Analysis of System Reliability for Cache Coherence Scheme in Multi-Processor

Sizhao Li[1], Shan Lin[1], Deming Chen[3], W. Eric Wong[4], and Donghui Guo[1, 2], Senior *Member, IEEE*
1. Dept. of Electronic Engineering, Xiamen University, Fujian 361005, China
2. IC Design & IT Research Center of Fujian Province, Xiamen University, Fujian 361005, China
3. Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, IL 61801, USA
4. Department of Computer Science, University of Texas at Dallas, TX 75083, USA
sizhao.li@gmail.com, dchen@illinois.edu, ewong@utdallas.edu, dhguo@xmu.edu.cn

Abstract—**In this paper, a cache coherence scheme in multi-processor is introduced. There is a specific model in each kind of software; cache coherence can be solved in AHB bus by these models. First, we use dynamic address mapping policy to realize data cache. Second, according to the randomness of application environment that set up shared cache adaptive configuration and management mechanism in the finite state machine timing sequence model of each kind of software, to ensure the system reliability. In order to support multi-tasking and multi-user operator system – Linux, the multi-processor must use shared memory technology, so this paper also introduced the memory management unit, and base on these, it focuses on how multi-processor and the AHB bus cooperate to ensure cache coherence of the whole system. We can use software execution model and hardware design to achieve instruction or data coherence between each cache and main memory.**

Keywords- cache coherence, memory management, system reliability, multi-processors, system failure.

I. Introduction

Research on multi-processor has been an important part of research in the field of microprocessors. Over the years, a variety of hardware architectures have been proposed to solve mutual cooperation and communication between the multi-processor, so as to improve processing speed and performance. In order to ensure the system reliability, it is necessary to ensure that the data is correct in each processor [1], cache provide instruction and data to the processor, therefore, cache coherence is the most important [2].

Currently, a variety of multi-processor cannot work without the cache on-chip [3]. In order to improve the performance of system, we can use multi-level cache for design on-chip commonly. Each processor unit usually has its own private L1 cache or L2 cache, and they can also share storage resources other on-chip through interconnection [4]. Multi-processor is often running different programs simultaneously; it needs to consider how to configure storage resource on-chip and sharing management issues, but different architecture adopts the management model may not be the same, and the same time the memory management has great complexity [5]. Therefore, optimal configuration of shared cache requires instruction and data reuse based on different software or environment to decide. If there is no good memory coherence management protocol, error detection and repair mechanism, the system might cause memory usage conflict or data transfer error, and it might causes system instability or collapse [6]. These errors will affect the whole system reliability through Mean Time to Failures (MTTF), Mean Residual Life (MRL) and other performance index [7].

The rest of this paper is organized as follows: Section II describes the related work. Section III. This section describes the mathematical model of cache coherence, and at the same time, it is important of coherence from the system reliability point of view. Section IV. This section describes the evaluation results and presents the discussion. Finally, conclusion will be included.

II. System Reliability and Cache Coherence Protocol

System reliability indicates that the system is a capability of complete specific function under the condition and the required time [8]. Factors affecting the system reliability are two aspects: one is self-reliability of system device; another is effect of external condition.

A. Failure model

First introduced the following four key concepts, T is the failure time, $f(t)$ is the probability density function of the failure time, and the distribution function is

$$F(t) = Pr(T \le t) = \int_0^t f(u)du\,, t > 0$$

1) The Reliability Function can be defined as

$$R(t) = 1 - F(t) = 1 - \int_0^t f(u)du = \int_t^\infty f(u)du$$

Where $R(t)$ is the no failure probability of device unit in the time interval $(0, t]$.

2) The Failure Rate Function $z(t)$ is the failure probability of device unit in the time interval $(t, t+\Delta t]$

$$Pr(t < T \le t + \Delta t | T > t) = \frac{Pr(t < T \le t + \Delta t)}{Pr(T > t)} = \frac{F(t+\Delta t) - F(t)}{R(t)}$$

Dividing both sides by $\Delta t \to 0$, and taking the limit, so

$$z(t) = \lim_{\Delta t \to 0} \frac{\Pr(t < T < t + \Delta t | T > t)}{\Delta t} = \lim_{\Delta t \to 0} \frac{F(t+\Delta t) - F(t)}{\Delta t} \cdot \frac{1}{R(t)} = \frac{f(t)}{R(t)}$$

3) The Mean Time to Failure (MTTF) is

$$MTTF = E(T) = \int_0^\infty tf(t)dt = \int_0^\infty R(t)dt$$

4) The failure time of device unit is T, it started working at $t=0$, and has been working to the time t. And then this reliability of device unit will be working normally in time x which is

$$R(x|t) = Pr(T > x + t|T > t) = \frac{Pr(T > x + t)}{Pr(T > t)} = \frac{R(x + t)}{R(t)}$$

$R(x|t)$ is called the conditional reliability function of device unit in time x. And in time x, Mean Residual Life (MRL) of device unit is

$$MRL(t) = \mu(t) = \int_0^\infty R(x|t)dx = \frac{1}{R(t)}\int_t^\infty R(x)dx$$

System failure is random, therefore, to represent cache failure with the distribution function of failure. According to the different of principle, failure can occur in discrete distribution or continuous distribution. However, the data error is a continuous distribution in the cache [9], so it mainly discusses the continuous distribution. As shown in Table I, there are five continuous distributions most commonly.

TABLE I. SYSTEM FAILURE DISTRIBUTION

Distribution Form	Failure Density Function $f(t)$	Reliability Function $R(t)$
Exponential Distribution	$\lambda e^{-\lambda t}$	$e^{-\lambda t}$
Weibull Distribution	$\alpha\lambda^{\alpha}t^{\alpha-1}e^{-(\lambda t)\alpha}$	$e^{-(\lambda t)\alpha}$
Normal Distribution	$\frac{1}{\sigma\sqrt{2\pi}}e^{-(t-\theta)^2/2\sigma^2}$	$\frac{1}{\sigma\sqrt{2\pi}}\int_t^\infty e^{-(t-\theta)^2/2\sigma^2}dr$
Log-Normal Distribution	$\frac{1}{\sigma\sqrt{2\pi}}e^{-(\ln t-\theta)^2/2\sigma^2}$	$\frac{1}{\sigma\sqrt{2\pi}}\int_t^\infty \frac{1}{\tau}e^{-\frac{(\ln t-\theta)^2}{2\sigma^2}}dr$
Anti-Gaussian Distribution	$\sqrt{\frac{\lambda}{2\pi t^3}}e^{-(\lambda/2\mu^2)[(t-\mu^2)/t]}$	$\frac{1}{\sqrt{2\pi}}\left(e^{-\left(\frac{\sqrt{\lambda}}{\mu}\sqrt{t}-\frac{\sqrt{\lambda}}{\sqrt{t}}\right)^2/2}+e^{2\lambda/\mu}\cdot e^{-\left(-\frac{\sqrt{\lambda}}{\mu}\sqrt{t}-\frac{\sqrt{\lambda}}{\sqrt{t}}\right)^2/2}\right)$

B. *Cache coherence protocol*

In computer, cache coherence refers to the consistency of data stored in local caches of a shared resource. Cache coherence refers to the same information and the copy subsequent memory is consistent. If a word has been modified in the cache, it also must be modified immediately or lastly on a higher level memory. In multi-processor, between adjacent levels or between the same levels can appear inconsistent data. Data inconsistency will lead to program error eventually; the system will generate an error or even collapse. Therefore, for any hardware architecture now, to solve the cache coherence problem is the first problem to solved, and in this way the system can be guaranteed reliability.

There are many protocols to solve the cache coherence. But so far, according to the principle of reliability, no one can still solve the coherence problem completely. Therefore, analyze the system reliability depending on the different protocol model. The following introduce each protocol model.

1) Write-Once

In cache coherence protocol, ***Write-Once*** is the first write-invalidate protocol defined. It is Write-Back and Write-Through comprehensive. When using this mechanism, a cache line write back for the first time, and the same time data is written back to cache and main memory using the Write-Through policy, after the write operation using Write-Back, only written back to cache and not written back to main memory.

There are four states in each CPU to identify the current state of cache line. ***I***nvalid indicates that the current cache line doesn't contain valid data, it is invalid cache line. ***V***alid indicates that the data of current cache is the latest, and there is the copy of cache line in main memory. ***R***eserved also indicates that the data of current cache is the latest, and the main memory has a copy data of cache line. ***D***irty represent data is the latest in the cache line, and only this cache line has the latest copy of data, doesn't synchronize with the main memory. Figure 1 is the Write-Once transition diagram.

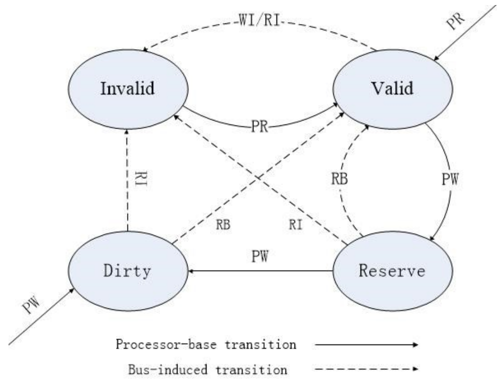

Fig.1. *Write-Once* transition diagram. (*PR-the local processor read copy of cache; PW-the local processor write copy of cache; RB-read a valid copy from another cache; WI-when write hit broadcast an invalid command on the bus; RI-when write miss broadcast an invalid command on the bus*)

2) MESI

MESI (aka Illinois) is the cache coherence protocol which is widely used to support the write back strategy. As the name suggests, it is divided into four states. ***M***odified, the cache line exists only in the current cache, and is dirty (modified), the value has been modified compared to the main memory. ***E***xclusive, the cache line exists only in current cache, but it is clean (unmodified), and the value is consistent compared to the main memory. ***S***hared, this state descript that cache line may be stored in other caches, and is also clean; the value is consistent compared to the main memory. ***I***nvalid, this state descript that cache line is invalid.

3) MOESI

In computer, *MOESI* is a full cache coherence protocol that encompasses all of the possible states commonly used in other protocols. In addition to the four common MESI protocol states, there is a fifth ***O***wned state representing data that is both modified and shared.

Owned, this cache line contains the latest data copy in current processor, and there must be a copy of the cache line in the other CPU, states of cache line is Shared in other CPUs. As shown in Figure 2, it is the cache coherence model based on MOESI protocol.

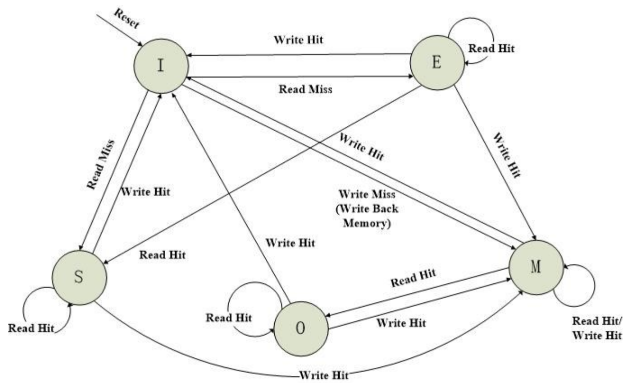

Fig.2. *MOESI* transition diagram.

III. RELIABILITY MODEL OF CACHE

In *Write-Once* protocol, each state will affect the whole system; any mistake of state can lead to cache inconsistent. These shows that are a series system [10], it is a basic mathematical model that the problem of system reliability can be triggered by cache coherence as shown in Figure 3. Mathematical model for the series system is

$$R_\tau(t) = R_1 \cdot R_2 \cdot R_3 \cdots R_n = \prod_{i=1}^{n} R_i$$

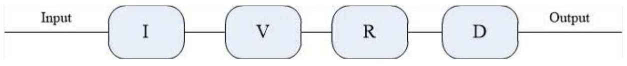

Fig.3. Write-Once series flow chart

Where n is the system state number of composition, R_i is the reliability of the i system state, R_s is the system reliability. When the failure distribution of each state is exponential distribution in *Write-Once* protocol, that is $R_i(t) = e^{-\lambda_i t}$, and then system reliability is

$$R_\tau(t) = \prod_{i=1}^{4} e^{-\lambda_i t} = e^{-\lambda_\tau t}$$

The Failure Rate is the probability of system which loss functions in the stipulated conditions and within specified time. λ_I, λ_V, λ_R and λ_D represent the parameters of exponential distribution I, V, R and D. Each state has a certain failure rate, the failure rate of system is

$$z(t) = \lambda_\tau = \lambda_I + \lambda_V + \lambda_R + \lambda_D = \sum_{i=1}^{4} \lambda_i$$

And the MTTF is

$$MTTF_\tau = \frac{1}{\lambda_\tau} = \frac{1}{\sum_{i=1}^{4} \lambda_i}$$

In *MESI* protocol, Exclusive is a special case of Shared, therefore, this shows that is a series-parallel system [10]. Mathematical model for the series-parallel system is shown in Figure 4. In (a), the state ***M*** is recorded as R_M and ***I*** is recorded as R_I, state ***E*** and state ***S*** parallel into R_p. Finally, R_1, R_2 and R_3 cascade into R_s as shown in Figure 4 (b). λ_M, λ_E, λ_S and λ_I represent the parameters of exponential distribution M, E, S and I. This model can be used to express the following formula,

$$R_p = R_E + R_S - R_E R_S$$

$$R_\tau(t) = R_M \cdot R_p \cdot R_I = R_M \cdot R_I \cdot (R_E + R_S - R_E R_S)$$

$$\lambda_\tau = \lambda_M + \lambda_E + \lambda_S + \lambda_I = \sum_{i=1}^{4} \lambda_i$$

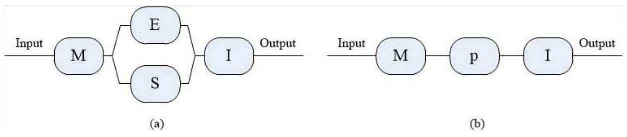

Fig.4. *MESI* series-parallel flow chart

The probability density function is

$$f(t) = \lambda_\tau e^{-\lambda_\tau t}$$

So the failure rate function of system is

$$z(t) = \frac{f(t)}{R_\tau(t)} = \frac{\lambda_\tau e^{-(\lambda_E+\lambda_S)t}}{e^{-\lambda_E t} + e^{-\lambda_S t} + e^{-(\lambda_E+\lambda_S)t}}$$

And the MTTF is

$$MTTF_\tau = \int_0^\infty R_\tau(t)dt$$

In *MOESI* protocol, it redefines the state Shared, the state ***M*** is recorded as R_M and ***I*** is recorded as R_I, state ***O***, state ***E*** and state ***S*** parallel into R_p, and process similar to

Figure 4. We can also use another way in Figure 5, it is called failure tree. When system is series, all the nodes are connected to the *OR* gate. And when system is parallel, all the nodes are connected to the *AND* gate.

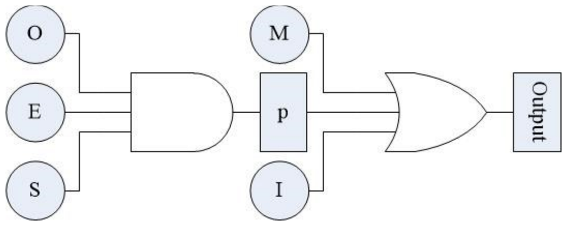

Fig.5. MOESI failure tree

λ_M, λ_O, λ_E, λ_S and λ_I represent the parameters of exponential distribution M, O, E, S and I. This model can be used to express the following formula,

$$R_p = R_E + R_O + R_S - R_E R_O - R_E R_S - R_O R_S + R_E R_O R_S$$

$$\lambda_\tau = \lambda_M + \lambda_E + \lambda_O + \lambda_S + \lambda_I = \sum_{i=1}^{5} \lambda_i$$

So the failure rate function is

$$z(t) = \frac{f(t)}{R_\tau(t)}$$
$$= \frac{\lambda_\tau e^{-(\lambda_E+\lambda_O+\lambda_S)t}}{e^{-\lambda_E t} + e^{-\lambda_O t} + e^{-\lambda_S t} - e^{-(\lambda_E+\lambda_O)t} - e^{-(\lambda_E+\lambda_S)t} - e^{-(\lambda_O+\lambda_S)t} + e^{-(\lambda_E+\lambda_O+\lambda_S)t}}$$

These models can analyze the reliability of different protocols, and can optimize hardware-software co-design.

IV. Experimental Result and Discussion

Reliability evaluations of cache coherence protocols help improve system stability. For quantitative analysis of mathematical models, is only the evaluation of protocol, and not a specific hardware platform. We can make some simulation calculation to assess the advantages and disadvantages of each protocol. In this paper, digital circuit design for the reliability analysis of cache coherence is as a target. In Table II, there is the reliability of the three models.

The failure rate of the control logic block is usually a fixed value: $\lambda = 0.31 \times 10^{-6}$ [10]. Assuming the failure rate of states are the same in these three cache coherence protocols, and the failure distribution meet exponential distribution. Based on reliability model of cache coherence, the reliability index can be calculated.

When t is a fixed value, $R_{(Write\text{-}Once)} < R_{(MESI)} < R_{(MOESI)}$ can be got as shown Figure 6. The reliability of *MOESI* is the highest, but the reliability of *MESI* is the most significant increase. In Figure 7, $z_{(Write\text{-}Once)}$ is fixed value, $z_{(MESI)} < z_{(Write\text{-}Once)}$, and the change trend of $z_{(MOESI)}$ is relatively large. And then, the mean time to failure of the three protocols is also increasing. Through quantitative analysis, the reliability of *MOESI* is the highest that can be clearly observed. Therefore, now the hardware architecture generally still uses the *MESI* protocol.

TABLE II. The Reliability Index Of Three Protocols

Protocol	$f(t)$	$R(t)$	$z(t)$	$MTTF$
Write-Once	$4\lambda e^{-4\lambda t}$	$e^{-4\lambda t}$	4λ	$1/4\lambda$
MESI	$4\lambda e^{-4\lambda t}$	$2e^{-3\lambda t} - e^{-4\lambda t}$	$\frac{4\lambda}{2e^{\lambda t}-1}$	$5/12\lambda$
MOESI	$5\lambda e^{-5\lambda t}$	$3e^{-3\lambda t} - 3e^{-4\lambda t} + e^{-5\lambda t}$	$\frac{5\lambda}{3e^{2\lambda t}-3e^{\lambda t}+1}$	$9/20\lambda$

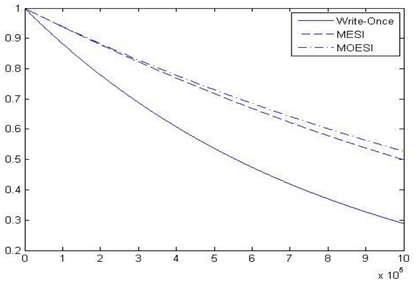

Fig.6. The Reliability Function curve diagram

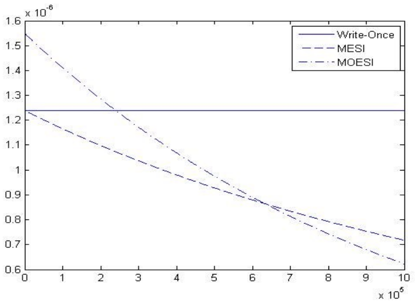

Fig.7. The Failure Rate Function curve diagram

V. Conclusion

To establish the reliability model is helpful to the analysis of cache coherence, and the optimal model can be selected by mathematical simulation. In the future, according to the hardware architecture, instruction set, specific complier system and the software execution model, a new cache coherence protocol can be proposed by the reliability model in this paper. And at the same time, any model cannot be completely reliable; failure will be able to be repaired by Markov process. And will be design and verification in hardware.

Acknowledgment

This research was supported by Research Fund for the Doctoral Program of Higher Education of China under Grants No.20090121110019 and National Natural Science

Foundation of China (General Program) under Grants No. 61274133

REFERENCES

[1] Hemayet Hossain, Sandhya Dwarkadas, and Michael C. Huang, "Coherence Protocol Optimization for both Private and Shared Data," Parallel Architectures and Compilation Techniques (PACT), 2011 international Conference, Galveston, pp. 45-55.

[2] Alberto Ros, Manue E. Acacio, and José M. García, "A Direct Coherence Protocol for Many-Core Chip Multiprocessors," IEEE Transactions on Parallel and Distributed Systems, vol. 21, no. 12, pp. 1779-1792, December 2010.

[3] Marzieh Lenjani, and Mahmoud Reza Hashemi, "Tree- based scheme for reducing shared cache miss rate leveraging regional, statistical and temporal similarities," IET Computers & Digital Techniques, vol. 8, no.1, pp. 30-48, 2014.

[4] D. Yun, S. Kim, and S. Ha, "A Parallel Simulation Technique for Multicore Embedded Systems and Its Performance Analysis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 31, no. 1, January 2012.

[5] Santosh Nagarakatte, Milo M. K. Martion, and Steve Zdancewic, "Hardware for Safe and Secure Manual Memory Management and Full Memory Safety," In the Proceedings of the 39th International Symposium on Computer Architecture (ISCA 2012), pp. 189-200.

[6] Azzam Haidar, Stanimire Tomov, and Jack Dongarra, "A novel hybrid CPU-GPU generalized eigensolver for electronic structure calculations based on fine grained memory aware tasks," International Journal of High Performance Computing Applications, August 2013.

[7] Mario Lodde, Jose Flich, and Manuel E. Acacio, "Heterogeneous Noc Design for Efficient Broadcast-based Coherence Protocol Support," 2012 Sixth IEEE/ACM International Symposium on Networks-on-Chip, pp. 59-66

[8] Xiaolin Teng, Hoang Pham, and Daniel R. Jeske, "Reliability Modeling of Hardware and Software Interactions, and Its Applications," IEEE Transactions on Reliability, vol. 55, no. 4, pp. 571-577, December 2006.

[9] Xi E. Chen, and Tor M. Aamodt, "Modeling Cache Contention and Throughput of Multiprogrammed Manycore Processors," IEEE Transactions on Computers, vol. 61, no. 7, pp. 913-927, July 2012.

[10] Marvin Rausand, "System Reliability Theory: Models, Statistical Methods, and Applocations, 2ND Edition," 2004.