Universidade Federal de Santa Catarina
Centro Tecnológico – CTC
Departamento de Engenharia Elétrica

"EEL5105 - Circuitos e Técnicas Digitais"

Prof. Héctor Pettenghi Roldán*

Hector@eel.ufsc.br

Florianópolis, março de 2017.

***Baseados nos slides do Professor Eduardo Bezerra e Eduardo Batista EEL5105 2015.2**

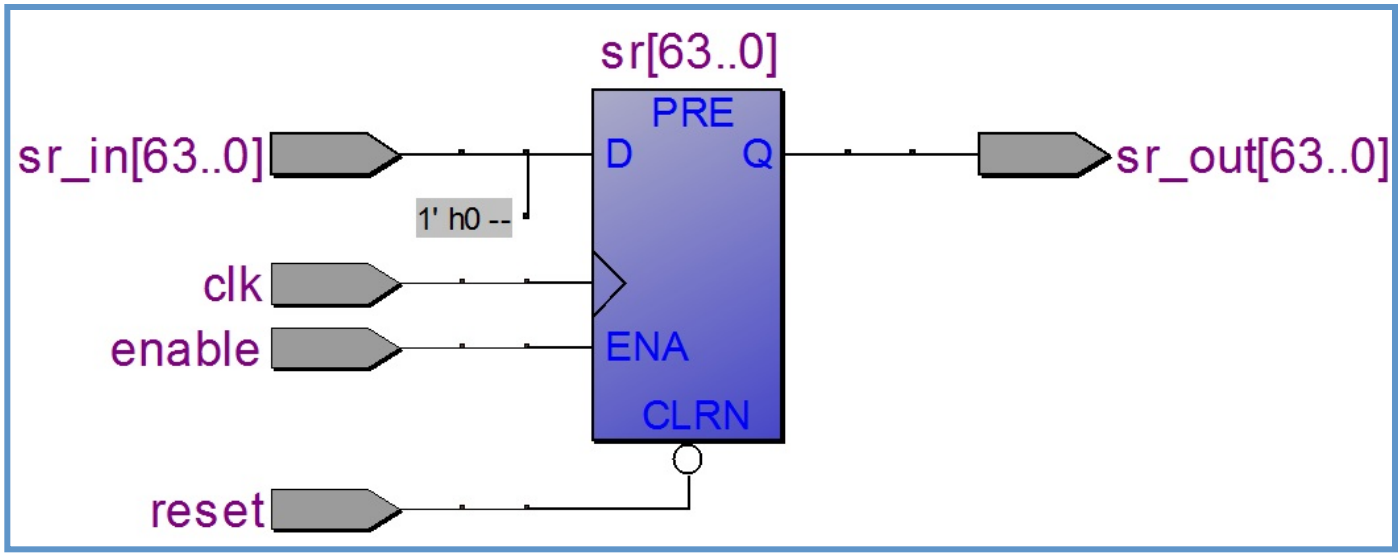
Estudo de caso: uso de processo explícito para implementar registrador com reset assíncrono, clock e sinal de enable

Deslocamento de vetor de entrada 1 bit à esquerda (1/2)

```
-- sr_in recebe palavra de N bits (vetor de N bits). A cada
-- pulso de clock, a palavra em sr_in é deslocada 1 bit para a
-- esquerda, e copiada para sr_out, também de N bits.
library ieee;
use ieee.std_logic_1164.all;

entity desloc_1_bit_esq is
   generic  ( N : natural := 64 );
   port       (  clk          : in std_logic;
                 enable       : in std_logic;
                 reset        : in std_logic;
                 sr_in        : in std_logic_vector((N - 1) downto 0);
                 sr_out       : out std_logic_vector((N - 1) downto 0)
              );
end entity;
```

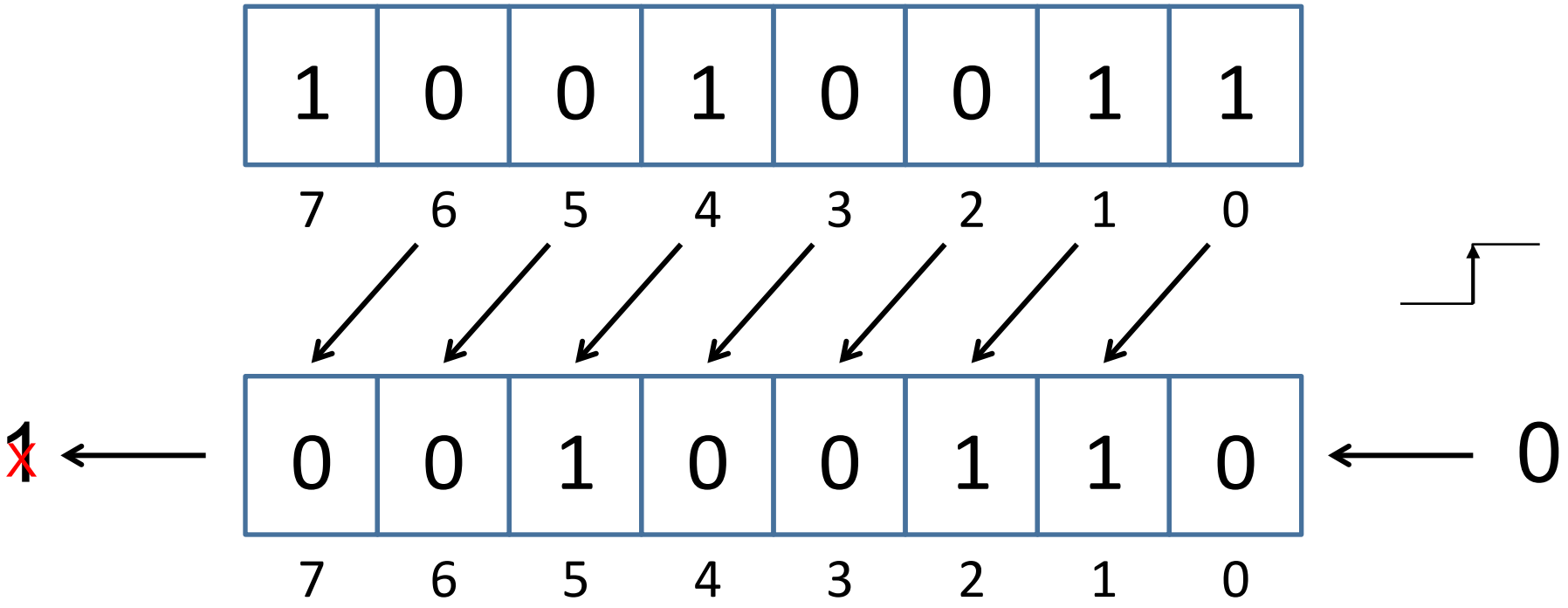
Deslocamento de vetor de entrada 1 bit à esquerda (2/2)

```
architecture rtl of desloc_1_bit_esq is
   signal sr: std_logic_vector ((N - 1) downto 0); -- Registrador de N bits
begin
   process (clk, reset)
   begin
      if (reset = '0') then                -- Reset assíncrono do registrador
         sr <= (others => '0');
      elsif (rising_edge(clk)) then   -- Sinal de clock do registrador (subida)
         if (enable = '1') then           -- Sinal de enable do registrador
            -- Desloca 1 bit para a esquerda. Bit mais significativo é perdido.
            sr((N - 1) downto 1) <= sr_in((N - 2) downto 0);
            sr(0) <= '0';
         end if;
      end if;
   end process;
   sr_out <= sr;
end rtl;
```



Deslocamento de vetor de entrada 1 bit à esquerda

Valor de entrada em *sr_in* = 93H



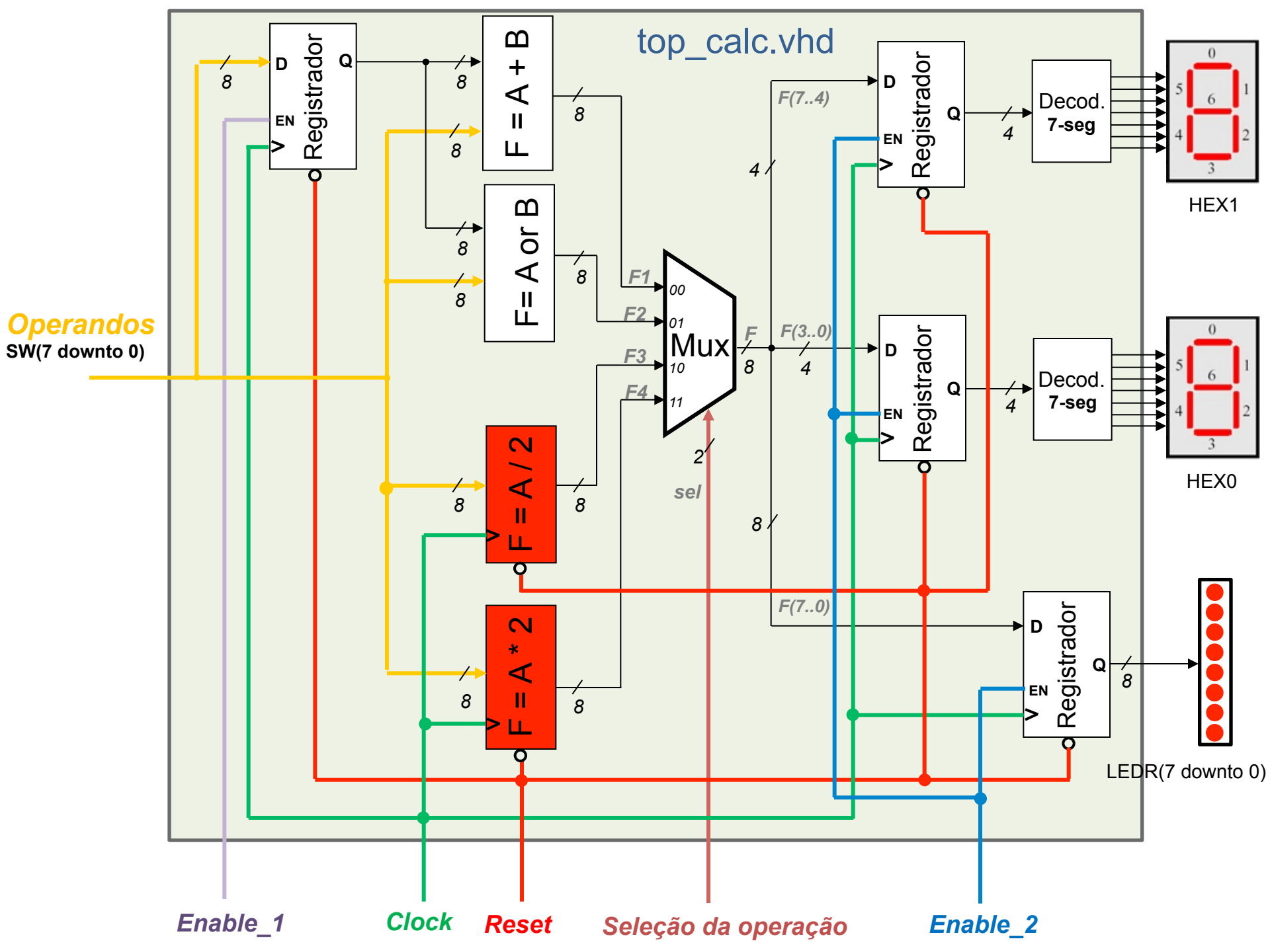
Valor de saída em *sr_out* = 26H

Tarefa a ser realizada:
mini-calculadora com multiplicação e divisão por 2

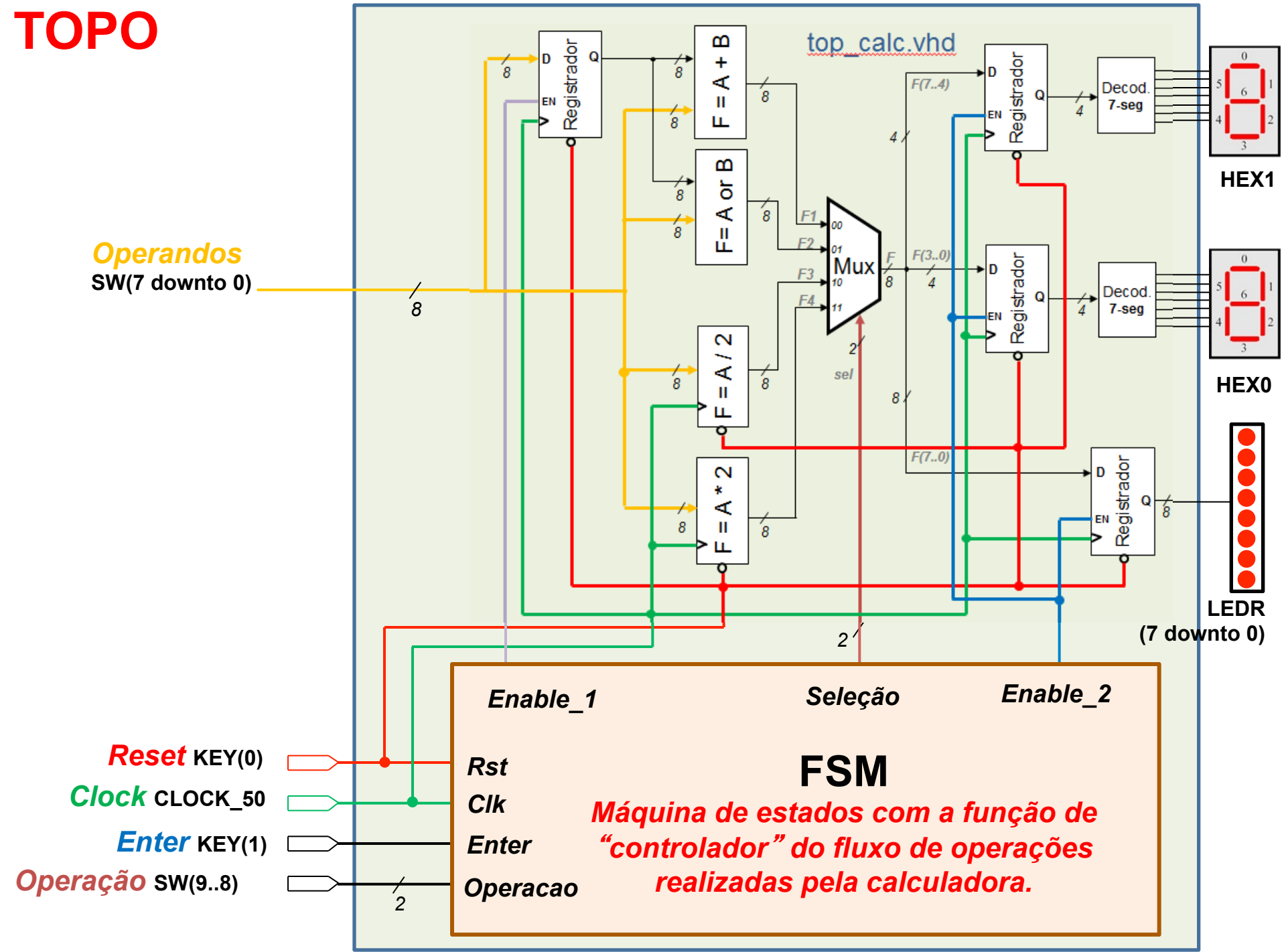
Tarefa – Descrição Textual

- **Alterar a mini-calculadora desenvolvida nas aulas anteriores, de forma a realizar a "multiplicação por 2" e a "divisão por 2".**
- **Utilizando um registrador de deslocamento, projetar um circuito para realizar operações de divisão por 2 (*shift right*).**
- **Utilizando um registrador de deslocamento, projetar um circuito para realizar operações de multiplicação por 2 (*shift left*).**
- **Substituir o componente que realiza a função F = not A pelo novo componente que realiza a multiplicação por 2.**
- **Substituir o componente que realiza a função F = A xor B pelo novo componente que realiza a divisão por 2.**

Atenção!! Nesse novo circuito existem dois componentes que utilizam apenas um operando, e dois componentes que utilizam dois operandos. É preciso alterar a FSM para se adequar a essa situação!!


top_calc.vhd
Operandos
SW(7 downto 0)
Registrador
D
Q
EN
F = A + B
F= A or B
F = A / 2
F = A * 2
F1
F2
F3
F4
00
01
10
11
Mux
sel
F
F(7..4)
F(3..0)
F(7..0)
Decod.
7-seg
HEX1
HEX0
LEDR(7 downto 0)
Enable_1
Clock
Reset
Seleção da operação
Enable_2

TOPO


top_calc.vhd
Operandos
SW(7 downto 0)
Registrador
F = A + B
F= A or B
F = A / 2
F = A * 2
Mux
F1
F2
F3
F4
sel
F(7..4)
F(3..0)
F(7..0)
Decod. 7-seg
HEX1
HEX0
LEDR
(7 downto 0)
Enable_1
Seleção
Enable_2
Reset KEY(0)
Clock CLOCK_50
Enter KEY(1)
Operação SW(9..8)
Rst
Clk
Enter
Operacao
FSM
Máquina de estados com a função de "controlador" do fluxo de operações realizadas pela calculadora.

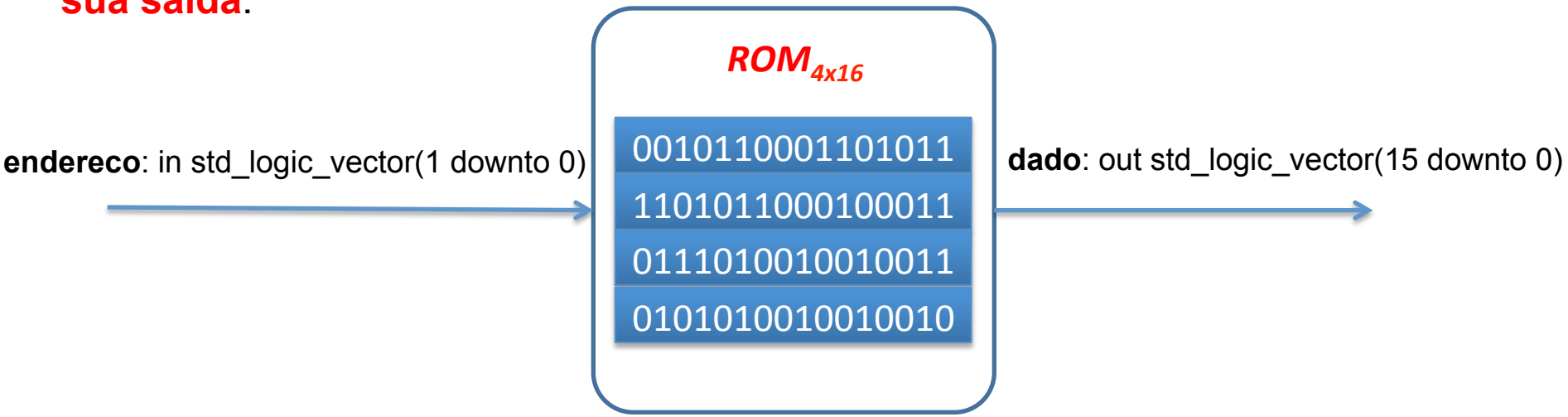
Memória ROM

Organização de uma memória ROM

Assumir:

- Memória com capacidade para armazenar 4 palavras, sendo que cada palavra possui um tamanho de 2 bytes (16 bits).

- Não existe nenhum tipo de controle adicional na leitura dos dados, ou seja, basta fornecer o **endereço** do dado desejado, e a memória deverá fornecer esse **dado na sua saída**.



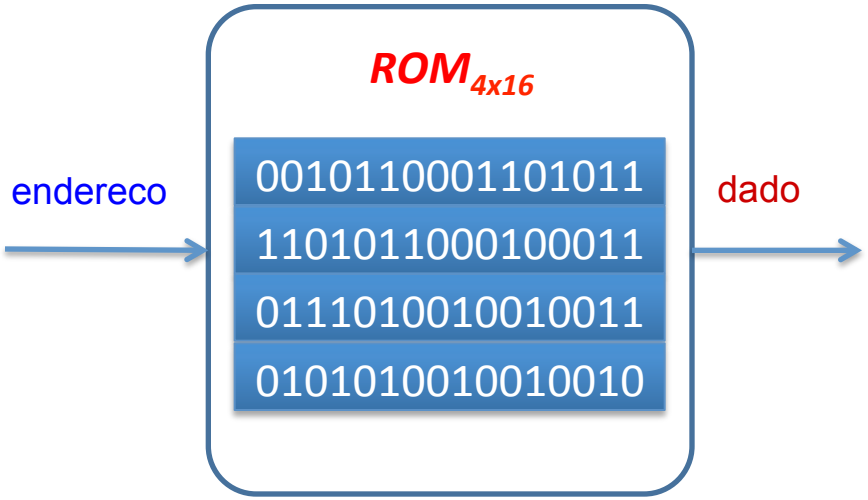
- Na posição (endereço) 0 dessa ROM está armazenado o valor x"2C6B"
- Na posição (endereço) 1 dessa ROM está armazenado o valor x"D623"
- Na posição (endereço) 2 dessa ROM está armazenado o valor x" 7493"
- Na posição (endereço) 3 dessa ROM está armazenado o valor x" A492"

Exemplo de ROM em VHDL obtido no site:
http://www.edaboard.com/thread38052.html

O VHDL original do site foi adaptado para se adequar ao exemplo desses slides. Esse exemplo foi compilado e simulado utilizando a ferramenta on-line:
http://www.edaplayground.com/

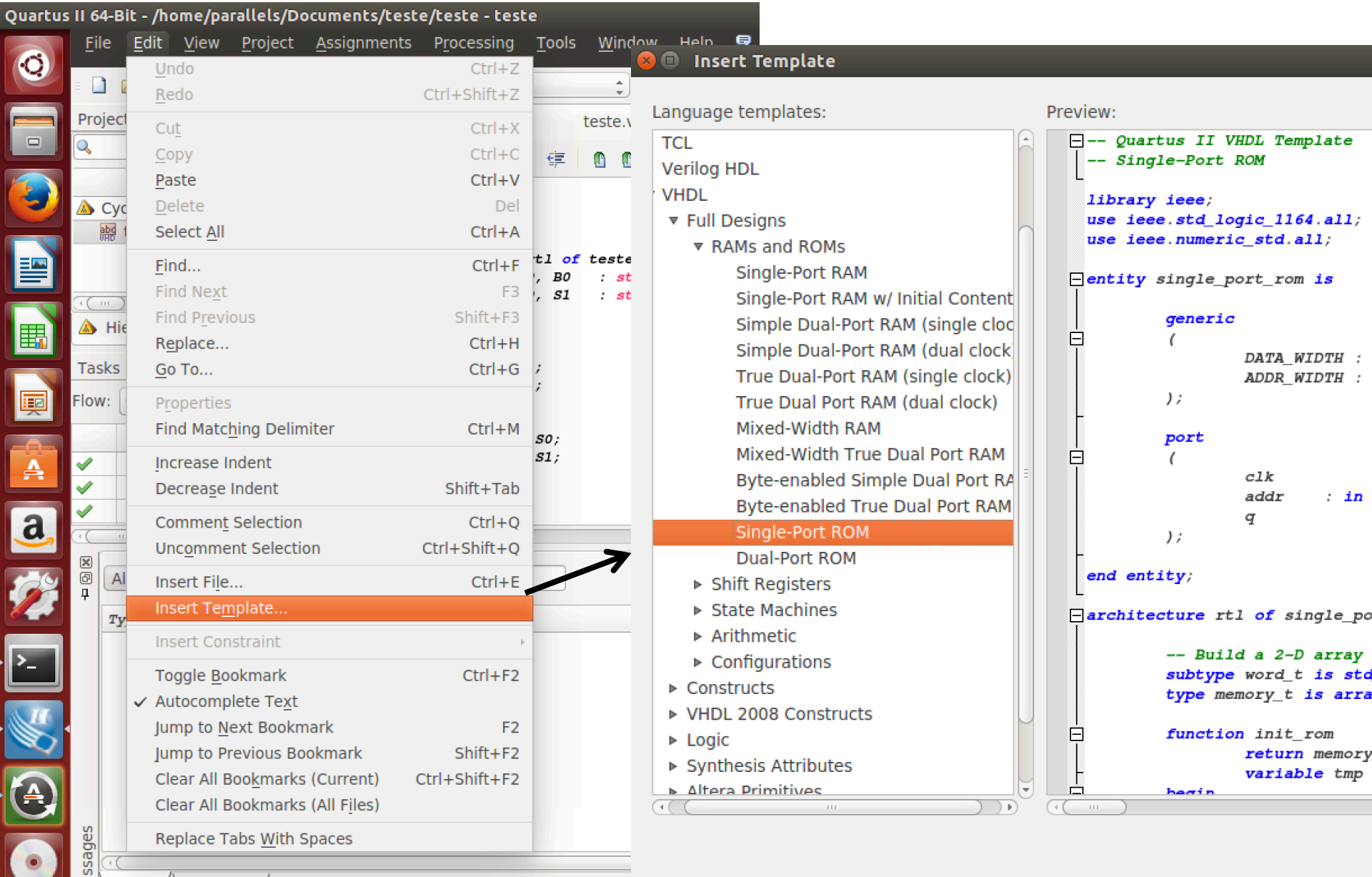
Nao foi testado com o Quartus/ModelSim, logo nao existe garantia de funcionamento, principalmente por ter sido encontrado na Internet.

```
LIBRARY IEEE;
   USE IEEE.STD_LOGIC_1164.ALL;
   USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY ROM IS
  PORT(
            endereco: in std_logic_vector(1 downto 0);
            dado: out std_logic_vector(15 downto 0)
      );
END ENTITY;

ARCHITECTURE BEV OF ROM IS
  type memoria is array ( 0 to 2**2 - 1) of std_logic_vector(15 downto 0);
  constant minhaROM: memoria := (
                    0  => "0010110001101011",
                    1  => "1101011000100011",
                    2  => "0111010010010011",
                    3  => "0101010010010010"
    );

BEGIN
   process (endereco)
   begin
       case endereco is
                          when "00"    => dado <= minhaROM (0);
                          when "01"    => dado <= minhaROM (1);
                          when "10"    => dado <= minhaROM (2);
                          when "11"    => dado <= minhaROM (3);
                         when others => dado <= (others => '0');
       end case;
  end process;
END BEV;
```



Template de ROM do Quartus II

Quartus II – Templates de descrições VHDL com processos



Entity

```
-- Quartus II VHDL Template
-- Single-Port ROM

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity single_port_rom is

	generic
	(
		DATA_WIDTH : natural := 8;
		ADDR_WIDTH : natural := 8
	);

	port
	(
		clk	: in std_logic;
		addr	: in natural range 0 to 2**ADDR_WIDTH - 1;
		q	: out std_logic_vector((DATA_WIDTH -1) downto 0)
	);

end entity;
```

Architecture

```
architecture rtl of single_port_rom is
        -- Build a 2-D array type for the ROM
        subtype word_t is std_logic_vector((DATA_WIDTH-1) downto 0);
        type memory_t is array(2**ADDR_WIDTH-1 downto 0) of word_t;

        function init_rom
                return memory_t is
                variable tmp : memory_t := (others => (others => '0'));
        begin
                for addr_pos in 0 to 2**ADDR_WIDTH - 1 loop
                     -- Initialize each address with the address itself
                    tmp(addr_pos) := std_logic_vector(to_unsigned(addr_pos, DATA_WIDTH));
                end loop;
                return tmp;
        end init_rom;

        -- Declare the ROM signal and specify a default value.
        -- Quartus II will create a memory initialization file (.mif) based on the default value.
        signal rom : memory_t := init_rom;
begin
        process(clk)
        begin
            if(rising_edge(clk)) then
                q <= rom(addr);
            end if;
        end process;
end rtl;
```

Outros templates de descrições em Quartus II

Quartus II – Templates de descrições VHDL com processos