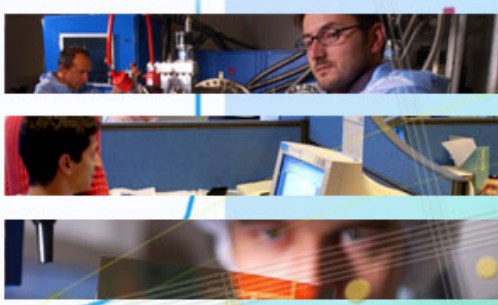
micro and nanoelectronics
microsystem
ambiant intelligence
image chain
biology and health

2008

Hardware dependant Software design

Ahmed A. Jerraya
CEA-LETI
Ahmed.jerraya@cea.fr



leti MINATEC

cea

Outline

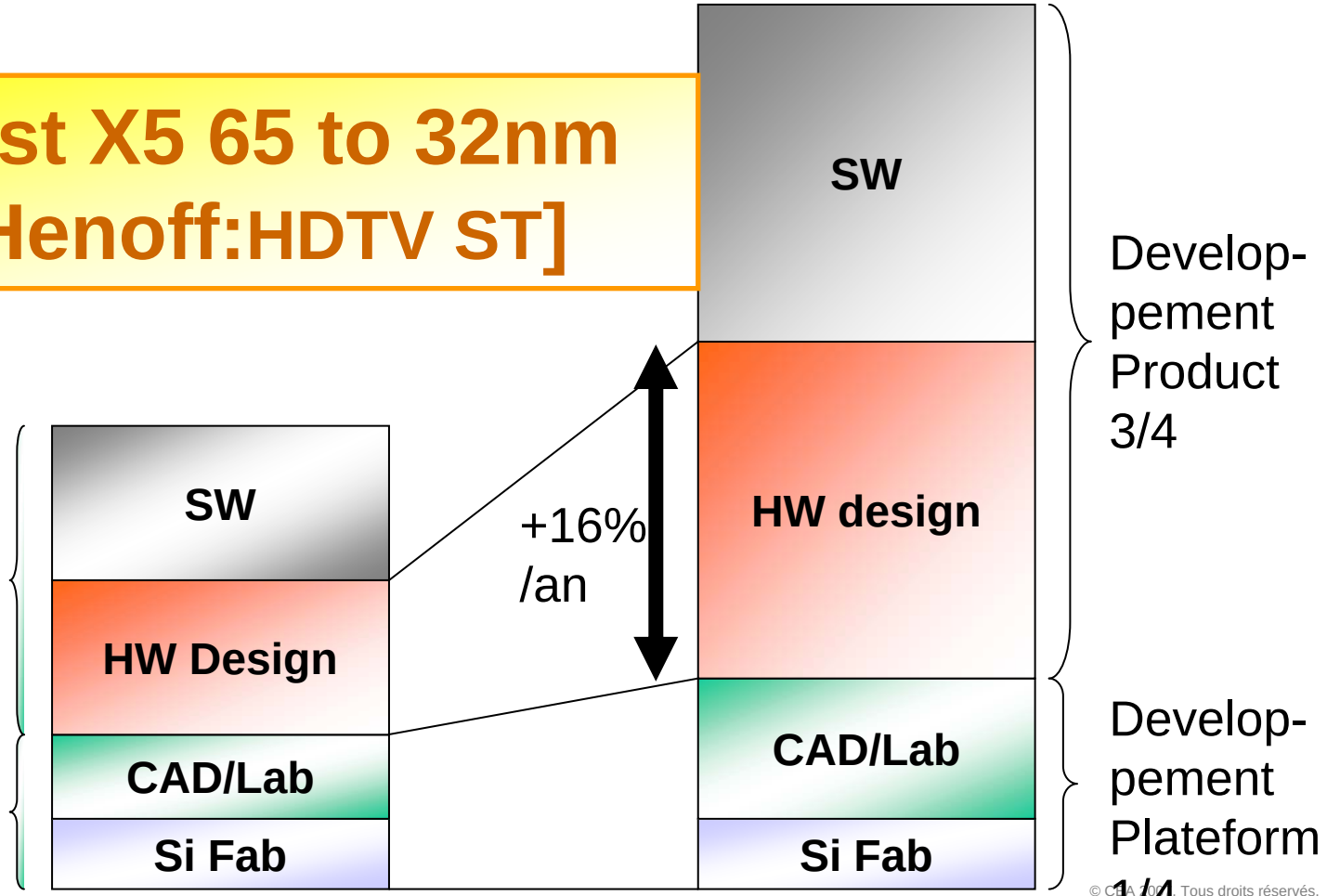
- Multiprocessor System on Chip: HW-SW Architectures
- HW/SW interfaces abstraction: Programming models
- Hardware dependent software design
- Long term trends: HW-SW codesign.

SoC Design Cost

Traditional Model

Trend

Cost X5 65 to 32nm [Henoff:HDTV ST]



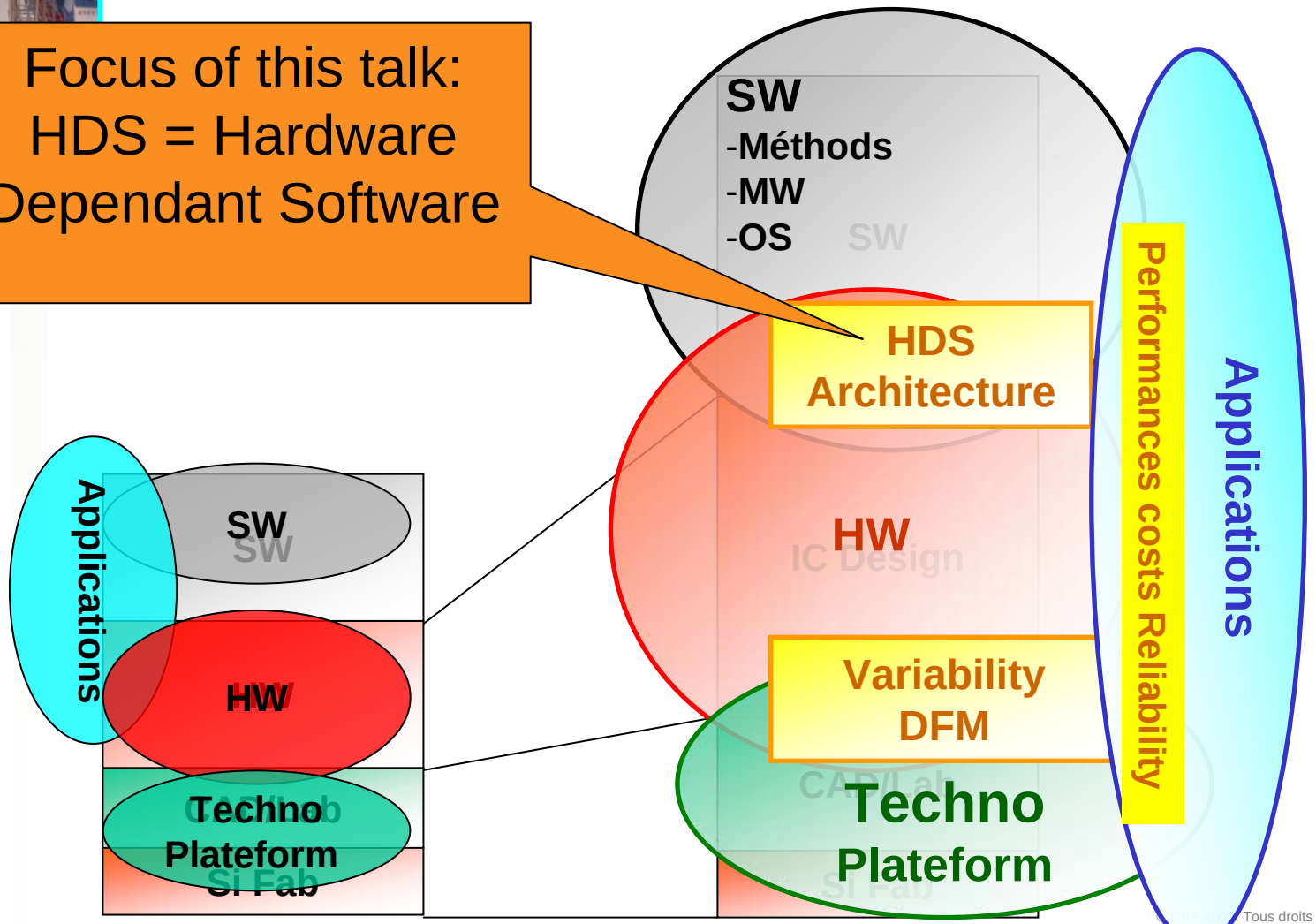
Développment Product 2/3

Developpment Plateform Techno. 1/3

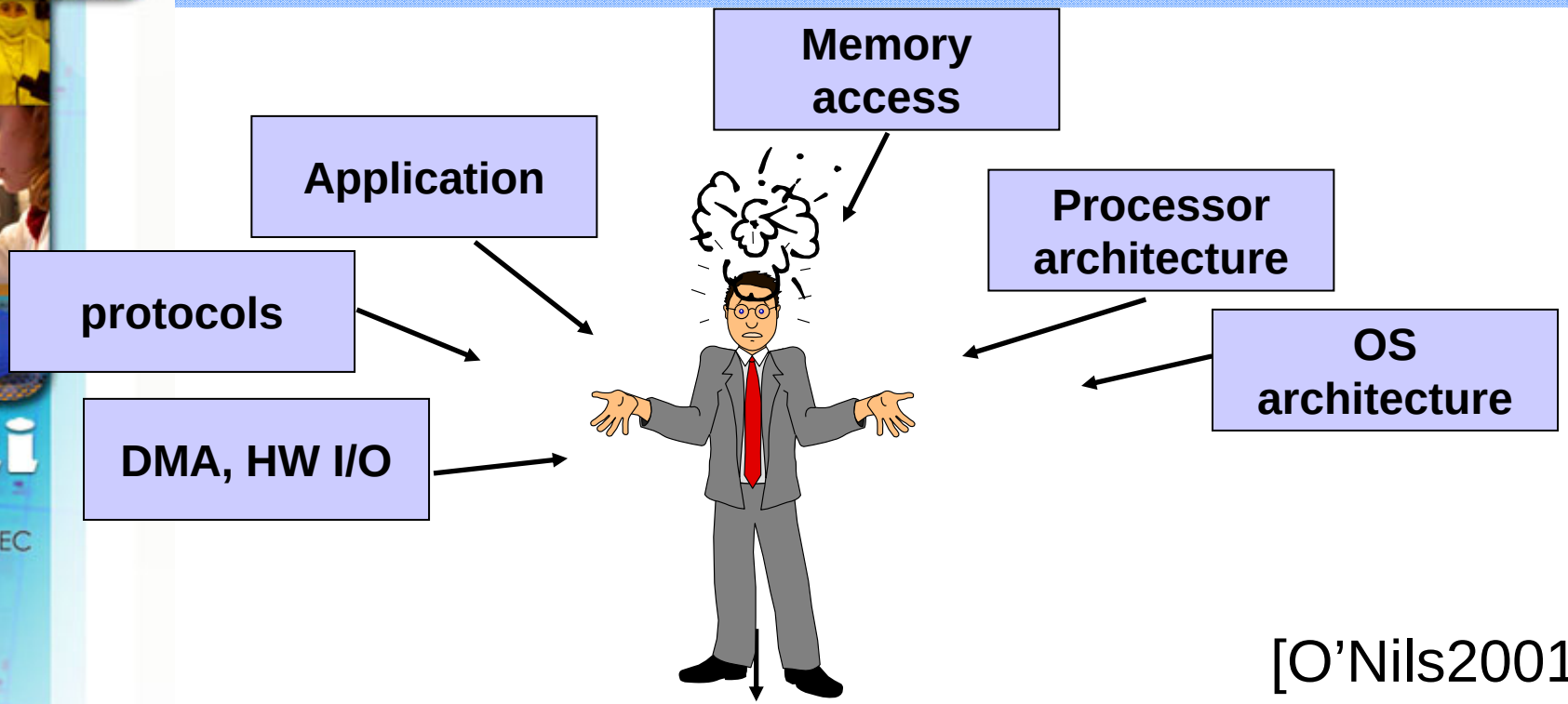
[MEDEA+Forum]

Global optimization required

Focus of this talk:
HDS = Hardware Dependant Software



Key Technology: HW/SW Interfaces



[O'Nils2001]

HW/SW interfaces design Challenges

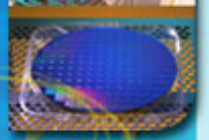
- **HW/SW interfaces design requires pluridisciplinary knowledge**
- **HW/SW interfaces are crucial for both cost and performances**
  - **Cost issues:**
    - **Hardware dependent Software is tedious to develop, difficult to debug and validate**
    - **Non optimized software leads to larger footprint and consequently to larger embedded memory**
  - **Performances :**
    - **Non optimized HW/SW interface introduces non acceptable global latencies**
    - **Processor + cache + embedded memory are responsible of the major power consumption part in a SoC**

A major challenge in MPSoC design is to efficiently design the HW/SW interfaces

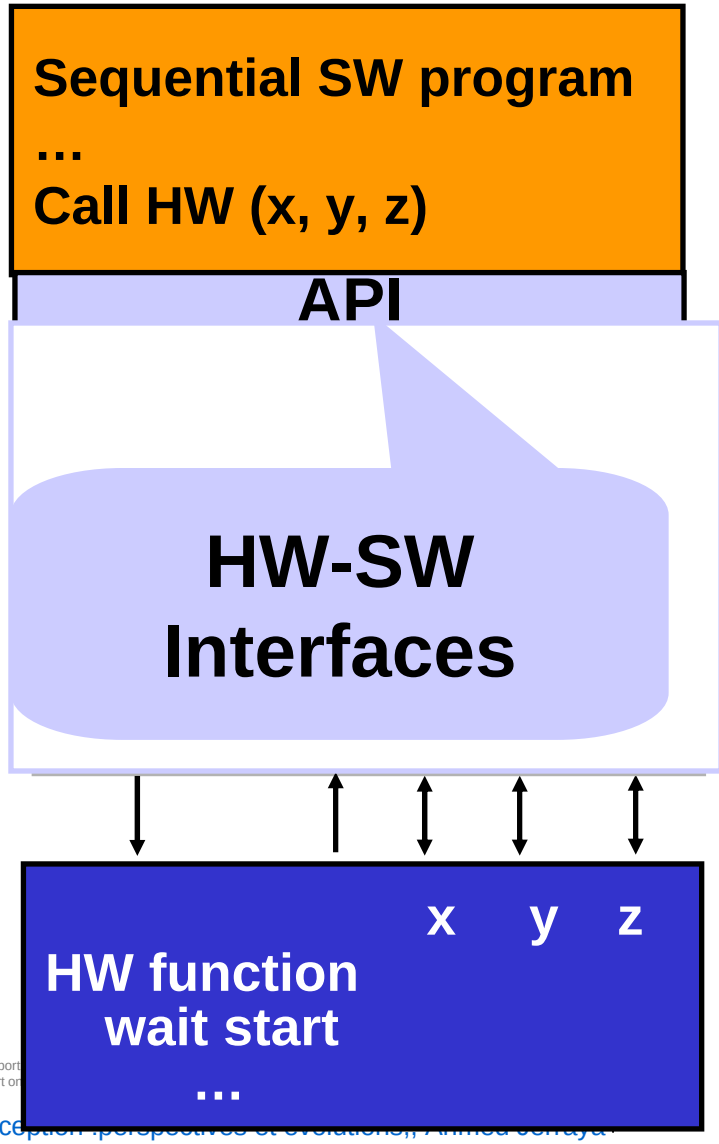
Outline

- Multiprocessor System on Chip: HW-SW Architectures
- HW/SW interfaces abstraction: Programming models
- Hardware dependent software design
- Long term trends: HW-SW codesign.

Defining HW-SW Interfaces

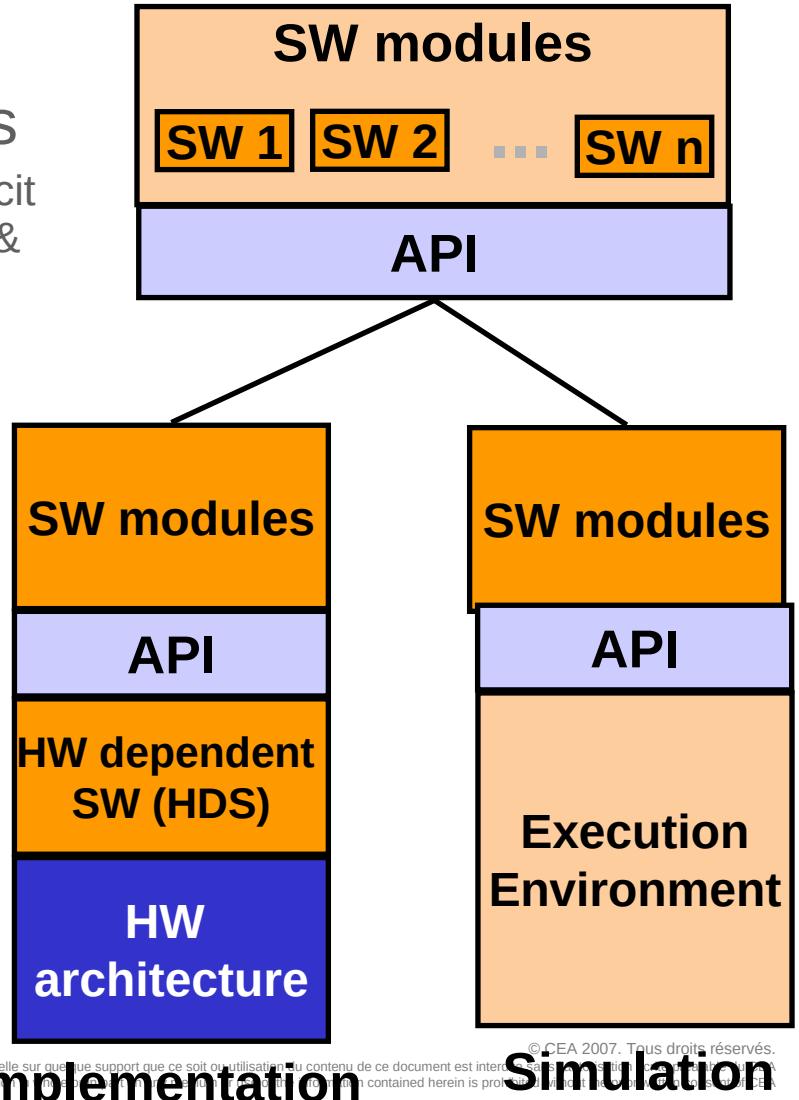


- **Application SW Designer: A set of system calls used to hide the underlying execution platform. Also Called Programming Model**
- **HW designer: A set of registers, control signals and more sophisticated adaptors to link CPU to HW subsystems.**
- **System SW designer: Low level SW implementation of the programming Model for a given HW architecture.**
- **CPU is the ultimate HW-SW Interface**
- **Sequential scheme assuming HW is ready to start low level SW design**
- **SoC Designer**
  - **Abstracts both HW and SW in addition to CPU**
  - **HW-SW interfaces tradeoff**



Programming Model

- The Classical Solution to Abstract HW-SW Interfaces
  - Programming language with implicit primitives (e.g. module hierarchy & threads in SystemC)
  - API: Application Programming Interface (MPI, Posix Threads)
  - Simulation model (MPICH, Linux)
- Used by SW community to free the SW designer from knowing HW details.
- Facilitate porting of application SW over different architectures that support the same API.



Programming models, the GAP

SoC Design → ← **Distributed SW Design**

Programming Model

RTL (Verilog, BinSW)	Virtual Prototype (RTL HW, BinSW, e.g. SystemC)	**Mixed Abstract HW-SW Interfaces models**	MPI	RT-CORBA	CORBA SDL

Explicit concepts

ALL	ALL		High Level Synchro-nisation	Communi-cation	- Concurrency - Threading
CPU imple-mentation	RTL HW CPU orga-nization		NONE	NONE	NONE

Programming Models for MPSoC

SoC Design → Distributed SW Design

Programming Model

RTL (Verilog, BinSW)	Virtual Prototype (RTL HW, BinSW, e.g. SystemC)	Transaction Accurate (TLM HW, TLM SW , e.g. SystemC++)	Virtual Architecture (Abstract HW, Threads, e.g. SystemC, Simulink)	MPI	RT-CORBA	CORBA SDL

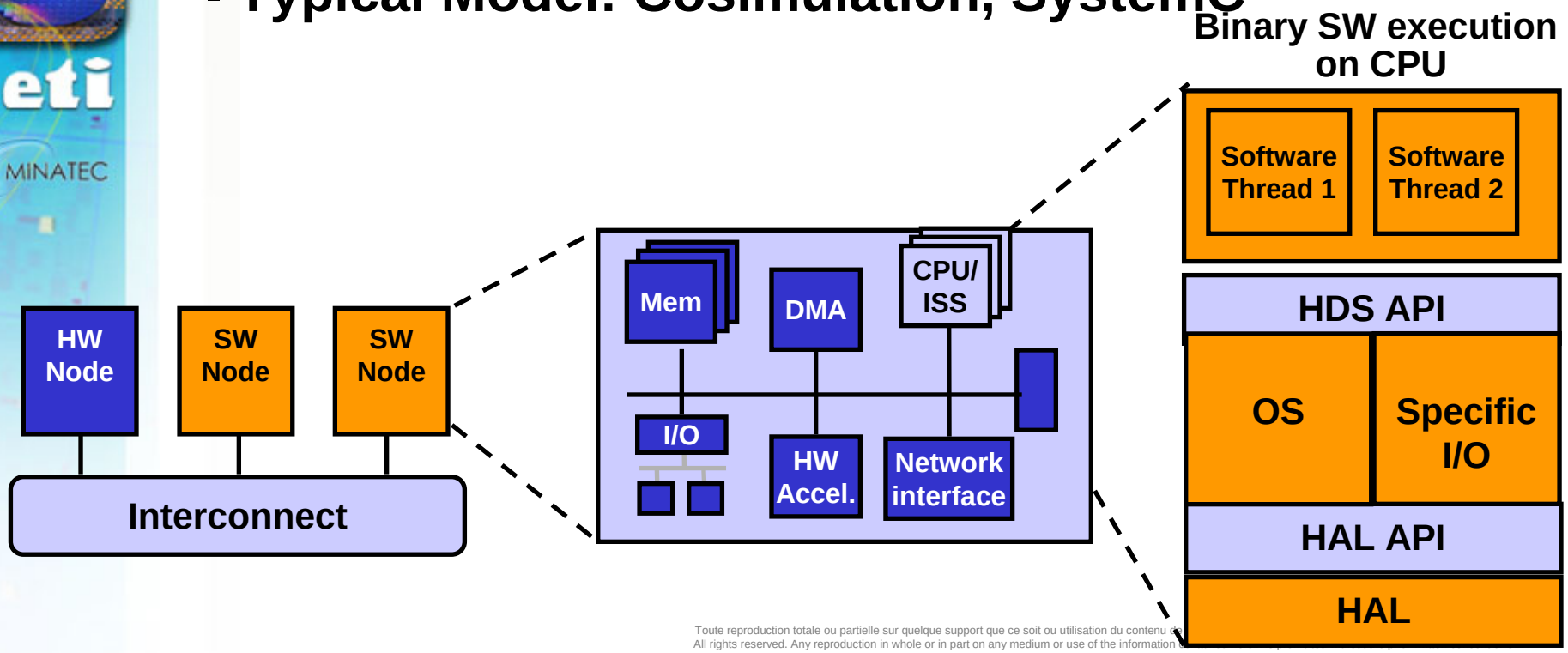
Explicit concepts

ALL	ALL	- OS - Specific I/O	Communication/ Computation Modules	Synchro-nisation	Communi-cation	- Concurrency - Threading
CPU imple-mentation	RTL HW CPU orga-nization	CPU Subsystem - Explicit HW modules	Abstract Interconnect	NONE	NONE	NONE

- New HW-SW Abstraction levels
- Hiding CPU in addition to HW & SW

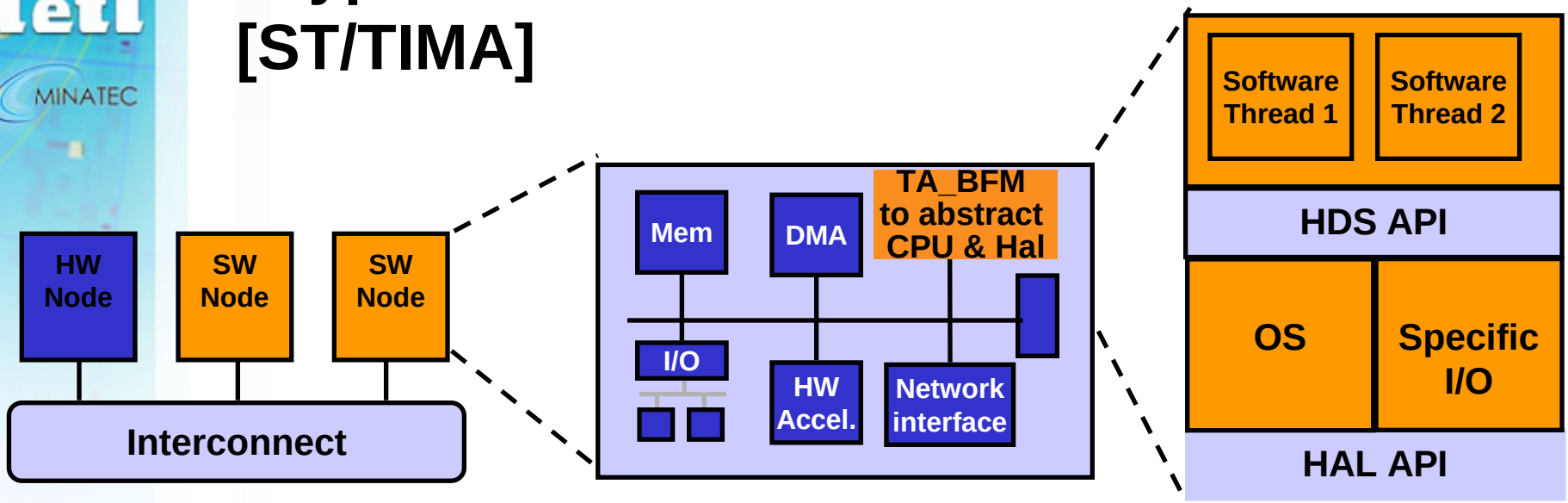
Virtual Prototype Model

- Explicit SW (Binary)
- Explicit HW
- Implicit CPU Implementation (ISS)
- Cycle accurate Model
- Typical Model: Cosimulation, SystemC



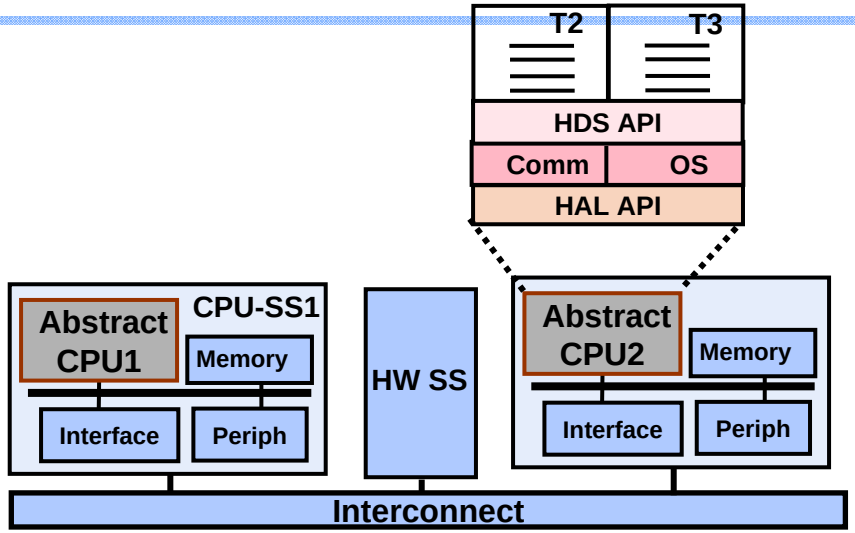
Transaction Accurate model

- Explicit OS
- Explicit CPU Subsystem Architecture
- Implicit CPU/HAL (TA_BFM)
- Transaction cycle accurate HW
- Typical Model : Native SW Execution [ST/TIMA]



SW code at Transaction Accurate level

- HdS responsible for task and HW resources management
- Task code integration with OS and HdS API implementation
- Communication (Explicit communication protocol)



Example of TA SW code

main.c

```
extern void task_t2 (void);
void __start (void){
…
create_task(task_t2);
… }
```

Communication SW

```
void recv_data (ch,dst,size) {
switch (ch.protocol){
case FIFO:
   If  (ch.state==EMPTY) __schedule();
…
case REG:
     dst = _io_read_reg (ch.addr,size);
```

task_T2.c

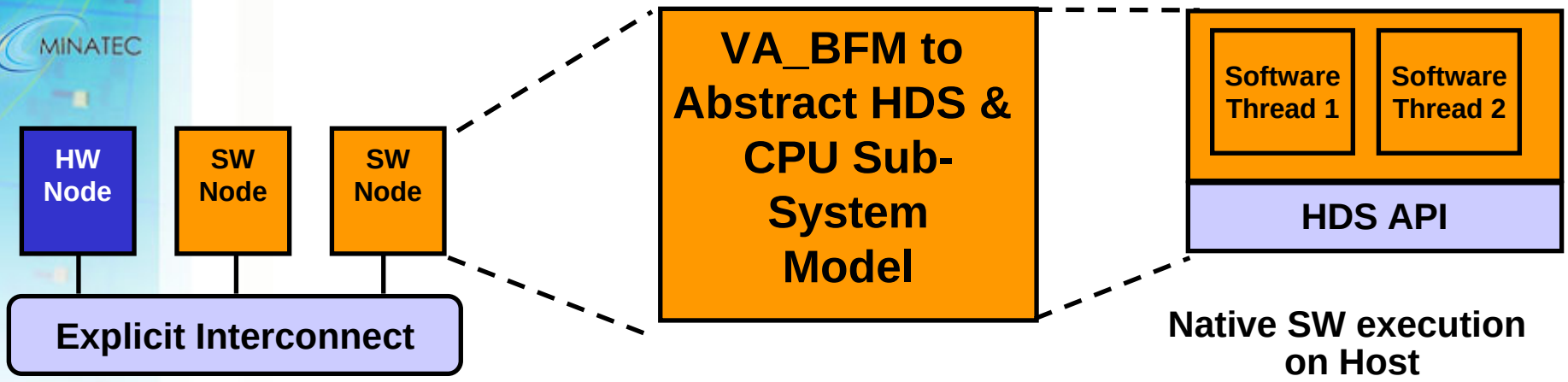
```
channel ch_fifo,ch_reg;
void task_t2(void ) {…
 recv_data (ch_fifo,B, 10);
 recv_data (ch_reg,C,20);
…}
```

OS

```
void __schedule(void) {
  int old_tid=cur_tid;
  cur_tid=new_tid;
  __cxt_switch(old_tid.cxt,cur_tid.cxt);
…}
```

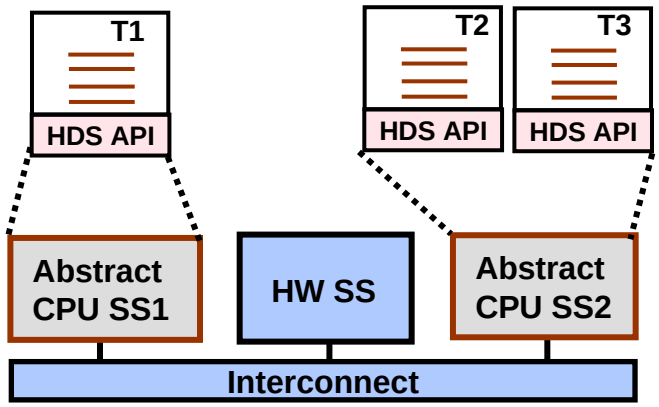
Virtual Architecture model

- Explicit SW communication API
- Explicit System Interconnect
- Implicit HDS (OS, Communication Implementation)
- Typical Models: TTL, DSoC, Pthreads



SW code at Virtual Architecture level

- Memory declaration
- Computation code
- Communication code using HdS API
  - Specific mapping to platform channels



Task code of T1

```
static int B[10], C[20], D[10];                    Memory
void task_T1( )
{
   while(1)
        {
         recv_data (reg_ch, B, 10);                Communication API
         F1(B,C);                                  Computation code
         F2(C,D);
         send_data (gmem_ch, D, 10);               Communication API
        }
}
```

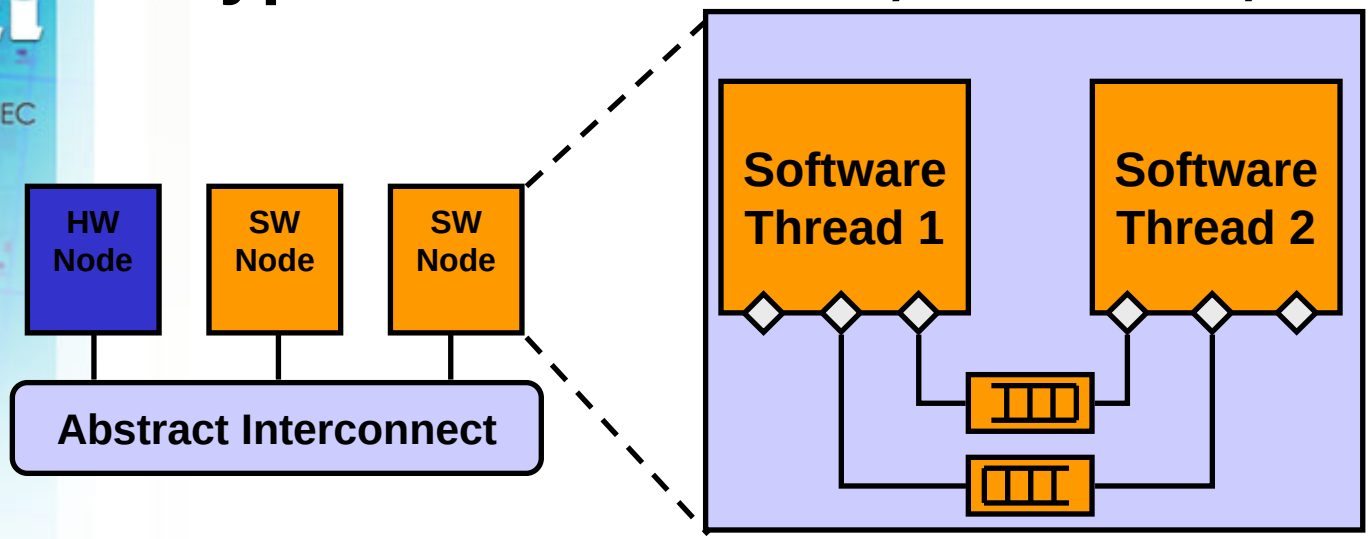
Communication primitive

```
void recv_data (REG_Ch*  ch, void* dst, int size) {
 dst = ch->read (ch->address,size);
}
```

```
class REG_Ch : public sc_prim_channel {
word *buffer;
public: word * read (unsigned int addr,int size) {
              for (i=0; i<size; i++)
                   *(ret+i)=*(buffer + addr +i);
              return ret;
              }
…};
```

System Architecture model

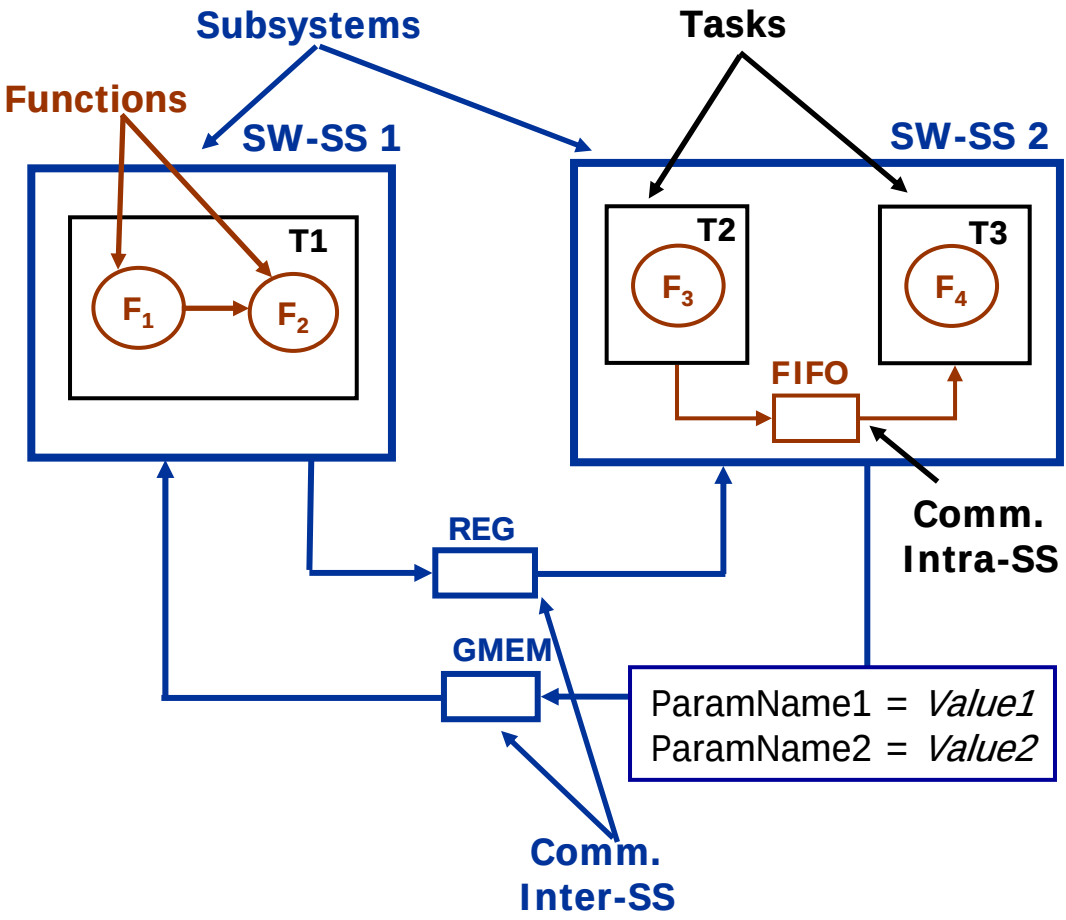
- Explicit thread/subsystems & HW-SW partition
- Implicit communication
- Typical Models: MPI, Simulink, KPN



Implicit SW execution as a simulation module

System Architecture Modeling in Simulink

- Capture application and mapping to architecture
- Task (set of functions: Simulink blocks, S-functions)
- Subsystem (set of tasks)
- Abstract communication types (Intra-SS , Inter-SS)
- Communication protocol (Generic Simulink I/Os, Explicit annotation for implementation)

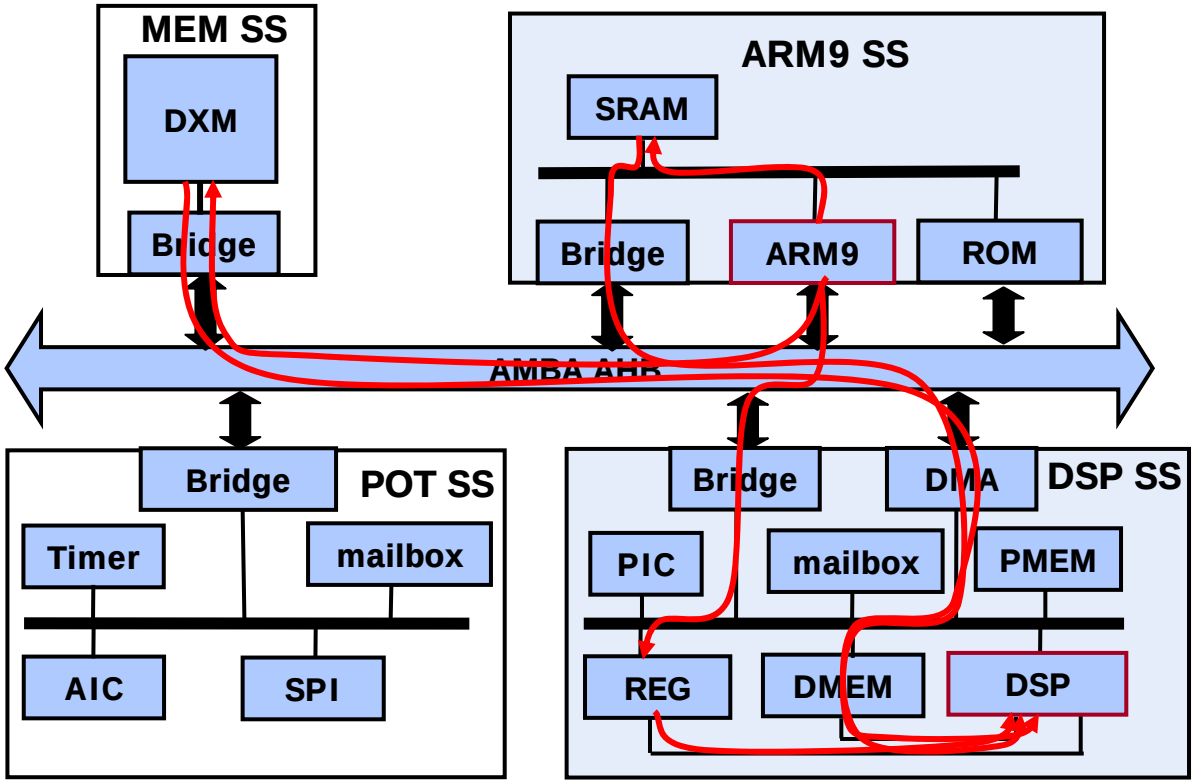


Outline

- Multiprocessor System on Chip: HW-SW Architectures
- HW/SW interfaces abstraction: Programming models
- Hardware dependent software design
- Long term trends: HW-SW codesign.

Heterogeneous MPSoC

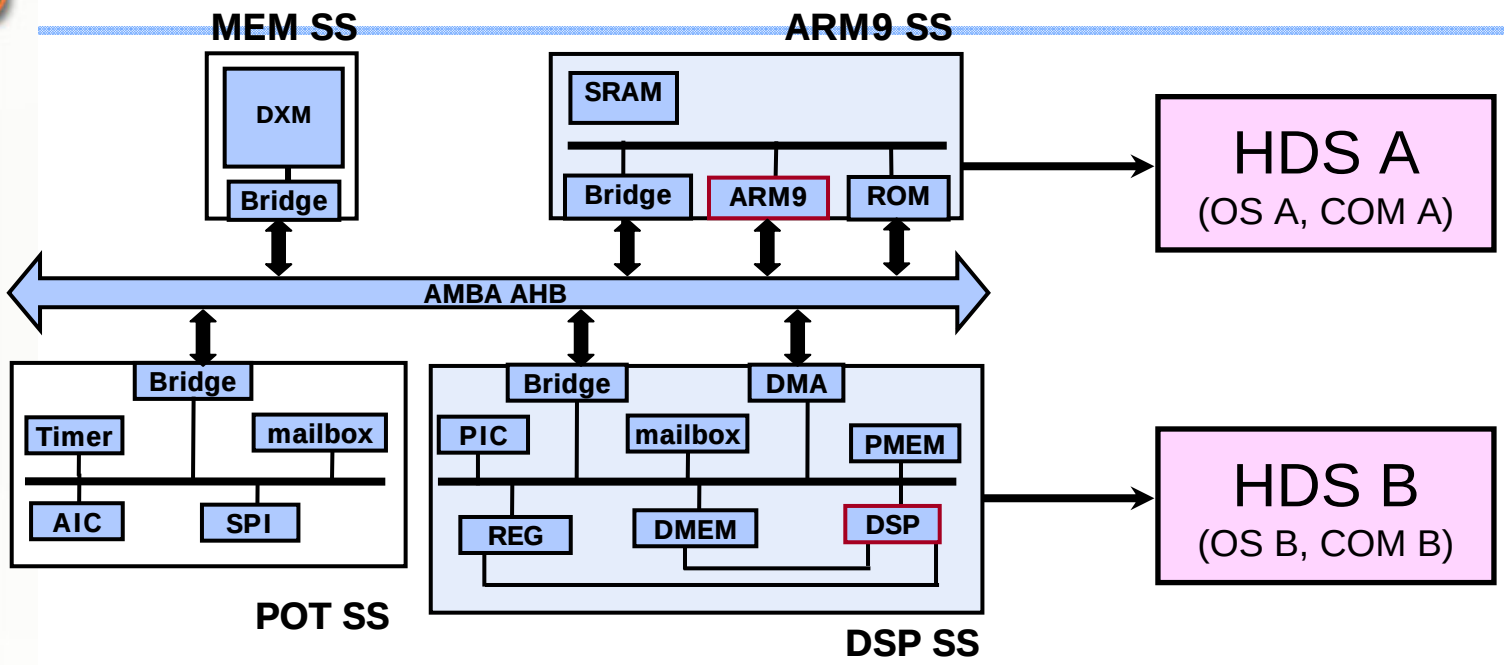
- **Diopsis Tile [ESWEEK06]**
- **ARM9 SS**
- **DSP SS**
- **MEM SS**
- **POT SS (Periph. On Tile):**
  - I/O peripherals
  - System peripherals
- **Interconnect: AMBA bus**



- ***Local & global memories accessible by both processing units***

➢ **Different communication schemes between CPUs**

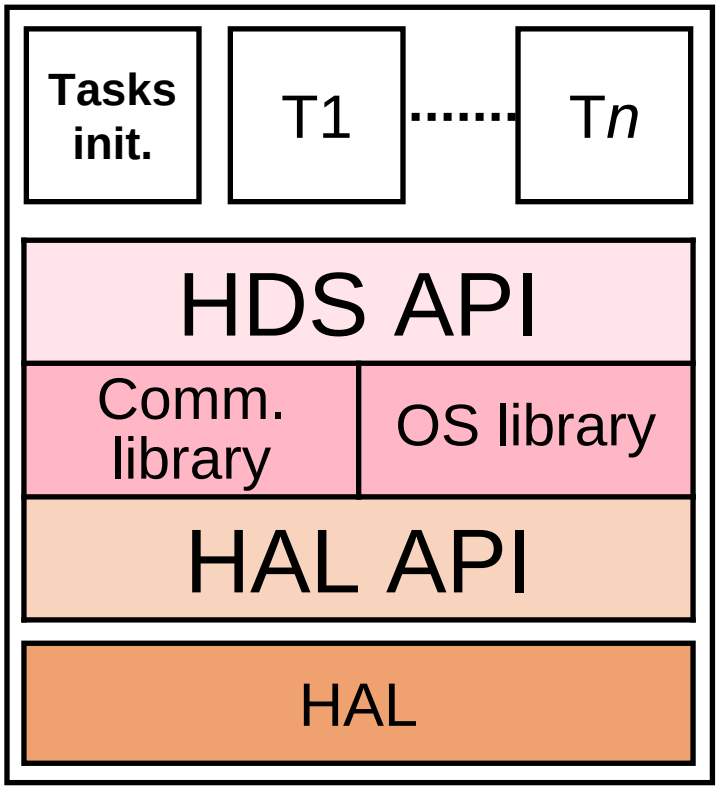
Multiple SW Stacks



- **Reduce the design cost**
  - Small memory footprint
- **Increase performances**
  - Specific platform devices
  - Efficient communication schemes
- **ARM specific OS and COM**
  - DXM & DSP REG access
- **DSP specific OS and COM**
  - PIC, Mailbox, DMA
  - REG & DXM access

SW Stack organization

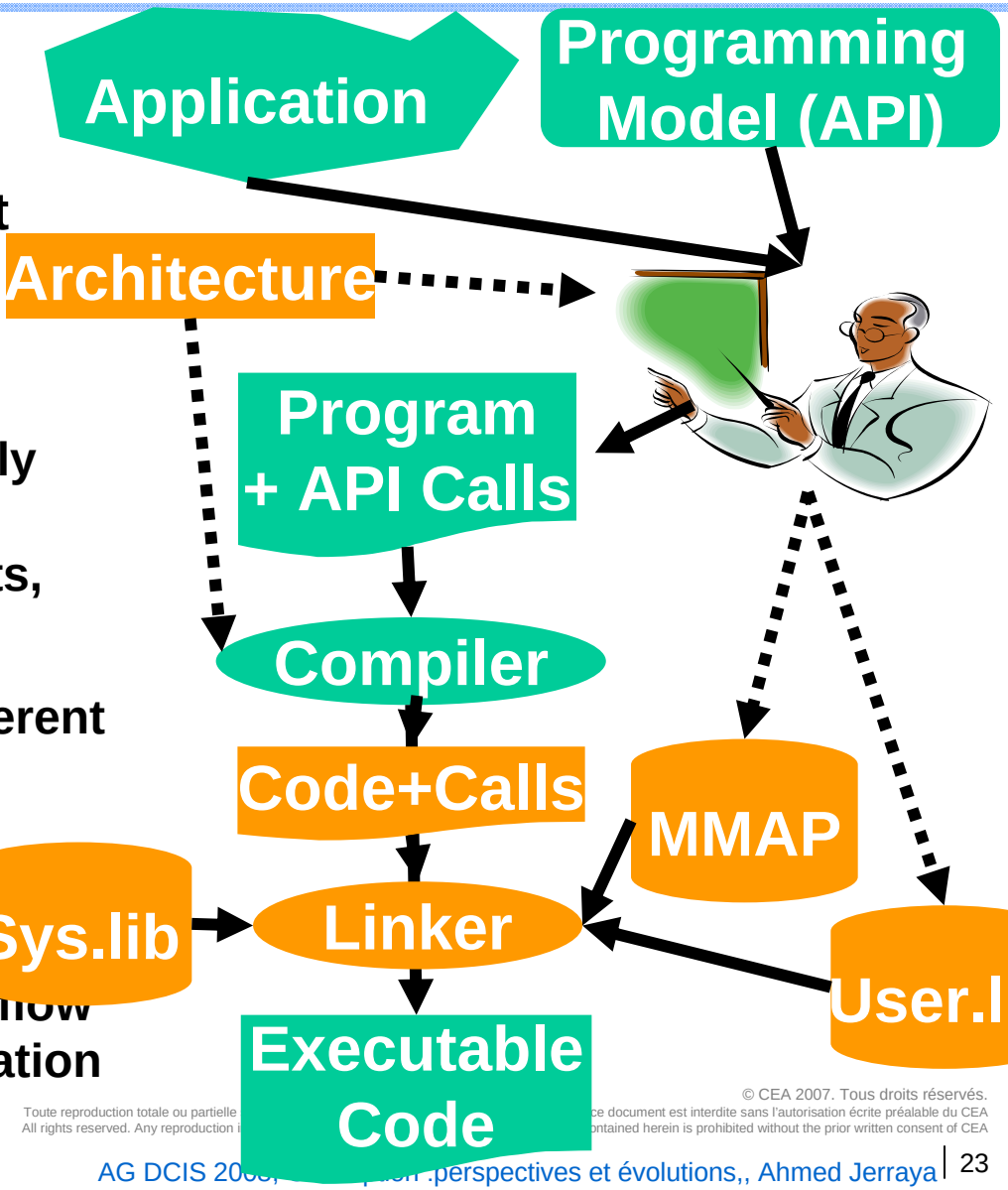
- **Layered SW Stack**
  - **Application code**
    - **SW code of tasks mapped on the CPU**
    - **INIT**
  - **HdS (Hardware dependent SW)**
    - **OS + Communication + HAL (Hardware Abstraction Layer)**
- **Different SW components need to be validated incrementally**
  - **Layers corresponds to Different SW abstraction levels**
  - **SW development platforms at each abstraction level**



Software Stack

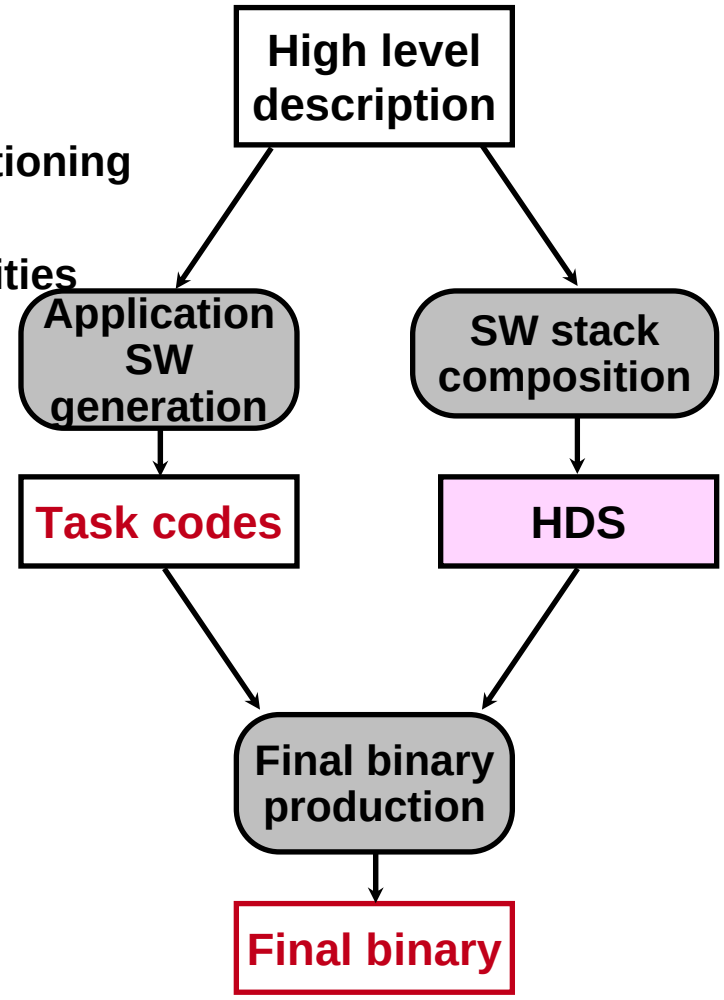
Classical SW design flow to interface HW

- **Programming Model: Abstract HW at Different level**
- **Discontinuities:**
  - **Compilation: Generally ignore the CPU environment (Interrupts, Complex I/O)**
  - **Sys.lib: adapt for different HW**
  - **MMAP: Adapt to diff… CPU-memory archite…**
  - **User.lib: to make the flow efficient for the application**

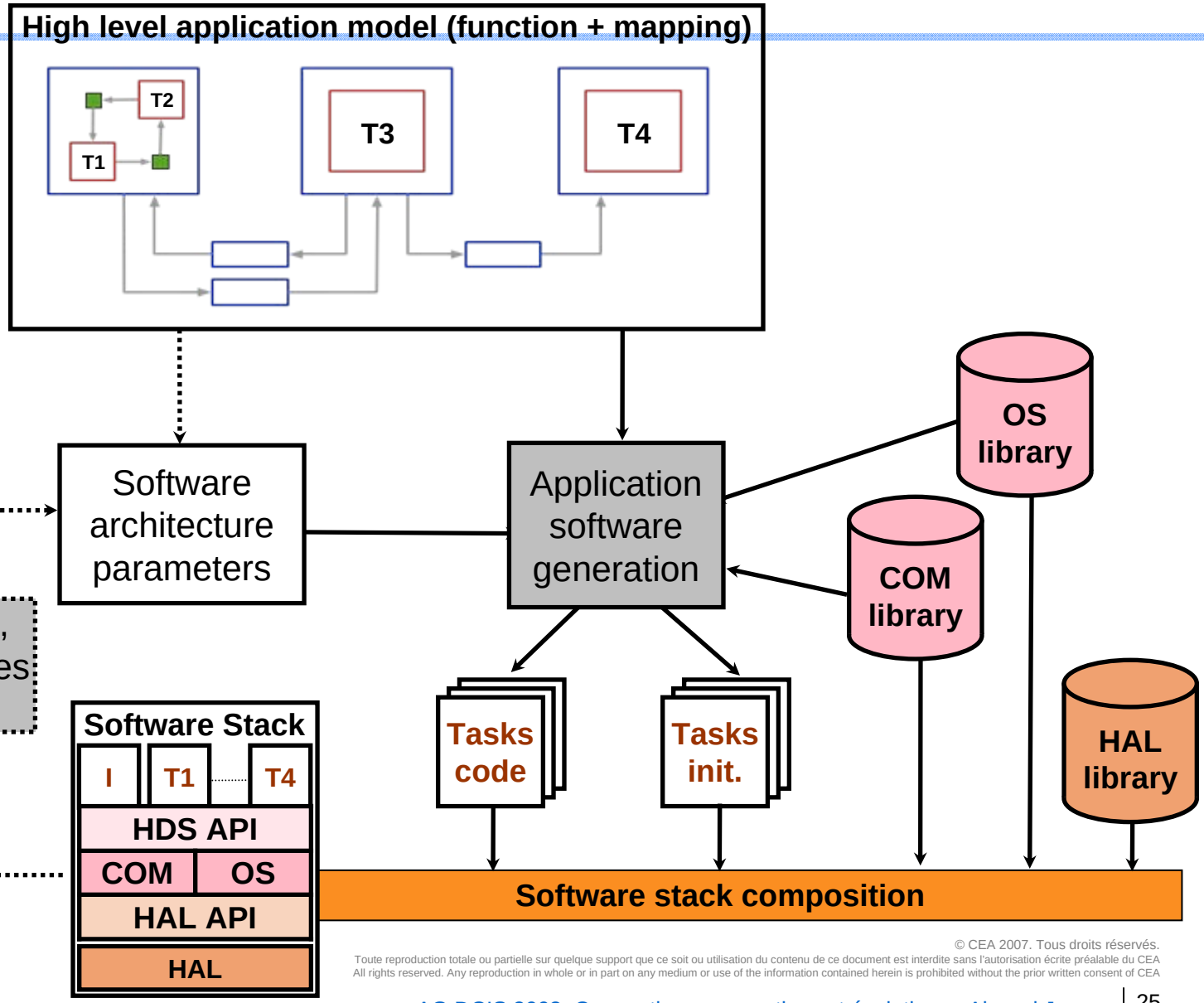


Code generation from a high level Model d

- **High level description**
  - Contains the application behavior, partitioning and resources mapping
  - Difficult to capture architecture specificities
- **SW architecture exploration**
- **Application SW generation**
  - Multitask and multiprocessor
  - Various operating systems and specific communications support
  - Speed and/or size code optimizations
- **Software stack composition**
  - Using existing or generated tasks
  - Various operating systems and specific communications support



Software generation environment



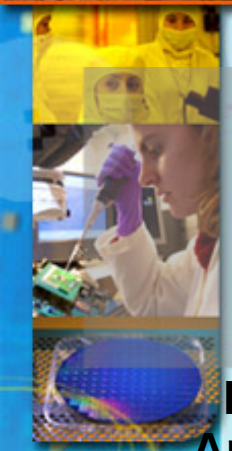
Outline

- Multiprocessor System on Chip: HW-SW Architectures
- HW/SW interfaces abstraction: Programming models
- Hardware dependent software design
- Long term trends: HW-SW codesign.

Challenges

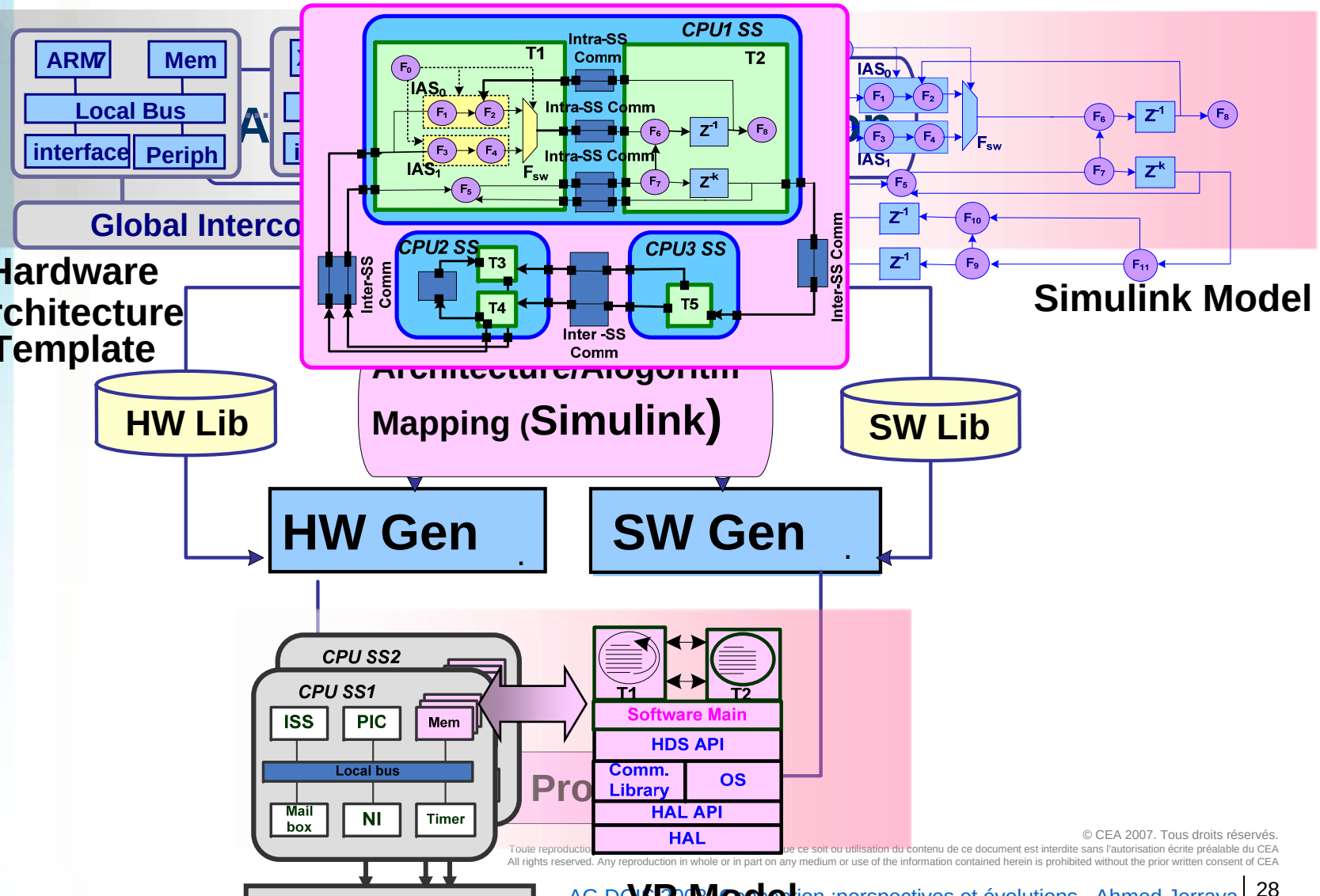
- **Affordable programming model for MPSoC end users**
  - Easy to port new application by end user
  - Based on high level programming model
- **Seamless HW-SW Integration**
  - Architecture exploration and Application SW mapping
- **Quality Proven SW Modules**
  - Ensure better design success using proven IP

Ideal Design Flow

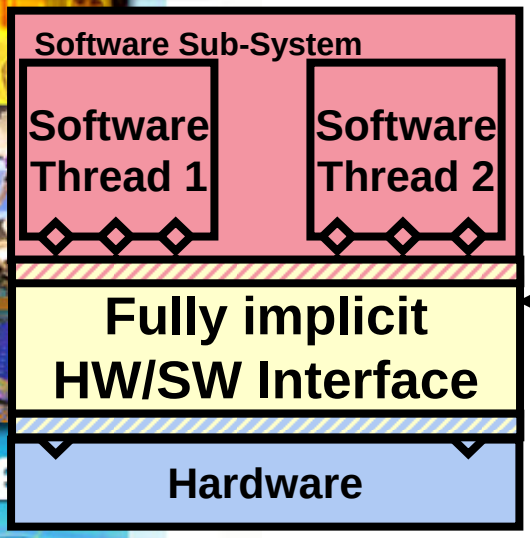


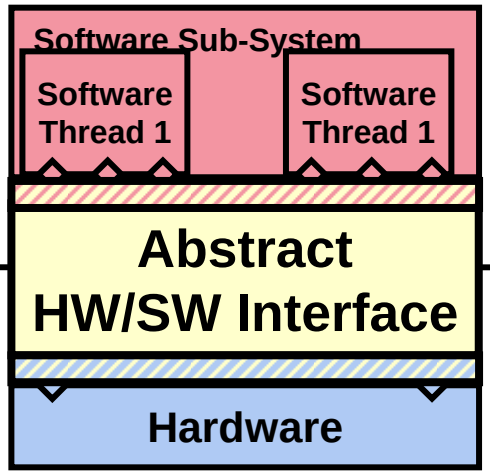
leti

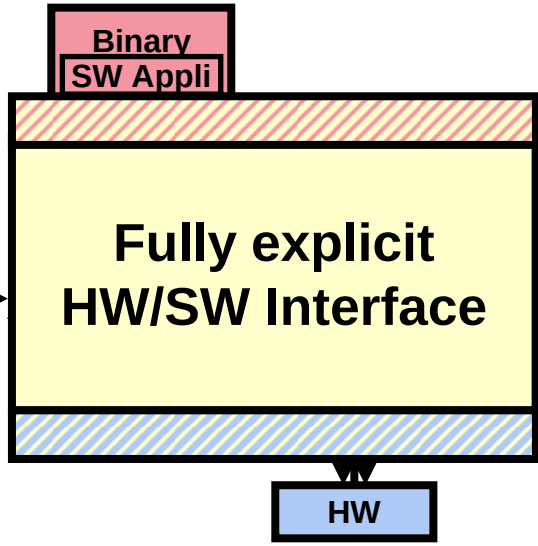
MINATEC

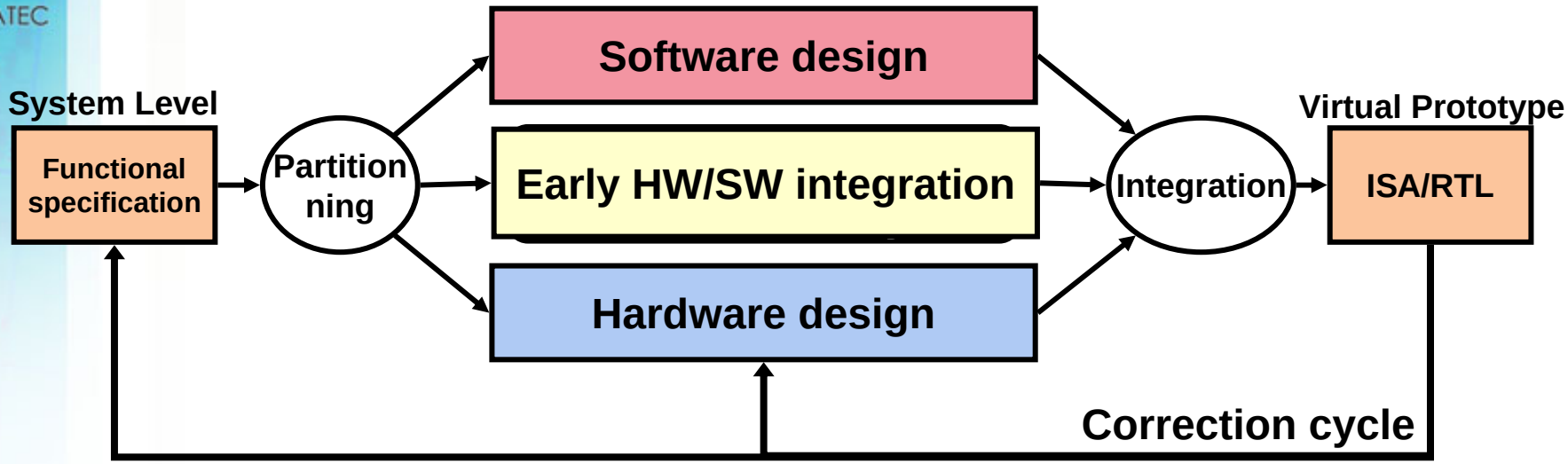


Classical HW/SW Design : The GAPS





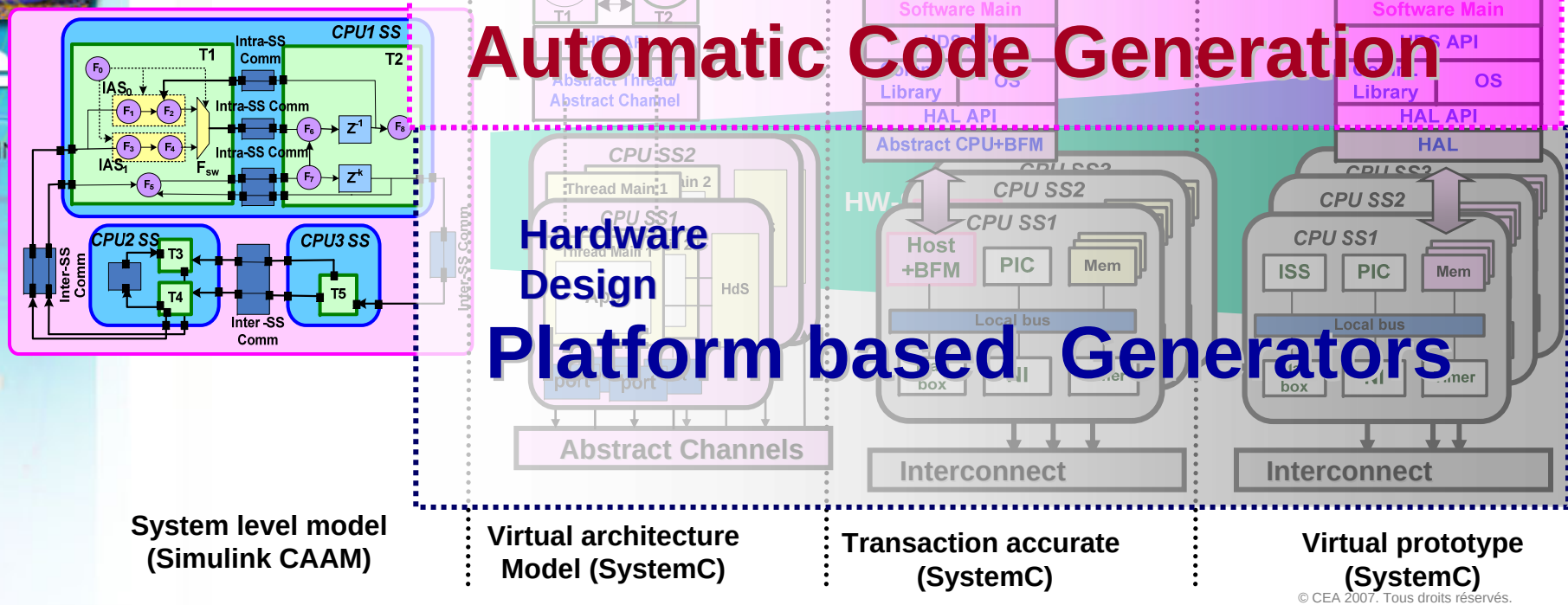




[Gerin ASPDAC 07]

HW and SW Mixed Models

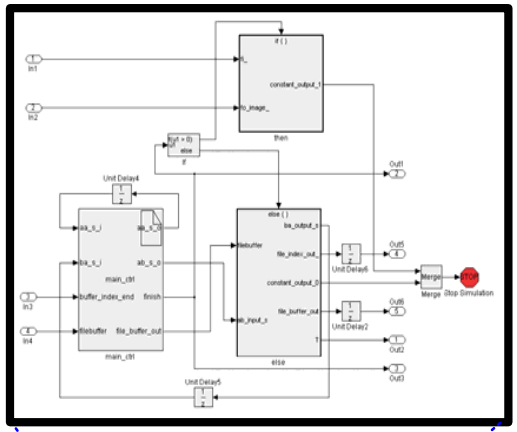
- Seamless refinement at four abstraction levels:
  - Simulink Combined Algorithm and Architecture Model (Simulink CAAM)
  - HW/SW Interfaces refinements

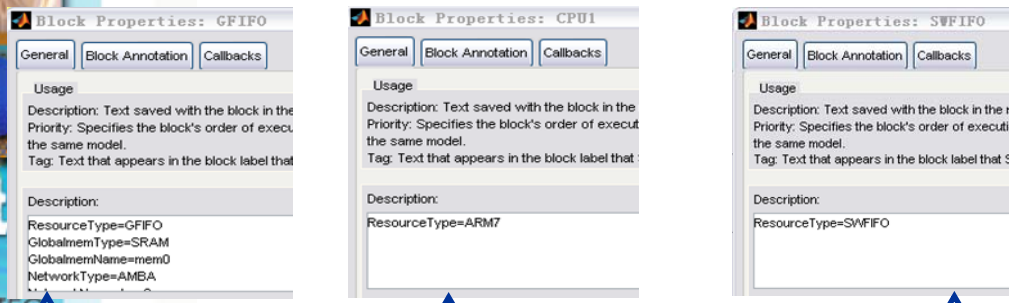


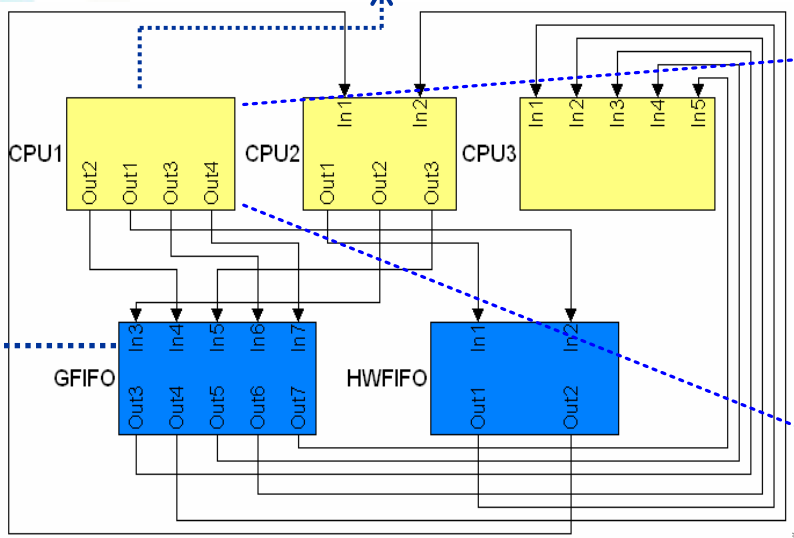
Simulink CAAM of M-JPEG Decoder

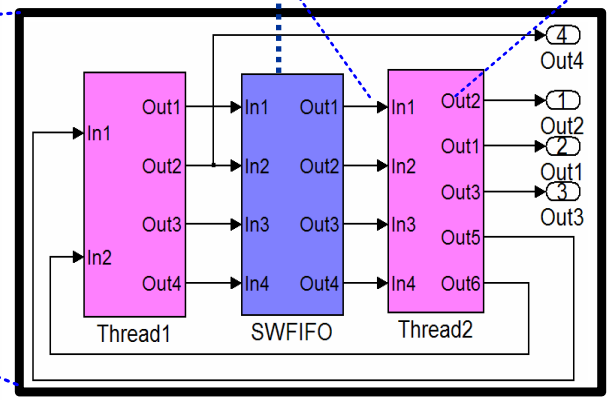
- Three CPUs (ARM, Xtensa)
- Communication Units (GFIFO, HWFIFO, SWFIFO)

[RSP2007 DAC07]









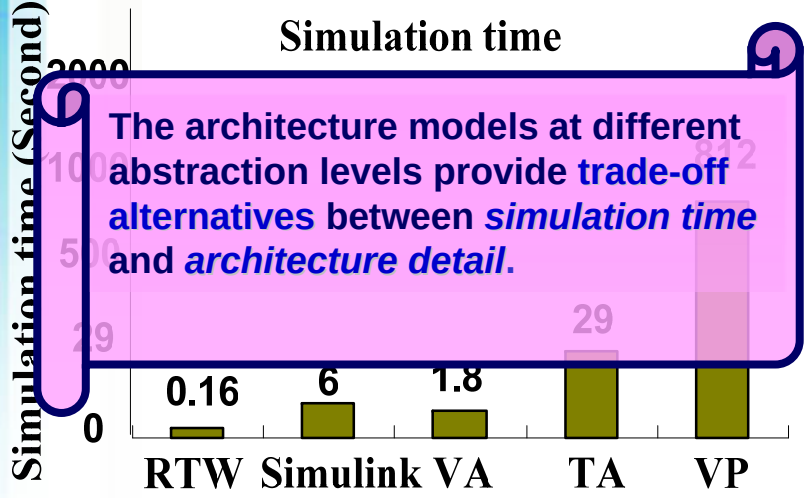
7 S-Functions
7 Delays
26 Links
4 IASs

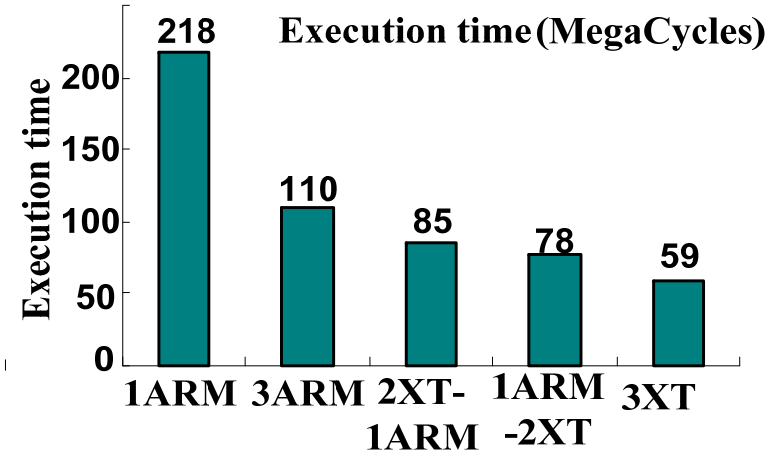
Experiment Results of MJPEG

- 10-frame QVGA (320x240) JPEG stream

RTW: **sequential program on the host machine**
Simulink: **Simulation in Simulink GUI**
VA: **System C model without HW information**
TA: **Abstract CPU + Other devices (SC TLM)**
VP: **Cycle Accurate ISS + Other devices (SC TLM)**

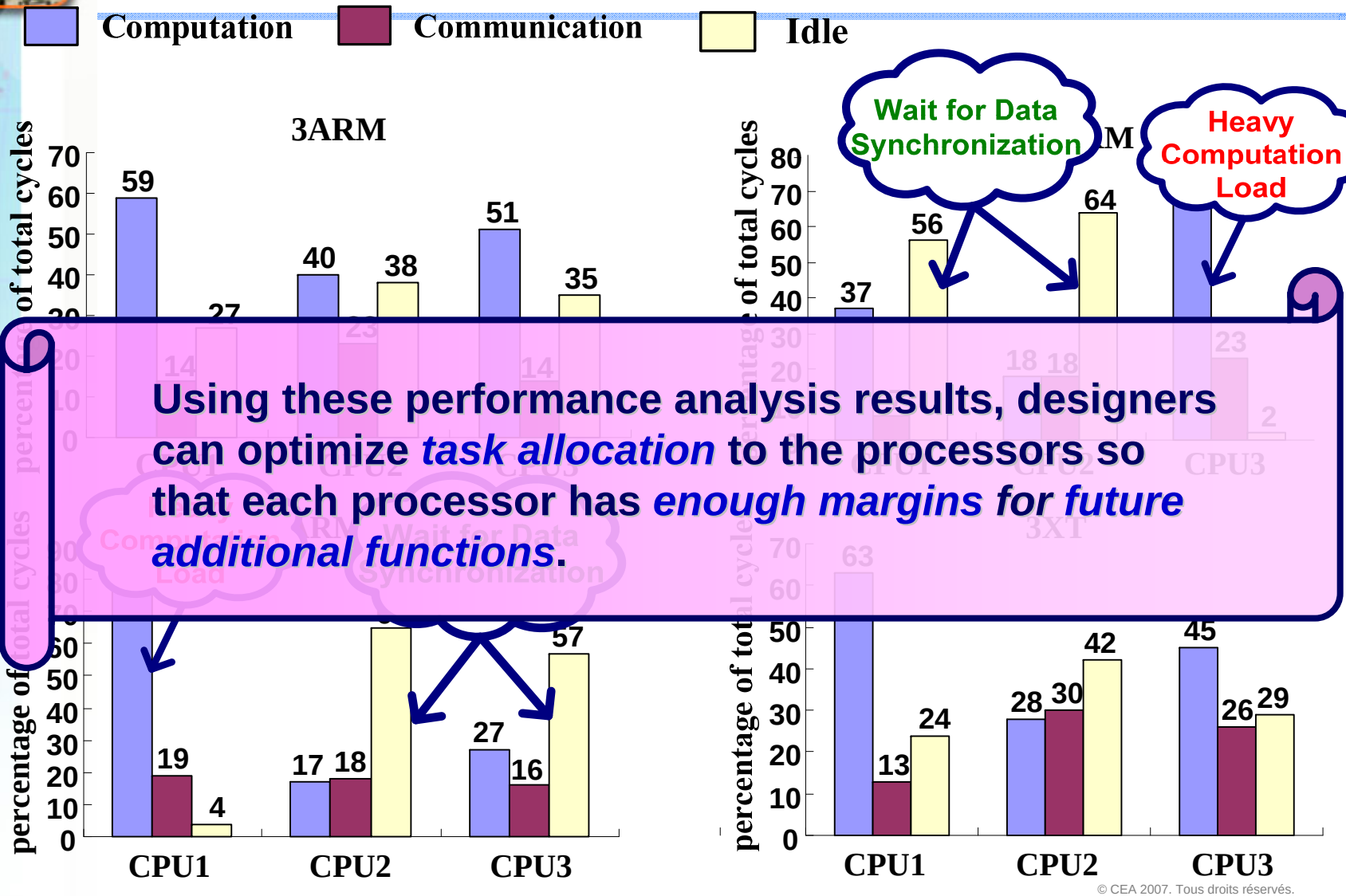
	CPU1	CPU2	CPU3
3ARM	ARM7	ARM7	ARM7
2XT1ARM	Xtensa	Xtensa	ARM
1ARM2XT	ARM	Xtensa	Xtensa
3XT	Xtensa	Xtensa	Xtensa





Three ARM7 Subsystems

Performance Analysis of MJPEG



Using these performance analysis results, designers can optimize *task allocation* to the processors so that each processor has *enough margins for future additional functions*.

Conclusions

- **HW-SW interfaces**
  - **Heterogeneous MPSoC require multiple SW stacks**
  - **Application-specific hardware & software**
- **Programming models for SoC**
  - **Abstract specific architectures for a better match between HW & SW**
  - **Different abstraction levels for early application SW & HDS validation**
- **HDS design: Platform based SW development at different abstraction levels**
- **Future trends: Architecture Exploration**
  - **HDS-CPU codesign**
  - **HW-SW interfaces architecture exploration**

Acknowledgement

- Prof F. Pétrot, Prof. F. Rousseau from TIMA
- PhD Students K. Popovici, P. Gerin from TIMA
- JR Lequepeys, CEA LETI

micro and nanoelectronics
microsystem
ambient intelligence
biology and health
image chain

Innovation for industry

Loyalty
Entrepreneurship
Team work
Innovation

leti

MINATEC
PÔLE D'INNOVATION

cea

You will be welcome to the

10th Leti Annual Review

24 - 26 June 2008 at Minatec

For more information :
http://www.leti.fr

leti

MINATEC
POLE D'INNOVATION

cea