Overview of *configurable* architectures

Prof. Kurt Keutzer
EECS
keutzer@eecs.berkeley.edu

Thanks to Andre Dehon, Jan Rabaey, and many vendors

Outline

- Motivation for *configurable* platforms
- Taxonomy of *configurable* platforms
- Examples of *configurable* platforms
- Projects in *configurable* platforms
**Increasing Device and Context Complexity**

- Exponential increase in device complexity—increasing with Moore’s law (or faster)!
- System context in which devices are deployed (e.g., cellular radio) are increasing in complexity as well exponential increases in design productivity

**We have exponentially more transistors!**

**Deep Submicron Effects**

- Smaller geometries are causing a wide variety of effects that we have largely ignored in the past:
  - Cross-coupled capacitances
  - Signal integrity
  - Resistance
  - Inductance

**Design of each transistor is getting more difficult!**
**Heterogeneity on Chip**

- Greater diversity of on-chip elements
  - Processors
  - Software
  - Memory
  - Analog

*More transistors doing different things!*

**Stronger Market Pressures**

- Decreasing design window
- Less tolerance for design revisions

*Exponentially more complex, greater design risk, greater variety, and a smaller design window!*
Motivation: Quadruple-Whammy

Exponentially more complex, greater design risk, greater variety, and a smaller design window!

Likely alternative...

- Unprecedented hunger for silicon customization but...
- The quadruple-whammy implies:
  - Higher NRE/design
  - Growing number of applications served through more highly-programmable platforms
  - With higher-design volume to compensate for higher NREs
  - From ASIC to ASIP
Key Problems in IC Design and Their Solution

Problem:
• High development (NRE) cost
• Need to use IC’s for multiple related applications
• Minimize design risk/increase time-to-market

Solution:
• Amortize cost over many designs by developing platforms
• Make platform programmable so that it can be re-used
• Use pre-developed platform where possible and tailor it to application through programming

System ASIC Design in 200x

• Less like synthesis of an integrated circuit from a high-level description
• More like programming of a complex application-specific processor

Prog Env

Impact
Front-End

MESCAL
Liberty
Back-End

Simulator / Visualization
Outline

- Motivation for *configurable* platforms
- Taxonomy of *configurable* platforms
- Projects in *configurable* platforms
- Examples of *configurable* platforms

A 3D Design Space
Programming Model

- The computation-abstraction level (name designed by committee) is about how is the level of configurability presented to the user
- Natural levels:
  - System architecture – e.g. hash engine
  - Instruction-set architecture - e.g. MAC x,y,z
  - Micro-architecture – operator-level
  - Logic level – moving bits

We’ll find systems offering configurability at all these levels

Design Space: Vertical Axis

<table>
<thead>
<tr>
<th>Computation Abstraction Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process/System Architecture</td>
</tr>
<tr>
<td>Instruction-Set Architecture</td>
</tr>
<tr>
<td>Micro-Architecture</td>
</tr>
<tr>
<td>Logic level</td>
</tr>
</tbody>
</table>

mov r5, r2
Reconfigurable Features

- The reconfigurable-feature (another name designed by committee) is about the manner in which the device supports the configurability

- Computation:
  - Processes – e.g. processing element
  - Datapaths - e.g. filtering elements
  - Operators – adders, multipliers,
  - Logic level – Look-up tables

- Communication:
  - Bus, mesh, hierarchical mesh, on-chip packet routing

- We’ll find systems using configurability at all these levels ... but

- Not as straightforward as it seems – processes may be supported on a logic level fabric
The Choice of the Computational Elements

Reconfigurable Logic

Reconfigurable Datapaths

Reconfigurable Arithmetic

Reconfigurable Processors

Bit-Level Operations
e.g. encoding

Dedicated data paths
e.g. MAC

Arithmetic kernels
e.g. Convolution,
Hash decoding

Processing elements

The Choice of the Communication Fabric

<table>
<thead>
<tr>
<th></th>
<th>dot_product</th>
<th>vector sum w/ scalar mult.</th>
<th>IIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-bus</td>
<td>50</td>
<td>50</td>
<td>138</td>
</tr>
<tr>
<td>Mesh</td>
<td>Best</td>
<td>8.7</td>
<td>24.6</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>17.7</td>
<td>43.4</td>
</tr>
<tr>
<td>H. Mesh</td>
<td>Best</td>
<td>4.7</td>
<td>18.8</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>11.1</td>
<td>31.3</td>
</tr>
</tbody>
</table>

Multi-Bus

Mesh

Hierarchical Mesh
**A 3D Design Space**

- **Computation Abstraction Level**
- **Reconfigurable Feature**
- **Binding Rate**

**Binding Rate**

- The binding-rate is about how the device supports the configurability over time.
- **``Configurable``**:  
  - Binding occurs at fabrication time.
  - Similar to ASIC except that the model used to create the devices and the model used to program them is much different.
  - ARC, Tensilica, Improv, Actel.
- **``Re-configurable``**:  
  - Binding occurs in field, but typically only at ``power-up``.
  - Xilinx, Altera, Triscend.
- **``Dynamically reconfigurable``**:  
  - Binding occurs in field, and may occur every 1000’s of cycles.
  - Principally explored in academia – GARP, but also Chameleon.
So you want to be a reconfigurable architect ...

- Think about the applications that you want to support:
  - WCDMA, UMTS, packet forwarding, etc.
- Think about the programming model you want to provide:
  - Ptolemy/Models-of-computation, MATLAB, Click, C, assembler, assembly + Verilog
- Think about the micro-architecture of your device and the role that reconfigurable fabric plays in it
  - 80% PE's and 20% reconfigurable ...
- Do you want to provide configurability, reconfigurability, dynamic reconfigurability ...

Don’t Forget to Think About This

- Components of Cost
  - Area of die / yield
  - Code density (memory is the major part of die size)
  - Packaging
  - Design effort
  - Programming cost
  - Time-to-market
  - Reusability
Outline

- Motivation for *configurable* platforms
- Taxonomy of *configurable* platforms
- Projects in *configurable* platforms
- Examples of *configurable* platforms

*configurable* related Project Ideas - 1

- Processor/reconfigurable-fabric interface
  - utility of a tight link between a processor and reconfigurable fabric depends largely on management of time to:
    » Reconfigure fabric
    » Communication overhead
  - Problem can be addressed by:
    » Tricks in fabric (caching configurations etc.)
    » Or ... how about learning to fill configuration time much as we learn to fill delay slots in a branch ...
• What’s the right fabric?
  – Chameleon uses a coarse granularity datapath oriented fabric
  – Triscend uses a fine-grained fabric
  – What’s right for your favorite application ....?

• Programming model
  – Ultimate success or failure of all these gadgets depends on developing of a programming model for these devices
  – Develop banana curve for one application for a device that has C-language support try:
    » Assembly programming the application
    » C-coding the application
    » Click, Matlab (details provided), Teja to the device
    » Compare results – how much do we give up for higher-level programmability
Outline

- Motivation for *configurable* platforms
- Taxonomy of *configurable* platforms
- Projects in *configurable* platforms
- Examples of *configurable* platforms

Variety of Platforms

Domain-Specialization

- Chameleon Systems
- Specialized Micro-Architectures
- Morphics
- PMC Sierra
- Specialized Instruction-Set Architectures
- Network Processors
- Improv Systems
- Frontier Design
- ARC
- Tensilica

FPGA

Xilinx
Altera
Actel
Adaptive Silicon

Processor

Triscend
Atmel
Proceler
Here’s the Deal … We can Talk about any of:

- Actel
- Adaptive silicon
- Altera
- Xilinx
- Chameleon
- Morphics
- Network processors – takes a whole lecture
- ARC
- Improv Systems
- Tensilica