Laboratório de Arquitetura e Organização de computadores II

Prática 1 - Parte 3

Alunos: Anélio Gonçalves Caldas e Ryan Eduardo Mansur Vasconcelos

1. Introdução

Na parte 3 dessa prática, vamos implementar uma hierarquia de memória inclusiva, com uma cache L1 mapeada associativamente por 2 vias, e uma memória principal diretamente mapeada utilizando a biblioteca LPM . Além disso, também é necessário utilizarmos uma política de write-back para fazer as substituições.

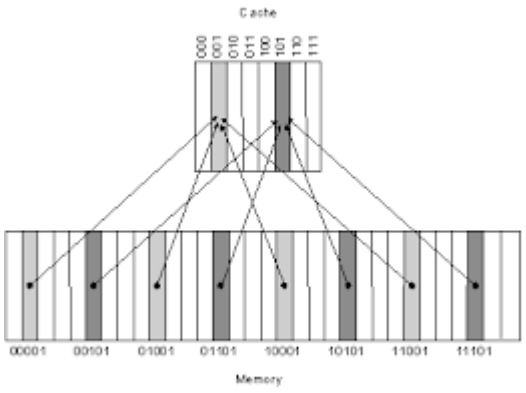
2. Objetivos

O objetivo desta prática é implementarmos os conceitos vistos nas aulas sobre hierarquia de memória, além de utilizar o que já implementamos anteriormente.

3. Desenvolvimento

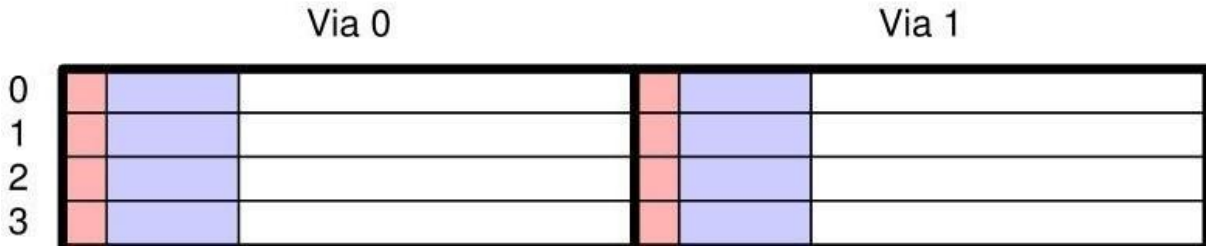
3.1 Projeto

Para esta prática, foi utilizada uma memória principal diretamente mapeada. Abaixo um exemplo de como funciona essa organização na imagem abaixo.



A posição da cache é mapeada pela posição de memória. Quando há a necessidade de subir um bloco da memória para cache é usado o resultado do Endereço de memória MOD Tamanho da cache.
Para esta prática, a memória principal tinha 16 posições.

A cache tinha uma organização com 2 vias, ilustrada na imagem abaixo.



Essas duas vias eram mapeadas, e a escrita nelas eram organizadas por um bit de LRU. Para a escrita na memória principal utilizamos o WriteBack. Foi criado um bit chamado Dirty, cuja função era controlar valores que já foram escritos para quando houvesse outra escrita, esse valor fosse guardado na memória principal.

3.2 Código

A memória principal foi criada utilizando a biblioteca LPM, e iniciada por meio de um arquivo .MIF.
A memória cache foi codificada para funcionar de acordo com as especificações da prática. Ela foi criada como um tipo “reg” e inicializada com um arquivo .mem.
Para ser possível fazer a leitura da memória, foram necessários 3 ciclos de clock, os dois primeiros para mapear a posição da memória principal e o último para controlar a cache, e analisar a necessidade de writeback, ou de usar o valor obtido da memória nos ciclos de clock anteriores. Para este controle, foi criado um sinal chamado “CLK”, que é ligado nos 2 primeiros clock indicando que será feita a leitura da memória principal. Abaixo um pequeno trecho do código que exemplifica o uso do CLK e da LPM.

```
//Ler ou escreve na memoria
ramlpm mem_principal(
endereco[4:0],
clock,
memw,
wback,
memr);

always @(posedge clock) begin

   index = address[1:0] << 1; //deslocando para pegar de 2 em 2

   if(clk) begin //se ta ligado a leitura de endereco
      endereco[4] <= 1'b0;
      endereco [3:0] <= address[3:0];
      wback <= 1'b0;
   end
```

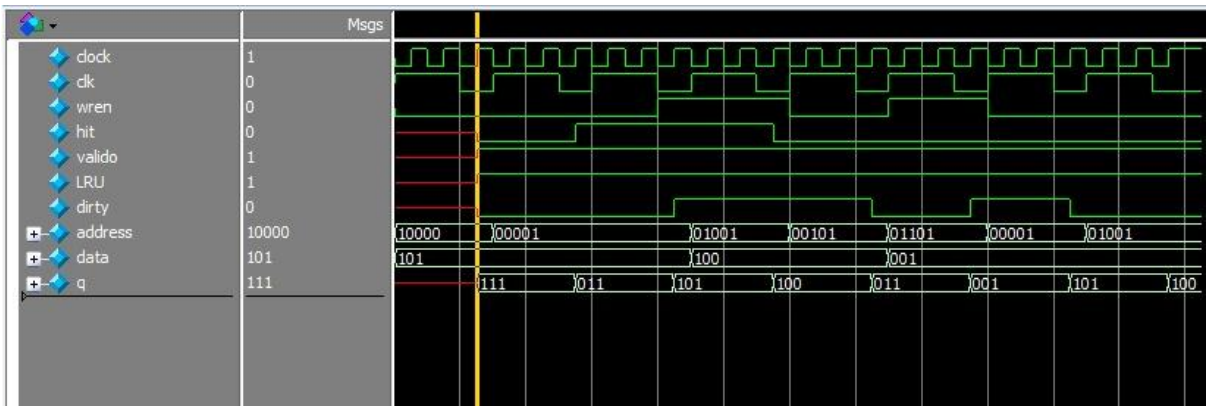
Note no código o uso do reg memw e memr, que indicam os valores que serão lidos e escritos na memória principal. É possível notar o uso do ``wback'', que indica a LPM quando deve ser escrita a palavra na memória principal. E por fim, o uso de um index, que será usado na codificação para o controle da via 1 e 2 da cache.

Para o teste na placa FPGA foi codificado o programa abaixo.

```
cache(KEY[0],//clock
      SW[17], //clk
      SW[8:4],//address
      SW[16],//wren
      LEDR[10],//hit
      LEDR[11],//valido
      LEDR[12],//LRU
      LEDR[13],//dirty
      SW[2:0],//data
      q);


  //Mostrando dados de entrada endereço
  decodificador_c h_address(SW[8],HEX7[0:6]);
  decodificador_c h2_address(SW[7:4],HEX6[0:6]);

  //Entrada do valor
  decodificador_c h_data(SW[2:0],HEX4[0:6]);
  decodificador_c h2_data(4'b0000,HEX5[0:6]);

  //Mostrando saida
  decodificador_c h_q(q[2:0],HEX0[0:6]);
  decodificador_c h2_q(4'b0000,HEX1[0:6]);
```

O decodificador usado, é o mesmo utilizado nas práticas anteriores.

3.3 Simulação



A simulação segue de acordo com o esperado no arquivo disponível na disciplina. Podemos notar os 3 ciclos de clock, e a saída q mudando para o valor esperado no terceiro ciclo, quando o clk recebe o valor 0. Também notamos na simulação o valor de hit, que é 0 quando há um miss e 1 quando tem um hit. Os demais valores, válido, LRU e dirty, as saídas mostram sempre para quais valores eles foram atualizados.

4. Conclusão

Após a execução desta prática, podemos assimilar melhor o como funcionam as hierarquias de memória, e quais são as vantagens e dificuldades de implementação, assim como, aprimorar conhecimentos vistos em aulas por meio da prática em verilog.