Eugenio Villar – University of Cantabria

FORMALIZATION OF THE MARTE/SYSTEMC INTEROPERABILITY FOR HW/SW CO-DESIGN
Context

- SystemC
  - Wide use for system modeling
  - Increasing interest for system specification
  - Untimed modeling > 25%

- Increasing interest in MARTE

- Increasing interest in MARTE/SystemC interoperability
**Context**

- **HW/SW Co-Design flow**
  - Requirements
  - Functional design
  - Executable Specification
  - Co-Design
  - Co-Simulation
  - HW Platform
  - Compilation

- **Temporal Transformations**
  - Untimed Models
  - Untimed (executable) Models
  - Timing Constraints
  - Timing Estimations

- **Architectures**
  - Discrete-Time Models
  - Discrete-Event Models

- **Languages**
  - VHDL
  - Verilog
  - SystemC

- **SystemC**
  - IP Reuse
  - Behavioral Synthesis
  - RTL Synthesis

- **Hardware/SW Implementation**
Agenda

- Motivation
- Formalization of the Saturn design methodology
- MARTE/SystemC interoperability formalization
- Conclusions
Motivation

- Mature HW design process
  - RTL description
  - RTL simulation (Discrete-Time MoC)
  - RTL synthesis
  - Logic simulation (Discrete-Event MoC)
  - Placement & Routing
  - Static Timing Analysis
  - Back-annotated logic simulation (Discrete-Event MoC)
Motivation

- The MPSoC
  - Multi-processing platform
  - With ‘some’ additional HW
  - Most of the functionality implemented as Embedded SW
Motivation

- Software Reliability

WASHINGTON (COMPUTERWORLD) - Software bugs are costing the U.S. economy an estimated $59.5 billion each year, with more than half of the cost borne by end users and the remainder by developers and vendors, according to a new federal study.

Improvements in testing could reduce this cost by about a third, or $22.5 billion, but it won't eliminate all software errors, the study said. Of the total $59.5 billion cost, users incurred 64% of the cost and developers 36%.

- http://www.cse.lehigh.edu/~gtan/bug/softwarebug.html
- http://www5.in.tum.de/~huckle/bugse.html ...
Motivation

- Clear need for sound HW/SW design methods
  - Formal methods in HW/SW System Engineering

- Formalization of the HW/SW Co-Design process
  - Formalization of the MoCs at each level
    - Horizontal heterogeneity
  - Formalization of the transformations among MoCs
    - Vertical heterogeneity
Objective

- Formalization of the MARTE/SystemC interoperability

- ForSyDe as formal framework
  - Untimed, Synchronous, Timed and Continuous MoCs
  - Unified modeling
    - Functionality
    - Timing
The Saturn design methodology

System Design

Artisan Studio

SysML

Synthesizable HW

Embedded SW

Saturn Verification Framework

Synthesizable HW

Embedded SW

SystemC

QEMU

Matlab/Simulink Models
The ForSyDe Formal Support

System Design

ForSyDe Formal Support

Artisan Studio

System Design

Modeling formalization

Transformation consistency

Modeling formalization

Saturn Verification Framework

1. SysML
2. Embedded SW
3. Synthesizable HW
4. Transformation consistency
5. Modeling formalization
6. Synthesizable HW
7. Embedded SW
8. Matlab/Simulink Models

SystemC

QEMU

EXITEME
The ForSyDe Formal Support

- SystemC/RTL synthesizable code
The ForSyDe Formal Support

- SysML SW model

B alternatesWith C
The ForSyDe Formal Support

- SysML HW/SW interaction

B \equiv F
D alternatesWith C
The ForSyDe Formal Support

- SW generation

B alternatesWith C, doesn’t?
The ForSyDe Formal Support

- HW/SW co-simulation

B ✟ F, doesn't?  B alternatesWith C, doesn't?

D alternatesWith C, doesn't?
Extension of the Saturn SysML profile

- System-level specification capabilities
  - System-level modeling
  - Reference to any HW/SW implementation

Artisan Studio

SysML

Synthesizable HW

Embedded SW

System Design
Extension of the Saturn SysML profile

- System-level specification capabilities
  - System-level modeling
  - HetSC profile
Extension of the Saturn SysML profile

- SystemC HetSC generation
MARTE/SystemC interoperability formalization

- MARTE computation and communication stereotypes
  - Generic Resource Modeling
MARTE/SystemC interoperability formalization

- MARTE computation and communication stereotypes
  - Generic Resource Modeling

- Different MoCs can be identified
MARTE/SystemC interoperability formalization

- MARTE computation and communication stereotypes
  - Generic Resource Modeling
MARTE/SystemC interoperability formalization

- MARTE computation
  - Activity Diagrams

\[
\text{mealyU}(\gamma, g, f, w_0) = p
\]
where
\[
p(s) = s'
\]
\[
f(w_i, \acute{a}_i) = \acute{a}'_i
\]
\[
g(w_i, a_i) = w'_{i+1}
\]
\[
\pi(\nu, s) = (\acute{a}_i), \nu(i) = \gamma(w_i)
\]
\[
\pi(\nu', s') = (\acute{a}'_i), \nu'(i) = \#f(w_i, \acute{a}_i)
\]
MARTe/SystemC interoperability formalization

- MARTe computation
  - Activation Cycles
MARTE/SystemC interoperability formalization

- MARTE timing
  - UML signals and MARTE clocks as ForSyDe signals
  - Clock relations
    - Clock Constraints Specification Language

UML Signal Values
{...a, b, c, d, e,...}

^C_n

S_{ForSyDe} = \langle e_1, e_{i+1}, e_{i+2}, e_{i+3}, e_{i+4}, ... \rangle
MARTE/SystemC interoperability formalization

- Verification of design constraints
  - MARTE clock relations
    - \( B \equiv F \)
    - \( D \text{ alternatesWith} C \)

<table>
<thead>
<tr>
<th>B \equiv F</th>
<th>D \text{ alternatesWith} C</th>
</tr>
</thead>
<tbody>
<tr>
<td>( B \text{ write}<em>\text{CALL()} ) and ( F \text{ write}</em>\text{CALL()} );</td>
<td>( D \text{ write}<em>\text{CALL()} ) -&gt; ( \text{next}(C \text{ write}</em>\text{CALL()} ) before ( D \text{ write}_\text{CALL()} ));</td>
</tr>
<tr>
<td>( \text{psl assert} B _\text{synchronous}_F );</td>
<td>( \text{psl assert} D _\text{alternates}_C );</td>
</tr>
</tbody>
</table>

- SystemC assertions
Conclusions

- ForSyDe as formalization metamodel for Saturn
  - Providing a formal framework
  - Defining design rules and constraints
- HetSC extension to the Saturn profile
- MARTE/SystemC interoperability formalization
  - First step towards a SystemC generation methodology
    - Based on a formal framework
    - Predictable executable specification results
  - Application to system design verification