International Journal of Electrical and Computer Engineering (IJECE)
Vol. 11, No. 2, April 2021, pp. 1043~1052
ISSN: 2088-8708, DOI: 10.11591/ijece.v11i2.pp1043-1052

Cache coherency controller for MESI protocol based on FPGA

Mays K. Faeq, Safaa S. Omran
Electrical Engineering Technical College, Middle Technical University, Iraq

Article Info

Article history:

Received Apr 15, 2020
Revised Sep 3, 2020
Accepted Sep 15, 2020

Keywords:

Cache coherency
FPGA
MESI protocol
MIPS processor
VHDL

ABSTRACT

In modern techniques of building processors, manufactures using more than one processor in the integrated circuit (chip) and each processor called a core. The new chips of processors called a multi-core processor. This new design makes the processors to work simultanously for more than one job or all the cores working in parallel for the same job. All cores are similar in their design, and each core has its own cache memory, while all cores shares the same main memory. So if one core requestes a block of data from main memory to its cache, there should be a protocol to declare the situation of this block in the main memory and other cores.This is called the cache coherency or cache consistency of multi-core. In this paper a special circuit is designed using very high speed integrated circuit hardware description language (VHDL) coding and implemented using ISE Xilinx software. The protocol used in this design is the modified, exclusive, shared and invalid (MESI) protocol. Test results were taken by using test bench, and showed all the states of the protocol are working correctly.

Corresponding Author:

Mays K. Faeq
Electrical Engineering Technical College
Middle Technical University
Baghdad, Iraq
Email: maysoomays@gmail.com

1. INTRODUCTION

Processors today are manufactured as multi-core processors; all cores are similar in their design and each has its cache memory and all of these cores are in one chip. All these cores shared one main memory located outside the chip [1-4]. In these kinds of systems, the cache plays very important part in that kind of processor design. The processor asks for data firstly from the cache and if the data is not present in the processor cache the processor will fetch that data from the main memory and put it in the cache. These data may be exisit in anothers core and that core works on that data and changes its value, hence the data exisit in the main memory are invalid and the data must be updated. Hence a protocol of cache coherency should be applied for the caches of the cores [5-8].

So that a snooping protocol should be applied to ensure that no core will use invalid data [9]. Different protocols were used for different system. One of these protocols is the MESI protocol which firstly used in the Pentium processor [10-12]. In modified, exclusive, shared and invalid (MESI) protocol the data represented by a block in the cache will be in state Modified or Exclusive or Shared or Invalid. A cache controller will makes snooping for all the caches of each core and updates its state for example from Invalid to shared or from shared to exclusive and so on [13, 14].

Hence the problem in this kind of system is to design a cache coherency circuit to snoope the system and apply the used protocol which is in this design is the MESI protocol [15]. The design implemented using very high speed integrated circuit hardware description language (VHDL) then integrated with field programmable gate arrays (FPGA) Xilinx Spartan 6 [15]. The results for the different parts of the processor are presented in the form of test bench waveform and the architecture of the system is demonstrated and the result was matched with theoretical result.

2. RESEARCH METHOD
2.1. The cache

The cache is a static memory while the memory which is used in the main memory is of type dynamic. The cache is faster than the RAM in main memory by a factor of (8-10) times, but the size of the cache is greater than RAM. To store one bit of data in cache it requires 6 transistors while in dynamic RAM it requires only one transistor. Because of that the designers put a small size from the faster memory (cache) inside the processor and a large size of danamic RAM outside the chip of the processor in the mother board. The processor always will ask for data from the cache and if it is exisits in the cache the processor will fetch the data in one bus cycle (2 clks) and this case is called read hit. If the data not exisit in the cache the processor must fetch it from main memory with extra clock cycles and this case is called read miss [16-20].

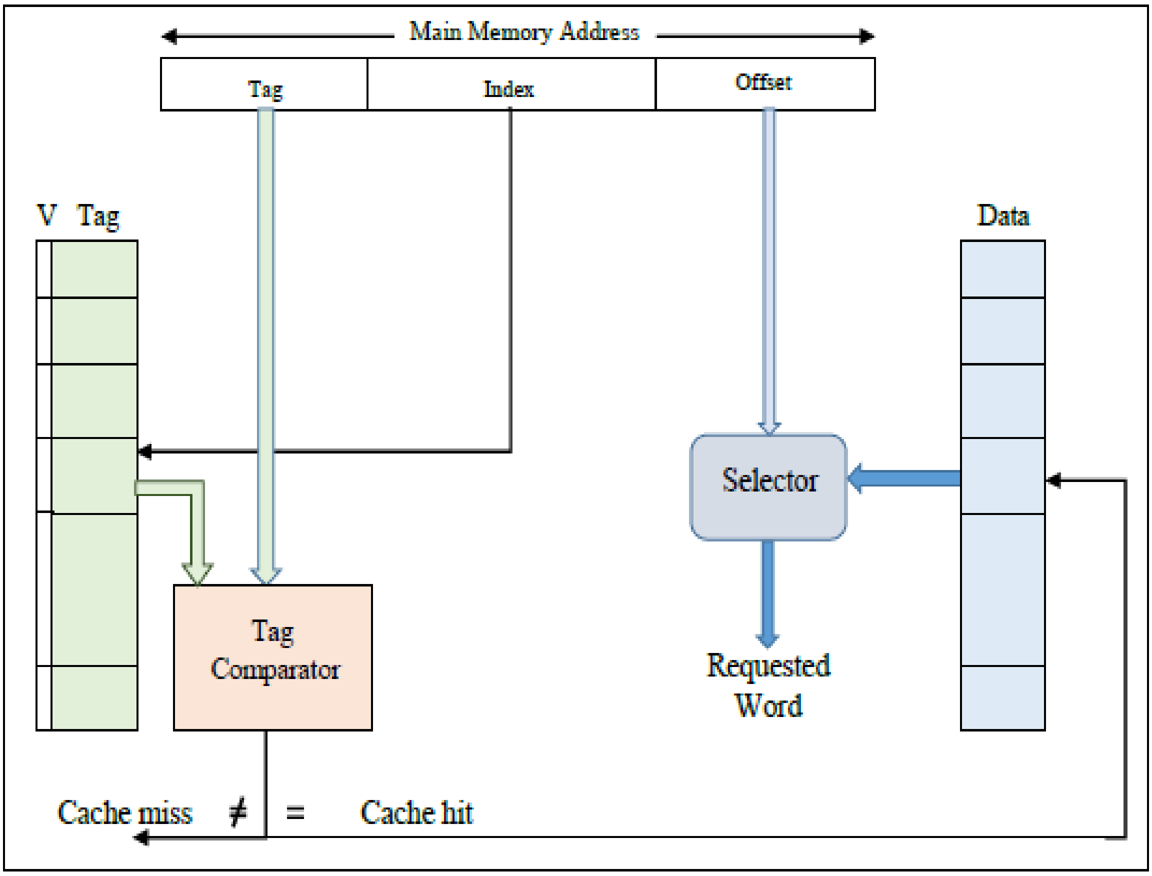

Figure 1. Direct mapped cache protocols

2.2. Architecture

There are three different types of cache organization. In this paper the direct mapped organization is used because it is easy and simple in design [21, 22]. Figure 1 shows a simple direct mapped cache organization desighed for the purpose of testing the cache coherence protocol. As shown in Figure 1, the main memory address is divided in to three parts, which they are the offset, index and the tag. In the designed cache the addres is partitioned to 4-bits for the offset as bits (0-1) for byte select and bits (2-3) for word select and bit (4-5) are used for index. The rest 26-bits are used for the tag [23-25]. Figure 2 shows the design of the used direct-mapped cache.

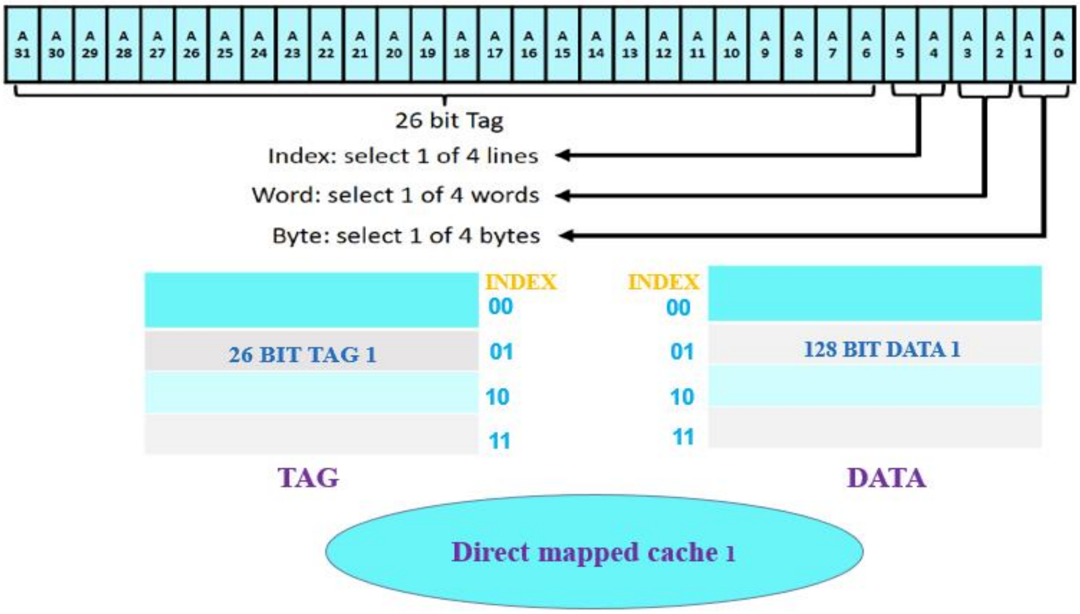

Figure 2. Design direct mapped cache

3. MESI PROTOCOL

This protocol consists of four different states, which they are invalid (I), shared (S), exclusive (E), and modified (M). This protocol is an advance to the previous MSI protocol. The new state exclusive (E) is added in order to reduce the number of bus messages. The Exclusive state means that the block or time of data a valid in the cache and main memory and not valid in other caches ,which give a flexibility to the processor to modify its cache without a need to snoop other caches [26-29]. Figure 3 shows a state transition diagram for the MESI protocol. In the left side of the figure represents the processor requests and the action of the cache controller circuit, while the right part of the figure represents the bus requests and corresponding actions [30, 31].

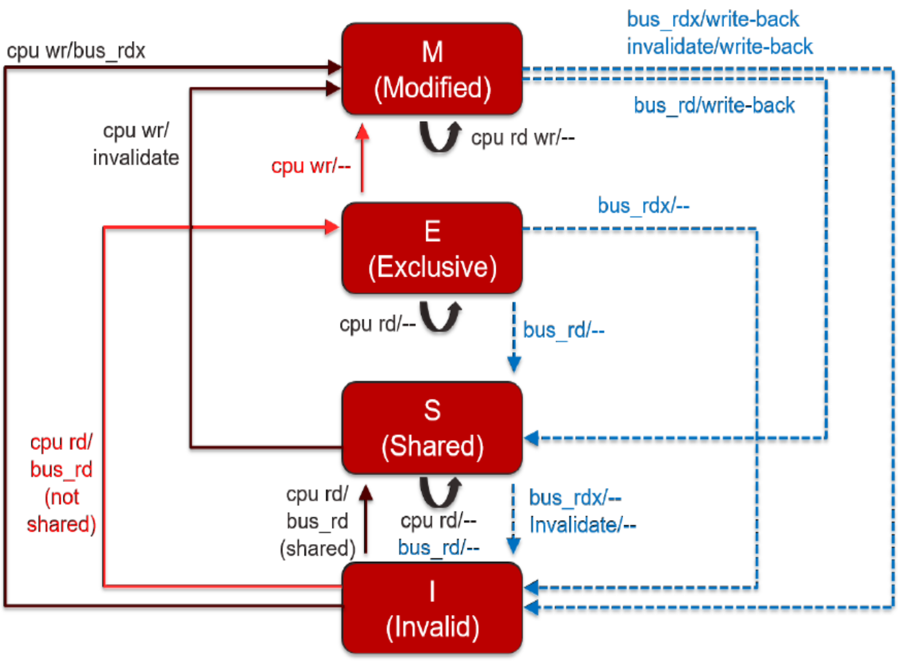

Figure 3. State transition of the MESI protocol

Figure 4 shows simplified MESI state diagram with transition states [32]. In this section, it is explained the results of research and at the same time is given the comprehensive discussion. Results can be presented in figures, graphs, tables and others that make the reader understand easily [2, 5]. The discussion can be made in several sub-chapters.

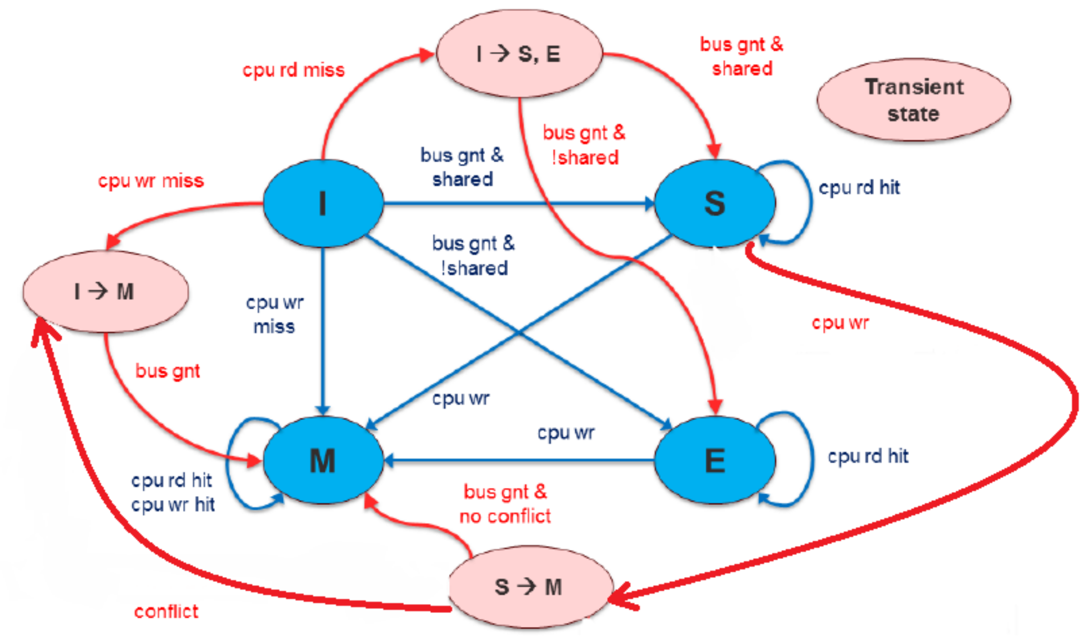

Figure 4. Simplified MESI state diagram with transition states

4. SYSTEM DESIGN

4.1. VHDL top_level implementation

A VHDL components of MIPS processor which was designed by [33] is combined with the VHDL components of this design, by using (Xilinx ISE Design Suite 14.1) all these components are connected together in order to compose the top level, later a test bench is written and used to enter the 2-bits of cache size controller and execute a written test program. Figure 5 shows Top_level and Figure 6 Shows the schematic view of top_level components for the designed system. It consists of three parts: pipelined MIPS, data memory system and instruction memory system.

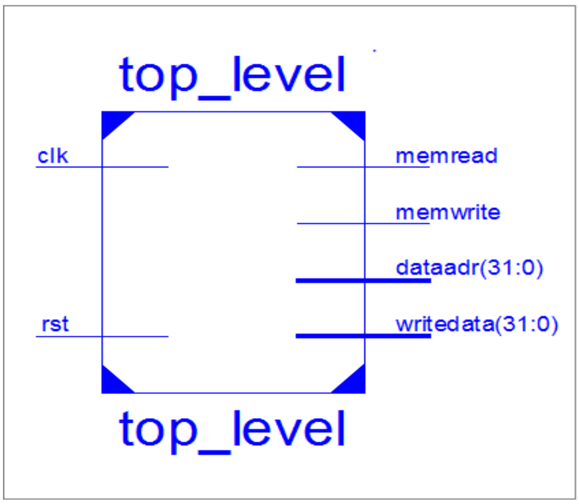

Figure 5. Top_level

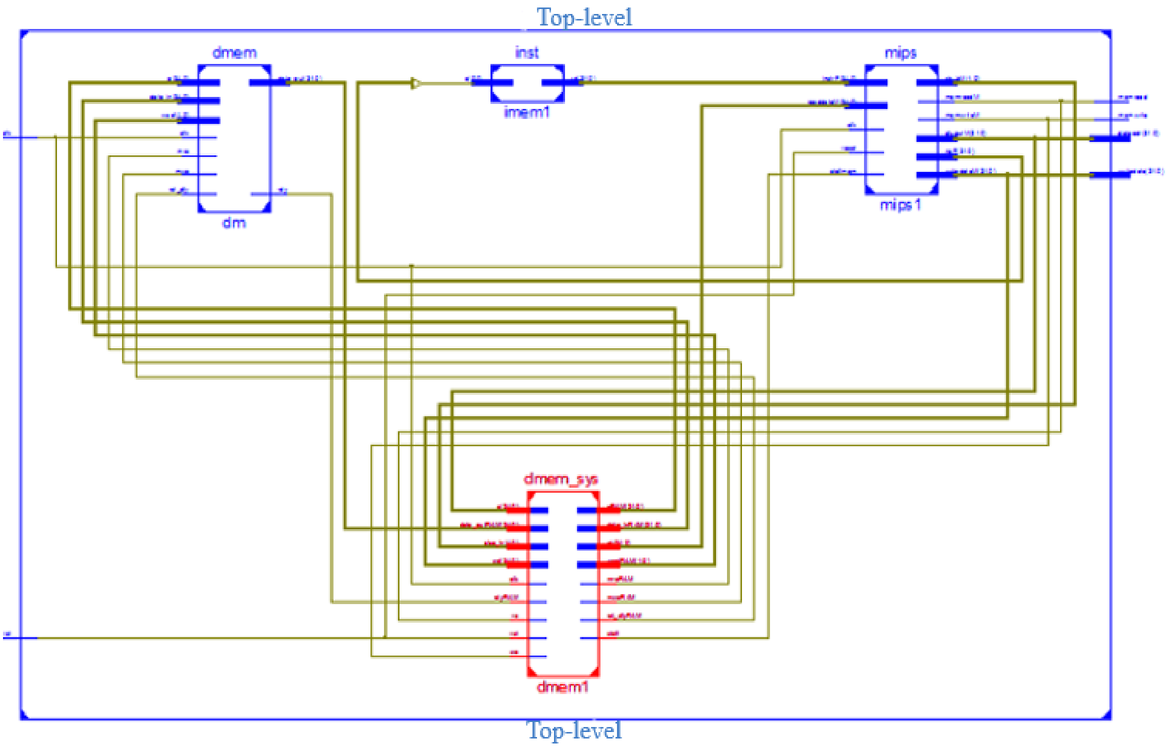

Figure 6. Schematic view of top-level components

4.2. Design cache coherency

In this work two microprocessors MIPS1 and MPIS2 were designed and each with separate cache. A cache coherency controller is designed which consist of two parts, the coherency tag and coherence controller by using FSM. All these components are connected together on chips and have the main memory of chip. Figure 7 shows the design for the cache coherency protocol. Tag cache: Data tag cache has 28-bits (26 tag bits, 2-bits for MESI protocol) for each data cache line. MESI bits are reset when the machine restart. Instruction tag cache contains 27-bits, it is similar to instruction tag cache in single core [34]. Figure 8 shows the RTL Coherence tag and coherence controller.

In this paper a 7-bits where used to indicate different states for MESI protocol, two bits for M(Modify), two bits for E (Exclusive), one bit for S(Shared) and two bits for I (invalid). Table 1 shows MESI states. Figure 9 shows the coherency tag with 7-bits for MESI states. Table 2 shows the different sates of MESI Protocol.

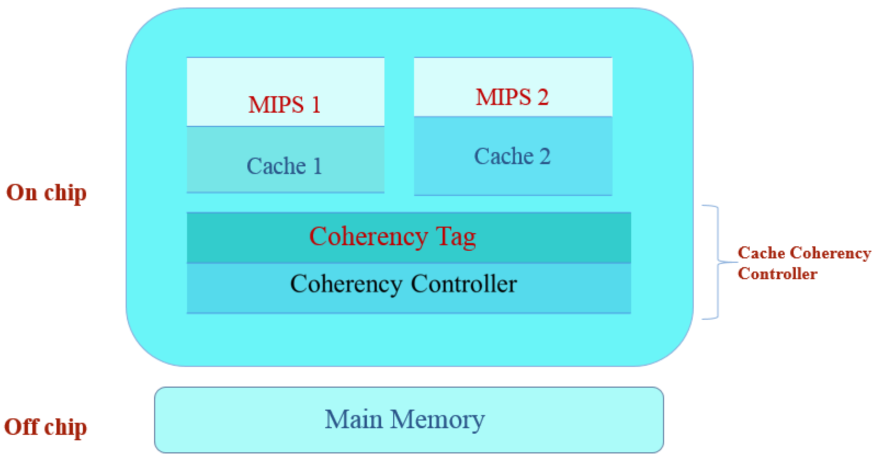

Figure 7. Design cache coherency protocol

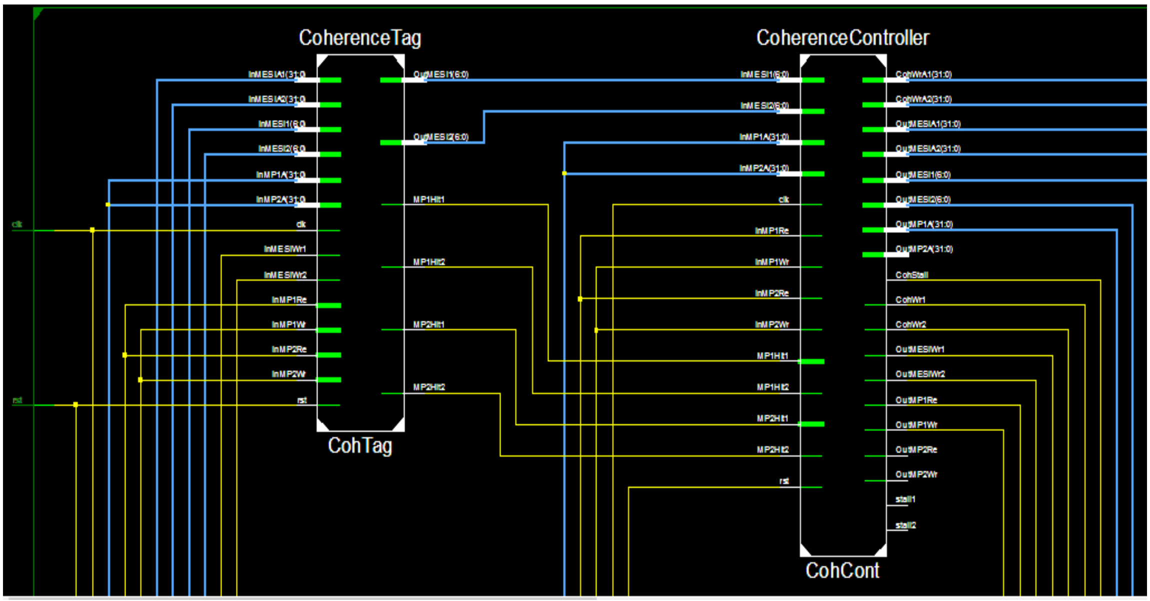

Figure 8. RTL coherence tag and coherence controller

Table 1. MESI states

M (Modify)	E (Exclusive)	S (Shared)	I (Invalid)
	MODIFY		
00	NOT MODIFIED		
01	MODIFIED FIELD MP1		
10	MODIFIED FIELD MP		
	EXCLUSIVE		
00	NOT EXCLUSIVE		
01	EXCLUSIVE FIELD MP1		
10	EXCLUSIVE FIELD MP2		
	SHARED		
0	NOT SHARED		
1	SHARED		
	VALID		
00	VALID BOTH		
01	NOT VALID MP1 (IN VALID 1)		
10	NOT VALID MP2 (IN VALID 2)		

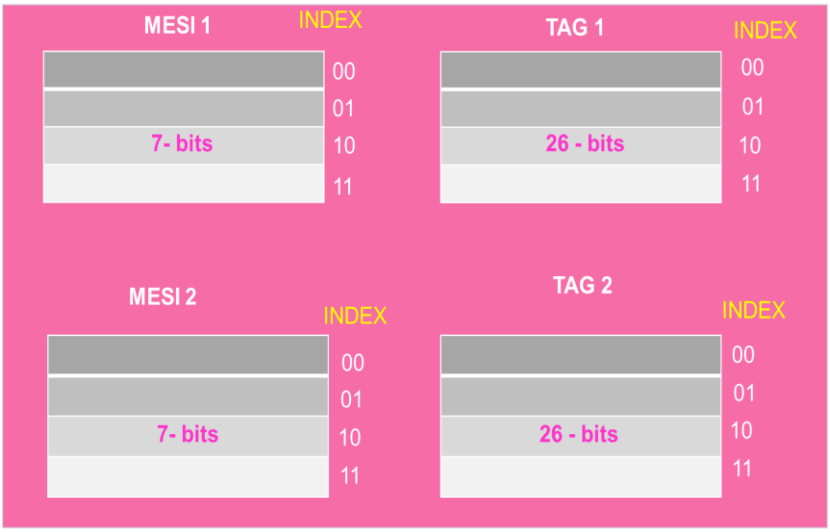

Figure 9. Coherency tag with 7-bits MESI states

Table 2. Sates of MESI protocol

		MP1 HIT Read (Direct Read)		
State	M (*Modify*)	E (*Exclusive*)	S (*Shared*)	I (*Invalid*)
St0	00	00	0	00
OutMESI1	**00**	**01**	**0**	**00**
	00	01	0	00
St1	01	00	0	00
	00	00	1	00
St2	00	10	0	00
OutMESI1	**00**	**00**	**1**	**00**
St3	10	00	0	00
OutMESI1	**00**	**10**	**0**	**00**
		MP1 HIT Write (Direct Write)		
State	M (*Modify*)	E (*Exclusive*)	S (*Shared*)	I (*Invalid*)
	00	00	0	00
St8	00	01	0	00
	00	10	0	00
	00	00	1	00
	01	00	0	00
OutMESI1	**01**	**00**	**0**	**00**
St9	01	00	0	00
OutMESI1	**01**	**00**	**0**	**00**
		MP1 MISS Read (Direct Read)		
State	M (*Modify*)	E (*Exclusive*)	S (*Shared*)	I (*Invalid*)
0St12	00	01	0	00
		MP1 MISS Write (Direct write)		
State	M (*Modify*)	E (*Exclusive*)	S (*Shared*)	I (*Invalid*)
St13	01	00	0	00
		MP2 HIT Read (Direct Read)		
State	M (*Modify*)	E (*Exclusive*)	S (*Shared*)	I (*Invalid*)
St4	00	00	0	00
OutMESI2	**00**	**10**	**0**	**00**
	00	10	0	00
St5	10	00	0	00
	00	00	1	00
St6	00	01	0	00
OutMESI2	**00**	**00**	**1**	**00**
St7	01	00	0	00
OutMESI2	**00**	**01**	**0**	**00**
		MP2 HIT Write (Direct Write)		
State	M (*Modify*)	E (*Exclusive*)	S (*Shared*)	I (*Invalid*)
	00	00	0	00
St10	00	01	0	00
	00	10	0	00
	00	00	1	00
	10	00	0	00
OutMESI2	**10**	**00**	**0**	**00**
St11	01	00	0	00
OutMESI2	**10**	**00**	**0**	**00**
		MP2 MISS Read (Direct Read)		
State	M (*Modify*)	E (*Exclusive*)	S (*Shared*)	I (*Invalid*)
St14	00	10	0	00
		MP2 MISS Write (Direct write)		
State	M (*Modify*)	E (*Exclusive*)	S (*Shared*)	I (*Invalid*)
St15	10	00	0	00
St400		**DO NOTHING**		

0 : MIPS1; 0 : MIPS2

4.3. RTL schematic of multi-core MIPS processor design

Two single core MIPS processors are combined together to generate a multicore MIPS processor. Each single core is a pipelined MIPS processor and has its own L1 cache memory. Both cores shared the main memory. Multicore processor exploits the parallel available in program to allow its cores to work together for the same job, therefor parallel program is needed to reach better performance from multicore processor. Since each core has L1 cache, then the same memory address may be found in both cores. This may cause consistency problems in data cache. Instruction cache does not have this problem because the processor cannot modify program instructions. In this paper snooping-based coherency is used as a cache coherency mechanism, and the coherency protocol used is MESI protocol. Figure 10 shows the RTL cache coherency controller for MESI protocol.

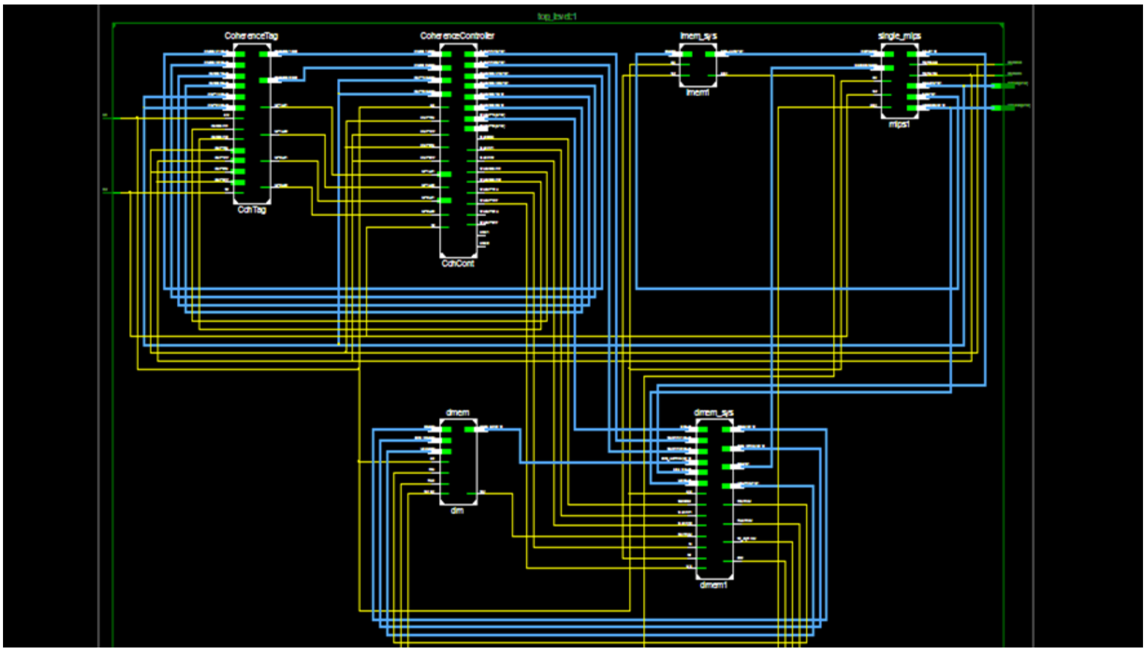

Figure 10. RTL cache coherency controller for MESI protocol

5. RESULTS AND DISCUSSIONS

To verify the validity of the design that was built in this paper, multiple programs were written for the purpose of examining the work of the MESI protocol by the coherency controller. The results were found to be identical to the different 15 states that designed in the protocol, Figure 11 shows the test bench results for some of these states.

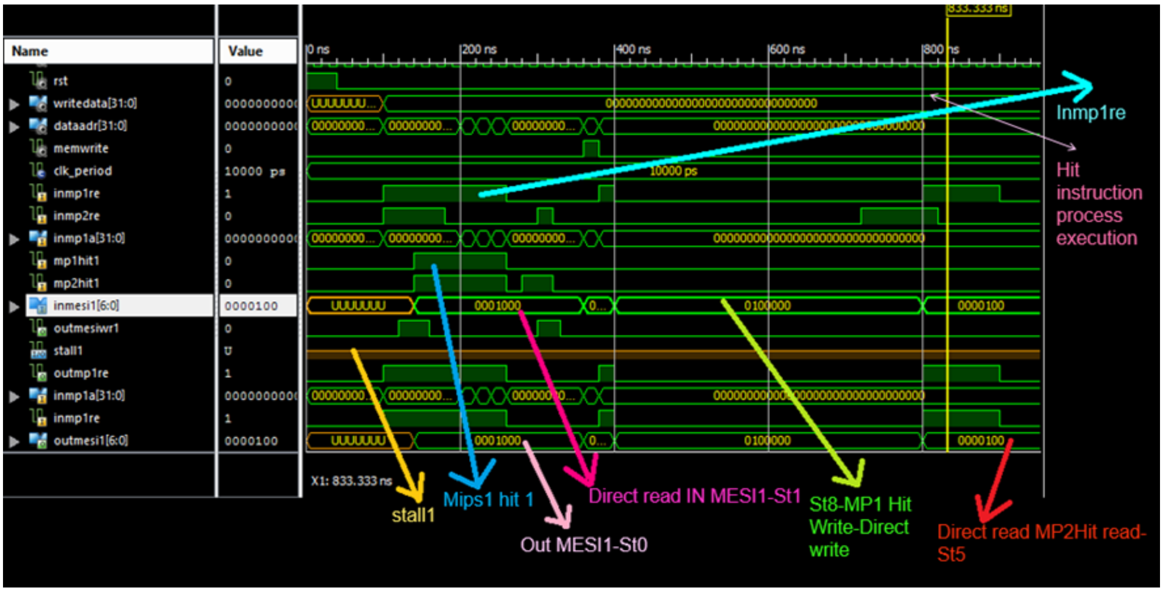

Figure 11. Test program execution

6. CONCLUSION

In this paper, a pre-designed MIPS type processor was used in the department; this processor is used to build another processor corresponding to it so that to have two cores and a cache was built for each processor and the type of mapping were used direct mapped type for ease of design. Then building a circuit of coherence controller in order to control the reading and writing processes for processors when reading and writing from the shared memory of each of the processors. A link to all the designed parts, and several programs were written for the purpose of operating and examining the MESI protocol, and the results were found to be compatible with the topic design and the purpose of this research for use in the scientific purposes of master's students in advanced computer technique Lab.

REFERENCES

[1] P. D. Devi and T. Ravindra, "Implementation of MESI Protocol Using Verilog," *International Journal of Advanced Research in Computer and Communication Engineering*, India, vol. 5, no. 10, pp. 228-232, 2016.

[2] G. E. Moore, "Cramming More Components onto Integrated Circuits," *Electronics*, vol. 38, no. 8, pp. 114, 1965.

[3] X. Lai, C. Liu, Z. Wang and Q. Feng, "A Cache Coherence Protocol Using Distributed Data Dependence Violation Checking in TLS," *2012 Second International Conference on Intelligent System Design and Engineering Application*, Sanya, Hainan, 2012, pp. 5-10.

[4] A. Ros, M. E. Acacio, and Jos´e M. Garc´ıa, "A direct coherence protocol for many-core chip multiprocessors," Parallel and Distributed Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 12, pp. 1779-1792, 2010.

[5] A. Saparon and F. N. Razlan, "Cache Coherence Protocols in Multi-Processor," In *International conference on Computer Science and Information Systems (ICSIS)*, Dubai (UAE), 2014, pp. 17-18.

[6] Y. J. Jang and W. W. Ro, "Evaluation of cache coherence protocols on multi-core systems with linear work loads," *2009 ISECS International Colloquium on Computing, Communication, Control, and Management*, Sanya, 2009, pp. 342-345.

[7] J. Li, P. Yang, N. Ding, H. Guan, and Jianpei, "A New Kind of Hybrid Cache Coherence Protocol for Multiprocessor with D-Cache," *2011 International Conference on Future Computer Science and Education*, Xi'an, 2011, pp. 641-645,.

[8] N. Chaturvedi, P. Sharma, and S. Gurunarayanan, "An adaptive coherence protocol with adaptive cache for multi-core architecture," *2013 International Conference on Advanced Electronic Systems (ICAES)*, Pilani, 2013, pp. 197-201.

[9] J. L. Hennessy and D. A. Patterson, "Computer architecture: a quantitative approach," *Elsevier, Fourth Edition*, 2012.

[10] M. Dalui, T. Som, S. Bansal, S. Pant, and B. K. Sikdar, "MASI: An eviction aware cache coherence protocol for CMPs," *2016 Sixth International Symposium on Embedded Computing and System Design (ISED)*, Patna, 2016, pp. 249-253.

[11] Z. Al-Waisi and M. O. Agyeman, "An overview of on-chip cache coherence protocols," *2017 Intelligent Systems Conference (IntelliSys)*, London, 2017, pp. 304-309.

[12] D. J. Lilja, "Cache coherence in large-scale shared-memory multiprocessors: Issues and comparisons," *ACM Computing Surveys (CSUR)*, vol. 25, no. 3, pp. 303-338, 1993.

[13] T. M. Austin, "DIVA: a reliable substrate for deep submicron microarchitecture design," *MICRO-32. Proceedings of the 32nd Annual ACM/IEEE International Symposium on Microarchitecture*, Haifa, Israel, 1999, pp. 196-207.

[14] A. Asaduzzaman and K. K. Chidella, "A novel directory based hybrid cache coherence protocol for shared memory multiprocessor," *2016 IEEE International Symposium on Phased Array Systems and Technology (PAST)*, Waltham, MA, 2016, pp. 1-6.

[15] D. L. Perry, "VHDL: Programming by Example," *4th ed., America: McGraw-Hill*, 2002.

[16] A. Sravanthi, C. R. Rao, K. K. Raju, and L. Rambabu, "Implementation of MESI Protocol using Verilog," *International Research Journal of Engineering and Technology (IRJET)*, vol. 06, no. 6, pp. 1763-1777, 2019.

[17] N. B. Mallya, G. Patil, and B. R. Aveendran, "Simulation based performance study of cache coherence protocols," *2015 IEEE International Symposium on Nanoelectronic and Information Systems*, Indore, 2015, pp. 125-130.

[18] A. D. Joshi and N. Ramasubramanian, "Comparison of significant in multicore cache coherence," *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, Noida, 2015, pp. 108-112.

[19] A. D. Joshi, S. Indrajeet, N. Ramasubramanian, and B. S. Begun, "Analysis of multi-core cache coherence protocols from energy and performance perspective," *2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE)*, Bhopal, 2017, pp. 381-388.

[20] Z. Wang and R. B. Lee, "A Novel cache architecture with enhanced performance and security," *2008 41st IEEE/ACM International Symposium on Microarchitecture*, Lake Como, 2008, pp. 83-93.

[21] P. Shree and Anitha V., "Design and implementation of direct mapped cache memory with same tag bit information," *International journal of computer science and mobile computing*, vol. 4, no. 6, pp. 978- 983, 2015.

[22] D. A. Patterison, J. I. Hennessy, "Computer organization and design," *4th edition, Morgan Kaufman*, 1998.

[23] A. Agarwal, R. Simoni, J. Hennessy, and M. Horowitz, "An evaluation of directory schemes for Cache coherence," *1988 The 15th Annual International Symposium on Computer Architecture. Conference Proceedings*, Honolulu, HI, USA, 1988, pp. 280-298.

[24] S. H. Pugsley, J. B. Spjut, D. W. Nellans, and R. Balasubramonian, "SWEL:Hardware cache coherence protocols to map shared data onto shared caches," *2010 19th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, Vienna, 2010, pp. 465-475.

[25] S. Al-Hothali, S.S oomro, K. Tanvir, and R. Tuli, "Snoopy and Directory Based Cache Coherence Protocols: A Critical Analysis," *Journal of Information & Communication Technology*, vol. 4, no. 1, pp. 1- 11, 2010.
[26] M. M. K. Martin, M. D. Hill, and D. Wood, "Token coherence: decoupling performance and correctness," *30th Annual International Symposium on Computer Architecture, 2003. Proceedings,* San Diego, CA, USA, 2003, pp. 182-193.
[27] S. Sun, H. An, and J. Chen, "Cache Coherence Method for Improving Multi-threaded Applications on Multicore Systems," *2014 6th International Conference on Multimedia, Computer Graphics and Broadcasting*, Haikou, 2014, pp. 47-50.
[28] R. E. Ahmed and M. K. Dhodhi, "Directory-Based cache coherence protocol for power-aware chip-multiprocessors," *2011 24th Canadian Conference on Electrical and Computer Engineering (CCECE),* Niagara Falls, ON, 2011, pp. 1036-1039.
[29] D. P. Kaur and V. Sulochana, "Design and Implementation of Cache Coherence Protocol for High-Speed Multiprocessor System," *2018 2nd IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, Delhi, India, 2018, pp. 1097-1102.
[30] K. Venkateshwar, "Formal Verification of a MESI-based Cache Implementation," (Master of Science) Texas A&M University, Agglead, Houston-Dallas, 2017.
[31] J. Handy, "The Cache Memory Book," *Academic Press*, 1998.
[32] G. Luna, H. Gómez, and B. Benítez, "MESI Cache Coherence Simulator for Teaching Purposes," *CLEI Electronic Journal*, vol. 12, no. 1, pp. 1-7, 2009.
[33] H. S. Mahmood, "VHDL Implementation of a Pipelined RISC Processor for Educational Purposes," MSc.thesis, College of Electrical and Electronic Engineering Techniques, Baghdad, Iraq, 2014.
[34] S. S. Omran, A. J. Ibada, "FPGA implementation of MIPS RISC processor for educational purposes," *Journal of University of Babylo*n, vol. 24, no. 7, pp. 1745-1761, 2016.

BIOGRAPHIES OF AUTHORS

Mays Kifah Faeq was born in Kirkuk, Iraq in 1993. She graduated from Al-Qalam University college in 2016, and now studying master at Electrical Engineering College/Middle Technical University, Baghdad, Iraq, her main interest in Computer Architecture Design, Computer engineering, embedded system and design.

Safaa S. Omran was born in Baghdad, Iraq in 1956. He graduated from University of Baghdad in 1978, and then he got the MSc from the same University in 1984. He is now a professor working at the Electrical Engineering College/Middle Technical University, Baghdad, Iraq. His main interest working researches are in the field of microprocessor design for embedded systems, Image processing and cryptography system design.