Bridging the Gap: Consistent SystemC Code Generation for Multi-Abstraction-Level State Charts

Rainer Findenig, Wolfgang Pauli, Thomas Leitner, Wolfgang Ecker

wolfgang.pauli@intel.com
Agenda

1. Single Source Modeling and Code Generation
2. Ensuring Consistency
3. Results
4. Summary
Code Generation

- Single source, multiple abstraction levels

- IP-XACT

- IF

- Model

- Templates

- Python-based code generator

- Abstraction Level Generation Parameters

- Selected abstraction level or cosimulation

- TLM & CC Cosim

- VHDL

- RTL

- SVA

- SystemC TLM & CC
Single Source, Multiple Abstraction Levels

User View: UML profile

- Integrate with standard UML
- Support multiple abstraction levels for behavior and interface
- Allow consistent modeling style
- Reference interface definition

Different abstraction levels can be mixed
# Advantages

<table>
<thead>
<tr>
<th>Model Reuse</th>
<th>Consistent Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Don’t throw away high-level models</td>
<td></td>
</tr>
<tr>
<td>• Reuse and refine</td>
<td></td>
</tr>
<tr>
<td>• Minimize need to modify several models</td>
<td></td>
</tr>
<tr>
<td>• Make corresponding parts visible</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mix and Match</th>
<th>Verification Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Different applications have different requirements</td>
<td></td>
</tr>
<tr>
<td>• Allow mixing different abstraction levels in a single model</td>
<td></td>
</tr>
<tr>
<td>• Ease verification of refinement</td>
<td></td>
</tr>
<tr>
<td>• Localize errors easier and earlier</td>
<td></td>
</tr>
</tbody>
</table>
Agenda

1. Single Source Modeling and Code Generation
2. Ensuring Consistency
3. Results
4. Summary
Consistency under Refinement

- Approach allows arbitrary code in the State Chart
  - Rules out existing static consistency approaches

- Cosimulation to check two abstraction levels:
  - Compare refinement to original
  - Use scoreboards to keep track of consistency
  - Currently no automation of testbench
Consistency under Refinement: States

State Refinement

• Liskov Substitution Principle
• Undecidable in general
• Cosimulation based on refinement borders
• Automatic scoreboard generation allows tests during cosimulation

Scoreboard for refining state 1 by state 2

Scoreboard for refining state 2.2 by state 3
Consistency under Refinement: Transitions

Transition Refinement

- Behavior: Liskov Substitution Principle
- State trajectory: additional scoreboard required

Scoreboard for effect

Scoreboard for state trajectory
Agenda

1. Single Source Modeling and Code Generation
2. Ensuring Consistency
3. Results
4. Summary
Code Generation: Module Structure

- Module is generated automatically
- Only transactions are defined externally (by deriving)
- Blue elements: Not used in cycle callable model

### Code Generation: Module Structure

- **Transactions**
  - SC_METHODs for input signals
  - SC_THREADS for State Chart's functionality
  - State Chart implementation
Results: AES Encryption Core

- Performance, normalized to highest
- Generated lines of code for model

Abstraction Levels
A1 OpenSSL (3rd party library)
A2 A1 with timing
A3 Software model, timed
A4 A3, AT
A5 partly CC
A6 CC

A1-4 Transaction IF
A5-6 Pin-true IF
Results: AES Encryption Core

- Elements on the different abstraction levels

![Bar chart showing states and transitions across different abstraction levels]

- Structure of the generated code

![Bar chart showing classes, SC_METHODs, SC_THREADS across different abstraction levels]

Abstraction Levels
A1 OpenSSL (3rd party library)
A2 A1 with timing
A3 Software model, timed
A4 A3, AT
A5 partly CC
A6 CC

A1-4 Transaction IF
A5-6 Pin-true IF
Results: AES Encryption Core

- Number of scoreboards generated for checking consistency in cosimulation

<table>
<thead>
<tr>
<th></th>
<th>Original Model</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A1</td>
<td>A2</td>
</tr>
<tr>
<td>A2</td>
<td>1 0 0 1</td>
<td>- - - -</td>
</tr>
<tr>
<td>A3</td>
<td>1 0 0 1</td>
<td>1 0 0 1</td>
</tr>
<tr>
<td>A4</td>
<td>1 2 1 4</td>
<td>1 2 1 4</td>
</tr>
<tr>
<td>A5</td>
<td>1 2 1 4</td>
<td>1 2 1 4</td>
</tr>
</tbody>
</table>

**S, T, R, Σ**: # of Scoreboards for States, Transitions, Regions, and total

**Abstraction Levels**
- A1 OpenSSL (3rd party library)
- A2 A1 with timing
- A3 Software model, timed
- A4 A3, AT
- A5 partly CC
- A6 CC

A1-4 Transaction IF
A5-6 Pin-true IF
Results: MIPS Core

- Performance, normalized to highest

  ![Bar chart showing performance comparison between abstraction levels A1, A2, A3, and A4.]

- Generated lines of code for model

  ![Bar chart showing generated lines of code for each abstraction level and the model.]

Abstraction Levels

- A1 Untimed
- A2 Untimed with hierarchy
- A3 AT
- A4 CC

A1-3 Transaction IF
A4 Pin-true IF

MIPS is © MIPS Technologies.
Results: MIPS Core

- Elements on the different abstraction levels

- Structure of the generated code
## Results: MIPS Core

- Number of scoreboards generated for checking consistency in cosimulation

### Abstraction Levels

<table>
<thead>
<tr>
<th>Abstraction Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1  Untimed</td>
</tr>
<tr>
<td>A2  Untimed with hierarchy</td>
</tr>
<tr>
<td>A3  AT</td>
</tr>
<tr>
<td>A4  CC</td>
</tr>
<tr>
<td>A1-3  Transaction IF</td>
</tr>
<tr>
<td>A4   Pin-true IF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Refined Model</th>
<th>Original Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A1</td>
</tr>
<tr>
<td></td>
<td>S</td>
</tr>
<tr>
<td>A2</td>
<td>1</td>
</tr>
<tr>
<td>A3</td>
<td>1</td>
</tr>
<tr>
<td>A4</td>
<td>1</td>
</tr>
</tbody>
</table>

**S, T, R, Σ:** # of Scoreboards for States, Transitions, Regions, and total
Agenda

1. Single Source Modeling and Code Generation
2. Ensuring Consistency
3. Results
4. Summary
Summary

• We presented
  • a single-source modeling approach
  • based on UML for behavior specification
  • and a code generation framework that allows
  • generating a required abstraction level or
  • generating a cosimulation framework for
  • consistency checking.

• Further work:
  • More static code checking
  • Model optimizations to further improve performance
Thank you!