A BMC-based formulation for the scheduling problem of hardware systems

Gianpiero Cabodi1, Alex Kondratyev2, Luciano Lavagno1,2, Sergio Nocco1, Stefano Quer1, Yosinori Watanabe2

1 Politecnico di Torino, Dip. di Automatica e Informatica, Turin, Italy
e-mail: stefano.quer@polito.it
2 Cadence Design Systems, Inc., Berkeley, CA, USA

Published online: 25 January 2005 – © Springer-Verlag 2005

Abstract. Hardware scheduling is a well-known and well-studied problem. This paper defines a new SAT-based formulation of automata-based scheduling and proposes for the first time a completely new resolution algorithm based on SAT solvers and bounded model checking (BMC).

The new formulation is specifically suited to control-dominated applications. Alternative executions are modeled as concurrency, where alternative behaviors are followed in parallel. This approach produces “single-path” scheduling traces instead of standard “treelike” solutions, thus enabling the use of BMC. This choice, however, creates the problem that resource bounds are treated incorrectly, due to the artificial concurrency modeling alternative behaviors. We then discuss how to take this into account, either by modifying the SAT solver or by adding extra clauses. Thus we are able to exploit SAT-based BMC to find the desired minimum latency schedule.

Our method shows significant improvements in terms of both computational efficiency and modeling power, when compared to the BDD-based approach, and in terms of the optimality of the results when compared to heuristic methods.

Keywords: Binary decision diagrams – Satisfiability solvers – Bounded model checking – Hardware scheduling

1 Introduction

Synthesis of efficient and high-performance control units and data paths from high-level behavioral specifications has long been considered a very promising technique for tackling the ever growing complexity of digital design. At the same time, it is a very elusive goal, because after more than 20 years of intensive research and despite the appearance on the market of some industrial CAD tools, high-level synthesis is still far from being as widely used as its predecessors, register-transfer level and logic synthesis.

Within this framework, symbolic BDD-based manipulations have recently attained interesting results as an alternative to ILP and heuristic techniques. The key idea of the symbolic approach of [9–11, 16] is to use a set of nondeterministic finite automata to describe design alternatives for highly constrained control-dominated models, in which complex if-then-else patterns constitute the body of kernel loops. In this approach, the automata state space is symbolically visited adopting state-of-the-art model checking techniques: forward and backward traversals find a scheduling solution as a trace connecting initial and terminal states.

In the simplest case of systems without control choices (i.e., if-then-else constructs), a schedule is a path, and symbolic scheduling works just like invariant checking with counterexample extraction. However, control-dependent behavior produces scheduling instances as directed acyclic graphs (DAGs) or trees, where fork and join nodes are introduced to represent scheduling choices, depending on values of control operands. This requires a specific backward traversal procedure (called validation in [10]), which, though not far from standard BDD-based traversals, does not directly correspond to standard model checking (e.g., CTL) procedures.

In this work, we propose to change the original automaton model introduced in [9–11, 16] for control-dependent systems, so that standard model checking procedures are supported. More specifically, we transform alternative subtraces to concurrent behaviors, which are followed in parallel. In this way the resulting scheduling is always a path (instead of a DAG) connecting initial and final states. As a byproduct, we can exploit SAT-based bounded model checking (BMC). BMC is indeed a very promising technique in this framework as the designer’s aim is to find a schedule, not to prove its

1 For the sake of simplicity, here we only consider acyclic specifications and discuss handling of loops in Sect. 6.
absence, and a schedule is always found given enough
time. In fact, BMC is a verification technique mainly ori-
tented toward bug hunting and counterexample extraction
rather than proof of correctness. Nevertheless, in
order to enable this method, we also must rewrite the
resource bounds so that they take into account the artifi-
cial concurrency introduced to model mutually exclusive
behaviors.

As far as we know, this is the first time such a model
has been proposed and SAT-based BMC applied effec-
tively to the scheduling problem.

As a final remark, note that many high-level synthesis
tools use control data flow graphs (CDFGs) as their in-
ternal model. While data dependencies are usually easily
handled, control is either ignored or handled by complete
case splitting. Constraints coming from input/output
operations with the external world (e.g., synchronization,
min/max rate, jitter, etc.) are often poorly handled by
such tools.

Although we use CDFGs as the input specification for
our tool, we adopt the model introduced by [10], which is
at the same time:

– **Formal**: it is based on concurrent automata.
– **Efficient**: it is possible to use symbolic representation
  techniques with enhancements derived from concur-
  rent specification models.
– **Control-oriented**: condition evaluation and specula-
  tive execution are specifically handled.
– **Flexible**: I/O constraints can be represented by re-
  strictions on the automata state space.

As in [10], we represent implicitly the full solution space
by means of the state space of a product of automata.

We compare our methodology with a state-of-the art
BDD-based technique on a set of standard high-level syn-
thesis benchmarks. We obtain a significant improvement
in terms of both computational efficiency and modeling
power.

2 Background

We assume that the reader is familiar with BDDs, SAT,
and BMC. As a consequence, we briefly review only
the basic concepts that are relevant for our application
framework.

2.1 High-level synthesis methodologies

Scheduling is a key step during the design of a digital
system. Throughout a manual or automatic process, a
designer decides when computational operations and com-
munication transactions take place. Such decisions are
constrained by hardware resource availability, operand
dependencies, and control decisions.

The following example introduces the scheduling
problem in a simple case involving only data operations
and no decision.

**Example 1.** Let us start from the pseudocode of Fig. 1a.
We start by representing the sequence of operations by
the data flow graph (DFG) shown in Fig. 1b. Once this is
done, it is evident that several scheduling solutions can be
found for it, depending on:

– The type of resources used for each operation, e.g., an
  ADDER or an ALU for an addition.
– The number of resources allocated for each type.
– The choice of where to put registers to hold the results
  of previous operations.

Figure 2 and Table 1 show some of the possible
scheduling instances, with different combinational re-
source and register allocations. In particular, solutions
(a) to (d) allocate exactly one register for each combi-
national operation, whereas solutions (e) and (f) allow
combinational propagation of data, thus requiring fewer
registers and cycles (traded off by a possibly longer cycle

2 Approaches specifically addressing control-intensive CDFGs
(such as [8, 13, 17]) have been introduced only recently.