

# Max-Log-MAP Filtering Algorithm for Decoding Product $F_{24}$ Code<sup>†</sup>

Li Ping, Sammy Chan and Kwan L. Yeung  
Department of Electronics Engineering  
City University of Hong Kong, Hong Kong  
Email for the contact person: eeliping@cityu.edu.hk

## Abstract

This paper presents a symbol-by-symbol decoding method for the  $F_{24}$  code. It forms the core part of an iterative Max-Log-MAP filtering algorithm for the product  $F_{24}$  code and noticeable coding gain is observed by simulation. The complexity of the proposed algorithm is very modest. The relatively short frame length of the product  $F_{24}$  code can be an advantage for its applications in some communication systems.

## 1. Introduction

Concatenated codes and their iterative decoding methods have attracted much research interest recently. This is to a large extent due to the discovery of the parallelly concatenated convolutional codes, the so-called turbo codes [1-3], which have demonstrated remarkable performance at very low signal to noise ratio (SNR).

Product block codes are very similar to concatenated codes [4]. Potential error protection capabilities of product block codes have long been known but their applications have been limited by the lack of efficient maximum likelihood (ML) decoding algorithms. The iterative decoding technique offers a promising solution to the problem [5-6].

Product block codes have the advantage of achieving high coding gain with relatively short frame length. Many communication systems have strict delay constraints which impose a limit on frame length [7]. We will show that for such applications, product block codes can be competitive candidates.

The iterative decoding algorithm of product codes requires symbol-by-symbol soft-in/soft-out decoding of the constituent codes. For simple codes, it can be

accomplished by the MAP (maximum *a posteriori*) algorithm [8] together with the trellis description technique [5, 9]. However this approach becomes too computationally costly for codes with even modest complexity, such as the code discussed below.

$F_{24}$  denotes a [24, 12, 4] self-dual code [10,11]. Its performance is very similar to that of the [24, 12, 8] Golay code but ML decoding complexity of the former is considerably lower [11, 12]. The two dimensional product  $F_{24}$  code, denoted by  $F_{24}^2$ , has a frame length of 576 bits including 144 information bits. In the following we will first derive a Max-Log-MAP filtering algorithm for product codes that avoids probability domain operations. We will then present a simple symbol-by-symbol soft-in/soft-out decoding method for  $F_{24}$ , which forms the core part of the Max-Log-MAP filtering algorithm for  $F_{24}^2$ .

The computational complexity of the proposed approach is 128 addition-equivalent operations per information bit per iteration.

## 2. Max-Log-MAP filtering

MAP filtering is a concept introduced in [5]. The following is a modified version of the MAP filtering algorithm in which probability domain operations are avoided. Consider a length  $n$  binary block code  $C$ . For every codeword  $c = \{c_b\} \in C$ , we generate a vector, denoted by  $\hat{c}$ , by applying the rule  $\{0 \rightarrow +1, 1 \rightarrow -1\}$  on  $c$ . Without confusion we will call both  $c$  and  $\hat{c}$  codewords. The distorted vector is denoted by  $x = \{x_b\} \in \mathbf{R}^n$ , which is the sum of  $\hat{c}$  and a noise vector. The MAP filter is a function of  $x$  whose output  $f(x)$  is a vector  $\{f_b\} \in \mathbf{R}^n$  defined by,

<sup>†</sup> This work is supported by the HK UGC grant of number 9040205.

$$f_b = \log \frac{\Pr\{\hat{c}_b = +1 | x, C\}}{\Pr\{\hat{c}_b = -1 | x, C\}} \quad (1)$$

For decoding a multi dimensional product code, the MAP filtering algorithm applies (1) to every dimension of  $x$  independently (therefore it is sub-optimal). After each filtering, a replacement of

$$\{f_b\} \Rightarrow \{x_b\} \quad (2)$$

is performed, resulting in a refined estimate of  $\{\hat{c}_b\}$ . Upon completing all the dimensions, the process can be restarted in an iterative manner. The convergence of the algorithm is treated in [6].

Solving (1) is generally a very computationally intensive task. Based on the Max-Log-MAP principle of [13], we can approximate (1) as,

$$f_b \approx \sigma^{-2} (W_b^+ - W_b^-) \quad (3)$$

$$\text{with } W_b^+ = \max_{\hat{c}_b = +1} \{ \langle x, \hat{c} \rangle \} \quad W_b^- = \max_{\hat{c}_b = -1} \{ \langle x, \hat{c} \rangle \} \quad (4)$$

In the above  $\langle x, \hat{c} \rangle$  denotes the inner product of  $x$  and a codeword  $\hat{c}$ . The maximization are over all the codewords whose  $b$ -th bit is  $+1$  and  $-1$ , respectively. Since  $\hat{c}_b = \pm 1$ , (4) involves no multiplication. It can be verified that the factor  $\sigma^{-2}$  in (3) has no impact on the final results in the above algorithm and therefore it can be ignored. This can greatly save the effort of estimating noise variance.

Direct computation of (3) is still difficult for most block codes with high coding gain. We will treat the problem for  $F_{24}$  in the next section.

Notice that the concept of symbol-by-symbol decoding based on (1) is slightly different from the original MAP algorithm of [8] in that (1) is applied to coded symbols, rather than information symbols.

### 3. Symbol-by-symbol decoding of $F_{24}$

The  $F_{24}$  code is a length 24 binary code. For convenience, we shall represent its codeword as a  $4 \times 6$  array [11],

$$c = \begin{bmatrix} c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} & c_{1,6} \\ c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} & c_{2,5} & c_{2,6} \\ c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} & c_{3,5} & c_{3,6} \\ c_{4,1} & c_{4,2} & c_{4,3} & c_{4,4} & c_{4,5} & c_{4,6} \end{bmatrix} \quad (5)$$

The generator matrix of  $F_{24}$  code is given in (6), in which every entry represents a vector with interpretation given in (7). Thus every row in (6) can be expanded to a  $4 \times 6$  array as that in (5).

$$\begin{pmatrix} G^A \\ G^B \end{pmatrix} = \begin{pmatrix} C^{(1)} & C^{(0)} & C^{(0)} & C^{(1)} & C^{(2)} & C^{(3)} \\ C^{(2)} & C^{(0)} & C^{(0)} & C^{(2)} & C^{(3)} & C^{(1)} \\ C^{(0)} & C^{(1)} & C^{(0)} & C^{(1)} & C^{(3)} & C^{(2)} \\ C^{(0)} & C^{(2)} & C^{(0)} & C^{(2)} & C^{(1)} & C^{(3)} \\ C^{(0)} & C^{(0)} & C^{(1)} & C^{(1)} & C^{(1)} & C^{(1)} \\ C^{(0)} & C^{(0)} & C^{(2)} & C^{(2)} & C^{(2)} & C^{(2)} \\ C^* & C^{(0)} & C^{(0)} & C^{(0)} & C^{(0)} & C^{(0)} \\ C^{(0)} & C^* & C^{(0)} & C^{(0)} & C^{(0)} & C^{(0)} \\ C^{(0)} & C^{(0)} & C^* & C^{(0)} & C^{(0)} & C^{(0)} \\ C^{(0)} & C^{(0)} & C^{(0)} & C^* & C^{(0)} & C^{(0)} \\ C^{(0)} & C^{(0)} & C^{(0)} & C^{(0)} & C^* & C^{(0)} \\ C^{(0)} & C^{(0)} & C^{(0)} & C^{(0)} & C^{(0)} & C^* \end{pmatrix} \quad (6)$$

$$C^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad C^{(1)} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad C^{(2)} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \quad C^{(3)} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \quad C^* = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad (7)$$

We will call the first four vectors in (7) characters. They are closed under the bit-wise modulo-2 addition. Including  $C^*$ , the space spanned by the above five vectors are characters and their complements.

As shown in (5),  $F_{24}$  can be divided into two subcodes, generated by  $G^A$  and  $G^B$  respectively. The subcode spanned by  $G^A$  is a binary interpretation of the hexacode [12] and that spanned by  $G^B$  is geometrically similar to the  $[6, 6, 1]$  code [11]. There are altogether 4096 codewords in  $F_{24}$ . They are divided into 64 cosets, referred to as R-cosets. Every R-coset leader is one codeword in the subcode spanned by  $G^A$ . Other codewords in a R-coset can be obtained by taking complements of the columns of

their coset leader. Denote by  $P_j^{(k)}$  the set of codewords in  $F_{24}$  whose  $j$ -th column is either  $C^{(k)}$  or  $C^{(k)}+C^*$ . It can be shown that  $\{P_j^{(k)} \mid k=0,1,2,3\}$  is also a coset partition of  $F_{24}$ . We will refer to  $P_j^{(k)}$  as a P-coset. Every P-coset is the union of 16 R-cosets. There are a total of 6 different partitions of  $F_{24}$  in this way for  $j=1,2,\dots,6$ , resulting 24 different P-cosets. P-cosets resulted from different partitions partially overlap.

We shall use  $\hat{C}^{(k)}$ ,  $k=0, 1, 2, 3$ , and  $C^*$  to denote the vectors obtained by applying the rule  $\{0 \rightarrow +1, 1 \rightarrow -1\}$  to the vectors in (5). A codeword obtained from the same rule is denoted by  $\hat{c} = \{\hat{c}_{i,j}\}$ . The distorted vector  $x$  is also arranged as a  $4 \times 6$  array. Given  $x$ , we call the inner product  $\langle x, \hat{c} \rangle$  the confidence value of  $\hat{c}$ . Our aim is to find  $W_{i,j}^+$  and  $W_{i,j}^-$  for all  $i$  and  $j$ , where  $W_{i,j}^+$  and  $W_{i,j}^-$  are the equivalence of  $W_b^+$  and  $W_b^-$  (The difference is the double subscripts here due to the array form arrangement of codewords). Using the terminology just introduced,  $W_{i,j}^+$  (resp.  $W_{i,j}^-$ ) is the highest confidence value among the codewords whose  $i,j$ -th element is  $+1$  (resp.  $-1$ ). The algorithm to find them is given below. For complexity evaluations, we will only count real addition equivalent operations and ignore operations such as parity check, memory addressing and multiply-by-2. This is in compliance to the convention [11, 12].

**Step 1)** Compute all the inner products between columns of  $x$  and characters,

$$M_j^{(k)} = \langle \hat{C}^{(k)}, X_j \rangle \quad k = 0, \dots, 3 \text{ and } j = 1, \dots, 6 \quad (8)$$

where  $X_j$  is the  $j$ -th column of  $x$ . Also record down

$$s_j^{(k)} = \begin{cases} 0 & \text{if } M_j^{(k)} > 0 \\ 1 & \text{if } M_j^{(k)} < 0 \end{cases} \quad (9)$$

**Complexity:**  $8 \times 6 = 48$  additions using the Gray code technique [12].

**Step 2)** Compute  $u_m = \sum_{j=1}^6 |M_j^{(*)}|$  for the  $m$ -th R-coset,  $m=1,2,\dots,64$ . The superscript  $*$  stands for the appropriate character for the  $j$ -th column determined by the coset leader. The resulting  $u_m$  is the maximum codeword confidence value in the  $m$ -th R-coset.

**Complexity:** Use two sub-steps. First compute all the possible partial sums,  $|M_j^{(k')}| + |M_{j+1}^{(k'')}|$  for  $k', k''=0, 1, 2, 3$  and  $j=1, 3, 5$ , costing  $16 \times 3 = 48$  additions in total. Secondly add three partial sums together for every codeword, costing  $2 \times 64 = 128$  additions in total.

**Step 3)** Select  $U = \max\{u_m\}$ . Denote the corresponding codeword as  $c' = \{c'_{i,j}\}$ . To find this codeword, first use  $U$  to locate the coset leader and then reverse the column parity for which  $s_j^{(k)} = 1$ .

**Complexity:** 63 comparisons.

The resulted  $U$  equals either  $W_{i,j}^+$  or  $W_{i,j}^-$  for every pair of  $i$  and  $j$ , i.e.,

$$U = \begin{cases} W_{i,j}^+ & \text{if } c'_{i,j} = 0 \\ W_{i,j}^- & \text{if } c'_{i,j} = 1 \end{cases} \quad (10)$$

We still need to find the other wanted term

$$V_{i,j} = \begin{cases} W_{i,j}^- & \text{if } c'_{i,j} = 0 \\ W_{i,j}^+ & \text{if } c'_{i,j} = 1 \end{cases} \quad (11)$$

This is done in two steps. We first find the highest confidence values, denoted by  $p_j^{(k)}$  and  $q_j^{(k)}$ , among all the codewords whose  $j$ -th column are  $C^{(k)}$  and  $C^{(k)}+C^*$ , respectively, for  $j=1,2,\dots,6$  and  $k=0,1,2,3$ . They can be found by searching in 24 P-cosets  $\{P_j^{(k)}\}$ . We then use the results to find  $V_{i,j}$ .

**Step 4)** For every P-coset  $P_j^{(k)}$  compute,

$$U_j^{(k)} = \max_{P_j^{(k)}} \{u_m\} \quad (12)$$

Define

$$p_j^{(k)} = \begin{cases} U_j^{(k)} & \text{if } s_j^{(k)} = 0 \\ U_j^{(k)} - 2|M_j^{(k)}| & \text{if } s_j^{(k)} = 1 \end{cases} \quad (13a)$$

$$q_j^{(k)} = \begin{cases} U_j^{(k)} - 2|M_j^{(k)}| & \text{if } s_j^{(k)} = 0 \\ U_j^{(k)} & \text{if } s_j^{(k)} = 1 \end{cases} \quad (13b)$$

Eqn. (13) can be verified based on the following observations. When  $s_j^{(k)}=0$ ,  $M_j^{(k)}>0$  and its sign has not been reversed in Step 2. When  $s_j^{(k)}=1$ ,  $M_j^{(k)}<0$  and its sign has been reversed in Step 2.

**Complexity:**  $15 \times 24 = 360$  comparisons and 24 additions. The computation of multiply-by-2 is ignored.

**Step 5)** Denote the  $i$ -th element of  $C^{(k)}$  by  $C_i^{(k)}$ .

Define an indication function

$$f_{i,j}^{(k)} = C_i^{(k)} \oplus c'_{ij} \quad (14)$$

where  $\oplus$  denotes the exclusive-OR function. Notice that  $f_{i,j}^{(k)}=0$  indicates  $c'_{ij}=C_i^{(k)}$  while  $f_{i,j}^{(k)}=1$  indicates  $c'_{ij}=\overline{C_i^{(k)}}$ , with  $\overline{C_i^{(k)}}$  being the logic complement of  $C_i^{(k)}$ .  $V_{ij}$  can be found as

$$V_{ij} = \max_k \{ f_{i,j}^{(k)} p_j^{(k)}, \overline{f_{i,j}^{(k)}} q_j^{(k)} : k = 0, 1, 2, 3 \} \quad (15)$$

Both  $f_{i,j}^{(k)}$  and  $\overline{f_{i,j}^{(k)}}$  are used as integers in (15) to pick out unwanted confidence values. These unwanted values belong to codewords with the same  $i,j$ -th element as  $c'$  and so they should not be used in searching for  $V_{ij}$ .

**Complexity:**  $24 \times 3 = 72$  comparisons.

There are another 24 subtractions to complete eqn. (3). The total cost of the above algorithm is  $48+176+63+384+72+24=767$  operations per decoding, or about 64 operations per information bit

in average. For a two dimensional code this amounts to 128 operations per information bit per recursion.

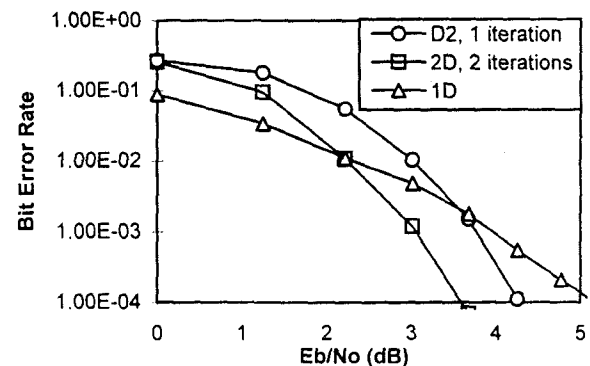
#### 4. Non-systematic $F_{24}^2$ code

It has been observed that the systematic  $F_{24}^2$  results in a higher BER than the non-systematic  $F_{24}^2$ , although their frame error rates are the same. This is for the following reason. The most probable error events are associated with the six weight 4 codewords in  $F_{24}$ , which are the rows of  $G^B$  in (6). Such an event causes parity reverse of a whole column in a codeword, resulting in one erroneous information bit in a non-systematic code. However in a systematic code, this resulting in two erroneous information bits in average, since one column contains two information bits in average in  $F_{24}$ .

The encoding of the non-systematic  $F_{24}$  is straightforward that the codeword is the direct product of the information vector and the generation matrix in (6).

#### 5. Performance

The simulated performances of the Max-Log-MAP filtering algorithm for the non-systematic  $F_{24}^2$  are shown in Fig.1 where performance of  $F_{24}$  is also included. BER of  $10^{-4}$  is observed at  $E_b/N_0 \approx 3.6$  dB with two iterations. More iterations lead to only marginal improvement.



**Fig.1** Performances of non-systematic  $F_{24}$  and  $F_{24}^2$  using the Max-Log-MAP filtering algorithm. Curves  $F_{24}$  and  $F_{24}^2$  are labeled by 1D and 2D respectively.

It is interesting to examine a modified method, in which instead of (2), the replacement is given by

$$x_b + \alpha (W_b^+ - W_b^-) \Rightarrow x_b \quad (16)$$

The purpose of (16) is to reduce the correlation caused by MAP filtering. The factor  $\alpha$  can also be regarded as a damping factor, if we treat the whole process as an optimization. Smaller  $\alpha$  may reduce convergence speed but the process is less prone to error. The same damping factor is applied to all iterations except the last one. Damping is not used in the last iteration since otherwise considerable noise component may remain in the final output, especially for small iteration numbers. The performance for  $\alpha=1/8$  is shown in Fig.2. With 4 iterations, BER of  $10^{-4}$  at  $E_b/N_0 \approx 2.8$  dB can be achieved. Further reducing  $\alpha$  or increasing iteration number only have marginal impact.

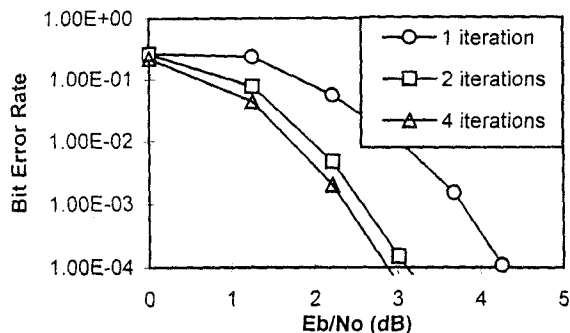


Fig.2 Performances of  $F_{24}^2$  with  $\alpha=1/8$ .

## 5. Conclusion

We have introduced a soft-in-soft-out algorithm for decoding  $F_{24}$  code and demonstrated its application in iterative decoding of the product  $F_{24}$  code. We have shown that noticeable coding gain can be achieved by the proposed scheme. The decoding complexity of the proposed scheme is very modest. The product  $F_{24}$  code has a relatively low rate of  $1/4$ . It may find applications in systems where low rate codes are suitable such as CDMA systems.

## References

[1] C. Berrou, A. Glavieux and P. Theitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes",

IEEE Proc. ICC-93, pp. 1064-1070, Geneva, Switzerland, May 1993.

[2] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systematic (Turbo) codes", Proc. IEEE ICC-94, pp. 1298-1303, New Orleans, May 1994.

[3] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes", IEEE Trans. Commun., vol. COM-44, pp. 591-600, May 1996.

[4] F.M. MacWilliams and N.J.A. Sloane, The Theory of Error-Correcting Codes, Vols. I and II, North-Holland, Amsterdam, 1977.

[5] J. Lodge, R. Young, P. Hoeher and J. Hagenaur, "Separable MAP "filter" for decoding of product and concatenated codes", Proc. IEEE ICC-93, pp.1740-1745, Geneva, May 1993.

[6] M. Moher, "Decoding via cross-entropy minimization", Proc. IEEE ICC-93, pp. 809-813, Geneva, May 1993.

[7] P. Jung and M. Naßhan, "Performance evaluation of turbo codes for short frame transmission systems", Electronics Letters, vol. 30, pp.111-112, Jan. 1994.

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

[9] J. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis", IEEE Trans. Inform. Theory, vol. IT-24, pp. 76-81, January 1978.

[10] V. Pless and N.J.A. Sloan, "On the classification of self-dual codes". J. Combinational Theory, vol. 18A, pp.313-335, 1975.

[11] M. Ran and J. Snyders, "On maximum likelihood soft decoding of some binary self-dual codes", IEEE Trans Inform. Theory, vol. IT-41, pp.439-443, March 1991.

[12] A. Vardy and Y Ber'ry, "More efficient soft decoding of the Golay codes", IEEE Trans Inform. Theory, vol. IT-37, p.667-672, May 1991.

[13] P. Robertson, E. Villebrun and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain", Proc. IEEE ICC-95, pp. 1009-1013, Seattle, June 1996.