A STRUCTURAL ANALYSIS OF ON-LINE FAULT DETECTION MECHANISMS IN NETWORK-ON-CHIP ARCHITECTURES

A Thesis

by

HAN BEE OH

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Duncan M. H. Walker
Co-Chair of Committee,	Gwan Choi
Committee Member,	Jeyavijayan Rajendran
Head of Department,	Miroslav M. Begovic

May 2020

Major Subject: Computer Engineering

ABSTRACT

Network-on-Chip (NoC) communication architectures are widely used as on-chip interconnect in multi-core systems. These systems are increasingly used in safety-critical applications, so it is essential to quickly detect faults within the NoC during system operation.

The current approach to detect a fault in an NoC system is to apply periodic test using built-in self-test (BIST) circuitry during system idle periods. This approach has the advantage that it is a structural test, so can quickly achieve high fault coverage. A second advantage is that the BIST infrastructure can be used during manufacturing test. The disadvantage is the need for the idle time to apply the test, and the time to save/restore the functional state that is overwritten during the test. An additional disadvantage is that the system is at risk of an undetected fault between self-tests.

In this research we propose to test the NoC system while it is in functional operation, which is an on-line test. We will use functional invariants to detect errors in functional operation, which can then trigger diagnosis and fault recovery or system reconfiguration. The advantage of this approach is that it minimizes fault detection latency, and avoids the need for a system idle period or for save/restore state operations. The disadvantage is that it is much more difficult to achieve high fault coverage since our approach detects functional errors based on existing functional network traffic, rather than self-test stimulus.

In order to evaluate our functional test approach, we have designed a gate-level NoC implementation, which can be the target for gate-level fault injection and simulation using realistic network traffic. We inject stuck-at faults and single event transients into the gate-level logic during simulation of synthetic NoC traffic. We found that the functional invariants proposed in prior work miss detection of many faults. Most of these escapes are detected by end-to-end cyclical redundancy checks. However, we found it necessary to create additional functional checkers to detect the remaining faults.

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Walker, my co-chair Dr. Choi, and my committee member Dr. Rajendran, for their guidance and support throughout the course of this research.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

Finally, thanks to my mother and father for their encouragement, patience and love.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis committee consisting of Professor Duncan M. Walker (advisor) of the Department of Computer Science and Engineering, and Professor Gwan Choi (co-advisor) and Professor J. V. Rajendran of the Department of Electrical and Computer Engineering.

The network-on-chip architecture and traffic patterns were taken from the open literature, as were the initial set of functional checkers. The open-source Booksim simulator was used for fault simulations.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

The student was self-funded during this research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Page

CHAPTER I

INTRODUCTION

Semiconductor technology scaling has allowed complete systems with hundreds of cores to be placed on a single chip [1]. Packet-based Network-on-Chip (NoC) have become the common communication mechanism for these designs [2][3][4][5]. There has been much academic research on NoC design and analysis [6], and commercial design automation tools are available [7][8].

Increasingly integrated circuits are used in safety-critical systems, most notably automotive applications [9]. Safety standards such as ISO 26262 [10] place very stringent reliability requirements on electronic systems to maintain system functional safety. This includes the NoC within the integrated circuits.

One of the traditional approaches to building high-reliability systems is to use redundancy (e.g. error detecting and correcting codes, repeated execution, duplicate modules) and reconfiguration to provide fault tolerance [11]. The drawback of these approaches is that their cost is too high for many applications, in terms of chip area, timing, or power. The approach being taken in many automotive systems is BIST, where testing can be interleaved with system operation. If a fault is detected, diagnosis and corrective action is taken, such as swapping in a spare module or operating in a degraded, but safe condition. This is now a very active area of research and commercial development, with its own dedicated workshop [12].

The drawback of periodic BIST is that it is very invasive to the design process. For example, when tests are applied to circuits that are not in functional operation, any current state is overwritten. If that state is needed for system operation, it must be saved and restored. This is very difficult when testing memories. A second drawback of periodic BIST is that the system is at risk during the period that it is in functional operation.

In the context of NoC, most faults in the packet payload can be detected using low-overhead error detecting codes, such as cyclical redundancy check (CRC) applied when the packet is received at the final destination. This end-to-end check has high fault coverage, but it can be difficult to localize the fault that occurred on the routing path, and faults in the buffering and control logic in the NoC routers can cause the packet to be damaged, lost or mis-delivered. The control logic can be protected with periodic BIST, but there are several challenges to implementing it in NoC architectures:

- The area overhead of the BIST engine is such that it must be shared across many NoC routers. This increases the time to deliver the test patterns and check the results. It also introduces extra routing overhead for the BIST information.
- The shared BIST engine means that a full NoC test cycle must be spread over many system clock cycles, due to the limited amount of time for periodic BIST, before the system must return to functional operation. This may result in an unacceptably long time window between complete tests.
- When a set of routers are in a BIST cycle, they block network traffic. Attempts have been made to determine network traffic patterns and schedule the BIST cycles during

idle times [13], but this is a complex process in systems with changing activity and traffic patterns.

As noted above, testing memories with periodic BIST is challenging due to the need to save and restore state, and the limited access path to and from the memory, and an alternate location to save the state. This is a particular problem in NoCs, due to the port buffers in every router. One option is to wait until the buffers are empty before executing the BIST cycle, but this will have a greater impact on network traffic. Packet-level CRC can detect some buffer faults, but rarely-used buffer entries (e.g. the last entry in the last virtual channel, only used when the buffer is full), may not be detected until the system is very busy, which may be during a system emergency.

Due to the shortcomings of periodic BIST to detect faults in NoC architectures, Prodromou, et al. [14] proposed that error detecting codes be used to check the data paths in the NoC, and that an on-line fault detection mechanism, termed NoCAlert, using functional invariance checkers, be used to detect faults in the router control logic and crossbar switch. As shown in Fig. 1, their invariance checkers compare the inputs and outputs of logic modules, and signal an error if any invariance is violated. This approach demonstrated high fault coverage with low fault detection latency and no false negatives (fault escapes). The checker designs had minimal power and area overhead.

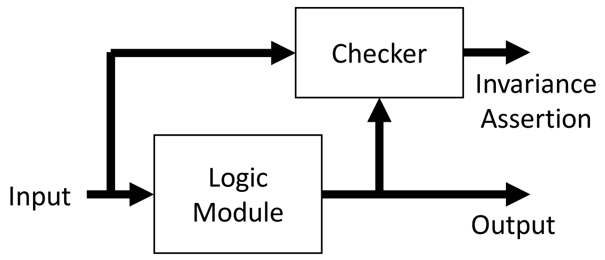


Figure 1. NoCAlert invariance checker.

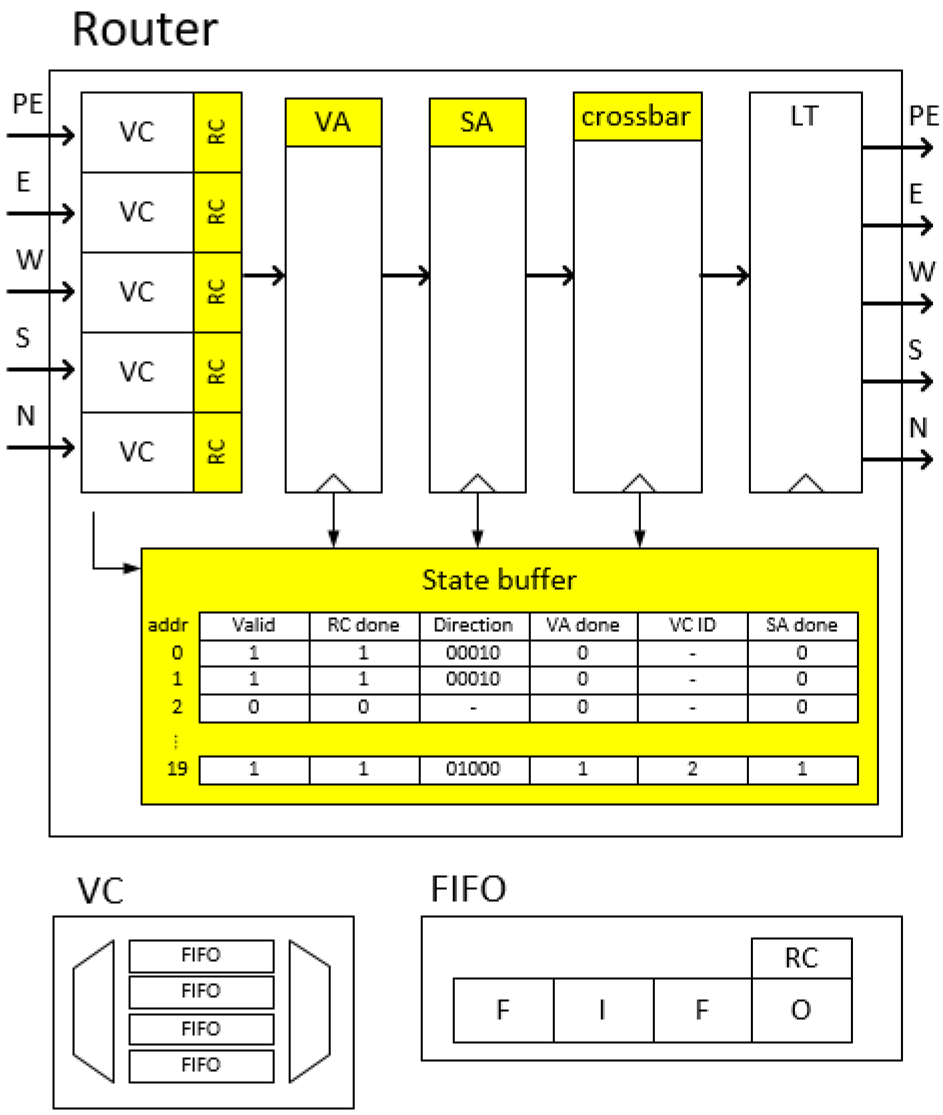
The analysis in [14] only considered single event transient (SET) faults on inputs and outputs of the behavioral modules covered by an invariance checker. This is a small fraction of the potential fault universe. Other research has examined detection of transistor short-channel faults in NoCs [15]. In this work, we evaluate the performance of module-level invariance checkers on gate-level faults within the routers of the NoC. This includes faults within the control logic, crossbar switch, and input and output ports. In addition, we include end-to-end CRC checking for each packet, and determine the coverage relationship between CRC checking and the invariance checkers. We show that invariance checkers have high coverage of gate-level faults within the control modules, and that CRC detects data path faults. The results show that some faults are not detected, and propose additional checkers to detect such faults.

CHAPTER II

NOC MICRO-ARCHITECTURE

In this work, we use a 2D mesh NoC architecture with wormhole switching, atomic Virtual Channel (VC) buffers, deterministic X-Y routing and credit-based flow control. Each packet is divided into 32-bit flits. The first two bits of each flit specifies the flit type. The head flit contains source and destination address (in terms of (x,y) location in the mesh), and packet ID (PID). A number of payload flits, which include the PID and data, follows the head flit. The tail flit contains the PID and a 24-bit CRC.

We will use a baseline router micro-architecture [16] that has been widely used in the literature, including [14]. Fig. 2 presents an abstract view of the router design. Flow control for the router uses a state buffer, which contains routing computation, virtual channel allocation, and switch allocation status. Each router has five input/output ports – one for each of the routing directions in a 2D mesh – and one port to communicate with the processing element at that location in the mesh. Each port has four virtual channels (VCs), each with four flit buffer slots. These VCs are used primarily to avoid deadlock in the network. A parallel crossbar switch connects the input and output ports together.



addr	Valid	RC done	Direction	VA done	VC ID	SA done
0	1	1	00010	0	-	0
1	1	1	00010	0	-	0
2	0	0	-	0	-	0
⋮						
19	1	1	01000	1	2	1

Figure 2. Overview of the router design.

The control logic consists of the routing computation (RC), virtual channel allocation (VA), and switch allocation (SA) modules. The RC module computes the output direction of an incoming packet, based on the routing algorithm and the destination address of the packet (in the head flit). The VA module allocates the VC that the packet will use in the next router. The SA module selects the flits switched by the crossbar in each clock cycle.

The router has a five-stage pipeline, consisting of RC, VA, SA, crossbar traversal, and link traversal. The RC and VA stages are only required for the header flit. Each VC only stores flits belonging to a single packet. The input and output control signals of each VC are shown in Fig. 3. The state buffer for the routing control is shown in Fig. 4. Since the CRC is generated at the originating processing element, and evaluated at the destination processing element, the CRC is treated as payload by the router. This work uses the 24-bit CRC from wideband code division multiple access (W-CDMA) [17]:

$$x^{24}+x^{23}+x^{6}+x^{5}+x^{1}+1$$

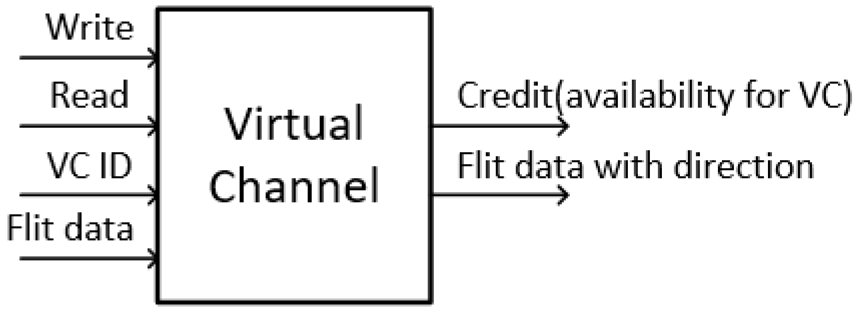
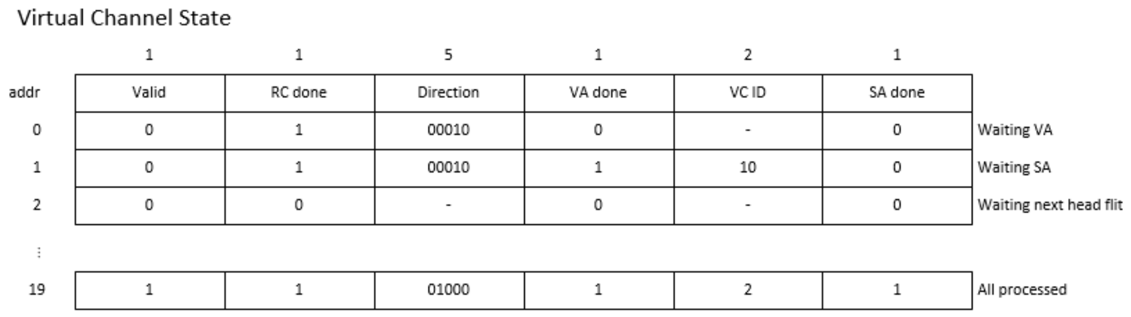


Figure 3. Virtual channel control signals.



Virtual Channel State

	1	1	5	1	2	1	
addr	Valid	RC done	Direction	VA done	VC ID	SA done	
0	0	1	00010	0	-	0	Waiting VA
1	0	1	00010	1	10	0	Waiting SA
2	0	0	-	0	-	0	Waiting next head flit
⋮							
19	1	1	01000	1	2	1	All processed

Figure 4. State buffer for routing control.

To avoid deadlock, the input port of the router contains a virtual channel module with four virtual channels. Each virtual channel contains a FIFO implemented as a circular buffer. If the head flit is at the head of the FIFO, the routing computation is done to calculate which direction the packet needs to go, as shown in Fig. 5.

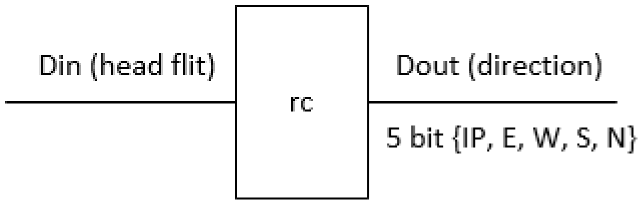


Figure 5. Routing computation.

When the routing computation is done, the virtual allocator calculates the virtual channel location in the next router (virtual channel ID or VCID). If all of the virtual channels are full, the virtual allocator deasserts the valid signal, so that the switch allocator enters an idle cycle, as shown in Fig. 6.

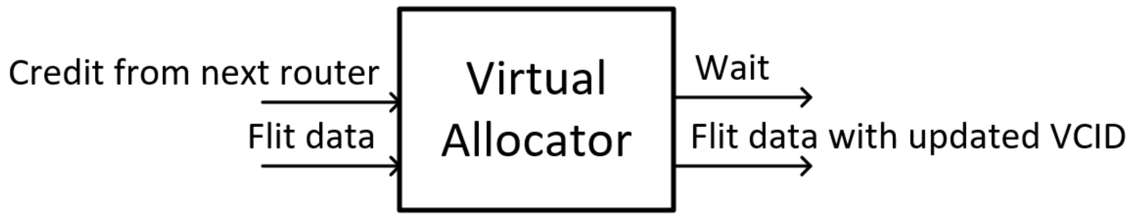


Figure 6. Virtual allocator.

The switch allocator calculates whether the flit under consideration can be routed through crossbar in the next cycle, as shown in Fig. 7. Routing computation, virtual allocator, and switch allocator is done only on head flits, so there is the possibility that a body or tail flit is being transmitted to the same output port requested by the head flit. The switch allocator does not grant the request until the output port is free.

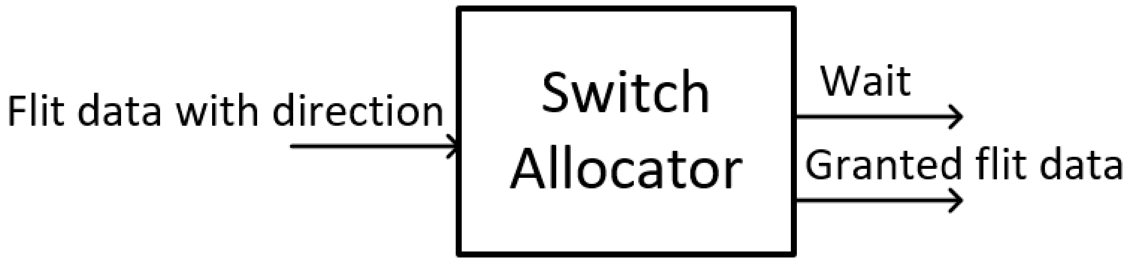


Figure 7. Switch allocator.

When the head flit completes the switch allocator process, all flits in the packet pass through the crossbar without pausing. The crossbar switch is implemented using multiplexers as shown in Fig. 8.

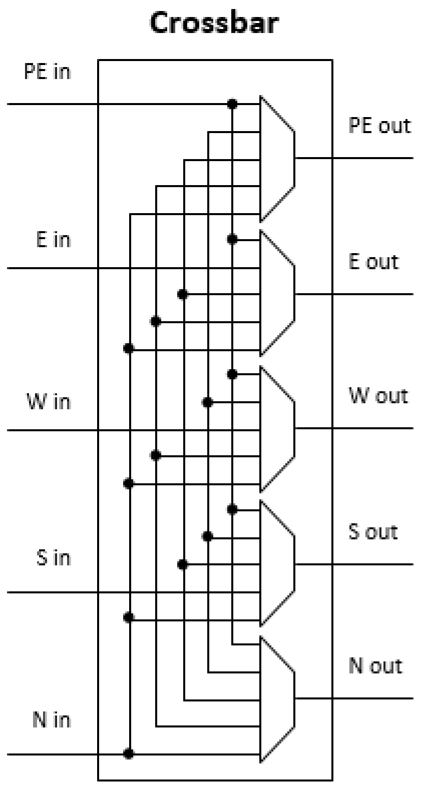


Figure 8. Multiplexer-based crossbar switch.

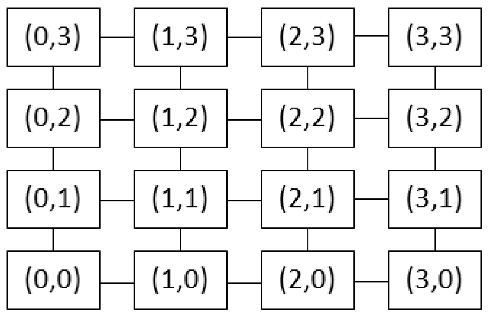


Figure 9. NoC mesh.

For our simulations, we use a 4x4 mesh NoC system, as shown in Fig. 9. Each router communicates with its neighbor routers by transmitting flit data and credit data. Flit data is transmitted using the message format in Fig. 10. A message is made up of multiple packets. A packet is made up of three types of flits (head flit, body flit, and tail flit). The head flit contains destination, source location, packet size and virtual channel ID. The body flit contains the data being transmitted from one processing element to another. The tail flit contains the CRC-24 code.

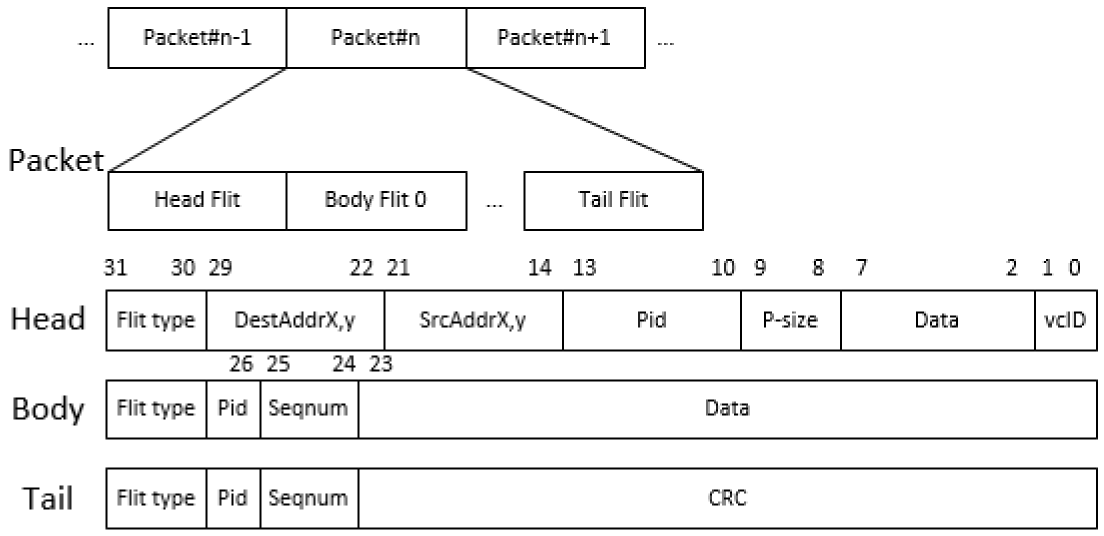


Figure 10. Message format.

Table 1. NoCAlert invariants.

Routing Computation (RC) Unit	
1	Illegal turn
2	Invalid RC output direction
3	Non-minimal routing (if required)
Arbiter Modules (VA and SA Stages)	
4	Grant w/o request
5	Grant to nobody
6	1-hot grant vector
7	Grant to occupied or full VC
8	One-to-One VC assignment
9	One-to-One port assignment
10	VA agrees with RC
11	SA agrees with RC
12	Intra-VA stage order
13	Intra-SA stage order
Crossbar	
14	1-hot column control vector
15	1-hot row control vector
16	# Incoming flips == # Outgoing flits
Buffer State (Each VC Buffer maintains its state)	
17	Consistent VC buffer state
18	Only header flits in free VC buffers
19	Invalid output VC value
20	Complete RC stage on a non-header flit
21	Complete RC stage on an empty VC
22	Complete VC stage on a non-header flit
23	Complete VA stage on an empty VC
24	Read from an empty buffer
25	Write to a full buffer
26	Buffer atomicity violation
27	Packet mixing in non-atomic buffer
28	Packet flit-count violation
Port-Level Invariances	
29	Concurrent read from multiple VCs
30	Concurrent write to multiple VCs
31	Concurrent RC completion of multiple VCs
Network-Level Invariance	
32	End-to-End delivery violation

Prodromou et al. [14] developed a set of 32 invariants covering the routing computation, arbiter modules (virtual channel allocation and switch allocation), crossbar switch, virtual channel buffer state, input/output ports, and end-to-end delivery. These invariants are listed in Table 1. Fourteen invariants (5, 8, 12, 13, 14, 15, 18, 19, 22, 26, 28, 29, 30, 31) are not considered here since these errors cannot occur in our gate-level design. In this design, the virtual channel ID is binary-encoded, not one-hot encoded, so checkers 8, 19, 29, 30, 31 are not needed in our design. Our crossbar switch design uses a multiplexer approach, rather than a switch matrix so checkers 14 and 15 are unnecessary. There is no intra/inter VA/SA separation in the design so checkers 12 and 13 are not implemented. There is possibility that VA does not give any grant when there is not available virtual channel for next hop, so 5 is not proper checker for our design. Also, the flit size needs to be flexible so checker 28 is not implemented in our design. In our design, there is no unique signal when the virtual allocator calculates virtual channel ID, so checker 22 is not implemented. There is some chance that a body flit can become stuck in a virtual channel buffer, so 18 is not proper checker for our design. Since our virtual channel can contain 4 flits (which is more than one packet), the head flit can become stuck in a non-free virtual channel, which means checker 26 is unnecessary.

As noted in [14], an invariance checker detects *illegal* outputs, but may not detect *incorrect*, but legal outputs. A later router or the destination (e.g. CRC calculation) may catch some of these incorrect outputs. The checkers may not catch some faults. For example, a bit flip in the packet destination address may deliver the packet to the wrong destination without any invariance violation. The destination processing element will presumably use higher level routing protocols, such as sequence numbers, to reject such packets, and depend on a timeout and retransmit by the source processing element. Such higher-level routing protocols are beyond the scope of this thesis.

CHAPTER III

FAULT MODELING

In this research we consider single stuck-at faults (SAF) and single-event transients (SET), which can be viewed as a stuck-at fault for a single clock cycle. We cannot use traditional structural fault simulation in this research, since we will be combining behavioral NoC system simulation with gate-level simulation of the router. In our approach we modify the gate model so that we can inject stuck-at-0 (SA0) and stuck-at-1 (SA1) faults into the inputs and outputs of each logic gate. Fig. 11 shows an example of a logical AND gate with additional circuitry to permit fault injection during the simulation. When the ST0 signal is zero on the output, the AND gate output is SA0. When the ST1 signal is zero, the output is SA1. Similarly for the ST0 and ST1 signals on the inputs. For fault-free operation, all ST0 and ST1 signals are one. To inject a fault, one of the ST0 or ST1 signals is set to zero. For *N* logic gates in the circuit, there will be *6N* control signals. For a single-fault model, only one signal is set zero at a time. For a stuck-at fault model, the signal is set zero for all clock cycles in the simulation. For a single-event transient, the signal is set zero for one clock cycle.

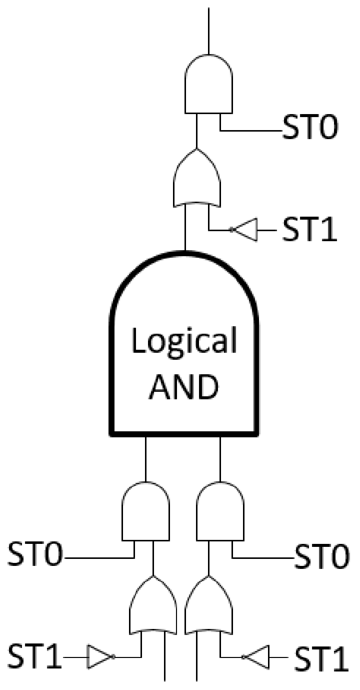


Figure 11. AND gate with fault insertion logic.

CHAPTER IV

EXPERIMENTAL SETUP AND RESULTS

The original NoCAlert analysis [14] performed cycle-accurate simulation of an 8x8 mesh NoC using the GARNET NoC simulator [18], modeling the routers at the micro-architectural level, with the addition of the invariance checkers. Single-event transient (SET) faults were injected at the inputs and outputs of the RC, VA, SA and crossbar switch modules. There were 205 module-level SET fault injection sites within each router, and a total of 11,808 fault sites within the 8x8 2D mesh. The mesh was simulated with synthetic traffic patterns with traffic injection rates varying from 0.1 to 0.4 flits/PE/cycle. SET fault injection was performed at cycle 0, 32K and 64K, for a total of 248K fault injection simulations.

To perform a gate-level analysis of the network, we replaced router (2,1) with a gate-level implementation, which was simulated using NCverilog. The NoC level behavioral simulation was performed using BookSim 2.0 with uniform traffic. This NoC level traffic is fed into the gate-level simulation.

In our simulations, we randomly injected gate-level stuck-at faults, by setting one of the gate injection control lines in Fig. 11 to zero. If the fault is an SET, it is set for only a single clock cycle, either specified or randomly chosen. Only one fault is injected in each simulation run. Faults are not injected into the invariance checkers, since our goal is to understand their ability to detect gate-level faults within the router. As a practical matter, checkers are unlikely to be faulty since they are only about 3% of the router area [14].

In our implementation, all parts of the router except the virtual channel buffer were implemented at gate level. There are 38,100 stuck-at fault sites (RC/VA/SA/Crossbar/ status buffer) in each router. For the fault simulations, we randomly injected 3800 faults (10% of the fault sites). The simulation was performed for 500 clock cycles, which is long compared to the 36 cycles for a packet to transit completely across the mesh. The injection rate for the uniform traffic pattern is 0.05 (0.05 packets/cycle injected by each processing element, addressed to a different random processing element). Experiments were done using both stuck-at faults and single-event transient faults.

In the stuck-at fault experiment, we examine the faults detected by the invariance checkers and the CRC. Faults labeled "masked" do not result in any faulty behavior, or one bit flipped in an all-0 idle flit. Single stuck-at faults are injected into random fault sites and are active across all clock cycles. The results are shown in Table 2.

Table 2. Single stuck-at faults.

	NoCAlert detected		NoCAlert not detected	
	CRC detected	CRC not detected	CRC detected	CRC not detected
Faulty	294	581	535	188
Masked	0	164	0	2028

The faults that are only detected by CRC are faults that only affect the packet payload or CRC.

In the 188 cases that are not detected by the checkers or CRC, 159 cases are benign – virtual channel ID (VCID) is changed, so the packet uses a different virtual channel than intended. The VCID is not checked by the CRC, since it changes with packet routing. The VCID is not checked by the invariance checkers, if a virtual channel is available.

There are 29 cases that escape detection, but do not cause a benign fault. There are four different categories: 1. Packet ID is duplicated, 2. Flit type is changed, 3. Bit flipped in Head/Body/Tail flit, and 4. Flit disappears.

For case 1, the packet ID is changed, and the changed packet ID is already used for another virtual channel, causing the body flit and tail flit to be sent to the wrong virtual channel. Such packets will ultimately be detected as truncated packets due to a receiver timeout, and a higher-level protocol would then retransmit them.

For case 2, where the flit type is changed, the destination processing element has to recognize the erroneous flit by using sequential numbers.

For case 3, where a bit is flipped in the Head/Body/Tail flit, normally they would be detected by the CRC checker, but these cases are similar to case 1, in that the flits cannot be assembled correctly for the CRC checker, and are eventually discarded. The reason is that with a packet insertion rate is 0.05 packets/cycle/PE for 16 PEs, and 500 simulation cycles, there are approximately 400 packets in the simulation. This would require a 9-bit packet ID field to uniquely identify them. But the packet ID field is only 4 bits, since packets only need to be uniquely identified on their routes, not in the entire array. But this limited packet ID field means that bit flips can cause failure in reassembling flits at the destination.

For case 4, a stuck-at fault can cause a flit to be lost. There are four causes of flit disappearance: 1. virtual allocator does not generate output even if there is no wait signal

(9 cases), 2. switch allocator does not generate output even if there is no wait signal (9 cases), 3. the flit data is changed in the virtual allocator (3 cases), and 4. flit data is changed in the switch allocator (1 case). We created a small invariance checker to detect these cases.

We repeated the stuck-at fault simulation with our new checkers (NC) inserted. The results are shown in Table 3.

Table 3. Single stuck-at fault with new checkers.

	NoCAlert detected				NoCAlert not detected			
	CRC detected		CRC not detected		CRC detected		CRC not detected	
	NC detected	NC not detected	NC detected	NC not detected	NC detected	NC not detected	NC detected	NC not detected
Faulty	103	191	145	436	136	399	63	135
Masked	0	0	0	164	0	0	1	2027

In the 135 cases that are not detected with the checkers or CRC, there are 120 cases of benign faults (virtual channel ID is changed) and 14 cases can be detected by higher level protocol, such as timeout (packet ID changed -11 cases, flit type is changed – 1 case). The remaining 3 cases are loss of flits due to insufficient number of bits in the packet ID (bit flipped in Head/Body/Tail flit), which must be detected by packet timeout.

For SET faults, three different experiments were performed. In the first experiment, the SET occurs in cycle 250 at random fault sites. In the second experiment, the fault site and fault cycle are randomly selected. In the third experiment, three SETs are randomly injected into fault sites on random cycles. The results for SET injection at cycle are shown in Table 4.

Table 4. SET injected at cycle 250.

	NoCAlert detected				NoCAlert not detected			
	CRC detected		CRC not detected		CRC detected		CRC not detected	
	NC detected	NC not detected	NC detected	NC not detected	NC detected	NC not detected	NC detected	NC not detected
Faulty	0	0	1	16	0	0	13	75
Masked	0	0	1	0	0	0	0	3582

As can be seen in the result, most SETs do not cause faulty behavior since their incorrect value is not propagated, either due to logical or temporal masking, or the circuit is idle.

In the 75 cases that are not detected with the checkers or CRC, there are 32 cases of benign faults (virtual channel ID is changed - 29 cases, bit flipped in all-0 idle flits - 3 cases). There are 9 cases that can be detected by a timeout (packet ID changed - 6 cases, flit type is changed - 3 cases), and 37 cases (bit flipped for Head/Body/Tail flit) where flits are lost (and must be detected by timeout) but would have been detected by CRC if more packet ID bits were available.

In the second SET experiment, SETs occur randomly in fault sites and clock cycles. The results are shown in Table 5.

Table 5. Result with one SET in different cycle.

	NoCAlert detected				NoCAlert not detected			
	CRC detected		CRC not detected		CRC detected		CRC not detected	
	NC detected	NC not detected	NC detected	NC not detected	NC detected	NC not detected	NC detected	NC not detected
Faulty	0	0	8	17	1	0	24	51
Masked	0	0	1	1	0	0	100	3598

In this experiment, there are 51 cases that are not detected by the CRC or the checkers. In 31 cases, the SET caused a benign change in the virtual channel ID. There are three cases where the packet ID is changed. In the remaining 17 cases, the SET causes bit flit in Head/Body/Tail flit, requiring timeout for detection.

In the third SET experiment, three SETs are injected into random fault sites on random cycles. The results are shown in Table 6.

Table 6. Result with three SET faults in random cycle.

	NoCAlert detected				NoCAlert not detected			
	CRC detected		CRC not detected		CRC detected		CRC not detected	
	NC detected	NC not detected	NC detected	NC not detected	NC detected	NC not detected	NC detected	NC not detected
Faulty	0	0	19	59	1	0	65	157
Masked	0	0	5	3	0	0	292	3200

In the 157 cases that are not detected by the checkers or CRC, 98 cases are benign faults (virtual channel ID is changed - 97 cases, flit is delayed in one cycle - 1 case). There are 4 cases where the packet ID is changed and 55 cases are bit flip in Head/Body/Tail flit, which must be detected by higher level protocols.

We evaluated a higher 0.1 packets/cycle/node packet injection rate for random traffic. This corresponds to an average of a 30% utilization of each PE port and a 7.5% utilization of the remaining ports. This utilization is high enough to sometimes cause virtual channel buffers to be full. The virtual allocator uses credit-based flow control to determine which virtual channel to assign to a packet in the next input port, and uses a round-robin approach to selecting the next available virtual channel. However, the allocation decision is only made based on the head flit, and then all remaining packet flits are sent, without any request/acknowledge signaling. As a result, flits will be dropped when there was enough space to send the head flit, but some of the remaining packet flits will not fit into the virtual channel. These truncated packets must be detected by the destination. Our checkers will flag these truncated packets, even though there was no fault. These can be viewed the same as cases where benign faults are flagged.

CHAPTER V

CONCLUSIONS

In this thesis, we performed a structural analysis of the invariant checkers and CRC proposed in [14]. These checkers had previously only been evaluated for a small set of the possible faulty behaviors that can occur in a router. We evaluated these checkers using single stuck-at faults and single-event transients injected into the gate-level circuit structure. We found that the CRC and invariant checkers had high coverage of the structural faults, but a significant number of faults still escaped. Most of these faults were benign, causing different timing or different use of virtual channels, but not affecting packet delivery.

In a small number of cases, undetected faults would cause packets to be damaged, so that they would be rejected by the destination processing element. The most serious case were

In the remaining cases we proposed additional on-line checkers to detect the faults that had escaped. We propose on-line test checkers which is missed from NoCAlert checker and CRC checker. In the stuck-at fault simulation, 63 more faults are detected by adding three checkers, increasing the number of detected faults from 3602 to 3665 (1.66% increase), out of the 3800 faults injected.

REFERENCES

[1] International Technology Roadmap for Semiconductors, 2017. [Online] Available: http://www.itrs2.net. [Accessed: 15-Sep-2018].

[2] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *Design Automation Conference*, Las Vegas, NV, 2001, pp. 684-689. doi: 10.1109/DAC.2001.156225

[3] L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm," *IEEE Computer*, vol. 35, no. 5, pp. 70-78, 2002.

[4] G. De Micheli and L. Benini, "Networks on Chips: 15 Years Later," *IEEE Computer*, vol. 50, no. 5, pp. 10-11, 2017.

[5] M. S. Gaur, V. Laxmi, M. Zwolinski, M. Kumar, N. Gupta and Ashish, "Network-on-Chip: Current Issues and Challenges," *IEEE International Symposium on VLSI Design and Test*, Ahmedabad, India, June 2015, pp. 1-3. doi: 10.1109/ISVDAT.2015.7208160

[6] S. K. Mandal, N. Gupta, A. Mandal, J. Malave, J. Lee and R. N. Mahapatra, "NoCBench: A Benchmarking Platform for Network on Chip," *International Workshop on Unique Chips and Systems*, Boston, MA, April 2009.

[7] Netspeed Systems Orion, 2018. [Online] Available: https://netspeedsystems.com/products/orion. [Accessed: 15-Sep-2018].

[8] J.-J. Lecler and G. Baillieu, "Application Driven Network-on-Chip Architecture Exploration and Refinement for a Complex SoC," *Springer Design Automation*

for Embedded Systems, vol. 15, pp. 133-158, 2011. Doi: 10.1007/s10617-011-9075-5

[9] K. Greb and R. Mariani, "Functional Safety Poses Challenges for Semiconductor Design," May 2011. [Online] Available: https://www.embedded.com. [Accessed: 15-Sep-2018].

[10] Road vehicles — Functional safety, ISO 26262-1:2018, International Standardization Organization, 2018.

[11] I. Koren and C. M. Krishna, Fault Tolerant Systems, San Francisco, CA: Morgan Kaufmann, 2007.

[12] *IEEE International Workshop on Automotive Reliability and Test*, Phoenix, AZ, November 2018.

[13] J. Liu, J. Harkin, Y. Li and L. Maguire, "Online Traffic-Aware Fault Detection for Networks-on-Chip", *Journal of Parallel and Distributed Computing*, vol. 74, no. 1, January 2014, pp. 1984-1993.

[14] A. Prodromou, A. Panteli, C. Nicopoulos, and Y. Sazeides, "NoCAlert: An On-Line and Real-Time Fault Detection Mechanism for Network-on-Chip Architectures," *IEEE/ACM International Symposium on Microarchitecture*, Vancouver, BC, 2012, pp. 60-71. doi: 10.1109/MICRO.2012.15

[15] B. Bhowmik, S. Biswas, J. K. Deka and B. B. Bhattacharya, "Reliability-Aware Test Methodology for Detecting Short-Channel Faults in On-Chip Networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 6, June 2018, pp. 1026-1039. doi: 10.1109/TVLSI.2018.2803478

[16] L. Peh and W. J. Dally, "A Delay Model for Router Microarchitectures," *IEEE Micro*, vol. 21, no. , pp. 26-34, 2001. doi:10.1109/40.903059

[17] L. B. Milstein, “Wideband Code Division Multiple Access,” *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 8, pp. 1344-1354, Aug. 2000. doi: 10.1109/49.864000

[18] N. Agarwal, T. Krishna, L. Peh and N. K. Jha, “GARNET: A Detailed On-Chip Network Model Inside a Full-System Simulator,” *IEEE International Symposium on Performance Analysis of Systems and Software*, Boston, MA, 2009, pp. 33-42. doi: 10.1109/ISPASS.2009.4919636J