More power to software developers!

(Kerstin Eder)

Whole Systems Energy Transparency

Trustworthy Systems Laboratory, University of Bristol
Verification and Validation for Safety in Robots, Bristol Robotics Laboratory
Learning Objectives

✓ Why software is key to energy efficient computing
✓ What energy transparency means and why we need energy transparency to achieve energy efficient computing
✓ How to measure the energy consumed by software
  ▪ How to estimate the energy consumed by software without measuring
  ▪ How to construct energy consumption models
✓ Why timing and energy analysis differ
Learning Objectives

- Why software is key to energy efficient computing
- What energy transparency means and why we need energy transparency to achieve energy efficient computing
- How to measure the energy consumed by software
  - How to estimate the energy consumed by software without measuring
- How to construct energy consumption models
- Why timing and energy analysis differ
Static Analysis of Energy Consumption
Whole Systems ENergy TRAnsparency

EC FP7 FET MINECC:

“Software models and programming methodologies supporting the strive for the energetic limit (e.g. energy cost awareness or exploiting the trade-off between energy and performance/precision).”
Acknowledgements

The partners in the EU ENTRA project

John Gallagher and team

Pedro López García and team

Henk Muller and team

Steve Kerrison, Kyriakos Gerogiou, James Pallister, Jeremy Morse and Neville Grech
SRA for Energy Consumption

- Adaptation of traditional resource usage analysis techniques to energy consumption.
- Techniques automatically infer upper and lower bounds on energy usage of a program.
- Bounds expressed using monotonic arithmetic functions per procedure parameterized by program’s input size.
- Verification can be done statically by checking that the upper and lower bounds on energy usage and any other resource defined in the specifications hold.
Specified Resource Usage

Source: Pedro Lopez Garcia, IMDEA Software Research Institute
Analysis Result

Source: Pedro Lopez Garcia, IMDEA Software Research Institute
Verification

![Diagram showing resource usage and input data size with specification and analysis bounds.](image)

Source: Pedro Lopez Garcia, IMDEA Software Research Institute
Static Energy Usage Analysis

Original Program:

```c
int fact (int x) {
    if (x<=0)\(^a\)
        return 1\(^b\);
    return (x \(*^d\) fact(x-1))\(^c\);
}
```

Extracted Cost Relations:

- \( C_{\text{fact}}(x) = C_a + C_b \quad \text{if} \ x \leq 0 \)
- \( C_{\text{fact}}(x) = C_a + C_{c}(x) \quad \text{if} \ x > 0 \)
- \( C_{c}(x) = C_d + C_{\text{fact}}(x-1) \)

- Substitute \( C_a, C_b, C_d \) with the actual energy required to execute the corresponding lower-level (machine) instructions.
Energy Modelling captures energy consumption
Modelling Considerations

- At what level should we model?
  - instruction level, i.e. machine code
  - intermediate representation of compiler
  - source code

- Models require measurements
  - need to associate entities at a given level with costs, i.e. energy consumption
    - accuracy
    - usefulness
Modelling Considerations

- **At what level should we model?**
  - instruction level, i.e. machine code
  - intermediate representation of compiler
  - source code

- **Models require measurements**
  - need to associate entities at a given level with costs, i.e. energy consumption
    - accuracy – the lower the better
    - usefulness – the higher the better

ISA-Level Energy Modelling

Energy Cost \((E)\) of a program \((P)\):

\[
E_P = \sum_i (B_i \times N_i) + \sum_{i,j} (O_{i,j} \times N_{i,j})
\]

Instruction Base Cost, \(B_i\), of each instruction \(i\)

Circuit State Overhead, \(O_{i,j}\), for each instruction pair

Components of an Energy Model:

\[ E_P = \sum_i (B_i \times N_i) + \sum_{i,j} (O_{i,j} \times N_{i,j}) \]

- \( B_i \) and \( O_{i,j} \) are energy costs.
- Characterization of a model through measurement produces these values for a given processor.

ISA-Level Energy Modelling

Components of an Energy Model:

\[ E_P = \sum_i (B_i \times N_i) + \sum_{i,j} (O_{i,j} \times N_{i,j}) \]

- \( N_i \) is the number of times that instruction \( i \) is executed, and
- \( N_{i,j} \) is the number of times that the execution of instruction \( i \) is followed by the execution of instruction \( j \).

Exercise: $E(\text{fact}(3))$?

```
int fact (int x) {
    int ret = x;
    while (--x)
    {
        ret *= x;
    }
    return ret;
}
```

How much energy does a call to $\text{fact}(3)$ consume?

```
fact:
sub    r3, r0, #1
cmp    r3, #0
beq    .L2
.L3:    mul    r0, r3
sub    r3, r3, #1
cmp    r3, #0
bne    .L3
.L2:    bx     lr
```
Base Cost Characterization

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Base Cost [pJ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>sub</td>
<td>600</td>
</tr>
<tr>
<td>cmp</td>
<td>300</td>
</tr>
<tr>
<td>beq</td>
<td>500</td>
</tr>
<tr>
<td>mul</td>
<td>900</td>
</tr>
<tr>
<td>bne</td>
<td>500</td>
</tr>
<tr>
<td>bx</td>
<td>700</td>
</tr>
</tbody>
</table>

fact:

```
sub    r3, r0, #1
cmp    r3, #0
beq    .L2

.L3:
mul    r0, r3
sub    r3, r3, #1
cmp    r3, #0
bne    .L3

.L2:
bx     lr
```
Overhead Characterization

fact:
  sub    r3, r0, #1
  cmp    r3, #0
  beq    .L2

.L3:
  mul    r0, r3
  sub    r3, r3, #1
  cmp    r3, #0
  bne    .L3

.L2:
  bx     lr

<table>
<thead>
<tr>
<th>$O_{i,j}$</th>
<th>beq</th>
<th>bne</th>
<th>bx</th>
<th>cmp</th>
<th>mul</th>
<th>sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>beq</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>bne</td>
<td>10</td>
<td>0</td>
<td>10</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>bx</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>cmp</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>mul</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>30</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>sub</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>0</td>
</tr>
</tbody>
</table>
## Instruction Characterization

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Base Cost [pJ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>beq</td>
<td>500</td>
</tr>
<tr>
<td>bne</td>
<td>500</td>
</tr>
<tr>
<td>bx</td>
<td>700</td>
</tr>
<tr>
<td>cmp</td>
<td>300</td>
</tr>
<tr>
<td>mul</td>
<td>900</td>
</tr>
<tr>
<td>sub</td>
<td>600</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$O_{i,j}$ [pJ]</th>
<th>beq</th>
<th>bne</th>
<th>bx</th>
<th>cmp</th>
<th>mul</th>
<th>sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>beq</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>bne</td>
<td>10</td>
<td>0</td>
<td>10</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>bx</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>cmp</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>mul</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>30</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>sub</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>0</td>
</tr>
</tbody>
</table>
 ISA-Level Energy Modelling

Components of an Energy Model:

\[ E_P = \sum_i (B_i \times N_i) + \sum_{i,j} (O_{i,j} \times N_{i,j}) \]

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Base Cost [pJ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>beq</td>
<td>500</td>
</tr>
<tr>
<td>bne</td>
<td>500</td>
</tr>
<tr>
<td>bx</td>
<td>700</td>
</tr>
<tr>
<td>cmp</td>
<td>300</td>
</tr>
<tr>
<td>mul</td>
<td>900</td>
</tr>
<tr>
<td>sub</td>
<td>600</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( O_{i,j} ) [pJ]</th>
<th>beq</th>
<th>bne</th>
<th>bx</th>
<th>cmp</th>
<th>mul</th>
<th>sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>beq</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>bne</td>
<td>10</td>
<td>0</td>
<td>10</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>bx</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>cmp</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>mul</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>30</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>sub</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>0</td>
</tr>
</tbody>
</table>

ISA-Level Energy Modelling

Components of an Energy Model:

\[ E_P = \sum_i (B_i \times N_i) + \sum_{i,j} (O_{i,j} \times N_{i,j}) \]

- \( N_i \) and \( N_{i,j} \) represent the number of times specific instructions and instruction pairs *are executed*.
- How can we determine these?

Which instruction sequence is being executed for a call to \texttt{fact}(3)?
A call to \texttt{fact(3)} would invoke the following instructions in this order:

- \texttt{sub, cmp, beq (not taken)},
- \texttt{mul, sub, cmp, bne (taken)},
- \texttt{mul, sub, cmp, bne (not taken)},
- \texttt{bx}
Exercise

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Base Cost [pJ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>beq</td>
<td>500</td>
</tr>
<tr>
<td>bne</td>
<td>500</td>
</tr>
<tr>
<td>bx</td>
<td>700</td>
</tr>
<tr>
<td>cmp</td>
<td>300</td>
</tr>
<tr>
<td>mul</td>
<td>900</td>
</tr>
<tr>
<td>sub</td>
<td>600</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$O_{i,j}$ [pJ]</th>
<th>beq</th>
<th>bne</th>
<th>bx</th>
<th>cmp</th>
<th>mul</th>
<th>sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>beq</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>bne</td>
<td>10</td>
<td>0</td>
<td>10</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>bx</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>cmp</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>mul</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>30</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>sub</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>0</td>
</tr>
</tbody>
</table>

A call to $\text{fact}(3)$ would invoke the following instructions in this order:
- sub, cmp, beq (not taken),
- mul, sub, cmp, bne (taken),
- mul, sub, cmp, bne (not taken),
- bx
Exercise

\[ E_P = \sum_i (B_i \times N_i) + \sum_{i,j} (O_{i,j} \times N_{i,j}) \]

\text{sub, cmp, beq (not taken), mul, sub, cmp, bne (taken), mul, sub, cmp, bne (not taken), bx}

\[ E_{fact(3)} = \]
Exercise

\[ E_P = \sum_i (B_i \times N_i) + \sum_{i,j} (O_{i,j} \times N_{i,j}) \]

sub, cmp, beq (not taken), mul, sub, cmp, bne (taken), mul, sub, cmp, bne (not taken), bx

\[ E_{\text{fact}(3)} = 3\times600\text{pJ} + 3\times300\text{pJ} + 500\text{pJ} + 2\times900 + 2\times500\text{pJ} + 700\text{pJ} + 3\times20\text{pJ} + 10\text{pJ} + 30\text{pJ} + 2\times30\text{pJ} + 2\times10\text{pJ} + 30\text{pJ} + 10\text{pJ} \]

\[ = 6920\text{pJ} = 6.92\text{nJ} \]
Energy Cost \((E)\) of a program \((P)\):

\[
E_P = \sum_i (B_i \times N_i) + \sum_{i,j} (O_{i,j} \times N_{i,j})
\]

- Instruction Base Cost, \(B_i\), of each instruction \(i\)
- Circuit State Overhead, \(O_{i,j}\), for each instruction pair

Is it really this easy?

Energy Cost \((E)\) of a program \((P)\):

\[
E_P = \sum_i (B_i \times N_i) + \sum_{i,j} (O_{i,j} \times N_{i,j}) + \sum_k E_k
\]

- Instruction Base Cost, \(B_i\), of each instruction \(i\)
- Circuit State Overhead, \(O_{i,j}\), for each instruction pair
- Other Instruction Effects

Energy Modelling

Energy Cost \( (E) \) of a program \( (P) \):

\[
E_P = \sum_i (B_i \times N_i) + \sum_{i,j} (O_{i,j} \times N_{i,j}) + \sum_k E_k
\]

- Instruction Base Cost, \( B_i \), of each instruction \( i \)
- Circuit State Overhead, \( O_{i,j} \), for each instruction pair
- Other Instruction Effects (stalls, cache misses, etc)

Energy Cost \((E)\) of a multi-threaded program \((P)\):

\[
E_P = P_{\text{base}} N_{\text{idle}} T_{\text{clk}} + \sum_{t=1}^{N_t} \sum_{i \in \text{ISA}} \left( (M_t P_i O + P_{\text{base}}) N_{i,t} T_{\text{clk}} \right)
\]

- Idle base power and duration
- Concurrency cost, instruction cost, generalised overhead, base power and duration

- Use of execution statistics rather than execution trace.
- Fast running model with an average error margin of less than 7%.

The set up...

ISA Characterization

ALU instructions - 32-bit data

Even threads instruction (name & encoding)

Odd threads instruction (name & encoding)

Power (mW)
The Cost of Communication

The Swallow Platform:
- 480 processor embedded system, based on the XMOS XS1 architecture
- 16 cores per slice
Biquad Filter Example

Implemented in various configurations on Swallow:

- 7 threads on 1 core,
- 7 threads across 2 cores,
  - Good spatial locality,
  - Poor spatial locality,
- 7 threads across 7 cores.
Comms example: Biquad filter

- 7-stage biquad filter implemented in various configurations on Swallow.
- Active cores, latency, contention and under/over-allocation all affect total energy.
- Power, energy & time a valuable triple.
Energy Consumption Analysis enables energy transparency
Energy Consumption Analysis enables energy transparency
SRA at the ISA Level

- Combine static resource analysis (SRA) with the ISA-level energy model.

- Provide energy consumption function parameterised by some property of the program or its data.
Static Energy Usage Analysis

Original Program:

```c
int fact (int x) {
    if (x<=0) {a
        return 1; b
    }
    return (x * fact(x-1)); c
}
```

Extracted Cost Relations:

- \( C_{\text{fact}}(x) = C_a + C_b \) if \( x\leq 0 \)
- \( C_{\text{fact}}(x) = C_a + C_c(x) \) if \( x>0 \)
- \( C_c(x) = C_d + C_{\text{fact}}(x-1) \)

- **Substitute** \( C_a, C_b, C_d \) with the actual energy required to execute the corresponding lower-level (machine) instructions.

- **Solve equation using off-the-shelf solvers.**

- **Result:** \( C_{\text{fact}}(x) = (26x + 19.4) \text{ nJ} \)
ISa-Level Analysis Results

ISA-Level Analysis Results

Analysis Options

- Moving away from the underlying model risks loss of accuracy.
- But it brings us closer to the original source code.
Energy Consumption of LLVM IR

\[ E(ir_i) = \sum_{isa_j \in R(ir_i)} E(isa_j) \]


http://dx.doi.org/10.1145/2764967.2764974
SRA for Energy Consumption

EC Static Analysis Results

Profiling-based Energy Estimation

Energy Consumption Profiling

Learning Objectives

✓ Why software is key to energy efficient computing
✓ What energy transparency means and why we need energy transparency to achieve energy efficient computing
✓ How to measure the energy consumed by software
✓ How to estimate the energy consumed by software without measuring
✓ How to construct energy consumption models
  ▪ Why timing and energy analysis differ
Learning Objectives

- Why software is key to energy efficient computing
- What energy transparency means and why we need energy transparency to achieve energy efficient computing
- How to measure the energy consumed by software
- How to estimate the energy consumed by software *without* measuring
- How to construct energy consumption models
- Why timing and energy analysis differ
The Worst Case ...
ISA Characterization
Static Resource **Bound** Analysis

Source: Pedro Lopez Garcia, IMDEA Software Research Institute
Worst Case Execution Time

- Worst Case Execution Time (WCET) Analysis:
  - WCET model
  - WCET bounds (often for safety critical applications)
    • safe, i.e. no underestimation
    • tight, i.e. ideally very little overestimation

Does this work for energy consumption analysis?

From “The Worst-Case Execution-Time Problem — Overview of Methods and Survey of Tools” by WILHELM et al. (2008)
Worst Case Energy Consumption

- WCEC analysis goes well beyond WCET analysis.
  - embedded real-time systems that are timing predictable execute instructions in a fixed number of clock cycles
  - timing variability has mostly been eliminated “by design” through the use of synchronous logic
  - WCET then depends only on the WC execution path

- But, energy consumption is data dependent.
ISA Characterization

ALU instructions - 32-bit data

Even threads instruction (name & encoding)

Odd threads instruction (name & encoding)
W/A/B-Case Energy Consumption
W/A/B-Case Energy Consumption
a*b = b*a
Energy(a*b) ≠ Energy(b*a)
Dynamic Energy can be significant

- Data dependent switching costs can be large, ~30%
- Some instructions can cause as much dynamic energy as static (sub)
- How can we account for context-dependent switching costs?
- Can WCEC be safe and tight?
Many instructions exhibit statistical properties

Different instruction distributions can be composed

Can statistically impossible energy be considered a safe upper bound?
ABSTRACT

Safely meeting Worst Case Energy Consumption (WCEC) criteria requires accurate energy modeling of software. We investigate the impact of instruction operand values upon energy consumption in cacheless embedded processors. Existing instruction-level energy models typically use measurements from random input data, providing estimates unsuitable for safe WCEC analysis.

We examine probabilistic energy distributions of instructions and propose a model for composing instruction sequences using distributions, enabling WCEC analysis on program basic blocks. The worst case is predicted with statistical analysis. Further, we verify that the energy of embedded benchmarks can be characterised as a distribution, and compare our proposed technique with other methods of estimating energy consumption.

1 INTRODUCTION

In real-time embedded systems, execution time of a program must be bounded. This can provide guarantees that tasks will meet hard deadlines and the system will function without failure. Recently, efforts have been made to give upper bounds on program energy consumption to determine if a task will complete within an available window without under-estimation. Current models have not been analysed in this context to provide sufficient confidence, and power figures from manufacturer datasheets are not sufficiently detailed to provide tight bounds.
Critical questions for WCEC modelling:

- *Which data* should be used to characterize a WCEC model?
- *Which data causes the WCEC* for a given program?
- *Which data triggers the most switching* during the execution of the program?
Energy of an Instruction Sequence

100 data values provided to a sequence of 8 instructions
ranking of the instruction sequence’s energy up to instruction x
Energy of an Instruction Sequence

100 data values provided to a sequence of 8 instructions
ranking of the instruction sequence’s energy up to instruction $x$

by input

experiments conducted by James Pallister
Energy of an Instruction Sequence

100 data values provided to a sequence of 8 instructions
ranking of the instruction sequence’s energy up to instruction x

by input

and by output

experiments conducted by James Pallister
On the Limitations of Analyzing Worst-Case Dynamic Energy of Processing

JEREMY MORSE, STEVE KERRISON, and KERSTIN EDER, University of Bristol

This article examines dynamic energy consumption caused by data during software execution on deeply embedded microprocessors, which can be significant on some devices. In worst-case energy consumption analysis, energy models are used to find the most costly execution path. Taking each instruction’s worst-case energy produces a safe but overly pessimistic upper bound. Algorithms for safe and tight bounds would be desirable. We show that finding exact worst-case energy is NP-hard, and that tight bounds cannot be approximated with guaranteed safety. We conclude that any energy model targeting tightness must either sacrifice safety or accept overapproximation proportional to data-dependent energy.

CCS Concepts: • Hardware → Chip-level power issues;

Additional Key Words and Phrases: Energy transparency, complexity, worst case energy consumption

ACM Reference format:

1 INTRODUCTION

A significant design constraint in the development of embedded systems is that of resource consumption. Software executed on embedded hardware typically has very limited memory and computing performance available, and yet must meet the requirements of the system. To aid the design process, analysis tools such as profilers or maximum-stack-depth estimators provide the developer
Complexity Analysis

- Determining switching costs is NP-hard
  - Amount of computation required increases exponentially with program size
  - Problem cannot be approximated accurately

- No algorithm can efficiently find dynamic energy, so other questions must be posed
  - *Is a less general solution acceptable?*
  - *What level of inaccuracy can be tolerated?*

Impact of Datapath Switching

Summing up

- To achieve *Energy Transparency*
  - Energy modelling is a challenge
    - Fundamental research questions
      - data-dependent energy models
      - compositional
      - probabilistic techniques
  - Analysis techniques for energy consumption
    - SRA works best for IoT-type systems
    - Hybrid, profiling-based techniques for more complex architectures
Learning Objectives

✓ Why software is key to energy efficient computing
✓ What energy transparency means and why we need energy transparency to achieve energy efficient computing
✓ How to measure the energy consumed by software
✓ How to estimate the energy consumed by software without measuring
✓ How to construct energy consumption models
✓ Why timing and energy analysis differ
Towards Energy Aware Software Engineering
Energy Transparency

- For HW designers:
  “Power is a 1\textsuperscript{st} and last order design constraint.”
  [Dan Hutcheson, VLSI Research, Inc., E\textsuperscript{3}S Keynote 2011]

- “Every design is a point in a 2D plane.”
  [Mark Horowitz, E\textsuperscript{3}S 2009]
Energy Transparency

- For HW designers:
  “Power is a 1\textsuperscript{st} and last order design constraint.”
  [Dan Hutcheson, VLSI Research, Inc., E\textsuperscript{3}S Keynote 2011]

- “Every design is a point in a 2D plane.”
  [Mark Horowitz, E\textsuperscript{3}S 2009]
Energy Transparency

- For HW designers:
  “Power is a 1st and last order design constraint.”
  [Dan Hutcheson, VLSI Research, Inc., E³S Keynote 2011]

- “Every design is a point in a 2D plane.”
  [Mark Horowitz, E³S 2009]
Energy Transparency

- For HW designers:
  “Power is a 1st and last order design constraint.”
  [Dan Hutcheson, VLSI Research, Inc., E³S Keynote 2011]

- “Every design is a point in a 2D plane.”
  [Mark Horowitz, E³S 2009]
More POWER to SW Developers

- Full Energy Transparency from HW to SW
- New programming models

“Cool” code for green software
A cool programming competition!
Energy rated software

We aim to promote energy efficiency to a 1st class SW design goal!

Pictures taken from the Energy Efficient Computing Brochure at:
Thank you for your attention

Kerstin.Eder@bristol.ac.uk