SystemC as an Heterogeneous System Specification Language

Eugenio Villar
Fernando Herrera
University of Cantabria

November 16-17, 2006 Zurich, Switzerland
Challenges

- Massive concurrency
  - Complexity
    - PCB
    - MPSoC with NoC
  - Nanoelectronics
Challenges

- Heterogeneity
  - Algorithmic heterogeneity
    - DSP, control, protocol stacks, multimedia, ...
  - Constraints heterogeneity
    - RT, reliability, resources, performance, ...
  - Resource heterogeneity
    - ASIC, FPGA, mP, nC, ...
    - GPP, ASIP, AS-HW, eP-HW, AMS, ...
  - Methodological heterogeneity
    - Languages and tools
Main motivation

Need for a specification methodology

- Supporting different MoCCs
- Heterogeneous
- Executable
- Link to implementation
  - HW and SW
- Based on a standard language
SystemC as a system specification candidate

- SystemC specification methodology
  - Supporting different MoCCs
  - Heterogeneous
  - Link to implementation
    - HW and SW
  - Executable
  - Based on a standard language
SystemC as a system specification candidate

SystemC is committed to support system design

- Valuable input to OSCI and the IEEE
- Theoretical foundations to the standardization process
SystemC as a ‘straw man’

System specification languages
- Supporting different MoCCs
- Heterogeneous
- Link to implementation
  - HW and SW
- Executable
- Widely-used standard language

UNIFIED MODELING LANGUAGE™
Agenda

- SystemC support for different MoCCs
- SystemC heterogeneous specification
- Link to implementation
- Future work
## SystemC architecture

<table>
<thead>
<tr>
<th><strong>Methodology-Specific Libraries</strong></th>
<th><strong>Layered Libraries</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>HetSC: Heterogeneous System</td>
<td>Verification library</td>
</tr>
<tr>
<td>Specification Methodology Library</td>
<td>TLM library, etc</td>
</tr>
</tbody>
</table>

### Primitive Channels
- Signal, Fifo, Mutex, Semaphore, etc

### Structural Elements
- Modules
- Ports
- Interfaces
- Channels

### Data Types
- 4-valued logic
- Bits and Bit Vectors
- Arbitrary Precision Integers
- Fixed-point types

### Discrete-Event (DE) simulation kernel
- Events
- Processes

### C++ Language Standard
MoCC abstraction

Discrete-Event (DE) simulation kernel

Discrete-Time (DT)

Clocked Synchronous (CS)

Synchronous Reactive (SR)

SDF

Analog solver

Analog solver

Analog solver

SystemC-AMS

PN

KPN

CSP

untimed MoCs

synchronous MoCs

other timed MoCs

untimed MoCs
Design refinement support

Untimed MoCs

Synchronous MoCs

Other timed MoCs

PN
KPN
CSP
SDF

Synchronous Reactive (SR)
Clocked Synchronous (CS)

Discrete-Time (DT)
RT-ISS

Discrete-Event (DE)
Logic model
Fundamental question

Which are the MoCCs that can be abstracted from the DE MoCC?

- Strict answer: Only strict-timed MoCCs

\[
F : \text{MoCC} \rightarrow \text{DE} \\
(\forall e = (v, t) \in (V, T) \rightarrow F(e) = [F_V(v, t), F_T(v, t)])
\]

\[
F_T : (V, T) \rightarrow \mathbb{N} \times \mathbb{N} \\
(\forall e = (v, t) \in (V, T) \rightarrow F_T(v, t) = [F_t(v, t) \; \text{tu}, F_{\tilde{a}}(v, t)])
\]

- Limited design scope: RT-Logic synthesis
Fundamental question

Which are the MoCCs that can be abstracted from the DE MoCC?

- **Strict answer:** Only strict-timed MoCCs

  \[ F : \text{MoCC} \rightarrow \text{DE} \]
  \[ e = (v, t) \in (V, T) \rightarrow F(e) = [F_V(v, t), F_T(v, t)] \]

  \[ F_T : (V, T) \rightarrow NxN \]
  \[ e = (v, t) \in (V, T) \rightarrow F_T(v, t) = [F_t(v, t) tu, F_{\alpha}(v, t)] \]

- **SystemC** is committed to support system design
Fundamental question

Which are the MoCCs that can be abstracted from the DE MoCC?

- Relaxed answer: Any computable MoCC
Fundamental problems

How to represent untimed events onto the DE MoCC?

- Breaking the order relationship of d cycles

\[ e_1 = (v_1, t_1) \rightarrow F_T(e_1) = (t_{e_1}, \text{ns}, \delta_{e_1}) \]
\[ e_2 = (v_2, t_2) \rightarrow F_T(e_2) = (t_{e_2}, \text{ns}, \delta_{e_2}) \]

\[ t_{e_1} < t_{e_2} \Rightarrow e_1 < e_2 \]
\[ t_{e_1} = t_{e_2} \text{ and } \delta_{e_1} < \delta_{e_2} \not\Rightarrow e_1 < e_2 \]
Fundamental problems

Why any MoCC?
- Because DE can model any algorithm running on any computer
- Efficiency
  - Ability to hide unnecessary details
SystemC specification structure

- SystemC processes connected by channels
  - Timing evolving during the design process
  - Strict-Timed Test Bench
SystemC specification syntax

Semantical channels

- Associated to concrete implementations
  - uc_fifo
  - uc_rv
  - uc_sr
  - ...

SystemC specification syntax

- Concurrent processes
  - As few restrictions as possible
    - Communication and synchronization through channels
SystemC specification syntax

Concurrent processes
- As few restrictions as possible

```c
SC_MODULE(csp_spec) {
    public:
    uc_rv *rvch;
    ... void P1();
    void P2();
    ...

    SCCTOR(csp_spec) {
        SC_THREAD(P1);
        SC_THREAD(P2);
        ...
    }
};

void csp_spec::P1() {
    int a, b;
    ...
    while(true) {
        ... rvch.call(a,b);
        ...
    }
};

void csp_spec::P2() {
    int a, b;
    ...
    while(true) {
        ... rvch.accept(a,b);
        ...
    }
};
```
SystemC heterogeneous specification

- Horizontal heterogeneity
  - Ability to combine several MoCCs in the same specification

Specification

- KPN
- CSP
- SDF
- SR

Refined model(s)

- SDF
- DT
- SDF
- SR

HW/SW implementation

- SW
- HW
- SW
- SW
SystemC heterogeneous specification

Vertical heterogeneity

- Ability to transform one MoCC to another while preserving the equivalence
SystemC heterogeneous specification structure

- SystemC processes connected by channels
  - Border channels
  - Border processes

Test Bench

System Specification

KPN

CSP

SystemC DE simulation kernel
SystemC heterogeneous specification structure

SystemC processes connected by channels

- Border channels

```cpp
SC_MODULE(csp_pn_spec) {
    public:
        uc_rv_fifo *rvfifoch;
        ...
    void P1();
    void P2();
    ...
    SCCTOR(csp_pn_spec) {
        SC_THREAD(P1);
        SC_THREAD(P2);
        ...
    }
};

void csp_pn_spec:P1() {
    int a, b;
    ...
    while(true) {
        ...
        rvfifoch.call(a,b);
        ...
    }
};

void csp_pn_spec:P2() {
    int a, b;
    ...
    while(true) {
        ...
        rvfifoch.read(a);
        rvfifoch.write(b);
        ...
    }
};
```
SystemC heterogeneous specification structure

- SystemC processes connected by channels
  - Border channels
  - Border processes

```
SC_MODULE(csp_pn_spec) {
    public:
    uc_rv *rvch;
    uc_fifo *fifoch;
    ...
    void P1();
    void P2();
    void P3();
    ...
    SC_CTOR(csp_pn_spec) {
        SC_THREAD(P1);
        SC_THREAD(P2);
        SC_THREAD(P3);
        ...
    }

    void csp_pn_spec::P2() {
        int a, b;
        ...
        while(true) {
            ...
            fifoch.read(a);
            ...
            rvch.call(a, b);
            ...
            fifoch.write(b);
            ...
        }
    }
};
```
HetSC

- SystemC heterogeneous specification methodology and associated library
- Available at [www.teisa.unican.es/HetSC](http://www.teisa.unican.es/HetSC)
Link to implementation

- **SW synthesis**
  - Substitution of the simulation kernel by the equivalent RTOS functions

- **HW synthesis**
  - New generation of behavioral synthesis from C
Future work

FPVI IST Project ANDRES

- Formal foundation to HetSC
- Based on ForSyDe
- Close collaboration UC-KTH