Area-Efficient Synthesis of Fault-Secure NoC Switches

Dalirsani, Atefe; Kochte, Michael A.; Wunderlich, Hans-Joachim

Proceedings of the 20th IEEE International On-Line Testing Symposium (IOLTS'14) Platja d'Aro, Catalunya, Spain, 7-9 July 2014

doi: http://dx.doi.org/10.1109/IOLTS.2014.6873662

Abstract: This paper introduces a hybrid method to synthesize area-efficient fault-secure NoC switches to detect all errors resulting from any single-point combinational or transition fault in switches and interconnect links. Firstly, the structural faults that are always detectable by data encoding at flit-level are identified. Next, the fault-secure structure is constructed with minimized area such that errors caused by the remaining faults are detected under any given input vector. The experimental evaluation shows significant area savings compared to conventional fault-secure schemes. In addition, the resulting structure can be reused for test compaction. This reduces the amount of test response data and test time without loss of fault coverage or diagnostic resolution.

Preprint

Area-Efficient Synthesis of Fault-Secure NoC Switches

Atefe Dalirsani, Michael A. Kochte, Hans-Joachim Wunderlich
Institut für Technische Informatik, Universität Stuttgart, Germany
email: {dalirsani,kochte}@iti.uni-stuttgart.de, wu@informatik.uni-stuttgart.de

***Abstract*—This paper introduces a hybrid method to synthesize area-efficient fault-secure NoC switches to detect all errors resulting from any single-point combinational or transition fault in switches and interconnect links. Firstly, the structural faults that are always detectable by data encoding at flit-level are identified. Next, the fault-secure structure is constructed with minimized area such that errors caused by the remaining faults are detected under any given input vector.**

The experimental evaluation shows significant area savings compared to conventional fault-secure schemes. In addition, the resulting structure can be reused for test compaction. This reduces the amount of test response data and test time without loss of fault coverage or diagnostic resolution.

***Index Terms*—Network-on-Chip, self-checking, fault-secure, online testing, concurrent error detection.**

I. Introduction

A Network-on-Chip (NoC) is a communication alternative for many-core System-on-Chips. In current deep sub-micron technologies, latent defects, wear-out, soft errors, cross-talk, power supply noise and radiation effects affect the reliability of the system [1], [2] as well as the NoC structure. Having a reliable communication through the NoC is essential because an erroneous message transfer over the NoC may lead to an erroneous data delivery to cores and consequently wrong system operation. Furthermore, it may generate spurious traffic leading to a deadlock for example. Therefore, it is crucial to concurrently detect a fault as soon as it causes an erroneous operation of NoC switches or interconnect links. This way fault tolerance mechanisms can be activated immediately to avoid error propgation in the system.

Concurrent error detection (CED) techniques are used in safety-critical applications to detect errors caused by permanent and transient faults during the operation of the circuit [3]. Conventionally, duplication with comparison has been used to detect any single- and multi-bit error. In order to decrease duplication overhead, Error Detecting Codes (EDC) are employed. Fig. 1 presents the general structure for CED using EDC. Check bits over output bits of the *circuit* are computed in the *checker*. The checker compares the check bits to those generated by the *prediction logic* and signals an error in case of a mismatch. A circuit is fault-secure for a set of faults if for every considered fault that produces an error at the circuit outputs, wrong check bits are generated [4]. Moreover, a circuit is self-testing, if for every considered fault, there exists at least one input vector for which the error at the outputs produces wrong check bits. Finally, a circuit is Totally Self-Checking (TSC) if it is self-testing and fault-secure.

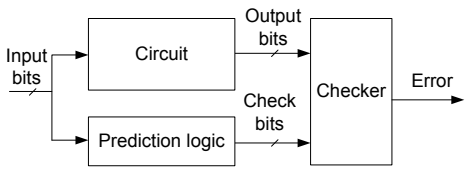

Fig. 1: General structure of Concurrent Error Detection (CED)

To conduct CED in NoCs, hardware redundancy schemes (such as duplication or triplication) have been examined [5]–[7]. Since the NoC may integrate hundreds of switches in a single chip, these methods impose a huge area overhead. Error detecting codes can also be exploited to synthesize a fault-secure NoC switch. However, considering the switch without regard to its datapath elements also leads to an unacceptable area cost.

In NoCs, data encoding using error detecting codes in combination with data retransmission is typically used to detect and correct errors during the system operation [6], [8]–[13]. Check bits are computed and appended to data bits which are transported over the network. Several researchers have investigated error detecting/correcting codes in terms of silicon area, encoder/decoder delay, performance, and energy consumption to trade-off costs and reliability [11]–[13]. Up to now, these methods are mainly devoted to detect faults in the inter-switch links and intra-switch datapath elements such as multiplexers.

Faults in the datapath elements mainly cause data corruption which may be detected and corrected by an error correcting code [14] even at the system-level. However, a part of the switch logic is dedicated to manage the flow of data, for example the routing algorithm, scheduling, and congestion control. Faults in this part can be even more severe and have uncorrectable effects leading to a system crash. For example, a fault in the routing logic may cause a misrouting of packets leading to a deadlock.

This paper proposes a hybrid method to synthesize an area efficient fault-secure NoC switch for any single point combinational or transition delay fault in the switch or interconnect links, irrespective of its temporal nature (permanent, transient or intermittent fault). The method incorporates error detecting codes for data flits and a low-area concurrent error detecting structure to handle faults not covered by the flit encoding. Firstly, structural faults that are always detectable by data encoding at the flit-level are identified. Next, the fault-secure structure is constructed such that the rest of the faults become detectable under any given input vector using a parity based code. A Boolean satisfiability (SAT) based approach examines the fault-secureness property of the data encoding scheme as well as the final fault-secure structure. The method

is architecture independent and can be applied to arbitrary switches of any desired NoC topology with arbitrary routing function. To enable efficient online and offline testing of the switch, the parity trees of the fault-secure switch are reused to reduce the test response data volume and the test time.

The rest of the paper is organized as follows: Section II describes the fault-secure architecture and introduces the synthesis flow. Sections III and IV explain the SAT model construction and the synthesis algorithm in detail. Section V describes the structure reuse for test compaction. Experimental results are presented in section VI, followed by a conclusion.

II. Overview

A. Fault-Secureness

A circuit is fault-secure for a set of faults F if and only if:

$$\forall f \in F : \forall i \in I^n : C(i) = C^f(i) \vee \Pi(C(i)) \neq \Pi(C^f(i)). \quad (1)$$

For I^n being the set of possible input vectors of n bits, $C(i)$ denotes the circuit response for input vector i in the fault free case, while $C^f(i)$ is the response under fault f. Π is a function that computes the check bits of the circuit response for the selected error detecting code. The formula expresses that for *any* input vector, the fault either is not propagated to the circuit outputs, i.e. $C(i) = C^f(i)$, or it is detected by the check bits, i.e. $\Pi(C(i)) \neq \Pi(C^f(i))$ [4]. If a fault propagates to the circuit outputs but it is not detectable by the check bits, *Silent Data Corruption (SDC)* occurs.

B. Fault-Secure Architecture for Error Detection at Flit-Level

The NoC switch incorporates multiple input/output ports, crossbar multiplexers and control logic to manage the dataflow between input and output ports. Each switch is connected to its neighboring switches and a network interface via interconnect links. Each interconnect link consists of parallel data bits in the width of a flit (flow control unit) and handshake signals. The fault-secure switch presented here uses data encoding at the flit-level. As depicted in Fig. 2 for a sample five port switch of a 2D mesh topology, flit checkers (*Flit chk*) are positioned at output ports of the switch over the data bits.

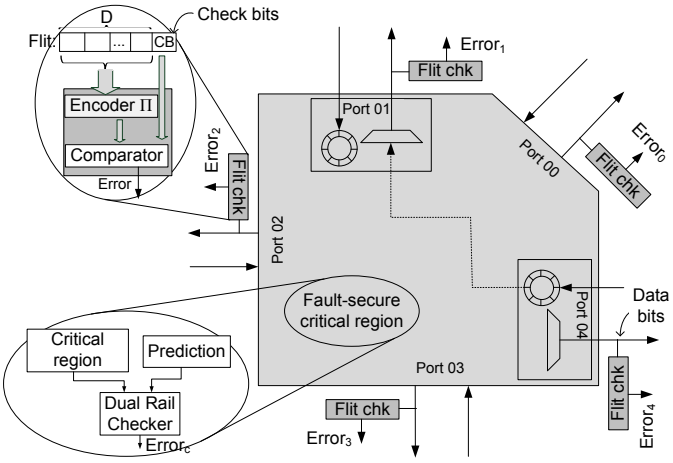

Fig. 2: Fault-secure structure of the NoC switch

$Error_0$ to $Error_4$ are the error signals of the flit checkers of port 00 to 04. Each flit includes a number of check bits appended to its content. Check bits are generated by the sender once a packet is injected into the network. Upon passing a flit through the flit checker, the encoder recomputes the check bits and compares them against the check bits stored in the flit. A mismatch causes the assertion of the error signal. Depending on the EDC, a subset of faults in the incoming data links and in the switch datapath are detectable by the flit checkers. In case of an end-to-end error control, e.g. [10], and omission of the flit checkers of the switch, the fault-secureness of the switch cannot be ensured anymore because a fault effect might be masked in upcoming switches.

Since flit checkers are constructed over a subset of switch outputs, i.e. data outputs, they are not sufficient to ensure the fault-secureness in the switch [15]. Some errors may propagate either to internal states of the switch or to other outputs than the data outputs. Moreover, certain faults may cause an error on the data outputs that is not detectable by the error detecting code (EDC) at flit-level. As an example, let us consider a 2-to-1 multiplexer as part of the crossbar in the switch datapath and assume a single parity bit is added to the flit as the check bit. In the 2-to-1 multiplexer of Fig. 3, $z = \neg s \cdot x + s \cdot y$. For Π that computes the parity check bit of a flit, when s is zero, $\Pi(z) = \Pi(x)$ and when s is one, $\Pi(z) = \Pi(y)$. Assume that there exist a stuck-at-1 fault at line a of the multiplexer as shown in Fig. 3. In this case, the multiplexer function for every bit i changes to: $z_i = x_i + s \cdot y_i$. When s is zero, z carries the value of x, and the fault is not detectable. However, when $s = 1$, z carries a bit-wise OR of the data on the lines x and y ($z = x + y$), and therefore $\Pi(z) = \Pi(x + y)$. In this case, erroneous data on z is not necessarily detected by the parity bit.

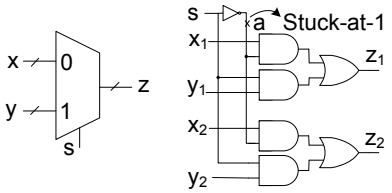

Fig. 3: 2-to-1 multiplexer: Stuck-at-1 fault violating fault-secureness

C. Synthesis Flow for Fault-Secureness

Figure 4 depicts the overall flow of the synthesis scheme. Initially, the switch including the flit checkers is analyzed to identify which faults propagate to data outputs and are detectable by flit checkers under *any* given input vector. These faults are named *covered faults*. The rest of the faults are categorized as *uncovered faults*, F_{uncov} (details in Section III).

With respect to the uncovered faults, a sub-block of the circuit is extracted, which is called *critical region*. It includes only the parts that influence the propagation of errors caused by uncovered faults. A CED structure is synthesized to make the critical region fault-secure. The method concentrates only on the uncovered faults in order to reduce the area overhead (details in Section IV).

Per switch, one error signal is generated by the disjunction over the error signals of the flit checkers and the error signal of the fault-secure critical region. It is sent to neighboring switches and the network interface via a dedicated error port in order to invoke an error recovery mechanism. The resulting switch, including flit checkers and CED circuitry of the critical region is fault-secure. If the original switch is non-redundant, it is also totally self-checking.

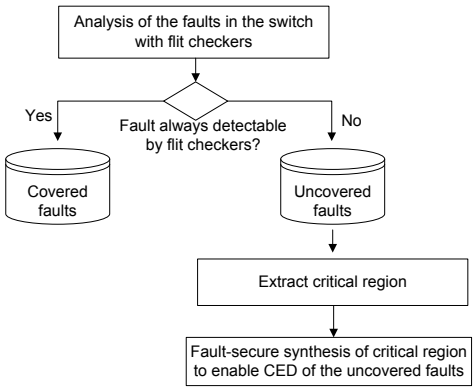

Fig. 4: Overall flow of the synthesis scheme

III. Identification of Uncovered Faults using Boolean Satisfiability

The switch with flit checkers is fault-secure, if the effect of faults that propagate to switch outputs is never masked at flit checkers under any input vector. Conducting exhaustive fault simulation to check this property for all faults in the switch is not possible due to the huge input vector space, I^n, when $n = \textit{\#switch input ports} \times (\textit{flit size} + \textit{\#handshake signals})$. However, to identify the faults that violate fault-secureness (i.e. uncovered faults F_{uncov}), it is sufficient to find one input vector ($i \in I^n$) for which the fault effect is propagated to switch outputs and masked at flit checkers. This is defined as a Boolean satisfiability (SAT) problem. Fault f violates fault secureness if and only if:

$$\exists i \in I^n : C(i) \neq C^f(i) \wedge \bigwedge_{0 \leq j < m} \Pi(D_j(i)) = \Pi(D_j^f(i)). \quad (2)$$

$C(i)$ is the switch response (including data outputs) for input vector i, and $C^f(i)$ is the switch response under fault f. For a switch with m output ports, $D_j(i)$ are the data outputs of port j in the fault-free case, and $D_j^f(i)$ are the data outputs under fault f. Π computes the check bits over the data bits of each port.

Inspired by statement (2), Fig. 5 shows the structure of the SAT instance to identify uncovered faults. The instance is satisfiable if and only if statement (2) is true. The combinational netlist of the switch is extracted by removing flipflops and replacing the input/output of flipflops by pseudo primary output/input ports respectively. As shown in Fig. 5, the outputs of the switch and the faulty copy, including primary and pseudo primary outputs, are compared bit-wise ①. The encoder structures Π compute the check bits which are then compared to those in the faulty instance ②.

The SAT instance Φ^f is a Boolean formula in Conjunctive Normal Form (CNF) of the characteristic function of the switch and the faulty copy, the encoder structures, and the functions required for comparison:

$$\Phi^f = CNF(S) \wedge CNF(S^f) \wedge (C \neq C^f) \wedge$$
$$CNF(\Pi) \wedge CNF(\Pi^f) \wedge \bigwedge_{0 \leq j < m} (\Pi_{D_j} = \Pi^f_{D_j}).$$

It includes the characteristic equations of the gates in the fault-free and faulty switch, $CNF(S)$ and $CNF(S^f)$. $CNF(\Pi)$ and $CNF(\Pi^f)$ represent the encoder structure over the data outputs, C and C^f represent switch outputs, and Π_{D_j} and $\Pi^f_{D_j}$ represent encoder outputs of each switch port j in the fault-free and faulty copy, respectively. For each fault, a new instance Φ^f is constructed. Φ^f is satisfiable if there exists one input assignment such that the outputs of the switch differ ($C \neq C^f$) and the encoder outputs are equal. In this case, the fault is an uncovered fault which is not detectable by flit checkers for at least the satisfying input vector found by the SAT solver.

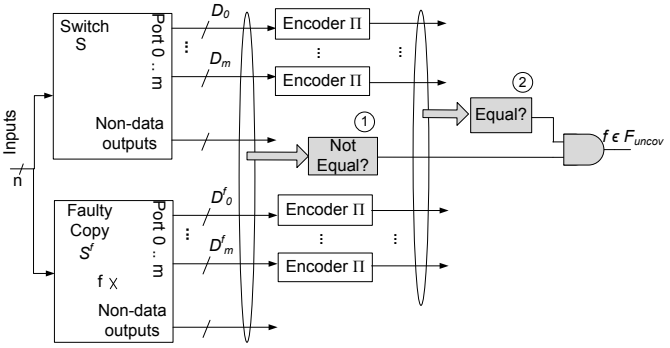

Fig. 5: Schematic of the SAT instance to identify uncovered faults

IV. Fault-Secure Synthesis for Uncovered Faults

We define the critical region $\mathcal{C}$ of the switch as the part of the logic that may influence the propagation of uncovered faults in the switch:

$$\mathcal{C} := \cup_{f \in F_{uncov}} \text{support}(f).$$

The support of a signal support(f) is the union of the gates in the input cones of all outputs reachable from f.

To ensure that uncovered faults do not violate fault-secureness by Silent Data Corruption (SDC), a multi-bit parity code is constructed over the outputs of the critical region. The outputs are distributed among the parity groups. A parity tree generates the parity bit over the outputs in the same group. To avoid masking of fault effects in parity trees, outputs in the same parity group must not share any logic in their input cone [16].

Figure 6 depicts the synthesis flow of the critical region. It starts with a topological analysis to construct initial parity groups. A SAT instance is constructed to identify the remaining faults in F_{uncov} that still cause SDC. The initial parity groups are split iteratively until all cases of SDC are resolved.

A. Topological Analysis

To reduce the area overhead of a parity-based fault-secure circuit, the number of parity groups must be minimized.

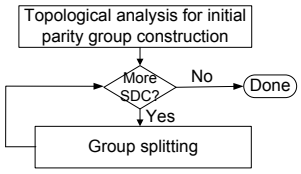

Fig. 6: Code synthesis for fault-secureness of the critical region

Therefore, the initial parity groups are constructed based on a topological analysis such that every uncovered fault effect is not masked in at least one parity group.

The topological analysis only considers the uncovered faults. If the gate of a fault $f \in F_{uncov}$ is not shared among the input cones of the outputs in at least one parity group, it is allowed to be shared among the outputs in the other groups. This is clarified with an example in Fig. 7. The striped area S is in the input cone of v_2, v_3, and v_4. Let us assume S contains an uncovered fault. Since v_2 is encoded in group 1 and is the only output in group 1 containing S in its input cone, it is allowed to put v_3 and v_4 in the same group, although they share the logic in S. Considering only the uncovered faults, the initial parity groups are constructed using the greedy algorithm given in [17].

The topological analysis reduces the probability of SDC and serves as a starting point to synthesize the fault-secure structure.

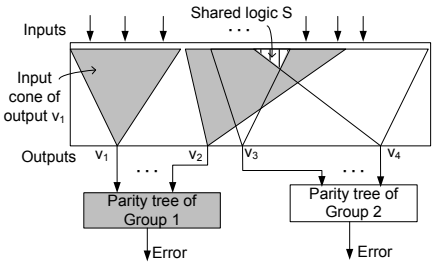

Fig. 7: Topological analysis for constructing initial parity groups

B. Resolving Silent Data Corruption

The topological analysis cannot guarantee the fault-secureness yet, since it only ensures that a fault effect is not masked in one parity group. There may exist input vectors that propagate the fault effect only to the groups where error masking is not inhibited, i.e. SDC occurs.

The SAT instance, Φ^f constructed to find the uncovered faults, is extended to find faults in F_{uncov} that cause SDC in the switch, which includes flit checkers and parity trees of the critical region. The clauses to represent the parity trees over the outputs of the critical region in the good copy $CNF(P_V)$, the parity trees over outputs in the faulty copy $CNF(P_{Vf})$, as well as the clauses to compare the parity bits in the good and faulty copy, i.e. $P(V)$ and $P(V^f)$, are added to the SAT instance:

$$\Phi^f_{SDC} = \Phi^f \wedge CNF(P_V) \wedge CNF(P_{Vf}) \wedge P(V) = P(V^f). \quad (3)$$

The SAT instance Φ^f_{SDC} is satisfiable if there exists one input vector i_f such that the fault is observable at switch outputs ($C \neq C^f$, defined in Φ^f), and it is not detectable by neither the flit checkers nor the parity groups of the critical region. Fault f is simulated with input vector i_f to find the outputs and parity groups to which the fault effect propagates. Then, using the group splitting algorithm of [17], the existing groups are partitioned so that the number of faults carry SDC is minimized. The splitting refines the parity groups iteratively until fault-secureness is achieved, i.e. any SDC is resolved.

V. Structure Reuse for Test Compaction

The parity trees of the fault-secure switch can be reused for test response compaction to reduce the storage and bandwidth requirements for manufacturing and in-field testing. For the compaction, the largest parity tree with n inputs is used. The scan chain is restructured (Fig. 8) such that in each shift-out cycle, the output bits included in a parity group are compacted by the largest parity tree. To compact and shift-out the responses of k parity groups, k cycles are required. The scan flip-flops of the outputs encoded at flit checkers are also included into the scan chains. A further reduction of the shift-out cycles is possible, if these flip-flops are used to balance the length of the parallel scan chains.

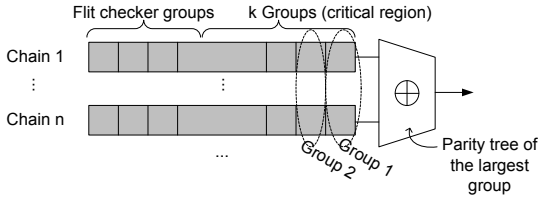

Fig. 8: Scan restructuring for test compaction

VI. Experimental Results

The fault-secure synthesis scheme is evaluated for a typical NoC switch. Firstly, we introduce the characteristics of the considered switch. Next, different error detecting codes (EDCs) for data encoding at flit-level are implemented and compared in terms of fault coverage and area overhead. Then, the result of the fault-secure switch is presented. Finally, we discuss the impact of the proposed fault-secure structure on test data compaction and diagnosis.

A. Switch Characteristics

The considered switch, designed for 2D mesh topology, consists of five input/output ports. It implements a wormhole XY routing and processes input ports in a round-robin fashion. The switch is synthesized using Synopsys Design Compiler. The target library lsi10k is constrained to basic gate primitives. Because memory elements are usually equipped with advanced BIST/BISR features [18], the storage elements of input ports are not considered in the fault-secure synthesis flow. Still, the error detecting code (EDC) of the flits partially protects the memory elements. The memory elements are also excluded from the area computations.

B. Evaluation of Flit Checkers

The choice of the code used for flit checkers influences both the number of covered faults and the area overhead. A single-bit parity code needs only a single check bit per flit and because of the simple encoder structure, it imposes a small

hardware overhead. It detects single bit errors and multi-bit errors of odd multiplicity. On the other hand, more complex EDCs such as cyclic codes are better suited to detect burst errors but impose a larger area or performance overhead.

The parity, Hamming, Berger, Hsiao, and CRC(4) (i.e. an unrolled cyclic code with a generator of degree 4) are implemented as the EDC in flit checkers with a flit width of eight bits. The SAT instances Φ^f are constructed for each code to compute the number of covered/uncovered faults. Fig. 9 compares the codes in terms of the area overhead of the flit checker with respect to the switch area and the portion of uncovered faults. A structural analysis shows that 6931 faults (56.85%) out of 12191 collapsed faults in the switch are not propagated to data outputs. Therefore, they are in principle not detectable by flit checkers. In all cases, around 60% of faults are recongnized as uncovered faults, that is they are not detectable by flit checkers for at least one input vector. Among the remaining 5260 faults that are propagated to data outputs, only the ones which are detectable by flit checkers belong to covered faults. In order to evaluate the detectability of EDCs in flit checkers, parameter γ is defined as:

$$\gamma = \frac{\text{number of covered faults}}{\text{number of faults propagated to data outputs}} \times 100\%.$$

γ quantifies the portion of faults that are detectable by flit checkers over the number of faults which are propagated to data outputs. Fig. 9 compares parameter γ of the selected EDCs. The Berger code has the highest detectability of 99.38%, while the other codes have a detectability of approximately 87%. In general, the flit checkers can detect around 40% of the faults of the 8-bit switch with a maximum area overhead of 15%.

To investigate the flit width impact on the effectiveness of flit checkers, the number of covered/uncovered faults are computed for the switch with the flit width of 16, 32, 64, and 128 bits while parity is selected as the EDC. Table I depicts the results. The second column reports the cell area of each switch next to the flit width. The third and forth column report the number of collapsed faults and the number of uncovered faults in each switch. The fifth column shows the number of faults which are propagated to data outputs and the next column reports γ.

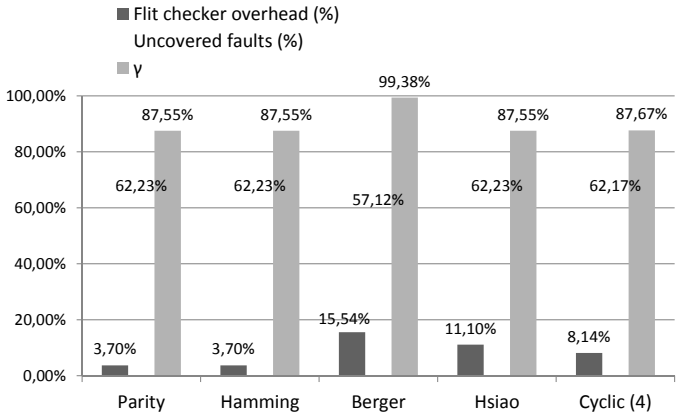

Fig. 9: Error detecting codes: flit checker overhead - uncovered faults - γ

The result reveals that in the switch with the larger flit width, the portion of uncovered faults is less. This is because more faults are propagated to switch data outputs and the fraction of faults detected by the flit checkers increases. When the flit width increases, the width of datapath elements such as crossbar multiplexers increases. Moreover, parameter γ indicates that as the flit width increases, a bigger portion of faults which are propagated to data outputs become detectable by flit checkers. In fact, the number of faults which are not detectable by flit checkers remained unchanged (940 faults). These faults are located at some of the control signals, for example multiplexer select signals, and resulting errors are propagated to data outputs. Changing the flit width does not influence the portion of control logic in the considered switch.

The time required to compute the uncovered faults and the corresponding critical region is reported in the last column of Table I. In bigger switches more faults are propagated to the data outputs that must be traced by SAT for SDC. But the time does not increases linearly because the size of the switch, which determines the complexity of the SAT instance, increases as well. Despite that, the method can still generate the fault-secure switch in a reasonable time.

C. Result of Fault-Secure Synthesis

Considering parity as the EDC in flit checkers and based on the list of uncovered faults, the critical region is constructed and the synthesis process is performed to enable concurrent error detection of uncovered faults.

Table II summarizes the result. As shown in the first row, by constructing only flit checkers with 3.7% area overhead, 7871 faults still cause SDC. According to the proposed synthesis procedure, by constructing 43 extra parity groups over outputs of the critical region, fault-secureness is ensured (i.e. no SDCs). The resulting fault-secure switch has 52% area overhead compared to the original switch. The overhead includes the flit checkers, prediction logic and dual rail checkers for the critical region. Any single transient and permanent fault that generates erroneous bits in the switch is detectable by this fault-secure structure.

As shown in the third row of Table II, duplication with comparison for the entire switch with 1040 primary and pseudo primary outputs imposes a huge area overhead of more than 370% due to the large number of outputs and the overhead of dual rail checkers which are required to ensure fault-secureness. Even without dual rail chackers duplication imposes an area overhead of 204%. It reveals that for bigger switches with even more outputs the area overhead of duplication will increase drastically.

The critical region occupies 41% of the switch area and contains 185 outputs. As shown in the last row of Table II, using duplication with comparison of just the critical region imposes an area overhead of 89% (including dual-rail checkers for output comparison). The presented fault-secure switch saves almost 37% area cost compared to duplication with comparison for the critical region and significantly reduces

TABLE I: Number of covered/uncovered faults with respect to the switch flit width

Flit Width	Cell Area	# Collapsed Faults	Uncovered Faults	Faults propagated to data outputs	γ (%)	Time (s)
16-bit	7259	18901	10261 (54.29%)	9580 (50.69%)	90.19	784
32-bit	11979	32341	15061 (46.57%)	18220 (56.34%)	94.84	1924
64-bit	21419	59221	24661 (41.64%)	35500 (59.94%)	97.35	7266
128-bit	39017	112991	43871 (38.83%)	70060 (62.00%)	98.66	42963

the area overhead compared to conventional duplication with comparison for the entire switch.

TABLE II: Result of fault-secure synthesis - 8-bit switch

Synthesis technique	# groups	# SDCs	overhead
Switch + flit checkers	5	7871	3.70%
Proposed fault-secure switch	*48*	*0*	*52.01%*
Duplicated switch	-	0	371.29%
Duplicated critical region	-	0	89.09%

D. Test Compaction and Diagnosis

Table III presents the result of test pattern generation for the original switch and the switch with parity trees (fault-secure switch) using a commercial ATPG tool. The result shows that with only 9% of test response data 100% stuck-at fault coverage is achieved in the fault-secure switch.

TABLE III: Comparison of test and diagnosis results

	Switch	Fault-secure switch
# Test patterns	372	394
Fault coverage (%)	100	100
Test response volume [bit]	386880	34672
Diagnostic success (%)	98.76	98.50

The compacted test responses can still be used for the diagnosis of the switch [19]. Using the algorithm in [20], diagnosis is performed for the switch and the switch with parity trees. The last row of Table III compares the diagnostic success (i.e. the top-ranked fault suspect corresponds to the defect [20]) of the uncompacted response of the original switch with the diagnosis of the parity streams in the fault-secure switch targeting stuck-at faults. Even with the compacted test response, the diagnostic success rate is approximately the same as the uncompacted response. Indeed, without loss of fault coverage and diagnostic success, the parity trees of the fault-secure switch can be used to reduce the amount of response data as well as the response shifting cycles by a factor of 11x.

VII. Conclusion

This paper presented a hybrid method to synthesize a fault-secure NoC switch employing data encoding at flit-level and concurrent error detection with multiple parity trees. Boolean satisfiability is used to identify faults that cause silent data corruption and to direct the construction of an error detecting code. With an area overhead of only 52%, the resulting fault-secure switch is able to detect any single combinational and transition delay fault in the switch and interconnect links. The structure can be reused for test compaction, which reduces the amount of response data as well as test time without loss of fault coverage and diagnosis success.

VIII. Acknowledgment

This work was supported by the German Research Foundation (DFG) under grant WU 245/12-1 (ROCK).

References

[1] G. Gielen *et al.*, "Emerging yield and reliability challenges in nanometer CMOS technologies," in *Proc. Design, Automation and Test in Europe (DATE)*, 2008, pp. 1322–1327.

[2] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, 2005.

[3] M. Nicolaidis and Y. Zorian, "On-line testing for VLSI - A compendium of approaches," *Journal of Electronic Testing – Theory and Applications*, vol. 12, no. 1-2, pp. 7–20, 1998.

[4] P. K. Lala, *Self-Checking and Fault Tolerant Digital Design*. Morgan Kaufmann, 2001.

[5] Y. Zhang, H. Li, and X. Li, "Reliable network-on-chip router for crosstalk and soft error tolerance," in *Proc. IEEE Asian Test Symp. (ATS)*, 2008, pp. 438–443.

[6] A. P. Frantz *et al.*, "Dependable network-on-chip router able to simultaneously tolerate soft errors and crosstalk," in *Proc. IEEE Intl. Test Conf. (ITC)*, 2006, pp. 1–9.

[7] A. Yanamandra *et al.*, "Optimizing power and performance for reliable on-chip networks," in *Proc. 15th Asia and South Pacific Design Automation Conf. (ASP-DAC)*, 2010, pp. 431–436.

[8] A. Ghofrani *et al.*, "Comprehensive online defect diagnosis in on-chip networks," in *Proc. IEEE VLSI Test Symp. (VTS)*, 2012, pp. 44–49.

[9] T. Lehtonen *et al.*, "Self-adaptive system for addressing permanent errors in on-chip interconnects," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 4, pp. 527–540, 2010.

[10] D. Rossi, P. Angelini, and C. Metra, "Configurable error control scheme for NoC signal integrity," in *Proc. IEEE Intl. On-Line Testing Symp. (IOLTS)*, 2007, pp. 43–48.

[11] M. Palesi *et al.*, "Data encoding schemes in networks on chip," *IEEE Trans. CAD*, vol. 30, no. 5, pp. 774–786, 2011.

[12] D. Bertozzi, L. Benini, and G. De Micheli, "Error control schemes for on-chip communication links: the energy-reliability tradeoff," *IEEE Trans. CAD*, vol. 24, no. 6, pp. 818–831, 2005.

[13] C. Grecu *et al.*, "Essential fault-tolerance metrics for NoC infrastructures," in *Proc. IEEE Intl. On-Line Testing Symp. (IOLTS)*, 2007, pp. 37–42.

[14] A. Dutta and N. A. Touba, "Reliable network-on-chip using a low cost unequal error protection code," in *Proc. IEEE Symp. Defect and Fault-Tolerance in VLSI Systems (DFT)*, 2007, pp. 3–11.

[15] M. Augustin, M. Gossel, and R. Kraemer, "Reducing the area overhead of TMR-systems by protecting specific signals," in *Proc. IEEE Intl. On-Line Testing Symp (IOLTS)*, 2010, pp. 268–273.

[16] N. Touba and E. McCluskey, "Logic synthesis of multilevel circuits with concurrent error detection," *IEEE Trans. CAD*, vol. 16, no. 7, pp. 783–789, 1997.

[17] A. Dalirsani, M. A. Kochte, and H.-J. Wunderlich, "SAT-based Code Synthesis for Fault-Secure Circuits," in *Proc. IEEE Symp. Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, 2013.

[18] M. Zhang *et al.*, "Sequential element design with built-in soft error resilience," *IEEE Trans. VLSI*, vol. 14, no. 12, pp. 1368–1378, 2006.

[19] A. Dalirsani *et al.*, "Structural test for graceful degradation of noc switches," in *Proc. IEEE European Test Symp. (ETS)*, 2011, pp. 183–188.

[20] S. Holst and H.-J. Wunderlich, "A diagnosis algorithm for extreme space compaction," in *Proc. Design, Automation and Test in Europe (DATE)*, 2009, pp. 1355–1360.