Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Departamento de Engenharia de Computação e Automação
DCA0212.2 - Circuitos Digitais
Docente: Tiago Barros

Laboratório 8 - Bancos de registradores e memórias

Objetivos

1. Experimentar a descrição em VHDL de bancos de registradores e memórias (RAM e ROM);
2. Reforçar os conceitos de latches, flip-flops e registradores;
3. Pôr em prática conceitos aprendidos na disciplina Circuitos Digitais - Teoria.

Introdução

Na aula de hoje serão apresentados os circuitos denominados bancos de registradores, os quais são componentes de memória utilizados em blocos operacionais de circuitos digitais.

Bancos de Registradores:

Um banco de registradores (ou registrador de arquivos) ***M x N*** é um componente de memória de blocos operacionais (*datapath*) de um circuito digital que propicia um acesso eficiente a um conjunto de *M* registradores, de forma que cada registrador possui largura de *N* bits. A Figura 1 ilustra o símbolo para diagrama de blocos de um banco de registradores.

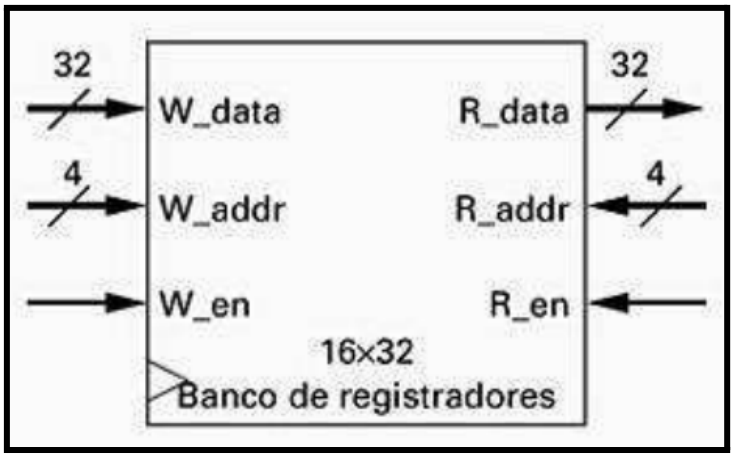

Figura 1: Símbolo para diagrama de blocos de um banco de registradores 16x32.

Para se escrever um valor no banco de registradores, coloca-se o dado a ser escrito em *W_data*. Para se indicar em qual dos registradores do banco de registradores o dado será escrito, coloca-se o endereço do registrador na entrada *W_addr*. Para habilitar uma escrita, em sincronismo com o *clock*, coloca-se *1* em *W_en*. O conjunto de entradas *W_data*, *W_addr* e *W_en* é conhecido como portas de escrita do banco de registradores.

No processo de leitura, especifica-se o endereço do registrador a ser lido na entrada *R_addr* e habilita-se a leitura fazendo *R_en = 1*. Esses valores farão com que o banco de registradores coloque na saída *R_data* o conteúdo do registrador que foi endereçado. O conjunto de *R_addr*, *R_en* e *R_data* é conhecido como portas de leitura de um banco de registradores. As portas de leitura e escrita são independentes entre si, podendo-se escrever em um registrador e ler de outro (ou do mesmo) registrador simultaneamente. A Figura 2 ilustra o projeto interno de um banco de registradores 4x32.

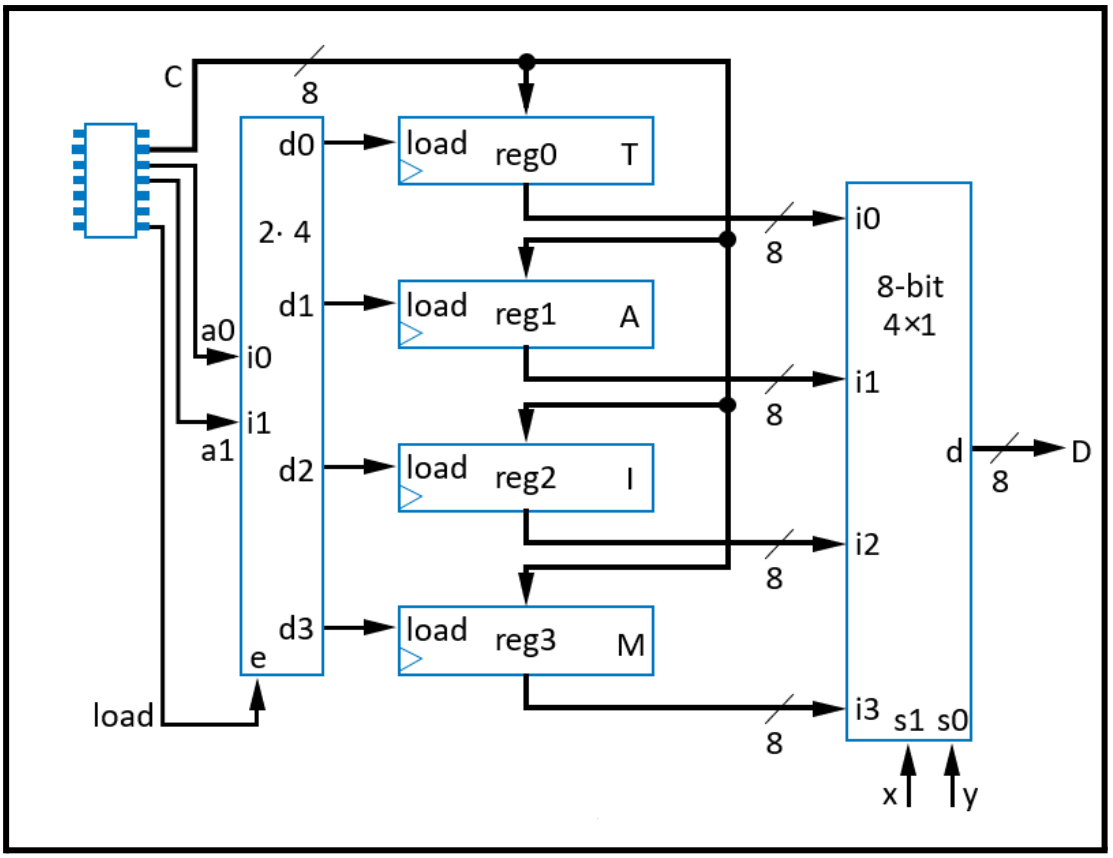

Figura 2: Projeto interno de um possível banco de registradores 4x8.

No projeto interno do banco de registradores da Figura 2, tem-se decodificadores de endereços de escrita e leitura; registradores formados por *flip-flops* e um multiplexador.

Memória ROM:

A memória ROM (Read-Only Memory) é um tipo de memória que só pode ser lida, mas não escrita. Portanto, no momento do projeto do circuito, o projetista deve escolher quais palavras devem ser gravadas permanentemente no circuito. Existem várias vantagens para se escolher esse tipo de memória: compacticidade, rapidez, baixa potência e não volatilidade (os dados não são apagados quando o circuito não é alimentado).

Como mostrado na Figura 3, a ROM só necessita de 4 portas: a entrada de endereço, a entrada *enable*, a entrada de clock e a saída de dados.

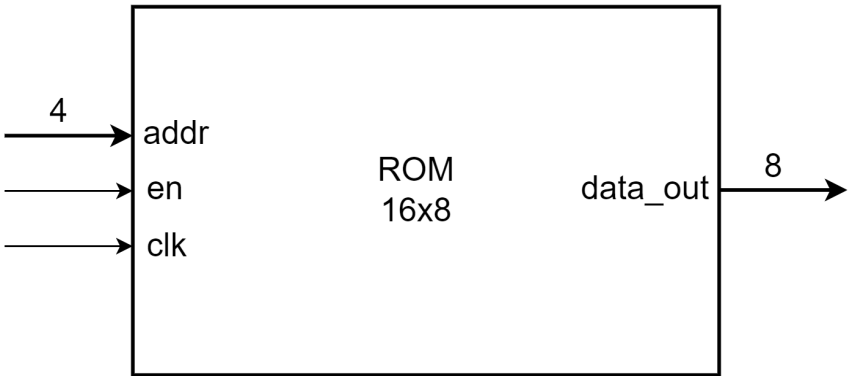

Figura 3: Entradas e saídas de uma ROM com 16 endereços de 8 bits.

Um exemplo de código VHDL com descrição comportamental de uma ROM 16x8 (16 endereços de 8 bits) é mostrado a seguir, no Quadro 1.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY ROM16x8 IS
PORT(
        clock : IN STD_LOGIC;
        rom_enable : IN STD_LOGIC;
        address : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        data_output : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
);
END ROM16x8;
ARCHITECTURE behav OF ROM16x8 IS
    TYPE rom_type IS ARRAY(0 to 15) OF STD_LOGIC_VECTOR(7 DOWNTO 0);

    CONSTANT mem: rom_type :=
        (2 => “01011001”, -- Aloca um dado no endereço 2
         3 => “00000100”, -- Aloca um dado no endereço 3
         4 => “00100101”, -- Aloca um dado no endereço 4
```

```
        others => "00000000"  -- Aloca 0 no outros endereços
        );

BEGIN

PROCESS(clock) IS
BEGIN
   IF (RISING_EDGE(clock) AND rom_enable = '1') THEN
        data_output <= mem(conv_integer(unsigned(address)));
   END IF;
END PROCESS;
END behav;
```

Quadro 1: Código exemplo em VHDL com descrição comportamental de uma memória ROM 16x8.

Na Figura 4 vemos a forma de onda da implementação da ROM de 16x8, quando rom_enable = '1', a saída da ROM é o valor colocado no endereço selecionado pela porta address.

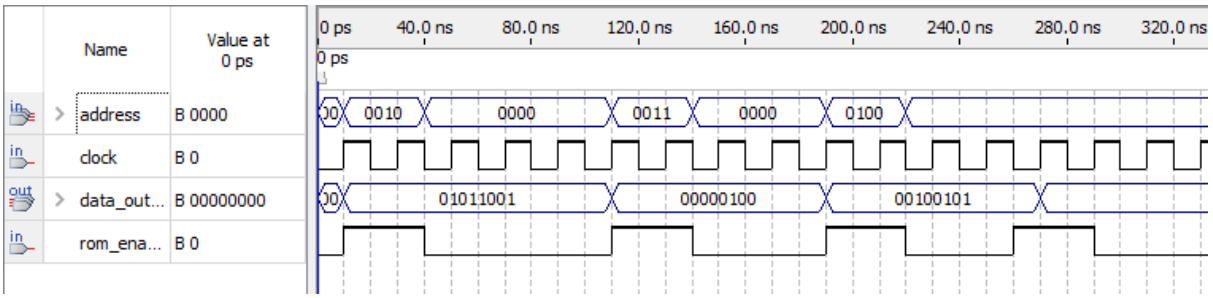

Figura 4: Exemplo da Waveform da ROM 16x8

Memória RAM:

Já a memória RAM (*Random Access Memory*) é um tipo de memória que pode ser lida e escrita. Portanto, além das entradas e saídas de uma ROM, a RAM tem mais duas entradas, a de dados e um bit para habilitar a escrita. É possível combinar o bit de escrita e o de leitura de modo que '0' signifique leitura e '1' escrita de dados. Na Figura 5 podemos ver o diagrama de uma RAM 16x8 e no Quadro 2 um exemplo de descrição VHDL desta memória.

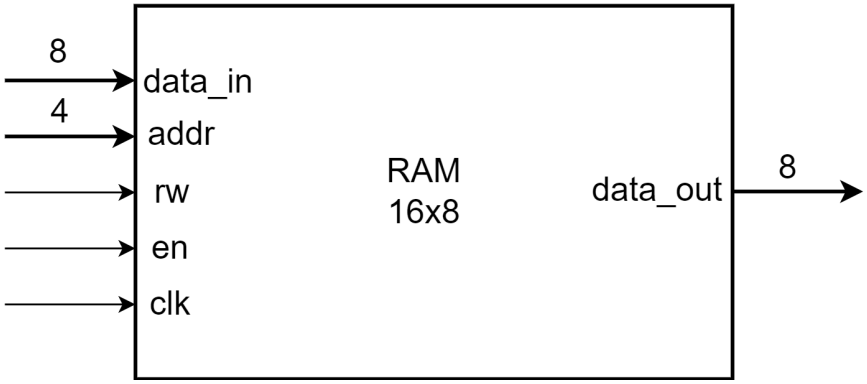

Figura 5: RAM 16x8.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY RAM16x8 IS
PORT(
        clock : IN STD_LOGIC;
        rw_enable : IN STD_LOGIC;
        mem_enable : IN STD_LOGIC;
        address : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        data_input : IN STD_LOGIC_VECTOR(7 DOWNTO 0)
        data_output : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
);
END RAM16x8;
ARCHITECTURE behav OF RAM16x8 IS
    TYPE ram_type IS ARRAY(0 to 15) OF STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL ram: ram_type;
    SIGNAL temp_address: STD_LOGIC_VECTOR(3 DOWNTO 0);

BEGIN

PROCESS(clock) IS
BEGIN
   IF (RISING_EDGE(clock) AND mem_enable = '1') THEN
        IF (rw_enable = '0') THEN
            temp_address <= address;
        ELSIF (rw_enable = '1') THEN
            ram(conv_integer(unsigned(address))) <= data_input;
        END IF;
        data_output <= ram(conv_integer(unsigned(temp_address)));
   END IF;
END PROCESS;
END behav;
```

Quadro 2: Código com descrição VHDL comportamental de uma RAM 16x8.

As Figuras 6 e 7 mostram um exemplo dos sinais da RAM de 16x8. quando mem_enable e rw_enable são iguais a ‘1’ o valor em data_input é gravado no endereço address. Posteriormente, quando mem_enable é ‘1’ e rw_enable é ‘0’ os valores são lidos nos endereços.

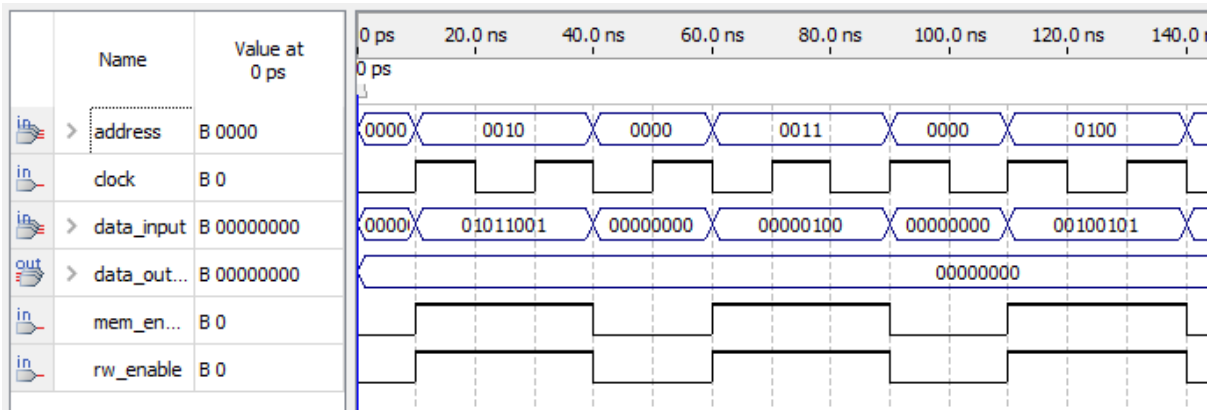

Figura 6: Exemplo de Waveform da RAM 16x8 quando rw_enable é igual a ‘1’. ou seja, no modo de escrita da RAM

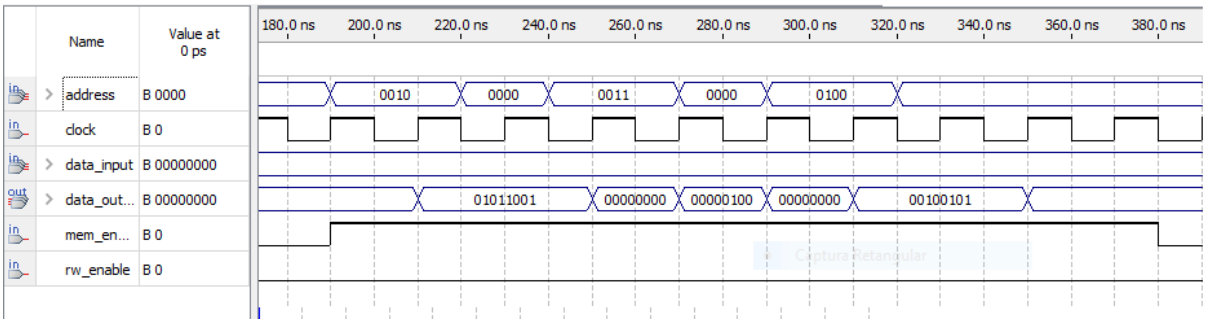

Figura 7: Exemplo de Waveform da RAM 16x8 quando rw_enable é igual a ‘0’, ou seja, no modo de leitura da RAM

Comando GENERATE:

O comando concorrente GENERATE utiliza dois esquemas de iteração para repetir comandos concorrentes:

- Esquema FOR;
- Esquema IF;

O esquema FOR repete um conjunto de comandos uma quantidade determinada de vezes. É fornecida uma variável local e os limites para esta variável.

A sintaxe de declaração do esquema FOR é como ilustrado no Quadro 3.

```
<rótulo_obrigatório>: FOR <variável_local> IN <limites_da_variável> GENERATE
   --Comandosconcorrentes
END GENERATE <rótulo_opcional>;
```

Quadro 3 - Esquema FOR/GENERATE.

Utilizando genéricos em conjunto com o comando FOR/GENERATE é possível, por exemplo, descrever facilmente a porta AND com quantidade de entradas variável, conforme ilustrado no Quadro 4.

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

ENTITY AND_n_entradas IS
   GENERIC(n : NATURAL := 4); -- Número de pinos de entrada
   PORT(E : IN  STD_LOGIC_VECTOR(n-1 DOWNTO 0);
            S: OUT STD_LOGIC);
END AND_n_entradas;

ARCHITECTURE arquitetura OF AND_n_entradas IS
   --Sinal intermediário para receber os resultados parciais
   SIGNAL Intermed: STD_LOGIC_VECTOR(n-1 DOWNTO 0);

BEGIN
   Intermed(0) <=E(0); --Primeiro pino

   and_for: FOR i IN 1 TO n-1 GENERATE --Cria (n-1) portas AND
      Intermed(i) <=Intermed(i-1) AND E(i);
   END GENERATE;
   S <=Intermed(n-1); --Saída

END arquitetura;
```

Quadro 4 - Porta And de N entradas.

É necessário atribuir separadamente os valores Intermed(0) e S através das linhas:

```
Intermed(0) <=E(0); --Primeiro pino
      S <=Intermed(n-1); --Saída
```

Isto é devido ao fatos de que as operações para i = 0 e i = n-1 não seguem a mesma regra de geração, pois não existe **Intermed(-1)** e **Intermed(n-1)** é desnecessário. Na Figura 8 podemos ver a ilustração dos circuitos descrito e sintetizado.

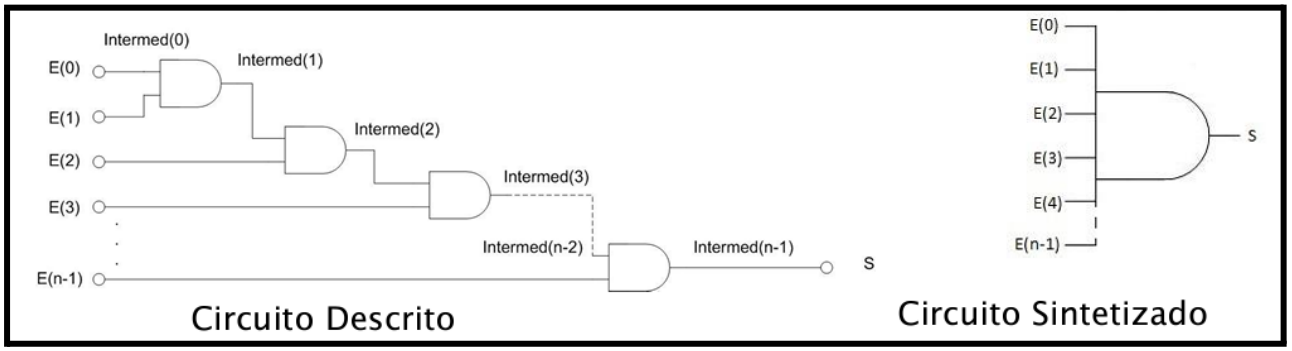

Figura 8: Circuito descrito e circuito sintetizado para a porta AND de N entradas.

Atividades:

Entregar um relatório descrevendo a execução das tarefas 1 e 2. O relatório deve conter todos os códigos VHDL, formas de onda das simulações e explicações dos resultados verificados nas formas de onda.

1) **Bancos de registradores:**
   - **a)** Implemente um banco de registradores 4x8, baseado na Figura 2. Siga as seguintes instruções:
     - **i)** Utilize *flip-flops* para formar os registradores.
     - **ii)** Na simulação, realize operações de escrita e leitura no banco de registradores.
2) **Memórias:**
   - **a)** Implemente um circuito capaz de ler valores de uma memória ROM e armazenar estes valores em uma memória RAM. A memória ROM deverá ter armazenados os valores entre 0 e 15 nos endereços de 0 a 15. Os valores lidos na memória ROM serão escritos em ordem reversa na RAM, ou seja, o valor 0 deve ser armazenado no endereço 15 da memória RAM, o valor 1 no endereço 14 e assim por diante. Siga as instruções:
     - **i)** Utilize o esquema FOR/GENERATE para armazenar as saídas da memória ROM nos endereços corretos da memória RAM.
     - **ii)** Utilize componentes para implementar o contador e a memória RAM.
     - **iii)** Simule operações de leitura na memória RAM.