CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
DEPARTAMENTO DE COMPUTAÇÃO

ALUNOS:

VITOR BRANDÃO RAPOSO

HENRIQUE VIANA GARCIA ALVES

LABORATÓRIO DE ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES II

Prática 01

BELO HORIZONTE - MG

2022

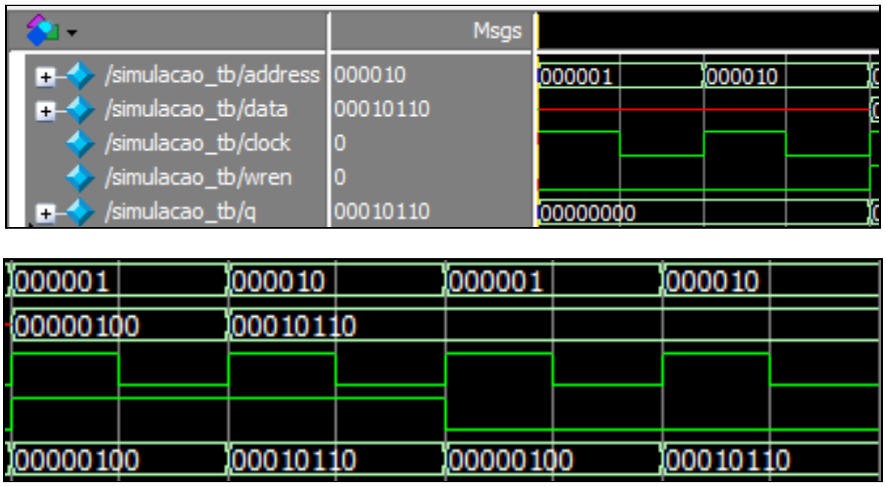
1 PARTE

Seguindo o primeiro roteiro de laboratório, testamos o funcionamento da memória RAM implementada pela biblioteca LPM.

A memória funciona de modo que *address* é o endereço de memória, *data* os dados inseridos e *wren* habilita a escrita ao nível alto e leitura ao nível baixo.

Para simular o funcionamento de nosso módulo, realizamos as seguintes ações:

- Lemos endereços vazios para demonstrar a ausência de valores,
- Escrevemos os números de chamada dos integrantes do grupo nos mesmos,
- Lemos os mesmos endereços para concluir a presença de tais valores nos endereços. Assim, obtivemos os seguintes resultados:



Figuras 1 e 2 – Simulação realizada pelo testbench *simulacao_tb.v*

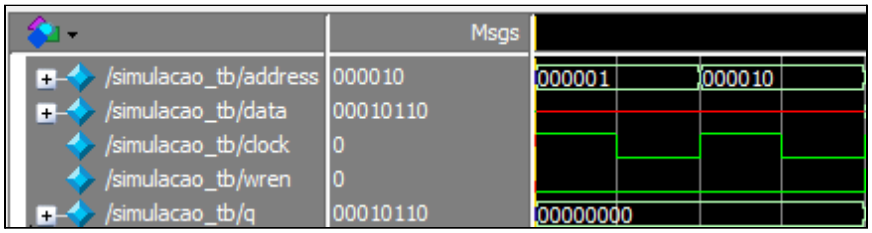
2 PARTE

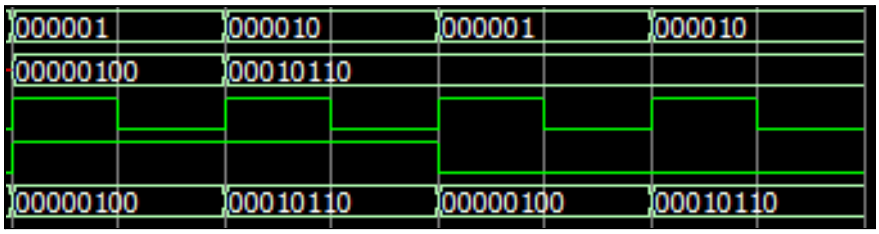
A segunda parte do trabalho foca em inicializar a memória utilizando um arquivo .mif.

```
WIDTH=8;
DEPTH=32;

ADDRESS_RADIX=UNS
DATA_RADIX=UNS;

CONTENT BEGIN
        0    :    4;
        1    :   22;
        2    :   23;
        3    :   24;
        4    :   25;
        5    :   26;
        6    :   27;
        7    :   28;
        8    :   29;
        9    :   30;
        10   :   31;
        11   :   32;
        12   :   33;
```

Figura 3 – Trecho do arquivo ramlpm.mif utilizado

Para demonstrá-la, foram inicializados 32 valores a partir de um arquivo .mif (memory initialization file) conforme exigido no roteiro, sendo as duas primeiras posições do arquivo sendo equivalentes ao nosso número de chamada. A partir disso, realizou- se a leitura de diversos endereços diferentes da memória para avaliar a presença dos valores. Assim, obtivemos os seguintes resultados da simulação:





Figuras 4 e 5 – Simulação realizada pelo testbench *simulacao_tb.v*

3 PARTE

Na parte foi requisitado a criação de uma memória cache associativa de duas vias, onde se demonstraria este processo de movimentação de dados entre a memória principal e o cache, além dos estados de todos os bits envolvidos ao longo dos testes.

Infelizmente não conseguimos terminar o código a tempo da apresentação e simulação, porém escrevemos bastantes lógicas para implementar o cache. Nosso raciocínio envolveu um for-loop que percorreria o cache nos locais requisitados (2 linhas para cada set) e realizaria uma busca da tag repassada pelo usuário, retornando hit ou miss e alterando os bits envolvidos como o dirty, valid e lru.

Nossa lógica dos for's implementados no código dessa parte 3 segue a lógica do diagrama de fluxos a seguir:

Próxima página!

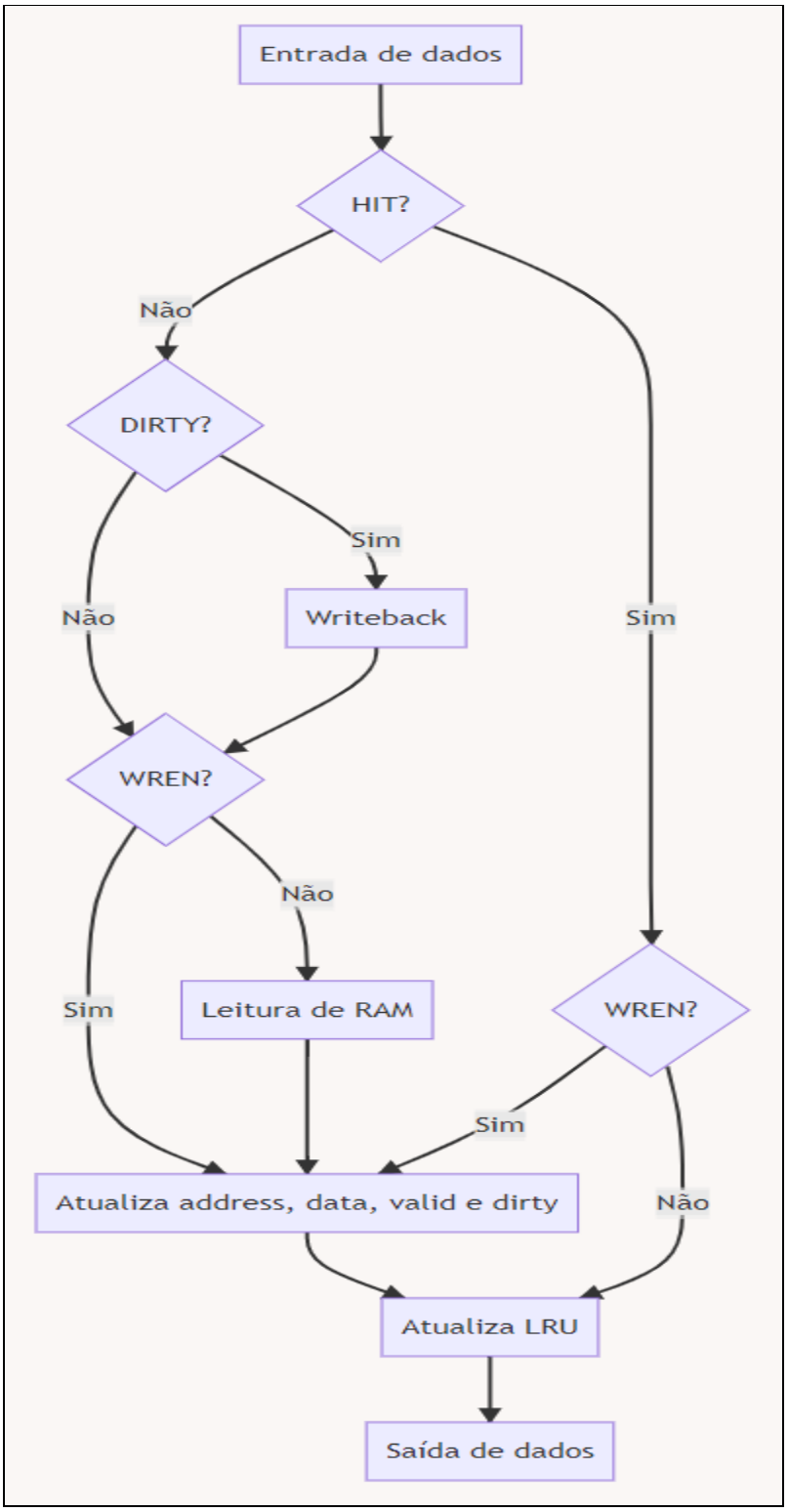


Figura 6 – Diagrama criado para demonstrar o fluxo de lógica do cache

Próxima página!

4 Conclusão

A partir da execução das práticas pudemos aprender novos conceitos e conhecimentos relacionados às bibliotecas do Quartus e à linguagem Verilog, além de poder assimilar e aplicar o conteúdo teórico visto em sala de aula.