Journal of Engineering Sciences

Vol 11, Issue 2, FEB/2020
ISSN NO:0377-9254

Smart Energy Management System In Low Power Iot Embedded Systems Using Advanced Energy Harvesting Methods

[1]Sugali Nenavath Vedasree
ECE specialization signals n communication system designing
Indian Institute Of Information Technology Design And Manufacturing
Chennai,India
esd18i020@iiitdm.a.in

[2]Valasingam Madhumitha
Very Large Scale Integration (Vlsi) & Electronic System Design
Indian Institute Of Information Technology Design And Manufacturing
Chennai,India
evd18i028@iiitdm.ac.in

[3]Sri Vatsan.O
ECE specialization signals n communication system designing
Indian Institute Of Information Technology Design And Manufacturing
Chennai,India
esd18i019@iiitdm.ac.in

[4]S.Krishna Kishore
ECE specialization signals n communication system designing
Indian Institute Of Information Technology Design And Manufacturing
Chennai,India
esd18i018@iiitdm.ac.in

***Abstract*—This paper presents some of the problematics of using energy harvesting in iot as a process of capturing energy from one or more renewable energy sources and converting them into usable electrical energy and how this concept can improve iot devices[3], we discuss some of these problematics such as having a two-dimensional maximum efficiency tracking technique for switched-capacitor based energy harvesting circuits in IoT applications also a smart management platform for low power IoT systems will certainly make a difference, and the one we're going to present which is some of the challenges of providing software primitives for transient computing to facilitate hardware-independent implementation using standard OS APIs, we will also present the integration of a state-of art transient approach, Hibernus into mbed OS.**

***Keywords*— Energy harvesting, Transient computing, IoT, unlimited energy, lowpower**

I. INTRODUCTION

There are many industries where reliable power sources are unavailable. IoT is one of them. Oftentimes, connected devices have to rely solely on battery power for operation, which adds cost and limits effectiveness in their application. There are also instances in which reliable power is accessible but undesirable. For instance, a door entry sensor could be plugged into the nearest outlet, or have power running through the wall, but this can drastically increase installationcosts and introduce potential security vulnerabilities. This makes battery life one of the biggest hurdles to IoT systems today. Normally, IoT devices are powered by the batteries. As the lifetime of battery is limited thus, there is strong need of self-powered devices or alternative sources of energy to continuously power the IoT devices. For the same purpose energy harvesting techniques are used nowadays. Energy har- vesting is the process of capturing energy from one or more renewable energy sources and converting them into usable electrical energy. This has large implications in IoT. It can increase the lifetime of a system while reducing maintenance. With these improvements, power usage is minimized, making energy harvesting methods viable for the lifetime of that IOT device.The concept of energy harvesting is used very broadly andimpliesthatthereisfreeenergyeverywherethatcanbe used on demand. When designing an IoT device, It'simportant to understand what sources of energy are available and know which source will provide enough energy consistently enough to power a device. As energy harvesting improves, the utility of IoT devices will continue to improve. The cost to maintain and operate systems will continue to decrease, making IoT more accessible and widespread. in this paper we discuss different aspects of energy harvesting in iot devices, some of the problematic it faces and some improvements that can be made to make the use of it more efficient namely establishing a smart management platform for low power IOT systems, also having a efficient tracking technique for energy harvesters[4] and finally Fuelling Adoption of Transient Computing in Embedded Systems and more specifically the one we're going to present: the integration of a state-ofart transient approach, Hibernus into mbed OS to demonstrate the applicability of transient approaches as partofanIoTOS.

II. RELATEDWORKS

The first article is entitled " An Input Power-Aware Effi- ciency Tracking Technique for Energy Harvestersing IoT" This paper presents a two-dimensional maximum efficiency track- ing technique for switched-capacitor based energy harvesting circuits in IoT applications. Conventional maximum power point trackers that sweep switching frequency (fSW) achieve high power conversion efficiency (PCE) for a limited range of input power. The purpose of this technique is to extend the input power harvesting range at which high power conversion efficiency can be achieved. To realize that, the proposed input- power aware system adds the capability of tuning the average load current to minimize converter power losses based on the sensed input power. This is fundamentally based on a discontinuous charging technique that delivers current to the load only during a controllable clock non-overlap time. Hence, the system can dynamically optimize the maximum PCE to actual ambient power levels. The system also tunes fSW to deliver the maximum output power. To demonstrate this technique, the system is designed and simulated in 90nm CMOS technology. For the same 40k load conditions, the proposed technique achieves at least 45% PCE at a 5uW input power, as opposed to a conventional system which requires at least 18uW of input power to maintain the same PCE. The harvestingrangeisthereforeextendedby72%.Thesecondon e is entitled "Energy harvesting and smart managementplatform for low power IoT systems" this article shows how Micro- Electro-Mechanical Systems (MEMS) and energy scavenging technologies are now able to provide viable solutions based on different application environments. For instance, rechargeable battery-operated Wireless Sensor Networks (WSN) can now be set upon a new basis that targets both ambient energy har- vesting and wireless charging technologies in different forms and scales. Hence, the assumption of finite energy budget can be over-passed by potentially unlimited energy supply. This article presents a platform that is able to effectively power alow powerIoT system with processing, sensing and wireless communication capabilities. It embeds an advanced energy management IC that allows extremely high efficiency energy harvesting, suitable for low-power and miniaturized energy generators. Unlike other systems it not only supports a variety of power supply options, but also a hybrid energystorage scheme. This paper also provide a description of the system's design, the embedded intelligence and the performance in terms of energy autonomy. The last article entitled "Energy Harvesting Meets IoT: Fuelling Adoption of Transient Com- puting in Embedded Systems introduce transient computing as systems that enables computation to be sustained despite power outages due to the variable nature of energy harvesting. But the issue is that existing approaches are largely designed for specific architectures, and hence are not broadly applica- ble across different IoT devices. Emerging platforms based on portable, hardware-independent software should rely on lightweight operating systems (OSs) designed specifically for embedded IoT applications. To enable the widespread use of transient computing, transient approaches need to beintegrated into these operating systems,which is why this article present the integration of a state-of art transient approach, Hibernus into mbed OS. Also transient computing is offered through a modular and layered structure that uses the available mbed OS APIs, including different strategies for retaining thesystem state designed for different types of flash memory. Which is why this article illustrate the applicability of the proposed design and implement Hibernus on two mbed platforms with different flash memories, which respectively requires 4.7mF and 4.9mF of additionalstorage.

III. HIBERNUSREDESIGN

We choose the Hibernus redesign model from the third article, this model discuss the redesign of Hibernus with mbed, starting with the software structure (Section A).

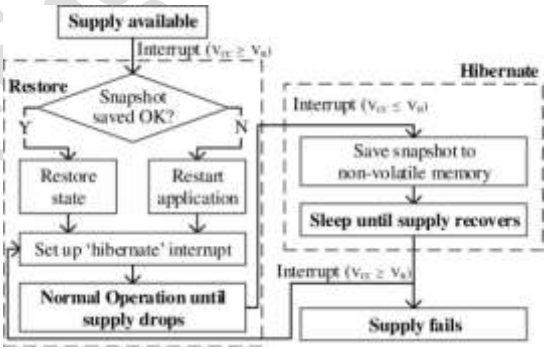

Fig. 1. Flow-chart illustrating the core algorithm of Hibernus[1]

We then present different snapshot strategies for flash memory (Section

B) and considerations related to the energy store required to enable the system state retention.

energy store required to enable the system state retention reliably (Section III-C).

A. Softwarestructure

Fig. 1 shows the operation of Hibernus. It moves between two states, active and hibernation, when the supply voltage VCC passes thresholds. It uses hardware interrupts to detect when VCC drops below VH or rises above VR. These can be triggered using an internal comparator (if available) or an external comparator. To integrate Hibernus into mbed, we designed a modular structure that operates at different layers of abstraction, separating platform independent (PI) and platform dependent (PD) functions. This OS was selected at first because it runs on 32-bit ARM embeddeddevices, provides a hardware abstraction layer and supports a large number of platforms. However, the structure presented here can be extended

to other IoT OSs. As shown in Fig. 2, the core algorithm can be considered as PI at the level of abstraction presented in Fig. 1. This is because it does not dictate any specific requirement for the type of NVM (and related snapshot strategy) or comparator needed to generate the interrupts. As a result, the algorithm is located in a PI library (hibernus.h and hibernus.cpp) that only uses standard mbed APIs for its methods and does not require any modification by the software designer. At a lower level of abstraction, a configuration layer defines the type of snapshot strategy and the comparator settings (i.e. internal or external). This layer is located into a separate file (config.h) that can be accessed by designers to set their parameters (a default configuration is already set for eachplatform).

The PI libraries are then implemented to enable the selected screenshot strategy and comparator settings. At this level of abstraction, methods are defined that use standard mbedAPI functions as well functions (e.g. copyRAMtoFlash()) whose implementation is specific for each platform. A lower device layer is, therefore, added to include the hardware and its software drivers. Specifically, these drivers relate to the platform- an memory-dependent methods to access the flash memory and to access the co-reregisters. This layer islocated

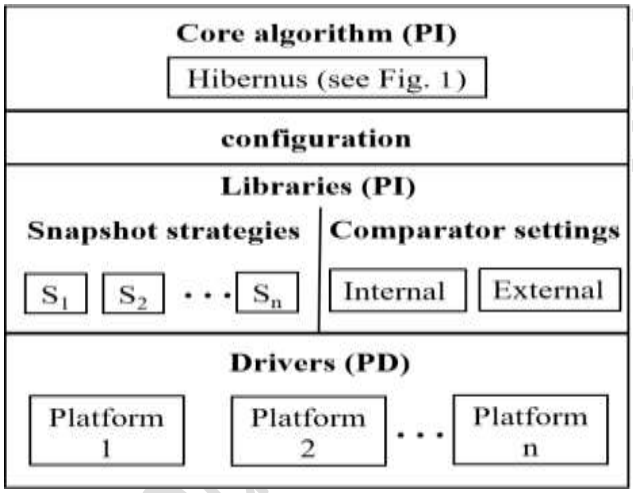

Fig. 2. API for Hibernus, based on a layered structure to separate platform.

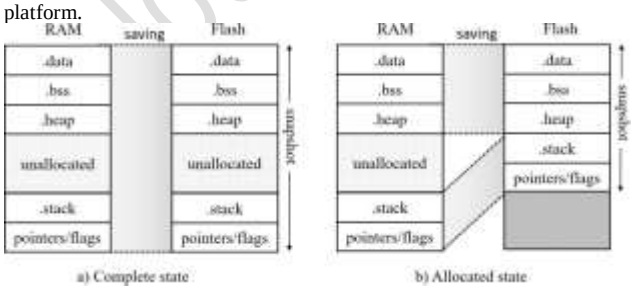

Fig. 3. Snapshot strategies

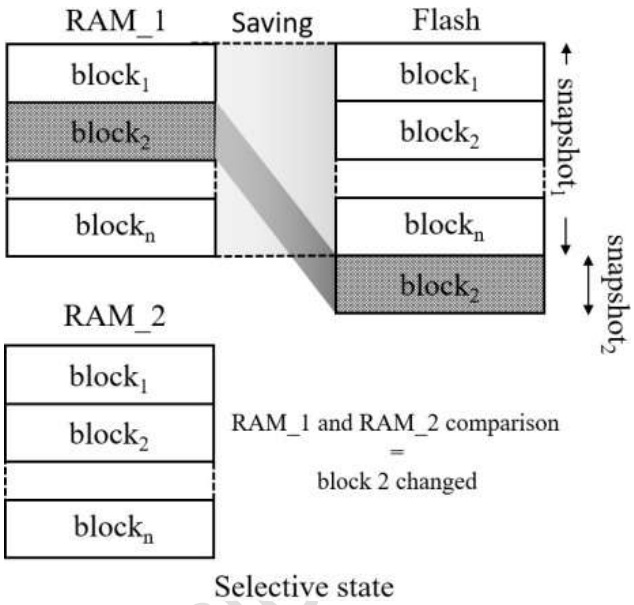

Fig. 4. Snapshot strategy illustrating selective state.

into a separate PD library (driver.h and driver.cpp) that only needs to be written once for each platform.

B. SnapshotStrategies

Different snapshot strategies are presented specifically de- signed for flash memories and related parameters. Flash memories operate by erasing a block of cells (page) before writing. The size of this page can range between a fewbytes to a few kilobytes. A given number of pages forms a sector. Erasing a sector is an energy consuming process because it involves generating a voltage pulse using a charge pump and directly impacts on the on the hibernation process Flash memories are asymmetrical as the reading is faster and more power efficient than the writing. For this reason, the presented strategies aim to optimize the time and energy overhead due to the writing process. As show in Fig. 3 a), a basic approach was implemented which saves the entire RAM to flash. This approach was originally implemented with Hibernus (designed for MCUs with FRAM) and relies on the fact that the size of the NVM is typically many times larger than the RAM. However, with flash memories, the number of sectors dedicated to other purposes other than holding the applicationimage(e.g..textsegment)canvary.Forthisreason, different options to reduce the amount of data saved from RAM to flash during hibernation have been considered. In the basic implementation each snapshot consists of the entire RAM including the unallocated part of the memory, as well as a dedicated segment, containing pointers and flags requiredfor the restore (pointers/flags in Fig. 3 a)). To reduce the number of sectors required in flash, this dedicated segment can be moved into the unallocated part (if available) of the RAM instead (using a static address)

before copying the RAM into flash. A better solution was recently proposed, which consists of saving only the allocated part of the RAM to flash (see Fig.3b)[2].In this case, each snapshot consists of the. data, bssand. Heap segments , aswell as the .stack and the dedicated section, containing pointers and flags required for the restore. To identify these segments, the saving process needs to track the end of the. Heap segment and the top of the.stack.Finally,a novel approach is proposed which aims to significantly reduce the number of erase operations, by copying in

flashonlytheblocksoftheRAMmemorythathavechanged since the last restoring process. As shown in Fig. 4, the RAM memory is divided into two identical segments (RAM-1 and RAM-2).ThemainapplicationcodeonlyactsontheRAM-

1 segment, which is divided into different blocks (block1, block2, ... , blockn). Similarly, RAM-2 is organized using the same block structure. At each restore, the content from flash is saved to both segments, RAM-1 and RAM-2. In this way, RAM-2 preserves an image, at the restore time, that is not modified during the execution of the main IoT application. At the hibernation time, RAM-1 and RAM-2 are compared and only the blocks that have changed are saved in flash. For example, in Fig. 4 only block2 is saved in flash during the second snapshot. Other selective policies for efficient state retention with transient systems which exploit the propertiesof different NVM memories were recently proposed[5]. Due to the modular structure of the presented software, these polices can be added and evaluated as separate modules.

C. EnergyStorage

The energy consumed for the state retention process with flash memory, E_σ, depends on the erasing cost and the energy required for saving the RAM and core registers to flash:

$$E_\sigma = n_s E_s + n_\alpha E_\alpha$$

where $n_s$is the number of sectors to be erased and n is the size of the RAM and registers (inbytes) . Es is the energy

required to erase one sector (J) and E are the energy required to copy each RAM byte to NVM (J/byte).

Hibernus requires enough energy to be stored in the capac- itance between the supply rails to save a snapshot. Given a defined VH , the energy E stored between VH and systems's minumum opration voltage Vminis

IV. NUMERICALRESULTS

In the model we selected, a numerous simulations were made , the model illustrate the applicability of the proposed designtoplatformswithdifferentflashmemory,apracticalim - plementation of Hibernus on two mbed platforms is presented. The first platform, a Freescale KL05Z, is built on the Arm Cortex M0+, with small RAM capabilities (4KB), 32KB of flash, and holds an internal comparator that can be used to trigger the interrupts for saving and restoring. The test platform used for this validation is where the MCU is connected to the output of the energy source through a schottky diode to prevent back-flow of charge to the source. We also display the power and time needed to save the system state with the KL05Z and LPC11U24, respectively. The first platform consumes an average power of 66mW at 3V, for 51.8ms, to erase 5 sectors (each sector is 1KB) and an average power of 50.4mW at 3V, for 66.6ms, to copy the entire RAM plus the pointers/flags segment, corresponding to an energy equal to 6.7mJ. . The required capacitance C is equal to 4.7mF, for a given VH equal 2.4V and Vmin equal to 1.7V The second one consumes an average power of 66.45mW at 3V,for 98.4ms, to erase 2 sectors (each sector is 4KB) and an average power of 66.33mW at 3V for 41ms, for copying the entire RAM, corresponding to an energy equal to 9.2mJ. It is interesting to notice that the time for erasing and writing between the two platforms changes significantly. This depends on the specific flash memory settings(i.e.theabilityoferasing multiple consecutive sectors at the same cost), the size and number of sectors to be erased and the size of the RAM memory.Finallythearticleillustratesthesystembehavior with a sinusoidal waveand a simple binary counter as application.

V. CONCLUSION

In this paper we talked about the articles we chose andlisted some of the problematic they mentioned and also some of the solutions they proposed, for our paper we chose the model of "Energy Harvesting Meets IoT: Fuelling Adoption ofTransient Computing in Embedded Systems" this model concerns the challenges in integrating transient computing, i.e., Hibernus as part of an IoT OS. These relate to the ability of pro- viding software primitives to facilitate hardware-independent implementation of these approaches using standard OSs APIs, and enabling the system state retention with flash memories. This support is offered through modular structure that uses novel approach is proposed which aims to significantly the system state designed for flash memory and validated

ACKNOWLEDGMENT

We would like to show our gratitude and thank to our professors who provided insight and expertise and their greatly time to examen this early version of the manuscript. although any errors are our own and should not tarnish the reputations of these esteemed persons.

REFERENCES

[1] Domenico Balsamo, Alex S Weddell, Geoff V Merrett,

Bashir M Al- Hashimi, Davide Brunelli, and Luca Benini. Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems. *IEEEEmbeddedSystemsLetters*,7(1):15–18,2014.

[2] Naveed Bhatti and Luca Mottola. Efficient state retention for transiently- powered embedded sensing. In *International Conference on Embedded WirelessSystemsandNetworks*,pages137–148,2016.

[3] Oliver Hahm, Emmanuel Baccelli, Hauke Petersen, and Nicolas Tsiftes. Operating systems for low-end devices in the internet of things: a survey. *IEEEInternetofThingsJournal*,3(5):720–734,2015.

[4] Xiaodong Meng, Xing Li, Chi-Ying Tsui, and Wing-Hung Ki. An indoor solar energy harvesting system using dual mode sido converter with fully digital time-based mppt. In *2016 IEEE International Symposium on CircuitsandSystems(ISCAS)*,pages2354–2357.IEEE,2016.

[5] Theodoros D Verykios, Domenico Balsamo, and Geoff V Merrett. Selec- tive policies for efficient state retention in transiently-powered embedded systems: Exploiting properties of nvm technologies. *Sustainable Com- puting:InformaticsandSystems*,22:167–178,2019.