Fast and Accurate Simulation of Biomonitoring Applications on a Wireless Body Area Network

Kathy Dang NGUYEN Ioana CUTCUTACHE Saravanan SINNADURAI Shanshan LIU
Cihat BASOL Edward SIM Linh Thi Xuan PHAN Teck Bok TOK
Lin XU Francis Eng Hock TAY Tulika MITRA Weng-Fai WONG

***Abstract*— Remote monitoring of the vital signs of a patient at home is becoming increasingly important. Wireless body area sensor networks (BAN) provide a convenient platform for health monitoring. A typical biomonitoring application running on wireless BAN has stringent timing and energy requirements. In this context, a cycle accurate simulator of biomonitoring application running on wireless BAN network is a critical tool for the developer. In this paper, we present a SystemC-based fast simulator that can provide accurate timing and energy estimated for health monitoring applications on wireless BAN, which is then used to strategize the optimizations towards critical functions and/or components. Our case study with MEMSWear-Biomonitoring application shows the accuracy and utility of our simulator.**

I. Introduction

Remote monitoring of the vital signs for home care is important especially for chronic and elderly patients. It is important to continuously monitor their vital signs so as to detect the onset of any critical condition early on and inform the doctor accordingly. Such a biomonitoring system is usually implemented over a body area wireless sensor network (BAN) consisting of a set of inter-communicating sensors attached to the human body, together with a gateway that collects and transmits data to hospitals.

The development of biomonitoring systems faces a conflicting trade-off. Any emergency or anomaly in the vital signs of the patient has to be processed quickly, and hence the biomonitoring application requires high processing power. It is also desirable to maximize battery life. An optimal design is dependent on both hardware and software, and it is only by marrying the application to the appropriate platform can one design a fast and efficient system.

A key component of this development process is the ability to measure/estimate the timing and energy consumption of the application on the platform. While it is possible to measure the timing and energy by running the application on the platform, this approach has two major drawbacks. First, most system designs involve exploring a design space in choosing the appropriate platform, i.e., the platform may not be available. Second, the measurement can only be performed at a coarse-grained level such as energy consumption of the entire platform. This granularity is not helpful for the developer to detect and optimize the critical components.

A fast but cycle-accurate simulator can overcome these drawbacks. It allows the developer to determine accurately the processing and energy performance of individual module in the application, under different configurations. This information can then be used to tune the system. Such flexibility facilitate quick design space exploration.

In this paper, we present a fast simulator for biomonitoring applications running on wireless BAN. Our work is part of Embedded and Hybrid System II program in Singapore, centered on the development of a Body Sensor Node (BSN) system and of relevant health-care applications. The project is carried out by Singapore Agency of Science Technology and Research (A*STAR) with collaborations from Institute of Infocomm Research (I^2R), Institute of Microelectronics (IME), Nanyang Technological University (NTU) and National University of Singapore (NUS).

Our simulator provides a high level executable model of wireless body area network platform. Our modeling is based on SystemC, an IEEE standard language for system level design of embedded systems. Our simulator provides detailed breakdown of the timing and energy behavior of individual software and hardware components in the system. For example, the workload on the sensors can be tuned according to their energy consumption characteristics so as to obtain optimal system efficiency. The simulation speed is fast thanks to the modeling level and techniques that we use to optimize simulation speed. The simulator has been tested with a medical application that monitors the patient well-being by measuring the ECG and the SpO_2 blood level [14].

II. Related Work

Recently there is a lot of interest in employing wireless body area networks for biomonitoring [6], [7], [10], [14]. All these studies have suggested that successful deployment is possible if the devices consume very little energy and can respond to events in a timely manner [9], [8].

Most previous work on high-level modeling or developing simulator for sensor networks [3], [13] only model the CPU at instruction set level. However, modeling of the peripherals is necessary in order to produce accurate performance and energy numbers. This is because for sensor network applications, the data input is done mostly through peripheral interrupts. An exception of the above is [15] that models both

This research has been supported by A*STAR under Project Number 052-118-0061 "EASEL: Engineering Architectures and Software in the Embedded Landscape."

The authors are with National University of Singapore. Contact author: kathyngu@comp.nus.edu.sg

the sensor node (including peripherals and software tasks) and sensor network by using finite state machines. This is a higher level of abstraction compared to our approach.

Another simulator that has modeled both the CPU and the peripherals of sensor nodes is Avrora simulator [1]. However, at this moment, Avrora does not support the architecture that we are targeting, namely BSN board with MSP430 micro-controller. In addition, with our implementation in SystemC, we aim to support better usability of components and faster simulation speed; thus providing better support for design space exploration. A commercial tool [5] bears some similarity with our work. However, we think that developing our own simulator based on SystemC gives us more freedom to configure and explore the architecture design space, not only by changing the configuration of the components in the system but also by replacing them.

III. Wireless BAN Simulator

In this section, we describe our SystemC-based simulator for wireless BAN platforms. The simulator takes in any application code written in NesC. The NesC code is compiled with TinyOS to generate executable code for the BSN mote. We simulate the execution of the latter code. Our goal is to perform fast but cycle-accurate execution-driven simulation of the application on the target platform to obtain accurate timing and energy estimates.

A. SystemC

SystemC is an IEEE standard system level design language that can be used to model platforms at different levels of abstraction. It is built entirely in standard C++ and comes with a simulation kernel that manages the execution and progress of time for all components in a system. The designers have the freedom to choose the level of abstraction that best suits their needs. We have chosen *transaction level abstraction* to model the communication among the components in our system. This is the level where communication among the components is done through function calls only, without synchronization and pins details. To enable faster computation, we exploy techniques such as minimizing context switches and using abstract constructs/datatypes. Our simulator also does not model all details at each clock cycle. Instead, delays are inserted at appropriate places to capture timing information and calculate energy consumption accurately.

SystemC allows separate modeling of computation and communication components. Thus it allows re-configuration and plug-and-play of components, thus allowing convenient design space exploration.

B. Simulator structure

Figure 1 shows the structure of our simulator. It consists of five main components: a micro-controller module, a ChipCon 2420 module, a sensor module, a power monitor and a base station. The micro-controller present in BSN IC mote is Texas Instruments MSP430. The micro-controller module incorporates a CPU module, a clock module, RAM, Flash and other peripherals. The power monitor is not part of

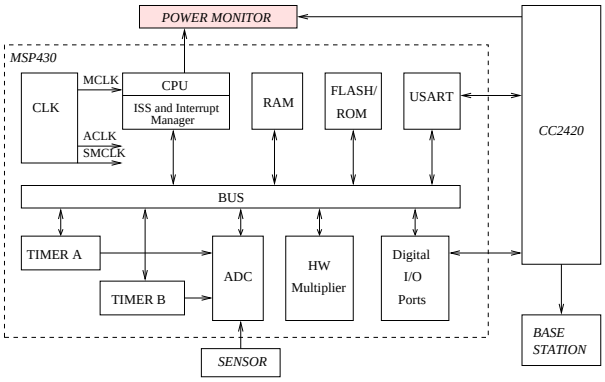


Fig. 1. The structure of the simulator.

the MSP430 architecture; it is designated solely to monitor power consumed by each component. To handle a network of sensors, the simulator creates multiple instances of sensor, MSP430 and CC2420 modules. Following is a brief description of the important modules.

1) Clock Module: The clock module includes three clock signals, a Master Clock (MCLK), a Submaster Clock (SMCLK) and a Auxiliary Clock (ACLK), and three sources, a low-frequency oscillator (LFXT1CLK) that operates at 32,768Hz, an optional high-frequency oscillator (XT2CLK) that operates in a range 450k–8MHz and an internal digitally controlled oscillator (DCOCLK) that operates at approximately 1MHz. MCLK and SMCLK can be wired to any of the three sources while ACLK needs to be wired to LFXT1CLK. MCLK is the main source for the microcontroller. SMCLK is the source for most peripherals and ACLK is for slower subsystems to conserve power.

2) CPU Module: The CPU model includes an instruction set simulator, interrupt manager and 16 registers. The MSP430 incorporates a 16-bit RISC CPU. The instruction set simulator handles 27 core instructions and 24 emulated instructions of this RISC CPU. The simulator also implements seven possible addressing modes.

During the execution of an instruction, the simulator updates the contents of the affected registers, memory and flags in the status register. In addition, it inserts cycle-accurate delay based on the instruction's opcode, source and destination addressing mode. The CPU module also features an interrupt manager that services any pending interrupts after each instruction is executed. Moreover, the interrupt manager can handle interrupt priorities and nested interrupts.

3) Timers: MPS430 includes two 16-bit timers: Timer A and B. Both timers can be used in four operating modes (up, continuous, up/down, stop). The timers have interrupt capabilities and the architecture includes a number of configurable capture/compare units: three for Timer A and seven for Timer B. Timer B has configurable length and can be programmed as 8, 10, 12 or 16 bits. The clock source is selectable via the control registers. The capture mode is used to record time events while the compare mode is used to generate interrupts and output signals at specific time intervals. Our implementation for each timer module consists of a SystemC thread. The thread waits for any of the

following events: (1) a capture signal, (2) counter reaching the next compare value, or (3) reconfiguration of the timer by modifying the control registers. The waiting delay is computed by considering the selected clock frequency and the next compare value. On capture, compare or overflow, the timers can request interrupts.

4) USART: The universal synchronous/asynchronous receive/transit (USART) peripheral interface supports two serial transmission modes, namely UART for asynchronous data transfers and SPI for synchronous data transfers. The USART is connected to the CC2420 to transfer data to and from the radio channels. Our simulator abstracts this connection but it contains sufficient details to capture the timing and power consumption information.

5) CC2420: Our simulator provides an abstracted implementation of the CC2420 radio chip. Only the basic functionalities are supported and the focus is to capture timing and power consumption information. The CC2420 model includes 33 16-bit configuration registers for status, 15 command strobe registers for instructions, 128-byte transmit(TX) RAM and 128-byte receive(RX) RAM for outgoing and incoming data packets respectively.

6) Sensor Module: The SpO_2 sensor is a TSL257 light-to-voltage converter produced by Texas Advanced Optoelectronic Solutions. The sensor converts light into voltage output through the use of a photodiode and a operational amplifier. The dynamic power dissipation depends on the input [4]. The energy model involves non-trivial calculations, and its integration with our framework is still in progress.

7) Power monitor: The power monitor module is designated to accumulate the total energy consumed by each component in the BSN mote. Every change in operating modes in any component will update the power monitor. The power monitor, in turn, will obtain the simulation time spent in the previous operating mode and compute the energy consumed during that period. By accumulating the energy consumed by the individual components and by the BSN mote, when the application completes execution, we obtain a breakdown of total energy consumption.

The simulator also performs profiling at functional level. Each time a function is invoked, the energy consumed while executing the instructions within that function is accumulated. Hence, when the application completes execution, we obtain a breakdown of energy consumption by each function. This information can identify candidate functions that might be optimized to reduce energy consumption.

C. Battery Life Analysis

Battery life is dependent on the comsumption rate as well as the physical constraints such as battery size and weight. In our simulator, we incorporated a battery analysis to estimate the battery lifetime under an application workload for a selected battery model. The estimation provides a means to understand battery behavior to facilitate in tweaking and optimizing the power consumption of the system.

Because the output energy of a battery depends on its discharged loads and it is also not possible to exhaust all energy stored in the battery [11], it is necessary to consider the discharge profiles and non-linearities of the battery when estimating its lifetime. Additionally, the speed in computing the lifetime is also a critical factor.

A number of battery models have been proposed. In this work, we adopt the analytic model by Rakhmatov *et al.* [12] for its flexibility in modeling diverse batteries types and its good balance between accuracy and efficiency. With this model, we are able to predict battery lifetime for any discharge profile and battery model. The integration of this model with our simulator is in progress.

D. GUI

The Graphical User Interface (GUI) is designed to ease the use of our simulator. First, the simulator can display application input and output in real time. RED, IR (red and infrared light input signal for SpO_2 computation) and ECG (output received at the gateway) signals are displayed in wave form; other output like heart rate, SpO_2 and blood pressure are also displayed. The left snapshot in Figure 2 shows the interface while the simulation is in progress.

Once the simulation completes, the energy consumption of the different software functions (Figure 2(ii)) and hardware components, as well as estimated battery life, are displayed. In addition, the simulator can report the time consumed by each function. All this information provide hints to the designers for optimizing both timing and energy performance of the application.

IV. Experimental Evaluation

In this section, we perform functional validation of the simulator through a case study of the MEMSWear-Biomonitoring application [14]. We run the application both on the actual hardware platform and on the simulator, with the same input taken in from memory, not sensors because we have not modeled the sensors yet. We compare and detect no difference in the heart rate, SpO_2 and blood pressure waveforms generated by actual execution on the hardware and those by the simulator.

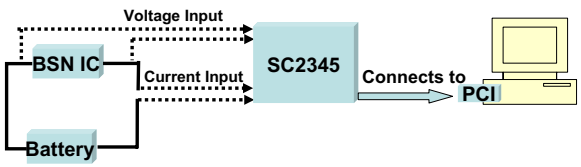
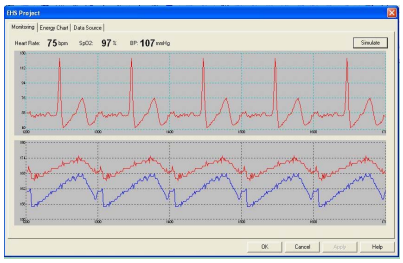
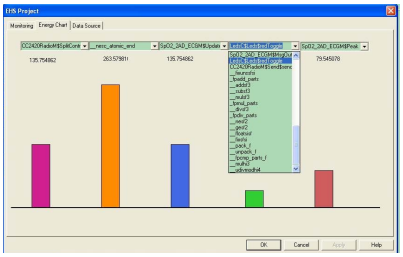


Fig. 3. Experimental set-up block diagram.

1) Energy Measurements: In order to validate the energy estimated by the simulator, we first measure the actual energy consumed by the BSN mote while running the application. We design a special experimental setup for this purpose. We use National Instruments SC2345 board along with a PCI-based data acquisition board to collect the voltage and current readings from the BSN mote, as shown in Figure 3. We connect the voltage input of SC2345 to the BSN mote in parallel and the current input of SC2345 in serial to the BSN mote. The data are forwarded to the PCI board attached to the PC, which runs LABVIEW software [2] to analyze the data.



(i)



(ii)

No of samples	Simulated power (μW)	Measured power (μW)
1000	51,633	52,485
3000	51,700	52,485
5000	51,900	52,485

(iii)

Fig. 2. Simulator GUI: snapshots showing (i) simulator in progress and (ii) energy consumption by functions. (iii) Power consumption for BSN board.

The voltage and current data are multiplied and summed up over time to calculate the energy consumption.

2) Simulator Accuracy: To compare the measured and simulated power usage, we used three sets of inputs, each containing different number of samples, to the biomonitoring application. The results are depicted in Figure 2(iii). In addition, we have determined the energy consumed by the CC2420 individually, which consumes 97% of energy consumed by the whole BSN board. We observe that the BAN simulator can predict the energy usage of the application rather accurately. In addition, the differences between the actual and simulated execution time range from 0.5–11% (average 3.5%) depending on the sampling rates. This verifies our claim that the BSN motes's timing/energy behavior has been well captured in our simulator.

We also perform additional experiments by modifying the sampling rate of the application. However, we do not observe any significant variations in the total power consumption. This observation can be attributed to the radio chip. By default, the radio chip is always in receiving mode and this increases the overall energy consumption. The energy used by the application during its computations is rather insignificant compared to the energy used by the radio chip. This information is vital to application developers because powering down the radio when it is not in use can conserve energy. However, the developer has to strike a balance between conserving energy and compromising deadlines due to the delays in powering up the radio chip. Our simulator can be of immense help in testing and obtaining optimal performance parameters to optimize the application.

Our simulator is quite fast: it incurs 4–17$\times$ slowdown compared to native execution, depending on the sampling rate (50–200Hz). Of course the slowdown is highly dependent on the application. For the current application, the CPU is in inactive mode for a short amount of time each cycle, thus saving some simulation time.

V. Conclusions

In this paper, we have described a fast simulator for biomonitoring applications implemented over a body area network. The simulation enables a developer to determine the energy and time requirements of individual components in the system. Experimental data, obtained by simulating an ECG and SpO_2 monitoring application over an Imperial College BSN architecture, shows that our simulator provides accurate readings. With this information, a design space exploration is possible at both the hardware and software levels. Our future work include providing timing and energy simulation for the base station (e.g. PDA) and providing more support for design space exploration.

Acknowledgements: We would like to thank Samarjit Chakraborty, Abhik Roychoudhury, P.S. Thiagarajan for their advice and Zhenxin Sun for preparing some of the figures.

References

[1] Avrora - The AVR simulation and analysis framework. http://compilers.cs.ucla.edu/avrora/, 2007.

[2] LABVIEW software. http://www.ni.com/labview/, 2007.

[3] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in network simulation. In *Procs. IEEE HLDVT*, 2000.

[4] Giorgio Casinovi and Chad Young. Estimation of power dissipation in switched-capacitor circuits. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 22(12):1625–1636, 2003.

[5] N. Fournel, A. Fraboulet, G. Chelius, E. Fleury, B. Allard, and O. Brevet. Worldsens: from lab to sensor network application development and deployment. In *Procs. 6th International Conference on Information Processing in Sensor Networks*, pages 551–552, 2007.

[6] K. Hung, Y. T. Zhang, and B. Tai. Wearable medical devices for tele-home healthcare. In *Procs. 26th Annual International Conference on the IEEE EMBS*, 2004.

[7] G. Y. Jeong, K. H. Yu, and Kim. N. G. Continuous blood pressure monitoring using pulse wave transit time. In *International Conference on Control, Automation and Systems (ICCAS)*, 2005.

[8] Y. Liang, A. Roychoudhury, and T. Mitra. Timing analysis of body area network application. In *7th International Workshop on Worst-Case Execution Time Analysis (WCET)*, 2007.

[9] Y. Liu, B. Veeravalli, and S. Viswanathan. Critical-path based low-energy scheduling algorithms for body area network. In *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2007.

[10] C. Otto, J. P. Gober, R. W. McMurtrey, A. Milenkovic, and E. Jovanov. An implementation of hierarchical signal processing on wireless sensor in tinyos environment. In *Procs. 43rd Annual Southeast Regional Conference - Volume 2*, 2005.

[11] D. Panigrahi, S. Dey, R. Rao, K. Lahiri, C. Chiasserini, and A. Raghunathan. Battery life estimation of mobile embedded systems. In *Proc. 14th International Conference on VLSI Design*, 2001.

[12] D. Rakhmatov, S. Vrudhula, and D.A. Wallach. A model for battery lifetime analysis for organizing applications on a pocket computer. In *Proc. IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2003.

[13] G. Sachdeva, R. Dömer, and P. Chou. System modeling a case study on a wireless sensor network. Technical Report CECS-TR-05-12, University of California, Irvine, 2005.

[14] F. E. H. Tay, D. G. Guo, L. Xu, M. N. Nyan, and Yap K. L. Memswear-Biomonitoring System for Remote Vital Signs Monitoring. In *Procs. 4th International Symposium on Mechatronics and its Applications (ISMA07)*, 2007.

[15] K. Virk, K. Hansen, and J. Madsen. System-level modeling of wireless integrated sensor networks. In *Procs. 2005 International Symposium on System-on-Chip*, 2005.