CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

TRABALHO PRÁTICO I - PARTES I E II

Hierarquia de memória

Arthur Estevão de Souza Machado
Marcelo Lopes de Macedo Ferreira Cândido

Engenharia de Computação
Laboratório de Arquitetura e Organização de Computadores II

17 de agosto de 2018

INTRODUÇÃO

A memória computacional é a forma pela qual os computadores podem armazenar e disponibilizar seus dados para processamento. Ela se manifesta em modelos cujas diferenças principais residem na rapidez para acesso, capacidade de armazenamento e preço por *byte*. Devido a essas diferenças e para se criar a ilusão de memória rápida e ilimitada ao programador (a ser explicada na parte III), dividiu-se os tipos de memória em uma **hierarquia**.

O preço por *byte* dos tipos de armazenamento cresce proporcionalmente a velocidade de acesso deles. Ou seja, quanto mais rápido, mais caro. Consequentemente, quanto mais caro, menor a capacidade de armazenamento utilizada para aquele tipo. A hierarquia de memória se organiza da seguinte forma:

- registrador;
- cache;
- memória principal;
- disco rígido;
- fita magnética

sendo o topo mais rápido, caro e menor em armazenamento e a base mais lenta, barata e maior em armazenamento [2].

Essa prática visa o estudo dos tipos de memória. Nas partes I e II da mesma, veremos como criar e simular uma memória RAM de uma porta através da LPM (Biblioteca de Módulos Parametrizados) do programa Quartus II (v. 13.0.SP1) e inicializá-la com a ajuda de um arquivo.

DETALHES DA IMPLEMENTAÇÃO

A implementação do projeto se deu, inicialmente, através do plug-in Mega Wizard[1], a fim de gerar um módulo LPM (Biblioteca de Módulos Parametrizados). Esse seria capaz de emular uma memória de uma porta em simulações para análise ou em uma placa Altera FPGA DE2 Cyclone II como as encontradas no laboratório [1].

As decisões sobre o número de *bits* utilizados para endereço e dado fornecido seguiram a estética. No entanto, para simplificação durante o teste, foram utilizados apenas quatro dos

[1] Ferramenta presente no Quartus 13.0.SP1.

cinco *bits* para endereço e quatro dos oito de dados enviados e retornados da placa. Isso se deu porque o *display* de sete segmentos que deve apresentar os dados manipulados, é capaz de representar números de zero a F, em hexadecimal.

Com isso, para a utilização do *display*, foi necessário um decodificador capaz de representar os dados na base numérica citada acima. Ademais, cada arquivo apresenta dois módulos principais[2], um para simulação e outro para teste na placa.

FUNCIONAMENTO

Através da LPM, obteve-se um módulo que possui como entrada: *address* (endereço, posição para leitura ou escrita), *wren* (habilita escrita, que decide se no *clock* atual se lerá ou escreverá sobre a memória no endereço indicado), *data* (dado a ser escrito) e *clock* . Já a saída (porta *q*), na leitura, exibe o dado recolhido da memória na posição pedida. Na escrita, a porta exibe o dado que deve ser escrito.

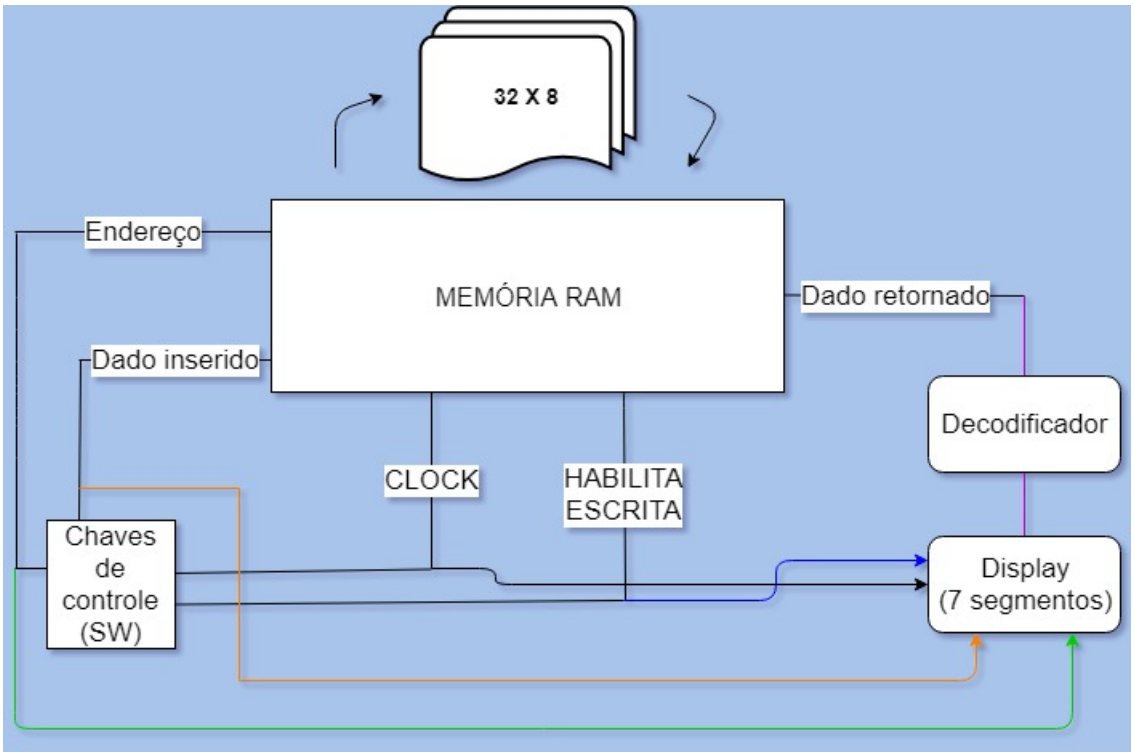

Figura 1: Diagrama esquemático

[2]Quando um está sendo utilizado, o outro deve estar em forma de comentário.Isso pode ser facilmente realizado selecionando o módulo desejado e utilizando a função para comentar ou descomentar dentro do Quartus.

A Figura 1 ilustra o funcionamento da memória na placa, mostrando suas portas da memória e as suas conexões com os elementos - as chaves SW e os *displays* de sete segmentos (juntamente com o decodificador que traduz o binário para a exibição no *display*) - da placa.

Parte I e Parte II

O teste do módulo LPM foi feito de duas formas:

1. Através da simulação do bloco de memória no ModelSim 10.1.d, que será apresentada na seção .

2. Utilizando a FPGA juntamente com seu *display* de sete segmentos, onde cada *display* apresentou informações enviadas e recebidas da placa, como apresentado na imagem a seguir.

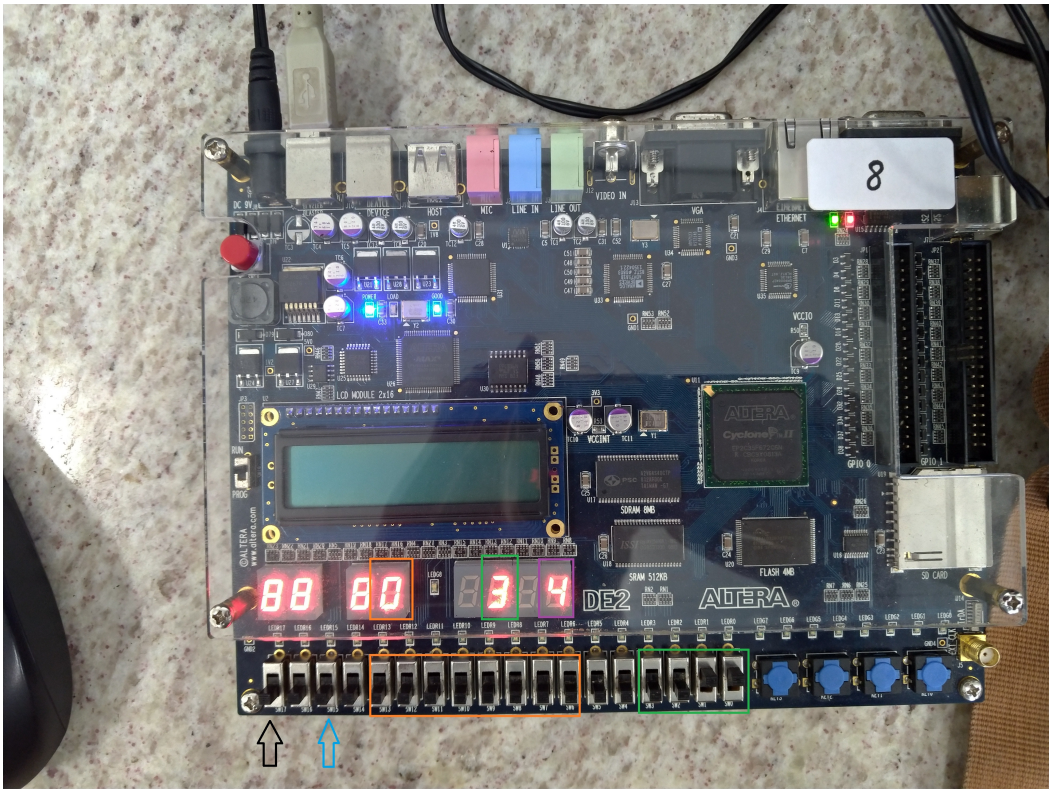

Figura 2: Exemplo de utilização na placa. As cores utilizadas seguem o diagrama esquemático da Figura 1

A Figura 2 apresenta um exemplo de utilização da placa na parte 2 e cada destaque representa uma utilização diferente. Os retângulos laranjas mostram a entrada de dados

para a memória, em que cada chave SW irá provocar uma alteração no *display*. O mesmo serve para os retângulos verdes, que apresentam a posição de memória a ser lida ou escrita. O destaque em lilás mostra o retorno dado pela memória enquanto as setas preta e azul representam *clock* e habilita escrita respectivamente.

Ambas as partes foram testadas realizando escritas e leituras sucessivas em diferentes posições de memória e observando a integridade dos dados apresentados. A parte II possui uma peculiaridade, a utilização de um arquivo com extensão ".mif" para carregamento prévio das posições da memória.

SIMULAÇÕES

Todas a simulações dos blocos obtidos foram realizadas no ModelSim 10.1.d, onde cada entrada teve seu valor forçado a cada pulso de clock de período igual a 100 ps.

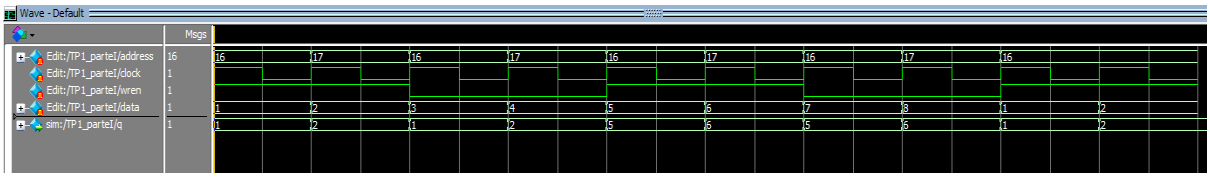

Figura 3: Leituras e escritas via simulação no ModelSim - Parte I

A captura de tela apresentada referente a parte um mostra escritas e leituras sucessivas nas posições de memória 16 e 17. Inicialmente preenchendo tais posições com os dados 1 e 2 respectivamente para em seguida verificar o conteúdo de cada posição com o habilita escrita desligado e se o bloco de memória responde aos comandos corretamente. Todos os dados se apresentaram coesos com o procedimento proposto.

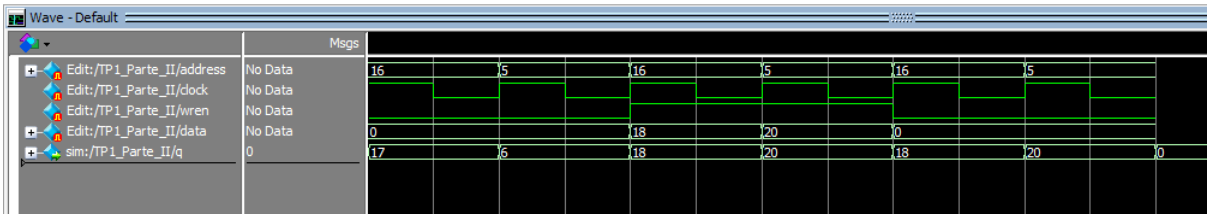

Figura 4: Leituras e escritas via simulação no ModelSim - Parte II

De fato, o procedimento encaminhado à parte dois do projeto é idêntico ao da parte 1, mas partindo do princípio de que a memória já deve estar carregada previamente. Com isso, foram lidas as posições de memória 16 e 5 para verificar se o carregamento prévio ocorreu, e

em seguida, tais posições foram escritas com os dados 18 e 20. Por fim, as posições foram lidas e os dados 18 e 20 retornados, conforme deveria ter sido.

DIFICULDADES E SUGESTÕES

Como principais dificuldades, foi observado a utilização do Verilog HDL e sincronização do conteúdo entre os programas de apoio: Quartus e Modelsim.

Como sugestão, a dupla afirma que uma forma de auxiliar futuros trabalhos práticos seria o anúncio prévio de possíveis problemas que podem aparecer e a disponibilização de sites de apoio sobre a linguagem utilizada, visando amenizar possíveis falhas no aprendizado anterior do aluno.

CONSIDERAÇÕES FINAIS

Com essas partes da prática I foi possível revisar sobre a utilização de algumas ferramentas do software Quartus II. Na próxima parte, a dupla aprofundará mais na questão dos tipos de memória, entendendo como funciona uma cache.

Referências Bibliográficas

[1] Ram megafunction: User guide, aug 2018.

[2] D. A. Patterson and J. L. Hennessy. *Computer Architecture: A quantitative approach*. Elsevier, 2011.