

# Low-Rate Turbo-Hadamard Codes

Li Ping, *Member, IEEE*, W. K. Leung, *Student Member, IEEE*, and K. Y. Wu, *Student Member, IEEE*

**Abstract**—This paper is concerned with a class of low-rate codes constructed from Hadamard code arrays. A recursive encoding principle is employed to introduce an interleaving gain. Very simple trellis codes with only two or four states are sufficient for this purpose, and the decoding cost involved in the trellis part is typically negligible. Both simulation and analytical results are provided to demonstrate the advantages of the proposed scheme. The proposed scheme is of theoretical interest as it can achieve performance of  $\text{BER} = 10^{-5}$  at  $E_b/N_0 \approx -1.2$  dB (only about 0.4 dB away from the ultimate low-rate Shannon limit) with an information block size of 65534. To the authors' knowledge, this is the best result achieved to date with respect to the ultimate Shannon limit. With regard to practical issues, the decoding complexity of the proposed code is considerably lower than that of existing low-rate turbo-type codes with comparable performance.

**Index Terms**—A posteriori probability (APP) decoding, Hadamard codes, iterative decoding, spread spectrum, turbo codes.

## I. INTRODUCTION

THE ultimate capacity of an additive white Gaussian noise (AWGN) channel is the Shannon limit of  $E_b/N_0 \approx -1.6$  dB for very-low-rate codes. One important application of low-rate codes is in code-division multiple-access (CDMA) systems [1]–[3]. Traditionally, coding and spreading are separated in CDMA systems (e.g., IS-95). Recently, it has been demonstrated [4] that an extra coding gain of about 1.3 dB (or an equivalent capacity gain [1]–[3]) can be obtained, by treating the coding and spreading functions in IS-95 as a concatenated coding scheme and applying turbo-type iterative decoding.

The work in [4] implies that a well-designed low-rate code can further enhance the performance of a CDMA system. Traditional low-rate codes, such as Hadamard codes [5], [6] and super-orthogonal convolutional codes [7], are not well suited for this purpose due to their relatively low coding gain. Low-rate concatenated codes that have recently emerged [8]–[12] have demonstrated good potential using turbo [13] or low-density parity-check (LDPC) [14], [15] coding principles. Among these codes, the methods in [8], [9] have relatively higher performance but also higher complexity. This is due to the complicated trellis structures involved.

This paper is concerned with a class of low-rate codes constructed from Hadamard code arrays [12], [16]. A recursive encoding principle is employed to introduce an interleaving gain. Very simple trellis codes with only two or four states are sufficient for this purpose, and the decoding cost involved in the

trellis part is typically negligible. We will develop a so-called *a posteriori* probability (APP) fast Hadamard transform (FHT) (APP-FHT) for the soft-in/soft-out decoding of Hadamard codes. Its efficiency is comparable to that of the well-known FHT [6]. (FHT is used for the soft-in/hard-out decoding of Hadamard codes.) FHT and APP-FHT form the core part of the decoder in the proposed scheme.

Both simulation and analytical results are provided to demonstrate the advantages of the proposed scheme. In particular, the proposed scheme can achieve a bit-error rate ( $\text{BER}$ ) =  $10^{-5}$  at  $E_b/N_0 \approx -1.2$  dB (only about 0.4 dB away from the ultimate Shannon limit of  $-1.6$  dB) with an information block size of 65534, which is the best result known to us with respect to the ultimate low-rate Shannon limit. We will also show that the overall decoding complexity of the proposed scheme is considerably lower than that of existing low-rate turbo codes with comparable performance.

## II. HADAMARD CODES

Hadamard codes, equivalent to the Walsh sequences or the first-order Reed–Muller codes up to interleaving [5], have been widely used in communication systems for synchronization and bandwidth spreading. However, the coding gain of Hadamard codes is relatively low, e.g., a length-4096 bi-orthogonal Hadamard code can achieve  $\text{BER} = 10^{-5}$  at  $E_b/N_0 \approx 4$  dB, about 6 dB away from the ultimate Shannon limit.

FHT is an efficient soft-in/hard-out decoding algorithm for Hadamard codes. In this section, an optimal soft-in/soft-out APP-FHT decoding algorithm will be developed for Hadamard codes. FHT and APP-FHT will form the core part in the iterative decoding of the turbo-Hadamard codes discussed later.

### A. Hadamard Matrix and Hadamard Codes

Start with a  $1 \times 1$  Hadamard matrix  $\mathbf{H}_1 = [+1]$ . An  $n \times n$  (with  $n = 2^r$ ) Hadamard matrix  $\mathbf{H}_n$  over  $\{+1, -1\}$  (in the Sylvester form [5]) can be constructed recursively as

$$\mathbf{H}_n = \begin{bmatrix} +\mathbf{H}_{n/2} & +\mathbf{H}_{n/2} \\ +\mathbf{H}_{n/2} & -\mathbf{H}_{n/2} \end{bmatrix}. \quad (1)$$

We will use  $\mathbf{H}$  to denote a length- $2^r$  bi-orthogonal Hadamard code, carrying  $r + 1$  bits of information. We call  $r$  the order of the Hadamard code. The codeword set of  $\mathbf{H}$  is formed by the columns of  $\pm\mathbf{H}_n$ , denoted by  $\{\pm\mathbf{h}^j : j = 0, 1, \dots, 2^r - 1\}$ . In a systematic encoding, the bit indexes  $\{0, 1, 2, 4, \dots, 2^{r-1}\}$  are used as information positions for  $\mathbf{H}$ . For simplicity, and without confusion, we will sometimes use  $\mathbf{H}$  to represent  $\mathbf{H}_n$ .

### B. Fast Hadamard Transform (FHT)

As we will see, the matrix–vector product  $\mathbf{y} = \mathbf{H}_n\mathbf{x}$  is an important operation in the decoder discussed in what follows.

Manuscript received January 30, 2001; revised July 23, 2002. This work was supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project CityU 1110/00E).

The authors are with the Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong (e-mail: eeliping@cityu.edu.hk).

Communicated by R. Urbanke, Associate Editor for Coding Techniques.

Digital Object Identifier 10.1109/TIT.2003.820018

It requires  $2^r(2^r - 1)$  additions using a straightforward implementation. This cost can be greatly reduced by the  $n$ -point FHT technique outlined below.

From (1), for  $n = 2^r$  and  $r \geq 1$

$$\mathbf{y} = \mathbf{H}_n \mathbf{x} = \begin{bmatrix} +\mathbf{I}_{n/2} & +\mathbf{I}_{n/2} \\ +\mathbf{I}_{n/2} & -\mathbf{I}_{n/2} \end{bmatrix} \begin{bmatrix} \mathbf{H}_{n/2} \mathbf{x}' \\ \mathbf{H}_{n/2} \mathbf{x}'' \end{bmatrix} \quad (2)$$

where  $\mathbf{I}_{n/2}$  is an  $(n/2) \times (n/2)$  identity matrix, and  $\mathbf{x}'$  and  $\mathbf{x}''$  contain the first and second halves of  $\mathbf{x}$ , respectively, i.e.,

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}' \\ \mathbf{x}'' \end{pmatrix}.$$

Correspondingly, let

$$\mathbf{y} = \begin{pmatrix} \mathbf{y}' \\ \mathbf{y}'' \end{pmatrix}.$$

Thus,  $\mathbf{H}_n \mathbf{x}$  can be evaluated according to the flow graph in Fig. 1(a). Notice that the matrix-vector product in

$$\begin{bmatrix} +\mathbf{I}_{n/2} & +\mathbf{I}_{n/2} \\ +\mathbf{I}_{n/2} & -\mathbf{I}_{n/2} \end{bmatrix}$$

costs exactly  $n$  additions. The factorization in (2) can be continued until  $\mathbf{H}_1$  appears, which can be realized directly. This leads to an  $r$ -stage decomposition for the evaluation of  $\mathbf{H}_n \mathbf{x}$ , costing  $2^r$  additions per stage and  $r2^r$  additions in total. For example, Fig. 1(b) illustrates a flow graph for evaluating  $\mathbf{y} = \mathbf{H}_4 \mathbf{x}$  (i.e.,  $n = 4$  and  $r = 2$ ).

The above factorization process is very similar to that used in a fast Fourier transform (FFT) [17]. The resultant butterfly-type flow graph in Fig. 1(a) is also very similar to that for an FFT.

In some special cases, the reduced forms of FHT lead to further cost reduction. See Appendix A.

### C. Soft-In/Soft-Out APP Decoding of Hadamard Codes

We now consider the APP decoding technique for Hadamard codes. Assume that all of the codewords have independent and equal probabilities of occurrence. Let  $\mathbf{c} = \{c[i]\}$  be the transmitted codeword and  $\mathbf{x} = \{x[i]\}$  be the noisy observation of  $\mathbf{c}$ . The APP decoding of a code over  $\{+1, -1\}$  involves evaluating the logarithm-likelihood ratio (LLR) values [13], [18], [19],

$$L[i] = \log \frac{\Pr(c[i] = +1|\mathbf{x})}{\Pr(c[i] = -1|\mathbf{x})} = \log \frac{\Pr(\mathbf{x}|c[i] = +1)}{\Pr(\mathbf{x}|c[i] = -1)}, \quad i = 0, 1, 2, \dots \quad (3)$$

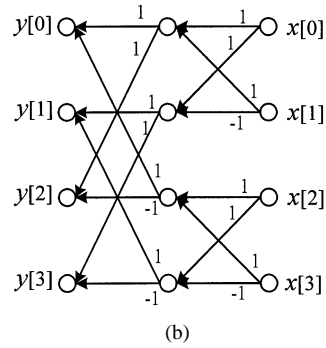
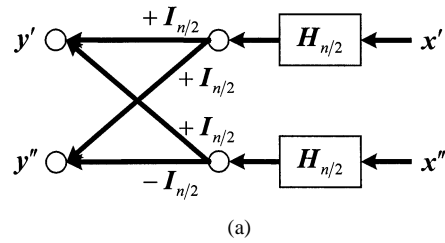


Fig. 1. (a) The graphical representation of (2). (b) The flow graph of a four-point ( $n = 4$  and  $r = 2$ ) FHT for realizing  $\mathbf{y} = \mathbf{H}_4 \mathbf{x}$ .

Here we assume that no *a priori* information is available. (See footnote 1 for the case of nonzero *a priori* information.) According to [18], (3) can be expanded as

$$L[i] = \log \frac{\Pr(\mathbf{x}|c[i] = +1)}{\Pr(\mathbf{x}|c[i] = -1)} = \log \frac{\sum_{c[i]=+1} \Pr(\mathbf{x}|\mathbf{c})}{\sum_{c[i]=-1} \Pr(\mathbf{x}|\mathbf{c})}, \quad i = 0, 1, 2, \dots \quad (4)$$

where the summations are over all  $\mathbf{c}$  constrained by  $c[i] = +1$  and  $c[i] = -1$ , respectively. Denote by  $H[i, j]$  the  $(i, j)$ th element of  $\mathbf{H}$ . The codeword  $\mathbf{c}$  of the Hadamard code is selected from  $\{\pm \mathbf{h}^j\}$  (the column set of  $\mathbf{H}$ , see Section II-A). Thus,  $c[i] = \pm H[i, j]$  for some  $j$ . There are two situations for  $c[i] = +1$ , i.e.,  $\mathbf{c} = \mathbf{h}^j$  with  $H[i, j] = +1$  or  $\mathbf{c} = -\mathbf{h}^j$  with  $H[i, j] = -1$ . Similarly,  $c[i] = -1$  implies that either  $\mathbf{c} = \mathbf{h}^j$  with  $H[i, j] = -1$  or  $\mathbf{c} = -\mathbf{h}^j$  with  $H[i, j] = +1$ . Thus, (4) can be expanded for an AWGN channel as [18] (5a) at the bottom of the page, where  $\sigma^2$  is the noise variance. In (5a),  $\gamma(\pm \mathbf{h}^j)$  is defined by

$$\gamma(\pm \mathbf{h}^j) \equiv \exp\left(\frac{1}{2} \langle \pm \mathbf{h}^j, \tilde{\mathbf{L}} \rangle\right), \quad j = 0, 1, \dots, 2^r - 1 \quad (5b)$$

$$\begin{aligned} L[i] &= \log \frac{\sum_{H[i,j]=+1} \exp\left(-\frac{\|\mathbf{h}^j - \mathbf{x}\|^2}{2\sigma^2}\right) + \sum_{H[i,j]=-1} \exp\left(-\frac{\|-\mathbf{h}^j - \mathbf{x}\|^2}{2\sigma^2}\right)}{\sum_{H[i,j]=-1} \exp\left(-\frac{\|\mathbf{h}^j - \mathbf{x}\|^2}{2\sigma^2}\right) + \sum_{H[i,j]=+1} \exp\left(-\frac{\|-\mathbf{h}^j - \mathbf{x}\|^2}{2\sigma^2}\right)} \\ &= \log \frac{\sum_{H[i,j]=\pm 1} \gamma(\pm \mathbf{h}^j)}{\sum_{H[i,j]=\mp 1} \gamma(\pm \mathbf{h}^j)}, \quad i = 0, 1, \dots, 2^r - 1 \end{aligned} \quad (5a)$$

where  $\tilde{\mathbf{L}} = 2\mathbf{x}/\sigma^2$  is directly calculated from the channel outputs.<sup>1</sup>

#### D. Efficient Techniques for Evaluating (5a) and (5b)

The inner products  $\langle \mathbf{h}^j, \tilde{\mathbf{L}} \rangle$ ,  $j = 0, 1, \dots, 2^r - 1$ , are elements in  $\mathbf{H}_n \tilde{\mathbf{L}} = \mathbf{H}_n^t \tilde{\mathbf{L}}$ . (Recall that  $\mathbf{H}_n$  is symmetric.) Thus,  $\{\gamma(\pm \mathbf{h}^j)\}$  can be evaluated by an FHT and  $2n$  exponential functions, providing an efficient way to evaluate (5b).

We now focus on the evaluation of (5a), assuming that all  $\{\gamma(\pm \mathbf{h}^j)\}$  are available. The key issue is how to carry out the summations in (5a) efficiently. Fix  $i$  and consider a particular  $H[i, j']$ . The summations in (5a) are related to the sign of  $H[i, j']$  as follows.

- When  $H[i, j'] = +1$

$$\begin{aligned} \begin{bmatrix} \sum_{H[i,j]=\pm 1} \gamma(\pm \mathbf{h}^j) \\ \sum_{H[i,j]=\mp 1} \gamma(\pm \mathbf{h}^j) \end{bmatrix} &= \begin{bmatrix} \gamma(+\mathbf{h}^{j'}) + \sum_{\substack{j \neq j' \\ H[i,j]=\pm 1}} \gamma(\pm \mathbf{h}^j) \\ \gamma(-\mathbf{h}^{j'}) + \sum_{\substack{j \neq j' \\ H[i,j]=\mp 1}} \gamma(\pm \mathbf{h}^j) \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \gamma(+\mathbf{h}^{j'}) \\ \gamma(-\mathbf{h}^{j'}) \end{bmatrix} \\ &\quad + \begin{bmatrix} \sum_{\substack{j \neq j' \\ H[i,j]=\pm 1}} \gamma(\pm \mathbf{h}^j) \\ \sum_{\substack{j \neq j' \\ H[i,j]=\mp 1}} \gamma(\pm \mathbf{h}^j) \end{bmatrix}. \end{aligned} \quad (6a)$$

- When  $H[i, j'] = -1$

$$\begin{aligned} \begin{bmatrix} \sum_{H[i,j]=\pm 1} \gamma(\pm \mathbf{h}^j) \\ \sum_{H[i,j]=\mp 1} \gamma(\pm \mathbf{h}^j) \end{bmatrix} &= \begin{bmatrix} \gamma(-\mathbf{h}^{j'}) + \sum_{\substack{j \neq j' \\ H[i,j]=\pm 1}} \gamma(\pm \mathbf{h}^j) \\ \gamma(+\mathbf{h}^{j'}) + \sum_{\substack{j \neq j' \\ H[i,j]=\mp 1}} \gamma(\pm \mathbf{h}^j) \end{bmatrix} \\ &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \gamma(+\mathbf{h}^{j'}) \\ \gamma(-\mathbf{h}^{j'}) \end{bmatrix} \\ &\quad + \begin{bmatrix} \sum_{\substack{j \neq j' \\ H[i,j]=\pm 1}} \gamma(\pm \mathbf{h}^j) \\ \sum_{\substack{j \neq j' \\ H[i,j]=\mp 1}} \gamma(\pm \mathbf{h}^j) \end{bmatrix}. \end{aligned} \quad (6b)$$

The relationship in (6) can be expressed in a more compact matrix form. We first introduce two  $2^r \times 1$  sized compound vectors  $\mathbf{a}$  and  $\mathbf{b}$  as follows. (*Note*: Every entry in  $\mathbf{a}$  or  $\mathbf{b}$  is a pair of real numbers.)

$$\mathbf{a}[j] = \begin{bmatrix} \gamma(+\mathbf{h}^j) \\ \gamma(-\mathbf{h}^j) \end{bmatrix}, \quad j = 0, 1, \dots, 2^r - 1 \quad (7a)$$

<sup>1</sup>For nonzero *a priori* LLR values

$$\tilde{\mathbf{L}} = \left\{ \tilde{L}[i] \right\} = \left\{ \frac{2x[i]}{\sigma^2} + \log \left( \frac{\Pr(c[i] = +1)}{\Pr(c[i] = -1)} \right) \right\}.$$

Here  $\{\Pr(c[i] = \pm 1)\}$  are the *a priori* probabilities and they are approximated by the extrinsic information during an iterative decoding.

$$\mathbf{b}[i] = \begin{bmatrix} \sum_{H[i,j]=\pm 1} \gamma(\pm \mathbf{h}^j) \\ \sum_{H[i,j]=\mp 1} \gamma(\pm \mathbf{h}^j) \end{bmatrix}, \quad i = 0, 1, \dots, 2^r - 1. \quad (7b)$$

From (6), it is seen that  $\mathbf{b}$  and  $\mathbf{a}$  are related as shown in (8a) at the bottom of the page, where

$$\mathbf{J} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad \hat{\mathbf{J}} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \quad (8b)$$

The matrix form of (8a) is

$$\mathbf{b} = \mathbf{Q}_n \mathbf{a} \quad (9)$$

where  $\mathbf{Q}_n$  is an  $n \times n$  compound matrix (or a  $2n \times 2n$  ordinary matrix) obtained from  $\mathbf{H}_n$  by the following replacements:

$$+1 \rightarrow \mathbf{J} \quad \text{and} \quad -1 \rightarrow \hat{\mathbf{J}}. \quad (10)$$

Based on (1), (9), and (10), for  $n = 2^r$  and  $r \geq 1$ ,  $\mathbf{Q}_n$  can be decomposed as

$$\mathbf{Q}_n = \begin{bmatrix} \mathbf{Q}_{n/2} & \mathbf{Q}_{n/2} \\ \mathbf{Q}_{n/2} & \hat{\mathbf{Q}}_{n/2} \end{bmatrix} \quad (11)$$

with  $\mathbf{Q}_{n/2}$  and  $\hat{\mathbf{Q}}_{n/2}$  obtained from  $\mathbf{H}_{n/2}$  and  $-\mathbf{H}_{n/2}$ , respectively, using the replacements in (10).

For example,

$$\begin{aligned} \mathbf{H}_2 &= \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix} \Rightarrow \mathbf{Q}_2 = \begin{bmatrix} \mathbf{J} & \mathbf{J} \\ \mathbf{J} & \hat{\mathbf{J}} \end{bmatrix} \\ \mathbf{H}_4 &= \begin{bmatrix} +\mathbf{H}_2 & +\mathbf{H}_2 \\ +\mathbf{H}_2 & -\mathbf{H}_2 \end{bmatrix} \Rightarrow \mathbf{Q}_4 = \begin{bmatrix} \mathbf{Q}_2 & \mathbf{Q}_2 \\ \mathbf{Q}_2 & \hat{\mathbf{Q}}_2 \end{bmatrix} \end{aligned} \quad (12a)$$

$$\begin{aligned} -\mathbf{H}_2 &= \begin{bmatrix} -1 & -1 \\ -1 & +1 \end{bmatrix} \Rightarrow \hat{\mathbf{Q}}_2 = \begin{bmatrix} \hat{\mathbf{J}} & \hat{\mathbf{J}} \\ \hat{\mathbf{J}} & \mathbf{J} \end{bmatrix} \quad \text{and} \\ -\mathbf{H}_4 &= \begin{bmatrix} -\mathbf{H}_2 & -\mathbf{H}_2 \\ -\mathbf{H}_2 & +\mathbf{H}_2 \end{bmatrix} \Rightarrow \hat{\mathbf{Q}}_4 = \begin{bmatrix} \hat{\mathbf{Q}}_2 & \hat{\mathbf{Q}}_2 \\ \hat{\mathbf{Q}}_2 & \mathbf{Q}_2 \end{bmatrix}. \end{aligned} \quad (12b)$$

A direct computation of (9) requires  $2 \times 2^r \times (2^r - 1)$  additions. This cost can be reduced to  $2r \times 2^r$  additions by the following technique. It is easy to verify that

$$\hat{\mathbf{Q}}_{n/2} = \hat{\mathbf{I}} \mathbf{Q}_{n/2} \quad (13)$$

where  $\hat{\mathbf{I}} = \text{diag}\{\hat{\mathbf{J}}, \hat{\mathbf{J}}, \dots, \hat{\mathbf{J}}\}$  is a block-diagonal matrix. (For simplicity, we will not indicate the orders of  $\hat{\mathbf{I}}$  and  $\hat{\mathbf{I}}$  as they are clear from the context.) Based on (11) and (13), (9) can be decomposed as

$$\mathbf{b} = \mathbf{Q}_n \mathbf{a} = \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & \hat{\mathbf{I}} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{n/2} \mathbf{a}' \\ \mathbf{Q}_{n/2} \mathbf{a}'' \end{bmatrix} \quad (14)$$

where  $\mathbf{a}'$  and  $\mathbf{a}''$  contain the first and second halves of  $\mathbf{a}$ , respectively. Similarly to (2), the factorization in (14) can be continued until  $\mathbf{Q}_1 \equiv \mathbf{J}$  appears. This evaluation technique is called an  $n$ -point APP-FHT. The similarity between FHT (see Section II-B) and APP-FHT is apparent.

$$\mathbf{b}[i] = \begin{cases} \mathbf{J} \times \mathbf{a}[j] + \text{contributions from other entries of } \mathbf{a}, & \text{if } H[i, j] = +1 \\ \hat{\mathbf{J}} \times \mathbf{a}[j] + \text{contributions from other entries of } \mathbf{a}, & \text{if } H[i, j] = -1 \end{cases} \quad (8a)$$

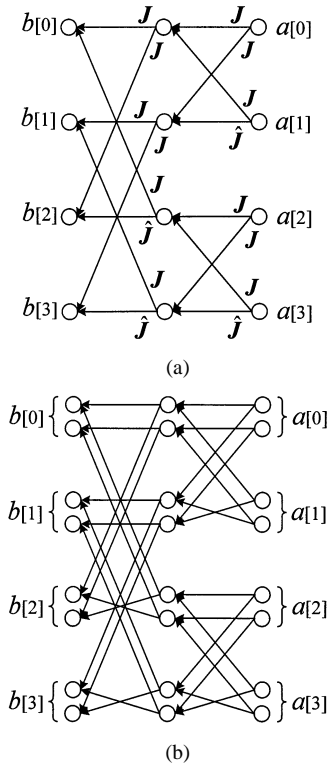


Fig. 2. An example of a four-point APP-FHT. (a) Directed form, where every node represents a pair of real values. (b) Expanded form, where every node represents a real value, and every branch has a unity gain.

An illustration of a four-point APP-FHT in its directed form is shown in Fig. 2(a). This result can also be obtained directly by applying the replacements in (10) to Fig. 1(b). Every node in Fig. 2(a) represents two real variables, which in turn can be split into two nodes, each representing a real variable, see Fig. 2(b). The number of nodes and additions in Fig. 2(b) is doubled compared to those in the original flow graph for FHT in Fig. 1(b). Therefore the total cost of an APP-FHT is  $2r \times 2^r$  additions. In turbo-type decoding, only the LLR values at information positions are required (see Section III). The decoding cost can be reduced to  $6 \times 2^r$  additions by a reduced output APP-FHT (similar to the reduced output FHT in Appendix A).

The APP decoding of a length- $2^r$  Hadamard code is summarized as follows.

*Algorithm 1. Fast APP Decoding of a Hadamard Code.*

Step 1) Obtain  $\mathbf{a}$  by performing an FHT on  $\hat{\mathbf{L}}$  (followed by exponential functions).

Step 2) Obtain  $\mathbf{b}$  by performing an APP-FHT on  $\mathbf{a}$  (followed by logarithm and subtraction operations).

### III. TURBO-HADAMARD CODES

The proposed turbo-Hadamard code involves a hybrid concatenation of convolutional codes and Hadamard codes. We will discuss the structure of the component codes, followed by that of the overall code. For the convenience of discussion, we will use codewords over  $\{0, 1\}$  in this section, which can be obtained from those in Section II by the standard mapping  $\{+1 \leftrightarrow 0, -1 \leftrightarrow 1\}$ .

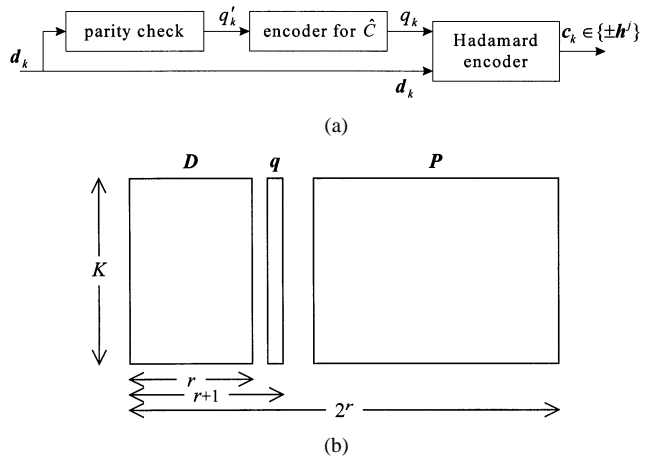


Fig. 3. (a) A convolutional-Hadamard encoder. (b) The structure of a codeword generated in (a).

#### A. The Component Codes: Convolutional-Hadamard Codes

We first introduce the convolutional-Hadamard code (denoted by  $C$ ). It will be used later as a component code in the proposed scheme.  $C$  is generated as follows (see Fig. 3(a)).

- The information bit stream is segmented into blocks  $\{\mathbf{d}_k\}$ . Each block  $\mathbf{d}_k$  contains  $r$  bits.
- Let  $q'_k$  be the parity check of  $\mathbf{d}_k$ .
- Encode  $\mathbf{q}' = \{q'_k\}$  using a rate-1/2 systematic recursive convolutional code  $\hat{C}$ , producing a parity sequence  $\mathbf{q} = \{q_k\}$ . (*Note:*  $\{\mathbf{q}', \mathbf{q}\}$  forms a codeword of  $\hat{C}$ .)
- Encode  $(\mathbf{d}_k, q_k)$  using a Hadamard code, producing  $\mathbf{c}_k = (\mathbf{d}_k, q_k, \mathbf{p}_k) \in \{\pm \mathbf{h}^j\}$ .<sup>2</sup> The resultant sequence  $\mathbf{c} = \{\mathbf{c}_k\}$  forms a codeword in  $C$ .

Fig. 3(b) illustrates a codeword in  $C$ . It is arranged as an array, with  $\mathbf{c}_k$  its  $k$ th row. It can be decomposed as  $\{\mathbf{D}, \mathbf{q}, \mathbf{P}\}$ , where  $\mathbf{d}_k$  is the  $k$ th row of  $\mathbf{D}$ , and  $\mathbf{p}_k$  is the  $k$ th row of  $\mathbf{P}$ . The intermediate vector  $\mathbf{q}'$  does not appear in the final codeword.

The trellis diagram  $T$  of  $C$  can be derived based on the trellis diagram  $\hat{T}$  of  $\hat{C}$ , Fig. 4. We will use the same set of states in  $T$  as in  $\hat{T}$ , with one-to-one correspondence between the indexes. Let  $(q'_k, q_k)$  be the coded bits for the state transition  $s_k \rightarrow s_{k+1}$  in  $\hat{T}$ . Since  $q'_k$  is the parity check of  $\mathbf{d}_k$ , there are  $2^{r-1}$  possible  $\mathbf{d}_k$  to satisfy a particular  $q'_k$ . We thus obtain  $2^{r-1}$  combinations of  $(\mathbf{d}_k, q_k)$ , each leading to a unique Hadamard codeword. We use  $h(q'_k, q_k)$  to denote these  $2^{r-1}$  codewords.  $T$  can then be constructed by placing the  $2^{r-1}$  parallel branches between the corresponding states  $s_k$  and  $s_{k+1}$  labeled by the elements in  $h(q'_k, q_k)$ . This trellis representation will be useful in developing the decoding algorithm and the BER bound for the proposed code in Section III-D.

For example, Fig. 4(b) illustrates a section of the trellis for a convolutional-Hadamard code constructed based on a two-state convolutional code with generator  $1/(1+x)$  and a length-8 ( $r=3$ ) Hadamard code.

<sup>2</sup>A rearrangement of bit positions is implied here since  $\mathbf{c}_k$  is not in the Sylvester form in (1). Note also that, for convenience of discussion, in Section II a Hadamard codeword is presented as a column sequence, while here it is in the form of a row sequence.

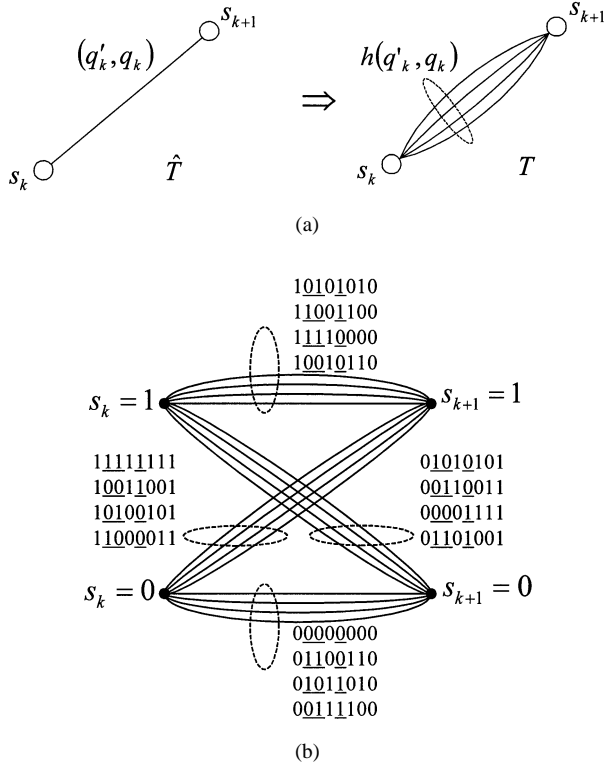


Fig. 4. (a)  $\hat{T}$  represents the trellis of a reference convolutional code  $\hat{C}$ .  $T$  represents the trellis of a convolutional-Hadamard code constructed from  $\hat{T}$ . (b) An example of the  $k$ th section of  $T$  based on a two-state convolutional code  $\hat{C}$  with generator polynomial  $1/(1+x)$  and a length-8 ( $r = 3$ ) Hadamard code. The first, underscored, and remaining bits represent  $q_k$ ,  $d_k$ , and  $p_k$ , respectively.

Let  $v$  be the constraint length of  $\hat{C}$ . We can append  $v$  extra rows to  $D$  to terminate the trellis of the resultant turbo-Hadamard code to state-0. In each of the  $v$  additional rows, we set all the information bits to 0 (**Note:** These zeros are not transmitted.) except one bit. By selecting a proper value for this bit for every additional row, we can always terminate the trellis to state-0.

Alternatively, since the trellis has  $2^v$  states, we can transmit an extra order- $(v - 1)$  Hadamard codeword to convey the termination information. (Each of the  $2^v$  codewords in this extra code represents a unique state.) To ensure reliability, this termination codeword can be transmitted repeatedly. The number of repetitions is determined by the tradeoff between code rate and performance.

**B. The Overall Code: Turbo-Hadamard Codes**

Turbo-Hadamard codes are constructed by concatenating several convolutional-Hadamard codes in parallel. Let  $D^{(m)}$  be the  $m$ th interleaved version of  $D$  and  $(q^{(m)}, P^{(m)})$  be the corresponding redundancy outputs, where  $m = 1, 2, \dots, M$ . The triplet  $\{D^{(m)}, q^{(m)}, P^{(m)}\}$  is referred to as the  $m$ th component codeword (rate =  $r/2^r$ ). After concatenating  $M$  such component codes, the coding rate reduces to  $r/(M2^r)$ . Note that here the information bits are repeated in  $M$  component codes. The transmission efficiency can be improved by redefining the overall codeword as  $\{D, q^{(1)}, P^{(1)}, \dots, q^{(M)}, P^{(M)}\}$ , see

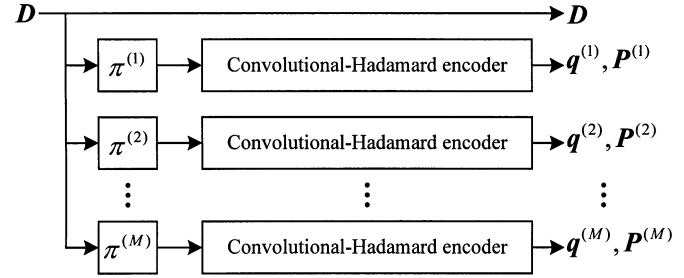


Fig. 5. A turbo-Hadamard encoder with  $M$  component convolutional-Hadamard encoders.  $\{\pi^{(m)}, m = 1, 2, \dots, M\}$  are  $M$  interleavers.

Fig. 5 (i.e.,  $D$  is transmitted only once). The overall code rate is then

$$R = \frac{r}{r + M(2^r - r)} \tag{15}$$

where  $r$  is the order of the Hadamard code.

We now explain the rationale behind the structure of turbo-Hadamard codes. The total parity weight  $W$  of an overall turbo-Hadamard code is given by

$$W = \text{the sum of the parity weights of } M \text{ component codes.} \tag{16}$$

Assume that the right-hand side of (16) involves (approximately)  $M$  independent and identically distributed (i.i.d.) random variables (the i.i.d. assumption). Applying the central limit theorem, for sufficiently large  $M$ ,  $W$  approaches Gaussian. This indicates a close resemblance between a turbo-Hadamard code and a random code. (A random code is obtained, say, by randomly tossing a coin, and has a binomial distance distribution that approaches Gaussian for a sufficiently long length.) It is well known that a random code is a “good code” in probability when length  $\rightarrow \infty$ .

The i.i.d. assumption requires that the parity weights of  $M$  component codes are independent of each other. To ensure this, we should reduce the dependency between input and output weights of each component code. This is the function of the recursive component code  $\hat{C}$  and the random interleavers used in a turbo-Hadamard code. The correlation between input and output weights in a recursive component code is relatively low (compared with a nonrecursive one). For example, Fig. 6 illustrates a codeword of turbo-Hadamard code generated by a weight-1 information word (i.e., with only one nonzero information bit). The white area represents zero elements and the black area represents elements that can be either zeros or ones. Assume that the nonzero bit is located in the  $i$ th row of  $D^{(m)}$ , and  $\hat{C}$  is recursive. Then  $q^{(m)}$  contains about half “0”s and half “1”s from the  $i$ th element onward, and so  $P^{(m)}$  contains about half nonzero Hadamard codewords from the  $i$ th row onward. The weights of the component codewords involved can be regarded as approximately independent of the input weight.

For a linear code, any codeword can be expressed as the sum of the codewords generated by weight-1 information words. Working from this premise, we can also show that the input and output weights of a turbo-Hadamard code are (approximately) uncorrelated for an input weight larger than 1.

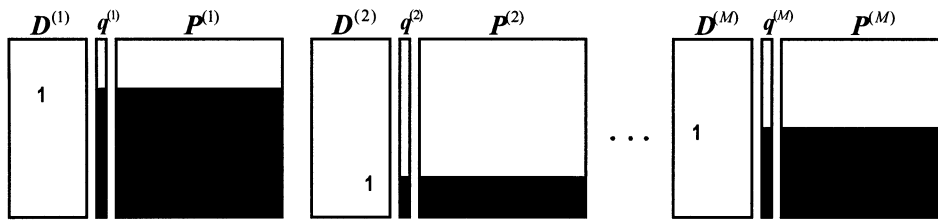


Fig. 6. Illustration of a turbo-Hadamard codeword with only one nonzero information bit.

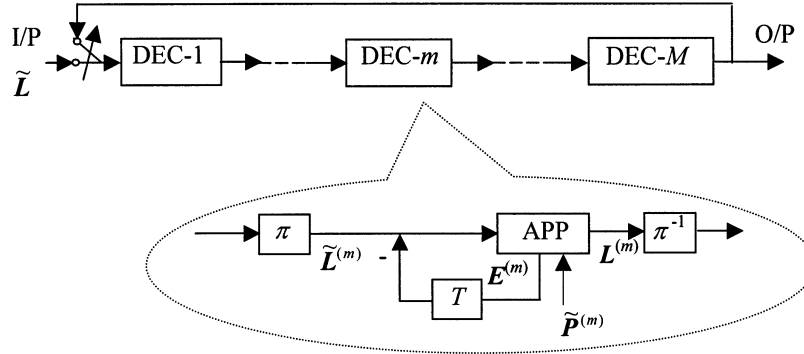


Fig. 7. The global decoding principle for a turbo-Hadamard decoder: “ $T$ ” represents a delay of one iteration and “ $\pi$ ” represents an interleaver.

Notice that without  $\{q^{(m)}\}$ , the code in Fig. 6 reduces to the simple (nonrecursive) concatenated Hadamard code considered in [12]. In this case, a weight-1 information word results in only one nonzero row in each component code. This implies a strong correlation between input and output weights (low input weight  $\rightarrow$  low output weight). Hence, the i.i.d. assumption is no longer valid. This explains the serious error floor problem observed in the simulation studies in [12]. Using a nonrecursive component convolutional code  $\hat{C}$  can lead to a similar correlation problem.

For a detailed discussion on the role of recursive component codes, see [20].

### C. The Decoder

The turbo-Hadamard code is a special turbo code with multiple component codes. The standard BCJR algorithm can be employed for the local APP decoding, based on the trellis diagram in Fig. 4. Due to the special structure, the FHT and APP-FHT can be employed at the input and output stages, respectively, to reduce the decoding cost. A detailed discussion will be provided in Appendix B. The global iterative decoder structure is illustrated in Fig. 7 [21]. It consists of  $M$  looped local decoders, each responsible for one component code. Between two local decoders, LLRs of information bits are exchanged during decoding.

The variables involved in Fig. 7 are as follows.

- $\tilde{\mathbf{L}}$  The initial *a priori* LLR vector for information bits.
- $\tilde{\mathbf{L}}^{(m)}$  The *a priori* LLR vector for information bits in the  $m$ th component code.
- $\mathbf{L}^{(m)}$  The *a posteriori* LLR vector for information bits in the  $m$ th component code.
- $\tilde{\mathbf{P}}^{(m)}$  The *a priori* LLR vector for parity bits in the  $m$ th component code.

$\mathbf{E}^{(m)}$  The extrinsic LLR vector for the  $m$ th component code, defined by [13]

$$\mathbf{E}^{(m)} = \mathbf{L}^{(m)} - \left( \tilde{\mathbf{L}}^{(m)} - \mathbf{E}_{\text{last iteration}}^{(m)} \right)$$

where  $\mathbf{E}_{\text{last iteration}}^{(m)}$  is the extrinsic LLR vector for the  $m$ th component code generated in the last iteration (with zero initial values).

At the start,  $\tilde{\mathbf{L}}$  is fed into DEC-1. The  $M$  local APP decoders are operated successively. The *a posteriori* LLR vector  $\mathbf{L}^{(m-1)}$  from DEC- $(m-1)$  is used, after appropriate interleaving, as the *a priori* LLR vector  $\tilde{\mathbf{L}}^{(m)}$  for DEC- $m$ .  $\mathbf{L}^{(M)}$  is fed back to DEC-1 for the next iteration. This process is then repeated iteratively. For the first iteration

$$\tilde{\mathbf{L}}^{(m)} = \tilde{\mathbf{L}} + \mathbf{E}^{(1)} + \mathbf{E}^{(2)} + \dots + \mathbf{E}^{(m-1)}$$

and after the first iteration

$$\tilde{\mathbf{L}}^{(m)} = \tilde{\mathbf{L}} + \underbrace{\mathbf{E}^{(m)} + \mathbf{E}^{(m+1)} + \dots + \mathbf{E}^{(M)}}_{\text{from the previous iteration}} + \underbrace{\mathbf{E}^{(1)} + \mathbf{E}^{(2)} + \dots + \mathbf{E}^{(m-1)}}_{\text{from the current iteration}}.$$

The principle is very similar to the concatenated tree (CT) decoder discussed in [21].

### D. Distance Spectra and BER Bounds

The distance spectrum of a convolutional-Hadamard code can be obtained using its trellis description [22], [23]. Define the *input-output weight enumerating function* (IOWEF) of a code  $C$  [24] as

$$A^C(W, Y) = \sum_{w, y} A_{w, y} W^w Y^y \quad (17)$$

where  $A_{w, y}$  denotes the number of weight- $y$  codewords generated by weight- $w$  information words. Notice that for a conven-

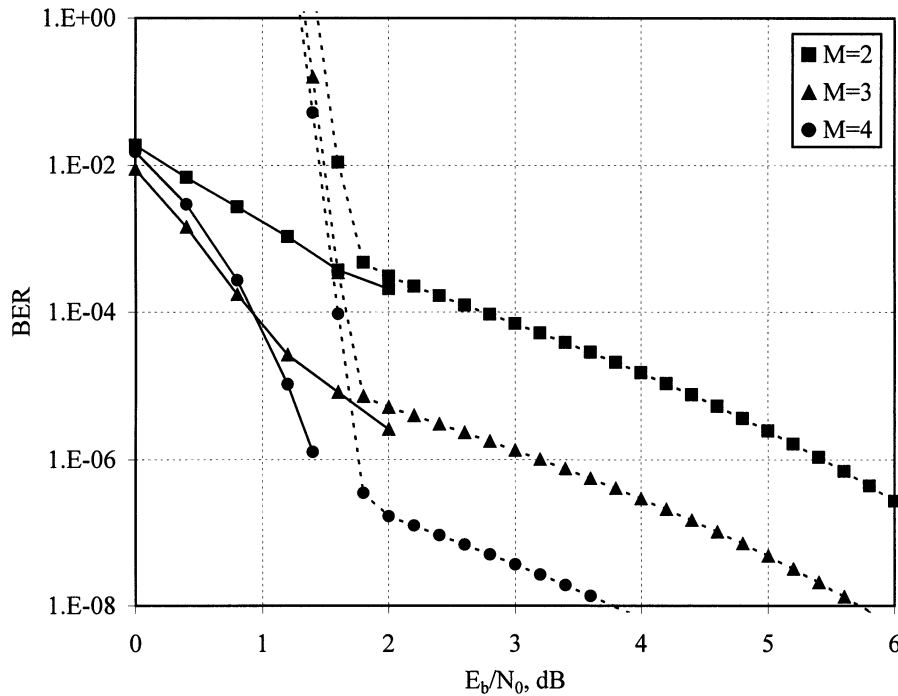


Fig. 8. Performance of turbo-Hadamard codes with  $r = 5$ ,  $N = 200$ ,  $S = 4$ , and  $G(x) = (1 + x)/(1 + x + x^2)$ . The solid lines represent the simulated results and the dashed lines represent the union bounds.

tional code,  $A_{w,y}$  is usually an integer. For a concatenated code,  $A_{w,y}$  is defined in the form of an ensemble weight distribution, which is a probability measurement averaged over all possible interleavers (for a detailed discussion, see [20], [24]).

There are only three possible weights:  $0$ ,  $2^{r-1}$ , and  $2^r$  associated with each branch (i.e., the weight of a length- $2^r$  Hadamard codeword). Therefore we only need to consider the terms associated with those  $y$  that are multiples of  $2^{r-1}$ . Consequently, the IOWEF approach is computationally more efficient than that based on the *input-redundancy weight enumerating function* (IRWEF) [20].

The *conditional output weight enumerating function* (COWEF) of a code  $C$  is

$$A_w^C(Y) = \sum_y A_{w,y} Y^y \quad (18)$$

which represents the weight distribution of the codewords generated by weight- $w$  information words. Assume that  $C^{(1)}, C^{(2)}, \dots, C^{(M)}$  (each containing  $N$  information bits) are concatenated in parallel using “uniform” interleavers [20] to form an overall code  $C$ . Let

$$\sum_y B_{w,y} Y^y = \prod_{m=1}^M \frac{A_w^{C^{(m)}}(Y)}{\binom{N}{w}}. \quad (19)$$

Here  $B_{w,y}$  is the probability that the combined weight of  $M$  component codes is  $y$ , conditioned on the information weight  $w$ . Notice that the information weight has been earlier counted repeatedly for  $M$  times. If every information bit occurs only once in  $C$  (see Section III-B), the required COWEF  $A_w^C(Y) = \sum_y A_{w,y} Y^y$  of  $C$  can be found by setting [20], [24]

$$A_{w,y} = \binom{N}{w} B_{w,y+(M-1)w}. \quad (20)$$

The IOWEF of  $C$  is given by

$$A^C(W, Y) = \sum_{w=0}^N W^w A_w^C(Y). \quad (21)$$

The BER of  $C$  is upper-bounded [20] by

$$P_b(e) \leq \frac{1}{2} \sum_y \sum_w \frac{w}{N} A_{w,y} \operatorname{erfc} \left( \sqrt{yR \frac{E_b}{N_0}} \right) \quad (22)$$

where  $R$  is the code rate of  $C$ .

#### IV. NUMERICAL RESULTS

We will use  $N$  for the interleaver length,  $M$  for the number of component codes,  $R$  for the code rate,  $r$  for the order of the included Hadamard code,  $S$  for the number of states of the reference convolutional code,  $G(x)$  for the generator polynomial of the reference convolutional code and  $IT$  for the iteration number. Unless specified otherwise,  $IT$  will be set to 30.

##### A. The Impact of the Number of Component Codes

Suppose that we fix the structure of the component codes and increase  $M$ . This has two consequences. First, the weight distribution will be closer to binomial (see (16)), which is beneficial. Second, for a fixed overall  $E_b/N_0$  value, each local component decoder will work in a noisier environment, since the overall code rate reduces with  $M$ . Thus, the output of each component decoder becomes less reliable. The combined effect of these factors is illustrated by the simulation results in Fig. 8. The error floor performance improves consistently with  $M$ , but the waterfall range performance stops improving for  $M \geq 4$ . Similar behavior has also been observed for other related turbo-type codes (such as CT codes [21]).

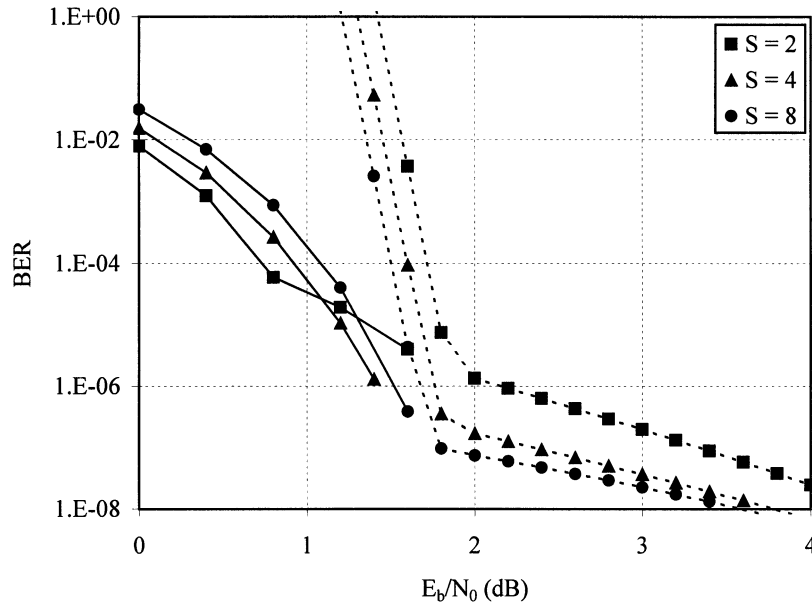


Fig. 9. Performance of turbo-Hadamard codes with  $r = 5$ ,  $N = 200$ ,  $M = 4$ , and different  $\hat{C}$  (with different numbers of states  $S$ ).  $G(x) = 1/(1+x)$ ,  $(1+x)/(1+x+x^2)$ , and  $(1+x^2+x^3)/(1+x+x^3)$  for  $S = 2, 4$ , and  $8$ , respectively. The solid lines represent the simulated results and the dashed lines represent the union bounds.

### B. The Impact of $\hat{C}$

Consider the minimum Hamming weight of a turbo-Hadamard code (Fig. 3). For the concatenated code, the minimum Hamming weight is associated with the following events.

- i) The information word contains an even number of ones, and the number of ones is less than or equal to  $r$ .
- ii) All of these information ones are located in the same row in every  $\mathbf{D}^{(m)}$ ,  $1 \leq m \leq M$ .

With i) and ii), the parity checks  $\mathbf{q}' = \mathbf{0}$  and so  $\mathbf{q} = \mathbf{0}$  for any linear  $\hat{C}$ . In this case, there is only one nonzero row in every component code (containing all the information ones). The joint probability of events i) and ii) is related to the interleaving method, and is independent of  $\hat{C}$ . Assuming random interleavers, the joint probability of events i) and ii) is not zero. Therefore, the minimum Hamming weight, and so the asymptotic performance, of a turbo-Hadamard code is independent of the choice of  $\hat{C}$ . This property is also similar to that of concatenated tree codes [21] where each trellis section also carries multiple information bits. For a standard turbo code [13], the situation is different since each trellis section carries only one information bit. Increasing the order of component codes can significantly reduce the error floor of a standard turbo code.

Employing more complicated  $\hat{C}$  may still bring about certain improvements in performance, since it reduces the multiplicities of codewords with small weights other than the minimum one. This can be seen from the examples provided in Fig. 9. The performance improvement is marginal for  $S > 8$ .

### C. The Impact of the Order of the Hadamard Code

Fig. 10 illustrates the impact of  $r$  on the performance of the turbo-Hadamard code. Increasing  $r$  has a beneficial effect since stronger component codes are used. However, it also has a negative effect in that the multiplicity of minimum-weight code-

words is increased, since the probability of all information ones located in the same row in every  $\mathbf{D}^{(m)}$  ( $1 \leq m \leq M$ ) is increased. We should also keep in mind that decoding costs increase exponentially with  $r$ .

### D. Convergence

Fig. 11 illustrates the convergence behavior of turbo-Hadamard codes with interleaver lengths  $N = 200$  and  $65534$ , respectively. It can be seen that the convergence speed decreases as the interleaver length increases. This may be due to the fact that the latter operates in noisier conditions.

### E. Performance Comparison With Other Low-Rate Codes

To our knowledge, the super-orthogonal turbo code [8] demonstrates the best performance of existing low-rate schemes. It carries only one information bit per trellis section and relies on the trellis code to provide coding gain. Hadamard codewords are only used to increase the distance between the branch outputs.

Fig. 12 contains a performance comparison between the proposed code and the super-orthogonal turbo code [8]. The generator polynomials of super-orthogonal turbo codes are  $1/(1+x^2+x^4)$  and  $1/(1+x^2+x^3+x^5+x^6)$  for  $S = 16$  and  $64$ , respectively. The generator polynomials of turbo-Hadamard codes are  $1/(1+x)$  for  $r = 7$ ,  $M = 3$ ,  $S = 2$ ,  $N = 65534$ , and  $(1+x)/(1+x+x^2)$  for  $r = 3$ ,  $M = 4$ ,  $S = 4$ ,  $N = 198$ .

It can be seen that for  $N = 65534$ , the rate- $1/52.9$  turbo-Hadamard code can achieve  $\text{BER} = 10^{-5}$  at  $E_b/N_0 \approx -1.2$  dB, only about 0.4 dB away from the ultimate Shannon limit. This is considerably better than that of the best super-orthogonal turbo code.

It can also be seen from Fig. 12 that, for  $N = 198$  and  $\text{BER} = 10^{-5}$ , the performance of the four-state turbo-Hadamard code with  $R \approx 1/7.7$  is very close to that of the 16-state super-or-

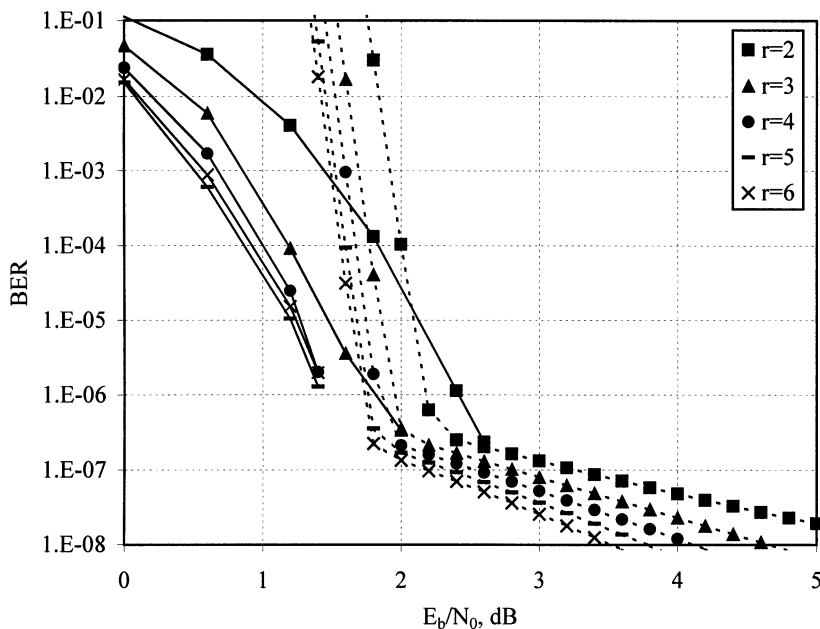


Fig. 10. Performance of turbo-Hadamard codes with  $M = 4, S = 4, G(x) = (1 + x)/(1 + x + x^2)$ , different values of  $r$ , and  $N \approx 200$ . (Note:  $N$  should be a multiple of  $r$ , so it is adjusted slightly for different  $r$ .) The solid lines represent the simulated results and the dashed lines represent the union bounds.

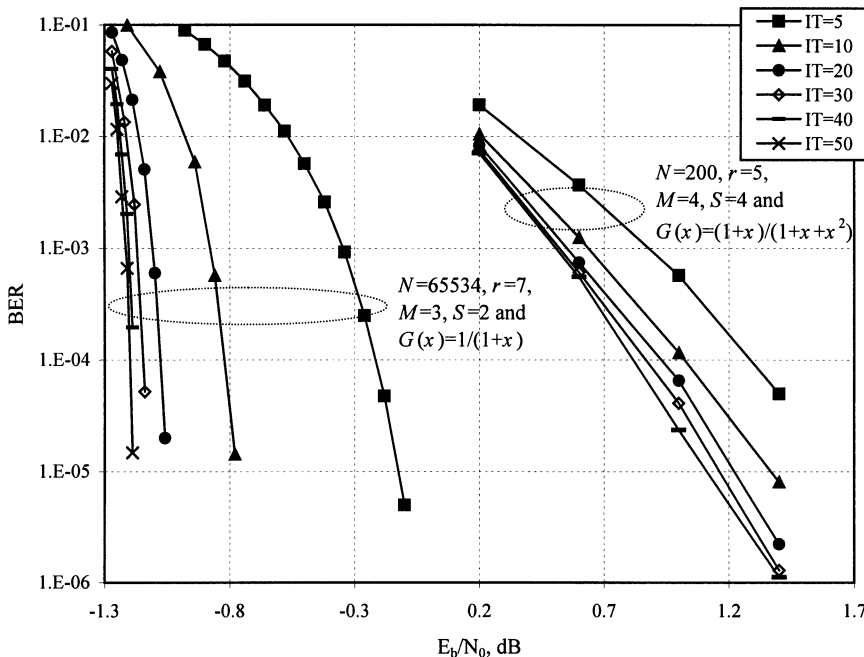


Fig. 11. Convergence property of turbo-Hadamard codes with interleaver lengths of 65534 and 200.

thogonal turbo code with  $R = 1/15$ . Table I compares the decoding costs of these two approaches. The complexity of the turbo-Hadamard code is (approximately) estimated according to Algorithm 2 (Appendix B). The complexity of the super-orthogonal turbo code is (approximately) estimated based on the BCJR algorithm for a 16-state convolutional code plus an FHT at the input stage (including normalization). The comparison takes into consideration that the turbo-Hadamard code involves four component codes and the super-orthogonal turbo code involves only two. Table I clearly indicates the advantages of the proposed code.

### V. DISCUSSION AND CONCLUSION

Several authors [8]–[10] have discussed the design of low-rate turbo-type codes. A common approach is to increase the length of the branch output while maintaining the basic structure of the turbo codes (i.e., two component codes and no parallel branches). In this case, while performance improves, the decoding cost increases rapidly when the number of states increases.

The proposed turbo-Hadamard code is characterized by trellis structures with a small number of states, and a large number of parallel branches. Although the existence of parallel branches is

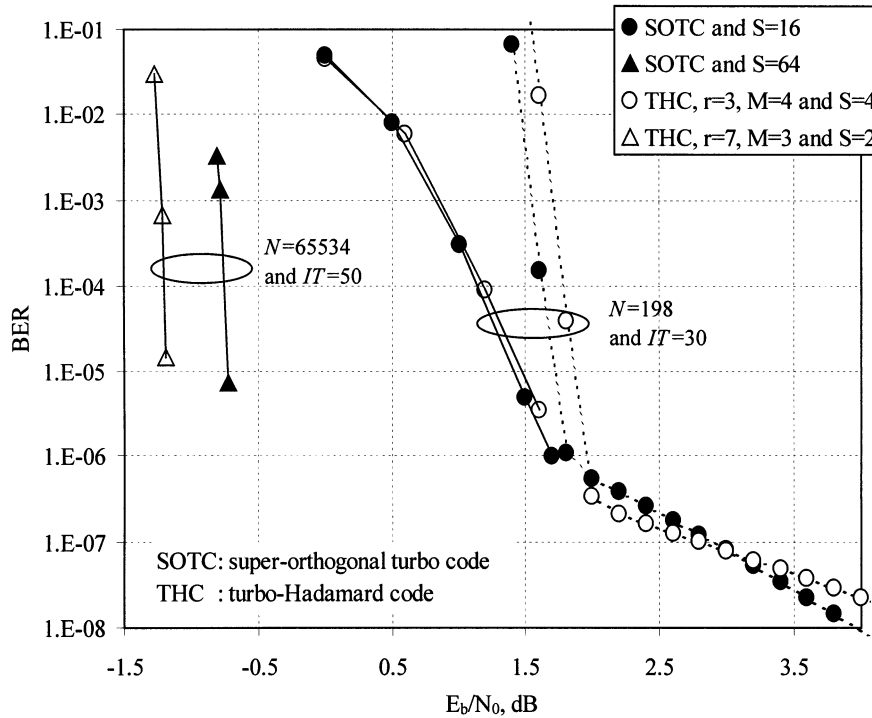


Fig. 12. Performance comparison between turbo-Hadamard codes and super-orthogonal turbo codes. The solid lines represent the simulated results and the dashed lines represent the union bounds. Due to the numerical difficulty, we do not have the BER bounds for  $N = 65534$  codes.

TABLE I  
COMPLEXITY COMPARISON OF  $N = 198$  CODES USED IN FIG. 12. (UNIT: OPERATION NUMBERS PER INFORMATION BIT PER ITERATION)

	Rate 1/7.7 turbo-Hadamard code ( $S=4, r=3$ and $M=4$ )	Rate 1/15 super-orthogonal turbo code ( $S=16$ and $M=2$ )
Multiplications/divisions	68	320
Exponential/logarithm functions	26	34
Additions	147	280

usually a disadvantage for ordinary trellis codes, the situation is different in the proposed scheme for the following reasons.

- The use of Hadamard codewords as branch labels guarantees good Hamming distances between parallel branches.
- Every trellis section carries multiple information bits, which increases energy efficiency as well as reduces normalized decoding costs.
- When very simple trellises are used, the main decoding complexity resides in the APP decoding of the Hadamard code, which can be accomplished by a very efficient APP-FHT.

The applications of the proposed codes can be found in [25], [26]. In [25], a multiple-access scheme based on interleaved low-rate codes is investigated. In [26], a space-time coding scheme based on low-rate codes is studied to exploit the expanded signaling dimension provided by multiple transmit antennas. Performance close to the theoretical limits is observed in both [25]

and [26]. These works clearly demonstrate the potentials of the proposed codes in future wireless communication systems.

#### APPENDIX A REDUCED FAST HADAMARD TRANSFORMS

In a systematic encoding, the index set

$$J_r = \{0, 1, 2, 4, \dots, 2^{r-1}\}$$

can be chosen as information positions for a length- $2^r$  Hadamard code. We consider two special FHTs. The first one is referred to as the reduced input FHT. In this case, we assume that all the input entries, except those with indexes in  $J_r$  are zeros. It can be verified that in a reduced input FHT, additions are only required for nodes with indexes  $\{0, 1, \dots, 2^{r'} - 1\}$ , at level  $r'$ ,  $r' = 1, 2, \dots, r$ , see Fig. 13(a). The number of required additions involved is

$$2^1 + 2^2 + \dots + 2^r = 2(2^r - 1) \approx 2 \times 2^r.$$

The second special case is referred to as the reduced output FHT. In this case, we assume that the outputs are only required at

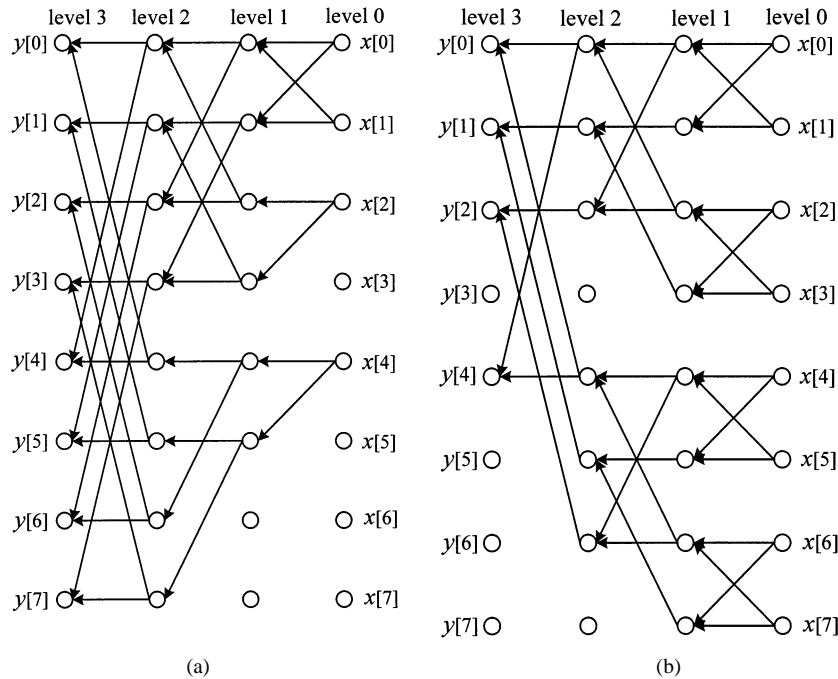


Fig. 13. Flow graphs of reduced FHT, where the information positions are  $\{0, 1, 2, \text{ and } 4\}$ . (a) Length-8 reduced input FHT, where branches delivering zero values are deleted. (b) Length-8 reduced output FHT, where branches not leading to the required outputs are deleted.

the information positions. Then, additions are only required for nodes with indexes  $\{k2^{r'} + j | j \in J_{r'}, k = 0, 1, \dots, 2^{r-r'} - 1\}$  at level  $r'$ , see Fig. 13(b). The total number of additions involved is

$$2 \times 2^{r-1} + 3 \times 2^{r-2} + 4 \times 2^{r-3} + \dots + (r+1) \times 2^0 = 3(2^r - 1) - r \approx 3 \times 2^r.$$

## APPENDIX B

### APP DECODING OF CONVOLUTIONAL-HADAMARD CODES

This appendix addresses the decoding of convolutional-Hadamard codes. Let  $\mathbf{c}$  be the transmitted codeword and  $\mathbf{x}$  be its noisy observation. Consider the  $k$ th section of a trellis, Fig. 4. Let  $\mathbf{c}_k$  be the corresponding  $k$ th section of  $\mathbf{c}$ , and  $\mathbf{x}_k$  be its noisy observation. Define

$$\gamma_k(\pm \mathbf{h}^j) \equiv \Pr(\mathbf{x}_k | \mathbf{c}_k = \pm \mathbf{h}^j) \quad (23)$$

which is the probability that  $\mathbf{x}_k$  is the observed signal, conditioned on  $\mathbf{c}_k = \pm \mathbf{h}^j$ . For an AWGN channel, it is easily seen that

$$\gamma_k(\pm \mathbf{h}^j) = A \exp\left(\frac{1}{2} \langle \pm \mathbf{h}^j, \tilde{\mathbf{L}}_k \rangle\right) \quad (24)$$

where  $\tilde{\mathbf{L}}_k = 2\mathbf{x}_k/\sigma^2$  (see (5b)) and  $A$  is a constant that will be canceled out (see (25d)). We will use  $L_k[i]$  to represent the LLR of the  $i$ th element of  $\mathbf{c}_k$ . The BCJR algorithm consists of the following steps [19]:

#### Preparation :

$$B(s_k, s_{k+1}) = \sum_{\text{all parallel branches between } s_k \text{ and } s_{k+1}} \gamma_k(\pm \mathbf{h}^j). \quad (25a)$$

#### Forward recursion :

$$\alpha(s_k) = \sum_{\forall s_{k-1}} B(s_{k-1}, s_k) \alpha(s_{k-1}). \quad (25b)$$

#### Backward recursion :

$$\beta(s_k) = \sum_{\forall s_{k+1}} B(s_k, s_{k+1}) \beta(s_{k+1}). \quad (25c)$$

#### Output stage:

$$L_k[i] = \log \frac{\sum_{H[i,j]=\pm 1} \gamma_k(\pm \mathbf{h}^j) \alpha(s_k) \beta(s_{k+1})}{\sum_{H[i,j]=\mp 1} \gamma_k(\pm \mathbf{h}^j) \alpha(s_k) \beta(s_{k+1})}. \quad (25d)$$

Compared with (5a), (25d) can be evaluated by substituting  $\gamma_k(\pm \mathbf{h}^j) \alpha(s_k) \beta(s_{k+1})$  into the position of  $\gamma(\pm \mathbf{h}^j)$  in (5). The APP-FHT algorithm developed in Section II can thus be used to evaluate (25d). Equation (25d) leads to the following decoding algorithm.

#### Algorithm 2. Decoding the Convolutional-Hadamard Code

Step 1) Perform a reduced input FHT to obtain  $\gamma_k(\mathbf{h}^j)$  (see Note a). (Cost: about  $2 \times (2^r - 1)$  additions and  $2 \times 2^r$  exponential functions per trellis section.)

Step 2) Evaluate (25a)–(25c) and generate

$$\gamma_k(\pm \mathbf{h}^j) \alpha(s_k) \beta(s_{k+1}), \quad \text{for all } j \text{ and } k$$

(Cost: about  $8S + 2 \times 2^r$  multiplication and  $4S + 4 \times (2^{r-1} - 1)$  additions per trellis section including normalization, where  $S$  is the number of states).

Step 3) For every section, perform a reduced output APP-FHT and evaluate (25d). (Cost: about  $6 \times (2^r - 1) - 2r$  additions,  $r$  divisions, and  $r$  logarithms per trellis section).

Notes: There are some additional operations as explained in the following.

a) In evaluating  $\gamma_k(\pm \mathbf{h}^j) = \exp(\frac{1}{2} \langle \pm \mathbf{h}^j, \tilde{\mathbf{L}}_k \rangle)$  in (24), we decompose  $\tilde{\mathbf{L}}_k$  into  $\tilde{\mathbf{L}}_k = \tilde{\mathbf{I}}_k + \tilde{\mathbf{P}}_k$  where  $\tilde{\mathbf{I}}_k$  (resp.,  $\tilde{\mathbf{P}}_k$ ) is

the vector obtained by replacing the parity (resp., information) entries in  $\tilde{\mathbf{L}}_k$  by zeros. In a turbo-type decoder,  $\tilde{\mathbf{P}}_k$  is not updated [13], [21] in all iterations. Thus we can use a reduced input FHT to update  $\mathbf{H}\tilde{\mathbf{I}}_k$  and then add it to a stored  $\mathbf{H}\tilde{\mathbf{P}}_k$  to obtain  $\mathbf{H}\tilde{\mathbf{L}}_k$ . The cost can be reduced in this way.

- b) Again, in evaluating (24), normalization may be necessary to avoid overflow.

#### ACKNOWLEDGMENT

The authors wish to thank the Associate Editor and the anonymous reviewers for their careful reading and constructive comments to improve the paper. They would also like to thank S. Chan for his valuable help.

#### REFERENCES

- [1] K. S. Gilhousen, I. M. Jacobs, R. Padovani, A. J. Viterbi, L. A. Weaver, and C. E. Wheatley Jr., "On the capacity of a cellular CDMA system," *IEEE Trans. Veh. Technol.*, vol. 40, no. 2, pp. 303–312, May 1991.
- [2] A. J. Viterbi, "The orthogonal-random waveform dichotomy for digital mobile personal communication," *IEEE Personal Commun.*, vol. 1, pp. 18–24, Mar. 1994.
- [3] —, "Very low rate convolutional codes for maximum theoretical performance of spread-spectrum multiple-access channels," *IEEE J. Selected Areas Commun.*, vol. 8, pp. 641–649, May 1990.
- [4] R. Herzog, J. Hagenauer, and A. Schmidbauer, "Soft-in/soft-out hadamard despreader for iterative decoding in the IS-95(A) system," *Proc. IEEE Vehicular Technology Conf. (VTC'97)*, vol. 2, pp. 1219–1222, 1997.
- [5] F. M. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1997, vol. 1 and 2.
- [6] R. K. R. Yarlagadda and J. E. Hershey, *Hadamard Matrix Analysis and Synthesis: With Applications to Communications and Signal/Image Processing*. Boston, MA: Kluwer, 1997.
- [7] A. J. Viterbi, "Orthogonal tree codes for communication in the presence of white Gaussian noise," *IEEE Trans. Commun. Technol.*, vol. COM-15, pp. 238–242, Apr. 1967.
- [8] P. Komulainen and K. Pehkonen, "Performance evaluation of super-orthogonal turbo codes in AWGN and flat Rayleigh fading channels," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 196–205, Feb. 1998.
- [9] M. Bingeman and A. K. Khandani, "Symbol based turbo codes," *IEEE Commun. Lett.*, vol. 3, pp. 285–287, Oct. 1999.
- [10] P. Sauve and F. R. Kschischang, "Decoding turbo codes by multi bit probability propagation," in *Proc. 19th Biennial Symp. Communications*, Kingston, ON, Canada, June 1998, pp. 89–93.
- [11] D. Divsalar, H. Jin, and R. McEliece, "Coding theorems for 'turbo-like' codes," in *Proc. 1998 Allerton Conf.*, 1998, pp. 201–210.
- [12] L. Ping and S. Chan, "Iterative decoding of concatenated hadamard codes," *Proc. IEEE Int. Communications Conf. (ICC'98)*, vol. 1, pp. 136–140, 1998.
- [13] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes," *Proc. IEEE Int. Communications Conf. (ICC'93)*, vol. 2, pp. 1064–1070, 1993.
- [14] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 21–28, Jan. 1962.
- [15] D. J. C. McKay and R. M. Neal, "Near shannon limit performance of low-density parity check codes," *Electron. Lett.*, vol. 33, no. 6, pp. 457–458, Mar. 1997.
- [16] W. K. Leung, K. S. Ho, and L. Ping, "Approaching low rate shannon limit using concatenated hadamard codes," *Electron. Lett.*, vol. 36, no. 1, pp. 45–46, Jan. 2000.
- [17] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing Principles, Algorithms, and Applications*. New York: Macmillan, 1992.
- [18] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 429–445, Mar. 1996.
- [19] J. Hagenauer, "The turbo principle: Tutorial introduction and state of the art," in *Proc. Int. Symp. Turbo Codes and Related Topics*, Brest, France, Sept. 1997, pp. 1–11.
- [20] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409–428, Mar. 1996.
- [21] L. Ping and K. Y. Wu, "Concatenated tree codes: A low complexity, high performance approach," *IEEE Trans. Inform. Theory*, vol. 47, pp. 791–799, Feb. 2001.
- [22] L. Ping, "Simple method for generating distance spectrum and multiplicities of convolutional codes," *Proc. Inst. Elec. Eng.—Commun.*, vol. 143, no. 5, pp. 247–249, Oct. 1996.
- [23] M. L. Cedervall and R. Johannesson, "A fast algorithm for computing distance spectrum of convolutional codes," *IEEE Trans. Inform. Theory*, vol. 35, pp. 1146–1159, Nov. 1989.
- [24] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Analysis, design, and iterative decoding of double serially concatenated codes with interleavers," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 231–244, Feb. 1998.
- [25] L. Ping, L. Liu, and W. K. Leung, "A simple approach to near-optimal multiuser detection: Interleaver-division multiple-access," in *Proc. IEEE Wireless Communications and Networking Conf. (WCNC'2003)*, New Orleans, LA, Mar. 2003, pp. 391–396.
- [26] W. K. Leung, K. Y. Wu, and L. Ping, "Interleave-division-multiplexing space-time codes," in *Proc. IEEE Vehicular Technology (VTC'03)*, Jeju, Korea, Apr. 2003, pp. 1094–1098.