

LSD → Resumos para o exame teórico.

Modelação de Memórias → 1 porto e multiporto

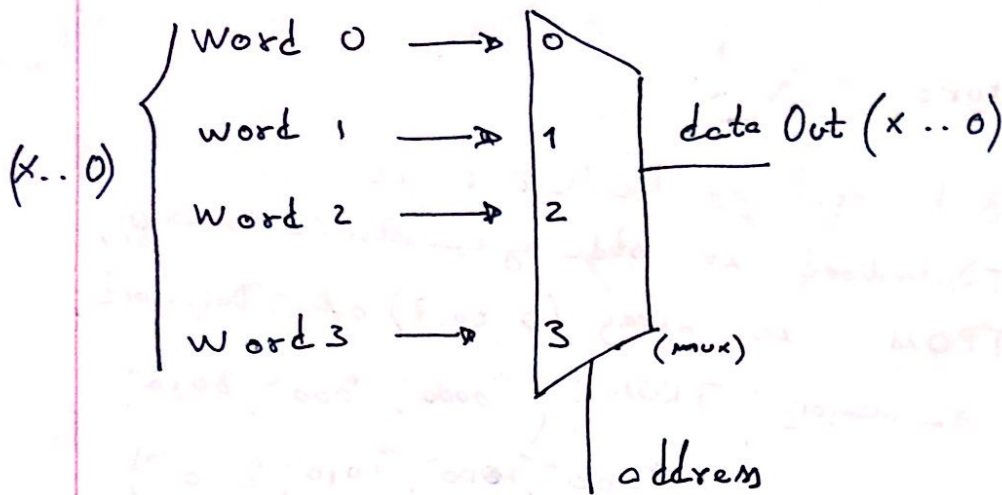
ROM (Read Only Memory)

RAM (Random Access Memory)

pod. ser síncrona ou assíncrona

geralmente
assíncrona

ROM



- Nada mais é que 1 array que guarda as entradas dos portos e passa para a saída o que queremos ler com base num address.

(Syntax) - Definição de um array em

VHDL:

type type_name is array (range) of element;

Ex:

type Memory is array (0 to 3) of std_logic_vector;

Modelação de uma ROM em VHDL

-- Entidade

entity ROM_8x4 is

```
port (address : in std_logic_vector (2 downto 0);  
      dataOut : out std_logic_vector (3 downto 0)  
      );
```

end ROM_8x4 ;

-- Arquitetura

architecture Behavioral of ROM_8x4 is

```
subtype TDataWord is std_logic_vector (3 downto 0);
```

```
type TROM is array (0 to 7) of TDataWord;
```

```
constant c_memory : TROM : ("0000", "0001", "0010",  
                              "0100", "1000", "1010", "0101");
```

begin



```
dataOut <= c_memory(to_integer(unsigned(address)));
```

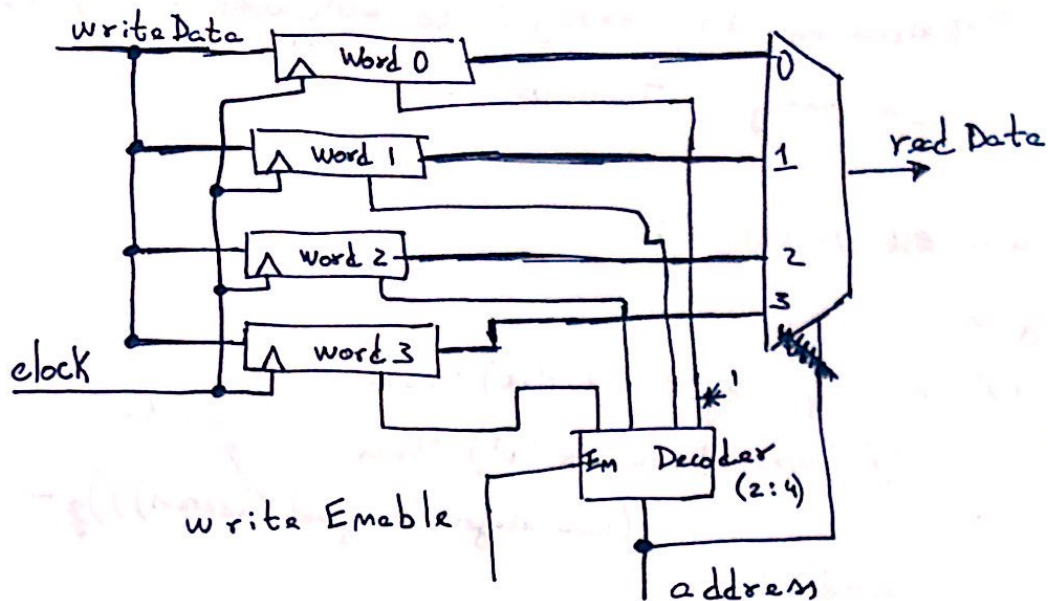
end Behavioral;

Nota:

↳ Poderíamos ligar esta ROM a um contador e ligá-lo ao address para que a cada contagem fossem lidos os valores da memória.

RAM

- Estática (SRAM) 
- Dinâmica (DRAM) 



*'s: mais
que dão ordem
à escrita

Código VHDL básico de uma RAM

```
-- Entidade
entity RAM is
    port (writeCLK: in std_logic;
          -- Enable: in std_logic;
          -- Address: in std_logic_vector(4 downto 0);
          -- Data: in std_logic_vector(7 downto 0);
          readCLK: in in std_logic;
          -- Address: in std_logic_vector(4 downto 0);
          -- Data: out std_logic_vector(7 downto 0);
    );
end RAM;
```


-- Arquitetura

* Nota que este código é diferente do que uma vez que tem leitura síncrona.

architecture Behavioral of RAM is

```
constant NUM_WORDS : integer := 32;
subtype TDataWord is std_logic_vector (7 downto 0);
type TMemory is array (0 to NUM_WORDS - 1) of TDataWord;
signal s_memory : TMemory.
```

begin

process (writeClk)

begin

if rising_edge(writeClk) then

if (writeEnable = '1') then

s_memory(to_integer(unsigned('address'))) <= writeData;

end if;

end process;

process (readClk)

begin

if rising_edge(readClk) then

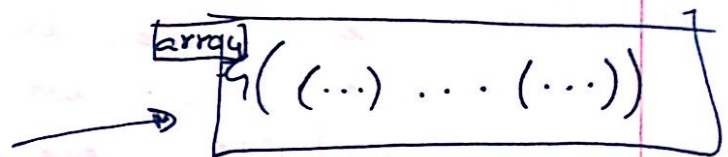
readData <= s_memory(to_integer(unsigned(readAddress)));

end if;

end process;

end Behavioral;

Para parametrizar:



parametrizar o address

```
generic (addrBusSize : integer := 4;
         dataBusSize : integer := 8);
```

parametrizar os dados

Na arquitetura: $NUM_WORDS = 2^{addrBusSize}$
 $TDataWords (dataBusSize - 1) \text{ downto } 1$