Energy Aware Computing

Why SW engineers are key to energy efficient computing

Kerstin Eder and James Pallister

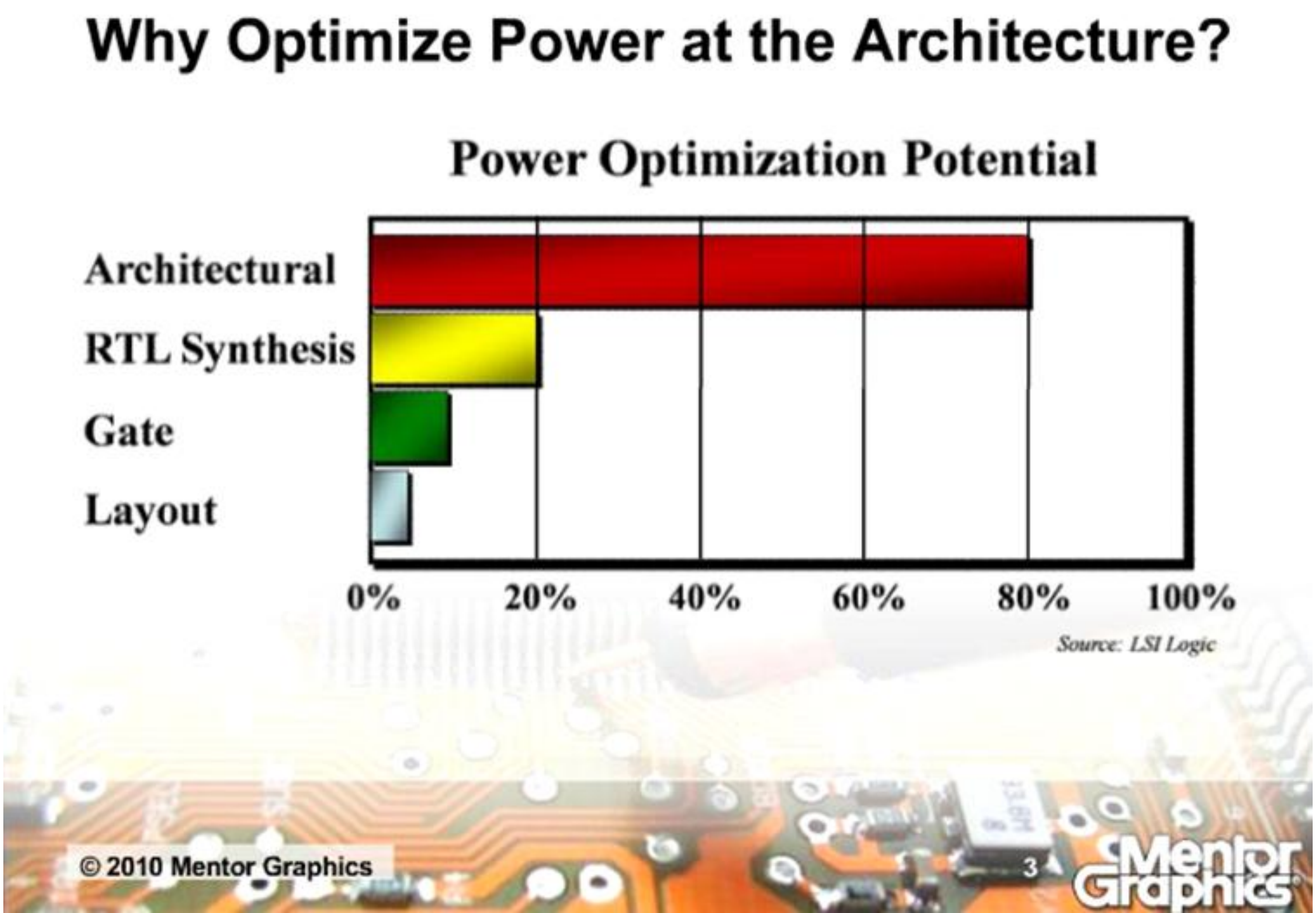
Design Automation and Verification, Microelectronics

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

Energy Aware System Design

- Power management largely in domain of Hardware Design
  - Considerations to minimize/optimize
    - Dynamic (switching) and static (leakage) power
    - Energy consumption
  - On-chip power management
    - DVFS
    - Modes: on, standby, suspend, sleep, off
- Where can the greatest savings be made?

Greater Savings at Higher Levels



LOW POWER

Lack of software support marks the low power scorecard at DAC

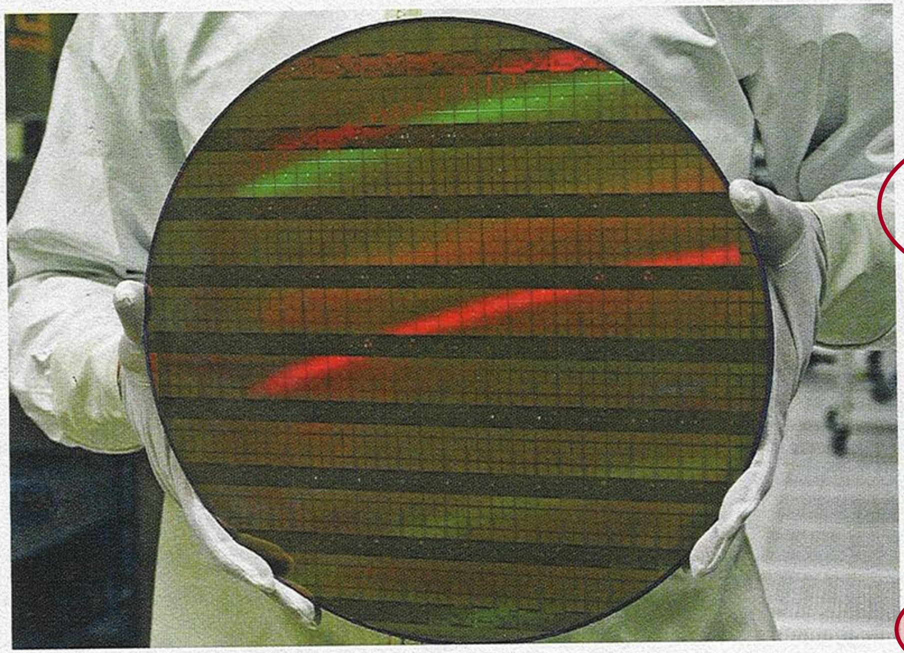
One of the panels at the Design Automation Conference (DAC), which took place in California in early June, set out to get an idea of how well the industry is doing at delivering lower-power systems.

It is becoming clear, *writes Chris Edwards*, that the system level is currently the missing link.

Processes can deliver some gains – and Globalfoundries' Andrew Brotman was able to outline some of the features that the foundry has put into its recently launched low-power high-k, metal gate (HKMG) process.

FinFETs should bring power down as those processes become available, although they are not the only options. But if the software keeps cores active for no good reason, the lower switching power per bit processed won't deliver a realised saving.

In his keynote speech Gadi Singer, vice-president IAG and general manager of the SoC enabling group at Intel Corporation, said that with limited software support, dedicated low-



Intel waits for better low-power software control

power circuitry could save maybe 20% in a typical multimedia-oriented core.

Make the software controlling it better at controlling the power states and that difference could be three to five times.

During an afternoon panel discussion Ambrose Low, director of design engineering at Broadcom said: "We have hundreds of knobs in the hardware to turn power down.

"The question is whether we can take the actual use-cases into consideration and optimise the software to power the logic circuits down. We still have a long way to go."

Ruggero Castagnetti of LSI argued that the desire to do more in software will grow.

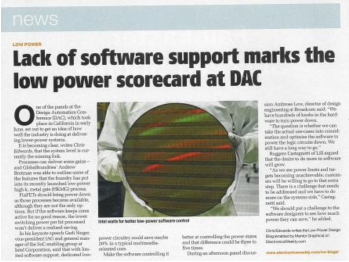
"As we see power limits and targets becoming unachievable, customers will be willing to go to that extra step. There is a challenge that needs to be addressed and we have to do more on the systems side," Castagnetti said.

"We should put a challenge to the software designers to see how much power they can save," he added.

Chris Edwards writes the Low-Power Design Blog (enabled by Mentor Graphics) on ElectronicsWeekly.com

www.electronicsweekly.com/ew-blogs/

Wasted Potential


news

Lack of software support marks the low power scorecard at DAC

Huge advances have been made in power-efficient hardware.

BUT – potential energy savings are wasted by

- software that does not exploit energy-saving features of hardware;
- poor dynamic management of tasks and resources.

The Focus is on Software


news

Lack of software support marks the low power scorecard at DAC

- Software controls the behaviour of the hardware
  - – Algorithms and Data Flow
  - – Compiler (optimizations)
    - Traditional SW design goals:
      - – **performance, performance, performance**
- Software engineers often **“blissfully unaware”**
  - – Implications of algorithm/code/data on power/energy?
  - – Power/Energy considerations
    - at best, secondary design goals
  - – BUT the biggest savings can be gained from optimizations at the higher levels of abstraction in the system stack – this includes Algorithms, Data and SW

Aligning SW Design Decisions with Energy Efficiency as Design Goal

Key steps*:

- *"Choose the **best algorithm** for the problem at hand and make sure it **fits well with** the computational **hardware**. Failure to do this can lead to costs far exceeding the benefit of more localized power optimizations.*
- *Minimize **memory size** and expensive **memory accesses** through algorithm transformations, efficient mapping of data into memory, and optimal use of memory bandwidth, registers and cache.*
- *Optimize the **performance** of the application, making **maximum use of available parallelism.***
- *Take advantage of **hardware support for power management**.*
- *Finally, select instructions, sequence them, and order operations in a way that **minimizes switching** in the CPU and datapath."*

* Kaushik Roy and Mark C. Johnson. **1997**. "Software design for low power". In *Low power design in deep submicron electronics*, Wolfgang Nebel and Jean Mermet (Eds.). Kluwer Nato Advanced Science Institutes Series, Vol. 337. Kluwer Academic Publishers, Norwell, MA, USA, pp 433-460.

6.3. SOFTWARE DESIGN FOR LOW POWER

KAUSHIK ROY AND MARK C. JOHNSON
School of Electrical and Computer Engineering
Purdue University
West Lafayette, Indiana, U.S.A.

1. Introduction

It is tempting to suppose that only hardware dissipates power, not software. However, that would be analogous to postulating that only automobiles burn gasoline, not people. In microprocessor, micro-controller, and digital signal processor based systems, it is software that directs much of the activity of the hardware. Consequently, the software can have a substantial impact on the power dissipation of a system. Until recently, there were no efficient and accurate methods to estimate the overall effect of a software design on power dissipation. Without a power estimator there was no way to reliably optimize software to minimize power. Since 1993, a few researchers have begun to crack this problem. In this chapter, you will learn

Performance, Performance, Performance

- **Key steps:**
  1. *"Choose the best algorithm…"* ***FEWER CYCLES***
  2. *"Minimize memory accesses…"* ***FEWER CYCLES***
  3. *"Make maximum use of parallelism..."*
     FEWER CYCLES
  4. *"Minimize switching…"*
  5. *"Exploit h/w support for power management…"*

Is there more to do?

... there is:

A Linux implementation famously wasted 70-90% of its energy, simply waking several times a second to drive a blinking cursor.

Idle software uses energy

	Configuration	Battery Life	Idle Power
1	OS + Minimum Drivers	6:32	8.6 W
2	+ IT Build	4:43	11.9 W
3	+ User Applications	4:17	13.1 W

Note: Measured in 2009 with actual OEM Notebook PC and corporate IT build

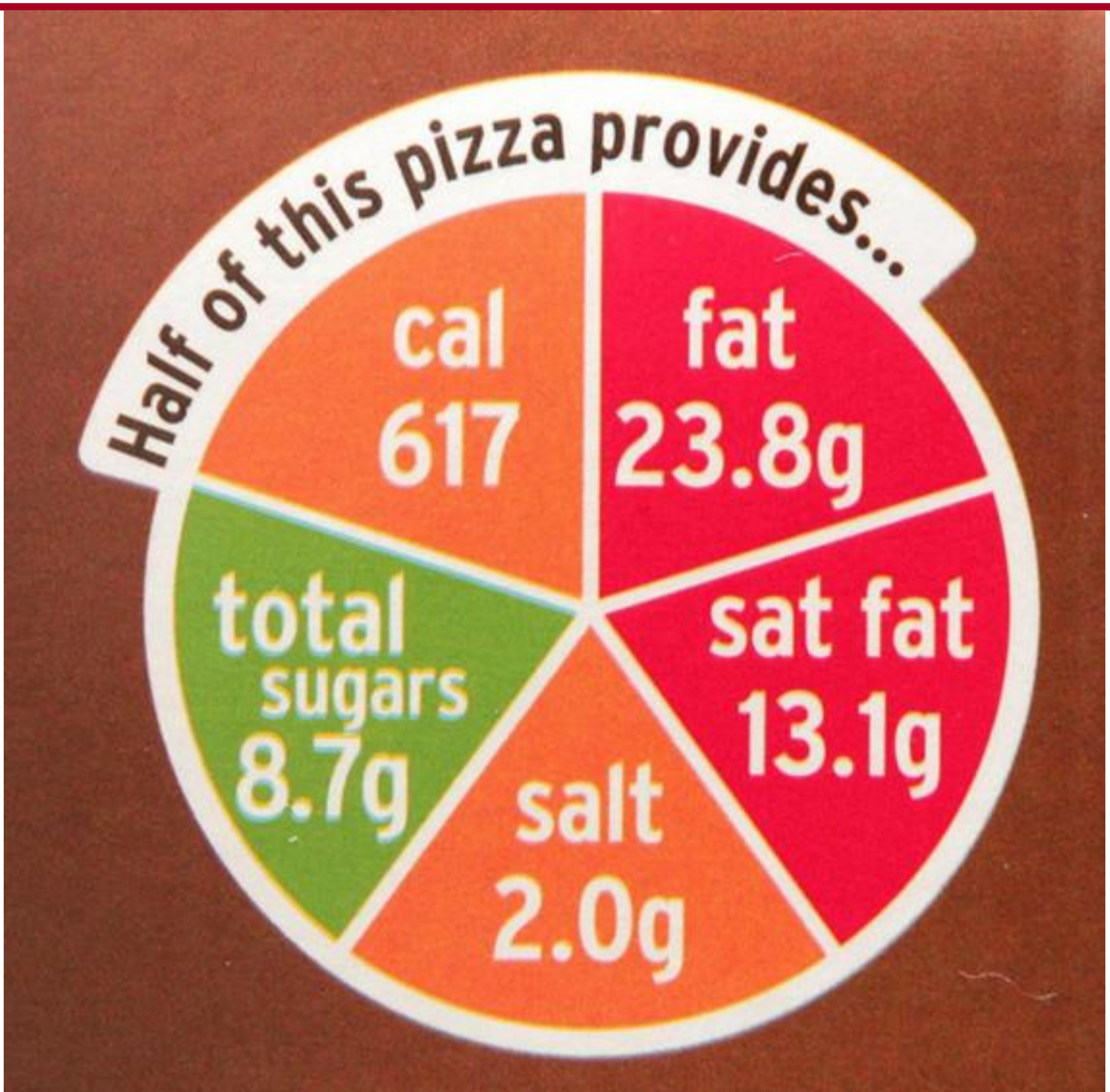
The LCD display remained on at minimum brightness with no screen saver, and the system did not go into standby or sleep.

Energy Transparency

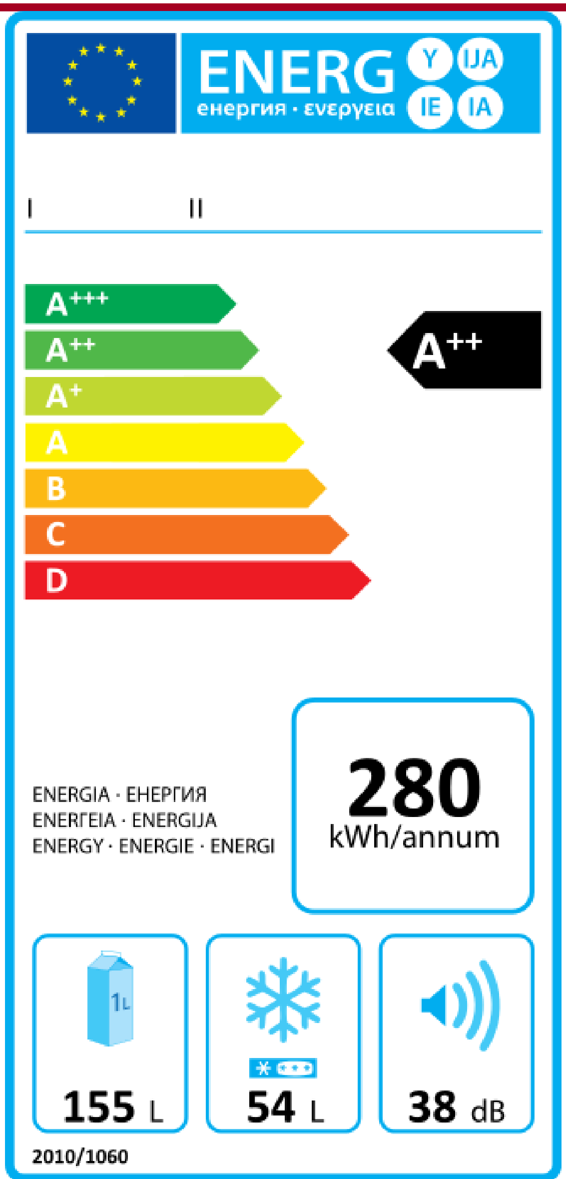
Information on energy usage is available for programs:

- ideally without executing them, and
- at all levels from machine code to high-level application code.

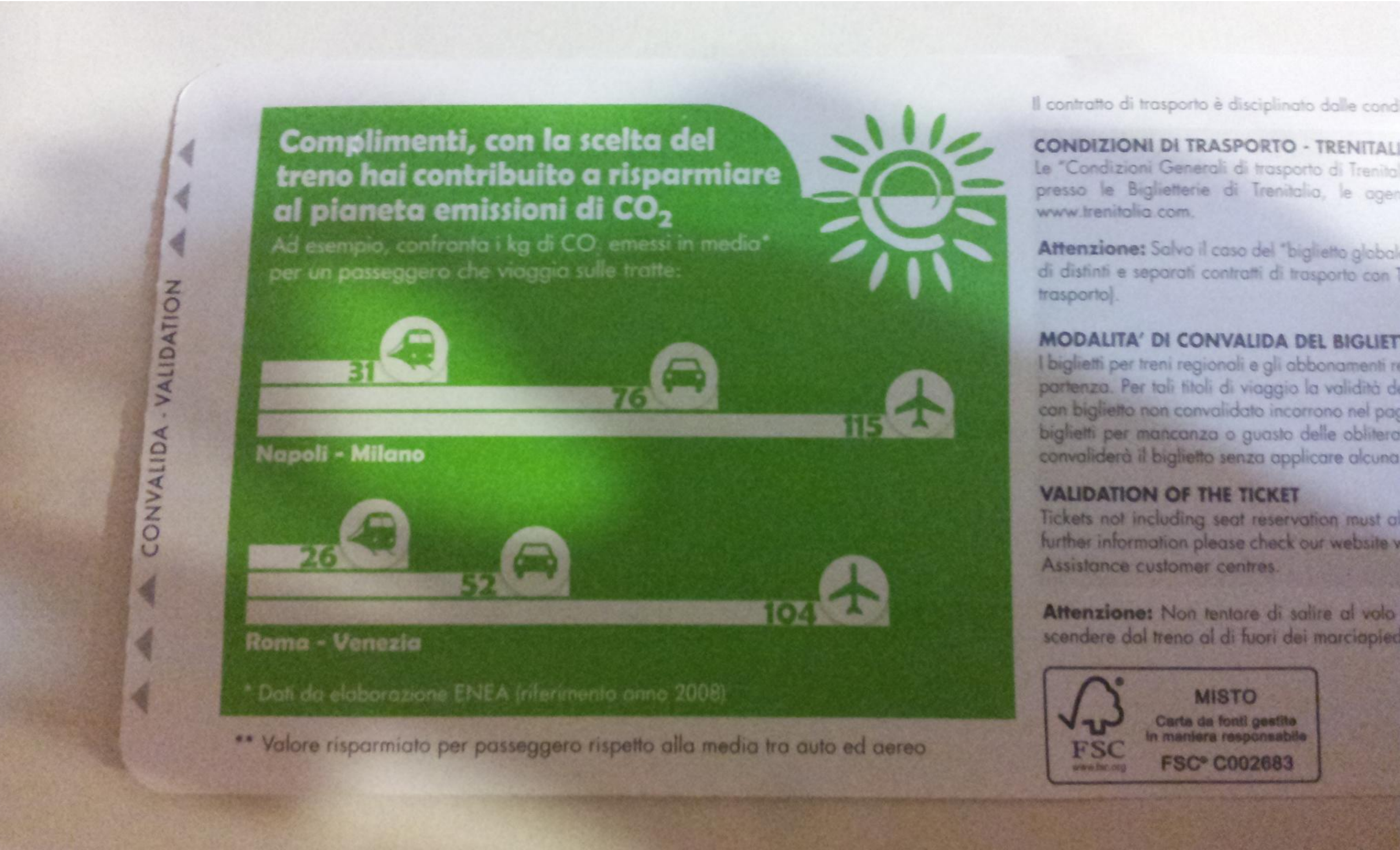
Transparency



Transparency



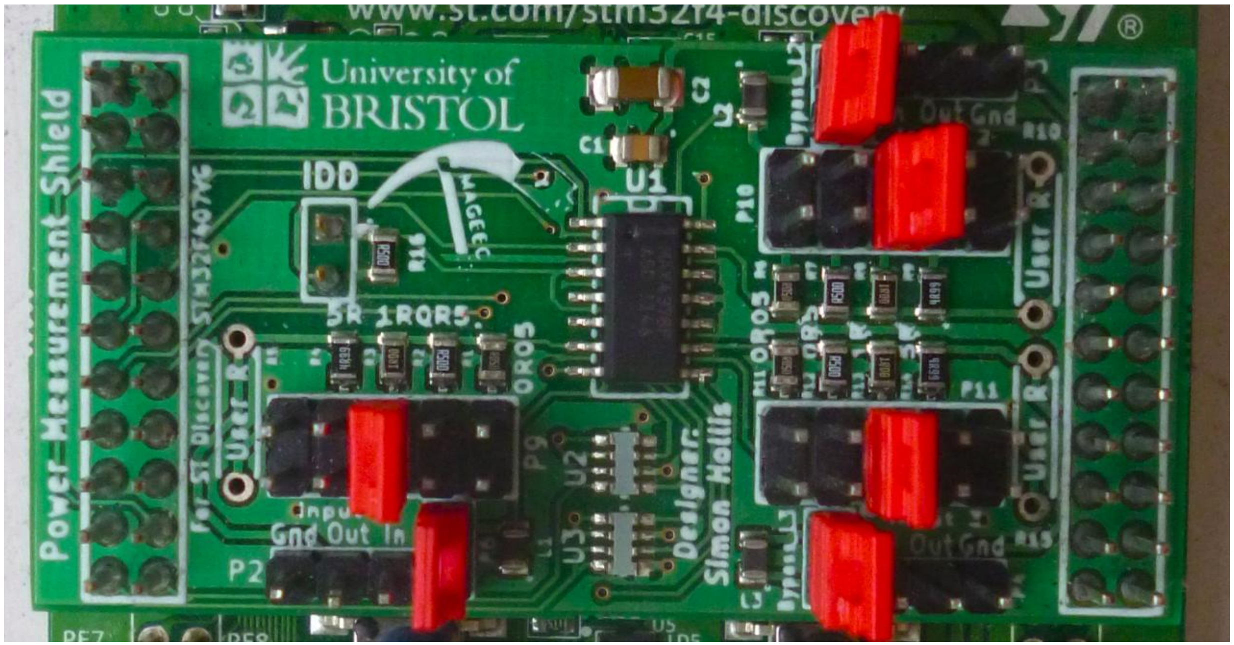
Transparency



Why Energy Transparency?

Energy transparency enables a deeper understanding of how algorithms and coding impact on the energy consumption of a computation when executed on hardware.

Open Energy Measurement Hardware



James Pallister, PhD Student, University of Bristol

Technical Advisor, Embecosm

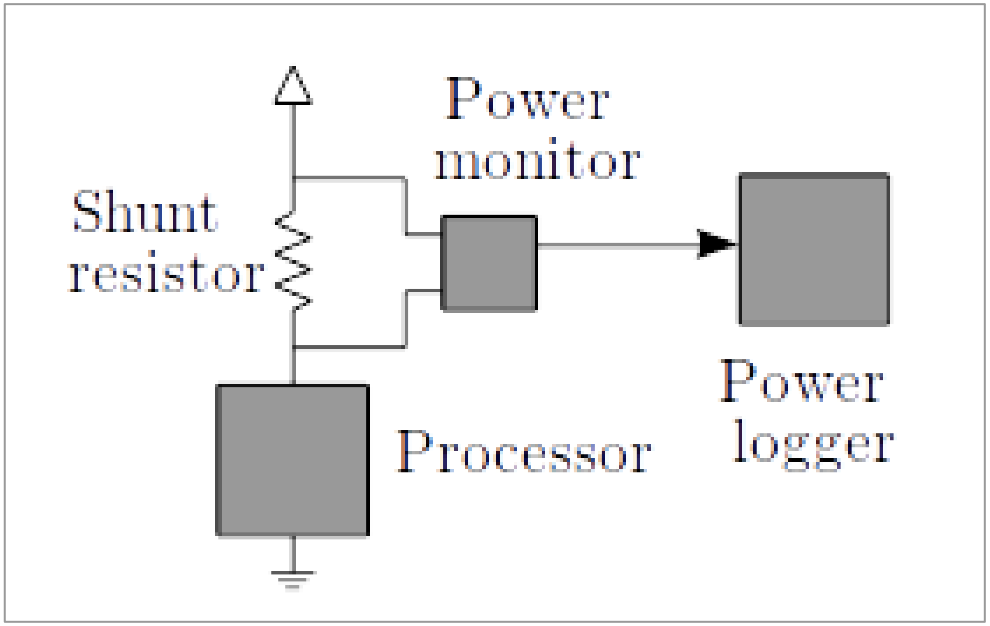
Measuring Power

Repeat frequently, timestamp each sample

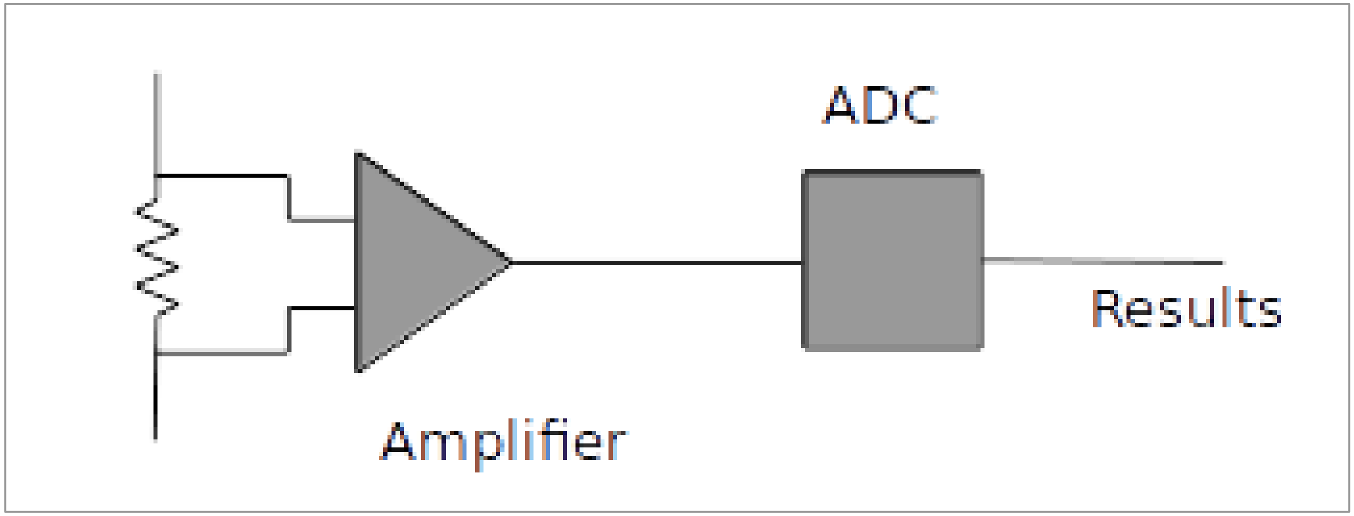
- Measure voltage drop across the resistor
- Measure voltage at one side of the resistor

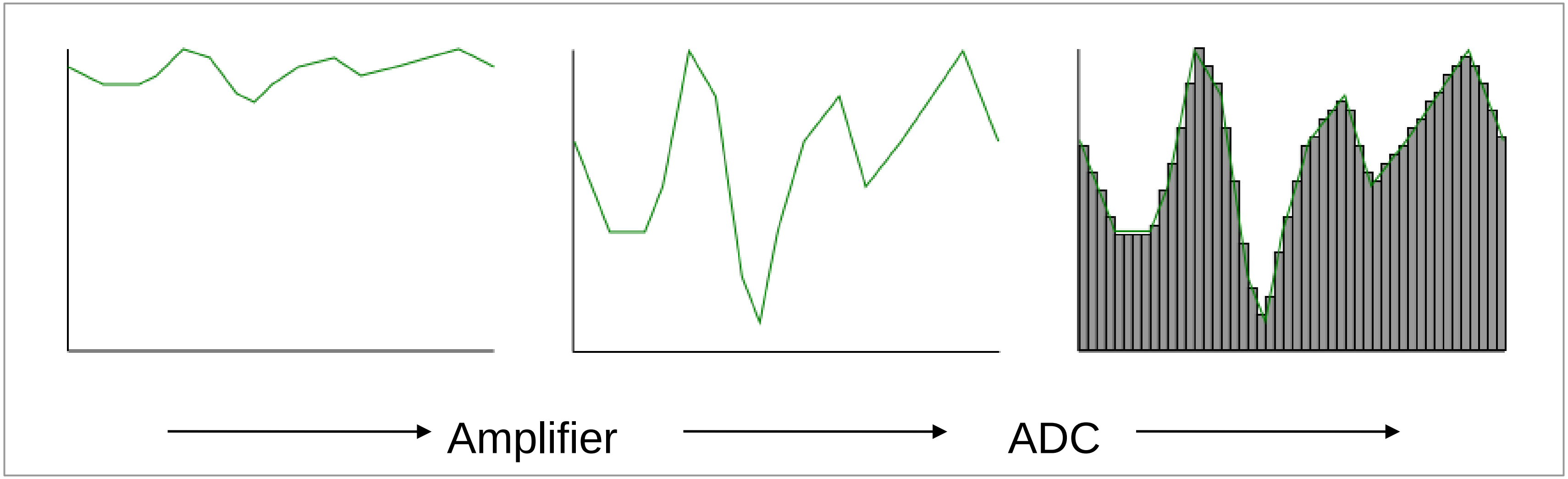
I = Vshunt / Rshunt to find the current

P = I × V to calculate the power



The Power Monitor

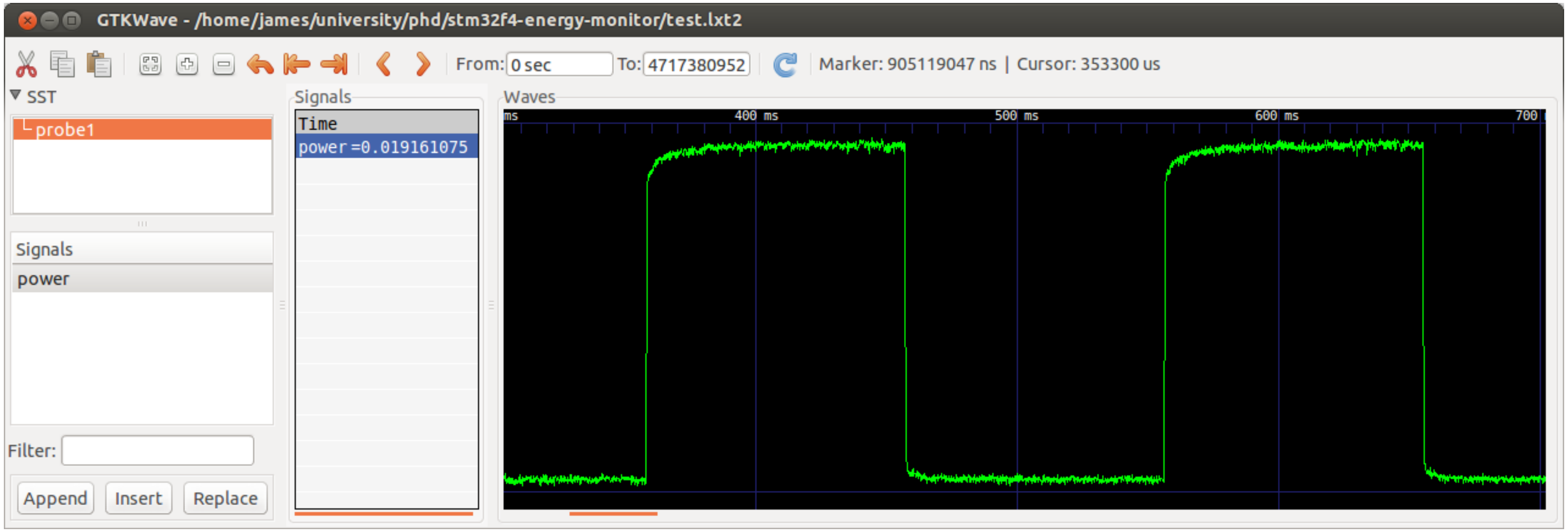




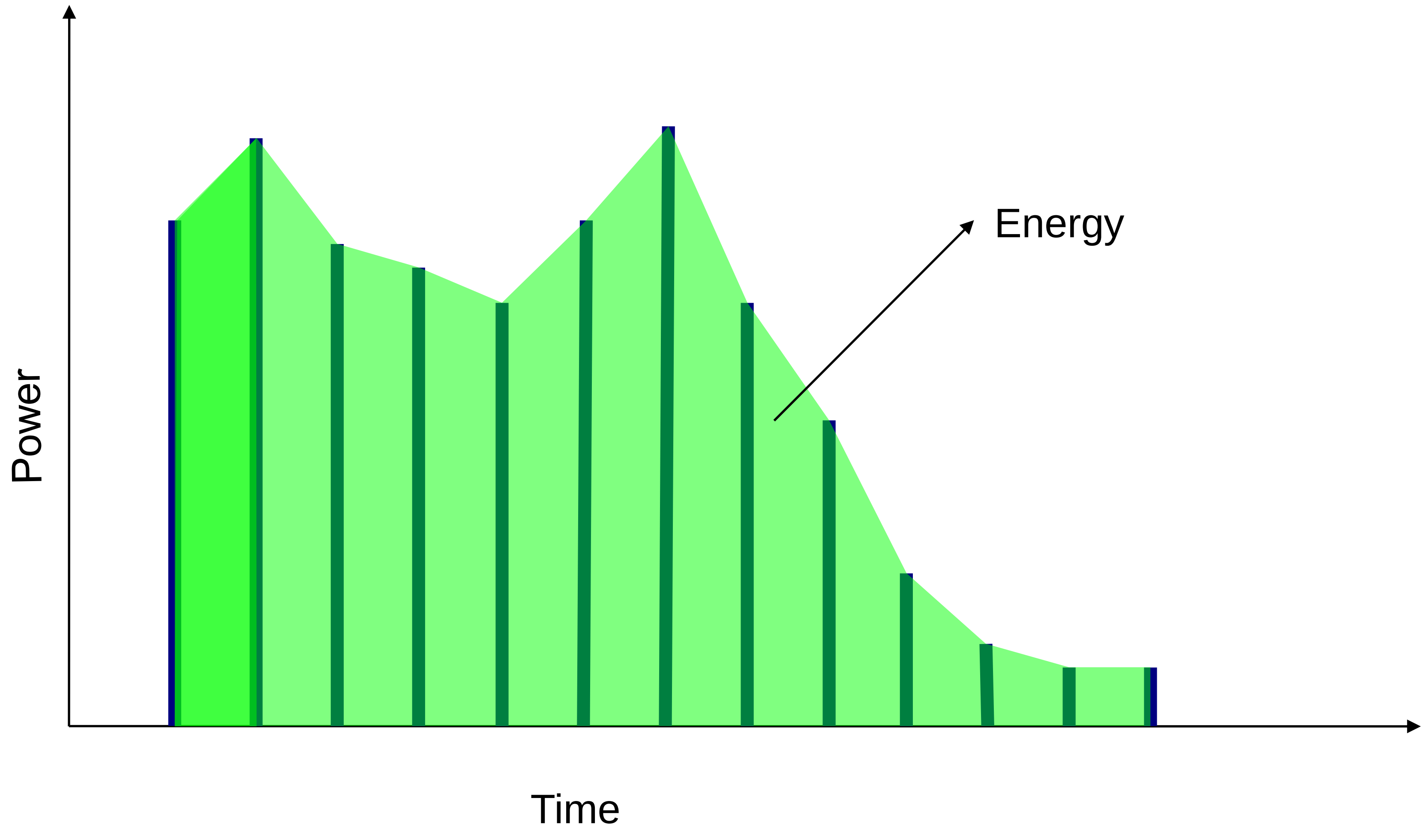
How much data?

Currently 500,000 Samples/second

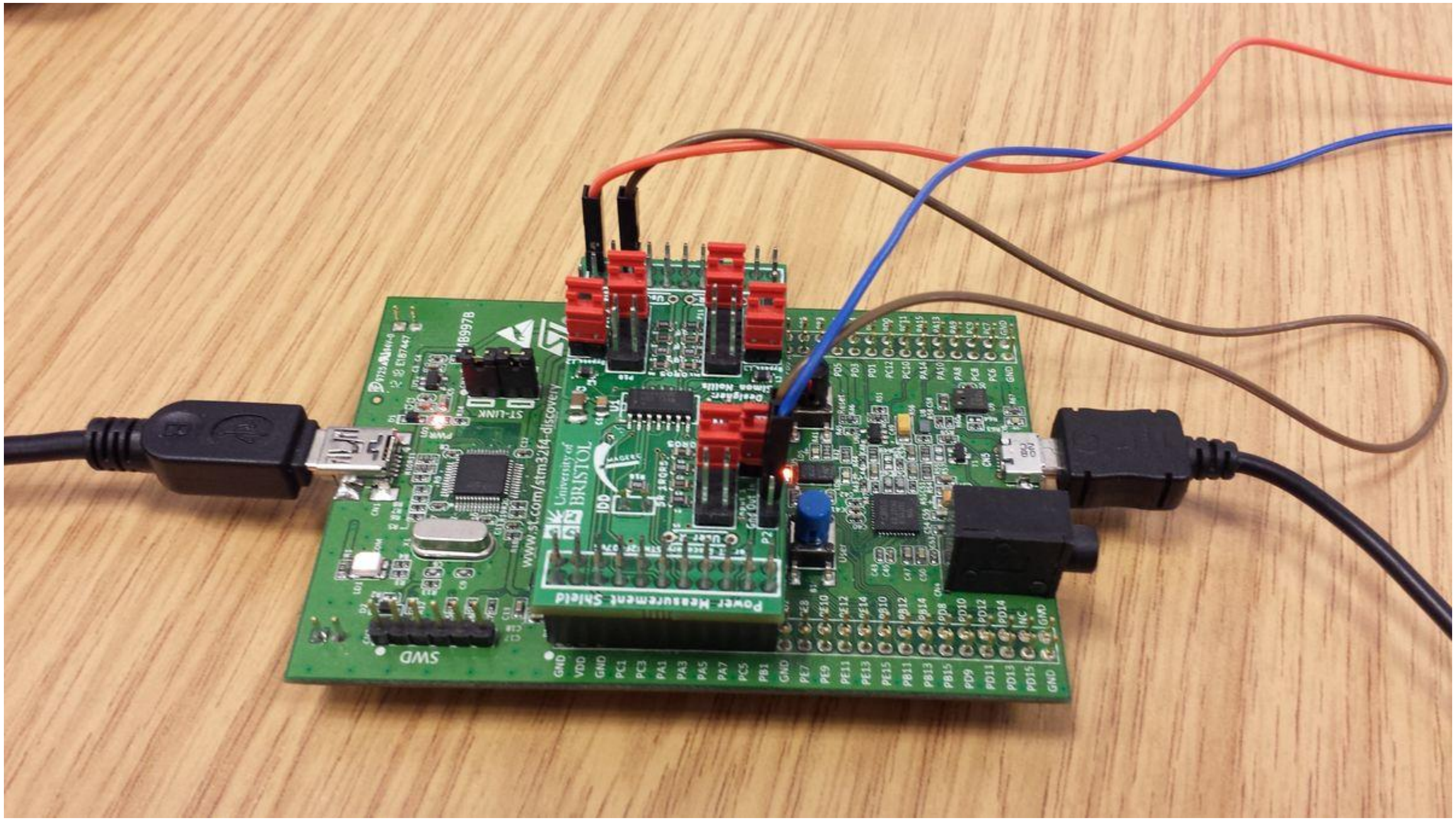
6,000,000 S/s possible in bursts



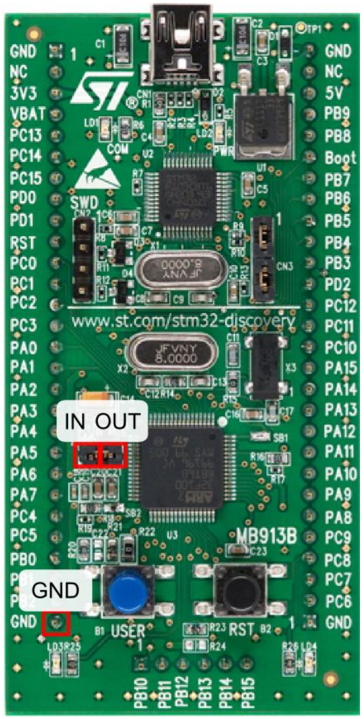
Measuring Energy

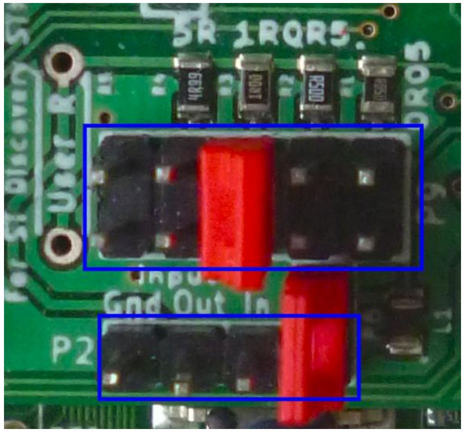


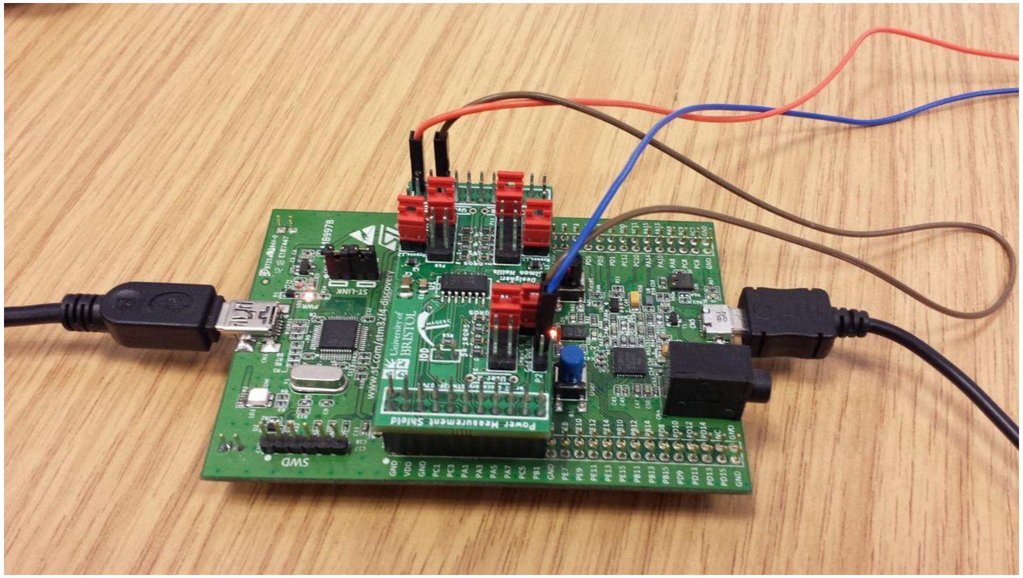
Energy Measurement



Connecting up a board







Live demo?

by James Pallister

and

Steve Kerrison

19 March 2012 Last updated at 17:34

Free mobile apps 'drain battery faster'

Free mobile apps which use third-party services to display advertising consume considerably more battery life, a new study suggests.

Like many games, Angry Birds has a free version supported by targeted advertising

Researchers used a special tool to monitor energy use by several apps on Android and Windows Mobile handsets.

Findings suggested that in one case 75% of an app's energy consumption was spent on powering advertisements.

Report author Abhinav Pathak said app makers must take energy optimisation more seriously.

Related Stories

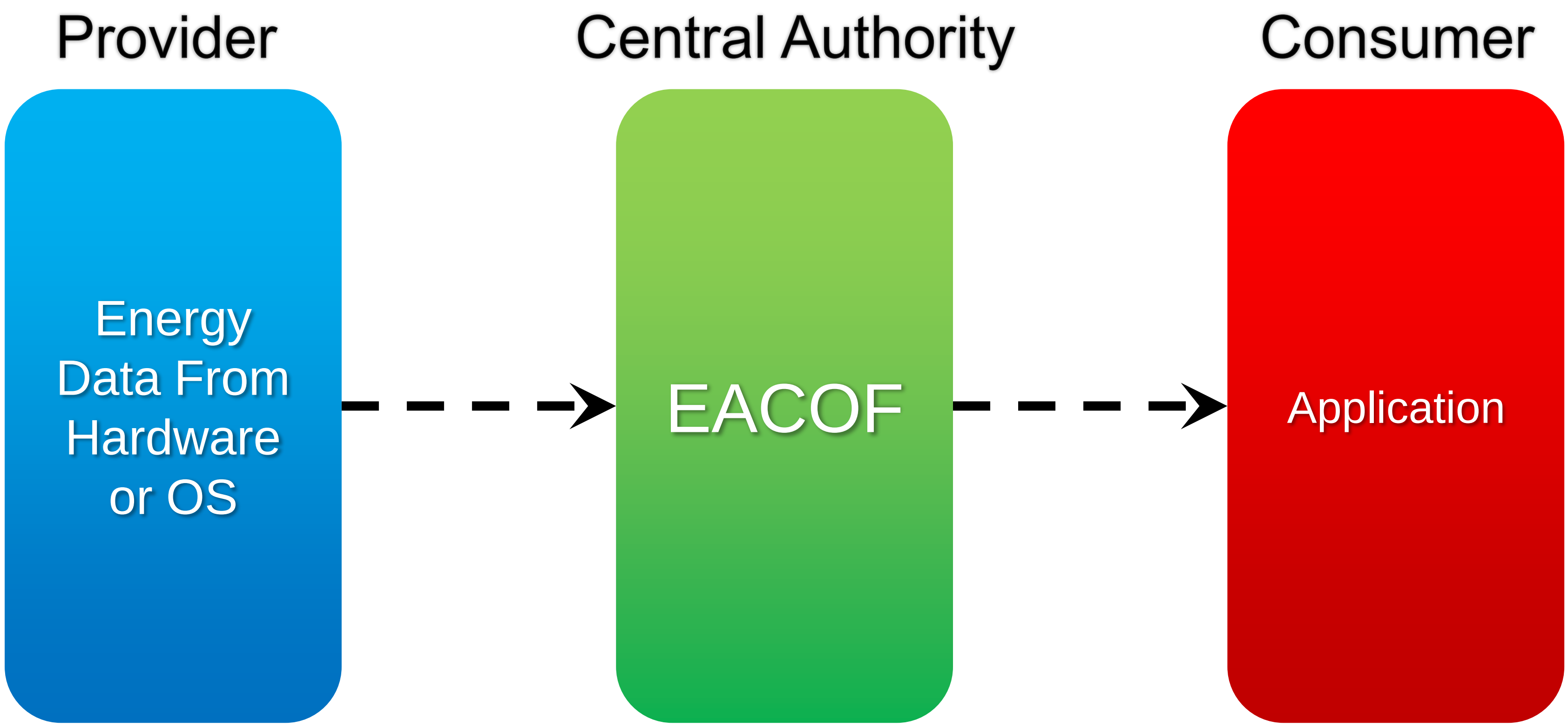
Hot mobile trends for

The EACOF

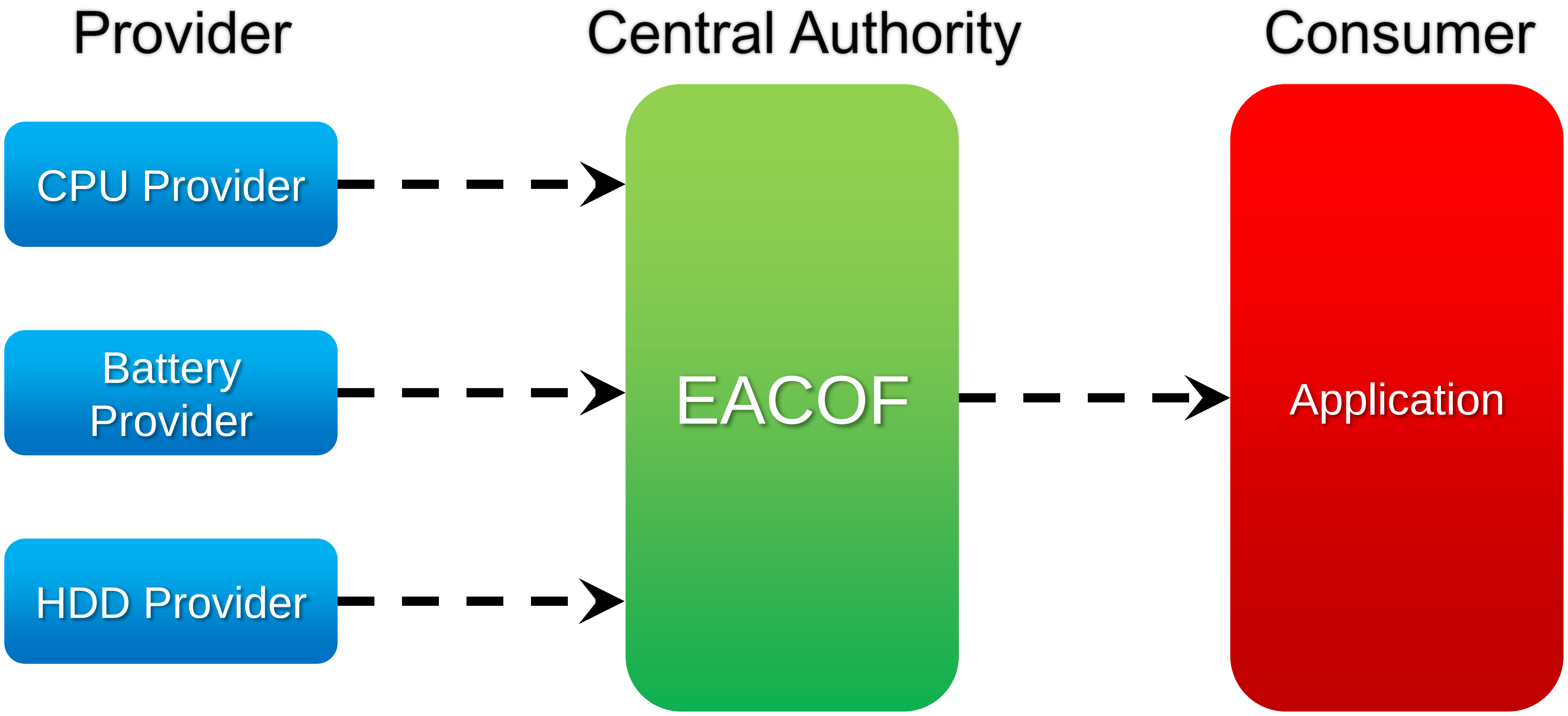
A simple Energy-Aware COmputing Framework

https://github.com/eacof

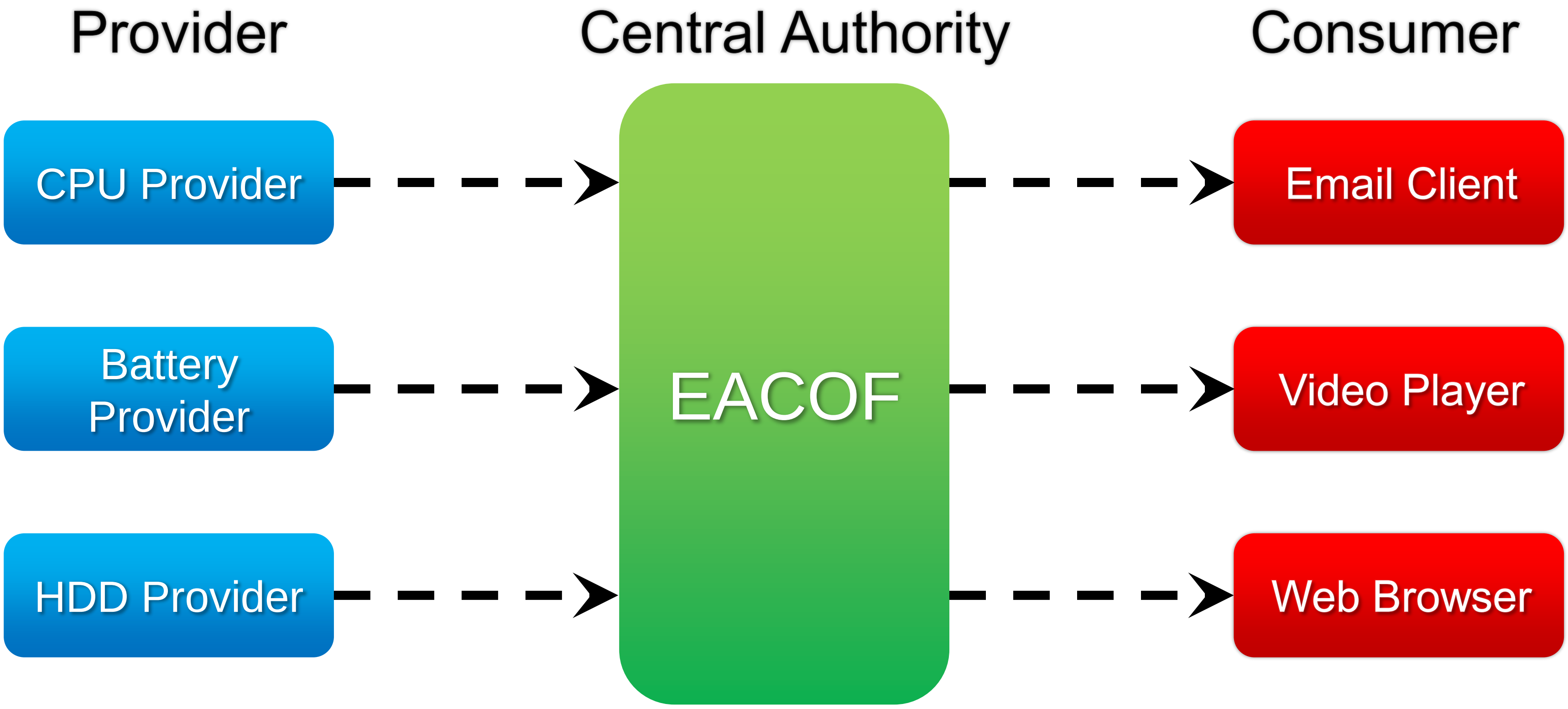
High Level


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

Comparing Sorting Algorithms

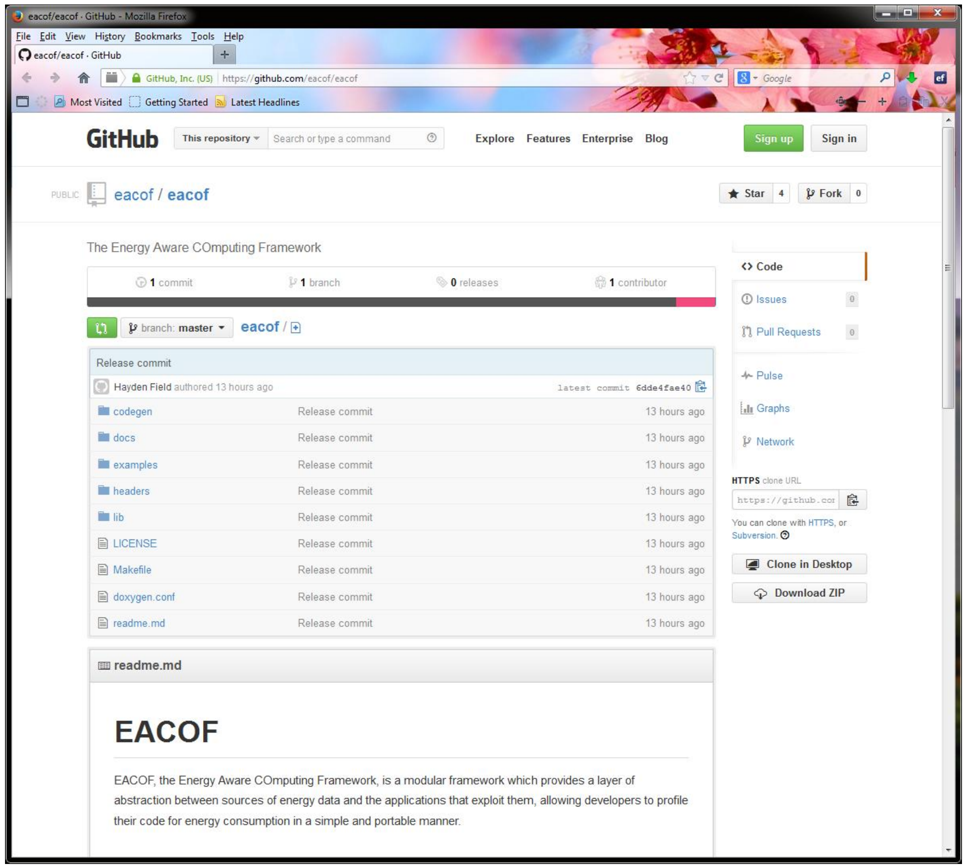
- Sorting of integers in [0,255]

		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.

First EACOF release: Get involved!



github.com/eacof

ENTRA

Whole-Systems
Energy Transparency

The ENTRA Project

- Whole Systems ENergy TRAnsparency
  - 1.10.2012 - 30.9.2015
  - 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).”

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:

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

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

Energy Modelling

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)*:

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

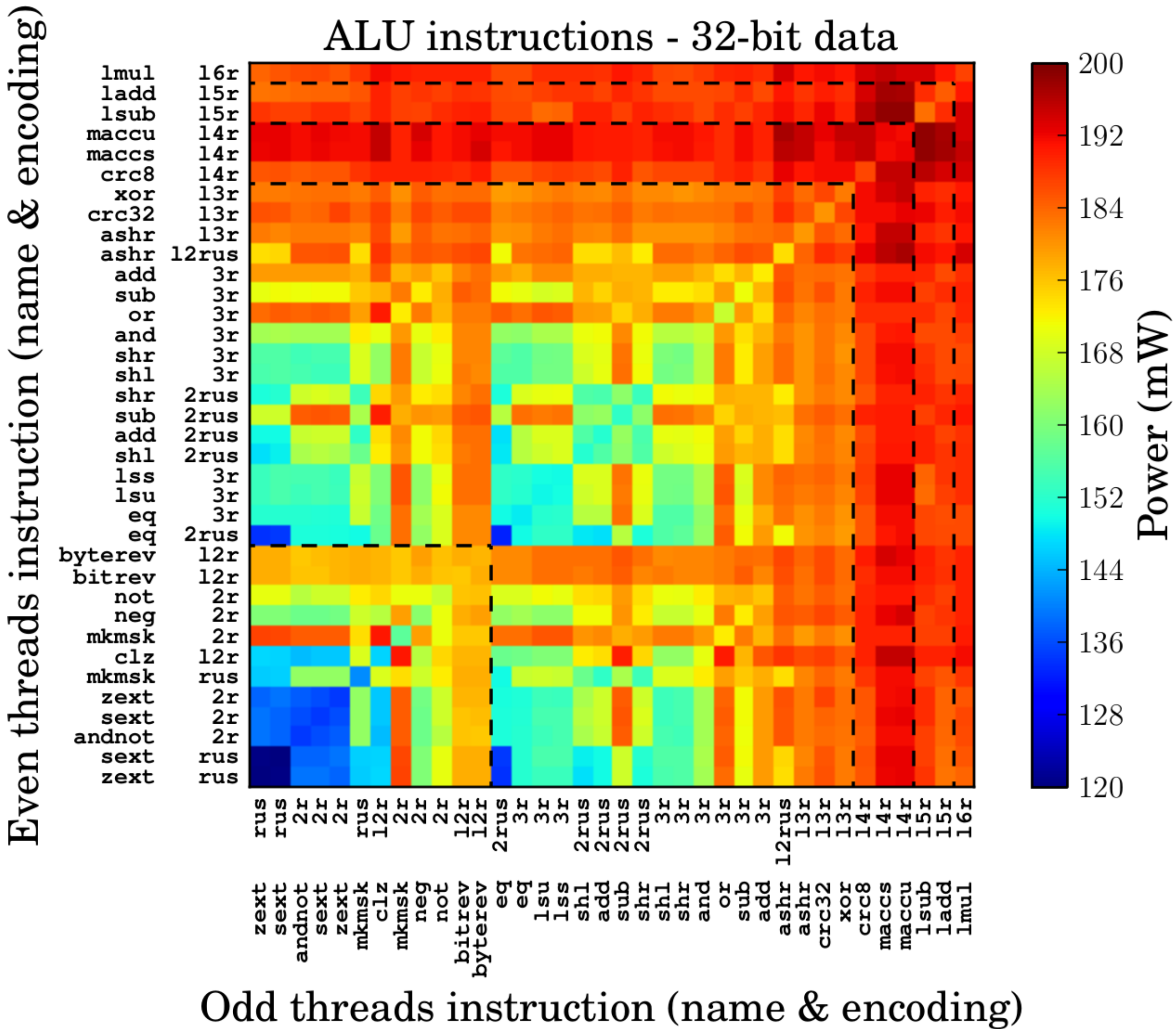
Idle base power and duration

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

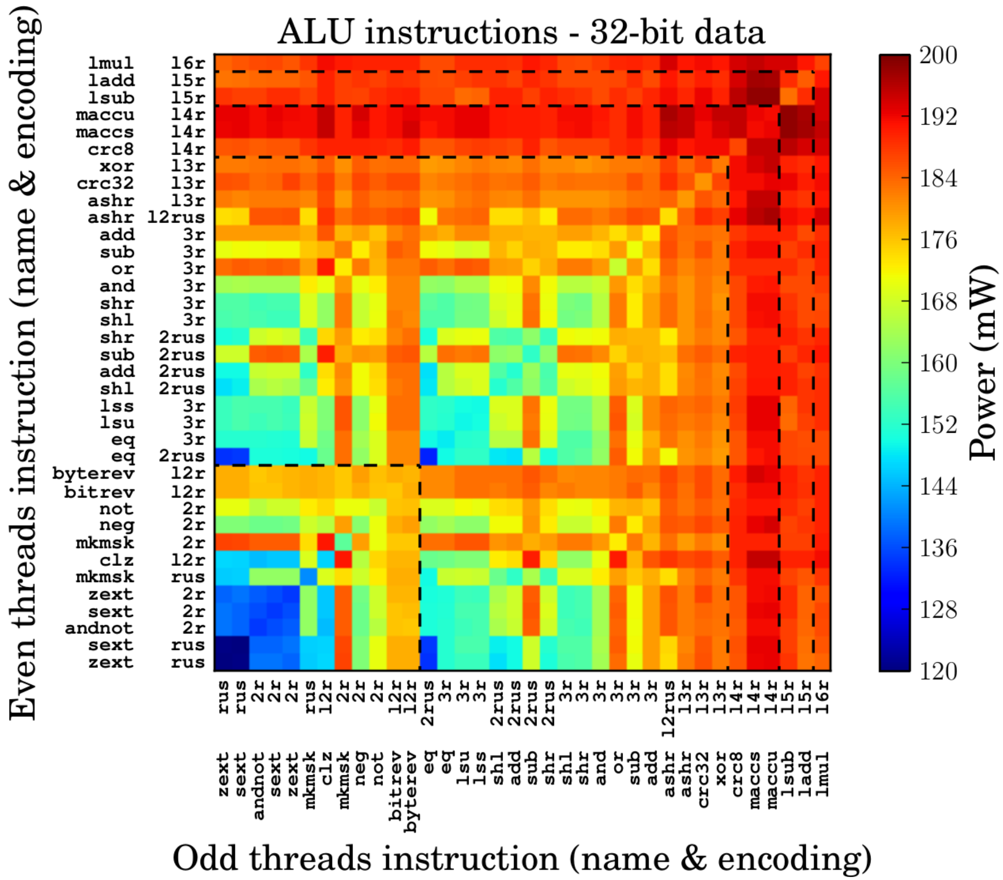
- Use of execution statistics rather than execution trace.
- Fast running model with an average error margin of less than 7%

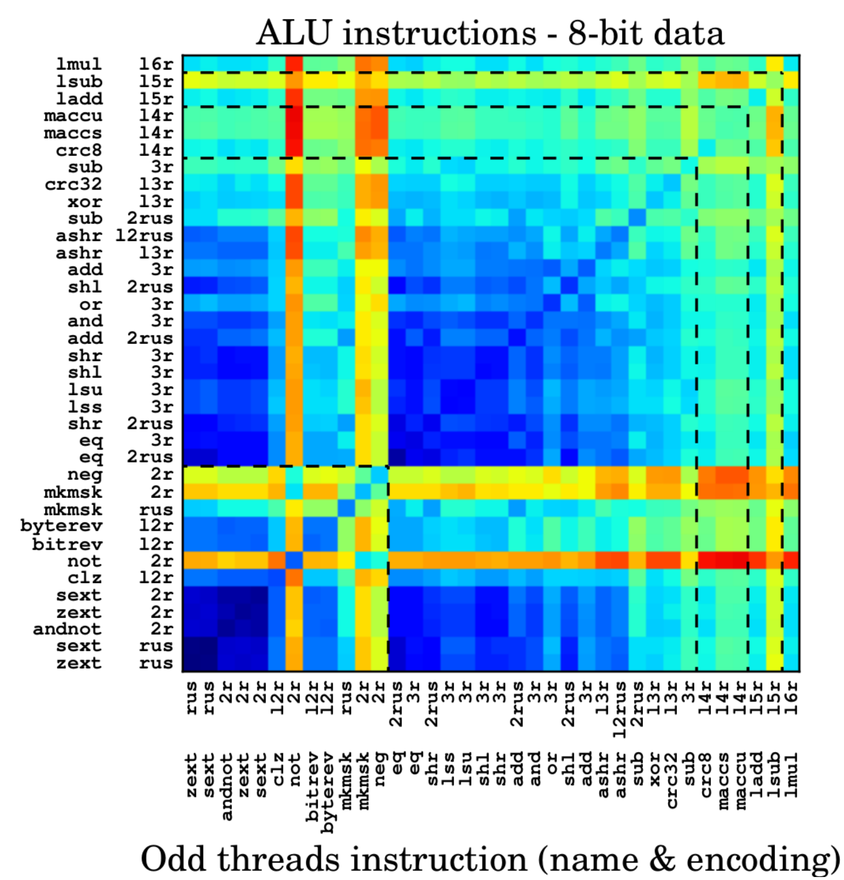
S. Kerrison and K. Eder, 2013. "Energy Modelling of Software for a Hardware Multi-threaded Embedded Microprocessor ", under review at ACM TECS.

ISA Characterization



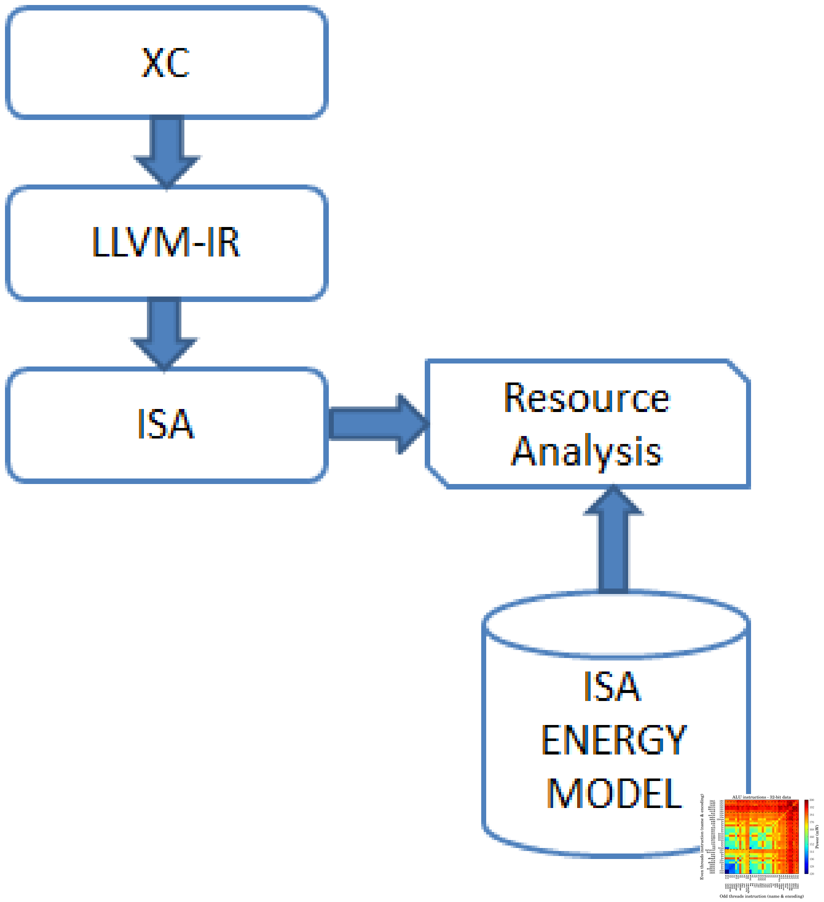
ISA Characterization



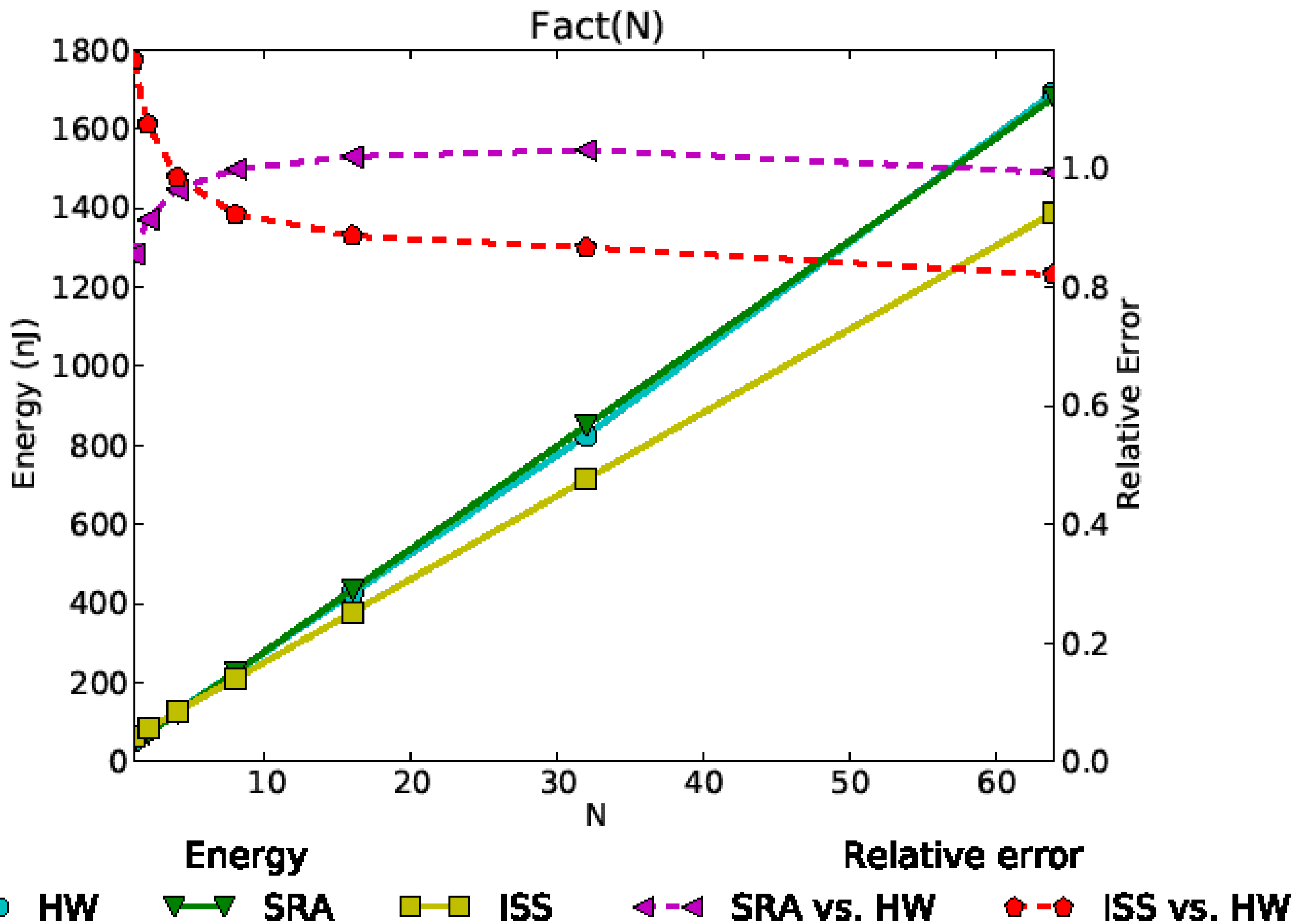


Analysis at the ISA Level

- 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*.

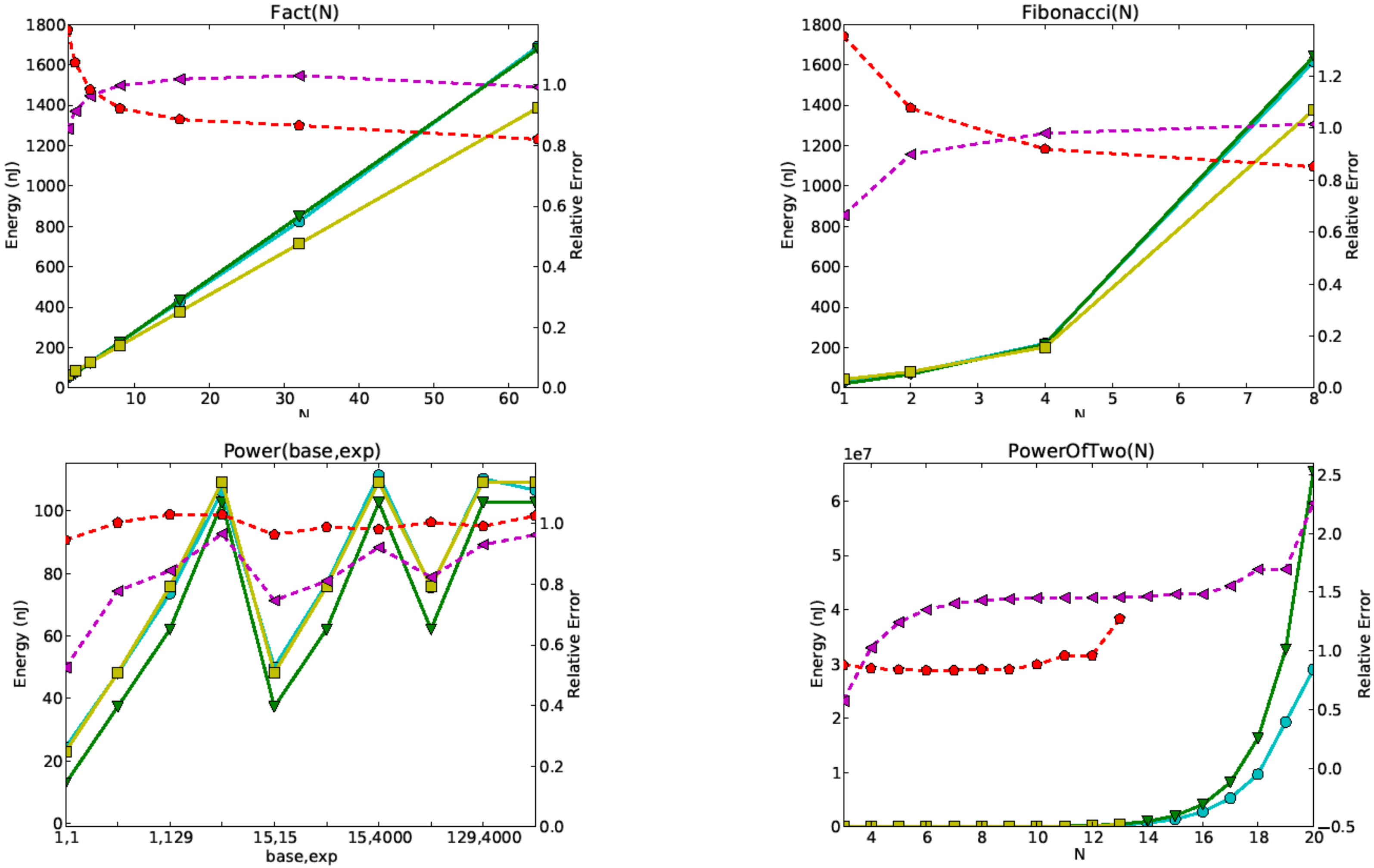


First Results



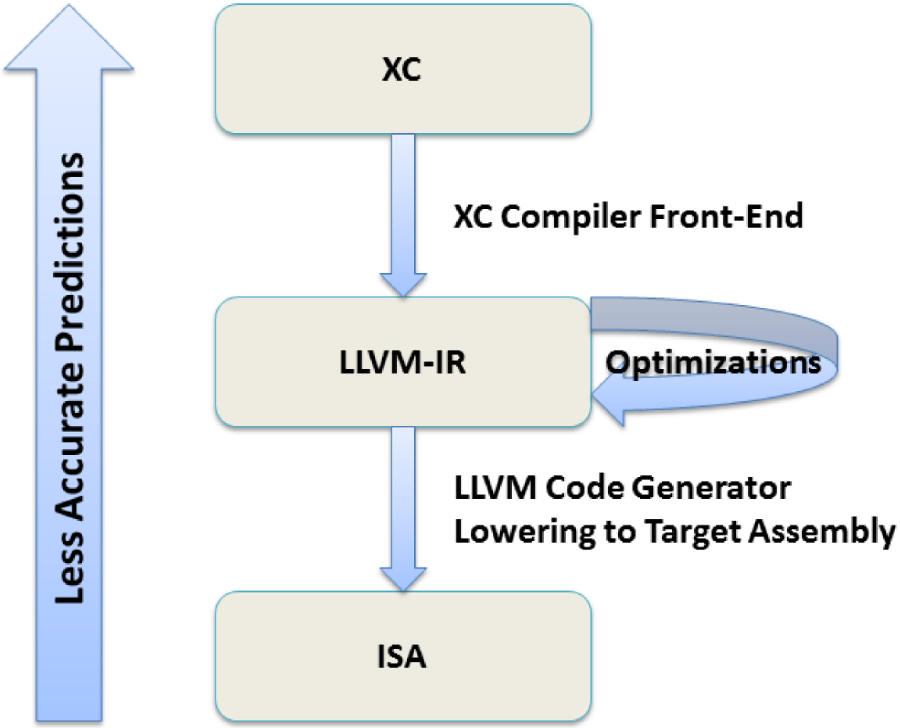
U. Liqat, S. Kerrison, A. Serrano, K. Georgiou, N. Grech, P. Lopez-Garcia, M.V. Hermenegildo and K. Eder. "Energy Consumption Analysis of Programs based on XMOS ISA-Level Models". LOPSTR 2013.

First Results



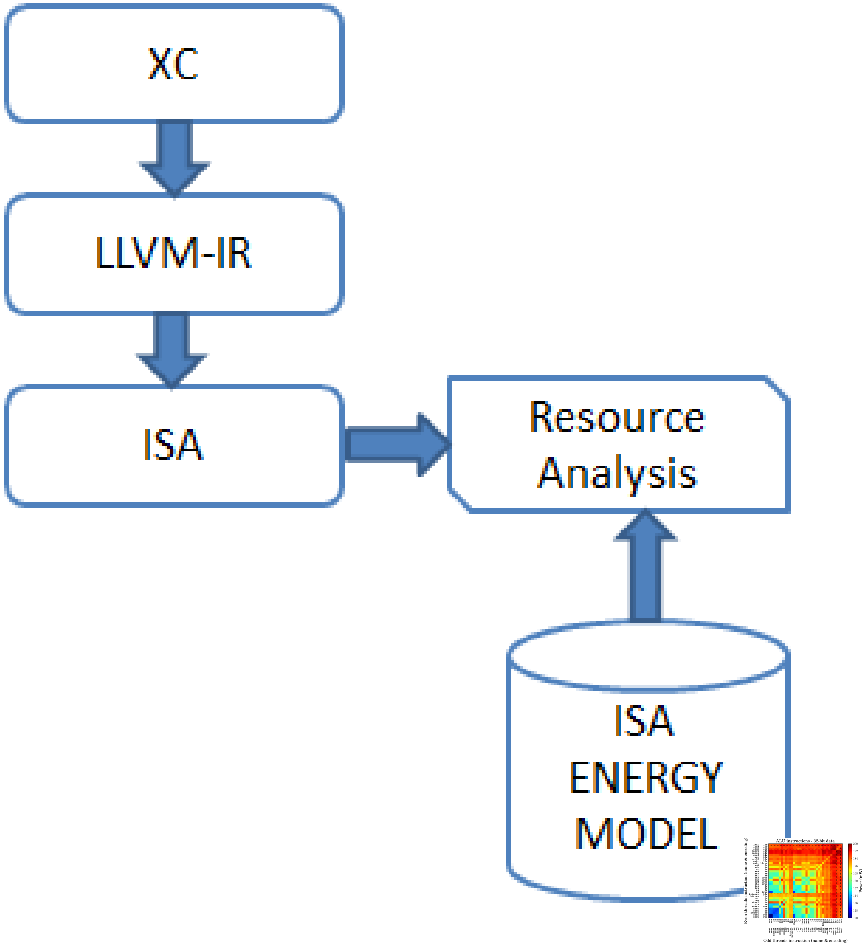
U. Liqat, S. Kerrison, A. Serrano, K. Georgiou, N. Grech, P. Lopez-Garcia, M.V. Hermenegildo and K. Eder. "Energy Consumption Analysis of Programs based on XMOS ISA-Level Models". LOPSTR 2013.

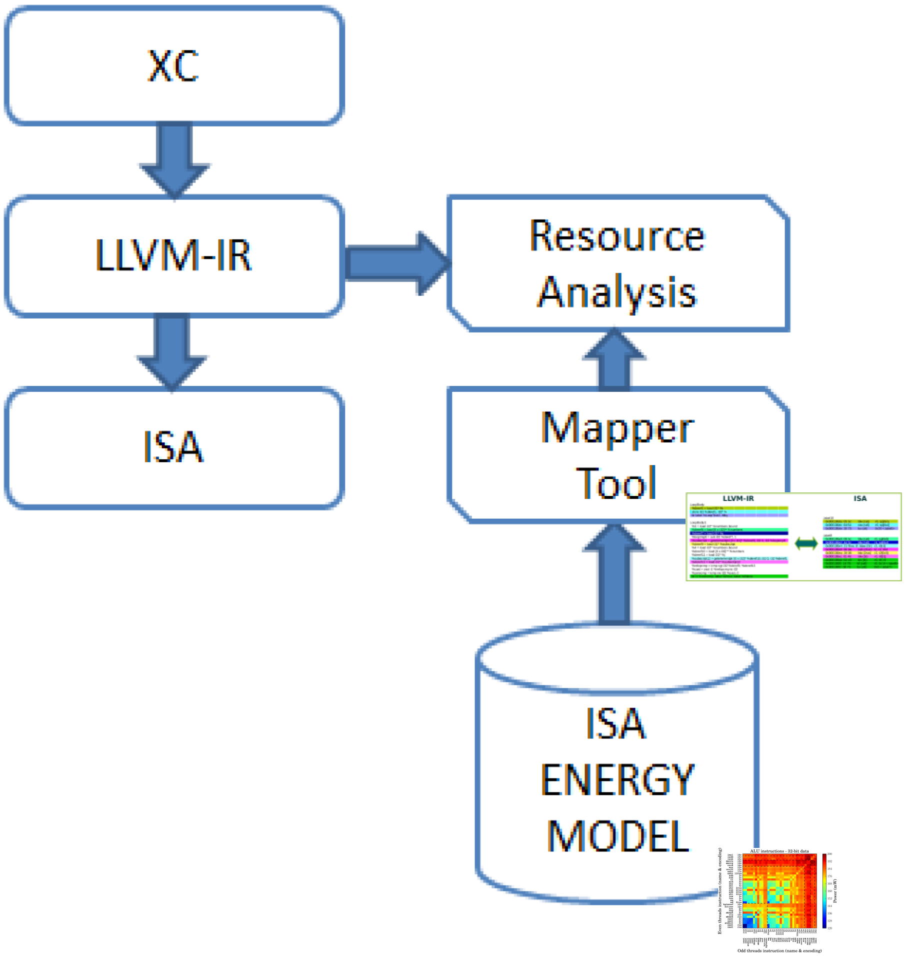
Analysis Options



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

Analysis at the LLVM-IR Level





Static Energy Usage Analysis

Original Program:

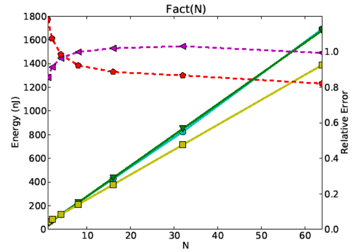
```
int fact (int x) {
  if (x<=0)a
     return 1b;
  return (x *d fact(x-1))c;
}
```

Extracted Cost Relations:

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

- 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: $C_{fact}(x) = 4563 + 7878x$ pJ

Energy Aware SWE



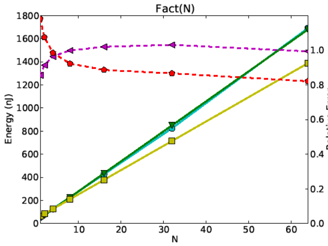

```
#pragma check energy(proc(x))<5pJ
int proc(int x) {
...
```

```
Output:

Checked 0≤x≤5⇒energy(proc(x))<5pJ
```

- A rich “assertion language” for checking power budgets
- SRA enables accurate energy estimations at early SW development phases

Energy Transparency



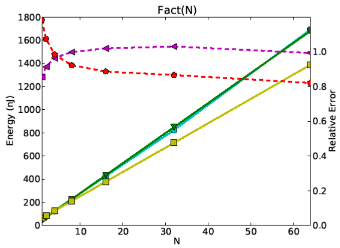
- For HW designers:
  "Power is a 1st and last order design constraint."
  [Dan Hutcheson, VLSI Research, Inc., E^{3}S Keynote 2011]

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

Scaling Power and the Future of CMOS

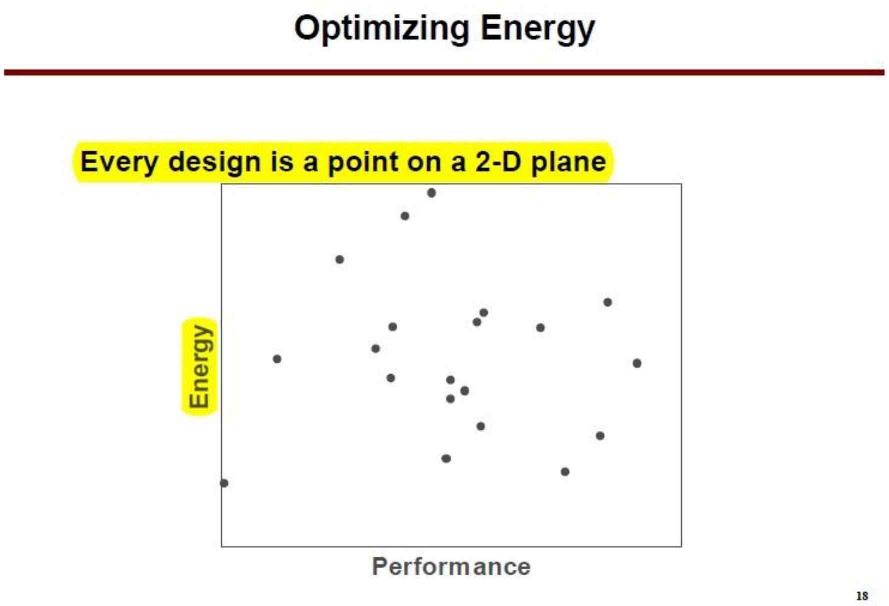
Mark Horowitz, EE/CS Stanford University

Energy Transparency

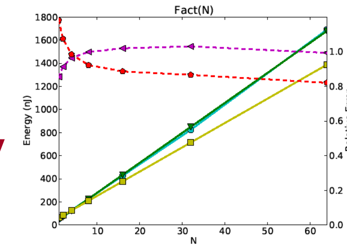


- For HW designers:
  "Power is a 1st and last order design constraint."
  [Dan Hutcheson, VLSI Research, Inc., E³S Keynote 2011]

- "Every design is a point in a 2D plane."
  [Mark Horowitz,E³S 2009]

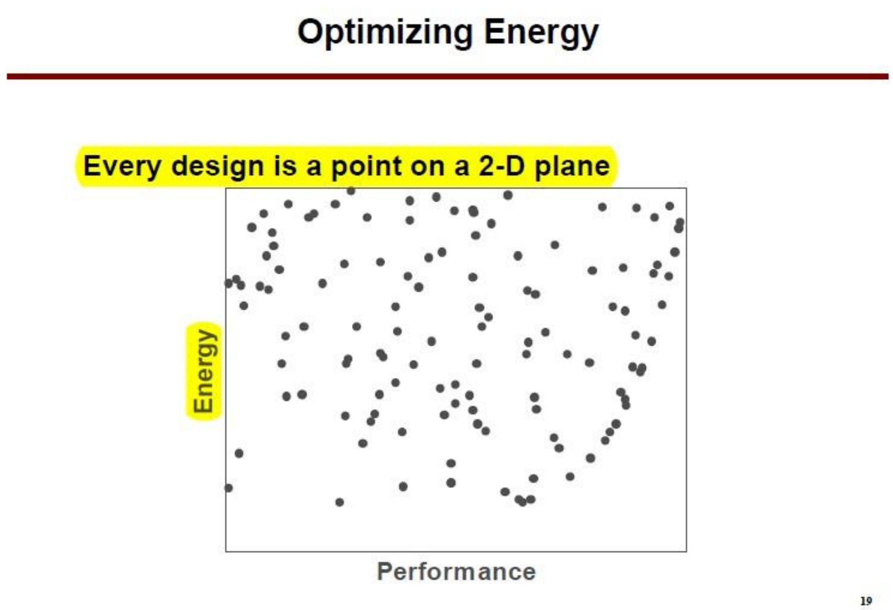


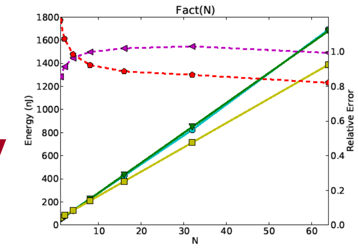
Energy Transparency



- For HW designers:
  “Power is a 1^{st} 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]

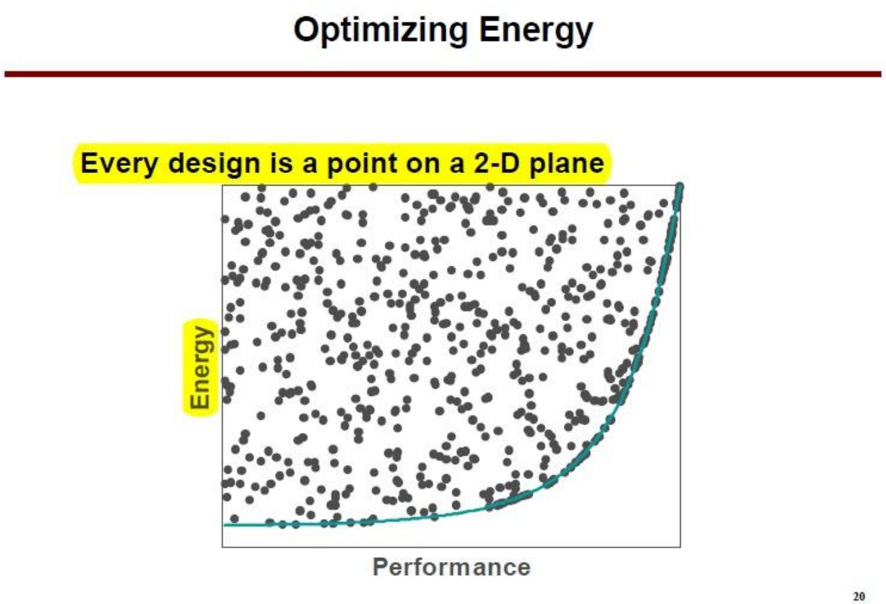




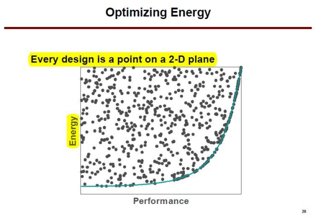
Energy Transparency

- 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 Engineers




```
in 5pJ do {...}
```

- Full Energy Transparency from HW to SW
- Location-centric programming model
- **"Cool" code**
  A cool programming competition!

Promoting energy efficiency to a 1st class SW design goal is an urgent research challenge.

A vision for the future

"Despite improved energy efficiency, **energy consumption through electronic devices will triple until 2030** because of a massive rise in overall demand."

Thank you for your attention

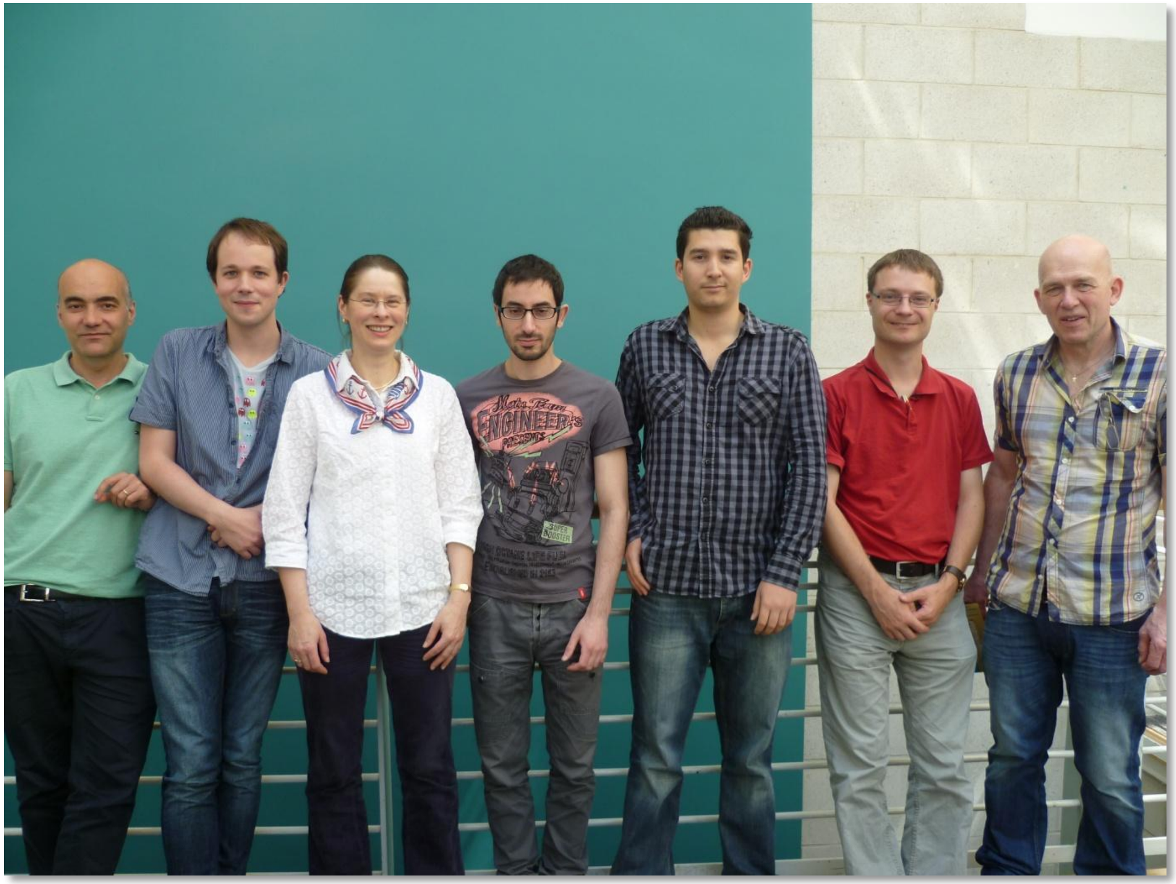
COOPERATION

institute imdea software

XMOS

The Royal Academy of Engineering

University of BRISTOL



Kerstin.Eder@bristol.ac.uk