Research Issues in Dependable Embedded Systems

Hermann Kopetz

January 2002

Outline

- ♦ Introduction
- ♦ Technology Drivers
- ♦ Important Research Problems
- ♦ Conclusion

Embedded Systems

- ♦ Are an enabling technology for many important industrial products
  - Automotive
  - Mobile Phones
  - Machinery
  - Consumer Electronics
  - Smart Cards
  - . . . .
- ♦ The macro-economic impact of embedded systems technology is much wider than the size of the embedded system market indicates.
- ♦ More than 99% of the computers produced worldwide are deployed in embedded systems
- ♦ Fierce battles about the world-wide market share are ongoing
- ♦ In many applications, dependability is a key selling argument (e.g.cars).

The Context of an (Embedded) System

Every system is embedded in a technological, economical, and sociological context. This context

- ♦ Determines the System Requirements
  - Functional
  - Economical
  - . . . . .
- ♦ Changes rapidly
  - New Applications have new dependability requirements
  - Moore’s Law changes the economic constraints
  - User experience and changing expectations
  - . . . .

We must try to understand the future context in order to pin down the future challenges.

Why Do Embedded Systems Fail?

- **Independant (Internal) Physical Faults:** E.g., a physical aging process. Can be transient (soft) or permanent. *Multiple failures of chips, but not within chips, are statistically independent*--will increase due to reduction of feature size.
- **Dependant (External) Physical Faults:** E.g., EMC, spikes in the power supply, mechanical shock. Can be transient or permanent. *Replication of components is not the solution.*
- **Design Faults:** The cause of the failure is the design (software or hardware) resulting in inconsistent states and actions. Reduce cognitive complexity of designs.
- **Malicious Attacks:** An evil adversary attacks the system.
- **Operator Error:** Mistakes of the operator at the MMI.

Technology Driver: Moore’s Law

What You Can Do Today with 1 mm^2 of Silicon?

- ♦ Build a 32 bit wide processor (e.g., the ARM 7 processor)
- ♦ implement 100 k-bytes of memory (e.g., the 256 Mbit memory chip from Infineon is less than 100 mm^2).

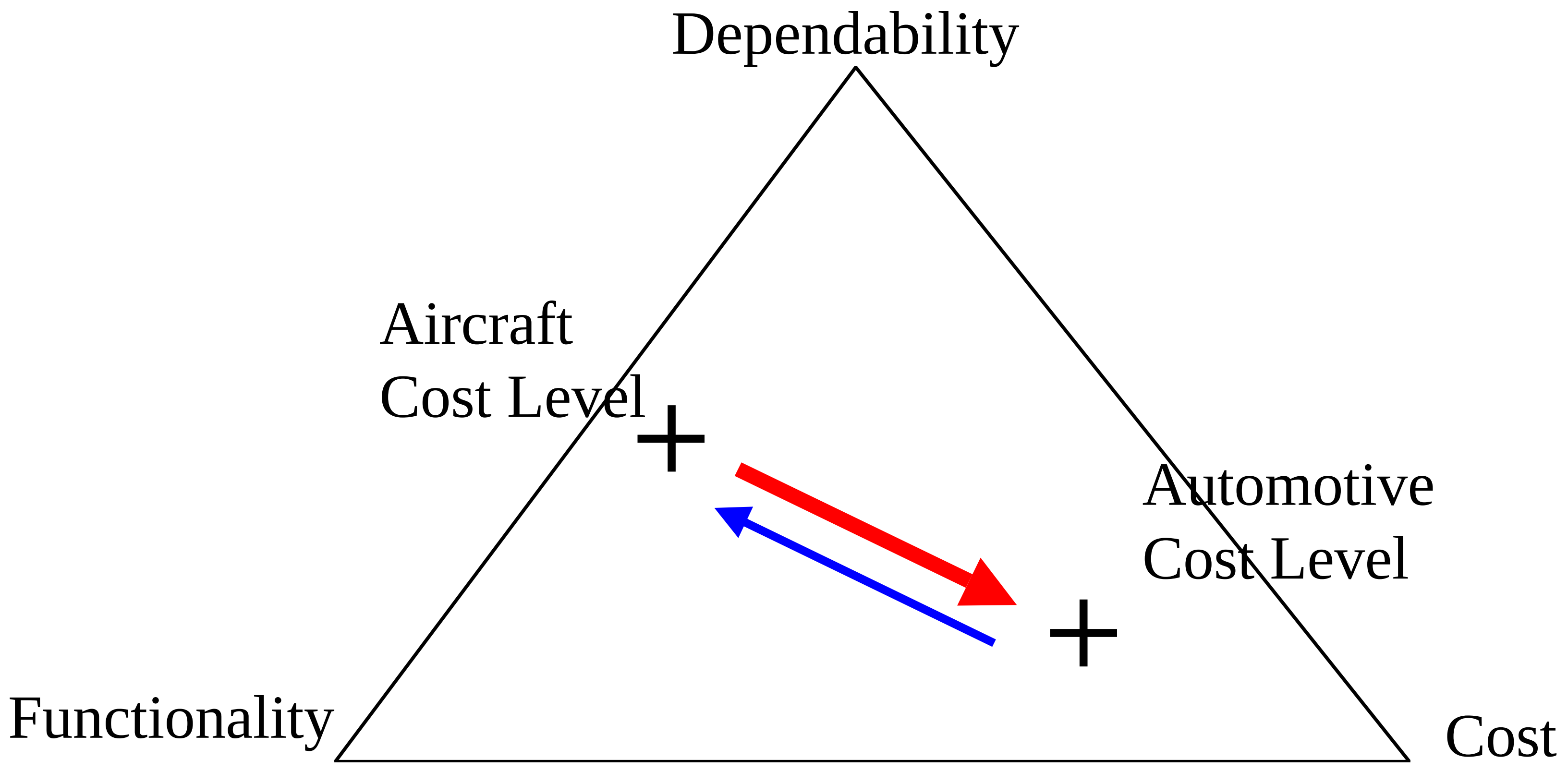
Today, the marginal production cost (without IP, packaging,etc.) of 1 mm2 of silicon is in the order of 10 US cent.

Communication capabilities increase even faster than processing capabilities.

Consequences of Moore’s Law

- Number of embedded control systems will increase significantly.
- Hardware-fault tolerance becomes affordable, even in mass-markets.
- In many cases, system hardware cost will be dominated more by the number of packages, than by the functionality of the silicon real-estate in each package.
- The use of the smart sensor technology will increase. Sensor nodes, built with mixed signal chips, will be (intelligent) nodes of a distributed system.
- Because of the decreasing feature size, the occurrence of multiple transient hardware faults will increase.
- It is not possible to have two independent fault-containment regions on the same chip.
- Distributed systems will prevail (also for reasons of fault containment).

The Challenge Problem in Embedded Systems. . .



to bring dependable embedded systems to the cost level of mass market applications

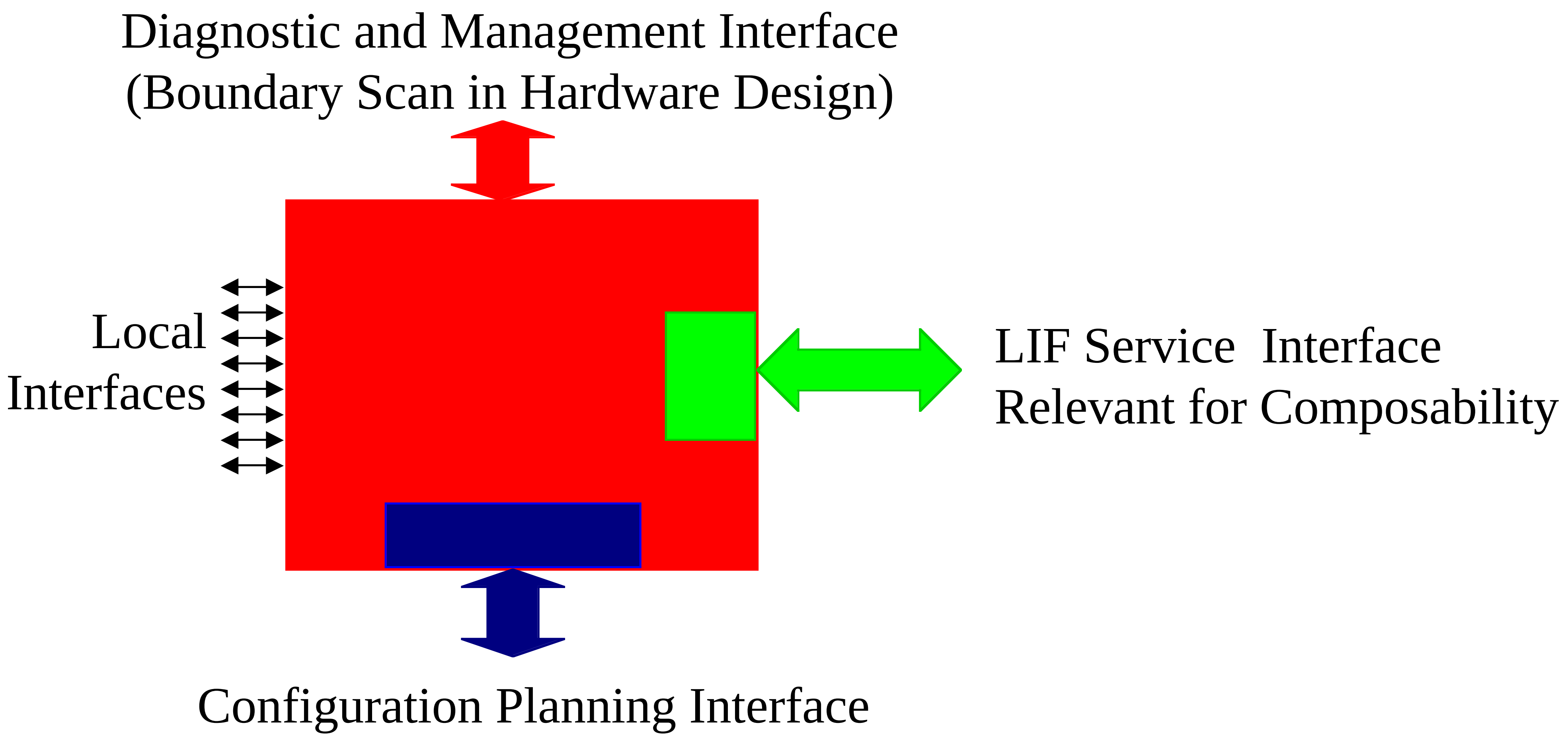
Important Research Problems

- ♦ Composability
- ♦ Secure Real-Time Systems
- ♦ Transparent Fault-Tolerance
- ♦ Certification of High-Dependability Applications
- ♦ Domain-Specific Architectures

Composability

- ♦ Precise (formal) specification of linking interfaces (time, value) of components
- ♦ State encapsulation, as viewed from a LIF
- ♦ Research into the cognitive complexity of interfaces.
- ♦ Reasoning about the composition of systems on the basis of the interface specification.
- ♦ Independent validation of component interface properties (time, value).
- ♦ Integration of legacy systems (Wrapper Design).
- ♦ Interface Standardization.

The Three Interfaces of a Component


Diagnostic and Management Interface
(Boundary Scan in Hardware Design)
Local
Interfaces
LIF Service Interface
Relevant for Composability
Configuration Planning Interface

Secure Real-Time Systems

Whereas in the past, low-level control software was mostly in ROM, recent technology-developments (flash memory) makes it possible to down-load control software remotely

- ♦ Secure fault diagnosis and maintenance, e.g., remote downloading of software into the flash memory of a car.
- ♦ The provision of the proper level of security in mass-market systems that are maintained by “non-trustable” institution.
- ♦ Security of normadic systems connected by wireless protocols.
- ♦ Security in dynamically reconfigurable RT systems.

Transparent Fault-Tolerance

- Provision of a generic fault-tolerance layer, independent of the application
- Tolerance w.r.t.*arbitrary* failure modes of components (VLSI chips)
- Generic correctness argument for the fault-tolerance function
- On-line maintenance of fault-tolerant systems
- Autonomous Reconfiguration
- Low Power

Certification of High-Dependability Applications

- Modular certification of a composable design
- Validation of ultra-high dependability
- Proof of absence of catastrophic failure modes
- Formal correctness proof of architecture claims
- Closing the gap between formal verification of a property (within a model) and its implementation
- Worst-case Execution time (WCET) research (hardware, algorithms, tools)

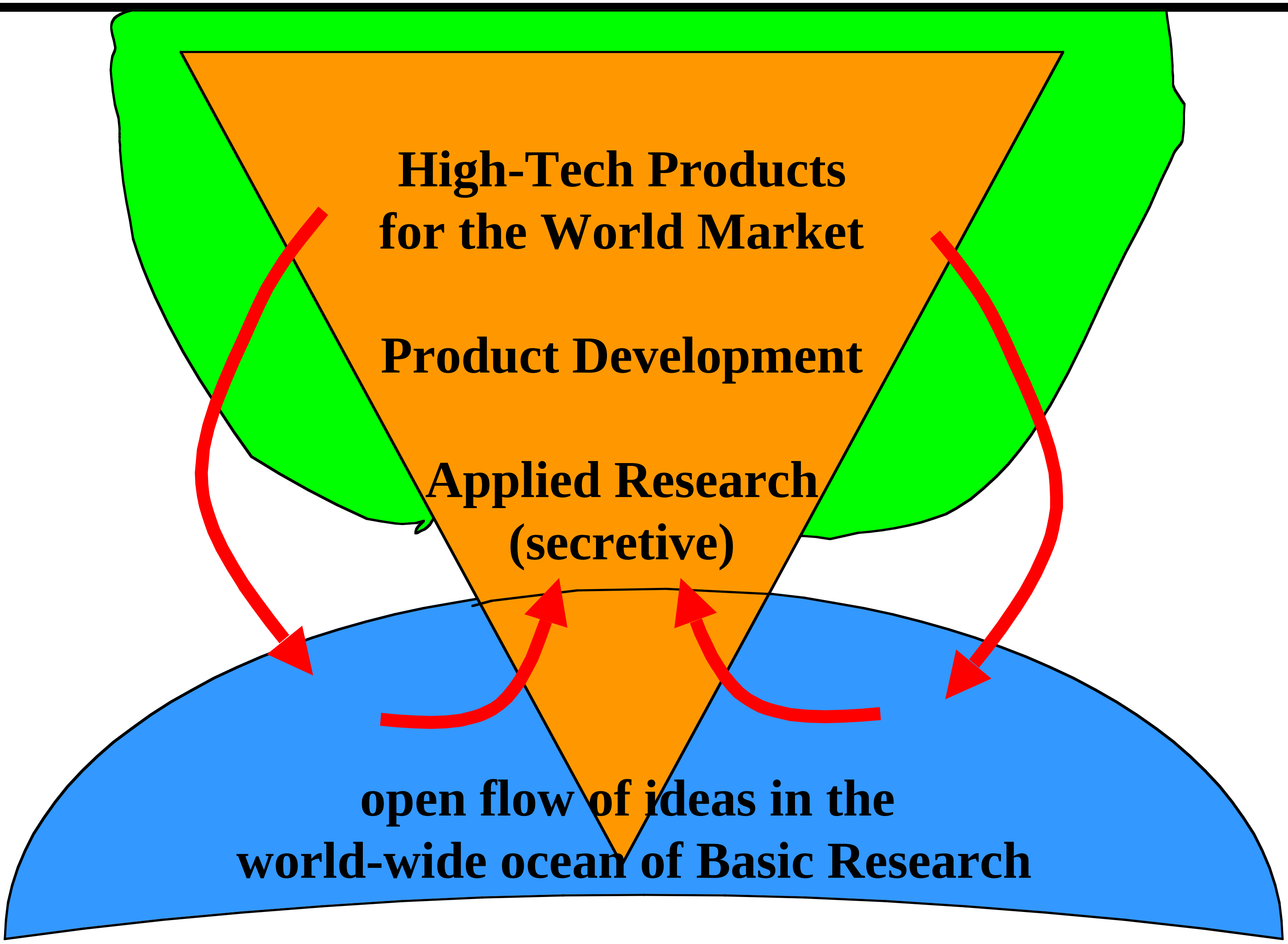
Domain-Specific Architectures

An architecture provides a framework for the implementation of applications in a particular domain. It provides the computational infrastructure.

The key challenge concerns finding abstractions that are specific enough in order to support strong claims that can be certified, but are still general enough to apply to a significant application domain.

- What are the generic certified services that should be provided by an architecture (e.g., clock synchronization, membership, . . .)
- Validation of the architecture claims by diverse means (formal, experimental, field experience, . . .)
- Design processes and tool support within an architecture context.

Holistic Approach



Conclusions

- ♦ A balanced combination of conceptual (theoretical) and experimental research within a project is required. The experimental research will consume the major part of the resources.
- ♦ New concepts and architectures must be implemented and experimentally evaluated
  - Design a complete system
  - Build a prototype with real hardware and software and compare its performance (and cost) to competing alternatives.
  - Evaluate the prototype experimentally (e.g., by fault-injection)
- ♦ Strong involvement of researchers in standardization bodies.

Credibility with respect to industry requires arguments substantiated by experimental evidence.