

Prática 4

Erick Henrique
Marina Bernardes

09/2021

—
Laboratório de Arquitetura e
Organização de Computadores II

2021.1

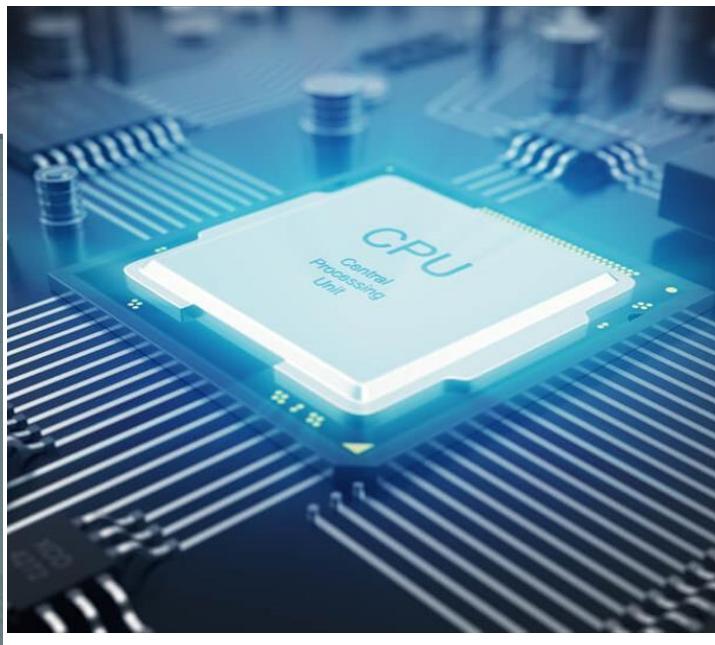
—

Daniela Cristina Cascini Kupsch

Centro Federal de Educação Tecnológica de Minas
Gerais

OBJETIVOS

Esta prática tem a finalidade de exercitar os conceitos relacionados ao protocolo Snooping.



O PROCESSO

A prática 4 consistiu primeiramente em projetar as máquinas de estados do protocolo MSI Snooping utilizando Verilog. Após essa implementação, e considerando o protocolo MSI de coerência de cache, implementar um projeto de contenha:

1. Três CPUs, que realizam operações de leitura e escrita nas caches.
2. Três Caches, ou seja, uma cache L1 para cada CPU.
3. E uma memória compartilhada pelas CPUs.

Introdução

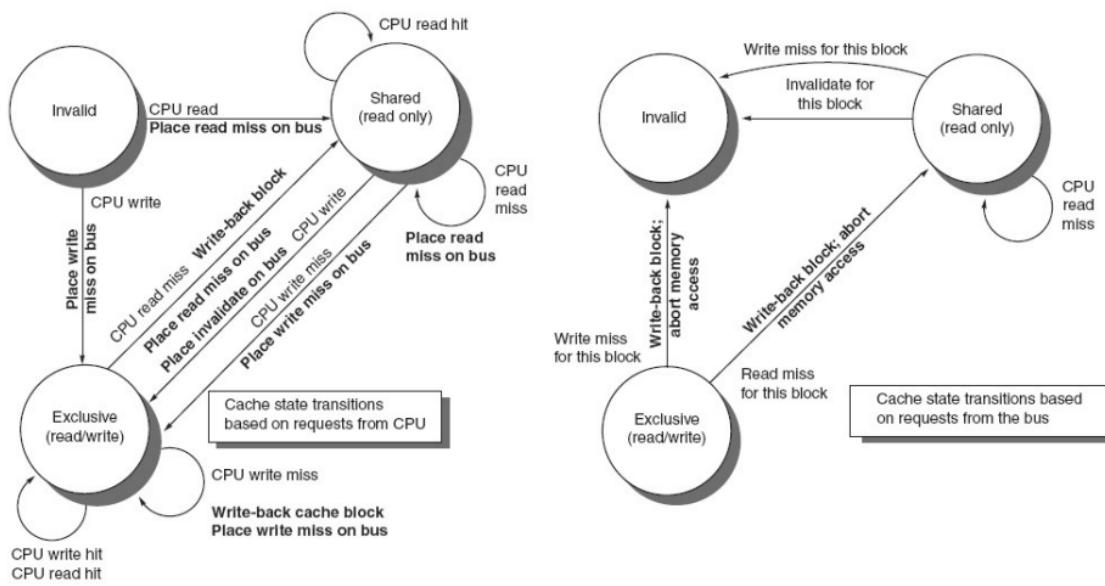
Snooping é um protocolo de coerência de caches, em que cada cache tem a cópia dos dados de um bloco de memória física que pode rastrear o estado de compartilhamento de dados do bloco, ou seja, cada processador bisbilhota o endereço colocado no barramento.

Parte I

Desenvolvimento do projeto

Para a implementação do algoritmo foi seguido o projeto presente nos slides da aula teórica.

Protocolo de coerência por snooping



Logo, implementamos duas máquinas:

- Transição de estados da cache baseada na requisição da CPU
- Transição de estados da cache baseada na requisição do BUS

Para ambos os módulos utilizamos a função case do verilog, passando por todos os estados: Exclusive, Invalid e Shared.

Sendo assim, temos que:

Máquina 1 - MaquinaEstadoCPU

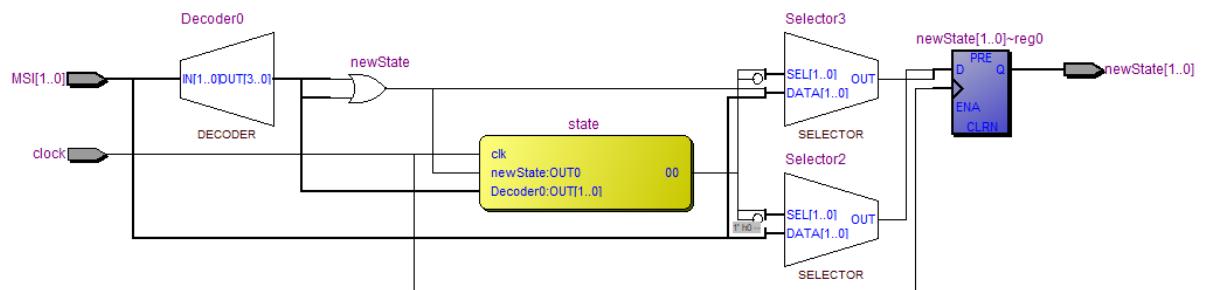
Estados do processador	
invalid	2^3 00
exclusive	2^3 01
shared	2^3 10

Possíveis operações	
read miss	2'boo
write miss	2'bo1
read hit	2'b10
write hit	2'b11

Mensagens do processador	
Place Read Miss on Bus	2'boo
Place Write Miss on Bus	2'bo1
Place Invalidate on Bus	2'b10
Empty Message	2'b11

Ações do processador	
Write Back Block	2'boo
Write Back Cache Block	2'bo1
Empty Action	2'b11

Interconexões



Simulação

- Caso Teste

Para a simulação do protocolo implementado, testamos todas as alternativas de cada máquina.

- Implementação

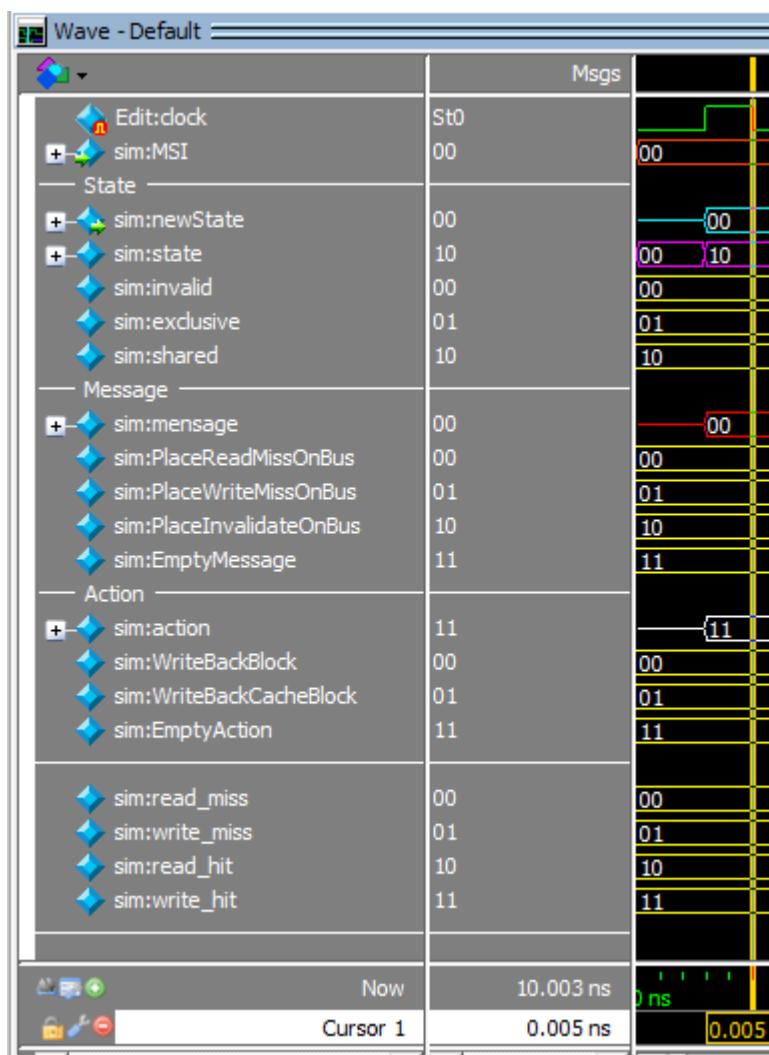
Para a simulação da transição de estados 1 (requisito da CPU):

1. Invalid -> read

Estado destino: Shared

Mensagem esperada: Place read miss on bus

Ação Esperada: Nenhuma ação



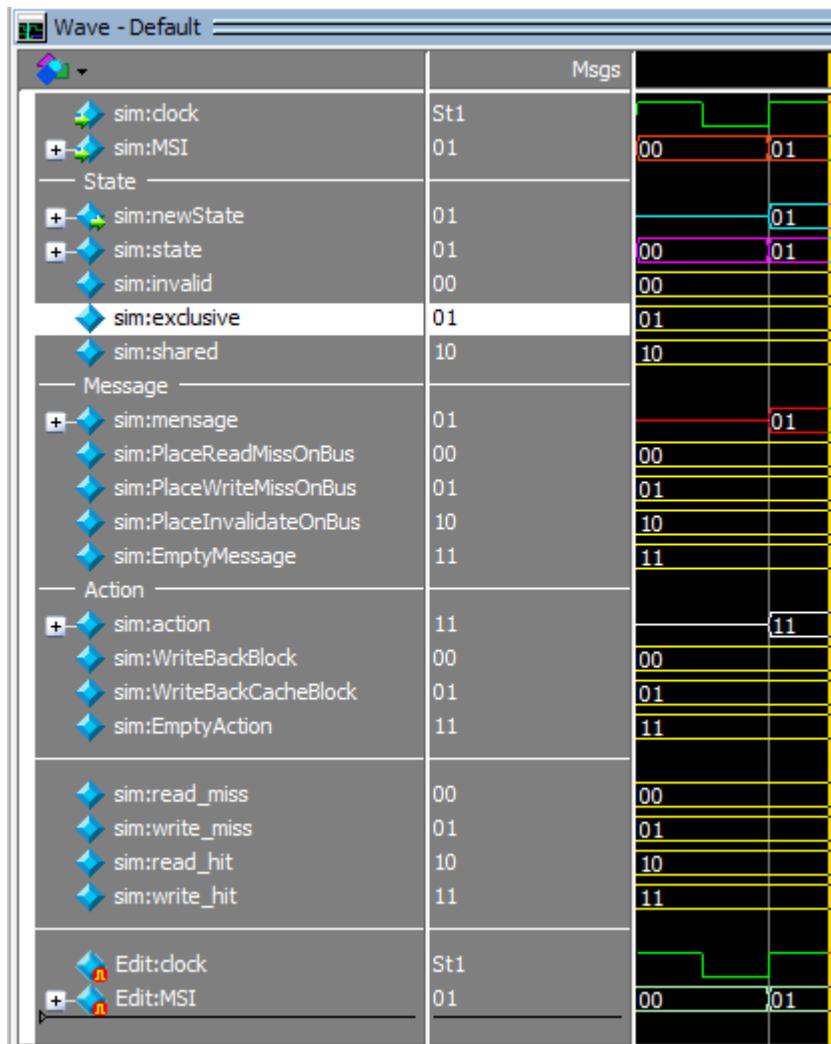
Executado com sucesso.

2. Invalid -> write

Estado destino: Exclusive

Mensagem esperada: Place write miss on bus

Ação Esperada: Nenhuma ação



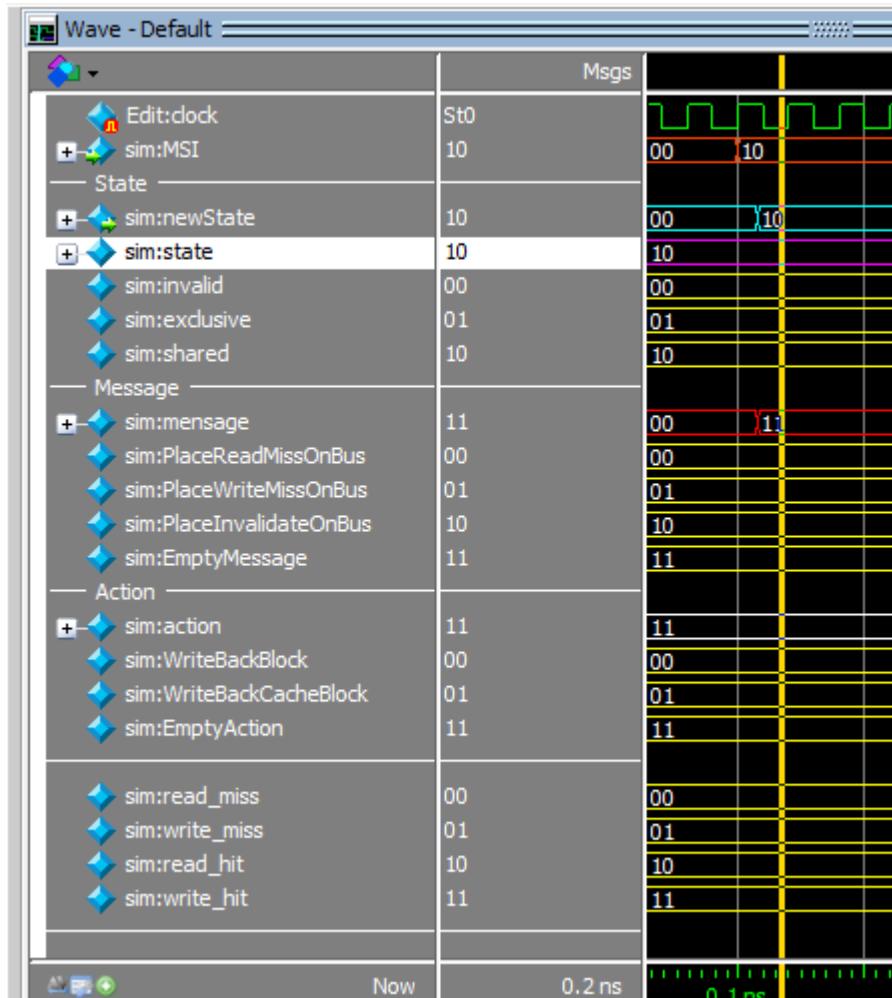
Executado com sucesso.

3. Shared -> read hit

Estado destino: Shared

Mensagem esperada: Nenhuma mensagem

Ação Esperada: Nenhuma ação



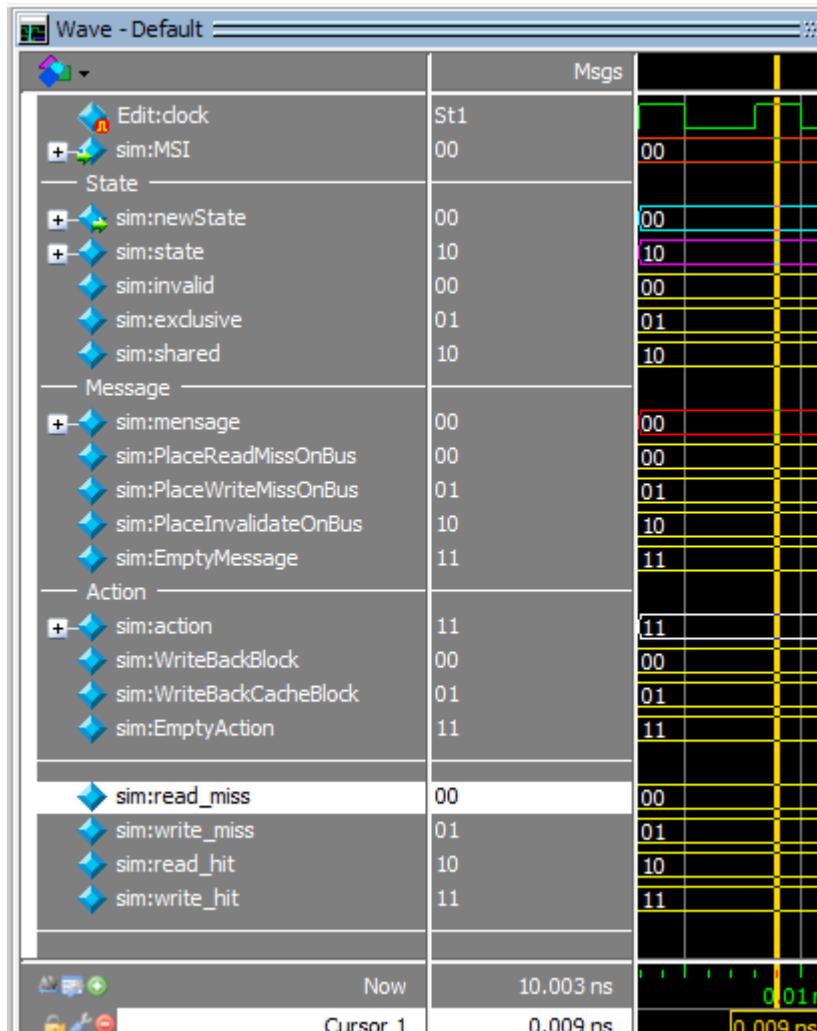
Executado com sucesso.

4. Shared -> read miss

Estado destino: Shared

Mensagem esperada: Place read miss on bus

Ação Esperada: Nenhuma ação



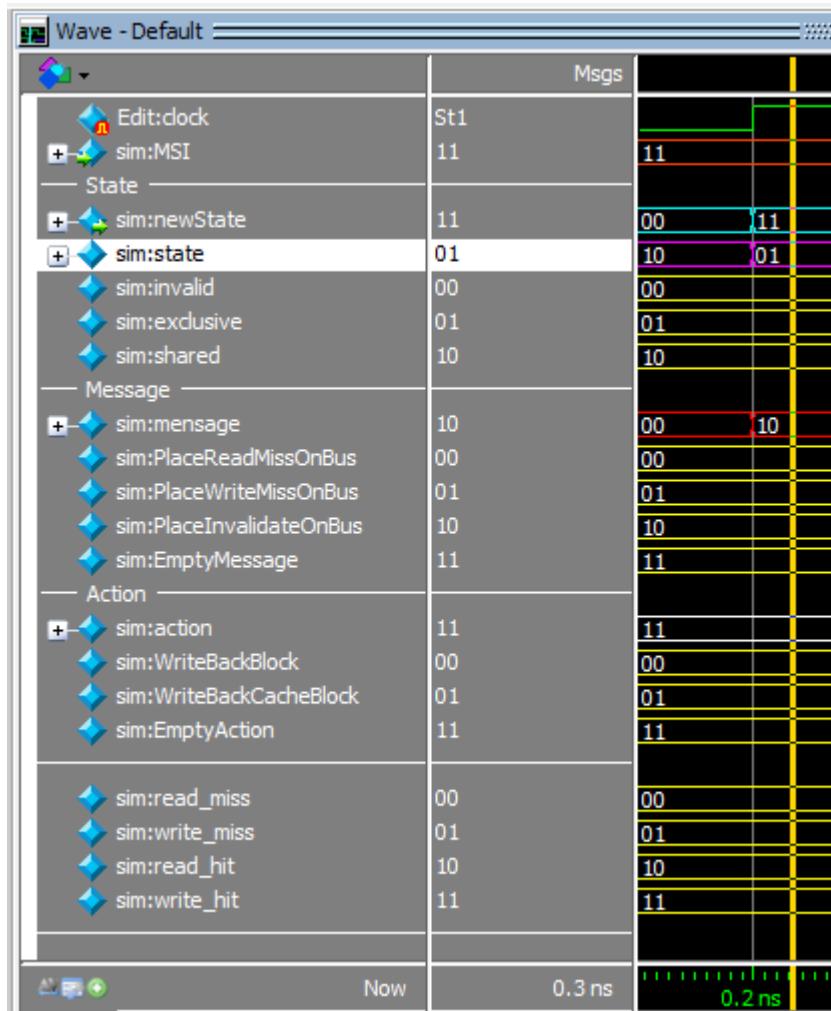
Executado com sucesso.

5. Shared -> write hit

Estado destino: Exclusive

Mensagem esperada: Place invalidate on bus

Ação Esperada: Nenhuma ação



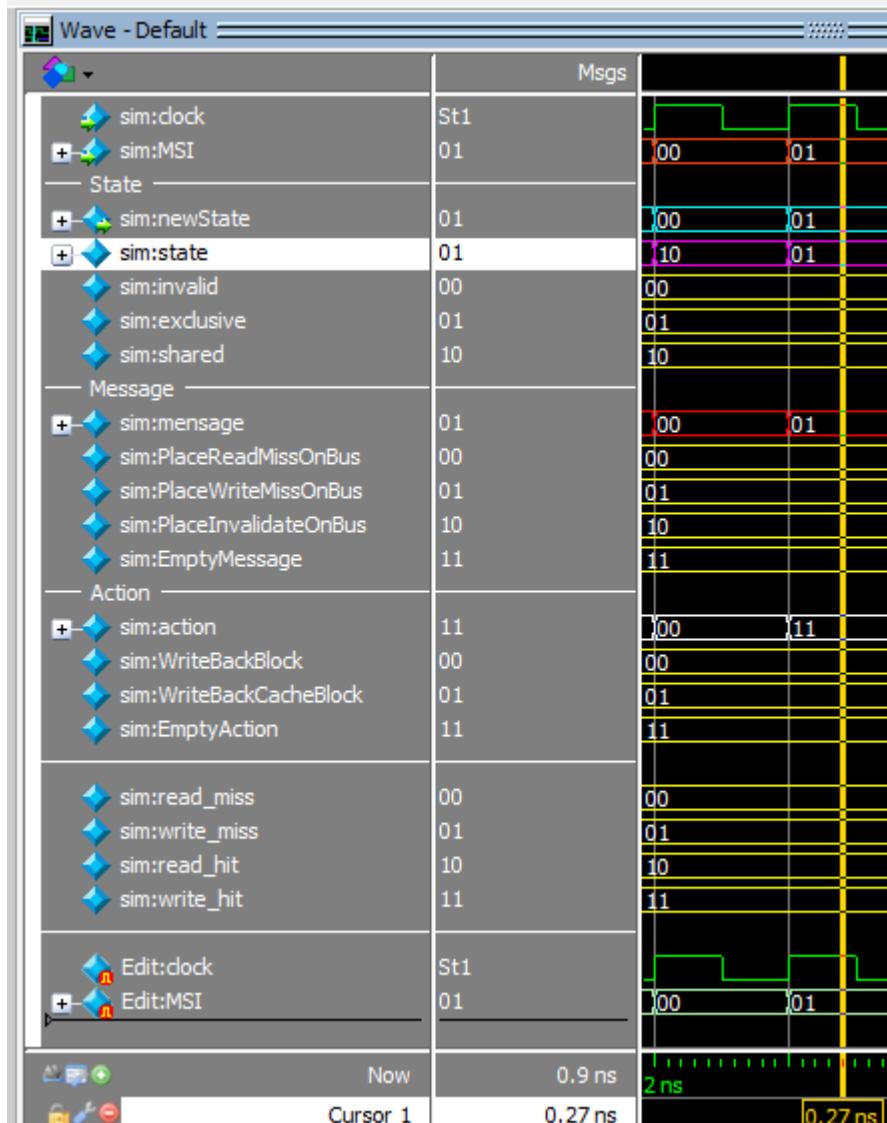
Executado com sucesso.

6. Shared -> write miss

Estado destino: Exclusive

Mensagem esperada: Place write miss on bus

Ação Esperada: Nenhuma ação



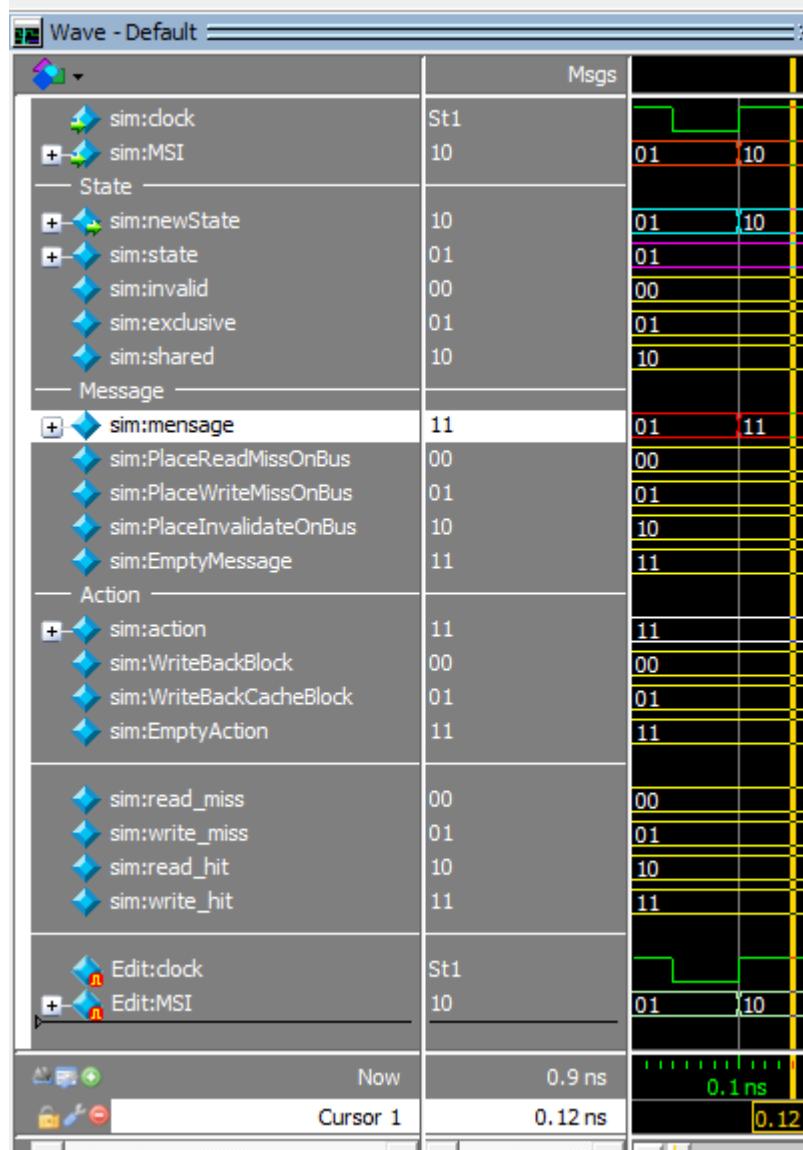
Executado com sucesso.

7. Exclusive -> read hit

Estado destino: Exclusive

Mensagem esperada: Nenhuma mensagem

Ação Esperada: Nenhuma ação



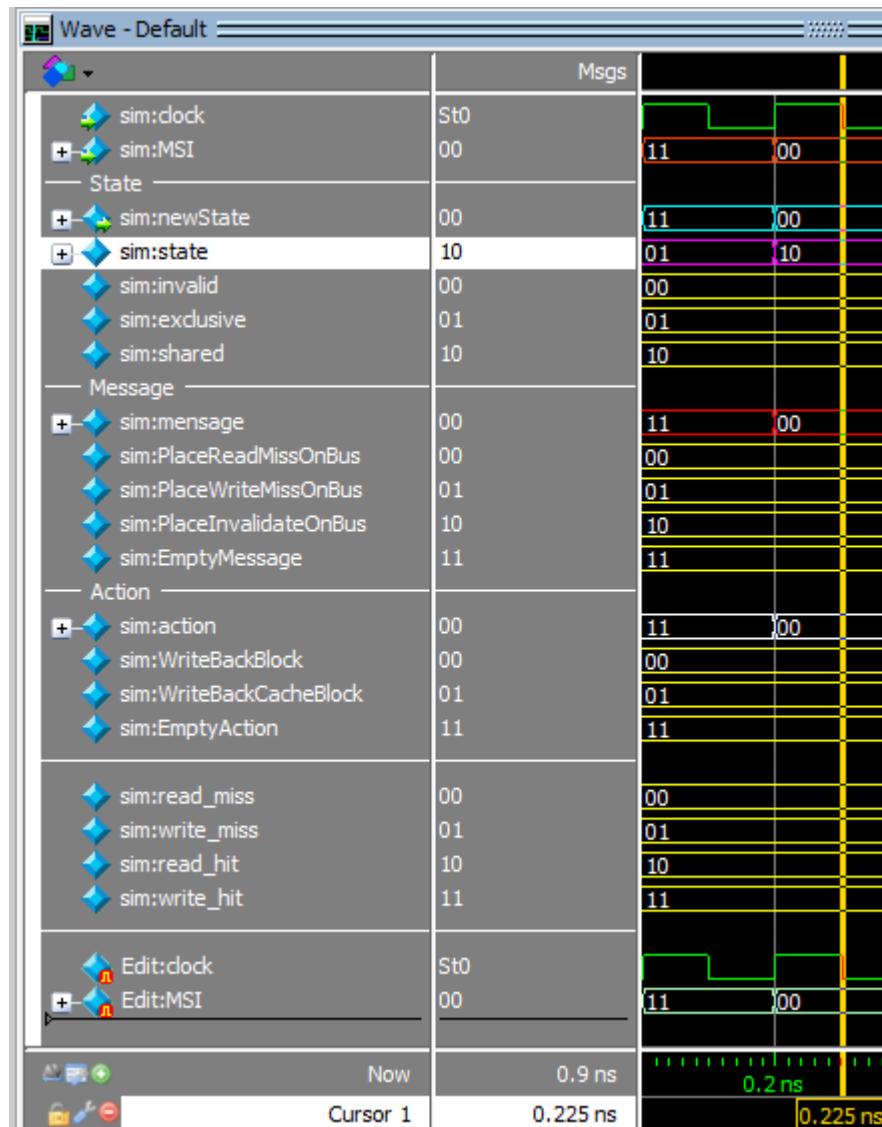
Executado com sucesso.

8. Exclusive -> read miss

Estado destino: Shared

Mensagem esperada: Place read miss on bus

Ação Esperada: Write-back block



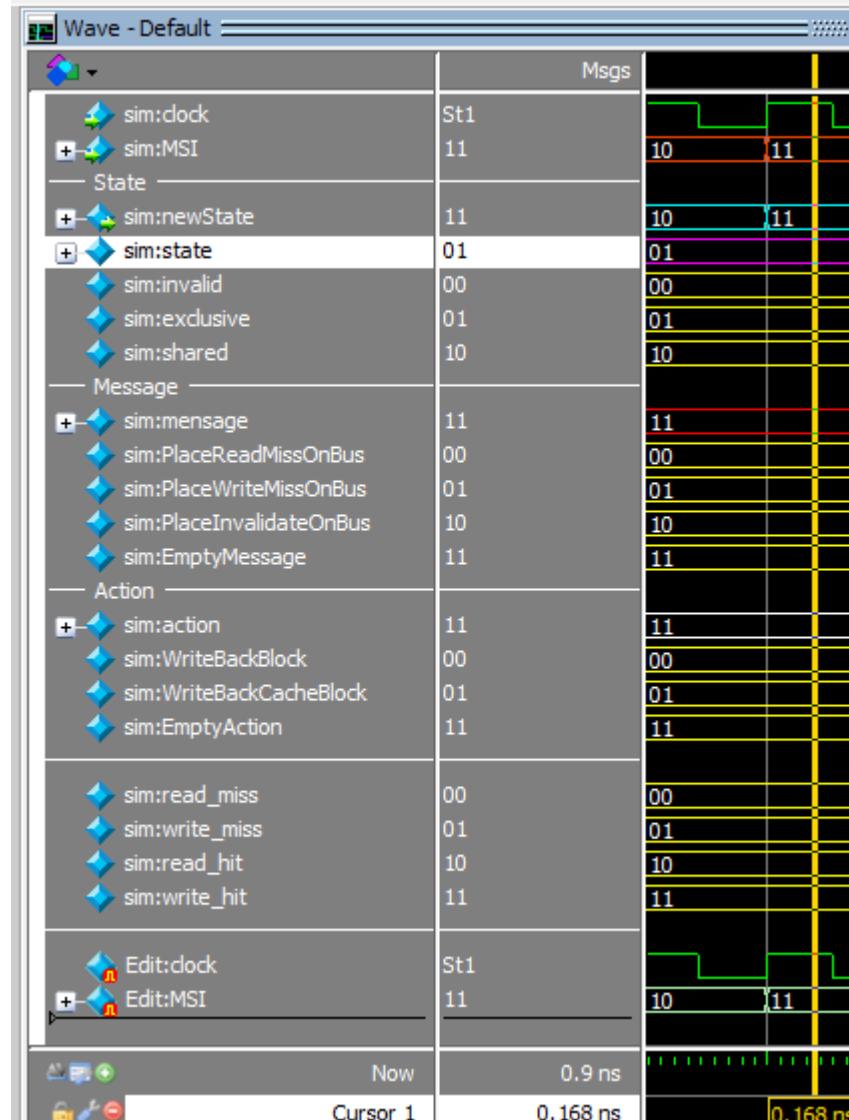
Executado com sucesso.

9. Exclusive -> write hit

Estado destino: Exclusive

Mensagem esperada: Nenhuma mensagem

Ação Esperada: Nenhuma ação



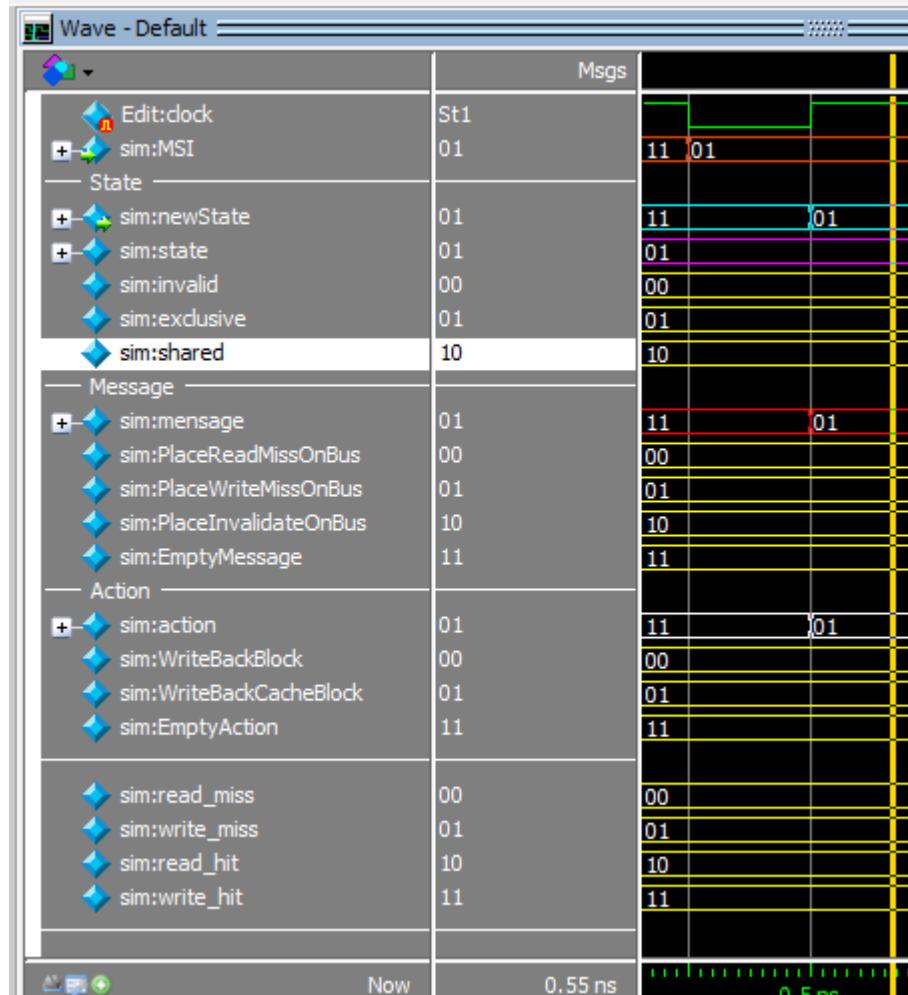
Executado com sucesso.

10. Exclusive -> write miss

Estado destino: Exclusive

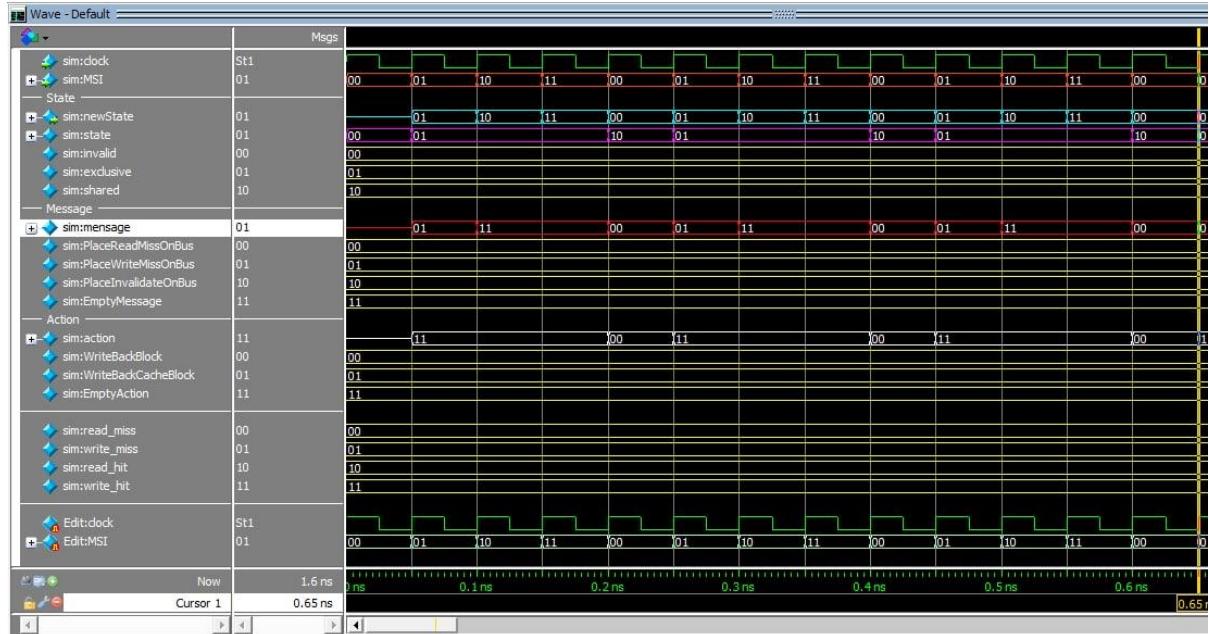
Mensagem esperada: Place write miss on bus

Ação Esperada: Write-back cache block



Executado com sucesso.

Em uma simulação com muitos ciclos de clock, observamos em 'state':



Começamos no estado inválido (00) e passamos para o estado exclusivo (01), devido ao contador MSI, ocasionando CPU Write. A mensagem enviada é Place Write Miss on Bus, sem nenhuma ação. Continuamos no exclusivo, devido ao contador MSI, deu Read Hit. Não houve mensagem e nem ação. Ainda no estado exclusivo, ocorre Write Hit, sem mensagem nem ação.

Após isso, o estado exclusivo dá Read Miss e o estado muda para Shared. A mensagem é Place Read Miss on Bus e a ação é Write Back Block.

De Shared vamos para Exclusivo, com Write Miss. A mensagem é Place Write Miss on Bus, sem nenhuma ação. Ainda no exclusivo, devido ao Read Hit, sem nenhuma mensagem ou ação. Continuamos no mesmo estado, porém com Write Hit, sem mensagem ou ação.

Analizando a imagem do protocolo, podemos perceber que a máquina irá transitar de Shared para Exclusivo para sempre. O que corresponde ao correto funcionamento de nossa implementação, visto que tudo ocorreu conforme o planejado.

Máquina 2 - MaquinaEstadoBUS

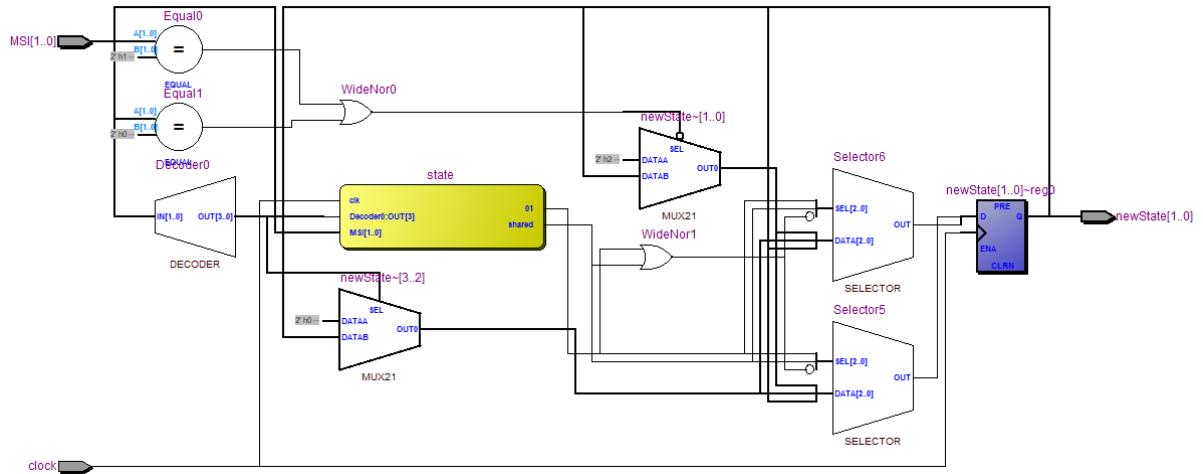
Estados do processador	
invalid	2'boo
exclusive	2'bo1
shared	2'b10

Possíveis operações	
read miss	2'boo
write miss	2'bo1
invalidate	2'b10

Mensagens do processador	
Read Miss For This Block	2'boo
CPU Read Miss	2'bo1
Write Miss For This Block	2'b10
Invalidate For This Block	2'b11

Ações do processador	
Write Back Block:Abort Memory Access	1'bo
Empty Action	1'b1

Interconexões



Simulação

- Caso Teste

Para a simulação do protocolo implementado, testamos todas as alternativas de cada máquina.

- Implementação

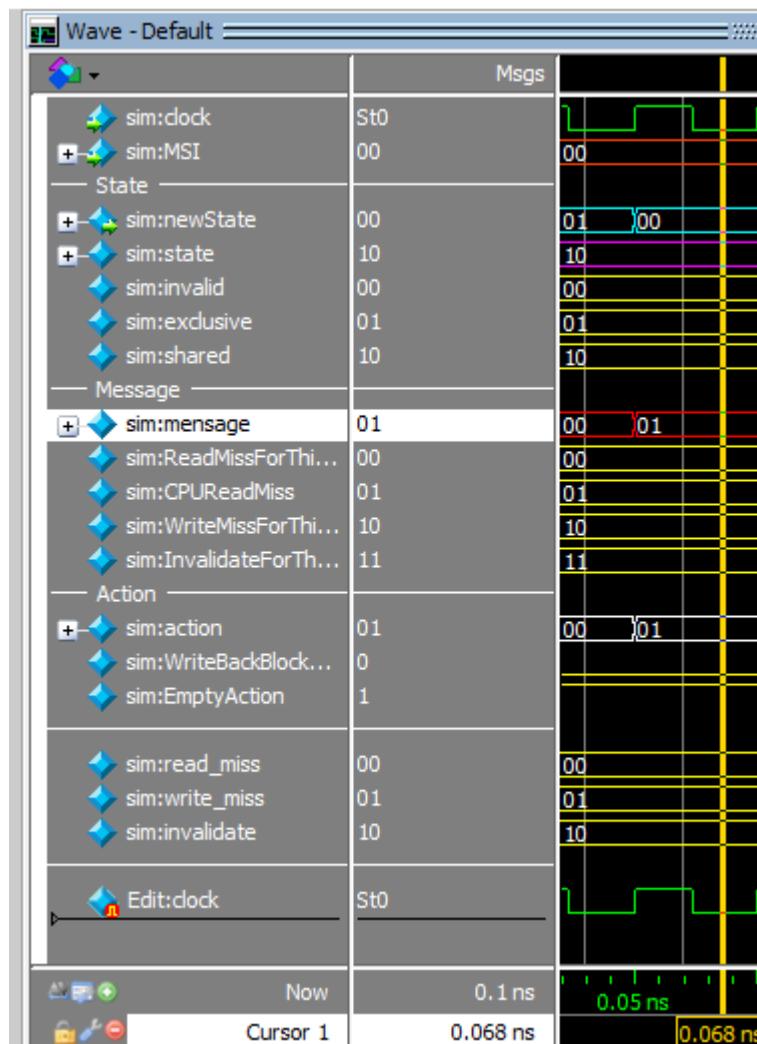
Para a simulação da transição de estados 2 (requisito da BUS):

1. Shared -> read miss

Estado destino: Shared

Mensagem de controle: CPU read miss

Ação Esperada: Nenhuma ação



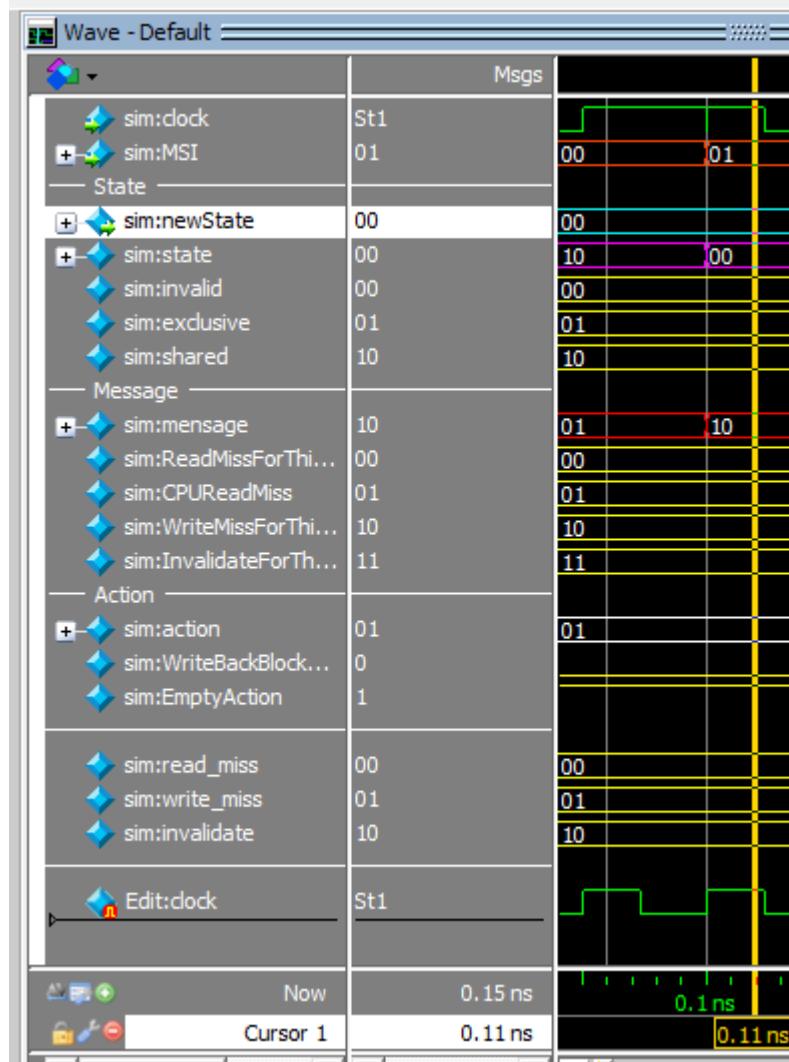
Executado com sucesso.

2. Shared -> write miss

Estado destino: Invalid

Mensagem de controle: Write miss for this block

Ação Esperada: Nenhuma ação



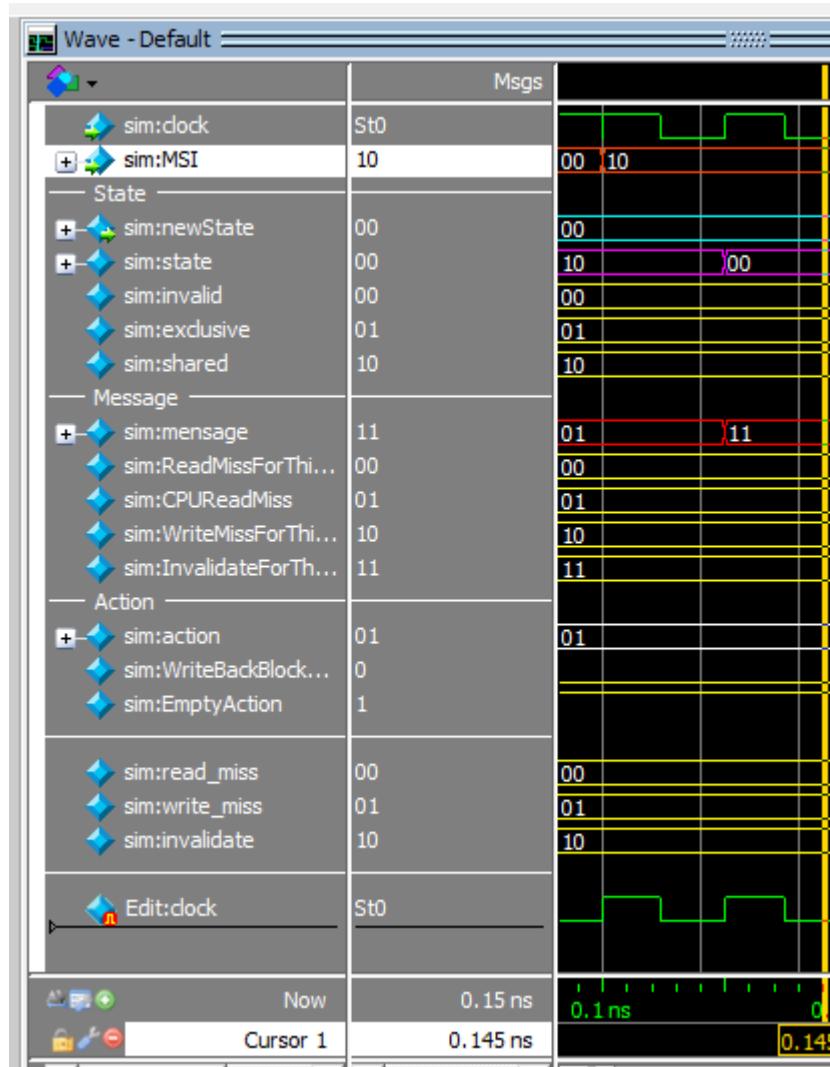
Executado com sucesso.

3. Shared -> invalidate

Estado destino: Invalid

Mensagem de controle: Invalidate for this block

Ação Esperada: Nenhuma ação



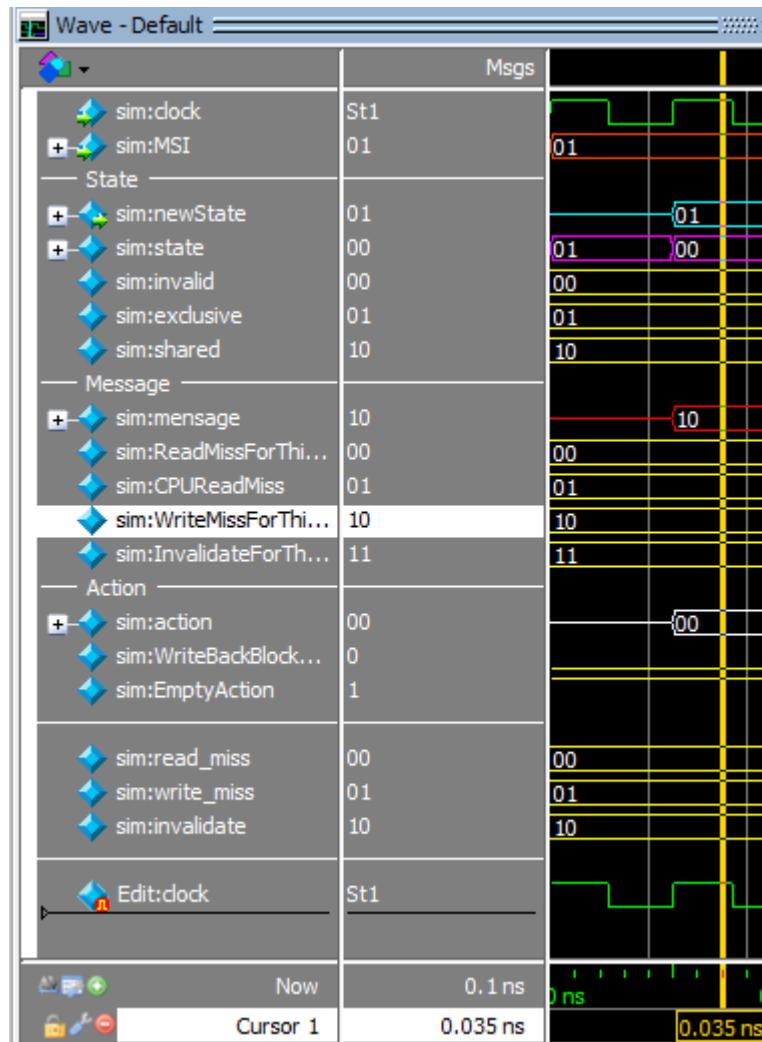
Executado com sucesso.

4. Exclusive -> write miss

Estado destino: Invalid

Mensagem de controle: Write miss for this block

Ação Esperada: Write-back block; abort memory access



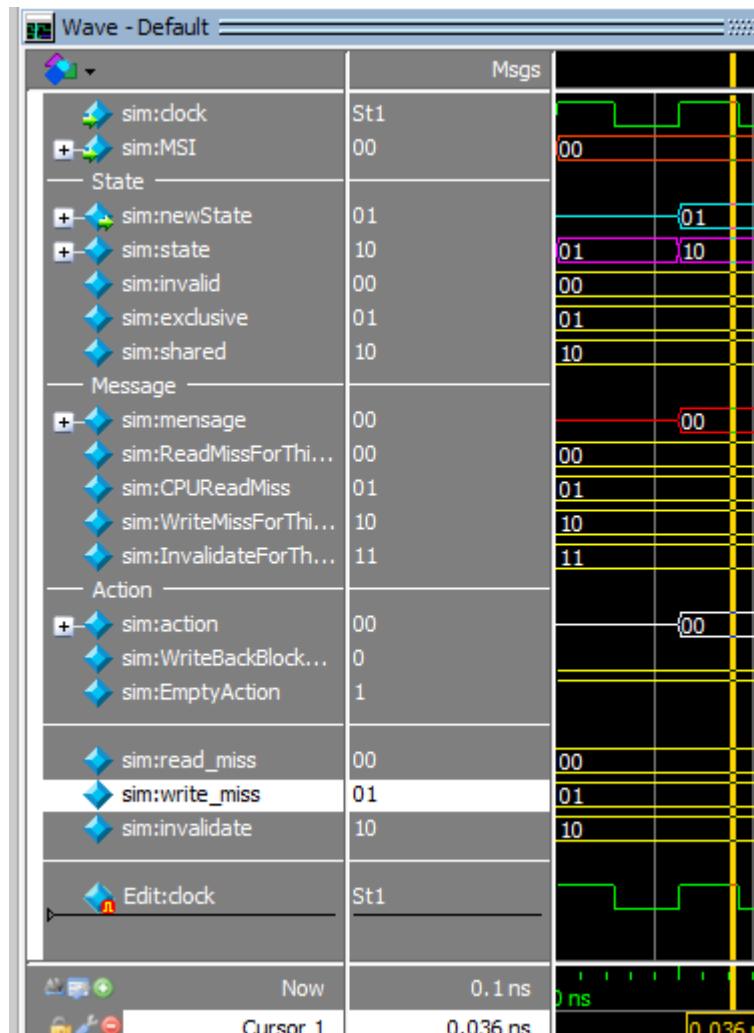
Executado com sucesso.

5. Exclusive -> read miss

Estado destino: Shared

Mensagem de controle: Read miss for this block

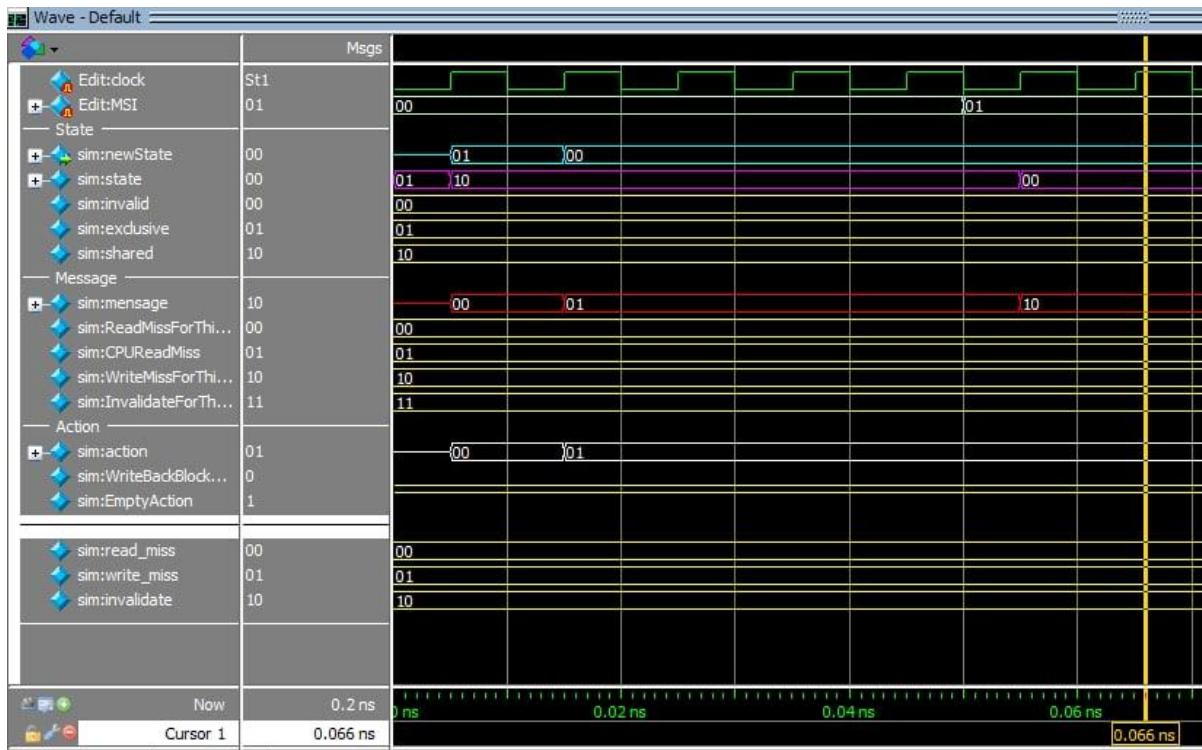
Ação Esperada: Write-back block; abort memory access



Executado com sucesso.

Lembrando que em as ondas contidas no bloco “Message” não são mensagens enviadas pela máquina, mas estão contidas nas ondas para fins de controle da simulação, onde pretendemos verificar se o caminho percorrido durante a simulação está correto.

Em uma simulação com muitos ciclos de clock, observamos em ‘state’:



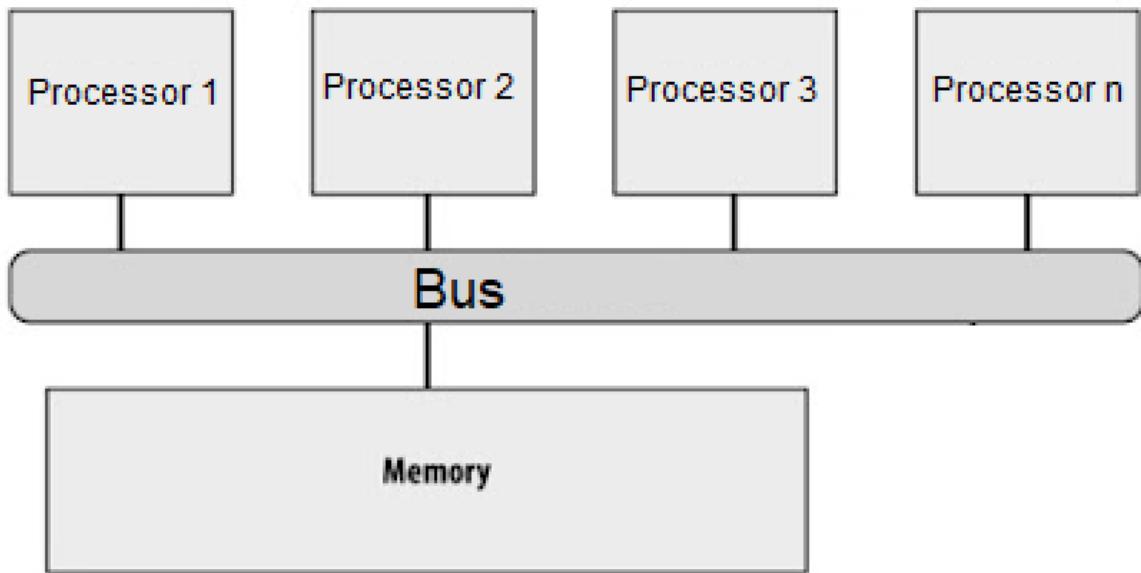
Começamos no estado Exclusivo (01) e fomos para Shared (10) com uma mensagem de Read Miss For This Block e com a ação de Write Back Block: Abort Memory Access. Continuamos em Shared, com a mensagem CPU Read Miss e sem nenhuma ação, até atingir MSI igual a 1, onde ocorre Write Miss e migração para o estado Invalid, com a mensagem Write Miss For This Block e sem ação.

Analisando a imagem do protocolo, podemos perceber que a máquina, quando atinge o estado Invalid, para. O que corresponde ao correto funcionamento de nossa implementação, visto que tudo ocorreu conforme o planejado.

Parte II

Desenvolvimento do projeto

Para a implementação do algoritmo foi seguido o conceito teórico passado nas aulas teóricas da disciplina, seguindo o seguinte modelo:



Módulos

1. MaquinaEmissora

Máquina implementada na parte 1, reconhecida como MaquinaEstadoCPU, responsável por emitir a mensagem ao sistema, que é frequentemente modificada de acordo com sua instrução atual.

2. MaquinaReceptora

Máquina implementada na parte 1, reconhecida como MaquinaEstadoBUS, responsável por receber a mensagem do sistema, que é frequentemente modificada de acordo com sua instrução enviada pela máquina emissora.

3. bus

Representa o barramento do protocolo Snooping.

4. processador

Representa a execução da instrução recebida.

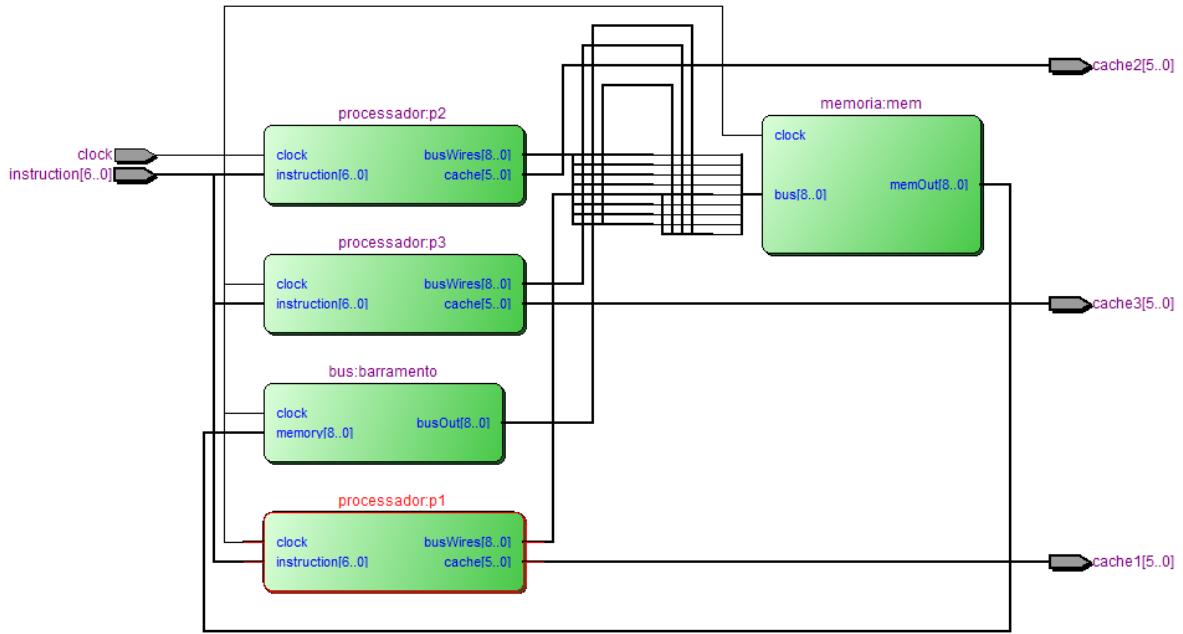
5. memória

Inicialização da memória.

6. SnoopingMSI

Instancia todos os processadores, memória, barramento e produz a ligação entre eles.

Logo, os modulos serão conectados da seguinte forma:



Simulação

- Caso Teste

Para a simulação do protocolo MSI, consideramos o seguinte caso teste:

Instrução	Resultado	Instrução em binário
P0: read 120	Po.Bo (S, 120, 00 20)	0001000
P0: write 120 <-- 80	Po.Bo (M, 120, 00 80) P3.Bo (I, 120 00 20)	0011011
P3: write 120 <--80	P3.Bo(M 120 0080) Po.Bo(I 120 0080) M (120 00 80)	1011011
P1: read 110	P1.B2 (S , 110 00 30) Po.B2 (S, 110 00 30) M (110 00 30)	0100100
P3: write 110 <-- 30	P1.B2 (I, 110 00 30) Po.B2 (I, 110 00 30) P3.B2(M, 110 00 30)	1010100
P3: read 110	Retorna 00 30 para o P3	1000100
P0: write 108 <-- 48	Po.B1 (M, 108 0048) P3.B1(I, 108 00 08)	0010001
P0: write 130 <--78	Po.B2 (M, 130 00 78)	0011110
P3: write 130 <--78	M(130 00 78) Po.B2 (I 130 00 78) P3.B2(M 130 0078) M (110, 00 10)	1011110

Onde as instruções correspondem:

Instrução:	7'b0000000				
	7'b	00	0	00	00
	Representa os 7 bits da instrução	Escolhe o processador	Determina a operação	Tag	Dado

E temos:

Processadores	Representação
P0	00
P1	01
P3	10

Operação	Representação
read	0
write	1

Address Tag	Representação
108	00
110	01
120	10
130	11

Data	Representação
30	00
48	01
78	10
80	11

- Implementação

Infelizmente, não conseguimos implementar por completo a parte 2 deste projeto a tempo de apresentar as simulações. Sendo assim, os módulos estão incompletos, e devido a isso, não conseguimos executar as simulações dos códigos testes.

Dificuldades

A própria linguagem Verilog se mostrou uma dificuldade na implementação. Tivemos dificuldades em aplicar o conhecimento teórico na prática utilizando a linguagem. Além disso, tendo em vista que estamos em final de semestre, o conteúdo cobrado por esta disciplina e outras presentes na grade, sobrecregou exaustivamente a carga horária dos integrantes deste relatório.

Sugestões

O protocolo MSI Snooping não é trivial, pois qualquer detalhe que foi mal implementado, ou um simples descuido pode ocasionar em diversos erros em formato exponencial durante a simulação do protocolo. Sendo assim, acreditamos que ao disponibilizar um pseudocódigo, ou até mesmo um esqueleto inicial de implementação, pode garantir um melhor entendimento do algoritmo, guiando-se para uma implementação segura e correta.

Comentários

Não conseguimos entregar a parte 2 totalmente implementada devido aos fatos já apresentados acima.