Half-Product Codes for Flash Memory

Santosh Emmadi†, Krishna R. Narayanan†, Henry D. Pfister§
†Department of Electrical and Computer Engineering, Texas A&M University
§Department of Electrical and Computer Engineering, Duke University

I. Introduction

Error correction coding (ECC) is a critical component in the design of reliable flash memories. Currently popular ECC schemes for use in flash memory include low density parity check (LDPC) codes and product codes (PCs) based on Bose-Chaudhuri Hocquenghem (BCH) and Reed-Solomon (RS) codes. In some applications where only hard decisions are available at the detector output, PCs can provide computationally more efficient solutions than LDPC codes. In this paper, we show that a special case of PCs, called *half-product codes* (HPCs) can outperform product codes both in the waterfall and error floor regions of the bit error rate performance, while still enjoying simple encoding and decoding algorithms. Half-product codes were first discussed as a possible idea by Justesen in [1], but they have remained relatively unknown. These codes are analyzed in detail and many important properties relating to their minimum distance and stopping set sizes are uncovered in [2]. The current paper is a follow up to [2] that provides a case study comparing PC with HPC for a typical situation of interest in flash memories with $K = 4$Kbytes of information, and rate 0.88 to show the advantages of HPC.

II. Product Codes

Product codes (PCs) are formed by arranging the code symbols in a matrix form with rows forming one component code and columns forming another. Let C_1 be a (n_1, k_1, d_1) linear block code and C_2 be a (n_2, k_2, d_2) linear block code. The product code $C = C_1 \otimes C_2$ consists of all matrices whose rows are in C_1 and whose columns are in C_2. The product code C is an $(n_1 n_2, k_1 k_2, d_1 d_2)$ linear block code.

A. Minimum stopping sets and their performance

A PC is typically decoded using a cascade decoder, which is an iterative row-column decoder operating on the rows and columns in succession. With this decoding there is scope for stopping sets which are the error patterns that lead to a non-codeword at the end of the cascade decoding which can not be further decoded. For PC with identical component codes,

- The minimum stopping set is given as $s_{min}^{PC} = (t+1)^2$, where t is the error correction capability of the component code.
- The total number of possible minimum stopping sets in an PC structure is given as $N_{PC} = \binom{n}{t+1}^2$.
- The error performance due to these error patterns can be approximated to $P_{E,PC}^{ss} = \binom{n}{t+1}^2 p^{(t+1)^2}$.

III. Half-product Codes

Half-product codes (HPCs) are formed by arranging the code elements in a symmetric matrix with rows and columns from the same component codes, and taking only the upper (or lower) half of this matrix. The diagonal elements are all taken to be zeros. The half-product code formed by the component code $C_1(n, k)$ is given by $C(N, K)$ with length $N = \frac{n(n-1)}{2}$ and dimension $K = \frac{k(k-1)}{2}$.

$$\begin{array}{ccccccc} 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ & 0 & 1 & 1 & 1 & 0 & 0 \\ & & 0 & 1 & 0 & 0 & 0 \\ & & & 0 & 1 & 0 & 0 \\ & & & & 0 & 0 & 0 \\ & & & & & 0 & 0 \\ & & & & & & 0 \end{array} \qquad \begin{array}{ccccccc} 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Half form Full form

Fig. 1. $(21, 6)$ Half-product codeword in half form, and full form

Even though we only take the half of the encoded matrix for transmission, we will consider its equivalent full form for analysis purposes. The codeword, in full symmetric matrix form, is called the *codeword in full form*, and the upper half triangle is called the *codeword in half form*. Figure 1 gives an example for $(21, 6)$ half-product codeword, formed by $(7, 4)$ Hamming code as component code.

A. Encoding and Decoding

For encoding of a half-product code, the message symbols are filled only in the upper half of a message matrix with diagonal elements being zeros. The lower half of it is filled with the transpose of the upper half. This matrix, let us denote it by M, is then encoded using encoding operation of a product code with same component codes, $C_1 = C_2$.

Theorem 1. *Encoding of M will give rise to a half-product codeword matrix which is symmetric and has all the diagonal positions as zeros if the component code is over $GF(2^m)$.*

We have chosen zeros in the diagonal because the diagonal positions may be weakly protected when compared to the off-diagonal positions. We can observe in a HPC matrix structure that each symbol in an off-diagonal position belongs to two different component codewords, whereas a symbol in the diagonal position belongs to only one codeword. It is also

possible for the diagonal symbols to carry information, but this structure loses the advantages of the HPC structure.

Decoding of HPC is carried out by simply considering the codeword in full form and applying a cascade decoder to the full form.

B. d_{min} of Half-product codes

HPCs are observed to have larger minimum distance than PCs. The minimum distance, d_{min} of HPCs is given by the following theorem.

Theorem 2. *Let a linear code C_1 with $d_{min} = d$ form a half-product code C. Then, the minimum distance of C is given by*

$$D_{min} \geq \begin{cases} \frac{3d^2}{4} & \text{if } d \text{ is even} \\ \frac{(d+1)(3d-1)}{4} & \text{if } d \text{ is odd} \end{cases}$$

C. Minimum stopping sets and their performance

Stopping sets are, as noted in the summary for product codes, the error patterns that lead to a non-codeword at the end of the cascade decoding which can not be further decoded.

- The minimum stopping set in an HPC is given as $s_{min}^{HPC} = \frac{(t+1)(t+2)}{2}$, where t is the error correction capability of the component code.
- The total number of possible minimum stopping sets in an HPC structure is given as $N_{HPC} = \binom{n}{t+2}$.
- The error performance due to these error patterns can be approximated to $P_{E,HPC}^{ss} = \binom{n}{t+2} p^{\frac{(t+1)(t+2)}{2}}$

D. Post processing

After the decoding process stops in a non-codeword, we cannot decode this word any further because the errors in each non-zero row is more than the error correcting capability of the component code $t = \frac{d-1}{2}$. However, we can detect a component word being in error. The process of *post processing* leverages this idea along with the product code structure to achieve the decoding of the stopping sets until a larger size.

The post processing for a product code is done as follows. Once the decoding stops in a non-codeword, the symbols of the detected error rows are all changed to erasures. Then the word is passed to column decoder. It decodes all the columns with number of erasures less than d. Other columns are all made to erasures. Then it is passed to the row decoder. The post-process decoding stops when either a word is fully decoded to a codeword or it stops in a non-codeword of larger size which can not be further decoded. This way the minimum stopping set of the product code is increased to $S_{min}^{PC} = d(t+1)$. The post processing in half-product code is similar to this, except the row-column decoding happens in every step of the iteration as given by the cascade decoder for half-product code. The minimum stopping set of the half-product code is now increased to $S_{min}^{HPC} = \frac{(d+1)(t+1)}{2}$.

IV. Comparative simulations of HPCs with PCs

We consider the design of a ECC scheme for $K = 32768$bits (4Kbytes) and rate 0.88. A good design for a PC is to use $(72, 68, 5)$, $t = 2$, shortened RS codes over GF(128) as component codes. It can be seen that using binary BCH codes does not allow an increase in the error correction capability of the component codes. The key advantage of the HPC comes from the fact that the component codes can be longer for the same overall block length and hence, for a fixed rate, can correct more errors. This allows us to use a $(102, 96, 7)$, $t = 3$, shortened RS code over GF(128). With post-processing, the minimum stopping set size for the PC is 15, whereas it is 16 for the half product code. As can be seen from Sections II and III, the multiplicities of these stopping sets of minimum size is substantially smaller for the half product code.

For a binary symmetric channel, the codeword error rate for both these codes is plotted as a function of the symbol error rate in Fig. 2. Since each RS code symbol corresponds to 7 bits, the cross over probability is 1/7 the symbol error rate. It can be seen that the HPC outperforms the PC in the waterfall region. It can be noted from the simulations that the post processing stage is crucial for both PC and HPC and that without this, the error floors can be pronounced. The bit error rates due to minimum stopping sets after post-processing for the PC and HPC codes are approximately 3.35×10^{-9} and 1.4×10^{-14}, respectively for a cross over probability of 4×10^{-3}. For these results, the actual decoder is not simulated and it is assumed that if the number of errors is less than the error correction capability of the code, the decoder will succeed and if the number of errors is more, the decoder will fail. This is optimistic since this ignores miscorrection of the decoder to a wrong codeword. Simulation results accounting for miscorrections and an analysis of the error floor will be presented at the workshop if the paper is accepted.

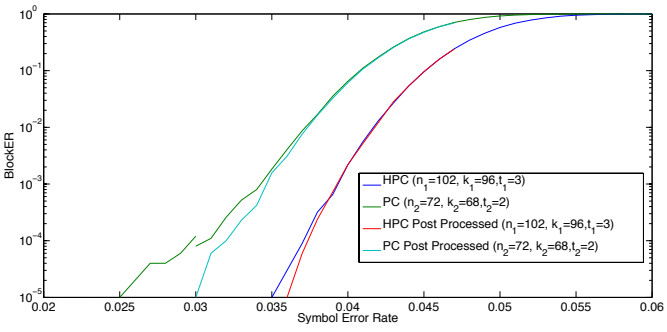

Fig. 2. Comparison of error performance of product code and half-product code; $K = 32768$, Rate=0.88

References

[1] J. Justesen, "Performance of product codes and related structures with iterated decoding," *Communications, IEEE Transactions on*, vol. 59, pp. 407–415, February 2011.

[2] H. Pfister, S. Emmadi, and K. Narayanan, "Symmetric product codes." to appear in proceedings of the Information Theory and its Applications Workshop, Feb 2015.