Protocol Transducer Synthesis using Divide and Conquer approach

†Shota Watanabe ‡Kenshu Seto †Yuji Ishikawa

‡Satoshi Komatsu ‡Masahiro Fujita

† Dept. of Electronics Engineering, Univ. of Tokyo
‡ VLSI Design and Education Center, Univ. of Tokyo

2007/01/24
ASPDAC2007
Outline

- Background
- Protocol Transducer Synthesis
- Baseline Method (Passerone’s Method)
- Proposed Method
- Experiment
Outline

Background

- Protocol Transducer Synthesis
- Baseline Method (Passerone’s Method)
- Proposed Method
- Experiment
Background

- Transistors continue to be shrunk
- More number of transistors on a single LSI
- These result in long design period
- However, it is important to shorten the time-to-market
Background

- IP reuse is an attractive solution
  - Reuse existing designs (IPs) for the new design.
  - It can also shorten verification period, because IPs are pre-verified.

- However, the interface mismatch prevents an IP from being reused...
In such case, designers usually insert a protocol transducer between incompatible interfaces.

However, designing a protocol transducer consumes much time...

Automatic synthesis of protocol transducer is expected to be useful.
Outline

- Background
- Protocol Transducer Synthesis
- Baseline Method (Passerone’s Method)
- Proposed Method
- Experiment
- Conclusion
Passerone et al proposed a protocol transducer synthesis method in [5].

Passerone’s Method – Search algorithm –

- Take a state from each automaton, and see if they are inconsistent or not.
- If not, search the next transitions.
- If more than two transitions are available under same condition, choose one having the least latency.

Output a value which is not arrived
The State-of-the-art Protocols

(a) Conventional Protocol

(b) Non-Blocking Transaction

(c) Out-of-Order completion of a transaction
The Limitations of the existing methods

- Most of the existing methods uses a **SINGLE automaton** as a specification of a protocol.

- These features with **parallelisms** are difficult to be described in an automaton.
Outline

- Background
- Protocol Transducer Synthesis
- Baseline Method (Passerone’s Method)
- Proposed Method
- Experiment
Objective

- Automatically synthesize protocol transducers even for complex protocol, by extending Passerone’s Method.

- Basic Idea: Divide-and-Conquer
  - Partition the exploration space into some small ones.
  - Construct the entire transducer from the partial transducers.
Outline of Proposed Method

1. Protocol Modeling Method
2. Sequence Level Synthesis
3. Automaton Level Synthesis
4. Construction of whole Transducer
Outline of Proposed Method

1. Protocol Modeling Method
2. Sequence Level Synthesis
3. Automaton Level Synthesis
4. Construction of whole Transducer
Outline of Proposed Method

1. Protocol Modeling Method
2. Sequence Level Synthesis
3. Automaton Level Synthesis
4. Construction of whole Transducer
Outline of Proposed Method

1. Protocol Modeling Method
2. Sequence Level Synthesis

Automaton Level Synthesis = Passerone’s Method + Extension
Outline of Proposed Method

1. Protocol Modeling Method
2. Sequence Level Synthesis
3. Automaton Level Synthesis
4. Construction of whole Transducer
1. Protocol Modeling Method

2. Sequence Level Synthesis
3. Automaton Level Synthesis
4. Construction of whole Transducer
We regard a protocol as a set of **Sequences**

A Sequence corresponds as an operation such as “Single Read”, “4-Burst Write”, etc.

A Sequence consists of a set of automata.
1. Protocol Modeling Method

- The **number of automata in a Sequence** depends on its protocol type.

- Protocol Types:
  - **Blocking Protocol**
  - **Non-Blocking Protocol**
  - **Out-of-Order Protocol**
1. Protocol Modeling Method

- The **number of automata in a Sequence** depends on its protocol type.

- Protocol Types:
  - **Blocking Protocol**
    - Sequence = an automaton
  - **Non-Blocking Protocol**
    - Sequence = two automata
      (Request / Response)
  - **Out-of-Order Protocol**
    - Sequence = two automata
      (Request / Response)
2. Sequence Level Synthesis

1. Protocol Modeling Method
2. Sequence Level Synthesis
3. Automaton Level Synthesis
4. Construction of whole Transducer
2. Sequence level Transducer Synthesis

- Synthesize a partial transducer from a pair of sequences.
- Each sequence has one or two automaton, according to its belonging protocol’s type.

<table>
<thead>
<tr>
<th>Belonging to a Blocking Protocol</th>
<th>Belonging to a Non-Blocking or Out-of-Order Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="Diagram of automaton for blocking protocol" /></td>
<td><img src="#" alt="Diagram of automaton for non-blocking protocol" /></td>
</tr>
</tbody>
</table>
2. Sequence level Transducer Synthesis

In case both sequences are Blocking.
2. Sequence level Transducer Synthesis

- In case both sequences are Non-Blocking or Out-of-Order.
2. Sequence level Transducer Synthesis

In case one is Blocking, the other is Non-Blocking or Out-of-Order.
3. Automaton Level Synthesis
   (Extended Passerone's method)

1. Protocol Modeling Method
2. Sequence Level Synthesis
3. Automaton Level Synthesis
4. Construction of whole Transducer
3. Automaton level Transducer Synthesis

- Automaton Level Synthesis is done by extended Passerone’s Method.
- Because Passerone’s method explores in depth-first-search, it cannot deal with loops in the automata.

- The extensions are following:
  - Handling of Loops
  - Multiple Data Sequences
Handling of Loops in automata

- Every automaton in the sequences has paths which returns to the initial state.
- So, all automata have loops in themselves.
- This prevents from being applied Passerone’s method.
Multiple Data Sequences

However, the insertion of “end state” cannot deal with “internal loop” shown in the figure. We call a sequence which includes this kind of automata “Multiple Data Sequence”.

We deal with Multiple Data Sequence by introducing super state Shell Graph = Kernel Graph
Multiple Data Sequences
4. Construction of whole Transducer

Input: Partial Transducers
Output: Entire transducer

1. Protocol Modeling Method
2. Sequence Level Synthesis
3. Automaton Level Synthesis
4. Construction of whole Transducer
4. Construction of whole Transducer

- We have to construct whole transducer from partial transducers.

- A Partial Transducer consists of
  - An FSM: in case input protocols are 
    (B,B),(NB,B),(B,NB), (OO,B),(B,OO)
  - A request FSM and a response FSM:
    In case (NB,NB),(OO,OO),(NB,OO),(OO,NB)
4. Construction of whole Transducer

- In case at least one is BK protocol

By regarding every initial state as the same one.
4. Construction of whole Transducer

- Otherwise (NB,NB),(OO,OO),(NB,OO),(OO,NB)
4. Construction of whole Transducer

- In case (NB,NB) or (OO,NB)

Diagram:
- FSM for Request
- FSM for Response
- FIFO
- FSM for Request
- FSM for Response
4. Construction of whole Transducer

- In case (OO,OO)

[Diagram showing FSM for Request and FSM for Response, with indication that nothing should be inserted]
4. Construction of whole Transducer

- In case (NB, OO)

FSM for Request

FSM for Response

FSM for Request

Re-Order Table

Send Response Transducer

Receive Response Transducer

Remove all access with slave

Remove all access with master

Set

Pop

WData

PUSH

Seq.ID

IsReady

Data
Outline

- Background
- Protocol Transducer Synthesis
- Baseline Method (Passerone’s Method)
- Proposed Method
- Experiment

Conclusion
Experiment 1: Non-Blocking Non-Blocking

**MASTER: OCP**
(Single Read, Non-Posted Write)

- **Request**
  - D: 0 idle
  - Single Read
  - Non-Posted Write

- **Response**
  - D: 0 idle
  - Single Read
  - Single Write

**SLAVE: OCP**
(Single Read, Single Write)

- **Request**
  - D: 0 idle
  - Single Read
  - Single Write

- **Response**
  - D: 0 idle
  - Single Read
  - Single Write
Test Bench

Master

Non-Posted Write Request

FSM for Request

FIFO
(2bit x 4)

Non-Posted Write Response

FSM for Response

Slave

Single Write Request

M_MCmd

M_MAddr

M_MData

M_SCmdAccept

S_MCmd

S_MAddr

S_MData

S_SCmdAccept

M_SResp

M_SData

S_SResp

S_SData

WData

PUSH

RData

PUSH

RST

CLK

D

Non-Posted Write Response

No Response

Single Write Request

Non-Posted Write Request

FIFO
(2bit x 4)
Simulation Waveform

Single Read Request

FIFO Push

Single Read Request

FIFO Push
Simulation Waveform

- Non-Posted Write Request
  - /TESTBENCH_REQ/M_Cmd: 000 001 011 000
  - /TESTBENCH_REQ/M_Addr: 0000 aaaa bbbb 0000
  - /TESTBENCH_REQ/M_Data: 0000 cccc 0000
  - /TESTBENCH_REQ/M_CmdAccept: 0000

- Single Write Request
  - /TESTBENCH_REQ/S_Cmd: 000 001 010 000
  - /TESTBENCH_REQ/S_Addr: 0000 aaaa bbbb
  - /TESTBENCH_REQ/S_Data: 0000 cccc
  - /TESTBENCH_REQ/S_CmdAccept: 0000

- FIFO Push
  - /TESTBENCH_REQ/FIFO_WD: 00 010 000
  - /TESTBENCH_REQ/FIFO_RD: 01 10 00
  - /TESTBENCH_REQ/FIFO_WEN: 0000
  - /TESTBENCH_REQ/FIFO_REN: 0000
  - /TESTBENCH_REQ/FIFO_EF: 0000
Simulation Waveform

Single Read Response

FIFO Pop
Simulation Waveform

/TESTBENCH_REQ/CLK

/TESTBENCH_REQ/RST

/TESTBENCH_REQ/M_Cmd 0000 0011 0000
/TESTBENCH_REQ/M.Addr 0000 aaaa bbbb 0000
/TESTBENCH_REQ/M.Data 0000 cccc 0000
/TESTBENCH_REQ/M_CmdAccept

/TESTBENCH_REQ/S_Cmd 0000 0010 0000
/TESTBENCH_REQ/S.Addr 0000 aaaa bbbb
/TESTBENCH_REQ/S.Data 0000 cccc
/TESTBENCH_REQ/S_CmdAccept

/TESTBENCH_REQ/M_SCmd 00 01 00
/TESTBENCH_REQ/M_SData 0000 ffff 0000
/TESTBENCH_REQ/M_SResp

/TESTBENCH_REQ/S_SCmd 00 01 00
/TESTBENCH_REQ/S_SData 0000 ffff 0000
/TESTBENCH_REQ/S_SResp

/TESTBENCH_REQ/FIFO_WD 00 01 00 10 00
/TESTBENCH_REQ/FIFO_RD 01 10
/TESTBENCH_REQ/FIFO_WEN
/TESTBENCH_REQ/FIFO_REN
/TESTBENCH_REQ/FIFO_EF
/TESTBENCH_REQ/FIFO_FF

Non-Posted Write Response

FIFO Pop
Conclusion

- We proposed a protocol transducer synthesis method using divide and conquer approach.
- Our method can be applied to the state-of-the-art protocols such as OCP, AMBA AXI, etc.
- We implemented our method on an original tool.
Demo will be available at EDS Fair

HERE!!
In University Plaza

10am～6pm
Jan. 25th, 26th
Protocol Transducer Synthesis using Divide and Conquer approach

Thank you for your attention.

Any Question?