Combined Equalization and Decoding for IEEE 802.11b Devices

Chris Heegard, *Fellow, IEEE*, Seán Coffey, *Member, IEEE*, Srikanth Gummadi, *Member, IEEE*, Eric J. Rossin, Matthew B. Shoemake, *Member, IEEE*, and Michael Wilhoyte, *Associate Member, IEEE*

***Abstract*—Receivers for wireless local area networks based on the IEEE 802.11b standard are required to operate well in heavy multipath, as well as additive noise impairments. This paper discusses a practical approach to combined equalizing and decoding for IEEE 802.11b systems, based on the Fano sequential decoding algorithm. Simulation results are presented demonstrating the greatly improved performance attained using this algorithm to the performance obtained using separate equalization and decoding blocks. This method has been implemented in TI's ACX100 chip. Measured laboratory results are presented that demonstrate superior throughput results obtained from this device compared with throughputs obtained by competitive devices available on the market.**

***Index Terms*—Equalization, Fano decoding, wireless networking.**

I. Introduction

THE IEEE 802.11b standard specifies a physical layer transmission system that transmits packets of information in an Ethernet type network. The standard specifies three basic forms of modulation coding, Barker code (1- and 2-Mb/s payload data rate), CCK (5.5 and 11 Mb/s), and packet binary convolutional coding (PBCC) (5.5 and 11 Mb/s). The latter modulation was proposed and developed by Alantro Communications, now a part of Texas Instruments Incorporated. All of these forms of modulation, as well as a 22-Mb/s PBCC extension, are implemented in the ACX100 chip offered by Texas Instruments (TI). (This high-rate PBCC mode is scheduled to be a part of the upcoming IEEE 802.11g standard.)

While there have been several descriptions of the transmitted signals that are part of the IEEE 802.11b specification and the ACX100 chip [1], [2], a description of the novel receiver structures of the ACX100 chip have not been published. This paper presents an overview of a key component of the receiver, namely the combined equalizer and decoder.

There are two main issues in the robust detection of a packet in a wireless local area network (WLAN) system: noise and multipath.

Manuscript received August 26, 2002; revised November 19, 2002.

C. Heegard was with Wireless Networking Business Unit, Texas Instruments Incorporated, Santa Rosa, CA 95401 USA. He is now at 141 Stony Circle, Suite 130, Santa Rosa, CA 95401 USA (e-mail: heegard@nativei.com).

E. Rossin, retired, was with Wireless Networking Business Unit, Texas Instruments Incorporated, Santa Rosa, CA 95401 USA.

S. Coffey, S. Gummadi, and M. Wilhoyte are with the Wireless Networking Business Unit, Texas Instruments Incorporated, Santa Rosa, CA 95401 USA (e-mail: coffey@ti.com).

M. Shoemake is with the Wireless Networking Business Unit, Texas Intsruments Incorporated, Dallas, TX 75243 USA.

Digital Object Identifier 10.1109/JSAC.2002.807558

The main source of noise can be attributed to the radio frequency (RF) front-end or the "radio." The signals that impinge on the antenna at the receiver are exceedingly small and must be significantly amplified in order for the baseband processor to detect the message contained in the received signal. This amplification process introduces distortion on the signal, which for a properly designed radio, will consist mainly of noise that is modeled as additive white Gaussian noise (AWGN). In an ideal wireless transition system, as depicted in Fig. 1, the message to be transmitted m would be transformed into a baseband signal $x(t)$ belonging to the set of signals $\mathcal{X}$ known as the signal set, codeword set or codebook. This signal is modulated to the proper carrier frequency and transmit level by the transmit radio and broadcast via the transmit antenna. At the receiver, the signal that appears on the receive antenna is amplified and translated to baseband for processing by the baseband processor that performs the message detection (i.e., decision). The baseband processor, observing the signal $y(t) = x(t) + n(t)$ uses the knowledge of the transmitted signal set $\mathcal{X}$, along with the characteristics of the noise $n(t)$ in order to make an accurate decision. For example, in AWGN, an optimal receiver would find the signal in the known signal set $\mathcal{X}$ that is closest to the received signal $y(t)$ in terms of signal energy.

However, realistic indoor environments, which have many reflective surfaces, are modeled by a more involved process than that shown in Fig. 1. A significant amount of signal distortion can occur due to the phenomenon known as multipath. In multipath environments, the signal observed at the received antenna is the sum of a variety of reflections of the transmitted signal. These reflections have a host of signal gain and delay parameters so the the received signal is smeared, or filtered, by the multipath channel. This is shown schematically in Fig. 2.

The process of undoing the effects of multipath is known as equalization, while the processing to compensate for the code is known as decoding. These processes may be carried out separately, and there are several different known strategies for doing so. Each, however, has performance disadvantages. As discussed in more detail in the theory section of this paper, the equalization and decoding processes require different strategies.

The theoretical development required to consider combined equalization and decoding has long been known. The performance of such a receiver is optimal; the remaining problem is one of receiver complexity. A straightforward implementation of the optimal combined equalization and decoding structure is far beyond the bounds of practicality. Essentially, the combination of code plus multipath forms a "supercode" consisting

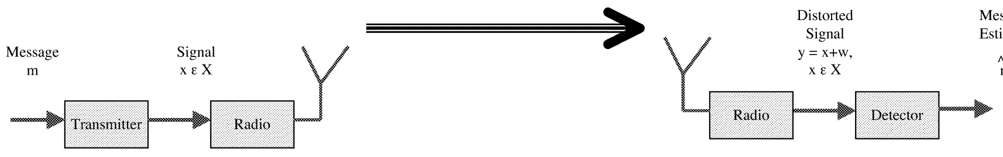


Fig. 1. Signal plus noise without multipath.

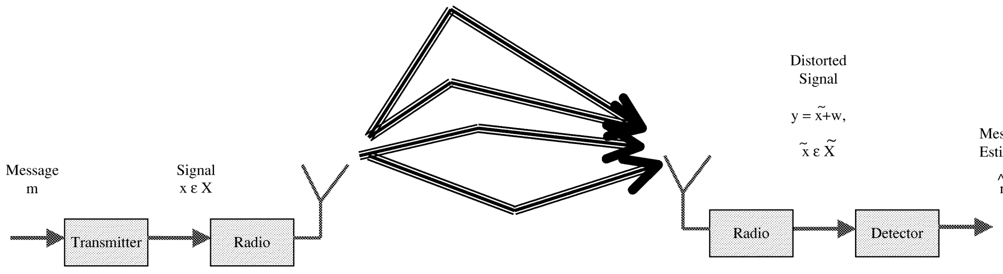


Fig. 2. Signal plus noise with multipath.

of a much larger number of states, and correspondingly larger complexity.

The approach taken in the ACX100 is to construct and decode the full combined equalization and decoding "supercode," but to achieve a practical solution by using the classical *Fano sequential decoding algorithm* [3]. The Fano algorithm was originally developed and extensively explored in the 1960s as a method for decoding convolutional codes containing very many states. The algorithm has at any given time one possible set of partially decoded data, which is continually altered and updated. The algorithm may backtrack and undo previous tentative decisions, as well as adding new tentative decisions. An ingenious set of update rules ensure that the Fano algorithm searches the data effectively. As there is at any time just one tentative set of partially decoded data, the complexity of the algorithm is very low.

Although the Fano algorithm was developed for convolutional codes, it may easily be adapted to decoding block codes, intersymbol interference (ISI), and in particular, any combination of block codes and ISI: these combinations can be thought of as producing time-varying convolutional codes.

In a recent (1998) special issue celebrating the 50th anniversary of the publication of the landmark paper of Shannon [4], the IEEE TRANSACTIONS ON INFORMATION THEORY published an article by G. D. Forney, Jr. and G. Ungerboeck titled "Modulation and Coding for Linear Gaussian Channels." In this paper, it is stated [5]:

"In view of [this list of] desirable properties, it is something of a mystery why sequential decoding has received so little practical attention during the past 30 years."

This paper describes how a sequential decoding approach to combined equalization and decoding is achieved. The paper considers three aspects of the problem: theory, simulation, and practice. In the theory section, an overview of the problem is presented and analyzed. In the simulation section, the performance of the approach is compared with alternatives. Finally, data gathered at our offices from actual hardware is presented. The data shows the robust nature of a combined equalizer decoder implementation. Finally, we end the paper with a few observations, conclusions and closing remarks.

II. THEORY OF COMBINED EQUALIZATION AND DECODING

Many important and practical codes used in wired and wireless communications system can be described via the notion of a finite-state machine (FSM). The notion of a FSM also models in a natural way *multipath distortion*, which is a major issue in wireless system design.

Data that is encoded and then sent across a channel in which there is multipath distortion, thus passes through two FSMs. The combined effect of this transformation is equivalent to that of another (composite) FSM. The aim of combined equalization and decoding is to build the receiver so that the composite FSM is decoded. This performs the entire receiving procedure in one step.

In this section, we review the various relevant facts on FSMs, codes, multipath, and suitable decoding algorithms.

A. FSMs, Trellises, and Trees

A FSM is characterized by a (finite) set of states $\mathcal{S}$, an input alphabet $\mathcal{X}$, and an output alphabet $\mathcal{Y}$, together with an *output rule* and a *next state* rule. The output rule $f(x_i, s_j) = y_k$ determines the output $y_k \in \mathcal{Y}$ when the FSM is in state $s_j \in \mathcal{S}$ and the input is $x_i \in \mathcal{X}$. The next state rule $g(x_i, s_j) = s_k$ determines the state at the next time unit $s_k \in \mathcal{S}$ given input x_i and current state s_j.

Two common ways of representing the action of a FSM in the context of channel coding are *trellises* and *trees*.

A *trellis* is a labeled, directed graph, in which each vertex has an associated time index. Vertices with the same time index are grouped together, and edges go only from a vertex at one time index to vertices at the next time index. The action of a state machine can be represented by a trellis by taking the set of vertices at a given time index to be the set of possible FSM states at that time index. If the FSM can pass from state s_i at time j to state s_k at time $j+1$, then the trellis at time j contains a directed edge from state s_i at time j to state s_k at time $j+1$. Various conventions are taken for the label attached to the edge; one straightforward convention is to label the edge with the FSM output corresponding to that state transition.

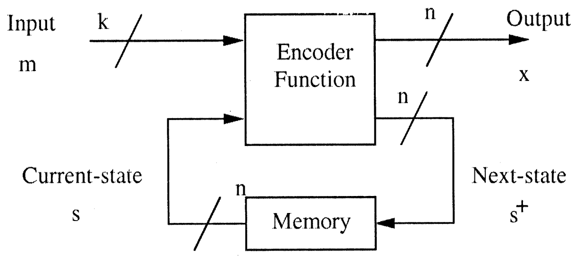


Fig. 3. BCC as a FSM.

A *tree* is a directed graph in which each vertex has at most one incoming edge. A tree representing an FSM has the same grouping of vertices into time indices as in the case of a trellis. In the case of a tree, there is a separate vertex for each possible sequence of FSM inputs. The edge connection and labeling convention is otherwise identical to the trellis.

In the case of both trellises and trees, the number of edges out of a state (vertex) equals the number of possibilities for the input at that time unit.

B. Codes as FSMs

1) Binary Convolutional Codes (BCCs) as FSMs: The most familiar example of an FSM representation of a code is the binary convolutional code BCC. In a BCC, binary data is encoded by passing the binary message stream through a linear, time invariant, filter with k binary inputs and $n > k$ binary outputs (see Fig. 3).

Taking the most common situation in which $k = 1$ and $n = 2$ and an finite-impulse response (FIR) encoder with memory ν, the BCC is a FSM in which the input alphabet $\mathcal{X} = 0, 1$, the output alphabet $\mathcal{Y} = \{00, 01, 10, 11\}$, the state consists of the past ν inputs, the output rule chooses the two output coded bits according to the FIR filter input and the past ν inputs. The next state rule is the simple one of $f(m_i, (m_{i-1}, \dots, m_{i-\nu})) = (m_i, m_{i-1}, \dots, m_{i-\nu+1})$, i.e., at the next time unit the state consists again of the last ν inputs.

For example, consider the four-state $(\nu = 2)$, $(k = 1, n = 2)$ BCC described by the equations

$$\begin{aligned} c_j^1 &= m_j \oplus m_{j-2} \\ c_j^2 &= m_j \oplus m_{j-1} \oplus m_{j-2} \end{aligned}$$

or, in terms of a FSM description

$$\begin{aligned} c_j^1 &= m_j \oplus s_j^2 \\ c_j^2 &= m_j \oplus s_j^1 \oplus s_j^2 \\ s_{j+1}^1 &= m_j \\ s_{j+1}^2 &= s_j^1 \end{aligned}$$

as shown in Fig. 4.

C. Block Codes as FSMs

Linear block codes (LBCs) may also be described via trellises and trees. The trellis description of block codes goes back to Forney in 1974 [6], and was the subject of much research in the 1990s [7], [8].

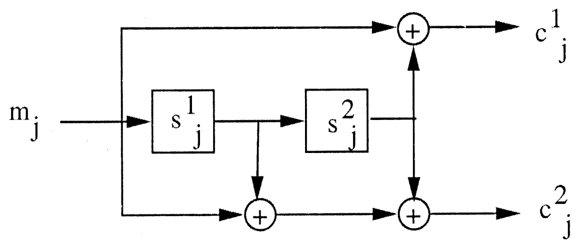


Fig. 4. An example of a four-state, rate 1/2, BCC.

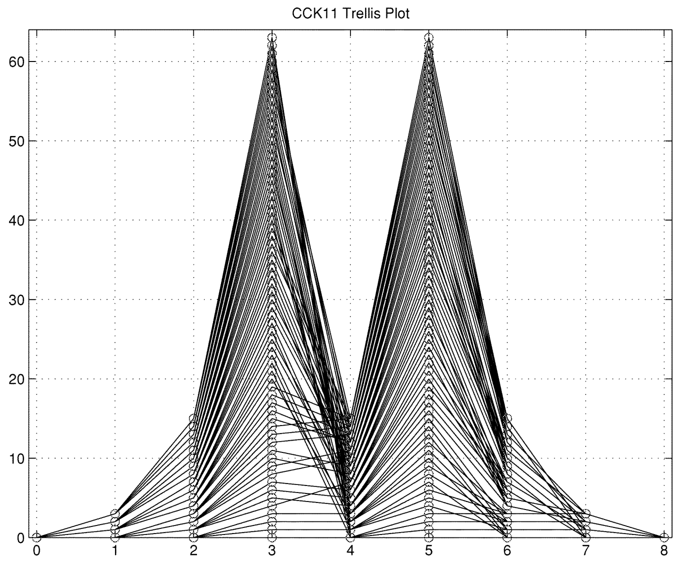


Fig. 5. The CCK-11 trellis.

In the case of LBCs, the number of states and the next-state update rule are more complicated than in the case of BCCs. One out of many equivalent ways of defining a trellis for a LBC is to take any parity check matrix H for the code, and to define the set of states at a given time index i to be the set of *partial syndromes of codewords* at that time index. Here, the partial syndrome of a codeword $(c_1, c_2, \dots, c_n)$ is defined as $s^{(i)} = c_1 h_1 + c_2 h_2 + \dots + c_i h_i$, where h_j is the jth column of the H matrix. Note that the nth "partial" syndrome s_n is the ordinary syndrome of the entire codeword; hence, the terminology. The trellis is then constructed by following the path of possible codewords: e.g., there is an edge between the state at time i, $s^{(i)} = c_1 h_1 + \dots + c_i h_i$, and the state at time $i + 1$, $s^{(i+1)} = c_1 h_1 + \dots + c_i h_i + c_{i+1} h_{i+1}$, for all codewords $(c_1, c_2, \dots, c_n)$. The edge is labeled by c_{i+1}.

In this method, the trellis may vary according to the H matrix that has been chosen. Efficient methods for choosing a good parity check matrix, i.e., of finding a minimal trellis for a LBC given any generator matrix or parity check matrix, are known [8].

Unlike the regular time-invariant structure of a BCC, the minimal trellis of an LBC has a number of states, edges and out degree that usually vary along the length of the block. The block code is described by a trellis that starts in a single state, expanding and contracting along the course of the blocklength, finally returning to a single state at the end of the block. The trellis description for a frame of LBC data is obtained by concatenating

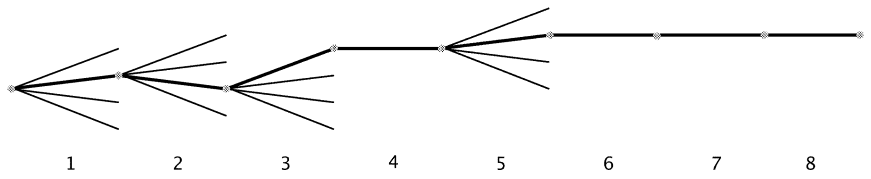


Fig. 6. The CCK-11 tree.

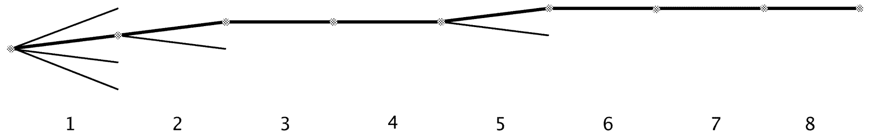


Fig. 7. The CCK-5.5 tree.

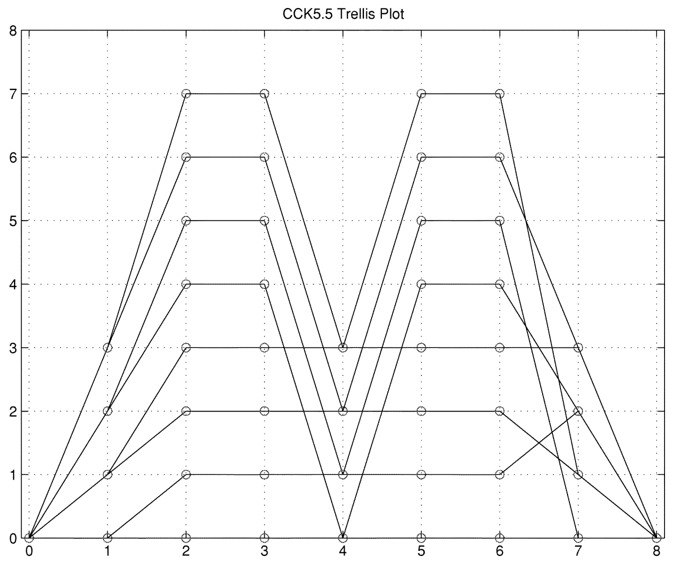


Fig. 8. The CCK-5.5 trellis.

the block code trellis into a series of trellises. The resulting concatenated trellis is a periodic trellis with a period that is equal to the blocklength.

The IEEE 802.11b standard specifies two LBCs of blocklength $n = 8$ for 11 and 5.5 Mb/s transmission, as described in [1], [2]. The higher rate code, CCK-11, has a trellis shown in Fig. 5. This code starts in single state and expands for three steps into 4, 16, and 64 states each with four-way branching $(R = 2)$. The trellis contracts to 16 states with one-way branching $(R = 0)$, expanding again to 64 states with four-way branching $(R = 2)$. The remaining three steps contract the number of states to 16, 4, and finally back to one-state with one-way branching $(R = 0)$. In terms of a tree description, a typical path is illustrated in Fig. 6. One can observe that while progressing through the tree, the information is distributed according to a pattern of information rates $R = [2, 2, 2, 0, 2, 0, 0, 0]$ bits; this pattern is repeated as a frame of data is encoded or decoded.

The lower rate code is the CCK-5.5 LBC. This code has a trellis shown in Fig. 8. In this case, the tree has a $R = [2, 1, 0, 0, 1, 0, 0, 0]$ information rate pattern (Fig. 7).

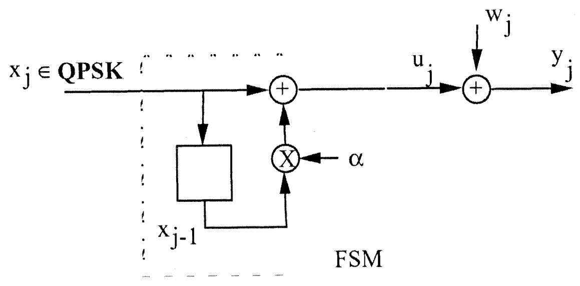


Fig. 9. The two-term QPSK FIR-ISI channel.

D. Multipath Channels as a FSM

FSMs can also be used to describe a FIR ISI channel with AWGN. The model is very important for wireless systems, such as IEEE 802.11b, since this accounts for *multipath distortion*. In this model, the output of the channel is described as a linear FIR filtering of the input signal followed by a memoryless noise channel [9], [10]. When the input signal set, called the *signal constellation* $\mathcal{X}$, is a *finite* set, then the FIR filter can be expressed as a FSM. This realization is the basis for the combined equalizer/decoder described in this paper.

The FIR-ISI channel is described in terms of:

1) a finite input signal set or constellation $\mathcal{X}$;
2) an FIR impulse response, often described by a polynomial $h(z) = h_0 + h_1 z^{-1} + \cdots h_\mu z^{-\mu}$ of degree μ (the number of terms is $\mu + 1$);
3) a statistical model of the noise sequence w_j.

The FIR-ISI channel is, thus, composed of two parts: the FSM component (items 1 and 2), followed by the additive noise model (item 3).

The FSM maps the input sequence $\ldots x_{i-1}, x_i, x_{i+1}, \ldots$ to the output sequence $\ldots u_{i-1}, u_i, u_{i+1}, \ldots$ via

$$u_j = h_0 x_j + h_1 x_{j-1} + \cdots + h_\mu x_{j-\mu}. \qquad (1)$$

As a FSM, this is fully analogous to the BCC discussion in Section II-B. The additive noise then adds w_j, giving the overall FIR-ISI channel

$$y_j = h_0 x_j + h_1 x_{j-1} + \cdots + h_\mu x_{j-\mu} + w_j. \qquad (2)$$

As an example (see Fig. 9) consider an FIR-ISI channel model with a channel input $x_j \in$ QPSK taking on values in the quadrature phase-shift keying (QPSK) signal set, QPSK $= \{1 +$

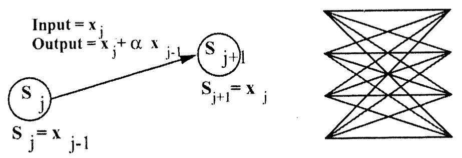


Fig. 10. The trellis for the two-term QPSK FIR-ISI channel (see Fig. 9).

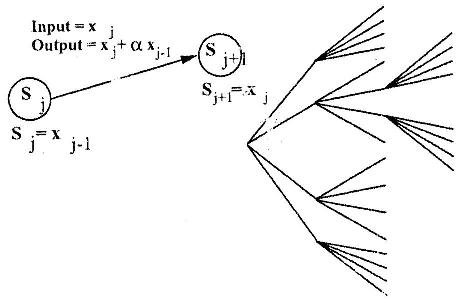


Fig. 11. The tree diagram for the two-term QPSK FIR-ISI channel (see Fig. 9).

$i, -1+i, -1-i, 1-i\}$, with two multipath terms, $u_j = x_j + \alpha \cdot x_{j-1}$, where α is a complex multipath gain, and additive noise consisting of independent, identically distributed complex Gaussian random variables w_j with mean 0 and variance $2\sigma^2 = N_0$. (Thus, the real and imaginary parts of w_j are iid Gaussian random variables with mean zero and variance $N_0/2$).

In this example, the input at time j consists of the QPSK signal x_j, the state consists of the QPSK signal from the previous time unit, x_{j-1}, and the output of the FSM is given by u_j. Thus, there are four states at each time unit other than the first, corresponding to the four possibilities for x_{j-1}. The tree has a four-way branching structure at all time units, corresponding to the four possibilities for x_j. The trellis section at all time units other than the first consists of 16 branches, to connect every possible state at time $j-1$ to every possible state at time j.

A schematic for the trellis section with 16 branches $(x_j, x_{j-1} \in \text{QPSK})$ is shown in Fig. 10 and a tree description is depicted in Fig. 11.

E. Combined Code Plus Multipath as a FSM

It is an easy exercise to prove that the cascade of two FSMs is itself a FSM where the number of states is at most equal to the product of the number of states of the two FSMs. For example, consider the cascade of the convolutional code in Fig. 4 with the FIR-ISI channel in Fig. 10. We arrive at the following FSM:

$$\begin{aligned} u_j &= x_j - \alpha \cdot x_{j-1} \\ x_j &= \text{QPSK}\left(c_j^1, c_j^2\right), \quad x_{j-1} = \text{QPSK}\left(c_{j-1}^1, c_{j-1}^2\right) \\ c_j^1 &= m_j \oplus m_{j-2}, \quad c_j^2 = m_j \oplus m_{j-1} \oplus m_{j-2} \\ c_{j-1}^1 &= m_{j-1} \oplus m_{j-3}, \quad c_{j-1}^2 = m_{j-1} \oplus m_{j-2} \oplus m_{j-3} \end{aligned}$$

where the QPSK mapping is given in Table I. The FSM model has a binary input m_j, a complex output u_j and a state $s_j = (m_{j-1}, m_{j-2}, m_{j-3})$. Thus, there are eight states, corresponding to the eight possibilities for $m_{j-1}, m_{j-2}, m_{j-3}$. The tree has two-way branching, corresponding to the two

TABLE I
QPSK MAPPING

Code Symbol c^1, c^2	Signal $x = \text{QPSK}(c^1, c^2)$
0 0	$+1\ +i$
1 0	$-1\ +i$
1 1	$-1\ -i$
0 1	$+1\ -i$

possibilities for the input m_j; note that the composite FSM inherits the two-way branching of Fig. 4 rather than the four-way branching of Fig. 10. The trellis section at time j consists of eight states at time j connected to eight states at time $j+1$, with two edges leaving every state.

It is interesting to note that the number of states of the cascade, in many practical situations, is less than the upper bound. Generalizing the example above, a ($k=1$, $n=2$) BCC with state length ν has 2^ν states, an FIR-ISI channel with $\mu+1$ terms and QPSK inputs has $4^\mu = 2^{2\mu}$ states. If the $n=2$ binary outputs of the BCC are mapped onto QPSK symbols in a one-to-one correspondence and then used as the input to the FIR-ISI channel, then the cascade has $2^{\nu+\mu}$ states, rather than $2^{\nu+2\mu}$.

When a LBC is used over a multipath channel, the combination of encoding and channel filtering can be described by a combined trellis. In this case, the number of states of the combined trellis will vary periodically with a period equal to the blocklength. In general, the number of states at the beginning/ending of a code block will be more than 1 reflecting the ISI that will force a dependency between the blocks of the LBC. However, from a tree point of view, the branching degree $(L = 2^R)$ will be the same. For example, the diagrams in Figs. 6 and 7 will remain the same with multipath filtering of the LBC coded signals.

F. Detection/Decoding of FSMs Over a Memoryless Channel

Many different approaches can be taken to decoding the composition of code and multipath. These possible approaches trade performance and complexity. In this section, we review the various possibilities, seeking to show how each fits inside the overall context of the FSM descriptions of the overall code-plus-multipath system.

In Section II-F1, we consider the optimal decoding of the composite system, via the Viterbi algorithm (maximum-likelihood (ML) sequence estimation). At the other end of the complexity spectrum, we consider decision feedback equalization followed by decoding of the code alone. Finally, we consider the use of the Fano sequential decoding algorithm.

1) The Viterbi Algorithm: Viterbi discovered an optimal method for decoding BCCs over memoryless channels [7], [11]–[13]. Shortly after the discovery of the Viterbi algorithm it was realized [9], [10] that the Viterbi algorithm could be applied to other communications problems where the transmission system involves a FSM description, including the FIR-ISI model for multipath distortion. The discovery that the Viterbi algorithm also applied to LBCs also dates, as noted earlier, to approximately the same time. More generally, the Viterbi algorithm applies to, and is an optimal decoding algorithm for, any code or model transmitted over a memoryless channel. The decoding complexity of Viterbi decoding is dominated by the maximum number of states that occurs at any stage of the trellis.

In one sense, this observation solves the problem of combined equalization and decoding: the optimal solution is to form the composite FSM and to decode it via Viterbi's algorithm. The sole problem with this approach is that the complexity can be prohibitive. For example, in the context of IEEE 802.11b systems, in decoding a CCK-11 packet that has been transmitted over a multipath channel with memory 8, the decoding algorithm has to handle $2^{14} = 16\,384$ states, which is unrealistic in almost all applications.

2) Decision Feedback Equalization: At the low end of complexity is the decision feedback equalizer (DFE). This method is much simpler than Viterbi detection, but can suffer considerable performance loss, especially when dealing with data that has been coded.

In the usual DFE approach, the equalizer attempts to compensate for the effect of the multipath distortion only, and then (if the data has also passed through a channel code) feeds the equalized symbols to the channel code decoder.

To compensate for the effect of the multipath distortion, the receiver recursively estimates the current symbol, and from this and previous estimates, plus knowledge of the channel-impulse response $h_0 + h_1 z^{-1} + \cdots + h_\mu z^{-\mu}$, computes what the multipath distortion impact at the next received symbol will be. This estimated multipath distortion is then subtracted from the next received symbol, leaving the actual symbol plus additive noise. The receiver makes its best estimate of that symbol and the process repeats.

Note that the DFE works well as long as each individual symbol estimate is correct: if all previous symbol estimates were correct, the next decision is "locally optimum." However when an incorrect estimate occurs, the DFE miscompensates at the next symbol, i.e., forms

$$h_0 x_{i+1} + h_1 (x_i - \hat{x}_i) + w_{i+1}$$

and this error is present for a window of μ consecutive symbols. This makes it more likely that another decision error occurs soon after the first, which in turns triggers more miscompensation, and so on. This leads to the "error propagation" problem that is often associated with the DFE method.

The problems of a separate DFE and decoder illustrate the central dilemma of building a receiver for a channel corrupted by both multipath distortion and noise. On the one hand, the channel code is designed to compensate for the additive noise, and is necessary in those conditions in which the noise level is high enough that individual decisions on symbols are unreliable. It is well known that decoders operate best when "hard decisions" are not made, but rather when they are fed "soft decision" information. On the other hand, the DFE operates precisely by making hard decisions on a symbol-by-symbol basis; it needs to do this in order to be able to subtract out the multipath at subsequent decision points. Thus, the two approaches are not well suited to each other. In practice, the approach of DFE followed by channel decoder works only when the noise level is so low that the channel code is essentially not needed.

G. Search-Based Decoding

After the advent of BCCs, much effort was expended in developing methods for decoding errors at the receiver in a BCC encoded data stream. Much of the early successful work involved tree searching algorithms, an area that is now commonly known as sequential decoding. Many variants of sequential decoding have been developed, including the stack algorithm and Fano decoding.

In all of these algorithms, the tree representation of the FSM is used. In a sequential decoding algorithm, the decoder attempts to follow promising paths through the code tree, and continually monitors the quality of the path currently being followed. As long as the path appears to be correct, as judged via some criterion, the search continues along that path. When a wrong turn occurs, the possible extensions to the current path all become substantially different to the received signal, and the decoder should soon decide that the path is probably incorrect. When this happens, another path is chosen.

The precise implementation of this general idea varies among the different sequential decoding algorithms. In the case of stack algorithms, a list of possible paths is maintained and an alternative path is considered once the current path looks problematic. The Fano algorithm is implementationally simpler, though conceptually slightly more involved. The Fano decoder maintains only one path; when the current path is judged to be incorrect, the decoder is allowed to step back one time unit on that path. The algorithm has an associated decoder action flowchart that ensures that the decoder can never go into an infinite loop.

1) Advantages and Disadvantages of Sequential Decoding: The history of sequential decoding provides an interesting example of the role of varying applications and requirements on communications theory, and the role of fashion in communications theory in influencing communications practice. The various algorithms in the sequential decoding family were throughly studied in the 1960s, and much insight into their properties was developed. The algorithms fell into disuse, however, after the advent of Viterbi decoding in the early 1970s.

The main differences between the two methods can be summed up as follows.

1) Viterbi decoding takes a deterministic time, while sequential decoding takes a variable amount of time; the computation time of a sequential decoder rises in noisier conditions.
2) Viterbi decoding is ML, while sequential decoding is not guaranteed, even when decoding completes, to choose the most likely path.
3) Sequential decoders have much smaller complexity than Viterbi decoders.

The main situation in which Viterbi decoders have an advantage is in the first item; in practice the nonmaximum-likelihood results from a sequential decoder are almost always due to excessive computation time leading to buffer overflow. For a communication system involving continuous data streaming, however, this feature of variable computation time and buffer overflow is serious as without further mechanisms to restore synchronization, the system will continue to make errors; typically the encoder needs to send known retraining data periodically. For such a system, therefore, there are attractions to choosing a code that has a small number of states by design, and decoding it with the Viterbi algorithm.

Note that this scenario has little in common with the situation in IEEE 802.11b wireless LANs. First, the system is packet-based, with each successive packet standing on its own as a coded entity; thus, known retraining is not an issue. Secondly, the codes standardized for IEEE 802.11b are not especially strong, particularly the mandatory CCK-11 and CCK-5.5 codes. Thus, the receiver operating points are relatively far from the "cutoff rate" region, i.e., the channel conditions are relatively quiet and the average amount of computation is low. Thirdly, the complexity of Viterbi decoding of the composite code-plus-multipath system is excessive; as we have no control over the amount of multipath introduced by the channel, we do not have direct control over the number of states required by such a Viterbi decoder. The complexity of the sequential decoder is essentially independent of the complexity of the composite code-plus-multipath channel. Finally, the system needs to be able to decode a FSM that will not be known until reception of the packet, as the multipath will not be known until then. This implies that a configurable decoder structure is needed. The Fano algorithm is well suited to this.

2) Path and Branch Metrics: A memoryless channel is often described in terms of a conditional probability

$$p(y|x)$$

that describes the distribution on the observation variable y conditioned on the input variable x. The channel is memoryless if the output at time j, y_j, is independent of the channel's inputs and outputs at other times given the input at time j, x_j. In search based decoding, the received sequence is sequentially compared with a path in the tree corresponding to a state sequence of the FSM. Typically, a *branch metric* is used as a measure of the closeness of the fit between the received symbol and the transition on a given branch. A typical branch metric for a memoryless channel is the *log likelihood metric*

$$B(m, s; y) = -\log(p(y|x))$$

where x is the FSM output that corresponds to the input m and the state s as in Fig. 3. In the case of AWGN, the metric can take the form

$$B(m, s; y) = ||x - y||^2$$

which describes the energy difference between the channel input x and the channel output y.

In sequential decoding, a modified metric is used. The modification enables the decoder, in effect, to compare paths at different depths in the tree. The most usual form of metric is of the form

$$\begin{aligned} B_{\text{Fano}}(m, s; y) &= \log\left(\frac{p(y|m, s)p(m)}{p(y)}\right) \\ &= \log(p(y|x)) - \log(p(y)) - R \end{aligned}$$

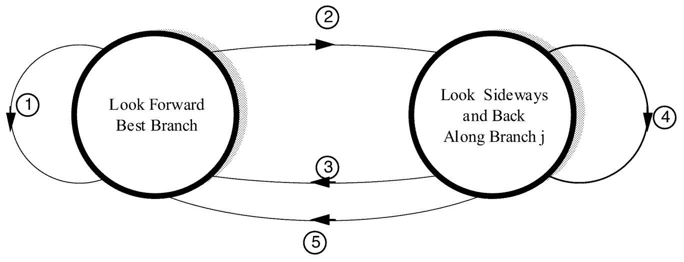


Fig. 12. The Fano algorithm.

or

$$B_{\text{Fano}}(m, s; y) = -(\log(p(y)) + ||x - y||^2 + R)$$

where 2^R is the number of possible inputs m (*i.e.*, for "random" data, each input has uniform probability 2^{-R}) and $p(y)$ is the marginal distribution for the channel output. For example, for the BCC encoder in Fig. 3, $R = k$ while for the FIR-ISI channel $R = \log_2(|\mathcal{X}|)$ ($R = 2$ in Fig. 9). This metric is known as the *Fano metric* and is explained by Massey in [14].

3) Fano's Sequential Decoding Algorithm: While Fano's algorithm has been widely described, we briefly describe a variation of the algorithm that is novel and compactly presented. The basic algorithm is presented in Fig. 12 and Table II. The low implementation complexity can be seen from the two-state five-branch description of the algorithm. Part of the simplicity and efficiency of this approach involves the notion of a "sideways" look. The "look back" part of Fano's procedure is agumented by the consideration of a sideways move. At any given time, it is an invariant that the path-metric, $\mathcal{M}_{\text{path}} \geq 0$.[1] At a given state of the tree, the L branches are sorted according to the branch metrics $\mathcal{M}_{BR(1)} \geq \mathcal{M}_{BR(2)} \geq \cdots \geq \mathcal{M}_{BR(L)}$. Note that with the Fano metric some branch metrics will be negative in general. The *best branch* is the one identified with $\mathcal{M}_{BR(1)}$. Also note that the number of branches, L, is constant for a BCC such as that used in PBCC while it is periodically varying for a LBC such as CCK, as described in Section II-C. The information rate for the Fano metric $R = \log_2(L)$. A sideways move can occur when the algorithm is looking back along a branch that is not the last (*i.e.*, not $\mathcal{M}_{BR(L)}$) as described in Step 3, Fig. 12, and Table II.

III. A PRACTICAL COMBINED RECEIVER

This method has been used in TI's receiver design for IEEE 802.11b wireless local area networks. It has proven to be an effective receiver with a practical implementation complexity.

In the realization of the technique in practice, an estimate of the channel response must be made at the receiver and given to the search based decoder for the detection process. In a packet based system, the estimate can be made on the preamble of the packet and presented to the decoder before the data portion of the packet is to be processed. In addition, an adaptive equalizer

[1]In the usual description of Fano's Algorithm, two values are maintained, the *path-metric* and the *threshold*; the invariant is that the path-metric must always be as large as the thresold. In practice, it is more convenient to maintain only the difference between the path metric and the thresold. This is $\mathcal{M}_{\text{path}}$.

TABLE II
FANO ALGORITHM STEPS

Branch	Condition	Action
①	Forward Path is Good $(\mathcal{M}_{path} + \mathcal{M}_{BR(1)} \geq 0)$	Move Forward and Update Path Metric $\mathcal{M}_{path} \leftarrow \begin{cases} \mathcal{M}_{path} + \mathcal{M}_{BR(1)} - \Delta \text{ [tighten]} & if \begin{array}{l}\mathcal{M}_{path} + \mathcal{M}_{BR(1)} \geq \Delta \\ and\ \mathcal{M}_{path} < \Delta,\end{array} \\ \mathcal{M}_{path} + \mathcal{M}_{BR(1)} & o.w. \end{cases}$
②	Forward Path Not Good	Stay
③	Backward Path is Good and Sideways Path is Good $\left(\begin{array}{c}\mathcal{M}_{path} \geq \mathcal{M}_{BR(j)} \\ and\ j < L\ [not\ last\ branch] \\ and\ \mathcal{M}_{path} + \mathcal{M}_{BR(j+1)} \geq \mathcal{M}_{BR(j)}\end{array}\right)$	Move Sideways and Update Path Metric $\mathcal{M}_{path} \leftarrow \mathcal{M}_{path} + \mathcal{M}_{BR(j+1)} - \mathcal{M}_{BR(j)}$
④	Backward Path is Good and Sideways Path is Not Good $\left(\begin{array}{c}\mathcal{M}_{path} \geq \mathcal{M}_{BR(j)} \\ and\ (j = L\ [last\ branch]\ or \\ \mathcal{M}_{path} + \mathcal{M}_{BR(j+1)} < \mathcal{M}_{BR(j)})\end{array}\right)$	Move Backwards and Update Path Metric $\mathcal{M}_{path} \leftarrow \mathcal{M}_{path} - \mathcal{M}_{BR(j)}$
⑤	Backward Path is not Good $(\mathcal{M}_{path} < \mathcal{M}_{BR(j)})$	Stay and Loosen Path Metric $\mathcal{M}_{path} \leftarrow \mathcal{M}_{path} + \Delta$ [loosen]

can be used to maintain the validity of the estimated FIR equalizer impulse response over the duration of the entire packet.

A. *High-Level Outline of Receiver*

1) An estimate of the impulse response $h(z)$ is estimated from the packet preamble.
2) The impulse response and the trellis code description is presented to the decoder.
3) The search based decoder detects the data based on the impulse response and the trellis code.
4) An adaptive equalizer is used before the decoder to maintain the channel model, $h(z)$, presented to the decoder.

These steps are implemented as follows.

Step 1) The incoming received samples are fed into an adaptive FFE/DFE structure. The feedforward equalizer (FFE) is a fractionally $(T/2)$ spaced equalizer, designed to eliminate the precursor ISI [15]. The DFE is T-spaced. The FFE and DFE are both trained on the preamble, via a standard least-mean-square (LMS) algorithm. As the preambles in IEEE 802.11b are relatively generous in length (192 μs for the mandatory long preamble, 96 μs for the optional short preamble) it is easy to get convergence.

At the end of the preamble, the FFE and DFE coefficients are frozen.

Step 2) Presentation of data plus model to the decoder. The key point is that the data to be presented to the decoder is tapped off after the FFE but without being filtered by the DFE. The FFE block filters the received signal to eliminate the precursor ISI; assuming it does so, the resulting system has the form of (1), in which the coefficients $h_1, \ldots, h_\mu$ embody the postcursor ISI.

The purpose of the DFE block is to estimate these ISI coefficients. In the normal operation of a DFE, the data is filtered by the transfer function $1/h'(x)$ and, in the absence of decision and channel estimation errors, this cancels the postcursor ISI exactly. It follows that the idealized DFE transfer function satisfies $h'(x) = h_0 + h_1 x + \cdots + h_\mu x^\mu$. In practice, the receiver should take the adaptively computed coefficients $\tilde{h}_1, \ldots, \tilde{h}_\mu$ as its best estimate of the coefficients $h_1, \ldots, h_\mu$. These coefficients are the ones fed to the decoder.

In this approach, the function being carried out by the adaptive computation of the DFE is *channel identification* rather than *equalization*; as noted by Benedetto *et al.* [16, Ch. 8], these two functions are in form very similar.

Step 3) The decoder forms the composite trellis corresponding to the concatenation of the code (CCK 5.5, CCK 11, PBCC 5.5, PBCC 11, or PBCC 22) with the estimated multipath distortion channel given by the coefficents in Step 2. The resulting trellis is decoded via the Fano algorithm.

IV. SIMULATION RESULTS

A. *Channel Model*

Consider the following discrete time channel model. The input to the channel is given by complex baseband signal represented by x_n, where n is the time index (samples). The x_n's are the output of the channel coder. The output of the channel is represented as y_n. For a multipath that spans $\mu + 1$ terms, the received signal can be represented as

$$y_n = \sum_{i=0}^{\mu} h_i x_{n-i} + w_n$$

where h_i is the complex scalar representing the gain and phase of path i and the w_n's form a sequence of complex iid Gaussian random variables with mean zero and variance $2\sigma^2$.

For each path, the gain is given by $|h_i|^2$ and the phase is given by the angle of the complex gain h_i.

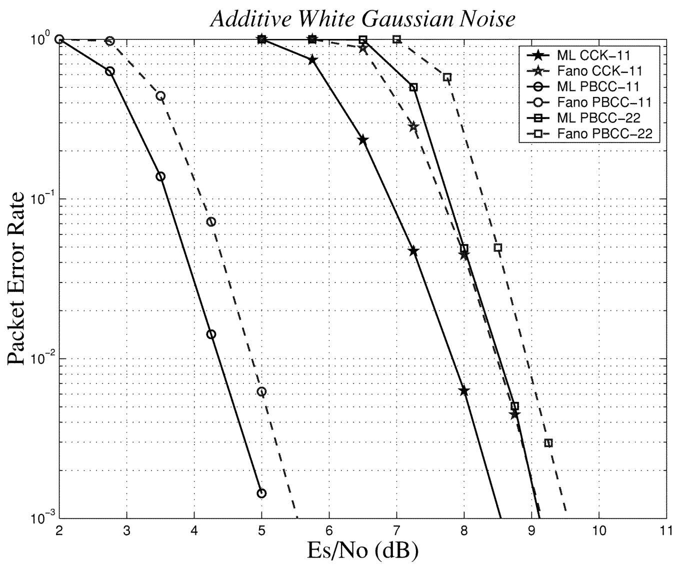


Fig. 13. AWGN performance.

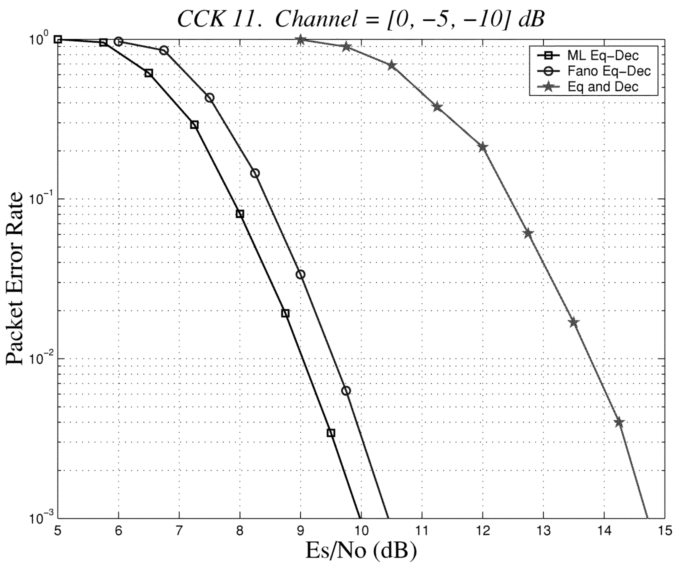


Fig. 14. Multipath performance.

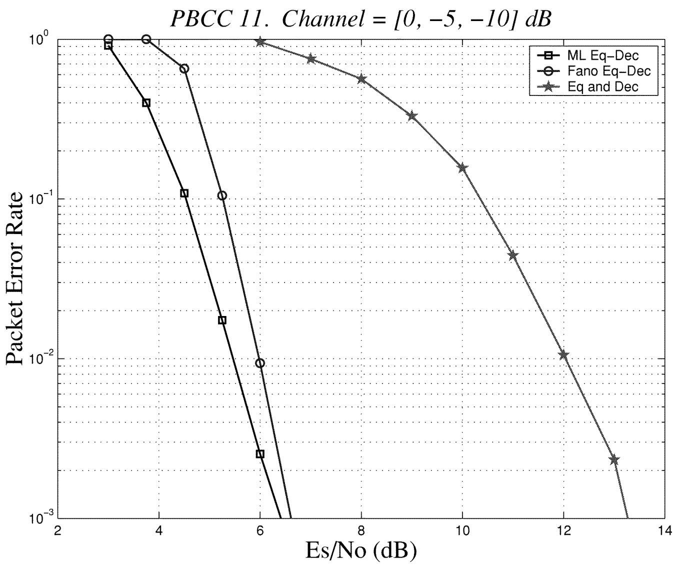


Fig. 15. Multipath performance.

In each simulation, the path with least delay is chosen to be the strongest and is normalized to 0 dB. The gain of other paths are specified as the fraction of power relative to the first path. Once the gains of all the paths are chosen they remain constant for every instantiation of the channel. The phase is varied for each instantiation of the channel and is chosen to be a uniform random variable in the range $[0, 2\pi)$.

The signal-to-noise ratio (SNR) of a received packet is calculated as

$$\mathrm{SNR} = 10\log_{10}\frac{\sum |h(i)|^2}{2\sigma^2}.$$

The parameter σ^2 is set to provide the desired SNR.

B. Performance Results

In this section, we provide some simulation results that highlight the performance of the Fano algorithm in various channel conditions. We compare the performance of the following three decoding methods: 1) ML joint equalization decoding; 2) sequential joint equalization decoding; and 3) equalization followed by decoding. In all cases, perfect knowledge of channel is assumed. In each simulation, packets with a payload of 1000 B are transmitted and the packet error rate is measured as a function of SNR. The performance is measured at a packet error rate of 10^{-2}.

Fig. 13 shows the performance in AWGN (no multipath) of ML decoder and the sequential decoder for the three modulation modes CCK-11, PBCC-11, and PBCC-22. It should be noted that in absence of multipath, combined equalization decoding, and equalization followed by decoding are identical as there is no equalization that needs to be done. It can be seen that the loss by using the sequential decoder is typically a fraction of a decibel (less than 0.5 dB) and is constant across the modulations.

Figs. 14–19 compare the performance for the three decoding methods for CCK-11, PBCC-11, and PBCC-22, respectively, in a multipath channel specified in Table III.

It should be noted that the complexity and memory requirements of ML joint equalization decoding makes the simulation of a channel model with more than 3 rays difficult.

Clearly, the Fano algorithm performs well across the range of modes.

V. Results in Practice

A. Measured Performance

At Texas Instruments in Santa Rosa, CA, indoor range/rate testing is performed at 21 test locations as shown in Fig. 20.

A fixed station or access point is placed on a shelf mounted to a wall labeled “AP” in the figure. A roaming station is placed at all test locations and throughput is measured and recorded. Multiple test locations at the same distance are used to get an average range/rate performance under varying channel conditions. The test locations vary in distances from 17 to 129 ft and their distribution over distance is as in Table IV.

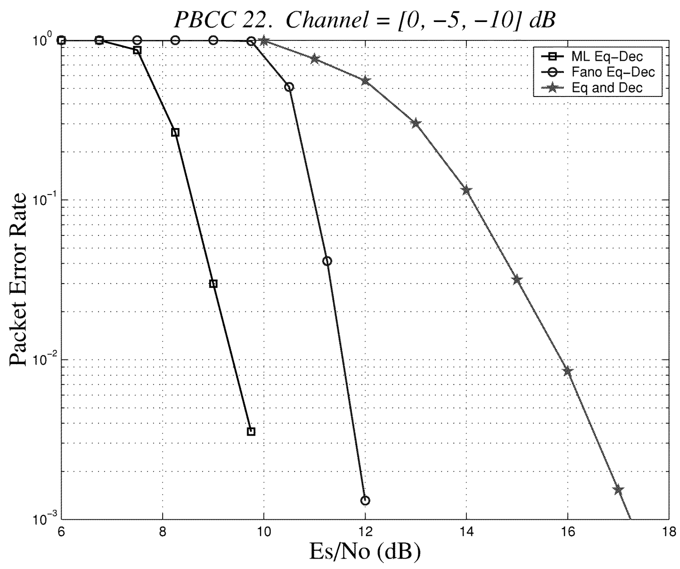


Fig. 16. Multipath performance.

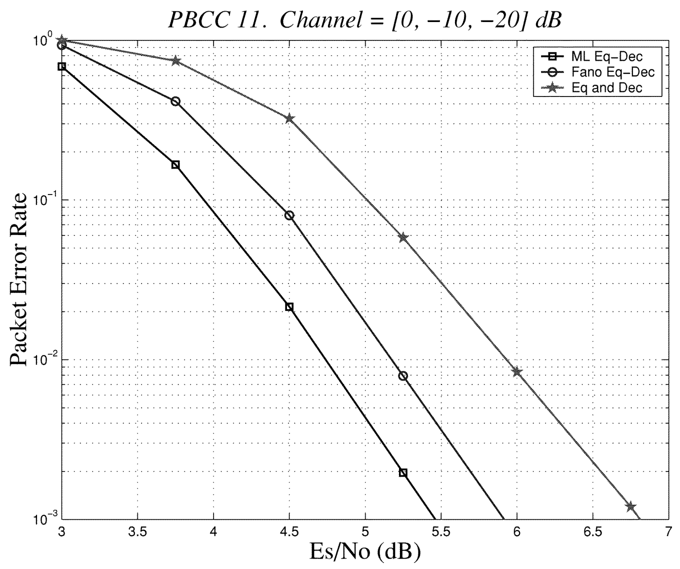


Fig. 18. Multipath performance.

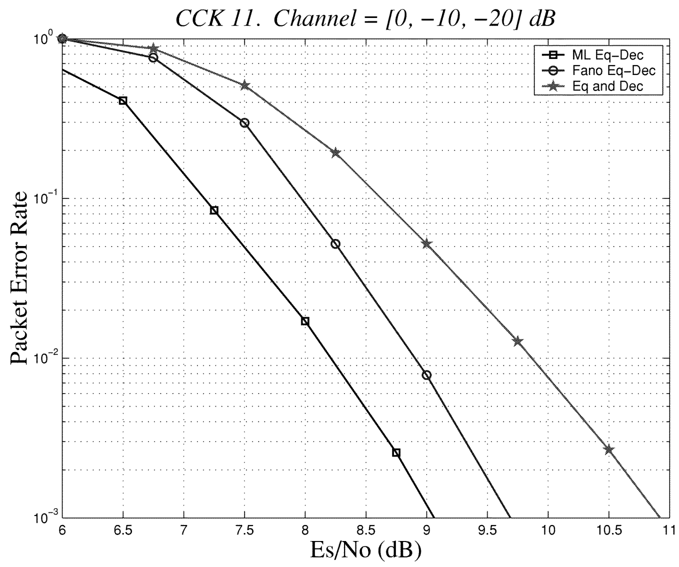


Fig. 17. Multipath performance.

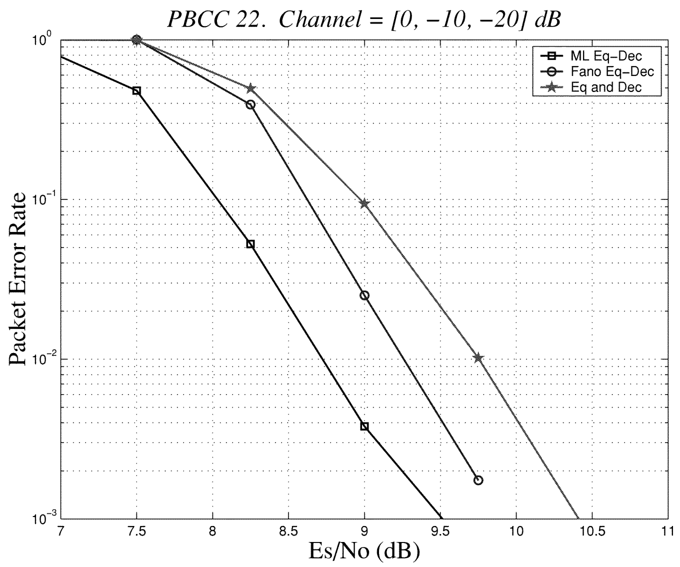


Fig. 19. Multipath performance.

1) Summary of Experimental Results: Table V shows a plot of average net rate (throughput) versus range for two 802.11b stations operating in an ad hoc network for cards based on the algorithm discussed in this paper versus two popular competitors. This plot shows consistent throughput advantage for the algorithm presented here over the alternatives at all ranges, with the advantage increasing with increasing range.

A rough measure of the average throughput delivered to a station card at any point within 129 ft can be obtained by forming a weighted sum of the throughput at each point on the curve in Fig. 23, that is

$$R_{\text{ave}} = \frac{1}{d_N^2} \sum_{k=1}^{N} \left(d_k^2 - d_{k-1}^2\right) \cdot r_k$$

where $(d_1, \ldots, d_6) = (17, 33, 67, 83, 116, 129)$ and the r_k's are the rates achieved at these distances.

2) Test Repeatability: The concern of any test fixture is to ensure repeatability to validate the test results at any given time.

TABLE III
MULTIPATH TERMS

delay (in samples)	0	1	2
gain (in dB)	0	-5	-10

There are two key areas of concern in this regard: *signal interference* and *sensitivity to orientation*.

Since testing is conducted in the open environment and not in an anechoic chamber, the test environment is subject to interference from other transmitters in the 2.4-GHz ISM band. To mitigate the effects of interference, testing is conducted in the middle of the night where interference is minimized. Scanning for other 802.11 transmitters is also conducted before testing occurs to ensure no other 802.11 network is active during the time testing is taking place.

Given the fading conditions, the test is also sensitive to placement of the roaming station including its orientation. To mitigate this, a slow moving turntable is used to rotate the station while

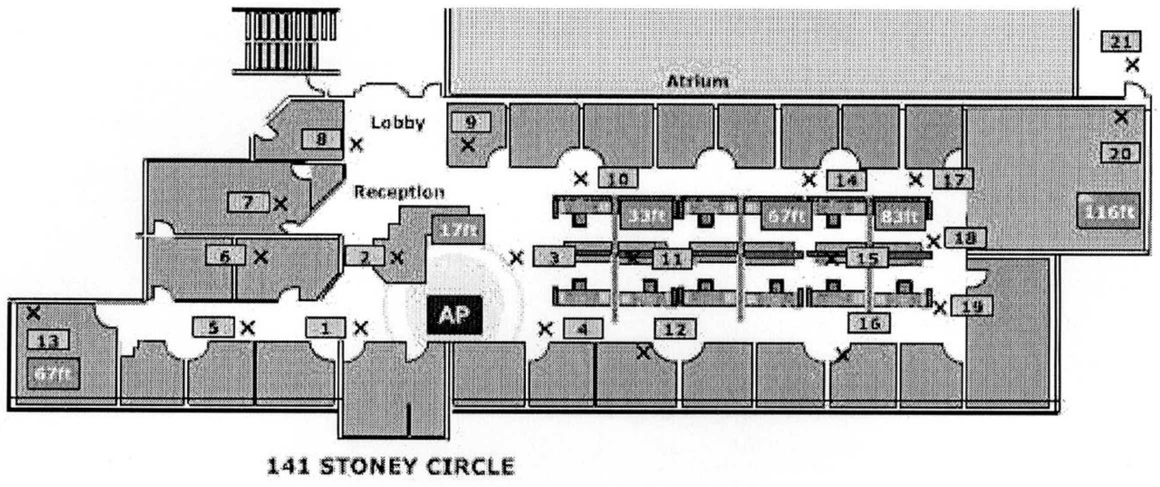


Fig. 20. Test locations at Texas Instruments, Santa Rosa, CA.

TABLE IV
TEST SETUP DISTANCES

Distance (ft)	Locations
17	1-4
33	5-12
67	13-16
83	17-19
116	20
129	21

TABLE V
AVERAGE RATE ADVANTAGE OF PBCC-22

Distance	PBCC-22	TI CCK-11	I CCK-11	A/O CCK-11
17	6.21	4.72	4.22	3.6
33	6.16	4.69	4.13	3.5
67	6	4.67	3.5	3.29
83	5.93	4.65	3.35	3.35
116	5.9	4.53	2.24	2.1
129	5.07	3.67	1.44	0.44
Average	5.8	4.4	2.6	2.3
PBCC-22 Advantage		30.8%	119.9%	151.6%

throughput measurements are being conducted. This ensures that the receiver moves through local flat fades that are present in the environment, as well as moving the antenna through all possible angles. Markers are also placed on the floor to ensure the same test location is used for all tests.

3) Test Data Collection: Throughput is measured using a readily available tool often used by customers of WLAN called WAR FTP which is easily downloadable from a web location. Version 1.65 is chosen for the data presented in this paper. The fixed station or AP is configured as the "server" and the roaming station is configured as the "client." At each test location, puts to and gets from the server are conducted on a 10-MB file consisting of random data. Three measurements of puts and gets are taken at each location. Table VI provides a partial example data set for 22 Mb/s.

The data is then averaged over location and converted to mega-bits-per-second (Mb/s) as shown in Table VII. The data is then averaged over range to produce the final rate versus range curve as shown in Fig. 23.

TABLE VI
EXAMPLE DATA RECORD: ACX100 PBCC-22, LONG PREAMBLE

Distance feet	Location	put kBytes/sec	get kBytes/sec
17	1	858.47	700.72
17	1	856.69	699.53
17	1	853.15	692.49
17	2	865.68	701.91
17	2	853.15	698.35
17	2	849.46	698.35
17	3	853.15	700.72
17	3	860.26	699.53
17	3	863.87	699.53
17	4	853.15	698.35
17	4	858.47	686.72
17	4	849.46	691.33
33	5	811.05	691.33
33	5	856.69	699.53
33	5	854.92	695.99
⋮	⋮	⋮	⋮

TABLE VII
AVERAGE DATA RECORD: ACX100 PBCC-22, LONG PREAMBLE

Distance feet	Location	put Mbps	get Mbps
17	1	6.85	5.58
17	2	6.85	5.60
17	3	6.87	5.60
17	4	6.83	5.54
33	5	6.73	5.56
33	6	6.70	5.57
33	7	6.68	5.60
33	8	6.81	5.57
33	9	6.82	5.58
33	10	6.79	5.58
33	11	6.79	5.60
33	12	6.71	5.52
67	13	6.81	5.59
67	14	6.39	5.41
67	15	6.73	5.50
67	16	6.30	5.25
⋮	⋮	⋮	⋮

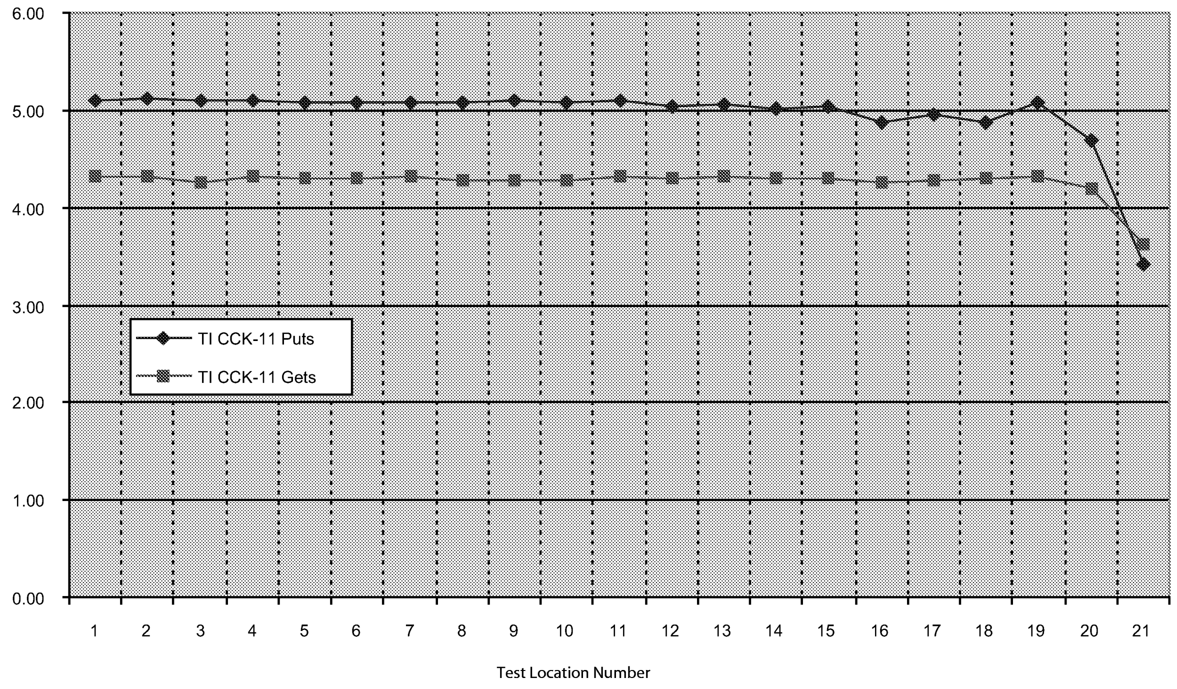


Fig. 21. ACX100 CCK-11 Mb/s average puts and gets over location.

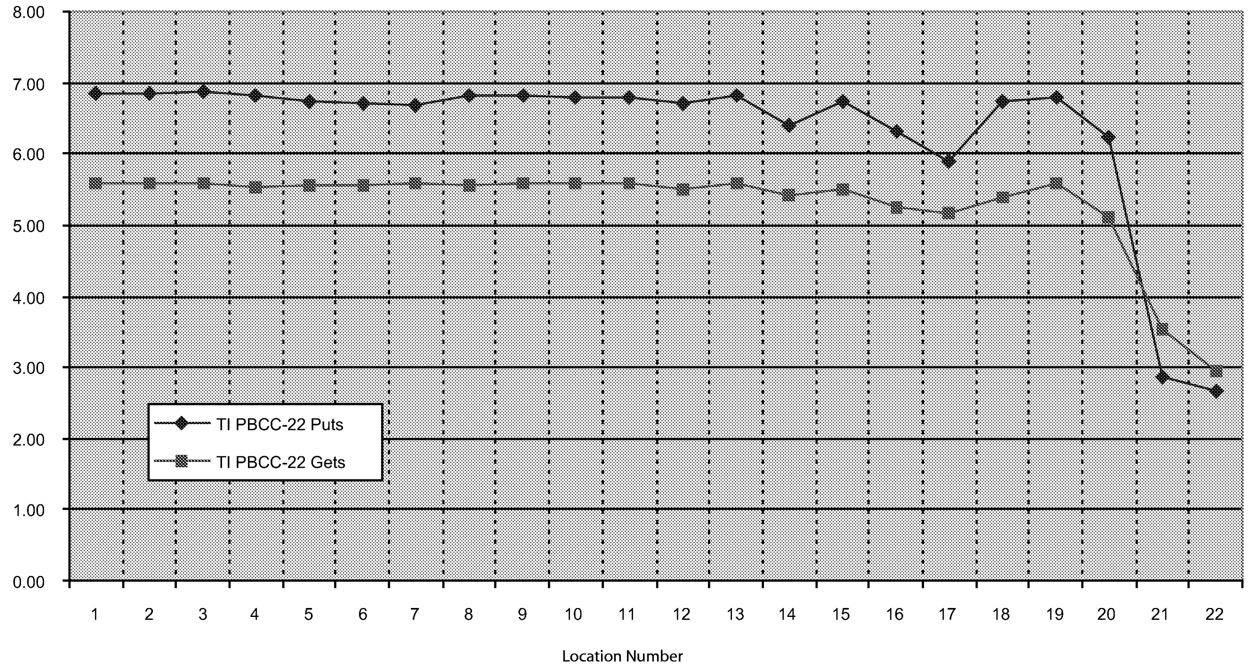


Fig. 22. ACX100 PBCC-22 Mb/s average puts and gets over location.

4) Test Results: The following provides test results between two stations configured in ad hoc mode collected using the test method described in Section V-A3. Fig. 21 shows FTP puts to and gets from the FTP server for TI's CCK-11 PCcard solution versus test location number. These results indicate robust transmitter/receiver performance in a real WLAN environment since the throughput measured does not vary substantially over location. Location 21 is behind a closed door at 129 ft, where the throughput begins to drop because of attenuation of the received signal due to path loss.

Fig. 22 shows results for TI's PBCC-22 PCcard solution versus test location number. Again, these results indicate relatively robust receiver performance in a real environment. As in the case of CCK-11, the throughput drops at location 21 because of attenuation of the received signal due to path loss.

Fig. 23 directly compares average throughput over distance of TI's CCK-11 and PBCC-22 solution to two CCK-11 PC-card solutions currently available on the market. The average throughput is computed by averaging puts and gets together over measurements taken from different test locations at the same

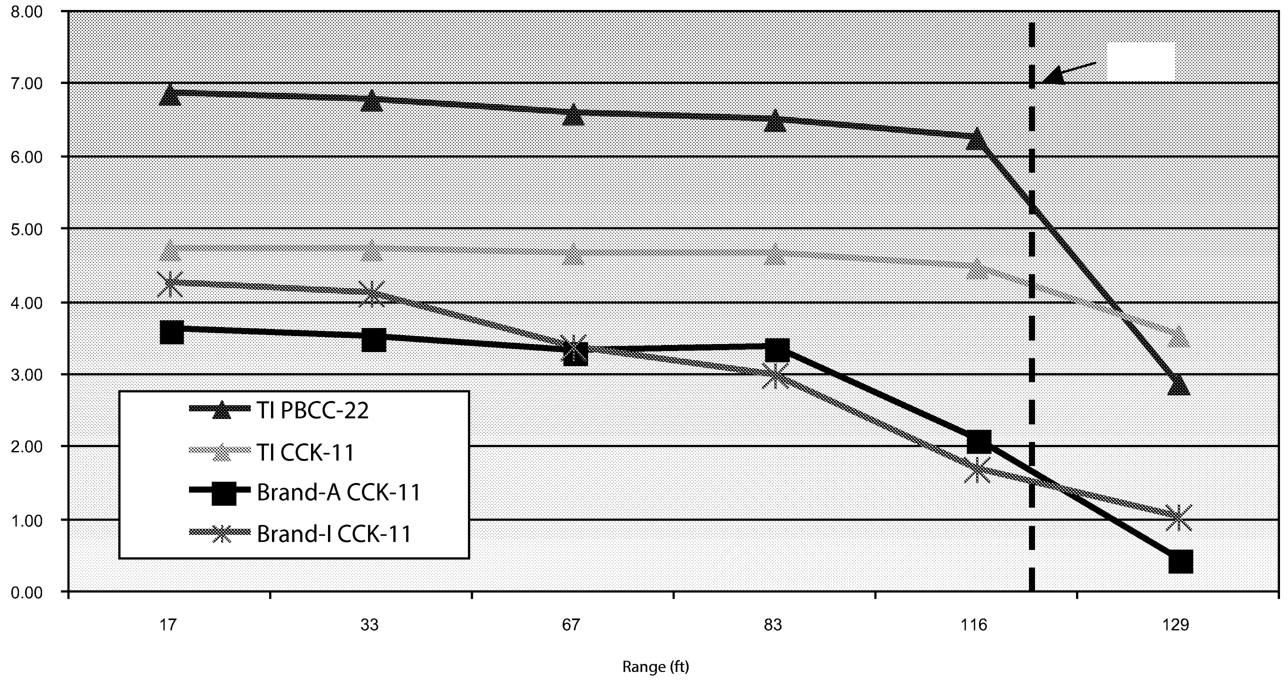


Fig. 23. Average rate versus range: ACX100, brand-I and brand-A/O.

distance thereby producing a single average throughput at each distance. For the TI solution, the average throughput was computed from CCK-11 data shown in Fig. 21 and from PBCC-22 data shown in Fig. 22.

It should be noted that maximum measured throughput depends on several factors including the measuring tool. For example, measured throughput depends heavily on packet size; this is because there is a fixed amount of overhead per packet from preamble times, channel contention times, etc., and with longer packet sizes this overhead occupies a smaller fraction of on-air time. Thus, for meaningful comparisons it is critical to establish a uniform method of measurement. The results presented in this paper were obtained using WAR FTP version 1.65 which is commonly used by many vendors building WLAN products. The measured difference between puts and gets in Figs. 21 and 22 is an artifact of this application. Because of this, the average throughput presented in Fig. 22 may be lower than other results. In addition, Fig. 23 shows the average over puts and gets from Figs. 21 and 22. However, the relative drop in throughput over range should be independent of the measuring tool which directly compares the robustness of the different solutions.

VI. Conclusion

An effective and novel solution to the problem of equalizing and decoding IEEE 802.11b transmissions has been presented. The essential features of the approach are to model the combination of code plus channel-induced multipath distortion as a combined "supercode"; to derive an estimate of the structure of this supercode from the packet preamble; and then to decode this supercode via the computationally efficient Fano sequential decoding algorithm. This approach has been shown to provide excellent results, both in idealized simulation and when implemented in real-world devices. In simulation, the approach is very much better than separate decoding and equalization, and comes very close to achieving maximum theoretically possible performance. In practice, an implementation of this algorithm provides substantial throughput gains over comparable receivers.

References

[1] C. Heegard, S. Coffey, S. Gummadi, P. A. Murphy, R. Provencio, E. J. Rossin, S. Schrum, and M. B. Shoemake, "High performance wireless Ethernet," *IEEE Commun. Mag.*, vol. 39, pp. 64–73, Nov. 2001.

[2] ——, "Evolution of 2.4 GHz wireless LANs," in *Wireless Local Area Networks—The New Wireless Revolution*, B. Bing, Ed. New York: Wiley, 2002, ch. 2.

[3] A. Wozencraft and I. Jacobs, *Principles of Communication Engineering*. New York: Wiley, 1963.

[4] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, Oct. 1948.

[5] G. D. Forney, Jr. and G. Ungerboeck, "Modulation and coding for linear Gaussian channels," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2384–2415, Oct. 1998.

[6] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, 1974.

[7] C. Heegard and S. Wicker, *Turbo Coding*. Norwell, MA: Kluwer, 1999.

[8] A. Vardy, "Trellis structure of codes," in *Handbook of Coding Theory*. Amsterdam, The Netherlands: North-Holland, 1999, vol. 2, pp. 1989–2118.

[9] H. Kobayashi, "Correlative level coding and maximum likelihood decoding," *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 586–594, Sept. 1971.

[10] G. D. Forney, Jr., "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 363–378, May 1972.

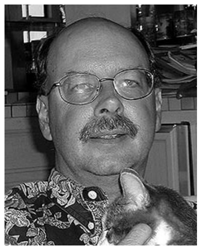
[11] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260–269, Apr. 1967.

[12] S. Lin and D. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice Hall, 1983.

[13] S. B. Wicker, *Error Control Systems for Digital Communications and Storage*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[14] J. L. Massey, "Variable length codes and the Fano metric," *IEEE Trans. Inform. Theory*, vol. IT-18, no. 1, pp. 196–198, 1972.

[15] J. G. Proakis, *Digital Communications*, 3 ed. New York: McGraw-Hill, 1995.
[16] S. Benedetto, E. Biglieri, and V. Castellani, *Digital Transmission Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1987.



Chris Heegard (S'75–M'76–SM'92–F'95) received the electrical engineering degrees from Stanford University, Stanford, CA, and the University of Massachusetts, Amherst.

He is currently a Consultant. Previously, he was the Chief Technology Officer for the Texas Instruments Wireless and Home Networking Business Units, Santa Rosa, CA. He cofounded and served as CEO of Alantro Communications, Santa Rosa, CA, a company specializing in WLAN semiconductor technology, which was acquired by Texas Instruments (TI), in September 2000. He served as a Faculty Member at the School of Engineering, Cornell University, Ithaca, NY, for 19 years. He is the author of numerous publications, the inventor on several patents and is coauthor of *Turbo Coding* (Kluwer, 1999). He is also founder of Native Intelligence, Ithaca, NY, a digital communications software company.

Dr. Heegard was a Texas Instruments Fellow.

Seán Coffey (S'86–M'89) received the B.E. degree from University College, Dublin, and the M.S. and Ph.D. degrees in electrical engineering from California Institute of Technology, Pasedena.

From 1989 to 2000, he was a Faculty Member in the Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor. In May 2000, he joined Alantro Communications, Santa Rosa, CA, now a part of Texas Instruments (TI). He manages the Advanced PHY Group in TI's Wireless Networking Business Unit.

Srikanth Gummadi (S'97–M'00) was born in India on May 30, 1976. He received the B.S.E.E. degree from Indian Institute of Technology, Madras, in 1997 and the M.S.E.E. degree from The University of Texas, Austin, in 1998.

From 1999 to 2000, he was a Research Engineer, Motorola Labs, Fort Worth, TX, and is presently a Systems Engineer at Texas Instruments (TI) (formerly, Alantro Communications), Santa Rosa, CA. His research interests include wireless communication, smart antennas, and array signal processing.

Eric J. Rossin received the B.S.E.E., M.Eng., and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY, in 1983, 1984, and 1995, respectively.

He is currently a consultant. He was previously with the Wireless LAN Business Unit, Texas Instruments (TI), Santa Rosa, CA. He was a cofounder and President of Alantro Communications from 1997 to 2000. He has also held engineering and management positions at Next Level Communications and Applied Signal Technology, Sunnyvale, CA, in addition to consulting for several corporations.

Matthew B. Shoemake (S'91–M'99) received the M.S. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY, and the B.S. degree in electrical engineering and computer science from Texas A&M University, College Station.

He is with the Office of the CTO in the Wireless Networking Business Unit, Texas Instruments (TI), Dallas, TX. He is the Chairperson for IEEE 802.11 Task Group G, the committee tasked with extending the IEEE 802.11b standard to higher data rates. Prior to TI, he was Member of the Technical Staff for Alantro Communications, a company specializing in WLAN semiconductor technology, which was acquired by TI in late 2000. He co-invented packet binary convolutional coding (PBCC), the high-performance modulation mode of the IEEE 802.11b standard, and is a key designer of TI's 802.11b compliant chips.

Michael Wilhoyte (A'96) received the B.S. and M.S. degrees in mathematics from the University of Toledo, Toledo, OH, in 1987 and 1989, respectively.

He is the Manager of the Radio Systems Test Group for Texas Instruments (TI), Wireless Networking Business Unit in Santa Rosa, CA. Mike has been with TI (previously Alantro) since April 2000. From 1993 to 2000, he was with Applied Signal Technology, Sunnyvale, CA, where he managed a DSP algorithm development group working on advanced demodulation and decoding techniques for many mobile wireless systems. From 1989 to 1993, he was with Lockheed Missles and Space Co., Sunnyvale, CA, where he was a Member of the DSP algorithm development team working on equalization and detection architectures for voice-grade modems.