Open Access
Baghdad Science Journal Vol.14(1)2017

DOI: http://dx.doi.org/10.21123/bsj.2017.14.1.0219

Cache Coherence Protocol Design and Simulation Using IES (Invalid Exclusive read/write Shared) State

*Luma Fayeq Jalil** *Maha Abdulkareem H. Al-Rawi***
*Abeer Diaa Al-Nakshabandi****

*Department of Computer Sciences, University of Technology, Baghdad, Baghdad, Iraq.
**Department of Computer Sciences, University of Technology, Baghdad, Baghdad, Iraq.
***Head of programmers oldest, Distribution office At Ministry of Electricity, Baghdad, Baghdad, Iraq.

E-mail: abeerdiaaphd@gmail.com

Received 25/ 5/2016
Accepted 26/7 /2016

Abstract

To improve the efficiency of a processor in recent multiprocessor systems to deal with data, cache memories are used to access data instead of main memory which reduces the latency of delay time. In such systems, when installing different caches in different processors in shared memory architecture, the difficulties appear when there is a need to maintain consistency between the cache memories of different processors. So, cache coherency protocol is very important in such kinds of system. MSI, MESI, MOSI, MOESI, etc. are the famous protocols to solve cache coherency problem.

We have proposed in this research integrating two states of MESI's cache coherence protocol which are Exclusive and Modified, which responds to a request from reading and writing at the same time and that are exclusive to these requests. Also back to the main memory from one of the other processor that has a modified state is removed in using a proposed protocol when it is invalidated as a result of writing to that location that has the same address because in all cases it depends on the latest value written and if back to memory is used to protect data from loss; preprocessing steps to IES protocol is used to maintain and saving data in main memory when it evict from the cache. All of this leads to increased processor efficiency by reducing access to main memory.

Key words: Cache Coherence Problem, Snooping Protocol, Directory-Based Cache Protocols, MESI, Cache Simulator, Dev. C++, Multiprocessor, Shared Memory.

Introduction:

The difference between main memory and the processor speed may take a lot of processor cycles to get to the main memory, which may cause an obstacle in performance in modern processor architecture [1]. So to deal with this problem, installing faster memory in such systems can store data from main memory to be used more often by the processor. These fast memories, which are either on chip or off-chip, used to improve latency and performance, are called cache memories [2].

High inconsistency which causes cache coherency problem might appear if there is any change in the shared data in a multiprocessor system having different processors with a different cache memories that share data in the main memory. Coherence between these caches is achieved through the application of two_hardware_based protocols which are snoopy and directory_based protocols [3, 4].

This paper gives the basic idea of how to simulate multi-level caches to implement IES protocol using DEV C++ language. To simulate caches, the number of cache levels and the parameters of a cache memory such as the "cache memory capacity" for each cache level, "the cache line size", "associativity", "the replacement policy", "the number of words per memory access", "the writing policy", etc. are determined. Finally, the processor efficiency is calculated from either finding the input address in the caches to become "hit" or not finding that address which gets from t`he main memory to become "Miss".

The Basic Concepts in Caching:

The caching data exploiting the locality in memory hierarchies get the illusion of a large and fast memory. There are two types of locality that benefit memory [5]. In most programs, the same address of memory is used repeatedly for reading or writing by processor. So, temporal locality appears as a result of a high degree of locality in these programs. Another feature called spatial locality is that if a processor reads or writes a memory location then there is a probability to read or write nearby locations also. To exploit the second behavior, caches may operate by holding a group of neighboring data known as cache lines (also called cache blocks) [6,7]. Memory hierarchy illustrated in Figure 1:

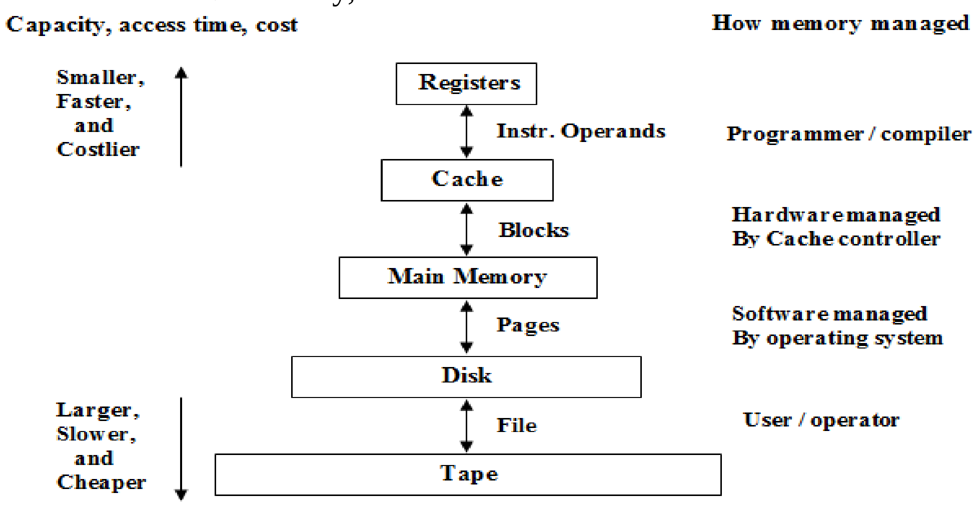


Fig. 1: The Memory Hierarchy [1,2,3,5]

Cache Coherency:

Cache coherency is the consistency and validation of the data value in the caches of a multi core processor such that any reading of a memory location via any caches will return the most recent data written to that location via any caches [8]. The exact replacement time and update method to these data is captured by the write policy. The main write policies include two: write through is the first policy that writes in both main memory and cache to become valid [3]. The other is a Write back which is not written into main memory unless another cache needs that cache line [6].

Inconsistency that leads to incorrect execution causes a cache coherence problem. This problem appears if the data is updated by one processor without informing the others. Two basic protocols are used to eliminate this problem in memory system [9]:

a- Bus_Snooping Protocols

This protocol is used in a not scalable bus-based SMP system as broadcast medium where all the processor can observe all memory access by cache controller and then either invalidate or update the local cache content [10].

b- Directory_Based Protocols:

Either the directory is located centrally in the main memory of a multiprocessor system or may be distributed among caches. This directory is used as a tracker to a shared cache block between processors for maintaining coherence and consistent data which is recently updated. Therefore this protocol is preferred to use in a large scale shared memory multiprocessors [11].

MESI Cache Coherence Protocol:

MESI protocol relies on four states to maintain consistency and coherence between the caches in a shared memory system. These states represent a shortcut to Modified, Exclusive, Shared and Invalid where each block in the cache has one of them according to the request of the processor. The states are illustrated as follows [5,8]:

a- "Modified":

The cache contains a copy which differs from that in the main memory and there are no other copies. Modified cache lines need to be written back to memory when they get evicted or invalidated. Modified may also be called Dirty Exclusive.

b- "Exclusive":

The cached copy is only in cache, which is the same in the main memory.

Exclusive may also be called Clean Exclusive.

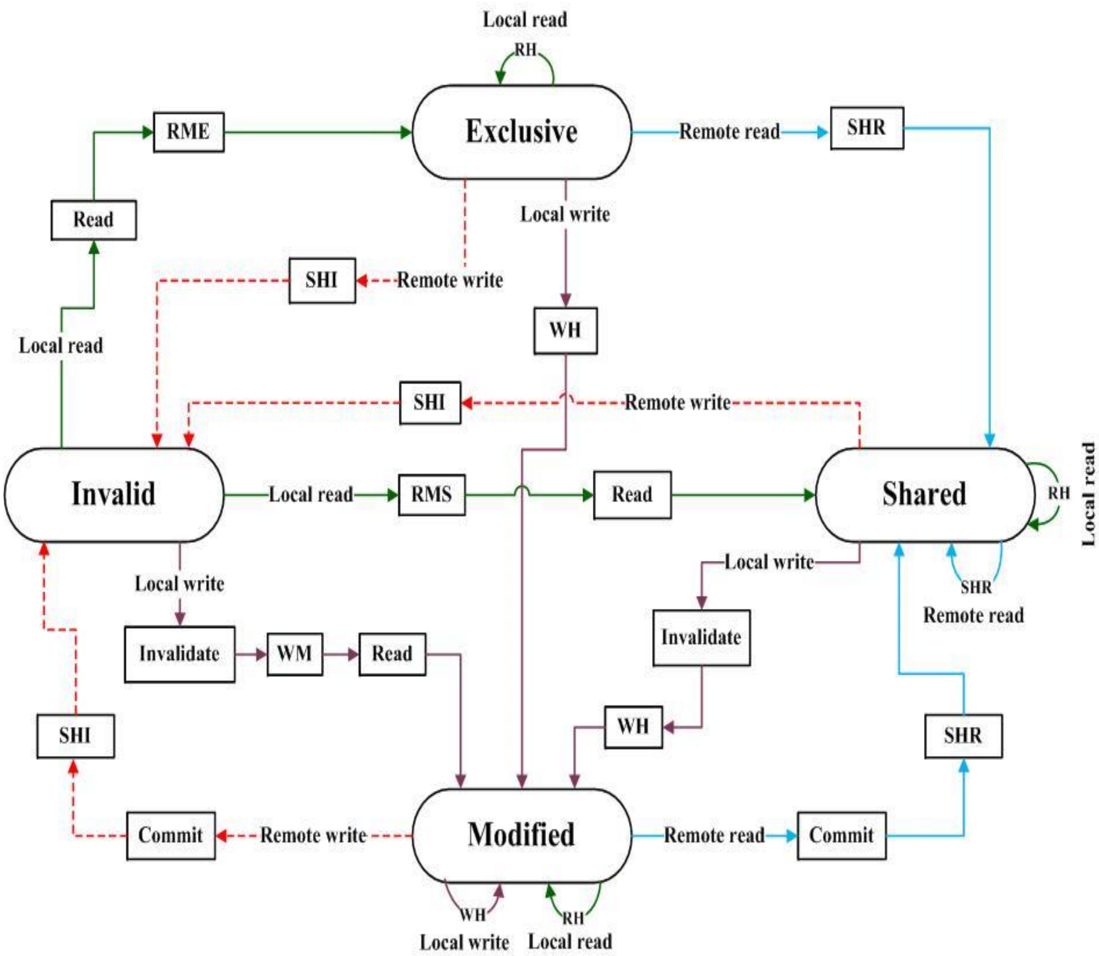
c- "Shared":

The cached copy is valid in each of the cache and main memory, and at least one of the shared memory caches to this copy.

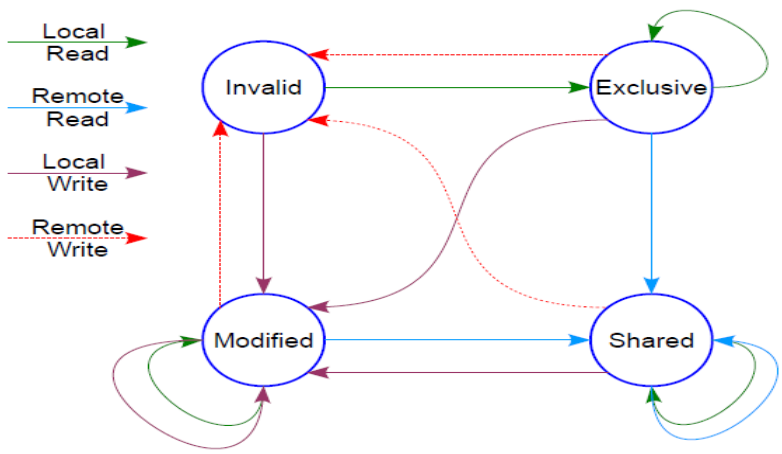
d- "Invalid":

This state indicates that the cache line does not have valid data.

Figure 2 demonstrates the transition between states of MESI:



(a)



(b)

Fig. 2: The State Transitions of MESI Cache Coherence Protocol (a) Detailed (b) abbreviation [1,5,7,8].

Methodology:

The steps of a proposed protocol using DEV C++ language are as follows:-

1. The Preprocessing steps of IES cache coherence protocol are represented in Fig. 3:

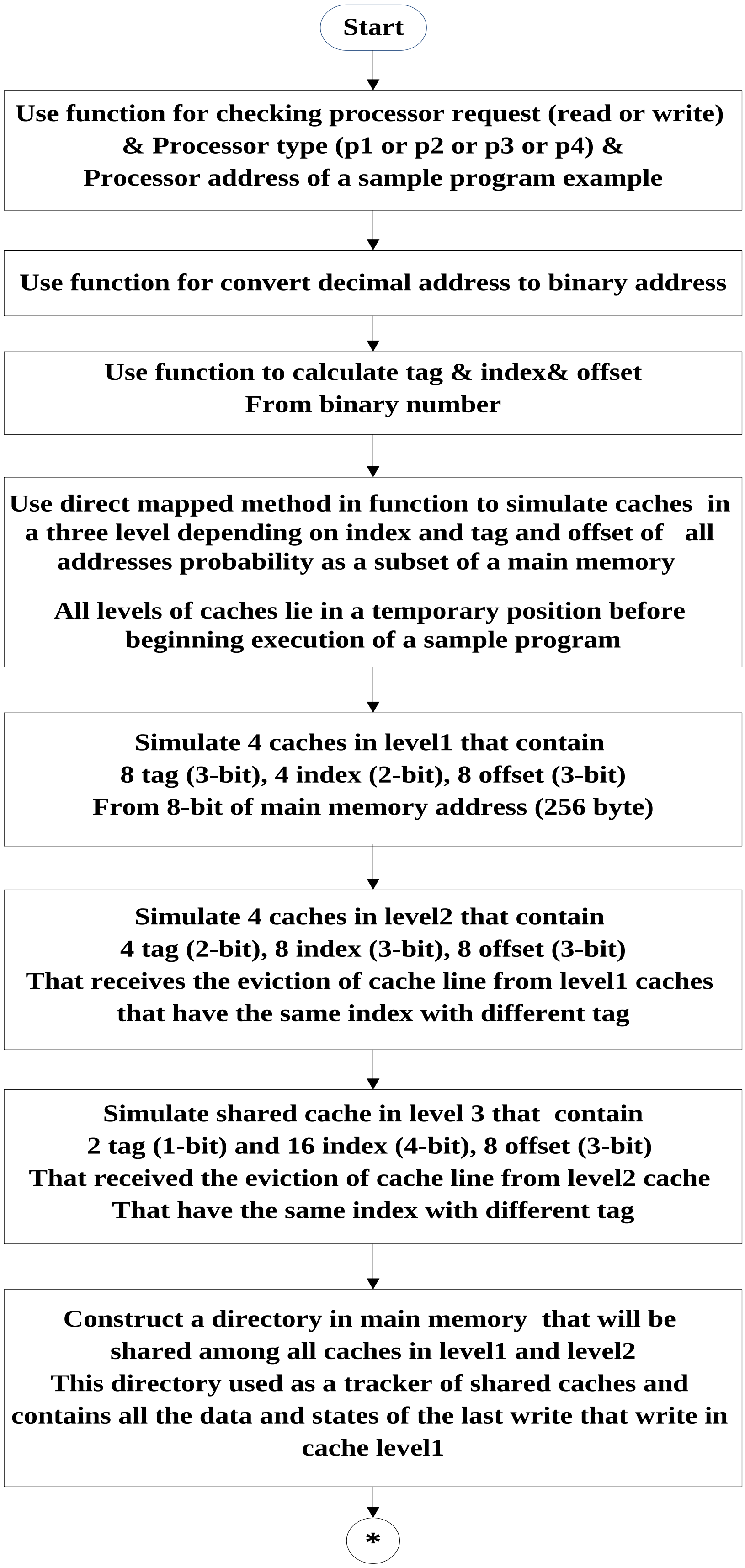


Fig.3: Flow Chart of Preprocessing Steps to Proposed Protocol

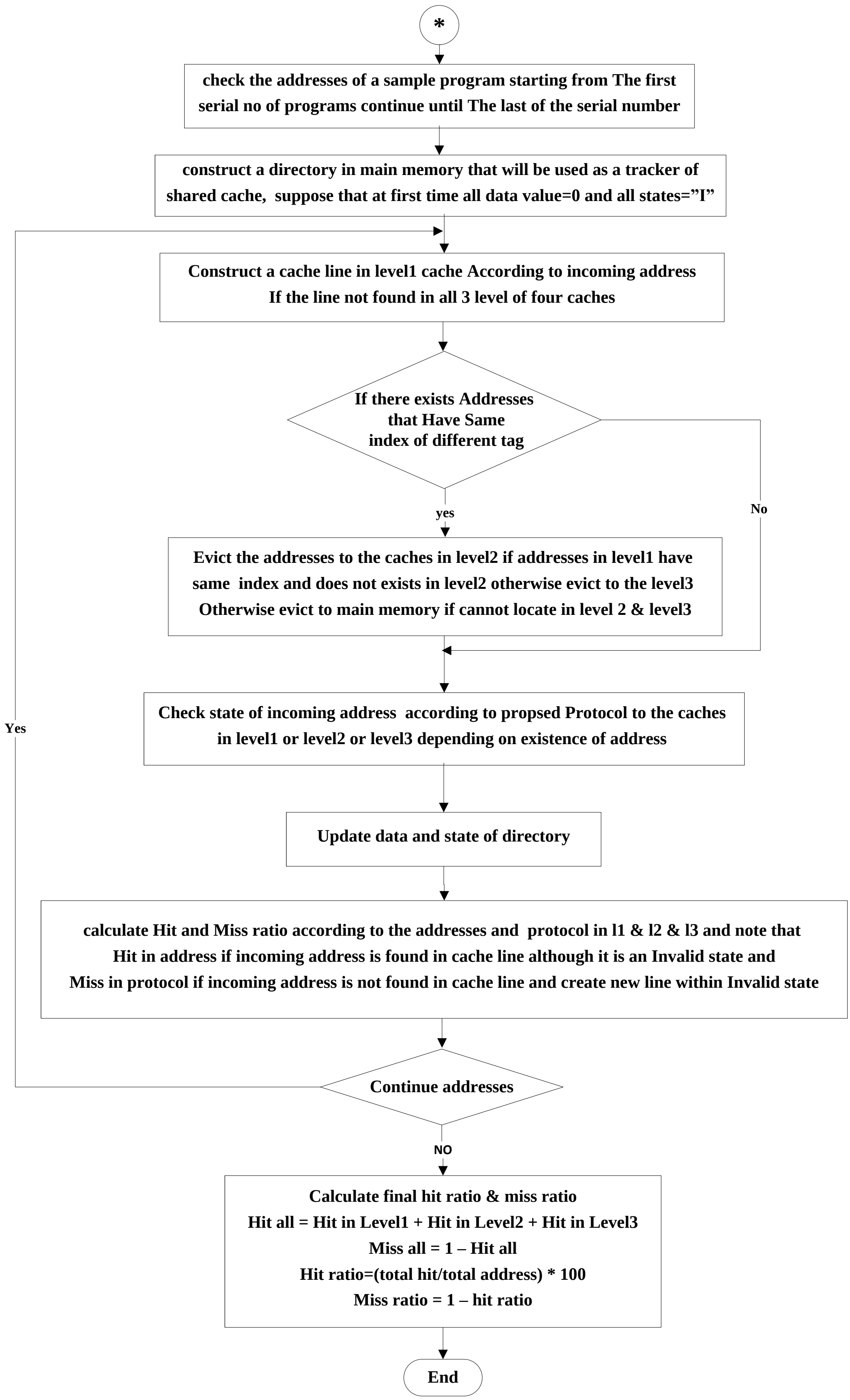


Fig. 3: Continue of Flow Chart of Preprocessing Steps

2 IES (Invalid, Exclusive read/write, Shared) Cache Coherence Protocol:

2.1 Cache Organization Using a Direct Mapped Method

At the beginning work in this research, the four caches in level 1, the four caches in level 2 and the shared cache in level 3 are simulated by using a direct mapped method taking advantage of spatial locality as follow:-

Suppose memory address = 256 [8-bit to represent the main memory address].

Level1 cache has 3-bit represent tag, 2-bit represent index, 3-bit represent offset

Level2 cache has 2-bit represent tag, 3-bit represent index, 3-bit represent offset

Level3 shared cache has 1-bit represent tag, 4-bit represent index, 3-bit represent offset

Main memory blocks are assigned to the lines of a cache mapped into one and only cache line as a result of a direct mapping method as in Table 1 where m represents number of lines in the cache and s bits specify one of 2^s blocks of main memory.

Table 2, Table 3 and Table 4 illustrate the assignment of a main memory blocks to the lines of caches in level1, level2 and level3 respectively as it is generally shown in Table1:

Table 1: Main Memory Blocks Assigned to Cache Line in General

Cache line	Main memory blocks assigned
0	$0, m, 2m, \ldots., 2^s - m$
1	$0, m+1, 2m+1, \ldots., 2^s - m + 1$
.	
m-1	$m-1, 2m-1, 3m-1, \ldots., 2^s - 1$

Table 2: Main Memory Blocks Assigned to Cache Line in Level1

Cache lines at level1	Main memory blocks assigned
0	0, 4, 8, 12, 16, 20, 24, 28
1	1, 5, 9, 13, 17, 21, 25, 29
2	2, 6, 10, 14, 18, 22, 26, 30
3	3, 7, 11, 15, 19, 23, 27, 31

Table 3: Main Memory Blocks Assigned to Cache Line in Level2

Cache lines at level2	Main memory blocks assigned
0	0, 8, 16, 24
1	1, 9, 17, 25
.	
.	
7	7, 15, 23, 31

Table 4: Main Memory Blocks Assigned to Cache Line in Level3

Cache lines at level3	Main memory blocks assigned
0	0, 16
1	1, 17
.	
.	
15	15, 31

2.2 IES Cache Coherence Protocol State Diagram

IES stands for the state of each cache line at any time. A cache line in each cache can be in one of the following states:-

Invalid: The block has been invalidated (possibly on the request of someone else)

Exclusive Read/Write: one processor has data and it is dirty which must respond to any write request in case of write. But in case of read it is clean that one processor has data and no need to inform others about further changes**.**

Shared: up-to-date the cached in more than one processors and memory

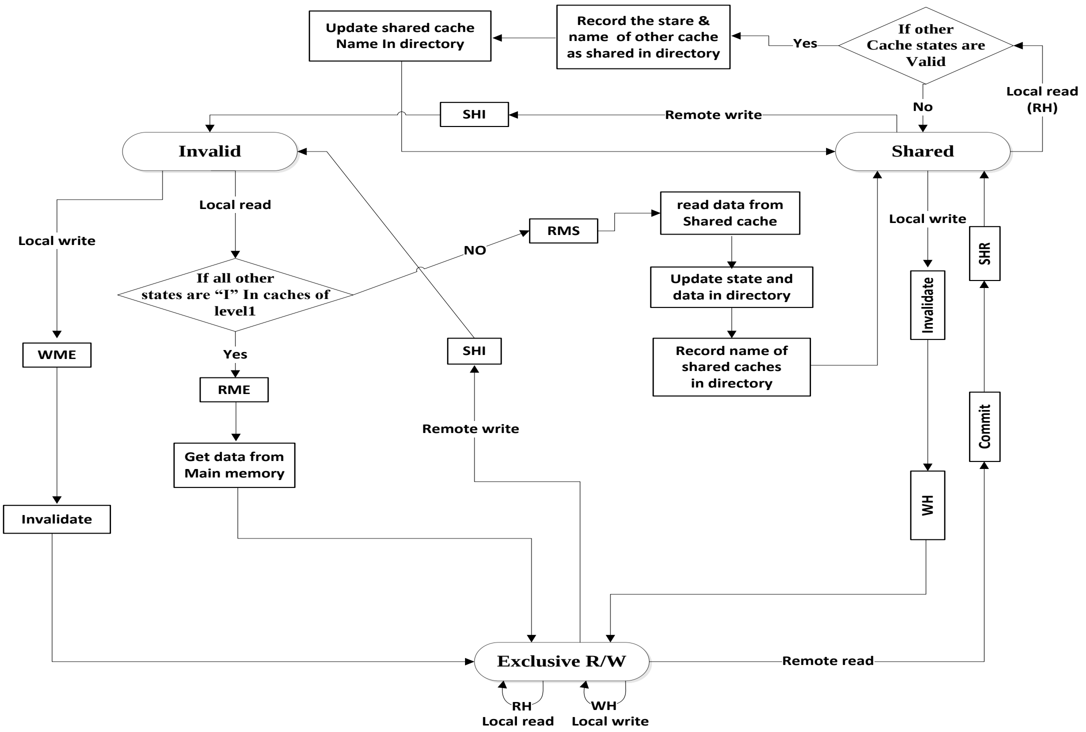


Fig. 4: IES Cache Coherence Protocol State Diagram

The abbreviate symbols of the buses are as follow:

<u>Bus transaction:</u>

<u>Events:</u>
RH = Read Hit, **RMS** = Read Miss Shared, **RME** = Read Miss Exclusive
WH = Write Hit, **WM** = Write Miss, **WME** = Write Miss Exclusive
SHR = Snoop Hit on Read, **SHI** = Snoop Hit on Invalidate.

Measuring Cache Performance:

Time CPU lapses in the implementation of the program as well as in waiting inside the memory, so CPU time is calculated as in the following equations [2]:

$$CPU\ time = (CPU\ execution\ clock\ cycles + Memory\ stall\ clock\ cycles) * clock\ cycle\ time \quad --(1)$$

$$CPU\ time = IC * \left(CPI_{Execution} + \frac{Mem\ Access}{Inst.} * Miss\ rate * Miss\ penality\right) * cycle\ time \quad --(2)$$

$$CPU\ time = IC * \left(\frac{ALUOPS}{Inst.} * CPI_{ALUOPS} + \frac{Mem\ Access}{Instruction} * AMAT\right) * cycle\ time --(3)$$

$$\mathrm{AMAT} = L1\ Hit\ time * L1\ Hit\ rate + L1\ Miss\ penality * L1\ miss\ rate \quad --(4)$$

$$L1\ Miss\ penality = Access\ time\ of\ L2 = L2\ Hit\ time * L2\ Hit\ rate + L2\ Miss\ penality * L2\ Miss\ rate --(5)$$

$$L2\ Miss\ penality = Access\ time\ of\ L3 = L3\ Hit\ time * L3\ Hit\ rate + L3\ Miss\ penality * L3\ Miss\ rate --(6)$$

$$L3\ Miss\ penality = Access\ time\ of\ Main\ Memory --(7)$$

$$\begin{aligned}AMAT &= L1\ hit\ time * L1\ Hit\ rate \\ &+ (L2\ Hit\ time * L2\ Hit\ rate \\ &+ (L3\ Hit\ time * L3\ Hit\ rate \\ &+ Access\ time\ of\ Main\ Memory \\ &* L3\ Miss\ rate) * L2\ Miss\ rate) \\ &* L1\ Miss\ rate --(8)\end{aligned}$$

Where

$$IC = Instruction\ Counter$$

$$CPI = Clock\ Cycle\ Per\ Instruction$$

$$AMAT = Average\ Memory\ Access\ Time$$

The Experimental Result Using DEV C++ Language

1. Binary Representation

Binary representation is one of a necessary preprocessing steps used to gain tag and index and offset of each decimal input address so as to facilitate the work of a mapping algorithm as in Table 5

Table 5: Binary Representation of Input Addresses Using a Proposed Protocol

address representation			cache level1			cache level2			cache level3		
Seq	binary no	no	tag	index	offset	Tag	index	offset	tag	index	offset
1	01000110	70	2	0	6	1	0	6	0	8	6
2	01000110	70	2	0	6	1	0	6	0	8	6
3	01000110	70	2	0	6	1	0	6	0	8	6
4	00100011	35	1	0	3	0	4	3	0	4	3
5	11100110	230	7	0	6	3	4	6	1	12	6
6	10100100	164	5	0	4	2	4	4	1	4	4
7	00101001	41	1	1	1	0	5	1	0	5	1
8	01000110	70	2	0	6	1	0	6	0	8	6
9	01000110	70	2	0	6	1	0	6	0	8	6
10	00110100	52	1	2	4	0	6	4	0	6	4
11	11100110	230	7	0	6	3	4	6	1	12	6
12	01100100	100	3	0	4	1	4	4	0	12	4
13	11000001	193	6	0	1	3	0	1	1	8	1
14	11000001	193	6	0	1	3	0	1	1	8	1
15	00101001	41	1	1	1	0	5	1	0	5	1
16	11000001	193	6	0	1	3	0	1	1	8	1
17	01100100	100	3	0	4	1	4	4	0	12	4
18	11000001	193	6	0	1	3	0	1	1	8	1

2. Cache Simulation

The caches are simuled by using a direct mapped method in all level of caches according to an input address of a sample program (Table 6).

Table 6: Cache Simulation Using Direct Mapping in All Level Using IES Protocol

Index	Simulation of Cache 1 in Level1							
0	96	97	98	99	100	101	102	103
1	40	41	42	43	44	45	46	47
2	48	49	50	51	52	53	54	55

Index	Simulation of Cache 2 in Level1							
0	192	193	194	195	196	197	198	199

Index	Simulation of Cache 3 in Level1							
0	96	97	98	99	100	101	102	103
1	40	41	42	43	44	45	46	47

Index	Simulation of Cache 4 in Level1							
0	192	193	194	195	196	197	198	199

Index	Simulation of Cache 1 in Level2							
0	64	65	66	67	68	69	70	71
4	32	33	34	35	36	37	38	39

Index	Simulation of Cache 2 in Level2							
0	64	65	66	67	68	69	70	71
4	160	161	162	163	164	165	166	167

Index	Simulation of Cache 3 in Level2							
0	64	65	66	67	68	69	70	71

Index	Simulation of Cache 4 in Level2							
0	64	65	66	67	68	69	70	71
4	224	225	226	227	228	229	230	231
Index	Simulation of a shared cache in Level3							
8	192	193	194	195	196	197	198	199
12	224	225	226	227	228	229	230	231

3. IES Cache Coherence Protocol Result

Table (7) demonstrates a result in applying IES protocol in Fig. (4) on a sample program. Initially all states of an input addresses are Invalid, so when the processor P1 in step1 read address

70, the state is translated from "I" to "E" because the address is not found in the all levels of caches and as a result, the cache line that contain address gets by a read miss from main memory. All the next steps of the program are applied using the protocol in the same way.

Table 7: The Results of a Proposed Protocol on a Sample Program

Seq	Core name	Core job	data	cache line			job	sharer of cores
				address	state	value		
1	P1	reads		70	E	0	R. M.	
2	P2	writes	12	70	E	12	W.M.	
3	P3	writes	29	70	E	29	W.M.	
4	P1	writes	58	35	E	58	W.M.	
5	P4	writes	80	230	E	80	W.M.	
6	P2	reads		164	E	0	R.M.	
7	P3	reads		41	E	0	R.M.	
8	P4	writes	30	70	E	30	W.M.	
9	P1	reads		70	S	30	R.H.	P1 in L2+P4 in L1
10	P1	writes	11	52	E	11	W.M.	
11	P1	reads		230	S	80	R.M.	P1 in L1+P4 in L2
12	P3	writes	92	100	E	92	W.M.	
13	P2	writes	73	193	E	73	W.M.	
14	P4	reads		193	S	73	R.M.	P2&P4 in L1
15	P1	writes	69	41	E	69	W.M.	
16	P1	Reads		193	S	73	R.M.	P1&P2&P4 in L1
17	P1	Reads		100	S	12	R.M.	P1&P3 in L1
18	P3	Reads		193	S	73	R.H.	P1&P3 in L3 + P2&P4 in L1

Note: The words of the abbreviation symbols in table 7 are as follows:
R. M. = Read Miss,
W. M. = Write Miss,
R. H. = Read Hit
L1 = Level 1 of caches**, L2 =**Level2 of caches**, L3=**Level3 of caches

4. Cache Performance Result

Cache performance can be measured by counting a program execution cycles that include cache Hit time and a memory stall cycles which result from cache misses. Depending on the clock speed of the central processor, it takes:
2 to 5 ns to access data in L1 cache, 10 to 20 ns to access data in L2 cache,
30 ns to access data in L3 cache, 50 to 100 ns to access data in Main Memory.
Hit and miss ratio result from table6 and table7 are as follow:

Hit ratio in L1 = (no. of hit in level1/ total no. of address)*100 = (1/18)*100= 5.56
Miss ratio in L1 =100-Hit ratio =100-5.56= 94.44

Hit and Miss ratio in level2 is the same as in level. But miss ratio in level3=100%

The Comparison between MESI and Proposed Protocol:

- The difference between MESI and IES protocol is that IES protocol merges modified and exclusive states to get one state named exclusive. In MESI protocol the state of incoming address in current processor becomes exclusive when the request of that processor is read and states of that address of all processors are invalid. While the request of a processor that enter the Exclusive state using IES protocol is either read or write.
- In the absence of a place to put the entrance address in the cache memories, in the proposed protocol it is taken to the reserve from the beginning put the line of that address to main memory and therefore data retention without loss. As a result, it is not needed to

use write back in the case of writing done by the rest cores.

- In MESI cache coherence protocol there are several times that a write back is used. The number of a write back (WB) to the main memory that results from a remote write of other processor of MESI protocol are:

- WB from the processor P2 in step 2 as a result of remote write of P3 in step3.
- WB from the processor P3 in step 3 as a result of remote write of P4 in step8.

By applying equation 2

$$\boldsymbol{CPU\ time} = \boldsymbol{IC} * \left(\boldsymbol{CPI_{Execution}} + \frac{\boldsymbol{Mem\ Access}}{\boldsymbol{Inst.}} * \boldsymbol{Miss\ rate} * \boldsymbol{Miss\ penality}\right) * \boldsymbol{cycle\ time}$$

The memory accesses are decreased as a result of reducing a write back to the main memory in using IES protocol. So, CPU performance is increased by reducing in CPU time.

A few differences between the two protocols show there because of the implementation of the program just a few steps. But the benefit of using this protocol appears when the program is implemented numerous steps.

Conclusion and Future Works:

Cache memory is a main component of memory hierarchy which plays an important role in the overall performance of the system and in the design of multicores. Multicores with shared memory architecture are used to satisfy increasing performance demands, which in turns are limited by cache coherence problem. Thus this survey focuses on the subject of the use of a protocol to solve the problem of data match and improve this protocol. In future work we try to increase size and the number of caches in level1 and level2 and to change in states and using another mapping algorithm that have more associativity.

References:

[1] Thomas, R. & Gudula, R. 2013. Parallel Programming For Multicore and Cluster Systems, Published by Springer-Verlag Berlin and Heidelberg GmbH & Co. KG, Berlin.

[2] David, A. P. & John, L. H. 2005. Computer Organization and Design the Hardware / software interface, Elsevier Inc.

[3] David, A. P. & John, L. H. 2012. Computer Architecture A quantitative approach, Morgan Kaufmann is an imprint of Elsevier.

[4] Azilah, S. and Najihah, B. R. F. 2014. Cache Coherence Protocols in Multi-Processor, International conference on Computer Science and Information Systems (ICSIS'2014) Oct 17-18, Dubai (UAE)

[5] William, S. 2010. Computer organization and architecture designing for performance, Printed in the United States of America by Pearson Education, Inc., Upper Saddle River, New Jersey, 07458.

[6] Bryon, M. 2013. Real World Multicore Embedded Systems, Elsevier Inc., United States of America.

[7] David C.; Jaswinder, S. P. and Anoop, G. 1997. Parallel Computer Architecture A Hardware/Software Approach, scalability, programmability, Morgan Kaufmann Publishers.

[8] Anoop, T. 2014. Performance Comparison of Cache Coherence Protocol on Multi-Core Architecture, Department of Computer Science and Engineering National Institute of Technology Rourkela Rourkela, Odisha, 769008, India.

[9] Al-Hothali S. and Soomro S. et.al. 2010. Snoopy and Directory Based Cache Coherence Protocols: A Critical Analysis, Journal of Information & Communication Technology (Spring 2010) 4(1): 01-10.

[10] Alberto, R. and Alexandra, J. 2015. A Dual-Consistency Cache Coherence Protocol, IEEE 29th International Parallel and Distributed Processing Symposium IPDPS, pp: 1119-1128, USA.

[11] Zaghloul, S. S., et.al. 2014. Index-Based Cache Coherence Protocol, David Publishing Company (DPC) Journal of communication and Computer 11:479-483.

تصميم ومحاكاة بروتوكول الترابط في الذاكرةالمخبئية باستخدام حالات (غير صالح، حصري في القراءة والكتابة، مشترك)

لمى فايق جليل*   مها عبد الكريم حمود الراوي   عبير ضياء النقشبندي*****

*استاذ مساعد، قسم علوم الحاسوب، الجامعة التكنولوجية، بغداد، بغداد، العراق.
** استاذ مساعد، قسم علوم الحاسوب، الجامعة التكنولوجية، بغداد، بغداد.
***طالبة مرحلة البحث لنيل شهادة الدكتوراه، رئيس مبرمجين أقدم، دائرة توزيع الطاقة، وزارة الكهرباء، بغداد، العراق.

الخلاصة:

في الانظمة الحديثة للمعالجات المتعددة تم استخدام الذواكر المخبئية بدلا من الذاكرة الرئيسة في حالة الوصول الى البيانات وذلك لتحسين كفاءة المعالج من خلال تقليل تأخيرالوصول الى البيانات. الا ان الصعوبة في هذه الانظمة والتي يتم فيها تنصيب مختلف الذواكر المخبئية في عدة معالجات التي تشترك بذاكرة واحدة تكمن في الحفاظ على التطابق بين ذاكرات الذواكر المخبئية ذات المعالجات المتعددة. ولهذا السبب من الضروري استخدام بروتوكول الترابط مابين الذواكر المخبئة. ومن انواع البروتوكولات المشهورة لحل المشكلة التي تظهر عند الترابط مابين الذواكر المخبئية هي MOESI, MOSI, MESI, MSI.
لقد اقترحنا في هذا البحث دمج حالتين من حالات بروتوكول ترابط الذواكر المخبئية ميسي والتي هي الحصرية والمعدلة والتي تستجيب لطلبات القراءة والكتابة في نفس الوقت والتي تعود حصرا لهذه الطلبات. وأيضا تم ازالة الرجوع الى الذاكرة الرئيسية باستخدام البروتوكول المقترح من احدى المعالجات التي تكون في حالة "معدلة" والتي تصبح في حالة "غير صالح" عند الكتابة من معالج اخر له نفس العنوان لانه في كل الاحوال يتم الاعتماد على القيمة الاخيرة التي يتم كتابتها واذا كان الرجوع الى الذاكرة يستخدم للحفاظ على البيانات من الضياع فانه باستخدام الخطوات المسبقة للبروتوكول المقترح يتم الاحتفاظ وخزن البيانات في الذاكرة الرئيسية عند خروجها من الذاكرة المخبئية. كل هذا يؤدي إلى زيادة كفاءة المعالج عن طريق الحد من الوصول إلى الذاكرة.

الكلمات المفتاحية: مشكلة الترابط في الذاكرة المخبئيـة، بروتوكول الاسـتطلاع، البروتوكول القائـم على الدليل، MESI، محاكاة الذاكرة المخبئية، DEV C++، المعالجات المتعددة، الذاكرة المشتركة.