Energy-Conscious Compilation Based on Voltage Scaling

H. Saputra, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, J. S. Hu
Microsystems Design Lab
Dept. of Computer Science and Engineering
The Pennsylvania State University
C-H. Hsu, U. Kremer
Dept. of Computer Science
Rutgers University
Overview

- Motivation
- Contributions
- Voltage Scaling Basics
- Static Voltage Scaling
- Dynamic Voltage Scaling
- Conclusions and Future Work
Motivation

- Minimizing energy consumption is being promoted to a first-class status in system designs;
- Software based work for energy optimization is still in its infancy;
- One of these areas that are in much need of work is energy-aware compilers;
- Compiler optimization for energy is utmost important for embedded systems in which software constitutes a significant portion;
Motivation

- Design and implement energy-conscious compiler for both performance and energy/power behavior optimization.
- Are current performance-oriented compiler optimization suitable from the energy perspective? If not, How to improve?
- Are there pure energy-oriented compiler optimizations that do not affect its performance? How do they interact with pure performance-oriented optimizations?
Contributions

- We have built a strategy where classical loop-level optimization techniques are used for improving energy consumption instead of improving performance;

- We present an ILP based compilation strategy that compiles a given code under multiple energy and performance constraints.
Voltage Scaling Basics

- Dynamic energy consumption is proportional to $KCV^2$
  
  Reducing $V$ has a quadratic effect

- Clock Cycle Time is proportional to $\frac{V}{(V-Vt)^2}$ for long channel CMOS
  
  Reducing $V$ increases clock cycle time
Static Voltage Scaling

- Apply a single (lower) voltage for the entire program/loop nests;
- Reducing total energy consumption without increasing its execution time.

⇒ How?
First apply performance-oriented loop optimization
Then do voltage scaling only for CPU part such that the execution time will be the same as the original one (flexible).
Static Voltage Scaling

- Original execution time: $X_0 = L_0 T_0$, $L_0$-number of execution cycles, $T_0$-clock cycle time;

- After optimization: $X_i = L_i T_0$, $L_i < L_0$ if optimization is successful;

- Decompose the optimized time: $X_i = L_{icpu} T_0 + X_m$
  $L_{icpu}$-number of cycles spent in CPU, $X_m$-time spent in off-chip memory;

- Apply voltage scaling to obtain a new clock cycle time $T_r (> T_0)$, such that $X_n = L_{icpu} T_r + X_m$
  and $X_n = X_0$ (flexible).
Static Voltage Scaling

Normalized Energy Consumption

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Original</th>
<th>Before Voltage Scaling</th>
<th>After Voltage Scaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>eflux</td>
<td>80</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>tsf</td>
<td>100</td>
<td>110</td>
<td>120</td>
</tr>
<tr>
<td>tomcatv</td>
<td>80</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>jpeg</td>
<td>70</td>
<td>80</td>
<td>90</td>
</tr>
</tbody>
</table>
Static Voltage Scaling

Normalized Execution Time

Normalized Execution Time

Benchmarks

Normalized Execution Time

Original  Before Voltage Scaling  After Voltage Scaling
Dynamic Voltage Scaling

- Uses multiple supply voltages within the execution of an application;
- There is an overhead time when voltage is switching from one level to another;
- This approach is based on ILP (Integer Linear Programming).
Dynamic Voltage Scaling

- Having tables that contain pre-computing energy consumption and execution time of multiple voltage versions;
- For this work, we used a granularity at the boundary of loop nests;
- If we have M different nests and N different voltage levels ➔ we have 2 tables, each has NM entries.
## Dynamic Voltage Scaling

### Supply Voltage Levels

<table>
<thead>
<tr>
<th></th>
<th>2.5 V</th>
<th>2.3 V</th>
<th>2.1 V</th>
<th>…………</th>
<th>1.1 V</th>
<th>0.9 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1</td>
<td>3329.24</td>
<td>3236.31</td>
<td>3151.12</td>
<td>…………</td>
<td>2841.35</td>
<td>2802.63</td>
</tr>
<tr>
<td>N2</td>
<td>318.18</td>
<td>315.87</td>
<td>313.76</td>
<td>…………</td>
<td>306.08</td>
<td>305.12</td>
</tr>
<tr>
<td>N3</td>
<td>15807.6</td>
<td>14687.5</td>
<td>13660.7</td>
<td>…………</td>
<td>9926.92</td>
<td>9460.20</td>
</tr>
<tr>
<td>………</td>
<td>…………</td>
<td>…………</td>
<td>…………</td>
<td>…………</td>
<td>…………</td>
<td>…………</td>
</tr>
<tr>
<td>N8</td>
<td>6570.82</td>
<td>6337.78</td>
<td>6124.17</td>
<td>…………</td>
<td>5347.38</td>
<td>5250.28</td>
</tr>
<tr>
<td>N9</td>
<td>6074.30</td>
<td>5917.53</td>
<td>5773.82</td>
<td>…………</td>
<td>5251.25</td>
<td>5185.93</td>
</tr>
</tbody>
</table>

**Energy Consumption in uJ (tomcatv)**
Dynamic Voltage Scaling

<table>
<thead>
<tr>
<th></th>
<th>2.5 V</th>
<th>2.3 V</th>
<th>2.1 V</th>
<th>..........</th>
<th>1.1 V</th>
<th>0.9 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1</td>
<td>565.66</td>
<td>619.45</td>
<td>688.01</td>
<td>..........</td>
<td>1850.86</td>
<td>2888.74</td>
</tr>
<tr>
<td>N2</td>
<td>26.43</td>
<td>27.35</td>
<td>28.54</td>
<td>..........</td>
<td>46.68</td>
<td>66.65</td>
</tr>
<tr>
<td>N3</td>
<td>6508.61</td>
<td>7270.75</td>
<td>8242.32</td>
<td>..........</td>
<td>24719.4</td>
<td>39425.8</td>
</tr>
<tr>
<td>N8</td>
<td>1649.88</td>
<td>1824.07</td>
<td>2046.12</td>
<td>..........</td>
<td>5811.98</td>
<td>9173.16</td>
</tr>
<tr>
<td>N9</td>
<td>1125.3</td>
<td>1234.51</td>
<td>1373.72</td>
<td>..........</td>
<td>3734.66</td>
<td>5841.89</td>
</tr>
</tbody>
</table>

Execution time in msec (tomcatv)
Dynamic Voltage Scaling

Let’s assume:

- $E_{ij}$: is the estimated energy consumption for nest $i$ using the $j^{th}$ voltage level;
- $X_{ij}$: is the estimated time for nest $i$ using the $j^{th}$ voltage level;
- $S_{ij}$: equals to 1 if and if nest $i$ using the $j^{th}$ voltage level;

where $1 = i = M$ and $1 = j = N$. 
Dynamic Voltage Scaling

- ILP formulation (ZILP):

  Voltage selection: \[ \sum_{j=1}^{N} s_{i,j} = 1 \]

  Total energy: \[ E = \sum_{i=1}^{M} \sum_{j=1}^{N} s_{i,j} E_{i,j} \]

  Total execution time: \[ X = \sum_{i=1}^{M} \sum_{j=1}^{N} s_{i,j} X_{i,j} \]

  Overhead for voltage switching:

\[
O = \sum_{i=1}^{M-1} \sum_{j=1}^{N} |s_{i,j} - s_{i+1,j}| \left( \frac{OVHD}{2} \right)
\]
Dynamic Voltage Scaling

Case-I: \( \min X \)  
Case-III: \( \min E \) with \( X = 15000 \)

Case-II: \( \min E \)  
Case-IV: \( \min X \) with \( E = 36000 \)

Case-V: \( \min E \) with \( X + O = 20000 \)

Case-VI: \( \min X + O \) with \( E = 36000 \)
Dynamic Vs Static

Percentage energy improvements over static voltage scaling

2002-10-10
Related Work

- Chandrakasan et. al., Optimizing power using transformation, 1995;
- F. Yao et. al., A scheduling model for reduced cpu energy, 1995;
- T. Ishihara and H. Yasuura, ISLPED-98;
- V. Swaminathan and K. Chakrabarty, ASPDAC’01;
- K. Govil et al., MOBICOM-95;
- C-H. Hsu et. al., ISLPED’01, PACS’02;
Conclusions

- We present two compiler-directed voltage scaling strategies:
  - static voltage scaling: single voltage for entire code (optimized);
  - dynamic voltage scaling: ILP based, multiple voltage selection for different loop nests;
- Dynamic strategy generates better energy results than the static one.
Future Work

- Apply voltage scaling at different granularities;
- Implement fully-dynamic strategies where the voltage levels to be used are negotiated at runtime;
- Investigate the impact of different optimizations on the effectiveness of static and dynamic voltage scaling.
Thank you!