Influence of Compiler Optimizations on System Power

IEEE TRANSACTIONS ON VLSI SYSTEMS
Mahmut Kandemir, N. Vijaykrishnan, Mary Jane Irwin, and Wu Ye
Microsystems Design Lab., Pennsylvania State Univ., University Park, PA, USA
Page (s) 801-804
DEC. 2001
Abstract

- Optimizing for energy constraints is of critical importance due to the proliferation of battery-operated embedded devices.
- Thus, it is important to explore both hardware and software solutions for optimizing energy.
- The focus of high-level compiler optimizations has traditionally been on improving performance.
- In this paper, we present an experimental evaluation of several state-of-the-art high-level compiler optimizations on energy consumption, considering both the processor core (datapath) and memory system.
- This is in contrast to many of the previous works that have considered them in isolation.
Outline

- What is the Problem
- Related Work
- Compiler Optimizations
- Experiments and Results
- Result Analysis
- Conclusions
What is the problem

- Embedded systems face stringent energy and area constraints. Software implementation relative to hardware increase requires powerful compiler technology.
- From a performance point of view, and from an energy point of view...

The problem is:
- How about high-level compiler optimizations on energy consumption?
- To gain some insight into tradeoffs between energy consumption and performance.
Related Work

- Utilize previous work tools:
  - **SimplePower**: energy simulator
    - Transition-sensitive
    - Execution-driven
    - Cycle-accurate
    - RTL power estimation tool
    - Five-stage pipelined datapath architecture that takes C code as input
  - **Source-to-source translator**: source optimizer
    - Perform various code transformation investigated
  - **A number of benchmark codes**
Compiler Optimizations summary

- With the advent of parallel architectures and systems with deep memory hierarchies, optimizations gained locality and parallelism optimizations.

- Loop-nest optimizations hold an important place in high-level optimizations. (especially in multimedia operation)
  - Linear loop transformations
  - Loop tiling
  - Loop unrolling
  - Loop fusion
  - Loop fission
Linear loop transformations

**Before:**

```plaintext
DO i=1, n
  DO j=1, n
    W(i,j) = X(i,j) + Y(j) + Z(j)
  END DO
END DO
END DO
```

No loop invariants before the optimization

**After:**

```plaintext
DO j=1, n
  DO I=1, n
    W(i,j) = X(i,j) + Y(j) + Z(j)
  END DO
END DO
```

Better utilization of cache

reverse

loop-variant expression
Loop tiling (blocking)

\[ A(i,1:n) = B(1:n,i) \]

- Accessing data in the A matrix assumes row major order
- Accessing data in the B matrix assumes column major order
- If A and B are in the same loop, locality can not be achieved in both.

\[ A(ii:ii+T_i-1,jj:jj+T_j-1) = B(jj:jj+T_j-1,ii:ii+T_i-1) \]

- Now, Both A and B have similar access patterns
- Locality in both A and B is achieved
Loop tiling cont.

\[ A(ii:ii+B_{i}-1,jj:jj+B_{j}-1) \]

\[
\begin{align*}
\text{DO } i=1,n \\
\text{DO } j=1,n \\
X(i,j) &= Y(j,i) \quad \text{----->} \quad X(i,j) = Y(j,i) \\
\text{END DO} \\
\text{END DO} \\
\end{align*}
\]

\[
\begin{align*}
\text{DO } jj=1,n,B \\
\text{DO } i=ii, \min(ii+B_{i}-1,n) \\
\text{DO } j[jj, \min(jj+B_{j}-1,n) \\
X(i,j) &= Y(j,i) \quad \text{----->} \quad X(i,j) = Y(j,i) \\
\text{END DO} \\
\text{END DO} \\
\text{END DO} \\
\end{align*}
\]

- The original loop bounds are \(N_1\) and \(N_2\)
- The indices \(i\) and \(j\) need to step by \(B_i\) and \(B_j\) respectively
- The indices \(ii\) and \(jj\) are used to iterate within the tile
**Loop unrolling**

Before unrolling:

DO i=1, n, 1
W(i)=X(i)+Y(i)+Z(i)
END DO

After 2x unrolling:

DO i=1, n-1, 2
W(i)=X(i)+Y(i)+Z(i)
W(i+1)=X(i+1)+Y(i+1)+Z(i+1)
END DO

- **Reduce trip count of loop**
- **Reduce memory access**
- **Promote register reuse**
Loop fusion

DO i = 1, n
  X(i) = Y(i) + Z(i)
END DO

DO i = 1, n
  Y(i) = X(i) + Z(i)
END DO

COST = 4n loads, 2n stores,

Cost = 2n loads, 2n stores

Improve data reuse

Loop eliminate
Loop fission

Suppose only two floating-point registers were available for allocation:

```
DO i = 1, n
  R = R + X(i)
  S = S + Y(i)
  T = T + Z(i)
END DO
```

# f.p. regs needed = 4

```
DO i = 1, n
  R = R + X(i)
END DO
DO i=1, n
  S= S + Y(i)
END DO
DO i = 1, n
  T = T + Z(i)
END DO
```

# f.p. regs needed = 2
Energy experimental results -1

- l : linear loop transformation
- u : loop unrolling
- t : tiling

Original code
Set associativity
Combine
Cache size

Table II
Energy Consumption in matmul

<table>
<thead>
<tr>
<th></th>
<th>Core Energy (J)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-way</td>
</tr>
<tr>
<td>l</td>
<td></td>
</tr>
<tr>
<td>orig</td>
<td>0.0344</td>
</tr>
<tr>
<td>u</td>
<td>0.0296</td>
</tr>
<tr>
<td>t</td>
<td>0.0837</td>
</tr>
<tr>
<td>lu</td>
<td>0.0423</td>
</tr>
<tr>
<td>lt</td>
<td>0.0617</td>
</tr>
<tr>
<td>tu</td>
<td>0.0457</td>
</tr>
<tr>
<td>tlu</td>
<td>0.0529</td>
</tr>
</tbody>
</table>

Presenter: ycliu
Energy experimental results - 2

Core is Very low to memory

<table>
<thead>
<tr>
<th></th>
<th>Core Energy (J)</th>
<th>Memory Energy (J)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-way</td>
<td>2-way</td>
</tr>
<tr>
<td>orig</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1K</td>
<td>0.1604</td>
<td>0.0915</td>
</tr>
<tr>
<td>2K</td>
<td>0.1159</td>
<td>0.0789</td>
</tr>
<tr>
<td>4K</td>
<td>0.1000</td>
<td>0.0763</td>
</tr>
<tr>
<td>8K</td>
<td>0.0730</td>
<td>0.0681</td>
</tr>
<tr>
<td>loop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1K</td>
<td>0.1418</td>
<td>0.0630</td>
</tr>
<tr>
<td>2K</td>
<td>0.0844</td>
<td>0.0493</td>
</tr>
<tr>
<td>4K</td>
<td>0.0609</td>
<td>0.0441</td>
</tr>
<tr>
<td>8K</td>
<td>0.0378</td>
<td>0.0283</td>
</tr>
<tr>
<td>tile</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1K</td>
<td>0.1404</td>
<td>0.0731</td>
</tr>
<tr>
<td>2K</td>
<td>0.0942</td>
<td>0.0646</td>
</tr>
<tr>
<td>4K</td>
<td>0.0550</td>
<td>0.0426</td>
</tr>
<tr>
<td>8K</td>
<td>0.0345</td>
<td>0.0228</td>
</tr>
</tbody>
</table>

- loop : linear loop transformation
- tile : tiling
### TABLE IV
**ENERGY CONSUMPTION IN nasa7/btrix**

<table>
<thead>
<tr>
<th>Core Energy (J)</th>
<th>Memory Energy (J)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-way</td>
</tr>
<tr>
<td>orig</td>
<td>0.1565</td>
</tr>
<tr>
<td>fuss</td>
<td>0.1748</td>
</tr>
<tr>
<td>1K</td>
<td>8.4840</td>
</tr>
<tr>
<td>2K</td>
<td>3.3221</td>
</tr>
<tr>
<td>4K</td>
<td>1.6816</td>
</tr>
<tr>
<td>8K</td>
<td>1.3291</td>
</tr>
</tbody>
</table>

### TABLE V
**ENERGY CONSUMPTION IN hydro2d/fct**

<table>
<thead>
<tr>
<th>Core Energy (J)</th>
<th>Memory Energy (J)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-way</td>
</tr>
<tr>
<td>orig</td>
<td>0.0008</td>
</tr>
<tr>
<td>fuss</td>
<td>0.0006</td>
</tr>
<tr>
<td>1K</td>
<td>0.0290</td>
</tr>
<tr>
<td>2K</td>
<td>0.0130</td>
</tr>
<tr>
<td>4K</td>
<td>0.0086</td>
</tr>
<tr>
<td>8K</td>
<td>0.0066</td>
</tr>
</tbody>
</table>

- **fuss**: loop fusion
- **TABLE IV** is a two large one-dimensional loop nest resulting a very large loop body (fused, but resource not enough)
- **TABLE V** is a smaller resulting loop (fused, resource enough)
Energy experimental results -4

- **fiss**: loop fission

### TABLE VI
Energy Consumption in **nasa7/cholesky**

<table>
<thead>
<tr>
<th>Core Energy (J)</th>
<th>Memory Energy (J)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-way</td>
</tr>
<tr>
<td>orig</td>
<td></td>
</tr>
<tr>
<td>1K</td>
<td>10.0226</td>
</tr>
<tr>
<td>2K</td>
<td>9.0886</td>
</tr>
<tr>
<td>4K</td>
<td>7.5097</td>
</tr>
<tr>
<td>8K</td>
<td>5.4660</td>
</tr>
<tr>
<td><strong>0.2717</strong></td>
<td></td>
</tr>
</tbody>
</table>

| fiss            |        |        |        |        |
| 1K              | 11.0594| 10.1304| 9.8596 | 9.8654 |
| 4K              | 7.4571 | 6.0289 | 6.1112 | 5.8381 |
| 8K              | 5.2994 | 5.2730 | 5.1896 | 5.2517 |
| **0.2875**      |        |        |        |        |
**Table VII**

**Core Energy Breakdown (in %s) for matmult in Hardware Components Level**

<table>
<thead>
<tr>
<th>Version</th>
<th>Register File</th>
<th>Pipeline Registers</th>
<th>Functional Units</th>
<th>Data-path Muxes</th>
</tr>
</thead>
<tbody>
<tr>
<td>orig</td>
<td>35.99</td>
<td>36.33</td>
<td>15.76</td>
<td>8.36</td>
</tr>
<tr>
<td>l</td>
<td>36.09</td>
<td>34.87</td>
<td>17.34</td>
<td>8.11</td>
</tr>
<tr>
<td>u</td>
<td>36.19</td>
<td>36.17</td>
<td>15.98</td>
<td>8.31</td>
</tr>
<tr>
<td>t</td>
<td>34.60</td>
<td>33.56</td>
<td>19.93</td>
<td>7.80</td>
</tr>
<tr>
<td>lu</td>
<td>35.87</td>
<td>34.12</td>
<td>18.19</td>
<td>7.93</td>
</tr>
<tr>
<td>lt</td>
<td>35.27</td>
<td>33.74</td>
<td>19.25</td>
<td>8.17</td>
</tr>
<tr>
<td>tu</td>
<td>35.31</td>
<td>35.07</td>
<td>17.89</td>
<td>8.06</td>
</tr>
<tr>
<td>tlu</td>
<td>35.41</td>
<td>34.15</td>
<td>18.38</td>
<td>7.96</td>
</tr>
</tbody>
</table>
Result Analysis

- Energy consumed: memory system is higher than core in unoptimized codes.

- Most performance optimizations:
  - Overall energy consumption
  - Data-path energy consumption

- Cache size and degree of associativity:
  - Data cache energy
  - Overall energy consumption by reducing number of accesses to main memory
Conclusions

- **Power minimization** is an important task of embedded system design.

- To be aimed at **Energy Consumption**, optimized methods of textbooks may not be best.

- More **optimal optimization parameters** from energy must be considered, such as **tile size in tiling**.