Relatório Prática 1

Nomes: Abdul-Kevin Alexis e Pedro Santos Oliveira

Parte 3

Inicialização da Cache:

VALID	DIRTY	LRU	TAG	DADO
1	**0**	**0**	**20**	**6**
1	**0**	**1**	**22**	**2**
0	**0**	**3**	**25**	**6**
1	**0**	**2**	**21**	**4**

Inicialização da RAM:

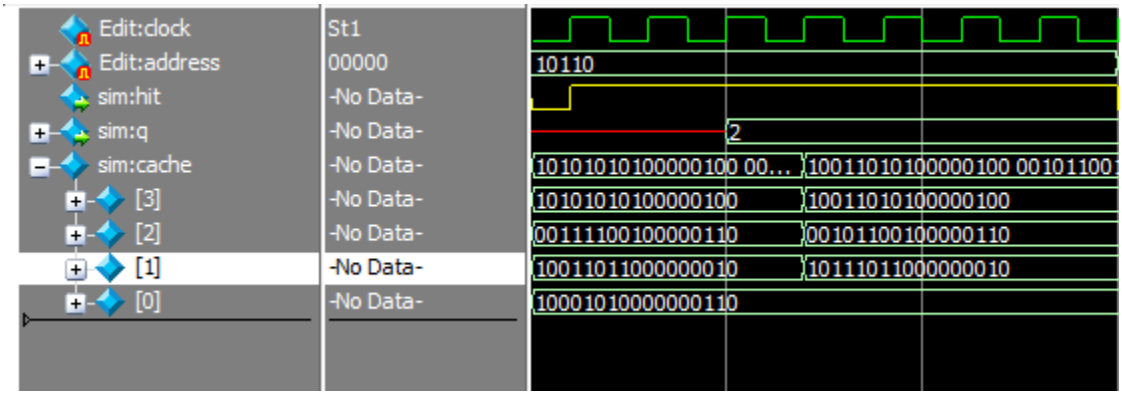
Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	0	0	0	0	0	0	0	0	.....
8	0	0	0	0	0	0	0	0	.....
16	0	0	0	0	6	4	2	4	.....
24	9	6	4	5	6	7	8	7	.....

Organização da Cache:

[7:0] => Dados armazenados
[12:8] => Tag do endereço
[14:13] => Bit de controle LRU (3 = mais recente, 0 = mais antigo)
[15:15] => Bit de controle Dirty
[16:16] => Bit de controle Valid

Teste Hit e Leitura: Testar se a posição acessada está mapeada na cache e então realizar a leitura do dado dessa posição. Checar atualização do LRU.

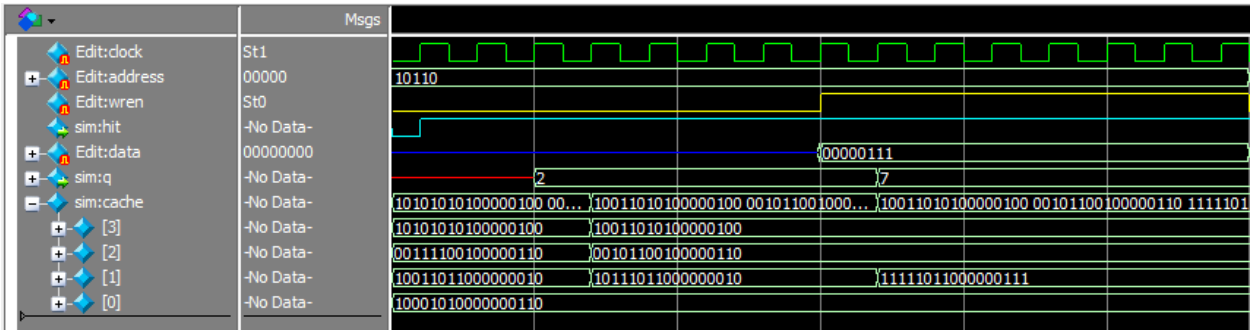
Simulação:



Conclusão: Na simulação acima acessamos o endereço de Tag 22, que está presente na posição 1 cache como mostrado acima. Portanto, ao acessar esse endereço podemos perceber que ocorreu um hit na cache, logo o sinal de “hit” fica com o valor 1, e o valor salvo no endereço de Tag 22 é retornado, no caso esse valor é 2 (decimal) como também foi mostrado acima. Por fim, após o acesso a cache os bits de LRU (bits 14:13) são atualizados, e podemos perceber então que o LRU da posição da cache onde está armazenado o endereço de Tag 22 era 1 (decimal) antes do acesso, e passa a ser 3 (decimal) depois do acesso. Assim, o comportamento do sistema para esse teste foi o comportamento correto e esperado.

Teste Hit e Escrita: Testar se a posição acessada está mapeada na cache e então realizar a escrita do dado passado na entrada na respectiva posição desejada. Checar atualização do Dirty e do LRU.

Simulação:

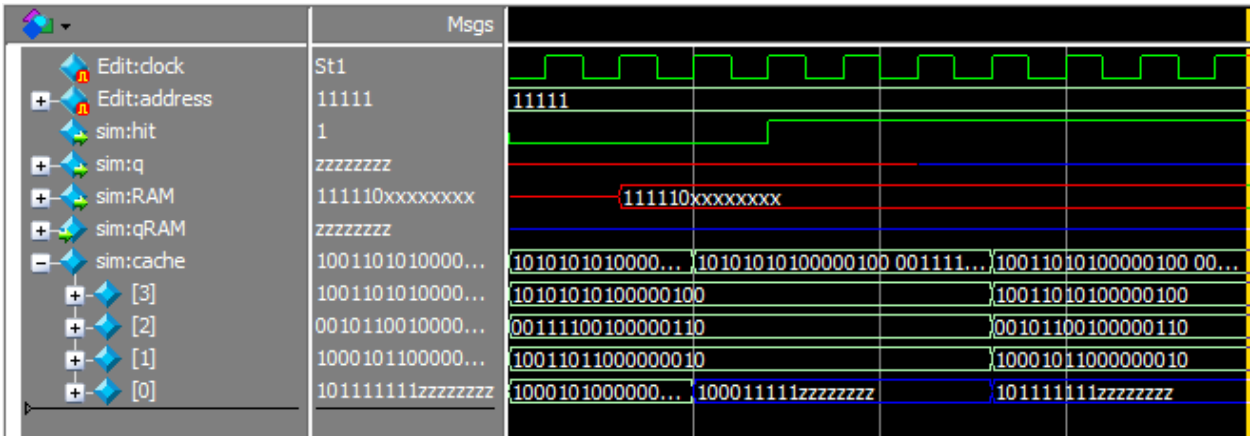


Conclusão: Na simulação acima acessamos o endereço de Tag 22, que está presente na posição 1 cache como mostrado acima. Inicialmente realizamos a leitura dessa posição e posteriormente realizamos a escrita nessa mesma posição. A primeira parte da simulação é apenas uma leitura da posição igual mostrada no teste anterior, já a segunda parte é quando ocorre a escrita na posição. Primeiramente, ao acessar esse endereço podemos perceber que ocorreu um hit na cache, logo o sinal de “hit” fica com o valor 1, e o valor salvo no endereço de Tag 22 é retornado. Em seguida, podemos ver que o sinal de

escrita é ativado, “wren” recebe o valor 1, e é passado um dado como entrada, nesse caso escolhemos escrever o valor 7 (decimal) na posição escolhida. Com o sinal de escrita ativo e o dado passado como entrada podemos perceber que o valor de “q”, que indica a saída da cache, muda de 2 para 7, indicando que a escrita foi realizada com sucesso na posição 1 da cache. Por fim, podemos perceber que o bit Dirty (bit 15) da posição onde foi realizada a escrita tem seu valor alterado de 0 para 1. Assim, o comportamento do sistema para esse teste foi o comportamento correto e esperado.

Teste Miss Sem Write Back: Testar se ocorreu um miss na cache ao acessar a posição desejada. Testar se a posição da memória desejada foi acessada corretamente (RAM, qRAM). Testar se a cache foi atualizada corretamente (LRU, Valid, Dirty, Tag, Dado). Por fim, analisar se ocorreu um hit na cache.

Simulação:



Conclusão: Nesta simulação podemos perceber que estamos acessando o endereço de Tag 31, que é um endereço que não está na cache. Portanto, num primeiro momento é indicado um miss e o sinal de “hit” recebe valor 0, em seguida a variável RAM recebe o endereço de Tag 31 para que ele possa ser acessado na memória principal e trazido para a cache. Podemos analisar que o bloco presente na posição da cache que possui os bits de LRU com valor 0 (decimal) é retirado e o bloco referente ao endereço de Tag 31 é inserido no lugar. Após isso, é indicado um hit na cache e o valor presente no endereço de Tag 31 é indicado na saída da cache, demonstrando o funcionamento correto e esperado do sistema. OBS: O Modelsim não reconheceu nosso arquivo .mif, por isso o dado presente na posição de Tag 31 consta como “zzzzzzzz”, porém na simulação podemos perceber que o funcionamento é correto pois a saída da cache indicada pelo sinal “q” muda para a cor azul quando o dado “zzzzzzzz” é lido, o único erro é que no lugar de “zzzzzzzz” deveria estar mostrando o valor 7 (decimal) como especificado no arquivo .mif mostrado acima.

Teste Miss com Write Back: Testar se ocorreu um miss na cache ao acessar a posição desejada. Checar se os dados passados para serem salvos na memória estão corretos (RAM). Testar se a posição da memória desejada foi acessada corretamente (RAM, qRAM). Testar se a cache foi atualizada corretamente (LRU, Valid, Dirty, Tag, Dado). Por fim, analisar se ocorreu um hit na cache.

Conclusão: No caso desse teste nós não conseguimos fazer com que o dado escrito no bloco da cache fosse salvo na memória quando este fosse retirado da cache. Testamos diversas simulações e em todas elas o comportamento do sistema funciona corretamente, porém quando buscamos o bloco que foi alterado da memória, este volta sempre com o valor inicializado ao invés de voltar com o valor escrito. Portanto, não conseguimos identificar o porquê do nosso sistema não escrever os dados na memória e não conseguimos realizar essa simulação.