KOCL:

Kernel-level Power Estimation for Arbitrary FPGA-SoC-accelerated OpenCL Applications

James Davis, Josh Levine, Ed Stott, Eddie Hung,
Peter Cheung and George Constantinides
Imperial College London
james.davis@imperial.ac.uk

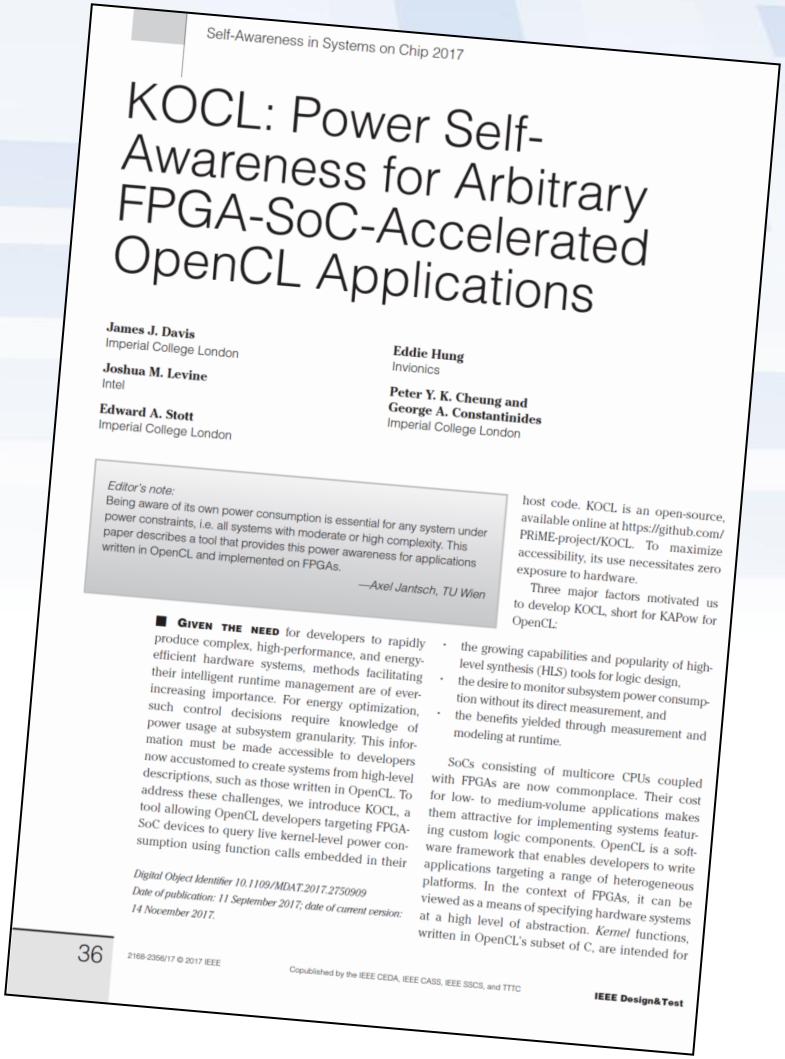
PRiME
www.prime-project.org

Executive Summary

- *KAPow* for Open**CL**
  - ‘**K**’ounting **A**ctivity for **Pow**er Estimation
- Hardware/software framework providing **kernel-level power estimates** for **OpenCL applications** running on **Altera FPGAs**
- Trains, adapts online with real workload
- Up to ±5mW accuracy
- Fully automated
- Minimalist API
- Open source
  - `https://github.com/PRiME-project/KOCL`

Shameless Self Promotion

Introduced in
IEEE D&T 34(6)



KOCL: Power Self-Awareness for Arbitrary FPGA-SoC-Accelerated OpenCL Applications

James J. Davis
Imperial College London

Joshua M. Levine
Intel

Edward A. Stott
Imperial College London

Eddie Hung
Invionics

Peter Y. K. Cheung and George A. Constantinides
Imperial College London

Editor's note:
Being aware of its own power consumption is essential for any system under power constraints, i.e. all systems with moderate or high complexity. This paper describes a tool that provides this power awareness for applications written in OpenCL and implemented on FPGAs.

—Axel Jantsch, TU Wien

■ **Given the need** for developers to rapidly produce complex, high-performance, and energy-efficient hardware systems, methods facilitating their intelligent runtime management are of ever-increasing importance. For energy optimization, such control decisions require knowledge of power usage at subsystem granularity. This information must be made accessible to developers now accustomed to create systems from high-level descriptions, such as those written in OpenCL. To address these challenges, we introduce KOCL, a tool allowing OpenCL developers targeting FPGA-SoC devices to query live kernel-level power consumption using function calls embedded in their host code. KOCL is an open-source, available online at https://github.com/PRiME-project/KOCL. To maximize accessibility, its use necessitates zero exposure to hardware.

Three major factors motivated us to develop KOCL, short for KAPow for OpenCL:

- the growing capabilities and popularity of high-level synthesis (HLS) tools for logic design,
- the desire to monitor subsystem power consumption without its direct measurement, and
- the benefits yielded through measurement and modeling at runtime.

SoCs consisting of multicore CPUs coupled with FPGAs are now commonplace. Their cost for low- to medium-volume applications makes them attractive for implementing systems featuring custom logic components. OpenCL is a software framework that enables developers to write applications targeting a range of heterogeneous platforms. In the context of FPGAs, it can be viewed as a means of specifying hardware systems at a high level of abstraction. *Kernel* functions, written in OpenCL's subset of C, are intended for

Digital Object Identifier 10.1109/MDAT.2017.2750909
Date of publication: 11 September 2017; date of current version: 14 November 2017.

Copublished by the IEEE CEDA, IEEE CASS, IEEE SSCS, and TTTC

Use Cases

- Hardware prototyping, design iteration
- Adaptive system deployment
  - Power-aware kernel selection
  - Fine-grained DVFS, clock gating, …
- Fault, malware detection
- Billing
- …

KAPow

- Hardware/software framework providing **power breakdowns** for **arbitrary FPGA-based systems** at **user-specified granularity**

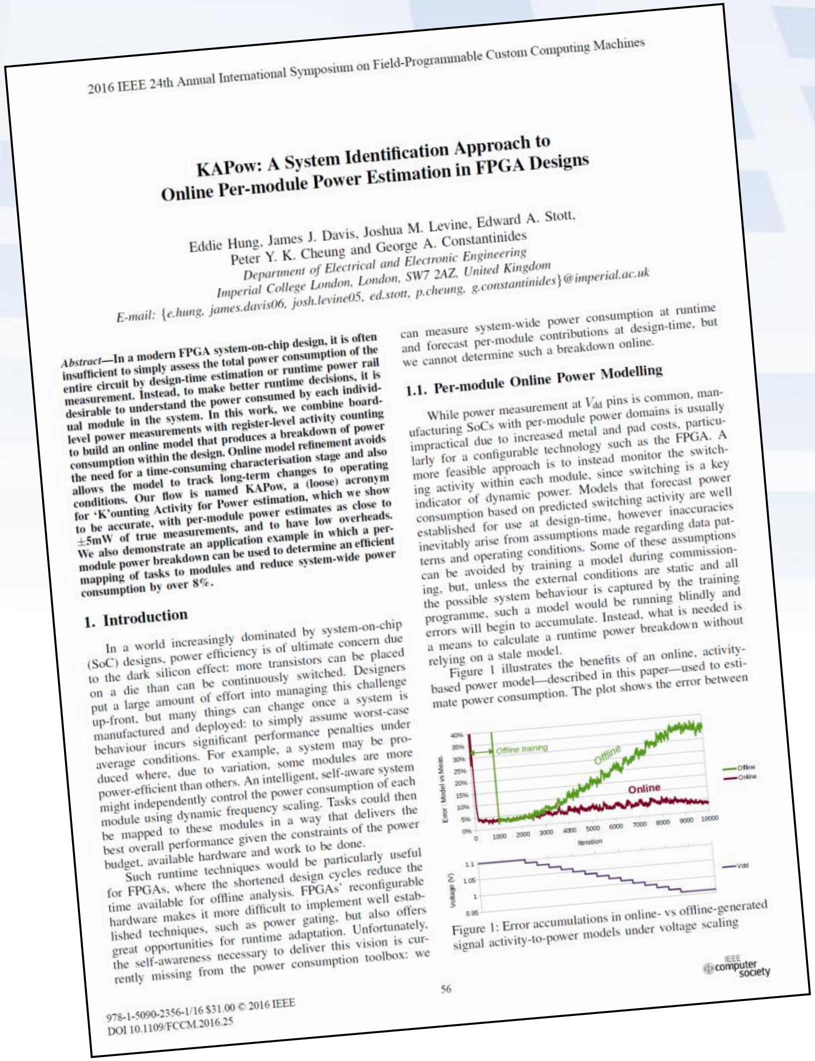
KAPow

- Hardware/software framework providing **power breakdowns** for **arbitrary FPGA-based systems** at **user-specified granularity**

- Monitoring of switching activities
  - Power-indicative signals selected
- Online modelling
  - Compensates for changes in environment, workload
- System power measurements split by module

KAPow: Further Reading

Introduced at
IEEE FCCM'16
(Best Paper)



2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines

KAPow: A System Identification Approach to Online Per-module Power Estimation in FPGA Designs

Eddie Hung, James J. Davis, Joshua M. Levine, Edward A. Stott,
Peter Y. K. Cheung and George A. Constantinides
Department of Electrical and Electronic Engineering
Imperial College London, London, SW7 2AZ, United Kingdom
E-mail: {e.hung, james.davis06, josh.levine05, ed.stott, p.cheung, g.constantinides}@imperial.ac.uk

Abstract—**In a modern FPGA system-on-chip design, it is often insufficient to simply assess the total power consumption of the entire circuit by design-time estimation or runtime power rail measurement. Instead, to make better runtime decisions, it is desirable to understand the power consumed by each individual module in the system. In this work, we combine board-level power measurements with register-level activity counting to build an online model that produces a breakdown of power consumption within the design. Online model refinement avoids the need for a time-consuming characterisation stage and also allows the model to track long-term changes to operating conditions. Our flow is named KAPow, a (loose) acronym for 'K'ounting Activity for Power estimation, which we show to be accurate, with per-module power estimates as close to ±5mW of true measurements, and to have low overheads. We also demonstrate an application example in which a per-module power breakdown can be used to determine an efficient mapping of tasks to modules and reduce system-wide power consumption by over 8%.**

1. Introduction

In a world increasingly dominated by system-on-chip (SoC) designs, power efficiency is of ultimate concern due to the dark silicon effect: more transistors can be placed on a die than can be continuously switched. Designers put a large amount of effort into managing this challenge up-front, but many things can change once a system is manufactured and deployed: to simply assume worst-case behaviour incurs significant performance penalties under average conditions. For example, a system may be produced where, due to variation, some modules are more power-efficient than others. An intelligent, self-aware system might independently control the power consumption of each module using dynamic frequency scaling. Tasks could then be mapped to these modules in a way that delivers the best overall performance given the constraints of the power budget, available hardware and work to be done.

Such runtime techniques would be particularly useful for FPGAs, where the shortened design cycles reduce the time available for offline analysis. FPGAs' reconfigurable hardware makes it more difficult to implement well established techniques, such as power gating, but also offers great opportunities for runtime adaptation. Unfortunately, the self-awareness necessary to deliver this vision is currently missing from the power consumption toolbox: we can measure system-wide power consumption at runtime and forecast per-module contributions at design-time, but we cannot determine such a breakdown online.

1.1. Per-module Online Power Modelling

While power measurement at V_{dd} pins is common, manufacturing SoCs with per-module power domains is usually impractical due to increased metal and pad costs, particularly for a configurable technology such as the FPGA. A more feasible approach is to instead monitor the switching activity within each module, since switching is a key indicator of dynamic power. Models that forecast power consumption based on predicted switching activity are well established for use at design-time, however inaccuracies inevitably arise from assumptions made regarding data patterns and operating conditions. Some of these assumptions can be avoided by training a model during commissioning, but, unless the external conditions are static and all the possible system behaviour is captured by the training programme, such a model would be running blindly and errors will begin to accumulate. Instead, what is needed is a means to calculate a runtime power breakdown without relying on a stale model.

Figure 1 illustrates the benefits of an online, activity-based power model—described in this paper—used to estimate power consumption. The plot shows the error between

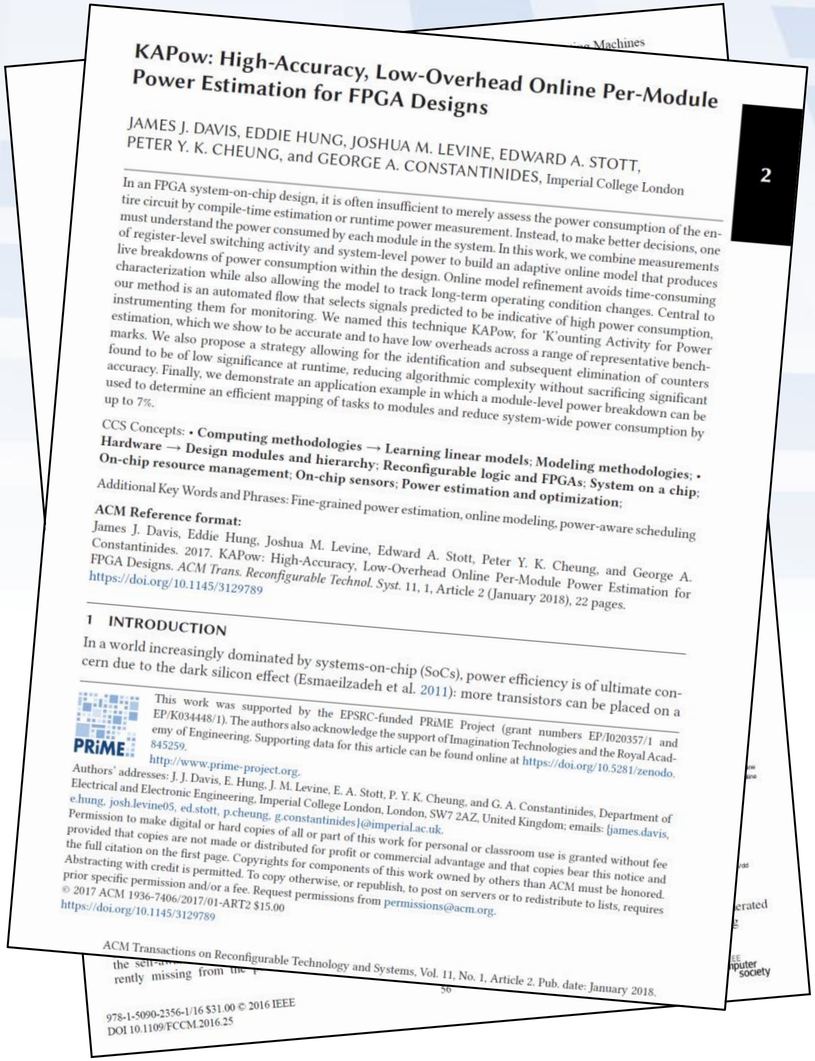
Figure 1: Error accumulations in online- vs offline-generated signal activity-to-power models under voltage scaling

978-1-5090-2356-1/16 $31.00 © 2016 IEEE
DOI 10.1109/FCCM.2016.25

IEEE computer society

KAPow: Further Reading

Introduced at
IEEE FCCM'16
(Best Paper)



Machines

KAPow: High-Accuracy, Low-Overhead Online Per-Module Power Estimation for FPGA Designs

JAMES J. DAVIS, EDDIE HUNG, JOSHUA M. LEVINE, EDWARD A. STOTT, PETER Y. K. CHEUNG, and GEORGE A. CONSTANTINIDES, Imperial College London

2

In an FPGA system-on-chip design, it is often insufficient to merely assess the power consumption of the entire circuit by compile-time estimation or runtime power measurement. Instead, to make better decisions, one must understand the power consumed by each module in the system. In this work, we combine measurements of register-level switching activity and system-level power to build an adaptive online model that produces live breakdowns of power consumption within the design. Online model refinement avoids time-consuming characterization while also allowing the model to track long-term operating condition changes. Central to our method is an automated flow that selects signals predicted to be indicative of high power consumption, instrumenting them for monitoring. We named this technique KAPow, for 'K'ounting Activity for Power estimation, which we show to be accurate and to have low overheads across a range of representative benchmarks. We also propose a strategy allowing for the identification and subsequent elimination of counters found to be of low significance at runtime, reducing algorithmic complexity without sacrificing significant accuracy. Finally, we demonstrate an application example in which a module-level power breakdown can be used to determine an efficient mapping of tasks to modules and reduce system-wide power consumption by up to 7%.

CCS Concepts: • **Computing methodologies → Learning linear models; Modeling methodologies;** • **Hardware → Design modules and hierarchy; Reconfigurable logic and FPGAs; System on a chip; On-chip resource management; On-chip sensors; Power estimation and optimization;**

Additional Key Words and Phrases: Fine-grained power estimation, online modeling, power-aware scheduling

ACM Reference format:
James J. Davis, Eddie Hung, Joshua M. Levine, Edward A. Stott, Peter Y. K. Cheung, and George A. Constantinides. 2017. KAPow: High-Accuracy, Low-Overhead Online Per-Module Power Estimation for FPGA Designs. *ACM Trans. Reconfigurable Technol. Syst.* 11, 1, Article 2 (January 2018), 22 pages.
https://doi.org/10.1145/3129789

1 INTRODUCTION

In a world increasingly dominated by systems-on-chip (SoCs), power efficiency is of ultimate concern due to the dark silicon effect (Esmaeilzadeh et al. 2011): more transistors can be placed on a

This work was supported by the EPSRC-funded PRiME Project (grant numbers EP/K034448/1 and EP/I020357/1). The authors also acknowledge the support of Imagination Technologies and the Royal Academy of Engineering. Supporting data for this article can be found online at https://doi.org/10.5281/zenodo.845259.
http://www.prime-project.org.

Authors' addresses: J. J. Davis, E. Hung, J. M. Levine, E. A. Stott, P. Y. K. Cheung, and G. A. Constantinides, Department of Electrical and Electronic Engineering, Imperial College London, London, SW7 2AZ, United Kingdom; emails: {james.davis, e.hung, josh.levine05, ed.stott, p.cheung, g.constantinides}@imperial.ac.uk.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2017 ACM 1936-7406/2017/01-ART2 $15.00
https://doi.org/10.1145/3129789

ACM Transactions on Reconfigurable Technology and Systems, Vol. 11, No. 1, Article 2. Pub. date: January 2018.

erated

the self-
rently missing from the

puter
society

56

978-1-5090-2356-1/16 $31.00 © 2016 IEEE
DOI 10.1109/FCCM.2016.25

Extended in
ACM TRETS
11(1)

Motivation

- Have existing fine-grained power estimation framework…
- … but it requires HDL expertise
- “Hardware is hard” – can we hide it?

Motivation

- Have existing fine-grained power estimation framework…
- … but it requires HDL expertise
- “Hardware is hard” – can we hide it?
- Aims:
  - Generality
  - Minimal user effort
  - Transparency
  - Low overheads

OpenCL for FPGAs

- Adopted as input language by Altera, Xilinx
- Front-ends to existing vendor tools
  - High-level synthesis
  - System integration
  - Mapping, placement, routing, …
- Kernel code compiled offline…
  - 1 kernel = 1 hardware accelerator
- … and stitched to supporting infrastructure
  - Global memory interfacing
  - Launching kernels

Developer Burden: Hardware

Developer Burden: Hardware

- Before:

```
./aoc <.cl file> --board <board name>
```

- After:

```
./koc <.cl file> --board <board name>
```

Developer Burden: Hardware

- Before:

```
./aoc <.cl file> --board <board name>
```

- After:

```
./koc <.cl file> --board <board name>
```

- Optional flags:

– `kernels`	Choose a subset of kernels to monitor
– `kapow_n`	Control fidelity of measurements
– `kapow_w`	

Developer Burden: Software

- Initialise:

```
#include "KOCL.h"
KOCL_init(float <update period>);
```

  - Controls reactiveness of power model

- Use:

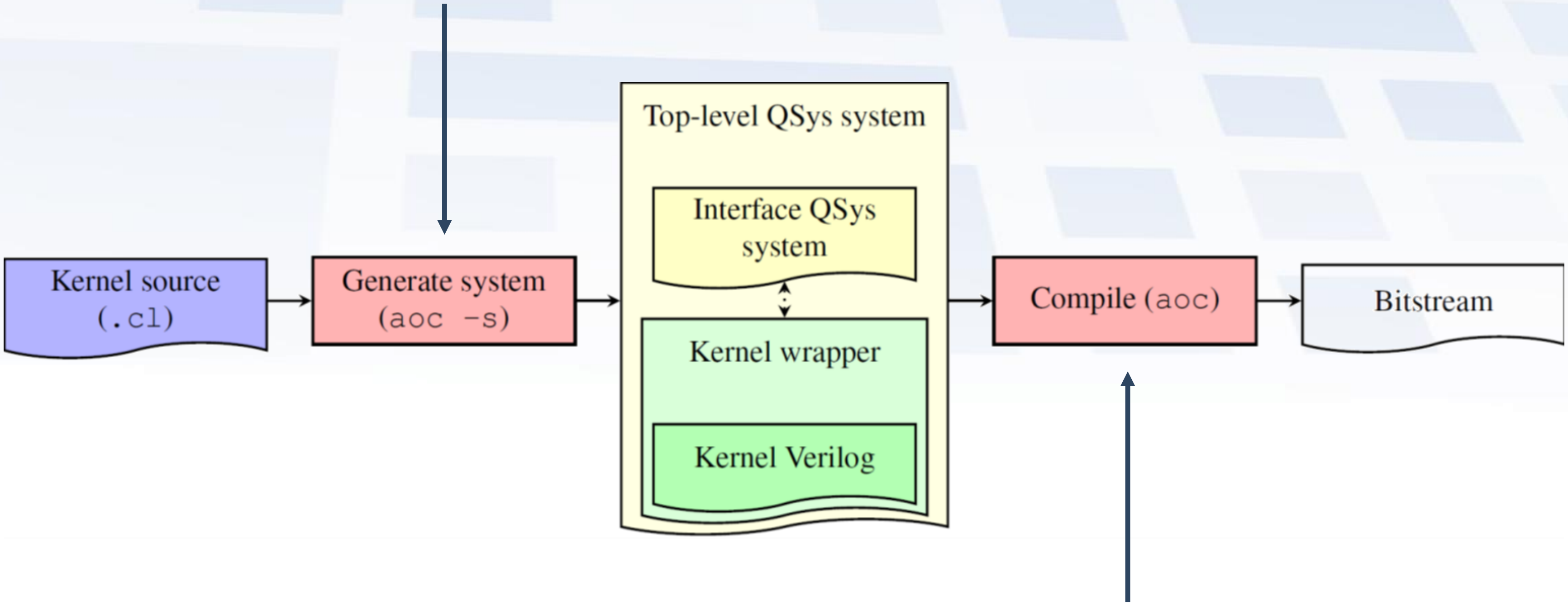
```
KOCL_built();
KOCL_get(char* <kernel name>);
KOCL_get("static");
```

- Clean up:

```
KOCL_del();
```

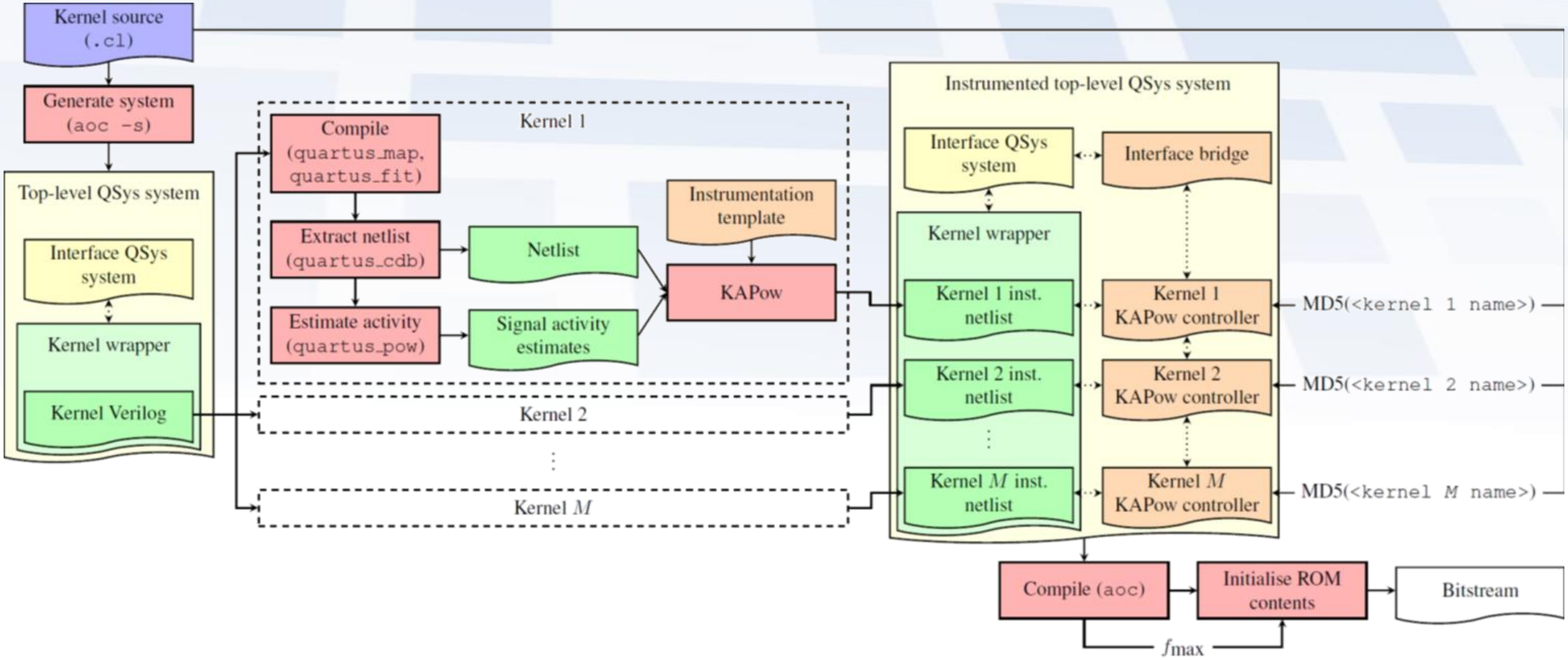
Vanilla Tool Flow

High-level synthesis,
system integration



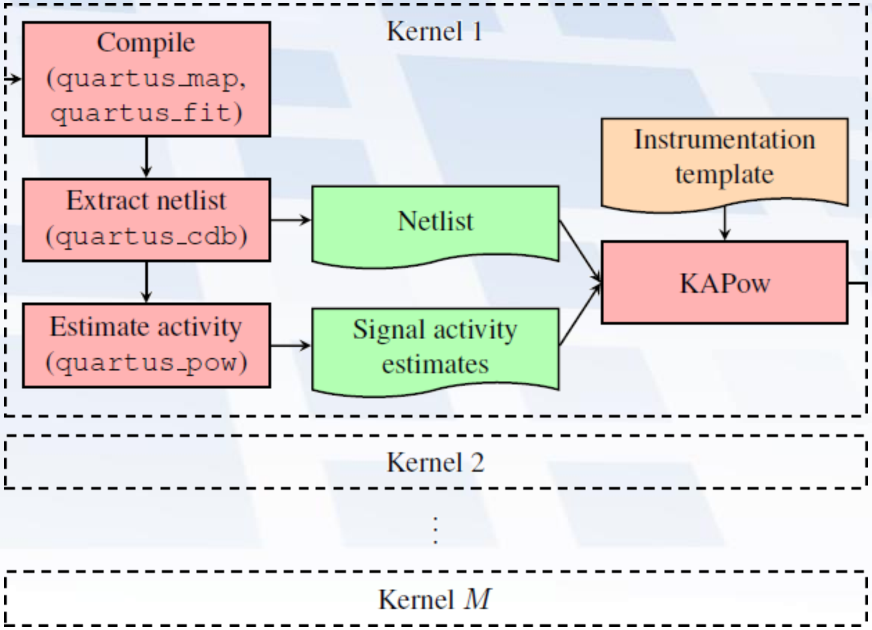
Mapping, placement, routing, …

KOCL Tool Flow



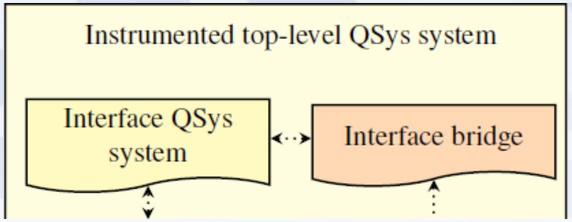
KOCL Tool Flow: HDL

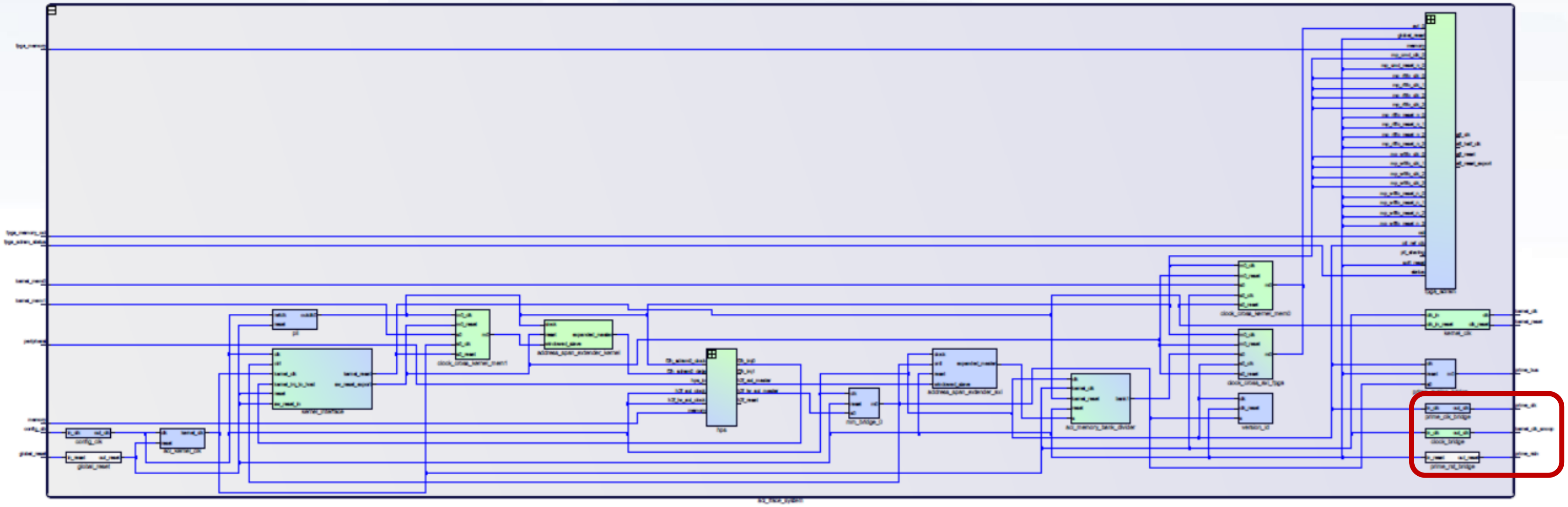
- Per kernel:
  - Compile → netlist
    - Specifies use of FPGA resources
  - Perform power simulation to obtain switching estimates
    - Fast
    - No user input
  - Augment N most-switching signals with W-bit activity counters
  - Substitute for original HDL


Kernel 1
Compile (quartus_map, quartus_fit)
Extract netlist (quartus_cdb)
Estimate activity (quartus_pow)
Netlist
Signal activity estimates
Instrumentation template
KAPow
Kernel 2
⋮
Kernel M

KOCL Tool Flow: Interfacing 1

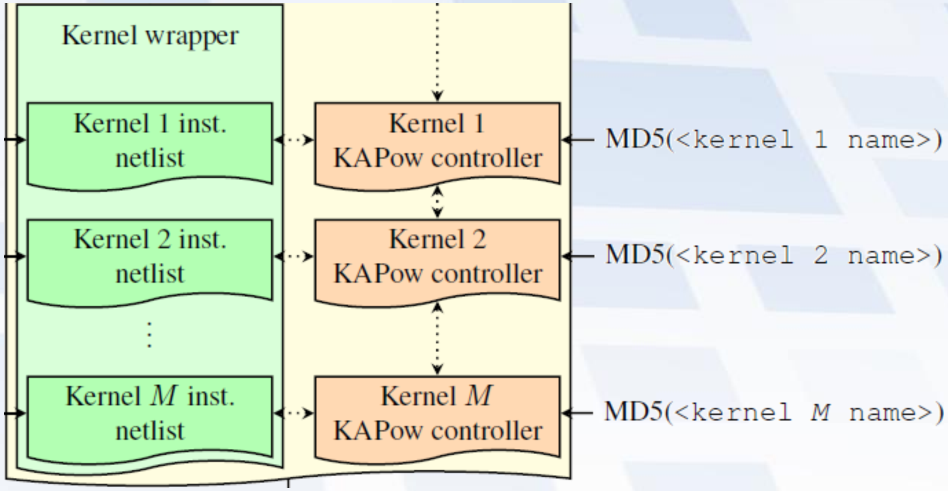
- Expose busses to allow counter control, readback





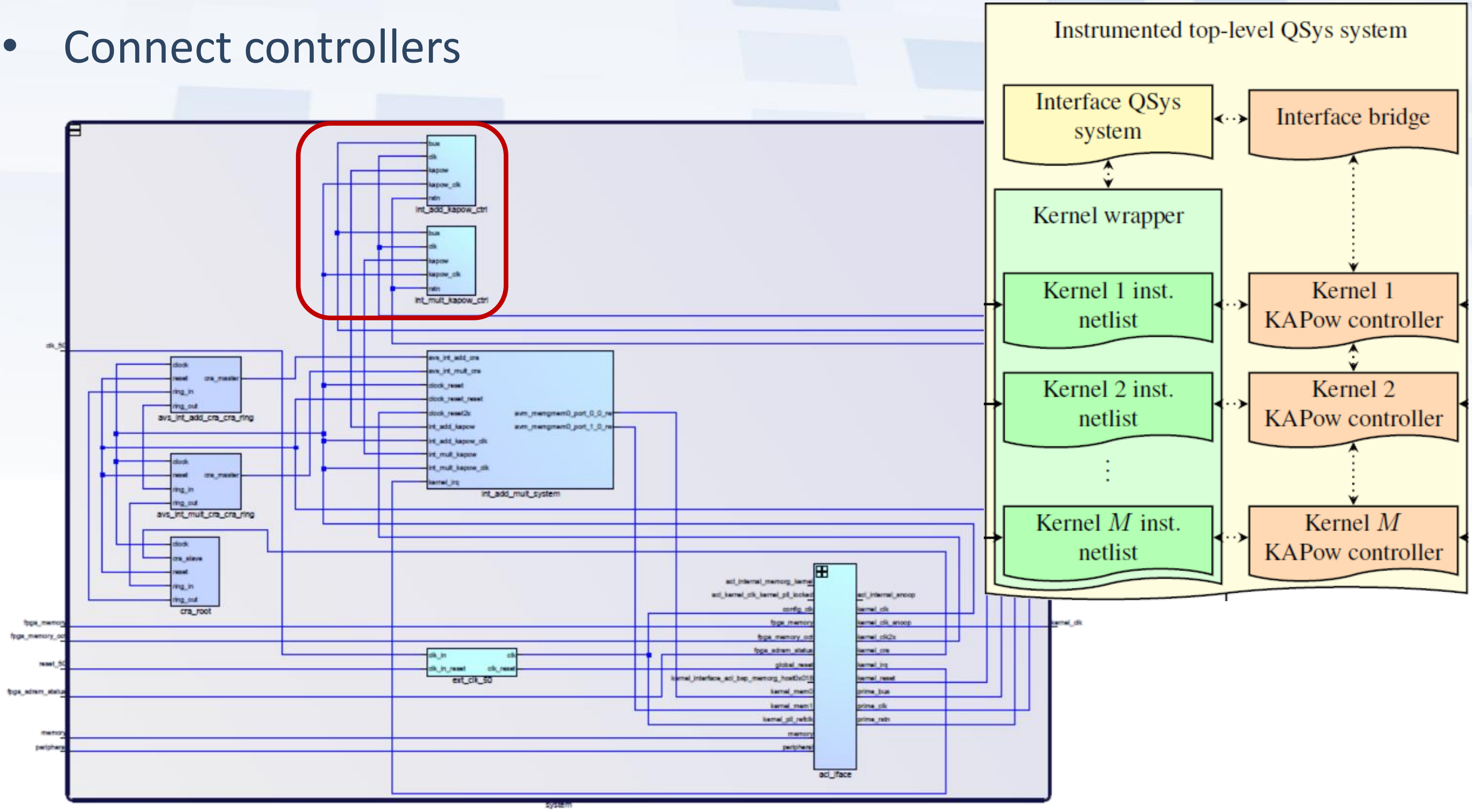
KOCL Tool Flow: Control

- Per kernel:
  - Add controller
  - Connect to counters in netlist
  - Parameterise with hash of kernel's name

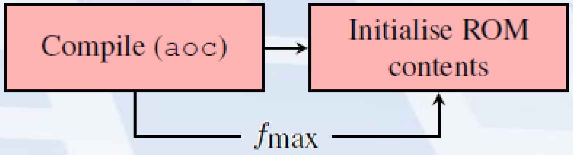


KOCL Tool Flow: Interfacing 2

- Connect controllers



KOCL Tool Flow: TTL



- Need to determine optimal measurement period
  - Too small: low dynamic range
  - Too large: potential overflow
- Read $f_{\max}$ from compilation report
- Given $f_{\max}$, W, calculate TTL
- Apply via controller ROMs

KOCL Software

- Launched by, runs alongside host code
- Python w/Numpy, C API

KOCL Software

- Launched by, runs alongside host code
- Python w/Numpy, C API

- Three threads:
  - **Model**
    - Talks to hardware
    - Performs power modelling
  - Interface
    - Responds to host code requests
  - Messenger
    - Model-interface communication

KOCL Software: Model

- Initialisation:
  - Establish kernel names from bitstream
  - Discover controllers in hardware
  - Match to kernel names using hashes
  - Read parameters (N, W) from controllers
  - Construct model

KOCL Software: Model

- Initialisation:
  - Establish kernel names from bitstream
  - Discover controllers in hardware
  - Match to kernel names using hashes
  - Read parameters (N, W) from controllers
  - Construct model

- Every `update_period`:
  - Get activity, system power measurements
  - Update model
  - Pass power breakdown to messenger

Results

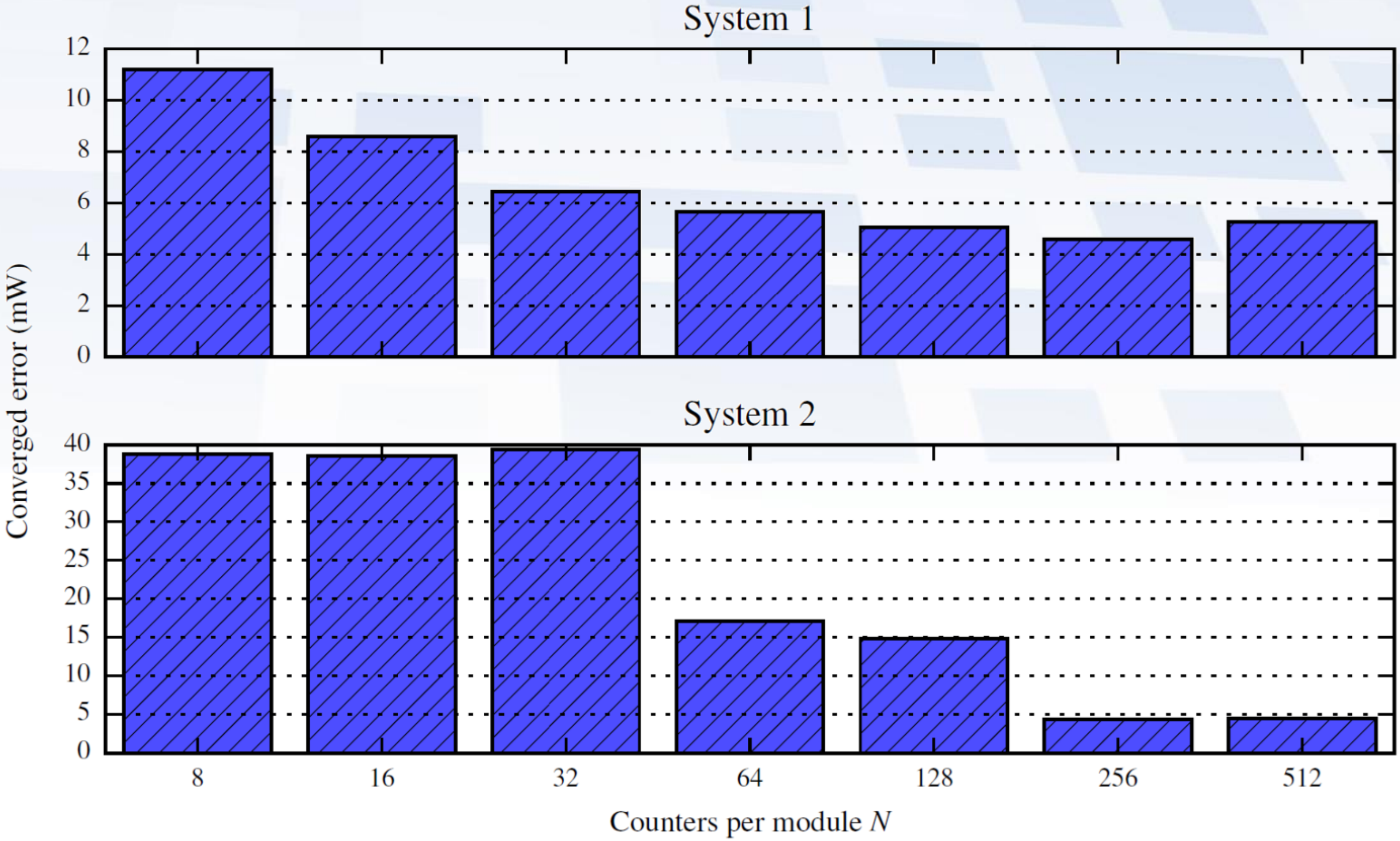
- Things of interest:
  - **Accuracy**
    - Estimate vs measurement
  - Compilation time overhead
  - Area overhead
  - Power overhead
  - Max. model update rate

Results

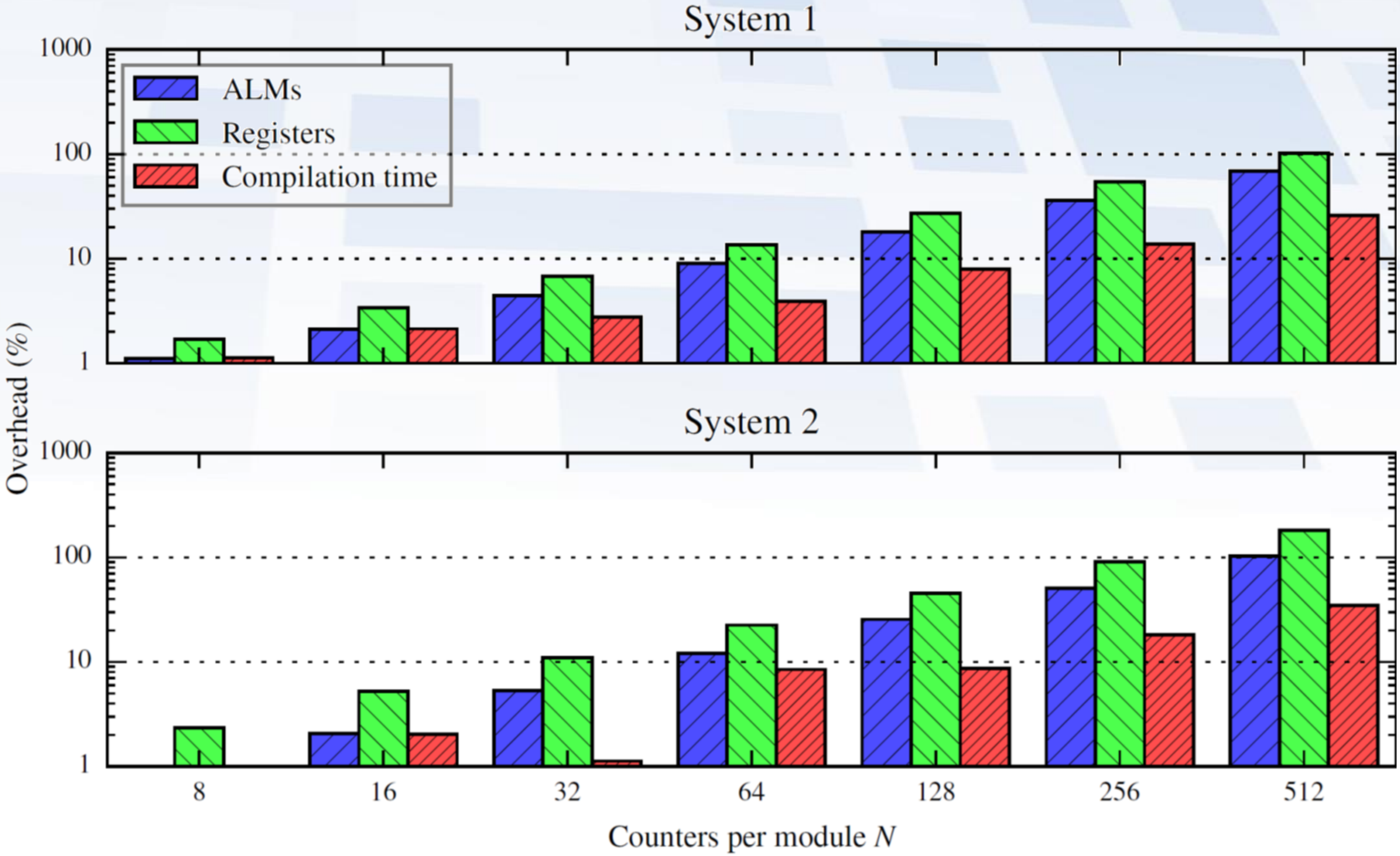
- Things of interest:
  - **Accuracy**
    - Estimate vs measurement
  - Compilation time overhead
  - Area overhead
  - Power overhead
  - Max. model update rate

- Particularly dependent on choice of N
- Found $W = 9$ generally best accuracy-overhead compromise

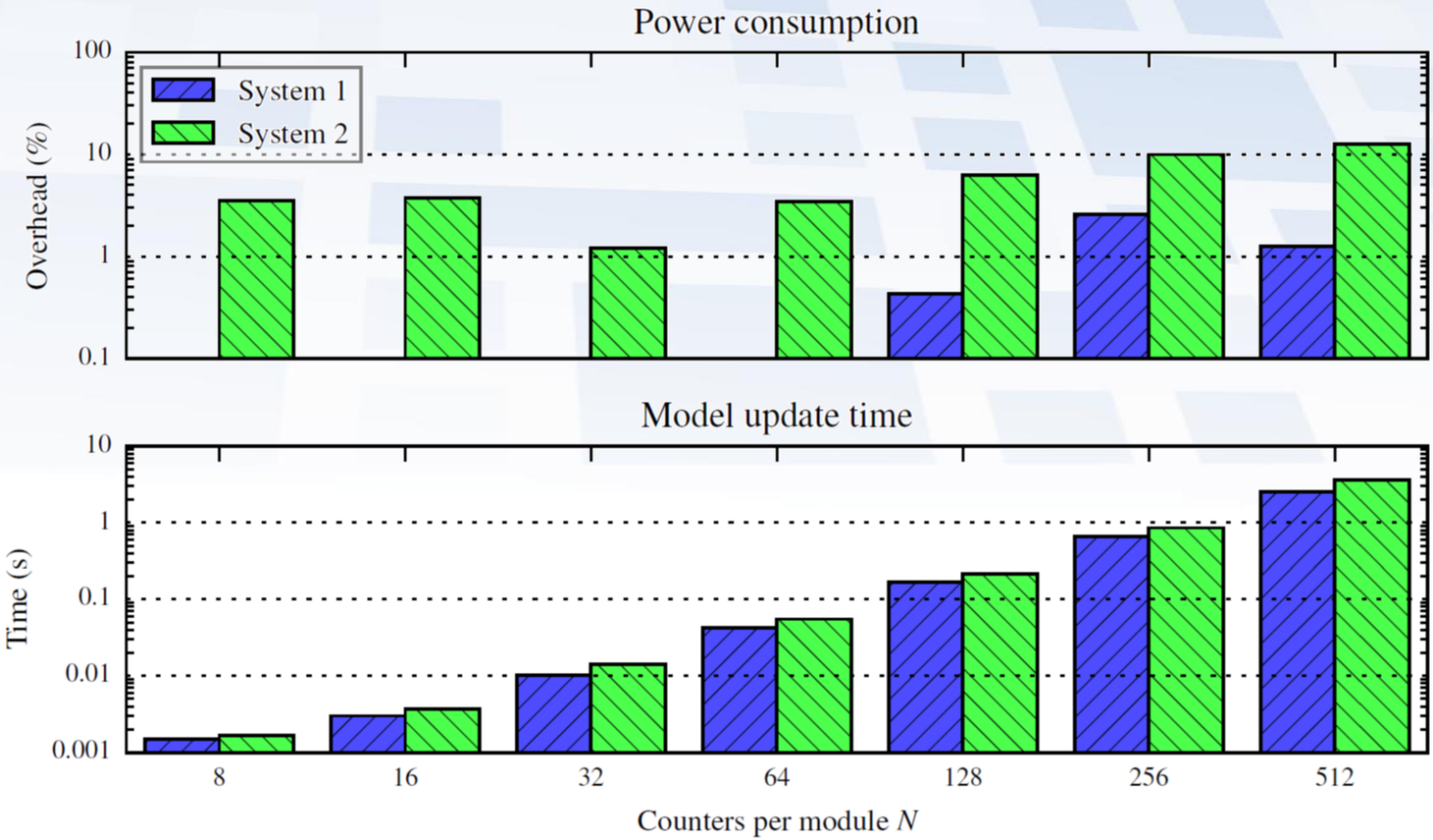
Accuracy


System 1
12
10
8
6
4
2
0
Converged error (mW)
System 2
40
35
30
25
20
15
10
5
0
8
16
32
64
128
256
512
Counters per module *N*

Compilation Overheads



Runtime Overheads

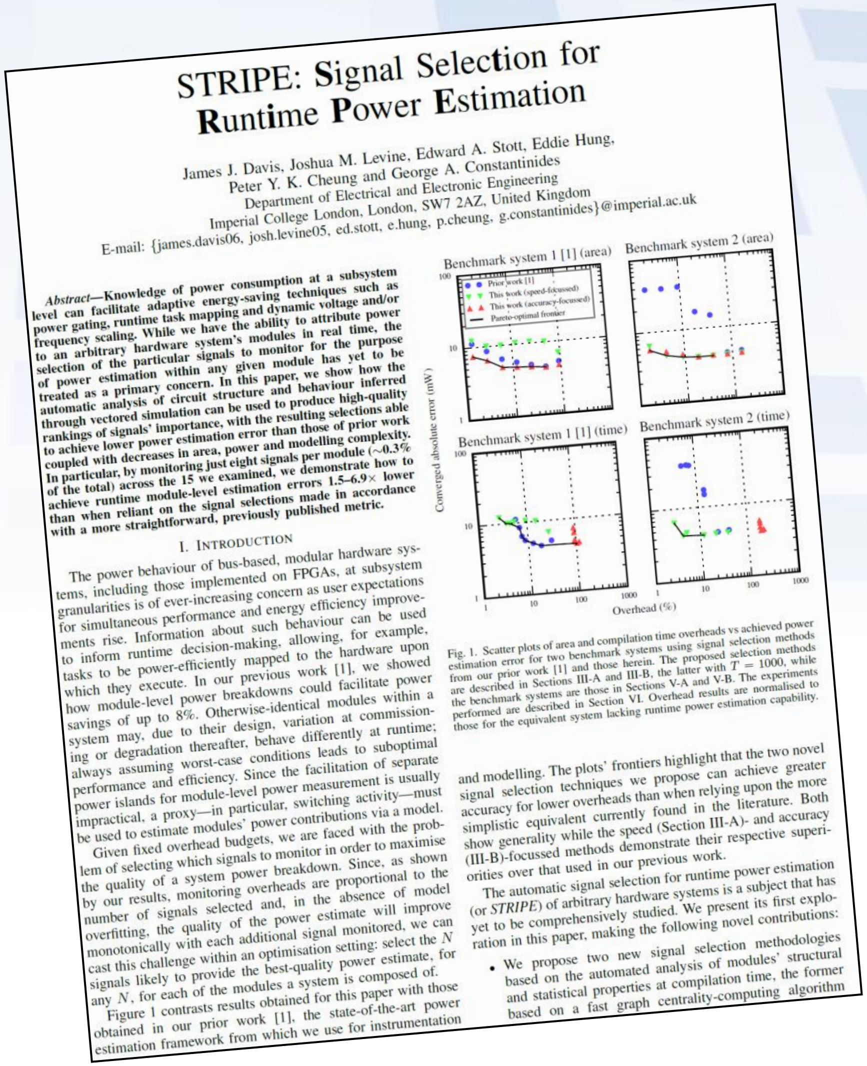


Further Work

- Improved signal selection
- Incorporation of macro modelling
- Use for system-level control
- More devices, vendors
- Similar tools for monitoring performance, reliability

Preliminary Improvements

Signal selection improved in FPL'17



STRIPE: Signal Selection for Runtime Power Estimation

James J. Davis, Joshua M. Levine, Edward A. Stott, Eddie Hung,
Peter Y. K. Cheung and George A. Constantinides
Department of Electrical and Electronic Engineering
Imperial College London, London, SW7 2AZ, United Kingdom
E-mail: {james.davis06, josh.levine05, ed.stott, e.hung, p.cheung, g.constantinides}@imperial.ac.uk

***Abstract*—Knowledge of power consumption at a subsystem level can facilitate adaptive energy-saving techniques such as power gating, runtime task mapping and dynamic voltage and/or frequency scaling. While we have the ability to attribute power to an arbitrary hardware system's modules in real time, the selection of the particular signals to monitor for the purpose of power estimation within any given module has yet to be treated as a primary concern. In this paper, we show how the automatic analysis of circuit structure and behaviour inferred through vectored simulation can be used to produce high-quality rankings of signals' importance, with the resulting selections able to achieve lower power estimation error than those of prior work coupled with decreases in area, power and modelling complexity. In particular, by monitoring just eight signals per module (∼0.3% of the total) across the 15 we examined, we demonstrate how to achieve runtime module-level estimation errors 1.5–6.9× lower than when reliant on the signal selections made in accordance with a more straightforward, previously published metric.**

I. Introduction

The power behaviour of bus-based, modular hardware systems, including those implemented on FPGAs, at subsystem granularities is of ever-increasing concern as user expectations for simultaneous performance and energy efficiency improvements rise. Information about such behaviour can be used to inform runtime decision-making, allowing, for example, tasks to be power-efficiently mapped to the hardware upon which they execute. In our previous work [1], we showed how module-level power breakdowns could facilitate power savings of up to 8%. Otherwise-identical modules within a system may, due to their design, variation at commissioning or degradation thereafter, behave differently at runtime; always assuming worst-case conditions leads to suboptimal performance and efficiency. Since the facilitation of separate power islands for module-level power measurement is usually impractical, a proxy—in particular, switching activity—must be used to estimate modules' power contributions via a model.

Given fixed overhead budgets, we are faced with the problem of selecting which signals to monitor in order to maximise the quality of a system power breakdown. Since, as shown by our results, monitoring overheads are proportional to the number of signals selected and, in the absence of model overfitting, the quality of the power estimate will improve monotonically with each additional signal monitored, we can cast this challenge within an optimisation setting: select the N signals likely to provide the best-quality power estimate, for any N, for each of the modules a system is composed of.

Figure 1 contrasts results obtained for this paper with those obtained in our prior work [1], the state-of-the-art power estimation framework from which we use for instrumentation and modelling. The plots' frontiers highlight that the two novel signal selection techniques we propose can achieve greater accuracy for lower overheads than when relying upon the more simplistic equivalent currently found in the literature. Both show generality while the speed (Section III-A)- and accuracy (III-B)-focussed methods demonstrate their respective superiorities over that used in our previous work.

Benchmark system 1 [1] (area)
Benchmark system 2 (area)
Benchmark system 1 [1] (time)
Benchmark system 2 (time)
● Prior work [1]
▼ This work (speed-focussed)
▲ This work (accuracy-focussed)
Pareto-optimal frontier
Converged absolute error (mW)
100
10
1
1
10
100
1000
Overhead (%)

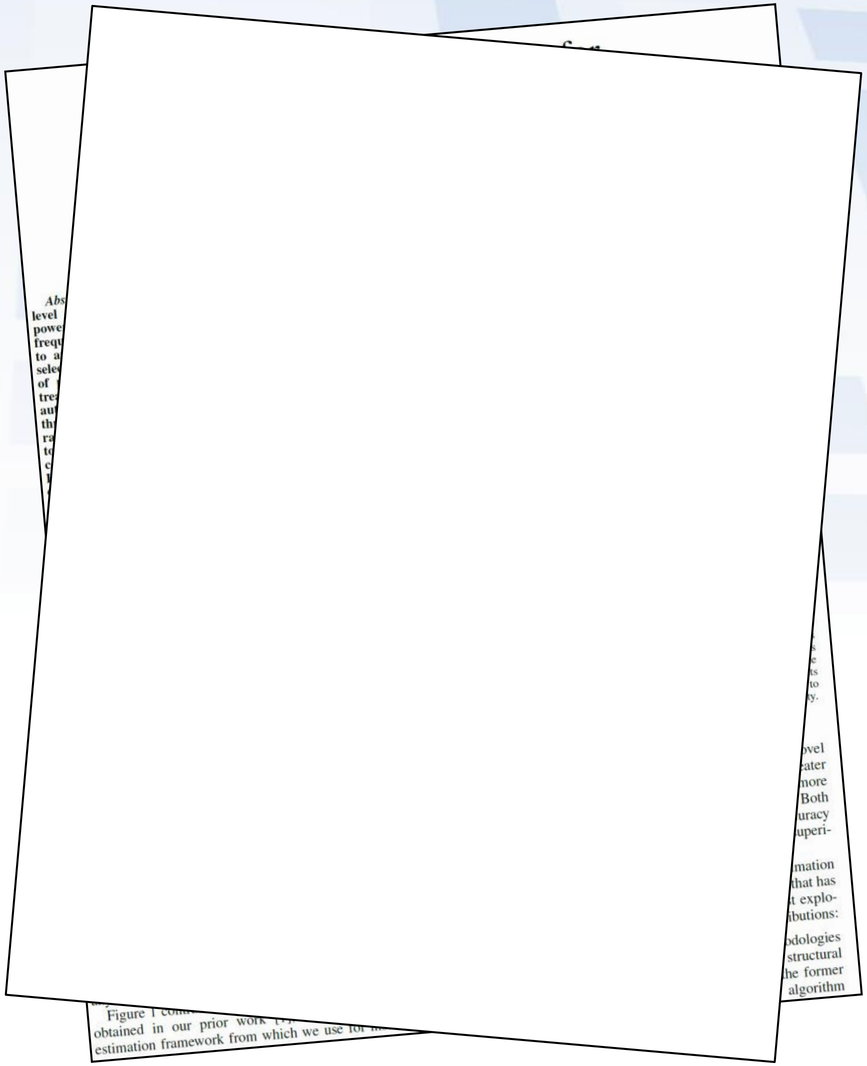
Fig. 1. Scatter plots of area and compilation time overheads vs achieved power estimation error for two benchmark systems using signal selection methods from our prior work [1] and those herein. The proposed selection methods are described in Sections III-A and III-B, the latter with $T = 1000$, while the benchmark systems are those in Sections V-A and V-B. The experiments performed are described in Section VI. Overhead results are normalised to those for the equivalent system lacking runtime power estimation capability.

The automatic signal selection for runtime power estimation (or *STRIPE*) of arbitrary hardware systems is a subject that has yet to be comprehensively studied. We present its first exploration in this paper, making the following novel contributions:

- We propose two new signal selection methodologies based on the automated analysis of modules' structural and statistical properties at compilation time, the former based on a fast graph centrality-computing algorithm

Preliminary Improvements

Signal selection improved in FPL'17



Extension in the works…

Summary

- Framework providing **kernel-level power estimates** of **arbitrary OpenCL systems** executing on **Altera FPGAs** to **host code**
- Easy to use
  - No hardware exposure
- ≥ order-of-magnitude accuracy improvement vs simulation
- Remains under active development

Summary

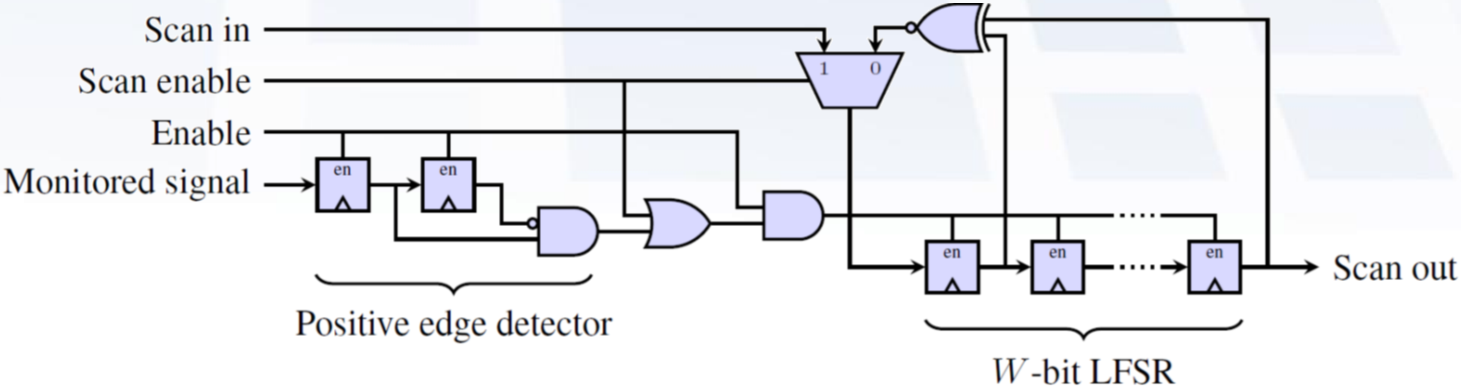
- Framework providing **kernel-level power estimates** of **arbitrary OpenCL systems** executing on **Altera FPGAs** to **host code**
- Easy to use
  - No hardware exposure
- ≥ order-of-magnitude accuracy improvement vs simulation
- Remains under active development

- Open source
  - `https://github.com/PRiME-project/KOCL`
  - Plug-and-play Linux image, demo apps included
- Please use and provide feedback!

Backup

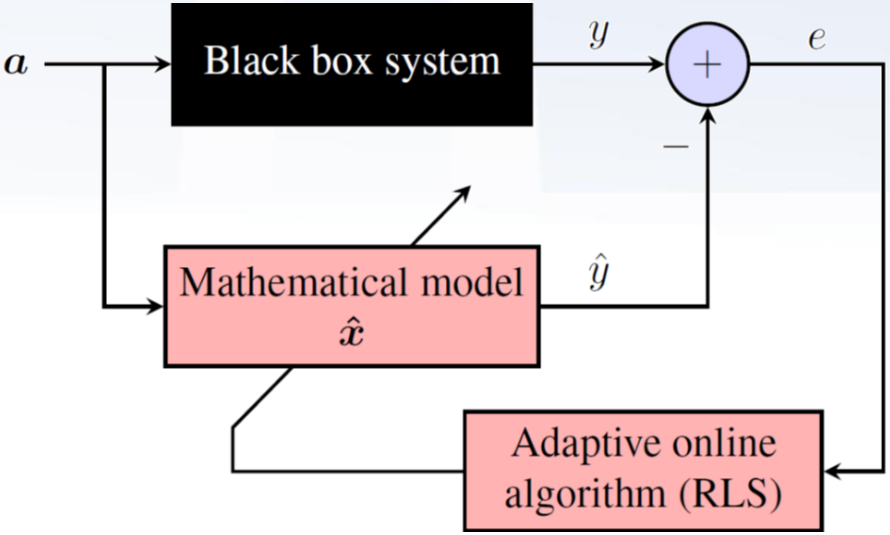
KAPow: Monitoring

- Modules analysed to identify power-indicative signals
- Lightweight activity counters transparently inserted



KAPow: Modelling

- Activities + system power → module-level power
- Online training, refinement
  - Adapts to changes in voltage, temperature, workload, noise, …



$\boldsymbol{a}$:	activity counts
y:	measured power
$\hat{y}$:	estimated power
e:	error
$\hat{\boldsymbol{x}}$:	model coefficients