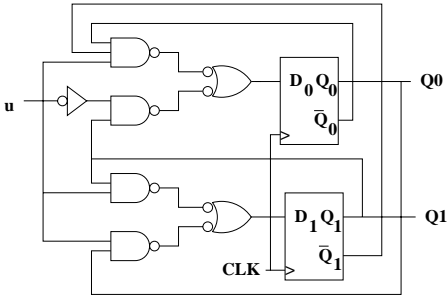
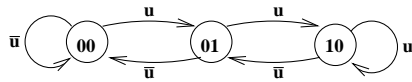


7.1 Simple FSM

Wednesday, February 21, 2001



Q1 Q0	00	01	11	10
u	0	0	X	0
1	0	1	X	1

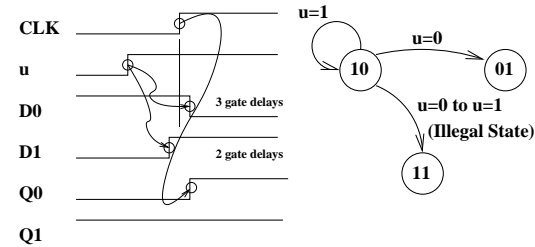
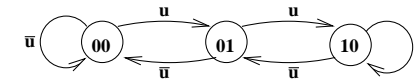
$$D1 = u \cdot Q0 + u \cdot Q1$$

Q1 Q0	00	01	11	10
u	0	0	X	1
1	1	0	X	0

$$D0 = u \cdot Q0 \cdot Q1 + \bar{u} \cdot Q1$$

7.2 Problem Transition

Consider the transition from state 10 (2) to state 01 (1),
if u changes from 0 to 1 close to the clock edge:



Most of the time, this circuit will work just fine. It screws up ONLY when the input transition is close to a clock edge.

Q1 Q0	00	01	11	10
u	0	0	X	0
1	0	1	X	1

$$D1 = u \cdot Q0 + u \cdot Q1$$

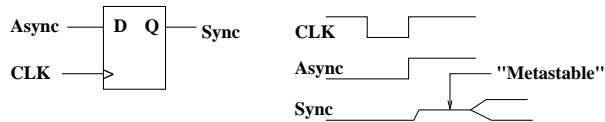
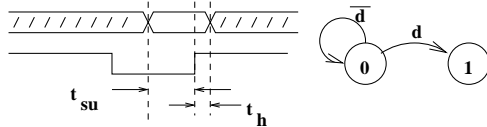
Q1 Q0	00	01	11	10
u	0	0	X	1
1	1	0	X	0

$$D0 = u \cdot Q0 \cdot Q1 + \bar{u} \cdot Q1$$

7.3 Important Design Rule

DESIGN RULE:

1. Synchronize ALL external signals.
2. Any asynchronous input must affect ONLY ONE flip-flop.



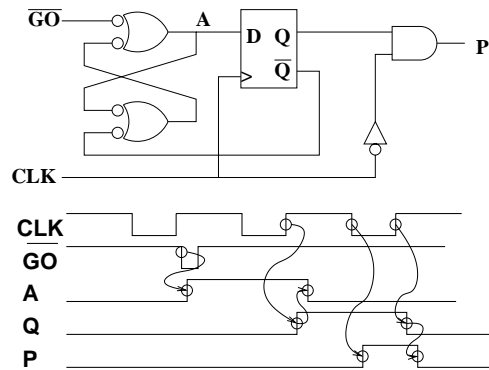
Any combinational logic with "Sync" as an input will be "glitchy" until after the metastable state has expired.
In particular, do NOT use "Sync" as a CLK input.

7.4 VHDL Code for Short Pulse Catcher

```

library ieee;
use ieee.std_logic_1164.all;
entity spulse is
    port(n_go, clk : in std_logic;
         p : out std_logic);
end spulse;
-- purpose: catch a short pulse
architecture behavioral of spulse is
    signal a, n_a, x, n_x, n_clk: std_logic;
    attribute synthesis_off of a: signal is true;
    attribute synthesis_off of n_a: signal is true;
begin -- behavioral
    a <= (not n_go) or (not n_a);
    n_a <= (not a) or (not n_x);
    n_x <= (not x);
    p <= x and (not clk);
ff: process(clk)
begin
    if rising_edge(clk) then
        x <= a;
    end if;
end process ff;
end behavioral;
    
```

7.5 Catch an Edge or a Short Pulse



7.6 Simulation of Catching a Short Pulse

```
begin -- behavioral
  a <= (not n_go) or (not n_a);
  n_a <= (not a) or (not n_x);
  n_x <= (not x);
  p <= x and (not clk);
ff: process(clk)
begin
  if rising_edge(clk) then
    x <= a;
  end if;
end process ff;
end behavioral;
```



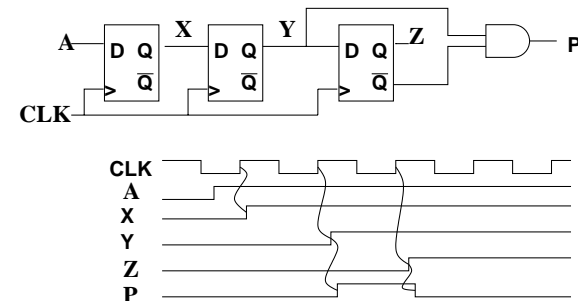
7.7 VHDL Code for a Pulse Shaper

```
library ieee;
use ieee.std_logic_1164.all;
entity pform is
  port(A, CLK: in std_logic;
        P: out std_logic);
end pform;

-- purpose: Turn a level into a short pulse
architecture behavioral of pform is
  signal X, Y, Z: std_logic;
begin -- behavioral
ff: process(CLK)
begin
  if rising_edge(CLK) then
    X <= A;
    Y <= X;
    Z <= Y
  end if;
end process ff;
  P <= Y AND (not Z);
end behavioral;
```

7.8 Turn a Level Into a Pulse

Or turn a level into a finite-width pulse:



7.9 Simulation of Level to Pulse

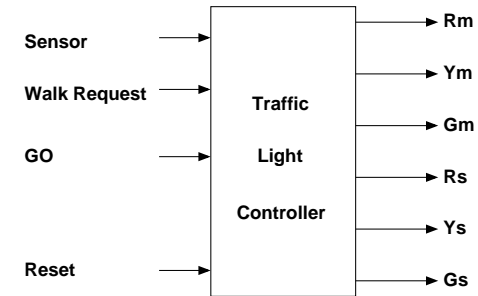
```

begin -- behavioral
ff: process(CLK)
begin
if rising_edge(CLK) then
X <= A;
Y <= X;
Z <= Y
end if;
end process ff;
P <= Y AND (not Z);
end behavioral;

```

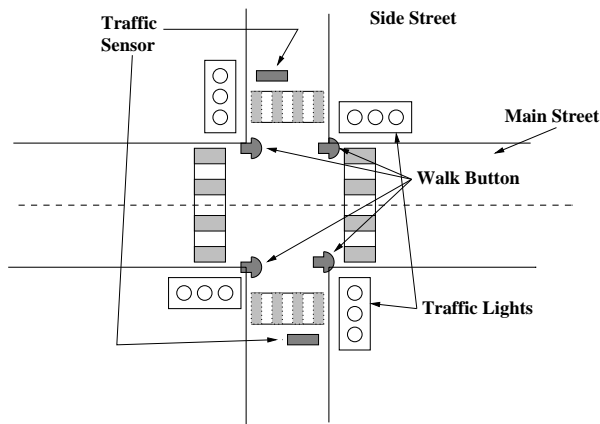


7.10 Top Level Block Diagram



7.11 Lab 2 Assignment

FSM to Control Traffic Signal



7.12 Detailed Block Diagram

