LAB 6.1
SARTHAK BHAGAT
2016189

TOP

```
`timescale 1ns / 1ps
module ttop(
input clk,
input clr,
input [2:0] btn,
input [7:0] sw,
output [1:0] led,
output [3:0] present

);
wire inp, dout,clk_190;
wire [1:0] din;
reg [1:0] d;
reg x;
reg y;
assign inp = btn[0] || btn[1] || btn[2];
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));
dl
m1(.clk_190(clk_190),.clk(out_pulse),.clr(clr_de),.din(din),.sw(sw),.dout(led),.present_state(pres
ent));
always @ (posedge clk)
begin
if (btn[0] == 1)
d <= 2'b00;
else if (btn[1] == 1)
d <= 2'b01;
else if (btn[2] == 1)
begin
d <= 2'b10;
end
end
assign din = d;
Endmodule
```

CLK DIVIDER

```
`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

```
`timescale 1ns / 1ps
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
```

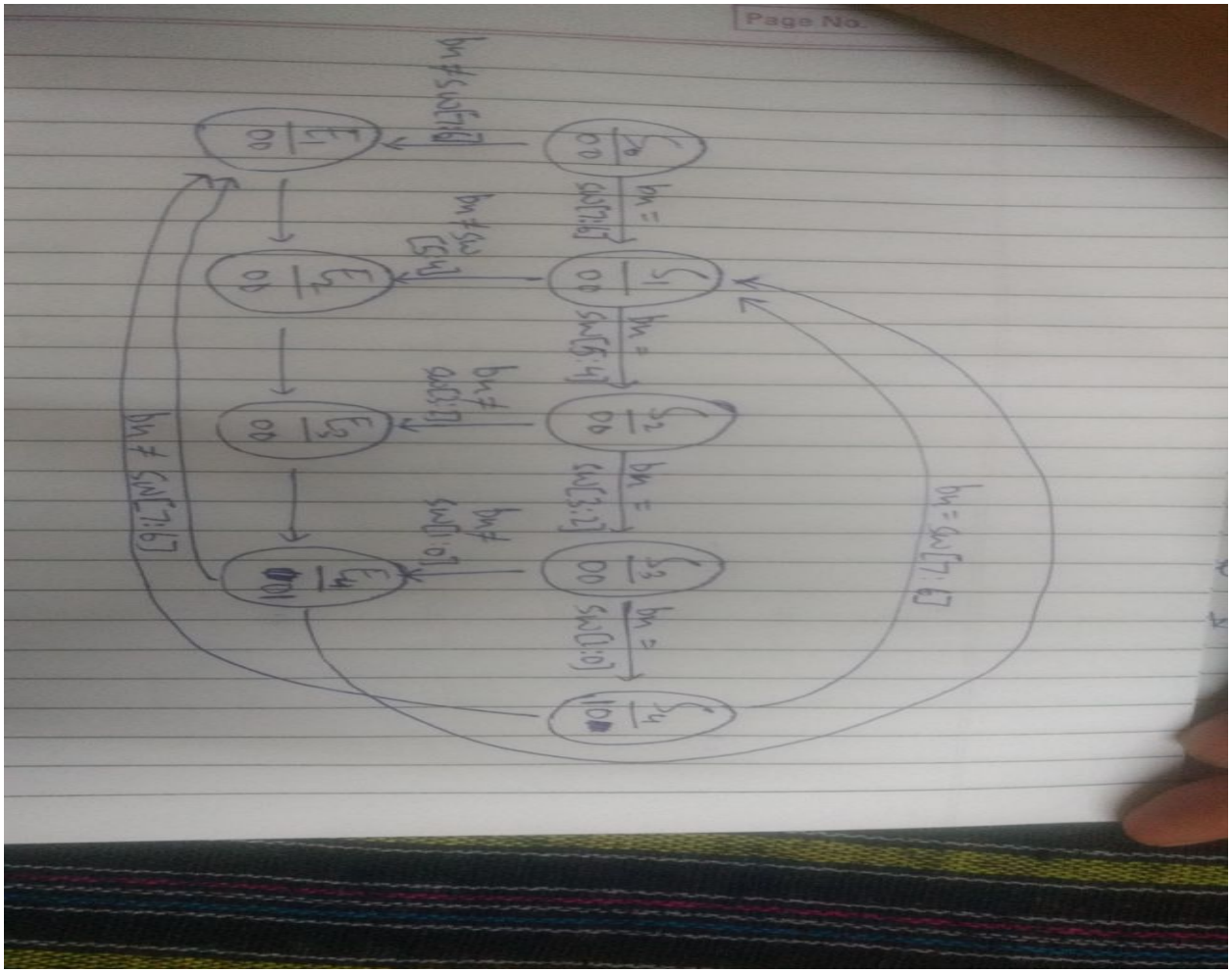
CLOCK PULSE

```
`timescale 1ns / 1ps
module clock_pulse(
input wire inp,
input wire cclk,
input wire clr,
output wire outp
);
reg delay1;
reg delay2;
reg delay3;
```

```
always @ (posedge cclk)
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
```

FSM FOR DOOR LOCK



```
`timescale 1ns / 1ps
module dl(
input clk_190,
input wire clk,
input wire clr,
input wire [1:0] din,
input [7:0] sw,
output reg [1:0] dout,
output reg [3:0] present_state
);
reg [3:0] next_state;
reg [1:0]dout=0;
parameter S0 = 4'b0000, S1 = 4'b0001, S2 = 4'b0010, S3 = 4'b0011, S4 = 4'b0100, E1 =
4'b0101, E2 = 4'b0110, E3 = 4'b0111, E4 = 4'b1000;
always @ (posedge clk or posedge clr)
begin
if (clr ==1)
```

```
begin
present_state <= S0;
end
else
present_state <= next_state;
end
always @ (*)
begin
case(present_state)
S0: if (din == sw[7:6])
next_state <= S1;
else
next_state <= E1;
S1: if (din == sw[5:4])
next_state <= S2;
else
next_state <= E2;
S2: if (din == sw[3:2])

next_state <= S3;
else
next_state <= E3;
S3: if (din == sw[1:0])
next_state <= S4;
else
next_state <= E4;
S4: if (din == sw[7:6])
next_state <= S1;
else
next_state <= E1;
E1:
next_state <= E2;
E2:
next_state <= E3;
E3:
next_state <= E4;
E4: if (din == sw[7:6])
next_state <= S1;
else
next_state <= E1;
default next_state <= S0;
endcase
end
```

```
always @ (posedge clk_190)
begin
if (clr==1)
begin
dout <= 2'b00;
end
else if ( present_state == S4)
dout <= 2'b10;
else if (present_state == E4)
dout <= 2'b01;
else
dout <= 2'b00;
end
Endmodule
```

EXPLANATION OF CODE:
There are 5 states in total for normal FSM while 4 pseudo states that help us to count the number of digits that have been entered. In case of a digit entered wrong we enter the E1-4 states and then get the output as wrong. In case we reach the state S4 we get the correct output for which we will have to enter all digits as correct.

Normal states for FSM:
S0 - The initial state. Next state will be S1 if the button input corresponds to the one set by switches[7 and 6], else the next state would be E1.
S1 - Next state will S2 if the button input corresponds to the one set by the switches[5 and 4], else the next state would be E2.
S2 - Next state will S3 if the button input corresponds to the one set by the switches[3 and 2], else the next state would be E3.
S3 - Next state will S4 if the button input corresponds to the one set by the switches[1 and 0], else the next state would be E4.
S4 - At this stage LED_CORRECT will light up. Next state will S1 if the button input corresponds to the one set by the switches[7 and 6], else the next state would be E1.

Pseudo states for wrong output:
E1 - Next stage will be E2.
E2 - Next stage will be E3.
E3 - Next stage will be E4.
E4 - At this stage LED_WRONG will light up. Next state will S1 if the button input corresponds to the one set by the switches[7 and 6], else the next state would be E1.