

Universidade Estadual de Campinas

LABORATÓRIO 04 – Uso da memória RAM com LPM

Isabela Caroline de Sousa 218071

José Victor Machuca 219118

Matheus Santos Sano 222370

Para esse laboratório utilizamos a memória RAM M9K 32 x 8 bits com um único clock, contida no IP Catalog. Importada a memória, criamos um arquivo contendo dados, no formato hexadecimal, a serem armazenados nas 32 posições da memória, de modo a inicializá-la. A seguir são mostrados os dados que inserimos na memória:

:010000000FF :0100010001FD :0100020002FB :0100030003F9 :0100040004F7 :0100050005F5 :0100060006F3 :0100070007F1 :0100080008EF :0100090009ED :01000A000AEB :01000B000BE9 :01000C000CE7 :01000D000DE5 :01000E000EE3 :01000F000FE1 :0100100010DF :0100110011DD :0100120012DB :0100130013D9 :0100140014D7 :0100150015D5 :0100160016D3 :0100170017D1 :0100180018CF :0100190019CD :01001A001ACB :01001B001BC9 :01001C001CC7 :01001D001DC5 :01001E001EC3 :01001F001FC1 :0000001FF

Para verificar o funcionamento da memória, implementamos um testbench. O primeiro teste consiste em fazer a leitura de todas as posições inicializadas, validando se estão corretas, em seguida fizemos um segundo teste que consiste em reescrever os conteúdos da memória, de modo que os valores das últimas posições sejam inseridos nas primeiras posições e vice-versa, e reler os novos valores.

A seguir, o código do testbench na íntegra:

```
library ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;
 1
2
3
4
5
6
7
         entity test_memoria is end;
  8
       □architecture dut of test_memoria is
10
              CONSTANT half_period : time := 10 ps;
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
30
31
32
33
34
35
              COMPONENT ramlpm IS
       PORT
                         address
                                         : IN STD_LOGIC_VECTOR(4 downto 0);
: IN STD_LOGIC;
: IN STD_LOGIC_VECTOR(7 downto 0);
: IN STD_LOGIC;
: OUT STD_LOGIC_VECTOR(7 downto 0)
                         clock
                         data
                         wren
                         q
              END COMPONENT;
              signal clk, simulando : STD_LOGIC := '0';
signal wren : STD_LOGIC := '0';
signal address : STD_LOGIC_VECTOR(4 downto 0);
signal data, q : STD_LOGIC_VECTOR(7 downto 0);
         BEGIN
              clk <= (simulando and not clk) after half_period;
              DUT: ramlpm port map (address, clk, data, wren, q);
              st: process is
variable Q_RTL : integer := 0;
36
37
38
39
40
                   assert false report "BOT" severity note; --inicio do teste simulando <= '1'; -- simulacao começou wait until rising_edge(clk);
412444454467489555555555555666234566677777777777777777777788812384
                    -- TESTE LEITURA
                   wren <= '0';
                   FOR I IN 0 TO 31 LOOP
   wait until falling_edge(clk);
                         address <= std_logic_vector(to_unsigned(I,5));</pre>
                         Q_RTL := I;
                         wait until rising_edge(clk);
wait until falling_edge(clk);
                         wait until rising_edge(clk);
assert Q_RTL = to_integer(signed(q)) report "Erro na leitura" severity failure;
                   END LOOP;
                    -- TESTE ESCRITA
                   wait until rising_edge(clk);
                   wren <= '1';
                   wait until falling_edge(clk);
                   FOR I IN 0 TO 31 LOOP
  wait until falling_edge(clk);
                         address <= std_logic_vector(to_unsigned(I,5));
data <= std_logic_vector(to_unsigned(31-I,8));
Q_RTL := 31-I;</pre>
                         wait until rising_edge(clk);
wait until falling_edge(clk);
                         wait until rising_edge(clk);
assert Q_RTL = to_integer(signed(q)) report "Erro na escrita" severity failure;
                   END LOOP;
                   assert false report "EOT" severity note; --fim do teste simulando <= '0'; --simulação parou
85
86
87
                   wait:
88
              END process;
        END DUT;
89
```

ERROS E CORREÇÕES

Um dos erros neste laboratório foi na sintaxe do registro de fim de arquivo, ou seja, no último dos registros de dados que foram inseridos na memória (:00000001FF). O erro era bem simples, foi apenas a existência de um zero a mais na instrução.

O outro erro foi no código do testbench no momento de inverter os valores das últimas posições para as primeiras. Na execução do código, o testbench não fazia a inversão de forma adequada. Sendo assim, era apresentado o erro "Erro na escrita" no terminal. Para solucionar esse problema invertemos os dados dentro do loop (linha 69) pela subtração "31 - I", sendo I identificando o endereço e a subtração 31 - I o dado. Dessa forma a lógica de inverter os dados funcionou corretamente.

RESULTADOS

Após corrigir todos os erros apresentados, a memória RAM e seu testbench funcionaram corretamente. Esse fato pode ser verificado pela onda a seguir e pela conclusão do testbench sem indicar qualquer erro.

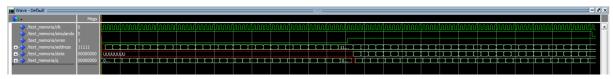


Figura 01: Gráfico da simulação da memória RAM.

```
# Time: 10 ps Iteration: 1 Instance: /test_memoria/DUT/altsyncram_component
# ** Note: EOT
# Time: 2610 ps Iteration: 0 Instance: /test_memoria
```

Figura 02: A informação "EOT" é apresentada no terminal, mostrando que o testbench foi concluído sem apresentar erro na memória RAM.