JouleTrack - A Web Based Tool for Software Energy Profiling

Amit Sinha and Anantha Chandrakasan
Massachusetts Institute of Technology

June 19, 2001
Outline

- Motivation for fast software energy profiling
- Experimental setup – StrongARM and SH-4
- Current profiling measurements and implications
  - Instruction level
  - Program level
  - Models
- Leakage energy estimation methodology
  - Processor leakage model
  - Leakage vs. switching
- JouleTrack design
- Conclusions
Motivation

- Proliferation of portable devices
  - 50X increase in µP power vs. 4X in battery life
  - Battery technology lags behind

- Programmable solutions preferred
  - Fast software profiling techniques required
Scenarios

- Portable devices downloading apps
- Predictive energy estimation

- Wireless sensor networks
- Lifetime estimation
- Energy-Quality tradeoffs

- Integrated Development Environments
- Energy aware software
Software Energy Estimation

- [Tiwari96] proposed instruction level analysis
- Disadvantages
  - Tedious instruction level analysis
  - Large simulation time and storage requirement
  - Very fine grained
- Macro-estimation?
Experimental Setup

- Intel StrongARM SA-1100 Brutus platform
- Hitachi SH-4 based SH7750 platform
SA-1100 Instruction Current

Supply $V_{DD}$ 1.5 V
Frequency 206 MHz
Load/Store 0.196 A
ALU 0.178 A

- Little variation in instruction currents!

[Montanaro, JSSC '96]
• Similar variation with MACs and Branches consuming more current
Current almost independent of program!

First order model, $I = f(V_{DD}, \text{frequency})$
Second Order Estimation

- Determine energy differentiated instruction classes and weights for those classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction</td>
<td>2.17</td>
</tr>
<tr>
<td>Sequential Memory access</td>
<td>0.03</td>
</tr>
<tr>
<td>Non-sequential Memory access</td>
<td>1.24</td>
</tr>
<tr>
<td>Internal</td>
<td>0.84</td>
</tr>
</tbody>
</table>

SA-1100 Example

Measure statistics for those classes at runtime

Program current = Weighted average

Error < 3%
Model Calibration

- Benchmark Programs
  - Execute
  - Measure Current
  - Measure Statistics
  - Determine Weights

\[
\bar{P} = \{P_0, P_1, \ldots, P_{N-1}\}
\]

\[
W = [w_0 \ w_1 \ \ldots \ w_{K-1}]^T
\]

\[
\bar{I} = [I_0 \ I_1 \ \ldots \ I_{N-1}]^T
\]

\[
\bar{C} =
\begin{bmatrix}
  c_0^0 & c_1^0 & c_{K-1}^0 \\
  c_0^1 & c_1^1 & c_{K-1}^1 \\
  c_0^{N-1} & c_1^{N-1} & c_{K-1}^{N-1}
\end{bmatrix}
\]

\[
\bar{W} = \frac{1}{I_0(V_{dd}, f)} (\bar{C}^T \bar{C})^{-1} \bar{C}^T \bar{I}
\]
Second order model reduces average error to 2.8% (based on 66 test programs)
Voltage Frequency Variation

\[ E_{\text{switch}} = C_{\text{tot}} V_{DD}^2 \]

\[ E_{\text{leak}} = V_{DD} I_0 \exp\left(\frac{V_{dd}}{nV_T}\right) t \]
Leakage Model

- Leakage currents account for 10% of energy dissipation
- Leakage behavior is exponential with supply voltage
- $I_0 = 1.2\text{mA}$, $n = 21.3$ for the StrongARM
Explanation of Exponential Model

\[ I = Ae^{\frac{(V_{GS} - V_{th} - \gamma V_S - \eta V_{DS})}{nV_T}} \left(1 - e^{-\frac{V_{DS}}{V_T}}\right) \]

\[ A = \mu_0 C_0 e^{\frac{W_{eff}}{L_{eff}}} V_T^2 e^{1.8} \]

<table>
<thead>
<tr>
<th>Vdd (V)</th>
<th>I_{leak} (mA)</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>20.41</td>
<td>20.10</td>
</tr>
<tr>
<td>1.4</td>
<td>16.35</td>
<td>16.65</td>
</tr>
<tr>
<td>1.3</td>
<td>13.26</td>
<td>13.80</td>
</tr>
<tr>
<td>1.2</td>
<td>12.07</td>
<td>11.43</td>
</tr>
<tr>
<td>1.1</td>
<td>9.39</td>
<td>9.47</td>
</tr>
<tr>
<td>1.0</td>
<td>7.96</td>
<td>7.85</td>
</tr>
<tr>
<td>0.9</td>
<td>6.39</td>
<td>6.53</td>
</tr>
</tbody>
</table>

- Processor leakage model can be explained from transistor subthreshold leakage model
- Model error < 6%

\[ I_1 : I_2 : I_3 = 1.8e^{n V_{DD}/n V_T} : 1.8 : 1 \]

[Gu96]
Leakage energy component can dominate switching energy for high supply voltage operation!
Low Duty Cycle Effects

- Leakage occurs at all times!
  \[ D = 100\% \Rightarrow \text{Leakage is 10}\% \]
  \[ D = 10\% \Rightarrow \text{Leakage > 50}\% \]

- Just in time processing will reduce leakage energy

Duty Cycle: \( D = \frac{T_1}{T} \)

Switching Energy

Leakage Energy

Task completes

Idle Mode

Time

Energy

Graph showing the relationship between energy and duty cycle.
JouleTrack Design

Remote User

Web Server

Common Gateway Interface

Compiler Linker

Simulator

Libraries

Errors

source.c

source.exe

Cycle counts
Statistics

Energy Estimation

Energy Profile

Logs

Average Currents, Weights

www

Compiler

Stats
Timing Estimation Engine

- Memory Map
- Executable Image
- Clock Speed
- Memory Timing:
  - 00000000 8000 SRAM 4 rw 1/1 1/1
  - 00008000 8000 ROM 2 r 100/100 100/100
  - 00010000 8000 DRAM 2 rw 150/100 150/100
- Cycle Accurate Simulator Core
- Execution Statistics
- RDI
- Profiler
- Tracer
- Memory Models
- Co-processor
- OS Models
JouleTrack Snapshot

http://dry-martini.mit.edu/JouleTrack
4000 hits since Sept-2000
Conclusions

- Fast macro-level program current estimation demonstrated
  - No need for elaborate trace profiling
  - Can be incorporated in an energy aware application development environment

- Very little instruction current variation because of common overheads

- Leakage currents becoming significant
  - 10% at 100% duty cycle on StrongARM
  - Leakage measurement technique presented