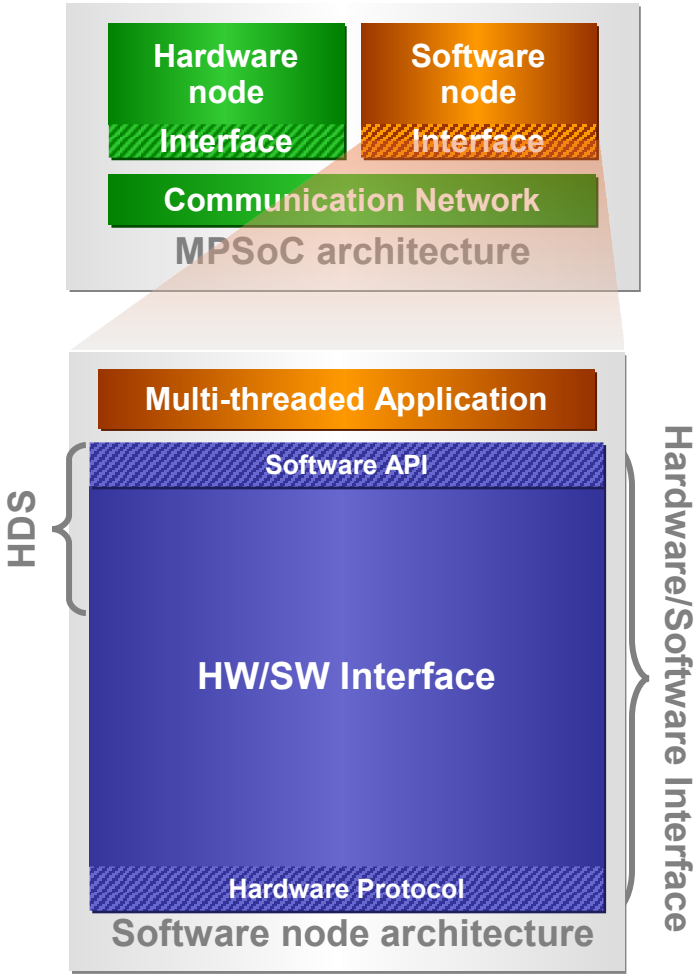
Flexible and Executable HW/SW Interface Modeling For MPSoC Design Using SystemC

Frédéric PÉTROT

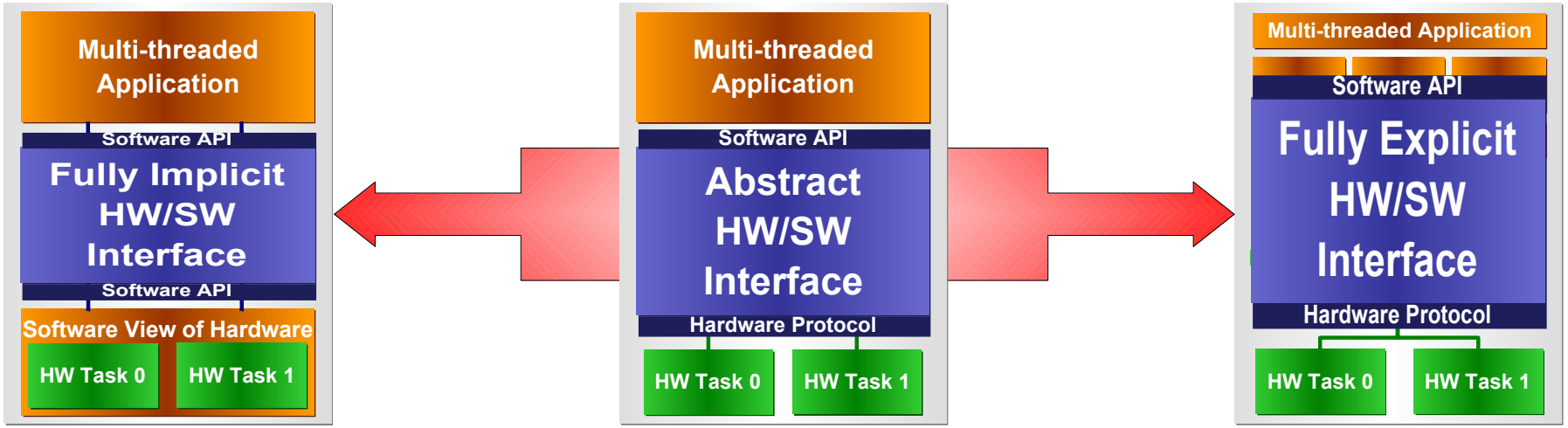
Thanks to: A. Bouchhima, P. Gerin & A. Jerraya

Definition : HW/SW Interface 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 Hardware
  - Provide SW API to high level code
  - Provide HW protocol
  - Offered different abstraction levels

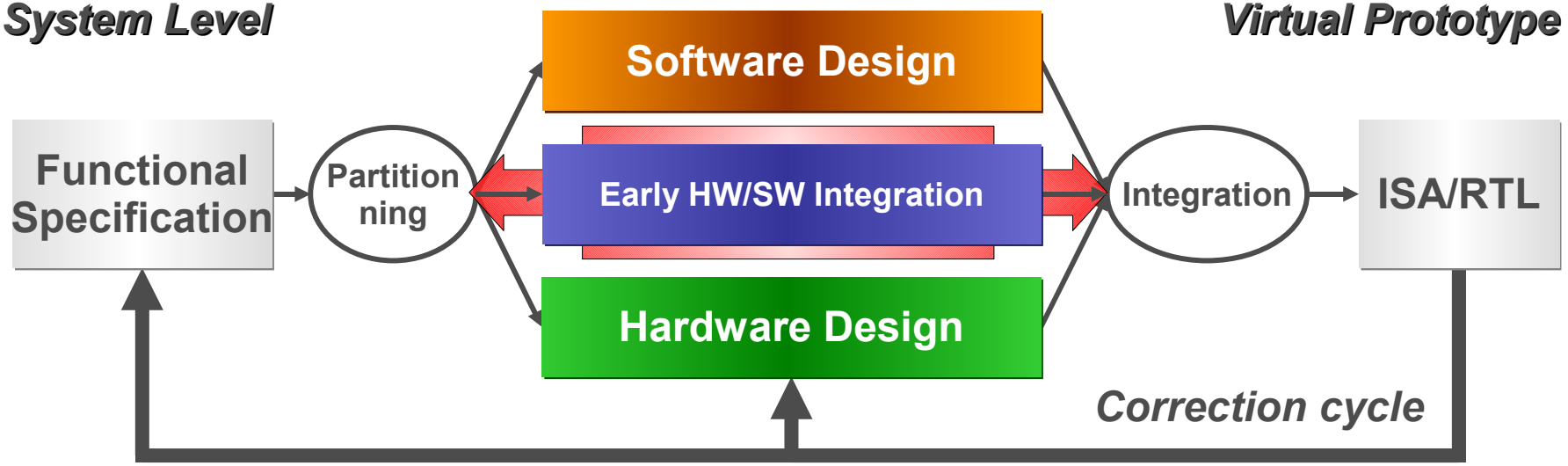


Classical HW/SW Interfaces Abstraction Models : The GAP



System Level

Virtual Prototype



Abstract HW/SW interfaces State Of Art

- 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)
- We need 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

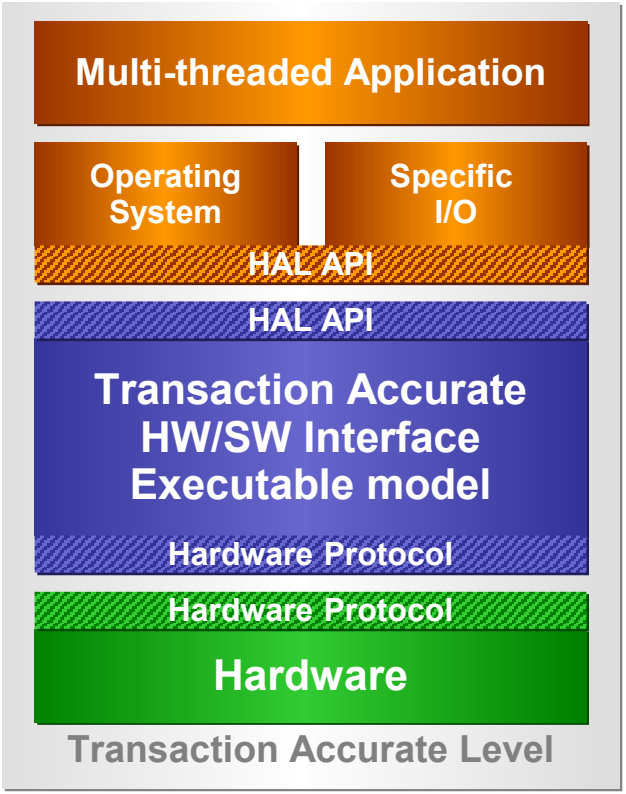
- Introduction
- Hardware/Software Interfaces modeling *Transaction Accurate Level*
- Executable model in SystemC
- Experiments
- Future Works, conclusion

Outline

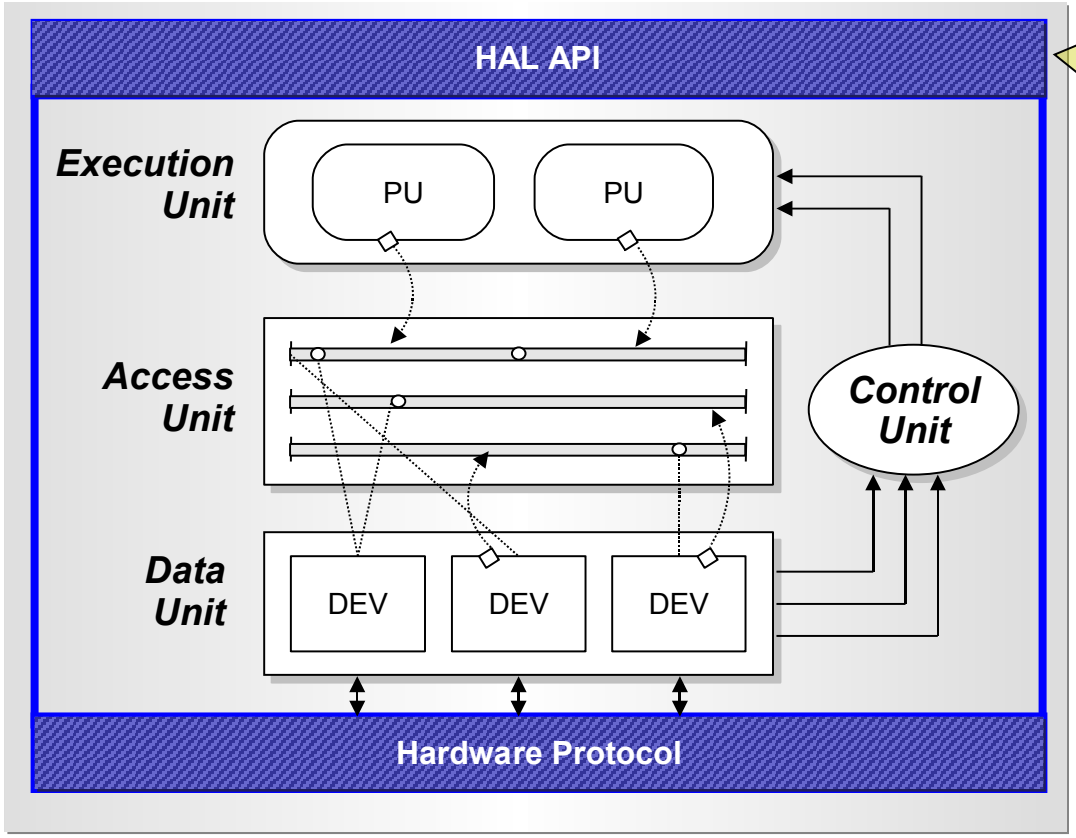
- Introduction
- **Hardware/Software Interfaces modeling *Transaction Accurate Level***
- Executable model in SystemC
- Experiments
- Future Works, conclusion

Hardware Software Interface modeling at *Transaction Accurate Level*

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



Hardware Software Interface modeling at *Transaction Accurate Level details*



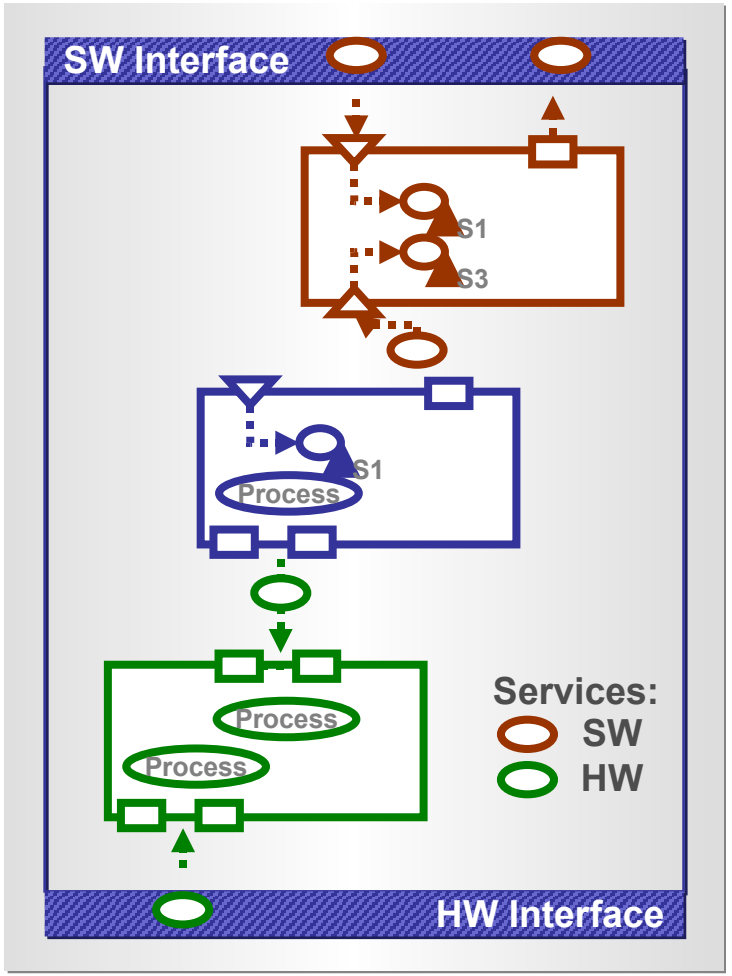
HAL API = Set of all SW services provided by individual components of the model

- Each unit may provide its own SW services
- Each "unit" model may be broken down in components decorated with timing information

Address space
Access Port
Map Port
reference
PE : Processing Element
DEV : device

HW/SW interface adaptation

- Both HW and SW interfaces are modeled as a set of services
- Component based interface adaptation
  - **Software Elements**
  - **Hardware Elements**
  - ***Hybrid Elements***



Outline

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

Executable model in SystemC

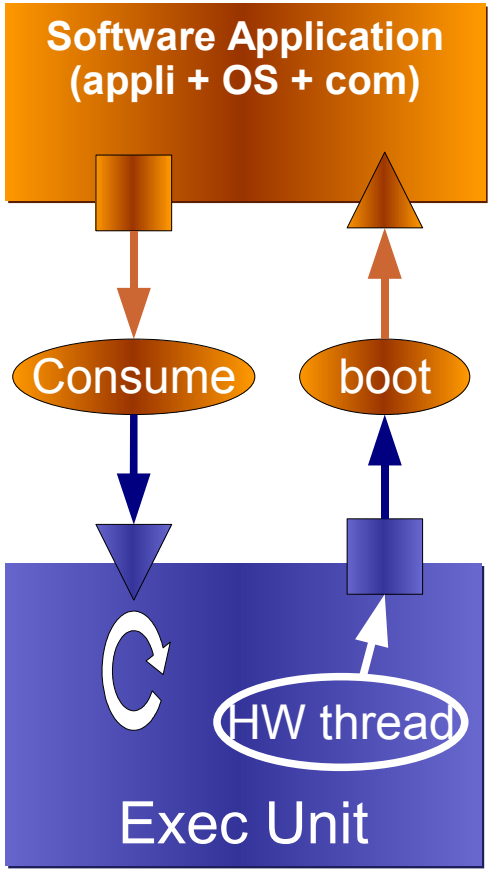
- Software elements
  - Based on SystemC *sc_interface* mechanism
    - sc_export to provide a service (a function)
    - sc_port to use a service
  - Only classical C++ methods implement software services
  - No SystemC SC_THREAD, SC_CTHREAD or SC_METHOD
  - No SystemC wait
- Hardware elements
  - All SystemC specificities can be used

Executable model in SystemC

- Hybrid elements
  - Combine HW and SW element
    - Exported C++ methods to implement software services
    - Calls to SystemC wait introduce time
    - Contain SystemC threads to implement HW services
  - Key element to model software sequential execution
    - More details in the demonstration

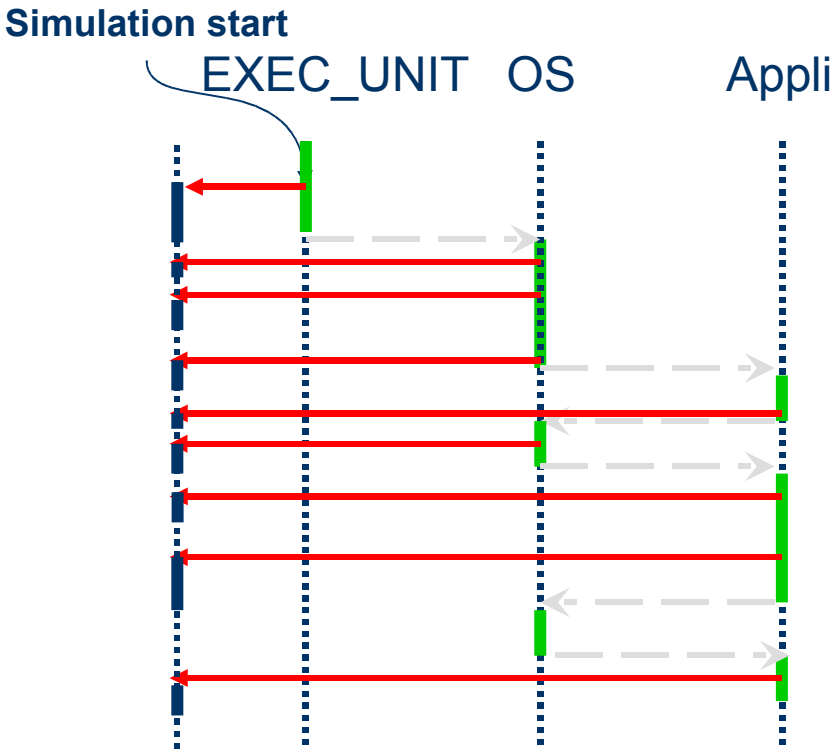
Hybrid element and Software execution model

- 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 model 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

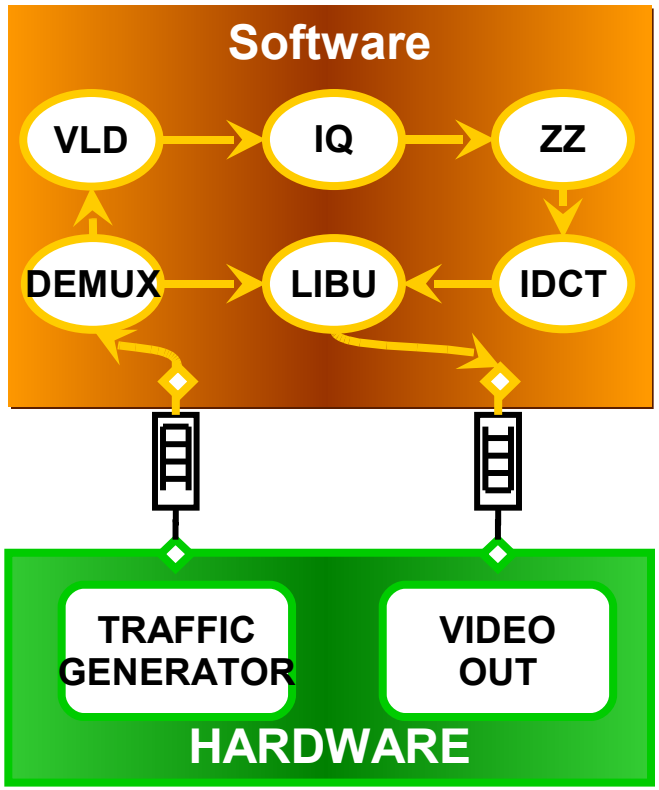
- 6 software and 2 hardware tasks
- Execution model synchronized with communications

✔ High simulation speed

✔ Easiest functional validation

✗ No Operating System details

✗ No details on communications



Motion JPEG application : Virtual Prototype Model

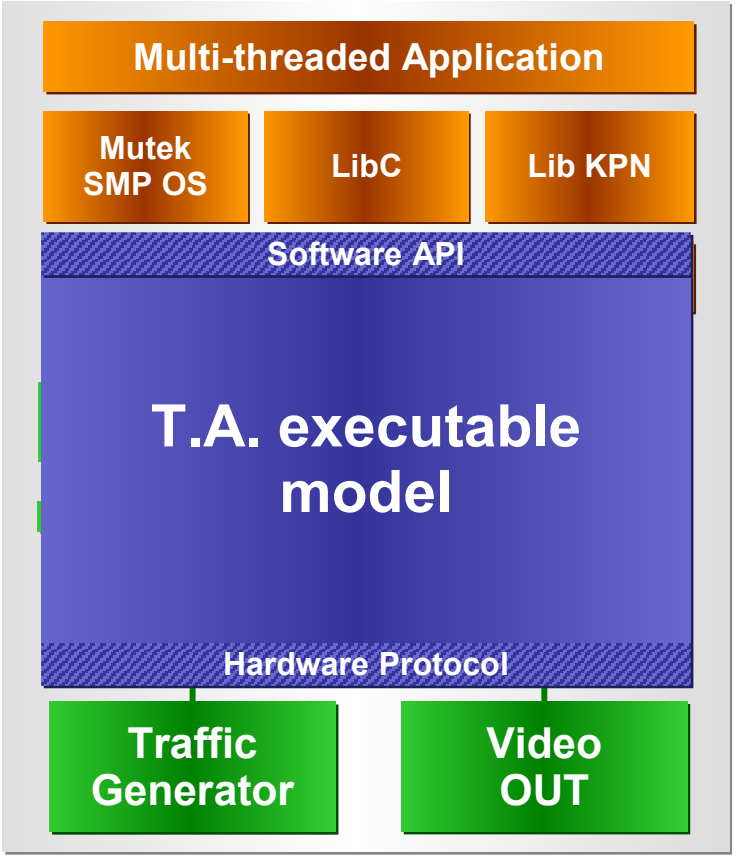
- Software tasks are executed on a POSIX compliant OS with SMP support : MuteK
- Software Interpreted by N SPARC ISS
- Rest of the system at RTL level

✔ Detailed communication

✔ Performances precision

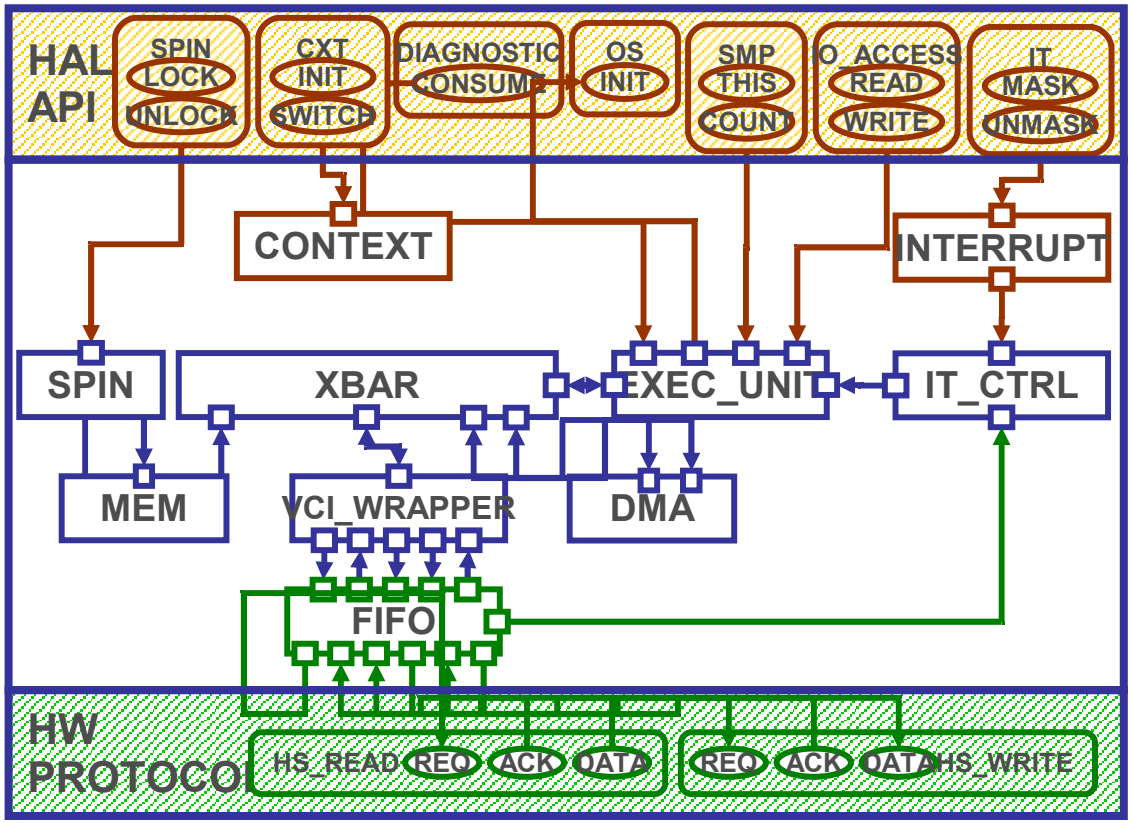
✗ Fastidious Operating System Validation

✗ Very slow simulation



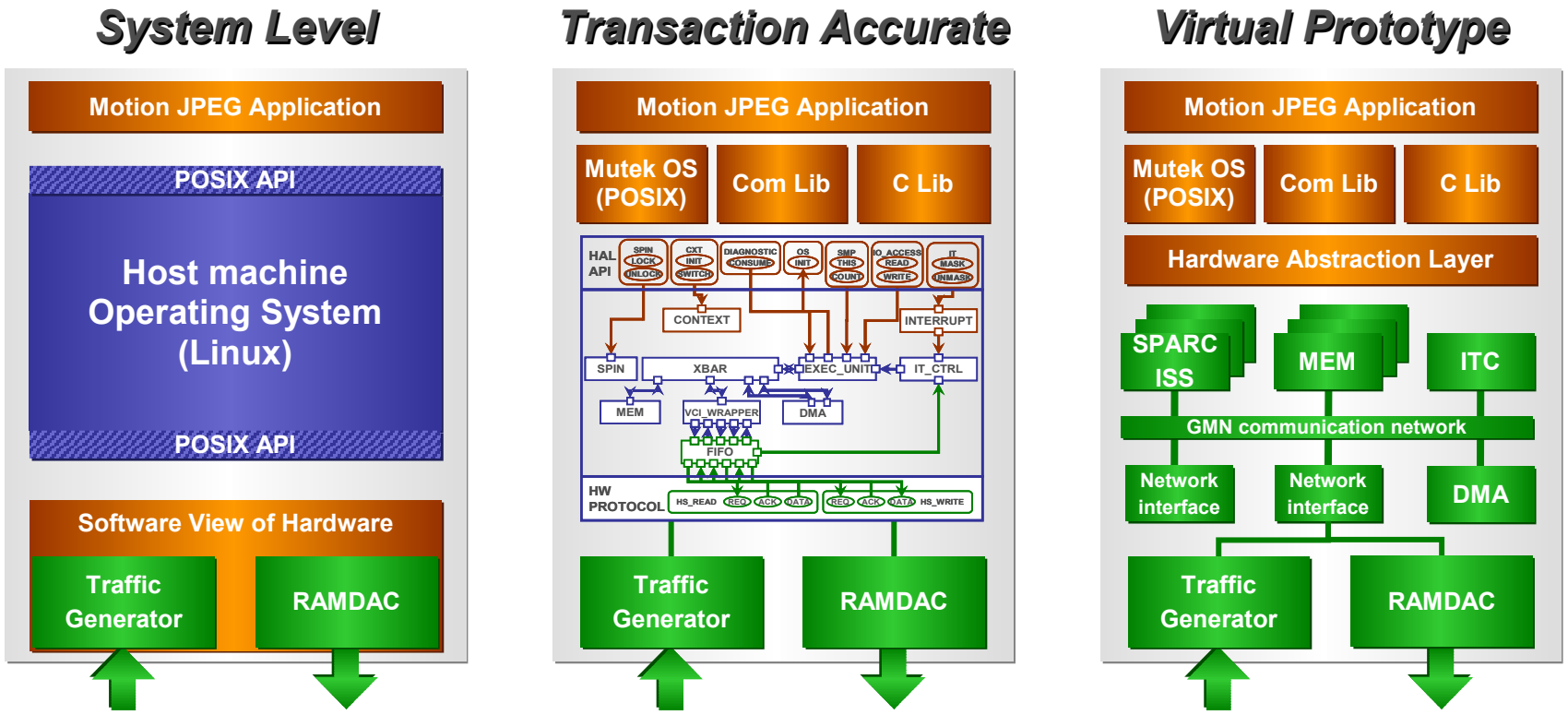
HW/SW interface model at T.A.

- Exec_Unit modeled multiprocessor parallelism
- Communication concurrency modeled by the XBAR component
- Software memory space is effectively in the model.
- Context switching is implemented by software component
- Interruptions are handled



Experiments

- 3 executable models
- Same SW application code in the 3 models
- Same OS code and libraries in TA and VP models

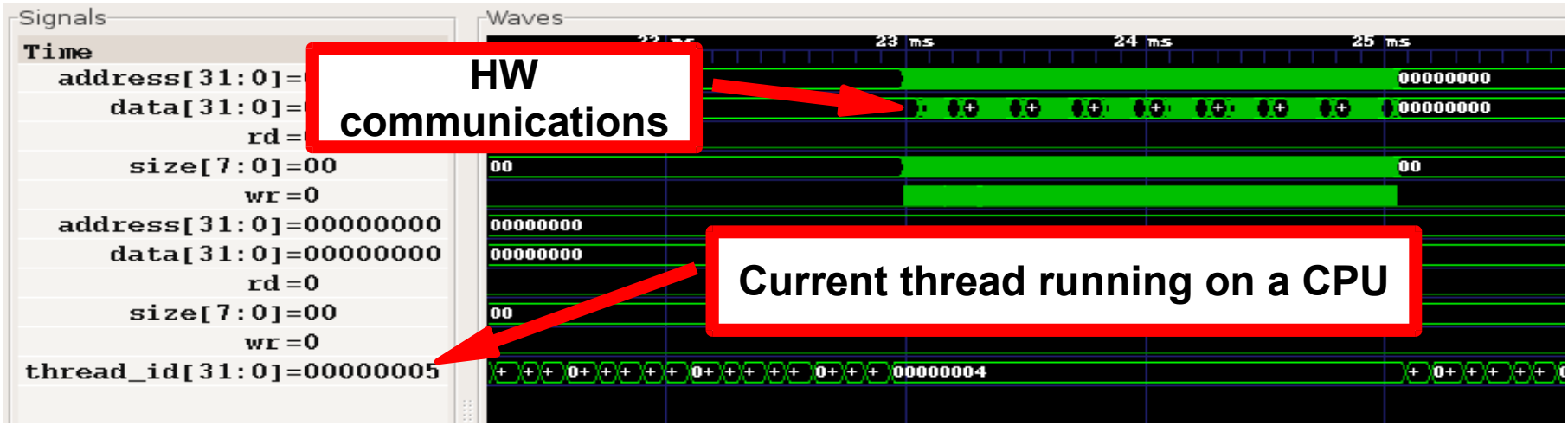


Simulation results at Transaction Accurate level

- Software point of view :
  - Give more Operating System debug capabilities thanks to hardware interaction (SpinLock, communication concurency, multiprocessor parallelism,...)
  - Allow to use standard debugger
- Hardware point of view :
  - More communication details compared to System Level and concurency
  - waveform trace

Simulation results at Transaction Accurate level

- Both point of view :
  - Hardware debugging
  - Software debug informations mixed with hardware waveforms
  - High simulation speed

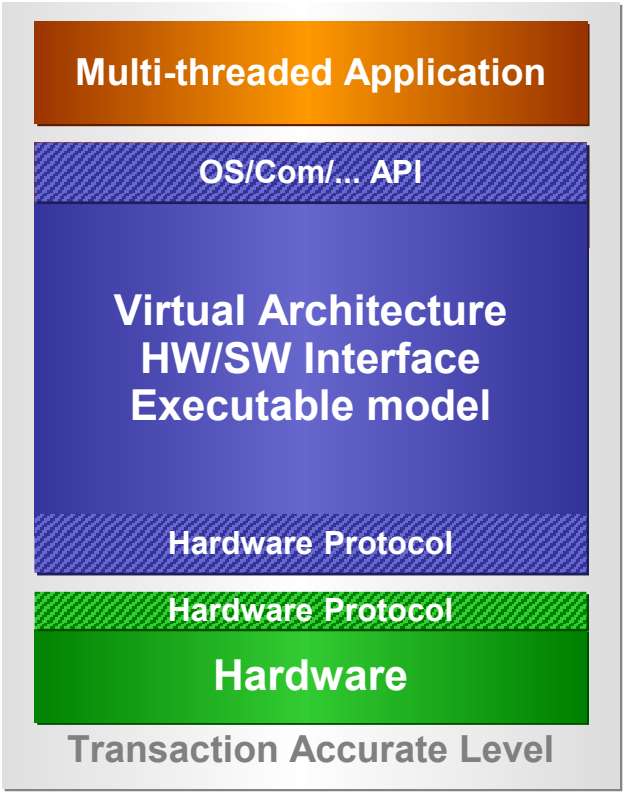


Outline

- 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
- Automatic native code annotation
  - pipeline + cache models
- 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...