Redefis: An SoC Platform for Implementing Application-Specific or User-Custom Logic

Kazuaki Murakami
Kyushu University
redefis@fleets.jp
Five Ways to Design & Implement Custom Logic (from MPSoC’04)

How to Implement General Logic
• General-Purpose Processor + Software

How to Implement Custom Logic
• “From Scratch” Approach
  – Design new hardware every time
• IP-Core-Based Approach
  – Reuse existing hardware designs, or IP cores
• Platform-Based Approaches
  – Processor + Software
  – Configurable Processor + Software
  – Reconfigurable Hardware
  ➔ Reconfigurable Processor + Software
Five Ways to Design & Implement Custom Logic (from MPSoC’04)
Five SoC Design Flows
(from MPSoC’04)

Behavior Level
- HDL/C
  - Logic Synthesis
  - Behavior Synthesis

RT Level
- HDL
  - Hardwired Logic
  - Reconfigurable Hardware

Hardware Configuration Data

Reconfigurable Processor

Software

Platform-Based Design

HDL/C + C

Compilation

Hardwired Logic

Reconfigurable Processor

Software
Redefis (Redefinable ISA Processor): A Reconfigurable Processor

- Normal Reconfigurable Processor
  - Algorithm
  - Compilation
  - HDL
  - HW Synthesis
  - HW
  - Configuration Setting
  - Program Execution
  - Processor Configuration Data
  - Reconfigurable Processor

- Redefis (Redefinable ISA Processor)
  - Algorithm
  - Compilation
  - ISA
  - HW
  - Configuration Setting
  - Program Execution
  - Processor Configuration Data
  - Reconfigurable Processor
  - ISA Gen
Where Does Redefis Go?

General-purpose
Application-specific

Processor+Software
Hardware

Yes
No

Redefis

Traditional Processor

Programmability/Reconfigurability

FPGA

Hardwired
Vulcan: An Implementation of Redefis

- Program Memory
- Controller
- Data Memory
- Reconfigurable Datapath (RDP)
- Register Files
- Configuration Memory

- 64-bit width
- 128 PE’s (Programmable Elements)
- 256-bit width

- RDP configuration data for 128 custom instructions
- Vulcan binary-code store

- Instruction execution control

Vulcan: An Implementation of Redefis
Vulcan Chip & Board
Reconfigurable Datpath (RDP)
Reconfigurable Datpath (RDP)

- 6-input & 2-output LUT (Lookup Table)
- Switches between VL and HL’s
- VL (64 bits)
- HL (64 bits)
- 128 PE’s (16*8)

- 128 PE's (16*8)
- Switches between VL and HL’s
- VL (64 bits)
- HL (64 bits)
- 6-input & 2-output LUT (Lookup Table)

VL (64 bits)

HL (64 bits)
Instruction Execution Flow

Program Memory

Controller

Data Memory

Reconfigurable Datapath (RDP)

Configuration Memory

Register Files
Instruction Execution Flow
- Phase 1: I-Fetch -

Program Memory

Controller

Data Memory

Reconfigurable Datapath (RDP)

Register Files

Configuration Memory

I-Fetch
Instruction Execution Flow
- Phase 1: I-Fetch -

- Phase 1: I-Fetch
  - Opcode
  - Register No. (2 operands)
  - Control
  - Branch Cond.
  - Data Memory Addr.

- Datapath (RDP)

- Memory
- Files

Configuration Memory
Instruction Execution Flow
- Phase 1: PC Update -

1. Instruction Fetch
2. PC increment
3. Controller
4. Reconfigurable Datapath (RDP)
5. Data Memory
6. Register Files
7. Configuration Memory

Program Memory

Instruction Execution Flow
- Phase 1: PC Update -
Instruction Execution Flow
- Phase 2: Configuration Data Load -

- Program Memory
- Controller
- Reconfigurable Datapath (RDP)
- Data Memory
- Register Files
- Configuration Memory

- Configuration Data Load -
Instruction Execution Flow
- Phase 2: Data Load -

Program Memory

Controller

Reconfigurable Datapath (RDP)

Configuration Memory

Data Load

Data Memory

Register Files
Instruction Execution Flow
- Phase 3: Execution -

Program Memory

Controller

Reconfigurable Datapath (RDP)

Data Memory

Execution

Data Load

Register Files

Configuration Memory
Instruction Execution Flow
- Phase 3: Execution -
Vulcan Instruction Pipeline

1 clock cycle
Redefis (Redefinable ISA Processor): A Reconfigurable Processor

- Normal Reconfigurable Processor
  - Algorithm
  - Compilation
  - HDL
  - HW Synthesis
  - HW
  - Configuration Setting
  - SW
  - Program Execution
  - Processor Configuration Data

- Redefis (Redefinable ISA Processor)
  - Algorithm
  - Compilation
  - ISA
  - HW Synthesis
  - ISA Gen
  - SW
  - Program Execution
  - Processor Configuration Data
  - Reconfigurable Processor
  - HW
  - Configuration Setting
Development Tool Chain

Functional Requirement Specification
  --------
  C Source Code
  --------
ISA Generator (w/Source Transformer)
  --------
  Transformed C Code
  --------
Retargettable Compiler (Object Generator)
  --------
  Data
  --------
  Assembly Code
  --------
Assembler
  --------
  Object code (Exe or Dll)
  --------
Vulcan
  --------
Vulcan ISS
  --------
  Result / Profile
  --------
Instruction Creator
  --------
Inst. Library (XML or VHDL)
  --------
  ISA
  --------
Place and Route
  --------
  Result File (Hard-mapped result)
  --------
Pre-defined Inst. Lib.
Demo
DES: A Redefis Application

1. **Ciphertext (64 bits)**
   - **Initial Permutation (IP)**
     - **XOR**
     - **F-Function**
     - **XOR**
     - 16 steps
     - **Inverse Initial Permutation (IP^-1)**
     - **Ciphertext (64 bits)**

2. **Key (56 bits)**
   - **Create 48-bit Keys**
   - **F-Function**
   - **Extension/Permutation**
     - **XOR**
     - **S-BOX**
     - **Permutation**
     - **Ciphertext (32 bits)**
   - **Key (48 bits)**
DES: A Redefis Application

- Object code size: 24 instructions
- Dynamic instruction count: 35 instructions
- Vulcan (6.25MHz) vs. P4 (2.4GHz):
  - Throughput: 570KB/s vs. 150KB/s

<table>
<thead>
<tr>
<th>Instruction</th>
<th>What to do</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Read key and permute it (PC-1)</td>
</tr>
<tr>
<td>1</td>
<td>Read cleartext and permute it (IP)</td>
</tr>
<tr>
<td>2</td>
<td>1-bit left rotate shift (LS1)</td>
</tr>
<tr>
<td>3</td>
<td>2-bit left rotate shift (LS2)</td>
</tr>
<tr>
<td>4</td>
<td>Permutation (PC-2), F-function, and XOR</td>
</tr>
<tr>
<td>5</td>
<td>Inverse initial permutation (IP-1), and output ciphertext</td>
</tr>
</tbody>
</table>

Cleartext (64 bits) → Key (56 bits) → Instruction 1 → Instruction 0 → Instruction 2 or 3 → Instruction 4 → Loop=16? → Instruction 5 → Ciphertext (64 bits)