High Level Power Estimation and Optimization

Diana Marculescu
Carnegie Mellon University
Dept. of Electrical and Computer Engineering
Center for Silicon System Implementation
http://www.ece.cmu.edu/~dianam

Why low power?

Handhelds

Ubiquitous Computing

Portable

Desktops

Servers
Current power challenges

<table>
<thead>
<tr>
<th>Power related system cost drivers</th>
<th>Servers</th>
<th>Desktops</th>
<th>Mobile</th>
<th>Handhelds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermal cost ($)</td>
<td>Thermal cost ($)</td>
<td>Thermal cost ($)</td>
<td>Form factor (in^3)</td>
<td></td>
</tr>
<tr>
<td>Delivery cost ($)</td>
<td>Delivery cost ($)</td>
<td>Delivery cost ($)</td>
<td>Battery size (lbs.)</td>
<td></td>
</tr>
<tr>
<td>Form factor (in^3)</td>
<td>Form factor (in^3)</td>
<td>Battery size (lbs.)</td>
<td>Battery cost ($)</td>
<td></td>
</tr>
</tbody>
</table>

Price drivers

<table>
<thead>
<tr>
<th>Perf (SPEC, TPC-C)</th>
<th>Perf (SPEC, MHz)</th>
<th>Perf (MIPS, MHz)</th>
<th>Perf/battery hrs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perf/in^3</td>
<td>Perf/MHz</td>
<td>Perf/MHz</td>
<td>Perf/battery hrs</td>
</tr>
</tbody>
</table>

Source: V. Tiwari, D. Singh, 1999

All price drivers are ultimately dependent on the power efficiency

Power consumption trends

Source: V. Tiwari, D. Singh, Intel, 1999
Power density trends

Source: F. Pollack, Intel, 1999

Effect on the economy…

- California blackouts
  - Most of e-commerce or web-based e-tailers are based in CA
  - Lots of MWhs, not enough infrastructure
Handheld/portable applications

What we’d like…

- Protocols, ECC, ...
- Voice I/O compression & decompression
- Handwriting recognition
- Text/Graphics processing
- Video decompression
- Speech recognition
- Java interpreter
- Up to 1 month of uninterrupted operation!

Desktop/server applications

What we’d like…

- Media processing (image, video, audio)
- High performance computing
- Server applications
  - E-commerce
  - Database
  - ........
- Seamless internet connections
- High MIPS/Watt rate, low power density!
Key issues

- How can we efficiently estimate power consumption early in the design cycle?

- How can we build power efficient systems?

Outline

- High level power estimation
  - Gate level (brief overview)
  - RT/microarchitectural level
  - SW and system level

- High level power optimization
  - Gate level (brief overview)
  - RT/microarchitecture/SW/system level
  - Energy aware computing (power management)
Key question

How can we efficiently estimate power consumption early in the design cycle?

Power estimation and abstraction level

- **System**: seconds, 20-50%
- **Architecture**: minutes, 1-5%
- **RT/Microarchitecture**: hours
- **Gate**: days
- **Circuit**: seconds, 20-50%
Power estimation accuracy vs. speed

Static techniques

Dynamic techniques

Sources of power dissipation

- Switching power: due to charging and discharging of output capacitance:
  \[
  \text{Energy/transition} = C_L \cdot V_{dd}^2 \\
  \text{Power} = \text{Energy/transition} \cdot f - C_L \cdot V_{dd}^2 \cdot f
  \]

- Short-circuit power: due to non-zero rise/fall times
- Leakage power (important with decreasing device sizes)
A Typical Example

\[ \begin{array}{c}
\text{S}_1 \quad \text{4-bit adder} \quad \text{S}[0:3] \\
\text{A}[0:3] \quad \text{B}[0:3] \\
\hline
00011110111000111010 \\
00111110111000111010 \\
011110111000111010 \\
11101110011110100000 \\
11101110011110100000 \\
11101110011110100000 \\
11110001111010000001 \\
\hline
\text{P}_{\text{total}} = 456 \, \mu\text{W}
\end{array} \]

\[ \begin{array}{c}
\text{S}_2 \quad \text{4-bit adder} \quad \text{S}[0:3] \\
\text{A}[0:3] \quad \text{B}[0:3] \\
\hline
111111111100000000 \\
000000000011111111 \\
000000000011111111 \\
111000000011111110 \\
001100100111100111 \\
101010101010101001 \\
\hline
\text{P}_{\text{total}} = 365 \, \mu\text{W}
\end{array} \]

A Typical Example (cont’d)

\[ \begin{array}{c}
\text{S}_1 \quad \text{4-bit adder} \quad \text{S}[0:3] \\
\text{A}[0:3] \quad \text{B}[0:3] \\
\hline
00011110111000111010 \\
00111110111000111010 \\
011110111000111010 \\
11101110011110100000 \\
11101110011110100000 \\
11101110011110100000 \\
11110001111010000001 \\
\hline
\text{P}_{\text{total}} = 456 \, \mu\text{W}
\end{array} \]

\[ \begin{array}{c}
\text{S}_{1}^\prime \quad \text{4-bit adder} \quad \text{S}[0:3] \\
\text{A}[0:3] \quad \text{B}[0:3] \\
\hline
100001111011 \\
000111101011 \\
000111101010 \\
011110110110 \\
001110111000 \\
001101110001 \\
001011100011 \\
010111000111 \\
\hline
\text{P}_{\text{total}} = 446 \, \mu\text{W}
\end{array} \]
Gate-level power estimation

- Challenges:
  - Need node-by-node accuracy
  - $V_{dd}$, $f_{clk}$, $C_L$ are known
    - Actually, layout will determine the interconnect capacitance
  - Need to estimate switching activity accurately – issues
    - Spatial and temporal dependencies – circuit (reconvergent fanout) and input induced

Transition density

- $D(z) = p(y) \cdot D(x) + p(x) \cdot D(y)$
- $\frac{\partial z}{\partial x} = y$, $\frac{\partial z}{\partial y} = x$; thus, if $p(x) = p(y) = 0.5$:
  - $D(x) = D(y) = 0.5$:
    - $D(z) = p(y) \cdot D(x) + p(x) \cdot D(y) = 0.5$
- $p(x) = p(y) = 0.5; D(x) = 0.5, D(y) = 1$
  - $D(z) = 0.75$ (actually: $D(z) = 0.5$)

Source: F. Najm, 1991
Probabilistic simulation

- Handles RFO
- Global OBDDs
  - Expensive

- Handles temporal correlations

```
\[ V_n V_{n-1} \ldots V_2 V_1 \]
```

```
\begin{array}{cccc}
1 & 0 & \ldots & 0 \\
1 & 1 & \ldots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 0 & \ldots & 1 \\
1 & 0 & \ldots & 1 \\
0 & 1 & \ldots & 0 \\
\end{array}
```

\[ x \]

```
Q = \begin{bmatrix}
x & x \\
p_{00} & p_{10} \\
p_{01} & p_{11}
\end{bmatrix}
```

Source: S. Devadas, 1992
Why temporal correlations?

\[ sw(x) = p(x_0 \rightarrow 1) + p(x_1 \rightarrow 0) = \frac{2p^x_{1,0}p^x_{0,1}}{p^x_{1,0} + p^x_{0,1}} \]

Why not just simply \( sw(x) = 2p(x = 1)p(x = 0) \)?

\[ sw(x) = 2 \cdot p(x = 0) \cdot p(x = 1) = 2 \frac{p^x_{1,0}p^x_{0,1}}{(p^x_{1,0} + p^x_{0,1})^2} \]

Spatial correlations

- Spatial correlation due to:
  - Circuit topology
  - Input dependence
Pairwise spatiotemporal correlations

- Exact spatiotemporal correlations → exponential complexity
- Pairwise correlations → quadratic complexity

Typical results

- Error [%]
- With spatiotemporal correlations
- Without spatiotemporal correlations

Diana Marculescu, CMU
RT-level power estimation

- Challenges:
  - Details of IP core implementation is largely unknown
  - Have to abstract switching activity and capacitance values
  - Less accurate, but more efficient
    - Efficient design space exploration

RT-level power estimation overview

- Macromodeling
  - Pre-characterization based
  - Regression-based

- Input training set selection
  - Sequence compaction
  - Sampling
An example

- Idea: Characterize the average switching activity of a module \( SW_{mj} \) by the average switching activity for a typical internal line in that module \( sw_{avg} \).
- \( P_{mj} \propto \sum c_i \cdot sw_i \approx sw_{avg} \cdot \sum c_i = sw_{avg} \cdot C_{mj} \)

Entropy and Informational energy

- For a complete set of events \( A_1, A_2, \ldots, A_n \) with probabilities \( p_1, p_2, \ldots, p_n \):
  - Entropy:
    - \( H(A) = -\sum p_i \log(p_i) \)
  - Informational energy:
    - \( E(A) = \sum p_i^2 \)
Entropy and switching activity

Example: The truth table for a randomly excited 1-bit full adder:

<table>
<thead>
<tr>
<th>$c_i$</th>
<th>$x_i$</th>
<th>$y_i$</th>
<th>$s_i$</th>
<th>$c_i + 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1</td>
<td>1 0 1</td>
<td>0 1 0</td>
<td>1 0 1</td>
<td></td>
</tr>
<tr>
<td>1 0 0</td>
<td>1 0 0</td>
<td>0 0 1</td>
<td>0 0 1</td>
<td></td>
</tr>
<tr>
<td>1 1 0</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td></td>
</tr>
<tr>
<td>1 1 1</td>
<td>0 1 1</td>
<td>1 0 1</td>
<td>0 1 1</td>
<td></td>
</tr>
<tr>
<td>0 1 1</td>
<td>1 1 0</td>
<td>1 0 0</td>
<td>1 0 0</td>
<td></td>
</tr>
<tr>
<td>1 0 1</td>
<td>0 1 1</td>
<td>1 1 0</td>
<td>1 1 0</td>
<td></td>
</tr>
<tr>
<td>0 1 0</td>
<td>1 0 1</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td></td>
</tr>
</tbody>
</table>

Output space is partitioned into 4 classes $\Pi = \{10, 01, 11, 00\}$

No transitions occur within a class, only across different classes.

We may thus assume that for any set $\xi$ of nets: $SW(\xi) \leq H(\xi)/2$.

The 1-bit case

If $x$ is modeled as a lag-one Markov chain,

$$sw(x) = \frac{2 \cdot p_{01} \cdot p_{10}}{p_{01} + p_{10}}$$

$$h(x|x) = -p_1 \left( p_{10} \log p_{10} + p_{11} \log p_{11} \right) - p_0 \left( p_{00} \log p_{00} + p_{01} \log p_{01} \right)$$

$$h(x|x) \geq 2 \cdot sw(x) \cdot (p_{00} + p_{11}) \Rightarrow h(x) \geq 2 \cdot sw(x).$$
Basic assumptions

- **A1.** Uniform Distribution: Nodes are uniformly distributed over the levels of the circuit.

- **A2.** Uniform Variation: The average bit entropy and informational energy are estimated in terms of the effective scaling factor as: \( h_j = h_0 \cdot d^j \) and \( e_j = 1 - (1 - e_0) \cdot d^j \).

- **A3.** Asymptotic Values: The number of levels \( N \) is large enough to apply \( N \to \infty \).

Theoretical result

- **Corollary:** If A1, A2, A3 hold, the average entropy/informational energy per bit are:

  \[
  h_{\text{avg}} = \frac{h_{\text{in}} - h_{\text{out}}}{\ln \left( \frac{h_{\text{in}}}{h_{\text{out}}} \right)} \quad \quad e_{\text{avg}} = 1 - \frac{e_{\text{in}} - e_{\text{out}}}{\ln \left( \frac{1 - e_{\text{out}}}{1 - e_{\text{in}}} \right)}
  \]

  \( \text{sW}_{\text{avg}} \approx h_{\text{avg}}/2 \approx 1 - e_{\text{avg}} \).
**DFG switching activity**

\[
\begin{align*}
&\text{SW}_{\text{avg(sim)}} = 0.1734 \\
&\text{SW}_{\text{avg(est)}} = 0.1805 (0.1683)
\end{align*}
\]

**Design exploration**

\[
\begin{align*}
&\text{SW}_{\text{avg(est)}} = 0.186 (0.197) \\
&\text{SW}_{\text{avg(est)}} = 0.242 (0.282)
\end{align*}
\]
Power Macromodeling

Consider a data path block:

\[ Pwr = C_0 + C_1 \cdot S_1 + C_2 \cdot S_2 + C_3 \cdot S_3 + C_4 \cdot S_4 \]

Sign bit

\[ S_1, S_2 : \text{avg. switching activity of LSB (MSB) region of operand 1} \]
\[ S_3, S_4 : \text{avg. switching activity of LSB (MSB) region of operand 2} \]
Input/Output Data Model

Consider a data path block:

\[ Pwr = C_0 + C_1 \cdot S_1 + C_2 \cdot S_2 + C_3 \cdot S_3 \]

- \( S_1, S_2 \): avg. switching activity of operands 1 and 2
- \( S_3 \): avg. switching activity of output

Bitwise Data Model

Consider a random logic block:

\[ Pwr = C_0 + \sum_{inputs} C_i \cdot S_i \]

- \( S_i \): avg. switching activity of input signal \( i \)

More parameters lead to a higher degree of accuracy, but increase the computational overhead.
Constrained Sequence Generation

Initial Input Sequence $L_0$ → Target Circuit

SSM Synthesis (Compaction/Generation)

Initial Input Sequence $L_0$ → Target Circuit
Compacted Sequence $L \ll L_0$

Stochastic vs. Deterministic

Example (Moore-type Machines)

<table>
<thead>
<tr>
<th>X</th>
<th>$S^n$</th>
<th>$S^{n+1}$</th>
<th>Y</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$1/2$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$1/2$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$1/4$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$3/4$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$2/3$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$1/3$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$1/2$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$1/2$</td>
</tr>
</tbody>
</table>

Deterministic Case

$A = \begin{bmatrix} A(0) \\ A(1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Stochastic Case

$A = \begin{bmatrix} A(0) \\ A(1) \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 2 & 2 \\ 4 & 4 \\ 2 & 2 \end{bmatrix}$
The Structure of the SSM

- A SSM = a set of deterministic FSMs, probabilistically selected.

The Decomposition Procedure

Example

- For M = (S, X, Y, {A(x)}, L) with S = X = Y = \{0, 1\}, L(0) = 1, L(1) = 0 and transition matrix A:

\[
A = \begin{bmatrix}
\frac{1}{2} & \frac{1}{2} \\
\frac{1}{4} & \frac{1}{4} \\
\frac{3}{8} & \frac{1}{3} \\
\frac{1}{2} & \frac{1}{2}
\end{bmatrix}
\]

1. \( + \frac{1}{4} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} 
2. \( + \frac{1}{6} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} 
3. \( + \frac{1}{12} \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}

- Need 4 symbols \{s_1, s_2, s_3, s_4\} with probability distribution \(P = (1/2, 1/4, 1/6, 1/12)\).
A possible solution

 aux inputs

\[ X \rightarrow w_1 \rightarrow w_2 \rightarrow S \rightarrow D \rightarrow Y \]

Experimental Results (ISCAS’85)

- Large compaction ratios (1-2 orders of magnitude) can be achieved without significant loss in accuracy.
Input Data Modeling

- A new methodology:
  - Probabilistic modeling of input streams;
  - Allows on-the-fly modeling and compaction

Input Data Model

- A first-order Markov model for the input stream
**Input Description**

- Transition matrix Q
  
  \[ p_{ij} = p(x_n = j \mid x_{n-1} = i) \]

  \[ Q = \begin{bmatrix}
  0 & 1 & 0 & 0 \\
  1/3 & 1/3 & 1/3 & 0 \\
  0 & 0 & 0 & 1 \\
  0 & 1 & 0 & 0
\end{bmatrix} \]

- State transition graph (STG)

**The case of sequential circuits**

Benchmark s8

The State Transition Graph
Finite-order memory effects

- Two sequences with the same first-order statistics:

- Same one-step transition probabilities $\neq$ same higher-order statistics.

Sequence Compaction for Sequential Circuits

- A high-order probabilistic model:

\[ z_n = \text{out}(x_m, s_n) \]

\[ s_{n+1} = \text{next}(x_m, s_n) \]
Effect of High-Order Statistics

- **Theorem:** If the sequence feeding the target circuit has order \( k \), then a lag-\( k \) Markov chain which correctly models the input sequence, also correctly models the \( k \)-step conditional probabilities of the primary inputs and internal states.

\[
p(x_n s_n | x_{n-1}s_{n-1} \ldots x_{n-k}s_{n-k}) = p(x_n | x_{n-1} \ldots x_{n-k})
\]

---

First-Order Dynamic Models

- **Example:** \( S_1 = (v1, v2, v3, v4, v5, v6, v7, v8) = (0000, 0001, 1001, 1100, 1001, 1100, 1001, 1100) \)
DMT1 Construction

- Example: $S_1 = (v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8) = (0000, 0001, 1001, 1100, 1001, 1100, 1001, 1100)$

High-Order Dynamic Markov Models

- $DMT_k$ completely captures temporal correlations of order $k$ and full spatial correlations.
The case of combinational circuits

- Flat models for compaction
  - disadvantage: do not adapt very well to input changes

Composite Sequences

- Construct $S^*$ as follows:
  - $S'_1$: 25 times $S_1$, $S'_2$: 100 times $S_2$
  - $S^*$: $S'_1$ @ $S'_2$ @ $S'_1$
Hierarchical Markov Modeling

- $(\varepsilon, \delta)$ - grouping:
  - $\{s_1, s_2, \ldots, s_n\}$ micro-states, $\{S_1, S_2, \ldots, S_p\}$ macro-states
  - $\varepsilon$ - criterion: $\forall s_i \in S_k, s_j \in S_l$ then $p(s_i|s_j) < \varepsilon$ and $p(s_j|s_i) < \varepsilon$
  - $\delta$ - criterion: $\forall S_k, \exists W_k$ s.t. $|W_k - w_{ij}| < \delta$, $s_i, s_j \in S_k$

A Hamming Distance Criterion

- Indicator for the level of activity:
  - average Hamming distance between two consecutive vectors
Experimental Setup

- Initial Sequence $L_0$
- One-step DMC modeling; build $DM^T_0$ and dynamically update counts on the edges
- Generate compacted sequence $L$
- Compacted Sequence $L << L_0$
- Gate-level logic simulation (comparison)

Experimental Results – Sequential circuits

Compaction Ratio 10

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Error [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>s9234</td>
<td></td>
</tr>
<tr>
<td>s820</td>
<td></td>
</tr>
<tr>
<td>s5378</td>
<td></td>
</tr>
<tr>
<td>s1423</td>
<td></td>
</tr>
<tr>
<td>s1196</td>
<td></td>
</tr>
<tr>
<td>shiftreg</td>
<td></td>
</tr>
<tr>
<td>planet</td>
<td></td>
</tr>
<tr>
<td>mc</td>
<td></td>
</tr>
<tr>
<td>dk17</td>
<td></td>
</tr>
<tr>
<td>bbara</td>
<td></td>
</tr>
</tbody>
</table>

Diana Marculescu, CMU
Experimental Results – Combinational circuits

Compaction Ratio 10

- C6288
- C3540
- C1908
- C1355
- C880
- C499
- C432

Error [%]

Sampling

1. Input vectors
2. Generate one sample of k units
3. Simulate
4. Converged?
   - NO
   - YES
5. “Difficult” distributions
6. Report power

Efficiency: depends on population characteristics and sampling procedure

Diana Marculescu, CMU
Comparison with compaction

L = 4,000
Compaction Ratio 10

Stratified random sampling
Estimate the average weight, assuming that gender and age of individuals are readily available

Lower sample variance leads to faster convergence
SW and system level power estimation

- Challenges:
  - Very few implementation details available
  - Need to rely on accurate lower level estimation tools
  - Complexity can be prohibitive for highly accurate estimates
    - Tools for achieving sufficiently accurate estimates, for very complex systems

Analytical cache power models

- On-chip cache
  - Energy = Ebus + Ecell + Epad + Emain...
  - Ecell = \( \beta \cdot (wl\_length) \cdot (bl\_length + 4.8) \cdot (Nhit + 2 \cdot Nmiss) \)
  - \( wl\_length = m \cdot (T + 8L + St) \)
  - \( bl\_length = C / (m \cdot L) \)
  - \( Nhit = \) number of hits; \( Nmiss = \) number of misses;
  - \( C = \) cache size; \( L = \) cache line size in bytes; \( m = \) set
  - associativity; \( T = \) tag size in bits; \( St = \) # of status bits
  - per line; \( \beta = 1.44e-14 \) (technology parameter)
Cache power results

Base Configuration:
- 4-way superscalar
- 32KB DM L1 I
- 32KB, 4-way SA L1 D
- 32B blocks, write back
- 128KB, 4-way SA L2
- 64B blocks, write back
- 1MB, 8-way SA off-chip L3
- 128B blocks, write thru

Interconnect widths:
- 16B between L1 and L2
- 32B between L2 and L3
- 64B between L3 and MM

From Ghose, 1999

Instruction level power estimation

Source: V. Tiwari et al., 1994
Software power analysis models

Table 1: Energy Cost Table (Proposed by Tiwari et al)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LOAD</td>
<td>1.06</td>
<td></td>
<td>0.13</td>
<td>0.15</td>
<td>1.19</td>
<td>0.02</td>
<td>1.25</td>
<td>1.06</td>
</tr>
<tr>
<td>DLOAD</td>
<td>2.37</td>
<td></td>
<td>0.15</td>
<td>0.17</td>
<td>1.19</td>
<td>0.02</td>
<td>1.32</td>
<td>1.06</td>
</tr>
<tr>
<td>ADD</td>
<td>0.09</td>
<td></td>
<td>1.19</td>
<td>1.19</td>
<td>0.26</td>
<td>0.53</td>
<td>0.86</td>
<td>0.99</td>
</tr>
<tr>
<td>MUL</td>
<td>1.19</td>
<td></td>
<td>0.92</td>
<td>0.92</td>
<td>0.53</td>
<td>0.06</td>
<td>0.79</td>
<td>0.96</td>
</tr>
<tr>
<td>LOADADD</td>
<td>2.10</td>
<td></td>
<td>1.25</td>
<td>1.32</td>
<td>0.86</td>
<td>0.79</td>
<td>0.40</td>
<td>0.53</td>
</tr>
<tr>
<td>LOADMUL</td>
<td>2.25</td>
<td></td>
<td>1.06</td>
<td>1.06</td>
<td>0.99</td>
<td>0.96</td>
<td>0.53</td>
<td>0.79</td>
</tr>
</tbody>
</table>

Diana Marculescu, CMU

Instruction Level Compaction

Original Program

Architectural Simulation

Characteristic Profile

New Program

Program Synthesis

CPU’s RTL Model

RTL Simulation

Power Estimation

Source: M. Pedram, 1997
### Program Synthesis Procedure

1. Block Allocation

2. Instruction Allocation
   - `add op1, op2`
   - `xor op1, op2`
   - `mul op1, op2`
   - `mov op1, mem1`
   - `branch`

3. Memory Allocation
   - `mov op1, mem1`

4. Operand Allocation/Instruction Scheduling
   - `add op1, op2`
   - `reg1`
   - `reg4`
   - reorder the sequence

---

### SimplePower

- SimplePower – microarchitectural simulator
  - Datapath intensive
  - Suitable for embedded applications
  - Macromodels for all functional units

Source: M.J. Irwin et al., 2000
Wattch

- Supports **superscalar, out-of-order** microarchitecture simulation
- Within 10% accurate compared to three different core processors

Source: D. Brooks et al., 2000

---

HW/SW power analysis

Source: M. Lajoie et al., 2000
Platform-based design

Application (application model + workload analysis)

Platform (architecture model + low-level support)

Design cycle (map application + evaluate model)

Example: MPEG-2 Decoder

Source: R. Marculescu, 2000
The Baseline Unit Model

Analysis results

- Allows resource utilization and process performance analysis
The Baseline Unit Model

- Single CPU mapping

Power Analysis Results

- Single vs. dual CPU mapping
RTOS-level power estimation

Source: R.P. Dick et al., 2000

RTOS power breakdown

Diana Marculescu, CMU
Power estimation summary

- Key step in optimizing the power cost of a system
- Software, microarchitecture and system level estimation are crucial
- Sufficiently accurate estimates
  - If lower level models are detailed/accurate enough
  - Can be up to 10-20% accurate compared to SPICE

Outline

- High level power estimation
  - Gate level (brief overview)
  - RT/microarchitectural level
  - SW and system level

- High level power optimization
  - Gate level (brief overview)
  - RT/microarchitecture/SW/system level
  - Energy aware computing (power management)
Key question

How can we build power efficient systems?

Power savings and abstraction level

- System: 50-90%
- Architecture: 5-10%
- RT/Microarchitecture
- Gate
- Circuit
Reducing power consumption

- Load capacitance ($C_L$)
  - Roughly proportional to the chip area
  - Can be reduced by logic re-synthesis or design for low power
- Switching activity ($sw = \text{avg. number of transitions/cycle}$)
  - Very data dependent
  - A big portion due to glitches (real-delay)
  - Can be reduced by power sensitive data encoding, re-synthesis (e.g. balanced designs)
- Voltage supply ($V_{dd}$)
  - Biggest impact: 50% reduction in $V_{dd}$, 75% reduction in power
  - Cannot be reduced indefinitely (cannot be too close to $V_t$ – lower $V_t$ means higher leakage power – and lower $Vdd$ increases latency)
- Clock frequency ($f$)
  - Has significant impact only in conjunction with $V_{dd}$ scaling
  - Lowering only $f$ decreases average power, but total energy is the same and throughput is worse

Gate level power optimization

- Mostly targets $C_L$ and switching activity reduction
- Reduce power via
  - Logic synthesis
  - Precomputation/guarded evaluation
  - Clock gating
Logic synthesis for low power

Logic restructuring

Technology mapping

Diana Marculescu, CMU

FSM state assignment

- code(s₂) = 001 (fixed)
- code(s₁) = 000
- code(s₃) = 011
- \( \bar{d}_2 \geq 1 \) trans/step

- code(s₂) = 001 (fixed)
- code(s₁) = 110
- code(s₃) = 111
- \( \bar{d}_2 \leq 2.5 \) trans/step

Diana Marculescu, CMU
Clock gating logic gates off the clock so that there’s no switching power in the downstream logic.

RT/behavioral/microarchitecture level power optimization

- Challenges:
  - Targets $C_L$, switching activity, as well as joint $V_{dd}/f_{clk}$ reduction
  - Need to rely on high-level (maybe not so accurate) estimates
Low power register sharing

Diana Marculescu, CMU

Low power module assignment

Source: J.-M.Chang et al., 1996

93

Source: J.-M.Chang et al., 1997

94
Multiple supply voltage assignment

- Two possible voltage assignments (under performance constraints)

Source: J.-M. Chang et al., 1997

Bus encoding

- Reduces number of bit toggles on the bus
- Different flavors
  - Bus-invert coding
    - Uses an extra bus line \textit{invert}:
      - if the number of transitions is \(< K/2\), \textit{invert} = 0 and the symbol is transmitted as is
      - if the number of transitions is \(> K/2\), \textit{invert} = 1 and the symbol is transmitted in a complemented form
  - Low-weight coding
    - Uses \textit{transition} signaling instead of \textit{level} signaling
Bus invert coding

Source: M.Stan et al., 1994

Power consumption breakdown

Typical high-performance µp

Typical embedded system
Typical high performance processor

- Issue window and datapath are the most power consuming modules (~35%) after clock power (~40%)

Trace-cache – Pentium 4

- The pipeline structure used by the Pentium Pro, Pentium II and III
- The new pipeline structure, used in the Pentium 4

Useful also for higher performance!
Trace cache for dynamic work reuse

Up to 75% of the time execute from the trace-cache

Reuse this work!

Potential for reuse

- More than 60% of the time, code can be executed directly from the trace cache
  - Front-end can be clock gated or completely shutdown
Power savings

- 20-30% on average (ideal case)
- 10-15% (10% overhead for idle modules)

Performance within 3-5% on average

Simple branch predictor for trace cache
- Less power overhead, more performance overhead
SW power optimization

- Per-instruction cost, plus inter-instruction effects
  \[ E_P = \sum (B_i \times N_i) + \sum (O_{i,j} \times N_{i;j}) + \sum E_k \]

- Modified list scheduling algorithm for low power instruction scheduling
- Operation packing
- Operand swapping

The effect of different compiler optimizations

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Original Energy (10^-8J)</th>
<th>Packing Energy</th>
<th>Scheduling Energy</th>
<th>Swapping Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>FJex1</td>
<td>2.79</td>
<td>2.46</td>
<td>2.12</td>
<td></td>
</tr>
<tr>
<td>FJex2</td>
<td>3.91</td>
<td>3.14</td>
<td>2.83</td>
<td></td>
</tr>
<tr>
<td>LP_FIT60</td>
<td>57.60</td>
<td>30.80</td>
<td>25.60</td>
<td></td>
</tr>
<tr>
<td>IIB4</td>
<td>10.10</td>
<td>7.47</td>
<td>6.78</td>
<td>6.37</td>
</tr>
<tr>
<td>FFT2</td>
<td>9.59</td>
<td>3.35</td>
<td>8.97</td>
<td>8.64</td>
</tr>
</tbody>
</table>

Source: V. Tiwari et al., 1996
Loop transformations

- Loop = loop transformation (inner vs. outer loop)
- Unrolled = loop unrolling
- Tiled = tiling for increasing locality
- TLU = all

Source: M. Kandemir et al., 2000

Low power system design

- Platform-based design
  - Workload specific mapping
  - Application and platform modeling and mapping
- Dynamic power management
  - OS-driven
  - Microarchitecture and SW supported
  - “Energy Aware Computing”
Two different media streams

Mapping choice 1

High performance mapping (consumer)
Mapping choice 2

- Low power mapping (consumer)

Energy aware computing

- Software
  (Application & System)

- Hardware
  (Microarchitecture and IP cores)
Portable system power breakdown

Advanced Computing and Power Interface (ACPI) standard
ACPI processor states

Latency:
$C_1 < C_2 < C_3$

Power throttling

Full speed $C_0$
Throttling

$C_1$
$C_2$
$C_3$

$G_0$ (working)

Power:
$C_1 > C_2 > C_3$

Power savings via Dynamic Power Management (DPM)

Source: Y.-H.Lu et al., 2000
Typical performance metrics for DPM

Classic Power Management Schemes

Diana Marculescu, CMU
Classic Power Management Schemes

- Circuit Power management logic
- Gate/RT-level power management
- Microarchitecture-driven power management

System-level power management

- C_0 (working)
- Throttling
- Full speed

C_1

C_2

C_3

Power throttling

Refine throttling state

Latency: C_1 < C_2 < C_3

Power: C_1 > C_2 > C_3

Synergistic Microarchitecture/System Power Management
Execution Profile

IPC (instructions per cycle)

Cycles
epic (image compression benchmark)

Hotspots: an Illustration

- Hotspot: collection of very frequently executing basic blocks
Fraction of Execution in Hotspots

- High temporal locality (>80%) in most common applications

Details of Profiling Hardware

Diana Marculescu, CMU
How useful is this?

- Hotspots have consistent behavior over long periods
  - 0.5M to 10s of M instructions
- Worthwhile to optimize the processor configuration for each hotspot
- Optimal configuration = minimum-energy configuration
  - Optimize energy per committed instruction, reciprocal of MIPS/W

Dynamic scaling - pipeline

- Changing the configuration means changing the active size of any unit
- A non-zero performance penalty is involved
- An example:
  - 64-entry RUU, 8-wide
  - 32-entry RUU, 4-wide
Dynamic scaling - pipeline

- Changing the configuration means changing the active size of any unit
- A non-zero performance penalty is involved
- An example:
  - 64-entry RUU, 8-wide
  - 32-entry RUU, 4-wide

Dynamic scaling - arrays

- Banked arrays:
  - Could turn off individual banks

All 4 banks enabled
Dynamic scaling – arrays

- Banked arrays:
  - Could turn off individual banks

![Only 2 banks enabled](image)

Total Energy

Average Saving: 12.3%; with DVS: 17.1%
With performance constraints: 6%
Average Power

Average Saving: 23.3%; with DVS: 28.1%
With performance constraints: 11.4%

Performance

Average IPC loss: 12.9%
With performance constraints: 5.8%
How good is all this?

- Better than static throttling methods

![Graph showing performance metrics provided by Diana Marculescu, CMU](image)

What more can we do?

- Implement as an intermediate stage in ACPI
- Add software control
What more can we do?

- Implement as an intermediate stage in ACPI
- Add software control

Power Optimization Summary

- Most of the savings:
  - Decisions taken at highest levels
    - More freedom
    - Efficient design exploration
  - Via energy aware computing or dynamic power management
  - Via application driven adaptability
    - Dynamic voltage or resource scaling
Challenges for the future

- **Short term needs**
  - More efficient battery systems: Battery lifetime trends are well below power consumption trends
  - Less power density: We are approaching sun’s power density
  - Less leakage power: For 0.1um technology, leakage is almost 50% of the total power
  - Modeling tools: To be able to optimize, we need to estimate power well before a system is built

- **Long term needs (open problems)**
  - Do we really need a clock? (Fully or partially asynchronous systems.)
  - Do we really need CMOS technology? (Nanotechnology, molecular computing…)
  - How about digital computing? (Analog is much less power consuming)
  - Any other wild ideas???

---

Industry/Consumer view

- The embedded industry has driven the quest for low power until now
- Major processor manufacturers will include power as a design constraint
- Customers will pay for MIPS/Watt or Watts, rather than for MHz
- Hopefully, the power problem will be solved before reaching the sun’s surface power density!
Summary

- For a complete low power solution, hw/sw synergy is essential

- Need for novel microarchitectural paradigms that address the problem of joint power-performance optimization

- Power consumption is still a HOT problem!

Diana Marculescu, CMU