Flexible and Formal Modeling of Microprocessors with Application to Retargetable Simulation by Wei Qin and Sharad Malik

EE 249 Paper Presentation
Wenchao Li
Fall 2007
Outline

- Introduction
- Motivation
- Prior Work
- The Operation State Machine (OSM) Model
- Case Studies
- Conclusion
Outline

- Introduction
- Motivation
- Prior Work
- The Operation State Machine (OSM) Model
- Case Studies
- Conclusion
Microprocessor = Instruction-set-architecture (ISA) + Microarchitecture.

Instruction-set-simulator emulates the functionality of programs.

Microarchitecture simulator provides performance metrics.
Introduction contd.

- Important characteristics for a high quality microarchitecture simulation framework:
  - Efficient
  - Expressive
  - Declarative
  - Productive
Outline

- Introduction
- Motivation
- Prior Work
- The Operation State Machine (OSM) Model
- Case Studies
- Conclusion
Motivation

- Growth in application-specific processors demands a *retargetable* modeling framework that is capable of accurately capturing complex process behavior and generating efficient simulators.
- Simulation techniques in the microarchitecture modeling domain are not mature.

*Yea, dammit, platform-based design again!*
Outline

- Introduction
- Motivation
- Prior Work
  - The Operation State Machine (OSM) Model
- Case Studies
- Conclusion
Prior Work

- **Operation-centric:**
  - nML, ISDL, EXPRESSION.

- **Hardware-centric:**
  - MIMOLA, HASE, SystemC, Asim, Liberty;
  - UPFAST.

- **Other formalism:**
  - LISA
  - BUILDABONG
  - SimpleScalar
Outline

- Introduction
- Motivation
- Prior Work
- The Operation State Machine (OSM) Model
- Case Studies
- Conclusion
Why the heck do we need another model?

- Microprocessor specifications can be partitioned into:
  1. Operation layer
  2. Hardware layer
- Existing frameworks focuses on one or the other, or has limited flexibility.
- So?
OSM aims to provide clean and formal semantics to distinguish these two layers and at the same time model complex interactions between the two. This also helps to *orthogonalize* design considerations.
The OSM Model
Operation Layer

Multiple edges arbitrated by static priorities

Token buffer for allocated resources

Multiple state machines are coordinated by a director to avoid non-determinism.
The OSM Model
In the OSM model, we model the resources as tokens. A token manager manages one or more closely related tokens. It can grant a token to, or reclaim a token from an OSM upon request. Token managers may check the identity of the requesting OSMs when making decisions.
The OSM Model
Token Manager Interface (TMI).

Token transactions:
- *Allocate*: transaction of exclusive resources;
- *Inquire*: transaction of non-exclusive resources;
- *Release*: opposite of allocate;
- *Discard*: reset.
Scheduling OSMs (sequential)

```
Director::control_step()
{
    updateOSMList();
    OSM = OSMList.head; // head.next is the first
    while ((OSM != OSM.next) != OSM.tail) {
        EdgeList = OSM.currentOutEdges();
        foreach edge in EdgeList {
            result = OSM.requestTransactions(edge);
            if (result == satisfied) {
                OSM.commitTransactions(edge);
                OSM.updateState(edge);
                OSMList.remove(OSM);
                OSM = OSMList.head;
                break;
            }
        }
    }
}
```

Deadlocks from cyclic resource dependency is considered pathological.
Simulating OSMs

```cpp
nextEdge = 0;
eventQueue.insert(new clock_event(nextEdge));
while (!eventQueue.empty())
{
    event = eventQueue.pop();
    if (event->timeStamp >= nextEdge) {
        director.control_step();
        nextEdge += regularInterval;
        eventQueue.insert(new clock_event(nextEdge));
        break;
    }
    event->run();
    delete event;
}
```

OSM MoC is embedded inside the DE scheduler.
Modeling Hazards

- Structure hazard
- Data hazard
- Variable latency
- Control hazard

OSM of a 5-stage pipelined RISC processor
Outline

- Introduction
- Motivation
- Prior Work
- The Operation State Machine (OSM) Model
- Case Studies
Case Studies

- StrongARM: a five-stage pipelined implementation of the Advanced RISC Machine architecture.
- PowerPC 750: a dual-issue out-of-order superscalar processor.
The resulting simulator runs at 650k cycles/sec compared to 550k cycles/sec by SimpleScalar.

Model validated against an iPAQ-3650 PDA containing a SA-1100.

<table>
<thead>
<tr>
<th>benchmark</th>
<th>iPAQ(sec)</th>
<th>Simulator(sec)</th>
<th>difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>gsm/dec</td>
<td>0.59</td>
<td>0.572</td>
<td>3.05%</td>
</tr>
<tr>
<td>gsm/enc</td>
<td>1.69</td>
<td>1.647</td>
<td>2.54%</td>
</tr>
<tr>
<td>g721/dec</td>
<td>2.23</td>
<td>2.205</td>
<td>1.12%</td>
</tr>
<tr>
<td>g721/enc</td>
<td>2.31</td>
<td>2.293</td>
<td>0.74%</td>
</tr>
<tr>
<td>mpeg2/dec</td>
<td>14.85</td>
<td>14.55</td>
<td>2.02%</td>
</tr>
<tr>
<td>mpeg2/enc</td>
<td>32.85</td>
<td>32.38</td>
<td>1.43%</td>
</tr>
</tbody>
</table>
PowerPC 750

- Validated against a SystemC based model using a benchmark mix from MediaBench and SPECint 2000, and found that differences in timing are within 3% in all cases.
- OSM simulator runs at 250k cycles/sec, 4 times of the SystemC model.
About 60% of the source code for the OSM model is dedicated to instruction decoding and OSM initialization, which can be automatically synthesized through the use of an architecture description language. Most hardware modules and their TMI sets were reused across the two targets.

<table>
<thead>
<tr>
<th>Lines of code</th>
<th>SA-1100</th>
<th>PowerPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSM Total</td>
<td>3032</td>
<td>5004</td>
</tr>
<tr>
<td>SimpleScalar/SystemC</td>
<td>4633</td>
<td>16000</td>
</tr>
</tbody>
</table>
Introduction
Motivation
Prior Work
The Operation State Machine (OSM) Model
Case Studies
Conclusion
Conclusion

- **Where the paper succeeds:**
  - OSM as an efficient retargetable simulator generation framework for different microprocessor architecture including scalar, superscalar, very-long-instruction-word and multi-threaded (add tag).

- **Where the paper fails:**
  - Architecture description language can be derived by not done (the declarative criterion).
  - Mentioned OSM as ASM but how to do successive refinement is not clear.
Thank you.