Towards Real-Power Computing
From Power-Efficient Electronics to Power-Centric Design

DESSERT 2018, Kyiv, 27 May 2018

Prof Alex Yakovlev
Dr Rishad Shafik
Newcastle University

Energy drives logic
μSystems Research Group

Newcastle University
Talk Overview

- Introduction: some facts, rationale and motivation
- Many shades of real-power computing
- Design for real-power systems
  - Power-compute co-design
  - Run-time adaptation
- Some initial investigation and results
- Ongoing research and conclusions
Trillions of ubiquitous systems (sensors, probes, monitors, actuators, controllers) are being deployed to operate in myriad of places (organisation, human, body part, household, offices, pets) using harvested energy or micro-batteries.

Swarm of devices – Future of ICT

- Trillions+: Energy constrained
- Billions+: Energy constrained
- Billions: Energy and performance constrained
- Millions+: Power and performance constrained
- Millions: Power constrained
- Thousands+: Power constrained

- 0.1μW
- 1W
- 5W
- 10W
- 50-100W
- 10kW

Kyiv DESSERT’18
Battery Technology Scaling

- **Alessandro Volta** invented battery in 1799; humans started using mobile electrical devices from late 1800s with *enormous* size batteries

- Battery capacities have been the **slowest** over the years since we started mobile electronics

![Graph showing improvements in various technologies since 1990](image)

**Paradiso and Starner, IEEE Pervasive Computing, 2005**

**Munan et al. Nano Energy, 2017**

_Battery weights, disposal and maintenance had been limiting many applications_
How are we managing?

Battery

Software

Hardware

User

Application Software

Operating System

Hardware
How are we managing?

Battery

Software

Hardware

Dynamic voltage/frequency scaling (DVFS)

Power gating circuits and systems

Intelligent design-/run-time task mapping and resource allocation

Mixed-signal designs
Energy is out there!

**RF Energy harvesting**

Maxwell: $E^2 / Z_0$

- E: Electrical field
- $Z_0$: Radiation resistance

- A few hundreds of mW
- Available round the clock

**Solar energy harvesting**

Solar to electrical

- 20%-30% efficiency

- A few hundreds of mW (area dependent)
- Time and weather dependent

**Vibration energy harvesting**

- uWs to mWs
- Subject dependent

**Thermal energy harvesting**

- Several mWs
- Subject dependent

**Implantable harvesting**

- Nanoribbons rectifier
- Piezoelectrics

**Multi-modal energy harvesting**

- e.g.: Thermal and solar
- Several mWs
- Subject dependent

Kyiv DESSERT’18
Trillions of ubiquitous systems are happening soon

But how are we going to make these autonomous?

How are we going to go scale productivity so fast?
Now, in Maxwell's theory there is the potential energy of the displacement produced in the dielectric parts by the electric force, and there is the kinetic and magnetic energy of the magnetic force in all parts of the field, including the conducting parts. They are supposed to be set up by the current in the wire. We reverse this; the current in the wire is set up by the energy transmitted through the medium around it. The energy of the electric machinery is transmitting energy from the battery to the wire. It is definite in amount, and the rate of transmission of energy (total) is also definite in amount.

Oliver Heaviside (1879)
What computing we need...

• Incoming power is the “first citizen”: definite incoming power should formulate power budget for delivering “somewhat definite” computation

• Autonomy
  – Battery-less ("no" re-charging or maintenance)
  – Self-adaptive
    * power-proportional computing (less power -> best possible functionality)
    * survivability: dynamically retain computation when power is lost
    * fire-and-forge

• Productivity
  – All components strictly integrated based around power supply/source
    * power delivery, power/energy models, software and hardware
  – Ensure high design integrity of all hardware and software components
Real-Power Computing

Power-compute Co-Design
  Programming model and requirement annotations
  Computation quality versus energy trade-offs

External Interface

Hardware Tasks

Software Tasks

Run-Time Adaptation
  Survivability against limited energy and/or power
  Run-time task partition, map and power scheduling

Power/Energy Transparency
  Power/energy/performance models at various abstractions;
  worst-case power consumption

Hardware Architecture
  Power-proportional hardware with dynamic retention
  Power delivery, on-chip sensing and control

Energy supply with unprecedented variations
Real-Power Computing

- **Typical requirements**
  - Performance

Minimise power/energy, while meeting performance

- **Typical requirements**
  - Power/Energy/Quality
  - Performance

Elastically control computation quality based on power/energy and performance

Traditional low-power computing systems do not automatically provide autonomy

Traditional computing: - power is a result of computation - no survival instinct

Real-power computing: - strict power control based on avail. energy - built-in survival instinct

Power-constrained computing ensures autonomy and energy-effectiveness
Hard and Soft Real-Power

**Hard real-power computing**
- No battery/no storage
- Extensive power-compute co-design needed

---

**Soft real-power computing**
- With energy storage
- Power-compute co-design
- + run-time adaptation
Power-compute co-design

Formulate design-time power/energy scheduling policies
Run-time adaptation

High-level Annotations (req.)  Run-time Support  Energy Transparency Models

Controls:
HW: Approx. HW, retention
System: DVFS, mapping
SW: algorithms

On-chip sensing:
Power, accuracy (MSEs), performance

Facilitate run-time survivability decisions
- Schedule HW/SW
- DVFS
- Dynamic retention

Kyiv DESSERT'18
Energy transparency model

Source: K. Eder, Bristol University, 2017
Programming model example

\[ f(a, X_f) \quad \text{// } a \text{ is a vector of data parameters, } X_f \text{ is a resource parameter} \]
\[ \text{requires } P(a, X_f), \quad \text{// Precondition on program state and resource budget} \]
\[ \text{ensures } Q(a, X_f) \quad \text{// Postcondition specifying requirements on program state in relation to resource budget} \]

\{
\[ \text{split } X_f \text{ as } X_g \oplus X_h \text{ respecting } \phi(X_g, X_h, a); \quad \text{// Incoming resource } X_f \text{ is divided into separate sub-resources } X_g \text{ and } X_h, \text{ with the resource division respecting a programmer-defined constraint } \phi \]
\[ \ldots \]
\[ g(a, X_g); \]
\[ h(a, X_h); \]
\[ \ldots \]
\}

Source: A. Donaldson, Imperial College, 2017
Case Study

Power-adaptive processing

Original (8x8) multiplier

SDLC-2
(8x8) Sig.driven logic compress. 2

SDLC-3
(8x8) Sig.driven logic compress. 3

SDLC-4
(8x8) Sig.driven logic compress. 4

Power

p

0.55p

0.3p

0.13p

mse

mse=0

mse=0.55

mse=0.89

mse=1.67

Qiqieh et al. DATE2017
Qiqieh et al. SIPS 2017
Layered power-computation activities

Layered mode of functionality

at lower available power

Reduced power

Low computation capacity

High computation capacity

active hardware/software resources

at higher available power

Reduced power
Simulated Power Source (with variations)

Formulate Power budgets (deducing losses)

Find the best scheduling that gives the optimised quality, while meeting power budget

Power scheduling policies

Energy/quality models

Multiplier selection over time

OpenCL/C++ (simulated retention registers)

Kyiv DESSERT'18
Case Study
Power-adaptive processing

Multiplier selection over different power budgets

Kyiv DESSERT'18
Newcastle’s fully self-timed CPU: Intel 8051 in 130nm CMOS (2013)

• 0.89V to 1.5V: full capability mode.

• 0.74V to 0.89V: at 0.89V the RAM starts to fail, so the chip operates using

• 0.22V to 0.74V: at 0.74V the program counter starts to fail, however the control logic synthesised using the CPOG model continues to operate correctly down to 0.22V

• 67 MIPS at 1.2 V.

• ~2700 instructions per second at 0.25V.
Energy Efficiency
(measurements on real silicon)

Energy Per Instruction = Current * Voltage * Latency

Minimum Energy point
Asynchronous (self-timed) logic can provide completion detection and thus reduce the interval of leakage to minimum, thereby doing nothing well!

Source: Akgun et al, ASYNC’10
Beyond self-powered systems

What’s in it for other types of systems

• **Power-constrained computing**
  (models/hardware/software) is unprecedented

  • Embedded computing can provide user-facing power/energy-budgeting options
    – Elastic computational capability
    – Extend operating lifetime significantly

  • Line-powered systems can be governed by strict power or energy budget policies (from use-facing interfaces)
    – Essentially savings £££££s
    – More £££s, more quality computation, otherwise less quality
Interesting research questions related to vulnerability

• How will power-constrained/driven computing affect vulnerability to:
  – Faults (e.g. due to low supply voltage)
  – Attacks (e.g. due to imprecise computing)

• How will the great autonomy of devices (e.g. in terms of extracting power from environment, state retention and self-learning capabilities) affect the vulnerability of the whole IoT?
On prodigality of self-powered computing

- Inspired by nature, where the key principle of LEAST ACTION is not parsimony but prodigality

Explore maximum computational possibilities
WITHIN
a given energy budget

(Hypothesis of T. Toffoli, “Action, or the fungibility of computation,” in Feynman and Computation. 1999)
Thank you!

See:

http://www.ncl.ac.uk/engineering/research/eee/microsystems/

http://async.org.uk/

For research projects, publications, staff profile’s profiles, industrial, academic and international collaborations, software tools, chip tapeouts, academic opportunities ...

More details in the IEEE Transactions paper “Real-Power Computing”