WIRELESS NETWORK TOOLKIT

Jonathan Hill
Electrical and Computer Engineering Department
University of Hartford
West Hartford, CT 06117

Abstract

The wireless network toolkit provides students with an inexpensive means to study and experiment with general wireless network principles. The toolkit provides tools that students can use in their own projects and can also be used to demonstrate wireless network principles in an undergraduate laboratory setting. The toolkit is also an example layered network model that students can study in a classroom setting. One goal of the toolkit is to provide a useful level of transparency so that students can be exposed to as much of the system as desired. The field programmable gate array (FPGA) provides students with entirely new opportunities to examine the details of such enabling technology in a meaningful way.

Introduction

The wireless network toolkit provides students with an inexpensive means to study and experiment with general wireless network principles. As technology moves forward the nature of what is practical in wireless networks advances. The need for such a toolkit arises however as consumer market pressure to make products easier to use actually distances users from the technology. While having very complicated wireless devices available that are also very easy to use helps promote sales, the details of the enabling technology that makes such products work is hidden from the user. The first goal of this toolkit is to provide students with an inexpensive means to study principles that are widely used in wireless networks. The toolkit is composed of the following components:

- The physical hardware – A wireless transceiver board and a field programmable gate array development board which hosts a soft-core processor system.
- Peripheral description – The logic corresponding to the wireless transceiver behaves as a peripheral device accessed with memory mapped registers.
- Low-level software library – Students using the toolkit in their own projects can use this function oriented library to access the hardware.
- Utilities software library – Middle level software such as that used to check for transmission errors.
- Demonstration applications – In a laboratory setting students will benefit from pre-written demonstration software.

The toolkit has three purposes. First, provide tools that students can use in their own projects. Second, serve as a demonstration tool in an undergraduate laboratory setting. Third, serve as an example for study in a classroom setting. Our research has focused on the physical and media access aspects of wireless networks. Two graduate students have been involved and have completed their master's research. A simple point to point keyboard chat program has been demonstrated. In the current form the toolkit is useful to undergraduate student projects and demonstration software is being developed. Undergraduate students are using the toolkit in the design of a wireless electro cardiogram (ECG). We have plans to incorporate the toolkit into a data networks course that will be developed for our technology students.

One goal of the toolkit is to provide a useful level of transparency so that students can be exposed to as much of the system as desired. The field programmable gate array (FPGA) provides students with entirely new opportunities to examine the details of such

enabling technology in a meaningful way. Given the need for a supporting microprocessor system, this project showcases a new paradigm, the use of a single FPGA to construct an entire microprocessor system. The Xilinx embedded developer's kit (EDK) conveniently constructs the underlying microprocessor system using classic computer engineering principles. Students interested in such detail are welcome to examine the underlying microprocessor system. Otherwise, students can use an existing system as-is.

Network Layers

Tanenbaum[1] presents the OSI network model to introduce how a computer network node is divided into *layers*. The toolkit is organized around the idea of the network *layers* in Figure 1. The results from our recent research have focused on the physical layer and the media access (MAC) sub-layer. The distinction seen here between layers and sub-layers arises as the division between layers is fairly distinct, while sub-layers in a given layer share information about each other.

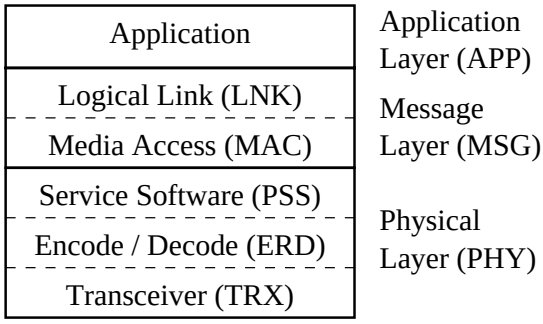

Figure 1: Network layers.

The transceiver (TRX) sub-layer comprises a hybrid analog device which fits on a small adapter board. The rest of the system is implemented on a FPGA development board. The Encode, Retime, and Decode (ERD) sub-layer uses Manchester Coding along with synchronous detection. The physical layer service software (PSS) sub-layer provides a function oriented interface to the hardware. The message (MSG) layer manages the details of reliably sending and receiving messages.

The primary assumption the MSG layer makes regarding the PHY layer is that it is an unreliable broadcast oriented data transport mechanism. In being broadcast oriented the media access (MAC) sub-layer handles the addressing of messages. The MAC and logical link (LNK) sub-layer work together to detect message errors and provide the appearance of a reliable data link. We currently do not have any intermediate inter-networking layers, so the application is atop the message layer.

The Host and Transceiver

Figure 2 is an actual wireless network node. The smaller upper board contains the transceiver and antenna. The lower board is an off the shelf Xilinx Spartan-3 FPGA starter board manufactured by Digilent Inc.[2]. The FPGA is a visible tan square to the middle right. The light colored cable to the left is the JTAG connector used for downloading the system and debugging purposes. The dark cable to the left provides power. The lower D-type connector is for RS232 serial communications. The upper

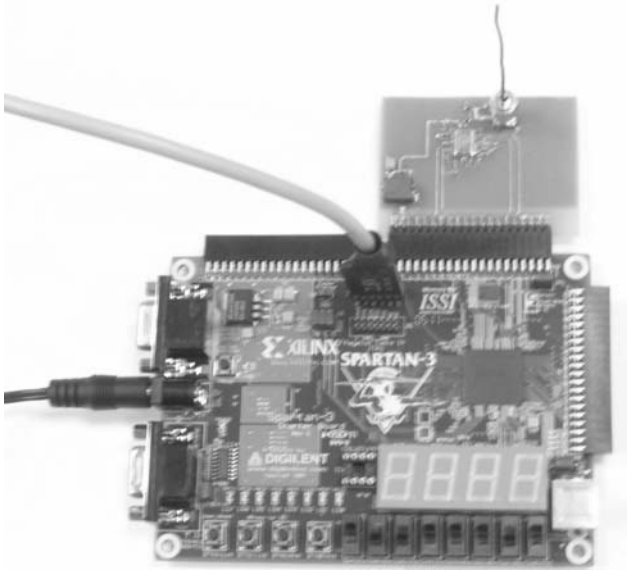

Figure 2: Wireless network node.

D-type VGA display connector and PS2 style keyboard connector to the lower right are currently not in use. Buttons, switches, LEDs, and four seven-segment displays are along the lower edge or the board. The FPGA provides ample space for a soft-core microprocessor as well as peripheral devices.

The transceiver selected is the RF Monolithics TR1100 hybrid[3] which operates in the 900MHz band. This transceiver is unique in its speedy transition between transmit and receive modes. The wireless link operates half-duplex, either transmitting or receiving. This transceiver is designed for short-range wireless data communications, supporting data rates up to 1Mbps. We arbitrarily selected 250 kbps as the Baud rate. The small size, low power consumption, operation with 3.3 volt logic, low cost, and respectable data rate make this transceiver ideal for our use.

In using a soft-core processor, the toolkit provides an opportunity for students to study an embedded microprocessor system. Apart from the use of an FPGA which provides an amazing level of flexibility, the soft-core microprocessor system in Figure 3 follows classic microprocessor systems design principles. The ERD and TRX sub-layers appear to the processor simply as another peripheral on the system bus. Apart from on-chip memory resources, the external memory controller (EMC) provides access to off-chip memory resources. The UART provides RS232 style serial communications.

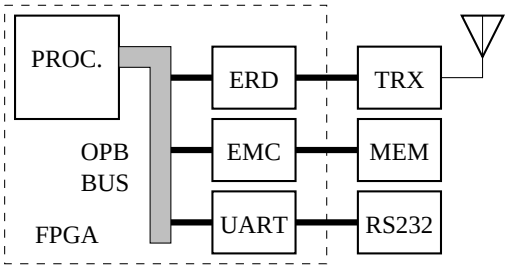

Figure 3: Soft-core microprocessor system.

While such a soft-core processor system is *described* by means of a hardware description language, the resulting *description* is not considered *software*. The description is *compiled* and *fit* to a device, producing an *image* or *bit file*, used to *configure* the FPGA resources. The bit file can be thought of as a *floor map* used to configure the FPGA. Once configured, the FPGA essentially becomes the desired system so that writing *programs* and *programming* the system involves the use of conventional software development tools. Software for the toolkit is currently written with the C language.

Synchronous Communications

Synchronous serial communications is a widely used technique whereby the transmitter uses a code to convey the data and clock together. A discrete time phase-lock loop inside the FPGA reproduces the corresponding clock to recover the received data. In our case, the selected transceiver[3] uses a blocking capacitor in the receiver to perform data slicing. Such AC coupling is significant as it rules out asynchronous communications protocols such as RS232 which have significant DC and near DC signal components.

To provide synchronous communications we selected the popular Manchester Code, a topic of nearly all computer network courses. In this research we follow IEEE 802.3[4] in that a '1' bit is represented by the sequence 'low-high' and that a '0' bit is represented by the sequence 'high-low' as in Figure 4. Here we refer to each such sequence as a *symbol*. The numbered vertical dash lines are the boundaries between *symbols*. Note that a transition always occurs at the center of each symbol.

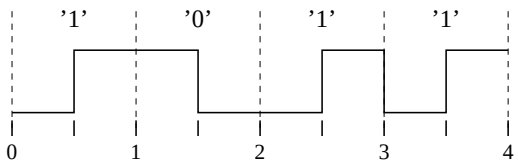

Figure 4: IEEE 802.3 Manchester coded data.

For the receiver to reproduce the corresponding clock, the toolkit uses a discrete time phase-lock loop. A broadcast message starts with a *preamble*, followed by a *start frame delimiter* (SFD). The preamble and SFD provide an easy mechanism for the retiming logic to obtain phase-lock. Starting from the left, the preamble is as follows:

10101010 10101010 10101010 10101010

The SFD continues the preamble pattern and ours ends with an all zeros *nibble flag*.

10101010 10101010 10101010 10100000

Figure 5 is how the preamble and SFD are encoded to form the start of a message. While the preamble pattern is broadcast, transitions occur only at the center of each bit-cell. Once phase-lock is achieved, the receiver is directed to ignore transitions that occur at the boundary between symbols.

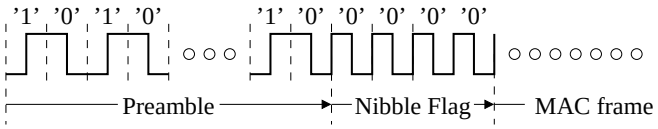

Figure 5: Beginning of a message.

Following the nibble flag is a MAC frame provided by the message layer. Encapsulation is an important principle for students to understand. The MAC frame is essentially treated by the PHY layer as cargo. Besides generating a Manchester coded message, to make it easy for students to observe Manchester coding, the toolkit provides a mechanism for the transmitter to continuously code an arbitrary 32 bit long pattern.

The transmit encoder in Figure 6 is double buffered in that a shift register and a one-bit register encode a 32 bit word while a another register holds the next word. Once the first word is encoded, the next word is taken and the transmit buffer empty flag is asserted. Students have the option to either use software polling or interrupts to service the encoder. Double buffering ensures that in transmitting a long message, the bit stream is continuous, without any breaks.

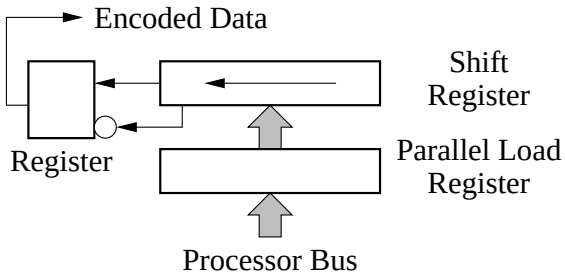

Figure 6: Manchester encoder.

Symbol Retiming

It is common practice to use a phase-lock loop to track a Manchester coded signal. In our case the actual transceiver output does not provide a signal-detect indicator. Without an actual received signal the demodulated output has a roughly periodic waveform that is otherwise quite different from Manchester coded data. The presence of a received signal is determined here by actually obtaining phase-lock. Given the shortness of the preamble, the time required to obtain phase-lock is of particular importance.

The phase-lock loop we use is constructed using well-known discrete time signal processing techniques, outlined below. In Figure 7 a flip-flop and exclusive-OR gate detect a signal transition. The discrete time signal generator (DCO) produces a saw-tooth waveform. Once phase-lock is established the PreLock signal is forced *low*, instructing the register (Reg.) by means of the control logic (Cntl) to load only near the center of each symbol. Each symbol is sampled 16 times, to produce one estimate of the phase error between the local clock and that corresponding to the

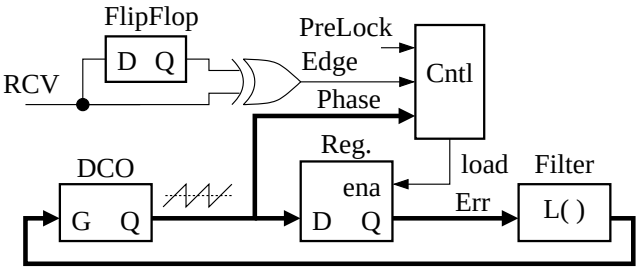

Figure 7: Discrete time phase-lock loop.

received data. The loop filter used to enhance the dynamic behavior is just a scaling factor.

The phase-lock loop is followed by an accumulate-and-dump module (ADM) that counts for each symbol the number of agreements between the received signal and DCO sign bit. A significant number of agreements or disagreements indicate phase-lock, where a significant number of agreements indicate a received '1' bit and disagreements indicate a '0' bit. Figure 8 outlines the receiver controller states.

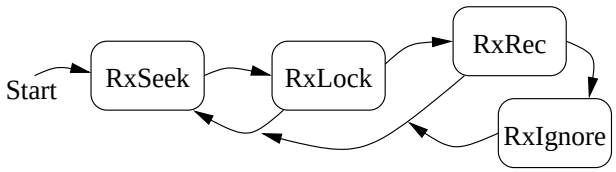

Figure 8: Receiver States.

In the RxSeek state the PreLock signal is asserted, with the assumption that during preamble transitions occur only at the center of each symbol. Upon achieving phase lock and then the reception of the nibble flag, the controller advances to the RxLock state and RxRec state, respectively. The RxIgnore state allows the receiver hardware to actively ignore a message that is damaged or not addressed to the station. The transmitter hardware in Figure 6 is also controlled by a state machine.

Phase-Lock Loop Analysis

Analysis of the phase-lock loop falls along the lines of classic theory and is of particular interest to students familiar with discrete time signal theory. As a useful reference I suggest Freeman[5]. The key here is that each received symbol is sampled N_s times, essentially scanned to produce one sample estimate of the phase error. In the following, the stability is considered and an optimal loop gain is selected to minimize the pull-in time. The model in Figure 9 assumes one estimate of the phase error per symbol. The values θ and ϕ as well as the corresponding values Θ and Φ are the received and local phase, respectively.

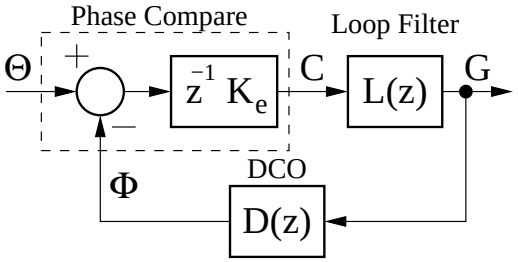

Figure 9: PLL signal model.

The delay in the phase comparison block models how an entire scan produces one sample estimate of the phase error. While steady state phase error can be eliminated by including an integrator in the loop filter, actually doing so increases the system order by one and also complicates the pull-in process. Given that a modest phase error can be tolerated, the loop filter $L(z)$ is a simple scaling factor K_ℓ. With the loop filter being a scaling factor, the phase lock loop is modeled neatly as a second order system. The DCO is modeled as a phase accumulator, with sensitivity K_d.

$$\frac{\Phi(z)}{G(z)} = \frac{K_d}{z-1} \qquad (1)$$

The system function from input θ to output g is as follows. The terms K_f and K are the forward gain and the overall loop gain, respectively.

$$\frac{G(z)}{\Theta(z)} = \frac{K_f(z-1)}{z^2 - z + K} \qquad (2)$$

where: $K_f = K_e K_\ell$

$K = K_d K_f$

To select a value for the loop gain and consider the stability of the system we inspect the denominator of (2) to identify regions for which the poles are contained by the unit circle. The Schur-Cohn stability criterion described by Freeman formally identifies the region of stability. Inspecting the denominator of (2) produces the ranges that we consider.

- $0 < K < 0.25$ - unique real poles
- $K = 0.25$ - poles repeat at $z = 0.5$
- $0.25 < K < 1.0$ - complex poles, decreasing dampening
- $K = 0$ or $K \geq 1.0$ - non-responsive or unstable system

The root locus in Figure 10 is the trajectory of the system poles with respect to K. In considering that each pole contributes to the system response, the furthest pole is closest to the origin, implying the greatest damping, when

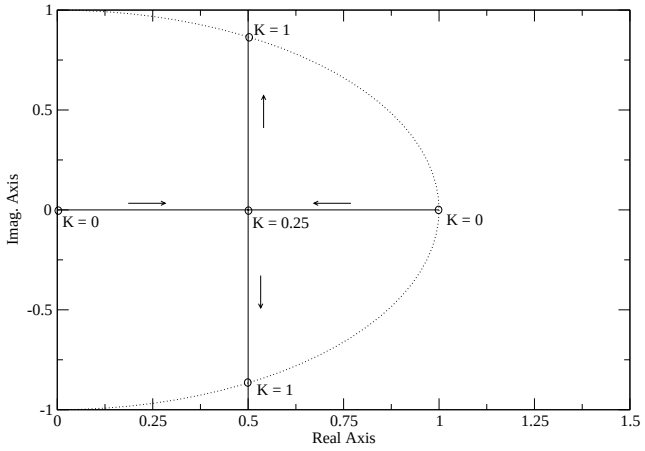

Figure 10: Phase-lock loop root locus.

Given the short preamble, the step response is of particular importance. With $K = 0.25$, the phase error step response is in (3). Given an initial phase error of one-half symbol, the model suggests that after 10 symbols, the phase error is less than 0.01 of a symbol. Further, in being an exponential response, we can have confidence that this will satisfy the requirements for acquisition with some margin to spare.

$$(\theta - \phi)_{step} = \frac{-n}{2^n}; n > 0 \qquad (3)$$

Message Layer

The message layer manages the physical layer in a way to provide the appearance of a reliable point-to-point data link. It does this with three mechanisms. First, addressing gives the appearance of a point to point link. Second, a cyclic redundancy check (CRC) code is used to detect transmission errors. Third, in the case of a transmission error, a mechanism retransmits data as is necessary. Tanenbaum[1] discusses retransmission techniques commonly used in networks. Having the MAC sub-layer examine the PHY status before broadcasting reduces the number of broadcast collisions that occur between stations. Other than a CRC which checks for bit errors, no actual method is currently used to detect collisions. To reduce the probability of a collision, the LNK can be responsible for establishing a protocol between stations, such as a token passing protocol.

The MAC frame in Figure 11 is produced by encapsulating a frame provided by the LNK sub-layer, with additional information that has meaning to the MAC sub-layer. In this case addressing is achieved by pre-pending a header to the LNK sub-frame and CRC checksums are used to check for transmission errors. The use of CRC codes is truly a classic computer data networking technique. Besides the myriad of reference material available on the Internet, Tang and Chien[6] as well as Williams[7] provide tutorial introductions to CRC codes. The wireless network toolbox uses the X.25 standard CRC generator polynomial as described by Williams.

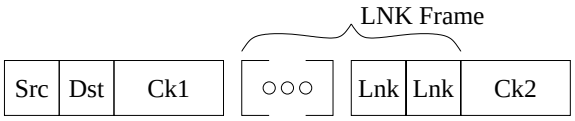

Figure 11: MAC message frame.

The MAC message frame header contains 8 bit message source (SRC) and destination (DST) address values. A first checksum (Ck1) in the header checks for errors in the source and destination addresses. A second checksum (Ck2) checks for errors in the LNK frame. In detecting a checksum error in the header the MAC layer only instructs the PHY layer to actively ignore the remainder of the message. In detecting a checksum error in the LNK frame however, the MAC layer also alerts the LNK that a damaged message was received from the indicated station. Address zero is reserved as a destination address for broadcast messages sent to all listening stations. The source address zero could represent a master station.

Use in the Undergraduate Curriculum.

The toolkit was developed with three uses in mind. First, provide tools that students can use in their own projects. Second, serve as a demonstration tool in an undergraduate laboratory setting. Third, serve as an example for study in a classroom setting. The project website[8] provides more technical information and useful information available to educators. In the following we elaborate on such uses of the wireless network toolkit.

The first student project using the toolkit is in cooperation with the neonatal intensive care unit (NICU) at Hartford Hospital. In this project the toolkit is used to construct a wireless electrocardiogram (ECG) that itself is a first step in the development of a wireless central apnea response system. The ECG system is wireless in the sense that it eliminates the use of long wires. Rather than using a simplex point to point wireless link, a personal area network oriented approach allows for a more robust system. In particular, bidirectional data transmission allows status information to be shared, to better monitor the reliability of the system.

The toolkit provides many opportunities to demonstrate wireless network principles in a laboratory setting. This aspect is being developed in preparation for a new computer networks course for our technology students. Access to the entire ERD sub-layer is through memory mapped registers. There is a transmit/receive data register, a configuration register, and a status register. The top-most PHY sub-layer is actually software that provides higher software with a function oriented interface to the hardware. The wxlib library provides a simple means for students to write programs that directly investigate the hardware. In the following, the receive-wait function is based on the simple idea of a spin-lock.

- WxSignal – start to continuously code a 32 bit message
- WxStop – stop continuously coding a 32 bit message
- WxRecWait – wait to receive a message
- WxSendStr – transmit an ASCII string message

Figure 12 has an exchange with a demonstration program written using the above functions. Choices 1 and 2 provide a means to start and stop producing a Manchester coded bit stream that students observe using an oscilloscope. Choice 3 is used to broadcast a message. A storage oscilloscope is used to examine the preamble and message body. Here the board receives a simple message from a remote station. This first demonstration program works the physical layer directly, so that all stations within range receive the message.

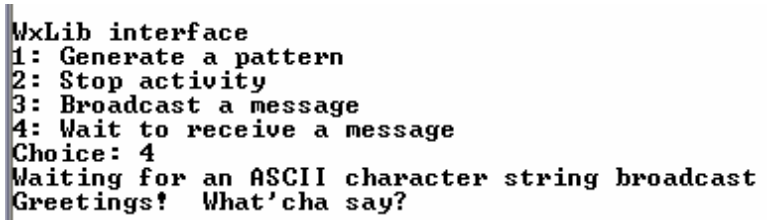

Figure 12: Screenshot demonstrating physical layer.

In a classroom setting the toolkit helps to crystallize together numerous wireless network topics that are in common use. In scanning this paper you will find references to such topics as Manchester encoding, retiming, CRC error detection, encapsulation, network layers, and more. In time a collection of classroom materials will be developed. Please visit the online website[8] for the project.

Summary

This paper outlines a wireless toolkit developed to provide students with an inexpensive means to study and experiment with general wireless network principles. In the current state students can use the toolkit in their own projects. One group is using the toolkit for a wireless electrocardiogram. Two graduate students completed their master's degree research by contributing to the development of the wireless toolkit. We are developing applications that can be used to demonstrate wireless network principles in an undergraduate laboratory setting and are planning on using the toolkit in a networking course being developed for our technology students. The toolkit also serves as an example worthy of classroom study. In closing, as a summer faculty research fellow, I send thank to my colleagues at the NASA Johnson Space Center for their discussion and interest in this project.

References

1. Andrew S. Tanenbaum, Computer Networks, Fourth Edition, copyright 2002 by Prentice Hall.

2. Digilent Inc., http://www.digilentinc.com/

3. RF Monolithics Inc., TR1100 data sheet, http://www.rfm.com/

4. IEEE Std 802.3-2002, *Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*, available at http://grouper.ieee.org/groups/802/11/

5. H. Freeman, Discrete-Time Systems, copyright 1965 by John Wiley & Sons.

6. D. T. Tang and R. T. Chien, *"Coding for Error Control,"* IBM System's Journal, volume 8, number 1, 1969, pp.48-86, available online from http://www.ibm.com

7. R. N. Williams, *"A Painless Guide to CRC Error Detection Algorithms,"* version 3, August 19, 1993, http://www.ross.net/crc/Downloar/crc_v3.txt

8. Project homepage, http://uhaweb.hartford.edu/jmhill/projects/WNTkit/index.htm

Biographical Information

Dr. Jonathan Hill is an Assistant Professor of Computer Engineering in the College of Engineering, Technology, and Architecture (CETA) at the University of Hartford, located in Connecticut. His Ph.D. and M.S. are from Worcester Polytechnic Institute and his bachelor's degree is from Northeastern University. He was previously an Applications Engineer with the Networks and Communications Division of Digital Corporation. He was a 2006 NASA Summer Faculty Fellow at the Johnson Space Center. His research interests involve embedded microprocessor based systems.