Part 2: Principles for a System-Level Design Methodology

• Separation of Concerns: Function versus Architecture
• Platform-based Design
Design Effort vs. System Design Value

Today

Effort/Value

Function

Level of Abstraction

HW/SW

Conceptual Gap

RTL - SW

RTL/Gate “platform”

Design Entry Level

Design Entry Level

RTL/Gate “platform”

Design Entry Level

Design Entry Level

RTL/Gate “platform”

Design Entry Level

Design Entry Level

Today

Tomorrow

Architecture

RTL - SW

Mask - ASM
Design Effort vs. System Design Value

Effort/Value vs. Level of Abstraction

- Function
- HW/SW
- Architecture
- RTL - SW
- Mask - ASM

Today

Tomorrow

Hand-off “platform”

Design Entry Level
System-Level Design Science

• Design Methodology:
  – Top Down Aspect:
    – Orthogonalization of Concerns:
      – Separate Implementation from Conceptual Aspects
      – Separate computation from communication
    – Formalization: precise unambiguous semantics
    – Abstraction: capture the desired system details (do not over-specify)
    – Decomposition: partitioning the system behavior into simpler behaviors
    – Successive Refinements: refine the abstraction level down to the implementation by filling in details and passing constraints
  – Bottom Up Aspect:
    – IP Re-use (even at the algorithmic and functional level)
    – Components of architecture from pre-existing library
Separate Behavior from Micro-architecture

- **System Behavior**
  - Functional Specification of System.
  - No notion of hardware or software!

- **Implementation Architecture**
  - Hardware and Software
  - Optimized Computer
Models of Computation: And There are More...

- Continuous time (ODEs)
- Spatial/temporal (PDEs)
- Discrete time
- Rendezvous
- Synchronous/Reactive
- Dataflow
- ...

Each of these provides a formal framework for reasoning about certain aspects of embedded systems.

Tower of Babel, Bruegel, 1563
Formalization

- Model of a design with precise unambiguous semantics:
- Implicit or explicit relations: inputs, outputs and (possibly) state variables
- Properties
- “Cost” functions
- Constraints

Formalization of Design + Environment = closed system of equations and inequalities over some algebra.
Validating Designs

• By construction
  – property is inherent.

• By verification
  – property is provable.

• By simulation
  – check behavior for all inputs.

• By intuition
  – property is true. I just know it is.

• By assertion
  – property is true. Wanna make something of it?

• By intimidation
  – Don’t even try to doubt whether it is true

• It is generally better to be higher in this list
Notion of Time

Salvador Dali, *The Persistence of Memory*, 1931

- Continuous time
- Discrete time
- Totally-ordered discrete events
- Multirate discrete time
- Partially-ordered discrete events
- Synchronous/reactive
System Specifications

Power Windows

[Diagram of power window system]

[Diagram showing window control system]
Two Basic Questions …
Question I - IP Authoring

- How to design a system block?
  - Starting from the system level
  - With a consistent test-bench
  - Getting from the abstract, untimed system model to the clocked HW or SW implementation model

- Example

- Rake Receiver
  - Which are the optimal algorithms?
  - How does it work fixed point?
  - How is it best implemented?
  - Does the implementation work as specified in the system level?
Two Basic Questions …

Question II – IP Integration

• How to integrate system blocks?
  – Starting from the system level
  – With a consistent test-bench
  – Getting from the abstract, un-timed system model to the clocked HW or SW implementation model
  – Communication between blocks
  – Addressing Platform Based design

• Example

• 3G Cell phone
  – Which are the optimal algorithms?
  – Do they work together functionally?
  – Is the architecture sufficient?
  – Does the implementation integration work?
Historical Perspective: 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:
The Essence of the Polis/Felix/VCC Approach

1988:

Traditional System Design

- System Behavior
- System Architecture
- System Implementation
- System Performance

VCC Separation and Mapping

1. Mapping
2. System Architecture
3. Behavior on Architecture
4. Refine

Data Sheets on paper

Executable Data Sheets
Example of System Behavior

[Diagram showing various components such as 'Front End 1', 'Transport Decode 2', 'Rate Buffer 5', 'Rate Buffer 9', 'User/Sys Control 3', 'Sensor', 'Video Output 8', 'Audio Decode/Output 10', 'Mem 11', 'Mem 13', 'Video Decode 6', 'Frame Buffer 7', 'Synch Control 4', 'Satellite Dish', 'Cable', 'remote', 'monitor', 'speakers']
IP-Based Design of the System Behavior

System Integration
Communication Protocol
Designed in Felix

Testbench
Designed in BONeS

Satellite Dish
Cable

Front End 1

Transport Decode 2
Rate Buffer 12
User/Sys Control 3
Synch Control 4

Rate Buffer 5
Video Decode 6
Frame Buffer 7
Video Output 8

Rate Buffer 9
Audio Decode/Output 10
Mem 11

Mem 13

Sensor

User Interface
Written in C

Baseband Processing
Designed in SPW

Transport Decode
Written in C

Decoding Algorithms
Designed in SPW

User Interface
Written in C
Challenges and Trends

- Cell-Based ASICs becoming prohibitively expensive for all but highest volume applications

**Shift to**
- Re-use Strategy at all levels
- Higher Level of Abstractions
- Software !!!
The Quest for the Next Level of Abstraction

- Transistor Model Capacity Load
- Gate Level Model Capacity Load
- SDF Wire Load
- IP Block Performance Inter IP Communication Performance Models

IP Blocks
- RTL Clusters
- SW Models

1970s 1980s 1990s 2000+

abstract

abstract

abstract

abstract
Can I Buy an MPEG2 Processor? Which One?

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

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

Which Microcontroller? ARM? HC11?

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

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

[Diagram showing architectural choices with axes labeled 'Flexibility' on the y-axis and '1/Efficiency (power, speed)' on the x-axis. The diagram includes categories such as 'Dedicated Logic', 'Direct Mapped Hardware', 'Satellite Processor', 'Hardware Reconfigurable Processor', 'Software Programmable DSP', 'MAC Unit', 'Addr Gen', 'Prog Mem', and 'General Purpose μP'.]
Map Between Behavior from Architecture

- **Transport Decode**: Implemented as Software Task Running on Microcontroller
- **Audio Decode**: Implemented on Dedicated Hardware
- **Communication Over Bus**
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