Scheduling instructions on a hierarchical architecture

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

STMicroelectronics & LIG
{Florent.Blachot, Guillaume.Huard, Johnatan.Pecero, Erik.Saule, Denis.Trystram}@imag.fr

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 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

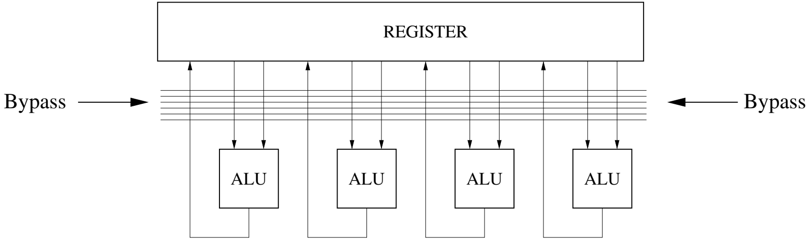


Figure: Current version of ST200

- The result of an operation on an ALU is immediately available on others

The ST200 processor

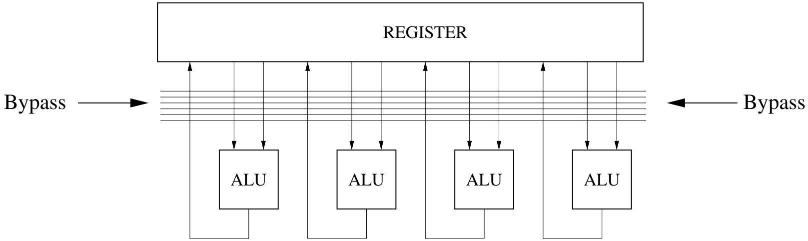


Figure: Current version of ST200

- 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 ALU

The ST200 processor with Incomplete Bypass

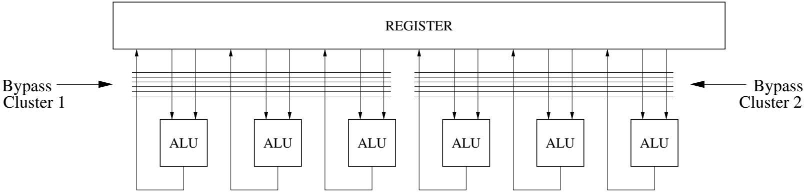


Figure: Future revision of the ST200 processor using an Incomplete Bypass

- 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 ST200 processor with Incomplete Bypass

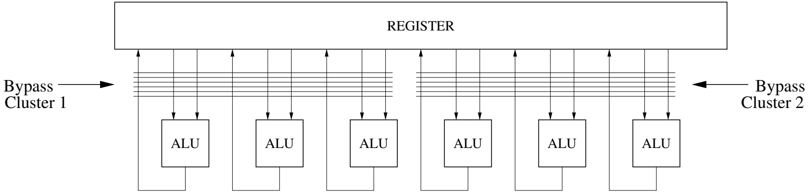


Figure: Future revision of the ST200 processor using an Incomplete Bypass

- 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 $P_m \mid prec, p_j = 1 \mid C_{\max}$.
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 ($i \neq j$), ρ time units of delay
- Min $C_{\max}$

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

The model

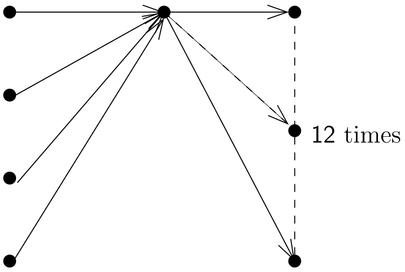
- DAG $G = (T, E)$ where T is a set of n unitary tasks.
- Processors are organized in M clusters of m processors. The l-th cluster is H_l.
- Solution : $\pi : T \rightarrow P$ and $\sigma : T \rightarrow \mathbb{N}^+$
- Between H_i and H_j ($i \neq j$), ρ time units of delay
- Min $C_{\max}$

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

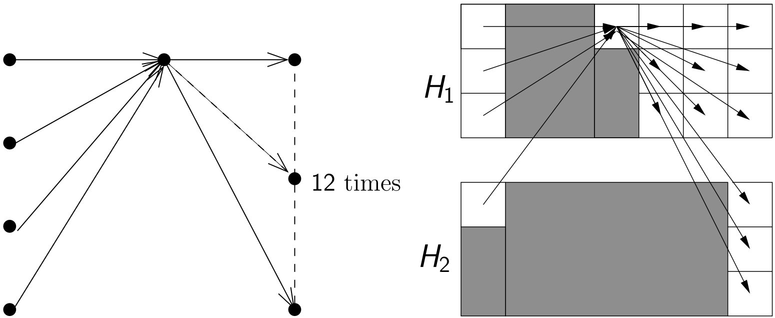
Remark

The $\textsc{st200}$ case is $m = 3$, $M = 2$, $\rho = 2$.

An Example



An Example



Related works

Complexity

$P_M(P_m)|prec, p_j = 1, c = (\rho, 0)|C_{\max}$ is NP-hard.
The complexity of the $\mathrm{ST200}$ case is not that obvious. It is at least as hard as $P3 \mid prec, p_j = 1 \mid C_{\max}$ which is known to be an open problem.

Related works

Complexity

$P_M(P_m)|prec, p_j = 1, c = (\rho, 0)|C_{\max}$ is NP-hard.
The complexity of the ST200 case is not that obvious. It is at least as hard as $P3 \mid prec, p_j = 1 \mid C_{\max}$ which is known to be an open problem.

Approximability

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

Related works

Complexity

$P_M(P_m)|prec, p_j = 1, c = (\rho, 0)|C_{\max}$ is NP-hard.
The complexity of the ST200 case is not that obvious. It is at least as hard as $P3 \mid prec, p_j = 1 \mid C_{\max}$ which is known to be an open problem.

Approximability

$P_2(P) \mid bipartite, p_j = 1, c = (1, 0) \mid C_{\max} = 3$ is NP-complete $\Rightarrow$ 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

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.

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_∞ (critical path).
Such a schedule have $C_{\max} \leq \frac{n}{m} + t_\infty$.
Thus $C_{\max} \leq MC^*_{\max} + C^*_{\max}$. □

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 $\frac{23}{6}$)*

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

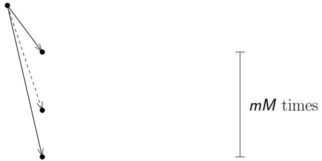
Principle

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

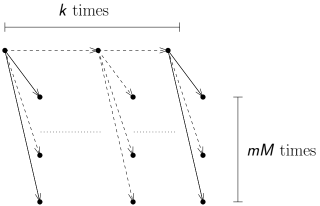
Favorite Cluster generates schedules without communicational and lateness idle. It is a $M + 1 - \frac{1}{m}$-approximation algorithm and the bound is tight.

Tightness



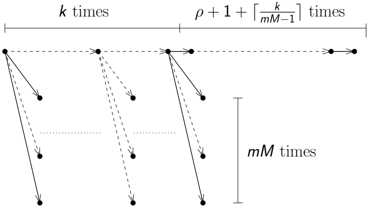
(a)DAG

Tightness



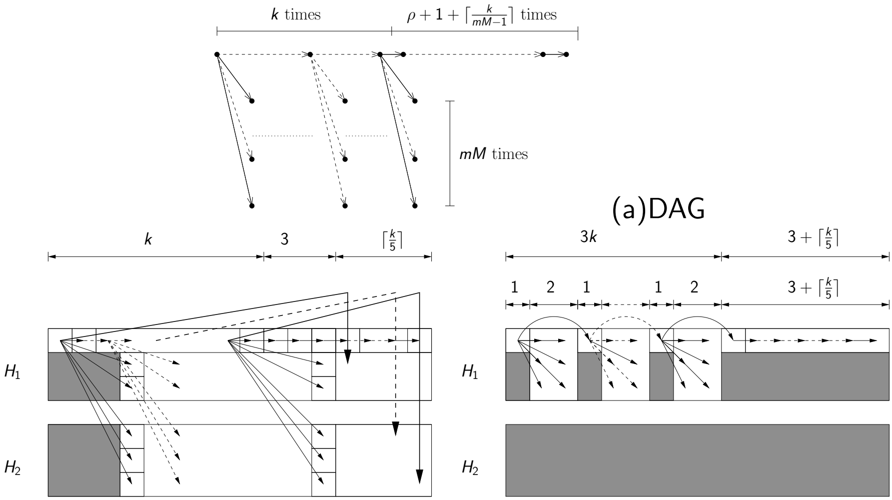
(a)DAG

Tightness



(a)DAG

Tightness



(a)DAG

(b)Optimal schedule

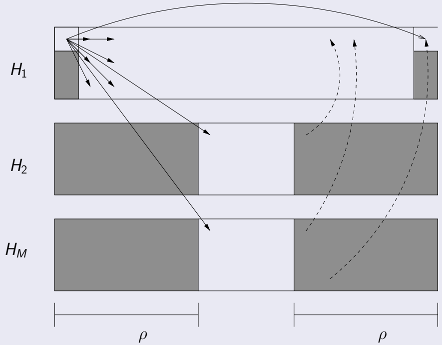
(c)Favorite Cluster schedule

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



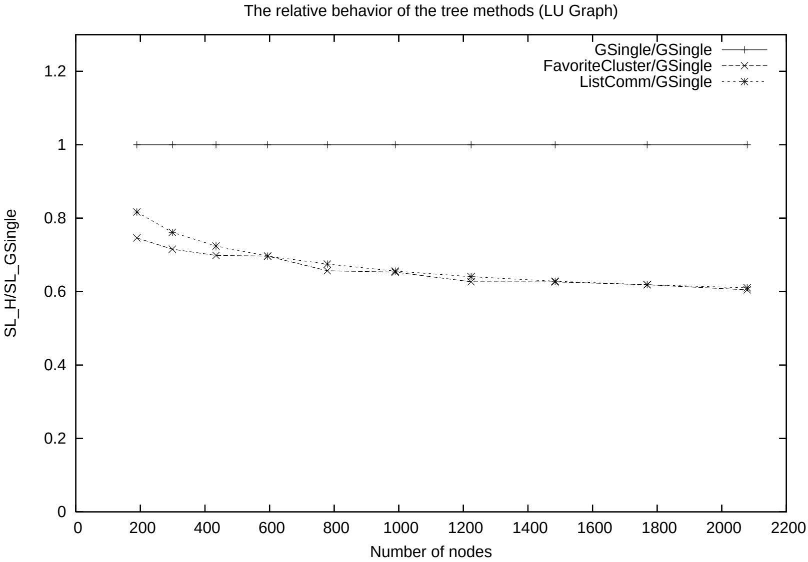
Outline

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

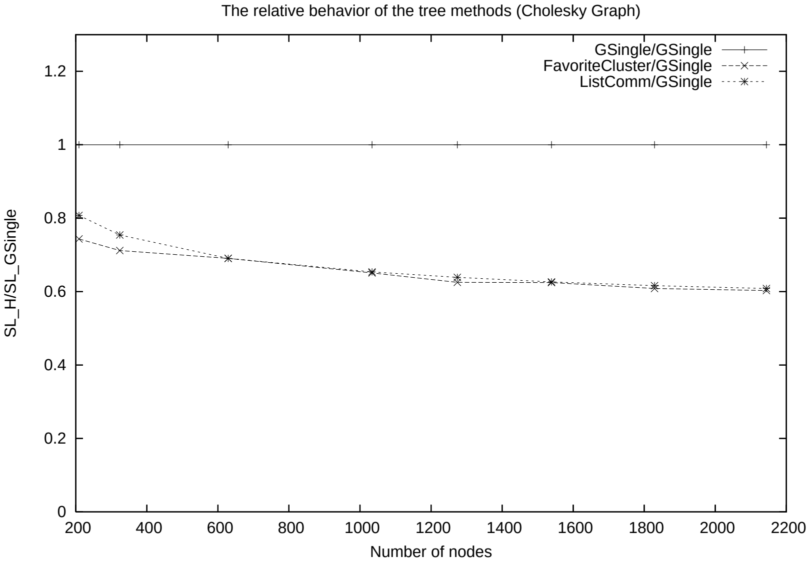
Experimental validation

Goal: compare GSingle, Favorite Cluster and List Scheduling.
From [KA98], benchmarks for $P \mid prec \mid C_{\max}$. Contains randomly generated graphs and **graphs extracted from a parallel compiler**.
On Random graphs: Layered graphs.

Structured Graphs(LU)



Structured Graphs(Cholesky)



Layered Graphs

$$Z = C_{\max}^{FavoriteCluster} - C_{\max}^{LS}$$

Size	30	40	50	60	70	80	90	100	110
$Z < 0$	107	138	198	210	214	219	243	154	239
$Z > 0$	42	52	69	94	103	114	106	89	116
$Z = 0$	351	310	233	196	183	167	151	102	145

Layered Graphs

$$Z = C_{\max}^{FavoriteCluster} - C_{\max}^{LS}$$

Size	30	40	50	60	70	80	90	100	110
$Z < 0$	107	138	198	210	214	219	243	154	239
$Z > 0$	42	52	69	94	103	114	106	89	116
$Z = 0$	351	310	233	196	183	167	151	102	145
$E[Z]$	-0,232	-0,336	-0,602	-0,654	-0,794	-0,784	-1,036	-0,8841	-0,974
$\sigma[Z]$	0,9433	1,1343	1,6875	1,9187	2,2513	2,4314	2,9053	2,6474	2,7896
$min(Z)$	-5	-6	-10	-9	-11	-11	-16	-11	-15
$max(Z)$	3	2	3	5	4	6	5	4	7
$E[Z] \leq$	-0.1251	-0.2180	-0.4265	-0.4544	-0.5598	-0.5311	-0.7338	-0.6087	-0.6838
$E[C_{\max}^{FavoriteCluster}]$	10,554	13,422	15,712	17,61	19,304	21,706	23,108	25,3188	26,822

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 $P_M(P_m)|prec, p_j = 1, c = (\rho, 0)|C_{\max}$
- Propose a heuristic with good behavior in practice

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).

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.

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.
- 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.
In B. Monien and R. Feldmann, editors, *EuroPar' 02*, pages 217–224, 2002.

E. Bampis, R. Giroudeau, and J-C. König.
An approximation algorithm for the precedence constrained scheduling problem with hierarchical communications.
Theor. Comput. Sci., 290(3):1883–1895, 2003.

Y-K. Kwok and I. Ahmad.
Benchmarking the task graph scheduling algorithms.
In *IPPS/SPDP*, pages 531–537, 1998.