Software-based online self-testing using bounded model checking for 3D network-on-chips

Pengfei Ji12, Ying Zhang12*, Jianhui Jiang1
1School of Software Engineering, Tongji University, Shanghai, China
2Zhejiang Lab, Zhejiang, China
*Corresponding author’s e-mail: yingzhang@tongji.edu.cn

Abstract. Online testing has become essential for the reliable operations of manycore systems using three-dimension (3D) network-on-chip (NoC) fabric. This paper presents a software-based online self-testing solution using bounded model checking (BMC) for 3D NoCs. The proposed method implements BMC on the extended finite state machine of the 3D NoC and extracts leading sequences to trigger internal functions. Next, it detects structural defects in each function activated by the leader sequence using constrained ATPG. Finally, it effectively organizes packets to test the 3D NoC and reduce test cost. Experimental results demonstrate that the proposed method achieves high fault coverage online with extremely lower test cost and outperforms existing methods.

1. Introduction
With its high reliability and demonstrated spatial scalability, network-on-chip (NoC) has been widely used in manycore systems [1]. Compared with the 2D NoC, the 3D NoC achieves higher integration density, shorter interconnection distance, and lower overall power consumption [2][3]. Ensuring the error-free operations of the many-core system based on the 3D NoC remains a challenging issue.

Cota et al. proposed a test mechanism using NoC [4] as the test access to optimize the existing design-for-test, and thereby reduced test time [5]. Xiang et al. described a unicast-based multicast test method for NoCs to reduce test time under certain thermal constraints [6]. However, these methods are based on scan chains and require an external ATE. Therefore, they are not suitable for online testing. In addition, the 3D NoC realizes the connections of the two-dimensional dies by the through silicon via technology (TSV). Bringing TSV into the scan chain still faces some technical difficulties, and the above design for testability’s is also extremely high.

This paper proposes a BMC-based SBST method based on the work in [7] for 3D NoCs. The proposed method implements online testing for 3D NoCs and achieves high fault coverage with extremely low test cost.

This paper is organized as follows. Section II introduces the background knowledge of 3D-NoC and SBST. We describe the proposed method in detail to test buffers and switches in 3D NoCs in Section III. Section IV presents the experimental results. Section V summarizes this paper at the end.
2. Background

2.1. 3D NoC
The 3D NoC is realized by connecting 2D NoC with TSV. Figure 1 presents an example of the 3D NoC structure. It is mainly composed of processing units (PE), routers, network interfaces, and links. The buffers and switches in the 3D NoC are the major test objects.

![Figure 1. The mesh structure of a 3D NoC.](image)

2.2. Software-based Self-testing
Software-based self-testing emerges as a promising online test method [8-11]. SBST is also capable of online testing NoC [12]. When the SBST programs load or store data packets from or to a remote node, NoC transmits these packets from the source to the target through the on-chip network, and the transmitted packets can detect functional and structural faults in the NoC.

However, the faults in the control circuit of the on-chip network is still difficult to test as many functions of the control circuit require a complex input sequence to be activated. Bounded model checking (BMC) [13][14] emerges as a novel verification technology, which can generate test sequences and trigger a given function in the state machine. With the help of BMC, the SBST method is expected to achieve high fault coverage on the 3D NoC.

3. SBST using BMC for 3D NoCs

3.1. The Workflow of SBST Using BMC
First, the algorithm automatically extracts the extended finite-state machine (EFSM) from the design. Second, it proposes a property set to trigger all functions in the module under test. Third, it applies the abstract technology to reduce the size of the EFSM according to the contribution to a given property. Fourth, it starts to individually check the properties in the set. Specifically, it uses a BMC tool to check the negative proposition of the property. Once the result is wrong, the BMC tool will provide a counterexample. The counterexample indeed meets the property. The algorithm sets input signal sequence in the counterexample as the leading sequence. Fifth, it generates test patterns using ATPG under the constraints in the leading sequence. Finally, it maps the generated test patterns to the leading sequence and generates test program for the sequential circuit.

3.2. Testing the Buffer
The buffer takes charge of handshaking with neighboring nodes or PE. In a 3D NoC, the handshake state machine is the same to that in 2D. Therefore, for a single buffer, this work directly uses the same
method in the work [7] to generate test programs. However, a large number of buffers exist in a 3D NoC, and individually testing requires too much test data and test time. Fortunately, a group of testing packets in the 3D NoC can pass through multiple buffers, and then test them at the same time. This method still requires a suitable test scheduling method to reduce the amount of test data and test time.

Since a 3D NoC is made up of multiple layers of 2D NoCs, this method intends to adopt a divide-and-conquer strategy to schedule test packet groups and reduce test overhead. Specifically, the method divides test packet groups into two types: vertical and horizontal. We adopt the test packet group in the vertical direction to test the TSV-related buffers while use the test packet group in the horizontal direction to test the buffers in each layer of these 2D NoCs.

Figure 2 presents the examples of scheduling test packet groups in a $2 \times 2 \times 2$ mesh 3D NoC and testing its buffers at the same time. In the vertical direction, the test packet group (purple line) can test multiple TSV-related buffers. For example, the packet group generated by the SBST program is sent from the upper port of router R0 to the lower port of router R4 in figure 2(a). The test packet group passes through two buffers, and then detects structural faults inside these two buffers. In the horizontal direction, the test packet groups generated by the SBST programs can be divided into three types. The first type (green line) of packet groups can test multiple buffers sequentially. For example, the SBST program generates a packet group and sent it from the north port of router R0 to the local port of router R3 in figure 2(b). In addition, the SBST program will generate the second (blue line) and the third (yellow line) types of packet groups. These two types of packet groups can appear at the same time without causing routing conflicts and they can significantly reduce test time.

3.3. Testing the Switch

In a 3D NoC, the switch takes charge of arbitrating input requests and forwarding data packets. First, the switch is responsible for setting priorities for input requests and processing these requests based on their priorities. In figure 3(a), the switch in this work adopts a round arbitration algorithm to determine
the priority of each input request. According to the priority, the switch continues the arbitration process until it selects a valid input port or finds that none of input request exists at any input port.

Next, the switch applies the XYZ routing algorithm to send packets to the output port in figure 3(b). For example, if the X and Y coordinates of the packets’ destination address are equal to those of the current router, but their Z coordinates are not equal, the router will try to forward these packets through the upper (or lower) port according to their Z coordinate. If the corresponding port is idle, the switch will have a state transition in its state machine to complete the packet transmission.

Testing the switch in a 3D NoC requires coordination of the SBST programs in PE. For example, when we test the state transition "S3→S7", the switch needs to select an appropriate input port. If the switch has allocated a channel for the request from input port 2, and it will process the request from input port 1 in the next arbitration, the test has to prohibit SBST programs on adjacent PE requesting the input port 3, 4, 5, 6, and 0 of the current router. Then the BMC tool loads in the property "CS=S7 & lx=tx & ly=ty & lz!=tz & Free [dx]=1", and obtains the leading sequence that activates the property. Later, the method adopts ATPG on the time-expansion to generate test patterns. Finally, it maps these test patterns into test packets and coordinates SBST programs to send packets correctly.

When SBST tests a switch, it needs the switch to process seven input ports’ requests at least once with its adjacent routers’ assistance. Furthermore, SBST requires test packets from different ports arriving at the same time and stimulating arbitration functions. In this way, SBST needs to calculate transmitting time of each packet group. In figure 4, the southern and eastern input ports of router R0 need the assistance of router R2 and R1, and thus their packet group can arrival with other groups at the same time. Assuming that test packets of R0’s western port need $t$ cycles to arrive the router. The R2’s and R1’s groups also have to arrive in $t$ cycles.

![Figure 4. SBST for 3D-NoC Switch.](image)

### Table 1. BMC results for buffers and switches

|            | No. of properties | CPU time |
|------------|-------------------|----------|
| Buffer     | 25                | 1.19s    |
| Switch     | 58                | 2.34s    |

Finally, we analyze the state machines in the buffer and switch, and develop properties to cover state transitions to be tested. Later, we use the BMC tool (i.e., NuSMV) to check if the property’s negative proposition is valid. Once it is not valid, NuSMV produces a counterexample to the negative proposition, which is the required leading sequence. In table 1, the property numbers of the buffer and the switch are 25 and 58, respectively. Then NuSMV just takes 1.19s and 2.34s to generate their leading sequences.

### 4. Experimental Results

We use the SBST using BMC (BSBST) to generate test packet groups and test the $2\times2\times2$ mesh 3D NoC, where each packet’s width is 16 bits. Then, we use the commercial tool Tetramax to evaluate the fault coverage achieved by the packet groups. Finally, we use the test method using full scan chain (FScan) and the test method using random packets (random test) as the reference.
4.1. Fault Coverage
As shown in table 2, BSBST effectively tests the 3D NoC in functional mode and achieves an extremely high fault coverage. The fault coverage of BSBST is very close to that of FScan. It is worth emphasizing that BSBST is very effective in testing buffers in the 3D NoC. Although FScan has achieved higher fault coverage, it cannot test 3D NoCs online.

Figure 5 describes the relationship between the fault coverage and the test data amount of the test method using random test packets, where the number ‘N’ in “NX” represents N times of the BSBST’s test packets. In figure 5, when test packets of random test increases to 16 times of those of BSBST, the growth of the fault coverage tends to converge. However, the fault coverage of random test still far lags behind that of BSBST. That in turn demonstrates that BSBST can effectively detect faults in 3D NoCs.

Table 2. Fault coverage of different methods.

| Method   | Buffer | Switch | NoC   |
|----------|--------|--------|-------|
| FScan    | --     | --     | 98.99 |
| 1X random packets | 61.17 | 61.63 | 61.19 |
| 2X random packets | 71.04 | 68.27 | 70.87 |
| 4X random packets | 77.86 | 70.11 | 77.40 |
| 8X random packets | 81.47 | 77.92 | 81.27 |
| 16X random packets | 83.89 | 79.08 | 83.64 |
| BSBST    | 98.99 | 88.56 | 98.38 |

Table 3. Test cost of FScan and BSBST.

|                        | FScan     | BSBST |
|------------------------|-----------|-------|
| Area overhead(%)       | 7.68      | 0     |
| ATE required           | Yes       | No    |
| Test generation time(s)| 3.44      | 3.53  |
| Test application time(cycles) | 22667328 | 127100 |
| Test data volume(KB)   | 7368      | 18.25 |

Figure 5. Fault coverage increasing with the growth of the random test’s packet amount.

4.2. Test Cost
Another reason for using BSBST to online test the 3D NoC is its low test cost. Because BSBST tests the 3D NoC by transmitting test packets among NoC nodes, as shown in table 3, it neither requires any intrusive hardware nor brings in any extra area overhead. Besides, BSBST does not require expensive ATE, thereby reducing test cost. On the contrary, FScan requires extra area of design-for-test, which can be as high as 7.68%. It also requires ATE, and the cost of testing is high.

Compared with FScan, BSBST requires much shorter test generation time and test application time. The main step of its test generation process is to execute BMC. Because BSBST reduces the size of the original design, its CPU time is very small and can be ignored. Besides, BSBST can test multiple buffers at the same time, hence it effectively reduces the test application time. Finally, the BSBST’s packet amount is reasonable. BSBST just tests faults activated during internal state transitions, that is,
functional testable faults. Compared with FScan executing in non-functional mode, BSBST avoids the over-testing problem. Therefore, the experimental results demonstrate that BSBST can effectively and efficiently test the 3D NoC in functional mode.

5. Conclusions
This paper proposes an effective SBST method using BMC, which tests the 3D NoC by transmitting test packets among NoC nodes. This method executes BMC on the EFSM model to generate a leading sequence that activates the specific function. Then, it uses the constrained ATPG to generate test patterns and maps these patterns into test packet groups. Finally, the method schedules test packet groups in the 3D NoC, and reduces the test time and test data volume. Experimental results demonstrate that BSBST achieves extremely high fault coverage in functional mode, which is comparable to manufacturing test.

Acknowledgments
This paper is partly supported by National Natural Science Foundation of China (NSFC) under grant No. (61432017, 61974105), and partly supported by Open Research Projects of Zhejiang Lab.

References
[1] L. Zhang, Y. Han, Q. Xu, X. Li, H. Li, “On topology reconfiguration for defect-tolerant NoC-based homogeneous manycore systems”, IEEE Trans. on VLSI Systems, vol. 17, no. 9, pp. 1173-1186.
[2] Y. Zhang, X. Hong, Z. Chen, Z. Peng, J. Jiang, "A deterministic-path routing algorithm for tolerating many faults on very-large-scale network-on-chip", ACM Trans. DAES, vol. 26, no. 1, 2020.
[3] N.K. Meena, H.K. Kapoor, S. Chakraborty, “A new recursive partitioning multicast routing algorithm for 3d network-on-chip”, VLSI-DAT’2014, Coimbatore. pp. 1-6.
[4] E. Cota, M. Kreutz, A. Zeferino, “The impact of NoC reuse on the testing of core-based system”, VTS’2003. Napa. pp. 128-133.
[5] E. Cota, C. Liu, “Constraint-driven test scheduling for NoC-based systems”, IEEE Trans. on CAD of Integrated Circuits and Systems, vol.25, no. 11, 2006, pp. 2465-2478.
[6] D. Xiang, K. Chakrabarty, H. Fujiwara, “Multicast-based testing and thermal-aware test scheduling for 3D ICs with a stacked Network-on-Chip”, IEEE Trans. on Computers, vol. 65, no. 9, 2016, pp. 2767-2779.
[7] Y. Zhang, H. Chakrabarty, H. Li, J.H. Jiang, “Software-based online self-testing of network-on-chip using bounded model checking”, ITC’2017, Fort Worth. pp. 1-10.
[8] Y. Zhang, H. Li, X. Li, "Automatic test program generation using executing-trace-based constraint extraction for embedded processors", IEEE Trans. on VLSI systems, vol. 21, no. 7, 2013, pp. 1220-1233.
[9] Y. Zhang, H. Li, X. Li, "Software-based self-testing of processors using expanded instructions", ATS’2010, pp. 415-420.
[10] Y. Zhang, Z. Peng, J. Jiang, H. Li, M. Fujita, "Temperature-aware software-based self-testing for delay faults", DATE’2015, pp.423-428.
[11] Y. Zhang, K. Chakrabarty, Z. Peng, A. Rezine, H. Li, P. Eles, J. Jiang, "Software-Based Self-Testing Using Bounded Model Checking for Out-of-Order Superscalar Processors", IEEE Trans. on CAD of integrated circuits and systems, vol. 39, no. 3, 2019, pp.714-727.
[12] A. Dalironsi, M.E. Imhof, H. Wunderlich, “Structural software-based self-test of Network-on-Chip”, VTS’2014, Napa. pp. 1-6.
[13] S. Gurumurthy, S. Vasudevan, J. A. Abraham, “Automated mapping of pre-computed module-level test sequences to processor instructions”, ITC’2005, Austin. pp. 10 pp.-303.
[14] A. Riefert, R. Cantoro, M. Sauer, M.S. Reorda, B. Becker, “A Flexible Framework for the Automatic Generation of SBST Programs”, IEEE Trans. on VLSI Systems. vol. 24, no. 10, 2016, pp. 3055-3066.