Formal Verification of Synchronization Issue of System-Level Design with Automatic Abstraction

Thanyapat Sakunkonchak and Masahiro Fujita
Department of Electronic Engineering, The University of Tokyo

Introduction

Our target verification for synchronization is based on “SpecC Language”, since SpecC
- is designed to support the SoC design
- can describe both HW and SW
- is promising the C-based specification/design/implementation
  [http://www.SpecC.org]

Objectives

- Construct a framework that can verify for the event synchronizations
- If the verification fails, the counter-example (the error path) must be automatically generated

This is based on the following concepts and tools:
- Model checking technique
- Boolean Program (Microsoft Research)
- Difference Decision Diagrams (DDDs) (IT University of Copenhagen)
- Cooperating Validity Checker (CVC) (Stanford University)

Boolean Program

- Proposed by Ball and Rajamani under SLAM project at Microsoft Research
- Boolean program abstracts irrelevant statements such that the transformed program contains only boolean variables
- Since boolean program abstracts the original one: some properties that hold in boolean program must eventually hold in the original program
- The verification process is completely automatic, with counter-example provided when the properties to be checked are failed

Difference Decision Diagrams

- Introduced by Möller, et.al. (IT University of Copenhagen)
- Symbolic representation of ‘non-boolean’, e.g. inequalities: less efficient if using BDDs

Represents graph for
\[ (x \leq 1) \cup (x \geq 0) \cup (y \leq 2) \]

Synchronization in SpecC Language

\[
\begin{align*}
\text{main}(i) & \text{ par } a.main(); \\
\text{behavior a} & \text{ main}() \{ a \geq 10; \quad /\ast \text{time} \geq 1 \ast/ \} \\
\text{behavior b} & \text{ main}() \{ a \geq 10; \quad /\ast \text{time} \geq 1 \ast/ \}
\end{align*}
\]

\[
\begin{align*}
\text{main}(i) & \text{ par } a.main(); \\
\text{behavior a} & \text{ main}() \{ a \geq 10; \quad /\ast \text{time} \geq 1 \ast/ \} \\
\text{behavior b} & \text{ main}() \{ a \geq 10; \quad /\ast \text{time} \geq 1 \ast/ \}
\end{align*}
\]

We use the timing constraints, e.g. \( T_{as} \leq T_{1s} \leq T_{2s} \leq T_{ae} \leq T_{ae} \), as our criteria for verification. And these series of inequalities can be represented using DDDs

Verification Flows (Both graphical representation and Pseudo code)

Conclusion and Outlook

Current Implementation
- can handle the basic SpecC constructs
- can verify the SpecC code that contains synchronization semantics
- the properties can be checked by simply check for the assertion violation (reachability problem)
- when the properties are not satisfied, the counter-example can be generated

Future plan
- the entire processes, from abstraction to generation of counter-example, are still not completely automatic (need to assemble all parts together and make them automatically running)
- upon the completion of making the entire flows running automatic, we will try to verify with some real implementations