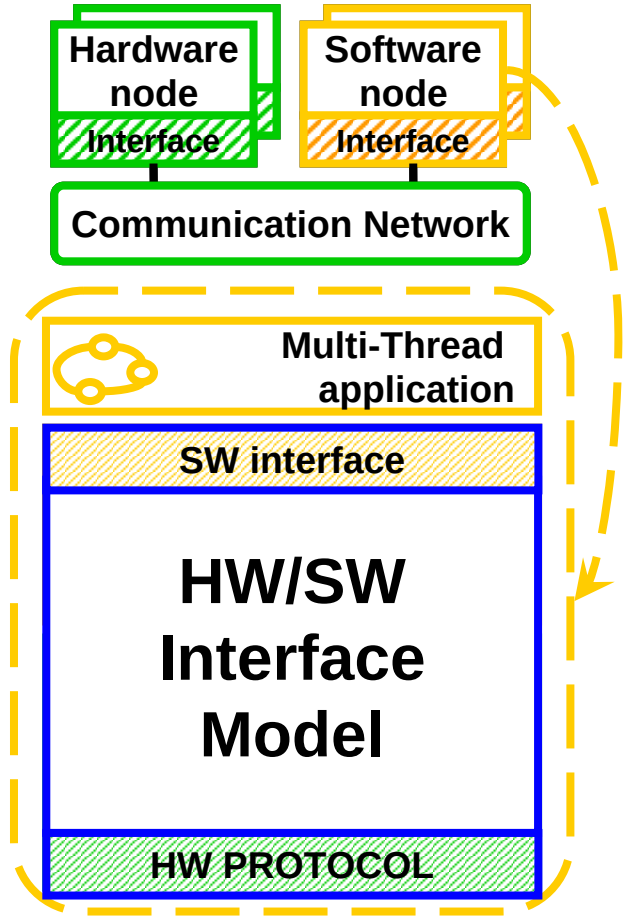
Flexible and Executable Hardware/Software Interface Modeling for Multiprocessor SoC Design Using SystemC

Patrice GERIN
Alexandre CHUREAU
Hao SHEN
Aimen BOUCHHIMA
Ahmed JERRAYA

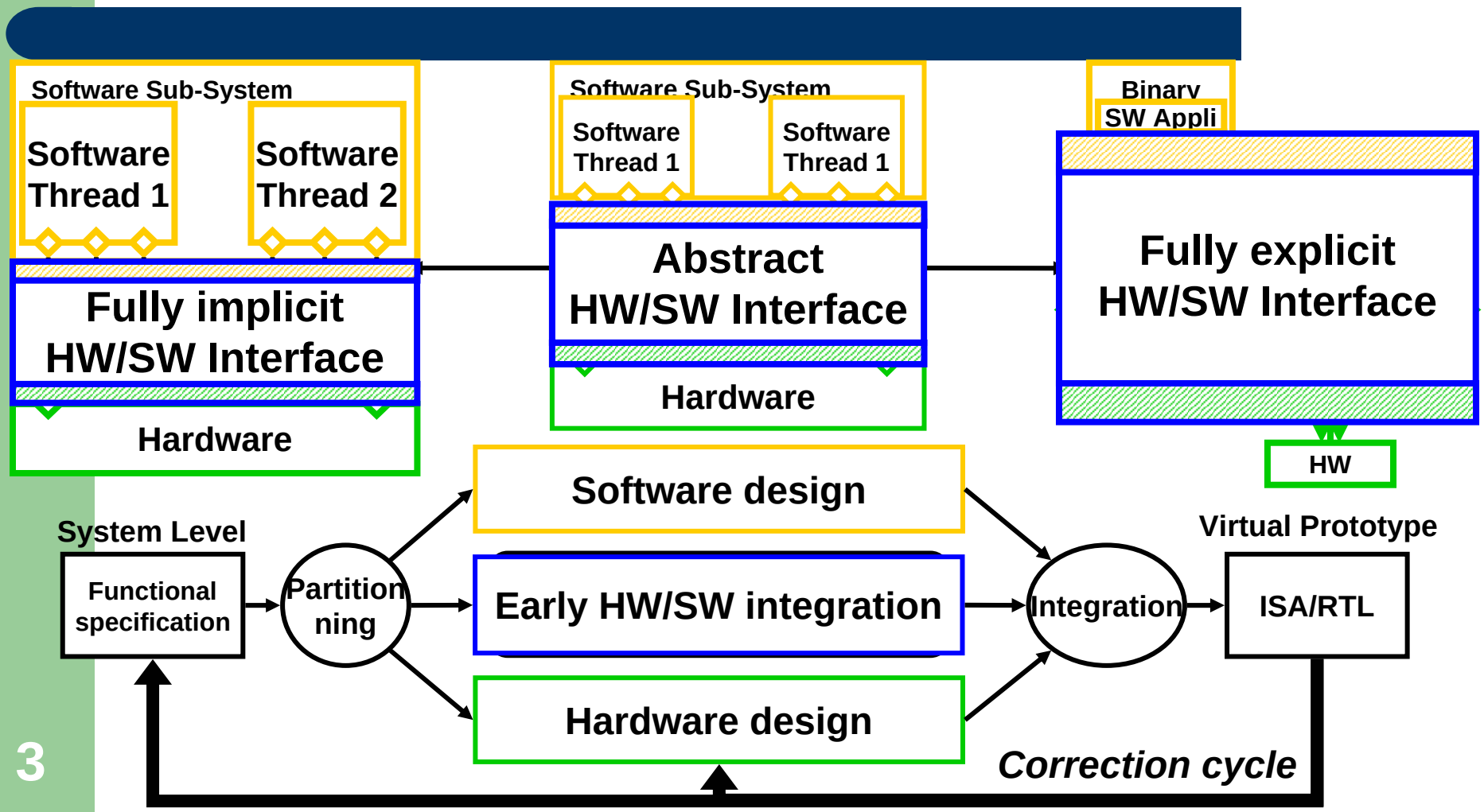
TIMA Laboratory– SLS Group
46 Avenue Félix VIALLET
38031 Grenoble Cedex France
Email : patrice.gerin@imag.fr

Definition : HW/SW Interfaces for MPSoC

- Heterogeneous MPSoC :
  - HW nodes
  - SW nodes
- Software node :
  - Specific CPU subsystem
    - GPP, DSP, ASIP
    - I/O, Memory architecture
  - Layered software architecture
    - High level application code.
    - Hardware Dependent Software (HDS)
- HW/SW Interfaces for SoC Design
  - Hide HDS and specific HW
  - Provide SW API to the HL code
  - Offered different abstraction levels



Classical HW/SW Interfaces Abstraction Models : The GAP



Abstract HW/SW interfaces State Of Arts

- Modeling HW/SW Interfaces
  - SW oriented approach : fully implicit hardware
    - OS validation can not include interaction with hardware
    - No accurate performances estimation
  - HW oriented approach : Binary software
    - OS debug is fastidious
    - Simulation time too long
- System Level Design Methods
  - Fixed architecture Model
  - Restricted application/architecture (TTL,DSOC)
- What is needed :
  Executable HW/SW interface model allowing early OS debug and accurate performance estimation

Objectives & Contributions

- **Objectives**
  - **Early Operating System validation**
  - **Early performances measurement**

- **Contributions**
  - **A unified executable model of HW/SW interfaces**
  - **A new design flow allowing fast and accurate simulation of abstract HW/SW interfaces**

Outline

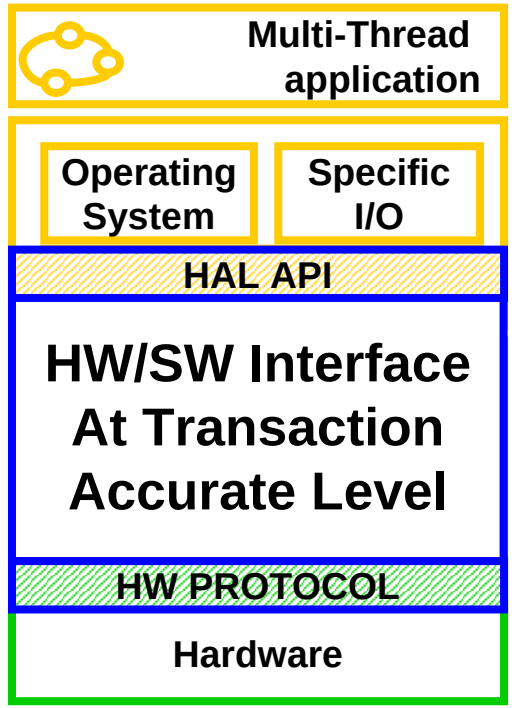
- Introduction
- Hardware/Software Interfaces modeling *Transaction Accurate Level*
- Executable model in SystemC
- Experiments
- Future Works, conclusion

Outline

- Introduction
- **Hardware/Software Interfaces modeling *Transaction Accurate Level***
- Executable model in SystemC
- Experiments
- Future Works, conclusion

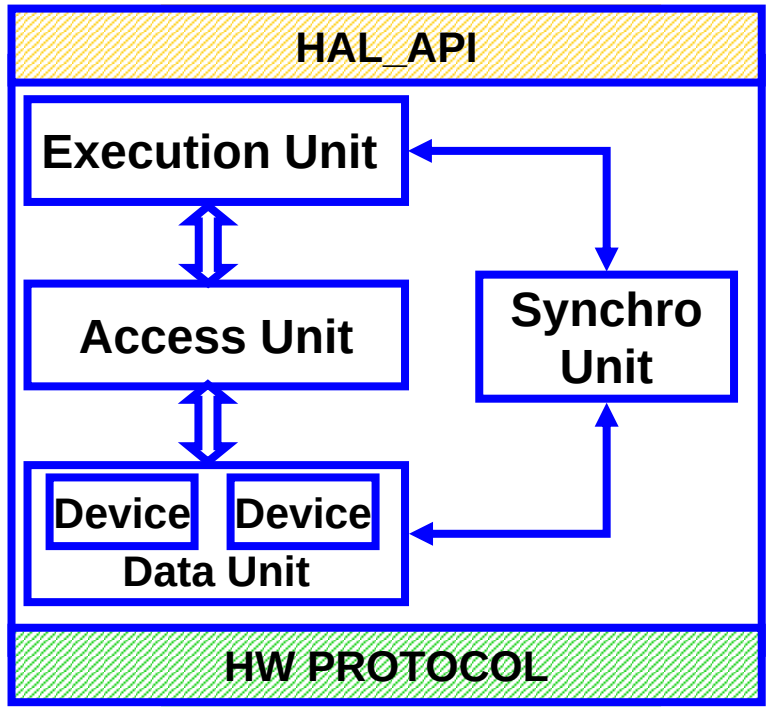
Interface modeling at *Transaction Accurate Level*

- To be abstracted
  - HAL software layer
  - Details of CPU subsystem
- SW interface : HAL API
  - Context switch
  - Spin lock
  - IO Read/Write
- HW interface : HW protocol
  - VCI, AMBA,…
  - Specific HW interface (FIFO)



Interface modeling at *Transaction Accurate Level*

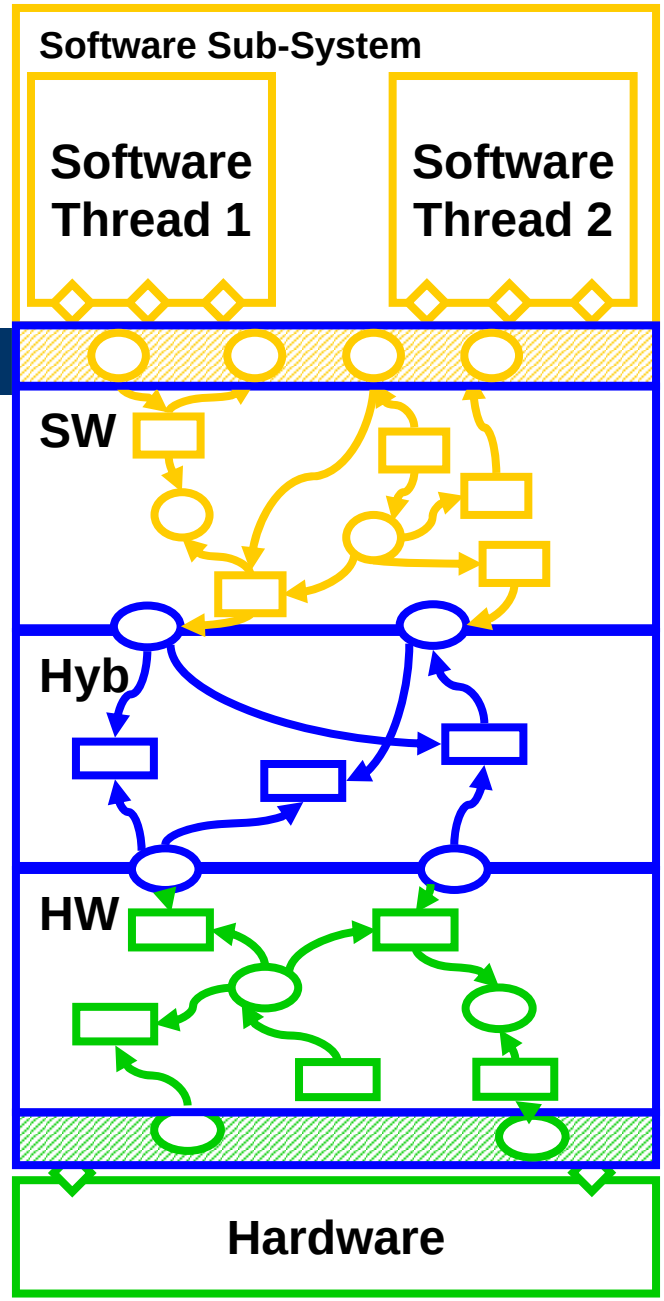
- Execution Unit modeled the parallel computation
- Access Unit models communication in the CPU subsystem
- Data Unit encapsulate model of physical devices
- Synchronization Unit model interrupt management and controller mechanism



This set of elements implement adaptation between the software and the hardware interface.

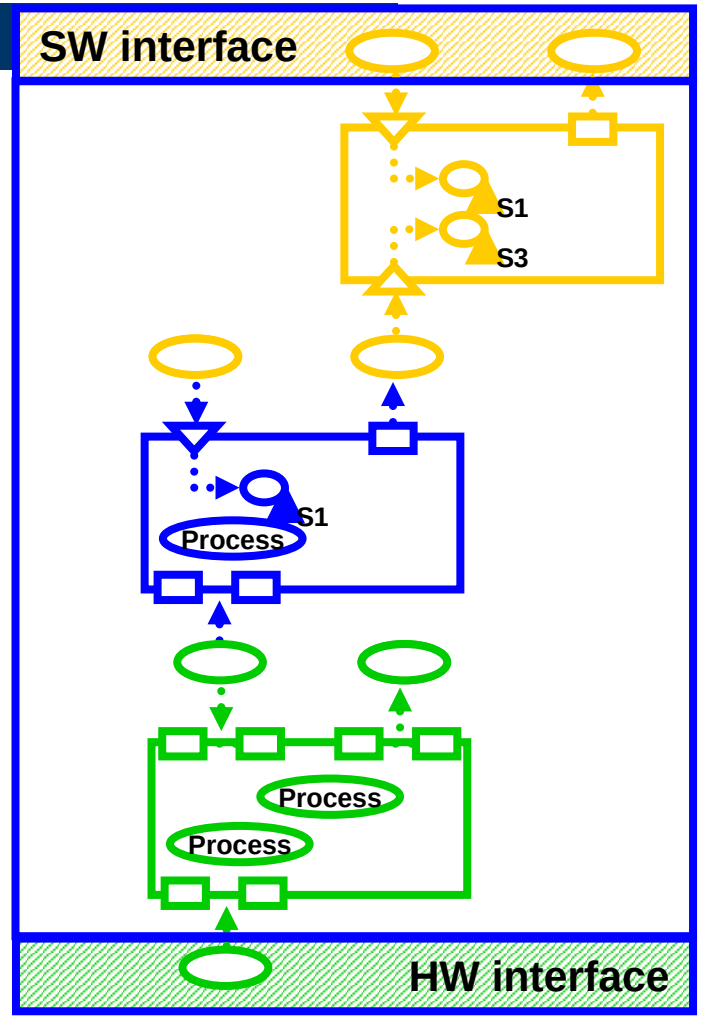
Services for HW/SW interfaces adaptation

- Both HW and SW interfaces are modeled as set of services (provided/required)
- Component based interface adaptation
  - Software elements
  - Hardware elements
  - ***Hybrid elements***



Interface implementation

- Interface element concept
  - Elements require or/and provide services
  - A services stand for a functionality
- A Hardware/Software interface consist in assembling software, hybrid and hardware elements.
- Already used to model :
  - Software (Zitterbart, Gauthier, Kriaa and Sarmento)
  - Hardware (Grasset)



Outline

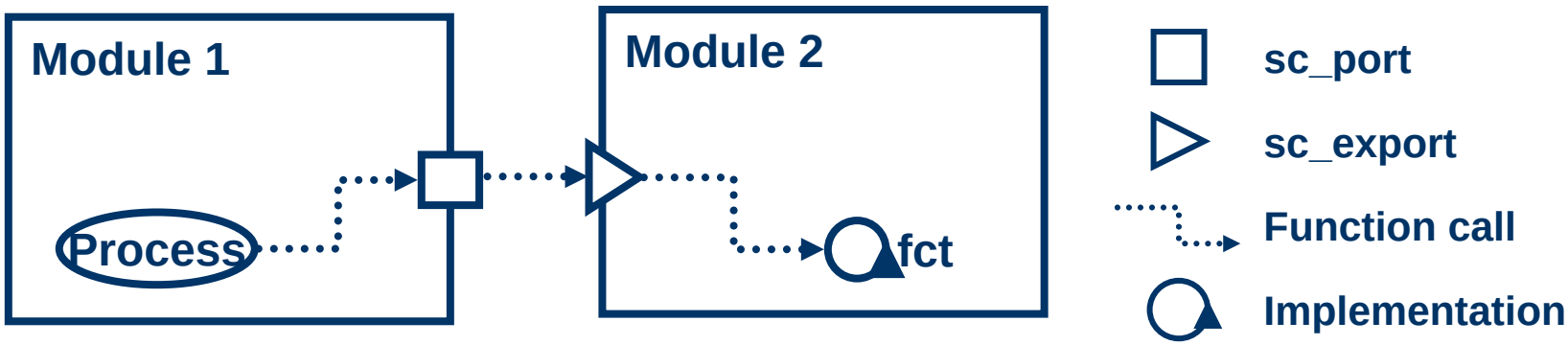
- Introduction
- Hardware/Software Interfaces modeling : *Transaction Accurate Level*
- **Executable model in SystemC**
- Experiments
- Future Works, conclusion

Software Services in SystemC

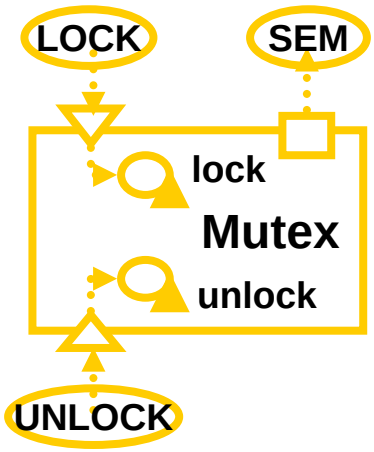
- SystemC *sc_interface* mechanism

Equivalence between SystemC objects and service concept

sc_interface	⇔	service
sc_export	⇔	Provided service
sc_port	⇔	Required service
sc_module	⇔	Element



Software element

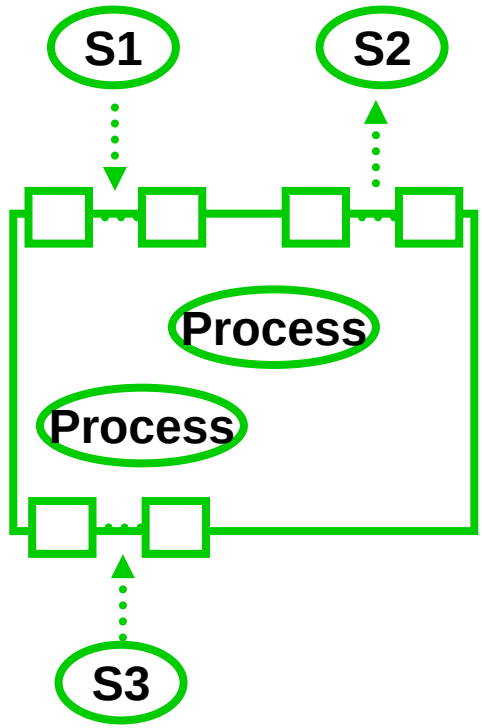


- Only exported C++ method to implement SW services
- No SystemC *SC_THREAD*, *SC_CTHREAD* or *SC_METHOD*
- No SystemC *wait()*

```
class LOCK : public sc_interface
{
   virtual int lock() = 0;
};

SC_MODULE(Mutex),
public LOCK, public UNLOCK
{
   sc_export<LOCK>          pLock;
   sc_export<UNLOCK>        pUnlock;
   sc_port<SEM>             pSem;

   int lock();
   int unlock();

   SC_CTOR(Mutex)
   :    pLock("pLock"),
        pUnlock("pUnlock")
        pSem("pSem")
   {
      pLock(*this);
      pUnlock(*this);
   }
};
```

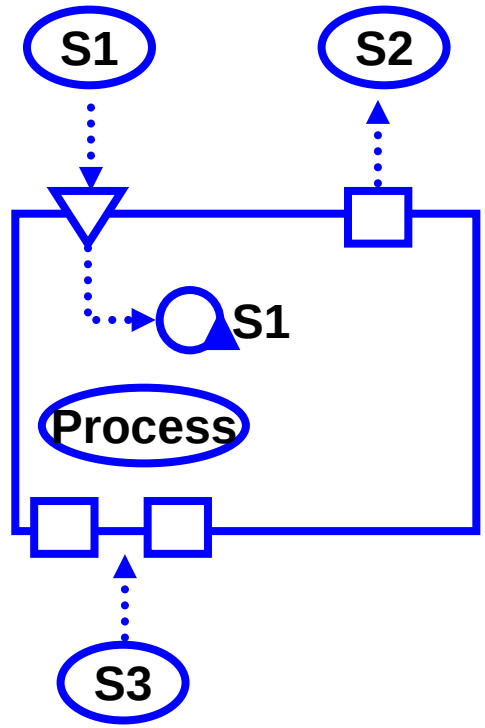
Hardware element

- Hardware
  - Services are implemented by *SC_THREAD* and accessible through a set of ports
  - No exported C++ method
- “Standard” SystemC implementation



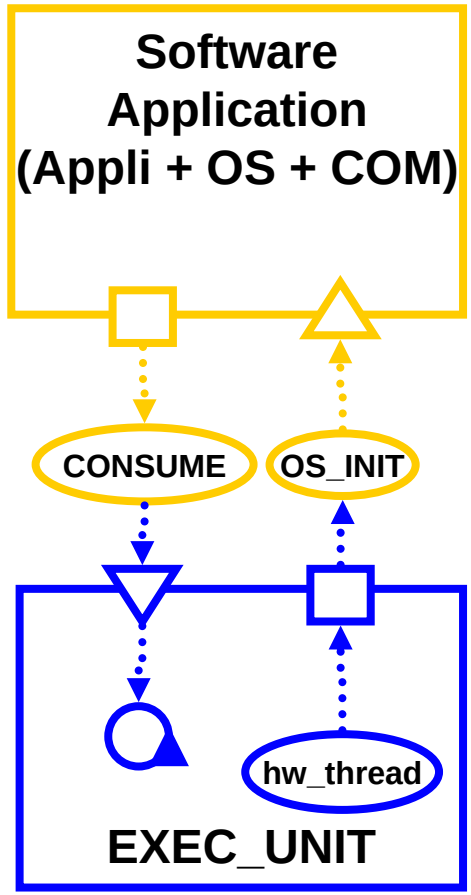
Hybrid element

- Hybrid combine HW and SW element:
  - exported C++ method to implement SW services
  - Contain SystemC threads to implement HW services
  - Can call SystemC wait to introduce time.



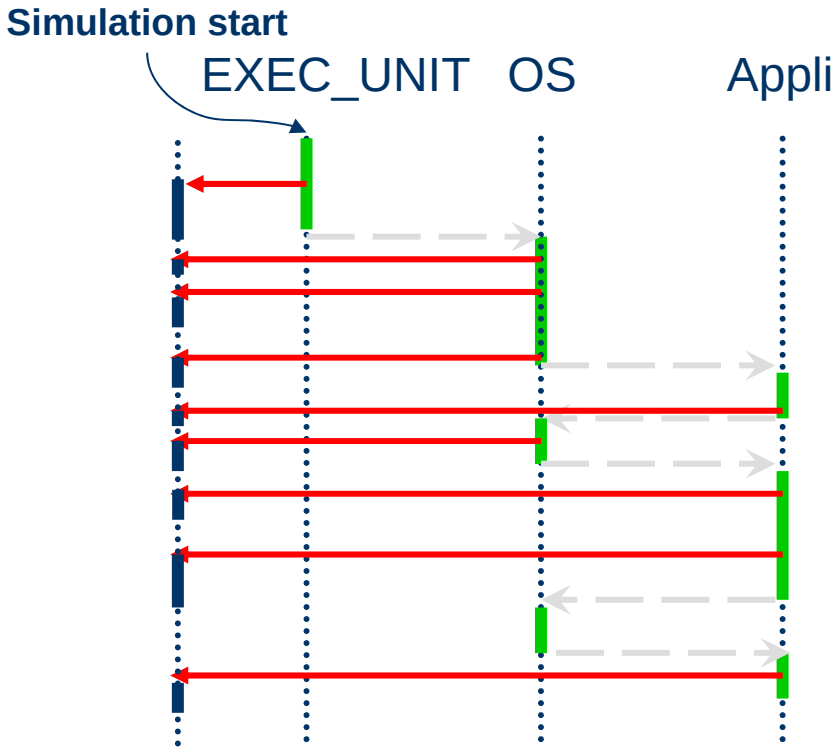
Hybrid element and Software execution model

- Hybrid elements are the key elements to model sequential software execution
  - Used to implement execution unit (CPU)
  - A hardware thread represent the processor
  - This hardware thread can model low level initialization.
  - Call the *OS_INIT* software service provided by the application
  - All the software is executed sequentially
- Software simulation time is introduced with annotation in the application.
  - Calls to a *consume* service will modeled the time consumed by the software in the processor thread context.



Software simulation detailed

- **EXEC_UNIT model the low level initializations and boot the OS**
- **OS and application are executed sequentially**
- **Call to *CONSUME* allow SystemC kernel to manage HW concurrent simulation.**
- ***consume* can also be called from the elements of the TA model to increase accuracy.**

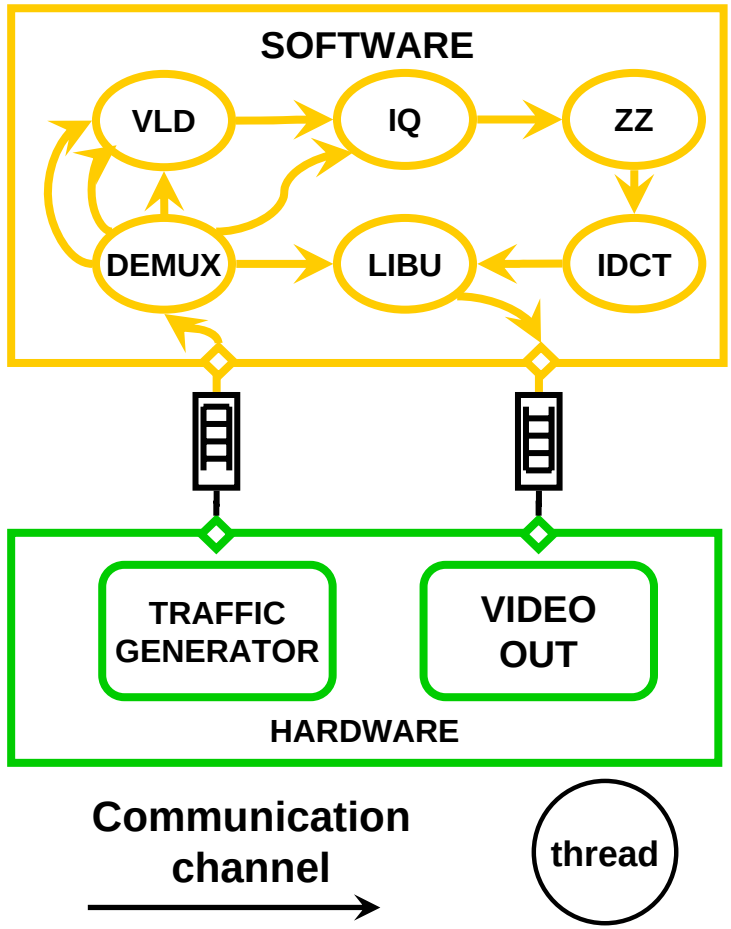


Outline

- Introduction
- Hardware/Software Interfaces modeling *Transaction Accurate Level*
- Executable model in SystemC
- **Experiments**
- Future Works, conclusion

Motion JPEG application : System Level Model

- 6 software and 2 hardware tasks
- Execution model synchronized with communications

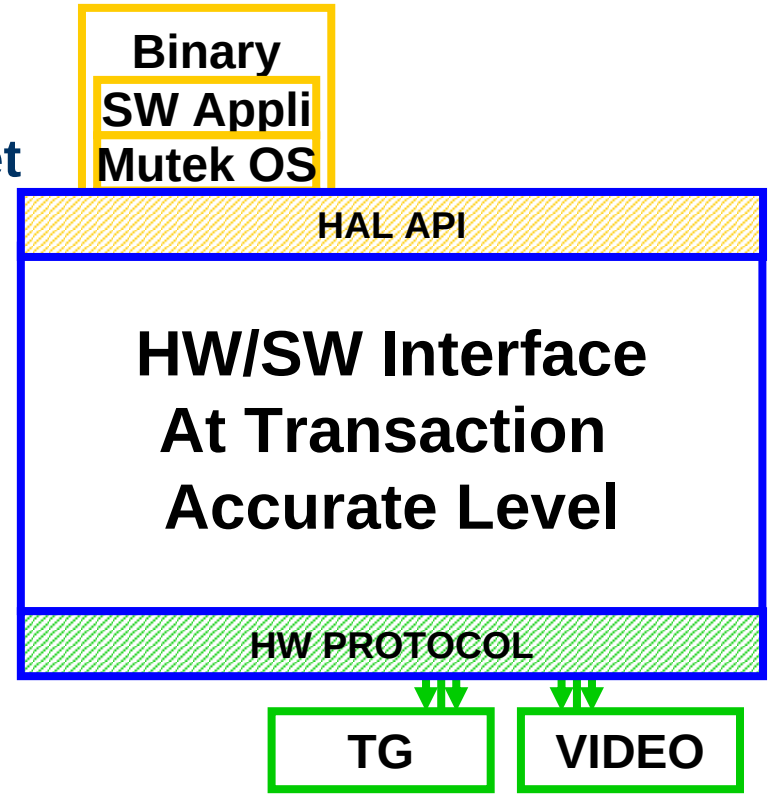


- ✓ High Simulation speed
- ✓ Easiest functional validation
- ✗ No Operating System details
- ✗ No details on communications

Motion JPEG application : Virtual Prototype Model

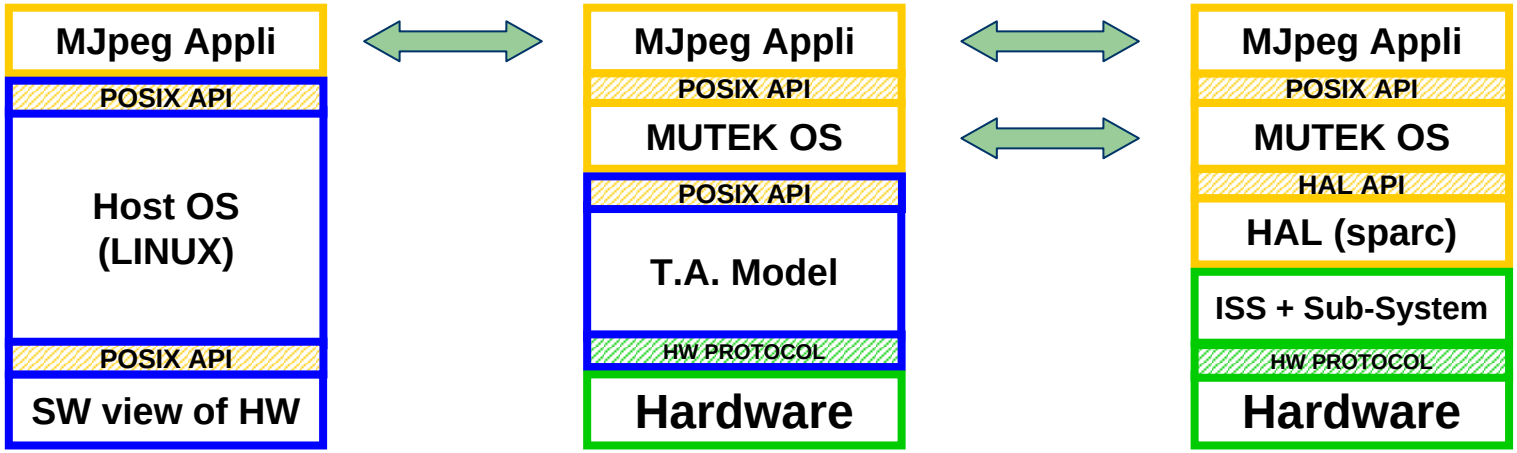
- Software tasks executed on a POSIX compliant OS : MUTEK
- Software interpreted by the target processor ISS
- Rest of the system at RTL level

✓ Detailed communications
✓ Performances precision
✗ Fastidious Operating System validation
✗ Very slow simulation



Experiment simulations

- 3 simulations
- Same SW application code in the 3 models
- Same OS code in T.A. and V.P.



System Level

0.017s/frame
(No CPU)

Transaction Accurate

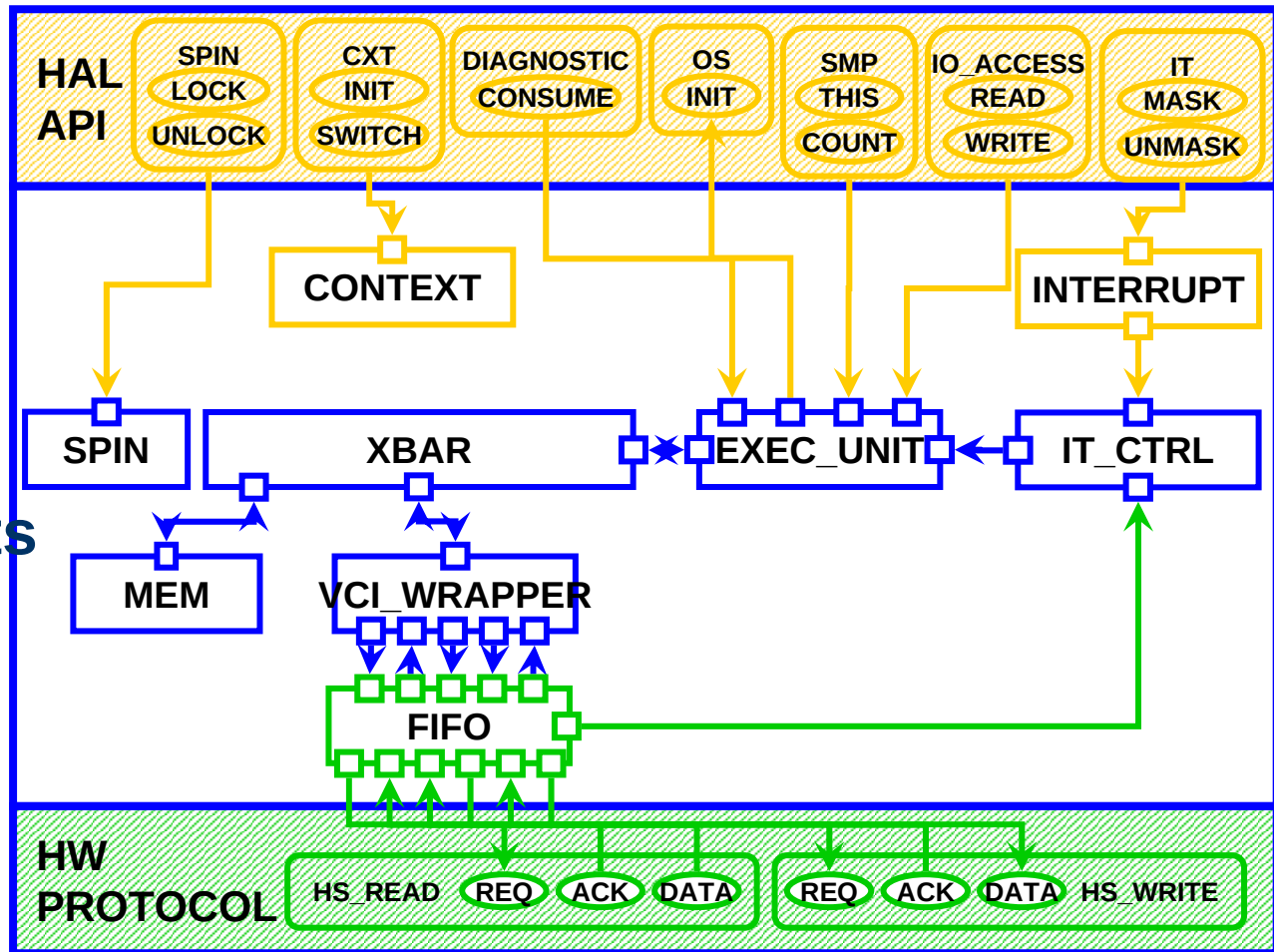
1.2s/frame
(2 Execution Unit)

Virtual Prototype

235s/frame
(2 Sparc ISS)

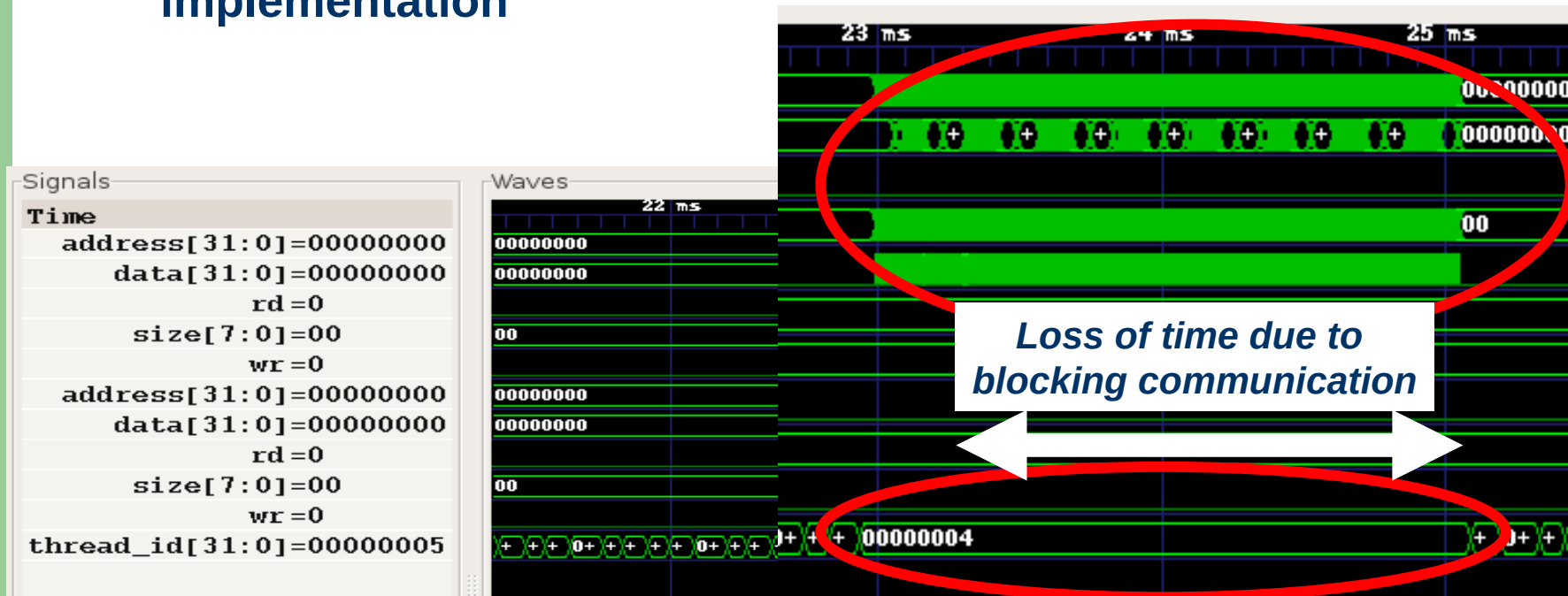
Motion JPEG application : Transaction Accurate Model

- Pure software elements
  - CONTEXT
  - INTERRUPT
- Pure Hardware
  - VCI FIFO
- Hybrid elements
  - CROSSBAR
  - VCI WRAPPER
  - EXEC_UNIT
  - …



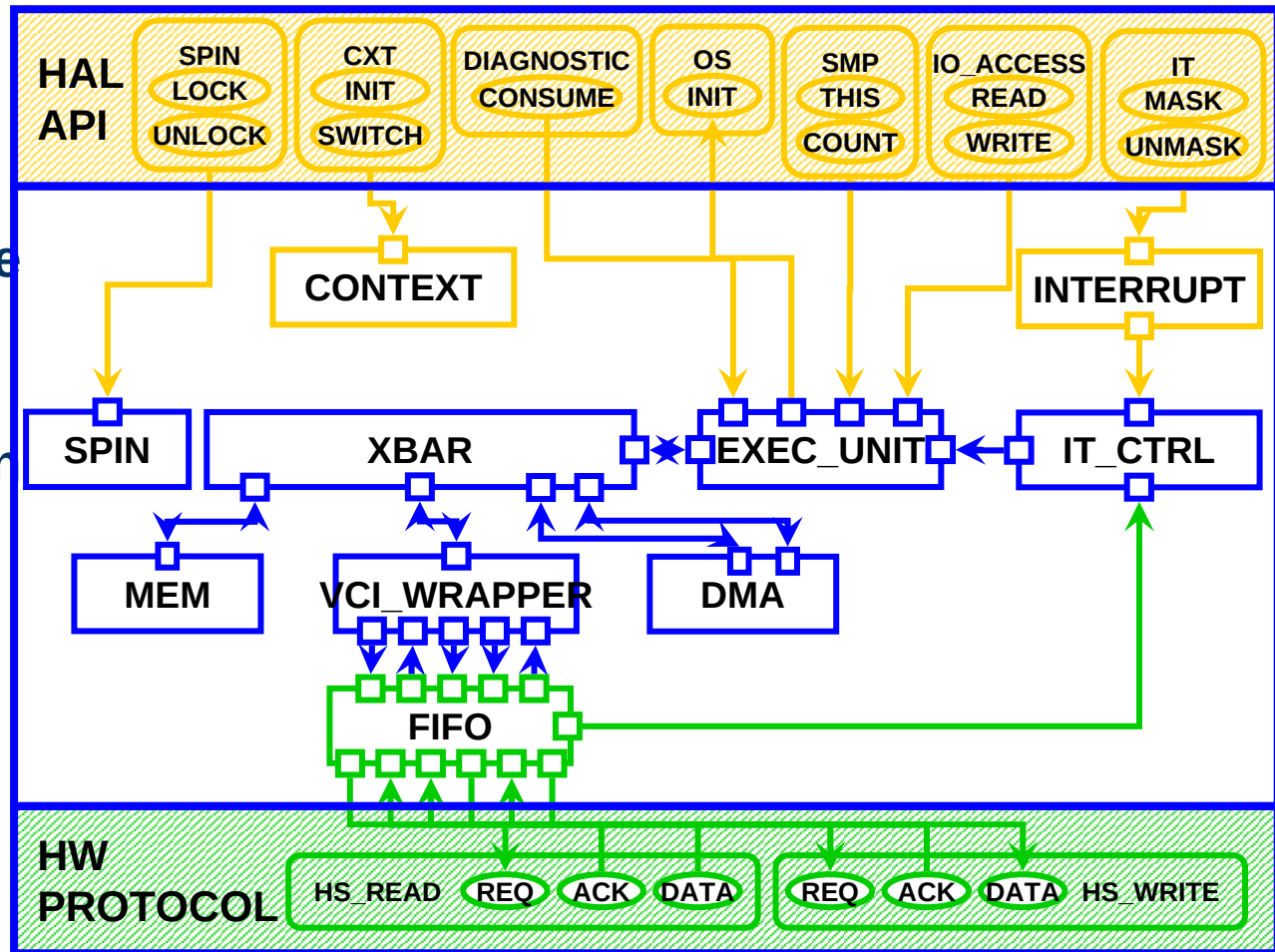
Simulation results at Transaction Accurate Level

- More detailed trace than System Level
- Operating System debug capabilities (global synchronizations, timing behaviour, standard debugger)
- 195 times faster than the Virtual Prototype for this implementation



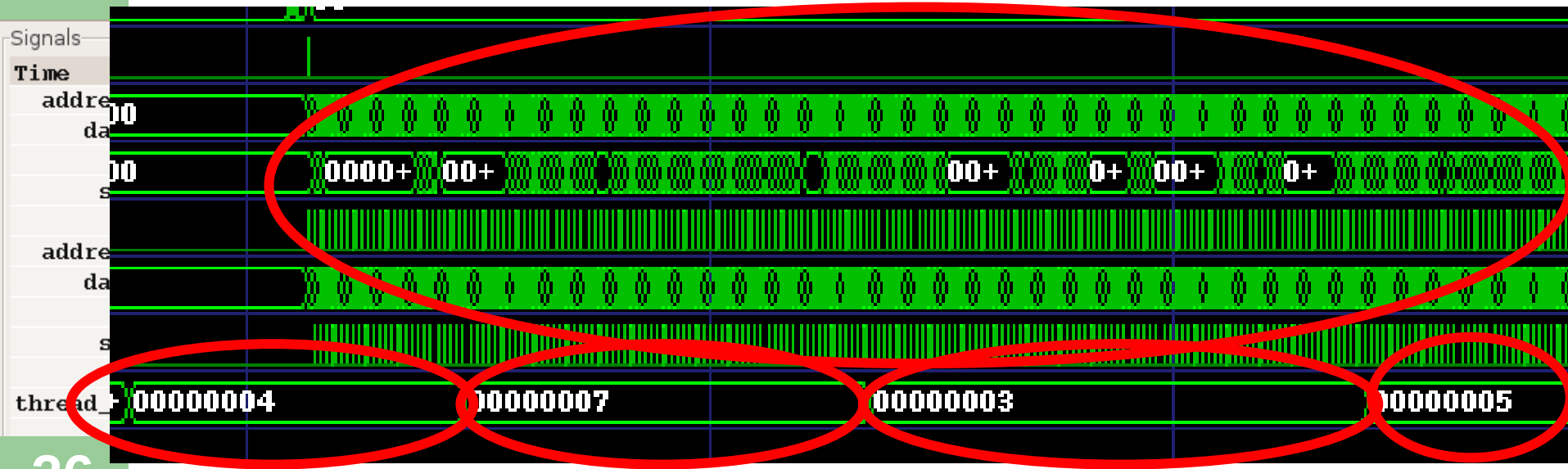
Modifying the HW/SW interface model

- Hybrid DMA element added
- Application modified to use DMA
- Interleaving of communication computation



Modifying the HW/SW interface model

- Simulation allow to validate the DMA communication and to verify the effect on the application threads execution

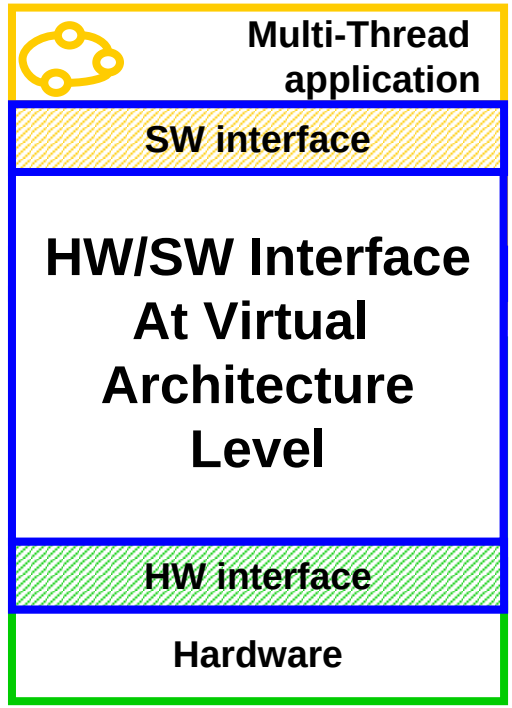


Outline

- Introduction
- Hardware/Software Interfaces modeling *Transaction Accurate Level*
- Executable model in SystemC
- Experiments
- **Future Works, conclusion**

Future works

- Apply the proposed approach to other abstraction level :
  - Virtual Architecture, abstract the operating system and the specific communication
- HW/SW interface design automation to enable :
  - Architecture exploration.
  - Refinement



Conclusion

- Executable Hardware/Software interface model.
- Results : Earlier HW/SW integration
  - Fast and accurate simulation of full MJPEG system
  - Executable model in a standard environment (SystemC)
- Benefits :
  - Operating System Validation
  - Early performance estimation

Thank you

Questions ?