Copyright © 2008 American Scientific Publishers
All rights reserved
Printed in the United States of America

Journal of
Low Power Electronics
Vol. 4, 1–10, 2008

Survey of Hardware Systems for Wireless Sensor Networks

Mark Hempstead, Michael J. Lyons, David Brooks, and Gu-Yeon Wei*

School of Engineering and Applied Science, Harvard University, Cambridge, MA 02138, USA

(Received: xx Xxxx xxxx; Revised/Accepted: xx Xxxx xxxx)

Wireless sensor networks have been gaining interest as a platform that changes how we interact with the physical world. Applications in medicine, military, inventory management, structural and environmental monitoring, and the like can benefit from low-power wireless nodes that communicate data collected via a variety of sensors. Current deployments of wireless sensor networks (WSN) rely on off-the-shelf commodity-based microcontrollers, but the unoptimized energy consumption of these systems can limit the effective lifetimes. Ideally, researchers would like to deeply embed wireless sensor network nodes in the physical world, relying on energy scavenged from the ambient environment. This paper provides a survey of ultra low power processors specifically designed for WSN applications that have begun to emerge from research labs, which require detailed understanding of tradeoffs between application space, architecture, and circuit techniques to implement these low-power systems.

Keywords: Wireless Sensor Networks, Low-Power Design.

1. INTRODUCTION

Wireless sensor networks are poised to transform how we interact with the physical world. Today, researchers and practitioners utilize low-power nodes composed of wireless radios, sensors, and computing elements for a variety of applications in medicine, military, biology, manufacturing, etc. Most deployments of wireless sensor networks (WSN) use off-the-shelf commodity based microcontrollers, though the energy consumption of these systems can limit the effective lifetimes of the WSN nodes. Ideally, researchers would like to deeply embed wireless sensor network nodes into the physical world and power the devices solely from energy scavenged from the ambient environment. To approach such low-energy requirements, ultra-low-power processors specifically designed for WSN applications have begun to emerge from various research labs. This survey paper presents an overview of the application space, architectures, and circuit techniques currently used to implement these low-power systems.

We have found that developing ultra low-power systems requires a holistic approach to design. Decisions of system architecture should be made in concert with an exploration of the application space and low-power circuit techniques. WSNs measure and react to the phenomena being observed. Consequently, the performance requirements of WSN systems can span a wide range—anywhere from taking samples every few minutes (temperature sensing) to thousands of samples a second (seismic sensing, audio or video). Transmitting every sensor sample on the radio would consume all of the wireless bandwidth available to the network and quickly drain the available stored energy. Therefore, many systems employ data filtering on the node so that only interesting sensor readings are communicated over the radio. This tradeoff between communication and computation places a higher burden on energy-efficient computation. In this work, we explore the different application classes and describe system architectures that can take advantage of the event-driven and regular nature of WSN applications.

New circuit techniques are aiding the quest to reduce power consumption. Some of the systems we survey remove clocking overhead by using asynchronous circuits. Others trade performance for lower power consumption by using supply voltages less than the threshold voltage. As transistor dimensions continue to shrink, leakage current increases. Some systems respond to this trend by adding architecture support for circuits that turn off the power supply of unused blocks to reduce leakage current.

In Section 2 we survey the application space of wireless sensor networks and describe how architecture and circuit design decisions can be informed by the application space.

*Author to whom correspondence should be addressed.
Email: guyeon@eecs.harvard.edu

We introduce a few of the more common low-power circuit techniques in Section 3. In Section 4 we present a range of hardware implementations and look closely at the differences between these systems.

2. APPLICATIONS FOR WIRELESS SENSOR NETWORKS

Informed microarchitecture decisions require an understanding of the wide range of sensor network applications and their requirements. In this section, we describe examples of sensor networks being deployed to solve real problems. We classify several application areas based on the amount of computation required, lifetime of nodes, and the observed phenomena.

In this work we define wireless sensor networks as: "networks of autonomous, energy-constrained nodes with in-network sensing, communication, and computation." We do not survey low-power single-node systems such as biomedical implants because these systems do not form networks and not do require nodes to relay data or perform in-network computation. Wired networks of sensors such as systems for building automation, or large-scale city-wide sensing are not considered either, because they do not face the same energy constraints.

2.1. Application Classes

Deployed sensor networks measure a wide range of phenomena including atmospheric temperature, heart rate, volcanic eruptions, and even the sound of a sniper rifle. The performance target (cycles of computation per second) for a WSN node is set by the sampling rate for the measured phenomena and the amount of on-node data filtering required. Table I lists the range of sampling rates for different physical phenomena. Environmental measurements such as temperature and pressure have time constants on the order of minutes. Consequently, nodes deployed to measure low-frequency phenomena will be idle most of the time. In contrast, nodes that measure higher-frequency phenomena will require higher performance processors.

WSNs have been deployed by practitioners in a variety of industries from science and medicine to the military and business. Sensor nodes are sometimes deployed in hard to reach places making it difficult and expensive to change batteries regularly. In this work, we classify node lifetime based on the availability of wired power sources or battery replacements. In some domains, such as military and security applications, nodes embedded deeply in the structure of a building would be difficult to manually maintain and would consequently require node lifetimes of several years on one battery. In medical domains (not including bio-implants) a patient or health care professional would be able to replace batteries daily. Table II lists a few example application domains with an estimate of their deployment lifetimes and computation requirements.

Table I. Sensor sampling rates of different phenomena.

Phenomena	Sample rate (in Hz)
Very low frequency	
Atmospheric temperature	0.017–1
Barometric pressure	0.017–1
Low frequency	
Heart rate	0.8–3.2
Volcanic infrasound	20–80
Natural seismic vibration	0.2–100
Mid frequency (100 Hz–1000 Hz)	
Earthquake vibrations	100–160 Hz
ECG (heart electrical activity)	100–250
High frequency (>1 kHz)	
Breathing sounds	100–5 k
Industrial vibrations	40 k
Audio (human hearing range)	15–44 k
Audio (muzzle shock-wave)	1 M
Video (digital television)	10 M

To provide concrete examples of wireless sensor applications we summarize a few recent deployments:

- *Great Duck Island – UC Berkley*—This project deployed a 100 + node sensor network on an island off the coast of Maine to study a rare sea bird. Sensor nodes were placed in nesting burrows and above ground weather stations.[21]
- *Countersniper Application – Vanderbilt University*—This application detects a gunshot and triangulates the position of the sniper from coordinated observations of independent sensor nodes. An FPGA was attached to each sensor node that implemented the signal processing algorithm and time-of-arrival calculation.[6]
- *Industrial WSN – Intel Corp and Arch Rock*—These organizations deployed a sensor network to measure the vibrations in a semiconductor fab and a ship in the North Sea with the goal to predict equipment failures.[13]
- *Volcano Monitoring – Harvard University*—This group has launched three separate deployments of wireless sensor networks on active volcanoes in Ecuador. The application includes an event detection algorithm to selectively transmit data back to a base station during eruptions.[25, 26]

All of these deployments provided detailed observations to domain scientists that were not otherwise possible. A few of the applications (Great Duck Island) did not filter any of the data on the nodes and transmitted all of the data back to the base station. The Volcano and Countersniper applications required filtering on the sensor node because the radio bandwidth was not high enough to transmit all of the sensor data (even without considering the energy costs).

2.2. Prevalence of Middleware

So far we have provided a simplified presentation of WSN deployments as mere filters and wireless relays of sensor data. In fact, deployments of WSN include rich layers

Table II. Example WSN application domains.

Application domain	Desired lifetimes	Computation requirements (Sample rates)	Example
Scientific applications			
Habitat/weather monitoring	Months/decades	very low	Great Duck Island[21]
Volcanic eruption detection	Months/decades	mid	Volcano WSN[26]
Military and security applications			
Building/border intrusion detection	Years/decades	low	
Structural and earthquake monitoring	Years/decade	low/mid	
Active battlefield sensing	Months	mid/high	Sniper detection/localization[6]
Medical applications			
Long-term health monitoring (pulse)	Days	low	
Untethered medical instruments (ECG)	Days	med	EKG mote[7]
Business applications			
Supply chain management	Months	low	
Expired/damaged goods tracking	Months	low	
Factory/fab monitoring	Months/years	med/high	Industrial WSN[13]

of network services that are necessary to form a reliable network. Systems researchers have developed open-source middleware algorithms that are used by many deployments. Providing acceleration for typical middleware computations has the potential to reduce the overall energy consumption of the system.

We now examine the Volcano monitoring application[25,26] to illustrate the use of middleware. The authors deployed a WSN on an active volcano on three occasions. Measurements of infrasound (audio signals less than 40 Hz) and seismic vibrations were recorded. A custom event detection algorithm consisting of a weighted moving average filter is the most frequent computation run on the nodes. It is used to detect a possible volcanic eruption and trigger a network-wide download of interesting data. A bulk data transfer algorithm developed by the authors (Fetch) is used to copy data back to the researchers several kilometers away. In this work the authors use FTSP, an open-source middleware package, to maintain time synchronization between nodes.[14] Because the network was spread over a large geographical area, nodes were required to relay messages for other nodes in the network. An open-source routing package called MintRoute was run on the nodes that enabled multi-hop routing of messages.[27] Many WSN deployments for scientific monitoring have similar needs in terms of sensing, bulk data transfer, networking, and time synchronization. All of the tasks were run on commodity, general-purpose hardware platforms without any hardware acceleration for the more frequent tasks.

3. CIRCUIT DESIGN TECHNIQUES

Several layers below the application, the circuits layer contains many opportunities to reduce power consumption. WSNs with low computation requirements are idle most of the time, so leakage current makes up the largest fraction of overall energy consumption. Moreover, as process technology dimensions scale, leakage current increases. In this section we describe circuit techniques which have been used to reduce energy consumption in WSN nodes.

3.1. Voltage and Frequency Scaling and Subthreshold Design

Sensor network applications are untethered from wired power sources, therefore conserving system energy consumption is the primary design concern. Energy can be expressed as the sum of active switching energy plus leakage current energy.[1]

$$E_{\text{total}} = V_{dd}(\alpha C_{sw} V_{dd} + I_{\text{leak}} \Delta t_{op}) \quad (1)$$

Where α is the switching activity for one second and Δt_{op} is the amount of time required to complete an operation. For low duty cycle applications, leakage energy will dominate because α will be small. One of the most effective ways to save energy is to scale the power supply voltage. As V_{dd} scales down, active energy decreases quadratically. As pointed out in the literature, there is an energy optimal point for V_{dd} that for most circuits is less than the threshold voltage (V_{th}). The main concern when lowering V_{dd} is that traditional SRAMs do not operate reliably below V_{th}. A few recent subthreshold SRAM prototypes have shown potential, but the designs are not yet ready for full-scale production.[2]

The relationship between supply voltage and delay is a major concern when operating circuits in subthreshold. The drive current of a transistor operating in subthreshold is exponentially related to the supply voltage. Consequently, delay of a CMOS gate in subthreshold can be represented by the simple expression:

$$\Delta t_{op} \propto e^{-kV_{dd}} \quad (2)$$

where k is a constant that depends on technology and temperature. Circuit delay will vary exponentially with any

variation in manufacturing or temperature, so selecting a fixed clock frequency where the chip will operate reliably can be difficult. As described in Section 2, computation requirements of WSN applications vary by several orders of magnitude. Synchronous subthreshold systems that run at low clock frequencies might not be able to meet the computation requirements of mid and high frequency sample rates. Subthreshold operation provides significant energy savings at the cost of performance and are suitable for WSN workloads that do not require significant computational resources.

3.2. Asynchronous Circuit Design

Asynchronous circuits do not rely on globally periodic timing signals (clocks), but they need to operate under additional design constraints. All asynchronous circuits must be glitch-free (under a certain timing model), which often results in additional logic. Asynchronous circuits also require explicit handshake signals between circuit blocks. While requiring additional overhead, handshaking does make it easy to connect circuit blocks together because timing and data requirements are communicated explicitly. As operating considerations change (temperature, V_{dd}, variations in manufacturing), providing explicit timing information allows the system to respond to changes in delay of the critical path. Because a clock signal is not driven while the processor is idle, asynchronous circuits provide a native way of reducing energy from circuit switching for low duty cycle applications.

One common class of asynchronous circuits are quasi-delay-insensitive (QDI) circuits that are insensitive to delays in gates and wires but require *isochronic* forks. An isochronic fork is a forked wire where all branches have exactly the same delay.[3] QDI circuits are used widely in the asynchronous design community to build full microprocessors. Synthesis techniques are available for QDI circuits.

3.3. VDD-Gating

Gating the power supply has been used as a circuit technique to reduce the subthreshold leakage current consumed in a system.[18] In this work, the authors show how VDD-gating can reduce leakage current by roughly 98% with an additional area cost of 3% and little performance penalty. The downside of this technique is the loss of circuit state once the block has been cut-off from the power supply. Therefore, microarchitecture support is required to efficiency manage gating transistors based on application needs.

4. HARDWARE FOR WIRELESS SENSOR NETWORKS

Recently proposed systems for wireless sensor networks utilize different circuit techniques and architecture approaches. In this section we describe several categories of hardware designs for WSN and reference prototypes for each category.

First we categorized systems based on the circuit techniques employed to reduce total energy consumption.

- *Subthreshold operation*—By using a power supply less than the threshold voltage, systems such as the Subliminal processor from the University of Michigan are able to trade off performance for reduced active power consumption.
- *Asynchronous Circuits*—Processors such as SNAP from Cornell University eliminate clock power by relying on asynchronous circuits.
- *Power Supply Gating*—To address increasing leakage current, systems from Harvard University and the University of California employ transistors that switch the power supplies of unused blocks.

Each of these systems take differing approaches to architecture support for applications.

- *General Purpose Computation*—Off-the-shelf and custom designed systems employ load-store or accumulator based processors as the core processing engine of the system.
- *Event Driven*—WSN applications respond to environmental events and are event-driven by nature. Consequently, systems from Cornell, Harvard, and Michigan support native handling of events in hardware.
- *Application Acceleration*—systems from Harvard and University of California provide hardware acceleration for common tasks to reduce active energy consumption and increase system performance.

Subsequent sections describe several example systems in more detail. In the final section we present a summary of discussed systems.

4.1. General Purpose Commodity Based Systems

Existing sensor network platforms are typically designed around off-the-shelf microcontrollers, such as the TI MSP430 or Atmel ATMega 128L. These processors are designed for low-power operation across a range of embedded application settings, but are fundamentally based on monolithic, general-purpose computing engines that are not well-suited to the event-driven nature of sensor network workloads. While such processors do support low-power idle states (consuming less than 5 μA in the case of the MSP430), these involve disabling the entire processor and waking it back up on the next interrupt, thus, limiting the use of these modes in interrupt-dominated, event-driven applications such as wireless sensor networking. The monolithic nature of such architectures also preclude finer-grained duty cycling of individual processor components.

The Mica2 mote—based on the Atmel ATMega 128L microcontroller—has been widely used by the systems research community. The Mica2 includes the 7.3 MHz

ATmega 128L processor with 128 KB of code memory, and 4 KB of data memory. Measurements of the Mica2 show that it consumes an average of 8 mA of current when active and 100 μA when in low power mode.[20] This is roughly 3.2 nJ per instruction.

Another popular microcontroller used for WSN platforms is the Texas Instruments MSP430 microcontroller. It has a 16-bit databus and has 10 KB of RAM and 48 KB of Flash. It consumes 2 mA at 8 MHz and 3.0 V and consumes a few microamps of current in low power sleep mode. This corresponds to an energy consumption of roughly 750 pJ per instruction.

Several operating systems have been written for wireless sensor network applications that provide event handling on top of commodity based microcontrollers. TinyOS[11] is widely used in the sensor network community and is available for both Atmel ATmega 128L and TI MSP430 based motes.

4.2. Smart Dust—Early Event Driven

One of the early microcontrollers for sensor network systems was designed for the *Smart Dust* platform at the University of California Berkeley.[23,24] Conceptually, a smart dust *mote* consists of several MEMS sensors, a solar cell, optical communication, and a microcontroller. The microcontroller is a load-store RISC processor with a Harvard architecture (separate instruction and data memories). Figure 1 presents a functional block diagram of the system. The design incorporates a continuously running low-speed oscillator that drives five on-chip timers for sensor sampling, radio transmission and reception, and data path operation. When the timers fire, a faster oscillator is powered on to drive the datapath and ADC. By including independent subsystems (each driven by its own timer) the system is able to clock-gate inactive blocks. However the system is not natively event driven and uses the timers to poll the data sources periodically. This system was designed in 0.25 μm technology and consequently did not need to address leakage current with architecture. At 1.0 V and 500 kHz the system consumes 12 pJ per instruction.

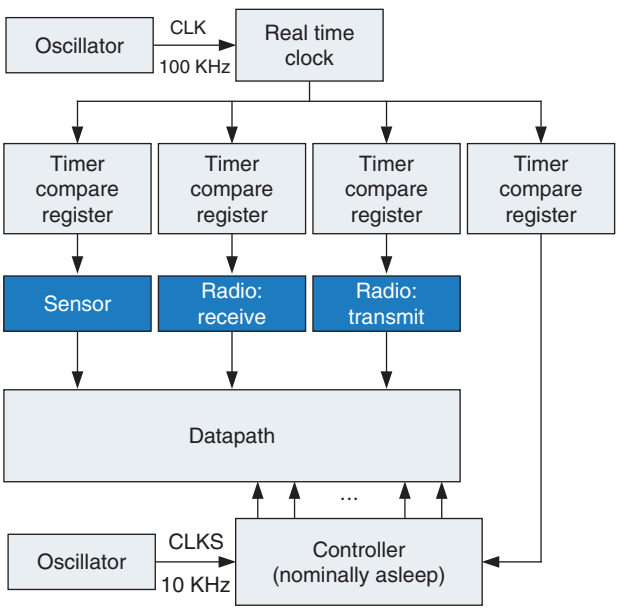

Fig. 1. Smart dust microarchitecture. Reprinted with permission from [23], B. A. Warneke and K. S. J. Pister, An ultra-low energy microcontroller for smart dust wireless sensor networks. *ISSCC*, January **(2004)**. © 2004.

4.3. Subthreshold Systems

As presented in Section 2, the throughput requirements of nodes for WSNs depend on the observed phenomena. For phenomena requiring low frequency sampling (less than 100 Hz), off-the-shelf systems easily keep up with the required real-time workloads resulting in long idle times. Several researchers have designed circuits that operate below the threshold voltage which trades performance for power consumption. Several system building blocks that run in subthreshold have been presented, including an FFT.[22]

Researchers at Michigan have designed one of the first complete processors for WSNs designed to run in subthreshold, the *Subliminal Processor*.[8,16,17,28] Both versions of the system center around a traditional general purpose Harvard based architecture. The authors swept several different architecture parameters such as register size, bus width, CISC/RISC, and number of pipeline stages. The authors evaluated code size, cycles-per-instruction (CPI), and energy.[17]

The architecture of version two of the Subliminal processor is presented in Figure 2. The system consists of an

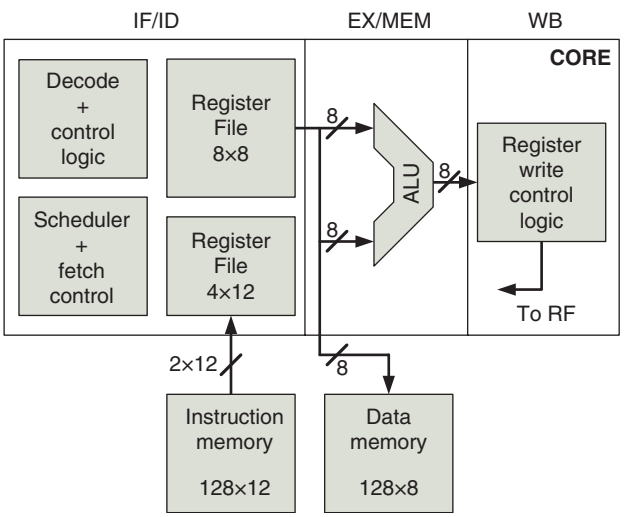

Fig. 2. Block diagram of the subliminal processor (University of Michigan). Reprinted with permission from [8], S. Hanson et al., Performance and variability optimization strategies in a Sub-200 mV, 3.5 pJ/inst, 11 nW subthreshold processor. *IEEE Symposium on VLSI Circuits (VLSI-Symp)*, June **(2007)**. © 2007.

8-bit datapath with separate instruction and data memories, a decode and fetch stage, and logic for event scheduling.

The authors implemented two versions of the Subliminal processor in 130 nm CMOS. Because standard bitline-based SRAMs do not function reliably in subthreshold, the prototypes used mux based memories. In both versions the authors measured significant delay variability when operating in subthreshold. The normalized maximum operating frequency of the first prototype varied from 0.6 to 1.8 at 260 mV across a distribution of 26 different chips.[28] The average energy consumption of the subliminal processor is 2.6 pJ/inst and 3.5 pJ/inst.

4.4. Asynchronous—SNAP

The SNAP processor from Cornell is an asynchronous 16-bit RISC based processor designed with sensor network workloads in mind.[4, 5, 12] SNAP is an event-driven processor by design and includes two accelerators that generate WSN specific events. SNAP uses *quasi delay-insensitive* (QDI) circuits (defined in Section 3.2). By designing SNAP completely with asynchronous circuits, it runs across a wide range of supply voltage from 1.8 V to 0.6 V.

Figure 3 presents a simplified block diagram of SNAP/LE, the first implementation of the SNAP architecture. SNAP does not perform computation continuously but instead responds to discrete *events*. The system includes an *event queue* containing tokens that designate a particular event. When the token reaches the head of the queue, SNAP fetches the corresponding *event handler* from instruction memory (IMEM). The decode block selects the appropriate execution unit for the opcode and gets operands from data memory (DMEM) and the register file. The instructions are processed in sequence until the "done" instruction is reached indicating the end of the *event handler*. If the event queue is empty, SNAP stalls after reaching the "done" instruction and waits for a new event token to enter the event queue.

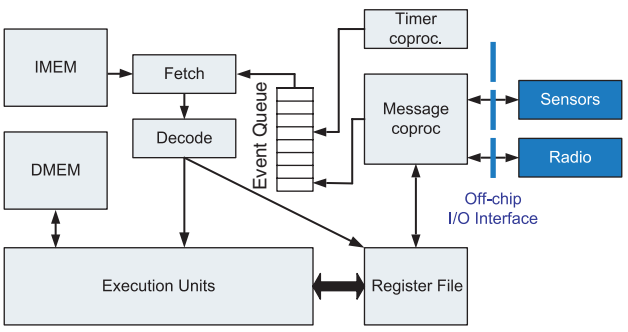

Fig. 3. Simplified block diagram of the SNAP processor for WSN. System includes separate instruction and data memories, a timer coprocessor, and a message processor which provides a FIFO interface to the off-chip radio and sensors. Reprinted with permission from [4], V. Ekanayake et al., An ultra low-power processor for sensor networks. *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems*, Boston, MA, October **(2004)**. © 2004.

SNAP includes two components that generate events, the timer coprocessor and message coprocessor. The timer coprocessor consists of three self-decrementing timer registers; it posts a timer event token when the registers reach zero. The message coprocessor serves as the interface between SNAP and off-chip devices by providing I/Os for sensors and the radio. SNAP communicates with the message coprocessor through a single register. Each time a byte arrives on the radio, the message coprocessor posts an event to the event queue. The message coprocessor does not perform computation on the radio data itself but serves as a relay to the datapath.

Many commodity microcontrollers include several low-power modes that put the processor to *sleep*; the amount of time it takes to go to sleep and wake up is very important. Asynchronous circuits by nature do not require a clock. The authors argue SNAP can fall asleep in nanoseconds because all circuit switching stops once the event queue is empty. This technique is effective for 180 nm technologies where leakage current is small. However, as leakage current increases, systems based on SNAP will need to incorporate low leakage techniques.

BitSNAP, the latest version of the SNAP architecture, replaces the parallel datapath in SNAP/LE with a bit-serial datapath that takes more time to compute operations but consumes 70% less energy than SNAP/LE.[5] BitSNAP also includes *dynamic significance compression* to selectively adjust the number of iterations run on the serial datapath depending on number of significant bits in the operands. Serial datapaths are interesting from a power perspective because they have fewer transistors than parallel datapaths and result in a smaller effective width on the leakage current path.

4.5. Charm—Network Stack Acceleration

The Charm Protocol processor out of the Berkeley Wireless Research Center realizes a large portion of a custom radio stack in hardware.[19] The custom radio stack roughly corresponds to the OSI reference model. Charm implements the application, network, data link, and digital baseband portion of the OSI radio stack as well as

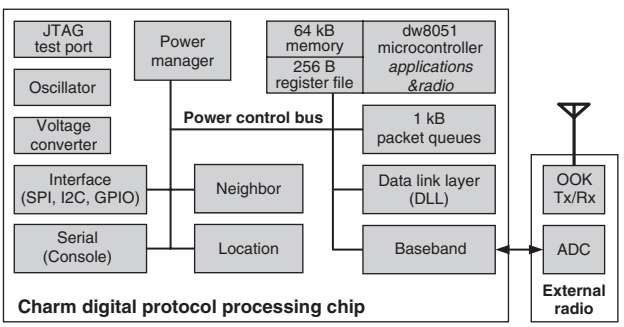

Fig. 4. The charm protocol processor microarchitecture. Reprinted with permission from [19], M. Sheets et al., A power-managed protocol processor for wireless sensor networks. *VLSI*, June **(2006)**. © 2006.

hardware for a localization subsystem. Figure 4 presents the Charm microarchitecture, which includes blocks for each of the major subsystems.

One unique feature of Charm is its power management infrastructure. The authors recognized that leakage current is a significant portion of total power consumption. Charm contains a separate power domain for each of the protocol stack subsystems. Instead of completely gating the power domain Charm includes switches to select between two different supply voltages VDD_{hi} and VDD_{low}. VDD_{low} is set to the lowest voltage that maintains state in the logic. The state of the power switches is controlled by a power manager that accepts requests from each of the power domains on a central bus.

Charm consumes 250.1 μW of leakage current when all of the power domains are connected to VDD_{hi} (1.0 V). When the power domains are connected to VDD_{lo} (0.3 V) leakage current is 10 μW plus 43.6 μW for the always-on circuitry. This is a savings of 5x for the power domains which can be switched. If the system samples the radio at a period of 100 ms then the average power consumption of Charm is 132 μW. The metric *energy per instruction* is not applicable to Charm because the primary unit of computation is radio packets. Charm provides energy-efficient acceleration of the radio stack, but relies on general-purpose processing for application specific tasks such as data filtering and compression. However, WSN radio stack standards change often, and Charm's custom radio stack (synthesized in hardware) cannot be easily changed to adapt to evolving standards.

4.6. Harvard Event-Driven Architecture

Our group at Harvard has taken an application-driven approach by including hardware acceleration to optimize typical operations in WSN applications and application-controlled VDD-gating to address leakage current.[9] The Harvard event-driven system for WSN uses three techniques to reduce energy consumption.

- *Lightweight event handling in hardware*—Initial responsibility for handling incoming interrupts is given to a specialized Event Processor, removing the software overhead that would be required to provide event handling on a general-purpose processor.
- *Hardware acceleration for typical WSN tasks*—Modular hardware accelerators are included to complete regular application tasks such as data filtering and message routing.
- *Application-controlled fine-grained VDD-gating*—Addressing leakage current with architecture support for VDD-gating enables accelerator blocks to be powered off when unused.

An overview of our proposed system architecture is provided before discussing the individual components in more detail. The system architecture is illustrated in Figure 5.

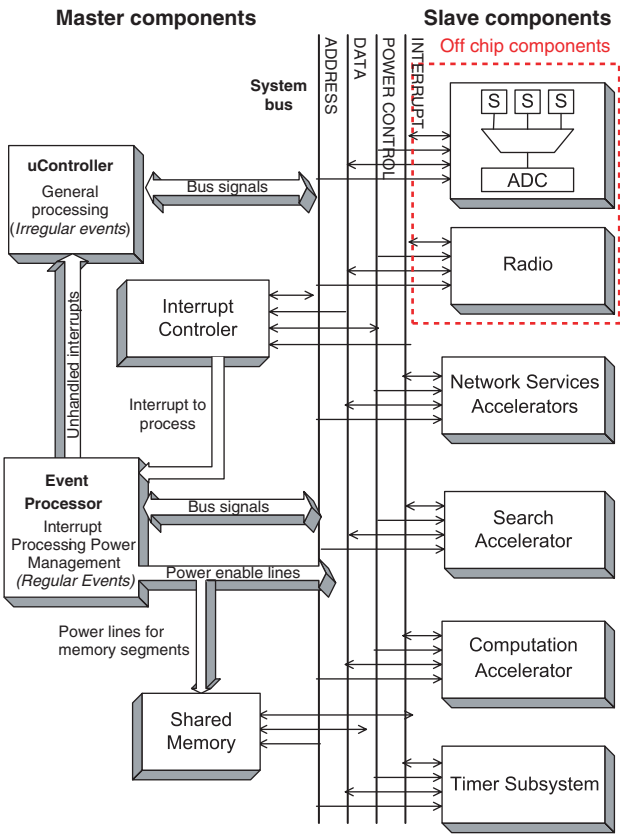

Fig. 5. Block diagram of the Harvard event-driven system for WSN. Reprinted with permission from [9], M. Hempstead et al., An ultra low power system architecture for sensor network applications. *The 32nd Annual International Symposium on Computer Architecture (ISCA)*, June **(2005)**. © 2005.

There are two distinct divisions within the system in terms of the ability of components to control the system bus. We refer to the components that have full control of the system address lines as *master* components and the remaining blocks that do not facilitate transfers on the databus as *accelerator* components. The system bus has three segments—an interrupt bus, a data bus, and power control lines. The accelerators respond to read or write requests from the master side of the data bus, thus allowing the masters to read information content and control execution of the accelerators. The two master devices consist of the event processor, comprising a small state machine, and an infrequently-used general-purpose microcontroller.

A key benefit of the modular design of the architecture is its ability to employ fine-grained power management of individual components (both masters and accelerators). Selectively turning off components and using VDD-gating enables the system to minimize leakage power. For example, the general-purpose microcontroller core could be relatively complex and power-hungry when active, but can be VDD-gated most of the time when idle. The event processor handles all interrupts, distributes tasks to accelerator devices, and wakes up the microcontroller only when necessary (rarely).

We now describe some of the more interesting system components in greater detail.

- *System Bus.* The system bus comprises the data bus, the interrupt bus, and power control lines. The data bus has address, data, and control signals indicating read and write operations.
- *Microcontroller.* The microcontroller block is a simple general-purpose CPU that implements the z80 instruction set. Rather than designing this core from scratch, we modified an existing core to serve as the general-purpose compute element. The microcontroller is used to handle irregular events.
- *Event Processor.* The event processor (EP) is a programmable state-machine that can process basic data transfers and power management instructions. This component orchestrates all of the other components' tasks. It performs no computation, but executes Interrupt Service Routines (ISR) that transfer data between blocks and initializes accelerator components (or the general-purpose processor) to perform computation and service interrupts. The interrupt service routines also contain power-management instructions that allow the EP to explicitly perform fine-grained power management of all other system components.
- *Interrupt Controller.* The interrupt controller selects one of the interrupt lines for service. It provides simple priority selection and the ability to mask interrupts.
- *Timer Subsystem.* The periodic nature of sensor network applications requires multiple, configurable timers. The timer subsystem supports four 16-bit counters that can be used to wake the event processor to handle different events.

Full system simulations based on a SystemC model of the architecture reveal a 10x performance savings over the same application written for the commodity based Mica2 mote. This optimization for regular tasks allows the system to either run 10x slower clock or run faster leaving more time to VDD-gate the hardware accelerators.

The system has been implemented in 130 nm CMOS and contains 4 KB of memory and runs at a power supply's from 0.55 V to 1.2 V. Measurements of the system show a leakage power savings of 100× with VDD-gating and a total active energy of 680 pJ per task (a task is equivalent to 1500 Atmel ATmega 128L instructions). The system runs up to 12.5 MHz at 0.55 V and higher frequency operation is possible at higher voltages.

4.7. Summary of Systems

In this section we surveyed several systems designed for wireless sensor networks. These systems differ in circuit techniques, architecture approaches, and support for applications. Figure 6 presents a summary of the discussed systems.

Unfortunately, standard benchmark suites do not exist for the WSN space though a few research groups have proposed some ideas.[10, 15] Without running the same application on each system, it is not possible to judge the programmability, energy efficiency, and performance of the different systems fairly. The efficacy of the energy per instruction metric to compare different systems has been questioned before but in this case could actually lead to completely misleading conclusions. The notion of an *instruction* is lost on both the Charm processor and the

System	Arch style	Data path width	Event driven (y/n)	Circuit techniques	Accelerators	Memory (KB)	Process	Voltage (V)	Throughput (MIPS)	Energy (pJ/ins)
Atmel ATmega128L	GP Off-the-shelf	8	N	N	N	132 KB	350 nm	3.0 V	7.3 MHz	3200
TI MSP430	GP Off-the-shelf	16	N	N	N	10 KB	NA	3.0	8 MHz	750
SNAP/LE	GP RISC	16	Y	Asyncronous	Timer, message interface	8 KB	180 nm	1.8 0.6	200 23	218 24
BitSNAP	GP RISC. Bit-serial datapath	16	Y	Asyncronous	Timer, message interface	8 KB	180 nm	1.8 0.6	54 6	152 17
Smart Dust	GP RISC	8	N	Syncronous-two clocks	None	3.125 KB	250 nm	1.0	0.5 (500 kHz)	12
Charm	Protocol processor	NA	N	Two power domains	Custom radio stack	68 KB	130 nm	1.0 V (high) 0.3–1.0 V (low)	8 MHz	150 μW 53.6 μW leakage
Michigan 1	GP	8	Y	Subthreshold	None	0.25 KB	130 nm	0.360	833 kHz	2.6
Michigan 2	GP	8	Y	Subthreshold	None	0.3125	130 nm	0.350	354 kHz	3.52
Harvard	Event driven accelerator	8	Y	VDD-gating	Timer, filter, message proc	4 KB	130 nm	0.55–1.2	12.5 MHz	680 pJ/task

Fig. 6. Summary of example hardware systems for wireless sensor networks.

Harvard Event-Driven system because most of the processing is handled by custom hardware accelerators. Even among the general-purpose architectures, the choice is not clear due to different instruction set architectures (ISAs), process technologies, and clock frequencies. One thing is clear: an intelligent combination of circuit techniques, application support, and architecture is required to build ultra low power systems.

5. CONCLUSION

Designing hardware for wireless sensor networks requires a holistic approach looking at all areas of the design space. Proposed designs combine low power circuit techniques and hardware support for typical WSN tasks. Systems researchers and domain practitioners are developing rich middleware and expanding the uses of WSNs, expecting more performance for less power out of the hardware platforms. Because the requirements for computation vary by several orders of magnitude, one-size-fits all hardware platforms will be difficult to develop. If designers choose to leverage technology scaling to provide more performance for less active energy consumption then an increase in leakage current will need to be addressed. With more research effort, we envision a future of WSNs made up of ultra low power nodes that provide high power computation and can be deployed for decades.

References

1. L. P. Alarcon, T.-T. Liu, M. D. Pierson, and J. M. Rabaey, Exploring very low-energy logic: A case study. *J. of Low Power Electronics* **(2007)**.
2. B. H. Calhoun and A. Chandrakasan, A 256 kb sub-threshold SRAM in 65 nm CMOS. *IEEE International Solid-State Circuits Conference (ISSCC)*, February **(2006)**.
3. A. Davis and S. M. Nowick, An Introduction to Asynchronous Circuit Design. Technical Report UUCS-97-013, University of Utah Technical Report, Department of Computer Science, September **(1997)**.
4. V. Ekanayake, C. Kelly, and R. Manohar, An ultra low-power processor for sensor networks. *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems*, Boston, MA, October **(2004)**.
5. V. Ekanayake, C. Kelly, and R. Manohar, BitSNAP: Dynamic significance compression for a low-energy sensor network asynchronous processor. *Proceedings of the 11th International Symposium on Asyncronous Circuits and Systems*, March **(2005)**.
6. G. S. et al., Sensor network-based countersniper system. *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, Baltimore, MD, November **(2004)**.
7. T. R. F. Fulford-Jones, G.-Y. Wei, and M. Welsh, A portable, low-power, wireless two-lead EKG system. *Proceedings of the 26th IEEE EMBS Annual International Conference*, San Francisco, CA, September **(2004)**.
8. S. Hanson, B. Zhai, M. Seok, B. Cline, K. Zhou, M. Singhal, M. Minuth, J. Olson, L. Nazhandali, T. Austin, D. Sylvester, and D. Blaauw, Performance and variability optimization strategies in a sub-200 mV, 3.5 pJ/inst, 11 nW subthreshold processor. *IEEE Symposium on VLSI Circuits (VLSI-Symp)*, June **(2007)**.
9. M. Hempstead, N. Tripathi, P. Mauro, G.-Y. Wei, and D. Brooks, An ultra low power system architecture for sensor network applications. *The 32nd Annual International Symposium on Computer Architecture (ISCA)*, June **(2005)**.
10. M. Hempstead, M. Welsh, and D. Brooks, TinyBench: The case for a standardized benchmark suite for TinyOS based wireless sensor network devices. *Proceedings of the First IEEE Workshop on Embedded Networked Sensors(EmNets'04)*, Tampa, FL, November **(2004)**.
11. J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, System architecture directions for networked sensors. *Architectural Support for Programming Languages and Operating Systems* 93 **(2000)**.
12. C. Kelly, V. Ekanayake, and R. Manohar, SNAP: A sensor-network asynchronous processor. *Proceedings of the 9th International Symposium on Asyncronous Circuits and Systems*, Vancouver, BC, May **(2003)**.
13. L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis, Design and deployment of industrial sensor networks: Experiences from a semiconductor plant and the north sea. *Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, San Diego, CA, November **(2005)**.
14. M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, The flooding time synchronization protocol. *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, Baltimore, MD, November **(2004)**.
15. L. Nazhandali, M. Minuth, and T. Austin, Sensebench: Toward an accurate evaluation of sensor network processors. *IEEE International Workload Characterization Symposium*, October **(2005)**.
16. L. Nazhandali, M. Minuth, B. Zhai, J. Olson, S. Hanson, T. Austin, and D. Blaauw, A second-generation sensor network processor with application-driven memory optimizations and out-of-order execution. *ACM/IEEE International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, September **(2005)**.
17. L. Nazhandali, B. Zhai, J. Olson, A. Reeves, M. Minuth, R. Helfand, S. Pant, T. Austin, and D. Blaauw, Energy optimization of subthreshold-voltage sensor network processors. *The 32nd Annual International Symposium on Computer Architecture (ISCA)*, June **(2005)**.
18. M. Powell, S.-H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar, Gated-Vdd: A circuit technique to reduce leakage in deep-submicron cache memories. *International Symposium on Low Power Electronics and Design (ISLPED)*, June **(2000)**.
19. M. Sheets, F. Burghardt, T. Karalar, J. Ammer, Y. H. Chee, and J. Rabaey, A power-managed protocol processor for wireless sensor networks. *VLSI*, June **(2006)**.
20. V. Shnayder, M. Hempstead, B.-R. Chen, G. W. Allen, and M. Welsh, Simulating the power consumption of largescale sensor network applications. *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, Baltimore, MD, November **(2004)**.
21. R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, Lessons from a sensor network expedition. *Proc. the First European Workshop on Wireless Sensor Networks (EWSN)*, January **(2004)**.
22. A. Wang and A. Chandrakasan, A 180 mV FFT processor using subthreshold circuit techniques. *ISSCC*, January **(2004)**.
23. B. A. Warneke and K. S. J. Pister, An ultra-low energy microcontroller for smart dust wireless sensor networks. *ISSCC*, January **(2004)**.
24. B. A. Warneke, M. D. Scott, B. S. Leibowitz, L. Zhou, C. L. Bellew, J. A. Chediak, J. M. Kahn, B. E. Boser, and K. S. Pister, An autonomous 16 mm3 solar-powered node for distributed wireless sensor networks. *Int'l Conference on Sensors*, June **(2002)**.

25. G. Werner-Allen, J. Johnson, M. Ruiz, J. Less, and M. Welsh, Monitoring volcanic eruptions with a wireless sensor network. *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN)*, January **(2005)**.

26. G. Werner-Allen, K. Lorincz, J. Johnson, J. Less, and M. Welsh, Fidelity and yield in a volcano monitoring sensor network. *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, November **(2006)**.

27. A. Woo, T. Tong, and D. Culler, Taming the underlying challenges of reliable multihop routing in sensor networks. *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*, Los Angeles, CA, November **(2003)**.

28. B. Zhai, L. Nazhandali, J. Olson, A. Reeves, M. Minuth, R. Helfand, S. Pant, D. Blaauw, and T. Austin, A 2.60 pJ/Inst subthreshold sensor processor for optimal energy efficiency. *IEEE Symposium on VLSI Circuits (VLSI-Symp)*, June **(2006)**.

Mark Hempstead

Mark Hempstead *grew up in Yarmouth Maine, USA. He received a B.S. in computer engineering from Tufts University in 2003 and a M.S. in engineering from Harvard University in 2005. He was the winner of the SRC SoC chip design contest in 2006. He is currently a Ph.D. student in the School of Engineering at Harvard University. His research interests are power-aware computer architecture, low-power VLSI design, and wireless sensor networks.*

Michael J. Lyons

Michael J. Lyons *is a native of Northampton, Massachusetts, USA. He earned a B.S.E. in electrical engineering at the University of Pennsylvania in 2004. He subsequently worked at the Microsoft Corporation's Entertainment and Devices Division until 2006. He is currently a Ph.D. student in the School of Engineering at Harvard University. His research interests include energy-efficient hardware and software design for mobile, networked systems.*

David Brooks

David Brooks *joined Harvard University in September of 2002 and is currently an Associate Professor of Computer Science. Dr. Brooks received his B.S. (1997) degree from the University of Southern California and his M.A. (1999) and Ph.D. (2001) degrees from Princeton University, all in Electrical Engineering. Prior to joining Harvard University, Dr. Brooks was a Research Staff Member at IBM T. J. Watson Research Center. His research interests include architecture and software approaches to address power, reliability, and thermal issues for embedded and high-performance computer systems.*

Gu-Yeon Wei

Gu-Yeon Wei *joined Harvard University in January 2002 and is currently an Associate Professor of Electrical Engineering. Prior to joining Harvard, he spent 18 months at Accelerant Networks in Beaverton, Oregon. Professor Wei received his B.S., M.S., and Ph.D. degrees in Electrical Engineering all from Stanford University in 1994, 1997, and 2001. His current research interests are in the areas of mixed-signal VLSI circuits and systems design for high-speed/low-power wireline data communication, energy-efficient computing devices for sensor networks, and collaborative software + architecture + circuit techniques to overcome variability in nanoscale IC technologies.*