System Design Paradigms

Alberto Sangiovanni Vincentelli

The Edgar L. and Harold H. Buttner Chair of Electr. Eng. and Comp. Science

University of California at Berkeley

Co-Founder, Chief Technology Advisor and Board Member

Cadence Design System

Founder and Scientific Director

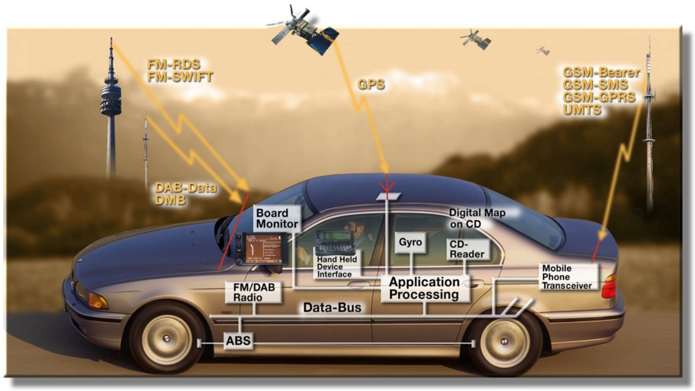
PARADES (Cadence, Magneti-Marelli, ST)

PARADES

DAC System Level Design with Embedded Platforms *Tutorial*

Electronics and the Car

- •More than 30% of the cost of a car is now in Electronics
- •90% of all innovations will be based on electronic systems



Outline

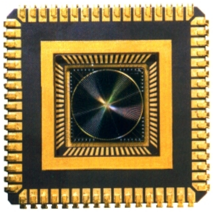
We are on the edge of a revolution in the way electronics systems are designed.

◆Electronic Systems

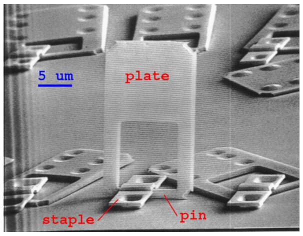
◆Platform-based Design

◆Embedded Software

Chips Everywhere!

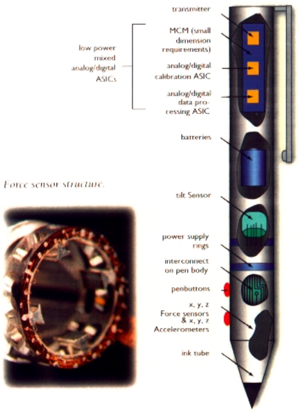


CMOS Camera

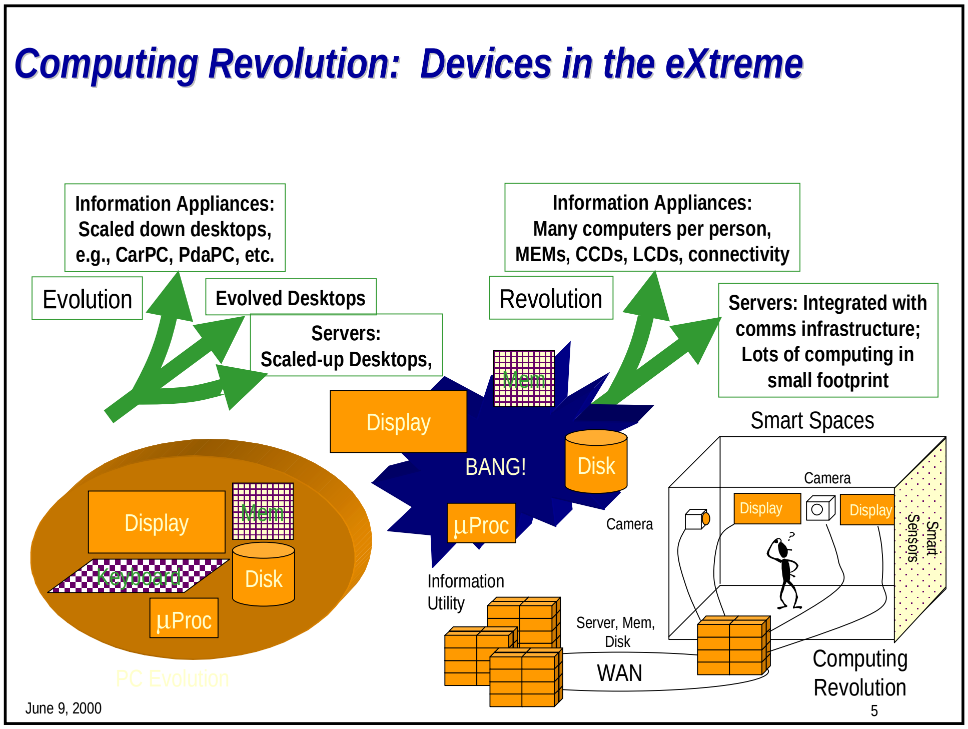


Source: Dr. K. Pister, UC Berkeley

Chips that Fly?


transmitter
MCM (small dimension requirements)
low power mixed analog/digital ASICs
analog/digital calibration ASIC
analog/digital data processing ASIC
batteries
Force sensor structure.
tilt Sensor
power supply rings
interconnect on pen body
penbuttons
x, y, z Force sensors & x, y, z Accelerometers
ink tube

SmartPen



Computing Revolution: Devices in the eXtreme

Information Appliances: Scaled down desktops, e.g., CarPC, PdaPC, etc.

Evolution

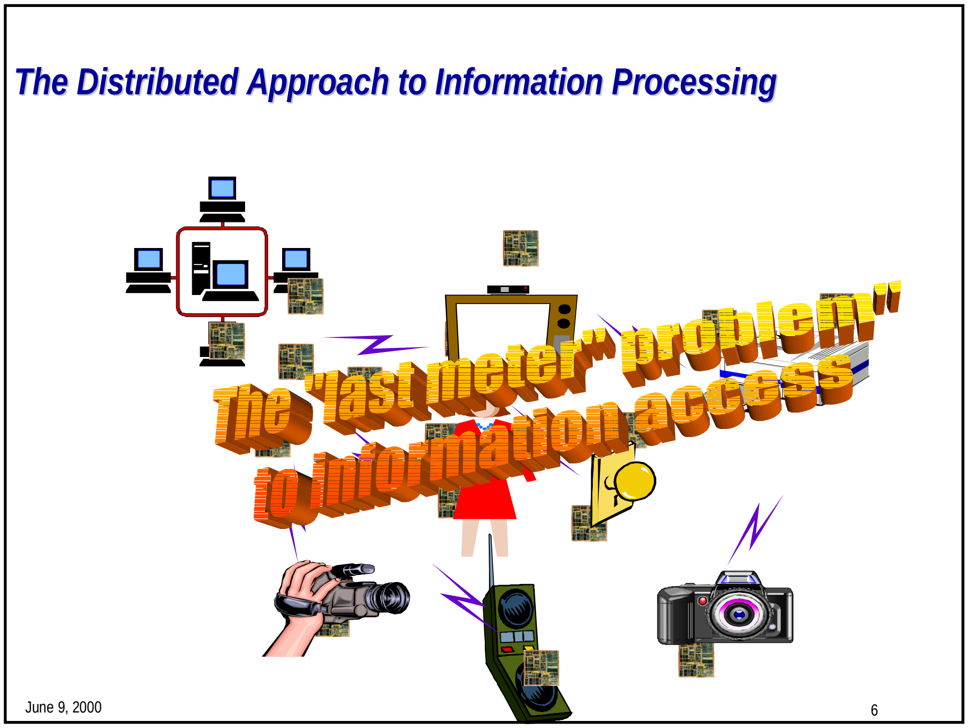
Evolved Desktops

Servers: Scaled-up Desktops,

Information Appliances: Many computers per person, MEMs, CCDs, LCDs, connectivity

Revolution

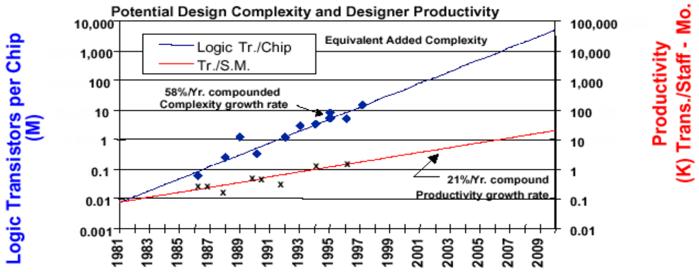
Servers: Integrated with comms infrastructure; Lots of computing in small footprint



The Distributed Approach to Information Processing

The "last meter" problem to information access

Productivity Gap

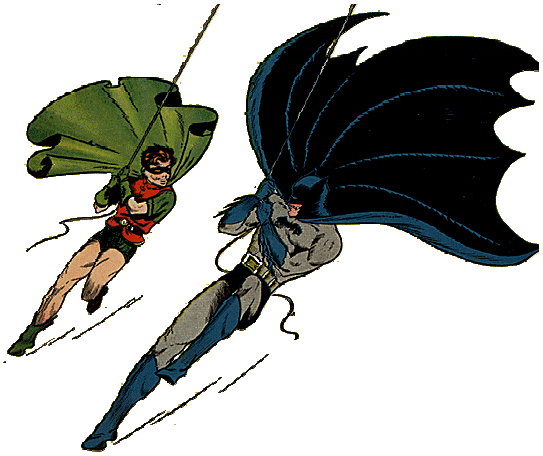


Year	Technology	Chip Complexity	Frequency	3 Yr. Design Staff	Staff Cost*
1997	250 nm	13 M Tr.	400	210	90 M
1998	250 nm	20 M Tr.	500	270	120 M
1999	180 nm	32 M Tr.	600	360	160 M
2002	130 nm	130 M Tr.	800	800	360 M

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

How are we going to solve the challenge of Design?

- Design Science
- Collaboration!



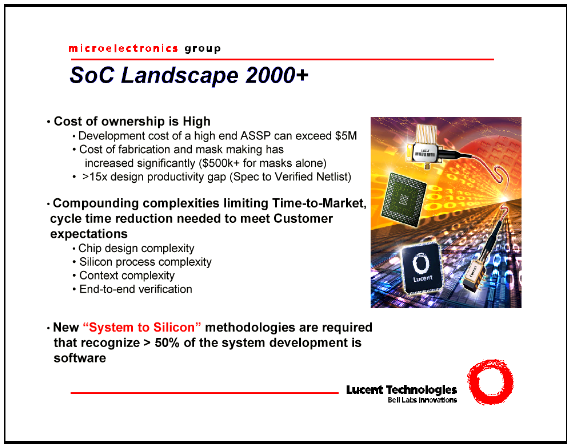
Challenges

Fact

- Total Number of Design Starts will decrease
- Complexity per Design Start is going up

Shift to

- Reuse Strategy
- Higher Level of Abstractions
- Software !!!



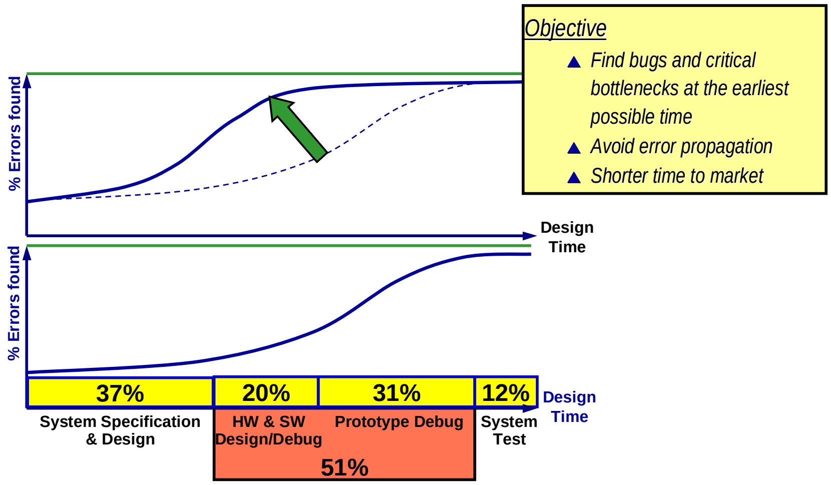
microelectronics group

SoC Landscape 2000+

- Cost of ownership is High
  - Development cost of a high end ASSP can exceed $5M
  - Cost of fabrication and mask making has increased significantly ($500k+ for masks alone)
  - >15x design productivity gap (Spec to Verified Netlist)
- Compounding complexities limiting Time-to-Market, cycle time reduction needed to meet Customer expectations
  - Chip design complexity
  - Silicon process complexity
  - Context complexity
  - End-to-end verification
- New "System to Silicon" methodologies are required that recognize > 50% of the system development is software

Lucent Technologies
Bell Labs Innovations

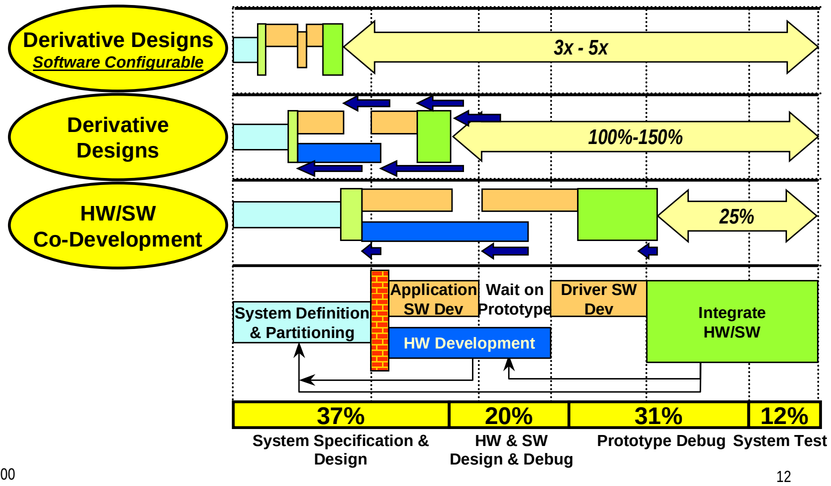
Challenge: Design Flow Predictability



Objective

- *Find bugs and critical bottlenecks at the earliest possible time*
- *Avoid error propagation*
- *Shorter time to market*

Challenge: Productivity



Manufacturing Cost and Design Cost

- Manufacturing costs skyrocketing
  - Mask set cost alone predicted to be $1.5M to $10M
- Design cost increasing exponentially with size of design

10% Decrease in ASIC starts for 1999 w.r.t. 1998.

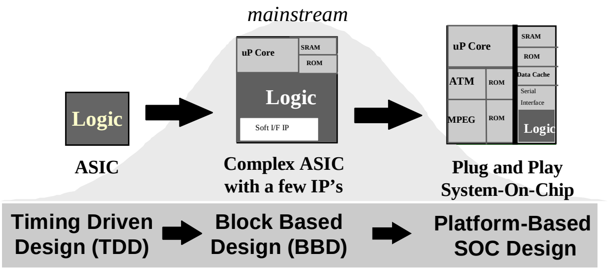
Must re-consider how design is carried out: re-use is main concern at all levels:

Platform-based Design

(H. Chang et al., A. Ferrari and ASV, K. McMillan and ASV)

Integration Platforms…

… the next step in the Evolution of Design Reuse



- In TDD, Reuse in ASIC design is of *Cell-level Libraries*
- In BBD, Reuse in hierarchical design is of *major IP Blocks* (e.g., digital blocks built out of standard cells)
- In SOC, Reuse is of *Collections of IP blocks* organized into HW-SW architectures: also known as *Integration Platforms*

Platform-based Design

- Build upon tools, methods and abstractions

  "We rest on the shoulders of the past. We are midgets on the shoulders of giants", Francis Bacon, Novum Organum

Platform-based Design

- Abstractions are layers upon layers

Photo By: Mike Buchheit

June 9, 2000

Platform-based Design

- The mapping between layers are the pillars of the platform

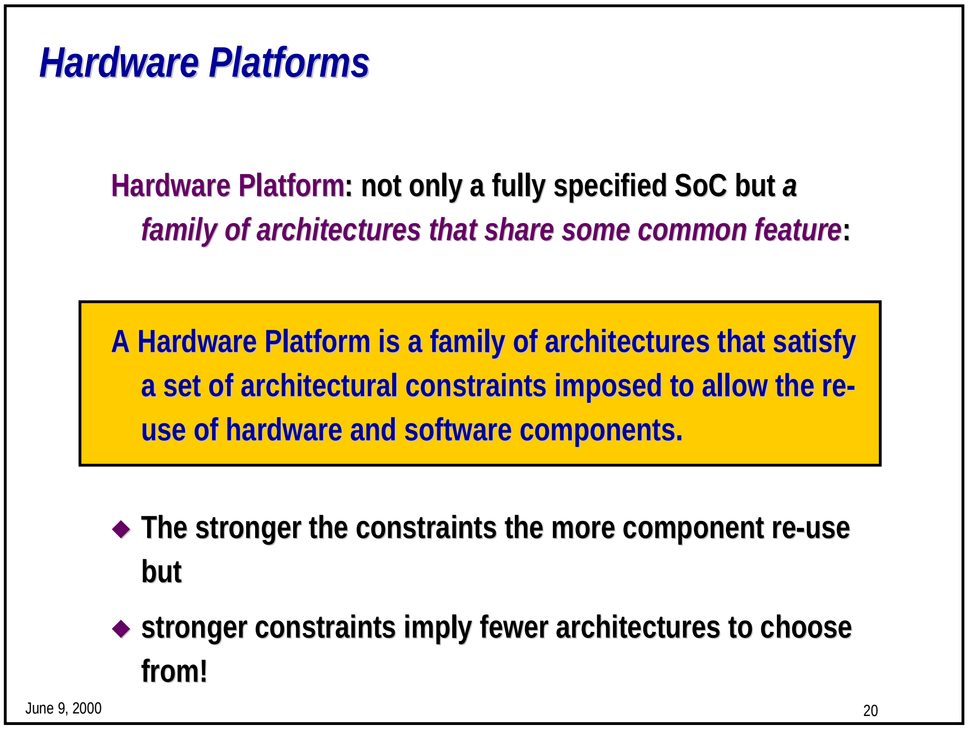
June 9, 2000

Platform-based Design

- The mapping between layers are the pillars of the platform

Platform-based Design

A platform is the combination of abstraction layers and (manual, partially or fully automated) methods for the mapping

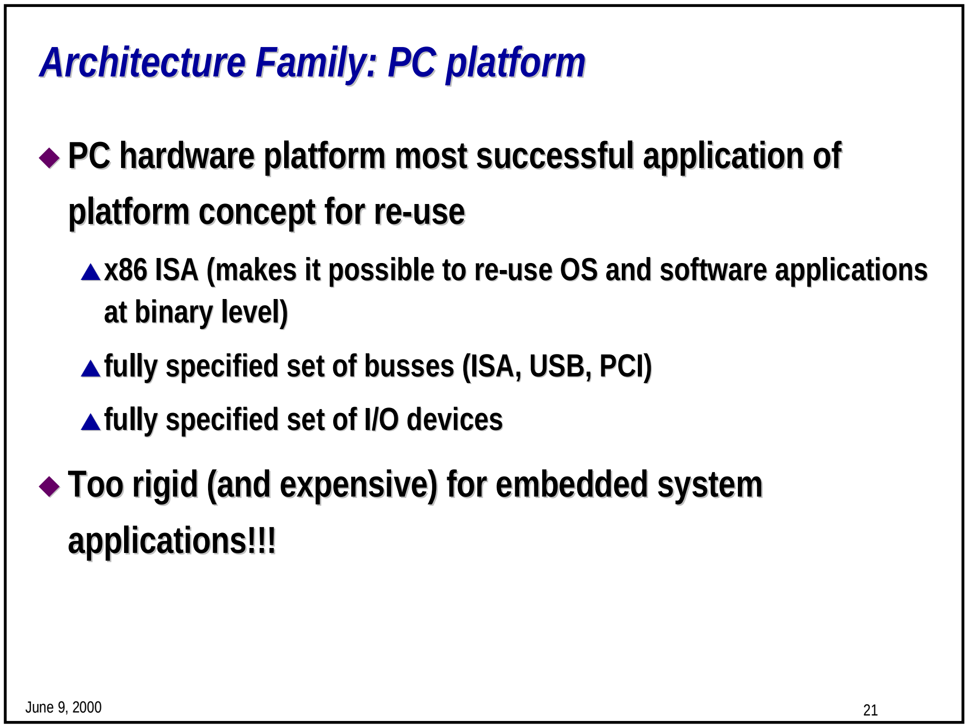


Hardware Platforms

Hardware Platform: not only a fully specified SoC but *a family of architectures that share some common feature*:

A Hardware Platform is a family of architectures that satisfy a set of architectural constraints imposed to allow the re-use of hardware and software components.

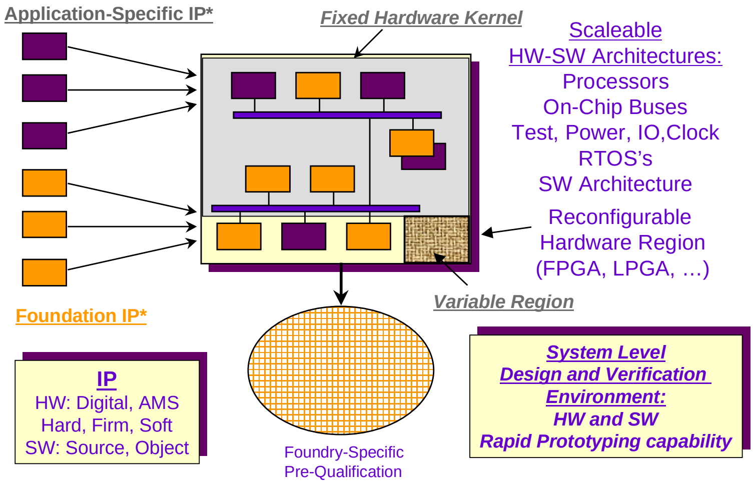
- The stronger the constraints the more component re-use but
- stronger constraints imply fewer architectures to choose from!



Architecture Family: PC platform

- PC hardware platform most successful application of platform concept for re-use
  - x86 ISA (makes it possible to re-use OS and software applications at binary level)
  - fully specified set of busses (ISA, USB, PCI)
  - fully specified set of I/O devices
- Too rigid (and expensive) for embedded system applications!!!

Application-Specific SOC Integration Platforms



Application-Specific IP*

Fixed Hardware Kernel

Scaleable
HW-SW Architectures:
Processors
On-Chip Buses
Test, Power, IO,Clock
RTOS's
SW Architecture

Reconfigurable
Hardware Region
(FPGA, LPGA, …)

Variable Region

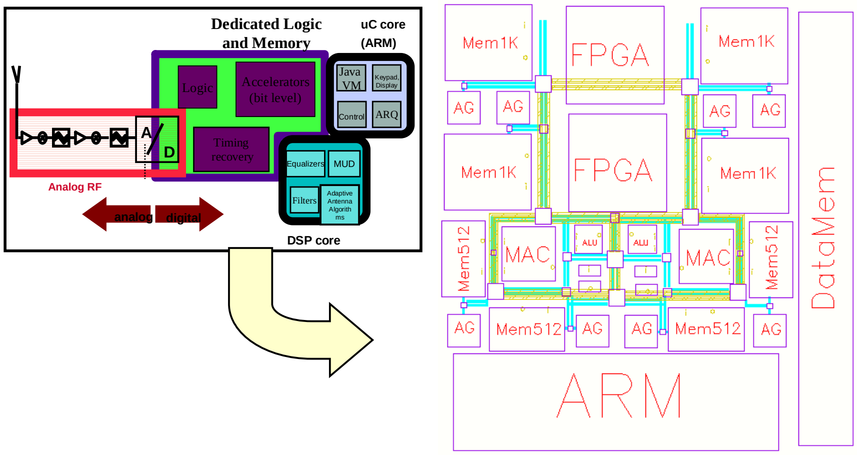
Foundation IP*

IP
HW: Digital, AMS
Hard, Firm, Soft
SW: Source, Object

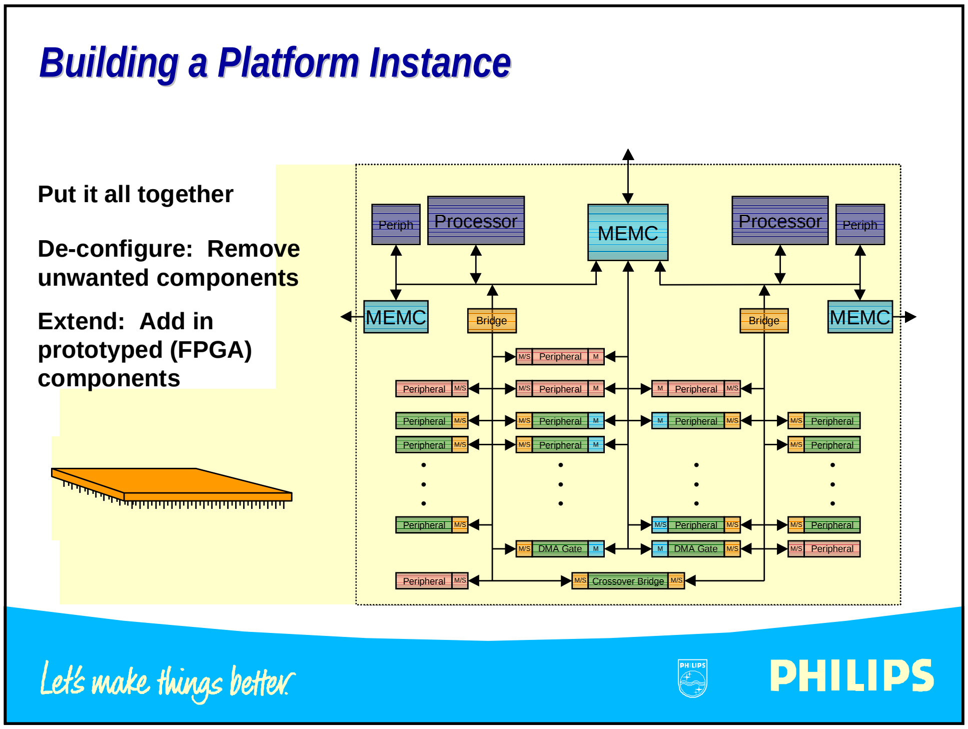
Foundry-Specific
Pre-Qualification

System Level
Design and Verification Environment:
HW and SW
Rapid Prototyping capability

Digital Wireless Platform



Source: Berkeley Wireless Research Center



Building a Platform Instance

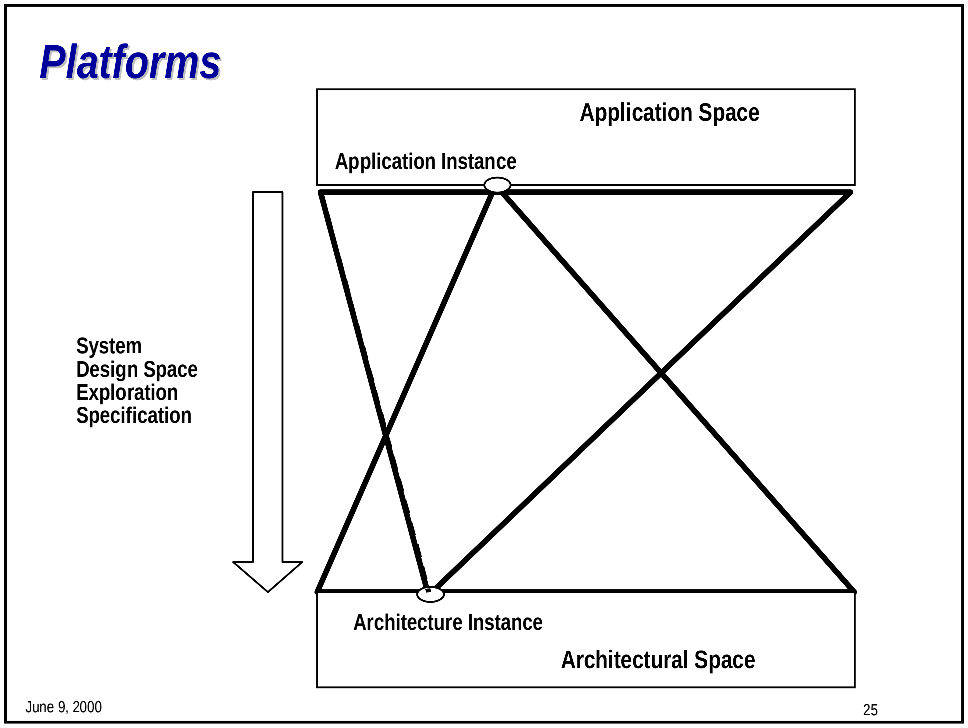
Put it all together

De-configure: Remove unwanted components

Extend: Add in prototyped (FPGA) components

Let's make things better.

PHILIPS



Platforms

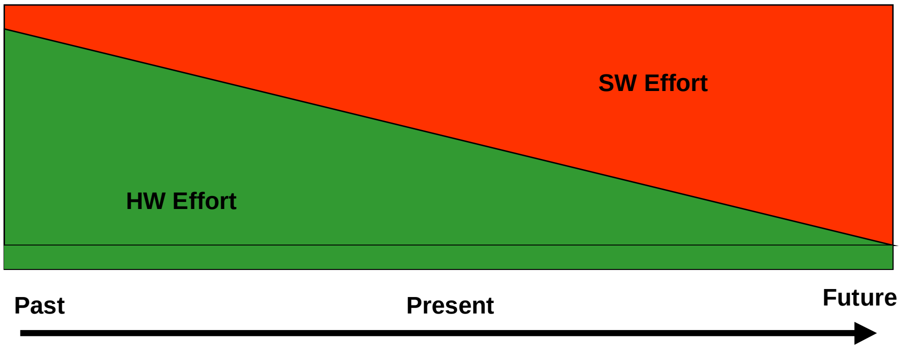
June 9, 2000

Hardware Platforms Not Enough!

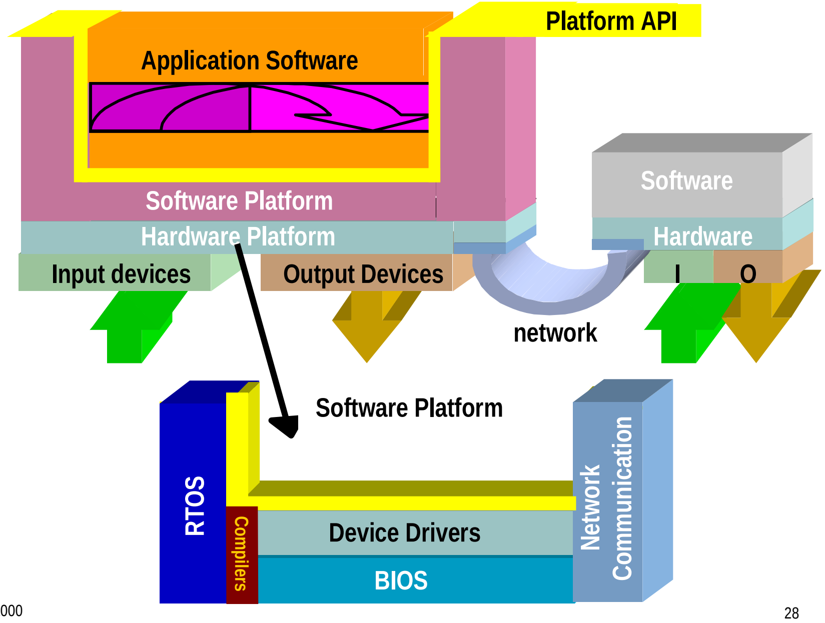
- Hardware platform has to be “extended” upwards to be really effective in time-to-market
- Interface to the application software is API
- Software layer performs abstraction:
  - Programmable cores and memory subsystem with (RT)OS
  - I/O subsystem via Device Drivers
  - Microsoft Windows OS and API is an example!

Why the Software Productivity Concern??

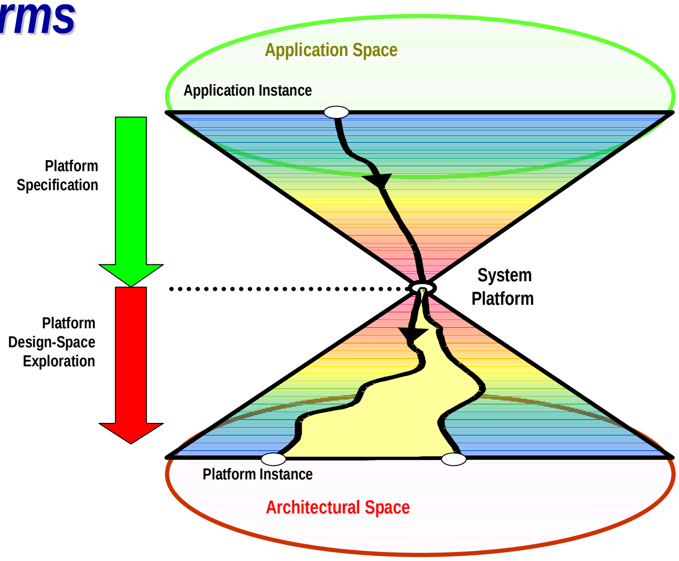
- In the end; if we solve the HW design productivity and fail to address the SW productivity we have accomplished little.



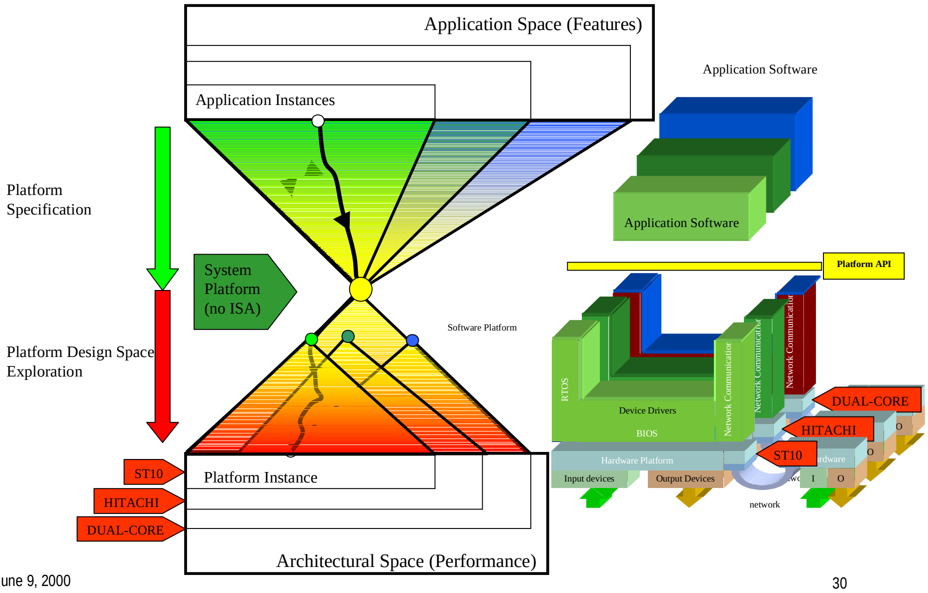
Software Platforms



Platforms



Platform Definition



How is a platform chosen?

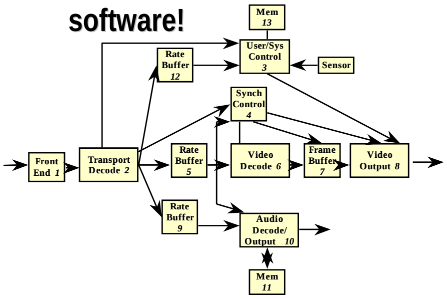
- Needs extensive analysis to optimize among competing criteria
- Performance vs. cost vs. re-use (time-to-market) vs. flexibility
- Millions of parts are needed to be profitable!

System Design: High Leverage Paradigms

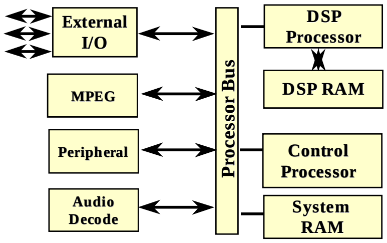
- Orthogonalization of concerns: view designs along axes that can be dealt with independently
  - Timing and functionality
  - Function and Architecture
  - *Computation* and **Communication**

Separate Behavior from Micro-architecture

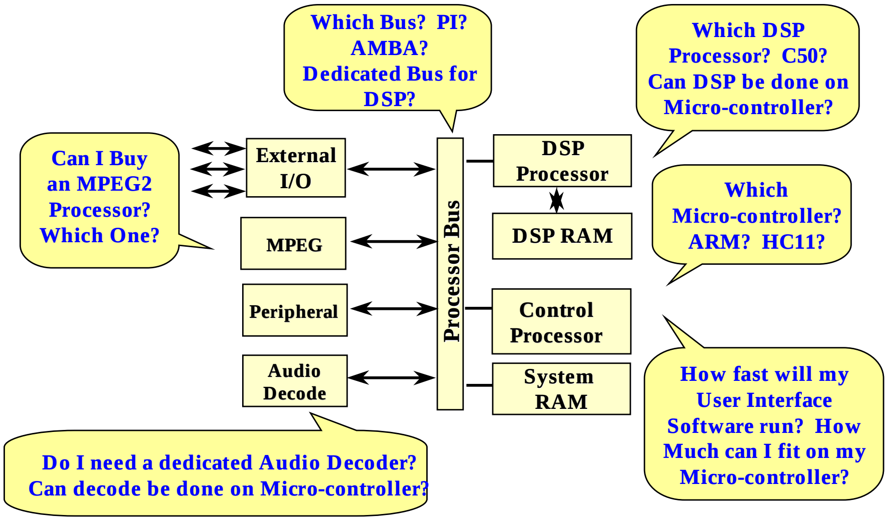
- System Behavior
  - Functional Specification of System.
  - No notion of hardware or software!



- Implementation Architecture
  - Hardware and Software
  - Optimized Computer



IP-Based Design of Implementation



Which Bus? PI? AMBA? Dedicated Bus for DSP?

Which DSP Processor? C50? Can DSP be done on Micro-controller?

Can I Buy an MPEG2 Processor? Which One?

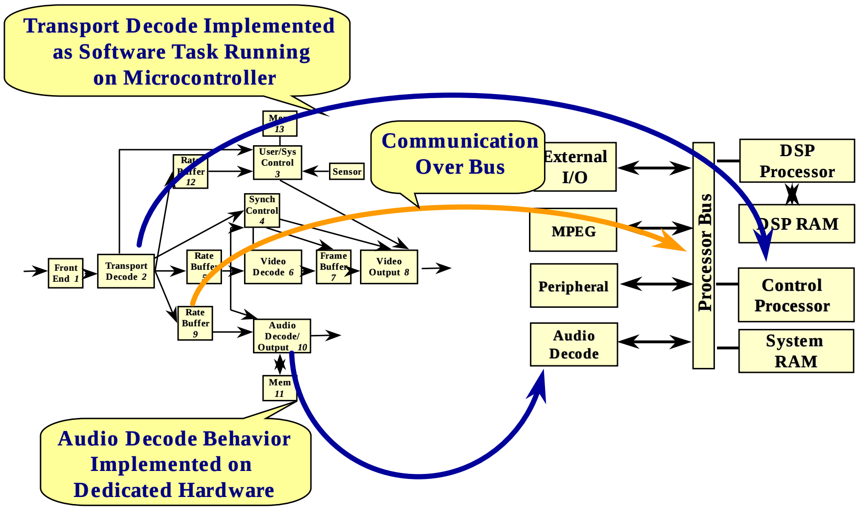
Which Micro-controller? ARM? HC11?

How fast will my User Interface Software run? How Much can I fit on my Micro-controller?

Do I need a dedicated Audio Decoder? Can decode be done on Micro-controller?

June 9, 2000

Map Between Behavior from Architecture



Transport Decode Implemented as Software Task Running on Microcontroller

Communication Over Bus

Audio Decode Behavior Implemented on Dedicated Hardware

June 9, 2000

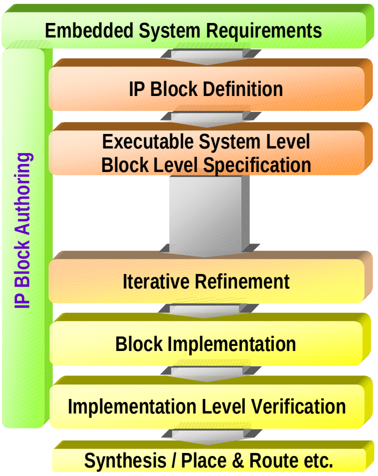
Two Basic Questions ...
Question I - IP Authoring

How to design a system block?

- ▲Starting from the system level
- ▲With a consistent test-bench
- ▲Getting from the abstract, un-timed system model to the clocked HW or SW implementation model

Example

- ◆ Rake Receiver
  - ▲Which are the optimal algorithms?
  - ▲How does it work fixed point?
  - ▲How is it best implemented?
  - ▲Does the implementation work as specified in the system level



Embedded System Requirements

IP Block Definition

Executable System Level Block Level Specification

Iterative Refinement

Block Implementation

Implementation Level Verification

Synthesis / Place & Route etc.

IP Block Authoring

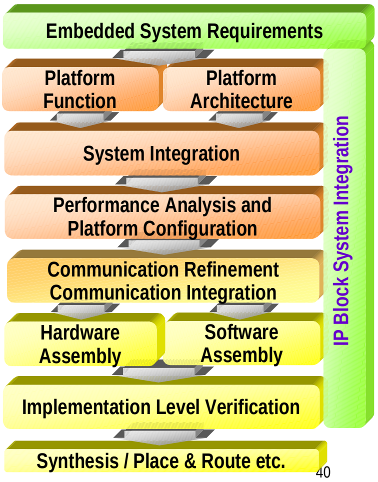
Two Basic Questions ...
Question II – IP Integration

How to integrate system blocks?

- ▲Starting from the system level
- ▲With a consistent test-bench
- ▲Getting from the abstract, un-timed system model to the clocked HW or SW implementation model
- ▲Communication between blocks
- ▲Addressing Platform Based design

Example

- ◆ 3G Cell phone
  - ▲Which are the optimal algorithms?
  - ▲Do they work together functionally?
  - ▲Is the architecture sufficient?
  - ▲Does the implementation integration work?



Embedded System Requirements

Platform Function

Platform Architecture

System Integration

Performance Analysis and Platform Configuration

Communication Refinement Communication Integration

Hardware Assembly

Software Assembly

Implementation Level Verification

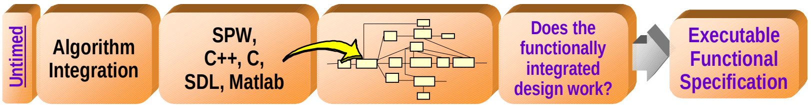
Synthesis / Place & Route etc.

IP Block System Integration

The new approach

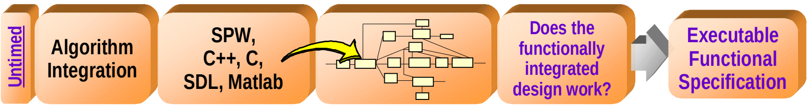
- Not the typical stepwise top-down refinement: we rest on platforms!
- Explicit mapping of applications onto architecture components
- The higher the level of abstraction, the faster is the design time

Functional IP Integration


Untimed
Algorithm Integration
SPW, C++, C, SDL, Matlab
Does the functionally integrated design work?
Executable Functional Specification

- *Question: How does the functional integrated design work?*
- VCC allows
  - ▲ **to** import **functional IP from different sources**
  - ▲ **to** integrate **functional IP from SPW, C++, C, SDL, Matlab etc.**
  - ▲ author **C++, C or FSM based additional IP**
  - ▲ **to** assess **the algorithmic** integration aspects
  - ▲ **to create an** unambiguous functional executable specification

Functional IP Integration



Untimed

Algorithm Integration

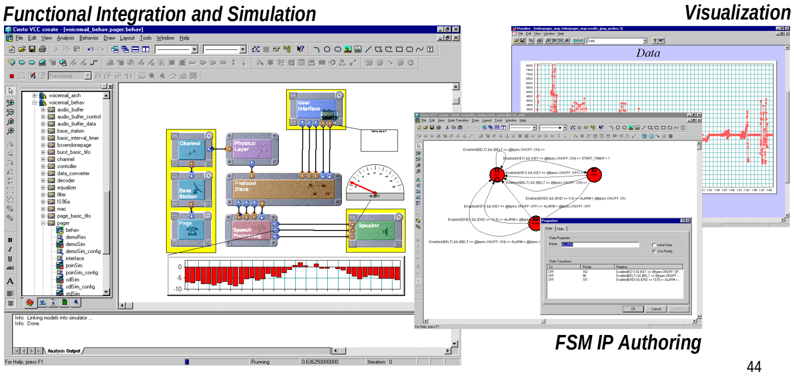
SPW, C++, C, SDL, Matlab

Does the functionally integrated design work?

Executable Functional Specification

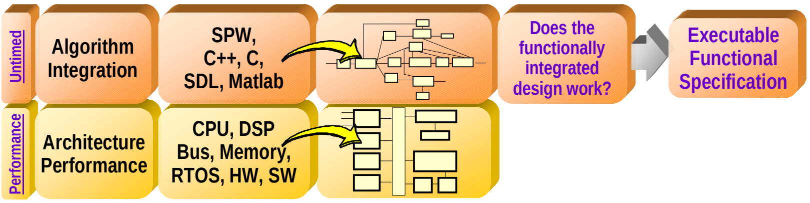
Functional Integration and Simulation

Visualization



FSM IP Authoring

Architectural IP Integration



Untimed

Algorithm Integration

SPW, C++, C, SDL, Matlab

Does the functionally integrated design work?

Executable Functional Specification

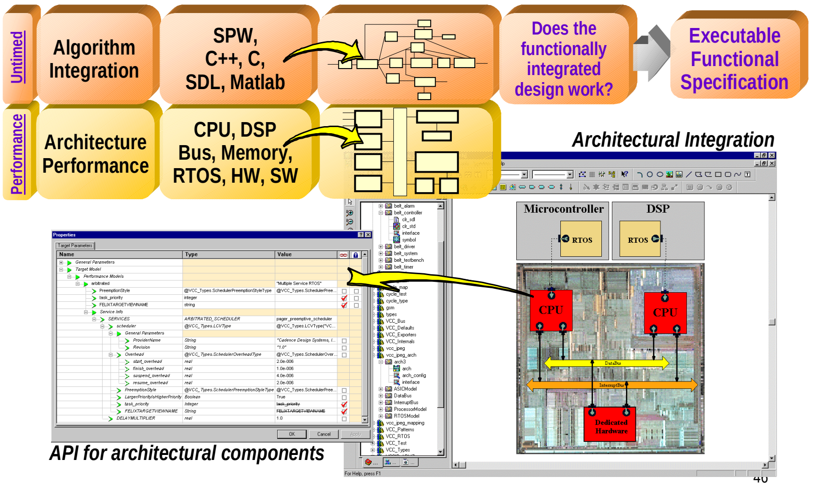
Performance

Architecture Performance

CPU, DSP Bus, Memory, RTOS, HW, SW

- *Question: What does the system architecture look like?*
- **VCC allows**
  - **to** model architectural IP **at the system level**
    - CPU, DSP, RTOS, Bus, Memory and dedicated HW/SW
  - **to** integrate **the** architectural models **defining the platform**
  - **to** present **a** system architecture **to system customers**
  - **to create an** unambiguous architectural specification

Architectural IP Integration


Untimed
Algorithm Integration
SPW, C++, C, SDL, Matlab
Does the functionally integrated design work?
Executable Functional Specification
Performance
Architecture Performance
CPU, DSP Bus, Memory, RTOS, HW, SW
Architectural Integration
Microcontroller
DSP
RTOS
RTOS
CPU
CPU
Dedicated Hardware
API for architectural components

New Research Plan: Logical Infrastructure for System Level Design and Verification

- Models of computation: new theory that will make it possible to link different models of computation
- Platform definition with rigorous formalism about levels of abstraction and mapping
- Three basic components of the framework
  - Proof manager
  - Design agent
  - Verification Agent

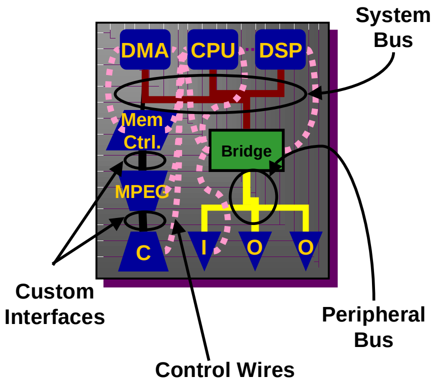
K. McMillan and ASV, R. Passerone, A. Ferrari and ASV

System Design: High Leverage Paradigms

- **Orthogonalization of concerns: view designs along axes that can be dealt with independently**
  - **Timing and functionality**
  - **Function and Architecture**
  - ***Computation* and Communication**

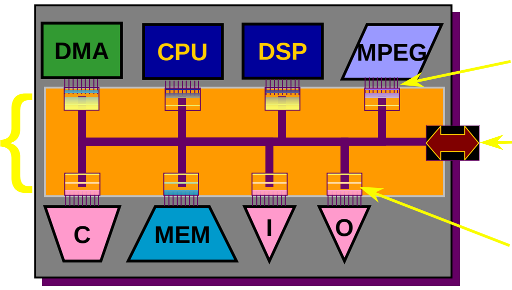
Key Problem: Ad Hoc Integration

Conventional SOC



- **Bus structures inadequate for global SOC quality of service needs**
- **Excessive interdependency between blocks**
- **Incomplete information for front-end modeling**
- **Verification and test unmanageable**

Sonics Integration Architecture (SonicsIA™) Features



SiliconBackplane™ (patented)

Open Core Protocol™ (OCP)

MultiChip Backplane™

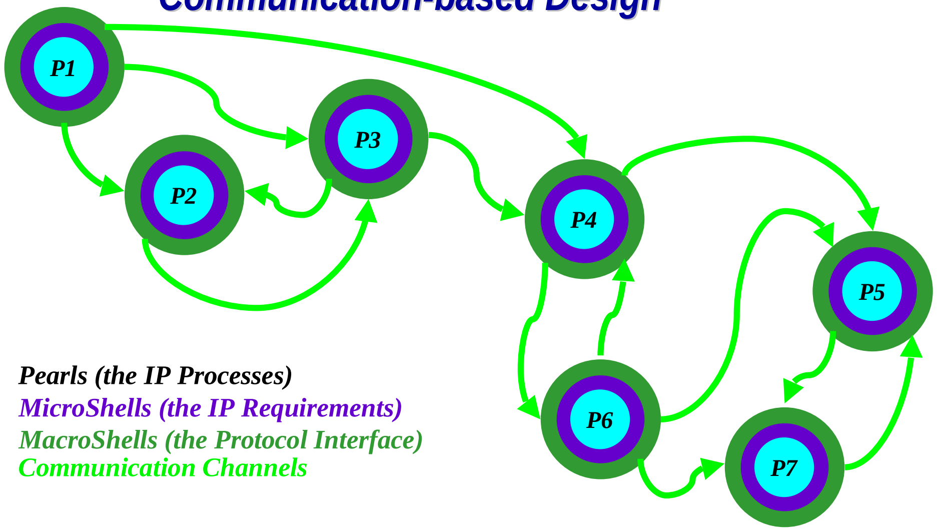
SiliconBackplane Agent™

- Provides critical decoupling (latency, bandwidth, frequency, address map, data width, protocol, control flow, etc.)
- Configures specifically to application
- Highly scalable bandwidth
- Fully observable and controllable

Bottom Line: Component Reuse

- The Challenge Is Not in the IP Itself, but is in the Component Integration Protocols
  - It's not just a "standard bus" problem
  - This is true for hardware, software, and so/rdware components
- Design Validation Remains the Key Bottleneck and is Likely to Get Even Harder

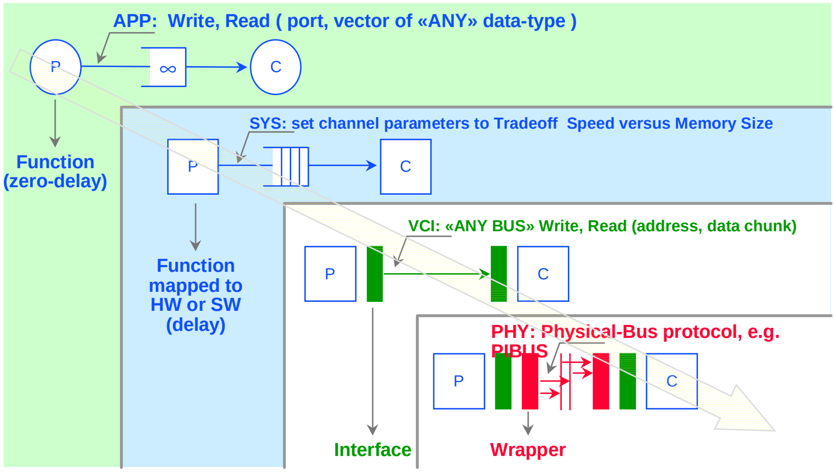
Communication-based Design



Pearls (the IP Processes)
MicroShells (the IP Requirements)
MacroShells (the Protocol Interface)
Communication Channels

COSY Communication Refinement

- Abstract from the concerns of HW or SW implementation (multi-target VC)
- Abstract from the concerns of a particular bus (bus-independent VC)



APP: Write, Read (port, vector of «ANY» data-type)

SYS: set channel parameters to Tradeoff Speed versus Memory Size

VCI: «ANY BUS» Write, Read (address, data chunk)

PHY: Physical-Bus protocol, e.g. PIBUS

Function (zero-delay)

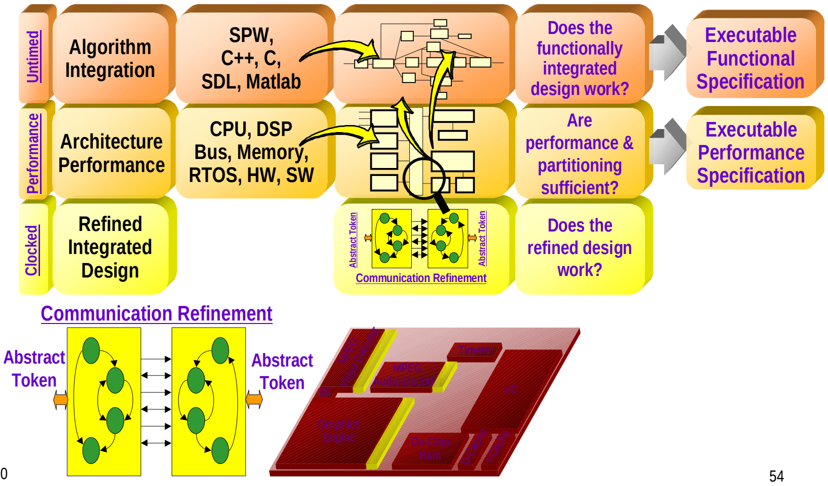
Function mapped to HW or SW (delay)

Interface

Wrapper

Communication Refinement

from Tokens to Signals



Untimed

Algorithm Integration

SPW, C++, C, SDL, Matlab

Does the functionally integrated design work?

Executable Functional Specification

Performance

Architecture Performance

CPU, DSP Bus, Memory, RTOS, HW, SW

Are performance & partitioning sufficient?

Executable Performance Specification

Clocked

Refined Integrated Design

Abstract Token

Communication Refinement

Abstract Token

Does the refined design work?

Communication Refinement

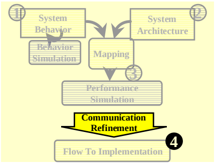
Abstract Token

Abstract Token

Key Technology

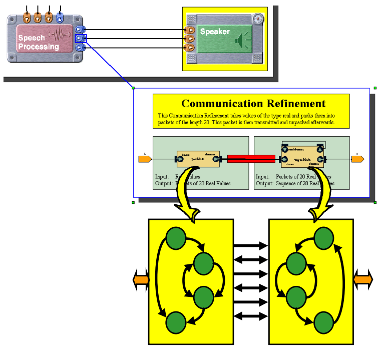
Communication Refinement

<ins>*Refinement from abstract tokens to articulated signals*</ins>



Value

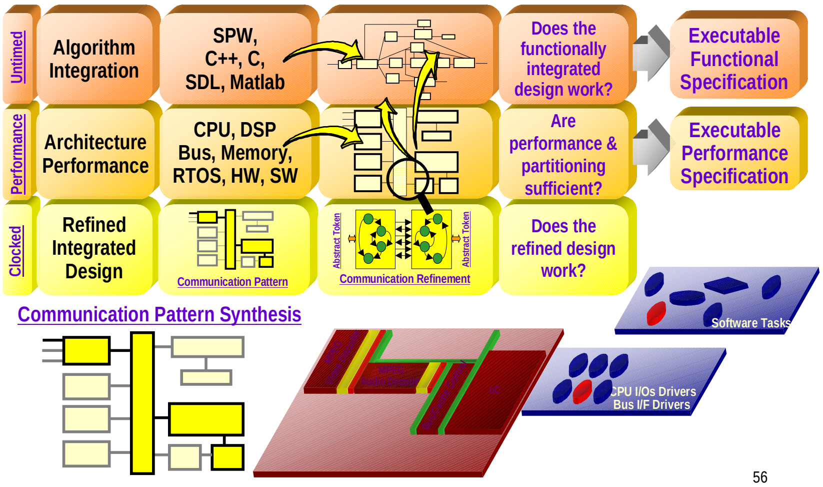
- ▲*Design* **and** *simulate* **at the level of abstraction at which designers think (e.g. ATM cell, GSM frame)**
- ▲*hide implementation details* **of the communication until it is required (but simulate it's overhead!)**
- ▲*refine* **from abstract token level down** *to implementation* **of interface signals**
- ▲*evaluate* **performance trade offs of** *communication effects*



Speech Processing
Speaker
Communication Refinement
i
packMats
o
unpackMats
Input: Real Values
Output: Packets of 20 Real Values
Input: Packets of 20 Real Values
Output: Sequence of 20 Real Values

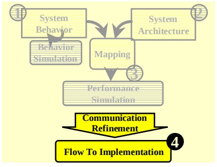
This Communication Refinement takes values of the type real and packs them into packets of the length 20. This packet is then transmitted and unpacked afterwards.

Communication Synthesis


Untimed
Algorithm Integration
SPW, C++, C, SDL, Matlab
Does the functionally integrated design work?
Executable Functional Specification
Performance
Architecture Performance
CPU, DSP Bus, Memory, RTOS, HW, SW
Are performance & partitioning sufficient?
Executable Performance Specification
Clocked
Refined Integrated Design
Communication Pattern
Abstract Token
Communication Refinement
Abstract Token
Does the refined design work?
Communication Pattern Synthesis
MPEG Video Decoder
MPEG Audio Decode
Bus/Cache Control
uC
Software Tasks
CPU I/Os Drivers
Bus I/F Drivers

Key Technology

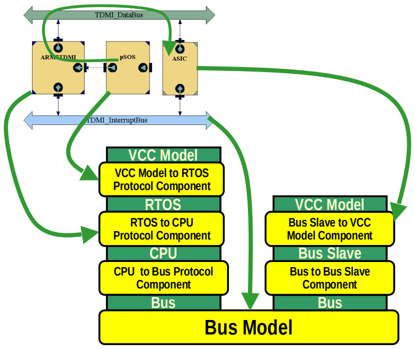
Communication Interface Synthesis



Synthesize communication pattern through architecture

Value

- ▲*Choose* **from comprehensive set of** *communication pattern*
- ▲**Pattern for** *HW-SW*, *SW-HW*, *HW-HW* **and** *SW-SW* **communication available**
- ▲*move function between HW and SW* **boundaries and** *re-synthesize* **the** *communication interface*
- ▲**customize platform communication environment through JAVA scripts**


TDMI_DataBus
ARM7TDMI
pSOS
ASIC
TDMI_InterruptBus
VCC Model
VCC Model to RTOS Protocol Component
RTOS
RTOS to CPU Protocol Component
CPU
CPU to Bus Protocol Component
Bus
VCC Model
Bus Slave to VCC Model Component
Bus Slave
Bus to Bus Slave Component
Bus
Bus Model

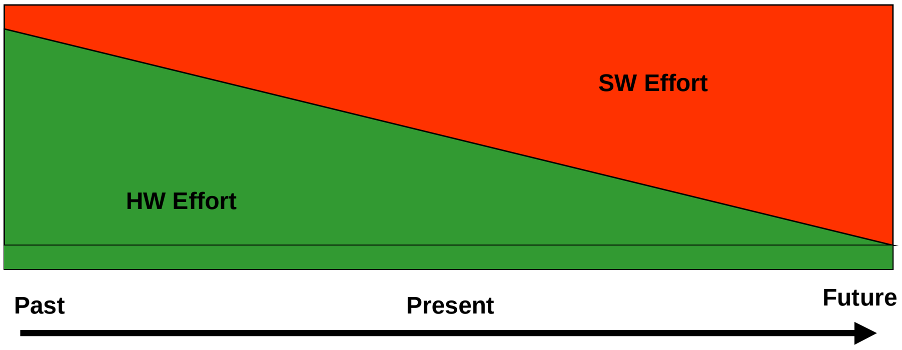
Outline

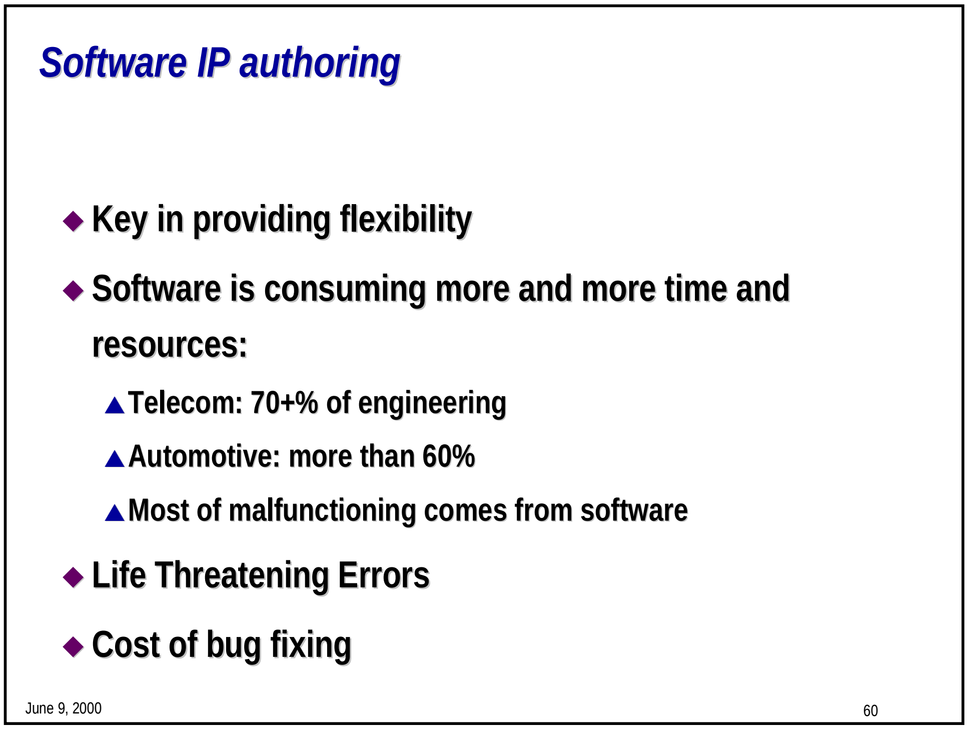
We are on the edge of a revolution in the way electronics systems are designed.

- Electronic Systems for the car
- Platform-based Design
- Embedded Software

Why the Software Productivity Concern??

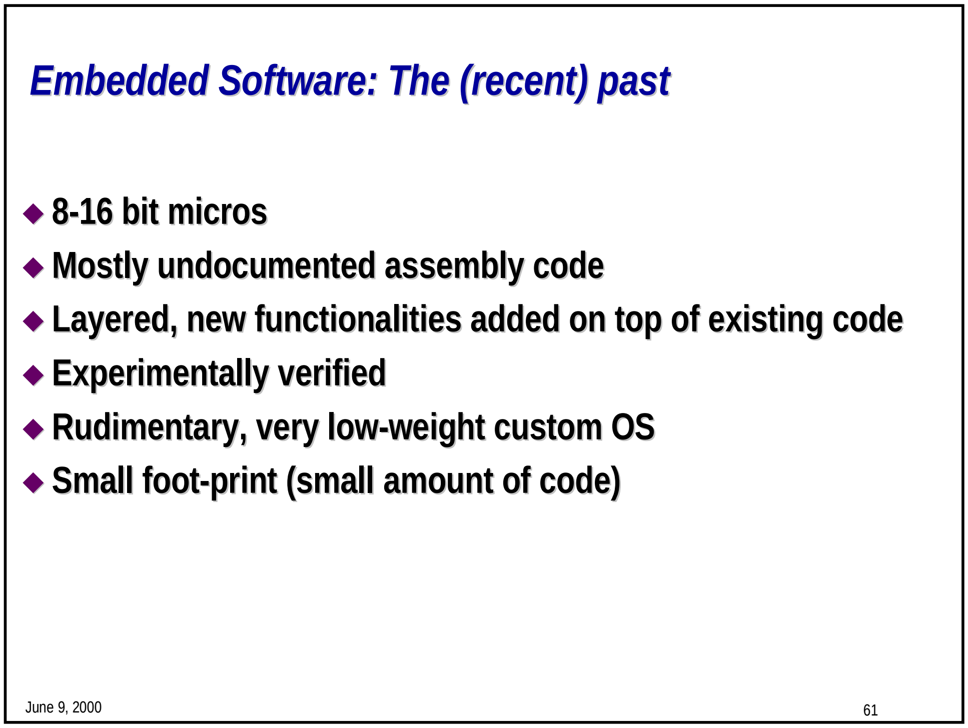
- In the end; if we solve the HW design productivity and fail to address the SW productivity we have accomplished little.





Software IP authoring

- Key in providing flexibility
- Software is consuming more and more time and resources:
  - Telecom: 70+% of engineering
  - Automotive: more than 60%
  - Most of malfunctioning comes from software
- Life Threatening Errors
- Cost of bug fixing



Embedded Software: The (recent) past

- 8-16 bit micros
- Mostly undocumented assembly code
- Layered, new functionalities added on top of existing code
- Experimentally verified
- Rudimentary, very low-weight custom OS
- Small foot-print (small amount of code)

Embedded Software: The present

- 32-bit high performance micros
- 100,000-1,000,000 lines of code with shorter and shorter time-to-market
- Mostly low level C-code (Micro-controllers) or assembly for DSPs
- Commercial RTOS (e.g., Wind River)
- Verification is a real challenge

Embedded Software: Requirements

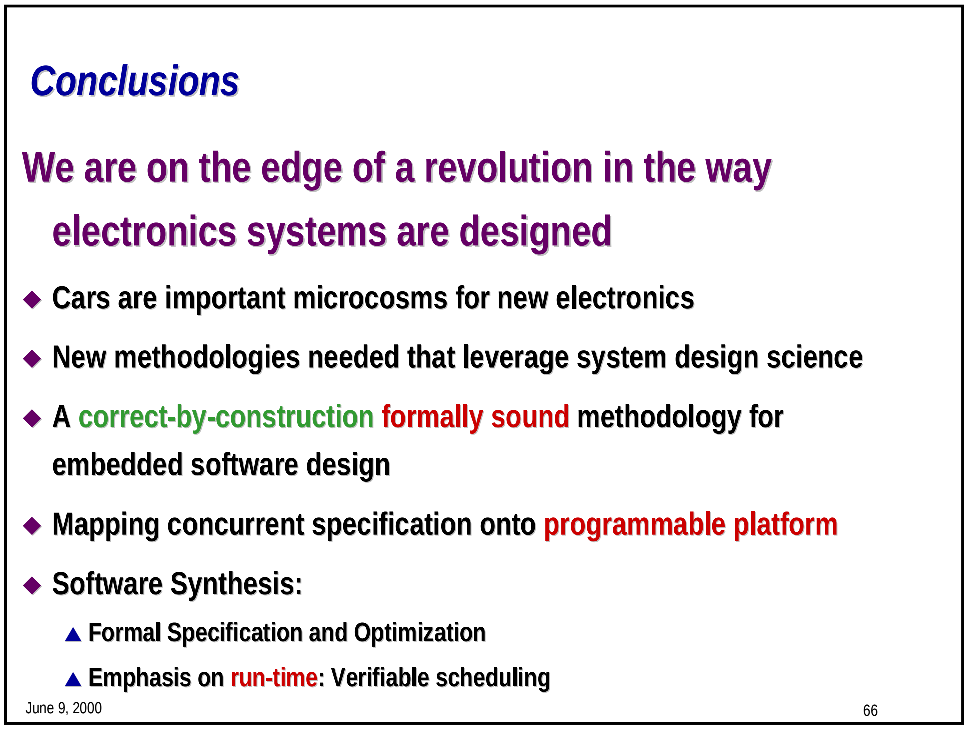
- Safety
  - full (formal?) verification
- Productivity
  - smaller number of software designers, much larger systems
    - “new” designs from 60,000 man/days for automotive engine control to 20,000
    - spins (e.g., new customer for same basic product) from 20,000 to 5,000
- Cost
  - max leverage of available architectures (being able to convert quickly software from one platform to another!)

Embedded Software: Agenda

- Raise levels of abstraction
- Formal models and techniques
- Simulation and Estimation for Platform Selection
- Automatic “synthesis” and assembly of components
  - highly optimized, correct by construction
- Application-driven: Engine Control (e.g., Magneti-Marelli, ST, Daimler and BMW) quite different from Digital Video Decoder (Philips) and from Wireless (BWRC and Ericsson)

System Building Focus

- Provide background, methodology and experience in system building.
- Designs will build on a variety of disciplines including computer hardware, communications, DSP, IC design, networks, operating systems and software.
- Make use and understand the advantages and limitations of CAD tools.



Conclusions

We are on the edge of a revolution in the way electronics systems are designed

- Cars are important microcosms for new electronics
- New methodologies needed that leverage system design science
- A correct-by-construction formally sound methodology for embedded software design
- Mapping concurrent specification onto programmable platform
- Software Synthesis:
  - Formal Specification and Optimization
  - Emphasis on run-time: Verifiable scheduling