Registers

Last updated 2/8/24

Registers

- Registers are collections of Flip-Flops
- Registers are created by creating code that conforms to the Flip-Flop synthesis 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

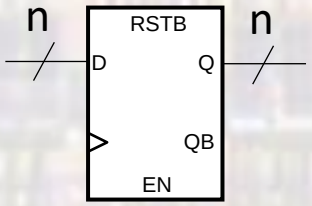
- you can include an else for clarity

© tj

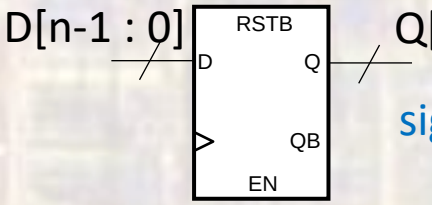
Registers

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

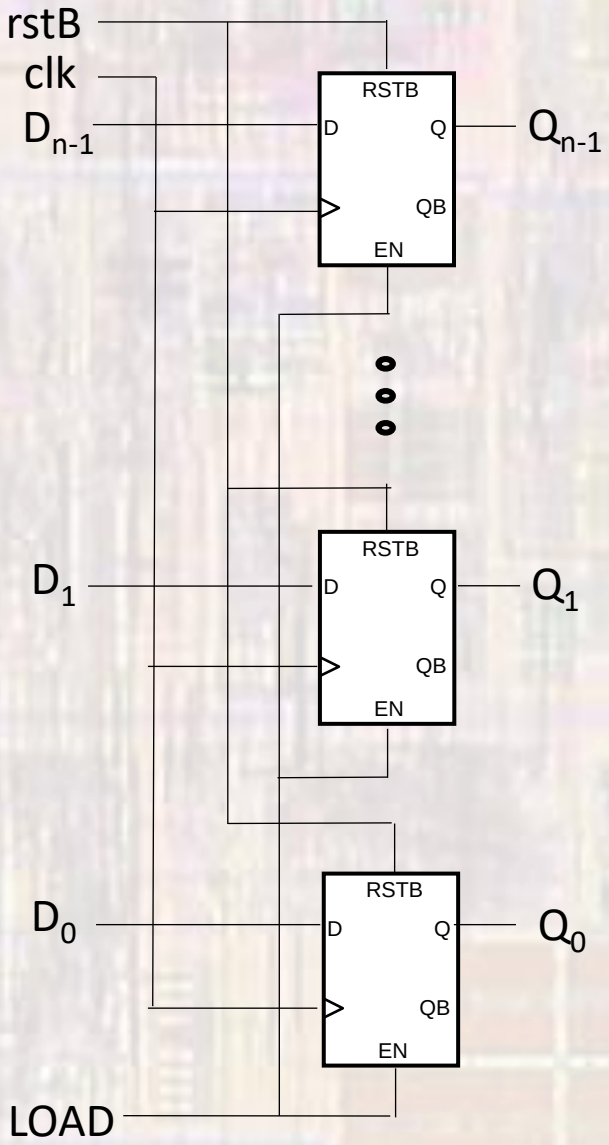
n-bit register



n bit wide bus



signal indices n-1 down to 0



© tj

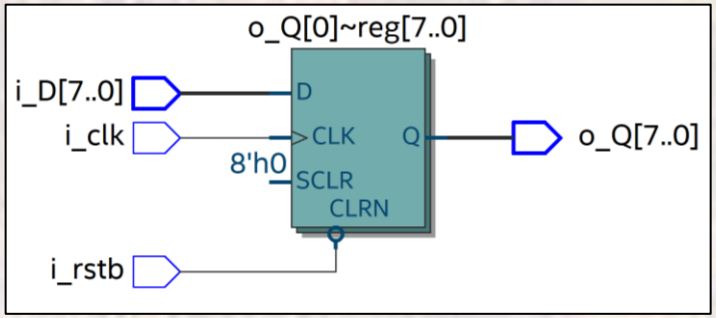
Registers

- N bit data 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

© tj

Registers

- N bit data register
  - Enhancements
    - Asynch/synch reset, set
    - Clear
    - Load
    - Preload
    - Enable

© tj

Registers

- Shift Register
  - Shift a series of bits within a register
  - Left/right
  - N bit shift
  - Wrapping
    - Load register with load signal
    - Direction, amount
  - Non-wrapping
    - Data input
    - Sign extend
    - 0/1 pad
  - Depending on what the shift register represents use
    - std_logic_vector, unsigned or signed

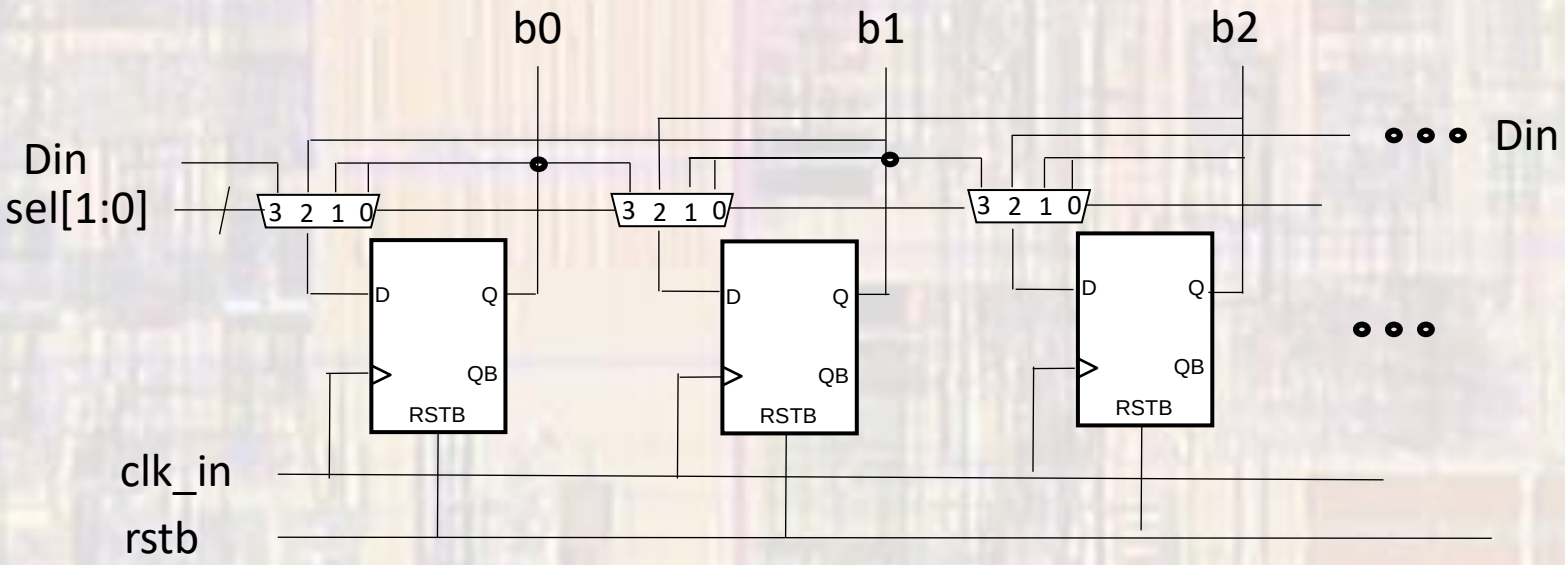
DATA	dir	amt	result
00100	R	1	00010
00010	R	3	01000
01000	L	2	00001

Din	DATA	dir	result
1	00100	R	10010
0	10010	R	01001
1	01001	L	10011

© tj

Registers

- Shift Register
  - Non-wrapping, Data in (from both ends)
    - sel[0]: dir : right(1) / left(0)
    - sel[1]: shift: shift(1) / no-shift(0)
    - Unsigned data



© tj

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;
```

Why do I need this

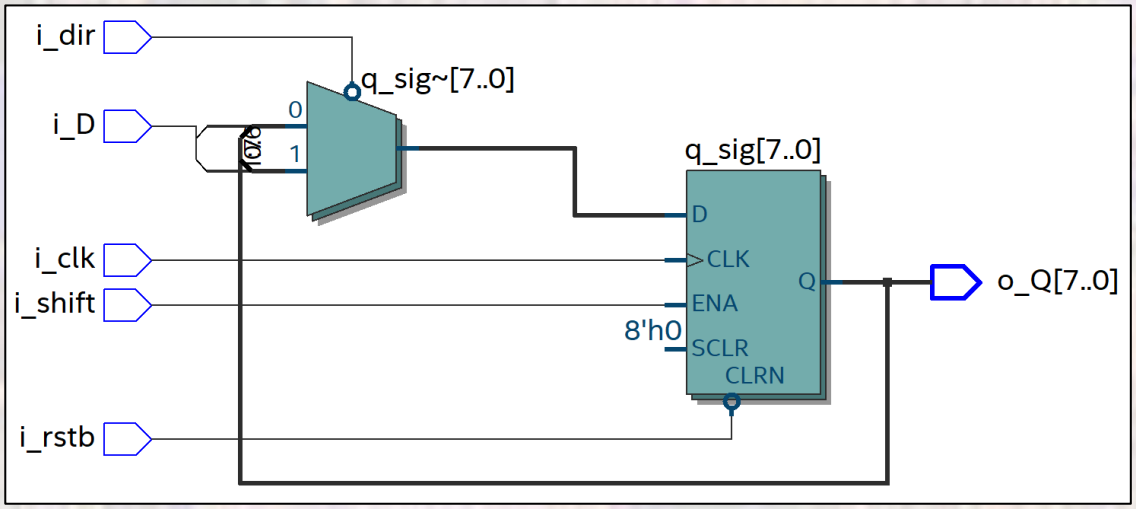
```
architecture behavioral of reg_shift_n_bit is

   --
   -- 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;
```

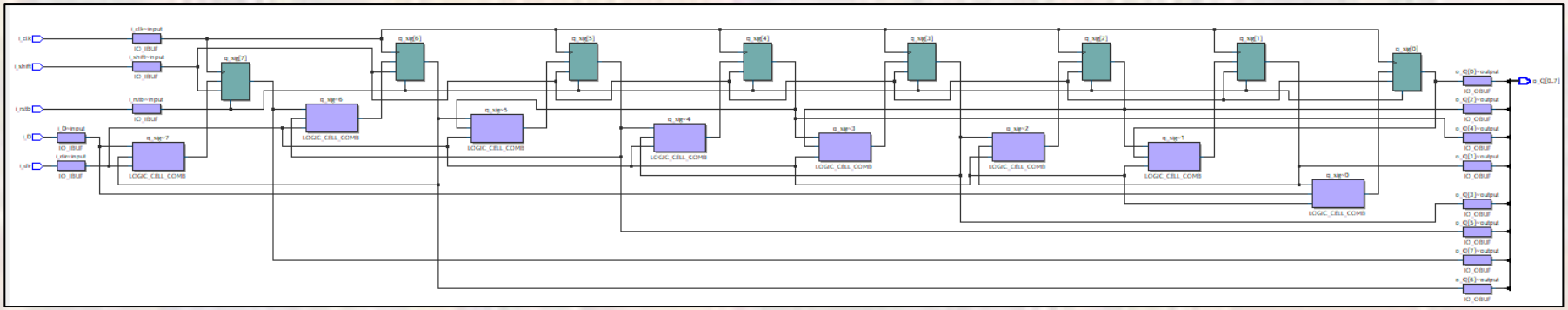
Why do I need this

© tj

Registers

- Shift Register - n

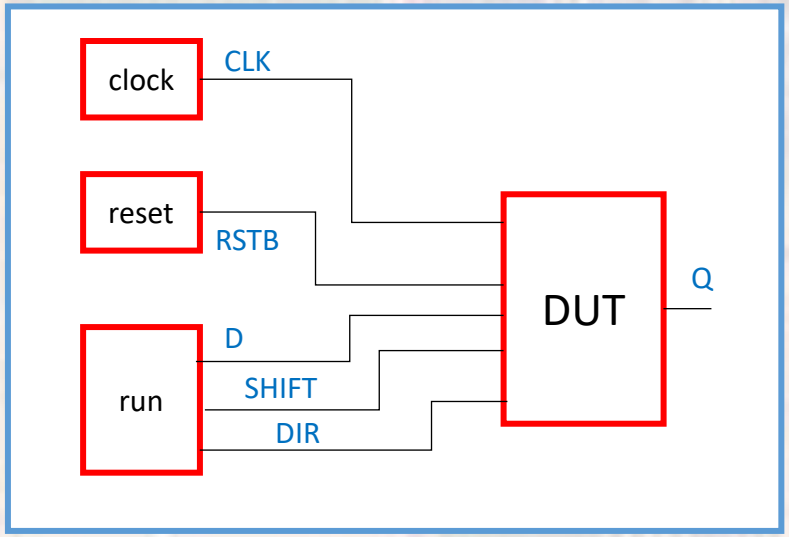




 © tj

Registers

- Shift Register – n
  - Testbench



 © tj

Registers

- Shift Register – n
  - Testbench

```
--------------------------------
--
-- 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(
            NUM_BITS: natural := 16
          );
   -- no entry - testbench
end entity;
```

testbench generic

```
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((NUM_BITS - 1) downto 0);

   constant PER:      time := 20 ns;

   ---------------------------------
   -- 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 => NUM_BITS
      )
      port map(
               i_clk     => CLK,
               i_rstb    => RSTB,
               i_D       => D,
               i_shift   => SHIFT,
               i_dir     => DIR,
               o_Q       => Q
               );
```

module generic (reset by TB)

© tj

Registers

- Shift Register – n
  - Testbench

```
-------------------------------------
-- 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;
```

Note comments

```
   -- 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;
```


© tj

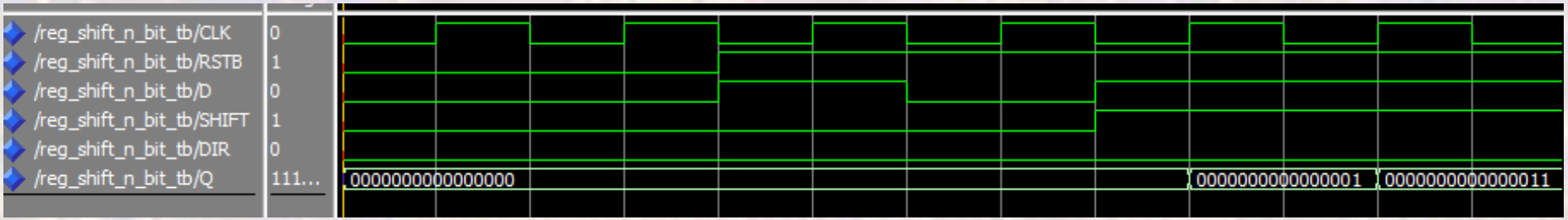
Registers

- Shift Register
  - Sim Results

reset

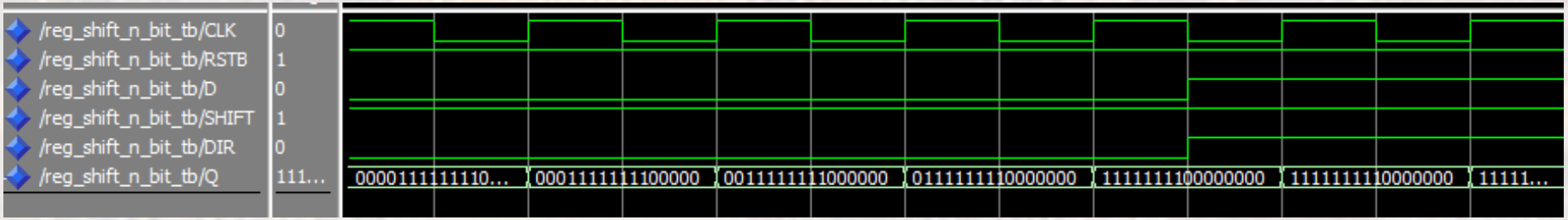
no shift

1 in , shift left

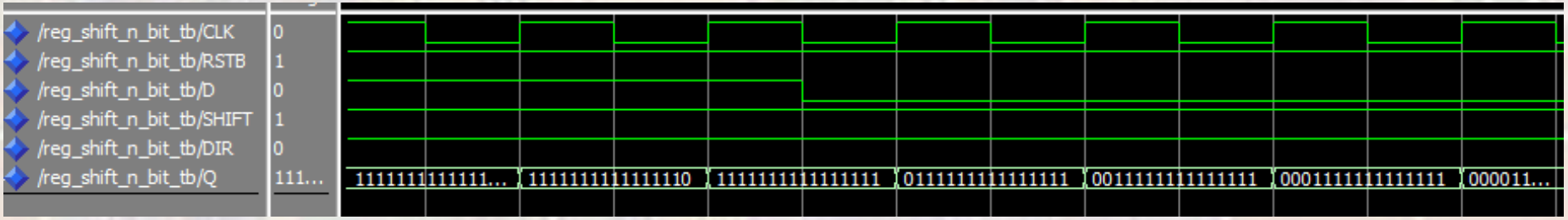


0 in , shift left

1 in , shift right



0 in , shift right



 © tj