CCSL@work:
How to explicitly define MoCCs within a model

AOSTE sophia
I3S/UNS/INRIA

Synchron 2010
CCSL@work: 
the RT-Simex project
(or “a mean to check an implementation against its specification”)

AOSTE sophia
I3S/UNS/INRIA

Frederic Thomas, Étienne Juliot
Obeo

Jean-Philippe Babau
UBO

and more....

Synchron 2010
Logical Time @ work: How to explicitly defines MoCCs within a model

AOSTE sophia
I3S/UNS/INRIA

Synchron 2010
Logical Time...

- focuses on causal relations between events
- Is independent of the abstraction level
- Is multi-clock (polychronous)
- Provides a partial order between events

After 23 starts of the computer, a disk check is done

The computation duration is 156 processor ticks
Logical Time...

- focuses on causal relations between events
- Is independent of the abstraction level
- Is multi-clock (polychronous)
- Provides a partial order between events

After 23 starts of the computer, a disk check is done

The computation duration is 156 processor ticks

In modern laptop, tick is logical since the processor speed depend on the battery level
Logical Time...

- Focuses on causal relations between events
- Is independent of the abstraction level
- Is multi-clock (polychronous)
- Provides a partial order between events
Physical Time...

- Can be seen as a special case of logical time

\[ \text{After 5 seconds, it stops...} \]

\[ \text{≈} \]

\[ \text{After 5 events of the “second” clock, it stops} \]
Modeling: an example

![Diagram of virtual architecture Diopsis model.](image)

**Structure**

**Behavior**

```cpp
class SWFIFO_Ch: public sc_prim_channel{
    list<int> buffer;
    void recv_data(void *dst, int size){
        for (int i=0; i<size; i++)
            dst[i]=buffer.pop;
}

SC_MODULE(TOP){
    ARM_SS *vARM;
    DXM *vDXM;
    BUS *vBUS;
    ...
    SCCTOR(TOP){
}

SC_MODULE(ARM_SS){
    Task1 *vTask1;
    SWFIFO_Ch swfifo1;
    SCCTOR(ARM_SS){
        ...
    }

SC_MODULE(Task1){
    SWFIFO_Port In1;
    ...
    void Task1_beh(); // behavior
    SCCTOR(Task1){
        SC_THREAD(Task1_beh, clk);
        ...
    }
```

Fig. 8. Virtual architecture Diopsis model.
Modeling: an example

Structure

Elaboration phase (SystemC)

Behavior

Simulation

Fig. 8. Virtual architecture Diopsis model.
Traditional modelling approach

- Structure
  - Black-box + Interfaces (*Ports, Data Types*)
- Behavioral abstraction
  - Messages + possibly *period and performance requirements*, ...

- **What we find missing:**
  - Detailed definition of timing and synchronization properties
  - Communication protocol requirements

- **This missing information is often deported elsewhere**
Time & Semantics

• “physical” time
  – Extra functional
  – Single time (total order)
  – Timing constraints to be satisfied at execution
  – Simulation semantics possibly different from synthesis
  – UML, SystemC

HDLs
Time & Semantics

• Logical functional time
  – **Functional**: sequence of reaction steps
  – Multiple times (local / global)
  – Synchronization primitives → constraints between local activation times
  – Synthesis / Compilation
  – Process networks (SDF), synchronous reactive formalisms, statecharts

• “physical” time
  – **Extra functional**
  – Single time (total order)
  – Timing constraints to be satisfied at execution
  – Simulation semantics possibly different from synthesis
  – UML, SystemC

HDLs
Time as semantics

- **Logical functional time**
  - Functional: sequence of reaction steps
  - Multiple times (local / global)
  - Synchronization primitives → constraints between local activation times
  - Synthesis / Compilation
  - Process networks (SDF), synchronous reactive formalisms, statecharts

- **“physical” time**
  - Extra functional
  - Single time (total order)
  - Timing constraints to be satisfied at execution
  - Simulation semantics possibly different from synthesis
  - UML, SystemC

**Our choice**

HDLs

Julien DeAntoni
Logical Time (in MARTE TIME MODEL)

- The main concept is the Clock.
  - It is a way to specify a, possibly infinite, ordered set of instants
  - It can be logical or chronometric, discrete or dense
  - Its type is a ClockType

```plaintext
«clockType»
MyClockType

itsResolution: Integer [1]

«Clock»
itsClock: MyClockType [1]
itsResolution = 1

«Clock»
nature = discrete
unitType = TimeUnitKind
isLogical = true
resolAttr = itsResolution

standard = TAI
type = MyClockType
unit = tick
```
Simplified view of the MARTE Time meta-Model
An extension....

Simplified view of the MARTE Time meta-Model
MARTE TIME MODEL

- Sketchy example of its use

User model

Producer

Consumer

c1
Logical time in models

• Sketchy example of its use

User model

```
Producer
```

```
Consumer
```

sendEvent

```
c1
```

receiveEvent
Logical time in models

• Sketchy example of its use

User model

Producer

sendEvent

Consumer

receiveEvent

MARTE model

LogicalClock : ClockType

IsLogical : True
Nature : discrete

The ordered set of sendEvent is bijective with the ordered set of instants of c1.sendEvent
CCSL

- **Clock Constraint Specification Language**
  - Firstly introduced in the MARTE TIME profile
  - Declarative model-based language integrated with Eclipse
  - Formal semantics (both denotational and operational)
  - Tooled (TimeSquare)

→ Explicitly represents / specifies relations between clocks
CCSL (Clock Constraint Specification Language)

- Relations: dependencies between clocks
  - Coincidence → =
  - Exclusion → ≠
  - Precedence → <
  - Alternance → ~

- Expressions: a mean to create new clocks from others
  - Delay → delayedFor X on aClock
  - Filtering → aClock filteredBy aBinaryWord
  - Union → aClock union anotherClock
  - Intersection → aClock inter anotherClock
  - Periodicity → periodicOn aClock period X offset Y
  - ...
CCSL (Clock Constraint Specification Language)

- Relations: dependencies between clocks
  - Coincidence $\rightarrow =$
  - Exclusion $\rightarrow #$
  - Precedence $\rightarrow <$
  - Alternance $\rightarrow ~$

- Expressions: a mean to create new clocks from others
  - Delay $\rightarrow \text{delayedFor } X \text{ on } aClock$
  - Filtering $\rightarrow aClock \text{ filteredBy } aBinaryWord$
  - Union $\rightarrow aClock \text{ union } anotherClock$
  - Intersection $\rightarrow aClock \text{ inter } anotherClock$
  - Periodicity $\rightarrow \text{periodicOn } aClock \text{ period } X \text{ offset } Y$

- Libraries: user-defined relations and expressions for a specific domain
CCSL

• Sketchy example of its use

User model

Producer

\[c_1.\text{sendEvent}: \text{LogicalClock}\]

on

Consumer

\[c_1.\text{receiveEvent}: \text{LogicalClock}\]

on

MARTE model

\[\text{LogicalClock : ClockType}\]

\[\text{IsLogical : True}\]

\[\text{Nature : discrete}\]

The ordered set of \text{sendEvent} is bijective with the ordered set of instants of \text{c1.sendEvent}
CCSL
• Sketchy example of its use

User model
Producer
c1 sendEvent
Consumer
c1 receiveEvent

MARTE model
LogicalClock : ClockType
IsLogical : True
Nature : discrete

R1 : Coincides
left
right
CCSL
• Sketchy example of its use

User model
Producer
sendEvent
c1
Consumer
receiveEvent

MARTE model
LogicalClock : ClockType
IsLogical : True
Nature : discrete

R1 : Precede
left
right

Julien DeAntoni
CCSL

• Sketchy example of its use

User model

MARTE model

LogicalClock : ClockType

IsLogical : True
Nature : discrete

R1 : Precedes
bound: 3

Julien DeAntoni
CCSL

- Sketchy example of its use

User model

MARTE model
CCSL

• Sketchy example of its use

User model

Producer

sendEvent

c1

Consumer

receiveEvent

MARTE model

LogicalClock : ClockType

IsLogical : True
Nature : discrete

LogicalClock : ClockType

R1 : Precedes
bound: 3

R1 : sample

R1 : delay
d: 3

R1 : period
p: 5
d: 2
b: 3

R1 : sample
p: 3

Julien DeAntoni
CCSL

• Sketchy example of its use

User model

MARTE model

com1 : MyComSemantics

Producer

Consumer

c1

sendEvent

receiveEvent

LogicalClock : ClockType

IsLogical : True
Nature : discrete

b: 3

d: 2

p: 5

R1 : Precedes

bound: 3

R1 : sample

R1 : delay
d: 3

R1 : period
p: 3
CCSL

- Sketchy example of its use

User model

MARTE model

Means to define formally **timed Models of Computations and Communications (MoCCs)**
Why CCSL?

- Means to define formally *timed Models of Computations and Communications* (MoCCs)
- Notation to describe *semantic relations* between timed behaviors (illustrated below)
- Specification of sophisticated *synchronizations*
- *Polychronous* system modeling
- Akin to *Tagged Systems* (Lee & Sangiovanni-Vincentelli)
Synchronous DataFlow

- Nodes are called **actors**
- **Arcs** have a delay
- Input/Output have a **weight** (Number of data samples consumed/produced)
Synchronous DataFlow

SDF Metamodel

SDF Model
Synchronous DataFlow

SDF firing rules:

- **Actor enabling** = each incoming arc carries at least *weight* tokens
- **Actor execution** = atomic consumption/production of tokens by an enabled actor
  - i.e., consume *weight* tokens on each incoming arcs and produce *weight* tokens on each outgoing arc
- **Delay** is an initial token load on an arc.

How can CCSL express this semantics?
SDF Example

Evolutions of the model

Static schedule: A A B A A B C C
How to model SDF graphs in UML?

Where is the semantics?
Is that compatible with the UML semantics?

CCSL makes the semantics explicit ...
... within the model
SDF semantics with CCSL (1/2)

**SDF**
- Actor A
  - Token T
    - write delay read
  - Input i
    - weight i
  - Output o
    - weight

**CCSL**

Clock A;

Clock write, read;
def token(clock write, clock read, int delay) ≜
  write ⊢ (read delayedFor delay)
def input(clock actor, clock read, int weight) ≜
  (read by weight) ⊢ actor
def output(clock actor, clock write, int weight) ≜
  actor ⊢ (write filteredBy \((1.0^\text{weight}-1)^\omega\))
SDF semantics with CCSL (2/2)

- SDF

```
def arc(int delay, clock source, int out, clock target, int in) ≜
    output(source, write, out); token(write, read, delay); input(target, read, in)
```
\[ S = \text{arc}(0, A, 1, B, 2) \mid \text{arc}(0, B, 2, C, 1) \mid \text{arc}(2, C, 1, B, 2) \]
Simulate and animate the UML/MARTE model in TimeSquare
(http://www-sop.inria.fr/aoste/dev/time_square/)
Simulate and animate the UML/MARTE model in TimeSquare
(http://www-sop.inria.fr/aoste/dev/time_square/)
AOSTE’s Tools

- **TimeSquare**
  - Software environment dedicated to the
    - Specification of CCSL constraints
    - Resolution of CCSL constraints
    - Simulation and generation of trace model
    - Animation of UML models
    - Exploration of augmented timing diagrams

- **K-Passa**
  - Computation of static schedules for specific MoCCs
    - Marked Graphs, Synchronous DataFlow, Latency-Insensitive Designs, K-periodical Routed Graphs
  - Analysis (deadlock freeness, safety)
  - Optimization (latency, throughput, interconnect buffer size)
  - Code generation (stand-alone simulator, co-simulation)
Tool download

- http://www-sop.inria.fr/aoste/

**Project-team AOSTE**
Models and methods for the analysis and optimization of systems with real-time and embedding constraints

---

### Softwares

**All our softwares**

*Esterel*: The academic compiler developed by the Tick team.
*KPASSA*: A simple tool to create and use formal models, and run algorithms onto.
*SyncCharts*: a graphical formalism dedicated to reactive system modeling.
*SynDEx*: a system level CAD software based on the “Algorithm-Architecture Adequation” (AAA) methodology, for rapid prototyping and optimizing the implementation of distributed real-time embedded applications.
*TimeSquare*: Eclipse plugin for using MARTE profile (Time model and CCSL) and clock visualization (VCD viewer).
CCSL @ work: the **RT-Simex** project

**AOSTE sophia**

**I3S/UNS/INRIA**

Frédéric Thomas, Étienne Juliots

Obeo

Jean-Philippe Babau

UBO

and more...
Retro-ingénierie de Traces d'analyse de SIMulation et d'EXécution de systèmes temps-réel

RT-Simex
Context: help the designer

Designer → Code → Real time Multi-task execution → Analysis of the execution
Context: execution traces

Designer → Code → Real time Multi-task execution

Execution analysis and reports → Temporal execution traces

JumpShot, Vampir, Linux Tool Eclipse Project

OTF, VCD
observations

Designer → Model → Code → Real time Multi-task execution → Temporal execution traces

SDL, AADL ...

Execution analysis and report
RT-Simex objectives

Execution analysis and report AT THE MODEL LEVEL

Real time Multi-task execution

Temporal execution traces

Designer

Models

Code

UML MARTE

Eclipse
Processus RT-Simex

1. Modèle de conception (contraintes temporelles)
2. Réconciliation et analyse des différences (spécifié vs observé)
3. Modèle des traces temporelles observés
4. Génération de code instrumenté
5. Code source
6. Rétro-ingénierie du code
7. Rétro-ingénierie des traces
8. Exécution
9. Traces au format OTF
Processus RT-Simex

Modèle de conception (contraintes temporelles)

Génération de code instrumenté

Code source

Rétro-ingénierie du code

Exécution

Rétro-ingénierie des traces

Modèle des traces temporelles observés

Réconciliation et analyse des différences (spécifié vs observé)

Traces au format OTF
Processus RT-Simex

CCSL specification

Rétro-ingénierie des traces

Rétro-ingénierie du code

Génération de code instrumenté

Code source

Exécution

Traces au format OTF

Réconciliation et analyse des différences (spécifié vs observé)

Modèle des traces temporelles observés
CCSL (Clock Constraint Specification Language)

- Relations: dependencies between clocks
  - Coincidence → =
  - Exclusion → #
  - Precedence → <
  - Alternance → ~

- Expressions: a mean to create new clocks from others
  - Delay → delayedFor X on aClock
  - Filtering → aClock filteredBy aBinaryWord
  - Union → aClock union anotherClock
  - Intersection → aClock inter anotherClock
  - Periodicity → periodicOn aClock period X offset Y

- Libraries: user-defined relations and expressions for a specific domain
Graphical formal annotation over a UML model for RT-Simex...
Processus RT-Simex

CCSL specification

Génération de code instrumenté

Code source

Rétro-ingénierie du code

Exécution

Rétro-ingénierie des traces

Traces au format OTF
Processus RT-Simex

“Clock instant” must be monitored at runtime

CCSL specification

Réconciliation et analyse des différences (spécifié vs observé)

Modèle des traces temporelles observés

Génération de code instrumenté

Code source

Rétro-ingénierie du code

Rétro-ingénierie des traces

Exécution

Traces au format OTF
Processus RT-Simex

CCSL specification

For a specific platform, this is a total order...

Génération de code instrumenté

Code source

Rétro-ingénierie du code

Exécution

Rétro-ingénierie des traces

Modèle des traces temporelles observés

Traces au format OTF

Réconciliation et analyse des différences (spécifié vs observé)
For a specific platform, this is a total order... For two communicating platforms, the traces are multi-clocks and must be synchronized... The result is a partial order
Processus RT-Simex

CCSL specification

Génération de code instrumenté

Code source

Rétro-ingénierie du code

Exécution

Traces au format OTF

Modèle des traces temporelles observés

Réconciliation et analyse des différences (spécifié vs observé)

If simulated, it replays the execution...
Processus RT-Simex

If simulated it replays the execution...
So you can have a timing diagram, animate the model and so on...

CCSL specification
A specification and the actual execution
Processus RT-Simex

CCSL specification

Génération de code instrumenté

Code source

Exécution

Rétro-ingénierie du code

Modèle des traces temporelles observés

Réconciliation et analyse des différences (spécifié vs observé)

Traces au format OTF

CCSL specification
Processus RT-Simex

CCSL assertions

Réconciliation et analyse des différences (spécifié vs observé)

Modèle des traces temporelles observés

CCSL specification

Génération de code instrumenté

Code source

Rétro-ingénierie du code

Exécution

Traces au format OTF
You can simulate both together to see if a violation occurs.
Processus RT-Simex

You can simulation both together to see if a violation occurs

CCSL assertions

CCSL specification

Violation of “Clk3b isSubClockOf clk3”
Conclusion

Two uses of CCSL:

➔ A formal language to specify MoCC explicitly within a model,
  – MOCC library construction,
  – No hidden rules or implementation
  – Directly linked to the model

➔ A formal language to specify real-time properties AND to check the correctness of a specific implementation
  – Possibility to specify construct for a specific domain,
  – Possibility to change the role of a CCSL model, from specification, to assertion.
Some future ongoing works

- Heterogeneous systems
  - Clocks and CCSL are a way to explicitly glue various MoCC
- Use of CCSL for Power Domain
- Add of user defined simulation policy
  - EDF, RMA and all you can imagine (use of back-end)
- Optimization of the TimeSquare solver
  - Optimize BDD depending on clock domains
- Instant refinement
- ...

Julien DeAntoni
Some future ongoing works

- Heterogeneous systems
  - Clocks and CCSL are a way to explicitly glue various MoCC
- Use of CCSL for Power Domain
- Add of user defined simulation policy
  - EDF, RMA and all you can imagine (use of back-end)
- Optimization of the TimeSquare solver
  - Optimize BDD depending on clock domains
- Instant refinement
- ...

Julien DeAntoni
Some future ongoing works

- Heterogenousous systems
  - Clocks and CCSL are a way to explicitly glue various MoCC
- Use of CCSL for Power Domain
  - Add of user defined simulation policies
    - EDF, RMA and all you can imagine (use of back-end)
  - Optimization of the TimeSquare solver
    - Optimize BDD depending on clock domains
- Instant refinement
- ...
Thank you!