Effective Abstraction for Response Proof of Communication Fabrics

Sayak Ray  
Dept. of Electrical Engineering  
Princeton University, USA  
sayakr@princeton.edu

Sharad Malik  
Dept. of Electrical Engineering  
Princeton University, USA  
sharad@princeton.edu

I. INTRODUCTION

Formal verification of liveness properties is a critical step in the design flow of communication fabrics as these micro-architectural modules are prone to liveness bugs like deadlock and livelock. Liveness properties, however, are challenging to prove for automated techniques such as model checking. This makes system-specific verification heuristics and designers’ insights about systems critical for scaling liveness verification algorithms to industrially relevant large designs. Recently, one such novel insight about the operations of communication fabrics, viz. ranking structures, has been proposed [1]. It has been shown that correct operation of a communication fabric can be arranged in a monotone sequence of ‘fairness events’ which gives rise to a sequence of ‘ranked regions’ in its state space and this can lead to a scalable proof of the fabric’s ‘responsiveness’. However, efficient algorithmic discovery of the ranking structure from a structural description of a communication fabric is still an open question. Towards answering this question, we present an algorithm based on iterative discovery of satisfiability backbones that efficiently mines the ranking structure from a bit-level netlist representation of a communication fabric. To the best of our knowledge, our work is the first in applying satisfiability backbone analysis technique in ranking structure abstraction as well as in liveness verification. In this extended abstract, we present the core notion of backbone-based formulation of ranking structure and our experimental observations. In our experiments, we used our algorithm to prove response property of a collection of Executable Micro-Architecture Specification (xMAS) [2] benchmarks. Both the xMAS formalism and the response property are described briefly below. Section II then presents our backbone-based formulation and the resulting mining algorithm. Further details are available in a longer version of the paper [3].

Communication fabrics are typically constructed using structural primitives like finite FIFO buffers, sources and sinks of flits ¹, function blocks acting on flits, decision primitives like switches and arbiters, and synchronization primitives like forks and joins. xMAS is an effective formalism for modeling communication fabrics and it provides precise formal definitions

for a selected set of such structural primitives [2]. A suitable compiler can translate a high-level description of an xMAS fabric into a synchronous sequential Boolean circuit that works under single global clock. The experiments reported in Table I are performed on a collection of such xMAS fabrics taken from the literature [2], [5], [6]. Our benchmark suite consists of a basic virtual channel (VC), virtual channels with channel buffers (VCB₂, VCB₃) and a virtual channel with ordering logic (VCO).

Informally, the response property specifies that any data transfer request will be eventually granted by a fabric under appropriate fairness assumptions. Suppose our target xMAS fabric has channels a, i₁, …, iₙ. Our objective is to prove that channel a satisfies the response property under the strong fairness assumptions [1] on channels i₁, …, iₙ. Formally, this property is described by the following linear temporal logic (LTL) formula: fairness ⇒ response(a)

where fairness := ∨ n k=0 GF(iₖ.gnt) and response(a) := G(a.req ⇒ F(a.gnt)). G and F are standard LTL operators representing temporal modalities ‘always’ and ‘eventually’ respectively. Response property is a well-known liveness property and is among the most-frequently model-checked liveness properties for communication fabrics. It is directly related to deadlock freedom of the underlying communication fabric. While in theory we can invoke any LTL liveness model checker to prove this property, but as argued in [1], reasoning with ranking structures can expedite this proof process substantially.

The technique presented in this paper is a proof technique. It manages to discover a ranking structure if one exists in the given fabric. If the fabric has any response bug, it cannot have a ranking structure in its state space. In that case, our algorithm will not be able to predict the absence of ranking structure and will potentially go into an infinite loop. We note that there are scalable techniques, mostly based on bounded model checking (BMC), to find response bugs in incorrect designs. While BMC offers scalability for discovering bugs, it cannot prove a property. In order to develop a complete verification procedure, our proof technique can be interleaved

¹A flit (flow-control unit) is a unit of data transfer in communication fabrics, see [4] for details.

²Here channel refers to a bundle of signals that xMAS formalism uses to connect two structural primitives.
with any such BMC based bug discovery technique, as done in other state-of-the-art tools [7].

II. RANKING STRUCTURE THROUGH BACKBONES

For a design satisfying the response property, its state space can be divided into regions that form a directed acyclic graph. These regions are called ranking regions. In general, a ranking region is defined as a Boolean formula \( \sigma \) over a set of predicates defined over design signals. As observed in [1], definition of a ranking region may involve determining predicates that are not present in the design netlist. However, our close scrutiny of the xMAS library reveals that some xMAS primitive signals present in the synthesized fabrics can often be used (directly) to define ranking regions instead of using externally introduced predicates. Therefore, while [1] argued that user should add additional predicates to the design as the basis for disjunctive stabilizing constraint discovery, we claim that an adequate basis set of signals is already present in most of the fabrics and the ranking regions can be represented as conjunctions of some of them. Toward this goal, we observe that if a ranking region \( \sigma \) is represented as \( \wedge_{i=1}^{k} l_i \) where \( l_1, \ldots, l_k \) are fabric signals with appropriate polarity, \( \wedge_{i=1}^{k} l_i \) essentially behaves as a satisfiability backbone if evaluated on any concrete state in \( \sigma \).

In order to discover backbone-based descriptions of the predicates \( \sigma_i \)'s, we invoke safety model checking and backbone analysis in an iterative and interleaved manner. A model checker is invoked to sample an adequate number of candidate concrete states from a region \( \sigma_i \) for \( i \geq 1 \). Then backbone analysis is performed on those concrete states and the fabric signals to produce a backbone-based description for \( \sigma_i \). With the description of \( \sigma_1, \sigma_2, \ldots, \sigma_i \) thus learned, we repeat the rounds of model checking and backbone analysis to learn \( \sigma_{i+1} \). We stop when we exhaustively partition the state space into a finite number of \( \sigma_i \)'s. This termination criterion makes our algorithm complete modulo the existence of a linear sequence of backbone-based predicates \( \sigma_i \)'s. We leverage the dual capability of contemporary safety model checkers like IC3 [8] that they can generate counter-examples to a safety property if the property is violated as well as can prove that no counter-example exists if the property is satisfied. We use the counter-example generation capability to sample candidate concrete states from \( \sigma_i \)'s and the proof capability to establish that some region \( \sigma_i \) has been sampled exhaustively before exploring \( \sigma_{i+1} \).

III. EXPERIMENTAL RESULTS

ABC [9] is used as the verification platform in our experiments. Our backbone-based algorithm is implemented in ABC as command bbd and its performance is compared against ABC’s implementation of response verification algorithm based on ranking structure discovery through disjunctive stabilizing constraint mining (available as command kcs). Our main experimental result is summarized in Table I. The experiments are performed on a laptop with 4 Intel(R) Core(TM) i5-3320M 2.6GHz cores (each core having 3MB cache) and 3.6GB RAM that runs the Ubuntu 12.04 LTS operating system. Run-times are reported in seconds. Since the correct depth of disjunction in command kcs is not known a priori, experiments are performed with depth 2 (column 4) and depth 3 (column 5). Runtime of command bbd is broken down into two parts: backbone mining (column 6) and reachability analysis (column 7). As evident from the overall runtime of bbd (Column 8), bbd outperforms kcs in most of the cases.

<table>
<thead>
<tr>
<th>D</th>
<th>PL</th>
<th>t/l</th>
<th>kcs -C (2)</th>
<th>kcs -C (3)</th>
<th>bb mining</th>
<th>verif</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC</td>
<td>18</td>
<td>52</td>
<td>0.43</td>
<td>1.35</td>
<td>0.18</td>
<td>0.02</td>
<td>0.2</td>
</tr>
<tr>
<td>VC,B2</td>
<td>18</td>
<td>68</td>
<td>132.78</td>
<td>1421.25</td>
<td>11.91</td>
<td>1.02</td>
<td>12.93</td>
</tr>
<tr>
<td>VC,B3</td>
<td>18</td>
<td>92</td>
<td>TO</td>
<td>TO</td>
<td>8.83</td>
<td>43.23</td>
<td>52.06</td>
</tr>
<tr>
<td>VCO</td>
<td>20</td>
<td>69</td>
<td>5.96</td>
<td>8.72</td>
<td>8.51</td>
<td>2.65</td>
<td>11.16</td>
</tr>
</tbody>
</table>

TABLE I
COMPARISON OF RUN-TIME BETWEEN THE PROPOSED ALGORITHM AND THE BASE-LINE ALGORITHM

IV. CONCLUSION

We present a satisfiability backbone-based formulation for ranking structure discovery and thereby present an alternative scalable proof technique for the response properties. Our algorithm offers enhanced automation by reducing the need for user supplied input information compared to the known technique for ranking structure discovery [1]. We demonstrate that backbone based response verification algorithm scales up or attains comparable scalability without user supplied safety invariants.

V. ACKNOWLEDGEMENT

This work was supported in part by C-FAR, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA. Sayak Ray thanks Robert K. Brayton and Alan Mishchenko for many helpful discussions.

REFERENCES

[9] "Abc verification system: http://www.eecs.berkeley.edu/~alanmi/abc/;"