codesign environment

디지털 시스템 설계방법론
(Codesign Methodology of Digital System Design)

Fall, 2007

Soonhoi Ha

Goals

- **Understand the Procedure and Issues of Embedded System Design**
- **Learn the Key Techniques of HW/SW Codesign Methodology**
  - Modeling and Abstraction
  - Synthesis
  - Simulation
  - And others
- **Experiment with CAD tools**
- **Apply the Methodology to Your Design of Interest**

PeaCE: Ptolemy extension as a Codesign Environment

Schedule

- 1. Introduction
- 2. System Modeling Language: System C *
- 3. HW/SW Cosimulation *
- 4. C-based Design *
- 5. Data-flow Model and SW Synthesis
- 6. HW and Interface Synthesis
  (Midterm)
- 7. Models of Computation
- 8. Model based Design of Embedded SW
- 9. Design Space Exploration
  (Final Exam)
  (Term Project)

Announcement

- **Class homepage: http://peace.snu.ac.kr/courses/codesign07/**

- **Grades: (tentative)**
  - midterm and final exam: 30% each
  - Homework and experiments: 25%
  - Final project: 15%

codesign environment

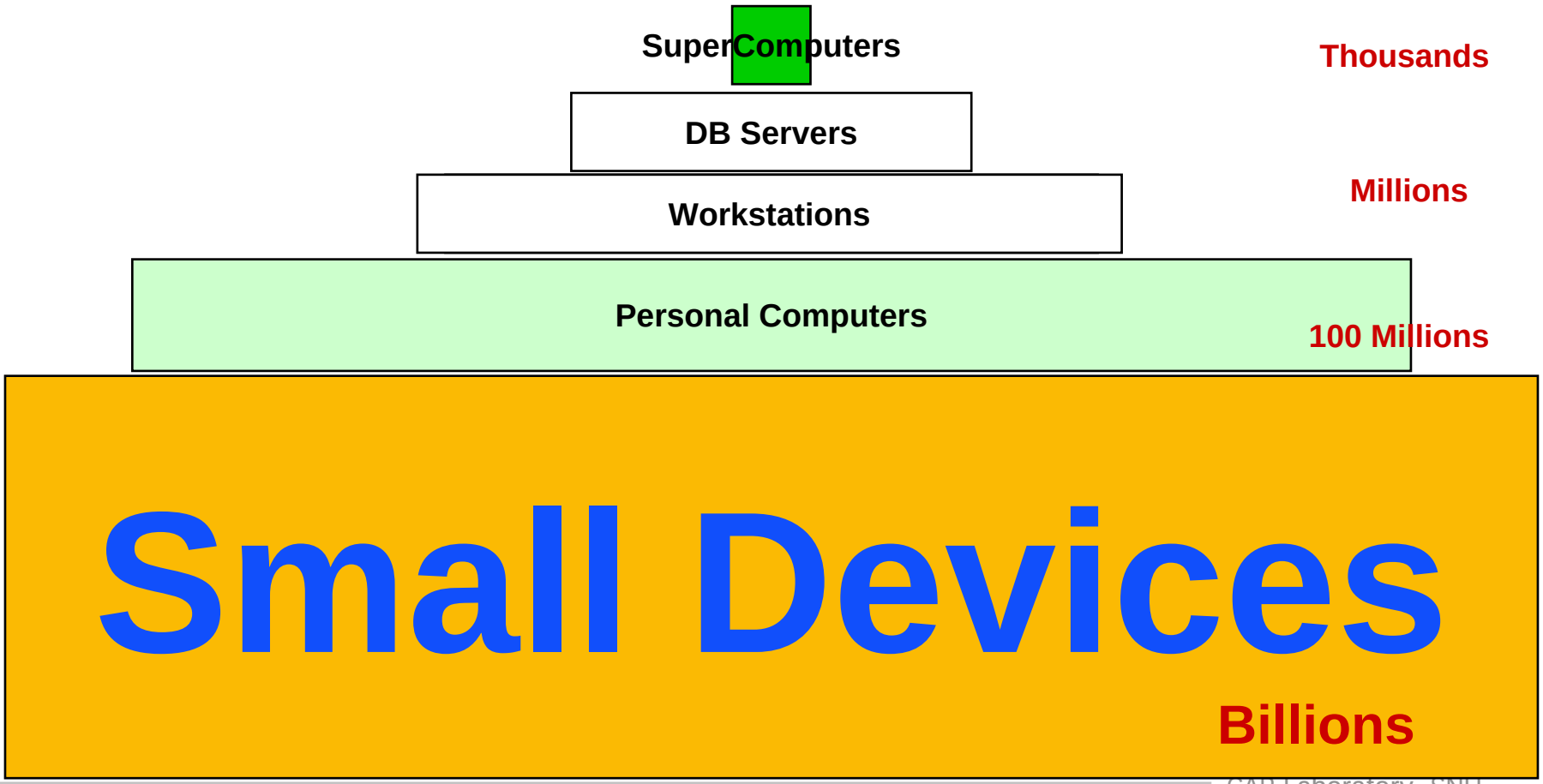
1. Introduction

- **References**
  - [1] Kurt Keutzer, et. al. “System-Level Design: Orthogonalization of Concerns and Platform-Based Design," IEEE TCAD, 19(12), December 2000.
  - [2] Stephen Edwards, et. al. "Design of Embedded Systems: Formal Models, Validation, and Synthesis," Proceedings of the IEEE, 85(3), March 1997.
  - [3] W. Wolf, “Modern VLSI Design: System-on-Chip Design”, Prentice Hall, 2002.
  - [4] D. Densmore, A. Sangiovanni-Vincentelli, and R. Passerone, “A Platform-Based Taxonomy for ESL Design,” IEEE Design&Test, (23)5, pp. 359-374, September, 2006.
  - [5] B. Bailey, G. Martin, and T. Anderson, “Taxonomies for the Development and Verification of Digital Systems,” Springer, 2005.
  - [6] M. Burton and A. Morawiec, “Platform Based Design at the Electronic System Level,” Springer, 2006.
  - [7] J. Rowson and A. Sangiovanni-Vincentelli, “Interface-based design,” 34th DAC 1997

Contents

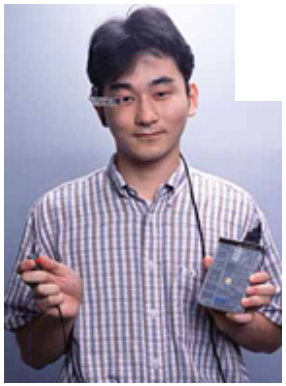
- Embedded System Design
- Design Methodology: Old and New
- Design Reuse: Platform Based Design (PBD)
- HW/SW Codesign
- Formal Model and CAD tools
- Other Design Issues – not to be covered in this course

Pyramid of Computing Platforms

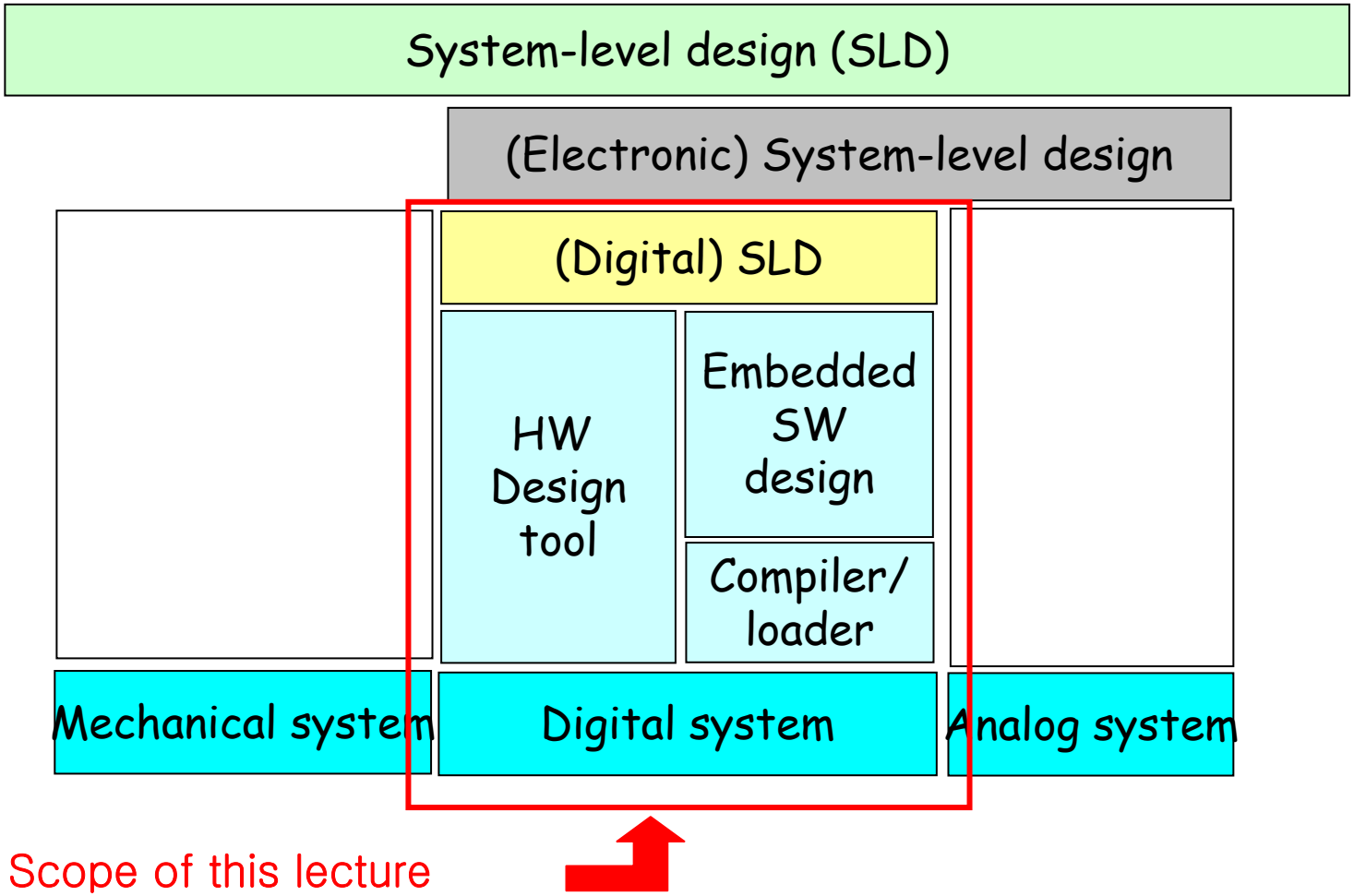


Small Devices: Post-PC Devices

* * *



System Level Design



codesign environment

Application-specific System

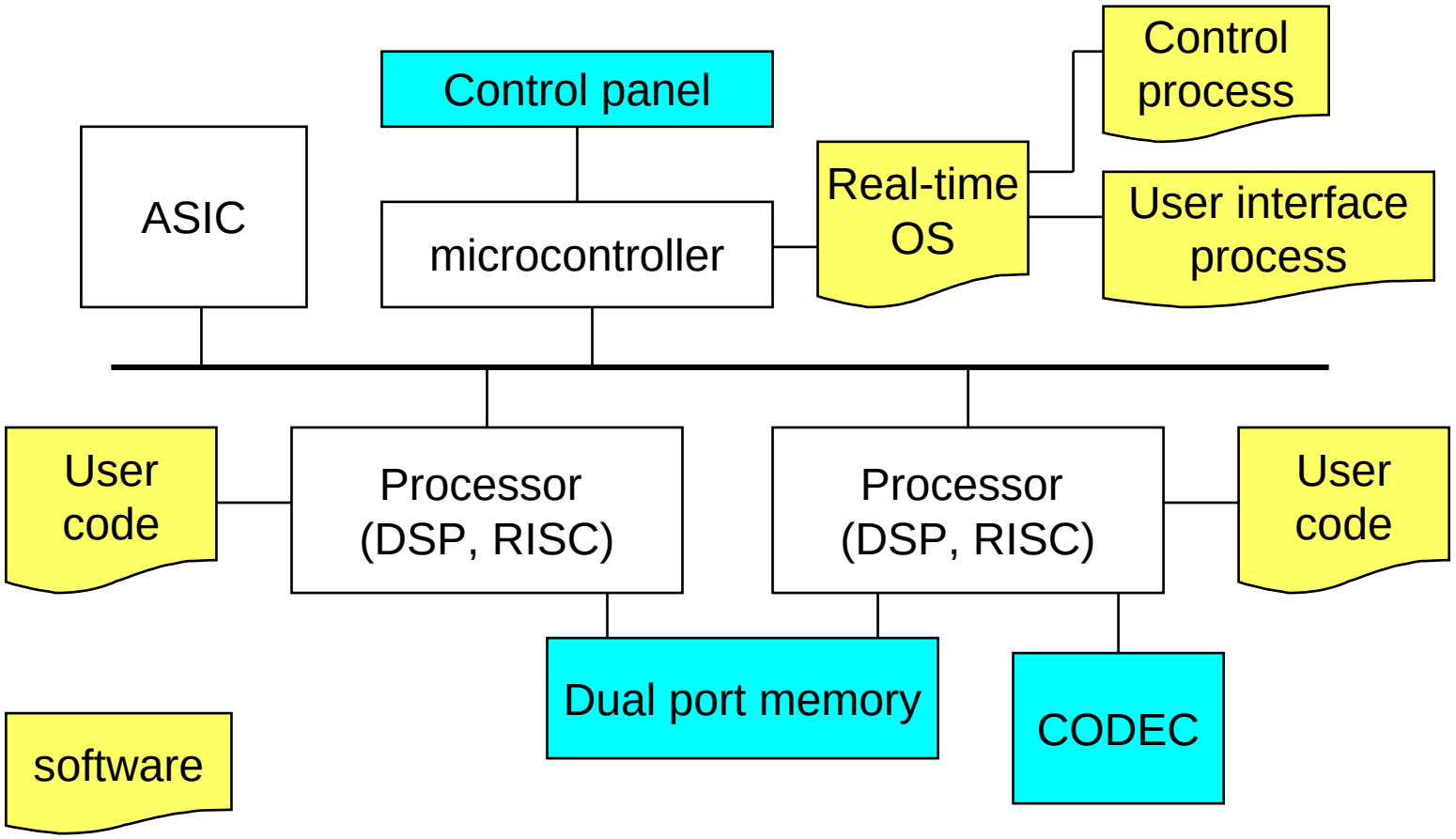
- **Computation oriented applications**
  - standalone DSP applications: MP3 player, MPEG decoder, etc
  - Graphic applications
- **Control oriented applications**
  - home appliances: microwave oven (user interactability)
  - industrial controller (reliability, availability)
  - safety-critical controller (reliability, safety)
- **Computation + control**
  - portable information devices: cellular phone
  - networked multimedia applications

codesign environment

Design Considerations

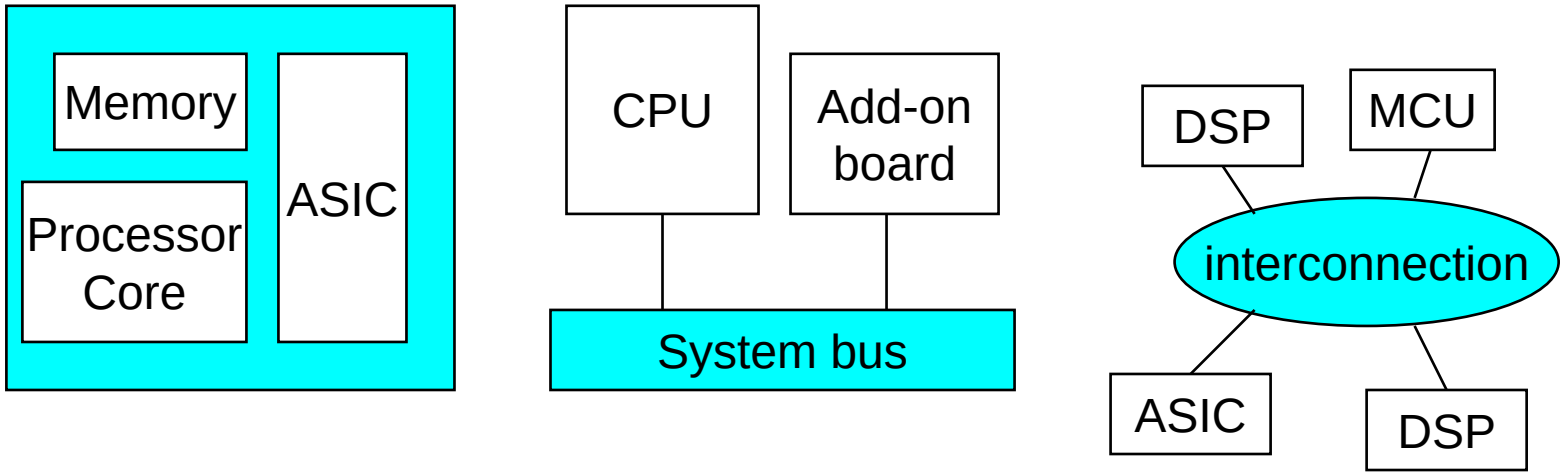
- **Real-time Constraints**
  - non-termination interaction with environment
  - Timing hardness
    - transformational system
    - interactive systems
    - reactive systems
- **Integration of concurrent components: SW and HW**
  - concurrency control, synchronization
- **Requirements**
  - Cost-performance ratio
  - Power/energy consumption
  - User friendliness
  - Shape and Weight
  - Time to market
  - reliability, safety, availability

An Architecture Example: Heterogeneity!



(courtesy of S.Edwards et. al.)

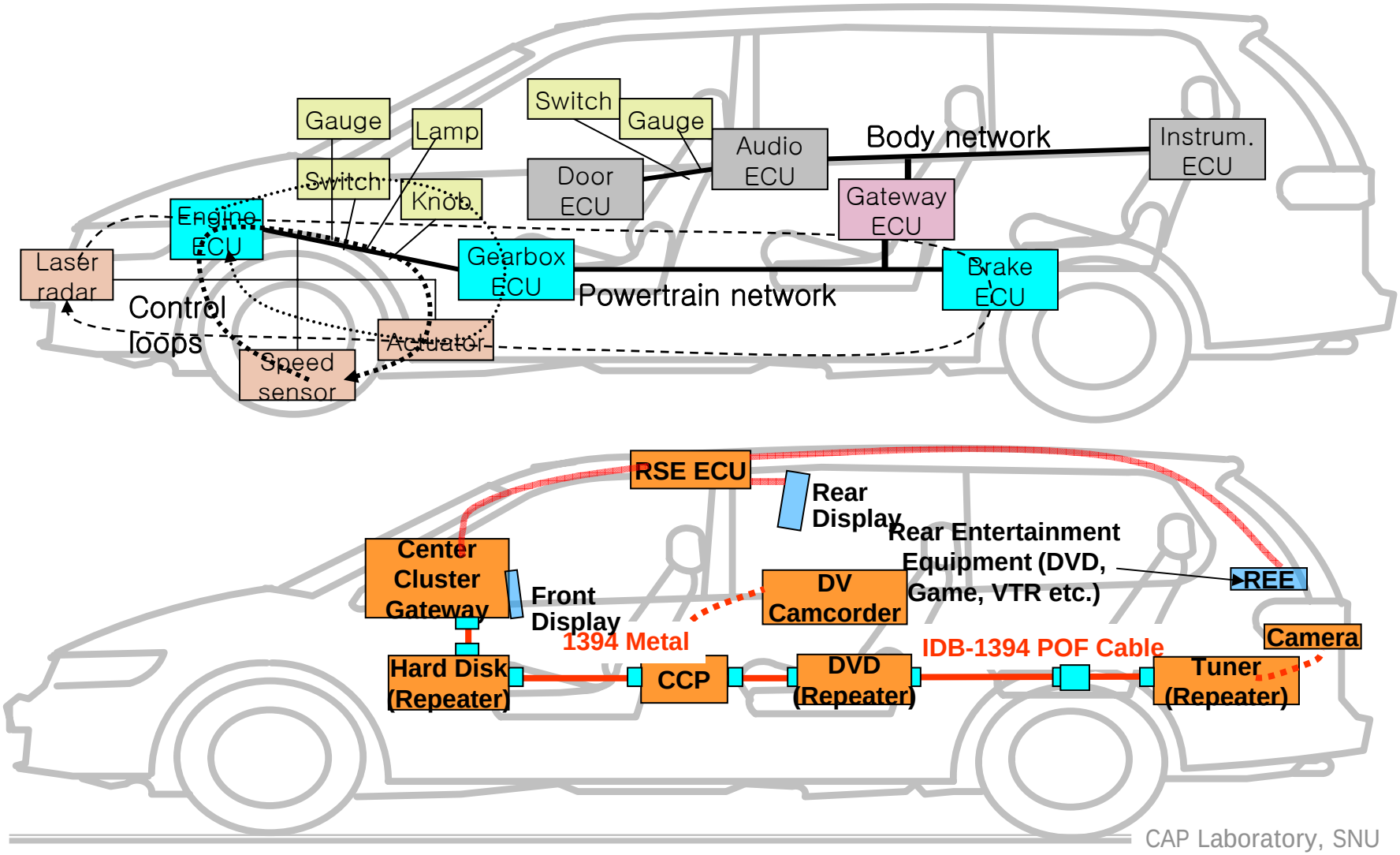
Spectrum of Target Architecture



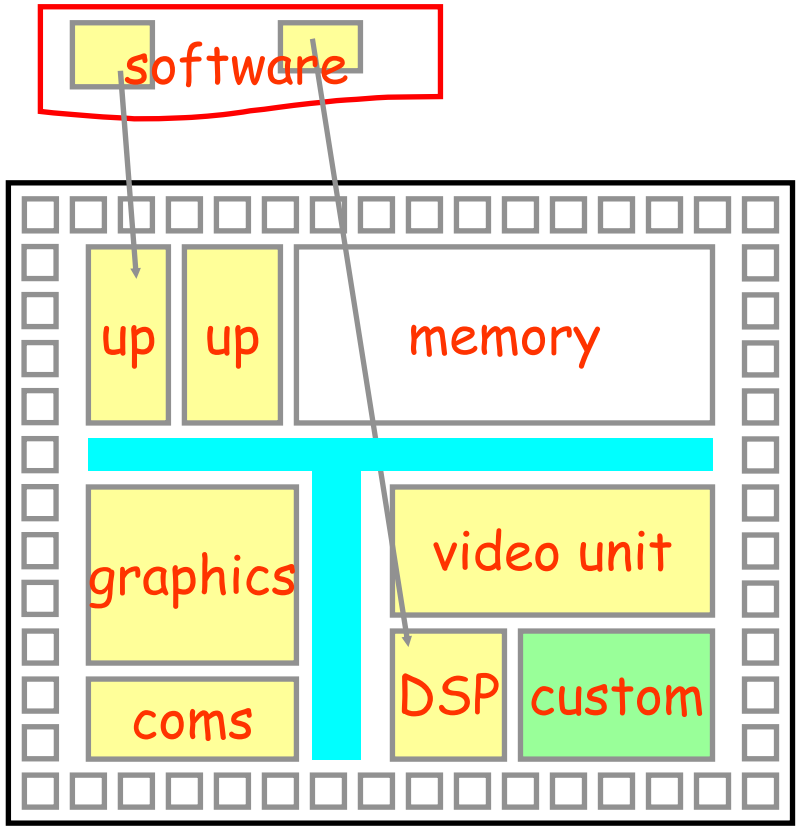
SOC (system on chip)

Distributed embedded system

Automobile is an Embedded System

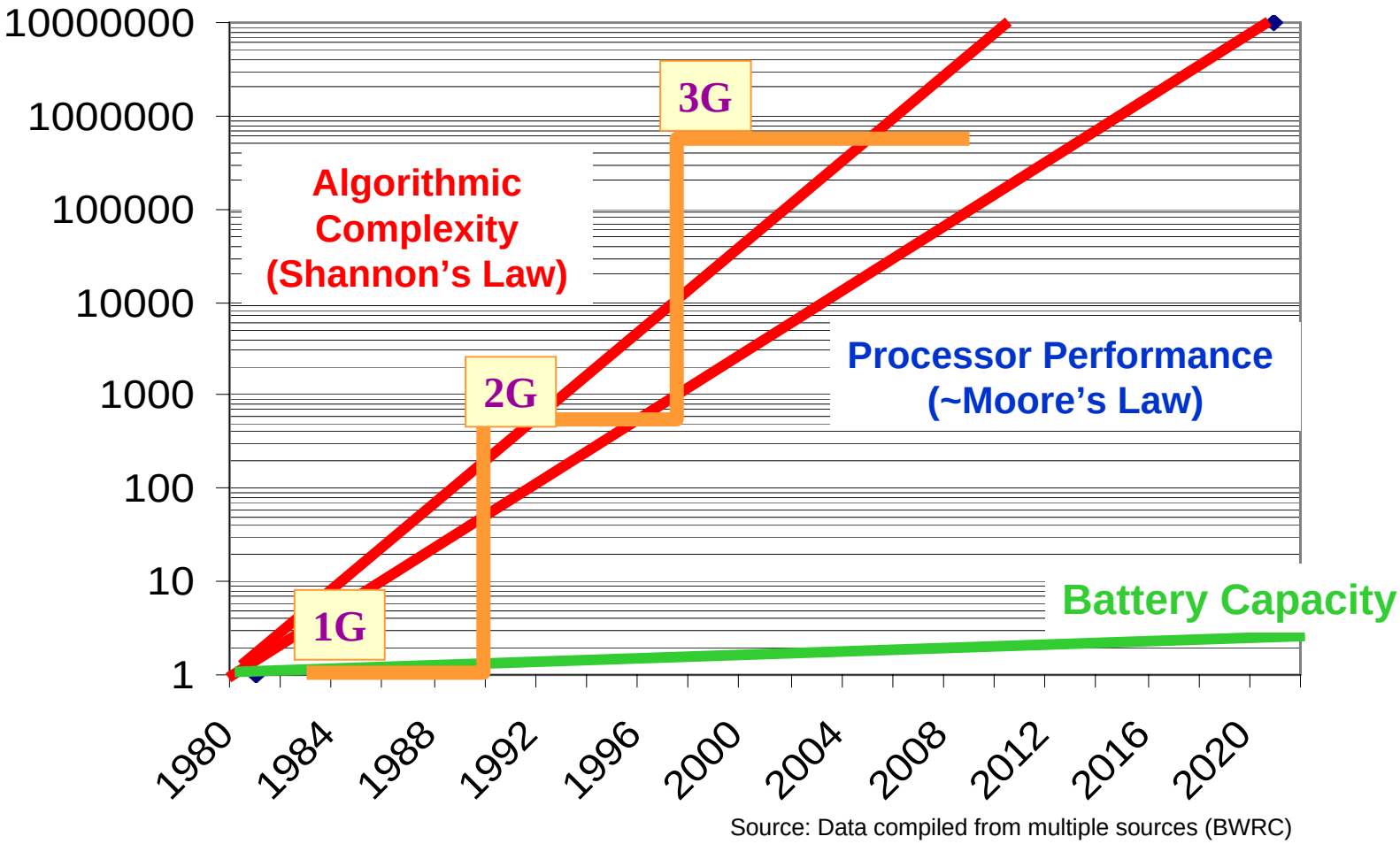


What is SoC ?



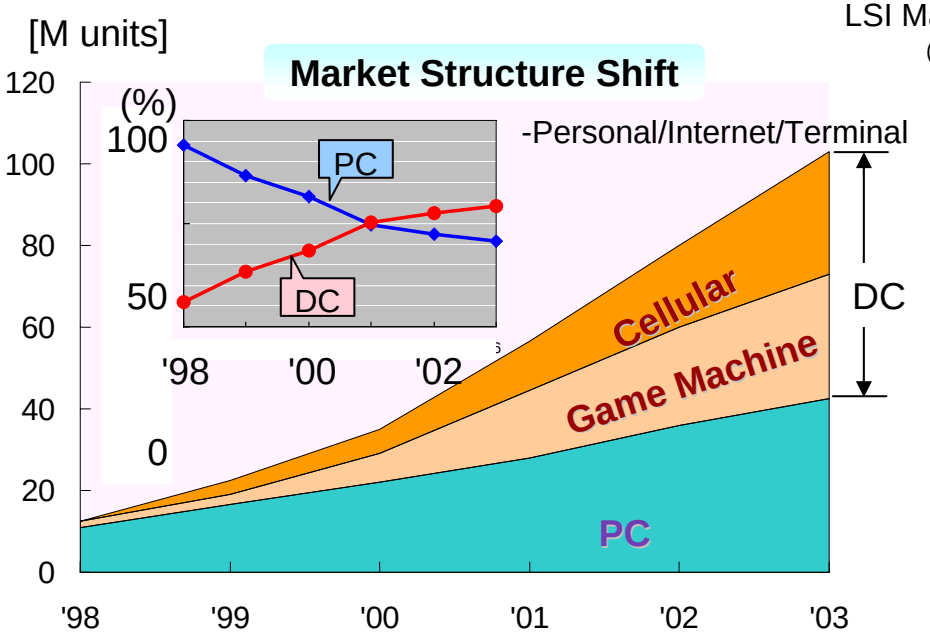
- **Increasing complexity MPSoC, NoC**
- **Assembly of "prefabricated component"**
  - Maximize VC(IP) reuse: over 90%
  - New economics: fast and correct design > optimum design
- **Design and Verification at the system level**
  - interface between VCs
  - SW becomes more important

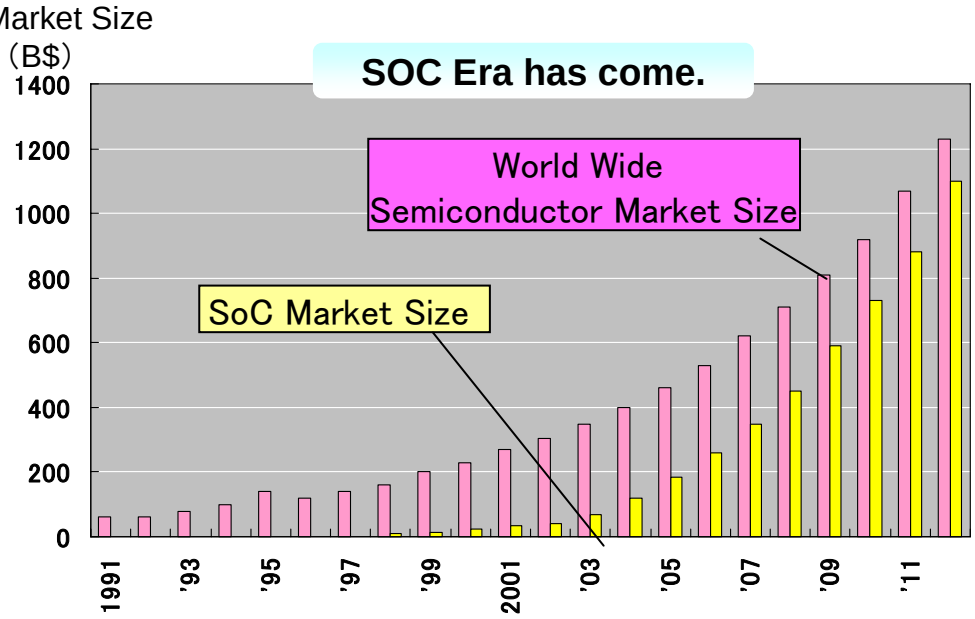
Technology Trend



Source: Data compiled from multiple sources (BWRC)

SoC Era Has Come





- PC →DC
- Wintel → Non-Wintel
- Shift of Technology Driver

90's
- PC

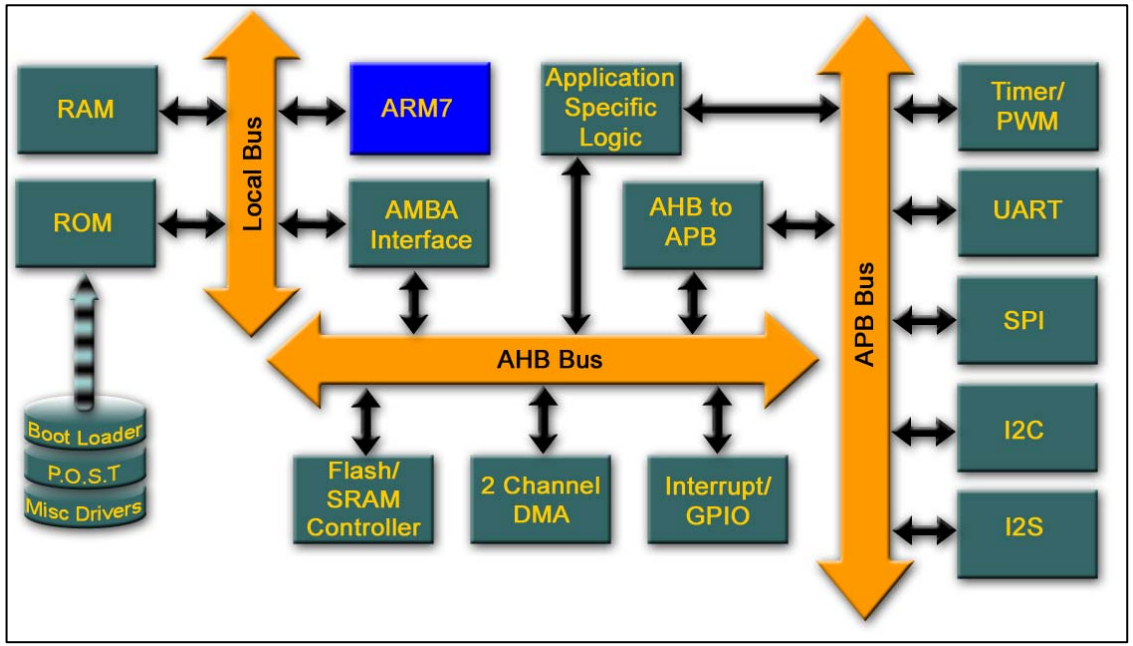
00's
- High Performance ~ Game Machine
- Low Power ~ Cellular

- Current Percentage of SoC Ratio is under 10%.
  ⇒40% in 2005, 70~80% in 2010
- SoC is "single-seat constituency ", "take or not".
- Key Factor is the Synergy between Semiconductor & Set Divisions.

출처: SONY

SoC Architecture

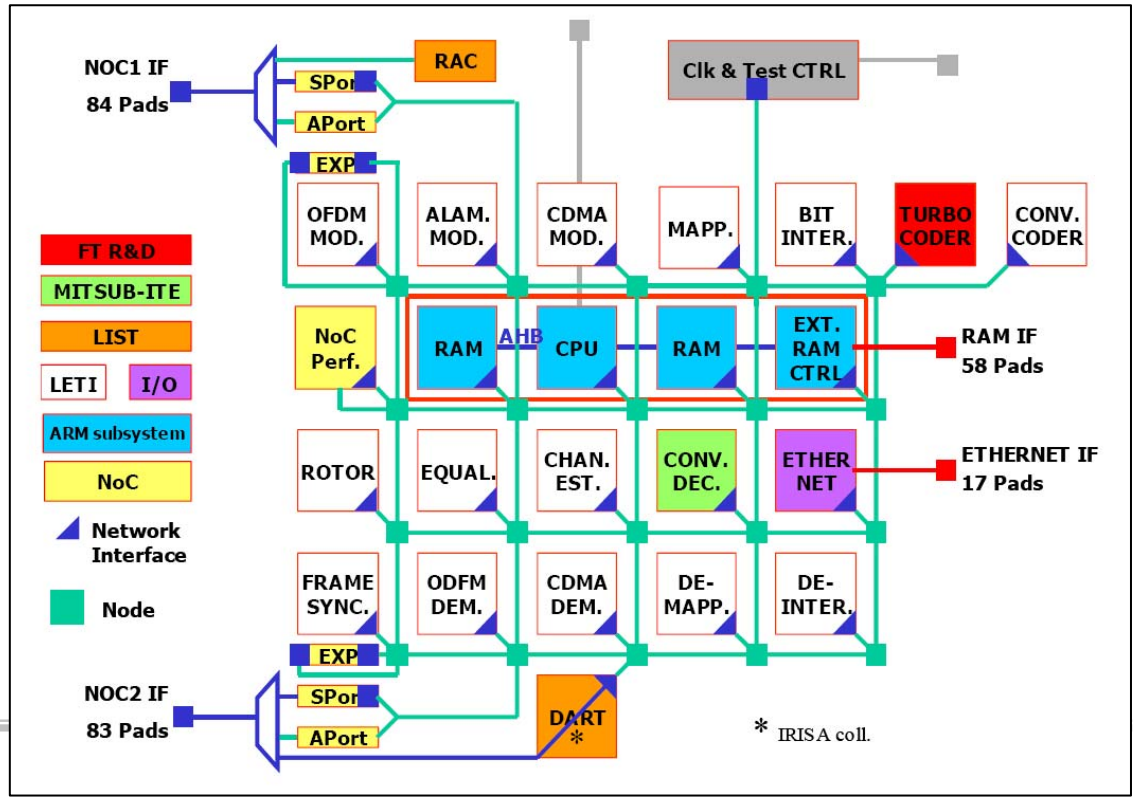
- **Hardware Architecture**
  - CPU, Hardware IP
  - Diverse memory elements
  - I/O interface
  - Bus
- **SoC Complexity is increasing**
  - # processing elements grows
  - Communication architecture
    - Switched bus
    - NoC (Network on Chip)



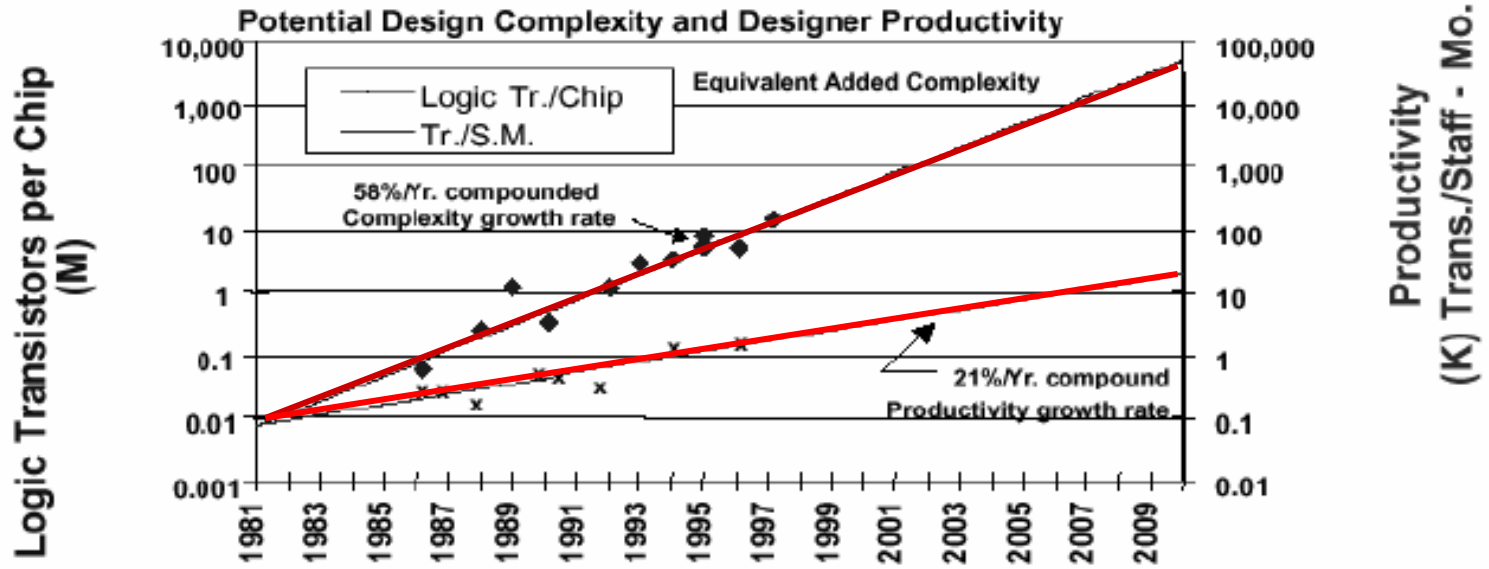
Intrinsix AMBA SoC Platform, Intrinsix Co.
http://www.intrinsix.com/intrinsix-ip/soc-ip/amba.htm

NoC Example

- **FAUST chip architecture**
  - 4G mobile application
  - 20 nodes with 23 IPs are connected by a network
  - 8MG / 0.13 μ CMOS technology (STMicroelectronics)



Productivity Gap



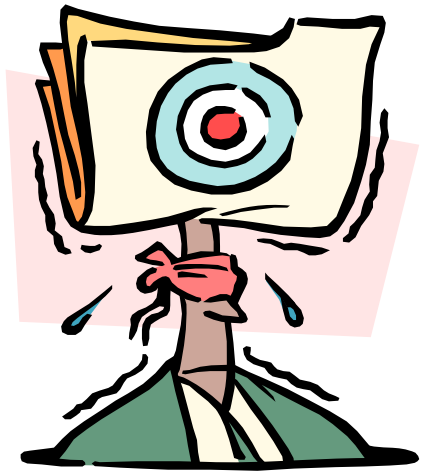
Year	Technology	Chip Complexity	Frequency	3 Yr. Design Staff	Staff Cost*
1997	250 nm	13 M Tr.	400	210	90 M
1998	250 nm	20 M Tr.	500	270	120 M
1999	180 nm	32 M Tr.	600	360	160 M
2002	130 nm	130 M Tr.	800	800	360 M

* @ $150K / StaffYr. (In 1997 Dollars)

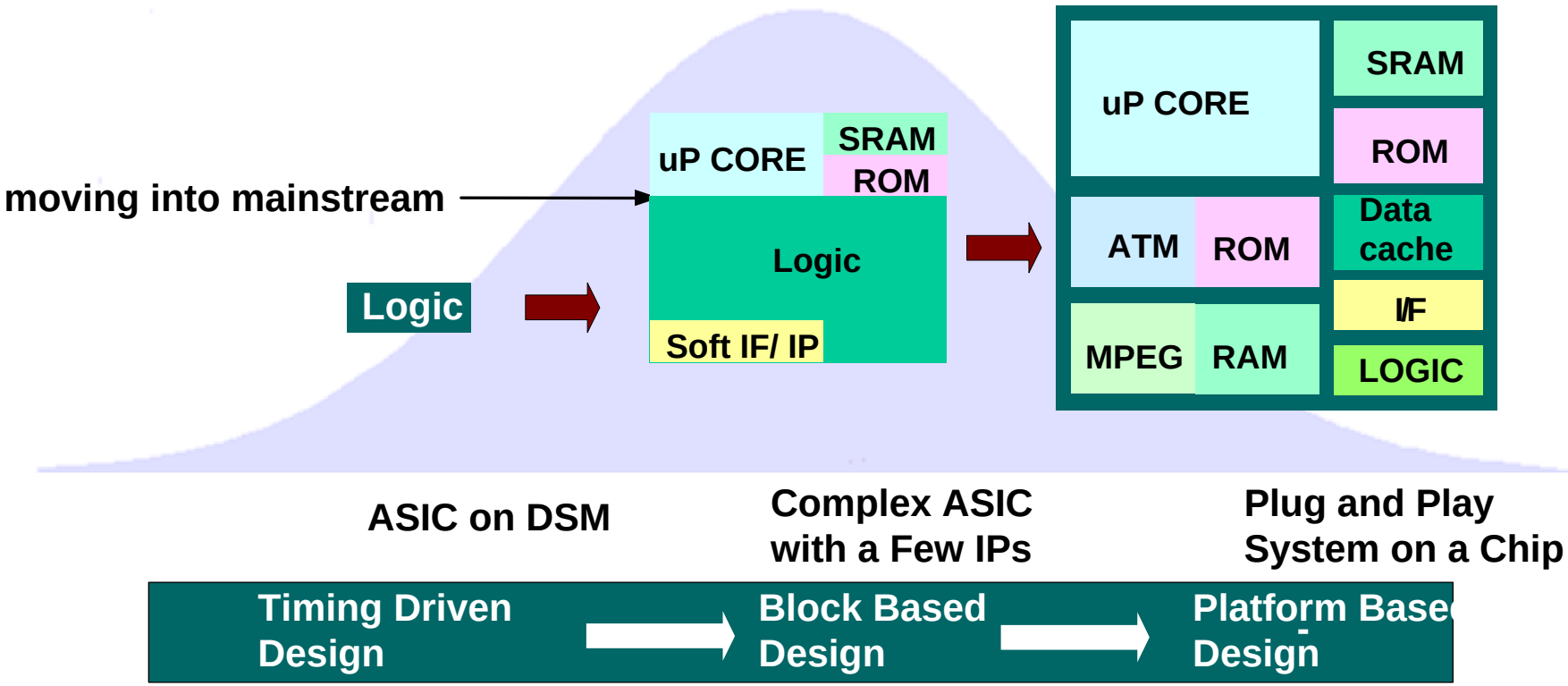
Source: ©SEMATECH

Conflicting Needs

- **Unsatiable need for higher performance**
  - Multiprocessors
  - Hardware + software architecture
- **Higher energy efficiency**
  - MIPS/mW is the first-class design objective
- **Complexity management**
  - Design space exploration
  - Design verification
- **Increased design productivity**
  - Reduce design cost and design time
- **DSM (Deep-submicron) technology**
  - Mask cost increases significantly
  - Need new CAD tools



Chip Design is Now System Design

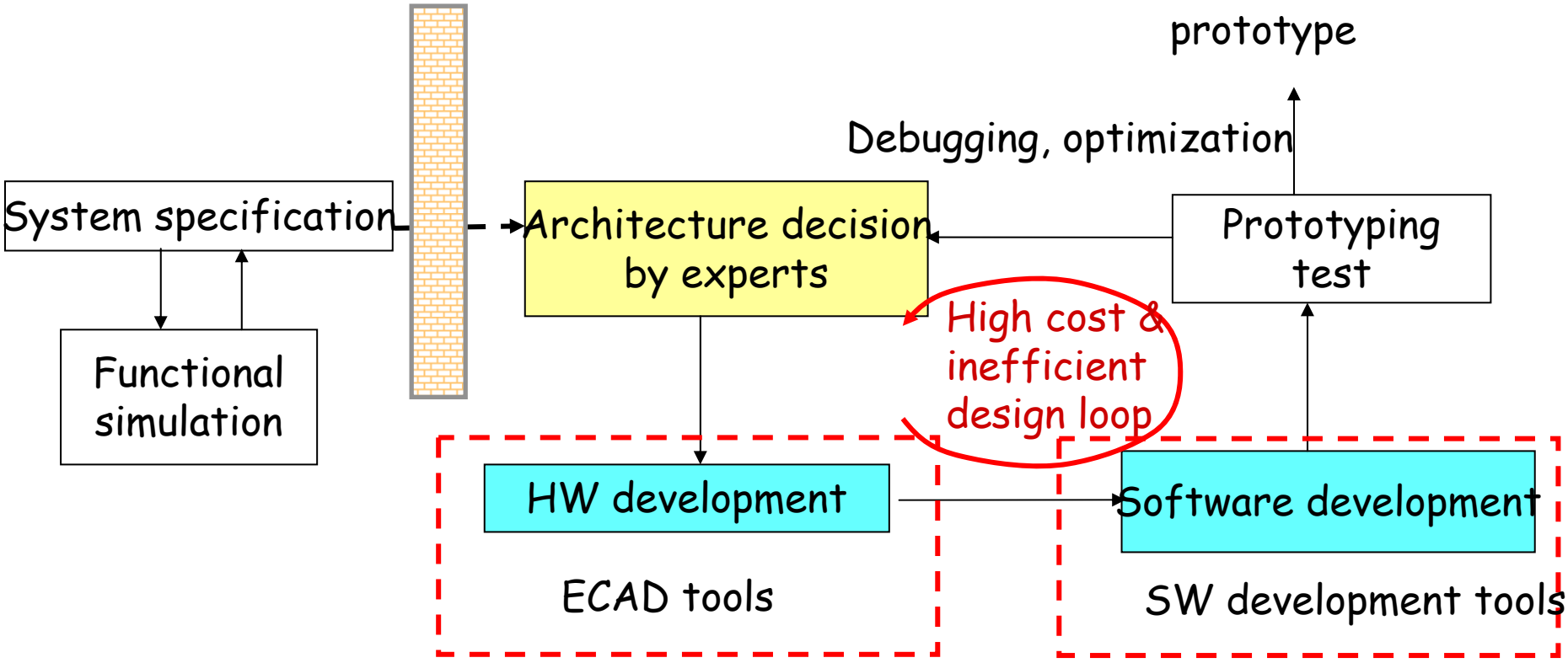


codesign environment

Contents

- Embedded System Design
- Design Methodology: Old and New
- Design Reuse: Platform Based Design (PBD)
- HW/SW Codesign
- Formal Model and CAD tools
- Other Design Issues – not to be covered in this course

Conventional Method



→ Early separation of HW and SW
Slow/ high cost/ hard to maintain and debug

Problems

- **Use different languages for modeling and implementation**
  - Cannot verify the desired functions directly
- **Hardware designers have to restart the design process by capturing the designs using the HDLs**
  - May have unmatched problems
- **Require many experts in system architecture for the partition of software and hardware parts**
  - The partition may not be the optimal solution
- **Hardware and software integration is often painful**
  - Hardware and software cannot work together
  - Co-verification of hardware and software is inefficient

- **Long design time and high-cost**
- **Verification and Debugging is painful**

Needs of New Methodology

- **[1] Kurt Keutzer, et. al. "System-Level Design: Orthogonalization of Concerns and Platform-Based Design," IEEE TCAD, 19(12), December 2000.**

"we believe that the lack of appropriate methodology and tool support for modeling of concurrency in its various forms is an essential limiting factor in the use of both RTL and commonly used programming languages to express design complexity"

New Methodology

- (1) Design Reuse: Virtual Component (IP) based design & platform based design
- (2) HW/SW Codesign: Systematic Design Methodology
- (3) Model-based design: formal model is good for verification
- (4) Effective Use of Design Automation Tools

System Level Design Science

- **Design Methodology:**
  - Top Down Aspect:
    - Orthogonalization of Concerns:
      - Separate Implementation from Conceptual Aspects
      - Separate computation from communication
    - Formalization: precise unambiguous semantics
    - Abstraction: capture the desired system details (do not overspecify)
    - Decomposition: partitioning the system behavior into simpler behaviors
    - Successive Refinements: refine the abstraction level down to the implementation by filling in details and passing constraints
  - Bottom Up Aspect:
    - IP Re-use (even at the algorithmic and functional level)
    - Platform architecture from pre-existing library

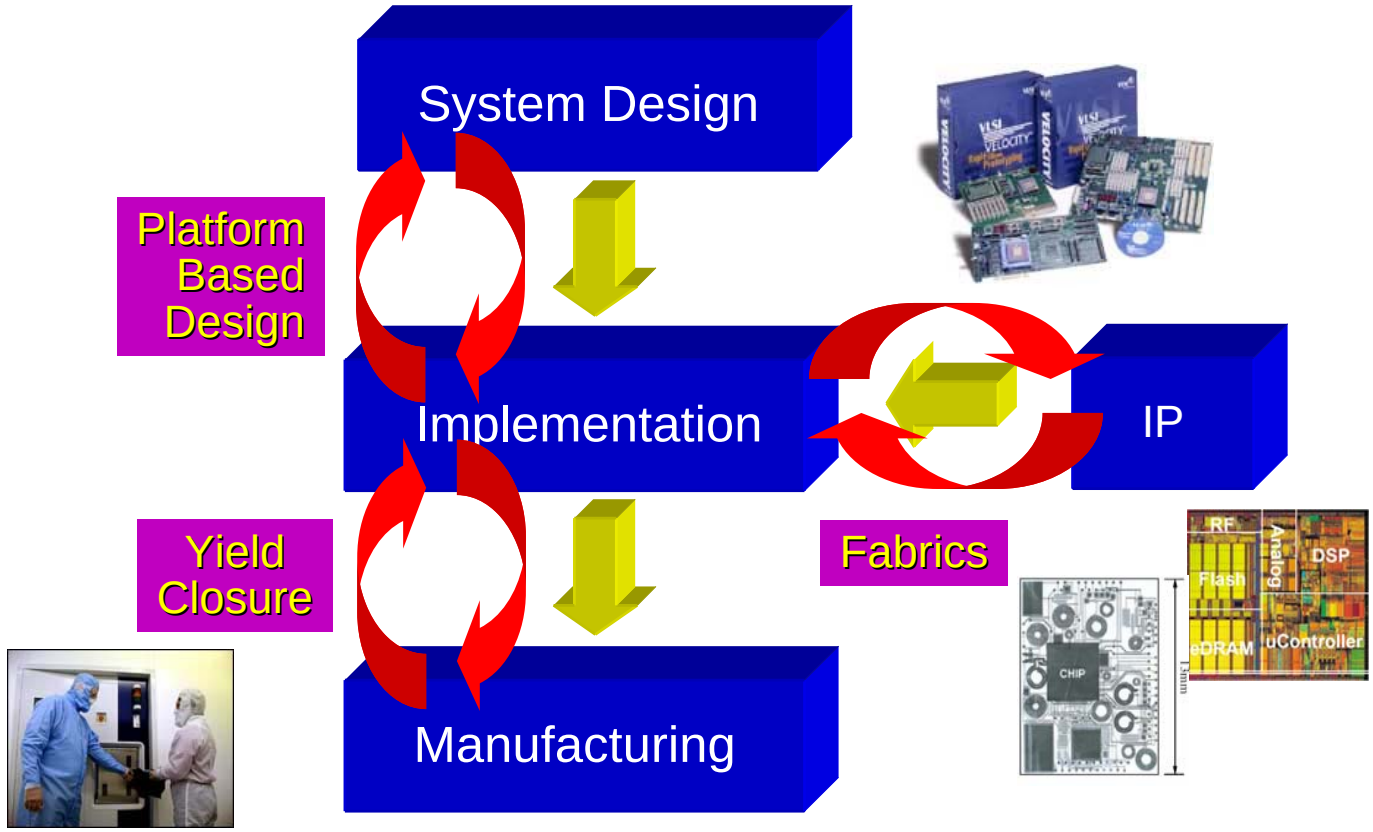
codesign environment

Contents

- Embedded System Design
- Design Methodology: Old and New
- Design Reuse: Platform Based Design (PBD)
- HW/SW Codesign
- Formal Model and CAD tools
- Other Design Issues – not to be covered in this course

Design Reuse Methodology

- **Reduce Verification overhead by using pre-verified design**
- **Reduce NRE (Non-Recurring Engineering) cost**



codesign environment

IP

- **A predefined, designed/verified, reusable building block for System-on-Chip**

- **IP needs**
  - Re-usable design
  - Interface Standard
  - Documentation
  - Simulation and/or verification
  - Protection

- **Classification**
  - Soft IP: RTL design, technology independent
  - Firm IP: gate-level netlist before Floor-planning
  - Hard IP: Polygon data, technology dependent
    - (ex) CPU, memory

IP Standard Organization(1)

- **OSCI**
  - Propose SystemC as the TLM level IP modeling language
  - IEEE 1666(2005)
- **Accellera**
  - VHDL International + Open Verilog International
  - HDL-based design standardization
  - Open Verification Library, Property Specification Language(PSL)
  - VHDL, verilogHDL, SystemVerilog
- **SPIRIT**
  - Stardardize IP interchange format at every abstraction level
  - Consortium of SoC, IP, EDA Industries (ST, Philips, ARM, Cadence, Mentor, Synopsys)

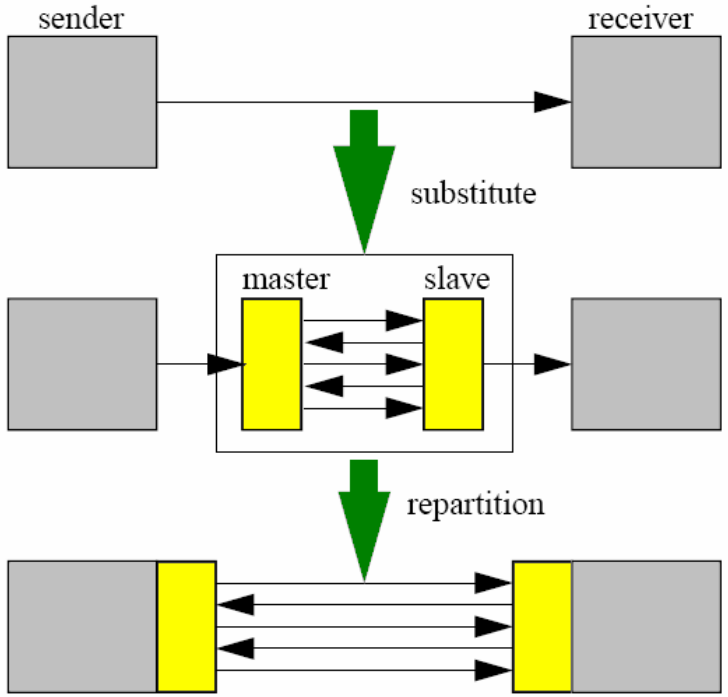
IP Standard Organization (2)

- VSI Alliance
- OCP-IP
- OMG (Object Management Group)
- D&R
- VCX
- OpenMORE
- OpenCores

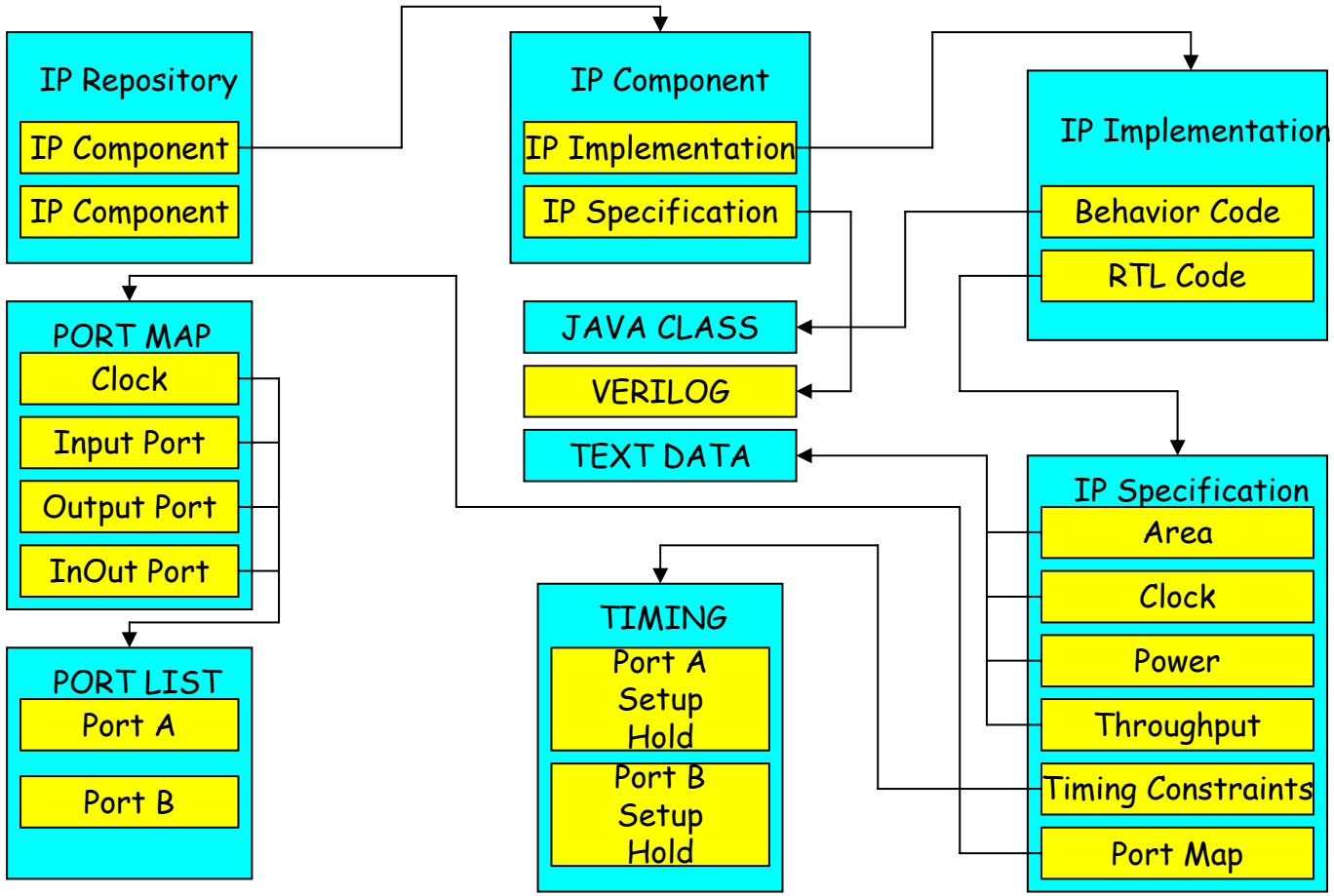
Homework: What are they doing for what?

Interface-based IP design

- **Concept**
  - Separate communication and behavior(computation)
  - Reuse computation and modify interface if necessary
- **Interface Synthesis: Communication Refinement**
  - Between HW and SW
    - Interrupt, Polling
  - Between SW and HW
    - I/O instr, registers



An Example of IP Specification



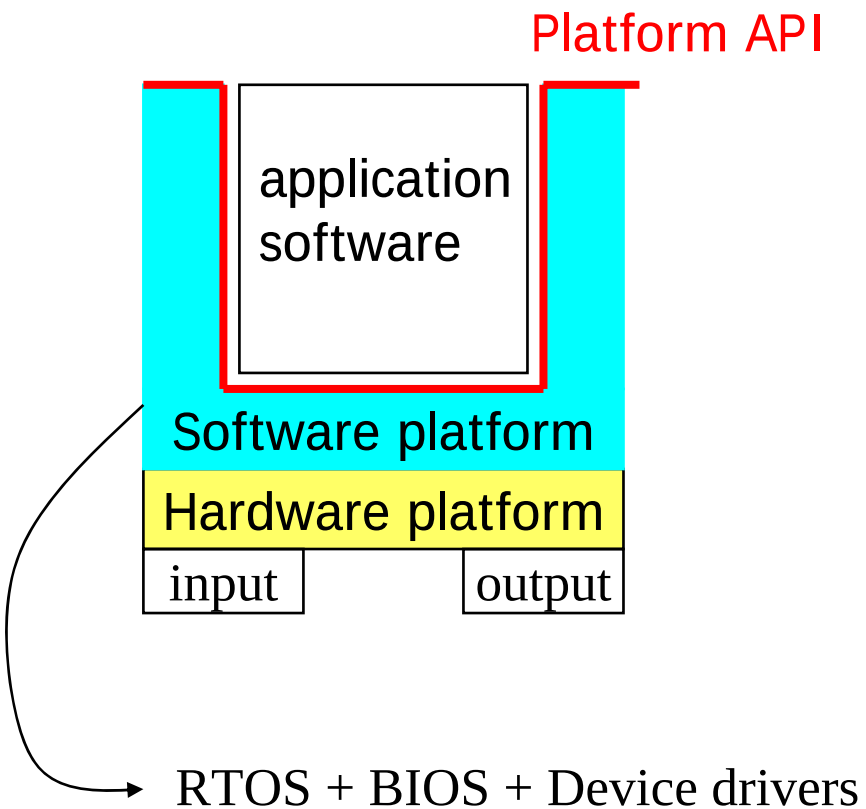
codesign environment

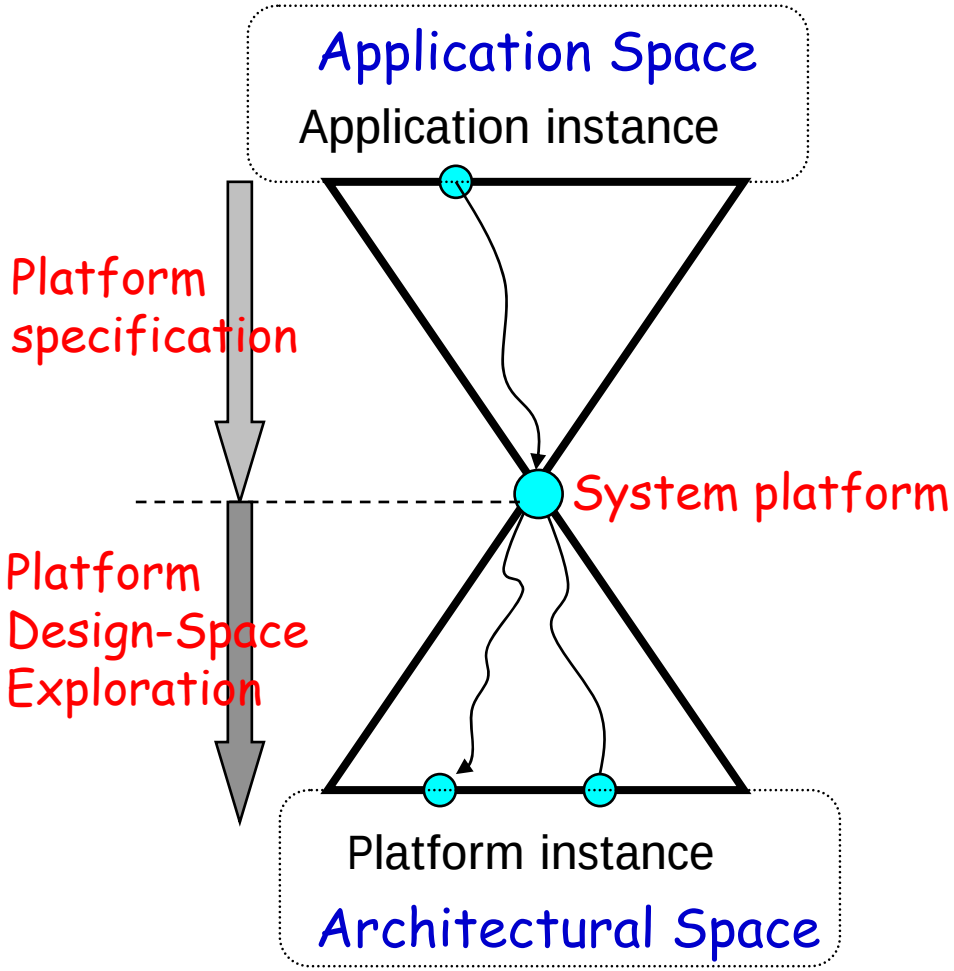
Platform-based Design

- **Need**
  - Technology push:
    - High-volume products;
    - feasible design.
  - Marketing push:
    - fast turnaround;
    - differentiated products.

- **Platform is the solution**
  - Reuse a pre-made architecture to a new application
    - Common denominator for a class of applications
  - Tradeoff between design time and design quality
  - Derivative design

Platform Based Design

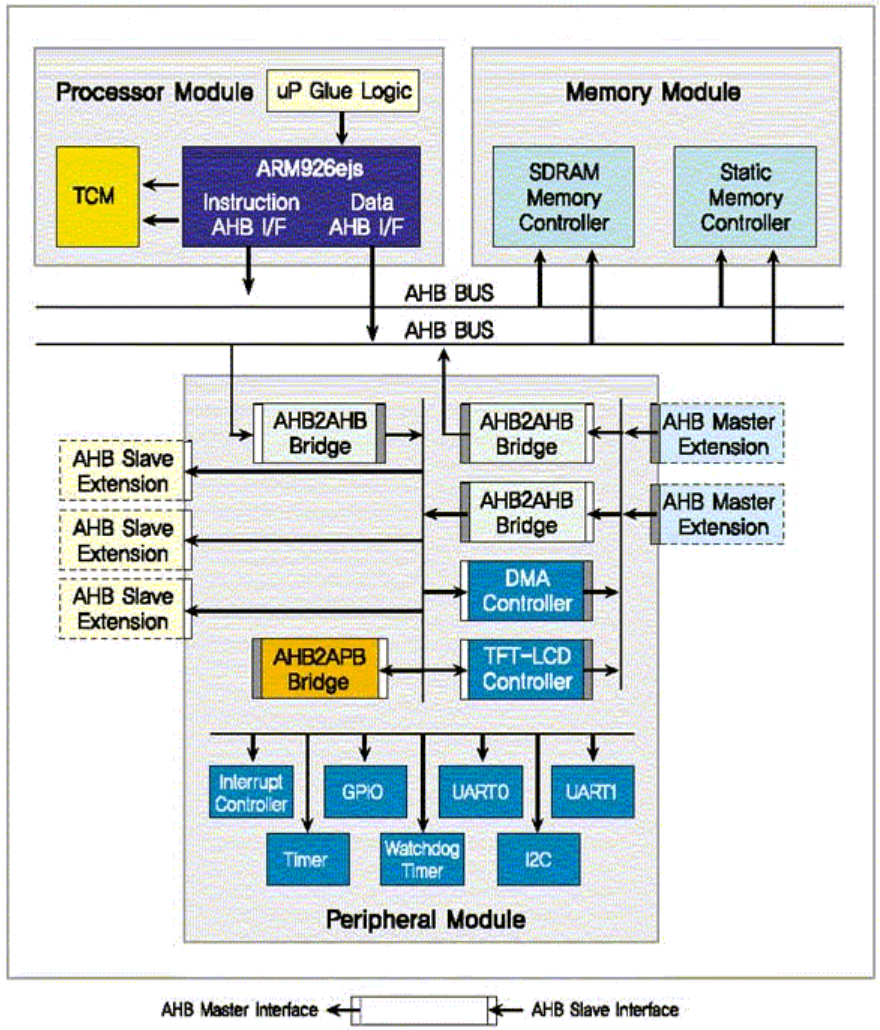




Platform is a partial design

- **A partial design:**
  - for a particular type of system;
  - includes embedded processor(s);
  - may include embedded software;
  - customizable to a customer's requirements:
    - software;
    - component changes.

SoCBase 1.0의 구조



Alternatives to Platform

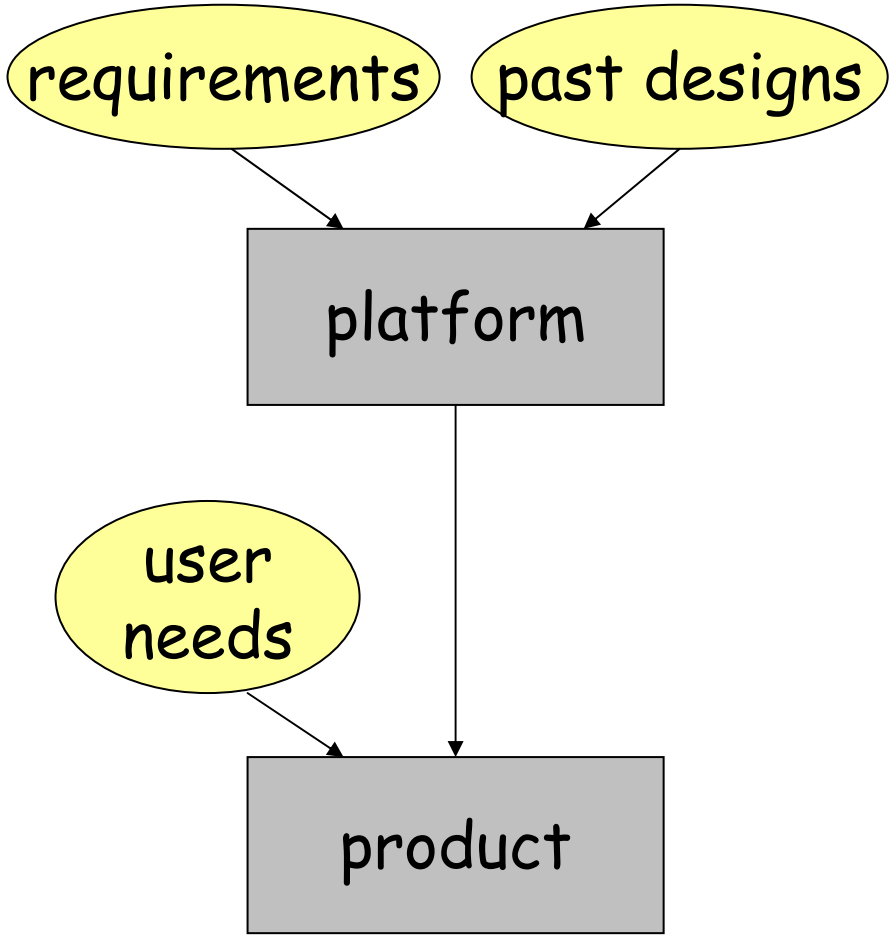
- **General Purpose Systems: Multiprocessor system**
  - May require much more area to accomplish the same task.
  - Often much less energy-efficient
- **Reconfigurable Systems: FPGA**
  - Good for parts of the system
  - Performance penalty
  - Viable as a future SoC architecture
- **Full Custom Design**
  - Extremely long design time
  - High engineering cost

Standards and Platforms

- **Many high-volume markets are standards-driven:**
  - wireless;
  - multimedia;
  - networking.
- **Standard defines the basic I/O requirements.**
- **Systems house chooses implementation of standards functions:**
  - improved quality, lower power, etc.
- **Product may be differentiated by added features:**
  - cell phone user interface.
- **Standards encourage platform-based design.**

Two Aspects of PBD

- **Platform Design**
  - Semiconductor company
  - Select the components and Design the architecture
- **Platform Use**
  - System house
  - Customize the platform
    - In-house
    - By contraction



Platform Design Methodology

- **Target the application area and size the problem**
  - For what applications?
  - How much horsepower? How much power?
- **Develop an initial architecture.**
- **Evaluate for performance, power, etc.**
- **Evaluate customizability.**
  - Is it sufficiently customizable? And in the right ways?
  - How long does it take to turn a platform into a product?
- **Improve platform after each use.**
- **Provide platform package**
  - HW platform + SW platform + Design methodology

Platform Use Methodology

- **Choose the right platform. How?**
  - Ideal: evaluate aspects of the platform critical to my product's requirements.
  - Base: marketing/sales decision.
- **Start with reference design, evaluate differences required for your features.**
- **Evaluate hardware changes.**
- **Implement hardware and software changes in parallel.**
- **Integrate and test.**

Platform Evaluation is the Key

- **Execute high-level models.**
  - Modeling Languages
    - SystemC (www.systemc.org).
    - SpecC (www.specc.org)
- **Co-simulation**
- **Run software on sample chip.**

SoC Platforms

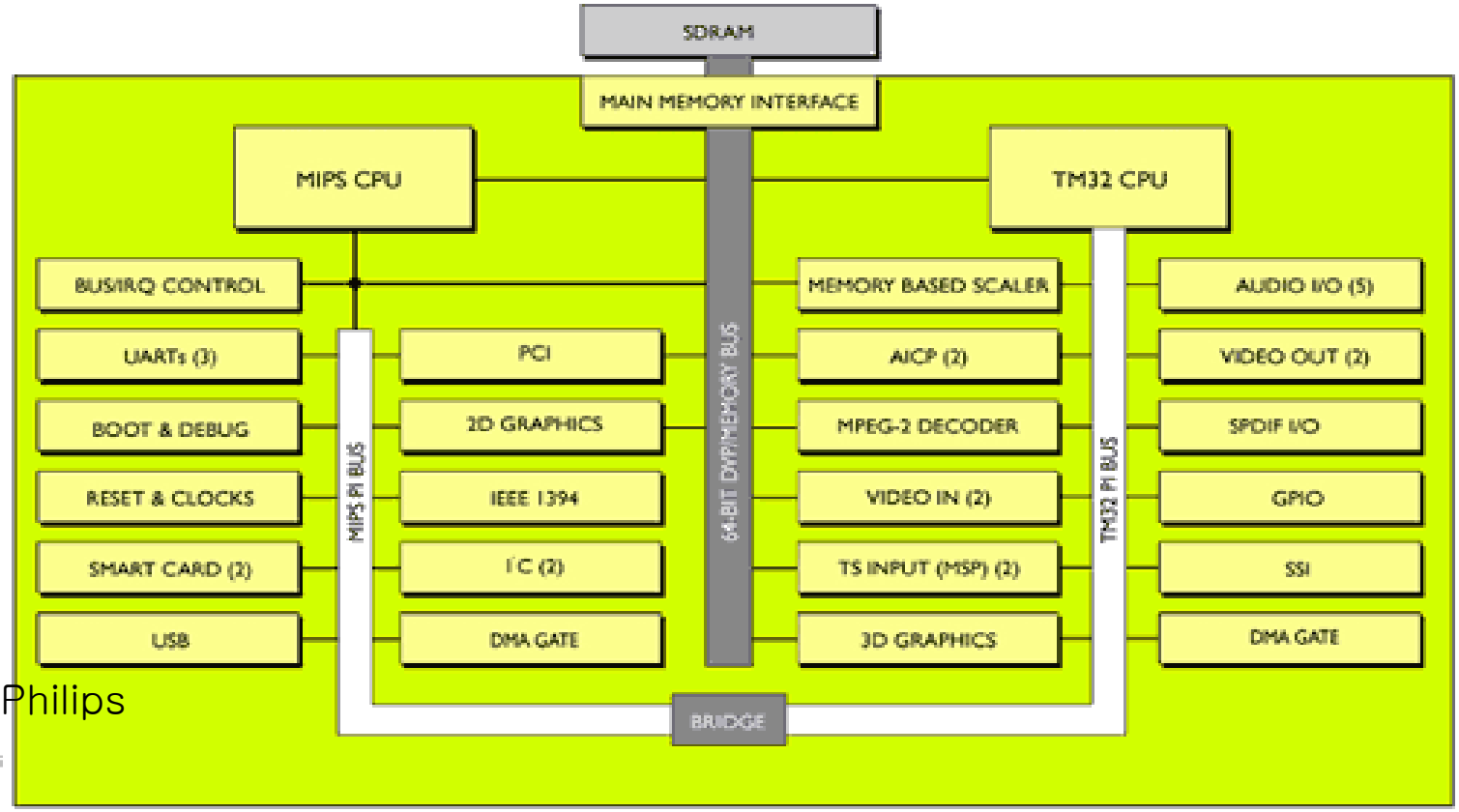
- **Full-application Platforms**
  - Philips Nexperia
  - TI OMAP (Open Multimedia Application Platform)
  - ARM's PrimeXsys
- **Communication-centric platform**
  - ARM AMBA bus architecture
  - Sonics uNetwork
  - IBM CoreConnect
- **Fully Programmable Platform**
  - Reconfigurable HW/processor: Xilinx Virtex II Pro, Triscend
  - Configurable microprocessor: ASIP (Application Specific Instruction-set Processor): Tensilica Extensa

D&R Classification

- **Processor Centric**
  - MIPS Technology - SOC-it® Platform
  - Intrinsix – AMBA IMD platform
  - Xilinx – PowerPC – based platform
  - NEC – SoCLiteTM
- **Interconnect Centric**
  - Sonics - SonicsStudioTM
  - OCP-IP CoreCreator®
  - …
- **Design Flow Centric**
  - MentorGraphics - Platform ExpressTM

Philips Nexperia

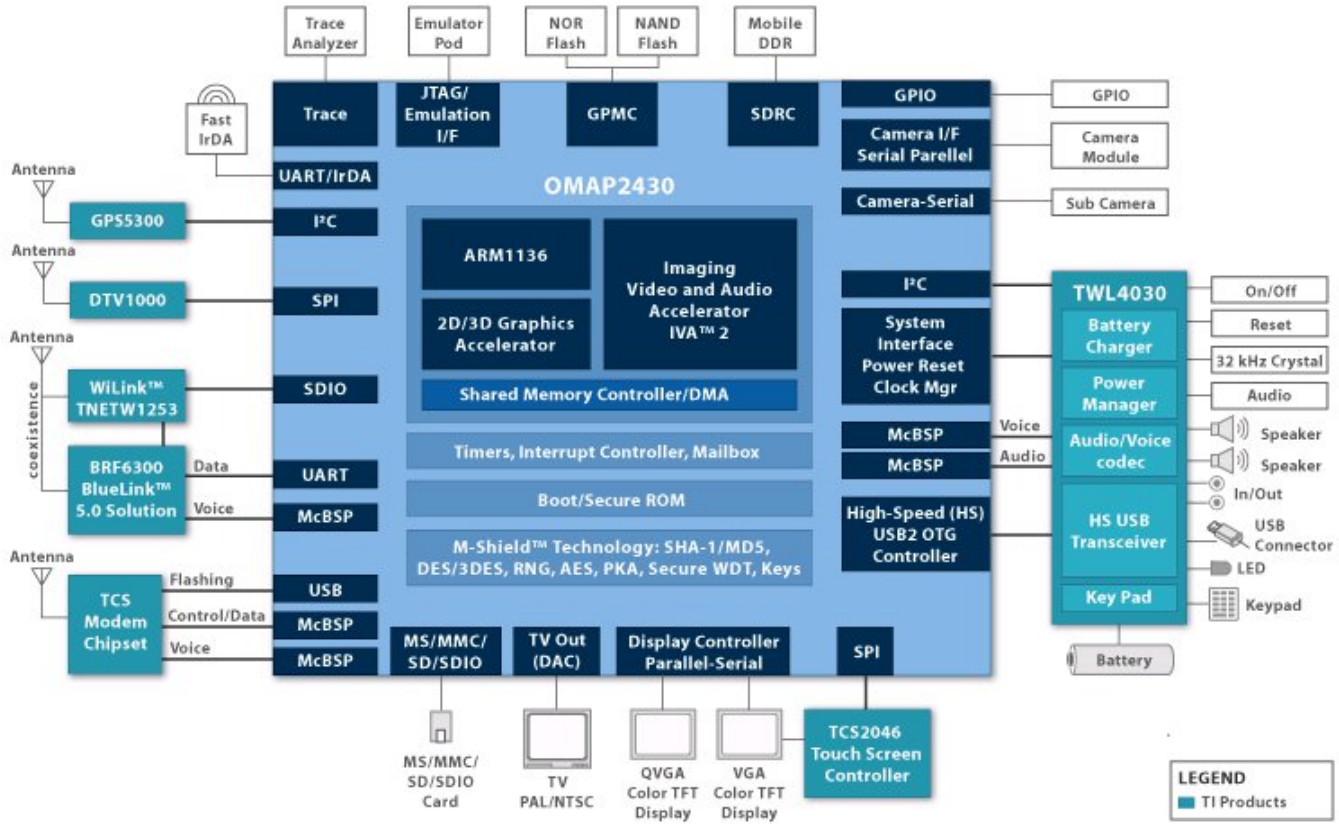
- **pns8500 home entertainment engine: "platform for new TV-based user services including web browsing, video e-mail, voice and video IP telephony, personal video recording/time-shifting and interactive digital TV (iDTV)"**



Source: ©Philips

TI OMAP

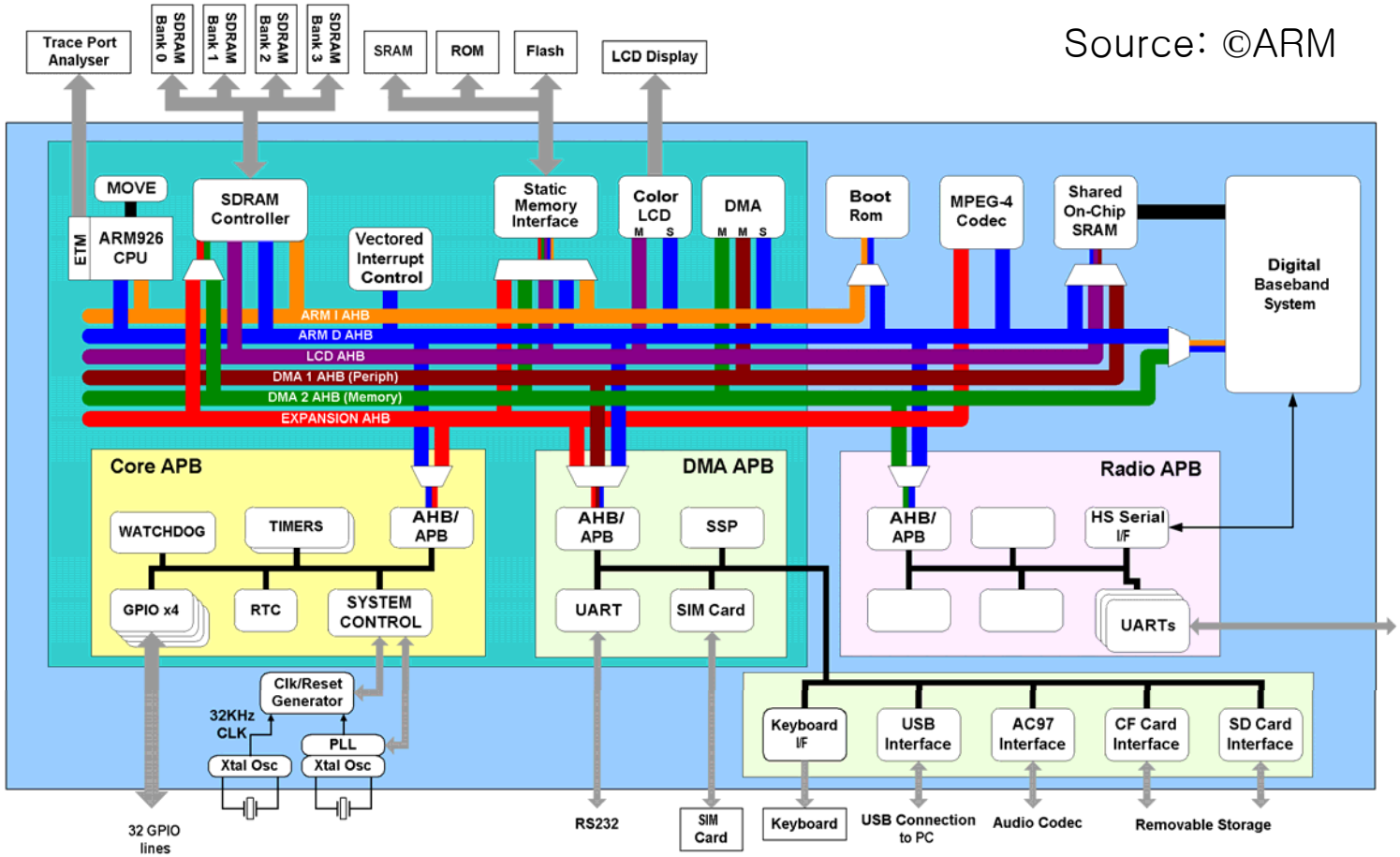
- **ARM-based platform**
- **Accelerators for Graphics and Video/Audio CODECs**



ARM PrimeXsys

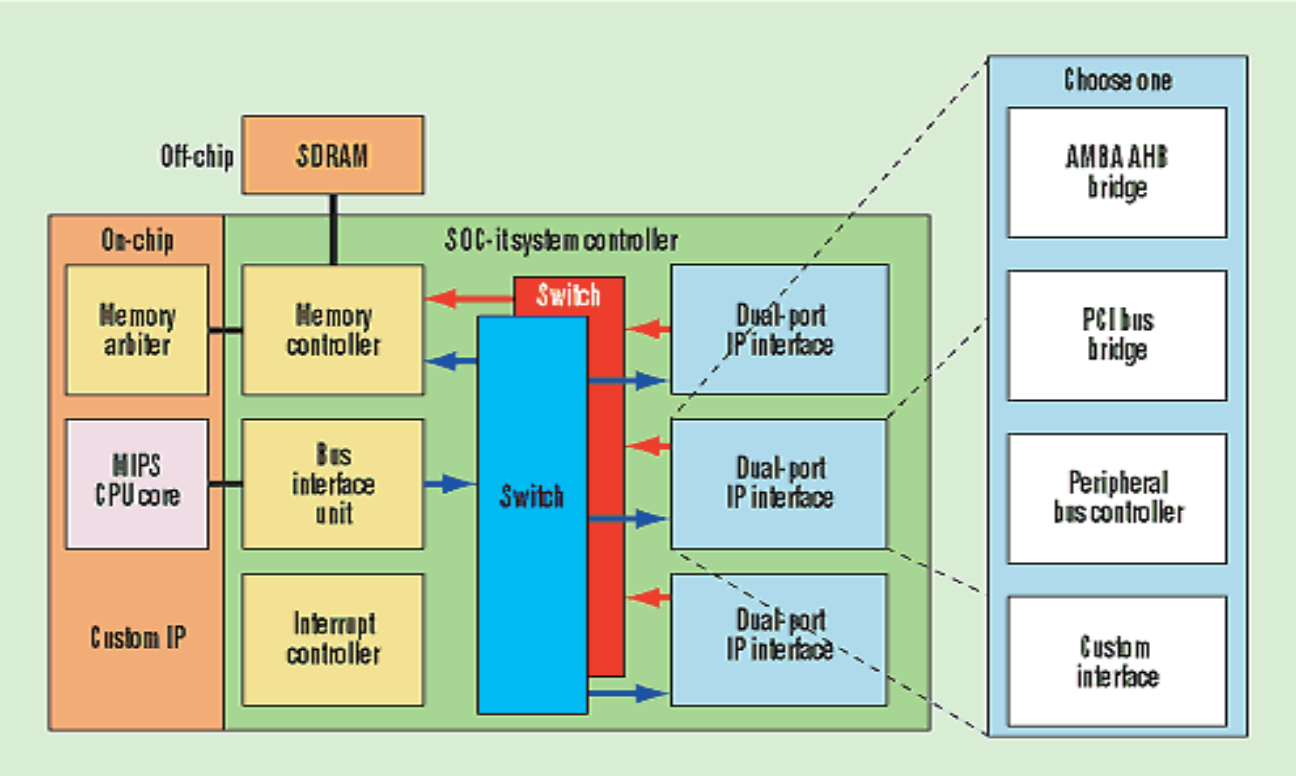
- **Wireless platform: Standard SoC Kernel based on ARM926EJ-S**

Source: ©ARM



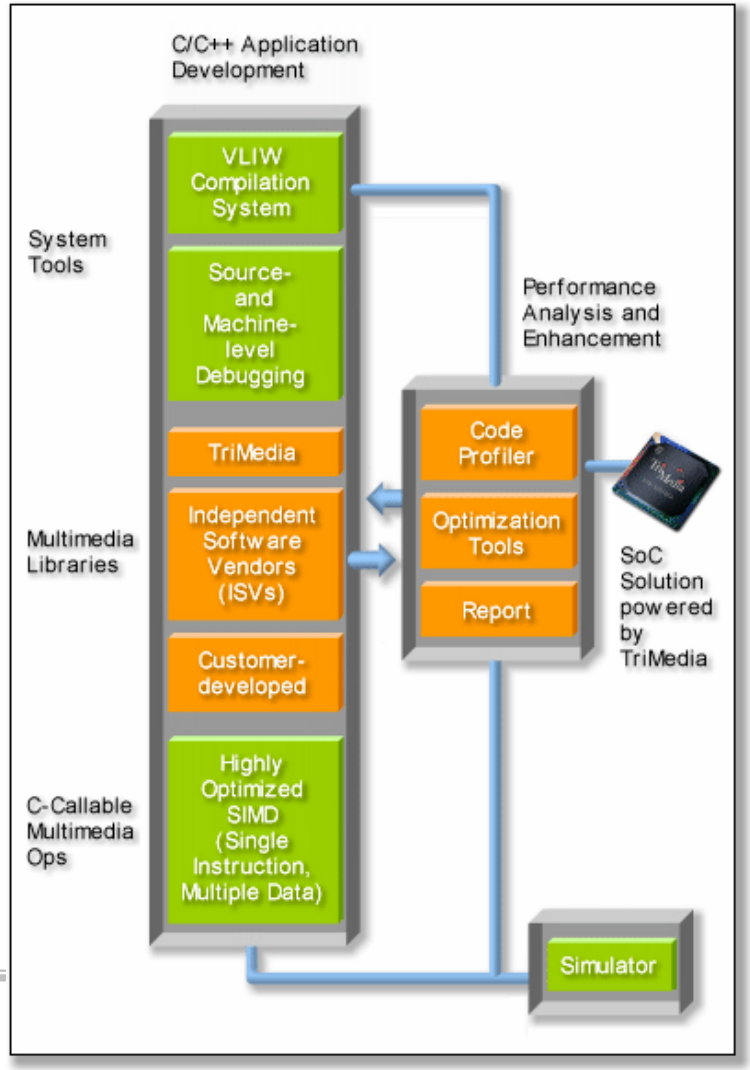
SoC-it®

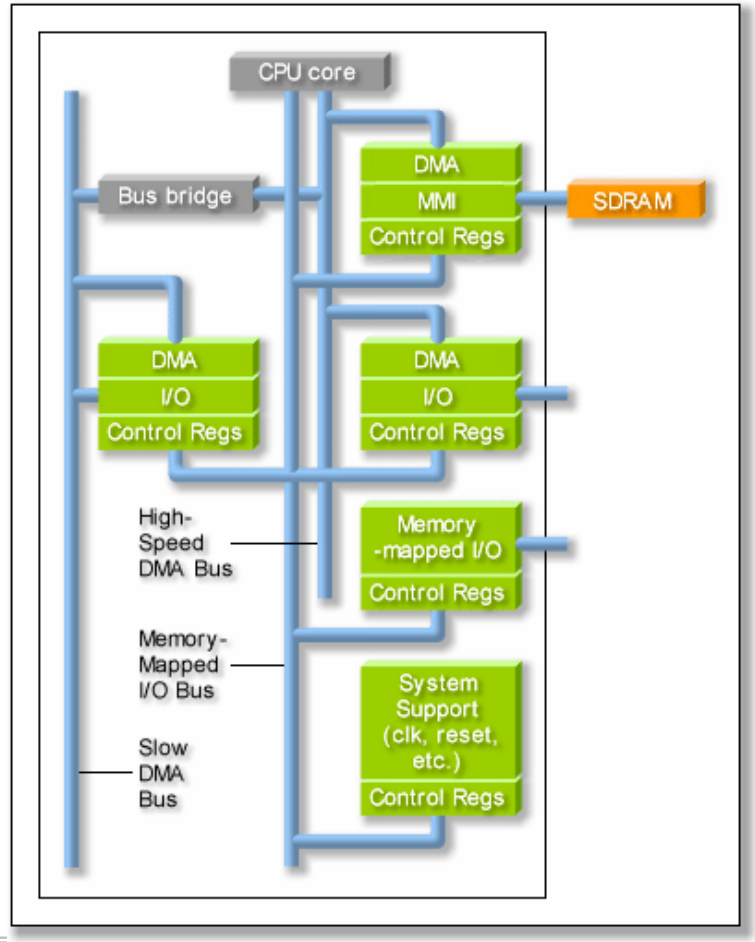
- **Platform is a partial design**
  - Provide IP interface for IP integration



Trimedia

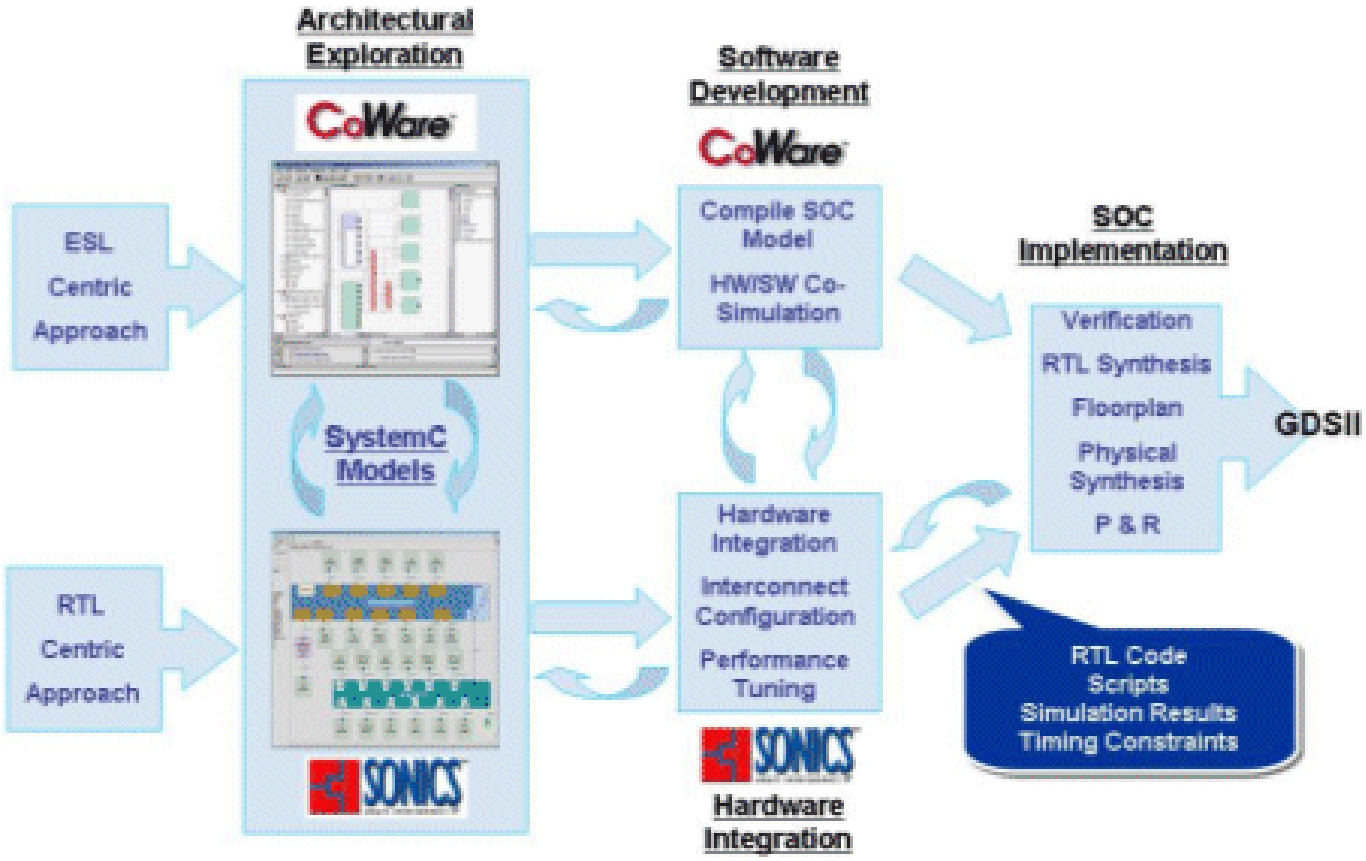
- **Hardware platform + Design environment**





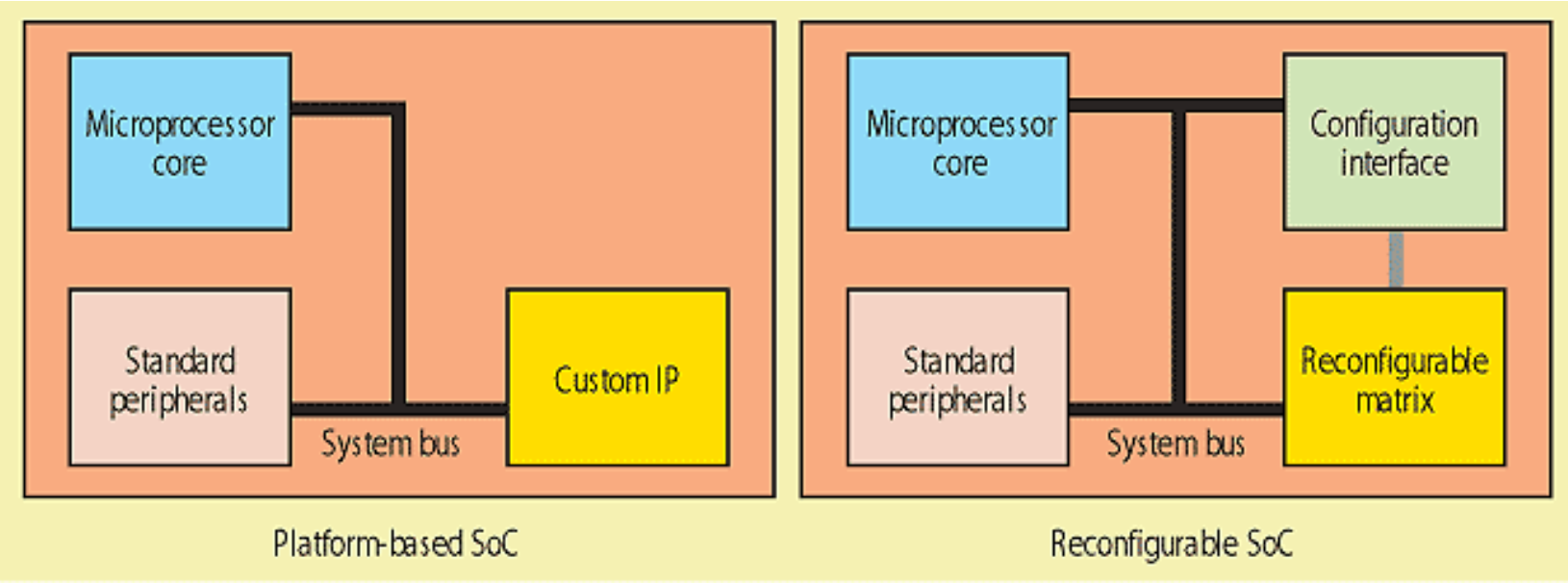
SonicsStudioTM

- **Communication centric platform**
- **Commnication refinement by CoWare ConvergenSC**



Reconfigurable Hardware

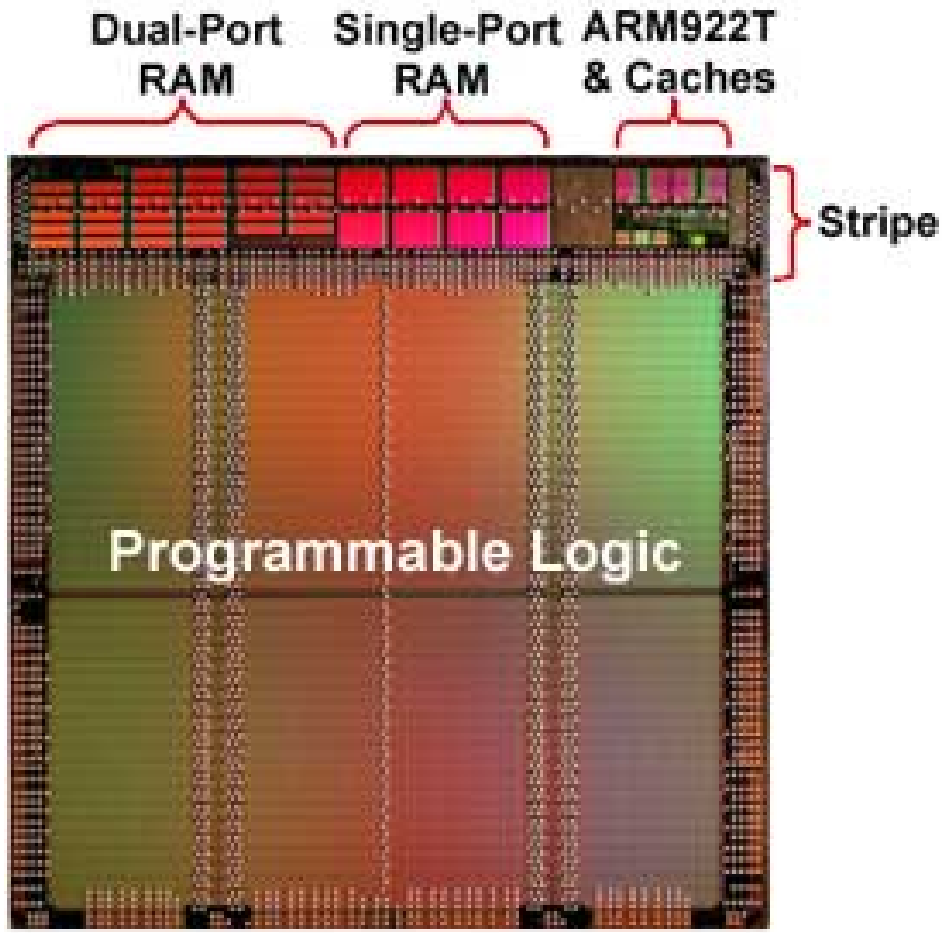
- **Reconfigurable Processor**
  - Reconfigurable HW as a coprocessor
  - Dynamic reconfiguration time is important
  - (ex) Triscend A7 CSoC, Chameleon Systems



FPGA Evolution

- **Hardware processor core inside the FPGA**

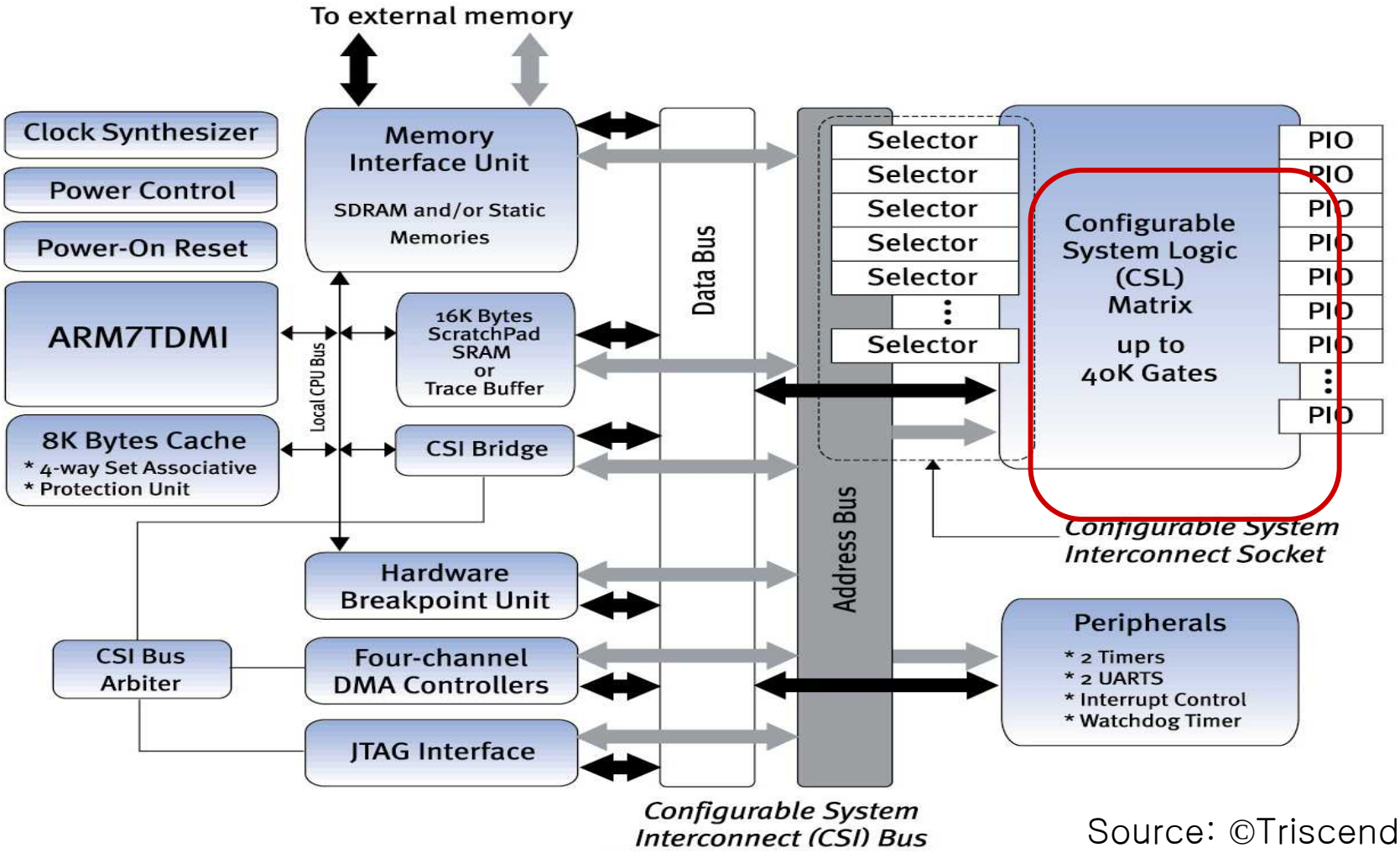
Altera Excalibur
EPXA10



Source: ©Altera

Triscend A7 CSoC

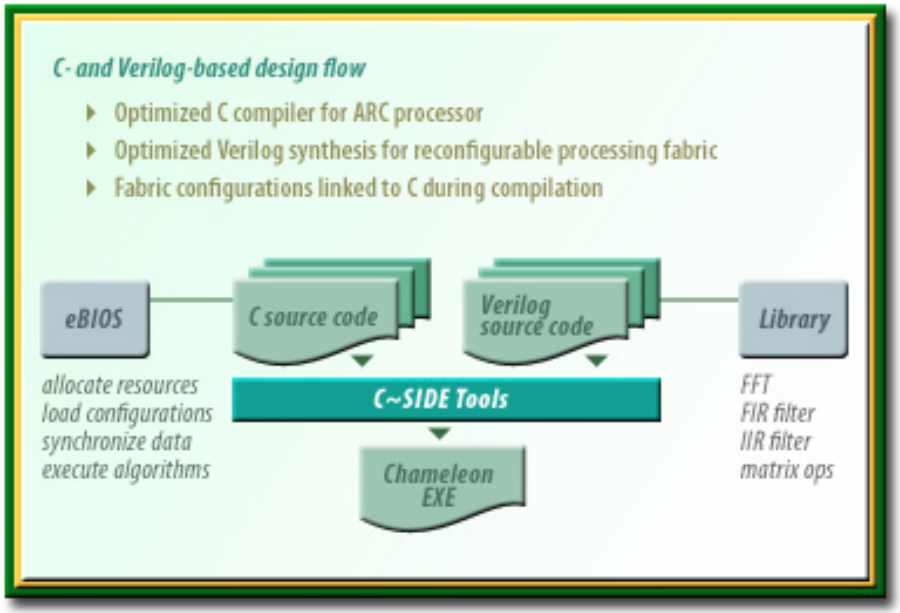
- **ARM7TDMI + FPGA**

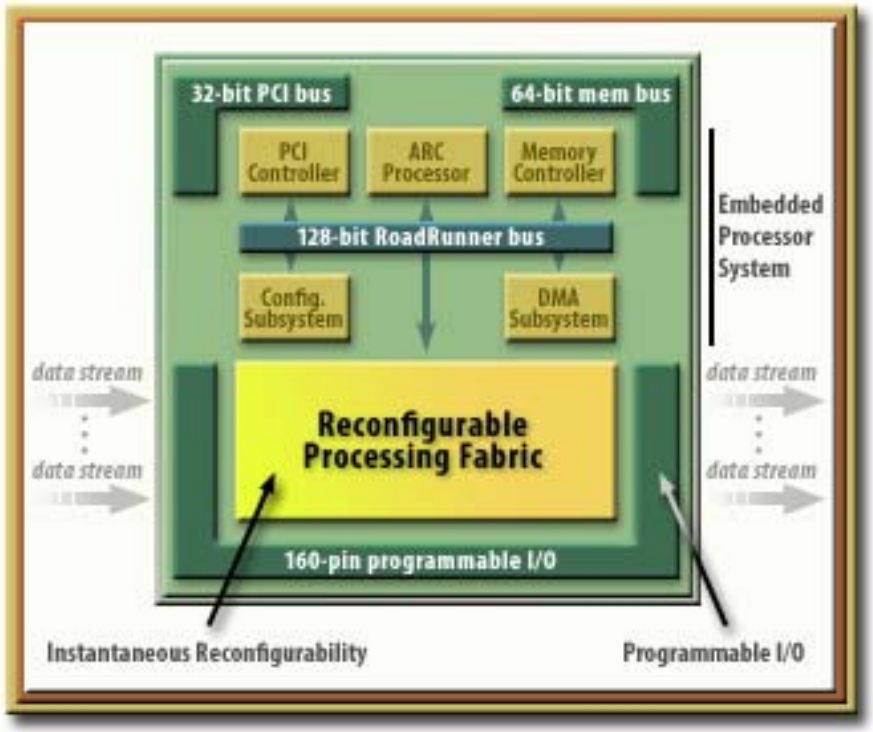


Source: ©Triscend

Chameleon

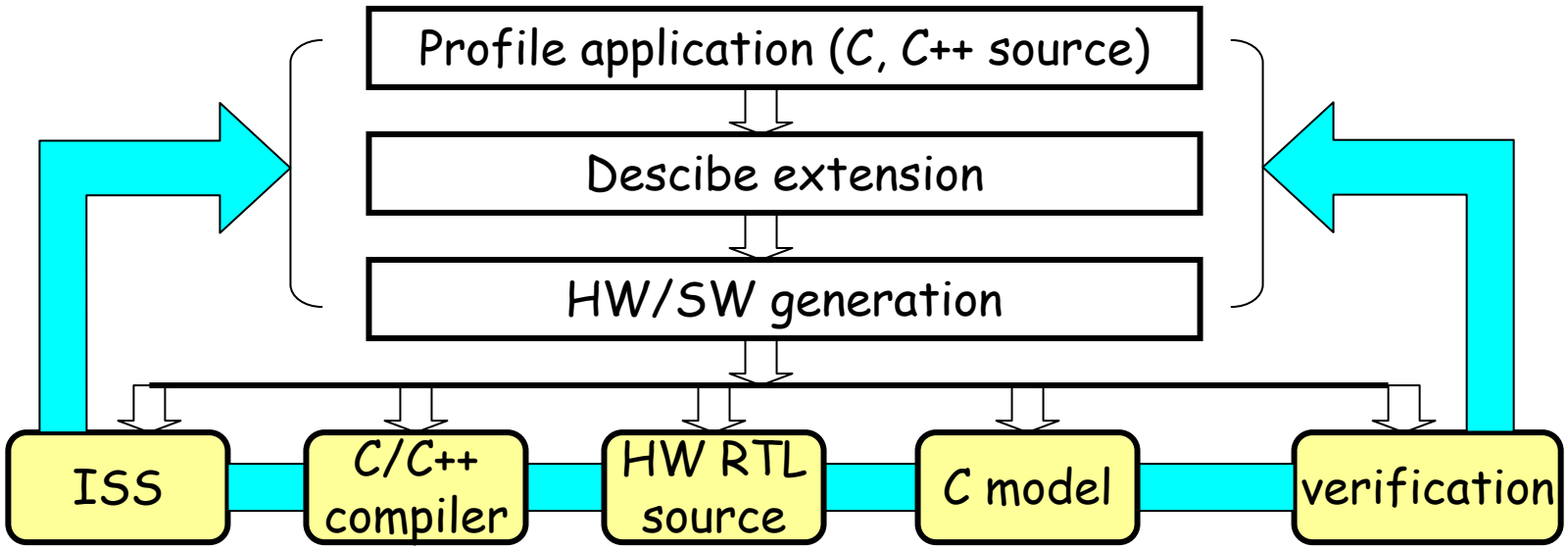
- **C/verilog –based reconfigurable platform**





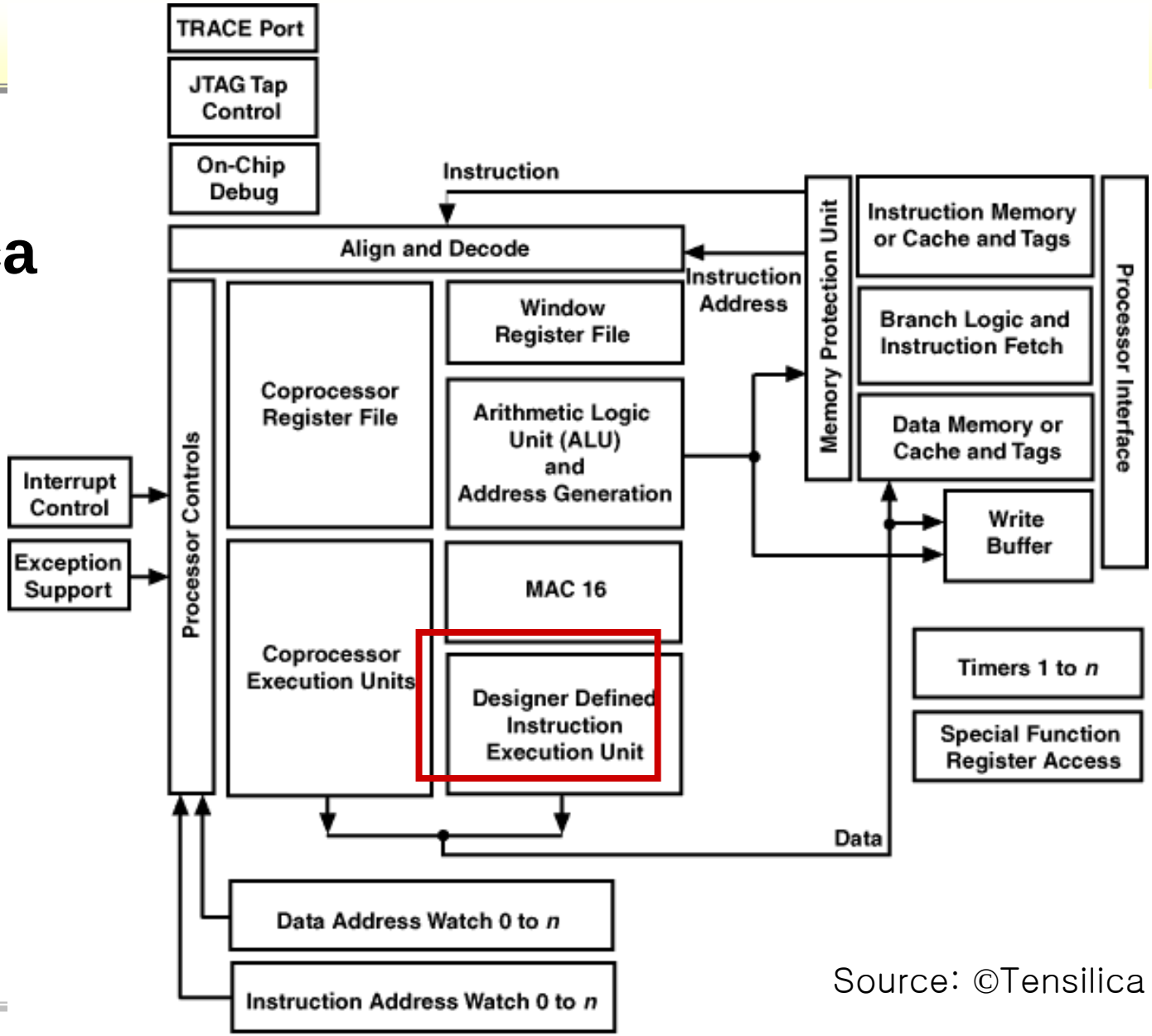
ASIP: reconfigurable microprocessor

- **Application Specific Instruction Set processor**
  - Configure processor pipeline
  - Generate complete software development environment



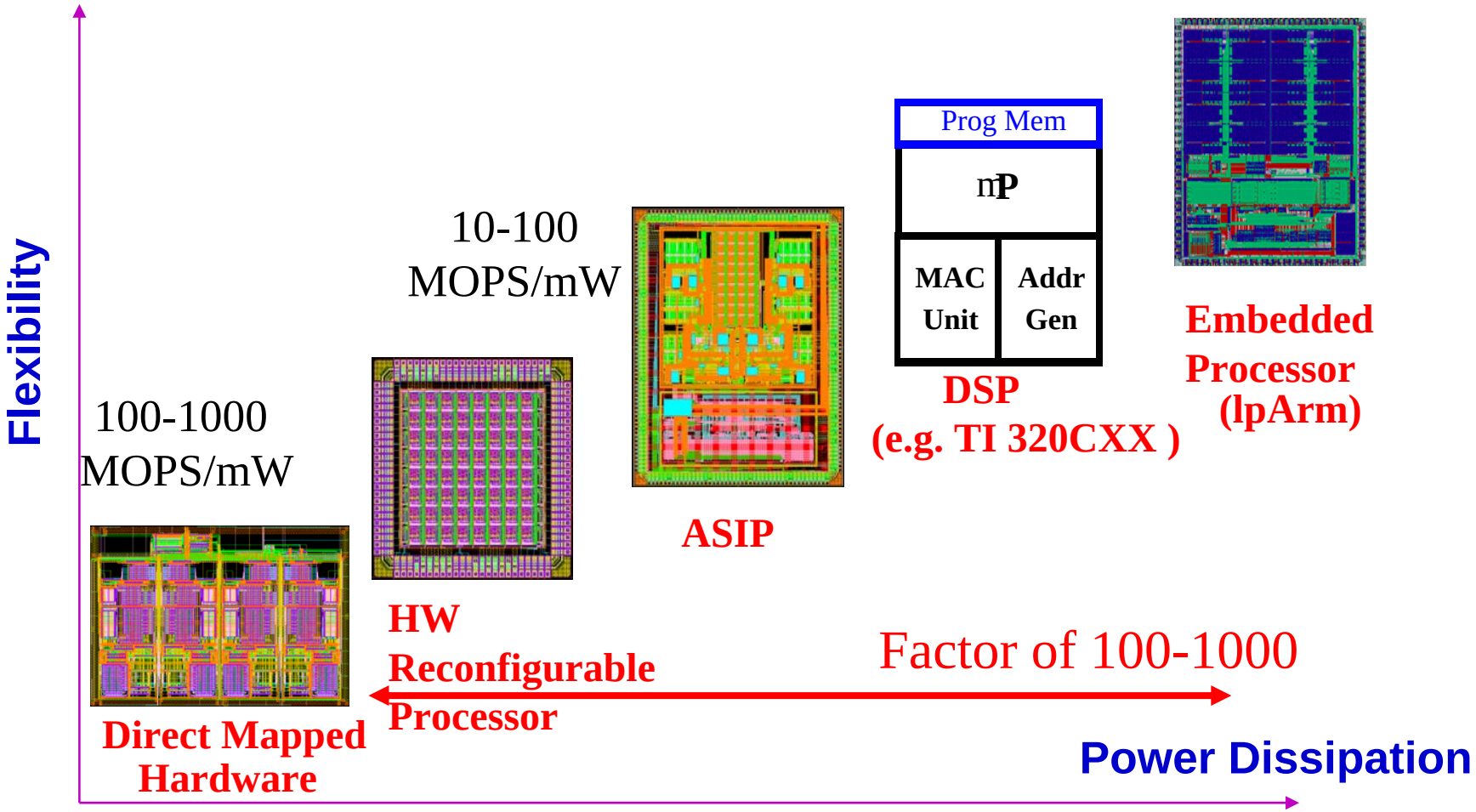
Source: ©Tensilica

Tensilica Xtensa



Source: ©Tensilica

Choose the Right Architecture



Source: ©Berkeley Wireless Research Center

Platform and IP-based Design

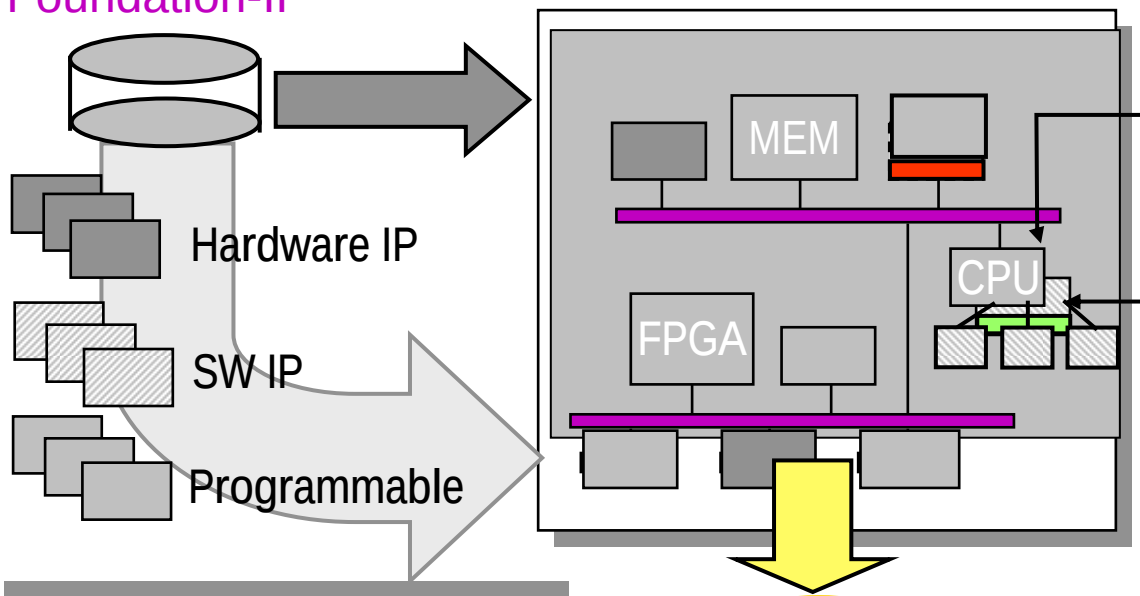
- **Platforms use IP: want to reuse as many hardware components as possible:**
  - CPUs;
  - memories;
  - I/O devices.
- **Want to use software libraries where possible.**
- **RTOS simplifies design of multi-tasking systems.**

- **Platforms are IP at the next level of abstraction.**

Summary of PBD

Pre-Qualified/Verified Foundation-IP*

Foundation Block + Reference Design



Scaleable bus, test, power, IO, clock, timing architectures

Processor(s), RTOS(es) and SW architecture

*IP can be hardware (digital or analogue) or software. IP can be hard, soft or 'firm' (HW), source or object (SW)

Foundry-Specific Pre-Qualification

Source@cadence

Methodology / Flows:

System-level performance evaluation environment

Rapid Prototype for End-Customer Evaluation

SoC Derivative Design Methodologies

Foundry Targetting Flow

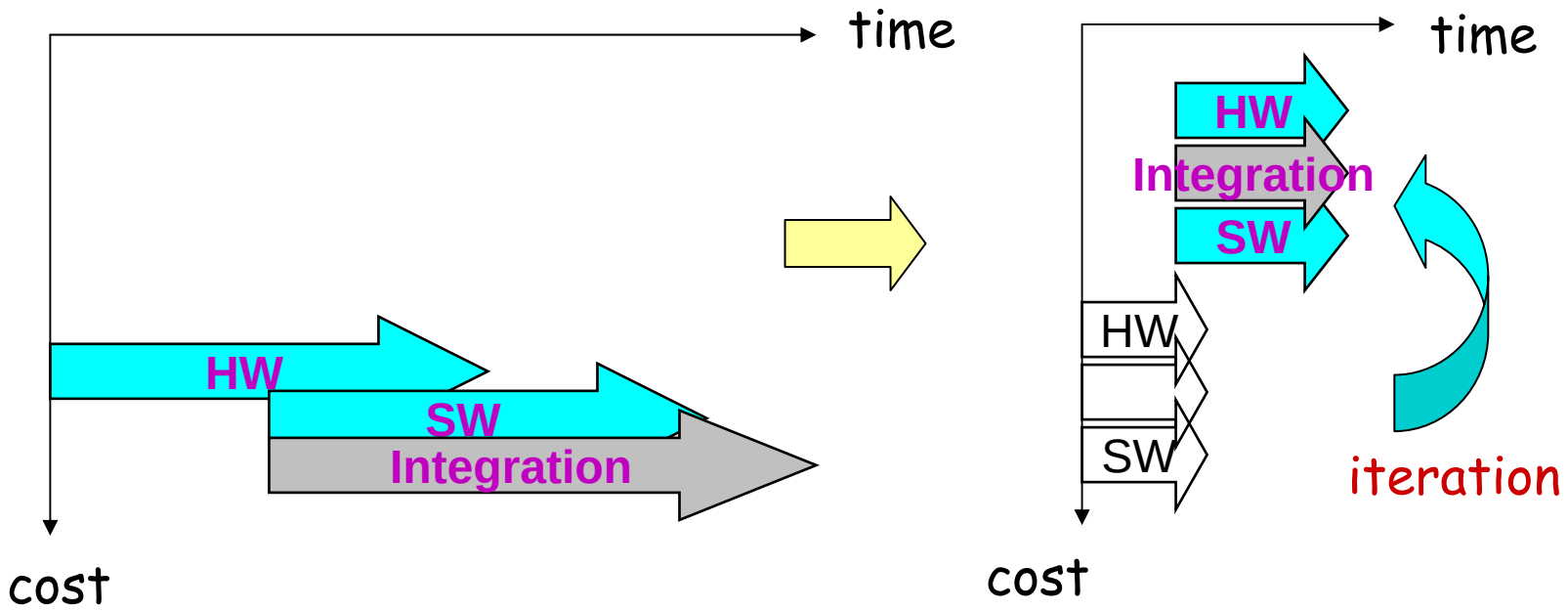
codesign environment

Contents

- Embedded System Design
- Design Methodology: Old and New
- Design Reuse: Platform Based Design (PBD)
- HW/SW Codesign
- Formal Model and CAD tools
- Other Design Issues – not to be covered in this course

HW/SW Codesign

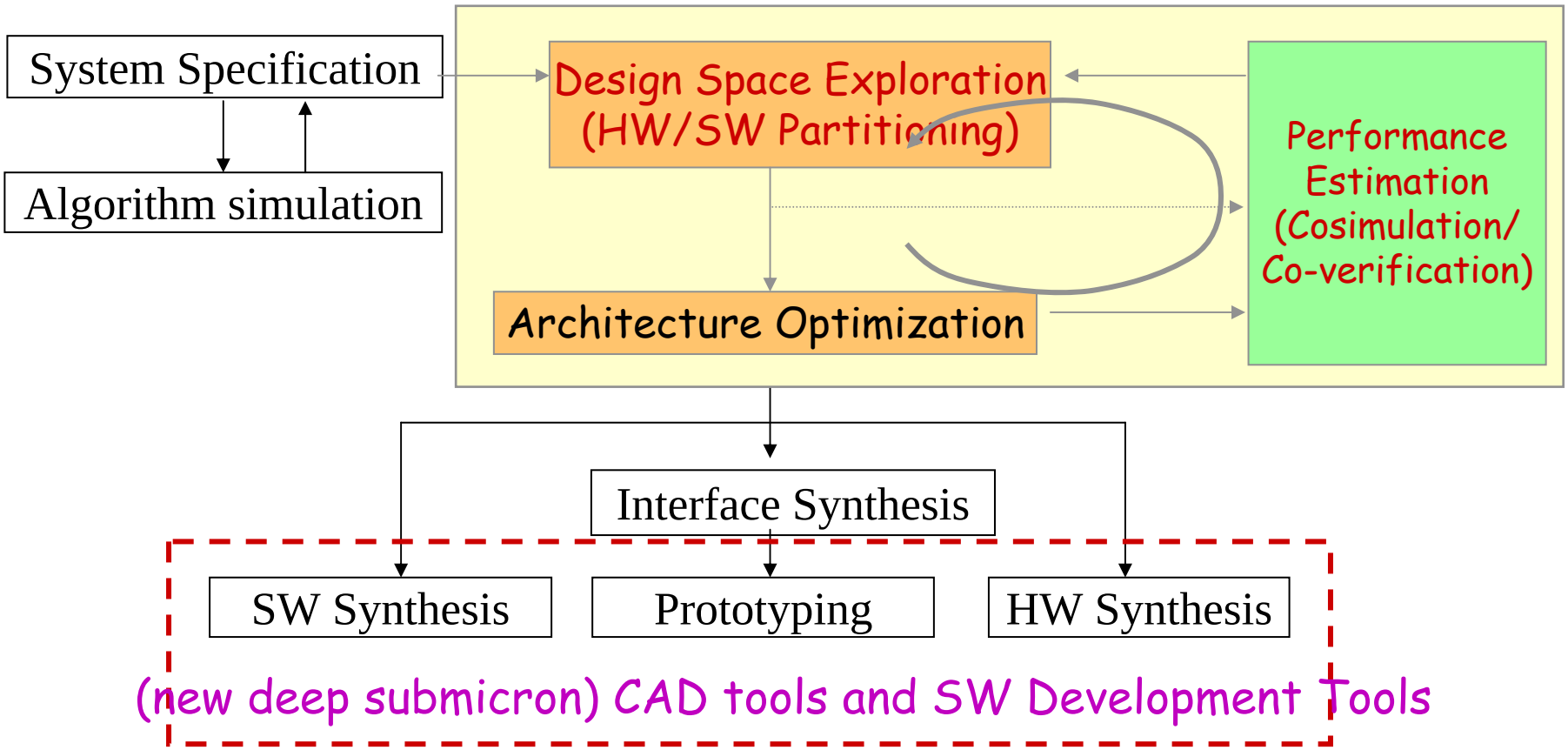
- Concurrent design of HW/SW components
- Evaluate the effect of a design decision at early stage by "virtual prototyping" to enable systematic design space exploration



Benefits of HW/SW Codesign

- **A significant performance improvement for embedded system design**
  - Earlier architecture closure
  - Reducing risk by 80%
- **HW/SW engineering groups to talk together**
- **Earlier HW/SW Integration**
- **Reducing design cycle**
  - Developing HW/SW in parallel

HW/SW Codesign Flow

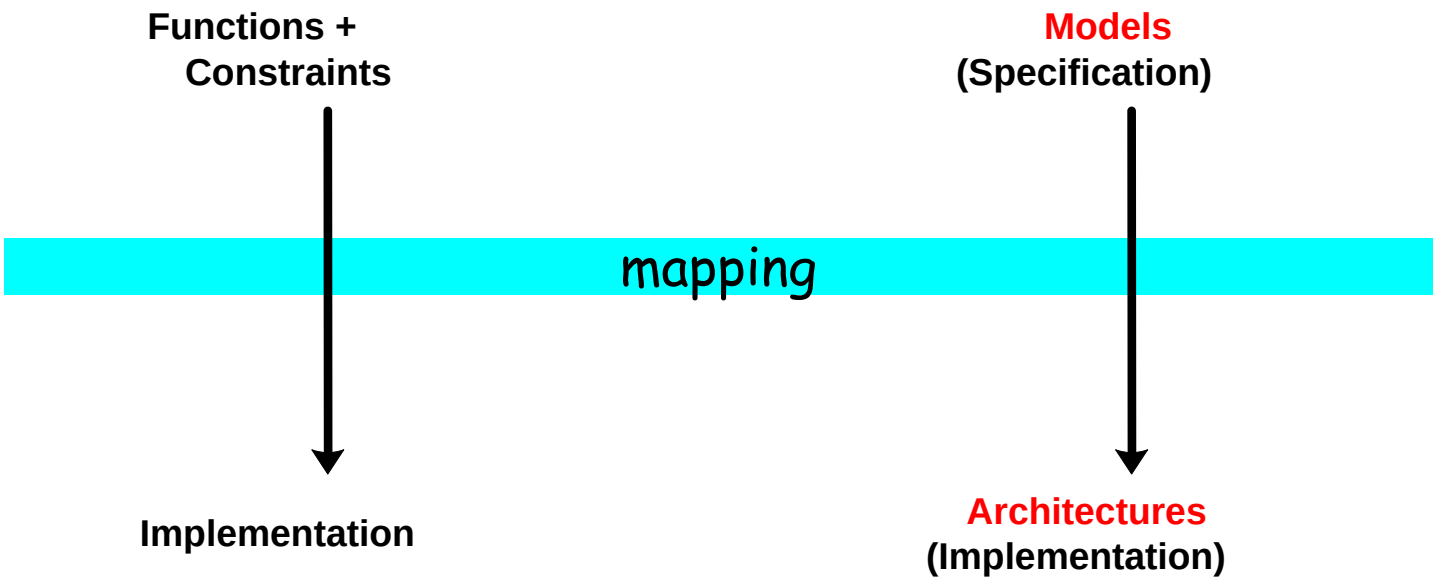


4 Main Issues

- **System level specification: System modeling**
- **Systematic design space exploration (DSE)**
- **HW/SW Co-simulation**
  - For design space exploration
  - For SW development
  - For HW/SW coverification
- **HW/SW/Interface Synthesis**

(Issue 1) System Modeling

- **Need**
  - demand of higher level abstraction to cope with complexity



Models **are conceptual views of the system's functionality**

Architectures **are abstract views of the system's Implementation**

Behavior & Architecture Model

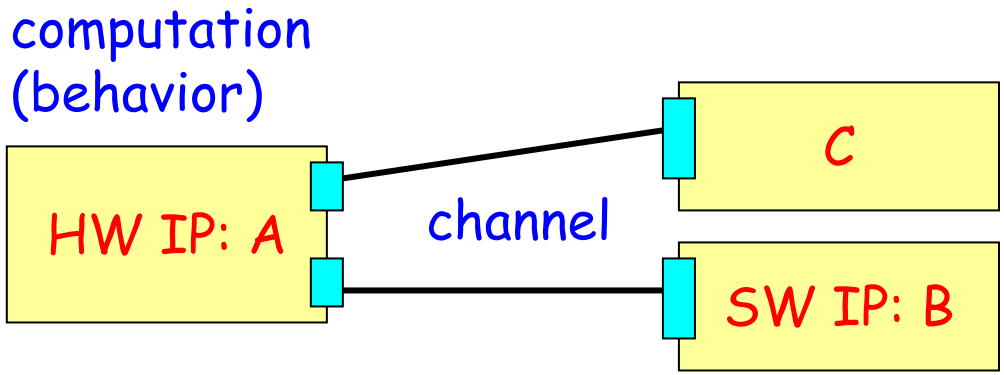
- **Behavior modeling**
  - Implementation (hardware or software) independent
  - Hardware/software partitioning
  - System-level functional verification
  - Refinement from functional simulation to implementation
  - Modeling languages
    - HW: HDL (Hardware Description Languages: VHDL, Verilog)
    - SW: C, C++, Java
    - System level: SystemC, SpecC, Esterel, Dataflow

- **Architecture modeling**
  - High-level system (SoC) architecture description
  - Extensible, reconfigurable
  - Performance estimation by simulation

Behavior Specification

- **Language-based specification**
  - Algorithm specification in a programming language, C or C++.
  - Good for functional simulation
  - Not good for synthesis

- **Model-based specification**
  - Algorithm specification with a proper model that represents concurrency naturally.



Models for Behavior Specification

- **State-oriented models**
  - Finite-State Machines (FSM), statechart, Esterel
- **Object-oriented model**
  - UML
- **Activity-oriented models**
  - Dataflow model, Discrete-Event model
  - SystemC model
  - SIMULINK model
- **Process Networks models**
  - CSP, Kahn process networks, Dataflow process network
- **Heterogeneous models**
  - Ptolemy, Metropolis, PeaCE
- **Others**
  - Synchronous-Reactive model

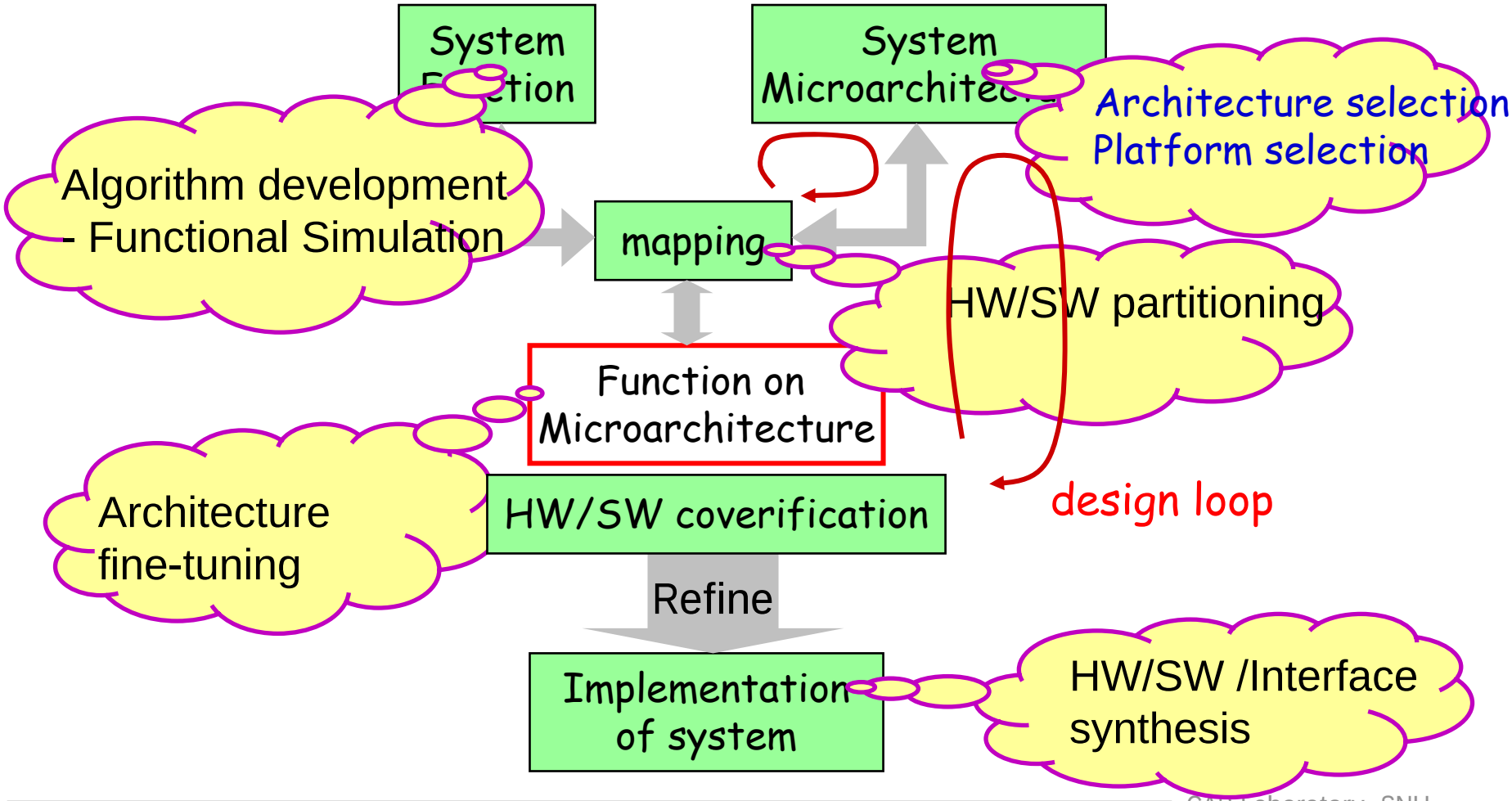
(Issue 2) Design Space Exploration

- **Problem Statement**
  - Find (an) optimal architecture(s) for the given system functions to satisfy the requirement performance with the minimum overhead (cost, power, etc).

- **Design Space**
  - Component selection: processing elements such as CPU, DSP, HW IP
  - HW/SW partitioning
  - Communication/memory architecture

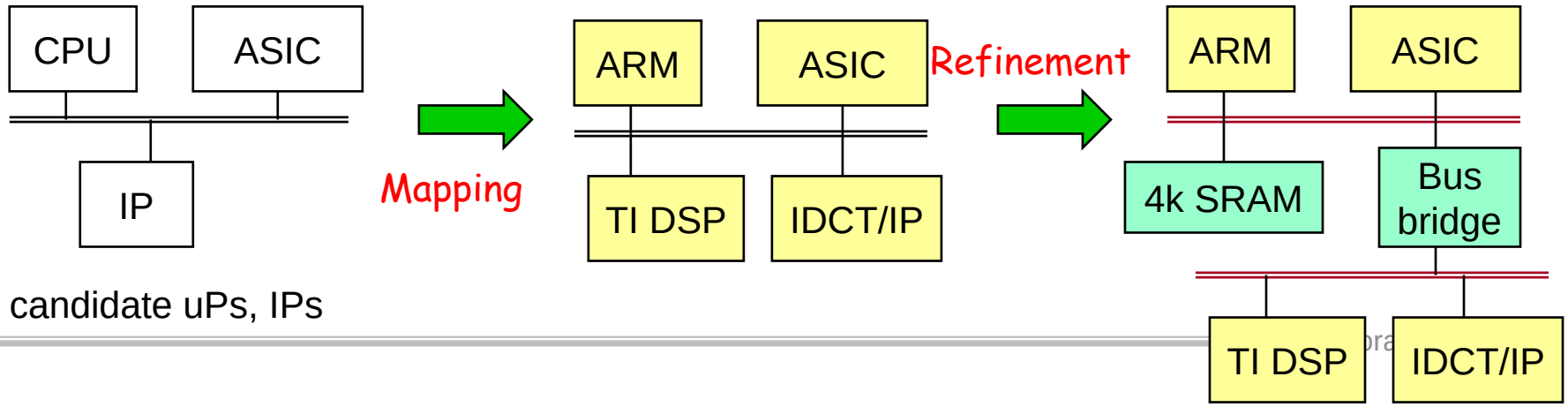
DSE by Mapping

- **Y-chart approach**



DSE Approaches

- **Current practice: Manual approach**
  - Mentor Platform Express, CoWare ConvergenSC
  - Drag-and-drop platform selection ← design library (architecture IP)
  - Manual mapping
  - Manual design refinement + communication synthesis
  - Fast performance evaluation: abstract modeling, virtual processor
- **Automated approach (on Research)**
  - Partitioning: component selection and mapping
  - Architecture refinement: memory and channel refinement

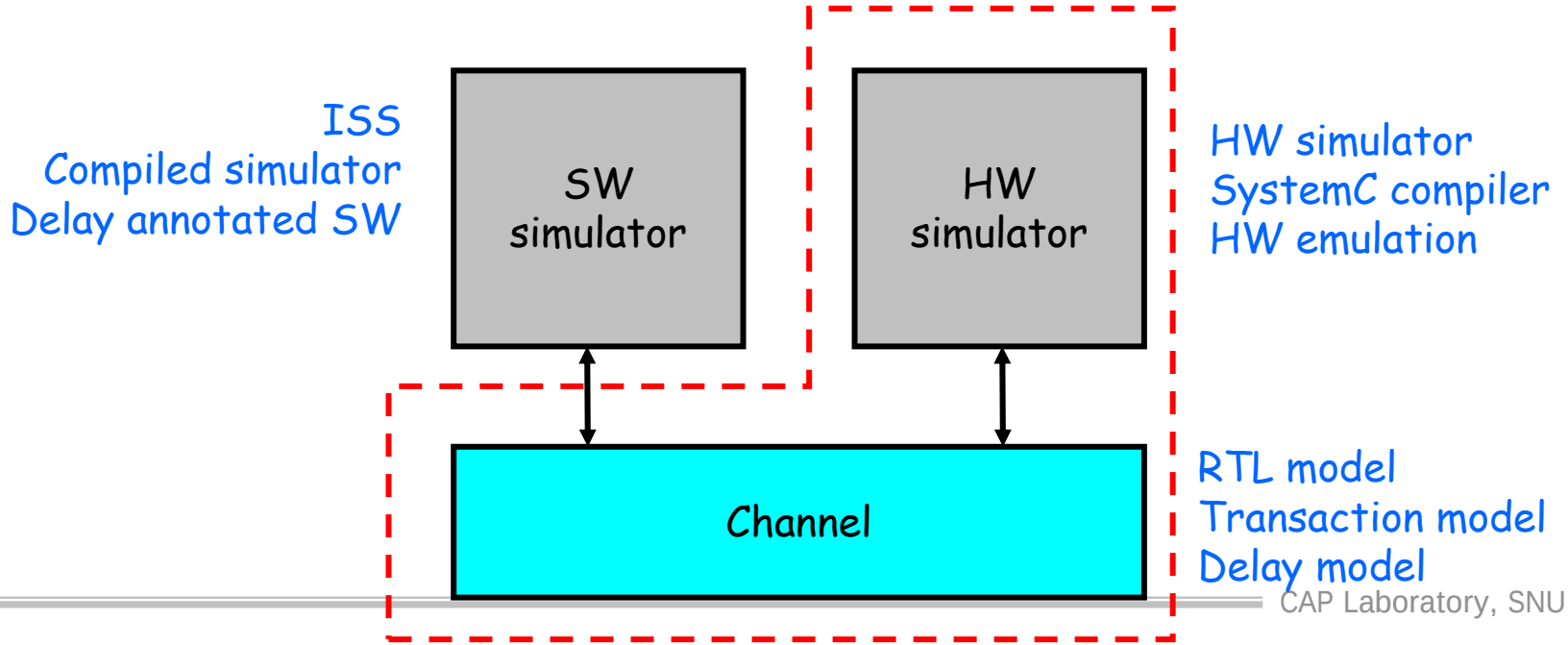


Key Issues in DSE

- **Performance estimation of function blocks on each processing element**
- **Performance estimation of partitioning decision**
  - Is HW/SW cosimulation fast enough?
- **Partitioning algorithm: multi-tasking applications with diverse types of parallelism**
  - Under resource constraints
  - Real-time requirements
- **Communication architecture exploration**

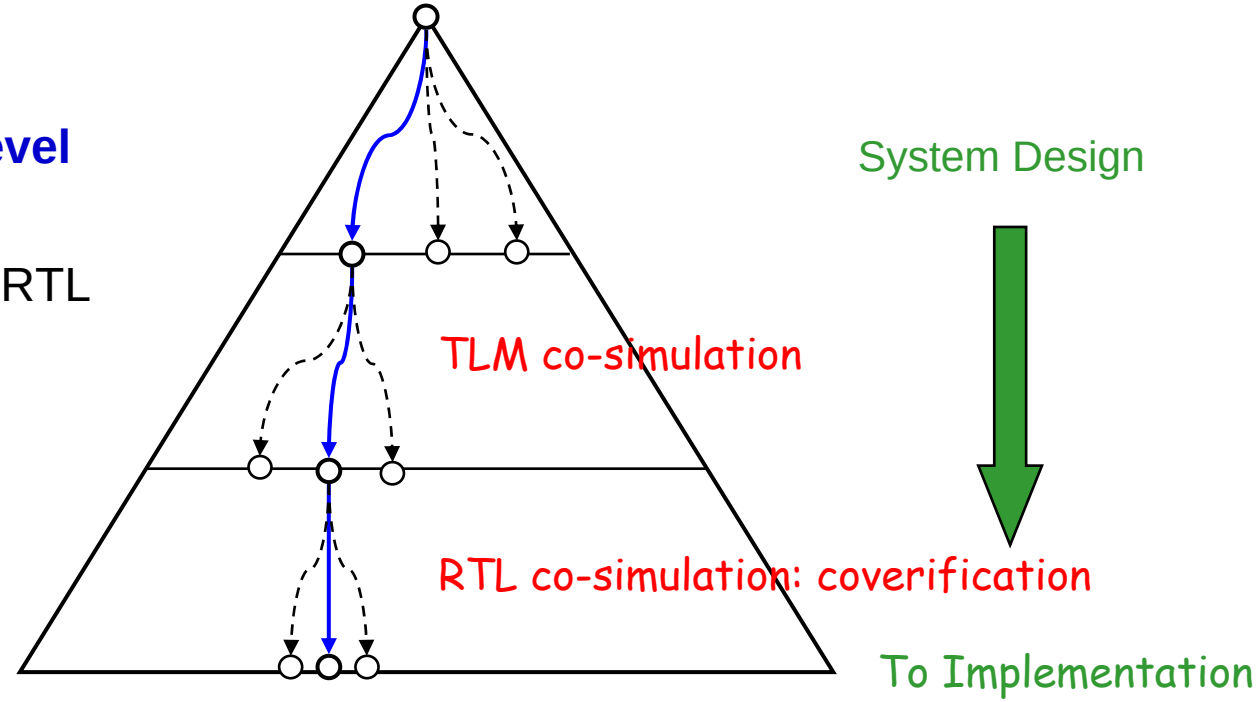
(Issue 3) HW/SW Cosimulatrion

- **Problem statement**
  - After HW/SW partitioning decision is made, estimate the sytem performance by co-simulating the software part (with processor simulator) and the hardware part.
  - Tradeoff between timing-accuracy and co-simulation speed
  - Synchronization overhead for timed co-simulation



TLM and RTL Cosimulation

- **RTL (Register-Transfer Level)**
  - Pin-accurate model
  - Too slow
- **TLM (Transaction-Level Model)**
  - Any model above RTL



Key Issues in HW/SW Cosimulation

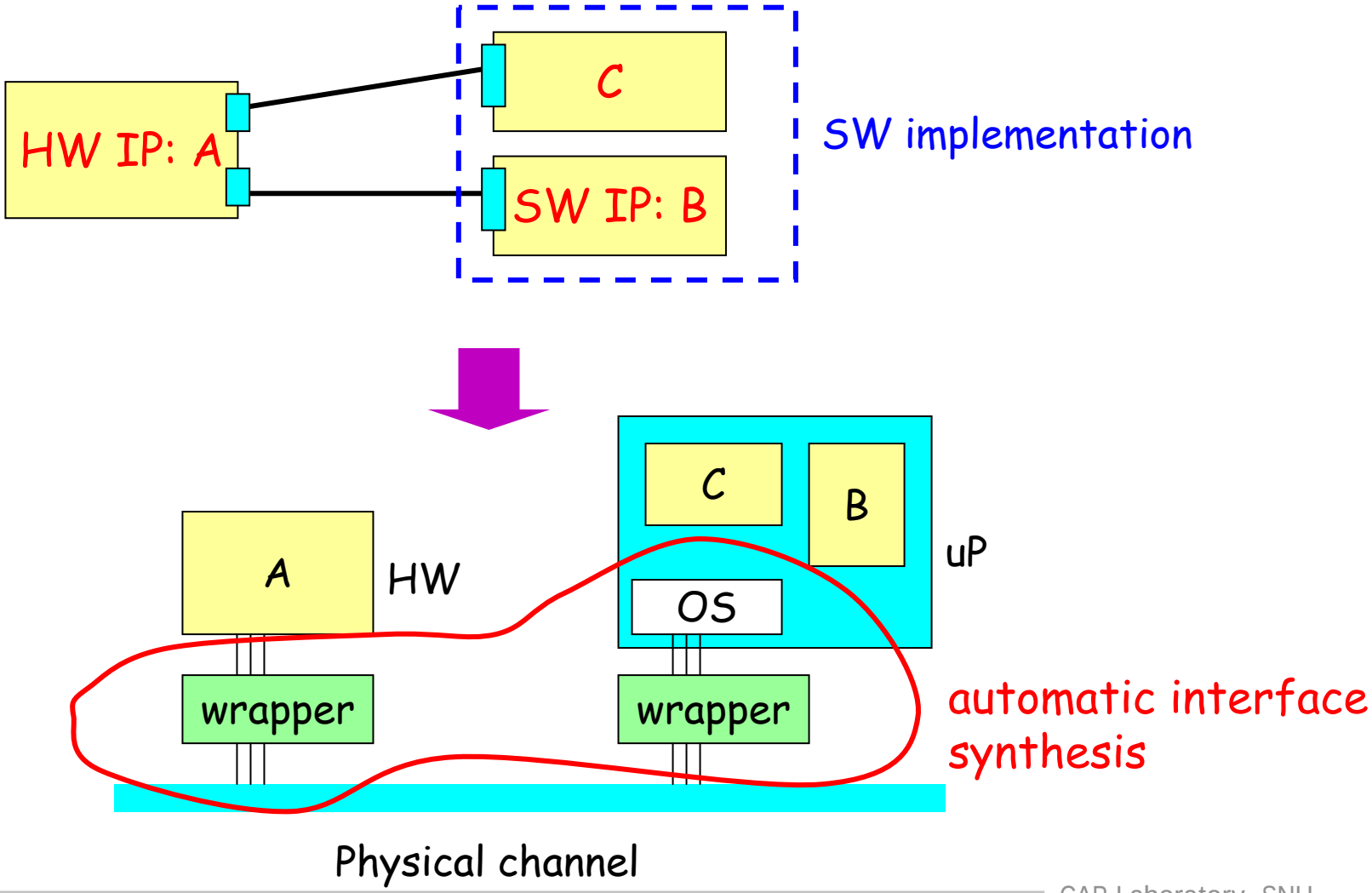
- **Cosimulation performance**
  - (Ex.) How long will it take to cosimulate a system that has 1G ops/sec performance with a simulator whose performance is 100Kcycle/sec?
  - Number of processing components keeps increasing
    - Parallel cosimulation?
- **Timing accuracy**
  - For DSE
  - For HW/SW coverification
- **Mixed-level simulation**
  - RTL and TLM
- **Extensibility & configurability**

(Issue 4) HW/SW/Interface Synthesis

- **Problem statement**
  - For a given architecture, refine the system-level specification to the implementation codes for software and hardware by automatically synthesizing the interface code.

- **Need**
  - Implementation can be “correct by construction” if formal specification is refined.
  - Interface code optimization improves the design productivity significantly

Model to Architecture Implementation



Codesign Researches

- **Early Researches**

Project / Tool	URL	Focus
Akka - Royal Institute of Technology (Prof. Axel Jantsch)	http://www.ele.kth.se/~nalle/codes/	HW/SW codesign toolkit based on C, C++
Chinook - U. Washington (Prof. G. Boriello)	http://www.cs.washington.edu/research/chinook/index.html	Communication synthesis
CodeSign - ETH Swiss (Prof. J. Teich)	http://www.tik.ee.ethz.ch/~codesign/	Formal specification and simulation
COOL (COdesign TOol) - Univ. of Dortmund, (Prof. P. Marwedel)	http://ls12-www.cs.uni-dortmund.de/~niemann/cool/cool.html	HW/SW partitioning
COSYMA - Technical University of Braunschweig (Prof. Rolf Ernst)	http://www.ida.ing.tu-bs.de/projects/cosyma/home.e.shtml	cosynthesis
Polis - U.C.Berkeley (Prof.Sangiovanni-Vincentelli)	www-ad.eecs.berkeley.edu/Respep/Research/hsc/abstract.html	Formal specification and analysis
RASSP (Rapid Prototyping of Application Specific Signal Processors) - Lockheed Martin	http://www.eda.org/rassp/	Design methodology for embedded DSPs

Summary for Early Researches

- **Language based design**
  - (ex) COSYMA: use an extension of C for initial specification
- **Single Processor + HW**
- **Partitioning algorithms with unrealistic assumptions**
- **RASSP Project**

"RASSP met many of these goals through a combination of advanced design methodology emphasizing virtual prototyping, concurrent engineering, and design re-use; modular, scalable signal processor architectures; and a comprehensive supporting base of electronic design infrastructure, including automation tools, hardware and software libraries, enterprise integration capabilities, and standards."

Codesign Researches: on-going

Project	URL	Focus
ForSyDe - Royal Institute of Technology (Prof. Axel Jantsch)	http://www.ele.kth.se/ForSyDe/	Formal specification and System synthesis
Moses (Modeling, simulation, and evaluation of systems) - ETH Swiss (Prof. L. Thiele)	http://www.tik.ee.ethz.ch/~moses/	Object-oriented Petri-net modeling formalism
PeaCE -Seoul National Univ. (Prof. S. Ha)	http://peace.snu.ac.kr/research/peace	Codesign environment
Ptolemy II - U.C.Berkeley (E.A.Lee)	http://ptolemy.eecs.berkeley.edu	Formal modeling and embedded SW synthesis
SPI (System Property Interval) - Technical Univ. of Braunschweig (Prof. Rolf Ernst)	http://www.ida.ing.tu-bs.de/research/projects/spi/home.e.shtml	Global system analysis of constraints and properties
SynDEx (Synchronized Distributed Executives) -INRIA (Dr. Y. Sorel)	http://www-rocq.inria.fr/syndex/	SW synthesis for real-time systems based on AAA methodology
SLS group - TIMA (Dr. A.A.Jerraya)	http://tima.imag.fr/SLS	Diverse issues of system level design of MPSoC

Current Research Trend

- **Model-based design**
  - UML-based approach
  - Others: Simulink-based approach
- **Multi-Processor System on Chip (MPSoC)**
  - Fast co-simulation (distributed simulation)
  - NoC (Network on Chip)
  - Mapping
- **Embedded Software Development**
  - MPSoC Project (HOPES)
  - HW/SW Interface synthesis: TIMA SLS group
- **Fault-resilient design (NT-related)**

Contents

- Embedded System Design
- Design Methodology: Old and New
- Design Reuse: Platform Based Design (PBD)
- HW/SW Codesign
- Formal Model and CAD tools
- Other Design Issues – not to be covered in this course

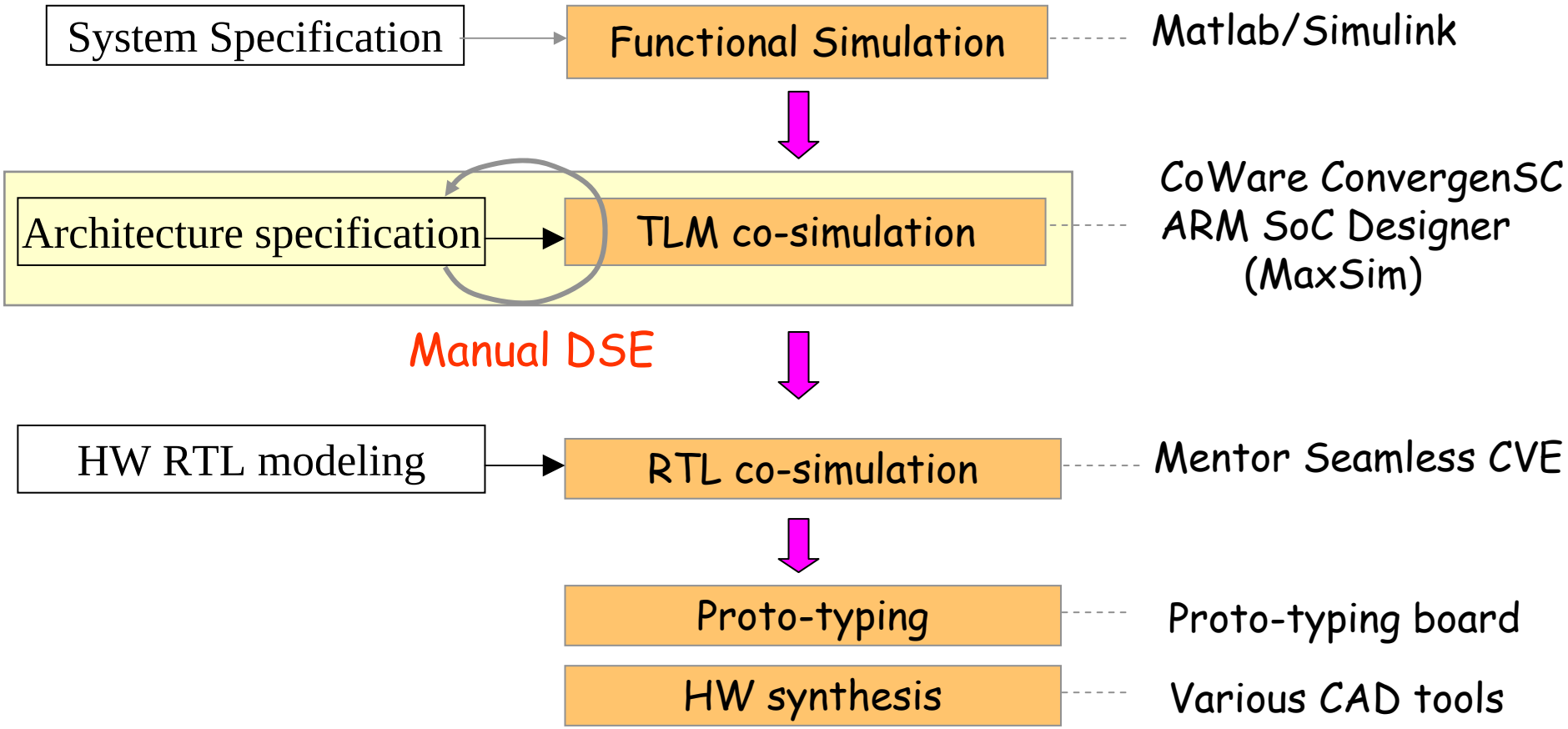
HW/SW Codesign Tools

Project / Tool	URL	Focus
CCSS - Synopsys Inc.	http://www.synopsys.com/products/cocentric_studio/cocentric_studio.html	TLM cosimulation
ConvergenSC - CoWare Inc.	http://www.coware.com	TLM cosimulation Interface synthesis
GEDAE - Gedae Inc.	http://www.gedae.com	Embedded SW synthesis for hetero. multiprocessor system
SoC Designer (MaxSim) -ARM	http://www.arm.com/products/DevTools/SoCDesigner.html	TLM cosimulation
ObjectGeode - Telelogic	http://www.telelogic.com/products/additional/objectgeode/index.cfm	Toolset for distributed real-time systems (UML, SDL, and MSC)
Seamless CVE -Mentor Graphics	http://www.ida.ing.tu-bs.de/research/projects/spi/home.e.shtml	RTL cosimulation
SIMULINK - The Mathworks	http://www.mathworks.com/products/simulink	System simulation, SW synthesis
SPW - CoWare	http://www.coware.com	DSP algorithm simulation
STATEMATE - Telelogic(i-Logix)	http://www.ilogix.com/statemate/statemate.cfm	System specification and rapid prototyping of complex embedded systems

HW/SW Codesign Tools

- **Focus on Simulation**
  - RTL simulation: Mentor Graphics (Seamless CVE)
  - TLM simulation: CoWare (ConvergenSC), ARM (MaxSim)
- **C-based design is becoming more popular**
  - Some companies sell C-to-HW tool
    - Mentor Graphics (Catapult C), Forte (Cynthesizer), Synfora (PICO Express), Y Explorations (eXCite), Celoxica
- **Difficulties**
  - Tools are hard to use
  - Manual partitioning and design space exploration

Current HW/SW Codesign Practice



Design Validation

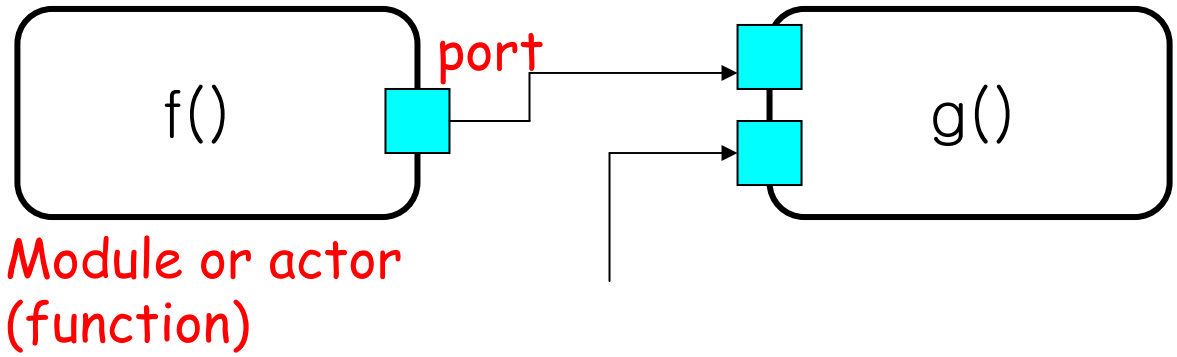
- **By construction**
  - property is inherent.
- **By verification**
  - property is provable.
- **By simulation**
  - check behavior for all inputs.
- **By intuition**
  - property is true. I just know it is.
- **By assertion**
  - property is true. Believe it.

Need of formal specification

It is generally better to be higher in this list

Formal Specification

- **Precise and unambiguous semantics**
  - Functional specification: f (input, output, state)
  - Well defined function composition
  - Properties and Constraints



Merits of Formal Models

- **Easy validation: "*correct by construction*"**
- **Design time reduction**
  - Early discovery of ambiguities, omissions, and contradictions
  - Component reuse
- **Collaboration**
  - Concurrent and interactive design activities
- **Design optimization**
  - Design space exploration
- **Maintenance**
  - Documentation
  - Hierarchical and structural decomposition

Formal Design Frameworks

- **DSP/multimedia applications: Dataflow model**
  - SPW: Alta group of Cadence
  - COSSAP: Synopsis
  - Grape-II (or Virtuoso Synchro) from Intelligent Systems, Belgium
- **Control applications: FSM model**
  - STATEMATE: I-Logix
  - POLIS: CFSM
- **Process Model**
  - COSY: Kahn process network (KPN)
- **Heterogeneous**
  - Ptolemy
  - PeaCE

codesign environment

Weakness of Formal Models

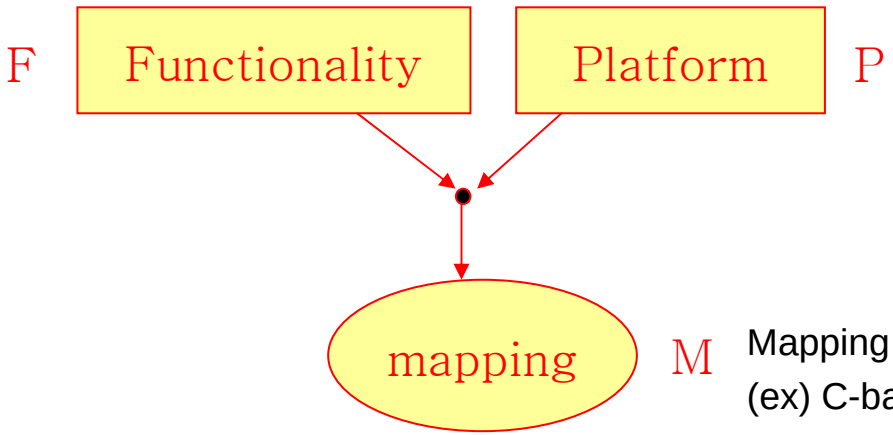
- **New specification**
  - Learning curve
  - Will be overcome if benefit is large
- **Expression capability**
  - No limitation on expressiveness?
  - Usually confined to a class of applications
- **Inefficient refinement**
  - How close to the manually optimized design

- **Current Situation**
  - Specific to a class of applications
  - Mainly for (functional) simulation

Platform-Based Taxonomy

- **Reference**
  - D. Densmore, A. Sangiovanni-Vincentelli, and R. Passerone, “A Platform-Based Taxonomy for ESL Design,” IEEE Design&Test, (23)5, pp. 359-374, September, 2006.
- **Platform-based design classification framework**

Functional simulation
Application development
(ex) Matlab, LabView, Mathematica, Verdi Ptolemy II, ForSyDe



Library of elements
Method to assess the elements
(ex) Sonics Studio, Spirit, Nepsys, Nucleus

Mapping and/or synthesis
(ex) C-based design tools (Catapult, Cynthesizer, etc)
Code generation tool (TargetLink, Real-time Workshop)

Tools in metabin FP

- **Languages that can express both functionality and architecture**
  - SystemC
  - UML
  - SystemVerilog
- **Industrial tools**
  - The Mathworks 사의 Simulink, Stateflow

Tools in metabin FM

- **Tools that provide the following capabilities for a fixed platform architecture**
  - Functional description
  - Analysis capability
  - Mapping and synthesis capability
- **Examples**
  - Telelogic: Rhapsody and Statemate - UML based
  - Esterel Technologies: SCADE, Esterel studio

Tools in metabin PM

- **Tools that combine architectural service and mapping**
  - ARM: RealView, SoC designer
  - CoWare: ConvergenSC, LisaTek
  - Tensilica: xTensa, XPRES
  - Cadence: Incisive: integrated platform for verification
  - Mentor: Platform Express
  - Summit: System Architect, Visual Elite: analysis of MPSoC
  - Arteris: Danube, NoCexplorer : synthesis of No

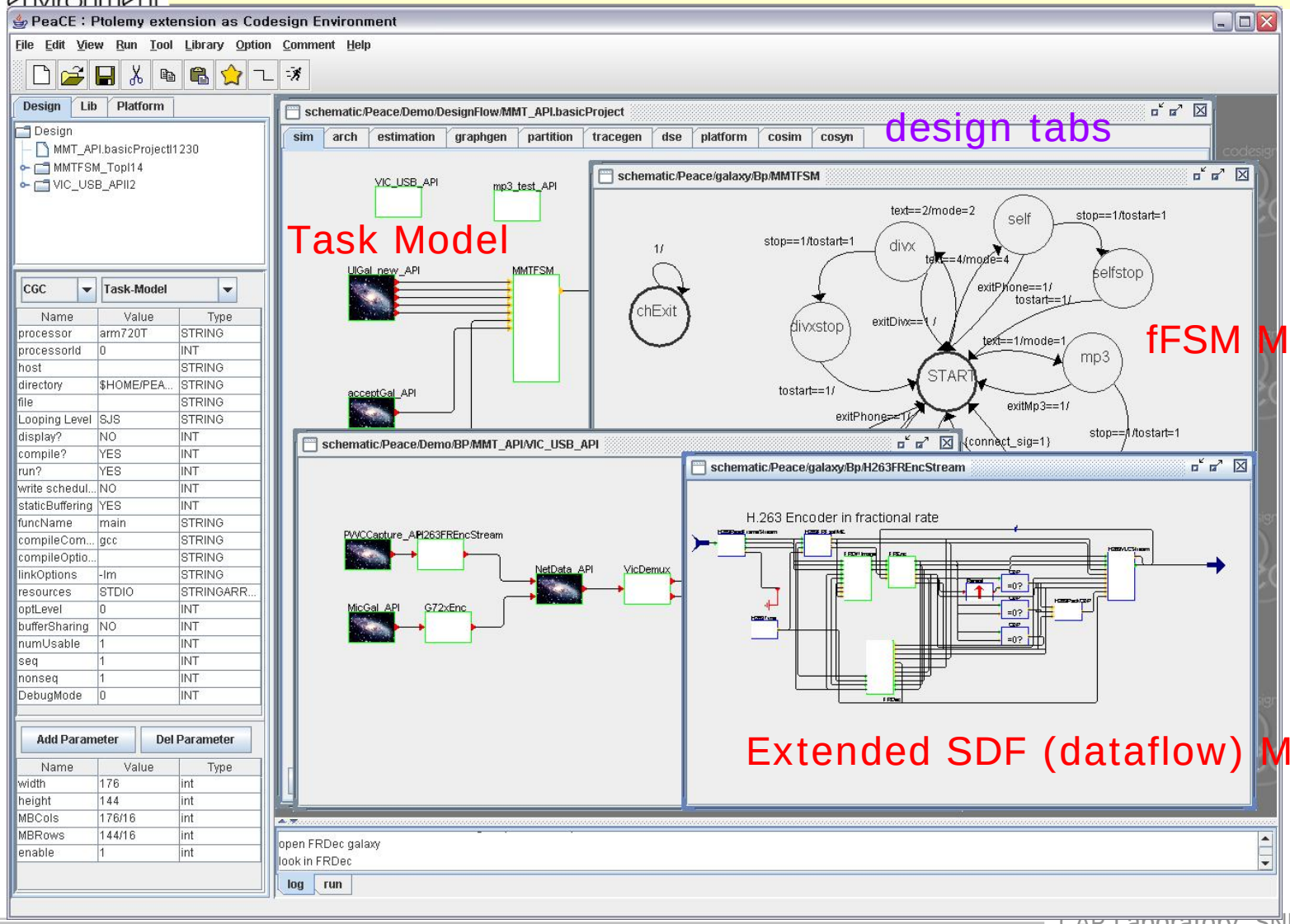
Tools for metabin FPM

- **Frameworks that support the system-level design flow**
  - Synopsys: System Studio: SystemC based platform
  - CoFluent Design: CoFluent Studio: SystemC based DSE tool
  - U.C.Berkeley Metropolis project: meta model based framework
  - U.C.Berkeley Mescal project: Extended Ptolemy II
  - Delft Univ. of Technology: Artemis, Compaan and Laura, Sesame, Spade tool sets: Kahn process network (KPN)-based tool sets
  - SNU PeaCE project

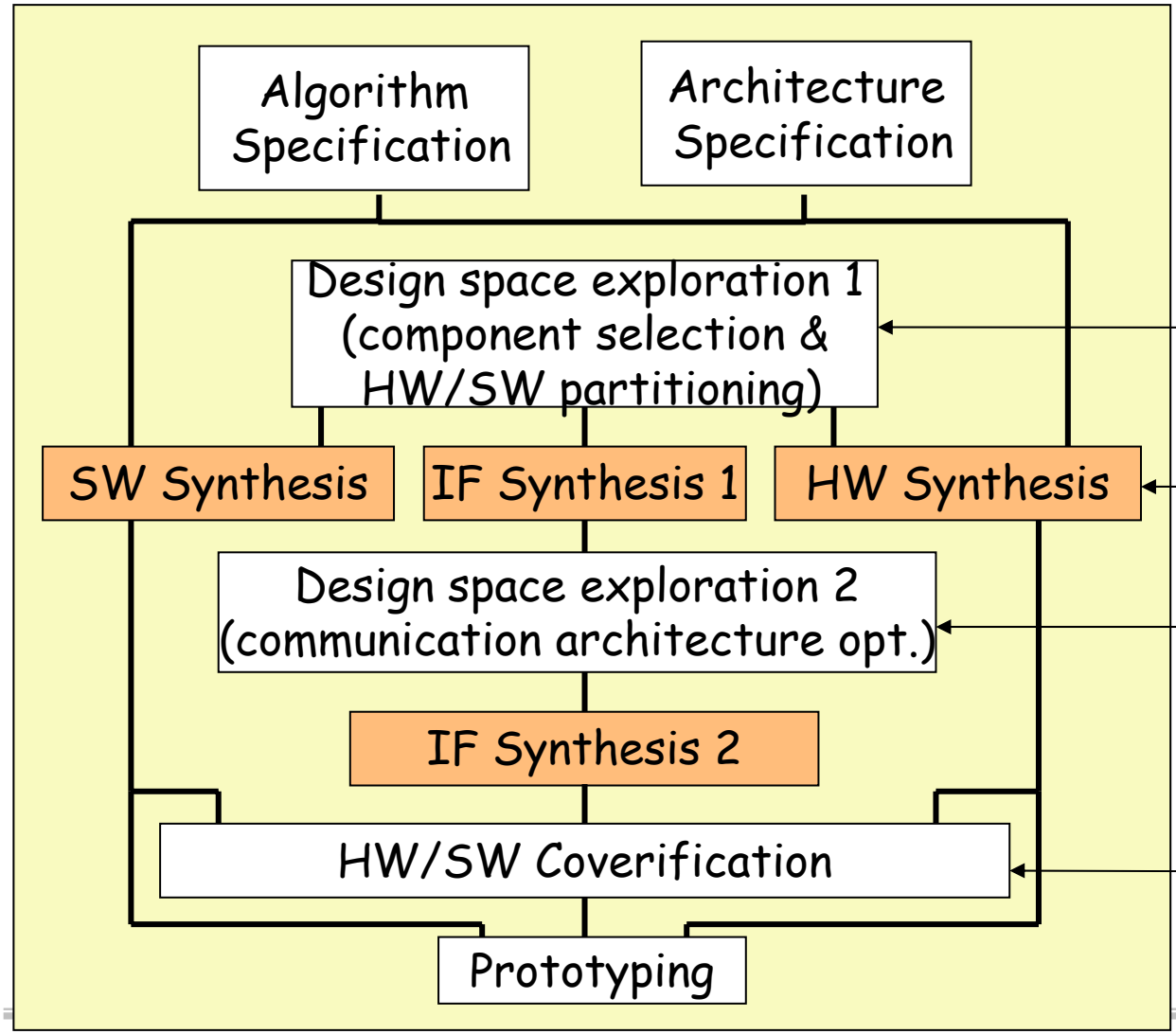
PeaCE

- **PeaCE home page**
  - http://peace.snu.ac.kr/research/peace
  - papers, manual, etc
- **Open source program**
  - Released on Nov. 2004.
  - Demonstrated at the Univ. Booth in DAC 2003-2006
- **Provide all design steps from system-level specification to synthesis**

PeaCE Design Environment



Reconfigurable Codesign Environment



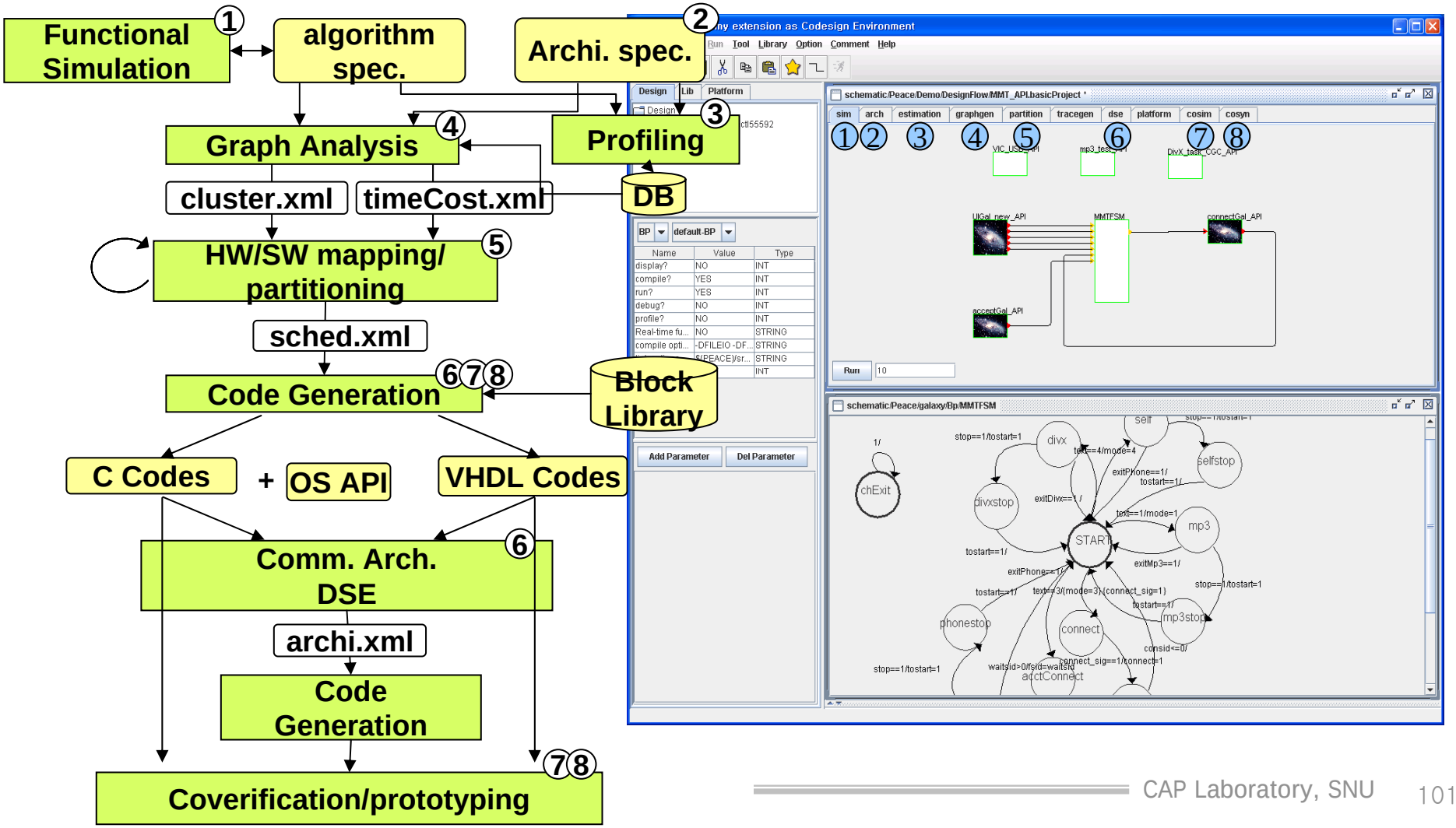
Manual partitioning
Automatic partitioning

Manual synthesis
Automatic generation

TLM simulation
(commercial tools)
Trace-driven simulation

Seamless CVE
Dynalith Environment
Virtual Synchronization

Seamless HW/SW Codesign Flow



HOPES Project

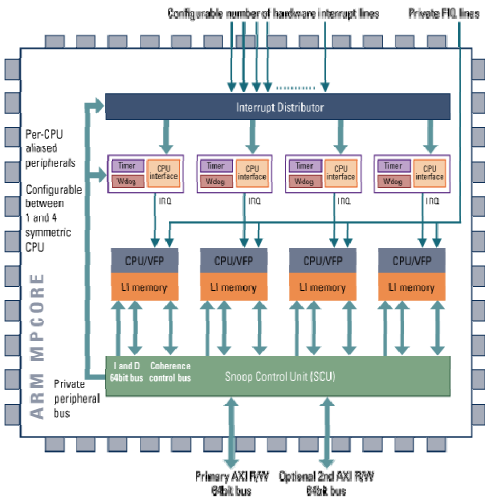
- **Target: MPSoC with high degree of parallelism**
  - Scalability
  - Heterogeneous processors with diverse communication architecture
  - Power-constrained system

- **Problem: parallel programming for MPSoC**
  - Parallelism extraction (multiple use case, multi-tasking apps.)
    - Functional parallelism, data-parallelism, temporal-parallelism
  - Partitioning and mapping
  - Parallel code generation: parallel programming is not easy
  - Performance estimation and verification
  - Design space exploration

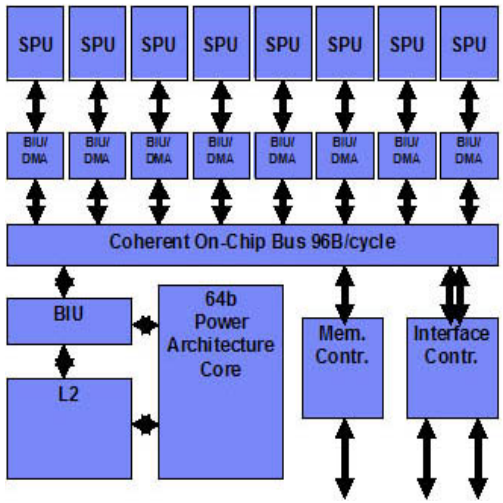
Need of sound (scalable and robust) methodology

Technology Trend

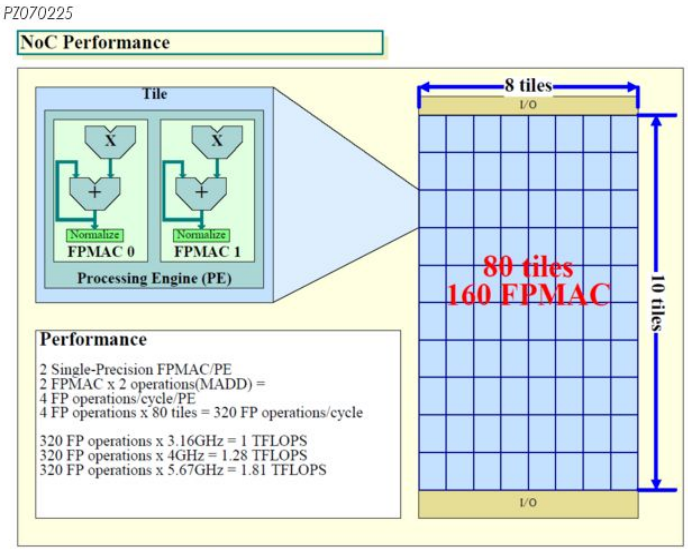
- **More processor cores in a single chip**
  - MPSoC (Multiprocessor System on Chip)
  - from multicore to manycore
    - UCB prediction: “The Landscape of Parallel Computing Research: A view from Berkeley,” Technical report, Dec. 18, 2006: 1000 processor cores with 30nm process
  - Heterogeneous architectures



ARM MPCore



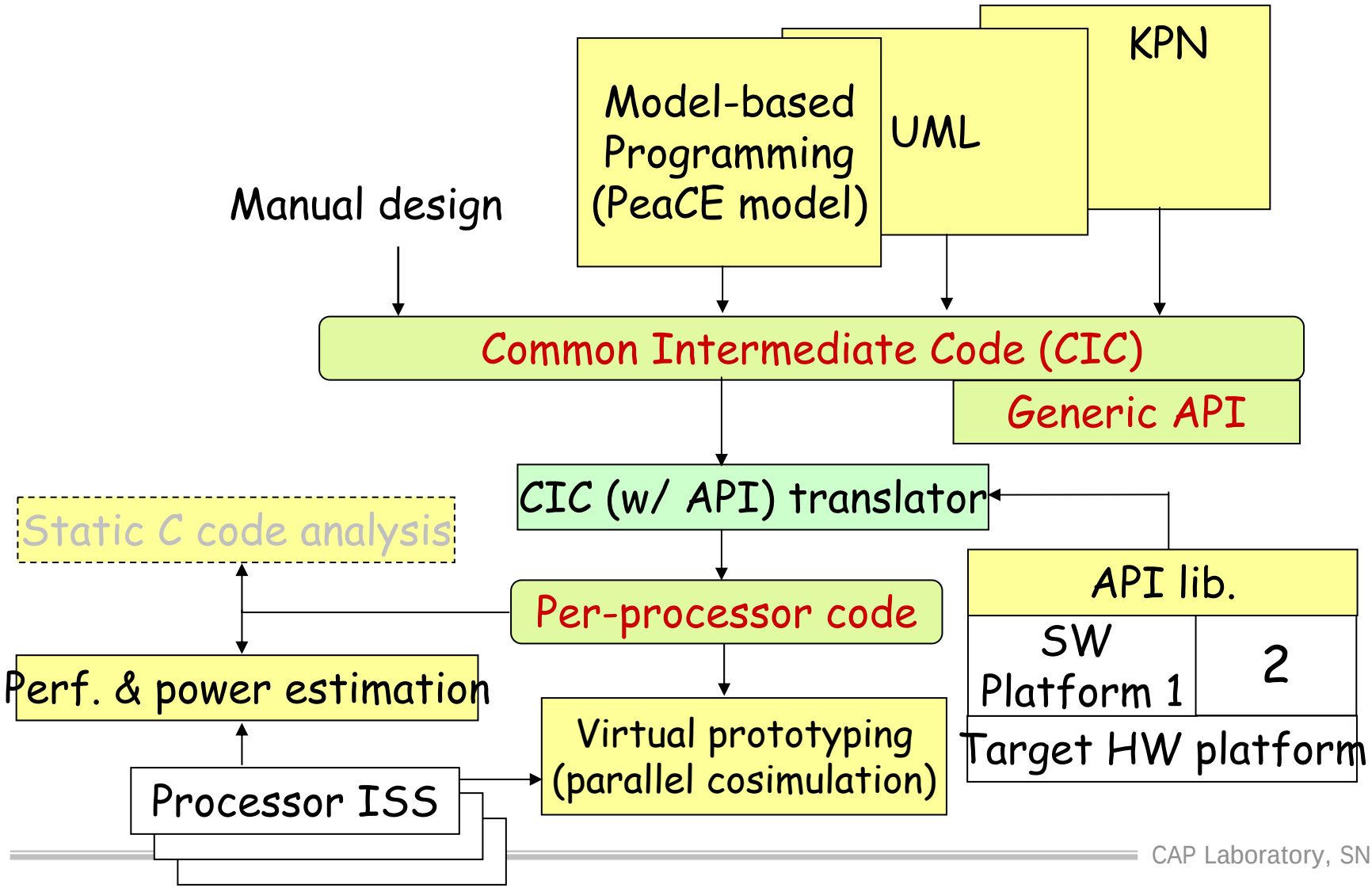
IBM Cell



We Need More than PeaCE

- **Parallelism**
  - PeaCE focuses on function parallelism: task model, dataflow model
  - How to extract data parallelism and temporal parallelism?
- **Decoupled specification and implementation**
  - PeaCE: tight coupling of model and implementation
  - Allow diverse pairs of {model, implementation}
- **Diverse communication architecture**
  - PeaCE: consider shared-bus architecture only
  - Design space exploration covering bus-matrix and NoC
- **Virtual prototyping**
  - PeaCE: virtual synchronization achieves good performance
  - How to improve the cosimulation performance further? Parallel cosimulation

HOPES Proposal



codesign environment

Contents

- Embedded System Design
- Design Methodology: Old and New
- Design Reuse: Platform Based Design (PBD)
- HW/SW Codesign
- Formal Model and CAD tools
- Other Design Issues – not to be covered in this course

Validation and Test

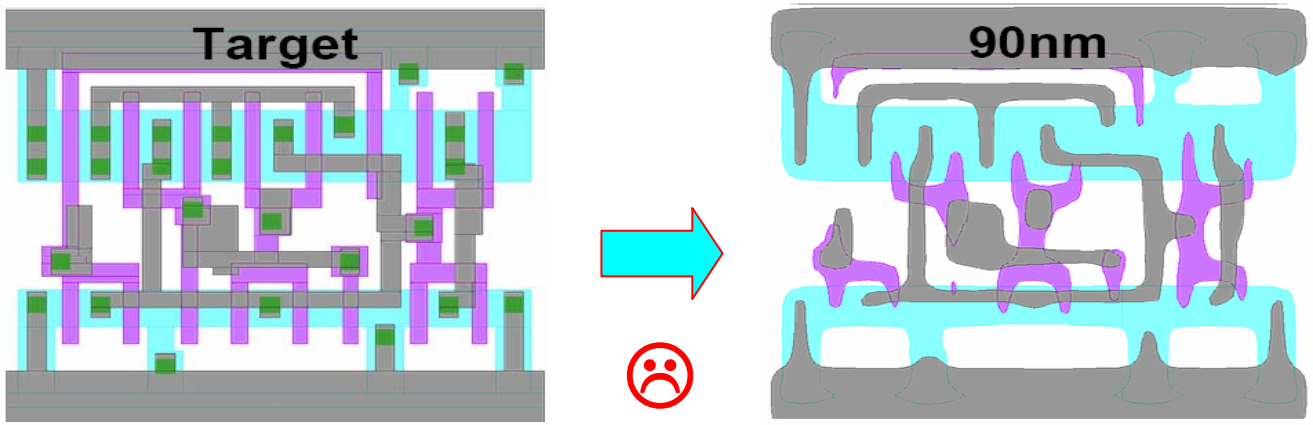
- **Validation Methods**
  - Simulation
  - Hardware Accelerated Simulation
  - Formal Verification
  - HW Emulation
  - Rapid Prototyping
- **Formal verification**
  - Component/IP verification: Combinational logic, Sequential logic
- **Language-based validation**
- **Testing and Testable Design**
  - Fault Modeling
  - Automatic Test Pattern Generation (ATPG)
  - SCAN Testing
  - Built-In Self-Test (BIST)
  - IDDQ Testing

Low Level Design of SoC

- **High-level Synthesis (or Behavioral Synthesis)**
  - From HDL, C
  - Issues: Operation scheduling, module allocation and binding
- **RTL and Logic Synthesis**
  - Timing optimization, retiming, pipelining
  - Gate level optimization
- **Physical Design**
  - Floor-planning, routing
- **Signal Integrity**
  - RC delays, Interconnect Coupling Capacitance, IR drop, etc.
- **DFM: Design for Manufacturability**

Design for Manufacturability

- **Who is responsible for pattern quality.**
  - Design, Photo, Cad, Integration
- **New Paradigm and Process Rule is needed !!**



Summary

- **New Design Methodology**
  - Design reuse: IP-based, Platform-based design
  - HW/SW Codesign
  - Model-based design
  - Design automation tools (CAD tools)

- **HW/SW Codesign Issues**
  - System modeling
  - Design space exploration
  - HW/SW Cosimulation
  - HW/SW/Interface Synthesis

- **HW/SW Codesign environment**
  - PeaCE

Questions

- **1. What is the main difference of SoC design from ASIC design**
- **2. Explain the basic concept of HW/SW codesign methodology. What are the major 4 issues in HW/SW codesign?**
- **3. Explain briefly the following terms**
  - Platform-based design
  - Abstraction
  - Correct-by-construction
  - Virtual prototyping