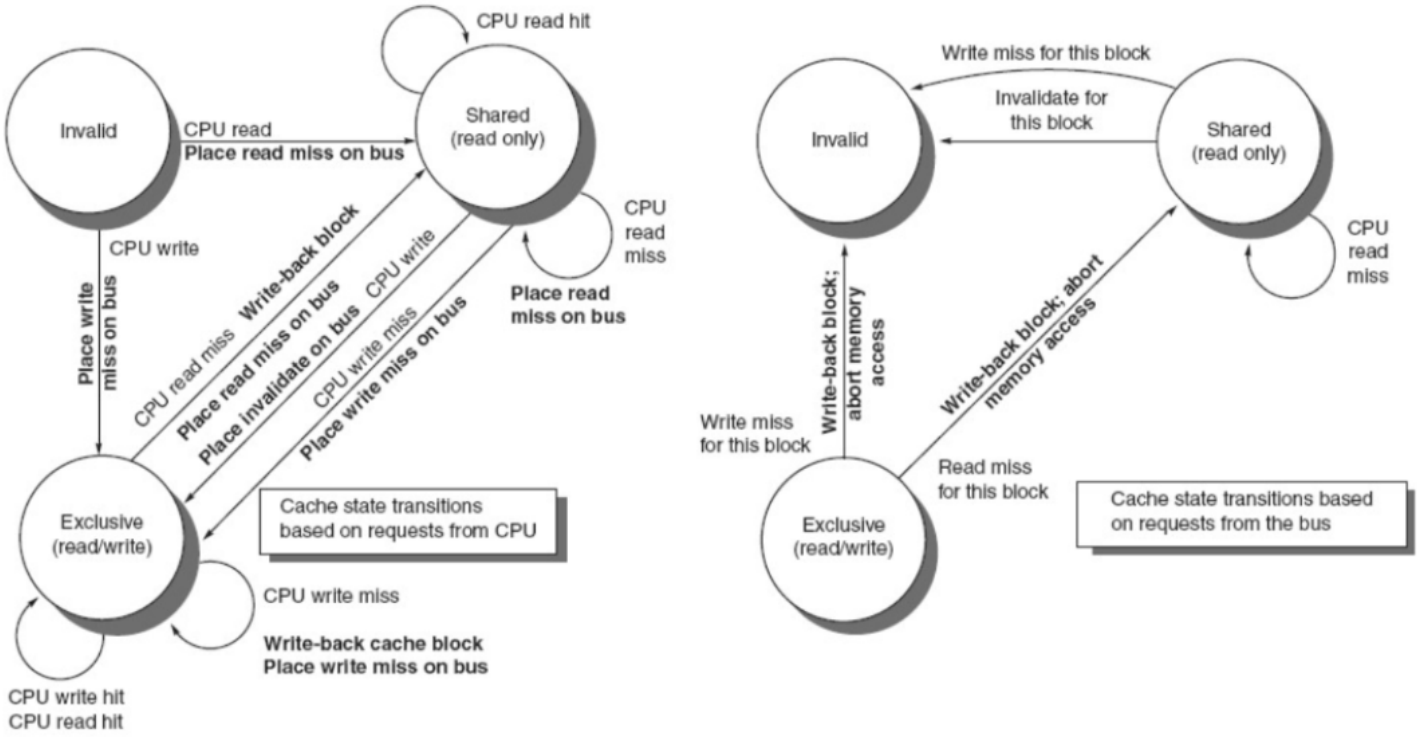
CEFET-MG

LABORATÓRIO DE ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES II

FELIPE MARTINS LEMOS DE MORAIS
MARIA IZAMARA CLARA COUTINHO

Prática IV — Protocolo Snooping (MSI)

Máquinas de Estados



Nos sistemas de memória compartilhada nos quais os processadores fazem uso de caches, é necessário manter consistência dos dados armazenados. Apesar de normalmente ser muito difícil, foi definido um protocolo para facilitar este processo, denominado *Snooping*, — ou bisbilhotar — que consiste na utilização de um barramento para o processo de comunicação.

PARTE 1

“

> Implementação das máquinas de estados do protocolo MSI Snooping. A simulação deve apresentar TODAS as transições das máquinas de estado, juntamente com as mensagens (read miss, write miss, invalidate) e ações (aborta o acesso à memória e write back).

“

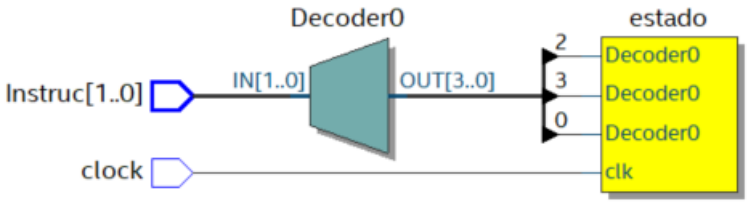
Para facilitar na implementação do protocolo e atendendo ao pedido da professora, elaboramos dois projetos, um referente ao envio de mensagens e outro para escuta das alterações no barramento.

A partir daí, seguindo a máquina de estados, temos diferentes ações e mensagens para cada uma das situações indicadas como de Read Miss, Read Hit, Write Miss e Write Hit.

Observando o esquema gráfico da máquina de estados, nos guiamos por meio dele para a atribuição correta de cada uma das consequências da

SIMULAÇÃO

Máquina 1



Legenda:

Operações
00 - Read miss
01 - Read hit
10 - Write miss
11 - Write hit

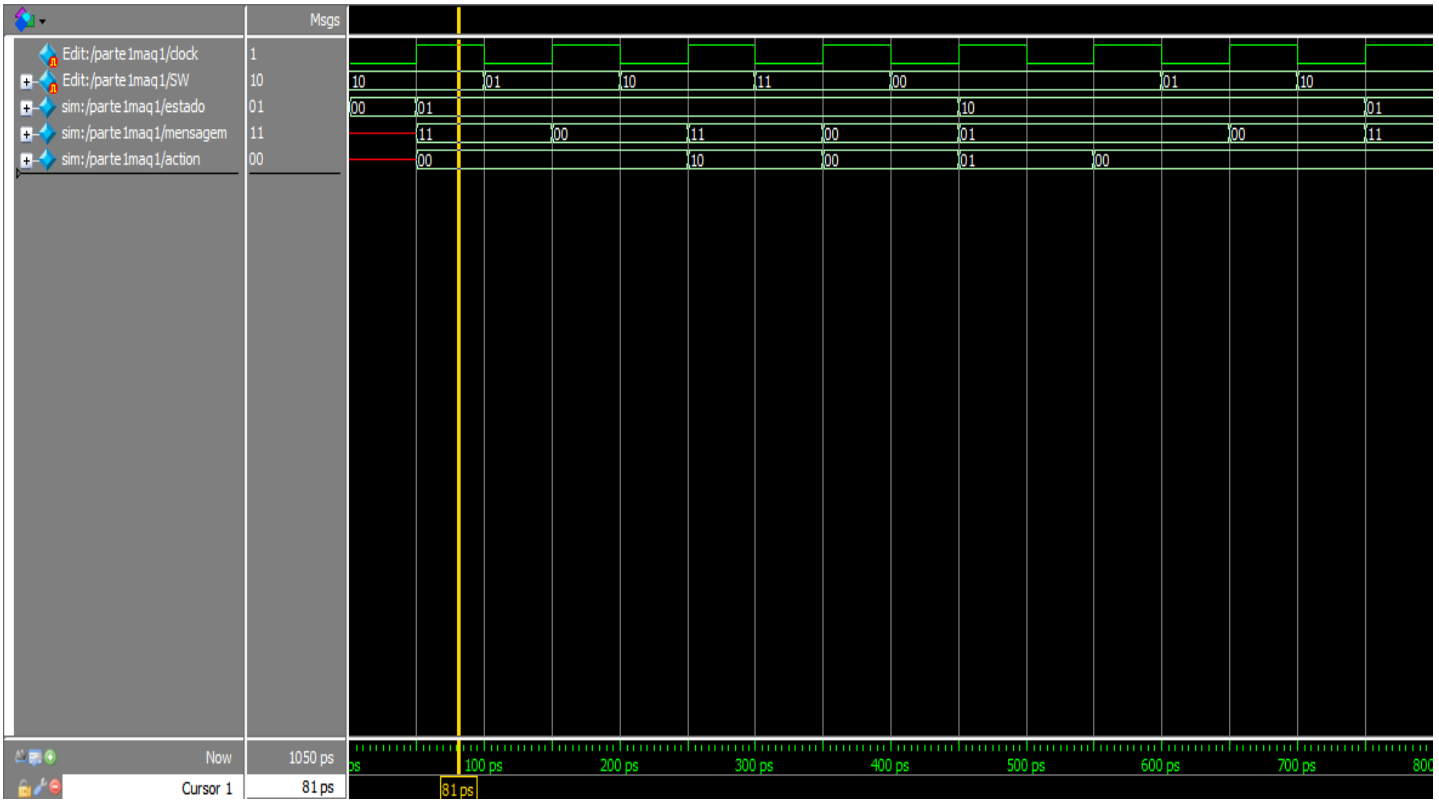
Estados
00 - Invalid
01 - Exclusive
10 - Shared

Mensagens
00 - Nenhuma
01 - Place read miss on bus
10 - Place Invalidate on bus
11 - Place write miss on bus

Ações
00 - Nenhuma
01 - Write back block
10 - Write back cache block

a)

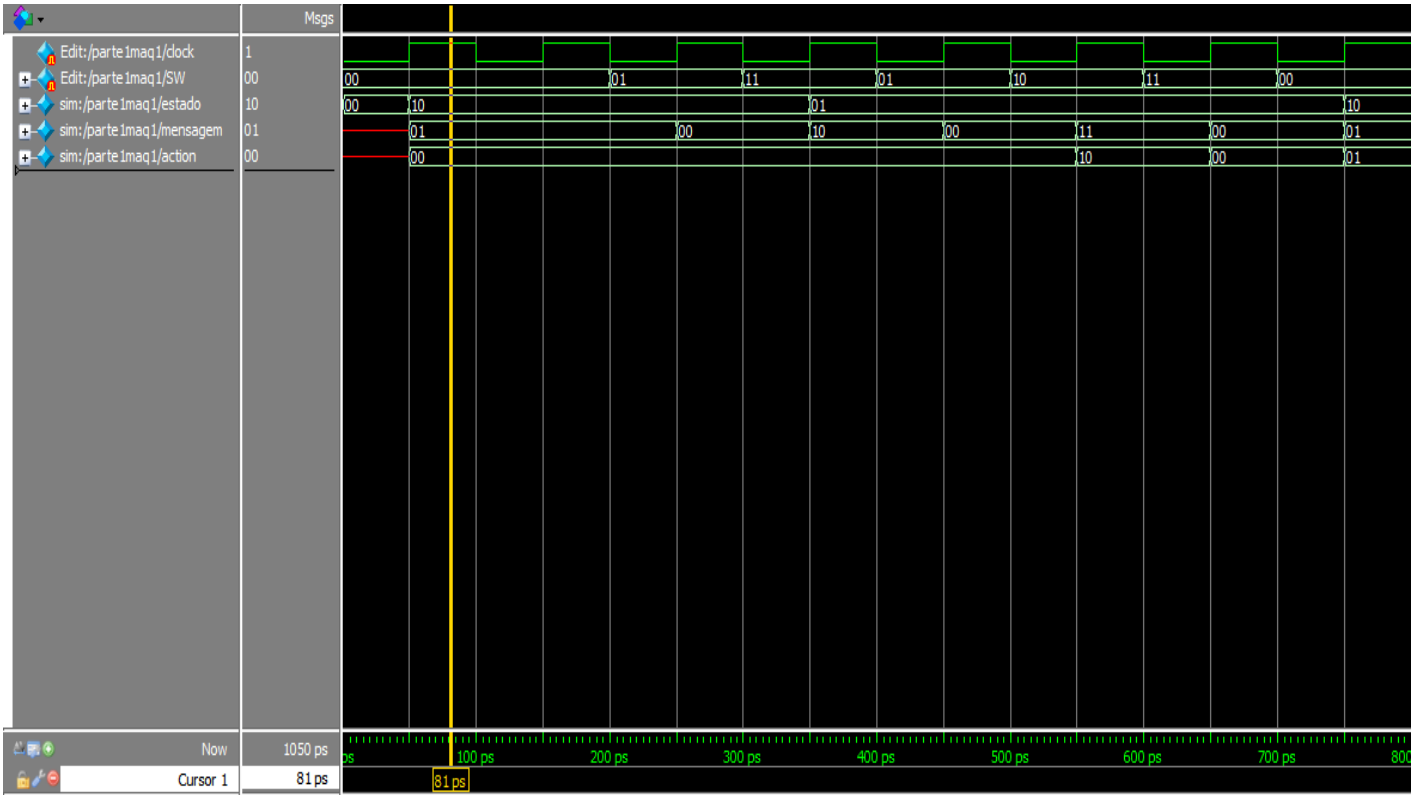
Ciclo	Estados	Instrução	Mensagem	Ação
1	I > E	Write miss	Place write miss on bus	
2	E > E	Read hit		
3	E > E	Write miss	Place write miss on bus	Write back cache block
4	E > E	Write hit		
5	E > S	Read miss	Place read miss on bus	Write back block
6	S > S	Read miss	Place read miss on bus	
7	S > S	Read hit		
8	S > E	Write miss	Place write miss on bus	



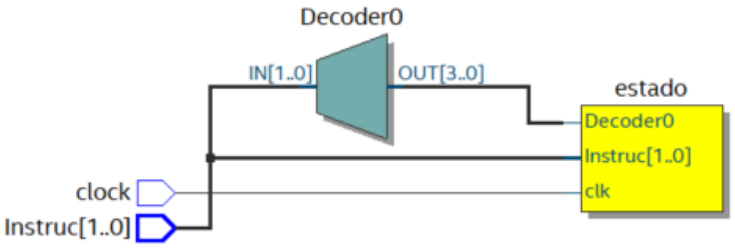
b)

Ciclo	Estados	Instrução	Mensagem	Ação
1	0 > 2	Read miss	Place read miss on bus	
2	2 > 2	Read miss	Place read miss on bus	

3	2 > 2	Read hit		
4	2 > 1	Write hit	Place Invalidate on bus	
5	1 > 1	Read hit		
6	1 > 1	Write miss	Place write miss on bus	Write back cache block
7	1 > 1	Write hit		
8	1 > 2	Read miss	Place read miss on bus	Write back block



Máquina 2



Legenda:

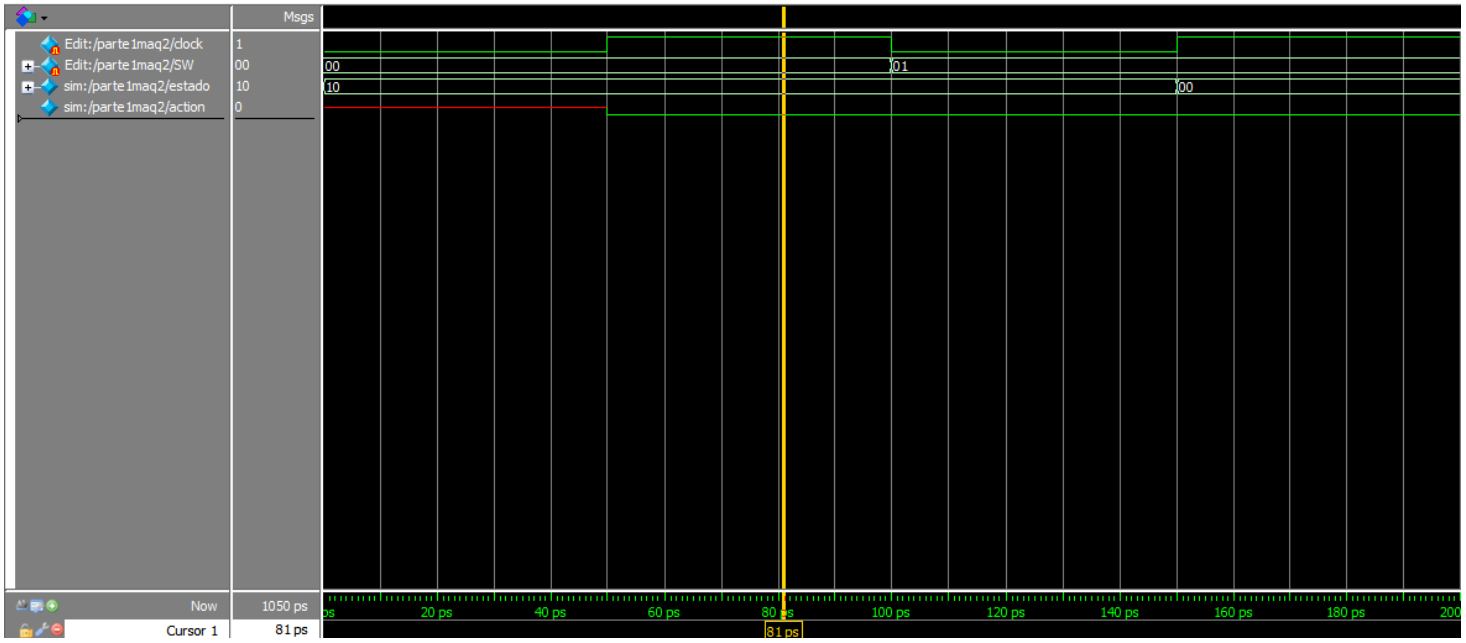
Operações
00 - Read miss
01 - Invalidate
10 - Write miss

Estados
00 - Invalid
01 - Exclusive
10 - Shared

Ações
0 - Nenhuma
1 - Write back block

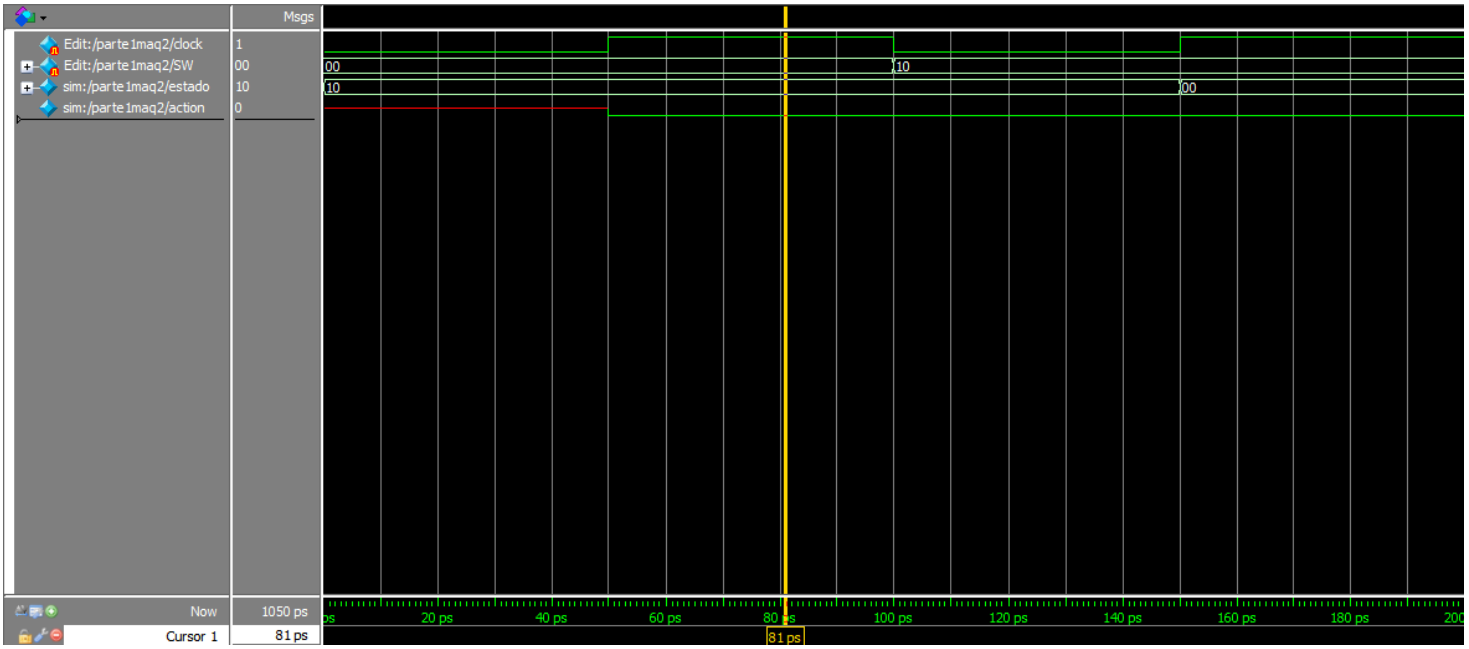
a)

Ciclo	Estados	Instrução	Ação
1	2 > 2	Read miss	
2	2 > 0	Invalidate	



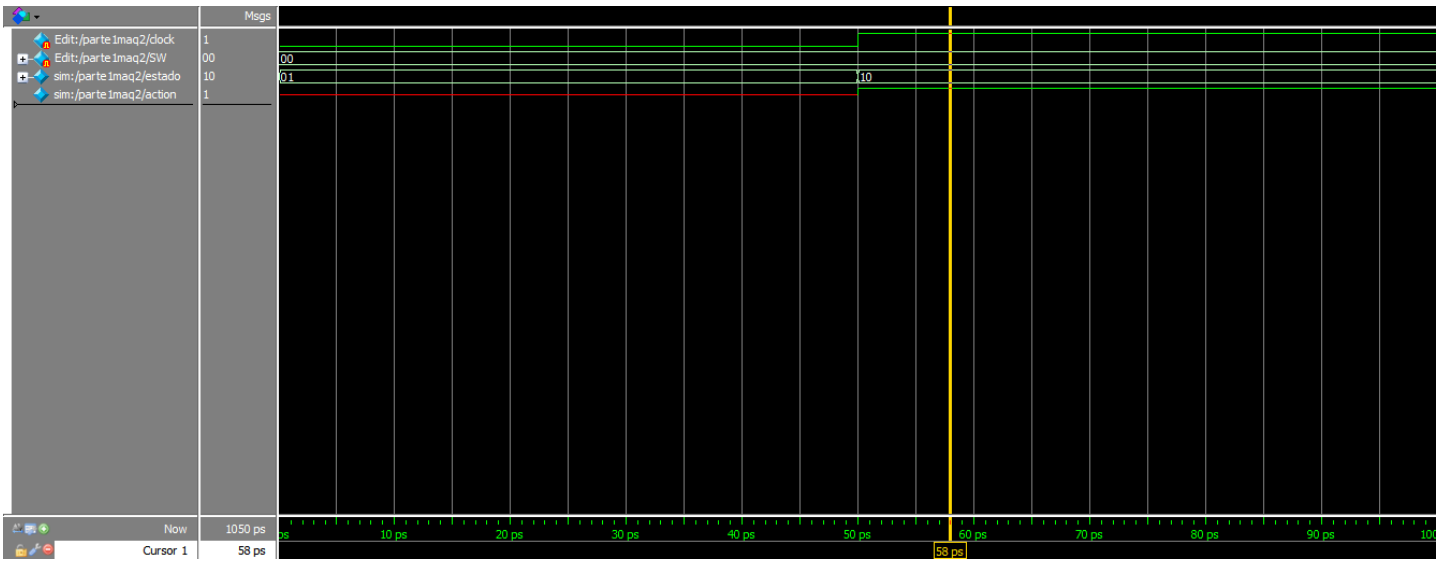
b)

Ciclo	Estados	Instrução	Ação
1	2 > 2	Read miss	
2	2 > 0	Write miss	



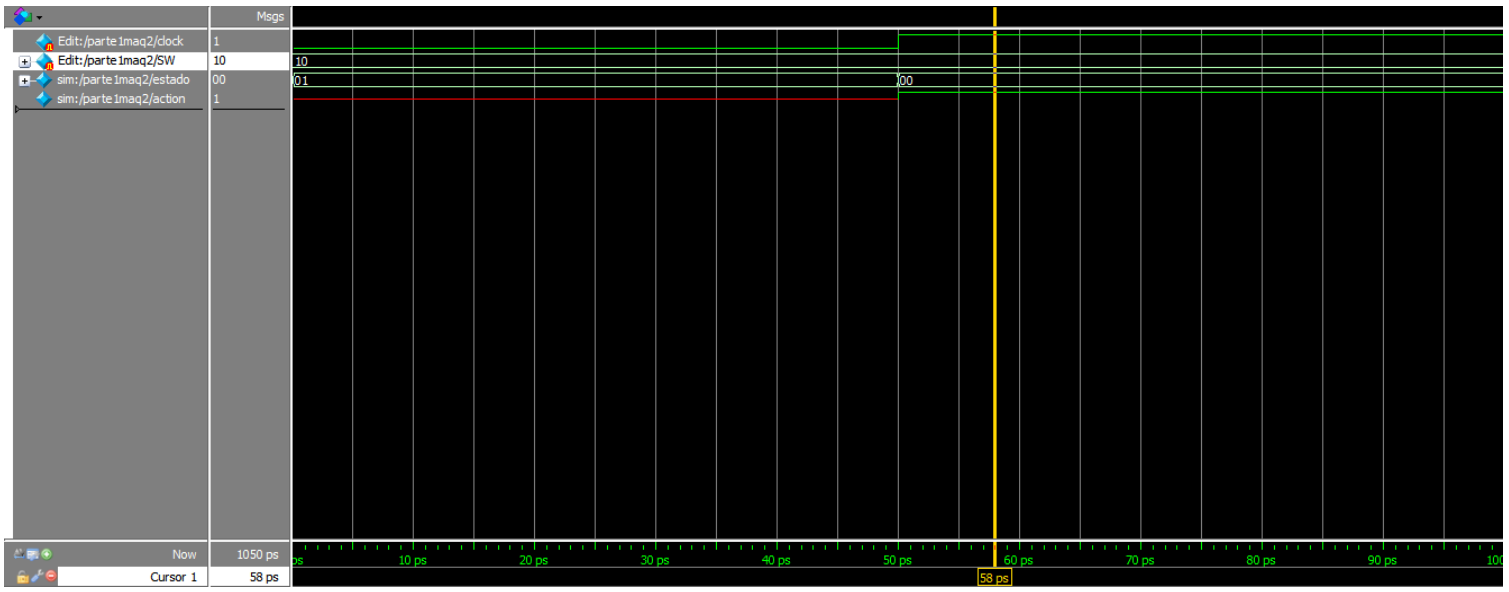
c)

Ciclo	Estados	Instrução	Ação
1	1 > 2	Read miss	Write back block



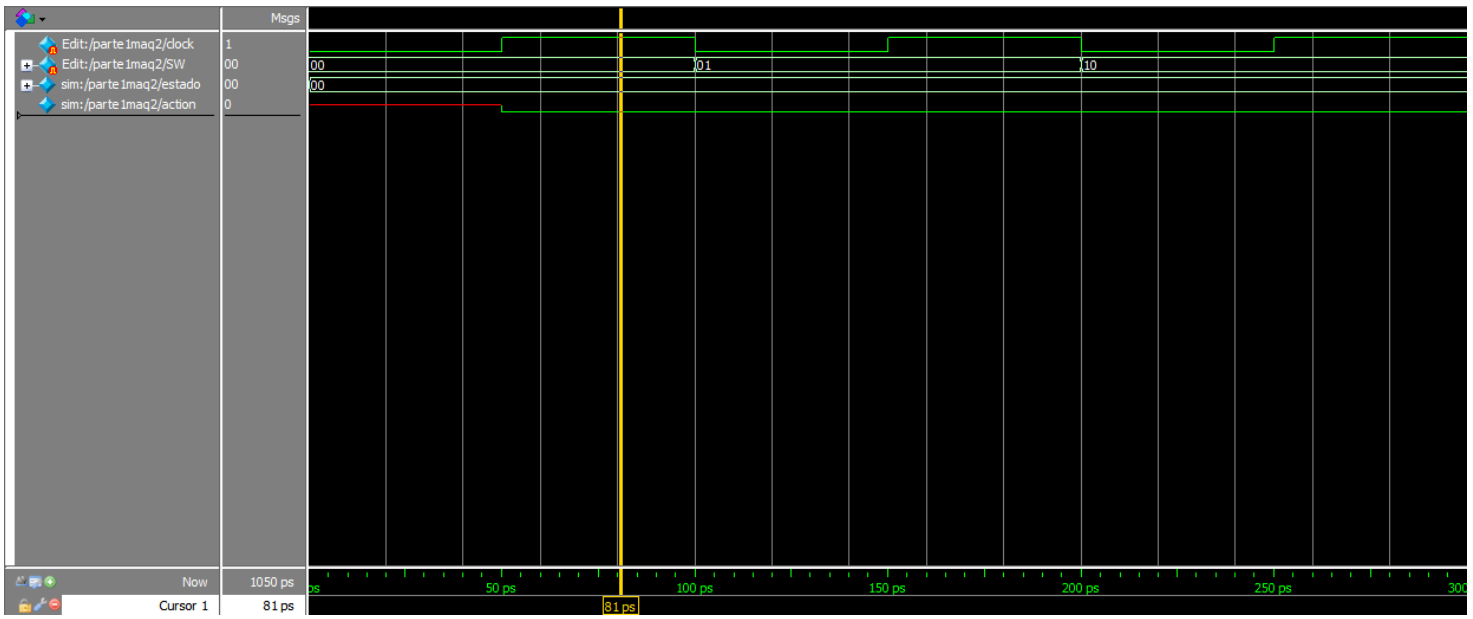
d)

Ciclo	Estados	Instrução	Ação
1	1 > 0	Write miss	Write back block

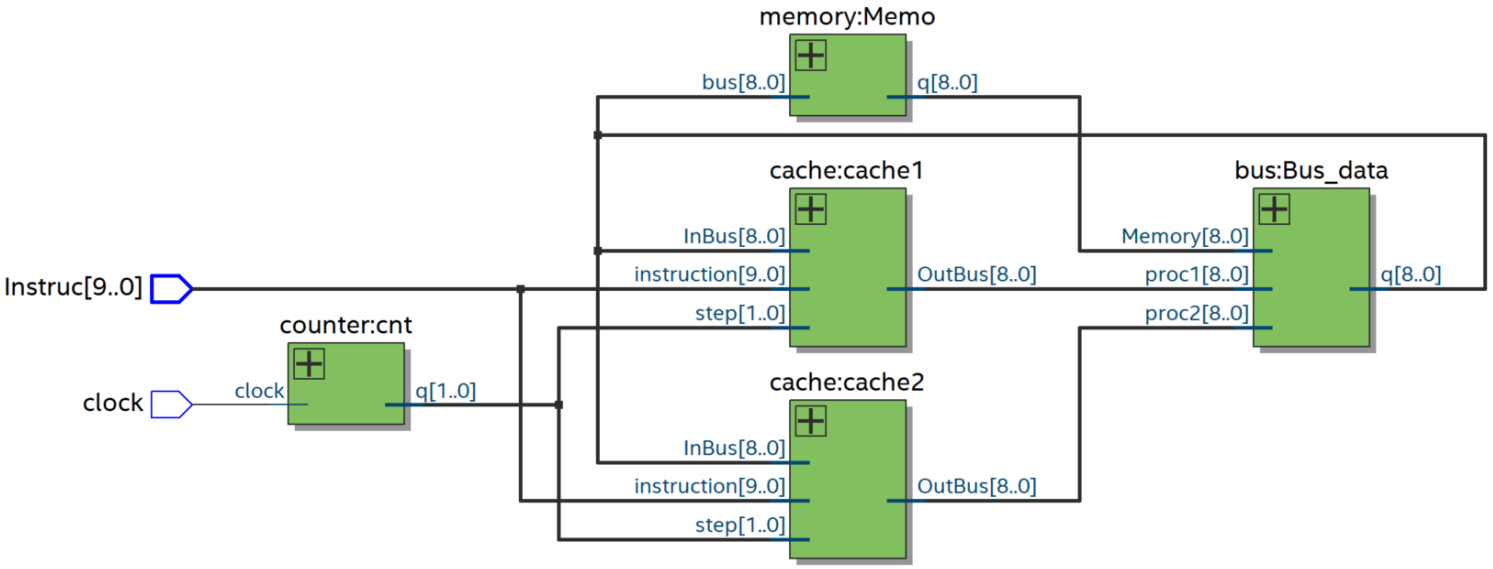


e)

Ciclo	Estados	Instrução	Ação
1	0 > 0	Read miss	
2	0 > 1	Invalidate	
3	0 > 2	Write miss	



PARTE 2



> Considerando o protocolo MSI de coerência de cache, implemente um projeto com os seguintes módulos:
>
> - Duas CPU, que basicamente realizam operações de leitura e escrita nas caches.
> - Duas Caches: Uma cache para cada CPU. Uma memória compartilhada pelas CPUs.

Para a elaboração do proposto, desenvolvemos 5 módulos diferentes para cada módulo presente:

- ➔ **Barramento Bus (‘bus.v’) :**
- ➔ **Memórias Cache dos Processadores (‘cache.v’) :**
- ➔ **Contador de etapas (‘counter.v’) :**
- ➔ **Memória Compartilhada (‘memory.v’) :**
- ➔ **Parte 2 (‘parte2.v’) é onde conectamos tudo.**

Estabelecemos então uma padronização de como funcionaria o sistema para que o processo pudesse ocorrer de maneira harmônica, dado uma instrução fornecida do formato:

Instrução:

O PP TTT VVVV — 10 bits
O- Operação **P-** Processador **T-** Tag **V-** Valor

De forma similar, as caches foram organizadas de maneira que cada endereço fosse assim retratado:

EE TTT DDDD — 9 bits
E- Estado **T-** Tag **D-** Dado a ser escrito

Similarmente, os dados no barramento Bus ficaram assim distribuídos:

RR TTT DDDD — 9 bits
R- Tipo de operação **T-** Tag **D-** Dados

Seguindo o protocolo, cada processador se comunica conjuntamente com a memória por intermédio do único barramento de dados Bus, o que exigiu uma coordenação de 4 ciclos de clock para cada instrução fornecida.

SIMULAÇÃO

Não foi possível utilizar as numerações exatas dos valores no caso de teste, uma vez que necessitariam de um número excessivo de bits para a sua fiel reprodução. Assim, atribuímos a seguinte correspondência aos valores:

Dados

08 = 0000
10 = 0001
18 = 0010
20 = 0011
28 = 0100
30 = 0101
68 = 0110
40 = 0111
38 = 1000
98 = 1001

Tag
100 = 000
108 = 001
110 = 010
118 = 011
120 = 100
128 = 101
130 = 110

E as respectivas memórias, inicializadas do seguinte modo:

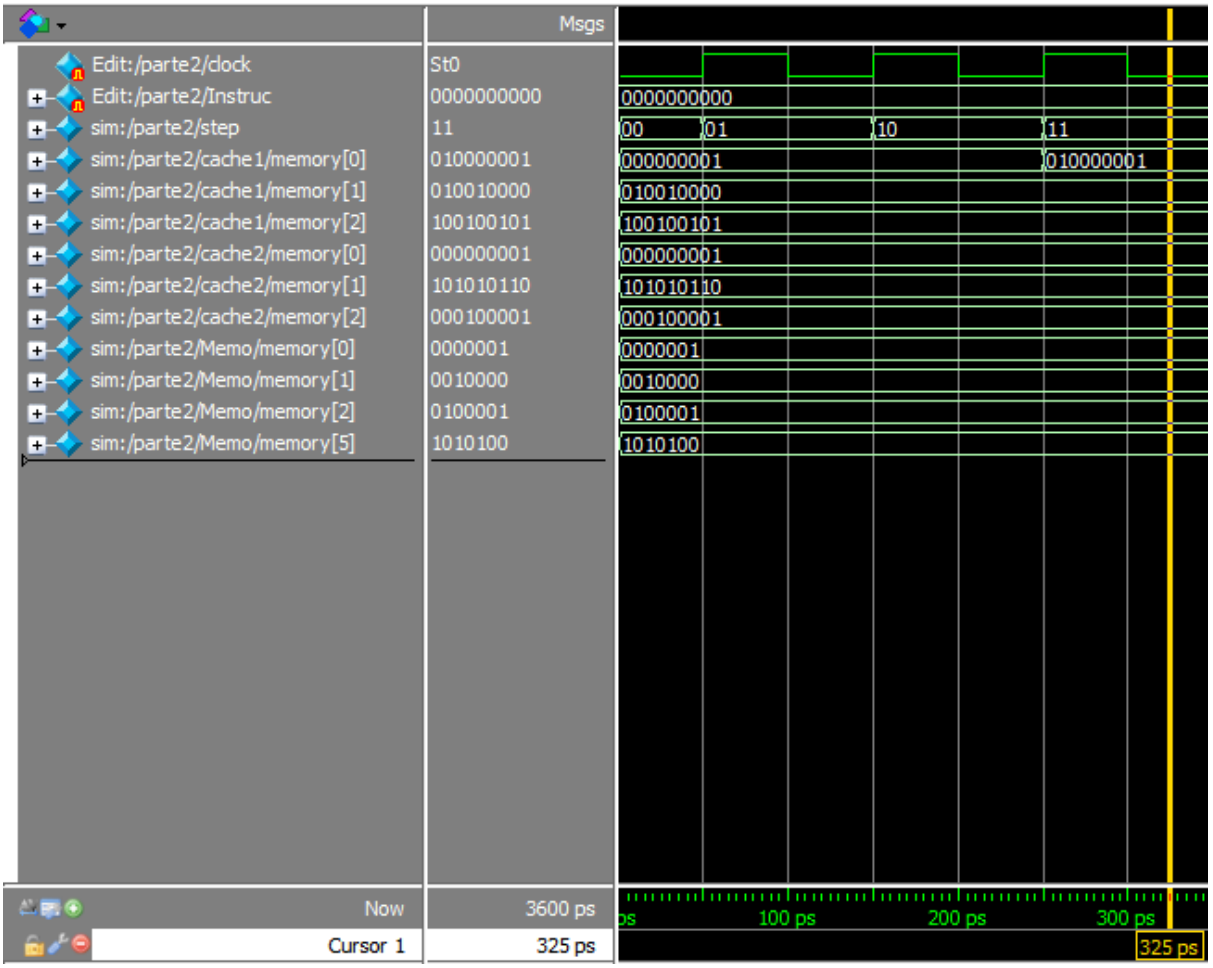
Memória
000 0001
001 0000
010 0001
011 0010
100 0011
101 0100
110 0101

Cache 1
00 000 0001
01 001 0000
10 010 0101
00 011 0001

a) P0: read 100

Read miss

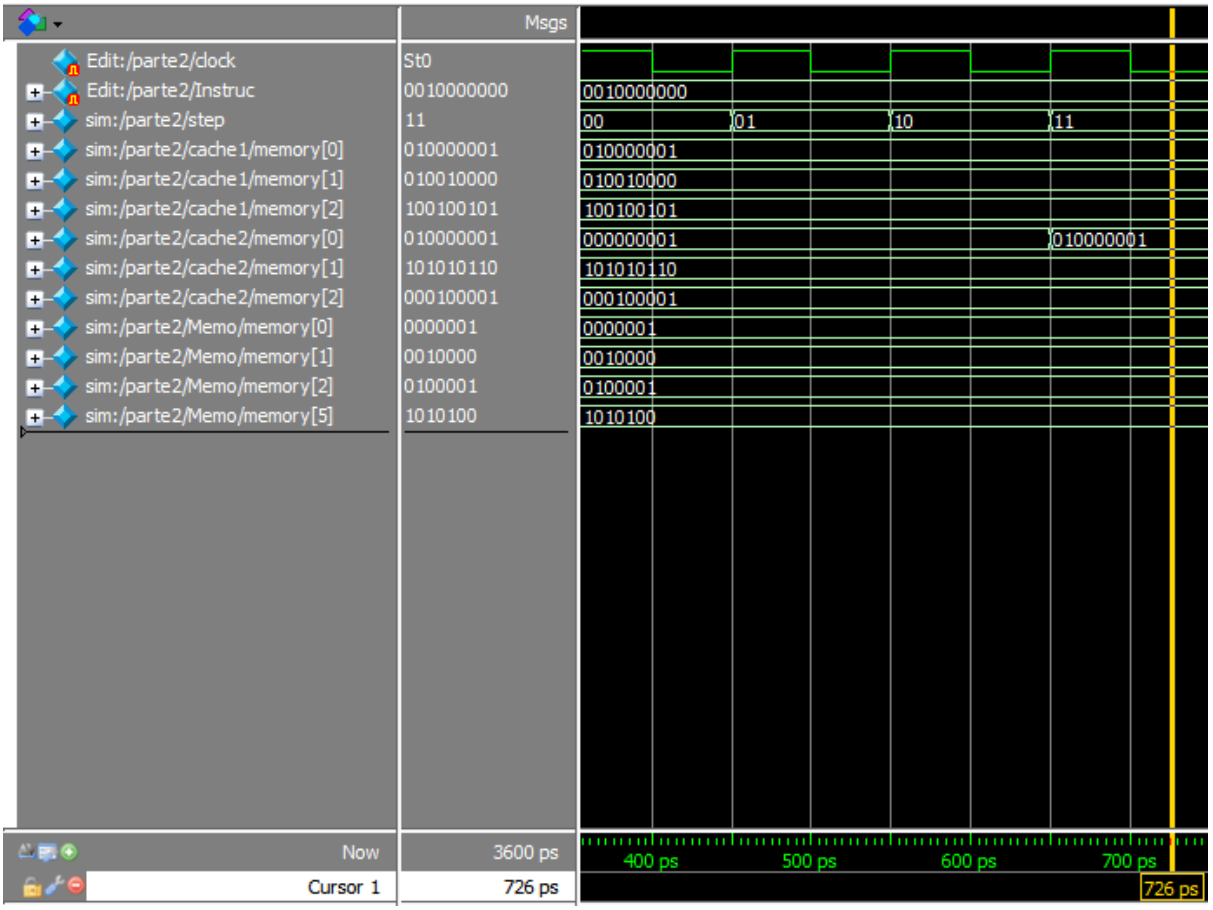
P0. B0 (S, 100 0010)



b) P1: read 100

Read miss (Busca na memória)

P1.B0 (S, 100, 0010)

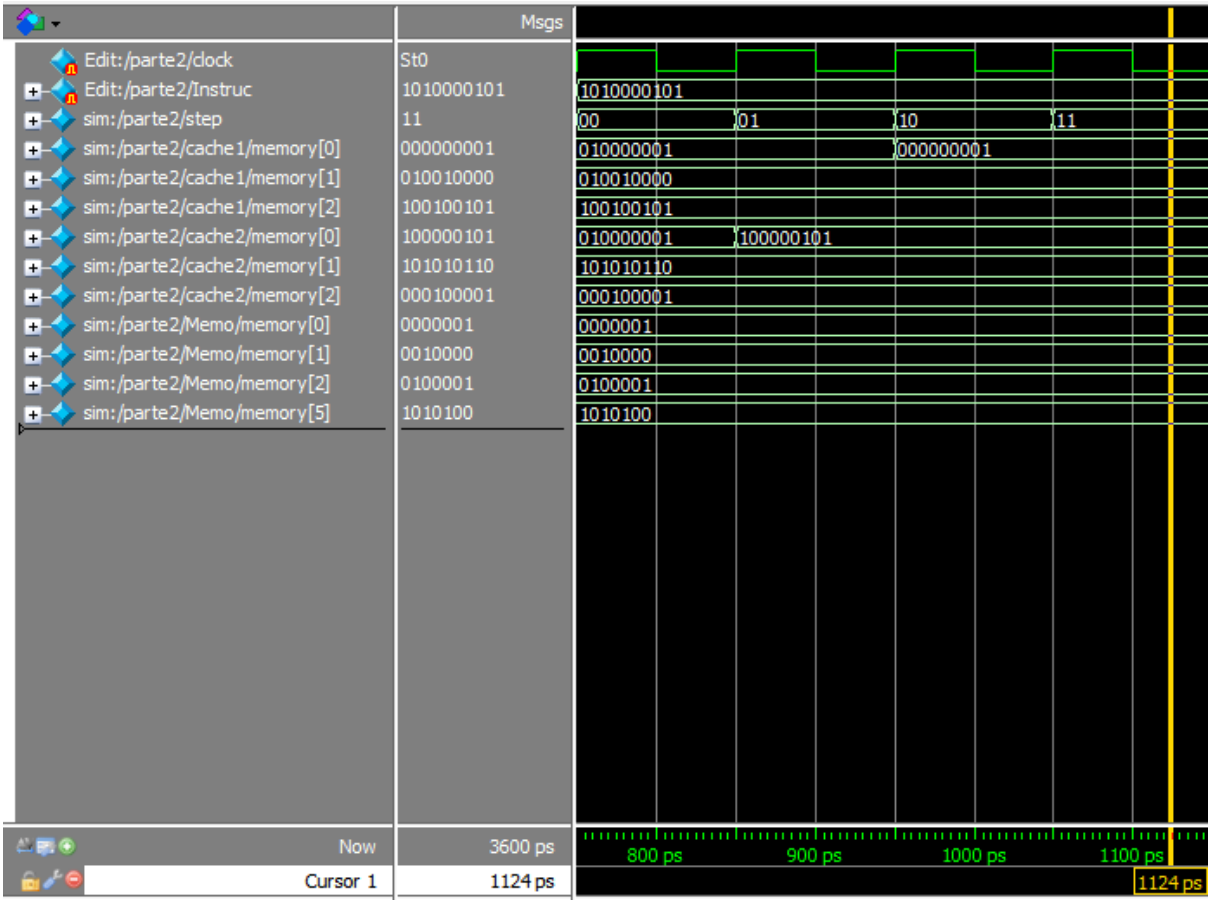


c) P1: write 100 <-30

Write hit, invalidate

P0.B0 (I ,100, 00 10)

P1.B0 (M, 100, 0030)



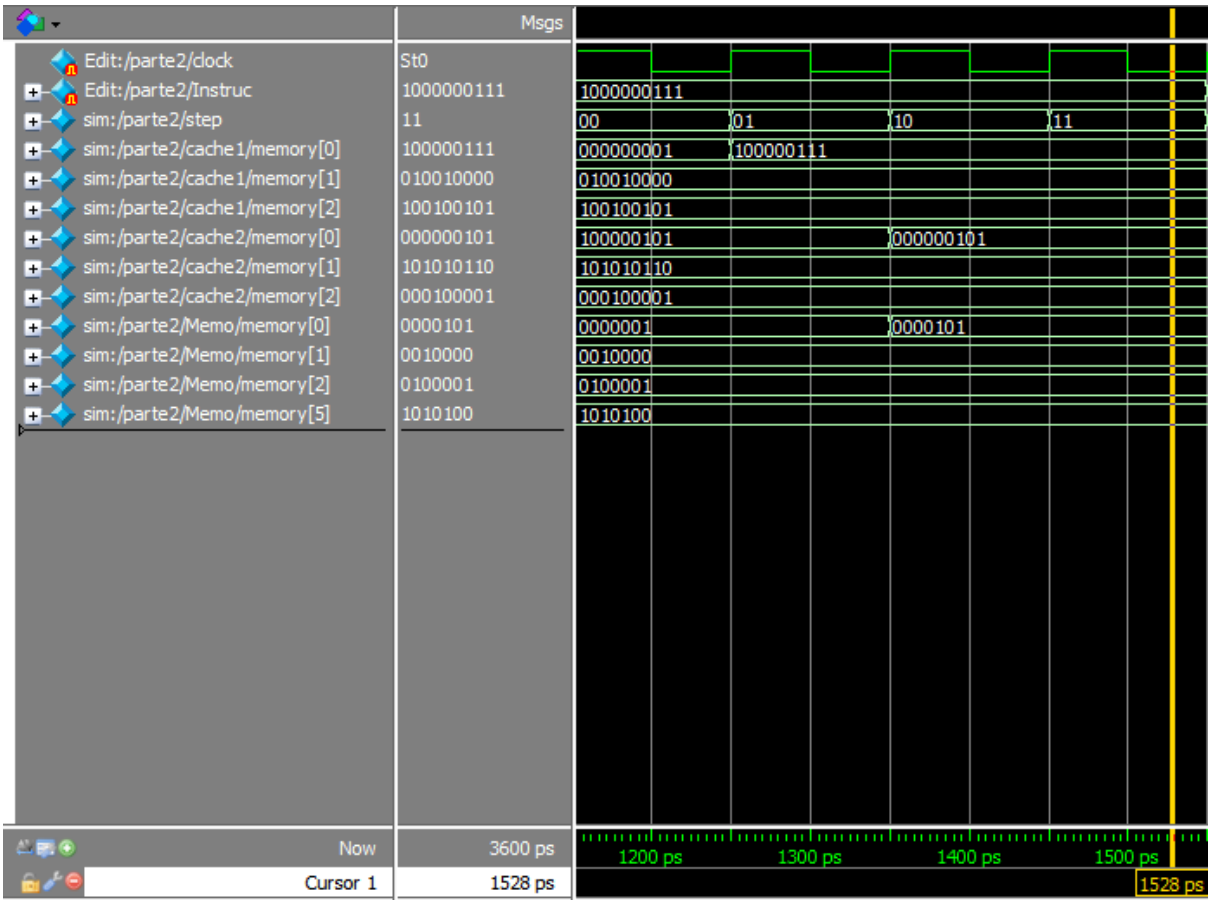
d) P0: write 100 <-40

Write miss, aborta o acesso à memória, Write Back

P0.B0 (M ,100, 00 40)

P1.B0 (I, 100, 0030)

M(100, 00 30)



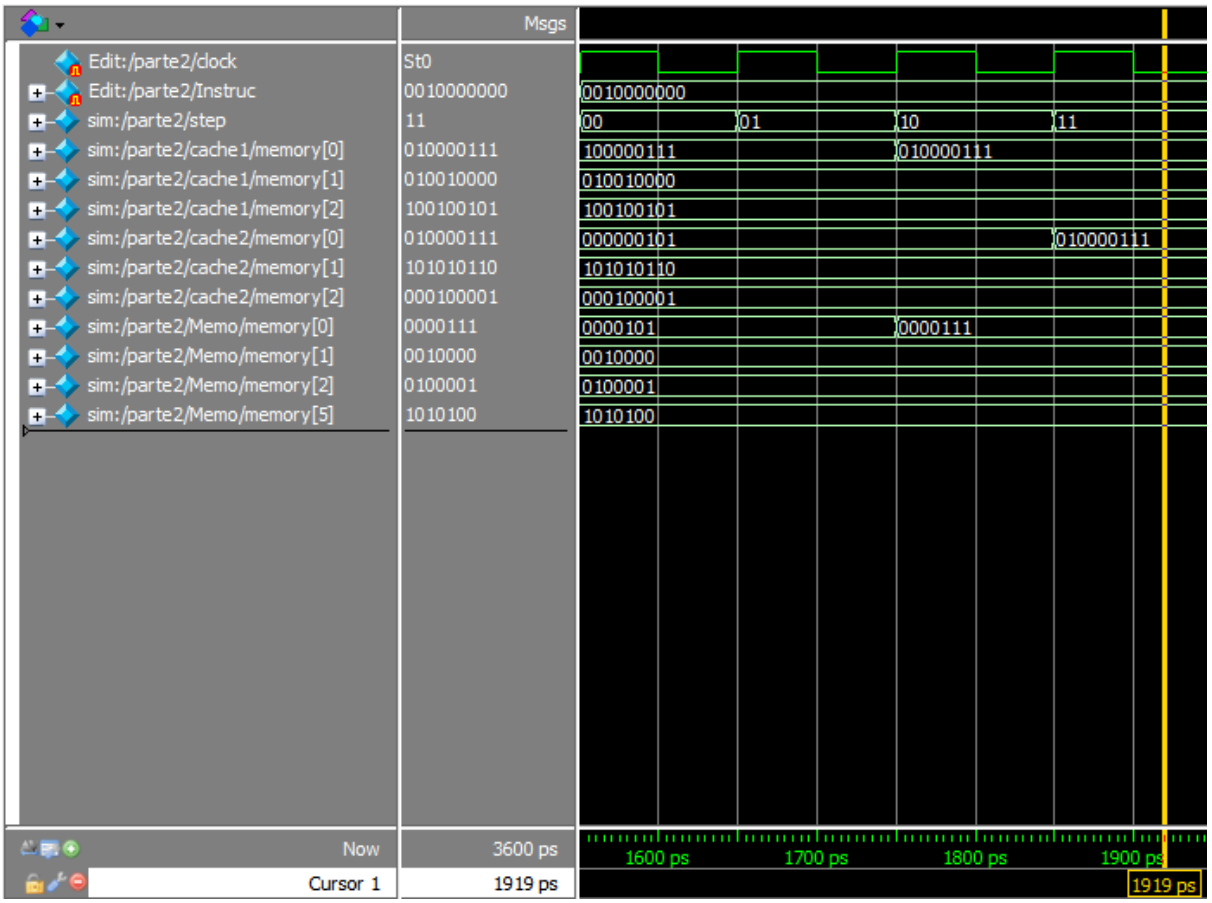
e) P1: read 100

Read miss, write back block

P0.B0 (S ,100, 00 40)

P1.B0 (S, 100, 0040)

M(100, 00 40)



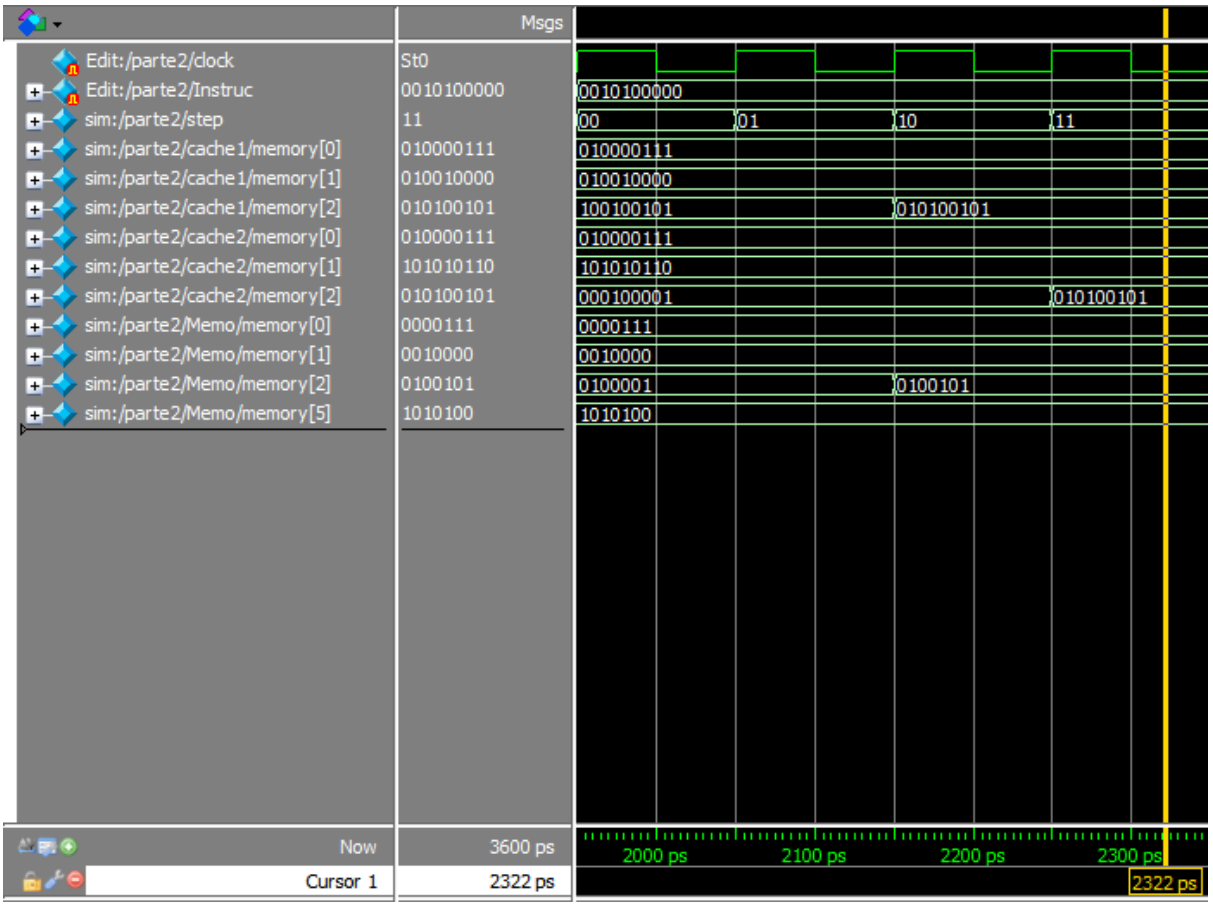
f) P1: read 110

Read miss, write back 110

M(110, 0030)

P0.B2 (S, 110, 0030)

P1.B2(S, 110,00 30)



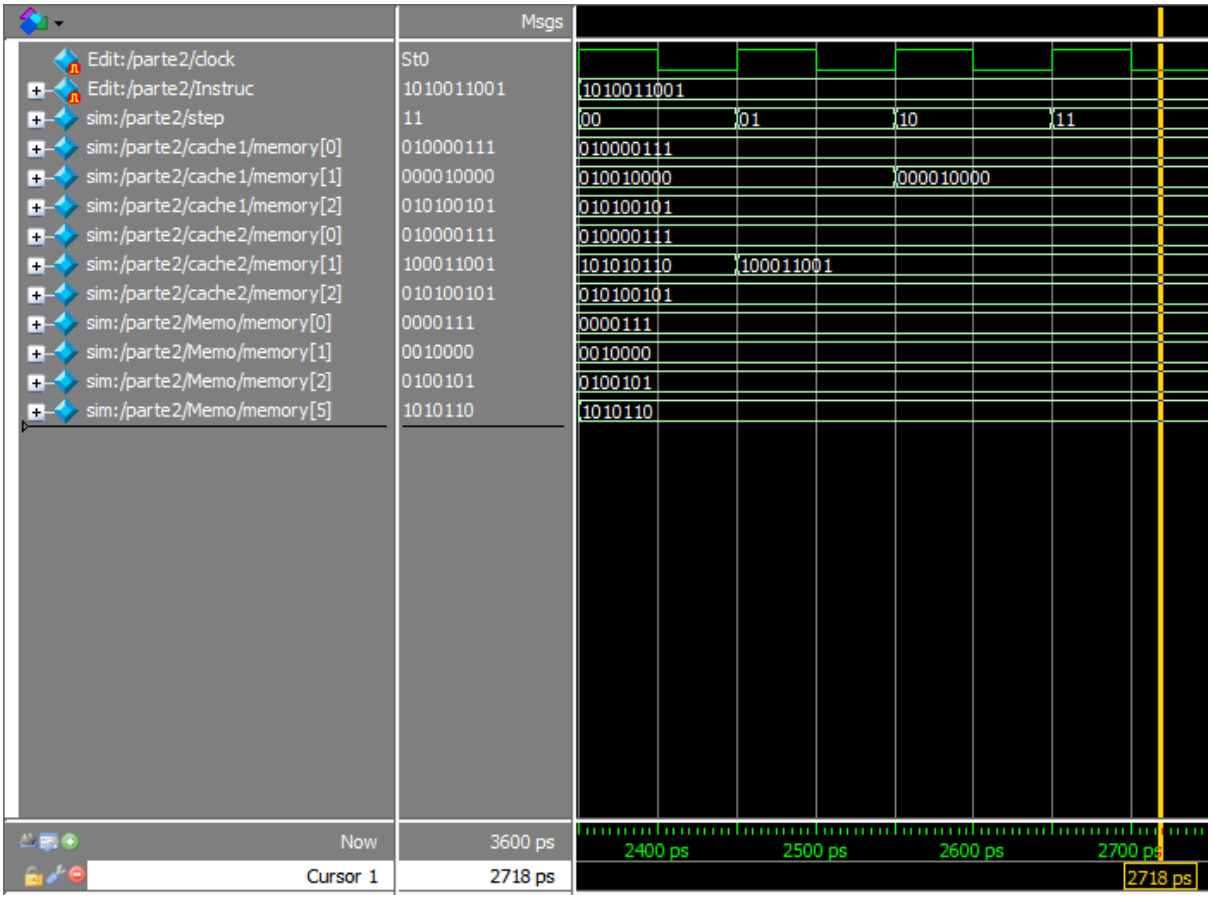
g) P1: write 108<-98

Write miss, write back 128

P1. B1(M, 108 0098)

M(128, 0068)

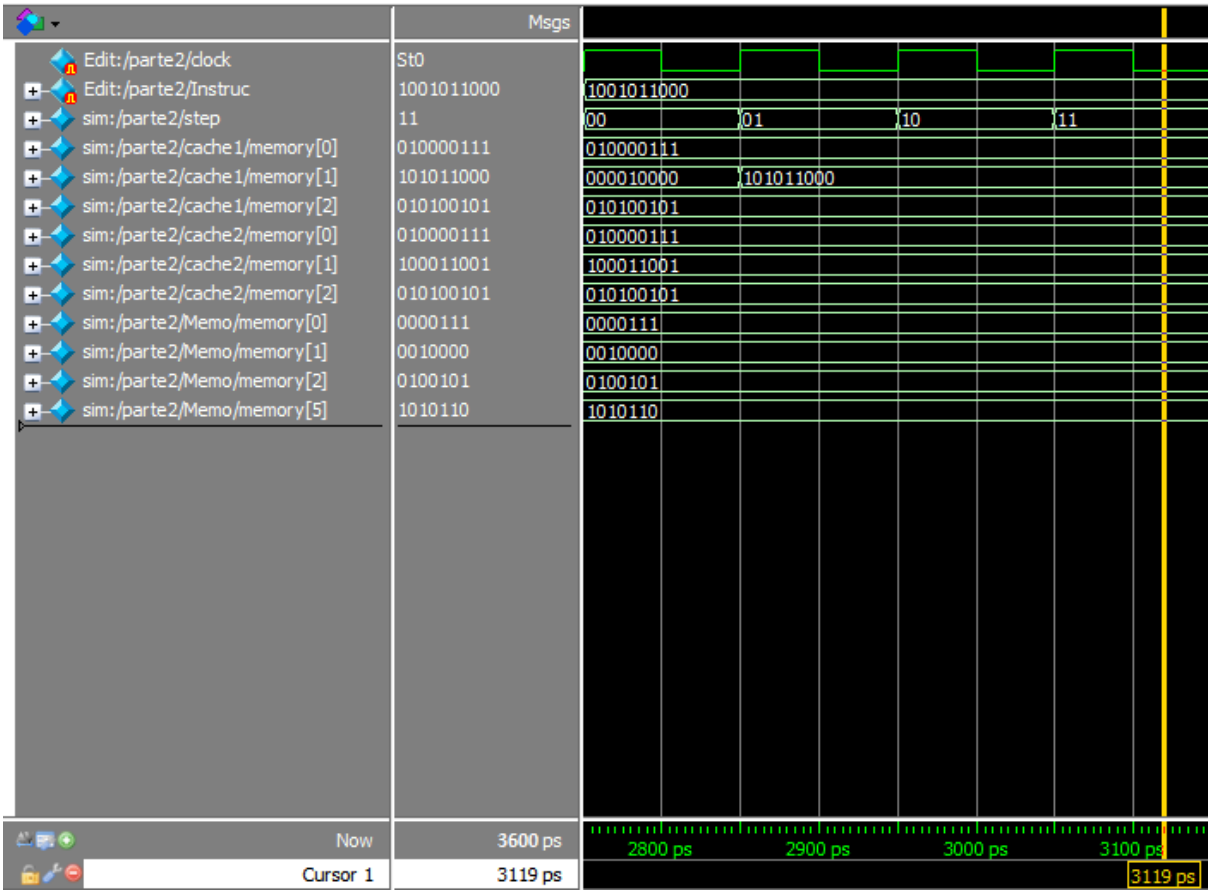
P0.B1(I, 108 00 08)



h) P0: write 128 <- 38

Write miss, busca na memória

P0.B1(M, 128 00 38)



i) **P1: write 128 <- 68**

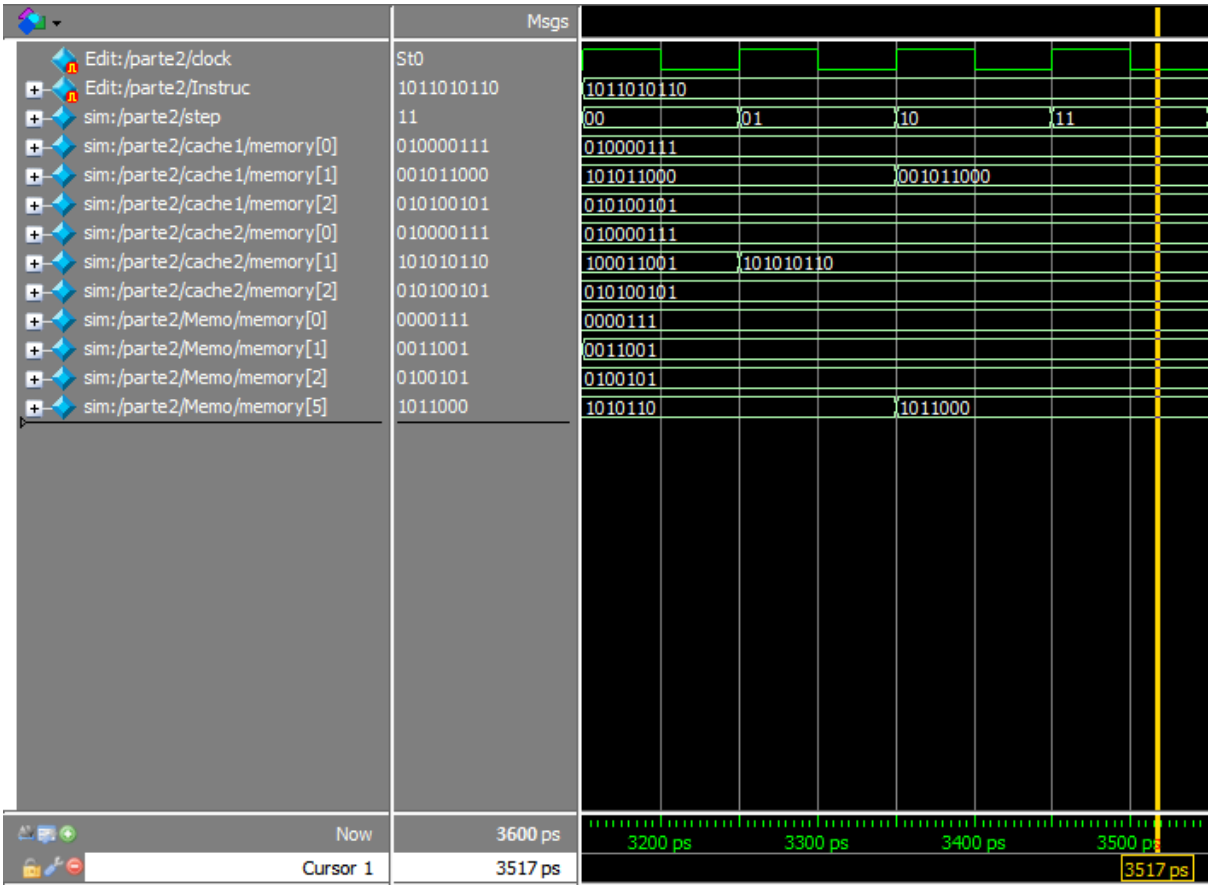
Write miss, write back 108, write back 128

P1.B1 (M, 128 0068)

Mem(108, 00 98)

P0.B1 (I, 128, 00 38)

Mem (128, 00 38)



Dificuldades Encontradas

Nossas principais dificuldades se concentraram em encontrar a melhor maneira de se implementar as memórias e como reproduzir um modelo de blocos que funcionasse corretamente, juntamente com a correspondência de bits para que tudo funcionasse; além de conseguir depois sincronizar todas as funcionalidades dos módulos de maneira a funcionarem em harmonia no tempo correto de execução em cada etapa, o que inicialmente tomou um grande tempo até encontrarmos a solução, resolvida com 3 caracteres a mais de código.

Sugestões de Melhoria da Prática

Uma ótima sugestão seria não deixar que os alunos muitas vezes “descubram por si próprios” alguns elementos nas práticas. Apesar de sempre disponível e aberta para ajudar, a maneira de explicar da professora não consegue ficar tão clara para quem tem mais dificuldade, provavelmente por já não ser tão complexo na visão de alguns, ao mesmo tempo que para outros com menor rapidez na compreensão torna a aprendizagem ainda mais penosa. Então deixar alguns aspectos-chaves das prática como “implícitos” não atinge um efeito de recompensar um aluno mais interessado com um “bônus” ou incentivo, mas sim dificulta para que os demais alunos com maior dificuldade consigam desenvolver a atividade de maneira mais lógica e completamente convicta do que estão fazendo.

Comentários adicionais

Como último trabalho da disciplina cumpriu um papel bom como maneira de resumir o requisitado durante a mesma. Agradecimentos especiais aos meus colegas do curso que também nos ajudaram durante todo o percurso da disciplina e as mais sinceras desculpas pelos eventuais deslizes cometidos no processo de aprendizagem e ressentimentos daí surgidos.