Application specific Multi-Processor SoC

System-Level Synthesis Group

TIMA Laboratory

46 Avenue Felix Viallet

38031 Grenoble Cedex France

Tel: +33 476 57 47 59

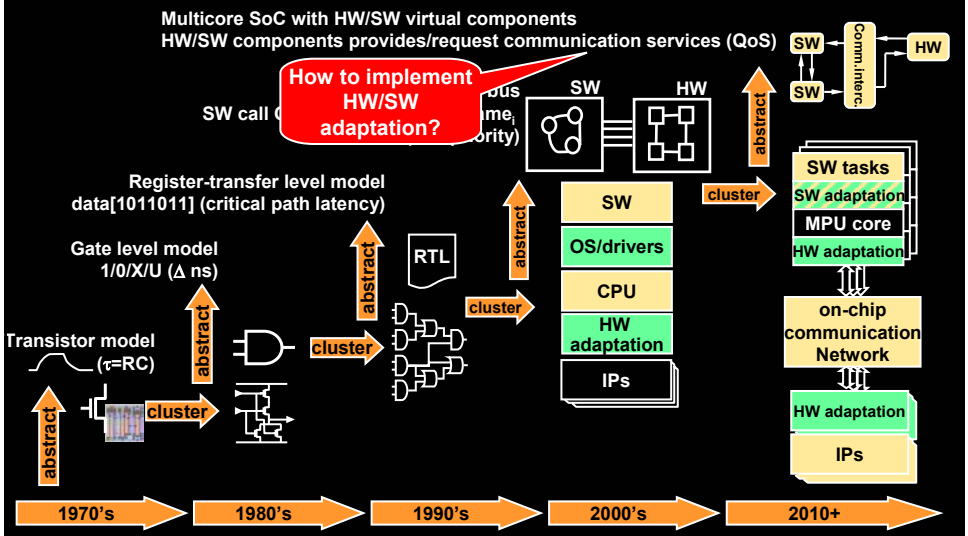
Fax: +33 476 47 38 14

Email: Ahmed.Jerraya@imag.fr

Ahmed Jerraya

Multicore System-on-Chip Components

The next level of abstraction in the architecture/communication space



Outline

1.- Network and System on Chip Design

2.- HW-SW Communication bottleneck

3.- Roses: Component Based Design

4. A VDSL design example

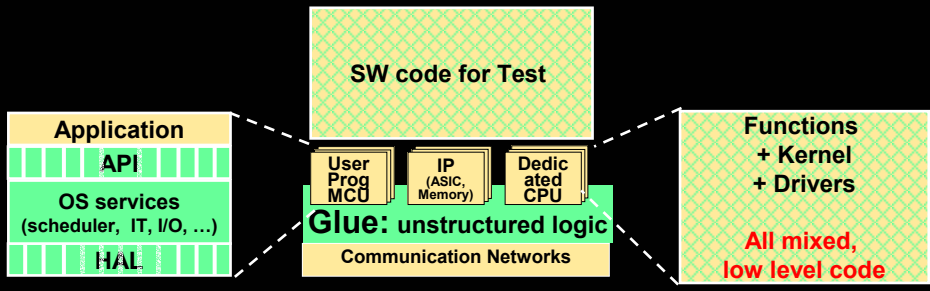
5.- Status and conclusions

Networks and System on Chip

- SoC: put on a chip what we used to put on boards, 2004:
  - ITRS, 70% of ASICS will be SoC
  - 10 000 SoC designs, 32 Bn$ Market
- Multi-processor:
  - More than one instruction set processor on chip
  - Complex HW/SW communication
- NOC: Network on Chip:
  - Multiple HW-SW Cores for computation
  - On chip communication network
- Challenges
  - ASIC design approaches do not scale
  - ASIC designers difficult to adapt (<10% will make it)

SOC/NOC Architectures

- **Heterogeneous Hardware :**
  - Specific hardware (DSP, MCU ASIC, IP, Memory, non digital….)
  - Communication Network (Bus, Switched)
  - Glue, non structured logic to links components to Network
- **Sophisticated software :**
  - High level SW (Application/OS)
  - Low level hand coded SW (Dedicated CPU, ASIP, DSP)
  - SW code to test Hardware



SoC/NoC Design

- **Challenges**
  - 200 Million gates SoC , 6 000 000 lines of RTL code, 600 MY effort
  - Dedicated Software complexity larger than HW in 2005
  - SW code to test HW, 70% design cost for multi-processor SoCs
  - Multiple Instruction set processors
- **Current SoC design**
  - RTL design for hardware System interconnects
  - Low level hand coded software for dedicated processors (e.g. DSP)
  - Low level hand coded test programs
  - Cycle true co-simulation using multiple ISSs
- **Requirements**
  - Design at a higher level than RTL of hardware interconnect
  - Higher level code for dedicated software and test Programs.
  - Global mixed and multilevel validation

Outline

1.- Network and System on Chip Design

2.- HW-SW Communication bottleneck

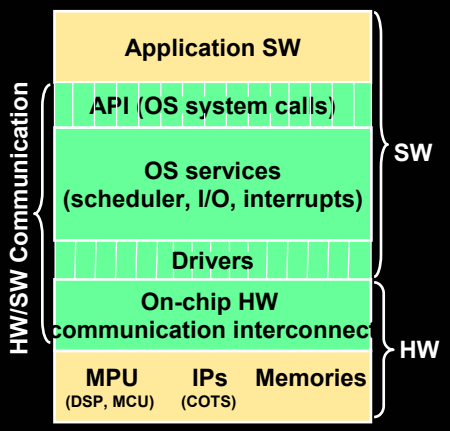
3.- Roses: Component Based Design

4. A VDSL design example

5.- Status and conclusions

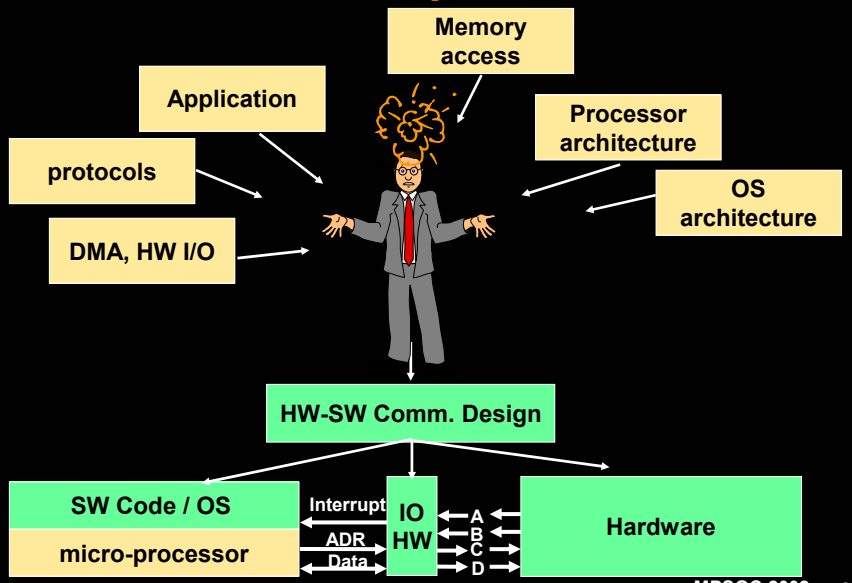
HW/SW communication design (1)

(1) Single microprocessor



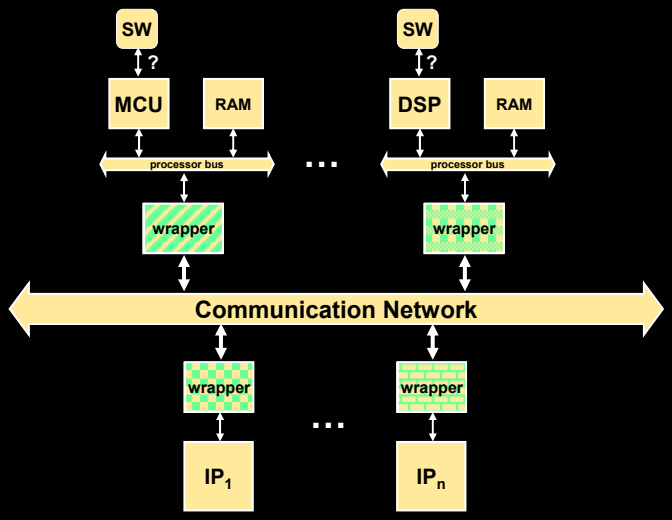
- Layered & hierarchical architecture to master complexity and coordination between multiple design teams
- Requires a combination of methods and tools that belong to different communities
- The bottleneck: HW/SW communication design and debug

Why is HW/SW communication Design So Complex?



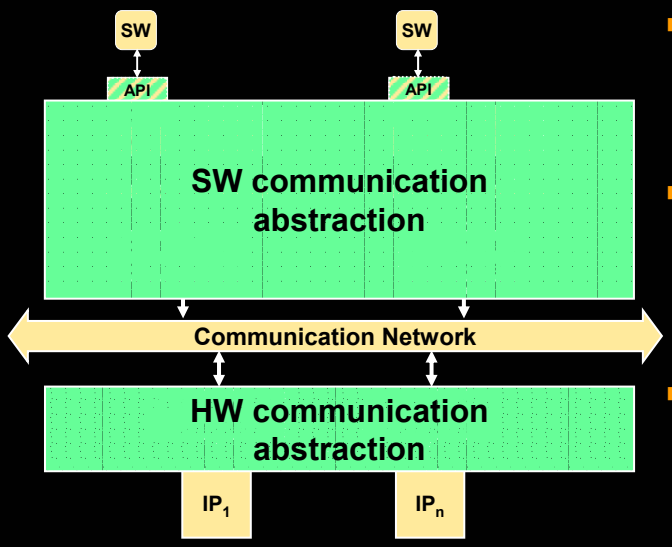
HW/SW communication design (2)

(2) Multiple cores



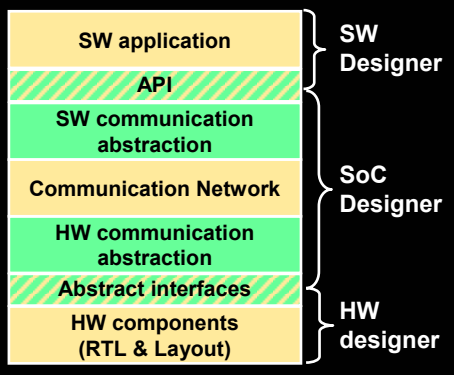
- One OS per processor?
- Distributed OS?
- No OS/static scheduling on DSPs?
- HAL designed by an external team?
- Manual wrapper design

HW/SW communication services



- One custom OS per processor providing communication services API
- Communication coprocessors/ controllers implementing communication services
- Automatic generation ?

HW/SW concurrent development



- **SW uses communication services API and configures architecture parameters (QoS requirements)**
- **HW uses an abstract interface to communication coprocessor/controller**
- **API enable independent refinement of communication and computation**
- **Reduce overheads through concurrent design**

SoC Design Organization: Multiple Teams Communicating through APIs

Practices for Boards and SoCs

- SW application
- SW communication
- Virtual architecture model

→ SW designers

- SW test bench
- HW communication
- ISSs, HW IP model

→ SoC designer

- Implementation test bench
- Components (RTL & Layout)

→ HW designer

Reducing Overheads through concurrent design

- SW application
- API implementation

→ SW Designer

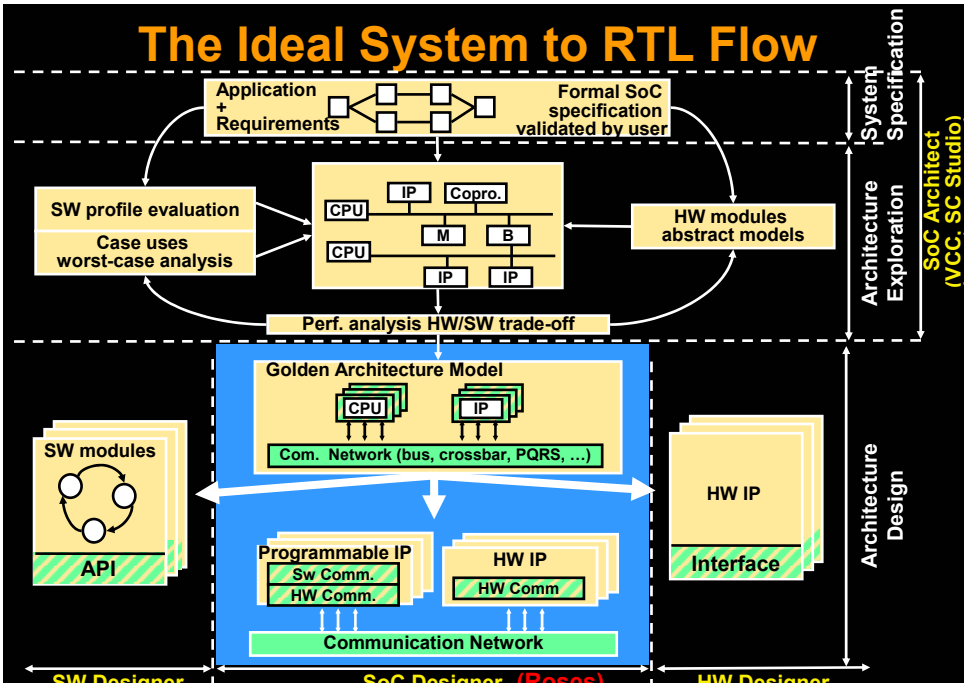
- HW/SW communication

→ SoC Designer

- Abstract Interfaces
- Components (RTL & Layout)

→ HW designer

The Ideal System to RTL Flow



Outline

1.- Network and System on Chip Design

2.- HW-SW Communication bottleneck

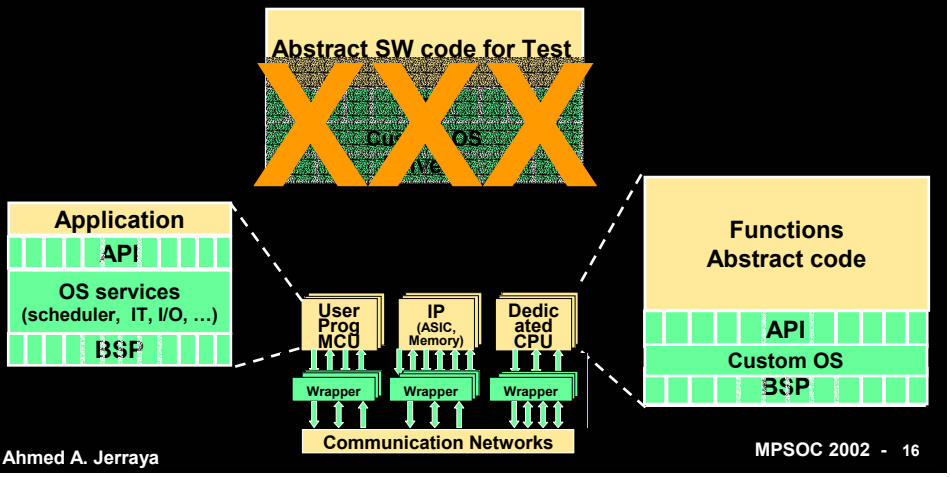
3.- Roses: Component Based Design

4. A VDSL design example

5.- Status and conclusions

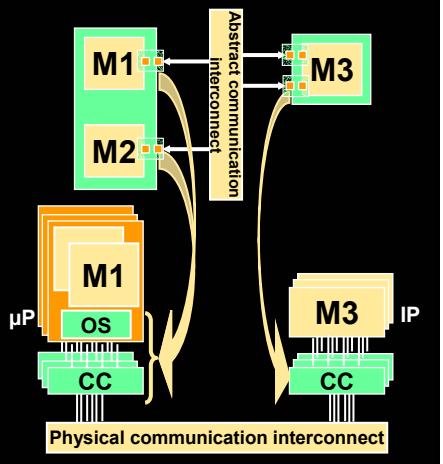
Key point: Structuring SoC Design

- Wrappers to abstract component interfaces
- Custom OS to abstract architecture
- Concurrent design, debug to cut down TTM and debug efforts



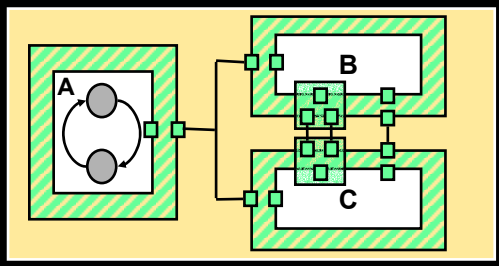
Multicore SoC design automation

- System Specification as virtual architecture: Virtual modules use wrappers to abstract HW/SW communication , Standard SW C++/SystemC Built on top of CSAPI
- Architecture implementation as: heterogeneous components and sophisticated on-chip communication interconnect linked through HW and SW wrappers, Same SW code, runs on implementation on top of OS
- Automatic generation of application-specific on-chip HW/SW communication: HW/SW CSAPI implementation

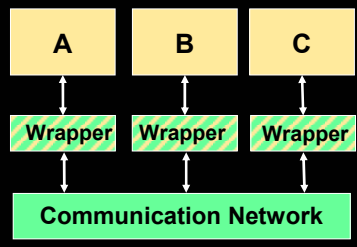


Virtual Architecture Model

- Basic model: a set of hierarchically interconnected modules representing an abstract architecture
- Basic concepts: module wrapper to isolate computation from communication
  - Internal port: connects behavior
  - External port: connects communication network
  - Virtual port: set of internal/external ports with communication API
- SystemC extended with virtual objects: virtual module, virtual port and virtual channel
- Execution model through automatic wrapper generation



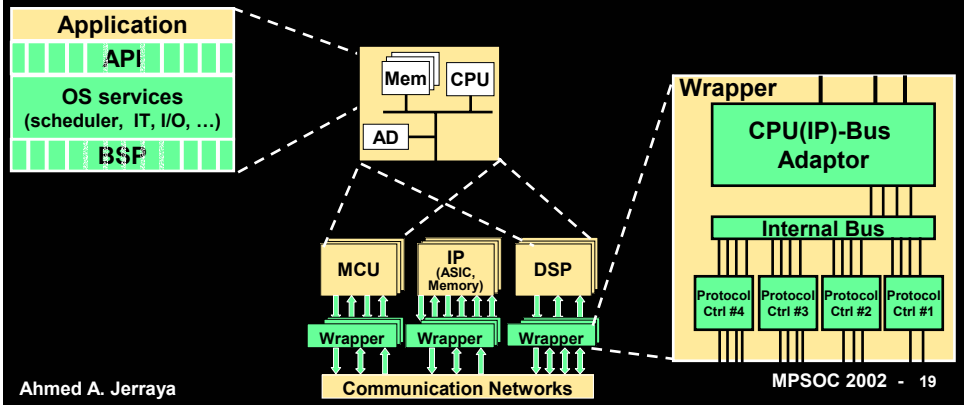
System Specification



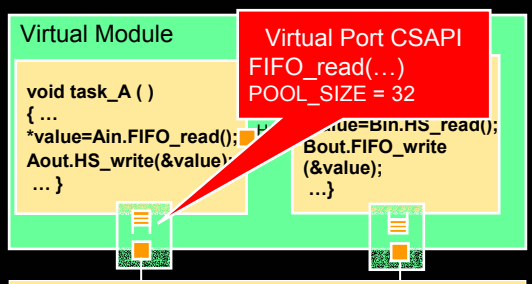
System Architecture

HW/SW Communication Architecture

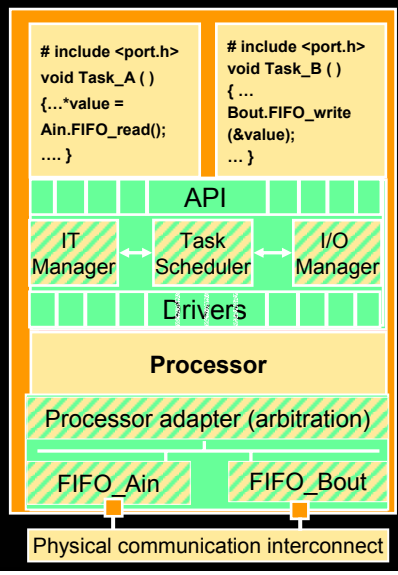
- Dissociate communication from computation
  - HW wrapper: multi-point multi-protocol
  - SW wrapper: multitask OS with preemption
- Application-specific on-chip HW-SW communication
  - Component-based systematic assembly approach for both HW and SW wrapper



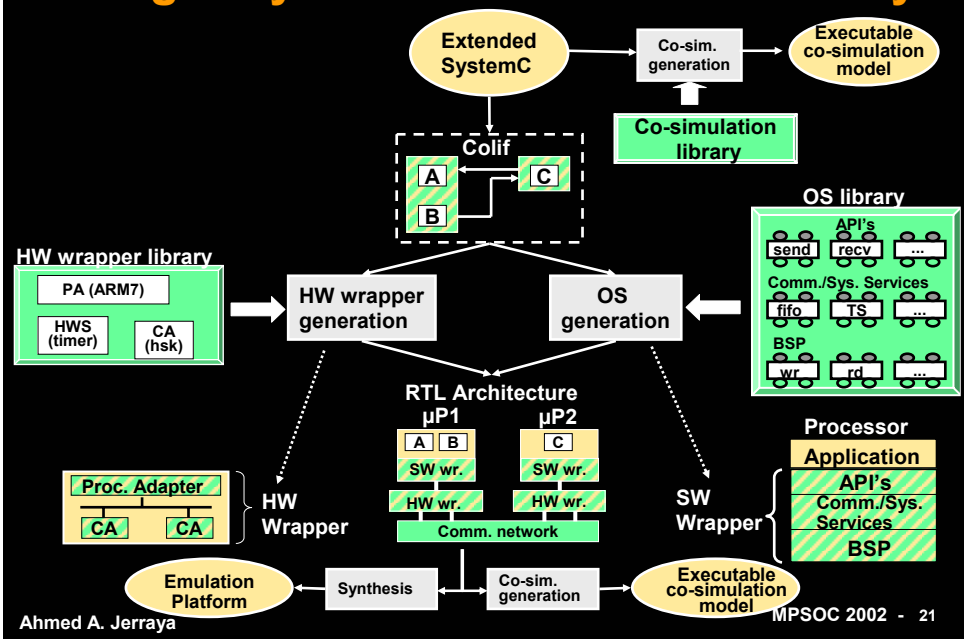
Communication services API example



- Virtual Architecture
  - HW/SW wrapper: virtual ports (API, parameters)
  - SW, standard C++/SystemC built on top of API
- RTL architecture
  - Same SW code, runs on implementation
  - SW wrapper (implements API, task control, interrupts, I/O)
  - HW wrapper (bridge to communication interconnect)

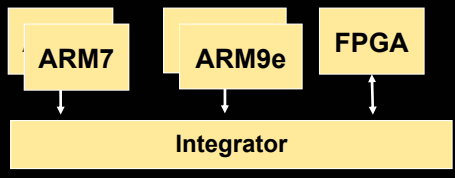


Design: Systematic HW-SW Assembly



Prototyping Platform

- Modular, Flexible, Scalable Platform
  - Integrator, 2 MGates, 2 ARM7, 2 ARM966E
- Automatic Mapping of Architecture
- Intensive debug through SW code
- Early SW development



Outline

1.- Network and System on Chip Design

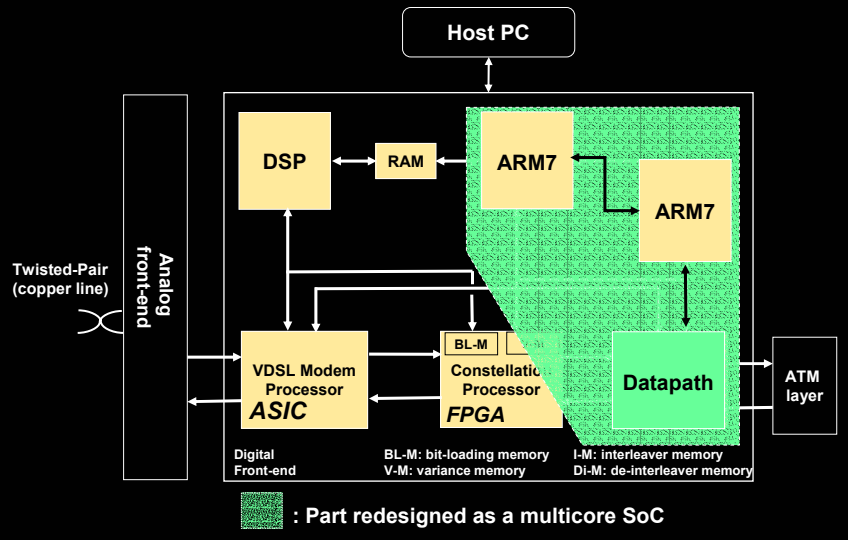
2.- HW-SW Communication bottleneck

3.- Roses: Component Based Design

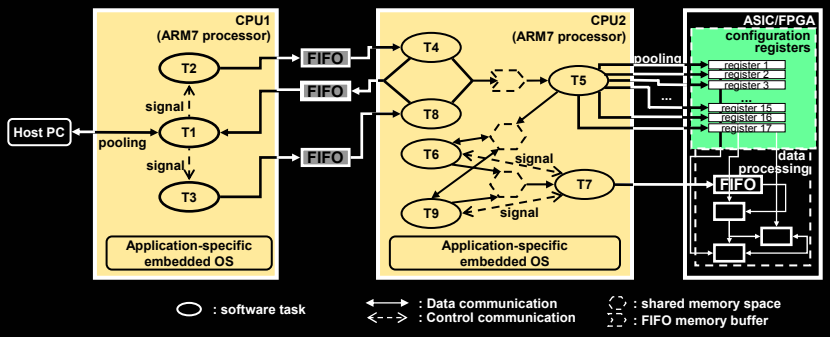
4. A VDSL design example

5.- Status and conclusions

VDSL application: the subset

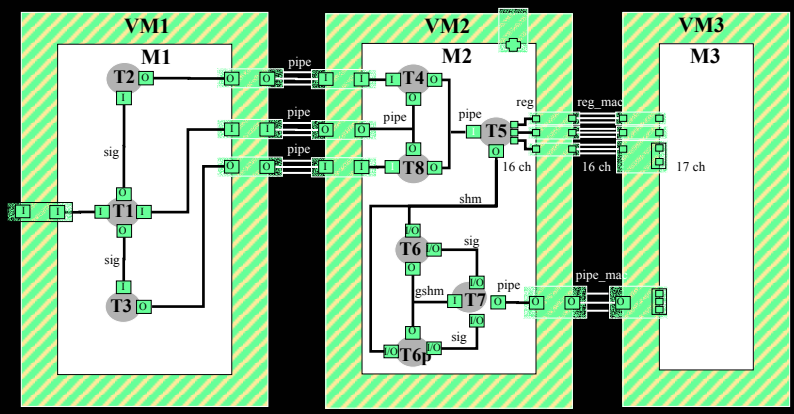


VDSL application : The Subset



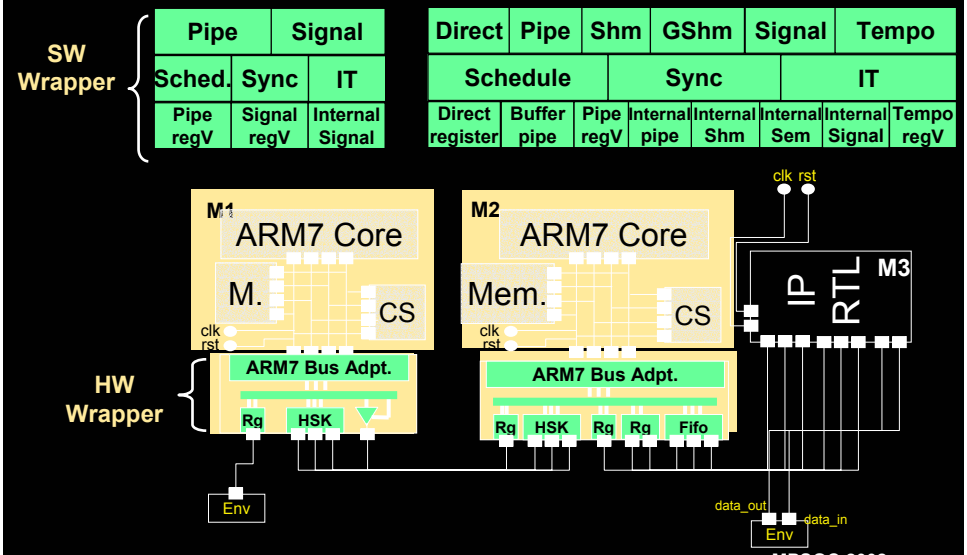
- Inter processor communication.
- Communication between processor and ASIC.
- Parallel communicating tasks
- different communication protocols.

The specification



- Powerful modular scheme
  - Parallel SW tasks, Sophisticated functions or test Programs
  - Clear separation between communication and computation
  - Reference model for all designers (HW, SW, SoC)

The Architecture



The Two Generated Operating Systems

M1

Pipe		Signal
Sched.	Sync	IT
Pipe regV	Signal regV	Internal Signal

M2

Direct	Pipe	Shm	GShm	Signal	Tempo
Schedule		Sync		IT	
Direct register	Buffer pipe	Pipe regV	Internal pipe	Internal Shm	Internal Sem	Internal Signal	Tempo regV

- **Sizes:**
  - M1 : 1484 byte code
    500 byte data
  - M2 : 2624 byte code
    1020 byte data
- **Performances:**
  - Context change: 36 cycles (1,44 µs)
  - Interrupt latency: 59+28 cycles (3.48µs)
  - System call latency: 50 cycles (2.00µs)
  - Task resume: 26 cycles (1.04µs)

VDSL design results (summary)

- Design effort gain: 15x (5 m.y vs. 4 person. months)
- Virtual architecture model: 1760 lines
- RTL architecture (0.35μm technology):

OS results	# of lines in C	# of lines in Assembly	Code size (bytes)	Data size (bytes)
VM1	968	281	3829	500
VM2	1872	281	6684	1020
Context switch (cycles)				36
Latency for interrupt treatment (cycles)				59 (OS) + 28 (ARM7)
System call latency (cycles)				50
Resume of task execution (cycles)				26

Communication coprocessors	# of gates	Critical path delay (ns)	Max. freq. (MHz)
VM1	3284	5.95	168
VM2	3795	6.16	162
Latency for read operation (clock cycles)			6
Latency for write operation (clock cycles)			2
Number of code lines (RTL VHDL)			2168

Conclusion

- Network and System-on-Chip:

  CPU(s) + IP(s) + on-chip communication
- Design bottleneck: HW/SW interfaces
- Structuring SoC Design
  - API separates Application SW from OS
  - Wrappers separate IP(s) from Network
- Perspectives:
  - HW-SW interfaces Debug and Test
  - Partitioning HW/SW interfaces
  - Computation/Communication Partitioning

Readings about MP_SoC

1. D.E. Culler, J. Pal Singh, "Parallel Computer Architecture," Morgan Kaufmann Publishers, 1999.
2. Oka and Suzuoki, "Designing and Programming the Emotion Engine," IEEE Micro, vol. 19:6, pp. 20-28, Nov/Dec 1999.
3. M. Diaz-Nava, G.S. Okvist, "The Zipper prototype: A Complete and Flexible VDSL Multi-carrier Solution", ST Journal special issue xDSL, September 2001.
4. J. T. J. van Eijndhoven, al. "TriMedia CPU64 Architecture," ICCD, Austin, TX, 1999, pp. 586-592.
5. K. Keutzer, "A Disciplined Approach to the Development of Platform Architectures," Synthesis and System Integration of Mixed Technologies, SASIMI, Nara, Japan, October 18 - 19, 2001.
6. K. Keutzer, et al., "System-level design: orthogonalization of concerns and platform-based design," IEEE TCAD, Dec. 2000.
7. M. Sgroi, et al., "Addressing the System-on-Chip Interconnect Woes Through Communication-Based Design," Proc. of 38th Design Automation Conference, Las Vegas, June 2001.
8. IBM Inc., Blue Logic Technology, http://www.chips.ibm.com/bluelogic/
9. D. Wingard, "MicroNetwork-Based Integration for SOCs," Proc. of DAC, Las Vegas, June 2001.
10. J. A. J. Leijten et al., "PROPHID : A Heterogeneous Multi-Processor Architecture for Multimedia," Proc. of ICCD, 1997.
11. L. Benini, G. De Micheli, "Networks on chips: A New SoC Paradigm", Computer, Vol. 35 No 1, pp. 70-78, January 2002.
12. K. Goossens, E. Rijpkema, P. Wielage, A. Peeters and J. van Meerbergen, "Networks on Silicon: The next Design paradigm for systems on Silicon", DATE 2002, Paris, France, March 2002.

Readings about Roses

1. W. CESARIO, al., "Component-Based Design Approach for Multicore SoCs", DAC'02, New Orleans, USA June 10-14 2002.
2. S. Yoo, al. « A Generic Wrapper Architecture for Multi-Processor SoC Cosimulation and Design », *CODES*, 2001.
3. W.O. Cesario, al. "Colif: a Multilevel Design Representation for Application-Specific Multiprocessor System-on-Chip Design", IEEE Design & Test, Sept, 2001.
4. A. Baghdadi, al. « An Efficient Architecture Model for Systematic Design of Application-Specific Multi-processor SoC », *Design Automation and Test in Europe*, March, 2001.
5. L. GAUTHIER, al., "Automatic Generation and Targeting of Application Specific Operating Systems and Embedded Systems Software", TCAD, IEEE Transactions on Computer-Aided Design, Vol. 20 Nr. 11, November 2001.
6. D. Lyonnard, al. "Automatic Generation of Application-Specific Architecture for Heterogeneous Multiprocessor System-on-Chip", DAC, June, 2001.
7. S. YOO, G. NICOLESCU, L. GAUTHIER, A.A. JERRAYA, "Automatic Generation Including Fast Timed Simulation Models of Operating Systems in Multiprocessor SoC Communication Design", DATE 2002, Paris, France, March 2002.
8. P. Gerin, al. « Scalable and Flexible Co-simulation of SoC Design with heterogeneous Multi-processor Target Architecture », *Asia South Pacific Design Automation conference*, January, 2001.
9. F. GHARSALLI, al., "Embedded Memory Wrapper Generation for Multiprocessor SoC", DAC'02, June 10-14 2002, New Orleans, USA.