INVITED: Energy Harvesting and Transient Computing: A Paradigm Shift for Embedded Systems?

Geoff V. Merrett
Department of Electronics and Computer Science
University of Southampton
gvm@ecs.soton.ac.uk

ABSTRACT

Embedded systems powered from time-varying energy harvesting sources traditionally operate using the principles of energy-neutral computing: over a certain period of time, the energy that they consume equals the energy that they harvest. This has the significant advantage of making the system ‘look like’ a battery-powered system, yet typically results in large, complex and expensive power conversion circuitry and introduces numerous challenges including fast and reliable cold-start. In recent years, the concept of transient computing has emerged to challenge this traditional approach, whereby low-power embedded systems are enabled to operate as usual while energy is available but, after loss of supply, can quickly regain state and continue where they left off. This paper provides a summary of these different approaches.

CCS Concepts

• Computer systems organization → Embedded and Cyber-Physical Systems

Keywords

Transient computing; energy harvesting; embedded systems.

1. INTRODUCTION

The powering of electronic devices from harvested energy (energy scavenged from the surrounding environment) has been gaining increasing interest over the past decade [1]. It removes the inconvenience and cost of battery replacement or recharging or, in applications where battery replacement is impossible or inefficient, can significantly increase the system’s operational lifetime. However, powering electronic systems from harvested energy poses a number of challenges that battery-powered systems to not exhibit. First, batteries can supply comparatively high levels of instantaneous power, but have a constrained energy capacity. In contrast, energy harvesting sources offer a theoretically limitless energy supply (assuming no fatigue or degradation), but highly constrained supply of power typically in the range of µW-mW. Furthermore, this supply of power usually exhibits high spatial and temporal variation; a transient supply. Consider, for example, a photovoltaic cell on a mobile device. The harvestable power will continually vary (e.g. with time-of-day, weather, orientation, environment) and may even drop to zero (e.g. if the device is put into a bag) [2]. For an AC source, e.g. a micro wind turbine (Fig.1), further variation is provided that typically needs to be rectified and smoothed through intermediate power conversion circuitry. In recent years, a range of approaches have been proposed to accommodate this variability; this paper provides a summary.

DAC '16, June 05-09, 2016, Austin, TX, USA
© 2016 ACM. ISBN 978-1-4503-4236-0/16/06…$15.00
DOI: http://dx.doi.org/10.1145/2897937.2905011

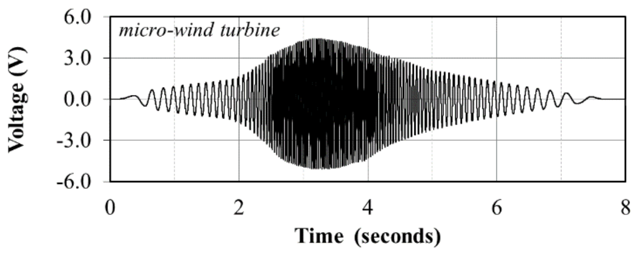

Figure 1. Typical output voltage from a micro wind turbine.

2. ENERGY-NEUTRAL COMPUTING

The traditional approach to accommodate the varying supply provided by energy harvesting is that of energy-neutral operation [3]. Here, an appropriately sized energy store (e.g. a supercapacitor) provides a ‘buffer’ between the harvester and load (Fig. 2).

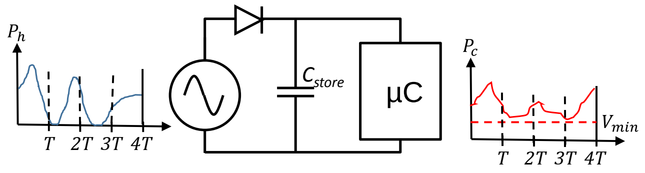

Figure 2. Principles of energy-neutral computing

Over a defined period of time (e.g. 24 hrs), the energy consumed by the load is equal to the energy harvested (Equation 1). Control of consumption is obtained through the run-time adaptation of workload intensity (e.g. reducing periodic task rates). The advantage of this is that, to the application developer, the system appears similar to a battery-powered system. Provided there is low-frequency control of workload intensity to manage consumption, the system can be considered as having a stable supply.

$$\int_{(n-1)\cdot T}^{n\cdot T} P_h(t)\,dt = \int_{(n-1)\cdot T}^{n\cdot T} P_c(t)\,dt \qquad (1)$$

However, the addition of an energy ‘buffer’ typically incurs additional cost, volume/weight, and circuit complexity (for power conversion circuitry required to manage charging and discharging).

3. TRANSIENT COMPUTING

Transient computing attempts to overcome these disadvantages, by removing the energy ‘buffer’ and instead operating directly from the harvester’s variable (or *transient*) supply. As a result, the supply can no longer be considered to be battery-like, due to significant temporal variation in available power. This creates a major issue: if the harvested power is not sufficient to supply the load, its operation will cease and, when a usable supply is restored, application execution will have to restart from the beginning.

A solution to this is to make checkpoints of the system state to non-volatile memory (NVM) and, at a later time, restore this state and continue execution. An early work in this area was Mementos [4], where checkpoints were taken regularly at strategic points during execution, in case power is lost. This incurs many unnecessary checkpoints however, adding time and energy overhead.

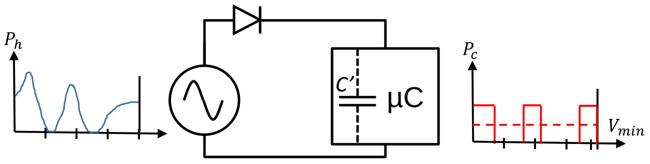

Figure 3. Principles of transient computing

Our recent work, Hibernus [5] takes advantage of the very small amount of capacitance in most embedded systems (parasitic or decoupling capacitance); *C'* in Figure 3. It uses this tiny buffer as a 'safety net' to allow the system to checkpoint its state once an outage is detected (when V_{CC} drops below a threshold), thus eliminating the need for routine speculative checkpoints and reducing overheads. Figure 4 illustrates the behavior of Hibernus when executing an FFT across multiple power outages, and the associated checkpoint (hibernate) and restore operations. QuickRecall [6] uses NVM as the processor's main memory, and hence little volatile state is left which requires checkpointing. This results in checkpoint operations being more efficient. However, as read/write operations to NVM typically consume more power than their volatile counterparts, the steady-state consumption increases [7]. Recent attempts overcome such limitations using a non-volatile processor, which have no volatile state to checkpoint at all [8].

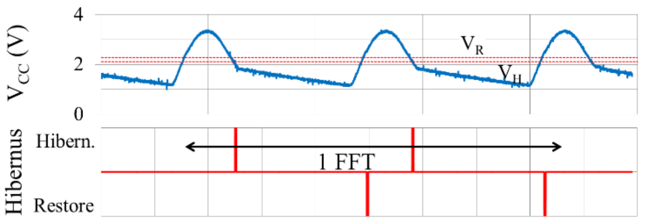

Figure 4. Example of transient computing permitting algorithm execution across multiple power outages.

4. POWER-NEUTRAL COMPUTING

The energy-neutral and transient computing approaches apply quite different techniques: transient computing executes a static workload but allows it to accommodate supply interruptions, while energy-neutral computing attempts to adapt workload intensity to meet an energy budget. Power-neutral computing (Fig. 5) complements transient computing with this adaptive operation.

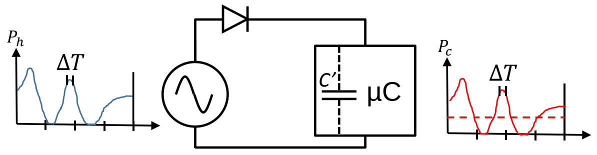

Figure 5. Principles of power-neutral computing

Instead of balancing energy across a large period of time (e.g. a day), power-neutral computing attempts balance across δT, a period effectively defined by the size of the system's decoupling capacitance *C'*. If harvested power drops below a usable level, the system uses transient computing approaches to save state to NVM, and restore at a later time when the supply has recovered. The term *power-neutral* is used as, with a tiny capacitance, the harvested and consumed *powers* are equal, rather than energies (equation 2).

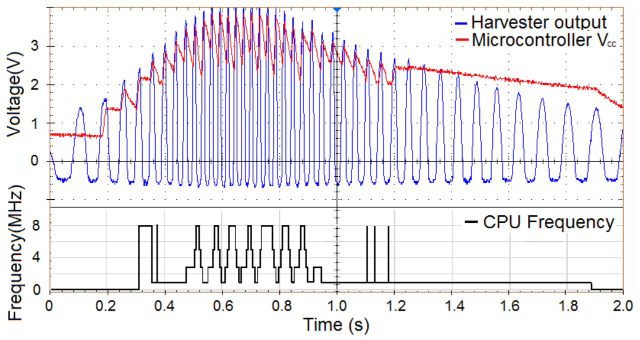

Figure 6. Power-neutral computing: adapting the CPU frequency to match harvested and consumed power.

$$\lim_{C' \to 0} \left(\frac{P_h(t)}{P_c(t)} \right) = 1 \qquad (2)$$

Figure 6 shows experimental results obtained from running a power-neutral system from a micro wind turbine (Fig. 1). At the start of the experiment, the harvester output (blue trace) is insufficient to power the microcontroller, and hence it sleeps. After ~300 ms, V_{cc} (red line) has risen to a usable level and the microcontroller wakes up. As the harvester output rises and decays, the CPU dynamically adjusts its frequency (and hence power consumption) in response (black trace) and, as a result, stays continually operational until ~1.1 s. At this point, V_{cc} drops below the minimum operating voltage, and the system checkpoints and then later restores (visible from the spike to 8 Mhz).

5. ACKNOWLEDGMENTS

This work was supported in part by EPSRC Grant EP/L000563/1. Experimental data used in this paper can be found at DOI: 10.5258/SOTON/4040469.

6. REFERENCES

[1] P. D. Mitcheson *et al.*, "Energy harvesting from human and machine motion for wireless electronic devices," Proc. IEEE, vol. 96, no. 9, pp. 1457–1486, Sep. 2008.

[2] C. Moser *et al.*, "Adaptive Power Management for Environmentally Powered Systems," *IEEE Trans. Computers*, vol. 59, no. 4, pp. 478-91, Apr 2010.

[3] A. Kansal *et al.*, "Power management in energy harvesting sensor networks," *ACM Trans. Embedded Computing Systems*, vol. 6, no. 4, 2007.

[4] B. Ransford, J. Sorber, and K. Fu, "Mementos: System support for long-running computation on rfid-scale devices," *ACM SIGPLAN Notices*, vol. 47, no. 4, pp. 159–170, 2012.

[5] D. Balsamo *et al.*, "Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems," *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 15–18, 2015.

[6] H. Jayakumar *et al.*, "QuickRecall: A HW/SW Approach for Computing across Power Cycles in Transiently Powered Computers," *in J. Emerg. Technol. Comput. Syst.*, Aug 2015.

[7] A. Rodriguez *et al.*, "Approaches to transient computing for energy harvesting systems: A quantitative evaluation," *in ENSsys* 2015, Seoul, Korea, November 2015.

[8] K. Ma *et al.*, "Architecture exploration for ambient energy harvesting nonvolatile processors," *HPCA 2015*: 526-537.

[9] D. Balsamo *et al.*, "Graceful Performance Modulation for Power-Neutral Transient Computing Systems," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* [in press].