Energy-Driven Computing:
Rethinking the Design of Energy Harvesting Systems

10 February 2020
UNIVERSITY OF SOUTHAMPTON

University of Southampton
• ~25,000 students
• Top 100 universities worldwide (QS‘19)
• Founding member of UK’s Russell Group

School of Electronics and Computer Science
• ~2,000 students
• ~250 PhD research students
• ~100 academics/faculty
• Top 3 in UK for EEE
• 14 research groups/centres
POWERING THE IOT

• We’ve got batteries!
  – So what’s the problem?

• More things = batteries/wires/people
  – Pervasive/IoT/ubiquitous

• Fit-and-forget/maintenance issues
  – Smart homes/grid/metering

• Weight vs volume vs lifetime
  – e.g. Wearables
Highly variable supply + variable consumption!
ENERGY-NEUTRAL COMPUTING

Schematic

Block Diagram

Stored Energy

Power Supply → Power Conversion → Energy Storage → Power Conversion → Computation (Load)

\[ \int_{(n-1)T}^{nT} P_h(t) dt = \int_{(n-1)T}^{nT} P_L(t) dt \]

ENERGY-NEUTRAL

\[ V_{CC} \geq V_{min}, \forall t \]

If energy-neutral rule is violated, system fails
ENERGY-DRIVEN COMPUTING

• What’s wrong with energy storage and complexity?

• Emerging applications demanding small dimensions, volumes, weight, cost, etc
INTERMITTENT COMPUTING

Compute operates across power outages

Static Approaches
• Application instrumented with checkpoints

Task-based Approaches
• Application divided into set of small tasks
• State saving during transitions

Reactive Approaches
• Save state on power failure
• No roll-back -> consistent memory
• Less overhead (runtime & mem)
**REACTIVE IC: hibernus**

- Make only a single (but always a single) snapshot per supply ‘failure’
  - Removes wasted snapshots
  - Ensures always makes valid snapshot
- Make it as late as possible
  - Avoids re-executing code
  - Maximises execution time

---

REACTIVE IC: *hibernus*

- Controlled source (signal generator)
- Real energy harvesting sources

Time overheads reduced by 75-100%
Energy overheads reduced by 50-80%

REACTIVE IC: *hibernus*

Developed a range of optimisations/extensions
- Self calibration for runtime hibernate threshold optimisation (*hibernus++*)
- Adaptive restore based on EH properties (*hibernus++*)
- Hibernation and restore of external peripheral state (*RESTOP*)
- Support for Arm Mbed
- Support for communication and mesh networking
- Applications (cycle computer, fitness monitor, etc)
- Fine-grained power adaption (*PowerNeutrality*)
- Efficient state retention (Selective Policies, *ManagedState*)
- Tooling for system design (*ENSPECT, FUSED, Device Sizing*)

[www.transient.ecs.soton.ac.uk](http://www.transient.ecs.soton.ac.uk)
EFFICIENT STATE RETENTION

- CPU
  - Execute app
  - Volatile Data
    - CPU registers
  - Volatile Data
    - SRAM

Diagram showing voltage (V) over time (T) with critical points:
- V_{CC}
- V_{ON}
- V_{RESTORE}
- V_{SUSPEND}
STATE RETENTION POLICIES

Allocated State (symmetric/asymmetric NVMs)

Multiple Blocks (asymmetric NVMs)

NVMs with no erase cost

NVMs with erase cost

NVMs with no erase cost

“MANAGED STATE”

A lightweight memory management software-layer that loads pages of memory on demand, and saves only modified pages.

MANAGED STATE: USAGE AND OVERHEADS

```c
#include <managed_state.h>
define MMDATA __attribute__((section(".mmdata")))

uint8_t data [2048] MMDATA;
uint8_t key [16] = ...;
...
for (uint8_t *ptr = data; ptr < end; ptr += 16) {
    ms_acquire(ptr, 16, MM_READWRITE);
    aes_encrypt(ptr, key);
    ms_release(ptr, 16);
}
```

The overhead of ManagedState for different operations is as follows:

- AES: 1.05x
- CRC32: 1.17x
- Matmul: 6.81x
- Matmul tiled: 1.48x

The diagram illustrates the interaction between the ManagedState and the App and NVM, showing operations like Acquire, Release, Read, Write, Load/Save, and Restore.
MANAGED STATE: RESULTS

Suspend + Restore time in relation to on-time

Completion time relative to AllocatedState. Powered by voltage pulses of variable width

CLOSED-LOOP PERFORMANCE & ENERGY SIMULATION

Developing energy-driven computing systems is difficult

- Operation driven directly by availability of energy
- Introspection/debug inevitably affects stored/harvested energy (and therefore operation)
- Typical embedded systems development can’t cope with a DUT that frequently powers off
- Repeatability of EH is problematic

FUSED (Full-system Simulation of Energy-Driven Computers)

- Open source full-system simulator for energy-driven computers
- SystemC for digital and mixed-signal simulation, modelling microcontroller and mixed-signal circuitry
- Models power supply/consumption and execution in a closed loop, modelling the interaction between.
- Accurate power model obtained correlates microarchitectural events with real power measurements
- Enables hardware-software codesign and design space exploration.

Download: www.arm.ecs.soton.ac.uk/technologies/fused

Fig. 9. Simulation trace of a reactive intermittent computing system powered by a 200 $\mu$A current-limited power supply. The top traces show the logic levels of GPIO pins indicating the operation of the device, and the lower traces show the microcontroller supply voltage ($v_{cc}$), the storage capacitor voltage ($v_{cap}$) and the current draw ($i_{cc}$).

Download: www.arm.ecs.soton.ac.uk/technologies/fused

Fig. 10. Completion time of AES encryption when running intermittently, powered by a current-limited power source.

Fig. 11. Full-system energy consumption when running AES encryption intermittently, powered by current-limited power source. The energy consumption is divided into stacked bars for the external circuitry (ext.), hardware boot (HW-Boot), and the operational phases restore, compute and suspend.
POWER NEUTRAL COMPUTING

• In Energy-Neutral computing, \( \int_{(n-1)T}^{nT} P_h(t) dt = \int_{(n-1)T}^{nT} P_c(t) dt \) over a ‘large’ \( T \)

• In Power-Neutral computing, \( P_c(t) \approx P_h(t) \)

• Power consumption is modulated, eg through:
  – Core frequency and/or voltage
  – Power gating processor elements

• We can approximate power-neutral behaviour if \( V_C(t) \approx k \), \( \forall t \)
POWERM NEUTRAL COMPUTING

- What happens if $V_C$ remains constant ($V_C(t) \approx k$, $\forall t$)?

- MPPT approaches are needed as $V_C(t) \neq V_{H_MPP}(t)$, $\forall t$

- Traditional EH systems decouple source and load using MPPT

- ‘Software-only’ MPPT can modulate $k$, i.e. $V_C(t) = V_{H_MPP}(t)$
MAXIMUM SYSTEM EFFICIENCY

MPPT maximises $\eta_{eh}$; we maximise $\eta_{sys}$

$$\eta_{sys} = \frac{P_{out}}{P_{max}} = \left( \frac{P_{out}}{P_{in}} \right) \cdot \left( \frac{P_{in}}{P_{max}} \right) = \eta_{eh} \eta_{vr}$$

Estimating $V_{cc}$ and $P_c$, we can identify MPP
MAXIMUM SYSTEM EFFICIENCY

MPPT maximises $\eta_{eh}$; we maximise $\eta_{sys}$

$$\eta_{sys} = \frac{P_{out}}{P_{max}} = \left( \frac{P_{out}}{P_{in}} \right) \cdot \left( \frac{P_{in}}{P_{max}} \right) = \eta_{eh} \eta_{lr}$$

Estimating $V_{cc}$ and $P_c$, we can identify MPP
EXPERIMENTAL RESULTS
Operating from a PV Energy Harvester

- Software-Based MPPT
EXPERIMENTAL RESULTS
Operating from a PV Energy Harvester

- Power-Neutral Behaviour

- Application Forward Progress

<table>
<thead>
<tr>
<th>Performance Scaling Technique</th>
<th>Number of FFTs per second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Approach [16]</td>
<td>1.07</td>
</tr>
<tr>
<td>MCU Power-Neutral Approach [17]</td>
<td>1.65</td>
</tr>
<tr>
<td>Momentum</td>
<td>1.83</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Performance Scaling Technique</th>
<th>Billions of Instructions Per Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux Powersave</td>
<td>0.69</td>
</tr>
<tr>
<td>Momentum</td>
<td>1.27</td>
</tr>
</tbody>
</table>
CONCLUSIONS

• We need to rethink the way that we design self-powered systems
• Lots of existing approaches to help with this...
• ...but lots of challenges
YOUR QUESTIONS

Professor Geoff Merrett | gvm@ecs.soton.ac.uk
Head of Centre for IoT and Pervasive Systems