IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 20, 2017, accepted May 2, 2017, date of publication May 4, 2017, date of current version June 7, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2701406

Cache Coherence Scheme for HCS-Based CMP and Its System Reliability Analysis

SIZHAO LI AND DONGHUI GUO, (Senior Member, IEEE)
Department of Electronic Engineering, School of Information Science and Engineering, Xiamen University, Xiamen 361005, China
Corresponding author: Donghui Guo (dhguo@xmu.edu.cn)

This work was supported by the National Natural Science Foundation of China (General Program) under Grant 61274133.

ABSTRACT In previous work, a new network switch architecture, hybrid circuit-switched (HCS) network, has been proposed and evaluated. In doing so, it has been studied for use in a multi-processor system, with a focus on power and throughput. However, cache coherence and its connection with chip reliability have not been fully studied previously for multi-processor systems. In this paper, we study this problem by discussing the implementation of cache coherence on a HCS-based chip multi-processor and present a way to model the reliability of these protocols based on fault tree analysis and two-terminal networking models. We focus our efforts on three cache coherence protocols: Write-Once, Modified, Exclusive, Shared, Invalid (MESI), and Modified, Owned, Exclusive, Shared, Invalid (MOESI), and obtain expressions for the reliability probabilities of the system. Our results show that the Write-Once protocol is 14% less reliable than MOESI, while the MESI protocol is 2.5% less reliable than MOESI. We also demonstrate that the reliability of these protocols are 40.22% and 59.83% better, on average, when implemented on an HCS network rather than an elastic buffer-based network or a bus-based network, respectively.

INDEX TERMS Cache coherence, networking switch, system reliability, fault tree analysis (FTA), *2*-terminal model.

I. INTRODUCTION

With the stagnation of *Moore's Law* and the end of *Denard Scaling*, our reliance on multi-core processor arrays is steadily growing as we try to provide higher performance at lower power budgets. In this multi-core eta, there is a new focus on cache coherency, and its protocols, for chip multi-processors (CMP) [1] as well as system reliability. Ensuring that read operations have access to the latest written value [2] is of particular importance for correctness of operation when we think in context of CMPs. While we look at consistency as a global issue concerning programming models, coherency protocols are necessary to ensure correctness at a hardware-level. However, as we approach sub 10 nanometer CMOS designs, there are increased concerns surrounding the reliability of such systems. Thus, modern CMPs need to have robust communication schemes for reliable coherency amongst the cores.

In the past, several bus-based monitoring cache coherency protocols have been proposed [3]–[5]. They usually rely on write-update and write-invalidate operations to ensure data consistency. As described in [6], the new state records data which some processors do not need to update, and notifies the shared memory not to update the cache line in these processors, thereby reducing the number of write-update, and greatly improving the bus efficiency [7]. Although this scheme reduces the number of write-updates, it does not consider that the write-invalidates that must be transferred on the bus [8]. In [9], protocols concerned with recording invalid cache lines are proposed. Using the new state to recording invalid cache line, [9] suggests that invalid data broadcasts on the bus can be prevented. Both of the previously described schemes have been implemented on their corresponding multi-processor architecture.

In previous work, the Hybird Circuit-Switched (HCS) structure was proposed, which is an on-chip network for large-scale CMPs [10]. However, application of coherence protocols to this architecture was not discussed. So, in this work, we shall focus on the HCS structure and establish mathematical models, to verify the feasibility of cache coherence protocols on the HCS structure.

Three key aspects of modern microprocessor design are: power, on-chip storage and reliability [1], [11], [12]. In this work, we will set up mathematical models to describe the reliability of cache coherence protocols, and utilize these models to verify whether different cache coherence protocols are feasible and reliable on the HCS architecture. The study of

the reliability of networks has been attempted before in works such as [13] and [15]. In particular, [13] presented a reliability model for a mesh network, but cache coherence protocols were not discussed in this study. Reference [16] analyzed the MTTF (mean time to failure) of data caches and compared the cache memory block using different cache replacement policies.

In order to calculate the reliability of cache coherence protocols, a fault tree analysis (FTA) model has been proposed and discussed in [17]. In this work, we use the FTA model and also explore the *2*-terminal model in this paper. Both models are capable of describing the reliability of mesh networks.

The FTA model is a logical diagram; it uses event symbols and logic gate symbols to describe the causality between the events in the system [20]. Cache coherence protocol for CMPs rely on multiple states to resolve data inconsistencies; FTA can demonstrate the reliability of the cache coherence protocol intuitively. Thus, we can establish a FTA model for the cache coherence to evaluate the reliability. We also explore the Order Binary Decision Diagram (OBDD) method to calculate reliability for the large-scale networks [13]. The OBDD denotes a boolean function, at the same time it describes the value of boolean function and reduces the redundancy state. In our system, cache coherence deals with data inconsistency for every processor (node), and the pipeline channels (edge) deal with data transmission. We use the OBDD method [13] to construct the *2*-terminal model, and analyze the reliability of node and edge for network. However, in [13], the node was considered unreliable and the edge was considered reliable. However, our work assumes that edges in the network are not reliable.

Reliability analysis can be classified into ex-ante analysis, processing analysis and ex-post analysis [18], [19]: (1) Ex-ante analysis refers to predicting and preventing occurrence of faults and hazards in product design stage. (2) Processing analysis refers to forecasting the faults through fault monitoring and diagnosis technology in the product using stage. (3) Ex-post analysis refers to analyzing failure mechanism in the failure occurring stage. The goal of our work is to analyze existing protocols on the HCS network during its operation, so, we select the processing analysis method.

The contribution of this work can be summarized in two points. First, we present an analysis on the reliability of cache coherency schemes on different networks. Second, we use the OBDD method to analyze node and edge reliability. This differs from previous works that have focussed on either node or edge reliability alone.

The rest of this paper is organized as follows: Section 2 briefly discusses cache coherence protocols for multi- processor architectures, and explains the difference between the HCS networks and EB networks. In Section 3, we introduce the analysis of the reliability of a multi-processor system. Then, in Section 4, we propose the reliability model for cache coherence protocols on a multi-processor. Finally, simulation results and discussions are presented in Section 5, after which Section 6 concludes the paper.

II. CACHE COHERENCE PROTOCOLS FOR MULTI-PROCESSOR ARCHITECTURES

This section discusses CMP architectures. In earlier designs, the bus architecture, shown in Fig. 1, was a very popular option. However, with the increasing number of cores, the efficiency of the bus architecture has reduced dramatically, prompting the adoption of the network-based architectures. In this work, we analyze the reliability of a 64-core 2D mesh structure, with HCS.

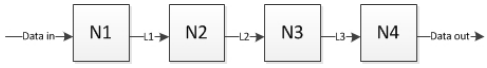


FIGURE 1. Bus architecture.

A. MESH STRUCTURE FOR MULTI-PROCESSOR

An HCS network is shown for a large-scale CMPs in Fig. 2. An HCS network consists of three components: network interfaces (NI), pipeline channels, and switches [10].

In the architecture shown, each NI that generates a data packet is linked to an L1 local cache, and all processors have a shared memory. Pipeline channels store packets generated by the NI, via a First-In-First-Out (FIFO) channel. The switch is comprised of five routing ports: one port for NI and four ports for transmission. A data arbiter selects data packet in the pipeline channel to be transmitted.

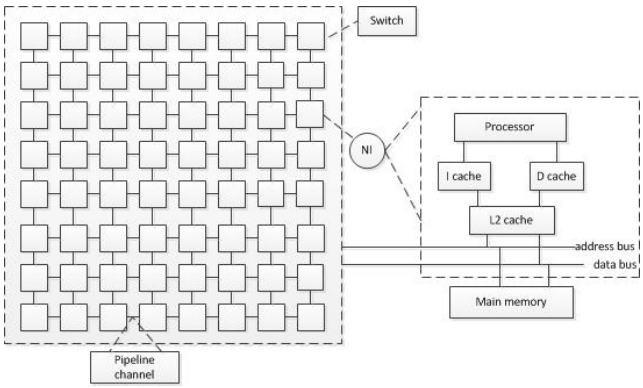


FIGURE 2. HCS and cache structure.

The links in an HCS network differ from that in an EB network. The links in an EB network (EB channels) are constructed using serial flip-flops (FF) [22], as illustrated in Fig. 3(a). In contrast, the HCS pipeline channels are constructed via parallel FFs, shown in Fig. 3(b) [10]. According to [22], the EB control logic uses 2 FFs, while the pipeline channel controller uses 4 FFs. Note that the structure of the EB channel creates multiple single-point failures, where a failure in a single FF can disrupt all communications. The reliability of the link R_L, will be calculated based on these structures.

B. CACHE COHERENCE PROTOCOL

Ensuring data consistency is a key problem in HCS and EB mesh structures. Cache coherence refers to the data consistency between local caches and main memory. If data has

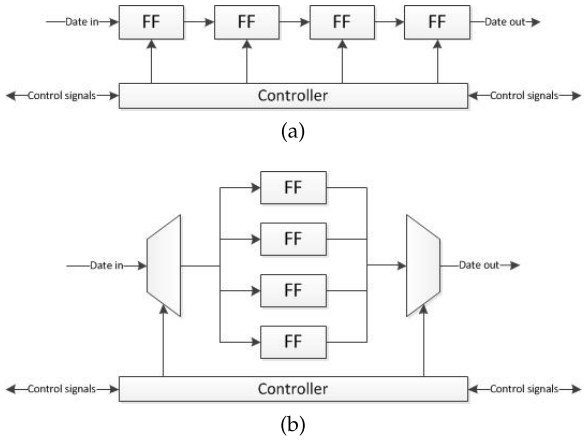


FIGURE 3. (a) serial FF and (b) parallel FF.

been modified in the LLC (low level cache), then it also must be modified in the HLC (high level cache) immediately. In a multi-processor, there may be inconsistent data between caches because the data didn't transfer from the HLC to the LLC [23], [24]. Data inconsistency can lead to program errors eventually, and the system will produce incorrect result that may lead to other errors or even a system crash. Therefore, solving the cache coherency problem is a major step towards system reliability.

In this work we will analyze the system reliability for different protocols. While several coherency protocols have be proposed and are in practice, we will discuss three classic coherency protocols. Other coherency protocols can be similar in spirit, with additional states added to address specific concerns or needs of the system.

1) WRITE-ONCE

The write-once [25] protocol is one of the earliest write-invalidate protocols. It is write-back and write-through comprehensive. The protocol defines four states in each node to identify the current state of the cache line: Invalid, Valid, Dirty, and Reserved. The invalid state indicates that the current cache line does not contain valid data. Valid indicates that the current data is the latest in the LLC, and there is a copy of the cache line in the HLC and there may be a copy in the cache line of other cores. Reserved also indicates that the data of current cache is the latest, and the HLC has a copy data of the cache line, however there are no other copies of the cache line on other cores. A dirty state represents that the data is the only copy and is not coherent with the main memory or any other cache line.

2) MESI AND MOESI

MESI (aka Illinois protocol) [26] is a cache coherence protocol widely used to support the write back strategy. As the name suggests, it defines four states: Modified, Shared, Exclusive, and Invalid.

In the modified state, the data only exists in the current LLC, and is different from HLC. Similarly, in the exclusive state, the data only exists in current LLC, but it is consistent to the HLC. In the shared state, data may be stored in other caches and is consistent with the HLC. Lastly, the invalid state describes a cache line that is invalid, similar to the write-once protocol.

The MOESI protocol [27] is similar to MESI, but it adds another state: Owned. In this state, the data available in the cache line is the latest and there must be a copy of this data in other local caches or LLCs. However, only the cache with the owned status may modify the data.

III. MULTI-PROCESSOR SYSTEM RELIABILITY

The growing popularity of CMPs mandates that rigourous analysis be done on their reliability; several studies have looked in to this [28]. With respect to this work, on-chip networks are becoming an integral part of CMPs and there have been past works that have dealt with their reliability, [13], [14]. For example, Yeh *et al.* [13] present a reliability analysis of multiple network structures via a quantitative analysis of data transmission over the network. However, these works just analyze the network reliability, and do not consider micro-architecture of the network structures or the coherency protocols. Hence, we explore the reliability of CMP networks in this work. First, we use fault tree analysis (FTA) to build a reliability model for common multi-processor models. Then, we use the Bayes formula to analyze the HCS network.

A. FAILURE MODEL AND RELIABILITY FUNCTION

When a processor is unable to complete its designated function, it is called a failure. We can use probability to describe the reliability of a system.

First we introduce the following three key concepts:

a) We define a time (t) function to denote the reliability of a multi-processor $R_{mp}(t)$. If T just represents the cache coherence protocol life, then the event ($T > t$) indicates that the system can work in time $[0, t]$.

$$R_{mp}(t) = \begin{cases} Pr(T > t), & t \geq 0, \\ 1, & t < 0. \end{cases} \tag{1}$$

If the event indicates that the system life is less than t, it is a failure in $[0, t]$, $u \in [0, t]$. So, $F_{mp}(t) = Pr(T \leq t)$ is called the failure distribution function of the multi-processor,

$$F_{mp}(t) = Pr(T \leq t) = \int_0^t f_{mp}(u)du, t > 0 \tag{2}$$

If this probability density function $f_{mp}(t)$ is known, then when given value T we can calculate $R_{mp}(t)$.

$$R_{mp}(t) = 1 - F_{mp}(t) = \int_t^\infty f_{mp}(u)du \tag{3}$$

where $R_{mp}(t)$ is the no failure probability of the state in the time interval $(0, t]$. From formulas (2) and (3), we can get $R_{mp}(0) = 1$, $R_{mp}(\infty) = 0$, $F_{mp}(0) = 0$ and $F_{mp}(\infty) = 1$. Note that with an increasing value of t, the failure possibility increases and the working time possibility reduces.

b) The system failure rate is an important index in our model. It is the primary data required to calculate the reliability index. The system failure rate is defined at time t and under the conditions that the protocol is working before time t and that failure occurs in the time interval $(t, t + \Delta t]$. It is denoted as $z_{mp}(t)$.

$$\begin{aligned} z_{mp}(t) &= \lim_{\Delta t \longrightarrow 0} \frac{Pr(t < T \leq t + \Delta t | T > t)}{\Delta t} \\ &= \lim_{\Delta t \longrightarrow 0} \frac{F_{mp}(t + \Delta t) - F_{mp}(t)}{\Delta t} \cdot \frac{1}{R_{mp}(t)} \\ &= \frac{f_{mp}(t)}{R_{mp}(t)} \end{aligned} \tag{4}$$

c) We define the life of the system as the total working time before the failure occurrs. So, the Mean Time to Failure is also the expectation of the system life ($MTTF_{mp}$),

$$MTTF_{mp} = E(T) = \int_0^\infty t f_{mp}(t)dt = \int_0^\infty R_{mp}(t)dt \tag{5}$$

The system failure rate is random and represents cache failure via the distribution function of failure. Depeding on the applied principle, failure can occur in discrete distribution or continuous distribution. However, the data error, due to system failure, is a continuous distribution in the cache and is represented through continuous distribution. So, $f_{mp}(t)$ meets an exponential distribution [30].

B. SYSTEM ANALYSIS

The system is composed of many components that are connected according to a certain production objective. The reliability of the system is a function of the reliability of its components and the structure of the system. The goal of system analysis, given component failure data and the system structure, is to predict the reliability of the systems via reliability indices such as failure probabilities. In this section, we use FTA and 2-terminal method to analyze system reliability.

This section also explores the reliability of the HCS network via a network analysis method. If the logic diagram and the failure probability of each component is known, through appropriate operations, the whole systemï£¡ï£¡s reliability index can be derived. Note that in order to use the network analysis method, a logic diagram of the object must exist and be expressed clearly.

We shall now discuss the analysis of various system topologies:

1) SERIAL SYSTEM

In a serial system, the failure of any unit can result in failure of the whole system [31].

Assume that the life of the i-th processor in the system is $t_i (i = 1, 2, \ldots, n)$, and $t_1, t_2, \ldots, t_n$ are independent of each other. Then the i-th processor's reliability can be obtained from Equation (1),

$$R_{mp}(t) = P|t_i > t| \quad (i = 1, 2, \ldots, n) \tag{6}$$

where t is the prescribed working time. Due to the serial nature of the system, its life is $\tau = \min(t_1, t_2, \ldots, t_n)$, and thus the system reliability can be expressed by:

$$\begin{aligned} R_{mp}(t) &= P|\tau > t| = P|t_1 > t, t_2 > t, \ldots, t_n > t| \\ &= \prod_{i=1}^{n} P|t_i > t| = \prod_{i=1}^{n} R_i(t) \end{aligned} \tag{7}$$

If the i-th processor's failure rate is $\lambda_i(t)$, then the unit reliability is:

$$R_i(t) = \exp[-\int_0^t z_i(u)du] \tag{8}$$

From equations (7) and (8), the system reliability can be expressed as:

$$\begin{aligned} R_{mp}(t) &= \prod_{i=1}^{n} R_i(t) = \exp[-\sum_{i=1}^{n} \int_0^t z_i(u)du] \\ &= \exp[-\int_0^t z_s(u)du] \end{aligned} \tag{9}$$

where, $z_s(t) = \sum_{i=1}^{n} z_i(t)$ is the system failure rate.

The serial system's mean time to failure is

$$MTTF_s = \int_0^\infty R_s(t)dt = \int_0^\infty \exp[-\int_0^t z_s(u)du] \tag{10}$$

This analysis reveals three key points. First, the reliability of a serial system, $R_s(t)$, is equal to the product of unit reliabilities R_i. Second, the reliability of a serial system, $R_s(t)$, is less than or at most equal to the reliability of the least reliable unit in the system. Third, from equation (10), when the number of components, n is increased, the $MTTF_s$ decreases i.e., when the number of units in the serial system increases, the system reliability lowers.

2) PARALLEL SYSTEM

A system is called a parallel system if the entire system fails when all units fail [31].

Assume that the life of the i-th processor in the system is $t_i (i = 1, 2, \ldots, n)$, and $t_1, t_2, \ldots, t_n$ are independent of each other. The i-th processor's reliability can be described as $R_{mp}(t) = P|t_i > t| (i = 1, 2, \ldots, n)$, given that the initial time $t = 0$, all units are normal, and begin to operate at exactly the same time. The parallel system's life is $\tau = \max(t_1, t_2, \ldots, t_n)$. Thus, the system reliability can be expressed as:

$$\begin{aligned} R_{mp}(t) &= P|\tau > t| = 1 - P|t_1 \leq t, t_2 \leq t, \ldots, t_n \leq t| \\ &= 1 - \prod_{i=1}^{n} [1 - R_i(t)] \end{aligned} \tag{11}$$

When the i-th processor's failure rate is z_i , the system reliability is:

$$R_{mp}(t) = 1 - \prod_{i=1}^{n} (1 - e^{-z_i t}) \tag{12}$$

The parallel system's mean time to failure is

$$MTTF_p = \int_0^\infty R_{mp}(t)dt = -\sum_{1\le i<j\le n}\frac{1}{z_i+z_j}+\cdots + (-1)^{n-1}(\sum_{i=1}^{n} z_i)^{-1} \quad (13)$$

3) SERIAL-PARALLEL (HYBRID CONNECT) SYSTEM

When the serial and parallel structures co-exist in the system, it is called a hybrid connected system. The entire system meets the criterion for serial or parallel, and smaller sub-parts are described as parallel or serial in structure. The reliability of such a system is solved step by step. First by calculating the reliability of smaller sub-parts, and then combining them to derive the reliability of the entire parallel (or serial) system.

4) 2-TERMINAL RELIABILITY FUNCTION

FTA cannot describe the mesh architecture accurately, so we can use the 2-terminal model to evaluate network.

A graph $G = (U, E)$ consists of two sets: U is the node set, and E is the edge set. A edge e is working if two nodes u_1, u_2 are connected. This working path is: 2-working path. The 2-working path include all edges which are connected to the two nodes either directly or indirectly [30]. Therefore, the 2-terminal reliability is the probability that a 2-working path exists between each pair of all nodes. Then, the reliability for network can be described by $R(G, \Phi, \Psi)$. Where, $\Phi \to [0, 1]$ is the probability that a node fault occurs, and $\Psi \to [0, 1]$ is the probability that an edge fault occurs.

We define two states in the graph G: working and faulty and the working probability distribution of components (nodes and edges) meets the $0-1$ distribution. A random variable x_i represents the i-th component such that $x_i = 1$ represents the i-th component working, with probability $Pr\{x_i = 1\}$ and $x_i = 0$ indicates that the i-th component is faulty, with probability $Pr = \{x_i = 0\} = 1 - Pr\{x_i = 1\}$. Thus, the random variable $x = (x_0, x_1, \cdots, x_{n-1})$ represents the state of a network G that has n components. The probability of the network in the state X can be expressed as

$$Pr\{x = X\} = \prod_{i=0}^{n-1}(x_i \times Pr\{x_i = 1\} + \{1 - x_i\} \times Pr\{x_i = 0\}) \quad (15)$$

Based on the state of each component, working or faulty, the state space of the network state, Ω is created with 2^n states. So the boolean function is

$$f_b(x) = \begin{cases} 1, & if\ all\ nodes\ can\ link, \\ 0, & otherwise. \end{cases} \quad (16)$$

And the reliability of networking can be expressed as

$$Rel(G) = \sum_{x\in\Omega} Pr\{f_b(x) = 1\} \quad (17)$$

Based on function (15), a binary tree can directly express a boolean function. However, binary tree have redundant states, so Binary Decision Diagrams (BDD) were proposed to express boolean functions [21]. In the BDD structure, a circle node is used to represent a boolean variable and a rectangle is used to represent logical true or false. Every circle node has two edges, in which the full line implies true and the dotted line implies false.

If the BDD created by the function can be in accordance with a determined variable order, then it is called an Ordered Binary Decision Diagram (OBDD) [13]. The operations on a OBDD includes the following.

1) *and* operation ($\wedge$): The *and* operation results in a OBDD function which is the boolean AND of the input OBDDs i.e. $(a \wedge b)$ and $(c \wedge d)$ produce $(a \wedge b) \wedge (c \wedge d)$ as shown in fig. 4.

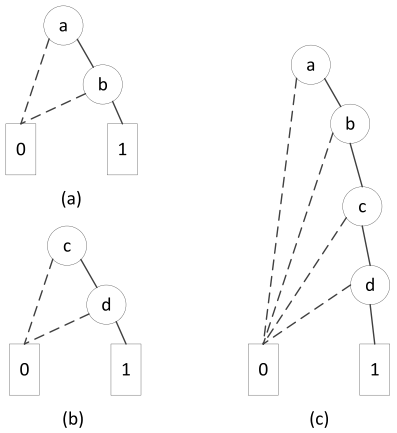


FIGURE 4. ***and*** **operation of the OBDD.**

2) *or* operation ($\vee$): The *or* operation result in a OBDD function which is the boolean OR of the input OBDDs i.e. $(a \wedge b)$ and $(c \wedge d)$ produce $(a \wedge b) \vee (c \wedge d)$ as shown in fig. 5.

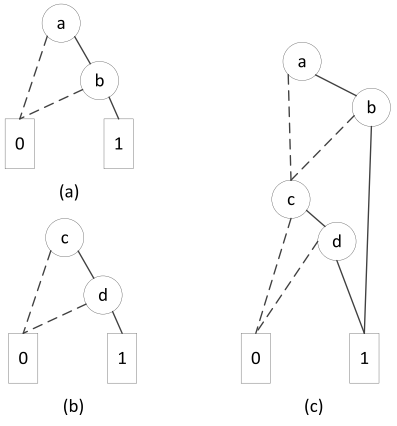


FIGURE 5. ***or*** **operation of the OBDD.**

Traditionally, reliability analysis has assumed that the edge was reliable and the node was unreliable [32].However, in this work, we analyze the reliability of node and edge in the HCS network. To calculate the reliability, we begin with the OBDD method to establish a method to analyze cache coherence on a HCS network. We use the OBDD method so that we may

build the *2*-terminal model effectively, and reduce redundant nodes.

The reliability evaluation algorithm, based on OBDD, includes two steps: constructing the OBDD for the network and traversing the OBDD to calculate network reliability:

(a) Constructing the $OBDD(G)$

Step 1 If source and end are merged into a single node, then return $OBDD(true)$.

Step 2 If there is redundant node in the current sub-network, then redundant node should be removed to reduce invalid calculation. Otherwise, execute the next step.

Step 3 Check whether the current sub-network is recorded in the hash table of the isomorphic sub-network or not. If there is a record, return OBDD node and save it in the hash table. Otherwise, execute the next step.

Step 4 For each edge e from source:

1) To simplify network, we can execute the operation of deleting an edge, which will give the sub-network G_{-e}: delete one of edge e from the source, combine both nodes of edge e.

2) Use "Step 1" to construct OBDD for sub-network G_{-e}, then return $OBDD(G_{-e})$.

3) Execute the OBDD logic operation:

$OBDD(G') = OBDD(e)\ and\ OBDD(G_{-e})$

$OBDD(G) = OBDD(G)\ or\ OBDD(G')$

Step 5 The $OBDD(G)$ will be recorded to the hash table.

(b) Calculating network reliability

We calculate the reliability of the network by traversing the OBDD. We use a recursive formula, (14), as shown at the bottom of this page, where $OBDD(G)|_{x_k=1}$ is $x_k = 1$ of the OBDD, i.e. x_k is the 1-child for the node; $OBDD(G)|_{x_k=0}$ is $x_k = 0$ of the OBDD, i.e. x_k is the 0-child for the node. $Pr(x_k = 1)$ is the working probability of the edge e_k; $Pr(x_k = 0)$ is the fault probability of the edge e_k. Thus, the reliability of the cache coherence protocol can be used to calculate node reliability and data transmission reliability can be used as edge reliability. Assuming *obdd-true* and *obdd-false* represent logical true and logical false respectively in the OBDD structure, and the initial value of k is 0. Calculating network reliability can be done as follows:

Step 1 Access the variable x_k for the root node in OBDD, to obtain the work probability p_k of the corresponding edge.

Step 2 Determination of $OBDD(G)$ value: If it is *obdd-true*, then return value 1.0; if it is *obdd-false*, then return value 0.0. Else, goto next step.

Step 3 Determine whether this OBDD in the hash table has reliability value or not. If it exists, then return this reliability value. Else, goto next step.

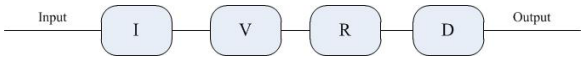


FIGURE 6. Write-Once serial flow chart.

Step 4 Calculate $Rel(OBDD(G)|_{x_k=0}$ and $Rel(OBDD(G)|_{x_k=1}$.

Step 5 The reliability value will be inserted into hash table, and then return the reliability of G, given by:

$Rel(OBDD(G)) = p_k \times Rel(OBDD(G)|_{x_k=1} + (1 - p_k) \times Rel(OBDD(G)|_{x_k=0}$

Step 6 Return to calculate network reliability.

IV. RELIABILITY MODEL FOR CACHE COHERENCE PROTOCOLS

By the definition of reliability for the multi-processor system, we get the required numerical value of the reliability model. These values are also required in some other reliability model.

Using the earlier defined parameters for reliability of the system, a numerical value can be derived to express the reliability of a CMP system based on an estimated reliability model. [29] uses the failure rate model and the MTTF to set up reliability model for a software system. We build on these medthods to develop our models. We establish a relability model for the HCS network using the Fault tree analsysis (FTA) model and the Baye's network model, discussed in section 3. In this section we discuss the modeling process.

A. RELIABILITY MODELS OF CACHE COHERENCE PROTOCOLS

We begin our analysis for the simplest of the three protocols; Write-Once. In the Write-Once protocol, each state affects the whole system i.e. a fault in any state can lead to cache inconsistency. The Write-Once protocol is composed of four states: Invalid(I), Valid(V), Reserved(R) and Dirty(D). When each state is functional, the whole protocol can work. So, the Write-Once protocol can be modeled as a serial system. The following formula refers to the Write-Once reliability model and shown in Fig. 6.

$$R_{write-once} = R_I \cdot R_V \cdot R_R \cdot R_D = \prod_{i=1}^{4} R_i$$

As mentioned earlier, the Write-Once protocol is a serial-system and can be analyzed as one. When the failure distribution of each state is an exponential distribution, that is

$$Rel(OBDD(G)) = \begin{cases} 1, & if\ OBDD(G) = ture, \\ 0, & if\ OBDD(G) = false, \\ Pr(x_k = 1) \times Rel(OBDD(G)|_{x_k=1}) & \\ \quad + Pr(x_k = 0) \times Rel(OBDD(G)|_{x_k=0}), & 0 \leqslant k < n, others. \end{cases} \tag{14}$$

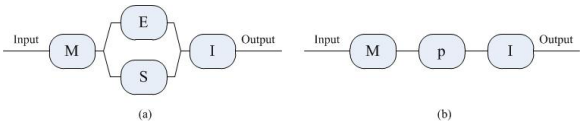


FIGURE 7. MESI serial-parallel flow chart.

$R_i(t) = e^{-\lambda_i t}$, then the system reliability is

$$R_{write-once} = \prod_{i=1}^{4} e^{-\lambda_i t} = e^{-\lambda_\xi t}$$

The failure rate is the probability of the loss of system functions in the stipulated conditions and within the 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 the system is

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

$$MTTF_{write-once} = \frac{1}{\lambda_\xi} = \frac{1}{\sum_{i=1}^{4} \lambda_i}$$

Building on this, we now analyze the MESI protocol. In the MESI protocol, Exclusive(E) is a special case of Shared (S) and we model MESI as a serial-parallel system. The model for the serial-parallel MESI system is shown in Fig. 7(a). In Fig 7(a), the state M is recorded as R_M and I is recorded as R_I, state E and S are in parallel and we can record these as R_p. Finally, R_M, R_I and R_p cascade into R_s as shown in Fig. 7(b). The following formula describes the MESI reliability model,

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

$$R_{MESI} = R_M R_p R_I = R_M (R_E + R_S - R_E R_S) R_I$$

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

The probability density function is

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

So the failure rate function of the system is

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

$$MTTF_{MESI} = \int_0^\infty R_{MESI}(t)dt$$

Finally, the MOESI protocol is similar to the MESI protocol. R_M and R_I have the same functions, and state O, state E and state S can also be considered as a parallel system, which can be recorded as R_p. We can use the Fig. 7 method to set up mathematical model. This model differs from what is shown in Fig. 8, a fault tree [20]. The fault tree model is an alternative description for reliability. In the fault tree model, if the system is serial, all the nodes are connected to a OR gate, and if the system is parallel, it can be modeled via an AND gate.

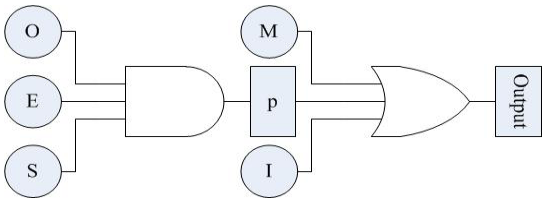


FIGURE 8. MOESI fault tree.

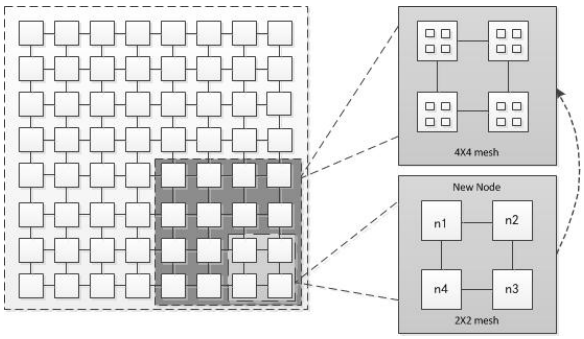


FIGURE 9. The process of the 2 × 2 mesh to the 4 × 4 mesh.

This model can be used to express the following formula:

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

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

So, the failure rate function is given by the formula (18), as shown at the bottom of this page.

B. ANALYSIS OF MULTI-PROCESSOR RELIABILITY MODEL

The HCS network has a symmetrical structure. Each NI has the same weight and a minimal subsystem can be seen as a 2 × 2 2D mesh structure. After computing the reliability of this subsystem, these four nodes can be seen a new node, we can also use this method to compute the reliability of a 4 × 4 mesh, as shown Fig. 9. Finally, the reliability of an 8 × 8 mesh can be calculated. Having derived the reliability of the network, a cache coherence protocol can be implemented on top of HCS or EB.

To calculate the system reliability, which includes the reliability of all pairs of nodes, we use the 2-terminal reliability model. Fig. 10 illustrates the computation of the reliability of a node. In Fig. 10, we illustrate the process for $n1$ to other nodes. Fig. 11 is the OBDD, and use it to calculate 2 × 2 network reliability.

To complete the process shown in Fig. 9, we need a value for the reliability of the edge, R_e. We assume that the

$$z(t) = \frac{f(t)}{R_{MOESI}(t)} = \frac{\lambda_\xi e^{-(\lambda_E + \lambda_O + \lambda_S)t}}{e^{-\lambda_E t} + e^{-\lambda_O t} + e^{-\lambda_S t} - e^{-(\lambda_E + \lambda_O)t} - e^{-(\lambda_E + \lambda_S)t} - e^{-(\lambda_O + \lambda_S)t} + e^{-(\lambda_E + \lambda_O + \lambda_S)t}} \quad (18)$$

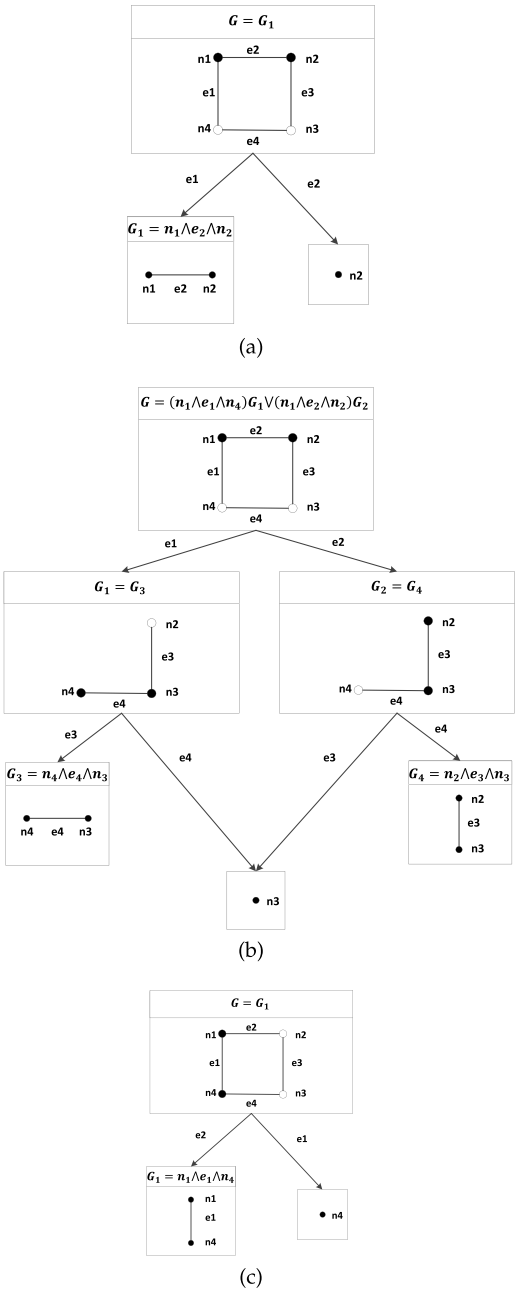


FIGURE 10. The process of construct the OBDD.

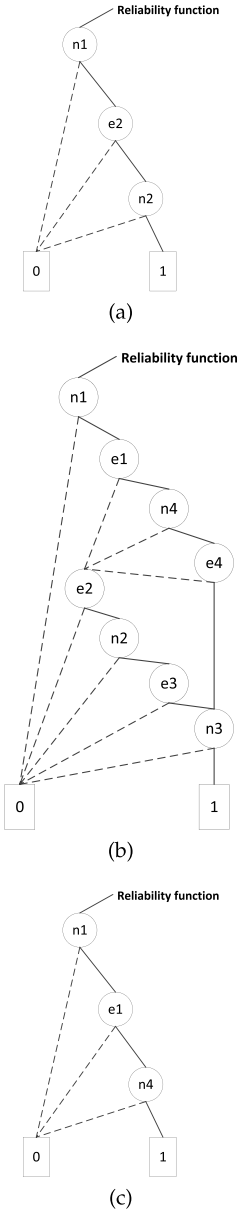


FIGURE 11. OBDD.

flip-flops (FF) reliability is R_F. Since the EB channel has serial flip flops, its reliability is given as $R_{eE} = (R_F)^4$. For the pipelined channel in the HCS network, $R_{eH} = 1-(1-R_F)^4$. This network is isomorphic, so $R_N = R_{n1} = R_{n2} = R_{n3} = R_{n4}$ and $R_e = R_{e1} = R_{e2} = R_{e3} = R_{e4}$.

Thus, using the methods illustrated in the section, for a 2×2 network, the reliability of a 64-core network can be computed by sequentially reducing the the analysis down to a 2×2 network.

V. SIMULATION RESULTS AND DISCUSSION

Evaluating the reliability of cache coherence protocols can help improve overall system stability. We have used our models to perform experiments comparing Write-Once, MESI and MOESI schemes. This section discusses the results of the MATLAB simulations based on the methods described in section 4, and a Virtex-7 FPGA is used verify the correctness of the model.

A. CACHE COHERENCE RELIABILITY RESULT

For a quantitative analysis of the mathematical models, we only evaluate the protocol, not a specific hardware platform. Through simulations, we can assess the advantages and disadvantages of each protocol. Table 1 summarizes the reliability models of the three protocols we evaluate. Based on Table 1, we can compute the MTTF, and use these results to compare the results on the FPGA. Since, the failure rate of the control

TABLE 1. 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	$\frac{1}{4}\lambda$
MESI	$4\lambda e^{-4\lambda t}$	$2e^{-3\lambda t} - e^{-4\lambda t}$	$\frac{4\lambda}{2e^{\lambda t}-1}$	$\frac{5}{12}\lambda$
MOESI	$5\lambda e^{-5\lambda t}$	$3e^{-3\lambda t} - 3e^{-4\lambda t} + e^{-5\lambda t}$	$\frac{5\lambda}{3e^{2\lambda t}-3e^{\lambda t}+1}$	$\frac{9}{20}\lambda$

TABLE 2. Write-once protocol comparison.

Model	MTTF($\times 10^6$ms)
Perm	7.79
Towers	12.62
Queens	9.15
Intmm	9.98
Mm	9.58
Puzzle	8.63
Quick	8.80
Bubble	8.01
Tree	9.65
FFT	10.24
Write-Once model	7.75

TABLE 3. MESI protocol comparison.

Model	MTTF($\times 10^6$ms)
Perm	16.93
Towers	16.18
Queens	17.01
Intmm	20.39
Mm	14.02
Puzzle	14.14
Quick	16.90
Bubble	17.10
Tree	16.26
FFT	15.05
MESI model	12.92

TABLE 4. MOESI protocol comparison.

Model	MTTF($\times 10^6$ms)
Perm	14.40
Towers	19.16
Queens	16.26
Intmm	15.03
Mm	11.86
Puzzle	14.50
Quick	16.00
Bubble	16.87
Tree	16.35
FFT	14.92
MOESI model	13.95

logic block is usually a fixed value: $\lambda = 0.31 \times 10^{-6}$ [30], we assume that the failure rates of all states are the same in all three cache coherence protocols, and that the failure distribution is exponential. Table 2 to Table 4 summarize the results and findings for the three cache coherence protocols. They present the final MTTF computed for the *stanford benchmark*.

Using the derived model, Fig. 12 and Fig. 13 show the results of our evaluation. From Fig. 12, we can see that $R_{Write-Once} < R_{MESI} < R_{MOESI}$, i.e., MOESI is the most reliable protocol, but only by a small margin when compared

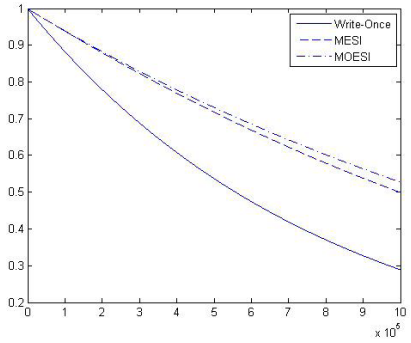


FIGURE 12. The reliability function curve diagram.

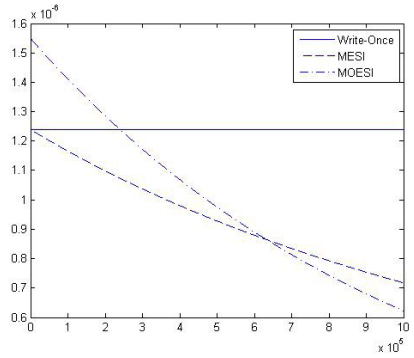


FIGURE 13. The failure rate function curve diagram.

to the MESI protocol. In Fig. 12, $z_{Write-Once}$ is a constant because it is the serial model in which all the parameters are constant. Over an extended period of time, z_{MESI} has the lowest failure rate, and during the same time, the failure rate of MOESI is consistently higher. This is one of the reasons that researchers suggest the use of MESI protocol [3].

B. HCS RELIABILITY RESULT

In previous work, our HCS network has been implemented [10], but only evaluated for power, area and throughput. In this work, we will use data acquired from ModelSim simulations and FPGA emulation.

We now set up a model for HCS reliability, which we will compare with a bus structure. Note that the bus structure is a series system, so $R_B = R_N \times (R_{e1} \times R_{e2} \times R_{e3})$.

Using results from Fig. 12, we extract reliability values at different times. The values we select for each node (Write-once, MESI, MOESI) are: $R_{n1} = (0.90, 0.95, 0.96)$, $R_{n2} = (0.80, 0.90, 0.92)$, $R_{n3} = (0.70, 0.82, 0.85)$ and $R_{n4} = (0.60, 0.79, 0.83)$.

These values represent the cache line reliability of the nodes shown in Fig. 10, and assume $R_F = 0.9$. So, we can get $R_{eE} = 0.81$ and $R_{eH} = 0.9999$.

Having derived the reliability of the four nodes, these four nodes can be seen as a single new node. Four of these supernodes can be used to analyze an HCS of 16 nodes. Using this method, the reliability of the entire 64 node HCS can be deduced.

The result of this evaluation is shown in Fig. 14. We can see that the reliability of the HCS structure is better than the EB and bus structure, and the reliability of MOESI is higher than that of others, which is agreement with our earlier analysis.

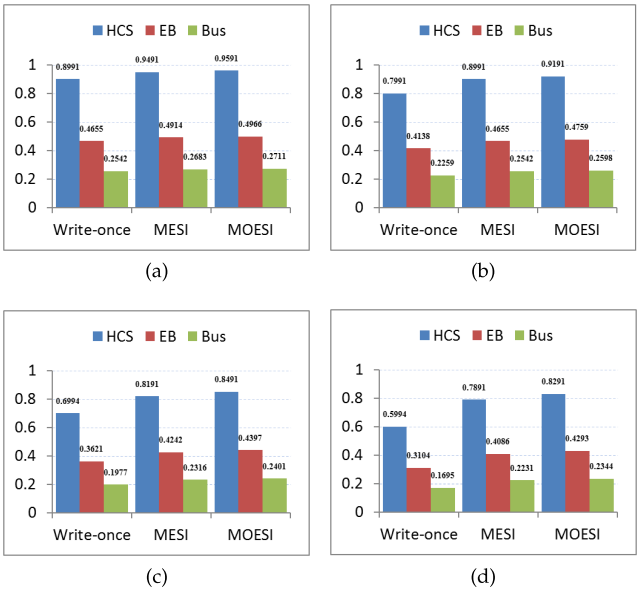


FIGURE 14. **Compare the reliability of cache coherence protocol between HCS, EB and Bus.**

VI. CONCLUSION

In this work, we proposed a model for cache coherence reliability, and compared the reliability of the HCS network with the EB network and the bus structure. There have been very few efforts in literature that have used mathematical models for reliability analysis. So, we proposed a reliability model for cache coherence protocol.

In this paper, we extend previous studies and, through a mathematical method, establish the reliability model of three cache coherence protocols. This work will help improve the reliability of hardware design. In particular we proposed a new math model for cache coherence protocols with the following: **a)** the failure model, MTTF and reliability function can describe the stability of cache coherence protocol; **b)** using fault tree model and Bayes formula to analyze the cache coherence of HCS network reliability.

By using these models to analyze a hardware system, a reliability result can be achieved. The results are presented in Section 5, in Fig. 11, the MOESI reliability is the highest, but in Fig. 12, before t equals 6.23×10^6, the failure rate of MESI is far less than that of MOESI. And now, though most hardware systems use the MESI protocol, they have not been mathematically verified. HCS network is more reliable than EB and bus structure. So we will focus our future studies on the HCS network in the future, based on these results. For example, we will investigate memory management, process scheduling, etc.

ACKNOWLEDGMENTS

The authors would like to thank their laboratory team member's assistance.

REFERENCES

[1] A. Ros, M. E. Acacio, and J. M. Garcia, "A direct coherence protocol for many-core chip multiprocessors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 12, pp. 1779–1792, Dec. 2010.

[2] H. Hossain, S. Dwarkadas, and M. C. Huang, " POPS: Coherence protocol optimization for both private and shared data," in *Proc. Parallel Archit. Compil. Techn. (PACT)*, 2011, pp. 45–55.

[3] G. Delzanno, "Automatic verification of parameterized cache coherence protocols," in *Computer Aided Verification* (Lecture Notes in Computer Science). 2000.

[4] P. Stenstrom, "A survey of cache coherence schemes for multiprocessors," *Computer*, vol. 23, no. 6, pp. 12–24, 1990.

[5] P. G. de Massas and F. Pétrot, "Comparison of memory write policies for NoC based multicore cache coherent systems," in *Proc. Design, Autom. Test Eur. (DATE)*, 2008, pp. 997–1002.

[6] Q. Yang, G. Thangadurai, and L. N. Bhuyan, "Design of an adaptive cache coherence protocol for large scale multiprocessors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 3, no. 3, pp. 281–293, May 1992.

[7] X. Yao and J. Wang, "RIMAC: A novel redundancy-based hierarchical cache architecture for energy efficient, high performance storage systems," in *Proc. 1st ACM SIGOPS/EuroSys Eur. Conf. Comput. Syst. (EuroSys)*, 2006, pp. 249–262.

[8] G. Quintana-Ortí, F. D. Igual, E. S. Quintana-Ortí, and R. van de Geijn, "Solving dense linear systems on platforms with multiple hardware accelerators," in *Proc. 14th ACM SIGPLAN Symp. Principles Pract. Parallel Program.*, 2009, pp. 121–130.

[9] M. Heinrich, V. Soundararajan, J. Hennessy, and A. Gupta, "A quantitative analysis of the performance and scalability of distributed shared memory cache coherence protocols," *IEEE Trans. Comput.*, vol. 48, no. 2, pp. 205–217, Feb. 1999.

[10] H. Luo, S. Wei, D. Chen, and D. Guo, "Hybrid circuit-switched network for on-chip communication in large-scale chip-multiprocessors," *J. Parallel Distrib. Comput.*, vol. 74, no. 9, pp. 2818–2830, 2014.

[11] A. M. Saleh, J. J. Serrano, and J. H. Patel, "Reliability of scrubbing recovery-techniques for memory systems," *IEEE Trans. Rel.*, vol. 39, no. 1, pp. 114–122, Apr. 1990.

[12] S. Borkar, "Thousand core chips: A technology perspective," in *Proc. 44th Annu. Design Autom. Conf.*, 2007, pp. 746–749.

[13] F.-M. Yeh, S.-K. Lu, and S.-Y. Kuo, "OBDD-based evaluation of K-terminal network reliability," *IEEE Trans. Rel.*, vol. 51, no. 4, pp. 443–451, Dec. 2002.

[14] M. Imai and T. Yoneda, "Fault diagnosis and reconfiguration method for network-on-chip based multiple processor systems with restricted private memories," *IEICE Trans. Inf. Syst.*, vol. E96-D, no. 9, pp. 1914–1925, 2013.

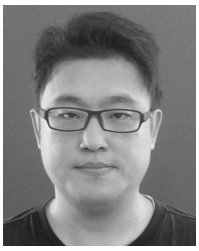
[15] L. Yang, S. Li, B. Xu, and Y. Xiao, "An OBDD-based method for the reliability analysis of construction system," in *Proc. Int. Conf. Manage. Service Sci. (MASS)*, 2010, pp. 1–4.

[16] M. Maghsoudloo and H. T. Zarandi, "Reliability improvement in private non-uniform cache architecture using two enhanced structures for coherence protocols and replacement policies," *Microprocessor Microsyst.*, vol. 38, no. 6, pp. 552–564, 2014.

[17] S. Li, S. Lin, D. Chen, W. E. Wong, and D. Guo, "Analysis of system reliability for cache coherence scheme in multi-processor," in *Proc. IEEE 8th Int. Conf. Softw. Secur. Rel.-Companion (SERE-C)*, Jun./Jul. 2014, pp. 247–251.

[18] M. Carey, "Ex ante heuristic measures of schedule reliability," *Transp. Res. B, Method.*, vol. 33, no. 7, pp. 473–494, 1999.

[19] F. Borrás and J. T. Pastor, "The ex-post evaluation of the minimum local reliability level: An enhanced probabilistic location set covering model," *Ann. Oper. Res.*, vol. 111, no. 1, pp. 51–74, 2002.
[20] W. S. Lee, D. L. Grosh, F. A. Tillman, and C. H. Lie, "Fault tree analysis, methods, and applications—A review," *IEEE Trans. Rel.*, vol. R-34, no. 3, pp. 194–203, Aug. 1985.
[21] R. E. Bryant, "Symbolic Boolean manipulation with ordered binary-decision diagrams," *ACM Comput. Surv.*, vol. 24, no. 3, pp. 293–318, 1992.
[22] G. Michelogiannakis and W. J. Dally, "Elastic buffer flow control for on-chip networks," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 295–309, Feb. 2013.
[23] J. Handy, *The Cache Memory Book*. San Mateo, CA, USA: Morgan Kaufmann, 1998.
[24] P. Prieto, V. Puente, and J.-A. Gregorio, "Multilevel cache modeling for chip-multiprocessor systems," *IEEE Comput. Archit. Lett.*, vol. 10, no. 2, pp. 49–52, Jul./Dec. 2011.
[25] J. R. Goodman, "Using cache memory to reduce processor-memory traffic," in *Proc. 10th Annu. Int. Symp. Comput. Archit. (ISCA)*, 1984, pp. 124–131.
[26] M. S. Papamarcos and J. H. Patel, "A low-overhead coherence solution for multiprocessors with private cache memories," in *Proc. 11th Annu. Int. Symp. Comput. Archit. (ISCA)*, 1983, pp. 348–354.
[27] *AMD64 Architecture Programmer's Manual Volume 2: System Programming*, Adv. Micro Devices, Sunnyvale, CA, USA, 2013.
[28] G. Beltrame, C. Bolchini, L. Fossati, A. Miele, and D. Sciuto, "A framework for reliability assessment and enhancement in multi-processor systems-on-chip," in *Proc. 22nd IEEE Int. Symp. Defect Fault-Tolerance VLSI Syst. (DFT)*, Sep. 2007, pp. 132–142.
[29] S. Inoue and S. Yamada, "Generalized discrete software reliability modeling with effect of program size," *IEEE Trans. Syst. Man, Cybern. A, Syst. Humans*, vol. 37, no. 2, pp. 170–179, Mar. 2007.
[30] M. Rausand and A. Høyland, *System Reliability Theory: Models, Statistical Methods, and Applications*. Hoboken, NJ, USA: Wiley, 2004.
[31] Y. Yamato, N. Shigematsu, and N. Miura, "Evaluation of agile software develeopment method for carrier cloud service platform development," *IEICE Trans. Inf. Syst.*, vol. E97-D, no. 11, pp. 2959–2962, 2014.
[32] G. Hardy, C. Lucet, and N. Limnios, "K-terminal network reliability measures with binary decision diagrams," *IEEE Trans. Rel.*, vol. 56, no. 3, pp. 506–515, Sep. 2007.



SIZHAO LI received the B.S. degree from Jilin University, Changchun, China, in 2007, and the M.S.E. degree from the University of Science and Technology of China, Hefei, China, in 2010. He is currently pursuing the Ph.D. degree in electronic engineering with Xiamen University. He is involved in software behavior feature and solving cache coherence based on NoC. His research interests include system reliability analysis, computer architecture, and hardware-software co-design.

DONGHUI GUO received the B.S. degree in radio physics, the M.S. degree, and the Ph.D. degree in semiconductor from Xiamen University, China, in 1988, 1991, and 1994, respectively. In 1994, he joined Xiamen University as a Faculty Member, where he has been a Full Professor, since 2002. He held a post-doctoral, a research fellow, a senior scientist, and a visiting scholar position, respectively, with the CityU of Hong Kong, University of Ulster, Lawrence Berkeley Laboratory, University California at Berkeley, and University of Illinois at Urbana-Champaign, and served as the Vice-Dean with the School of Information Science & Technology, Xiamen University, from 2008 to 2012. He is currently the Director of the IC Design & IT Research Center. His research areas include artificial intelligence, network computing, IC design, nano device, and BioMEMS.