“Rate-Optimal” Resource-Constrained Software Pipelining
Objectives

• Establish good bounds
  - Help compiler writers
  - Help architects
• Study pragmatic issues when implemented as an option of compilers
• Study its payoffs
A Motivating Example

for i = 0 to n
do
  0: a[i] = X + d[i - 2];
  1: b[i] = a[i] * F + f[i - 2] + e[i - 2];
  2: c[i] = Y - b[i];
  3: d[i] = 2 * c[i];
  4: e[i] = X - b[i];
  5: f[i] = Y + b[i]
end
L: for ( i = 0; i < n; i ++) {
S0: a [i] = X + d [i - 2];
S1: b [i] = a [i] * F + f [i - 2] + e [i - 2];
S2: c [i] = Y - b [i];
S3: d [i] = 2 * c [i];
S4: e [i] = X - b [i];
S5: f [i] = Y + b [i]
}

Program Representation
Assume all statements have a delay = 1

Data Dependence Graph
Question

• How fast can L run (i.e. optimal computation rate) without resource constraints?
• How many FUs it needs minimally to achieve the optimal rate?
Schedule A:

\[ t(i, S_j) = 2i + t_{s_j} \]

where:

- \( t_{s_0} = 0 \)
- \( t_{s_1} = 1 \)
- \( t_{s_2} = t_{s_4} = t_{s_5} = 2 \)
- \( t_{s_3} = 3 \)

<table>
<thead>
<tr>
<th>iter #1</th>
<th>#2</th>
<th>#3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( S_0 )</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>( S_1 )</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( S_2, S_4, S_5 )</td>
<td>( S_0 )</td>
</tr>
<tr>
<td>3</td>
<td>( S_3 )</td>
<td>( S_1 )</td>
</tr>
<tr>
<td>4</td>
<td>( S_2, S_4, S_5 )</td>
<td>( S_0 )</td>
</tr>
<tr>
<td>5</td>
<td>( S_3 )</td>
<td>( S_1 )</td>
</tr>
<tr>
<td>6</td>
<td>( S_2, S_4, S_5 )</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>( S_3 )</td>
<td></td>
</tr>
</tbody>
</table>
Can we do better?
Schedule B:

\[ t(i, S_j) = 2i + t_{sj} \]

where:

- \( t_{s0} = 0 \)
- \( t_{s1} = 1 \)
- \( t_{s2} = t_{s4} = 2 \)
- \( t_{s3} = t_{s5} = 3 \)

<table>
<thead>
<tr>
<th>iter #1</th>
<th>#2</th>
<th>#3</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( S_0 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>( S_1 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( S_2, S_4 )</td>
<td>( S_0 )</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>( S_3, S_5 )</td>
<td>( S_1 )</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>( S_2, S_4 )</td>
<td>( S_0 )</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>( S_3, S_5 )</td>
<td>( S_1 )</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>( S_2, S_4 )</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>( S_3, S_5 )</td>
<td></td>
</tr>
</tbody>
</table>

**Schedule B : # of FUs = 3**
Problem Statements

Assumptions: homogeneous function units

Problem 0: Given a loop L, determine a “rate-optimal” schedule for L which uses minimum # of FUs.

Problem I (Fixed Rate Software Pipelining with minimum Resource - FIRST):
Given a loop L and a fixed initiation rate determine a schedule for L which uses minimum # of FUs.

Problem II (Resource Constrained Software Pipelining - REST)
Given a loop L and the # of Fus, determine an optimal schedule which can run under the given resource constraints.
Problem Formulation of FIRST

How to formulate resource constraints into a linear form?
R = Max (width)

The SWP kernel:  
--- The “frustum”

Min R

Subject to:
- all dependence constraints
- R is the maximum width
- for a fixed rate (or period)
Min (max $\sum_{r} a_{ri}$)

Frustum A

Now $R = \max \sum_{i=0}^{n-1} a_{ri}$

$\forall \ r \in [0, II - 1]$
FIRST: A Linear Programming Formulation Example

\[
\begin{bmatrix}
0 & 1 & 2 & 3 & 4 & 5 \\
00 & 01 & 02 & 03 & 04 & 05 \\
10 & 11 & 12 & 13 & 14 & 15 \\
\end{bmatrix}
\]

So for node I:

\[
t_i = k_i \|I\| + (a_{oi}, a_{1i}) \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

Where \(a_{ri} = 1\) means node i starts at step r

= 0 otherwise
FIRST:
A Linear Programming Formulation

Minimize $R$

Subject to

$$R - \sum_{i=0}^{N-1} a_{0i} \geq 0, R - \sum_{i=0}^{N-1} a_{1i} \geq 0, ..., R - \sum_{i=0}^{N-1} a_{i(i-1)i} \geq 0$$

$R \cdot K + \vec{A} \times [0,1,...II-1] = \vec{T}$

$$\sum_{i=0}^{II-1} a_{ir} = 1 \quad \forall i \in [0,N-1]$$

$t_i - t_i \geq 1 - II \cdot m_{ij} \quad \forall (i,j) \in E$

$t_i \geq 0, k_i \geq 0$ and $a_{ir} \geq 0$ are integers \quad $\forall i \in [0,N-1], \forall r \in [0,II-1]$
Example Revisited

Min R

Subjected to:

\[ R - (a_{00} + a_{01} + a_{02} + a_{03} + a_{04} + a_{05}) \geq 0 \]
\[ R - (a_{10} + a_{11} + a_{12} + a_{13} + a_{14} + a_{15}) \geq 0 \]
\[ 2 \cdot k_0 + 0 \cdot a_{00} + 1 \cdot a_{10} = t_0 \]
\[ 2 \cdot k_1 + 0 \cdot a_{01} + 1 \cdot a_{11} = t_1 \]
\[ \vdots \]
\[ 2 \cdot k_5 + 0 \cdot a_{05} + 1 \cdot a_{15} = t_5 \]
\[ a_{00} + a_{10} = 1 \]
\[ a_{01} + a_{11} = 1 \]
\[ \vdots \]
\[ a_{05} + a_{15} = 1 \]
\[ t_1 - t_0 \geq 1 \]
\[ \vdots \]
\[ t_1 - t_4 \geq -3 \]
\[ t_1 - t_5 \geq -3 \]
\[ t_1 \geq 0, k \geq 0, a_{ij} \geq 0 \]
Solution

Schedule C

\[ t(i, s) = 2i + t_s \]

where

\[ t_0 = 0 \]
\[ t_1 = 1 \]
\[ t_2 = 2 \]
\[ t_3 = 3 \]
\[ t_4 = 3 \]
\[ t_5 = 2 \]

so, # of FUs = 3!
Linear Programming

• A Linear Program is a problem that can be expressed in the following form:
  • minimize $cx$
  • subject to
    • $Ax = b$
    • $x \geq 0$

where $x$: vector of variables to be solved
  $A$: matrix of known coefficients
  $c, b$: vectors of known coefficients

$cx$: it is called the objective function
$Ax=b$ is called the constraints
Linear Programming

• “programming” actually means “planning” here.
• Importance of LP
  - many applications
  - the existence of good general-purpose techniques for finding optimal solutions
Integer Linear Programming (ILP)

- Integer programming have proved valuable for modeling many and diverse types of problems in:
  - planning,
  - routing,
  - scheduling,
  - assignment, and
  - design.
- Industry applications:
  - transportation, energy, telecommunications, and manufacturing
Integer Linear Programming

- Classification:
  - mixed integer (part of variables)
  - pure integer (all variables)
  - zero-one (only takes 0 or 1)
- Much harder to solve
- Many existing tools and product
  - solve the LP/ILP problem and get optimal solution
  - help to model the problem, and then solve the problem
Quiz

How to handle cases

where \( d_i \geq 1 \) for node \( i \)

?
The “Trick”

At time step $r$, the # of FUs required

$$\alpha_{ri} = \sum_{l=0}^{d_i-1} \alpha_{((r - 1) \mod II)i}$$

Contributed by node $i$

{Started at steps in $[0, (t + d_i - 1) \mod II]$}

Note: $\alpha_i = 1$ if actor $i$ requires a FU at time $r$ or 0 otherwise.
So objective function becomes:

$$\min \left( \max \sum_{i=0}^{N-1} \alpha_{ri} \right)$$

$\forall r \in [0, II - 1]$
How to extend the FIRST formulation to heterogeneous function units?
Heterogeneous Function Units

Solution Hints: “weighted Sum”!

For FU of type k:

\[ M_k = \max \sum_{i=0}^{N-1} \alpha_{ri} \]

\[ \forall r \in [0, II - 1] \]

and so, the objective should be

\[ \min \sum_{k=0}^{h-1} C_k \cdot M_k \]

Where \( h \) is the total # of FU types

\[ M_k - \sum_{i=0}^{n-1} \alpha_{ri} \geq 0 \]

\[ \forall k \in [0, h - 1], \forall r \in [0, II - 1] \]

\[ \text{FU}(i) = k \]
Solution of the REST

Resource - constraint rate-optimal software pipelining problem
Hints

Based on scheme for
“FIRST”
and play
“trial-and improve”
Solution Space for REST

\[ MII = \max \{ \text{RecMII}, \text{ResMII} \} \]
Note: *The bounds of initiation interval*

1. Loop-carried dependence constraints:

\[
\text{RecMII} = \max_{\forall \text{cycles } C} \frac{d(C)}{m(C)}
\]

2. Resource constraints

\[
\text{ResMII} = \frac{\# \text{ of nodes}}{\# \text{ of FUs}}
\]

As a result:

\[
\text{MII} = \max\{\text{RecMII, ResMII}\}
\]
Other Work

- Extend the technique to “more benchmarks” and establish “bounds”
- Extend the framework to include register constraints
- Extend the model for pipelined architectures (with structural “hazards”)
- Extend the model to consider superscalar and multithreaded with dynamic scheduling support.