

# SoCLib : Une plate-forme de prototypage virtuel pour systèmes multi-processeurs intégrés sur puce

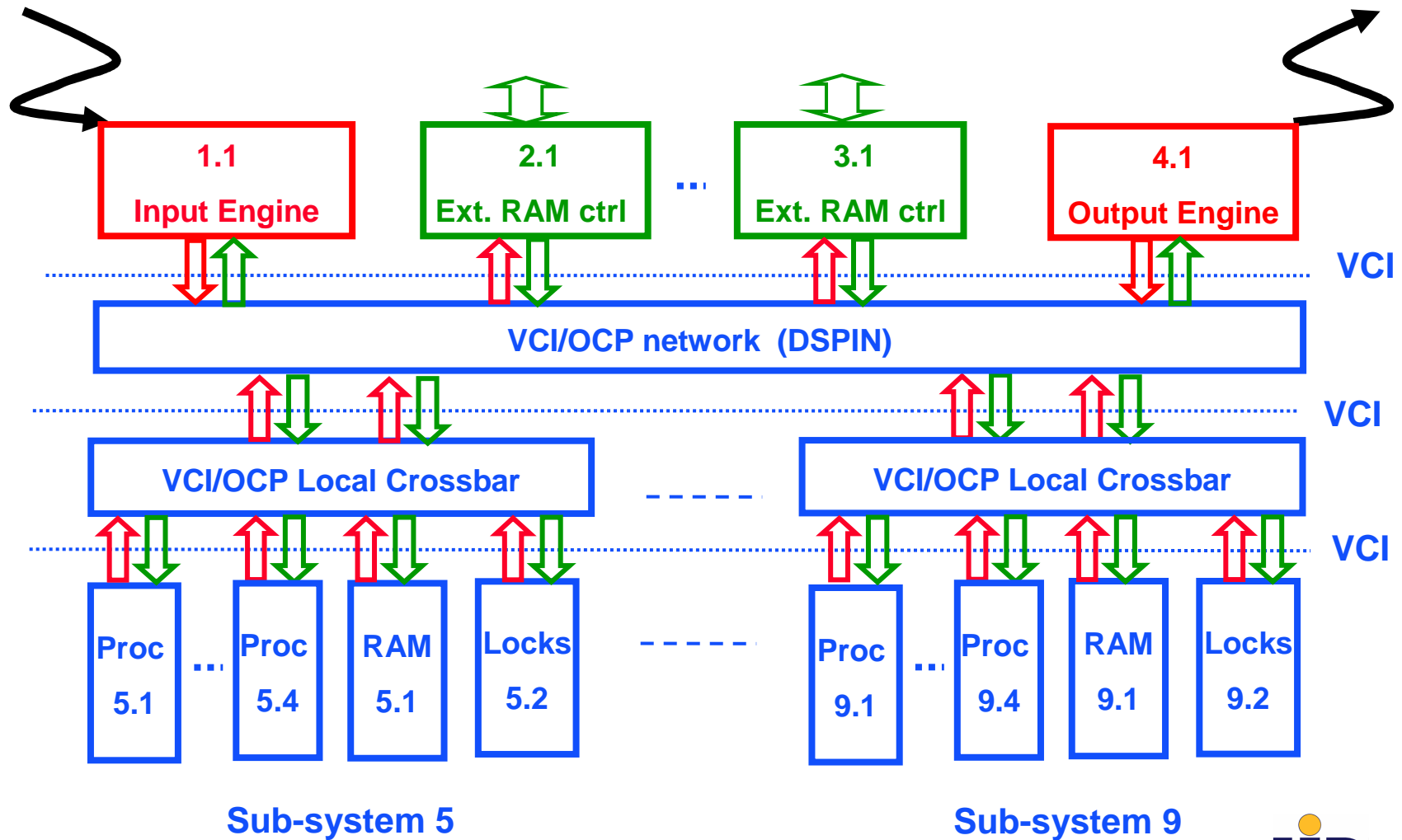
Alain Greiner

# Outline

---

- © SoCLib goals
- © SystemC modeling principles
- © The Mutek Real-Time Operating System
- © The MWMR communication middleware
- © The DSX hardware/software codesign tool
- © An application example : Stereovision

# Example of a multi-processor architecture



# GOALS

---

- © **Design Space Exploration** : fast performance evaluation for the mapping of a multi-threaded software application on a multi-processor hardware architecture.
- © **Plat-form based design** : provide the system designer one (or several) generic hardware/software multi-processor architecture(s).
- © **IP reuse** : clear separation between computing resources and communication resources : VCI/OCP standard.

# SoCLib : Industrial Partners

---

© ST Micro-electronics	Grenoble	(B.Jego)
© Thales	Paris	(P.Kajfasz)
© Thomson SC	Rennes	(H.Heijnen)
© Silicomp	Grenoble	(V.Joloboff)
© Prosilog	Cergy	(M.Saussay)
© TurboConcept	Brest	(J.Tousch)

# SoCLib : Académic Partners

---

LIP6	Paris	(A.Greiner)
INRIA	Orsay	(O.Temam)
LISIF	Paris	(P. Garda)
CEA-LIST	Saclay	(F.Derepas)
ENST	Paris	(A.Polti)
IRISA	Rennes	(F.Charot)
LESTER	Lorient	(P.Coussy)
IETR	Rennes	(D. Houzet)
CEA-LETI	Grenoble	(J-R. Lequepeys)
TIMA	Grenoble	(F.Petrot)
CITI	Lyon	(T. Risset)

# SoCLib : General Principles

---

- © The modeling language is SystemC.
- © Two simulation models for each hardware component:
  - CABA (Cycle-Accurate / Bit-Accurate)
  - TLMT (Transaction Level Model with Time)
- © All hardware components respect the same (VCI/OCP) communication protocol.
- © All simulation models will be available as free software.
- © For each SoCLib component, there is a - commercially available - synthesizable RTL model.

# SoCLib : Planned Components

	COMPONENTS	ORIGINE
<b>General Purpose Processors</b>	MIPS R3000 SPARC V8 OpenRISC 1000 ARM 7 ARM 9 PowerPC 450 PowerPC 750	LIP6 ENST LIP6 LIP6 CEA LRI CEA
<b>Digital Signal Processors</b>	ST200 C6X Texas	ST / LRI IRISA
<b>System Utilities</b>	Interrupt controler Locks engine DMA controler MWMR controler	LIP6 LIP6 IETR LIP6
<b>VCI/OCP Interconnects</b>	VCI Generic VCI on DSPIN VCI on AMBA VCI on AVALON	LIP6 LIP6 PROSILOG IRISA

# SoCLib : Planned Components

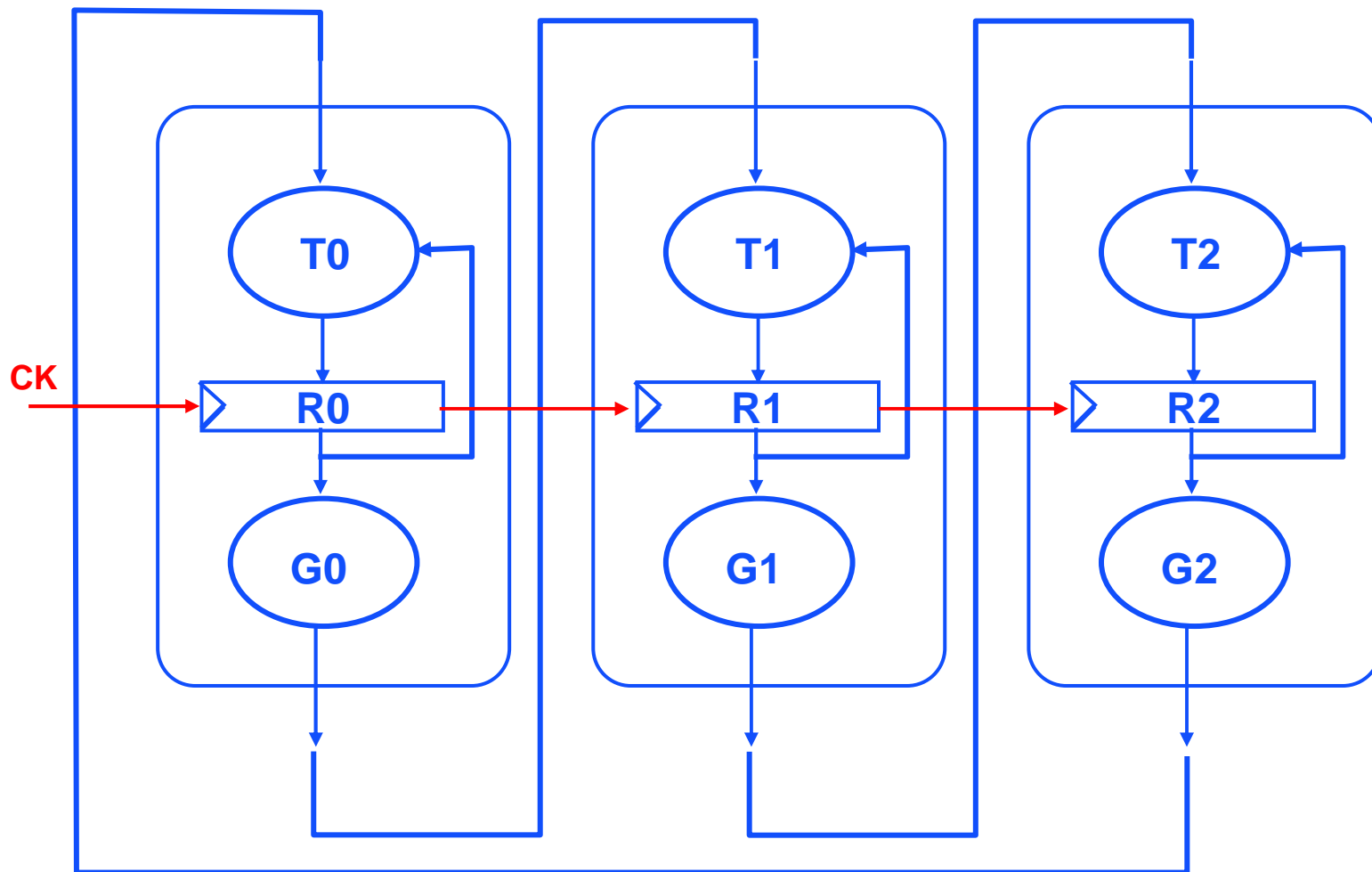
	COMPOSANT	ORIGINE
<b>Memory Controlers</b>	Embedded SRAM External SDRAM External DDR/SDRAM	LIP6 PROSILOG PROSILOG
<b>Dedicated Coproprocessors</b>	Turbocodeur TC1000 Turbocodeur TC3000 LDPC DWT MOD DEM	TURBOCONCEPT TURBOCONCEPT ENST LESTER LESTER LESTER
<b>I/O Controlers</b>	VCI / PCI bridge VCI / USB bridge VCI / CAN bridge VCI / I2C bridge	IETR ENST ENST LISIF

# Outline

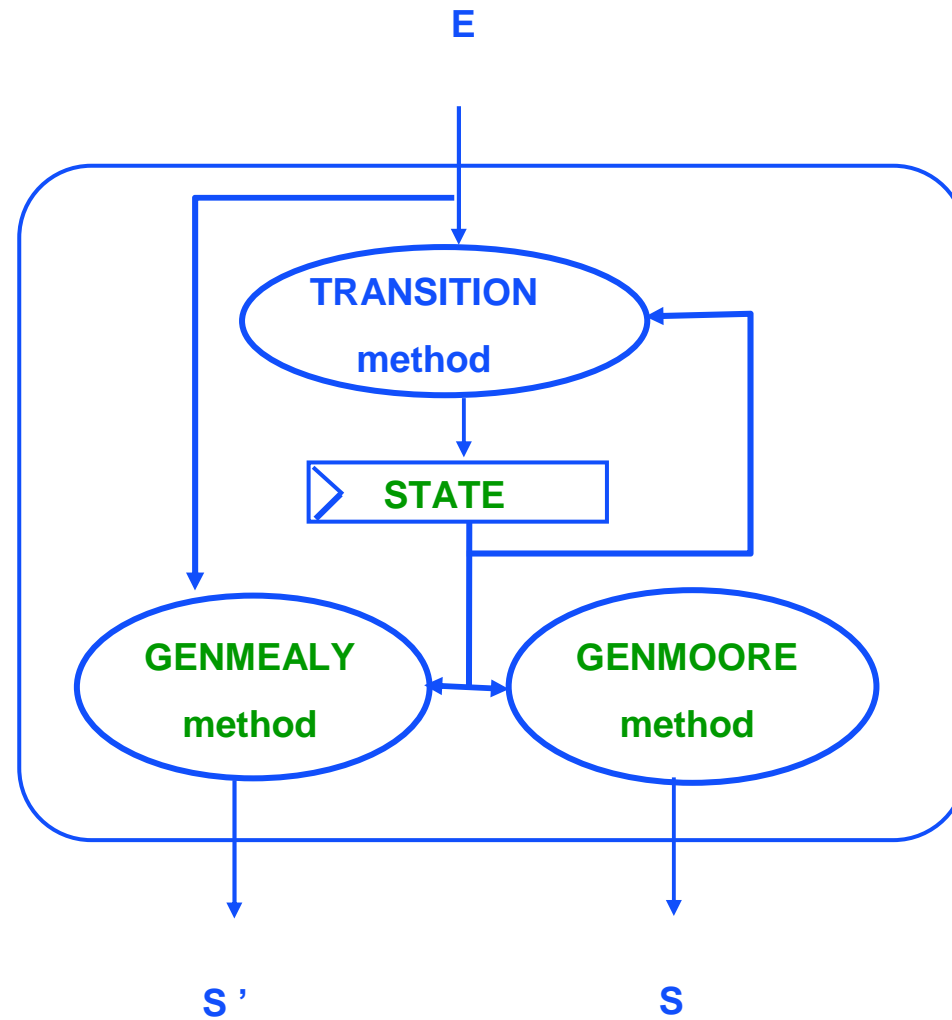
---

- © SoCLib goals
- © **SystemC modeling principles**
- © The Mutek Real-Time Operating System
- © The MWMR communication middleware
- © The DSX hardware/software codesign environment
- © An application example : Stereovision

## Communicating Synchronous FSM Model



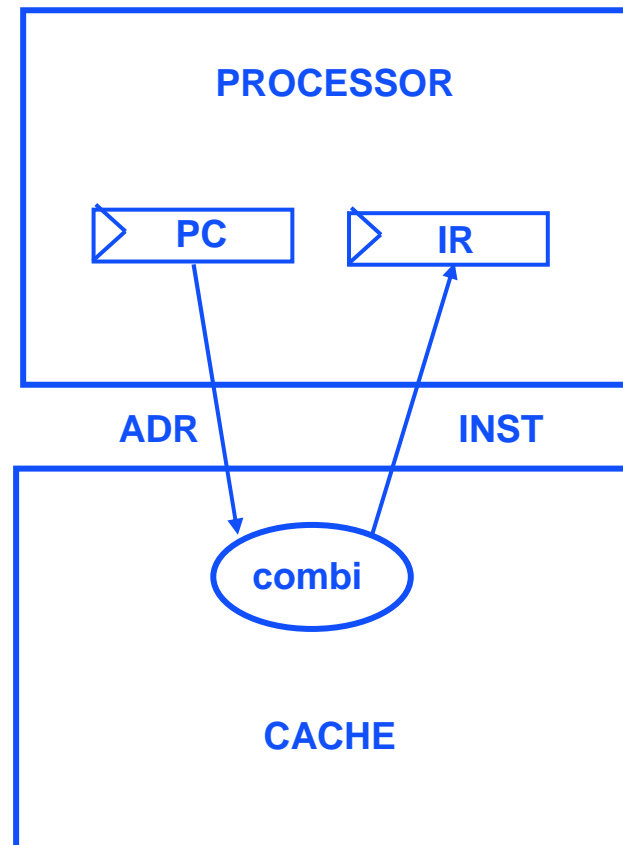
# Component Modeling



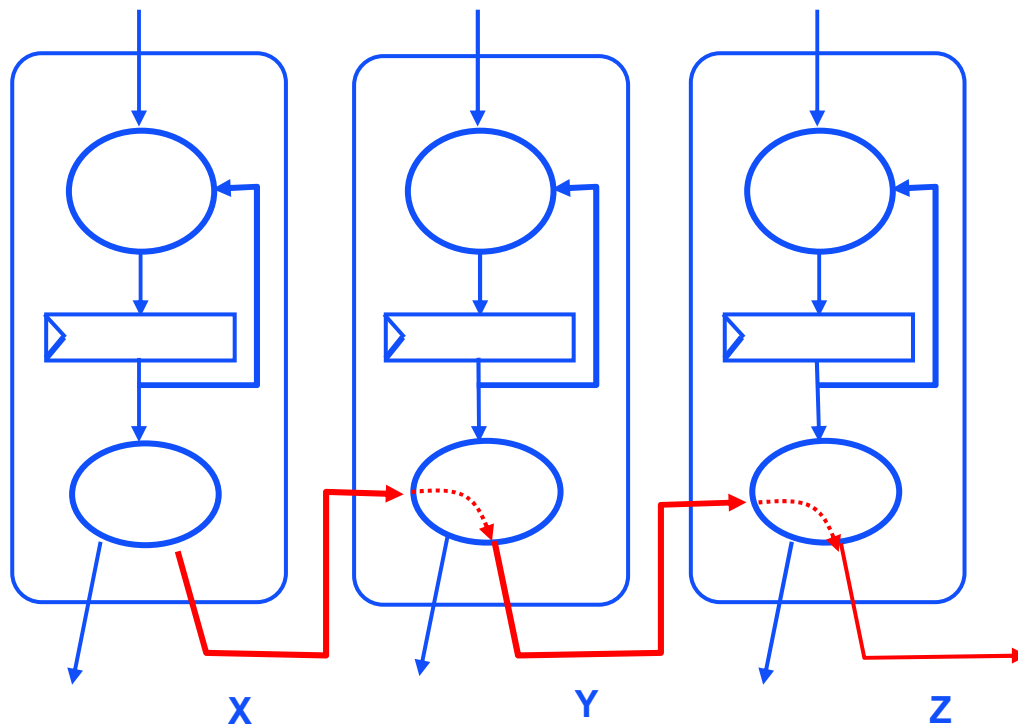
# The Mealy FSMs issue...

The general behaviour of the cache component is a Mealy FSM, because there is a combinational dependency between ADR signal and INST signal

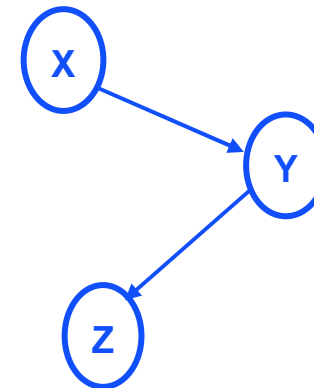
=> The PROCESSOR component must be evaluated before the CACHE component !



# Mealy Dependency Graph



- The nodes are the signals
- The arrows are Mealy dependencies between signals



## Static Scheduling

---

As there is no cycle in the Mealy Dependency Graph, it is possible to define a static scheduling that respects all dependencies:

```
For (cycle = 0 ; cycle < MAX ; cycle++)  
{  
  MODULE_0::Transition() ;  
  ...  
  MODULE_N::Transition() ;  
  MODULE_0::genMoore()  
  ...  
  MODULE_N::genMoore() ;  
  MODULE_0::genMealy() ;  
  ...  
  MODULE_N::genMealy() ;  
}
```

The genMealy methods ordering must respect the dependencies, and can be derived from the sensitivity sets of the methods.

# The Sensitivity Lists

---

```
SC_METHOD(transition);
```

```
sensitive_pos << CLK;
```

```
SC_METHOD(genMoore);
```

```
sensitive_neg << CLK;
```

```
SC_METHOD(genMealy);
```

```
sensitive_neg << CLK;
```

```
sensitive << input0;
```

```
sensitive << input1;
```

To precisely describe the input -> output dependancies, there is one genMealy method per output signal for each hardware component.

# SystemCASS

---

© All SocLib models can be efficiently used with the standard OSCI SystemC simulation engine that uses dynamic scheduling technics: All Transition » & « GenMoore » methods will be evaluated only one time per cycle...

© The SoCLib models can be simulated with the SystemCASS simulation engine that uses static scheduling, and provides a simulation speedup of one order of magnitude !

# Outline

---

- © SoCLib goals
- © The SoCLib virtual prototyping platform
- © SystemC modeling principles
- © **The Mutek Real-Time Operating System**
- © The MWMR communication middleware
- © The DSX hardware/software codesign environment
- © An application example : Stereovision

# Mutek : Embedded Operating System

---

MUTEK is an embedded OS for MP-SoC :

- © **POSIX compliant** : The multi-threaded software application can be executed on any workstation.
- © **Distributed scheduler** : one scheduler per processor, for true scalability.
- © **Real-Time support** : provides automatic re-initialisation in case of dead-line violation (soft real-time constraints).
- © **Controlled Placement** : allows the system designer to statically place the tasks on the processors, and the communication buffers on the memory banks.
- © **small foot-print** : about 25 kBytes.

# Mutek : status and evolution

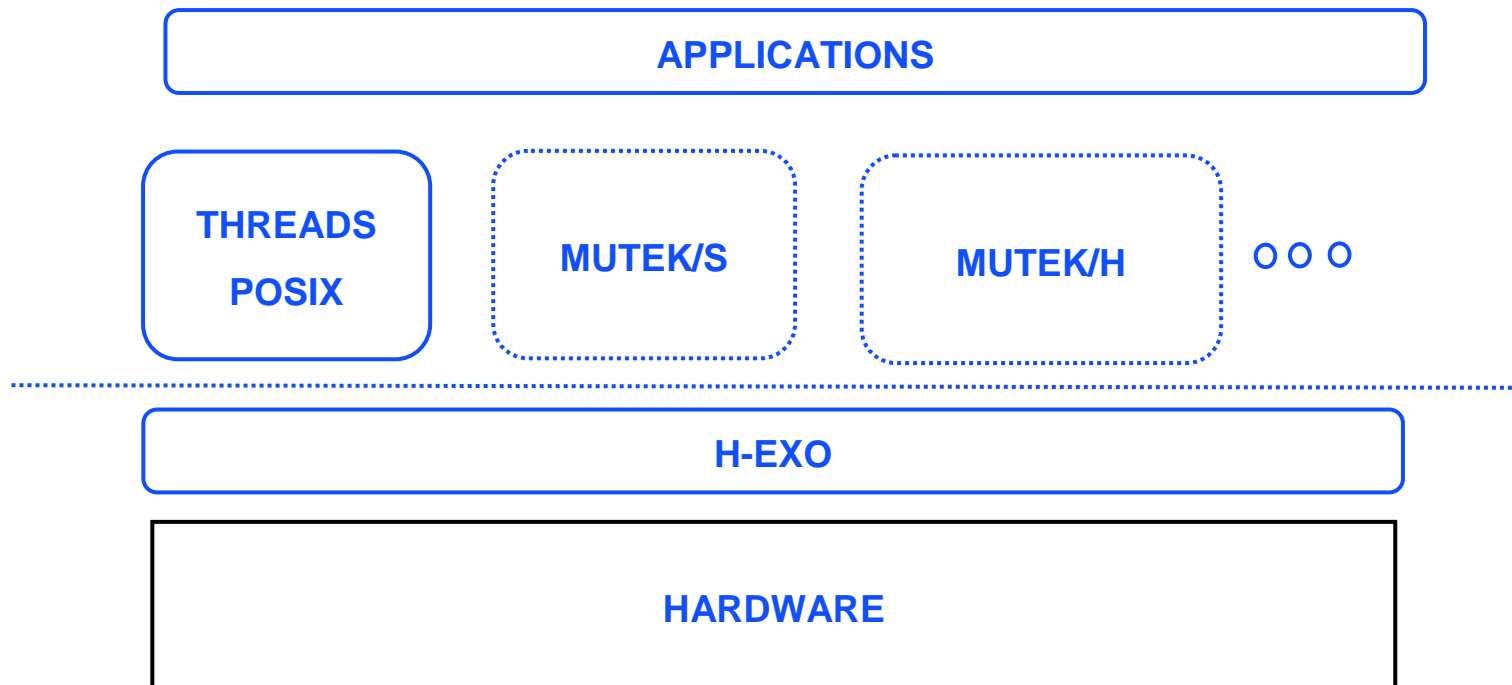
---

- © Mutek is jointly developed by several french laboratories (LIP6, TIMA, IRISA, CITI)
- © The present version supports only symmetric, homogeneous multi-processor architectures, and has been ported on several micro-processor (MIPS, SPARC, OPENRISC, ARM).
- © Two evolutions are in progress at LIP6
  - **MUTEK/S : Static version** (faster, but without POSIX compatibility)
  - **MUTEK/H : Heterogeneous version** (supporting mixed hardware architectures, with both General Purposes Processors & Digital Signal Processors)

# H-EXO layer

---

H-EXO is a very thin hardware abstraction layer (exo-kernel), defined to support heterogeneous multi-processor architectures



# Outline

---

- © SoCLib goals
- © The SoCLib virtual prototyping platform
- © SystemC modeling principles
- © The Mutek Real-Time Operating System
- © **The MWMR communication middleware**
- © The DSX hardware/software codesign environment
- © An application example : Stereovision

# Software application modeling

---

© A parallel application is described with two types of components:

- A set of **concurrent tasks**
- A set of **MWMM communication channels**

© Each communication channel is implemented as a **software FIFO**. Each FIFO can have several writers and several readers.

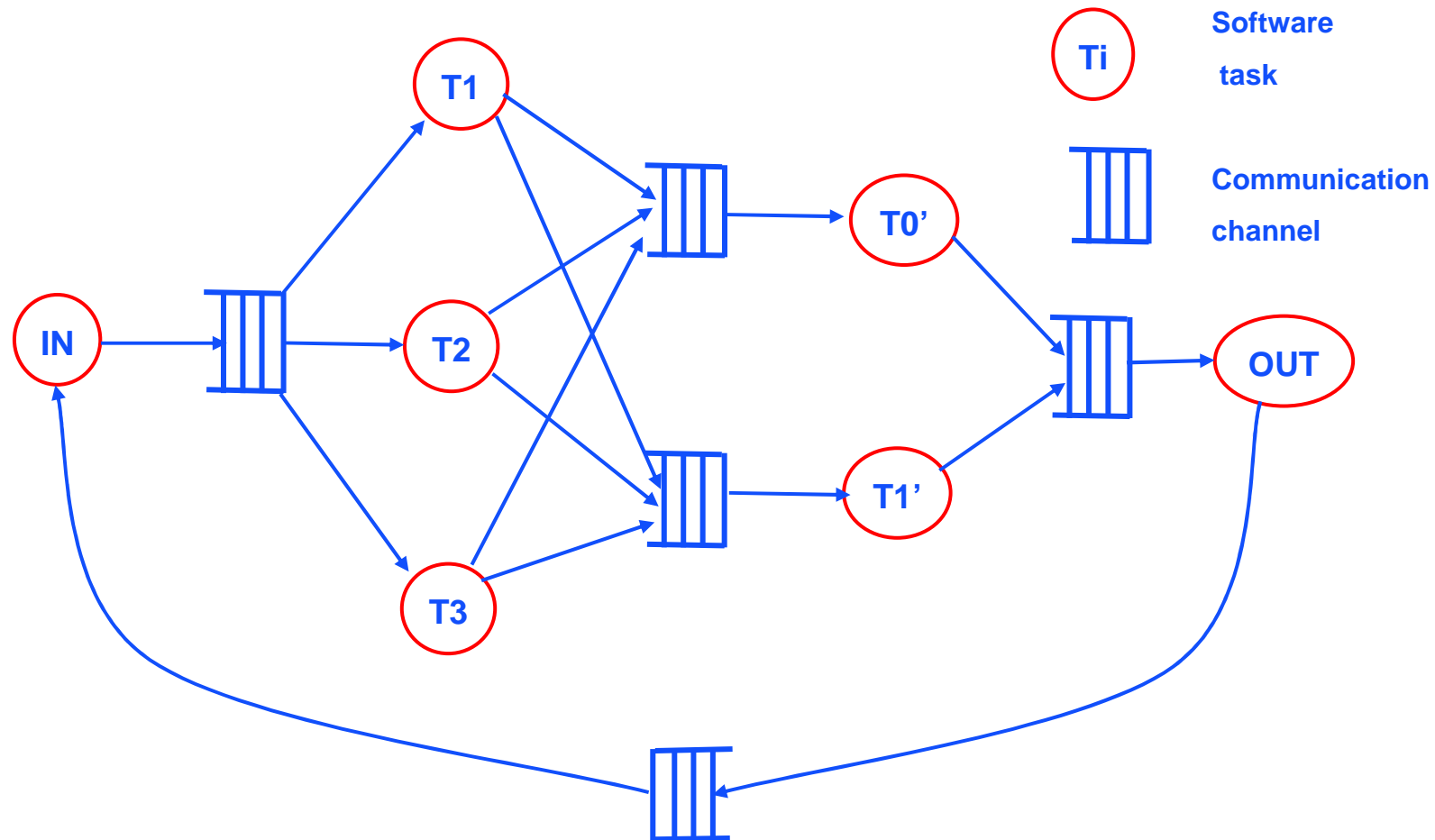
© Software tasks use only two (blocking) **communication functions** :

- `MWMM_read(channel, Buf_address, item_number)`
- `MWMM_write(channel, Buf_address, item_number)`

© Each software task can be implemented as

- A **software thread** (without modification)
- A **synthesized hardware coprocessor**.

# Task & Communication Graph

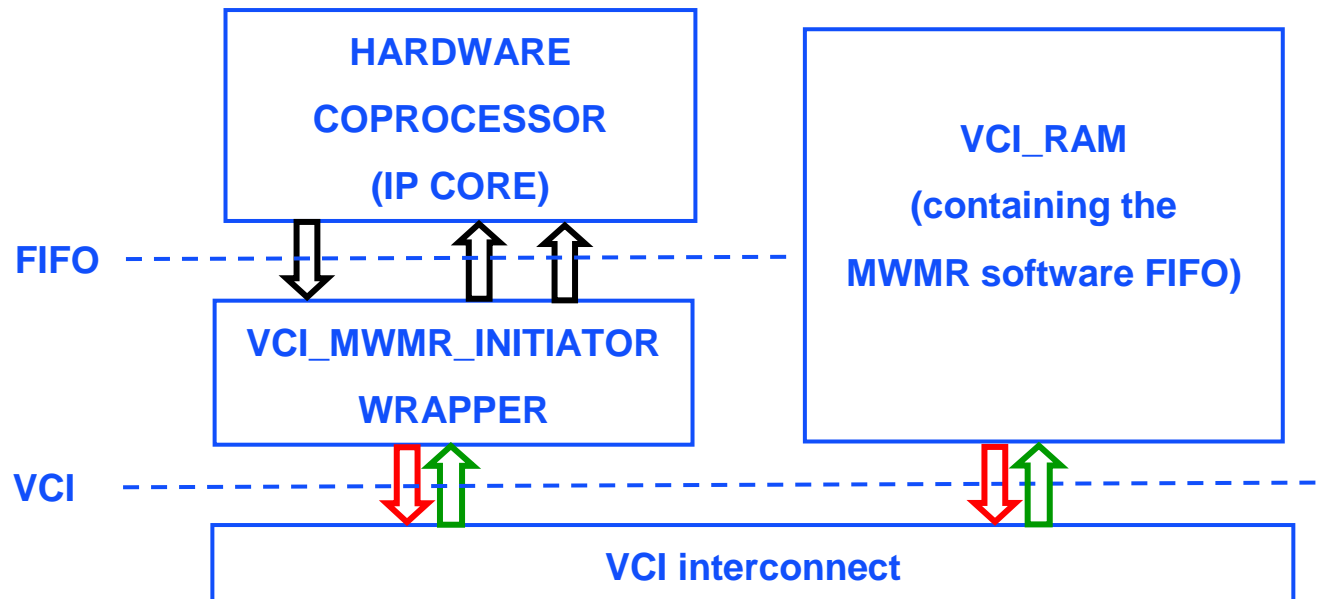


# MWMMR communication channel

---

- © MWMMR communication channels are implemented as **software FIFOs**, in a shared memory segment. Each MWMMR FIFO is protected by a system lock.
- © A FIFO can have several « producer » tasks and several « consumer » tasks, and each of those tasks can be implemented as **hardware, or software** : Both hardware and software implementations use the same protocol to access the software FIFO.
- © In the MWMMR access protocol, each MWMMR access generates **six memory transactions**.
- © The **software implementation** of the MWMMR\_write() & MWMMR\_read() primitives is based on the POSIX threads => It can be directly simulated on any POSIX compliant workstation.
- © The **hardware implementation** of the MWMMR\_write() & MWMMR\_read() primitives is based on a generic - VCI compliant - MWMMR initiator. This hardware « wrapper » provides the synthesized hardware coprocessor a simple FIFO interface.

# MWMMR « wrapper »



The VCI\_MWMMR\_initiator wrapper is a generic hardware component, that implements a variable number of read or write MWMMR FIFO channels.

# Outline

---

- © SoCLib goals
- © The SoCLib virtual prototyping platform
- © SystemC modeling principles
- © The Mutek Real-Time Operating System
- © The MWMR communication middleware
- © **The DSX hardware/software codesign environment**
- © An application example : Stereovision

# DSX : Design Space eXploration

---

The DSX environment provides 3 services :

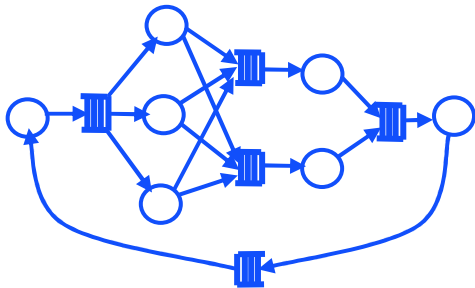
- ◆ **define the hardware architecture parameters**  
(using several generic, parameterized platforms)
- ◆ **define the software application architecture**  
(Tasks & Communication Graph)
- ◆ **map the software application on the architecture**  
(memory space segmentation & placement constraints)

This tool is based on the Python language. It generates :

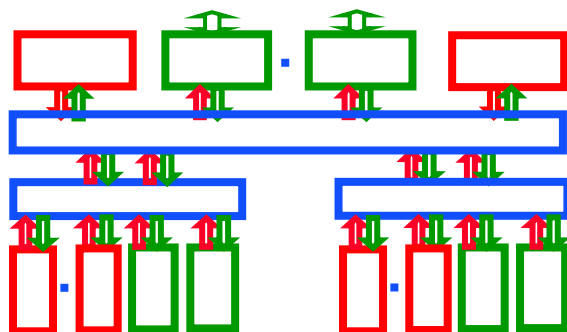
- ◆ A SystemC description of the hardware architecture
- ◆ All script files for the binary code generation (GCC)

# DSX : Mapping

Multi-thread application



Multi-processor architecture



The system designer can :

- **choose** the hard/soft implementation for each task
- **map** the software tasks on the programmable processors and the hardware tasks on synthesized (or existing) coprocessors
- **map** the inter-tasks communication buffers on the physical memory banks.
- **map** for each task, the private code & data segments on the physical memory banks.

# Outline

---

- © SoCLib goals
- © The SoCLib virtual prototyping platform
- © SystemC modeling principles
- © The Mutek Real-Time Operating System
- © The MWMR communication middleware
- © The DSX hardware/software codesign environment
- © **An application example : Stereovision**

## An example : stéréo-vision

---

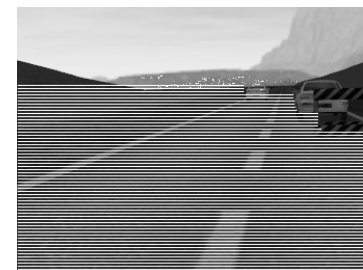
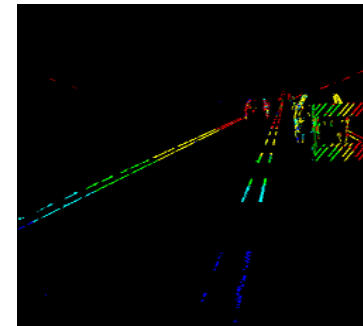
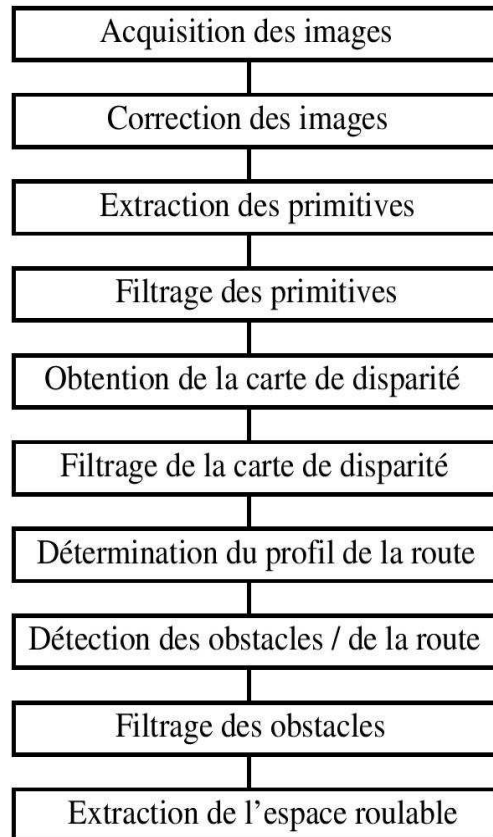
The goal is obstacle detection for cars, by stereo-vision technics, using two synchronized cameras.

- **Image size : 256 \* 512 pixels (8 bits pixel)**
- **Fréquence : 40 couple of images per second**

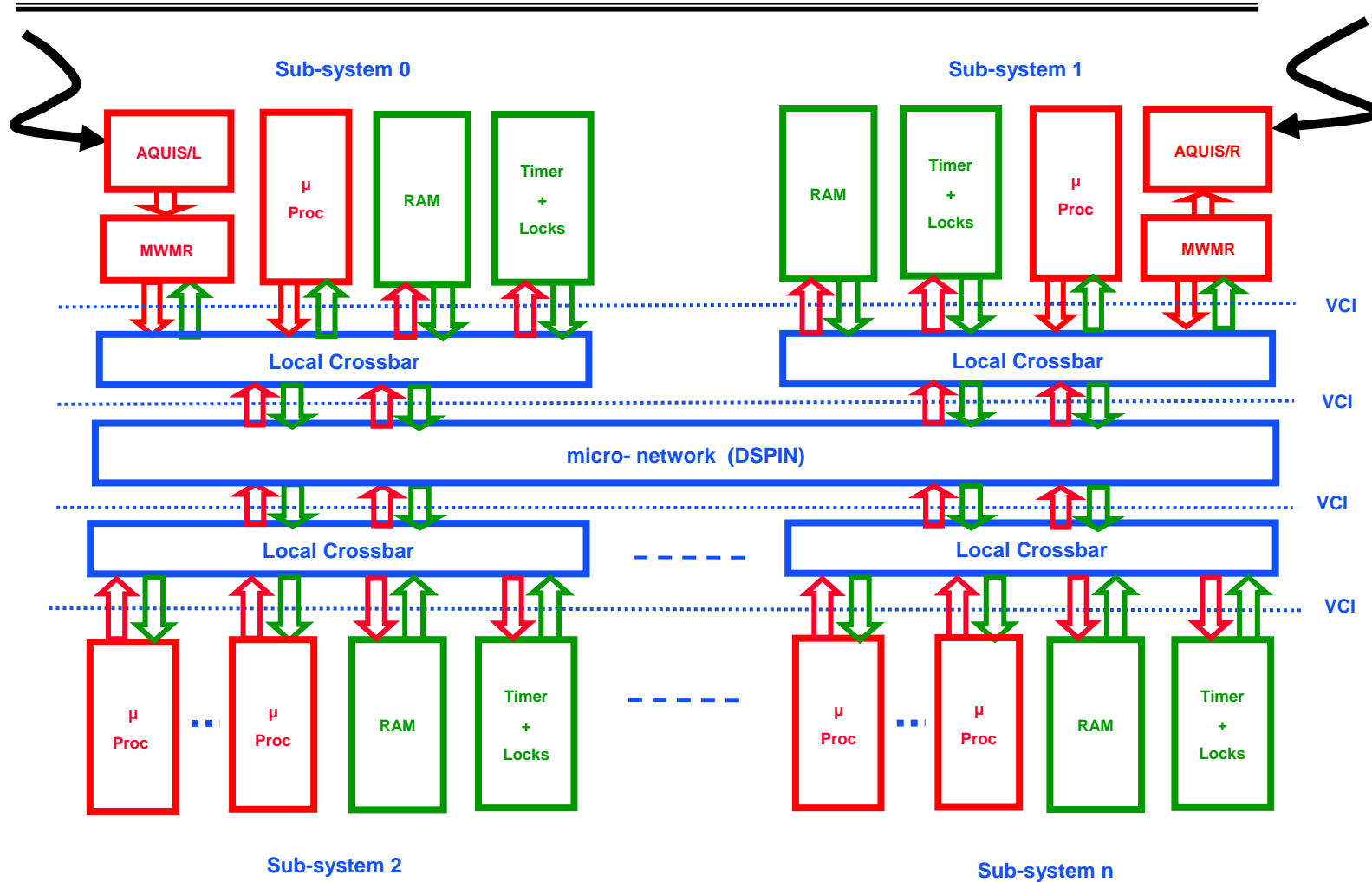
The stereo-vision algorithm and the sequential software application have been developed by the LIVIC laboratory.

We want to parallelize the algorithm and to map the corresponding multi-threaded software application on a MP-SoC architecture.

# Stereo-vision : Genera Principle



# Generic Hardware Architecture



# Stereo-Vision : Résultats

---

## Hardware

- 8 clusters : (6 computation clusters, 2 acquisition clusters)
- 30 general purpose processors (MIPS R3000)
- 16 embedded memory banks (total memory capacity : 700 Koctets)
- silicon area (estimated) : 60 mm<sup>2</sup>

## Software :

- 30 threads exécuting under the MUTEK OS
- Optimized task & communication channels placement (DSX)
- Soft real time violation management.

## Virtual prototyping (Cycle Accurate) :

- Simulation speed : 15000 cycles/s (on a 3GHz PC workstation)
- results : 6.8 million cycles per couple of images.

**=> 22.6 ms at 300 MHz**

# Conclusion

---

**La plate-forme SoCLib vise le prototypage virtuel d'applications logicielles embarquées complexes sur systèmes multi-processeurs intégrés sur puce.**

**Cette plate-forme, financée par l'ANR, sera accessible aux industriels et aux universitaires sous forme de logiciel libre : modèles de simulation des composants matériels, et outils logiciels de base.**

**La première version accessible au public est prévue fin 2007.**

**Cette plate-forme a clairement vocation à s'intégrer dans un projet européen, qui reste à définir...**