Differentiated Assignment of Extrinsic Information in Iterated Decoding of Fixed Weight Codewords

Wonsun Bong*, Yong Cheol Kim*
*Dept. of Electrical and Computer Eng., University of Seoul, Korea
gaam@uos.ac.kr, yckim@uos.ac.kr

***Abstract*—Constant amplitude multi-code (CAMC) CDMA has the same structure as a recursively generated single parity check product code. A top-level codeword of CAMC is recursively constructed from lower-level codewords. In the iterative decoding of CAMC, log likelihood ratio (LLR), *a priori* information and *extrinsic* information (EI) of a codeword is a weighted sum of LLR values of associated codewords from which it is despread or into which it is spread. In this paper, we show that differentiated assignment of EI in the computation of LLR can improve the performance of bit error correction. The weights of CAMC codewords are fixed at two fixed values. We let EI converge fast to saturation value when a codeword has the correct weight. The proposed method achieved performance improvement of $0.1 \sim 0.3$ dB in E_b/N_0 over the regular iterated decoding of CAMC. When compared with despreading ON/OFF control, a gain of about 0.1 dB is achieved, which is meaningful near the Shannon capacity limit.**

***Keywords*—Constant Amplitude Multi Code, Code Weight, Extrinsic Information, Iterated Decoding, Single Parity Check Product Code**

I. Introduction

PRODUCT codes were first introduced by Elias in 1954 [1]. The concept of product codes is that powerful long block codes can be constructed by concatenating two or more shorter constituent codes. Single parity check product code (SPCPC) is a product code in a simple structure, where a parity bit is appended to a sequence of information bits [2].

A codeword of 3-D (dimensional) SPCPC is shown in Fig. 1, which is composed of the data block, the parity checks along all three directions and parity on parity check bits. Multi-dimensional SPCPC are constructed in a similar way. In the encoder, a parity bit is appended to each of $(n-1)$-bit-long sequences along all the dimensions of a Q-D hypercube consisting of $(n-1)^Q$ information bits. The encoded output of n^Q bits is a Q-D product code with a code rate of $(1-1/n)^Q$.

Manuscript received date is November 9, 2015. This research was supported by Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (grant number NRF-2013R1A1A2012745). This paper is a follow-up of the invited journal to the accepted conference paper of the 16th International Conference on Advanced Communication Technology.

Wonsun Bong is with the department of Electrical and Computer Engineering, University of Seoul, Seoul 02504, Korea (email: gaam@uos.ac.kr). Yong Cheol Kim is with the department of Electrical and Computer Engineering, University of Seoul, Seoul 02504, Korea (corresponding author; +82-2-6490-2331; e-mail: yckim@uos.ac.kr).

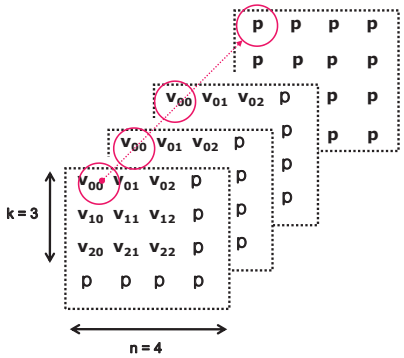

Fig. 1: 3-D SPCPC: Parities are in all three directions.

Kim presented a recursive SPCPC, where a codeword is recursively constructed by pseudo-Hadamard spreading of three lower-dimensional codewords concatenated with their parity bits [3]. Recursive SPCPC was originally developed as a constant amplitude multi-code (CAMC) CDMA.

Multi-code CDMA is a technique of providing versatile data rates by assigning multiple channels to a single user [4] [5] [6]. Multi-code signal has a large amplitude variation since it is a sum of random binary signals from several channels. Large variation signal requires a highly linear power amplifier, which consumes a large power. CAMC was developed to perfectly remove the amplitude fluctuation of multi-code signal.

The encoder of CAMC accommodates $M(=3^Q)$ information bits, $\mathbf{b}^{\mathrm{M}} = [b_0, b_1, ..., b_{M-1}]$, and generates a $N(=4^Q)$-bit-long string, $\mathbf{v}^{\mathrm{N}} = [v_0, v_1, ..., v_{N-1}]$, of constant amplitude by pseudo-Hadamard spreading in a recursive manner. The code rate of CAMC is $R = (3/4)^Q$, which is equivalent to that of SPCPC with $n = 4$. Throughout this paper, recursive SPCPC with $n = 4$ is interchangeably referred to as CAMC.

Previous works show that CAMC outperforms conventional SPCPC. CAMC has some advantageous features as follows [7] [8]. First, CAMC benefits from the despreading process which is performed after the iterative decoding. Second, CAMC does not have any low weight codewords which usually degrade the performance at low SNR. The weights of codewords are evenly distributed at two fixed values. Third, the property of fixed weight can guide the computation of extrinsic information (EI) which is the key element in the iterative decoding.

Analysis on the first and the second features were reported in [8] [9]. In this paper, we focus on the third feature. We show that, with differentiated assignment of EI based on the integrity of code weights, we get performance improvement of 0.1∼0.3 dB when compared to previous works.

This paper is organized as follows: In Section II, encoding and decoding of CAMC are briefly presented. In Section III, iterated decoding of conventional SPCPC is described. In Section IV, iterative decoding and despreading of CAMC are presented. In Section V, ON/OFF control of despreading after iterative decoding is presented. In Section VI, the proposed differentiated assignment of EI, based on the fixed weight property of CAMC, is presented. In Section VII, computer simulation results on performance improvement are presented. Finally, a conclusion is drawn in Section VIII.

II. Constant Amplitude Multi Code

In this Section, the generation of CAMC signal vectors is briefly described [3]. Throughout this paper, the polarity of a bit is bipolar, either (+1) or (-1). An encoded CAMC vector at J-level has a length of $L(=4^J)$ bits. Then, for every three J-level vectors, a bit-by-bit J-level parity vector is generated. Spreading the concatenation of three J-level vectors and their parity vector generates a $(J+1)$-level vector with a length of $4L$ bits. Inversely, despreading a J-level vector results in three $(J-1)$-level signal vectors and their bit-by-bit parity vector, each with a length of $L/4$ bits.

In the following notations, a superscript represents the size of the vector, except when the subscript is of the form $i/4$. Integer subscripts of $\{0,1,2,3\}$, if any, stand for the distinct number of vectors. Either $\mathbf{v}_i^{\mathrm{L}}$ or $\mathbf{v}^{\mathrm{L}}$ is a L-bit-long CAMC vector, There is no meaningful difference between them. Subscripts of $\{0/4, 1/4, 2/4, 3/4\}$ represent the index of the four quadrants of a regular CAMC vector. For example, $\mathbf{v}_{i/4}^{\mathrm{L}},\ i \in \{0,1,2,3\}$, is not a regular CAMC vector by itself, but just a quadrant of a regular L-bit-long CAMC vector, $\mathbf{v}^{\mathrm{L}}$, as shown in (1). The vertical bar represents concatenation.

$$\mathbf{v}^{\mathrm{L}} = [\mathbf{v}_{0/4}^{\mathrm{L}} \mid \mathbf{v}_{1/4}^{\mathrm{L}} \mid \mathbf{v}_{2/4}^{\mathrm{L}} \mid \mathbf{v}_{3/4}^{\mathrm{L}}] \tag{1}$$

Fig. 2 and Fig. 3 show the generation of parity bits in a CAMC vector [3]. An input of M information bits is divided into $M/3$ strings of three bits each. Each 3-bit-long string is encoded at the basic-level encoder $\mathbf{Q^4}$. For input $[b_0, b_1, b_2]$, a parity bit $p_3 = -b_0 \cdot b_1 \cdot b_2$ is appended to them. Then, four bits of unit amplitude, $\mathbf{v}_0^4 = [v_0, v_1, v_2, v_3]$ are generated from spreading by 4×4 Hadamard matrix.

$$\mathbf{v}_0^4 = \frac{1}{2} \cdot [b_0 \quad b_1 \quad b_2 \quad p_3] \cdot \mathbf{H}^4 \tag{2}$$

$$\mathbf{H}^4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{3}$$

In a similar way, $[b_3, b_4, b_5]$ and $[b_6, b_7, b_8]$ are encoded into $\mathbf{v}_1^4$ and $\mathbf{v}_2^4$ of constant amplitude. A 4-bit-long parity $\mathbf{p}^4$ is

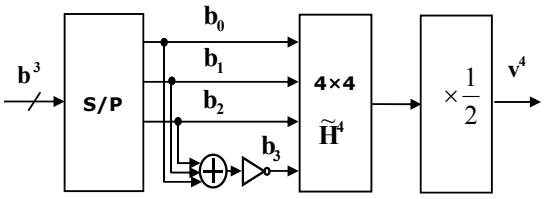

Fig. 2: Generation of 4-bit CAMC vector

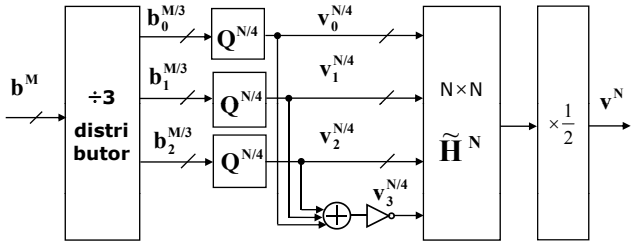

Fig. 3: Generation of N-bit CAMC vector

generated from bit-by-bit product (denoted as $.*$) of $\mathbf{v_0^4}$, $\mathbf{v_1^4}$ and $\mathbf{v_2^4}$.

$$\mathbf{p}^4 = -\mathbf{v_0^4}. * \mathbf{v_1^4}. * \mathbf{v_2^4} \tag{4}$$

For the spreading of concatenation of three CAMC vectors and one parity vector, we use a pseudo-Hadamard matrix, $\tilde{\mathbf{H}}^{\mathrm{N}}$. This is in the form of Hadamard matrix with "1" element in (3) replaced by an identity matrix $\mathbf{I}^{\mathrm{N}/4}$ of size $N/4 \times N/4$. As a special case, $\tilde{\mathbf{H}}^4$ is identical to $\mathbf{H}^4$.

$$\tilde{\mathbf{H}}^{\mathrm{N}} = \begin{bmatrix} \mathbf{I}^{\mathrm{N}/4} & \mathbf{I}^{\mathrm{N}/4} & \mathbf{I}^{\mathrm{N}/4} & \mathbf{I}^{\mathrm{N}/4} \\ \mathbf{I}^{\mathrm{N}/4} & -\mathbf{I}^{\mathrm{N}/4} & \mathbf{I}^{\mathrm{N}/4} & -\mathbf{I}^{\mathrm{N}/4} \\ \mathbf{I}^{\mathrm{N}/4} & \mathbf{I}^{\mathrm{N}/4} & -\mathbf{I}^{\mathrm{N}/4} & -\mathbf{I}^{\mathrm{N}/4} \\ \mathbf{I}^{\mathrm{N}/4} & -\mathbf{I}^{\mathrm{N}/4} & -\mathbf{I}^{\mathrm{N}/4} & \mathbf{I}^{\mathrm{N}/4} \end{bmatrix} \tag{5}$$

Both $\tilde{\mathbf{H}}^{\mathrm{N}}$ and $\mathbf{H}^{\mathrm{N}}$ are orthogonal, subject to a scaling factor.

$$\tilde{\mathbf{H}}^{\mathrm{N}} \cdot (\tilde{\mathbf{H}}^{\mathrm{N}})^t = \tilde{\mathbf{H}}^{\mathrm{N}} \cdot \tilde{\mathbf{H}}^{\mathrm{N}} = 4 \cdot \mathbf{I}^{\mathrm{N}} \tag{6}$$

$$\mathbf{H}^{\mathrm{N}} \cdot (\mathbf{H}^{\mathrm{N}})^t = \mathbf{H}^{\mathrm{N}} \cdot \mathbf{H}^{\mathrm{N}} = N \cdot \mathbf{I}^{\mathrm{N}} \tag{7}$$

The 16-bit-long concatenation of $\mathbf{v_0^4}$, $\mathbf{v_1^4}$, $\mathbf{v_2^4}$ and $\mathbf{p}^4$ is spread into $\mathbf{v}^{16}$ of unit amplitude.

$$\mathbf{v}^{16} = \frac{1}{2} \cdot [\mathbf{v}_0^4 \mid \mathbf{v}_1^4 \mid \mathbf{v}_2^4 \mid \mathbf{p}^4] \cdot \tilde{\mathbf{H}}^{16} \tag{8}$$

Continuing this way, the output of three $\mathbf{Q}^{\mathrm{N}/4}$ encoders ($3N/4$ bits, in all) and their bit-by-bit parity vector ($N/4$ bits) are generated.

$$\mathbf{p}^{\mathrm{N}/4} = -\mathbf{v}_0^{\mathrm{N}/4}. * \mathbf{v}_1^{\mathrm{N}/4}. * \mathbf{v}_2^{\mathrm{N}/4} \tag{9}$$

Finally, thee N-bit-long concatenation of $\mathbf{v_0}^{\mathrm{N}/4}$, $\mathbf{v_1}^{\mathrm{N}/4}$, $\mathbf{v_2}^{\mathrm{N}/4}$ and $\mathbf{p}^{\mathrm{N}/4}$ is spread by $\tilde{\mathbf{H}}^{\mathrm{N}}$, into $\mathbf{v}^{\mathrm{N}} = [v_0, v_1, ..., v_{N-1}]$ of unit amplitude.

$$\mathbf{v}^{\mathrm{N}} = \frac{1}{2} \cdot [\mathbf{v}_0^{\mathrm{N}/4} \mid \mathbf{v}_1^{\mathrm{N}/4} \mid \mathbf{v}_2^{\mathrm{N}/4} \mid \mathbf{p}^{\mathrm{N}/4}] \cdot \tilde{\mathbf{H}}^{\mathrm{N}} \tag{10}$$

While a top-level codeword $\mathbf{v}^{\mathrm{N}}$ is encoded from recursive spreading by pseudo-Hadamard $\tilde{\mathbf{H}}^{\mathrm{N}}$, decoding of $\mathbf{v}^{\mathrm{N}}$ is obtained from one-time despreading by a regular $\mathbf{H}^{\mathrm{N}}$.

$$\begin{aligned} & \mathbf{v}^{\mathrm{N}} \cdot \mathbf{H}^{\mathrm{N}} \qquad (11) \\ &= 2[\mathbf{v}_0^{\mathrm{N}/4}\mathbf{H}^{\mathrm{N}/4} \mid \mathbf{v}_1^{\mathrm{N}/4}\mathbf{H}^{\mathrm{N}/4} \mid \mathbf{v}_2^{\mathrm{N}/4}\mathbf{H}^{\mathrm{N}/4} \mid \mathbf{v}_3^{\mathrm{N}/4}\mathbf{H}^{\mathrm{N}/4}] \\ &= 2^{(\log_4 N-1)}[\mathbf{v}_0^4\mathbf{H}^4 \mid \mathbf{v}_1^4\mathbf{H}^4 \mid \mathbf{v}_2^4\mathbf{H}^4 \mid \mathbf{v}_3^4\mathbf{H}^4 \mid \ldots] \\ &= 2^{\log_4 N}[d_0, d_1, d_2, d_3, d_4, \ldots] \end{aligned}$$

The values of correlation vector, $\mathbf{D}$, of noise-free $\mathbf{v}^{\mathrm{N}}$ are $d_i = \pm 1$. But, $\mathbf{D}$ obtained from a noisy received signal at the receiver takes on non-binary values. In this case, we hard-limit $\mathbf{D}$ into binary values. The correlation vector $[d_0, d_1, d_2, ...]$ is in the form of a hypercube consisting of information bits and parity bits, just like a conventional SPCPC as shown in Fig. 1. The information bits are extracted from the corresponding positions. Bits $\{d_3, d_7, d_{11}, d_{12}, d_{13}, d_{14}, d_{15}, d_{19}, ...\}$ are in the positions which correspond to the parity positions in the hypercube. Removing such bits, we get the information-only bits $[b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, ...]$.

III. Iterative Decoding of a Product Code

The decoding process of a product code is similar to solving a crossword puzzle. In the product array, one symbol is associated with two values through *diversity* effect: One is the very value of the received symbol itself and the other is the *extrinsic* value which can be inferred from the other symbols. When bit errors through a channel , these two values may be different. This discrepancy can be relaxed in the iterative decoding process where the range of the possible values of a target symbol is adjusted through exchange of EI among neighboring symbols.

Being a product code, SPCPC has powerful error correcting capability, SPCPC has the same basic features of turbo codes: interleaving, iteration and soft-output decoding [10]. Hagenauer developed a soft input, soft output-based decoding algorithm for a multi-dimensional product code [11].

Rankin [2] extended the works of [11] and proposed an iterative decoding algorithm for SPCPC. It is an iterative algorithm which refines the log likelihood ratio (LLR) for each bit by iteratively exchanging information in a relaxational scheme.

In SPCPC, a parity bit is appended along all the dimensions of a Q-D interleaved hypercube. The LLR, $L_q(X_k)$, for the k-th bit in the q-th dimension is iteratively refined by exchanging the extrinsic information between dimensions.

LLR consists of three terms: the channel reliability which is proportional to the signal strength, the *a priori* information (API, $A_q(X_k)$) and the extrinsic information (EI, $E_q(X_k)$). $\mathbf{X} = [X_0, X_1, ..., X_{N-1}]$ is the input vector and $\mathbf{Y} = [Y_0, Y_1, ..., Y_{N-1}]$ is the received signal vector through a binary-input AWGN channel.

$$L_q(X_k) = \log \frac{\Pr(X_k = +1 \mid \mathbf{Y})}{\Pr(X_k = -1 \mid \mathbf{Y})} = \frac{2}{\sigma^2} Y_k + E_q(X_k) + A_q(X_k) \tag{12}$$

$$E_q(X_k) = 2 \tanh^{-1} \left[\prod_{j=0, j \neq k}^{N-1} \tanh\left(\frac{A_q(X_j) + \frac{2}{\sigma^2} Y_j}{2} \right) \right] \tag{13}$$

$$A_q(X_k) = \sum_{i=1, i \neq q}^{Q} E_i(X_k) \tag{14}$$

IV. Decoding of CAMC

CAMC is generated by direct sequence spreading of lower dimensional codewords concatenated with their bit-by-bit parity vector. The internal structure of CAMC has both aspects of a product code and a spread spectrum signal.

The decoding process of CAMC consists of two stages: the iterative decoding for SPCPC and the despreading for spread spectrum signal. First, the noisy received bits, ***Y***, are iteratively decoded into ***X***, which consists of information bits and parity bits, as encoded in the recursive encoder. ***X*** is despread into ***D***, which consists only of information bits. Fig. 4 illustrates the flow of data, the recursive spreading, transmission through a noisy channel, iterated decoding and despreading.

Since CAMC belongs to SPCPC, a similar decoding algorithm could be used for CAMC. The decoding algorithm for conventional SPCPC described in Section III, however, cannot be directly applied to CAMC. Unlike SPCPC, there are no *raw* parity bits in a CAMC vector since parity bits are mixed with information bits through the spreading process.

Kim developed a decoding algorithm which separates parity bits and then iteratively refines the LLR of the bits [7]. Parity bits required for the iterative decoding are recursively extracted by decomposing higher level CAMC vectors into lower level vectors. One J-level vector, $\mathbf{v}^{\mathrm{L}}$, is despread by $\tilde{\mathbf{H}}^{\mathrm{L}}$ into four $(J-1)$-level vectors, $(\mathbf{v}_0^{\mathrm{L}/4}, \mathbf{v}_1^{\mathrm{L}/4}, \mathbf{v}_2^{\mathrm{L}/4}, \mathbf{p}^{\mathrm{L}/4})$.

$$\begin{aligned} & \frac{1}{2} \cdot \mathbf{v}^{\mathrm{L}} \cdot \tilde{\mathbf{H}}^{\mathrm{L}} \qquad (15) \\ &= \frac{1}{4} \cdot [\mathbf{v}_0^{\mathrm{L}/4} \mid \mathbf{v}_1^{\mathrm{L}/4} \mid \mathbf{v}_2^{\mathrm{L}/4} \mid \mathbf{p}^{\mathrm{L}/4}] \cdot \tilde{\mathbf{H}}^{\mathrm{L}} \cdot \tilde{\mathbf{H}}^{\mathrm{L}} \\ &= [\mathbf{v}_0^{\mathrm{L}/4} \mid \mathbf{v}_1^{\mathrm{L}/4} \mid \mathbf{v}_2^{\mathrm{L}/4} \mid \mathbf{p}^{\mathrm{L}/4}] \end{aligned}$$

The fourth vector, $\mathbf{p}^{\mathrm{L}/4}$, is the bit-by-bit parity vector of the three preceding CAMC vectors. Each of the three vectors, $\{\mathbf{v}_0^{\mathrm{L}/4}, \mathbf{v}_1^{\mathrm{L}/4}, \mathbf{v}_2^{\mathrm{L}/4}\}$, can be despread into four $(J-2)$-level CAMC vectors, $[\mathbf{v}_0^{\mathrm{L}/16} \mid \mathbf{v}_1^{\mathrm{L}/16} \mid \mathbf{v}_2^{\mathrm{L}/16} \mid \mathbf{p}^{\mathrm{L}/16}]$.

The parity relation among four J-level vectors holds only in the context of the J-th dimension. Likewise, EI and API associated with J-level CAMC vectors are valid only in the J-th dimension. For other dimensions, we need to spread or despread CAMC vectors as needed. Hence, when EI is exchanged between dimensions, it needs to be spread or

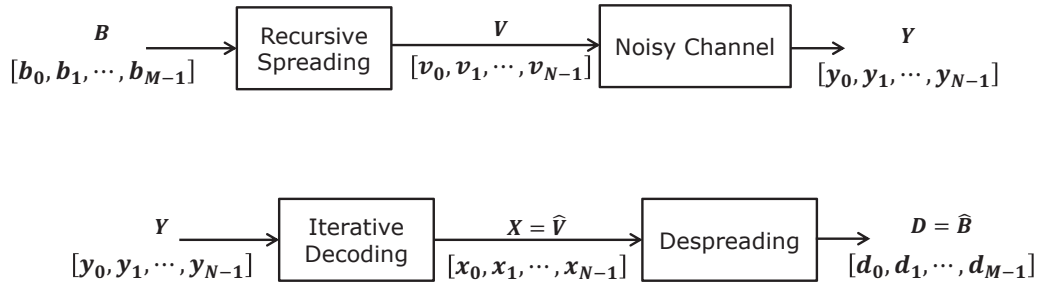

Fig. 4: Received bits ($\boldsymbol{Y}$) are iteratively decoded ($\boldsymbol{X}$) and then despread into information bits ($\boldsymbol{D}$).

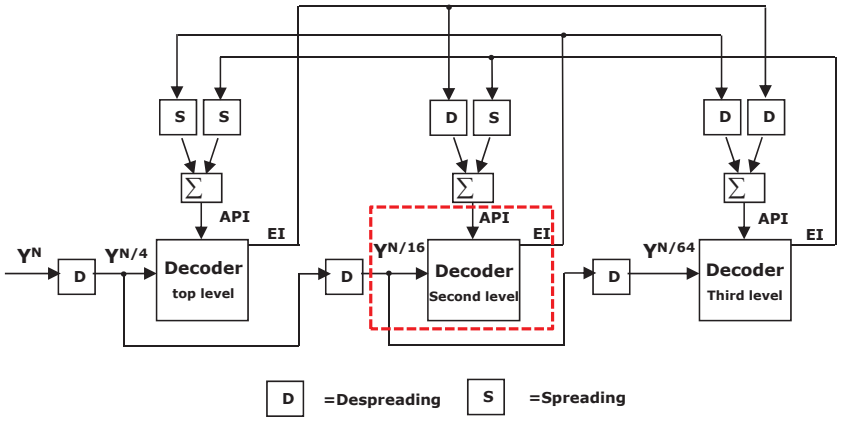

Fig. 5: EI is despread/spread between dimensions.

despread to fit into the structure of three CAMC vectors plus one parity vector in the corresponding dimensions.

Fig. 5 illustrates the block diagram for the decoding of CAMC for $N = 64$. The blocks S(.) and D(.) stand for the spreading process and the despreading process, respectively. Expressions for LLR ($L_q(X_k^q)$), EI ($E_q(X_k^q)$) and API ($A_q(X_k^q)$) for CAMC are shown in (16) through (18).

X_k^q is the k-th bit of the vector reconfigured into the q-th dimension. $[X_0^q, X_1^q, \ldots, X_{L-1}^q]$ and $[Y_0^q, Y_1^q, \ldots, Y_{L-1}^q]$, $(L = 4^q)$, stand for an input vector and the received vector reconfigured into the q-th dimension.

$$L_q(X_k^q) = \frac{2}{\sigma^2} Y_k^q + E_q(X_k^q) + A_q(X_k^q) \tag{16}$$

$$E_q(X_k^q) = 2\tanh^{-1}\left[\prod_{j=0, j\neq k}^{N-1} \tanh\left(\frac{A_q(X_j^q) + \frac{2}{\sigma^2} Y_j^q}{2}\right)\right] \tag{17}$$

$$A_q(X_k^q) = \sum_{i=1}^{q-1} S(E_i(X_k^q)) + \sum_{i=q+1}^{Q} D(E_i(X_k^q)) \tag{18}$$

The final estimate of X_k^Q, is obtained from hard-limiting of the top-level LLR. In the case of conventional SPCPC, the decoding process would end here. Unlike SPCPC, the decoding process of CAMC passes through one more stage, as shown in Fig. 4. In CAMC, the decoded output is obtained from despreading of iteratively decoded signal. The LLR values of the iteratively decoded bits are multiplied by $\mathbf{H}^{\mathrm{N}}$ and then the information bits are extracted from the despread hypercube.

Though the structure of CAMC is similar to conventional SPCPC, some distinct features of CAMC provides advantage over SPCPC. The first advantage is the processing gain which comes from the spread spectrum property of CAMC. While, in conventional SPCPC, correction of bit errors is performed only in the iterative decoding stage, additional correction of bit errors is also achieved in the despreading of the iteratively decoded bits in CAMC.

Another advantageous feature is the fixed weight of CAMC, as will be described in Section VI. In error correcting codes, the minimum distance of the code has been regarded as an important factor for BER performances. Battail [13] suggested that, in SPCPC, the weight distribution is more important than the minimum distance.

Later, Biglieri [14] showed that iterated product codes have Gaussian weight distribution even when they are relatively

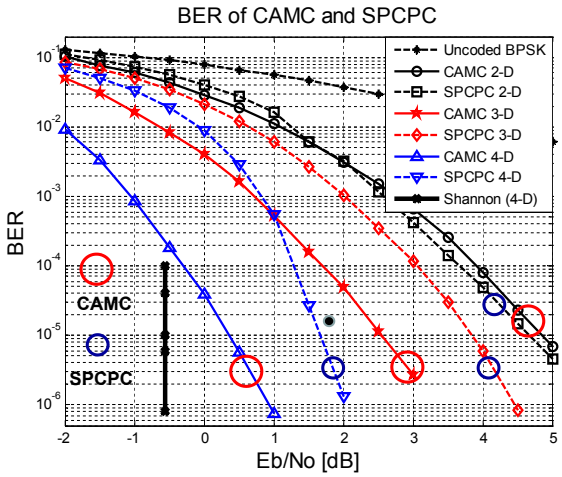

Fig. 6: CAMC, SPCPC and Shannon limit

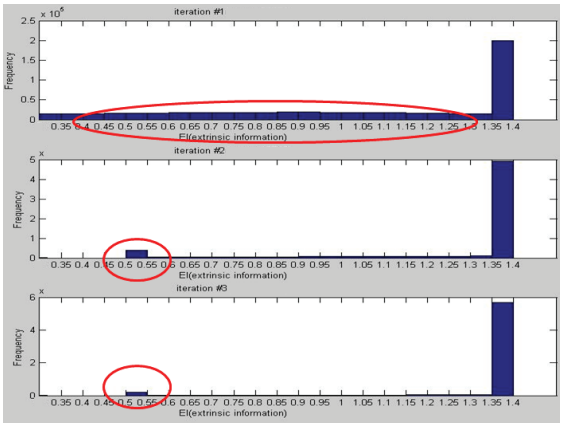

Fig. 7: Over iterations, |EI| converges to $|E_{max}|$

short codes. The random-like criterion for designing a product code (or a turbo code) was supported by several other works [15]. A product code with Gaussian weight usually has good performance. Some of the codewords, however, at the tail of the Gaussian distribution have small weights and can affect the performance adversely.

While conventional SPCPC usually has Gaussian weight distribution, the codeword weights of CAMC are very close to the $N/2$. Hence, CAMC does not have any low weight codewords which usually degrade the performance at low SNR.

As a comparison of (12),(13),(14) with(16),(17),(18) shows, iterative decoding of CAMC is more expensive than the iterative decoding of conventional product codes since the computation is crossing over the whole dimensionality of CAMC. This computational cost, however, helps to achieves a performance improvement in error correction.

In Fig. 6 is shown the performance of CAMC compared with that of corresponding SPCPC with $n = 4$ [7]. BPSK modulated signal is transmitted through a binary-input AWGN channel. The code rates are $R_2 = 9/16$ (2-D), $R_3 = 27/64$ (3-D) and $R_4 = 81/256$ (4-D), respectively. For BER of 10^{-5}, CAMC outperforms SPCPC by 1.3∼1.4 dB. The Shannon capacity limit of the binary-input AWGN channel for code rate $81/256$ is -0.55 dB. Hence, 4-D CAMC is only 0.95 dB away from Shannon limit [12].

V. On/Off Controlled Despreading of CAMC

EI is a rough estimate of the reliability for the received signal. We examined how the distribution of EI changes over the iterative steps [8]. Fig. 7 shows the histogram of |EI| during the process of the iterated decoding. At the initial stage, a large part of the EI values are randomly distributed in the range between $-E_{max}$ and $+E_{max}$. Gradually, EI converges either to $+E_{max}$ for a positive bit or to $-E_{max}$ for a negative bit. $\pm|E_{max}|$ is the saturation value of EI which is set to prevent EI from diverging.

The convergence of EI is highly correlated with the process of error correction. Fast convergence to $\pm|E_{max}|$ implies that errors continue to be corrected. On the contrary, slow convergence or no convergence imply that soft decision values are close to zero and the polarity of EI randomly toggles. Practically, no errors are being corrected when EI does not converge.

We can use EI as a performance predictor of despreading control, where the information bits are finally obtained from despreading of iteratively decoded signal. In the despreading process of the previous work [7], the LLR values of *all* the iteratively decoded bits are multiplied by $\mathbf{H}^{\mathrm{N}}$.

We modified this scheme such that such that only those bits with EI converged to $\pm|E_{max}|$ keep their LLR values. The other bits with $|\mathrm{EI}| < |E_{max}|$ are assigned LLR values of zero [8]. The basic idea is that we perform despreading only when it is likely to help to reduce the bit errors. This algorithm has been tested and it brings a performance gain of about 0.2 dB for 4-D CAMC [16].

VI. Differentiated Assignment of EI by Weight

An interesting feature of CAMC is that, unlike conventional SPCPC, the weights of CAMC are evenly distributed at two fixed values [9].

$$w(\mathbf{v}^{\mathrm{N}}) = (N \pm \sqrt{N})/2 \tag{19}$$

This is true of codewords at any level. When a N-bit-long codeword is despread into sub-level codewords of L-bit-long, they also have weights of $(L \pm \sqrt{L})/2$. Weights for code length of $N = 4 \sim 64$ are shown in Table I.

TABLE I
Weights of CAMC for $N = 4 \sim 64$

Code Length (N)	4	16	64
Dimension	1	2	3
Weights	1,3	6,10	28,36

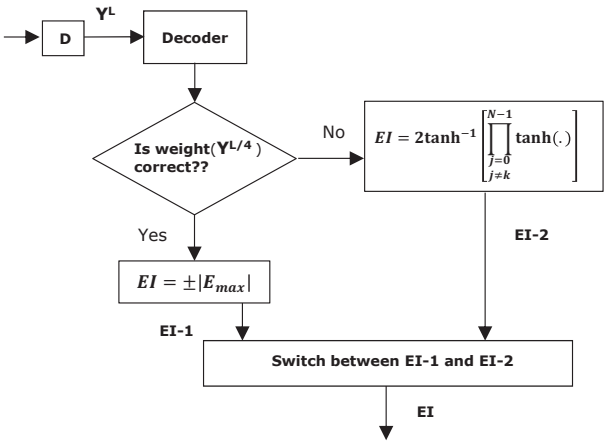

Fig. 8: EI converges fast to $\pm|E_{max}|$ if weight is correct.

The property that CAMC has fixed weight can be used to detect erroneous received vectors. When the weight of $\mathbf{y}^{\mathrm{L}}$ is not equal to $(L \pm \sqrt{L})/2$, then at least one bit of $\mathbf{y}^{\mathrm{L}}$ is erroneous. The uncertainty range of the bit error is the entire length of the received vector. This range can be reduced to $1/4$ by despreading $\mathbf{y}^{\mathrm{L}}$ into four quarter-sized vectors and examining the weight of each of them. This is the motivation of differentiated assignment of EI to codewords in accordance with their weight integrity.

A. Reduction of Uncertainty Range by Despreading

The received signal $\mathbf{y^L}$ is a sum of a CAMC vector and a noise vector.

$$\begin{aligned} \mathbf{y}^{\mathrm{L}} &= \mathbf{v}^{\mathrm{L}} + \mathbf{n}^{\mathrm{L}} \\ v_i &\in \{+1, -1\} \\ y_i &\begin{cases} \text{no error} & -1 \le n_i \le +1 \\ \text{error} & \text{if } v_i = -1 \text{ AND } n_i \ge +1 \\ \text{error} & \text{if } v_i = +1 \text{ AND } n_i \le -1 \end{cases} \end{aligned} \tag{20}$$

Despreading $\mathbf{y}^{\mathrm{L}}$ by $\tilde{\mathbf{H}}^{\mathrm{N}}$ generates three CAMC vectors and a parity vector. Using (6) and (10), we get:

$$\begin{aligned} & \frac{1}{2} \cdot \mathbf{y}^{\mathrm{L}} \cdot \tilde{\mathbf{H}}^{\mathrm{L}} \\ &= \frac{1}{2} \cdot [\mathbf{v}^{\mathrm{L}}_{0/4} + \mathbf{n}^{\mathrm{L}}_{0/4} \mid \mathbf{v}^{\mathrm{L}}_{1/4} + \mathbf{n}^{\mathrm{L}}_{1/4} \mid \ldots] \cdot \tilde{\mathbf{H}}^{\mathrm{L}} \\ &= \quad [\mathbf{y}^{\mathrm{L}/4}_0 \mid \mathbf{y}^{\mathrm{L}/4}_1 \mid \mathbf{y}^{\mathrm{L}/4}_2 \mid \mathbf{y}^{\mathrm{L}/4}_3] \end{aligned} \tag{21}$$

Then, it follows that each of the four despread vectors, $\{\mathbf{y}^{\mathrm{L}/4}_0, \mathbf{y}^{\mathrm{L}/4}_1, \mathbf{y}^{\mathrm{L}/4}_2, \mathbf{y}^{\mathrm{L}/4}_3\}$ is a sum of a regular CAMC vector (or a parity vector) and noise vectors.

$$\mathbf{n}^{\mathrm{L}} = [\mathbf{n}^{\mathrm{L}}_{0/4} \mid \mathbf{n}^{\mathrm{L}}_{1/4} \mid \mathbf{n}^{\mathrm{L}}_{2/4} \mid \mathbf{n}^{\mathrm{L}}_{3/4}] \tag{22}$$

$$\begin{aligned} \mathbf{y}^{\mathrm{L}/4}_0 &= \mathbf{v}^{\mathrm{L}/4}_0 + \frac{1}{2}\left\{\mathbf{n}^{\mathrm{L}}_{0/4} + \mathbf{n}^{\mathrm{L}}_{1/4} + \mathbf{n}^{\mathrm{L}}_{2/4} + \mathbf{n}^{\mathrm{L}}_{3/4}\right\} \\ \mathbf{y}^{\mathrm{L}/4}_1 &= \mathbf{v}^{\mathrm{L}/4}_1 + \frac{1}{2}\left\{\mathbf{n}^{\mathrm{L}}_{0/4} - \mathbf{n}^{\mathrm{L}}_{1/4} + \mathbf{n}^{\mathrm{L}}_{2/4} - \mathbf{n}^{\mathrm{L}}_{3/4}\right\} \\ \mathbf{y}^{\mathrm{L}/4}_2 &= \mathbf{v}^{\mathrm{L}/4}_2 + \frac{1}{2}\left\{\mathbf{n}^{\mathrm{L}}_{0/4} + \mathbf{n}^{\mathrm{L}}_{1/4} - \mathbf{n}^{\mathrm{L}}_{2/4} - \mathbf{n}^{\mathrm{L}}_{3/4}\right\} \\ \mathbf{y}^{\mathrm{L}/4}_3 &= \mathbf{p}^{\mathrm{L}/4} + \frac{1}{2}\left\{\mathbf{n}^{\mathrm{L}}_{0/4} - \mathbf{n}^{\mathrm{L}}_{1/4} - \mathbf{n}^{\mathrm{L}}_{2/4} + \mathbf{n}^{\mathrm{L}}_{3/4}\right\} \end{aligned} \tag{23}$$

The added noise is one half of the sum of four quadrants of the noise vector. If we assume that $\mathbf{n}^{\mathrm{L}}$ is an AWGN random process, then the noise component in $\mathbf{y}^{\mathrm{L}/4}_i$ is also an AWGN random process and the noise variance in each bit of $\mathbf{y}^{\mathrm{L}/4}_i$ is the same as that of each bit of $\mathbf{y}^{\mathrm{L}}$.

With equal noise power, the expected number of bit errors is also the same in $\mathbf{y}^{\mathrm{L}}$ and in $[\mathbf{y}^{\mathrm{L}/4}_0 \mid \mathbf{y}^{\mathrm{L}/4}_1 \mid \mathbf{y}^{\mathrm{L}/4}_2 \mid \mathbf{y}^{\mathrm{L}/4}_3]$. For example, if $\mathbf{y}^{\mathrm{L}}$ has a single bit error, then it is highly likely that only one of $\mathbf{y}^{\mathrm{L}/4}_i,\ i \in \{0, 1, 2, 3\}$, is in error and the other three are free from errors. In that case, we can easily find which one of the four is in error by examining the weight of each of $\mathbf{y}^{\mathrm{L}/4}_i$. As a result, the uncertainty range of the position of bit error is reduced to $1/4$ of the initial range.

B. Assignment of EI Based on Weight Integrity

The fixed value of code weight can guide the iterative decoding. In a product code, the magnitude LLR of a bit is a rough estimate of the reliability of the bit, that is, how well the parity relation is consistent with other bits. The magnitude of LLR for noisy bits is usually low.

In the iterative decoding of CAMC, the LLR of a codeword is a weighted sum of LLR values of associated codewords from which it is despread or into which it is spread. If any of the codewords have a wrong weight, then they are definitely in error. The iterative decoding can be improved by differentiated handling of EI of codewords in accordance with their weight integrity. We can give larger confidence to those codewords with correct weights.

We propose to let EI quickly converge to $\pm E_{max}$ if the weight is correct. When the weight of the codeword has the correct weight, EI for the bits in the codeword takes on the saturation value. We set EI $= +|E_{max}|$ or EI $= -|E_{max}|$ if the sum in (17) is positive or negative, respectively. For the other bits in codewords with wrong weights, the computation of EI follows the normal computation in (17).

Fig. 8 is the modified block for EI computation which replaces the dotted block in Fig. 5. The modified version of EI computation in (17) is shown in (24).

$$E_q(X_k^q) = \begin{cases} 2\tanh^{-1}\left[\prod_{j=0,j\neq k}^{N-1}\tanh\left(\frac{A_q(X_j^q)+\frac{2}{\sigma^2}Y_j^q}{2}\right)\right] \\ \qquad \text{bits in wrong codeword} \\ \pm E_{max} \\ \qquad \text{bits in correct codeword} \end{cases} \tag{24}$$

VII. Results of Differentiated EI

The performance of the differentiated EI assignment is tested in a computer simulation. Comparison with two previous results are presented. Both results are for BPSK modulations of 2-D, 3-D and 4-D of CAMC in binary-input AWGN channel. First, a comparison with plain iterated decoding of CAMC [7] is shown in Fig. 9. A gain of 0.1∼0.3 dB in E_b/N_0 is achieved.

Second, a comparison with despreading On/Off control in Section V is shown in Fig. 10. We get a gain of about 0.1 dB. Though the amount of gain appears small, this gain is meaningful near the Shannon capacity limit, which is achieved by CAMC. We can observe that most of the improvement are obtained in channels of high SNR. For low SNR channel, little improvement is obtained. A logical analysis is as follows:

There are two possibilities of a codeword having the correct weight. One is that an erroneous codeword with even number of bit errors with opposite polarity canceling each other happens to have a correct weight. When the channel SNR is low, this probability is not negligible. Favored assignment of EI to these *unqualified* codewords does not help. It may even slow down the convergence of EI.

The other possibility is that the codeword is indeed error-free. This probability gets larger as SNR is higher. Preferred assignment of EI at high SNR helps.

VIII. Conclusions

The codewords of CAMC have fixed weight of $(N \pm \sqrt{N})/2$. This feature provides outperformance of CAMC over conventional SPCPC. The uncertainty range of the position of bit errors can be reduced to $1/4$ by examining the weights of despread quarter-sized vectors. We showed that differentiated computation of EI depending on the integrity of a code weight helps to improve the performance.

In the proposed scheme, a codeword with a correct weight is given a larger confidence. EI values for their bits quickly converge to $\pm E_{max}$. Performance improvement of $0.1 \sim 0.3$ dB in E_b/N_0 are obtained, when compared with plain iterated decoding. A gain of about 0.1 dB is when compared with despreading control of CAMC where the improvement is mostly found in high SNR channel. Near the Shannon capacity limit, which is achieved by CAMC, even slight values of gains are meaningful.

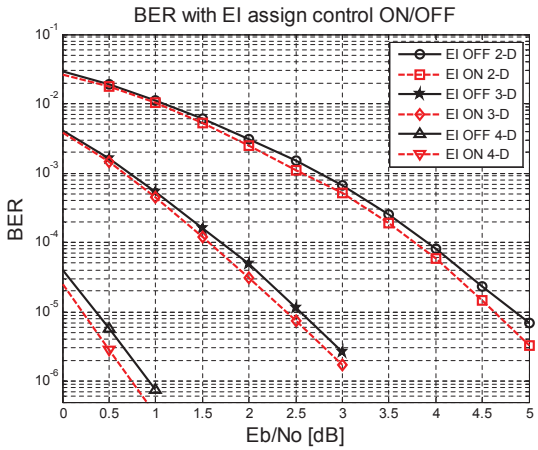

Fig. 9: Differentiated EI vs. plain decoding

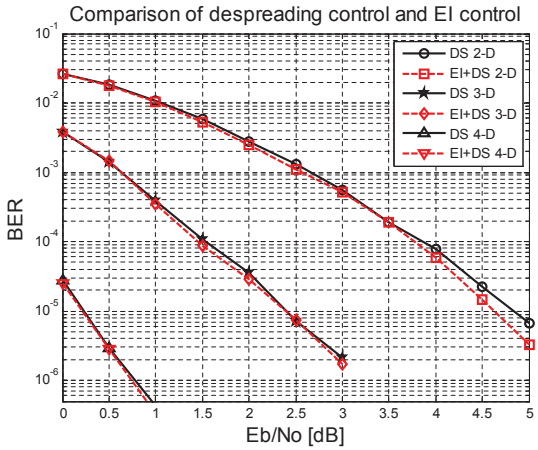

Fig. 10: Differentiated EI vs. despreading control

References

[1] P. Elias, "Error free coding", *IRE Trans. on Inform. Theory*, vol. IT-4, pp. 29-37, Sep. 1954

[2] D. Rankin and T. Gulliver, "Single parity check product codes," *IEEE Trans. on Comm.*, Vol.49, No.8, pp.1354-1362, Aug. 2001

[3] Y. Kim, "Recursive generation of constant amplitude multi-code S-CDMA signal," *IET Electronics Letters*, Vol. 39, No.25, pp.1782-1783, Dec. 2003

[4] I. Chih-Lin and R. Gitlin, "Multi-code CDMA wireless personal communications networks," in *IEEE Proceedings of ICC 1995*, pp.1060-1064, June, 1995

[5] T. Wada *et al.*, "A constant amplitude coding for orthogonal multi-code CDMA systems," *IEICE Trans. on Fund.*, vol. E80-A, pp. 2477-2484, Dec. 1997

[6] A. Shiozaki, M. Kishimoto and G. Maruoka, "Close-to-capacity performance of extended single parity check product codes," *IET Electronics Letters* Vol. 47 No. 1, Jan. 2011

[7] Y. Kim, "Constant amplitude multi-code CDMA with built-in single parity check product code," *IEEE Communications Letters*, Vol. 10, No.1, pp.4-6, Jan. 2006

[8] K. Lee, I. Park, B. Kim and Y. Kim, "Processing Gain in a Recursive Single Parity Check Product Code with Non-Gaussian Weight Distribution," *Proceedings of IEEE VTC 2009-Spring*, Barcelona, Spain, Apr. 2009

[9] I. Park, T. Kim and Y. Kim, "A Recursive Single Parity Check Product Code with Non-Gaussian Fixed Weight Distribution," *Proceedings of IEEE ATNAC 2008*, Adelaide, Australia, Dec. 2008

[10] G. Battail, "A conceptual framework for understanding turbo codes," *IEEE J. Selected Areas in Comm*., Vol.16, No.2, pp.245-254, Feb. 1998

[11] J. Hagenauer, E. Offer and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. on Inform. Theory*, Vol.42, pp. 429-445, Mar. 1974

[12] S. Benedetto and E. Biglieri, *Principles of digital communications with wireless applications*, Kluwer Academic, Chap.10, 1999

[13] G. Caire, G. Taricco and G. Battail, "A Weight distribution and performance of the iterated product of single-parity-check codes," *Proceddings of IEEE Globecom 1994*, pp.206-211, Dec. 1994

[14] E. Biglieri and V. Volski, "Approximately Gaussian weight distribution of the iterated product of single-parity-check codes," *IEE Electronincs Letters*, vol.30, No.12, pp.923-924, June, 1994

[15] D. Yue and E. Yang, "Aymptotically Gaussian weight distribution and performance of multicomponent turbo block codes and product codes," *IEEE Trans. on Comm*, vol.52, No.5, pp.728-736, May, 2004

[16] S. Chon and Y. Kim, "Contribution of Processing Gain in Turbo Decoding of a Recursive Single Parity Check Product Code," *Information Journal* vol.13 no. 2, pp. 329-338, March, 2010, International Information Institute, Japan

[17] S. Lin and D. Costello, *Error control coding*, 2nd ed. Pearson Prentice Hall, 2004

Wonsun Bong received the B.S. in electrical engineering from Cheongju University, Korea, in 2009 and the M.S. degree in the electrical and computer engineering from the University of Seoul, Korea, in 2011. He is currently working toward the Ph.D. degree at the department of electrical and computer engineering in the University of Seoul, Korea.

His research interests are wireless communications, signal processing with an application to security surveillance systems from image sand videos.

Yong Cheol Kim (M'93) received the B.S. degree in electronics engineering from Seoul National University, Korea, in 1981 and the M.S. degree in electrical engineering from KAIST, Korea, in 1983 and the Ph.D. degree in electrical engineering from University of Southern California, Los Angeles, in 1993.

From 1993 to 1996, he was a team leader in Military Digital Communications Sector in LIG Nex1, Korea. In 1996, he joined the faculty at the Department of Electrical Engineering, University of Seoul, Korea. He is currently a Professor at the Department of Electrical and Computer Engineering. His research interests include mobile communications, image processing and computer vision.

Professor Kim is a member of Association for Computing Machinery, Institute of Electronics Engineers of Korea and The Korean Institute of Communications and Information Sciences.