A New Approach to Design

- Formal Specifications
- Top Down Design
  - System Level Design and Rapid Prototyping
  - Hardware/Software Co-Design
  - Component Design
- Bottom Up Design: IP Use and IP Delivery
System Specifications
Closed loop vehicle model

Driver → Vehicle
force, speed, acceleration, jerk, rpm, fuel consumption,...

Vehicle

Engine & Driveline
spark advance, injection time, throttle angle

Controller

emissions, external noise, temperature, ...

Key, Brake, Gas, Transm.,...
INP U T S:
- K - Key
- G - Gas Pedal
- T - Clutch Pedal & Gear Stick
- B - Brake Pedal
- C - Cruise Control

O U T P U T S:
- n - Engine Speed
- FG - Generated Force
- VG - Vehicle Speed
- D - Comfort

---

- Fast Negative Force Transient
  - max D
  - τ < τ_max
  - FG = FG(G,T,n)
  - f_I(n) = 0 & G=0

- Fast Positive Force Transient
  - min τ
  - M_fuel < M_max
  - D > D_min
  - FG = FG(G,T,n)

---

- Rpm Tracking
  - n = n(G)
  - FG = 0

- Force Tracking
  - min f(D, M_fuel)
  - τ < τ_max
  - FG = FG(G,T,n)

- Speed Tracking
  - VG = VG(.)

- Idle
  - n = argmin(M_fuel)
  - FG = 0

- Idle & Trasm On
  - n = n(.)

---

- Stop
  - K = Start
  - n = 0
  - FG = 0

- Startup
  - n = .
  - FG = 0

---

OUT P U T:
- n - Engine Speed
- FG - Generated Force
- VG - Vehicle Speed
- D - Comfort

---

INP U T S:
- G - Gas Pedal
- T - Clutch Pedal & Gear Stick
- B - Brake Pedal
- C - Cruise Control
- K - Key
- D - Comfort

---

INP U T S:
- G - Gas Pedal
- T - Clutch Pedal & Gear Stick
- B - Brake Pedal
- C - Cruise Control
- K - Key
- D - Comfort

---

OUT P U T:
- n - Engine Speed
- FG - Generated Force
- VG - Vehicle Speed
- D - Comfort
Engine Model

- Air Management
- Fuel Management
- Mix Management & Injection
- Ignition (Spark) Timing
- Torque Generation
- Exhaust Gas Treatment

Combustion Engine

Torque Generation

Exhaust Gas Treatment
Abstraction

- The Plant consists of Engine+Drive-Line
- Torque Generation Model abstracted from very complex chemical-mechanical-thermo-dynamical process = FSM!
- Drive Line Dynamics represented as 3rd order linear dynamical system
- Control variables: spark + fuel injection
- Abstraction validated by theoretical devices (formal verification) + measurements on actual cars
Hybrid Systems in Automotive

- Driver (Reference) Model
- Engine + Drive-line (Plant) Model
Cut-off Control: the Problem

- When accelerator pedal is released, no torque is requested.
- Intuitive solution: reduce injection to zero immediately!
  - minimizes consumption and emissions
  - but, sharp torque variations can cause unpleasant power-train oscillations!
- Control Problem: Power-train oscillation reduction via injection signal control
- Present solution: open loop air/fuel modulation
  - engine speed transient during gear changes
  - power-train state not taken into account
Short History

- Specifications obtained November 1996
- Hybrid Modeling November-December 1996
- Control Problem partially solved January 1997
- Complete Simulations May 1997
- Implementation and Field Test October 1997
- Planned for production 1998
Summary of Results

- Hybrid engine+power-train model
- Cut-off control as hybrid control
- Convergence and Optimality Properties
- Experimental results very encouraging:
  - better performance
  - for a commercial car, 50% of memory occupation for data and 75% of memory occupation for code, 1% CPU utilization (Motorola 68020)
  - Approach can be extended to most regions of operations
Target HW/SW Architecture
Choosing the Architecture

Core Processors
(ARM, x86, MIPS)

DSP Processors
(TI320x, Pine, Trimedia)

Configurable Hardware
(Clay, Xilinx, Atmel)

Dedicated Hardware
(Fixed, Synthesized)

Metrics
Programmability
Coverage
Cost
Performance
Power
Mapping

- Associates functional units with architectural units
- Performs HW/SW partitioning
- Associates functional communication with resources (buffers, busses, serial links, etc.)
- Provides estimates of performance of a given function on a given architectural unit
Behavior-Architecture Binding

Meaningful decision making requires fast and educated information on impact of design choices.
Estimation and Modeling

Combines Behavioral Parameters and Architectural Models
Example of System Behavior

Satellite Dish
Front End 1
Transport Decode 2
Rate Buffer 12
Rate Buffer 5
Rate Buffer 9

Mem 13
User/Sys Control 3
Synch Control 4
Video Decode 6
Frame Buffer 7
Video Output 8

Mem 11
Audio Decode/Output 10
Sensor

remote
monitor
speakers

Cable
IP-Based Design of the System Behavior

System Integration
Communication Protocol
Designed in Felix

Testbench
Designed in BONeS

Baseband Processing
Designed in SPW

Transport Decode
Written in C

Decoding Algorithms
Designed in SPW

User Interface
Written in C

Satellite Dish
Cable
Front End
Transport Decode
Rate Buffer
Video Decode
Audio Decode/Output
Frame Buffer
Video Output
User/Sys Control
Sensor
Rate Buffer
Rate Buffer
Rate Buffer
Mem
Mem
Mem
Mem
Mem
Mem
IP-Based Design of the Implementation

Which Bus? PI? AMBA? Dedicated Bus for DSP?

Which DSP Processor? C50? Can DSP be done on Microcontroller?

Which Microcontroller? ARM? HC11?

How fast will my User Interface Software run? How Much can I fit on my Microcontroller?

Can I Buy an MPEG2 Processor? Which One?

Do I need a dedicated Audio Decoder? Can decode be done on Microcontroller?
Architecture Evaluation
Problem

- System Behavior
- System Architecture
- Out of Spec
- Time and Money
- High Cost
- HDL

Refine
Refine
Refine
Refine

EE249
Architecture Evaluation

System Behavior

System Architecture
System Architecture
System Architecture

HDL

In Spec Low Cost

Refine

Time and Money

EE249
Separate Behavior from Architecture

- **System Behavior**
  - Functional Specification of System.

- **Implementation Architecture**
  - Hardware and Software
Map Between Behavior from Architecture

Transport Decode Implemented as Software Task Running on Microcontroller

Audio Decode Behavior Implemented on Dedicated Hardware

Communication Over Bus

- External I/O
- MPEG
- Peripheral
- Audio Decode

Processor Bus
- DSP Processor
- DSP RAM
- Control Processor
- System RAM

EE249
Architecture Determines Performance

Transport Decode → Bus → Control Processor → Bus → Audio Decoder

Audio Decode/Output
Communication Refinement

- Separate *Function* of blocks from inter-block communication
- Substitute lower-level detail for communications behavior

![Diagram showing IP Block With Generic Data Transfer and Protocol Converters]

EE249
Insert Communication Design

Transport Decode

Audio Decode/Output
Engine Control Unit SW Arch.
Mapping and Performance Estimation

- Mapping the system behavior to each ECU architecture
- Performance estimation based on CPU and peripheral models:
  - automatic estimation for untimed behavior running on CPUs
  - manual estimation for timers and TPUs
Software IP authoring

- Key in providing flexibility
- Software is consuming more and more time and resources:
  - Telecom: 70+% of engineering
  - Automotive: from 40% to more than 60%
  - Most of malfunctioning comes from software
- Life Threatening Errors
- Cost of bug fixing
Software Issues

■ Error free requirements
■ Architecture of embedded software:
  ◆ typical of 8-bit architectures
  ◆ assembly code
  ◆ mostly obsolete
  ◆ layered
  ◆ little, if any, documentation
■ Almost no re-use
■ Cost of developing software often not recognized by clients
Opportunities for Embedded Software Design

- A structured approach geared towards re-use and verification is not impossible, but requires investing in new human resources and tools!
- Embedded software has peculiarities that allow effective automatic synthesis and optimization
- Operating systems are RT and micro-kernel
- Could be synthesized as well….to take advantage of the characteristics of the problem
- Validation can be carried out formally if a rigorous approach to modeling is used
SW Architecture Guideline

- Strong separation between “basic” software and application software.
  - Basic software must encapsulate hardware details and sensor/actuator implementation details
  - Application software should reflect the control algorithm hierarchy with no explicit dependency on hardware architecture.

A.Ferrari, M.Antoniotti, and A.S.V.
System Design

Synthesis

Architecture

Function

Mapping

HW

SW

Verification
Proposed Design Methodology

Functional Level

Behavioral Libraries
- Capture Behavior
- Verify Behavior

Architecture Libraries
- Capture Architecture
- Verify Architecture

Mapping Level

Map Behavior to Architecture
- Verify Performance

Architectural Level

Performance Back-Annotation
- Refine HW/SW Architecture
- Link to HW/SW Implementation
- Link to uArchitecture Verification
Ptolemy

- E. Lee Project at UC Berkeley
- Multiple models of computation
- DSP beginnings: Static Dataflow
- Many other models: FSM, Discrete Event
- Mixed model verification
A bit of history: the POLIS project

1988:

The problem:

The target architecture:
Aptix Board consists of:
- micro of choice
- FPGA’s
- FPIC’s

POLIS Methodology

Graphical EFSM

ESTEREL

Sw Synthesis

CFSMs

Hw Synthesis

Sw Estimation

Partitioning

Hw Estimation

Hw/Sw Co-Simulation

Performance/trade-off Evaluation

Formal Verification

Sw Code + RTOS

Logic Netlist

Physical Prototyping

EE249

Aptix Board consists of:
- micro of choice
- FPGA’s
- FPIC’s
Product Design Kit

Reference Implementation
Architecture
Library

Peripheral
Catalog

Sample
Design
Library

Product Development
Peripheral Selection
Application SW Mapping

Toolset
Simulation Estimation Partitioning
Conclusion

- Separate the Behavior from Architecture.
- Map between the Behavior and Architecture.
- Mapping paradigm extends to communication and refinement.
Classic A/D, HW/SW tradeoff

- **RF Front End**
- Can trade custom analog for hardware, even for software
  - Power, area critical criteria, or easy functional modification

Suppose digital limit is pushed
Example: Voice Mail Pager

Design considerations cross design layers

Trade-offs require systematic methodology and constraint-based hierarchical approach for clear justification
Where All is Going

- HW/SW Co-Design Paradigm (Felix)
- VSI Design Paradigm
- Analog Top-Down Design Methodology

Create paradigm shift- not just link methods
- New levels of abstraction to fluidly tradeoff HW/SW, A/D, HF/IF, interfaces, etc- to exploit heterogeneous nature of components
- Links already being forged
Concluding Remarks

- The Industry Structure is undergoing a revolutionary change
- The Design Problems are changing radically their main characteristics
- System Design is becoming more and more the key to success
- System implies Major Emphasis on Software
- Analog, Sensors, Actuators, RF must be part of design
- Deep Submicron makes most of the tools obsolete