Is Embedded Software for Safety Critical Automotive Systems Really a Software Problem?

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

Co-founder, Chief Technology Advisor and Member of the Board
Cadence Design Systems

SanDiego 2006
Outline

• Electronics in Automotive: Design Chain Transformation Challenges
• Design Methodology and Autosar
• Future Use of Advanced Electronics Concepts: Wireless Sensor Networks
• System Level Design and Architecture Exploration
Electronics, Controls & Software

Shifting the Basis of Competition in Vehicles

- More functions & features
- Less hardware
- Faster

Potential inflection point. Now!

Value from Electronics & Software

- Software $ 13%
- Electronics $ 9%
- Other $ 2%

Mechanical $

Electric Ignition
Electric Fan
Fuel Cell
Wheel Motor
Hybrid PT
OnStar
Rear Vision
ACC
Rear aud/vid
Side Airbags
Passive Entry
Head Airbags
ABS
TCC
EGR
Electric Brake
Electric Fan
... 

1970s 1980s 1990s 2000s 2010s 2020s

System Connection

Subsystem Controls & Features

Forefront of Innovation

Source: Matt Tsien, GM

Copyright: A. Sangiovanni-Vincentelli
Opportunity: Electronic Systems Design Chain

Design Science

System Design

Implementation

IP

Fabrics

Manufacturing

Interfaces
Automotive Supply Chain: Car Manufacturers

- Product Specification & Architecture Definition (e.g., determination of Protocols and Communication)
- System Partitioning and Subsystem Specification
- Critical Software Development
- System Integration
Electronics for the Car: A Distributed System

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

• “Today 30% of the cost of a my top-of-the-line car is in electronics” (W. Reitzle, BMW, 2000)

• “90% of the innovation in the car will be in electronics” (Daimler-Chrysler Technology Conference, Stuttgart, 2000)

• “The industry is fighting to solve problems that are coming from electronics and companies that introduce new technologies face additional risks. We have experienced blackouts on our cockpit management and navigation command system and there have been problems with telephone connections and seat heating.” (J.Hubbert, Daimler-Chrysler, June 2003)

• “The cost of integration is skyrocketing especially for high-end cars. Model introduction is delayed to fix integration problems.”

• “We need to take control of integration and restrict the subsystem footprint”
Automotive Supply Chain:
Tier 1 Subsystem Providers

1. Transmission ECU
2. Actuation group
3. Engine ECU
4. DBW
5. Active shift display
6/7. Up/Down buttons
8. City mode button
9. Up/Down lever
10. Accelerator pedal position sensor
11. Brake switch

- Subsystem Partitioning
- Subsystem Integration
- Software Design: Control Algorithms, Data Processing
- Physical Implementation and Production

Copyright: A. Sangiovanni-Vincentelli
Notable Quotes

• “We have more than 500 calibration parameters in our subsystems and the number is raising”

• “The software productivity in our industry is about 10 lines per day per software designer and the residual defects are about 1,000 ppm”

• “We need to be able to leverage the advances of IC technology quickly and safely”

• “I wish we could enlarge the footprint of our subsystems to optimize the solution better!”
## Complexity, Quality, Time-to-Market: TODAY

<table>
<thead>
<tr>
<th></th>
<th>PWT UNIT</th>
<th>BODY GATEWAY</th>
<th>INSTRUMENT CLUSTER</th>
<th>TELEMATIC UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MEMORY</strong></td>
<td>256 KB</td>
<td>128 KB</td>
<td>184 KB</td>
<td>8 MB</td>
</tr>
<tr>
<td><strong>LINES OF CODE</strong></td>
<td>50,000</td>
<td>30,000</td>
<td>45,000</td>
<td>300,000</td>
</tr>
<tr>
<td><strong>PRODUCTIVITY</strong></td>
<td>6 LINES/DAY</td>
<td>10 LINES/DAY</td>
<td>6 LINES/DAY</td>
<td>10 LINES/DAY*</td>
</tr>
<tr>
<td><strong>RESIDUAL DEFECT RATE @ END OF DEV</strong></td>
<td>3000 PPM</td>
<td>2500 PPM</td>
<td>2000 PPM</td>
<td>1000 PPM</td>
</tr>
<tr>
<td><strong>CHANGING RATE</strong></td>
<td>3 YEARS</td>
<td>2 YEARS</td>
<td>1 YEAR</td>
<td>&lt; 1 YEAR</td>
</tr>
<tr>
<td><strong>DEV. EFFORT</strong></td>
<td>40 MAN-YEAR</td>
<td>12 MAN-YEAR</td>
<td>30 MAN-YEAR</td>
<td>200 MAN-YEAR</td>
</tr>
<tr>
<td><strong>VALIDATION TIME</strong></td>
<td>5 MONTHS</td>
<td>1 MONTH</td>
<td>2 MONTHS</td>
<td>2 MONTHS</td>
</tr>
<tr>
<td><strong>TIME TO MARKET</strong></td>
<td>24 MONTHS</td>
<td>18 MONTHS</td>
<td>12 MONTHS</td>
<td>&lt; 12 MONTHS</td>
</tr>
</tbody>
</table>

* C++ CODE

**FABIO ROMEO, Magneti-Marelli**
Design Automation Conference, Las Vegas, June 20th, 2001
Embedded Software?
Notable Quotes

• The Nihon Keizai Shimbun reported that Japan’s Ministry of Economy, Trade and Industry that Japanese companies spend more than 100 billion yen (USD $903 million) per year developing automotive-related software. And it isn’t going to get any cheaper, with some analysts estimating costs escalating to 1 trillion yen (USD $9.1 billion) by 2014, according to the daily newspaper.

• So is the industry ultimately moving toward ‘plug-and-play’? Taking the idea of multiplexing to its logical extreme, a carmaker could potentially wait until relatively late in the vehicle’s development cycle before committing to specific electronic hardware yet avoid having to delay - or worse, tear up - its electrical architecture in the last minute.
Real-Time Multitasking: Plug and Play?

Plug and Pray!
How Safe is Our Real-Time Software?
Embedded Software Architecture Today
Outline

• Electronics in Automotive: Design Chain Transformation Challenges

• **Design Methodology and AUTOSAR**

• Future Use of Advanced Electronics Concepts: Wireless Sensor Networks

• System Level Design and Architecture Exploration
The Intellectual Agenda

• To create a modern computational systems science and systems design practice with
  – Concurrency
  – Composability
  – Time
  – Hierarchy
  – Heterogeneity
  – Resource constraints
  – Verifiability
  – Understandability
The Platform Concept

- 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!
Platform-based Design
(ASV Triangles 1998)

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

Intercom Platform (BWRC, 2001)
Separation of Concerns (1990)

Development Process
- Analysis
- Specification
- Implementation
- Calibration
- After Sales Service

Behavior Components
- C-Code
- Matlab
- ASCET

Virtual Architectural Components
- CPUs
- Buses
- Operating Systems

System Behavior
- Mapping
- Performance Analysis
- Refinement

System Platform
- ECU-1
- ECU-2
- ECU-3
- Bus

Evaluation of Architectural and Partitioning Alternatives

IPs
- Analysis
- Specification
- Implementation
- Calibration
- After Sales Service

Copyright: A. Sangiovanni-Vincentelli
A Software-Centric View of Platforms (1998)

- **Input devices**
- **Output devices**
- **Hardware Platform**
- **Software Platform**
- **Application Software**
- **Platform API**
- **RTOS**
- **BIOS**
- **Device Drivers**
- **Network Communication**
- **Compiler**

Network communication and I/O connections are depicted with arrows and labels.
Integration approach is a **Central Platform** for software-based applications (vehicle functions)

The subsystems have an **open** software architecture with the following features
- Extendable and Updateable
- Easy integration of new functions
AUTOSAR: a Disruptive Event?

The AUTOSAR project goals will be met by specifying and standardizing the central architectural elements across functional domains, allowing industry competition to focus on implementation.

### Project Objectives
- Consideration of availability and safety requirements
- Redundancy activation
- Scalability to different vehicle and platform variants
- Implementation and standardization of basic system functions as an OEM wide “Standard Core” solution
- Transferability of functions throughout network
- Integration of functional modules from multiple suppliers
- Maintainability throughout the whole “Product Life Cycle”
- Increased use of “Commercial off the shelf hardware”
- Software updates and upgrades over vehicle lifetime

### Functional Domains
- Engine
- Chassis
- Multimedia/Telematics
- Safety (active/passive)
- Man Machine Interface
- Passenger “centric”
- Body/Comfort

Cooperate on standards, compete on implementation
To achieve the objectives, AUTOSAR has first to address the main topics: basic software, functional APIs, and software integration.

<table>
<thead>
<tr>
<th>Project Objectives</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consideration of availability and safety requirements</td>
<td>Basic Software</td>
</tr>
<tr>
<td>Redundancy activation</td>
<td>Functional APIs</td>
</tr>
<tr>
<td>Scalability to different vehicle and platform variants</td>
<td>Methods of Software Integration</td>
</tr>
<tr>
<td>Implementation and standardization of basic system</td>
<td></td>
</tr>
<tr>
<td>functions as an OEM wide “Standard Core“ solution</td>
<td></td>
</tr>
<tr>
<td>Transferability of functions throughout network</td>
<td></td>
</tr>
<tr>
<td>Integration of functional modules from multiple suppliers</td>
<td></td>
</tr>
<tr>
<td>Maintainability throughout the whole “Product Life Cycle“</td>
<td></td>
</tr>
<tr>
<td>Increased use of “Commercial off the shelf hardware“</td>
<td></td>
</tr>
<tr>
<td>Software updates and upgrades over vehicle lifetime</td>
<td></td>
</tr>
</tbody>
</table>
Business Implications

The realization of an AUTOSAR industry standard will provide significant benefits for OEMs, leading suppliers as well as for tool providers and new market entrants.

- **OEM**
  - OEM overlapping reuse of software modules which are irrelevant to competition
  - Functions of competitive nature can be developed separately
  - Focus on innovation/ functions
  - Standardized certification

- **Supplier**
  - Reduction of version proliferation
  - Development partitioning among suppliers
  - Increase of efficiency in functional development
  - New business models possible

- **Tool provider**
  - Common interfaces with development processes
  - Seamless, manageable, task optimized (time dependent) tool landscape

- **New market entrant**
  - Transparent and defined interfaces enable new business models

An industry standard is established.
Implications?

The business and technical relationships among players in the automotive electronics chain are likely to change substantially as:

- OEMs take more control of the electronics content of the car,
- Tier 1 suppliers will have to interact and exchange IPs with their peers,
- Tier 2 suppliers will have to interact more tightly with their customers.
- All suppliers will face commoditization of their business.
- EDA suppliers will have to integrate their offerings.
Standardized interfaces support HW independence, an effective transferability of functions and redundancy activation.

- **Automotive Open System Architecture (AUTOSAR)**
  - Specification of interfaces
  - Standardized, openly disclosed interfaces
  - HW-independent SW-layer
  - Transferability of functions
  - Redundancy activation

- **System Architecture**
  - Describes the system structure regarding
    - functional interaction and integration of the system elements
    - interfaces to other systems
    - system environment and flow of data
    - Data and SW architecture

- **System**
  - Combination of elements designed to fulfill a function

- **Element**
  - e.g. Sensors, Processor Nodes, Networks, Actuators etc.

1) HW: Hardware
2) SW: Software
Embedded Software Architecture Tomorrow?
What about “real time”? 

Make it faster!
Outline

• Electronics in Automotive: Design Chain Transformation Challenges
• Design Methodology and Autosar
• Future Use of Advanced Electronics Concepts: Wireless Sensor Networks
• System Level Design and Architecture Exploration
Bell’s Law: A New Computer Class Every 10 Years

Meaning in the Device

Meaning in the Connection

Meaning in the Collection

Year

1940’s

2000’s

log (people per computer)

1940’s

2000’s

Bell’s Law: A New Computer Class Every 10 Years

Meaning in the Device

Meaning in the Connection

Meaning in the Collection

1940’s

2000’s

log (people per computer)
Wireless Sensor Networks


Berkeley Dust Mote\(^1\)  

\(^1\)From Pister et al., Berkeley Smart Dust Project
Creating a Whole New World of Applications

From Monitoring

To Automation
Enabled by Combined Technology Advancements

Moore’s law and size

Moore’s law and cost

Ubiquitous wireless as the glue

True system integration
Automotive Electronics: Driver Assistance

Product and Technology Overview

**Ultra-Sonic**
- Standard Parking
- Parking Space Measurement
- Semi-autonomous Parking Assistant

**Long Range Radar**
- ACC > 30km/h
- ACC plus 0 to ~200km/h
- Predictive Safety Systems (PSS)

**Intelligent Vision System**
- Night Vision Support
- Lane Departure Warning
- Video Park Pilot

Copyright: A. Sangiovanni-Vincentelli
Tire to Vehicle

SW Code

Rx/Tx Antenna

Sensing Device

RF Link

Computing

Power Management

Energy Scavenging

Smart antenna

Stability Control System

Body computer

Copyright: A. Sangiovanni-Vincentelli
Outline

• Electronics in Automotive: Design Chain Transformation Challenges
• Design Methodology and Autosar
• Future Use of Advanced Electronics Concepts: Wireless Sensor Networks
• System Level Design and Architecture Exploration
Putting it all together....

• We need an integration platform
  – To deal with heterogeneity:
    – Where we can deal with Hardware and Software
    – Where we can mix digital and analog
    – Where we can assemble internal and external IPs
    – Where we can work at different levels of abstraction
  – To handle the design chain
  – To support integration
    – e.g. tool integration
    – e.g. IP integration

• The integration platform must subsume the traditional design flow, rather than displacing it
Metropolis: An Integrated Electronic System Design Environment

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

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

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

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

DESIGN OVERVIEW

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

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

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

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

Industry

- Cadence Berkeley Labs (Design Methods and Tool Development)
- Intel (Communication subsystem of the Centrino platform, Imaging-Video subsystems, Mixed Analog-RF-Digital)
- GM (distributed architecture)
- Infineon (Software defined radios)
- Nokia (Platform-based cell phone design)
- Cypress (Network processor)
- ST (Automotive design)
- Magneti Marelli (Distributed subsystems)
- United Technology (Security, Conditioning and Elevator Systems)
- Xilinx (Virtex II front-end)

Universities

- University of California at Berkeley
- Carnegie Mellon
- MIT
- UCLA
- Politecnico di Torino
- Universita’ di Pisa
- UC Riverside
- Universitat Politecnica de Catalunya

Others

- BWRC
- PARADES
Metropolis Framework (Berkeley)

- Function Specification (Operational)
- Design Constraints & Assertions (Denotational)
- Architecture (Platform) Specification (Operational)

Metropolis Infrastructure
- Design methodology
- Meta model of computation
- Base tools
  - Design imports
  - Meta model compiler
  - Simulation

Synthesis/Refinement
- Compile-time scheduling of concurrency
- Communication-driven hardware synthesis
- Protocol interface generation

Analysis/Verification
- Static timing analysis of reactive systems
- Invariant analysis of sequential programs
- Refinement verification
- Formal verification of embedded software
Selecting a Platform

Selecting a platform implies searching a large design space trying to optimize multiple objectives

- Cost
- Performance
- Flexibility
- Reliability
- Size
- Technology
- Power
- Reuse
Quantitative “What if” Analysis Framework

- Constraints
- Degrees Of Freedom
- Logical/Physical Architectures
- Architecture Exploration
- Metrics- Based Quantitative Analysis
- Metrics- Scored Architectures
- Done?
- End
Key Issues for the OEM automotive industry

- Commonize as much as possible electronic platforms
- Include fail-safe, fail-soft issues
- Optimization and integration
- Robustness to changes
- Need for a virtual integration environment that allows the architect to take advantage of the architectural degrees of freedom and efficiently analyze the impact of the changes.
Strategy for Commonization

• Potential areas of commonization
  – Process
    – Development and deployment
  – Architectures
    – Functional architecture
    – Subsystem architectures
    – Hardware architecture
    – Software architecture
  – Components
    – ECU components
    – Software components
    – Sensor/Actuator components
Architecture Exploration

- Constraints
  - Business
  - Technical
- Metrics
  - Quantitative
  - Qualitative
- Optimization criteria’s
  - Automotive design attributes
    - Cost, fault tolerance, reusability…
Architecture Development: Constraints

- Government standards
  - Emission diagnostic requirements dictate bus topology
- Supplier hardware constraints
  - Sensor/Actuator does not meet GM bus timing specifications
- Technical constraints
  - Flash programming and OBD II diagnostic
    - Must have direct bus connection
  - Back and front compatibility
  - Non functional constraints
    - Temporal
    - Fault tolerance
    - Synchronization
- VDP (Vehicle Development Process) risk analysis
  - Ambiguous features
  - Time to market
Architecture Exploration: Centralized vs. Decentralized

• Architectural Degrees of freedom
  – Serial data topology
  – Mapping of function to ECU’s
  – Sensor/actuator (IO) allocation
    – Decentralized IO (IO accessed through the bus) vs. Centralized IO (IO accessed directly on an ECU)
  – Computing infrastructure
    – Decentralized computing (Many ECU’s) vs. centralized computing (few ECU’s)

• Metrics
  – Control Latency
    – end-to-end latency from sensors to actuators
  – Geometric Attributes
    – total wire length
    – number of cut leads*
  – Serial Data Metrics
    – bus Utilization
  – Flexibility
    – degree of architecture resilience to specific future design changes (scenarios)
Architecture Exploration: Centralized vs. Decentralized

• Functional Architecture
  – a supervisory control system that coordinates the steering, braking and suspension systems, and an electro-chromic window control system.
Architecture Exploration: Centralized vs. Decentralized

• Architecture Models
  – Mixed Decentralized IO
  Centralized Computing (MDICC)
    – Hand wheel sensors and wheel speed sensors connected onto the serial data bus and a partially centralized computing structure with the supervisor and brake features allocated to one module
Architecture Exploration: Centralized vs. Decentralized

• Architecture Models
  – Centralized IO Centralized Computing (CICC)
Architecture Exploration: Centralized vs. Decentralized

- Architecture Models
  - Centralized IO Decentralized Computing (CIDC)
Architecture Exploration: Centralized vs. Decentralized

• Architecture Models
  – Decentralized IO Decentralized Computing (DIDC)
Example Implementation – Steer-by-wire (SBW)

- Vehicle-level requirements derived from a concept vehicle
  - Rich example with hierarchy, safety critical requirements and interactions with supplier subsystems (brake by wire, power coordinator)
  - Most safety critical and most complex within the x-by-wire domain
Highlights of Architecture Exploration on an Industrial Example (Details are confidential)

- Architecture exploration on Steer-by-wire subsystem
- 36 different architectures explored within a 1.5 month timeframe with 3 part time people
- 5 different metrics considered in qualitative and quantitative means
- Assume larger degree of freedom
  - Independence between software and hardware
Summary and Perspectives

• Electronic Automotive Industry facing an array of complex problems from design to manufacturing involving complexity, power, reliability, reconfigurability, embedded software

• Design Methods and Tools lacking: active research field

• Investment needed from IC and Automotive System industry otherwise the situation is bound to become more critical. Not an issue of languages or point tools

• Wireless Sensor Networks open a new perspective in the field