Universidade Federal Rural do Semiárido
Departamento de Computação
Ciência da Computação
Arquitetura e Organização de Computadores
Prof. Sílvio Fernandes

3.2 – Trabalho de Implementação

Trabalho em dupla ou individual
Data de entrega (via SIGAA): 09 de Outubro de 2022
Pontuação: 50% da nota da III Unidade

Descrição:
Construir uma memória cache com mapeamento direto usando o Logisim.

Para isso são necessárias 4 coisas:
1) Criar um circuito relativo à memória cache com mapeamento direto
2) Ligar essa cache ao circuito testador que simula o caminho de dados do MIPS
3) Verificar se seu projeto funciona e se os testes fornecidos contabilizam a quantidade correta de *hit* da cache

No projeto do LogiSim deve ter **os nomes dos componentes** e o arquivo deve ser salvo também com o nome dos componentes. Exemplo: NomeA-NomeB-mips-datapth-lab.circ.
Este arquivo deve ser submetido na tarefa do SIGAA dentro do prazo.

Para começar:

Leia o capítulo 5 do livro principal desta disciplina listado nas referências no final deste texto (livro do Patterson).

Instale o Logisim, versão 2.7.1, no seu computador, também disponível no SIGAA. Para executar este software é necessário que em seu computador também esteja instalado o *Java Runtime*.

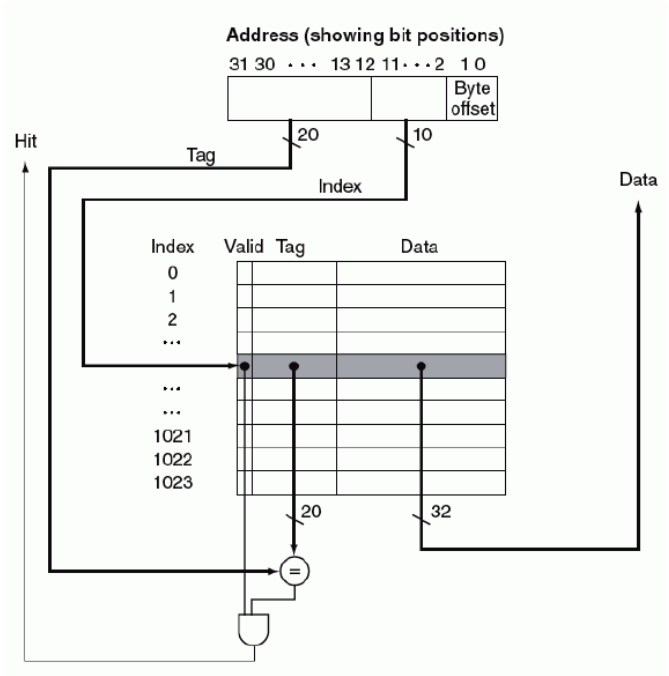
Utilize como ponto de partida o arquivo "cache-lab.zip", fornecido juntamente com a definição deste trabalho. No arquivo .zip será encontrado os seguintes arquivos:

- cache-isolada.circ: um "esqueleto" do processador MIPS básico com importantes partes faltando propositalmente.
- address.mem: um arquivo de memória hexadecimal que contém os valores dos endereços de memória que serão acessados nos testes
- data.mem: um arquivo de memória hexadecimal que contém os dados que serão escritos na cache quando um acesso for de escrita na memória, usado para os testes
- we.mem: um arquivo de memória hexadecimal que contém os sinais que indicam quando a cache deve ser escrita ou não, usado para os testes

Passo 1: criar o circuito da cache

Desenvolveremos uma cache de mapeamento direto na qual cada linha armazena 4 palavras do MIPS. Assim, a cache deve receber o endereço de memória (32 bits) e dividi-lo como a

Figura a seguir. A quantidade de palavras utiliza os bits mais a direita, que neste caso será fixa em 2 bits para o *offset* de palavras (apesar da figura mostrar byte offset).

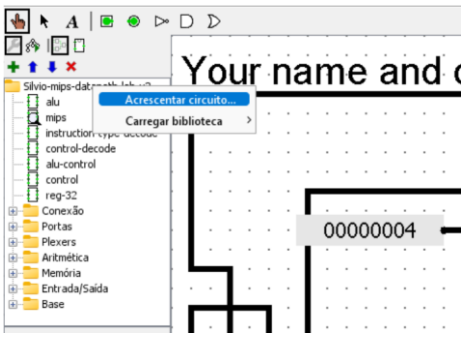


Logo após os bits de offset, mais a esquerda, estão os bits de índice, que indicam a linha da cache que será acessada. No exemplo da Figura temos 10 bits de índice, logo, a cache deve ter 1024 linhas. Na nossa implementação, por questões didáticas, podemos usar números menores para os bits de índice, então usaremos 2, o que corresponde a 4 linhas de cache, respectivamente. A quantidade de bits para *tag* corresponde ao restante dos 32 bits de endereço, ou seja, no exemplo da Figura, como foram usados 10 bits de índice, sobraram 20 bits para a *tag*. No nosso caso, os bits de *tag* serão 32-2-2, ou seja, 28 bits.

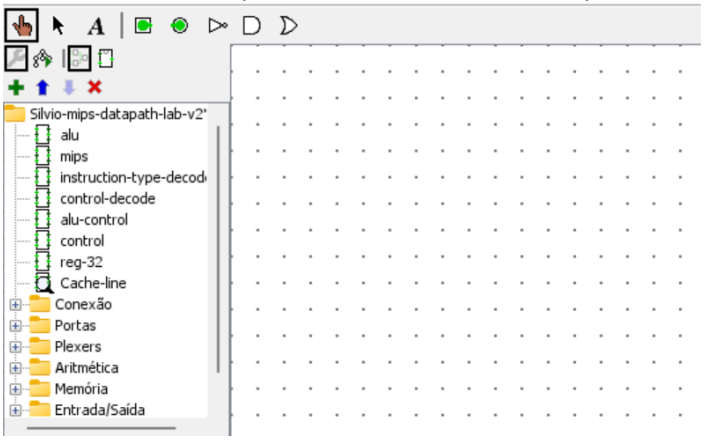
Para facilitar os testes, no arquivo disponibilizado do LogiSim já possui os componentes “Main”, “cache”, “cache-line” e “Testbench”. O “main” é o principal que instancia a “cache” e “Testbench”. O “Testbench” possui 3 memórias (Data, Address e We) que devem ser inicializadas respectivamente com os arquivo “data.mem”, “address.mem” e “we.mem”, pois indicam uma sequência de acessos a memória (escrita ou leitura), conforme planilha “Acessos.xlsx”. Logo, seu trabalho é implementar **apenas** os componentes “cache” e “cache-line”, não alterando em nada os demais. Inclusive os componentes “cache” e “cache-line” já possuem as interfaces (entrada/saída) necessárias para o teste, e portanto, também não devem ser alteradas. A seguir são descritos como “cache” e “cache-line” foram criados e quais funções você deve implementar.

Circuito *Cache line*

Para criar o circuito da cache vamos primeiro foi criado o componente que implementa a linha da cache. Este circuito poderá ser instanciado várias vezes pelo circuito geral da cache. Para isso, foi usado o botão direito do mouse sobre a pasta do seu projeto no menu esquerdo no LogiSim, e escolha a opção “Acrescentar circuito...”, como mostra a Figura a seguir.

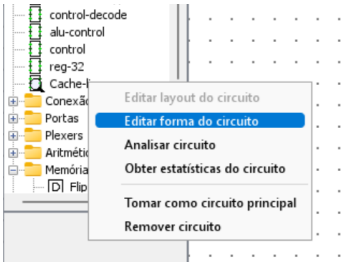


Em seguida foi dado o nome para seu circuito como “Cache-line” e confirmado em “ok”. Assim, aparece um espaço em branco do lado direito para implementá-lo e o nome do novo circuito também aparecerá na lista do lado esquerdo, como os demais componentes do seu projeto.

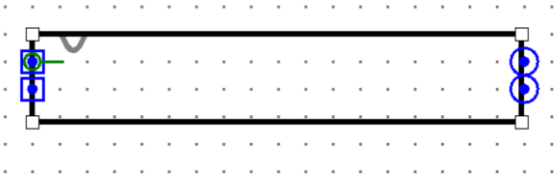


Utilize a estrutura de registradores, no menu à esquerda escolha “Memória -> Registrador”, para o armazenamento interno da linha da cache. Além de armazenar as 4 palavras de 32 bits a linha da cache também deve armazenar o bit de validade e a *tag*, indicando qual bloco da memória foi armazenado na cache. Sempre que receber um sinal de *reset* a linha da cache deve zerar as informações armazenadas anteriormente. Adicione outras estruturas (multiplexadores, portas lógicas, etc.) para o funcionamento adequado desse circuito. O circuito ainda tem as entradas para *tag*, dados e *offset*, além de *clock* e *reset*, além de habilitação de escrita ou WE (*write enable)*. As informações de entrada serão armazenadas apenas quando WE for 1, na subida do *clock*. As saídas devem mostrar o valor da *tag*, do dado e bit válido armazenados. As portas de entrada e saída possuem a quantidade de bits adequada e rótulos para cada uma delas, para facilitar quando for utilizar esse circuito.

O layout do circuito foi modificado, inclusive modificando o posicionamento das portas, para aparece em formato retangular na horizontal, quando instanciado no circuito “cache”. Para isso usou-se o botão direito do mouse sobre “cache line”, no menu à esquerda na lista de componente. Foi escolhida a opção “Editar forma do circuito”, conforme a Figura a seguir.



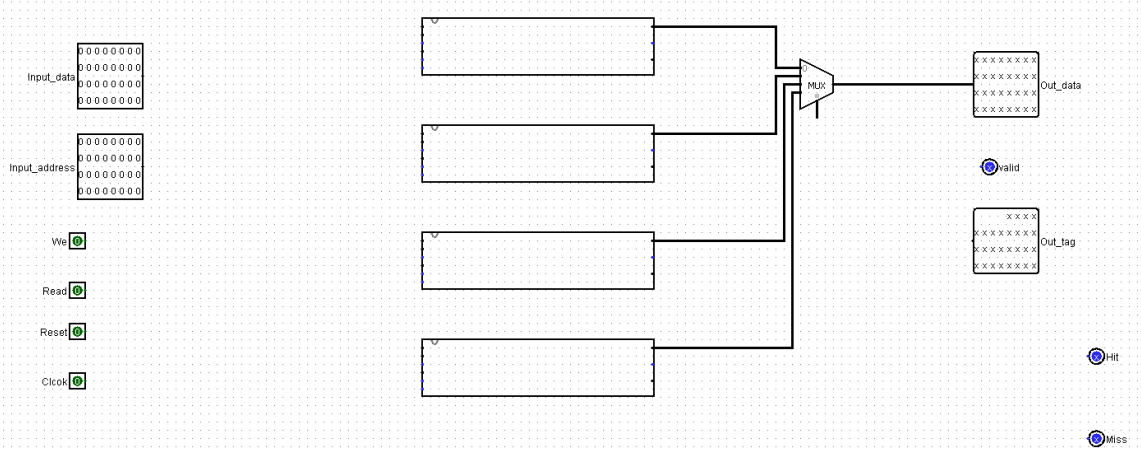
Modificou-se para forma retangular que lembre uma linha, veja o exemplo na Figura a seguir. Algumas portas de entrada ou saída do circuito foram movidas para melhor adequá-las na nova forma.



Após implementar este circuito teste-o de forma isolada para ver se os dados são armazenados adequadamente, nos locais corretos. Da mesma forma, teste se as informações armazenadas são recuperadas adequadamente, como se espera quando uma linha da cache é selecionada.

Circuito *Cache*

Após concluir e testar o circuito *cache line* ele poderá ser instanciado no circuito "cache". No componente fornecido isso já foi feito, instanciando 4 cópias do componente anterior. Para isso foi criado um novo circuito, da mesma forma que o anterior e nomeado como "Cache". Em seguida, podem ser instanciadas várias linhas de cache, para isso basta clicar no componente "Cache Line" (menu à esquerda) e depois escolher o local onde colocá-lo no espaço de implementação. Veja o exemplo na Figura a seguir, onde foram instanciadas 4 "Cache Lines". As entradas e saídas do componente "Cache" já estão incluídas com a quantidade de bits corretas. Um exemplo da saída de como selecionar os dados das 4 linhas para a saída da cache "out_data", utilizando um multiplexador, é mostrado na Figura.



Adicione outras estruturas (multiplexadores, portas lógicas, etc.) para o funcionamento adequado desse circuito. Utilize corretamente as informações das entradas para ligar as entradas das "cache lines", bem como para selecionar os multiplexadores e/ou lógica booleana adicionada. Este circuito também indica quando acontece um *hit* ou um *miss*. O *hit* acontece quando um dado válido está sendo lido (não está sendo gravado) e o mesmo tem a mesma *tag* armazenado que a *tag* informado na entrada. Um *miss* acontece um dado está sendo lido (não escrito) e ou não é válido ou não possui a mesma *tag* que o informado no endereço recebido.
Teste seu circuito colocando valores na entrada, analise as informações armazenadas internamente e as saídas produzidas por ele para avaliar se seu funcionamento está correto.

Passo 2: testar a cache

Seu circuito de cache está conectado ao *testbench* que fornece a sequência de acesso. O teste deve ser feito no circuito principal "main", o qual exige a quantidade de miss, de hit, o total de

acesso, as informações de saída da cache (tag, bit de válido e dado), além do endereço de entrada enviado para a cache. As memórias do testbench devem ser inicializadas com os arquivos de mesmos nomes delas, os quais representam 25 acessos a memória. Assim, o teste completo desse circuito é realizado quando o contador de acessos chegar a 27, uma vez que atualizações dos acessos acontecem com atrasos de ciclos.

REFERÊNCIAS:

HENNESSY, John. Organização e Projeto de Computadores. 5 ed. Rio de Janeiro: Elsevier. Grupo GEN, 2017. E-book. ISBN 9788595152908. Disponível em: https://app.minhabiblioteca.com.br/#/books/9788595152908/.

FOYER, Per. Notas de aula de “Computer Architecture - 1DT016”. Disponível em: https://xyx.se/1DT016/labs.php#lab1