Automatic Generation of Cycle Accurate and Cycle Count Accurate Transaction Level Bus Models from a Formal Model

Chen Kang Lo, Ren Song Tsay,
Logos Advanced System Lab,
National Tsing-Hua University, Hsinchu, Taiwan
Outline

- Introduction
- Related Work
- Problem Formulation
- Transaction Level Bus Model Generation
- Experimental Results
- Conclusion and Future Work
Introduction to TLM

- Transaction Level Modeling (TLM) is proposed to perform architecture exploration and verification.

<table>
<thead>
<tr>
<th>Level</th>
<th>Protocol</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>TL3</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>TL2</td>
<td>approximate</td>
<td></td>
</tr>
<tr>
<td>TL1</td>
<td>cycle accurate</td>
<td></td>
</tr>
<tr>
<td>TL0 (RTL)</td>
<td>pin/cycle accurate</td>
<td></td>
</tr>
</tbody>
</table>

Faster Simulation Speed!!

More Accurate!!
Problems of Manual Refinement

| TL0 (RTL) | pin/cycle accurate |
| TL1       | cycle accurate     |
| TL2       | approximate        |
| TL3       | none               |

- **Protocol Timing**
  - TL0 (RTL): pin/cycle accurate
  - TL1: cycle accurate
  - TL2: approximate
  - TL3: none

- **Manually Refine**
  - (a) TL3 Model
  - (b) TL2 Bus Model
  - (c) TL1 Bus Model
  - (d) TL0 Model (RTL)

- Difficult Consistency Checking!!
- Tedious and Error-prone Re-modeling!!

- **Operations**
  - read()
  - write()
Proposed Automatic Approach

- An automatic approach to generate:
  - TL1 (Cycle Accurate) model
  - Cycle Count Accurate TL2 model (CCA-TL2)
Outline

- Introduction
- Related Work
- Problem Formulation
- Transaction Level Bus Model Generation
- Experimental Results
- Conclusion and Future Work
Related Work

- **Manual Modeling Techniques**
    - Cycle Count Accurate at Transaction Boundary (CCATB)

- **Library Based Approaches**
Outline

- Introduction
- Related Work
- Problem Formulation
- Transaction Level Bus Model Generation
- Experimental Results
- Conclusion and Future Work
A bus transaction is a read/write transfer between master and slave computation modules.
A SPA is a state machine designed for bus modeling.
- with data and control signals
- whose state progressing is synchronous with clock

A Burst Transaction Modeled by a SPA-pair

(a) master interface

(b) slave interface

Cycle: 0
A Burst Transaction Modeled by a SPA-pair

(a) master interface

(b) slave interface

Cycle: 3
Observations for Abstracting CA to CCA Model

1. Most transitions of the SPA-pair can be *pre-determined* before simulation (at static time).

2. A computation module concerns only with data content transferred.
   - Reduce the simulation overhead.

```plaintext
Master Computation Module
\[ \text{MADDR!} \rightarrow \text{MADDR?} \rightarrow \text{SRDATA!} \rightarrow \text{SRDATA?} \]\n
Slave Computation Module
\[ \text{MADDR!} \rightarrow \text{MADDR?} \rightarrow \text{SRDATA!} \rightarrow \text{SRDATA?} \]\n```
Problem Formulation

A SPA-pair (A transaction)

Compression Algorithm
Compressed Automata
SystemC Model Generation
CCA-TL2 Bus Model
SystemC Model Generation
TL1 Bus Model
SystemC Simulator
Outline

- Introduction
- Related Work
- Problem Formulation
- Transaction Level Bus Model Generation
  - Compression Algorithm
  - SystemC Bus Model Generation
- Experimental Result
- Conclusion and Future Work
Compression Algorithm

- Traces of the compression algorithm with an example with predetermined transitions:

(a) Master interface

(b) Slave interface
Compression Algorithm

• Traces of the compression algorithm with an example with predetermined transitions:

Collected data operation:
MADDR!
MADDR?

Weight: 1
Compression Algorithm

- Traces of the compression algorithm with an example with predetermined transitions:

```
Collected data operation:
MADDR!
MADDR?
SRDATA!
SRDATA?
MADDR!
MADDR?

Weight: 2
```
Compression Algorithm

- Traces of the compression algorithm with an example with predetermined transitions:

Collected data operation:
- MADDR!
- MADDR?
- SRDATA!
- SRDATA?
- MADDR!
- MADDR?
- SRDATA!
- SRDATA?

Weight: 3
Compression Algorithm

- Traces of the compression algorithm with an example with predetermined transitions:

```
MADDR!
MADDR?
SRDATA!
SRDATA?
MADDR!
MADDR?
SRDATA!
SRDATA?
```

Weight: 3
Non-predetermined transition

- Two kinds of non-predetermined transitions:
  - control dependent
  - data dependent
- Traverses and compresses each possible path.
Control-dependent Case

(a) master interface

(b) A control-dependent slave interface
1st Branch

(a) Master interface

(b) A control-dependent slave interface
1st Branch (cont’d)

Master interface

A control-dependent slave interface

~SREADY?

SREADY?

MADDR!

SREADY?

~SREADY?

MADDR!

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

SREADY?

~SREADY?

MADDR!

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!

~SREADY?

MADDR!
2\textsuperscript{nd} Branch

- The compression is finished.
  - Cycle Count Timing is preserved.
SystemC Bus Model Generation

- Is implemented in SystemC interface and channel pattern
  - the signals in SPA are translated into
    - variables
    - read/write events
  - data operations implemented as IMC (Interface Method Call).
- Is scheduled
  - in the clock-driven style for the TL1 bus models
  - in the event-driven style for the CCA-TL2 bus models
Experimental Results

- The core protocol, *burst write with handshake*, from OCP-IP is chosen.
  - Intel 3.40 GHz Xeon CPU
- For speed comparison:
  - Compare TL1 bus model cycle-by-cycle.
  - Compare CCA-TL2 bus model at transaction boundaries.
Conclusion & Future Work

- Formally discusses what information between different transaction levels can be simplified.
- Proposes the first automatic approach.
- Considers an automatic approach for multiple masters and multiple slaves with an arbiter in the future.
Thanks for your attention!!
Synchronous Protocol Automata (SPA)

- A synchronous protocol automaton is a tuple \((Q, D, C, A, V, \rightarrow, \text{clk}, q_0, q_f)\), where:
  - \(Q\): a finite set of control states
  - \(q_0, q_f\): initial state and final state
  - \(D, C\): a set of input or output of data and control signals
  - \(V\): a set of internal variables
  - \(A\): a set of actions
  - \(\rightarrow \subset Q \times Q \times \text{clk?} \times A\): transition relations