Computing Systems for Signal Processing

Part 4: PC vs. Embedded → Power, Reliability, Real-time

October 25th 2011

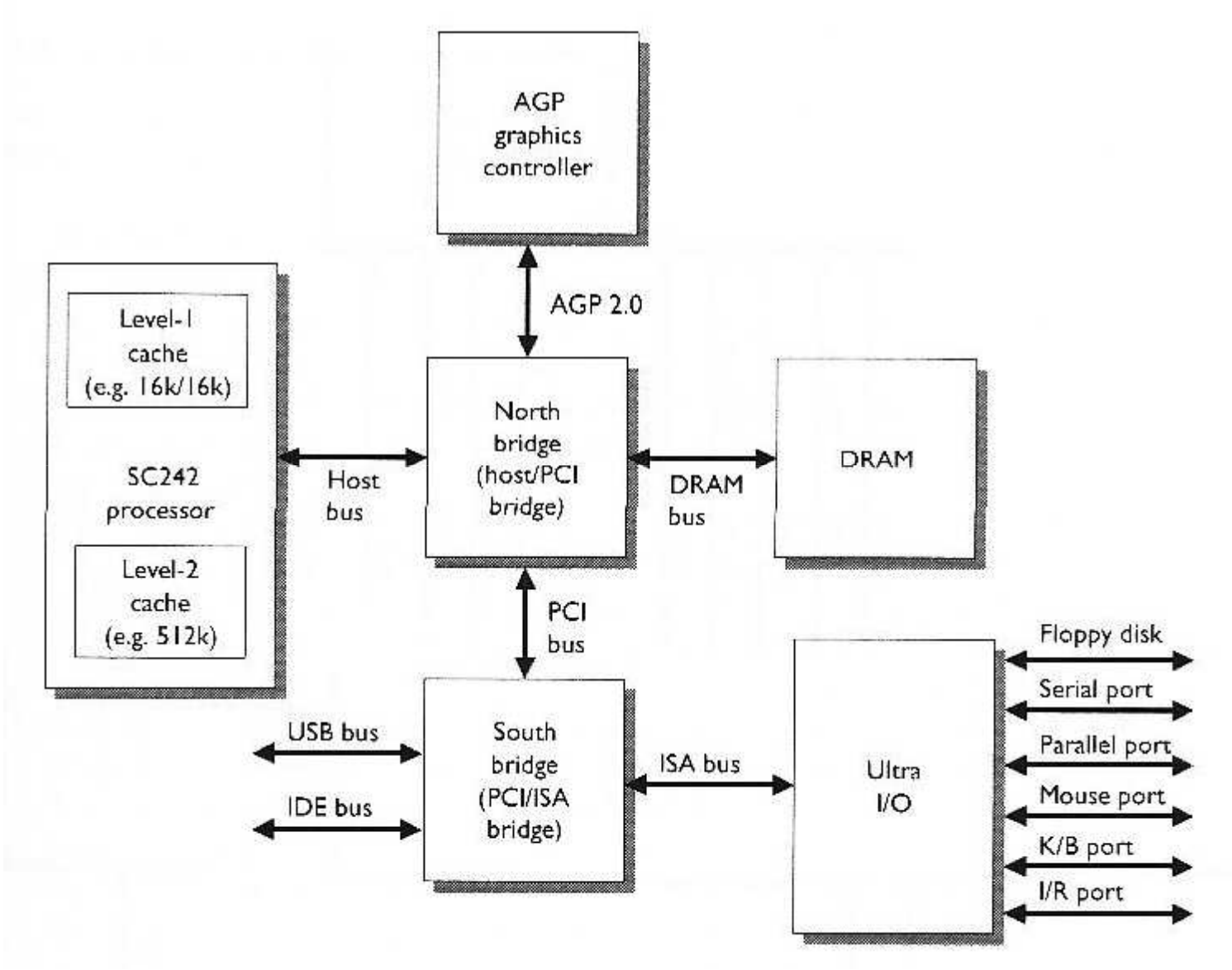
Summary

- **New architecture is driven by power and thermal**
  - Transistor count continues to increase thanks to Moore's law
  - Most systems are limited by thermals
- **Parallelism is needed for perf and power efficiency**
  - Instruction level parallelism: Pipeline, OOO, VLIW
  - Data-level parallelism: SIMD, Vector, 2D SIMD Matrices
  - Thread level parallelism: SMP, CMP, SMT/HT
  - System level parallelism: I/Os, Memory Hierarchy
- **Key Issues with Parallelism**
  - Amdahl's law
  - Extracting parallelism from applications
  - Systems Issues → the rest of the system needs to be well balanced
  - Programming models need to be portable, easy to learn and efficient
- **Application Specific Signal Processors and SoCs**
  - Spectrum: ASICs, FPGA, Media Proc, DSP, GPP + ISA extensions
  - Depending on power/performance constraints, often a mix (SoC)

PC Architecture

- PCs have been driving innovation in processors for 30 years
- What are the key components in the PC architecture?
- What is the difference between a PC and an embedded architecture?

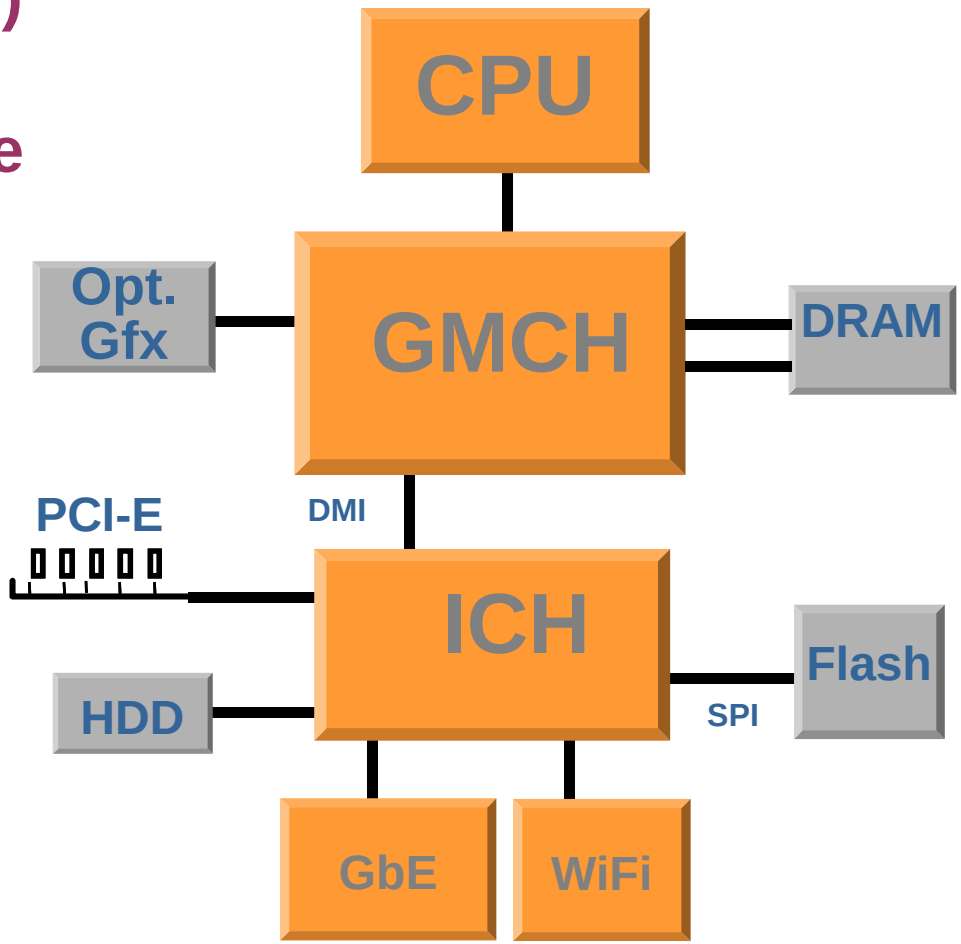
Key Component in the Early PC Architecture



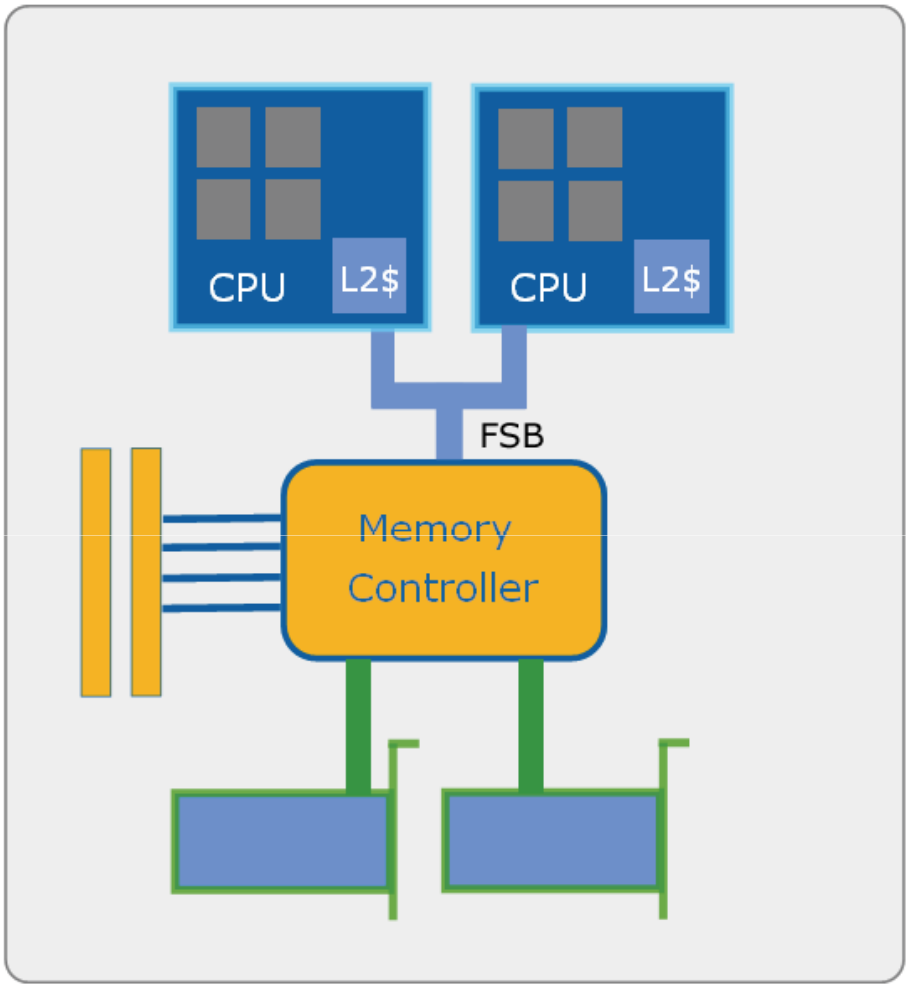
Typical PC Platform Architecture

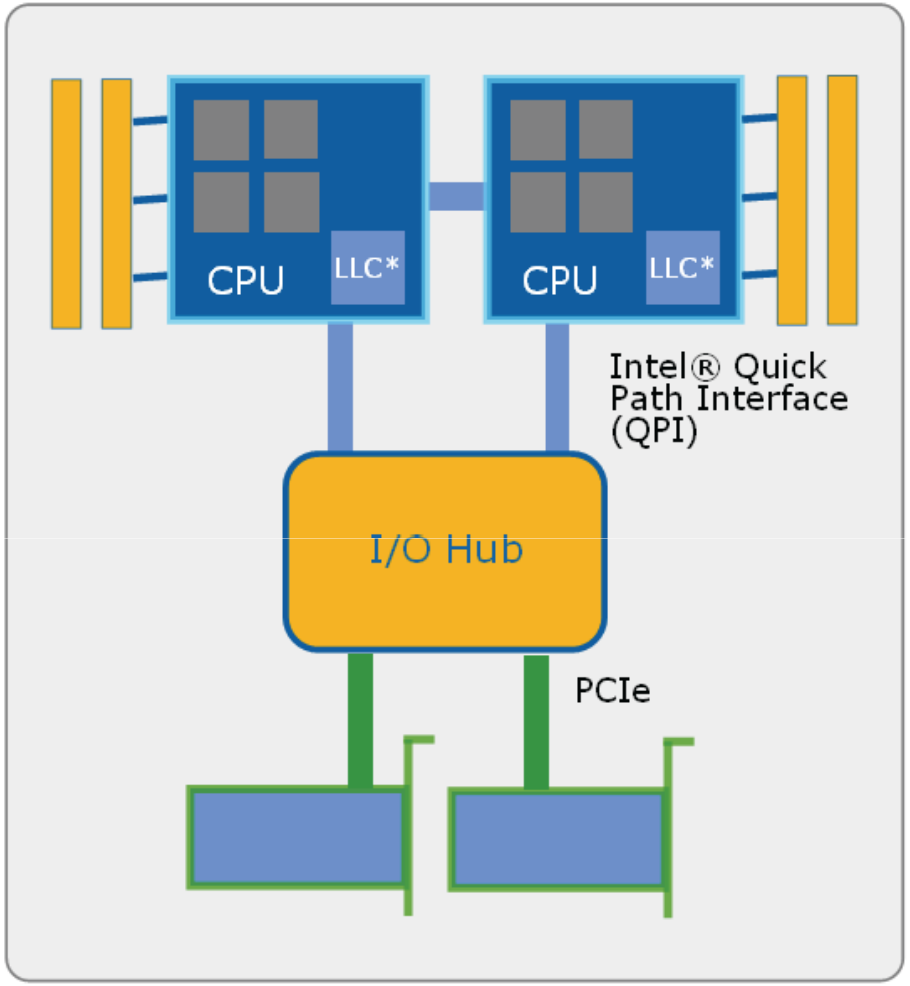
Typical PC platform (2008)

New architecture integrate the MCH in the CPU



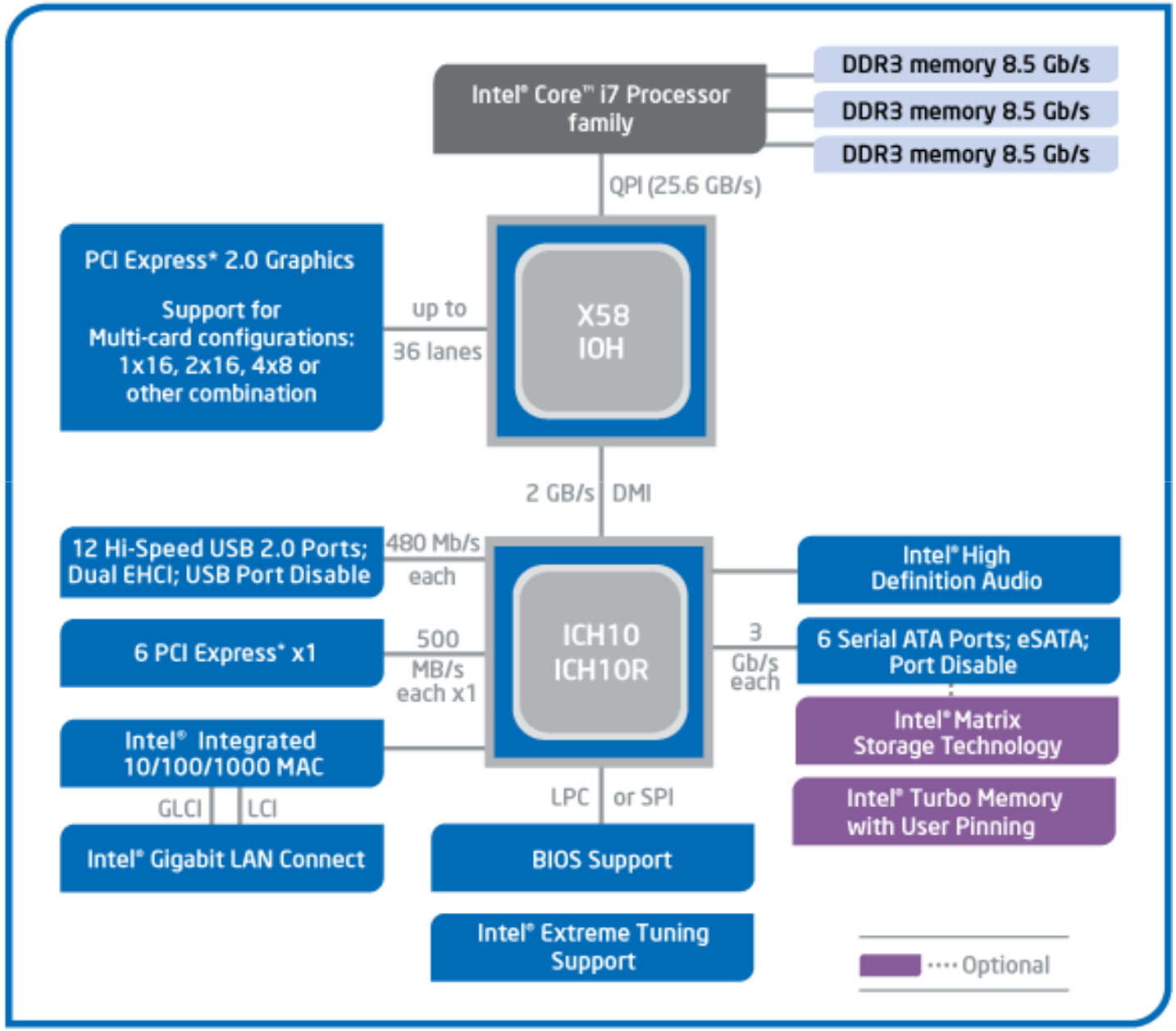
QPI vs FSB based Platform





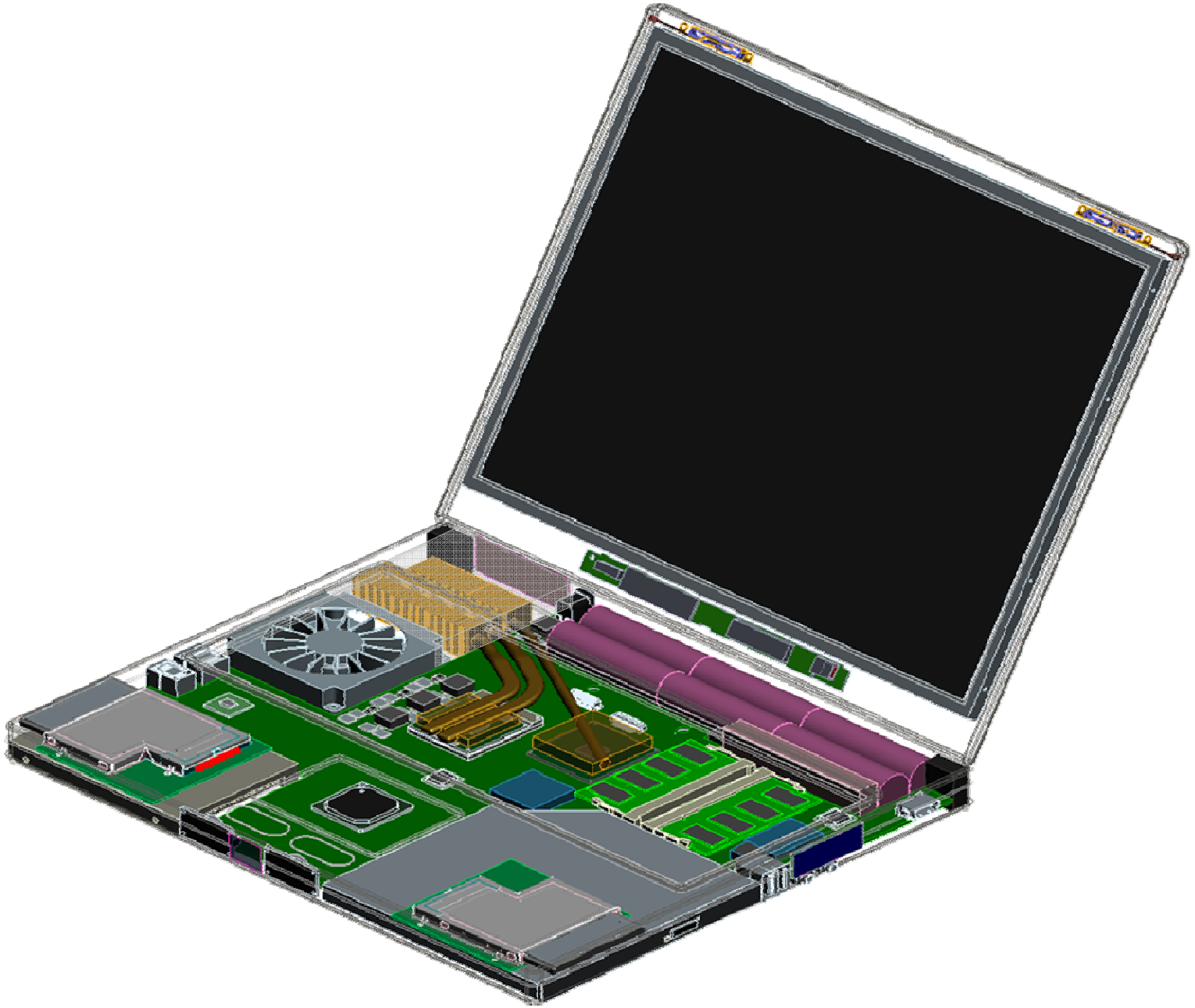
MCH is now integrated on chip

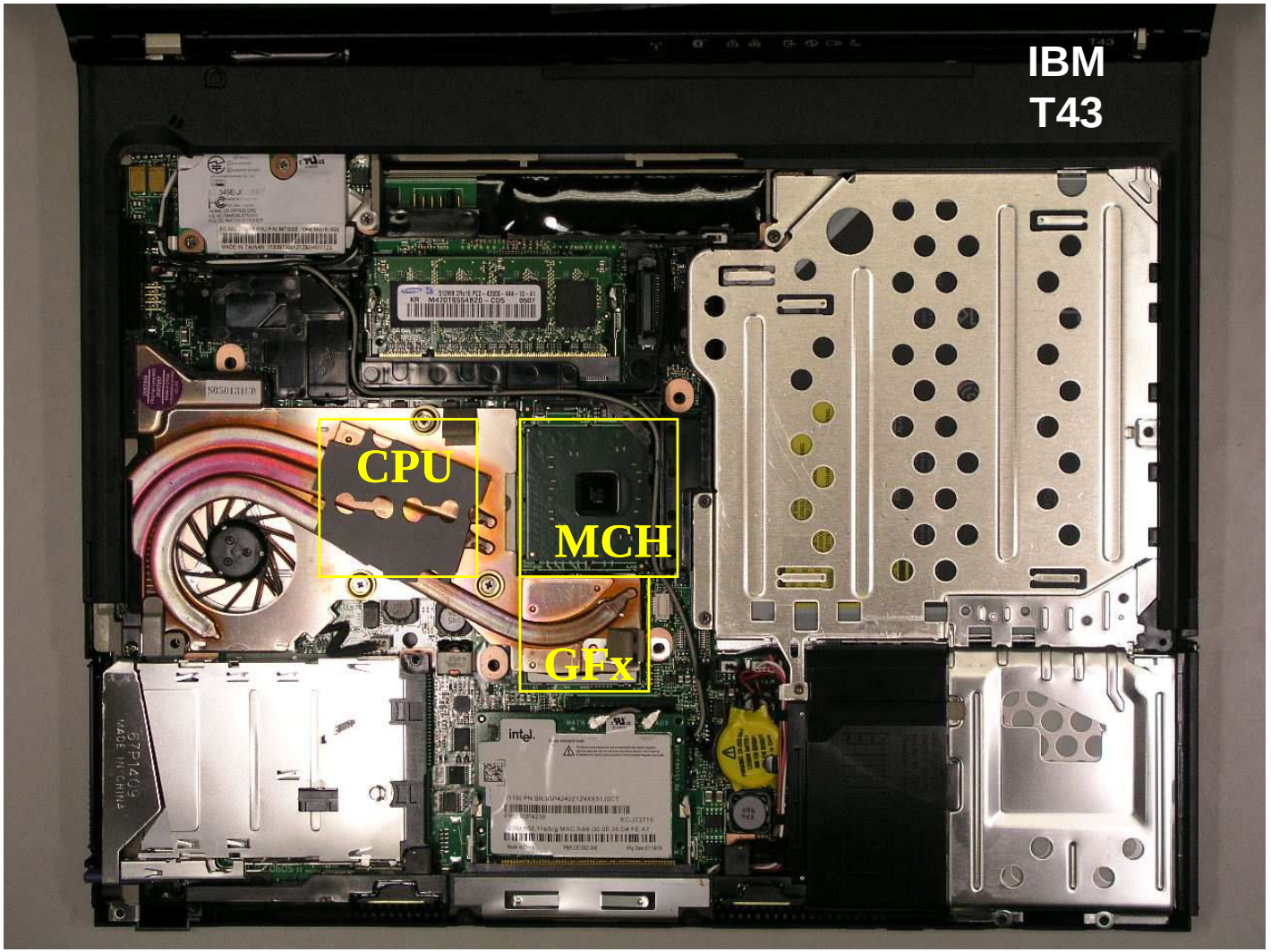
Core i7 Block Diagram



Intel® X58 Express Chipset Block Diagram

Laptop Internal View




IBM
T43
CPU
MCH
GFx

Future: PC on a Chip

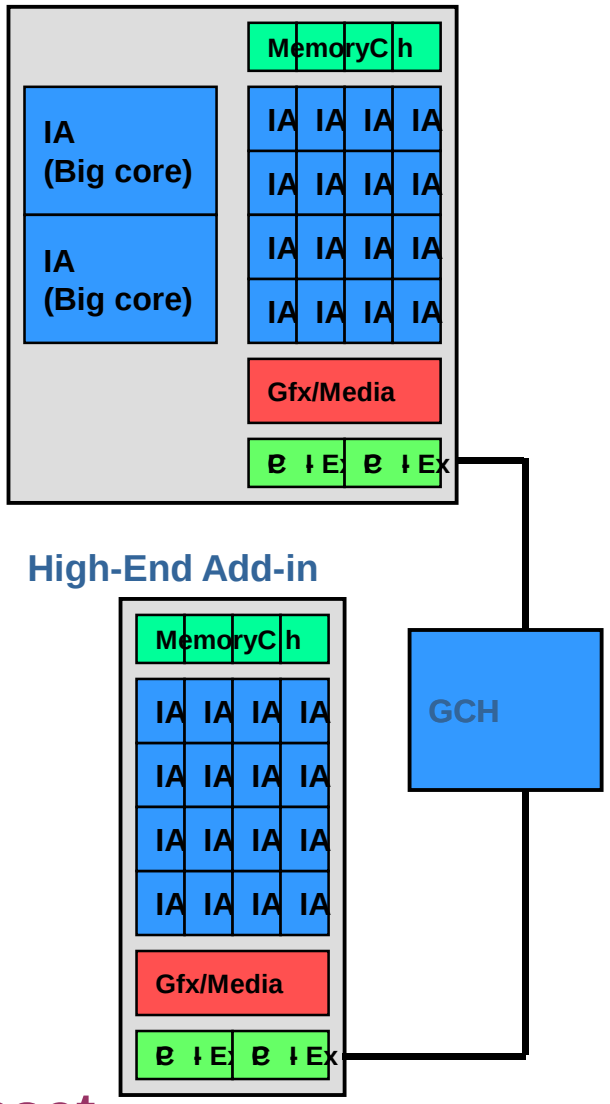
Processor will integrate

- Big core for single thread perf

- Small core for multithreaded perf

- some dedicated hardware units for

- graphics
- media
- encryption
- networking function
- other function specific logic



Systems will be heterogeneous

Processor core will be connected to

- one or multiple many-core cards

- and dedicated function hw in the chipset

+ reconfigurable logic in the system or on chip?

Embedded Architecture: What's different?

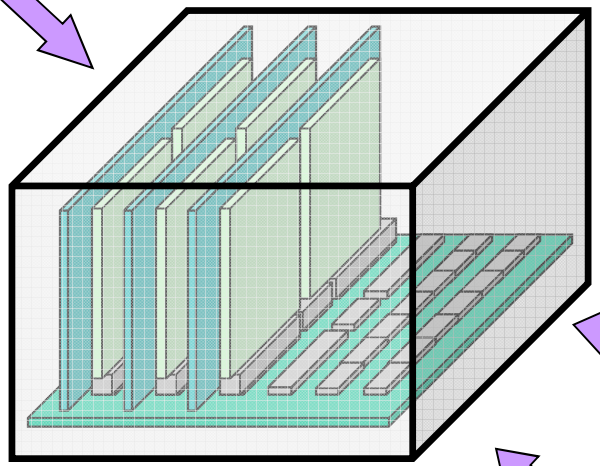
- Power constraints
- Reliability
- Redundancy
- Predictability (for Certification)

- EXAMPLES:
  - Rack
  - Airbus
  - Rafale: radar
  - Portable devices: cellphone, MP3 player
  - Consumer set top boxes
  - Satellite
  - Train

Reminder: Embedded System Examples

- Consumer : DVD/video players, Set-top-box, Playstation, printers, disk drives, GPS, cameras, mp3 players
- Communications: Cellphone, Mobile Internet Devices, Netbooks, PDAs with WiFi, GSM/3G, WiMax, GPS, cameras, music/video
- Automotive: Driving innovation for many embedded applications, e.g. Sensors, buses, info-tainment
- Industrial Applications: Process control, Instrumentation
- Other niche markets: video surveillance, satellites, airplanes, sonars, radars, military applications

A wide range of Software and Hardware



Applications
- Target Recognition
- Radar Processing
- Sensor Fusion
- Image Processing
- Real Time Recorders
- Mission Computing
- Digital Map
- Signal Intelligence

Software
- VxWorks
- Linux
- Integrity
- Windows
- BIT
- BSP
- VMEdriver
- 1553
- RDMA
- IPC/GBM
- DMA
- HDLC/SDLC
- TCP/IP

IO Connections
- Video
- RGB/DVI
- LVDS
- GigE
- Ethernet
- Fibre Channel
- SCSI
- SFPDP
- RocketIO
- Canbus
- MIL-STD-1553
- ARINC
- RF Analog
- Low Speed Analog
- Discrete IO
- Serial RSxxx
- C40 Links
- SharcLinks
- SCRAMnet
- Syncro/Resolver

Platforms
- VME
- VPX
- cPCI
- VITA 46/48
- Custom form Factors

Processing
- PowerPC SBC
- Quad PowerPC DSP
- FPGA Processing
- Graphics Processing
- Pentium SBC
- Front End Processing

Backplane Buses
- Raceway
- StarFabric
- SRIO
- PCIe
- ASI

Typical Embedded PC Architecture

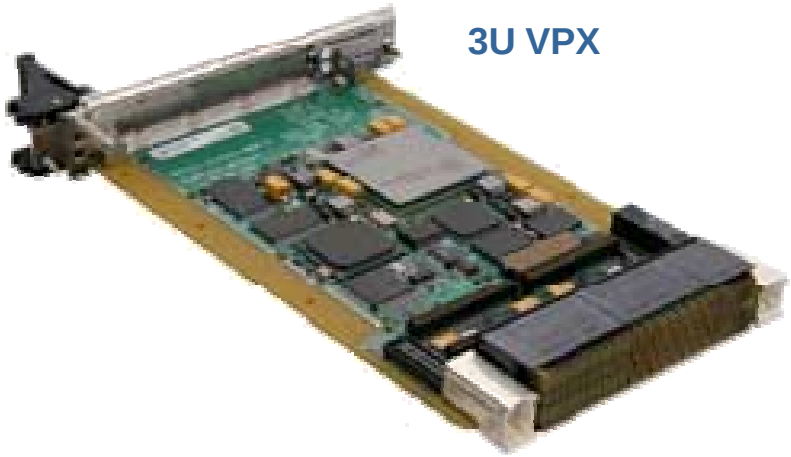
Hardened PC

Examples of Embedded Boards

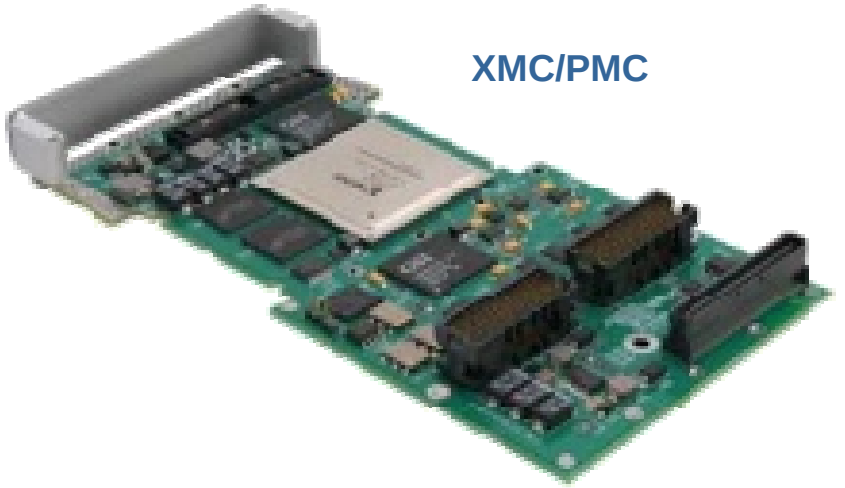
6U VPX

6U VME

3U VPX



XMC/PMC



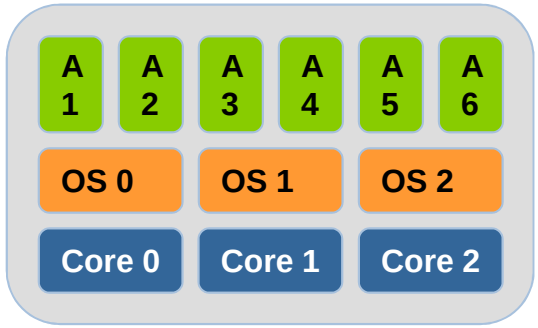
Ruggedisation Levels

	Level 0	Level 50 AC	Level 100 AC	Level 200 AC	Level 100 CC	Level 200 CC
Operating Temperature	**0C – 50C**	**-20C to 65C**	**-40°C to 71°C**	**-40°C to 85°C**	**-40°C to 71°C**	**-40°C to 85°C**
Storage	**-40C to85C**	**-40°C to 85°C**	**-55°C to 125°C**	**-55°C to 125°C**	**-55°C to 125°C**	**-55°C to 125°C**
Humidity operating	**0 to 95% non-condensing**	**0 to 95% non-condensing**	**0 to 100% non-condensing**	**0 to 100% non-condensing**	**0 to 100% non-condensing**	**0 to 100% non-condensing**
Humidity Storage	**0 to 95% Non-condensing**	**0 to 95% Non-condensing**	**0 to 100% condensing**	**0 to 100% condensing**	**0 to 100% condensing**	**0 to 100% condensing**
Vibration sign	**2 g peak 15-2 kHz**	**2 g peak 15-2 kHz**	**10 g peak 15-2 kHz**	**10 g peak 15-2 kHz**	**10 g peak 15-2 kHz**	**10 g peak 15-2 kHz**
Vibration random	**0.01 g2/Hz 15-2 kHz**	**0.01 g2/Hz 15-2 kHz**	**0.04 g2/Hz 15-2 kHz**	**0.04 g2/Hz 15-2 kHz**	**0.1 g2/Hz 15-2 kHz**	**0.1 g2/Hz 15 Hz-2 kHz**
Shock	**20 g Peak**	**20 g Peak**	**30 g peak**	**30 g peak**	**40 g peak**	**40 g peak**
Conformal coat	**No**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

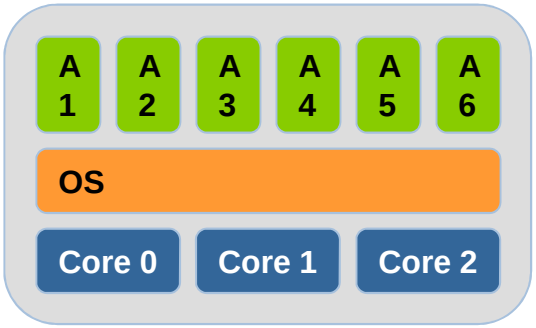
Challenge: How to use Multi-core?

- Obviously, a multi-core platform shall offer the same level of safety as a single-core processor device:
  - WCET must be computable
  - Partitioning must be ensured

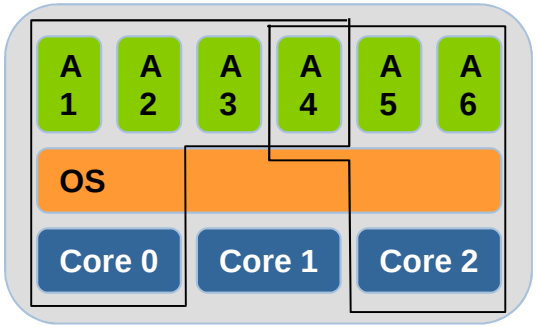
- Main usage models:
  - Two main ones: AMP and SMP
  - Some others proposed by some RTOS vendors (e.g: BMP)
  - Or a mix of AMP and SMP



AMP
Asymmetric multi-processing



SMP
Symmetric multi-processing



BMP
Bound multi-processing

Computing Systems for Signal Processing

Part 5: Hard Real-Time Systems and RTOS

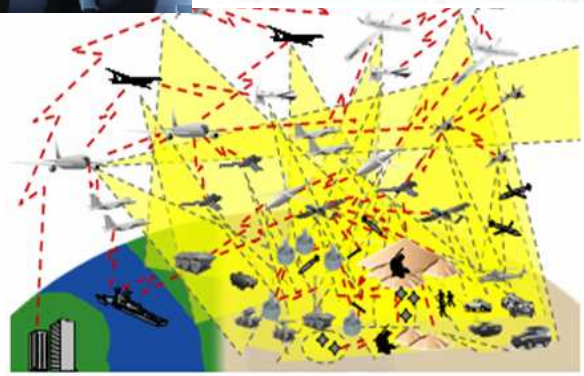
October 25th 2011

A critical system is any system whose 'failure' could threaten human life, the system's environment or the business of the organisation which operates the system.

'Failure' in this context does NOT mean failure to conform to a specification but means any potentially threatening system behaviour.








Critical Systems classes

Safety-critical systems

- Failure results in loss of life, injury or major environmental damage;
- e.g. Flight control system, Nuclear plant protection system;

Mission-critical systems

- Failure results in failure of some goal-directed activity;
- e.g. spacecraft navigation system;

Business-critical systems

- Failure results in high economic losses;
- e.g. customer accounting system in a bank;

Many embedded systems are critical !

Dimensions of criticity

Dependability

- The dependability in a system reflects the user's trust in that system

Time-sensitiveness

Integration with the physical/environmental processes

Two classes of safety-critical embedded software systems:

Primary safety-critical systems

- Embedded software systems whose failure can cause the associated hardware to fail and directly threaten people.

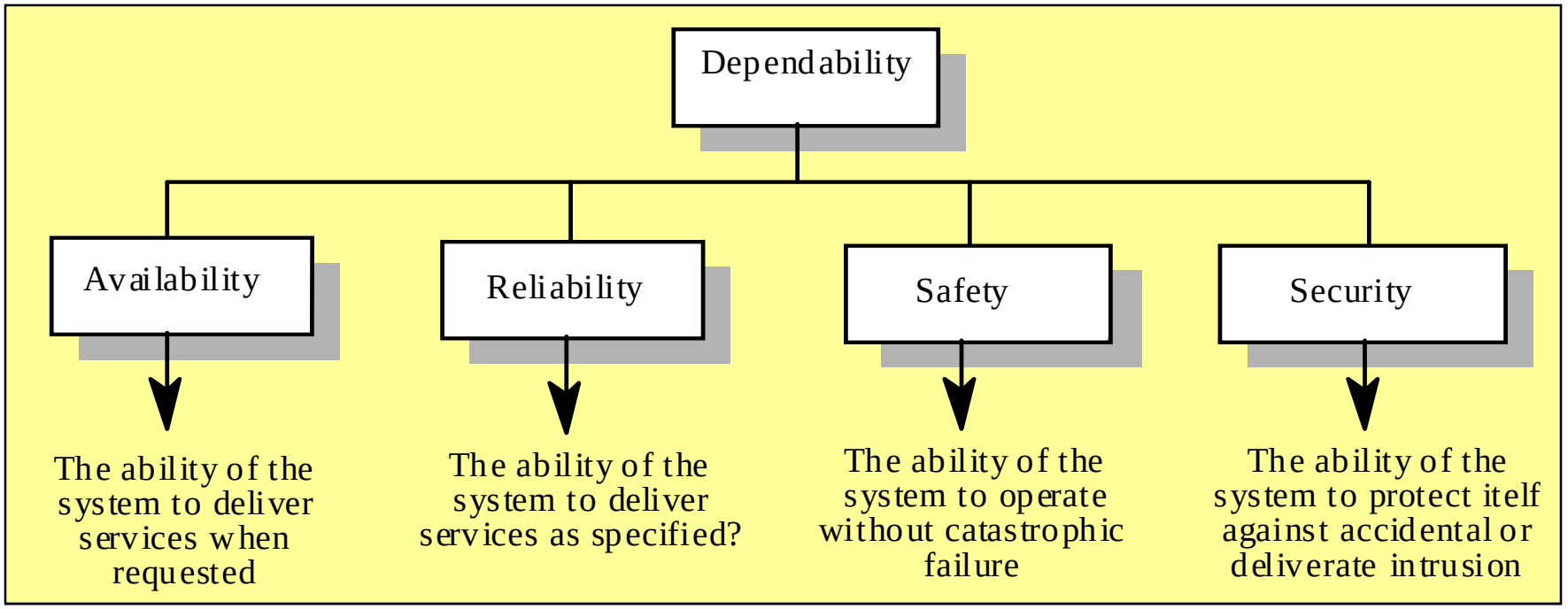
Secondary safety-critical systems

- Systems whose failure results in faults in other systems which can threaten people

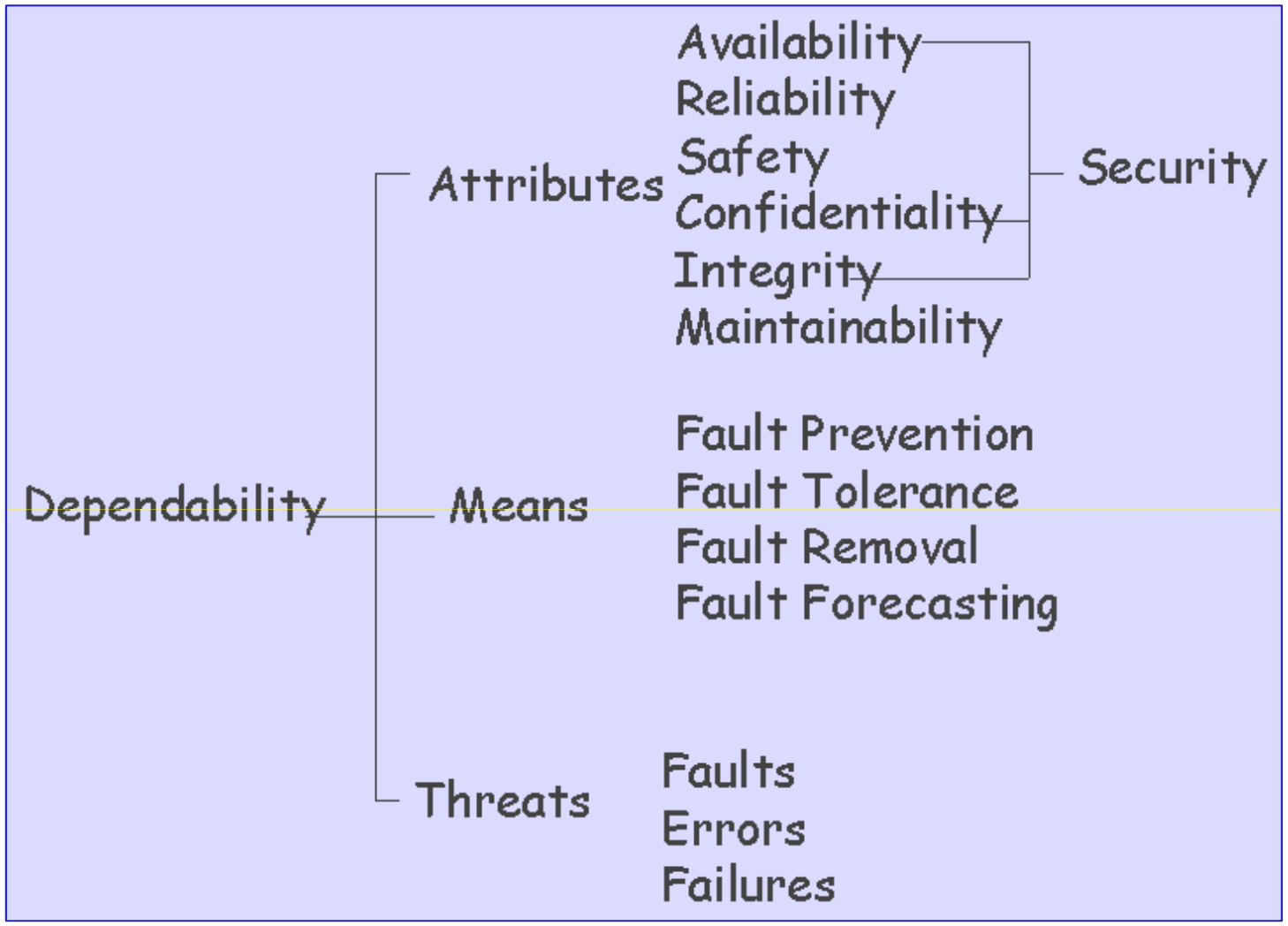
The concept of dependability

For critical systems, it is usually the case that the most important system property is the dependability of the system

- The dependability of a system reflects the **user's degree of trust** in that system. It reflects the extent of the user's confidence that it will operate as users expect and that it will not 'fail' in normal use



Dependability Terminology



Availability and reliability

Reliability

- The probability of failure-free system operation over a specified time in a given environment for a given purpose. ***R(t) =*** probability of functional correctness if it was satisfied at *t*=0

Availability

- The probability that a system, at a point in time, will be operational and able to deliver the requested services.

Other adjacent dependability properties

- Repairability: Reflects the extent to which the system can be repaired in the event of a failure
- Maintainability : Reflects the extent to which the system can be adapted to new requirements;
- Survivability : Reflects the extent to which the system can deliver services whilst under hostile attack;
- Error tolerance : Reflects the extent to which user input errors can be avoided and tolerated.

Socio-technical view of critical systems

Hardware failure

- Hardware fails because of design and manufacturing errors or because components have reached the end of their natural life.

Software failure

- Software fails due to errors in its specification, design or implementation.

Operational failure

- Human operators make mistakes. Now perhaps the largest single cause of system failures.
- Eg Ariane V failure despite redundant code: process issue

Reactive and Real-time

Embedded Systems are often reactive, real time, critical

„A reactive system is one which is in continual interaction with is environment and executes at a pace determined by that environment“ [Bergé, 1995]

Reactive systems means Real Time responsiveness :

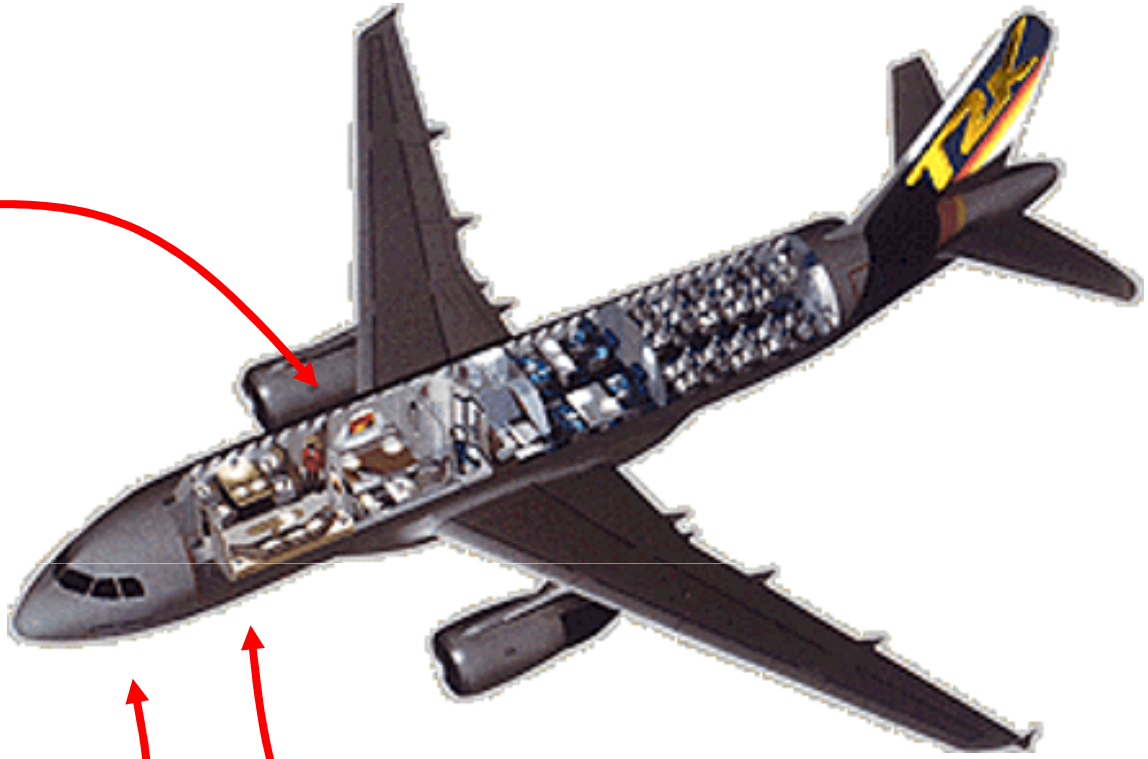
- Timeliness : response time within a given time slot
- A late response is a fault

Time critical systems „A real-time constraint is called hard, if not meeting that constraint could result in a catastrophe“ [Kopetz, 1997].

- Other constraints are called soft RT
- Response time is not statistical : worst case

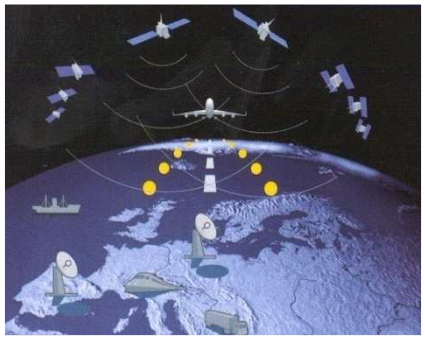
Key Example: Avionics Systems

Flight management



Cockpit system

Galileo Egnos Comm.



Some Technical Challenges

Embedded safety critical software development

- Productivity
- Validation, Verification, Certification

Design challenge: networked, embedded, hard real-time, safe and secure

Technical approaches

- Built-in reliability and real time behaviour on safe hybrid systems
  - Joint Modelling of physical and computational features
  - Formal methods, Modelling techniques (formalisms)
- Automated test & validation processes
  - Software development productivity
  - Cost-effective certification and security
  - Modularity of embedded software architecture
- Mission configurable reliable platforms
  - Design techniques for cost effective reliable architectures: certifiably fault – tolerant networks and middlewares
  - dependable adaptive distributed middleware services, standards

Safety, Security

Safety: system's ability to operate, normally or abnormally, without danger of causing human injury or death and without damage to the system's environment

Safety and reliability are related but distinct

- Reliability is concerned with conformance to a given specification and delivery of service
- Safety is concerned with ensuring system cannot cause damage irrespective of whether or not it conforms to its specification

Safety achievements : built-in properties for hazard avoidance, hazard detection and removal, damage limitation

Security is a system property that reflects the system's ability to protect itself from accidental or deliberate external attack

- Security is becoming increasingly important as systems are networked so that external access to the system through the Internet is possible

Security is an essential pre-requisite for availability, reliability and safety : safety validation relies on demonstrating that a particular system is safe

In summary

Computer-based systems are socio-technical systems which include hardware, software, operational processes, procedures and people.

An increasing number of socio-technical systems are critical systems

Systems have emergent properties i.e. properties which are only apparent when all sub-systems are integrated.

Critical systems have dependability attributes - reliability, availability, safety and security

Design & Engineering for Safety

- Most Critical Information Systems have to comply with safety regulations (SWAL, DO-178B, SIL …)
- Design and engineering for safety is currently costly and cumbersome
- Need for technologies enabling « safety proven » design and engineering
- Costs of critical system failure are so high that development methods may be used that are not cost-effective for other types of system.

Some Research issues

- Insertion of formal methods for system specification, verification/ testing, timing analysis (eg WCET)
- High integrity programming: incorporation of redundant code and self-checking in programs, threads analysis, timing properties analysis (WCET)
- Control theory/functional modeling with software architecture
- Simulation based seamless integration from specification to test means
- Multidisciplinary / multiviewpoint engineering: include methodology, architectures, and applications while ensuring efficiency of the architecture =>**standard design techniques must be adapted**

Computing Systems for Signal Processing

Part 6: Power Constraints

October 25th 2011

Platform Power Measurements

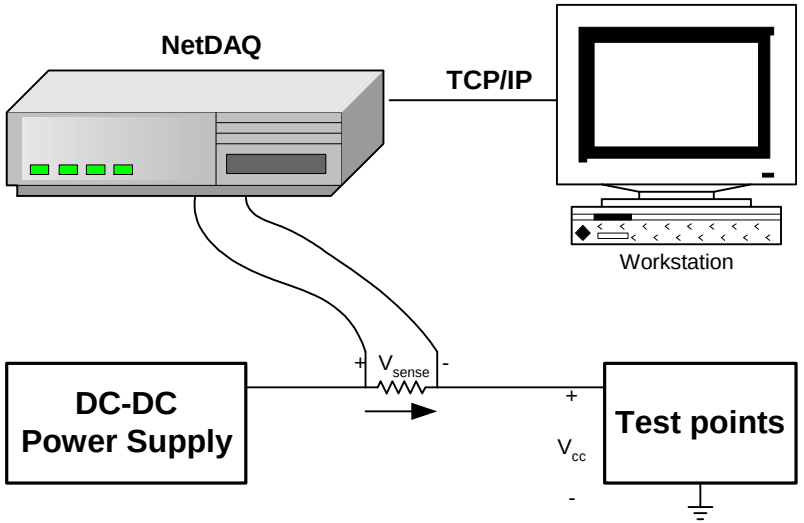
Fluke NetDAQ
Connected to a PC → log files
Sense Resistors
Power: P = I x V

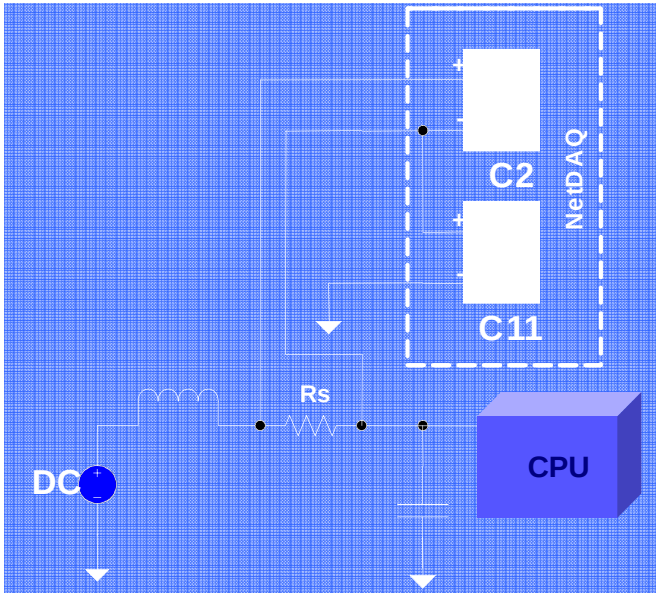
- Current: I
- Voltage: V

Shunt Resistor Method

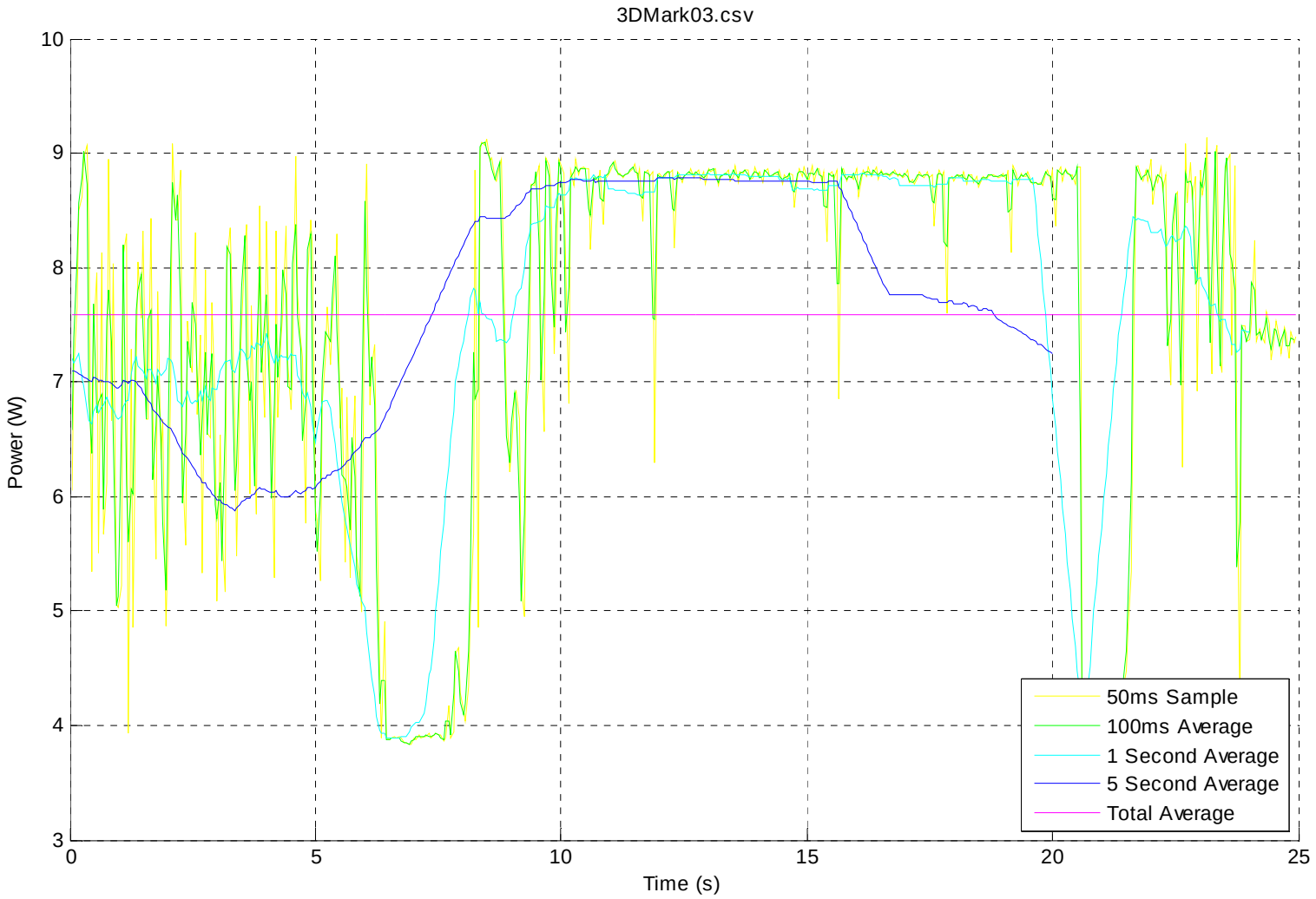
- V = Voltage at Input to CPU
- I = Vs/Rs
  - Vs = Voltage Drop Across Rs
  - With Rs = 100 mOhm

The same methodology is applied to each power rail for each component (CPU, Memory, GMCH, ICH)



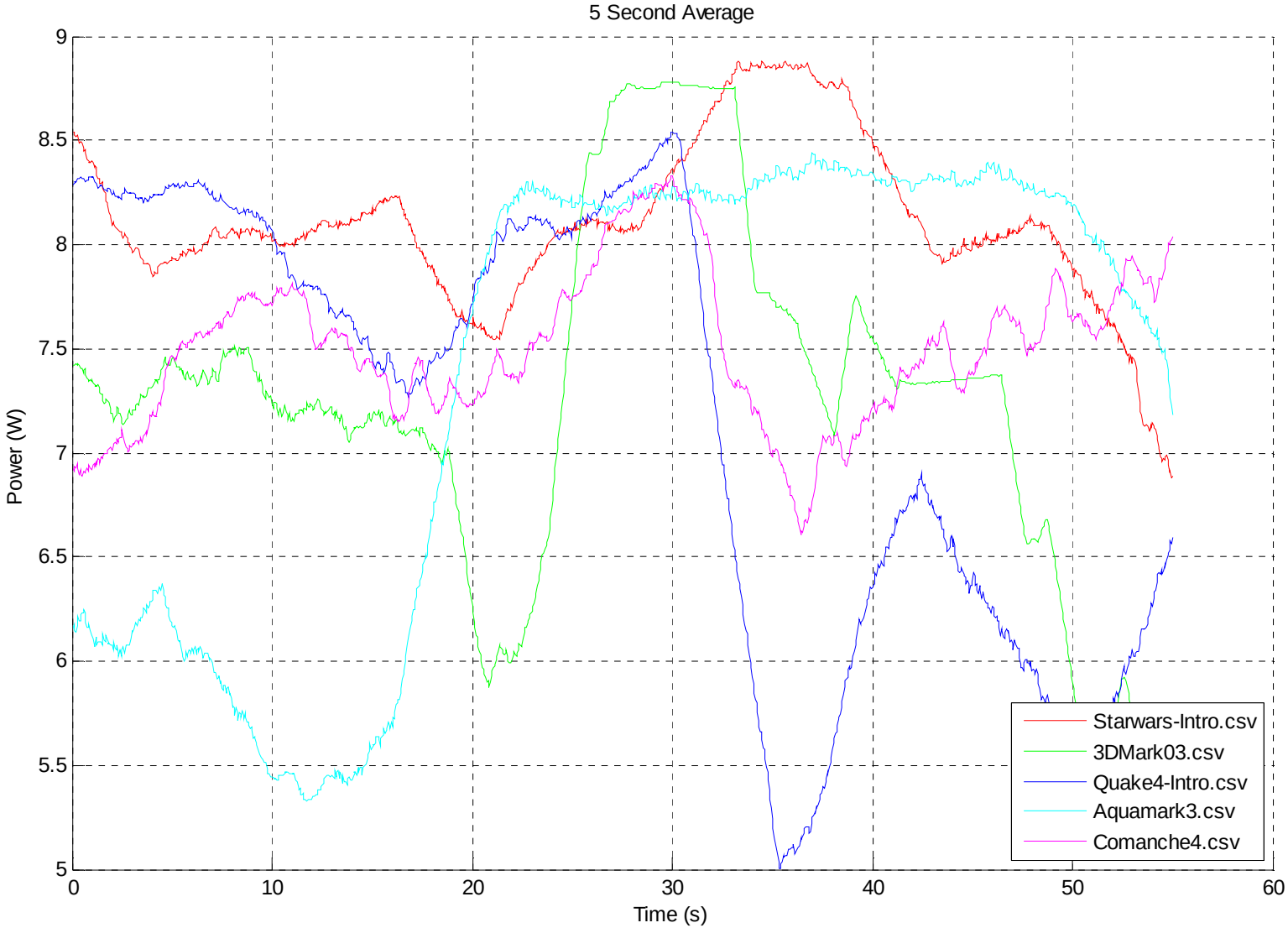


Platform Power graphs



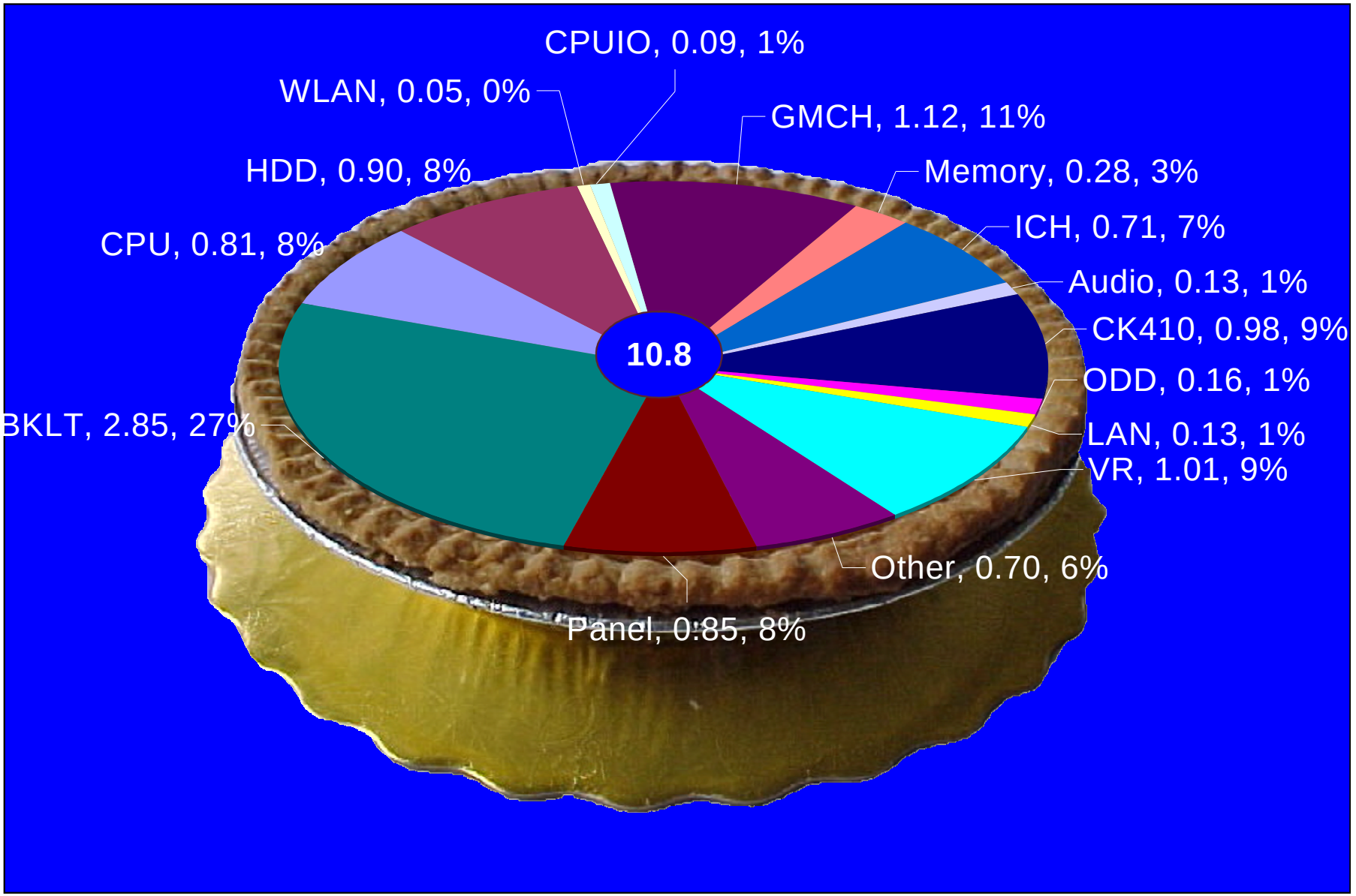
Data is acquired with a 50ms sampling rate and processed to estimate power with other moving average window

Measured GMCH power



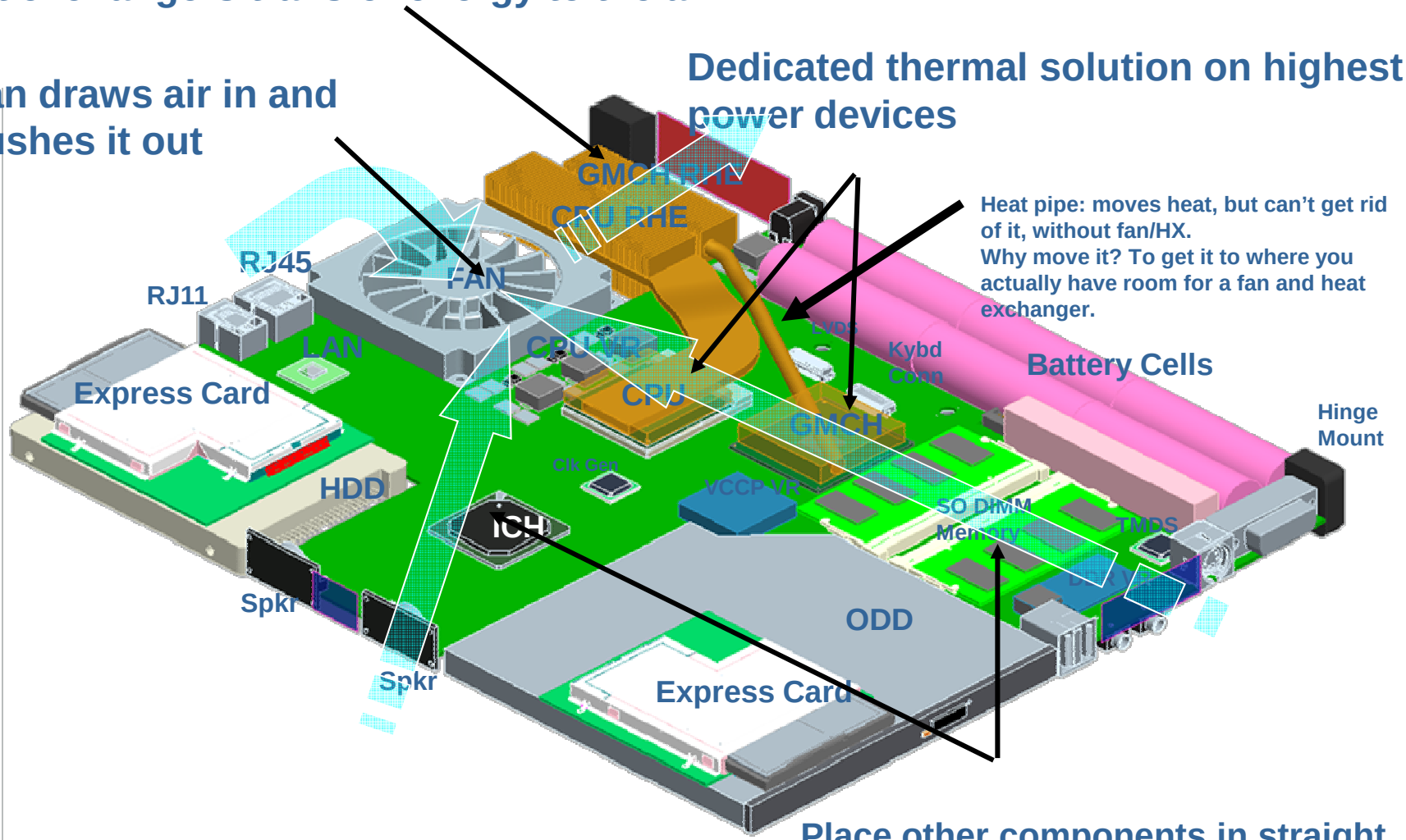
Power Profile for top 5 GMCH apps on Crestline with 5s moving average

Average Platform Power Distribution



T&L Cooling Design & Approach

Heat exchangers transfer energy to the air



Computing Systems for Signal Processing

Part 7: Critical and Complex Systems

October 25th 2011

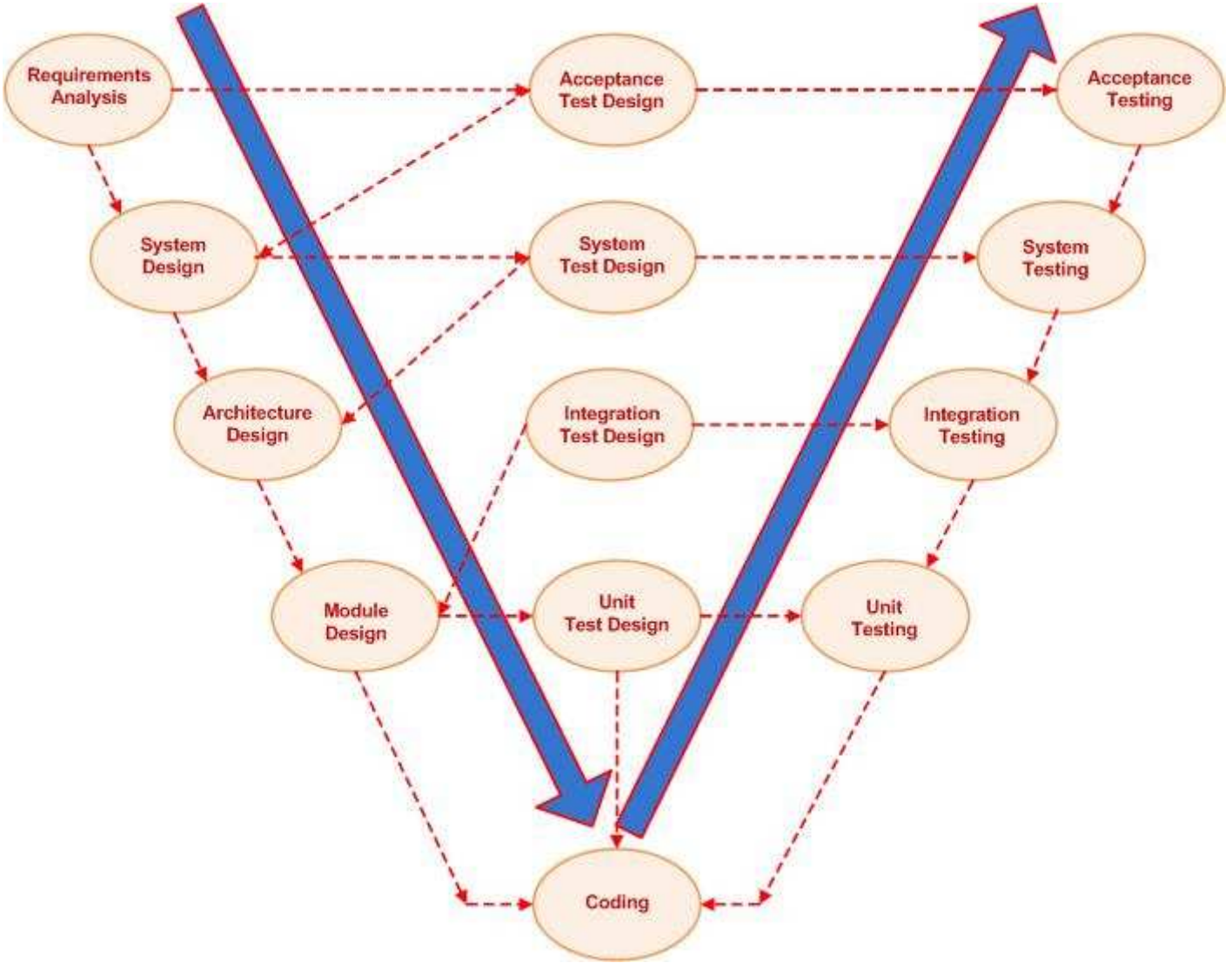
Outline

1. Embedded System Development
   - V cycle
2. System Modelling
   - The right level of abstraction
3. Platform Based Design
   - What's a platform?
   - Meet in the middle
4. Platform Simulation
   - Requirements
   - Example: SystemC and TLM

Systems Development Lifecycle: V-Model

- The V-model is a graphical representation of the systems development lifecycle. It summarizes the main steps to be taken in conjunction with the corresponding deliverables within computerized system validation framework
- The V-model is a process that represents the sequence of steps in a project life cycle development. It describes the activities and results that have to be produced during product development.
- The left side of the V represents the decomposition of requirements and creation of system specifications
- The right side of the V represents integration of parts and their verification

Systems Development Lifecycle: V-Model



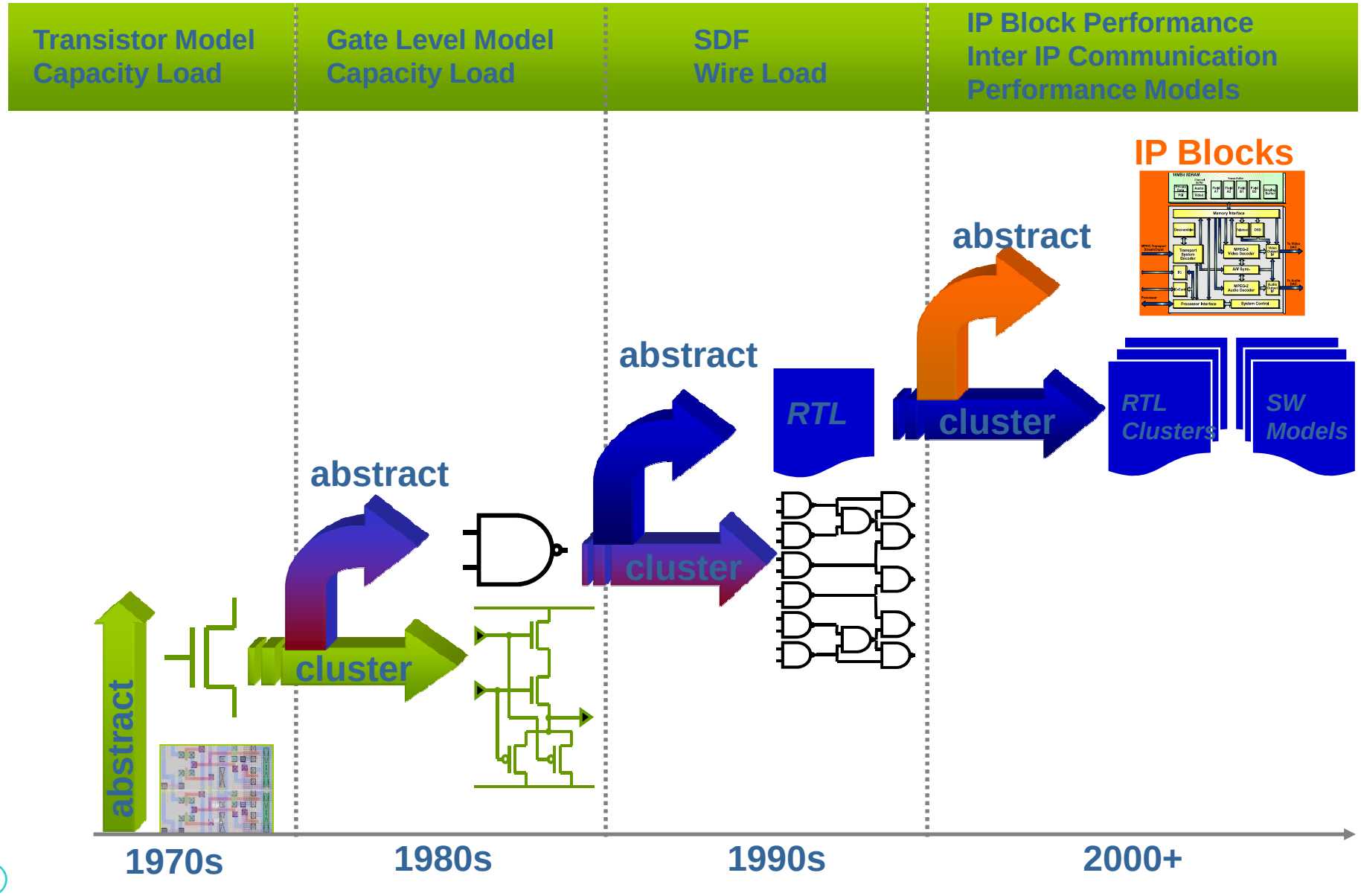
V-Model Objectives

- Minimization of Project Risks:
  - The V-Model improves project transparency and project control by specifying standardized approaches and describing the corresponding results and responsible roles. It permits an early recognition of planning deviations and risks and improves process management, thus reducing the project risk.
- Improvement and Guarantee of Quality:
  - As a standardized process model, the V-Model ensures that the results to be provided are complete and have the desired quality. Defined interim results can be checked at an early stage. Uniform product contents will improve readability, understandability and verifiability.
- Reduction of Total Cost over the Entire Project and System Life Cycle:
  - The effort for the development, production, operation and maintenance of a system can be calculated, estimated and controlled in a transparent manner by applying a standardized process model. The results obtained are uniform and easily retraced. This reduces the acquirers dependency on the supplier and the effort for subsequent activities and projects.
- Improvement of Communication between all Stakeholders:
  - The standardized and uniform description of all relevant elements and terms is the basis for the mutual understanding between all stakeholders. Thus, the frictional loss between user, acquirer, supplier and developer is reduced.

Outline

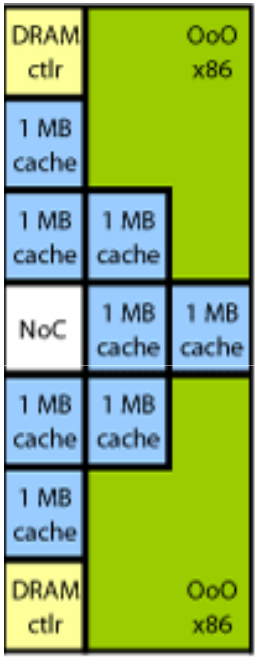
1. Embedded System Development
   - V cycle
2. System Modelling
   - The right level of abstraction
3. Platform Based Design
   - What's a platform?
   - Meet in the middle
4. Platform Simulation
   - Requirements
   - Example: SystemC and TLM

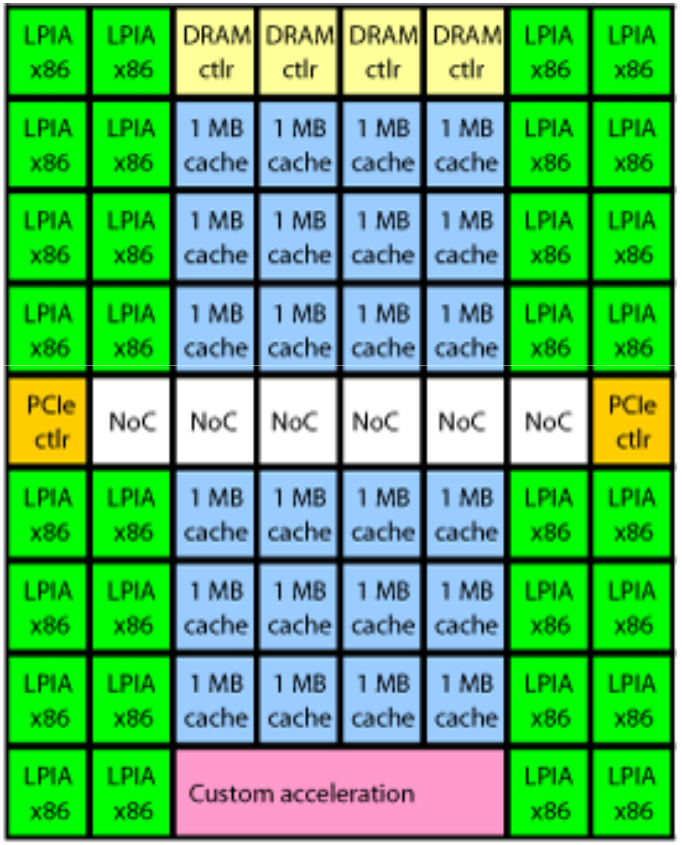
The Quest for the Next Level of Abstraction

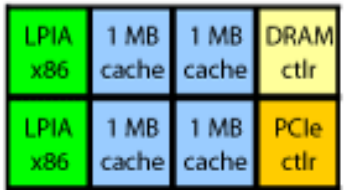


3 Classes of Applications→ 3 Types of Processors

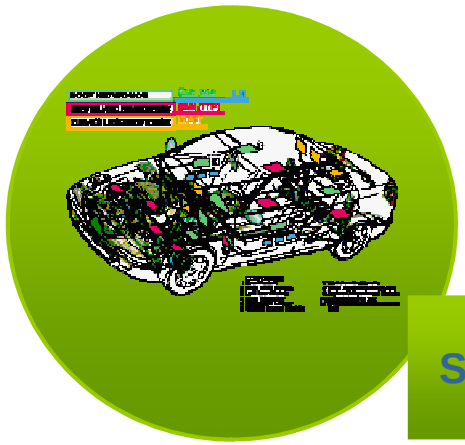
- **Reuse IP building blocks**
  - LPIA
  - DRAM Controller
  - Cache
  - Network on Chip
  - PCIe controller
  - Accelerators
- **Target multiple apps**
  - Low-power
  - Laptop
  - Desktop
  - Many-cores
  - GPUs





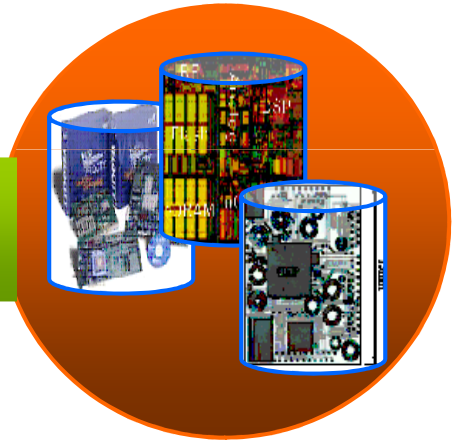


Electronic Systems Design Chain



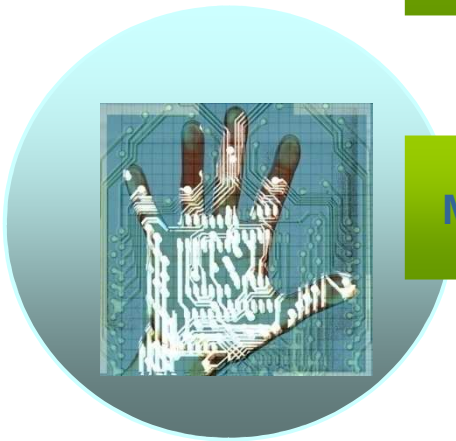
System Design

Interfaces



Implementation

IP



Manufacturing

Outline

1. Embedded System Development
   - V cycle
2. System Modelling
   - The right level of abstraction
3. Platform Based Design
   - What's a platform?
   - Meet in the middle
4. Platform Simulation
   - Requirements
   - Example: SystemC and TLM

Why platform? Why not focus on processor?

- Power efficiency requires a mix of GPP and ASSP (and DSP/Media processors)
- Partition the application between the different cores in the most power efficient manner

What's different in Platform modeling vs. GPP uarch?

- Asymmetric
- Mix of programmable (GPP, DSP) and non programmable cores (ASSP)

→ Platform simulation is different from cycle-accurate uarch simulators

What's different about platform-based design?

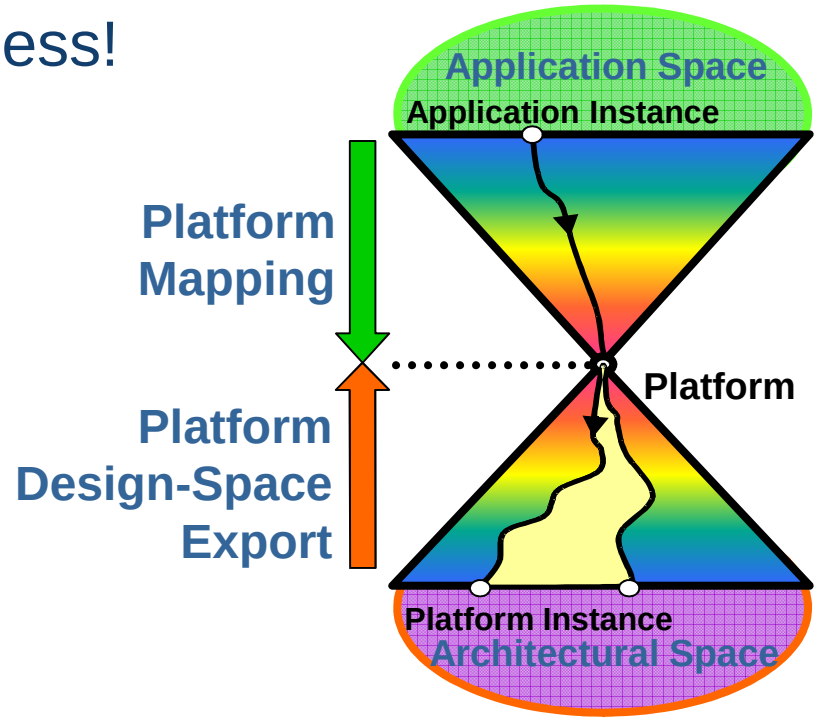
- Enable IP reuse
- Drag & Drop composability, menu-based architecture
- Validation through accurate modeling

What's a platform anyway?

- "A coordinated family of architectures that satisfy a set of architectural constraints imposed to support reuse of hardware and software components"

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!



Meet in the Middle

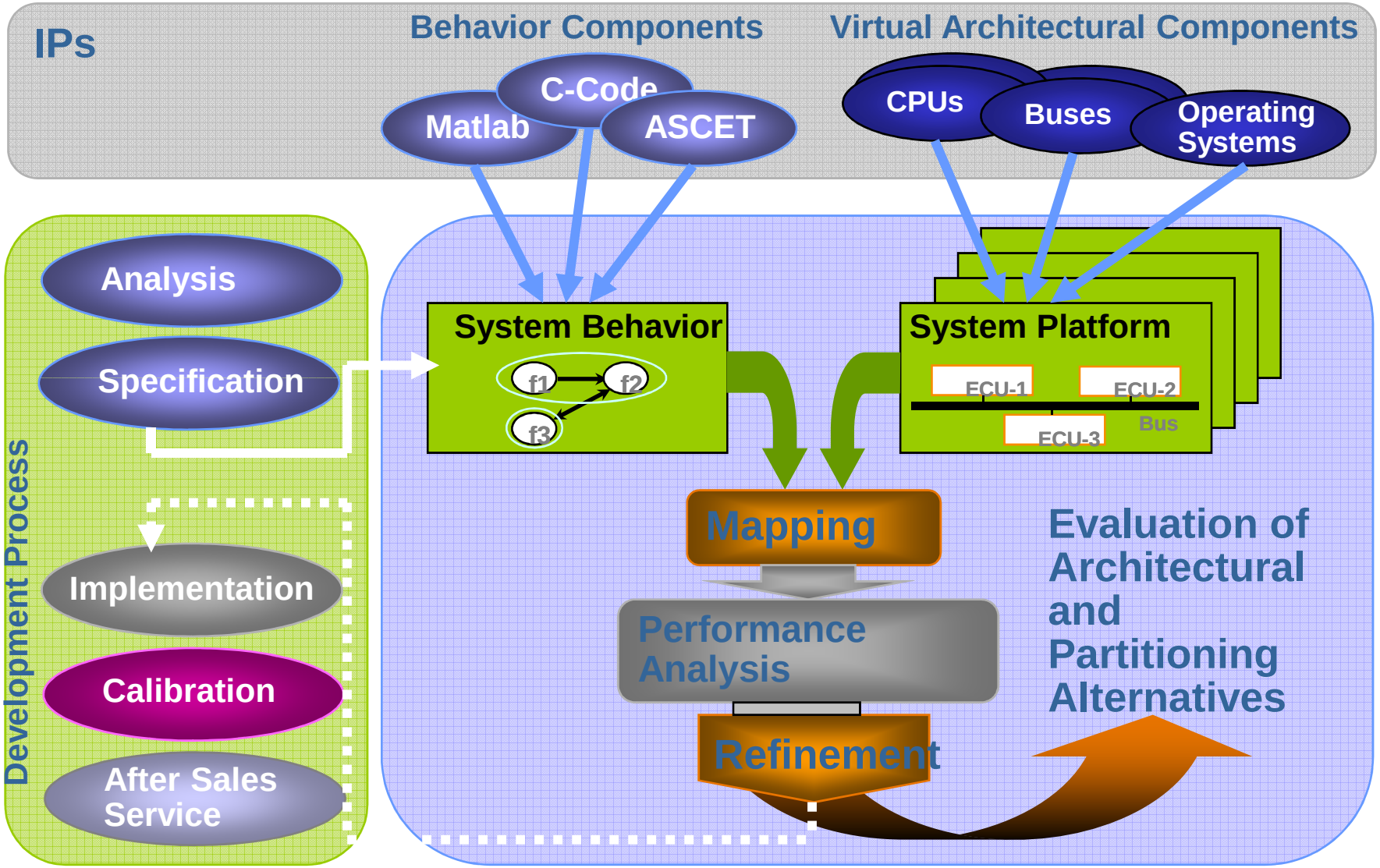
Top-Down:

- Define a set of abstraction layers
- From specifications at a given level, select a solution (controls, components) in terms of components (Platforms) of the following layer and propagate constraints

Bottom-Up:

- Platform components (e.g., micro-controller, RTOS, communication primitives) at a given level are abstracted to a higher level by their functionality and a set of parameters that help guiding the solution selection process. The selection process is equivalent to a covering problem if a common semantic domain is used.

Separation of Concerns (1990 Vintage!)



What's needed for PBD?

High Level modeling of each core and of the interconnect to enable

- Fast simulation
- Accurate results
- Power and performance models

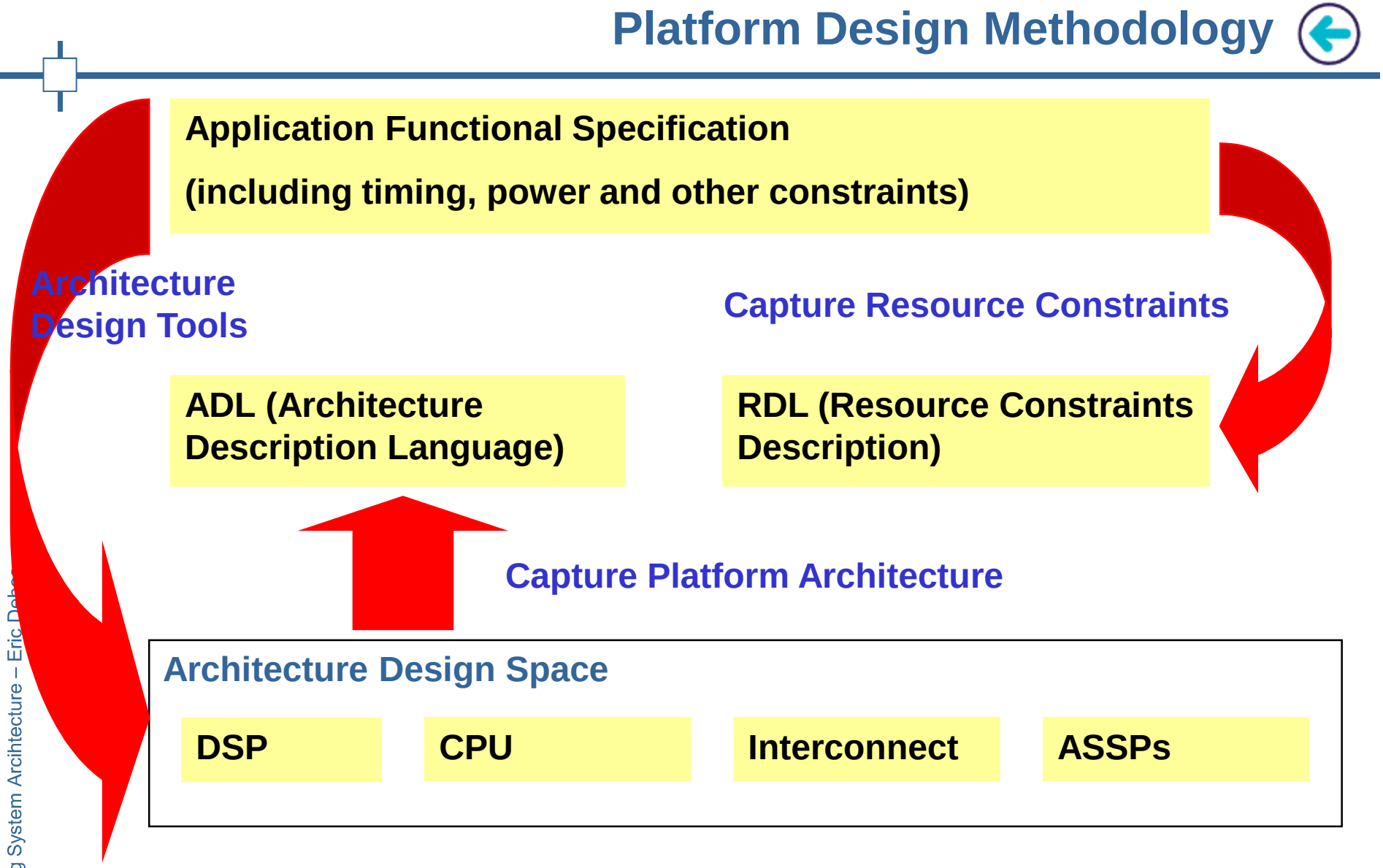
Connecting the modules

- Can it be automatic
- Type inference

Architecture Description Language
Constraint Description Language

- Performance,
- Size
- Power

Platform Design Methodology



Explore the design space and simulate the perf/power of different platform instances at multiple abstraction levels

Impact on process and skills evolution

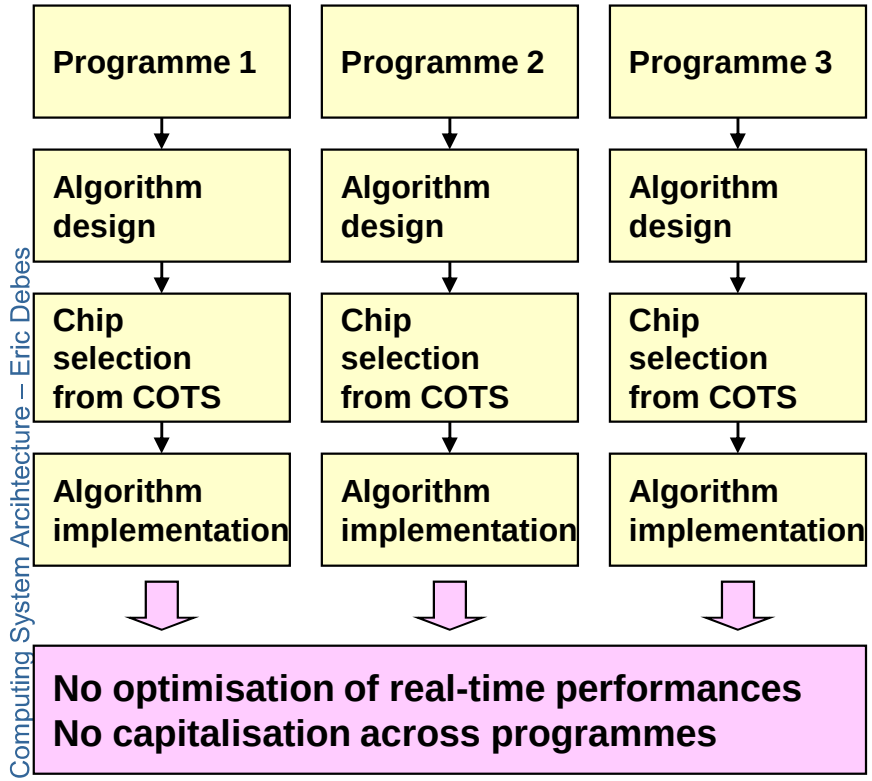
Massively parallel domain dedicated programmable processors

Required skills to support Platform-based Process

- **Platform architects:** capable to design domain dedicated massively parallel processing platforms
- **Algorithms architects:** capable to design algorithms which can efficiently map to the platform
- ***Domain architects*:** capable to ***cross optimise*** algorithm and platform design strategies
- **Parallel programmers:** capable to quickly develop the parallel code for a given algorithm

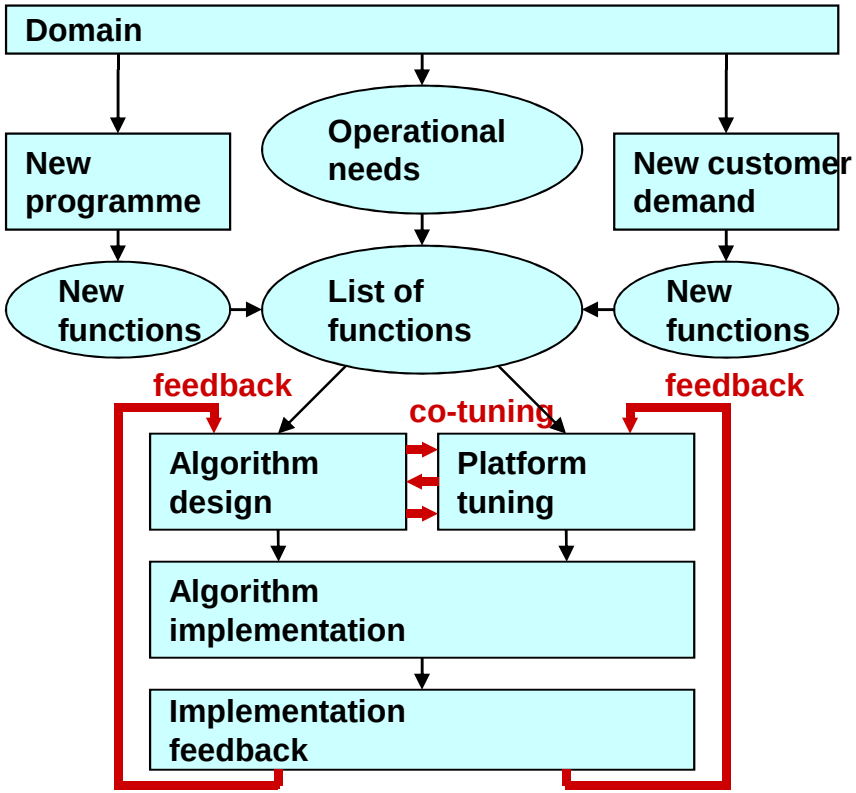
Programme Driven Process

Top-down design + independent programmes



Platform Based Process

Algorithms and platform co-design and tuning



Requirements for Platform based Simulation

1. **Need efficient architecture experimentation, simulation, analysis framework.**
   - Component Integration of big IP blocks is cumbersome.
   - Changing RTL is very time consuming and not desirable

2. **Support IP variety in SoCs**
   - Want to leverage existing C++, VHDL simulations
     - Need wrapper & glue
   - Simulation needs a “global/unifying” simulation queue.
   - IP may be coming from external IP vendors

3. **Desire to run “real SW” on simulations**
   - More than trace driven simulations!
   - Complete Operating systems
   - Driver and App development in advance of real silicon.

Simulation Taxonomy

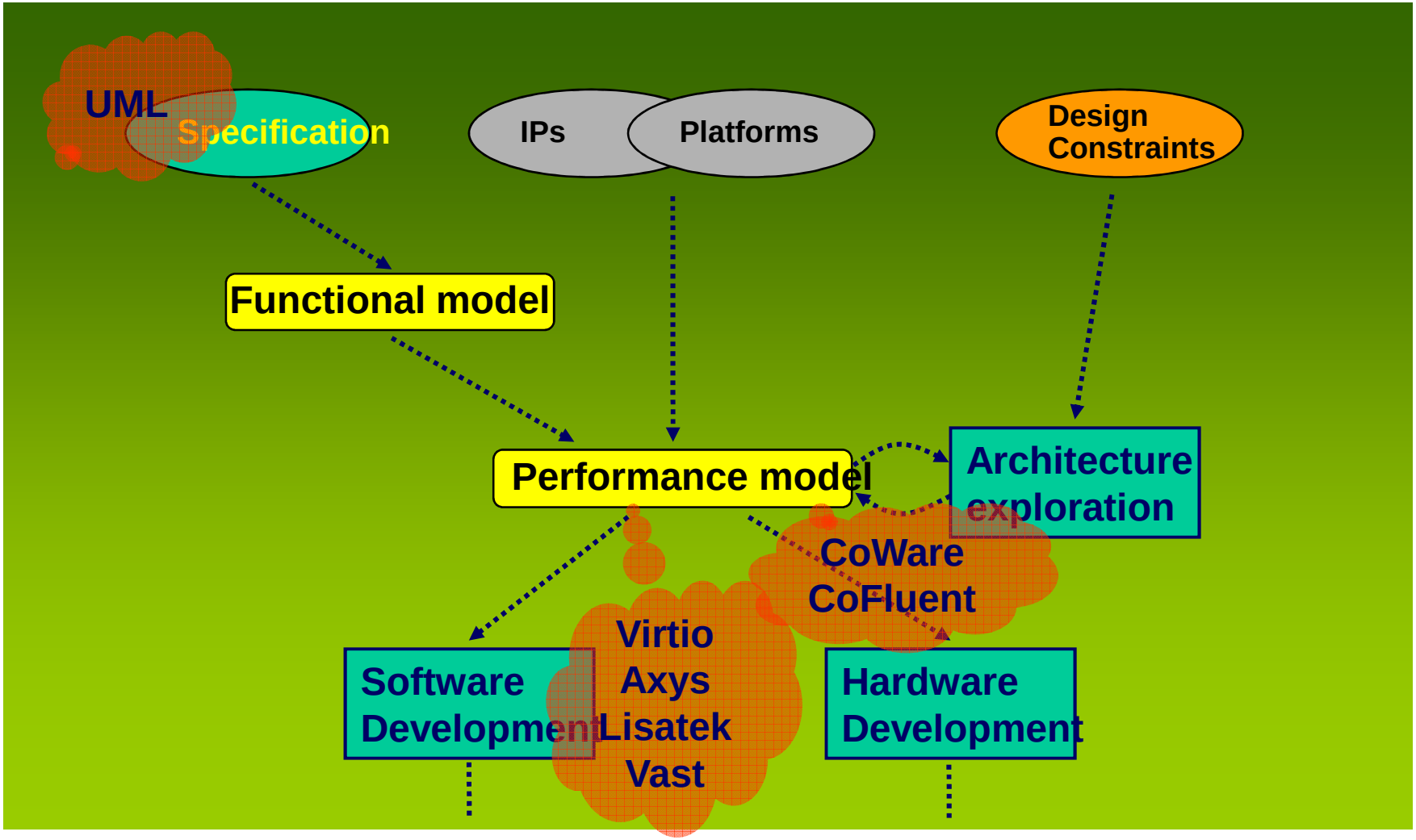
Types of Simulations

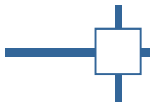
- System Architectural
- System Performance
- Functional Model
- Transaction Level Model (TLM)
- Behavior Synthesis Model
- Register Transfer Level model (RTL)
- Gate Level

Simulations Qualifiers

- UnTimed Functional
- Timed Functional
- Bus Cycle Accurate
- Pin Cycle Accurate
- Register Transfer Accurate

Industry landscape (an incomplete map..)





Open System C Initiative (OSCI)

- Open source libraries and reference runtime.

Commercial tools from Synopsis, CoWare, Frontier, etc.

SystemC is:

- C++, Class libraries, Run time simulation system
- Provides:
  - Simulation queue and time based events
  - Concurrency models
  - HW abstractions (Modules, ports, buses)

Why use SystemC?

Use SystemC to create silicon IP simulations

Can model an IP blocks

- RTL level to SoC level

Component composition “glue”

- Enable easy architectural experimentation and analysis

Integrates heterogeneous solutions – can wrap existing VHDL or C++ simulations.

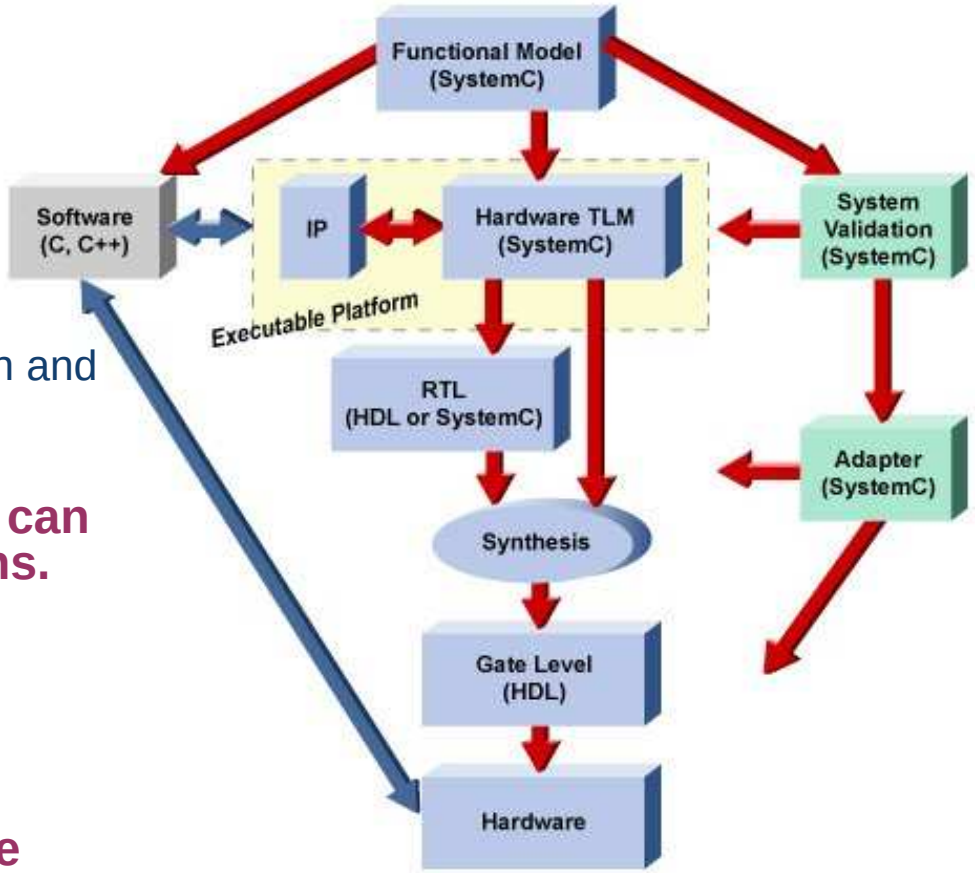
- Very effective for SoCs

Reference model for RTL

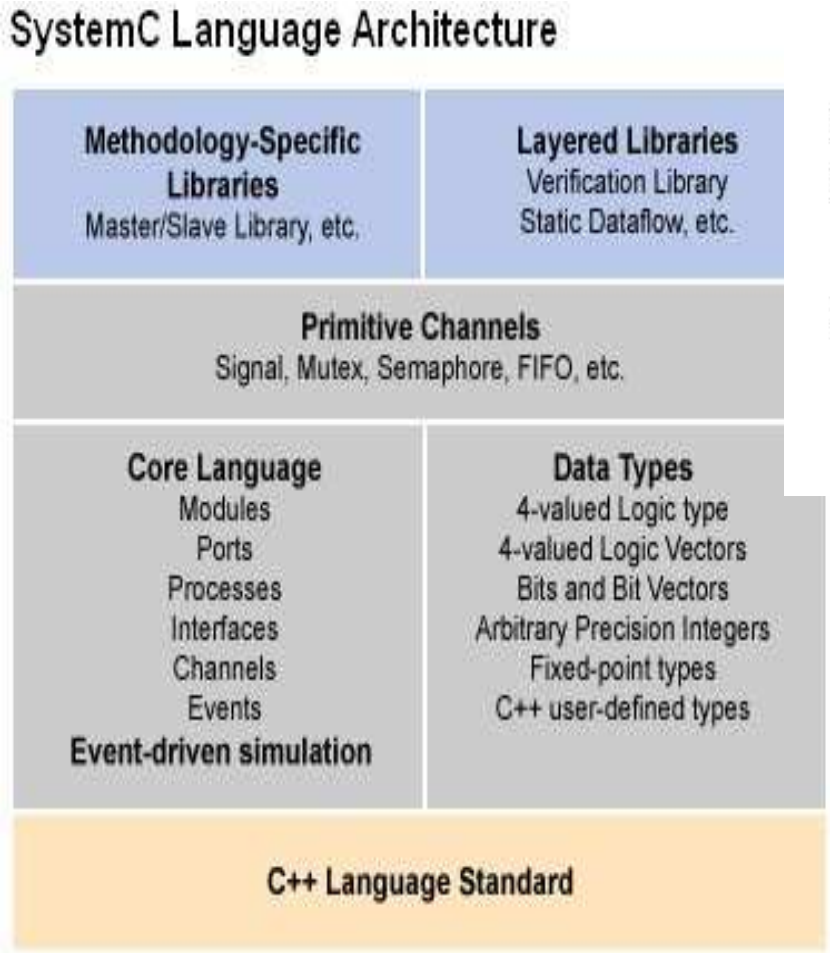
Performance! –very effective at coarse grained simulation

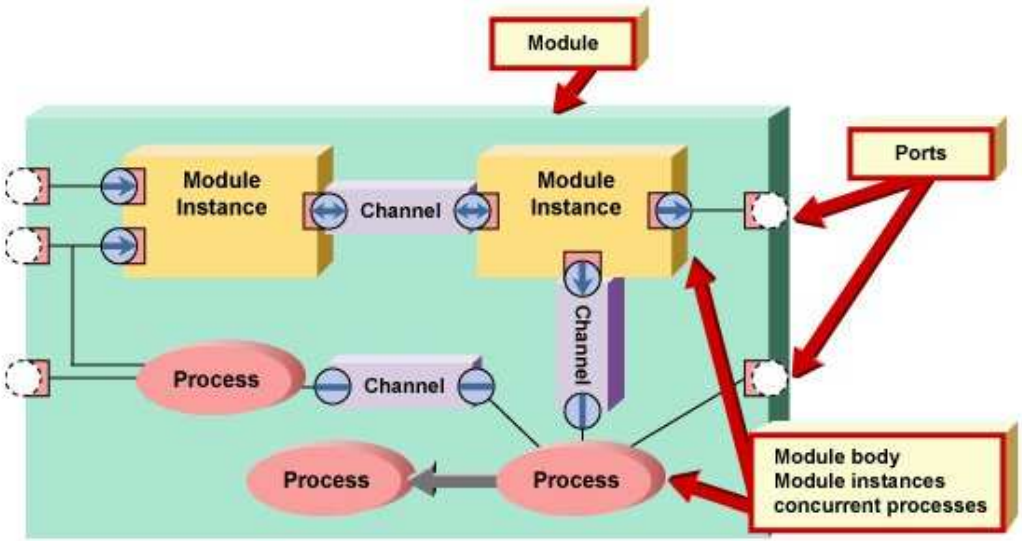
Enable advance SW development and analysis before HW.

Tool vendors provide SystemC based analysis hooks



SystemC Modeling





- **Simulations are built from SystemC classes: *Modules*, *Ports*, *Channels***
- ***Processes* describe module functionality…really just methods.**
- **Class are "simulation aware"**
- **Class Macros provided as cheats for EE folks: SC_MODULE()**

Transaction Level Modeling (TLM)

It's a simulation methodology.

- Dictates event granularity, standardized interfaces

Simulation has no "clock edge"

- *Discrete events consume X time.*
- Accurate, yet fast enough to execute real SW.

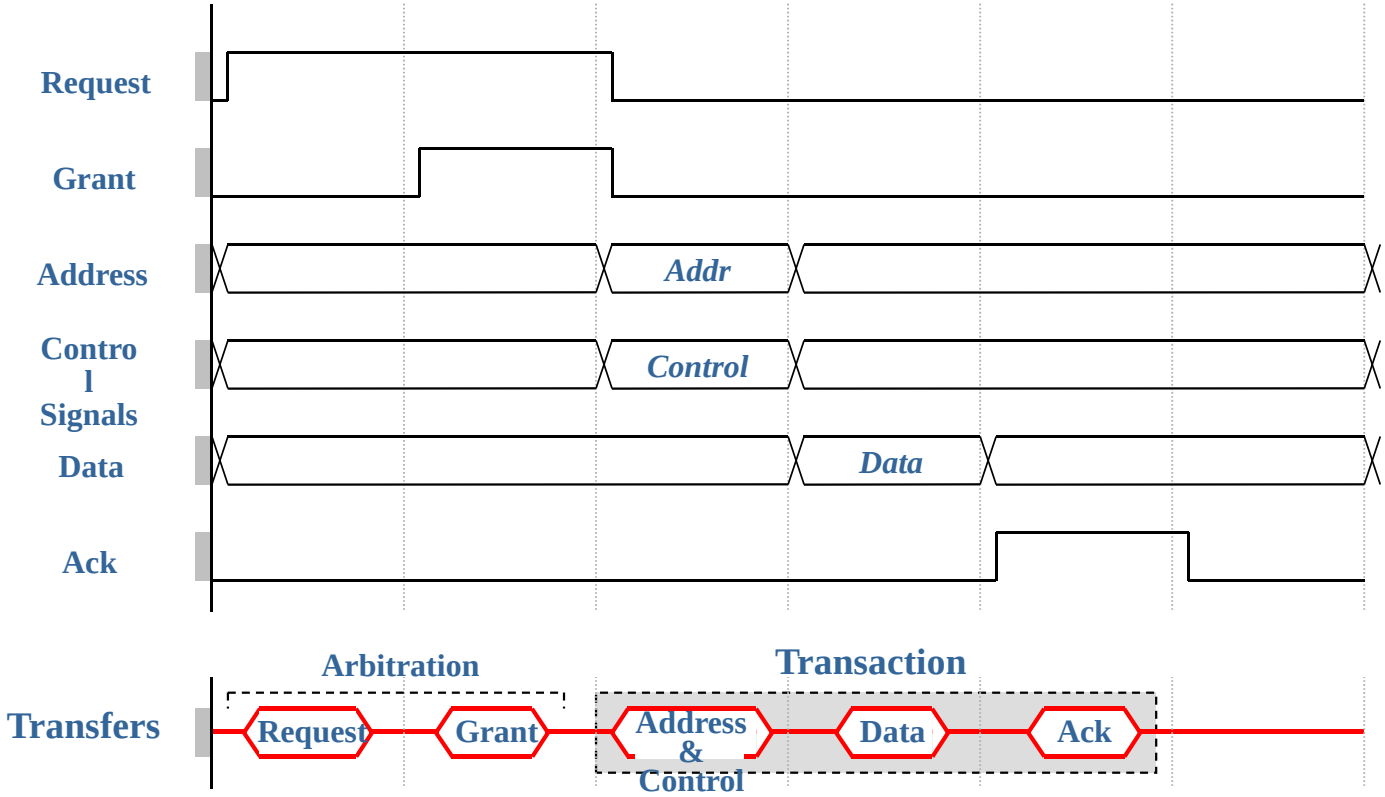
"The primary goal of Transaction Level Modeling is to achieve dramatically increased simulation speeds, while still offering enough accuracy for determining hardware response times."

- **High speed simulation**
- **Cycle accuracy**
- **Reduce detail & simplify modeling**
- **Handle complex bus topologies**
- **Support HDL Co-Simulation**

TLM: Reducing Communication Detail

Transfers are used to reduce communication detail to a small number of events. A Transaction refers to the data-exchange transfers *(It excludes the arbitration transfers).*

Transfers are used to consolidate signal handshakes



Summary

- V-Cycle and separation of concerns are typically used for system development to reduce risk & cost and improve quality & communication between stakeholders.
- System modelling should be done at the highest possible abstraction to integrate large IP sw&hw building blocks
- Platform-based design enable cost reduction and reuse of hardware and software components
- A combination of top down (from apps) and bottom up (from architecture space) enables optimal solution
- HW/SW co-design with constant feedback between sw and hw architects is required for a power/perf optimized system
- Fast simulation (e.g. SystemC) of power and performance is needed for sw development and platform refinements.