

Gallager Codes for CDMA Applications—Part II: Implementations, Complexity, and System Capacity

Vladislav Sorokine, *Member, IEEE*, Frank R. Kschischang, *Member, IEEE*, and Subbarayan Pasupathy, *Fellow, IEEE*

Abstract—We focus our attention on Gallager codes with parameters compatible with the IS-95 cellular radio standard. We discuss low complexity software and hardware implementations of an iterative decoder for $\{N, -, 3\}$ Gallager codes. We estimate that by using Gallager codes, a factor of five improvement in the code-division multiple-access system capacity relative to an uncoded system can be achieved, equivalent to a factor of two improvement relative to state-of-the-art orthogonal convolutional codes. Simulation results demonstrate the good performance of short-frame Gallager codes in the additive white Gaussian noise and certain fading channels.

Index Terms—Complexity theory, decoder architecture, information rates, low-density parity-check codes.

I. INTRODUCTION

CODE-DIVISION multiple access (CDMA) is one of the three most popular approaches (along with time-division multiple access and frequency-division multiple access) for allocating the available signal space to multiple users. In the IS-95 CDMA standard, a particular type of CDMA called *spread-spectrum multiple access* is used. Briefly, the forward link of the system operates as follows. A binary data sequence is produced by a data source at the rate of 9.6 kbits/s (in digital speech transmission), spread by a factor of 128, modulated and transmitted in a communications channel. Spreading is used partly to combat the multiple-access interference and partly to combat the channel noise with forward error-correction.

The CDMA system capacity depends to a large degree on the efficiency of the error-control code. In the forward link, IS-95 uses a rate 1/2 convolutional code for error-protection.

A much larger portion of the total bandwidth expansion can be allocated to error-control coding. Better codes will increase the system capacity [1], [2]. However, other factors, such as the decoding complexity, may negate the positive effect.

Paper approved by W. E. Ryan, the Editor for Modulation, Coding, and Equalization of the IEEE Communications Society. Manuscript received February 16, 1999; revised May 20, 1999 and August 31, 1999. This work was supported in part by the Natural Sciences and Engineering Research Council, Canada. This paper was presented in part at the 1997 Canadian Workshop on Information Theory, Toronto, ON, Canada, June 1997, at the 1998 Information Theory Workshop, Killarney, Ireland, June 1998, and at the Ninth International Symposium on Personal, Indoor and Mobile Communications (PIMRC'98), Boston, MA, September 1998.

V. Sorokine was with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada. He is now with Qualcomm Inc., San Diego, CA 92121 USA (e-mail: vsorokin@qualcomm.com).

F. R. Kschischang and S. Pasupathy are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada.

Publisher Item Identifier S 0090-6778(00)09883-4.

We will argue that the $\{1536, -, 3\}$, rate 1/8 Gallager codes of [part I] are good candidates to provide an enhanced error-protection scheme in CDMA systems. Good performance in additive white Gaussian noise (AWGN) and fading channels makes them particularly suitable for mobile communications. Moreover, one can devise low-complexity decoders for these codes, an important consideration for hand-held radio units where power consumption is at a premium.

This paper is organized as follows. In Section II, we give a brief review of the sum-product decoding algorithm along with most of the relevant equations that are used extensively in later sections. In Sections III and IV, we discuss low complexity implementations of the sum-product decoder both in software and in hardware. In Section V, we discuss the computational complexity of encoding and iterative decoding for the $\{1536, -, 3\}$, rate 1/8 Gallager codes. In Section VI, we present simulation results for Gallager codes for the AWGN and fading channels. We use simulation results for the AWGN channel in a simplified analysis of the single cell capacity of a CDMA system in Section VII, and show that Gallager codes provide a twofold increase in the system capacity relative to state-of-the-art orthogonal convolutional codes [4]. Finally, we present the discussion of the results in Section VIII.

II. SUM-PRODUCT ALGORITHM

To decode Gallager codes, we use the sum-product algorithm [5], [6], a version of which was described in the original work of Gallager [7], and which is sometimes referred to as “belief propagation” [8], [9]. The sum-product algorithm operates by passing messages along the edges of a bipartite factor graph [10] that describes the conditional joint probability mass function of the codeword symbols given the received channel output. The sum-product algorithm is known by various names in special cases and in different application areas. For example, in the special case when the factor graph represents a trellis or hidden Markov model, the sum-product algorithm is known (in coding) as the BCJR algorithm [11] or (in signal processing) as the forward/backward algorithm [12]; in statistics, it is an integral part of the Baum-Welch algorithm. For a tutorial treatment of factor graphs, the sum-product algorithm and applications, see [10]. MacKay [13] gives an excellent description of the sum-product algorithm that is used to decode Gallager codes, and we adopt his notation here.

As in Section II, Part I, let H be a low-density parity-check matrix for a Gallager code \mathcal{C} , let H^T denote the transpose of H , and let $\mathbf{x} = (x_1, \dots, x_n)$ be a transmitted codeword. The channel adds a noise sequence $\mathbf{n} = (n_1, \dots, n_n)$ to \mathbf{x} to produce the channel output $\mathbf{y} = (y_1, \dots, y_n)$. Assuming ± 1 an-

tipodal signaling, the decoder starts by quantizing the channel output, forming the hard-decision vector $\bar{\mathbf{y}}$ whose i th component is

$$\bar{y}_i = \begin{cases} 0, & \text{if } y_i < 0 \\ 1, & \text{otherwise.} \end{cases}$$

Next, using $GF(2)$ arithmetic, the syndrome $\mathbf{z} = \bar{\mathbf{y}} \cdot H^T$ is formed. If the syndrome evaluates to the zero vector, the decoder produces $\bar{\mathbf{y}}$ as its output; otherwise, it tries to find an error-pattern that has the same syndrome as $\bar{\mathbf{y}}$. For this purpose, the decoder performs a number of algorithmically identical steps called *iterations*. This process is described next.

With the parity-check matrix H we associate a sequence of four real “message matrices”: $Q^0(i)$, $Q^1(i)$, $R^0(i)$ and $R^1(i)$, where $i \geq 0$ is the iteration index. Matrices $Q^{0/1}(i+1)$ and $R^{0/1}(i+1)$ are computed from $Q^{0/1}(i)$ and $R^{0/1}(i)$. The elements of these matrices represent the messages passed back and forth between the symbol vertices and the check vertices in the corresponding factor graph. Following [5] and [6], let us call each received codeword symbol a *site*, and a row of the parity-check matrix H a *check*. Matrices $Q^0(i)$ and $Q^1(i)$ store *site-to-check* messages and matrices $R^0(i)$ and $R^1(i)$ store *check-to-site* messages.

Let $q_{kl}^0(i)$, $q_{kl}^1(i)$, $r_{kl}^0(i)$, $r_{kl}^1(i)$ and h_{kl} denote the elements of matrices $Q^0(i)$, $Q^1(i)$, $R^0(i)$, $R^1(i)$ and H , respectively. Elements $q_{kl}^0(i)$, $q_{kl}^1(i)$, $r_{kl}^0(i)$ and $r_{kl}^1(i)$ may assume nonzero values only if $h_{kl} = 1$. Since H is sparse, so are the message matrices. (Note that the subscript k denotes here the row index of an element and is different from the parameter k in the $\{N, j, k\}$ notation.)

The algorithm itself consists of the following steps [13].

- 1) *Initialization*. For each site (code symbol) l of the received sequence \mathbf{y} we initialize p_l^1 to the likelihood that the noise vector has a one in position l (i.e., it “flips” the transmitted bit), and p_l^0 to the likelihood that the noise vector has a zero in position l . Elements $q_{kl}^0(0)$, $q_{kl}^1(0)$, $r_{kl}^0(0)$, and $r_{kl}^1(0)$ are initialized to 1 if $h_{kl} = 1$, otherwise, to 0.

- 2) *Iteration*.

- *Site-to-check propagation (Vertical Step)*. The algorithm runs over the matrices $R^0(i)$ and $R^1(i)$ column-wise and computes elements of $Q^0(i+1)$ and $Q^1(i+1)$ (hence the name “vertical step”). Let $s(l)$ be the set of row indices of nonzero elements in l th column of H . Then

$$q_{kl}^0(i+1) = \gamma_{kl} p_l^0 \prod_{k' \in s(l) \setminus k} r_{k'l}^0(i) \quad (1)$$

and

$$q_{kl}^1(i+1) = \gamma_{kl} p_l^1 \prod_{k' \in s(l) \setminus k} r_{k'l}^1(i) \quad (2)$$

where $s(l) \setminus k$ denotes the set $s(l)$ with the element $\{k\}$ excluded, and γ_{kl} is a normalizing factor such that $q_{kl}^0 + q_{kl}^1 = 1$. In factor graph terminology, a site-to-check message passed on a given factor graph edge, is computed at a site as the normalized product of messages arriving on all *other*

edges at that site, in accordance with the general sum-product rule [10].

- *Check-to-site propagation (Horizontal Step)*. The decoder runs over $Q^0(i)$ and $Q^1(i)$ row-wise and computes the entries of $R^0(i+1)$ and $R^1(i+1)$ (hence the name “horizontal step”). Let $E(k)$ be the set of column indices of nonzero elements in the k th row of H . Then

$$\begin{aligned} r_{kl}^0(i+1) &= \\ & \sum_{\{x_{l'}: l' \in E(k) \setminus l\}} P(z_k | x_l = 0, \{x_{l'}: l' \in E(k) \setminus l\}) \\ & \times \prod_{l' \in E(k) \setminus l} q_{kl'}^{x_{l'}}(i+1) \end{aligned} \quad (3)$$

and

$$\begin{aligned} r_{kl}^1(i+1) &= \\ & \sum_{\{x_{l'}: l' \in E(k) \setminus l\}} P(z_k | x_l = 1, \{x_{l'}: l' \in E(k) \setminus l\}) \\ & \times \prod_{l' \in E(k) \setminus l} q_{kl'}^{x_{l'}}(i+1) \end{aligned} \quad (4)$$

where $E(k) \setminus l$ denotes the set $E(k)$ with the element $\{l\}$ excluded and $P(\cdot)$ is the probability of observing the k th component of the syndrome \mathbf{z} given the site values $x_l = 0/1$ and $\{x_{l'}: l' \in E(k) \setminus l\}$ (this probability is either 0 or 1). In factor graph terminology, a check-to-site message passed on a given factor graph edge is computed at the check as the product of messages arriving on all *other* edges at that check, multiplied with a local indicator function P , and then marginalized for the variable associated with the given edge, all in accordance with the general sum-product rule [10].

- 3) *Termination*. After the vertical and horizontal steps, the posterior probabilities q_l^0 and q_l^1 are computed as follows:

$$q_l^0 = p_l^0 \prod_{k \in s(l)} r_{kl}^0(i) \quad (5)$$

and

$$q_l^1 = p_l^1 \prod_{k \in s(l)} r_{kl}^1(i). \quad (6)$$

If $q_l^0 > q_l^1$ then $n_l = 0$, otherwise $n_l = 1$. Let $\hat{\mathbf{n}}$ be the tentative noise vector produced by the algorithm. If $\hat{\mathbf{n}}$ has the same syndrome as $\bar{\mathbf{y}}$, the decoder stops, reveals $\mathbf{n} = \hat{\mathbf{n}}$ and outputs $\mathbf{x} = \bar{\mathbf{y}} + \mathbf{n}$; otherwise it proceeds with the next iteration step. An upper limit is placed on the number of iterations and a decoder failure is declared when this upper limit is reached without convergence to the given syndrome.

An example of iterative decoding is shown in Fig. 1 where a block of 256 bits (an 8×32 binary image) was encoded with a rate 1/2 code (so the block length is 512) and transmitted over a noisy Gaussian channel with noise level of $E_b/N_0 = 1.5$ dB. The iterations are arranged column-wise in Fig. 1, with only information bits shown. The first image in the top-left corner corresponds to the received codeword and the final image in the

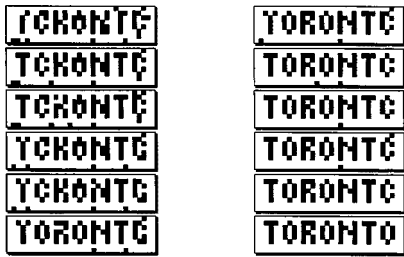


Fig. 1. Instance of iterative decoding: iterations increase from top to bottom and from left to right.

bottom-right corner corresponds to the correctly decoded code-word. It took 11 iterations to achieve correct decoding. (Reference [13, Figs. 12 and 13] gives another interesting decoding example.)

As MacKay [13] points out, the horizontal step (computation of check-to-site messages) can be simplified by operating with the differences between the elements of the Q matrices. Writing

$$\begin{aligned} (r_{kl}^0(i+1) - r_{kl}^1(i+1)) \\ = (-1)^{z_k} \prod_{v \in E(k) \setminus l} (q_{klv}^0(i+1) - q_{klv}^1(i+1)) \end{aligned}$$

yields a simple product-form that is easily computed, from which the desired values are recovered as

$$\begin{aligned} r_{kl}^0(i+1) &= (1 + (r_{kl}^0(i+1) - r_{kl}^1(i+1)))/2 \\ r_{kl}^1(i+1) &= (1 - (r_{kl}^0(i+1) - r_{kl}^1(i+1)))/2. \end{aligned}$$

This simplification arises from the well-known fact [14] (see [15] for a recent generalization) that APP (*a posteriori* probability) computation can be performed by processing the dual of a given linear code taking the Fourier transforms of the symbol probabilities used in decoding the primal code. In the binary case, the Fourier transform matrix is 2×2 matrix that maps (q_0, q_1) to $(q_0 + q_1, q_0 - q_1)$, the first component of which is unity when (q_0, q_1) represents a probability mass function. Each check represents a simple parity-check code, the dual of which is a repetition code with a trivial trellis structure.

III. SOFTWARE IMPLEMENTATION OF ITERATIVE DECODER

A software iterative decoder for Gallager codes can be implemented effectively if:

- it exploits the sparsity of the code's parity-check matrix;
- it pre-computes operations that are identical in every iteration and accomplishes them by a low-cost table look-up in each decoding step.

We represent the parity-check matrix H as two related lists H_c and H_r ; each list contains the location of the nonzero elements of H . Matrices H_c and H_r describe H column-wise and row-wise, respectively.

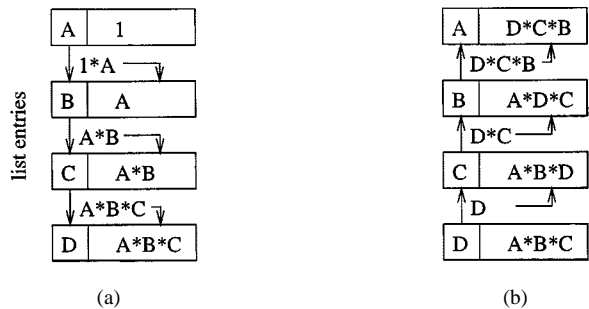


Fig. 2. (a) Forward pass. (b) Backward pass. Forward-backward passes for computing (1)–(4). Symbols A, B, C, D correspond to quantities $p_{kl}^{0/1}$ and $q_{kl}^{0/1}$. Note that after the backward pass is completed [diagram (b)], the right part of each cell contains the product of quantities from the left parts of all *other* cells (excluding the cell itself).

Example: The matrix [part I: (2)] can be represented by the following two matrices.

$$H_c = \begin{Bmatrix} (1, 1) & 1 \\ (5, 1) & 1 \\ (7, 1) & 1 \\ (1, 2) & 1 \\ \dots & \dots \\ (5, 9) & 1 \end{Bmatrix}$$

and

$$H_r = \begin{Bmatrix} (1, 1) & 1 \\ (1, 2) & 1 \\ (1, 3) & 1 \\ (2, 4) & 1 \\ \dots & \dots \\ (7, 5) & 1 \end{Bmatrix}.$$

Similarly, matrices $Q^0(i)$ and $Q^1(i)$ are represented by row-wise lists (the algorithm runs over these matrices in the horizontal step), and matrices $R^0(i)$ and $R^1(i)$ are represented by column-wise lists (these are used in the vertical step).

Since the algorithm alternates between the vertical and horizontal steps, we create lists $SORT$ and $SORT_{inverse}$ that establish a correspondence between the elements of H_c and H_r , and H_r and H_c , respectively. These lists are computed only once for a particular Gallager code.

In the vertical step, the sum-product algorithm runs through the lists that are the column-wise representations of $R^0(i)$ and $R^1(i)$. The results of the computations (1) and (2) are placed in appropriate positions in the lists representing $Q^0(i+1)$ and $Q^1(i+1)$ by using list $SORT$. In the horizontal step, the algorithm runs over the lists representing $Q^0(i)$ and $Q^1(i)$. Expressions (3) and (4) yield values that are placed in appropriate positions in the lists representing $R^0(i+1)$ and $R^1(i+1)$ by using $SORT_{inverse}$.

Note that the products in (1)–(4) require computations over sets $s(l) \setminus k$ and $E(k) \setminus l$. An efficient way to do it is to run forward-backward passes over the lists R and Q as illustrated in Fig. 2 [16].

Computations in the Logarithmic Domain: In the vertical and horizontal steps, we compute products of certain local costs (1)–(4). Multiplications are computationally expensive compared with additions. It is possible to replace multiplications with additions if one operates in the logarithmic domain. An

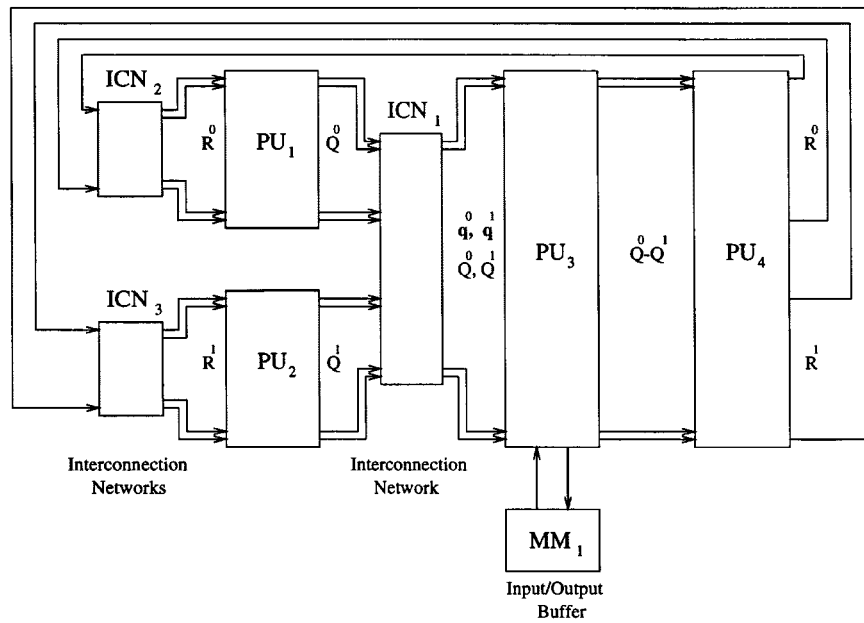


Fig. 3. Decoder's architecture: PU—processor units, ICN—interconnection networks, MM—memory module (data buffer).

addition can be evaluated then by a table look-up procedure. Such look-up table will have a very moderate size even if high accuracy is required.

Example: If $x = \ln X$ and $y = \ln Y$, then [17]

$$\begin{aligned} \ln(X + Y) &= \ln(e^x + e^y) \\ &= \ln \left[e^{\max(x, y)} (1 + e^{-|x-y|}) \right] \\ &= \max(x, y) + \ln(1 + e^{-|x-y|}). \end{aligned} \quad (7)$$

The function $f(z) = \ln(1 + e^{-z})$ is tabulated for positive z . Since $f(z)$ decays quickly as z increases, a small look-up table can provide high accuracy in evaluating the function [18].

IV. HARDWARE ARCHITECTURE OF ITERATIVE DECODER

In this section we shall discuss general hardware algorithms that could be used to implement the iterative decoder. We emphasize the algorithm's inherent parallelism that could lead to low-cost hardware implementations.

The decoder consists of four processor units and three interconnection networks that form a pipeline design. A memory module is used as a data buffer. The schematic configuration that illustrates parallelism and pipelineability [19] of the decoder is shown in Fig. 3.

In Fig. 3 the processor unit PU₃ computes the difference between matrices Q^0 and Q^1 rather than the matrices themselves, making use of the simplification explained in Section II.

Recall that the sum-product algorithm alternates between computing the elements of matrices Q and R . Hence, if the decoder is dedicated to decoding of a single codeword at a time, only a fraction of circuitry will perform decoding operations at any given moment of the decoding cycle (iteration step). To increase the circuit utilization, we suggest simultaneous decoding of several received codewords. This calls for use of a data buffer to store these codewords and enter them into the decoder when a free slot is available. The tradeoff in

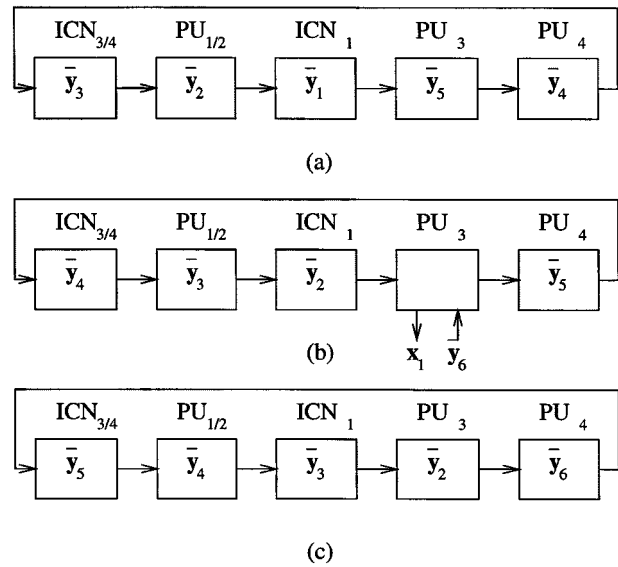


Fig. 4. Snapshots of decoder's states. (b) Decoded codeword x_1 exits the decoder.

the proposed scheme is an additional decoding latency. (The decoder in Fig. 3 can contain five codewords in PUs and ICNs simultaneously so the total decoder's latency is $20 \text{ ms} \times 5 \text{ codewords} = 100 \text{ ms}$.)

Iterative decoding is generally characterized by a variable decoding effort. Usually the number of iterations performed by a decoder is hard-limited in order to avoid these variations. In our scheme, the decoder may perform a larger number of iterations for particularly noisy received codewords since the data buffer will damp such computational peaks. This is illustrated in Fig. 4.

The signal flow graph for the processor units PU_{1/2} is shown in Fig. 5. Processor units PU_{1/2} execute the vertical step of the sum-product algorithm. Recall that the number of nonzero elements in a column of H is either 2 or 3. Thus, the processor unit consists of the top part a) (weight 2 columns) and the bottom part

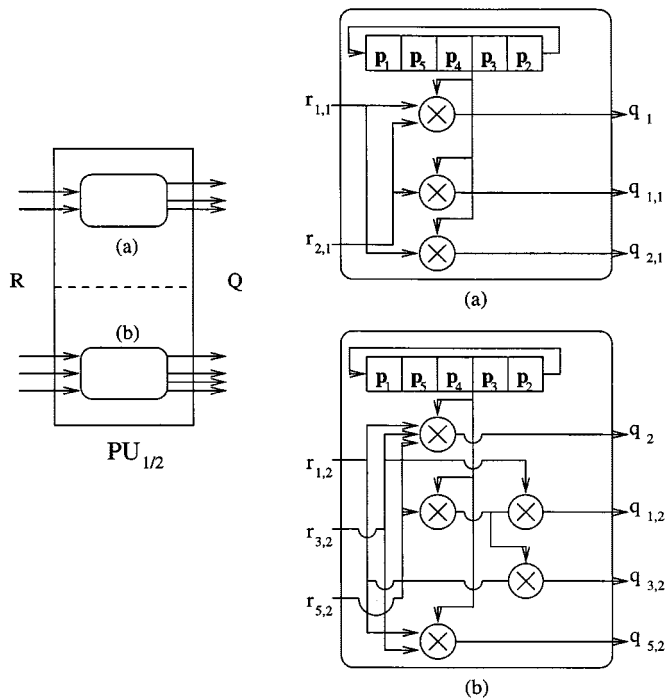


Fig. 5. Implementing the vertical step of the algorithm—signal flow graph for processor units $PU_{1/2}$. Variables r_{kl} , q_{kl} are the elements of R and Q , q_l are tentative posterior probabilities, p_i are the likelihood functions for i th received codeword.

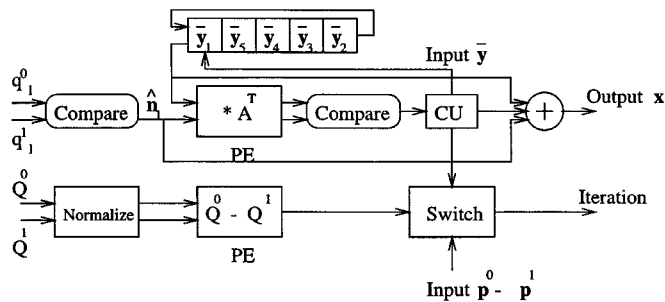


Fig. 6. Signal flow graph for the processor unit PU_3 : CU—control unit; PE—processor elements; processor element $*A^T$ computes syndromes of \hat{n} and \bar{y} , x is the decoded vector, p^0 and p^1 are the likelihood functions of a new vector \bar{y} .

b) (weight 3 columns). Appropriate routing for this arrangement is done by ICN_2 and ICN_3 .

Fig. 6 shows the processor unit PU_3 . It performs the syndrome computation of the tentative noise vector \hat{n} and the received vector \bar{y} and compares these syndromes. If they are identical, the control unit CU gives a command to output the decoded vector x and load new p^0 and p^1 (i.e., start decoding of a new vector \bar{y}). If the syndromes are not identical, PU_3 computes the difference $Q^0 - Q^1$ which is used in the next iteration for the same \bar{y} .

The signal flow graph for the processor array PU_4 is shown in Fig. 7. Note that in our case k is always 3, so PU_4 has a “uniform” structure (in contrast with processor units PU_1 and PU_2).

The function of interconnection network ICN_1 is twofold. It switches signals from the column-wise structure of PU_1 and

PU_2 [that corresponds to Fig. 5(a) and (b)] to the row-wise structure of PU_3 , and routes signals q_{kl}^0 and q_{kl}^1 to adjacent inputs of PU_3 . Interconnection networks ICN_2 and ICN_3 perform routing from row-wise to column-wise arrangement of signals required for PU_1 and PU_2 . These networks are programmed only once for each Gallager code.

The networks ICN_1 , ICN_2 and ICN_3 have global but static connections. One possible realization of these networks is a programmable cross-bar switch such that every input to the network can be connected to any output. This configuration is shown in Fig. 8.

A better architecture for ICN_1 , ICN_2 and ICN_3 is achieved by using *Banyan networks* [20], [21]. An example of a three-stage Banyan network is given in Fig. 9.

Let us denote the input (and output) size of a Banyan network as S . The number of stages required to route any input to any output in a Banyan network is $\log_2(S)$. Hence, for (1536, 1344, -), {1536, -, 3} Gallager code $\lceil \log_2(4032) \rceil = 12$ stages are required. The complexity of an $S \times S$ Banyan network is $S \log_2 S$ cross points as compared with S^2 cross points of a cross-bar switch. Hence, the complexity of a Banyan network for a {1536, -, 3} Gallager code is 48 292 cross points.

The most immediate problem that arises in a Banyan network but not in a cross-bar switch, is message blocking. If two messages ($q_{k_1 l_1}^{0/1}$ and $q_{k_2 l_2}^{0/1}$) try to arrive at the same output of a 2×2 cross-bar switch (the basic block in Fig. 9) then one message is rejected. One way to circumvent this difficulty is to introduce a small number of redundant paths into the Banyan network. An alternative is to introduce a certain depth buffer at the network's input. Since the arrival of messages to the network's input is not random but fixed for any given Gallager code, pre-computed routing of the messages through the network will allow to avoid any possible message blocking.

V. ENCODING AND DECODING COMPUTATIONAL COMPLEXITY

In this section we discuss one of the most important issues that determines the feasibility of a practical coding scheme. Whereas there are many good block and convolutional codes whose performance is known to be good, a restricting factor in the practical implementation of these codes is the high complexity of software and hardware that is required to extract this good performance.

As we recall, Gallager's motivation for studying low-density parity-check codes was precisely the existence of low-complexity decoders for such codes. Beside being amicable to low-complexity implementations, it is fortunate that Gallager codes also turned out to have many good properties that result in good performance in a variety of communications channels.

The possibility of implementing a coding scheme with low complexity is particularly important in the context of this work. In mobile cellular telephony it is next to impossible to implement expensive software or hardware algorithms on the mobile side because of a variety of severely restrictive factors, e.g., the need to maximize time intervals between battery re-chargings, to maintain small data delays, etc.

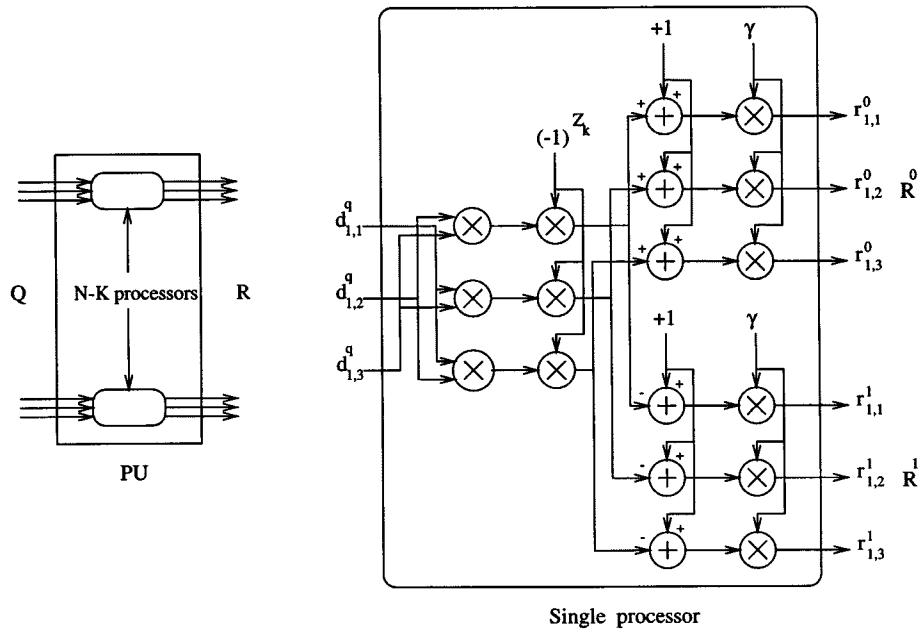


Fig. 7. Implementing the horizontal step of the algorithm—signal flow graph for the processor unit PU_4 . Variables d_{kl}^q are the differences of q_{kl}^0 and q_{kl}^1 , z_k is the k th component of the syndrome, γ is a normalization coefficient, and r_{kl} are elements of R .

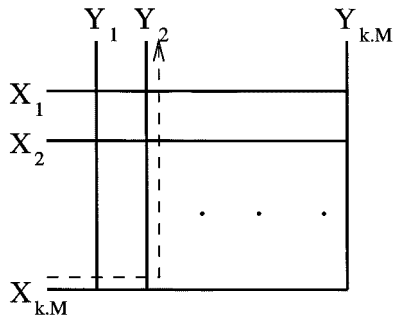


Fig. 8. Cross-bar switch for interconnection networks.

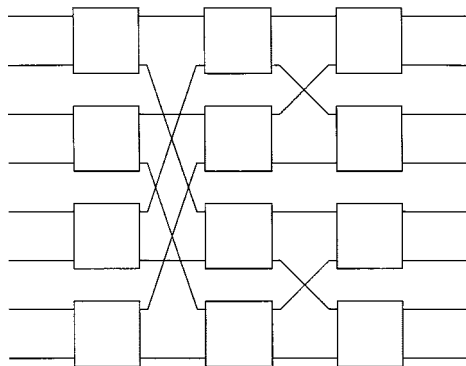


Fig. 9. An example of 8×8 Banyan network.

There are two major contributing factors to the complexity of the coding scheme based on Gallager codes, namely, the costs of encoding and decoding.

A. Encoding

Let us first discuss the encoding complexity for $(1536, 192, -)$, $\{1536, -3\}$, Gallager codes. The code's generator matrix has dimensions 192×1536 . Unlike the corresponding parity-

check matrix, the generator matrix is not sparse. Hence, the encoder has to store the entire generator matrix. The storage space required for the matrix is $192 \times 1536/8 = 36\,864$ bytes. The size of the storage space can be slightly reduced if the generator matrix is in a systematic form, and the encoder needs to store only the nonsystematic part. For $(1536, 192, -)$ Gallager codes this will save $192 \times 192/8 = 4608$ bytes.

Each bit of an output codeword is obtained by multiplying in a bit-wise manner a length 192 binary vector of input data with a column of the generator matrix and computing the overall parity. The number of clock cycles required to accomplish such multiplication depends on the length of processor's registers. For example, if the processor's word length is 32 bits then the operation will require $192/32 = 6$ cycles, and the total number of clock cycles required for encoding is $6 \times 1536 = 9216$ ($6 \times 1344 = 8064$ clock cycles if the matrix is in the systematic form).

A multiplier in a digital signal processing (DSP) chip does not normally contain a circuit for the parity computation. However, if this operation is done often (as in our case) such a circuit can be easily implemented with exclusive-OR gates. Hence, the cost of the parity computation is negligible in any application specific hardware.

For comparison, encoding of a rate $1/2$, constraint length 7 convolutional code will require (for information block length of 192) $192 + 6 = 198$ register shifts and $198 \times 2 = 396$ bitwise multiplications, with two multiplications for each register shift (since the code rate is $1/2$). The total number of clock cycles required for encoding is $198 + 396 = 594$. Hence, encoding of a Gallager code is slower than encoding of a convolutional code with parameters discussed above. Similar analysis applies to turbo codes since in a turbo code component codes are recursive convolutional codes of rate 1.

However, boolean operations bear a lower cost compared with arithmetic operations required for decoding of the received codeword. (By an arithmetic operation we mean either an addition or

multiplication of two floating or fixed point real numbers.) The overall encoding/decoding complexity is dominated by the decoding complexity since decoding requires thousands of arithmetic operations per frame for Gallager codes.

B. Decoding

Let us calculate the decoding complexity for the $\{1536, -3\}$ Gallager codes. The computational complexity of the sum-product algorithm is roughly proportional to the number of nonzero elements in the parity-check matrix H . In what follows, we compute the number of arithmetic operations (OPs) required for decoding (the number format could be either floating or fixed point depending on the implementation), i.e., we consider operations of adding or multiplying two numbers to have equal cost.

Consider a $(1536, 192, -)$, $\{1536, -, 3\}$ Gallager code. We start with the horizontal step of the algorithm.

To compute the total cost of the horizontal step we need to take into account all parity-checks. $M \times k = 1344 \times 3 = 4032$ additions are required to compute the difference of Q^0 and Q^1 , which is necessary in the horizontal step [13]. Computation of the elements of R^0 and R^1 requires two multiplications and one addition per element which amounts to the total complexity is 12 096 OPs. Thus, the cost of the horizontal step is 16 128 OPs.

The vertical step involves computations of particular elements q_{kl}^0 and q_{kl}^1 as well as computations of tentative posterior probabilities q_i^0 and q_i^1 . Construction 2 produces $k \times (M \bmod N/k)$ columns of weight 3 and $N - k \times (M \bmod N/k)$ columns of weight 2 (for code rates $R < 1$). Substituting the numerical values, we obtain 960 columns of weight 3 and 576 columns of weight 2.

For each weight 3 column a total of 5 multiplications is required to compute the quantities $q_{\{k_1, k_2, k_3\}l}^0$ or $q_{\{k_1, k_2, k_3\}l}^1$. Suppose we need to compute the quantities $q_{k_1 l}^0$, $q_{k_2 l}^0$ and $q_{k_3 l}^0$. We notice that $q_{k_1 l}^0 = p_l^0 \times r_{k_3 l}^0 \times r_{k_2 l}^0$ and $q_{k_2 l}^0 = p_l^0 \times r_{k_3 l}^0 \times r_{k_1 l}^0$ (a total of 3 multiplications); and $q_{k_3 l}^0 = p_l^0 \times r_{k_1 l}^0 \times r_{k_2 l}^0$ (two additional multiplications). The normalization factor γ_{kl} is convenient but not essential, and presently we do not take it into account. Therefore, $2 \times 5 \times 960 = 9600$ OPs are required. For each column with 2 nonzero elements only 1 multiplication is required for q_{kl}^x so the complexity is $2 \times 1 \times 2 \times 576 = 2304$ OPs.

The computation of the tentative posterior probabilities (5) and (6) will require only $2 \times N = 2 \times 1536 = 3072$ OPs as these can be obtained by multiplying (1) and (2) by an appropriate column element r_{kl}^0 and r_{kl}^1 .

Thus, the total complexity of the sum-product algorithm per iteration is $16\,128 + 9600 + 2304 + 3072 = 31\,104$ OPs. The complexity per information bit per iteration is then $31\,104/192 = 162$ OPs.

For comparison, iterative decoding of 16-state turbo codes has a cost of 192 OPs per information bit per iteration [22]. [For a turbo code the number of multiplications per information bit is $12S$, where S is the state complexity of constituent codes. Note that this expression is independent of code rate. We will obtain the complexity of turbo-decoding at 192 OPs per infor-

mation bit per iteration if we assume that the state complexity of constituent codes is $S = 16$ (as in [23]).]

The overall decoding complexity of Gallager codes depends on the number of iterations required for decoding. The mean of this number depends in its turn on the channel SNR and also on the definition of the mean, i.e., if one computes the average over all decodings (in which case the average will also depend to a large degree on the maximum number of iterations the decoder is allowed to perform) or over only successful decodings. For higher channel SNR both definitions will yield approximately the same results as the number of unsuccessful (nonconvergent) decodings becomes small (but the computed averages may diverge significantly for these two definitions as SNR decreases). We observed in simulations (where we computed the average over all decodings) that the decoder requires on the order of ten iterations for signal-to-noise ratios in the range of 2.5–3.5 dB.

VI. SIMULATION RESULTS

A. Performance of Gallager Codes in AWGN Channels

In this set of simulations we modeled the performance of the decoder based on the sum-product algorithm. The site likelihoods for AWGN channel and ± 1 antipodal signaling are [13]

$$p(n_l = 1|y_l) = \frac{1}{1 + \exp(2y_l/\sigma^2)} \quad (8)$$

and

$$p(n_l = 0|y_l) = \frac{1}{1 + \exp(-2y_l/\sigma^2)}. \quad (9)$$

If the received signal component y_l is negative then the expressions (8) and (9) for the conditional probabilities $p(n_l = 1|y_l)$ and $p(n_l = 0|y_l)$ should be exchanged. In our simulations we ran the decoder on 5×10^6 frames for $E_b/N_0 > 2$ dB and 10^5 frames for $E_b/N_0 < 2$ dB. For both the AWGN and fading channels, we observed that the decoder produces no undetected errors, i.e., all decoding errors resulted from nonconvergent decoder's behavior (we declare nonconvergence if the number of iterations reaches the upper limit that in our simulations was set at 2000 iterations). Similar result was observed for Gallager codes with longer block lengths in [13].

Fig. 10 shows the performance of $(1536, 192, -)$, $\{1536, -, 3\}$ Gallager code generated by Construction 2 in AWGN channel (while it is difficult to make a fully meaningful comparison between the result of Fig. 10 and the results reported in literature for other classes of codes due to different code rates, decoding complexities etc., the reader may be interested in [3] where certain results for short-frame turbo codes are reported).

B. Performance of Gallager Codes in Fully Interleaved Flat Rayleigh-Fading Channels

We simulated the performance of the sum-product iterative decoder for short frame Gallager codes in fully interleaved flat Rayleigh-fading channels, i.e., we assumed that fading amplitudes at different time instants are independent Rayleigh random variables with a unit second moment. The assumption of fully interleaved Rayleigh-fading channel may be somewhat unrealistic in IS-95 systems since only a single code frame is inter-

¹We thank one of the reviewers for pointing out available computational savings in this step.

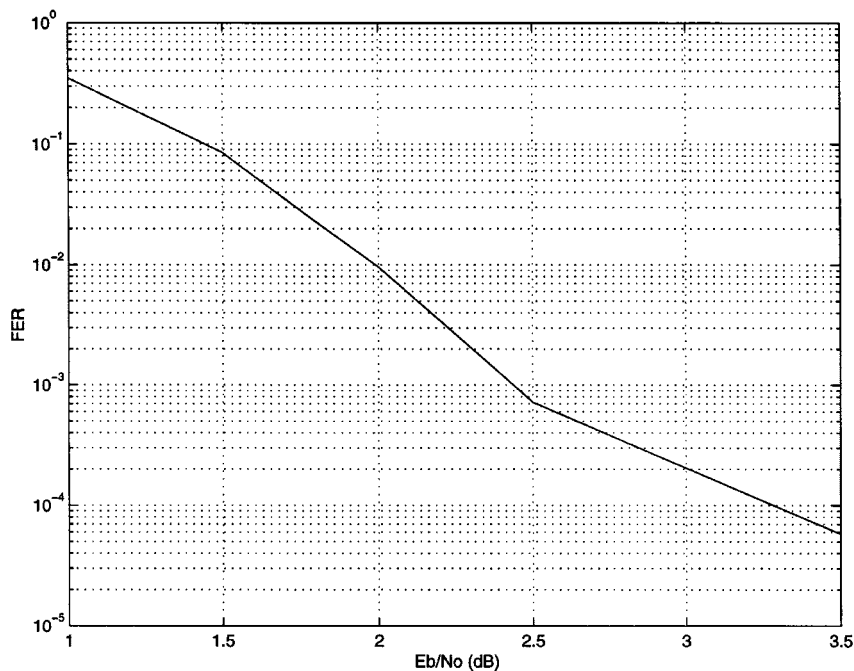


Fig. 10. Frame error rates for ($N = 1536, K = 192, -$) Gallager code with the sum-product decoder in the AWGN channel.

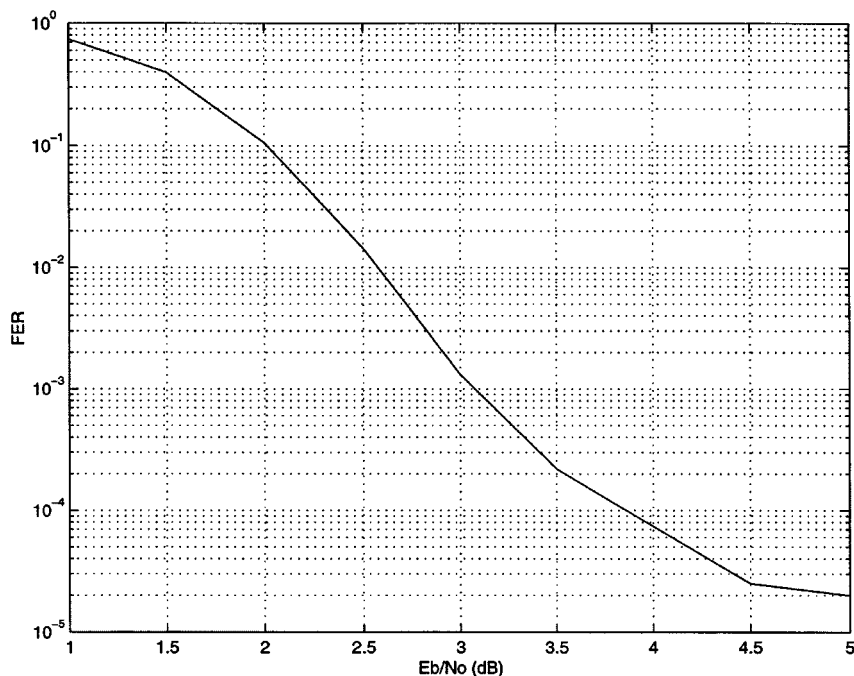


Fig. 11. Performance of ($N = 1536, K = 192, -$) Gallager code in flat Rayleigh-fading channels. The channel state information is made available to the decoder.

leaved. However, good code performance in this type of channels may be an indication of good performance in channels with correlated fading (and certainly warrants further investigation in this direction).

We considered ± 1 antipodal signaling. We assumed perfect synchronization at the receiver. Then the received signal is

$$y(t) = \alpha x(t) + n(t)$$

where $x(t)$ is the transmitted signal (± 1), $n(t)$ is the white Gaussian noise with variance σ^2 and α is a Rayleigh random variable such that $E[\alpha^2]$ is 1.

In the first set of simulations we assumed that channel state information (CSI) is available to the decoder; i.e., the decoder knows the fading amplitudes for each transmitted symbol. Fig. 11 shows the performance of the sum-product iterative decoder for Gallager codes in a flat Rayleigh fading channel (some results regarding the performance of turbo codes in fading channels can be found in [3] and, most recently, in [25]).

If the channel state information (Fig. 12) is not available to the decoder, the performance deteriorates by about 1 dB (we marginalize over α to obtain an effective likelihood ratio [25], [26]).

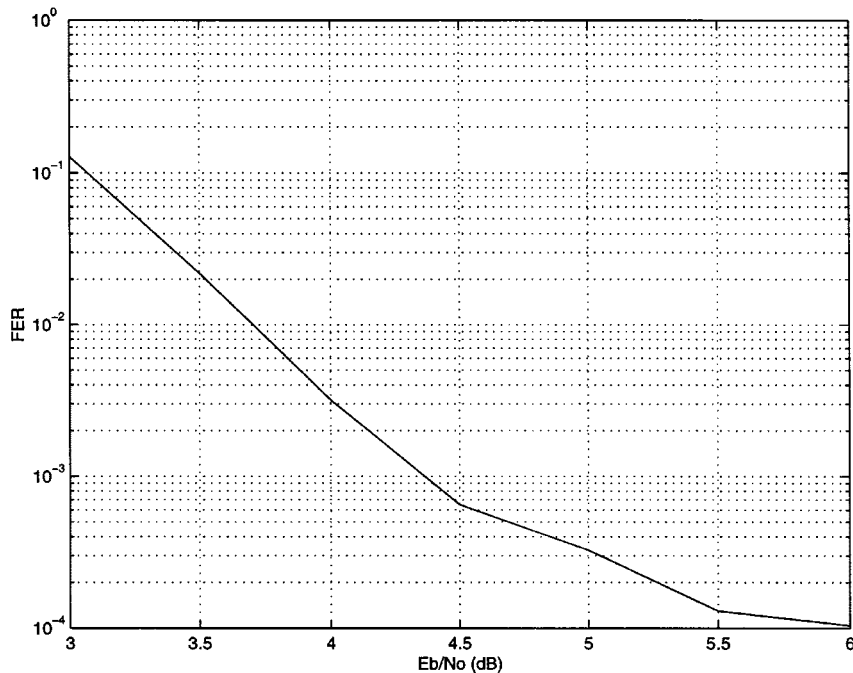


Fig. 12. Performance of ($N = 1536$, $K = 192$, —) Gallager code in a flat Rayleigh-fading channel with no channel state information.

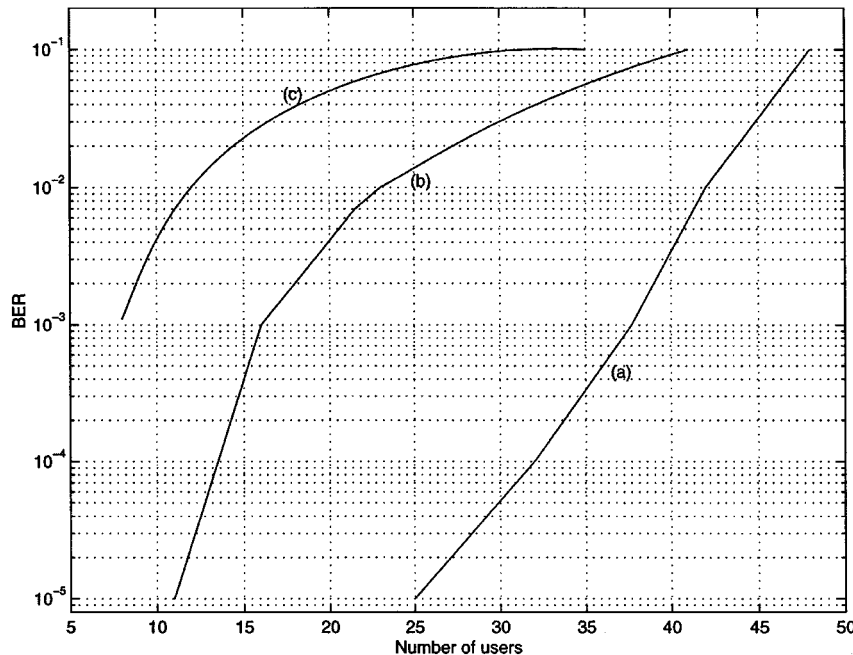


Fig. 13. Data BER versus number of users in a cell for: (a) direct sequence (DS) CDMA that employs Gallager codes; (b) DS-CDMA with low rate orthogonal codes [4]; (c) uncoded DS-CDMA.

VII. CAPACITY OF A CDMA CELL

It is known [27] that the system capacity of the reverse CDMA link assuming single cell and perfect receiver synchronization, can be approximated as

$$n \approx \frac{\eta}{E_b/I_0} \quad (10)$$

where n is the number of users, η is the processing gain, and E_b/I_0 is signal-to-interference ratio.

We now use expression (10) to evaluate the CDMA cell capacity, i.e., the number of admissible users for a given bit-error

rate (BER). We assume coherent detection in both forward and reverse links. The system capacity can be computed by using bit-error probabilities for Gallager codes in AWGN environment. From Fig. 10 and (10) we obtain the CDMA cell capacity versus signal-to-interference ratio. The result is shown in Fig. 13.

Fig. 13 also compares the cell capacity of a CDMA system that uses Gallager codes, with other CDMA systems built on similar principles. Ormondroyd and Maxey [4] consider a CDMA system that uses very low rate orthogonal convolutional codes. In particular, they consider rate 1/64 orthogonal convolutional codes. To make a fair comparison with their

results we assumed that the processing gain η is 64 in (10) (where 64 is formed is a product of the reciprocal of the code rate (1/8) and an additional spreading factor of 8). Note that the decoding complexity per information bit of rate 1/64 low rate orthogonal convolutional codes is comparable [27] with the decoding complexity (per information bit per iteration) of {1536, $-$, 3} Gallager codes, but Gallager codes will typically require several iterations for decoding.

In digital speech transmission, typical data BERs that the system can tolerate are on the order of 10^{-3} . From Fig. 13, we observe that an uncoded CDMA system does not admit more than seven users without an increase in the data BER, i.e., without a reduction in quality of service. The CDMA system with orthogonal convolutional codes supports 16 users at BERs of 10^{-3} , and the CDMA system with Gallager codes supports 37 users. Hence, a CDMA system with Gallager codes achieves more than a twofold increase in the system capacity compared with [4], and more than a fivefold increase in the system capacity compared with an uncoded CDMA system.

VIII. DISCUSSION

In the present work we searched for good short frame codes that could be used in CDMA applications in digital speech transmission where processing delay constraints are critical. We were particularly interested in codes for which very efficient (hardware or software) decoding algorithms exist. Iterative decoding algorithms are the focal point of most recent research effort in the coding theory, and they were the primary objects of interest in our work.

We found a family of low rate Gallager codes that exhibits a good performance under the desired constraints. Theoretical results show that Gallager codes are good even for relatively short block lengths. CDMA systems that use Gallager codes could outperform other CDMA systems that use weaker error-protection codes.

Gallager codes admit highly parallel software and hardware implementations that guarantee high speed data processing. Moreover, low complexity iterative decoders are readily available for Gallager codes.

We note that Richardson *et al.* [9], [28] have recently provided analysis and design guidelines for irregular Gallager codes (introduced by Luby *et al.* [29]) that perform remarkably well for long block lengths, coming even closer to the Shannon limit on AWGN channels than turbo codes do. It is conceivable that irregular Gallager codes might also provide some performance advantage for the relatively short block lengths considered here, but we have not investigated this possibility.

Since Gallager codes exhibit good performance in AWGN channels and, perhaps, more importantly, in fading channels that arise frequently in mobile communications, we conclude that Gallager codes could, perhaps should, become the error-control scheme of choice in future generations of CDMA systems.

ACKNOWLEDGMENT

The authors would like to thank D. MacKay, B. Frey, and the anonymous reviewers for many helpful comments and suggestions.

REFERENCES

- [1] A. J. Viterbi, "Very low rate convolutional codes for maximum theoretical performance of spread-spectrum multiple-access channels," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 641–649, May 1990.
- [2] J. Chaib and H. Leib, "Benefits of error control coding for CDMA," in *Proc. 17th Biennial Symp. Communications*, Kingston, ON, Canada, May 30–June 1, 1994, pp. 88–91.
- [3] P. Jung, "Comparison of turbo-code decoders applied to short frame transmission systems," *IEEE J. Select. Areas Commun.*, vol. 14, pp. 530–537, Apr. 1996.
- [4] R. F. Ormondroyd and J. J. Maxey, "Performance of low-rate orthogonal convolutional codes in DS-CDMA applications," *IEEE Trans. Veh. Technol.*, vol. 46, pp. 320–328, May 1997.
- [5] N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *Eur. Trans. Telecommun.*, vol. 6, pp. 513–525, Sept./Oct. 1995.
- [6] J. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533–547, Sept. 1981.
- [7] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.
- [8] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.
- [9] T. Richardson and R. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, to be published.
- [10] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, submitted for publication.
- [11] 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.
- [12] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, pp. 257–286, Feb. 1989.
- [13] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Mar. 1999.
- [14] C. R. P. Hartmann and L. D. Rudolph, "An optimum symbol-by-symbol decoding rule for linear codes," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 514–517, Sept. 1976.
- [15] G. D. Forney Jr., "Codes on graphs: Generalized state realizations," *IEEE Trans. Inform. Theory*, to be published.
- [16] D. J. C. MacKay and R. M. Neal, "Good codes based on very sparse matrices," in *Cryptography and Coding, 5th IMA Conference*, ser. Lecture Notes in Computer Science, C. Boyd, Ed., 1995, vol. 1025, pp. 110–111.
- [17] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 260–264, Feb. 1998.
- [18] J. A. Erfanian, S. Pasupathy, and G. Gulak, "Reduced complexity symbol detectors with parallel structures for ISI channels," *IEEE Trans. Commun.*, vol. 42, pp. 1661–1671, Feb./Mar./Apr. 1994.
- [19] S. Y. Kung, *VLSI Array Processors*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
- [20] L. Goke and G. Lipovski, "Banyan network for partitioning multiprocessor systems," in *Proc. ICC'73*, 1973, pp. 21–30.
- [21] H. S. Kim, "Asynchronous Transfer Mode Switch for Broadband ISDN," Ph.D. dissertation, Dept. of Elect. Eng., Univ. of Toronto, 1990.
- [22] B. J. Frey, "Bayesian networks for pattern classification, data compression and channel coding," Ph.D. dissertation, Dept. of Elect. and Comput. Eng., Univ. of Toronto, 1997.
- [23] G. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *IEEE Int. Conf. Communications (ICC'93)*, Geneva, Switzerland, May 1993, pp. 2.1064–2.1070.
- [24] S. W. Golomb, R. E. Peile, and R. A. Scholtz, *Basic Concepts in Information Theory and Coding: The Adventures of Secret Agent 00111*. New York: Plenum, 1994.
- [25] E. K. Hall and S. G. Wilson, "Design and analysis of turbo codes on Rayleigh fading channels," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 160–174, Feb. 1998.
- [26] R. E. Blahut, *Digital Transmission of Information*. Reading, MA: Addison-Wesley, 1990.
- [27] A. J. Viterbi, *CDMA Principles of Spread Spectrum Communication*. Reading, MA: Addison-Wesley, 1995.
- [28] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of provably good low-density parity check codes," *IEEE Trans. Inform. Theory*, to be published.
- [29] M. G. Luby, M. Mitzenmacher, A. Shokrollahi, and D. A. Spielman, "Improved low density parity check codes using irregular graphs," *IEEE Trans. Inform. Theory*, to be published.



Vladislav Sorokine (M'98) received the degree of Engineer-Physicist with honors from Moscow Engineering Physics Institute (MIFI) in 1985, and the M.A.Sc. and Ph.D. degrees in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 1993 and 1998, respectively.

He is a Systems Engineer with Qualcomm, Inc., San Diego, CA. His research interests include the area of coding theory and wireless communications. He is currently involved in the design and performance analysis of third generation wireless systems.



Frank R. Kschischang (S'83-M'91) received the B.A.Sc. degree with honors from the University of British Columbia, Vancouver, BC, Canada, in 1985, and the M.A.Sc. and Ph.D. degrees from the University of Toronto, Toronto, ON, Canada, in 1988 and 1991, respectively, all in electrical engineering. During his graduate studies, he held a variety of scholarships and fellowships, including the IEEE Communications Society Scholarship. He is a 1999 recipient of the Province of Ontario's Premier Research Excellence Award.

He is an Associate Professor of Electrical and Computer Engineering at the University of Toronto, Toronto, ON, Canada. His research interests include the area of coding theory, particularly in soft-decision decoding algorithms and codes defined on graphs.

Dr. Kschischang is currently an Associate Editor for Coding Theory of the IEEE TRANSACTIONS ON INFORMATION THEORY. He served as Publicity Chair of the 1998 IEEE International Symposium on Information Theory held in Cambridge, MA.



Subbarayan Pasupathy (M'73-SM'81-F'91) was born in Chennai (Madras), India, on September 21, 1940. He received the B.E. degree in telecommunications from the University of Madras in 1963, the M.Tech. degree in electrical engineering from the Indian Institute of Technology, Madras, India, in 1966, and the M.Phil. and Ph.D. degrees in engineering and applied science from Yale University, New Haven, CT, in 1970 and 1972, respectively.

He joined the faculty of the University of Toronto in 1973 and became a Professor of Electrical Engineering in 1983. He has served as the Chairman of the Communications Group and as the Associate Chairman of the Department of Electrical Engineering at the University of Toronto. His research interests are in the areas of communication theory, digital communications, and statistical signal processing.

Dr. Pasupathy is a Registered Professional Engineer in the Province of Ontario. He has served as a Technical Associate Editor for the *IEEE Communications Magazine* (1979-1982) and as an Associate Editor for the *Canadian Electrical Engineering Journal* (1980-1983). During 1982-1989, he was an Area Editor for Data Communications and Modulation for the IEEE TRANSACTIONS ON COMMUNICATIONS. Since 1984, he has been writing a regular column entitled "Light Traffic" for the *IEEE Communications Magazine*. He was elected Fellow of the IEEE in 1991 "for contributions to bandwidth-efficient coding and modulation schemes in digital communication."