A Framework for Comprehensive Automated Evaluation of Concurrent Online Checkers

Pietro Saltarelli1,2, Behrad Niazmand1, Jaan Raik1, Ranganathan Hariharan1, Gert Jervan1, Thomas Hollstein1
1Tallinn University of Technology
Tallinn, Estonia
2Università degli Studi di Ferrara
Ferrara, Italy
pietro.saltarelli@student.unife.it
{bniazmand, jaan, ranga, gert, thomas}@ati.ttu.ee

Abstract— This paper proposes a framework for automated evaluation of concurrent online checkers. The novelty of the underlying approach lies in its completeness (i.e. ability of formally proving the presence or absence of true misses), minimal fault detection latency and accurate, fully automated evaluation of the fault detection characteristics of the checkers. The methodology consists of creating a pseudo-combinational version of the circuit under test, specifying the environment in terms of valid input stimuli and providing the assertions for generating the checkers, which will thereafter be evaluated by the framework. In this paper, a case-study on the control part (routing and arbitration) of a Network-on-Chip (NoC) router has been carried out. It shows on a realistic application that the framework is capable of accurately and formally evaluating the quality of individual concurrent checkers which constitutes an important task in fault tolerant system design. The case study shows that the proposed approach helps achieving high fault coverage in a single clock-cycle.

Keywords— concurrent online checking, Network-on-Chip, routing logic, arbitration.

I. INTRODUCTION

Extreme scaling of nanometer technologies has made the electronic systems increasingly vulnerable to wear-out and environmental effects (e.g. soft errors, electro-magnetic interference). These are issues occurring during the life-time of the system and cannot be filtered out by manufacturing testing. Thus, online solutions for detecting faults are needed. These solutions should preferably be concurrent to the normal circuit operation.

One of the possible solutions for concurrent online test is the use of checkers for monitoring faults occurring within the circuit. In this paper, we introduce a framework for accurate, automated evaluation of concurrent online checkers. The methodology includes preparation of the checkers in the form of verification assertions (or reuse of existing assertions), creation of a pseudo-combinational version of the circuit under test and specifying the environment in terms of valid input stimuli for it.

Subsequently, the set of obtained checkers, together with the stimuli and the circuit are given to the framework that accurately evaluates the fault detection characteristics of the given checkers. The underlying approach in the framework is complete, i.e. it allows proving the absence or presence of true misses by the checkers. In addition, it provides minimal fault detection latency due to the fact that the circuit is transformed into a pseudo-combinational one and therefore only checkers with a single clock cycle latency are considered.

The proposed approach is applicable to control-oriented designs. In this paper, a case-study on the control part (routing and arbitration) of an NoC router has been carried out. It shows on a realistic application that the framework is capable of evaluating the quality of individual concurrent checkers which constitutes an important task in fault-tolerant system design.

The paper is organized as follows. Section 2 provides an overview of related works in concurrent online testing. Section 3 gives an overview of the concurrent online checking concept. In Section 4, the proposed framework and the corresponding methodology are presented. Section 5 presents the target architecture of the control part of a Network-on-Chip (NoC) router. Section 6 provides the checker evaluation experiments. Finally, Section 7 concludes the paper.

II. RELATED WORKS

Online detection of errors in logic is a thoroughly studied research area. Traditional Triple-Modular Redundancy (TMR) and duplication-based approach are too costly in terms of multiplying the area and correspondingly the power consumption. An alternative to minimize this overhead is the selective TMR that identifies Single Event Upset (SEU) sensitive sub-circuits that are to be protected [1].

In addition, there exists a variety of solutions based on coding techniques such as Berger [2] or Bose-Lin [3] codes. In many works the coding techniques are combined with synthesis [4,5]. The approaches suffer from significant area overhead as well as require alteration of the original circuit in order to generate the codes.
Concurrent on-line built-in self-test techniques such as Built-In Concurrent Self-Test (BICST) [6] and Reduced Observation Width Replication (ROWR) [7] provide high fault coverage at low area overhead but only consider a limited subset of pre-computed test vectors. Hence these approaches are likely to miss faults occurring in a normal circuit operation.

Several alternatives based on fault monitors and checkers that do not require modification of the circuit under test have been developed. Creating checkers automatically based on logic implications derived from the circuit structure [8] is feasible but suffers from low fault coverage and high area overhead, often exceeding the duplex solutions. On the other hand, deriving checkers from functional assertions, or reusing verification assertions, is similarly known to yield low coverage of structural faults as it is difficult to correlate functional coverage to structural one [9].

The framework and methodology presented in this paper exceed the existing state-of-the-art in concurrent online checking in the following aspects:

- It allows formally proving the absence or presence of true misses over all possible valid inputs for a checker, whereas in the case of traditional fault injection only statistical probabilities can be calculated without providing the user with full confidence of fault detection capabilities.
- The methodology targets the minimum fault detection latency of a single clock-cycle. This is achieved by representing the circuit under test as a pseudo-combinational design and concentrating on combinational checkers.
- The framework provides accurate, fully automated evaluation for the fault detection characteristics of the checkers. It allows finding cost-efficient trade-offs between the fault detection capabilities and the required overhead area.

III. THE CONCEPT OF CONCURRENT CHECKERS

Fig. 1 presents the role of concurrent on-line checkers in detecting faults within a circuit. In addition to the original circuit (functional logic), a set of checkers (checker logic) will be connected to functional inputs/outputs of the circuit. These checkers are introduced based on functional assertions derived from relationships between variables corresponding to inputs and outputs of the circuit. The checker logic targets the faults at lines within the functional logic (marked by green circles). The lines at the functional outputs succeeding the checker inputs (marked by a red cross) can not be detected by the checker. In addition, the checkers are not targeting the faults at functional inputs preceding checker inputs since the checker may not detect that the input value has been altered by a fault. (Such functional input lines are also marked by a red cross in Fig. 1). In this paper, we consider the single stuck-at fault model. However, due to the fact that concurrent checkers are implemented and a single time-frame is targeted, the model also covers timing related faults.

Given a fault at a line within the functional logic and a set of input stimuli, four possible scenarios may occur:

- Case 1: Fault occurs at an internal line but is visible at functional output(s) and checker logic flags a violation. The term True Detection is used to describe this situation, since a critical fault is effectively detected by the checker.
- Case 2: Fault occurs at an internal line but is not visible at functional output(s). Checker catches the fault and flags a violation. The term False Positive is used to describe this situation. False positive is not harmful because an error is flagged which did not have any effect. However, it has negative impact on designs performance because normally it causes re-execution of the task. In the experiments in this paper we did not encounter any cases of false positives.
- Case 3: Fault occurs at internal line but is not visible at functional output(s) and the checker logic does not detect the violation. The term Benign Miss is used to describe this situation. Benign miss shows correct operation by the checker.
- Case 4: Fault occurs at internal node and is visible at functional output(s). Checker does not detect violation. The term True Miss is used to describe this situation, which is the worst possible case. True miss means that the fault propagates to the functional outputs and onwards to the system. However, the system has no information that a critical fault has occurred.

Traditionally, in order to evaluate the fault detection quality of the checkers, fault injection has been applied. Fault injection refers to injecting faults into a circuit at a certain time step and simulating it with the input stimuli to see whether any functional output of the circuit changes and whether any of the checker output fires. Due to the fact that it is generally impossible to inject and simulate all the faults at each circuit line at each time step, a statistically significant sample of random faults would normally be injected and simulated.

However, in this paper a methodology is proposed which is based on automated extraction of a pseudo-combinational circuit out of the original functional logic by breaking the flip-flops and converting them to pseudo-primary inputs and pseudo-primary outputs. Further, an exhaustive test for the extracted circuit is fed through a filtering tool in order to derive the exhaustive valid set of input stimuli which will serve as the environment for checker evaluation. This means that in this paper full evaluation of the checkers with all possible stimuli and faults is obtained.

Let D be the number of true detections, X be the number of benign misses and W be the number of true misses over all the injection runs. Then we define the metrics of Fault Coverage (FC) and Checkers’ Efficiency Index (CEI) as follows.

- Fault Coverage (FC) = \( \frac{D}{D + X + W} \)
- Checkers’ Efficiency Index (CEI) = \( \frac{D}{D + X + W} \)
Here, FC shows the probability of the checkers behaving correctly over all possible cases and CEI shows the probability of checkers' ability to detect critical faults. As mentioned above, the approach proposed in this paper is able to formally prove the presence or absence of true misses. Due to the fact that none of the checkers resulted in false positives, this information is excluded from the metrics.

**IV. FRAMEWORK FOR CHECKER EVALUATION**

Fig. 2 presents the flow of the checker evaluation framework together with the respective methodology. The flow starts with synthesizing the checkers (described in RTL Verilog) from a set of combinational assertions. Thereafter, a pseudo-combinational circuit will be extracted from the circuit of the design under checking. The pseudo-combinational circuit is derived out of the original circuit by breaking the flip-flops and converting them to pseudo-primary inputs and pseudo-primary outputs. Note, that at this point additional checkers that describe relations also on the pseudo-primary inputs/outputs may be added to the checker suite in order to increase the fault coverage.

Subsequently, the checker evaluation environment is created by generating exhaustive test stimuli for the extracted pseudo-combinational circuit. These stimuli are fed through a filtering tool which selects only the stimuli that correspond to functionally valid inputs for the circuit. As a result, the exhaustive valid set of input stimuli which will serve as the environment for checker evaluation is obtained.

The obtained environment, pseudo-combinational circuit and synthesized checkers are applied to fault free simulation. The simulation calculates fault free values for all the lines within the circuit. Additionally, if any of the checkers fires within fault simulation, it means a bug in the checker or an incorrect environment. During the case study presented in Section 5 several bugs were detected by this simulation step.

If none of the checkers are firing in the fault-free mode, then checker evaluation takes place. The tool injects faults to all the lines within all the vectors. As a result, the overall fault detection capabilities for the set of checkers, in terms of FC and CEI metrics will be calculated. In addition, each individual checker will be weighted by summing up the total number of true detections by the checker.

The framework is developed as an extension of a freeware test system Turbo Tester [10]. The system applies Structurally Synthesized Binary Decision Diagram (SSBDD) models [11] for circuit modelling.

**Figure 2.** The proposed checker evaluation flow

**V. CASE-STUDY DESIGN: NoC ROUTER**

Fig. 3 demonstrates the high-level overview of a 5-port 2D NoC router that we have chosen as a target architecture for applying the checkers. Mainly, the router consists of a data path and a control part. The data path is composed of input buffers (implemented as First-In-First-Out (FIFO)), one for each input port, a crossbar switch and an output buffer for each output port. The main responsibility of the data path is to transmit data to destination.

The flow of data through the datapath is managed and controlled by the control part, which consists of a routing computation unit for each input port and an arbitration unit (arbiter) for each output port, which prioritizes the requests from different input ports to the corresponding output port. The router has 5 input/output ports, four ports connected to four cardinal directions (North – N, East – E, South – S, West – W) and one Local (L) port connected to the local processing element. The NoC router utilizes wormhole switching. Therefore, packets are sent in form of flits, consisting of header flit, body flit(s) and tail flit.

For the routing computation unit of our target architecture, we have opted for Logic-Based Distributed Routing (LBDR) [12], which is considered as a scalable solution compared to routing tables. The mechanism describes the topology and the
routing function in form of connectivity and routing bits, respectively. Therefore, the logic can be easily re-configure
d. Routing decision is distributed and only requires local 
destination addresses for forwarding flits.

In this work we focus on a 2D Mesh topology and we 
consider XY as the routing algorithm, which is a determi
nistic dimension-ordered algorithm, and we assume that 180
degrees turns are not allowed. This would in turn lead to 
further simplification of the logic of LBDR. The basic 
mechanism of the logic is shown in Fig. 4, customized for the 
East input port.

<table>
<thead>
<tr>
<th>Header Flit</th>
<th>flit_type</th>
<th>dst_addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing and Connectivity bits (based on XY Routing and 2D Mesh topology)</td>
<td>Rne = 0 Rn = 1</td>
<td>Rse = 0</td>
</tr>
<tr>
<td>Cn = 1 Ce = 1</td>
<td>Rwn = 1</td>
<td>Rsw = 0</td>
</tr>
<tr>
<td>Cw = 1 Cs = 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

First part of the routing logic:
- X_curn
- Y_curn
- CMP
- E'
- N'
- W'
- S'
- CMP

Second part of the routing logic (for ELBDR):
- N' = N'.E'.W'
- W' = W'.N'.S + W'.N'
- S' = S'.Cs
- L = N'.E'.W'.S'

For the arbitration unit (arbiter) we have chosen Round
Robin (RR) policy for prioritizing the requests from the 
routing logic of different input ports. Prioritization is circula
r, thus ensuring the absence of starvation, and guaranteeing that 
eventually any input port will get access to the requested 
output port.

Arbiter grants the access to the requesting input port 
winning the eventual contention, allowing data to go from the 
input FIFO to the corresponding output port, through the 
crossbar switch. The arbitration mechanism is based on an 
internal Finite State Machine (FSM). In this work one-hot 
encoding has been considered for the state variable, in order 
to improve detections of faults in the logic. Moreover, one-hot 
encoding is extended to grant signals and select lines for the 
crossbar switch.

The design decision to implement a one-hot encoded 
arbiter state machine versus a binary encoded one did increase 
the area of the arbiter by 27.7%. However, the CEI nearly 
doubled from 58.55% to 100% and the fault coverage increased from 93.69% to 100%.

Extracting the pseudo-combinational circuit

For the arbitration unit (arbiter) we have chosen Round
Robin (RR) policy for prioritizing the requests from the 
routing logic of different input ports. Prioritization is circular, 
thus ensuring the absence of starvation, and guaranteeing that 
eventually any input port will get access to the requested 
output port.

For the arbitration unit (arbiter) we have chosen Round
Robin (RR) policy for prioritizing the requests from the 
routing logic of different input ports. Prioritization is circular, 
thus ensuring the absence of starvation, and guaranteeing that 
eventually any input port will get access to the requested 
output port.

For the arbitration unit (arbiter) we have chosen Round
Robin (RR) policy for prioritizing the requests from the 
routing logic of different input ports. Prioritization is circular, 
thus ensuring the absence of starvation, and guaranteeing that 
eventually any input port will get access to the requested 
output port.

For the arbitration unit (arbiter) we have chosen Round
Robin (RR) policy for prioritizing the requests from the 
routing logic of different input ports. Prioritization is circular, 
thus ensuring the absence of starvation, and guaranteeing that 
eventually any input port will get access to the requested 
output port.

For the arbitration unit (arbiter) we have chosen Round
Robin (RR) policy for prioritizing the requests from the 
routing logic of different input ports. Prioritization is circular, 
thus ensuring the absence of starvation, and guaranteeing that 
eventually any input port will get access to the requested 
output port.

For the arbitration unit (arbiter) we have chosen Round
Robin (RR) policy for prioritizing the requests from the 
routing logic of different input ports. Prioritization is circular, 
thus ensuring the absence of starvation, and guaranteeing that 
eventually any input port will get access to the requested 
output port.

Filtering the stimuli

The exhaustive test for the pseudo-combinational circuit 
would require $2^{22}=4,194,304$ input stimuli. However, in order 
I to minimize the stimuli, and more important, to avoid 
checkers being evaluated in non-realistic conditions, the 
exhaustive set of stimuli has to be filtered to contain only the 
functionally feasible values.

For the pseudo-combinational partial control part of an 
NoC router studied here the filtering step is based on the 
implemented routing algorithm and restrictions in the routing 
logic, as well as on invalid conditions for the state and the 
stimuli of the arbiter logic. Its use allowed us to shrink the 
exhaustive set of $2^{22}$ stimuli to a valid and complete set 
consisting of 40,960 input vectors, which is less than 1% of 
the initial number. It is important to stress the fact that none 
of the checkers fires in fault free simulation with any of the 
considered input stimuli.
Preparing the checkers

The set of checkers consists of 37 checkers, based on the functionality of the considered circuit, 3 of them focusing on the ELBDR logic, 34 (12 types) focusing on the SArbiter logic. Due to spatial limitations, we have not explicitly reported the list of checkers individually in this paper.

VI. EXPERIMENTAL RESULTS

Experiments for the checker evaluation framework were carried out on the circuit, set of checkers and test stimuli described in previous section. As a result, the Fault Coverage (FC) of 99.777% and the Checkers’ Efficiency Index (CEI) of 99.320% were obtained. Each individual checker was weighted by the tool by summing up the total number of true detections by the checker. Fig. 6 lists the 14 checkers with the highest weights in a descending order. As it can be seen, four checkers are detecting considerably more faults than the others.

The proposed framework produced checkers that achieve 99.777% fault coverage for the NoC control part in a single clock cycle. Similar number for NoCAAlert [14] was 97%, and the technique reached 100% after 28 cycles. The respective numbers for ForEVeR [13] were 30% and 11950 cycles. This gain in the proposed approach was achieved due to the facts that there were checkers devised for arbiter states and that the implemented state encoding was one-hot.

The area overhead of the initial set of checkers was relatively high, doubling the size of the LBDR and arbiter combined. However, the size of the checkers compared to the entire router area was negligible. Moreover, the checkers’ weighting data allows to further compact the number of checkers in a straightforward manner.

VII. CONCLUSIONS AND FUTURE WORK

The paper presented a framework for automated evaluation of concurrent online checkers, which is formal (able of proving the presence or absence of true misses), yields minimal fault detection latency and enables accurate, fully automated evaluation of the fault detection characteristics of a given set of checkers.

A case-study on the control part (routing and arbitration) of a Network-on-Chip (NoC) router showed on a realistic application the feasibility and efficiency of the framework and the underlying methodology. Experimental results also showed that the proposed approach allows reaching higher fault coverage within a single clock-cycle compared to previously published approaches.

As a future work we consider extending the framework with the support of temporal checkers in order to further increase the fault coverage, for those designs where pseudo-combinational extraction is either not feasible or not sufficient. In addition, we plan to develop algorithms for minimization of checkers based on the weights calculated by the proposed framework.

ACKNOWLEDGEMENTS

The work has been supported by EU’s FP7 STREP BASTION, EU’s H2020 RIA IMMORTAL, Estonian Science Foundation grant ETF9429, Estonian institutional research grant IUT 19-1, funded by Estonian Ministry of Education and Research, and by EU through the European Structural and Regional Development Funds.

REFERENCES