An introduction to CoCentric

Las Palmas de G. C., Spain
Jun, 27th, 2002

Agenda

• System-level SoC design

• What is SystemC?

• CoCentric System Studio
  – SystemC based designs verification

• CoCentric SystemC Compiler
  – SystemC based design implementation
System-level SoC design

- System-level language, tools and methodology needed.

Early co-design of SoC architecture

- early • abstract • fast
- late • detailed • slow
SoC verification gaps

- Different languages are spoken
  - at different levels of abstraction
  - by HW/SW/system people

- Executable platform models become available too late in the design cycle

- Need to improve SW integration
  - more speed and easier SW debugging needed

What is SystemC?

- Modelling language for HW / SW systems

- Implemented as a C++ class library

- Delivered in two versions
  - SystemC 1.0 for hardware designs
  - SystemC 2.0 for complete SoC design
SystemC 2.0: the solution to SoC verification gaps

- Single language, multiple levels of abstraction
  - Remove the language barrier

- Ease of modeling @ transaction level
  - Get to executable platform model ASAP

- Simulation speed @ transaction level
  - Enable early HW/SW integration and verification

SystemC 2.0

- Design at multiple levels of abstraction

  - (untimed) functional level
    → executable specification

  - transaction level
    → platform design, SW and architecture verification

  - pin level
    → RTL/behavioral HW design and verification
SystemC 2.0

- No need to convert the whole design in one single step into a RTL model.

- No need to glue together different simulators for co-simulation.

System level design landscape

Synopsys CoCentric tools

C/SystemC design and verification solution that spans from concept to implementation in HW and SW
CoCentric tools

- CoCentric System Studio
  - SystemC simulator
  - Verification & Analysis of algorithms, architectures, HW and SW
  - Powerful debugging
  - Support of different levels of abstraction

- CoCentric SystemC Compiler
  - Closes the gap from system design to gates
  - Provides HW synthesis from SystemC
  - QoR identical to using RTL starting point
  - Allows early and more complete HW/SW verification

CoCentric SoC Design Flow
CoCentric System Studio Architecture

Smooth design & verification flow
Steps in SoC design

- **Design entry**
  - data flow graphs / control flow graphs
  - blocks
  - SystemC code

- **Simulation & Debugging**
  - Block level (micro-debugging)
  - System level (macro-debugging)

- **Implementation**
  - HW & SW blocks

CoCentric System Studio main workspace
Design entry: example FSM

Simulating with System Studio

- Automatic model detection (dataflow, FSM, SystemC)
- Optimized simulation
- Allows simulation of mixed architectural & algorithmic models
- Works with leading simulation tools like:
  - VCS, Sirocco
  - ModelSim
  - Verilog-XL
  - Matlab
- Interactive simulation control panel
- Also simulation control by SCFs
Simulation & Debugging

Debugging of HW in the system context
A Camellia Hand-Out
IST-2001-34410
Key Action IV.1.1

Debugging of SW in the system context

Let’s see an example

• Simple adder
  – Untimed functional modelling
  – Timed functional modelling
  – Transaction level modelling
  – Pin accurate modelling
Adder example: untimed functional model

Adder example: timed functional model
Adder example: Transaction level model

Adder example: pin level modelling
Implementing individual blocks

- Once the algorithm has been designed and simulated we face implementation:
  - SW implementation
    - simply continue down the design flow through traditional software compilers.
  - HW implementation
    - just replace the block by its description in:
      - Behavioral SystemC
      - RTL SystemC
      - Verilog gate-level

Implementation

- CSS enables the fastest route to implementation of HW blocks right from the system level of abstraction.

- For unit rate algorithms specified using DFGs, FSMs and leaf level C-based models CSS automatically generates synthesizable SystemC code.

- CSS also generates scripts to aid synthesis using SystemC Compiler and Design Compiler.

- This code output will be the input for CoCentric SystemC Compiler to continue down the implementation process.
CoCentric SystemC Compiler

- SystemC Compiler does complete HW synthesis from SystemC
- Produce same QoR as traditional HDL synthesis does
- Fits into any implementation flow: FPGAs, ASICs, SoC
- It’s tightly integrated with Synopsys synthesis technologies in Design Compiler and Physical Compiler

How to use SystemC Compiler

- Our example: The Reed-Solomon decoder

[Diagram of Reed-Solomon decoder]
A Camellia Hand-Out
IST-2001-34410
Key Action IV.1.1

Behavioral SystemC

- Mathematical algorithm → C++ → SystemC

RTL SystemC

- Structural description of the block
**SystemC Design Flow**

- Select library
- Select RTL coding style
- Set clock frequency
- Select wire load model
- Synthesize
- Write .db and .v output

**The synthesis script**
Synthesis results

SystemC Compiler: summary

- Fast bridge from specification to implementation by using SystemC
- Links with FPGA design flows enable fast prototyping
- Same QoR as traditional HDL synthesis
Conclusions

- SystemC solves SoC verification gaps

- CoCentric System Studio provides fast and efficient SystemC-based designs verification

- CoCentric SystemC Compiler provides HW synthesis directly from BL/RTL SystemC code

- Both System Studio and SystemC Compiler provide a complete SoC design flow based on SystemC

Any question?

Thanks!