Simulating Power/Energy Consumption of Sensor Nodes with Flexible Hardware in Wireless Networks

Jingyao Zhang, Srikrishna Iyer, Patrick Schaumont, and Yaling Yang
Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24060
Email: {jingyao, skr, schaum, yyang8}@vt.edu

***Abstract*—Energy consumption and real-time performance are two important metrics for wireless sensor networks (WSNs). To estimate these metrics, a number of simulation environments have been developed. However, these environments were made specifically for sensor nodes with fixed architectures. The recent generation of sensor nodes often has flexible architectures through the use of programmable hardware components, i.e., Field-programmable gate arrays (FPGAs). So far, no simulators have been developed to evaluate the performance of such flexible nodes in wireless networks. In this paper, we present PowerSUNSHINE, a power- and energy-estimation tool that fills the void. PowerSUNSHINE is the first scalable power/energy estimation tool for WSNs that provides an accurate prediction for both fixed and flexible sensor nodes. In this paper, we first describe requirements and challenges of building PowerSUNSHINE. Then, we present power/energy models for both fixed and flexible sensor nodes. Two testbeds, a MicaZ platform and a flexible node consisting of a microcontroller, a radio and a FPGA based co-processor, are provided to demonstrate the simulation fidelity of PowerSUNSHINE. We also discuss several evaluation results based on simulation and testbeds to show that PowerSUNSHINE is a scalable simulation tool that provides accurate estimation of power/energy consumption for both fixed and flexible sensor nodes.**

I. Introduction

Nowadays, WSNs are proposed to be used in many applications, such as structure and environment monitoring, health care, and so forth. In the past, these WSNs were composed of sensor nodes that mainly consist of a microcontroller and a wireless transceiver. However, the microcontroller's processing capability may cause a real-time bottleneck when sensor nodes have to execute compute-intensive tasks, such as message encryption/decryption and large data compression/decompression. To accelerate the execution speed of the sensor nodes, adding a hardware accelerator to form a flexible sensor node has been recently proposed in [1] [2].

Apart from fixed components, such as a transceiver and a microcontroller, a flexible sensor node has a programmable hardware component, i.e., FPGA. In contrast to the fixed sensor node whose hardware functionalities, such as circuitry, clock frequency and I/O ports are fixed, the programmable logic of FPGA can be configured to perform either complex algorithms by programming thousands of logic cells or simple calculations that just uses one AND or OR gate. Based on this functionality, executing compute-intensive tasks in parallel on FPGA instead of sequentially on microcontroller can make the flexible sensor node's execution speed orders of magnitude faster than the fixed sensor node's.

Due to the high cost of building, deploying and debugging distributed sensor network prototypes in real environments, it is better to evaluate applications in simulation before deploying applications on actual WSNs. Unfortunately, no simulators have been developed to evaluate the real-time performance and energy consumption of such flexible platforms. Therefore, it is difficult to identify what specific applications can benefit from flexible platforms in large WSNs.

To evaluate the real-time performance of flexible platforms, in our previous work, we built SUNSHINE [3]. SUNSHINE is a cycle-accurate simulator that can emulate the behaviors of flexible sensor nodes in wireless networks. While we have demonstrated that SUNSHINE can accurately capture the timing behaviors of WSNs' applications on flexible hardware platforms, estimating their power/energy consumption has turned out to be very challenging and has remained unsolved until this paper.

Predicting the power consumption for flexible sensor nodes is challenging for two reasons. First, predicting the power/energy consumption of fixed (microcontroller) and flexible (FPGA) components' interactions in wireless network environment is difficult. Second, the power estimation processes for fixed and flexible components are completely different from each other. Because of the above challenges for estimating power consumption of flexible nodes, existing power estimation tools [6] [7] only support fixed sensor nodes. The lack of capability on analyzing power consumption of flexible nodes would result in restricting analysis and development of flexible sensor platforms in large networks.

The focus of this paper is to describe our novel design of a power/energy estimation tool called PowerSUNSHINE for WSNs. PowerSUNSHINE is able to predict power/energy consumption of not only fixed-platform sensor nodes, such as MicaZ nodes, but also flexible sensor nodes with reconfigurable FPGAs. To the best of our knowledge, PowerSUNSHINE is the first to provide power/energy estimation of flexible sensor nodes.

Our major contributions are summarized as follows.

1) We developed a methodology for estimating power/energy consumption of flexible sensor platforms in wireless network environment. Based on this method,

power/energy consumption models for each component, including microprocessor, radio transceiver, and FPGA-based component, are established, so that a wide range of sensor platforms' power/energy consumption can be captured by combining the power/energy consumption of their components.

2) Following our methodology, we built a power/energy modeling extension, called PowerSUNSHINE, into the SUNSHINE simulator. Unlike other power tools that only evaluate fixed hardware platforms, PowerSUNSHINE supports both fixed and flexible sensor platforms.
3) We set up two testbeds, a MicaZ platform and a flexible sensor platform with a FPGA-based co-processor, to evaluate the fidelity of PowerSUNSHINE.

The rest of the paper is organized as follows. Section II presents related work of power tools for wireless sensor networks. Section III first introduces the architecture of SUNSHINE, and then presents PowerSUNSHINE's characteristics, architecture, and challenges. Section IV presents power/energy models of fix-function components. Section V discusses power/energy models of reconfigurable components. Section VI provides the setup of actual hardware platforms. Section VII offers evaluation results of PowerSUNSHINE. Section VIII discusses future work. Finally, Section IX provides conclusions.

II. Related Work

To measure actual sensor nodes' power consumption directly, several papers [4] [5] measured actual sensor nodes' current at real-time via specialized circuits. Even though these methods have high-precision results, building hundreds of circuits to measure large WSNs' power/energy turns out to be time-consuming and impractical. In such a case, building a system to estimate the WSNs power/energy consumption is crucial in the area of sensor networks.

Several simulation tools for energy profiling of sensor nodes have been developed in existing work. For example, PowerTOSSIM [6] has been built on top of TOSSIM simulator to estimate Mica2's energy consumption. Since TOSSIM cannot emulate a microcontroller's execution time, to estimate the microcontroller's power consumption, PowerTOSSIM has to estimate microcontroller's execution time based on the intermediate C code generated by tinyOS applications. This estimation, however, may be fairly inaccurate in many cases. By comparison, in PowerSUNSHINE, the microcontroller's cycle counts are precisely counted by SUNSHINE. Therefore, the microcontroller's energy consumption can be more accurately captured.

AEON [7] is developed based on a cycle accurate simulator AVRORA to profile Mica2's energy. AEON breaks down Mica2's components and calculates each hardware's energy in the system. AEON is able to capture Mica2 nodes' power consumption accurately since AVRORA can simulate microcontroller's cycle-accurate behavior.

None of PowerTOSSIM or AEON is able to evaluate the power consumption of flexible sensor nodes. They are dedicated for fixed sensor nodes. PowerSUNSHINE is able to capture both fixed and flexible sensor nodes' power consumption.

III. PowerSUNSHINE Overview

In this section, we first briefly introduce the architecture of SUNSHINE, which is the foundation of PowerSUNSHINE. Then, we describe the characteristics, architecture and challenges of PowerSUNSHINE.

A. *SUNSHINE Simulator*

PowerSUNSHINE's ability to profile the power consumption of fixed and flexible sensor nodes is based on SUNSHINE, a cycle-accurate hardware-software simulator for sensor networks. SUNSHINE is developed by the authors in their previous efforts and is the only existing simulator that can simulate flexible sensor platforms. Other existing sensor network simulators can only capture fixed hardware platforms and do not support simulation of reconfigurable hardware designs. In the following, we give an overview of SUNSHINE.

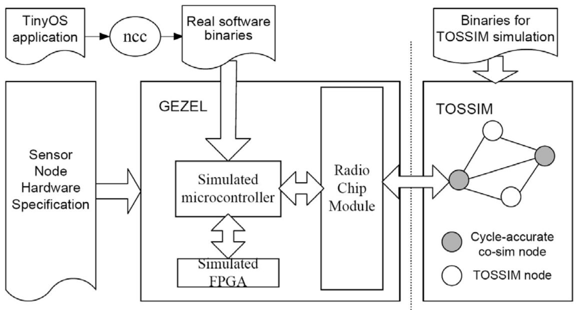

TinyOS application
ncc
Real software binaries
Binaries for TOSSIM simulation
GEZEL
TOSSIM
Sensor Node Hardware Specification
Simulated microcontroller
Radio Chip Module
Simulated FPGA
Cycle-accurate co-sim node
TOSSIM node

Fig. 1. SUNSHINE software architecture

Fig. 1 illustrates SUNSHINE's software architecture [3]. A sensor node can be simulated by SUNSHINE in two different modes: co-sim mode or TOSSIM mode. For nodes simulated in TOSSIM mode (called TOSSIM nodes), only high-level functional behaviors are captured while for nodes in co-sim mode (called co-sim nodes), the behaviors of hardware co-processors are described by a hardware description language, GEZEL [24] and are simulated at cycle-level accuracy. The cycle-accurate behaviors of other components in co-sim nodes, such as microcontrollers and transceivers, are also captured in SUNSHINE.

With the support of SUNSHINE, especially its ability of simulating accurate behaviors of co-sim nodes, building a power/energy estimation tool for both fixed and flexible sensor platforms in network environment becomes feasible. Furthermore, building PowerSUNSHINE over SUNSHINE simulator has the following advantages:

Accuracy: SUNSHINE accurately captures the behaviors of sensor nodes at cycle level. This provides the foundation to ensure that the power/energy consumption of sensor nodes estimated by PowerSUNSHINE is close to the measurement results of actual boards.

Flexibility: Based on SUNSHINE's capability to simulate arbitrary hardware platforms, PowerSUNSHINE supports estimating power/energy consumption of different sensor platforms.

Compatibility: Since TinyOS applications can run in SUNSHINE, PowerSUNSHINE can profile power/energy consumption of sensor nodes running TinyOS applications directly. This is useful because TinyOS is the dominating operating system for WSNs.

Path to Implementation: Both SUNSHINE and PowerSUNSHINE bridges the gap between design and implementation of flexible sensor nodes' applications. The applications evaluated by SUNSHINE and PowerSUNSHINE in simulation can be loaded and run on actual hardware.

B. PowerSUNSHINE Architecture

Building a power/energy simulation model for flexible hardware platforms (with fixed hardware platform as a special case) is a non-trivial task. PowerSUNSHINE aims to capture a wide range of possible platform designs that are formed by different combinations of hardware components. Thus, power models based on measurement of the power consumption of existing platforms as a whole will not work, since one platform cannot represent the power consumption of another platform with different hardware designs.

To solve this problem, PowerSUNSHINE decomposes the power consumption of a sensor platform into a combination of power consumption of individual hardware components. Fig. 2 illustrates the block diagram of PowerSUNSHINE architecture. PowerSUNSHINE is associated with co-sim nodes, whose cycle accurate hardware-software behaviors are captured by SUNSHINE. When SUNSHINE is simulating applications of sensor nodes, PowerSUNSHINE breaks down sensor nodes into components, calculates power/energy consumption of each component, and then adds all the components power/energy consumption together.

To be specific, if PowerSUNSHINE is applied for fixed sensor nodes in the simulation, it tracks cycle accurate activities of every component, and uses the power/energy model to calculate the total power/energy consumption of the nodes according to their components' activities.

Compared with fixed nodes, a flexible node has an extra programmable FPGA. If PowerSUNSHINE is applied for the flexible node, the additional power/energy dissipation of FPGA should be considered. Therefore, the total power/energy profiling should contain the power/energy consumption of both fixed hardware components and the reconfigurable FPGA.

By establishing a power/energy model for each hardware component, PowerSUNSHINE can estimate the power/energy consumption of arbitrary platform designs.

C. Challenges

Establishing power models for individual hardware components is a fairly challenging task. First, hardware components with fixed functions, such as microcontrollers and radio

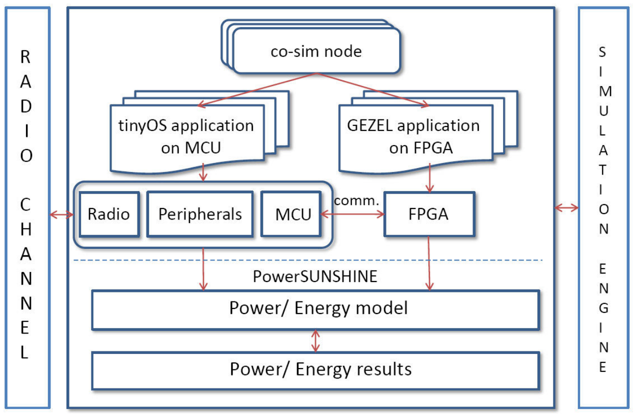


Fig. 2. Block diagram of PowerSUNSHINE architecture

chips, have different operation states with different power consumption. Hence, PowerSUNSHINE's model of these fixed hardware components must estimate the power consumption of each operation state during the simulation of the sensor platforms.

Second, reconfigurable hardware components like FPGA chips do not have fixed operation states. The power consumption of FPGA depends on how the FPGA is configured and cannot be possibly known at the time of PowerSUNSHINE's development. Hence, PowerSUNSHINE must be able to derive the power consumption of the FPGA based on the descriptions of its functions at the simulation time.

In the following two sections, we illustrate PowerSUNSHINE's methods to address the above two challenges by showing how we model the power/energy consumption of radio chip, microcontroller, LEDs, and FPGA chip. These are common hardware components on sensor platforms. The power consumption of other possible hardware components can also be obtained with the same methods.

IV. Power/Energy Models for Fix-Function Components

In this section, we first describe the power/energy model of a fixed sensor node. Then, we present how we obtain the power/energy consumption of each hardware component, such as microcontroller, radio, and LEDs. In this paper, we use MicaZ platform as an example of the fixed sensor nodes.

A. Power/Energy Model of Fixed Senor Node

Fixed sensor nodes' energy consumption depends on their hardware components. Therefore, the energy model can be presented as shown below:

$$\begin{aligned} E_{total} &= E_{mcu} + E_{radio} + E_{otherperipherals} \\ &= \textstyle\sum_{devices}(\sum_{states} V \cdot i_{state} \cdot n_{cycles\_state}). \end{aligned} \tag{1}$$

where "devices" contain microcontroller, radio, and other peripherals on the board, "states" represent different devices' states in the simulation, i_{state} is the current of a device at the dedicated state, "n_{cycles_states}" is the microcontroller's cycle numbers spent on the state, and V is the constant voltage.

We describe how we calculate the power/energy consumption of different components shown in formula (1) in the following.

B. Measurement Setup and Results

Since sensor nodes' current varies due to different environments, to accurately capture the nodes' power consumption, we measure the nodes current in our own environment. To measure the individual power consumption of ATmeg128L microcontroller, CC2420 radio chip, and LEDs on a MicaZ platform, we use MicaZ OEM nodes [17], LeCroy WaveSurfer 24Xs-A Oscilloscope with a 2.5 GS/s sampling rate [18], CADDOCK high performance 0.50 Ohm shunt resistors [19] with a tolerance of $\pm 1\%$, and a TENMA 72-6905 4CH laboratory DC power supply [20]. We used similar method as [6] to get the current of the sensor nodes. The current can be obtained via measuring the voltage drop on the shunt resistor by the oscilloscope. The measurement setup is shown in Fig. 3(a). For MicaZ nodes, the programs are loaded via MIB510 programmer to the microcontroller.

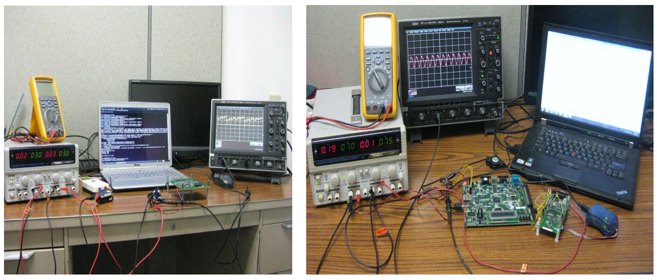

(a) (b)

Fig. 3. Testbeds for measuring power consumption of (a) MicaZ and (b) flexible nodes.

Based on the measurement setup, the current draw of applications running on MicaZ can be captured. To be specific, the current of CC2420 radio transceiver, ATmega128L microcontroller and LEDs on a MicaZ sensor platform can be obtained by the measurement setup using TinyOS codes. To identify each component's current from measurement, we took the following steps. First, we measured the current draw of microcontroller in different modes, including active, idle, extended standby, power-down, power-save, ADC noise reduction and standby [21]. To measure the microcontroller's current on the sensor node, we only turned on the microcontroller of the sensor node, and set the microcontroller in different modes using TinyOS codes. We measured the corresponding microcontroller's current respectively, and recorded the relevant results as shown in Table I. Second, we captured the current draw of LEDs on the sensor node. We let the microcontroller tweak one LED at one time, and measured the corresponding LED's current. Then, we got each LED's current by subtracting the microcontroller's current from the sensor node's current. Finally, we need to capture radio transceiver's current. Since the radio transceiver supports different transmission power to send out packets, and different transmission power costs different power consumption of the transceiver, it is essential that the transceiver's current with different transmission power should be captured. In the following, we will show the methods of capturing radio's current with 0dBm transmission power (default in TinyOS). Other transmission power's current of the transceiver is obtained using the same method except setting different transmission power in TinyOS code.

To obtain the radio's current, we turned on the radio and let the sensor node transmit and receive packets from the wireless channel. We captured the current of the whole sensor node based on the measurement setup. The results are shown in Fig. 4 to Fig. 6. Fig. 4 shows the current draw for transmitting and receiving six packets between two nodes. As shown in Fig. 4, as soon as sending out one packet to the air, the transmitting node sends out another packet. When finishing the transmission of six packets, both microcontroller and radio on the transmitting node go to sleep. The receiving node keeps listening to the channel to receive data. As Fig. 4 indicates, by sampling the node's current waveform over time, the time-dependent power consumption of the sensor node becomes obvious.

Fig. 5 and 6 show parts of Fig. 4 and present transmitting and receiving one packet respectively. As Fig. 5 shows, a transmitting node first calibrates the radio, let microcontroller transfer packet data to the radio, and asks the radio to listen to the channel. After getting a "send" command from the microcontroller, the radio sends out the packet data when the channel is available. As Fig. 6 shows, for a receiving node, the radio keeps listening to the channel. When the radio on the node receives data from the air, it wakes up the microcontroller. After receiving one packet, the radio sends the packet to the microcontroller [16].

After knowing the node's behaviors and corresponding current value shown in the Figures, it is feasible to get the radio transceiver's current by subtracting the microcontroller's current from the whole node's current. The results shown in Table I provide reference for PowerSUNSHINE to calculate the power/energy consumption of sensor nodes.

Based on these results, the current of sensor node's components on different states are known. In order to predict the power/energy consumption of individual components, we also need to identify each component's transitions at simulation runtime so that we can derive the time duration of these states during the execution of an application in simulation. In the following, we present how PowerSUNSHINE profiles components' state transition and eventually derive power/energy consumption of sensor nodes in simulation.

C. Power/Energy Estimation Method

1) Microcontroller: The estimation of microcontroller's power/energy consumption is achieved by identifying microcontroller's states and time duration at cycle level. We will present how PowerSUNSHINE predicts microcontroller's power/energy consumption in the following.

We assume that WSN applications' software are written in nesC [15] and run over TinyOS operating system. NesC is a

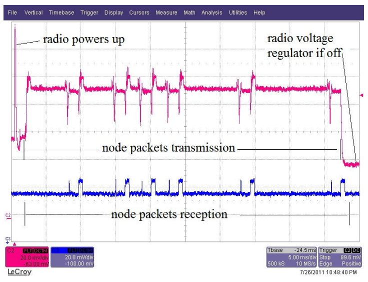


Fig. 4. Transmission & reception of six packets. After sending out all the six packets, the radio voltage regulator is turned off.

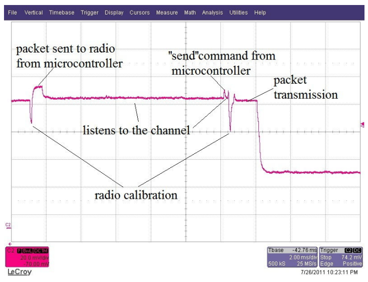


Fig. 5. One packet transmission

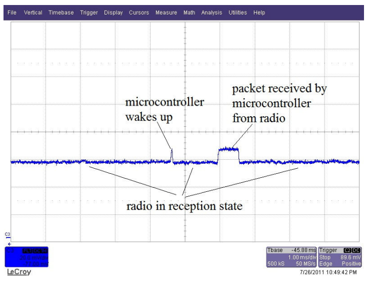


Fig. 6. One packet reception

TABLE I
MEASUREMENT RESULTS FOR THE MICAZ WITH A 3V POWER SUPPLY.

Device	Current (mA)	Device	Current (mA)
MCU		**Radio** (2.4 GHz)	
active	7.24	Rx	19.30
idle	3.98	Tx (0 dBm)	17.32
Ext standby	0.24	Tx (-3 dBm)	15.97
Power-down	0.09	Tx (-5 dBm)	13.8
Power-save	0.10	Tx (-7 dBm)	12.80
ADC Noise	1.2	Tx (-10 dBm)	11.3
Standby	0.23	Tx (-15 dBm)	9.7
Led		Tx (-25 dBm)	8.2
Red	2.96		
Green	2.64	Power down	0.22
Yellow	2.77	Idle	0.41
Device	time	Device	time
CPU bootup	154.72 ms	**Radio** bootup	2.138ms
timer0 duration	275.53 μs	oscillator stabilization	247 μs

high-level programming language that can be compiled to C file using ncc compiler. The compiled C file includes firmware programs that reflect how actual hardware should behave.

In PowerSUNSHINE, instructions to toggle several unused general Input/output pins (I/Os) of the microcontroller are added to the C file right before every line of C code that will change the state of the microcontroller during execution. Different values of these I/Os (called state pins) after the toggles are used to identify different states of the microcontroller. During the simulation of the sensor node at cycle level, the hardware cycles between the toggles are recorded so that the time duration that the microcontroller spent on each state can be computed.

Since the microcontroller needs to spend time on toggling SUNSHINE state pins, the overhead of the toggling is compensated in the calculation as follows. We calculate the number of state pins' toggles and subtract the number from the total estimated clock cycles spent on the corresponding states.

By the above modeling, the time duration of the microcontroller's states and their corresponding current (shown in table I) are known. As the sensor node is supplied by a constant power supply in the experiments, according to the energy formula $E = V \cdot I \cdot t$, where V, I, and t are voltage, current and time duration respectively, the microcontroller's energy consumption can be accurately estimated using PowerSUNSHINE.

2) Peripherals: Peripherals are any fixed sensor node components apart from the microcontroller. These peripherals include radio transceiver, LEDs and etc. PowerSUNSHINE can also accurately predict these peripherals power/energy consumption in simulation.

For radio transceiver, PowerSUNSHINE traces the CC2420 radio's activities in simulation at cycle level. This is feasible because the CC2420 radio is implemented inside SUNSHINE as a hardware module of a transceiver, whose activities are built according to CC2420's datasheet [16]. In simulation, the cycle-accurate behaviors of the radio can be captured. For example, how the radio interconnects with microcontroller, what packets the radio transmits and receives, when the radio sleeps and wakes up, are all simulated. In addition, the time duration of the radio's different activities can be captured. Combining with the measured power consumption for different activities, the radio's energy consumption can be profiled in the simulation by PowerSUNSHINE.

Other peripherals, such as LEDs, which only have ON/OFF states, can be modeled by recording the duration of ON states in simulation. At the end of the simulation, the peripherals' energy consumption can be calculated using the energy formula $E = V \cdot I \cdot t$, where V, I, and t are voltage, current and time duration respectively.

V. POWER/ENERGY MODELS OF RECONFIGURABLE COMPONENTS

Since the power consumption of reconfigurable FPGA is defined by its configuration, the power estimation method of FPGA is different from other fixed hardware components, for example, microcontroller and radio, whose power consumption are constant at one certain state. For the flexible sensor node, the power/energy consumption of the FPGA is due to the FPGA core's activities, i.e. executing tasks on the FPGA. In this section, we present how we model the power/energy consumption of flexible sensor nodes.

A. Power/Energy Consumption of FPGA Core

PowerSUNSHINE predicts power consumption of FPGA core by leveraging existing power estimation tools. Almost all of FPGA manufacturers provide power estimation tools for their specific FPGAs. For example, IGLOO Power Calculator for IGLOO series FPGAs, ProASIC3 Power Calculator for ProASIC3 series FPGAs [9], Power Analyzer for Altera FPGAs [10] and XPower Analyzer [13] for Xilinx FPGAs.

In this paper, we use Spartan-3E XC3S500E-4FG320C FPGA [11] on Xilinx Spartan-3E starter kit. In PowerSUNSHINE, XPower Analyzer [13] is incorporated to estimate power consumption of the FPGA. XPower Analyzer supports power estimation of different hardware blocks, for example, registers, signals, clocks, etc.

To accurately profile FPGA's power, several design files should be provided [22]. In SUNSHINE simulation, we use GEZEL [24] to describe the architecture of sensor nodes. Since GEZEL code can be translated to synthesizable VHDL code, it can also be used to generate the input files for FPGA power estimation. Thus, we can use GEZEL and existing power estimation tools to provide accurate power analysis of FPGA component.

B. Power/Energy Model of Flexible Platform

With all the power/energy models established, PowerSUNSHINE can compute the energy consumption of a flexible platform as follows:

$$\begin{aligned} E_{total} &= \textstyle\sum_{devices}(\sum_{states} V \cdot i_{state} \cdot n_{cycles\_state}) \\ &+ E_{FPGA\_core}, \end{aligned} \tag{2}$$

where the first element is the energy consumption of components with fixed functions, E_{FPGA_core} is the energy dissipation of FPGA core.

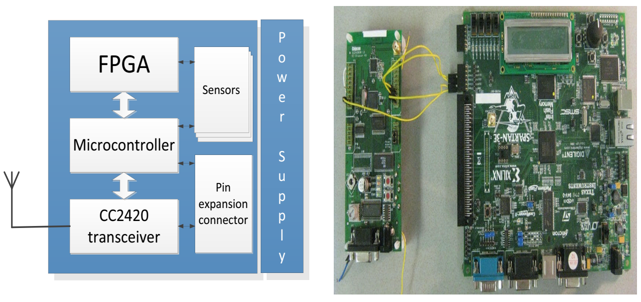
Based on the energy models described in Section IV and V, the energy consumption of both fixed and flexible sensor nodes can be estimated using PowerSUNSHINE.

VI. Test Platform Setup

We evaluate the simulation fidelity of PowerSUNSHINE by comparing its simulation results with two platforms. The first is an off-the-shelf MicaZ OEM node, which is mainly composed of an ATmega128L microcontroller, a CC2420 radio and three LEDs. The testbed is shown in Fig. 3(a). The second platform is a customized flexible platform, which mainly consists of an ATmega128L microcontroller, a CC2420 radio and a FPGA. In this section, we present the architecture and setup of this flexible platform.

A. Flexible Platform Architecture

On the flexible hardware platform built for our validation purpose, the FPGA is used as a co-processor that handles compute-intensive tasks to speed up the node's execution time. The block diagram of the platform is shown in Fig. 7(a). In the Figure, FPGA, microcontroller and radio are interconnected. The interconnection between microcontroller and FPGA is via communication protocols, such as SPI, UART, I^{2}C, parallel, and so on. SPI communication protocol was developed between FPGA and microcontroller in SUNSHINE environment in our previous work [8], and is used in this paper. In addition, SPI arbitration between SPI master, microcontroller, and two SPI slaves, FPGA and radio chip is also implemented in SUNSHINE. Therefore, the behaviors of flexible sensor nodes can be emulated in simulation and evaluated on actual hardware platforms.



(a) (b)

Fig. 7. (a) Block diagram of flexible node (b) One flexible node setup

It is worth to note that the platform shown in Fig. 7(a) is not the only possible flexible hardware platform design. Other hardware architectures, for example, placing FPGA between microcontroller and radio can also be simulated, and these architectures' power/energy consumption can be profiled by PowerSUNSHINE. In addition, sensors on the node can be added to either FPGA or microcontroller according to the requirements of applications.

B. Flexible Platform Testbed

To validate simulation fidelity of PowerSUNSHINE, we provide a real platform with Spartan-3E XC3S500E-4FG320C FPGA on Xilinx Spartan-3E starter kit, Atmega128L and CC2420 on the TI CC2420DBK [12] as shown in Fig. 7(b).

We choose Spartan-3E starter kit as the FPGA component because it provides LCD display, eight individual LEDs, three 6-pin expansion connectors and JTAG interface [11] which would be helpful for debugging on actual hardware. Note that the estimation method of PowerSUNSHINE can be applied to many different FPGA chips. We use Spartan-3E as a demonstration for the validation of PowerSUNSHINE. Other low-power FPGAs can be used in place of Spartan-3E.

We also use microcontroller and radio on CC2420DBK to configure the flexible node as shown in Fig. 7(a). CC2420DBK has similar hardware components as MicaZ node. The main difference between them is that CC2420DBK provides interface to connect FPGA with microcontroller, and it does not have a 32.768 KHz external oscillator. With the external oscillator, the microcontroller can go into power-save mode while without the oscillator, the microcontroller can only stay at power idle state that consumes much more power than staying at power-save state as shown in Table I.

The communication between Spartan-3E FPGA and CC2420DBK is based on SPI protocol. The FPGA and the radio can work coordinately with the microcontroller based on SPI arbitration. On the software side, we have modified TinyOS codes to ensure that the codes can operate on the new platform. When programming the flexible nodes, the programs for the microcontroller are loaded via AVRISP mkII

TABLE II
ENERGY CONSUMPTION (IN mJ) OF TINYOS APPLICATIONS ON MICAZ. ESTIMATED WITH POWERSUNSHINE.

Application	MCU idle	MCU active	Radio	Leds	Total in simulation	Measured	Accuracy (%)
10^4 empty loops	0	2.172	0	0	2.172	2.193	99.0%
Blink	14.98	1.33	0	627.75	644.062	631.8	98.1%
RxCount	596.04	1.73	2895	0	3492.78	3450.8	98.8%
TxCntToAir	595.4	2.92	2894.75	0	3493.07	3398.4	97.3%
RxCntToLeds	596.04	1.73	2895	611.13	4103.91	3953.4	96.3%

programmer, while the programs for the FPGA are loaded via a general USB cable.

C. *Flexible Platform Measurement*

The microcontroller and the radio on CC2420DBK are the same as the components on MicaZ, hence the current measurement method of these two components is similar to the measurement of MicaZ as shown in Section IV-B. In this section, the measurement of FPGA is addressed. Since Spartan-3E starter kit provides current sense [11] for FPGA core and I/O pins, a CADDOCK 0.50 Ohm shunt resistor is connected to FPGA core's voltage regulator to measure the power of FPGA core.

Since the execution speed of FPGA is much faster than microcontroller, a compute-intensive algorithm that takes a few seconds to execute on the microcontroller only takes hundreds of nanoseconds on the FPGA. To measure the power/energy consumption in such a short time, we let the same algorithm be continuously executed on FPGA millions of times in order to prolong FPGA's execution time. When executing the repeated algorithm on FPGA, the oscilloscope is able to capture the voltage drop on the shunt resistor that is connected with the core and hence get the core's current. In addition, to measure the actual FPGA's elapsed time on executing the algorithm, we toggle one I/O pin at the beginning point and the end point of the algorithm execution. Then, the energy consumption of FPGA core can be captured.

By the measurement discussed above, the total energy consumption of the actual flexible hardware platform is obtained by the sum of all the components measurement results.

VII. EVALUATION

In this section, evaluation results of PowerSUNSHINE are provided. First, the validation of the simulated results of energy consumption against actual hardware on both fixed and flexible sensor nodes are examined. Second, the scalability of PowerSUNSHINE on simulating fixed and flexible sensor nodes is described. The applications are simulated in SUNSHINE simulator. The testbeds are presented in Fig. 3.

A. *Simulation Fidelity for Fixed Platform*

To evaluate PowerSUNSHINE's power/energy model of fixed platform, we ran several TinyOS applications both on MicaZ OEM boards and in PowerSUNSHINE simulation. All the applications' source code can be checked at [23].

Table II shows both simulation and measurement results of MicaZ nodes running TinyOS applications. The simulation results also provide energy consumption of every hardware component in each application. The first empty-loops application is used to demonstrate that PowerSUNSHINE provides accurate energy consumption of the microcontroller in simulation. In the experiment, the application ends as soon as the microcontroller finishes executing 10^4 empty loops. Other applications are executed for a period of 50 second run. As the table indicates, both simulation and measurement results are within 3.7%. The noise of radio channel, measurement temperature and other testbed's uncertainties may cause the difference between measurement and simulation. This demonstrates that PowerSUNSHINE provides accurate estimation of power/energy consumption for fixed sensor nodes compared with actual hardware. Compared with PowerTOSSIM [6], PowerSUNSHINE offers more reliable results because it uses accurate cycle counts to predict the power/energy consumption of the microcontroller.

B. *Simulation Fidelity for Flexible Platform*

The power/energy model of PowerSUNSHINE is based on calculating power/energy consumption of separate components. For flexible sensor node, it contains microcontroller, radio, and FPGA. Since the power/energy consumption of microcontroller and radio can be accurately profiled by PowerSUNSHINE as shown in Section VII-A, to clearly show the effectiveness of the power/energy model on flexible sensor nodes, we focus on validating the power/energy consumption of FPGA in the following. The power/energy consumption of FPGA core is estimated by incorporating XPower Analyzer.

PowerSUNSHINE's ability of estimating power/energy consumption of FPGA is evaluated via three algorithms: Advanced Encryption Standard (AES) [25] with 128-bit key (AES-128), CubeHash [26] with 512 output bits (CubeHash-512), and Cordic (Coordinate Rotation Digital Computer Algorithm) [27]. Both AES and CubeHash are cryptographic algorithms. Cordic is an algorithm using additions, subtractions and shift operations to switch between polar coordinates and rectangular coordinates in two-dimensional coordinate system.

To validate the simulation results, both AES-128 and Cordic algorithms are continuously executed 10^7 times, and Cubehash-512 is repeatedly executed 10^5 times in simulation and actual hardware. The reason of executing algorithms repeatedly is described in Section VI-C. Fig. 8 presents the simulation and measurement results of the flexible node's energy consumption. As the figure shows, the power/energy dissipation of FPGA consists of static and dynamic power/energy consumption. Static power is related to the device's transistor leakage current while dynamic power results from the actual

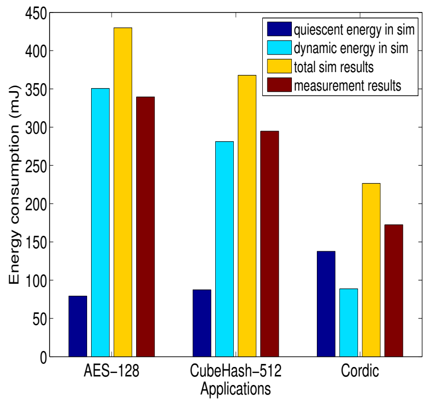


Fig. 8. Validation results of flexible component

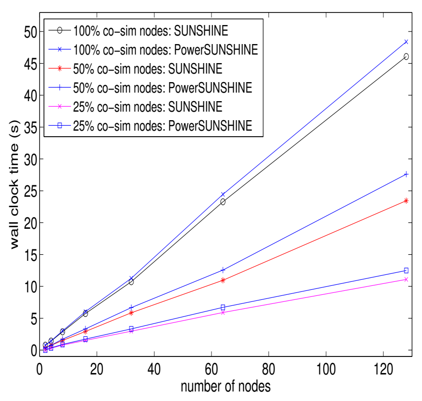


Fig. 9. Scalability of PowerSUNSHINE on simulating MicaZ nodes

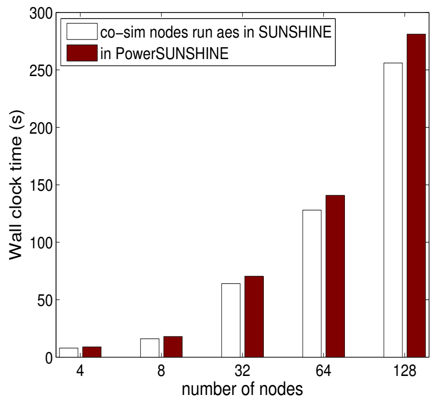


Fig. 10. Scalability of PowerSUNSHINE on simulating flexible sensor nodes

core's activities, such as toggles of gates and signals, value changes of registers, etc.

Fig. 8 shows the power/energy estimation results for FPGA on the flexible nodes. The reason why the simulation results are not as accurate as fixed nodes is due to the different working schemes between microcontroller and FPGA. The current of a microcontroller depends on the microcontroller's states. The microcontroller's different states have corresponding current values; each state's current value has small variations when executing tasks in that state and thus the current value of each state can be optimized as a fixed value. As a result, the power consumption can be easily obtained by the multiplication of the microcontroller's voltage, current and execution time. However, FPGA's power consumption is quite different. FPGA contains logic blocks which are composed of low level circuits. When executing tasks, FPGA's power consumption is due to the current draw of the occupied circuits, especially, charging and discharging of the capacitors. In other words, the current of the FPGA has large variations when the FPGA is executing tasks. Thus, even the most advanced existing FPGA power estimation tools can only give a much rougher prediction comparing to power estimation of fixed components. Since PowerSUNSHINE leverages these existing power estimation tools, it is expected that PowerSUNSHINE's power estimation for FPGA component is not as accurate to the measurement results as its estimation of fixed components. Despite the inaccuracy due to the current limitation of technology, PowerSUNSHINE's slight overestimation for flexible FPGA components is still accurate enough to serve as a conservative guideline for flexible sensor platform designs as shown in Fig. 8.

C. Scalability

Since PowerSUNSHINE is built on top of SUNSHINE, in order to show PowerSUNSHINE's scalability, it is wise to show the scalability of PowerSUNSHINE together with SUNSHINE. As PowerSUNSHINE can estimate both fixed and flexible sensor nodes' power consumption, we used two applications to show PowerSUNSHINE's scalability.

The first application is used to evaluate MicaZ's power/energy consumption. The application is same as the one setup in our previous described in [3]: nodes are randomly distributed from 2 to 128 and are paired to communicate with each other. The simulation ends when all the reception nodes receive a packet from its neighbor. The number of co-sim nodes is varied from 25% to 100%. In Fig. 9, wall clock time represents the simulator's run time. The time overhead of PowerSUNSHINE is very small compared to SUNSHINE. Therefore, it is feasible to use PowerSUNSHINE to estimate fixed nodes power/energy consumption in large sensor networks.

The second application is to demonstrate PowerSUNSHINE's scalability on simulating flexible sensor nodes. The application is similar as the first one except only 25% nodes are emulated as flexible co-sim nodes. In addition, these co-sim nodes let their FPGAs run AES-128 algorithm to encrypt the packet and then send the encrypted packet to their neighbors. The simulation ends when all the neighbors receive the packet. As shown in Fig. 10, both SUNSHINE and PowerSUNSHINE are a little slow when simulating 128 nodes. This is reasonable because SUNSHINE needs to simulate the sensor nodes' behaviors of both software (microcontroller and radio) and hardware (FPGA). SUNSHINE has to spend much time on capturing detailed and accurate information of the flexible sensor nodes. Fig. 10 also indicates that PowerSUNSHINE only takes a little more time than SUNSHINE when capturing the power/energy consumption of flexible sensor nodes.

VIII. Discussion and Future Work

In this paper, we provide one flexible sensor platform for validation. However, PowerSUNSHINE is not limited to this particular platform. Other flexible hardware architecture can also be configured and emulated in PowerSUNSHINE, such as placing FPGA between microcontroller and radio, using a different microcontroller, radio and FPGA, etc. In addition, while in our current simulation and testbed, the communication protocol between microcontroller and FPGA is SPI, other protocols such as I^2C, UART, and parallel, can also be used in platform construction and be evaluated by PowerSUNSHINE.

Currently, we connect Spartan-3E with CC2420DBK to validate PowerSUNSHINE. It is worth to note that Spartan-3E FPGA is SRAM-based which is not low-power oriented. If we change Spartan-3E FPGA to low-power oriented antifuse-based FPGA, the total energy would be significantly decreased. In addition, the communication energy consumption would be much smaller if all the hardware components are placed on one printed circuit board (PCB), and are connected with each other through etched wires. In the next step, we will make our own PCB that mainly contains a microcontroller, a radio and a low power FPGA. We will use PowerSUNSHINE to explore potential power savings from flexible sensor platforms and use the designed board to do validation.

We used constant power supply to measure the nodes power consumption. For accurately predicting the lifetime of a node, PowerSUNSHINE would need a model for the battery voltage as a function of energy consumption and time. We will study the battery's characteristics to investigate how to prolong the network's lifetime.

In this paper, several applications are developed to demonstrate that using flexible sensor nodes can help network meet real-time requirement. These applications are working in both simulation and real hardware. We are working on developing more applications and show more advantages of flexible sensor nodes.

IX. Conclusion

In this paper, we developed PowerSUNSHINE to accurately estimate the power/energy consumption of both fixed and flexible sensor nodes in wireless networks. PowerSUNSHINE is based on SUNSHINE, a flexible hardware-software emulator for WSNs. To estimate power/energy consumption of flexible sensor platforms, PowerSUNSHINE establishes power/energy models of fixed components, incorporates hardware power analyzer for reconfigurable hardware components and finally utilizes the simulation data provided by SUNSHINE to eventually derive accurate power estimation results. Two testbeds of MicaZ and a flexible sensor node are built for validation. Our extensive experiments on the testbeds show that PowerSUNSHINE provides accurate simulation results for power/energy consumption. PowerSUNSHINE also scales to simulate large sensor networks and hence serves as an effective tool for wireless sensor network design.

Acknowledgment

This work is supported by National Science Foundation award CCF-0916763.

References

[1] J. Portilla, T. Riesgo, and A. de Castro. A Reconfigurable Fpga-based Architecture for Modular Nodes in Wireless Sensor Networks. 3rd Southern Conference on Programmable Logic, 2007.

[2] Y. E. Krasteva. J. Portilla, E. de la Torre, and T. Riesgo. Embedded Run-time Reconfigurable Nodes for Wireless Sensor Networks Applications. IEEE Sensors Journal, to be published during 2011.

[3] J. Zhang, Y. Tang, S. Hirve, S. Iyer, P. Schaumont, and Y. Yang. A Software-Hardware Emulator for Sensor Networks. In *IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2011.

[4] C. C. Chang, D. J. Nagel, and S. Muftic. Assessment of Energy Consumption in Wireless Sensor Networks: A Case Study for Security Algorithms. In *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, 2007.

[5] M. Tancreti, M. S. Hossain, S. Bagchi, and V. Raghunathan. Aveksha: A Hardware-Software Approach for Non-intrusive Tracing and Profiling of Wireless Embedded Systems. In *9th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2011.

[6] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, and M. Welsh. Simulaitng the power consumption of large-scale sensor network applications. In *Proc. the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[7] O. Landsiedel, K. Wehrle, and S. Gotz. Accurate Prediction of Power Consumption in Sensor Networks. In *IEEE Workshop on Embedded Networked Sensors (EmNets)*, 2005.

[8] S. Iyer, J. Zhang, Y. Yang, and P. Schaumont. A Unifying Interface Abstraction for Accelerated Computing in Sensor Nodes. 2011 Electronic System Level Synthesis Conference, 2011.

[9] Power Calculators for Actel FPGAs. http://www.actel.com/techdocs/calculators.aspx.

[10] PowerPlay Early Power Estimators (EPE) and Power Analyzer. http://www.altera.com/support/devices/estimator/pow-powerplay.jsp.

[11] Spartan-3E. http://www.xilinx.com/support/documentation/spartan-3e.htm.

[12] CC2420DBK user manual. http://focus.ti.com/lit/ug/swru043/swru043.pdf.

[13] Xilinx Logic Design: XPower. http://www.xilinx.com/products/technology/power/index.htm.

[14] B. L. Titzer, K. D. Lee, and J. Palsberg. Avrora: Scalable sensor network simulation with precise timing. In *Proc. of the 4th Intl. Conf. on Information Processing in Sensor Networks (IPSN)*, pages 477–482, 2005.

[15] nesC: A Programming Language for Deeply Networked Systems. http://nescc.sourceforge.net.

[16] Texas instruments cc2420 radio transceiver. http://focus.ti.com/docs/prod/folders/print/cc2420.html.

[17] OEM development kit. http://bullseye.xbow.com:81/Products/Product_pdf_files /Wireless_pdf/OEM_Development_Kit_dis.pdf.

[18] WaveSurfer 24Xs-A. http://www.lecroy.com/files/pdf/LeCroy _WaveSurfer_XS-a_Datasheet.pdf.

[19] MP900 and MP9000 Series Kool-Pak Power Film Resistors TO-126, TO-220 and TO-247 Style. http://www.caddock.com/Online_catalog/Mrktg_Lit/MP9000_Series.pdf.

[20] Tenma 72-6905 datasheet. http://datasheet.octopart.com/72-6905-Tenma-datasheet-92910.pdf.

[21] Atmega128/L Datasheet. http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf.

[22] Xilinx Power Tools Tutorial. http://www.xilinx.com/support/documentation /sw_manuals/xilinx11/ug733.pdf.

[23] SUNSHINE simulator source codes. http://sourceforge.net/projects/sunshine-sim/.

[24] GEZEL: Hardware/Software Codesign Environment. http://rijndael.ece.vt.edu/gezel2/.

[25] Advanced Encryption Standard. http://en.wikipedia.org/wiki/Advanced_Encryption_Standard.

[26] CubeHash. http://en.wikipedia.org/wiki/CubeHash.

[27] The Cordic Algorithm. http://www.andraka.com/cordic.htm.