Component-Based Design Environment for Multiprocessor SoC Platforms

Dr. Wander Cesário
TIMA Laboratory
46, avenue Félix Viallet
38031 Grenoble - France
http://tima.imag.fr/

Outline

- Introduction
- Virtual Component Model
- ROSES: Component-based design environment
- Case study: MPEG-4 encoder
- Summary

Motivation

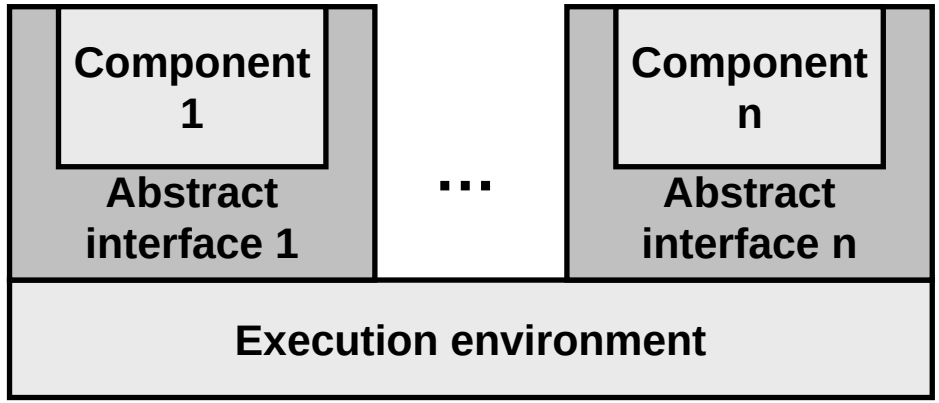
- Reuse is the enabling technology for SoC design
- Global SoC models may include software, hardware and functional components
- Component integration is a source of design bottlenecks (e.g., bridges, I/O drivers, ad-hoc co-simulation setups, etc.)
  - may be automated, the key technologies are:
    - virtual component model
    - component-based design environment

Outline

- Introduction
- Virtual Component Model
- ROSES: Component-based design environment
- Case study: MPEG-4 encoder
- Summary

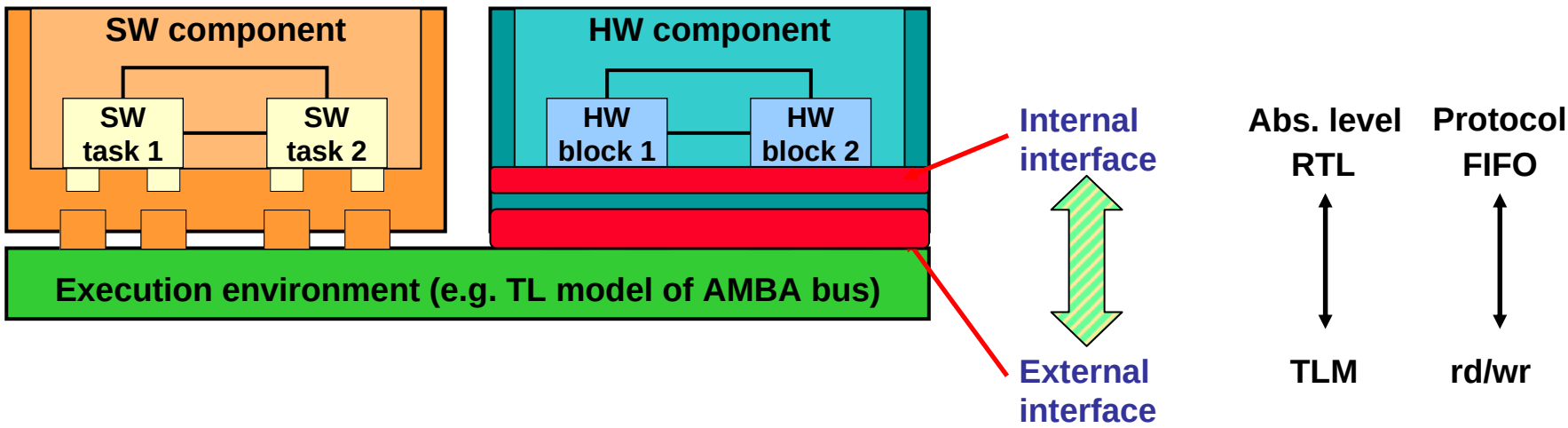
The Virtual Component Model (1/2)

- Virtual component
  - Component
    - Hardware
    - Software
    - Functional
  - Abstract interfaces
    - Required services
    - Provided services
    - Control services
    - Synchronization services
    - Parameters
- Execution environment
  - Abstract platform (e.g. NoC, cosimulation backplane, …)
  - Heterogeneous component thanks to adaptation



The Virtual Component Model (2/2)

- Basic model: a set of hierarchically interconnected virtual modules and an execution environment
- Virtual Module:
  - Content: Tasks/Instances + communication channels)
  - Abstract interface: set of virtual ports (internal, external)

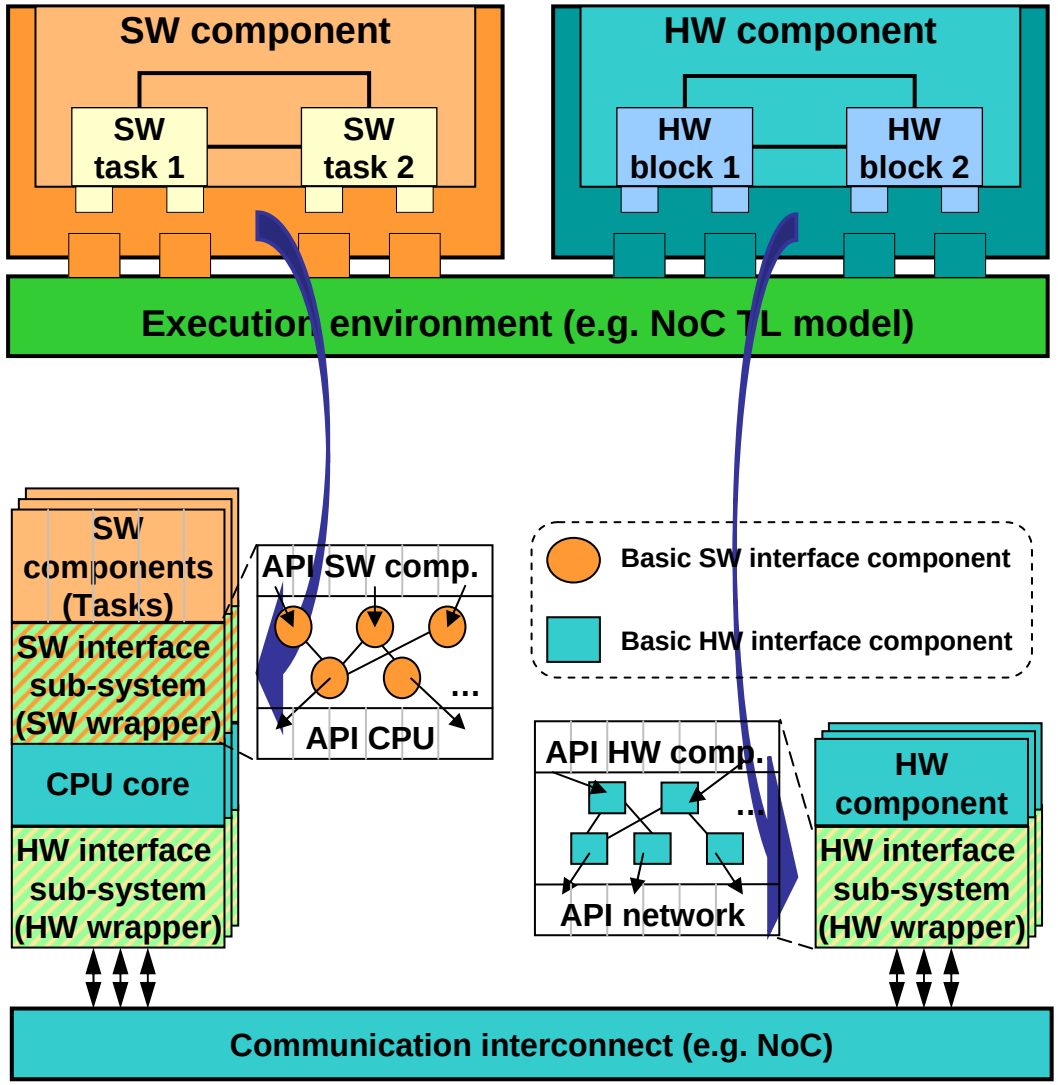


Outline

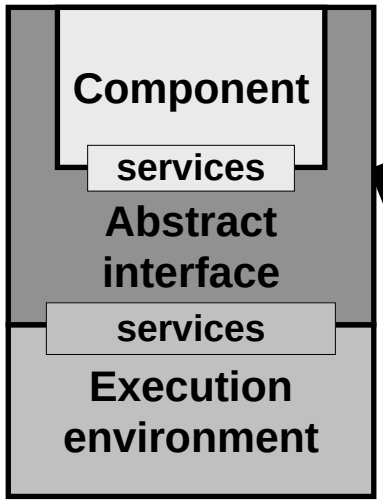
- Introduction
- Virtual Component Model
- ROSES: Component-based design environment
- Case study: MPEG-4 encoder
- Summary

System-level SoC Design Flow

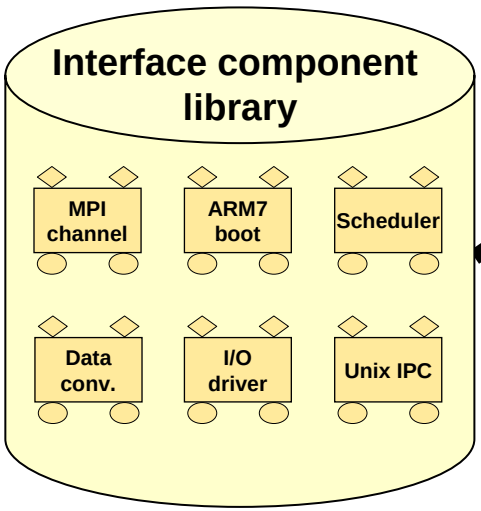
- System specification is a virtual architecture: virtual modules use abstract interfaces for HW/SW communication
- Architecture implementation: heterogeneous components and sophisticated communication interconnect linked through HW and SW interface sub-systems
- Automatic generation of application-specific HW/SW interface sub-systems from basic interface components



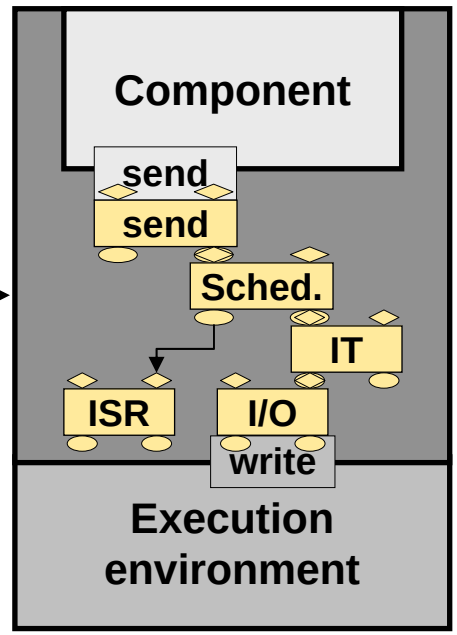
Key Technology: composing Interfaces



- Component interface
  - Required/Provided services
  - Control and Synchronization services
  - Parameters
- Interface sub-system composition
  - Services matching
  - User-extensible library
  - Code specialization

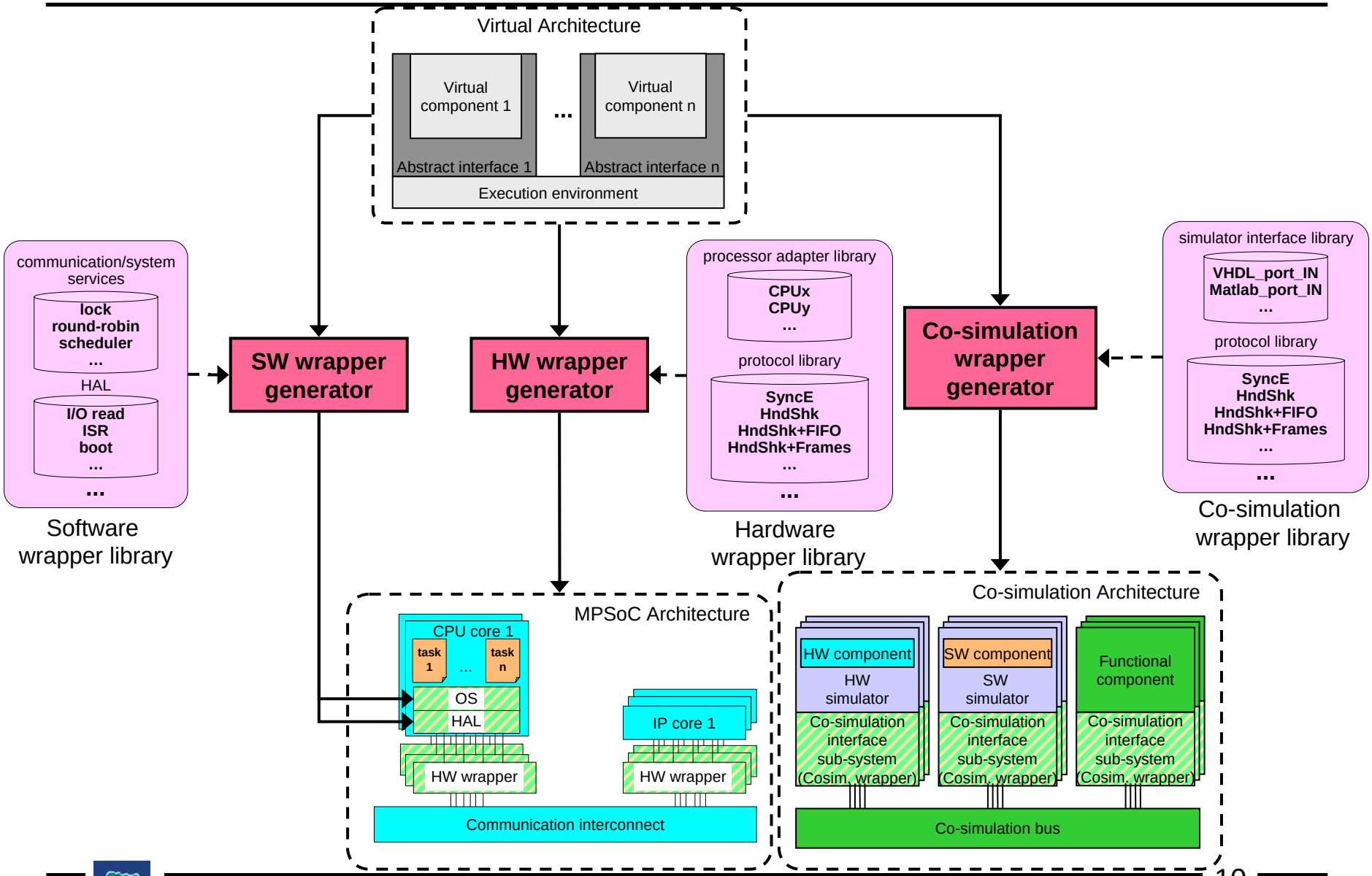


Interface sub-system composition



Works for HW, SW, and Functional interface sub-systems

ROSES: Component-based Design Environment

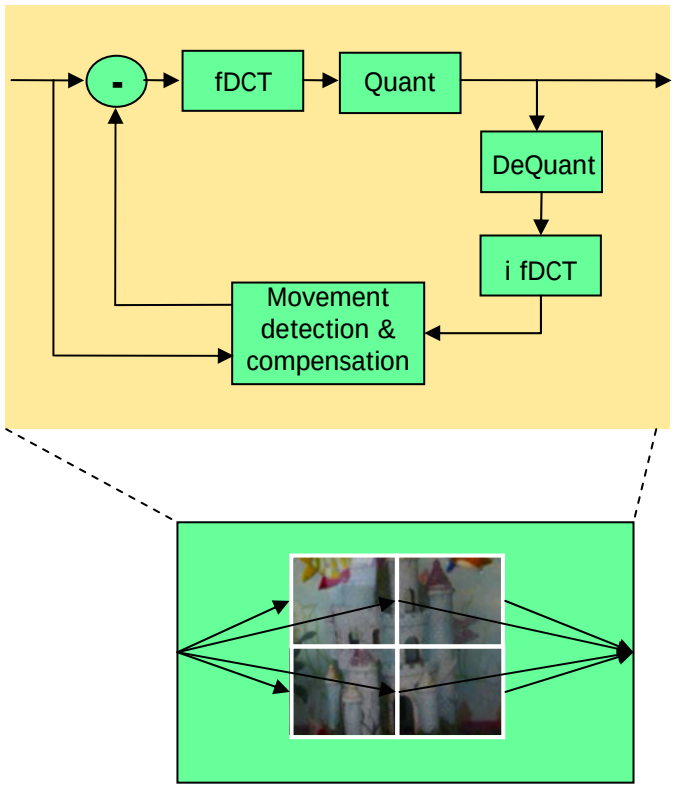


Outline

- Introduction
- Virtual Component Model
- ROSES: Component-based design environment
- Case study: MPEG-4 encoder
- Summary

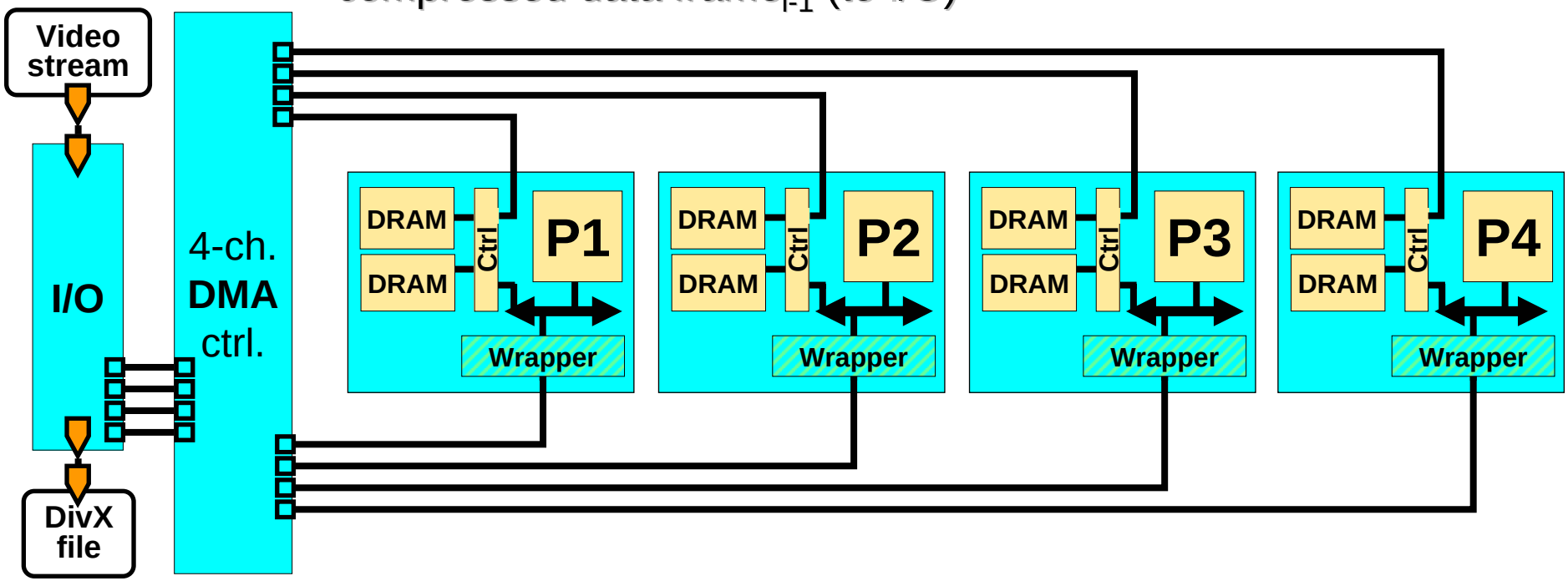
MPSoC design of a DivX encoder

- OpenDivX
  - Open source MPEG-4 encoder/decoder
  - Modified to work concurrently on 1/4th of each frame



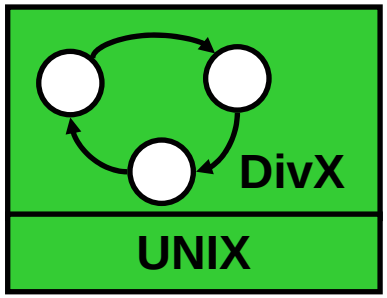
DivX encoder target architecture

- Nexperia-like 4-processors architecture
- Point-to-point communication networks
- Application-specific 4-channel DMA controller
- Use of a double banc DRAM for local memories:
  - CPU: 1/4th frame$_i$ + compressed data frame$_i$
  - DMA: 1/4th frame$_{i+1}$ (from I/O)
    + compressed data frame$_{i-1}$ (to I/O)

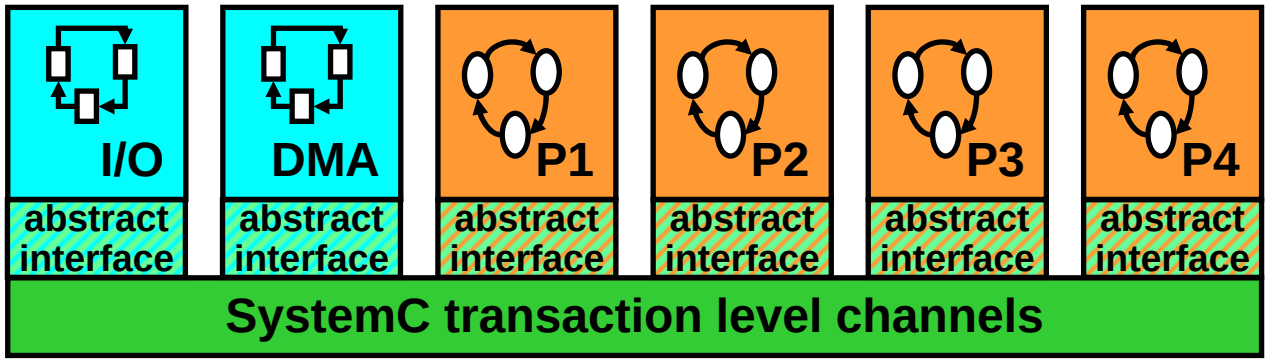


DivX main design steps

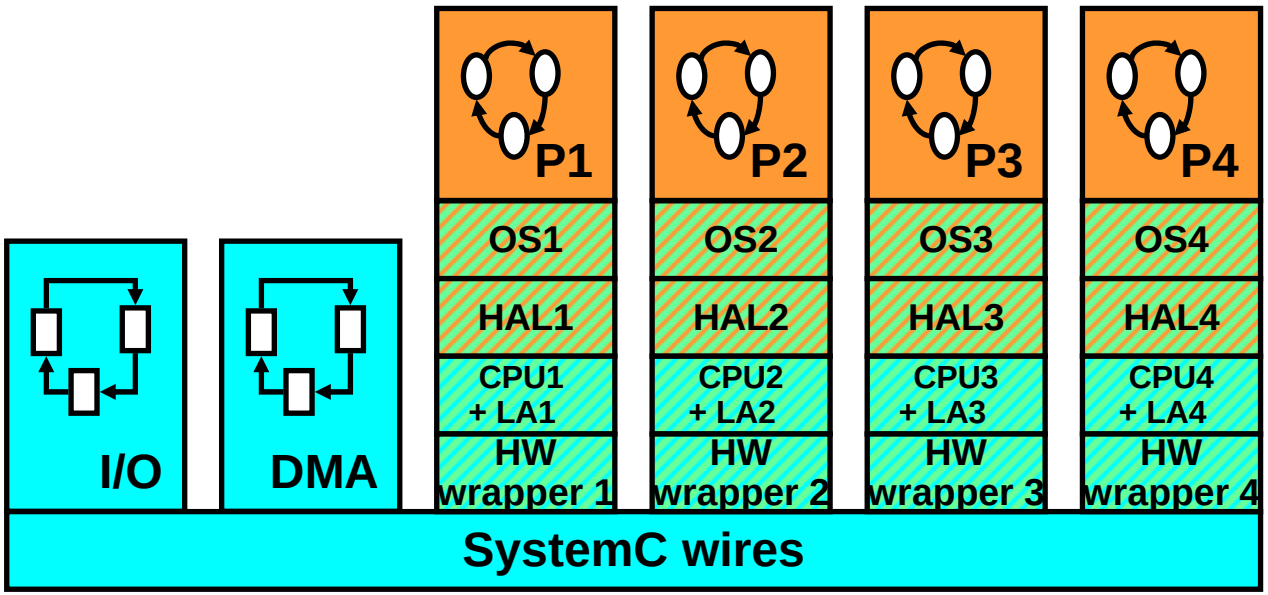
1. Algorithm



2. Transaction/RT-level mixed model in SystemC



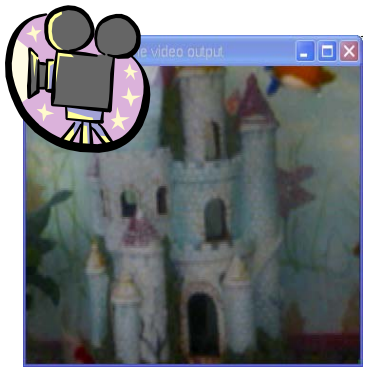
3. After refinement of HW/SW interfaces

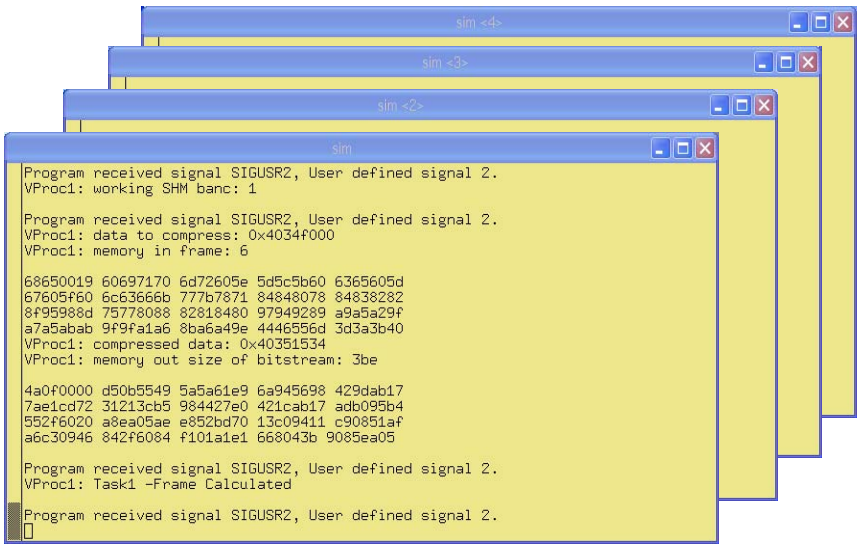


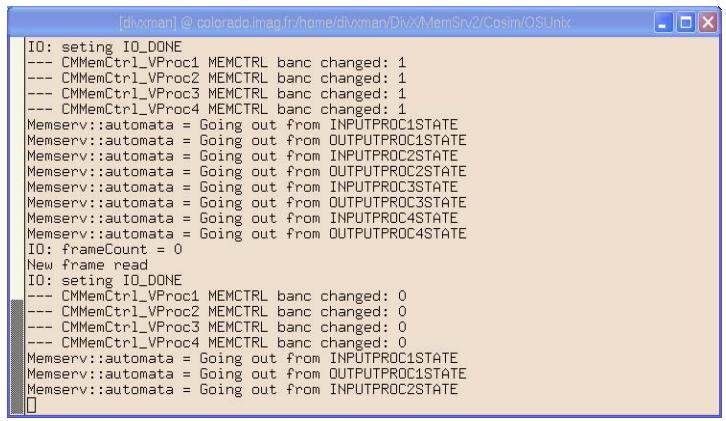
4. Implementation

- RTL design flow
- ARM integrator

DivX MPEG-4 encoder demonstrator







Summary

- Component-based design environment starting from a virtual component model specification
- Eases integration of hardware, software and functional components for global MPSoC design and validation
- Automates the integration of complex interface sub-systems by composing components from an user-extensible open-library