In Proceedings of the IEEE International Workshop on Computer Architecture for Machine Perception and Sensing, Montreal, Canada, September 2006.

A Rapid Prototyping Methodology for Application-Specific Sensor Networks

Chung-Ching Shen, Celine Badr, Kamiar Kordari, Shuvra S. Bhattacharyya,
Gilmer L. Blankenship, and Neil Goldsman
Department of Electrical and Computer Engineering
University of Maryland, College Park 20742
{ccshen, cbadr, kkordari, ssb, glimer, neil}@eng.umd.edu

Abstract

Wireless sensor network systems depend on many interrelated system parameters. The associated design space is vast, and effective optimization in this space is challenging. In this paper, we introduce a system-level design methodology to find efficient configurations for an application-specific sensor network system where optimization of energy consumption is a primary implementation criterion. This methodology incorporates fine-grained, system-level energy models; analyzes critical parameters of candidate off-the-shelf components; integrates the associated parameters into a comprehensive optimization framework; and applies optimized configurations to the actual hardware implementation of the targeted sensor network system. The results of fidelity analysis of the models that underlie our optimization framework together with the results of our hardware implementation demonstrate the accuracy and applicability of our methodology and supporting tools for optimized configuration of application-specific sensor networks.

1. Introduction

Generally, wireless sensor network (WSN) nodes combine four subsystems under constraints of limited hardware resources, very small size, and low cost. These subsystems are for computation, communication, sensing, and power [2, 7, 11]. For demonstrating any developed sensor network applications on real hardware platforms for the sensor nodes, users usually need to design experimental prototyping platforms for WSN systems by drawing from an increasing variety of off-the-shelf hardware and software components. Such components are often reconfigurable with different parameters across a range of settings to allow users to tune the functionality and associated implementation trade-offs. Building a complete WSN system is complicated and time consuming, and the design space associated with optimization of WSN configurations is vast due to the combinatorial growth of admissible system configurations and the complexity of interactions among component and system parameters. For efficient design space exploration in this challenging context, we introduce a system-level design methodology for prototyping WSN systems with optimized configurations for energy efficiency. By using off-the-shelf components for data processing, communication, control, and optimizing their configuration settings comprehensively in the context of the targeted application, our approach integrates advantages of commodity hardware (reuse of intensive component-level verification and optimization) and of application-specific, system-level analysis. Such an integration is highly useful for wireless sensor network applications, which must often satisfy severe constraints on size, energy consumption, and cost.

To support the methodology, we have developed a Java-based software tool that is based on careful modeling and extensive optimization of sensor network components along with their associated configuration options. The optimization approach in our tool is based on a general strategy called *particle swarm optimization* (PSO), which was introduced in [6] as a population-based, optimization technique for simulating the social behavior of individuals. In our prototyping methodology, we have developed a generic PSO package along with novel plug-ins to this package that provide customizations for WSN-related optimization. Given a user-defined sensor network, mutable and immutable parameters for configuring components and system parameters in the network, and application-specific models for evaluating the relevant design evaluation metrics as a function of the network configuration, our optimization framework derives an efficient WSN configuration that is optimized for minimum energy consumption. Due to the accuracy of the evaluation methods employed in our optimization framework, the effectiveness of the optimization approach, and the thoroughness with which configurations are managed, solutions derived from the framework can be mapped efficiently into hardware/software implementations of complete, application-specific WSN systems.

2. Related work

Akyildiz et al. [2] provide a comprehensive survey on applications, design factors, and communication architectures for WSN, including elaboration on the physical constraints on sensor nodes and protocols proposed in all network layers. Various research groups have built sensor node platforms with interesting combinations of features [1, 4, 7, 8]. These approaches generally involve off-the-shelf components, and include detailed measurement of power consumption or performance analysis from the constructed platforms. However, few such works are integrated with strategies for system-level modeling and optimization.

Singh et al. [12] discuss system-level trade-offs related to energy costs of state-of-art WSN technologies. An integrated, system level model and energy trade-off analysis is presented for application development in wireless networks. Jin et al. [5] discuss an approach to sensor network optimization of systematically clustering groups of WSN nodes using a genetic algorithm. In this paper, the authors discuss how clustering is an NP hard problem, and outline a method to determine an efficient selection of cluster heads using a genetic algorithm approach.

The technique of particle swarm optimization (PSO) [6] has been the subject of extensive research in recent years. Due to the simplicity of its implementation and the small number of parameters involved in its fine tuning, PSO has been used to solve optimization problems in a wide variety of applications. For WSN, the PSO approach has been adopted in [13] to optimize clustering techniques.

Our work differs from these approaches in that our approach aims to provide a more general methodology and associated computer-aided design tool for taking into account arbitrary combinations of WSN network configuration parameters and their interactions. The reason why we choose PSO as our optimization strategy is not only because the PSO-based approach converges relatively quickly, which we demonstrate from our experimental results, but also because it maintains the features of flexibility and scalability for integration with user-defined evaluation models and application-specific configuration formats. These features are especially useful for supporting optimized, energy-efficient configuration in the rapid prototyping of WSN systems with arbitrarily chosen components and parameters.

3. System-level energy modeling

For finding effective application-specific sensor network configurations based on energy consumption considerations, a system-level energy model is required for our design methodology. System-level energy consumption is determined by processors and communication interfaces, hardware configurations, and the dependencies imposed by the application. This integrated energy model is used in our optimization framework so that alternative system configurations can be evaluated by running simulations for estimating system-level energy consumption. The resulting approach to system-level modeling is one important contribution of this paper, and this contribution helps us to more comprehensively explore the design space of a sensor network application.

3.1. A WSN-based Application Example

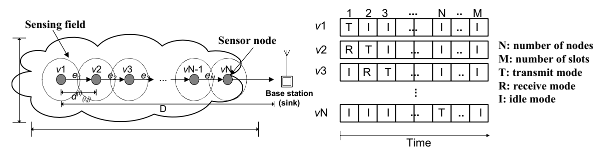
To illustrate the system-level modeling and evaluation capabilities of our optimization framework as well as the implementation of derived WSN configurations on our prototyping platform, we have developed an example of a WSN-based line-crossing application, where all the sensor nodes are placed as a linear network topology. Each node runs according to a TDMA scheme with three operation modes of transmission, reception, and idle status for preventing collisions during wireless communication.

Each sensor node enables its associated microphone sensor when it enters transmission mode and senses an acoustic signal from the environment. Whenever the sensed data is above a user-defined threshold, a message will be encoded with a data token as well as a node ID and a header from memory for transmission to the next node. The data token involved here is formatted with a separate bit for each node. Thus, once a token arrives the base station, it can easily be decoded to determine whether a moving body has crossed the line of sensor nodes, and if so, which nodes were closest to the line crossing event.

In the reception mode, a sensor node needs to enable its transceiver and wait for a message to arrive from the previous node in the sensor array. Upon receiving a message, the node will decode the message and compare its contents with the expected header and node ID. If both comparisons match, the data token will be stored into memory, and the node will wait for its next transmission slot according to the TDMA scheme. When a sensor node enters the idle mode, the microcontroller powers down the transceiver and sensor devices, and then it enters a low-power mode as well to save power throughout the rest of the idle interval. Figure 1 illustrates the operation of this WSN-based line-crossing application, as well as the associated TDMA operations.

3.2. Network-level Energy Modeling

We classify our system-level cross-layer energy models into the two levels of *network-level* and *node-level* modeling. For our network-level modeling, we first define a network topology based on a graph-based representation of the application, and then we identify the critical parameters that can affect energy consumption throughout the system.



	1	2	3	...	N	..	M
v1	T	I	I	...	I	..	I
v2	R	T	I	...	I	..	I
v3	I	R	T	...	I	..	I
⋮							
vN	I	I	I	...	T	..	I

Figure 1. A WSN-based line-crossing application shown with the associated TDMA scheduling.

Next, we formulate constraints associated with maintaining the minimum acceptable functionality from the overall application. Then, based on methods presented in [10], we derive a *power model* that represents the minimum received power strength at a given node for correct communication of data across nodes. Here, we assume that nodes can receive the correct data pattern as long as the received power strength is above a particular threshold.

For example, for a line crossing application with N nodes, the topology is illustrated in Figure 1, and is specified by $G = (V, E)$, where $V = \{v_1, \ldots, v_N\}$, and $E = \{(v_i, v_{i+1}), 1 \leq i \leq N-1\}$. The WSN-related parameters associated with this application are given by $P_T(e)$, $P_R(e)$, $A(v)$, $d(e)$, and f_c, where $P_{T/R}(e)$ are the transmitted/received power values associated with the edge e in the network topology; $A(v)$ is the effective area of the antenna for the node v; $d(e)$ is the distance between the transmitter and receiver that is associated with the edge e; and f_c is the carrier frequency. Based on these parameters, the power model described above can be formulated as $P_R(e) = (\frac{\lambda}{4\pi d})^2 \cdot P_T(e) \cdot G_T \cdot G_R$, where $G_R = \frac{4\pi \cdot A(v)}{\lambda^2}$, and $\lambda = \frac{c}{f_c}$ give the receiver antenna gain and wavelength, respectively; c represents the speed of light; and G_T gives the transmitter antenna gain. Our goal in network-level modeling and optimization is to find the minimum $P_T(e)$ for each node so that the energy consumption of the whole system-level application can be minimized while maintaining the functionality of the application.

3.3. Node-level Energy Modeling

A sensor node platform typically consists of a microcontroller for data computation and peripheral control, a transceiver for communication with other nodes, one or more sensors for data acquisition, and a battery for energy support. Based on the assumed scheme of TDMA operations in the network, we have integrated various models of node-level energy consumption for data acquisition, computation, and packet transmission. For the energy modeling of the data computation, we consider both the usual formulation for the power consumption in data processing [9] and the usage of peripherals by the microcontroller. In this context, the MSP430 microcontroller from Texas Instruments is our targeted microcontroller platform.

For energy modeling of the transmission mode, we model the active time for each device that is used in this mode. For example, the active time for the CPU core, A/D converter, UART, timer, and transceiver can be modeled, respectively, with the following equations:

$$\begin{aligned}
f_{clk} &= (2.14 \cdot V_{cc} + 0.296) MHz \\
t_{cpu-active}|_{tx} &= \frac{N_{cpu-active}}{f_{clk}}|_{tx} \\
t_{ADC-active}|_{tx} &= t_{sample} + \frac{13 \cdot ADCCLK}{5 \times 10^6} \\
t_{UART-active}|_{tx} &= t_{UART-startup} + \frac{M}{R} \\
t_{radio-active}|_{tx} &= t_{radio-startup} + \frac{M}{R}
\end{aligned}$$

Using these models of active time, the energy consumption in transmission mode can be modeled according to

$$\begin{aligned}
E_{t_s}|_{tx} &= (E_{mcu} + E_{radio} + E_{sensor})|_{tx} \\
&= E_{cpu-sleep} + E_{cpu-active} + \\
&\quad E_{ADC-active} + E_{UART-active} + \\
&\quad E_{timer-active} + E_{sensor-active} + \\
&\quad E_{radio-sleep} + E_{radio-active} \\
E_{cpu-sleep} &= I_{cpu-sleep} \cdot V_{cc} \cdot (t_s - t_{cpu-active}) \\
E_{cpu-active} &= C \cdot {V_{cc}}^2 \cdot f_{clk} \cdot t_{cpu-active} + \\
&\quad (V_{cc} \cdot I_0 \cdot e^{\frac{V_{cc}}{n \cdot V_T}}) \cdot t_{cpu-active} \\
E_{radio-sleep} &= I_{radio-sleep} \cdot V_{cc} \cdot (t_s - t_{radio-active}) \\
E_{radio-active} &= I_{radio}|_{tx} \cdot V_{cc} \cdot t_{radio-active} + \\
&\quad (P_{out} \cdot t_{radio-active}) \\
E_{sensor-active} &= I_{sensor-active} \cdot V_{cc} \cdot t_{ADC-active}
\end{aligned}$$

The equations above are formulations that we have derived to provide energy models for data acquisition, computation, and transmission for sensor, microcontroller, and transceiver devices. We also need models for peripheral control in the microcontroller. Here, for the internal devices in the microcontroller, we just use the average current consumption values for calculations because it is difficult to observe the actual current variations of each internal device on a chip. Thus, we employ models of the following forms:

$$\begin{aligned}
E_{ADC-active} &= I_{ADC} \cdot V_{cc} \cdot t_{ADC-active} \\
E_{UART-active} &= I_{UART} \cdot V_{cc} \cdot t_{UART-active} \\
E_{timer-active} &= I_{timer} \cdot V_{cc} \cdot t_s
\end{aligned}$$

For energy modeling in reception mode, where the sensor stays in a powered-down state, for example, the energy consumption can be modeled by using a similar approach as in transmission mode. The resulting models can be formulated as follows.

$$
\begin{aligned}
t_{cpu-active}|_{rx} &= \frac{N_{cpu-active}}{f_{clk}}|_{rx} \\
t_{UART-active}|_{rx} &= t_{UART-startup} + \frac{M}{R} \\
E_{t_s}|_{rx} &= (E_{mcu} + E_{radio})|_{rx} \\
&= E_{cpu-sleep} + E_{cpu-active} + \\
&\quad E_{UART-active} + E_{timer-active} + \\
&\quad E_{radio-active} \\
E_{cpu-sleep} &= I_{cpu-sleep} \cdot V_{cc} \cdot (t_s - t_{cpu-active}) \\
E_{cpu-active} &= C \cdot {V_{cc}}^2 \cdot f_{clk} \cdot t_{cpu-active} + \\
&\quad (V_{cc} \cdot I_0 \cdot e^{\frac{V_{cc}}{n \cdot V_T}}) \cdot t_{cpu-active} \\
E_{radio-active} &= I_{radio}|_{rx} \cdot V_{cc} \cdot t_s
\end{aligned}
$$

Also, the corresponding energy model for using the internal devices (UART and timer devices) in the microcontroller in reception mode are represented in the same way as in transmission mode.

For energy modeling of the idle mode, we need to consider that a typical sensor node platform can be turned off so that there is no execution of operations for computation nor communication. After such turning off, the microcontroller and transceiver remain in power saving states until they are activated again. For this scenario, energy models can be derived as follows:

$$
\begin{aligned}
E_{t_s}|_{idle} &= (E_{mcu} + E_{radio})|_{idle} \\
&= (E_{cpu-sleep} + E_{timer}) + E_{radio-sleep} \\
&= (I_{cpu-sleep} + I_{timer} + I_{radio-sleep}) \cdot V_{cc} \cdot t_s
\end{aligned}
$$

Note that a timer is required in our assumed implementation target for coordinating TDMA operations. Thus, the energy consumption for that timer device is considered in the energy model for each mode. Table 1 summarizes the symbols that we use for the energy models that are developed in this section.

4. Optimization framework

As discussed in Section 1 , our optimization strategy is built around the framework of particle swarm optimization (PSO).

4.1. Overview

In our exploration tool, PSO is implemented as a generic optimization package using Java, and used as a search technique for exploring combinations of mutable components.

Table 1. Notation for energy modeling.

Symbols	Description	Symbols	Description
$N_{cpu\text{-}active}$	number of clock cycles executed by CPU	f_{clk}	processor clock frequency
$t_{radio\text{-}startup}$	startup time from power-on to valid transmit/receive	$ADCCLK$	sampling cycles for ADC device
$t_{tx/rx}$	transmit/receive on time	P_{out}	transmission output power
t_s	slot time	C	total switching capacitance
$t_{device\text{-}active}$	active time for devices: ADC, UART, or timer	M	transmission message length
t_{sample}	ADC sampling time	R	data rate
$t_{UART\text{-}startup}$	UART startup time	I_0	processor leakage current
I_{sleep}	average current consumption in sleep mode with respect to corresponding devices	V_T	processor threshold voltage
I_{device}	average current consumption for device: ADC, UART, or timer		

Mutable parameters are defined as parameters that can be tuned during design space exploration. In our PSO package, a swarm of particles is initialized with random particle positions and velocities, and then this swarm is iteratively operated on to explore the underlying design space until a pre-specified exit condition is satisfied. Specifically, the condition for exiting is that either a solution is found having a cost function value that is within a pre-specified range, or a fixed maximum number of iterations has been completed. An interfacing class is included to serve as a connection layer between an application that uses PSO and the PSO package itself. The role of this interfacing class is mainly to convert input information from the application-specific format into swarm particle data, and to format swarm results to be displayed as output.

The application-specific part consists of implementations of the *fitness* evaluator (i.e., the mapping of candidate solutions into values of the relevant cost function), and the *move* algorithm for particles, in addition to class extensions needed for the application representation. In particular, the fitness evaluator and the application-specific classes comply with the requirements of the line-crossing application model described in Section 3, with the cost function being derived from the corresponding system-level energy models of the sensor network. Another application-specific addition to PSO comprises the extension of the coordinate class model to include properties pertaining to the sensor network application. That is, a WSN particle's coordinates are a representation of the sensor node properties, and of the network parameters being optimized.

4.2. Estimation of Fidelity

We calculate the fidelity of the estimator employed in the optimization framework based on the results of simulated vs. measured energy consumption, where in these experiments we continue to use the linear, line-crossing application as the driving application example, and for each simulated configuration, we evaluate the same configuration on our hardware testbed (see Section 5) to obtain a corresponding measured energy consumption result. The estimator is based on the energy models developed in the previous section and is used for simulation-based fitness evaluation of

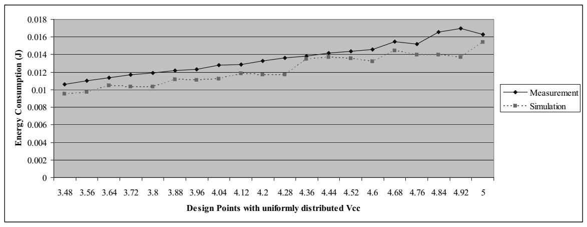


Figure 2. Accuracy of energy consumption estimation for 20 design points.

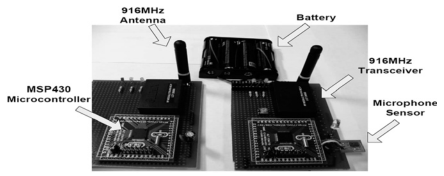


Figure 3. WSN prototype platform.

Table 2. Settings of immutable parameters with values.

immutable parameters	values	immutable parameters	values	immutable parameters	values
$N_{cpu\text{-}active}\|_{tx}$	579	$I_{cpu\text{-}sleep}$	$200uA$	R	$9600bps$
$N_{cpu\text{-}active}\|_{rx}$	309	$I_{sensor\text{-}sleep}$	$200uA$	f_{clk}	$8MHz$
$t_{cpu\text{-}active}\|_{tx}$	$0.072ms$	I_{ADC}	$300uA$	C	$100pF$
$t_{cpu\text{-}active}\|_{rx}$	$0.039ms$	I_{timer}	$300uA$	I_0	$2mA$
$t_{radio\text{-}startup}\|_{tx}$	$3mA$	$I_{radio\text{-}sleep}$	$200uA$	t_s	$125ms$
$t_{ADC\text{-}active}$	$53.6us$	I_{UART}	$300uA$	$t_{radio\text{-}active}$	$6.33ms$
$t_{radio\text{-}startup}\|_{rx}$	$6ms$	N/A	N/A	N/A	N/A

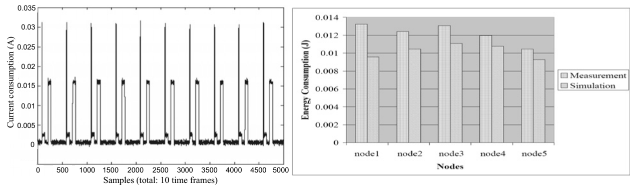


Figure 4. Plot of current consumption measured from the prototype platform.

candidate solutions. In this way, the user-defined evaluation model in the PSO framework can be shown to have high accuracy compared to the measured results. Thus, solutions obtained from PSO can be applied with high confidence to build a prototype system. For the equations of fidelity calculation, we refer the reader to [3] for more details on this model for estimation of fidelity.

For experimenting with the calculation of fidelity for our optimization framework, we generated 20 design points, where each design point consists of three critical parameters from the models, in particular, V_{cc}, $ADCCLK$, and P_{out}. We uniformly distributed the V_{cc} values and randomly selected values for the other two parameters over the 20 design points. From these experiments, we obtained a fidelity value of 0.89474 (from the detailed fidelity formulation in [3], a fidelity value of 1 corresponds to perfect fidelity). Figure 2 shows the measured and simulated energy consumption results for our fidelity calculation.

5. Implementation and evaluation

Our experiments have been carried out with a prototype WSN platform, illustrated in Figure 3, that we have developed at the University of Maryland. The platform is equipped with a Texas Instruments MSP430 microcontroller and an 916MHz transceiver.

For evaluating WSN-related optimized configurations, we implemented the line-crossing application described in Section 3.1, and we conducted experiments with mutable parameters chosen in Section 4.2 to compare energy consumption results associated with simulation from the PSO-based optimization framework, and measurement from the constructed prototype platform. In addition, in our experiments we employed the settings in Table 2 as immutable parameter values.

The measured current consumption result from one node on the prototype platform is shown in Figure 4(left). We used a 4GHz digital phosphor oscilloscope to measure the corresponding voltage variations on each platform for 10 time frames of execution, where each time frame takes 8 TDMA time slots with 125ms for each time slot. The actual energy consumption on the prototype platform can be calculated according to: $E = \int_t P(t)dt$. The experimental results of the optimized configurations for the whole 5-node line-crossing application through our optimization framework are shown in Figure 4(right). We compared the results from the simulation of the optimization framework and measurement from the corresponding implementations on our WSN testbed. For these comparisons, we chose 20 particles with $c_1 = c_2 = 2.0$ and $\omega = 0.95$ when running the PSO optimization algorithm for the experiment.

We conducted our test runs for simulation by varying the tightness of the binding restriction around the target "optimum" value. That is, since our fitness function measures the absolute offset from a pre-specified target value, we changed the range in which a fitness value that is not exactly equal to the target will nonetheless be considered as an acceptable solution, and trigger termination of the search. In the example of the 5-node line-crossing application, we chose the target optimum value as 0.05(J) for total energy consumption during one simulation time frame. One set of candidate results generated from our optimization framework for configuring the whole application with chosen mutable parameters and a binding constraint of *0.0013* are

Table 3. Candidate result for configuring the 5-node line-crossing application with binding constraint ± 0.0013.

Node I.D.	V_{cc}(V)	ADCCLK(CC)	P_{out}(dBm)	d
1	5.25	1024	-10	15.79
2	4.59	1024	-3	28.13
3	3.81	1024	-7	27.85
4	3.94	512	-11	10.62
5	3.70	256	-8	11.79

Table 4. Number of iterations and percentage of runs that found a solution using various binding values.

binding constraints	±0.0011	±0.00115	±0.00117	±0.0012	±0.00125	±0.0013
avg. iterations	5.67	10.5	8.5	6.43	5.25	6.67
rate of success	30	50	60	70	80	90

listed in Table 3. From such candidate results, we can verify that required transmission power increases with distance, and we can quantify this fundamental dependence in terms of the technology that we are using in our targeted platform.

We noted the number of iterations the program performed before reaching a solution within the range of each exit bound imposed on the search. For each binding constraint, 10 runs were performed, and the average numbers of iterations from these runs are shown in Table 4. Also, Table 4 shows the ratio of the number of solutions found out of each set of 10 runs when we conducted our tests for each of the binding constraints. While running on a 1GHz Intel Pentium workstation with 512MB RAM, the average time for converging to the optimal solution is approximately 2 seconds if a solution can be found; otherwise, the average time for evolving the swarm through 4000 iterations (the maximum number of iterations allowed before the optimization terminates) and 20 particles is approximately 150 seconds. From these results, we observe that the PSO can explore a vast design space with a relatively high speed of convergence (the speed for finding a solution), and the results for the rate of success confirm our expectation that a tighter binding constraint around the target results in a smaller percentage of successful runs.

6. Conclusions

This paper has explored a system-level design methodology to derive optimized configurations for prototyping application-specific, wireless sensor networks. In this paper, we have proposed a number of fine-grained, system-level energy models as efficient evaluation metrics in the PSO-based framework for WSN system analysis and optimization. To demonstrate the efficacy of our models and optimization methods, we used configurations that were derived from our optimization framework to map a practical WSN application into complete hardware/software implementations. From these implementations, we analyzed various parameters from the models that we employed, and we calculated the fidelity of the high level estimation methods used in our optimization framework. Our results showed that, relative to their high level of abstraction and efficiency in exploring the design space, our integrated models and estimation techniques result in a high accuracy of relating candidate solutions during optimization to their equivalent realizations as actual WSN system implementations.

References

[1] J. R. Agre, L. P. Clare, G. J. Pottie, and N. Romanov. Development platform for self-organizing wireless sensor networks. In *Proceedings of the International Symposium on Aerospace/Defense Sensing, Simulation, and Controls*, April 1999.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, pages 393–422, March 2002.

[3] N. K. Bambha and S. S. Bhattacharyya. A joint power/performance optimization technique for multiprocessor systems using a period graph construct. In *Proceedings of the International Symposium on System Synthesis*, pages 91–97, Madrid, Spain, September 2000.

[4] J. Hill and D. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, pages 12–24, November 2002.

[5] S. Jin, M. Zhou, and A. S. Wu. Sensor network optimization using a genetic algorithm. In *Proceedings of the World Multiconference on Systemics, Cybernetics, and Informatics*, July 2003.

[6] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, November 1995.

[7] M. Kohvakka, M. Hnnikinen, and T. D. Hmlinen. Wireless sensor prototype platform. In *Proceedings of the IEEE Industrial Electronics Society*, pages 1499–1504, 2003.

[8] R. Min, M. Bhardwaj, S. Cho, A. Sinha, E. Shih, A. Wang, and A. Chandrakasan. An architecture for a power-aware distributed microsensor node. In *Proceedings of the IEEE Workshop on Signal Processing Systems*, October 2000.

[9] K. K. Parhi. *VLSI Digital Signal Processing Systems: Design and Implementation*. John Wiley & Sons, Inc., 1999.

[10] J. Paul and M. G. Linmartz. *Wireless Communication, The Interactive Multimedia CD-ROM*. Baltzer Science, 1996.

[11] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava. Energy-aware wireless microsensor networks. In *IEEE Signal Processing Magazine*, pages 40–50, March 2002.

[12] M. Singh and V. K. Prasanna. System level energy trade-offs for collaborative computation in wireless networks. In *IEEE International Conference on Communications*, May 2002.

[13] J. C. Tillett, R. M. Rao, Sahin, and T. M. Rao. Particle swarm optimization for the clustering of wireless sensors. In *Proceedings of SPIE*, pages 73–83, 2003.