Protograph E^2RC Codes

Cuizhu Shi and Aditya Ramamoorthy
Department of Electrical and Computer Engineering
Iowa State University
Ames, Iowa 50011
Email: {cshi, adityar}@iastate.edu

***Abstract*—We propose a construction of rate-compatible punctured codes based on protographs that have a special E^2RC structure in their parity part (E^2RC codes were introduced in Kim, Ramamoorthy and McLaughlin '06) . The protograph representation of these codes facilitates their asymptotic performance analysis and allows the implementation of high speed decoders. The construction process starts with a good high rate protograph. The protographs of lower rate codes are derived from the higher rate protographs via the process of check-splitting. The check-splitting is done in a specific manner so that the parity nodes in the protograph have the special E^2RC structure. We also present additional design rules that ensure that the gap to capacity remains low across the range of rates. Using our approach we exhibit protographs that have a gap of at most 0.27 dB to capacity across the range of rates $1/2$ to $8/9$. These conclusions are supported by our simulation results. Our work, therefore presents a systematic method for the design of E^2RC-like codes.**

I. Introduction

LDPC codes introduced in [1] have near-capacity performance on a large variety of channels and low decoding complexity, and have been proposed in a number of new applications and standards. There have been numerous constructions of LDPC codes proposed in the literature ranging from random choice to algebraic constructions [2] [3] [4] [5] [6]. Most LDPC codes used in industry need to have some structure that allows parallelizable decoding. Moreover the amount of storage required to store the description of the parity-check matrix needs to be small, i.e., storing completely random permutations is not feasible due to implementation issues. The protograph based LDPC codes introduced in [4] address this issue in part. The main idea here is to start with a small mini-graph (called a protograph) and construct the LDPC code by replacing each edge in the protograph by a random permutation of a fixed size. Protograph codes have the advantage that the asymptotic threshold of the code can be found by performing density evolution [2] [7] [3] on the protograph. Moreover, if instead of a random permutation we choose a random circulant permutation that can be specified by a circular shift of the identity matrix then the storage requirement can be reduced tremendously and a fast parallelizable decoder can be implemented in hardware.

Another desirable feature in practice especially for wireless channels is rate-compatibility. Rate-compatible puncturing was introduced in [8] as a technique to obtain a family of codes of varying rates while retaining the same encoder-decoder pair. Rate-compatible punctured codes are a practical low-complexity solution that are useful in hybrid-ARQ protocols and situations where the channel quality varies over time. The mother code (which is systematic) in these systems corresponds to the lowest code rate. Higher code rates can be obtained by only transmitting a subset of the parity bits. The parity bits that are not transmitted are said to be punctured. The parity bits of higher-rate codes are chosen to be a subset of the parity bits of the lower rate codes.

A number of papers have investigated the construction of rate-compatible punctured LDPC codes [9] [10] [11] [12] [13] [14] [15]. The main challenge here is the design of a mother code and the puncturing pattern such that the BER/FER of the codes of all rates is low. There are two main approaches that address the problem of rate compatibility.

- *Optimizing puncturing patterns or distributions.* In [10], the authors found the optimal degree distributions for puncturing using density evolution analysis. In [11] [12] [14], the authors proposed algorithms for finding good puncturing patterns for given mother code.
- *Design of a good mother code and the puncturing pattern.* Here the attempt is to design LDPC codes with a specific structure that allows good performance across a range of rates. Some recent works include [9] [13] [15]. These approaches have been guided in part by the criteria used in design good puncturing patterns as well.

In this paper, we consider the construction of high-performance rate-compatible LDPC codes based on protographs. As mentioned before rate-compatibility is a desirable goal, while the protograph structure allows asymptotic analysis and efficient implementation. Our protograph construction technique is inspired by the structure of the E^2RC codes in [9]. We start from a high-rate protograph and present a step-by-step check-splitting technique guided by density evolution analysis to generate the protograph of the low-rate mother code. The check-splitting is done in a specially designed order so that the parity part of the low-rate mother code exhibits an E^2RC-like structure which ensures good performance under puncturing. This yields a systematic technique for the design of E^2RC-like codes, as it becomes possible to perform asymptotic threshold analysis of the mother code and the punctured codes, while ensuring that practical implementation is possible.

We note that the idea of check-splitting to obtain new codes from existing codes has been around in the work of [16]

[17] [18]. The contribution of this paper is the usage of this technique in constructing protographs such that their parity part retain the E^2RC structure that ensures good performance across rates. Moreover, the design space for the remaining part of the protograph can be potentially very large. We provide some criteria for pruning the search space while maintaining a low threshold gap from capacity across all rates.

The paper is organized as follows. In section II we overview the related work and basic notations that we use in this paper. Section III presents the proposed construction of the protograph E^2RC codes. The material in section IV contains the simulation results with comparisons with previous work and section V outlines the conclusions.

II. Preliminaries and Related Work

A. LDPC Codes and Protograph Based LDPC Codes

An LDPC code can be defined by its parity-check matrix or equivalently by a bipartite graph representation, where each edge in the graph connects a variable node (representing the bits) on one side with a check node (representing the parity-check equations) on the other side. Throughout this paper for the bipartite graph representation, we follow the convention that a blank circle represents an unpunctured variable node that participates in the transmission and a filled circle represents a punctured variable node that does not participate in transmission. A check node is represented by a blank circle with a cross sign in it.

LDPC codes based on protographs were introduced in [4]. The main idea here is to start with a small mini-graph (called a protograph) and construct the LDPC code by replacing each edge in the protograph by a random permutation of a fixed size (see Figure 1 for an example). Protograph based codes can be considered as a subclass of the multi-edge type LDPC codes [3]. In the protograph representation, each edge represents one edge type, and parallel edges are allowed. Density evolution can be performed on protographs to determine their asymptotic performance.

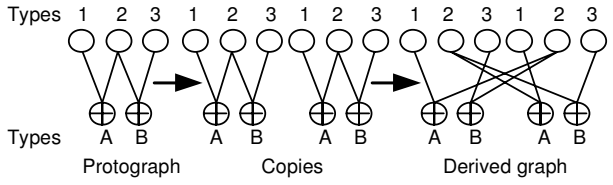

Fig. 1. Copy-and-permute in generating larger graph from protograph

B. Efficiently Encodable Rate-Compatible LDPC Codes

A significant amount of research work has dealt with the construction and analysis of punctured LDPC codes [9] [10] [11] [12] [13] [14] [15]. In this subsection we briefly explain the construction introduced by Kim, Ramamoorthy and Mclaughlin [9] as our work is inspired by it. Let the parity-check matrix of a systematic LDPC code be denoted $H = [H_1|H_2]$ where H_1 denotes the sub-matrix corresponding to the information bits and H_2 denotes the sub-matrix corresponding to the parity bits. We say that a parity node in H_2 is k-step recoverable (or $k - SR$) if it can be recovered in exactly k iterations of iterative decoding assuming that all the parity bits are punctured and all the information bits are unpunctured (Figure 2 shows an example). Intuitively, a large number of low-SR nodes tend to reduce the required number of decoding iterations in the high SNR regime and result in good puncturing performance.

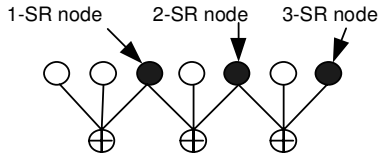

Fig. 2. The figure shows an example of 1-SR, 2-SR and 3-SR nodes.

In [9] the submatrix H_2 consists of exclusively degree-2 and degree-1 nodes. Moreover half the nodes in H_2 are $1 - SR$, one-fourth are $2 - SR$ and so on[1]. The special structure of H_2 for E^2RC codes results in good puncturing performance with a simple puncturing pattern. The puncturing pattern is such that $1-SR$ nodes should be punctured first and then the $2-SR$ nodes and so on depending upon the rate requirement.

In this paper we design a class of protographs that is inspired by the approach in [9]. These protographs have thresholds 0.3 dB away from the Shannon limit across a wide range of rates. Moreover they have excellent finite length performance as well. In estimating the asymptotic performance of the protograph based LDPC codes we use the reciprocal channel approximation [19] of density evolution that has been found to be fast and very accurate in practice.

Throughout this paper, we consider the case when the codes are transmitted over additive white gaussian noise (AWGN) channel with binary phase-shift keying (BPSK) modulation and belief propagation decoding is used.

III. Protograph E^2RC codes construction

In this section, we outline our technique for the construction of the protograph E^2RC codes. Given the desirable code rate range $R_{min} \leq R \leq R_{max}$ for the code family, The construction algorithm goes as follows.

1) *Find a good high-rate protograph.*
   Using density evolution we first identify a high-rate protograph with a low threshold.
2) *Use check-splitting to obtain lower-rate protographs with low thresholds.*
   We split check nodes in the high-rate protograph in a systematic manner to obtain good low-rate protographs. The check-splitting operation is explained in detail in subsections III-B and III-C.
3) *Construct the LDPC code by replacing edges with carefully chosen permutations.*
   With the protograph constructed from above steps, a larger graph defining the LDPC code is constructed by using techniques such as those in [20] [21] [22].

The construction procedure ensures that the parity part of the code maintains a E^2RC like structure during the process.

[1]This is strictly true if the number of parity nodes is a power of two and approximately true otherwsie

One minor difference is that we do not have any degree-1 check node in the protograph. In the sequel we illustrate the construction process by means of an example.

A. Starting Protograph

A protograph of size $M_0 \times N_0$ (M_0 - number of constraint nodes, N_0 - number of variable nodes) with low threshold serves as a starting point; we call it the starting protograph. Given the desirable code rate range $R_{min} \leq R \leq R_{max}$, we decide the size of the mother code protograph $M \times N$ and the size of the starting protograph $M_0 \times N_0$, such that $\frac{N-M}{N} \leq R_{min}$, $\frac{N_0-M_0}{N_0} \geq R_{max}$ and $N - M = N_0 - M_0$. These conditions guarantee that the desirable code rate range is achievable by the construction. In addition, these parameters should be kept relatively small, say less than 50, to keep the construction complexity manageable. The construction will give a family of protographs with code rates ranging from $\frac{N-M}{N}$ to $\frac{N_0-M_0}{N_0}$. We impose the constraint that the degree of all variable nodes in the starting protograph is at least three. This is because the presence of degree two nodes causes a relatively high error floor.

In our example, $R_{min} = 0.5$, $R_{max} = 0.85$, and the size parameters are decided as $M_0 = 1, N_0 = 9$ and $M = 8, N = 16$. The protograph shown in Figure 3 is used as our starting protograph. It consists of one check node and nine variable nodes of degree $\{24,8,3,3,3,3,3,3,3\}$ respectively. It has a threshold of 3.27 dB which is 0.24 dB away from the theoretical limit of 3.03 dB.

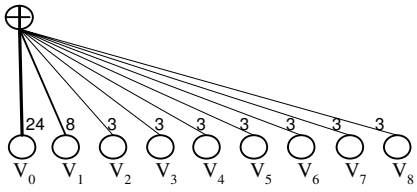

Fig. 3. Example of a starting protograph. The number next to an edge denotes the number of parallel edges between the variable and the check node.

B. Check-Splitting

Consider a check node c of degree d in the protograph. The operation of check-splitting proceeds as follows. We split c into two check nodes c_1 and c_2, such that the degree of c_1 is d_1 and the degree of c_2 is d_2 and $d = d_1 + d_2$. Next we introduce a punctured variable node v' and introduce edges $c_1 - v'$ and $c_2 - v'$, so that the degree of v' is two. This is shown pictorially in Fig. 4.

Check-splitting was used in [16] to construct protograph LDPC codes with linear minimum distance. In [17], a rigorous proof was given in lemma 3 and theorem 5 about the asymptotic decoding performance equivalence between a non-punctured high-rate LDPC code and a punctured low-rate LDPC code whose parity-check matrix can be built from that of the high-rate LDPC code through check-splitting. This is confirmed by our density evolution analysis on the first two graphs in Figure 4, which give the same asymptotic iterative decoding threshold.

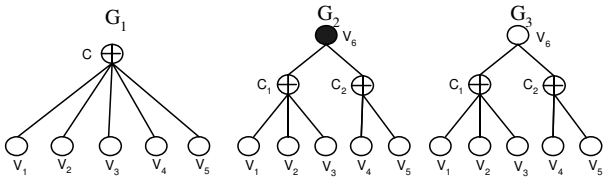

Fig. 4. Check c with degree-5 is divided into c_1 and c_2. The new check nodes c_1 and c_2 are connected by a new degree-2 variable node v_6.

In this paper we use check-splitting as follows. We start with the high-rate protograph (as explained above) and split its check nodes in a specific manner. To obtain lower rates, after splitting a given check node, we convert the newly introduced punctured node into a transmitted node (e.g. see G_3 in Fig. 4). By applying check-splitting to a protograph of higher rate codes repeatedly, we finally arrive at the protograph of low rate mother code. In fact, the protographs produced in the check-splitting process form a family of rate-compatible protographs if we consider the newly added degree-two variable nodes in check-splitting as parity nodes that are used to provide incremental redundancy. However we note that the check-splitting needs to be done carefully (as shown later), otherwise it may not be possible to have good performance across all rates.

C. Constructing Protographs with E^2RC Like Structure

In this subsection we present a specific check-splitting technique that ensures the parity part of the code has the E^2RC structure. In what follows we call the original variable nodes of degree at least three in the starting protograph, *old nodes* and the variable nodes of degree two introduced due to check-splitting, *new nodes*.

During the construction, each check node has some connections with the old nodes and some connections with the new nodes. Accordingly, for a given check node we define its old node degree to be the number of connections to the old nodes and its new node degree to be the number of connections to the new nodes. Consider a check node c_0. When we split c_0 into c_{01} and c_{02}, a decision needs to be made on how the connections of c_0 are divided between them. To obtain the E^2RC structure, c_{01} is allocated all of c_0's new node degree. The parity node that is introduced in the check-splitting process has one connection to both c_{01} and c_{02}. The old node degree also needs to be divided between c_{01} and c_{02} in a manner that ensures that the threshold of the new protograph is low. The division of the old node degree is discussed in more detail in the next subsection.

The construction proceeds in different stages. In each stage, we perform check-splitting on all check nodes in the current protograph. We now use the starting protograph in Figure 3 to demonstrate the process. Note that here we have $M_0 = 1$. We want to obtain a protograph with $M = 8$. Thus in this case we shall have $\log_2 8 = 3$ stages in the construction.

	old									new
	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	
c_0	24	8	3	3	3	3	3	3	3	-

Let M_{n_s} denote the number of check nodes in the current protograph at the beginning of stage n_s. In this stage, we perform check-splitting on each of the M_{n_s} check nodes. Each check-splitting operation on the current protograph generates a protograph of next lower code rate. So at stage n_s, a set of M_{n_s} protographs of decreasing rates are generated. The order in which the check nodes are chosen for splitting can affect the thresholds at those rates.

We now show the first and second splitting stages for protograph in Figure 3. In the first stage, check-splitting on the single check node generates a new protograph of code rate $\frac{8}{10}$.

	old									new
	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
c_{01}	12	4	2	1	2	1	2	1	2	1
c_{02}	12	4	1	2	1	2	1	2	1	1

In the second stage, there are two check nodes in the protograph for splitting. Density evolution analysis tells us that performing check-splitting on c_{01} first gives a protograph of code rate $\frac{8}{11}$ with a better decoding threshold than that of protograph generated by performing check-splitting on c_{02}. So in this stage, we first split c_{01} to generate a protograph of rate $\frac{8}{11}$ and then split c_{02} to generate a protograph of rate $\frac{8}{12}$. The corresponding protographs are shown below.

	old									new	
	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
c_{011}	6	2	1	1	1	0	1	1	1	1	1
c_{012}	6	2	1	0	1	1	1	0	1	0	1
c_{02}	12	4	1	2	1	2	1	2	1	1	0

	old									new		
	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
c_{011}	6	2	1	1	1	0	1	1	1	1	1	0
c_{012}	6	2	1	0	1	1	1	0	1	0	1	0
c_{021}	6	2	1	1	0	1	1	1	0	1	0	1
c_{022}	6	2	0	1	1	1	0	1	1	0	0	1

The third stage proceeds in a similar manner. Note that following this procedure we get node v_9 is 2-SR, while nodes v_{10} and v_{11} are 1-SR. More generally, we can show that when the algorithm finishes executing stage n_s, it adds M_{n_s} new parity nodes all of which are 1-SR. Consider parity nodes that existed at the beginning of stage n_s as k-SR nodes. We can also show that at the end of stage n_s, all those nodes become $(k+1)$-SR. The proof is skipped due to lack of space. It turns out that the construction ensures that half of the parity nodes in the final protograph are 1-SR, one-fourth are 2-SR and so on i.e. our construction technique results in protographs that have the E^2RC structure.

We puncture the parity nodes of the resultant protograph to obtain higher rates. The puncturing order is the inverse of the order in which the parity nodes are added during the construction i.e. a parity node that was added at the end will be punctured first, a parity node that was added at the penultimate stage will be punctured second and so on. This puncturing order ensures that the gap to capacity at the different rates is as predicted by the construction process.

D. Deciding Splitting Patterns

As pointed out before, in check-splitting, the old node degree of a given check node can be divided among the new check nodes in many ways. Now we explain how to decide proper splitting patterns to ensure good code performance. Let c_0 be the check node that needs to be split. Let s_0 be the vector of connections between c_0 and the old nodes e.g. $s_0 = [24\ 8\ 3\ 3\ 3\ 3\ 3\ 3\ 3]$. The splitting pattern refers to the set of vectors $s_{01} = [12\ 4\ 2\ 1\ 2\ 1\ 2\ 1\ 2]$ and $s_{02} = [12\ 4\ 1\ 2\ 1\ 2\ 1\ 2\ 1]$ that determine the connections between the nodes c_{01} and c_{02} and the old nodes. Thus $s_0 = s_{01} + s_{02}$.

Our search for good splitting patterns is guided by two main points.

a) *Trade-off between the performance of high-rate and low-rate protographs.* In our experiments, we have found that there exists a tradeoff between the performance of high-rate and low-rate protographs during the construction. For example, it is possible to obtain very low thresholds for the higher rate protographs, however this typically comes at the expense of higher thresholds for the low-rate protographs in a later stage of the construction.
b) *Uniform splitting patterns give good performance.* Note that considering all possible splitting patterns at all possible stages is essentially computationally infeasible since the number of possible splitting patterns grows exponentially. We have found that splitting patterns that split the connections roughly equally between the new check nodes have good performance across all code rates in the family. This reduces the search space a lot and it becomes possible to perform density evolution analysis (using the fast reciprocal channel approximation) to determine the thresholds.

In the tables shown below we present two of the protographs that we have constructed using the construction algorithm described above. Both protographs are constructed from starting protograph in Figure 3. At the bottom of each table we show the gap to Shannon limit for the protograph at different puncturing levels for rates $8/9 - 8/16$ from left to right. For protograph-1 , there is a good balance on the code performance at all code rates. For protograph-2, we chose the splitting patterns to lower the threshold for the high rate protographs, which resulted in somewhat worse performance at the low code rates.

Mother code Protograph-1															
v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}
3	1	1	0	1	0	0	1	0	1	1	0	1	0	0	0
3	1	0	1	0	0	1	0	1	0	0	0	1	0	0	0
3	1	0	0	1	0	1	0	0	0	1	0	0	0	1	0
3	1	1	0	0	1	0	0	1	0	0	0	0	0	1	0
3	1	0	1	0	0	1	0	0	1	0	1	0	1	0	0
3	1	1	0	0	1	0	1	0	0	0	0	0	1	0	0
3	1	0	0	1	0	0	1	0	0	0	1	0	0	0	1
3	1	0	1	0	1	0	0	1	0	0	0	0	0	0	1
Gap to Shannon limit in dB (rates 8/9, 8/10 - 8/16)															
0.24	0.25	0.22	0.21	0.24	0.26	0.27	0.26								

Mother code Protograph-2															
v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}
2	0	1	1	0	1	0	1	1	1	1	0	1	0	0	0
4	2	0	1	0	1	0	1	1	0	0	0	1	0	0	0
3	0	0	0	0	0	2	0	0	0	1	0	0	1	0	0
3	2	1	0	2	0	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1
3	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1
3	0	0	0	1	0	1	0	0	0	0	1	0	0	1	0
3	4	1	0	0	0	0	0	1	0	0	0	0	0	1	0
Gap to Shannon limit in dB (rates 8/9, 8/10 - 8/16)															
0.24	0.17	0.19	0.24	0.21	0.23	0.35	0.36								

E. Comparison with Previous Results

Here we give comparisons with some existing results in the literature. The asymptotic performance comparison between the protograph E^2RC code family represented by protograph-1 above and the AR4JA code family in [16] is given in Figure 5. Higher rate codes in the protograph E^2RC code family are obtained by puncturing parity nodes of the same type as v_{15}, v_{14},...v_9 in turn. We note that the thresholds of our protographs are closer to the Shannon limit than that of AR4JA codes for all the rates $1/2, 2/3$ and $4/5$. The average variable node degree of our codes is a little higher than that of AR4JA family. Note, however that the codes in [16] are not rate-compatible punctured codes.

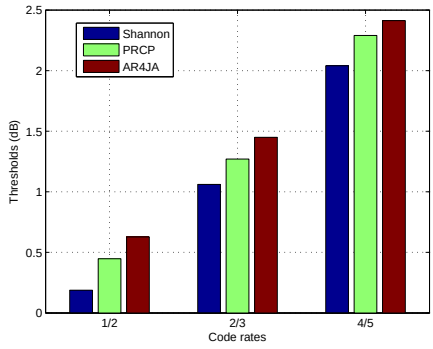

Fig. 5. Asymptotic performance comparison between protograph E^2RC codes and AR4JA codes.

IV. Simulations

The simulation results for the protograph E^2RC codes and the comparison with E^2RC codes [23] are presented in this section. Larger graphs defining protograph E^2RC codes with longer block length are constructed from protograph-1 above by using ACE algorithm to select circulants for each edge in the protograph. Mother codes of block length 1200 bits and 4096 bits are used. Puncturing to get higher rate codes in the family follows the order that parity nodes of the same type as v_{15} be punctured first, followed by parity nodes of the same type as v_{14}, v_{13}, v_{12}, v_{11}, v_{10} and v_9. The simulation results and comparison with E^2RC codes are shown in Figure 6 and Figure 7. From the simulation, we see that protograph E^2RC codes are almost as good as E^2RC codes for block length 1200 bits and when the block gets longer, say 4096 bits, protograph E^2RC codes get better than E^2RC codes especially considering performance in terms of frame error rate. For the block length 4096 codes, at code rate $R = 0.5$, the performance gain in terms of bit error rate at 10^{-4} is more than 0.2 dB and in terms of frame error rate at 10^{-2} is much more than 0.4 dB. The performance gain reduces when the code rate gets close to 1. The performance gaps of protograph E^2RC code family of block length 4096 bits to Shannon limits are measured at bit error rate 10^{-4} in simulation which show a uniform 1 dB gap over all code rates.

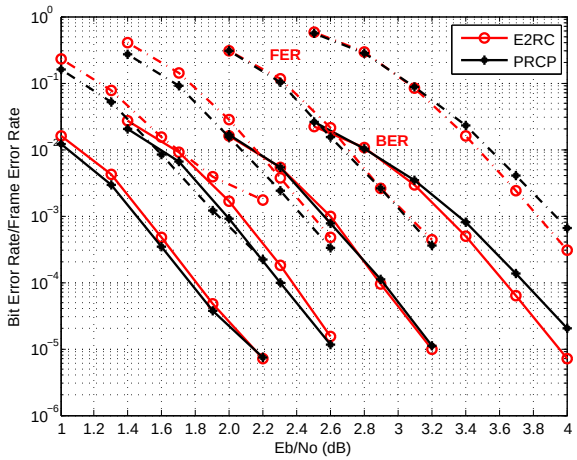

Fig. 6. Performance comparison between protograph E^2RC codes and E^2RC codes: block length 1200 bits; code rates are 0.5, 0.6, 0.7 and 0.8 from left to right. The solid lines represent the BER and the dotted lines represent FER.

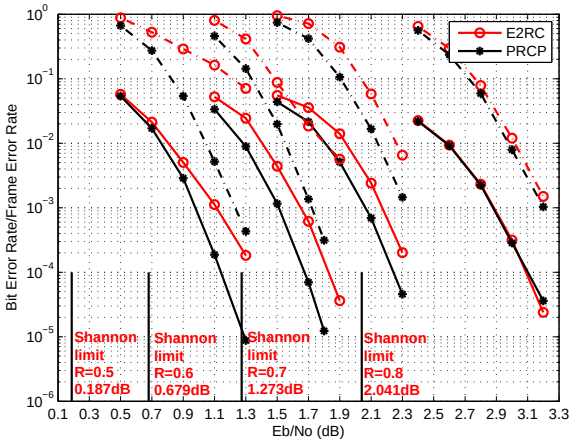

Fig. 7. Performance comparison between protograph E^2RC codes and E^2RC codes: block length 4096 bits; code rates are 0.5, 0.6, 0.7 and 0.8 from left to right. The solid lines represent the BER and the dotted lines represent FER. Shannon limits at the different rates are also indentified.

V. Conclusion

We propose a construction of rate-compatible codes based on protographs inspired by the E^2RC codes designed in [23]. Protograph E^2RC codes have protograph representations which facilitate their asymptotic performance analysis and allow the implementation of high speed decoders. The construction starts with a high rate protograph with low threshold. Protographs of lower rate codes are iteratively derived from the higher rate protographs via the process of check-splitting.

The check-splitting process is specially designed to ensure that the parity nodes in the protograph have the E^2RC structure. Furthermore the construction process guided by density evolution also produces protographs that have low thresholds at all rates. Thus, this paper introduces a systematic technique for the design of E^2RC-like codes. Both asymptotic performance analysis and simulation results demonstrate that the protograph E^2RC code family has good performance across all code rates.

REFERENCES

[1] R. G. Gallager, *Low Density Parity Check Codes*. MIT press, Cambridge, MA, 1963.

[2] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619–637, 2001.

[3] T. Richardson, "Multi-edge type ldpc codes," *presented at the Workshop honoring Prof. Bob McEliece on his 60th birthday, California Institute of Technology, Pasadena, California*, May. 2002.

[4] J. Thorpe, "Low density parity check (ldpc) codes constructed from protographs," *JPL INP Progress Report 42-154*, Aug., 2003.

[5] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Analysis of low density codes and improved designs using irregular graphs," *IEEE Trans. Inform. Theory*, vol. 47, pp. 585–598, 2001.

[6] Y. Kou, S. Lin, and M. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Transactions on Information Theory*, vol. 47, pp. 2711 –2736, Nov. 2001.

[7] T. Richardson and R. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599–618, 2001.

[8] J. Hagenauer, "Rate compatible punctured convolutional codes (rcpc codes) and their applications," *IEEE Trans. Commun.*, vol. 36, no. 4, p. 389C400, Apr. 1988.

[9] J. Kim, A. Ramamoorthy, and S. W. McLaughlin, "Design of efficiently-encodable rate-compatible irregular ldpc codes," in *IEEE International Conference on Communications*, 2006.

[10] J. Ha, J. Kim, and S. W. McLaughlin, "Rate-compatible puncturing of low-density parity-check codes," *IEEE Trans. Inform Theory*, vol. 50, no. 11, Nov. 2004.

[11] J. Ha, J. Kim, D. Klinc, and S. W. McLaughlin, "Rate-compatible punctured low-density parity-check codes with short block lengths," *IEEE Trans. Inform. Theory*, vol. 52, no. 2, Feb. 2006.

[12] J. Ha, J. Kim, and S. W. McLaughlin, "Puncturing for finite length low-density parity-check codes," in *Proc. Int. Symp. Inform. Theory, Chicago*, 2004.

[13] M. R. Yazdani and A. H. Banihashemi, "On construction of rate-compatible low-density parity-check codes," *IEEE Comm. Letters*, vol. 8, no. 3, pp. 159–161, Mar. 2004.

[14] G. Yue, X. Wang, and M. Madihian, "Design of rate-compatible irregular repeat accumulate codes," *IEEE Trans. Commun.*, vol. 55, no. 6, pp. 1153–1163, Jun. 2007.

[15] J. Li and K. R. Narayanan, "Rate-compatible low-density parity-check codes for capacity-approaching arq schemes in packet data communications," in *Proc. Int. Conf. Commun., Internet, and Inform. Tech. (CIIT), US Virgin Islands*, Nov. 2002.

[16] D. Divsalar, S. Dolinar, and C. Jones, "Construction of protograph ldpc codes with linear minimum distance," in *Proc. International Symposium on Information Theory*, Jul. 2006.

[17] H. Pishro-Nik and F. Fekri, "Results on punctured low-density parity-check codes and improved iterative decoding techniques," *IEEE Transactions on Information Theory*, vol. 53, no. 2, pp. 599 – 614, Feb., 2007.

[18] M. Good and F. R. Kschischang, "Incremental redundancy via check splitting," *IEEE 23rd Biennial Symposium on Communications*, 2006.

[19] S. Y. Chung, *On the Construction of Some Capacity-Approaching Coding Schemes*. Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge,Massachusetts, Sep., 2000.

[20] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth tanner graphs," *IEEE Global Telecommunications Conference 2001*, vol. 2, pp. 995 – 1001, Nov., 2001.

[21] T. Tian, C. Jones, J. D. Villasenor, and R. D. Wesel, "Selective avoidance of cycles in irregular ldpc code construction," *IEEE Transactions on Communications*, vol. 52, no. 8, pp. 1242 – 1247, Aug., 2004.

[22] A. Ramamoorthy and R. D. Wesel, "Construction of short block length irregular ldpc codes," in *IEEE International Conference on Communications*, 2004.

[23] J. Kim, A. Ramamoorthy, and S. W. McLaughlin, "Design of efficiently-encodable rate-compatible irregular ldpc codes," *IEEE Trans. Commun. (to appear).available at http://arxiv.org/abs/0705.0543*.