Complete FPGA-Based Emulation Framework for Multi-Processor Systems-on-Chip

David Atienza

(LSI / EPFL)

Laboratoire des Sistèmes Intégrés (LSI) / EPFL (Switzerland)

Computer Architecture and Automation Department (DACYA) / UCM (Spain)

Department of Electrical Engineering and Computer Science (DEIS) / Bologna (Italy)

Project Team

- From LSI/EPFL:
  - Post-Doc David Atienza
  - Prof. Giovanni De Micheli
- From DACYA/UCM:
  - 3 Master Students: Pablo García, Javier García, Esther Andrés
  - 2 PhD students: Miguel Peón, Iván Magán
  - Prof. José M. Mendías
- From DEIS/Bologna:
  - 1 PhD student: Federico Angiolini
  - Prof. Luca Benini

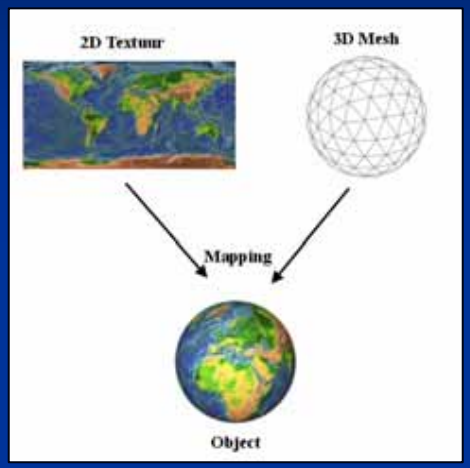
Outline

1. Introduction & Related Work
2. Emulation Framework Description
3. Case Studies & Experimental Results
4. Conclusions
5. Future Work

Outline

1. **Introduction & Related Work**
2. Emulation Framework Description
3. Case Studies & Experimental Results
4. Conclusions
5. Future Work

SoC Sw: Multimedia & Communication



Scalable video rendering

3D Virtual reality games

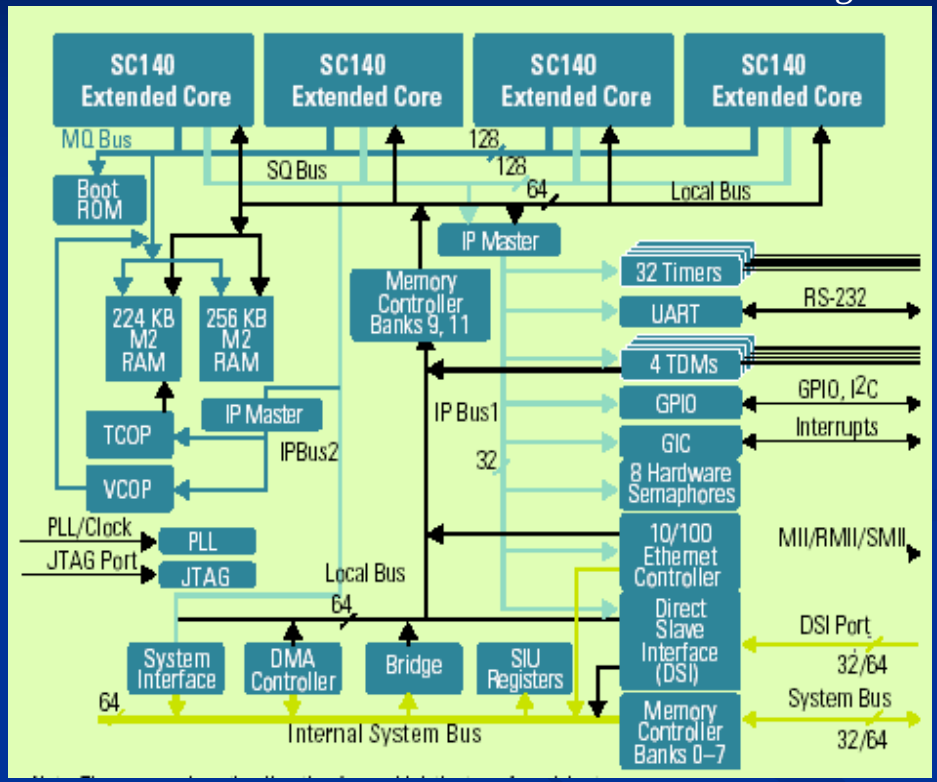
Wireless protocols

- Complex object-oriented high-level design (C++, Java)
- Very dynamic (variable use of resources for each input)
- High demands for processing (energy cost?)

SoC Hw: MPSoC

Source: Federico Angiolini

- **6 Cores**: Motorola's MSC8126
  - Four 400 MHz StarCore SC140 DSP, 16 ALUs: 5600/6400 MMACS
  - 1436 KB SRAM & multi-level memory hierarchy
  - Two internal coprocesssors (TCOP and VCOP) to provide special-purpose processing capability in parallel with the core processors
  - Complex hierarchy of buses



How to design/tune them fast and well?

Project Goals

- Fast exploration of MPSoC architectures
- Easy addition of new HW & SW components
- Configurable & cycle-accurate statistics
- Cheap HW/SW framework: standard tools & IPs
- Tests with real-life inputs (long executions)
- Not intrusive statistics: transparent to tested MPSoC

MPSoC SW Simulation

- Analytical and high-level studies (C or C++ based)
  - Fast pruning, not real applications [P. Mishra et al., J. Braun et al., R. Gupta et al. …]
- Transaction-Level Modelling (SystemC, 100-200 KHz)
  - To evaluate latency & throughtput [J. Madsen et al, F. Angiolini et al, P.G. Paulin, W. Hang-Sheng et al, S. Kolson et al...]
  - In industry [ARM PrimeXsys, CoWare LisaTek…]
- Cycle-Accurate (HDL, SystemC, ~100 KHz):
  - Industry [Synopsys Realview, Mentor Graphics Primecell]
  - Explore energy-performance trade-offs [MPARM]

At the desired cycle-accurate level, they are too slow for real-life applications!

MPSoC HW Prototyping

- Industry, large systems (few MHz, very expensive):
  - Cadence Palladium II (256M gates, $1M, 1.6 MHz)
- Industry, smaller systems, but proprietary cores (~50-150 MHz):
  - ARM Integrator IP (ARM Cores, AMBA buses)
  - Heron Engineering (Few cores and interconnections, plug-and-play)
- Industry, multi-FPGA (fixed protocols, few MHz):
  - Aptix System Explore, Verification Engineering (Zebu XL and ZV)
- Academia (or mixed with industry):
  - T4SoC, fixed cores & NIs exploration [STMicroelect. & A. Jerraya's group]
  - To validate NIs & NoCs [T. Marescaux et al, C. Zeferino et al]

Very expensive and lots of man-power required to perform MPSoC exploration!

Emulation vs Simulation / Protot.

- "To emulate an electronic system is to build a platform capable of imitating its behaviour in an accurate and analyzable way." Lenoski, Gupta, Henessy et al. '92

SW Simulation	HW Emulation	Prototyping
•Lower performance •Maximum flexibility	•Great performance •Good flexibility •Independent from actual HW	•Close to final performance •Reduced flexibility •Dependent on available HW (May alter results)

- More efficient than SW simulation (HW efficiency in signal management)
- Conclusions can be drawn earlier in development than HW prototyping

Outline

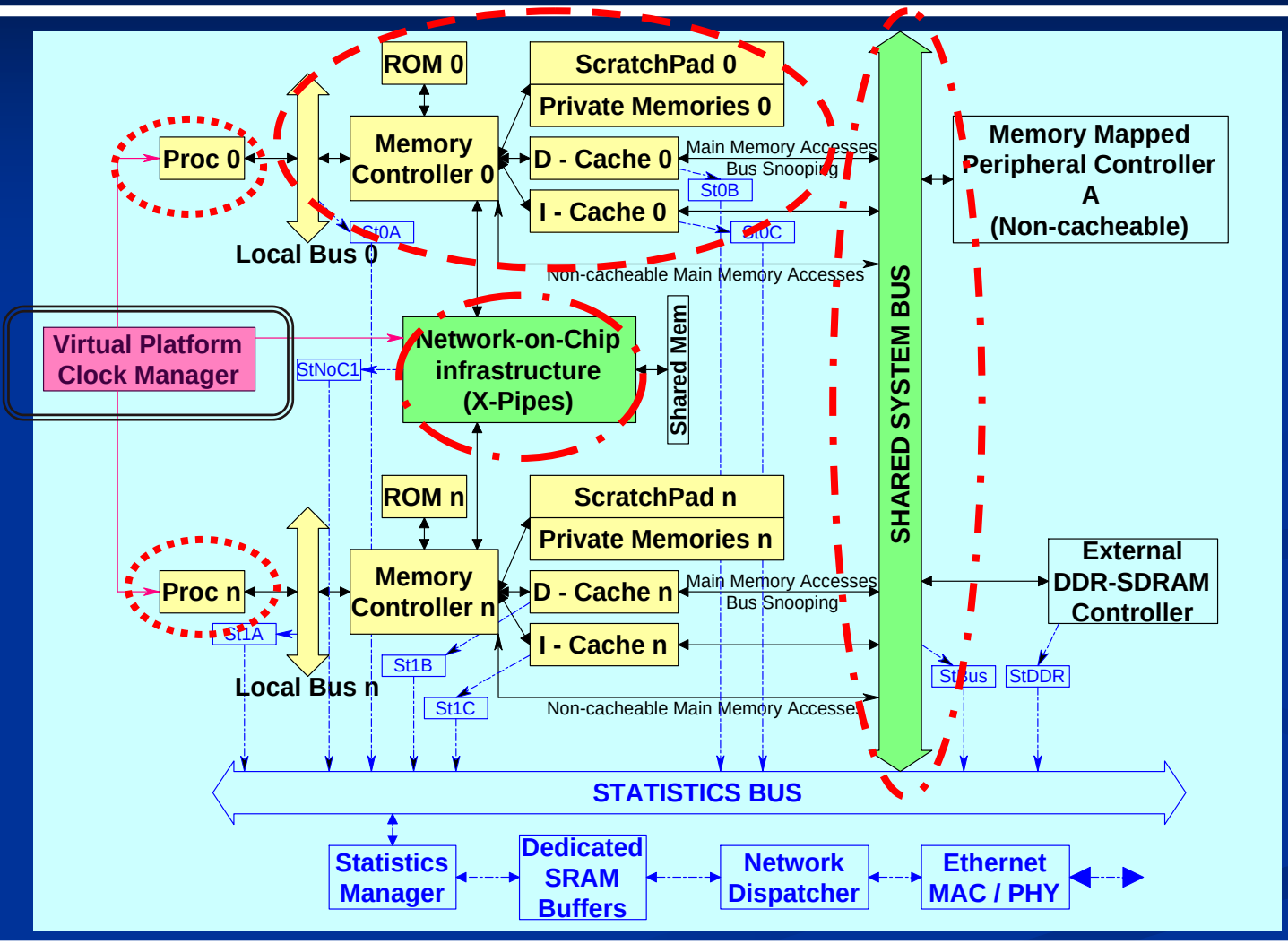
1. Motivation & Related Work
2. **Emulation Framework Description**
3. Case Studies & Experimental Results
4. Conclusions
5. Future Work

Emulated MPSoC Features

- Exploration of 3 MPSoC HW architectural levels:
  1. Scalable number of processors (softcores and hardcores)
  2. Interconnections
     - NoC topologies
     - Real-life and configurable buses (latencies, arbitration, etc.)
  3. Complex memory hierarchies
     - Shared main memories: SRAM, SDRAM, DDR, …
     - I- and D-caches, scratchpads
     - Private SRAM memories
  - Additional peripherals:
    - Custom-made IPs (e.g. VHDL module for data/instr. prefetching)
- SW running coming from real-life programming lenguages
  - C, C++ or Java

MPSoC Emulation Architecture

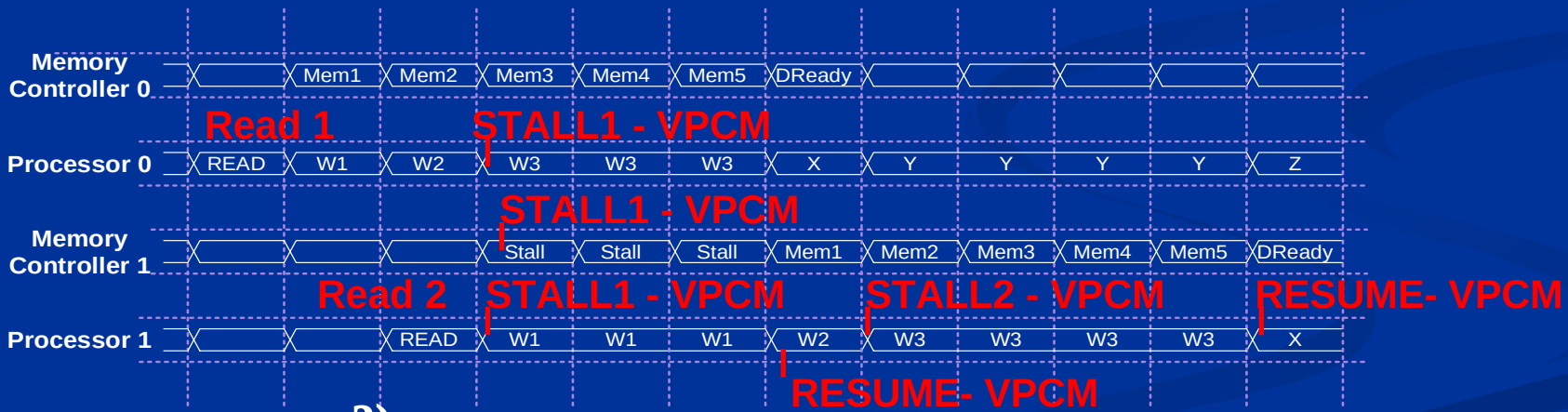
Additional Element



Virtual Platform Clock Manager (VPCM)

1) Concurrent reads without collision in memory hierarchy

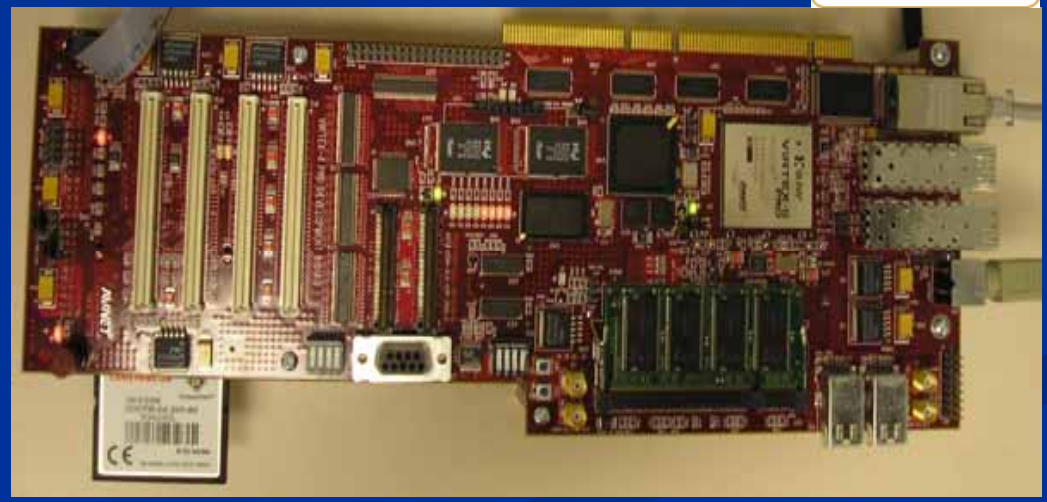
- Desired latency: 3 cycles
- Real latency: 5 cycles



2) Concurrent reads wich collide in memory hierarchy

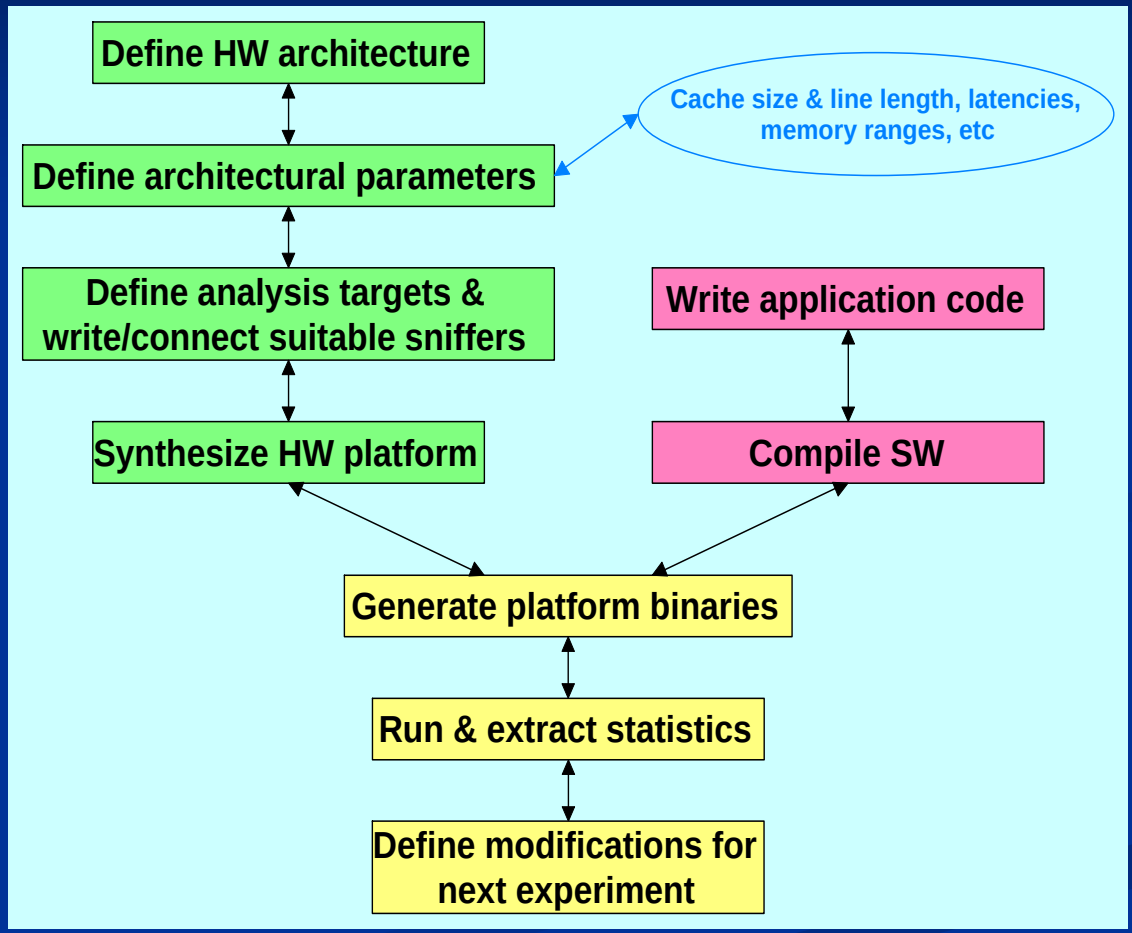
Standard Hw Components & Tools

- **Hardware:** ($1800)
  - Virtex-II Pro XC2VP30 AVNET (2 Power PCs and softcore Microblaze)
  - Micron Mobile SDRAM 32 MB, DDR 128 MB, Cypress SRAM 2 MB, Intel StrataFlash 16 MB, Compact FLASH card
  - Ethernet connection 10/100/1000 MBit/s Ethernet , PCI connector
- Software: (All donated, but buying it: ~$4000)
  - Xilinx EDK v6.3
  - Xilinx ISE v6.3
  - Mentor ModelSim v7.0
  - Xilinx ChipScope v6.3
  - Xilinx CoreGen v6.3

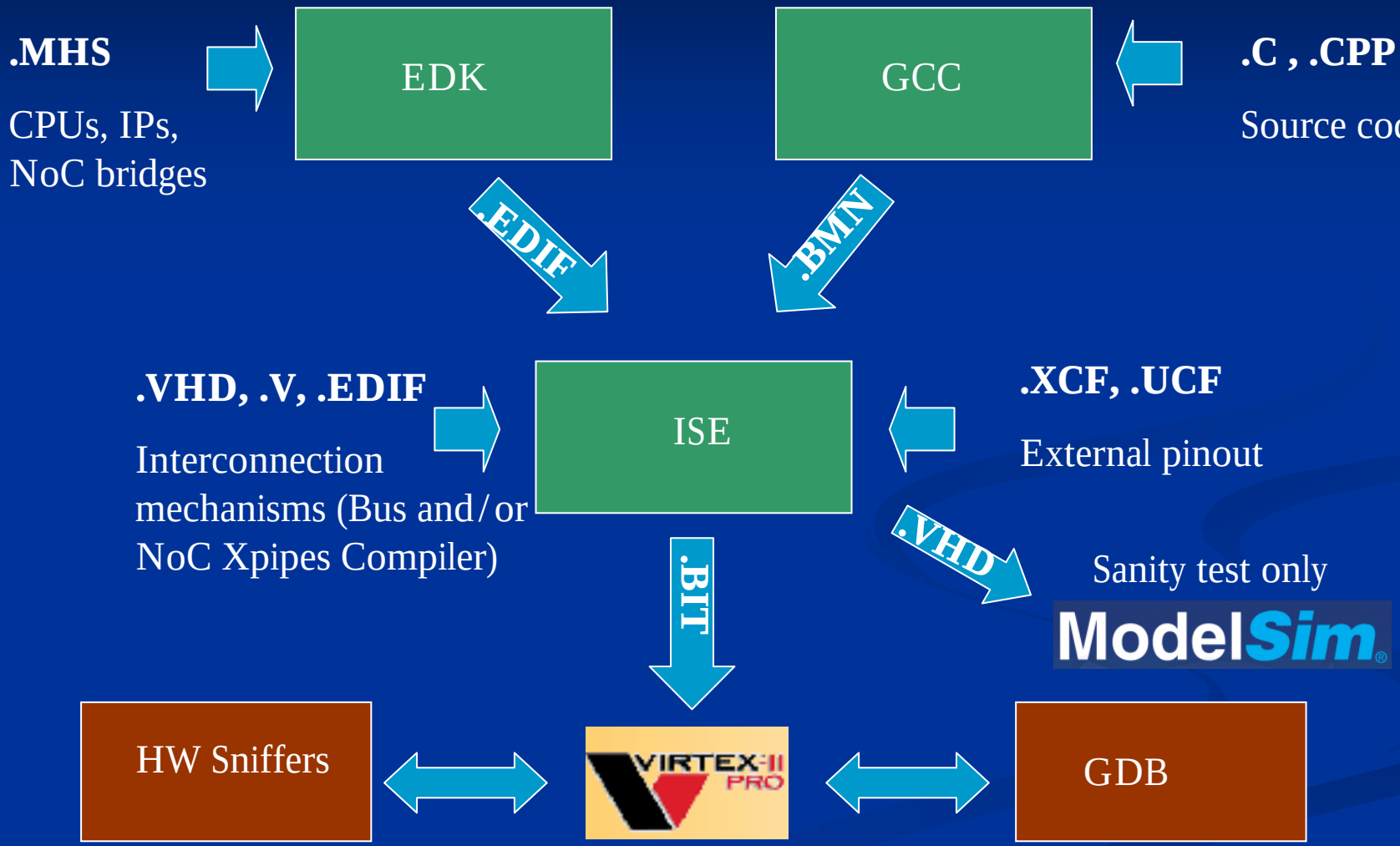


Proposed Emulation Toolflow

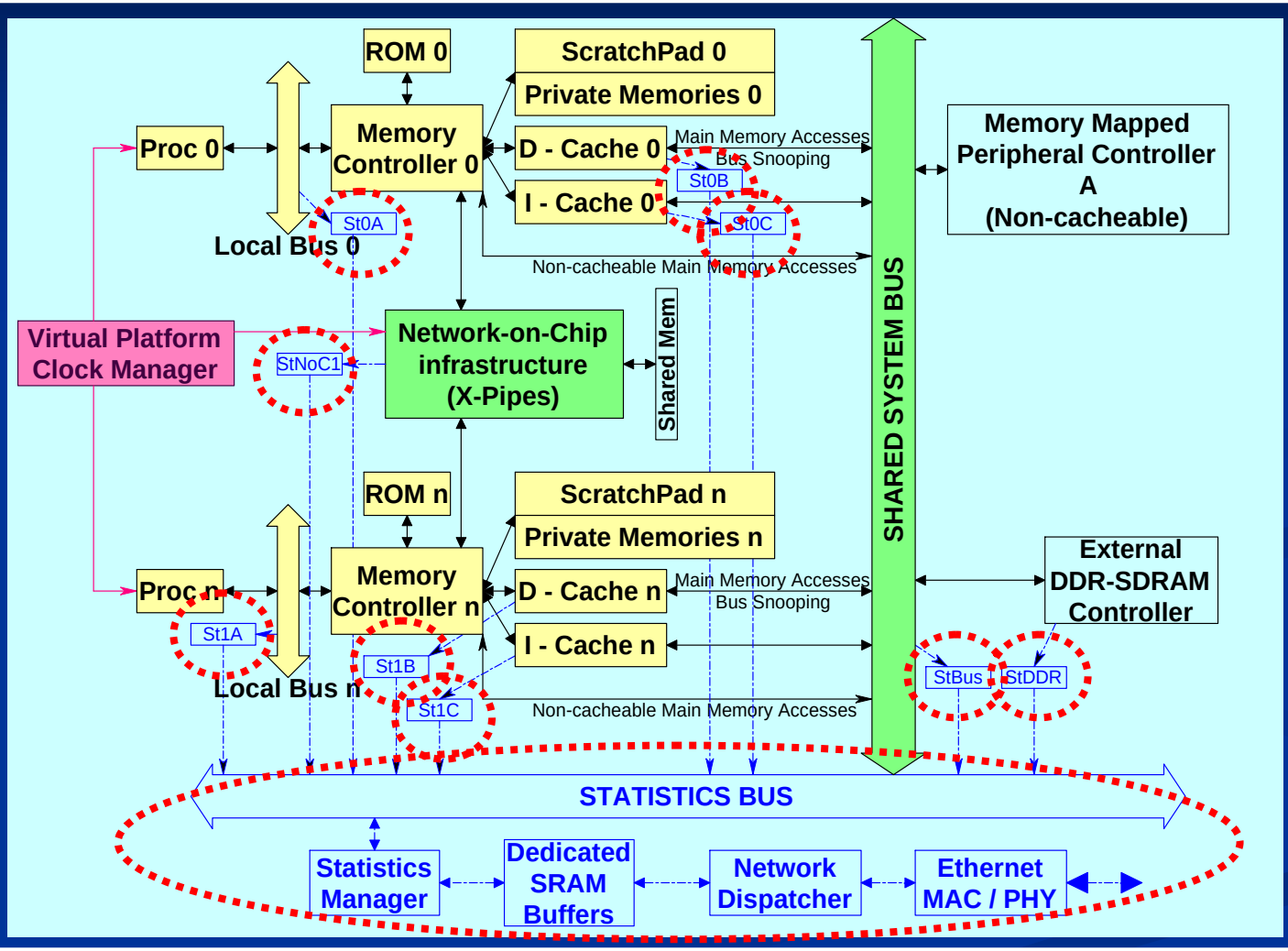
- Double integration of HW & SW flows in one overall framework.
- Minimal interaction:
  - Modifications in one branch do not tipically affect the other.
- Just standard tools



Realization of HW/SW Toolflow

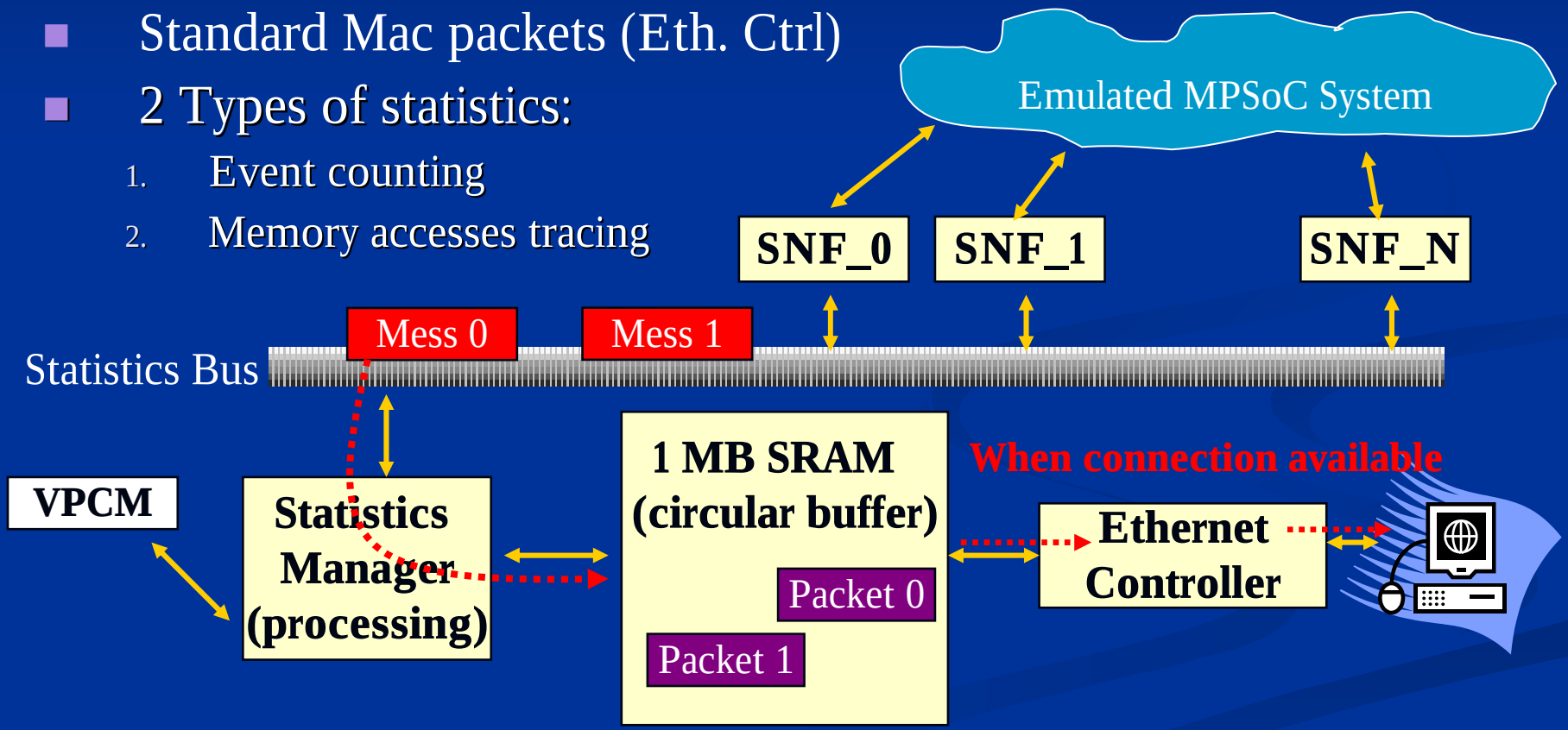


Statistics Extraction Subsystem



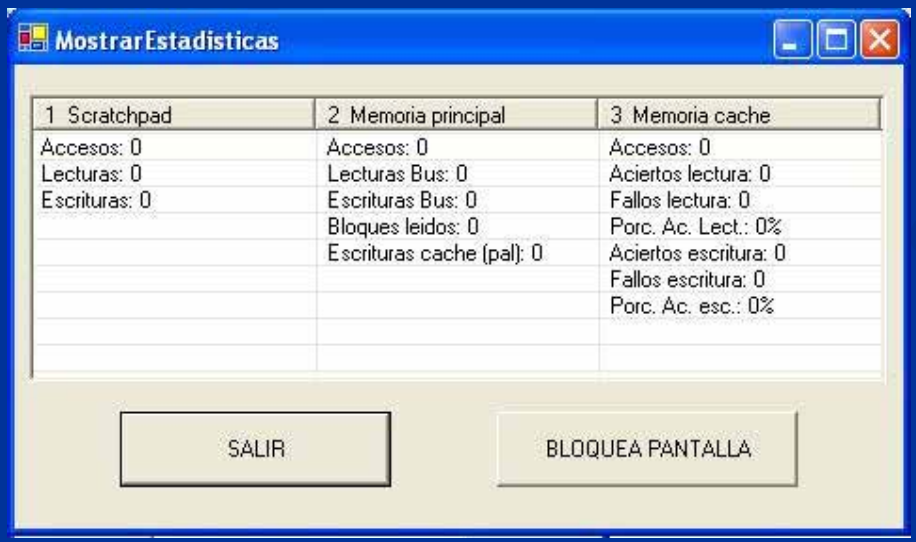
Statistics Extraction Subsystem: HW Part

- Dedicated bus & HW: Not intrusive, concurrently running
- Another clock domain (150 MHz)
- Standard Mac packets (Eth. Ctrl)
- 2 Types of statistics:
  1. Event counting
  2. Memory accesses tracing



Statistics Extraction Subsystem: SW Part

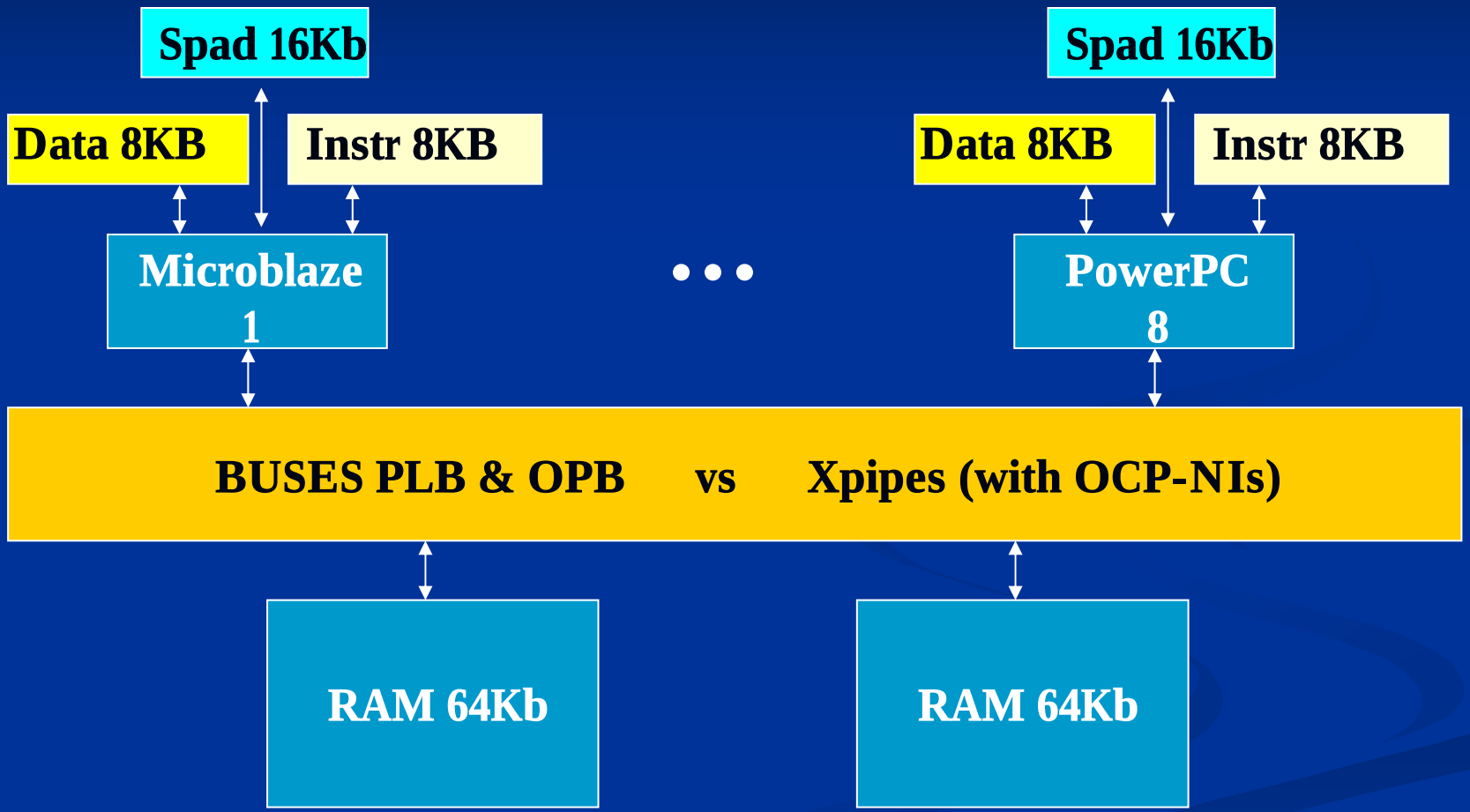
- MPSoC emulation platform sends data through Ethernet connection
- Graphical tool in host pc displays real-time information
- Logs the data for post-emulation analyses



Outline

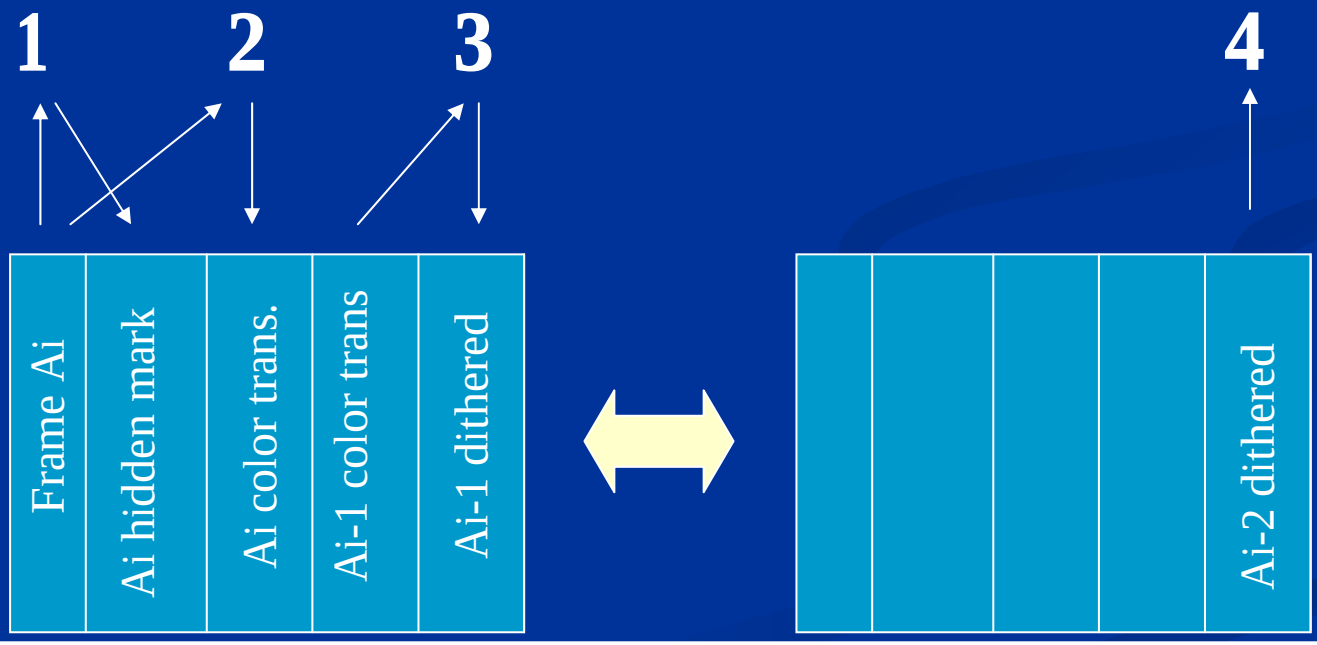
1. Motivation & Related Work
2. Emulation Framework Description
3. **Case Studies & Experimental Results**
4. Conclusions
5. Future Work

MPSoC HW Component



MPSoC SW Component

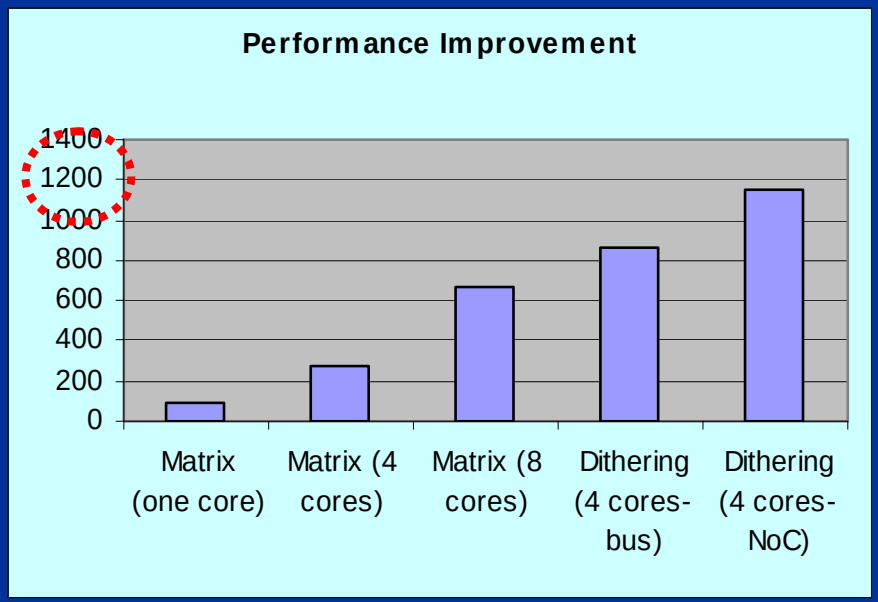
- Case 1: Matrix multiplication
- Case 2: Multimedia pipeline with multiple buffering (synchro: semaphores)
  - Steganography techniques (i.e. watermark checking)
  - Color space transformation
  - Dithering
  - Visualization



Results 1: Performance Comparisons

- HW emulation does not lag with large number of processing elements (100 MHz) or signals managed (NoC).

Speed-ups of 3 orders of magnitude with cycle-accurate simulators!



	MPARM	HW Emulator
Matrix (one core)	106 sec	1.2 sec (88x)
Matrix (4 cores)	5' 23 sec	1.2 sec (269x)
Matrix (8 cores)	13' 17 sec	1.2 sec (664x)
Multimedia pipeline (4 cores-bus)	2' 35 sec	0.18 sec (861x)
Multimedia pipeline (4 cores-NoC)	3' 15 sec	0.17 sec (1147x)

Results 2: Accuracy & Flexibility of statistics

- Similar kind of statistics than obtained with MPARM and other SW-based simulators.
- Same cycle-accuracy, e.g. data allocated in scratchpad & DMA:

Metric	Matrix (1 core)	Matrix (4 core)	Matrix (8 cores)
Processor cycles	$1.3x10^7$	$5.2x10^7$	$1.1x10^8$
Memory reads	0	0	0
Memory block-reads	$32x10^3$	$11x10^4$	$24x10^4$
Memory writes	$256x10^3$	$101x10^4$	$271x10^4$
Scratchpad reads	$4,096x10^3$	$1,23x10^4$	$3,132x10^4$
Scratchpad writes	0	0	0
Cache read hits	$4,064x10^3$	$1,697x10^4$	$3,214x10^4$
Cache read misses	$32x10^3$	$12x10^4$	$25x10^4$

Outline

1. Introduction
2. Related Work
3. Emulation Framework Description
4. Case Studies & Experimental Results
5. **Conclusions**
6. Future Work

Conclusions

- First Version FPGA-Based MPSoC Emulation Platform
  - Real-time, it runs real applications (C or C++)
  - Use of standard IPs & tools (not expensive or time-consuming)
  - Architectural MPSoC exploration
- Speed-ups of 3 orders of magnitude with simulators
  - HW emulation scales, SW simulation does not
- Cycle-accurate and flexible statistics
  - Plug-and-play use of HW sniffers
  - Easily extensible as in SW simulators, simple interface

Outline

1. Introduction
2. Related Work
3. Emulation Framework Description
4. Case Studies & Experimental Results
5. Conclusions
6. **Future Work**

Future Work

- Ethernet interface inefficient for huge amounts of data (Gigabit interface?)
- Statistics from inside processing elements and core architecture exploration: Integration of URLAP.
- Porting O.S. to selected processor (e.g. uCLinux).
- Extensions of memory controller for other memories (DDR interface not fully done).
- External pinout limitations, multi-FPGA protocols
- Virtual file system on the Compact Flash memory.

QUESTIONS ?