A Comparative Study of Modulo Scheduling Techniques

Josep M. Codina, Josep Llosa and Antonio González

Dept. of Computer Architecture
Universitat Politècnica de Catalunya
Barcelona, SPAIN

E-mail: {jmcodina,josepll,antonio}@ac.upc.es
Software Pipelining

- **Instruction Scheduling for VLIW/Superscalar Processors**
  - VLIW processors in DSP market
  - EPIC/IPF

- **Loop Scheduling: Software Pipelining**
  - Loops consume most of the application’s execution time

- **Software Pipelining** a loop is an NP-complete problem

- **Software Pipelining** big family of techniques
  - Modulo Scheduling based on heuristics
Motivation

- Modulo Scheduling is an environment to define techniques
  - Different factors to take into account
  - Lot of techniques can fit in the environment. Different ideas

- Proposals in the literature evaluated without common
  - Platform (i.e. compiler)
  - Benchmarks
  - Target architectures
  - Measures

- Lack of a thorough comparison
Objectives

- Perform a comparison of state-of-the-art MS techniques
  - Qualitative
  - Quantitative

- The work is target to compiler writers
  - Is one of the techniques better than the others for all architectures?
  - Which is the most powerful technique for a given architecture?
Talk Outline

- Modulo Scheduling Background
- Selection Criteria
- Techniques Compared
- Study Environment
- Results
- Conclusions
Talk Outline

- Modulo Scheduling Background
- Selection Criteria
- Techniques Compared
- Study Environment
- Results
- Conclusions
Basic Ideas

UPC

MODULO SCHEDULING

Initiation Interval (II)

Stage 1
Stage 2
Stage 3
Stage 2
Stage 1

Iteration 1
Iteration 2
Iteration 3
Iteration 4

Prolog
Kernel
Epilog
Basic Scheme

Find MII and Set II=MII

Look for a schedule

Found it ?

Si

No

Increase the II
Basic Scheme

Find MII and Set II=MII

Look for a schedule

Found it?  

- Si
  - Yes
  - No

MII depends on

- Resources
- Recurrences

Increase the II
Basic Scheme

Find MII and Set II=MII

Look for a schedule

- Ordering the nodes
- Finding a feasible cycle
  - Top-Down/Bottom-up
  - Bi-directional
- When no feasible cycle
  - Use of backtracking
  - Increase the II

Found it? 

Si
Basic Scheme

Find MII and Set II = MII

Look for a schedule

Found it ?

Can we meet the constraints?
- Resources
- Dependences

Increase the II

Si

No
Basic Scheme

- The larger the II, the more likely to find a schedule
- The larger the II, the lower the performance
- II lower than the length of a single iteration

[Diagram showing the flow of the scheme with decision points and actions such as 'Found it?', 'No', 'Increase the II', 'Si.']
Backtracking

- Not always beneficial
  - Can produce better schedules
  - Can just increase the process of finding a schedule
- In some cases, no feasible schedule for a given II

\[
\text{Backtracking must be limited}
\]

- BudgetRatio:
  Ratio of the maximum number of operation scheduling steps attempted before increasing the II
Talk Outline

- Modulo Scheduling Background
- Selection Criteria
- Techniques Compared
- Study Environment
- Results
- Conclusions
- **Value of the code generated**
  - Parallelism
  - Register pressure
  - Code size
  - Execution time

- **Effectiveness/Cost of the technique**
- **Value of the code generated**
  - Parallelism
  - Register pressure
  - Code size
  - Execution time

- **Effectiveness/Cost of the technique**
  - Effectiveness on exploiting ILP
  - What is the difference between II and MII?
<table>
<thead>
<tr>
<th>Selection Criteria</th>
<th>Value of the code generated</th>
<th>Effectiveness/Cost of the technique</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parallelism</td>
<td>Software pipelining puts high</td>
</tr>
<tr>
<td></td>
<td>Register pressure</td>
<td>demands on register pressure</td>
</tr>
<tr>
<td></td>
<td>Code size</td>
<td>How many regs are needed?</td>
</tr>
<tr>
<td></td>
<td>Execution time</td>
<td>How many loops within a given</td>
</tr>
<tr>
<td></td>
<td></td>
<td>number of registers?</td>
</tr>
</tbody>
</table>
- **Value of the code generated**
  - Parallelism
  - Register pressure
  - Code size
  - Execution time

- **Effectiveness/Cost of the technique**
  - Crucial in embedded domains
  - Stages of a schedule
- **Value of the code generated**
  - Parallelism
  - Register pressure
  - Code size
  - Execution time

- **Effectiveness/Cost of the technique**
- **Value of the code generated**
  - Parallelism
  - Register pressure
  - Code size
  - Execution time

- **Effectiveness/Cost of the technique**
  - Can all the loops be scheduled?
  - Compilation time
Talk Outline

- Modulo Scheduling Background
- Selection Criteria
- Techniques Compared
- Study Environment
- Results
- Conclusions
Techniques

Modulo Scheduling Techniques

- Iterative Modulo Scheduling (IMS)
- Swing Modulo Scheduling (SMS)
- Slack Modulo Scheduling (Slack MS)
- Integrated Register-sensitive Iterative Software Pipelining method (IRIS)

Complementary techniques

- Stage Modulo Scheduling (Stage MS)

**Stage MS**

- Post-pass that can be applied after a MS technique
- To reduce the Register Pressure
- Without increasing the II
- Moves operations by II
- Various heuristics. We selected 3UP+RSS heuristic
## Main Differences

<table>
<thead>
<tr>
<th></th>
<th>IMS</th>
<th>SMS</th>
<th>Slack MS</th>
<th>IRIS</th>
</tr>
</thead>
</table>
| **Order of nodes**     | Top-Down| •Priority to recurrences  
                         |         | •No pred. and succ.
                         |         | scheduled in partial schedule |
|                        | Slack MS | IRIS   | Slack MS | IRIS               |
| **Finding a cycle**    | Top-Down| •Bi-directional  
                         |         | •Close to pred or succ depending on the benefit |
|                        | Slack MS | IRIS   | Slack MS | IRIS               |
| **Backtracking**       | Yes     | No      | Yes      | Yes                |

**Techniques Compared**

- **IMS**
- **SMS**
- **Slack MS**
- **IRIS**
<table>
<thead>
<tr>
<th></th>
<th>IMS</th>
<th>SMS</th>
<th>Slack MS</th>
<th>IRIS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parallelism</strong></td>
<td>Backtracking</td>
<td>Order</td>
<td>• Order</td>
<td>Backtracking</td>
</tr>
<tr>
<td><strong>Register Pressure</strong></td>
<td>No</td>
<td>•Order</td>
<td>Bi-directional</td>
<td>Stage Heuristics</td>
</tr>
<tr>
<td><strong>Code Size</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Effectiveness</strong></td>
<td>Backtracking</td>
<td>Backtracking</td>
<td>Backtracking</td>
<td>Backtracking</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td></td>
<td>No Backtracking</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- **Platform (i.e. compiler)**
  - ICTINEO

- **Benchmarks**
  - SPECfp95
  - Perfect Club

- **Target architectures**
  - Some architectures varying the complexity
    - Low Complexity architecture
    - Medium Complexity architecture
    - Complex architecture

- **1936 loops**

- **Less Constrained**
  - More Constrained
## Architectures Description

<table>
<thead>
<tr>
<th></th>
<th>Low Complexity</th>
<th>Medium Complexity</th>
<th>Complex Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fully Pipelined Simple ops</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Non-Pipelined Complex ops</td>
</tr>
<tr>
<td>Fully Pipelined ops</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-Issue</td>
<td>4-Issue</td>
<td>4 write-ports</td>
<td></td>
</tr>
<tr>
<td>Unlimited register ports</td>
<td></td>
<td>8 read-ports</td>
<td></td>
</tr>
<tr>
<td>2 memory ports</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 Int FU and 2 FP FU</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Latencies

<table>
<thead>
<tr>
<th>Latencies</th>
<th>Low/Medium</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>INT ADD, SUB, COMP</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>INT MUL</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>INT DIV, MOD, SQRT</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>FP ADD, SUB, COMP</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>FP MUL</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>FP DIV, MOD, SQRT</td>
<td>18</td>
<td>20</td>
</tr>
</tbody>
</table>
Methodology

- Study of the Budget Ratio for each architecture: 1, 2.5, 5 and 10
  - Effectiveness
  - Performance
  - Cost

- Measures for each technique with and without Stage MS
  - Effectiveness and cost
  - Parallelism
  - Register pressure
  - Code size
  - Execution
Modulo Scheduling Background

Selection Criteria

Techniques Compared

Study Environment

Results

Conclusions
## Budget Ratio Study

### Low Complexity Architecture

<table>
<thead>
<tr>
<th>Low Complexity</th>
<th>Medium Complexity</th>
<th>Complex Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>2.5</td>
</tr>
</tbody>
</table>

### Results

- **Low Complexity**
  - Architecture

- **Medium Complexity**
  - Effectiveness: 5
  - Medium Complexity

- **Complex Architecture**
  - Complex Architecture
  - 2.5

### Graphs

- **Graph 1**: Budget Ratio vs. Total Time
- **Graph 2**: Budget Ratio vs. % non scheduled ops
- **Graph 3**: Budget Ratio vs. Slack

### Tables

<table>
<thead>
<tr>
<th>Budget Ratio</th>
<th>Sum II/Sum MI</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0,9</td>
</tr>
<tr>
<td>0,95</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1,05</td>
</tr>
<tr>
<td>1,05</td>
<td>1,1</td>
</tr>
<tr>
<td>1,1</td>
<td>1,15</td>
</tr>
<tr>
<td>1,15</td>
<td>1,2</td>
</tr>
</tbody>
</table>

### Diagrams

- IMS
- IRIS
- Slack
II vs MII

Average (II/MII)

Architectures

Low

Medium

RESULTS
Register Pressure

RESULTS

<table>
<thead>
<tr>
<th>MaxLive/MinAvg</th>
<th>Low</th>
<th>Medium</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMS+ST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMS+ST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IRIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IRIS+ST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slack</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slack+ST</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Talk Outline

- Modulo Scheduling Background
- Selection Criteria
- Techniques Compared
- Study Environment
- Results
- Conclusions
We have performed a comparison of well known techniques

- IMS, SMS, Slack MS, IRIS and Stage MS

We have use a common

- Compiler platform
- Benchmarks
- Target architectures
- Measures

To perform a Quantitative Comparison

- Study of the BudgetRatio
- Measures of the techniques (Code quality, Effectiveness and cost)
Some interesting results:

- SMS best II and register pressure for lower/medium complex
- IMS slightly better II for complex architectures
- Slack MS, close to SMS in register pressure
- IMS and IRIS improved by Stage MS but still beyond SMS and Slack

Bottomline

- SMS is generally the best or close to the best

The tool can be used to assess any particular microarchitecture
A Comparative Study of Modulo Scheduling Techniques

Josep M. Codina, Josep Llosa and Antonio González

Dept. of Computer Architecture
Universitat Politècnica de Catalunya
Barcelona, SPAIN

E-mail: {jmcodina, josepll, antonio}@ac.upc.es