System Design Considerations for Sensor Network Applications

Mark Hempstead, Gu-Yeon Wei, David Brooks
School of Engineering and Applied Sciences
Harvard University
Email: {mhempste, guyeon, dbrooks}@eecs.harvard.edu

***Abstract*— Systems research in the emerging space of wireless sensor networks has exploded. Researchers have deployed nodes composed of a wireless radio, MEMS sensors and low power computation for applications from medical sensing to volcanic monitoring. We must consider several requirements — including the need for inexpensive, long-lasting, highly reliable devices coupled with very low performance requirements — when designing devices for wireless sensor networks. An untethered, fully-integrated node that operates off of energy scavenged from the ambient environment is the ultimate goal. We take an application-driven approach to the design of a wireless sensor network node. Our approach addresses the event-driven nature that is characteristic of many sensor network workloads.**

We have completed a detailed architectural analysis of this space using a full-system simulator and RTL model. From this analysis, we chose to implement a design that best achieves the power goals and performance requirements of wireless sensor network applications.

I. Introduction and Background

Driven by an explosion of systems research, wireless sensor networks (WSNs) are poised to transform the way society interacts with the physical world. WSNs have been deployed for a wide range of applications, such as; habitat and volcano monitoring [1], [2], structural monitoring, military applications, and emergency medical response [3], [4]. The operating lifetime of the battery-operated wireless sensor nodes that comprise the network actually limits the uses of WSNs. Current deployments rely on commercially available wireless sensor network nodes called *motes* (e.g., Mica2 Mote [5]). Commercial motes typically consist of an off-the-shelf microcontroller, a radio, and a variety of (often MEMS-based) sensors. Commodity chips are not specifically designed for wireless sensor networks, a key limitation to the energy efficiency of these systems. To address this limitation, this paper presents the design and analysis of an ultra low-power device created for wireless sensor network applications. In commercial motes, the CPU, radio, and sensor devices are responsible for the majority of the total system power. We show that the general-purpose nature of commodity microcontrollers results in inefficient power usage and thus presents an opportunity to significantly reduce its power.

We outline our design approach in this paper which is the study of all levels of the system design space, from the applications down to the circuits. This holistic approach enables us to uncover architectural and circuit-level design trade-offs. These trade-offs guide design decisions so that we can achieve our dual goals; ultra low-power, and a long deployment lifetimes. 100 μW is the power target of the system architecture for normal workloads. This low level was chosen with the ultimate objective of implementing a truly untethered device that can operate indefinitely off of energy scavenged from the environment.

Several architectural design features are motivated by the intermittent, event-driven nature of wireless sensor network application workloads. The architecture is optimized for the frequent, repetitive behavior characteristic of sensor network applications. We look at optimizations, including hardware acceleration and efficient event handling that we offload from the general purpose computing component. We eliminate unnecessary operating system overhead by including features such as event-driven computation and hardware acceleration to improve performance, which permits a slower system clock, and, thus, reduce power consumption. In order to meet the long-lifetime demands of many wireless sensor network deployments, our architecture enables fine-grain power management to minimize extraneous dynamic and static power consumption.

II. Architectural Motivation and Goals

A key aspect of our design philosophy is that the system architecture should be driven by the application space. Furthermore, our architectural design choices should enable key low-power circuit techniques. Existing sensor network devices utilize a microcontroller for the main processing engine. The microcontroller must idle in a mode that allows it to wait for timer and external events and process them as they arrive, even if these events occur rarely and in regular patterns (as is characteristic of simple monitoring applications). For applications with extremely low data rates, the main component of energy consumption can be idle power. Since wireless sensor network devices often perform repetitive actions — some of which can be common to a large class of applications — it is important to optimize the architecture of these devices for behavior commonly found in applications.

Our system employs an event driven architecture designed for the recurrent nature of sensor network applications. The general-purpose microcontroller spends most of the time in a low power state only waking to handle irregular events such as system reprogramming. The event processor, a small state machine, handles all system interrupt and transfers data between modularized accelerator components. Keeping the above discussion in mind, we summarize our architectural design goals. A detailed description of the design goals and architecture was presented at ISCA 2005 [6].

1) *Event-Driven Computation*
2) *Hardware Acceleration to Improve Performance and Power*
3) *Exploiting Regularity of Operations within an Application*
4) *Optimization for the Monitoring Class of Applications*

978-1-4244-1684-4/08/$25.00 ©2008 IEEE

Authorized licensed use limited to: Harvard Library. Downloaded on April 29,2022 at 13:24:34 UTC from IEEE Xplore. Restrictions apply.

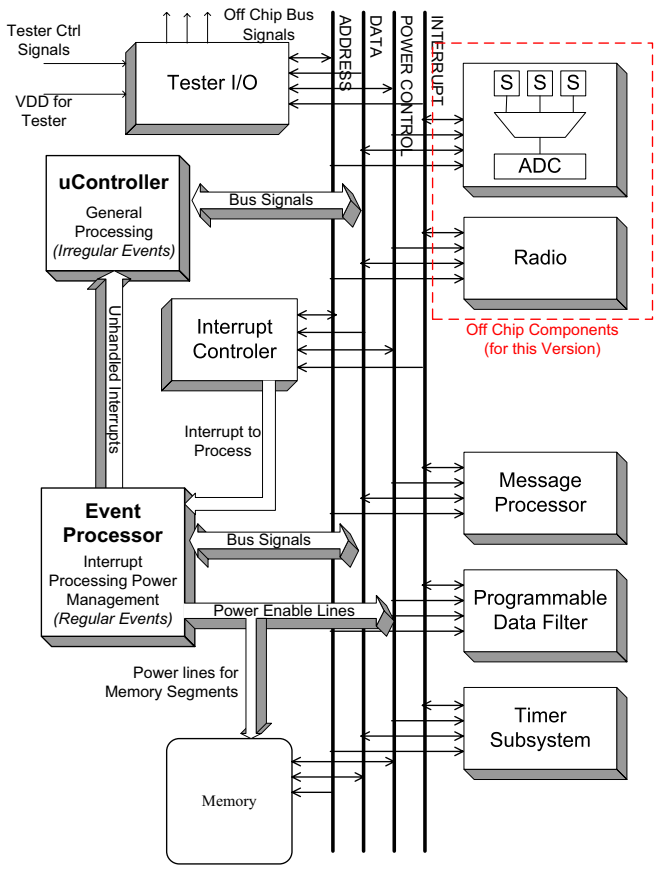

Fig. 1. **Block Diagram of System Architecture**

5) *Modularity*
6) *Fine-grained Power Management Based on Computational Requirements*

III. Architecture Description

The system architecture is illustrated in Figure 1. There are two distinct divisions within the system in terms of the component's ability 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 components: 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 the execution of the accelerators. The two master devices consist of a general-purpose microcontroller and a small state machine, the event processor.

Employing fine-grained power management of individual components (both masters and accelerators) is key benefit of the modular design of our architecture. Using VDD-gating to selectively turn-off components, allows us 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 idling. The event processor handles all interrupts, distributes tasks to accelerator devices, and wakes up the microcontroller only when necessary (which is rare). We now describe some of the more interesting system components in greater detail.

- **System Bus.** The system bus is comprised of 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 is a simple, general-purpose microcontroller. The microcontroller is used to handle irregular events, as discussed in the previous section, and also to set up the system at startup. We modified a core from opencores.org to serve as our general-purpose computation component.
- **Event Processor.** The event processor is essentially a state machine designed to perform the repetitive task of interrupt handling and, therefore, is the most active component.
- **Interrupt Controller.** The interrupt controller selects one of the interrupt lines for service. It provides for simple priority selection and the ability to mask interrupts.
- **Timer Subsystem.** The periodic nature of sensor network applications requires multiple, configurable timers supported by this subsystem.
- **Message Processor.** Our architecture seeks to leverage hardware accelerators designed for common tasks to improve power efficiency. Since communication is a frequent event, we implement a message processor block that handles regular message processing tasks such as message preparation and routing.
- **Radio.** The first version of the system assumes an off chip radio, the low power CC2420 802.15.4 radio from ChipCon [7]. However, we hope to include radio IP blocks into future versions of the SoC.
- **ADC and Sensors.** The ADC and sensors are an integral part of wireless sensor network devices. For this version of the system we will use an off-chip ADC and sensors.
- **Programmable Filter.** In sensor network devices, one must balance the power consumed by computation and communication. Often, data filtering and signal processing can significantly reduce communication requirements. Several wireless sensor network monitoring applications like volcano monitoring require that the sensor node be able to detect an *event* such as an eruption. Often, moving average filters are used to detect an event and can potentially avoid triggering due to random sampling noise. We implement an eight entry FIR filter capable of performing this function in our system. The filter component also includes a threshold filter as a feature.
- **I/O and Tester** To facilitate testing, data collection, and off-chip I/O, we include a comprehensive I/O and tester block. The tester component consists of a set of scan latches and off-chip drivers which allow an off-chip FPGA or logic analyzer to observe and control the internal bus. The tester is used for both programing the system and for debugging it.

IV. Simulation Results

We evaluated our system for performance and power. To complete this evaluation, we created a model of the system in systemC. We have implemented an entire system in a semi-custom 130nm design flow. The majority of the blocks were synthesized from verilog and

Authorized licensed use limited to: Harvard Library. Downloaded on April 29,2022 at 13:24:34 UTC from IEEE Xplore. Restrictions apply.

placed and routed using a standard cell library. We used a memory generator available with the standard cell library to implement two 2KB SRAMs. We attach VDD-gating transistors to one of the SRAMs while leaving the other SRAM untouched. The VDD-gating circuit was developed using a custom design flow and integrated with the rest of the system. The power estimates presented in this section are from full system circuit level simulations in 130nm CMOS.

A. Test Application

In order to evaluate our architecture, we began with the simplest application representative of existing real-world applications such as habitat monitoring [8]. We then added complexity to this application in stages; the final application is fairly complex, including standard sensing and transmission of data, multi-hop routing, and remote application reconfiguration.

We describe the four application versions according to the complexity added in each stage:

1) Periodically collect samples and transmit packets containing the samples.
2) Periodically collect samples and transmit packets containing the samples if it is above a certain threshold.
3) Receive and forward incoming messages from other sensor nodes.
4) Receive and handle incoming reconfiguration messages. (These messages include changes to the sampling period and the sensor threshold value.)

The base application collects samples and transmits the packets. For our architecture, the processing of a sample is initiated by the timer firing an interrupt. The event processor responds to this interrupt by sampling the ADC and transferring the value to the message processor. The message processor prepares the message and signals an event that causes the event processor to transfer the packet to the radio block and setup the radio for transmission.

In a multi-hop routing environment, message forwarding is expected to be a fairly frequent activity and we, therefore, map it as a regular event in our architecture. When a message arrives, an interrupt is fired by the radio block to indicate that a packet has been received. The event processor responds by transferring the packet to the message processor, which signals whether the message has been previously received (this is performed by searching for the packet ID in the routing table). If the message has been previously received, the packet is dropped, otherwise the event processor sets up the radio to forward the packet.

The last version of the test application contains two irregular events that require intervention from our general-purpose microcontroller. In this case, message handling is the same as in the preceding case until the message processor receives the packet. If the message processor determines that the message is not a simple forwarding request, then it signals an interrupt indicating that intervention by the microcontroller is required. The event processor wakes up the microcontroller in response to an irregular event signaled by the message processor. The microcontroller decodes the message to determine whether the timer needs to be reconfigured or whether the filter threshold needs reconfiguration.

Measurement	**Mica2**	**Our System**	**Speedup**
Total send path w/out filter	1522	102	14.9
Total send path w/ filter	1532	127	12.1
Process regular message	429	165	2.6
Process irregular message			
Timer change	234	136	1.7
Threshold change	11	114	0.096
Units	Cycles	Cycles	×

TABLE I

Comparison of cycle count for the test application written on our architecture and on TinyOS for the Mica Platform.

B. Cycle Performance Estimates

Cycle count results using our SystemC simulator and the Mica2 cycle simulator, Atemu [9], for each application task are shown in Table I. Each row represents the measurement of a particular segment of code. The first two rows provide measurements of the send path of our application as described in the previous section. The next two rows display cycle comparisons of the receive path for both regular and irregular messages.

For the Mica2 platform, processing a sample includes the software-equivalent implementation of our test applications, in addition to the overhead of TinyOS, required for context switching and task scheduling. Our architecture handles task scheduling natively in the design and, therefore, we see a large difference in cycle counts. Because our architecture is optimized for regular events, it does not show improvements for irregular tasks that require the general-purpose microcontroller.

It is clear that the emphasis of our proposed architecture, for typical events, has significant advantages over commodity systems in terms of the number of cycles required per task. These advantages enable our architecture to operate at significantly lower clock rates while maintaining sufficient performance to keep up with the 802.15.4 radio standard and process sensor data requests at a level required by typical applications.

As can be seen from Table I, the number of cycles taken to process one timer event for the sample, filter, and transmit application takes 127 cycles. The cycle count at 100 KHz gives us a maximum sample rate of roughly 800 samples/second. This maximum rate seems very reasonable considering the fact that most documented sensor network applications have sample rates less than a 100 samples/second. It should also be noted that the clock rate was chosen to accommodate the radio communication data rate of the 802.15.4 standard, 250 Kbits/second [10].

C. Power Estimation Methodology and Results

1) Methodology: We estimated the power consumption of the system by running simulations on a netlist extracted from the full system layout. We used the HSIM simulation tool from Synopsis. The functionality of the system was verified at the RTL level and the same test vectors were provided to HSIM. Wire cap was not extracted in order to reduce simulation time. To isolate the contributions of different components in the system, our circuit implementation

Authorized licensed use limited to: Harvard Library. Downloaded on April 29,2022 at 13:24:34 UTC from IEEE Xplore. Restrictions apply.

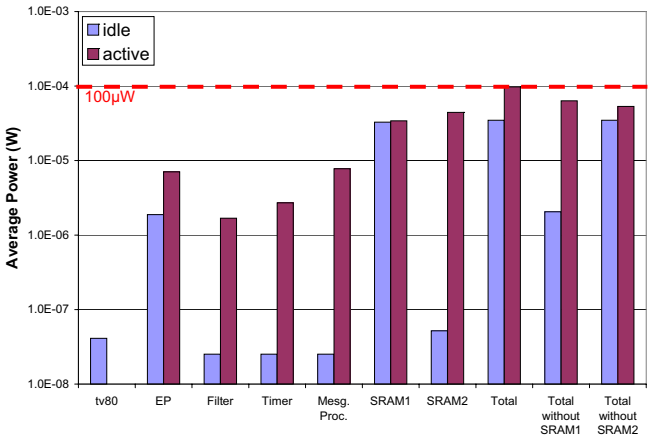

Fig. 2. **Power Estimates per block.** *Active power estimates indicates 100% activity. Estimates of idle power are for the lowest power state including vdd-gating when applicable. 130nm CMOS technology with 1.2 V supply running at 100 kHz.*

includes seven different power domains. We include separate power domains for the event processor, general purpose microcontroller, the accelerator blocks, SRAM1 (normal), SRAM2 (VDD-gated), tester component, and the I/O pads. The hardware accelerator blocks can be gated individually. We are concerned with regular operations only so the general purpose microcontroller is VDD-gated during the simulation.

We ran two different benchmarks on our system. The Power-Modeling benchmark exercises each of the system components individually. all of the components are VDD-gated if possible, the components that cannot be gated such as the event processor and SRAM1 sit idle. Using this benchmark, we can derive the upper and lower bound of our system's power consumption. Of course there is a cost to turning on and off the VDD-gating transistors but we do not capture it at this time.

The second benchmark is the sense application including data filtering described in Section IV-A. We measure the energy used processing a task and the energy consumed between tasks. The results of this measurement are used to develop a workload dependent power model for this application.

2) Power Estimates: The power estimates for the main components of the system are presented in Fig 2. The power numbers are shown for each component's and active and idle modes at a supply voltage of 1.2V and a clock frequency of 100KHz. These estimates give us the upper and lower bounds of system power consumption. A line indicating our goal of $100\mu W$ is drawn on the plot. Components that implement VDD-gating (filter, timer, mp, SRAM2) consume over two orders of magnitude less power when idle than when the component is active.

We notice that SRAM1 consumes roughly the same amount of power when idle as when reading and writing. Therefore, we conclude that leakage current is the primary source of power consumption for the SRAMs. Techniques such as drowsy-caches have been developed to reduce idle power while retaining data [11]. Hopefully, future versions of SRAM generators will implement similar techniques to reduce leakage power.

We provide bars for total power in several different configurations; total for the system, total without including SRAM1 and total without including SRAM2. Our total active power consumption is less than $100\mu W$. The total idle power consumption is less than $100\mu W$ if SRAM1 is used and in the tens of uW if the VDD-gated SRAM is used. These results assume nominal VDD of 1.2V: if the power supply was reduced, we could expect a further reduction in power consumption. The off-chip radio is not included in this plot. Depending on the duty cycle of the application and the amount of data filtering, it is possible trade communication for computation.

V. Conclusion

This work describes a holistic approach to the design of a wireless sensor network device. Employing an application-driven design philosophy, this work describes the application space, and a novel system architecture for sensor devices. In order to provide efficient operation and enable fine-grain power control, our architecture provides explicit support for the event-driven nature of sensor network applications and provides key functionality in separate hardware blocks. Our estimates for the key components of the system include a total active power of less than $\sim 100\mu$W and idle power of less than $\sim 10\mu$W depending on the SRAM used. These results represent a substantial savings over existing systems. We plan on investigating new ways to reduce the power consumption of our system, including new circuit implementations and expanding the capabilities of the hardware accellerators.

References

[1] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, Sept. 2002. [Online]. Available: citeseer.nj.nec.com/mainwaring02wireless.html

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

[3] T. R. F. Fulford-Jones, G.-Y. Wei, and M. Welsh, "A Portable, Low-Power, Wireless Two-Lead EKG System," in *In Proceedings of the 26th IEEE EMBS Annual International Conference*, San Francisco, CA, Sept 2004.

[4] K. Lorincz, D. Malan, T. R. F. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, S. Moulton, and M. Welsh, "Sensor networks for emergency response: Challenges and opportunities," *IEEE Pervasive Computing*, Oct-Dec 2004.

[5] Crossbow Technology Inc, "Mica2 sensor node," http://www.xbow.com.

[6] M. Hempstead, N. Tripathi, P. Mauro, G.-Y. Wei, and D. Brooks, "An ultra low power system architecture for sensor network applications," in *The 32nd Annual International Symposium on Computer Architecture (ISCA)*, June 2005.

[7] Chipcon AS, "CC2420 2.4GHz IEEE 802.15.4/ZigBee-ready RF Transceiver," http://www.chipcon.com.

[8] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, "Lessons from a sensor network expedition," in *Proc. the First European Workshop on Wireless Sensor Networks (EWSN)*, January 2004.

[9] M. Karir, J. Polley, D. Blazakis, J. McGee, D. Rusk, and J. S. Baras, "ATEMU: A Fine-grained Sensor Network Simulator," in *Proceedings of First IEEE International Conference on Sensor and Ad Hoc Communication Networks (SECON'04)*, Santa Clara, CA, Oct 2004.

[10] *IEEE 802.15.4 Standard*, ZigBee Alliance, http://www.zigbee.org.

[11] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy caches: Simple techniques for reducing leakage power," in *The 29th Annual International Symposium on Computer Architecture (ISCA)*, May 2002.

Authorized licensed use limited to: Harvard Library. Downloaded on April 29,2022 at 13:24:34 UTC from IEEE Xplore. Restrictions apply.