**RELATÓRIO LAOC2**

**Prática I (Parte III)**

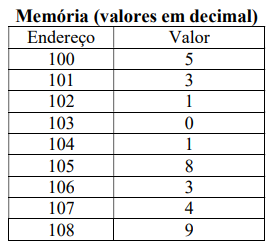
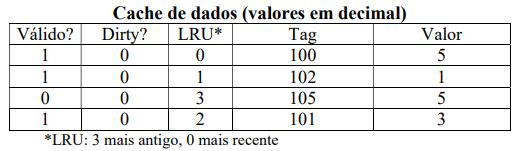
Dupla:

\*Arthur Severo de Souza

\*Victor Le Roy Matos

Na terceira parte da prática sobre hierarquia de memória fomos orientados à implementar um nível (L1) de memória cache totalmente associativa de quatro vias, este em vínculo com uma memória principal diretamente mapeada e atualizada por meio de *Write-Back*.

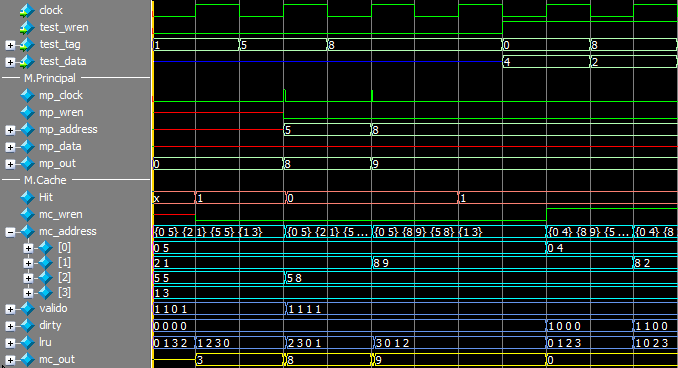
Os seguintes valores iniciais foram definidos:



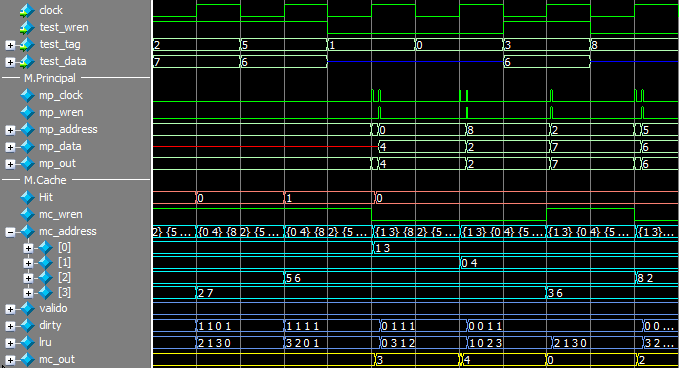
**Apresentação das simulações**

As operações requisitadas foram efetuadas em ordem de apresentação e acontecem em bordas de subida do clock.

Operações **1 -> 6**

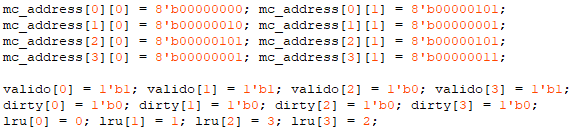


Operações **7 -> 12**

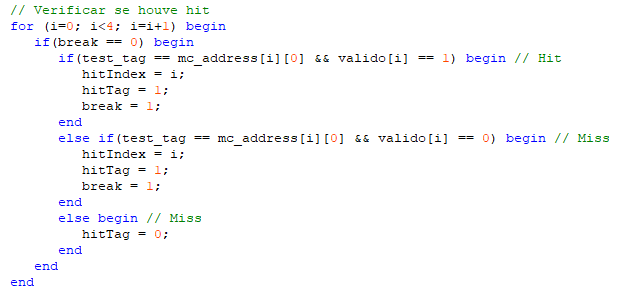


**Análise do código**

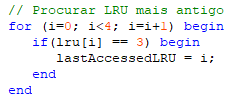
No código, a tabela inicial da cache foi traduzida em uma matriz contendo as tags e os dados. É importante ressaltar que os códigos das tags foram traduzidos da tabela de valores iniciais, isto é, os códigos foram de (100-108) para (0-8). Os bits de validade, dirty e o LRU foram armazenados em um array de index equivalente às linhas da matriz.



Então, a partir dos valores iniciais implementamos um bloco que identifique se houve hit ou miss ao acessar a memória cache. O loop percorre todos as quatro posições da cache e, caso encontre uma posição onde o bit de validade é de nível alto e a tag inserida seja equivalente à tag lida, identifica um hit no acesso, salva o index dos dados e abandona o laço.

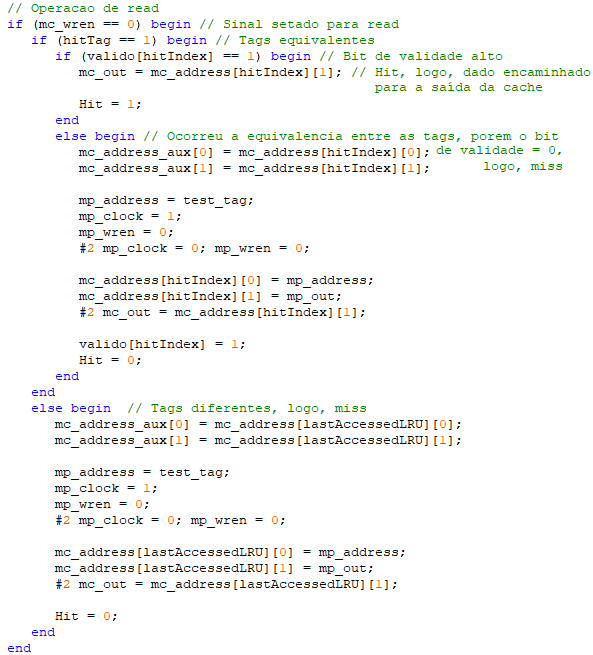


O laço seguinte é responsável por armazenar o index do acesso mais antigo para uso futuro.

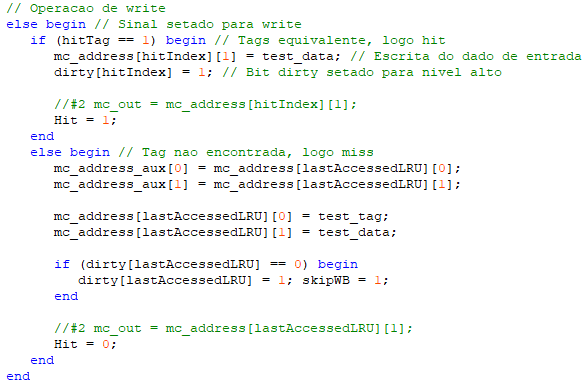


Implementamos as operações de read e write a depender do resultado do bloco de verificação de hit ou miss.

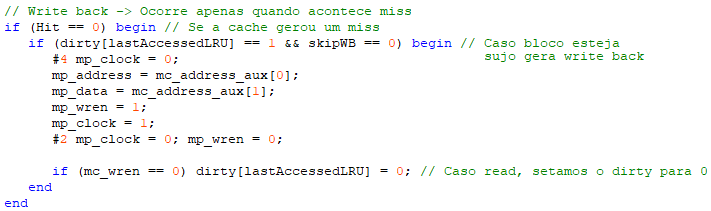
O bloco seguinte representa a operação de leitura. Primeiro, conferimos se o sinal wren está definido como leitura. Confirmamos também que as tags lida e recebida foram identificadas como equivalentes e se o bit de validade é de nível alto. Se ambos os sinais se mostrarem adequados, localizamos o index do dado procurado na matriz inicial e enviamos este para a saída da cache. Caso contrário, é preparado o ambiente para a possibilidade de um *Write-Back*.



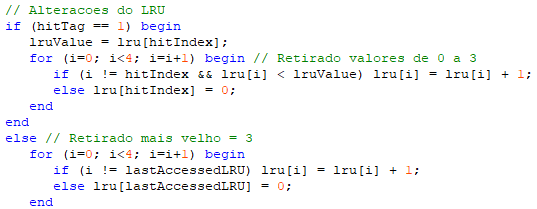
Já o bloco de escrita depende do sinal wren em nível alto e basta que as tags inseridas e lida sejam equivalentes. Se for o caso, o dado inserido é escrito no index localizado durante a conferência do hit e o bit dirty tem seu valor definido como alto. Se ocorreu miss, o ambiente é preparado para a ocorrência de um *Write-Back*.



O *Write-Back* então é efetuado pelo bloco a seguir, se foi identificado um miss no acesso. São tratados todos os possíveis casos responsáveis pela ocorrência do *Write-Back*. Os sinais auxiliares têm seus valores armazenados na posição mapeada da memória principal e, no caso de uma operação read, o bit dirty do index de acesso mais antigo é definido como baixo.

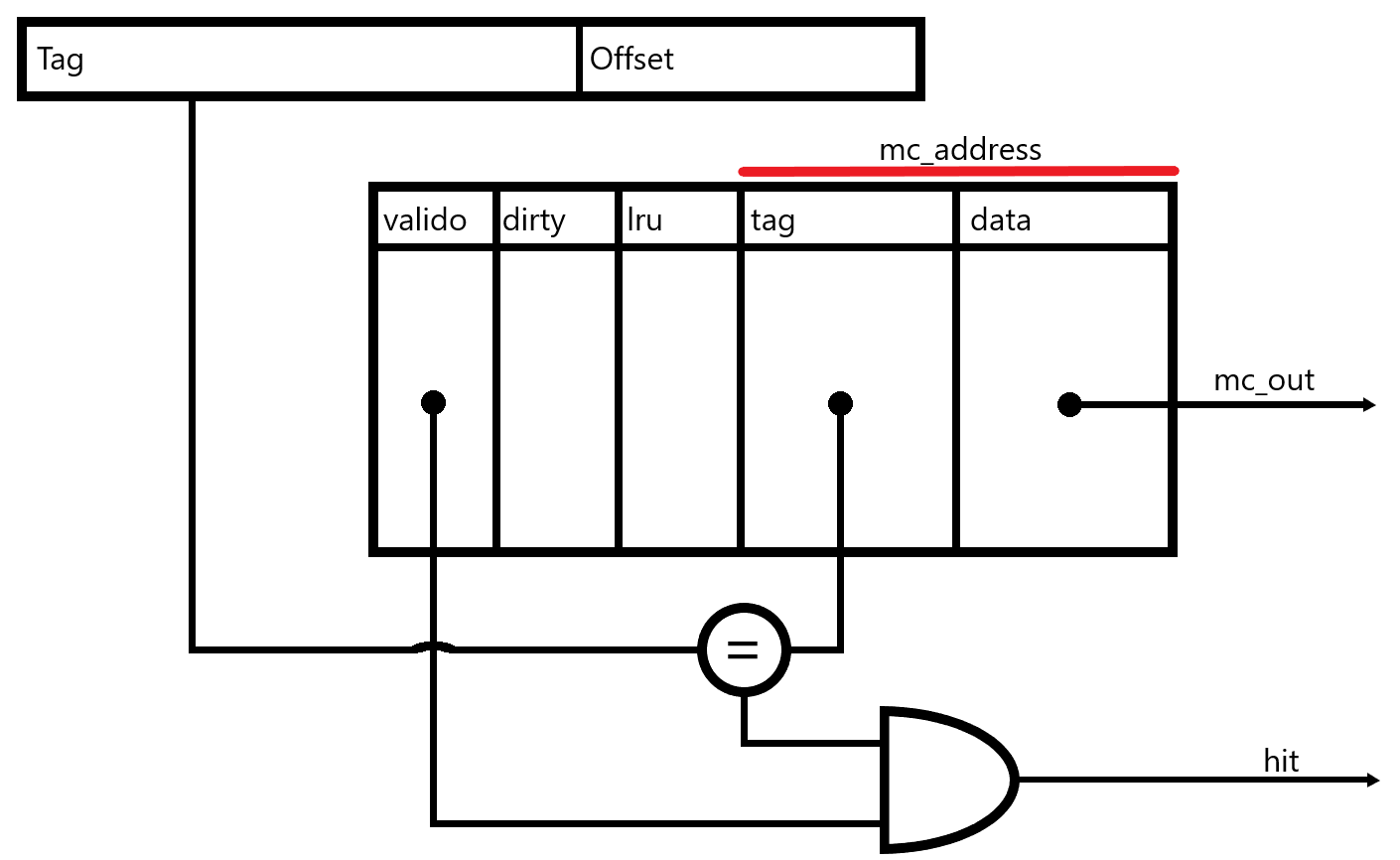


Por fim, condizente com o funcionamento em uma hierarquia implementada fisicamente, efetuamos a atualização do LRU. Os laços percorrem cada index da memória cache, incrementando os números menores que o LRU da posição acessada e zerando o valor na posição em questão. No caso de acesso à posição de LRU mais alto, esta é zerada.

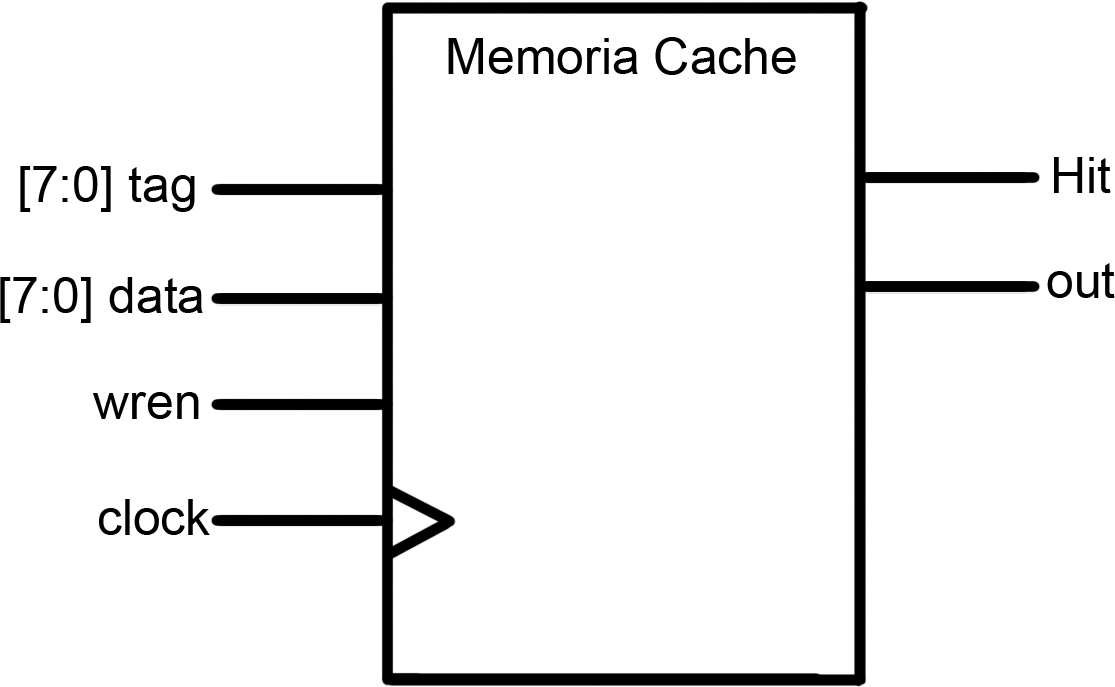
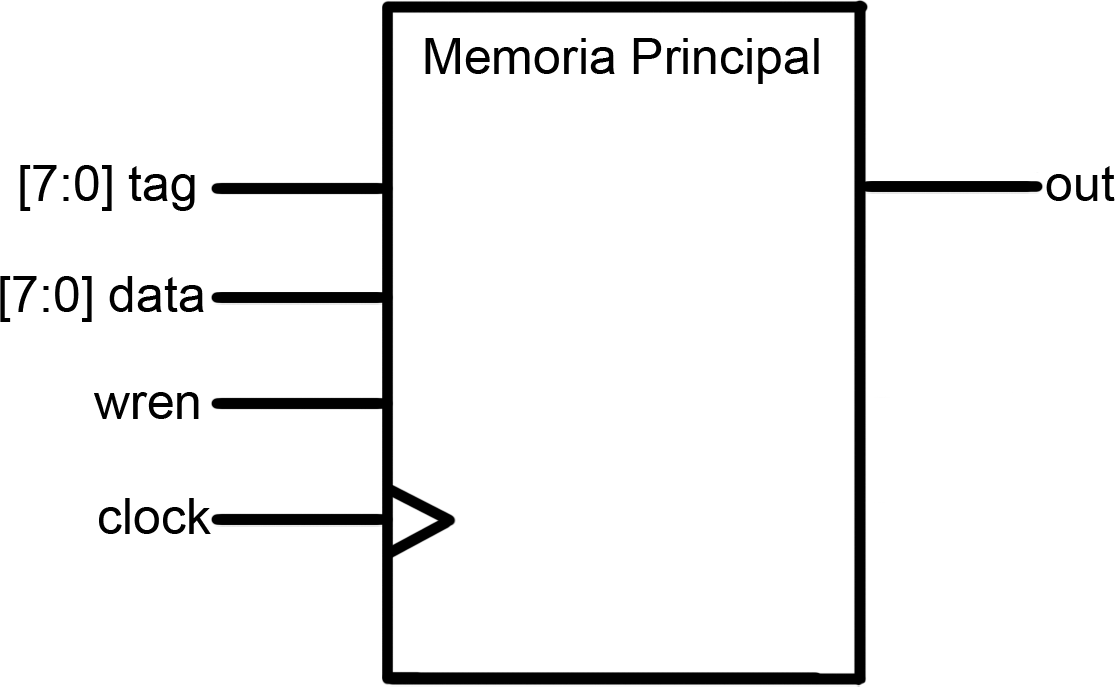


**Projeto**

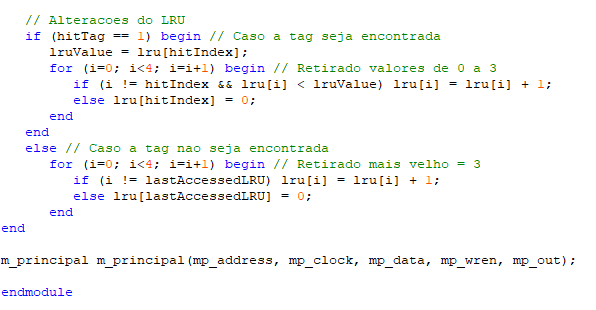
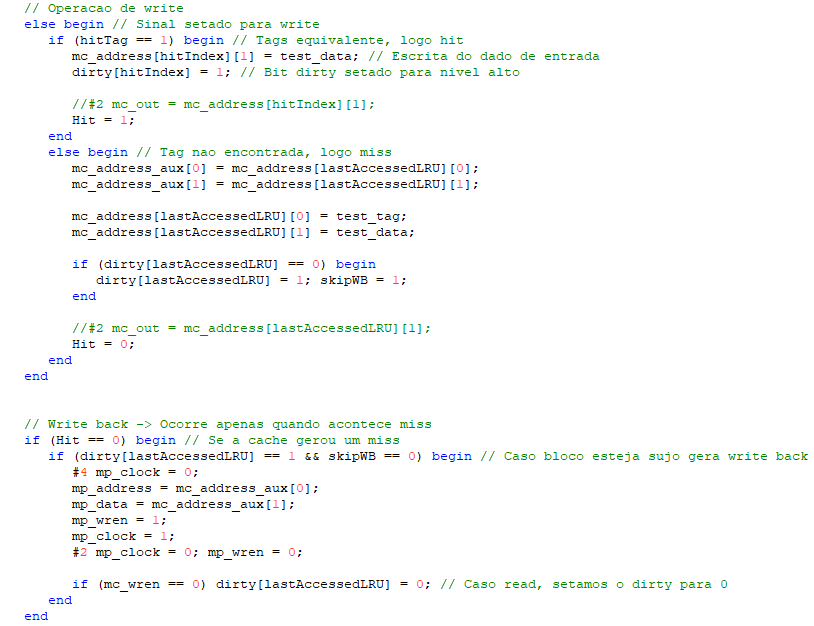
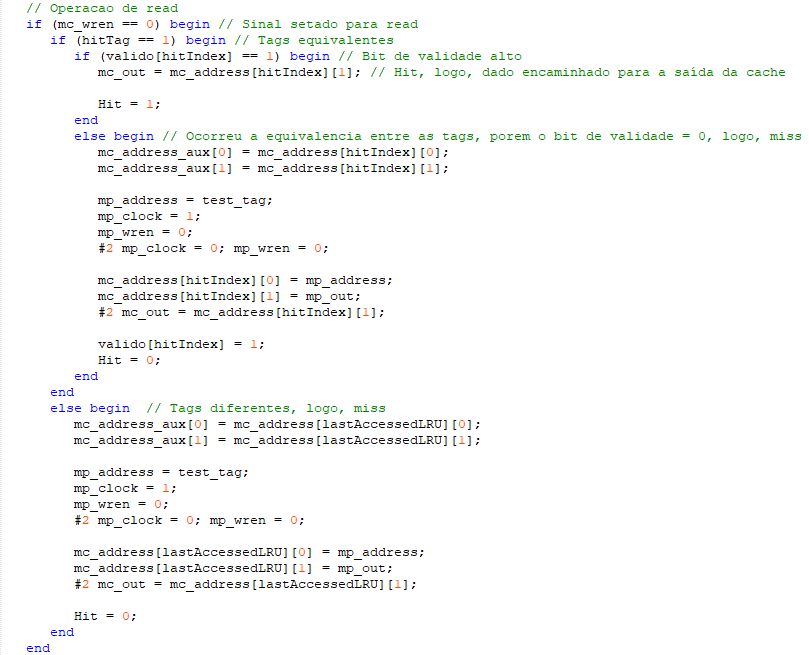
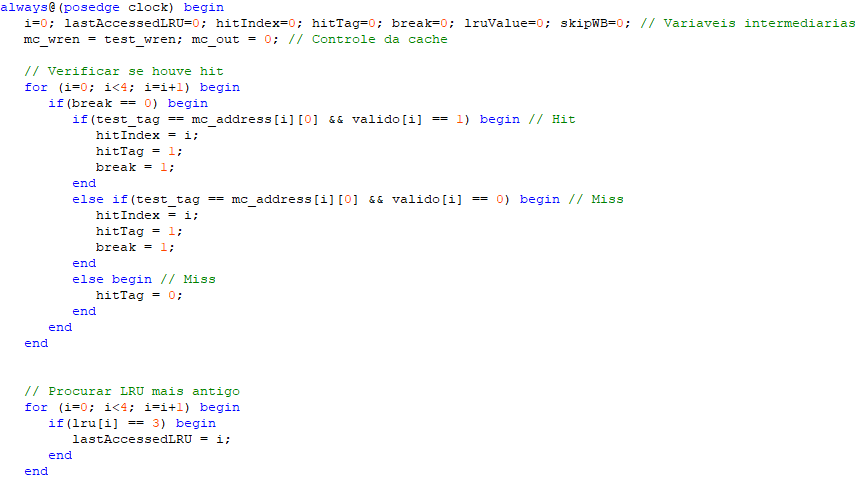
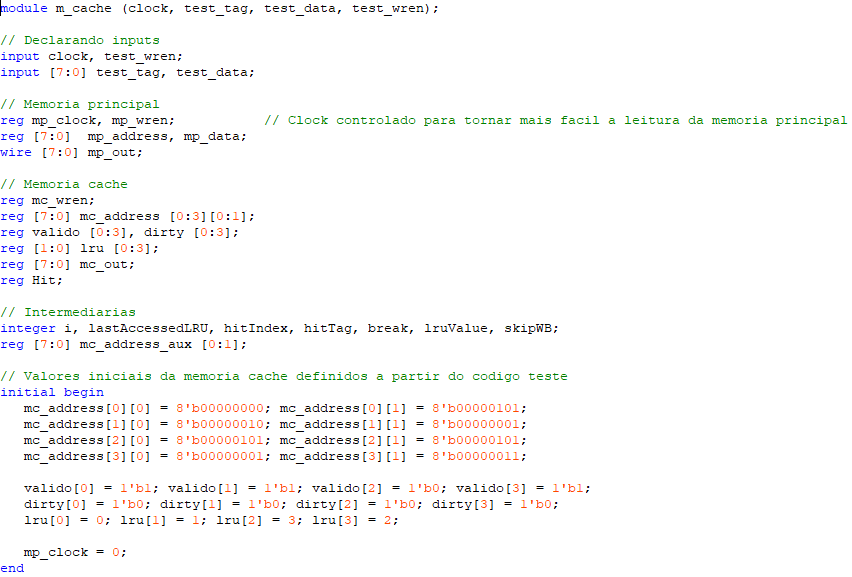
Simplificação das conexões da cache (input e output)

****

Blocos presentes no projeto



**Código completo da simulação**



**Configuração modelsim**

Para realizar a simulação no modelsim, foi necessário configurar o clock:

*Offset: 0*

*Duty: 50*

*Period: 100*

*First Edge: Falling*

Além disso, para melhor visualização, adicionamos todas as variáveis de input/output, além das principais da memória cache e memória principal.