TEACHING THE CACHE MEMORY COHERENCE WITH THE MESI PROTOCOL SIMULATOR

F. J. JIMÉNEZ[1], J. GÓMEZ[1], A. MESONES[1], E. HERRUZO[1], J. I. BENAVIDES[1] Y F. J. SÁNCHEZ[2]
[1]*Dpto. Electrotecnia y Electrónica. Escuela Politécnica Superior. Universidad de Córdoba. Av. Menéndez Pidal s/n. 14081. Córdoba. Spain.*
[2]*I.E. S. Emilio Canalejo Olmeda. Av. Constitución, 18. 14550. Montilla. Córdoba. Spain.*

This document is a short review of the MESI protocol simulator. This simulator is used for teaching the cache memory coherence on the computer systems with hierarchical memory system and for explaining the process of the cache memory location in multilevel cache memory systems. This paper begins with a description of the course in which the simulator is used, an explanation about the MESI protocol and how the simulator works. Then the experimental results in a real teaching environment are described.

1. Introduction

The MESI cache coherence protocol simulator is presented in this paper [1]. The MESI protocol is a method to maintain the coherence of the cache memory content in hierarchical memory systems [2], [3]. It is based on four possible states of the cache blocks: Modified, Exclusive, Shared and Invalid. Each accessed block is in one of these stages and the transitions among them define the MESI protocol. Nowadays, most processors (Intel, AMD) use this protocol or its versions. Knowing how these processors maintain the cache coherence is very important for the students. The MESI simulator is a software tool which has been implemented in the JAVA language. It has been developed specifically for teaching purposes. It has been realized to show how the MESI protocol works to maintain the cache memory coherence in a multi-user system for a single processor. The simulator permits to configure the number of cache levels, and for each cache level the user can indicate the whole cache memory parameters as the cache memory capacity, the cache line size, the associativity, the replacement policy, the number of words per memory access, the writing policy, etc. Moreover, the user can configure the statistics of the memory access to study and how they are shown (the number of cache misses and their type, the number of memory access, hits, etc.). The code to simulate is introduced in the program as a memory reference sequence, then the user indicates when an Input/Output interruption or DMA is produced [4], the first memory block of the Input/Output process and the number of words to move [5]. Finally, at the simulation time, the user can study the memory content changes and the transitions among the four MESI states.

The sections in this paper are organised as follows: Section 2 describes the educational objectives for the simulator. Section 3 explains the MESI protocol. Section 4 presents some works related to the MESI protocol. Section 5 shows the main characteristics of the MESI simulator, a description of pedagogical issues and some performance examples. Section 6 describes the experimental results in a real teaching environment. Section 7 indicates our future works about the cache memory coherence protocols. Finally, section 8 concludes this paper.

2. Educational objectives

The MESI protocol simulator is widely used in several courses about Computer Architecture, Computer Design and Multiprocessor Systems in the University of Cordoba. The syllabus of these courses includes studying the datapath of a RISC processor, pipelining, the memory hierarchy, superscalar processors or multiprocessor systems. The MESI protocol simulator is an indispensable tool to reach some important objectives in these courses:

- **Introducing** the existing copy-back coherency protocols for cache memory through the understanding of the MESI protocol performance.

- **Understanding** the meaning of each state of the MESI protocol in the cache memory shown by the simulator and the corresponding state in any other cache memory in a multiprocessor environment.

- **Realizing** when a transition between states is needed in order to reflect the actions taken by a processor, an Input/Output interruption or DMA device.

- **Strengthening** the knowledge about hierarchical memory systems, developing experiences with up to three cache levels.

- **Studying** in depth the cache memory parameters, such as the associativity, the replacement policy, the writing policy, etc.

The development of the MESI protocol simulator consists of keeping with some of the principles established by the Computing Curricula 2001 [6]. It helps to include an appropriate experience in the computer engineering curriculum. It also provides an interesting tool for students who are not on campus, such as distance learning and Internet courses.

3. MESI protocol

The MESI protocol makes it possible to maintain the coherence in cached systems. It is based on the four states that a block in the cache memory can have. These four states are the abbreviations for MESI: modified, exclusive, shared and invalid. States are explained below:

- Invalid: It is a non-valid state. The data you are looking for are not in the cache, or the local copy of these data is not correct because another processor has updated the corresponding memory position.

- Shared: Shared without having been modified. Another processor can have the data into the cache memory and both copies are in their current version.

- Exclusive: Exclusive without having been modified. That is, this cache is the only one that has the correct value of the block. Data blocks are according to the existing ones in the main memory.

- Modified: Actually, it is an exclusive-modified state. It means that the cache has the only copy that is correct in the whole system. The data which are in the main memory are wrong.

Figure 1 explains this in a more detailed way.

	M Modified	E Exclusive	S Shared	I Invalid
Is this cache line valid?	Yes	Yes	Yes	No
Memory copy ...	Is out of date	Valid	Valid	-
Scripture in this line	Do not go to the bus	Do not go to the bus	Go to the bus and updates the cache	Go to the bus directly
Is there any copy in other cache?	No	No	Could be	Could be

Figure 1. States and their characteristics.

The state of each cache memory block can change depending on the actions taken by the CPU [7].

Figure 2 presents these transitions clearly.

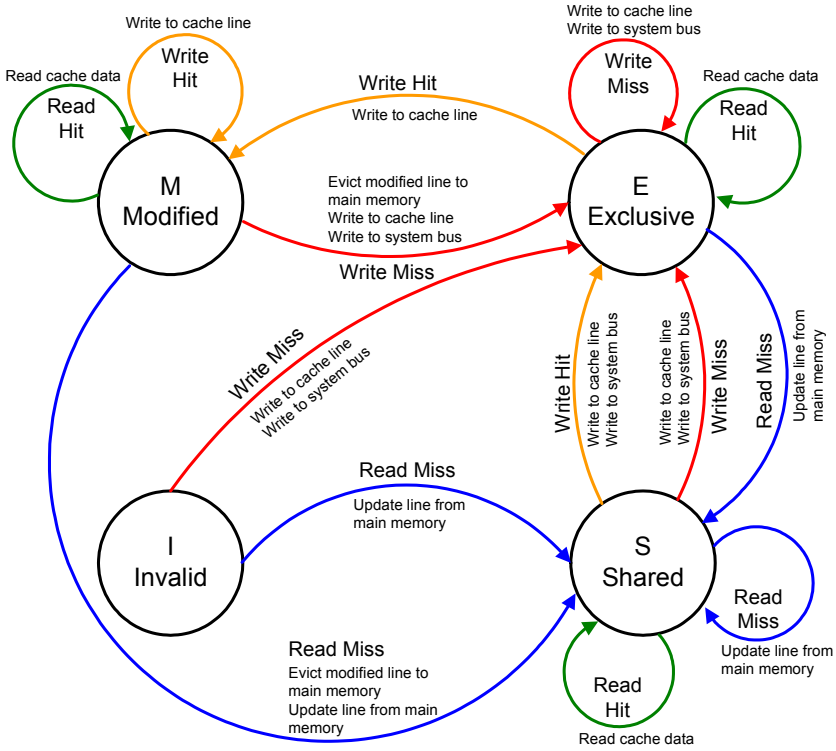

Figure 2. Transitions from CPU bus.

Although the figure 2 is very clear, here is a brief explanation: at the beginning, when the cache is empty and a block of memory is written into the cache by the processor, this block has the exclusive state because there are no copies of that block in the cache. Then, if this block is written, it changes to a modified state, because the block is only in one cache but it has been modified and the block that is in the main memory is different to it.

On the other hand, if a block is in the exclusive state, when the CPU tries to read it and it does not find the block, it has to find it in the main memory and loads it into its cache memory. Then, that block is in two different caches so its state is shared. Then, if a CPU wants to write into a block that is in the modified state and it is not in its cache, this block has to be cleared from the cache where it was and it has to be loaded into the main memory because it was the most current copy of that block in the system. In that case, the CPU writes the block and it is loaded in its cache memory with the exclusive state, because it is the most current version now. If the CPU wants to read a block and it does not find the block in its cache, this is because there is a more recent copy, so the system has to clear the block from the cache where it was and to load it in the main memory. From there, the block is read and the new state is shared because there are two current copies in the system. Another option is that a CPU writes into a shared block, in this case the block changes its state into exclusive. The transitions explained above are represented in figure 2.

It should be taken in account that the state of a cache memory block can change because of the actions of another CPU, an Input/Output interruption or a DMA. These transitions are shown in figure 3.

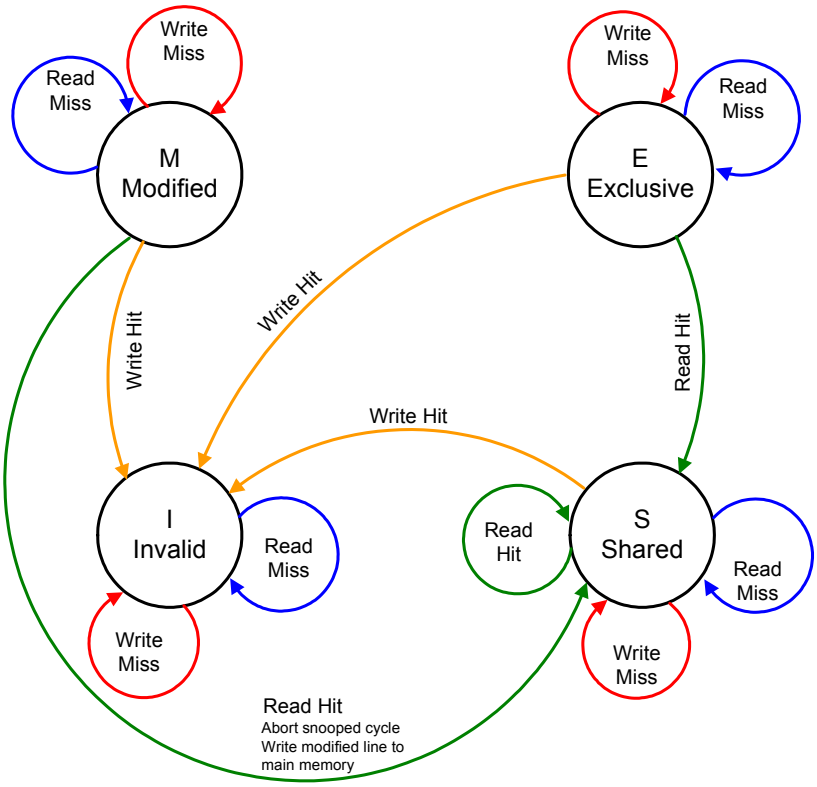

Figure 3. Transitions from System bus.

Hence, the processor is going to use the valid data in its operations. We do not have to worry if a processor has changed data from the main memory and has the most current value of these data in its cache. With the MESI protocol, the processor obtains the most current value every time it is required.

4. Related works

There are some tools with similar functions to the MESI protocol simulator. The outstanding ones are presented in the following lines:

- SMPCache: It simulates cache memory systems in symmetrical multiprocessors. It is useful to study cached multiprocessor systems in parallel computing. One of the configuration options enables using the MESI protocol.

- LIMES: It uses the MESI protocol to maintain the cache memory coherency in parallel multiprocessor systems.

- MINT: It studies the memory hierarchy in multiprocessor systems with shared memory. It uses the MESI protocol, as well as MOESI and MOES protocols.

However, none of them show how the cache memory coherence protocols work. The MESI protocol simulator shows the internal functions of the protocol and the transitions of the different states. That is what makes it to be a useful and unique tool.

5. MESI simulator

5.1. Introduction to the MESI simulator

The MESI protocol simulator is a software system for educational purposes that simulates a run of a software application in a cached multiprocessor system and uses the MESI protocol to maintain the data coherence.

The simulator has a very simple and clear interface that makes it become very easy to use. To start using the application is required to enter a code. This code is a memory reference sequence. These references should be understood as the actions done by a CPU during the execution of a program: instruction readings, data readings and data writings.

During the execution of the code, I/O interruptions will occur. How to introduce these signals is explained below. These I/O interruptions could be interpreted as the actions taken by other processors in a multiprocessor environment or by DMA devices.

5.2. Pedagogical issues

The simulator is helpful for users in some ways. First of all, it can be configured depending on your requirements because it has a lot of configurable parameters. As shown in the figure, the simulator has a typical task bar that makes easier the interaction with the simulator.

The most interesting option is the “Configuration” option. The main memory parameters, cache memory parameters, such as the cache memory capacity, the cache line size, the associativity, the replacement policy, the number of words per memory access, the writing policy, etc. can be configured.

Moreover, the user can configure the statistics of the memory access to study and how they are shown (the number of cache misses and their type, the number of memory access, hits, etc.).

The MESI protocol simulator is a great tool for learning because it generates a lot of useful statistics that can be interpreted by the student and it makes the study of the protocol easier.

In addition, the simulator has two ways of running. The first is the normal one, pushing the start key the simulation begins. But there is another way, step by step, in which the simulator stops and it is possible to see how the system is working at this time.

As seen in figure 4, the simulator has the four states shown at the bottom of the screen. When a code is being run, the states change their colours and the students can see what state is working each time. If this option is used with the different ways of running, it makes the simulator very useful for learning the protocol.

In conclusion, the results are statistics and they can be interpreted by anyone who has some technical knowledge. Therefore the simulator can be used by a person with technical knowledge or by a student who is learning with the application. The code is entered into the simulator as a reference sequence. The simulator has an option which makes possible to introduce Input/output signals and they will appear by inserting them into the code which will be simulated.

5.3. Conflicts generation in memory access

The I/O option allows to introduce I/O signals into the simulator and when to run them. Also, the simulator provides a visualization option, in which can be seen all the statistics that the simulator has made according to the simulated code. Furthermore, it can be seen the content of the main memory, I/O content and the graphics made by the simulator.

Figure 4. Interface of the MESI simulator and statistics screen.

5.4. Performance examples

The following examples show the answer of the simulator by entering a code of real software. This software consists of a matrix multiplication. Two different configurations are presented.

5.4.1. Configuration 1

A cache memory with the following configuration in level 1: Block size: 8; cache size: 1kb; replace: LRU; Mapping: Associative by groups: 8; Unified.

This configuration has a 91% correct reading. Although it is a very simple configuration, it has a very low failure rate. This is because it is not necessary to replace blocks, which is one of the main fault causes. It depends on the chosen replacement because you can replace a block which is going to be read soon.

5.4.2. Configuration 2

A cache memory hierarchy with the following configuration in level 1: Block size: 4; cache size: 1kb; replace: random; Mapping: Direct; Mixed. The configuration in level 2 is: Block size: 8; cache size: 4kb; replace: LRU; Mapping: Associative by groups: 8; Unified.

This configuration has a 84% correct reading in level 1 of the cache and 43% in level 2. Level 1 has failed the 7 times that the system has accessed level 2. Level 2 has a block size bigger than the first one. With each failure of level 2, the system loads more words than when level 1 fails. So, it is more likely to find a word in level 2.

This configuration presents more failures than the previous one. The reasons are the following:

Block size: The main reason is this one because with the previous configuration, with each failure, the system loads eight pages per block; now, when a failure is produced, the system loads four pages with each block.

Replacement policy: A random policy is used in level 1, and the most effective policy is the LRU one.

Distribution: This configuration is mixed, so there are more failures than in the unified one, because with the mixed configuration, in each failure, the system loads a block of data or instructions. On the other hand, with the unified configuration, the system loads data and instructions.

6. Experimental results

The MESI simulator has been used in a real teaching environment at the University of Cordoba (Spain). Up to 60 students have made use of the MESI simulator with up to 4 different simulator configurations. Approximately 1200 tests have been executed for 10 hours, averaging 120 tests per hour; 92.5 % of those tests being successful. The 7.5 % remaining tests failed because students made a wrong introduction of I/O signals or syntactic failures in the code. In future versions, a precompiler will detect these failures and will notice to fix them.

Students have been working with different configurations and I/O signals, which show every transition of the MESI protocol. They have made their own conclusions in the practice report and, definitely, they have taken a good understanding of the MESI protocol performance.

Table 1 presents the statistics about general usage (for all the 60 students) and for a specific student (Student_1) picked at random.

	General usage	*Student_1*
Connections	1187	21
Failed	89	2
Successful	1098	19
Invalid to Shared transitions	141	3
Invalid to Exclusive transitions	145	2
Shared to Invalid transitions	143	2
Shared to Exclusive transitions	109	2
Exclusive to Modified transitions	151	3
Exclusive to Shared transitions	120	2
Exclusive to Invalid transitions	117	2
Modified to Shared transitions	76	1
Modified to Exclusive transitions	49	1
Modified to Invalid transitions	46	1

Table 1. Statistics about general usage and for a specific student.

7. Future works

The MESI protocol simulator is the first step in a strong effort to develop pedagogical tools. It is an evidence that the students achieve a better understanding of theoretical concepts through a direct experience. Hence, to use a simulator is the best way to see what is happening in a cache memory hierarchy.

In future developments, an extended version of the MESI protocol simulator will present other coherency protocols [7] such as MOESI, N+1, Futurebus+, Berkeley, etc.

8. Conclusions

The presented software is a very suitable tool to show the cache memory occupation from memory references during the execution of a program. It is specially indicated to teach the cache memory concepts like the capacity, the cache line size, the associativity, the replacement policy, etc. Besides, it shows other concepts like the input/output interruptions and how these interruptions determine the transference of information (by applying it). On these cases, the understanding of the MESI memory coherence protocol is possible by showing the different stages of the protocol. Experimental results confirm the perfect applicability in a real classroom environment.

References

[1] F.J. Jiménez Maturana, *Simulador de memoria caché con aplicación del protocolo de coherencia MESI*. Escuela Politécnica Superior. Universidad de Córdoba. Spain. 2003.

[2] C. Hamacher, Z. Vranesic, S. Zaky, *Organization of Computers*. McGraw-Hill/Interamericana of Spain S.A. 2003. 804 p. ISBN: 84-481-3951-8.

[3] A. S. Tanembaum, *Organización de Computadores. Un enfoque estructurado*. 3ª Ed. Prentice Hall. 1992. pp. 327-333.

[4] W. Stalling, *Computer Architecture*. 5th Ed. Prentice-Hall. 2000. 760 p. ISBN: 84-205-2993-1.

[5] D. A. Patterson, J. L. Hennessy, *Structures and Design of Computers. Interficie circuitería/programación.* Edition 2000. Spain. Editorial Reverté, S.A. 2000. 758 p. ISBN: 84-291-2617-1.

[6] A. McGettrick, M. D. Thies, D. L. Soldan, P. K. Srimani, *Computer Engineering Curriculum in the New Millennium*. IEEE Transactions on Education, vol. 46, no. 4, November 2003.

[7] J. Handy, *The Cache Memory Book*. 2nd Ed. Academic Press. 1998. 229 p. ISBN: 0-12-322980-4.