The MARCO/DARPA Gigascale Silicon Research Center for Design & Test: Form

The Essence of Polis/Felix/VCC Project: Virtual Component Co-design

1988:
- Describe & verify product behavior
- Describe product architectures
- Explore HW/SW design tradeoffs
  - Map behavior to architecture
  - Use performance simulation
  - Perform communication refinement
- Integrated flow to implementation

The next level of Abstraction …

- 1970’s
- 1980’s
- 1990’s
- Year 2000 +
Architectural Choices

OMAP™ Block Diagram

1/Efficiency (power, speed)
Hardware Platforms Not Enough!

- Hardware platform has to be abstracted
- Interface to the application software is the “API”
- Software layer performs abstraction:
  - Programmable cores and memory subsystem “hidden” by RTOS and compilers
  - I/O subsystem with Device Drivers
  - Network with Network Communication Software
Software Platforms

Nexperia-DVP Software

- Nexperia™ -DVP Software Architecture
  - Supports multiple OSs and middleware software
  - Abstracts platform functionality via consistent APIs
- Nexperia™-DVP Streaming Software
  - Encapsulates implementation of streaming media components (hardware and software)
- Nexperia™ Platform Software
  - OS independent device drivers for on-chip and off-chip devices
MOSAIC SW Architecture & Components for Automotive Dashboard and Body Control

Platforms

Application Platform layer
- 10% of total SW

SW Platform layer
- 60% of total SW

μControllers Library

OS/2
OSEK
RTOS
Application Programming Interface
Boot Loader
Sys. Config.
Transport
KWP 2000
CCP
Application
Specific
Software
Speedometer
Tachometer
Water temp.
ODE

Application Libraries
Nec78k
HC12
HC08
H8S26
MB90

SW Platform Reuse
> 70% of total SW

Customer Libraries

Platform Specification
Platform Design-Space Exploration
Platform Instance
Architectural Space

Application Space
System Platform
Application Instance
A platform is, in general, an abstraction that covers a number of possible refinements into a lower level. For every platform, there is a view that is used to map the upper layers of abstraction into the platform and a view that is used to define the class of lower level abstractions implied by the platform.
Application example:

- Automotive Power-Train Control Design: from car manufacturer specs to software design to architecture selection to IC implementation
- Project in collaboration with Cadence, Magneti-Marelli, ST Microelectronics, Accent

Power-Train Control System

- Electronic device controlling an internal combustion engine and a gearbox
- The goal
  - offer appropriate driving performance (e.g. torque, comfort, safety)
  - minimize fuel consumption and emissions
- Relevant characteristics
  - strictly coupled with mechanical parts
  - hard real-time constraints
  - complex algorithms for controlling fuel injection, spark ignition, throttle position, gear shift ...
  - 135,000 lines of C code with no comments
- First Step was to redesign software with methodology to map into different hardware platforms with little effort!
**System Specifications**

**Functional View for System Validation**

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

- **Outputs:**
  - n - Engine Speed
  - F - Generated Force
  - V - Vehicle Speed

- **Specifications:**
  - Fast Negative Force Transient
  - Fast Positive Force Transient
  - Speed Tracking
  - Idle & Trans On

- **Equations:**
  - \( f(I(n)) = 0 \) & \( G = 0 \)
  - \( n = \text{argmin} (M_{\text{fuel}}) \)
  - \( F_G = 0 \)
  - \( V_G = V_G(.) \)
  - \( n = n(G) \)

- **Diagram:**
  - Various states and transitions including:
    - Stop
    - Startup
    - Idle
    - Rpm Tracking
    - Force Tracking
    - Fast Negative Force Transient
    - Fast Positive Force Transient
Goal

- Develop guaranteed properties control algorithms for all power-train modes
- Implement control strategies on embedded controllers “optimally” with respect to production cost, design time, reliability, safety

Model of Power-train

Simple?
Single Cylinder Hybrid Model

Hybrid Model vs Mean-Value Model

- Mean-Value Model: accurate over a longer time window
  - regulation control problems
  - low performance transient problems
- Hybrid Model: cycle accurate
  - transient control problems
  - stability of delay-sensitive control algorithms
  - high performance control algorithms
The Dual-Arm Architecture

A symmetric dual processor architecture with a high-bandwidth interconnection network among processors, memory, and I/O sub-systems

- 11 Million Tr., 4% more area than single processor solution but twice the performance on application
- Most performing architecture for Power-train applications, designed to be re-used over two generations (3-4 years cycles) of system products or more
- Entirely designed using the methodology in less than 1 3/4 year from conception (March 99) to first silicon (Jan 01)
- In production by 2002, shipments to car manufacturer 2003

Design “Practice”
Design Science: Build upon solid foundations

- Design of Function Blocks
- Design of Communication Media
- Design of Architecture Components

Metropolis Infrastructure
- Model of computation
- Design methodology
  - Abstraction levels
  - Refinement
- Base tools
  - Design imports
  - Simulation

Metropolis: Synthesis/Refinement
- Compile-time scheduling of concurrency
- Communication-driven hardware synthesis
- Protocol interface generation
- Latency insensitive protocols

ETROPOLIS
- (20+ from Academia (UCB,CMU) and Industry (Intel, ST,BMW, Cadence))