Practical Issues in Implementing Fast and Accurate SystemC-Constructed Virtual Platform Simulation

Authors: Yu-Fu Yeh¹,² and Chung-Yang (Ric) Huang²
Affiliation: ¹ICL @ ITRI; ²GIEE @ NTU
Presenter: Yu-Fu Yeh
Outlines

- Introduction
- Preliminaries
  - Simulation overhead in virtual platform simulation
  - Synchronization reduction by an asynchronous discrete event simulation scheme
- QuteVP+ Implementations:
  - QuteVP+ engine
  - QuteVP+ utility library
- Experimental Results
- Conclusions
What is virtual platform simulation

- A software-constructed hardware simulation platform
  - Hardware components are constructed by software language (e.g. SystemC)
  - Software program can be executed on the processor model (e.g. Instruction set simulator, ISS)
- Usually contains everything for a system
  - Hardware: processor, bus, memory, DMA…
  - Software: OS, firmware, drivers, embedded programs
- Objectives:
  - System design optimization, architecture exploration, system-level verification

Implemented in C/C++/SystemC

<table>
<thead>
<tr>
<th>AP1</th>
<th>AP2</th>
<th>AP3</th>
<th>AP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware</td>
<td>Driver</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTOS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Processor</td>
<td>DSP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cache</td>
<td>ROM</td>
<td>RAM</td>
<td></td>
</tr>
</tbody>
</table>

HW/SW Co-design Co-Verification
The problem in virtual platform simulation

- A trade-off between simulation efficiency and simulation accuracy

**v.s.**

- Simulated models with higher abstract levels
  - better simulation efficiency
  - E.g. functional simulation

- Simulated models with lower abstract levels
  - More accurate outcome
  - E.g. cycle accurate simulation
Introduction

Modeling Language:
-- SystemC v2.2 + TLM 1.0

Abstract level:
Processor model:
-- ARM v5T Instruction Set Simulator

Other hardware modules:
-- Cycle Accurate

Software IPs
(C/ASM...)

ARM v5T Compiler

Plain-Binary file

32-bit Interconnection (QuteBus)

Processor

DMA Controller

Interrupt Controller

Timer Controller

ROM

RAM

DCT

Virtual Output

(ppm Files)
Our experience of SoC-based virtual platform simulation

m13 Version
- 2007.05 Finished
- Cycle Accurate
- Pin Accurate
- About 12.0 KIPS

m17 Version
- 2008.05 Finished
- Cycle Accurate
- OSCI TLM 1.0
- About 27.5 KIPS

UNACCEPTABLE SIMULATION SPEED
Introduction

- **QuteVP+**, a simulation framework, is proposed to follow our proposed simulation scheme to conduct fast and accurate SystemC-constructed virtual platform simulation.
Preliminaries

- Virtual platform simulation must consider dependent and concurrent relations among hardware components
  - Schedule Hardware Simulation Process (HSP) in a proper chronological order
To accurately mimic the concurrent hardware behavior, SystemC simulator schedules the HSPs created by SC_METHOD, SC_THREAD with synchronous discrete-event scheme (Sync-DES), or called clock-step simulation scheme (CSSM)
Preliminaries

- To schedule HSPs, SystemC kernel evokes synchronization (thread context switches), during simulation.
Preliminaries

Each module gets scheduled one or multiple times in one clock cycle

- Using serial simulator to mimic concurrent behavior
- Synchronizing HSPs with big simulation overhead

Context switches across one clock cycle over simulation time chart
What’s the problem?

Observation
- The biggest bottleneck of SystemC simulation is in the simulation kernel

Simulation Time (percentage)
- Hardware Module
- Functional Simulation
- Simulation Engine (synchronization/scheduling)
- Context Switches
- Scheduling
- Data copy

Virtual platform simulation time profiling
Asynchronous discrete even simulation for synchronization reduction

- In contrast to sync-DES, asynchronous discrete event simulation (async-DES) scheme benefits synchronization reduction.

Simulation time

Data dependency with HSP₁

Data dependency with HSP₃

Bus contention

Insert delay to ensure temporal accuracy of HSP₂

Sync

Delay

HSP₁

HSP₂

HSP₃

HSP₁ Ends

HSP₂ Ends

HSP₃ Ends
Asynchronous discrete event simulation (async-DES)

- In the virtual platform simulation with “async-DES”, there are two requirements
  - A synchronization checking mechanism to “avoid dependency violation”
  - A timing reconstruction technique to “maintain temporal accuracy”
Synchronization reduction with our proposed simulation scheme

- This work is based on our proposed simulation scheme \footnote{1} with USCM\footnote{2} and Trace-drive simulation to conduct fast and accurate MPSoC virtual platform simulation.

\begin{center}
\begin{tikzpicture}[node distance = 2cm, auto,>=latex, align=center, thick, every label/.style={font=\small}]
\node (init) {Initialization};
\node (trigger) [below of=init] {Trigger an HSP};
\node (synchronize) [below of=trigger] {Synchronize HSPs};
\node (execute) [below of=synchronize] {Execute trace-driven simulation};
\node (execute_hsp) [right of=execute, xshift=3cm] {Execute HSP};
\node (data) [above of=execute_hsp, yshift=-1cm] {Data dependency? (Checked by USCM)};
\node (data_y) [right of=data, xshift=1cm, yshift=0.5cm] {Yes};
\node (data_n) [right of=data, xshift=1cm, yshift=-0.5cm] {No};

\draw [->] (init) -- (trigger);
\draw [->] (trigger) -- (synchronize);
\draw [->] (synchronize) -- (execute);
\draw [->] (execute) -- (execute_hsp);
\draw [->] (execute_hsp) -- (data);
\draw [->] (data) -- (data_y);
\draw [->] (data) -- (data_n);
\end{tikzpicture}
\end{center}


Implementation of QuteVP+

• Goal:
  ◦ Realize the introduced async-DES scheme on SystemC-Constructed virtual platform simulation

• Difficulties
  ◦ The simulation scheme in SystemC follows Sync-DES (Clock-step simulation scheme)
  ◦ Modifying simulation scheme, the simulation must solve issues in “compatibility” and “adaptability”
Implementation of QuteVP+

• Compatibility
  ◦ Ensure the replacement of simulation scheme without affecting the primitive SystemC-defined functions
    • E.g. the functions, such as event notify(), wait() is relevant to the scheduling behavior

• Easy to use
  ◦ Consider the convenience to adapt the parallel out-of-order execution approach on SystemC-Constructed virtual platform
Implementation of QuteVP+

- QuteVP+ Overview

SystemC Virtual Prototypes

- Processor Model 1
- IP 1
- IP N

QuteVP+ Interface

QuteVP+ Engine

- Request Information Recorder
- Timing Restorer
- Memory Exclusivity Table
- Memory Exclusivity Checker
- HSP Trigger

SystemC Kernel
Use QuteVP+ interface to connect hardware model and QuteVP+ Engine

DMA
qvp+ Interface

QuteCore processor
qvp+ Interface

Static Memory
qvp+ Interface

Arbiter
qvp+ Interface

Pin Interface
qvp+ Interface

Memory Map

Master port

Slave port
Implementation of QuteVP+

- QuteVP+ creates an independent process to manipulate HSPs by out-of-order execution
  - Delta cycle scheduling
    - HSPs use timeless wait function for synchronization
      - QuteVP+ engine enables async-DES scheduling in each delta cycle
    - Record simulation traces of each HSP for trace-driven simulation
      - Maintain dependency relation
      - Reconstruct accurate simulation time
    - Process notification
Implementations of QuteVP+

- QuteVP+ utility library
  - While requesting a memory access, an HSP can use our utility library to check data dependency

```c
#include <QuteVP_Utility.h>
...
// QVP+ communication channel inheritance
void ARM_ISS::send_request() {
    // replace TLM communication function call, e.g. m_master_port->nb_put(mReqs);
    // by calling data-dependency checking function
    // and execute synchronization if necessary
    if (QuteVP_Engine->DataDependencyChecker(mReq, mResp)) {
        wait(sync_ok_event);
        QuteVP_Engine->RequestTransmitter(mReq, mResp);
    }
    if (mReq.get_command()==MEM_READ)
        M_resp_data = mResp.get_data();
}
```
Implementations of QuteVP+

- QuteVP+ utility library
  - `RequestTransmitter()` performs “direct data access” to reduce data copy

```cpp
#include <QuteVP_Utility.h>
// Using targetID mapping to the corresponding get_request function
qvp_response Qutevp_Engine::RequestTransmitter(qvp_request& mReq) {
    targetAddr = mReq.get_address();
    targetID = findIDfromMemMap(targetAddr);
    return pHSP[targetID].HSPptr()->get_request(mReq);
}

// Calculate the request address to seek the target ID
unsigned int pSysc::findIDfromMemMap(unsigned int& Addr) {
    ....
    return TargetID
}
```
Experimental results

- We compare the simulation efficiency of MPSoC virtual platform simulation where QuteVP+ performs with different simulation approaches
  - **CSSM**: Clock-Step Simulation Method, the synchronous discrete event simulation scheme that the primitive SystemC follows
  - **USCM**: USCM, the asynchronous discrete event simulation scheme in our previous work
Experimental results

- We construct an CELL-Like MPSoC virtual platform and JPEG encode and sparse matrix multiplication programs as test software cases running on the MPSoC virtual platform.

- Experimental environment
  - Workstation with Intel xeon CPU (qual-core*2) 2.2 GHz, 16GB RAM
  - CentOS kernel 2.6
  - Virtual platform constructed with SystemC v2.2
## Experimental results

- The comparison of synchronization count (Sync-Count) with CSSM and USCM

### Sparse matrix multiplication

<table>
<thead>
<tr>
<th>#CPU</th>
<th>#Inst</th>
<th>Simulation Cycle</th>
<th>Sync-Count by CSSM</th>
<th>Sync-Count by USCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>169,314,561</td>
<td>254,261,472</td>
<td>722,859,678</td>
<td>28,423,967</td>
</tr>
<tr>
<td>2</td>
<td>121,085,896</td>
<td>156,417,989</td>
<td>460,436,669</td>
<td>12,508,290</td>
</tr>
<tr>
<td>4</td>
<td>107,747,016</td>
<td>124,834,536</td>
<td>365,547,740</td>
<td>7,564,476</td>
</tr>
<tr>
<td>8</td>
<td>105,501,921</td>
<td>114,852,199</td>
<td>335,037,092</td>
<td>3,891,842</td>
</tr>
<tr>
<td>16</td>
<td>112,941,976</td>
<td>119,699,619</td>
<td>349,944,254</td>
<td>2,075,848</td>
</tr>
<tr>
<td>32</td>
<td>135,121,633</td>
<td>145,001,298</td>
<td>419,101,187</td>
<td>1,220,659</td>
</tr>
</tbody>
</table>

### JPEG-Encode

<table>
<thead>
<tr>
<th>#CPU</th>
<th>#Inst</th>
<th>Simulation Cycle</th>
<th>Sync-Count by CSSM</th>
<th>Sync-Count by USCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>512,513,774</td>
<td>771,969,678</td>
<td>2,212,558,094</td>
<td>87,409,800</td>
</tr>
<tr>
<td>2</td>
<td>386,970,594</td>
<td>507,338,577</td>
<td>1,520,454,360</td>
<td>45,790,479</td>
</tr>
<tr>
<td>4</td>
<td>314,622,268</td>
<td>376,936,991</td>
<td>1,093,803,850</td>
<td>21,489,726</td>
</tr>
<tr>
<td>8</td>
<td>287,006,102</td>
<td>327,838,743</td>
<td>926,966,612</td>
<td>10,945,309</td>
</tr>
<tr>
<td>16</td>
<td>272,113,534</td>
<td>308,490,886</td>
<td>860,594,492</td>
<td>5,436,347</td>
</tr>
<tr>
<td>32</td>
<td>264,576,313</td>
<td>295,742,881</td>
<td>831,027,316</td>
<td>2,918,037</td>
</tr>
</tbody>
</table>
Experimental results

Simulation speed = \( \frac{\sum_{i=0}^{n=\#CPU} \text{the number of simulated instructions}}{\text{Simulation runtime}_{\text{Simulation scheme}} (\text{sec})} \)

Speedup Ratio\(_{(\text{Scheme}_A \text{ with respect to Scheme}_B)} = \frac{\text{Simulation speed}_{\text{Scheme}_A}}{\text{Simulation speed}_{\text{Scheme}_B}} \)

<table>
<thead>
<tr>
<th>#CPU</th>
<th>SMM</th>
<th>JPEG-Enc</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CSSM(_{(\text{KIPS})})</td>
<td>USCM(_{(\text{KIPS})})</td>
</tr>
<tr>
<td>1</td>
<td>11.4</td>
<td>1376.5</td>
</tr>
<tr>
<td>2</td>
<td>16.3</td>
<td>1121.2</td>
</tr>
<tr>
<td>4</td>
<td>21.7</td>
<td>1007.0</td>
</tr>
<tr>
<td>8</td>
<td>32.4</td>
<td>933.6</td>
</tr>
<tr>
<td>16</td>
<td>24.3</td>
<td>824.4</td>
</tr>
<tr>
<td>32</td>
<td>21.0</td>
<td>682.4</td>
</tr>
</tbody>
</table>

KIPS means “Kilo Instruction Per Second

The rate of simulation runtime for data-dependency checking (DDC) with respect to total simulation runtime
Conclusions

- QuteVP+ can perform an async-DES to reduce unnecessary synchronization and reconstruct accurate simulation time
- QuteVP+ can
  - Perform the better simulation speed over 100+ times with respect to the conventional SystemC-based virtual platform
  - Offer accurate simulation outcome