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

David Atienza
(LSI/EPFL)

Laboratoire des Systèmes 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

- **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 coprocessors (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
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)
- 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

<table>
<thead>
<tr>
<th>SW Simulation</th>
<th>HW Emulation</th>
<th>Prototyping</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Lower performance</td>
<td>• Great performance</td>
<td>• Close to final performance</td>
</tr>
<tr>
<td>• Maximum flexibility</td>
<td>• Good flexibility</td>
<td>• Reduced flexibility</td>
</tr>
<tr>
<td></td>
<td>• Independent from actual HW</td>
<td>• Dependent on available HW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(May alter results)</td>
</tr>
</tbody>
</table>

- 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 languages
  - C, C++ or Java
MPSoC Emulation Architecture

Additional Element

Virtual Platform
Clock Manager

Network-on-Chip infrastructure (X-Pipes)

Shared Mem

Statistics BUS

Statistics Manager

Dedicated SRAM Buffers

Network Dispatcher

Ethernet MAC / PHY

Proc 0

Local Bus 0

Memory Controller 0

ROM 0

ScratchPad 0

Private Memories 0

D - Cache 0

I - Cache 0

Shared Mem

Proc n

Local Bus n

Memory Controller n

ROM n

ScratchPad n

Private Memories n

D - Cache n

I - Cache n

Shared Mem

External DDR-SDRAM Controller

Memory Mapped Peripheral Controller A (Non-cacheable)

Non-cacheable Main Memory Accesses

SHARED SYSTEM BUS

St0A

St0B

St0C

StDDR

Stat0

Stat1

Stat2

Stat3

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn

Statn
<table>
<thead>
<tr>
<th>Memory Controller 0</th>
<th>Mem1</th>
<th>Mem2</th>
<th>Mem3</th>
<th>Mem4</th>
<th>Mem5</th>
<th>DReady</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Processor 0</td>
<td>READ</td>
<td>W1</td>
<td>W2</td>
<td>W3</td>
<td>W3</td>
<td></td>
</tr>
<tr>
<td>Memory Controller 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Processor 1</td>
<td>READ</td>
<td>W1</td>
<td>W1</td>
<td>W1</td>
<td>W2</td>
<td>W3</td>
</tr>
</tbody>
</table>

1) Concurrent reads without collision in memory hierarchy

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

<table>
<thead>
<tr>
<th>Memory Controller 0</th>
<th>Mem1</th>
<th>Mem2</th>
<th>Mem3</th>
<th>Mem4</th>
<th>Mem5</th>
<th>DReady</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Processor 0</td>
<td>READ</td>
<td>W1</td>
<td>W2</td>
<td>W3</td>
<td>W3</td>
<td>X</td>
</tr>
<tr>
<td>Memory Controller 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Processor 1</td>
<td>READ</td>
<td>W1</td>
<td>W1</td>
<td>W1</td>
<td>W2</td>
<td>W3</td>
</tr>
</tbody>
</table>

2) Concurrent reads with collision in memory hierarchy

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

David Atienza Alonso  LSI/ EPFL & DACYA/ UCM

Workshop SI/ EPFL 07-10-2005
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

Diagram:

1. Define HW architecture
2. Define architectural parameters
3. Synthesize HW platform
4. Define analysis targets & write/connect suitable snifferers
5. Generate platform binaries
6. Run & extract statistics
7. Define modifications for next experiment
8. Compile SW

Cache size & line length, latencies, memory ranges, etc

Define HW architecture

Define architectural parameters

Synthesize HW platform

Define analysis targets & write/connect suitable snifferers

Generate platform binaries

Run & extract statistics

Define modifications for next experiment

Compile SW
Realization of HW/SW Toolflow

- .MHS
  - CPUs, IPs, NoC bridges
  - CPUs, IPs, NoC bridges

- .VHD, .V, .EDIF
  - Interconnection mechanisms (Bus and/or NoC Xpipes Compiler)
  - Interconnection mechanisms (Bus and/or NoC Xpipes Compiler)

- .EDIF
  - Source code
  - Source code

- .BMN
  - External pinout
  - External pinout

- .XCF, .UCF
  - Sanity test only
  - Sanity test only

- .BIT
  - ModelSim
  - ModelSim

- EDK
  - GCC
  - ISE

- HW Sniffers
  - GDB

- VIRTEX II PRO
  - VIRTEX II PRO

David Atienza Alonso • LSI/EPFL & DACYA/UCM
Workshop SI/EPFL 07-10-2005
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

Types of statistics:
1. Event counting
2. Memory accesses tracing

Emulated MPSoC System

1 MB SRAM (circular buffer)

When connection available

Statistics Extraction Subsystem

VPCM

Statistics Manager (processing)

Packet 0
Packet 1
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

```plaintext
<table>
<thead>
<tr>
<th>1 Scratchpad</th>
<th>2 Memoria principal</th>
<th>3 Memoria cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access: 0</td>
<td>Access: 0</td>
<td>Access: 0</td>
</tr>
<tr>
<td>Lecturas: 0</td>
<td>Lecturas Bus: 0</td>
<td>Lecturas lectura: 0</td>
</tr>
<tr>
<td>Escrituras: 0</td>
<td>Escrituras Bus: 0</td>
<td>Fallos lectura: 0</td>
</tr>
<tr>
<td></td>
<td>Bloques leídos: 0</td>
<td>Poc. Ac. Lect.: 0%</td>
</tr>
<tr>
<td></td>
<td>Escrituras cache (sal): 0</td>
<td>Accesos escrituras: 0</td>
</tr>
</tbody>
</table>
```

Graphical tool in host pc displays real-time information

```plaintext
Logs the data for post-emulation analyses
```

```
Form1

<table>
<thead>
<tr>
<th>Identificador</th>
<th>Número de bits de dirección</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Tipo de sniffer

- Memoria cache
- Scratchpad
- Memoria principal

2 bits

2 Memoria principal 4 bits

Iniciar

Iniciar
```

Outline

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

Spad 16Kb

Data 8KB

Instr 8KB

Microblaze

1

Spad 16Kb

Data 8KB

Instr 8KB

PowerPC

8

BUSES PLB & OPB vs Xpipes (with OCP-NIs)

RAM 64Kb

RAM 64Kb

David Atienza Alonso • LSI/ EPFL & DACYA/ UCM
Workshop SI/ EPFL 07-10-2005
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

1. Frame Ai
2. Ai hidden mark
3. Ai color trans.
4. Ai-1 color trans.
5. Ai-1 dithered
6. Ai-2 dithered

David Atienza Alonso • LSI/ EPFL & DACYA/ UCM
Workshop SI/ EPFL 07-10-2005
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!

<table>
<thead>
<tr>
<th></th>
<th>MPARM</th>
<th>HW Emulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix (one core)</td>
<td>106 sec</td>
<td>1.2 sec (88x)</td>
</tr>
<tr>
<td>Matrix (4 cores)</td>
<td>5’ 23 sec</td>
<td>1.2 sec (269x)</td>
</tr>
<tr>
<td>Matrix (8 cores)</td>
<td>13’ 17 sec</td>
<td>1.2 sec (664x)</td>
</tr>
<tr>
<td>Multimedia pipeline (4 cores-bus)</td>
<td>2’ 35 sec</td>
<td>0.18 sec (861x)</td>
</tr>
<tr>
<td>Multimedia pipeline (4 cores-NoC)</td>
<td>3’ 15 sec</td>
<td>0.17 sec (1147x)</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Matrix (1 core)</th>
<th>Matrix (4 core)</th>
<th>Matrix (8 cores)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor cycles</td>
<td>$1.3 \times 10^7$</td>
<td>$5.2 \times 10^7$</td>
<td>$1.1 \times 10^8$</td>
</tr>
<tr>
<td>Memory reads</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Memory block reads</td>
<td>$32 \times 10^3$</td>
<td>$11 \times 10^4$</td>
<td>$24 \times 10^4$</td>
</tr>
<tr>
<td>Memory writes</td>
<td>$256 \times 10^3$</td>
<td>$101 \times 10^4$</td>
<td>$271 \times 10^4$</td>
</tr>
<tr>
<td>Scratchpad reads</td>
<td>$4.096 \times 10^3$</td>
<td>$1.23 \times 10^4$</td>
<td>$3.132 \times 10^4$</td>
</tr>
<tr>
<td>Scratchpad writes</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cache read hits</td>
<td>$4.064 \times 10^3$</td>
<td>$1.697 \times 10^4$</td>
<td>$3.214 \times 10^4$</td>
</tr>
<tr>
<td>Cache read misses</td>
<td>$32 \times 10^3$</td>
<td>$12 \times 10^4$</td>
<td>$25 \times 10^4$</td>
</tr>
</tbody>
</table>
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?