Registers

Last updated 5/14/20

Registers

These slides review the design for several types of registers

Upon completion: You should be able to design generic sized registers with various control capabilities

(synchronous and asynchronous set/clr/rst/…)

Registers

- Registers are created by creating code that conforms to the template

```
process (clock signal)
  begin
    if(clock edge detection) then
      actions
    end if;
  end process;
```

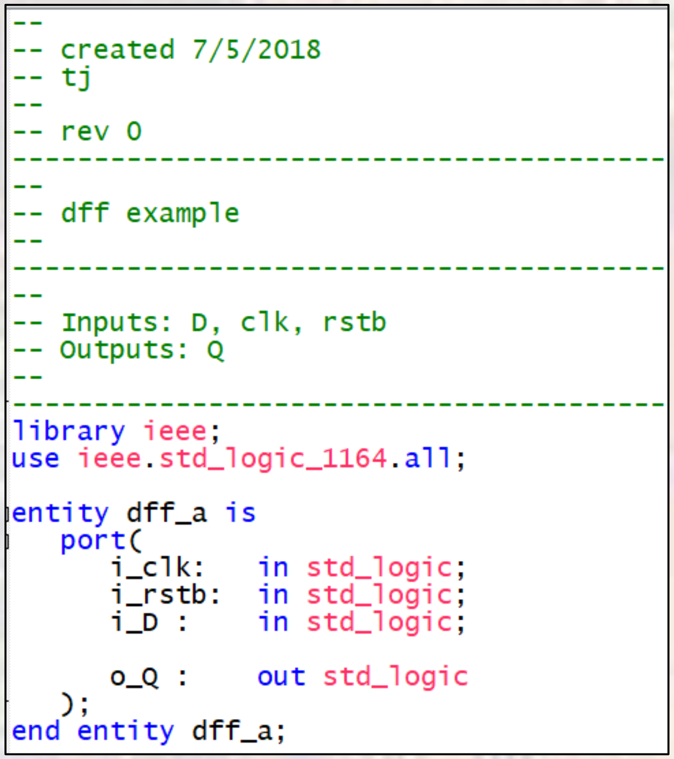
note:
- here the else is not required because the synthesizer recognizes the edge detection

- you can include an else for clarity

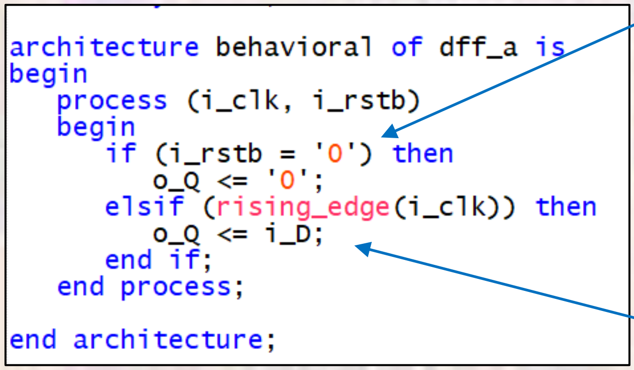
- A Flipflop is the simplest register: 1-bit register

Registers

- D-FF w/ asynchronous rstb



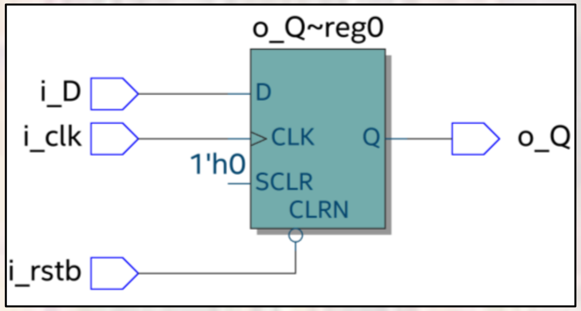
```
--
-- created 7/5/2018
-- tj
--
-- rev 0
----------------------------------------
--
-- dff example
--
----------------------------------------
--
-- Inputs: D, clk, rstb
-- Outputs: Q
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity dff_a is
    port(
        i_clk:   in std_logic;
        i_rstb:  in std_logic;
        i_D :    in std_logic;

        o_Q :    out std_logic
    );
end entity dff_a;
```



```
architecture behavioral of dff_a is
begin
    process (i_clk, i_rstb)
    begin
        if (i_rstb = '0') then
            o_Q <= '0';
        elsif (rising_edge(i_clk)) then
            o_Q <= i_D;
        end if;
    end process;

end architecture;
```

outside the rising_edge → asynchronous

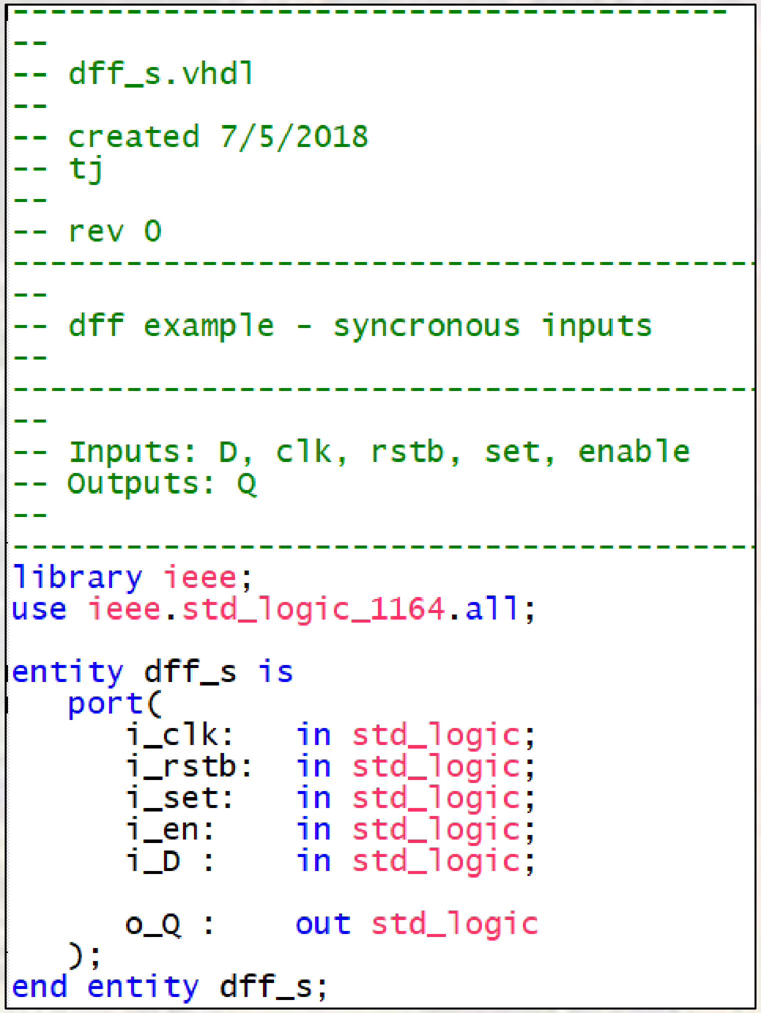
inside the rising_edge → synchronous



Most of our designs will use DFFs with an asynchronous reset
Data Path designs will use DFFs with no reset

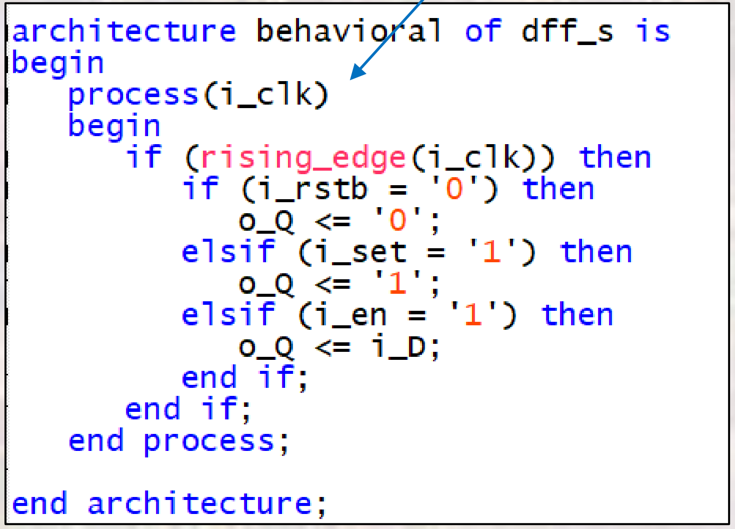
Registers

- D-FF w/ synchronous set, rstb, en



```
-------------------------------------
--
-- dff_s.vhdl
--
-- created 7/5/2018
-- tj
--
-- rev 0
----------------------------------------
--
-- dff example - syncronous inputs
--
----------------------------------------
--
-- Inputs: D, clk, rstb, set, enable
-- Outputs: Q
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity dff_s is
   port(
      i_clk:   in std_logic;
      i_rstb:  in std_logic;
      i_set:   in std_logic;
      i_en:    in std_logic;
      i_D :    in std_logic;

      o_Q :    out std_logic
   );
end entity dff_s;
```

Note rstb, set, and en are NOT in the sensitivity list



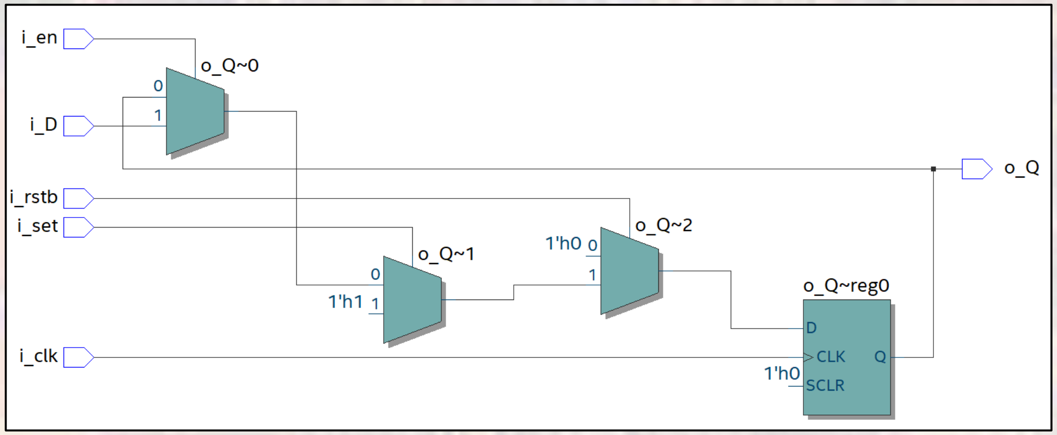
```
architecture behavioral of dff_s is
begin
   process(i_clk)
   begin
      if (rising_edge(i_clk)) then
         if (i_rstb = '0') then
            o_Q <= '0';
         elsif (i_set = '1') then
            o_Q <= '1';
         elsif (i_en = '1') then
            o_Q <= i_D;
         end if;
      end if;
   end process;

end architecture;
```

Note: there is an inherent priority in this design

Registers

- D-FF w/ synchronous set, rstb, en

```
architecture behavioral of dff_s is
begin
   process(i_clk)
   begin
      if (rising_edge(i_clk)) then
         if (i_rstb = '0') then
            o_Q <= '0';
         elsif (i_set = '1') then
            o_Q <= '1';
         elsif (i_en = '1') then
            o_Q <= i_D;
         end if;
      end if;
   end process;

end architecture;
```

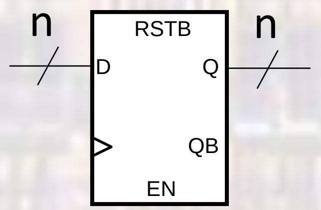


priority
rstb > set > en

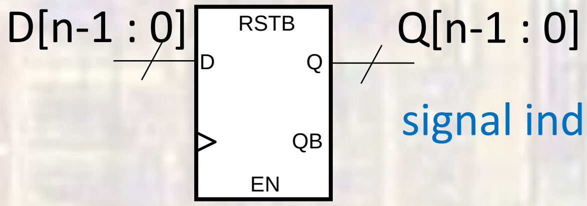
Registers

- Register
  - Collection of n Flip-Flops
  - Configured in parallel
  - Common clock, set/reset, enable
  - Stores an n-bit state variable

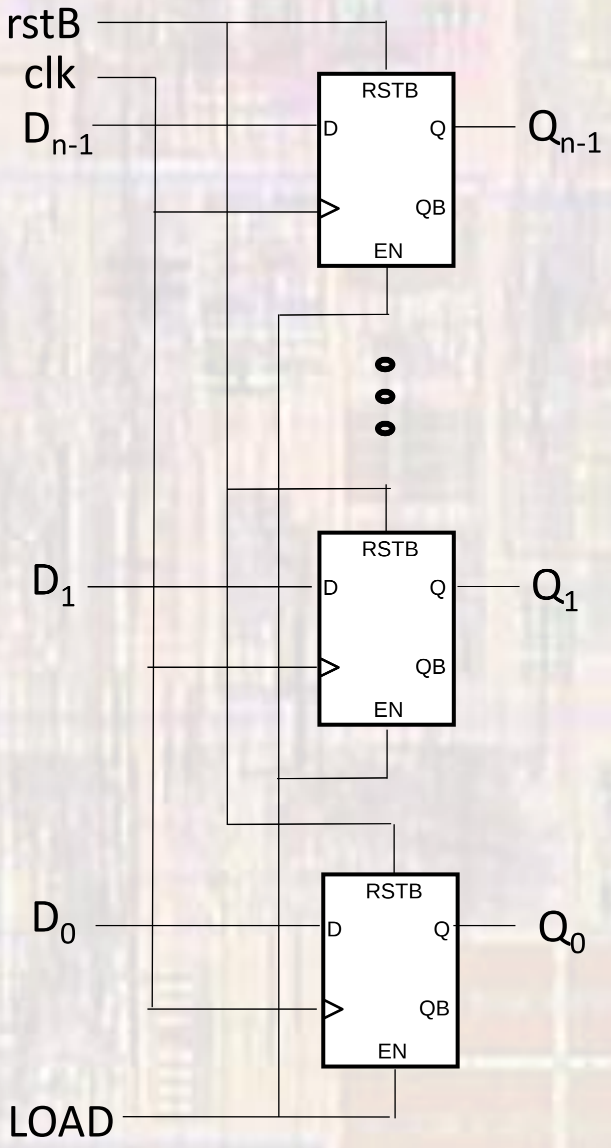
n-bit register



n bit wide bus

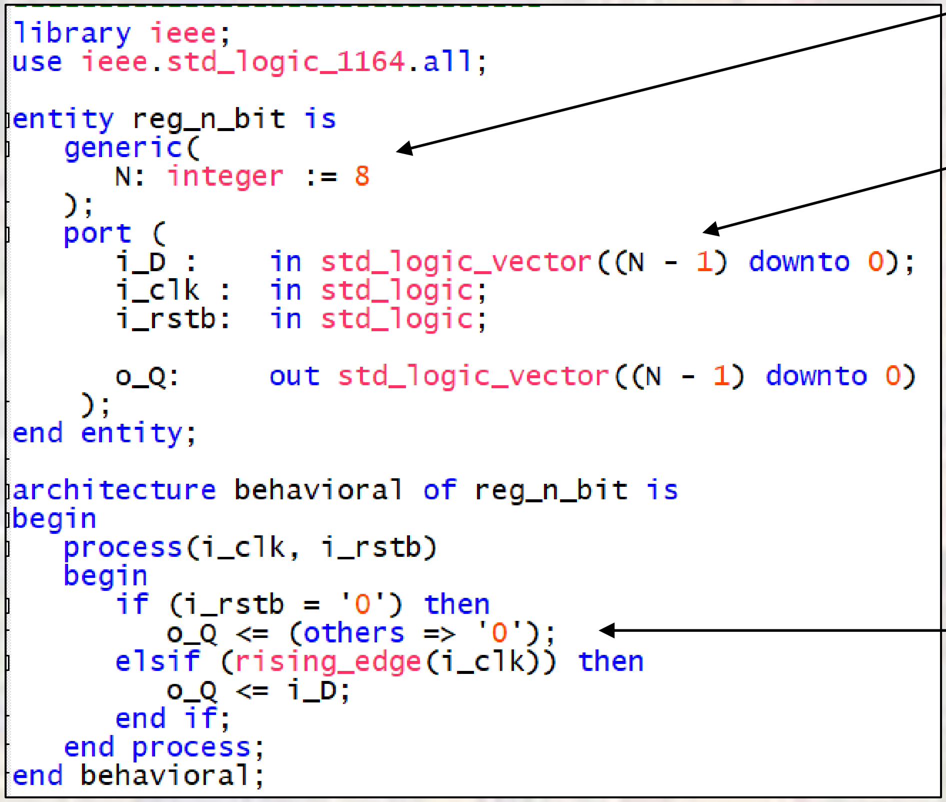


signal indices n-1 down to 0



Registers

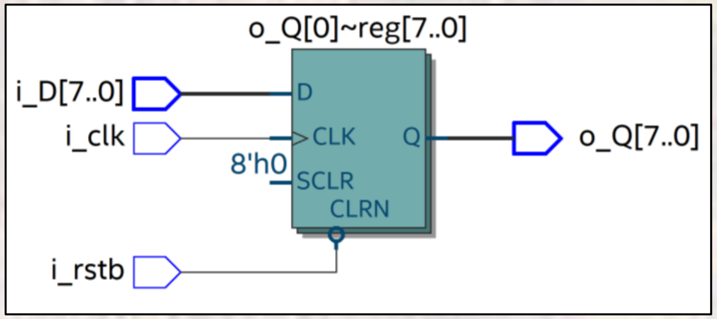
- N bit register



```
library ieee;
use ieee.std_logic_1164.all;

entity reg_n_bit is
   generic(
      N: integer := 8
   );
   port (
      i_D :    in std_logic_vector((N - 1) downto 0);
      i_clk :  in std_logic;
      i_rstb:  in std_logic;

      o_Q:     out std_logic_vector((N - 1) downto 0)
    );
end entity;

architecture behavioral of reg_n_bit is
begin
   process(i_clk, i_rstb)
   begin
      if (i_rstb = '0') then
         o_Q <= (others => '0');
      elsif (rising_edge(i_clk)) then
         o_Q <= i_D;
      end if;
   end process;
end behavioral;
```

generic section
- defines N
- defaults N to 8
- can be overwritten when instantiated

Vector sizes now defined with N

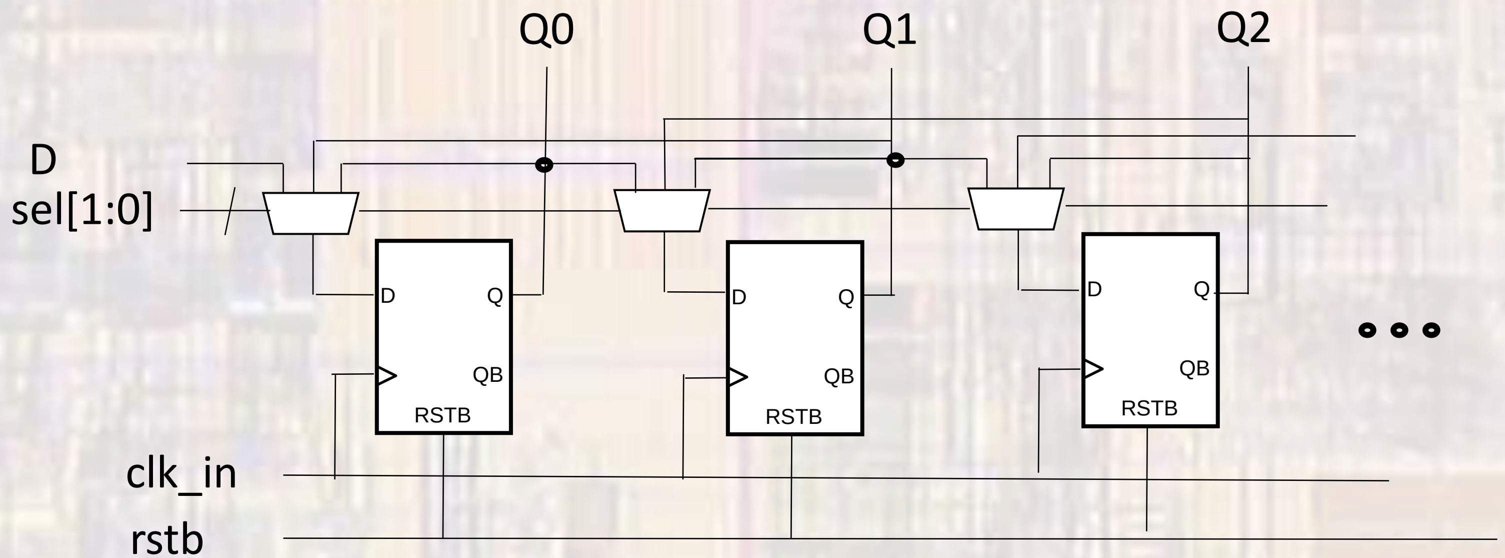


(others => '0') used since N can change

Registers

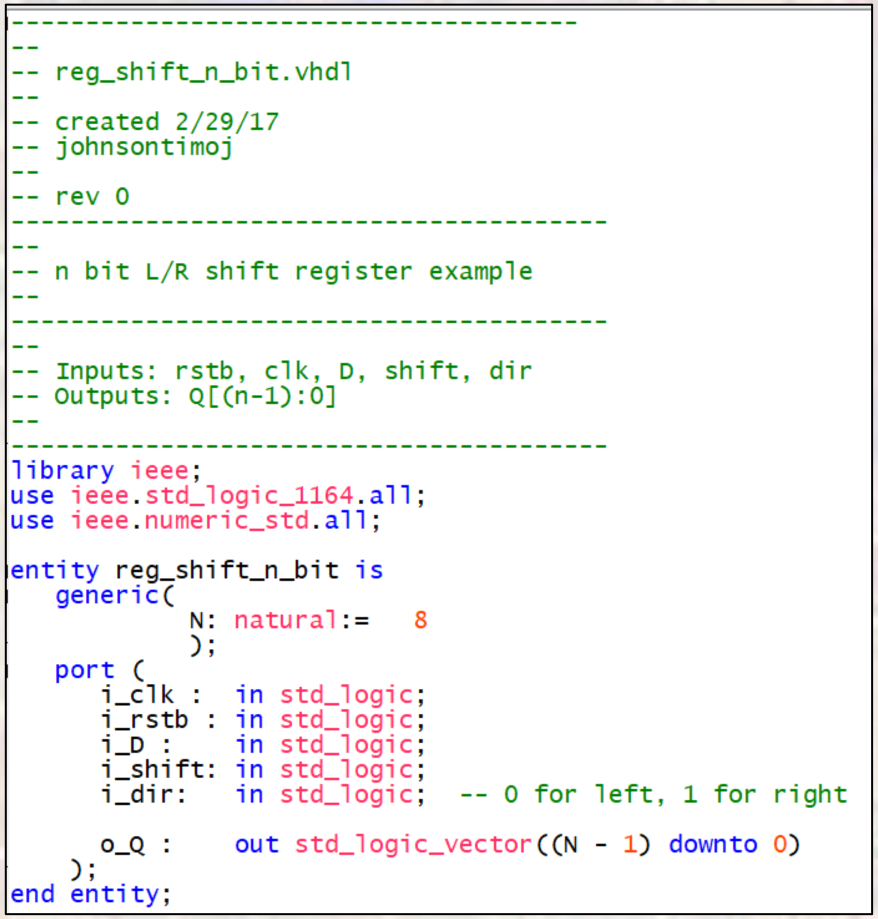
- Shift Register

  - Shift a series of bits within a register
  - sel[0]: shift right / left
  - sel[1]: shift / no-shift

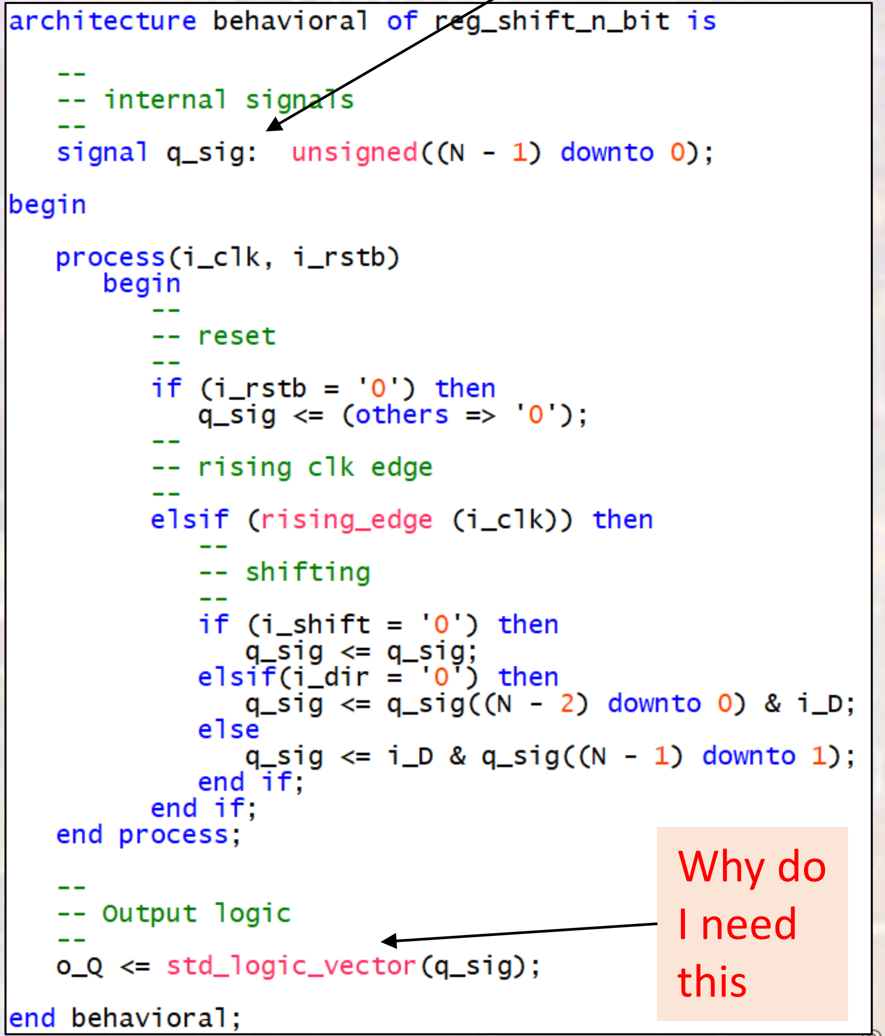


Registers

- Shift Register – n
  - D shifts in from left or right



```
----------------------------------------
--
-- reg_shift_n_bit.vhdl
--
-- created 2/29/17
-- johnsontimoj
--
-- rev 0
----------------------------------------
--
-- n bit L/R shift register example
--
----------------------------------------
--
-- Inputs: rstb, clk, D, shift, dir
-- Outputs: Q[(n-1):0]
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity reg_shift_n_bit is
   generic(
            N: natural:=   8
            );
   port (
      i_clk :  in std_logic;
      i_rstb : in std_logic;
      i_D :    in std_logic;
      i_shift: in std_logic;
      i_dir:   in std_logic;  -- 0 for left, 1 for right

      o_Q :    out std_logic_vector((N - 1) downto 0)
   );
end entity;
```



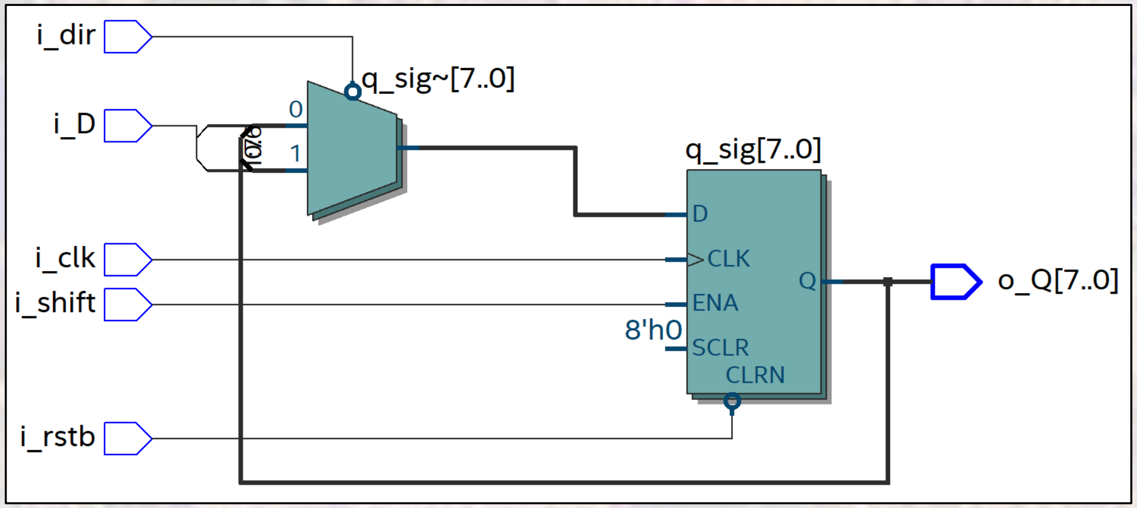
```
architecture behavioral of reg_shift_n_bit is

   --
   -- internal signals
   --
   signal q_sig:  unsigned((N - 1) downto 0);

begin

   process(i_clk, i_rstb)
      begin
         --
         -- reset
         --
         if (i_rstb = '0') then
            q_sig <= (others => '0');
         --
         -- rising clk edge
         --
         elsif (rising_edge (i_clk)) then
            --
            -- shifting
            --
            if (i_shift = '0') then
               q_sig <= q_sig;
            elsif(i_dir = '0') then
               q_sig <= q_sig((N - 2) downto 0) & i_D;
            else
               q_sig <= i_D & q_sig((N - 1) downto 1);
            end if;
         end if;
   end process;

   --
   -- Output logic
   --
   o_Q <= std_logic_vector(q_sig);

end behavioral;
```

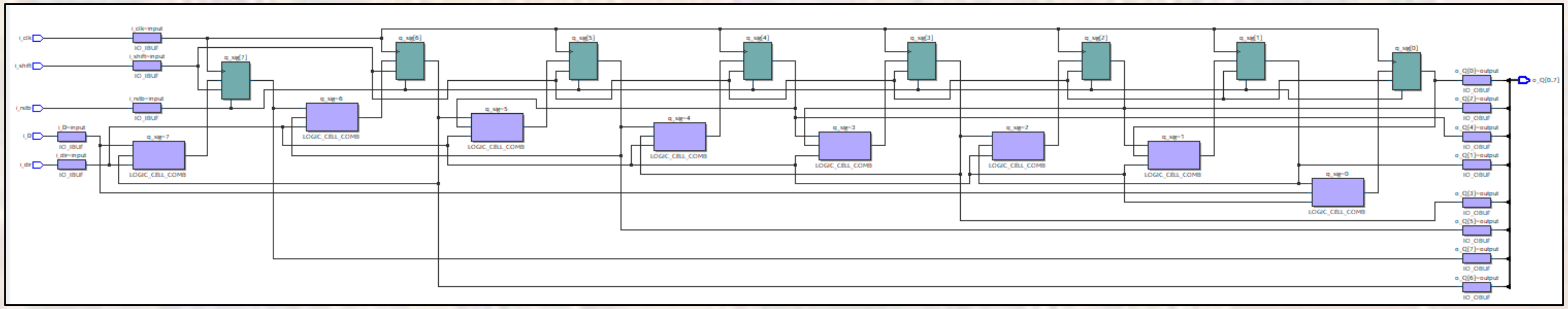
Why do I need this

Why do I need this

Registers

- Shift Register - n





Registers

- Shift Register - n

```
-------------------------------
--
-- reg_shift_n_bit_tb.vhdl
--
-- created: 1/26/18
-- by: johnsontimoj
-- rev: 0
--
-- testbench for n bit shift register
-- of reg_shift_n_bit.vhdl
--
-- brute force implementation
--
--------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity reg_shift_n_bit_tb is
   generic(
            N: natural := 16
         );
   -- no entry - testbench
end entity;

architecture testbench of reg_shift_n_bit_tb is
   signal   CLK:     std_logic;
   signal   RSTB:    std_logic;
   signal   D:       std_logic;
   signal   SHIFT:   std_logic;
   signal   DIR:     std_logic;

   signal   Q:       std_logic_vector((N - 1) downto 0);

   constant PER:     time := 20 ns;
```

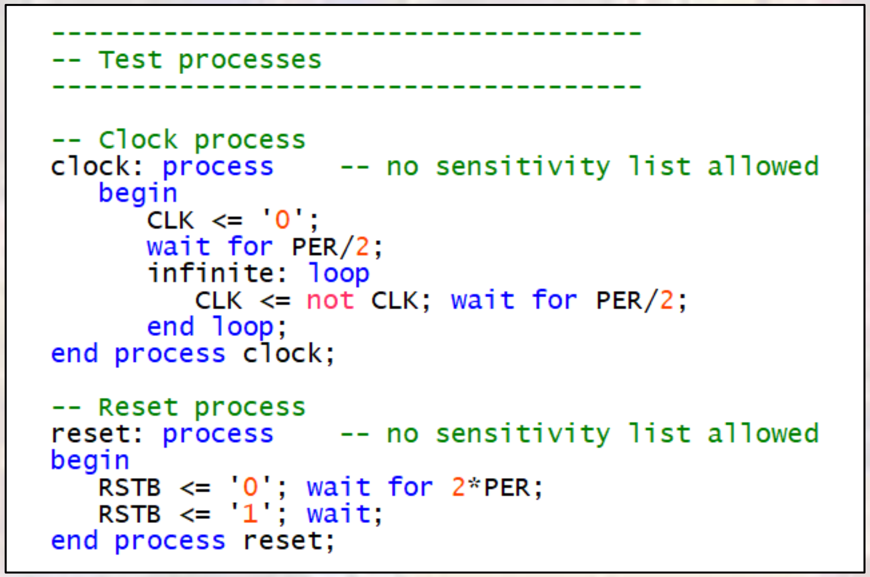
testbench generic (only set once)

```
--------------------------------
-- Component prototype
---------------------------------
COMPONENT reg_shift_n_bit
   GENERIC ( N : INTEGER := 8 );
   PORT
   (
      i_clk     :    IN STD_LOGIC;
      i_rstb    :    IN STD_LOGIC;
      i_D       :    IN STD_LOGIC;
      i_shift   :    IN STD_LOGIC;
      i_dir     :    IN STD_LOGIC;
      o_Q       :    OUT STD_LOGIC_VECTOR(n-1 DOWNTO 0)
   );
END COMPONENT;
-----------------------------------

begin

----------------------------------
-- Device under test (DUT)
----------------------------------
DUT: reg_shift_n_bit
   generic map(
               N => N
   )
   port map(
            i_clk     => CLK,
            i_rstb    => RSTB,
            i_D       => D,
            i_shift   => SHIFT,
            i_dir     => DIR,
            o_Q       => Q
            );
```

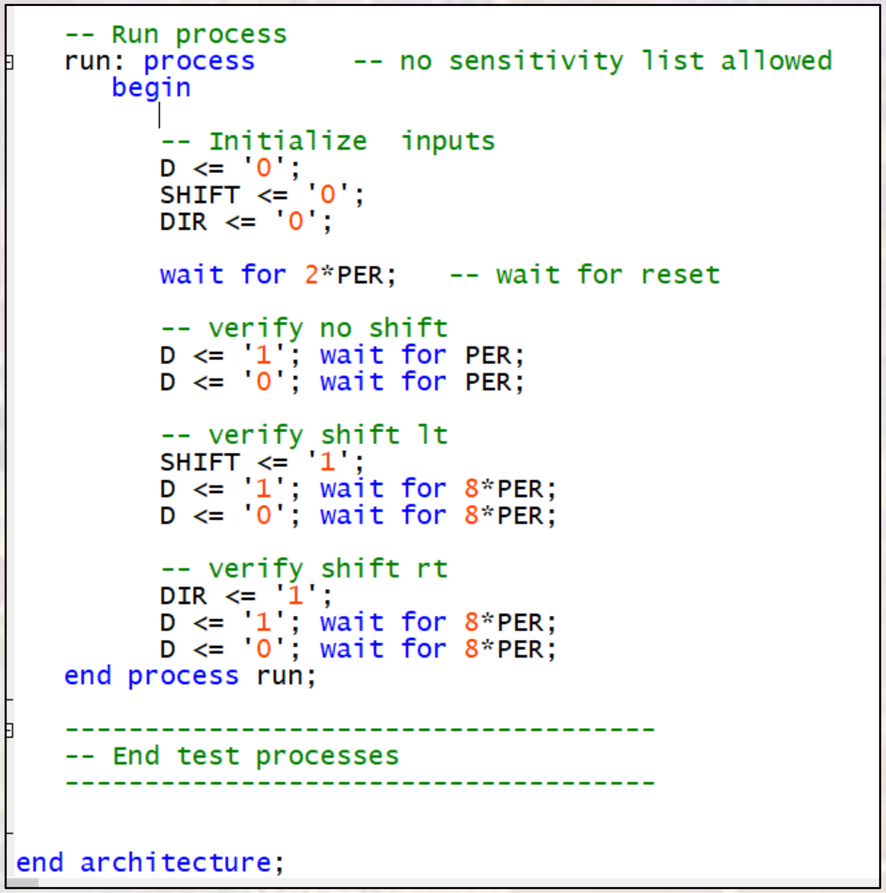
module generic (set by TB)

Registers

- Shift Register - n



```
---------------------------------------
-- Test processes
---------------------------------------

-- Clock process
clock: process    -- no sensitivity list allowed
   begin
      CLK <= '0';
      wait for PER/2;
      infinite: loop
         CLK <= not CLK; wait for PER/2;
      end loop;
end process clock;

-- Reset process
reset: process    -- no sensitivity list allowed
begin
   RSTB <= '0'; wait for 2*PER;
   RSTB <= '1'; wait;
end process reset;
```

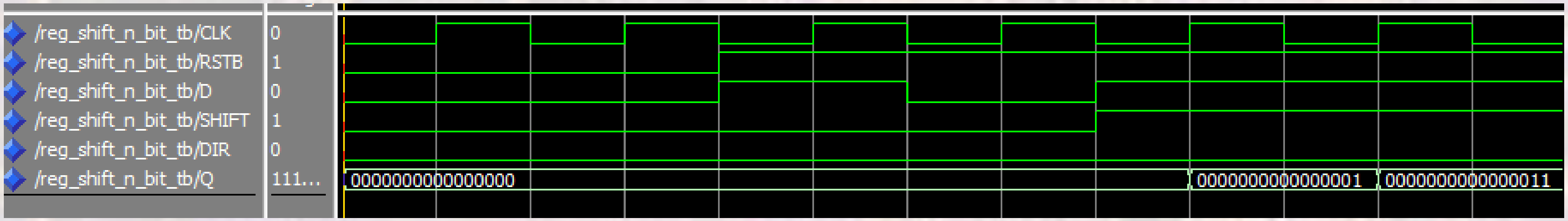


```
-- Run process
run: process      -- no sensitivity list allowed
   begin

      -- Initialize  inputs
      D <= '0';
      SHIFT <= '0';
      DIR <= '0';

      wait for 2*PER;    -- wait for reset

      -- verify no shift
      D <= '1'; wait for PER;
      D <= '0'; wait for PER;

      -- verify shift lt
      SHIFT <= '1';
      D <= '1'; wait for 8*PER;
      D <= '0'; wait for 8*PER;

      -- verify shift rt
      DIR <= '1';
      D <= '1'; wait for 8*PER;
      D <= '0'; wait for 8*PER;
end process run;

---------------------------------------
-- End test processes
---------------------------------------


end architecture;
```

Registers

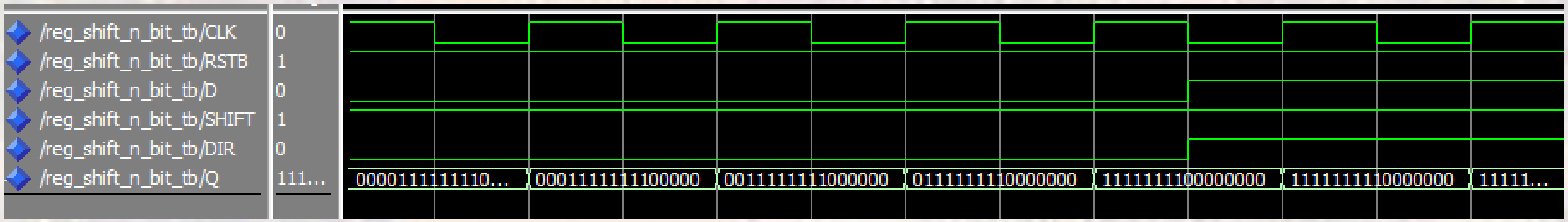
- Shift Register

reset no shift 1 in , shift left



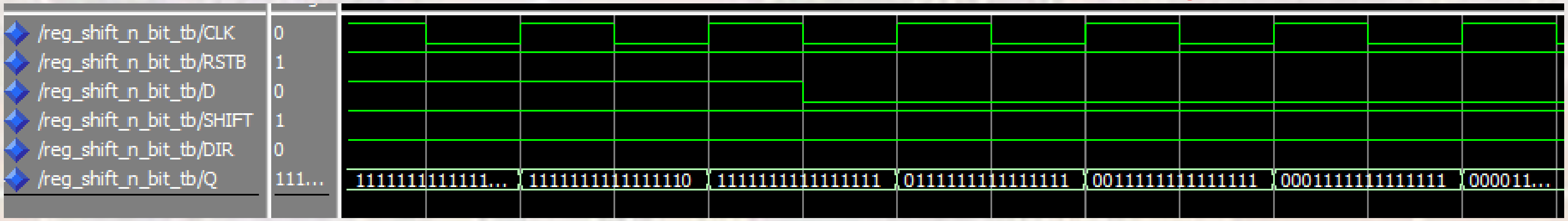
/reg_shift_n_bit_tb/CLK 0
/reg_shift_n_bit_tb/RSTB 1
/reg_shift_n_bit_tb/D 0
/reg_shift_n_bit_tb/SHIFT 1
/reg_shift_n_bit_tb/DIR 0
/reg_shift_n_bit_tb/Q 111...

0 in , shift left 1 in , shift right



/reg_shift_n_bit_tb/CLK 0
/reg_shift_n_bit_tb/RSTB 1
/reg_shift_n_bit_tb/D 0
/reg_shift_n_bit_tb/SHIFT 1
/reg_shift_n_bit_tb/DIR 0
/reg_shift_n_bit_tb/Q 111...

0 in , shift right



/reg_shift_n_bit_tb/CLK 0
/reg_shift_n_bit_tb/RSTB 1
/reg_shift_n_bit_tb/D 0
/reg_shift_n_bit_tb/SHIFT 1
/reg_shift_n_bit_tb/DIR 0
/reg_shift_n_bit_tb/Q 111...