Protocol Stacks for Power-Aware Wireless Microsensor Networks

by

Piyada Phanaphat

B.S. in Electrical Engineering and Computer Science,
Massachusetts Institute of Technology (2001)

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2002

Author ....
Electrical Engineering and Computer Science
May 24, 2002

Certified by ....
Anantha P. Chandrakasan
Associate Professor
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Professor of Electrical Engineering and Computer Science

Protocol Stacks for Power-Aware Wireless Microsensor Networks

by

Piyada Phanaphat

Submitted to the Department of Electrical Engineering and Computer Science on May 24, 2002, in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering and Computer Science

Abstract

In a distributed wireless sensor system, a need to prolong the lifetime of the network is crucial and limited by battery capacity. As communication traffic among sensor nodes is triggered by sensing events, the network can exploit these time-varying scenarios to obtain power savings by adjusting its operating conditions accordingly. A coherent design of application-specific network protocol stacks is the key. Specifically, embedding power-aware features in the link layer and media access control (MAC) layer promises to extend the lifetime of the sensor network.

The power-aware design will be illustrated on μAMPS sensor node prototypes. With the integrated design framework, lower layers of the network stack provides configurable power-aware features to be controlled by higher network layers that maintain broader-view knowledge of the environment.

TDMA has been chosen as a MAC Layer protocol for its inherited power-aware mechanism of radio shutdowns outside its TDMA slot and in absence of sensing events. Another level of power-aware features can be deployed in MAC ID and TDMA slot assignments. In a field of scattered sensor nodes, not all the nodes are in radio range of one another or of the base station. Hence, assigning N TDMA slots for the network of N sensor nodes that are not all in radio range will waste the receiver energy and link bandwidth. An algorithm for a re-use of MAC ID and MAC time slot is proposed based on the number of neighboring nodes. Hence, varying the number of neighboring nodes by varying the transmit power can optimize the system lifetime and bandwidth.

An implementation of the Link and MAC infrastructure is completed. Power scalability is illustrated on μAMPS node prototypes, with TDMA Media Access and a vehicle tracking application demonstration.

Thesis Supervisor: Anantha Chandrakasan
Title: Associate Professor, EECS

Acknowledgement

I would like to thank Professor Anantha Chandrakasan for offering me such a wonderful opportunity to be working in the μAMPS project for the past years. His advise and energy, one for which I am very grateful for, has always been inspiring and encouraging.

I would also like to extend my sincere gratitude to the people in the μAMPS project--to Nathan Ickes and Fred Lee for their expertise and knowledge on μAMPS nodes and their kind willingness to help out. Fred's music and positive attitudes help me survive some late nights in the lab. I also owe Rex Min for his mentoring and guidelines on how to approach problems and attack them by parts, Alice Wang for her constant encouragement and suggestions on beamforming application, Kevin Atkinson for the superb work on μAMPS implementation, and finally Ben Calhoun and Gap Thirathon for their contribution on the VHDL codes.

In my early stage of working on this thesis, many friends and co-workers also inspired me greatly. A big part of my thesis would not have been solved without Tengo Saengudomlert's wisdom. As a UROP in the group, I learned from Eugene Shih's supervision. Manish Bhardwaj's infinite energy and insights helped solve many unknowns.

My years in the lab would have been hard without others in the Anantha's group and their support: Seongwan Cho, Frank Honere, Travis Simpkins, Raul Blazquez Fernandez, Puneet Newaskar, Theodore Konstantakopulos, Paul-Peter Solateris and Margaret Falahy.

If there is any value in this work, I would like to dedicate it to my family. To Papy, whose ethic and philosophy I will always remember. To mamy, who planted in me an early love of reading and exploration. To A-Ma, who keeps me warm all these years with the knitted sweaters and gloves. To all E-E's whose supports mean the most. And certainly, to Bee Jae and Bua, who made me feel just wonderful.

Table of Contents

List of Figures

Chapter 1

Chapter 2

Chapter 3

Chapter 4

List of Tables

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 1

Introduction

1.1 Wireless Microsensor Networks

Wireless communication enables unprecedented network topologies whose deployment might have previously been restricted by wired infrastructure. One of the challenges wireless network designers face is the highly flexible, time varying nature of the network, thus there is increasing demand for more adaptive network protocol stacks. In developing protocol stacks for wireless networks, designers must focus not only on dynamic parameters, which may change in real-time and require instantaneous attention, but also focus on any unique characteristics of that particular class of wireless network. Parameters such as network lifetime, acceptable latency, node mobility, required signal bandwidth, etc. cannot be overlooked. This thesis presents the design of a protocol stack for wireless microsensor networks, a specific category of wireless networks.

Wireless microsensor networks [12][16] promise to be an efficient and low-cost alternative to traditional macrosensor systems. Enabled by advancing technologies in low-power sensors, analog and digital design, and RF radios, a network of small tens to hundreds of microsensors can be placed in a terrain to obtain a view of the environment. A microsensor network, unlike its predecessor macrosensor network, does not rely on few highly sensitive sensors to gather the data on environment dynamics. Instead, deploying hundreds of inexpensive and fault tolerant microsensor nodes can offer a high resolution, multi-dimensional view, without sacrificing economic cost or fault tolerance. System robustness is achieved partly from the number of nodes in a network; if one or more microsensor nodes cease to function, hundreds of others can still sense and relay the information to the base station.

A prime example of microsensor applications is remote sensing. In a field of microsensors equipped with acoustic sensors, each node is capable of monitoring changes in its proximity. Information from multiple sensors can bring about multidimensional knowledge of an acoustic source. For example, with a Line of Bearing estimation method, a network can triangulate the position of an acoustic source [7]. Together with other signal processing algorithms, a network may also yield information on number of sources, source velocity, or classification of sources.

1.2 Basic Layered Network

A standard OSI seven-layer network [15], as shown in Figure 1.1, represents network design abstractions. In real-world systems, not all the layers exist, and some networks may not demonstrate concrete separations between layers. As parameters differ from one class of network to the other, application-specific cross-layer protocol stacks may prove more efficient in terms of power, bandwidth, and delay optimization rather than traditional stacks [1].

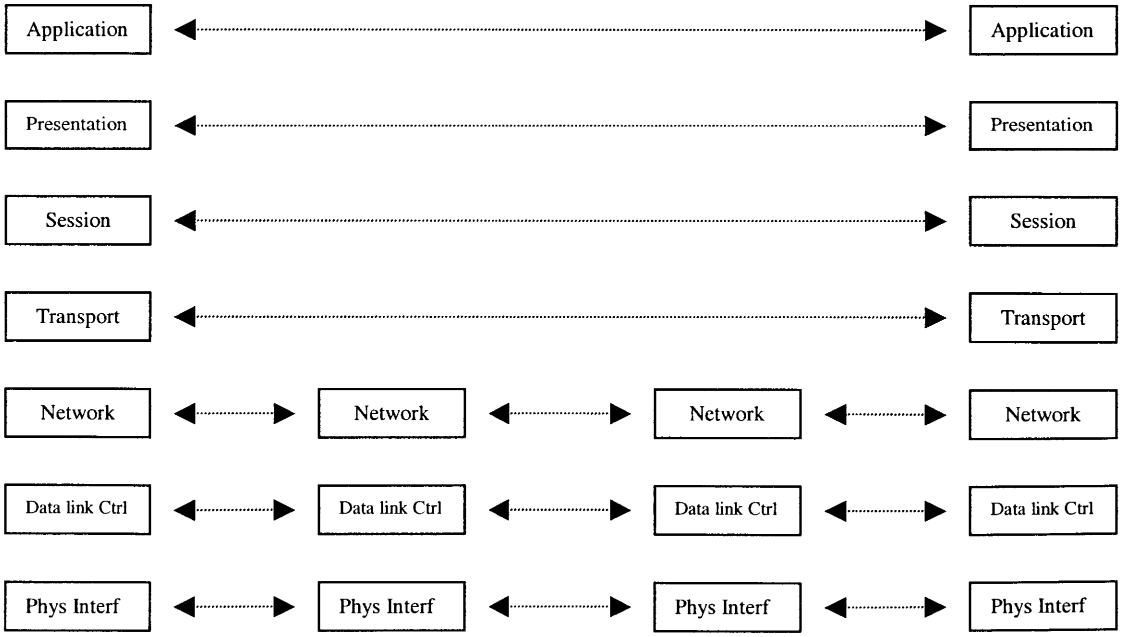


Figure 1.1: Seven-Layer OSI Network Layer

The scope of this thesis includes the design, implementation and performance analysis of the link layer and media access control layer for the example of a wireless sensor application. The following sections present basic concepts of each network layer relevant to my thesis work.

1.2.1 Data Link Control Layer

The data link control (DLC) layer is responsible for facilitating higher-level protocols to communicate over a physical channel. Specifically, a standard DLC should provide:

- Encoding/formatting mechanisms for the data received from higher layers before transmitting it through a physical channel;
- framing the data into a suitable size, with proper header, and sometimes a trailer,
- variable error detection, which may also include error correction mechanisms; and
- Media Access Control (MAC).

1.2.2 MAC Layer

The MAC is considered a lower sublayer of the conventional DLC. The responsibility of the MAC is to allocate the multi-access channel to all the nodes that share the same bandwidth so that communications between any nodes succeed without collisions. In choosing an appropriate MAC scheme for any network, specific characteristics of a network must be factored in. For example, although sensor nodes are deployed in a scattered fashion, the average density is predetermined. Therefor, the number of nodes sharing the same channel, which can be inferred from the average density of the network, certainly effects the MAC design. Also important is the fact that nodes are energy constrained. This suggests a MAC protocol that allows nodes to shutdown the radio receiver when not needed.

1.2.3 Network Layer

The network is concerned with routing data packets to their destinations. A wireless microsensor network is an example of an *ad-hoc* network. In one possible network protocol, nodes communicate over a wireless channel in a multihop fashion [12][13][16]. Other alternative routing protocols for wireless sensor networks may include direct transmission and cluster-head techniques [1].

1.3 μAMPS Node Introduction and Architectural Overview

The μAMPS (Micro-Adaptive Multi-domain Power-aware Sensors) project focuses on developing an application-specific microsensor system that embeds power-aware features in each subsystem. The current revision of the μAMPS node (μAMPS-1) uses commercial off-the-shelf (COTS) components. The μAMPS-1 node's hardware architecture incorporates four subsystems: sensing, processing, radio and baseband, and power supply subsystems as in Figure 1.2.

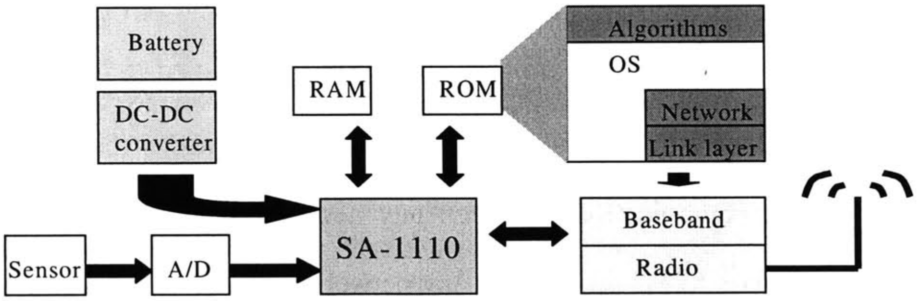


Figure 1.2: μAMPS-1 architectural subsystem

Aside from exploring various design alternatives for generic wireless microsensor networks, this thesis illustrates the subsystems that are related to the design of protocol stacks: the processing and the radio modules. The processor unit consists of an Intel StrongARM-1110 (SA1110) microprocessor, selected for its variable clock speeds and low power consumption. Incorporated also on the subsystem are RAM and ROM. RedHat eCos is selected as an operating system with its low-power and fully configurable features.

The radio and baseband subsystem consists of a National LMX3162 radio transceiver, operating in ISM band at 2.45 GHz [11]. The radio is half duplex, sharing one phase lock loop circuit, and offering a 1MHz bit rate. A Xilinx VirtexE XCV300E-FG256 FPGA is inserted onto the μAMPS-1 node to perform baseband operations, to manage the link interface between the radio and processing subsystems, and also to handle MAC-level implementations. Chapter 2 and 3 will present more details on the processing and the radio subsystems.

Figure 1.3 displays the current state-of-the-art of μAMPS-1 node. Each node consists of three stackable boards, including a radio and baseband board, a processor board, and a sensor and battery board. The base station board can replace the battery and sensor board so that the data can be transported onto a display or data storage, such as that of PDA or PC.

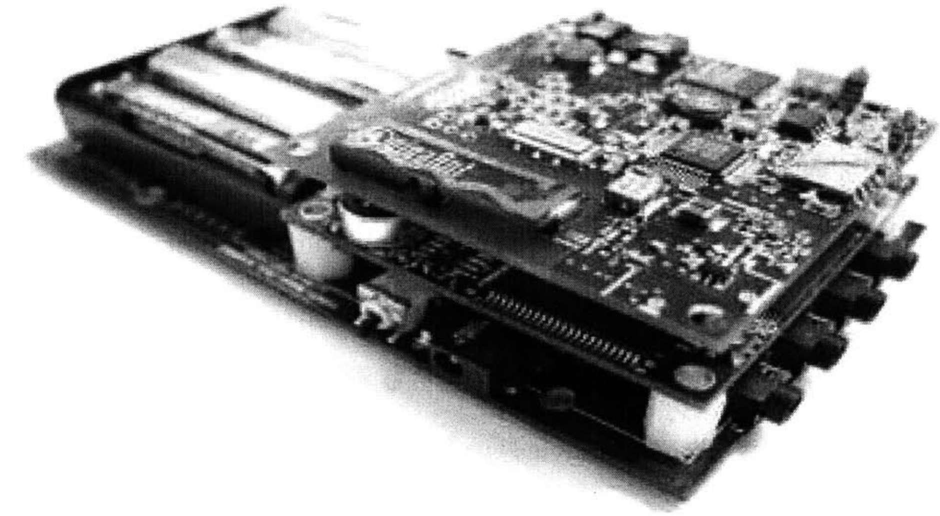


Figure 1.3: The current version of the stackable μAMPS node (Courtesy Fred S. Lee)

1.4 Related Work

Recent research in protocol stacks for power-aware microsensor networks has focused on the Link Layer, MAC protocols, and sensor network applications.

At the link layer, there are many protocols designed to optimize for various parameters such as output transmit power, energy efficiency, and error correction schemes [21][22][23]. A study and comparison in the cellular radio network [23] and wireless over ATM [22] on adaptive frame length and forward-error correction is done to optimize the power consumption for a given bit error rate. Similar ideas on adjusting power have also been explored in an the design of the 802.11b wireless LAN protocol [21].

Among different MAC protocols proposed for wireless networks [6] [4], Time Domain Multiple Access (TDMA) is the most widely deployed in sensor systems. The main challenge of incorporating TDMA onto sensor nodes lies in TDMA synchronization. In distributed *ad-hoc* networks, synchronization without a bus master [18] allows a flat hierarchy and makes the system more robust as it needs not rely on a specific master node. Claesson, Lonn, and Suri [9] proposed another novel bus-based approach that utilizes information about each node's unique message lengths to achieve low-cost synchronization. A hybrid method of TDMA and FDMA for *ad-hoc* networks [5][10][19] enables the system to operate without any global frame synchronization. An algorithm for MAC ID assignments for TDMA network is explored in [8].

Wireless sensor network applications range from security to biomedical applications. Environment monitoring can be used in security-related and surveillance applications [7]. The fact that microsensor nodes can be easily deployed makes a microsensor network ideal in remote terrains such as jungles or desserts, where direct frequent access by humans is limited.

Aside from the μAMPS project at MIT [20], other architectures and protocols for wireless microsensors have been developed. The Smart Dust [17] platform offers a small, low-power sensor node. WINS [16] and PicoRadio [13] present hardware architectures for sensor nodes quite similar to μAMPS.

1.5 Contribution and Outline of the Thesis

A major contribution of this thesis is on the integration of power-aware concepts across different network layers into a coherent, application-specific design. At the link layer, effort has been put into extending the capability of the link layer design from the previous version to include more power-aware functions. Highlights of the new design include increasing scalability and fault tolerance of the link and creating a configurable Link Manager Interface. The ease of adjusting transmit power levels, packet size, and encoding options from the higher level of the stacks are examples of innovations in this integrated design framework. At the MAC layer, a MAC algorithm that optimizes the number of TDMA slots in a TDMA frame based on the number of neighboring nodes within a radio range is explored. The algorithm leverages the configurability of the power amplifier and the knowledge of node density to achieve a desired number of nodes within a radio range. The algorithm proposes a slot re-use technique by mapping the slot to the MAC ID. Envisioned to support an application-specific network layer, the μAMPS MAC implementation, however, also is designed such that it can accommodate most routing algorithm's common features such as multicast communication. A vehicle tracking application is also demonstrated as a part of the work.

The rest of the thesis will be presented as follows: Chapter 2 explains the design and implementation of the link layer extended from the μAMPS-0. The chapter also presents various features of the Link Manager Interface that are implemented to support power-aware features of higher layers. Chapter 3 starts with a review of several MAC options. The chapter then leads to a MAC algorithm, featuring MAC ID re-use and TDMA slot mapping. The last part of the chapter offers details on a μAMPS MAC implementation. Chapter 4 analyzes the performances of the μAMPS network and discusses a specific application on vehicle tracking.

Chapter 2

Data Link Control Layer

The design of the Data Link Control (DLC) layer in μAMPS-1 is an extended revision of the DLC designed for the μAMPS-0 by Eugene Shih [2]. This chapter presents the design and architecture of different subunits of the DLC and its interface with the processing and the radio units. This chapter also presents power-aware design features, both as intrinsic characteristics of the link and as a setup control interface for higher layers to access the power-aware features of the radio.

2.1 Link Interface Design

The μAMPS-1 features a new Link Manager Interface (LMI) between the SA1110 and the radio subsystem. The interface provides the processor with convenient access for configuring and reconfiguring the radio via the processor. The LMI is implemented on a Xilinx FPGA. The software driver and Application Programming Interface (API) written for the SA1110 in C is provided for the users. More details regarding the API will be discussed in Section 3.3.3.

At one end of the LMI (denoted hereafter as the *link-to-driver interface*), the Xilinx is connected to the SA1110 via a 4-bit address bus, a 16-bit data bus, a chip select control signal, an output enable control signal, and a write enable control signal. The SA1110 and the Xilinx are connected to each other via a memory map technique; the microprocessor communicates to the Xilinx the same way it does to other memory peripherals. Hence, the Xilinx LMI is programmed to look like a bank of 16 addressable registers at the link-to-driver interface, each of which is 16 bits wide. The microprocessor may have write access, read access, or both, to these 16 *interface registers*. To ensure signal integrity, the LMI synchronizes all its I/O signals on this interface to a 10 MHz on-board oscillator.

One important signal at the link-to-driver interface is the interrupt request pin (IRQ). Connected to the external interrupt pin of the microprocessor, the IRQ is asserted when there is a received packet waiting at the link layer. Together with the interrupt request interface register, this IRQ allows the processor to pull a packet out of the DLC's receive FIFO once the packet arrives without having to poll for it. The detail of the data-passing mechanism from the link layer to a higher layer will be discussed in Section 2.3.

The other end of the LMI offers a link to the radio, or *link-to-physical interface*. As mentioned in Section 1.3, the LMX3162 is a single radio chip with its transmit and receive paths sharing one Phase Lock Loop (PLL) circuit. The link layer is responsible for ensuring that the PLL and the power controls of the TX path, RX path, and different levels of power amplifier (PA) are set correctly when the radio switches between transmit, receive and idle modes. However, the task of managing when each node should receive, transmit, or stay idle with respect to other nodes in the network will be left for the MAC sublayer (Chapter 3). In other words, the link layer sets up and oversees the mechanisms at the link-to-physical interface, with the commands from the MAC sublayer selecting the process and when to execute it based on the MAC layer's knowledge of other nodes, which also share the same channel.

At the link-to-physical interface, `ProgramRadio_data`, `ProgramRadio_clk`, and `ProgramRadio_load` signals from the Xilinx are connected to the LMX3162 via the MICROWIRE^{TM} interface. The MICROWIRE^{TM} is a serial interface that allows the baseband to configure the settings of the frequency synthesizer and power-down modes via the three registers within the LMX3162. Register N and R determine the tuning voltage of the PLL within the frequency synthesizer. Figure 2.1 displays the schematic of the LMX3162 frequency synthesizer [11].

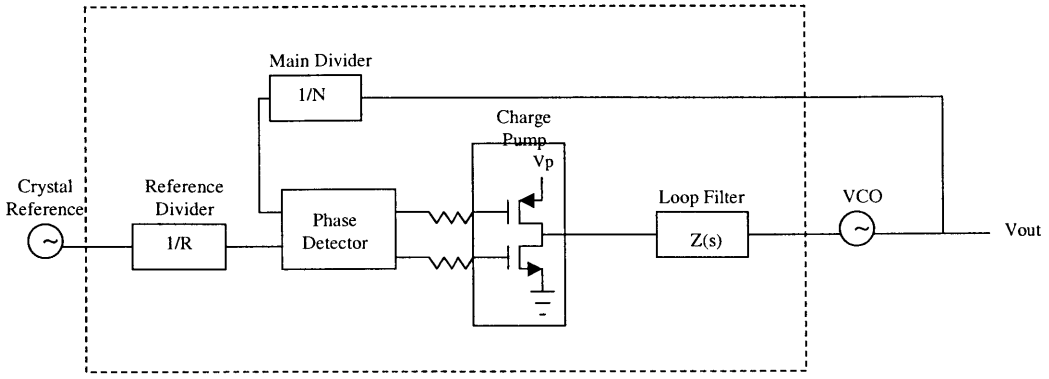


Figure 2.1: The Voltage Control Oscillator and PLL circuits of the LMX3162

Values in the R and N registers are used as frequency dividers for an input frequency and a feedback frequency. The phase detector circuit compares the output of the two dividers, and the phase difference results in an increase or decrease in input voltage to the VCO.

The F register is used for miscellaneous configurations such as setting the transmit or receive power mode, and tuning bias and gain of the demodulator circuit. The details on how these F, N, and R registers are programmed will be presented in Section 2.4.

2.2 Transmit Unit

The transmit unit of the baseband link layer architecture consists of a buffer (or a FIFO), two status registers, a Finite State Machine (FSM) controller, and a parallel to serial shift register. The FIFO, the two status registers, and the FIFO occupancy (with its empty and full flags) are mapped to four different interface registers of the Xilinx LMI, as seen by the microprocessor. Figure 2.2 presents the architecture of the link transmit unit.

A buffer is a crucial component in the link layer design of all network types, as it helps absorb the congestion over the communication channels. Most of the time, a network's bottleneck is in the bandwidth or data rate of the channel, since the maximum bit rate over the channel is generally fixed at a much lower rate than the processing ability. As for the μAMPS node, the SA1110 processing power is determined by a clock running at variable frequencies between 50 and 206 MHz. However, read or write interface with the Xilinx occurs at a slower speed, requiring 0.4-1.0 μsec per access. A processor can be expected to write 16 bits of data onto the link within one write cycle, resulting in approximately 0.045 to 0.065 μsec per bit. Even with this slow peripheral interface access, a transceiver with 1 Mbs bit-rate (1 μsec/bit) still exhibits the slowest data transfer rate in the transmit path, verifying the need for a buffer.

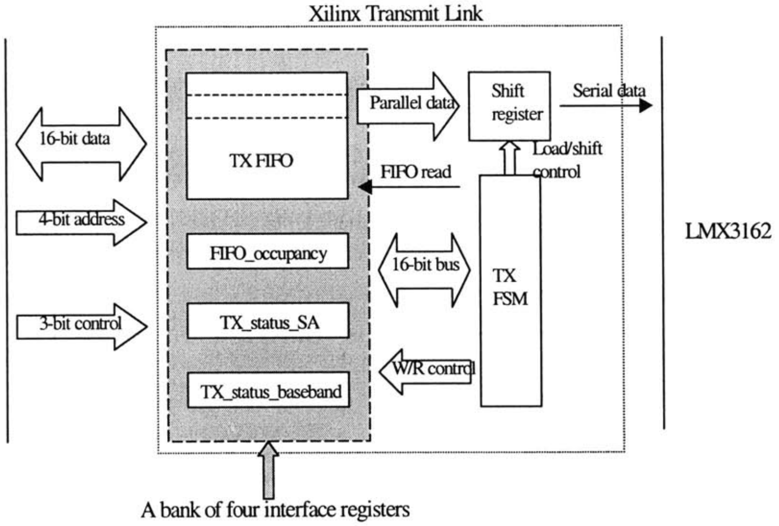


Figure 2.2: Transmit (hardware) unit of the link layer

Different parameters affect the decision on the size of the FIFO. Implemented using a Xilinx CORE generator, the FIFO size cannot be left as a variable and hence must be predetermined. Some of the criteria for choosing the size of the FIFO include: expected data rate, expected packet size, channel vs. processing capacity, memory usage, and the controller algorithm in queuing and dequeuing data at the FIFO. In general the FIFO should provide enough space for packets that are waiting to be transmitted. If a FIFO is too small and does not provide enough space for a packet waiting to be transmitted, the packet is lost. The FIFO size for μAMPS-1 is chosen to be 512 rows deep and 16 bits wide. Another issue of FIFO implementation concerns the mapping of an entire FIFO space to one addressable interface register. Care must be taken to assure that one write by the driver queues the buffer only once.

The FSM controller relies on the mechanism of a `TX_status_SA` register--a specific interface register to which the SA1110 driver has a write access--in starting the transmit mechanism in the link layer. Once a MAC layer gives an order to start a transmission process, the FSM controller checks the status of a packet arrival at the link by reading the `TX_status_SA` register. A driver, upon finishing queuing an entire packet on the FIFO, indicates the arrival of the packet at the link by writing to this `TX_status_SA` register. If there is a packet ready to be transmitted, the controller proceeds to send the packet to the link-to-physical interface and clear the `TX_status_SA` register. Note that a current occupancy of the FIFO is accessible to the SA driver via another interface register, and also internally to the FSM controller. However, the precision of the FIFO occupancy is coarse with a three-bit resolution. The FSM controller, therefore, cannot obtain precise knowledge of a packet arrival by only reading the FIFO occupancy. Note that the SA driver also checks the FIFO occupancy and waits for availability before starting to queue the packet.

For the data to be transmitted over the channel, the radio has to modulate it with the carrier frequency and amplify the signal power. The LMX3162 employs a direct Voltage Control Oscillator (VCO) modulation technique, offering lower power consumption relative to other techniques. The VCO directly modulates the baseband signal using a Gaussian Frequency Shift Keying (GFSK) modulation scheme and produces a positive or negative deviation from the centered PLL frequency for a digital ‘1’ or ‘0’ respectively.

Figure 2.3 illustrates the LMX3162 direct modulation VCO circuit, whose output signal's spectral density is shown in Figure 2.4.

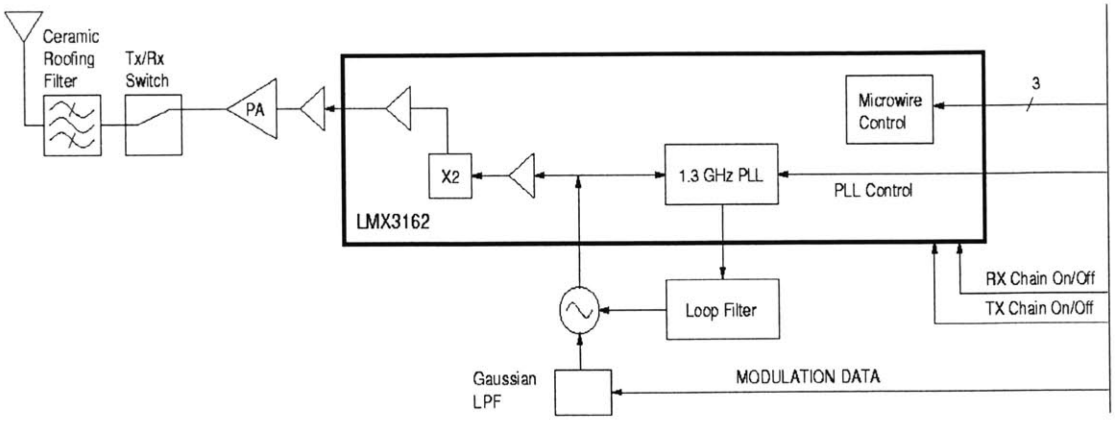


Figure 2.3: LMX3162 Transmitter with direct VCO modulation

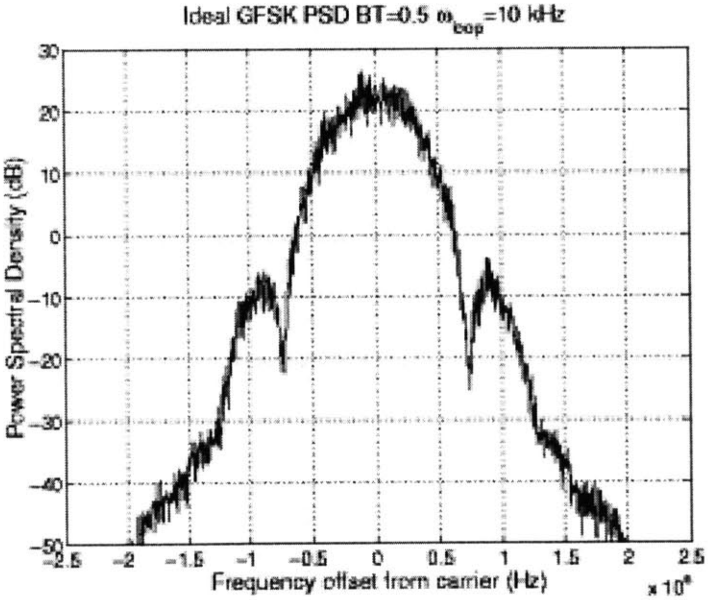
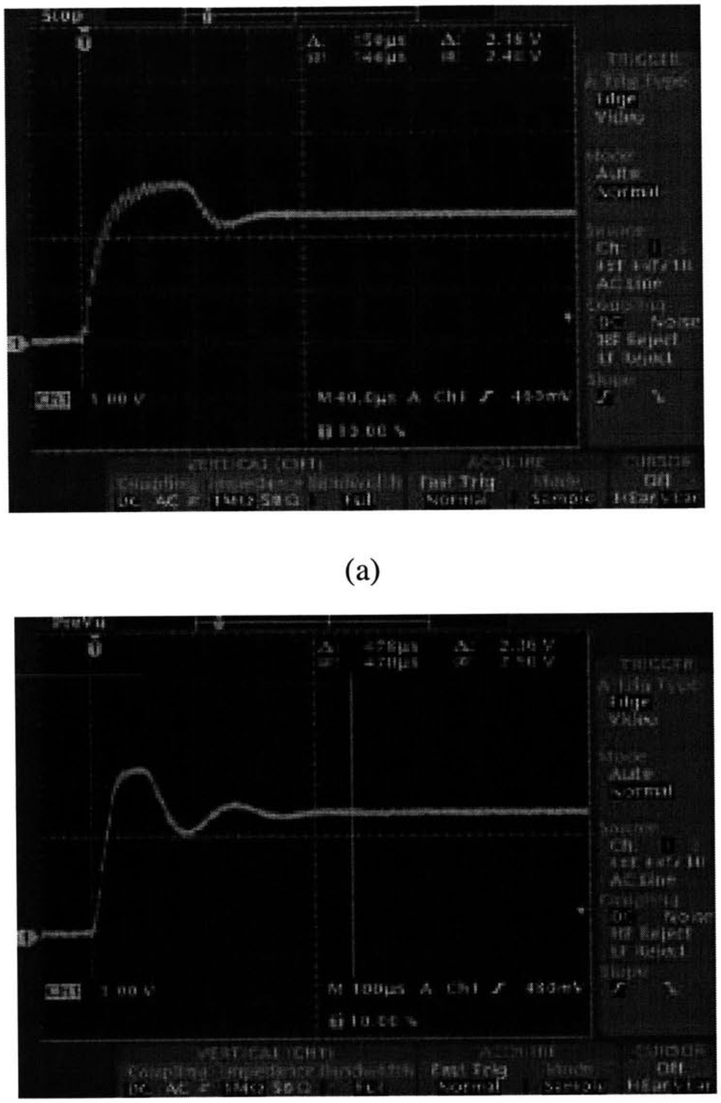


Figure 2.4: Simulated GFSK power spectral density of the transmitted signal [Courtesy Eugene Shih]

The PLL takes a fixed period of time to lock to the correct frequency, depending on a particular radio implementation. For the μAMPS-1 node, the lock time varies with different configurable settings of the charge pump via the LMI and MICROWIRE^{TM} interface. An approximate value of the lock time ranges from 164 to 470 μsec, as illustrated in Figure 2.5. The controller must be assured that the PLL is locked to the correct frequency before it starts to send the baseband data to the radio.



(a)

(b)

Figure 2.5: (a) the fast PLL lock time, and (b) the slow PLL lock time (Courtesy Fred S. Lee)

Once the PLL has settled on a desired frequency, the controller starts dequeuing the FIFO and passing the 16-bit parallel data to a parallel-to-serial shift register. The shift register is shifted at a rate of 1 MHz, sending the data to the radio circuit. Another interface register, called `TX_status_baseband`, allows the driver to monitor the status, as the controller writes different status codes onto this status register during the process of passing down the data from the link to the physical layer.

The power control unit, which is a part of the general control unit in Section 2.4, assigns an appropriate power control level to the PLL circuit, TX circuit, and power amplifier circuit. During the period that the data is transmitted, the PLL circuit can either be closed or open. The two alternatives differ in their power usage and constraints on transmitted data. In open loop mode, after the correct frequency has been settled on prior to the transmission, the loop is unlocked during the data transmission. This method conserves the PLL power since it will be turned off during the transmission time. The baseband data from the shift register directly modulates the VCO, and whose frequency may drift away from the center frequency initially set by the PLL when a long data stream is being transmitted. As a consequence, the open loop technique puts a constraint on the length of the data stream, with the exact number depending on how fast the frequency drifts and on the sensitivity of the receiver. The mentioned frequency drift may cause the performance of the receiver to incorrectly decode the bit to be either all ones or all zeros, depending on the direction of the drift.

The PLL in closed loop mode consumes additional power since it is left on during the transmission so that the PLL can continually adjust the frequency. Figure 2.6 displays the timing diagram of closed and open loop modes. The loop frequency (f_{loop}) or loop bandwidth of the PLL, however, imposes a constraint on sequences of bits being transmitted. A long sequence of ones or zeros will result in a positive or negative frequency deviation from the center frequency. The PLL is responsible for tuning the center frequency by lowering its input to the VCO in order to offset the frequency deviation caused by the long stream of ones or zeros. However, this action will take $1/f_{loop}$ which totals 0.05 - 0.1 milliseconds for a typical value of f_{loop} 10 - 20 kHz. If long sequences of ones and zeros can be avoided, closed loop mode has the advantage of unconstrained numbers of bits that can be transmitted. Some of the solutions to ensure that the data do not have long sequences of ones or zeros are the use of the Manchester encoding technique or the bit-stuffed technique. In the μAMPS node, the Manchester encoding performed by the driver is incorporated into of the link implementation. An alternative to the Manchester coding is also explored by Kevin Atkinson, another member of the μAMPS project. The run-length limited (RLL) block code is simulated and considered as a potential replacement. The main advantage of using the RLL code over the Manchester is that it can achieve approximately 80% of the original data rate and still guarantees no more than three ones or zeros in a row, while the Manchester cuts the rate by half. Note also that if the PLL power is left on without any bits being transmitted, the PLL will adjust its frequency to offset the drift caused by the direct modulation of an all zero sequence. Hence, a preamble data of alternate ones and zeros is recommended as training data for the PLL to adjust to the correct frequency before a real data packet is transmitted.

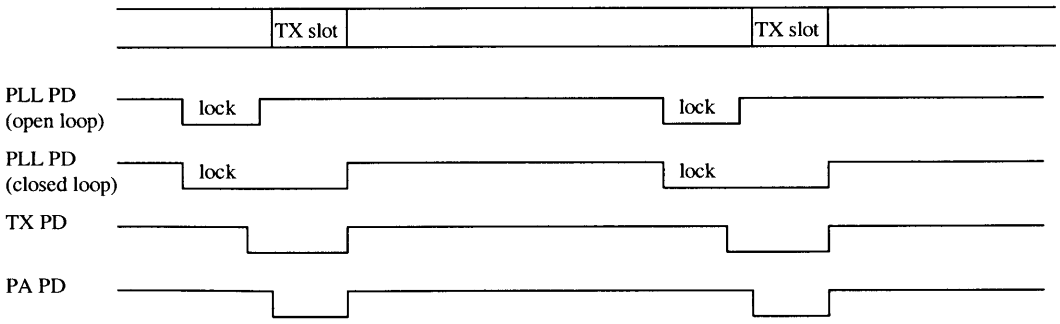


Figure 2.6: Timing diagram of PLL open loop mode and closed loop mode
The concerned signals are: PLL power down control (PLL PD), transmit power down control (TX PD), and power amplifier power down control (PA PD), all of which are active low.

The details regarding packetization will be best explained at the MAC layer, and hence will be postponed until the next chapter.

2.3 Receive Unit

The receive unit is implemented as a mirror image of the transmit unit, with some additional blocks for data recovery and signal decoding. The general diagram of the receive unit is shown in Figure 2.7. This section will discuss only the features that have not been discussed in the previous section on the transmit unit.

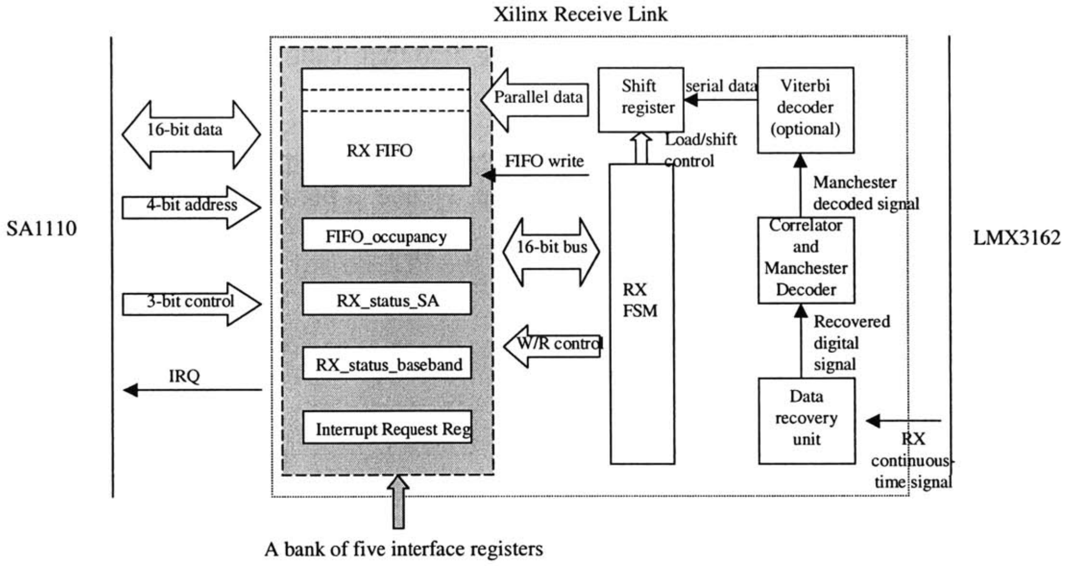


Figure 2.7: Receive (hardware) unit of the link layer

At the radio level, the receiver mixes down the received signal to an intermediate frequency (IF) of 110.6 MHz. After some filtering, the signal is demodulated using a threshold technique (declared one if the frequency is higher than 110.6 MHz and zero otherwise). The received demodulated waveform is further mixed down to baseband and eventually passed up to the link layer baseband.

Data recovery refers to a baseband process during which the received demodulated waveform is converted into a digital bit stream. Two alternatives fully explored in [2] include the clock recovery technique and the over-sampling technique. The former method attempts to recover the clock (frequency and phase) of the transmitter, which can be synthesized from the received data using PLL. This method then proceeds to recover the bits by sampling the analog received waveform with the recovered clock. The latter scheme bypasses the first step of recovering the clock and converts a received continuous-time signal into a discrete-time signal. In this oversampling technique, a received analog signal is sampled with a local clock. If a local clock is ideal, meaning there are no phase or frequency shifts, nor jitter, sampling the continuous-time data with a local clock of frequency equal to the data rate will sufficiently produce a correct discrete-time signal. However, in reality, clocks are subject to imperfection, and a local clock with a different frequency may lose or insert bits, depending on the sign of the difference in the two frequencies, during the data recovery process. As a result, a clock used to sample the waveform must be running at a frequency $f_l > R,$ when R is the data rate of the transmitted signal. The fast clock will guarantee that all the bits will be sampled. By oversampling the waveform, edge-detecting and comparing the oversampled sequence of ones or sequence of zeros with a threshold, a more accurate representation of a digital stream is obtained. The details of the oversampling technique and oversampling frequency selection can be found in [2].

The receiver must have a way to determine whether the recovered bit stream is a part of a transmitted packet. It achieves this goal by detecting a *SYNC,* a unique bit pattern that was appended at the beginning of each packet by the transmitter. The receiver recognizes the beginning of a packet when it sees the SYNC (using matched filter algorithm with programmable threshold value). To guarantee that a SYNC is unique, no other sequence of bits in the packet can have the same bit sequence. For Manchester encoded data, there exists no sequence of bits with three ones or three zeros in a row, so a bit pattern which includes either three ones or three zeros in a row can potentially be used as a SYNC. Adopted from the μAMPS version 0, a SYNC in μAMPS version I is chosen to be 0x74 or 0b01110100. For an eight-bit SYNC for Manchester encoded data, 0x74[1] has the highest auto-correlation and the lowest cross-correlation [24], and hence increases the chance of detection and decreases the probability of false-alarm.

A bit stream that is a part of a packet will be decoded if any encoding schemes are used. In μAMPS implementation, the data stream is Manchester encoded and hence subjected to being decoded at the receiver. Optional error correction coding can also be provided at the link layer to reduce the probability of errors of the received waveform. In communication theory, signal-to-noise power directly affects the error rate of the received signals. Intuitively, the higher the output power of the transmitter, there is a higher chance the receiver will receive a clean waveform. Error correction codes allow the transmitter to transmit at lower power and trade off lower bit rates. A Viterbi 1/2 code with constraint length of 5 was implemented in VHDL and simulated with the rest of the DLC subunits, but not ported onto the hardware due to limited resources in the Xilinx FPGA. The low bit error rate was observed over the channel.

An additional feature in the receiver unit is an interrupt request mechanism. This mechanism is responsible for notifying the processor upon arrival of a packet in the receive FIFO. Once an entire packet is queued at the receive FIFO, the FSM controller of the RX path reads the Interrupt Request Register, an interface register that stores the value of the number of current packets waiting in the FIFO. The FSM controller then increments

[1] 0x74 is a seven-bit SYNC, but 8 bits are allotted for the SYNC for convenience with the 16-bit bus interface.

the number of waiting packets and writes back to the Interrupt Request Register. The controller also asserts an Interrupt Request (IRQ) tristate output pin which is connected to the external interrupt pin of the SA1110. Configured to operate with the activated external interrupt, the microprocessor will automatically call an interrupt service routine (ISR) written for the external interrupt.

At the software level within this ISR, the processor must verify that the interrupt source is from the Xilinx in case there is more than one peripheral sharing this external interrupt pin. The software driver achieves this goal by reading the Interrupt Request Register in the Xilinx to learn the numbers of the packets waiting (zero means no packets, which corresponds to the fact that the interrupt source may be triggered by other peripherals). Any read to this interface register results in the register being automatically reset, and the IRQ pin de-asserted. The de-assertion of the IRQ pin only upon the notification from the driver provides a flexible interface that is compatible with either the edge trigger or level trigger modes of the microprocessor's interrupt features.

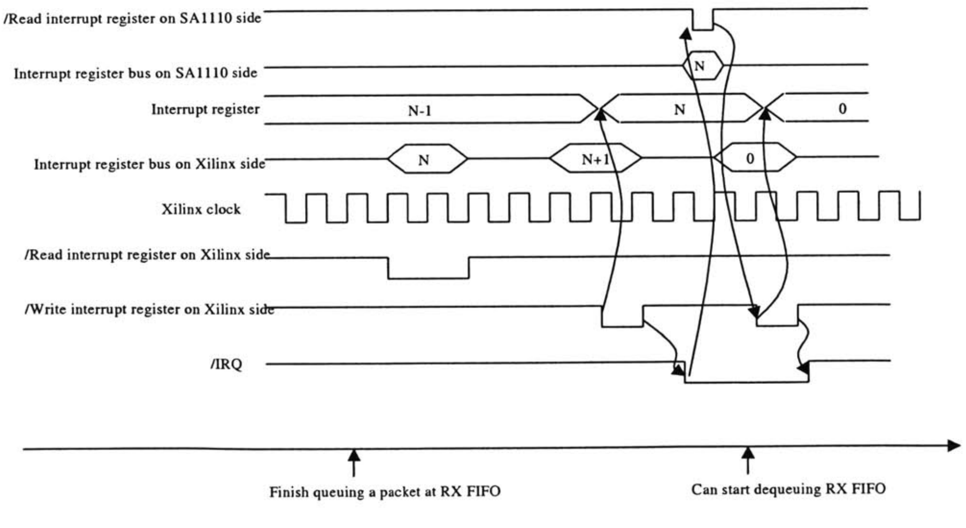

/Read interrupt register on SA1110 side
Interrupt register bus on SA1110 side
N
Interrupt register
N-1
N
0
Interrupt register bus on Xilinx side
N
N+1
0
Xilinx clock
/Read interrupt register on Xilinx side
/Write interrupt register on Xilinx side
/IRQ
Finish queuing a packet at RX FIFO
Can start dequeuing RX FIFO

Figure 2.8: Timing diagram of the interrupt handshake when a packet arrives at the RX FIFO.

A packet dequeuing can then be performed at the Delayed Service Routine (DSR)[1] in the software driver. How each packet is treated depends on the packet type. Chapter 3 discusses the packet format and the handling of synchronization packets. A data packet, after its correctness is verified, will be stored in a bigger memory buffer in the SA1110. Memory pool, a feature intrinsic to the RedHat eCos, is used to implement this buffer. Memory pool allows a specified size of the memories to be allocated. These memory blocks can be created and used to store values, and also be returned to the pool for future use after they are cleared. The memory buffer, designed to be a linked-list, provides a clean interface to the higher layer (e.g., application layer) as it will return the first packet in the list, if one exists, to the caller.

2.4 General Purpose Unit

The general purpose unit (GPU) extends the interface access of the Link Manager Interface. The GPU permits the processor's software driver to interface with additional miscellaneous link and MAC configurations. The current configurations within the GPU include the power control unit, the correlator threshold setting unit, the program radio unit, the MAC control unit, and other miscellaneous settings such as packet size setting. This section will introduce the architecture of this unit and details of some of the units related to the link layer.

Figure 2.9 shows the architecture of the general purpose unit, which consists of three interface registers, an FSM controller and a bank of internal registers. The unit was implemented with a goal of accommodating the need for additional interface registers, with the

[1] Since an ISR call has priority over normal instructions in the processor, the processor will stop its current task to service the interrupt request. To avoid monopolizing the processor, an ISR should be kept to a minimum size. Any complicated processing or memory access can be delayed. At the end of the ISR, the routine can opt to call a Delayed Service Routine (DSR) if there are more operations to be called, but they do not need the highest priority in getting the processor attention.

current number limited by the 4-bit address bus. Two of the three 16-bit interface registers, the *General Purpose Data* and the *General Purpose Address* registers, are set by the processor to indirectly pass desired values to the Xilinx. Upon a write to the General Purpose Data register, the Xilinx reads that value and proceeds according to the instruction code specified in the General Purpose Address register. The instruction in the General Purpose Address can trigger different actions to handle the data obtained from the General Purpose Data register. Because the FSM controller branches actions for each code, the GPU allows up to 2^{16} or approximately 64,000 different settings or configurations. The internal register bank is needed when instruction codes require the FSM to store intermediate data. The last interface register within the GPU, *General Purpose Status* register, is provided for the SA driver to monitor the status of the FSM controller.

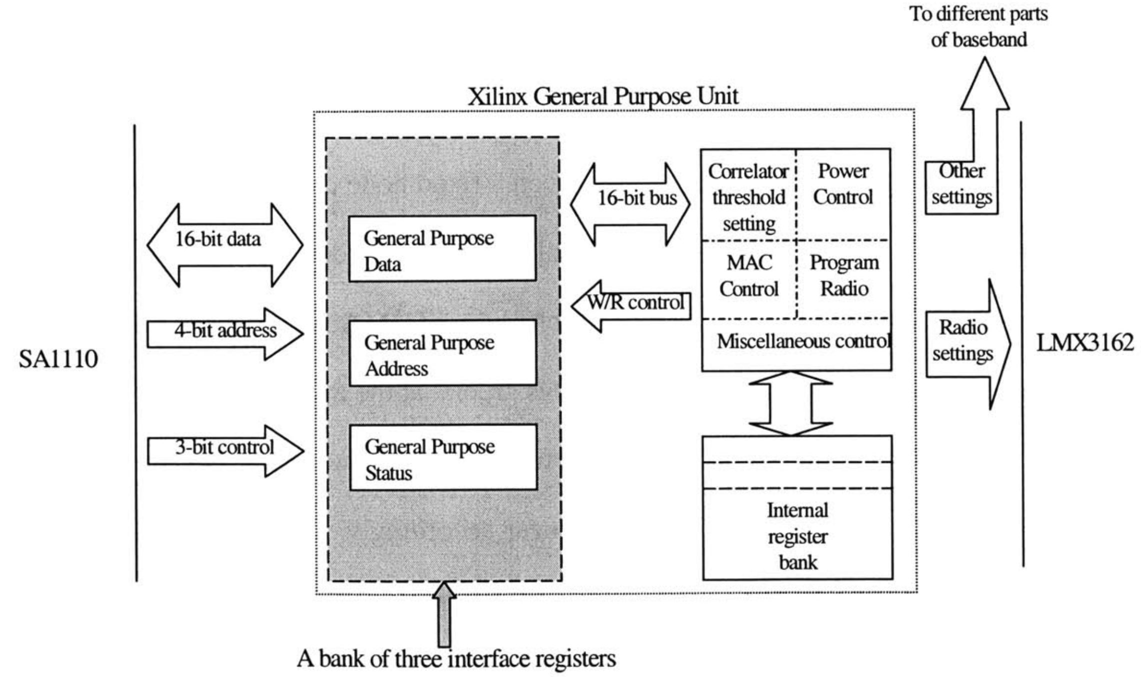


Figure 2.9: General Purpose Unit schematic

The power control unit within the GPU is responsible for controlling the power amplifier, LMX3162 related power modes, and power modes for the other radio circuits in addition to the LMX. Two registers in the internal register bank are set by the GPU for programming the radio. One register is used to program the power amplifier with 6-bit bias levels, ranging from 0 to 20 dBm. The other register is used to turn on the TX or RX path of the radio circuit. This latter register also controls the TX, RX, and PLL power down modes of the LMX3162 in case a hardware setting option is selected instead of a MICROWIRE^{TM} software option. This hardware option offers fast switching time of the LMX power-down compared to the slower MICROWIRE^{TM} serial interface. This feature will prove useful in the MAC layer control.

The ability to adjust power levels and power modes of the radio circuit at run-time is one of the μAMPS' design goals. This ability offers power-aware features which can be controlled at higher layers of the protocol stack. At the MAC layer, for example, the transmit power, which is controlled by the power amplifier, dictates how many neighboring nodes will be able to receive packets for a given a fixed node density. The MAC protocol can leverage this feature to obtain an optimal number of nodes that are within its radio range. Section 3.2 proposes an algorithm to schedule a TDMA frame based on this knowledge. In addition, the real-time power control is crucial at the application level. As operating environments vary greatly from time to time in sensor networks, a radio that offers power mode options can adjust its system power accordingly. During the sensing or processing-only period, the node can save power by shutting its radio off.

The correlator threshold setting unit offers a means to tune the correlator circuit in the receive path. A correlator detects the beginning of the packet by matching the SYNC pattern. Implemented in VHDL, the correlator is a matched-filter that slides a window of the known SYNC pattern along the received bit stream. At each point in time, the correlator

XNORs the sampled window pattern with the sampled recovered bit stream and sums up the result. A perfect match occurs when the sum is exactly the number of samples, meaning each XNOR operation returns a digital one. In reality, a received waveform may be distorted, and the sum will be less than the total number of samples. The correlator threshold setting unit allows the processor to adjust the sensitivity of the correlator. The correlator will declare a "match" when the sum is greater than the threshold. Depending on the signal quality and specific radio implementation, the best value for the threshold varies, but is limited to the highest value of 80 for an 8-bit SYNC with 10 samples per bit. The processor programs the threshold setting unit through the general purpose interface, and the correlator threshold setting unit in the Xilinx then programs the correlator via a 7-bit bus. The correlator performs well with the current threshold setting at 77.

The program radio unit sets up an infrastructure for the processor to configure the LMX3162 via MICROWIRE^{TM} interface, discussed earlier in the interface design. The MICROWIRE^{TM} interface requires a 20-bit-long serial data stream to be loaded to the F, N or R registers with the load enable on the rising edge of a `program_radio_clk`. The F, N and R values require two instructions via the 16-bit bus interface of the general purpose unit; the processor writes the low 10 bits and the high 10 bits separately with two separate instruction code set in the General Purpose Address register.

Chapter 3

Media Access Control (MAC) Layer

3.1 MAC Layer Protocol and Design Consideration

In wireless communication, four of the most widely deployed MAC protocols include Time Division Multiple Access (TDMA), Frequency Division Multiple Access (FDMA), Code Division Multiple Access (CDMA) and Carrier Sense Multiple Access and Collision Avoidance (CSMA/CA). The first three methods, generally known as fixed-assignment multiple methods, predetermine the bandwidth allocation for multiple users. CSMA/CA, on the other hand, is considered a random-access method. Table 3.1 gives examples of media access schemes in wireless voice and data applications [26][27][28][29][30]. Note that some of the application systems employ a hybrid MAC or single MAC with extra features such as frequency hopping (FH) with TDMA in Bluetooth, or carrier sense with pre-transmission control messages *request to send* and *clear to send* (CTS-RTS) (e.g., in 802.11b wireless LAN).

	GSM	AMPS	IS-95	CDMA2000 W-CDMA	802.11b	Bluetooth	HomeRF
TDMA		x				x (with FH)	x (with FH)
FDMA	x						
CDMA			x	x			
CSMA/CA					x		x

Table 3.1: Different MAC protocols and their applications

In choosing MAC layer protocol for our power-aware wireless microsensor network, the following features should be considered:

- simplicity and ease of deployment,

- fault tolerability and ability to adapt to dynamic topology,
- power-aware features and low power consumption, and
- compatibility with network and higher layers.

FDMA divides the bandwidth along the frequency for *N* multiple nodes such that they are each given *1/N* of the entire bandwidth. By assigning the channel's bandwidth through frequency division, all the nodes can transmit continuously at the same time without interference, except possibly at the boarder of two adjacent frequency bands. FDMA may not be a natural MAC for sensor networks because it consumes more power relative to TDMA; If an FDMA channel transmits a packet within T sec, a TDMA channel transmits a packet of the same size within approximately 1/T sec (plus additional overheads). Therefore, FDMA exhibits high energy consumed per packet. In CDMA, Pulse Code Modulation Method is used to convert a waveform between analog and digital representations. In transmitting bit streams, different 'codes' or mathematical basis functions allow the system to 'channelize' or divide the different bit streams (e.g., of two different transmit nodes) into different channels of the bandwidth. CDMA relies on complicated computations and can possibly put burden on simple processors of small sensor nodes. Since the μAMPS hardware prototypes are also not capable of carrier sensing, TDMA is an appropriate MAC option for μAMPS. TDMA provides a simple means for each node to avoid collision by reserving a channel in a fixed interval or slot. Generic TDMA (Figure 3.1) naturally has a power-saving and power-aware feature. The time division scheme allows the transmitter to transmit only in its time slot, which means the transmitter has to be powered up only 1/N of the time where N is the number of slots. TDMA is suited to the event-driven nature of sensor networks, as each radio receiver can also be powered down when detecting no data transmitted in the first 1/x of the timer slot period (x > 1) of each RX slot period. Moreover, generic TDMA supports multicast communication at the MAC level; a transmitting node needs to transmit only once when there are multiple targeted receivers (given they are in the transmitter's radio range) because all of the potential receivers can be scheduled to receive from the same node at the same time.

Although TDMA illustrates several desired qualities for wireless sensor networks, the protocol also demonstrates many drawbacks. In order to align the TDMA frames of all the nodes, a global frame synchronization must be performed. Studies on global synchronization [4], especially in a distributed fashion without a master node, confirm that the problem is non-trivial, especially if some of the nodes are not in radio range of each other. However, the problem can be simplified in sensor networks if a master node is allowed, such as in the setup of a clustering network [1]. The second drawback concerns the limitation of a fixed number of slots in generic TDMA. Potentially unused slots and hence bandwidth and power inefficiencies define the third drawback. Finally, there exists no standard algorithm in mapping sensor nodes, such as A, B, C, D and E in Figure 3.1 to slot numbers 0, 1, 2, 3, 4. In our illustration, Node A is assumed to be assigned an ID of 0, and hence allowed to transmit in slot 0. Nodes B, C, D and E are assigned to IDs 1, 2, 3, and 4 respectively.

To simplify the challenge of global synchronization, a network master can be assigned. A realistic and simple approach to selecting a master node might be to assign a base station node as a master node. In the μAMPS project, a base station node has the same hardware as other nodes, with only different software configuration running on the processor and a connection to a display or extra data storage (such as a laptop or palmtop device). A master node should be able to initiate a frame synchronization with all other nodes within its neighborhood. Once any node receives a frame synchronization notice, it aligns its frame with the master node, and also propagates the frame synchronization message to other nodes which are not in the master node's radio range. Eventually, all the nodes should be able to start every frame at an agreed upon point in time. Details of a frame synchronization implementation will be discussed in Section 3.3.3.

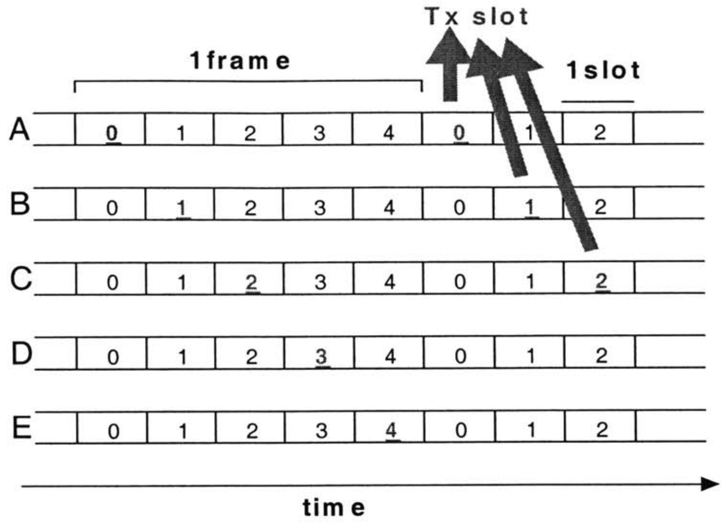


Figure 3.1: Generic TDMA frame.

In this illustration, one frame consists of 5 TDMA slots, 0,1,...,4. In slot 0, node A is allowed to transmit while other nodes are in receiving mode. In slot 1, node B is in turn allowed to transmit while other nodes are in receiving mode. And the logic holds for other slots. The frame cycles back to slot 0 after the end of the last slot (slot 4 in this illustration).

The remaining optimization of a generic TDMA is to increase efficiency in bandwidth and power usage by designing a slot/MAC ID re-use algorithm and defining an optimal numbers of slots needed for each TDMA frame. The design is proposed in the following section

3.2 MAC ID assignment and mapping of TDMA slots

Since each network has different number of nodes, the TDMA startup process must estimate how many slots will be needed in a frame. In a field of scattered microsensor nodes where not all the nodes can hear one another, pre-allotting a frame with one slot for every node is a waste of bandwidth and radio power when some slots are known not to be used.

An ideal scenario of MAC slot assignments is that the number of slots in one frame is exactly the same as the number of neighbors, which refers to the number of nodes within one's radio range (including itself since a node is also within its own radio range). However, this number is highly variable since a radio range of one node in the network may include more neighbors than a radio range of another node. Hence, the global optimal number of slots is defined as an upper-bound on the optimal number of slots determined at any node in the network. Given a network of sensor nodes, each denoted by μ_i when $i \in \{1, 2, \ldots, N\}$ and the number of neighbors of node μ_i is M_i, the optimal number of slots $S_{opt,i}$ determined for μ_i is equal to its number of neighbors M_i. The global optimal number of slots $S_{opt,global}$ is therefore $\underset{i \in [1, N]}{max} (S_{\mathrm{opt,i}})$.

Although an ideal number of slots $S_{opt,global}$ can be derived for a network with a known maximum number of neighbors M_{max}, the actual assignment of slots to nodes may not always be accomplished with the optimal number of slots. Only in an ideal case, such as in a network with an array of regular-hexagonal-grid nodes, the number of TDMA slot can be kept optimally at $M_{max} = 7$ as illustrated in Figure 3.2. Although this result is congruent with the cellular ratio system where a hexagonal cell is chosen for frequency reuse, it is not practical for sensor networks with non-deterministic sensor location.

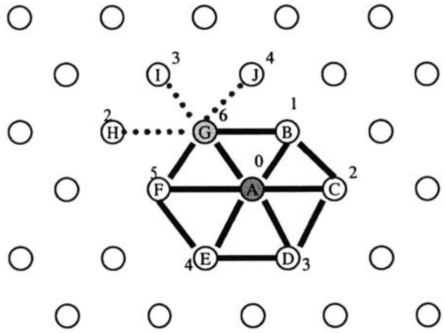


Figure 3.2: Regular-hexagonal grid topology

Assume the radio distance is equal to the length of the edge of the hexagon. The MAC ID 0-6 have been assigned to node A-G respectively, with number of neighbors M_A at node A being 7. Reassigning the IDs 2, 3 and 4 to neighbors of node G can be done as shown.

When the topology is not a regular-hexagonal grid, the $S_{opt,global}$ is not guaranteed to equal M_{max}. The topology in Figure 3.3 exhibits the same M_{max} as the network in Figure 3.2. However only seven MAC IDs, or equivalently seven slots if each node with a MAC ID i_i is mapped to a TDMA slot T_i, are not sufficient in this topology.

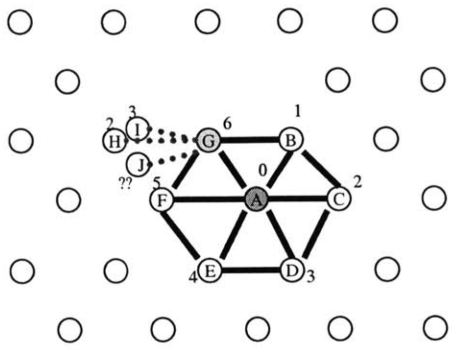


Figure 3.3: Another kind of network topology

The difference in the two topologies, which is the limiting factor in the MAC ID or a TDMA slot reassignment, is the number of neighbors that are two hops away. Observe that in the former topology, each of the re-used MAC IDs do not have any of their two-hop-away neighbors that own the similar ID. However, in the latter topology, the MAC ID 4 cannot be reassigned to node J, which has node E within a distance of two hops. In terms of the TDMA slot, a reassignment of slot 4 to node J will cause both node J and E to transmit during slot 4 of the TDMA frame, and a collision will occur at node F if it is an intended recipient.

The conclusion is as follows: knowledge of the MAC ID owned by every neighbor within two radio hops is sufficient for a reassignment decision. In other words, in assigning node μ_i a MAC ID σ, interference is guaranteed not to occur if there is no single node in μ_i's two-hop neighborhood with the MAC ID σ. However, note that the nodes within μ_i's two-hop neighborhood are permitted have the same MAC ID among themselves, as long as those nodes are not within two hops of each other. In the worse case, all the nodes within μ_i's two-hop neighborhood are also within two-hop neighborhood of one another, forcing a number of MAC IDs to be the number of the two-hop neighbors. From this analysis, the sufficient number of slots needed for a network with the number of single-hop neighbors $M_{max,1hop}$ and the number of two-hop neighbors $M_{max,2hop}$ is bounded by:

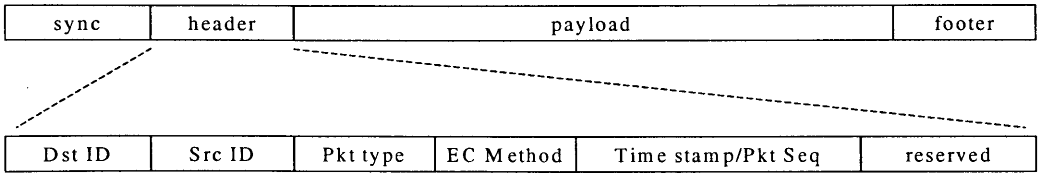
$$M_{max,\,1hop} \le S_{opt,\,global} \le M_{max,\,2hop}\,. \qquad (3.1)$$

3.3 MAC Implementation and API

The proposed MAC algorithm requires configurable MAC ID and number of time slots. The MAC implementation of the μAMPS node, therefore, focuses on an efficient structure and reconfigurable interface. The implementation offers options in deploying either the proposed algorithm or other algorithms.

3.3.1 Packet Format

As described in the Section 1.2, the link layer is responsible for framing data into suitable size, called a packet, with proper header. Figure 3.6 represents a packet format design. In general a packet consists of a header, a footer and a payload or a message that will be delivered to the next layer of the stacks, possibly after being decoded if any encoding scheme is used. Each layer of the stacks, such as MAC, network, or transport layer may have its own packetization, with a payload being the entire data sent from a higher layer including that higher layer's header and footer. A Checksum can be included to verify the correctness of each packet.



Note: The width is not to scale

Figure 3.4: Packet format and MAC header

For the μAMPS network, a MAC layer packet header is designed to have six separate fields. Destination ID and Source ID slots are allocated eight bits each, allowing a maximum of 2^8-1 different MAC IDs (although only a maximum of 16 IDs are being used with the current μAMPS node's hardware implementation of maximum 16 slots per frame). A MAC ID of 0xF is dedicated as a multicast ID, meaning a packet sent with a destination ID 0xF is intended for all the nodes in the network. The next field is a four-bit Packet Type field. Possible packet types include data and control (syn, ACK, NACK etc.). The next two bits are Encoding Method (e.g., none, viterbi 1/2 code, etc.). The next ten bits are allotted for time stamps or packet sequence number. And finally, the extra 16 bits are reserved for future use. Note that an additional Length field is not necessary for a fixed-size packet design in μAMPS implementation. Table 3.2 summarizes the packet header design, which totals 6 uncoded bytes overhead.

Field	Length (bits)	Options
Destination ID	8	0-15
Source ID	8	0-15
Packet Type	4	0 syn 1 data 2 NACK (not currently used)
Encoding Method	2	0 Uncoded (pure Manchester) 1 Viterbi half code
Time stamp/Sequence No.	10	0-1027 (not currently used)
Reserve	16	future use

Table 3.2: Packet Header Format

A 16-bit checksum of an entire packet is appended at the end as an error checking mechanism before the packet gets encoded (e.g., Viterbi and Manchester). Upon receiving any packet, a receiver recomputes the checksum and compares it to the one appended at the end of the packet. Any error detected in the receiver will be forwarded to the next level where the decision on how to handle a corrupted packet will be made. The current μAMPS implementation simply discards corrupted packets, but other alternatives include requesting for a resend (by sending a NACK back to the source) or repeating the content of the previous packet. For future consideration, more comprehensive checksums such as a two-dimensional checksum or a cyclical redundancy check (CRC) can be used.

After the final packetizing process in the stacks (in case there are several layers of packetization), the unique bit pattern or a SYNC will be appended in the front of each packet. A receiver detects the beginning of a packet when it sees the SYNC as earlier discussed.

3.3.2 Hardware

3.3.2.1 MAC registers and control: The hardware portion of the μAMPS MAC layer is implemented on a Xilinx FPGA, embedded in the LMI's General Purpose Unit. As described in Chapter 2, the FPGA interfaces with both the processor and the radio unit. The task of the Xilinx in the MAC layer is to assign the radio transmitter and receiver slots according to the MAC schedule specified by the software MAC driver (Section 3.3.3). In the link layer's link-to-physical interface via the MICROWIRE^{TM}, the LMI controls the power mode of the transmitter, receiver, and the PLL in real time, and also the frequency of the transmitter and recover.

Five internal 16-bit registers in the Xilinx GPU are set up to store 5 TDMA parameters: slot length in the `slotlen` register, number of slots/frame in the `numslot` register, TX slots in the `txslot` register, RX slots in the `rxslot` register, and MAC ID in the `macid` register. The GPU refers to these parameters when programming the radio. The processor passes these five parameters to the Xilinx through the three general purpose registers described in Chapter 2.

Out of the five mentioned internal registers, two are 16-bit bitmasked registers, `txslot` register and `rxslot` register. The values in these two registers determine which of the 16 slots in the TDMA frame are scheduled to be TX or RX slots. For example, if a value 0b11 or 0x3 is programmed to a `txslot` register, the radio is scheduled to operate in a TX mode in slot 0 and slot 1. With the same logic, the radio with 0b10100 or 0x14 in its `rxslot` register is scheduled to operate in an RX mode in slots 2 and 4. Section 3.5 illustrates an example of TDMA slot assignment.

Internal Register	Value in binary	Description
rxslot	0000000000010100	Slot 2, 4 are RX slots
txslot	0000000000000011	Slot 0, 1 are TX slots
numslot	0000000000000110	There are 6 slots in one frame
slotlen	1111110111101000	Each slot 13 msec (slotlen * 2/10 sec)

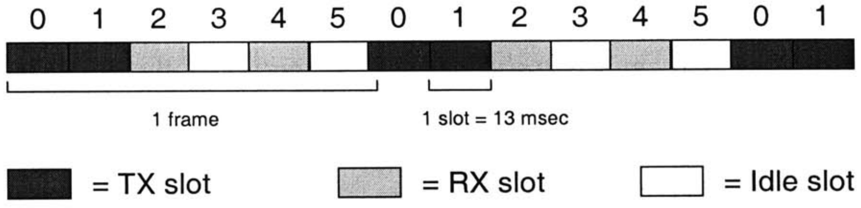


Figure 3.5: Example of TDMA settings in Xilinx internal registers

Once a TDMA mode is enabled, a counter in the FPGA starts counting down from the value derived from the parameter stored in the `slotlen` register. Counter expiration (at zero) marks the end of the slot, and the counter is automatically reset for the next slot. At the beginning of each slot, the radio will be powered up and configured for RX mode if that slot is scheduled to be an RX slot. The same is true for a TX slot, except the radio will be powered up during the TX slot only if there is data waiting to be transmitted in the TX FIFO buffer; the power control unit preserves the transmit power by shutting the radio off if there is no data to be transmitted. With a similar mechanism, the receiver can be programmed via the power control unit to be powered off when it does not detect any packets during RX slots within the first x μsec of the slot length (excluding the phase lock time), where x is set at 1,200 in the current implementation.

Another internal registers in the GPU related to the MAC is the `reset_frame` register. This internal register controls the frame synchronization mechanism in the MAC hardware. The `reset_frame` register keeps a twelve-bit value of a countdown counter *C*, and a four-bit value of the next slot ID after reset *I*. After a write to the `reset_frame` register, the countdown counter *C* will start counting. Once the counter reaches zero, the MAC FSM will jump to the next slot and reset the value of that slot to equal *I*.

3.3.2.2 Frequency Setting Registers and control: The radio is designed to filter the received signal at an intermediate frequency (IF) of 110.6 MHz [3]. In order to achieve that, the frequency of the transmitted carrier must be 110.6 MHz higher than the frequency of the demodulator in the receiver. This offset is strictly one way; if the receiver's frequency is higher than that of the transmitter, the received bits will be negated in amplitude.

The Xilinx is designed to automatically program the transmit and receive frequencies for the VCO of the radio's PLL as discussed in Chapter 2. The input voltage to the VCO is determined by the two programmable registers in the LMX3162: N and R registers. The F register programs the power down mode. Two of the 20 bits in the F register select whether the power down order will be taken from software or hardwire. Hence, different sets of F, N, R values correspond to different frequencies and power settings. Four transmit-receive frequency pair settings, or four pairs of F, N and R values, can be stored in the Xilinx hardware internal registers via the three interface registers in the GPU. The corresponding software driver sets these frequency pairs and selects which pair to be used at each point in time. By dynamically switching from one pair of settings to the other, the μAMPS MAC layer can hop among different frequencies. Frequency hopping is one of the important supporting features for many MAC protocols.

Table 3.3 summarizes different MAC features to be programmed via the Xilinx GPU

Setting	Code for GPU Address
TDMA slot length	1000
TDMA number of slots/frame	1001
TX slot bit mask	1010
RX slot bit mask	1011
MAC ID	1100
TDMA reset	1111
Set FNR array (lower 10 bits)	0101
Set FNR array (higher ten bits)	0110
Select FNR (for both TX and RX settings)	0111

Table 3.3: The summary of values to be programmed by the MAC driver

3.3.2.3 Signal Detection and Waveform Integrity control: Several extra features are added to the generic TDMA to guarantee the correct detection of packets and to minimize unnecessary radio power dissipation. At the beginning of the TX or RX packet, guard time is provided before the transmitter sends out a packet in the TX slot or before the receiver starts detecting the beginning of a packet. The guard time is crucial in two aspects: first, the frequency programmed to the PLL of the radio requires approximately 150-500 microseconds to lock to the correct value. Second, guard time, together with a gap time at the end of the TX slot prior to the next slot, help relax the need for accurate frame synchronizations (Section 3.3.3). Absorbing more frame shifts due to variation among oscillator's frequencies, this extra guard and gap time demands less frequent resynchronizations. During the guard time of the transmitter, it sends a training sequence preamble of 101010 instead of all zeros to prevent the drift in the receiver's PLL. The second crucial feature that enhances the packet detection is a received signal strength indicator (RSSI). The RSSI hardware outputs a digital one (by comparing an analog RSSI to a threshold) when it detects a strong enough carrier signal that matches the receiver's frequency (when the difference between the two frequencies fall into the IF frequency band centered at 110.6 MHz). The correlator circuitry, which detects the SYNC at the beginning of a packet, has RSSI as an enable pin. Measurements of the μAMPS node shows that the transition of RSSI may create an overshoot and hence may invalidate the correctness of the signal strength.

3.3.3 Software Application Programming Interface (API)

The API serves as a driver interface between the end users (e.g., an application layer in the processing unit) and the LMI and MAC hardware. The driver is implemented in C based on the μAMPS' eCos operating system and is compiled using a publicly available GNU toolchain. The toolchain is configured as a cross-compiler, meaning dual capability on a PC or a Linux box, but creates specific binary files for the ARM family processor [25].

The main API program `radio.c` code contains both function calls available for higher layer interface and private routines to be called internally by the driver. For the radio and MAC setup, the function `radio_mac_startup(setradio, setpa, mid, slotlen, numslot, txslot, rxslot)` will reset the radio, select the power amplifier, configure all the settings for the MAC-related registers and choose the default frequency pair for TX and RX. This function also calls other default settings such as the default packet size and correlator threshold value. To begin a transmission process, a node will call `radio_send_data(buffer_ptr, destination_id)` if it wants to send data, and `radio_send_data(buffer_ptr, destination_id)` if the node wants to send a synchronization packet to others. These two calls will append an appropriate header and a checksum, perform Viterbi encoding (optional based on the hardware implementation) and perform a final Manchester encoding before transferring the packet through the LMI. The function call checks the available space in the TX FIFO and returns an error if a packet must be dropped.

At the receiver, the μAMPS node relies on the external interrupt mechanism, which can be activated using ext_interupt_en() routine. The routine sets an appropriate register to turn on the interrupt bit in the SA1110, and the higher layer of the network stack can poll for the received packet by calling dequeue_rxpacket() and frees the memory buffer using free_rxpool().

As discussed in section 3.1, one of the most crucial challenges in deploying TDMA in a sensor network system lies in global synchronization. When not all the nodes are within one another's radio range, our algorithm simplifies the task by assigning the base station to be a master node that initiates the frame synchronization packet. As a master node, the base station is responsible for aligning the frames of all the neighbor nodes within its radio range. By calling a routine send_syn(destination_ID) with the destination_ID being multicast ID, the master node transmits a packet of packet type *syn*. Upon receiving the packet, neighbor nodes recognize the packet as a syn packet from the master node. These nodes, then, align their TDMA frames to match that of the master. The aligning task is done in a delayed service routine for external interrupt. If MAC ID 0, which corresponds to TX time slot 0 in our scheme, is assigned to the master node, then the master node can only transmit in time slot 0. Any neighboring nodes receiving the syn packet from the master node can instantly interpret that the current slot of the master is slot 0. With prior knowledge of guard time and slot length, each node calculates when the next slot should begin and then sets that next slot to be slot 1. The routine tdma_syncframe(next_slot, countdown_counter) is responsible for reconfiguring MAC hardware to start a new slot with slot ID next_slot once the countdown_counter expires. If we assume a master node with the MAC ID 0, the `next_slot` argument should always take the value of one, regardless of the receive node's ID. Once synchronized, each node then assigns a TX slot based on its MAC ID, leaving the rest of the slots as either RX or idle slots.

The API leaves the task of adjusting the MAC schedule to match other network parameters to a higher layer on the stack. An example is the reconfigurable value of the packet size, which is set via another API routine. This configurable packet size provides a convenient mechanism for the higher layer to adjust its ideal fixed packet size to suit each application. For example, in a vehicle tracking application, each sensor node may be able to compute a Line of Bearing if it has at least three microphones connected to the node. A node then may choose to communicate only the information on estimated angle to the vehicle instead of the entire acoustic information of the three sensors. Each decision, differing in terms of data rate and computation requirement, affects the choice of a packet size. A shorter packet length suits applications with low data rate which comes in regular intervals while a longer packet size better suits applications with bursty data stream. The driver must also configure the TDMA slot length with the constraints set by the packet length. Each TDMA slot must allow sufficient time for an entire packet to be transmitted or received. Since the link implementation of the μAMPS only allows at most one packet to be transmitted or received per TDMA slot, an unnecessarily extra-long slot will waste a shared bandwidth of the channel.

Chapter 4

Performance Analysis and Microsensor Network Applications

4.1 Fault Models of the Protocol Stacks

This section presents various fault models at different levels of the protocol stacks. Generic and μAMPS specific models, as well as masking techniques, will be discussed.

4.1.1 Bit level errors

Errors at a bit level can occur in a noisy communication channel and link layer. Regardless of the source of the errors, bit-level errors can be categorized into two of the following groups.

4.1.1.1 Substitution errors: A substitution error causes a bit to flip its value, from a one to a zero or from a zero to a one, and is caused by specific characteristics of the communication channel. For example, a channel may be modeled as an additive white Gaussian noise (AWGN) channel, probabilistically causing bit flips at a receiver. A channel may suffer from path loss and fading, which attenuate the transmitted information and may result in all bits being zero at a receiver. An error correction code, such as Viterbi algorithm, can be used to correct substitution errors.

4.1.1.2 Deletion and insertion errors: These two types of errors are likely to occur at the bit recovery process, i.e., during which the continuous-time signal is transformed into a discrete-time signal. Due to clock frequency variations and jitters, the clock at the receiver node may sample more or less than regular threshold values, causing a deletion or insertion. An edge detection technique in the data recovery also suffers deletion and insertion errors from glitchy signals. At every new edge detected, the data recovery is prompted to presume a start of a new bit. The μAMPS implementation of edge detection, however, is designed to ignore a glitch of width below 10% of the regular width of one bit, hence can absorb most errors of this type.

4.1.2 Packet and MAC level errors

Bit level errors can propagate the fault to the packet and MAC level. This section attempts to characterize two kinds of errors in this category.

4.1.2.1 False packet detection and missed packet detection: These two types of errors occur when the correlator fails to declare a correct "match" of the SYNC pattern. Configured with a low threshold value, the correlator has a higher probability of detecting a SYNC as well as a higher probability of false alarm. The false detection of a packet, however, can be discarded once the checksum invalidates the correctness of the packet.

4.1.2.2 TDMA frame synchronization error: A frame synchronization error occurs when oscillators of any two nodes have drifted so the TX and RX frames do not align. The μAMPS TDMA relies on periodic synchronization packets, so a frame synchronization error occurs only when synchronization packets are lost or not sufficiently sent. A receiver will align its frame to the node that sends out the packet of type *syn* after verifying the validity of the packet. If a 10 MHz oscillator with an accuracy of 50 parts/million is used (9.9995 to 10.0005 MHz), the maximum difference between the frequencies of an oscillator at a receiver and an oscillator at a transmitter is $10.0005MHz - 9.9995MHz = 1kHz$. Guard time at the beginning of the slot and gap time at the end of the slot can absorb the frequency shift. With the phase lock time of 500 μsec for the μAMPS radio, a guard time of 500 + τ μsec at the beginning and gap time of τ μsec at the end of the TX slot can absorb a drift of τ μsec as shown in Figure 4.1.

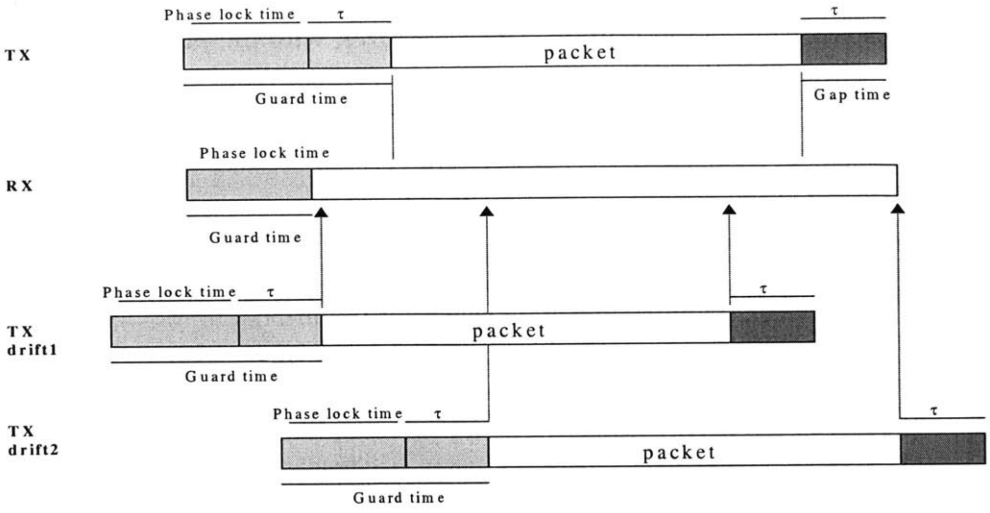


Figure 4.1: Frequency drift absorption by guard time and gap time
A perfect alignment of TX and RX slots places a received packet at the middle of an RX slot. When a frame shifts due to an imperfect clock, a guard time and gap time of size τ can absorb the misalignment, allowing the packet to be received at the left most or the right most of an RX slot

Figure 4.2 illustrates the μAMPS shortest interval between any two synchronization packets needed to keep the probability of synchronization error below a certain specified value. The plots assume a conservative estimate of probability of error of each bit being 10^{-4}. For a current packet size of 256 bytes with a 6-byte header and a 2-byte checksum, which totals $264 \cdot 16$ Manchester encoded bytes, the probability of a packet being corrupted $P_e = 1-(1-0.9999)^{264 \cdot 16} \approx 0.34$ (which is a conservative estimate compared to the experimental data of $P_e < 0.21$ in Figure 4.5 of Section 4.2.2). Denote T_{drift} as a period of time taken (after the most recent arrival of a synchronization packet at the receiver) for the starting point of a TDMA frame of the transmitter and that of the receiver to be τ μsec apart (τ bits at 1 μsec/bit), such as RX and TX-drift1 or RX and TX-drift2 in Figure 4.1. According to a simple calculation, a drift takes $_{drift} \approx \frac{(\tau \cdot 10^{-6})\sec \cdot (10\text{MHz})^2}{1kHz} = (\tau \cdot 10^{-1})\sec$ (with the previous assumption of a worse-case difference in frequencies being 1 kHz). For example, for a τ of 300 μsec, the frame will shift within approximately 30 sec. If x synchronization packets are sent during the 30 sec interval, with the probability of each one being corrupted $P_e = 0.34$, then we can find the value x to keep the probability of frame shift (which is the probability that none of the synchronization packets arrive correctly during that time period) below a specified value. According to the relationship in Figure 4.2, the μAMPS network with its current guard time τ (additional to the phase lock time of 500 μsec) being 300 μsec, should resend synchronization packets every 1.5 seconds.

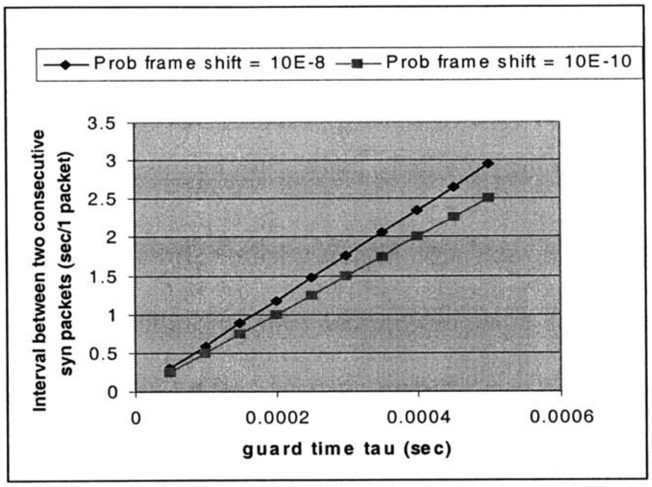


Figure 4.2: Required intervals between any two consecutive synchronization packets

4.2 Performance Analysis

Possible performance metrics for a wireless sensor network include resource utilization, power and energy consumption, probability of errors, etc. This section explores and characterizes various aspects of metrics, with specific data based on the μAMPS node implementation.

4.2.1 Hardware resource utilization

The digital baseband Link Manager and MAC VHDL is limited by the resources available in the Xilinx FPGA. Relative to other Xilinx FPGA with 256 I/O pins, the VirtexE XCV300E-FG256 already provides the most resources in terms of logic slices and memory. The analysis of the resources for the protocol stacks implementation, with and without the Viterbi coding algorithm, is shown in Figure 4.3. The design consists of 108,052 gates, and the number of gates increase by 13% to 121,718 gates with the Viterbi decoder. The graph shows that the number of slices are the limiting factor of the implementation with Viterbi. Each slice in an FPGA contains two registers and two four-input look-up tables (LUTs), both of which are the underlying building blocks for the FPGA logical computation. A computationally intensive Viterbi decoder requires additional logic units, meaning additional slices. Note that the Viterbi algorithm does not require extra RAM, or buffers.

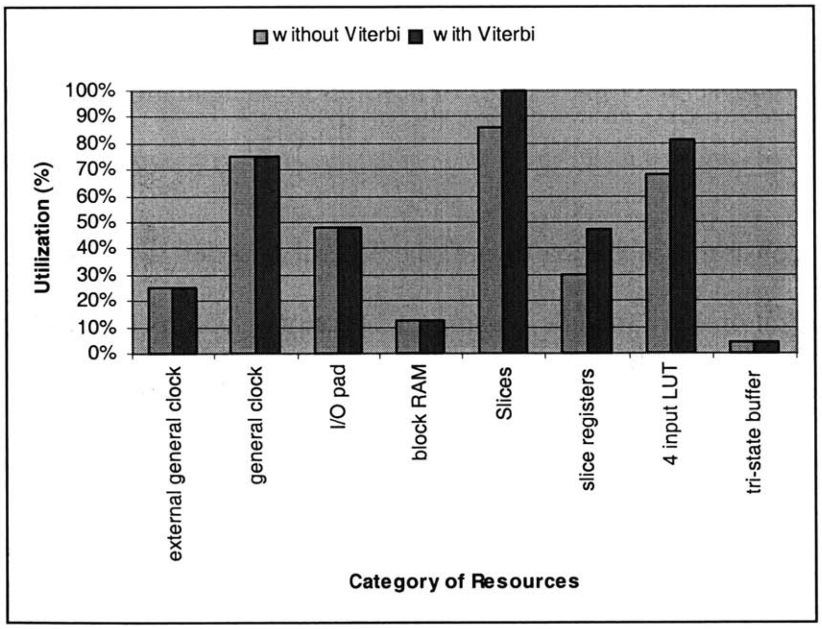


Figure 4.3: Xilinx VirtexE XCV300E-FG256 resource utilization

Two changes are made in order to fit the Viterbi algorithm with the rest of the implementation. First, the area, instead of speed, is optimized for the implementation of the VHDL code onto the chip. Second, slices with unrelated logic must be allowed. A slice with unrelated logic refers to a slice whose one or more input to its slice registers does not come from an output of the LUTs of the same slice. Both changes impose routing inefficiency. Figure 4.4 illustrates the impact of these changes. The graph displays the increase in the number of slices containing unrelated logic, from 0% in the case without the Viterbi decoder to 45% in the case with the Viterbi decoder.

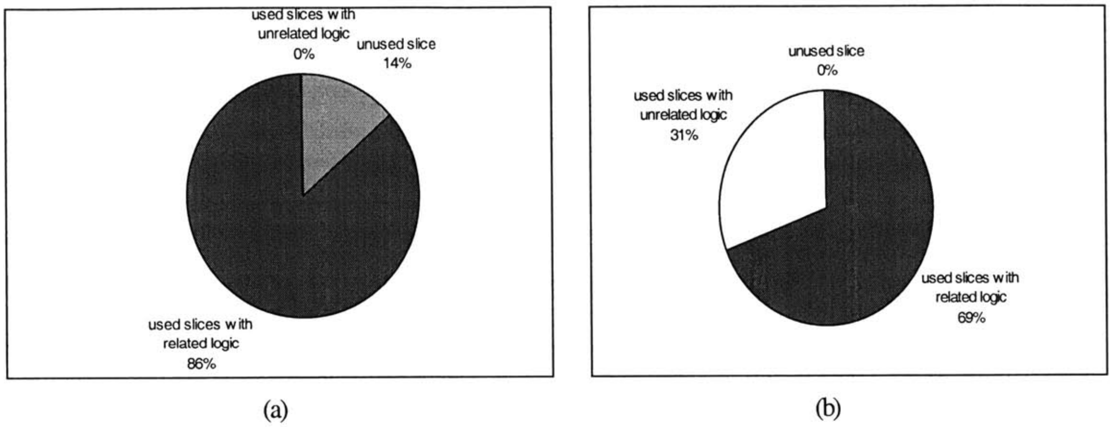


Figure 4.4: Analysis on Xilinx's slices of μAMPS implementation (a) with and (b) without the Viterbi decoder

With the constraint in the hardware, the Viterbi decoder is left as an option. Other VHDL optimization may include reusing of operators, such as counters or adders. It is also worth mentioning that the four block-RAMs used in the design are all within the FIFO core. No other parts of the design make use of the other 28 available blocks. In some situations, a RAM may be used as a replacement for a register, freeing some slices with a delay trade-off as the RAM's addressing scheme proceeds slower than a register's direct access.

4.2.2 Power, distance, and bit error rate

Specific wireless applications require specific levels of data reliability and attention to energy. In critical tasks such as machine monitoring, a high level of data reliability may be needed, while other tasks such as cell phone or other voice applications may tolerate poorer quality of data transmission. This section investigates the data reliability and power consumption of the μAMPS node.

Experimental data on μAMPS nodes is obtained in an open field. Five thousand packets of size 256 known bytes plus a header and a checksum are Manchester encoded and transmitted in each experiment, resulting in 1,024,000 useful bits being communicated per experiment. Power amplifier levels and the physical distances between the transmitter and the receiver are two parameters varied. The three power levels tested are 3, 10 and 15 dBm, and the distances measured are 5, 9, 13.5, 18, 27, 36, and 50 meters. When a packet arrives at the receiver, it first checks the validity of the header. The receiver counts a packet as a 'received' packet only if the header is verified (i.e., the destination node is correct), and it proceeds to verify the correctness of the data for the received packet. A packet with no errors is marked as uncorrupted, while a packet with one or more errors is marked as corrupted and the corrupted bytes are calculated. The difference between the received and sent packets are considered lost. Note also that additional synchronization packets are sent approximately every two seconds.

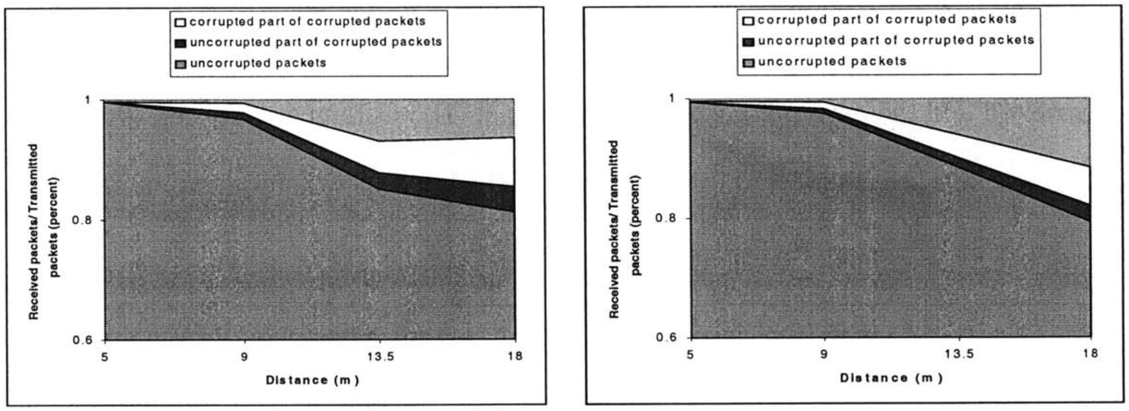
Distance	**Received Packets**	**Uncorrupted Packets**	**Probability of Errors (on received packets)**	**Probability of Errors (on transmitted packets)**[a]
5	4995	4995	$< 10^{-7}$	10^{-3}
9	4994	4994	$< 10^{-7}$	$1.2 * 10^{-3}$
18	4982	4982	$< 10^{-7}$	$3.6 * 10^{-3}$
27	4982	4982	$< 10^{-7}$	$3.6 * 10^{-3}$
36	4982	4982	$< 10^{-7}$	$3.6 * 10^{-3}$
50	4979	4896	$9.7 * 10^{-3}$	$1.39 * 10^{-2}$

a. A lost packet is counted as 256 bytes of errors

Table 4.1: Experimental Data on μAMPS node with Output Transmit power 3 dBm

Table 4.1 lists the data collected at the output transmit power of 3 dBm. The probability of error on received packets is less than 10^{-7} even at the distance of 36 meters. The perfect bits received once a packet is detected suggest that lowering the threshold value of the correlator may help decrease the packet lost ratio while maintaining the quality of the received bits

The data collected for the output transmit power of 3 dBm are obtained when the transmitter and the receiver face each other. This setup allows the best radiation pattern as the energy from the transmitter can propagate directly to the receiver. The two other experiments, with output transmit power of 10 dBm and 15 dBm, have a different setup where both nodes are facing up in the air. The result is plotted in Figure 4.5



(a)

(b)

Figure 4.5: Probability of error with transmit power of (a) 10 dBm and (b) 15dBm.

Both (a) and (b) of Figure 4.5 confirm the trend of the trade-off between distance and probability of errors when an output transmit power is fixed. However, the trade-off between the output transmit power and the probability of errors is not as evident from the plots. Figure 4.6 better represents the relationship between the output transmit power and the probability of error. Figure 4.6 (a) represents the probability of error on received packets (byte level), and Figure 4.6 (b) represents the probability of uncorrupted packets on received packets (packet level), both of which exhibit the expected relationship between the output power and probability of errors. As the distance increases, the probability of byte errors increases and the probability of uncorrupted packets decreases.

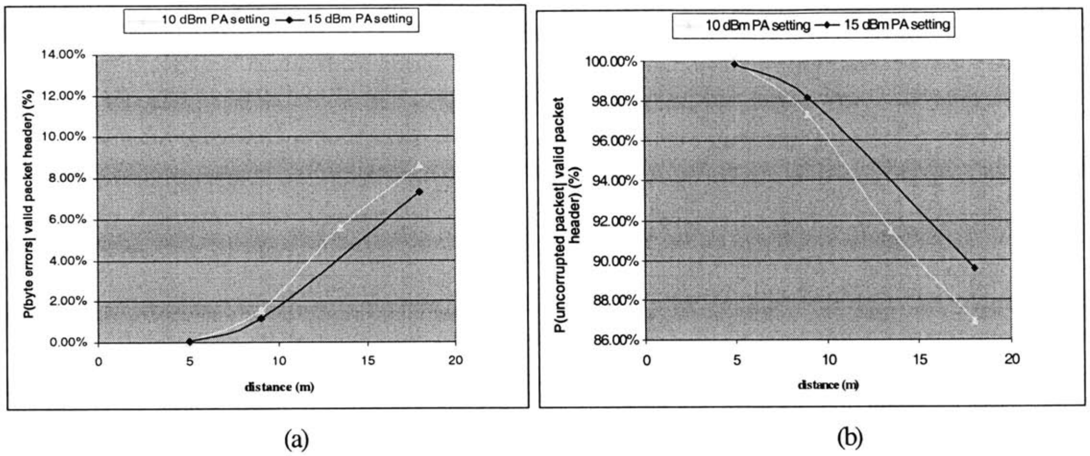


Figure 4.6: The probability of (a) byte error and (b) uncorrupted packets, based on received packets, versus distance.

Finally, the experimental data is compared to the theoretical benchmark. The probability of error P_b depends on the received energy per bit to noise power ratio $\frac{E_b}{N_0}$, which in turn depends on both output transmit power and distance. The received energy per bit to noise power ratio $\frac{E_b}{N_0}$ follows the relationship $\frac{E_b}{N_0} = P_{out} - N_f + 10\log W - P_{loss}\bar{\alpha} - 174\text{dBm/Hz}$ where N_f is the noise figure at the receiver, N_{th} is the thermal noise (-174 dBm/Hz), W is the bandwidth of the signal, P_{loss} is the radio high path loss, and $\bar{\alpha}$ is the average attenuation factor of the fading channel. The path loss exponent varies between 2 to 4 with the formula $P_{loss}\bar{\alpha} = A + B\log d$ in dBm where B is the path exponent multiplied by 10. With the varying distance d, Figure 4.7 plots the relationship between P_b and $\frac{E_b}{N_0}$ of the data collected with the output transmit power 3 dBm. A path loss of the μAMPS node can be extrapolated to be $60 + 20\log d$ (path loss exponent of 2), seen as a dotted line in the figure. The graph is plotted with a signal bandwidth W of 1 MHz and an estimated noise figure $N_f \approx 10$.

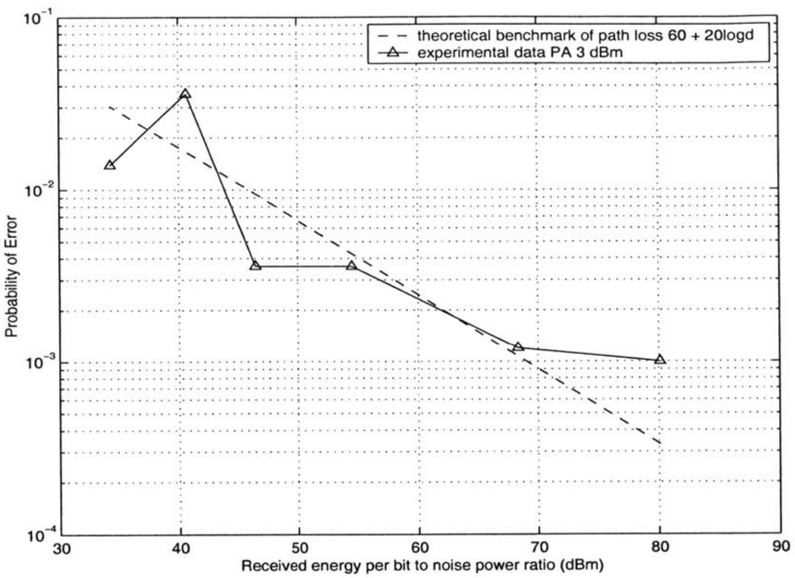


Figure 4.7: Received energy per bit to noise power ratio vs. probability of error. The path loss of μAMPS node can be estimated as $60 + 20\log d$, as seen on the theoretical benmark.

Another metric for a power-aware system is energy consumption. In order to assess energy consumption of the μAMPS-1 implementation, an energy model of the radio must first be formulated. A model used in μAMPS-0 by Eugene Shih [2] will be reviewed as follows. Generally, the average energy consumed by the μAMPS radio implementation is $E_{radio} = E_{tx} + E_{rx}$, which can be expanded as in equation (4.1).

$$E_{radio} = [P_{tx}(T_{on,\,tx} + T_{startup}) + P_{out}T_{on,\,tx}] + P_{rx}(T_{on,\,rx} + T_{startup}) \quad (4.1)$$

P_{tx} is the power of the transmitter without the transmit power amplifier. Note that $T_{startup}$ is the time during which the radio waits for the PLL frequency to be settled and that there is no need for a power amplifier during that period. With a worst-case PLL lock time of 500 μsec, the additional guard time beyond the PLL lock time and gap time of 300 μsec each, and the packet transmission time of $264 \cdot 16 = 4224\mathrm{usec}$, $T_{startup} = 500\mathrm{usec}$, $T_{on,tx} = 300 + 4224 = 4524\mathrm{usec}$, and $T_{on,rx} = 600 + 4224 = 4824\mathrm{usec}$. Note that the transmitter does not have to be on during the gap time after it finishes sending a packet.

The power at different power amplifier settings is shown in Table 4.2. The last column shows the total radio energy consumption per useful bit (256 * 8 = 2048 bits per packet). As for the power measurement consumed by the Xilinx baseband, the steady state current is < 7 mA at 1.8 V. The energy consumed by the baseband is trivial compared to the radio power, and hence it is not considered in this calculation.

Unit	Settings	Power	$E_{radio} = E_{tx} + E_{rx}$ per useful bit
TX radio & baseband	0 dBm	204 mW	1.36 μJ
	3 dBm	303 mW	1.38 μJ
	10 dBm	367 mW	1.40 μJ
	13 dBm	510 mW	1.43 μJ
	15 dBm	1224 mW	1.61 μJ
RX radio & baseband		364 mW	

Table 4.2: μAMPS radio and baseband power consumption

4.3 Application Demonstration

4.3.1 Computation vs. Communication

Two of the most important units in any wireless network are the data and control processing and the communication hardware. Advanced circuit techniques allow today's transistors to run at gigahertz speeds. Technology scaling also allows the processor to be designed to operate at a lower supply voltage, and hence can be expected to consume much less energy. An important characteristic of processors for power-aware systems is that the processors be enabled for dynamic voltage scaling. A StrongARM processor, for example, can also operate at different frequencies, offering further power savings and power-awareness.

The signal strength of the transmitter is mainly attenuated by a high path loss. In typical wireless systems, power dissipation is dominated by the communication module, since communication tends to cost more than computation. The performance of the communication module is related to the signal-to-noise power ratio, as presented in Equation (4.1). Most likely, a wireless system's power is drained in the communication unit. The data rate of the network is also governed by the nature of the Shannon Limit. As a result, any application running on a wireless network should consider this trade-off between the two units.

In a remote sensing application, multiple nodes operate from scattered locations to collect information on any changing environment. To optimize the energy consumption, each node should process as much information as possible locally before routing the data or results to other nodes or to the base station.

Vehicle tracking, a specific application of a remote sensing application using beamforming, has been implemented onto the μAMPS node. The next section discusses the specific details of the experiment.

4.3.2 Vehicle Tracking Application

The μAMPS nodes are equipped with four acoustic sensor channels, connected to the processor via four separate A/Ds [25]. With three or more microphones per node, a sensor node is capable of processing data from the three microphones and performing a signal processing algorithm, resulting in an estimation of the direction of the source. Beamforming is the signal processing algorithm used to find the estimated angle. The basic rule is to find the delay between the waveforms collected at each sensor and then to use the speed of sound to compute the distance of the source from each sensor. With *a priori* knowledge of the microphones' locations, the algorithm can estimate an angle or line of bearing (LOB) of the acoustic source from the center of the node. With two or more nodes, the system can triangulate the specific location of the acoustic source. Figure 4.8 illustrates the beamforming application with three sensor nodes and a base station.

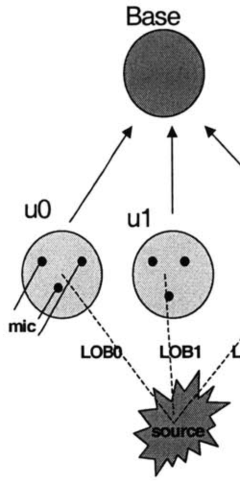


Figure 4.8: An example of a beamforming application setup
Each μAMPS node is equipped with three microphones, collecting an acoustic source from the environment, as well as performing signal processing algorithm to produce an LOB that estimates the direction of the source.

With the beamforming software from the ARL and ISI with collaboration of Alice Wang, Nathan Ickes and Kevin Atkinson, an application has been demonstrated in lab with a setup of one μAMPS sensor node and one μAMPS base station node. A μAMPS sensor node, connected with three microphones, performs all the computation locally. Four packets are sent each minute with updated information on the estimated angle of the acoustic source. A base station node, upon receiving the updated angle, sends the information to a PC via a serial port. A JAVA application on the PC updates the screen, drawing a new picture of a vehicle with the location based on the new estimated angle.

The demonstration illustrates the functionality of a two-way communication over the protocol stacks. A base station, considered a master node of the network, periodically transmits a synchronization packet to the sensor node. The sensor node, upon receiving a synchronization packet, aligns its TDMA frame to that of the master node. This demonstration also illustrates the idea of communication vs. computation, as only ten bytes are sent per packet for information on angles. A packet size and slot length can be adjusted for this specific application to minimize the energy consumption and bandwidth utility.

4.4 Conclusion and Future Work

My thesis has demonstrated a design and implementation of a wireless network protocol stack on μAMPS node prototypes. The design of the protocol stacks, although presented separately for each layer, is in fact highly integrated and application-specific. The integrated design framework reduces the overhead and allows some tasks to be operated cross-layered, such as power-management. Packet and packet header design is another example of the cross-layered design, as the header combines the MAC level information such as the destination and source ID with the link level information such as encoding options, reducing the packetizing and depacketizing process.

The highlight of the DLC and MAC implementation is on its configurability. Different transmit output power levels can be adjusted with ease and ECC algorithm is an option to be declared in the packet header. The MAC implementation offers a convenient software interface and the hardware internal registers that govern the timing and power control of the TDMA schedule. In addition, a MAC reuse slot algorithm is proposed for bandwidth and power efficiency, accomplished with the use of a variable transmit output power. The analysis and the application demonstrations confirm the functionality and performance of the network protocol stacks.

Continuing research can go in several directions. First, one can improve the existing link layer. As the resources are limited in the FPGA, an ASIC or a chip design are some of the alternatives so that the architecture is optimized. If an ECC is not limited by the FPGA resources, experimental data should be taken to verify the link performance with the coded communication, which also demonstrates lower energy consumption.

At the MAC layer, a startup algorithm was designed for a distributed network with the proposed slot reuse algorithm. Due to the limited number of nodes built, a network of three or more have not yet been tested, although the hardware and software algorithm and implementation is transparent and already in place. The existing implementation functions with any network of up to 15 nodes.

The link layer and MAC design was implemented on the μAMPS-1 node to prove functionality and for valuable power and performance measurements for real-time wireless networks. Through this analysis and study, we showed a proof of concept of a power-aware baseband system for wireless sensor networking.

References

[1] Wendi Heinzelman. PhD's Thesis, Massachusetts Institute of Technology, May 2000.

[2] Eugene Shih. Master's thesis, Massachusetts Institute of Technology, May 2001.

[3] Fred S. Lee. Master's thesis, Massachusetts Institute of Technology, May 2002.

[4] Henrik Lonn. Initial Synchronization of TDMA Communication in Distributed Real-Time Systems. *Proc. of 19th IEEE International Conference on Distributed Computing Systems,* 1999.

[5] Katayoun Sohrabi, Jay Gao, Vishal Ailawadhi, and Gregory J. Pottie. Protocols for Self-Organization of a Wireless Sensor Network. *IEEE Personal Communications*, October 2000.

[6] Christian Rohl, Hagan Woesner, Adam Wolisz. A Short Look on Power Saving Mechanisms in the Wireless LAN Standard Draft IEEE 802.11.

[7] Alice Wang and Anantha Chandrakasan. Energy-Efficient System Partitioning for Distributed Wireless Sensor Networks. *ICASSP 2001*, May 2001.

[8] Curt Schurgers, Gautam Kulkarni, and Mani B. Srivastava. Distributed Assignment of Encoded MAC Addresses in Sensor Networks. *ACM MobiHOC 2001.*

[9] Vilgot Claesson, Herik Lonn, and Neeraj Suri. An Efficient TDMA Synchronization Approach for Distributed Embedded Systems.

[10] G. J. Pottie and W.J. Kaiser. Wireless Integrated Network Sensors. *Communication of the ACM*, May 2000.

[11] National Semiconductor Corporation, *LMX3162 Single Chip Radio Transceiver*, November 1999.

[12] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. *Proc. MobiCom*, August 2000.

[13] J. Rabaey et al. PicoRadio supports *ad hoc* ultra-low power wireless networking. Computer, vol. 33, no. 7, July 2000.

[14] E. Shih, S. Cho., et al. Physical Layer Driven Protocol and Algorithm Design for Energy Efficient Wireless Sensor Networks. *Proc. 7th Annual International Conference on Mobile Computing and Networking, MobiCom 2001.*

[15] Dimitri Bertsekas and Robert Gallager. *Data Networks, second Edition.* Prentice Hall, NJ 1992.

[16] G. Asada et.al, Wireless Integrated Network Sensors: low power systems on a chip. *Proc. 1998 ESSCIRC*, September 1998.

[17] B. Warneke, B. Atwood, K.S.J. Pister. Smart dust mote forerunners. *Proc. 14th IEEE International Conference on MEMS.* January 2001.

[18] B.P. Upender and P.J Koopman. Embedded Communication Protocol Options. *Proc. of Embedded Systems Conference 1993.* October 1993.

[19] Katayoun Sohrabi and Gregory J. Pottie. Performance of a Novel Self-organization protocol for Wireless ad-hoc sensor networks. *Proc. of the IEEE 50th Vehicular TEchnology Conference, 1999.*

[20] R. Min et al. Low-Power Wireless Sensor Networks. *Proc. 14th International Conference on VLSI Design.* Bangalor, India, Jan 2001.

[21] Jean-Pierre Ebert and Adam Wolisz. Combined Tuning of RF Power and Medium Access Control for WLANs. *Proc. MoMUC ‘99*, 1999.

[22] Paul Lettieri and Mani B. Srivastava. Adaptive Frame Length Control for Improving Wireless Link Throughput, Range, and Energy Efficiency. *Proc. INFOCOM ‘98*, March 1998.

[23] B. Narendran, P. Agarwal, S. Yajnik, and J. Sienicki. Evaluation of an Adaptive Power and Error Control Algorithm for Wireless Systems. *Proc. ICC ‘97*, May 1997.

[24] Intersil Corporation. *AN9633: Processing Gain for Direct Sequence Spread Spectrum Communication Systems and PRISM*, August 1996.

[25] Nathan Ickes. Master’s Thesis, Massachusetts Institute of Technology, May 2002.

[26] John Scourias, “Overview of the Global System for Mobile Communications”, http://www.gsmdata.com/overview.htm, October 1997.

[27] http://www.bluetooth.com.

[28] http//www.homerf.org.

[29] http//grouper.ieee.org/groups/802/11/index.html.