A Novel Approach for Ultra Low-Power WSN Node Generation

Adeel Pasha, Steven Derrien, Olivier Sentieys

A Novel Approach for Ultra Low-Power WSN Node Generation

Muhammad Adeel Pasha, Steven Derrien and Olivier Sentieys

University of Rennes1, IRISA/INRIA
Campus de Beaulieu, Rennes
FRANCE

E-mail: adeel.pasha@irisa.fr steven.derrien@irisa.fr
olivier.sentieys@irisa.fr

Abstract — **Wireless Sensor Network (WSN) technology is now emerging with applications in various domains of human life e.g. medicine, environmental monitoring and military surveillance etc. WSN systems consist of low-cost and low-power sensor nodes that communicate efficiently over short distances. It has been shown that power consumption is the biggest design constraint for such systems. Currently, WSN nodes are being designed using low-power microcontrollers. However, their power dissipation is still orders of magnitude too high and limits the wide-spreading of WSN technology. In this paper, we propose an alternative approach that uses hardware specialization and power-gating to generate distributed hardware micro-tasks. We target control-oriented tasks running on WSN nodes and present, as a case study, a lamp-switching application. Our approach is validated experimentally and shows prominent power gains over software implementation on a low-power microcontroller such as the MSP430.**

Keywords — **Hardware Specialization, Low-Power Micro-Architecture, WSN**

I Introduction

Recent advancements in micro-electro-mechanical-systems (MEMS) technology, wireless communication, and digital electronics have facilitated the development of low-cost, low-power, multi-functional sensor nodes that are small in size and communicate efficiently over short distances. Systems of 1000s or even 10,000s of such nodes are anticipated and can revolutionize the way we live and work. A Wireless Sensor Network (WSN) is composed of a large number of sensor nodes, that are densely deployed either inside a region of interest or very close to it. Each node consists of processing capability (one or more microcontrollers with associated memory (RAM/Flash)), communication capability (an RF transceiver) and can accommodate various sensors and actuators.

Power consumption has been realized as the biggest constraint in the design of a WSN node. Since the nodes must be low-cost and small in size [1], it is not possible to equip them with a huge source of energy. To make the situation worse, WSN nodes may have to work unattended for long durations due to difficult access to them or a huge number of nodes. As a result, they must survive with self-harvested (e.g. solar cells) or non-replenishing (e.g. batteries) sources of energy. All these restrictions toward energy retrieval make power consumption the most important design parameter.

In recent years, WSN nodes have been designed using low-power microcontrollers such as the MSP430 [2] from Texas Instrument or ATMega128L [3] from Atmel Corporation. These programmable processors share common characteristics such as a reasonable processing power with low power consumption at a very low cost. However, power dissipation of these devices is still orders of magnitude too high for application domains such as WSN, since these systems expect sensor nodes to operate with extremely limited energy resources for very long time periods (months if not years). Moreover, because these nodes remain idle during most of their lifetime, their static power consumption plays a major role in their actual energy budget.

In such situations, the only way to further improve the energy efficiency of such a system is to customize its design to the application at hand. The approach consists in implementing each task of a control-oriented application graph on a power-gated specialized hardware architecture (called hardware *micro-task*). This architecture is in the

form of a minimalistic datapath controlled by a custom Finite State Machine (FSM) and is being automatically generated from a task specification in C, using an ASIP-like retargeted design environment. The approach results in improving both the dynamic (thanks to hardware specialization) and static (thanks to power-gating) power of the WSN node.

In this paper, we are investigating the application of this hardware specialization approach from WSN perspective. We propose, as a case study, a lamp-switching application in which a transmitting node demands a receiving node to switch on/off its lamp if a button is pressed at transmitter end.

The main contribution of this article lies in investigating the power benefits of *power-gated hardware micro-tasks* based approach. A simple yet realistic case study of a WSN example also serves as an experimental validation that the approach is conceivable for real-life WSN applications. We also provide the SPICE transistor-level timing simulation results to show that the on/off-switching delays of the power-gated micro-tasks are within the acceptable range and even better than that of traditional low-power microcontrollers such as the MSP430.

Our experiments show that dynamic power savings of one to two orders of magnitude can be obtained for different control-oriented tasks of our application (w.r.t. a low-power MCU such as the MSP430). Moreover, since the tasks are power-gated, their static power consumption will be virtually zero when the WSN nodes will be in sleep mode.

The rest of this paper is organized as follows. We start by presenting the related work in Section II and describe thoroughly our proposed case study in Section III. In Section IV, we present experimental results which confirm the validity of the approach. Finally, conclusion and future research directions are drawn in Section V.

II Related Work

In the last decade, a wide range of applications for sensor network have been developed. Some of the application areas are environment, military, health, and security. WSN may consist of many different types of sensors such as seismic, low sampling rate magnetic, thermal, visual, infrared, acoustic and radar. These sensors are able to monitor a wide variety of ambient conditions such as temperature, humidity, lightning, pressure, and vehicular movements etc [4]. This section details the literature study of some of such WSN applications. Later in the section, we highlight some application benchmarks that have been proposed for WSN. Finally, we present some power optimization efforts done at micro-architectural and operating system level in the context of WSN.

a) Important WSN applications

Environmental monitoring is an important application of WSN. In reference [5], a habitat monitoring system is discussed. Similarly, forest fire detection and prevention [6], and detection of volcanic eruptions [7] are other examples of environment-monitoring WSN systems.

WSN can also be used as an integral part of military command, control, communication, computing, intelligence, surveillance, reconnaissance and targeting (*C4ISRT*) systems [8]. The rapid deployment, self-organization and fault tolerance are some characteristics that make WSN a very promising sensing technique for military *C4ISRT* systems.

Moreover, the benefits of WSN have also been proved in other domains of human life such as health monitoring and home applications [9, 10].

b) WSN application benchmarks

We have seen in this section that WSN applications consist of a heterogeneous nature as they are pretty different in their overall goals to be achieved. However, the basic tasks performed in a WSN node are quite similar. These tasks are: sensing a certain phenomenon, gathering its relevant data and forwarding it to a base-station in a pre/post-processed state. Several attempts have been conceived to profile the workload of a generic WSN node. Two of the recent application benchmarks for WSN are SenseBench [11] and WiSeN-Bench [12]. Both of them have tried very well to cover the general applications and algorithms that can be run on a typical WSN node.

c) Low-power MCUs and operating systems

As far as power optimization of WSN domain is concerned, many research efforts have been made. These works cover all the design aspects of a WSN from application layer of the communication stack to the physical layer (e.g. efficient routing algorithms, low-power medium access control (MAC) protocols etc.). However, since the focus of our research work is the micro-architectural level, we try to summarize the characteristics of low-power micro-controllers (such as the MSP430 and the ATMega128L) that have been developed for low-power applications and the light-weight operating systems running on them.

The common characteristics of such MCUs are: a simple datapath (8/16-bit wide), a reduced number of instructions (only 27 instructions for the MSP430), and several power saving modes that allow the system to select the best compromise between power saving and reactivity (i.e. wake-up time). These processors are designed for low-power operation across a range of embedded system application settings but are not necessarily well-suited to the event-driven behavior of WSN nodes as they are based on a general purpose, monolithic compute engine.

For example, Mica2 mote [13] has been widely used by the research community. It is a complete WSN node based on a ATmega128L MCU, with I/O peripherals, an RF transceiver and sensor devices. Measurements of Mica2 show that its MCU consumes an average 8 mA of current when active and approximately 15 μA when in low-power mode. The MSP430F1611 used by Hydrowatch platform consumes a nominal 500 μA [14] whereas the latest version of MSP430 (MSP430F21x2) consumes approximately 8.8 mW (@ 16 MHz).

It is an acknowledged fact that the power budget of a WSN node that would rely only on energy harvesting technologies is estimated to be around 100 μW [15]. Comparing this constraint with that of current MCUs power consumption profiles clearly drives us toward alternative architectural solutions (e.g. hardware specialization of the system).

Moreover, most of currently used MCU packages include a limited amount of RAM (only a few hundred Bytes to a few kilo-Bytes). This limited amount of storage resources poses great challenges to the software designers since both the user application and operating system must work with this very small amount of memory.

As a consequence, there have been several attempts to reduce the complexity of the operating system (OS) on these devices. In particular, many approaches have been proposed to reduce the overhead caused by dynamic scheduling of the threads by using alternative concurrency computational models. For example, TinyOS [16] is built upon an event-driven approach, without explicit thread management, and Contiki [17] proposes a simplified thread execution model (named *protothread*), in which preemption can only occur at specific points in the task control flow.

III Proposed approach

The section starts by explaining the case study used in the current work and later on, outlines the notion of micro-task and the generic architecture of a WSN node built using the proposed approach.

a) Proposed case study

WSN applications, being event-driven in nature, can be represented as Tasks Flow Graphs (TFGs) where a task execution is triggered by *events*, be they external or produced by another task.

This section highlights the important control tasks of our lamp-switching WSN application during transmit as well as receive mode. The control-flow of the proposed node is based on RICER (Receiver Initiated CyclEd Receiver) MAC protocol [18]. Briefly speaking, data transmission by a transmitter node is initiated upon reception of a wake-up beacon from the desired receiver node. Fig. 1 (a and b) shows the TFGs of the proposed node in transmit and receive mode, respectively.

a).1 Control tasks running on a WSN node in transmit mode

Here are the basic control tasks running on a our WSN node in transmit mode:

- Wait for wake-up beacon: Upon reception of an external event (*buttonPushed*) or an *ackNOK* event, the transmitter node waits for the wake-up beacon from the desired receiver node according to RICER protocol. We have written a C-function, called `waitForBeacon()`, that actually starts a timer and reads the data packets received by its RF transceiver through Serial Peripheral Interface (SPI) bus. In our example, we are using CC2420 radio chip from Texas Instrument [19] as RF transceiver. If the transmitter receives the required wake-up beacon before the timer expiry, it generates a *beaconReceived* event. Otherwise, a *timeOut2* event is generated by the task.
- Sending data: The next task is data transmission that is described in `sendData()` function. This function writes data frame to the physical interface of the radio transceiver through SPI bus and generates a *dataSent* event.
- Receiving acknowledgment: After sending data, the transmitter node waits for an acknowledgment frame from the receiver using `receiveAck()` function. If it receives the acknowledgment correctly, it generates an *ackOK* event, otherwise an *ackNOK* event is generated.
- Shutting down the transceiver: Upon reception of *ackOK* or *timeOut2* event, the transmitter node will shut down its RF transceiver to save energy. This is done through `shutDownRadio()` function, that sends appropriate signals to RF transceiver to shut it down.

a).2 Control tasks running on a WSN node in receive mode

The control tasks running on a WSN receiver node of our case study are as follows:

- Sending a wake-up beacon: Our proposed WSN node periodically broadcasts a wake-up beacon to announce the neighbors to initialize a communication. This control task waits for an external event *timerEvent* for its activation. This external event is periodically generated by a hardware timer. The corresponding C-function for the task is described in `sendBeacon()` function that generates a *beaconSent* event.
- Receiving and analyzing data: After sending the beacon, the receiver waits for the data

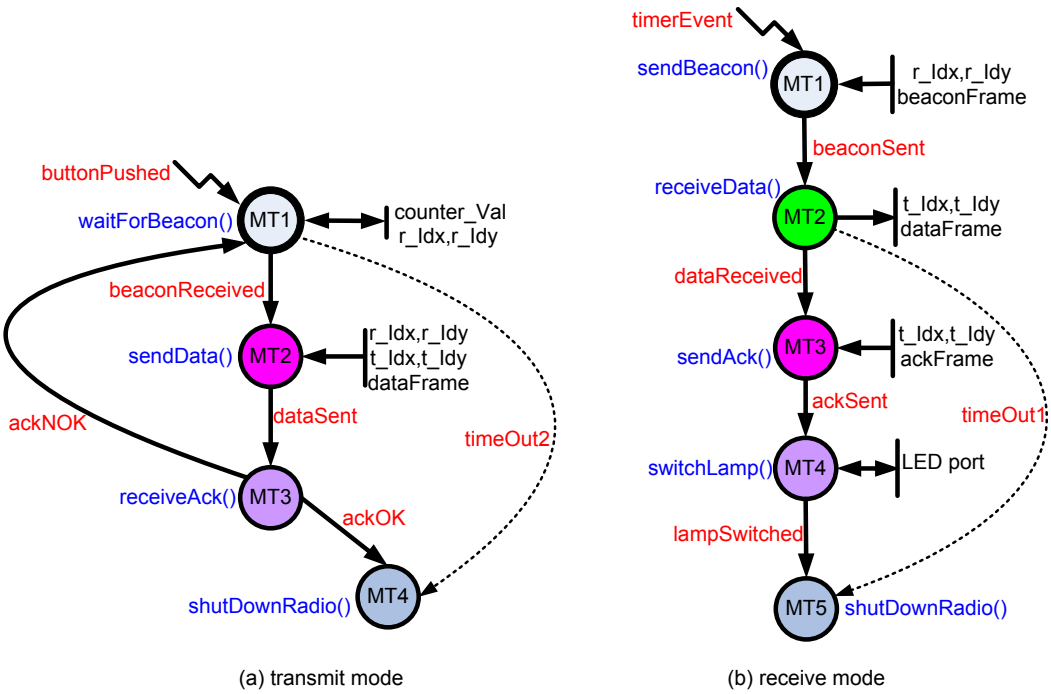


Fig. 1: System-level task flow graphs of a WSN node in transmit and receive modes

frame from any transmitter. The task is described in `receiveData()` function that starts a timer for possible time-out and receives and analyzes the data frame if it is destined for the receiver node or not. In case of valid data, it generates *dataReceived* event whereas if the timer is expired and no valid data is received, a *timeOut1* event is generated.

- Sending acknowledgment: Upon successful data reception, the receiver generates an acknowledgment for the transmitter node by calling `sendAck()` function that sends an acknowledgment frame to its RF transceiver and generates a *ackSent* event.

- Switching the lamp: The next task, in receiver TFG, is switching the lamp that is accomplished by calling the `switchLamp()` function. This function analyzes the previous state of the lamp by reading its corresponding port and inverses it to switch the lamp state. Then it generates a *lampSwitched* event.

- Shutting down the transceiver: Upon reception of *lampSwitched* or *timeOut1* event, the receiver node shuts down its RF transceiver.

b) *Notion of micro-task and generic architecture of proposed WSN node*

Each task of a TFG can be mapped onto a specialized hardware structure called a *micro-task*. This hardware can be seen as a small MCU datapath micro-architecture, driven by a control FSM that executes the micro-code corresponding to the task at hand. The micro-task can access some shared memories, and can be directly connected to some of the I/O peripheral ports.

Fig. 2(a) shows the template of a micro-task architecture with an 8-bit data-path; dotted lines represent control signals generated by the control FSM whereas solid lines represent the data-flow connections between the various datapath components.

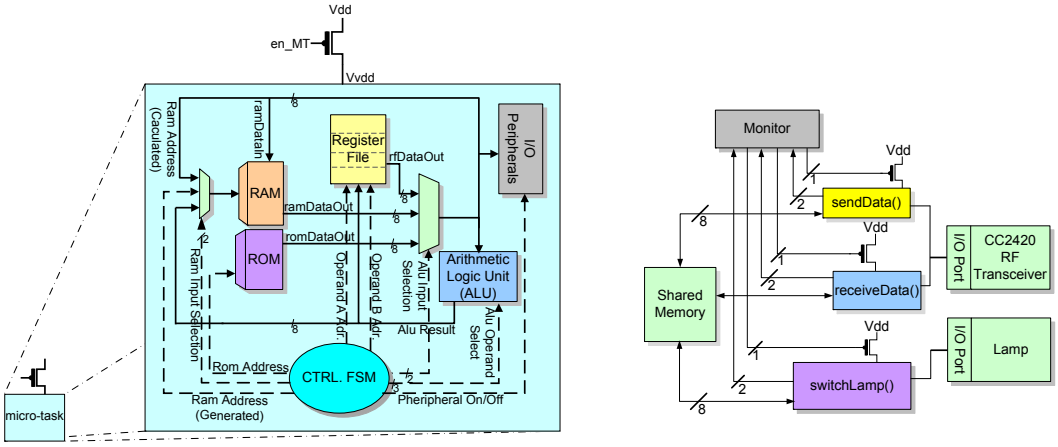
Fig. 2(b) presents the generic architecture of a node designed using micro-tasking. *Monitor* is a system-level controller that is responsible for the activation and deactivation of the individual micro-tasks and is automatically generated by our design tool (for details about its features, working and architecture see [20]).

IV Experimental results

This section starts by outlining the design-flow used to generate the micro-tasks from their corresponding C-descriptions and summarizes the power benefits of the proposed approach w.r.t. corresponding software implementation. Later on, it describes the experimental setup used for the extraction of output switching timings for a power-gated logic block.

a) *Automatic generation of micro-tasks*

The proposed design-flow for micro-task generation (Fig. 3) is based on *GeCoS* compiler infrastructure [21], a retargetable C compiler framework, whose instruction selection phase is retargeted to generate the assembly instructions for our simplified datapath model. This low-level program representation is then used to generate VHDL descriptions of (i) a custom datapath which implements the minimum required set of operations for the task at hand, and (ii) a micro-coded FSM that



(a) Architecture of a generic micro-task

(b) System-level application mapping of a generic WSN node

Fig. 2: Architecture of a common micro-task and system-level task-mapping of a generic WSN node.

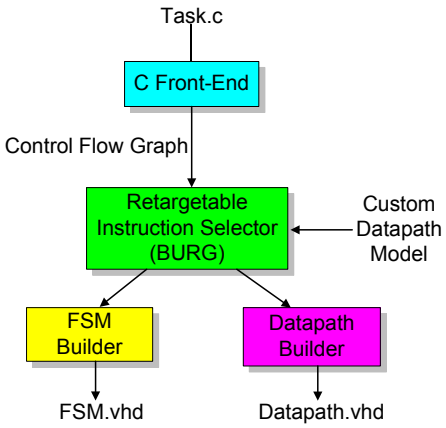


Fig. 3: Software design flow for micro-task generation

Table 1: Dynamic power consumption for various control tasks (@ 16 MHz).

Micro-Task				
Name	Power (μW)	Gain (x)	Area (μm^2)	Eq. Nand Gates (#)
sendData	28.2	312/34	6435	805
sendBeacon	28.3	310/33.9	6435	805
sendAck	28.2	312/34	6435	805
receiveData	28.3	310/33.9	6473	810
receiveBeacon	28	314/34.3	6469	809
receiveAck	27.9	315/34.5	6600	825
switchLamp	18.5	475/51	4208	526
shutDownRadio	20.3	433/47	4356	545

controls different entities of the datapath (for details of our design-flow, see [22]) .

b) Power benefits of micro-tasking

The micro-task VHDL designs have been synthesized for 130 nm CMOS technology using Synopsys *Design Compiler*. We used these synthesis results to extract gate-level static and dynamic power estimations assuming a 16 MHz operating frequency. For the sake of comparison, with a software implementation, we used as baseline the MSP430F21x2 (MCU core with memory and peripherals) dissipation of 8.8 mW normalized at 16 MHz (the data sheet indicates a dissipation of 550 μW at 1 MHz). We also synthesized an open-source MSP430-like MCU core to get the statistical power estimation. Early estimates show 0.96 mW at 16 MHz without accounting for memory and peripherals. We expect the actual power consumption of the MSP430 to lie between the two figures and compared the power consumption of micro-tasks to both of them.

The results are given in Table 1 where it can be observed that power benefits of one to two orders of magnitude can be gained over software implementation. The fourth column of Table 1 summarizes the surface areas consumed by the micro-tasks (FSM and datapath without the shared data RAM). The surface area of open-source MSP430-like MCU core is around 74,000 μm^2. Hence, it can be seen that our approach does not consume much area and several micro-tasks can be placed in the same area as is consumed by an MSP430-like MCU.

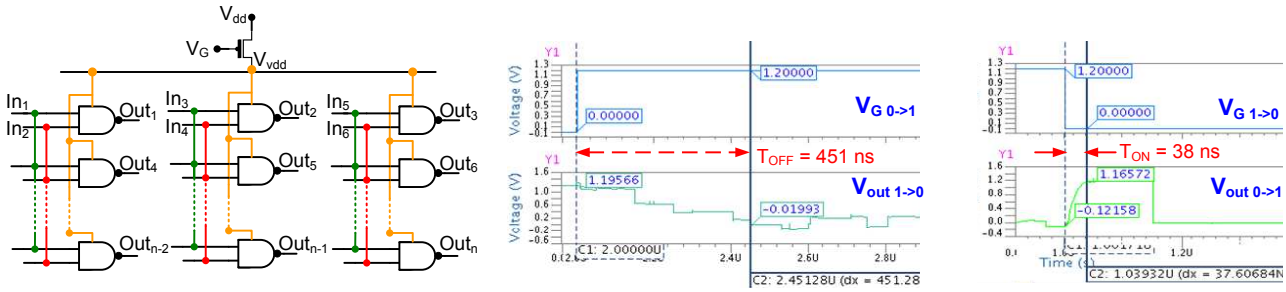
c) Switching delays for a power-gated block

To check the feasibility of applying power-gating to a micro-task, we used similar model of a power-gated block as was used in [23]. However, as no quantitative data for the switching delays specific to a CMOS technology was given, we re-run the experiments to extract the delay information.

For this purpose, We used Mentor Graphics *Eldo* to perform the transistor-level SPICE simulations using a 130 nm CMOS technology. We used parallel NAND gates to model the timing behavior of a power-gated block (Fig. 4(a)). Fig. 4 (b and c) also shows that for a block consisting of 3000 gates (a sleep transistor width of 2.04 μm), we have a turn-on delay of 38 ns and turn-off delay of 451 ns (between cut-off and active mode). This must be compared with MSP430's typical wake-up delay of 1 μs from the standby mode.

V Conclusion

In this paper, we have proposed a novel approach for ultra low-power implementation of control-oriented tasks running on a WSN node. Our ap-



(a) Parallel NAND gate model (b) Output turn-off delay T_{OFF} (c) Output turn-on delay T_{ON}

Fig. 4: Parallel NAND gates model used to perform the SPICE transistor level simulations and the output turn on and off delays for ($n = 3000$).

proach is based on *power-gated micro-tasks* that are implemented as specialized hardware blocks. We presented as a case study a WSN system implementing a lamp-switching application.

The synthesis results obtained for the micro-tasks of the case study show that, compared with a software implementation such as the MSP430 microcontroller, power reductions of one to two orders of magnitude are possible.

In future, we would also like to evaluate the feasibility of our approach on control-oriented reconfigurable structures, which would provide support for small grain power-gating techniques [24]. We also envisage to interface the proposed micro-task-based node with the WSN simulators such as WSim and WSNet for network-level validation.

References

[1] University of California, Berkeley, "Tech. project: Smart dust." [Online]. Available: http://robotics.eecs.berkeley.edu/~pister/SmartDust/

[2] Texas Instruments, "MSP430 User Guide," Tech. Report, 2009.

[3] Atmel Corporation, "ATmega 128L 8-bit AVR Low-Power MCU," Tech. Report, 2009.

[4] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. New York, NY, USA: ACM, 1999, pp. 263–270.

[5] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," in *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. New York, NY, USA: ACM, 2002, pp. 88–97.

[6] Y. Li, Z. Wang, and Y. Song, "Wireless Sensor Network Design for Wildfire Monitoring," *The Sixth World Congress on Intelligent Control and Automation, WCICA 2006.*, vol. 1, pp. 109–113, 2006.

[7] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh, "Monitoring Volcanic Eruptions with A Wireless Sensor Network," *Proceeedings of the Second European Workshop on Wireless Sensor Networks, 2005.*, pp. 108–120, Jan.-2 Feb. 2005.

[8] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks:A Survey," *Computer Networks*, vol. 38, no. 4, March 2002.

[9] D. Malan, T. Fulford-jones, M. Welsh, and S. Moulton, "CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care," in *In International Workshop on Wearable and Implantable Body Sensor Networks*, 2004.

[10] C. Herring and S. Kaplan, "Component-Based Software Systems for Smart Environments," *Personal Communications, IEEE [see also IEEE Wireless Communications]*, vol. 7, no. 5, pp. 60–61, Oct 2000.

[11] L. Nazhandali, M. Minuth, and T. Austin, "SenseBench: Toward an Accurate Evaluation of Sensor Network Processors," in *Proceedings of IISWC'05*, Oct. 2005.

[12] S. Mysore, B. Agrawal, F. Chong, and T. Sherwood, "Exploring the Processor and ISA Design for Wireless Sensor Network Applications," in *Proceedings of VLSI'08*, Jan. 2008.

[13] Crossbow Technology, "Mica2 motes," http://www.xbow.com/. [Online]. Available: http://www.xbow.com/Products/productdetails.aspx?sid=174

[14] R. Fonseca, P. Dutta, P. Levis, and I. Stoica, "Quanto: Tracking Energy in Networked Embedded Systems," in *OSDI'08*, 2008.

[15] S. Roundy, P. Wright, and J. Rabaey, *Energy Scavenging for Wireless Sensor Networks: with Special Focus on Vibrations*. Springer, 2004.

[16] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, D. Culler, *TinyOS: An Operating System for Sensor Networks* . Book Chapter in Ambient Intelligence by Springer, 2005.

[17] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors," *in LCN'04*, Nov. 2004.

[18] E.-Y. Lin, J. Rabaey, and A. Wolisz, "Power-efficient rendez-vous schemes for dense wireless sensor networks," in *Communications, 2004 IEEE International Conference on*, vol. 7, June 2004, pp. 3769–3776 Vol.7.

[19] Texas Instruments, "Single-Chip 2.4 GHz IEEE 802.15.4 Compliant and ZigBee RF Transceiver." [Online]. Available: http://focus.ti.com/docs/prod/folders/print/cc2420.html

[20] M. A. Pasha, S. Derrien, and O. Sentieys, "System Level Synthesis for Ultra Low-Power Wireless Sensor Nodes," in *DSD'10: 13th Euromicro Conference on Digital System Design*, Lille, France, 2010.

[21] L.L'Hours, "Generating Efficient Custom FPGA Soft-Cores for Control-Dominated Applications," in *Proceedings of ASAP '05*, Washington, DC, USA, 2005.

[22] M. A. Pasha, S. Derrien, and O. Sentieys, "A Complete Design-Flow for the Generation of Ultra Low-Power WSN Node Architectures Based on Micro-Tasking," in *proceedings of Design Automation Conference, 2010, DAC*, 2010.

[23] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural Techniques for Power Gating of Execution Units," *International Symposium on Low Power Electronics and Design, 2004. ISLPED '04. Proceedings of the 2004*, pp. 32–37, 2004.

[24] A. Rahman, S. Das, T. Tuan, and S. Trimberger, "Determination of Power Gating Granularity for FPGA Fabric," *Conference 2006, IEEE Custom Integrated Circuits*, pp. 9–12, Sept. 2006.