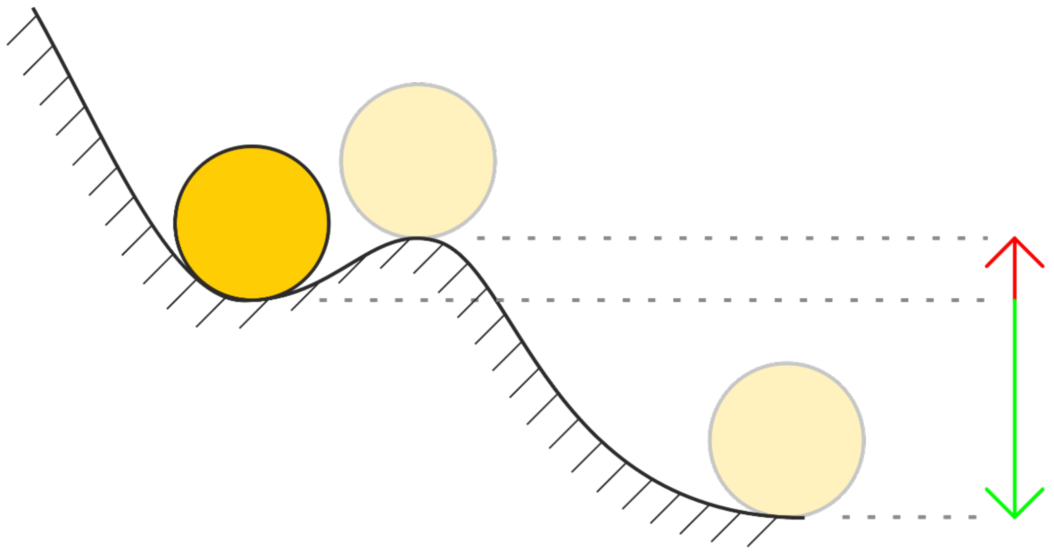
Metastability and Synchronizing Circuits



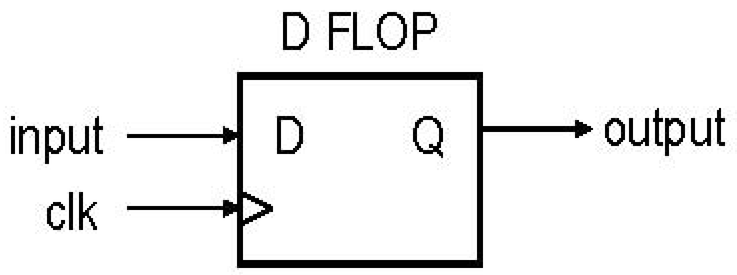
Announcements

- Homework #12 due today
- Final Exam: Monday December 16
  - 4:15 – 6:45 PM
  - Please schedule DSO
  - From RIT D11.0 Final Examination Policies

  "Students shall not be required to take more than two examinations or more than twelve (12) hours of exams on a single day. In such an instance, the student may request a scheduling change"
  - Let me know if you need to reschedule

Flip Flop Operation

- Consider a D Flip-flop
  - Its output Q is either a 1 or a 0
  - Q gets the value of the input D on the active edge of the clock



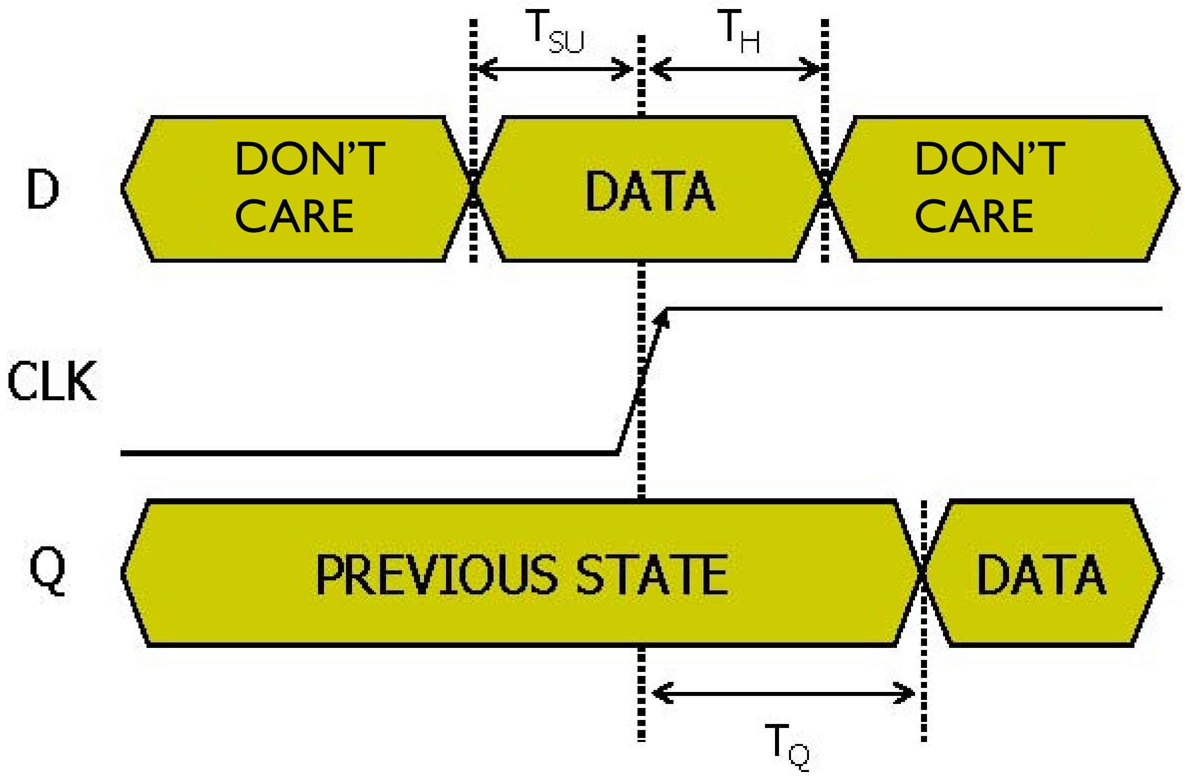
  - However, input data must be stable a short time prior to and a short time after the active clock edge
    - Times are known as setup time t_{su} and hold time t_h

Setup and Hold Times

- Setup Time (t_{su})
  - The time interval immediately preceding the active transition of the CLK signal during which the D and enable inputs must be maintained at the proper level.
- Hold Time (t_H)
  - The time interval immediately following the active transition of the CLK signal during which the D and enable inputs must be maintained at the proper level
- If the input changes within the setup or hold time window, the Q output is not guaranteed to be predictable.

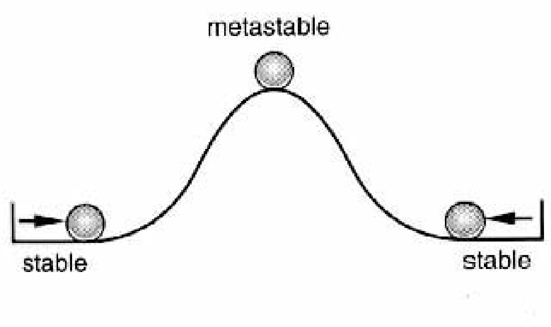
Setup and Hold Times

- Predictable behavior of a flip flop is guaranteed if input data is stable for setup and hold time



What is a metastable state?

- If data changes within setup and hold time window, the output of the flip flop cannot be predicted
  - It may go to 1 or 0
  - Or it may go **metastable**
- The metastable state is halfway between 1 and 0
  - Considered unstable equilibrium
  - Like a ball on a hill



What is a metastable state?

- It will eventually settle to a stable state of 0 or 1
  - How long will it take?
  - What state will it settle to?
  - The probability of remaining in a metastable state decreases exponentially with time.
- There is no cure for metastability
  - You can't prevent it
  - But….You can reduce the chances of it happening

Inputs to Synchronous systems

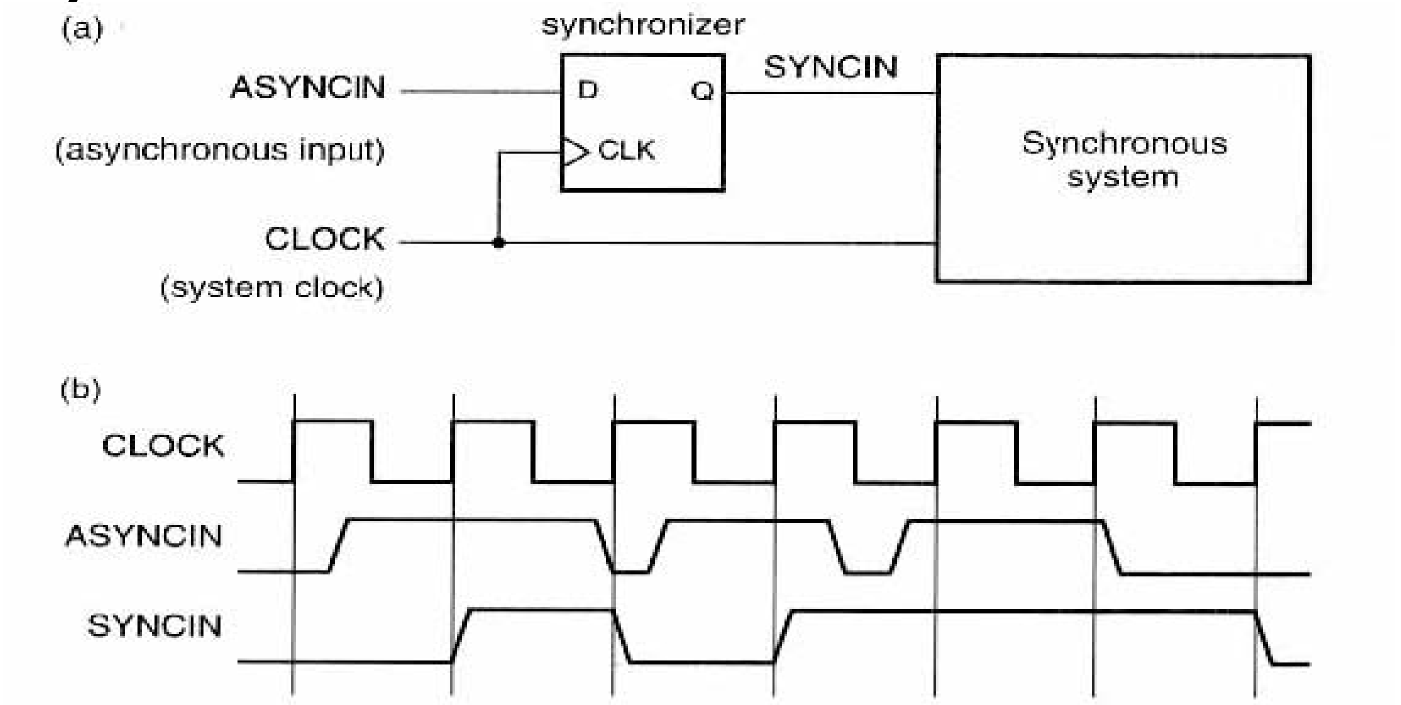
- Many synchronous systems need to interface to asynchronous input signals:
  - Consider a computer system running at some clock frequency, say 1GHz with:
    - Interrupts from I/O devices, keystrokes, etc.
  - Data transfers from devices with their own clocks
    - Ethernet has its own 100MHz clock
    - PCI bus transfers, 66MHz standard clock.
  - These signals could have no known timing relationship with the system clock of the CPU.

Inputs to synchronous systems

- If the input is not synchronous with the system's clock the potential exists to violate setup or hold times
  - May result in a metastable state
  - System may not work as expected
- If we can't prevent metastability how do we reduce the chance of it occurring?

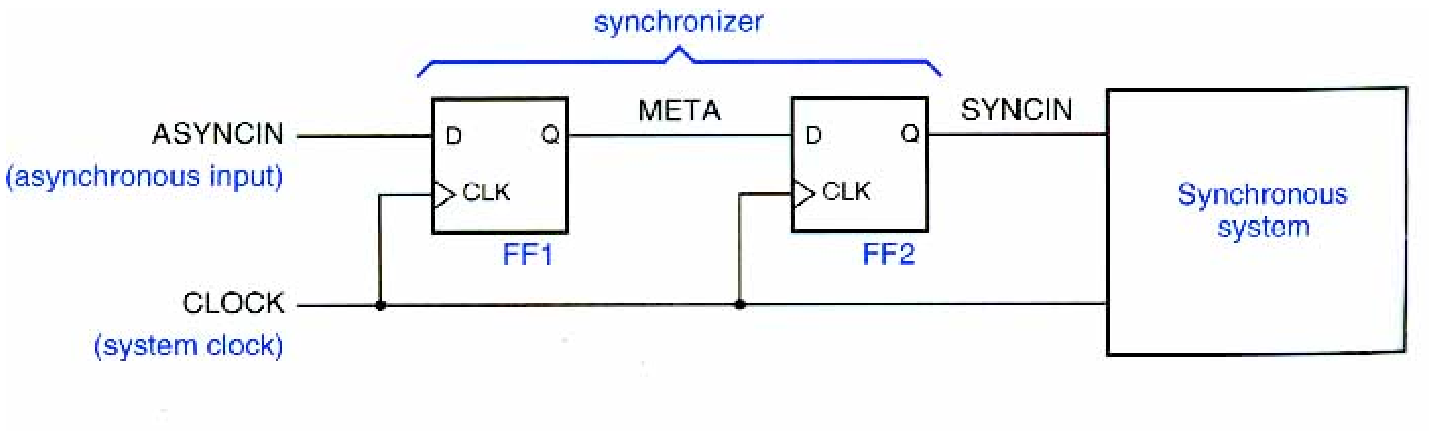
Synchronizing Circuit

- For a single asynchronous input, use a simple flip-flop to bring the external input signal into the timing domain of the system clock:



Reliable Synchronizer Design

- The probability that a flip-flop stays in the metastable state decreases exponentially with time
  - Therefore, any scheme that delays using the signal can be used to decrease the probability of failure
  - Recommended design



Clock Domains

- Why would you have multiple clock domains?
  - Independent (sub)systems with different reference clocks, needing to share/exchange information.
  - Impractical to distribute or use a reference clock.

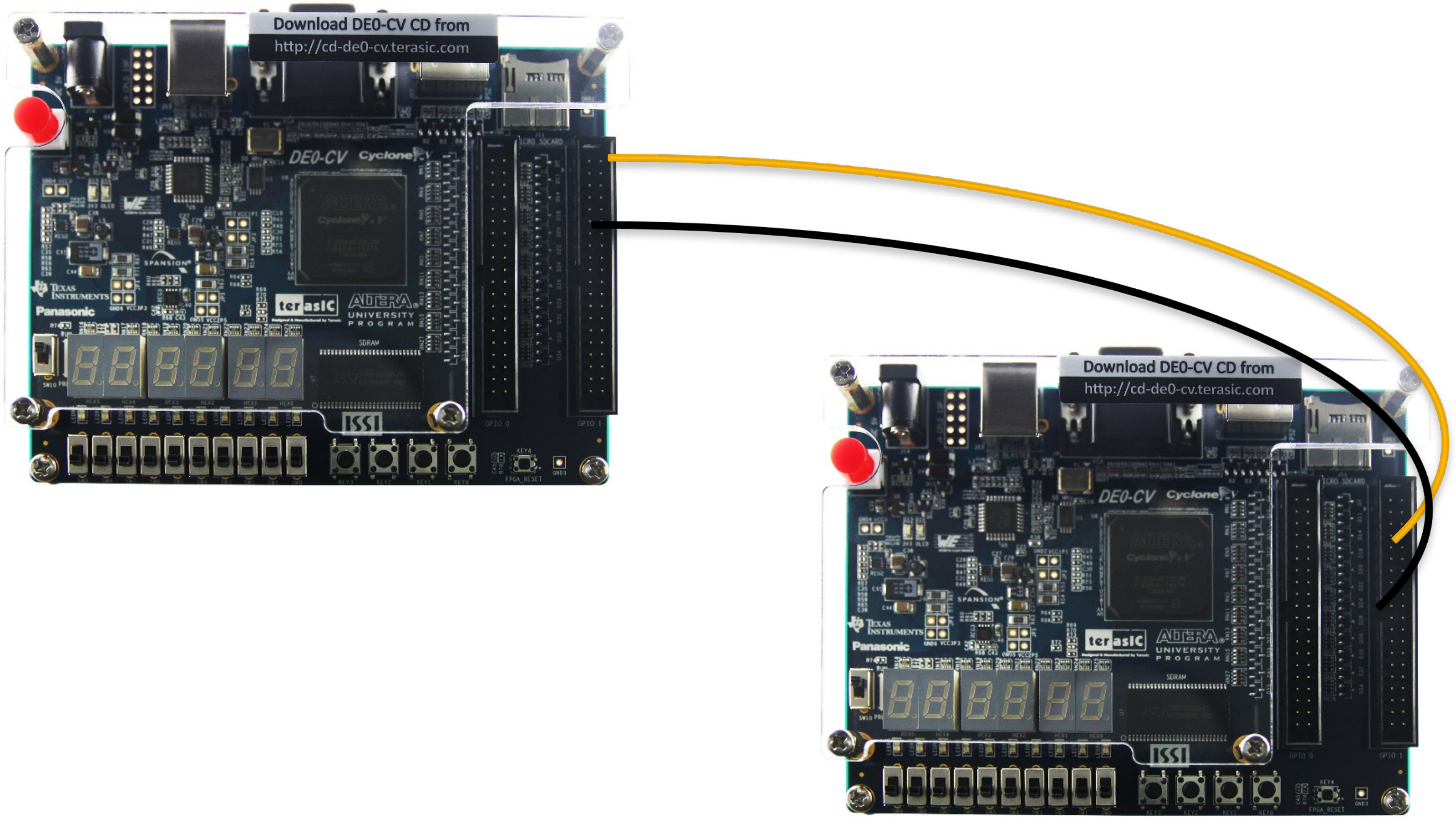
Crossing Clock Domains

- For asynchronous clock domain relationships:
  - For **a single signal**, use the same three flip-flop synchronizer used for asynchronous inputs
  - For a parallel data transfer (more than 1 bit at a time) a FIFO is used.
    - FIFO – first-in, first-out synchronous data buffer
    - Will be covered in HDL/ESD I

Transmitting / Receiving Synchronization

- Still need to synchronize even if both transmitter and receiver have same frequency clock
  - Even though they are both on the same clock frequency, there is no guarantee that they are in sync
  - The incoming data must be synchronized to the receiver's clock.
  - This is done with synchronizing flip-flops

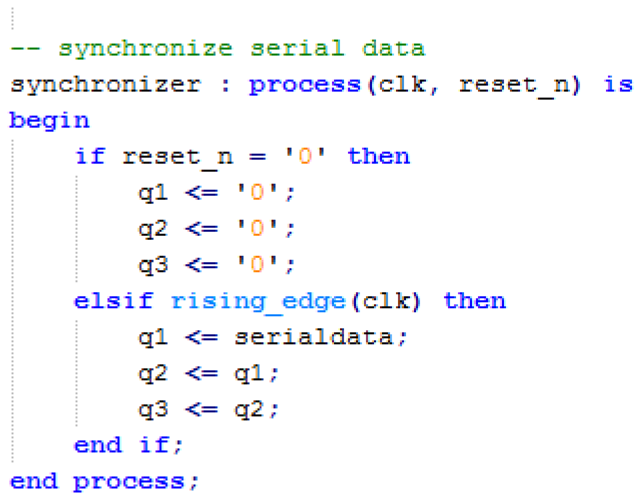
Transmitting/Receiving Synchronization



Transmitter

Receiver

Transmitting / Receiving Synchronization

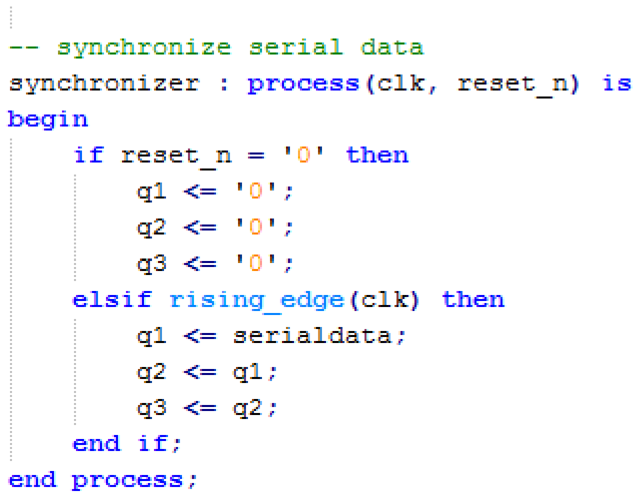


```vhdl
-- synchronize serial data
synchronizer : process(clk, reset_n) is
begin
    if reset_n = '0' then
        q1 <= '0';
        q2 <= '0';
        q3 <= '0';
    elsif rising_edge(clk) then
        q1 <= serialdata;
        q2 <= q1;
        q3 <= q2;
    end if;
end process;
```

- What does this look like?

Edge Detection

- Synchronizing circuit can also be used to detect a rising or falling edge in the incoming data
  - Useful because some transmission protocols “wake up” the receiver with a falling edge



```
-- synchronize serial data
synchronizer : process(clk, reset_n) is
begin
    if reset_n = '0' then
        q1 <= '0';
        q2 <= '0';
        q3 <= '0';
    elsif rising_edge(clk) then
        q1 <= serialdata;
        q2 <= q1;
        q3 <= q2;
    end if;
end process;
```

How can we use this circuit to detect a falling edge on the input data?

Digital Filtering

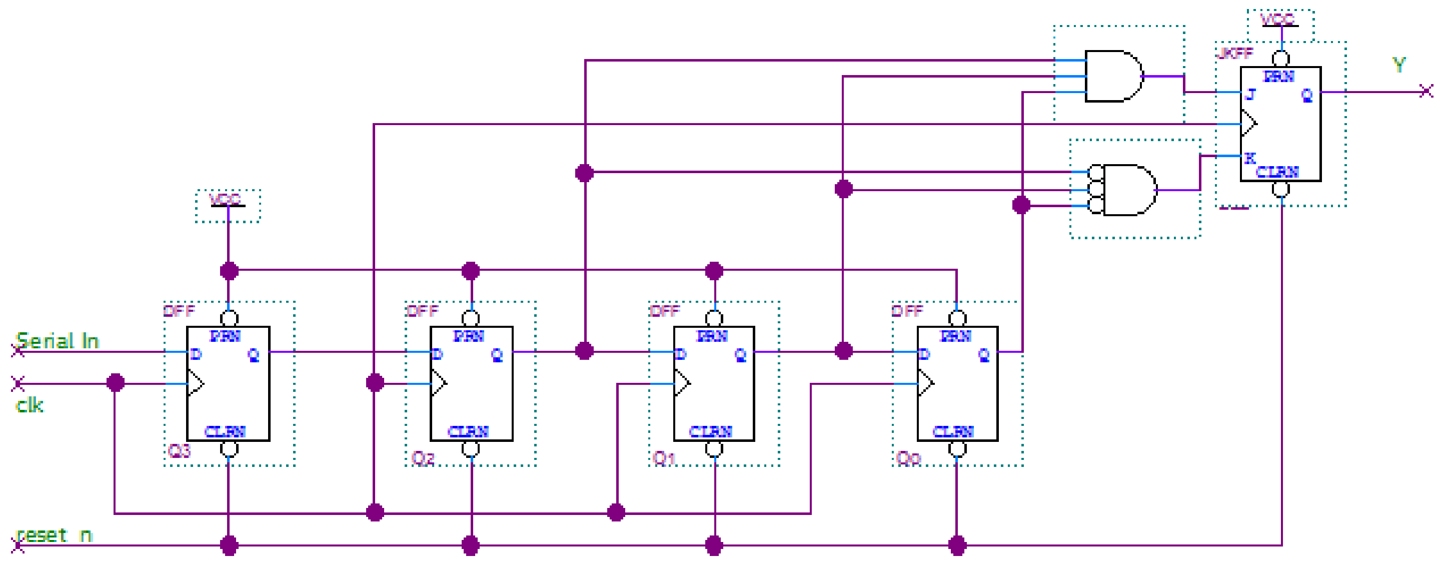
- Digital systems can be noisy
- Random spikes on incoming data lines need to be filtered out
- This can be achieved with modifications to the synchronization circuit

Digital Filtering

```
entity filter is
   port(serial_in, clk, reset_n : in std_logic;
        y : out std_logic);
end filter;

architecture behave of filter is
begin
   synch: process (clk)
      variable q : std_logic_vector(3 downto 0);
   begin
      if (rising_edge(clk)) then
         if reset_n = '0' then
            q := "0000";
            y <= '0';
         else
            q := serial_in & q(3 downto 1) ;  --right shift
--verify the incoming data has been the same state for 3 clks
            if q(2 downto 0) = "111" then
               y <= '1';
            elsif q(2 downto 0) = "000" then
               y <= '0';
            end if;
         end if;
      end if;
   end process;
end behave;
```

Digital Filtering


VCC
VCC
JKFF
PRN
J
Q
K
CLRN
Y
DFF
PRN
D
Q
CLRN
Q3
Q2
Q1
Q0
Serial In
clk
reset_n