A SAT-based Scheduling Framework for Multi-Processor Systems on Chips

Christian Schöler∗ René Krenz-Bååth† Roman Obermaisser†
∗Hamm-Lippstadt - University of Applied Sciences
christian.schoeler, rene.krenz-baath}@hshl.de
†University of Siegen
roman.obermaisser@uni-siegen.de

Abstract—This thesis contributes to the field of design, implemen-
tation and verification of Multi-Processor System-on-a-Chip (MPSoC) architectures realizing embedded real-time systems. It is based on three pillars: At first the development and implementation of a Boolean Satisfiability (SAT)-based scheduler is described. Secondly the generation of benchmarks arising from scheduling problems on Time-Triggered-Networks-on-a-Chips (TT-NoCs) is outlined and it is explained how Satisfiability Modulo Theories (SMT)-solver can be applied. Finally we report on the evaluation of performance and scalability of the proposed scheduler on an MPSoC emulating target system.

I. INTRODUCTION

Formal verification techniques have become an important part in modern design flows. Techniques such as model checking [1], assertion-based verification [2] or constrained random simulation [3] have been successfully applied to processor designs or protocol verification as well as complex systems on a chip (SoC). The main purpose of this work is the development of a verification framework tailored to meet the tremendous complexity of evolving MPSoC architectures and their specific architectural properties.

In an initial step the characteristics and requirements of the systems under verification are translated into a SAT problem and solutions, for instance a feasible configuration of a TT-NoC that meets all constraints, are computed using a set of well established SAT-solvers. Among those are MiniSat v1.13 [4] and MiniSat v2.1, a largely incremental update introducing a number of data structure improvements including CNF based pre-processing [5]. Furthermore we use the Integer-Linear-Programming (ILP)-Solver MiniSat+ [6], which we used to solve several MPSoC-related optimization problems. Additionally we will also investigate the impact of solving scheduling problems applying Satisfiability Modulo Theories (SMT) solvers such as Yices 2.2 [7] to the benchmarks arising from scheduling problems in TT-NoCs.

The main contribution of the developed framework is its ability to be efficiently executed on embedded multi-core platforms. Thus it allows online verification and can also be used in reconfigurable systems where necessary safety constraints must be proven before reconfiguration.

In order to deploy the proposed framework on MPSoC architectures we propound different strategies to parallelize the problem solving in order to account for limited memory and processor resources of the target system. We are able to solve scheduling problems on the target system and can provide an optimal communication schedule for message injection. The framework is in a final step analysed with respect to runtime and scalability on an MPSoC emulating system.

This paper is structured as follows: in section II the research objectives and the novelty of the approach are explained. Section III gives an overview of the current status, reports initial results and describes the challenges faced. Finally section IV illustrates some of the possible applications of our verification framework.

II. RESEARCH OBJECTIVES AND APPROACH

The presented thesis is based on three pillars: implementation of a parallelized SAT or SMT based scheduler exploiting different cores of a multi-core processor, generation of benchmarks from TT-NoCs and finally evaluation of runtime and scalability of parallelized SMT/SAT solving in comparison to state-of-the-art schedulers such as CPLEX.

First of all the formulation of the problem as a SAT problem is considered and a conjunctive normal form (CNF) denoted \( \Phi \) is introduced. As it is aspired to exploit multiple resources and cores for parallelization, various approaches to distribute the SAT-problem arising from the necessities of the system under consideration to several cores are discussed. Since we are describing CNF-decomposition techniques we will denote the resulting CNF-partitions \( \phi_1 \cdots \phi_n \) such that \( \forall \phi_i \in \{1, \cdots, n\} : \phi_i \subseteq \Phi \).

The challenge consists of two specific problems: Firstly it has been shown that achieving a robust runtime improvement over a large set of hard SAT problems by applying any form of distributed SAT remains an unsolved problem. Secondly running distributed SAT solvers on MPSoC architectures, containing a large number of computing cores and providing only a limited communication bandwidth between them, leads to significant limitations for example with respect to knowledge exchange during the solving process and the scalability of the reasoning engine. These aspects are crucial in order to enable online verification capabilities for the system under consideration.

In the proposed framework the CNF partitions \( \phi_1 \cdots \phi_n \) are analysed separately and the knowledge obtained by the concurrent diagnosis of conflicts for all \( \phi_i \) is used to speed-up a final SAT run of the original problem instance. A detailed discussion of strategies tested with industrial benchmarks from the SAT competition 2013 [8] prompted the need to generate benchmarks from the area of multi-core design applications.

Afterwards the SAT-based verification framework was in a second step applied to scheduling problems, as a growing number of embedded systems is deployed with multi-core platforms where processor cores are interconnected by networks-on-a-chip. TT-NoCs are considered in the scheduling problem, because these networks exhibit significant advantages.
with respect to predictability and dependability in safety-relevant applications [9]. For these NoCs it is necessary to compute a communication schedule that determines for each message the points in time for the injection at the network interface as well as conflict-free paths through the network-on-a-chip. We propose a time-discrete model allowing us to map the scheduling problem into a pseudo-Boolean SAT problem. Thus we were able to introduce an optimal scheduler based on a Boolean SAT solver for TT-NoCs [10] enabling us to calculate optimal solutions to scheduling problems finding the fastest way to deliver all messages required. The third pillar comprises evaluation of performance and scalability of our implementations on an MPSoC emulating platform. For this we have chosen a cluster consisting of six Raspberry Pis 700 MHz ARM1176JZF-S and one switch interconnecting the nodes to emulate a multi-core platform with limited memory and processor resources.

### III. CURRENT WORK AND PRELIMINARY RESULTS

This section will briefly outline the current status of the three pillars introduced: For the first two pillars we have so far implemented a SAT-based verification framework which can be parallelized to exploit multiple resources in a network. After initially testing various techniques to parallelize SAT instances using large industrial benchmarks from the 2013 SAT competition we have subsequently generated benchmarks arising from TT-NoC scheduling problems.

In a first step we have evaluated the solutions to these problems and compared our results to an optimal scheduler using Mixed Integer Linear Programming (MILP) implemented using CPLEX. The comparison between SAT and CPLEX models was based on random example scenarios generated using the Stanford Network Analyzer Platform library (SNAP) [11]. The results of this work are listed in Table I. The scenarios examined distinguished between the number of routers (RS) and cores (CS) and specify the number of jobs that have to be performed as well as the corresponding messages. Furthermore we have implemented logical constraints to model temporal dependencies of the jobs. To simplify the model jobs are in this phase assigned to nodes as a constant. The examples show on the one hand that the SAT-based scheduler is not only able to reproduce the results from CPLEX in terms of finding optimal transmission times but furthermore can significantly decrease the required runtime. For comparison the results were obtained using CPLEX 12.6.1 and MiniSat+ 1.4 running on a 12 processor Intel(R) Xeon(R), 2.2 GHz server with the operating system Linux Ubuntu 14.04.1.

In a second phase we have increased complexity by increasing the the number of cores, routers, jobs and messages in the system under verification. Simultaneously we have allowed an arbitrary allocation of jobs making the scenarios more realistic. In order to satisfy the rapidly growing demands with respect to the number of variables and constraints the framework has been extended to apply SMT solvers allowing us to formulate a model with fewer restrictions. Furthermore SMT enables us to deploy MILP, which results in a more compact representation of the problem. In this context we have identified the SMT solver Yices 2.3 [7] to be ideal for our purposes.

As expected, however, the complexity of the scheduling problem and the subsequent application of a first-order logic SMT solver increased the runtime of our framework. Therefore our main focus is currently on the third pillar evaluating the results on the target system and introducing parallelization techniques to speed up the SAT approach. For this purpose we have developed an incremental SMT-based scheduler which has been evaluated using different approaches to compute solutions to the scheduling problem in parallel [12].

### IV. FUTURE WORK

We are currently extending the proposed framework to compute scheduling tables for embedded systems at runtime enabling tasks to be shifted to other nodes if a failure occurs hence increasing safety of the system. Since fast computation times for valid schedules are vital for this purpose we will analyze in more detail how the proposed SMT-based scheduler will perform on the Raspberry Pi cluster described in Section III. In this context we focus on different techniques to parallelize the scheduling problem and to make the presented framework more efficient.

### REFERENCES


### TABLE I

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Physical Model</th>
<th>Logical Model</th>
<th>CPLEX</th>
<th>Minisat</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS</td>
<td>CS</td>
<td>Jobs</td>
<td>runtime (s)</td>
<td>constraints</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>0.014</td>
<td>1.040</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>0.014</td>
<td>1.040</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0.014</td>
<td>1.040</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>0.014</td>
<td>1.040</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>3</td>
<td>0.014</td>
<td>1.040</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>3</td>
<td>0.014</td>
<td>1.040</td>
</tr>
</tbody>
</table>

*This table shows the results of 9 example scenarios, comparing the runtime and number of constraints between CPLEX and Minisat.*