Architectural/Micro-architectural Exploration on Virtual Platforms

Qi Zhu, Michael Kishinevsky, Zhu Zhou, Atul Kwatra
Intel Corporation
Outline

• Motivations
• Transaction-Level Modeling Methodology
• SOC Platform Exploration
• Summary and Future Directions
Motivations

pen&paper spreadsheets

manual and abstract (hard to explore complex designs)

Need a framework to
– explore multiple abstraction levels in between
– standard interfaces and modular designs for easier exploration and reusability

detailed perf. models (take time to build, hard to extend)

Virtual Platform Workshop at DAC 2009
Transaction-Level Modeling Methodology

• A system-level modeling framework (virtual platform) based on TLM
  – Separation of communication and computation for modular design and easier exploration.
  – Multiple abstraction levels of modeling for trade-off between complexity and accuracy.
  – Integration with external IPs.
  – Leverage of vendor tools for development, debugging and analysis capabilities.
  – Configurable domain-specific libraries with standard interfaces for design efficiency.

• Use SystemC 2.2 and TLM 2.0 library from OSCI
  – TLM 2.0 AT (approximately-timed) coding style is utilized.
    Validate the timing accuracy of TLM models with in-house detailed performance simulator.
RT Level

1. Interface

- No encapsulation of the communication signals.
- No abstraction of the communication protocol.
- Hard to explore different communication mechanisms.
- Hard to abstract the model to another level.

2. Internal components

always@(posedge clk)
if (Request_IRdy) {
......
}

always@(posedge clk)
if (Request_IBdy) {
......
}
Signal Abstraction – toward higher level

1. Interface
   (Address, Cmd, Length, Data, Status)

   - Internal computation components actively checking “firing” conditions every clock cycle.

   ```
   Clock() {
     if (!request_buffer.empty()) {
       ..... 
     }
   }
   ```

   - Data signals are encapsulated. Some separation of comm. and comp.

   - Still no abstraction of the communication protocol. Hard to explore different protocols.

   - Hard to explore different abstraction levels.

A port defines a set of services (through interface functions) provided by the component.
TLM Model

1. Interface

Sockets
- higher level abstraction of ports
- implements a generic set of services that works on specific comm. payload and protocol

Generic payload
- Command
- Address
- Data
- Byte enables
- Response status

Extensions

Protocol phases
- BEGIN_REQ
- END_REQ
- BEGIN_RESP
- END_RESP

M.H. — Sockets

nb_transport_bw(payload, phase...)
invalidate_direct_mem_ptr(...)

M.C.

b_transport (payload)
nb_transport_fw(payload, phase...)
get_direct_mem_ptr(...)
transport_dbg(...)
In TLM, computation components are passively waiting to be triggered by Transaction FSMs that handle transaction phase transitions (including communication protocol phases and internal phases).
TLM Benefits Summary

• Communication is explicitly modeled by transactions and separated from computation
  – Two aspects of transactions: data and protocol.
  – Data is encapsulated in generic payload and its extensions.
  – Protocol can be described by FSMs whose states are phases of transactions.

• Multiple abstraction levels of the design can be explored easier
  – Data: payload extensions.
  – Communication protocol: protocol phases.
  – Computation: transaction phases (including protocol phases and internal phases).
Timing Modeling in TLM

Timing accuracy is decided by the choice of transaction phases

- Untimed (no time annotation)
- Loosely-timed (single phase w/time annotation)
- Approximately-timed (multiple phases w/time annotation)
- Cycle-count accurate (w.r.t. the care set of behavior)

More accurate timing

Functional Model

Functional Model

Architecture Model

Architecture Model

Micro-Architecture Model

Micro-Architecture Model
Outline

• Motivations
• Transaction-Level Modeling Methodology
• SOC Platform Exploration
• Summary Future Directions
Arch/uArch Exploration on SOC Platform

• SOC characteristics
  – Communication is the focus – important to explicitly model communication.
  – Time-to-market is crucial (while resources are usually limited)
    – System-level modeling for early exploration.
    – Multiple abstraction levels for various design purposes.
    – Standard interfaces for IP integration.
  – Many derivatives from a base platform
    – Build a library of configurable component models.
    – Mixed-level modeling to support exploration on part of the platform.
Focus is on communication between various agents and memory.

A Virtual SOC Platform with approximate-timing is being built for arch/uarch exploration.
**Exploration Example: DDR Memory Controller**

- Configurable memory controller model (DDR, DDR2, DDR3, LP-DDR, etc.)
  - Number of incoming ports
  - Arbitration policy (RR, priority-based)
  - Scheduling
    - In-order vs. out-of-order scheduling of requests
    - Out-of-order request queue size
    - In-order request queue size
    - Out-of-order scheduling policy
    - SDRAM command level scheduling
  - Auto refresh (refresh burst)
- SDRAM configuration
  - # of ranks, banks, row bits, column bits
  - Timing specification
- Address mapping
- Data width
- Burst length
Exploration of Memory Controller

- Impact of priority-based scheduling between two request agents A1 and A2

<table>
<thead>
<tr>
<th></th>
<th>Avg. A1 latency</th>
<th>Avg. A2 latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 always over A2</td>
<td>13.1</td>
<td>17</td>
</tr>
<tr>
<td>50% A1 over A2</td>
<td>14.3</td>
<td>15.5</td>
</tr>
<tr>
<td>A2 always over A1</td>
<td>16.8</td>
<td>13.5</td>
</tr>
</tbody>
</table>

- Impact of hitting an open page (with all request from A1)

<table>
<thead>
<tr>
<th></th>
<th>Avg. A1 latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always hits open page</td>
<td>6.2</td>
</tr>
<tr>
<td>50% hits open page</td>
<td>8</td>
</tr>
<tr>
<td>10% hits open page</td>
<td>9.1</td>
</tr>
<tr>
<td>Never hits open page</td>
<td>9.3</td>
</tr>
</tbody>
</table>
Simulation Complexity

• Denotation:
  – \( P \): # of processes (components)
  – \( C \): simulation length in cycles
  – \( E \): average # of events per process during simulation
  – \( t_o \): overhead of switching processes
  – \( t_p \): average runtime for one invocation of a process

• Discrete-event simulation in our TLM model
  \[ E \times P \times (t_o + t_p) \]

• Clock-driven simulation in reference performance model
  \[ C \times P \times (t_o + t_p) \]

• Simulation speed is decided by \( E \), which relates to the modeling abstraction level.

• Our current model can simulate >100K transactions per second, with accuracy within 5% of RTL.
Model Development

- 4 man-month for memory controller and memory controller hub model, including
  - Study the architecture documents and reference models.
  - Develop the TLM models.
  - Validate the timing accuracy of TLM with reference models.
  - Study and use external and internal tools for development and analysis.

- Time could be much shorter if
  - There are library components available.
  - Modeling methodology is well defined.
  - Developers are more familiar with the architecture and reference models.
  - Abstraction level is higher.
Outline

• Motivations
• Transaction-Level Modeling Methodology
• SOC Platform Exploration
• Summary and Future Directions
What we learned

• TLM is the right way to build Virtual Platforms for architecture exploration
  – Separation of communication and computation.
  – Support multiple abstraction levels and mixed-level modeling. Approximately-timed TLM model can provide relatively accurate timing with fast simulation speed.
  – Interoperability through standard modeling methodology and communication interfaces.
  – Enable modular and configurable designs.
What we need for Exploration

• Models:
  - Library of configurable components with well-defined interface at various abstraction levels.

• Methodology
  - Methodology for exploring abstraction levels (through transaction phases). Trade-off analysis between complexity and accuracy.
  - Mixed-level modeling of components at multiple abstraction levels. Dynamic switching between abstraction levels.
  - Methodology for guided simulation to search design space. Combine simulation-based exploration with analytical methods.
  - Transaction-level power modeling for trade-off analysis between performance and power.

• Tools
  - How to leverage vendor tools - analysis, debug, etc.
Unified System-Level Framework based on TLM

- Library
  - Abstraction levels
    - (at each level there is a set of primitives)
    - Same component described at different abstraction levels
      (this is a conceptual description, multiple-level implementations can be put into the same module)

- Model
  - Function Model (untimed)
  - Arch Model (LT - AT)
  - uArch Model (AT - CA)

- Early design space exploration (arch, uarch, mapping)
- Dynamic validation (arch/uarch)
- SW Dev. and validation
- Reference model
- Synthesis
- Formal Verification