TURBO PRODUCT CODES BASED ON CONVOLUTIONAL CODES

Orhan Gazi, A. Özgür Yılmaz

ABSTRACT — In this article we introduce a new class of product codes based on convolutional codes, convolutional product codes (CPCs). The structure of product codes enables parallel decoding which can significantly increase decoder speed in practice. The use of convolutional codes in a product code setting lays the ground for utilizing all the flexibility and vast knowledge base for convolutional codes in fast parallel decoders. Interleaving turns out to be critical for the performance of convolutional product codes just as in turbo codes. The practical decoding advantages over serially concatenated convolutional codes are emphasized.

I. INTRODUCTION

One of the most successful works to approach the Shannon limit was published in 1993 by C. Berrou, A. Glavieux and P. Thitimajshima [1]. They introduced turbo codes, a.k.a. parallel concatenated convolutional codes (PCCC). Turbo codes can achieve bit error rate (BER) levels around 10^{-5} at code rates quite close to the corresponding capacity with reasonable decoding complexity. The use of soft-in soft-out decoding algorithms was a key in this success. In the last decade, similar codes such as serially concatenated convolutional codes (SCCCs) [2], LDPC codes [3], and block product codes have been extensively studied. The studies on product codes were initiated by Elias [4]. Product codes enjoy a high degree of parallelization as opposed to many other forms of concatenated code structures, e.g., PCCC. Product codes studied so far have been constructed using linear block codes, such as Hamming, extended Hamming [5]-[6], BCH [7]-[8], and Reed Solomon [9] codes. Single parity check (SPC) product codes are studied in [10]. Three and more dimensional SPC product codes are studied in [11]. Product codes have also attracted practical attention lately. DSP and FPGA implementations are studied in [12].

Product codes are traditionally constructed by linear block codes. Block codes have a trellis structure with a time varying property [13]. The product code we propose in this letter is constructed by using time-invariant convolutional codes. Its component codes' trellis structure does not vary in time as in product codes constructed with Hamming, extended Hamming, BCH, and Reed Solomon block codes. Moreover, the number of states in the trellis structure of a block code may grow exponentially with the difference of codeword and data block lengths [13], whereas the number of states in a convolutional code can be set as desired. The time invariant trellis structure of convolutional codes makes them more convenient for implementation. In addition, numerous practical techniques such as trellis coded modulation and puncturing can be simply utilized with convolutional codes as opposed to linear block codes. A code from the same family was previously studied for OFDM in [14] but wasn't analyzed and further elaborated.

Multi-input multi-output (MIMO) techniques are quite important to enhance the capacity of wireless communication systems. Space-time trellis codes provide both diversity and coding gain in MIMO channels and are widely used [15]. Space-time trellis codes usually have time-invariant trellis structures just like convolutional codes. Thus, a product code based on convolutional codes is more suitable for integration

Orhan Gazi (phone: +90 312 284 45 00 / 40 15 , e-mail: o.gazi@ari.cankaya.edu.tr), is with Electronics and Communication Engineering Department, Cankaya University, Ankara, Turkie.

A. Ozgur Yilmaz (e-mail: aoyilmaz@eee.metu.edu.tr) is with Electrical and Electronics Engineering Department, Middle East Technical University, Ankara, Turkie.

with MIMO channels and poses an alternative to block product codes.

Due to the advantages of convolutional codes mentioned above, we propose a class of product codes constructed by using convolutional codes and call them convolutional product codes (CPCs). In this paper, we will investigate the factors that affect the performance of CPCs and leave the issues with regard to space time trellis codes to other publications.

The outline of the paper is as follows. The proposed code structure and the decoding algorithm for CPCs are given in Section II. Minimum distance of these codes is studied in Section III. In Section IV, implementation advantages of CPCs are given. Simulation results are presented in Section V. Concluding remarks are given in Section VI.

II. CPC ENCODER AND DECODER

1. CPC Encoder

A regular product code is constructed by placing the information bits/symbols into a matrix. The rows and columns are encoded separately using linear block codes [5]-[8]. This type of a product encoder is shown in Fig. 1. It is seen from the figure that the data and parity bits are grouped separately.

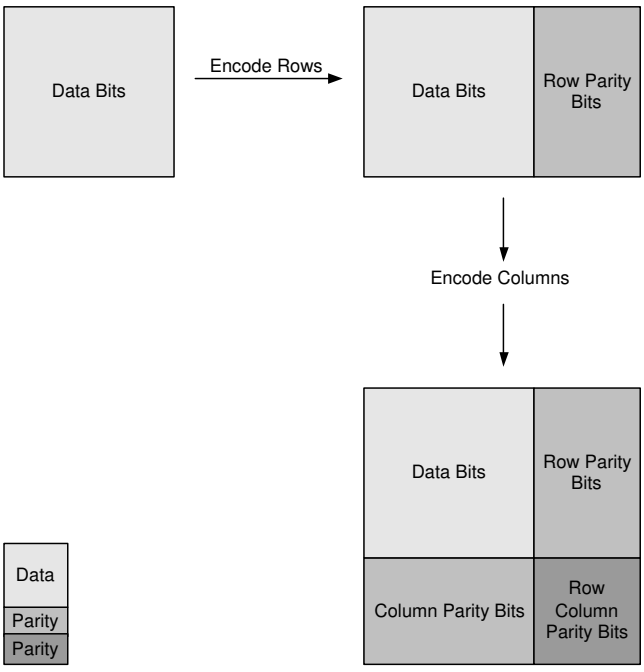

Fig. 1. Regular product code encoding procedure, where a block code is used to encode rows and columns.

In our case we use convolutional codes instead of linear block codes to encode rows and columns. This is illustrated in Fig. 2. When compared to Fig. 1, it is obvious that data and parity bits are mixed uniformly.

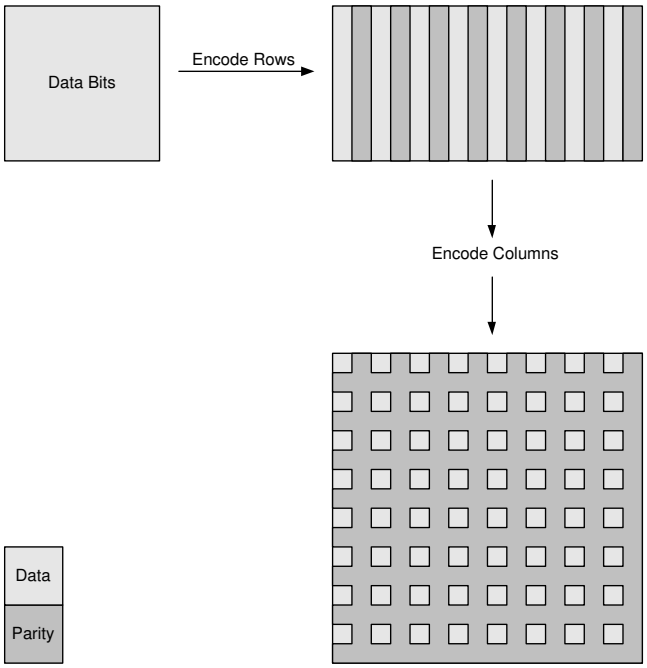

Fig. 2. CPC encoding procedure without an interleaver.

Encoding is performed by using a matrix which determines how each encoder works. The data to be sent is put into a matrix. Each row of the matrix is encoded using a convolutional code. We use the same recursive systematic convolutional code (RSC) to encode each row, although different convolutional codes can be used for this purpose. Once each row is encoded, the matrix is sent, if desired, to an interleaver. Our data matrix dimension is $k \times k$ and the encoded data matrix dimension is nxn, i.e., our code is an $(n \times n, k \times k)$ code. The interleaved matrix is coded column-wise. In our simulation we used the rate $1/2$ recursive systematic convolutional code with the matrix generator $(1, 5/7)_{octal}$ to encode each row and column. Hence, the overall code rate is $1/4$. The general encoding procedure, which includes any type of interleaver, is illustrated in Fig. 3.

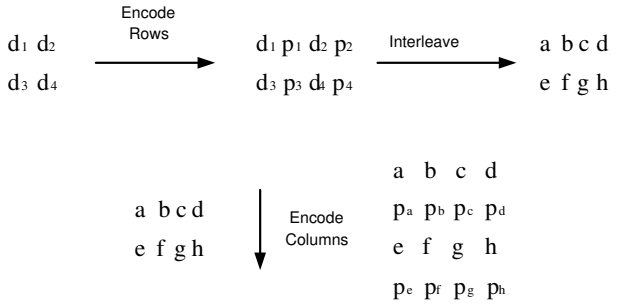

Fig. 3. Convolutional product code encoder with any type of interleaver (d denotes data bits and p denotes parity bits).

2. CPC Decoder

Convolutional product coded data is multiplexed to a single stream and binary phase shift key (BPSK) modulated. The BPSK modulated signal is passed through an additive white Gaussian noise (AWGN) channel with double-sided noise power spectral density $\frac{N_0}{2}$, i.e. noise variance is $\sigma^2 = \frac{N_0}{2}$. We used the log-MAP soft decoding algorithm [16]-[17] to iteratively decode the convolutional product code. Each column is independently decoded one by one since columns were encoded last. The extrinsic information obtained from the columns is passed to the row decoder after being de-interleaved. Then row decoding proceeds; rows are decoded one by one and interleaved extrinsic information is passed to the column decoder. The CPC decoding procedure is depicted in Fig. 4. This procedure is repeated for a sufficiently number of times. The decoding structure employed in this study is the same as that of serially concatenated codes in Fig. 5 [17]. For frames of equal length, an SCCC decoder uses two log-MAP decoders and performs quite well at low rates. CPC decoders can utilize many log-MAP decoders in parallel, thus showing smaller decoding delays. Therefore, we will compare the proposed CPC structure to that of SCCC.

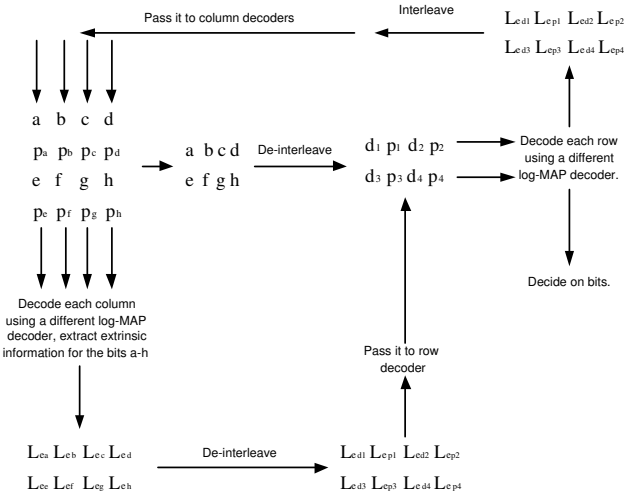

Fig. 4. Decoding operation of the convolutional product code.

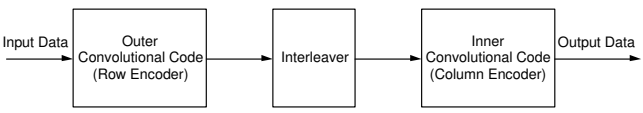

Fig. 5. SCCC Encoding Operation.

3. Puncturing

Puncturing is a widely used tool to increase the code rate of convolutional codes [18]. The puncturing operation increases the rate of a code, but decreases the free distance. This results in a worse error rate performance compared to the non-punctured case. We used the puncturing matrix

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

to puncture the convolutional component codes. We studied two cases. In the first case, puncturing is applied only to the column encoders, resulting in a code rate of $2/3$ each. The overall code rate becomes $(1/2) \times (2/3) = 1/3$. When trellis termination is used for rows and columns, a convolutional code with a slightly smaller overall code rate ($\leq 1/3$) is produced. In the other case, we apply puncturing to each row and column encoder, resulting in an increased code rate of approximately $(2/3) \times (2/3) = 4/9$. Simulation results for punctured convolutional product codes (PCPCs) will be presented in Section V.

III. CPC MINIMUM DISTANCE AND ITS ASYMPTOTIC PERFORMANCE

The Hamming weight of a binary codeword is defined as the number of '1's available in the codeword [13]. The minimum distance of a linear code is the minimum Hamming weight of all the codewords. The minimum distance plays an important role in the code performance. As it gets larger, code performance becomes better, especially at high SNR values [13]. We assume that d_{free} is the free distance of the component convolutional codes used in CPCs with trellis termination. We will investigate the minimum distance of the CPCs according to the usage of the interleavers.

1. No Interleaving

After the first stage of the CPC encoding operation (row encoding), it is obvious that one of the rows of the row-encoded matrix should contain at least d_{free} number of '1's. This means that there are d_{free} columns containing at least a single '1' in row-encoded matrix. When columns are encoded, there exist at least d_{free} number of columns each containing at least d_{free} '1's. Hence, in total there are at least d_{free}^2 '1's in the coded matrix [6]. This is the d_{min} distance of the CPC whose component convolutional codes have trellis termination constraint. In Figs. 6 and 7, this concept is explained for $(1, 5/7)_{octal}$ component convolutional codes whose free distance is 5. In summary, if no interleaver is used the CPC minimum distance is d_{free}^2.

2. Column S-random Interleaver

Both to preserve the d^2_{free} minimum distance of the CPC, and to get benefit from the interleaving gain, after row encoding we used S-random interleavers for each column, i.e., each column is interleaved but different column elements are not mixed. In this way we guarantee that d_{free} number of columns contain a single '1' before column encoding operation. We call this type of interleaving column S-random interleaving to distinguish it from regular S-random interleaving. A helical interleaver [19] also does not mix the different column elements. A helical interleaver and a combination of helical and column S-random interleavers will also be considered.

3. Full S-random Interleaver

If an S-random interleaver is used for all the elements of matrix after row encoding, the number of columns that contain a single '1' is not necessarily equal to $d_{free} = 5$. This leads to the fact that CPC minimum distance is not necessarily equal to d^2_{free} anymore. In fact, after interleaving operation all the '1's may appear in a single column. This means that CPC minimum distance is lower bounded by d_{free}. We call this type of interleaving full S-random interleaving. In Fig. 7 the effect of the full S-random interleaver is illustrated. It is seen from the Fig. 7 that when the row encoded matrix is S-random interleaved all the '1's appearing in a row may go to a single column. This verifies that CPC minimum distance is lower bounded by d_{free}.

4. Punctured CPCs

The puncturing operation decreases the free distance of convolutional codes. In our case, we puncture the $(1, 5/7)_{octal}$ component convolutional code which has $d_{free} = 5$, i.e., an input sequence '0111' produces minimum Hamming weight codeword '00111011'. When the puncturing matrix is applied its, free distance decreases to $d^{'}_{free} = 3$, i.e., deleting every second parity bit in a periodic manner '001x101x' is obtained from minimum Hamming weight codeword. Hence, the CPCs constructed using punctured component convolutional codes have smaller minimum distance. In fact, the minimum distance is equals $d^2_{free} = 9$, if no interleaving operation is performed or column S-random interleaver is used.

5. Asymptotic Performance

If row and column convolutional codes are trellis terminated, the row and column convolutional codes can be considered as block codes. Asymptotic performance studies

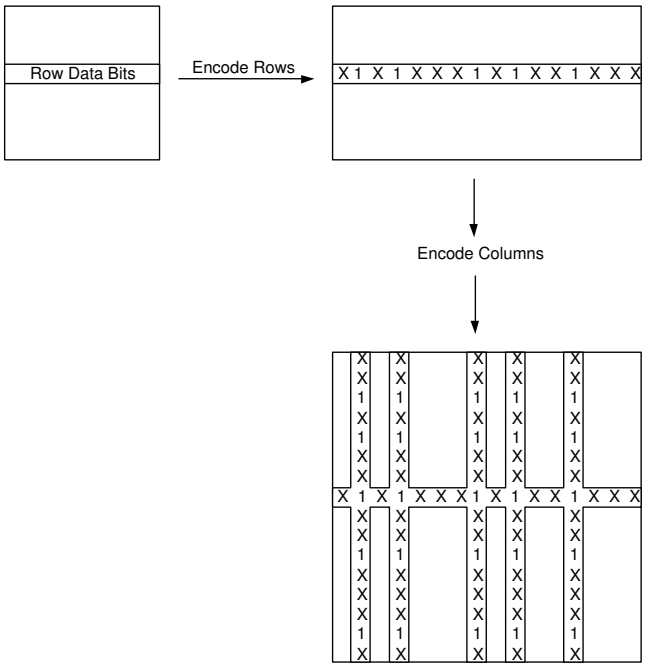

Fig. 6. If the columns elements are not mixed d^2_{free} is preserved.

made for block product codes are also valid for convolutional product code. BER probability of the CPCs can be approximated using the formula,

$$P_b \simeq \frac{N^2_{c,d_{free}} w^2_{c,d_{free}}}{k^2} Q\left(\sqrt{d^2_{free}\frac{2E_s}{N_0}}\right), E_s = r^2 E_b, \tag{1}$$

where d_{free} is the free distance of the convolutional code used to construct the convolutional product code. $N_{c,d_{free}}$ is the number of convolutional codewords with free distance d_{free}, k^2 is the length of the data information frame, and $w_{c,d_{free}}$ is the average Hamming weight of the information words that produces convolutional codewords with Hamming weight d_{free}. r is the rate of the component convolutional codes and is equal to $1/2$ or $2/3$ in our case. Q is the error function given as.

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{\frac{-t^2}{2}} dt \tag{2}$$

The BER approximation in (1) is valid if no interleaver is used during the CPC encoding operation. If an interleaver is used it does not hold anymore.

IV. PRACTICAL IMPLEMENTATION ADVANTAGES

The implementation advantage of CPC will be discussed herein with the parameters used in this study. Trellis

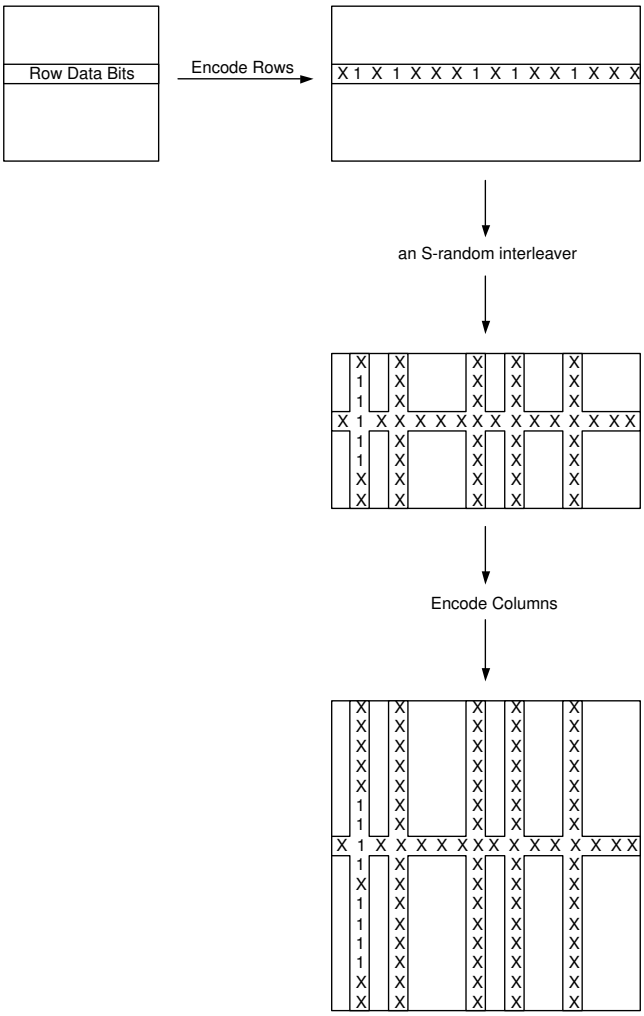

Fig. 7. d^2_{free} is not preserved if S-random interleaver is used ('x' stands for a single or a group of 0's).

termination will be neglected in calculation and will not alter the results significantly. In SCCC, for a given transmit data vector of length L, two log-MAP decoders are needed. The first decoder has a complexity of order $O(2L)$ and a time delay of $O(2L)$. The second decoder has a shorter input, thus it has a complexity of $O(L)$ and a time delay of $O(L)$. In total, the complexity is of $O(3L)$ and the time delay is of $O(3L)$. In CPC columns are decoded first. The use of separate log-MAP decoders for each row and column makes parallel processing operations possible. Each column decoder has complexity of $O(\sqrt{L})$ and time delay of $O(\sqrt{L})$. Since these decoders are run in parallel the total column decoding complexity is of $O(2L)$ but the time delay is of $O(\sqrt{L})$. Similarly, row decoding has a total complexity of $O(L)$ and time delay of $O(\sqrt{L})$. Hence, although both complexities are the same, time delays differ very much and brings about a $O(\sqrt{L})$ times increase in decoding rate. Hence, the main advantage of CPCs lies on its suitability for parallel decoding procedure. Although there are some proposed methods for the parallel decoding of SCCCs and PCCCs, these methods usually propose extra algorithms to solve problems met in parallel processing. (Such algorithms not only bring extra complexity to the decoding operation [20]-[21], but also may suffer from performance loss). This situation is totally remedied with the proposed CPCs.

V. SIMULATION RESULTS

We used a recursive convolutional code in CPC. Non-recursive convolutional codes were also tried and it was seen that performance is not as good as the product code with recursive convolutional codes. The convolutional product code was constructed using a rate $(1/2)$ systematic recursive convolutional code with component codes $(1, 5/7)_{octal}$ whose free distance is $d_{free} = 5$. In order to have sufficient statistical significance, we generated one million frames for each experiment, where each frame consists of 1024 bits, i.e., data matrix dimension is 32×32. We used 12 iterations to decode the CPCs. After row encoding operation, interleaving procedure is carried out. The type of and usage of the interleaver is very critical on the performance of the CPCs. We will separately investigate the effects of each case. Trellis termination and puncturing effects will be investigated separately. The signal-to-noise ratio values given in the figures are normalized with the proper code rates for all scenarios with trellis termination taken into account.

1. Interleaving Effects

1) No Interleaver: In this case, no interleaving operation is performed after row encoding. Trellis termination bits are added both to rows and columns. The minimum distance of the CPC is $d_{min} = d^2_{free} = 25$. Trellis termination bits are necessary to guarantee $d_{min} = d^2_{free}$, otherwise d_{min} is not equal to d^2_{free} anymore. The performance graph of this code is shown in Fig. 8. It is seen from the graph that the performance of this CPC is not good for low SNR values, although its minimum distance is large. As well known, minimum distance dominates the performance of the code at high SNR values.

2) Full S-random Interleaver: After the row encoding operation, an S-random interleaver (S=18) is used. We also simulated a serially concatenated convolutional code to compare against CPC due to similarity of the code structure and good performance at small rates. The performance graph is

seen in Fig. 8. As seen from the performance curve, the performance is very good compared to the cases where interleavers different than S-random are used. Due to the S-random interleaver used after row encoding, the minimum distance of the CPC is not necessarily equal to d^2_{free}. CPC with a full S-random interleaver shows the best performance at low rates due to the large interleaver gain.

3) Column S-random Interleaver: To obtain both better performance than the no interleaver case and to preserve the $d_{min} = d^2_{free}$ of CPC, we applied an S-random interleaver (S=3) to each column separately. We called such interleaving as column S-random interleaving. Different column elements are not mixed. From Fig. 8 it is seen that the performance is better compared to the CPC in which no interleaver is used. Its performance is worse then CPC where a full S-random interleaver is used after row encoding. The usage of the helical interleaver also guarantees that minimum distance of CPC equals d^2_{free}. We also investigated the case that an helical interleaver is followed by a column S-random interleaver. It is seen that such an interleaver results in slightly better performance then the one where only column S-random interleaver is used during the encoding procedure.

2. Trellis Termination Effects

We simulated three trellis termination cases where trellis termination bits are added to the rows only (CPC R-T), to both rows and columns (CPC TT), and neither to rows nor to columns (CPC No TT). Although addition of trellis termination bits decreases the code rate, they are critical for good performance of the convolutional product code as seen in Fig. 9. Addition of trellis termination bits in turbo or serially concatenated code shows negligible improvement on the code performance [22]. Without trellis termination, the performance of the CPC degrades drastically. The performance graphs are seen in Fig. 9. When only rows are trellis terminated, convolutional product code has better performance at very low E_b/N_0 levels. However the BER slope decreases at higher E_b/N_0 levels when compared to the case where both rows and columns are trellis terminated. We see that CPC R-T is better than the SCCC and CPC TT at low E_b/N_0 regions. Though quite close up to BER 10^{-7}, SCCC seems to have an error curve of higher slope compared to CPC TT at higher E_b/N_0 values.

3. Puncturing Effects

Puncturing is first applied only to rows, resulting in a rate $2/3$ CPC. From Fig. 10, it is seen that the performance of the CPC with a full S-random interleaver is good after being punctured. When the puncturing process is applied to both rows and columns, it results in a CPC of rate approximately $4/9$. From Fig. 10, it is seen that the performance becomes very bad for CPC with a full S-random interleaver. Recall that d_{min} is not necessarily lower bounded by d^2_{free} when an S-random interleaver is used. Thus, the particular interleaver we used resulted in a low d_{min}. When $d_{min} \geq d^2_{free}$ is ensured by column S-random interleaving, performance is enhanced significantly.

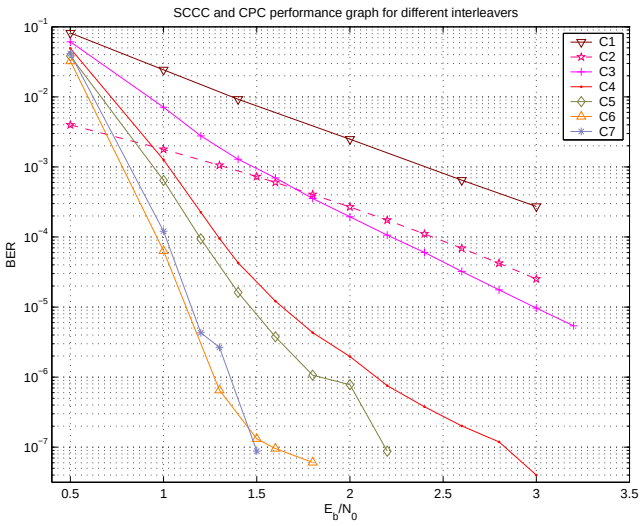

Fig. 8. SCC and CPC performance graph for different interleavers. Iteration number=12. Frame Length 1024. The graph is explained below.

C1: No interleaver is used (Rate $\approx 1/4$)
C2: Theoretical bound (Rate $\approx 1/4$)
C3: Helical interleaver is used (Rate $\approx 1/4$)
C4: Each column is S-random interleaved (Column S-random) (Rate $\approx 1/4$)
C5: Helical + column S-random interleaver is used (Rate $\approx 1/4$)
C6: Full S-random interleaver is used (Rate $\approx 1/4$)
C7: SCCC with S-random interleaver (Rate $\approx 1/4$)

VI. CONCLUSION

In this article, we study a new class of product codes based on convolutional codes. This type of product code has component codes with a time invariant trellis structure, as opposed to product codes constructed with linear block codes (Hamming, BCH, Reed Solomon est.). Hence, CPC may be more favourable for implementation than linear block product codes. When compared to serially concatenated convolutional codes, it exhibits comparable BER levels of practical interest.

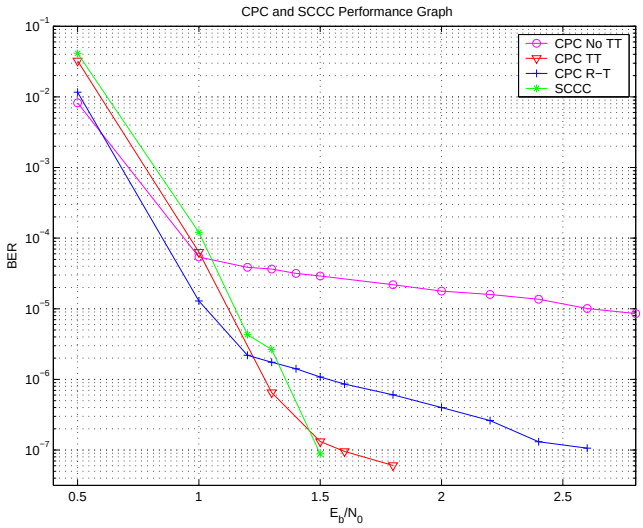

Fig. 9. CPC and SCCC performance graph. CPC with no trellis termination (CPC No TT). CPC when rows are trellis terminated (CPC R-T). CPC when rows and columns are trellis terminated (CPC TT). Frame Length=1024, Iteration number=12.

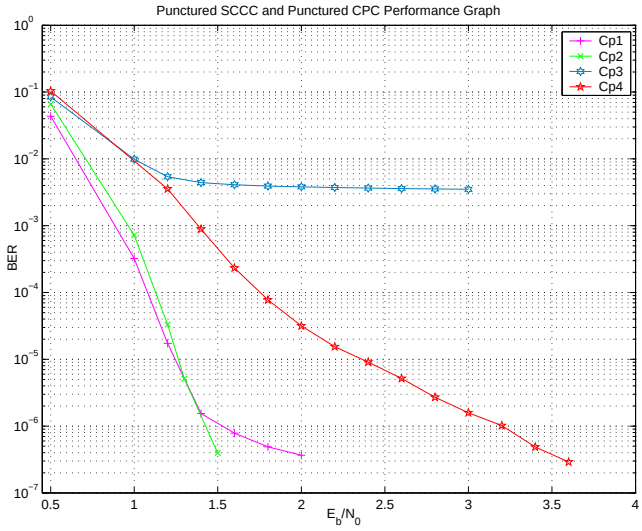

Fig. 10. Punctured CPC and Punctured SCCC performance graph. CPC rows and columns are trellis terminated. Frame Length=1024, Iteration Number=12.

Cp1: SCCC with S-random interleaver (Rate $\approx 1/3$)
Cp2: CPC with Full S-random interleaver (Rate $\approx 1/3$)
Cp3: CPC with Full S-random interleaver (Rate $\approx 4/9$)
Cp4: Each column is S-random interleaved (Column S-random) (Rate $\approx 4/9$)

We investigated the effects of different interleavers on the performance of CPCs. It was seen that CPCs are outperformed by other codes unless good interleavers are used. We proposed interleaving methods to preserve the greatest minimum distance of CPCs. It is seen that the performance of a CPC is best at low rates when a full S-random interleaver is used. Column S-random interleavers are much better for punctured CPCs.

Currently we investigate the effects of various interleavers and incorporation of trellis coded modulation in row and column encoding. Since CPC employ matrices in encoding, it can be easily extended to multi-carrier modulation where the vertical dimension can correspond to the sub-carriers. The approach presented here can be successfully extended to space-time trellis coding. So, our future studies will also include a joint structure for CPCs and MIMO space-time-frequency codes.

ACKNOWLEDGMENT

The authors would like to thank anonymous reviewers for their valuable comments and helpful corrections.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes, "in *Proc. ICC'93* (Geneva, Switzerland, May 1993), pp. 1064-1070.

[2] S. Benedetto, L. Gaggero, R. Garello, and G. Montorsi, "On the design of binary serially concatenated convolutional codes," in *Proc. VIII Communication Theory Mini-Conf.* CTMC, Vancouver, BC, Canada, June 1999, pp. 32-36.

[3] R. G. Gallager, "Low Density Parity Check Codes, "*IRE Trans. Inform. Theory,* IT-8:21-28, Jan., 1962.

[4] P. Elias, "Error free decoding, "*IRE Trans. Inform. Theory,* vol. IT-4, pp. 29-37, Sept., 1954.

[5] E. Hewitt, "Turbo Product Codes for LMDS, "*IEEE Radio and Wireless Conference,* August 1998.

[6] Nam Yul Yu, Young Kim, Pil Joong Lee, "Iterative decoding of product codes composed of extended Hamming Codes, "*Fifth IEEE Symposium on Computers and Communications (ISCC 2000),* July 04 - 06, 2000 Antibes, France.

[7] R. M. Pyndiah, "Near-optimum decoding of product codes: block turbo codes, "*IEEE Trans. Commun.*, vol. 46, No. 8, Aug, 1998.

[8] T. Shohon, Y. Soutome, H. Ogiwara, ”Simple computation method of soft value for iterative decoding of product code composed of linear block code, *”IEIC Trans. Fundamentals*, Vol. E82-A, No. 10 October 1999.

[9] Omar Aitsab, Ramesh Pyndiah, ”Performance of Reed Solomon block turbo codes, ”in *Proc. IEEE GLOBECOM’96 Conf.*, vol. 1/3, London, U.K., Nov. 1996, pp. 121-125.

[10] David Rankin, T. Aaron Gulliver, ” Single parity check product codes, *”IEEE Trans. Commun.*, vol. 49, no. 8, August 2001 pp. 1354-1362

[11] D.M. Rankin and T.A. Gulliver, ”Randomly Interleaved Single Parity Check Product Codes, *”Proc. IEEE Int. Symp. on Inform. Theory*, p. 88, June 2000.

[12] A. Goalic, R. Pyndiah, ”Real time turbo decoding of product codes on a digital signal processor, *”Int. Symposium on turbo codes and related topics*, Brest, Sept. 1997.

[13] Shun Lin, Daniel J. Costello, Jr., Error Control Coding. Prentice Hall, 2004.

[14] Sanzi, F., ten Brink, S.,”Iterative channel estimation and decoding with product codes in multicarrier systems, *”IEEE VTS Fall VTC2000 52nd Vehicular Technology Conference* Boston, Ma USA September 24-28, 2000.

[15] V. Tarokh V, N. Seshadri, and A. R. Calderbank, ”Space-time Codes for High Data Rate Wireless Communication: Performance Criterion and Code Construction,” *IEEE Trans. Inform. Theory*, Vol. 44, No. 2, pp. 744-765, March 1998.

[16] J. Hagenauer, E. Offer and L. Papke, ”Iterative decoding of binary block and convolutional codes,” *IEEE Trans. Inform. Theory*, vol. 42, No. 2, Mar., 1996.

[17] S. Benedetto, D. Divsalar, G. Montorsi and F. Pollara, ”Serially concatenation of interleaved codes: Design and performance analysis,” *IEEE Trans. Inform. Theory*, Theory, vol. 44, pp. 909-926, May, 1998.

[18] J. Hagenauer, ”Rate-Compatible Punctured Convolutional Codes (RCPC Codes) and their Applications,” *IEEE Trans. Inform. Theory*, Vol. 44, No. 3, pp. 909-926, May 1998.

[19] Branka Vucetic, Jinhong Yuan, ”Turbo Codes Principles and Applications”, Kluwer Academic Publishers.

[20] A. Tarable, S. Benedetto, G. Montorsi, ”Mapping interleaving Laws to Parallel Turbo and LDPC decoder Architectures”, *IEEE Transactions on Information Theory*, Vol. 50, No. 9, Sept. 2004.

[21] Seokhyun Yoon, Yeheskel Bar-Ness, ”A Parallel MAP Algorithm for Low Latency Turbo Decoding”, *IEEE Communication Letters*, Vol. 6, No. 7, July 2002.

[22] P. Robertson, ”Illuminating the structure of parallel concatenated recursive (TURBO) codes, ” in *Proc. GLOBECOM’94*, San Francisco, CA, Nov. 1994, pp. 1298-1303.