

University of Engineering and Technology, Peshawar

Department of Computer Systems Engineering.

Course Lab: CSE-308 Digital System Design

Zubair Khan
Section
Batch
Submitted to



19 PWCSE 1797
A
21 (Spring_2022)
Engr. Madiha Sher

LAB # 7 TITLE

DESIGN OF A LIGHT SWITCH

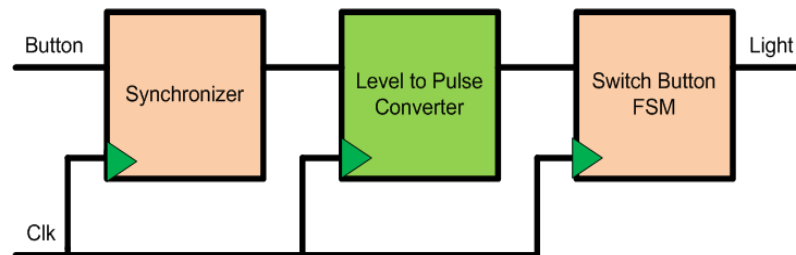
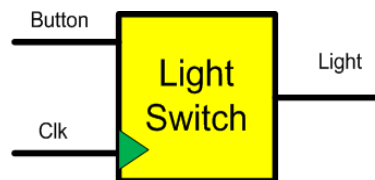
Objective:

Build a light switch controller such that when the push button is pressed the light if "off" turns "on" and if "on" turns "off".

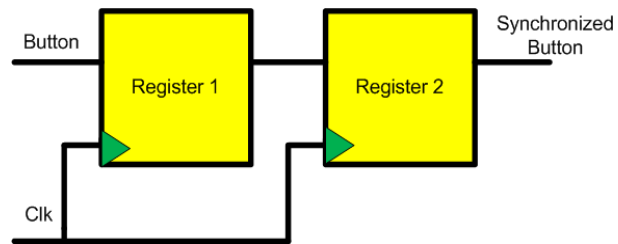
Block Diagram:

The top level block diagram is shown in the following figure. The second diagram is a more detailed view of the top level diagram. Synchronizer circuit is used to avoid the metastability problem which arises when synchronous digital system are fed with an asynchronous input. The level to pulse converter is there to convert level input from a push button into a pulse that is high for only one clock cycle. The switch button FSM is actually implementing the state machine for the requirement i.e. when the push button is pressed the light if “off” turns “on” and if “on” turns “off”.

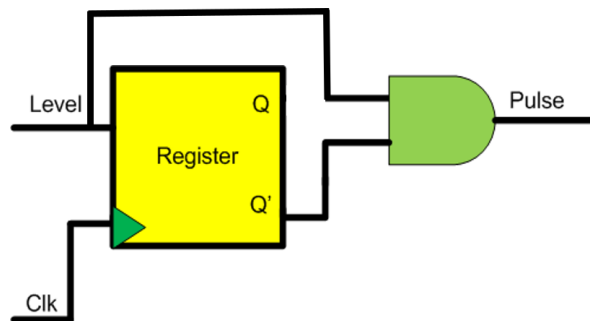
I/O connection: Connect the button input to a push button on the S3BOARD and light output to LED.



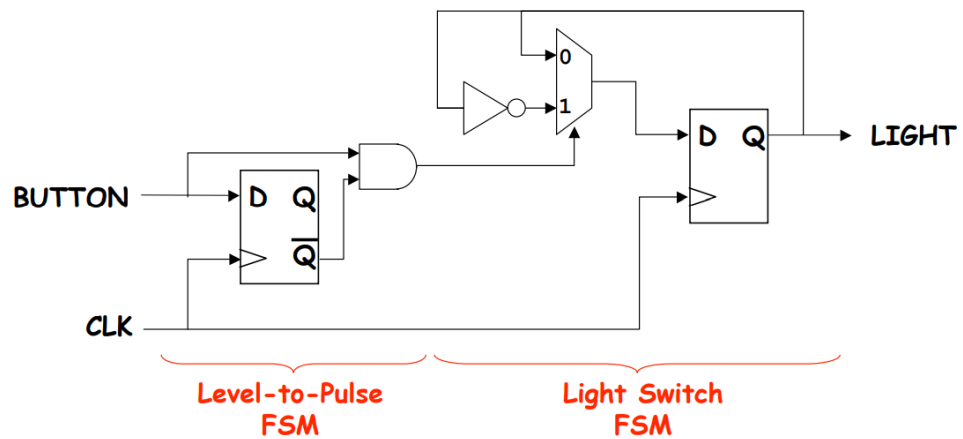
Synchronizer Circuit:



Level to Pulse FSM:



Switch Button FSM:



CODE:

```
module task2(clk,rst,in,out,pulse);

input clk,rst,in;
output
out,pulse;
wire synin, CLK_1hz;

CLOCK_Divider CD1(clk,rst,CLK_1hz);
synchron ss2(in,CLK_1hz,rst,synin); task1
t1(clk,rst,synin,pulse);

parameter
s0=0;parameter
s1=1;
reg state, next_state,out;

always @(posedge CLK_1hz)
begin
    if(~rst)
        begin
            state=s0;
        end
    else
        state=next_state;
end
always @(*)
begin
    case(state)
        s0:
            begin
                if(pulse==0)
                    begin
                        next_state=s0
                        ;out=0;
                    end
                else
                    begin
                        next_state=s1
                        ;out=1;
                    end
            end
        s1:
            begin
                if(pulse==1)
                    begin
                        next_state=s0
                        ;out=0;
                    end
                else
                    begin
```

```

                                out=1;
                                end
                                end
                                endcas
                                end
                                e
                                endmodule

//designing the pulse
generatormodule
task1(clk,rst,in,out); input clk,
rst,in;
output out;
wire synin, CLK_1hz;

CLOCK_Divider CD(clk,rst,CLK_1hz);
synchron ss1(in,CLK_1hz,rst,synin);

parameter
s0=0;parameter
s1=1;
reg state, next_state,out;

always @(posedge CLK_1hz)
begin
    if(~rst)
    begin
        state=s0
        ;end
    else
        state=next_state;
end
always @(*)
begin
    case(state)
        s0:
        begin
            if(synin==0)
            begin
                next_state=s0
                ;out=0;
            end
            else
            begin
                next_state=s1
                ;out=1;
            end
        end
        s1:
        begin

```

```

        if(synin==1)
        begin
            next_state=s1
            ;out=0;

        end

    endcase

end

end else begin
end
next_state=s0;out=0;
endmodule
//designing the synchronizer
module synchron(in, clk, rst,synin);

input in,clk,rst;
output synin;

wire out;

D_FF ff1(out,in,clk,rst);
D_FF ff2(synin,out,clk,rst);

endmodule
//designing the D flipflop
module D_FF (q, d, clock, reset);

    output q;
    input d, clock,

    reset;reg q;

    always @(posedge clock)
    begin
        if (~reset)
            q = 1'b0;
        els
        e      q = d;
    end

endmodule
//designing the clock divider
module CLOCK_Divider(input CLK_100MHz,input RESET,output reg CLK_1hz);

    integer c=0;
    always @(posedge CLK_100MHz)
    begin
        if(~RESET)
        begin

```

```

end
else
c = 0; CLK_1hz=1;
begin
CLK_1hz = ~CLK_1hz;
c=0;
end

```

