CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

ENGENHARIA DE COMPUTAÇÃO

Abdul Kevin Alexis e Pedro Santos Oliveira

RELATÓRIO DA PRÁTICA 4: SNOOPING

Belo Horizonte

24.11.2023

Abdul Kevin Alexis e Pedro Santos Oliveira

RELATÓRIO DA PRÁTICA 4: SNOOPING

Relatório da prática 4 da disciplina de laboratório de arquitetura e organização do computador, curso Engenharia de Computação, CEFET-MG/Centro Federal de Educação Tecnológica de Minas Gerais.

Professor: Poliana Aparecida Correa De Oliveira

Belo Horizonte

24.11.2023

RESUMO

Neste trabalho está descrito com precisão, todas as atividades desenvolvidas pelos alunos Abdul Kevin Alexis e Pedro Santos Oliveira do CEFET-MG do curso Engenharia de Computação sobre a quarta atividade de laboratório de arquitetura e organização do computador.

Todas as atividades desenvolvidas nesta prática visam atender a teoria aprendida na disciplina arquitetura e organização do computador.

A prática realizada é fundamentalmente importante , pois proporcionou uma excelente oportunidade para a aplicação das teorias estudadas até o momento.

Palavras-Chave: MESI, Coerência de cache

1 - Decisões Gerais de Projeto

1.1 - Parte 1

a) Para a Parte 1 deste projeto iremos considerar 4 entradas e 3 saídas no módulo principal.

b) As entradas são: *clock*, *state*, *transition* e *listener_or_executer*.

c) A entrada *state* tem como objetivo dizer o estado atual do bloco que está sendo simulado. Já a entrada *transition* procura dizer qual a transição da máquina de estados será considerada para a simulação. Por fim, a entrada *listener_or_executer* tem o papel de indicar se o bloco simulado é um bloco emissor ou ouvinte.

d) As saídas são: *bus*, *write_back* e *new_state*.

e) A saída *bus* visa mostrar a mensagem que será colocada no barramento quando ocorrer a simulação. Já a saída *write_back* servirá para indicar quando uma transição realizou escrita na memória. Por fim, a saída *new_state* vai indicar qual o novo estado do bloco simulado após a simulação.

f) A saída que indica a ocorrência de escrita na memória é inicializada com um bit igual a 0.

1.2 - Parte 2

a) Posições da memória de instruções que possuem os 16 bits iguais a 1 são posições inválidas.

b) A cache L1 de cada processador foi implementada utilizando mapeamento direto, e cada posição da cache guarda a Tag, Estado e Dado do bloco em questão. Abaixo será mostrada a disposição dos bits detalhadamente.

c) Em relação às transições de Read Miss, foi implementado um sinal *read miss* responsável por avisar os componentes do projeto quando ocorre uma transição de Read Miss. Sempre que esse sinal é enviado os processadores ouvintes executam uma certa

ação específica e podem ou não enviar dados contidos neles, já a memória ao ouvir um Read Miss, sempre enviará o dado necessário. O tratamento e a distinção entre Read Miss Shared e Read Miss Exclusive, é feito sempre quando chega um dado ou da memória, ou de algum processador, ou de ambos.

d) Em relação à transição de Write Miss, o bloco faltante sempre será buscado na memória de dados.

e) Em caso de Read Miss, o dado demora 4 ciclos, contando a partir do início da execução, para ser retornado para o processador.

f) Para realizar o controle do despacho de instruções foi implementado um bloco always no módulo principal do projeto, que tem como objetivo testar o sinal de *done* de cada processador a cada ciclo de clock, sempre que o sinal de *done* for 1 para todos os processadores uma nova instrução será enviada.

g) Para facilitar e simplificar a implementação, e evitar trabalhar com valores muito elevados, foi implementada uma memória de dados com as posições variando de 0 a 19, no entanto, foram consideradas para teste apenas das posições 10 a 14.

h) Foi implementado um arbiter para controlar quem pode escrever no Bus, a inclusão desse arbiter acaba atrasando a execução em 1 ciclo, porém é uma inclusão necessária para que não ocorram escritas simultâneas no Bus. Apenas o processador emissor pode escrever no Bus durante a execução de uma instrução.

i) Os sinais que representam as transições do Bus e os Estados dos blocos na cache L1 foram mantidos da primeira parte do projeto (Seção 2).

j) O sinal que representa o Bus será composto pela mensagem escrita pelo processador emissor e pela Tag da posição referenciada na instrução em execução.

2 - Sinais de Estados, Transições, Bus e Write Back - Parte 1

Sinais de Transições:

Transição	Numeração em Binário	Numeração em Decimal
Read Miss (Shared)	000	0
Read Miss (Exclusive)	110	6
Read Hit	001	1
Write Miss	010	2
Write Hit	011	3
Invalidate	100	4

Sinais de Estados:

Estado	Numeração em Binário	Numeração em Decimal
I	00	0
S	01	1
E	10	2
M	11	3

Sinais do Bus:

Mensagens do Bus	Numeração em Binário	Numeração em Decimal
Read Miss	00	0
Write Miss	01	1
Invalidate	10	2

Sinal de Write Back:

write_back = 1	Ocorre write back
write_back = 0	Não ocorre write back

3 - Sinais de Estados, Transições, Bus e Write Back - Parte 2

Sinais das possíveis instruções:

Instrução	Sinal em bits
Read	00
Write	01

Sinais dos processadores:

P0	00
P1	01
P2	10

Valor em binário das posições de memória utilizadas:

10	1010
11	1011
12	1100
13	1101
14	1110

Disposição de bits da instrução:

Instrução na Memória de Instruções	Disposição dos bits [15:0]
Instrução (Read ou Write)	[15:14]
Opcode	[13:12]
Tag	[11:8]
Dado	[7:0]

Sinais da Cache:

Bloco da Cache L1	[15:0]
Tag	[15:12]
Estado	[11:10]
Dado	[9:0]

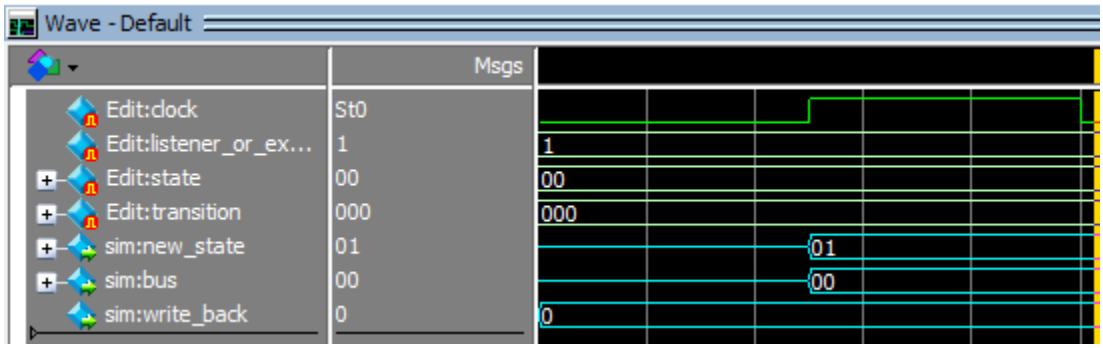
Sinais do Bus:

Bus	Disposição dos bits [15:0]
Mensagem	[15:14]
Tag	[13:10]

4 - Testes - Parte 1

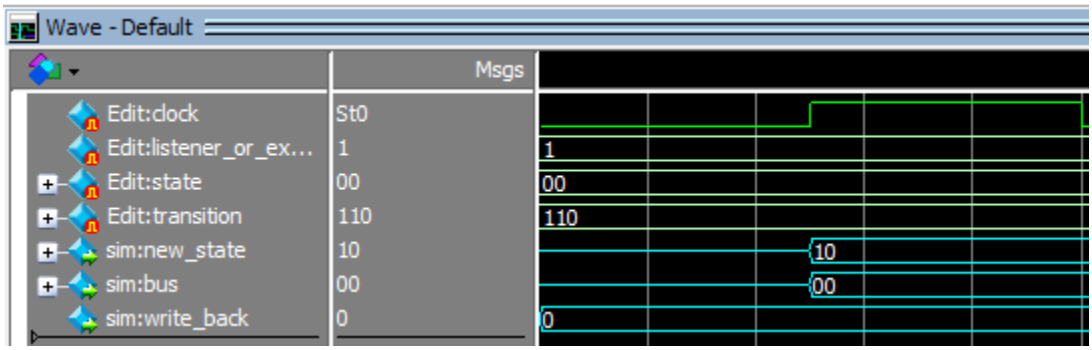
4.1 - Bloco Emissor

a) Invalido / Read Miss Shared



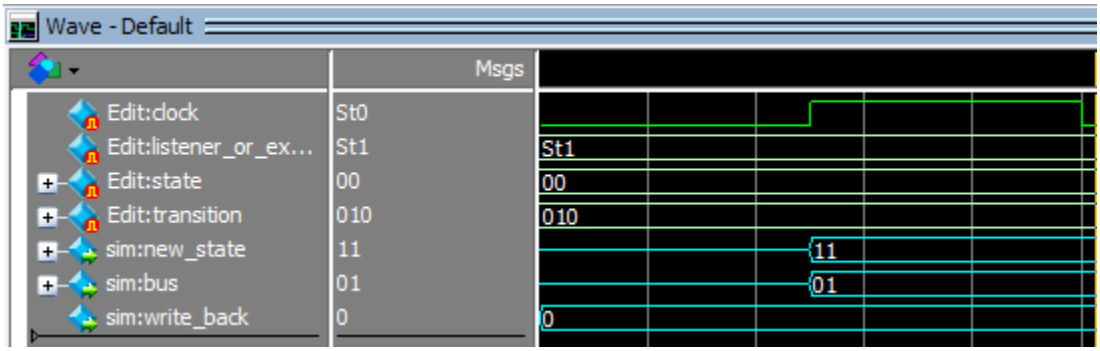
Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco emissor que se encontra no estado inválido e realiza uma transição de Read Miss Shared vai para o estado Shared e escreve Read Miss no Bus. Não ocorre Write Back nessa transição.

b) Invalido / Read Miss exclusive



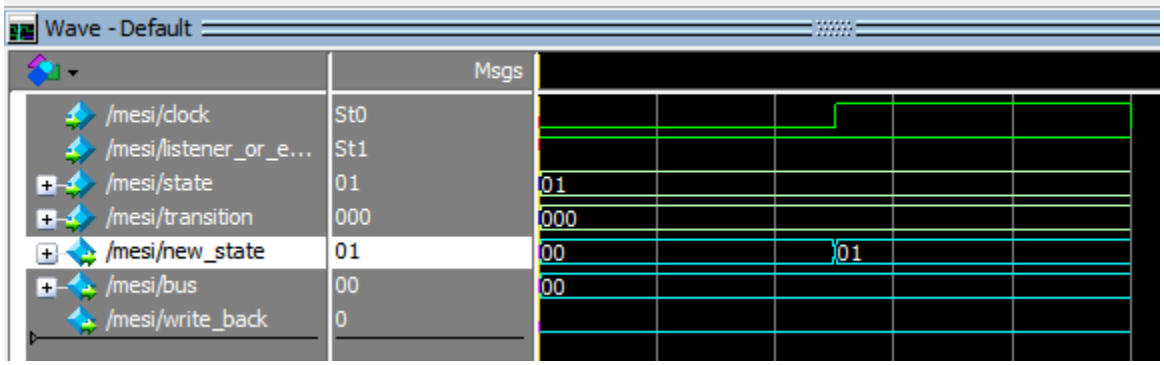
Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco emissor que se encontra no estado inválido e realiza uma transição de Read Miss Exclusive vai para o estado Shared e escreve Read Miss no Bus. Não ocorre Write Back nessa transição.

c) Invalido / Write Miss



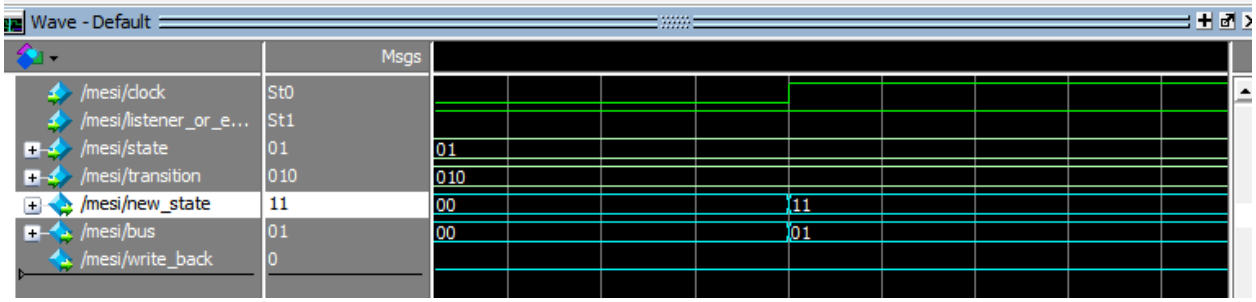
Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco emissor que se encontra no estado inválido e realiza uma transição de Write Miss vai para o estado Shared e escreve Write Miss no Bus. Não ocorre Write Back nessa transição.

d) Shared / Read Miss



Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco emissor que se encontra no estado Shared e realiza uma transição de Read Miss fica no estado Shared e escreve Read Miss no Bus. Não ocorre Write Back nessa transição.

e) Shared / Write Miss

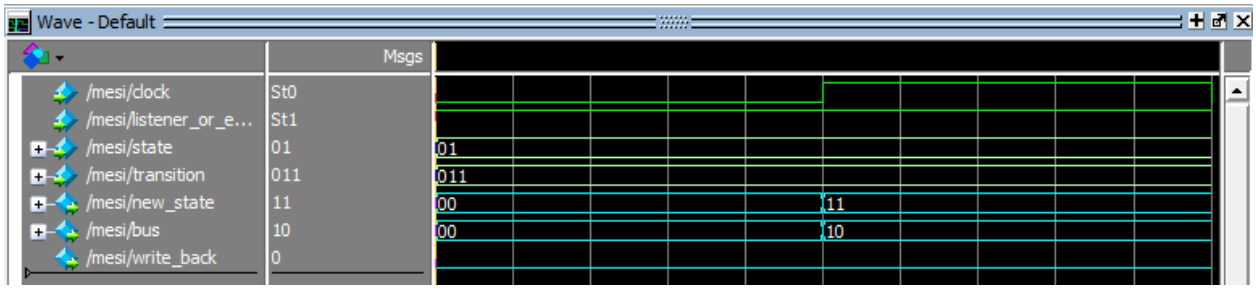


Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco emissor que se encontra no estado Shared e realiza uma transição de Write Miss vai para o estado Modified e escreve Write Miss no Bus. Não ocorre Write Back nessa transição.

f) Shared / Read Hit

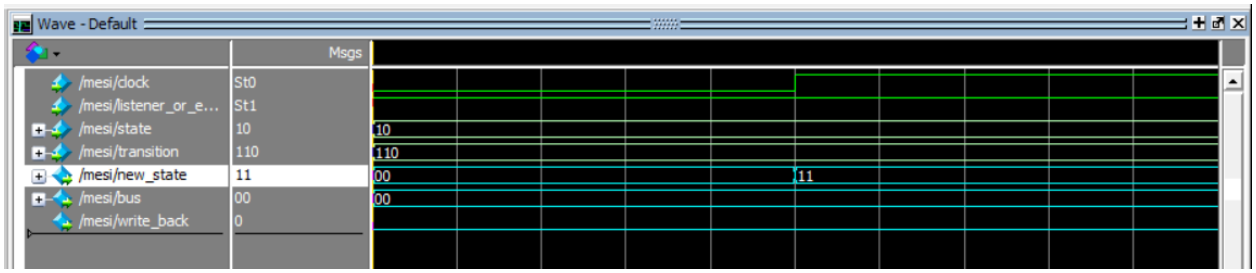
Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco emissor que se encontra no estado Shared e realiza uma transição de Read Hit fica no estado Shared e não escreve no Bus. Não ocorre Write Back nessa transição.

g) Shared / Write Hit



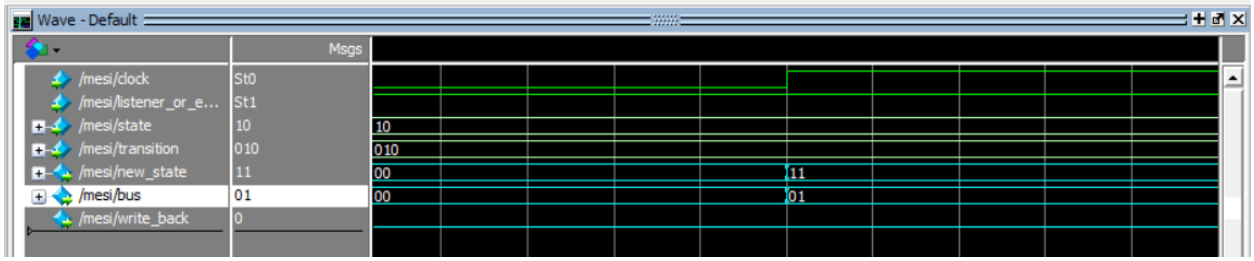
Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco emissor que se encontra no estado Shared e realiza uma transição de Write Hit vai para o estado Modified e escreve Invalidate no Bus. Não ocorre Write Back nessa transição.

h) Exclusive / Read Miss



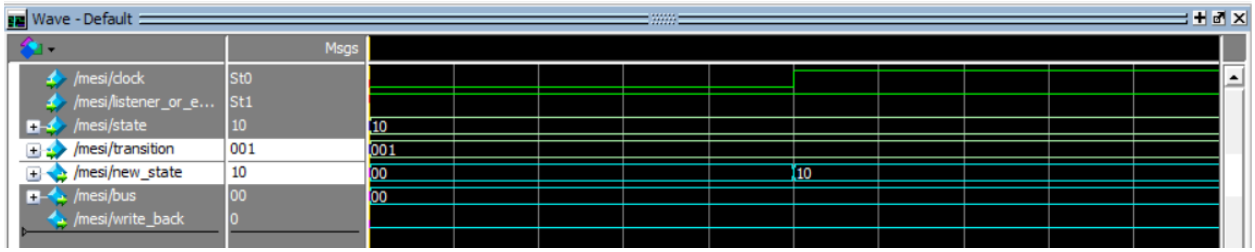
Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco emissor que se encontra no estado Exclusive e realiza uma transição de ReadMiss vai para o estado Modified e escreve Read Miss no Bus. Não ocorre Write Back nessa transição.

i) Exclusive / Write Miss



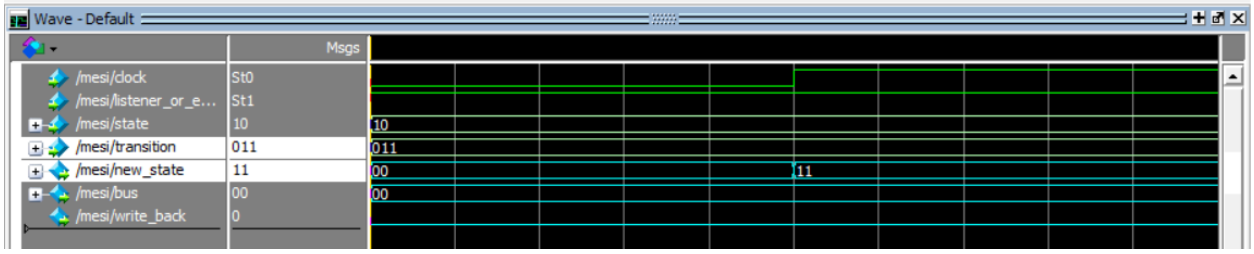
Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco emissor que se encontra no estado Exclusive e realiza uma transição de Write Miss vai para o estado Modificado e escreve Write Miss no Bus. Não ocorre Write Back nessa transição.

j) Exclusive / Read Hit



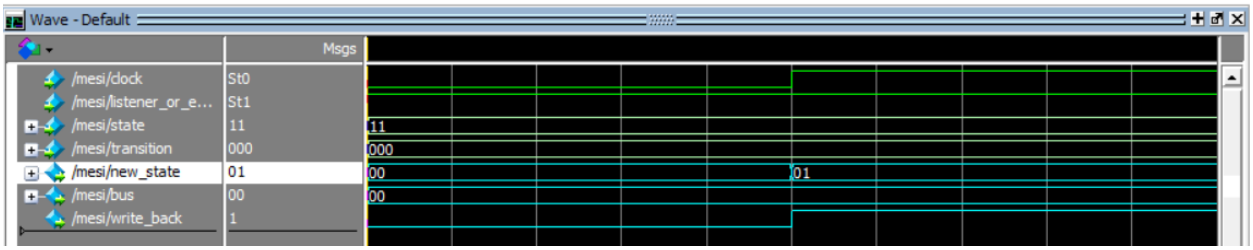
Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco emissor que se encontra no estado Exclusive e realiza uma transição de Read Hit fica no estado Exclusive e não escreve no Bus. Não ocorre Write Back nessa transição.

k) Exclusive / Write Hit



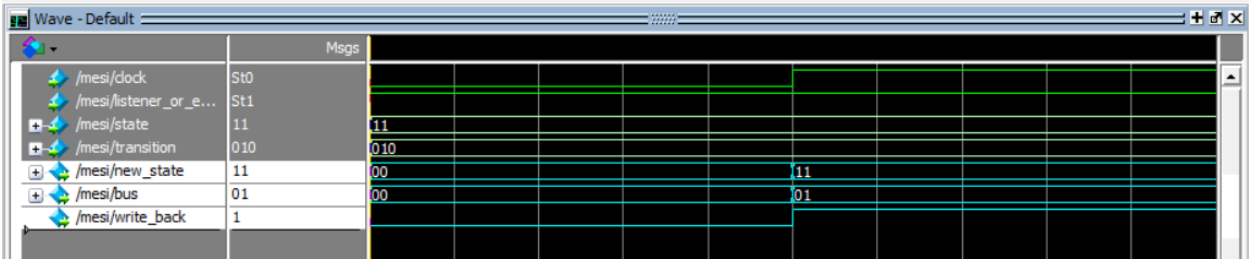
Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco emissor que se encontra no estado Exclusive e realiza uma transição de Write Hit vai para o estado Modified e não escreve no Bus. Não ocorre Write Back nessa transição.

l) Modified / Read Miss



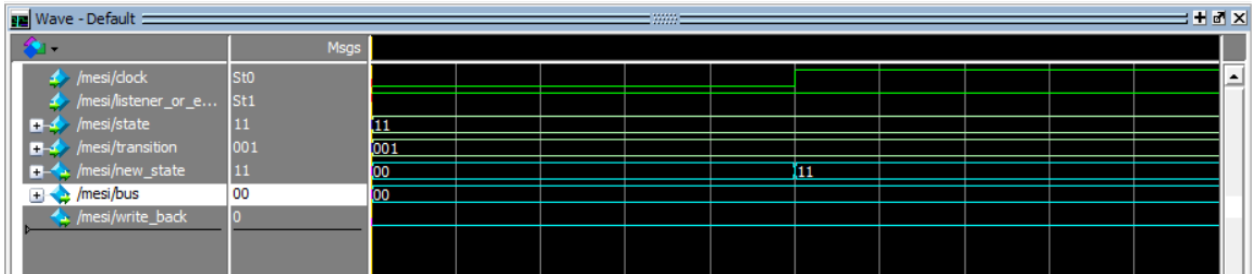
Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco emissor que se encontra no estado Modified e realiza uma transição de Read Miss vai para o estado Shared e escreve Read Miss no Bus. Ocorre Write Back nessa transição.

m) Modified / Write Miss



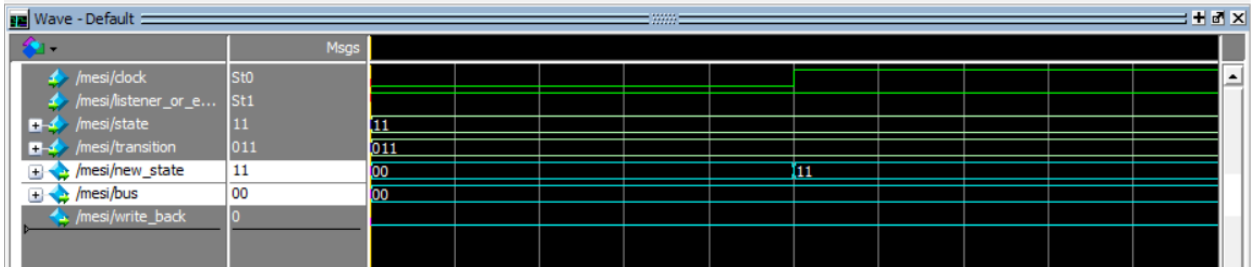
Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco emissor que se encontra no estado Modified e realiza uma transição de Write Miss fica no estado Modified e escreve Write Miss no Bus. Ocorre Write Back nessa transição.

n) Modified / Read Hit



Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco emissor que se encontra no estado Modified e realiza uma transição de Read Hit fica no estado Modified e não escreve no Bus. Não ocorre Write Back nessa transição.

o) Modified / Write Hit



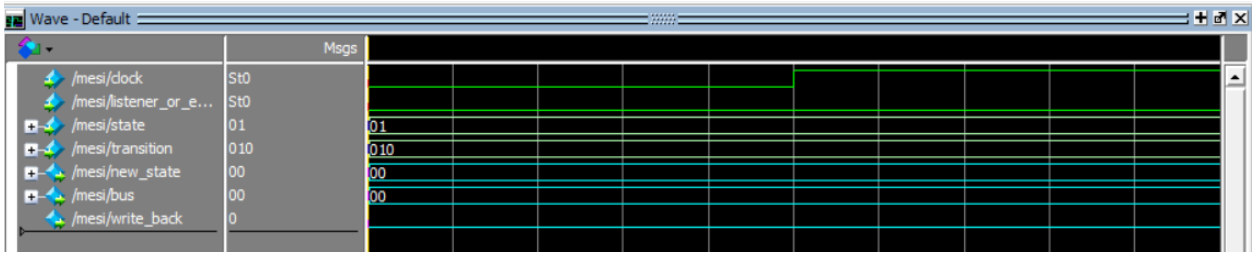
Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco emissor que se encontra no estado Modified e realiza uma transição de Write Hit fica no estado Modified e escreve Write Miss no Bus. Não ocorre Write Back nessa transição.

4.2 - Bloco Ouvinte

a) Shared/ Read Miss

Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco ouvinte que se encontra no estado Shared e realiza uma transição de Read Miss fica no estado Shared e não escreve no Bus. Não ocorre Write Back nessa transição.

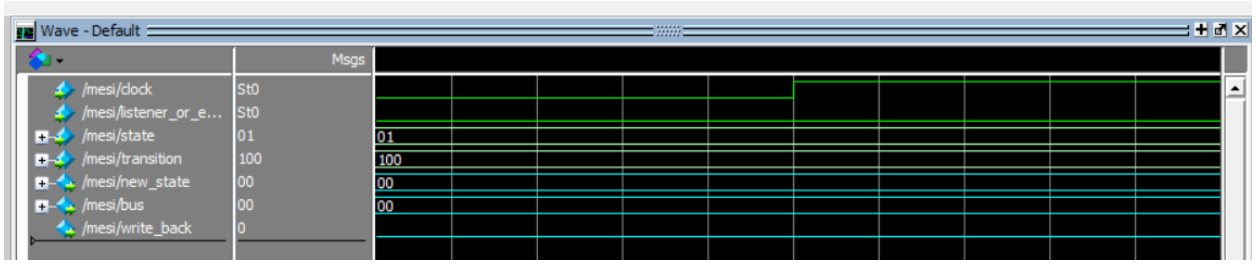
b) Shared/ WriteMiss



Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco ouvinte que se encontra no estado

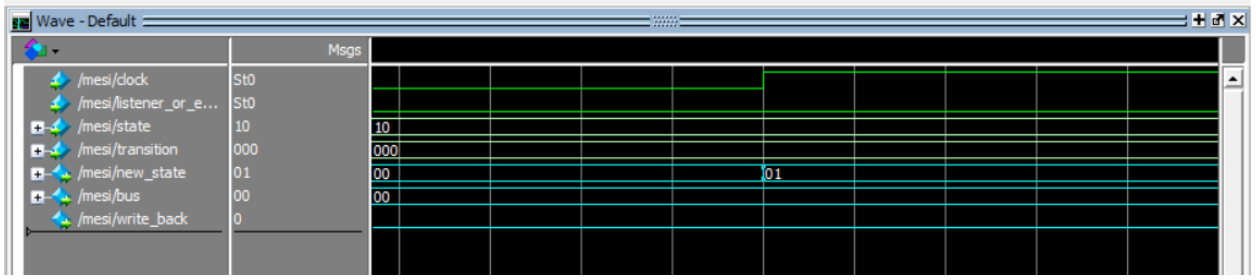
Shared e realiza uma transição de Write Miss vai para o estado Invalido e não escreve no Bus. Não ocorre Write Back nessa transição.

c) Shared/ Invalidate



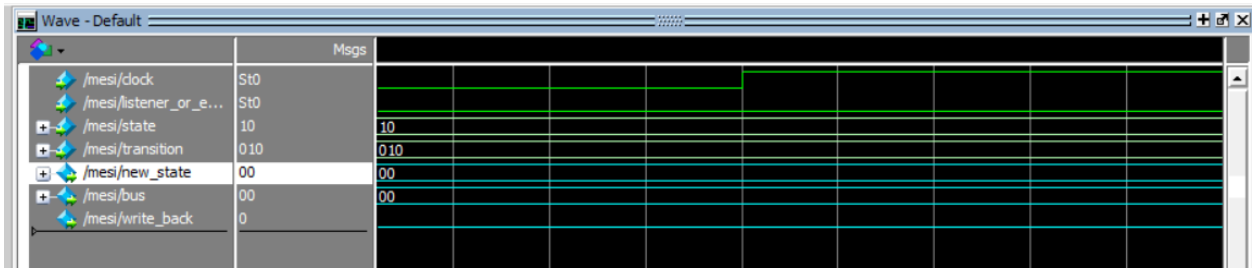
Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco ouvinte que se encontra no estado Shared e realiza uma transição de Invalidate vai para o estado Invalido e não escreve no Bus. Não ocorre Write Back nessa transição.

d) Exclusive/ Read Miss



Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco ouvinte que se encontra no estado Exclusive e realiza uma transição de Read Miss vai para o estado Shared e não escreve no Bus. Não ocorre Write Back nessa transição.

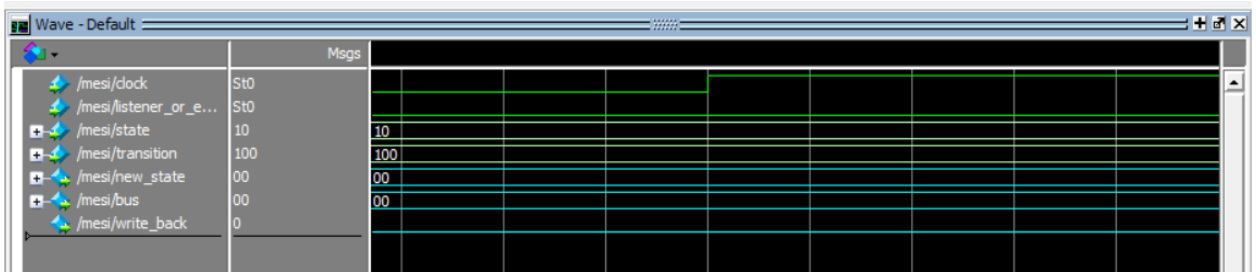
e) Exclusive/ WriteMiss



Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco ouvinte que se encontra no estado

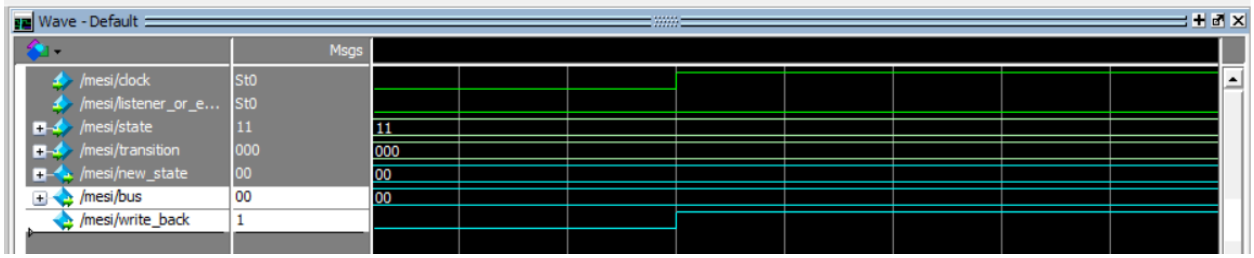
Exclusive e realiza uma transição de Write Miss vai para o estado Inavalido e não escreve no Bus. Não ocorre Write Back nessa transição.

e) Exclusive/ Invalidate



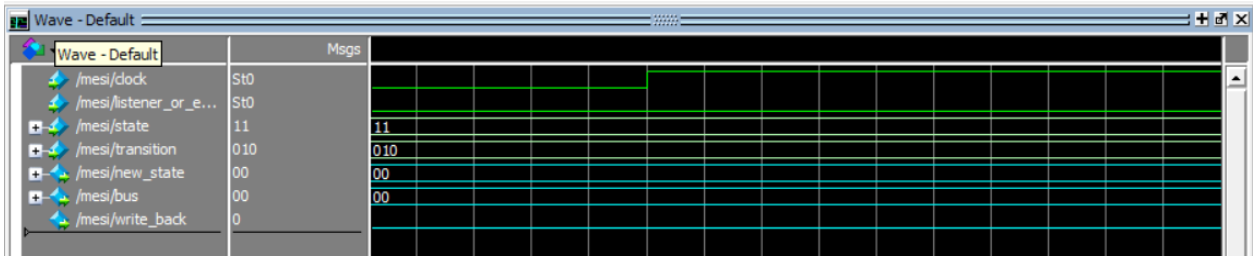
Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco ouvinte que se encontra no estado Exclusive e realiza uma transição de Invalidate vai para o estado Inavalido e não escreve no Bus. Não ocorre Write Back nessa transição.

f) Modified/ Read Miss



Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco ouvinte que se encontra no estado Modified e realiza uma transição de Read Miss vai para o estado Invalido e não escreve no Bus. Ocorre Write Back nessa transição.

g) Modified/ WriteMiss



Analisando a simulação podemos perceber que a transição ocorre corretamente, assim como apresentado na máquina de estados. Um bloco ouvinte que se encontra no estado

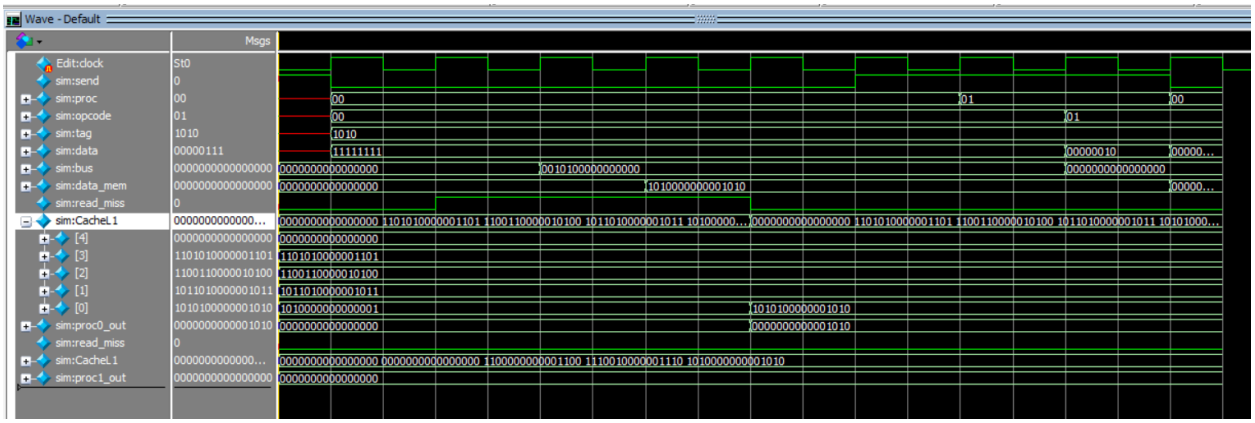
Modified e realiza uma transição de Write Miss vai para o estado Invalido e não escreve no Bus. Ocorre Write Back nessa transição.

5 - Testes - Parte 2

Todos os testes foram realizados levando em consideração a inicialização da cache descrita pelo arquivo txt “Inicialização Caches e Memória.txt” contido dentro da pasta do projeto.

a) P0 read 10

Simulação:



Ciclo 1: No primeiro ciclo ocorre o despacho da instrução P0 read 10, podemos perceber isso analisando os sinais de *proc, opcode, tag* e *data,* que são os sinais de saída da memória de instruções.

Ciclo 2: No segundo ciclo a instrução é recebida pelos processadores, e como a posição 10 está inválida em P0, podemos perceber a ativação do sinal que indica a ocorrência do Read Miss, indicado pelo sinal *read_miss.* Além disso, neste ciclo ocorre também a escrita no Bus por parte do processador emissor.

Ciclo 3: No terceiro ciclo o Bus recebe a mensagem escrita pelo processador emissor no ciclo 2, essa mensagem só é recebida no ciclo 3 pois ela deve passar pelo arbiter antes de ser colocada no Bus. Nesse mesmo ciclo, a memória e os demais processadores lêem a informação colocada no Bus.

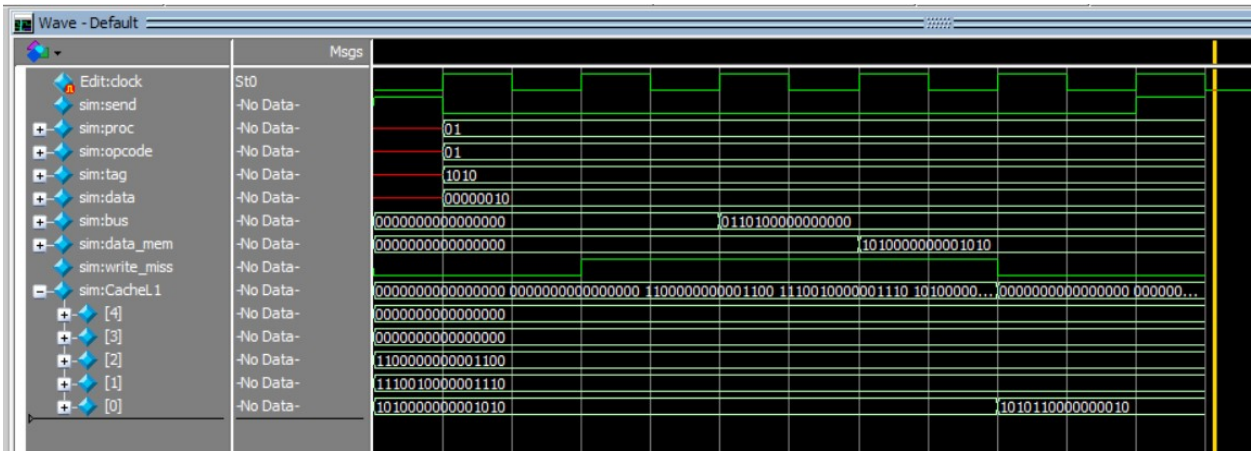
Ciclo 4: No ciclo quatro a memória retorna o dado referente a tag referenciada na instrução, e os demais processadores retornam informações dizendo se possuem um bloco com mesma tag, e caso possuam um bloco com mesma tag e modificado, retornam o bloco também. Analisando o

sinal *data_mem* podemos observar um dado chegando da memória neste ciclo. Como nenhum outro processador possui a tag 10 modificada, então o dado retornado da memória é utilizado.

Ciclo 5: No ciclo cinco, a informação trazida da memória é atualizada na Cache L1 de P0, e então o dado se torna disponível para leitura. Portanto, se analisarmos o sinal *proc0_out* podemos perceber que o valor correto é retornado para a saída do processador P0.

b) P1 write 2

Simulação:



Ciclo 1: No primeiro ciclo ocorre o despacho da instrução P1 write 10, podemos perceber isso analisando os sinais de *proc, opcode, tag* e *data,* que são os sinais de saída da memória de instruções.

Ciclo 2: No segundo ciclo a instrução é recebida pelos processadores, e como a posição 10 está inválida em P1, podemos perceber a ativação do sinal que indica a ocorrência do Write Miss, indicado pelo sinal *write_miss.* Além disso, neste ciclo ocorre também a escrita no Bus por parte do processador emissor.

Ciclo 3: No terceiro ciclo o Bus recebe a mensagem escrita pelo processador emissor no ciclo 2, essa mensagem só é recebida no ciclo 3 pois ela deve passar pelo arbiter antes de ser colocada no Bus. Nesse mesmo ciclo, a memória e os demais processadores lêem a informação colocada no Bus.

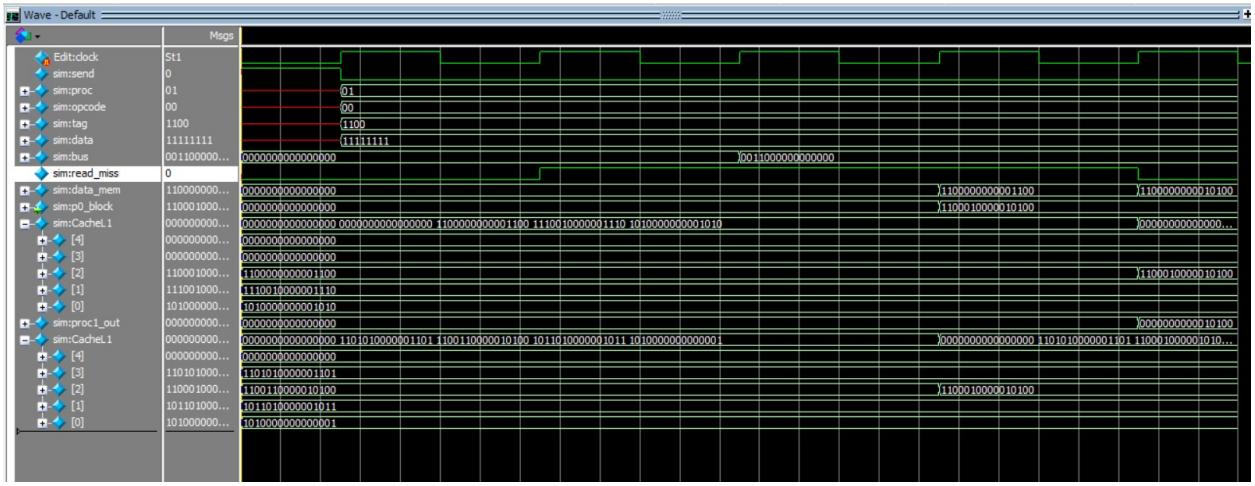
Ciclo 4: No ciclo quatro a memória retorna o dado referente a tag referenciada na instrução, e os demais processadores retornam informações dizendo se possuem um bloco com mesma tag, e caso possuam um bloco com mesma tag e modificado, retornam o bloco também. Analisando o

sinal *data_mem* podemos observar um dado chegando da memória neste ciclo. Como nenhum outro processador possui a tag 10 modificada, então o dado retornado da memória é utilizado.

Ciclo 5: No ciclo cinco, a informação trazida da memória fica disponível para o processador P1. Sendo assim, para finalizar a execução o processador P1 escreve na Cache L1 na posição referente a Tag 10 o valor 2.

c) P1 read 12

Simulação:



Ciclo 1: No primeiro ciclo ocorre o despacho da instrução P1 read 12, podemos perceber isso analisando os sinais de *proc, opcode, tag* e *data,* que são os sinais de saída da memória de instruções.

Ciclo 2: No segundo ciclo a instrução é recebida pelos processadores, e como a posição 12 está inválida em P1, podemos perceber a ativação do sinal que indica a ocorrência do Read Miss, indicado pelo sinal *read_miss.* Além disso, neste ciclo ocorre também a escrita no Bus por parte do processador emissor.

Ciclo 3: No terceiro ciclo o Bus recebe a mensagem escrita pelo processador emissor no ciclo 2, essa mensagem só é recebida no ciclo 3 pois ela deve passar pelo arbiter antes de ser colocada no Bus. Nesse mesmo ciclo, a memória e os demais processadores lêem a informação colocada no Bus.

Ciclo 4: No ciclo quatro a memória retorna o dado referente a tag referenciada na instrução, e os demais processadores retornam informações dizendo se possuem um bloco com mesma tag, e caso possuam um bloco com mesma tag e modificado, retornam o bloco também. Analisando o

sinal *data_mem* podemos observar um dado chegando da memória neste ciclo. Como nesse caso o processador P0 possui a tag 12 modificada, ele também envia o bloco para P1, podemos analisar isso observando o sinal *p0_block*. Tanto *data_mem* quanto *p0_block* chegam neste ciclo. Podemos observar também, analisando os últimos sinais da parte de baixo da foto, que a Cache L1 de P0 muda o estado da tag 12 para Shared assim que ele envia esse bloco para o processador P1.

Ciclo 5: No ciclo cinco, podemos perceber que o bloco com a tag 12 é atualizado na Cache L1 de P1 utilizando o bloco que veio de P0. Além disso, uma vez que o dado fica pronto na Cache L1 de P1, ele é retornado para a saída do processador, percebemos isso analisando *proc1_out*.

6 - Considerações finais

A parte 1 foi bem sucedida, uma vez que conseguimos implementar a máquina de estados de forma correta, conforme as fotos passadas no moodle. Já em relação a parte 2 não foi possível realizar todos os testes. A implementação da parte 2 foi concluída e alguns testes foram executados, no entanto, a lógica de despacho das instruções não executou corretamente, atrapalhando a execução dos demais testes. Um próximo passo para o projeto seria testar e ajustar o despacho das instruções e realizar os demais testes para testar o funcionamento total do código.

7 - Referências

J.L. Hennessy & D. A. PATTERSON. Computer Architecture: A Quantitative Approach (5th edition). Morgan Kaufmann, 2012.