Methodology for Efficient Design of Continuous/Discrete Co-Simulation Tool

G. Nicolescu, H. Boucheneb, L. Gheorghe, F. Bouchimma

Ecole Polytechnique de Montréal
Tel: (514) 340 4711 ext 5434
Fax: (514) 340 3240
Email: gabriela.nicolescu@polymtl.ca
Continuous/Discrete Systems

- Two types of components integration
  - Continuous timed
  - Discrete event

- Illustrative systems
  - Electro-mechanical
  - Mixed-signal
  - Radio
  - Hybrid
  - …
Continuous/Discrete Systems Design

- Collaboration between different teams
- Incremental refinements through different abstraction levels with specific execution models
- Validation requires joint execution of heterogeneous execution models
  - Co-Simulation Technique
Challenges for Continuous/Discrete Co-Simulation

- Defining new tools facilitating cooperation between different teams
  - Exploiting powerful existing tools (Simulink, SystemC, …)
  - Taking into account implementation choices
  - Enabling easy specification, automatic generation for co-simulation interfaces
Outline

- Introduction
- Methodology for co-simulation tool design
- Application for CODIS tool design
- Conclusions
### Continuous/Discrete Systems

**- Heterogeneous Execution Models -**

<table>
<thead>
<tr>
<th>Concept Model</th>
<th>Time</th>
<th>Communication means</th>
<th>Processes activation rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete</td>
<td>It advances discretely</td>
<td>Set of events</td>
<td>Processes are sensitive to events</td>
</tr>
<tr>
<td>Continuous</td>
<td>It advances by integration steps (IS)</td>
<td>Piecewise-Continuous signals</td>
<td>Processes are executed at each IS</td>
</tr>
</tbody>
</table>

- Events exchanged between the two models the discrete models
  - Sampling events
  - Update signal events
  - State events
Continuous/Discrete Systems
- Heterogeneous Execution Models -

Simulation Step  |  Synchronization

State Event  |  Signal Update/Sampling Event
Methodology for Co-Simulation Tool Design

Generic Stage

- Definition of the synchronization operational semantic
- Distribution of the synchronization functionality to co-sim. interfaces
- Interfaces behavior formalization and verification
- Definition of the internal architecture and co-simulation library

Implementation stage

- Sim. tools analysis
- Library elements implementation
- Implementation validation

Discrete

Continuous
Méthodologie de conception des outils de co-simulation

Generic Stage

- Definition of the synchronization operational semantic
- Distribution of the synchronization functionality to co-sim. interfaces
- Interfaces behavior formalization and verification
- Definition of the internal architecture and co-simulation library

Implementation stage

- Sim. tools analysis
- Library elements implementation
- Implementation validation

Discrete
- Co-Simulation Interface
- Co-Simulation Bus

Continu
Méthodologie de conception des outils de co-simulation

Generic Stage
- Definition of the synchronization operational semantic
- Distribution of the synchronization functionality to co-sim. interfaces
- Interfaces behavior formalization and verification
- Definition of the internal architecture and co-simulation library

Implementation stage
- Sim. tools analysis
- Library elements implementation
- Implementation validation

Discrete
- Couche 1
- Couche 2
- Couche 3
- ... 
- Couche N

Continu
- Couche 1
- Couche 2
- Couche 3
- ... 
- Couche N

Co-Simulation Bus
Méthodologie de conception des outils de co-simulation

**Generic Stage**
- Definition of the synchronization operational semantic
- Distribution of the synchronization functionality to co-sim. interfaces
- Interfaces behavior formalization and verification
- Definition of the internal architecture and co-simulation library

**Implementation stage**
- Sim. tools analysis
- Library elements implementation
- Implementation validation

**Discrete**
- Co-Simulation Interface

**Continu**
- Co-Simulation Interface

**Biblio. Co-Sim.**

**Co-Simulation Bus**
Méthodologie de conception des outils de co-simulation

Generic Stage
- Definition of the synchronization operational semantic
- Distribution of the synchronization functionality to co-sim. interfaces
- Interfaces behavior formalization and verification
- Definition of the internal architecture and co-simulation library

Implementation stage
- Sim. tools analysis
- Library elements implementation
- Implementation validation

Discrete
- Co-Simulation Interface
- Co-Simulation Bus

Continu
- Co-Simulation Interface
Outline

- Introduction
- Methodology for co-simulation tool design
- Application for CODIS tool design
- Conclusions
Synchronization
– Operational Semantic –

- Based on DEVS

\[ \text{DEV} = \langle X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta \rangle \]

\( X = \{ (pd, vd) \mid pd \in \text{InPorts}, vd \in Xpd \} \) - set of input ports and their values in the discrete domain

\( Y = \{ (pd, vd) \mid pd \in \text{OutPorts}, vd \in Ypd \} \) - set of output ports and their values in the discrete domain

\( S = \) is the set of sequential state (locations)

\( e = \) the elapsed time from the last transition

\( Q = \{ (s,e) \mid s \in S, 0 \leq e \leq ta(s) \} \) is the total state set

\( \delta_{\text{int}} : S \rightarrow S \) - internal transition function

\( \delta_{\text{ext}} : Q \times X \rightarrow S \) - external transition function where:

\( \lambda : S \rightarrow Y \) - output function

\( ta : \) time advance function – real and positive, can be 0 and \( \infty \).
Synchronization
– Operational Semantic –

DESS = ∠X, Y, Q, f, λ∠

X = \{(p_c, v_c)|p_c \in \text{InPorts}, v_c \in \mathbb{R}\} set of input ports and their values in the continuous domain
Y = \{(p_c, v_c)|p_c \in \text{OutPorts}, v_c \in \mathbb{R}\} set of output ports and their values in the continuous domain
Q = set of states
f: Q \times X \rightarrow Q is the rate of change function
λ: Q(+X) \rightarrow Y output function
Hétérogénéité continu/discret

\[ q = \delta_{ext}(q) \land \text{synch} = 1 \land \text{flag} = 1 \]

\[
(s_d, e_d) \rightarrow \lambda(s_d); t_a(s_d); \text{flag}:=0 \rightarrow (s_d, e_d); q \rightarrow \lambda(s_d); t_a(s_d); \text{synch}:=0 \rightarrow q
\]
Hétérogénéité continu/discret

\[
\text{synch} = 1 \land \text{flag} = 1 \land q = \delta_{\text{ext}}(q)
\]
\[
(s_d, e_d) \xrightarrow{\lambda(s_d)t_a(s_d); \text{flag}=0} (s_d, e_d); q \xrightarrow{\lambda(s_d)t_d(s_d); \text{synch}=0} q
\]

\[
\text{flag} = 0 \land \neg \text{statevent}(t) \land q' = \delta_{\text{int}}(q)
\]
\[
q \xrightarrow{!\lambda(q)} q' \xrightarrow{!\lambda(q); \text{flag}=1} q'
\]

\[
\text{synch} = 0 \land \text{flag} = 1 \land \text{statevent} \land s'_d = \delta_{\text{ext}}(s_d)
\]
\[
(s_d, e_d) \xrightarrow{t_a(s_d)-e_d} (s_d, t_a(s_d)) \xrightarrow{!\lambda(q); \delta_{\text{int}}(s'_d); \lambda(s'_d); \text{synch}=1} (s'_d, 0)
\]

\[
\text{flag} = 0 \land \text{statevent} \land q' = \delta_{\text{int}}(q)
\]
\[
q \xrightarrow{!\lambda(q)} q' \xrightarrow{!\lambda(q); \text{flag}=1} q'
\]

\[
\text{synch} = 0 \land \text{flag} = 1 \land \text{statevent} \land s'_d = \delta_{\text{ext}}(s_d, t)
\]
\[
(s_d, e_d) \xrightarrow{t_{se}} (s_d, t_{se}) \xrightarrow{!\lambda(q); \delta_{\text{int}}(s'_d); \lambda(s'_d); \text{synch}=1} (s'_d, 0)
\]
Distribution de la fonctionnalité de synchro aux interfaces de co-sim.
Distribution de la fonctionnalité de synchro aux interfaces de co-sim.

- get data from the continuous simulator
- send data to the continuous simulator
- send time of nearest event to the continuous simulator
- activate state event detection module
- send time of state event to the co-simulation bus
- send data from the continuous simulator to the co-simulation bus
- wait data from the discrete simulator
- wait data from the discrete simulator
- was a state event detected

Start ()

Stop

Discrete
Continuous
Co-Simulation Interface
Co-Simulation Interface
Co-Simulation Bus
Verification of Co-Simulation interfaces using UPAAL
Verification of Co-Simulation interfaces using UPAAL
Verification of Co-Simulation interfaces using UPAAL
Verification of Co-Simulation interfaces using UPAAL

Properties verification

- No deadlock
- Timing synchronization
- All state events are detected
- No false state events
Generic Architecture for Continuous/Discrete Co-Simulation

- **Discrete execution model**
- **Continuous execution model**

- **Synchronization layer**
- **Communication layer**

Co-Simulation interface
Generic Architecture for Continuous/Discrete Simulation

- Discrete execution model
  - State events consideration
  - End of discrete sim. cycle detection, event sending
  - Context switch
  - Communication layer

- Continuous execution model
  - Indication of state events and time sending
  - Events detection
  - Context switch
  - Communication layer

Cosimulation bus

Co-Simulation interface
Définition architecture interne des intf. et éléments de la bibliothèque

\[ N = (X, Y, D, \{ Md | d \in D \}, EIC, EOC, IC) \]

- **EIC** (External Input Coupling): \[ EIC = \{ ((N, ip_d), (d, ip_d)) | ip_d \in IPorts_d, d \in D, ip_d \in IPorts \} \]
  - \( X \) - set of input ports and values
  - \( Y \) - set of output ports and values

- **EOC** (External Output Coupling): \[ EOC = \{ ((d, op_d), (N, op_d)) | d \in D, op_d \in OPorts_d, op_d \in OPorts \} \]
  - \( D \) - set of components names

- **IC** (Internal Coupling): \[ IC = \{ ((a, op_a), (b, ip_b)) | a, b \in D, op_a \in OPorts_d, a \in D, ip_b \in IPorts_b \} \]
  - \( Y_d \) - set of output ports and values
CODIS Framework
– Simulation Flow –

Input Flow

Co-Simulation Interfaces Generation

Output Flow

Discrete execution model
Synchronization layer
Communication layer

Continuous execution model
Synchronization layer
Communication layer

Co-Simulation Interface

G. Nicolescu
Generation of Continuous Model Interfaces

Simulink input specification

Co-Simulation Library

- Simulink Co-Simulation Library Blocks
  - State-Event detection and signalling
  - Integration Step Adjustment
  - Synchronization with sampling and update events
  - Communication

Simulink Specification with co-simulation interfaces
Generation of Discrete Model Interfaces

- Based on a SystemC Co-Simulation Library
  - State events management blocks
  - Communication blocks

- Automatic generation of co-simulation interfaces by a code generator that has as input user-defined parameters
  - Data types
  - No. and type of ports
  - Synchronization model
CODIS

- Continuous/Discrete simulation -

- SystemC/Simulink accurate simulation
  - Easy integration, generic library elements

SystemC

Simulink

Speed

HLSLA 2007
G. Nicolescu
CODIS
- Applications -

- Control applications - Robot arm manipulator, Bottle filling system
- Mixed signal application: $\Sigma/\Delta$ Converter
- Wireless application: Radar system
Conclusions

- Continuous/discrete systems designs requires global validation
  - Co-simulation Technique

- Challenges for continuous/discrete co-simulation
  - Definition of global execution models
  - Automatic generation of co-simulation interfaces

- Methodology for co-simulation tools design
  - Application for SystemC/Simulink
Performances analysis

- Inter-Simulators Communication overhead
  - 20% of the total simulation time

- Overhead caused by the Simulink integration step adjustment
  - max. 5% of total simulation time

- SystemC Synchronization overhead
  - max. 0.2% of the total simulation time
## Continuous/Discrete Systems - Synchronization Models -

<table>
<thead>
<tr>
<th>Synchronization model</th>
<th>Synchronization Step</th>
<th>Advantages</th>
<th>Inconvenient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full synchronisation mode</td>
<td>Each discrete step and/or state event occurrence</td>
<td>General</td>
<td>Synchronization overhead</td>
</tr>
<tr>
<td>Predictable events mode</td>
<td>Each update/sampling events and/or state event occurrence</td>
<td>Improve performances</td>
<td>The prediction of update/sample events is required</td>
</tr>
<tr>
<td>Unpredictable events mode</td>
<td>Each update/sampling events and/or state event occurrence</td>
<td>Non-periodic update/sample events</td>
<td>Rollback may be required</td>
</tr>
</tbody>
</table>