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

**Inputs:**
- K - Key
- G - Gas Pedal
- T - Clutch Pedal & Gear Stick
- B - Brake Pedal
- C - Cruise Control

**Outputs:**
- Engine Speed
- Generated Force
- Vehicle Speed
- Comfort

**Controller**
- Engine & Driveline
  - spark advance, injection time, throttle angle

**Driver**
- Key, Brake, Gas, Transm.

**Vehicle**
- force, speed, acceleration, jerk, rpm, fuel consumption,...
- emissions, external noise, temperature,...
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

![Diagram of Hybrid Systems]

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

Behavioral Function

Software Path

Hardware Path

Model Library

Code Generation

SW library

ARM8

Processor Model

Network of Modules

Model Library

Mapping

Combines Behavioral Parameters and Architectural Models

Example of System Behavior
IP-Based Design of the System Behavior

Testbench Designed in BONeS
Baseband Processing Designed in SPW
Transport Decode Written in C
Decoding Algorithms Designed in SPW

User Interface Written in C
System Integration Communication Protocol Designed in Felix

IP-Based Design of the Implementation

Processor Bus
- DSP Processor
- DSP RAM
- Control Processor
- System RAM
- External I/O
- MPEG
- Peripheral
- Audio Decode

Questions:
- Which Bus? PI? AMBA?
- Which DSP Processor? C50? Can DSP be done on Microcontroller?
- Which Microcontroller? ARM? HC11?
- Can I Buy an MPEG2 Processor? Which One?
- How fast will my User Interface Software run? How Much can I fit on my Microcontroller?
- Can I Buy an MPEG2 Processor? Which One?
- Can I Buy an MPEG2 Processor? Which One?
- Can I Buy an MPEG2 Processor? Which One?
- Can I Buy an MPEG2 Processor? Which One?
- Can I Buy an MPEG2 Processor? Which One?
Architecture Evaluation

Problem

System Behavior

Refine

System Architecture

Refine

Out of Spec

System Behavior

Refine

System Architecture

Refine

High Cost

HDL

Behavior

Architecture

Time and Money

EE249

Architecture Evaluation

System Behavior

System Architecture

System Architecture

System Architecture

Refine

In Spec

Low Cost

HDL

Behavior

Architecture

In Spec

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
- Communication Over Bus
- Audio Decode Behavior Implemented on Dedicated Hardware
Architecture Determines Performance

Communication Refinement

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

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.
Proposed Design Methodology

Functional Level
- Capture Behavior
- Verify Behavior
- Capture Architecture
- Verify Architecture

Mapping Level
- Map Behavior to Architecture
- Verify Performance
- Refine HW/SW Architecture
- Performance Back-Annotation
- Link to HW/SW Implementation
- Link to uArchitecture Verification

Architectural Level

Ptrolemy
- 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:

POLIS Methodology

- Graphical EFSM
- ESTEREL
- CFSMs
- Partitioning
- Hw Synthesis
- Hw Estimation
- Logic Netlist
- Physical Prototyping
- Compilers
Design System

- Software Development Tools
- Poindexter C
- Behavior Libraries
- Architecture Libraries
- HW/SW Coverification
- SW Compile HW Synthesis

Toolset
- System Behavior Specification
- System Architecture Evaluation
- Design Verification

Product Design Kit

- Reference Implementation
- Architecture Library
- Peripheral Catalog
- Sample Design Library
- Product Development
- Peripheral Selection
- Application SW Mapping

Toolset
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
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

- 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