 0;
        end
     else
     begin
        next_state <= S0;
        dout <= 0;
     end
S1 : if(din == 0)
        begin
        next_state <= S2;
        dout <= 0;
        end
     else
        begin
        next_state <= S1;
        dout <= 0;
        end
S2 : if(din == 1)
        begin
        next_state <= S3;
        dout <=0;
        end
     else
        begin
        next_state <= S0;
        dout <= 0;
        end
S3 : if(din == 0)
        begin
        next_state <= S4;
        dout <= 0;
        end
     else
        begin
        next_state <= S1;
        dout <= 0;
        end
S4 : if(din == 1)
    begin
        next_state <= S0;
        dout <= 1;
```

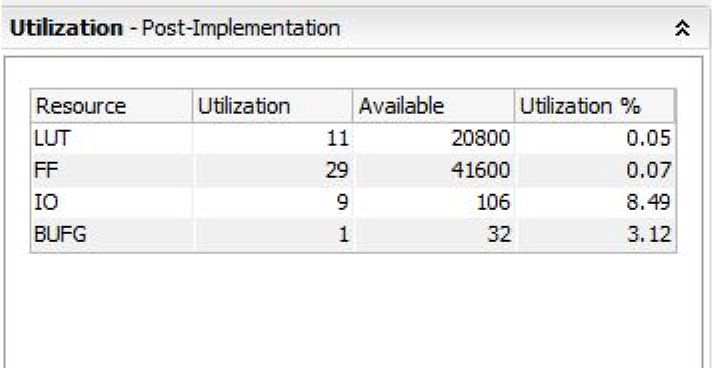
```
            end
         else
         begin
            next_state <= S0;
            dout <= 0;
            end
    default:
begin
 next_state <= S0;
dout <= 0;
end
    endcase
    end
    assign present_state=next_state;

endmodule
```

//const.xdc

```
set_property PACKAGE_PIN W5 [get_ports {clk}]
set_property IOSTANDARD LVCMOS33 [get_ports {clk}]
set_property PACKAGE_PIN T17 [get_ports {clr}]
set_property IOSTANDARD LVCMOS33 [get_ports {clr}]
set_property PACKAGE_PIN W19 [get_ports {btn[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {btn[1]}]
set_property PACKAGE_PIN U18 [get_ports {btn[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {btn[0]}]
set_property PACKAGE_PIN L1 [get_ports {led[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
set_property PACKAGE_PIN P1 [get_ports {led[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
set_property PACKAGE_PIN N3 [get_ports {present_state[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {present_state[2]}]
set_property PACKAGE_PIN P3 [get_ports {present_state[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {present_state[1]}]
set_property PACKAGE_PIN U3 [get_ports {present_state[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {present_state[0]}]
```

Conclusion :

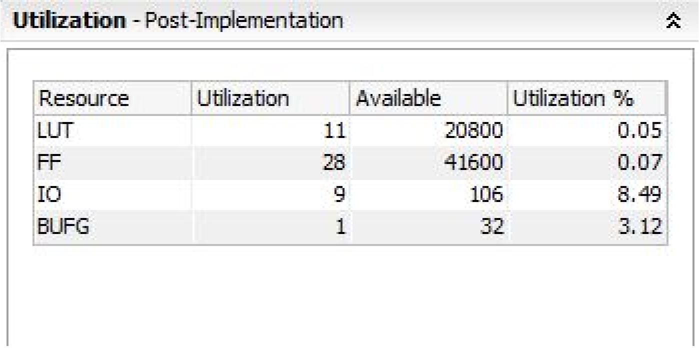
Mealy circuit :



Utilization - Post-Implementation

Resource	Utilization	Available	Utilization %
LUT	11	20800	0.05
FF	29	41600	0.07
IO	9	106	8.49
BUFG	1	32	3.12

Moore circuit :



Utilization - Post-Implementation

Resource	Utilization	Available	Utilization %
LUT	11	20800	0.05
FF	28	41600	0.07
IO	9	106	8.49
BUFG	1	32	3.12

Mealy consumes one flip-flop more than moore circuit .