Embedded System Design: From Art to Science

Alberto Sangiovanni-Vincentelli
Founder and Scientific Director, PARADES

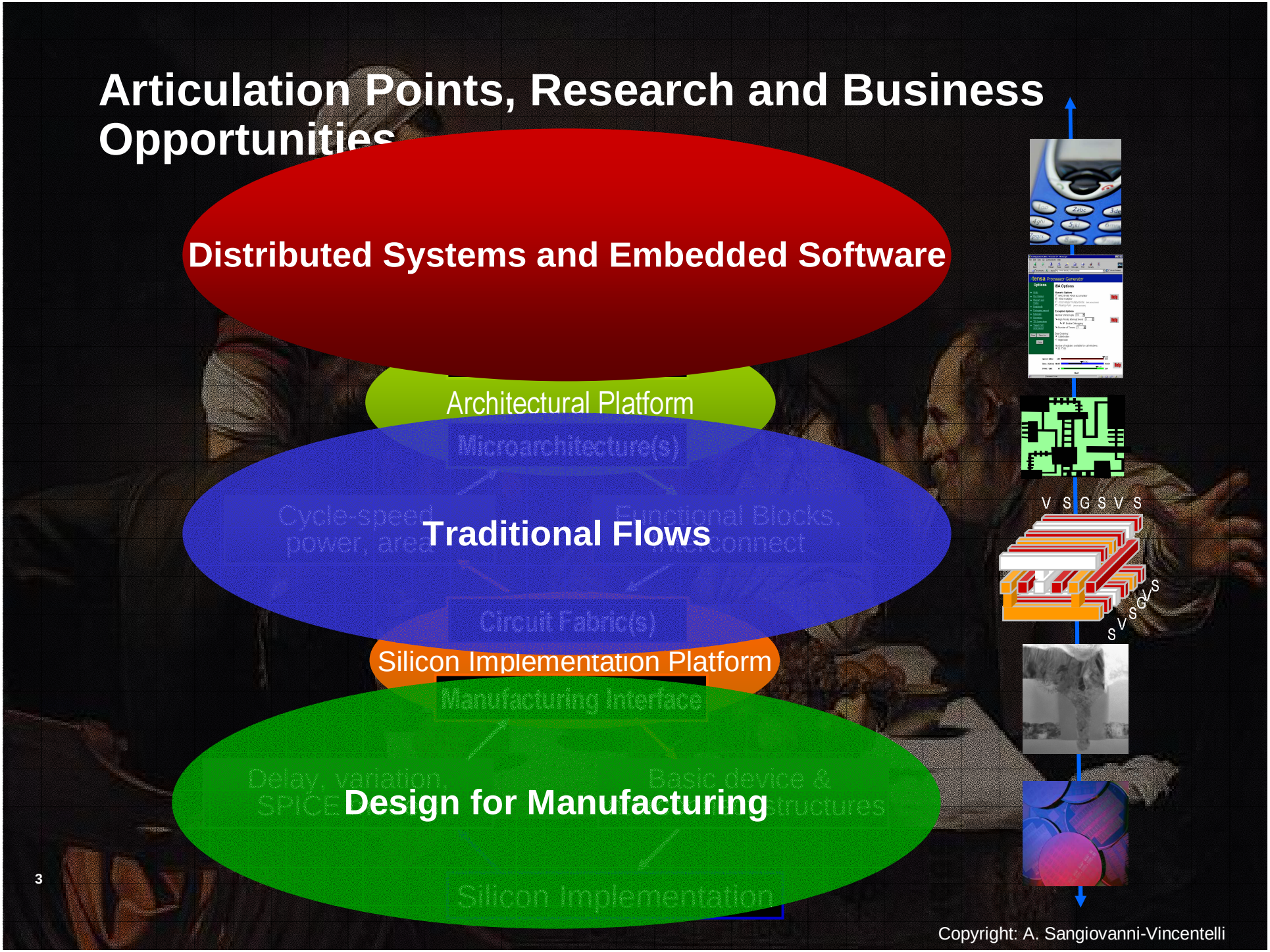
The Edgar L. and Harold H. Buttner Chair of EECS
University of California at Berkeley

Co-Founder, CTA and Member of the Board,
Cadence Design Systems

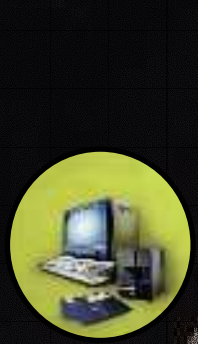
Outline

- Embedded Software Status in Industry
- Platform-based Design
- Applications of PBD to Distributed Building Control
- A synthesis method
- Metropolis
- Automotive and Multi-media

Articulation Points, Research and Business Opportunities

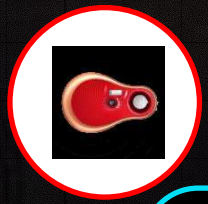


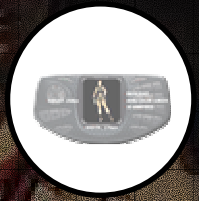
Challenge: The Changing Face Of The Consumer



Late 80s to mid 90s

Bulky. Expensive.
'Executive Jewelry'

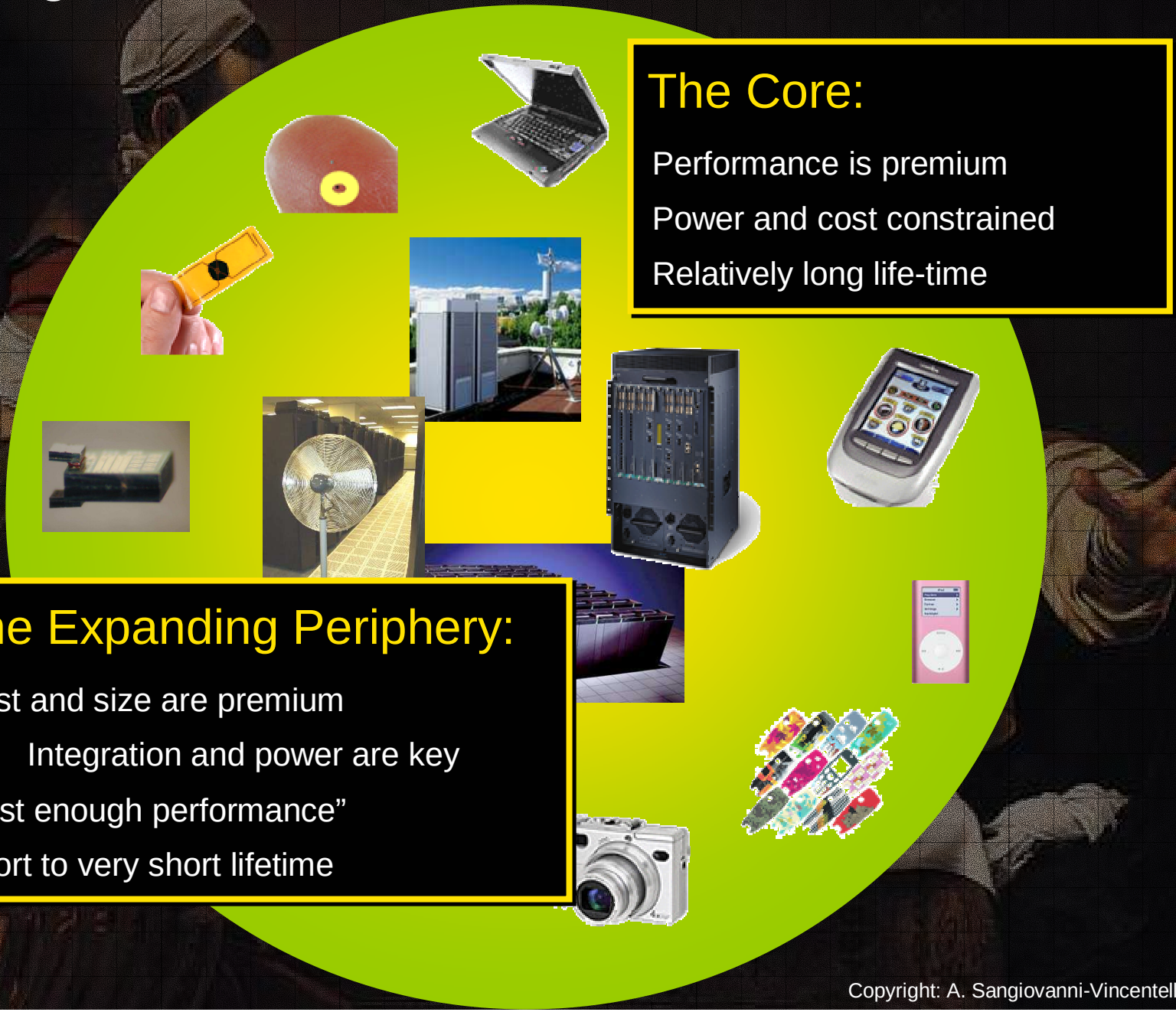




2000s

Cooler. Cheaper. Faster.
'Electronic Fashion'

Challenge: The Bifurcation of the Market



The Core:

Performance is premium

Power and cost constrained

Relatively long life-time

The Expanding Periphery:

Cost and size are premium

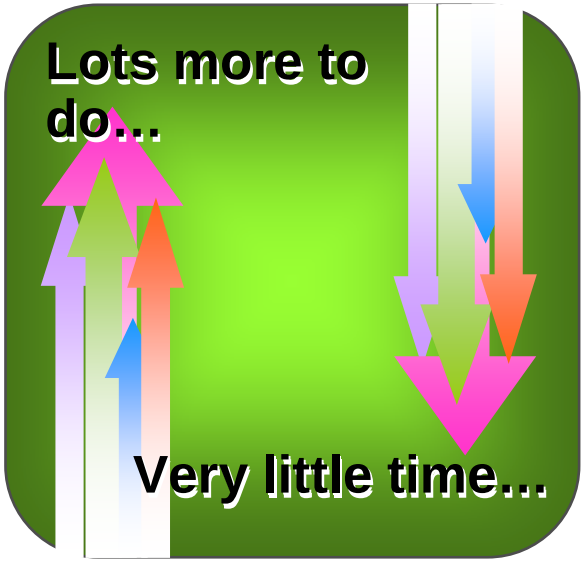
Integration and power are key

“Just enough performance”

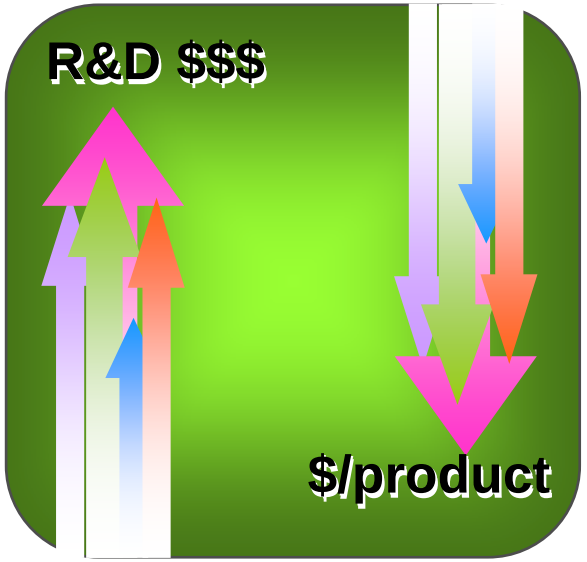
Short to very short lifetime

Lots In The Way…

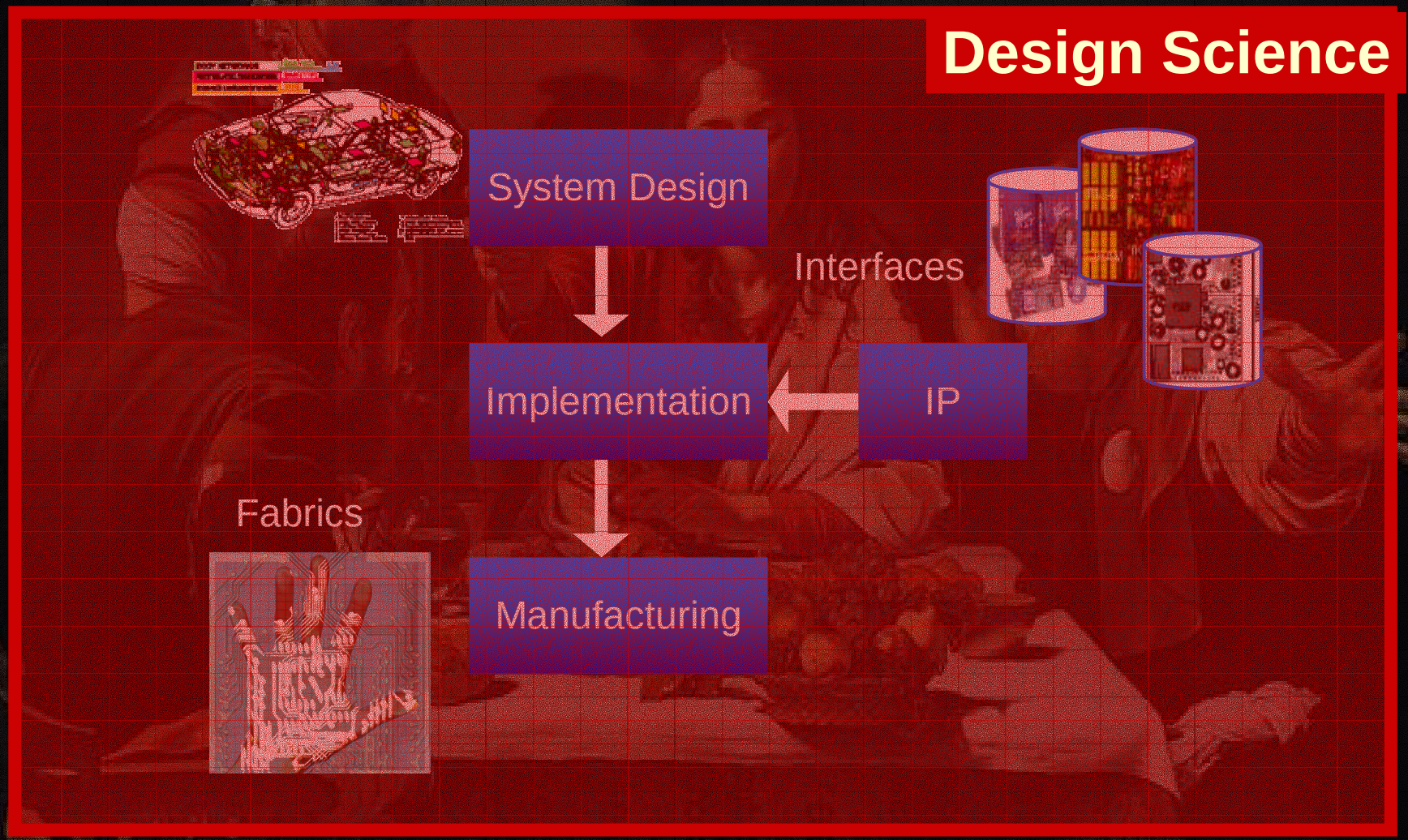
Dizzying list of technology challenges



And the need for yield

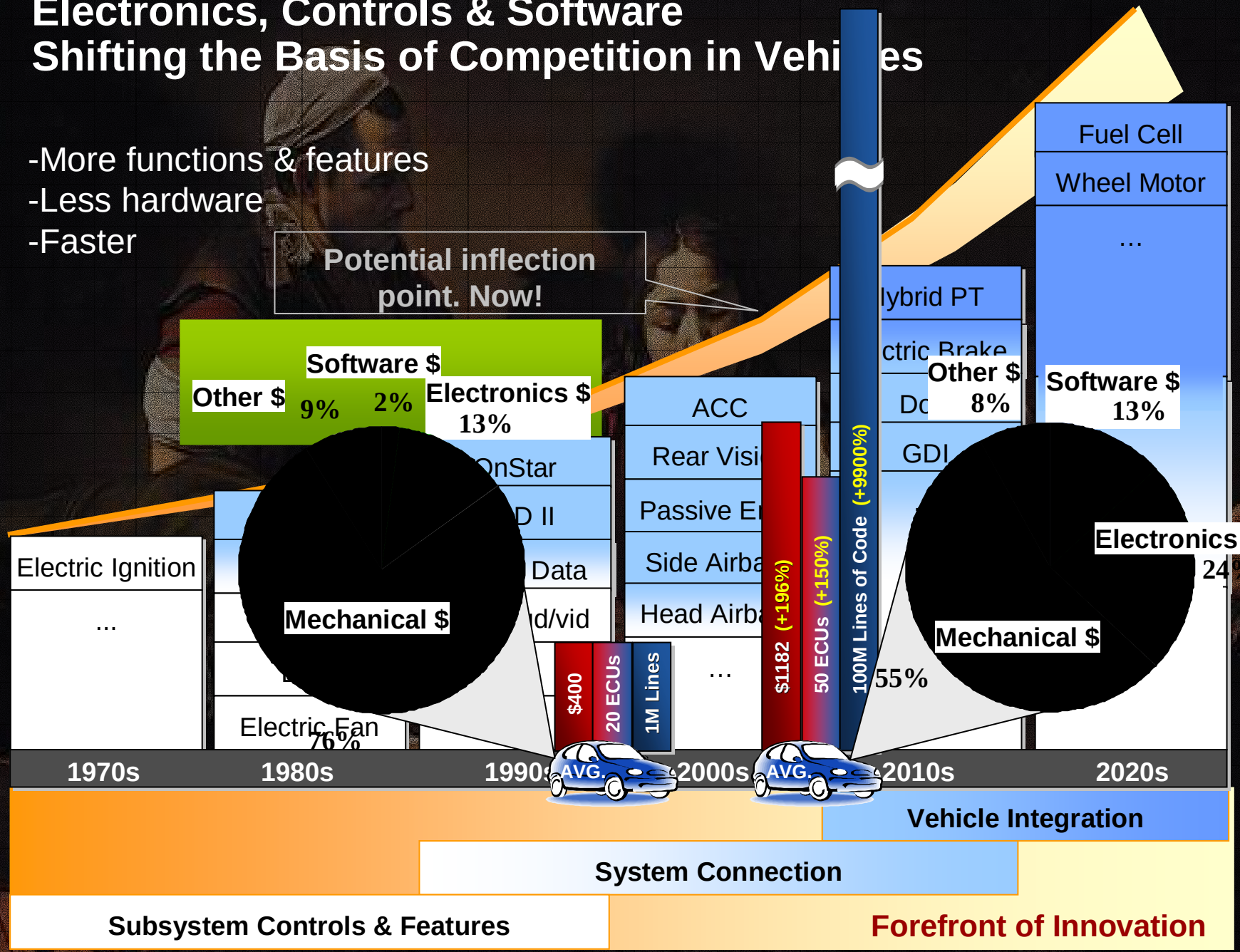


Opportunity: Electronic Systems Design Chain



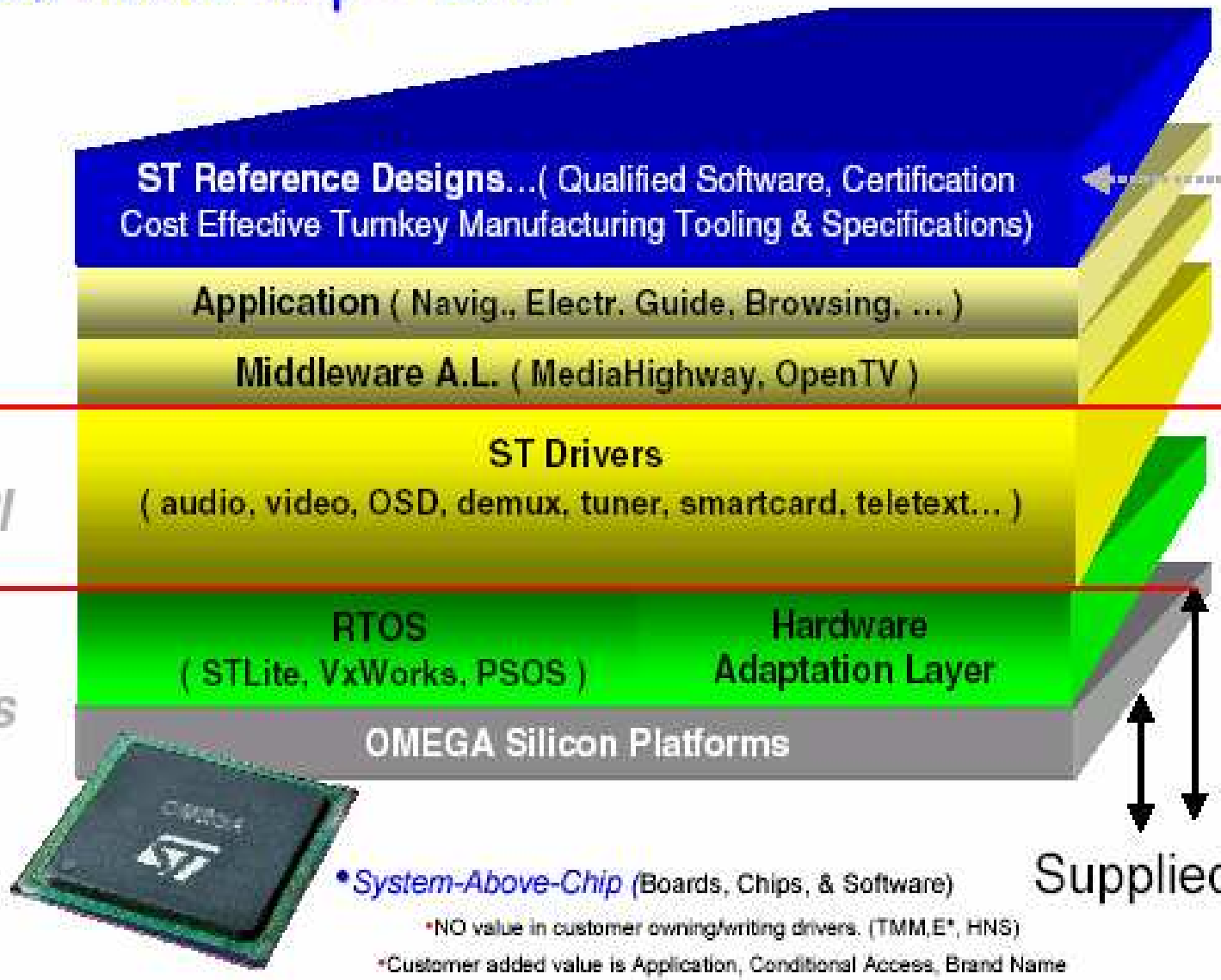
Electronics, Controls & Software Shifting the Basis of Competition in Vehi es

-More functions & features
-Less hardware
-Faster



Source: Matt Tsien, GM

System Above Chip - SAC



- *System-Above-Chip* (Boards, Chips, & Software)
  - NO value in customer owning/writing drivers. (TMM,E*, HNS)
  - Customer added value is Application, Conditional Access, Brand Name
- ST supplies the complete base system BELOW MIDDLEWARE to save time to market

Automotive Industry
Three Levels of Players

Automakers

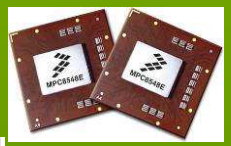
- 2005 Revenue: $1.1T
- CAGR 2.8% (2004-2010)

Tier 1 Suppliers

90%+ of revenue from automotive

- 2004 Revenue ~$200B
- CAGR 5.4% (2004-2010)

IC Vendors

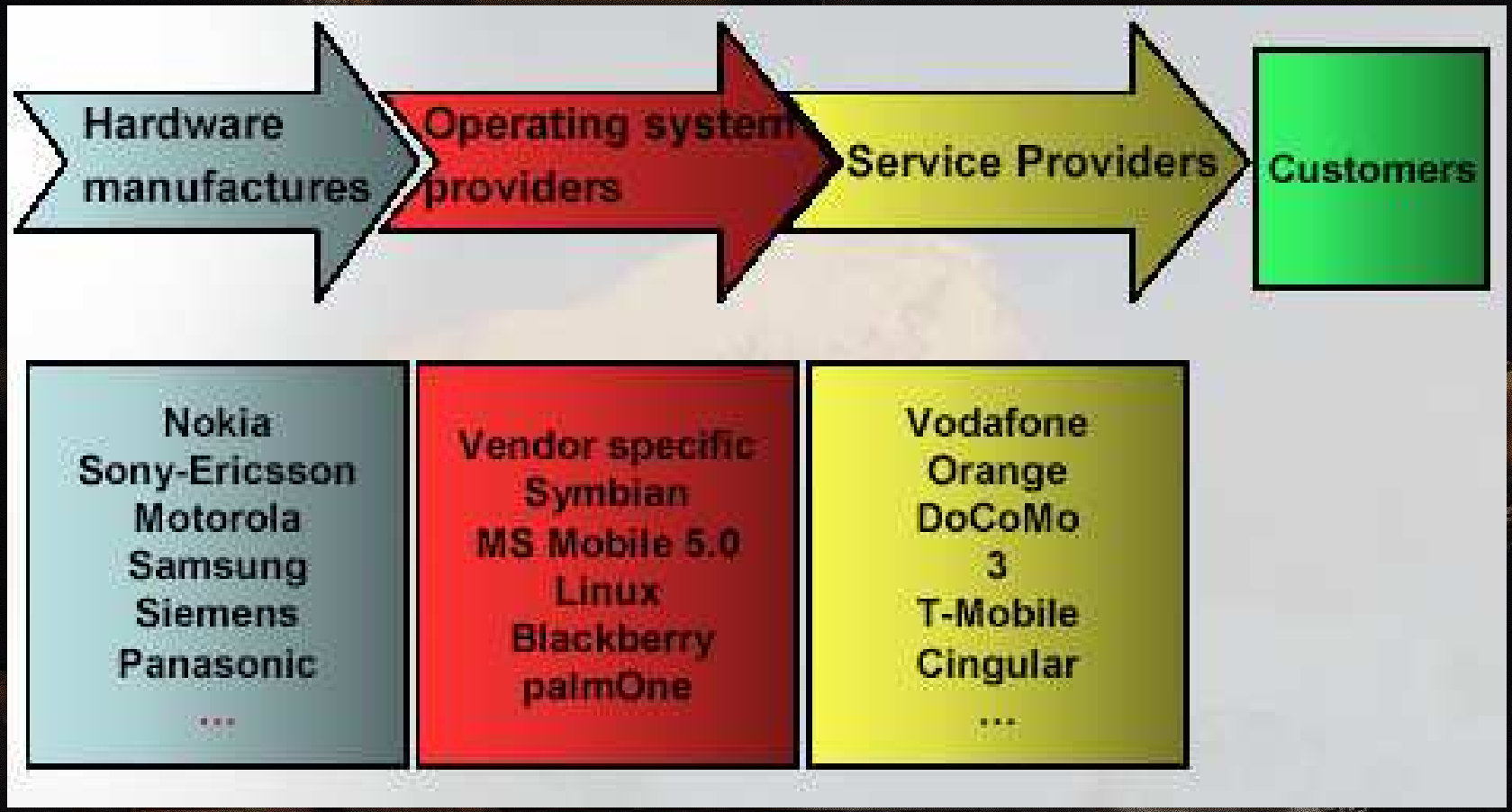


~15% of revenue from automotive

- 2005 revenue $17.4B
- CAGR 10% (2004-2010)

Source: Public financials, Gartner 2005

The Design Chain for Cellular Phones



How Safe is Our Real-Time Software?

Source: E. Lee

Software Bugs Cost $59.5 Billion A Year

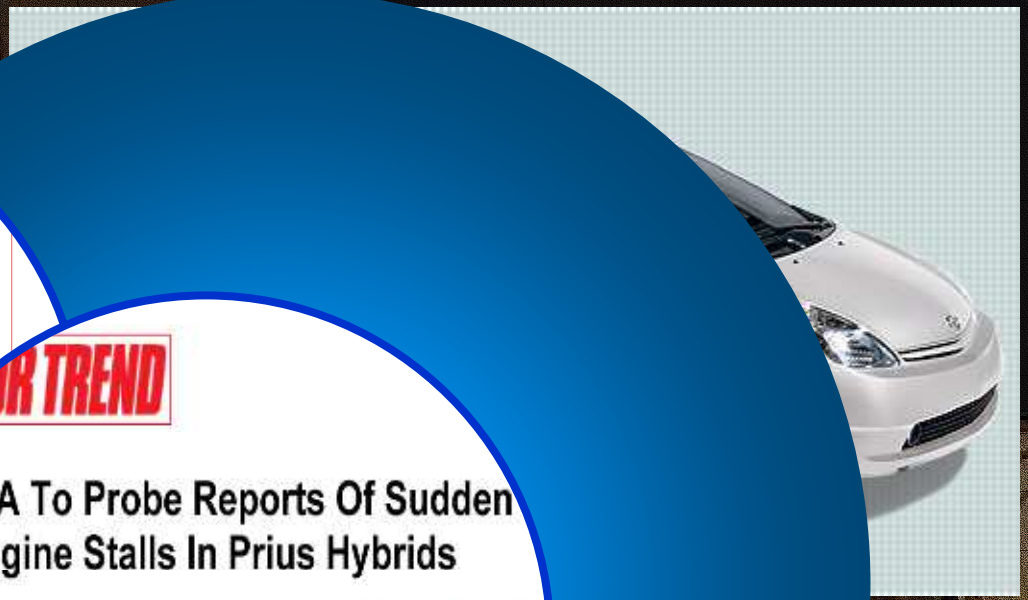
INFOWORLD, JUNE 28, 2002 – BY PAUL KRILL

Software bugs cost $59.5 billion a year, study says

Software bugs cost the U.S. economy an estimated $59.5 billion per year, or 0.6 percent of the gross domestic product, according to a newly released study by the U.S. Department of Commerce National Institute of Standards and Technology (NIST). In a statement released on Friday, NIST said more than half the costs are borne by software users and the remainder by software developers and vendors.

Additionally, the study found that although errors cannot be removed, more than a third of the costs, or an estimated $22.2 billion, could be eliminated by improved testing that enables earlier and more effective identification and removal of defects. Currently, more than half of errors are not found until “downstream” in the development process or during post-sale use of software, according to NIST.

We Live In An Imperfect World!



SUNDAY, FEBRUARY 6, 2005 – THE NEW YORK TIMES (by Tim Mo

What's Bugging the High-Tech Car?

On a hot summer trip to Cape Cod, the Mills family minivan did a peculiar thing. After an hour on the road, it began to bake the children. Mom and Dad were cool and comfortable up front, but heat was blasting into the rear of the van and it could not be turned off.

Fortunately for the Mills children, their father – W. Nathaniel Mills III, an expert on computer networking at I.B.M. – is persistent. When three dealership visits, days of waiting and the cumbersome replacement of mechanical parts failed to fix e problem, he took the van out and drove il the oven fired up again. Then he the mechanic to look for a

Additionally, the study found that although errors cannot be removed, more than a"It took two minutes for them to hook up their diagnostic tool and find the fault," said Mills, senior technical staff member I.B.M.'s T.J. Watson Research C Hawthorne, N.Y. "I can almo software code; a sensor wa

Indeed, the high-tech co confused. The rear te the 2001 Dodge van h sending a signal that freezing at 32 degree loyal van was dogge up. third of the costs, billion, could be elimin

OR TREND

NHTSA To Probe Reports Of Sudden Engine Stalls In Prius Hybrids

The National Highway Traffic Safety Administration said yesterday it is investigating reports that a software problem can cause the engine of Toyota's Prius hybrid to stall without warning at highway speeds. No accidents have been reported thus far.

TSA has received 33 reports of stalling in Prius cars model years 2004 and 2005, according to the 's initial report. More than 85 percent of the cars d did so at speeds between 35 and 65 miles

Real-Time Multitasking: Plug and Play?

Plug and Pray!

Compositionality

Non-compositional formalisms lead to very awkward architectures.

Source: Ed Lee

The Embedded System Design Nightmare

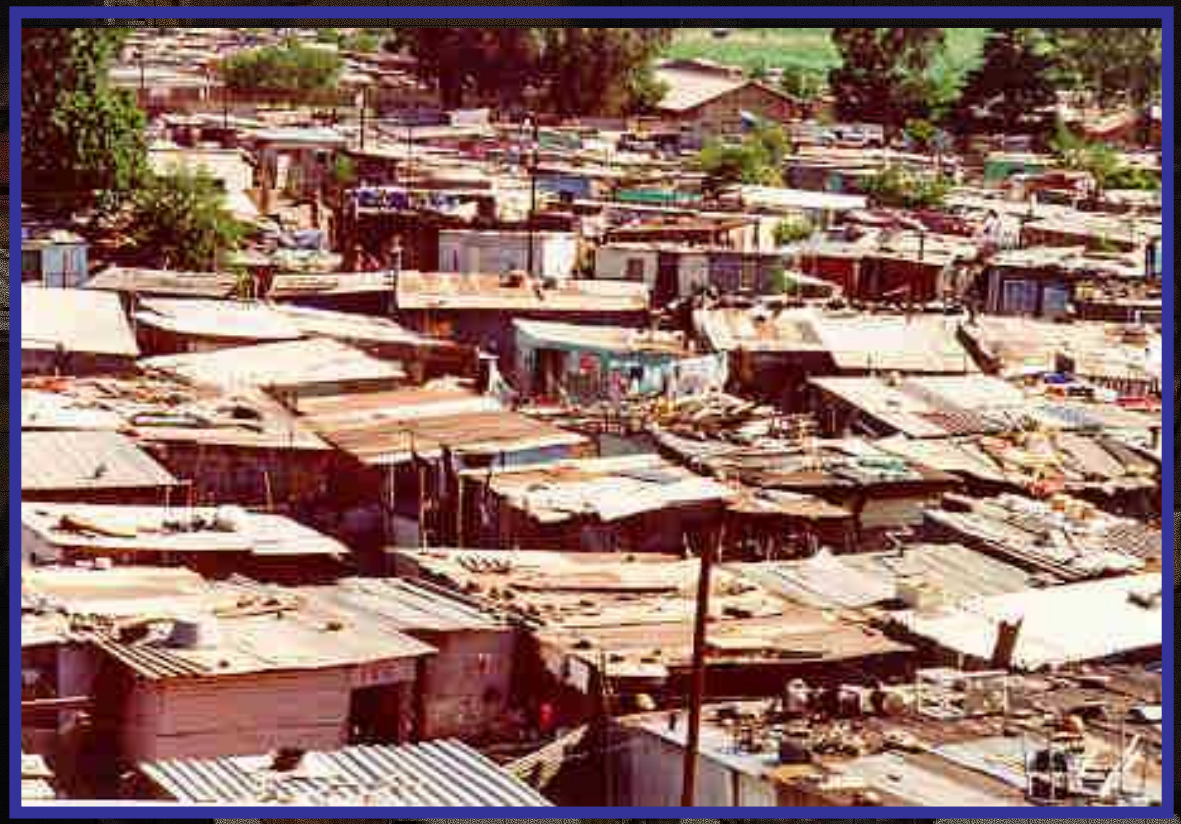
Specification:

Implementation:

P. Picasso “Femme se coiffant” 1940

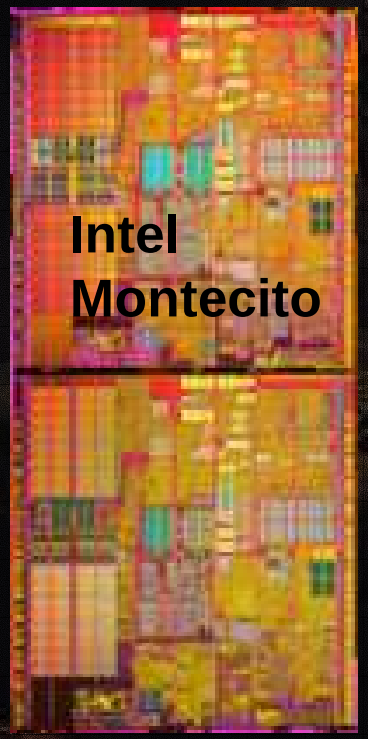
P. Picasso, Blue Period

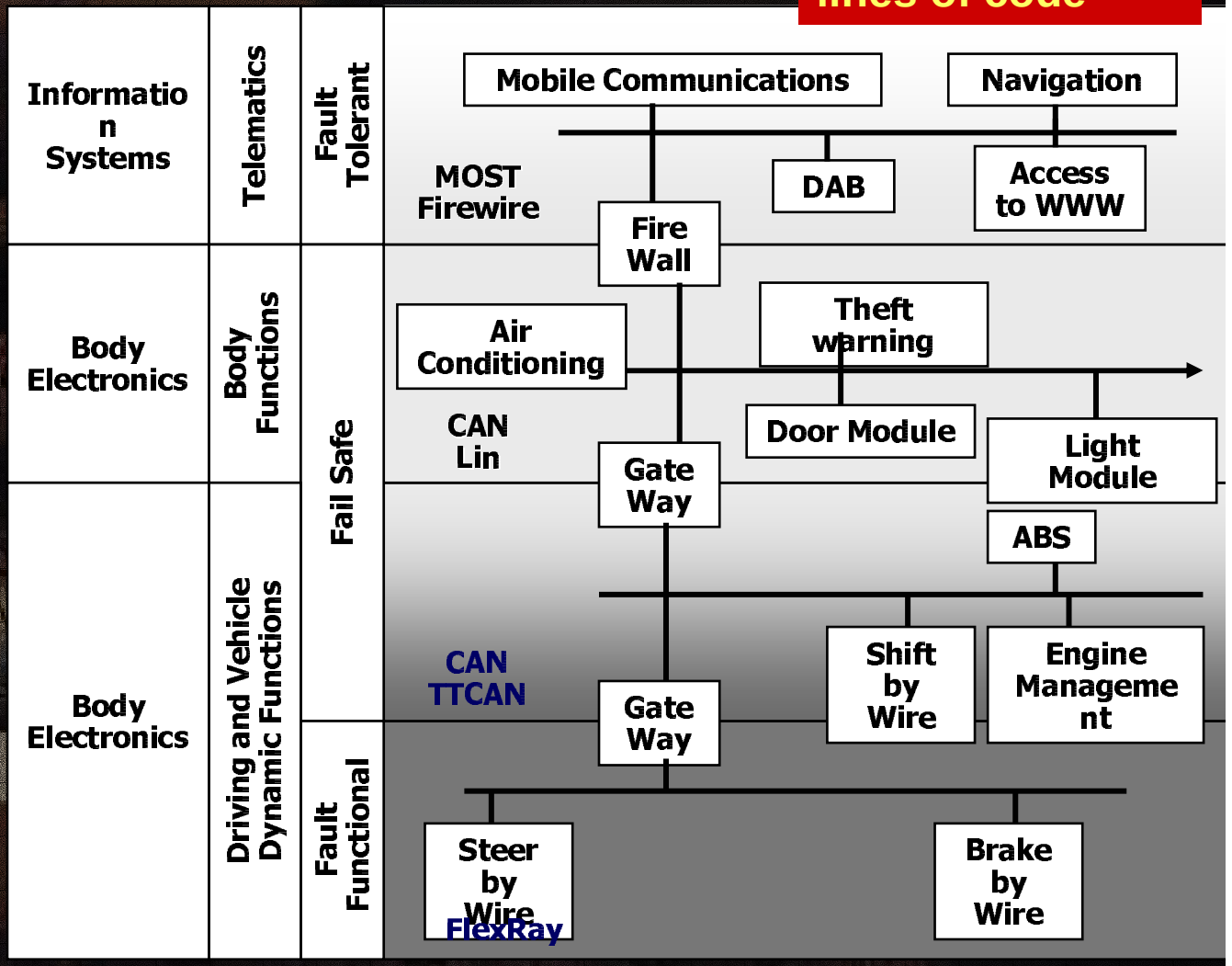
Embedded Software Architecture Today



Concurrency and Heterogeneity

Today, more than 80 Microprocessors and millions of lines of code

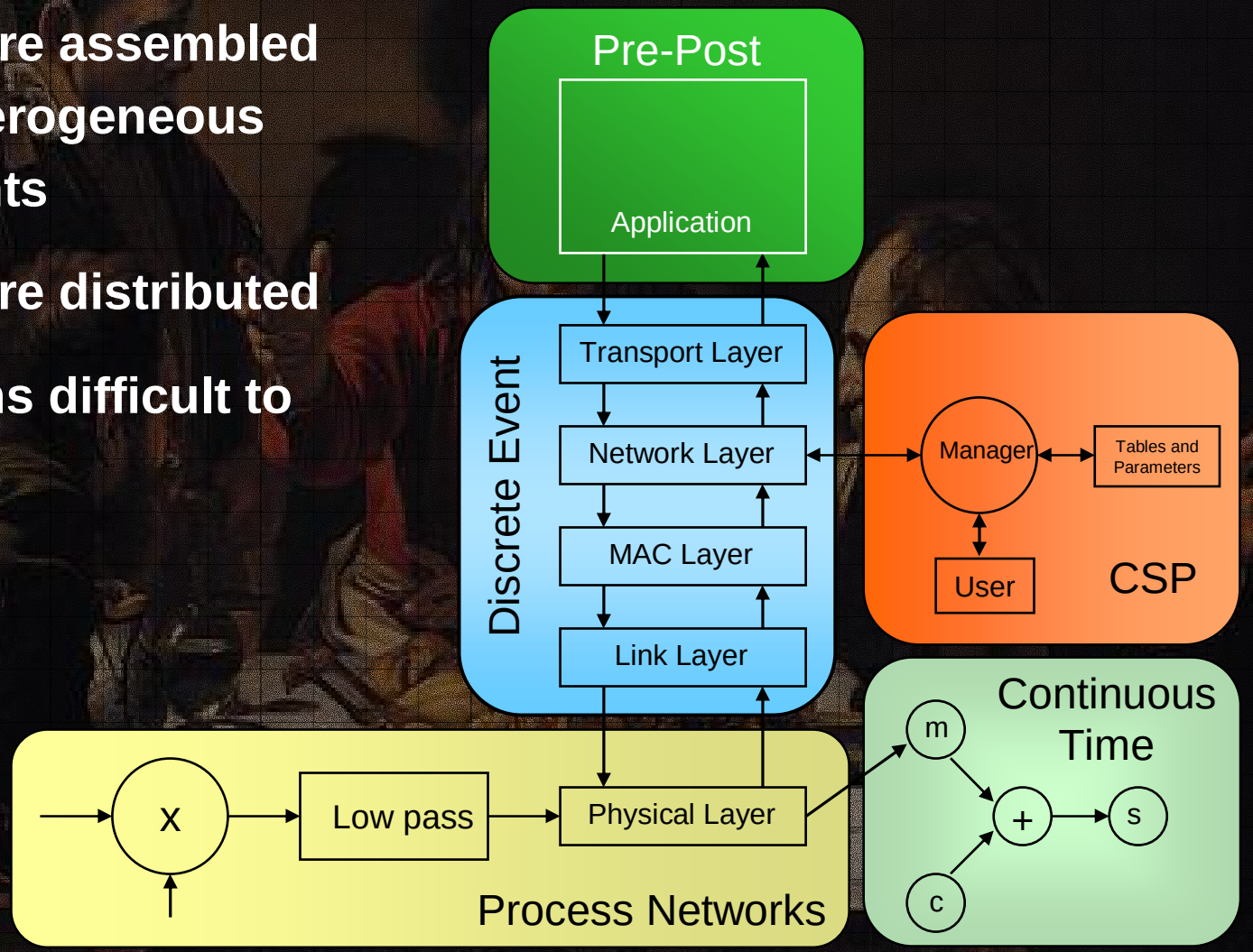




Source: Bosch

Common features

- Systems are assembled out of heterogeneous components
- Systems are distributed
- Interactions difficult to define



Embedded Software Systems

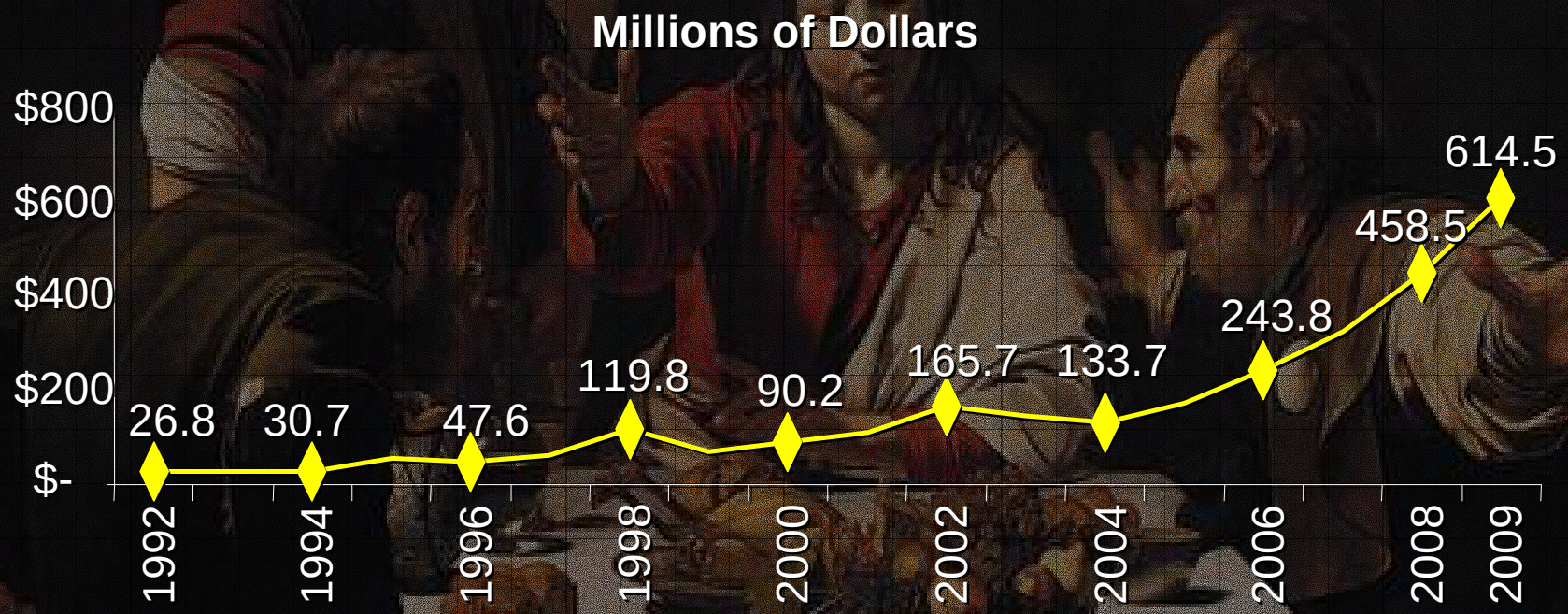
- **Computational**
  - but not first-and-foremost a computer
- **Integral with physical processes**
  - sensors, actuators
- **Reactive**
  - at the speed of the environment
- **Heterogeneous**
  - hardware/software, mixed architectures
- **Networked**
  - shared, adaptive

The Intellectual Agenda

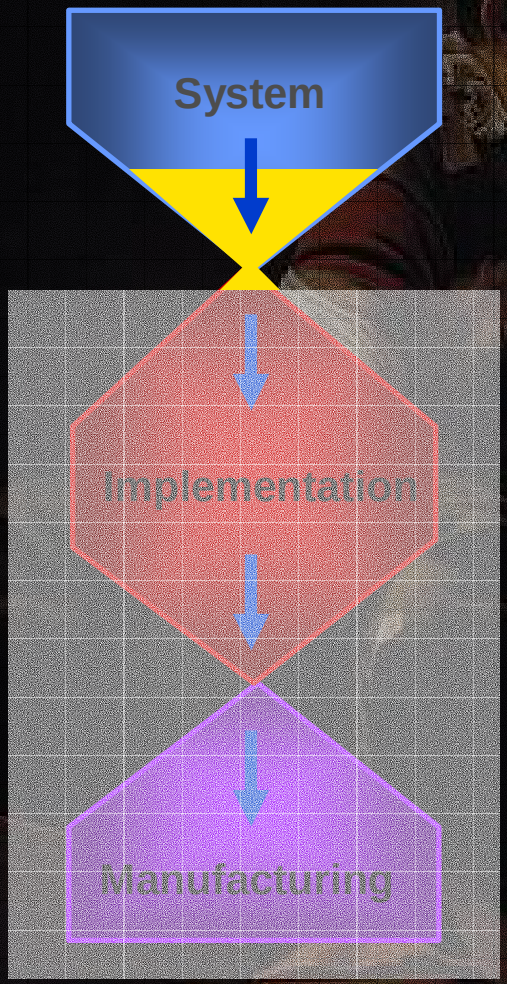
To create a modern computational systems science and systems design practice with

- **Concurrency**
- **Composability**
- **Time**
- **Hierarchy**
- **Heterogeneity**
- **Resource constraints**
- **Verifiability**
- **Understandability**

ESL History



Design Process Transformation



Embedded System Design Gaps

NO FORMAL SEMANTICS

MANUAL

MANUAL

MANUAL

MANUAL

Functional
High Level
HW MODELS

C/C++ SW
CODE

RTOS
CODE

Driver
CODE

Test
Vectors

Cycle Accurate RTL

Validate

Validate

Simulate
RTL

Synthesis

Verify
Gates

Timing Accurate
Gate-Level Netlist

Validate

intel

Source: Greg Spirakis

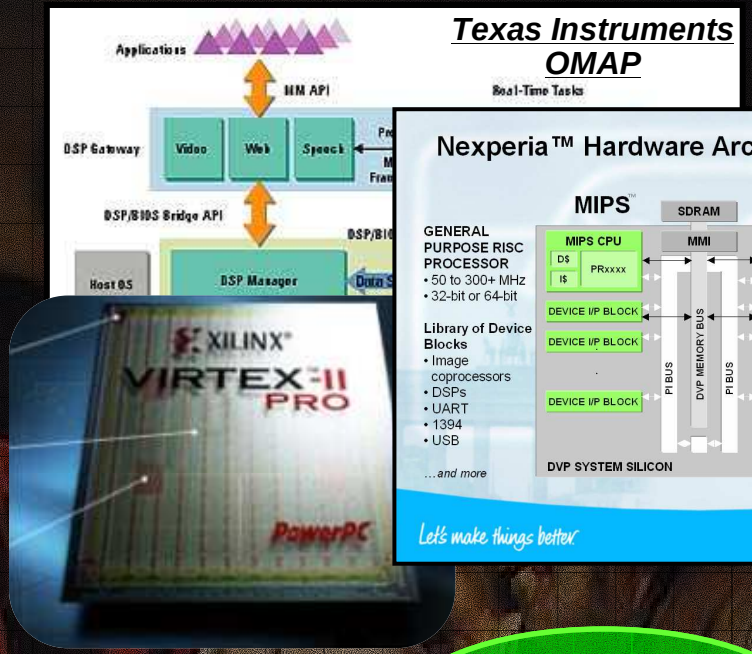
6

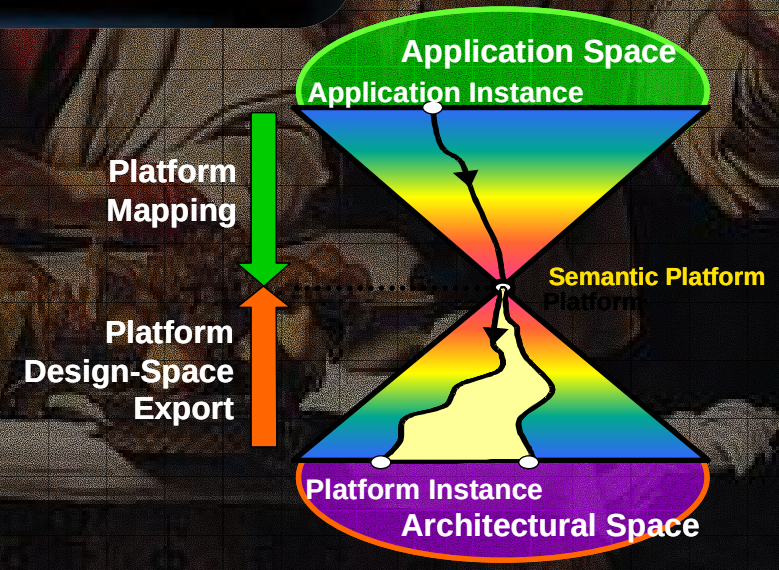
Outline

- Embedded Software Status in Industry
- Platform-based Design
- Applications of PBD to Distributed Building Control
- A synthesis method
- Metropolis
- Automotive and Multi-media

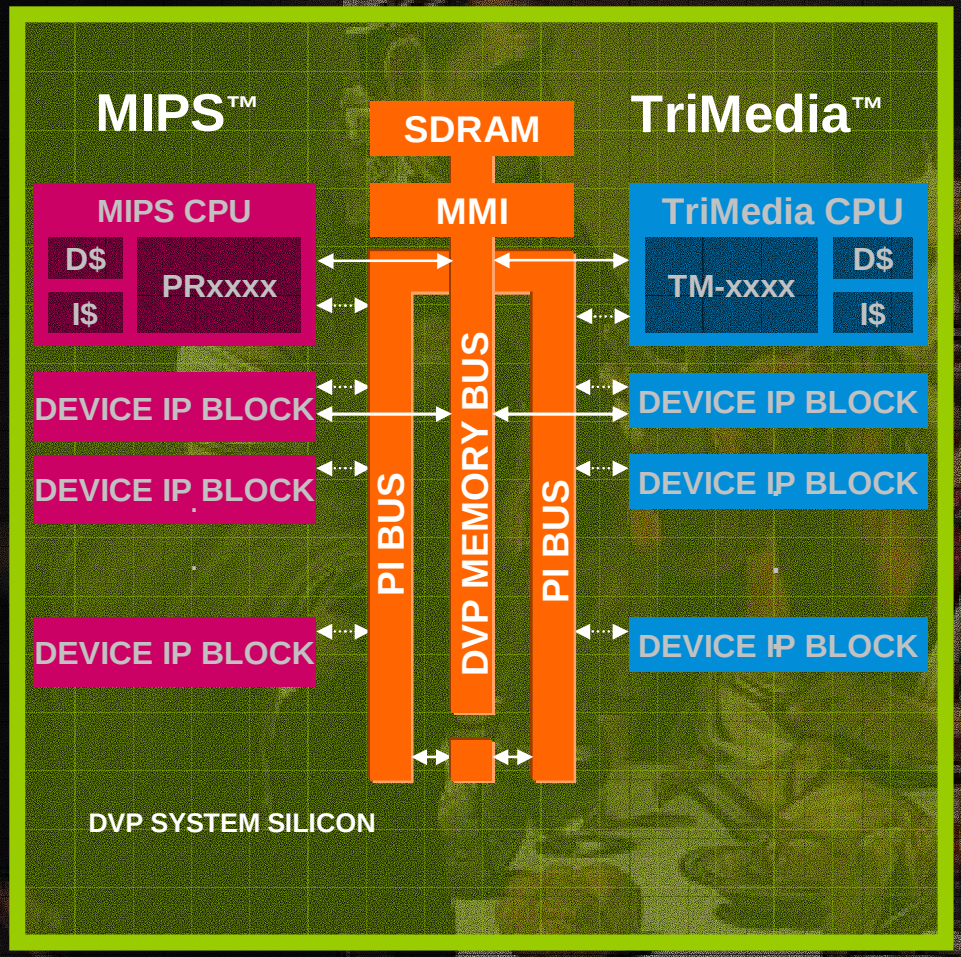
The Platform Concept

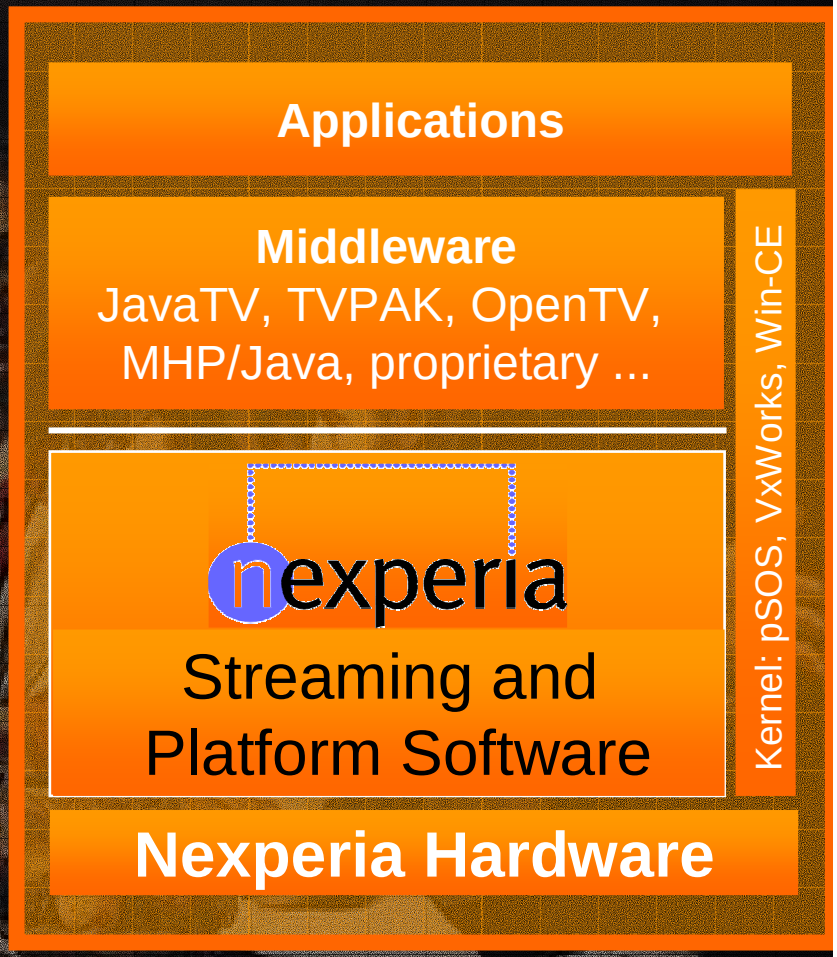
- Meet-in-the-Middle Structured methodology that limits the space of exploration, yet achieves good results in limited time;
- A formal mechanism for identifying the most critical hand-off points in the design chain;
- A method for design re-use at all abstraction levels;
- **An intellectual framework for the complete electronic design process!**





Early Platform Architecture:
Philips *Nexperia*

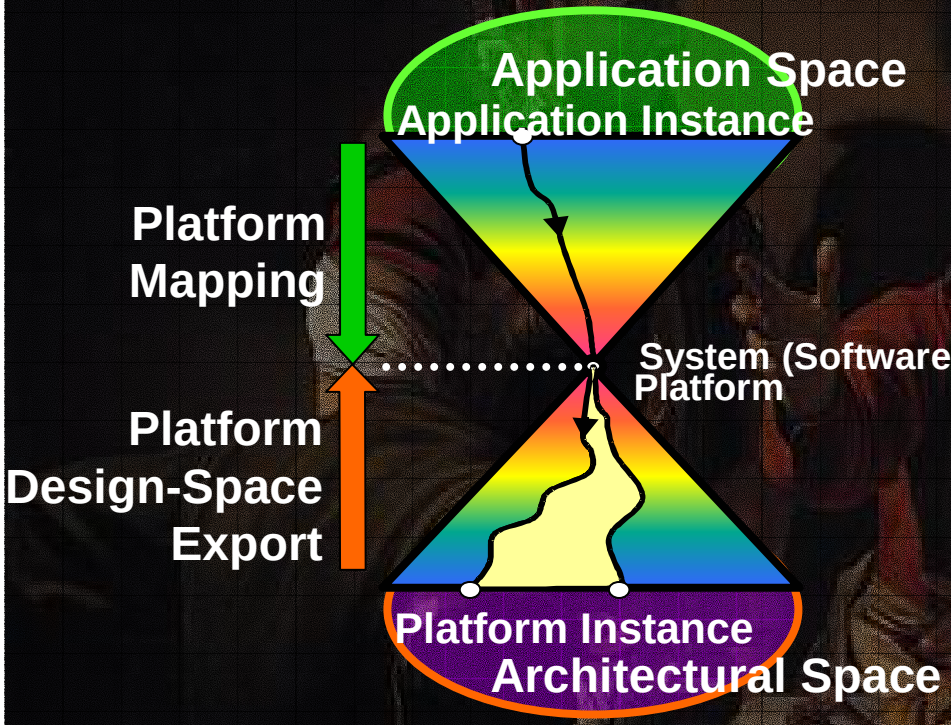


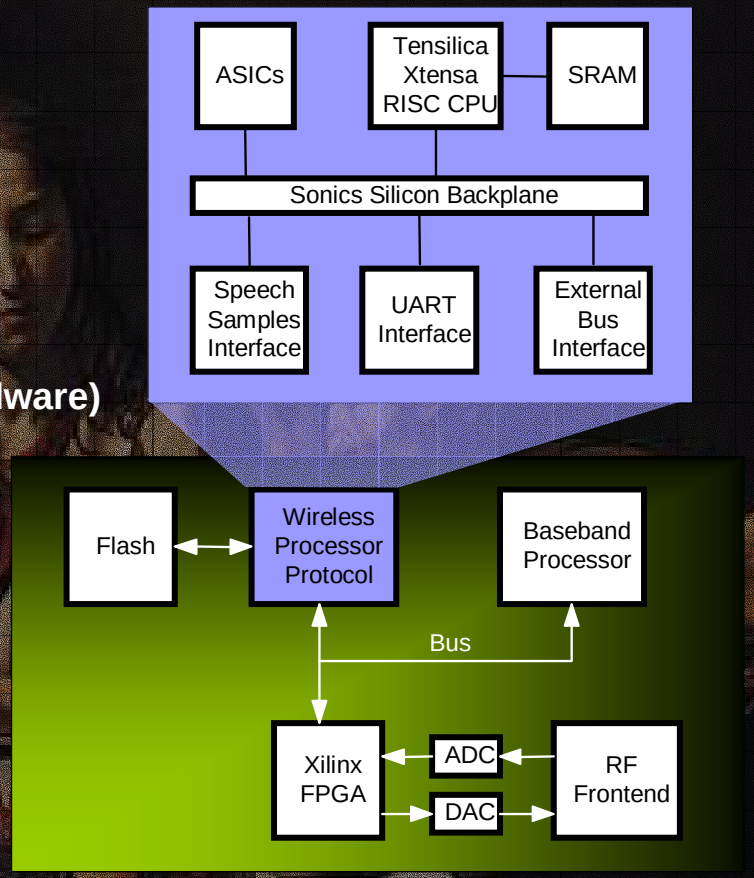


Hardware **Software**

Source: Philips

Platform-based Design
(ASV Triangles 1998)

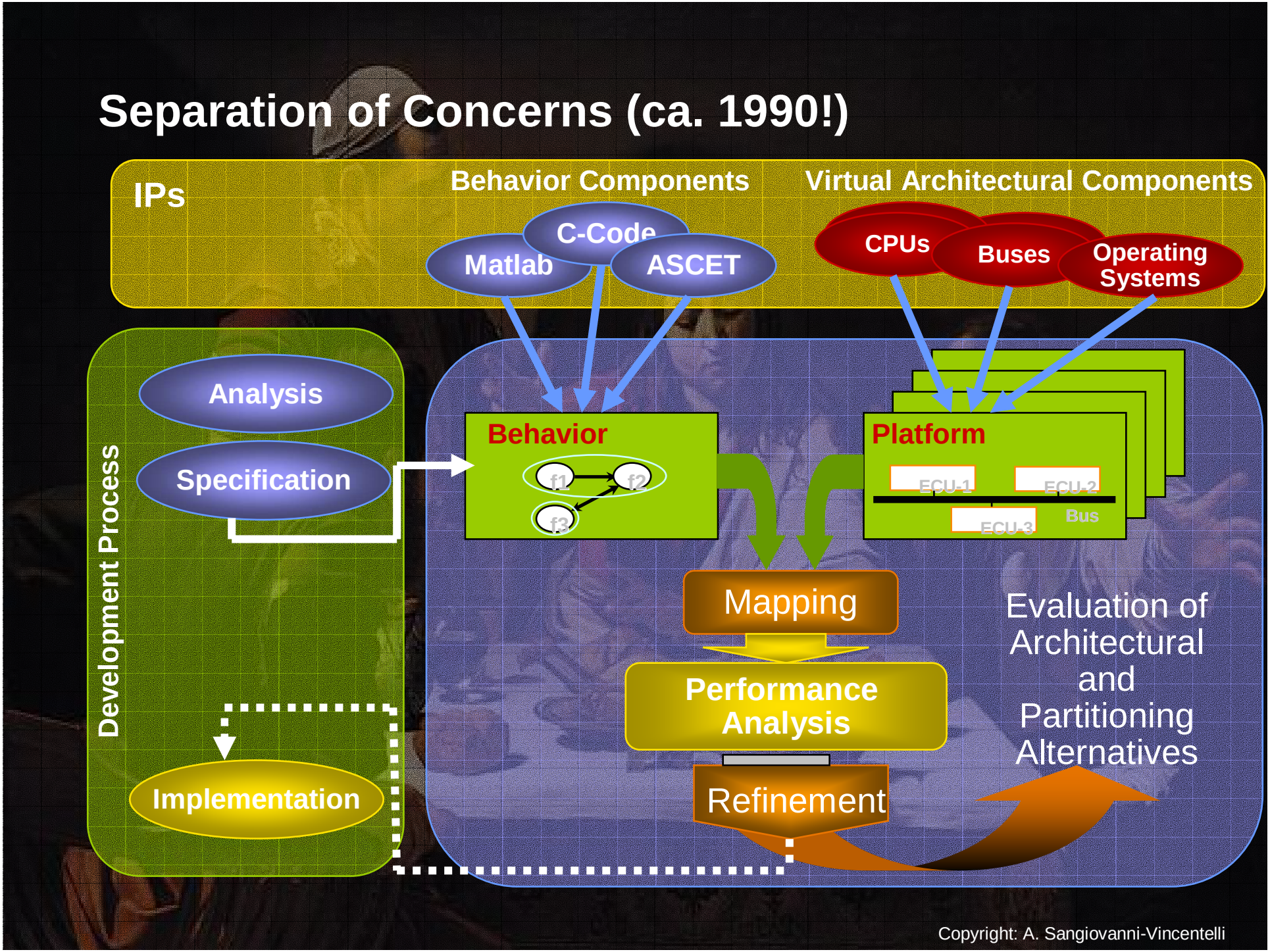




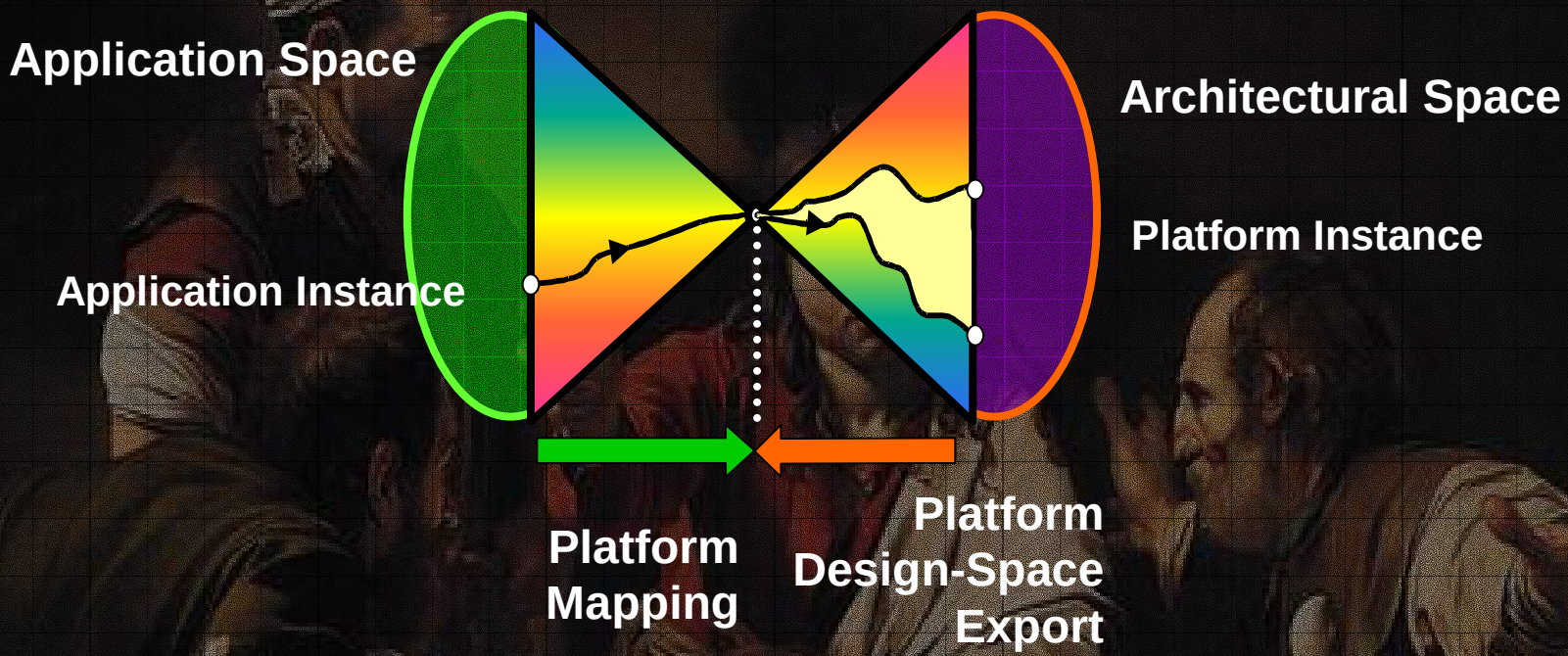
Intercom Platform (BWRC, 2001)

- **Platform: library of resources defining an abstraction layer**
  - **hide unnecessary details**
  - **expose only relevant parameters for the next step**

Separation of Concerns (ca. 1990!)

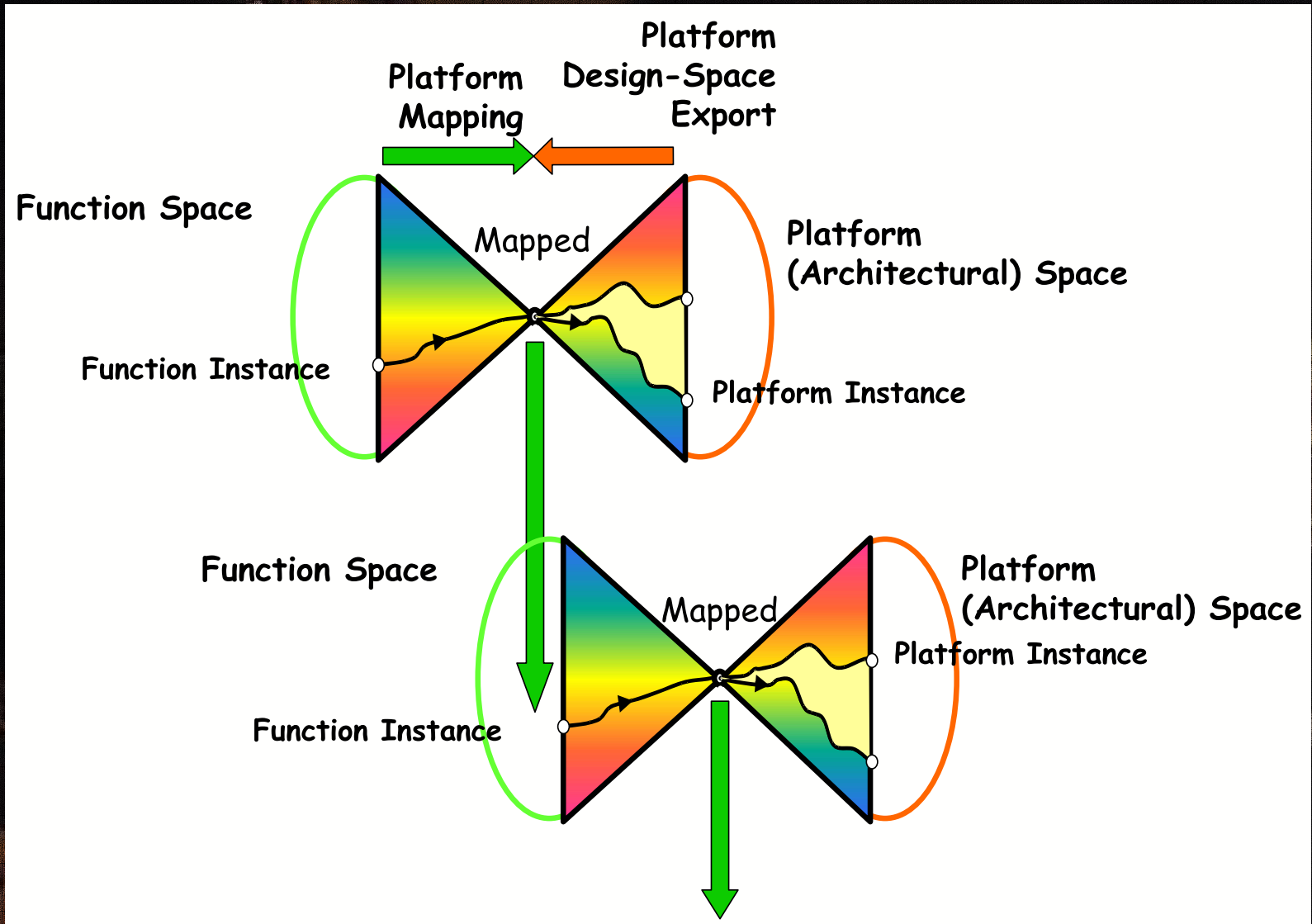


Platform-based Design



- Platform: library of resources defining an abstraction layer
  - Resources do contain virtual components i.e., place holders that will be customized in the implementation phase to meet constraints
  - Very important resources are interconnections and communication protocols

Fractal Nature of Design



Basic Goals

- Identify precisely and formally the concept of *communication, its level of abstraction* and of the corresponding models of computation.
- New theories to combine different models of computation
- Determine a set of properties that characterizes each level of abstraction.

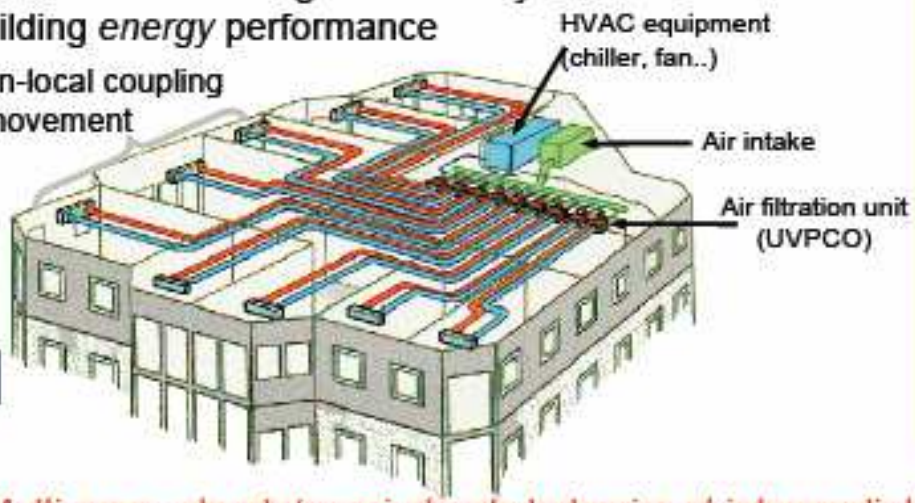
Outline

- Embedded Software Status in Industry
- Platform-based Design
- Applications of PBD to Distributed Building Control
- A synthesis method
- Metropolis
- Automotive and Multi-media

HVAC: High Performance Buildings

Objectives: Efficient energy utilization and occupant comfort for normal building operation
Robust response to health and safety threats and events

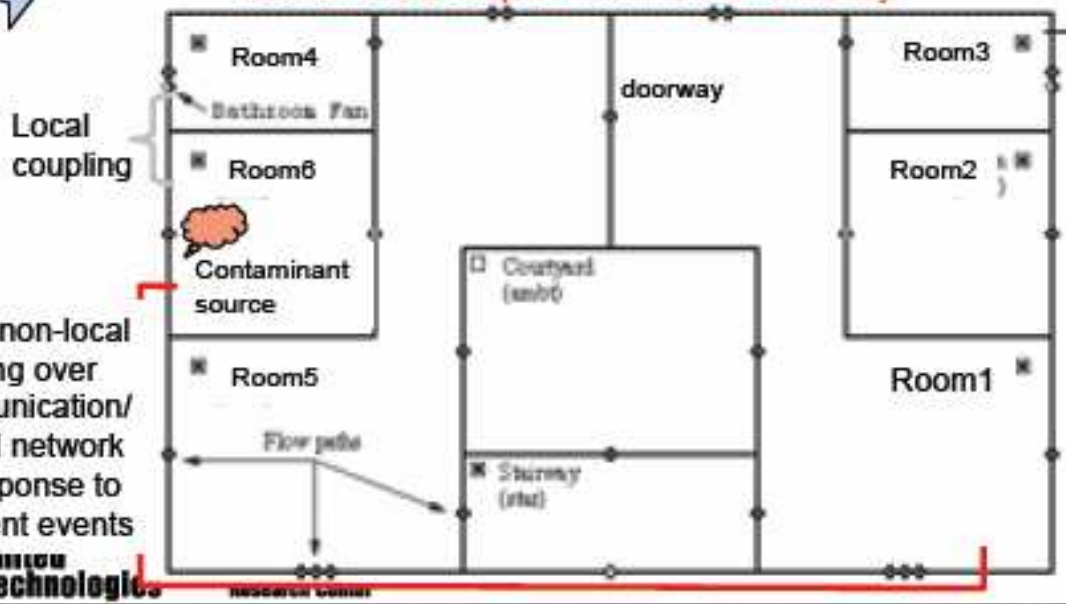
Energy & mass balances govern steady-state building *energy* performance

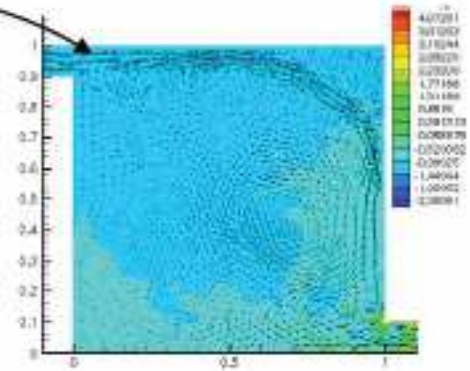


"Slow" non-local coupling from air movement system

- Large, spatially distributed system
- Interconnected system
  - Room neighborhood scale
  - Building floor scale
  - Whole building scale
- Multi-scale dynamic system & its control
  - Wide time scale separation
  - "Close" coupling leading to dynamic constraints between network and physical system

Multi-zone, steady/quasi-steady behavior at intermediate scales relevant to occupant *comfort* and *safety*



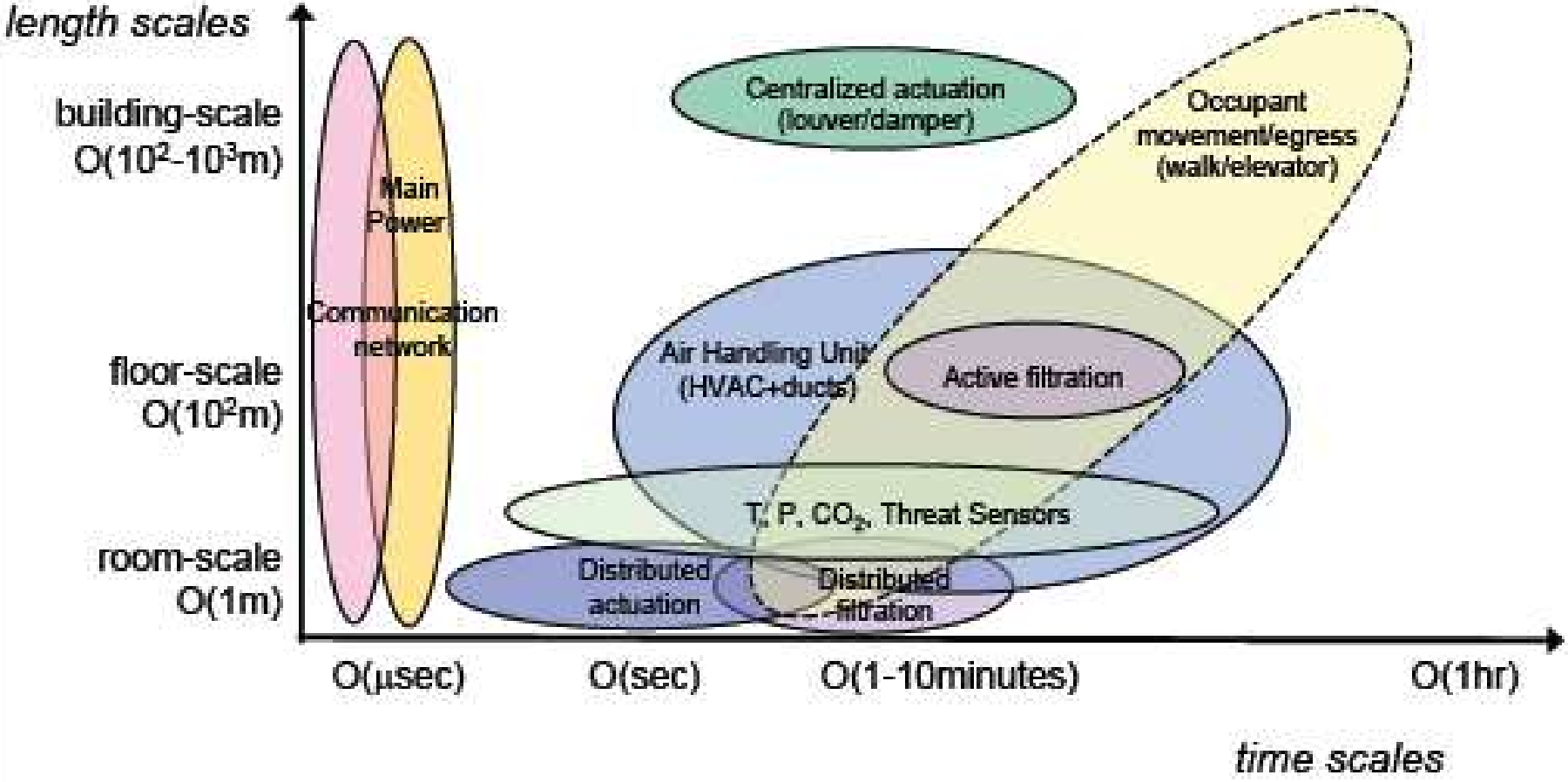


Spatiotemporal airflow dynamics at room scale relevant to *safe building environment*

United Technologies Research Center

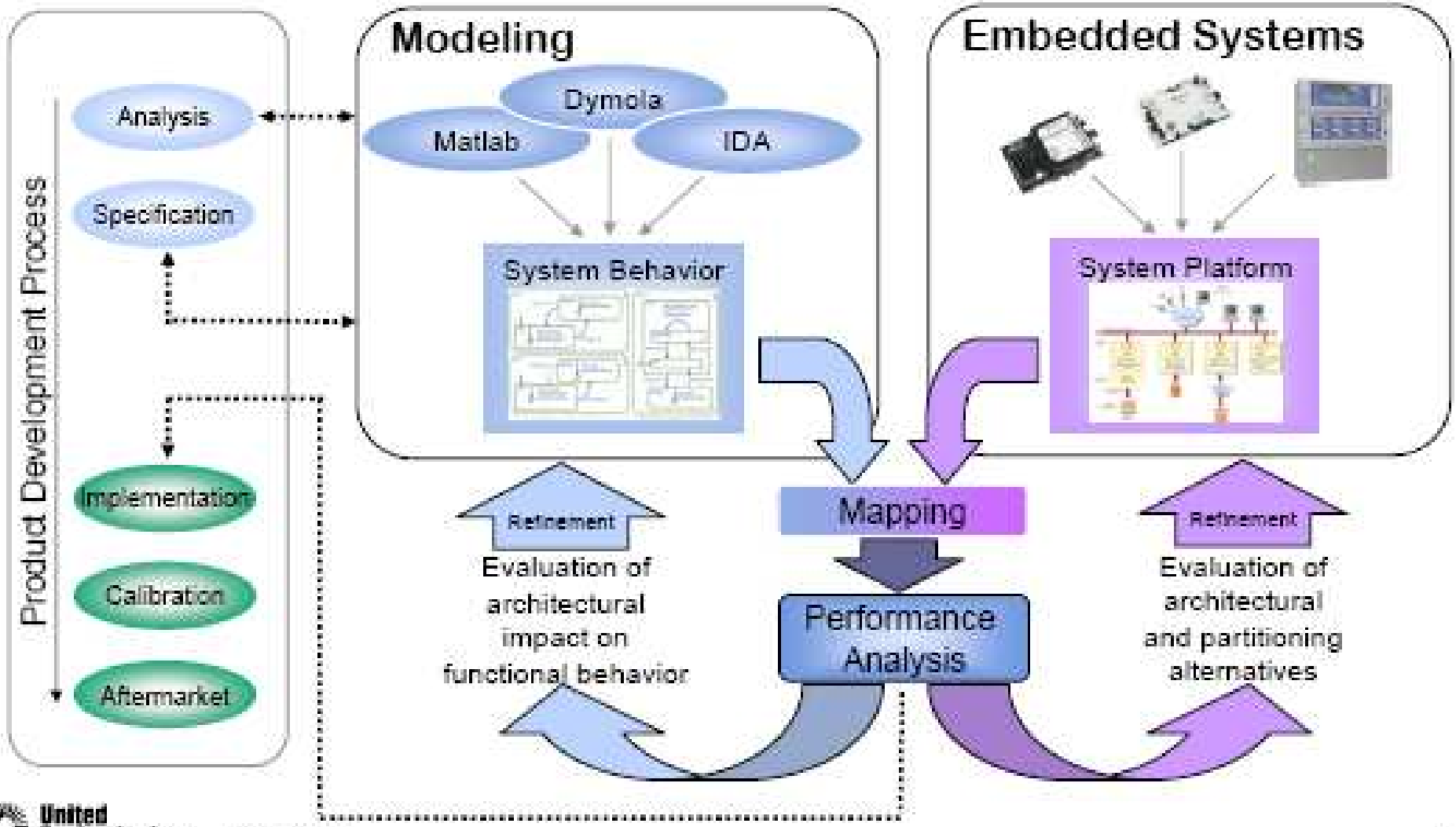
Building System Physics

Dynamics at multiple spatial & temporal scales must be addressed

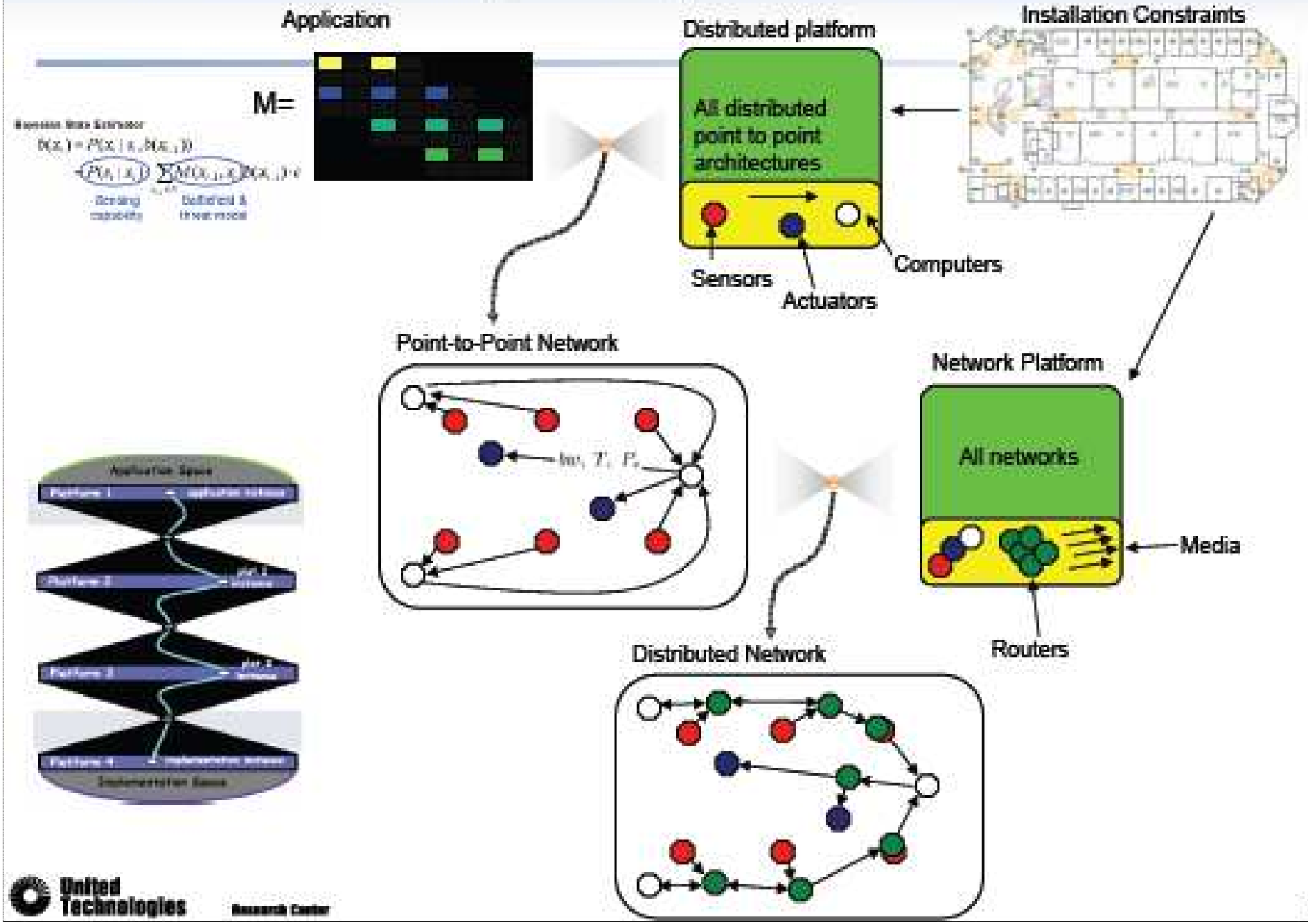


Design Tools: Platform-Based Design

Adopt to networked cyber/physical security systems: fire and HVAC control



Platform-Based Design for Dynamic Networks

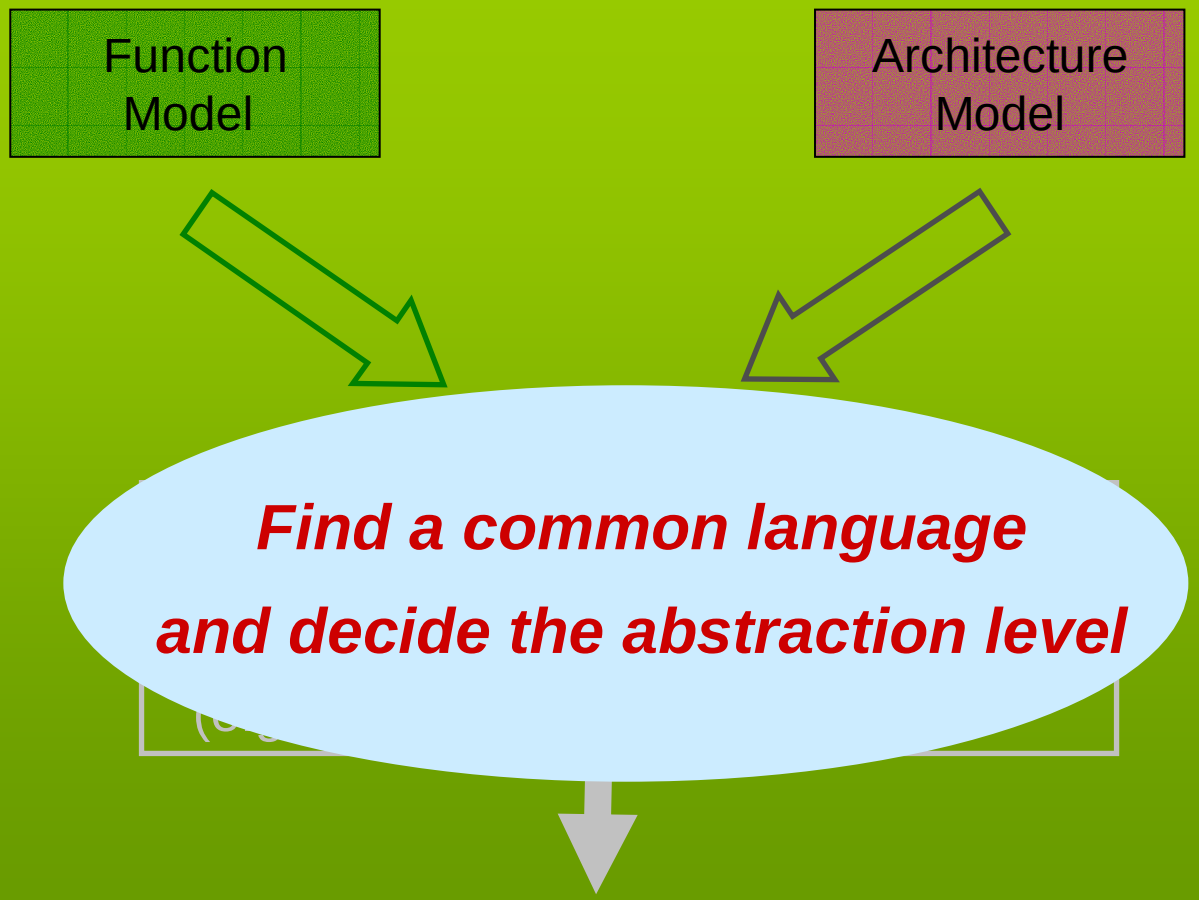


Outline

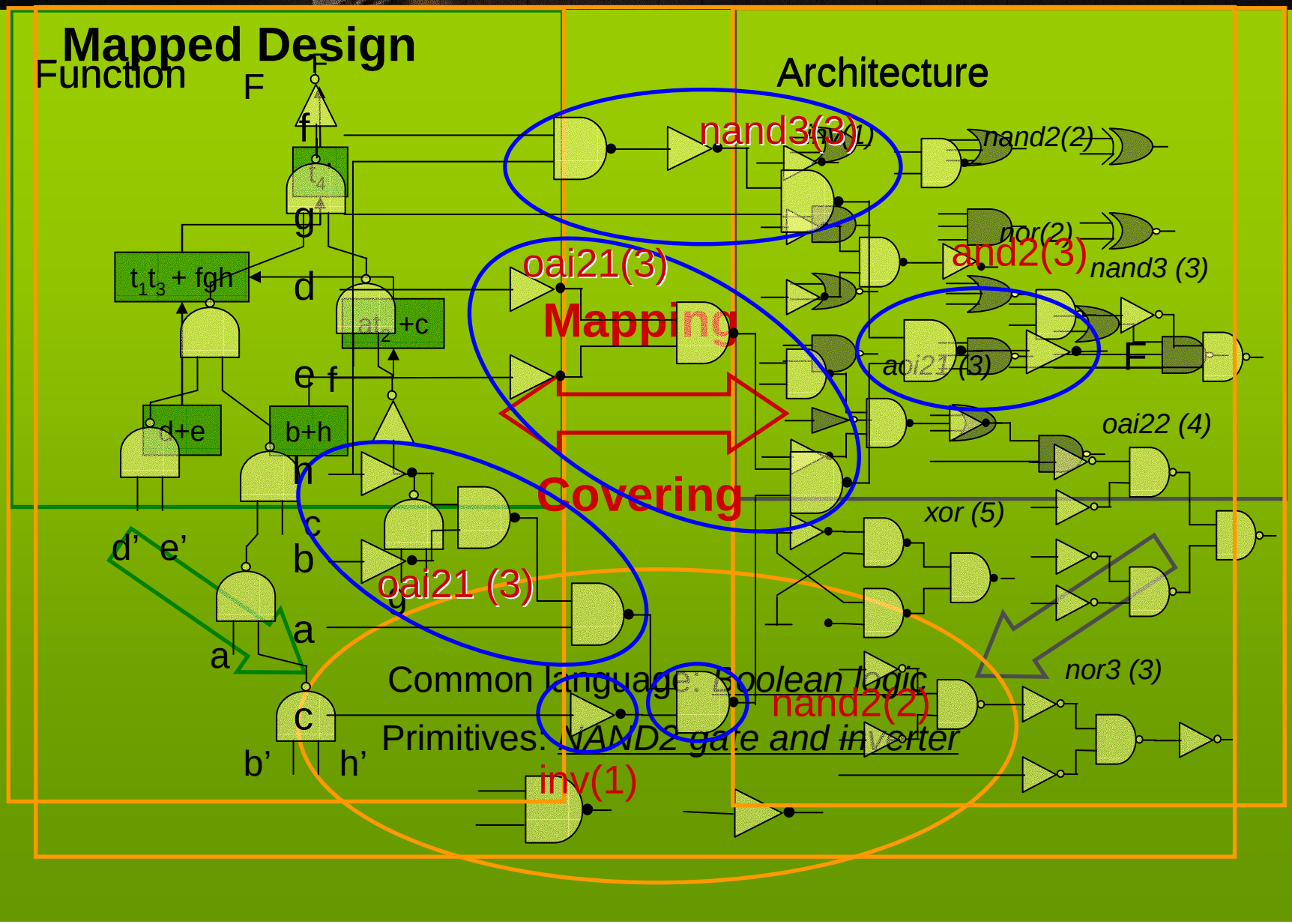
- Embedded Software Status in Industry
- Platform-based Design
- Applications of PBD to Distributed Building Control
- A synthesis method
- Metropolis
- Automotive and Multi-media

Challenge

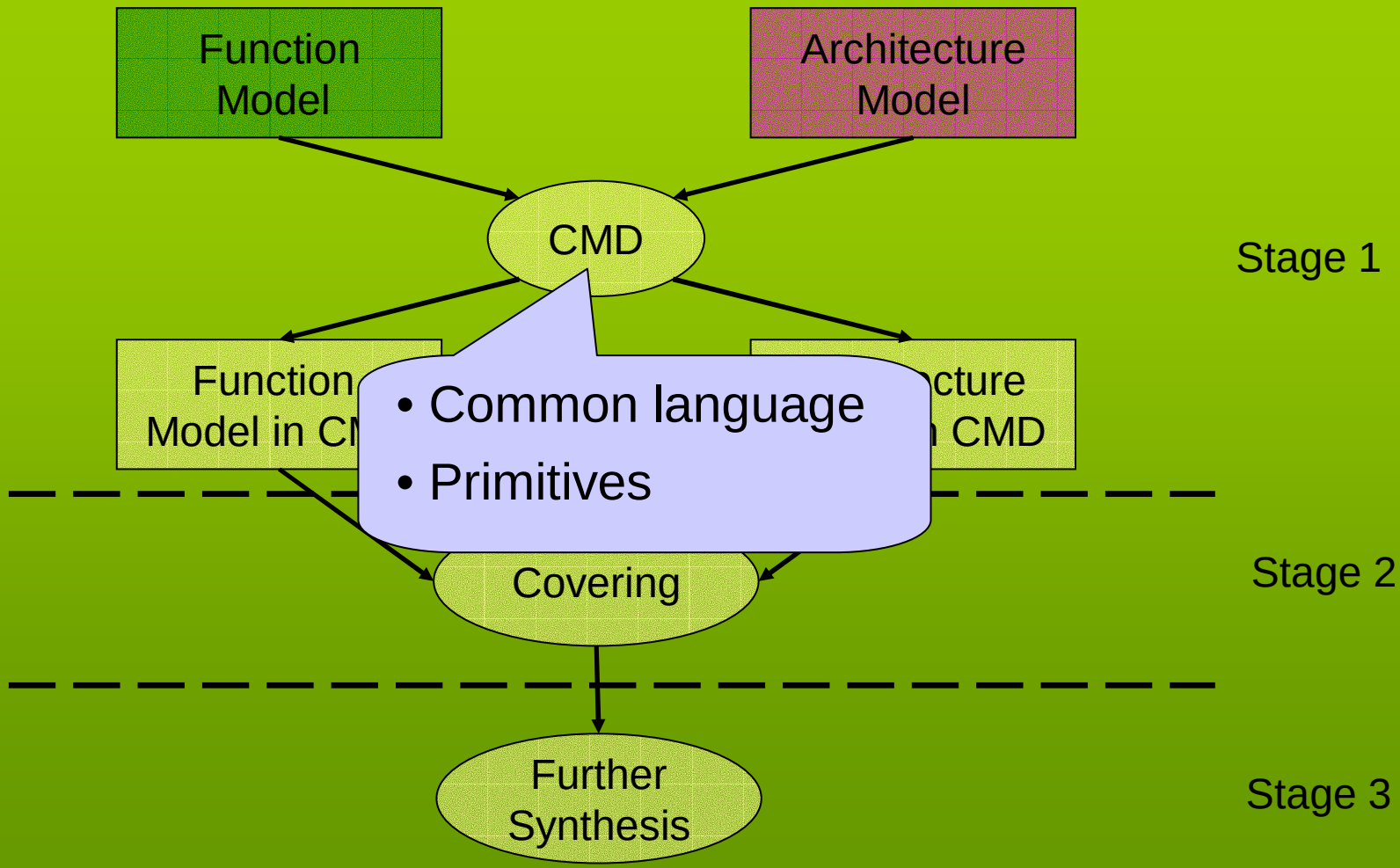
Functionality and Architecture are modeled separately

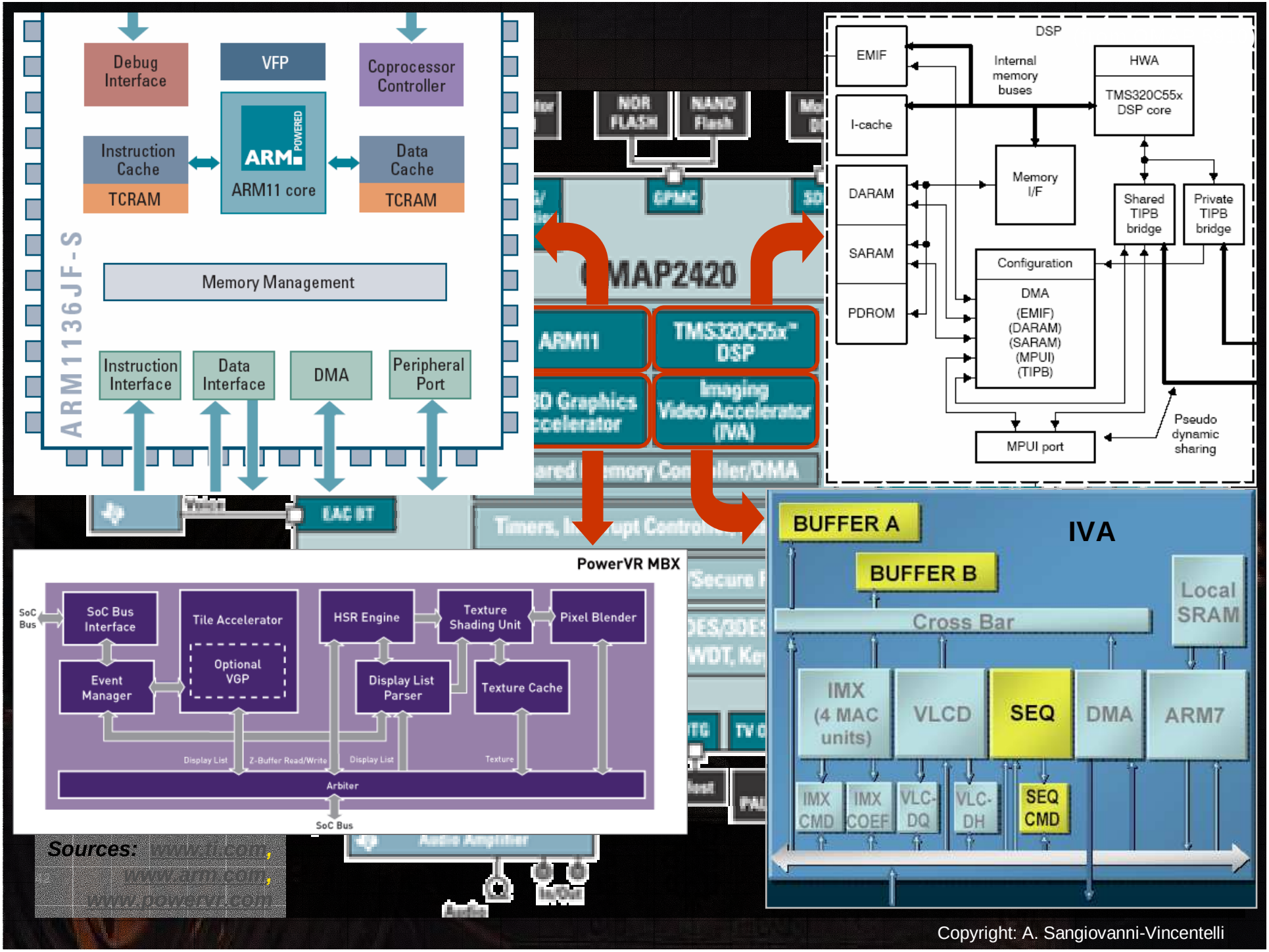


Technology Mapping



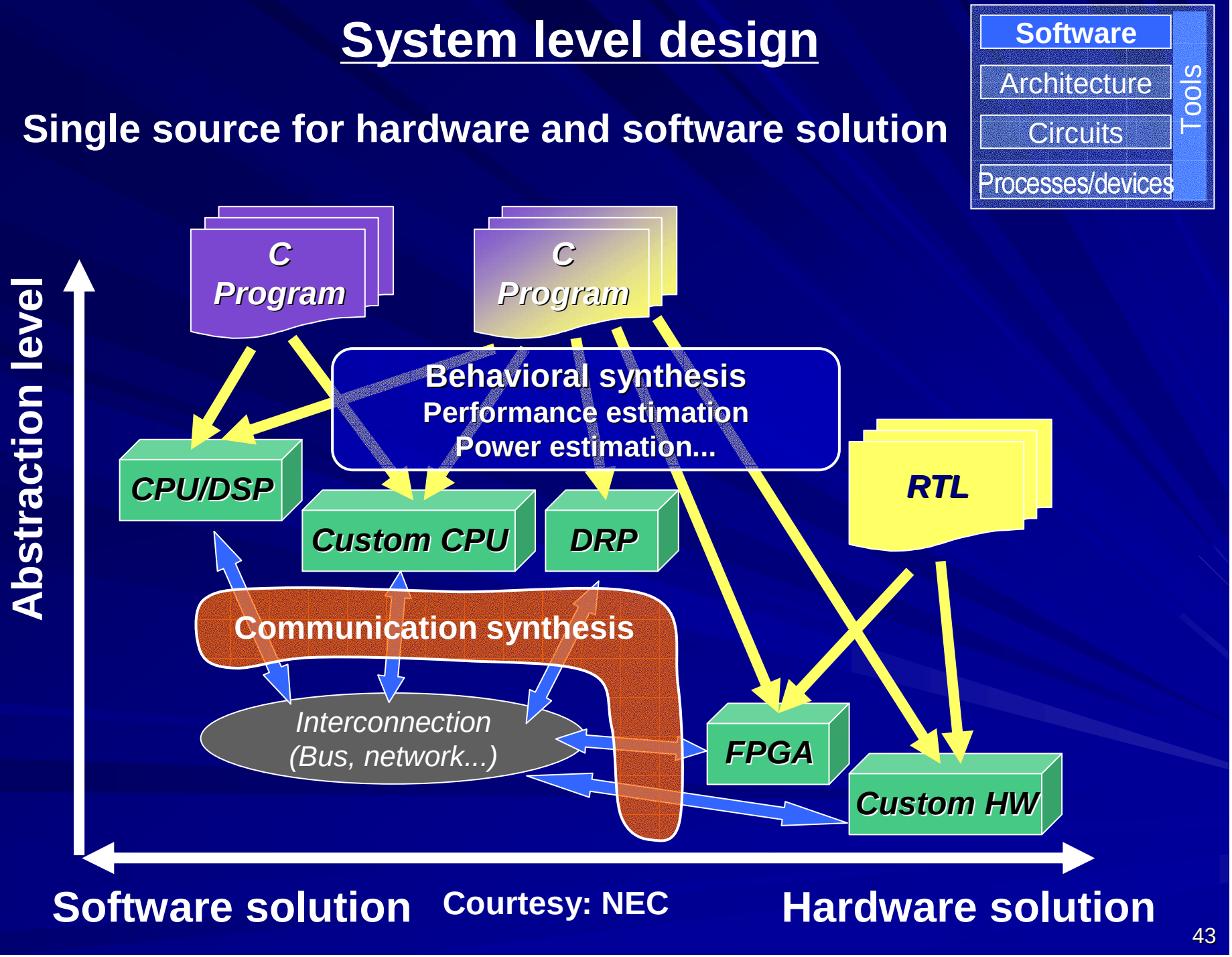
Synthesis Flow



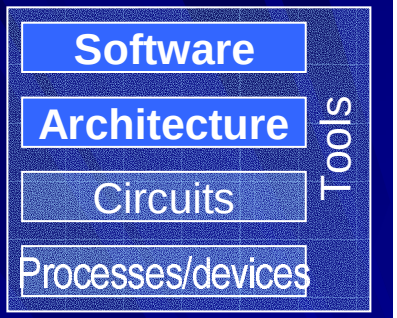


System level design

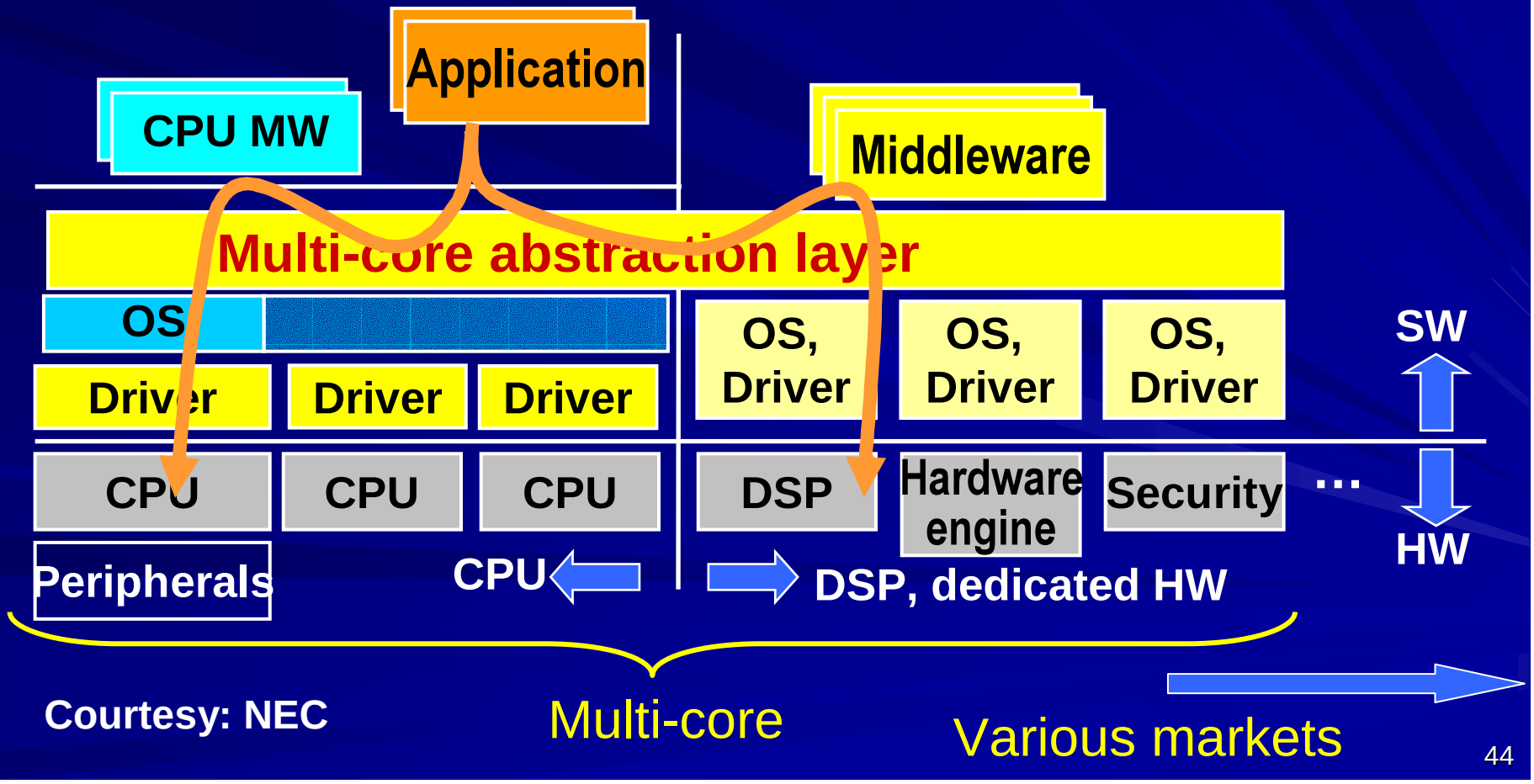
Single source for hardware and software solution



Hetero and scalable architecture

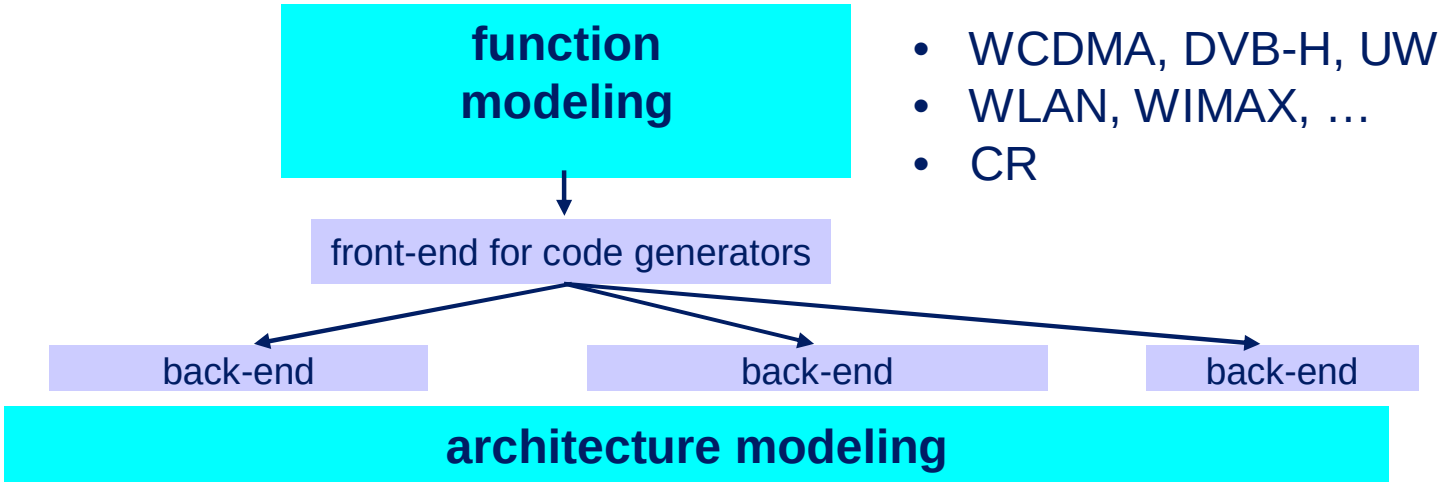


- **Heterogeneous and scalable architecture** for various markets
- **Multi-core abstraction layer** for software virtualization



Courtesy: NEC

Designing Beyond the 3rd Generation



- WCDMA, DVB-H, UWB,
- WLAN, WIMAX, …
- CR

- multiple program-mable processors
- reconfigurable DP+IC architecture
- fpga-inspired architectures

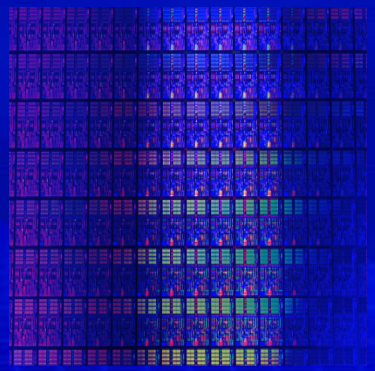
Common Research Agenda:

- benchmarks
- real-time in f&a space
- common rules for modeling
- connection between function and architecture model
- systematic design space exploration
- synergy in designing code generators
- „Future-proof“ framework for Function&Architecture Modeling (Matlab, Simulink, Metropolis …?)

Framework for Platform-Based Design

Typical Application: The Intel MXP5800

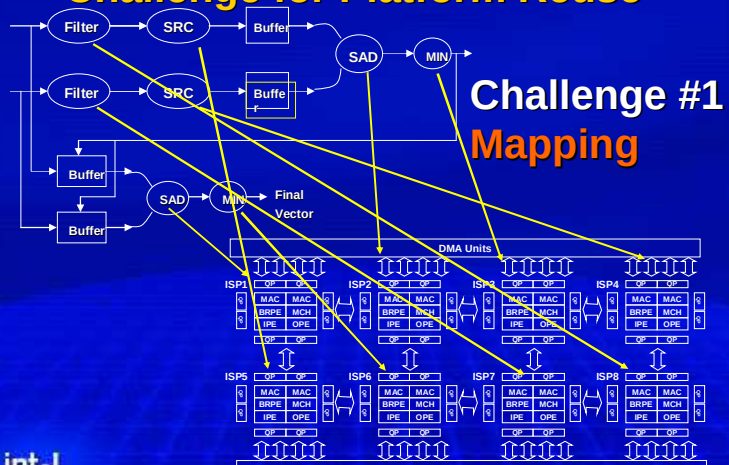
- Complete Solution for high performance Digital Imaging Applications
  - Multifunction Printers
  - High End scanners



intel

18

Challenge for Platform Reuse



Challenge #1
Mapping

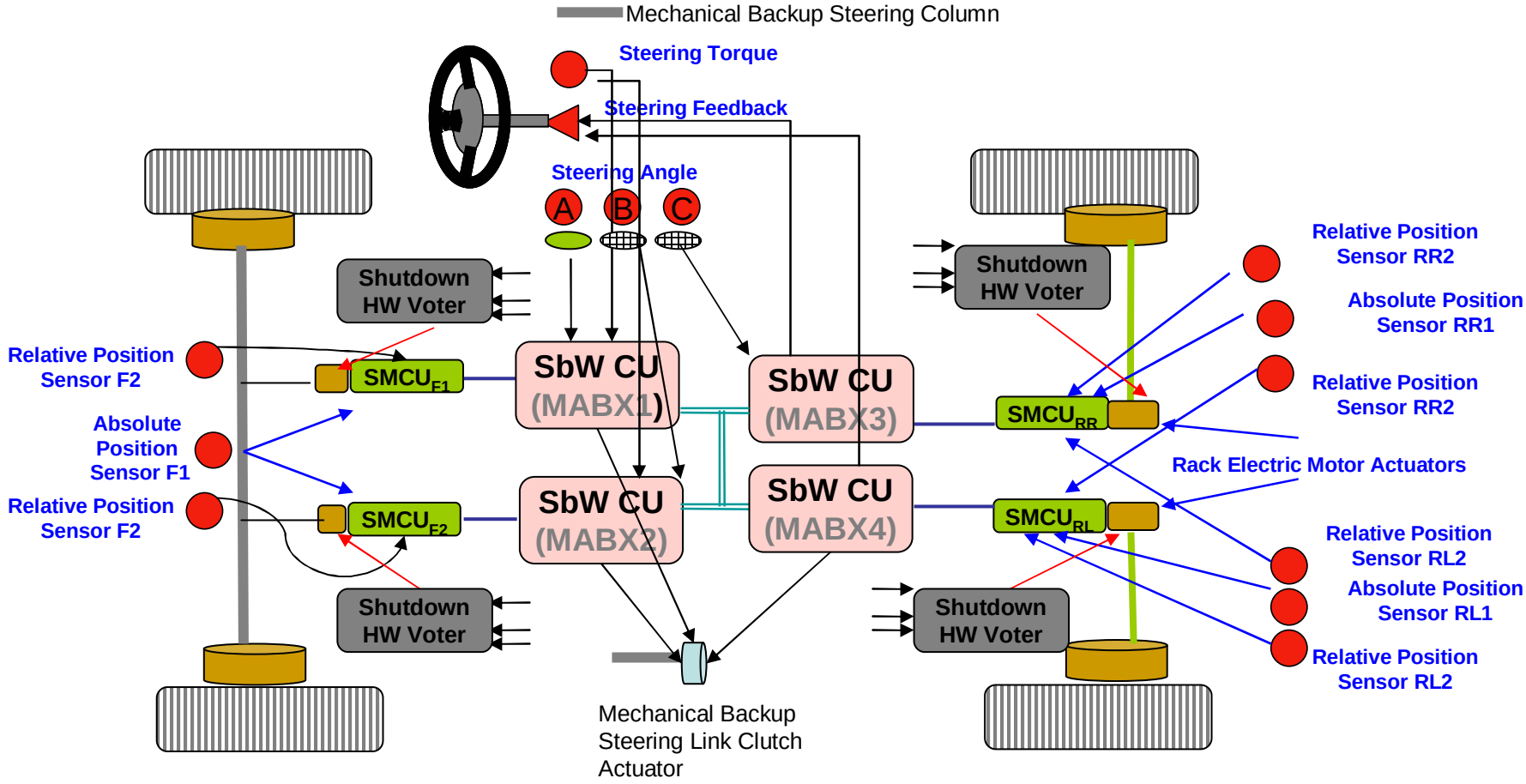
intel

20

Courtesy: G. Spirakis, Intel

GSRC

Architectural Exploration:
FX3 Baseline SBW Hardware Architecture 1



Motor temp sensors in all SMCU.
Motor current, voltage sensors in all SMCU's.
Shutdown HW voter gets votes from all replicated supervisors to shutdown power to the rack electric motor actuators.

Putting it all together….

- We need an integration platform
  - **To deal with heterogeneity:**
    - **Where we can deal with Hardware and Software**
    - **Where we can mix digital and analog**
    - **Where we can assemble internal and external IPs**
    - **Where we can work at different levels of abstraction**
  - **To handle the design chain**
  - **To support integration**
    - **e.g. tool integration**
    - **e.g. IP integration**
- The integration platform must subsume the traditional design flow, rather than displacing it

Outline

- Embedded Software Status in Industry
- Platform-based Design
- Applications of PBD to Distributed Building Control
- A synthesis method
- Metropolis
- Automotive and Multi-media

Metropol esign

- Motivat
  - Sem is necessary
- Platfor
  - F
  - C
  - C
- Metrop
  - Exte and synthesis capa
  - Easi o external tools
- Releas

COVER FEATURE

Metropolis: An Integrated Electronic System Design Environment

Based on a metamodel with formal semantics that developers can use to capture designs, Metropolis provides an environment for complex electronic-system design that supports simulation, formal analysis, and synthesis.

Felice Balarin
Yosinori Watanabe
Cadence Berkeley Labs

Harry Hsieh
University of California, Riverside

Luciano Lavagno
Claudio Passerone
Politecnico di Torino

Alberto Sangiovanni-Vincentelli
University of California, Berkeley

A solid design flow must capture designs at well-defined levels of abstraction and proceed toward an efficient implementation. The critical decisions involve the system's architecture, which will execute the computation and communication tasks associated with the design's overall specification. Understanding the application domain is essential to ensure efficient use of the design flow.

Today, the design chain lacks adequate support. Most system-level designers use a collection of unlinked tools. The implementation then proceeds with informal techniques that involve numerous human-language interactions that create unnecessary and unwanted iterations among groups of designers in different companies or different divisions. These groups share little understanding of their respective knowledge domains. Developers thus cannot be sure that these tools, linked by manual or empirical translation of intermediate formats, will preserve the design's semantics. This uncertainty often results in errors that are difficult to identify and debug.

The move toward programmable platforms shifts the design implementation task toward embedded software design. When embedded software reaches the complexity typical of today's designs, the risk that the software will not function correctly increases dramatically. This risk stems mainly from poor design methodologies and fragile software system architectures, the result of growing functionality over an existing implementation that may be quite old and undocumented. The Metropolis project seeks to develop a unified framework that can cope with these challenges.

DESIGN OVERVIEW

We designed Metropolis to provide an infrastructure based on a model with precise semantics that remain general enough to support existing computation models[1] and accommodate new ones. This *metamodel* can support not only functionality capture and analysis, but also architecture description and the mapping of functionality to architectural elements.

Metropolis uses a logic language to capture nonfunctional and declarative constraints. Because the model has a precise semantics, it can support several synthesis and formal analysis tools in addition to simulation.

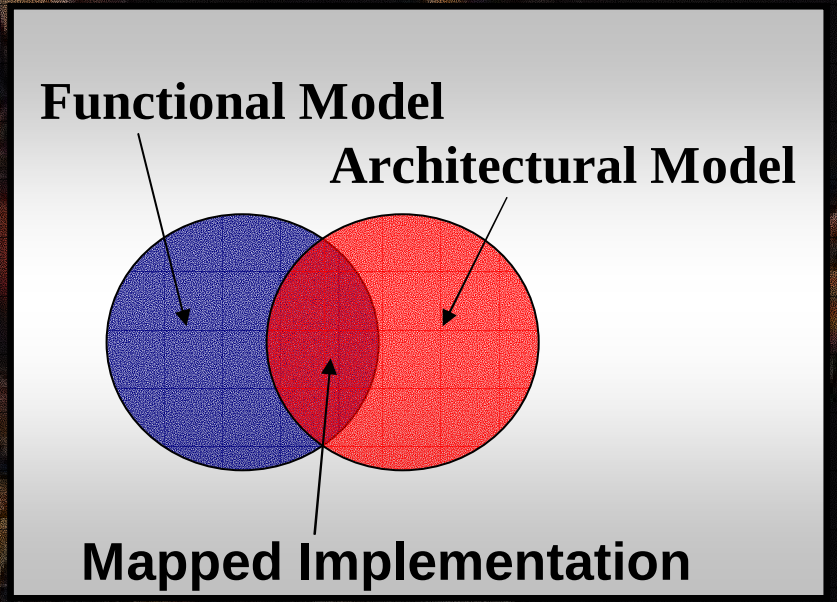
The first design activity that Metropolis supports, communication of design intent and results, focuses on the interactions among people working at different abstraction levels and among people working concurrently at the same abstraction level. The metamodel includes constraints that represent in abstract form requirements not yet implemented or assumed to be satisfied by the rest of the system and its environment.

Fundamental Concepts

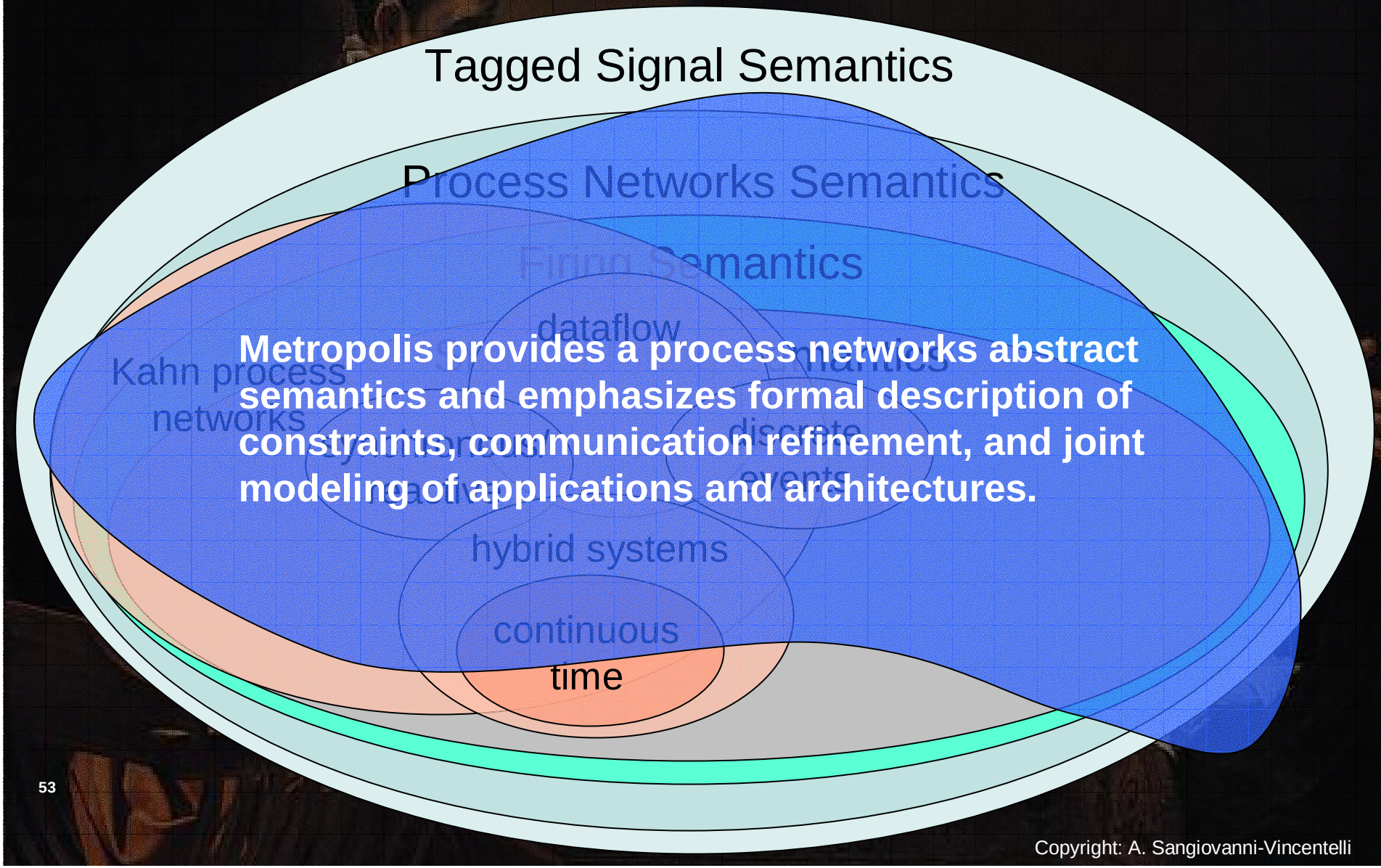
- Support for different Models of Computation
- Support for Architecture Specification and Analysis
- Mix of imperative and declarative specification styles
- Quantities of interest dictated by the designer, not the framework
- Framework designed to allow interfacing with external tools

Mapping methodology in Metropolis

- An architecture provides services
  - Cost of services
  - Concurrency
- A functional model uses these services
  - Ordering
  - Data values
- Mapping binds the two together formally
- Restrict non-determinism

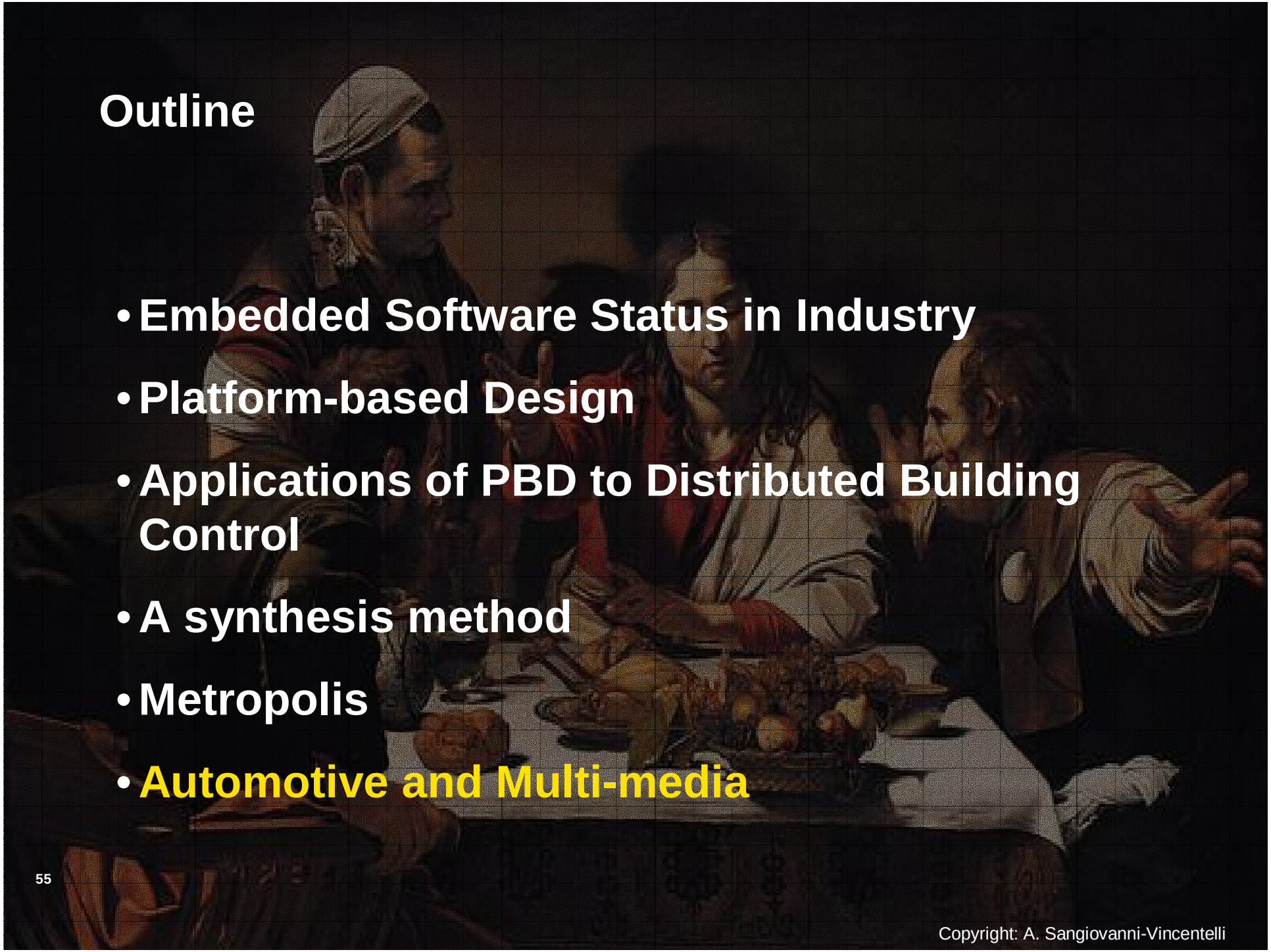


Meta Frameworks: Metropolis



Industrial Collaborations

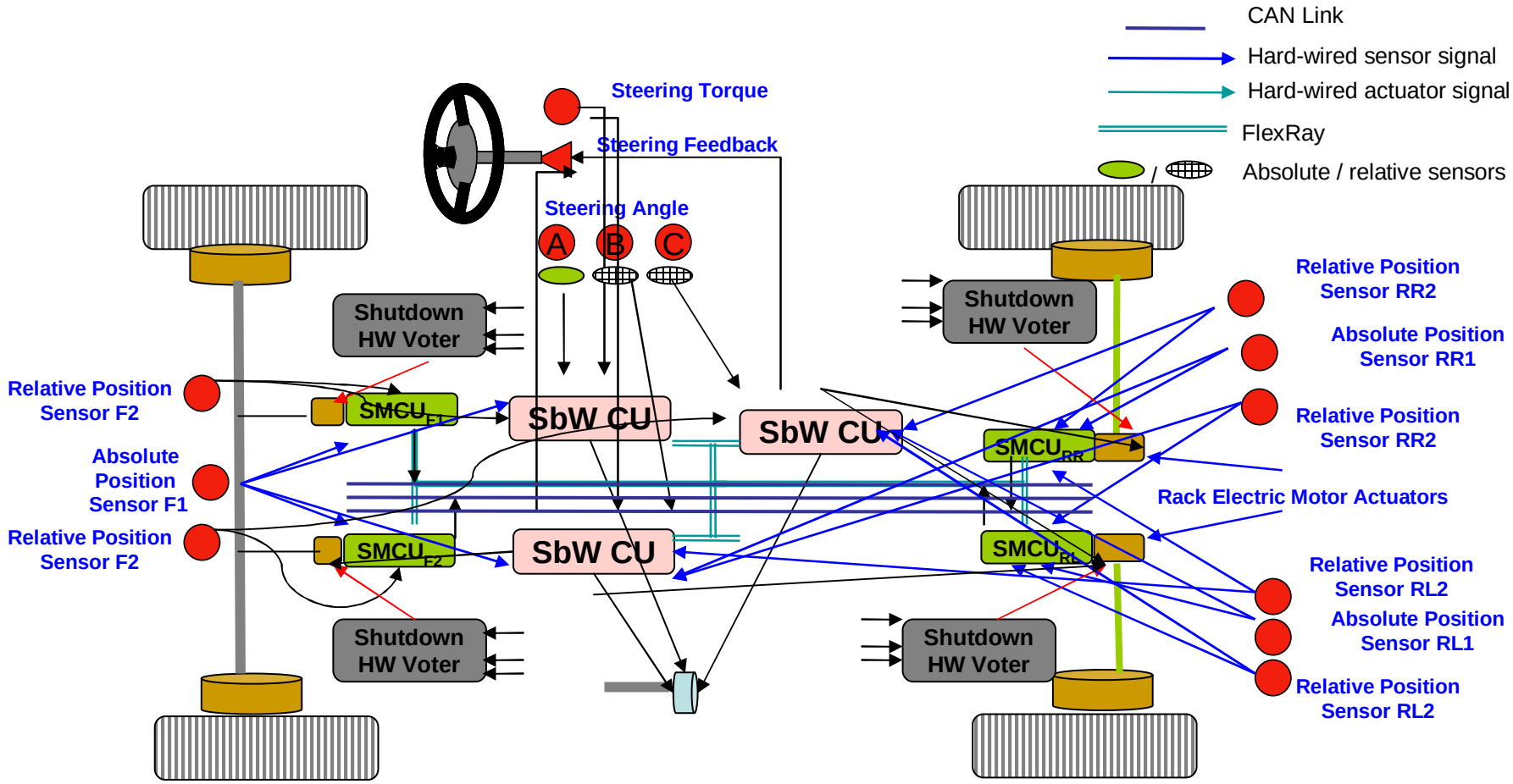
- Infineon: Software Defined Radios
- Intel:
  - Mobile Platforms
  - Multi-media Platforms
- General Motors
  - Next generation car architectures
- United Technologies
  - Elevator (OTIS), Air conditioning (CARRIER), Security (Chubb)
- Xilinx
  - Programmable platform modeling



Outline

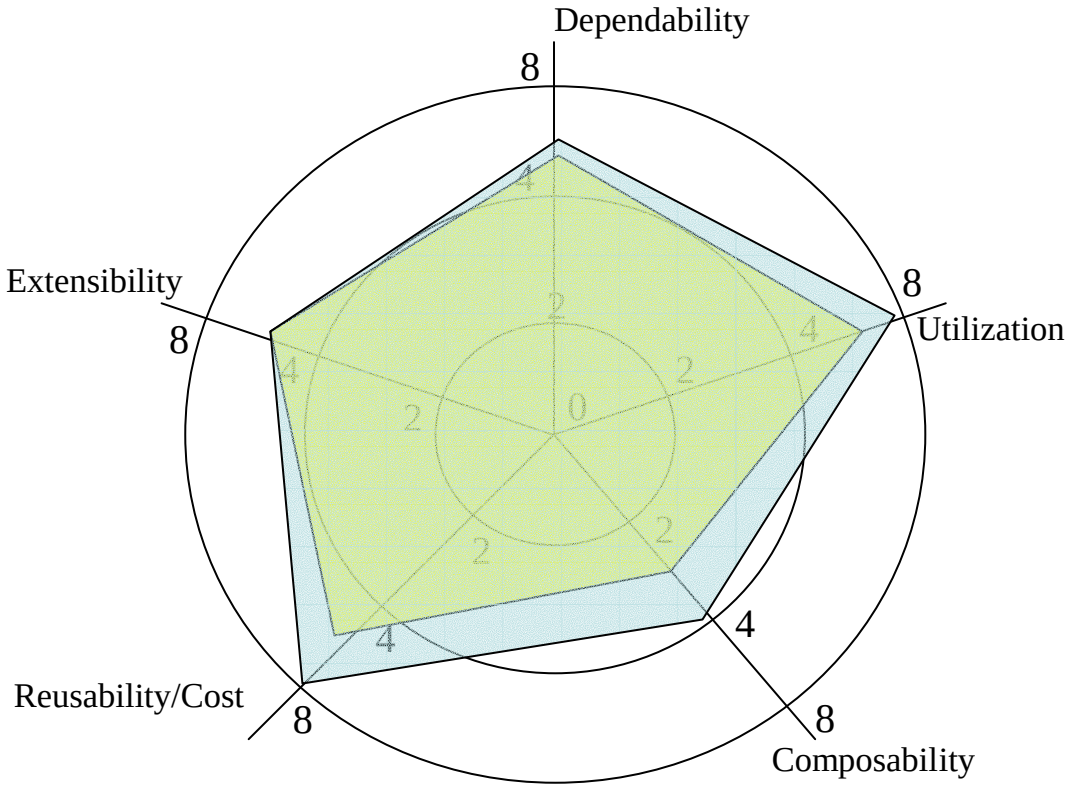
- Embedded Software Status in Industry
- Platform-based Design
- Applications of PBD to Distributed Building Control
- A synthesis method
- Metropolis
- Automotive and Multi-media

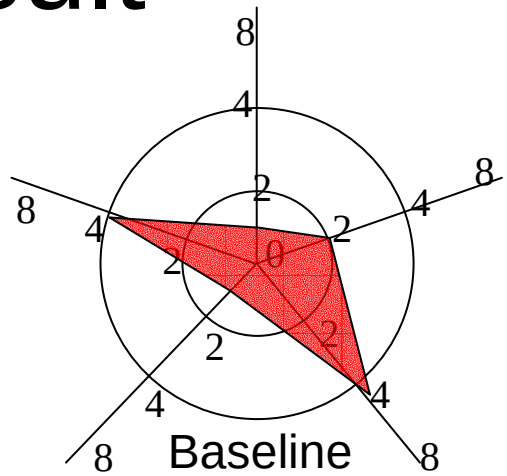
SBW Hardware Architecture 5



Architecture 5 has no SMCU's and
the IO's are redistributed to the other components with
redundant actuators

Architecture 5: Result





Baseline Architecture

Architecture 5

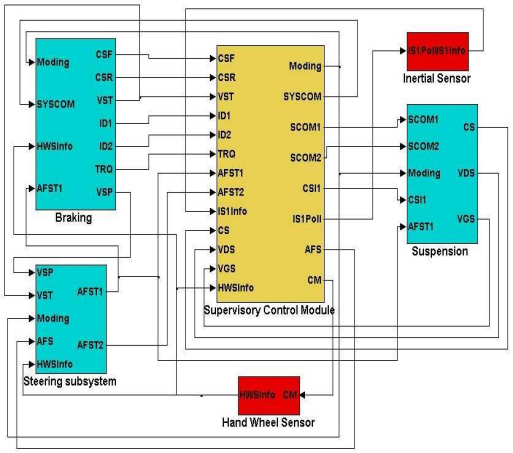
Architecture 4

Automotive Stability Control

Find common language between functionality and architecture by choosing semantic domain Q in CMD $C_Q(P)$.

Functionality

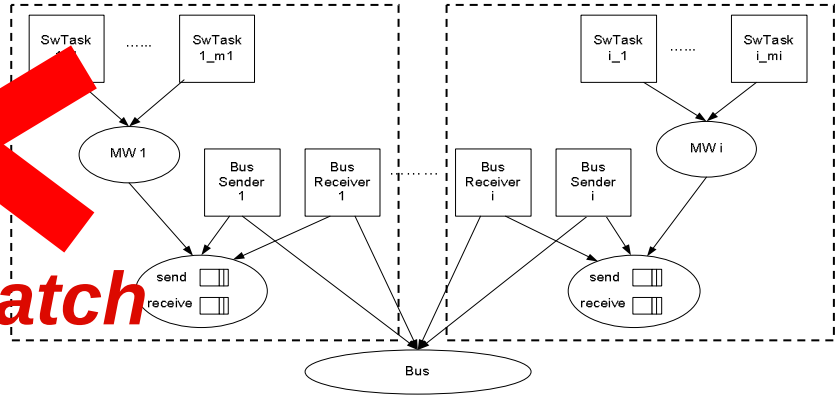
- synchronous Simulink model
- no message loss or duplication



mismatch

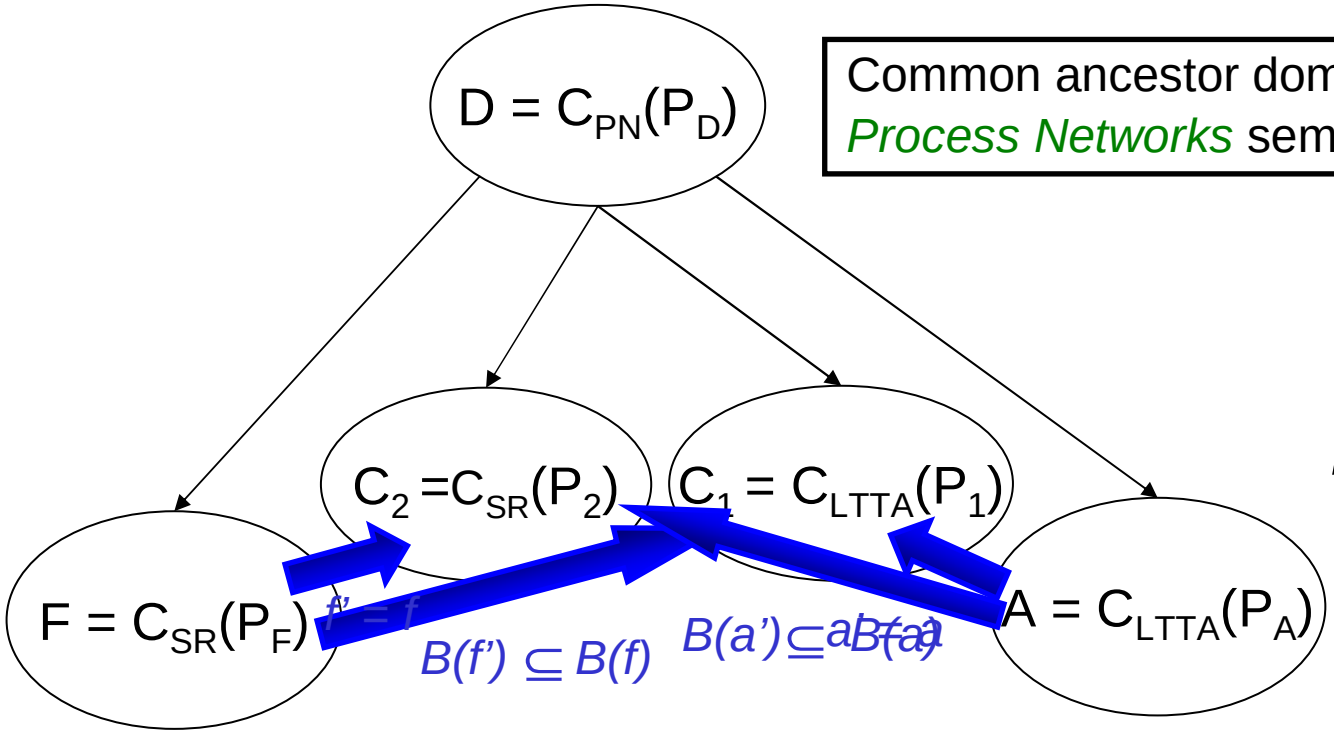
Architecture

- clock drift between distributed ECUs, asynchronous comm.
- data loss and duplication



CMD Selection (Stage 1)

- Find CMD from modeling domain relation graph



Common ancestor domain:
Process Networks semantics

$P_1 = P_{F'} \cup P_A$

$P_2 = P_F \cup P_{A'}$

Original Function modeling domain:
Synchronous/Reactive semantics

Original Architecture modeling domain:
Semantics of *LTTA*
(loosely time-triggered architecture)

[1] H. Zeng, A. Davare, A. L. Sangiovanni-Vincentelli, et. al, "Design Space Exploration of Automotive Platforms in Metropolis," SAE 2006

Transformation of architecture model

- CMD uses synchronous/reactive semantics
  - Function model does not need to be changed
  - Architecture primitives P_A should be transformed to $P_{A'}$ in CMD to support synchronous/reactive semantics
    - Protocol to avoid data loss [1]
      - Constraints on process periods
      - Requires clock drift to be within a certain range
    - Clock synchronization to restrict the clock drift [2]
    - Alternating bit to avoid data duplication
    - *B(a')* $\subseteq$ *B(a)*, but correct behaviors can be assured when function model *f'* is mapped to architecture model *a'*

[1] A. Benveniste, P. Caspi, P. Le Guernic, H. Marchand, J.P. Talpin and Stavros Tripakis, "A Protocol for Loosely Time-Triggered Architectures", EMSOFT '02

[2] M. Gergeleit and H. Streich, "Implementing a Distributed High-Resolution Real-Time Clock using the CAN-Bus", 1st International CAN-Conference

Automatic Synthesis

- Covering problem
  - Function primitives instance f_i : tasks and messages
  - Architecture primitives instance a_j : ECUs and buses
  - Quantity constraints : maximum workload on ECU
  - Objective function: minimize inter-ECU communication and balance computation load across ECUs
  - Utilize the Scotch [1] package
    - Function model graph and architecture graph
    - Partition-based algorithm solves the covering problem
- Further synthesis
  - Task priorities – based on pre-assigned message priorities
  - Task periods
    - Adjusted to satisfy end-to-end latency requirements
    - Task periods need to satisfy the protocol to ensure the correctness of semantics

[1] Scotch, http://www.labri.fr/perso/pelegrin/scotch

Experimental Results

- 14 tasks, 48 messages
- 6 ECUs, 1 bus
- Automatic synthesized design vs. 6 manual designs
- Simulated in Metropolis framework

Design	Total delay (ms)	Bus utilization	Max ECU utilization
Manual 1	787.10	57%	60%
Manual 2	761.84	50%	72%
Automatic	559.51	34%	48%

Summary and Perspectives

- Electronic Industry facing an array of complex problems from design to manufacturing involving complexity, power, reliability, reconfigurability, embedded software
- Design Methods and Tools lacking: active research field
- Investment needed from IC and System industry otherwise the situation is bound to become more critical. Not an issue of languages or point tools
- EDA vendors have to extend their reach into the system space
- Innovation of this magnitude is difficult to achieve within the present EDA industrial infrastructure
  - Incubation?
  - Well-endowed Start-ups?
  - Consortia?
- Future ……

Educational Challenge