Design Methodology for PicoRadio Networks

J. L. da Silva Jr., J. Shamberger, M. J. Ammer,
C. Guo, S. Li, R. Shah, T. Tuan, M. Sheets,
J. M. Rabaey, B. Nikolic, A. Sangiovanni-Vincentelli, P. Wright

University of California at Berkeley

Abstract

One of the most compelling challenges of the next decade is the "last-meter" problem, extending the expanding data network into end-user data-collection and monitoring devices. PicoRadio supports the assembly of an ad hoc wireless network of self-contained mesoscale, low-cost, low-energy sensor and monitor nodes. While technology advances have made it conceivable to deploy wireless networks of heterogeneous nodes, the design of a low-power, low-cost, adaptive node in a reduced time to market is still a challenge. We present a design methodology for PicoRadio Networks, from system conception and optimization to silicon platform implementation. For each phase of the design, we demonstrate the applicability of our methodology through promising experimental results.

1. Introduction

Current technology allows us to build and deploy dense wireless networks of heterogeneous nodes collecting and disseminating wide ranges of environmental data. An inspired reader can easily imagine a multiplicity of scenarios in which these sensor and actuator networks might excel. To just mention a few: environmental control in office buildings, robot control and guidance in automatic manufacturing environments, warehouse inventory, integrated patient monitoring, diagnostics, and drug administration in hospitals, interactive toys, the smart home providing security, identification, and personalization, and interactive museums. The mind-boggling opportunities emerging from this technology indeed give rise to new definitions of distributed computing and user interface.

Crucial to the success of these ubiquitous networks is the availability of small, lightweight, low-cost network elements, which we call PicoNodes. These nodes must be smaller than one cubic centimeter, weigh less than 100 grams, and cost substantially less than one dollar. Even more important, the nodes must use ultra-low power to eliminate frequent battery replacement. We envision a power-dissipation level below 100 microwatts, as this would enable self-powered nodes using energy extracted from the environment, an approach called energy scavenging or harvesting.

This paper describes the main challenges and opportunities in PicoRadio Networks (Section 2), the system conception and optimization phase of the design process (Section 3), and the design methodology to develop a silicon platform for a node of the network (Section 4).

2. Challenges and Opportunities

To put our power dissipation goals into perspective, we can compare it with the state-of-the-art commercial devices available today. One of the closest matches is the Bluetooth transceiver [1], an emerging standard for short-range wireless communications. While meeting the volume requirement, Bluetooth radios cost more than 10 dollars and consume more than 100 milliwatts. Although Bluetooth's price point and power consumption will inevitably drop with technology scaling, this will by no means suffice to address the orders-of-magnitude reductions required for sensor network applications.

To reach these aggressive power dissipation levels, we must limit the effective range of each PicoNode to a couple of meters at most. Extending the reachable data range requires a scalable network infrastructure that allows distant nodes to communicate with each other. A self-configuring ad-hoc networking approach is key to the deployment of such a network with many hundreds of nodes.

Reducing the PicoNode's energy dissipation to the sub-miliwatt level is our focus here. The secret lies in a meticulous concern for energy reduction throughout all layers of the system design process. The largest opportunity lies in the protocol stack where a trade-off between communication and computation, as well as elimination of overhead, can lead to a many orders-of-magnitude energy reduction. An efficient configurable silicon platform can also contribute to large power savings. Other opportunities lie in the adoption and introduction of novel self-optimizing radio architectures and opportunities for energy scavenging.

This section presents an application example (Section 2.1), the main characteristics of PicoRadio Networks (Section 2.2), an introduction to multi-hop networks (Section 2.3), and energy scavenging possibilities for PicoRadio (Section 2.4).

2.1 PicoRadio Application Example

As an example application of PicoRadio networks, consider the management of environmental control systems in large office buildings. Any person who has spent a significant amount of time in such an environment is acutely aware of its problems: The temperature or the airflow is never right, and there is too little or too much light. A distributed building monitor and control approach might go a long way in addressing these problems, for example, creating local microclimates adapting to an occupant's preferences through distributed air-ducts, might vastly improve the living conditions for the building's population. At the same time, such an approach can dramatically reduce the energy budget needed to manage the environment. First-order estimations indicate that such technology could reduce source energy consumption by two-quadrillion BTUs (British Thermal Units) in the US alone. This translates to \$55 billion per year, and 35 million metric tons of reduced carbon emissions.

Wiring the huge number of sensor and actuator nodes needed to deploy such a system is impractical and uneconomical. The cost of installing wiring for a single sensor in a commercial building averages \$200 in addition to the cost of the sensor. For low-cost devices such as temperature sensors, the cost of the wiring may be as much as 90 percent of the installed cost. In these cases, eliminating the cost of wire by using a wireless connection could reduce the installed cost per sensor by an order of magnitude and enable the deployment of ubiquitous sensor networks in contrast to the currently used sensor-starved solutions. We can even envision a future in which the sensor nodes are prebuilt into construction materials such as ceiling and floor tiles. To realize this vision, the communication/sensor nodes must be completely self-contained for the lifetime of the building.

2.2 Ultra-Low Energy PicoRadio Networks

The scenarios detailed above expose both the challenges and opportunities that PicoRadio networks offer in terms of energy efficiency. A number of prime properties are worth identifying:

- Sensor data rates are quite low, typically less than one hertz.
- Sensor nodes don't need to be awake all the time; in fact, a single node's activity duty cycle is typically less than 1 percent.
- Sensing data without knowing the sensor's location is meaningless. Localization should therefore be considered as an implicit feature of the sensor network. This greatly simplifies the network discovery and maintenance effort and leads to substantial energy savings. For example, the sensor network can prune requests for information and direct them to the region of interest.
- Sensor networks require different addressing techniques than traditional data networks. Data requests are typically in the style of "Give me the temperature readings in room 30," compared to "Set up a connection between node A and B." The content- and localization-based addressing concepts make the overall network discovery and management a lot simpler.

Based on these specifications and properties, we can develop energy-efficient network, transport, media-access, and physical layer protocols. These in turn set the constraints and requirements for the hardware architecture and components of the transceiver nodes, including radio frequency (RF), base-band, and protocol processors. A number of innovations at the protocol stack level will make the intended energy reductions possible (Section 3).

2.3 Multihop networks

A main challenge in the design of an energy-efficient wireless network is that sending a bit of information through free space directly from node A to node B incurs an energy cost E_t, which is a strong function of the distance d between the nodes. More precisely, $E_t = \beta \times d^\gamma$, with $\gamma > 1$ as the path-loss exponent (a factor that depends on the RF environment, and is generally between 2 and 4 for indoor environments) and β is a proportionality constant. Given this greater than linear relationship between energy and distance, using several short intermediate hops to send a bit is more energy-efficient than using one longer hop. For example, assuming $\gamma = 4$, which is a common case in indoor environments, and $\beta = 0.2$ femtojoules/meter$^\gamma$, one hop over 50 meters requires 1.25 nanojoules per bit, whereas five hops of 10 meters require only 5×2 picojoules per bit. The multihop approach in this example reduces transmission energy by a factor of 125. This situation is somewhat analogous to the problem of sending a bit over a wire on a chip, where the introduction of intermediate repeaters can help to increase the performance and energy efficiency.

In its simplest form, multihop network energy analysis argues for an infinite number of hops over the smallest possible distance. In reality, however, the number of nodes between A and B limits the number of intermediate hops. Moreover, we must include not only the energy radiated through the antenna, but also the energy dissipated in the radio for receiving the bit and readying the bit for retransmission. (Given the relative costs of transmission and processing, we can compute an optimal number of hops.)

This leads to some interesting observations:

- Technology scaling will gradually reduce the cost of processing, with transmission cost remaining constant. Thus, shorter hops will become more favorable over time.
- Computation cost is not a constant either. Using compression techniques, we can reduce the number of transmitted bits, thus reducing the cost of transmission at the expense of more computation. This only makes sense if the communication cost dominates, as with long distance connections.

This *communication-computation trade-off* is one of the core ideas behind the low-energy networks we propose. The optimal trade-off has to be determined adaptively, based on data properties, node densities, and environmental circumstances. This dynamic nature has a profound impact on the hardware composition and architecture of the network nodes.

2.4 Energy Scavenging

Our project's Holy Grail is for the PicoNodes to be self-contained and self-powered using energy extracted from the environment. Reaching this goal requires new advances both in reducing the nodes' energy consumption and in increasing the amount of energy the nodes can extract from the environment.

Harvesting ambient energy requires compliance with two major constraints: applicability within the environments envisioned for the PicoNodes (office buildings and homes) and the size constraint of the one cubic centimeter chip.

Although batteries can store harvested energy that can't be used immediately, a continuous source of energy is desirable. Solar cells can contribute up to 15 milliwatts per square centimeter during direct sunlight hours and up to 0.15 milliwatts on cloudy days. Averaging over daylight and nighttime hours, and considering nodes in the interior of the building or embedded in ceiling tiles, shows that solar cells can just barely serve as the sole energy source for PicoNodes, and additional sources of energy would be welcome.

Harvesting energy from vibrations is promising for this application. Raised floors and dropped ceilings in most office buildings exhibit measurable vibrations (from trucks driving down nearby streets and people walking on the raised floors) that can be harnessed. Advances in MEMS devices make integrated and tiny variable capacitors a reality. These capacitors are used to make chip-scale electrostatic vibration generators that will integrate well with the other PicoNode components. Power outputs between 10-100 microwatts per cubic centimeter are plausible from vibrations in a normal office building using existing MEMS technology.

3. System Conception and Optimization

Traditional efforts in low power design have focused on optimizations at the circuit level. We believe that in order to meet the aggressive design goals of the PicoRadio project, it is necessary to begin the design at the system level. Starting at a higher level enables us to explore a larger design space, and arrive at a more optimal solution. The goal of this section is to present a methodology for identifying and evaluating decisions made in the design of the PicoRadio protocol stack.

Section 3 introduces the design methodology and the use of UML for system specification (Section 3.1), the network layer design issues, (Section 3.2), and finally discusses the Media Access Layer (MAC) design (Section 3.3).

3.1 Design Methodology Overview

One of the goals of a design methodology is to orthogonalize concerns whenever possible. This allows for a more efficient exploration of the design space, since each orthogonal concern can be optimized separately from the others. Such concerns are function/architecture [2] and communication/computation [4], the latter being the more novel idea in this design methodology. The essence of the design methodology is to identify the computational kernels, or *processes,* and then implement communication between them using a series of *adapters.* The object of this is to separate the behavior of a component from how it interacts with other components or the environment, providing location transparency, which in turn promotes component reuse.

The PicoRadio protocol has been designed using such a communication based design methodology. The first step in this design flow is the capture of specifications and functional decomposition at the system level. For this task we have used the Unified Modeling Language (UML) [5], which is discussed in the following section. The second step is functional description, which has been performed using the Specification and Description Language (SDL) [16] and Cadence's Virtual Component Codesign (VCC) tool [3]. This representation of the system can be functionally verified in VCC or SDL, but for performance measurements the description must be ported to one of several different simulation environments, including OpNet [14], MatLab [15], or a testbench wireless platform. When a satisfactory design has emerged, the functional description can be implemented.

Next, we present the work in system specification, functional description, and simulation stages followed by a brief review of the protocol design for the Smart Building scenario.

3.1.1 UML for system specification

The Unified Modeling Language (UML) is used to capture the structure of the system at a high level of abstraction. The UML diagrams aid in identifying design decisions and tradeoffs during the design process, and serve as a form of documentation for components and the interfaces between them in the finished design. Information captured about the application requirements is used to drive simulations, which ensures that the system design will meet all the performance constraints.

The different diagrams supported in the UML capture different aspects of the system, and are used at different points in the design process. The diagrams we use in our modeling are *use case diagrams, class diagrams, state diagrams* and *sequence diagrams.*

Use cases are the first step, and serve to document the interactions between the environment and the system, thus characterizing the requirements the system must meet. Once these interactions with the are identified, we describe the components that make up our system. The structure is described using class diagrams and the behavior using state diagrams.

We have defined primitives to capture particular semantics specific to our domain, using the stereotyping mechanism provided in UML. The stereotypes we have defined are *process, channel,* and *signal.* A process represents a reactive component, which has a set of input and output ports, each of which has a signal type associated with it. This structural description of a process is equivalent to a Codesign Finite State Machine (CFSM), the basic building block in VCC. Channels represent the communication paths between processes. They are characterized by the relationship of the output signal to the input signal, which can be captured with a state diagram for a complex channel. Signals are the datatypes that are input or output from a process.

Using the above stereotypes we can begin to describe the functionality of the system. At the top level, the behavior of the system is decomposed into a set of communicating processes, with ideal channels providing communication between them. If the behaviors of the processes are incompatible, it is necessary to insert *behavior adapters*. These incompatibilities arise when a sending process produces output signals that are not valid inputs of the receiving process. A behavior adapter must map these invalid signals into acceptable signals.

Once the behaviors are compatible, it is necessary to refine the communication medium. This refinement can be performed in multiple steps, and at each step abstractions made in the original assumption of an ideal channel are removed. When an abstraction is removed, it is necessary to ensure that communication over the refined channel is equivalent to that over the original channel. If this is not the case, it is necessary to insert *channel adapters* to resolve this situation. As an example, consider a system with two processes that require lossless communication. When we first define the behavior of the processes, we assume communication over an ideal lossless channel. As a result of mapping to specific communication architecture, we have a revised channel that is lossy. Thus we insert channel adapters which implement lossless communication over a lossy channel, using a method such as an acknowledgement/retransmission scheme or error correcting codes. As long as the resulting communication meets all other constraints, such as latency or power consumption, the new channel is successfully adapted.

The final representation captures all the processes and adapters in our system, and the interfaces and signals between them. However, we have only captured the structure, not the behavior. It is now necessary to choose the appropriate Model of Computation (MoC) to describe the behavior of each block, both processes and adapters and attach a description to the block. UML currently only supports state machines as MoC for behavioral descriptions. Since the design of the PicoRadio protocol will be implemented using CFSMs in VCC, this is a perfect match.

To further document the interaction between different components, sequence diagrams are used. Sequence diagrams capture the exchange of signals between two or more components during the transactions identified in the use case diagrams. These traces of signal exchanges can serve as an aid in debugging a functional description and ensuring it is performing as intended.

3.1.2 Smart Building Scenario

We illustrate the use of our methodology on the Smart Building example described in Section 2.1. There are three distinct classes of applications in the network: sensors, actuators, and monitors. Sensors take measurements of some aspect of the environment and produce a data value. Actuators accept a command and cause some change to happen in the physical surroundings. Monitors are the controlling applications in the system, which can issue requests to sensors, commands to actuators, and accept data from sensors. The monitors could be embedded control applications, user-interface nodes that allow the users to observe and change their environment, or system administration nodes that control system-wide operation, log performance data, and allow for system setup and debugging.

As a result of the decomposition of the system and refinement of the communication media, we have identified the layers of the protocol, and the design decisions to be made in each of them. The layers described below correspond to the adapters introduced during the communication refinement process.

By analyzing the needs of the application, two application-specific design decisions were identified. The first is to allow monitors to place periodic or threshold based requests, since many of the control algorithms require periodic data. These periodic requests reduce the network traffic, and thereby save power. Adding this functionality requires a behavior adapter at the sensor to match the monitor's behavior. This behavior adapter forms an *application layer*, which allows a sensor to be programmed for periodic or conditional requests. The second decision is based on the fact that the topology of this system will be fairly static. Most sensor or actuator nodes will be stationary. Thus we can allow the application a mechanism to perform data aggregation. For example, if a user wishes to know the average temperature from a region on the other side of the building, the average temperature can be computed locally in that region, and only this average value is sent back to the user. This creates less network traffic than having the individual data samples sent back to the user, and thus is an optimization as long as the cost of computation required for the aggregation is less than the cost saved in network traffic. In order to provide this service we add an *aggregation layer*.

Below the application layer, there are two layers needed to support multiple connections on one PicoNode. The first is an *addressing layer*, which takes the multiple input and output streams from the layers above and maps them to one network connection. It also places the data in the correct format for network packets. The second layer is a *muxing layer*, which allows multiple applications to run on one node. This layer has to decide to which applications on a node to deliver incoming packets.

The lowest layers in our protocol are the *routing layer*, whose purpose is to route packets to their destination in a multi-hop environment, and the *MAC layer,* which allocates channels among nodes and controls access to the shared medium. A key point guiding the design of these two layers is that the application in our scenario generates very low bit-rate data, with a low duty cycle. This leads us to design a network layer which routes reactively, rather than expending a lot of energy on maintaining routing tables. For the MAC layer, we search for an alternative that will allow us to shut off the radios as much as possible to conserve power. In the following sections we present more detail on the power optimizing design of these two layers.

3.2 Network Layer Design

The network layer lies beneath the application and aggregation layers, and performs the functions of addressing, multiplexing and routing. There are a number of existing routing protocols for ad-hoc wireless networks such as Destination-Sequenced Distance Vector (DSDV) [6], Ad-hoc On Demand Distance Vector (AODV) [7], Zone Routing Protocol (ZRP) [8], Link state routing [9] and Diffusion based routing [10]. In general, all these protocols can be classified as either *proactive* or *reactive*. Proactive schemes always keep an updated idea of the network topology, and maintain routes to all destinations, whereas reactive schemes find and maintain routes only on an as-needed basis.

From the application scenario, we discerned the following information that drives the design process:

- Average bit rate: fairly low, ~10-100 bps
- Density of nodes: About 0.1-1 node/cu. m.
- Mobility of nodes: Low mobility, most of the nodes are stationary, while some are slow-moving
- Periodicity of data: Significant portions of the traffic would be periodic in nature, such as sensor data
- Application Data Unit (ADU) sizes: These would be very small, ~ few bytes each
- Reliability of data delivery: Due to the redundancy of sensor data, 100% delivery guarantees are not always required
- Total number of nodes: 100s –1000s of nodes

Using these specifications, we can evaluate different addressing and routing methods.

3.2.1 Addressing schemes

Addressing of nodes is an issue where we depart from the traditional model. Trying to maintain unique IDs in an ad-hoc network would be expensive in both computation and communication. First, there is the problem of allocating the addresses uniquely. Secondly, with routes to mobile nodes continually changing, keeping track of these routes with regard to their IDs would be expensive.

Instead, we propose the use of *class-based addressing*. Class-based addresses are composed of one or more fields, and each address identifies a class of nodes. Specifically, the PicoRadio addresses consist of the triplet <*Location, NodeType, DataType*>. The Location contains the coordinates of the node, NodeType can be a sensor, monitor or actuator, and DataType can be temperature, humidity, etc. This matches the application requirements, since typical sensor queries are of the form: "temperature data from sensors in a region." Any node that satisfies the address requirements can answer such a query, and not having to map such requests to unique network IDs is cleaner, more scalable, and less costly for the network to handle.

3.2.2 Routing methodology

In a wireless network, data can be routed to the destination in either a single hop or multiple hops. Single hop requires transmitting with enough power to reach the destination, which not only causes interference at a large number of other nodes, but also wastes energy as explained in Section 2.3. Therefore, routing is done in multiple smaller hops to optimize energy.

Since the bit rate of the network is low, the routing protocol has to be lightweight to ensure that most of the energy is spent in moving data, rather than wasted in routing updates. We therefore concentrate on reactive routing protocols. The routing tables that are maintained at each node should also be small compared to the size of the network to avoid requiring large caches. At the same time, the protocol should be tolerant to nodes coming up or down, and thus fault tolerance and protection is an integral part of the design.

Two approaches are promising for the network layer of PicoRadio networks. First, the directed diffusion paradigm [10], which is a reactive scheme where the monitor broadcasts its *interest* in certain data, and the sensors that can provide such data reply back, setting up multiple paths to the monitor in the process. Due to the multiple paths created, the scheme assures data flow in the event of node failure. However, diffusion requires the use of fairly large caches at every intermediate hop. Secondly, geographical information can also be exploited to route in the correct direction towards the destination [11]. We are currently developing a protocol based on these schemes that would be optimized for PicoRadio and would maximize the survivability of such a network.

3.2.3 Energy vs. latency tradeoff

One interesting characteristic of the application is that the size of each ADU is small, so delivering such small packets individually is energy inefficient since a large percentage of energy is wasted on overhead. This overhead is in the form of network layer headers enlarging the packet and the MAC layer cost of allocating the channel. However, most of the data is delay insensitive, giving us the opportunity to trade-off energy with latency. We can buffer data at a node and then burst it in a combined packet, increasing the latency, but reducing the total energy consumed.

3.3 Media Access Layer (MAC) Design

The main principles guiding low power distributed MAC design are the following:

- Collisions should be avoided since retransmission leads to unnecessary energy consumption and possibly unbounded delays. At the same time, collision avoidance may result in substantial overhead, so a trade-off must be evaluated to achieve an optimal solution.
- Protocol overhead should be reduced as much as possible, including packets dedicated for network control and header bits for data packets.
- In typical wireless systems, the receiver has to be powered on at all times resulting in significant energy consumption. The more time the radio can be powered off, the greater the power savings.
- For a large-scale system with limited mobility, we need an adaptive algorithm to support mobility while avoiding unnecessary overhead.

In this section, we introduce a new class of MAC protocols to take advantage of opportunities provided by new radio technology and to provide a more flexible interface to higher-level protocols. We present a dynamic channel assignment method, a sleep and wake-up technique, a mobility adaptive MAC, and the results obtained for our MAC design.

3.3.1 Dynamic channel assignment

Among MAC protocols currently used, an important subset is the one that alleviate media access conflict by using multiple channels. A given bandwidth of wireless media can be divided into a number of channels in various domains in different schemes such as FDM, CDMA, OFDM or TDMA. Single channel and contention-based protocols attempt to achieve conflict resolution in the time domain, but suffer an instability problem as the network get denser and traffic increases. In PicoRadio, a multiple channel scheme based on dynamic channel assignment has been chosen, and each node is assigned a locally unique channel. We assume that a spread spectrum CDMA scheme will be used in our system, although an OFDM solution is also under consideration.

For a flexible and scalable mobile multi-hop network, a limited number (~30) of channels have to be assigned to an almost unlimited number (thousands) of nodes in a distributed and dynamic way. Thus, local uniqueness and global reuse is the key. The goal of our code assignment algorithm is to assign different codes for all neighbors of a given node. This channel assignment problem is equivalent with the two-hop coloring problem in graph theory, which is formulated as "Color the nodes of a graph such that any pair of nodes one or two hops apart have different colors." The color assignment is a NP complete problem, and we use a distributed heuristic solution. The nodes listen to a common control channel and periodically broadcast a channel assignment packet to their neighbors using the same channel. Every node keeps a channel assignment table to record the channel usage by its one-hop and two-hop neighbors, and makes sure its own channel is different from all its two-hop neighbors. Simulation shows that the algorithm converges quickly and is robust to topology change.

3.3.2 Sleep and wake-up scheme

It is observed that in a typical Smart Building scenario, where nodes spend only 1% of time doing data transmission, more than 90% of the energy is spent on channel monitoring when nothing is happening. This is because existing radio system designs use the same hardware to do data transmission and channel monitoring. By using a separate super-low-power radio (*wake-up radio*), we can power down the normal data radio when it is neither transmitting nor receiving any data, while at the same time the wakeup radio monitors a wake-up channel. If the node wishes to transmit a packet, it wakes its own radio, then sends a short *wake-up beacon* to the recipient using the wake-up radio channel. Upon reception of this beacon, the receiver's wake-up radio will trigger the power up the normal radio up to be ready for the reception.

Preliminary research shows that the wake-up receiver radio operation may only take around 1μW, compared to 10mW for a CDMA radio in monitoring mode. In our targeted traffic scenario, the energy for channel monitoring is almost negligible.

3.3.3 Mobility adaptive MAC

We integrate the multi-channel scheme and wake-up radio to create a low power distributed MAC protocol for mobile nodes in an ad-hoc radio network. The fact that nodes may be mobile means that channel allocation does not occur solely in a startup phase. It is necessary for nodes to periodically exchange channel assignment packets containing information on neighboring nodes. The problem is that once a node goes to sleep it can no longer keep track of the dynamic neighborhood information. However, in a sensor network such as the Smart Building scenario, only a small number of nodes in the network are mobile, and with limited speed. Thus, we propose an adaptive scheme with the notion of *mobility aware nodes*. We assume that each node is aware of its mobility, either by an embedded sensor or instruction from upper layer applications. These mobile nodes will periodically broadcast a beacon using the wakeup channel to keep its neighbors awake, thus maintaining a dynamic *active zone* within two hops. Static nodes in the active zone will remain awake until the beacon has not been received for a predefined period, then go back to sleep again. A mobile node can also go to sleep to save power, provided that it has no interest in its dynamic neighborhood information, and has no wish to be found in the near future.

3.3.4 Results

Simulation has been performed for a regular network of 100 nodes where each node has 6 neighbors. A pool of 32 channels is available to be assigned to the nodes using the coloring algorithm described above. Power performance is evaluated after the channel assignment algorithm converged and no further topology changed. Only data transmission related traffic is evaluated at this time. We compare the Energy Per useful Bit (EPB) of the proposed protocols under conditions when a wake-up radio is used and when a traditional signaling channel is used.

Simulation results show the proposed protocol provides a varying degree of power savings depending on the traffic. In typical PicoRadio applications, we achieved an EPB of 2.5~4.0μJ/bit for MAC payload data. The results show at low network traffic, the EPB of the new protocol is mainly a function of data packet length. EPB decreases, as the packets get longer. This is understandable since for longer packets, the fixed overhead signaling will take a smaller part of the overall energy consumption. In low traffic zones, the EPB of the proposed protocol is relatively stable, but the EPB of the traditional radio protocols will increase dramatically as traffic decrease. This proves the efficiency of our protocol for a low traffic distributed sensor network.

In Figure 1, the top curve represents a traditional radio, the bottom curve represents the proposed new protocol and the middle curve represents a fictional system where monitoring power consumption is lower than the existing system but higher than PicoRadio.

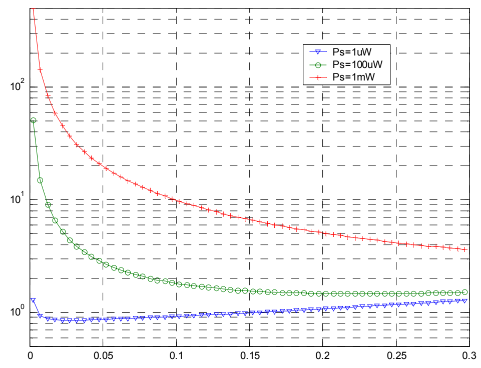

Figure 1: Energy per Bit as a function of traffic per node (from 50bits/second ~ 3000bits/second)

4. Designing Low-power Silicon Platforms

Crucial to the success of PicoRadio networks is the availability of small, lightweight, low-cost network elements, which we call PicoNodes. These nodes must use ultra-low power to eliminate frequent battery replacement. We envision a power-dissipation level below 100 microwatts, as this would enable self-powered nodes using energy extracted from the environment. These nodes must also be able to adapt themselves to changing conditions of the network, requirements of different data types to be transmitted, and the distance this data will travel. For instance, a PicoNode should compress more data that is going to travel farther distances, and just send plain data for short distance communications.

While technology advances have made it conceivable to build and deploy dense wireless networks of heterogeneous nodes, the design of a low-power, low-cost, adaptive PicoNode in a reduced time to market is still a challenge. Progress in wireless design development is limited by the lack of adequate design methodologies and tools.

The goal of this section is to address a design methodology to implement a silicon platform for PicoRadio networks. A design methodology for PicoNodes should address the mapping of the PicoNode functionality into an efficient silicon platform. Moreover, this platform should be flexible to enable dynamic adaptation for different working scenarios.

A wireless sensor is a complex system that implements a variety of functions: protocols, signal processing, position location, A/D converters, RF transceivers. Traditionally, wireless transceivers were almost completely implemented using RF and analog circuit modules. A mostly digital approach is currently coming into vogue. This trend is inspired by the observation that digital circuitry improves exponentially with the scaling of technology, while analog circuits get linearly worse, mostly due to reduction of the supply voltage. Hence, the PicoNode implementation relies on a small noncritical analog front-end and use digital back-end processing to correct for the nonidealities. Therefore in this section we mainly focus on the problem of designing the protocol and signal processing part of these sensors.

Protocol specifications are heterogeneous in the sense that they include both control and data processing functions, but significantly more control functions. Data processing, (e.g. in encryption, error correction) is typically applied at regular time intervals to streams of incoming data and is often subject to tight timing constraints. Control functions are of two types: real-time, if data processing is enabled by the occurrence of external (e.g. user request) or internal (e.g. timers) events, data-dependent, if some action is taken as a consequence of a data value (e.g. depending on the CRC result a packet is discarded).

Signal processing blocks, like protocols, are also heterogeneous including both data and control functions, but are heavily weighted towards the data processing functions. Data processing is typically in the form of datapath blocks were semi-infinite data streams are processed. Control processing is typically in the form of "steering" the data through datapath blocks, and less often in the form of real-time when responding to a request from the protocol.

While the layered heterogeneous protocol specifications are usually implemented by mapping them into a heterogeneous architecture including custom logic, programmable logic and an embedded processor, the signal processing part is mapped almost exclusively to custom logic.

Due to their differences in specification and target architecture, protocols are designed independently of the signal processing part of the wireless sensor. Section 2.2 presents a design methodology for protocols and Section 2.3 focuses on a design methodology for the signal processing part.

4.1 Protocol Processor Design Methodology

To meet the ambitious constraints in a short period of time, the design process must make efficient utilization of highly reusable architectures. To do so, we must look to a new approach to system design that facilitates design reuse. Our protocol processor design methodology is based on the concepts of platform-based design [12]. This methodology is demonstrated by describing its application in the early stage design of the next-generation PicoNode.

Platform-based design achieves design reuse by abstracting hardware to a higher level (platforms) that is visible to the application software. The system platform comprises a family of parameterizable architectures that adequately supports the functions in the application space. Upon identifying a system platform, the final system design involves programming a set of architectural modules using the programmer's model.

We have defined a three-phase approach to the design protocols. In **phase I**, the system platform is conceived through consideration of the application domain and available architectural modules. **Phase II** performs the design exploration to find a suitable set of architectural modules for specific applications and constraints. Lastly, **phase III** completes the final implementation (hardware and software synthesis) of a specific application onto the platform instance.

4.1.1 Platform Conception (Phase I)

A typical platform for wireless systems consists of programmable processors, reconfigurable logic, dedicated logic, memories, and peripherals. Constructing the system platform is a two-fold problem: We need to (1) identify the key functions and their constraints, (2) explore the available architecture modules and their performance behavior. The former is achieved through *functional profiling* of a suite of candidate applications. The latter requires *architecture exploration* of various means of implementation.

The concept of functional profiling is to gain an in-depth understanding of the application space by extracting a set of key operations (kernels) common to these applications. An efficient implementation could only be realized if the performance-critical operations are classified and specially targeted. The important issues in functional profiling are profiling granularity, classification, and interpretation of the collected data. If the granularity is either too coarse or too fine, regularity and commonality may not be fully exposed. It is often necessary to reorganize the application code (e.g. insert wrapper functions) to reach optimal granularity. Proper classification and interpretation of the profiling data require some insight into the class of application algorithms. Particular care should be paid to the separation of application code from the simulator kernel to ensure that only meaningful data is collected.

Architecture exploration has a couple of purposes. First, it identifies a family of computational and interconnect architectures that can efficiently support the key operations in the application domain. This may involve surveying existing architectures, or seeking new and creative ways to implement the functions. Failing to use the right architecture to implement a particular function can have detrimental effects on the performance of the resulting design. Second, it should provide first-order performance and cost models for these architectures. Having numerical models can greatly simplify and expedite the process of design exploration in the latter phases of the methodology. Lastly, it should provide software means for programming the functions onto the architectures. This makes up the application programming interface (API) for the system platform.

4.1.2 Platform Instantiation (Phase II)

In this phase we explore within the system platform to find a particular platform instance that is suitable for a specific set of applications. The classic Y-chart approach [13] is used: A platform instance is identified by choosing the appropriate parameterized architectural modules as well as interconnect model from the library obtained in phase I. The functional specification of the application is then mapped onto the platform instance. The quality of the resulting mapping, given by the performance and cost models, is evaluated under the given set of functional constraints. If the constraints are not met, the process is iterated by either instantiating a different platform or modifying the functional specification or both. By examining multiple platforms with varying performance, we can quickly converge to an efficient platform that is best suited for our applications.

4.1.3 Implementation (Phase III)

Once we have a platform instance that meets the design constraints, mapping any specific application onto hardware becomes a software issue. Through the software API, the hardware platform can be programmed or configured to perform the desired functionality. Remaining issues include generation and compilation of application code, real-time operating system (RTOS), and any necessary design synthesis.

By thinking at a platform level, we are able to produce a solution that allows fast design time through extensive software and hardware reuse. To effectively use platform-based design methodology, it is imperative to have a good system platform in place before proceeding further in the design flow.

4.1.4 Case Study: PicoRadio

In designing the PicoRadio protocol processor, we use the design methodology described in the previous section to devise an architecture that is optimized for size, cost, and most importantly, energy. This section presents our current efforts in the PicoRadio protocol processor design as a case study of the design methodology. The project is currently in phase I of the methodology, which consists of functional profiling and architecture exploration.

Functional Profiling

To study the PicoRadio application space, we have collected a benchmark suite with a wide range of mobile wireless applications. Preliminary results have been obtained from our first benchmark application, a mobile *ad hoc* network that supports different protocols. The application program is written in OpNet/Radio Modeler from Millennium 3 Technologies [14].

In this experiment, four different scenarios are studied with two different routing protocols and two types of node distributions. The first protocol is the Ad-hoc On-Demand Distance Vector Routing (AODV) [7], a reactive protocol. The second is the Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) [6], a proactive protocol. The first type of network distribution is a uniform grid of nodes in which the neighbors can hear each other. The second is a randomly generated distribution. The nodes have fewer neighbors on average in the random distribution case and generate less network traffic.

Opnet only produces profiling information at the level of the leaf funtions. The generated fine grained profiling data are grouped and classified. The results are presented in Figure 2. Searching consumes 20%-45% of the total time and packet manipulation (parsing, modification, re-assembly etc) consumes 18%-28% of the total time. Clearly this suggests that the implementation of dedicated packet processing and table searching engines may greatly improve the overall system performance. Furthermore, since searching is mostly routing and forwarding table lookups, a protocol that does not involve the maintenance of large tables should lead to a cheaper implementation.

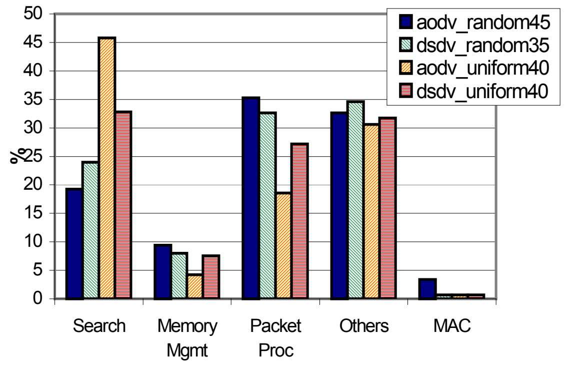

Figure 2: Profiling results of an ad hoc wireless network

Architecture Exploration

The enable hardware reuse and exploration of different protocols in various network scenarios, PicoRadio protocol processor must provide some degree of flexibility. While microprocessor provides the most flexibility, they fail to meet the tight energy requirements of wireless applications. In contrast, reconfigurable fabrics, such as field-programmable gate array (FPGA) and programmable logic device (PLD) are more promising options.

The traditional FPGA and PLD architectures differ significantly in the granularity. The FPGA comprises an array of thousands of configurable blocks, each implemented with lookup tables, which can each implement any arbitrary N-input logic (typically N<5). Since the lookup tables are easily chained together to implement more multilevel logic, this type of architecture is very well suited for complex operations such as arithmetic and signal processing. On the other hand, the PLD uses a small number of (typically <32) programmable array logic (PAL) blocks that can each implement sum-of-product logic of many inputs but limited output. This makes the PLD structure suitable for control FSMs. Figure 3 shows experimental results that support these claims.

For PicoRadio, the low-level (MAC and physical layers) protocols take the form of extended FSMs, which include both datapath and control elements. Consequently, we are constructing a hybrid, reconfigurable architecture, using PAL and LUT blocks for control and datapath respective. The architecture uses hybrid cells, each consisting of a small PAL block for control, and a small array of LUTs and flip-flops for data processing. Each hybrid cell is effectively a small protocol FSM; an array of these hybrid cells will be suitable for implementing complex, interacting FSMs. In order to keep the hybrid cells small, it is necessary to partition large FSMs. It will also be possible to perform power management on the hybrid cells to turn off inactive FSMs to conserve energy.

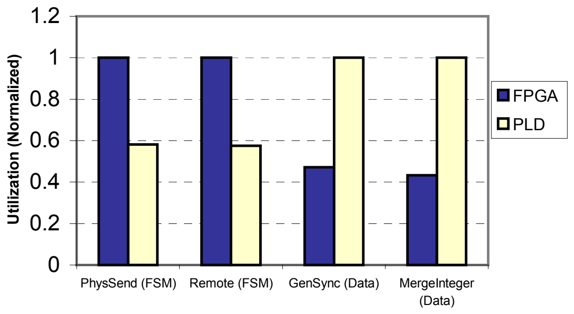

Figure 3: Implementation results of wireless protocol blocks.

4.2 Scaleable Low-energy Signal Processing Design Methodology

Scaleable signal processing systems can operate over a range of quality of service requirements trading energy efficiency for performance. Low energy and energy scaleable signal processing is a critical component of a PicoNode. First, scaleable signal processing algorithms must be designed at the conceptual level, and then implemented in the most energy efficient way to offer the most energy efficient solution for a wide range of operation modes. Scalability and energy efficiency cannot be implemented blindly; behavioral level designers must be able to evaluate the lower level implications of their decisions. We use a unified design environment for signal processing that allows all levels of the design to be completed within a single simulation environment for both datapath and control operations.

The unified design environment is based on in house tools in conjunction with the Simulink language from the MathWorks. The integrated design environment allows inter-level design decisions to be evaluated and implemented. The unified description environment makes resolving conflicts between levels straightforward and deterministic [17]. Analog or other peripheral blocks can be described in Simulink and can be incorporated with the structural level design description in a unified simulation. This allows the system level simulation to include the interaction of realistic phenomena such as analog and RF impairments with structural level decisions such as bit-width scaling.

Within our design environment, we use a four-phase design process: decisions are made at behavioral, functional, structural, and physical levels. Behavioral descriptions capture the algorithmic behavior. Functional descriptions capture the cycle-by-cycle behavior of the system, describing details such as parallelism and required throughput per unit. Structural descriptions include implementation details such as fixed-point types, pipeline depth, and adder and multiplier types. Decisions made at the physical level capture synthesis, placement and routing information.

4.2.1 Low-energy Direct-Mapped Data-path Design

Behavioral level decisions are based on algorithm exploration in Matlab or Simulink. Functional level design specifies Simulink implementations of the behavioral models. For instance, rotation of a complex vector through by an angle can be done with a complex multiply or a CORDIC. At the structural level, bit widths are determined, and micro-architectural decisions are made, such as types of adders and multipliers to be used, the use of carry save arithmetic, direct or transpose filter implementations, etc. Another type of structural optimization is in the use of multiple clock domains. The design flow recognizes the Simulink enable block as a gated clock specification and builds a clock tree automatically. Gated clocks are the basis for creating energy scaleable datapaths where unused blocks are shut off and hence do not consume energy.

The design flow assumes that each structural block corresponds to a unit of layout (called a "hard macro"), which has been extracted and characterized for power, area, and delay. The hard macros are created from soft macros, by fixing parameters such as bit width, adder type, and micro-architecture. Creation of a hard macro from a soft macro or a composition of macros is a process called "hardening" and is a heavily automated part of the design flow. Designers have the ability to create their own macros and integrate them into the flow through a well-defined process described below. Once the entire design is hardened, the chip is ready to be fabricated as all further steps (power routing, clock tree generation, LVS and DRC, etc) are automatically completed by the flow.

4.2.2 Control Design

Scaleable datapath blocks must be tightly integrated with control functions that manage the flow of data through the blocks, deciding the minimum amount of computation required and shutting off blocks that are not needed. It is thus desirable to implement both datapath and control in the same environment. Within the Simulink environment is Stateflow, a graphical language for describing control flow using state machines. An in-house tool translates Stateflow diagrams to VHDL, which can then be synthesized and included in the functional level description as a macro. Using the data-path and control primitives provided with Simulink, it is possible to create a discrete-time simulation with completely specifies the behavior of a system and therefore find the most energy efficient solution for a range of situations.

4.2.3 Module Generation

This design flow is based on intensive reuse of parameterized soft macros. When a macro is needed that doesn't already exist, there is a well-defined process for creating one and incorporating it into the flow. Since a macro corresponds to a piece of layout, several options are available for creating new macros. They can be composed of existing macros within Simulink, synthesized using a high-level description language, or built on tiled standard cells. The requirement is that the designer creates a functionally equivalent Simulink model to be used within the flow. Synopsys' Module Compiler is especially oriented towards creating large datapath modules allowing designers to easily explore different micro architectures for area, power and speed tradeoffs, and to create parameterized modules to expose these design choices to the module user. The use of Module Compiler modules is integrated into the flow such that parameters specified in Simulink are passed to Module Compiler and used to automatically create the desired hard macro.

4.2.4 Baseband Signal Processing Design Example

A 1.6Mbps DSSS Digital Baseband timing recovery unit (Figure 4) is used to demonstrate our flow. This system provides coherent timing recovery and code acquisition for a stream of soft symbols. The system receives a direct sequence spread spectrum (DSSS) QPSK modulated signal and recovers the intended bits. The spreading code consists of 31 chips per symbol. Chips are sent at a rate of 25MHz giving a symbol rate of 806.4KHz. The QPSK modulation sends two bits per symbol yielding a data rate of 1.6Mb/s. Analog signals from the front end are over sampled by a factor of 8 according to a free-running 200MHz (200MHz=8*25MHz) clock and converted to digital samples. The coarse-timing block performs code acquisition using a matched filter, and estimates the correct sample time within 3/8 chips (3/8*40ns=15ns). The fine timing block (using a synch word sent by the transmitter) estimates the frequency offset between the transmitter and receiver and refines the timing estimate to within 1/8 chips (1/8*40ns=5ns). The digital PLL correlates the data, corrects the phase offset, and tracks any residual frequency offset producing a coherent stream of soft symbols. These soft symbols can be sent to a simple slicer, or through a more elaborate decision device like a Viterbi decoder.

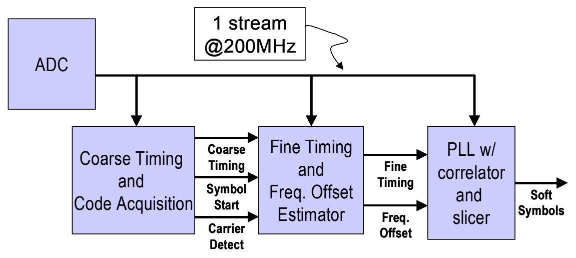
1 stream
@200MHz
ADC
Coarse Timing and Code Acquisition
Coarse Timing
Symbol Start
Carrier Detect
Fine Timing and Freq. Offset Estimator
Fine Timing
Freq. Offset
PLL w/ correlator and slicer
Soft Symbols

Figure 4 – Timing Recovery Block Diagram

As a concrete example in the use of the flow, we will discuss the progression of the fine timing block through the four levels of design. The inputs to the fine timing block are the oversampled data from the converter and the coarse-timing estimate. The outputs are the fine timing, and the frequency offset estimate. The frequency estimation algorithm is based on the data aided algorithm described in [18].

The most interesting functional level decision is in determining how to compute a composite operation of angle finding (determine the phase difference between symbol N and symbol N+1) and averaging (average phase difference between successive symbols over several symbol periods). It is determined through simulation and analysis that a complex multiply accumulate (CMAC) block can be used.

A large number of structural level decisions center on the CMAC. The input bit width and the accumulator bit width are parameters. In addition, our soft macro offers 4 micro-architectures to choose from shown in Figure 5. The characterization data provided by the flow shows that architecture 2 is the most energy efficient for the 6-bit input and a 16-bit accumulator length required in this design.

The fine timing block has three major sub-blocks that can be composed by the designer in two ways. Since all three of these blocks are implemented as Module Compiler macros, it is possible to compose them into one block within Module Compiler and then incorporate the composition into the flow in Simulink. Alternatively, the designer could compose the blocks in Simulink and use the flow to harden them into a composite block. Either choice produces nearly identical results as far as power, area and speed. However, by composing the blocks within Simulink, the

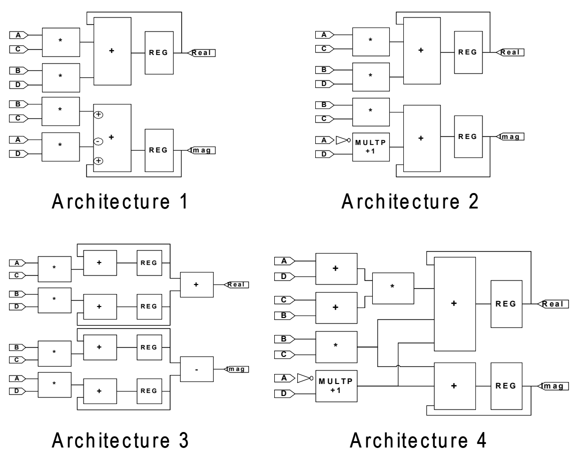

Figure 5: CMAC architectures

designer can specify different gated clocks to each block so that they could be shut off when not in use to save energy, while gated clocks cannot be implemented in Module Compiler. So, the most energy scaleable design is reached by composing the blocks in Simulink.

A module for the CMAC has been created for this design using Module compiler. Bit widths of the inputs and accumulator were parameterized and exposed to the module user, and four different micro-architectures were implemented. A parameterized, functionally equivalent model was created in Simulink for inclusion into the flow.

5. Conclusions

In our vision of distributed computing, thousands of tiny nodes scattered throughout the daily living environment gather, process, and communicate information in a self-organizing fashion. The major challenge in the implementation of these wireless ad-hoc sensor, monitor, and actuator networks is minimizing energy-consumption. We believe that the two key enablers for PicoRadio networks are the energy-conscious system-design and implementation methodology and a configurable architecture that permits these opportunities to be efficiently realized in silicon leading in the future to radio nodes that are two orders of magnitude more efficient than existing solutions.

A system conception and optimization methodology that allows efficient design exploration has been presented. Using this methodology, we have designed the protocol stack for the Smart Building application in UML with attached semantics. As a result we constrained each layer of the protocol. To further illustrate the methodology we presented the low-energy network and MAC layers for PicoRadio networks under development. Simulations have shown that the MAC layer proposed is 100 times more energy efficient than traditional ones. The network layer is combines the advantages of geographical routing and directed diffusion algorithms.

A protocol processor design methodology has been presented. It allows efficient creation of reusable silicon platforms. As part of this methodology, functional profiling and architecture exploration have been illustrated. Functional profiling has been applied to different network layer algorithms to identify the most costly operations. Architecture exploration has identified the best-suited type of fabrics for different blocks of a MAC and physical layer protocols. Based on these experiments, a new hybrid architecture is being designed.

A unified design environment for signal processing has been presented. It allows energy efficient and scaleable solutions to be realized and immediately reflects the impact of high-level decisions in the lower levels of the design. To illustrate this methodology, we presented a fine timing unit design. It took approximately one week to create the fully parameterized functionally equivalent soft macro blocks (correlator, complex MAC, and magnitude unit) in Simulink and Module Compiler. It took less than one day to compose them into a single Simulink simulation, and less than half a day to completely harden this subsystem including placement and routing. This block was implemented in a 0.25um process, had 37,000 transistors, an area of 0.22mm^2, a critical path delay of 5.2ns, and power consumption of 5.2mW.

References

[1] The official Bluetooth Website. http://www.bluetooth.com/

[2] A. Sangiovanni-Vincentelli, M. Sgroi, L. Lavagno. Formal Models for Embedded System Design. *IEEE Design & Test of Computers*, pp. 14-27, 2000.

[3] Cierto Virtual Component Codesign (VCC). Cadence. http://www.cadence.com/technology/hwsw/ciertovcc/

[4] A. Sangiovanni-Vincentelli, et. al. *Formal Models for Communication-based Design*, Proceedings of CONCUR '00, Aug. 2000.

[5] J. Rumbaugh, et. al., *The Unified Modeling Language Reference Manual.* Addison-Wesley, 1999.

[6] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing for Mobile Computers", Comp. Comm. Rev., pp. 234-244, Oct. 1994.

[7] C. E. Perkins and E. M. Royer, "Ad-hoc On-demand Distance Vector Routing", Proc. 2nd IEEE Wksp. Mobile Comp. Sys. and Apps., pp. 90-100, Feb. 1999.

[8] Z. J. Haas and M. R. Pearlman, "The Zone Routing Protocol (ZRP) for Ad-hoc Networks", Internet Draft, June 1999.

[9] P. Jacquet et. al., "Optimized Link State Routing Protocol", Internet Draft, Nov. 2000.

[10] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks", Proc. Mobicom 2000.

[11] Y. Ko and N. H. Vaidya, "Location-Aided Routing in Mobile Ad-hoc Networks", Proc. Mobicom 1998.

[12] A. Ferrari and A. Sangiovanni-Vincentelli, System Design: Traditional Concepts and New Paradigms, Proceedings of the 1999 Int. Conf. On Comp. Des., Austin, Oct. 1999.

[13] B. Kienhuis et al, *An Approach for Quantitative Analysis of Application-specific Dataflow Architectures*, Proceedings of Int. Conf. of Application-specific Systems, Architectures and Processors, pp. 338-349, Zurich, Switzerland 1997

[14] Opnet Radio Modeler, Millenium 3. http://www.mil3.com

[15] The MathWorks, Inc. http://www.mathworks.com/

[16] The SDL Forum Society. http://www.sdl-forum.org/

[17] W. R. Davis, et al. "A Design Environment for High Throughput, Low Power Dedicated Signal Processing Systems", submitted to CICC, San Diego, CA, May, 2001.

[18] H. Meyr, et. al., *Digital Communication Receivers*, John Wiley and Sons, New York, 1998.