Co-design of Reconfigurable FPGA System

李宗演 (Trong-Yen Lee)
電子系暨電腦通訊研究所
國立台北科技大學
tylee@ntut.edu.tw

Outline

- Introduction
- Reconfigurable Systems
- Partitioning and Scheduling
- Operating System for Reconfigurable Systems
- Design Example
- Conclusions
Outline

- Introduction
  - Reconfigurable Systems
  - Partitioning and Scheduling
  - Operating System for Reconfigurable Systems
  - Design Example
  - Conclusions

SoC Design Trend

System-on-Board (SoB)  System-on-Chip (SoC)
The Next Design Challenge: SoC

- System-on-Chip
  - IP cores
  - Lots of software
- How do you design such systems?
- Codesign!!!

SGS-Thomson Videophone

<table>
<thead>
<tr>
<th>DSP core 1 (ST18950):</th>
</tr>
</thead>
<tbody>
<tr>
<td>modem</td>
</tr>
<tr>
<td>Sound codec</td>
</tr>
<tr>
<td>MCU 1 (ASIP):</td>
</tr>
<tr>
<td>Master control</td>
</tr>
<tr>
<td>MCU 2 (ASIP): Mem. Controller</td>
</tr>
<tr>
<td>MCU 3 (ASIP): Bit manip.</td>
</tr>
<tr>
<td>VLIV DSP: Programmable video operations std. extensions</td>
</tr>
<tr>
<td>High-speed H/W: Video operators for DCT, inverse DCT, motion estimation</td>
</tr>
<tr>
<td>Memory: Video RAM</td>
</tr>
<tr>
<td>Glue logic</td>
</tr>
<tr>
<td>A/D &amp; D/A</td>
</tr>
<tr>
<td>I/O: Serial interface</td>
</tr>
<tr>
<td>I/O: Host interface</td>
</tr>
</tbody>
</table>

Hardware | Embedded Real-time Software

System Design

Flexibility | Performance

Microprocessor

Reconfigurable SoC

ASIP and Configurable µp

ASIC

2007/03/27 NTU
Codesign Definition and Key Concepts

- **Codesign Definition**
  - The meeting of system-level objectives by exploiting the trade-offs between hardware and software in a system through their concurrent design

- **Key Concepts**
  - **Concurrent**: hardware and software developed at the same time on parallel paths
  - **Integrated**: interaction between hardware and software developments to produce designs that meet performance criteria and functional specifications
Motivations for Co-design

- Factors driving co-design (hardware/software systems):
  - Instruction Set Processors (ISPs) available as cores in many design kits
  - Systems on Silicon
  - Increasing capacity of field programmable devices (FPGAs, CPLDs, etc.)
  - Efficient C compilers for embedded processors
  - Hardware synthesis capabilities
Motivations for Co-design (cont’d)

- The importance of co-design in designing hardware/software systems:
  - **Improves** design quality, design cycle time, and cost
  - **Supports** growing complexity of embedded systems
  - **Takes advantage of** advances in tools and technologies
Co-design Flow (cont’d)

- Scheduling
  - Hardware Tasks
  - Software Tasks
- Cosynthesis
- Verification
- End

Co-design for Reconfigurable Systems

- Application Stage
  - Discrete Event System Specification
  - Design Constraints
- Static Stage
  - Discrete Event Class & Object Extraction
  - Discrete Event Class Estimation
  - HW/SW Class Partitioning
- Dynamic Stage
  - HW/Synthesis
  - SW Synthesis
  - HW/SW Scheduling
  - DRL Multi-context Scheduling
Reconfigurable Systems

- RAM or FLASH memory-based FPGA/CPLD can be dynamically reconfigured.
- Its configuration can be changed while the rest of the circuit is fully operational.
- Two techniques used to reduce the configuration time are:
  - Partial configuration
  - Multiple-context configuration memory
Overview -- Categories of Programmable Logic Devices (PLDs)

- Simple Programmable Logic Devices (SPLDs)
- Complex Programmable Logic Devices (CPLDs)
- Field Programmable Gate Arrays (FPGAs)

SPLD Device Overview

- Simplest PLD (SPLD) devices
  - PAL: Programmable Array Logic
  - PLA: Programmable Logic Array
    - Two logic gate array architectures
    - Boolean ANDs and ORs
CPLD Device Overview

- CPLD: Complex Programmable Logic Device
- Complexity and density:
  - FPGAs > CPLDs > SPLDs
Field-programmable Gate Arrays

- FPGAs are programmable logic devices:
  - Logic elements + interconnect.
  - Provide multi-level logic.

Connection Blocks:
- Provide programmable multiplexers, selecting which of the signals in the given routing channel will be connected to the logic block’s terminals.

Switch Boxes:
- There are connections between the horizontal and vertical routing resources to allow signals to change their routing direction.
FPGA Manufacture

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Altera®</td>
<td>SRAM, Flash</td>
</tr>
<tr>
<td>Actel</td>
<td>Antifuse</td>
</tr>
<tr>
<td>Lattice</td>
<td>SRAM, Flash</td>
</tr>
<tr>
<td>Quicklogic</td>
<td>Antifuse</td>
</tr>
<tr>
<td>Xilinx</td>
<td>SRAM</td>
</tr>
</tbody>
</table>

2007/03/27 NTU

Popular Reconfigurable Logics

- Two main architectures of reconfigurable logic (FPGA)
  - **Altera**
    - Logic Array Blocks of **Logic Elements** (LE)
    - **ARM 922T** core in embedded stripe
    - **Nios**
  - **Xilinx**
    - Slice of **Configurable Logic Blocks** (CLB)
    - 4 **PowerPC 405** hard cores
    - **MicroBlaze**
Partial/Dynamical Reconfigurable

Partial / dynamic reconfiguration is the ability to update only a portion of the configuration memory in a FPGA with a new configuration without stopping the functionality of unchanged sections of the FPGA.

Reconfigurable Models

- Single Context
- Multi-context
- Partially reconfigurable
Single Context

- Most commercial FPGAs are of this variety
- To implement runtime reconfiguration on this type of device, configurations must be grouped into full contexts
- Complete contexts are swapped in and out of the hardware as needed

Multi-context

- Multiple layers of programming bits, where each layer can be active at a different point in time
- Allows for an extremely fast context switch (on the order of nanoseconds)
- Allow for background loading, permitting one context to be configuring while another is in execution
Partially Reconfigurable

- **Selectively** programmed without a complete reconfiguration
- **Small areas** of the array can be modified without requiring that the **entire logic array** be reprogrammed
- Require **much less time** than a full-chip reconfiguration due to the **reduced data traffic**

FPGA Self-Reconfiguration Architecture
Task Context Relocation Architecture

- Fixed module
  - Controller
  - Command register
  - ICAP
  - Filter
  - Memory
  - Task sel

- Reconfigurable module
  - Module A
  - Module B

Task Context Saving and Restoring Architecture

- Task Registers Database
- Controller Configuration / Readback
- PROM
  - task.bit

- CPLD
- FPGA
  - Reconfiguration System
  - SelectMAP

Load Database
SelectMAP
**Database Generation**

![Database Generation Diagram](image)

**Dynamically Partially Reconfiguration Architecture**

![Dynamically Partially Reconfiguration Architecture Diagram](image)
Task Context Saving and Restoring Flow

Begin

Select Task module

Task stop

Swap out? Y N

Configuration

Readback

Restoring context

Saving context

Task start

End

Reconfigurable System Example

...
Outline

- Introduction
- Reconfigurable Systems
  - Partitioning and Scheduling
- Operating System for Reconfigurable Systems
- Design Example
- Conclusions

Relations Among Partitioning Issues

High Level Abstraction

Decomposition of functional objects
- Metrics and estimations
- Partitioning algorithms
- Objective and closeness functions

Component allocation

Output
Functional and Temporal Partitioning

- **Functional Partitioning**
  - Mapping an application model to an RSoC architecture model
  - Example: UML profile for SystemC

- **Temporal Partitioning**
  - Large circuit is partitioned to fit an FPGA
  - The partial circuits *temporally* share an FPGA

---

Functional Partitioning

- **Network Flow Graph**
  - Optimal solution, slow, not practical

- **Network Clubbing**
  - Heuristic solution, practical, profiling

- **Simulated Annealing**
  - Heuristic sub-optimal solution, fast, practical

- **Genetic Algorithm**
  - Heuristic good solution, slow, practical
Temporal Partitioning

- List Scheduling
- Force-Directed Scheduling
- Network Flow Graph
- Integer Linear Programming
- Hierarchical Flow

Partitioning Example
Source Code Function Analysis

![Diagram showing the flow of encoded data through various functions and files]

Profiling

- **Step 1**
  - Build a **Makefile as batch** to compile the source code.
- **Step 2**
  - Adding **–pg** parameter.
- **Step 3**
  - Using **make** command to compile then execute the created executable file.
- **Step 4**
  - Type **gprof** to create the profiling table.
## Profiling Table 1

- **foreman.mpg**
- **352x288 pixels**
- **50 frames**
- **2 seconds**
- **25f/s**

<table>
<thead>
<tr>
<th>% time</th>
<th>Cumulative seconds</th>
<th>Self seconds</th>
<th>calls</th>
<th>Self ms/call</th>
<th>Total ms/call</th>
<th>function name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>10144</td>
<td>0</td>
<td>0</td>
<td>mac_motion_eucles</td>
</tr>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>17584</td>
<td>0</td>
<td>0</td>
<td>reconstruct</td>
</tr>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>13236</td>
<td>0</td>
<td>0</td>
<td>getintrablack</td>
</tr>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>12975</td>
<td>0</td>
<td>0</td>
<td>getMBA</td>
</tr>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>8824</td>
<td>0</td>
<td>0</td>
<td>getDClum</td>
</tr>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>4412</td>
<td>0</td>
<td>0</td>
<td>getDClchrom</td>
</tr>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>1014</td>
<td>0</td>
<td>0</td>
<td>startcode</td>
</tr>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>900</td>
<td>0</td>
<td>0</td>
<td>getintrablack</td>
</tr>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>56</td>
<td>0</td>
<td>0</td>
<td>ext_user_data</td>
</tr>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>51</td>
<td>0</td>
<td>0</td>
<td>gethdr</td>
</tr>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>51</td>
<td>0</td>
<td>0</td>
<td>gethdr</td>
</tr>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>0</td>
<td>ext_bit_info</td>
</tr>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>1.4</td>
<td>store_env</td>
</tr>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>1.4</td>
<td>store_env</td>
</tr>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>getavail</td>
</tr>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>init_idct</td>
</tr>
<tr>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1.4</td>
<td>putlast</td>
</tr>
</tbody>
</table>

## Profiling Table 2

- **foreman.mpg**
- **352x288 pixels**
- **300 frames**
- **12 seconds**
- **25f/s**

<table>
<thead>
<tr>
<th>% time</th>
<th>Cumulative seconds</th>
<th>Self seconds</th>
<th>calls</th>
<th>Self ms/call</th>
<th>Total ms/call</th>
<th>function name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>436143</td>
<td>0</td>
<td>0</td>
<td>getMBA</td>
</tr>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>280334</td>
<td>0</td>
<td>0</td>
<td>calcMV</td>
</tr>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>216307</td>
<td>0</td>
<td>0</td>
<td>TBV</td>
</tr>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>114915</td>
<td>0</td>
<td>0</td>
<td>getMBA</td>
</tr>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>114915</td>
<td>0</td>
<td>0</td>
<td>getMBtype</td>
</tr>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>105132</td>
<td>0</td>
<td>0</td>
<td>getintrablack</td>
</tr>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>35044</td>
<td>0</td>
<td>0</td>
<td>getDClchrom</td>
</tr>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>6056</td>
<td>0</td>
<td>0</td>
<td>startcode</td>
</tr>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>5400</td>
<td>0</td>
<td>0</td>
<td>getintrablack</td>
</tr>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>527</td>
<td>0</td>
<td>0</td>
<td>ext_user_data</td>
</tr>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>301</td>
<td>0</td>
<td>0</td>
<td>gethdr</td>
</tr>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>301</td>
<td>0</td>
<td>0</td>
<td>getheader</td>
</tr>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>300</td>
<td>0</td>
<td>1.23</td>
<td>store_env</td>
</tr>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>300</td>
<td>0</td>
<td>1.23</td>
<td>store_env</td>
</tr>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>getavail</td>
</tr>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>init_idct</td>
</tr>
<tr>
<td>0</td>
<td>1.77</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1.23</td>
<td>putlast</td>
</tr>
</tbody>
</table>
### Profiling Table 3

- **101_test.mpg**
- **352x288 pixels**
- **1608 frames**
- **64 seconds**
- **25f/s**

<table>
<thead>
<tr>
<th>% time</th>
<th>Cumulative seconds</th>
<th>Self seconds</th>
<th>calls</th>
<th>Self ms/call</th>
<th>Total ms/call</th>
<th>function name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.13</td>
<td>7.48</td>
<td>0.01</td>
<td>265584</td>
<td>0</td>
<td>0</td>
<td>macroblock_move</td>
</tr>
<tr>
<td>0.13</td>
<td>7.49</td>
<td>0.01</td>
<td>152269</td>
<td>0</td>
<td>0</td>
<td>getSBP</td>
</tr>
<tr>
<td>0.13</td>
<td>7.50</td>
<td>0.01</td>
<td>96472</td>
<td>0</td>
<td>0</td>
<td>getOCClean</td>
</tr>
<tr>
<td>0</td>
<td>7.53</td>
<td>0.00</td>
<td>756200</td>
<td>0</td>
<td>0</td>
<td>getblk</td>
</tr>
<tr>
<td>0</td>
<td>7.57</td>
<td>0.00</td>
<td>329382</td>
<td>0</td>
<td>0</td>
<td>setHRV</td>
</tr>
<tr>
<td>0</td>
<td>7.64</td>
<td>0.00</td>
<td>265584</td>
<td>0</td>
<td>0</td>
<td>putMBtype</td>
</tr>
<tr>
<td>0</td>
<td>7.75</td>
<td>0.00</td>
<td>264796</td>
<td>0</td>
<td>0</td>
<td>motion_vector</td>
</tr>
<tr>
<td>0</td>
<td>7.80</td>
<td>0.00</td>
<td>180944</td>
<td>0</td>
<td>0</td>
<td>getDClm</td>
</tr>
<tr>
<td>0</td>
<td>7.80</td>
<td>0.01</td>
<td>18100</td>
<td>0</td>
<td>0</td>
<td>ext_user_data</td>
</tr>
<tr>
<td>0</td>
<td>7.81</td>
<td>0.01</td>
<td>14088</td>
<td>0</td>
<td>0</td>
<td>putlast</td>
</tr>
<tr>
<td>0</td>
<td>7.81</td>
<td>0.01</td>
<td>14088</td>
<td>0</td>
<td>0</td>
<td>getheader</td>
</tr>
<tr>
<td>0</td>
<td>7.81</td>
<td>0.01</td>
<td>1407</td>
<td>0</td>
<td>0</td>
<td>getblocks</td>
</tr>
<tr>
<td>0</td>
<td>7.82</td>
<td>0.00</td>
<td>1407</td>
<td>0</td>
<td>0</td>
<td>ext_user_hdr</td>
</tr>
<tr>
<td>0</td>
<td>7.84</td>
<td>0.00</td>
<td>1407</td>
<td>0</td>
<td>0</td>
<td>store_one</td>
</tr>
<tr>
<td>0</td>
<td>7.85</td>
<td>0.00</td>
<td>1407</td>
<td>0</td>
<td>0</td>
<td>storeframe</td>
</tr>
<tr>
<td>0</td>
<td>7.86</td>
<td>0.00</td>
<td>1407</td>
<td>0</td>
<td>0</td>
<td>yuv_data</td>
</tr>
<tr>
<td>0</td>
<td>7.86</td>
<td>0.00</td>
<td>1407</td>
<td>0</td>
<td>0</td>
<td>initl</td>
</tr>
<tr>
<td>0</td>
<td>7.86</td>
<td>0.00</td>
<td>1407</td>
<td>0</td>
<td>0</td>
<td>pristine</td>
</tr>
</tbody>
</table>

### Table 1 Profiling Result

(352X288,50frames,25f/s)

<table>
<thead>
<tr>
<th>Decoder Component</th>
<th>Source Code</th>
<th>% Time</th>
<th>Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coded Data Input</td>
<td>getblk.c</td>
<td>4.55</td>
<td>2220905</td>
</tr>
<tr>
<td></td>
<td>getvcl.c</td>
<td>0.00</td>
<td>113214</td>
</tr>
<tr>
<td></td>
<td>getvcl.h</td>
<td>0.00</td>
<td>51024</td>
</tr>
<tr>
<td>Inverse Scan</td>
<td>getblk.c</td>
<td>0.00</td>
<td>118801</td>
</tr>
<tr>
<td></td>
<td>idct.c</td>
<td>31.82</td>
<td>297595</td>
</tr>
<tr>
<td></td>
<td>idstref.c</td>
<td>31.82</td>
<td>297595</td>
</tr>
<tr>
<td></td>
<td>motion.c</td>
<td>31.82</td>
<td>297595</td>
</tr>
<tr>
<td></td>
<td>recon.c</td>
<td>31.82</td>
<td>297595</td>
</tr>
<tr>
<td></td>
<td>spatscal.c</td>
<td>31.82</td>
<td>297595</td>
</tr>
<tr>
<td>Frame-Store Memory</td>
<td>store.c</td>
<td>4.55</td>
<td>7603450</td>
</tr>
<tr>
<td>Decoded Samples &amp; Display</td>
<td>getpic.c</td>
<td>27.27</td>
<td>138035</td>
</tr>
</tbody>
</table>

**Total Running Time**: 0.27 seconds

2007/03/27 NTU
### Table 2 Profiling Result
**(352x288, 300 frames, 25f/s)**

<table>
<thead>
<tr>
<th>Decoder Component</th>
<th>Source Code</th>
<th>% Time</th>
<th>Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coded Data Input</td>
<td>getbits.c</td>
<td>6.07</td>
<td>13198148</td>
</tr>
<tr>
<td>Variable Length Decoding</td>
<td>getvlc.c</td>
<td>2.25</td>
<td>692957</td>
</tr>
<tr>
<td>Inverse Scan</td>
<td>getblk.c</td>
<td>4.83</td>
<td>342925</td>
</tr>
<tr>
<td>Inverse Quantization</td>
<td>idct.c</td>
<td>29.29</td>
<td>1055726</td>
</tr>
<tr>
<td>Inverse DCT</td>
<td>idctref.c</td>
<td>29.29</td>
<td></td>
</tr>
<tr>
<td>Motion Compensation</td>
<td>motion.c</td>
<td>35.98</td>
<td>1718631</td>
</tr>
<tr>
<td>Frame-store Memory</td>
<td>store.c</td>
<td>5.21</td>
<td>45620700</td>
</tr>
<tr>
<td>Decoded Samples &amp; Display</td>
<td>getpic.c</td>
<td>16.37</td>
<td>828016</td>
</tr>
<tr>
<td></td>
<td>display.c</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Total Running Time:** 1.77 seconds

---

### Table 3 Profiling Result
**(352x288, 1608 frames, 25f/s)**

<table>
<thead>
<tr>
<th>Decoder Component</th>
<th>Source Code</th>
<th>% Time</th>
<th>Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coded Data Input</td>
<td>getbits.c</td>
<td>6.10</td>
<td>67198148</td>
</tr>
<tr>
<td>Variable Length Decoding</td>
<td>getvlc.c</td>
<td>3.10</td>
<td>3665634</td>
</tr>
<tr>
<td>Inverse Scan</td>
<td>getblk.c</td>
<td>5.02</td>
<td>1863452</td>
</tr>
<tr>
<td>Inverse Quantization</td>
<td>idct.c</td>
<td>31.60</td>
<td>5318232</td>
</tr>
<tr>
<td>Inverse DCT</td>
<td>idctref.c</td>
<td>31.60</td>
<td></td>
</tr>
<tr>
<td>Motion Compensation</td>
<td>motion.c</td>
<td>30.79</td>
<td>4217184</td>
</tr>
<tr>
<td>Frame-store Memory</td>
<td>store.c</td>
<td>6.21</td>
<td>221520355</td>
</tr>
<tr>
<td>Decoded Samples &amp; Display</td>
<td>getpic.c</td>
<td>16.92</td>
<td>4523052</td>
</tr>
<tr>
<td></td>
<td>display.c</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Total Running Time:** 7.50 seconds
Partition Flow

- Function Block
- Profiling Table
- Select BBs execution time
- Decision
- BB ordering & selection
- HW
- SW

2007/03/27 NTU

Partition Result

- Function Block
- Profiling Table
- Select BBs execution time
- Decision
- BB ordering & selection
- HW
- SW
- Final Report
- Coded Data Input
- Variable Length Decoding
- Inverse Scan
- Inverse Quantization
- Frame-store Memory
- Decoded Samples & Display

2007/03/27 NTU
Performance Analysis

![Graph showing performance analysis](image)

Static and Dynamic Scheduling

- **Static Scheduling**
  - Schedule fixed **before run-time**
  - Performed by **CAD synthesis tools**
    - Determine **task start time**
    - Determine **context configuration time**
    - With placement / Without placement
  - **More optimal** solutions than dynamic scheduling
    - Requires **longer time** than dynamic scheduling
  - **Inflexible**

2007/03/27 NTU

2007/03/27 NTU
Static and Dynamic Scheduling

- **Dynamic Scheduling**
  - **Run-time** scheduling
  - Performed by OS4RS (OS for Reconfig. Sys.)
    - Need hardware task information
      - Execution time, deadline, priority, ...
      - Area size, configuration time, locatability,
  - Faster than static scheduling
  - Sub-optimal solutions
  - Flexible

Static Scheduling

- **Random Priority** List Scheduling

Reconfigurable Environment Scheduling Model
Dynamic Scheduling

- Dynamic scheduling methods
  - Priority based
  - Non-preemptive
  - Preemptive
  - Horizon and Stuffing

Dynamic Scheduling

Task and Context Schedulers
Dynamic Scheduling

Architecture Model

Online Scheduler

Dynamically Reconfigurable Resource Models

Eg: Xilinx Virtex II Pro

Eg: Atmel AT40K
Independence Task Scheduling

- Partial reconfigurable architectures are useful for applications with multi-function in one chip solution.

- To optimize the task scheduling can save space, consumption and increase efficiency.
Traditional Task Architecture (Cont’d)

Task 1 and Task 5 executed same function

Independence Task Architecture

Task 1 and Task 5 executed same function

1. Considering that the task(F1) is independence
2. The area needs of T1(F1) task + any task area listed in the task window \( \leq \) HW area
3. The next reconfigured area + previous executed task area + T1(F1) area > HW area

Exchange task 5 and task 4
Change priority of task 5
The Benefit of Independence Task Architecture

Example of Independence Task Architecture

- One Slot for Multi-Peripheral in Embedded System Architecture
Outline

- Introduction
- Reconfigurable Systems
- Partitioning and Scheduling
  - Operating System for Reconfigurable Systems
- Design Example
- Conclusions

Operation System for Reconfigurable Systems (OS4RS)

- Mobile, wireless multimedia applications demand more dynamic performance and pervasiveness
- State-of-art FPGA chips allow hardware tasks to be changed at run-time
- An OS is thus needed for managing all those hardware tasks efficiently and correctly
  - Allocation, partitioning, scheduling, placement, routing, relocating, etc.
OS4RS … Research Issues

Operating System for Reconfigurable Systems (OS4RS)

Services
- Loading
- Partitioning
- Scheduling
- Routing

Components
- Loader
- Partitioner
- Scheduler
- Router

Abstractions
- Relocation
- Application
- HW/SW Comm
- HW/HW Comm

Performance
- Overhead
- Application
- Fragmentation
- Portability

Cam-E-leon:
A Run-Time Reconfigurable Web Camera
OS4RS … HW support

- Communication infrastructure
  - Standard Bus
  - Network-on-Chip (NoC)
- Scheduler support
  - hardware scheduler
- Synchronization support
  - hardware semaphores, etc.
- Memory management
  - configuration storage, readback, …

Outline

- Introduction
- Reconfigurable Systems
- Partitioning and Scheduling
- Operating System for Reconfigurable Systems
  - Design Example
- Conclusions
Module-Based Partial Reconfiguration (PR)

- Counter Design Tutorial

Reconfiguration Modules in Xilinx FPGA
Bus Macro Structure

- 8 Tri-state buffers

Counter Design Architecture

- A basic schematic of the design
Partial Reconfigurable Design Flow

Recommended Project Directory Structure

- Directory structure used for a modular design project
  - Implementation of independent modules: translate, map, place & route
  - Top level initial budgeting and final assembly directories
  - Pre-routed bus macro (hard macro)
  - VHDL source code
  - Independent module design and synthesis ISE projects
Module-Based PR Design Flow

- Design entry (Top-level design)
  - HDL entry / synthesis (use ISE)
  - Synthesis options (in ISE)
    - Add I/O buffer (for top-level): Yes
    - Keep Hierarchy: Soft
- Design entry (Independent module design)
  - HDL entry / synthesis (use ISE)
  - One ISE project for each module
  - Synthesis options (in ISE)
    - Add I/O buffer (for modules): No
    - Keep Hierarchy: Yes
- ISE projects are created in synthesis/ directory
Module-Based PR Design Flow (cont’d)

- **Initial budgeting**
  - Change to directory top/initial/
  - Copy the synthesized top.ngc and bus macros here
  - Run `ngdbuild -modular initial top.ngc`
- **Setup user constraints (.ucf) if needed**
- **Floorplanning: set area constraints w/ Floorplanner**
  - Take care of the reconfig. module requirements
    - Full column module height
    - Min. module width 4 columns
    - Boundaries fall on X=0,4,8, etc.
    - In this example, the boundary is at X=20
  - For our board, assign clk to pin p80 and result to p44

Manually modify .ucf (top.ucf)

- Add in the ucf
  ```
  AREA_GROUP "AG_increm" MODE = RECONFIG;
  AREA_GROUP "AG_myReg" MODE = RECONFIG;
  INST "busRegToInc/bus1" LOC = "TBUF_R18C16.0";
  INST "busRegToInc/bus2" LOC = "TBUF_R20C16.0";
  INST "busIncToReg/bus1" LOC = "TBUF_R22C16.0";
  INST "busIncToReg/bus2" LOC = "TBUF_R24C16.0";
  ```
- Bus macro has to start at 4 columns left of boundary e.g., X=16 when boundary at X=20
- Bus macro must start at even row (Y=even)
- Run `ngdbuild -modular initial top.ngc` to produce a new ngd to be loaded into Floorplanner for verification
Module-Based PR Design Flow (cont’d)

- XC2S200E PQ208 floorplanning
- Floorplanning: set area constraints, IOBs …

Active module implementation
- Change to the module dir at modules/[module]/
- Copy the synthesized .ngc for each module from synthesis/[module]/
- If module-specific constraints are needed, copy the previously produced top.ucf into the directory and add constraints
- Implement and publish each module
  - ngdbuild -modular module -active 
    incrementer ..\..\initial\top.ngc
  - map top.ngd
  - par -w top.ncd top1.ncd
  - pimcreate -ncd top1.ncd ..\..\pims
Module-Based PR Design Flow (cont’d)

- Final assembly
  - Change dir to top/final/
  - Copy bus macros here
  - Assemble the modules
    - ngdbuild -modular assemble -pimpath ..\..\pims ..\initial\top.ngdgc
    - map top.ngd
    - par -w top.ncd top_routed.ncd

Module-Based PR Design Flow (cont’d)

- Verify the design
  - Use FPGA Editor to view the final P&R result
  - See if the modules only connect at bus macros
Module-Based PR Design Flow (cont’d)

- Creating bitstreams
  - Initial bitstream of the complete design
    - Change to dir top/final/
    - `bitgen -w top_routed.ncd top_routed.bit`
  - Partial bitstream for each module
    - Change to the dir of each module (modules/[module]/)
    - `bitgen -g ActiveReconfig:Yes -d top1.ncd top_partial.bit`

Module-Based PR Design Flow (cont’d)

- Configure the real board
  - Download bitstreams with iMPACT to XC2S200e
  - When the board is running, download a partial bitstream to partially reconfig. the chip
Implementation Results

- FPGA Routing

Implementation Results (cont’d)

- Load Full Bitstream to FPGA
  - Top level bitstream
  - Counter up/down
- Load Partial Bitstream to FPGA
  - Incremementer bitstream
  - Myregister bitstream
Outline

- Introduction
- Reconfigurable Systems
- Partitioning and Scheduling
- Operating System for Reconfigurable Systems
- Design Example
  - Conclusions

Conclusions

- Reconfigurable systems will be a promising system design architecture for future powerful applications
- There are lots of research issues still left open in this field
- There are lots of practical issues still left unsolved in this field
Thank you for your attention!!

Q & A