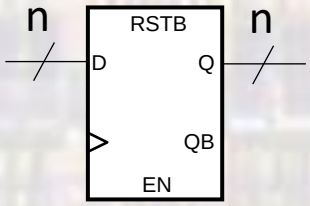
VHDL Data Registers

Last updated 1/9/25

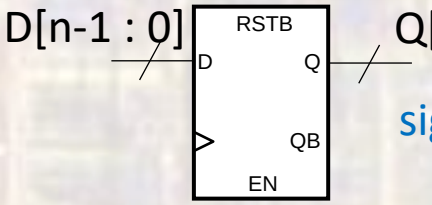
VHDL Data Registers

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

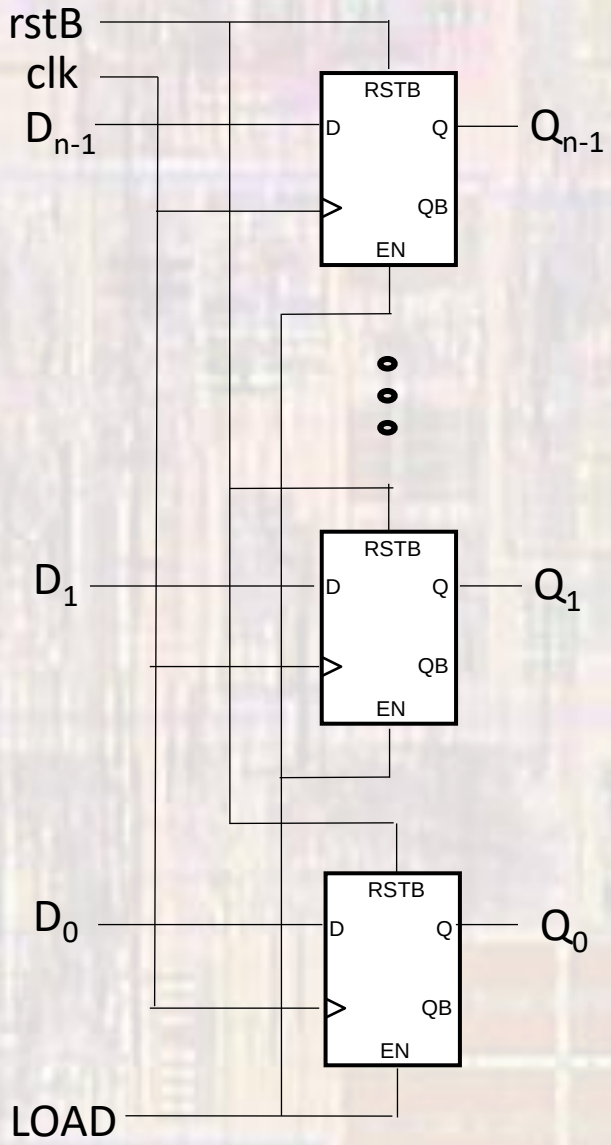
n-bit register



n bit wide bus



signal indices n-1 down to 0



VHDL Data 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

VHDL Data Registers

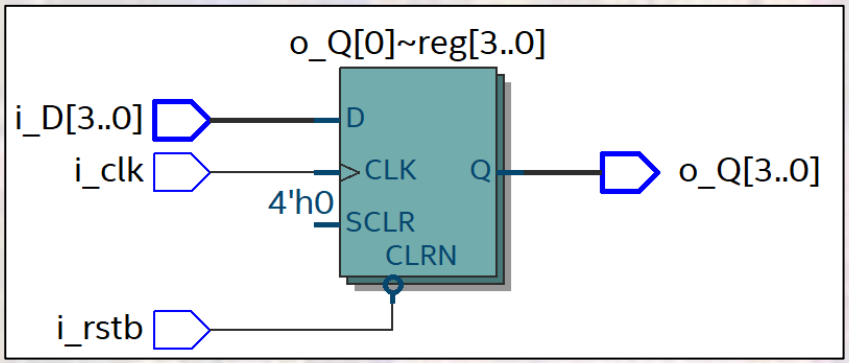
- 4 bit data register

```
----------------------------------
--
--  reg_4bit.vhdl
--
--  by: johnsontimoj
--
--  created: 12/31/24
--
--  version: 0.0
--
----------------------------------
--
--  standard 4 bit register
--  inputs: clk, rstb, d
--
--  outputs: q
--
----------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity reg_4bit is
    port(
        i_clk:   in std_logic;
        i_rstb:  in std_logic;
        i_D :    in std_logic_vector(3 downto 0);

        o_Q :    out std_logic_vector(3 downto 0)
    );
end entity;
```

```
architecture behavioral of reg_4bit is

begin

    process (i_clk, i_rstb)
    begin
        if (i_rstb = '0') then
            o_Q <= "0000";
        elsif (rising_edge(i_clk)) then
            o_Q <= i_D;
        end if;
    end process;

end architecture;
```

VHDL Data Registers

- 4 bit data register



Flow Summary

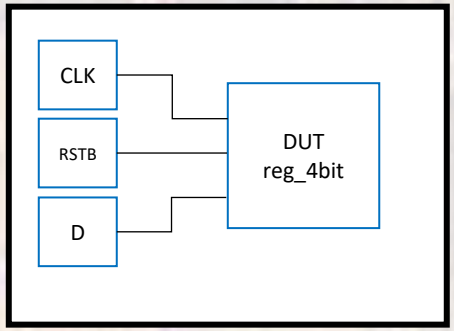
<<Filter>>

Flow Status	Successful - Thu Jan 09 11:52:16 2025
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	general
Top-level Entity Name	reg_4bit
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	4
Total registers	4
Total pins	10
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

VHDL Data Registers

- 4 bit data register - testbench

```
----------------------------------
--
--  reg_4bit_tb.vhdl
--
--  by: johnsontimoj
--
--  created: 12/31/24
--
--  version: 0.0
--
----------------------------------
--
--  testbench for standard 4 bit register
--  inputs: clk, rstb, d
--
--  outputs: q
--
----------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity reg_4bit_tb is
   -- no ports in TB
end entity;
```

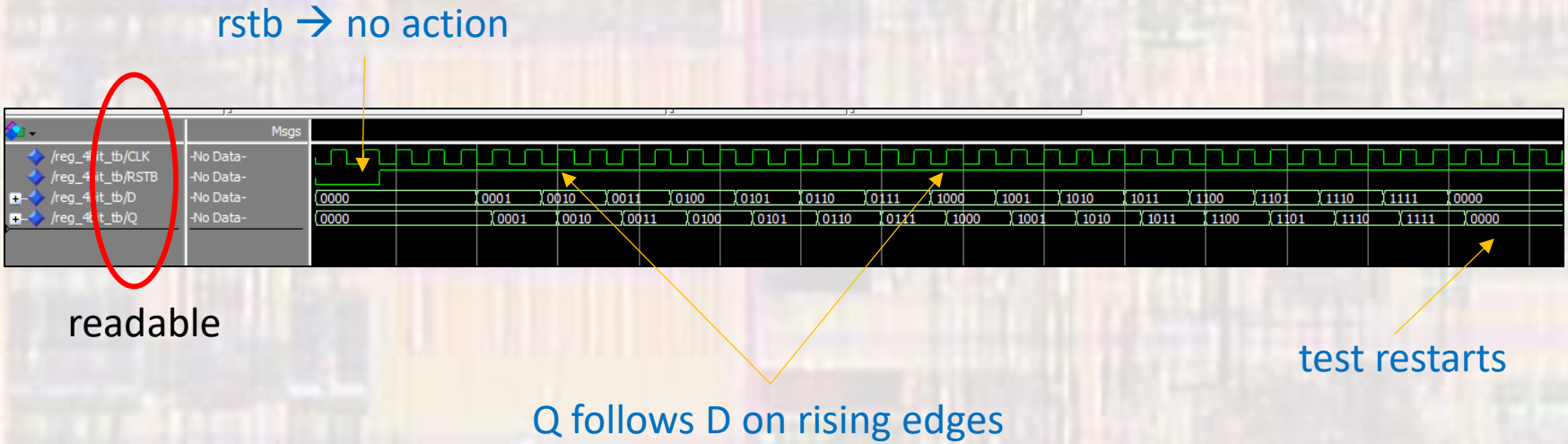



```
architecture testbench of reg_4bit_tb is

   signal CLK:    std_logic;
   signal RSTB:   std_logic;
   signal D:      std_logic_vector(3 downto 0);
   signal Q:      std_logic_vector(3 downto 0);

   constant PER:  time  := 20 ns;

   component reg_4bit is
      port(
         i_clk:   in std_logic;
         i_rstb:  in std_logic;
         i_D :    in std_logic_vector(3 downto 0);

         o_Q :    out std_logic_vector(3 downto 0)
      );
   end component;
```

```
   ----------------------------------
   -- Device under test (DUT)
   ----------------------------------
   DUT: reg_4bit
      port map(
               i_clk       => CLK,
               i_rstb      => RSTB,
               i_d         => D,
               o_q         => Q
      );


   ----------------------------------
   -- Test processes
   ----------------------------------

   -- Clock process
   clock: process
      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
   begin
      RSTB <= '0'; wait for 2*PER;
      RSTB <= '1'; wait;
   end process reset;

   -- Run process
   data: process
   begin
      -- initialize D
      D <= std_logic_vector(to_unsigned(0, 4));

      -- wait for reset
      wait for 3 * PER;

      -- loop exhaustive inputs
      binary_cnt: for i in 0 to 15 loop
         D <= std_logic_vector(to_unsigned(i, 4));
         wait for 2 * PER;
      end loop;
   end process;

end architecture;
```

VHDL Data Registers

- 4 bit data register - testbench



VHDL Data Registers

- N bit data register

```
----------------------------------
--
--  reg_nbit.vhdl
--
--  by: johnsontimoj
--
--  created: 12/31/24
--
--  version: 0.0
--
----------------------------------
--
--  standard N bit register
--  inputs: clk, rstb, d
--
--  outputs: q
--
----------------------------------
library ieee;
use ieee.std_logic_1164.all;

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

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

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

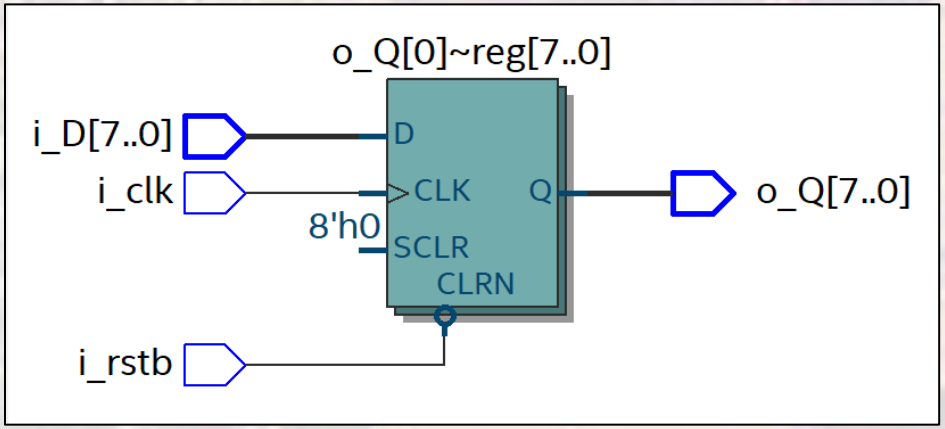
(others => '0') used since N can change

```
architecture behavioral of reg_nbit 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 architecture;
```

Vector sizes now defined with N

VHDL Data Registers

- N bit data register



Flow Summary

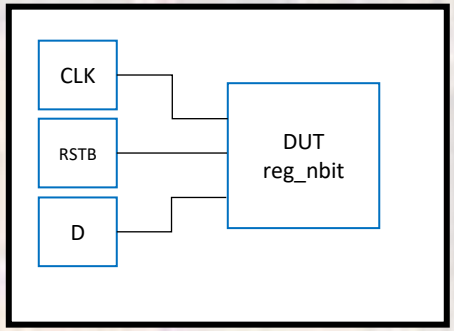
<<Filter>>

Flow Status	Successful - Thu Jan 09 12:06:01 2025
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	general
Top-level Entity Name	reg_nbit
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	8
Total registers	8
Total pins	18
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

VHDL Data Registers

- n bit data register - testbench

```
----------------------------------
--
--  reg_nbit_tb.vhdl
--
--  by: johnsontimoj
--
--  created: 12/31/24
--
--  version: 0.0
--
----------------------------------
--
--  testbench for standard n bit register
--  inputs: clk, rstb, d
--
--  outputs: q
--
----------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity reg_nbit_tb is
   generic(
            NN:      positive := 4
          );
   -- no ports in TB
end entity;
```

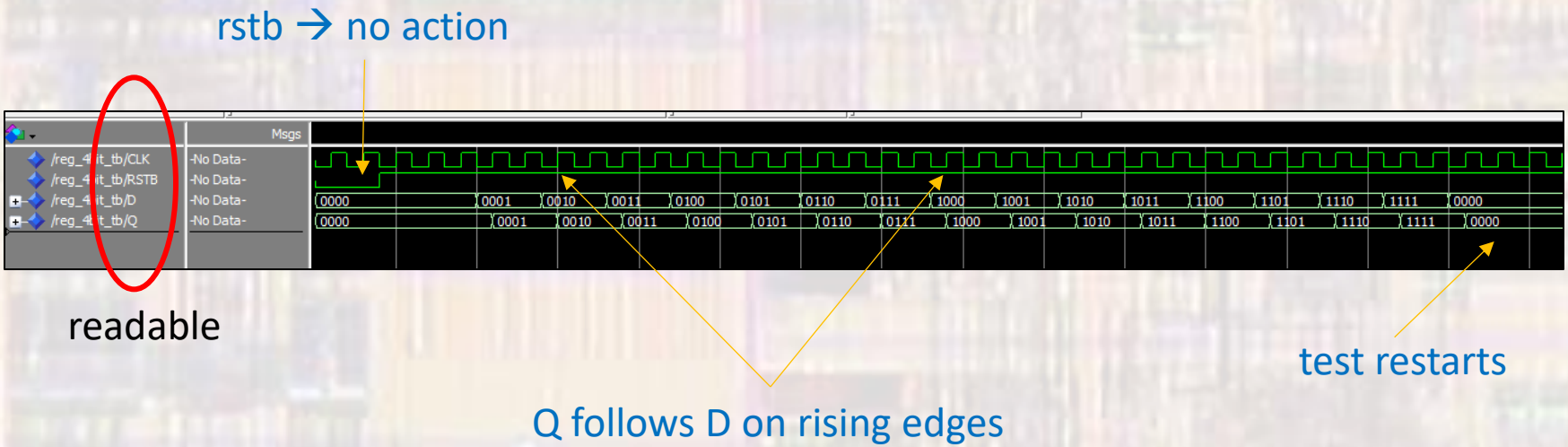



```
architecture testbench of reg_nbit_tb is

   signal CLK:    std_logic;
   signal RSTB:   std_logic;
   signal D:      std_logic_vector((NN - 1) downto 0);
   signal Q:      std_logic_vector((NN - 1) downto 0);

   constant PER:  time  := 20 ns;

   component reg_nbit is
   generic(
            N:    positive := 8
          );
   port(
         i_clk:   in std_logic;
         i_rstb:  in std_logic;
         i_D :    in std_logic_vector((N - 1) downto 0);

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

```
begin

   ----------------------------------
   -- Device under test (DUT)
   ----------------------------------
   DUT: reg_nbit
   generic map(
               N  =>    NN
               )
   port map(
               i_clk       => CLK,
               i_rstb      => RSTB,
               i_d         => D,
               o_q         => Q
      );


   ----------------------------------
   -- Test processes
   ----------------------------------

   -- Clock process
   clock: process
      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
   begin
      RSTB <= '0'; wait for 2*PER;
      RSTB <= '1'; wait;
   end process reset;

   -- Run process
   data: process
   begin
      -- initialize D
      D <= std_logic_vector(to_unsigned(0, NN));

      -- wait for reset
      wait for 3 * PER;

      -- loop exhaustive inputs
      binary_cnt: for i in 0 to (2**NN - 1) loop
         D <= std_logic_vector(to_unsigned(i, NN));
         wait for 2 * PER;
      end loop;
   end process;

end architecture;
```

VHDL Data Registers

- n bit data register – testbench
  - Default for the nbit register is 8 bits
  - Instantiated a 4 bit version



VHDL Data Registers

- Data Register Enhancements
  - Asynch/synch reset, set
  - Clear
  - Load
  - Preload
  - Enable