Sequencing

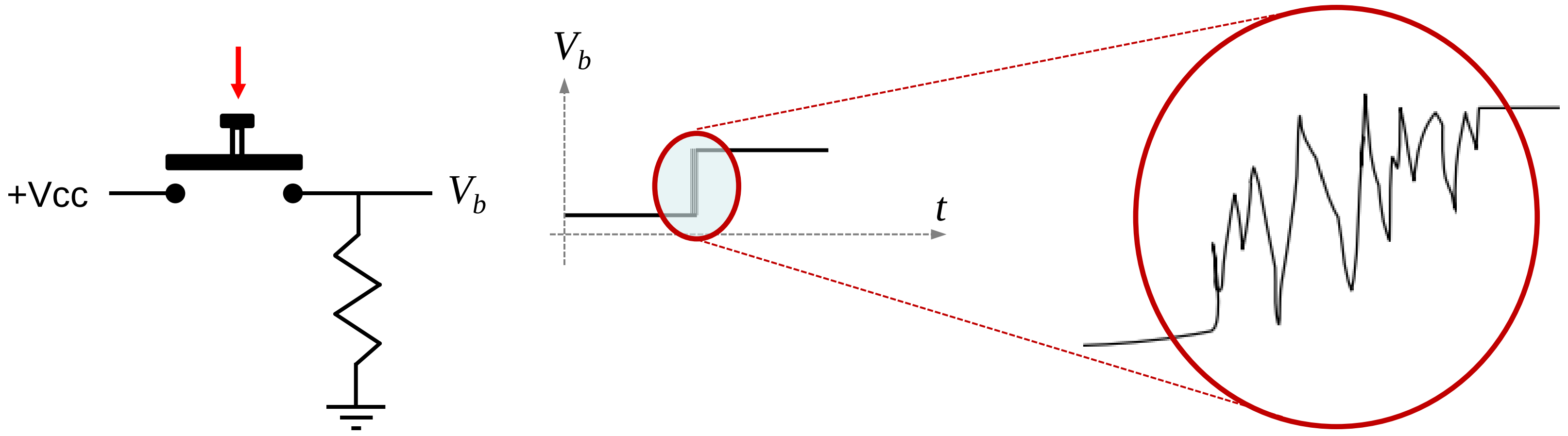
by Erol Seke

For the course "`Introduction to VHDL`"

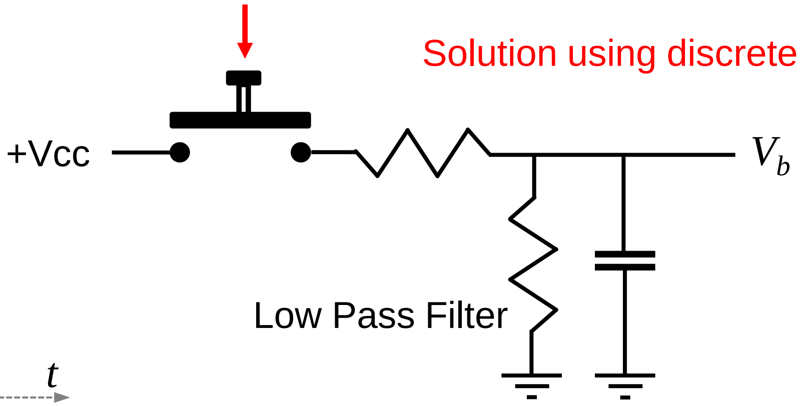
ESKİŞEHİR OSMANGAZI UNIVERSITY

Key Bounce

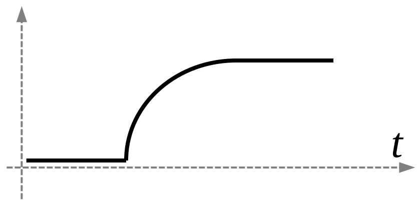
Making or opening circuits in mechanical switches take a finite amount of time



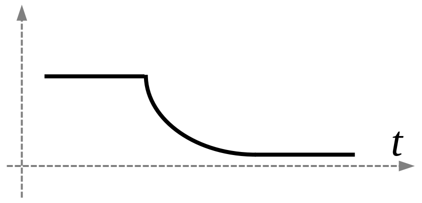
Solution using discrete passive elements



Low Pass Filter



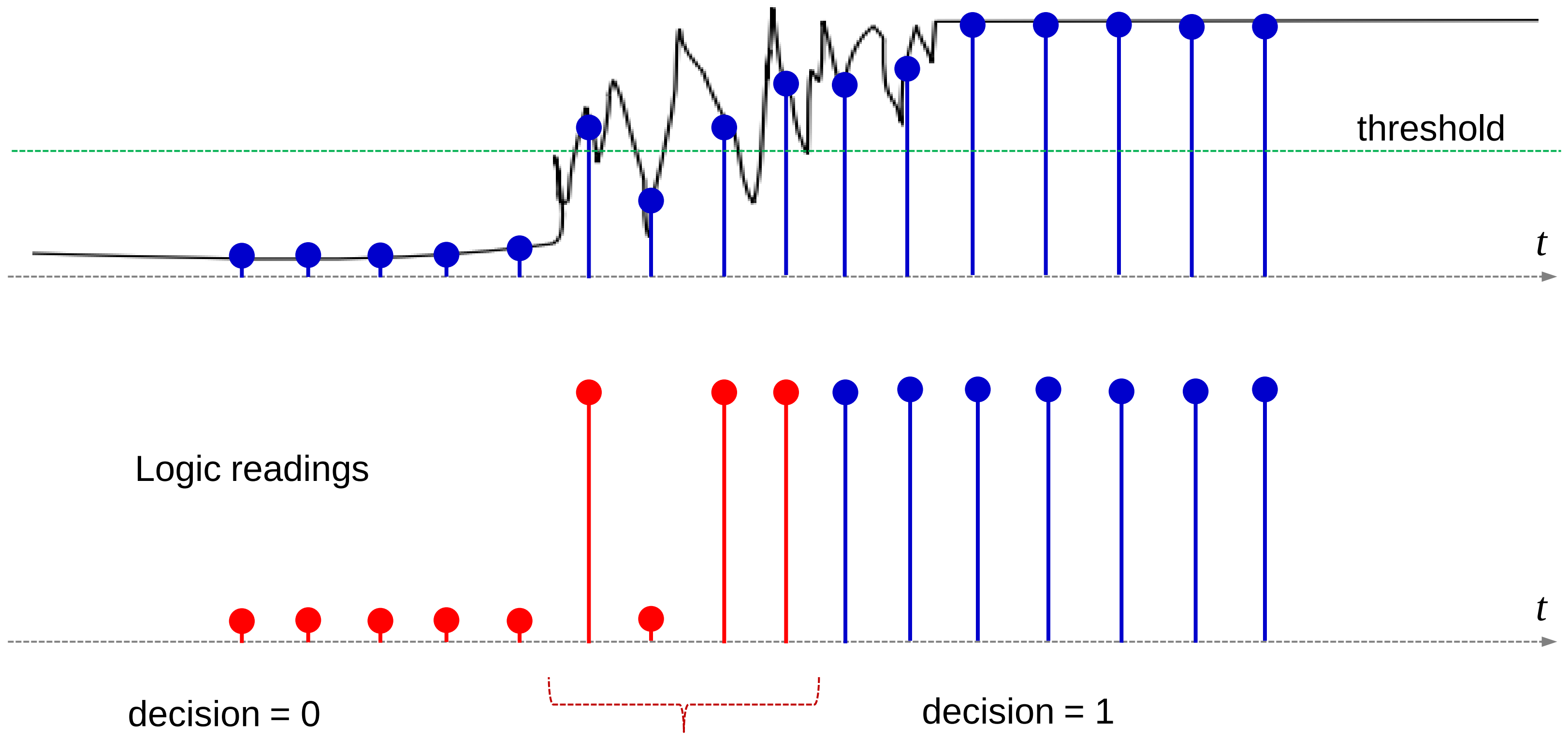
Connect



Disconnect

Simple Digital Solution

Measure the logic input 3 times with approximately 5 ms apart.
If all 3 are the same then set the output (decision) to that value.



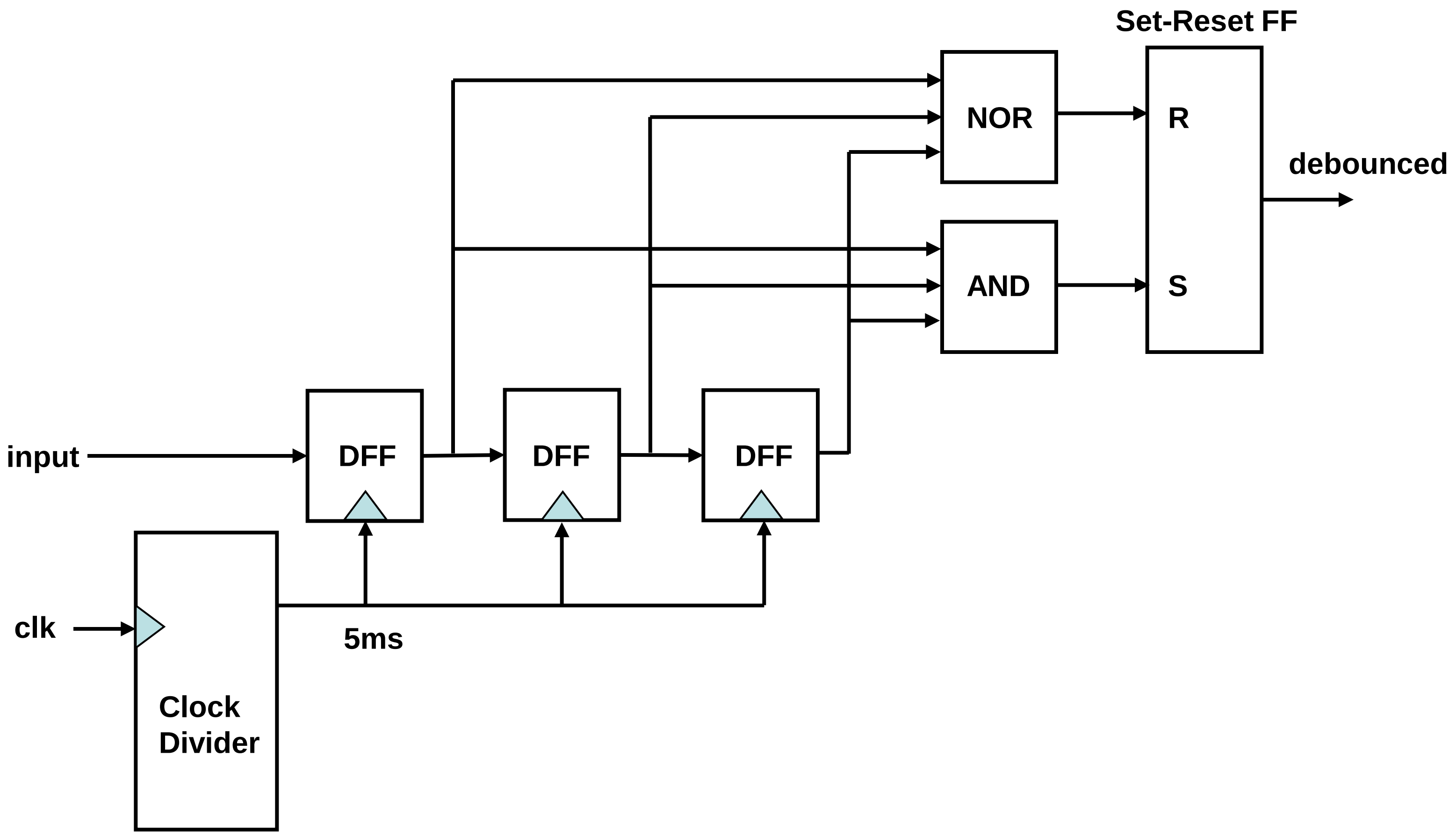
decision = 0

decision = 1

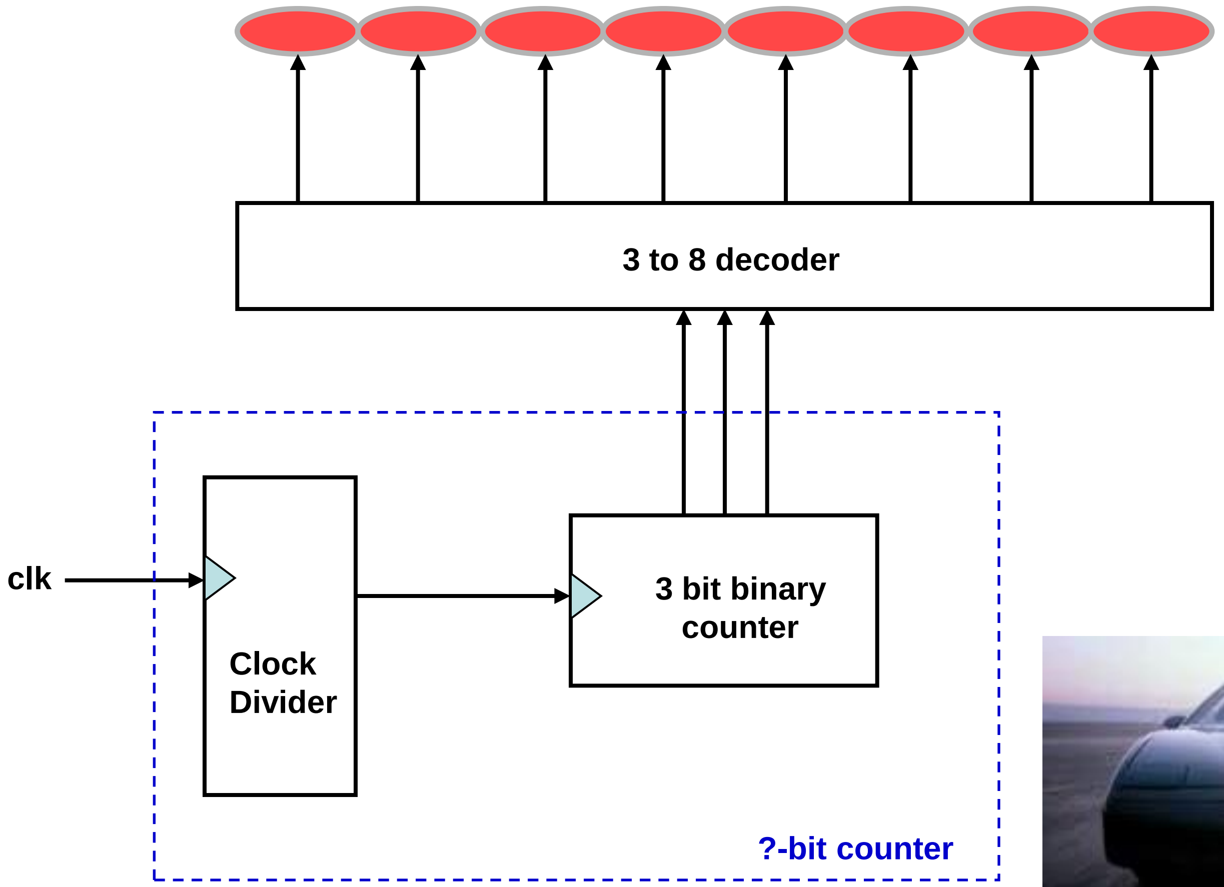
no change at the decision

Note : 5 ms is not decisive. It really depends on the mechanical characteristics of the contacts.
It can be from 2 ms upto 50 ms.

Debouncer Circuit



Sliding LEDs



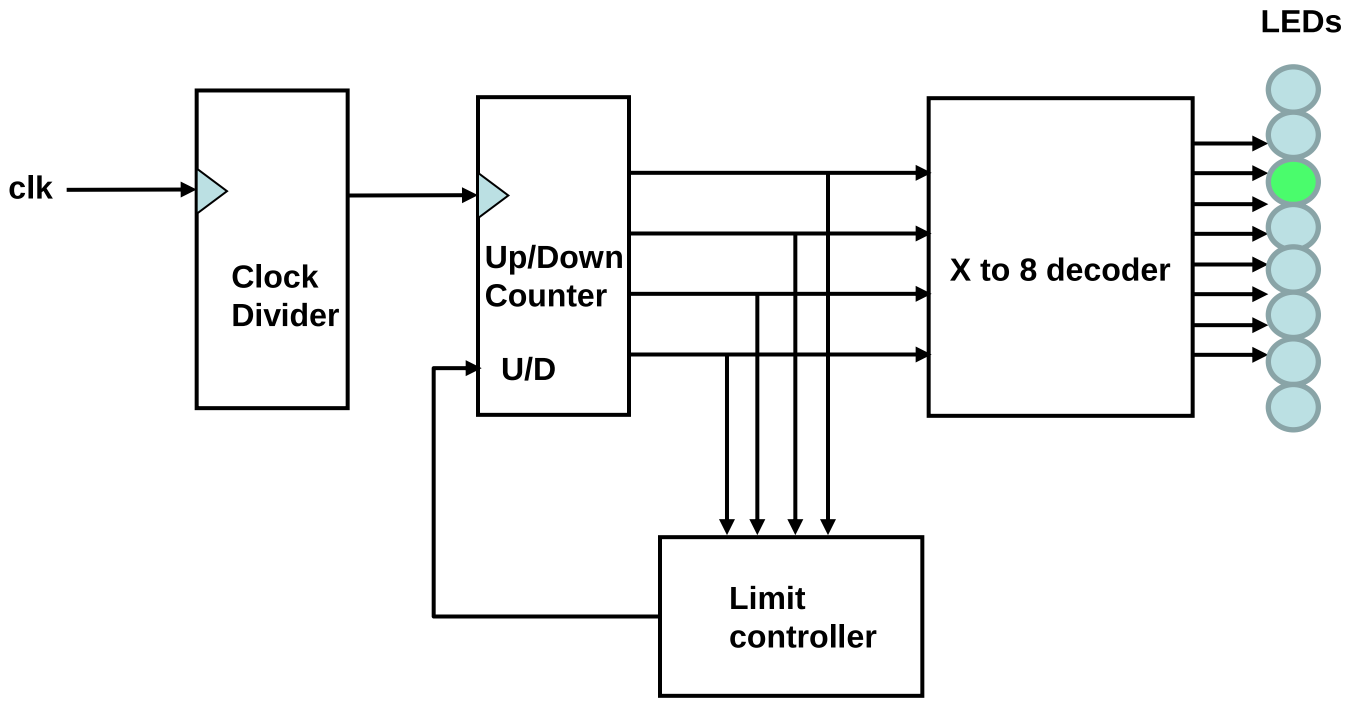
How about *knight rider*?

Knight-Rider with shift operators

```vhdl
architecture KnightRider of KnightRider is
  signal direction: BIT := '0'; -- init is for simulation only
  signal cntr: STD_LOGIC_VECTOR(21 downto 0);
begin
  CLKDIV: process(clk) is begin   -- clock divider 50MHz/2^22
    if(RISING_EDGE(clk)) then
      cntr <= cntr +1;
    end if;
  end process; -- can have as many processes as wished

  RIDER: process(cntr(21)) is begin
    if(cntr(21)'EVENT and cntr(21)='1') then -- same as RISING_EDGE
      if(Rst='1') then                       -- synchronous reset
        LEDS <= (LEDS'LOW=>'1',others=>'0');
      elsif(LEDS(LEDS'LOW)='1') then
        direction <= '1';
        LEDS <= (LEDS'LOW => '0', LEDS'LOW +1 => '1', others=>'0');
      elsif(LEDS(LEDS'HIGH)='1') then
        direction <= '0';
        LEDS <= (LEDS'HIGH => '0', LEDS'HIGH -1 => '1', others=>'0');
      elsif(direction='1') then
        LEDS <= LEDS sll 1;  -- sll & srl work for BIT_VECTOR type
      else
        LEDS <= LEDS srl 1;
      end if;
    end if;
  end process;
end KnightRider;
```

```vhdl
entity KnightRider is Port (
  clk : in  STD_LOGIC;
  Rst : in  STD_LOGIC;
  LEDS : inout  BIT_VECTOR (7 downto 0));
end KnightRider;
```

Knight-Rider (another design)



IEEE Standard Library Packages

numeric_bit	: defines numeric types and arithmetic functions for unsigned & signed bit
numeric_std	: defines numeric types and arithmetic functions for unsigned & signed std_logic
std_logic_1164	: some conversions and overloaded logic operators. rising_edge etc.
std_logic_arith	: a set of arithmetic, conversion, and comparison functions
std_logic_misc	: supplemental types, subtypes constants, and functions for the Std_logic_1164
std_logic_signed	: a set of arith., conversion, and comparison functions for STD_LOGIC_VECTOR
std_logic_unsigned	: a set of unsigned arith., conv., and comp. functions for STD_LOGIC_VECTOR
std_logic_textio	:
math_complex	:
math_real	:

```
FUNCTION rising_edge (SIGNAL s : std_ulogic) RETURN BOOLEAN IS
BEGIN
  RETURN (s'EVENT AND (To_X01(s) = '1') AND
                (To_X01(s'LAST_VALUE) = '0'));
END;
```

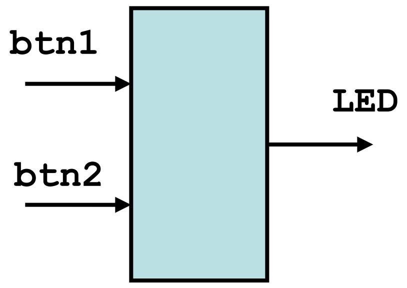
```
function max(L, R: INTEGER) return INTEGER is
begin
  if L > R then
    return L;
  else
    return R;
  end if;
end;
```

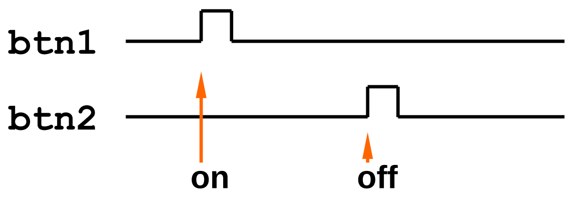
```
function CONV_INTEGER(ARG: STD_ULOGIC)
return SMALL_INT is
  variable tmp: STD_ULOGIC;
begin
  tmp := tbl_BINARY(ARG);
  if tmp = '1' then
    return 1;
  elsif tmp = 'X' then
    assert false
    report "<message truncated here>"
    severity WARNING;
    return 0;
  else
    return 0;
  end if;
end;
```

Three example functions from std_logic_arith and std_logic_1164

Signals with Multiple Drivers

Design : Inputs `btn1` and `btn2` are connected to S1 and S2 buttons respectively. Buttons are normally open and inputs are pulled-down. Other ends of the buttons are connected to Vcc. S1 button turns a `LED` on whilst S2 turns it off.





Incorrect Design

```
architecture MultDriven of MultDriven is
begin
  P1: process(btn1) is begin
    if(RISING_EDGE(btn1)) then
      LED <= '1';
    end if;
  end process;

  P2: process(btn2) is begin
    if(RISING_EDGE(btn2)) then
      LED <= '0';
    end if;
  end process;
end MultDriven;
```

When you try to synthesize your code, you will get
```
ERROR:Xst:528 - Multi-source in Unit <MultDriven> on signal <LED>
```
or something similar.

Bad Synchronous Description

```
architecture MultDriven of MultDriven is
begin
  process(btn1, btn2) is begin
    if(RISING_EDGE(btn1)) then
      LED <= '1';
    elsif(RISING_EDGE(btn2)) then
      LED <= '0';
    end if;
  end process;
end MultDriven;
```

Try to think of a circuit that is described by the code above.
Is it possible?

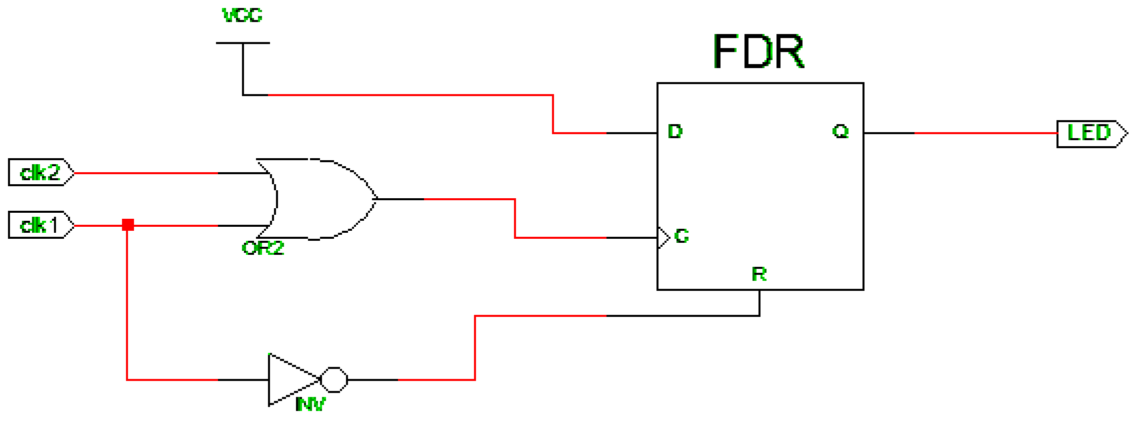
Try to think of a circuit that does what is needed.

A Working Design

```
architecture SinglyDriven of SinglyDriven is
  signal clk: STD_LOGIC;
begin
  clk <= btn1 or btn2;
  process(clk) is begin
    if(RISING_EDGE(clk)) then
      if(btn1='1') then
        LED <= '1';
      else
        LED <= '0';
      end if;
    end if;
  end process;
end SinglyDriven;
```

some combinatorial
some sequential

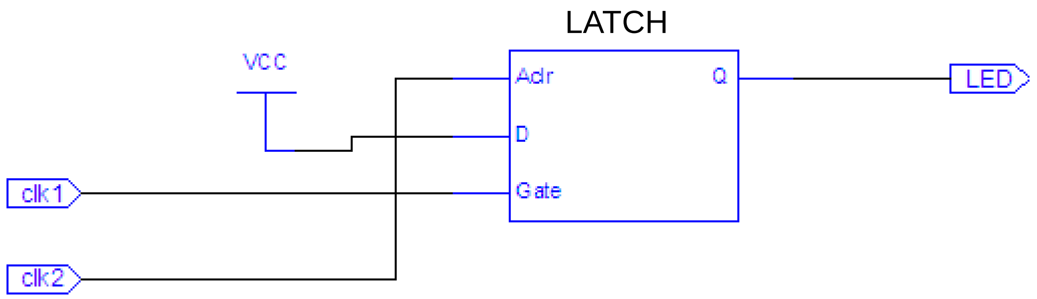
Obviously, when both inputs goes up simultaneously, code lets `btn1` to win



A Simpler Design

```
architecture SRLatch of SRLatch is
begin
  process(btn1, btn2) is
  begin
    if(btn2='1') then
      LED <= '0';
    elsif(btn1='1') then
      LED <= '1';
    end if;
  end process;
end SRLatch;
```

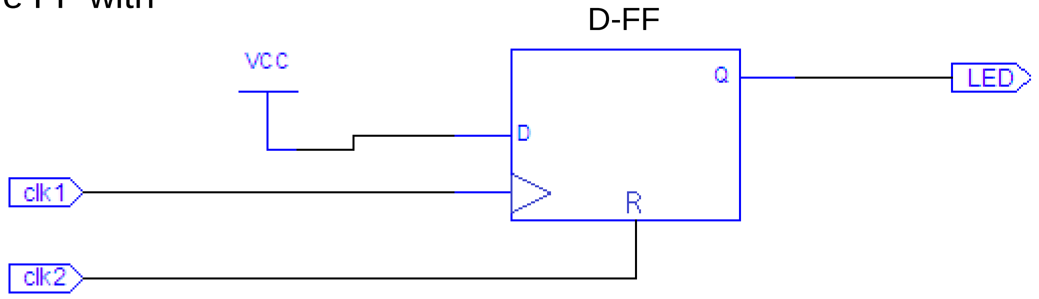
Synthesizer may complain about inference of a latch. If this is what we want, so we shall ignore the warning here.



Another Simple Design

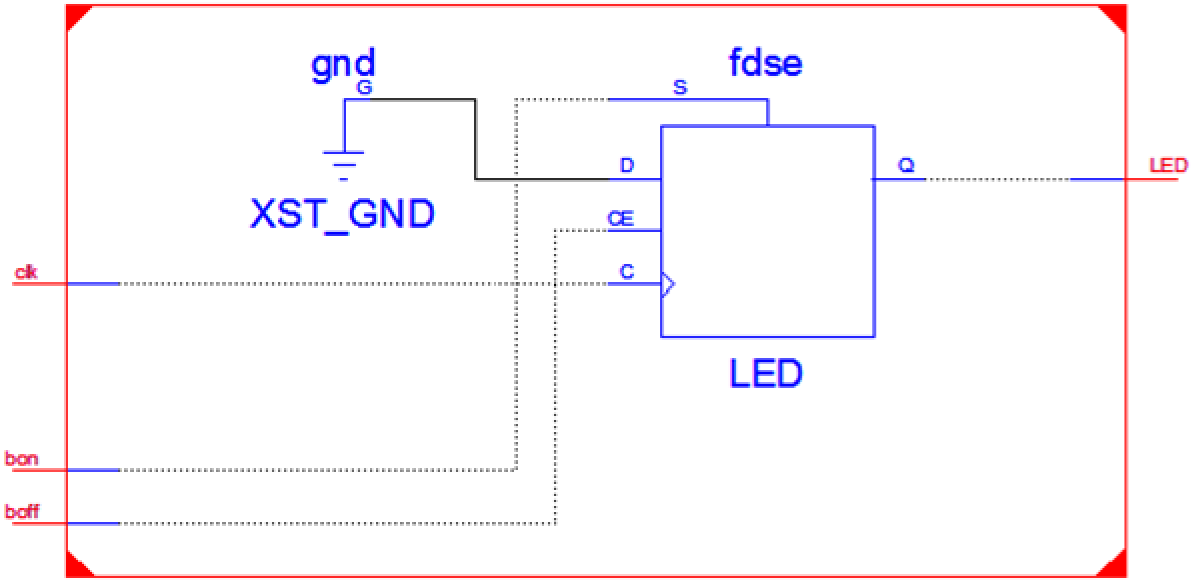
```
architecture SRasync of SRasync is
begin
  process(btn1, btn2) is
  begin
    if(btn2='1') then
      LED <= '0';
    elsif(RISING_EDGE(btn1)) then
      LED <= '1';
    end if;
  end process;
end SRasync;
```

This line is changed.
The circuit will look like a simple FF with asynchronous reset



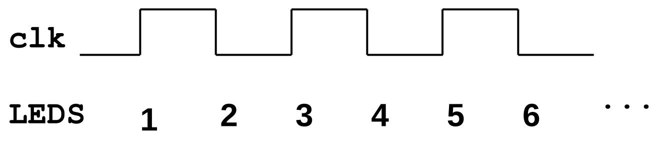
A Synchronous Design

```
architecture SRSync of SRSync is
begin
  process(clk, bon, boff) is
  begin
    if(RISING_EDGE(clk)) then
      if(bon='1') then
        LED <= '1';
      elsif(boff='1') then
        LED <= '0';
      end if;
    end if;
  end process;
end SRSync;
```



Double Rate Counter

Design : Output LEDS (8 bit) counts up one on both rising and falling edge of the single input `clk`.



Incorrect Design

```
architecture DoubleRate of DoubleRate is
begin
  process(clk) is begin
    if(RISING_EDGE(clk)) then
      cntr <= cntr + 1;
    end if;
  end process;

  process(clk) is begin
    if(FALLING_EDGE(clk)) then
      cntr <= cntr + 1;
    end if;
  end process;
```

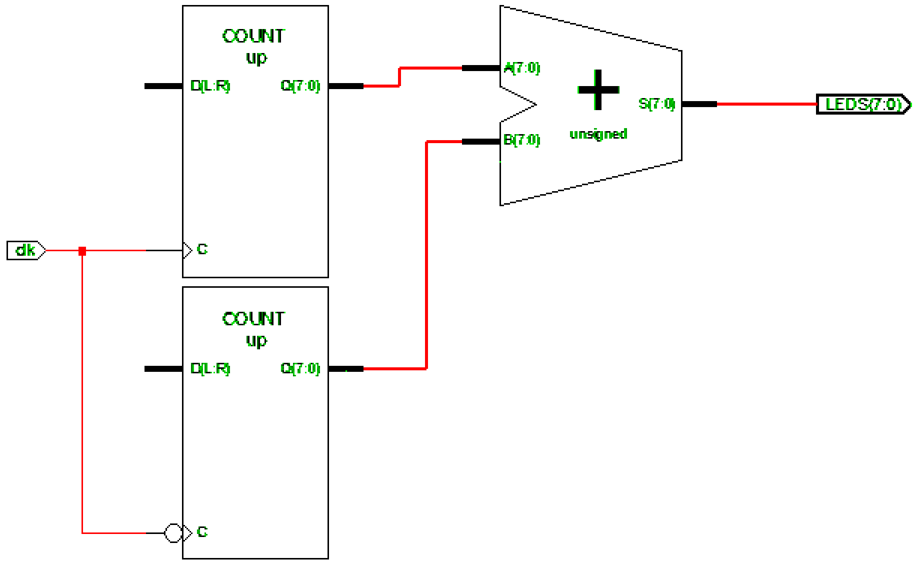
When you try to synthesize the code, you will get

```
ERROR:Xst:528 - Multi-source in Unit <DoubleRate> on signal <LEDS<7>>
...
```

or similar.

Double Rate Counter - Working Design

```
architecture DoubleRate of DoubleRate is
  signal cntr1: STD_LOGIC_VECTOR(7 downto 0);
  signal cntr2: STD_LOGIC_VECTOR(7 downto 0);
begin
  LEDS <= cntr1 + cntr2; -- + is a valid operator
                         -- for std_logic_vector

  process(clk) is begin
    if(RISING_EDGE(clk)) then
      cntr1 <= cntr1 + 1;
    end if;
  end process;

  process(clk) is begin
    if(FALLING_EDGE(clk)) then
      cntr2 <= cntr2 + 1;
    end if;
  end process;
```



Double Rate Counter – Synchronous w/ High Clock

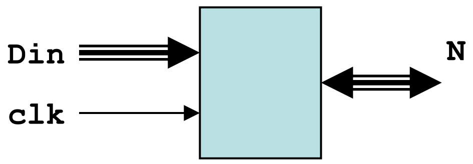
```
architecture DoubleRate of DoubleRate is
  signal Reg: STD_LOGIC;
  signal cntr: STD_LOGIC_VECTOR(7 downto 0);
begin

  process(clkH) is begin
    if(RISING_EDGE(clkH)) then
      if(Reg~=clk) then
        cntr <= cntr + 1;
      end if;
      Reg <= clk;
    end if;
  end process;
```

Assumption : rate of the `clkH` is at least twice higher that the rate of the `clk`

Homework : Draw the logic circuit you expect before typing in the code.

Variables

Design : Calculate the number of ones in an 8 bit signal fed with a clock

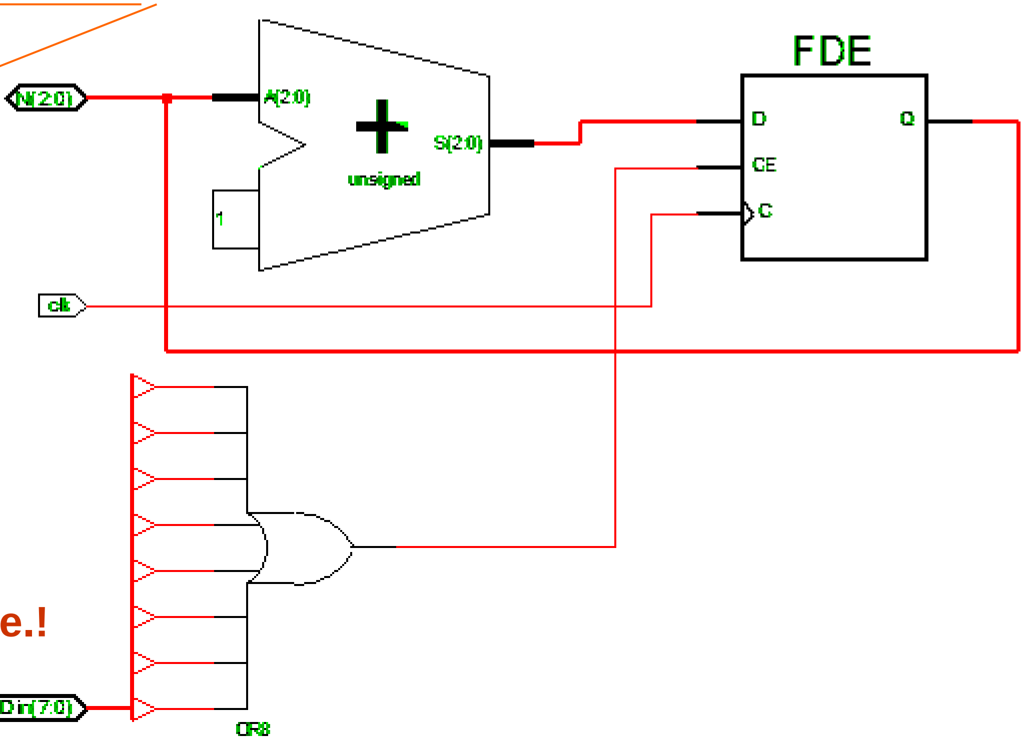


Incorrect

```
architecture Count1s of Count1s is
begin
  process(clk) is begin
    if(rising_edge(clk)) then
      --N <= "000";
      for i in 0 to 7 loop
        if(Din(i)='1') then
          N <= N + 1;
        end if;
      end loop;
    end if;
  end process;
end Count1s;
```

Putting this will generate "multiple driver" error.

new keywords



Incorrect circuit.!
But no warning on synthesize.!

Variables

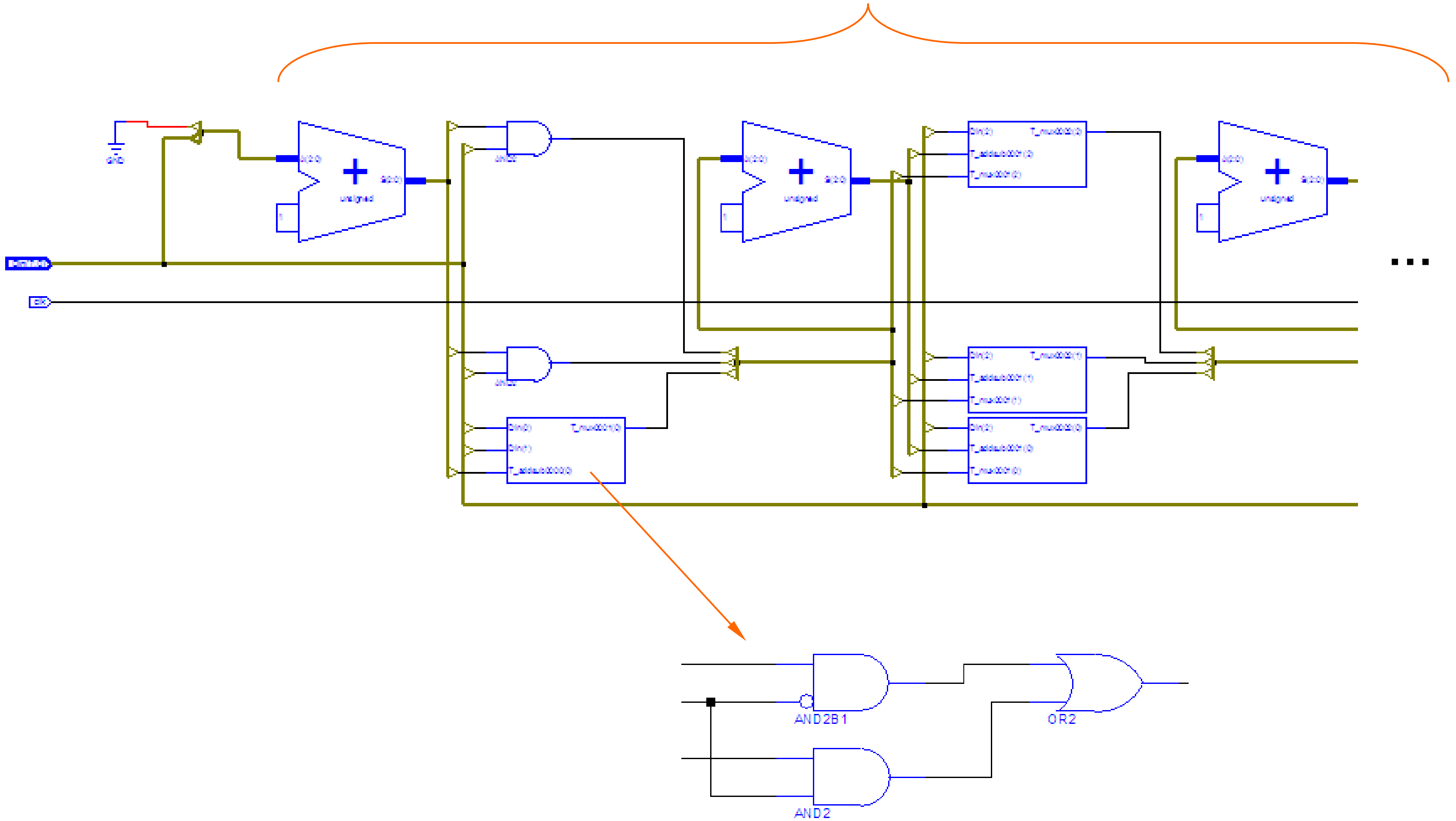
Correct

```
architecture Count1s of Count1s is
begin
  process(clk) is
    variable T: STD_LOGIC_VECTOR(2 downto 0); -- declared here
  begin
    if(rising_edge(clk)) then
      T := "000";      -- := is used for assignment to variables
      for i in 0 to 7 loop
        if(Din(i)='1') then
          T := T + 1;
        end if;
      end loop;
      N <= T;    -- assignment to variables is immediate
    end if;      -- no need to wait for the next clock pulse
  end process;
end Count1s;
```

Homework : Design a parity bit calculator. 7 bit input and 8 bit output.

Resulting circuit

Total of 7 adder and mux stages



HW: Could it be simpler?

Homework : Read sections 5, 6, 7
Do problems 5.2, 5.6, 6.1, 6.4, 6.8, 7.2, 7.5.
Design an Up-Down counter with Up, Down, clk and Rst inputs.
It shall not count when Up=Down='0' or Rst='1'.

END