Out-of-Order Parallel Simulation of SystemC Models on Many-Core Architectures

Collaborative Project with Intel Corp.

Rainer Dömer, Guantao Liu, Tim Schmidt
{doemer,guantaol,schmidtt}@uci.edu

Center for Embedded and Cyber-Physical Systems
University of California, Irvine

Project Key Points

• Project on Parallel SystemC Simulation
  – Faster simulation on multi- and many-core hosts
  – Maximum compliance with current execution semantics
  – Support for parallel execution of virtual platforms

• Introduction of a SystemC Compiler
  – Recoding Infrastructure for SystemC (RISC)
  – Advanced static analysis for parallel execution
  – Model instrumentation and code generation

• Parallel SystemC Core Library
  – Out-of-order parallel scheduler, multi-threading safe primitives
  – Many-core target platform (e.g. Xeon Phi™)

• Open Source
  – Collaboration with Accellera SystemC Language WG
Outline

- Out-of-Order Parallel SystemC Simulation
  - Traditional Discrete Event Simulation (DES)
  - Parallel Discrete Event Simulation (PDES)
  - Out-of-Order Parallel Discrete Event Simulation (OoO PDES)
- Promising Experimental Results
  - Embedded application example
  - Parallel benchmarks
- Project Overview and Status
  - SystemC compiler and out-of-order parallel simulator
  - Many-core target architecture
  - Virtual Platform (VP) integration
  - SystemC model analysis and recoding
- Concluding Remarks

Project Context: ESL Design

- Electronic System Level Models
  - Abstract description of a complete system
  - Hardware + Software
- Key Concepts in System Modeling
  - Explicit Structure
    - Block diagram structure
    - Connectivity through ports
  - Explicit Hierarchy
    - System composed of components
  - Explicit Concurrency
    - Potential for parallel execution
    - Potential for pipelined execution
  - Explicit Communication and Computation
    - Modules
    - Channels and Interfaces
Project Context: ESL Design

- Model Validation through Simulation!
  - Efficient system-level simulation is critical
    - Fast, and
    - Accurate!
  - Complexity of system models grows constantly
    - Need for speed!
- Parallel Simulation!
  - Parallelism explicitly specified in model
    - System-level Description Language (SLDL)
      - SystemC [Groetker et al., 2002]: \texttt{SC\_THREAD, SC\_METHOD}
      - SpecC [Gajski et al., 2000]: \texttt{par {}, pipe {}}
  - Parallel processing available in standard PCs
    - Multi-core host PCs readily available
    - Many-core technology is arriving

ESL Simulation: Related Work

Modeling Techniques
- Transaction-level modeling (TLM),
- TLM temporal decoupling,
- Savoiu et al. [MEMOCODE’05]
- Razaghi et al. [ASPDAC’12]

Hardware-based Acceleration
- Sirowy et al. [DAC’10]
- Nanjundappa et al. [ASPDAC’10]
- Sinha et al. [ASPDAC’12]

Discrete Event Simulation is slow

Distributed Simulation
- Chandy et al. [TSE’79]
- Huang et al. [SIES’08]
- Chen et al. [CECS’11]

SMP Parallel Simulation
- Fujimoto. [CACM’90]
- Chopard et al. [ICCS’06]
- Ezudheen et al. [PADS’09]
- Mello et al. [DATE’10]
- Schumacher et al. [CODES’11]
- Chen et al. [IEEEED&T’11]
- Yun et al. [TCAD’12]
Out-of-Order Parallel Simulation of SystemC Models on Many-Core Architectures

Out-of-Order PDES Technology

• Traditional Discrete Event Simulation (DES)
  – Reference simulators run *sequentially*, only one thread at a time (cooperative multi-threading model)
  – Cannot utilize the capabilities of multi- or many-core hosts
• Parallel Discrete Event Simulation (PDES)
  – Threads run in *parallel* (if at the same delta cycle and time)
  – Simulation-cycles are absolute barriers!
  ➢ Out-of-order Parallel DE simulation (OoO PDES)
    – Best technique known today, developed by CECS [DATE’12]
    – Threads run in *parallel and out-of-order*
even in different delta and time cycles if there are no conflicts!
    – Aggressive, runs maximum number of threads in parallel,
      but *fully preserves DES semantics and model accuracy*!

Discrete Event Simulation (DES)

• Traditional DES
  – Concurrent threads of execution
  – Managed by a central scheduler
  – Driven by events and time advances
    • Delta-cycle
    • Time-cycle
  ➢ Partial temporal order with barriers
• Reference Simulator
  – SystemC reference simulator uses cooperative multi-threading
  ➢ A single thread is active at any time!
  ➢ Cannot exploit parallelism
  ➢ Cannot utilize multiple cores
Parallel Discrete Event Simulation

- Parallel DES
  - Threads execute in parallel if
    • in the same delta cycle, and
    • in the same time cycle
  - Significant speed up!
  - *Synchronous PDES:* Cycle boundaries are *absolute barriers!*

- Aggressive Parallel DES
  - Conservative Approaches
    • Careful static analysis prevents conflicts
  - Optimistic Approaches
    • Conflicts are detected and addressed (*roll back*)

Out-of-Order Parallel DES

- Out-of-Order PDES
  - Threads execute in parallel if
    • in the same delta cycle, and
    • in the same time cycle,
    • OR if there are no conflicts!
  - Can utilize advanced compiler for static data conflict analysis
  - Allows as many threads in parallel as possible
  - Significantly higher speedup!
    • Results at [DATE’12], [IEEE TCAD14]
  - Fully preserves...
    • DES execution semantics
    • Accuracy in results and timing
Synchronous vs. Out-of-Order PDES

- Simple Example:
  - Parallel video and audio decoding with different frame rates

- Synchronous PDES
  - Observes time and delta cycles
  - Global time

- Out-of-Order PDES
  - Breaks cycle barrier
  - Local times (per thread)

### PDES:

### OoO PDES:
Promising Experimental Results

- What speedup can OoO PDES technology achieve on today’s many-core host platforms?
  - Early results using manually coded examples
    - Experimental setup
      - Many Integrated Core (MIC) Platform
        - 1 Intel® Xeon Phi™ Coprocessor 5110P at 1.053 GHz
        - 60 cores on ring-bus, 4 hyper-threads per core
      - 240 parallel hardware threads available
    - Highly parallel benchmarks
      - GPU pipeline example (Mandelbrot)
        - Embedded system example with test bench and DUT
      - Parallel floating-point multiplications (fmul)
        - Independent parallelism, balanced load, no communication
      - Parallel Fibonacci calculation (fibo)
        - Dependent parallelism, unbalanced load, some communication

OoO Parallel SystemC Project, U. of Tokyo, Jan. 19, 2015 (c) 2015 R. Doemer, CECS 13

GPU Pipeline Benchmark

- Graphics Application: Mandelbrot Set Rendering
  - Experimental Results
    - Sequence of 100 Mandelbrot images (640x448, depth 4096)
    - Manually created PDES model (Posix-threads based)
    - Multi-core platforms: Intel® Xeon® CPUs (4 cores, 2x6 cores)
    - Many-core platform: Intel® Xeon Phi™ (60 x 4 cores)

Mandelbrot Benchmark Results on Intel® Xeon® and Xeon Phi™

- 4 Core Host, PDES 3.7x speedup
- 2 CPU 6 Core Host, PDES 5.9x speedup
- 2 CPU 6 Core Host, OoO PDES 6.3x speedup
- 60x4 Core Xeon Phi, Posix PDES 46x speedup

OoO Parallel SystemC Project, U. of Tokyo, Jan. 19, 2015 (c) 2015 R. Doemer, CECS 14
Out-of-Order Parallel Simulation of SystemC Models on Many-Core Architectures

Parallel Benchmark Results

- Experimental Results (Intel® Xeon Phi Coprocessor, 60x4 cores)

Out-of-Order PDES Technology

- OoO PDES Key Ideas
  1. Dedicated SystemC compiler with advanced model analysis
     - Static conflict analysis based on Segment Graphs
  2. Parallel simulator with out-of-order scheduling on many cores
     - Fast decision making at run-time, optimized mapping

- Fundamental Data Structure: Segment Graph
  - Key to semantics-compliant out-of-order execution [DATE’12]
  - Key to prediction of future thread state [DATE’13]
    - “Optimized Out-of-Order Parallel DE Simulation Using Predictions”
  - Key to May-Happen-in-Parallel Analysis [DATE’14]
    - “May-Happen-in-Parallel Analysis based on Segment Graphs for Safe ESL Models” (Best Paper Award)
  - Journal publication: “OoO PDES for TLM” [IEEE TCAD’14]
    - Comprehensive article with HybridThreads extension
Project Overview and Tool Flow

- Research and Development Tasks
  1) Dedicated SystemC compiler (RISC infrastructure)
  2) Parallel SystemC library
  3) Performance tuning for many-core hosts
  4) Virtual Platform (VP) integration
  5) Model analysis (may-happen-in-parallel, MHP)
  6) Model recoding, transformation and optimization

R&D Task 1: SystemC Compiler

- RISC Software Stack
  - Recoding Infrastructure for SystemC
    - C/C++ foundation
    - ROSE compiler infrastructure
  - ROSE Internal Representation
  - Explicit support for
    - Source code analysis
    - Source-to-source transformations

(c) 2015 R. Doemer, CECS
R&D Task 1: SystemC Compiler

- RISC Software Stack
  - Recoding Infrastructure for SystemC
    - SystemC Internal Representation
  - Class hierarchy to represent SystemC objects

- RISC Software Stack
  - Recoding Infrastructure for SystemC
    - Segment Graph
      - Segment boundaries (wait)
      - Variable access conflict analysis

(c) 2015 R. Doemer, CECS
R&D Task 1: SystemC Compiler

- RISC Software Stack
  - Recoding Infrastructure for SystemC
    - Segment Graph
      - Current Status: Segment Graph for straight-line code

```c
MyModule(sc_module_name mn):
    sc_module(mn)
    {  SC_THREAD(straight1);
     void straight2()
     {  x = 42;
      y = 43;
      y;
      int o = y;
      wait(10, SC_NS);
      int kk;
      wait();
      int oo;
     }
```

R&D Task 2: Parallel SystemC Library

- Parallel Simulator with Out-of-Order Scheduler
  - OoO PDES execution

- Fast conflict table lookup
- Truly parallel threads
- Optimal thread-to-core mapping
R&D Task 2: Parallel SystemC Library

- Parallel Simulator with Out-of-Order Scheduler
  - SystemC kernel extension
- POSIX multi-threading
- MT-safe SystemC primitives
- Protected scheduling resources
- Protected communication
- Example: Life-time of a SC_THREAD

R&D Task 2: Parallel SystemC Library

- Parallel Simulator with Out-of-Order Scheduler
  - SystemC kernel extension
  - Current Status:
    - Multi-threading: Completed
      - SC_THREAD
      - SC_METHOD
    - Synchronization: Completed
      - sc_event, sc_event_or_list, sc_event_and_list
      - wait(sc_event), wait(sc_time)
      - next_trigger(sc_event), next_trigger(sc_time)
      - notify(), notify(SC_ZERO_TIME), notify(sc_time)
    - Communication: Ongoing
      - sc_prim_channel
      - sc_channel
    - Advanced features: Future
      - sc_trace, sc_signal, sc_bv, etc.
R&D Task 2: Parallel SystemC Library

- Parallel Simulator with Out-of-Order Scheduler
  - Test cases and benchmarks:
    | Benchmark example | SC_THREAD | SC_METHOD | wait | next_trigger | notify | sc_prim_channel |
    |-------------------|-----------|-----------|------|--------------|--------|----------------|
    | fmul.cpp          | ☑         | ☑         | ☑    |              |        |                |
    | wait_event.cpp    | ☑         | ☑         | ☑    |              |        |                |
    | wait_or_list.cpp  | ☑         | ☑         | ☑    |              |        |                |
    | wait_and_list.cpp | ☑         | ☑         | ☑    |              |        |                |
    | immediate_notify.cpp | ☑     | ☑         | ☑    |              |        |                |
    | next_trigger.cpp  | ☑         | ☑         | ☑    |              |        |                |
    | ProdCons_event.cpp | ☑    | ☑         | ☑    |              |        |                |
    | ProdCons_prim_channel.cpp | ☑ | ☑         | ☑    |              |        |                |
    | Mandelbrot.cpp    | ☑         | ☑         | ☑    |              |        |                |

- Reaches 7.97x on 8-core host!

R&D Task 3: Many-Core Target

- Intel® Many Integrated Core Architecture
  - Intel® Xeon Phi™ Coprocessor
    - Provides
      - 60 processor cores
      - 4 hyper-threads per core
      - 240 parallel hardware threads!
    - Hardware Features
      - Vector processing unit (VPU)
      - Extended Math Unit (EMU) for transcendental operations
      - Bidirectional ring interconnect
    - Peak performance
      - over 1 teraFLOPS (double-precision)
      - Uses familiar and standard programming models
        - Appears as a regular Linux machine with 240 cores!
R&D Task 3: Many-Core Target

- Performance Tuning for Intel® MIC Architecture
  - Optimize thread-to-core mapping
    - Core distance matters!
  - Architecture study
    - Xeon Phi™ ring network
    - Distributed Tag Directory (TD)
    - Unknown hash-function
  - Optimization Approach
    - Profile core distances
    - Determine TD
    - Place threads close to TD
    - Speedup 145% on average
    - To be presented at ASPDAC in Tokyo, January 21, 2015

R&D Task 4: Virtual Platforms (VP)

- Virtual Platform and Processor Models
  - Development of SW before physical HW is available
    - aka. “left shift” paradigm
  - Reach execution speed close to real-time
  - Provide accurate programmer’s view of the system
  - Offer advanced debugging capabilities
    - Checkpoints
    - Reverse execution
  - Examples:
    - Open Virtual Platforms (OVP) [Imperas’14]
    - Wind River® Simics® VP [Simics’13]
    - Develop a Virtual Prototype Environment (VPE) that allows seamless integration with SystemC models (Project year 2)
R&D Task 4: Virtual Platforms (VP)

- Virtual Prototype Environment (VPE) with SystemC
  - Simulator integration
    - SystemC simulation engine
    - VP simulator
  - Identify best integration scheme
    - Option A: Master and slaves
    - Option B: Multiple masters connected by bridges

- Example: Multiple VPs integrated with SystemC simulator
  - Virtual processors in Wrapper modules appear as SC_THREAD’s
  - Can execute in parallel, and be scheduled out-of-order

R&D focus (year 2):
- Maximize parallel execution speed
- Maximize timing accuracy (untimed, approx. timed, time-accurate)
R&D Task 5: SystemC Analysis

- Compile-time Model Analysis
  - Dedicated SystemC compiler
    - Abstract Syntax Tree with SystemC semantics
    - Recoding Infrastructure for SystemC (RISC)
  - Static analysis R&D opportunities (year 3)
    - Statistics of modules, channels, interfaces, …
    - Deadlock and live-lock analysis
    - Memory and stack size estimation
    - Hot-spot identification
    - Worst-Case-Execution-Time (WCET) analysis
    - Race-condition and parallel access conflict analysis
    - May-Happen-in-Parallel (MHP) analysis
      - Based on [DATE’14, best paper award]
    - ...

R&D Task 6: SystemC Recoding

- Model Transformation and Optimization
  - SystemC source-to-source transformation
    - ROSE-compiler infrastructure
    - Source re-coding and re-factoring
    - Model optimization
  - Recoding R&D opportunities (year 3)
    - Source code maintenance, formatting
    - Model customization and tuning
    - Back-annotation of profiling or other data
      - Performance
      - Power
    - Code instrumentation
    - Documentation
    - Model refinement, synthesis
    - ...

Project Timeline and Milestones

- **Year 1: Parallel SystemC Compiler and Simulator**
  - F’14: RISC for straight-line code, library with MT-safe primitives
  - W’15: RISC for functions, library with protected data structures
  - S’15: RISC code generator, simulator with OoO scheduling
  - Su’15: Event conflicts, simulator for parallel SystemC subset
  - **Goal:** Stable prototype for synthesizable SystemC, 10x speedup!

- **Year 2: Virtual Prototype Environment**
  - Simics VP integration, timing-accuracy optimization, compiler and simulator optimization, code hardening
  - **Goal:** Parallel SystemC tool suite release, 100x speedup!

- **Year 3: Open Source Release and Standardization Efforts**
  - SystemC analysis for hot-spots and MHP, source-to-source transformations, standardization efforts with Accellera
  - **Goal:** SystemC Virtual Prototype Environment, 100x speedup!

Concluding Remarks

- **Project on Advanced Parallel SystemC Simulation**
  - OoO PDES on multi- and many-core platforms
  - Maximum compliance with current execution semantics
  - Support for parallel execution of virtual platforms (VP)

- **Dedicated SystemC Compiler**
  - Recoding Infrastructure for SystemC (RISC)
  - Advanced static analysis for parallel execution
  - Model instrumentation, code optimization, transformation

- **Parallel SystemC Simulator**
  - Out-of-order parallel scheduler, multi-threading safe primitives
  - Many-core target platform (e.g. Xeon Phi™)

- **Open Source**
  - Collaboration with Accellera SystemC Language WG
References (1)


References (2)