Overview of SOC
Architecture design

Tien-Fu Chen
National Chung Cheng Univ.

SOC design Issues

- SOC architecture
- Reconfigurable
  - System-level
  - Programmable processors
  - Low-level reconfiguration
- On-chip bus
- Embedded Software Issues
Embedded Systems vs. General Purpose Computing - 1

- Embedded System
  - Runs a few applications often known at design time
  - Not end-user programmable
  - Operates in fixed run-time constraints, additional performance may not be useful/valuable

- General purpose computing
  - Intended to run a fully general set of applications
  - End-user programmable
  - Faster is always better

Embedded Systems vs. General Purpose Computing - 2

- Embedded System
  - Differentiating features:
    - power
    - cost
    - speed (must be predictable)

- General purpose computing
  - Differentiating features
    - speed (need not be fully predictable)
    - speed
    - did we mention speed?
    - cost (largest component power)
**Embedded System: Examples**

Source: Collett International Research- December 2000 Research on 360 IC/ASIC Design Teams in North America

© by Tien-Fu Chen@CCU

**Design Complexity Increase**

<table>
<thead>
<tr>
<th>SoC Characteristics</th>
<th>Current Design</th>
<th>Next Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Gate Count</td>
<td>616K</td>
<td>1,000K</td>
</tr>
<tr>
<td>Designs with Gate Count &gt; 1M</td>
<td>18%</td>
<td>31%</td>
</tr>
<tr>
<td>Units Shipped &gt;5M</td>
<td>20%</td>
<td>37%</td>
</tr>
<tr>
<td># of lines of DSP SW</td>
<td>54K</td>
<td>295K</td>
</tr>
<tr>
<td># of lines of uP SW</td>
<td>75K</td>
<td>266K</td>
</tr>
<tr>
<td>Clock Speed &gt;133mhz</td>
<td>44%</td>
<td>61%</td>
</tr>
<tr>
<td>&gt;400mhz</td>
<td>15%</td>
<td>22%</td>
</tr>
<tr>
<td>&gt;5 Clock Domains</td>
<td>25%</td>
<td>35%</td>
</tr>
<tr>
<td>&gt;9 Clock Domains</td>
<td>10%</td>
<td>12%</td>
</tr>
</tbody>
</table>

Source: Collett International Research- December 2000 Research on 360 IC/ASIC Design Teams in North America

© by Tien-Fu Chen@CCU
Architectures

- Supplements *Models* by specifying how the system will actually be implemented

- Goal of each architecture is to describe
  - Number of components
  - Type of each component
  - Type of each connection among above components

- General classification
  - Application-specific architectures: DSP
  - General-purpose architectures: CISC, RISC
  - Parallel processors: VLIW, SIMD, MIMD
**System Architecture Design**

- **System Architecture & Exploration**
  - **What**
    - Hardware/Software partitioning; processor, and memory architecture choices; system timing budget, power management strategy, system verification strategy…
    - Partitioning into HW block hierarchy, cycle time budgeting, block interfaces, block verification, clock architecture and test strategy
    - Fixed point architecture exploration and design
  - **How** - Quickly assemble architecture(s) for exploration to measure system timing/performance. Need to accurately (enough) model the bottlenecks

**System Integration & Verification**

- **System Design Environment for HW/SW Refinement, Verification and Integration**
  - **What**
    - Enables hierarchical (manual or automatic) refinement of individual blocks of design in context of system. Maintain system and hierarchical test benches
    - Verification of refined hardware/software with entire system design
    - Define next level of clock architecture (derived) and test strategy
  - **How** - Build a system verification hierarchy that allows integration of HW blocks, system software (HAL), embedded application SW and eventually verifying the entire design at cycle accurate (or RTL) level
CoDesign and Co-Synthesis

- Specification
- Partition
- Co-Synthesis
- Synthesis

HW: HDL (Behavioral, DataFlow, Structural), Schematic

SW: Algorithm, Textual/Graphical representation

- RTL, Gate level, Transistors, Layout
- Executable or Compilable code: The program(s), OS routines

Traditional design

Traditional System Design Process

- Tasks
- System design
- SW design
- ASIC design
- Fabrication
- Test
- SW test
- PCB test

© by Tien-Fu Chen@CCU

SOC - 10

SOC - 11
**System-level Co-design**

![Co-Design Process Diagram]

- **Tasks** vs **Time**
  - System design
  - Shared Design
  - SW design
  - ASIC design
  - Fabrication
  - Test
  - PCB test
  - SW test

**System-Level Partitioning**

---

**Configurability and Embedded Systems**

- Advantages of configuration:
  - Pay (in power, design time, area) only for what you use
  - Gain additional performance by adding features tailored to your application:

- Particularly for embedded systems:
  - Principally in embedded controller microprocessor applications
  - Some us in DSP
What to Configure?

- What parts of the microcontroller/microprocessor system to configure?

- Easy answers:
  - Memory and Cache Sizes - get precisely the sizes your applications need
  - Register file sizes
  - Interrupt handling and addresses

- Harder answers:
  - Peripherals
  - Instructions

- But first we need more context

---

Trickle Down Theory of Embedded Architectures

- Mainframe/supercomputers
- High-end servers/workstations
- High-end personal computers
- Personal computers
- Lap tops/palm tops
- Gadgets
- Watches
- ...
Configurability in ARM Processor

- ARM allows for configurability via AMBA bus
- Offers "prime cell" peripherals which hook into AMBA Peripheral Bus (APB)
- UART
- Real Time Clock
- Audio Codec Interface
- Keyboard and mouse interface
- General purpose I/O
- Smart card interface
- Generic IR interface

ARM7 core
**ARM’s Amba open standard**

- Advanced System Bus, (ASB) - high performance, CPU, DMA, external
- Advanced Peripheral Bus, (APB) - low speed, low power, parallel I/O, UART’s
- External interface

http://www.arm.com/Documentation/Overviews/AMBA_Intro/#intro

---

**Ex: ARM Infrared (IR) Interface**

---

© by Tien-Fu Chen@CCU

SOC - 18

SOC - 19
**Another Kind of Configurability**

Synthesis of a processor core from an RTL description allows for:

- full range of other types of configurability
- additional degrees of freedom in quality of implementation

Examples:

- ARM7
- Motorola Coldfire
- Tensilica Xtensa
Issues in low-level Configurable Design

- Choice and Granularity of Computational Elements
- Choice and Granularity of Interconnect Network

(Re)configuration Time and Rate
- Fabrication time --> Fixed function devices
- Beginning of product use --> Actel/Quicklogic FPGAs
- Beginning of usage epoch --> (Re)configurable FPGAs
- Every cycle --> traditional Instruction Set Processors

The Choice of the Computational Elements

<table>
<thead>
<tr>
<th>Reconfigurable Logic</th>
<th>Reconfigurable Datapaths</th>
<th>Reconfigurable Arithmetic</th>
<th>Reconfigurable Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLB</td>
<td>mux</td>
<td>AddGen</td>
<td>Data Memory</td>
</tr>
<tr>
<td>CLB</td>
<td>reg0</td>
<td>Memory</td>
<td>Instruction Decoder &amp; Controller</td>
</tr>
<tr>
<td>CLB</td>
<td>reg1</td>
<td>MAC</td>
<td>Data Memory</td>
</tr>
<tr>
<td>CLB</td>
<td>adder</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>buffer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit-Level Operations
- e.g. encoding

Dedicated data paths
- e.g. Filters, AGU

Arithmetic kernels
- e.g. Convolution

RTOS
- Process management
**Multi-granularity Reconfigurable Architecture: The Berkeley Pleiades Architecture**

- Computational kernels are “spawned” to satellite processors
- Control processor supports RTOS and reconfiguration
- Order(s) of magnitude energy-reduction over traditional programmable architectures

---

**Matching Computation and Architecture**

Two models of computation: 
communicating processes + data-flow  

Two architectural models: 
sequential control+ data-driven
### Execution Model of a Data-Flow Kernel


code_segment

for (i=1; i<=L; i++)
    for (k=i; k<=L; k++)
        phi[i][k] = phi[i-1][k-1] + in[NP-i]*in[NP-k] - in[NA-1-i]*in[NA-1-k];

code_segment

- Distributed control and memory

© by Tien-Fu Chen@CCU

---

### Software Methodology Flow

- **Algorithms**
  - Kernel Detection
  - Behavioral Estimation/Exploration
  - SUIF+ C-IF
  - Power & Timing Estimation of Various Kernel Implementations
- **C++**
- **Partitioning**
- **Software Compilation**
  - Reconfig. Hardware Mapping
  - Interface Code Generation

© by Tien-Fu Chen@CCU

SOC - 26

SOC - 27
The System-on-a-Chip Nightmare

The “Board-on-a-Chip” Approach

Custom Interfaces

Control Wires

Peripheral Bus

System Bus

Sonics SOC Integration Architecture

SiliconBackplane™ (patented)

Open Core Protocol™

MultiChip Backplane™

SiliconBackplane Agent™
**Master vs. Slave**

- **Master vs. Slave**
  - IP Core
  - Master vs. Slave
  - Open Core Protocol
  - Request Response
  - Initiator Target
  - On-Chip Bus

**The Backplane: Why Not Use a Computer Bus?**

- **Transmit FIFO**
- **Receive FIFO**
- **Arbiter**
- **Address**
- **Computer Bus**
- **Time**

- Expensive to decouple
- Not designed for real-time

© by Tien-Fu Chen@CCU
Communication Buses Decouple and Guarantee Real Time

- Connections are expensive
- Poor read latency

On-Chip Bus for SOC

- Example on-chip bus interconnects
  - ARM’s AMBA bus
  - IBM’s Core Connect
  - Virtual Socket Interface Alliance group
  - Open Connect Protocol group

- Example processor cores
  - ARM
  - MIPS
  - PowerPC
Design Example: The Radio-on-a-Chip

- DSP and control intensive
- Mixed-mode
- Combines programmable, flexible, and application-specific modules
- Cost and energy are the key metrics

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 overspecify)
    - 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

Map Between Behavior from Architecture

- Transport Decode Implemented as Software Task Running on Microcontroller
- Audio Decode Behavior Implemented on Dedicated Hardware
**Embedded Software Crisis**

J. Fiddler - WRS

Cheaper, more powerful Microprocessors

Increasing Time-to-market pressure

More Applications

Bigger, More Complex Applications

© by Tien-Fu Chen@CCU

**SW: Embedded Software Tools**

© by Tien-Fu Chen@CCU
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
**Platform-based methodology**

- **Platform based design:**
  - Application mapped on architecture
  - Performance evaluation and iterative refinement

- **Challenges:**
  - Complete system simulation
  - Complexity management
  - Composability and reuse

- **Key elements for composability**
  - Identification and use of useful models of computation
    - FSMD, DE, DF, CSP, ...
  - A flexible, extensible language platform to capture the functionality.

- **Composability can be achieved using Object-oriented mechanisms:**

---

**Final goal for SOC: Platform-Based Design**

**Taking Design Block Reuse to the Next Level**

- **Pre-Qualified/Verified Foundation-IP**
- **Hardware IP**
- **SW IP**
- **Programmable**

**Foundation Block** + Reference Design

- **Scaleable bus, test, power, IO, clock, timing architectures**
- **Processor(s), RTOS(es) and SW architecture**

**Methodology / Flows:**

- System-level performance evaluation environment
- Rapid Prototype for End-Customer Evaluation
- SoC Derivative Design Methodologies
- Foundry Targetting Flow

---

*IP can be hardware (digital or analogue) or software. IP can be hard, soft or 'firm' (HW), source or object (SW)
Summary of SOC Design