Software and energy aware computing

(Part II)

Kerstin Eder

Design Automation and Verification, Microelectronics
Verification and Validation for Safety in Robots, Bristol Robotics Laboratory

University of BRISTOL
Department of COMPUTER SCIENCE
The Royal Academy of Engineering
XMOS®

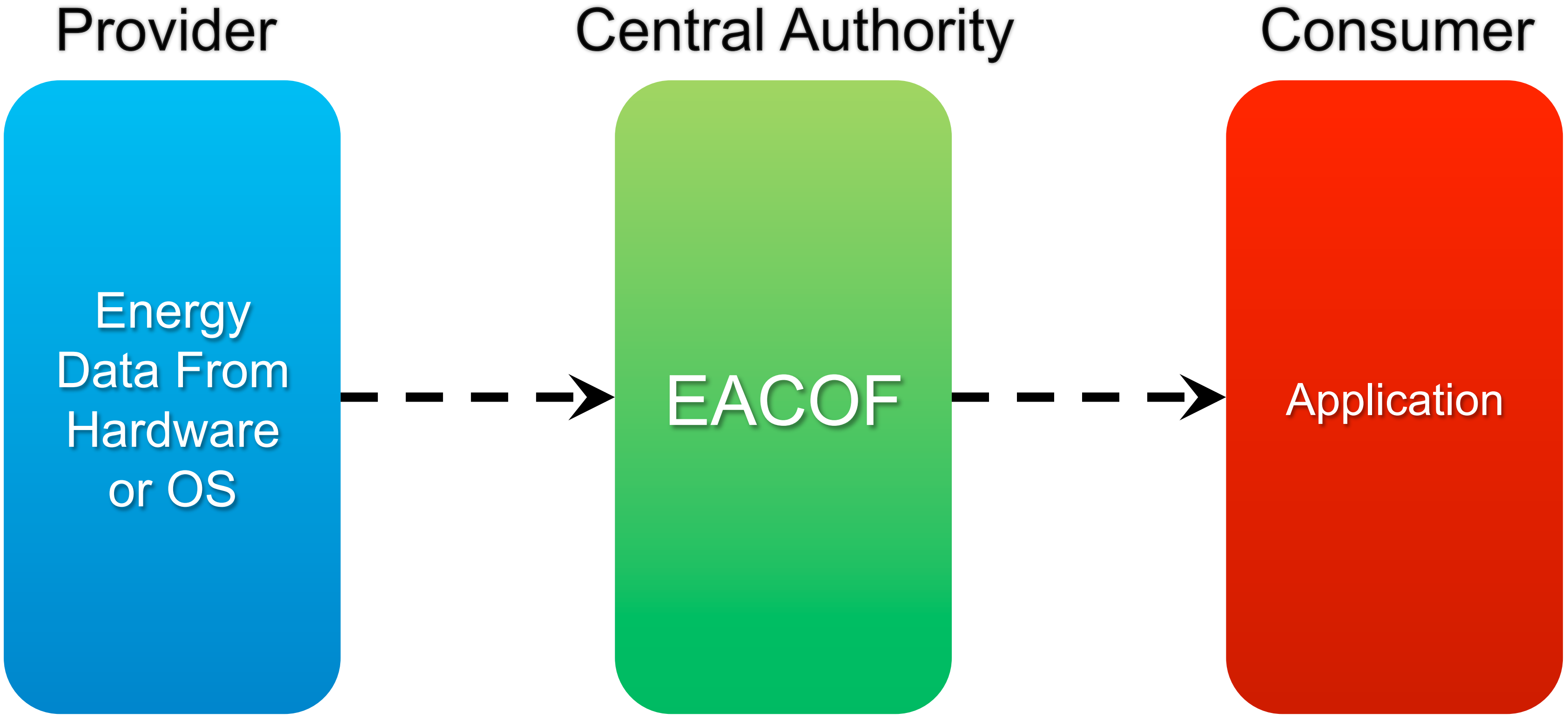
Dynamic Energy Monitoring for desktop applications

The EACOF

A simple Energy-Aware COmputing Framework

https://github.com/eacof

High Level



Provider

Central Authority

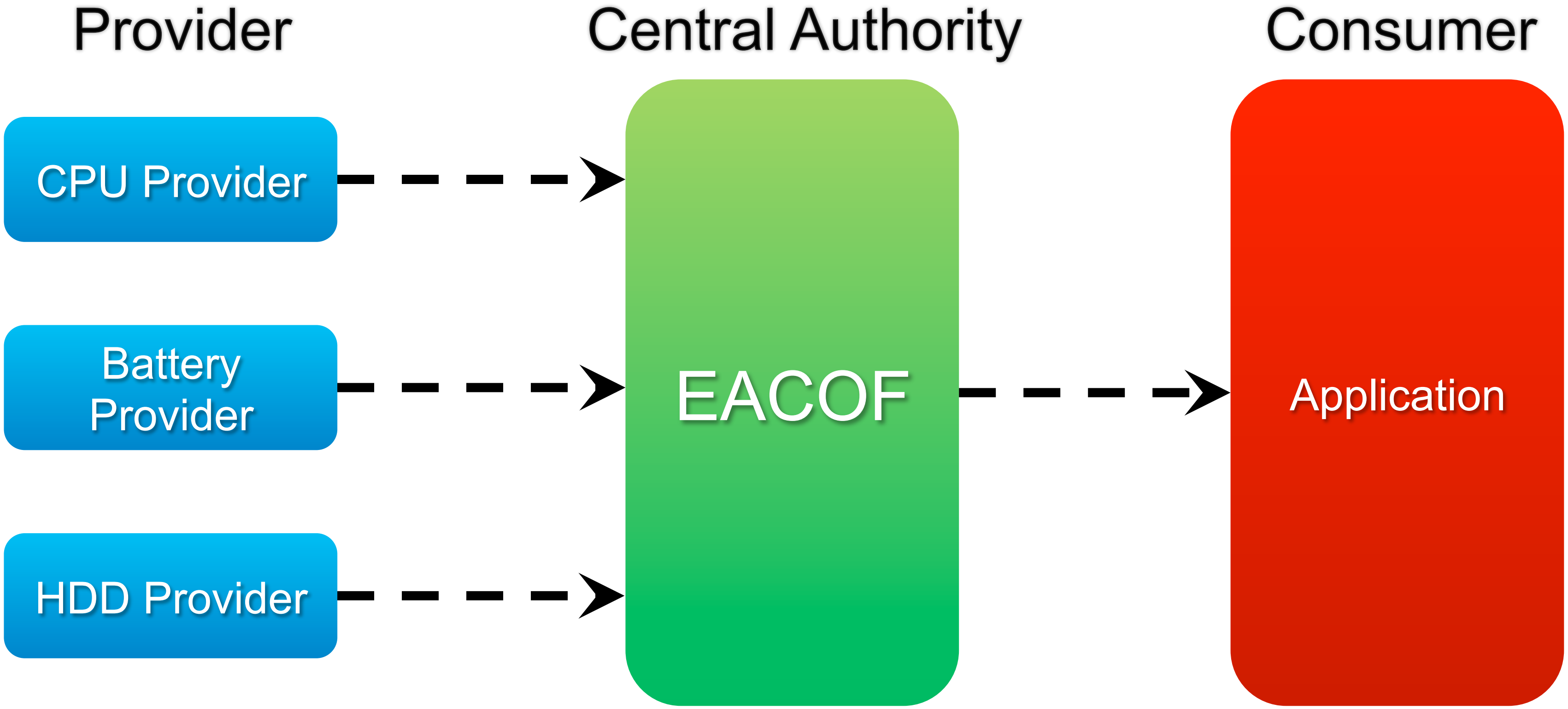
Consumer

Energy Data From Hardware or OS

EACOF

Application

Providers



Provider

Central Authority

Consumer

CPU Provider

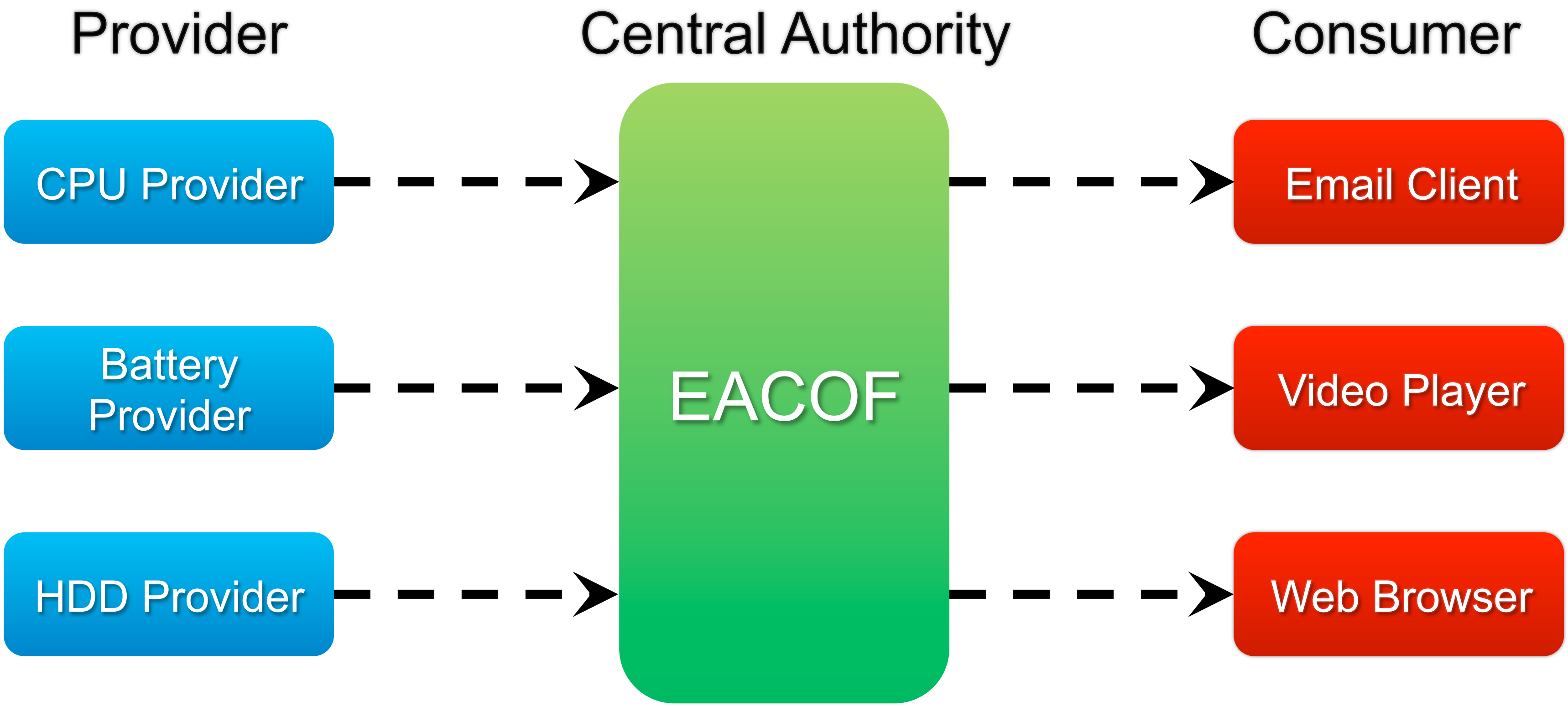
Battery Provider

HDD Provider

EACOF

Application

Consumers



Provider

Central Authority

Consumer

CPU Provider

Battery Provider

HDD Provider

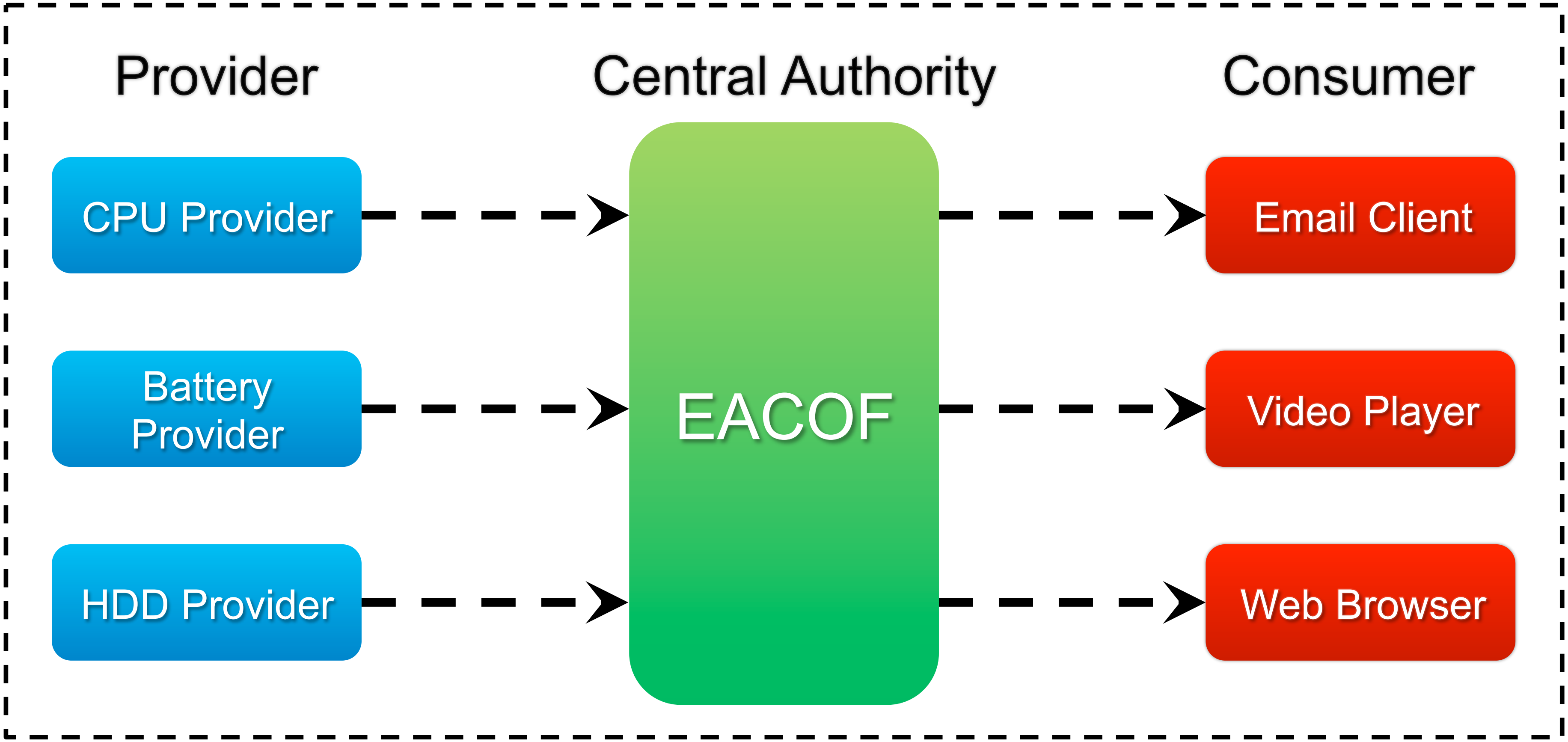
EACOF

Email Client

Video Player

Web Browser

One Machine



Provider

Central Authority

Consumer

CPU Provider

Battery Provider

HDD Provider

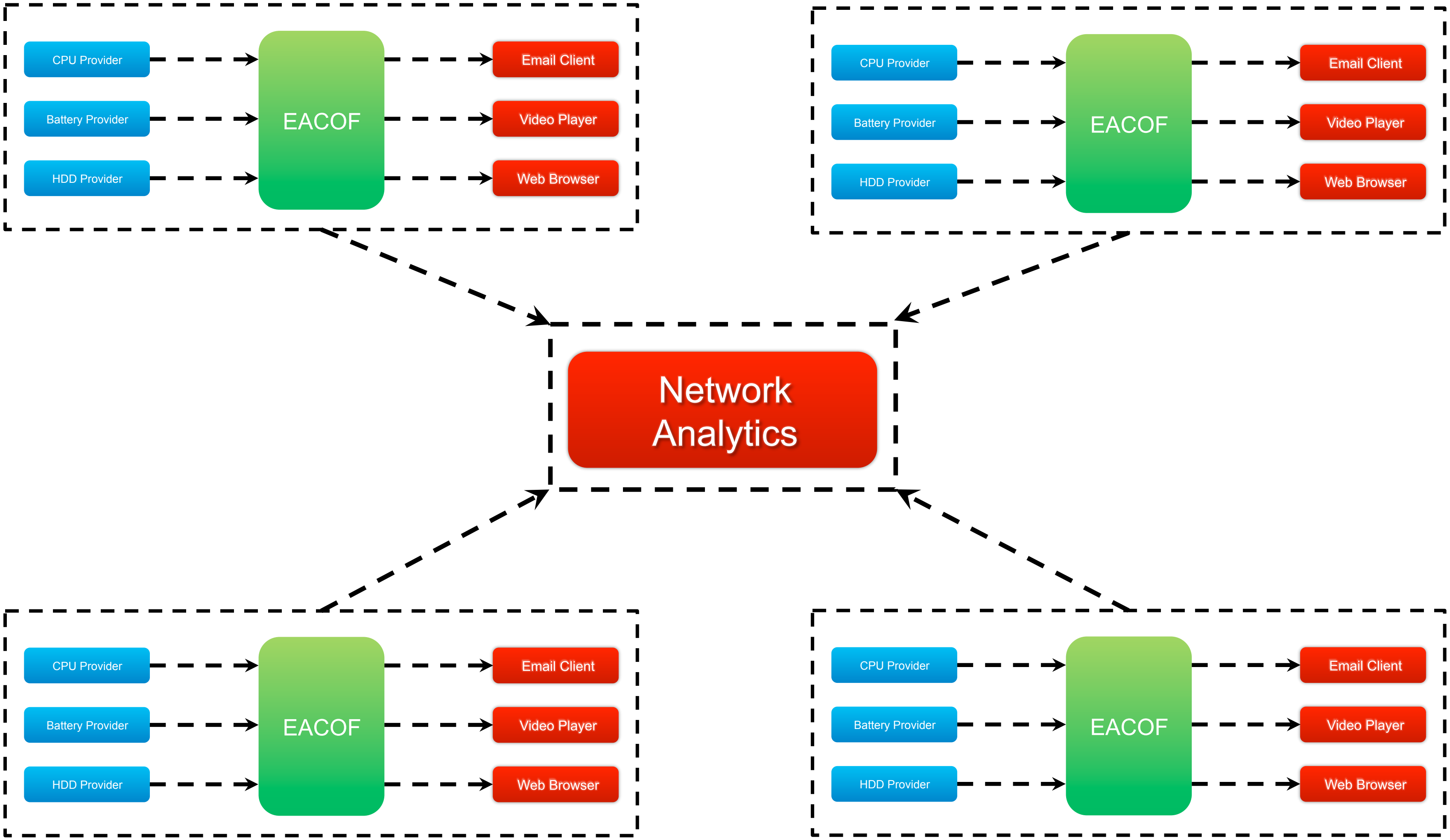
EACOF

Email Client

Video Player

Web Browser

Networked


CPU Provider
Battery Provider
HDD Provider
EACOF
Email Client
Video Player
Web Browser
CPU Provider
Battery Provider
HDD Provider
EACOF
Email Client
Video Player
Web Browser
Network Analytics
CPU Provider
Battery Provider
HDD Provider
EACOF
Email Client
Video Player
Web Browser
CPU Provider
Battery Provider
HDD Provider
EACOF
Email Client
Video Player
Web Browser

How to use EACOF

Simple Provider Example

```
while(1) {
  collectEnergyData();
  waitABit();
}
```

Simple Provider Example + EACOF

```
#include <eacof.h>
eacof_Probe *probe;
eacof_Sample sample;
initEACOF();
createProbe(&probe, 1, EACOF_DEVICE_BATTERY_ALL);
while(1) {
  sample = collectEnergyData();
  addSample(probe, sample);
  waitABit();
}
deleteProbe(&probe);
```

Simple Consumer Example

```
for (int i = 0; i < 10000; i++) {
   printf(“Hello EACOF!”);
}
```

Simple Consumer Example + EACOF

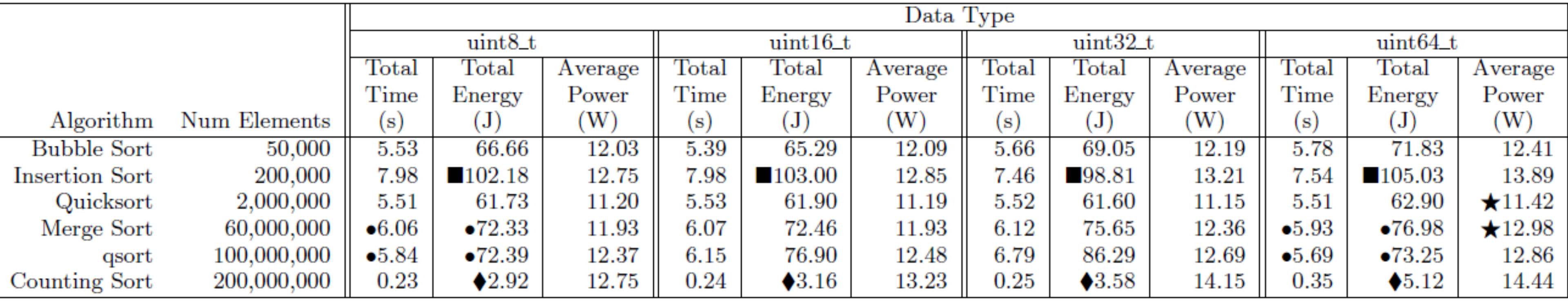
```
#include <eacof.h>
eacof_Checkpoint *checkpoint;
eacof_Sample sample;
initEACOF();
setCheckpoint(&checkpoint, EACOF_PSPEC_ALL, 1,
   EACOF_DEVICE_BATTERY_ALL);
for (int i = 0; i < 10000; i++) {
   printf(“Hello EACOF!\n”);
   sampleCheckpoint(checkpoint, &sample);
}
deleteCheckpoint(&checkpoint);
```

The EACOF API

```
#include <eacof.h>
initEACOF();
createProbe(); deleteProbe();

activateProbe(); deactivateProbe();

addSample();

setCheckpoint(); deleteCheckpoint();
sampleCheckpoint();
```

Comparing Sorting Algorithms

- Sorting of integers in [0,255]



		Data Type											
		uint8_t			uint16_t			uint32_t			uint64_t		
Algorithm	Num Elements	Total Time (s)	Total Energy (J)	Average Power (W)	Total Time (s)	Total Energy (J)	Average Power (W)	Total Time (s)	Total Energy (J)	Average Power (W)	Total Time (s)	Total Energy (J)	Average Power (W)
Bubble Sort	50,000	5.53	66.66	12.03	5.39	65.29	12.09	5.66	69.05	12.19	5.78	71.83	12.41
Insertion Sort	200,000	7.98	■102.18	12.75	7.98	■103.00	12.85	7.46	■98.81	13.21	7.54	■105.03	13.89
Quicksort	2,000,000	5.51	61.73	11.20	5.53	61.90	11.19	5.52	61.60	11.15	5.51	62.90	★11.42
Merge Sort	60,000,000	•6.06	•72.33	11.93	6.07	72.46	11.93	6.12	75.65	12.36	•5.93	•76.98	★12.98
qsort	100,000,000	•5.84	•72.39	12.37	6.15	76.90	12.48	6.79	86.29	12.69	•5.69	•73.25	12.86
Counting Sort	200,000,000	0.23	♦2.92	12.75	0.24	♦3.16	13.23	0.25	♦3.58	14.15	0.35	♦5.12	14.44

■ Insertion Sort: 32 bit version more optimized

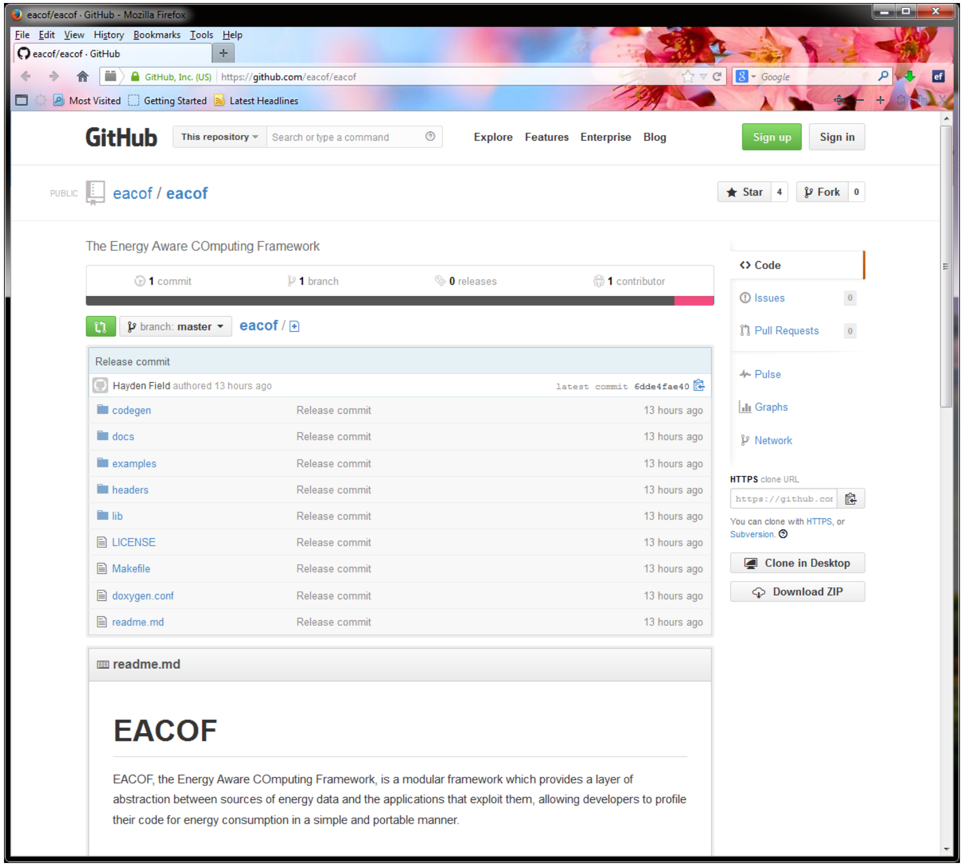
♦ Counting Sort:
75% more energy for 64 bit compared to 8 bit values

• Sorting 64 bit values takes less time than sorting 8 bit values, but consumed more energy

★ Average power variations between algorithms

H. Field, G. Anderson and K. Eder. "EACOF: A Framework for Providing Energy Transparency to enable Energy-Aware Software Development". *29th ACM Symposium On Applied Computing*. pp. 1194–1199. March 2014, ACM.

Invitation: EACOF is open source!



The Energy Aware COmputing Framework

Release commit

Hayden Field authored 13 hours ago — latest commit 6dde4fae40

codegen	Release commit	13 hours ago
docs	Release commit	13 hours ago
examples	Release commit	13 hours ago
headers	Release commit	13 hours ago
lib	Release commit	13 hours ago
LICENSE	Release commit	13 hours ago
Makefile	Release commit	13 hours ago
doxygen.conf	Release commit	13 hours ago
readme.md	Release commit	13 hours ago

You can clone with HTTPS, or Subversion.

readme.md

EACOF

EACOF, the Energy Aware COmputing Framework, is a modular framework which provides a layer of abstraction between sources of energy data and the applications that exploit them, allowing developers to profile their code for energy consumption in a simple and portable manner.

github.com/eacof

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 m*easuring
- How to construct energy consumption models

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

Static Analysis of Energy Consumption

ENTRA

Whole-Systems
Energy Transparency

The ENTRA Project

- 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

UNIVERSITAS ROSKILDENSIS · IN TRANQUILLO MORS · IN FLUCTU VITA

John Gallagher and team

imdea
instituto madrileño de estudios avanzados

Pedro López García and team

XMOS®

Henk Muller and team

University of BRISTOL

Steve Kerrison, Kyriakos Gerogiou, James Pallister, Jeremy Morse and Neville Grech

Static Energy Usage Analysis

Original Program:

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

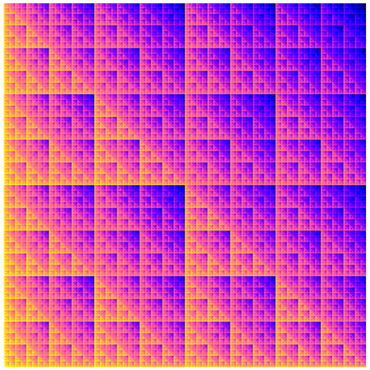
Extracted Cost Relations:

$$
\begin{array}{llll}
C_{fact}(x) & = C_a + C_b & & \text{if } x<=0 \\
C_{fact}(x) & = C_a + C_c(x) & & \text{if } x>0 \\
C_c(x) & = C_d + C_{fact}(x-1) & &
\end{array}
$$

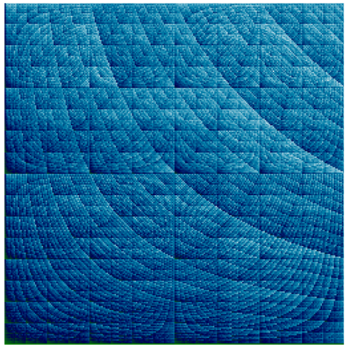
- Substitute C_a, C_b, C_d with the **actual energy required to execute the corresponding lower-level (machine) instructions.**

Energy Modelling

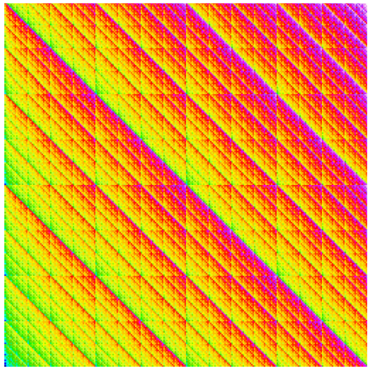
captures energy consumption



∧



×



+

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

http://www.speechinaction.org/wp-content/uploads/2012/10/dilemma.jpg

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

Based on V. Tiwari, S. Malik and A. Wolfe. “Instruction Level Power Analysis and Optimization of Software”, Journal of VLSI Signal Processing Systems, 13, pp 223-238, 1996.

ISA-Level Energy Modelling

Components of an Energy Model:

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

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

Based on V. Tiwari, S. Malik and A. Wolfe. “Instruction Level Power Analysis and Optimization of Software”, Journal of VLSI Signal Processing Systems, 13, pp 223-238, 1996.

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.

Based on V. Tiwari, S. Malik and A. Wolfe. “Instruction Level Power Analysis and Optimization of Software”, Journal of VLSI Signal Processing Systems, 13, pp 223-238, 1996.

Exercise: E(`fact(3)`)?

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

```
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
```

How much energy does a call to `fact(3)` consume?

Base Cost Characterization

Instruction	Base Cost [pJ]
sub	600
cmp	300
beq	500
mul	900
bne	500
bx	700

```
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
```

$O_{i,j}$ [pJ]	beq	bne	bx	cmp	mul	sub
beq	0	10	10	30	30	30
bne	10	0	10	30	30	30
bx	10	10	0	60	60	60
cmp	10	10	10	0	20	20
mul	10	10	10	30	0	30
sub	10	10	10	20	30	0

Instruction Characterization

Instruction	**Base Cost [pJ]**
beq	500
bne	500
bx	700
cmp	300
mul	900
sub	600

$O_{i,j}$ [pJ]	beq	bne	bx	cmp	mul	sub
beq	0	10	10	30	30	30
bne	10	0	10	30	30	30
bx	10	10	0	60	60	60
cmp	10	10	10	0	20	20
mul	10	10	10	30	0	30
sub	10	10	10	20	30	0

ISA-Level Energy Modelling

Components of an Energy Model:

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

Instruction	Base Cost [pJ]
beq	500
bne	500
bx	700
cmp	300
mul	900
sub	600

$O_{i,j}$ [pJ]	beq	bne	bx	cmp	mul	sub
beq	0	10	10	30	30	30
bne	10	0	10	30	30	30
bx	10	10	0	60	60	60
cmp	10	10	10	0	20	20
mul	10	10	10	30	0	30
sub	10	10	10	20	30	0

Based on V. Tiwari, S. Malik and A. Wolfe. “Instruction Level Power Analysis and Optimization of Software”, Journal of VLSI Signal Processing Systems, 13, pp 223-238, 1996.

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?

V. Tiwari, S. Malik and A. Wolfe. “Instruction Level Power Analysis and Optimization of Software”, Journal of VLSI Signal Processing Systems, 13, pp 223-238, 1996.

Exercise

```
@ Argument is in r0
fact:
        sub      r3, r0, #1
        cmp      r3, #0
        beq      .L2           @ Never iterate loop if num == 1
.L3:
        mul      r0, r3        @ Accumulate factorial value in r0
        sub      r3, r3, #1    @ r3 is decrementing counter
        cmp      r3, #0
        bne      .L3           @ Loop if we haven't reached 0
.L2:
        bx       lr            @ Return, answer is in r0
```

Which instruction sequence is being executed for a call to `fact(3)`?

Exercise

```
@ Argument is in r0
fact:
        sub     r3, r0, #1
        cmp     r3, #0
        beq     .L2           @ Never iterate loop if num == 1
.L3:
        mul     r0, r3        @ Accumulate factorial value in r0
        sub     r3, r3, #1    @ r3 is decrementing counter
        cmp     r3, #0
        bne     .L3           @ Loop if we haven't reached 0
.L2:
        bx      lr            @ Return, answer is in r0
```

A call to `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

Instruction	**Base Cost [pJ]**
beq	500
bne	500
bx	700
cmp	300
mul	900
sub	600

$O_{i,j}$ [pJ]	beq	bne	bx	cmp	mul	sub
beq	0	10	10	30	30	30
bne	10	0	10	30	30	30
bx	10	10	0	60	60	60
cmp	10	10	10	0	20	20
mul	10	10	10	30	0	30
sub	10	10	10	20	30	0

A call to `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

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

```
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
```

$$\begin{aligned} \boldsymbol{E}_{fact(3)} &= 3*600pJ + 3*300pJ + 500pJ + 2*900 + 2*500pJ + 700pJ \\ &+ 3*20pJ + 10pJ + 30pJ + 2*30pJ + 2*10pJ + 30pJ + 10pJ \\ &= 6920pJ \; \underline{\boldsymbol{= 6.92nJ}} \end{aligned}$$

Is it really this easy?

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

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

Instruction Base Cost, B_i, of each instruction i

Circuit State Overhead, $O_{i,j}$, for each instruction pair

Based on V. Tiwari, S. Malik and A. Wolfe. “Instruction Level Power Analysis and Optimization of Software”, Journal of VLSI Signal Processing Systems, 13, pp 223-238, 1996.

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

V. Tiwari, S. Malik and A. Wolfe. "Instruction Level Power Analysis and Optimization of Software", Journal of VLSI Signal Processing Systems, 13, pp 223-238, 1996.

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)

V. Tiwari, S. Malik and A. Wolfe. “Instruction Level Power Analysis and Optimization of Software”, Journal of VLSI Signal Processing Systems, 13, pp 223-238, 1996.

XCore Energy Modelling

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

$$E_{\mathrm{p}} = P_{\mathrm{base}} N_{\mathrm{idle}} T_{\mathrm{clk}} + \sum_{t=1}^{N_t} \sum_{i \in \mathrm{ISA}} \left(\left(M_t P_i O + P_{\mathrm{base}} \right) N_{i,t} T_{\mathrm{clk}} \right)$$

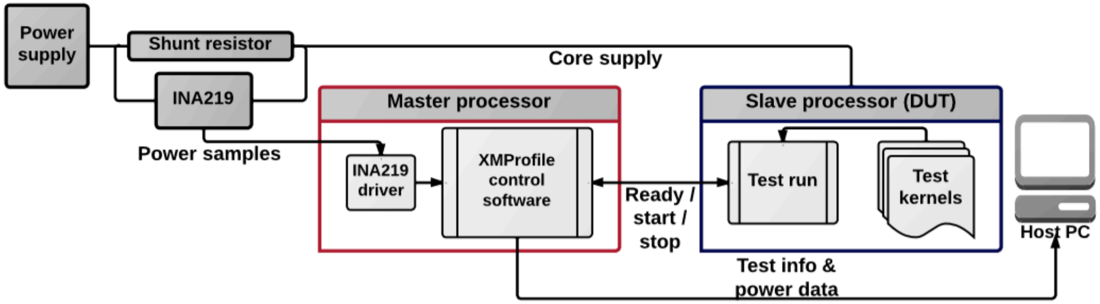
Idle base power and duration

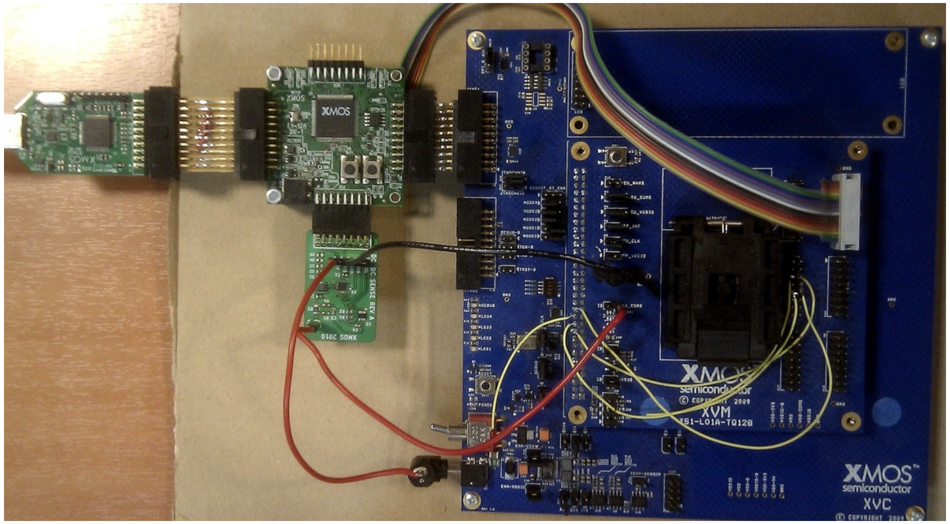
Concurrency cost, instruction cost, generalised overhead, base power and duration

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

S. Kerrison and K. Eder. 2015. “Energy Modeling of Software for a Hardware Multithreaded Embedded Microprocessor”. ACM Trans. Embed. Comput. Syst. 14, 3, Article 56 (April 2015), 25 pages. DOI=10.1145/2700104 http://doi.acm.org/10.1145/2700104

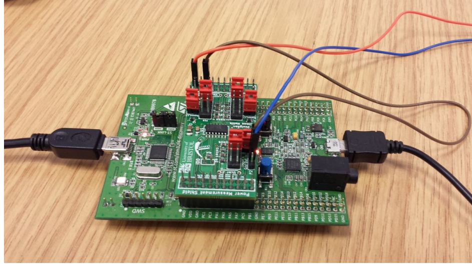
The set up...

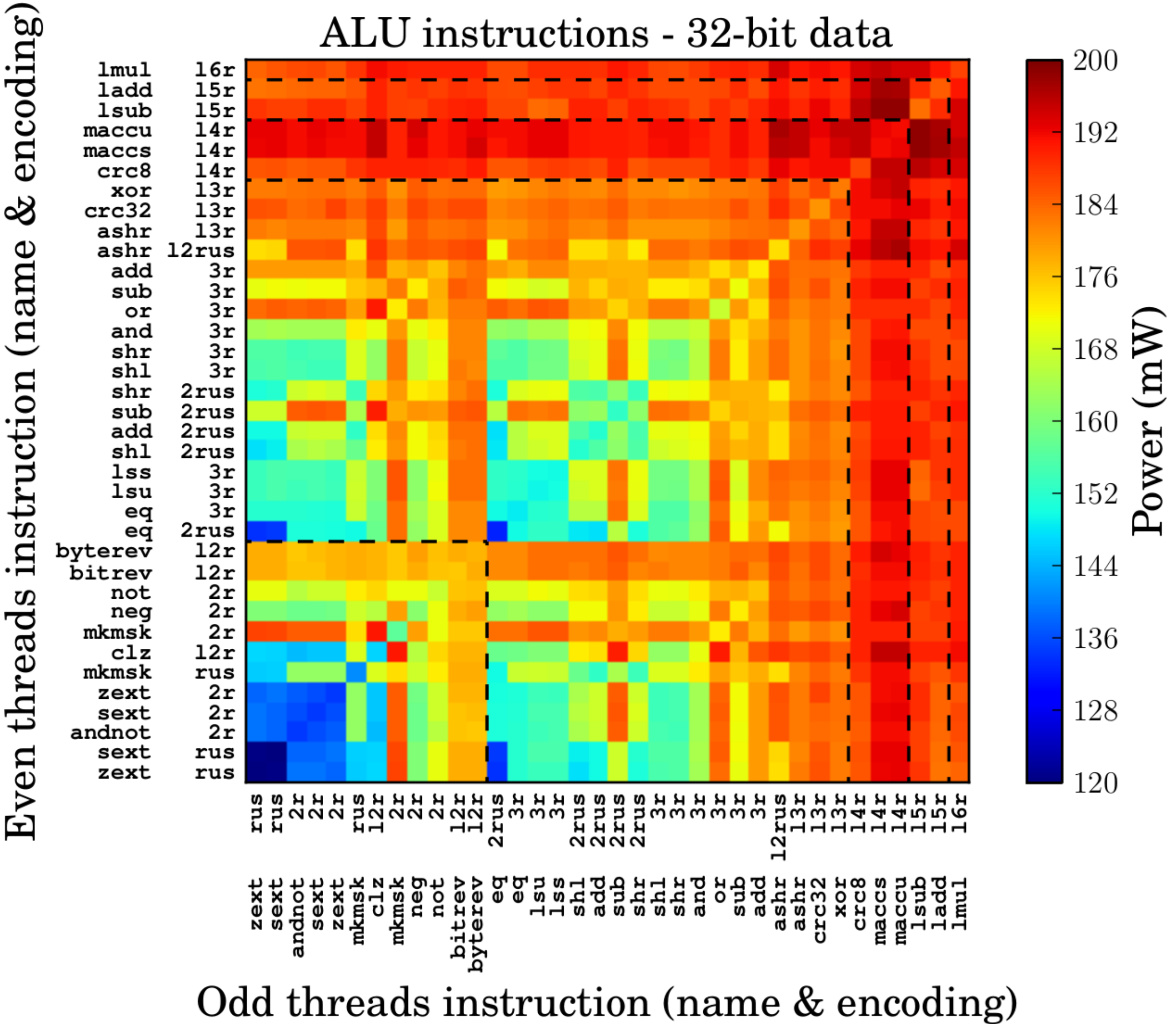




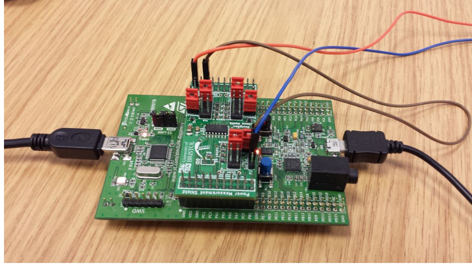
S. Kerrison and K. Eder. 2015. “Energy Modeling of Software for a Hardware Multithreaded Embedded Microprocessor”. ACM Trans. Embed. Comput. Syst. 14, 3, Article 56 (April 2015), 25 pages. DOI=10.1145/2700104 http://doi.acm.org/10.1145/2700104

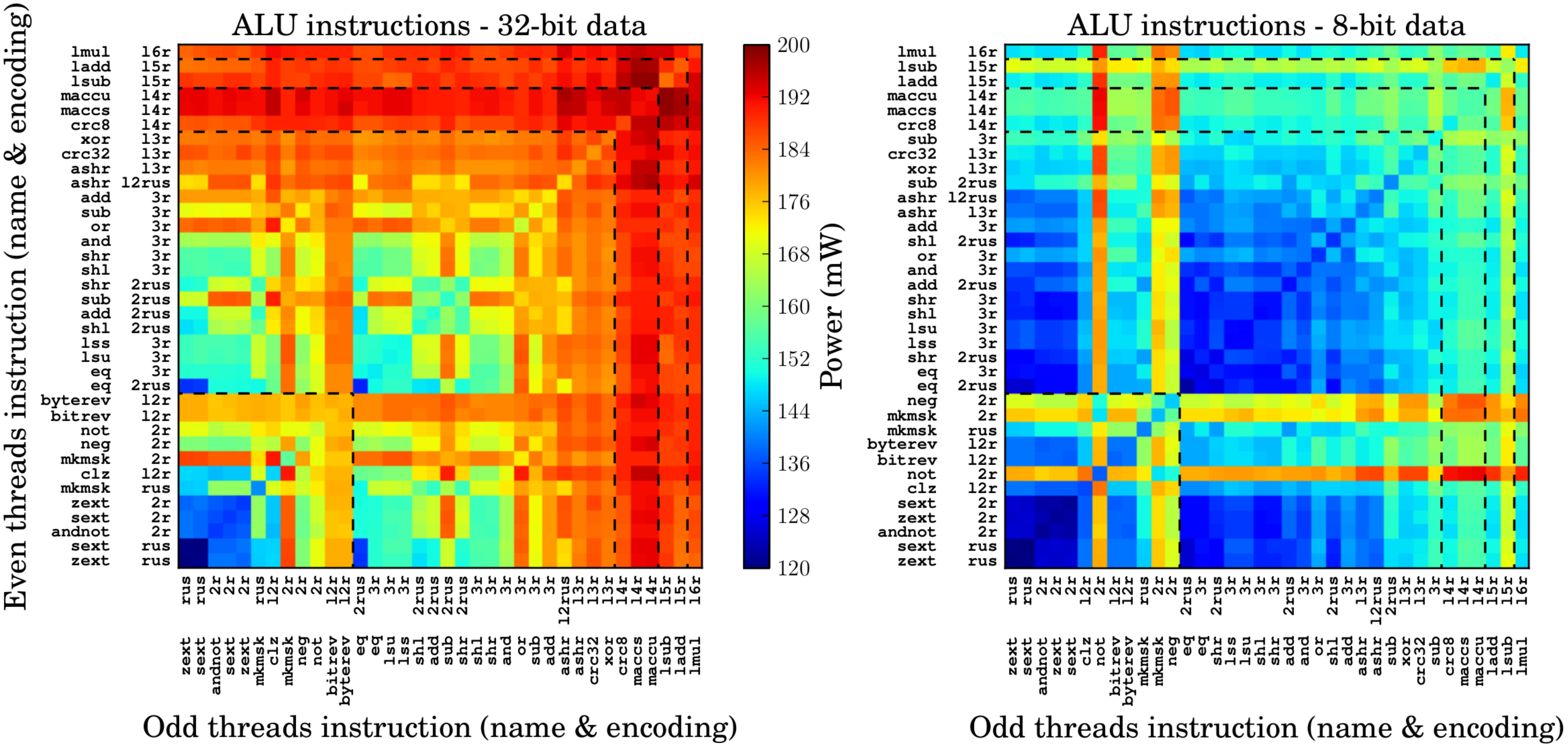
ISA Characterization





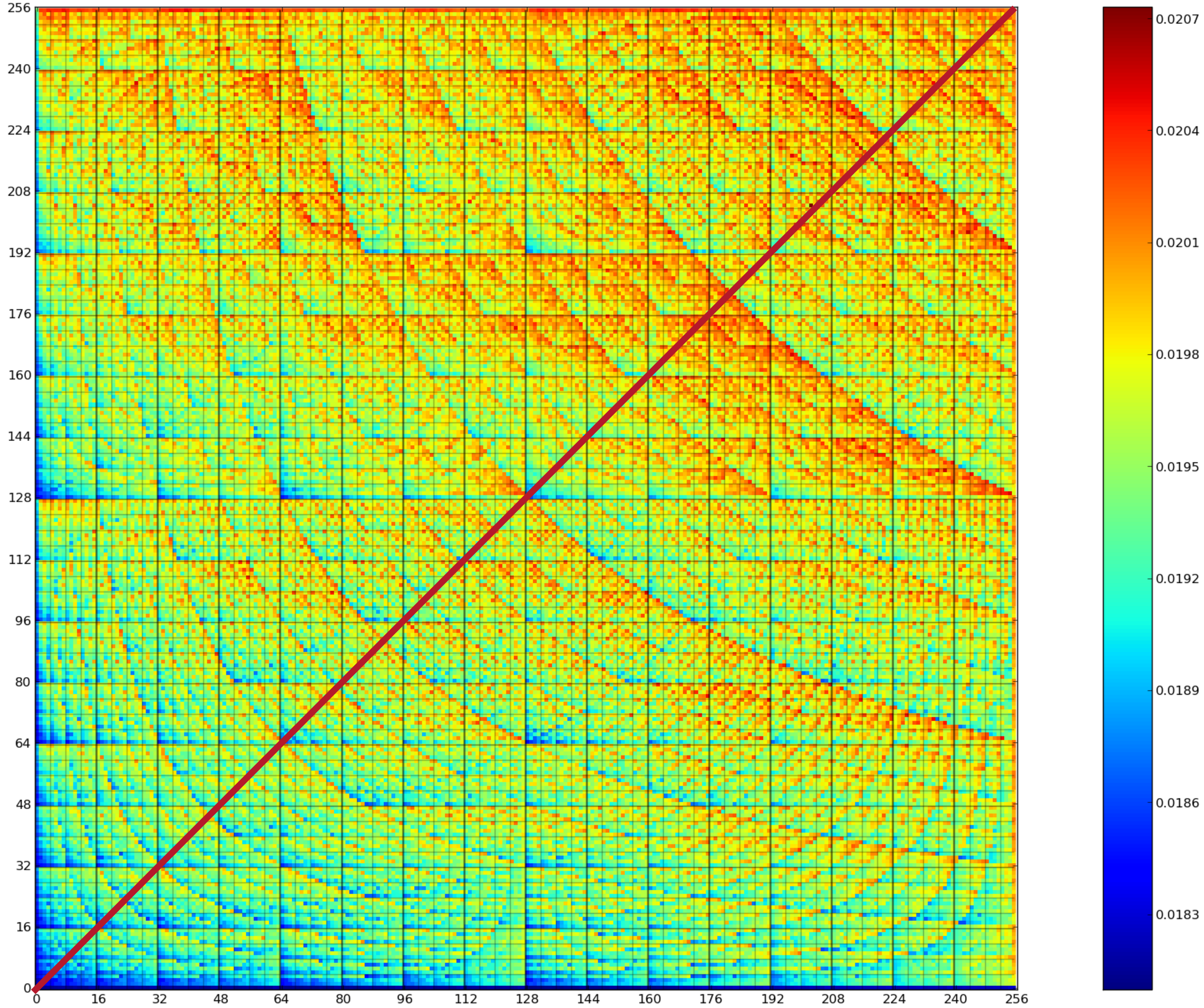
ISA Characterization





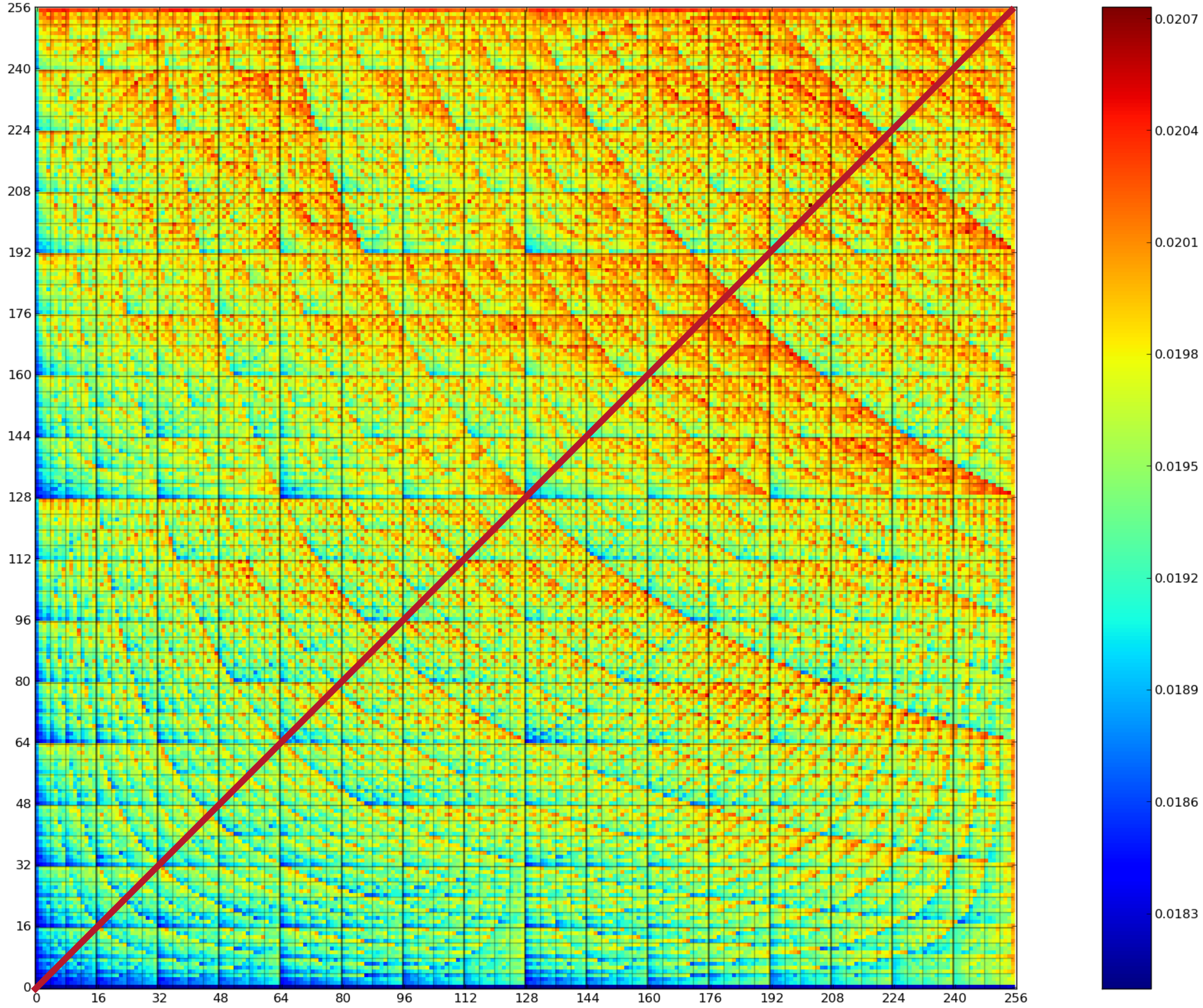
S. Kerrison and K. Eder. 2015. "Energy Modeling of Software for a Hardware Multithreaded Embedded Microprocessor". ACM Trans. Embed. Comput. Syst. 14, 3, Article 56 (April 2015), 25 pages. DOI=10.1145/2700104 http://doi.acm.org/10.1145/2700104

a*b = b*a

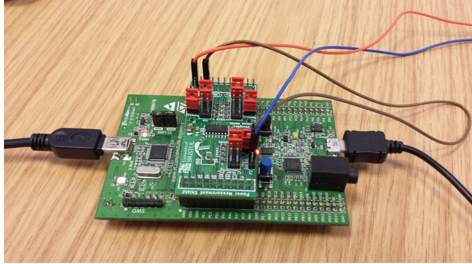


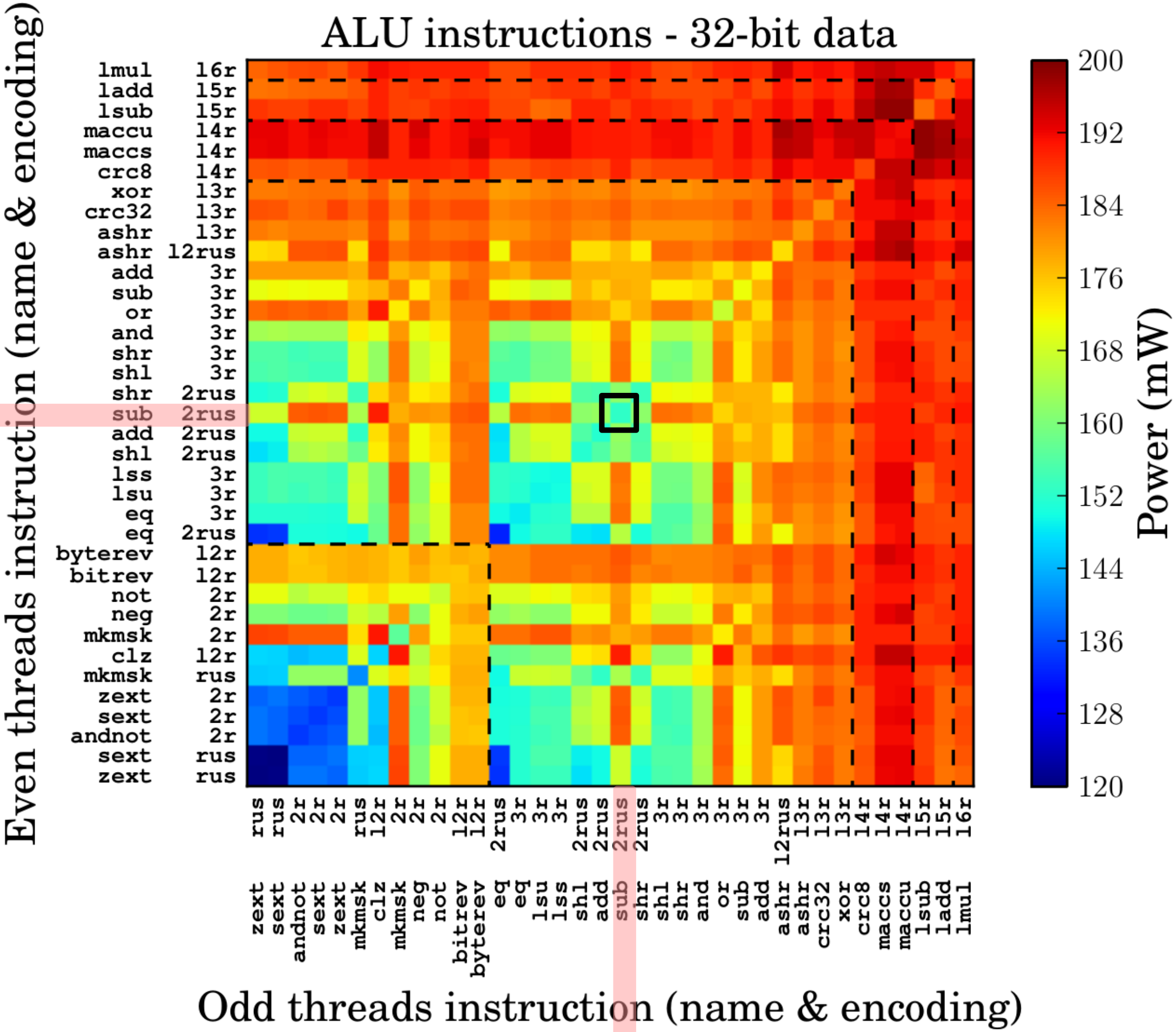
0
16
32
48
64
80
96
112
128
144
160
176
192
208
224
240
256
0.0207
0.0204
0.0201
0.0198
0.0195
0.0192
0.0189
0.0186
0.0183

Energy(a*b) ≠ Energy(b*a)

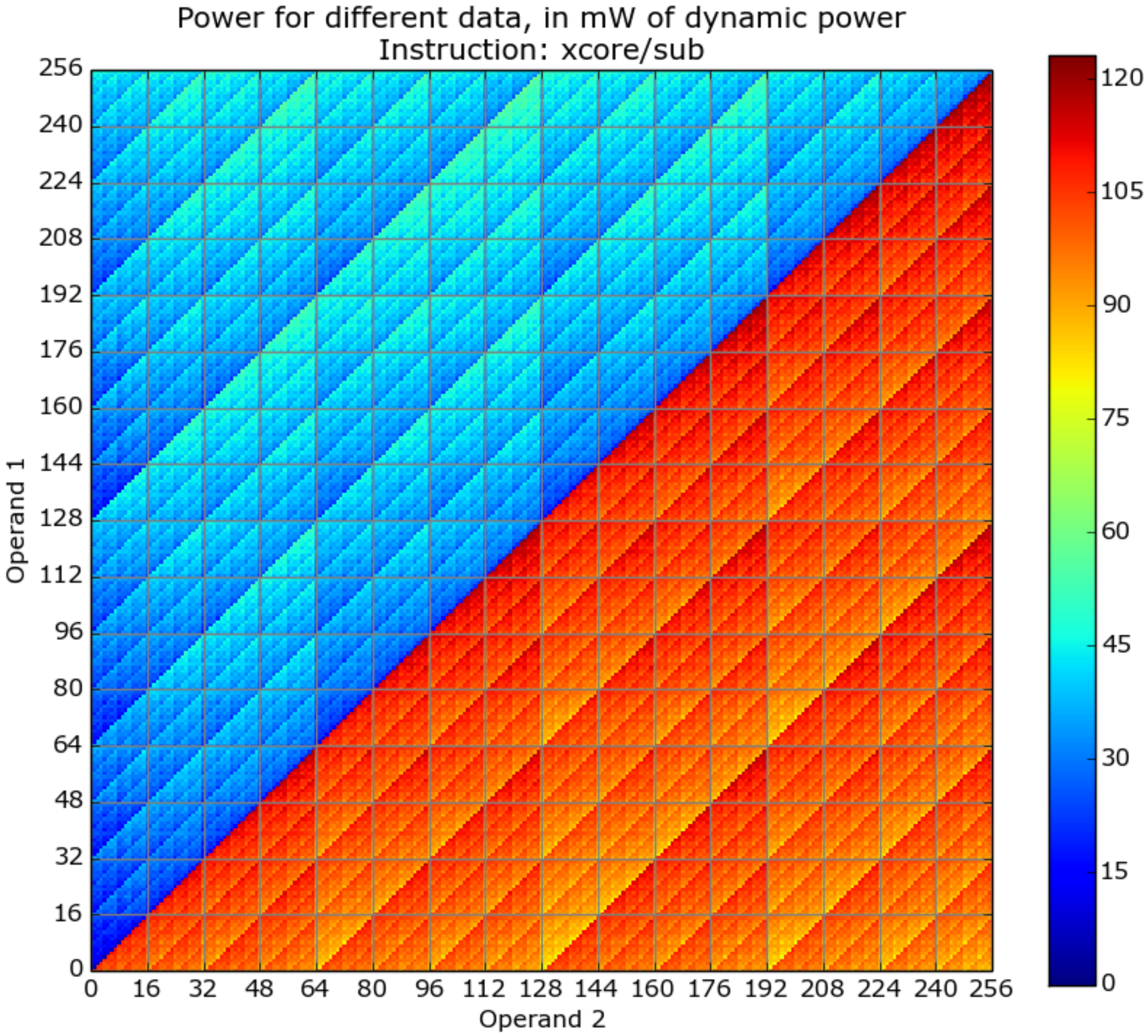


ISA Characterization

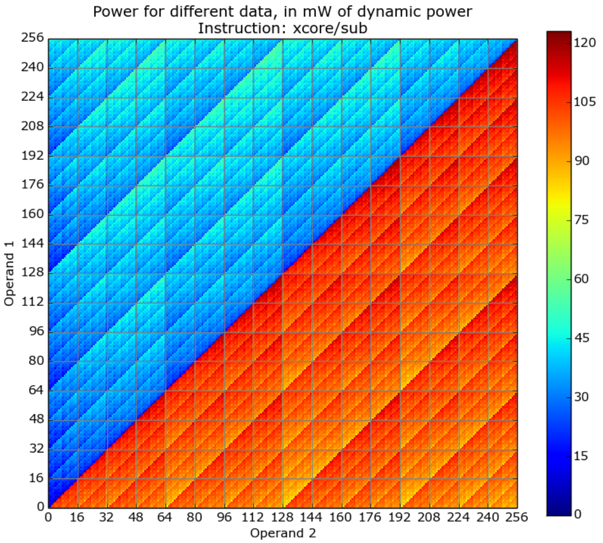


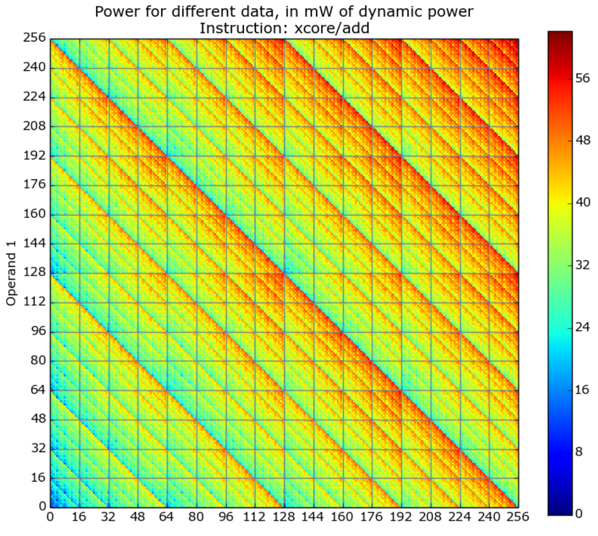


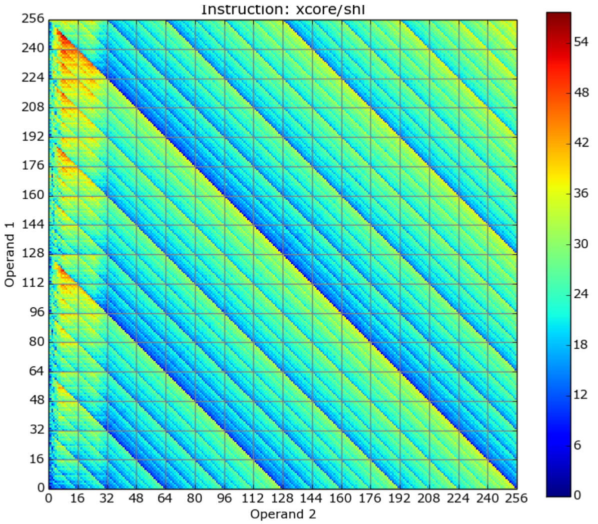
The Impact of Data on Energy Consumption

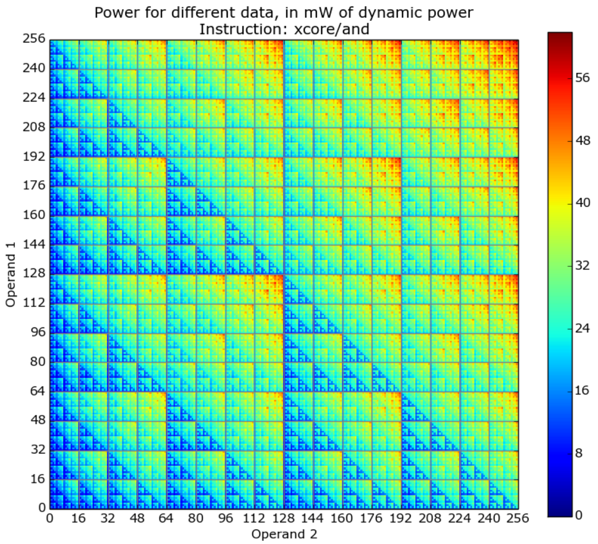


W/A/B-Case Energy Consumption


Power for different data, in mW of dynamic power
Instruction: xcore/sub


Power for different data, in mW of dynamic power
Instruction: xcore/add


Instruction: xcore/shl

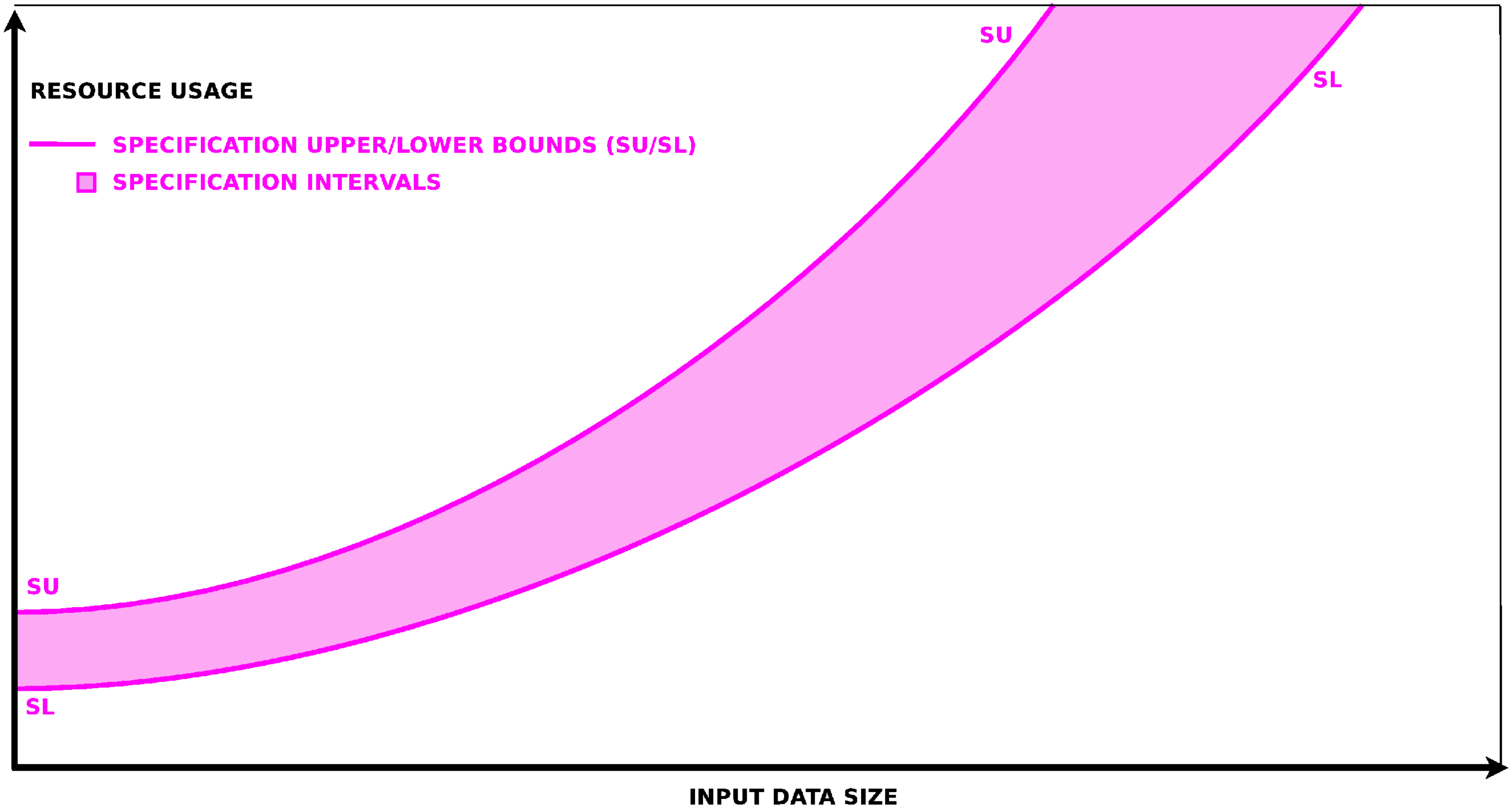

Power for different data, in mW of dynamic power
Instruction: xcore/and

A quick jump forward to Static Resource consumption Analysis

Static Resource Analysis

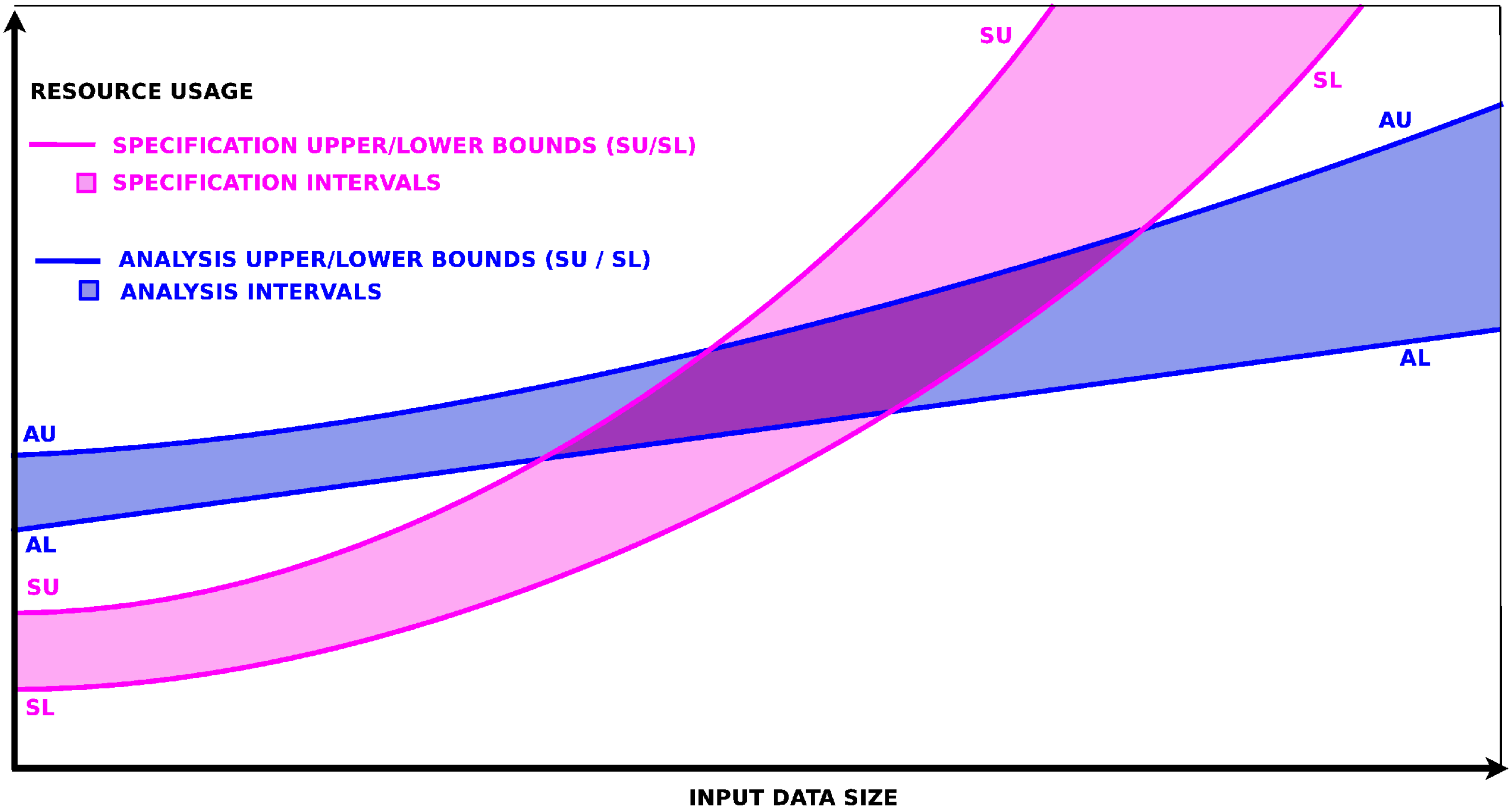
- Techniques automatically infer upper and lower bounds on resource usage of a program.
- Bounds expressed using monotonic arithmetic functions per procedure parameterized by program's input size.
- Verification can be done statically by checking that the upper and lower bounds on resource usage defined in the specifications hold.

Specified Resource Usage



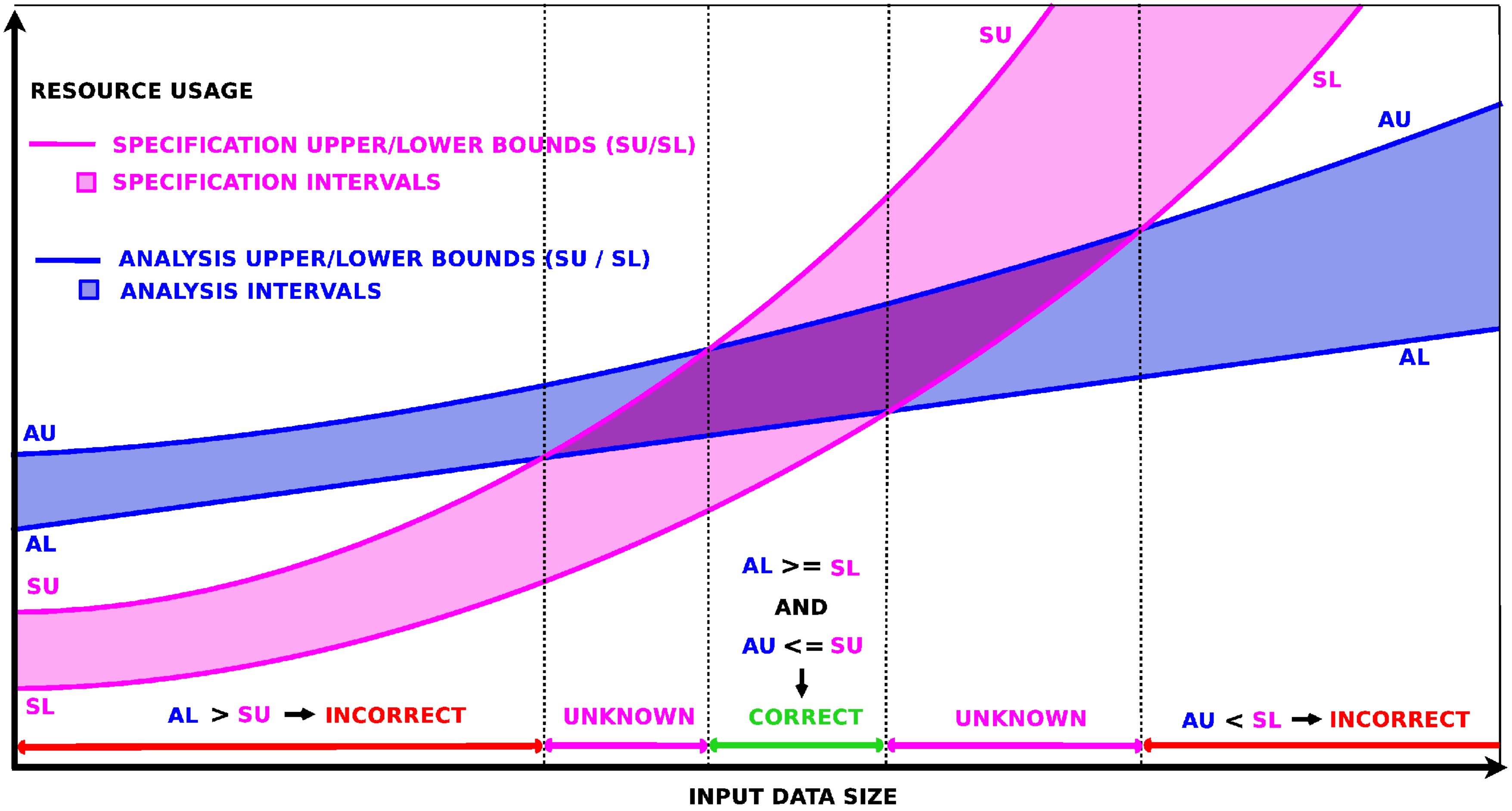
Source: Pedro Lopez Garcia, IMDEA Software Research Institute

Analysis Result



Source: Pedro Lopez Garcia, IMDEA Software Research Institute

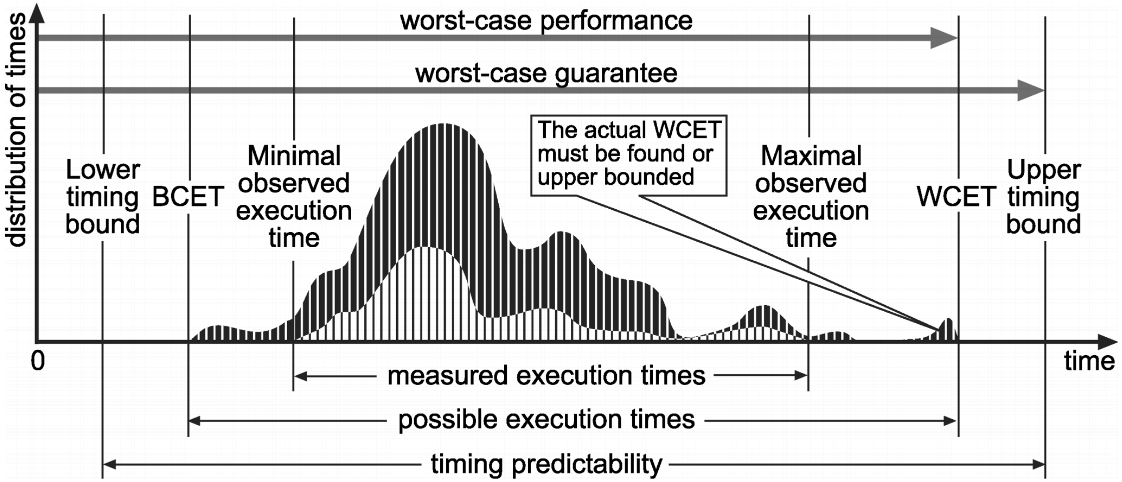
Verification



Source: Pedro Lopez Garcia, IMDEA Software Research Institute

Worst Case Execution Time

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



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

Does this work for energy consumption analysis?

Worst Case Energy Consumption

- WCEC analysis goes well beyond WCET analysis.
  - data independence of execution time through the use of synchronous logic
  - embedded real-time systems that are timing predictable execute instructions in a fixed number of clock cycles
  - WCET then depends only on the WC execution path
- Energy consumption is data dependent.
  - Data dependent energy modelling

Data Dependent Energy Modeling for Worst Case Energy Consumption Analysis

James Pallister, Steve Kerrison, Jeremy Morse and Kerstin Eder
Dept. Computer Science, Merchant Venturers Building,
Bristol, BS8 1UB. Email: firstname.lastname@bristol.ac.uk

***Abstract*—This paper examines the impact of operand values upon instruction level energy models of embedded processors, to explore whether the requirements for safe worst case energy consumption (WCEC) analysis can be met. WCEC is similar to worst case execution time (WCET) analysis, but seeks to determine whether a task can be completed within an energy budget rather than within a deadline. Existing energy models that underpin such analysis typically use energy measurements from random input data, providing average or otherwise unbounded estimates not necessarily suitable for worst case analysis.**

We examine energy consumption distributions of two benchmarks under a range of input data on two cache-less embedded architectures, AVR and XS1-L. We find that the worst case can be predicted with a distribution created from random data. We propose a model to obtain energy distributions for instruction sequences that can be composed, enabling WCEC analysis on program basic blocks. Data dependency between instructions is also examined, giving a case where dependencies create a bimodal energy distribution. The worst case energy prediction remains safe. We conclude that worst-case energy models based on a probabilistic approach are suitable for safe WCEC analysis.

I. 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 energy budget. However, such analysis often uses energy models that do not explicitly consider the dynamic power drawn by switching of data, instead producing an upper-bound using averaged random or scaled instruction models [1], [2].

A safe and tightly bound model for WCEC analysis must be close to the hardware's actual behavior, but also give confidence that it never under-estimates. Current models have not been analyzed in this context to provide sufficient confidence, and power figures from manufacturer datasheets are not sufficiently detailed to provide tight bounds.

Energy modeling allows the energy consumption of software to be estimated without taking physical measurements. Models may assign an energy value to each instruction [3], to a predefined set of processor modes [4], or use a detailed approach that considers wider processor state, such as the data for each instruction [5]. Although measurements are typically more accurate, models require no hardware instrumentation, are more versatile and can be used in many situations, such as

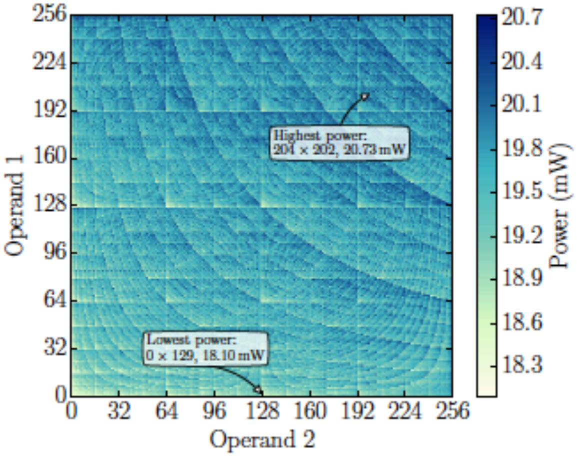


Fig. 1. Power map of `mul` instruction, total range is 15 % of SoC power.

In this paper, we find 15 % difference in a simple 8-bit AVR processor. This device has no caches, no OS and no high power peripherals. This difference can be seen in Figure 1, which shows the power for a single cycle, 8-bit multiply instruction in this processor. The diagram was constructed by taking hardware measurements for every possible eight bit input.

Accounting for data dependent effects in an energy model is a challenging task, which we split into two parts. Firstly, the energy effect of an instruction's manipulation of processor state needs to be modeled. This is an infeasible amount of data to exhaustively collect. A 32-bit three-operand instruction has 2^{96} possible data value combinations.

Secondly, a technique is required to derive the energy consumption for a sequence of instructions from such a model. The composition of data dependent instruction energy models is a particularly difficult task. The data causing maximum energy consumption for one instruction may minimize the cost in a subsequent, dependent instruction. Finding the greatest cost for such sequences requires searching for inputs that maximize a property after an arbitrary computation, which is again an infeasibly large task. Over-approximating by summing the worst possible data dependent energy consumption of each instruction in a sequence, regardless of whether such a computation can occur, would lead to a significant overestimation of

Worst Case Energy Consumption

- WCEC analysis goes well beyond WCET analysis.
  - data independence of execution time through the use of synchronous logic
  - embedded real-time systems that are timing predictable execute instructions in a fixed number of clock cycles
  - WCET then depends only on the WC execution path
- Energy consumption is data dependent.
  - Data dependent energy modelling
  - Critical questions:
    - *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?*

On the infeasibility of analysing worst-case dynamic energy

Jeremy Morse, Steve Kerrison and Kerstin Eder
University of Bristol

March 9, 2016

Abstract

In this paper we study the sources of dynamic energy during the execution of software on microprocessors suited for the Internet of Things (IoT) domain. Estimating the energy consumed by executing software is typically achieved by determining the most costly path through the program according to some energy model of the processor. Few models, however, adequately tackle the matter of dynamic energy caused by operand data. We find that the contribution of operand data to overall energy can be significant, prove that finding the worst-case input data is NP-hard, and further, that it cannot be estimated to any useful factor. Our work shows that accurate worst-case analysis of data dependent energy is infeasible, and that other techniques for energy estimation should be considered.

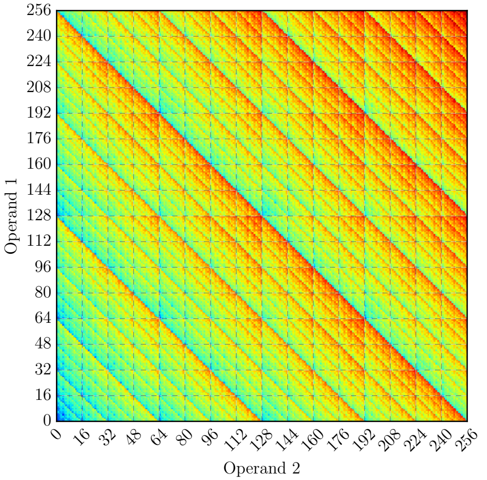
1 Introduction

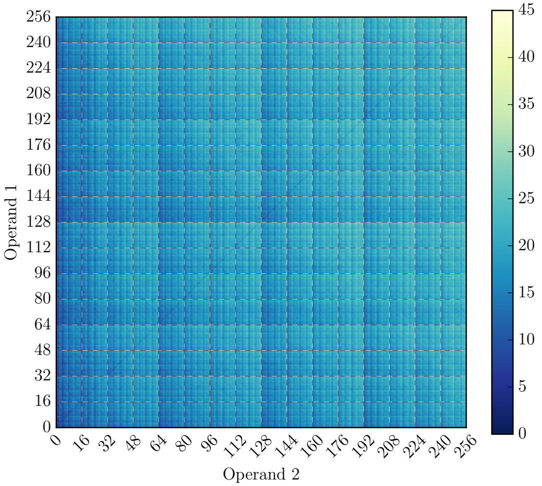
A significant design constraint in the development of embedded systems is that of resource consumption. Software executing on such systems typically has very limited memory and computing power available, and yet must meet the requirements of the system. To aid the design process, analysis tools such as profilers or maximum-stack-depth estimators provide the developer with information allowing them to refine their designs and satisfy constraints.

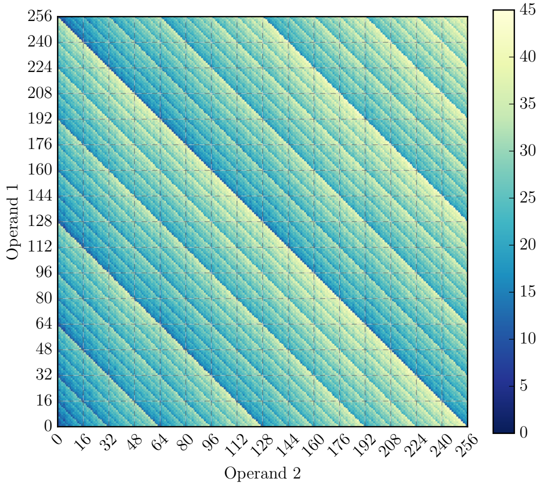
A less well studied constraint is the limited energy budgets that deeply embedded systems possess. A typical example would be a wireless sensor powered by battery, that must operate for a minimum period without the battery being replaced. Other examples would be systems dependent on energy harvesting, or systems with low thermal design points that thus have a maximum power dissipation level. These constraints can also be approached with software analysis tools, and several techniques have been developed that allow the estimation of software's energy consumption [17, 7, 18].

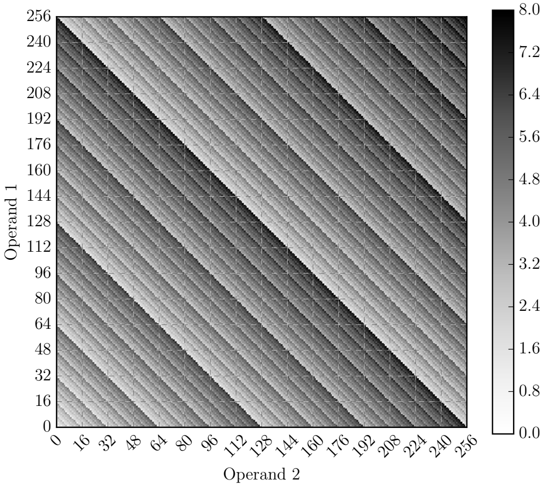
Within energy estimation, focus has been given to *Worst Case* Energy Consumption (WCEC): determining the maximum amount of energy that can be consumed during the execution of the software. In this paper, we shall study the calculation of worst case energy, considering only the effects that different software and inputs can have on a system. The objective is to determine

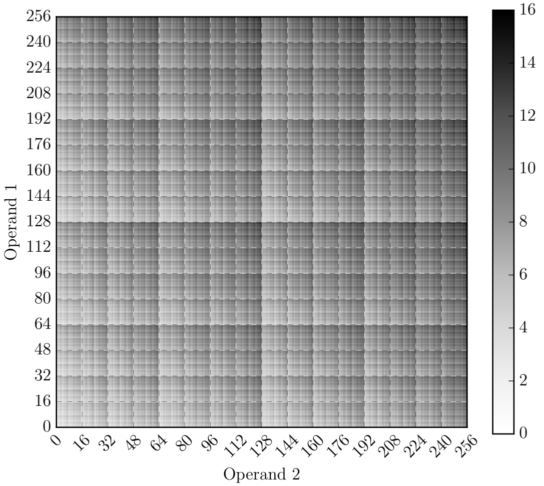
Impact of datapath switching

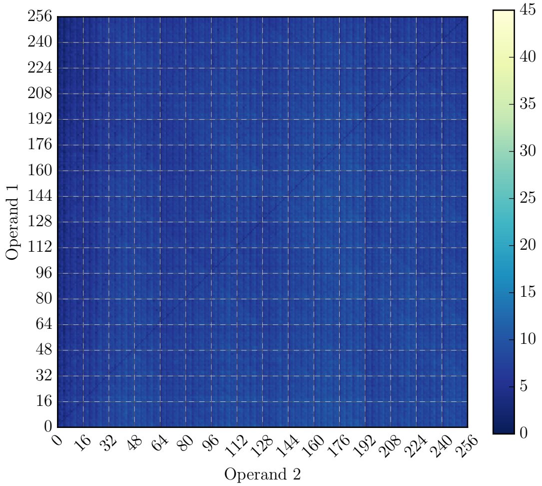












J. Morse, S. Kerrison and K. Eder. 2016. “On the infeasibility of analysing worst case dynamic energy”. (under review) http://arxiv.org/abs/1603.02580

Energy Consumption Analysis
enables energy transparency

www.theguardian.com

Energy Consumption Analysis enables energy transparency

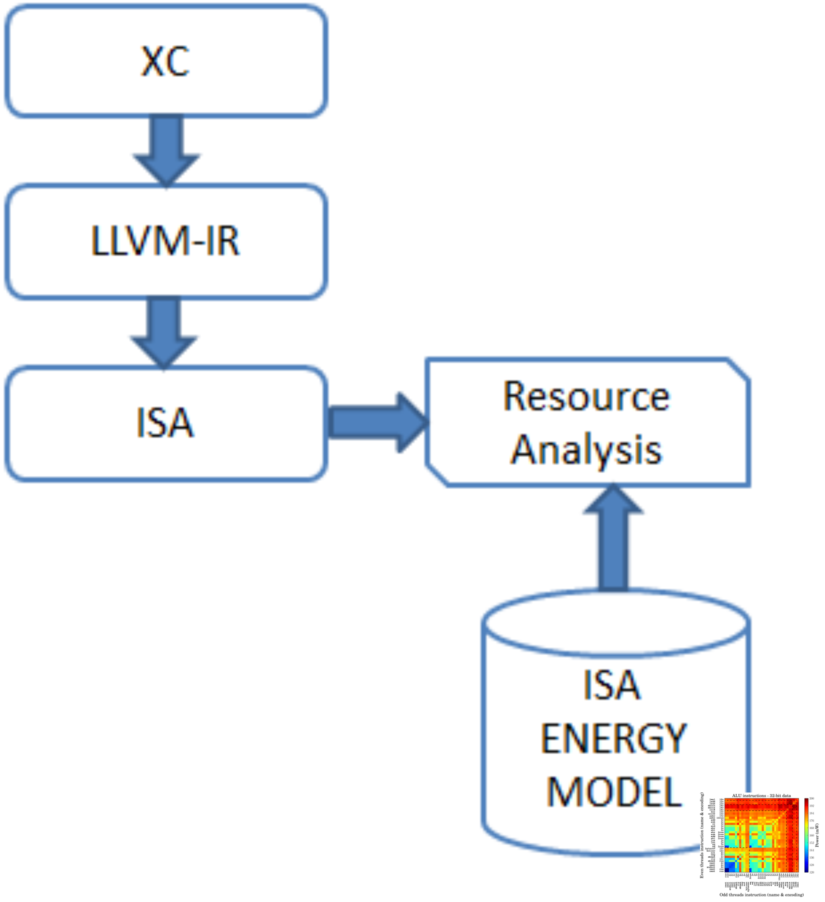
www.theguardian.com

http://us.123rf.com/450wm/kentoh/kentoh1006/kentoh100600301/7129999-futuristic-technology-data-flow-as-art-background.jpg

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:

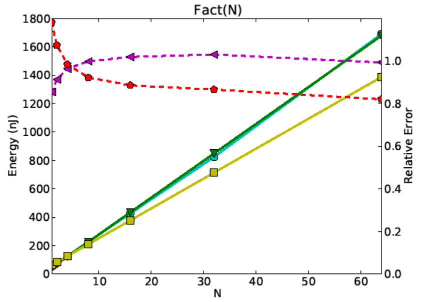
```
int fact (int x) {
  if (x<=0)^a
     return 1^b;
  return (x *^d fact(x-1))^c;
}
```

Extracted Cost Relations:

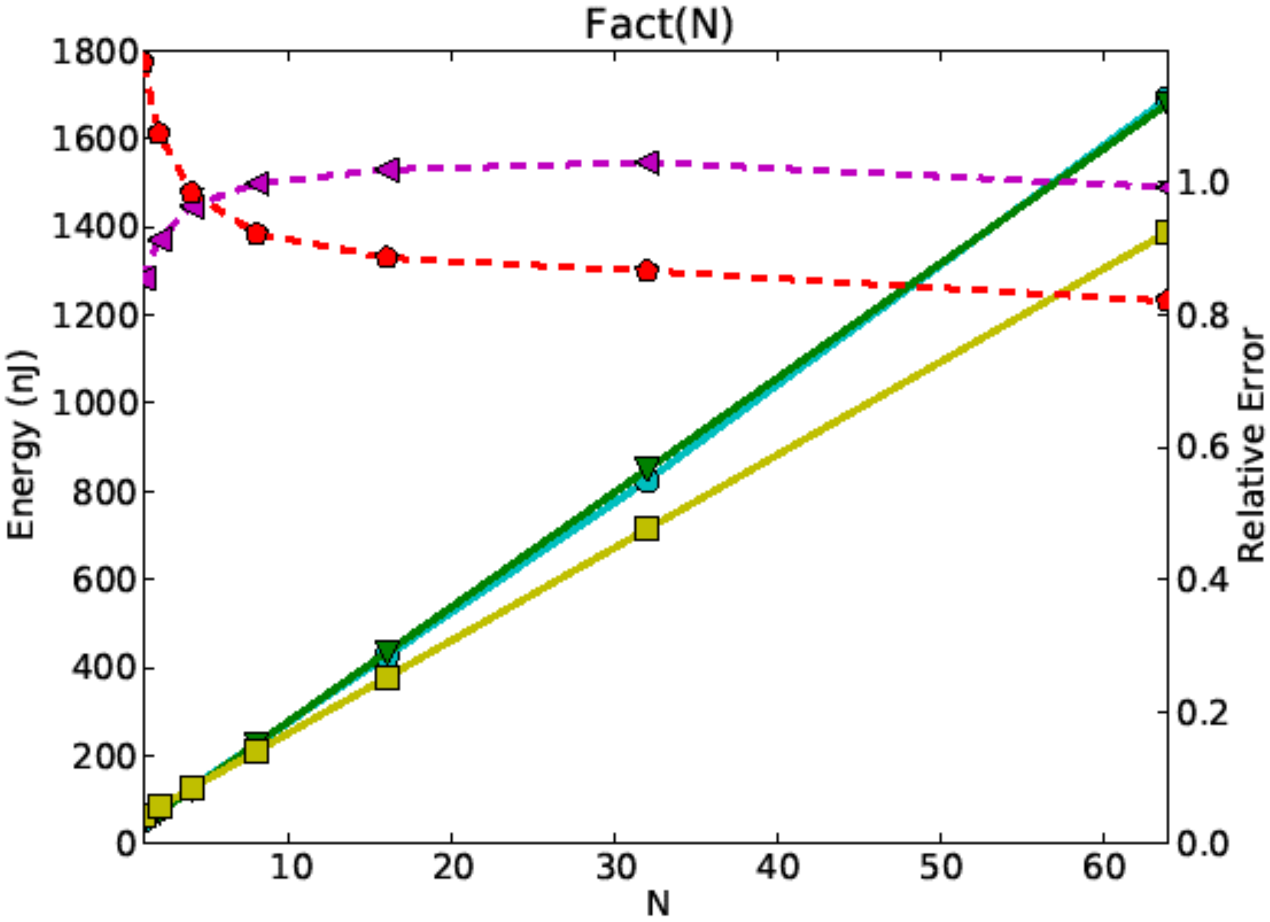
$$
\begin{array}{llll}
C_{fact}(x) & = C_a + C_b & & \text{if } x<=0 \\
C_{fact}(x) & = C_a + C_c(x) & & \text{if } x>0 \\
C_c(x) & = C_d + C_{fact}(x-1) & &
\end{array}
$$

- 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: $C_{fact}(x) = (26x + 19.4)$ nJ

(Note: The above result is based on the XMOS XCore Energy model introduced earlier. It is not using the energy model from the Exercise.)

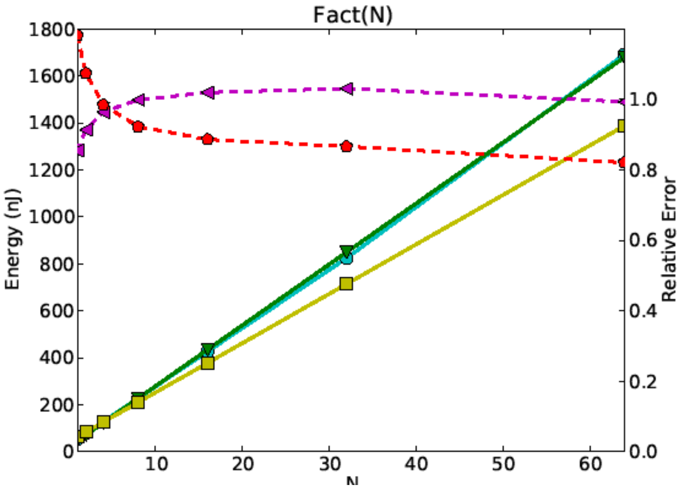


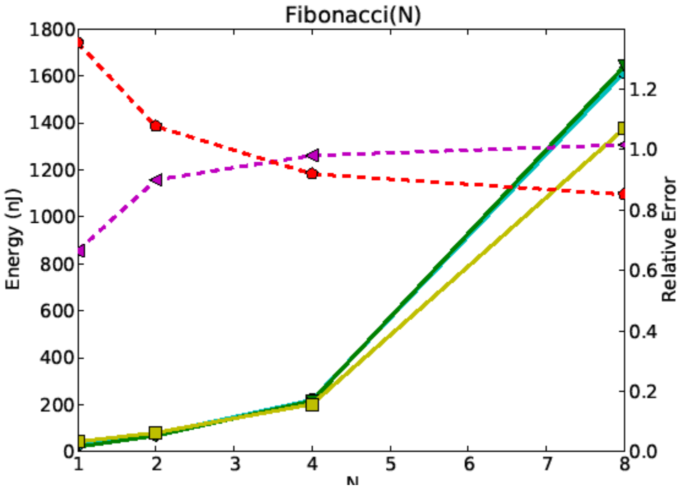
ISA-Level Analysis Results

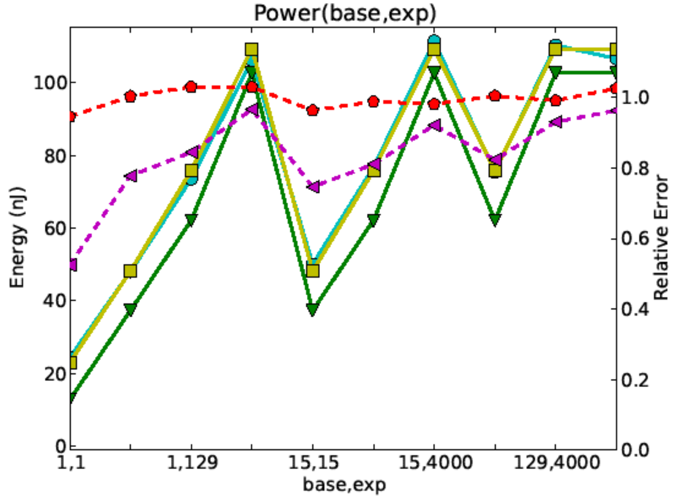


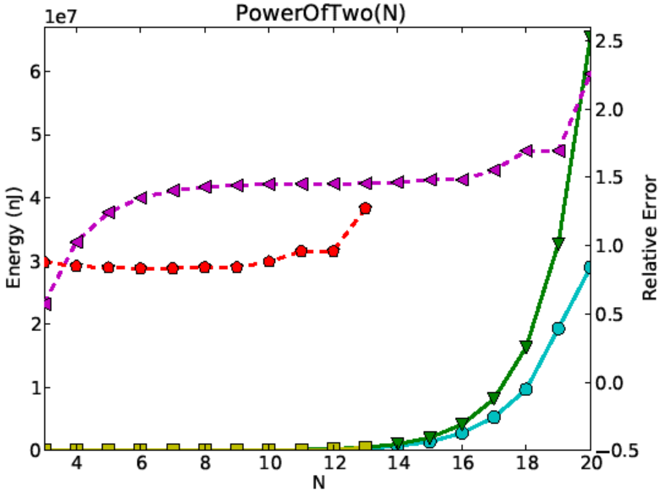
U. Liqat, S. Kerrison, A. Serrano, K. Georgiou, N. Grech, P. Lopez-Garcia, M.V. Hermenegildo and K. Eder. "Energy Consumption Analysis of Programs based on XMOS ISA-Level Models". LOPSTR 2013.

ISA-Level Analysis Results



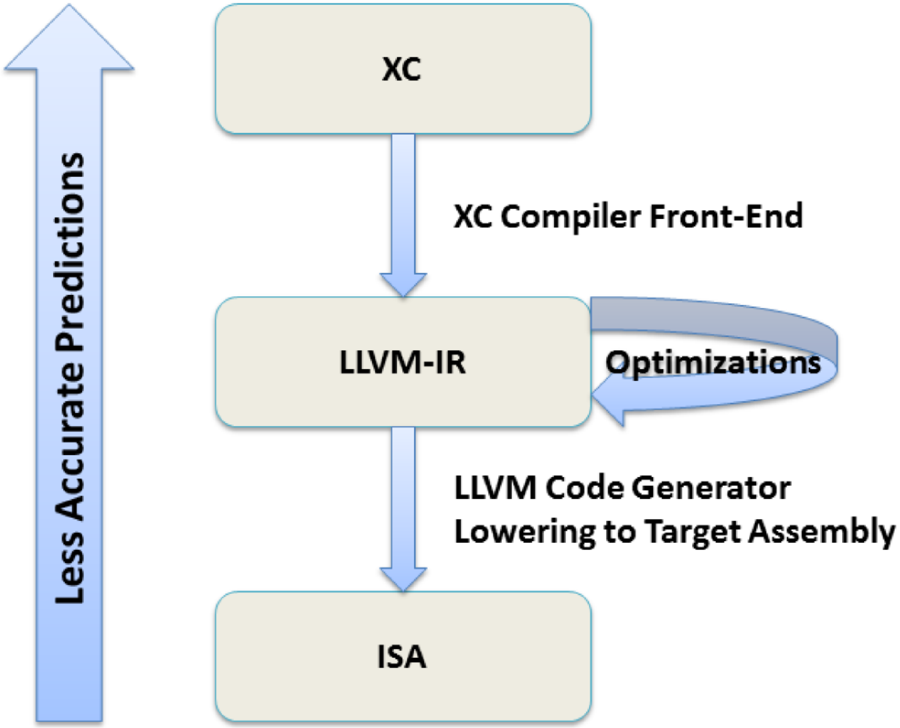






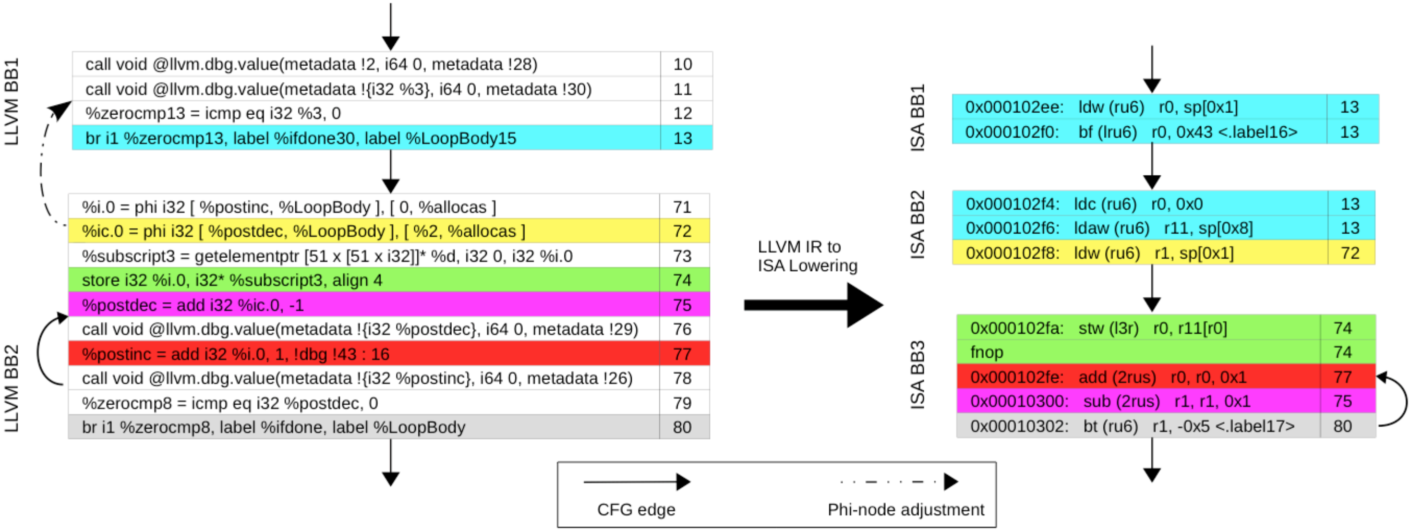
U. Liqat, S. Kerrison, A. Serrano, K. Georgiou, N. Grech, P. Lopez-Garcia, M.V. Hermenegildo and K. Eder. "Energy Consumption Analysis of Programs based on XMOS ISA-Level Models". LOPSTR 2013.

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



LLVM BB1

call void @llvm.dbg.value(metadata !2, i64 0, metadata !28)	10
call void @llvm.dbg.value(metadata !{i32 %3}, i64 0, metadata !30)	11
%zerocmp13 = icmp eq i32 %3, 0	12
br i1 %zerocmp13, label %ifdone30, label %LoopBody15	13

LLVM BB2

%i.0 = phi i32 [%postinc, %LoopBody], [0, %allocas]	71
%ic.0 = phi i32 [%postdec, %LoopBody], [%2, %allocas]	72
%subscript3 = getelementptr [51 x [51 x i32]]* %d, i32 0, i32 %i.0	73
store i32 %i.0, i32* %subscript3, align 4	74
%postdec = add i32 %ic.0, -1	75
call void @llvm.dbg.value(metadata !{i32 %postdec}, i64 0, metadata !29)	76
%postinc = add i32 %i.0, 1, !dbg !43 : 16	77
call void @llvm.dbg.value(metadata !{i32 %postinc}, i64 0, metadata !26)	78
%zerocmp8 = icmp eq i32 %postdec, 0	79
br i1 %zerocmp8, label %ifdone, label %LoopBody	80

LLVM IR to
ISA Lowering

ISA BB1

0x000102ee: ldw (ru6) r0, sp[0x1]	13
0x000102f0: bf (lru6) r0, 0x43 <.label16>	13

ISA BB2

0x000102f4: ldc (ru6) r0, 0x0	13
0x000102f6: ldaw (ru6) r11, sp[0x8]	13
0x000102f8: ldw (ru6) r1, sp[0x1]	72

ISA BB3

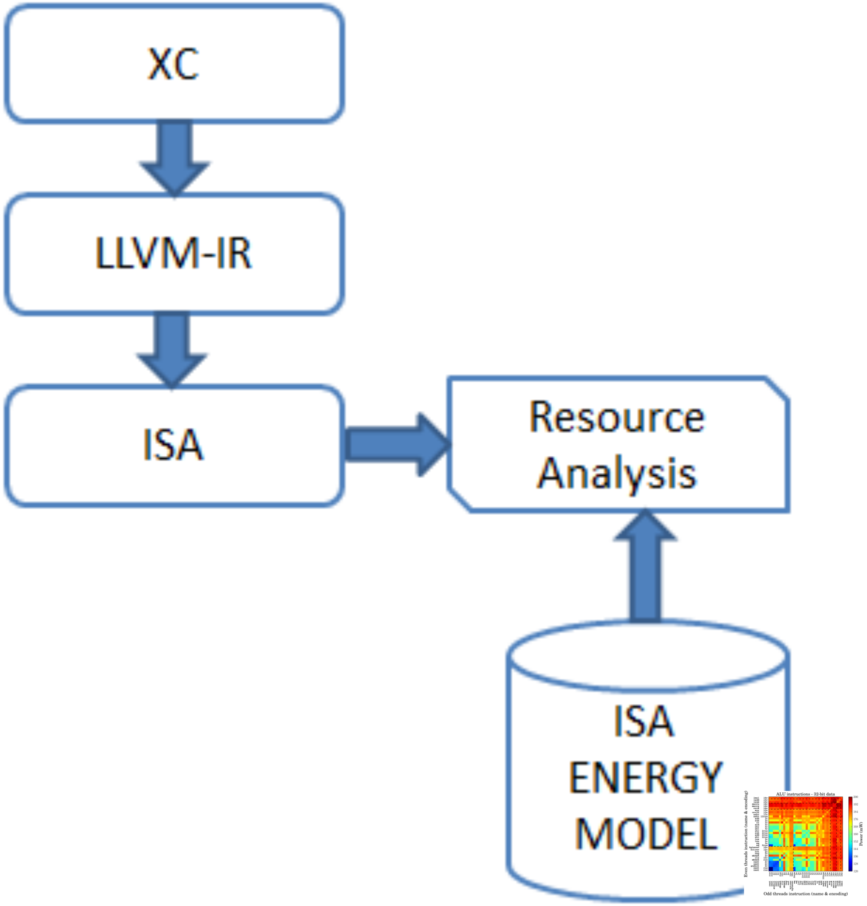
0x000102fa: stw (l3r) r0, r11[r0]	74
fnop	74
0x000102fe: add (2rus) r0, r0, 0x1	77
0x00010300: sub (2rus) r1, r1, 0x1	75
0x00010302: bt (ru6) r1, -0x5 <.label17>	80

$$E(ir_i) = \sum_{isa_j \in S} E(isa_j)$$

K. Georgiou, S. Kerrison and K. Eder, Oct 2015. “On the Value and Limits of Multi-level Energy Consumption Static Analysis for Deeply Embedded Single and Multi-threaded Programs”. http://arxiv.org/abs/1510.07095

U. Liqat, K. Georgiou, S. Kerrison, P. Lopez-Garcia, J.P. Gallagher, M.V. Hermenegildo, K. Eder. Inferring Parametric Energy Consumption Functions at Different Software Levels: ISA vs. LLVM IR. In Proceedings of FOPARA 2015. http://arxiv.org/abs/1511.01413

Analysis at the LLVM-IR Level

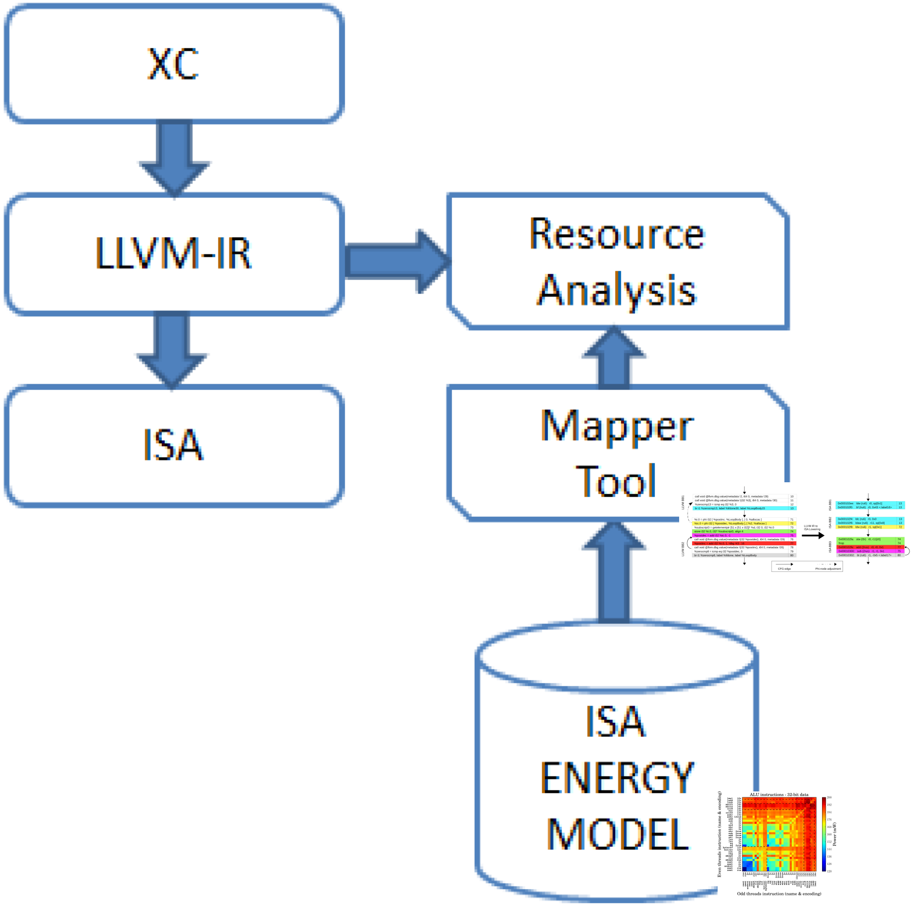

XC

LLVM-IR

ISA

Resource Analysis

ISA ENERGY MODEL


XC

LLVM-IR

ISA

Resource Analysis

Mapper Tool

ISA ENERGY MODEL

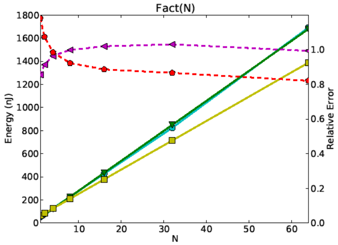
N. Grech, K. Georgiou, J. Pallister, S. Kerrison, J. Morse, K. Eder. 2015. Static analysis of energy consumption for LLVM IR programs. In Proceedings of the 18th International Workshop on Software and Compilers for Embedded Systems (SCOPES '15). ACM, New York, NY, USA, pages 12-21. http://dx.doi.org/10.1145/2764967.2764974

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

Towards Energy Aware Software Engineering

Energy Transparency

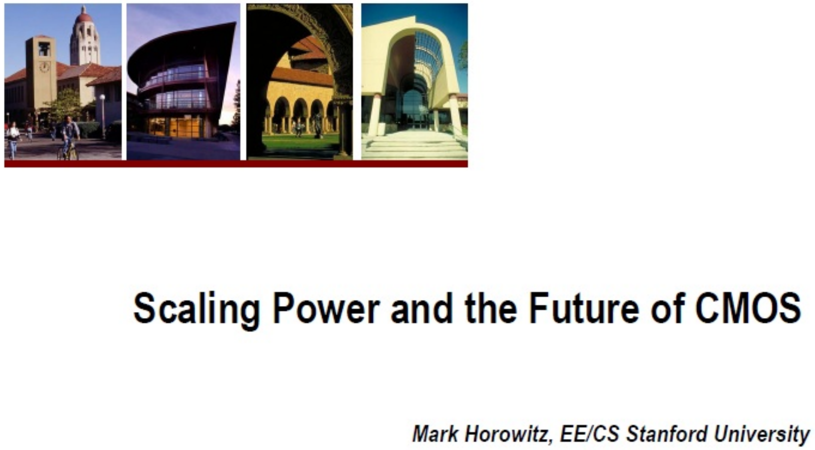


- For HW designers:
  “Power is a 1st and last order design constraint.”

  [Dan Hutcheson, VLSI Research, Inc., E^{3}S Keynote 2011]

- “Every design is a point in a 2D plane.”

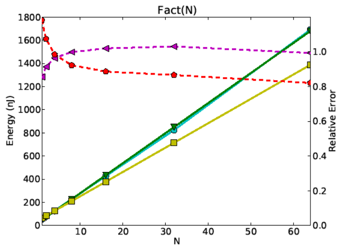
  [Mark Horowitz,E^{3}S 2009]



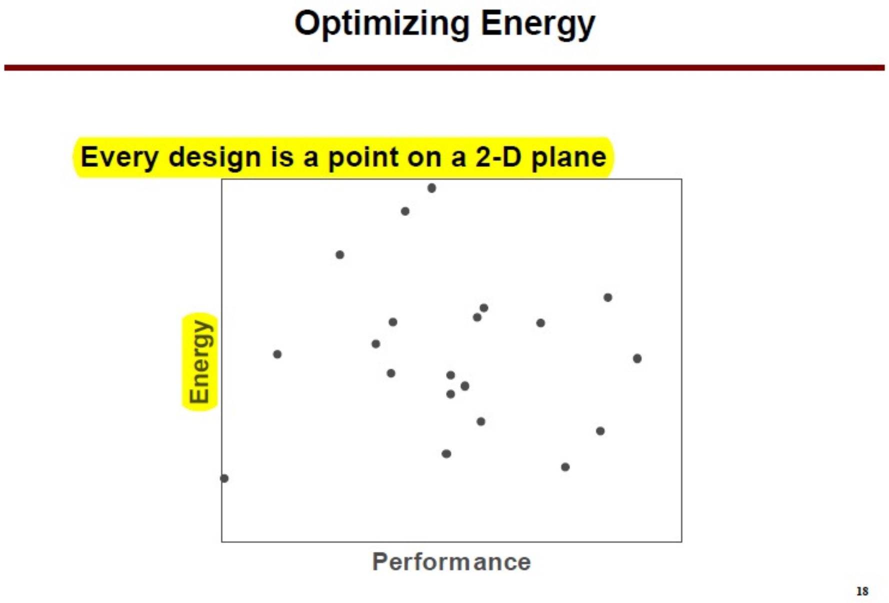
Scaling Power and the Future of CMOS

Mark Horowitz, EE/CS Stanford University

Energy Transparency

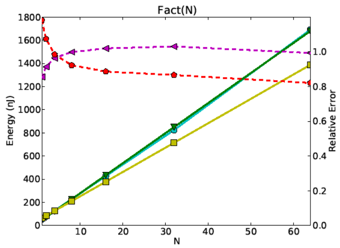


- For HW designers:
  “Power is a 1st and last order design constraint.”
  [Dan Hutcheson, VLSI Research, Inc., E^{3}S Keynote 2011]
- “Every design is a point in a 2D plane.”
  [Mark Horowitz,E^{3}S 2009]


Optimizing Energy

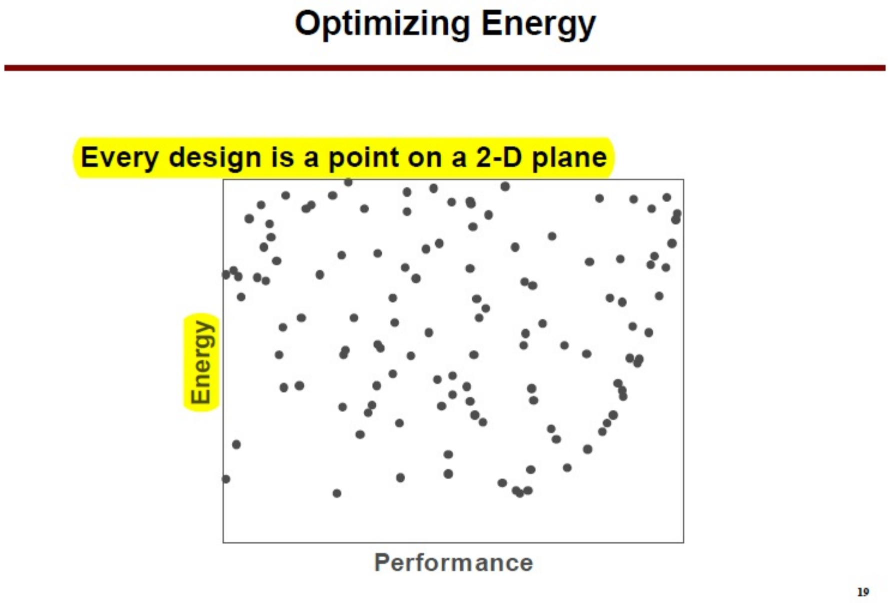
Every design is a point on a 2-D plane

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]

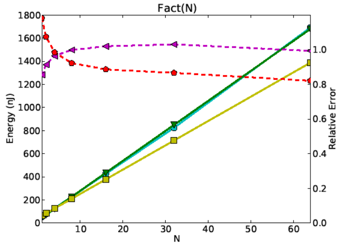


Optimizing Energy

Every design is a point on a 2-D plane

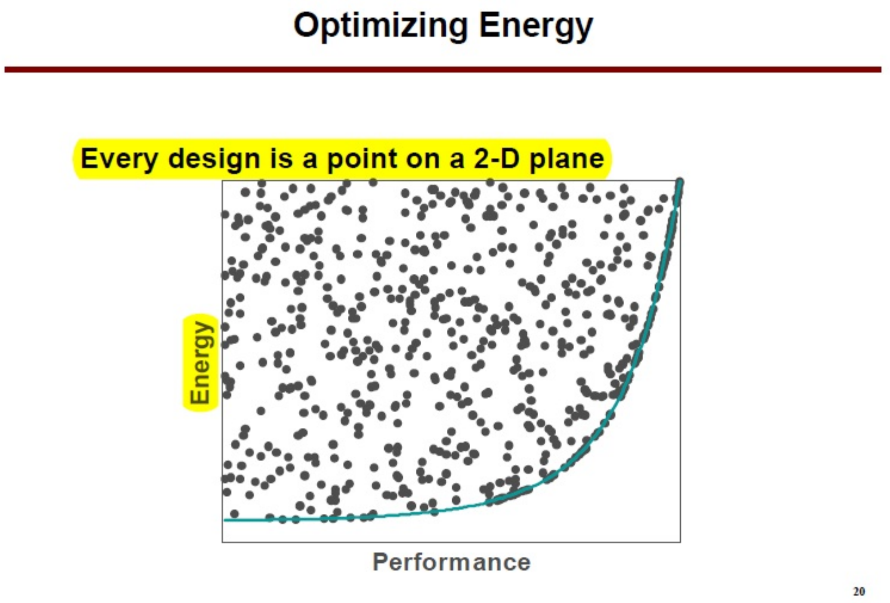
19

Energy Transparency



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

- “Every design is a point in a 2D plane.”
  [Mark Horowitz, E^3S 2009]



Optimizing Energy

Every design is a point on a 2-D plane

20

More POWER to SW Developers

`in 5pJ do {...}`

- Full **Energy Transparency** from HW to SW
- Location-centric programming model

“Cool” code for green software

A cool programming competition!

Promoting energy efficiency to a 1st class SW design goal is still a very important research challenge.

Pictures taken from the Energy Efficient Computing Brochure at:
https://connect.innovateuk.org/documents/3158891/9517074/Energy%20Efficient%20Computing%20Magazine?version=1.0

If you want an ultimate low-power system, then you have to worry about *energy usage at every level in the system design*, and you have to get it right from top to bottom, because any level at which you get it wrong is going to lose you perhaps an order of magnitude in terms of power efficiency.

The hardware technology has a first-order impact on the power efficiency of the system, but you've also got to have software at the top that avoids waste wherever it can. You need to avoid, for instance, anything that resembles a polling loop because that's just burning power to do nothing.

I think one of the hard questions is whether you can pass the responsibility for the software efficiency right back to the programmer.

Do programmers really have any understanding of how much energy their algorithms consume?

I work in a computer science department, and it's not clear to me that we teach the students much about how long their algorithms take to execute, let alone how much energy they consume in the course of executing and how you go about optimizing an algorithm for its energy consumption.

Some of the responsibility for that will probably get pushed down into compilers, but I still think that fundamentally, at the top level, **programmers will not be able to afford to be ignorant about the energy cost of the programs they write.**

What you need in order to be able to work in this way at all is instrumentation that tells you that running this algorithm has this kind of energy cost and running that algorithm has that kind of energy cost.

You need tools that give you feedback and tell you how good your decisions are.

Currently the tools don't give you that kind of feedback.

February 2010, acmqueue Interview with Steve Furber
The designer of the ARM chip shares lessons on energy-efficient computing at: http://queue.acm.org/detail.cfm?id=1716385

Steve Furber

Thank you for your attention

ENTRA
Whole-Systems Energy Transparency

MAGEEC

EMBECOSM®

ICTenergy

cadence®

XMOS®

The Royal Academy of Engineering

Kerstin.Eder@bristol.ac.uk