Formal refinement -- Embedded system

Hocine Mokrani

LABSOC (Telecom-ParisTech)

hmokrani@telecom-paristech.fr

Sophia Antipolis

12-10-2011

Objective

Our work aims

to improve the architecture exploration phase in order to produce reliable SoC systems using formal methods.

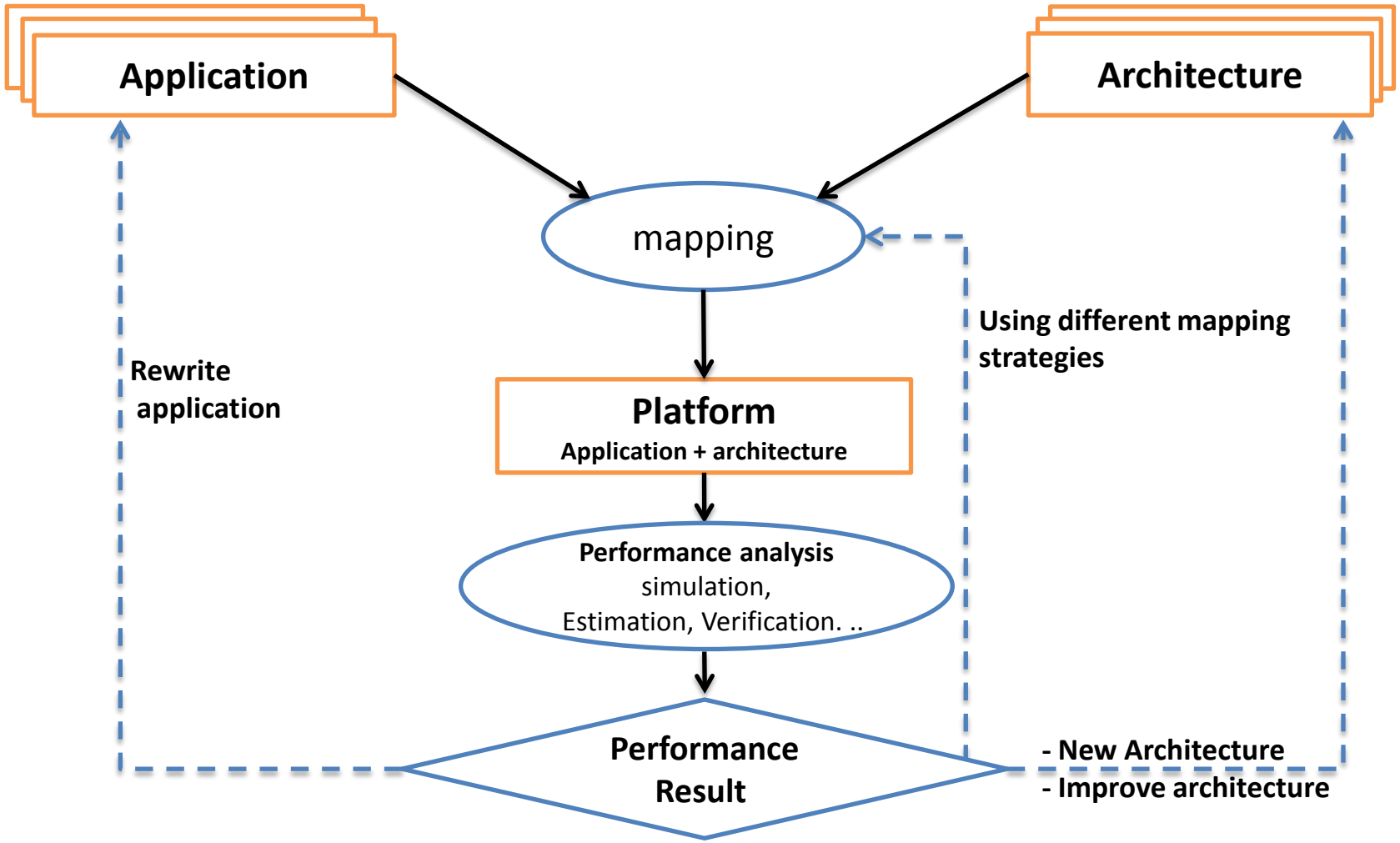
Plan

- Exploration architecture.
- Our approach For EA.
- Illustration.
- Conclusion.

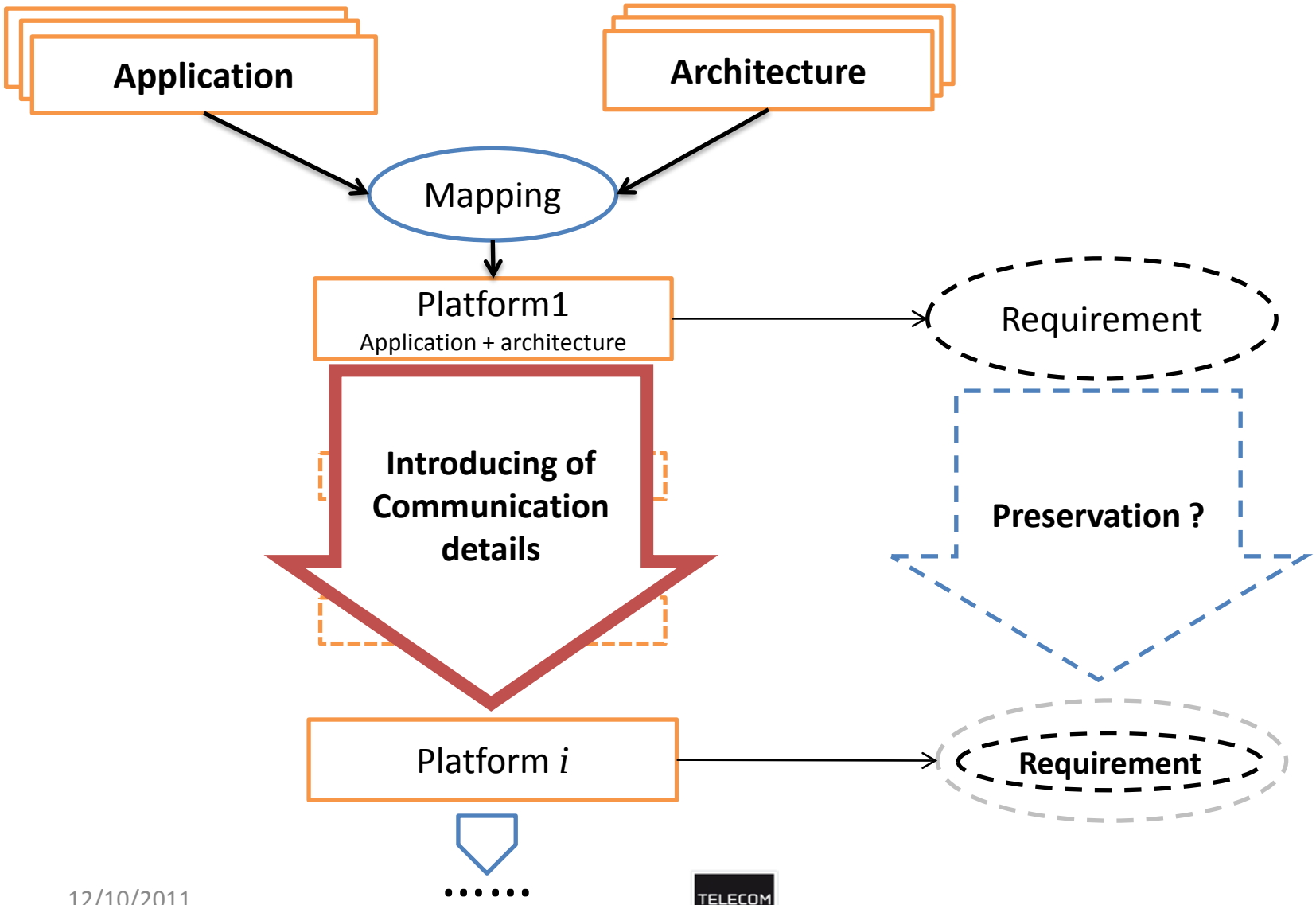
Architecture Exploration

Select from the different architectural alternatives the one that has the best answer to the objectives and requirements set (The good functioning of the system, Optimum performance).

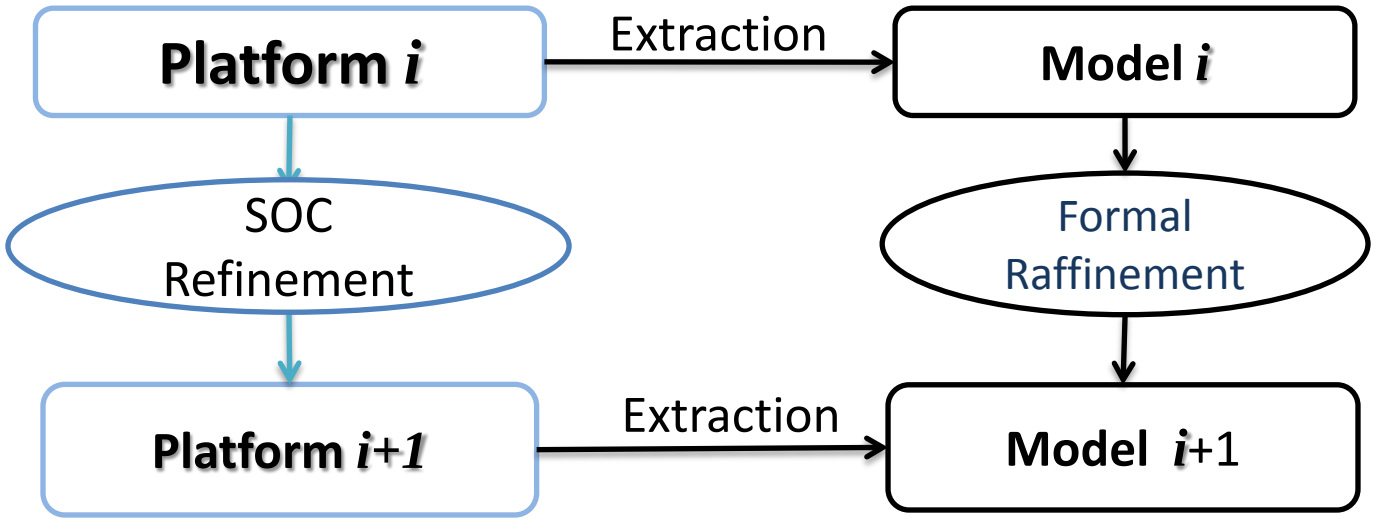
Architecture Exploration methodology



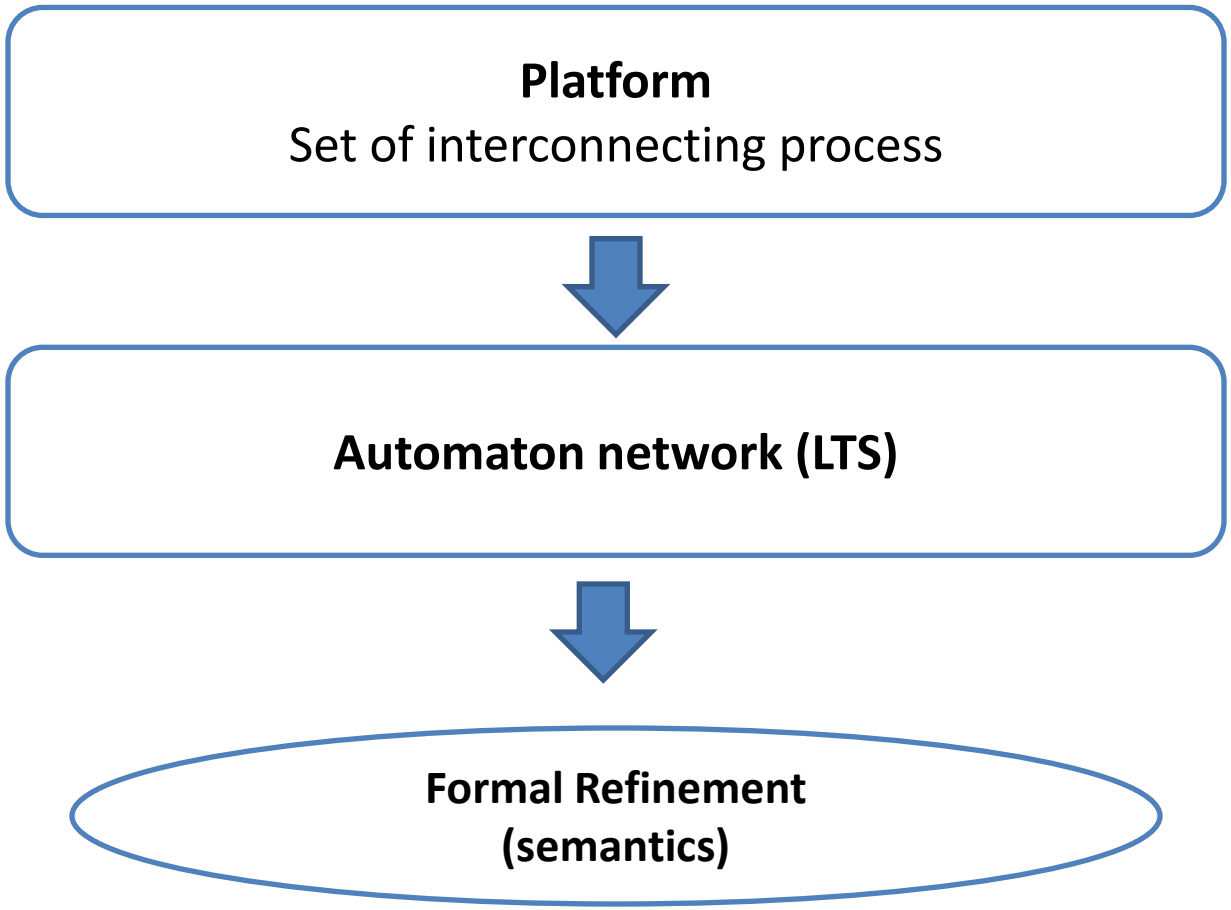
Refinement problem



Formalisation



Models

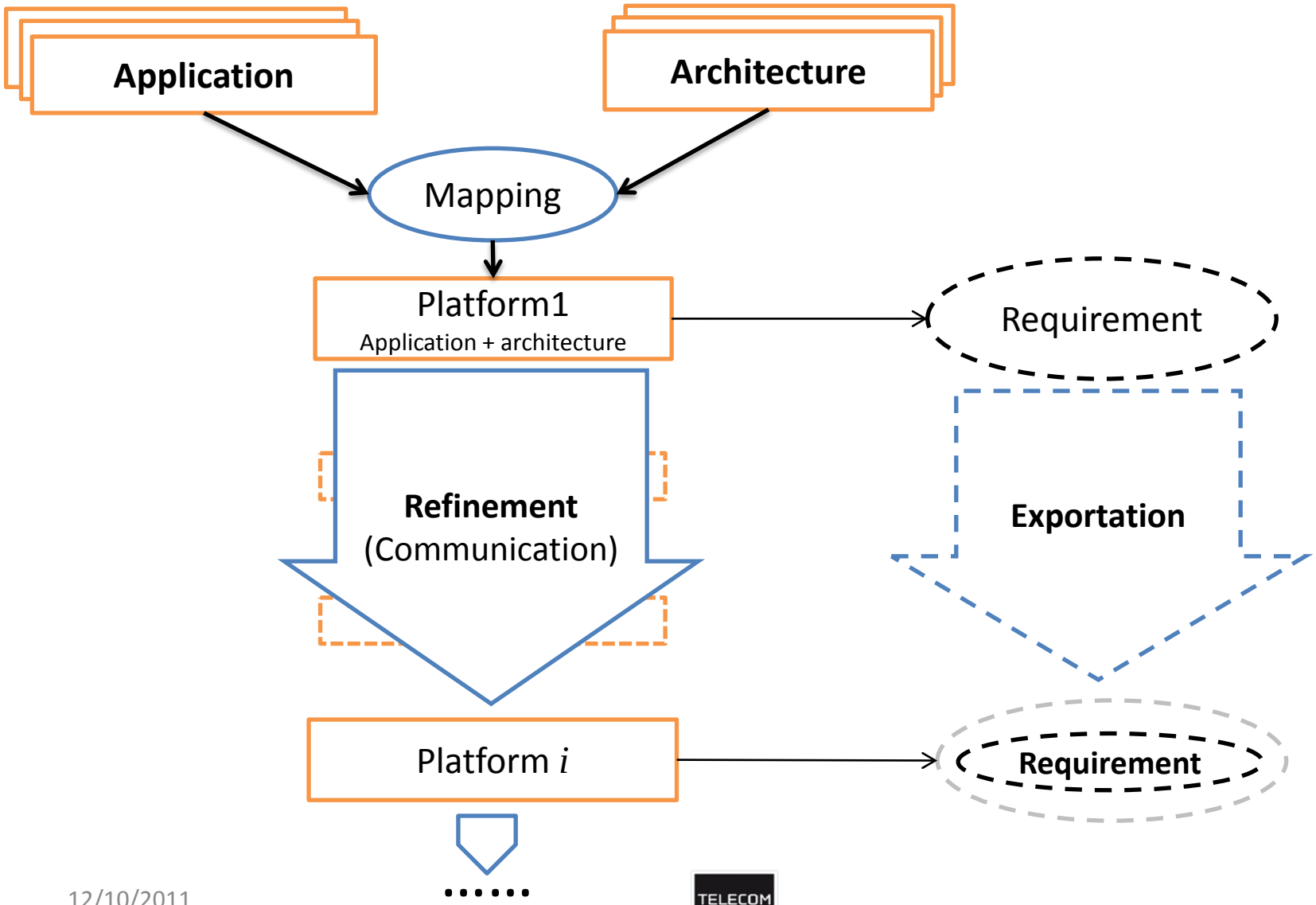

Platform
Set of interconnecting process
Automaton network (LTS)
Formal Refinement
(semantics)

Semantics

1. • Trace
   • P1 $\sqsubseteq_{Trace}$ P2 $\Leftrightarrow$ Trace(P2) $\subseteq$ Trace(P1)
2. • Complet Trace
   • P1 $\sqsubseteq_{CTrace}$ P2 $\Leftrightarrow$ Ctrace(P2) $\subseteq$ Ctrace(P1)
3. • Failure
   • P1 $\sqsubseteq_{Failure}$ P2 $\Leftrightarrow$ Failure(P2) $\subseteq$ Failure(P1)
4. • Readiness
   • P1 $\sqsubseteq_{Readiness}$ P2 $\Leftrightarrow$ Readiness(P2) $\subseteq$ Readiness(P1)

[1] J.R van Glabbeek, *The linear time-branching time spectrum I; the semantics of concrete, sequentiel processes. In Handbook of process Algebra, chapitre1, pages 3-99,Elsevier, 2001*

Transformation in action



Task Modeling Language (TML)

- ✔ Abstract architectural details.
- ✔ Abstract calculation data.
- ✔ TML Modeling the system by a tasks network communicating via channels.

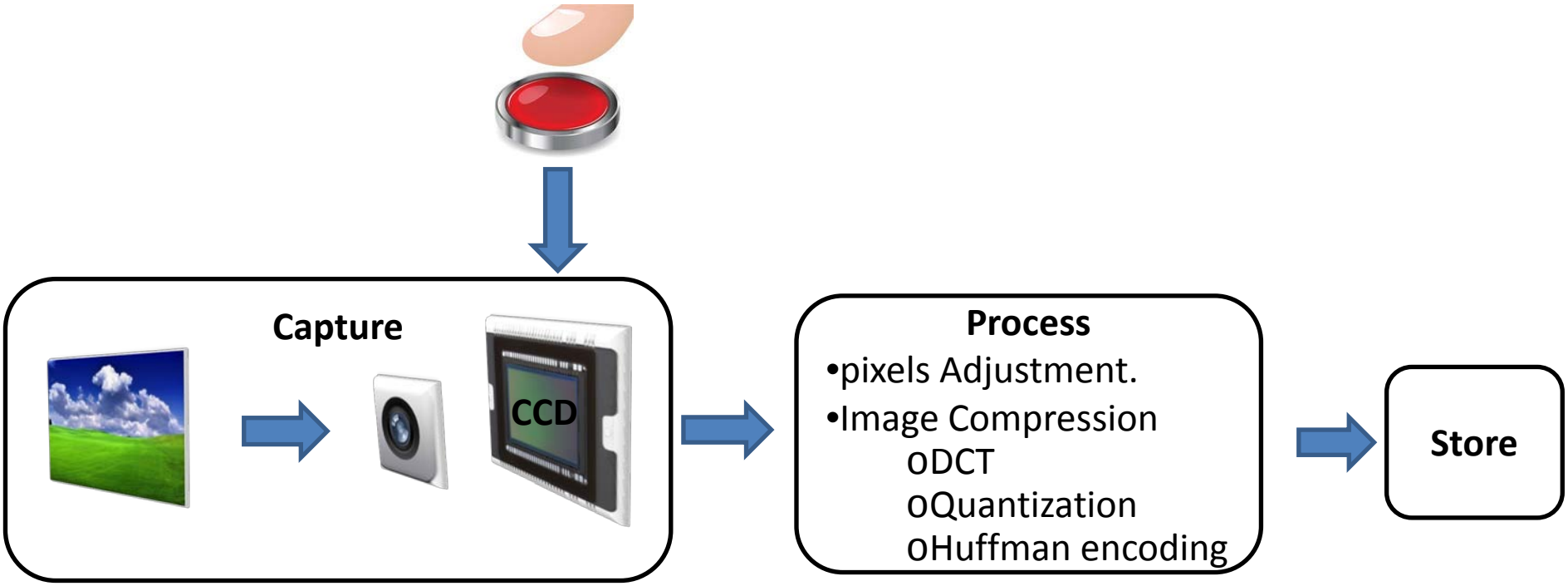
  •***Task***: "Calculus/communication" Instructions, variable Setting, Tests and loops.

  •***Channels:***

  - •Channels: BR-NBW, NBR-NBW, BR-BW
  - •Events: FIFO fini, FIFO infini.
  - •Requests: FIFO infinie.

Example of real Application

Example of real Application



- **CCD** : simulate the capture of picture. (picture generator)
- **CCDPP** : pixels Adjustment.
- **CODEC**: picture compression.
- **CNTRL**: system control and data routing.
- **UART**: simulate data exchange with external peripheral.

TML application

- **Channel** Image1 , CCD,CCDPP,1;
- **Channel** Image2,CCDPP,CNTRL,1;
- …….

- **Event** Start1, CNTRL,CCDPP,1;
- **Event** Start2 , CCDPP,CCD,1;

- **Task** CCD {

  Wait(Start2);
  Exec;
  write(Image1);
  }

- **Task** CODEC

```
{
Repeat(N times)
   Repeat(M times)
         Read(Block);
         Exec;
         write(Block);
   Endrepeat
Endrepeat
}
```

- **Task** CCDPP{

```
Wait(Start1);
Notify(Start2);
Read(Image1);
Exec;
write(Image2);
}
```

- **Task** UART

```
{
 Read(Image3);
Exec;
}
```

- **Task** CNTRL

```
{
Notify (Start1);
Read(Image2);
Repeat(N times)
   Repeat(M times)
         Write(Block1);
         Read(Block2,);
          Exec;
    Endrepeat
Endrepeat
write(Image3);
}
```

TML application

- **Channel** Image1 , CCD,CCDPP,1;
- **Channel** Image2,CCDPP,CNTRL,1;
- …….

- **Event** Start1, CNTRL,CCDPP,1;
- **Event** Start2 , CCDPP,CCD,1;

- **Task** CCD {
  Wait(Start2);
  Exec;
  write(Image1);
  }
- **Task** CODEC

```
{
Repeat(N times)
   Repeat(M times)
         Read(Block);
         Exec;
         write(Block);
   Endrepeat
Endrepeat
}
```

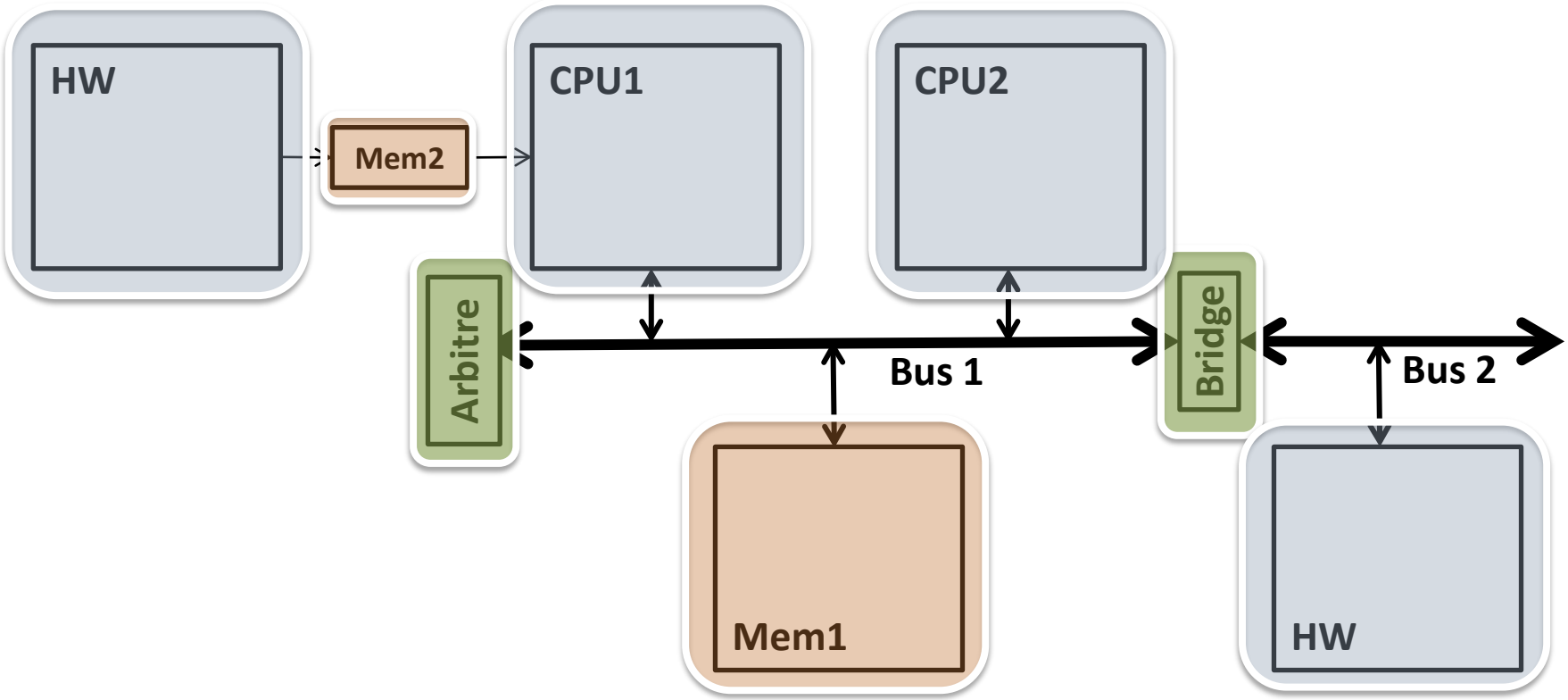
- **Task** CCDPP{

Wait(Start1);
Notify(Start2);
Read(Image1);
Exec;
write(Image2);
}

- **Task** CNTRL

```
{
Notify (Start1);
Read(Image2);
Repeat(N times)
   Repeat(M times)
         Write(Block1);
         Read(Block2,);
         Exec;
   Endrepeat
Endrepeat
write(Image3);
}
```

Architecture



PE (Processing Element)

SE (Storage Element)

IF (Interface Element)

CE (Communication Element)

Mapping

Platform = Architecture ⊕ Application

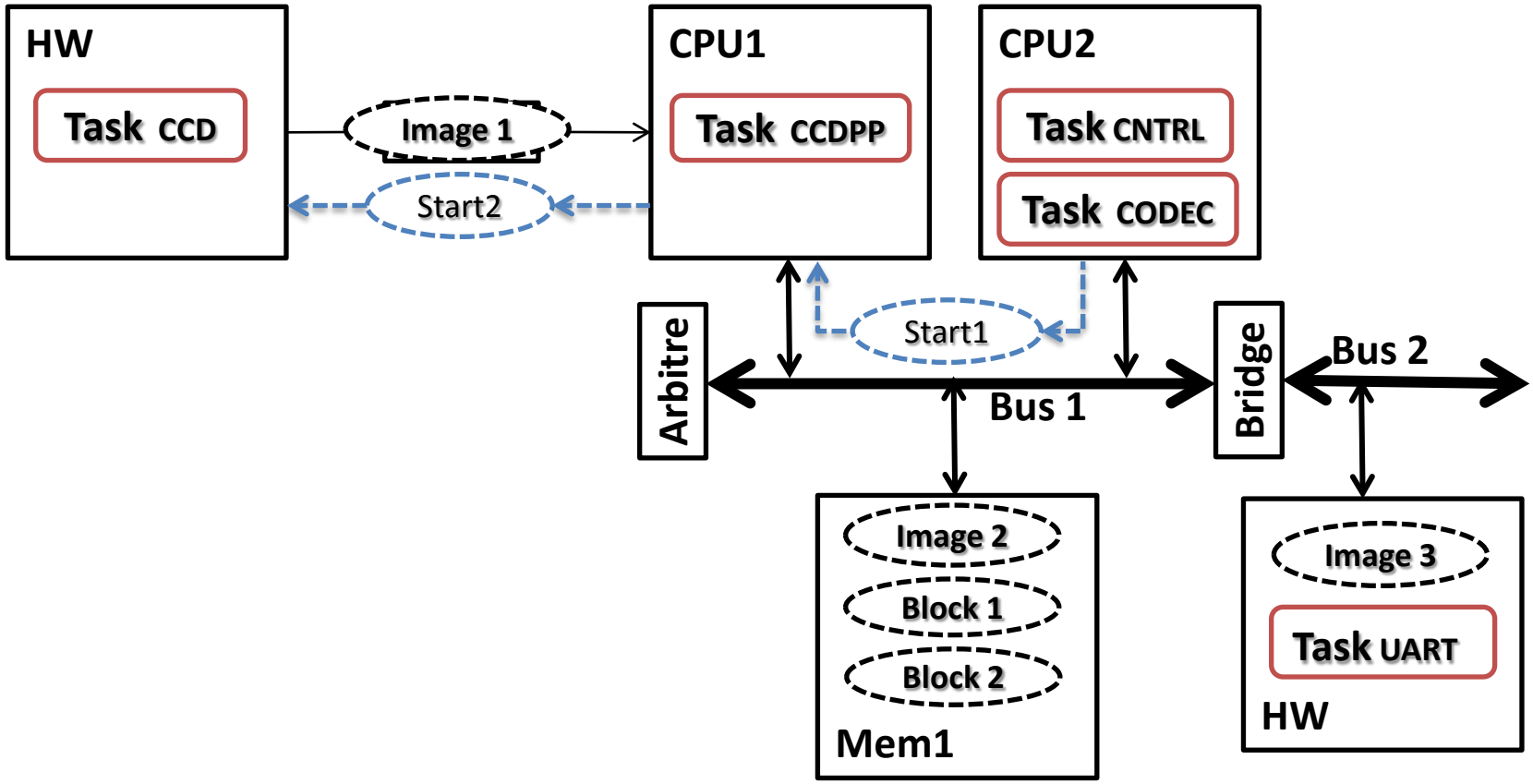


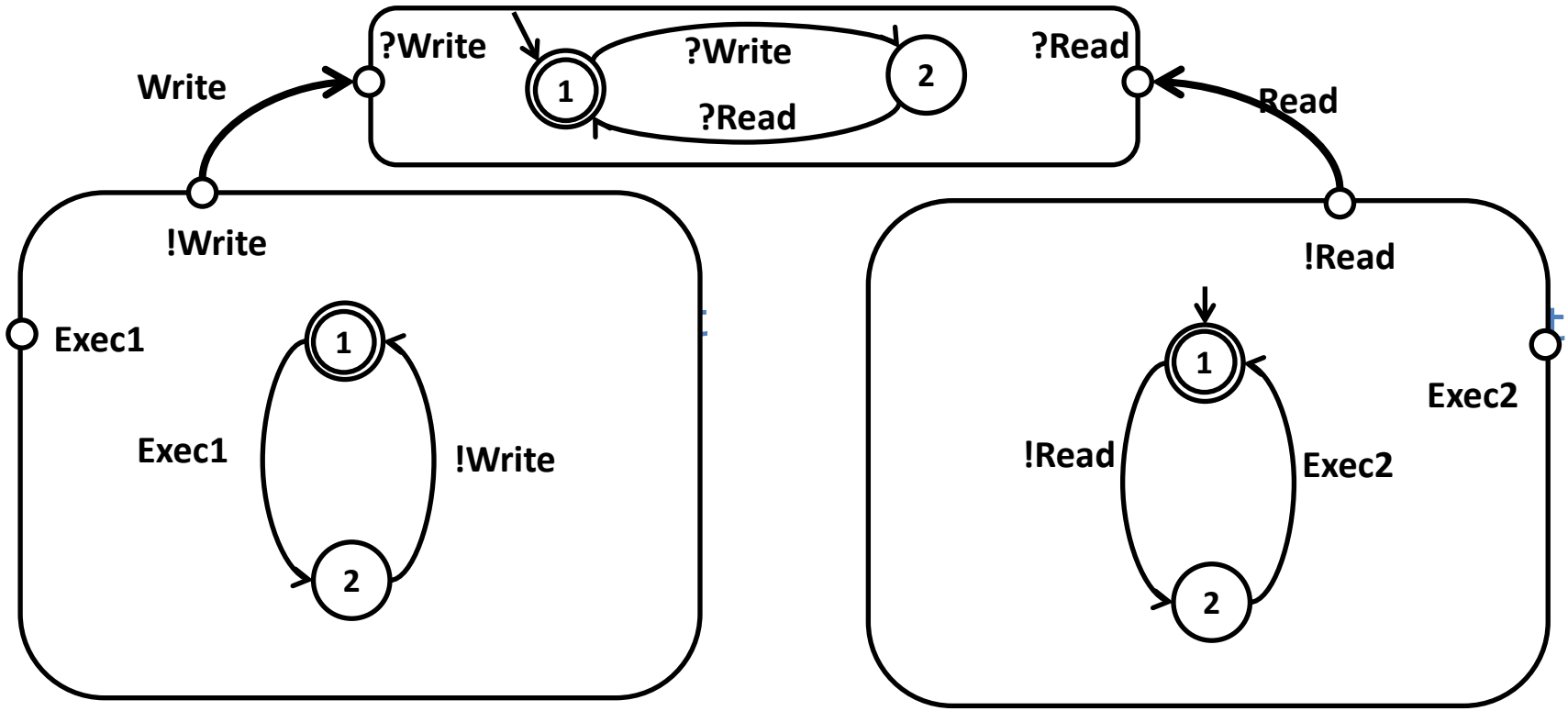
Illustration example

```
CHANNEL Buffer, BRBW, 1
```

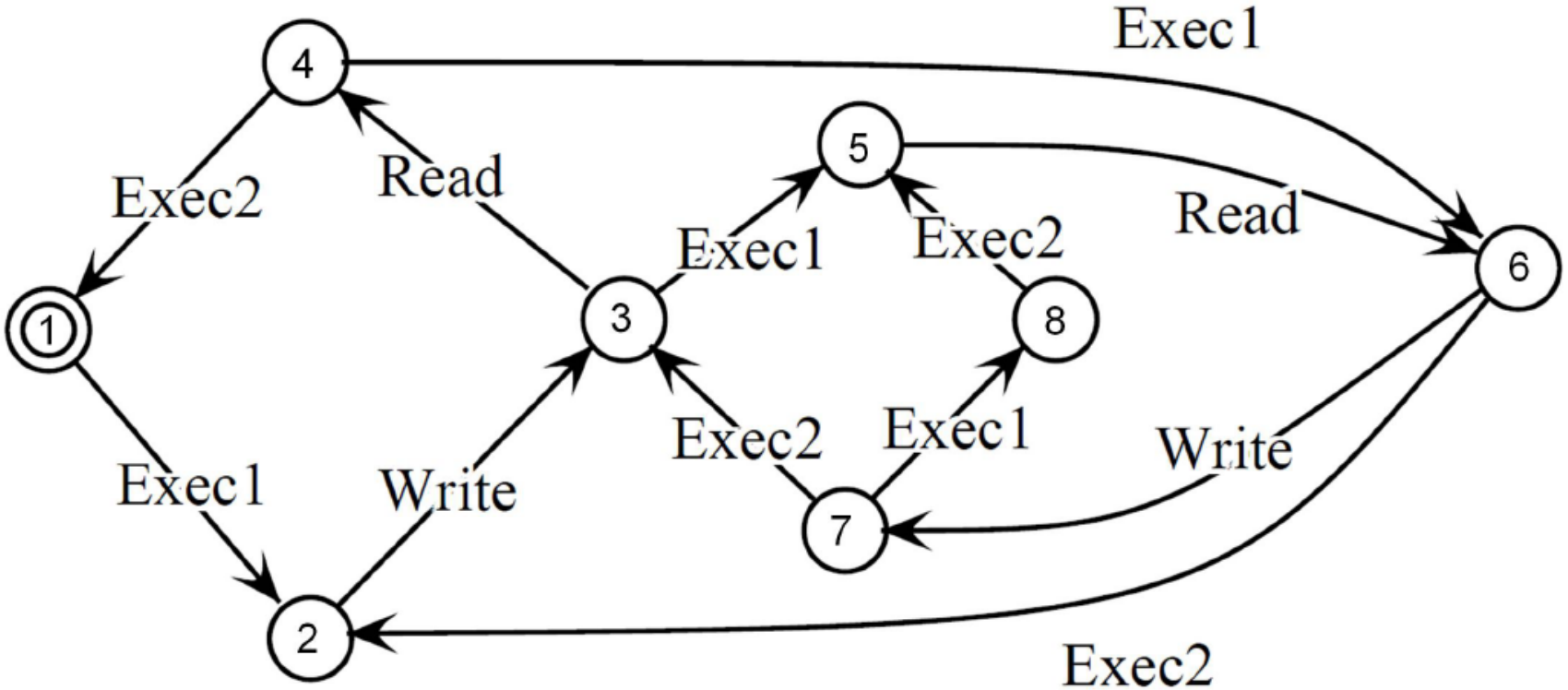
```
TASK Task1{
channel Buffer:output
  REPEAT
    EXEC
    WRITE buffer
  ENDREPEAT
}
```

```
TASK Task2{
  channel Buffer:input
  REPEAT
    READ Buffer
    EXEC
  ENDREPEAT
}
```

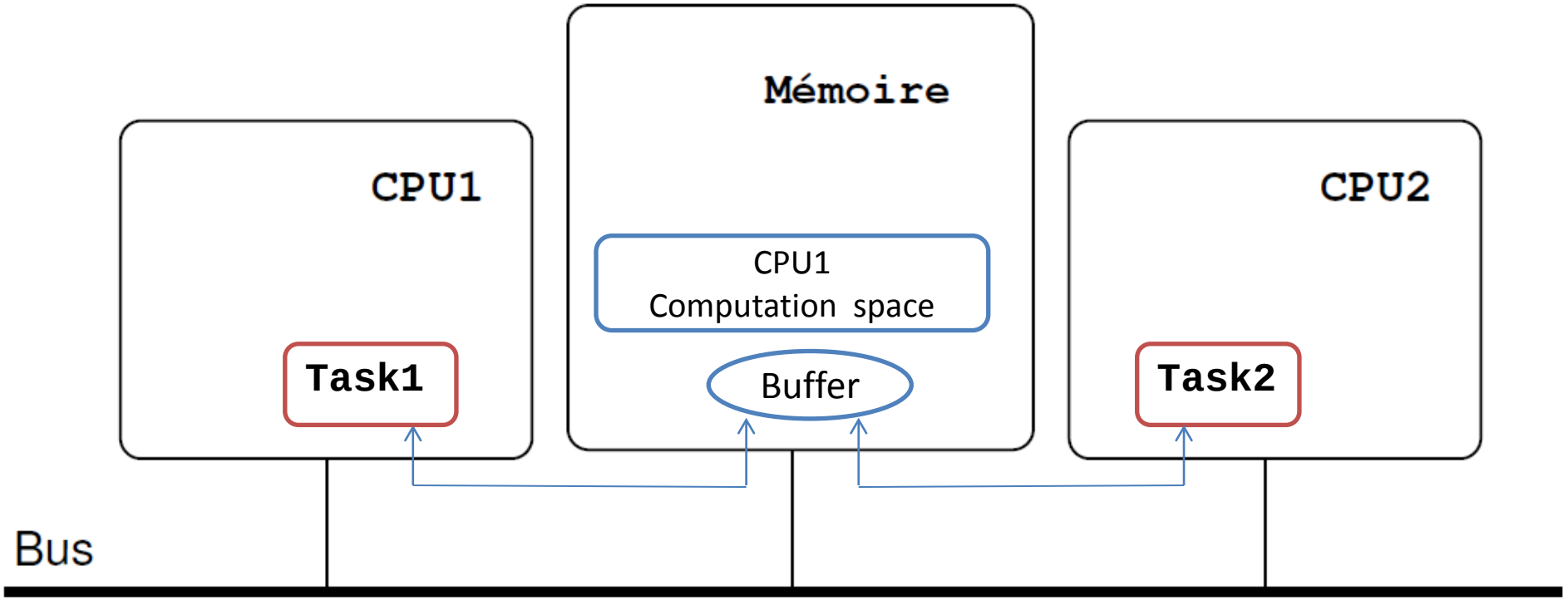
Illustration example



Global model of application



Architecture and mapping



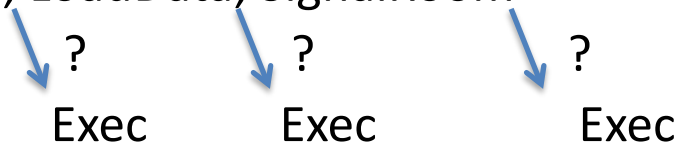
Abstract Model VS Concrete Model

- **Abstract action ≜ (One/Sequence) Concrete action**

  Read ≜ CheckSignalData; LoadData; SignalRoom.

- **Combinatory Abstract action ≜ Combinatory Concrete action**

  Read; Exec ≜ CheckSignalData; LoadData; SignalRoom

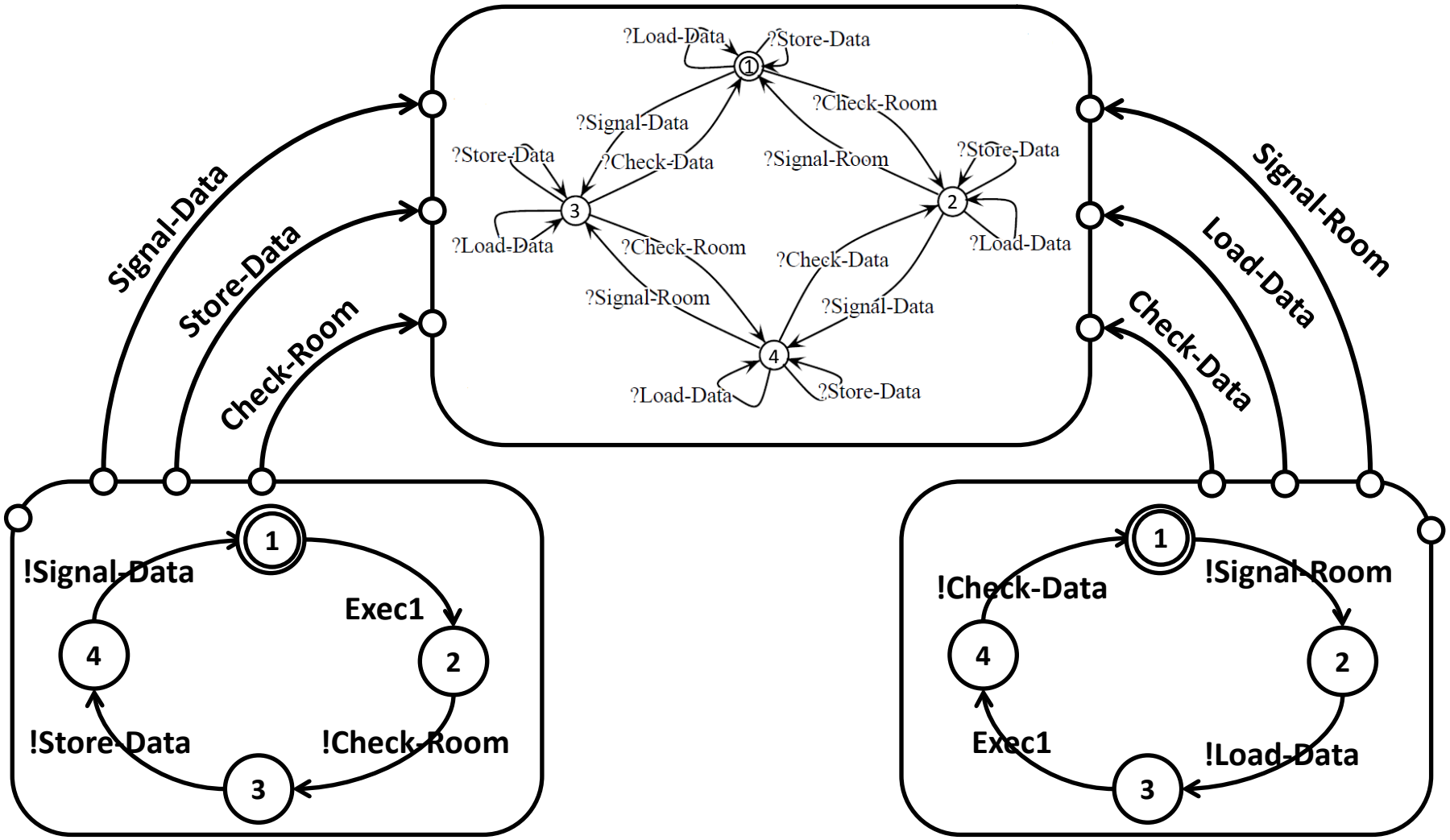
- **Semantic correspondence**

  Read ↦ LoadData,

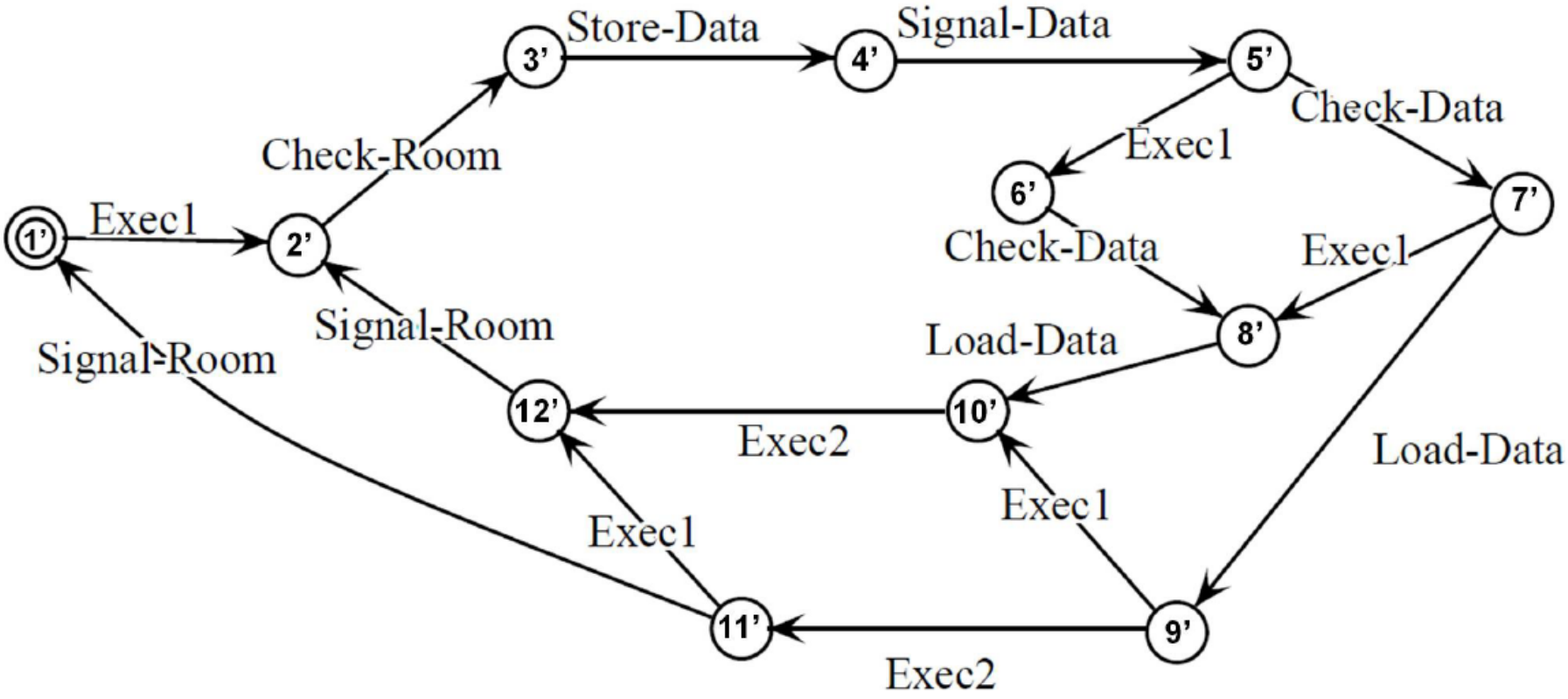
  Write ↦ StoreData,

  Exec ↦ Exec

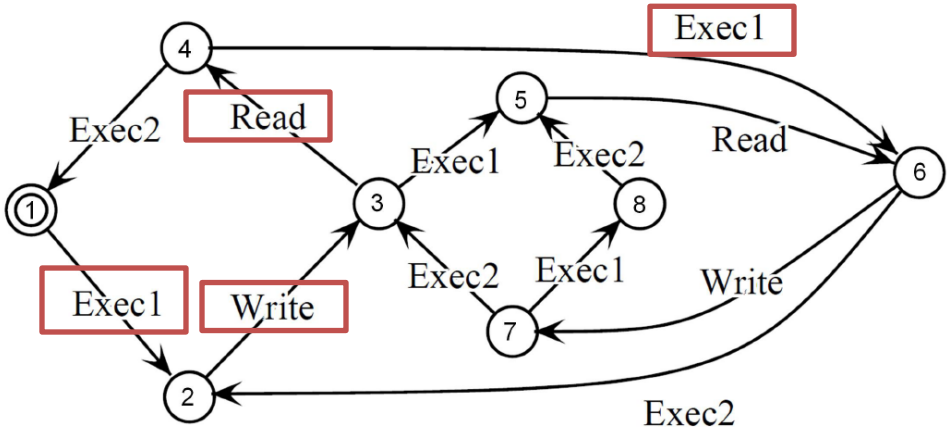
Application on architecture



Global model of platform



Refinement analysis

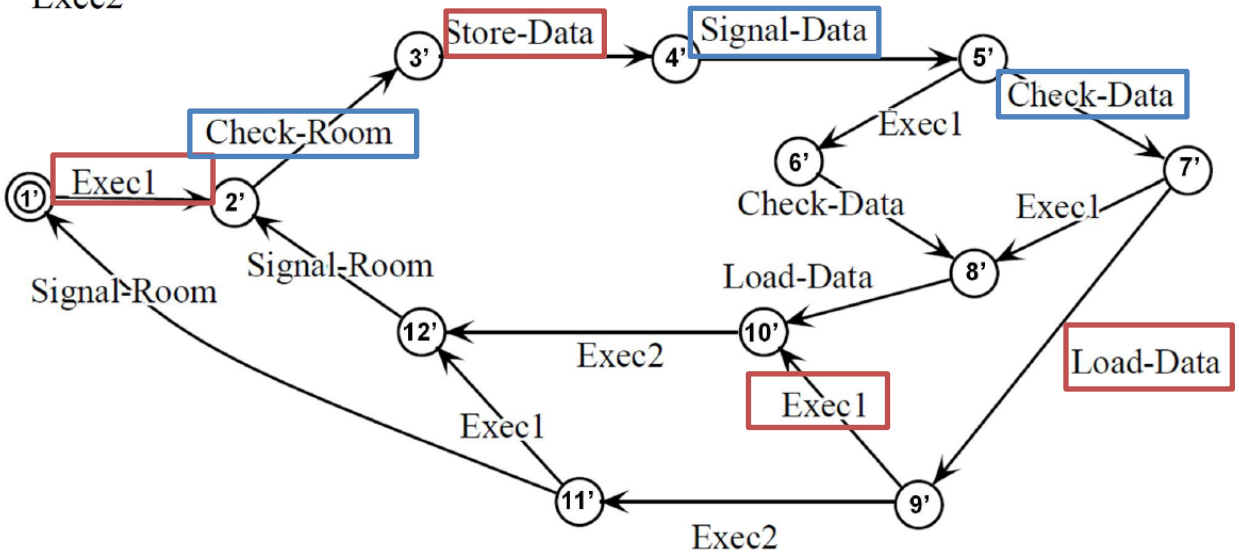


Application

$Sys_1 \sqsubseteq_{ICTrace} Sys_2$

$Sys_1 \not\sqsubseteq_{Failure} Sys_2$

$Sys_1 \not\sqsubseteq_{Readiness} Sys_2$



Platform

Model	Proof

Application

/*TML specification*/

Architecture

/*Interconnected Component*/

Mapping

/*mapping roles*/

/*Platform = Architecture ⊕ Application*/

Requirement

/*Properties description (LTL ,CTL, ACTL……..)*/

Model	Proof

Application

Tasks : (Task1, Task2)

Canals: Buffer

....

Mapping

PEs: CPU1 <- Task1 ; CPU2 <- Task2

CEs: ...

Requirement:

P1: Write $\Rightarrow \mathbf{X}(\neg$ Write $\boldsymbol{U}$ Read)

P2: $\boldsymbol{EF}$ (Read $\wedge \boldsymbol{AX}$ (Exec1 $=>$ ($\boldsymbol{EX}$ Write $\wedge$ $\boldsymbol{EX}$ Exec2)))

Model	Proof

Requirement:

Application

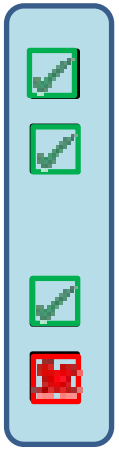
P1: ***F*** Write ⇒(¬ Write ***U*** Read)

P2: ***EF***(Read ⋀ ***AX*** Exec1) ⇒(***EX*** Write ⋀ ***EX*** Exec2)

Platform

P1: ***F*** StoreData ⇒(¬ StoreData ***U*** LoadData)

P2: ***EF***(LoadData ⋀ ***AX*** Exec1) ⇒(***EX*** StoreData ⋀ ***EX*** Exec2)



Model	Proof

Requirement:

Application

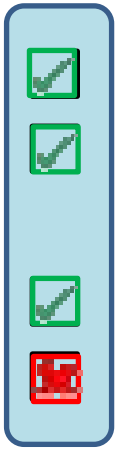
P1: ***F*** Write ⇒(¬ Write ***U*** Read)

P2: ***EF***(Read ⋀ ***AX*** T1.Exec) ⇒(***EX*** Write ⋀ ***EX*** T2.Exec)

Platform

P1: $Sys_1 \sqsubseteq_{\text{ICTrace}} Sys_2$

P2: $Sys1 \not\sqsubseteq_{\text{Readiness}} Sys2$



Conclusion

Our goal is to provide a rigorous framework to assist SoC transformation between different levels.

- Define elementary communication transformations.
- Identify their order.
- Determine the preserved semantics and remaining proof obligation.
- Tool Implementation.

Thank you