A Framework for Area-Efficient Concurrent Online Checkers Design

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

Abstract—This paper proposes a framework for automated evaluation and minimization of concurrent online checkers, with the aim of both achieving minimal fault detection latency, while at the same time maintaining the fault detection capabilities and keeping area consumption within the acceptable range. The proposed framework can be utilized for any digital circuit, however, our focus in this work as a case study is applying the framework to the control part of a Network-on-Chip router consisting of the routing computation and arbitration units. The novelty of the framework is its ability to formally prove the presence or absence of true misses. Experiments are performed both regarding the latency and the fault coverage of the checkers devised by the framework which indicate 100% fault coverage with acceptable area overhead and also instant (one cycle) detection of faults in the control part of a NoC router.

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

I. INTRODUCTION

The extreme scaling of nanometer technologies has caused the circuits to be more susceptible to faults caused by wear-out and external sources. Such faults cannot be captured during the manufacturing of the device and therefore there is a need for a mechanism to detect them during the life time of the system. Therefore, this necessitates the introduction of a sort of online fault monitors, named as checkers which detect such faults during the runtime of the device.

In this paper, an automated framework is proposed which provides a minimized list of online checkers for checking the control part of on-chip routers. The tool flow in the framework is based on accurate, automated evaluation of concurrent online checkers. The checkers are first prepared as a set of verification assertions, afterwards a pseudo-combinational version of the circuit is created along with a set of valid input stimuli for it, which finally they are all fed to the evaluation part of the framework. The output of this part of the framework is computation of weights for each checker based on the number of true detections it has had.

Thereafter, minimization step comes into play which minimizes the number of checkers in each set, which is performed based on a divide-and-conquer approach. Weight information of the checkers is applied in a heuristic minimization method. The final result provided by the framework will be a subset of the initial checkers which is minimized and can achieve a target fault coverage level.

The proposed framework is capable of proving the absence or presence of true misses when faults are being detected by the checkers. Moreover, the fault detection latency of the checkers is one clock cycle as the framework works based on the pseudo-combinational version of the circuit and elapse of time is not considered. Our experimental results will show that the proposed framework provides a minimized set of checkers that reaches 100% fault coverage for a case study of the control part of an NoC router, consisting of the routing and arbitration units with acceptable area overhead.

The paper is organized as follows. Section 2 provides an overview of related works in concurrent online testing. Section 3 explains the concurrent online checking concept. In Section 4, the automated tool flow and the corresponding methodology for checkers’ minimization are presented. Section 5 presents the target architecture of the control part of a Network-on-Chip (NoC) router. Section 6 discusses application of the checker evaluation and minimization framework to the NoC Router design. Section 7 provides the checkers’ evaluation and minimization experiments. Finally, Section 8 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 approaches 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 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].

Many previous works have focused on addressing faults in the control logic of NoC routers. In [15], Yu et al. have addressed fault tolerance for NoC topologies and proposed an error control method for detecting transient errors in routing logic implemented using Logic-Based Distributed Routing (LBDR) mechanism and its extension for high-radix topologies, LBDRhr. The proposed error control method utilizes the inherent information redundancy (IIR) to reduce the error control overhead. However, the method does not guarantee full fault coverage.

Authors of [16] have presented a method for online error detection and diagnosis of NoC switches. The proposed method deals with routing faults that cause NoC packets to be forwarded to output ports that are not intended to. Regarding modeling routing faults in switches, a high-level fault model has been introduced in this work. The fault coverage is measured only at the functional level and there is no estimates of correlation to gate-level fault coverage.

Parikh et al. have proposed ForEVeR [13], where in order to deliver correctness guarantees for the complete network, a network-level detection and recovery solution is devised that monitors the traffic in the NoC and protects it against functional bugs that were not detected during design time. To this end, ForEVeR augments the baseline NoC with a lightweight checker network that alerts destination nodes of incoming packets ahead of time and is used for the recovery process. The approach suffers from extremely high latency. Only 30% of the faults will be detected during the first clock cycle by the approach.

[14] proposes checkers synthesized from a set of 32 verification assertions. The checkers detect most of the injected faults. The faults that are not covered correspond to non-catastrophic failures. The work proposed in [14] lacks the completeness and minimization aspects present in the current paper.

This paper exceeds the existing state-of-the-art in concurrent online checking by proposing a tool flow for automated evaluation and minimization of the verification checkers. We show that starting from a realistic set of verification assertions a minimal set of checkers will be synthesized that provide 100% fault coverage at a low area overhead and the minimum fault detection latency of a single clock-cycle. The latter is especially crucial for enabling rapid fault recovery in reliable real-time systems.

Additional features of the proposed approach is that 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.

### III. THE CONCEPT OF CONCURRENT CHECKERS

Concurrent on-line checking of faults in a digital circuit by means of checkers is demonstrated in Fig. 1. As it can be seen the main circuit (functional logic) has some inputs and outputs which are fed to the checker logic and the checkers possess some outputs themselves which go high in case of a fault detection. The checker logic targets the faults at lines at the inputs of each gate within the functional logic (marked by green circles). The lines at the functional outputs succeeding the checker inputs (marked by a red cross) cannot 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, as classified in [14]: 1) Fault is propagated to the output of the circuit and also detected by the checkers (True Detection), 2) Fault is not propagated to the output of the circuit, however it is detected by the checkers (False Positive), 3) Fault is not propagated to the output of the circuit and checkers do not detect it as well (Benign Miss) or 4) Fault propagates to the output of the circuit, but checkers do not capture it (True Miss) which is a critical case.

Traditionally, in order to evaluate the fault detection quality of the checkers, fault injection has been applied. However, due to the difficulty of injecting and simulating all the faults at each circuit line at each time step, a statistically significant sample of random faults is normally considered.

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. Afterwards, via a filtering tool, the valid set of input stimuli for the generated circuit is extracted from a set of exhaustive input patterns, which will serve as the environment for checkers' evaluation. This means that in this paper formal evaluation of the checkers with all the valid stimuli and faults is obtained.
In order to evaluate the fault detection capability of the checkers, we have made use of two metrics in this work. If we assumed \( D \) as the number of true detections, \( X \) as the number of benign misses and \( W \) as the number of true misses over all the injection runs, then the metrics of Fault Coverage (FC) and Checkers’ Efficiency Index (CEI) can be defined as follows, which are used for formally proving the presence or absence of true misses:

\[
FC = \frac{D + X}{D + X + W}
\]

\[
CEI = \frac{D}{D + W}
\]

Due to the fact that none of the checkers resulted in false positives, this information is excluded from the metrics.

IV. CHECKERS EVALUATION AND MINIMIZATION FLOW

The flow, which is part of the proposed framework, starts with synthesizing the checkers from a set of combinational assertions described in RTL Verilog by Synopsys Design Compiler [17]. Afterwards, the pseudo-combinational version of the circuit is created in which the sequential parts (such as flip-flops) are broken and changed into pseudo-primary inputs and pseudo-primary outputs. Additional checkers can be added later if the following evaluation stage points out some uncovered faults.

Thereafter, the exhaustive input patterns for the pseudo-combinational circuit is created, however a filtering tool will in the end extract only the set of valid patterns from the exhaustive set constituting the environment for the evaluation. After that, the set of valid input patterns, the pseudo-combinational circuit and the checkers are applied to fault free simulation. Undesired firing of the checkers at this point would indicate a bug in a checker or incorrect/invalid input test pattern. We have tried to show in the case study of a NoC router in Section 5 via experiments how such bugs in checkers are detected.

During fault simulation, the tool injects faults to all the lines within the circuit and it is repeated for each input vector. Based on that, the fault detection capability of the checkers will be computed using the metrics mentioned earlier in the paper, FC and CEI. Moreover, each checker is weighted by the number of times the checker has fired correctly (true detection). Finally, the weighting information will be utilized in order to minimize the number of checkers, eventually allowing to outline a trade-off between CEI, or FC, and the area overhead due to the introduction of checker logic.

The minimization 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 and fault modelling.

V. TARGET ARCHITECTURE: NOC ROUTER

Fig. 2 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 datapath and a control part. The datapath is composed of input buffers (implemented as First-In-First-Out (FIFO)), one for each input port, crossbar switch and output buffers, one for each output port.

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, therefore the logic can be easily re-configured. Routing decision is distributed and only requires local and destination addresses for forwarding flits. In this work we focus on a 2D Mesh topology, we consider XY as the routing algorithm, which is a deterministic 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. 3, for instance for the East input port and based on XY routing.

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. 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).

Based on the requests from the input ports, the internal state of the arbiter can change and it can give grant to the winner input port. Different encoding for the states of the Finite State Machine (FSM) of the arbiter may be chosen. In this work, they are encoded using one-hot coding. Moreover, one-hot encoding is extended to grant signals and select lines for the crossbar switch. This way, it would be
easier for the checkers to check if for example a fault occurs in the state register and it violates the one-hot rule. Also, note that the grant signals in arbiter also follow the one-hot rule, that is, during each arbitration, at most one grant signal to the inputs can be given by an arbiter.

The design decision to implement a one-hot encoded arbiter state machine versus a binary encoded one did increase the area of the arbiter, however, the CEI nearly doubled.

VI. APPLICATION OF THE FRAMEWORK TO DESIGN

As mentioned earlier, in the control part of the router, we have limited our focus to the case in which the LBDR (routing) and arbiter logic have the most number of connected signals, more specifically considering ELBDR and SArbiter. Such scenario provides the case with the most number of connections between LBDR and arbiter logic. The checkers that cover faults for such scenario, are symmetrical to the other cases, which may present a different number of connections due to routing algorithm and restrictions.

Our experiments via the checker evaluation tool showed that the two sets of checkers for ELBDR and SArbiter are independent, that means they cover faults for different and separate parts of the circuit, without any overlaps. Thus, this would cause the fault table to be partitioned into two clusters. First, the ELBDR logic is considered separately, afterwards, in the second step, the whole circuit consisting of the connection of ELBDR to SArbiter will be considered (as shown in Fig. 4).

The configuration bits of the ELBDR logic are set to fixed values based on the following scenario: XY routing algorithm, 180 degrees turns not allowed, and current address fixed to router with ID 5 in a 4x4 2D Mesh NoC. This scenario allows minimizing the number of circuit inputs and previous state pseudo-input bits to be considered in the experiments. When ELBDR and SArbiter interconnected are considered, the amount of inputs of the circuit under study would be as follows (19 bits):

- 2 flit type bits;
- 4 destination address bits;
- 4 ELBDR previous output values bits;
- 1 empty bit (coming from East input buffer);
- 3 SArbiter request signals bits;
- 5 SArbiter previous state bits.

This, in turn, makes the exhaustive approach in checker evaluation fully feasible.

After the pseudo-combinational version of the circuit is generated, a set of combinational checkers is devised, based on the functional behavior of the logic under study. These checkers might have connection to the inputs or outputs of the circuit. It is worth noting that it might be difficult to outline the effectiveness of a single checker or the overlap of different checkers in detection, a priori.

Together with the considered pseudo-combinational circuit and its set of checkers, a set of input patterns is needed for performing fault simulation. The exhaustive test for the case in which ELBDR and SArbiter are considered would consist of $2^{19}=524,288$ input stimuli. However, in order to minimize the stimuli, and more important, to avoid checkers being evaluated in non-realistic conditions, only a valid set of input stimuli should be considered and the rest of patterns are discarded, based on the considered routing algorithm, restriction of 180 degree turns, and emptiness condition of the input buffer (FIFO), and also based on the invalid conditions for the state variable of the arbitration unit (such as violation of one-hot encoding). It is worth emphasizing that in our experiments (in Section 7) none of the checkers fired in fault-free simulation under the set of exhaustive valid input stimuli.

VII. EXPERIMENTAL RESULTS

As mentioned in the previous section, in order to evaluate the minimization framework, we did two types of experiments both focusing on the control part of an NoC router, first one emphasizing only on the routing logic (ELBDR) and the second one interconnecting the routing logic to an arbiter unit (SArbiter) (shown in Fig. 4).

In both cases an initial set of checkers was devised a priori, together with a filtering scheme to obtain a valid set of input stimuli for the circuit. Each individual checker was weighted by the tool by summing up the total number of true detections by the checker, and this information was used in a heuristic way to minimize the initial set of checkers, with the final aim of achieving highest possible CEI and FC, and
at the same time with the lowest possible area overhead. These quantities were evaluated iterating the fault simulation, including at each step the next heaviest checker still not included in the currently considered set of checkers, initialized only with the first heaviest checker.

The experiment on ELBDR alone led to a minimized set of 3 checkers, ensuring maximal coverage in the routing logic.

**ELBDR + SArbiter scenario experiment**

In this experiment, ELBDR is connected to SArbiter according to Fig. 4, thus providing the East request signal to the South arbitrating logic of the NoC router. For this scenario, initially a set consisting of 28 checkers were devised for the SArbiter, while taking into consideration the minimized set of 3 checkers for ELBDR.

The exhaustive test for the considered pseudo-combinational circuit (ELBDR+SArbiter), would require $2^{19} = 524,288$ input stimuli as the circuit has 19 inputs. The considered filtering scheme for this experiment provides the set of valid input patterns for the circuit based on the following rules that must hold for the routing and arbitration logics:

- if input buffer empty signal is high, any other input bit in ELBDR is meaningless and therefore, any value is allowed for it;
- if the incoming flit to ELBDR is a header, the destination address has to be valid according to the XY routing and turn restrictions;
- if the incoming flit to ELBDR is a body or tail flit, the previous output values must be valid and they must follow a one-hot fashion, according to XY routing;
- the state variable of SArbiter must always follow the one-hot fashion.

The above-mentioned rules, led to the reduction of the initial set of $2^{19}$ input stimuli for the circuit to shrink to a valid and complete set consisting of 61,440 input vectors, which is less that 12% of the initial number. This may be considered as a proof of the effectiveness of the one-hot encoding for the arbiter state variable, which will be later noticeable for its detection capability too.

In the first place the evaluation tool is run considering the whole set of checkers for the SArbiter, altogether with the minimized set of 3 checkers for the ELBDR. Figure 8 lists the considered 31 checkers, with their corresponding weights in a descending order. Focusing on the arbitrating unit, two checkers look to be far more significant than the others, $Serr_{validgrant}$, $Serr_{invalidstate}$, both of them monitoring, in a different manner, the one-hot encoding for the arbiter state variable, which will be later noticeable for its detection capability too.

From the output of the evaluation tool it can be observed that the two set of checkers for the ELBDR and the SArbiter are independent, i.e. they cover faults for different and separate parts of the circuit, without any overlap. For this reason the minimized set of ELBDR checkers is used and the previously introduced weight-based greedy minimization heuristic is applied to the SArbiter checkers set.

Impact of clustering the faults

Assuming that we had no information of the overlap of faults detected by the checkers for ELBDR and SArbiter, the weight-based greedy heuristic, starting from the heaviest checker $Serr_{validgrant}$, would add at each step the next heaviest checker still not considered in the current set of checkers, based on the weight information displayed in Fig. 5. Fig. 7 shows the inefficiency of the heuristic approach caused by the lack of the clustering information. The number of steps in the greedy procedure is heavily increased, and only after 19 steps, when the $Err_{singleLBDRout}$ checker is considered, the 100% upper bound for CEI and FC is reached.

However, when partitioning of the fault set to clusters is taken into account and minimization is performed on the clusters separately then total of five steps are needed. Fig. 8 illustrates the importance of considering the clustering information. It can be observed that the weights of the ELBDR checkers are far less than those of the SArbiter, but they are still needed to achieve full coverage for the considered design.
The paper proposed an automated flow for evaluation and minimization 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. Experiments on the control part (routing and arbitration) of a Network-on-Chip (NoC) router carried out showed on a realistic application the feasibility and efficiency of the framework and the underlying methodology. Experimental results showed that the approach allowed selecting the minimal set of 5 checkers out of 31 verification assertions with the fault coverage of 100% and area overhead of only 56.82%.

ACKNOWLEDGEMENT

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


