SoC Design with UML and SystemC

Alberto Sardini
Software Engineering Specialist
Presentation Structure

- Modeling Languages for a SoC Design Process
- UML 2.0 Overview
- SystemC to UML 2 Mapping
- Summary
Modeling Languages for a SoC Design Process
UML in the SoC Design Methodology

New SoC Design Methodology

- Software Algorithms
- System Specifications
- Requirements, Standards
- UML/SystemC

- Hardware Spec.
- Software Spec.

Verification
- Untimed
- Timed
- RTL
- Hardware Refinement
- Gate Level, P&R Implementation

RTOS: Real Time Operating System
- Untimed: Behavioral Model without Clock Definition
- Timed: Behavioral Model with (partial) Clock Definition

Optimized Architecture by Quantitative Analysis
Division of HW/SW Work Performance Evaluation

Verification Environment for Levels of Abstraction Timing Accuracy
Modeling Languages

- **UML 2** = first major revision of UML
  - Supports SystemC-like structure modeling
  - Concepts currently available in Rose RT
  - Due: mid-2003

- **SystemC.UML Profile** = a profile dedicated to SystemC
  - Standard stereotypes for HW-oriented System C concepts
    (e.g., elementary channels, HW signal types)
UML 2.0 Overview
The OMG’s Model Driven Architecture

- The OMG has formulated an initiative called “Model-Driven Architecture” (MDA)
  - A framework for a set of standards in support of a _model-centered_ style of development

- Key characteristic of MDA:
  - _The focus and principal products of software development are models_ (instead of programs)
  - Models all the way – the design _is_ the implementation

- Rational is one of the principal drivers of MDA
  - Rose RT = executable models, code generation
MDA Implications and Advantages

- Ultimately, MDA should be possible to:
  - Execute (run) UML models
  - Implement Models automatically (…the model is the code!…)
  - Deploy Models automatically onto different platforms (…3GL, OS, uP,…)
    - Platform independent models (PIMs)
Structured Classes in UML 2.0

- **Structured classes:**
  - A Class that contain a collaboration structure of interconnected ‘parts’ and ‘ports’
  - ‘Parts’ can be instances of structured or unstructured classes
  - Based on modeling concepts found in popular architectural description languages (UML-RT)
Structured Classes: External Structure

- Objects that may have multiple interaction points: *ports*
  - For accessing the functional capabilities of the object
  - Different ports may offer different capabilities
Port Semantics

- A port can support **multiple interface specifications**
  - Provided interfaces (what the object can do)
  - Required interfaces (what the object needs to do its job)

```plaintext
«interface»
ControllerIF

stateChange ( s : state ) : void

«interface»
ControlleeIF

start () : void
stop () : void
queryState () : state
```

- **Incoming signals/calls**
  - c:ClassX

- **Outgoing signals/calls**
  - p1

«provides»
«uses»
Dynamics of Interface Usage (Protocols)

- **Interface specifications** define what objects can do
  - For greater architectural control, it is also necessary to define (constrain) the order in which things are done
  - e.g., operator-assisted call

![Diagram of call flow between Caller, Operator, and Callee]

caller call
ack
number
transfer
caller call
ack
talk

time
**Connecting Ports**

- Ports can be joined by **connectors** to create peer collaborations composed of structured classes.

![Diagram](image)

Connectors model *communication channels*.
 Systems as Hierarchies of Structured Classes

- Structured Classes may have an internal structure of parts (structured classes) and connectors
- Structured Classes can be used throughout development cycle
  - System architecture to lowest level design elements
Structured Classes: Internal Behavior

- Events may occur on any one of the ports
  - Hierarchical state machine

Action Code

```cpp
transitionS1toS2:
{int x;
 x = 0;
 p2.send(s1);
 p3.send(s2);
 ... }
```
SystemC to UML 2 Mapping
**SystemC to UML 2 Mapping**

- Module = *Structured Class (Capsule in RoseRT)*
- Interface = *Interface*
- Port = *Port*
- Primitive Channel = *(stereotyped) Connector*
- Hierarchical Channel = *Structured Class*
- Event = *(Stereotyped) Event*
- Sensitivity = *Trigger + State Combination*
- Process = *Behavior (of Structured Class); Statechart*
- `wait()` = *Statechart receive*
Example: Simple Bus Spec (2 of 2)

// MEM: Memory
// DSP: DSPc
// microC: MicroCont
// BUS: Bus
// ASIC: ASICc
// ARB: Arbiter

// clockR1: Clock
// ASIC: ASICc
// MEM: Memory
A SoC design process based on a combination of UML and SystemC seems advantageous

- Relies on standards
- Proven technologies
- Supported by tools

Recent developments in UML promise to make this an even more appealing approach

- UML 2 structure modeling provides a direct graphical representation of SystemC concepts