Scheduling instructions on a hierarchical architecture

Florent Blachot, Guillaume Huard, Johnatan Pecero, Erik Saule, Denis Trystram

STMicroelectronics & LIG

GOTHA - 4 Avril 2008
Outline of the talk

1. The ST200 Processor
2. The Scheduling Problem
3. Analysis
4. Experimental Validation
5. Conclusion
Outline

1. The ST200 Processor
2. The Scheduling Problem
3. Analysis
4. Experimental Validation
5. Conclusion
The **ST200** processor

The ST200 processor produced by STmicroelectronics, used in “set top box” such as DVD player. It has a not so common architecture.

Interested in scheduling instruction on this processor.
The **ST200** processor

![Diagram of the ST200 processor](image)

**Figure**: Current version of ST200

- The result of an operation on an ALU is immediately available on others.
The **st200** processor

![Diagram of the st200 processor]

**Figure:** Current version of *st200*

- The result of an operation on an ALU is immediately available on others.
- The cost in silicon increases in the square of the number of ALUs.
The **ST200** processor with Incomplete Bypass

The result of an operation on one ALU is immediately available on ALUs of the same cluster, but 2 time clocks later on other clusters.

*Figure: Future revision of the ST200 processor using an Incomplete Bypass*
The **ST200** processor with Incomplete Bypass

**Figure:** Future revision of the **ST200** processor using an Incomplete Bypass

- The result of an operation on one ALU is immediately available on ALUs of the same cluster, but 2 time clocks later on other clusters.
- The cost in silicon increases in the square of the number of ALU in a cluster and linearly in the number of clusters.
A compiler problem

How to compile a code for these architectures?
Mainly 2 problems:

- register allocation
- **instruction scheduling**
A compiler problem

How to compile a code for these architectures?
Mainly 2 problems:
- register allocation
- instruction scheduling

Remark
On complete bypass system, the problem is $P_m \mid prec, p_j = 1 \mid C_{\text{max}}$. 
On incomplete bypass?
Outline

1. The ST200 Processor
2. The Scheduling Problem
3. Analysis
4. Experimental Validation
5. Conclusion
The model

- DAG $G = (T, E)$ where $T$ is a set of $n$ unitary tasks.
- Processors are organized in $M$ clusters of $m$ processors. The $l$-th cluster is $H_l$.
- Solution: $\pi : T \rightarrow P$ and $\sigma : T \rightarrow \mathbb{N}^+$
- Between $H_i$ and $H_j$ ($i \neq j$), $\rho$ time units of delay
- Min $C_{\text{max}}$

The problem is denoted by $P_M(P_m)|\text{prec}, p_j = 1, c = (\rho, 0)|C_{\text{max}}$ [BGK03]
The model

- DAG $G = (T, E)$ where $T$ is a set of $n$ unitary tasks.
- Processors are organized in $M$ clusters of $m$ processors. The $l$-th cluster is $H_l$.
- Solution: $\pi : T \to P$ and $\sigma : T \to \mathbb{N}^+$
- Between $H_i$ and $H_j$ ($i \neq j$), $\rho$ time units of delay
- Min $C_{\max}$

The problem is denoted by $P_M(P_m)|\text{prec}, p_j = 1, c = (\rho, 0)|C_{\max}$ [BGK03]

Remark

The $st200$ case is $m = 3$, $M = 2$, $\rho = 2$. 
An Example
An Example

12 times

$H_1$

$H_2$
Related works

Complexity

\( P_M(P_m)|\text{prec}, p_j = 1, c = (\rho, 0)|C_{\text{max}} \) is NP-hard.

The complexity of the \( \text{ST200} \) case is not that obvious. It is at least as hard as \( P3 | \text{prec}, p_j = 1 | C_{\text{max}} \) which is known to be an open problem.
Related works

Complexity

$P_M(P_m)|\text{prec}, p_j = 1, c = (\rho, 0)|C_{\text{max}}$ is NP-hard.

The complexity of the ST200 case is not that obvious. It is at least as hard as $P3 | \text{prec}, p_j = 1 | C_{\text{max}}$ which is known to be an open problem.

Approximability

$P_2(P) | \text{bipartite}, p_j = 1, c = (1, 0) | C_{\text{max}} = 3$ is NP-complete $\Rightarrow$ no approximation algorithm with a performance ratio better than $4/3$ [ABG02].
Related works

Complexity

\[ P_M(P_m) | \text{prec}, p_j = 1, c = (\rho, 0) | C_{\text{max}} \text{ is NP-hard.} \]

The complexity of the ST200 case is not that obvious. It is at least as hard as \( P3 | \text{prec}, p_j = 1 | C_{\text{max}} \) which is known to be an open problem.

Approximability

\[ P_2(P) | \text{bipartite}, p_j = 1, c = (1, 0) | C_{\text{max}} = 3 \text{ is NP-complete } \Rightarrow \text{ no approximation algorithm with a performance ratio better than } 4/3 \]

[ABG02].

List Scheduling with communication has a performance ratio of \( 2 - \frac{1}{mM} + \rho \)
Outline

1. The ST200 Processor
2. The Scheduling Problem
3. Analysis
4. Experimental Validation
5. Conclusion
An idle at \( t \) is an IdleCP if all tasks scheduled after the idle time depend on a task scheduled at \( t \).
Reinventing the idle

Definition

An idle at $t$ is an IdleCP if all tasks scheduled after the idle time depend on a task scheduled at $t$.

Definition

An idle at $t$ is a communicationnal idle if all tasks scheduled after the idle time depend on a task scheduled before $t$ and could not be scheduled on the idle.
Reinventing the idle

**Definition**
An idle at $t$ is an IdleCP if all tasks scheduled after the idle time depend on a task scheduled at $t$.

**Definition**
An idle at $t$ is a communicationnal idle if all tasks scheduled after the idle time depend on a task scheduled before $t$ and could not be scheduled on the idle.

**Definition**
An idle at $t$ is an lateness idle if there exists a task released at $t$ scheduled after $t$. 
A nice property

Proposition

A schedule without communicational idle and lateness idle on at least one cluster is $M + 1 - \frac{1}{m}$ optimal.

Proof.

sketch:
Two lower bounds. $\frac{n}{Mm}$ (work) and $t_\infty$ (critical path).
Such a schedule have $C_{\max} \leq \frac{n}{m} + t_\infty$.
Thus $C_{\max} \leq MC_{\max}^* + C_{\max}^*$. 
Use List Scheduling on one cluster only.

Corollary

*GSingle generates schedules without communicational and lateness idle.*

*Thus it is* $M + 1 - \frac{1}{m}$ *optimal.*

*In the* $ST200$ *case* ($M = 2$ *and* $m = 3$), *GSingle is* $\frac{8}{3}$ *optimal.* *Better than LS which is* $\frac{23}{6}$.
GSingle

**Algo**

Use List Scheduling on one cluster only.

**Corollary**

*GSingle generates schedules without communicational and lateness idle.*

*Thus it is $M + 1 - \frac{1}{m}$ optimal.*

*In the ST200 case ($M = 2$ and $m = 3$), GSingle is $\frac{8}{3}$ optimal. (better than LS which is $\frac{23}{6}$)*

**Remark**

It uses only $\frac{1}{M}$ of the computational power.
**Favorite Cluster**

### Principle
Let $H_1$ be the master cluster. Use List scheduling on $H_1$. On other clusters $H_i$. Schedule a task on $H_i$ only if it will be available on $H_1$ the next time. If $H_1$ has a communicational idle, export the last task from $H_i$ to $H_1$.

### Bound
Favorite Cluster generates schedules without communicational and lateness idle. It is a $M + 1 - \frac{1}{m}$-approximation algorithm and the bound is tight.
Tightness

(a) DAG

$mM$ times
Tightness

\[ k \times m \times (a) \text{DAG} \]

(a) DAG
Tightness

\[ \rho + 1 + \left\lceil \frac{k}{mM-1} \right\rceil \times k \times mM \times (a)DAG \]
Tightness

\[ \rho + 1 + \left\lceil \frac{k}{mM-1} \right\rceil \text{ times} \]

\[ k \text{ times} \]

\[ \left\lceil \frac{k}{5} \right\rceil \text{ times} \]

\[ mM \text{ times} \]

(a) DAG

(b) Optimal schedule

(c) Favorite Cluster schedule
Another Approximation Ratio

Theorem

Favorite Cluster is a \(2 + 2\rho - \frac{2\rho}{M} - \frac{1}{Mm}\)-approximation algorithm and the bound is tight.

Proof idea
Outline

1. The ST200 Processor
2. The Scheduling Problem
3. Analysis
4. Experimental Validation
5. Conclusion
Goal: compare GSingle, Favorite Cluster and List Scheduling. From [KA98], benchmarks for $P \mid prec \mid C_{\text{max}}$. Contains randomly generated graphs and graphs extracted from a parallel compiler. On Random graphs: Layered graphs.
The relative behavior of the tree methods (LU Graph)

![Graph showing the relative behavior of tree methods for LU factorizations.](image-url)

Erik Saule (LIG)  
Scheduling instructions on a hierarchical architecture  
GOTHA - 4 Avril 2008  
21 / 26
Structured Graphs (Cholesky)

The relative behavior of the tree methods (Cholesky Graph)

![Graph showing the relative behavior of tree methods](image)

- **GSingle/GSingle**
- **FavoriteCluster/GSingle**
- **ListComm/GSingle**

Figure: Normalized makespans for the three heuristics on Cholesky factorizations.
Layered Graphs

\[ Z = C_{\text{FavoriteCluster}} - C_{\text{max}} \]

<table>
<thead>
<tr>
<th>Size</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
<th>110</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z &lt; 0</td>
<td>107</td>
<td>138</td>
<td>198</td>
<td>210</td>
<td>214</td>
<td>219</td>
<td>243</td>
<td>154</td>
<td>239</td>
</tr>
<tr>
<td>Z &gt; 0</td>
<td>42</td>
<td>52</td>
<td>69</td>
<td>94</td>
<td>103</td>
<td>114</td>
<td>106</td>
<td>89</td>
<td>116</td>
</tr>
<tr>
<td>Z = 0</td>
<td>351</td>
<td>310</td>
<td>233</td>
<td>196</td>
<td>183</td>
<td>167</td>
<td>151</td>
<td>102</td>
<td>145</td>
</tr>
</tbody>
</table>
Layered Graphs

\[ Z = C_{\text{FavoriteCluster}} - C_{\text{max}} \]

<table>
<thead>
<tr>
<th>Size</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
<th>110</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Z &lt; 0 )</td>
<td>107</td>
<td>138</td>
<td>198</td>
<td>210</td>
<td>214</td>
<td>219</td>
<td>243</td>
<td>154</td>
<td>239</td>
</tr>
<tr>
<td>( Z &gt; 0 )</td>
<td>42</td>
<td>52</td>
<td>69</td>
<td>94</td>
<td>103</td>
<td>114</td>
<td>106</td>
<td>89</td>
<td>116</td>
</tr>
<tr>
<td>( Z = 0 )</td>
<td>351</td>
<td>310</td>
<td>233</td>
<td>196</td>
<td>183</td>
<td>167</td>
<td>151</td>
<td>102</td>
<td>145</td>
</tr>
<tr>
<td>( E[Z] )</td>
<td>-0.232</td>
<td>-0.336</td>
<td>-0.602</td>
<td>-0.654</td>
<td>-0.794</td>
<td>-0.784</td>
<td>-1.036</td>
<td>-0.8841</td>
<td>-0.974</td>
</tr>
<tr>
<td>( \sigma[Z] )</td>
<td>0.9433</td>
<td>1.1343</td>
<td>1.6875</td>
<td>1.9187</td>
<td>2.2513</td>
<td>2.4314</td>
<td>2.9053</td>
<td>2.6474</td>
<td>2.7896</td>
</tr>
<tr>
<td>( \min(Z) )</td>
<td>-5</td>
<td>-6</td>
<td>-10</td>
<td>-9</td>
<td>-11</td>
<td>-11</td>
<td>-16</td>
<td>-11</td>
<td>-15</td>
</tr>
<tr>
<td>( \max(Z) )</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>( E[Z] \leq )</td>
<td>-0.1251</td>
<td>-0.2180</td>
<td>-0.4265</td>
<td>-0.4544</td>
<td>-0.5598</td>
<td>-0.5311</td>
<td>-0.7338</td>
<td>-0.6087</td>
<td>-0.6838</td>
</tr>
</tbody>
</table>
Outline

1. The ST200 Processor
2. The Scheduling Problem
3. Analysis
4. Experimental Validation
5. Conclusion
Conclusion

- Present a scheduling problem from the compiler community
- Define different Idle time
- Generalize List Scheduling for $P_M(P_m)|\text{prec}, p_j = 1, c = (\rho, 0)|C_{\max}$
- Propose a heuristic with good behavior in practice
Derive a better approximation algorithm (that grows with $M$)
- FavoriteCluster does not use the UET assumption.
- Task’s in-degree is less than 2 (or equal).
Perspective

- Derive a better approximation algorithm (that grows with $M$)
  - FavoriteCluster does not use the UET assumption.
  - Task’s in-degree is less than 2 (or equal).
- ... or find some inapproximability bounds.
Derive a better approximation algorithm (that grows with $M$)
- FavoriteCluster does not use the UET assumption.
- Task’s in-degree is less than 2 (or equal).

... or find some inapproximability bounds.

FavoriteCluster applies to cluster scheduling. Investigate it.
E Angel, E Bampis, and R Giroudeau.
Non-approximability results for the hierarchical communication problem with a bounded number of clusters.

E. Bampis, R. Giroudeau, and J-C. König.
An approximation algorithm for the precedence constrained scheduling problem with hierarchical communications.

Y-K. Kwok and I. Ahmad.
Benchmarking the task graph scheduling algorithms.