METHODS AND TOOLS FOR THE DESIGN OF ELECTRONIC SYSTEMS

Alberto Sangiovanni-Vincentelli

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

Co-Founder and Member of the Board
Cadence Design Systems

Outline

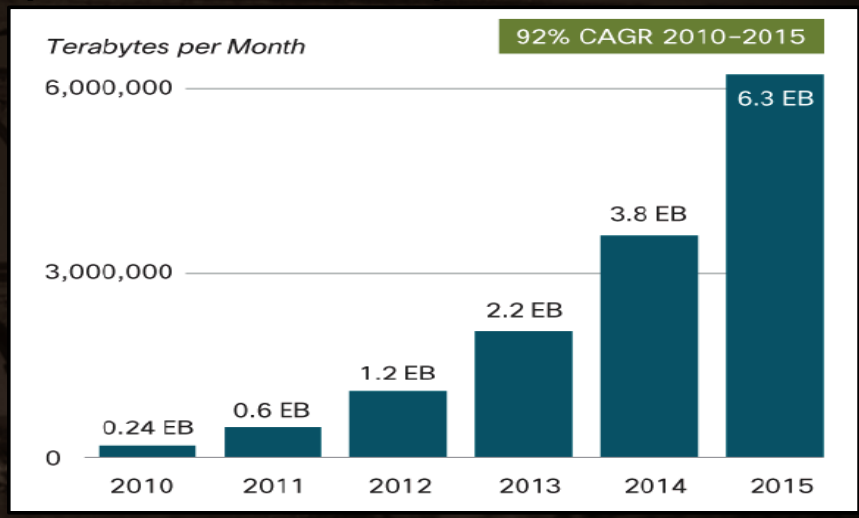
- Introduction and Motivation using Automotive as Test Case
- The V design process and Platform Based Design
- The Role of Autosar
- Semiconductor Design Economics
- Extensions and Open Issues

THANKS TO BMW, Cadence, GM, Intel, Magneti Marelli, ST and UTC

The IT Platform of Today: Mobiles at the Edge of the Cloud

Mobile data growth

[Source: Cisco VNI Mobile, 2011]

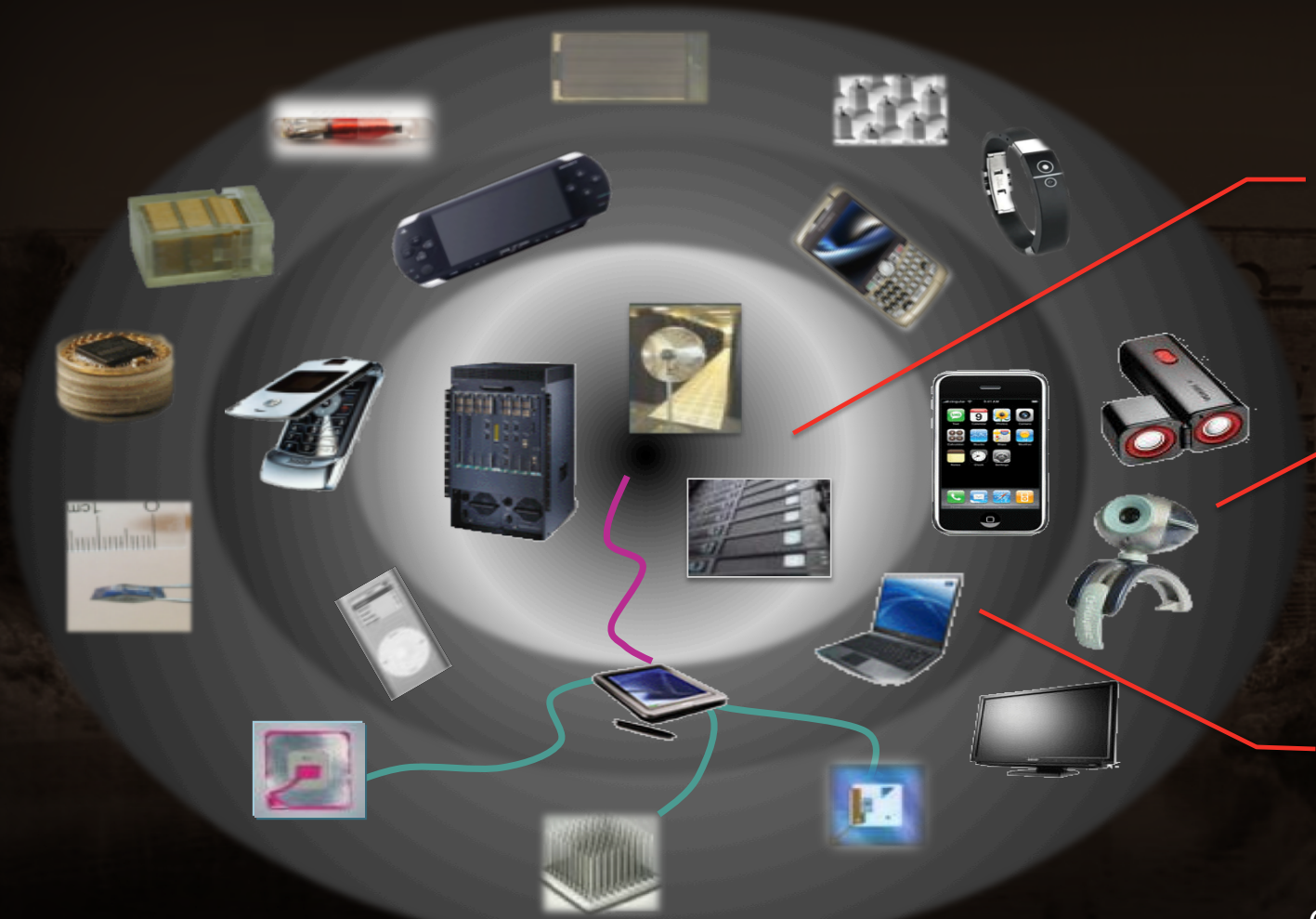


Mobile traffic grew 2.6x in 2010 (nearly tripling for 3rd year)

Driven by Tablets

[J. Rabaey, ASPDAC'08]

The Emerging IT Scene: The Swarm at the Edge of the Cloud



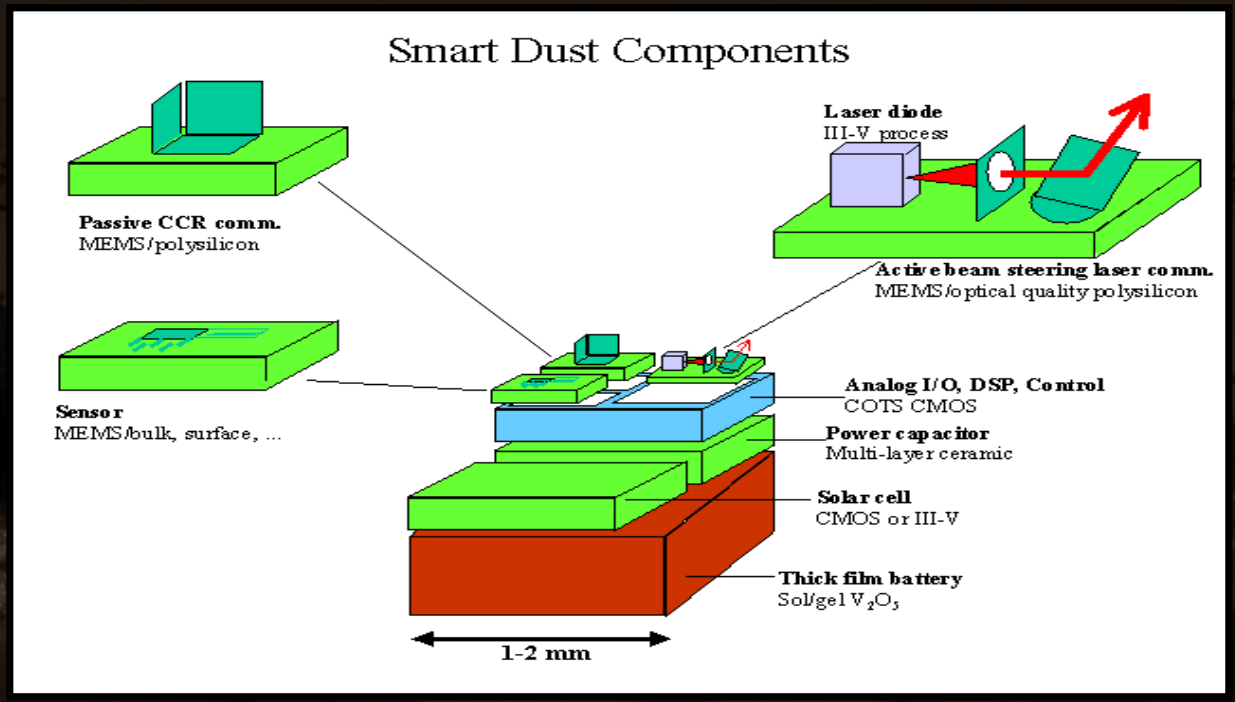
Infrastructural Core: the Cloud

Sensory swarm

Mobile access

Courtesy: J. Rabaey

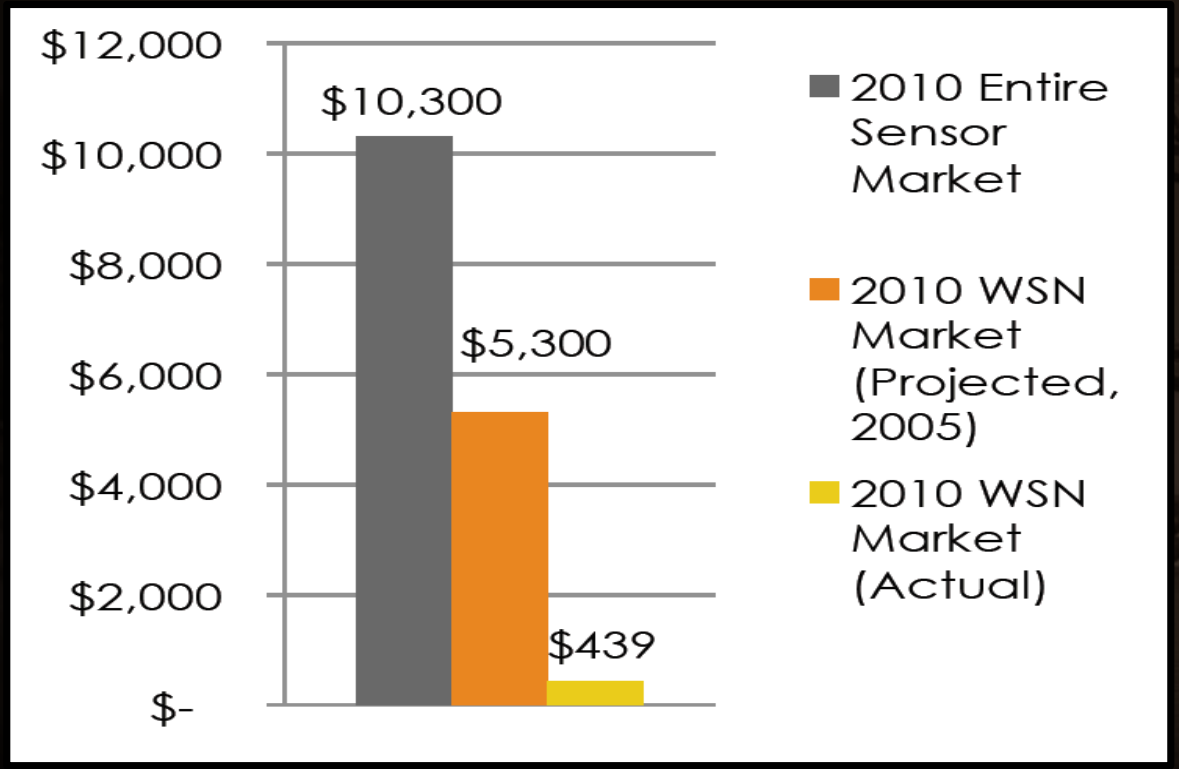
1995 Question: What happens if sensors become tiny, wireless, and self-contained?



... Wireless Sensor Networks

[Courtesy: K. Pister, UC Berkeley]

2010 Outcome: The Unfullfilled Promise of Wireless Sensor Nets



Source: On World

What slowed them down?

(Source: On World)

- Cost savings not yet disruptive
- Reliability
- Energy (battery life)
- Ease of use

Wireless Sensor Nets

What REALLY slows them down: NO Economy of Scale

Stovepipes, Fragmentation, Non-interoperability, Lack of Virtualization

Industrial automation, smart buildings, renewable energy, data centers, …

802.11x (WiFi), 802.15.4x (Zigbee), 802.15.1 (Bluetooth (LE)), 802.15.6 (WPANs), NFC, …

TinyOS, eCOS, LiteOS, Contiki, Arch Rock

Predictions

CMP
United Business Media

EE Times

Wireless is everywhere; ignore it at your peril

Bolaji Ojo
(01/07/2008 9:00 AM EST)
URL: http://www.eetimes.com/showArticle.jhtml?articleID=205208620

The search is over for the next killer app. It is wireless, it is all around you, and it will leave no sector of the global economy untouched.

- 5 Billion people to be connected by 2015 (Source: NSN)
- The emergence of Web2.0
  - The “always connected” community network
- 7 trillion wireless devices serving 7 billion people in 2017 (Source: WirelessWorldResearchForum (WWRF)
  - 1000 wireless devices per person? (Courtesy: Niko Kiukkonen, Nokia)

Vision 2025

- Integrated components will be approaching molecular limits and/ or may cover complete walls
- Every object will have a wireless connection, hence leading to trillions of connected devices,
- Collaborating to present unifying experiences or to fulfill common goals

What will it Enable?

The Birth of the Swarm

CyberPhysical Systems
Linking the Cyber and Physical Words






[H. Gill, NSF 2008]

Aka: The Internet of Things, Societal IT Systems, …

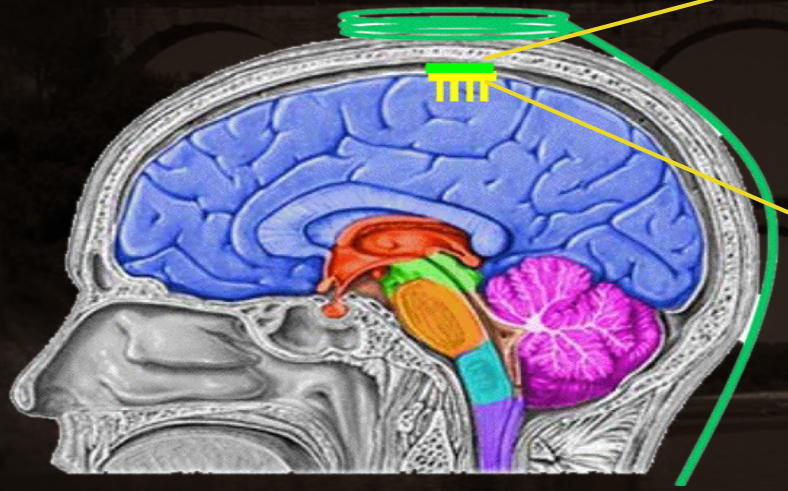
CyberBiological Systems (BioCyber)

Linking the Cyber and Biological Worlds

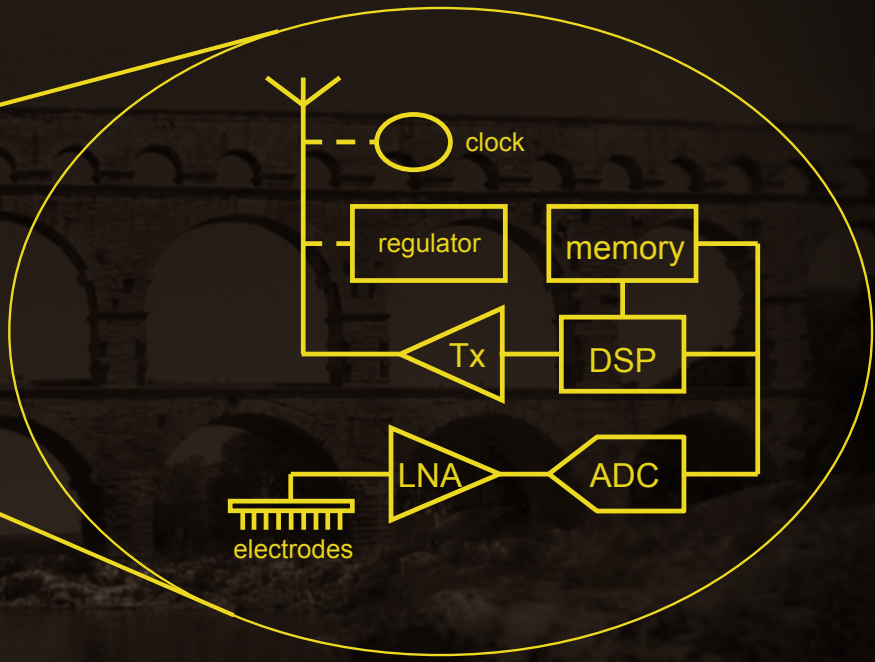
Examples: Telesurgery, Body-area networks, health diagnostics, drug delivery, brain-machine interfaces, …

Towards Integrated Wireless Implanted Interfaces

Moving the state-of-the-art in wireless sensing



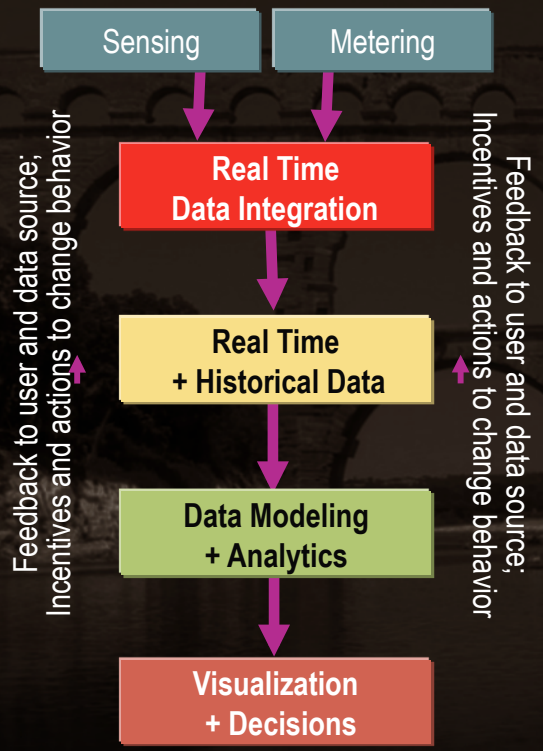
[Illustration art: Subbu Venkatraman]



Power budget: mWs to 1 mW

What does it mean to become Smarter?

Measuring, Monitoring, Modeling and Managing



- **Data collection**
- **Data Integration**
- **Comparison of historical data, with newly collected data**
- **Data modeling and analytics to create insights from data to feed decision support and actions**

Source: IBM Corporate Strategy

Outline

- Introduction and Motivation using Automotive as Test Case
- The V design process and Platform Based Design
- The Role of Autosar
- Semiconductor Design Economics
- Extensions and Open Issues

THANKS TO BMW, Cadence, GM, Intel, Magneti Marelli, ST and UTC

Design Chain Integration

Automotive Industry

Automakers

GM, Ford, TOYOTA, Volkswagen, HONDA, DaimlerChrysler

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

Tier 1 Suppliers

Visteon, DENSO, Johnson Controls, BOSCH, DELPHI

90%+ of revenue from automotive

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

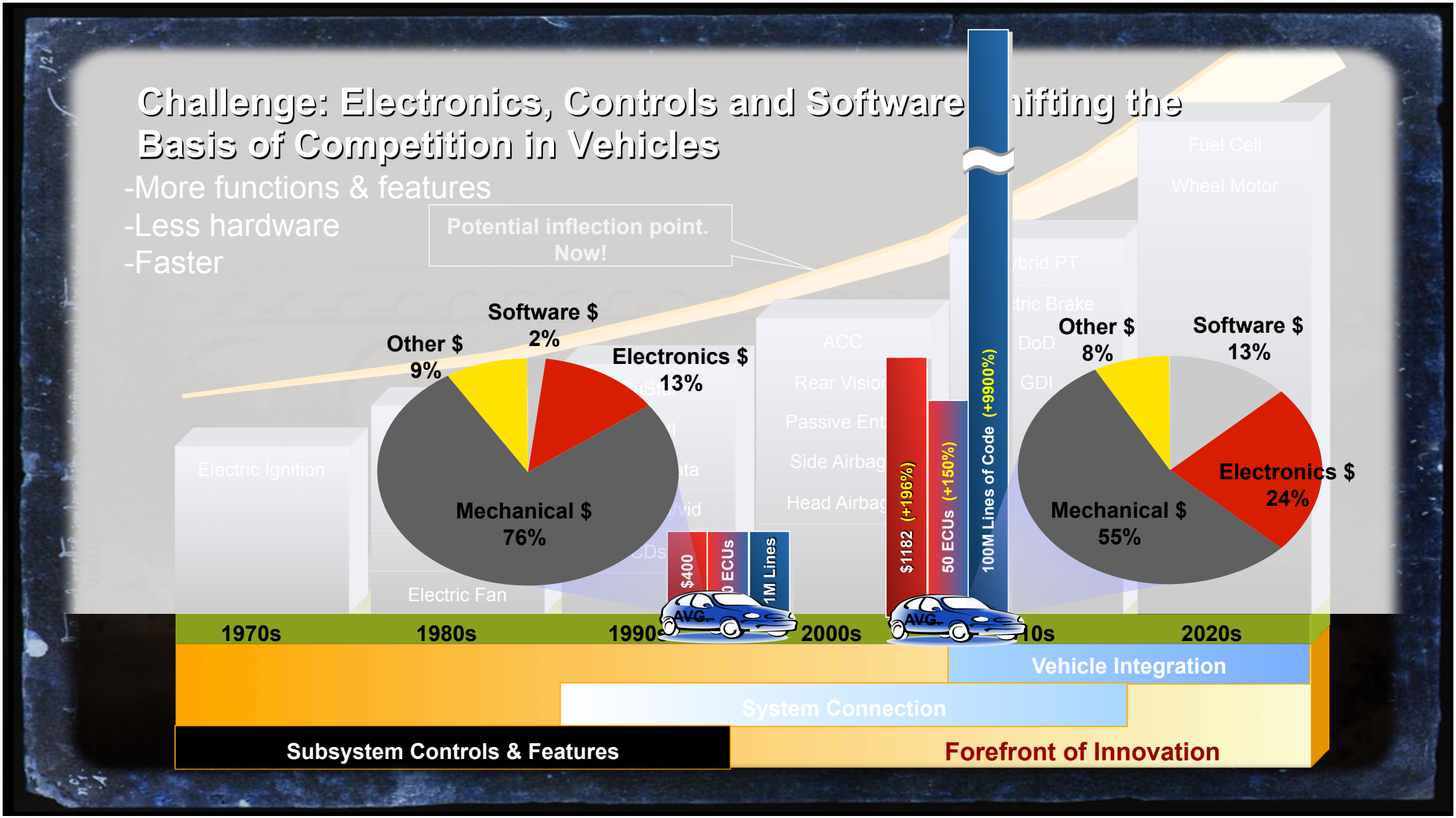
IC Vendors

freescale semiconductor, RENESAS, ST, Infineon, NEC

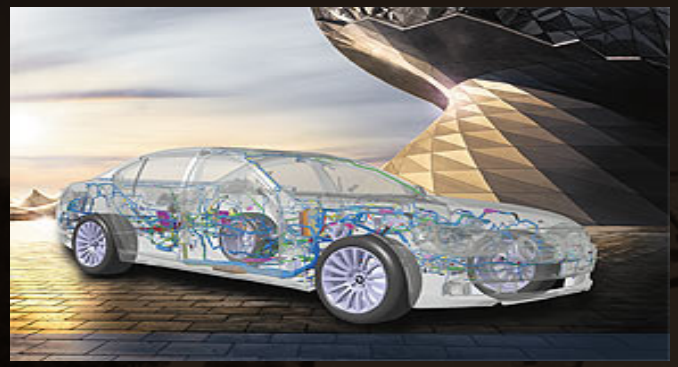
~15% of revenue from automotive

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

Source: Public financials, Gartner 2005


Challenge: Electronics, Controls and Software Shifting the Basis of Competition in Vehicles
-More functions & features
-Less hardware
-Faster
Potential inflection point. Now!
Other $ 9%
Software $ 2%
Electronics $ 13%
Mechanical $ 76%
$400
0 ECUs
1M Lines
AVG.
$1182 (+196%)
50 ECUs (+150%)
100M Lines of Code (+9900%)
AVG.
Other $ 8%
Software $ 13%
Electronics $ 24%
Mechanical $ 55%
Electric Ignition
Electric Fan
ACC
Rear Visio
Passive Ent
Side Airbag
Head Airbag
DoD
GDI
Fuel Cell
Wheel Motor
1970s
1980s
1990s
2000s
10s
2020s
Vehicle Integration
System Connection
Subsystem Controls & Features
Forefront of Innovation

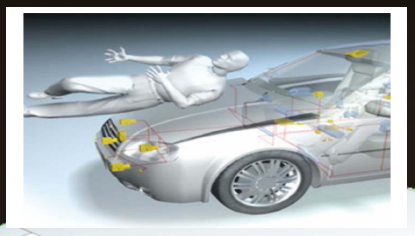
Challenges in Automotive Electronics Development

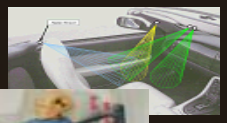


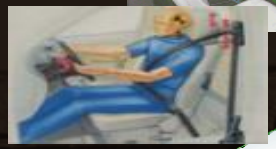
- Increasing functionality:
  - Safety (active/passive)
  - Fuel efficiency (hybrid)
  - Reduced emissions (less CO2)
  - Comfort
- Increasing quality:
  - 2000: ~1000 - 10ppm (per ECU)
  - 2010: ~1 - 0ppm (per ECU)

- Increasing value:
  - Electronic Share (value):
    - 2004: 20% -> 2015: 40%
  - Software Share (value):
    - 2000: 4.5% -> 2010: 13%
- Reduce time to market:
  - 2000: ~ 20 – 26 months
  - 2010: < 18-20 month

Convergence Towards Active Safety







Collision Avoidance

Traffic Sign Recognition

Pedestrian Protection

Lane Departure

Pre-Tensioner

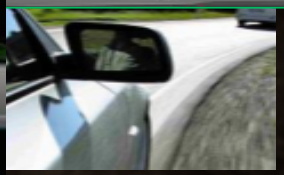
Lateral Control

LateralAir bag

Night Vision

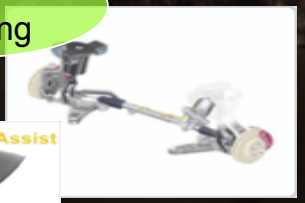
Airbag

Integrated Vehicle Stability



Active Front Steering

Electronic Stability



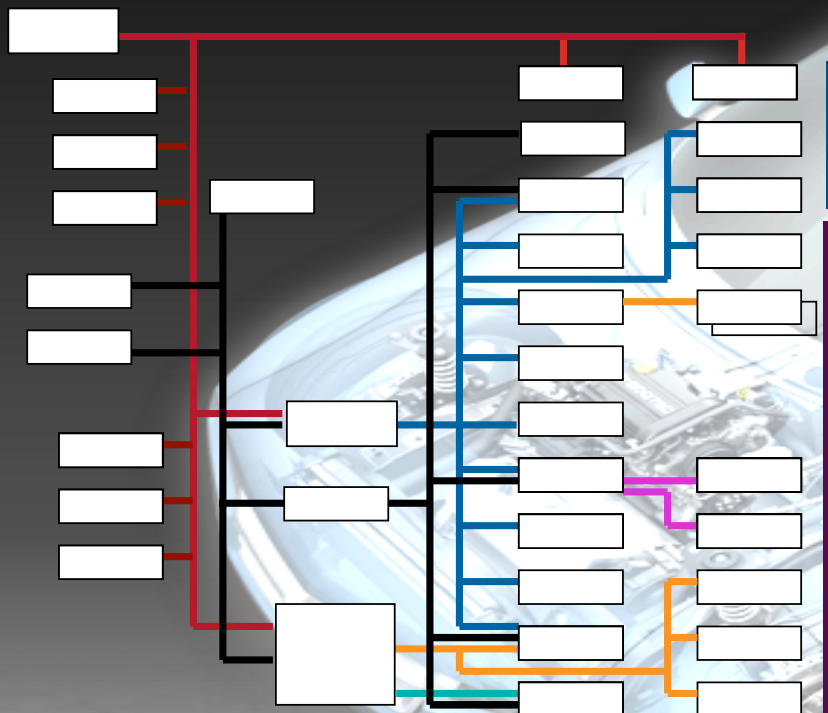
ABS

Electric Steering

Active Suspension

From federated to integrated architectures

Today:
Federated Architectures



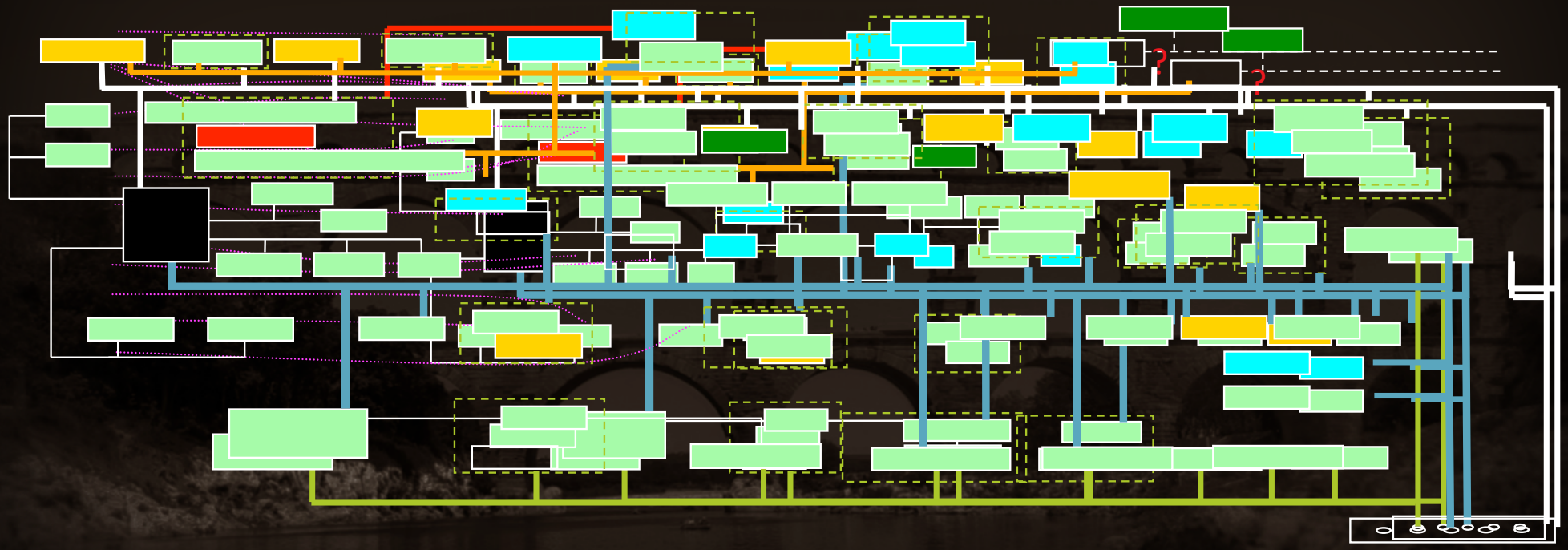
Each time a new function is required, the OEM starts a request to suppliers for a new ECU (an integrated HW/SW device realizing the function) to be integrated on the existing networks

The device is developed by the supplier with its own choice of HW, RTOS, device drivers and communication layers (with some standardization)

The result is

- Proliferation of ECUs (reaching 100)
- Complex distributed architectures with the need of high bandwidth and therefore multiple networks and gateways
- Complex functional and not-functional (timing) dependencies across the network, which OEMs struggle to control
- Missing opportunities for common set of libraries and (sub) functions
- Limited standardization, flexibility and extensibility
- Limited control on the execution platform by OEMs

The Distributed System Problem: Typical Car Electrical Architecture



Fighting Obsolescence:
How to harmonize fast evolving electronics with products whose lifetime is >10 years

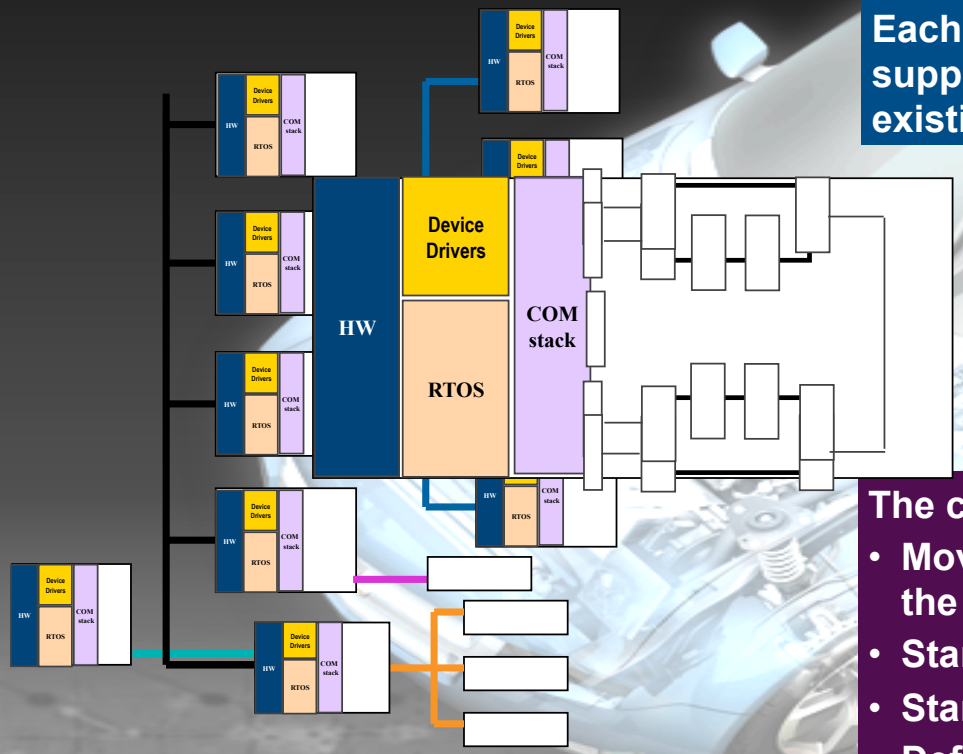
- Decouple Function and Architecture
- Bring back to OEM design control
- Flexibility and Extensibility of Architectures
- Manage a Complex Supply and Design Chain
- Place your bets so that you leverage maximally your core competence while leveraging as much as possible qualified PARTNERS

From federated to integrated architectures

Tomorrow?: Integrated Architectures

The execution architecture is completely selected and planned by the OEM. OEMs are free to standardize HW, drivers, RTOS and communication layers, leveraging competition among suppliers

Each time a new function is required, the OEM starts a request to suppliers for new functional content (SW) to be integrated on the existing platform



The challenges are:

- Moving from specifications of ECUs with message interfaces to the specs of SW components
- Standardize interoperability among components
- Standardize access to the platform services
- Define models that allow to predict the result of the composition (functional and not-functional)

The Larger Picture

PAGE 14 – SUNDAY, FEBRUARY 6, 2005 – THE NEW YORK TIMES (by Tim Mc

What's Bugging the High-Tech Ca

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 the problem, he took the van out and drove it until the oven fired up again. Then he rushed to the mechanic to look for a software error.

Additionally, the study found that altho errors cannot be removed, more than took two minutes for them to hook up diagnostic tool and find the fault," said Mills, senior technical staff member at I.B.M.'s T.J. Watson Research Center Hawthorne, N.Y. "I can almost see th software code, a sensor was bad."

Indeed, the high-tech comfort system confus the 200 sendin freezin loyal va up, thir billion,

Toyota Problems

The Washington Post, March 7

Attention has been focused on mechanical and electronic issues with Toyotas, but another possible cause of the runaway acceleration maybe a software glitch. Each vehicle contains layers of computer code that may be added from one model year to next" that control nearly every system, from acceleration to braking to stability. This software is rigorously tested, but t is well-known in our community that there is no scientific, firm way of actually completely verifying and validating software.

MOTOR T

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.

NHTSA has received 33 reports of stalling in Prius cars from model years 2004 and 2005, according to the agency's initial report. More than 85 percent of the cars that stalled did so at speeds between 35 and 65 miles per hour.

It’s Not Over Yet!

THE WALL STREET JOURNAL.
WSJ.com

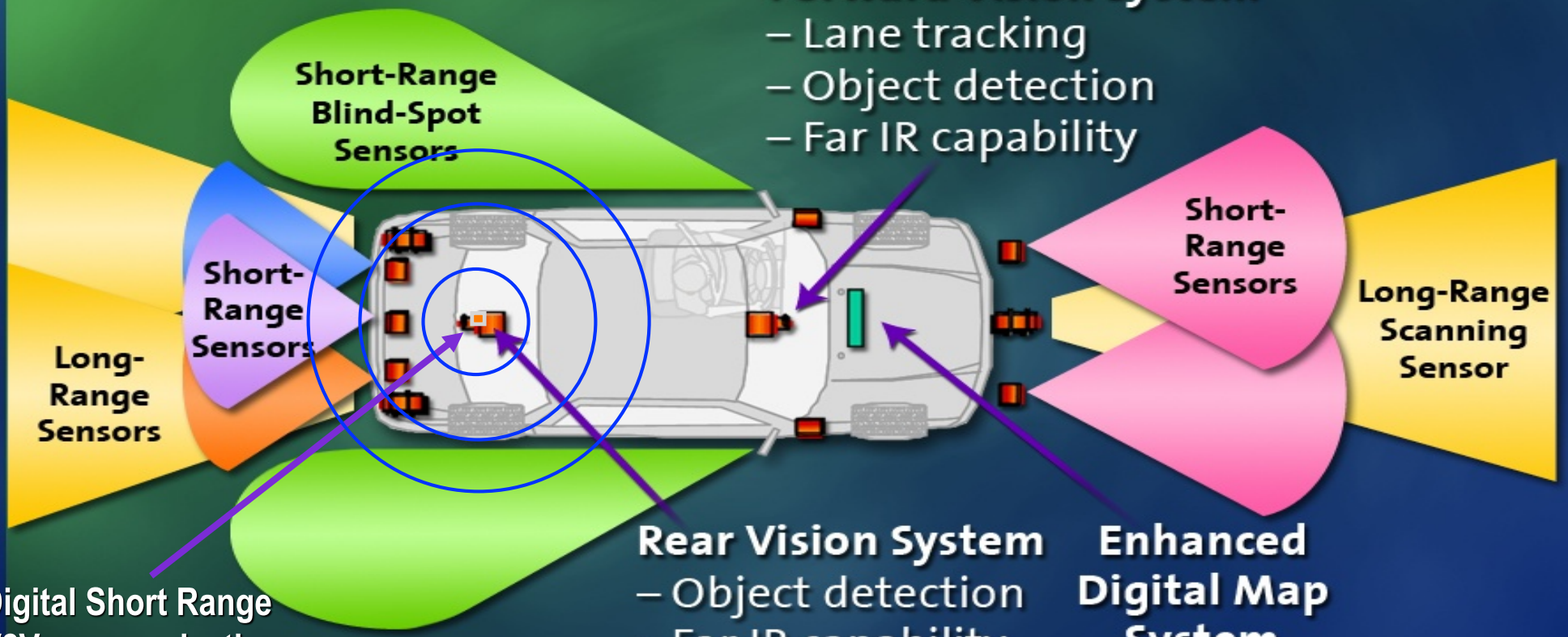
BUSINESS | NOVEMBER 25, 2010

Boeing 787 Is Set Back as Blaze Forces Fix

By PETER SANDERS

The refuse-to-collide car!

360° Safety with Integrated Sensor Strategy



CMOS mmWave Circuits and SoC: 60GHz Today

- Multiple 60GHz standards complete
- WirelessHD products available
  - SiBeam (BWRC startup)
  - Wall-powered
  - Dissipate <2W
- A $10 Radar is a possibility!
- 60GHz link in mobile applications?
  - Energy-efficiency is key: <~250mW transceiver
  - Solution must scale to 10+ Gb/s
  - Low cost = Single chip RF+phased array+BB+BIST

Integration Challenges: Plug and Play?

Plug and Pray!

The Design Integration Nightmare

Specification:

Implementation:

P. Picasso,
Blue Period

P. Picasso
“Femme se coiffant”
1940

To Enable Success…

We need an integration platform

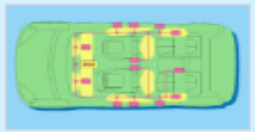
- To deal with heterogeneity:
  - Where we can deal with Hardware and Software
  - Where we can mix digital and analog, cyber and physical
  - Where we can assemble internal and external IPs with different physical domains
  - Where we can work at different levels of abstraction
- To handle the design chain
- To support integration
  - Tool integration
  - IP integration

The integration platform must subsume the traditional design flow, rather than displacing it

Outline

- Introduction and Motivation using Automotive as Test Case
- The V design process and Platform Based Design
- The Role of Autosar
- Semiconductor Design Economics
- Extensions and Open Issues

Automotive V-Models: a ‘Linear’ Development Process



Development of Car System

Development of Sub-System

Development of Mechanical Part (s)

ECU Development

ECU SW Development

ECU HW Development

ECU SW Implementation

Car System Sign-Off!

Sub-System(s) Integration, Test, and Validation

Sub-System Sign-Off!

ECU/ Sens./Actrs./Mech. Part(s) Integration, Calibration, and Test

ECU Sign-Off!

ECU HW/SW Integration and Test

ECU HW Sign-Off!

ECU SW Integration and Test

ECU:
Electrical Control Unit

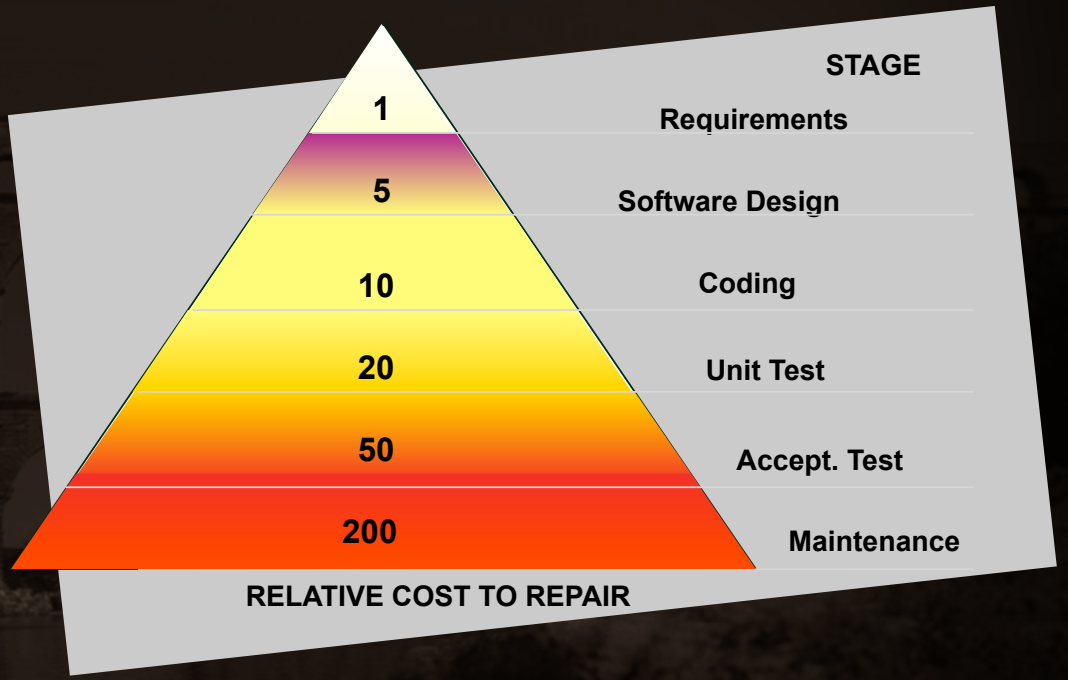
Vertical Design Chain and Design Error Costs

Sub-System Requirements (e.g. ABS)

Sub-System Implementation

Components Requirements (e.g. ABS)

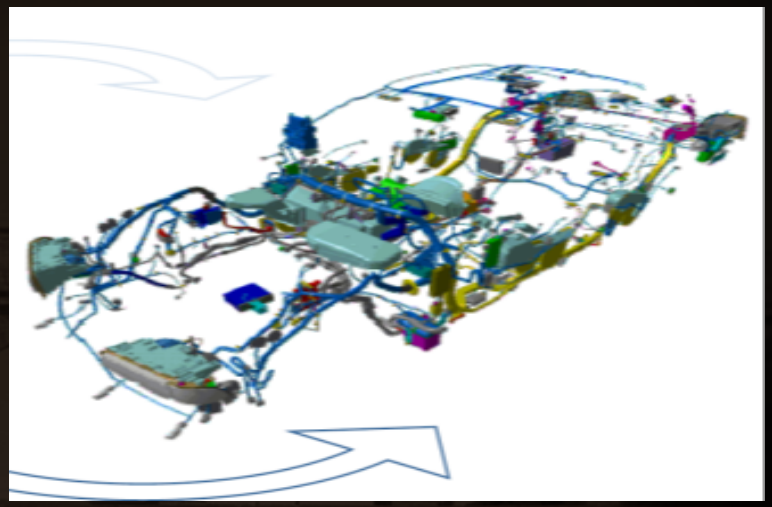
Component Implementation



"Finding and fixing requirements errors consumes between 70% - 85% of total project rework costs."

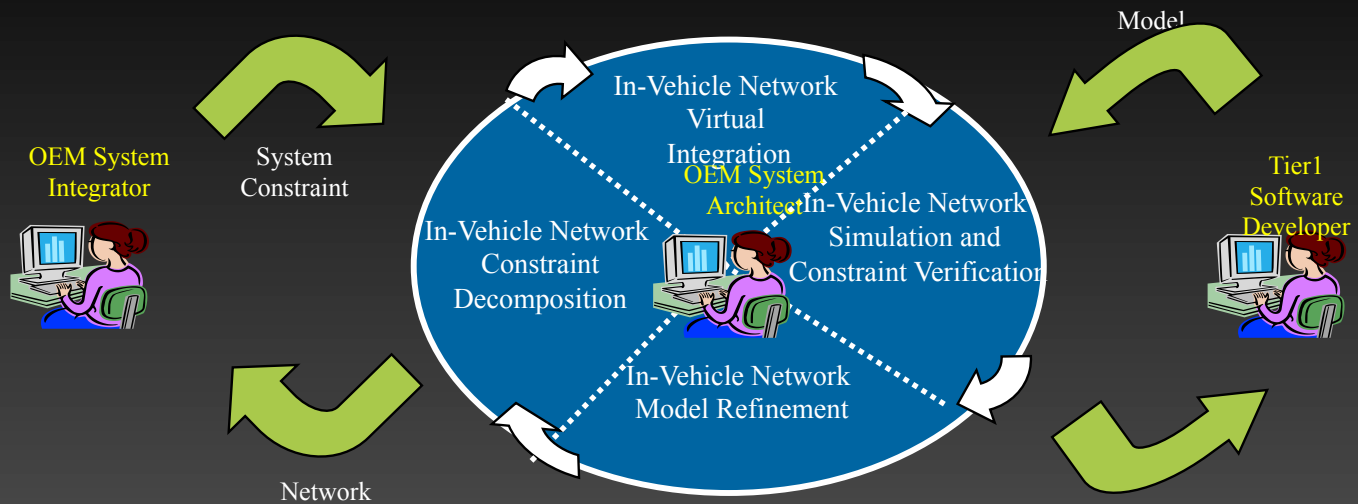
OEM Design Challenges

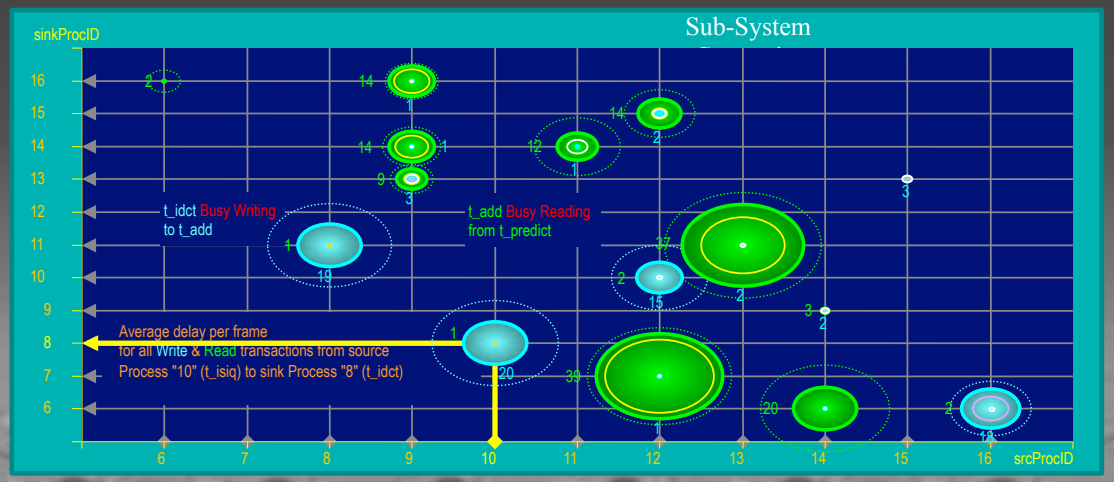
- Distribution:
  - Functions are distributed over several subsystems/ components (M/E/E)
  - Exploration of solutions involve several subsystems
  - Integration is very expensive
  - Tool support is critical



- Packaging:
  - space for electronics in the car is reducing
  - Functional integration might be different from functional one

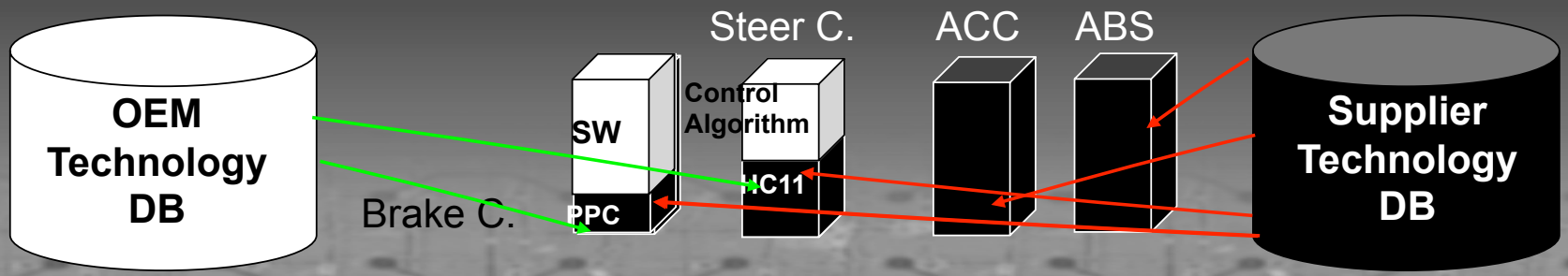
Enabling OEM and Tier1 Co-Design Space Exploration and Co-Verification



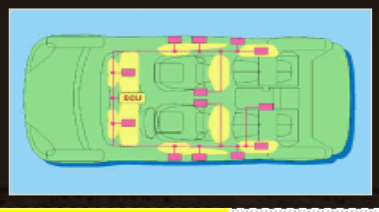


Derivative design

- **The derivative design approach:**
  - Every two-three years a new generation of products is designed
    - Product generations are conceived to accommodate the specification of all customers for the next years
  - For each commitment, the electronic control unit is obtained by derivation from the current generation
- **In the derivative design approach, reuse is extensively employed to minimize cost and development time**
  - for each class of applications, products are variants of a same originating design



Platform Models for Model Based Development



Development of Distributed System

Distributed System Requirements

Distributed System Partitioning

Sub-Systems *Model Based* Development

Sub-Systems Requirements

Network Protocol Requirements

Sub-System(s) Sign-Off!

Platform Abstraction

Distributed System Sign-Off!

Sub-System(s) Integration, Test, and Validation

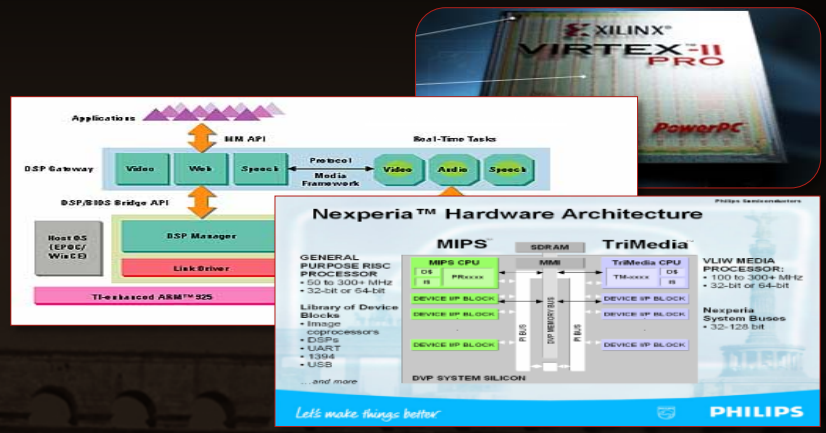
Virtual Integration of Sub-System(s) w/ Network Protocol, Test, and Validation

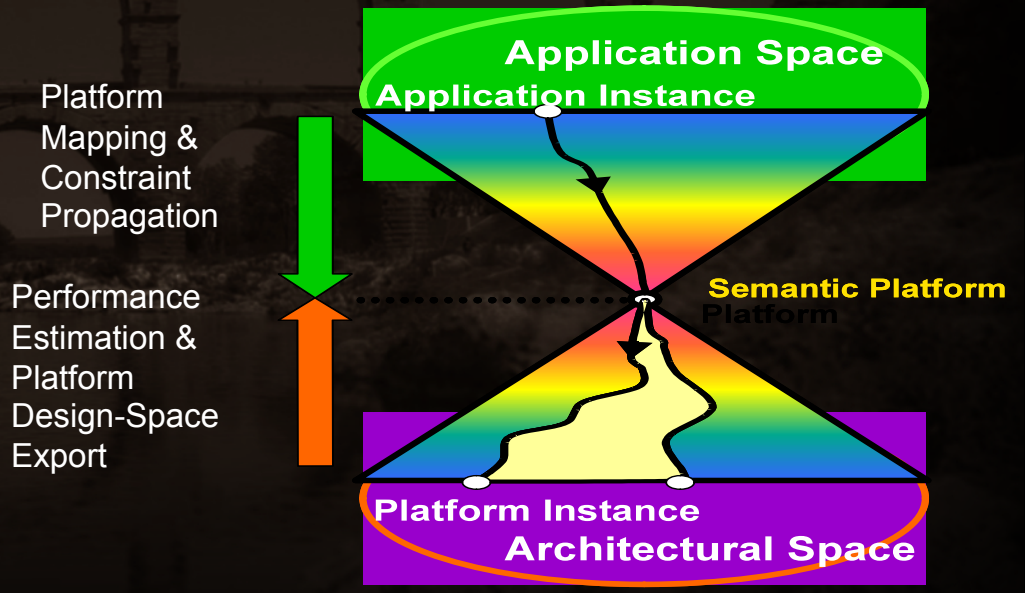
Sub-System(s) Implementation Models Sign-Off!

Network Communication Protocol Sign-Off!

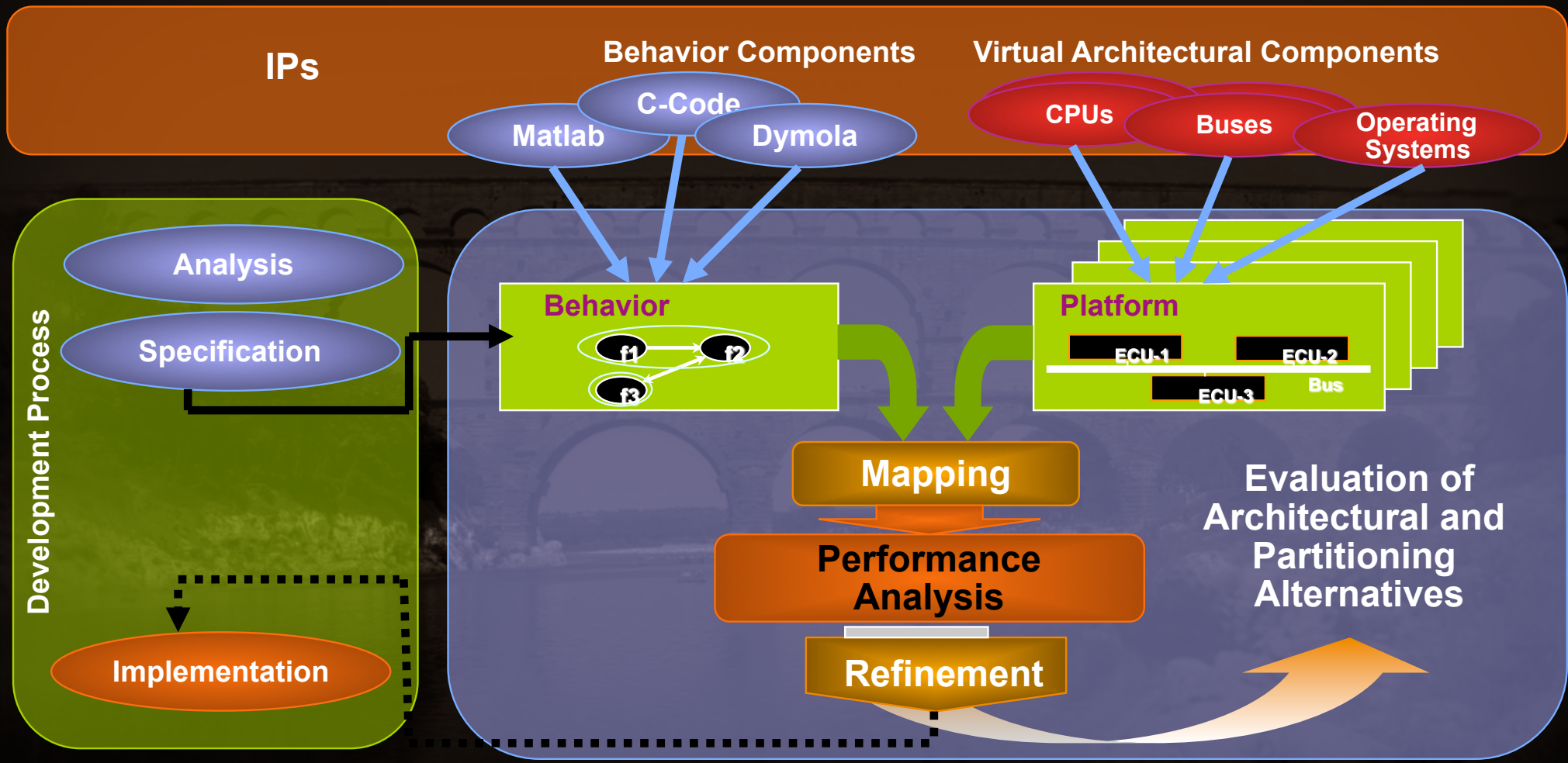
Platform Based Design

- A “meet-in-middle” design method
  - Platform: an abstraction layer that hides the details of several possible implementation refinements of the underlying layers
- Function model:
  - abstraction of what the system is supposed to do
- Architecture model:
  - lower level of abstraction describing how the system realizes the function
- Mapping:
  - Process by which function and architecture meet
  - Propagates constraints from above to meet performance estimations from below
  - Phases: Allocation, binding, scheduling

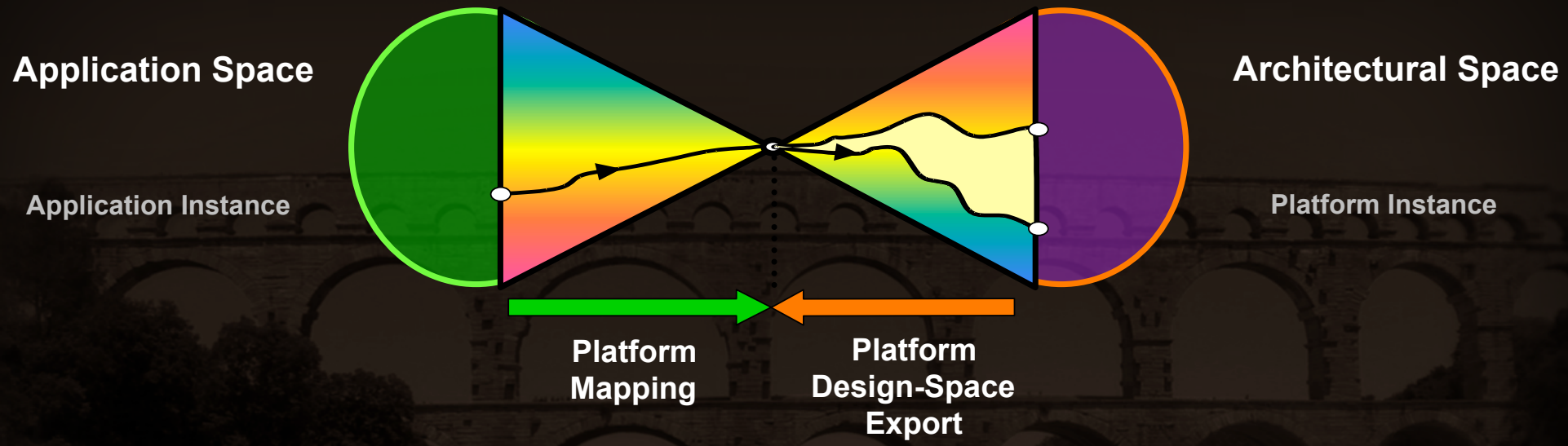




Separation of Concerns



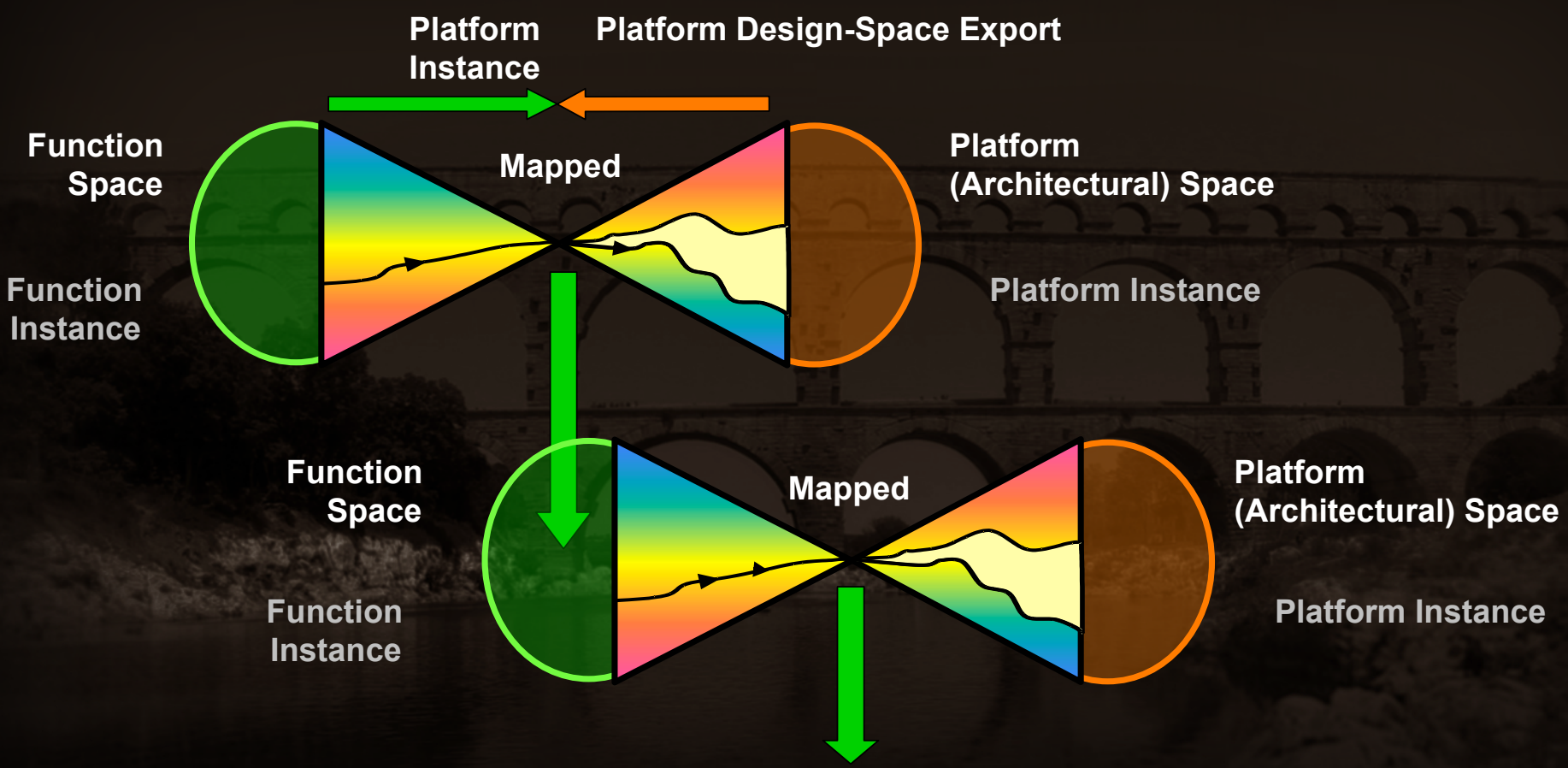
Platform-Based Design



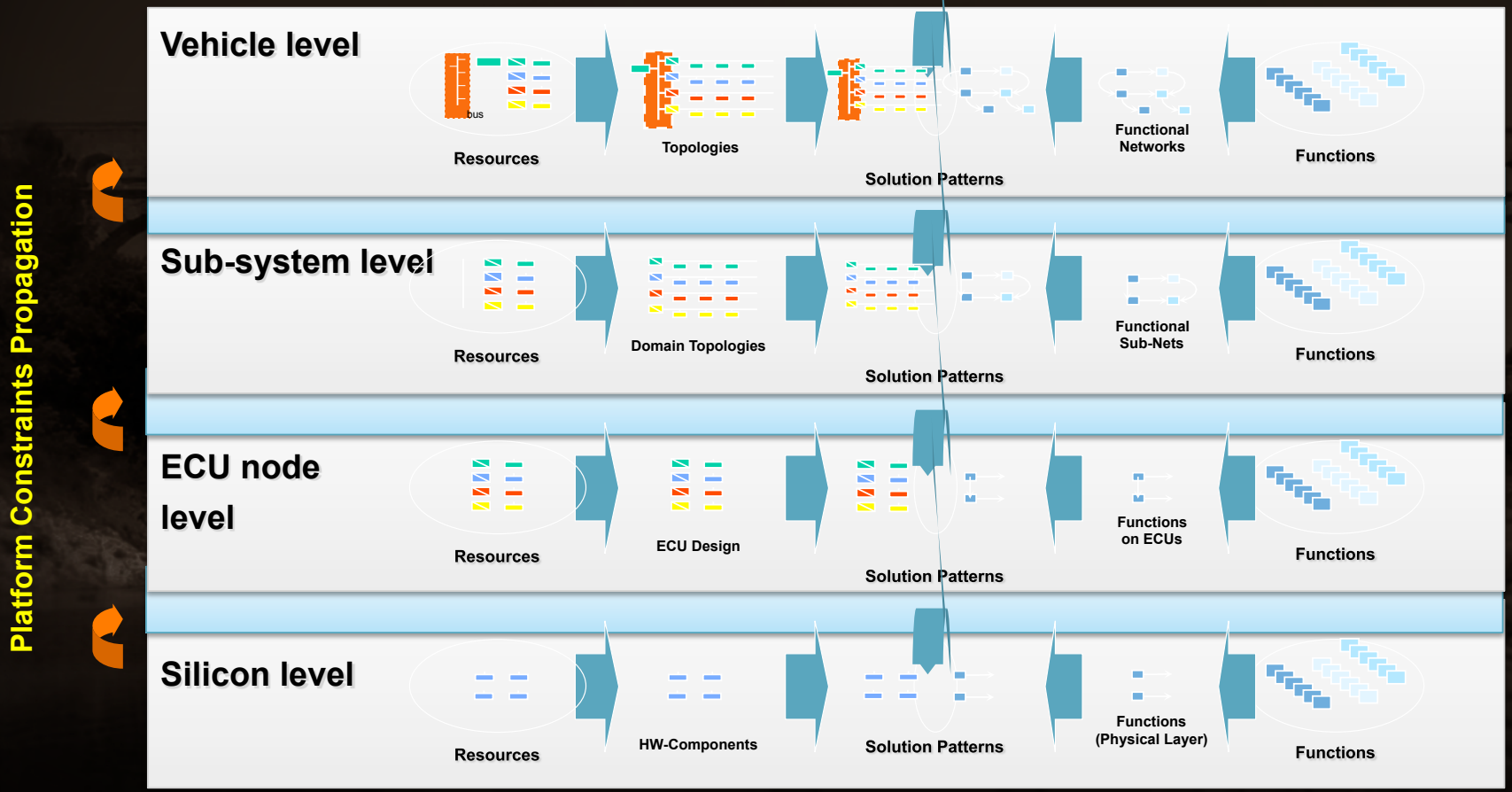
Platform: library of resources defining an abstraction layer with interfaces that identify legal connections

- Resources do contain virtual components i.e., placeholders that will be customized in the implementation phase to meet constraints
- Very important resources are interconnections and communication protocols

Fractal Nature of Design

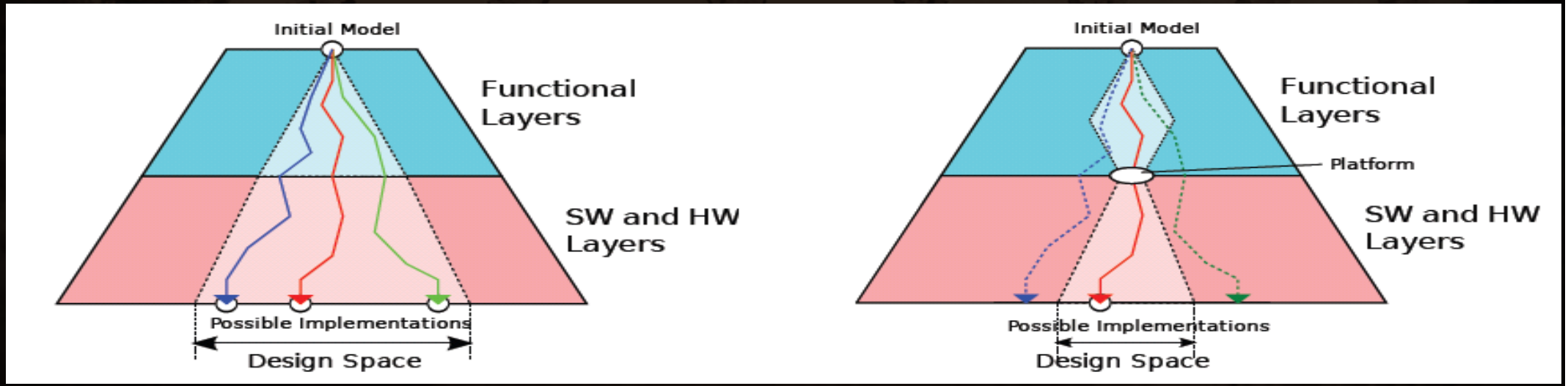


Design Methodology



Design Space vs. Time-to-Market in Platform Based Design

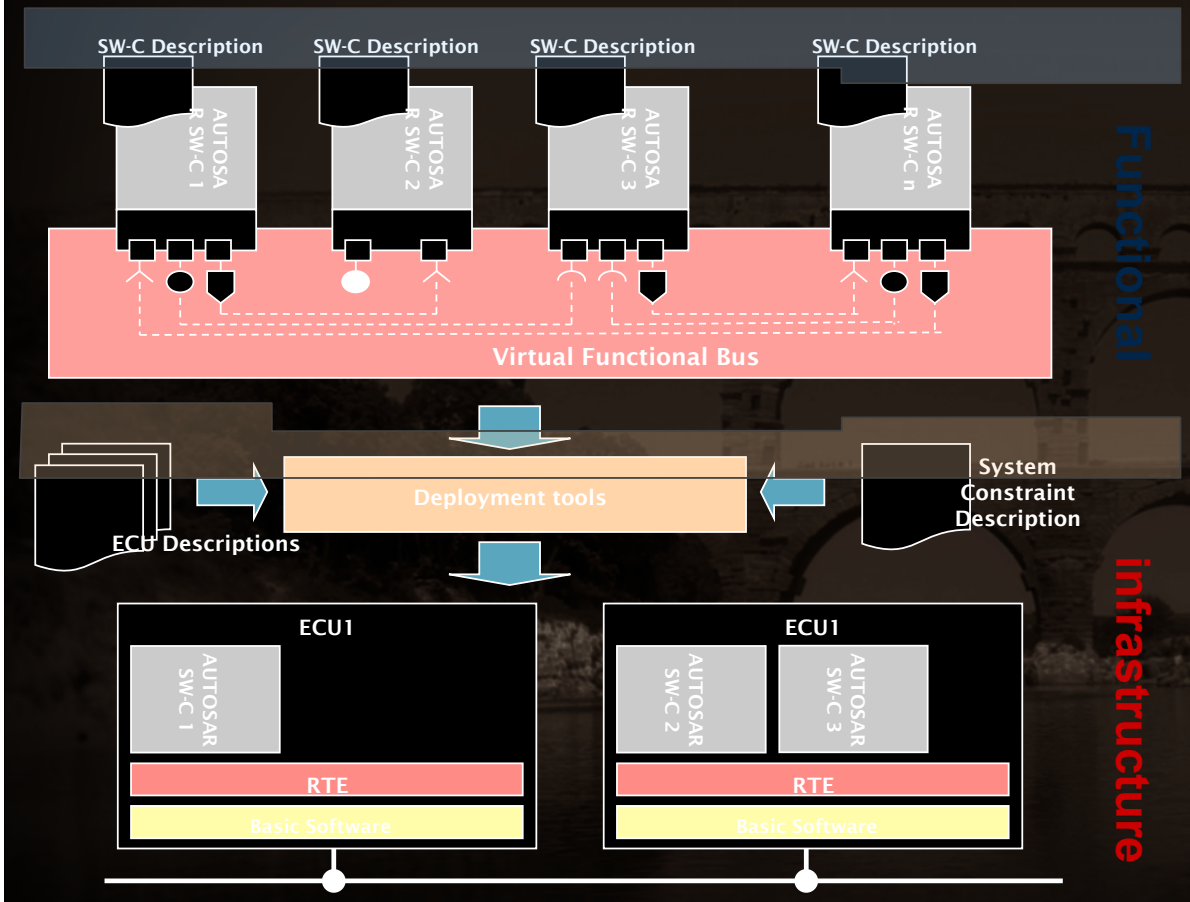
- Platform limits the design choices
- Designer only needs to analyze alternatives that are implemented by the platform



AUTOSAR initiative

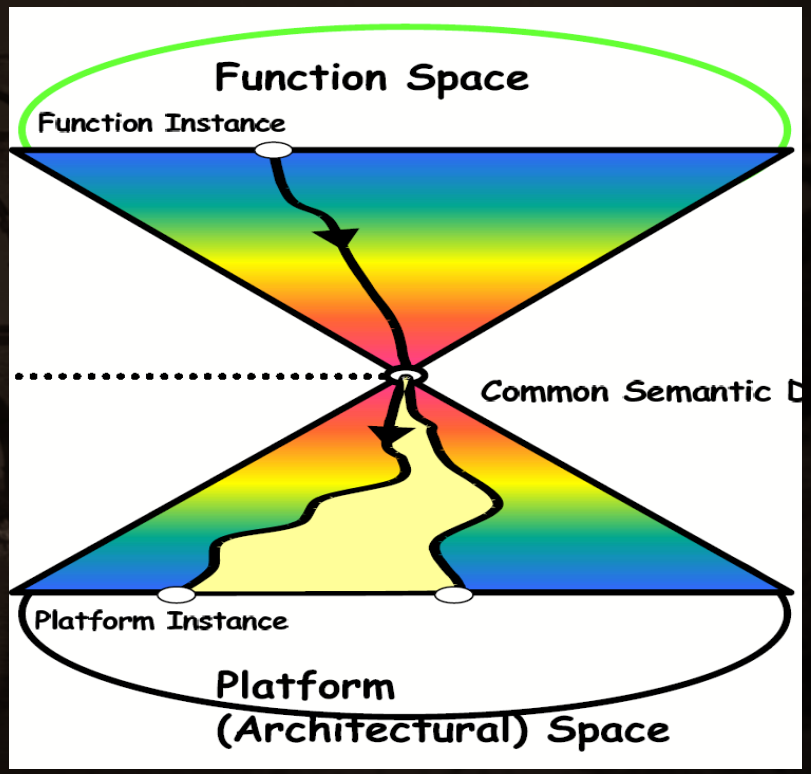
- Each ECU may include functionalities developed by different suppliers and as well as the OEM
- Started as a partnership between leading European OEMs and Suppliers for
  - establishment of an open standard for automotive E/E architecture
  - a basic infrastructure for the management of functions within both future applications and standard software modules
- Objectives
  - standardization of basic system functions and functional interfaces
    - modularity and scalability
  - the ability to integrate and transfer functions
    - transferability and re-usability
  - substantially improve software updates and upgrades over the vehicle lifetime
- The AUTOSAR scope includes all vehicle domains

AUTOSAR Architecture

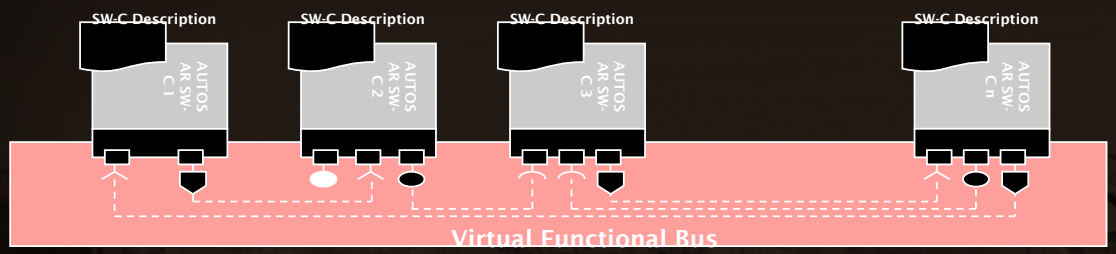


A fundamental concept of AUTOSAR is the separation between:

- **(functional) application** and
- **infrastructure**



AUTOSAR Architecture and Methodology

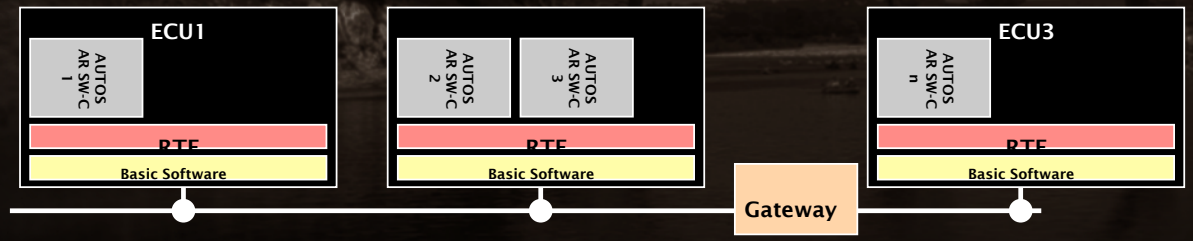


Functionality is described as a logical network of “Software Components” (SW-C)

The ”Virtual Functional Bus“ validates the interfaces before software implementation.

Mapping of “Software Components” to ECUs and configuration of basic software.

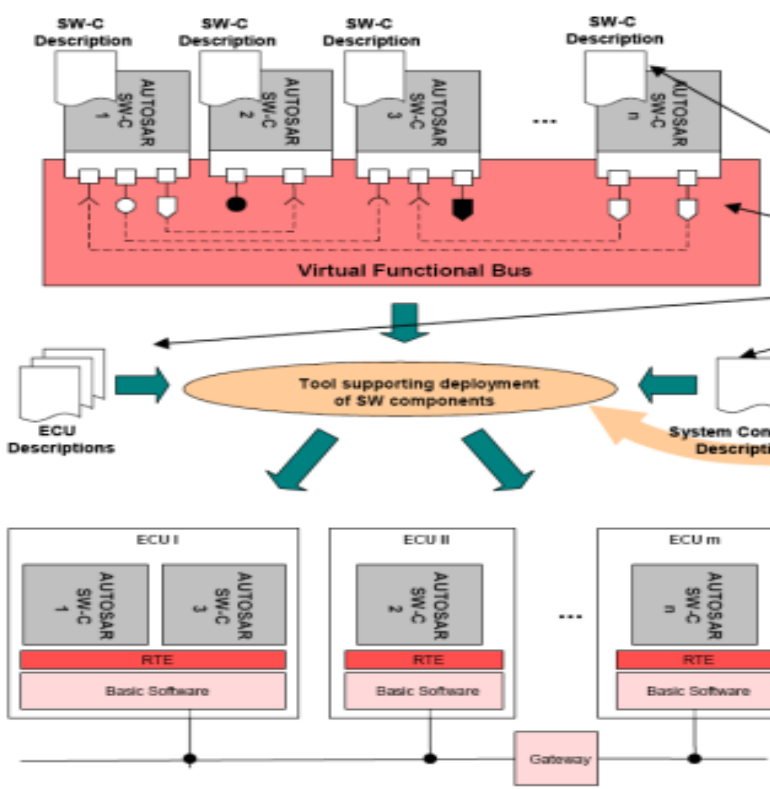
Automatic generation of configuration files for building the set of software applications



The AUTOSAR Platforms introduces the “Run Time Environment” and “Basic Software” to abstract the physical network topology and hardware

Autosar Standardization and Tools

AUTOSAR Methodology



Standardized description templates for application software components (interfaces and BSW requirements)

Standardized exchange formats and methodology for component, ECU, and system level

Tools for
- support of component mapping
- generation of RTE, i.e. inter- and intra ECU communication

Standardized Basic Software (BSW) architecture, detailed specifications for implementation and configuration of BSW

Control synthesis - algorithm development

Characteristics of the overall electronic control system

- Multi-rate control system composed of nested control loops that interact with other embedded controllers
  - frequency and phase drifts between sampling frequencies
  - event driven actions
  - asynchronous communication on the network
- Implements both continuous and discrete functionalities
  - more discrete than continuous
  - control algorithms may have many operation modes
    - nominal operation modes
    - safety, protection and recovery modes
  - computations performed at transition time are very important
    - switching conditions
    - controller initializations
- A large part of algorithms devoted to diagnosis, fault tolerance and safety

Complexity: more than 150 I/O and 200 algorithms in engine control units

Control synthesis - algorithm development

Electronic control units do not implement control and estimation algorithms only

- **Fault tolerance**
  - recovery algorithms that guarantee minimal operability under fault conditions
- **Diagnosis**
  - Diagnosis specs, enforced by OBDII (On Board Diagnosis II - USA) and EOBD (European On Board Diagnosis EU), require that
    - every fault, malfunction or simple component degradation that lead to the production of pollutant emissions over given thresholds should be diagnosed and notified to the driver
- **Safety**
  - relevant to vehicle longitudinal and stability control, and next-generation X-by-wire systems
    - e.g. brake-by-wire and steer-by-wire

Specification for hw/sw implementation

The description of the hw/sw specification have to

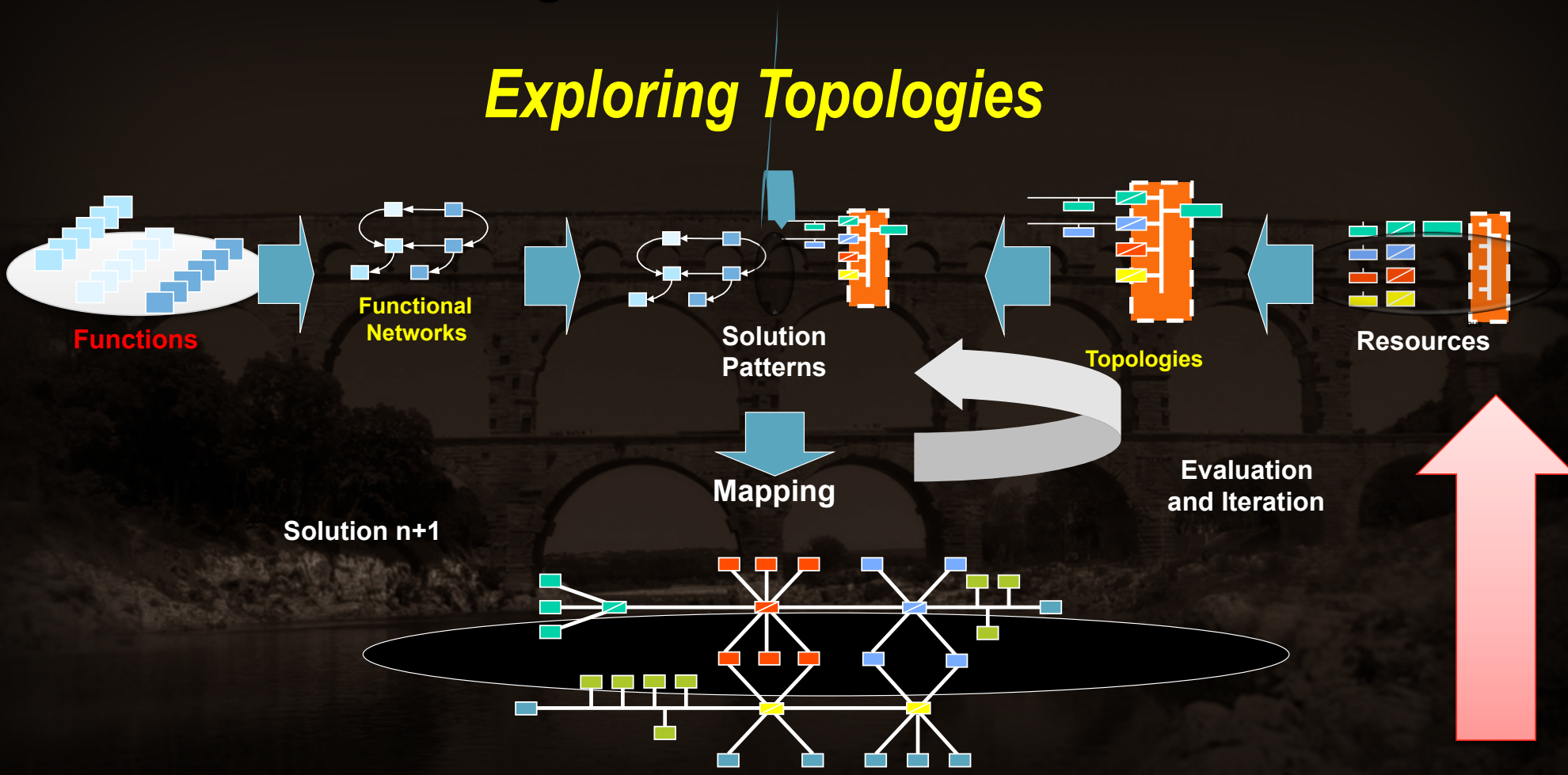
- include all the details for a correct implementation of the algorithms
  - complete functional description
  - computation accuracy bounds
    - value domain: computation precision (fixed-point arithmetic), threshold detection, ...
    - time domain: latency, jitter, delay in event detection, ...
  - execution order, synchronization and communication
  - priorities in case of shared resource (cpu, communication,etc)
  - data storage requirements
- be model-based
- be suitable for automatic code generation
- be compliant to AUTOSAR middle-ware RTE layer specification

Specification for hw/sw implementation

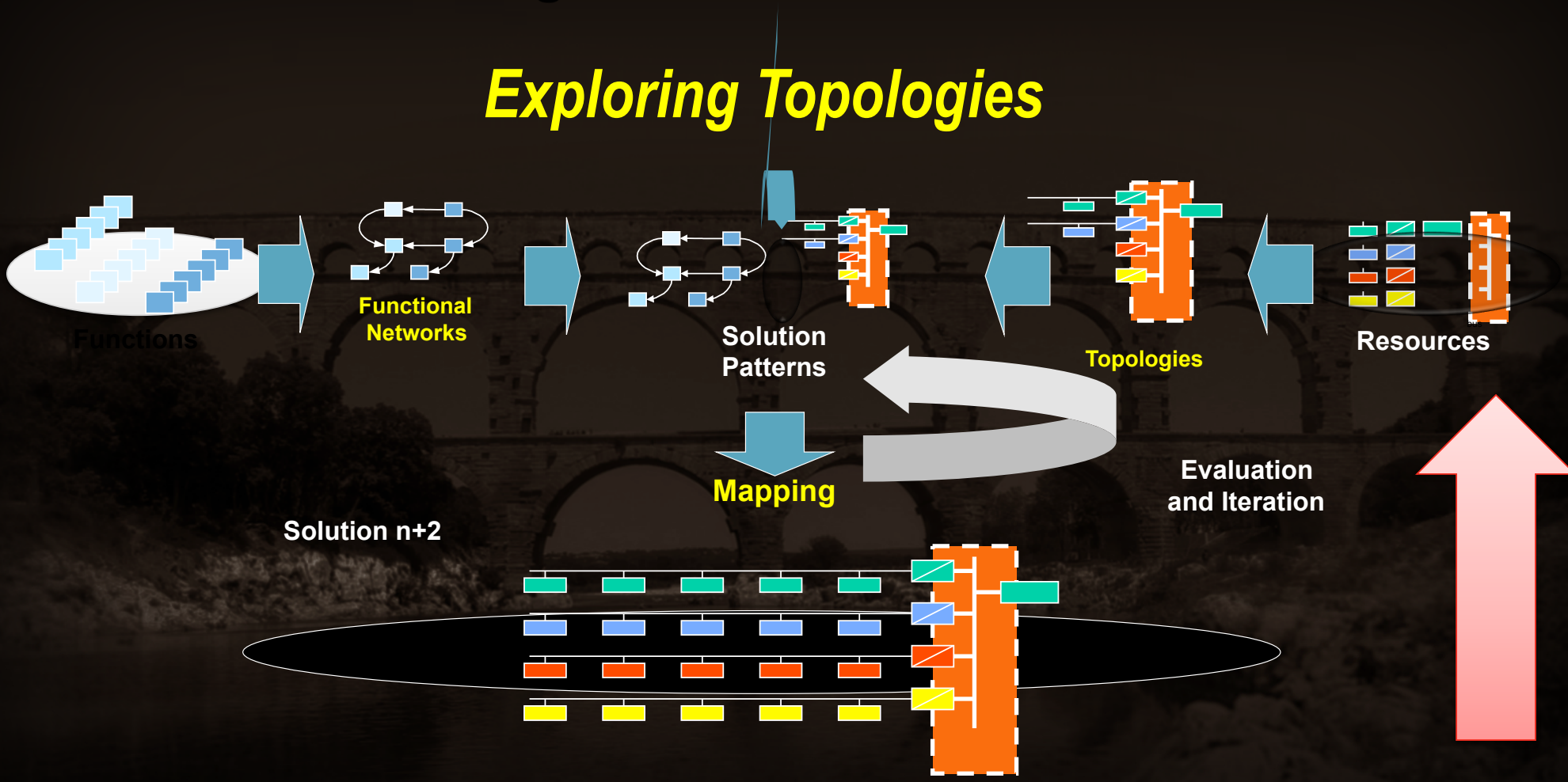
- Description of the implementation specification (hybrid formalisms)
- Methodologies and tools for the definition and validation of implementation constraints
  - modeling of the degradation due to the implementation of algorithms on bounded resource platforms
  - definition of acceptance criteria for the hw/sw implementation
  - exploration of hw/sw implementation requirements and constraints
  - validation of candidate implementation platforms described in abstract form
- Tools supporting the specification for hw/sw implementation have to
  - allow the description of the implementation constraints and acceptance criteria
  - be efficiently integrated with software development tools
  - either provide automatic code generation or be linked to auto-coding tools

Platform Based Design

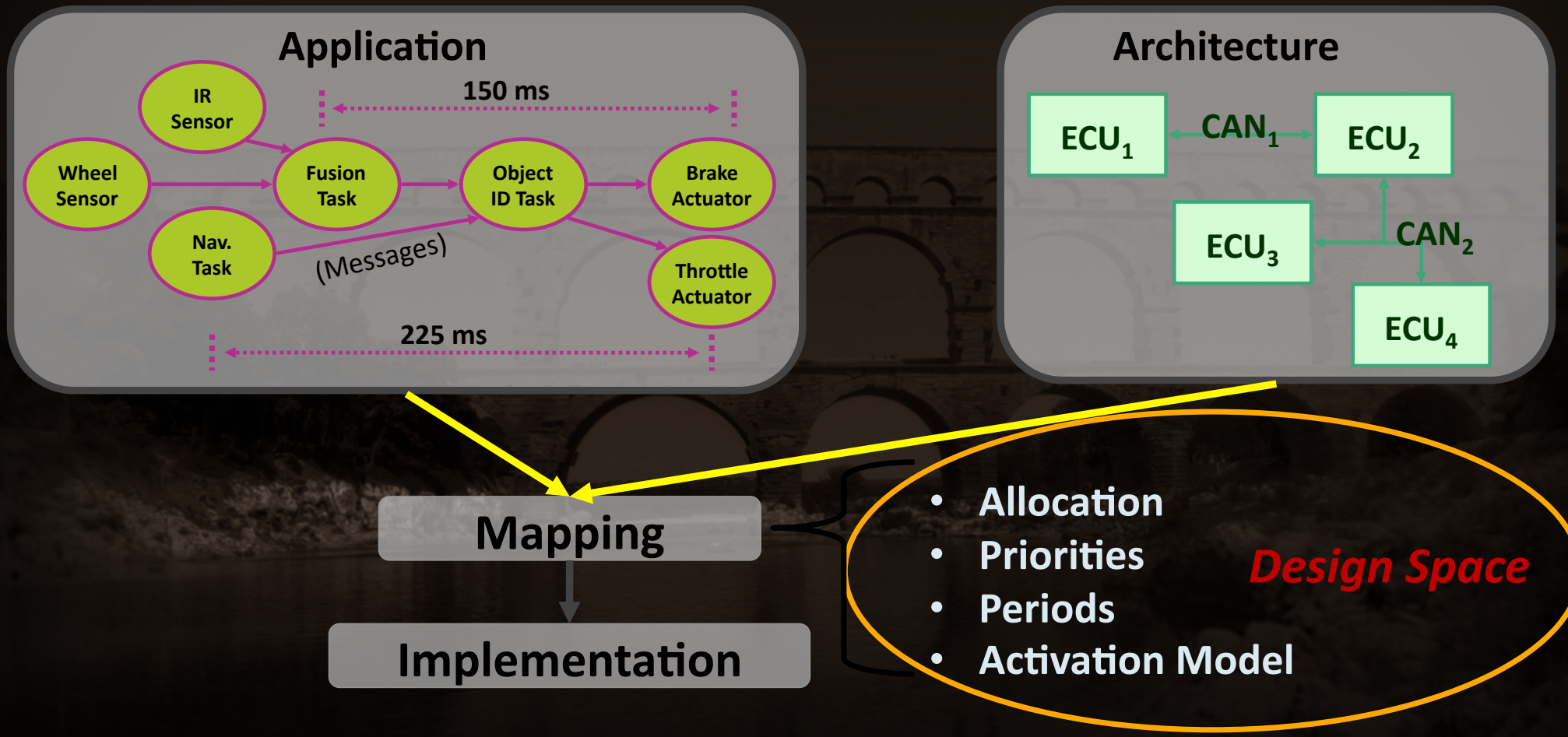
Exploring Topologies



Platform Based Design



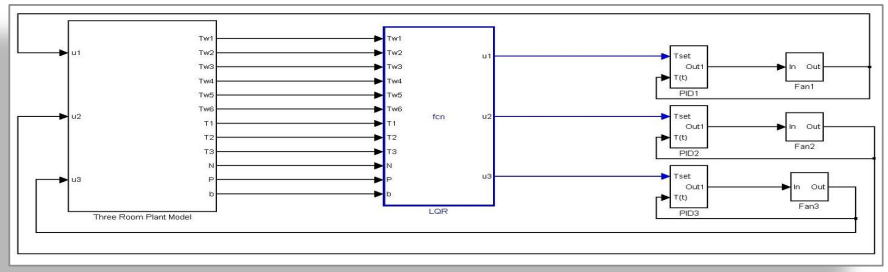
Design Flow


Application
IR Sensor
150 ms
Wheel Sensor
Fusion Task
Object ID Task
Brake Actuator
Nav. Task
(Messages)
Throttle Actuator
225 ms
Architecture
ECU1
CAN1
ECU2
ECU3
CAN2
ECU4
Mapping
Implementation
Allocation
Priorities
Periods
Activation Model
Design Space

MBD: CODE GENERATION

e.g. Mathworks RTW, dSpace TargetLink

High level input models
(Simulink, Modelica, …)



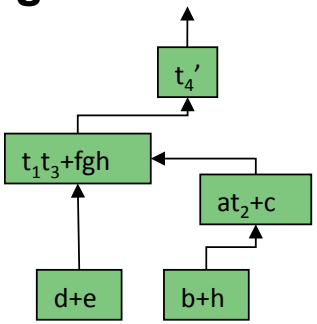
Direct code generation

- *No significant restructuring*
- *Low level optimization*
- *Manual partition*

Target code

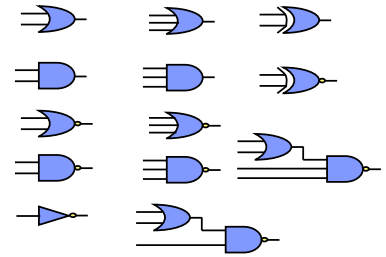
LEARNING FROM LOGIC SYNTHESIS

High level function model

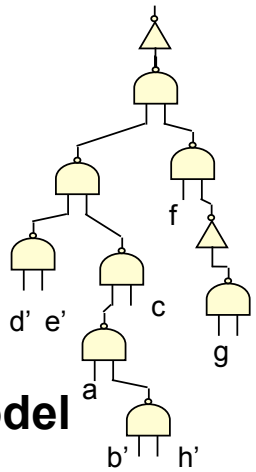


- Separation of func and arch
- Common language for func and arch netlists (Boolean logic, NAND2 gate)
- Automatic mapping

Gate library (platform)

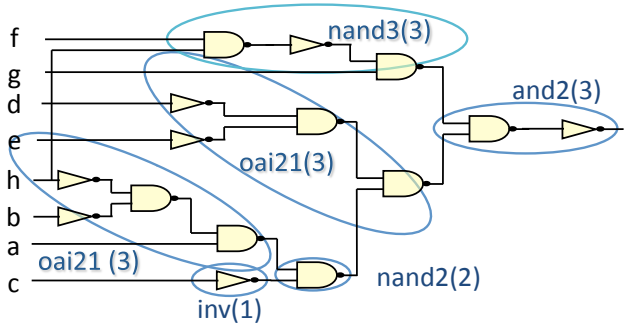


restructuring

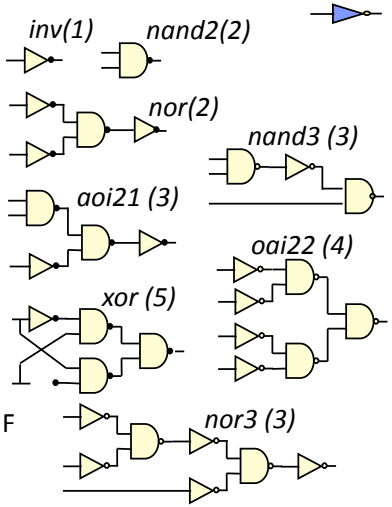


Function model in netlist

Technology Mapping (covering)



Mapped design

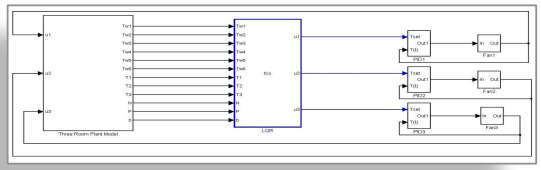


restructuring

Gate library in netlist

OUR SOFTWARE SYNTHESIS FLOW

Function Model

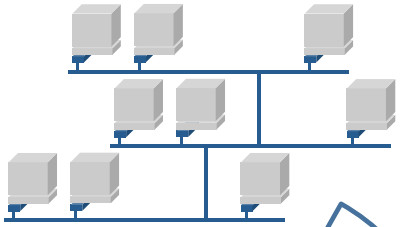


Stage 1: Common modeling domain (CMD) selection

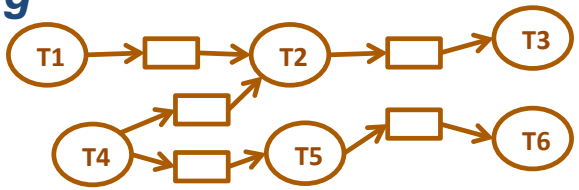
Common semantics for func and arch

Primitives to decide abstraction level

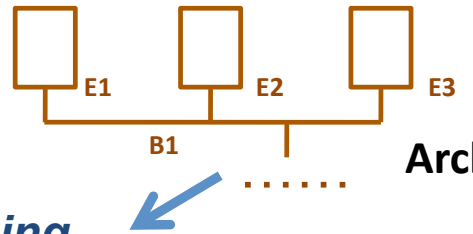
Architecture Platform



restructuring



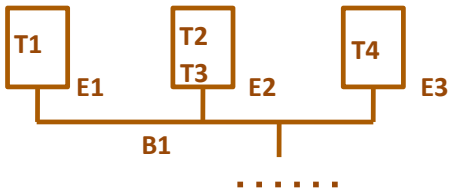
Function Model in CMD



restructuring

Architecture Model in CMD

Stage 2: Automatic mapping



Mapped Design in CMD

Stage 3: Code generation

CHALLENGES IN THE FLOW

- Stage 1: Common modeling domain selection
  - Various models of computation exist in system level.
  - Trade-off between expressiveness and ease of manipulation when selecting the common semantics.
  - Trade-off between granularity and optimality when selecting the primitives.
- Stage 2: Automatic mapping
  - Various constraints and objectives.
  - Domain-specific algorithms may be used albeit not necessary.
- Stage 3: Code generation
  - Communication interface synthesis maybe needed to guarantee correct semantics.

MODELING DOMAIN

- Semantic domain Q - *the language*
  - Formally defined as trace-based agent algebra [1].
  - $Q.D$: domain of agents - "building blocks".
  - $Q.A$: master alphabet – "set of all signals between blocks".
  - $Q.\alpha : Q.D \rightarrow 2^{Q.A}$, each agent has an alphabet – "each block has a set of signals"
  - Operators: renaming, projection and parallel composition – "rules to initialize and compose blocks"
- Primitives P – *abstraction level*
  - Primitives are a set of agents in a semantic domain, $P \subseteq Q.D$.
  - No agent in P can be constructed from other agents in P.
- Modeling domain $C_Q(P)$: all agents constructed from primitives P by applying operators in semantic domain Q.

[1] R. Passerone, *Semantic Foundations for Heterogeneous Systems. PhD* thesis, University of California, Berkeley, 2004.

COMMON MODELING DOMAIN (CMD)

- A model is an agent in the modeling domain.
- Function model $f \in F$, architecture model $a \in A$.
- $B(s)$ denotes the behavior of model s.
- Modeling domain M is a common modeling domain between f and a if there exists $f' \in M$ and $a' \in M$, s.t. $B(f') \subseteq B(f)$ and $B(a') \subseteq B(a)$.

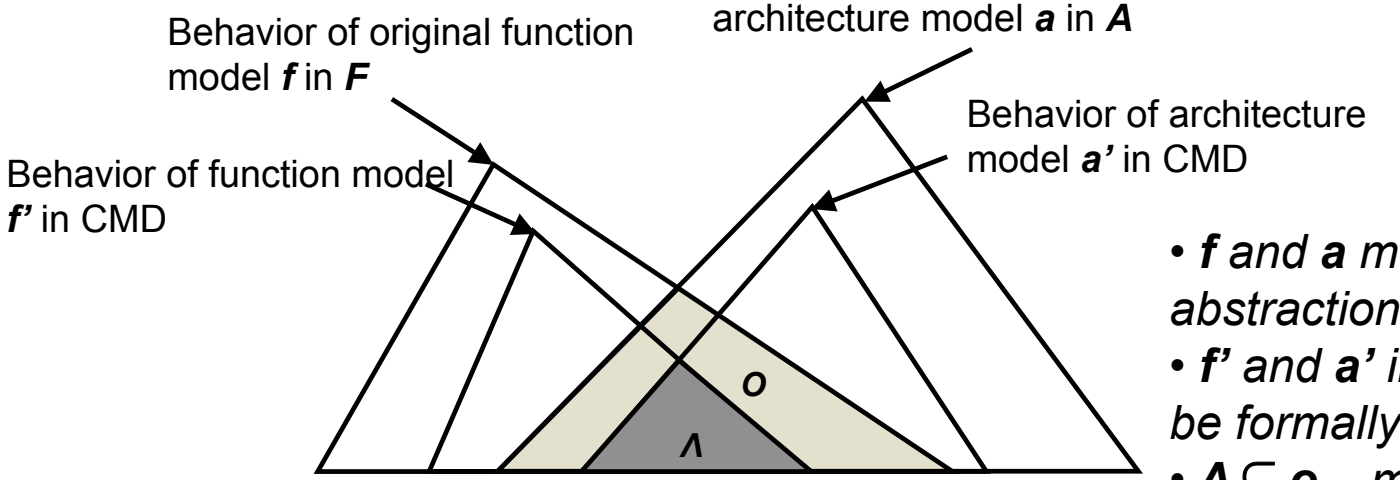
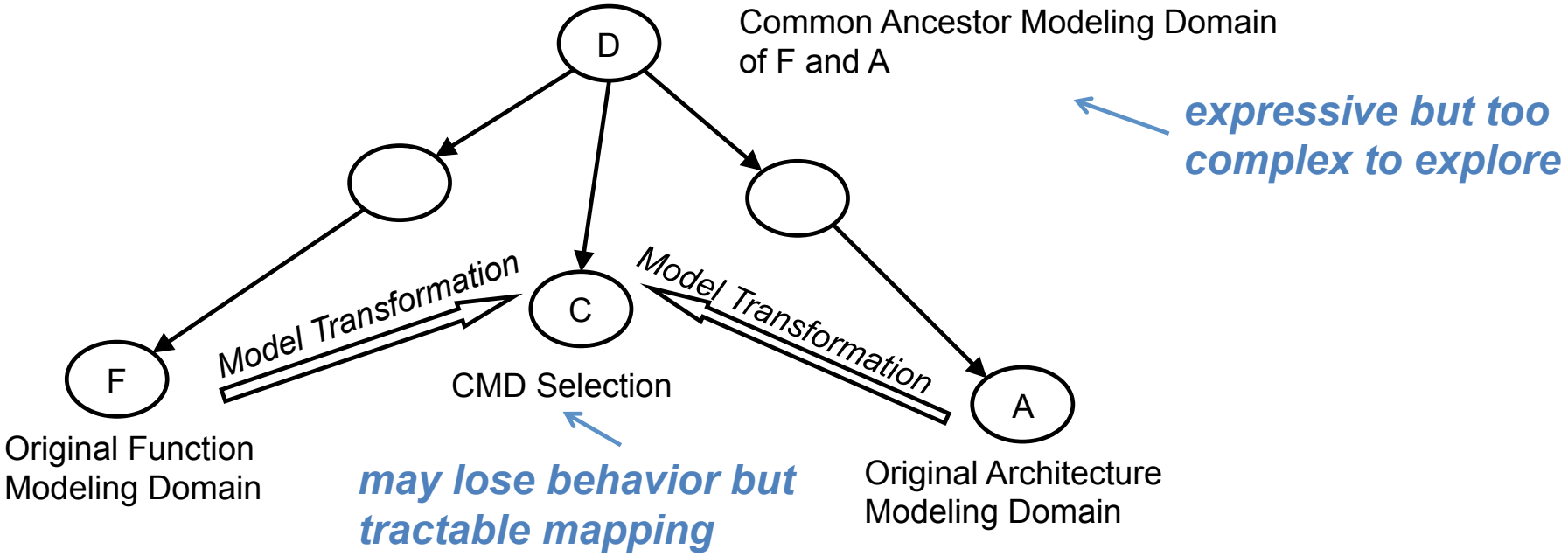


Illustration of mapping space in CMD

- ***f*** *and* ***a*** *may have different semantics or abstraction level – hard to explore* ***O****.*
- ***f'*** *and* ***a'*** *in CMD – mapping space* ***Λ*** *can be formally explored.*
- ***Λ*** $\subseteq$ ***O*** *– mapped behavior is* ***legal****.*

CMD SELECTION

- Ancestor-child relation between modeling domains.
  - Define $\Phi(M) = \{B(s) \mid s \quad C_Q(P)\} \in$ set of all agent behavior.
  - $M_1 = C_{Q_1}(P_1)$ is the ancestor of $M_2 = C_{Q_2}(P_2)$ iff $\Phi(M_2) \quad \Phi(M_1)$. $\subseteq$
- Search CMDs on modeling domain relation graph (directed edges representing ancestor-child relation).



CMD SELECTION CONTD.

- Two design aspects when selecting CMD $C = C_Q(P)$
  - Semantics – decided by semantic domain *Q*
    - Expressiveness vs. analyzability, e.g. dataflow vs. static dataflow.
    - May first choose semantic domain for common ancestor domain *D*, then refine it in *C.*
  - Abstraction level – depends on primitives *P*
    - Explore different abstraction level by choosing different primitives.
    - Carried out when selecting *C* as child domain of *D.*
  - For both, it is a trade-off between the size of mapping space and complexity.

COVERING PROBLEM AFTER CMD SELECTION

- Symbols:
  - Function primitive instances : $F = (f_1, f_2, \ldots, f_n)$
  - Architecture primitive instances : $A = (a_1, a_2, \ldots, a_m)$
  - Mapping decision variables : d_{f_i,a_j}
  - Architecture selection variables: s_{a_j}
  - Quantities (power, area, bandwidth…): $Q_{f_i,a_j,t}$ $Q_{a_j,t}$
- General covering formulation

Function covering constraints

$$\forall f_i \in \mathcal{F}, \quad \sum_{a_j \in \mathcal{A}_{f_i}} d_{f_i,a_j} = 1$$

$$\forall f_i \in \mathcal{F}, a_j \notin \mathcal{A}_{f_i}, \quad d_{f_i,a_j} = 0$$

Architecture selection constraints

$$\forall a_j \in \mathcal{A}, \quad \sum_{f_i \in \mathcal{F}} d_{f_i,a_j} \geq s_{a_j}$$

$$\forall f_i \in \mathcal{F}, a_j \in \mathcal{A}, \quad d_{f_i,a_j} \leq s_{a_j}$$

Domain specific. Determines complexity!

Quantity constraints

$$H_{t,l}(\{d_{f_i,a_j}\}, \{Q_{f_i,a_j,t}\}, \{s_{a_j}\}, \{Q_{a_j,t}\}) \leq 0$$

Objective functions

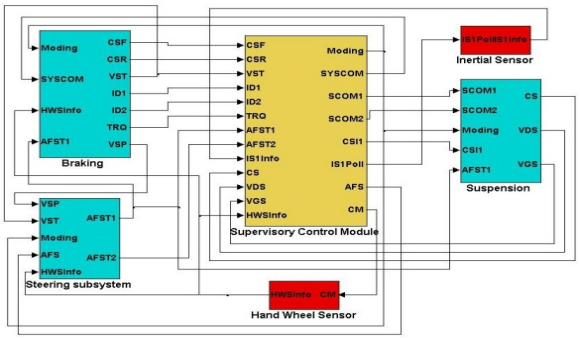
$$\min \; G_t(\{d_{f_i,a_j}\}, \{Q_{f_i,a_j,t}\}, \{s_{a_j}\}, \{Q_{a_j,t}\})$$

CASE STUDY: ACTIVE SAFETY VEHICLE

- Functional correctness and cost-efficiency are both important for active safety applications.
- Function and architecture mismatch.

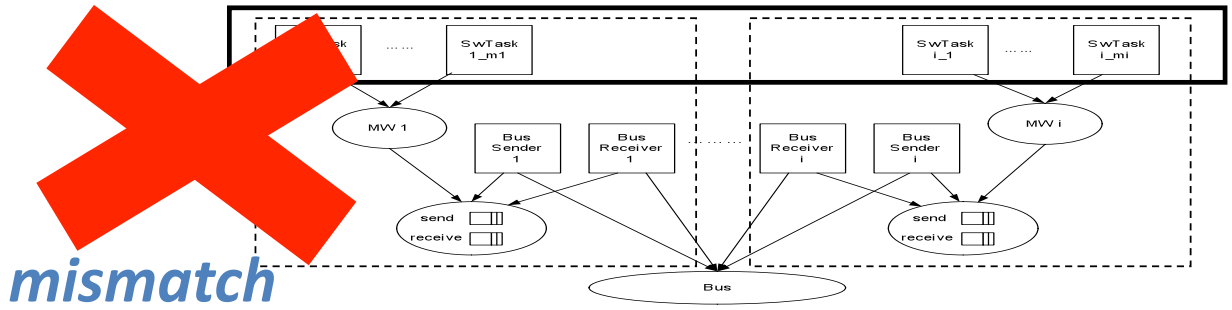
Function model

- synchronous model.
- no message loss or duplication.

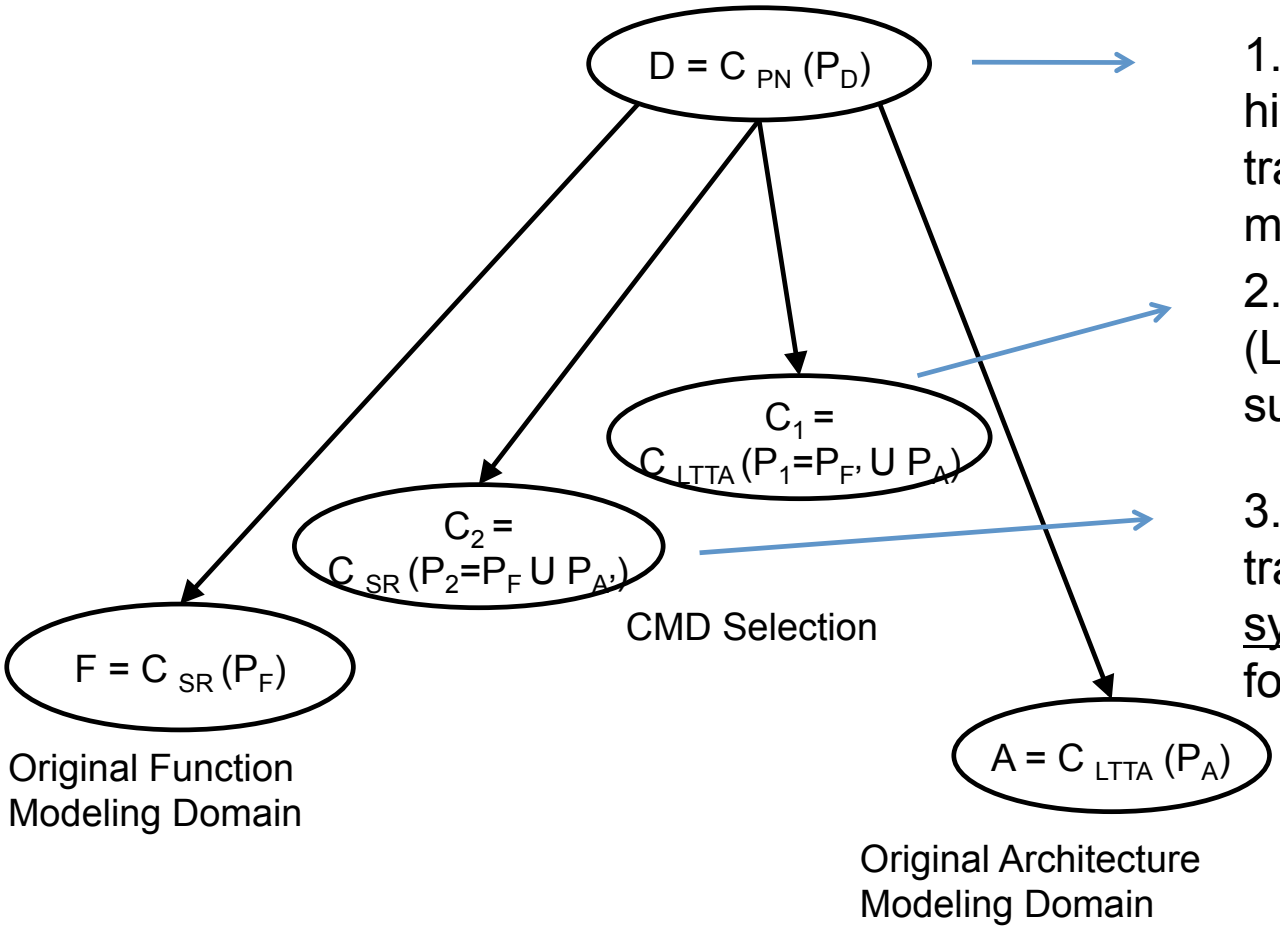


Architecture platform

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



STAGE 1: CMD SELECTION – COMMON SEMANTICS

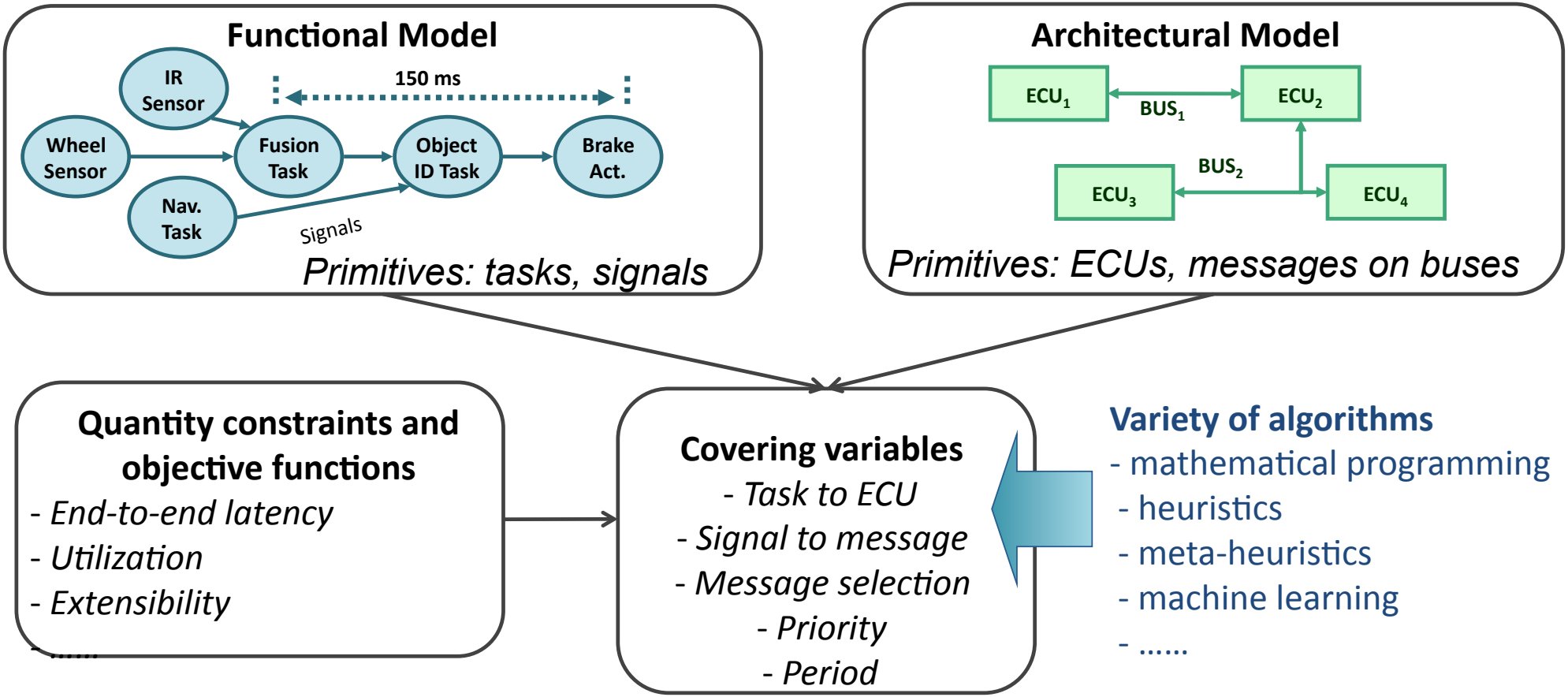


1. Process Networks (PN): expressive but high modeling complexity. Need transformation of both func and arch models.
2. Loosely time triggered architecture (LTTA): transformation of func model to support asynchronous communication.

Chosen in this case study

3. Synchronous reactive (SR): transformation of the arch to support synchronous communication, by applying following protocols.
   - Clock synchronization.
   - Constraints on task periods.

STAGE 2: COVERING PROBLEM



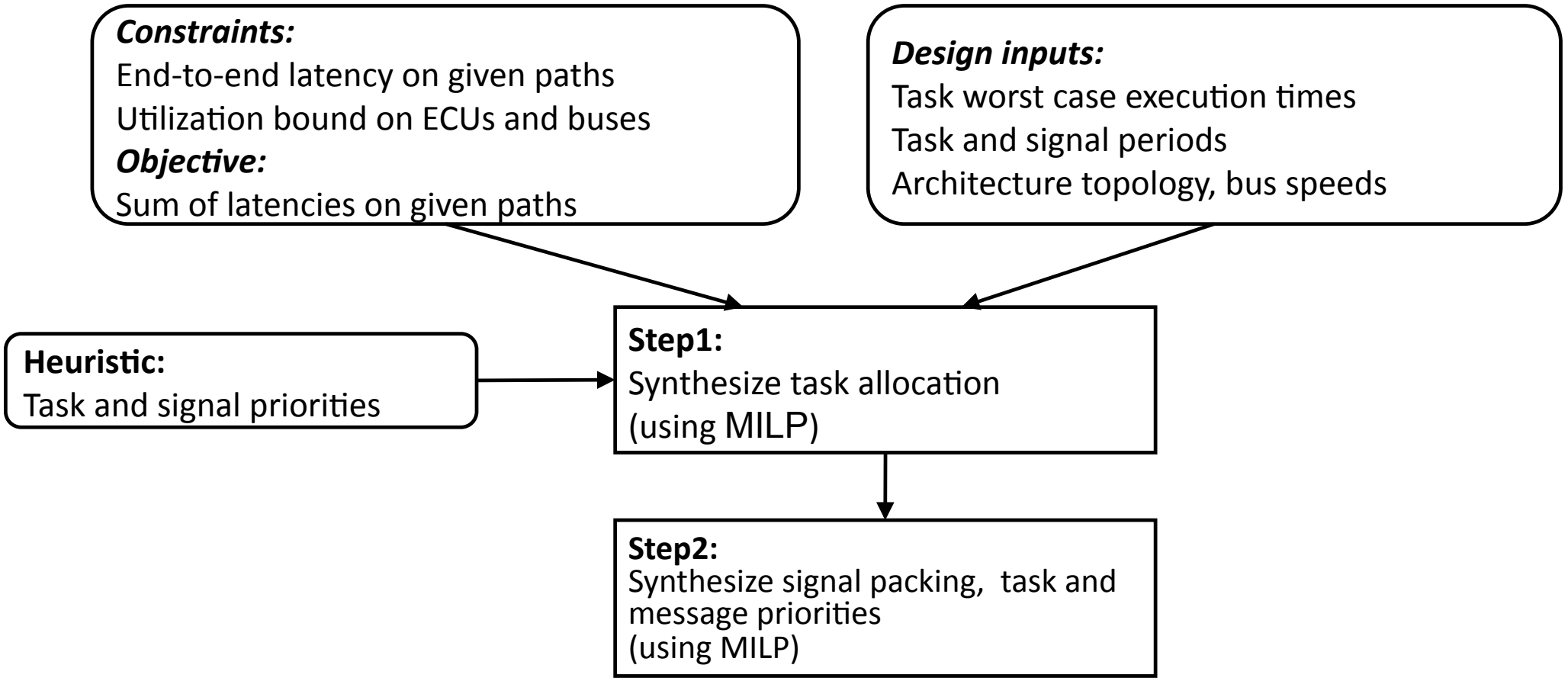
STAGE 2: COVERING PROBLEM CONTD.

- Worst case analysis for CAN systems with periodic tasks and messages.
- A complete formulation with all design variables does not scale for industrial size problems.
- We start with tackling following sub-problems.

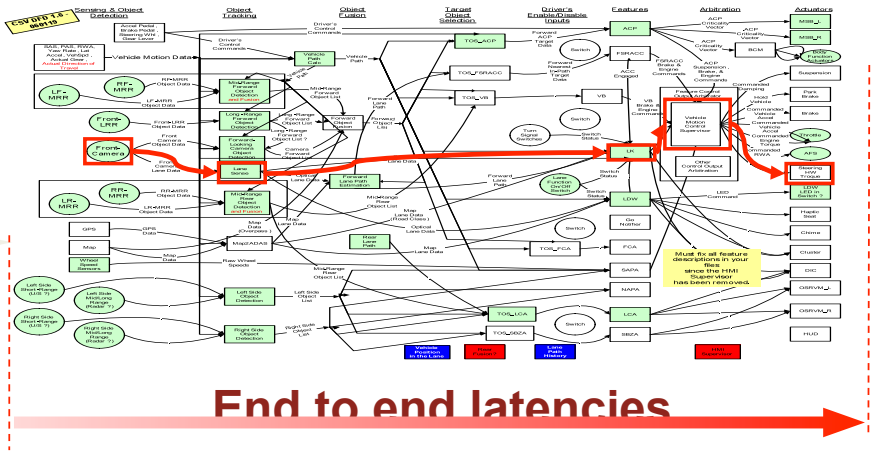
Problems	Period Synthesis [1]	Allocation & Priority Synthesis [2]	Extensibility Optimization [3, 4]
Variables	Period	Allocation Priority	Allocation Priority
Objective	Latency	Latency	Extensibility
Approach	Geometric programming (GP)	Mixed integer linear programming (MILP)	Multi-step Heuristic

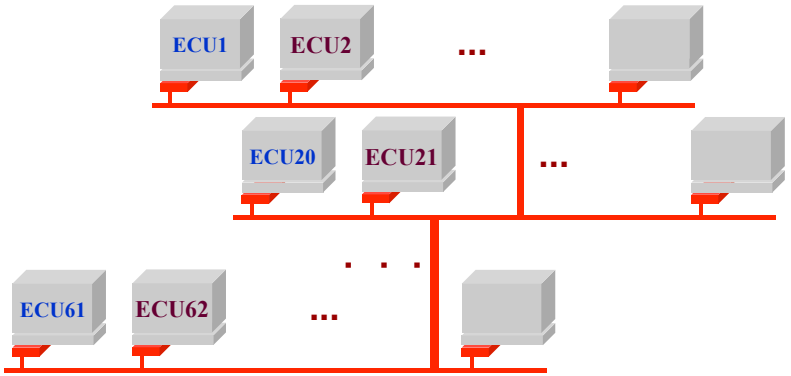
[1] “Period Optimization for Hard Real-time Distributed Automotive Systems”, 44th DAC, 2007.
[2] “Definition of Task Allocation and Priority Assignment in Hard Real-Time Distributed Systems”, 28th RTSS, 2007.
[3] “Optimizing Extensibility in Hard Real-time Distributed Systems”, 15th RTAS, 2009.
[4] “Optimizing the Software Architecture for Extensibility in Hard Real-Time Distributed Systems”, TII, 2010.

ALLOCATION & PRIORITY SYNTHESIS (MILP BASED)



ALLOCATION & PRIORITY SYNTHESIS RESULTS





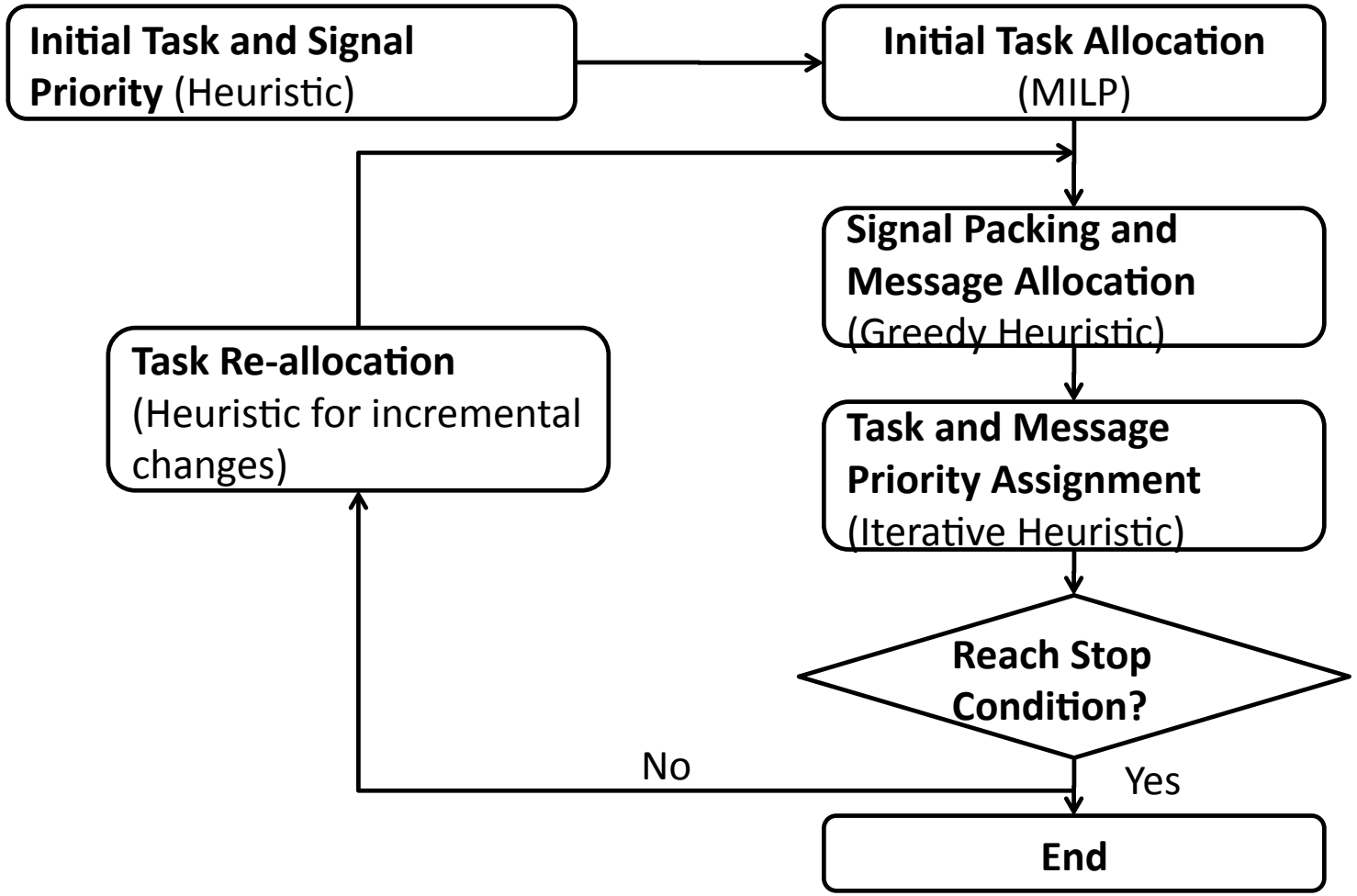
Function Model
- 41 Tasks
- 83 Signals
- 171 paths

Mapping

After mapping
- ***Meet all requirements***
- ***Total latency from 36486ms in manual design to 12900ms***

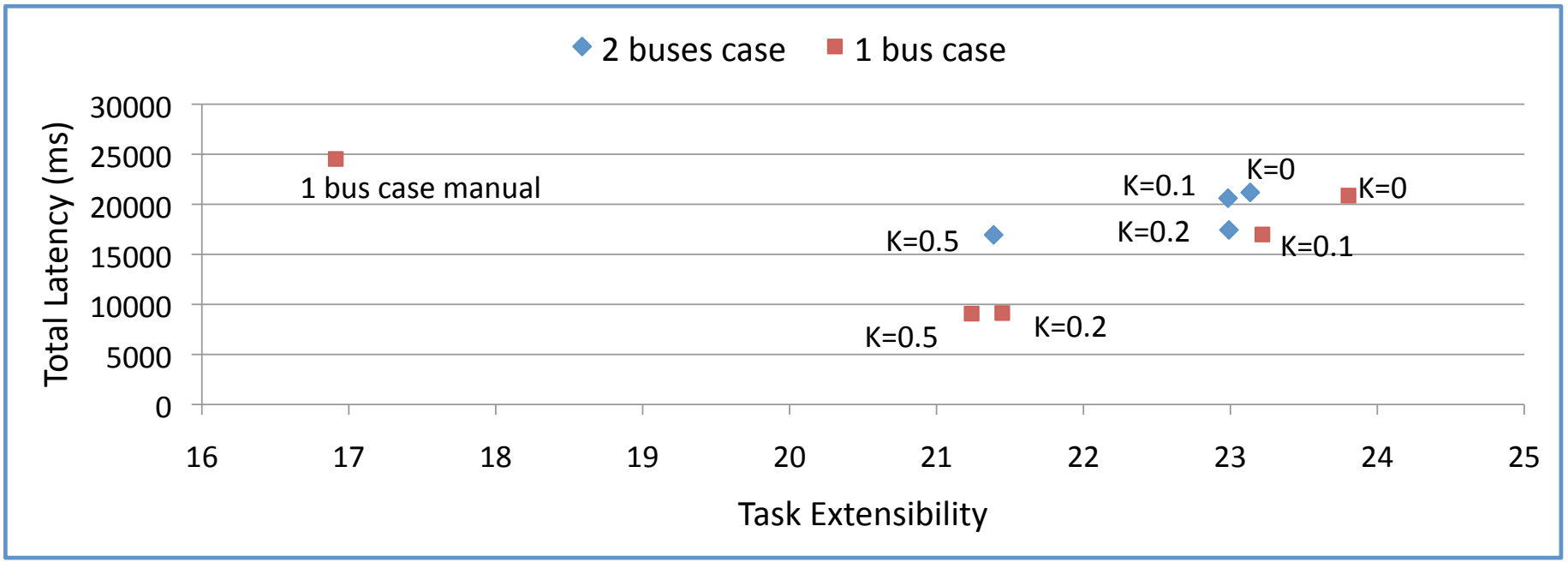
Architecture platform
- 9 ECUs
- single bus

EXTENSIBILITY OPTIMIZATION (MILP AND HEURISTIC)



EXTENSIBILITY OPTIMIZATION RESULTS

- Same active safety vehicle as in allocation and priority synthesis.
- Single-bus and dual-bus options.
- Parameter *K* to trade off between extensibility and latency.
- Compared with a simulated annealing algorithm: maximum extensibility within 0.3%, runtime 0.5 hour vs. 12 hours.



CASE STUDIES IN OTHER DOMAINS

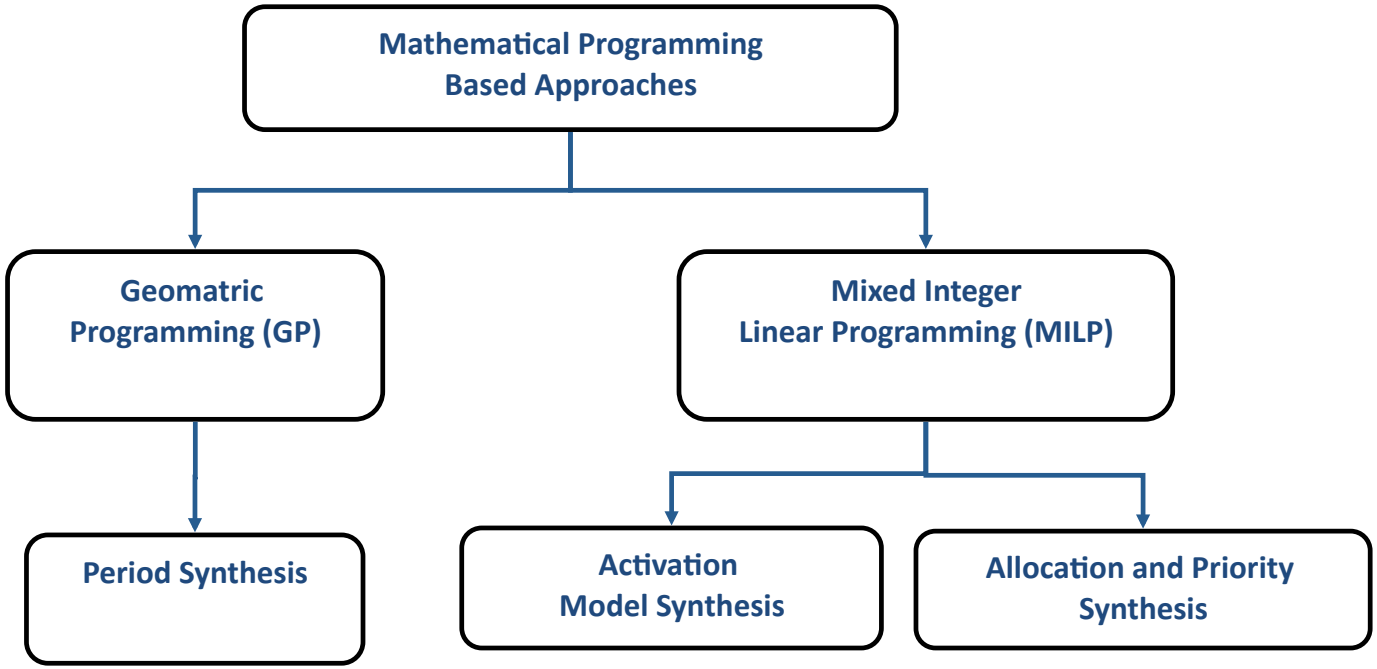
- Building automation domain [1]
  - Similar semantics as in automotive – synchronous function model and LTTA architecture platform.
  - Also choose SR as the common semantics, however additional timing constraints are added to the architecture for preserving synchronism, as we consider the physical interaction with environment.
  - Mapping leverages COSI for communication network synthesis.
- Multimedia domain [2]
  - JPEG encoder application. Intel MXP architecture platform.
  - Semantics for both function and architecture are dataflow.
  - Challenge is to choose the proper abstraction level. Different levels are explored and compared through choices of primitives.

[1] "A Design Flow for Building Automation and Control Systems", 31st RTSS, 2010.
[2] "JPEG Encoding on the Intel MXP5800: A Platform-Based Design Case Study", ESTIMedia'05, 2005.

Mathematical Programming Approaches

Extensibility to add additional constraints for system-specific situations

Design Space
- Allocation
- Priorities
- Periods
- Activation Model



2009 IEEE Trans on Industrial Informatics (best Paper); 3 Invited Proc of the IEEE papers

CONCLUDING REMARKS

- Software (and hardware) synthesis based on a formal mapping procedure
  - Formally determines the semantics and abstraction level of the design by choosing a common modeling domain.
  - Automatic and optimal mapping algorithms.
  - *Generality* – applied to various domains with different models of computation as well as different implementation platforms. Domain-specific mapping algorithms may be leveraged in the framework.
  - *Optimality* – trade-off between complexity and mapping space through the selection of CMD.
  - *Reusability* – common semantic selection requires designers' expertise. However proper selection is typically general for particular domains.

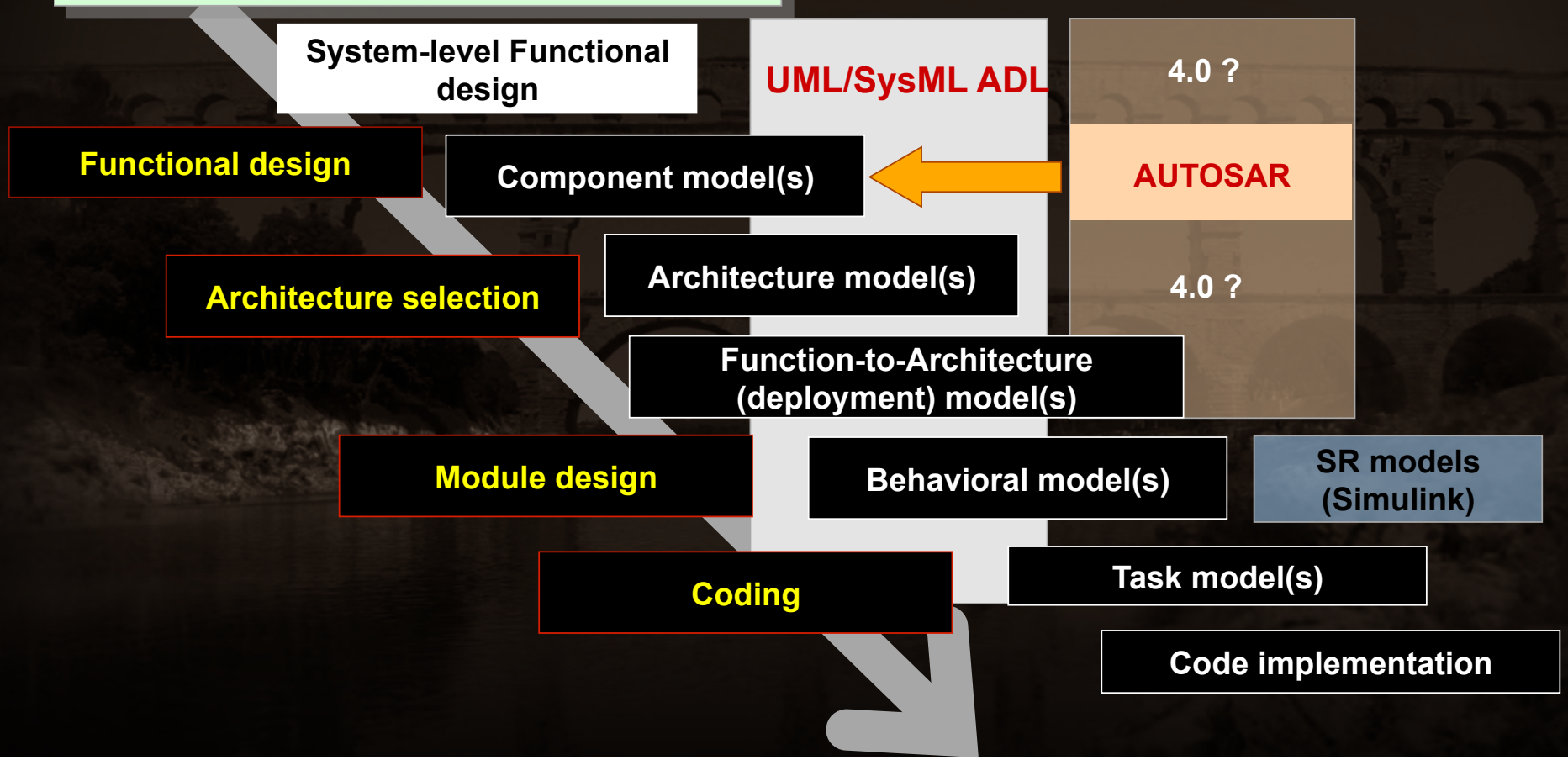
Magneti Marelli Results

Model and Platform based design methodology successfully implemented in Magneti Marelli Powertrain leading to:

- **Significant increase in application software productivity** up to 4 time faster than in the traditional hand-coding cycle.
- **The model compiler has been applied only to models mapped in the application software partition with a 90% coverage.**
- **Virtually bug-free application software 100% compliant to executable specifications.** No need for unitary verification tests, already performed in a simulated environment and components already quoted in terms of CPU load with the PIL environment.
- **Automatically generated software components** created for GDI project have been re-used by other projects (MPI, Diesel) currently in production.

Heterogeneous models

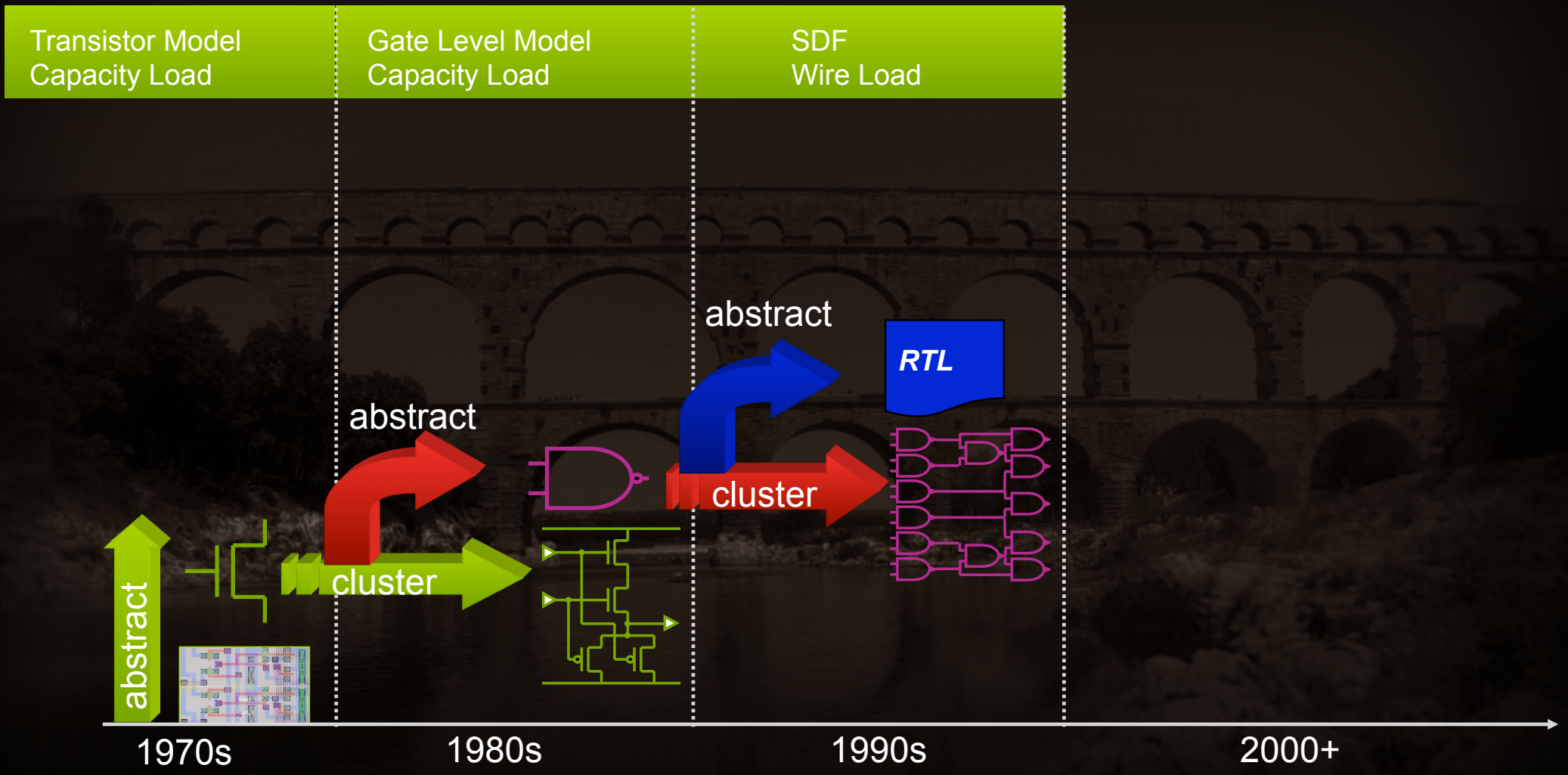
+ *separation between the functional model and the architecture model*



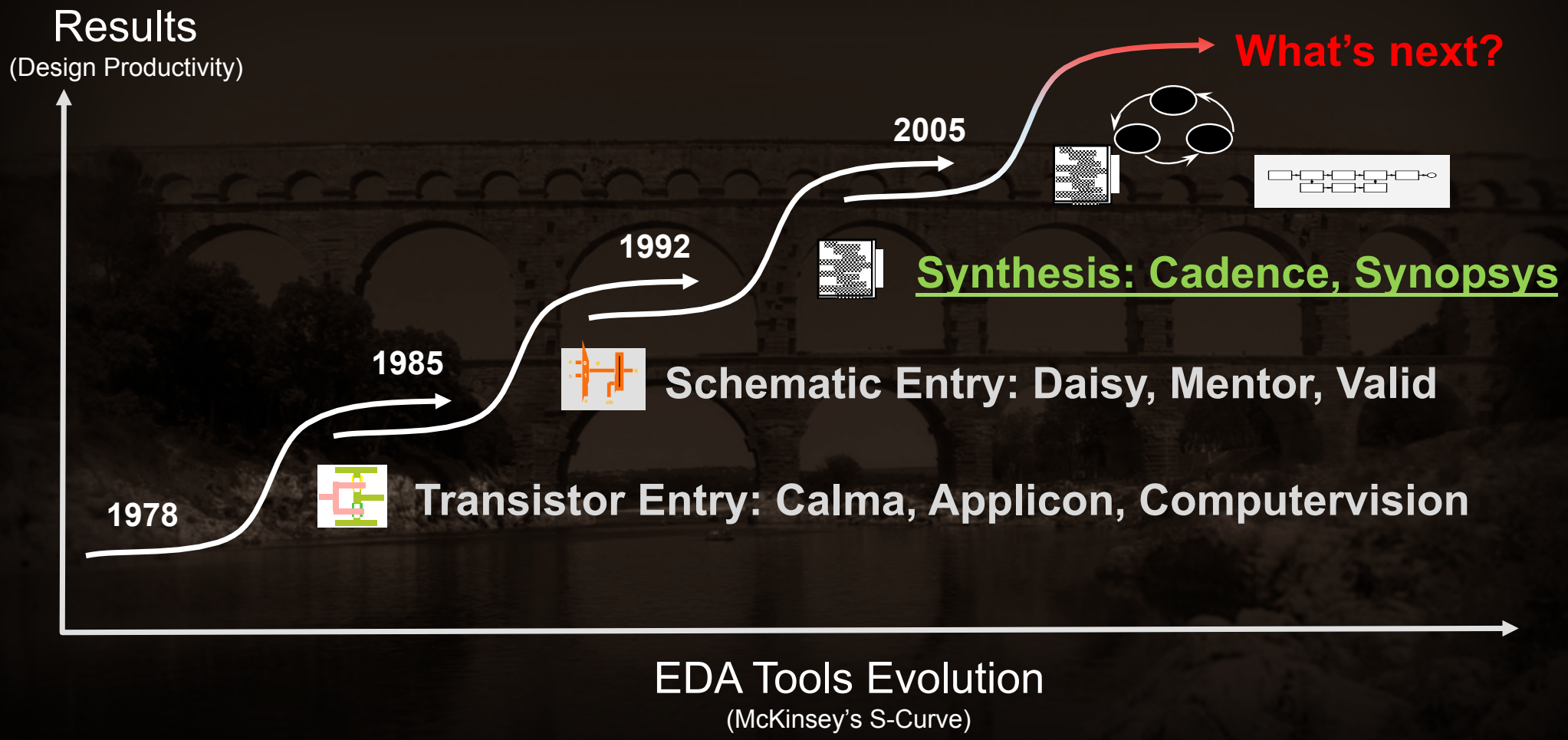
Outline

- Introduction and Motivation using Automotive as Test Case
- The V design process and Platform Based Design
- The Role of Autosar
- Semiconductor Design Economics
- Extensions and Open Issues

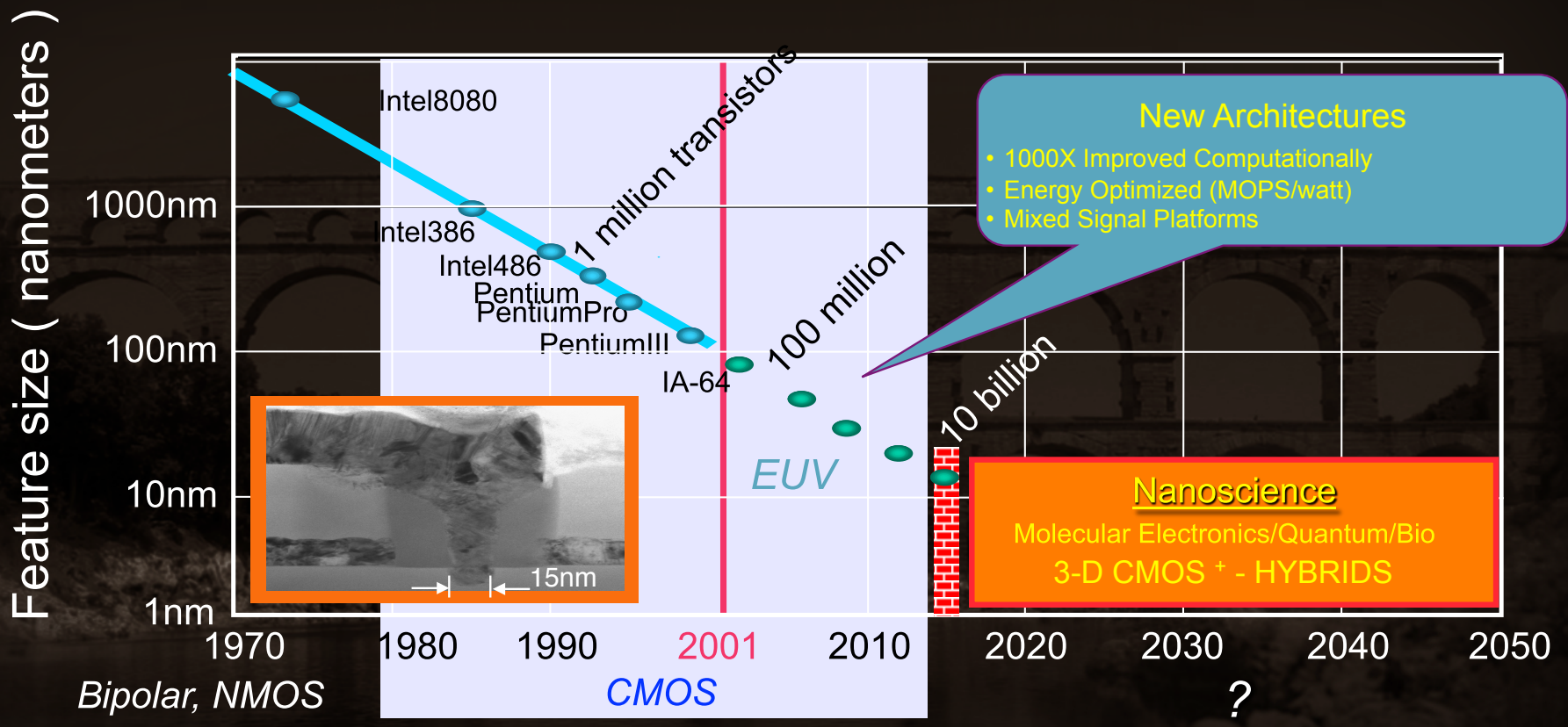
The Story of EDA: The Quest for the Next Level of Abstraction



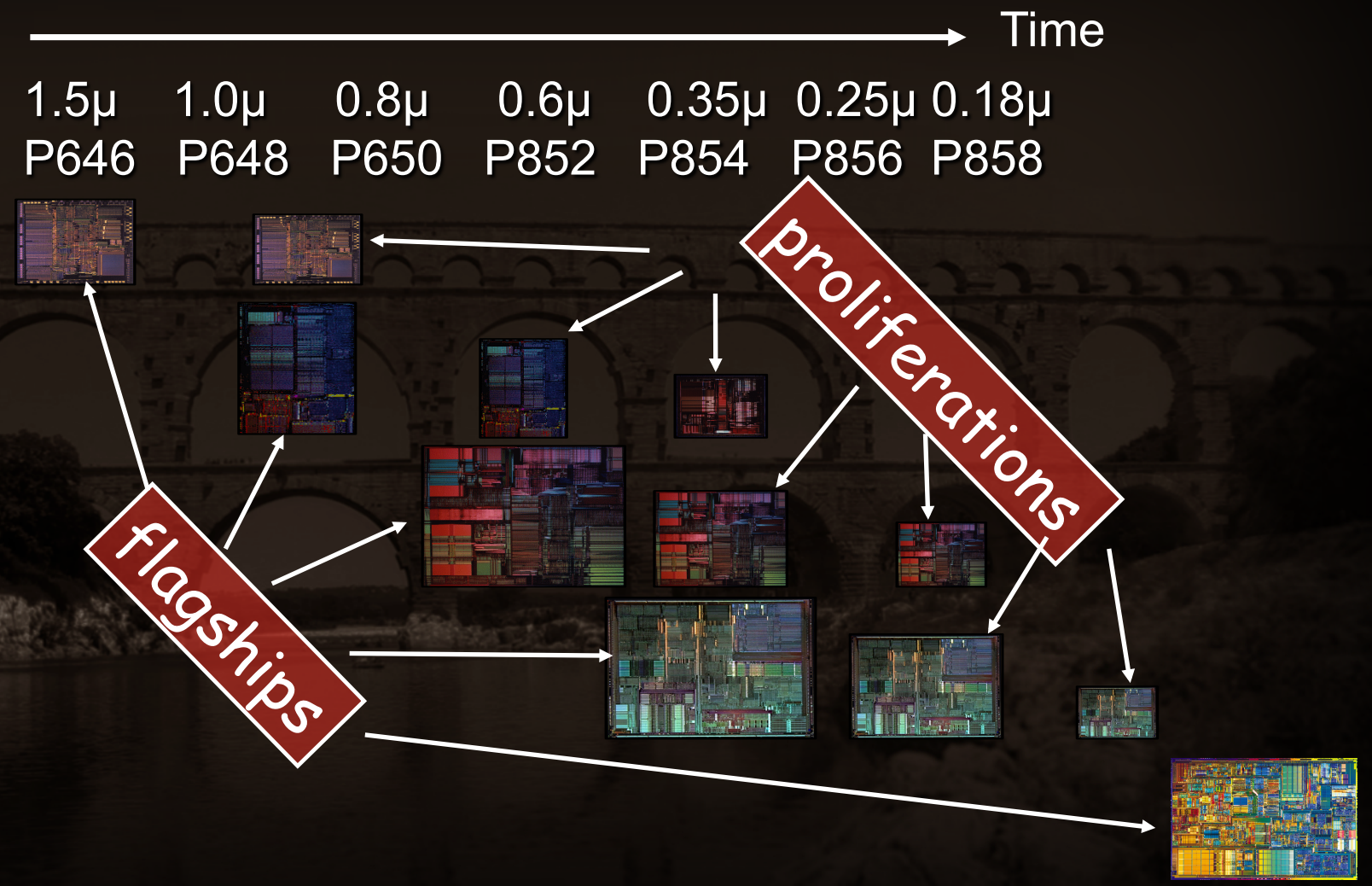
Evolution of the EDA Industry



Moore's Law


Feature size (nanometers)
1000nm
100nm
10nm
1nm
1970
1980
1990
2001
2010
2020
2030
2040
2050
Bipolar, NMOS
CMOS
?
Intel8080
Intel386
Intel486
Pentium
PentiumPro
PentiumIII
IA-64
1 million transistors
100 million
10 billion
EUV
15nm
New Architectures
• 1000X Improved Computationally
• Energy Optimized (MOPS/watt)
• Mixed Signal Platforms
Nanoscience
Molecular Electronics/Quantum/Bio
3-D CMOS + - HYBRIDS

The Magic of Moore's Law

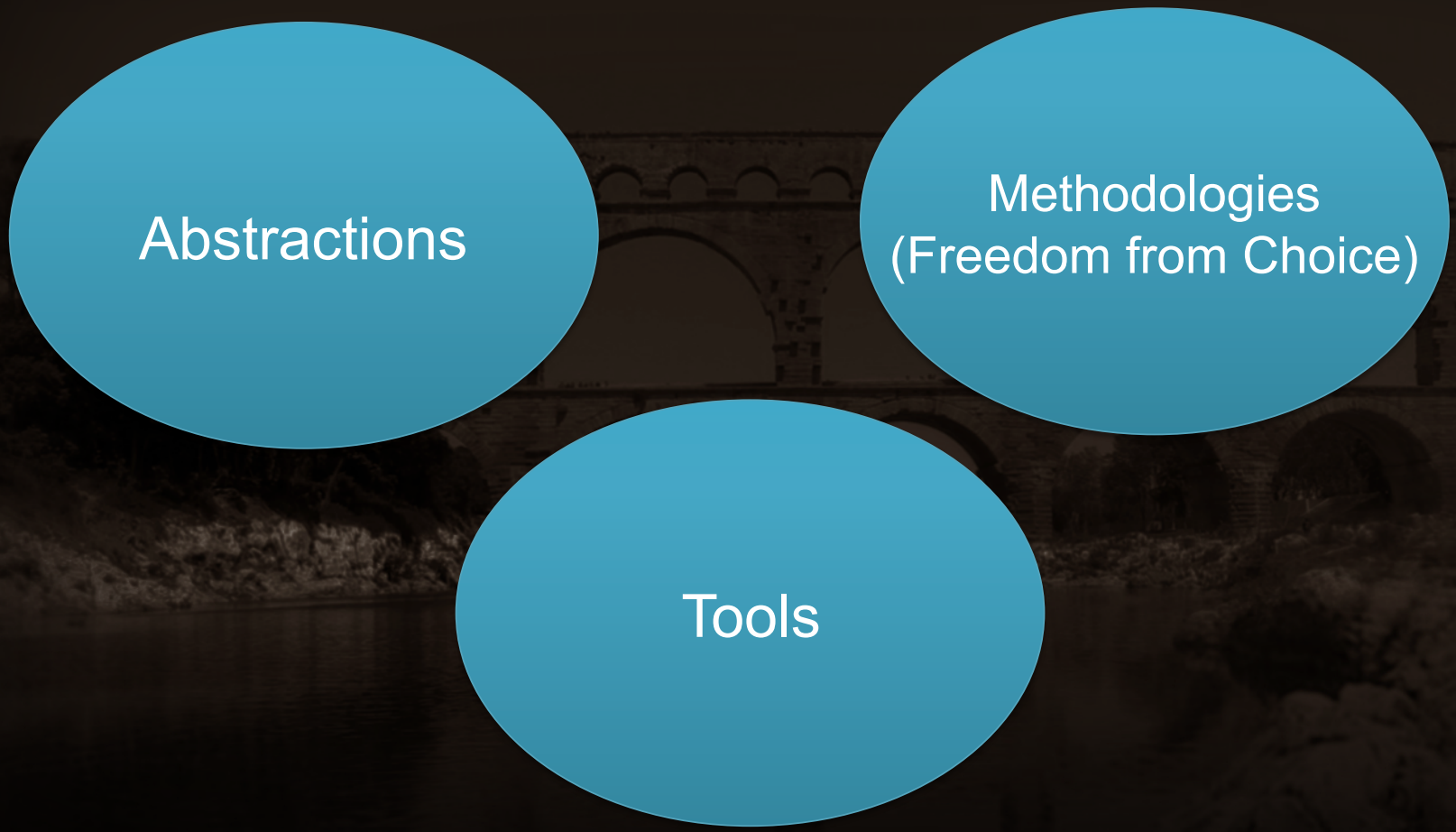


Evolution of Digital Design Productivity

Year	DT Productivity Impact	Gates Per Designer per Year	Description
1990	Design Technology	4,000	
1993	In House Place and Route	5,550	Automated Block Placement and Routing
1995	Tall Thin Engineer	9,090	Engineer can purse all tasks to complete a design block from RTL to GDSII
1997	Small (2K-75K) Block Reuse	40,000	Blocks from 2,500 –74,999 gates
1999	Large (75K-1M) Block Reuse	56,000	Blocks from 75,000-1M gates
2001	IC Implementation Suite	91,000	Tightly integrated toolset that goes from RTL synthesis to GDSIII through IC palace and route
2003	Intelligent Test Bench	125,000	RTL verification tool (cockpit) that takes and ES-level description and partitions it into verifiable blocks, then executes verification tools on the blocks, while tracking and reporting code coverage
2005	ES Level Methodology	200,000	Level above RTL, including both HW and SW design. It consists of a behavioral (where the system function has not been partitioned) and an architectural level (where HW and SW are identified and handed off to design teams).
2007	Very Large (>1M) Block Reuse	600,000	Blocks >1M gates; intellectual-property cores

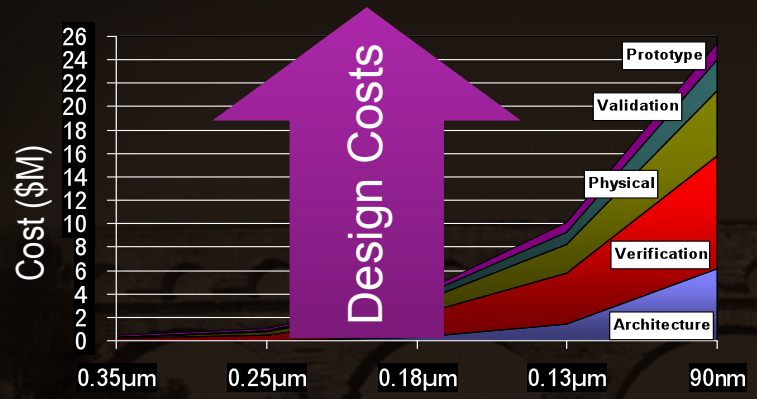
Source: J. Weekly, Synopsys

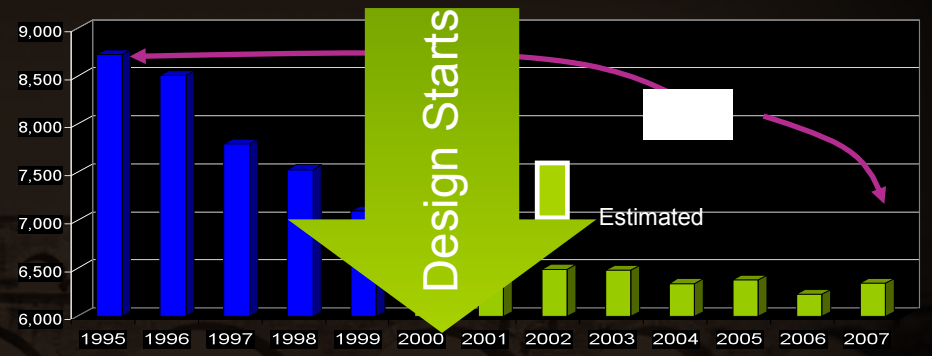
How did we cope?

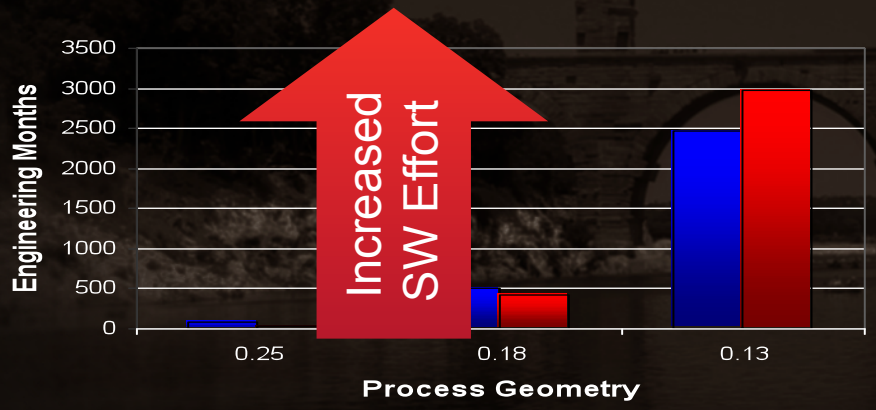


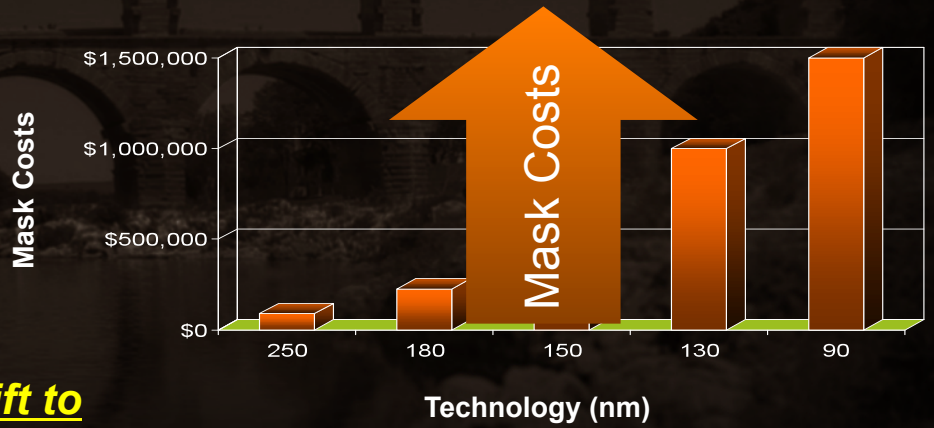
The Full Day: the Maturity of EDA

Challenges and Trends









Cell-Based ASICs prohibitively expensive for all but highest volume applications

Shift to

- Re-use strategy at all levels
- Higher level of abstractions
- Software !!!

Definitions

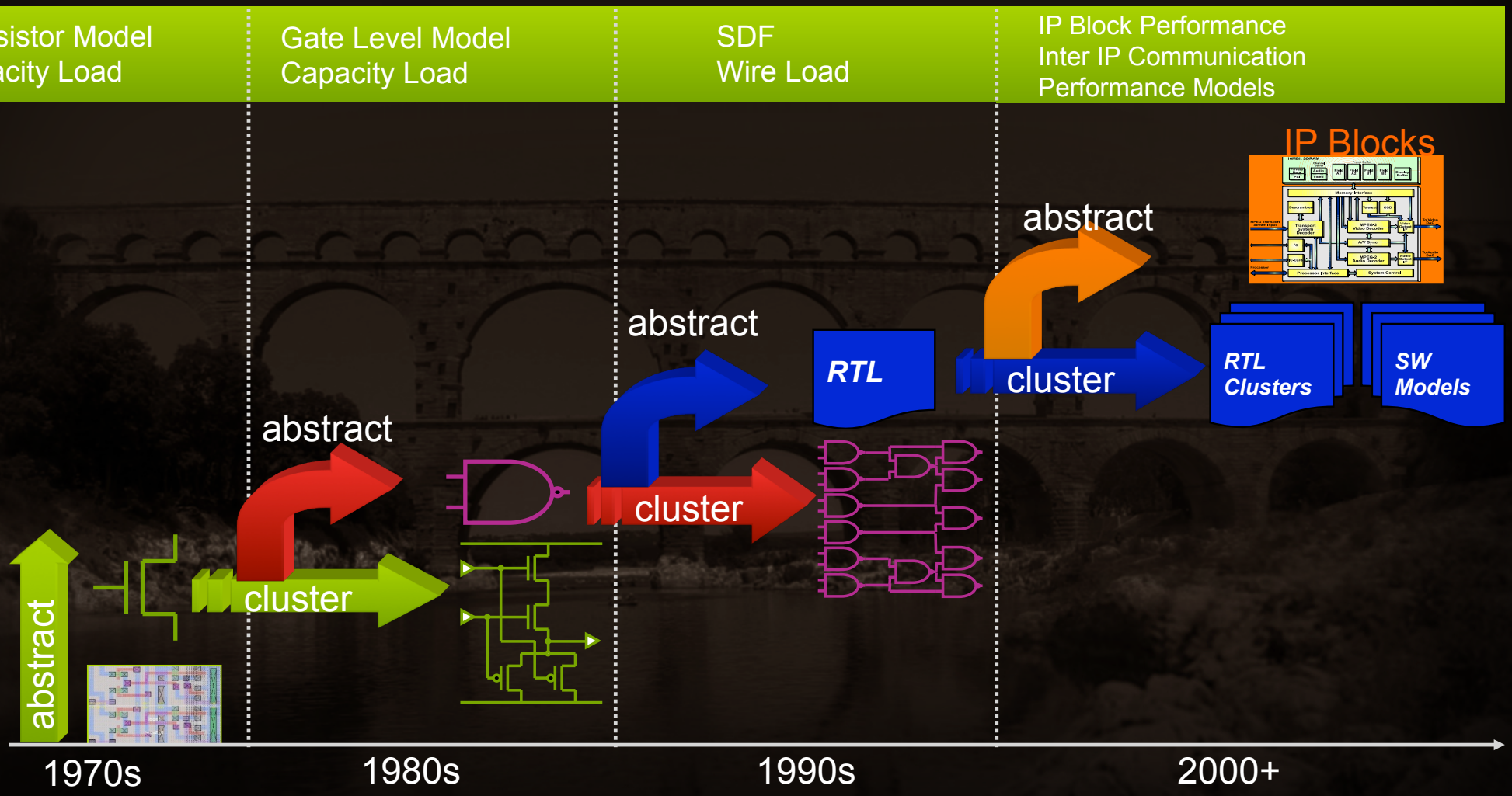
- ASPP: Application Specific Programmable Platform
  - Instruction set and architecture are customized to application
  - Application customers help providing specs

- PSPP: Platform Specific Programmable Product
  - ASPPs: End-user/designers will only program the part
  - PSPPs: End-user/designers will partially configure/design the hardware then program the part
  - Will there be more PSPPs than ASPPs in 2010?

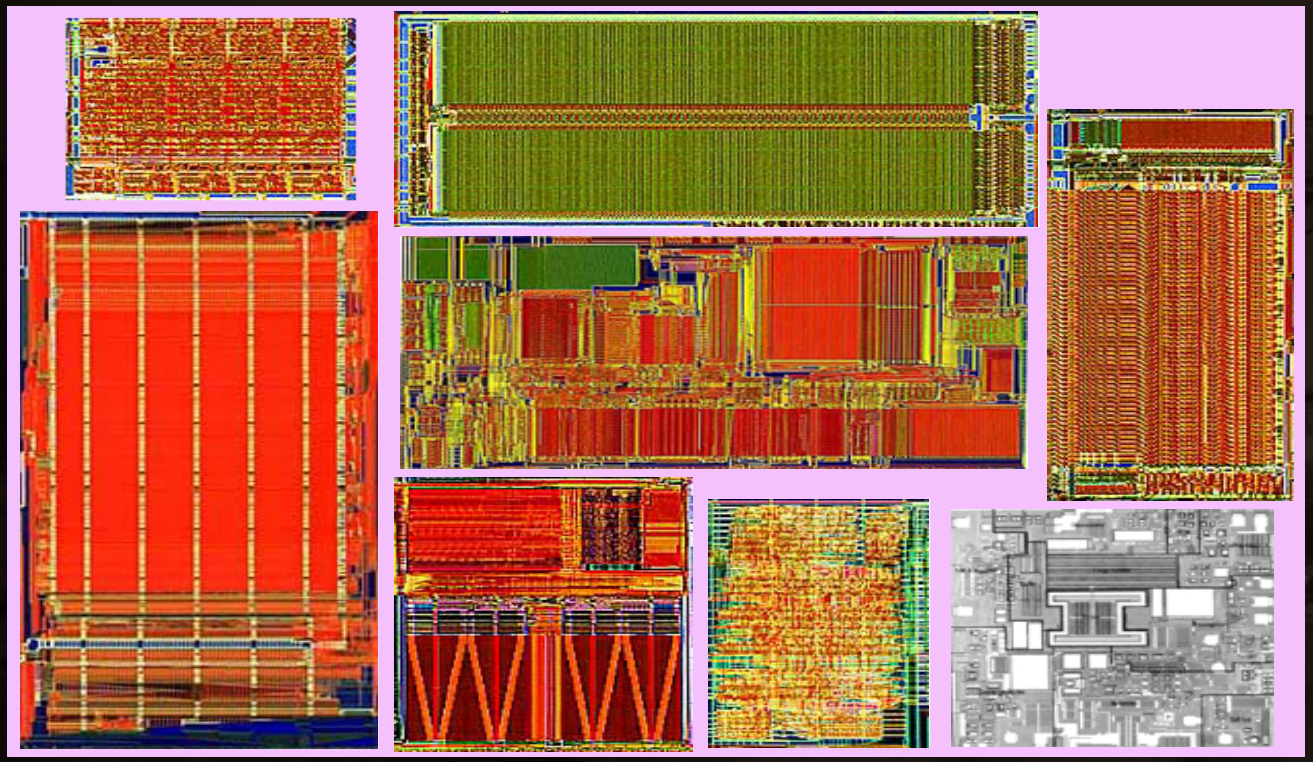
Questions

- What applications/businesses will be building ASICs instead of PSPPs and ASPPs?
- Bleeding edge performance will be less of a factor for a larger percentage of applications moving forward: audio, video?
- Where will cost be less of a factor than super-optimized performance?
- Will an optimized PSPP be more timely, cost-friendly, and even higher in performance given that it has been optimized for speed for a particular application area/platform?
- Who can afford to design a completely non-reconfigurable, non-programmable ASIC?

The Story of EDA: The Quest for the Next Level of Abstraction



The Design Object



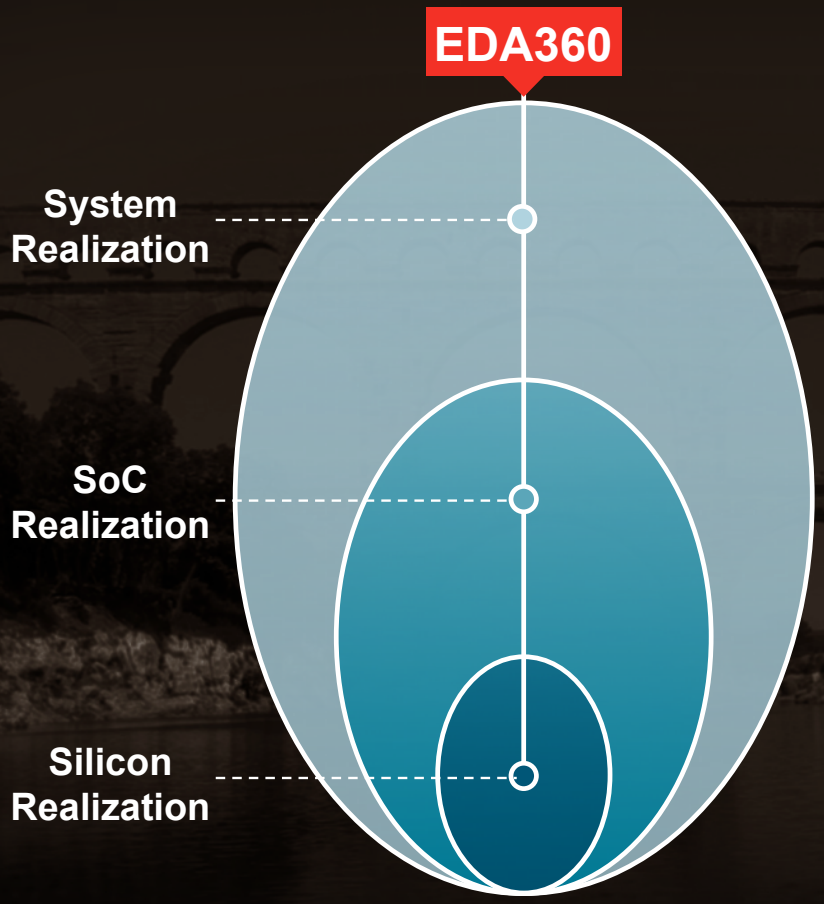
- **Assemble Components from parameterized library**
- **Including:**
  - Configurable processor core
  - Memories (RAM, ROM)
  - Special-purpose standard blocks (ASSPs)
  - Glue Logic
- **Third-party special-purpose logic/MEMS/MEOS**
- **Integrate using standard approach to on-chip communication**

Developing a New ASIC

VERY EXPENSIVE PROPOSITION

- Core IP (microprocessor) may cost more than 4 Billion (see NYTimes article on AP4, the new core by Apple)
- NRE are from 8 to 10 Million each chip
- Trend is to leverage maximally platforms: few cutomized parts
- More important to team with right IC maker and select the right architecture and design tools to support the application

Cadence EDA360 Vision

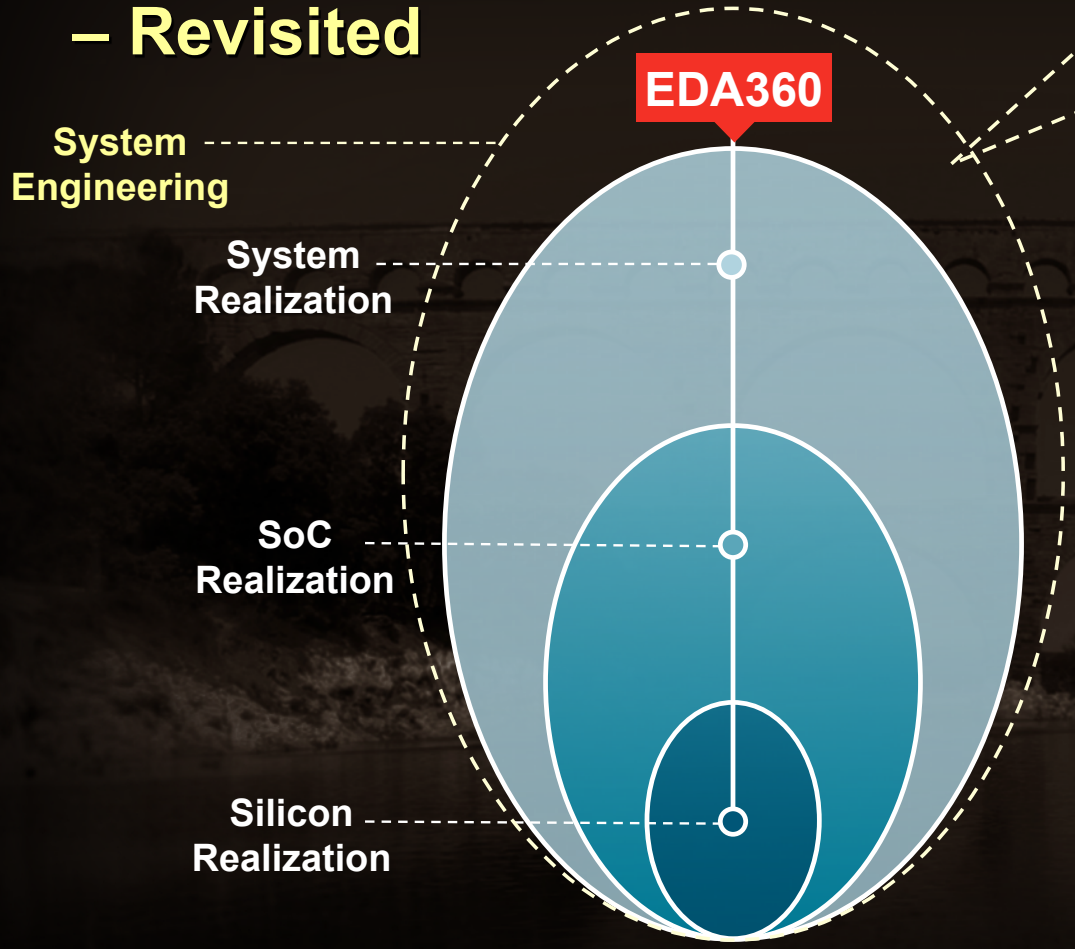


- Architecture, design, integration, and verification of complex electronic systems
- Includes hardware-software design and verification, system modeling, verification IP, and services
- Delivered via **open**, **connected**, and **scalable** offerings

- Design IP and Services with technologies for architectural exploration, integration, and verification of complex SoCs
- Includes memory and storage subsystems, interfaces, and chip planning capabilities
- Delivered via **differentiated**, **integrated**, and **proven** offerings

- Design, verification, and implementation of complex designs across silicon, package and board
- Includes comprehensive flows for low power, mixed signal, verification, large scale/GHz, and 3D-IC/SiP co-design
- Delivered via **intent**, **abstraction**, and **convergence** in flows

Cadence EDA360 Vision – Revisited



- **Formal Modeling of multi-physical systems and library definitions**
- **Includes requirement capture**
- **Cyber Physical System formal verification and virutalization**
- **Design Space Exploration**

- Architecture, design, integration, and verification of complex electronic systems
- Includes hardware-software design and verification, system modeling, verification IP, and services
- Delivered via **open**, **connected**, and **scalable** offerings

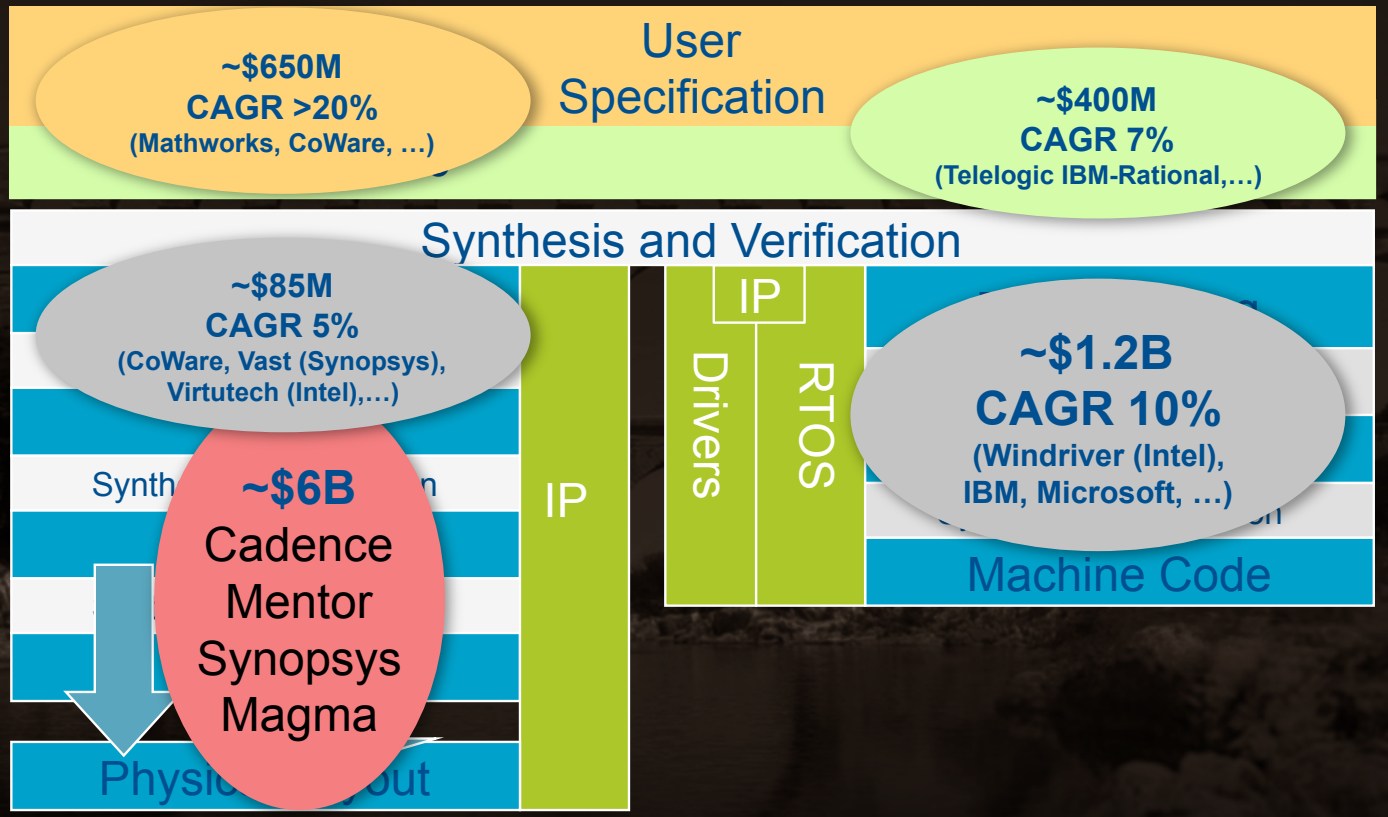
- Design IP and Services with technologies for architectural exploration, integration, and verification of complex SoCs
- Includes memory and storage subsystems, interfaces, and chip planning capabilities
- Delivered via **differentiated**, **integrated**, and **proven** offerings

- Design, verification, and implementation of complex designs across silicon, package and board
- Includes comprehensive flows for low power, mixed signal, verification, large scale/GHz, and 3D-IC/SiP co-design
- Delivered via **intent**, **abstraction**, and **convergence** in flows

Business Challenge: Tool Landscape

System Design Entry

Hardware Teams

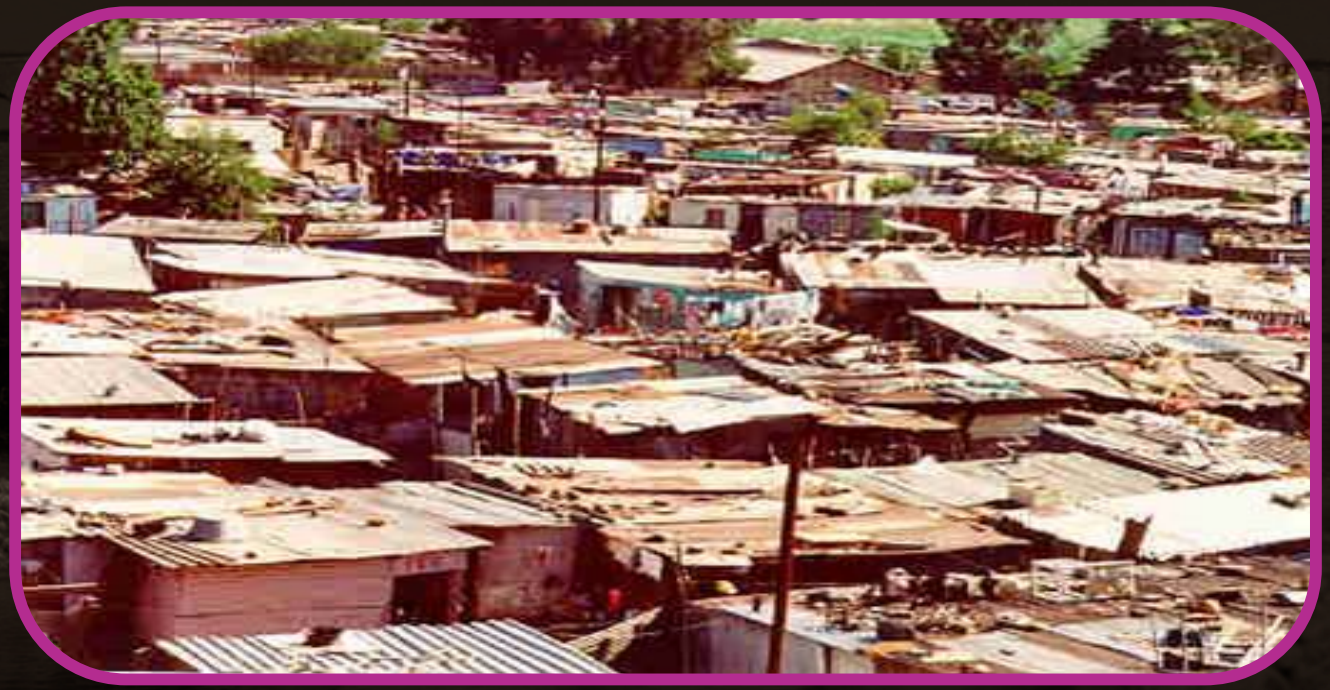


Software Entry

Embedded Software Teams

Final Words of Wisdom

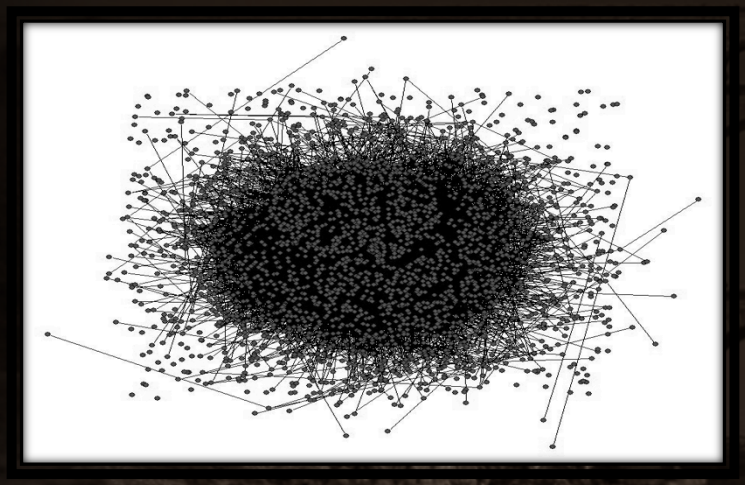
The EDA Challenge: Software Architecture Today



The Swarm Opportunity

It's A Connected World

Time to Abandon the "Component"-Oriented Vision



The functionality is in the swarm!

- There is power in numbers
- Resources can be dynamically provided based on availability

Moore's Law morphs into Metcalfe's Law:
Scaling is in number of connected devices, no longer in number of transistors/chip

[J. Rabaey, MuSyC 2009]

Supporting Theory

- Provide a semantic foundations for integrating different models of computation
  - Independent of the design language
- Maximize flexibility for using different levels of abstraction
  - For different parts of the design
  - At different stages of the design process
  - For different kinds of analysis
- Support many forms of abstraction
  - Model of computation (model of time, synchronization, etc.)
  - Scoping
  - Structure (hierarchy)

Platform Based Design!

Distributed Resources

- Communication (Spectrum)
- Computation
- Sensing Actuation
- Storage
- Energy

The Swarm Operating System -
Dynamically trading off resources

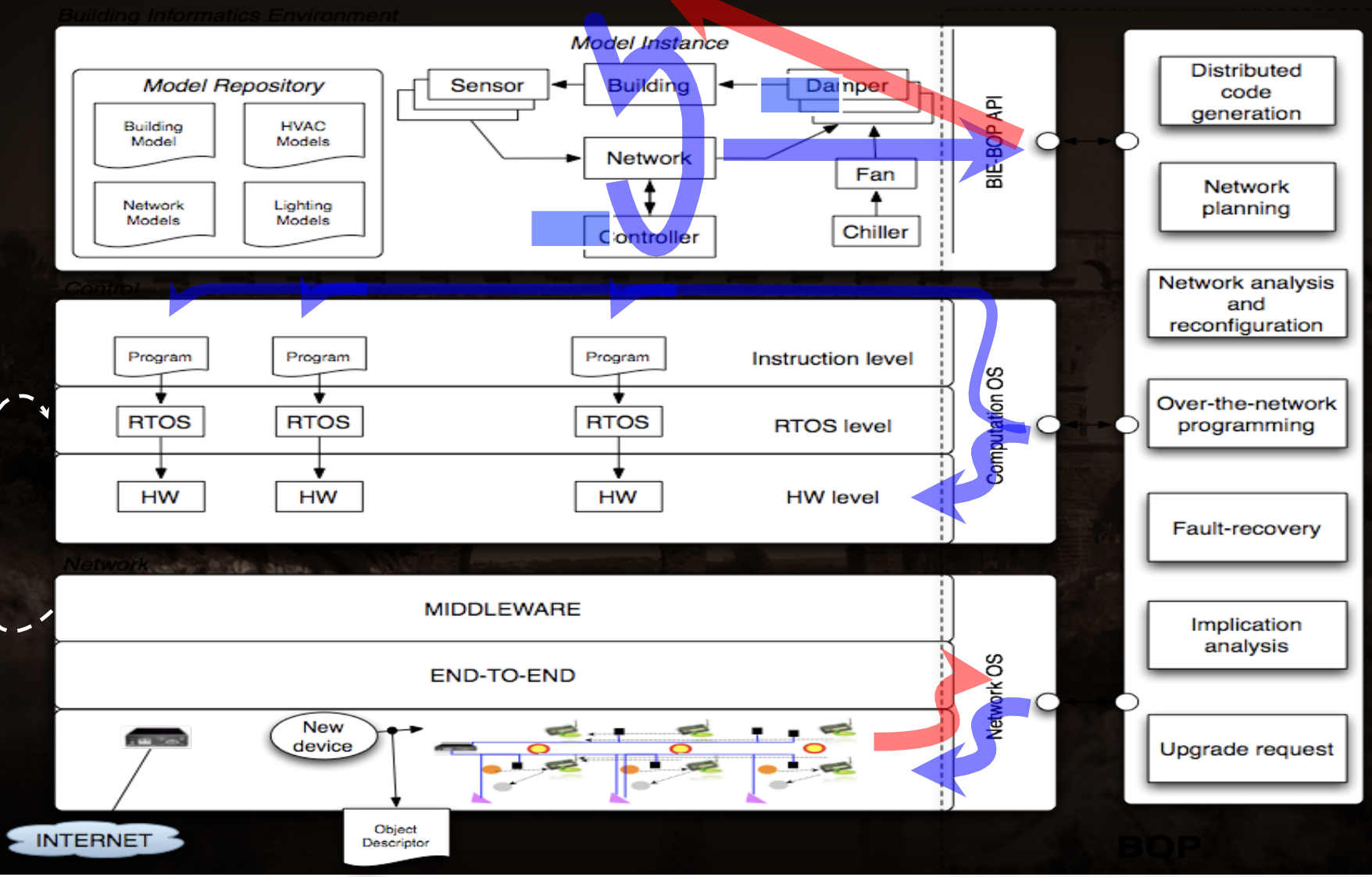
The "Unpad" Services and Applications

Utility Maximization

"What matters in the end is the utility delivered to the user"

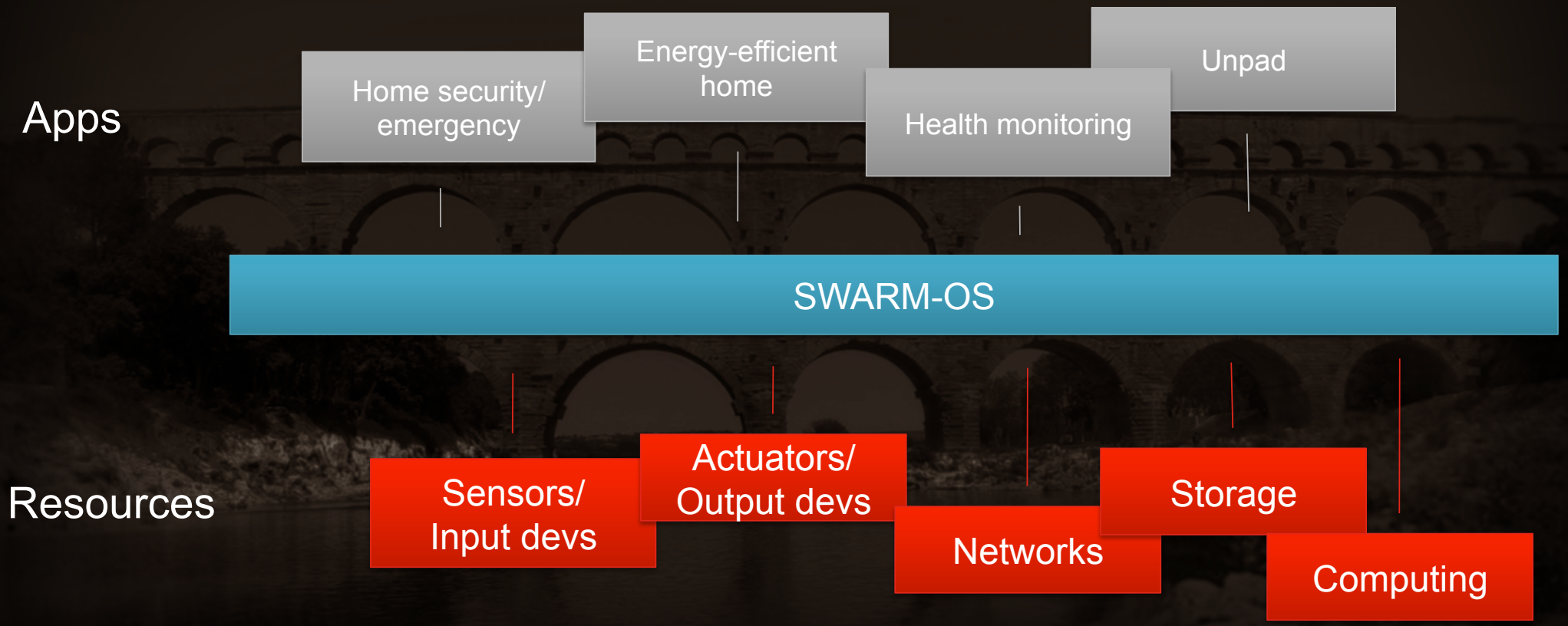
A continuously changing alignment
(environment, density, activity)

Building Operating Platform



The Swarm as a Platform

A mediation layer



Presenting a uniform API to Apps Developers (similar to trends in the Cloud)

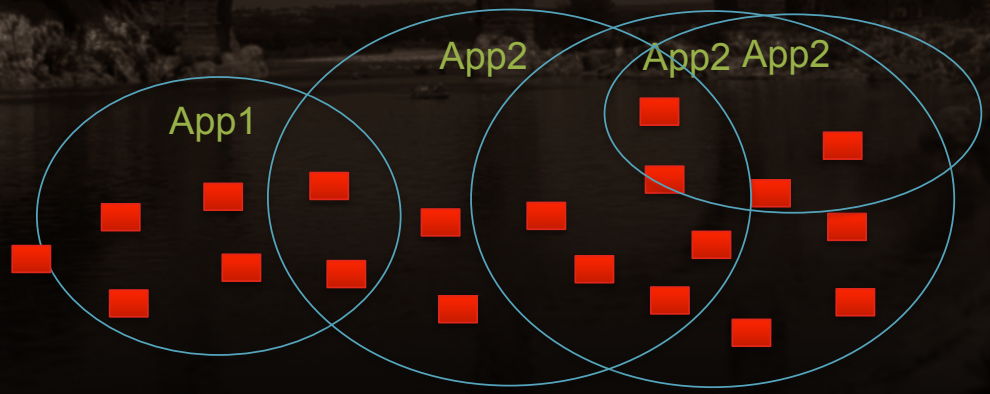
The Swarm as a Platform

Operating System (Broad Sense): Environment that

- Presents abstracted vision of hardware to applications
- Dynamically balances application needs versus available resources under time and energy constraints

What makes SWARM-OS different (and hard)?

- Distributed
- Space/context-aware
- Heterogeneous shared (and sparse) resources
- Dynamic
  - Mobility, scope, resources, connectivity, …



How to Deal with Dynamics
Structured versus ad-hoc?

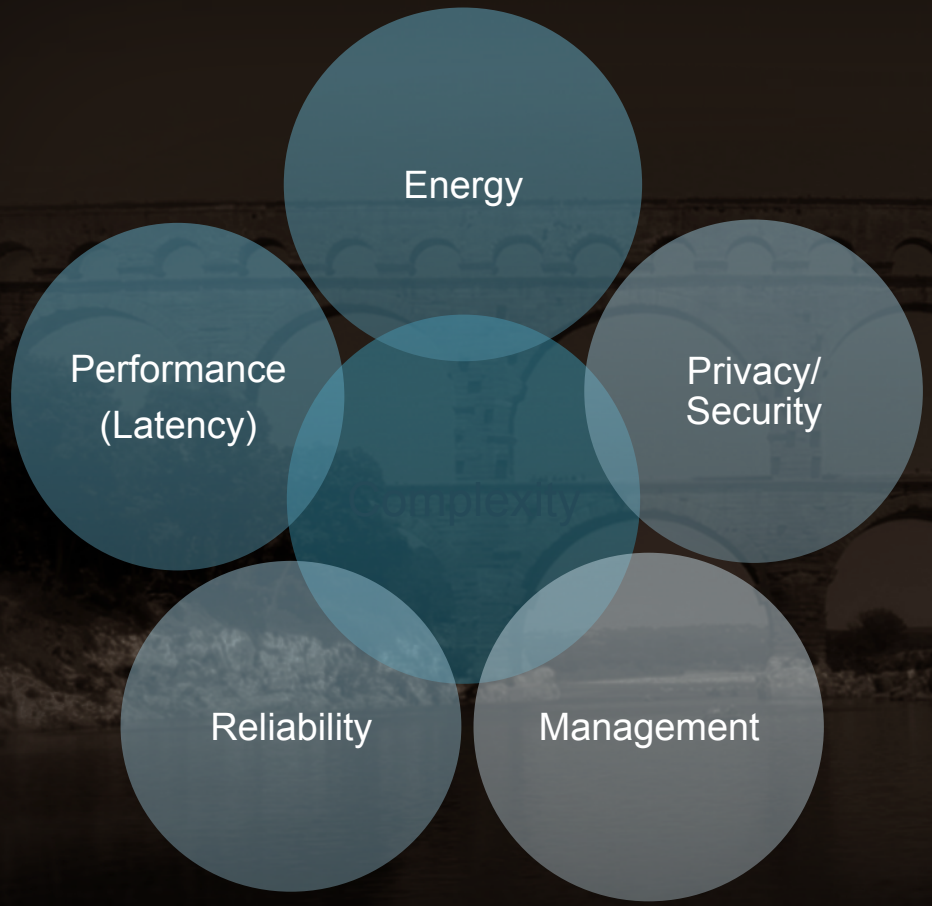
BOTH OF THE ABOVE!

Exploiting the Edge of the Cloud (or The Fog*)
Packs plenty of computation, communication, storage and energy resources
Avoids the overhead of the Cloud

Not an "OS as usual"
Reactive or opportunistic emergence of capabilities desirable

[F. Bonomi, Cisco, "Cloud and Fog Computing"– EON June 11]

The Swarm Challenge(s)



Complex distributed control systems combining heterogeneous components under dynamically varying conditions

The SCIENCE-Application Dilemma

Raffaello Sanzio, The Athens School

Concluding Remarks

- Challenging problems in many domains exist and are amenable to a rigorous approach to design methodologies, models and tools
- Model-based and Platform-based Design is a MUST
- Control algorithm design is critical
- Optimized Architecture selection is essential
- Holistic view of the design problem must be established
- Multidisciplinary approach needed: a challenge for education and recruiting