On Code Design for Joint Energy and Information Transfer

Mehdi Dabirnia, *Student Member, IEEE*, and Tolga M. Duman, *Fellow, IEEE*

Abstract—**Harvesting energy from radio frequency signals along with transmitting data through them is appealing for different wireless communication scenarios, such as radio frequency identification (RFID) systems and implantable devices. In this paper, we propose a technique to design nonlinear codes for the use in such systems taking into account both energy transmission and error rate requirements. In particular, we propose using concatenation of a nonlinear trellis code (NLTC) with an outer low-density parity-check (LDPC) code. We design the NLTC based on maximization of its free distance. We give necessary and sufficient conditions for its catastrophicity; in order to avoid catastrophic codes, we connect each designed NLTC to a corresponding linear convolutional code allowing for the use of simpler conditions for verification. Furthermore, we use EXIT charts to design the outer LDPC code while fixing the inner NLTC. Via examples, we demonstrate that our designed codes operate at ∼0.8 dB away from the information theoretic limits, and they outperform both regular LDPC codes and optimized irregular LDPC codes for additive white Gaussian noise (AWGN) channels. In addition, we show that the proposed scheme outperforms the reference schemes of concatenating LDPC codes with nonlinear memoryless mappers and using classical linear block codes in a time switching mode.**

Index Terms—**RF energy harvesting, joint energy and information transfer, nonlinear codes, low density parity check codes.**

I. Introduction

Radio frequency (RF) energy harvesting is a wireless power transfer technique which relies on collecting energy from the radiated RF signals at the receiver for use in information processing and transmission processes. Potential applications of RF energy harvesting can be found in different areas including wireless sensor networks, wireless body networks and wireless charging systems [1].

Wireless energy transfer and wireless information transmission have previously been considered as separate problems. However, recent work on dual use of RF signals for delivering energy and information demonstrates that there is a natural trade-off on the design of such systems [2]. For systems with joint energy and information transfer it is of interest to increase received power levels and information rates at the same time. Using the most energetic symbol all the time is desirable for the first goal, whereas a uniform distribution on the channel input is required to maximize the mutual information over a symmetric noisy channel. Consequently, there is a natural trade-off and the amount of transmitted information and transferred energy cannot be generally maximized at the same time. Varshney in [2] described this fundamental tradeoff between transmission of energy and Information through the same signal. He proved that the capacity-energy function is a nonincreasing concave function, and obtained closed form expressions for it for several channels such as the noiseless and noisy binary symmetric channel and the Z-channel. He has also shown that for an AWGN channel with a given minimum received energy and maximum input amplitude constraint, the capacity achieving input distribution consists of a finite number of mass points [2].

In a related recent study, energy usage of the receiver has been modelled stochastically and battery overflow and underflow probabilities have been computed using classical codes and constrained run-length limited (RLL) codes [3]. The results show that constrained RLL codes are better suited for the receiver's energy utilization pattern compared to classical unconstrained ones. Binary code design for simultaneous energy and information transfer has been studied in [4] where achievable rates using constrained RLL codes on binary input noisy channels have been investigated. Most of the existing research in the area of joint energy and information transfer is on information theoretic approaches, specifically on capacity energy functions for different channels [2], on performance achievable with RLL codes [3], and achievable rates over some noisy binary channels using RLL codes [4]. Our focus in this paper is on the design of practical codes for simultaneous energy and information transfer complementing the existing results in the literature.

A trade-off between transmission of energy and information emerges when the amount of received energy differs for different channel input symbols (which is not the case for BPSK modulation). A simple model that makes this trade-off clear is using on-off signaling which has already been studied in some information theoretic works [2], [3]. For a more general case one might consider transmission of any set of symbols with different energy levels (amplitudes) such as QAM modulation. Here, we consider the case of on-off signaling in which only two symbols "0" and "1" are used, and with the primary objective to complement the existing information theoretic results, we investigate the joint energy and information transfer from a communication theoretic perspective.

We note that a traditional information receiver architecture designed for information reception is not able to harvest energy

Manuscript received November 27, 2015; revised March 16, 2016; accepted April 12, 2016. Date of publication April 28, 2016; date of current version June 14, 2016. This publication was made possible by NPRP Grant 4-1293-2-513 from Qatar National Research Fund (a member of Qatar Foundation). This work was presented at the 2015 IEEE International Conference on Communications, London, U.K., June 2015. The associate editor coordinating the review of this paper and approving it for publication was W. Zhang.

The authors are with the Department of Electrical and Electronics Engineering, Bilkent University, Ankara 06800, Turkey (e-mail: mehdi@ee.bilkent.edu.tr; duman@ee.bilkent.edu.tr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCOMM.2016.2557796

from the received RF signals. Motivated with this, there are ongoing efforts on designing receivers for joint energy and information transfer. These include separated receiver architectures [5], co-located receiver architectures [5], [6] which can further be categorized into two models, i.e., time-switching and power-splitting architectures, integrated receiver architectures [6], and ideal receiver architectures. The assumption for an ideal receiver is that it can harvest energy from the same signals used for information decoding without any energy loss, however, as mentioned in [6], this assumption is not practical. In this paper, we assume an ideal receiver as also done in several other recent related papers in the literature [2]–[4], and investigate the achievable reliable transmission rates using the proposed coding scheme. While an ideal receiver is adopted in this study, the same scheme and designed codes can be used with separated receivers, power-splitting architectures and integrated receivers to provide gains over the schemes using classical linear codes. Specifically, for the integrated receiver architecture which is the only proposed receiver with a single front-end that can perform both energy harvesting and information decoding at the same time, one needs to consider energy modulation [6] such as on-off signaling and design nonlinear codes with the required ones density to satisfy both the energy and reliable transmission requirements.

Linear codes such as convolutional codes and low density parity check (LDPC) codes have equal density of ones and zeros [10]. Hence, in order to transmit more than $\frac{1}{2}$ (normalized) energy per symbol, there is a need to design nonlinear codes with a desired ones' density which provide good error correction capabilities. With this motivation, we propose a coding scheme based on concatenation of a nonlinear trellis code (NLTC) with an outer linear block code, specifically an LDPC code. We describe an algorithm for the inner nonlinear trellis code design based on maximization of the minimum distance of the code. Then, we fix the designed NLTC and optimize the outer LDPC code using EXIT charts. Via several examples, we observe that the designed codes based on the proposed solution offer excellent performance in terms of operating near information theoretic limits, for instance, a particular design is only about 0.8 dB away.

The rest of the paper is organized as follows. In Section II, the channel model is described, information theoretic limits for the considered scenario are given and the proposed scheme of concatenation of an outer linear block code with a nonlinear trellis code is presented. The design of nonlinear trellis codes for our purposes is then introduced in Section III. Ways of avoiding catastrophic codes are discussed in Section IV. EXIT charts and LDPC code optimization are detailed in Section V. In Section VI, several numerical examples are provided, and finally, the paper is concluded in Section VII.

II. Proposed Coding Scheme

A. Channel Model

We consider an additive white Gaussian noise channel for which the input-output relationship is

$$Y = X + Z, \tag{1}$$

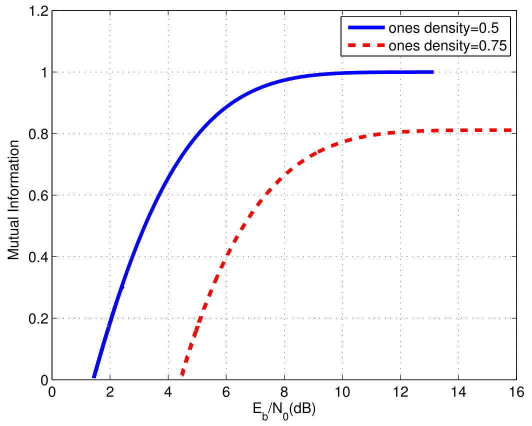

Fig. 1. Maximum transmission rate over an AWGN channel with on-off signaling for $p = \frac{1}{2}$ and $p = \frac{3}{4}$.

where $X \in \{0, 1\}$ and Z is independent and identically distributed (i.i.d.) Gaussian noise with zero mean and variance $\frac{N_0}{2}$. In order to model the trade-off between simultaneous energy and information transfer we need to consider signals with different energy levels. Here, we consider using on-off signaling where "1" (resp. "0") corresponds to the presence (resp. absence) of a signal. Using such a representation enables us to transmit more energy through the channel by using a code with a higher ones' density. Signal to noise ratio (SNR) at the receiver side with average ones' density p is defined as $\frac{E_b}{N_0} = \frac{p}{N_0}$. We assume that the receiver needs to harvest at least a certain amount of energy on average. In order to provide this required energy at the receiver side, we place a constraint on the average ones' density p at the channel input, i.e., on the coded symbols. Therefore, our aim is to design practical codes with a predetermined constraint on the average ones' density of the transmitted codewords.

B. Information Theoretic Limits

Assuming that the required ones' density is p and i.i.d. channel input symbols are used, the mutual information between the input and the output of an AWGN channel with a predetermined input distribution (in this case (i.i.d.) Bernoulli(p) probability mass function) is given by [7]

$$I(X; Y) = h(Y) - \frac{1}{2}log(\pi e N_0), \tag{2}$$

where

$$h(Y) = \int_{-\infty}^{\infty} f_Y(y) log\left(\frac{1}{f_Y(y)}\right) dy, \tag{3}$$

$$f_Y(y) = \frac{1}{\sqrt{\pi N_0}}\left((1-p)e^{-\frac{y^2}{N_0}} + pe^{-\frac{(y-1)^2}{N_0}}\right). \tag{4}$$

As an illustration, (2) is computed for $p = \frac{1}{2}$ and $p = \frac{3}{4}$, and the results are shown in Fig. 1 which clearly illustrates that there is a trade-off between the ones' density and the maximum possible transmission rate through the channel. That is, by choosing a ones' density of $p = \frac{3}{4}$, we can send more energy compared to the uniform input case, however, we sacrifice some data rate.

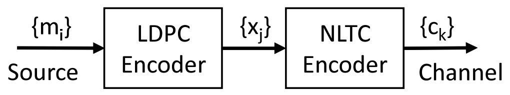

Fig. 2. Block diagram of the transmitter.

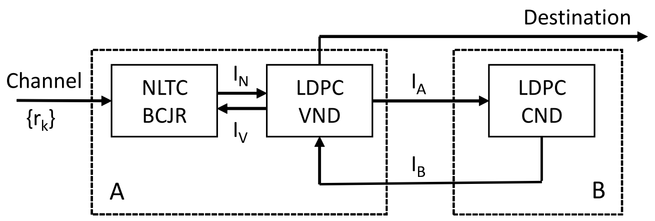

Fig. 3. Iterative decoder.

C. Concatenation of LDPC and Nonlinear Trellis Codes

We propose using concatenation of an outer linear block code such as an LDPC code with a nonlinear trellis code as a practical coding solution for joint energy and information transfer. The transmitter side shown in Fig. 2 consists of concatenation of an outer LDPC encoder and an inner NLTC encoder, which is directly connected to the channel. As shown in Fig. 2, binary message sequence $\{m_i\}$ is first encoded by a rate R_1 LDPC code into a binary sequence $\{x_j\}$. The binary symbols $\{x_j\}$ are then encoded with a rate R_2 and ones' density p NLTC to channel input symbols $\{c_k\}$ which results in an overall code rate of $R_1 R_2$.

The receiver is depicted in Fig. 3. The sequence of channel observations $\{r_k\}$ are the input of the receiver. We follow the scheme that is described in [8] and partition the receiver into two blocks, denoted as Block A and Block B. LDPC variable node decoder and LDPC check node decoder is represented as LDPC VND and LDPC CND subblocks in Fig. 3, respectively. Block A includes following subblocks:

- A BCJR decoder which is matched to the NLTC and the channel. This subblock computes a posteriori LLR values of the binary symbols $\{x_j\}$ based on the channel observations $\{r_k\}$ and the relevant apriori information from the subblock "LDPC VND".
- An LDPC VND that computes the LLR values of the binary symbols $\{x_j\}$ using LLR values from NLTC-BCJR and the information received from Block B based on the LDPC code constraints.

Block B includes the LDPC CND which computes the extrinsic LLR of binary symbols $\{x_j\}$ using a priori information received from Block A based on LDPC code constraints.

We note that only extrinsic LLR values are passed between component decoders and they are interpreted as a priori information by the recipient block. The overall decoding algorithm can be described as follows:

1) For initialization, the a priori LLRs of binary symbols $\{x_j\}$ at the input of Block A (from Block B) is set to zero (complete uncertainty).
2) Inside Block A the VND computes the a priori input for NLTC-BCJR by summing all the incoming messages from check nodes at each variable node.
3) NLTC-BCJR computes extrinsic LLRs of binary symbols $\{x_j\}$ based on the channel input and the input from the VND and passes it to VND as a priori input.
4) The VND computes the messages to be sent to Block B according to standard LDPC decoding, but using, as a priori input, the messages from NLTC-BCJR.
5) The CND computes the extrinsic LLRs to be passed to Block A according to the standard LDPC decoding.
6) The VND computes total LLRs and checks if the obtained codeword is valid. The algorithm iterates from step (2) until a valid codeword is obtained or the maximum number of allowed iterations is reached.

III. Nonlinear Trellis Code Design

In this section, we present a design technique for nonlinear trellis codes for use over an AWGN channel with the purpose of joint energy and information transfer. Our goal is to maximize the minimum Hamming distance between the codewords through the trellis while keeping a certain ones' density. Specifically, we use a finite-state shift register which consists of K (k-bit) stages and input data is shifted into and along the shift register (from the left) k bits at a time. The contents of the shift register (Kk previous input bits) specify the state of the encoder, and the state transitions and corresponding branches are determined by the previous state of the encoder and the input data at that time instant. A nonlinear look-up table is used to assign encoder outputs for each branch. Encoder outputs are chosen to provide the required ones' density p. An example of a trellis with memory $M = 3$ and $k = 1$ is shown in Fig. 5. We note that although we do not consider all types of trellises, the specific class that we consider is rich enough to obtain good results for our purposes as will be illustrated later.

A. Generating and Partitioning the Labels

In this step, considering the desired ones' density p and code rate $R = \frac{k}{n_0}$, we generate labels with length n_0 and average Hamming weight $\omega = pn_0$. We assume that all the labels are used with the same frequency, hence the average Hamming weight of the selected subset of binary labels should be equal to ω. For the rest of the paper, we consider NLTC of rate $R = \frac{1}{n_0}$ and simply note that one can carefully generalize these ideas to the case with $R = \frac{k}{n_0}$, $k > 1$.

Given the selected subset of binary labels, with the goal of maximizing the minimum distance between the codewords, we perform set partitioning. In order to do this, we first partition the labels into pairs in such a way that the minimum pairwise Hamming distance between labels in each pair ($d_{min}^{(1)}$) is maximized. Then, we partition the pairs into groups of two pairs such that the minimum pairwise distance between the quadruples ($d_{min}^{(2)}$) is maximized and continue partitioning in this manner. We denote the minimum pairwise Hamming distance of groups of 2^i labels as ($d_{min}^{(i)}$). Assuming that the subset of labels has size 2^h, we obtain a partition tree with h levels. There may be many ways to accomplish this, however, we select one of the possibilities that maximizes $\sum_{i=1}^{h} d_{min}^{(i)}$. Using the partitioned labels and applying the

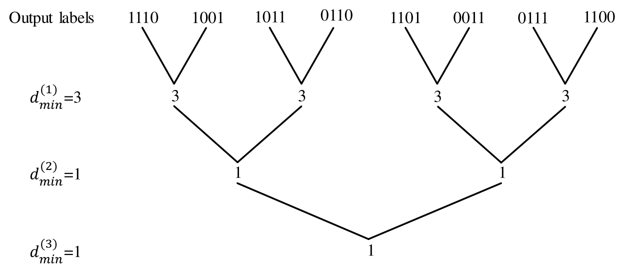

Fig. 4. Generated labels and their partition tree for $R = \frac{1}{4}$ and $p = \frac{5}{8}$ and the "within-distance" at different levels.

extended Ungerboeck's rule (which will be further discussed in Section III-B), we can generate a nonlinear trellis code of memory M, where $M + 1 \geq 2h$, with a lower bound on its minimum distance given by

$$\text{min-distance} \geq 2 \sum_{i=1}^{h} d_{min}^{(i)}. \tag{5}$$

Assigning 2^i labels that have a distance of at least d_{min}^i in the i-th level of the partition tree with the branches emanating from (combining at) a split (merge) i sections before (later), we add d_{min}^i to the distance between two corresponding paths. One should note here that the first merge after any split from a given path will be at least $M + 1$ trellis sections later, and we need h trellis sections after split and before merge without overlap between them to include every $d_{min}^{(i)}$, $\forall i$ s.t. $1 \leq i \leq h$ in calculating the lower bound. The paths merging in a larger number of steps than $M + 1$ will also have at least the same minimum distance. If $M + 1 < 2h$, there are not a sufficient number of trellis sections until the first merge after any split to include all the distance levels d_{min}^i, $i = 1, 2, \ldots, h$, in other words, the distances can be included until $d_{min}^{\lfloor \frac{M+1}{2} \rfloor}$ after each split and before each merge plus one more level with distance $d_{min}^{\lceil \frac{M+1}{2} \rceil}$ on either split or merge side if M is even. Hence, the lower bound on minimum distance for the case of $M+1 < 2h$ can be rewritten as,

$$\text{min-distance} \geq 2 \sum_{i=1}^{\lfloor \frac{M+1}{2} \rfloor} d_{min}^{(i)} + \mathbb{1}_{even}(M) d_{min}^{(\lceil \frac{M+1}{2} \rceil)}, \tag{6}$$

where $\mathbb{1}_{even}(M)$ is the indicator function defined as,

$$\mathbb{1}_{even}(M) \triangleq \begin{cases} 1 & \text{if } M \text{ is even,} \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

In the case where more than one such subset of labels are available, we select the one that results in the largest lower bound on the minimum distance.

As an example, we consider a design with rate $R = \frac{1}{4}$ and ones' density $p = \frac{5}{8}$. Generated labels, a partition tree with three levels and minimum distance at each level are shown in Fig. 4. For this example, the minimum distance of the code with memory $M \geq 5$ using these labels satisfies min-distance ≥ 10.

B. General Rules for Label Assignment

The main step in the NLTC design is to assign output values to the branches of the trellis to maximize the minimum distance of the code while keeping the desired ones' density. In our design, assignment of output labels to branches is performed according to the extended Ungerboeck's rule [9] which maximizes the minimum Hamming distance of the code. Ungerboeck noted that every incorrect codeword, in its trellis representation, departs from the correct path (split) and returns to it (merge) at least once, so he maximized the distance between splits and merges. One can extend the Ungerboeck's rule further into the trellis and maximize the Hamming distance between the branches emanating from a split h trellis sections before, where h is a natural number that can be at most M. The same procedure can be followed for the branches that merge h sections later. Having the set of 2^h distinct labels partitioned in the previous sections for a rate $\frac{1}{n_0}$ code, we can assign the labels according to the extended Ungerboeck's rule as follows:

- 2^i labels in the same partition at level i of the partition tree are assigned to 2^i branches emanating from (merging at) the same state i trellis sections before (later).
- All the labels are used equally often.

The main difference between our approach for NLTC design and the earlier work of [9] is in the target code rates. That is, the codes designed in [9] are of small rates and are intended for use over a multiple access channel, therefore the sum rate is important for their goal. However, here we design codes of high rates with a specified ones' density. One of the design constraints used in [9] is to ensure that all the branches produced by the same input to have different output labels which cannot be satisfied for codes of high rates with large memories. However, we know that in order to obtain large minimum distances for higher rate codes we should use trellises with large memories. Note that as the memory of the trellis increases, assignment of the labels to branches becomes more complicated, hence we need to have a systematic algorithm to apply the design rules.

In order to design higher rate codes, one needs to consider codes of rate $\frac{k}{n_0}$, with $k > 1$. In the following we provide a sketch of how the NLTC code design principles can be generalized to the case of rate $\frac{k}{n_0}$, however, due to the space limitations, we do not give any specific code design examples. The required modifications are as follows:

- selecting a subset of size 2^h from binary sequences of length n_0 where $h \geq k + 1$ that satisfies the desired ones density p,
- organizing the selected labels in a partition tree as the case of rate $\frac{1}{n_0}$ with the only difference of partitioning in groups of size 2^k at the first level of the partition tree,
- assigning the partitioned labels to the branches of trellis such to satisfy Ungerboeck's rule for branches emanating from (merging at) each state.

C. Grouping of Branches for a Specific Trellis

The aim of this section is to describe an algorithm to arrange the trellis branches in 2^h groups (where h is the number of trellis sections for which the Ungerboeck's rule will be extended after each split and before each merge) and to assign partitioned labels to these groups with the same order in which they appear in the partition tree. First, we number the states in natural order starting from zero and assign indexes to outgoing branches from each state as follows: for state number $i \in \{0, 1, \ldots, 2^M - 1\}$

$$branch-index = \begin{cases} 2i & \text{if input } u = 0, \\ 2i+1 & \text{if input } u = 1. \end{cases} \quad (8)$$

Then, we arrange each set of four branches with consecutive indexes (first one starting with branch with index zero) in two subgroups and represent them using two blocks A_l and A_l^* as follows

(I) A_l

(0)	(1)
$4l$	$4l+1$
$4l+3$	$4l+2$

(II) A_l^*

(0)	(1)
$4l+1$	$4l$
$4l+2$	$4l+3$

$\forall l = 0, 1, \ldots, 2^{M-1} - 1$. The columns of these blocks represent two different subgroups (0) and (1). We define the star operation (*) above as exchanging the branches between subgroup (0) and subgroup (1). A trivial observation is that $(A^*)^* = A$. Two branches emanating from each split and two branches combining at each merge are placed in different subgroups inside these blocks, hence assigning different labels to subgroup (0) and subgroup (1) of each block ensures that the Ungerboeck's rule is satisfied for the first section of splits and merges. Using these blocks simplifies grouping of the branches. Hence, we need to arrange these blocks in 2^{h-1} groups for which, each of these groups have two subgroups (0) and (1). We define the group operator $G(.)$ over blocks (and branches) which specifies the group index for the input block (branch), i.e., $G(A_i)$ is the group index of block A_i and $G(S = 2i)_{u=0}$ is the group index of branch corresponding to state $2i$ with input $u = 0$. Placing block A_i in group j which can be shown by $G(A_i) = j$ means that $G(S = 2i)_{u=0} = j(0)$, $G(S = 2i)_{u=1} = j(1)$, $G(S = 2i+1)_{u=0} = j(1)$ and $G(S = 2i+1)_{u=1} = j(0)$. Accordingly if $G(A_i^*) = j$ then $G(S = 2i)_{u=0} = j(1)$, $G(S = 2i)_{u=1} = j(0)$, $G(S = 2i+1)_{u=0} = j(0)$ and $G(S = 2i+1)_{u=1} = j(1)$. After completing grouping of these blocks we will need to assign the ith pair of labels from the partition tree to the group with index i to complete the label assignment process.

In the following we go on with an example to illustrate the grouping step and then we extend the rules to the general case. We consider the 8-state trellis shown in Fig. 5 with $h = 3$ and rate $\frac{1}{n_0}$. Considering the second stage, four branches with indexes $\{0, 1, 8, 9\}$ at the second section of the trellis in Fig. 5 emanating from a split at first section needs to be arranged in different groups, i.e., blocks A_0 and A_2 should be placed in different groups. Extended rule for the third section after a split forces to arrange eight branches with indexes $\{0, 1, 4, 5, 8, 9, 12, 13\}$ in 8 different groups, i.e., blocks A_0, A_1, A_2 and A_3 should be placed in different groups. Note that applying the rule to the third section after splits also

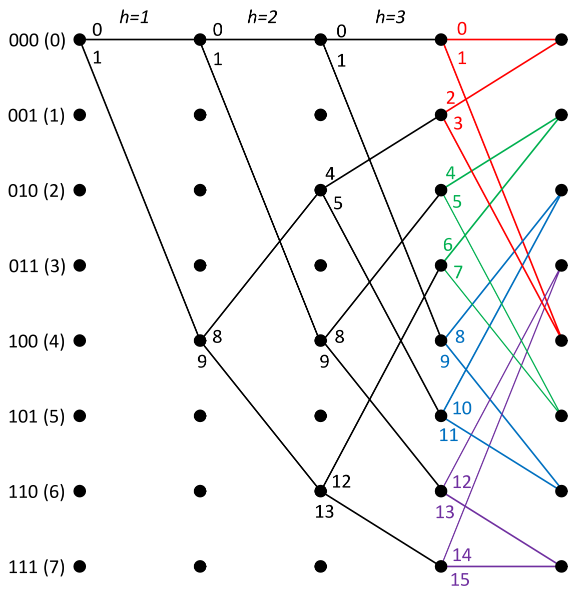

Fig. 5. 8-state trellis diagram and extension of the Ungerboeck's rule.

TABLE I
ASSIGNMENT OF LABELS FOR THE EXAMPLE OF 8-STATE TRELLIS

Output label	1110	1001	1011	0110	1101	0011	0111	1100
Group index	0(0)	0(1)	1(0)	1(1)	2(0)	2(1)	3(0)	3(1)
Blocks	A_0		A_1		A_2		A_3	

satisfies the rule for the first and the second sections. The same idea can be applied to the merges. Following these rules for this example and using the partitioned labels in Fig. 4 result in Table I.

The design rules can be generalized for any h as follows:

Split Rule: Place 2^{h-1} blocks with indexes $\left\{i, \frac{2^M}{2^h} + i, \frac{2\times 2^M}{2^h} + i, \ldots, \frac{(2^{h-1}-1)\times 2^M}{2^h} + i\right\}$, $\forall i = 0, 1, \ldots, \frac{2^M}{2^h} - 1$ in different groups.

Merge Rule: Place 2^{h-1} blocks with indexes $\left\{2^{h-1}i + l | l \in \{0, 1, \ldots, 2^{h-1} - 1\}\right\}$, $\forall i = 0, 1, \ldots, \frac{2^M}{2^h} - 1$ in different groups.

One should note that for the trivial case of $h > M$ which corresponds to the case where the number of distinct labels are greater than or equal to the number of branches, these rules will not apply and one can easily assign distinct output labels to each of the branches.

According to the above rules we propose the following algorithm for the grouping step:

1. Decide about h (number of trellis sections for which the Ungerboeck's rule will be extended) and memory of the trellis M (number of blocks will be 2^{M-1}).
2. Place each block with index j, $\forall j = 0, 1, \ldots, 2^{h-1} - 1$, in the group with index j as A_j, and assign $i = h$.
3. Considering the split rule, put the block with index 2^{i-1} in one of the admissible groups as $A_{2^{i-1}}$ or $A^*_{2^{i-1}}$ (split rule may restrict these admissible placements),

TABLE II
GROUPING RESULT OF TRELLIS OF MEMORY $M = 8$ FOR $h = 2$

0	1	0	1	0	1	0	1
A_0	A_1	A^*_{32}	A^*_{33}	A_{65}	A_{64}	A^*_{97}	A^*_{96}
A^*_2	A^*_3	A_{34}	A_{35}	A^*_{67}	A^*_{66}	A_{99}	A_{98}
A^*_5	A^*_4	A_{37}	A_{36}	A^*_{68}	A^*_{69}	A_{100}	A_{101}
A_7	A_6	A^*_{38}	A^*_{39}	A_{71}	A_{70}	A^*_{103}	A^*_{102}
A^*_8	A^*_9	A_{40}	A_{41}	A^*_{73}	A^*_{72}	A_{105}	A_{104}
A_{10}	A_{11}	A^*_{42}	A^*_{43}	A_{75}	A_{74}	A^*_{107}	A^*_{106}
A_{13}	A_{12}	A^*_{44}	A^*_{45}	A_{77}	A_{76}	A^*_{109}	A^*_{108}
A^*_{15}	A^*_{14}	A_{47}	A_{46}	A^*_{78}	A^*_{79}	A_{110}	A_{111}
A^*_{17}	A^*_{16}	A_{49}	A_{48}	A^*_{80}	A^*_{81}	A_{112}	A_{113}
A_{19}	A_{18}	A^*_{50}	A^*_{51}	A_{83}	A_{82}	A^*_{115}	A^*_{114}
A_{20}	A_{21}	A^*_{52}	A^*_{53}	A_{85}	A_{84}	A^*_{117}	A^*_{116}
A^*_{22}	A^*_{23}	A_{54}	A_{55}	A^*_{87}	A^*_{86}	A_{119}	A_{118}
A_{25}	A_{24}	A^*_{56}	A^*_{57}	A_{89}	A_{88}	A^*_{121}	A^*_{120}
A^*_{27}	A^*_{26}	A_{59}	A_{58}	A^*_{90}	A^*_{91}	A_{122}	A_{123}
A^*_{28}	A^*_{29}	A_{60}	A_{61}	A^*_{93}	A^*_{92}	A_{125}	A_{124}
A_{30}	A_{31}	A^*_{62}	A^*_{63}	A_{95}	A_{94}	A^*_{127}	A^*_{126}

then place blocks with indexes $2^{i-1} + j$, $\forall j$ s.t. $1 \leq j \leq 2^{i-1} - 1$ in the group with index $g \oplus g'$, and place it as $A^*_{2^{i-1}+j}$ if only one of the $A_{2^{i-1}}$ and A_j is placed as A^* otherwise place it as $A_{2^{i-1}+j}$, where g and g' are binary group indexes of $A_{2^{i-1}}$ and A_j respectively, and $\oplus$ is bitwise XOR operation.

4. If there are blocks that are not placed inside any groups yet, increment i by one and repeat step 3.

To make this more clear we give an example of trellis with memory $M = 8$ in which we have 128 blocks and we want to arrange these blocks in 2 different groups $h = 2$. We start by placing blocks with indexes 0 to 1 in groups with the same indexes. Then, at the second step there is no restriction for placing block with index 2 regarding the split rule and blocks that are already placed inside groups so we can select either block A_2 or A^*_2 and place it in one of the groups $\{0, 1\}$. For instance, here we select the block A^*_2 and place it in group 0, Then according to the third step of the algorithm we should place block A^*_3 in group 1. A sample result after completing the grouping for this trellis is shown in Table II in which the group indexes are shown in the first row.

At the end, after completing the grouping of the blocks, we assign distinct labels to each group. We use the partitioned labels obtained previously and assign them to the groups with the same order in which they appear in the partition tree.

IV. AVOIDING CATASTROPHIC CODES

We refer to an NLTC that is prone to catastrophic error propagation as a catastrophic code for which a finite number of channel errors may cause an infinite number of decoder errors. With this reference, we derive the necessary and sufficient conditions for catastrophicity of nonlinear trellis codes which is stated in Theorem 1.

Theorem 1: A nonlinear trellis code for which the trellis is defined based on sequence of shift registers is catastrophic if and only if one of these two conditions occur: 1) there is a cycle in its state diagram such that starting from at least two different states of the cycle and traversing around results in the same output sequences corresponding to different input sequences, 2) there are at least two different cycles in its state diagram with different input sequences giving rise to the same output sequence.

Proof: See Appendix A. ■

Catastrophic codes clearly need to be avoided to achieve good error correction capabilities, however, checking for the conditions mentioned in Theorem 1 when the memory of the trellis grows can be very complicated. We already know catastrophicity conditions for linear convolutional codes [10], that is a convolutional code is catastrophic if and only if its corresponding state diagram contains a circuit in which a nonzero input sequence corresponds to an all-zero output sequence. Therefore, checking for catastrohicity of convolutional codes and avoiding such codes is much simpler compared to a nonlinear trellis code. Fortunately, for the specific design (grouping) that we introduced in the previous section, we can show that checking for the code being catastrophic can be performed in an easy and systematic manner by connecting the design to that of standard convolutional codes. In the following, we introduce two theorems and one corollary in order to detect and avoid catastrophic codes in our specific design in Section III-C.

Theorem 2: For the design in Section III, there exists a one-to-one mapping (which is not necessarily unique) from the full set of binary labels with h bits to the group indexes such that assignment of the corresponding binary labels to the branches inside the group results in a linear convolutional code. The resulting convolutional code is called the corresponding convolutional code of the original NLTC.

Proof: The proof is constructive. We will show that if we assign binary values of $(2j, (2^h - 1) - 2j)$ to groups with indexes $(j(0), j(1))$, $j \in 0, 1, \ldots, 2^{h-1} - 1$, respectively, the following will be the generator polynomials for the resulting convolutional code (note that two labels are one's complement of each other):

$$G_1(D) = 1 + \sum_{i=h}^{M-1} a_{1,i} D^{M-i} + D^M,$$

$$G_2(D) = 1 + \sum_{i=h}^{M-1} a_{2,i} D^{M-i} + D^{M-1} + D^M,$$

$$\ldots = \qquad \ldots$$

$$G_h(D) = 1 + \sum_{i=h}^{M-1} a_{h,i} D^{M-i} + D^{M-h+1} + D^M,$$

where $a_{h,i} \ldots a_{1,i}$ and its one's complement are, respectively, the corresponding outputs for subgroup (0) and (1) of the block with index 2^{i-1}. Note that the coefficient of D^M in multiplication of polynomials $G_i(D)$ by $u_k D^M + u_{k-1} D^{M-1} + \ldots + u_{k-M}$ (where u_k is the current input and $u_{k-1} \ldots u_{k-M}$ is the binary value of the current state) gives the i-th bit of the output.

In the following, in two parts, we will show that for any branch both generator polynomials and the look-up

TABLE III

ASSIGNMENT OF LABELS FOR THE CORRESPONDING CONVOLUTIONAL CODE

Output label	000	111	010	101	100	011	110	001
Group index	0(0)	0(1)	1(0)	1(1)	2(0)	2(1)	3(0)	3(1)
Blocks	A_0 A_5^*		A_1 A_4^*		A_2 A_7^*		A_3 A_6^*	

table from the proposed grouping algorithm give the same output.

First we show that if the claim is true for the first branch inside a block then it will also be true for the rest of the branches inside that block. The second and third branches inside a block have the same output which is one's complement of the first branch's output. The same will be obtained by generator polynomials since the only change for second branch is that the input is changed from zero to one for the same state and for the third branch s_1 (LSB of state value) is changed from zero to one for the same input. The fourth branch and the first branch have the same output which can again be obtained by generator polynomials since now both input and s_1 are changed from zero to one and they cancel each other at every bit of the output. Hence, without loss of generality, we can check the claim for the first branch inside each block (which corresponds to even states with input zero) and make sure that the rest will be correct if the first one is correct.

The second part of the proof follows using induction. As the initial step of the induction it can be shown that the claim is true for any branch inside blocks 0 to $2^{h-1} - 1$. Then in the second step of induction by assuming that the claim is true for any branch inside blocks 0 to $2^{i-1} - 1$, it can be shown that it is also true for any branch inside blocks 2^{i-1} to $2^i - 1$. We relegate the details of the proof to Appendix B. ■

We give a specific example of constructing corresponding convolutional code for a designed grouping. Let the trellis memory be $M = 4$ and $h = 3$. For the grouping in Table III obtained earlier, we can assign the outputs (as stated in the proof of Theorem 2) and obtain the generating polynomials as $G_1 = 1 + D + D^4, G_2 = 1 + D^3 + D^4, G_3 = 1 + D + D^2 + D^4$.

Theorem 3: An NLTC is catastrophic if and only if the corresponding convolutional code is catastrophic.

Proof: First assume that the corresponding convolutional code is catastrophic, by definition there is a cycle in its state diagram with a nonzero input sequence which corresponds to the all-zero output sequence. We also know that every convolutional code has a cycle from state zero to itself with zero input and all-zero output. Due to the one-to-one mapping between labels, the NLTC will have two different cycles with different input sequences but the same output sequence which means that NLTC is catastrophic. Conversely if NLTC is catastrophic, from the definition at least one of the following conditions is true, 1) there is a cycle in its state diagram such starting from at least two different states of the cycle and traversing around, results in the same output sequence, 2) there are at least two different cycles in its state diagram with same input sequence but different output sequences. Again because of the one-to-one mapping in either one of these cases, the same condition for the convolutional code will also be true which show that the corresponding convolutional code is prone to catastrophic error propagation. ■

Corollary 1: All non-recursive NLTCs of rate $\frac{1}{n_0}$ designed using less than four distinct output labels by our algorithm are catastrophic.

Proof: We already showed in Theorem 2 that our designed NLTCs are combinations of a non-recursive convolutional code and a one-to-one mapping, hence we can use the theorem in [11, Sec. 4.2] which states that if the encoder of a rate $\frac{1}{n_0}$ constraint length K fixed binary convolutional code is initially in any nonzero state and $K - 1$ input symbols are shifted into the shift register, then all $n_0(K - 1)$ output symbols can be zero only if the code exhibits catastrophic error propagation. We consider that only two distinct output labels are used in NLTC design which are mapped into 0 and 1 in corresponding convolutional code, also we know that by our design algorithm we make sure that two branches of each split have different labels, so, the corresponding convolutional code at each split in the trellis has one branch with output label 0. Using the mentioned theorem from [11] it is obvious that the corresponding convolutional code and hence the NLTC is catastrophic. ■

In a convolutional code with two distinct labels, if we map one of the output labels (0 or 1) to two different output labels (each one-half of the time), then we will obtain an NLTC with three distinct labels in which one of the labels is used with twice the frequency of the other two. It is straightforward to show that the resulting NLTC will again be a catastrophic code.[1] The above two theorems and the corollary allow us to avoid catastrophic codes in a simple way, and they are utilized in the next section for our specific design examples.

V. OUTER LDPC CODE OPTIMIZATION

A. EXIT Chart-Based Analysis

In this section, we use EXIT charts to characterize the iterative decoder's operation. In order to do this we need to draw the EXIT curve for each subblock in the iterative decoder. Following the notation in [8], we denote the mutual information terms at the output of block A and B as I_A and I_B, respectively. Also, mutual information at the input and output of the NLTC-BCJR subblock is shown by I_V and I_N (Fig. 3). We follow the iterative update of mutual information as the measure of decoding performance. When the mutual information converges to 1 it shows that the probability of error will converge to zero. By combining the EXIT curve of LDPC VND with that of the NLTC-BCJR, we can obtain the exit curve for block A. EXIT curve of block B is simply the EXIT curve of LDPC CND. By assuming the Gaussian distribution for exchanged messages between these subblocks we can use analytical formulas for I_A, I_B and I_V. In order to calculate the I_N at the output of the NLTC-BCJR, we use

[1]NLTC of rate $\frac{1}{2}$ in [12, Table I] is catastrophic.

Monte Carlo simulations. We have

$$I_A = \sum_i \lambda_i J\left(\sqrt{(i-1)(J^{-1}(I_B))^2 + (J^{-1}(I_S))^2}\right), \quad (9)$$

$$I_B = 1 - \sum_j \rho_j J\left(\sqrt{(j-1)}J^{-1}(1-I_A)\right), \quad (10)$$

$$I_V = \sum_i \lambda_i J\left(\sqrt{i}J^{-1}(I_B)\right), \quad (11)$$

where $J(.)$ is defined as

$$J(\sigma) = 1 - \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(l-\frac{\sigma^2}{2})^2}{2\sigma^2}} \log_2(1+e^{-l})dl. \quad (12)$$

B. Degree Distribution Optimization Using EXIT Charts

As it is studied previously in [8], using powerful off-the-shelf LDPC codes designed for AWGN channels is not sufficient to achieve near channel capacity performance for all possible inner codes or modulation schemes. In general, the LDPC code has to be optimized for the specific coding and modulation scheme, and EXIT analysis is a powerful tool that can be exploited for this purpose.

In order to perform the LDPC code optimization we fix the inner NLTC and do the optimization over the LDPC degree distribution. Inspired by the results of [8] which suggest that using multiple degrees at both the variable node and the check node sides could result in lower thresholds for the concatenated scheme, we select an initial degree distribution with multiple degrees at both sides satisfying the required code rate. Using the analytical formulas at VND and CND, and performing Monte Carlo simulations for sufficiently long block-lengths for NLTC-BCJR, we calculate the threshold for this initial degree distribution. We track the evolution of the mutual information at the output of blocks A and B, and stop and call the degree distribution admissible if I_B (MI at the output of check nodes) evolves to 0.995.

At the NLTC-BCJR decoder, we consider a random input sequence of length 10^6 and based on input mutual information I_v, we generate Gaussian distributed intrinsic LLRs with the same MI for the random input sequence and apply it as input to the soft-input soft-output BCJR decoder. We calculate the MI at the output of this subblock by estimating the probability density functions $P_L(l|0)$, $P_L(l|1)$ from the obtained extrinsic LLRs and using the following formula

$$\begin{aligned} I(X;L) &= 1 - E\left[\log_2\left(\frac{1}{P_{X|L}(x|l)}\right)\right] \\ &= 1 - \sum_{x=0,1}\frac{1}{2}\int_{-\infty}^{\infty} P_L(l|x)\log_2\left(\frac{P_L(l|0)+P_L(l|1)}{P_L(l|x)}\right)dl. \quad (13)\end{aligned}$$

We employ a specific implementation of differential evolution (DE) [13] for designing the LDPC code. We use perturbing vectors to generate new instances of degree distributions with lower thresholds following the approach utilized in [8] in an iterative fashion. Both variable and check node degree distributions are perturbed as $\widetilde{\lambda_i} = \lambda_i + e_{1i}$, $\widetilde{\rho_j} = \rho_j + e_{2j}$ where e_{1i} and e_{2j} denote the ith and the jth elements of perturbing vectors. For the degree distribution to be valid the following equations should be satisfied

$$\sum_{i=2}^{d_v} \lambda_i + e_{1i} = 1, 0 \le \lambda_i + e_{1i} \le 1, 2 \le i \le d_v, \quad (14)$$

$$\sum_{j=2}^{d_c} \rho_j + e_{2j} = 1, 0 \le \rho_j + e_{2j} \le 1, 2 \le j \le d_c. \quad (15)$$

Also we keep the rate of the code unchanged during the optimization, i.e., we take

$$1 - \frac{\sum_{j=2}^{d_c}\frac{\rho_j+e_{2j}}{j}}{\sum_{i=2}^{d_v}\frac{\lambda_i+e_{2i}}{i}} = r. \quad (16)$$

We draw all the elements of the perturbing vectors except three from a normal distribution $\mathcal{N}(0,\sigma^2)$ where σ is a design coefficient. The remaining three elements are obtained by solving linear equations in (14)-(16). The perturbed degree distribution will replace the current one if it has a lower threshold, otherwise it is dismissed and new perturbation is performed. The procedure is terminated if no improvement can be obtained after a predetermined number of iterations.

TABLE IV
LABEL ASSIGNMENT TO THE BRANCHES OF 16-STATE TRELLIS ($M = 4$) USING THE PROPOSED ALGORITHM

Branch index								Output label (rate $\frac{1}{3}/\frac{1}{4}$)
1	4	10	11	21	24	30	31	011/0111
2	3	9	12	22	23	29	32	101/1011
5	8	14	15	17	20	26	27	110/1101
6	7	13	16	18	19	25	28	111/1110

VI. NUMERICAL EXAMPLES

In this section, we present several examples of the designed codes (both inner NLTC and outer LDPC codes) for joint information and energy transfer, and evaluate their performance over an AWGN channel. As a first example, we consider an NLTC of rate $\frac{1}{3}$ with memory $M = 4$ and ones' density $p = \frac{3}{4}$. Label assignment table for the trellis of this NLTC is shown in Table IV (note that the branches are represented by the branch number introduced in (8)). As the outer LDPC code, we consider three different codes all of rate $\frac{1}{2}$: the first one is the regular (3, 6) LDPC code, the second one is an optimized irregular LDPC code for AWGN channel with maximum variable degree 50 obtained from [14], and the third one is the optimized LDPC code specifically for the inner NLTC employed by the algorithm developed in the previous section with

$$\rho_3 = 0.48052,\ \rho_4 = 0.00315,\ \rho_8 = 0.01327,\ \rho_{15} = 0.50306,$$
$$\lambda_2 = 0.55833,\ \lambda_3 = 0.03322,\ \lambda_4 = 0.40845.$$

We evaluate the performance of the optimized code ensemble through finite block-length simulations and computation of decoding thresholds. The overall rate of the coding scheme is $R = \frac{1}{6}$. The information theoretic results indicate that we need about 4.99 dB for reliable communication at this

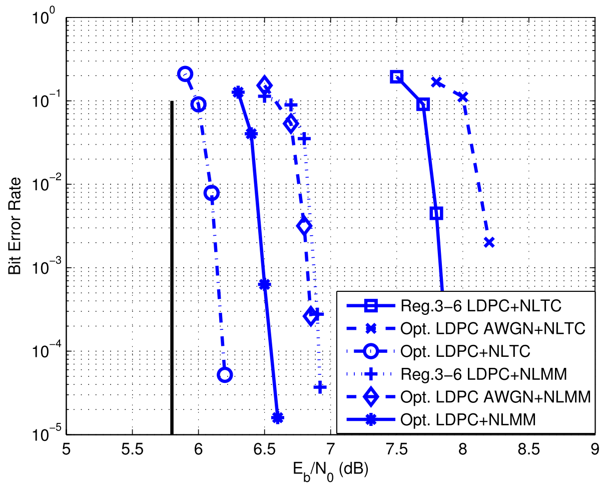

Fig. 6. Bit error rate performance of three LDPC codes concatenated with the NLTC of rate $R = \frac{1}{3}$, memory ($M = 4$), and ones' density $p = \frac{3}{4}$. Outer LDPC codes are of rate $R = \frac{1}{2}$ and block-length 100k.

transmission rate for the specified ones' density of $p = \frac{3}{4}$ (Fig. 1). Considering that the decoding threshold for the optimized degree distribution is at 5.8 dB (which is obtained by Monte Carlo simulations and without any Gaussian approximation), we observe that the proposed coding scheme has a performance about 0.8 dB away from this limit. To study the performance of specific codes from the designed ensemble, parity check matrices for a block-length of 100k are obtained using the tools in [15] where the length-4 cycles are removed for improved performance. The resulting bit error rates are depicted in Fig. 6. We observe that the optimized LDPC code for an AWGN channel has the worst performance, and even the regular (3, 6) code performs better when concatenated with the NLTC. The optimized code for the specific NLTC performs the best with a gain of about 1.65 dB compared to the regular (3, 6) code at an error rate of 10^{-3}.

For comparison purposes, we also consider a reference scheme of using a nonlinear memoryless mapper (NLMM) instead of an NLTC, concatenated with an outer LDPC code. For the ongoing example, we use the NLMM shown in Table V and design a rate $\frac{1}{2}$ outer LDPC code using the optimization method described in the previous section. The resulting degree distribution is

$$\rho_7 = 0.94397, \quad \rho_8 = 0.05603,$$
$$\lambda_2 = 0.33052, \lambda_3 = 0.21239, \lambda_4 = 0.01314, \lambda_{10} = 0.44395.$$

Bit error rate results for codes (of length 100k) picked from this ensemble, along with two other reference codes are also reported in Fig. 6. We observe that the proposed scheme using NLTCs yields a gain of about 0.4 dB in terms of the BER performance over the reference scheme with NLMMs.

As another example, we consider an NLTC of rate $R = \frac{1}{4}$ with memory $M = 4$ and ones' density $p = \frac{3}{4}$. The label assignment for this code is based on the look-up table shown in Table IV. Again as the outer codes, we consider three different LDPC codes of rate $\frac{1}{2}$, of which the first and the second ones are the regular (3, 6) LDPC code and the optimized irregular LDPC code for an AWGN channel, respectively. The third one is an optimized LDPC code for the specific NLTC used in this example with

$$\rho_3 = 0.46241, \ \rho_4 = 0.03137, \ \rho_8 = 0.00871, \ \rho_{15} = 0.49751,$$
$$\lambda_2 = 0.55613, \ \lambda_3 = 0.04170, \ \lambda_4 = 0.40217.$$

The overall rate of the coding scheme is $R = \frac{1}{8}$. The information theoretic limit for this transmission rate with ones' density $p = \frac{3}{4}$ is at about 4.84 dB and the threshold for the optimized degree distribution is at 5.6 dB (which is obtained without any Gaussian approximation). We observe that the proposed scheme operates at about 0.8 dB from the information theoretic limit. Parity check matrices for the outer LDPC codes are generated using the tools in [15] and also optimized by removing length-4 cycles. Fig. 7 shows the decoding results for concatenation of these three different LDPC codes with the inner NLTC. We observe that the optimized LDPC code beats the two other alternatives as expected. It has gain of about 1.65 dB compared to the regular (3,6) code and of about 2 dB over the AWGN-optimized LDPC code at an error rate of 10^{-3}.

The examples above show the importance of the LDPC code optimization for the specific inner NLTC, and illustrate that large performance improvements can be obtained by using optimized degree distributions for each inner code.

As another example, we consider both NLTCs from the two previous examples, and optimize an outer LDPC code of

TABLE V
NONLINEAR MEMORYLESS MAPPER OF RATE $R = \frac{1}{3}$ AND ONES' DENSITY $p = \frac{3}{4}$

Mapper input	Mapper output
00	011011
01	101101
10	110111
11	111110

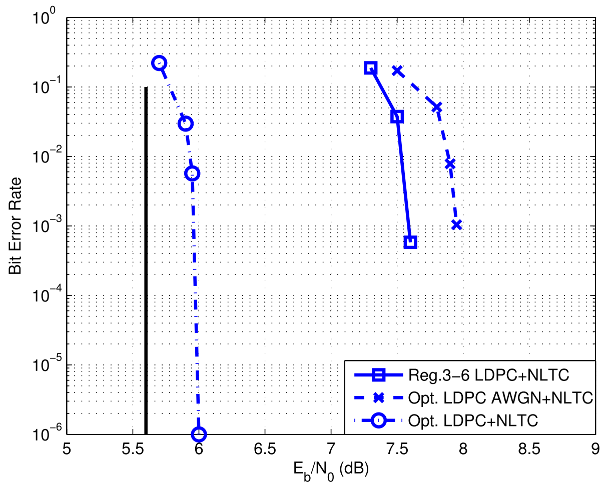

Fig. 7. Bit error rate performance of three LDPC codes concatenated with the NLTC of rate $R = \frac{1}{4}$, memory ($M = 4$), and ones' density $p = \frac{3}{4}$. Outer LDPC codes are of rate $R = \frac{1}{2}$ and block-length 100k.

TABLE VI

OPTIMIZED DEGREE DISTRIBUTIONS OF LDPC CODES OF RATE $R = 0.823$ FOR CONCATENATION WITH NLTCS OF TABLE IV

NLTC rate	$\frac{1}{3}$	$\frac{1}{4}$		$\frac{1}{3}$	$\frac{1}{4}$
λ_2	0.53462	0.52982	ρ_3	0.01075	0.00467
λ_3	0.02631	0.00838	ρ_4	0.02634	0.03621
λ_7	0.0004	0.01381	ρ_9	0.21403	0.17650
λ_8	0.06054	0.03963	ρ_{10}	0.04209	0.06318
λ_{11}	0.09629	0.02539	ρ_{39}	0.32566	0.39733
λ_{12}	0.28184	0.38297	ρ_{40}	0.38113	0.32211

TABLE VII

OPTIMIZED DEGREE DISTRIBUTIONS OF LDPC CODES OF RATE $R = \frac{3}{4}$ FOR CONCATENATION WITH NLTC OF RATE $\frac{1}{3}$ OF TABLE IV

λ_2	0.35789	ρ_3	0.02806
λ_3	0.12842	ρ_4	0.07483
λ_7	0.03020	ρ_9	0.15047
λ_8	0.04853	ρ_{10}	0.04484
λ_{11}	0.03455	ρ_{39}	0.54824
λ_{12}	0.40041	ρ_{40}	0.15356

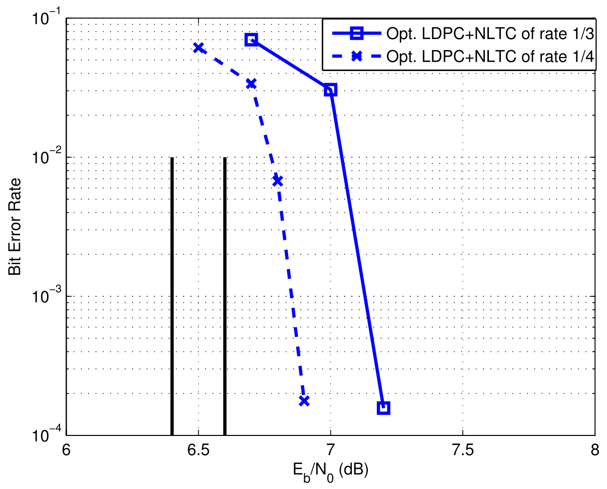

Fig. 8. Bit error rate performance of optimized LDPC code of Table VI concatenated with NLTCs of rates $R = \frac{1}{3}$ and $\frac{1}{4}$, memory ($M = 4$) and ones' density $p = \frac{3}{4}$. Outer LDPC code is of rate $R = 0.823$ and block-length 100k.

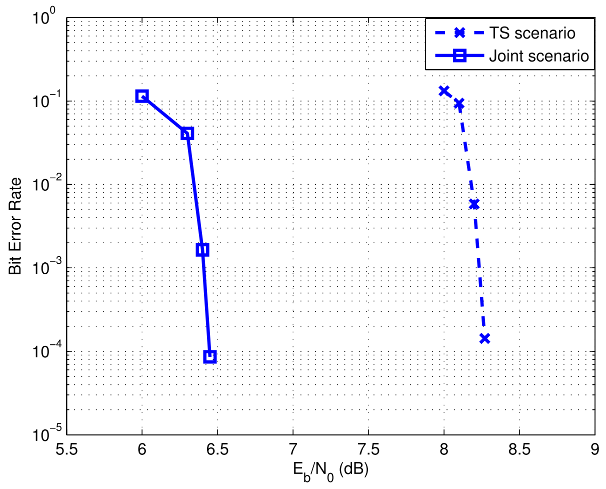

Fig. 9. Comparison of bit error rate performance between joint transfer and time switching scenarios.

rate $R = 0.823$ for each. The optimization procedures for these two cases result in the degree distributions which are given in Table VI. Using Monte Carlo simulations, the threshold for the concatenation of the optimized codes with NLTCs of rate $\frac{1}{3}$ and $\frac{1}{4}$ are calculated as 6.6 dB and 6.4 dB, respectively, while the information theoretic limits are 5.42 dB and 5.14 dB. Hence the proposed scheme operates at about 1.2 dB from the limit in both cases. We also generate sample parity check matrices and report the resulting BER performances in Fig. 8. We observe that the gap to the information theoretic limits are larger in this example compared to the previous ones. We attribute this to the following: for high rate LDPC codes, there are check nodes of large degrees and the Gaussian approximation for the outgoing messages of these check nodes may be less accurate, and since we use the Gaussian approximation in the EXIT chart analysis for the code design, the larger gaps to the information theoretic limits result.

Finally, we compare the performance of the designed codes for joint energy and information transfer with that of classical linear codes used with time switching (TS). For the time switching alternative, sending both information and energy half the time using on-off signaling, and sending only "1" for the other half will result in a ones' density of $p = \frac{3}{4}$. In this example, for the TS scenario, the degree distribution of the outer LDPC code is obtained from [14], and the parity check matrices are obtained with tools in [15]. An irregular LDPC code of rate $\frac{1}{2}$ with block-length 100k is used for the TS option, hence the overall information transfer rate is $\frac{1}{4}$. As an example of the proposed design, a nonlinear code realized with concatenation of the NLTC of rate $\frac{1}{3}$, memory $m = 4$ and ones' density $p = \frac{3}{4}$ from the first example with an optimized outer LDPC code of rate $\frac{3}{4}$ (with the degree distribution in Table VII) and block-length 100k (resulting in the same overall rate of $\frac{1}{4}$) is used for joint energy and information transfer. Fig. 9 shows the bit error rate performance results which clearly show that using the designed nonlinear codes for joint energy and information transfer provides significant SNR benefits compared to linear codes in a time switching mode.

VII. CONCLUSIONS

In this paper, a coding scheme based on concatenation of a nonlinear trellis code with an outer LDPC code for joint energy and information transfer is proposed. In order to design the NLTCs, an algorithm based on maximizing the minimum distance of the code is provided. Also, necessary and sufficient conditions for catastrophicity of nonlinear trellis codes are obtained; and, in order to avoid such catastrophic codes, each designed NLTC is connected to a corresponding linear convolutional code. This allows for the use of simpler conditions for checking for the catastrophicity of the designed NLTC.

Furthermore, we employ EXIT charts to design the outer LDPC codes while fixing the inner NLTCs. Several design examples are provided, and their performances are evaluated. The results indicate that the designed codes operate at about 0.8 dB away from the information theoretic limits, and they outperform both regular LDPC codes and optimized irregular LDPC codes for AWGN channels when used with NLTCs for joint energy and information transfer. In order to have a practical code design, we employ small degrees in both check and variable nodes, however, we expect that by using larger degrees the gap to the information theoretic limits can be decreased. Furthermore, our results also show that the designed codes outperform the alternative of using classical linear block codes with time switching and the reference schemes of concatenating LDPC codes with nonlinear memoryless mappers considerably.

Appendix A
Proof of Theorem 1

The sufficiency part is straightforward. We assume that either one of these two conditions is true and then we find two different paths of infinite length with finite number of differences in the output and infinite number of differences in the input sequence. First note that starting from any state one can find a path of length M to any other state in the state diagram (due to the specific structure of the evolution of memory). Next if condition 1 is true then we consider the two different states of the cycle as state a and b, or if condition 2 is true we consider state a from the first cycle and state b from the second one. We can find two different paths with starting part of length M from state zero to state a or b and final state transitions the other way around, the middle part can be arbitrarily long consisting of traversing around the cycle corresponding to state a or b. The starting and ending parts will result in finite number of differences in the output, and for the middle part which is arbitrarily long the output sequence is the same for both paths but the input sequences have arbitrarily large number of differences.

For the necessity part we know that we have two input sequences with infinite number of differences with corresponding output sequences of finite number of differences. We separate the parts in which two input sequences are different but their corresponding outputs are the same. Two situations can occur: 1- There are infinite number of such finite-length subsequences of different inputs that have same corresponding output. 2- There is at least one infinite length subsequence of different inputs for each sequence that have same corresponding output. For each case we can argue that at least one of the catastrophicity conditions must be satisfied.

Case 1: Since the trellis has a finite number of states and a finite number of branches, the number of different paths with the same output in the state diagram is finite, this means that at least one of them must repeat infinitely many times in each sequence, and in order to repeat a path we need to go back to its starting point which means we have traversed around separate cycles infinitely many times with each one of the two sequences and these separate cycles having different input sequences but the same output sequence.

Case 2: The infinite length subsequence needs to contain cycles (since trellis has finite number of branches) that are repeating infinitely many times and have different inputs, hence, there are separate cycles in each subsequence with different inputs but same output (separate cycles can be due to a single sequence of paths starting from different states).

Appendix B
Proof of Theorem 2

For the first step of induction, we select a branch corresponding to an even state $0 \le S \le 2^h - 1$ with input $u = 0$, so $S = \underbrace{s_M s_{M-1} \ldots s_{h+1}}_{\text{all are zero}} s_h \ldots s_2 \underbrace{s_1}_{\text{zero}}$ and $G(S)_{u=0} = s_h s_{h-1} \ldots s_2(0)$ (according to second step of the grouping algorithm) and the output for this branch is $Output(S_i)_{u=0} = s_h s_{h-1} \ldots s_2 0$ (according to the assumed label assignment) which can be exactly obtained by the generators given that $u = 0$, $s_1 = 0$ and s_{h+1} to s_M are all zeros. For the k^{th} bit: $G(D).U(D) = (1 + \sum_{i=h}^{M-1} a_{2,i} D^{M-i} + D^{M-k+1} + D^M).(\underbrace{u D^M + s_M D^{M-1} + \ldots + s_{h+1} D^h}_{0} + s_h D^{h-1} + \ldots + s_k D^{k-1} + \ldots + s_2 D + \underbrace{s_1}_{0}) = s_k D^M$ for all $k = 2, \ldots, h$.

For the second step, assuming that for all the branches corresponding to the states $0 \le S \le 2^i - 1$ the claim is true, then we will show that it is true for all the branches corresponding to states $2^i \le S \le 2^{i+1} - 1$. Hence, we select a branch corresponding to an even state S with input $u = 0$ where $2^i \le S \le 2^{i+1} - 1$, so $S = \underbrace{s_M s_{M-1} \ldots s_{i+2}}_{\text{all are zero}} \underbrace{s_{i+1}}_{\text{one}} s_i \ldots s_2 \underbrace{s_1}_{\text{zero}}$. For $S = 2^i$ by the assumption $Output(S = 2^i)_{u=0} = a_{h,i} a_{h-1,i} \ldots a_{1,i}$ which can be exactly obtained by the generator polynomials since s_{i+1} is one and all the other bits are zero. Also

$$G(S = 2^i)_{u=0} = \begin{cases} a_{h,i} a_{h-1,i} \ldots a_{2,i} (a_{1,i}) & \text{if } a_{1,i} = 0, \\ \overline{a_{h,i} a_{h-1,i} \ldots a_{2,i}} (a_{1,i}) & \text{if } a_{1,i} = 1. \end{cases}$$
$$G(A_{2^{i-1}}) = a_{h,i} a_{h-1,i} \ldots a_{2,i} \quad \text{if } a_{1,i} = 0,$$
$$G(A^*_{2^{i-1}}) = \overline{a_{h,i} a_{h-1,i} \ldots a_{2,i}} \quad \text{if } a_{1,i} = 1$$

where $\overline{()}$ is the one's complement operator.

Now we want to prove the case for $S = 2^i + 2j$ where $1 \le j \le 2^{i-1} - 1$. If $G(A_j) = g' = g'_{h-1} \ldots g'_1$ then $Output(S = 2j)_{u=0} = g'_{h-1} \ldots g'_1 0$, following the rule in step 3 of grouping algorithm we can see that

$$G(A_{2^{i-1}+j}) = (g'_{h-1} \oplus a_{h,i}) \ldots (g'_1 \oplus a_{2,i}) \quad \text{if } a_{1,i} = 0,$$
$$G(A^*_{2^{i-1}+j}) = (g'_{h-1} \oplus \overline{a_{h,i}}) \ldots (g'_1 \oplus \overline{a_{2,i}}) \quad \text{if } a_{1,i} = 1,$$

and hence the output will be

$$Output(S = 2^i + 2j)_{u=0} = (g'_{h-1} \oplus a_{h,i}) \ldots (g'_1 \oplus a_{2,i}) a_{1,i} \quad \text{if } a_{1,i} = 0,$$
$$Output(S = 2^i + 2j)_{u=0} = \overline{(g'_{h-1} \oplus \overline{a_{h,i}}) \ldots (g'_{h-1} \oplus \overline{a_{2,i}})} a_{1,i}$$
$$= (g'_{h-1} \oplus a_{h,i}) \ldots (g'_1 \oplus a_{2,i}) a_{1,i} \quad \text{if } a_{1,i} = 1,$$

which is consistent with the output obtained by generator polynomials. If $G(A^*_j) = g' = g'_{h-1} \ldots g'_1$ then

$Output(S = 2j)_{u=0} = \overline{g'_{h-1}...g'_1 1} = \overline{g'_{h-1}}...\overline{g'_1}1$, Now following the rule in step 3 of grouping algorithm we can see that

$$\begin{aligned} G(A^*_{2^{i-1}+j}) &= (g'_{h-1} \oplus a_{h,i})...(g'_1 \oplus a_{2,i}) \quad \text{if } a_{1,i} = 0, \\ G(A_{2^{i-1}+j}) &= (g'_{h-1} \oplus \overline{a_{h,i}})...(g'_1 \oplus \overline{a_{2,i}}) \quad \text{if } a_{1,i} = 1, \end{aligned}$$

and hence the output will be

$$\begin{aligned} Output(S = 2^i + 2j)_{u=0} &= \overline{(g'_{h-1} \oplus a_{h,i})...(g'_1 \oplus a_{2,i})a_{1,i}} \\ &= (\overline{g'_{h-1}} \oplus a_{h,i})...(\overline{g'_1} \oplus a_{2,i})\overline{a_{1,i}} \quad \text{if } a_{1,i} = 0, \\ Output(S = 2^i + 2j)_{u=0} &= (g'_{h-1} \oplus \overline{a_{h,i}})...(g'_{h-1} \oplus \overline{a_{2,i}})\overline{a_{1,i}} \\ &= (\overline{g'_{h-1}} \oplus a_{h,i})...(\overline{g'_1} \oplus a_{2,i})\overline{a_{1,i}} \quad \text{if } a_{1,i} = 1, \end{aligned}$$

which is consistent with the output obtained by generator polynomials.

References

[1] X. Lu, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "Wireless networks with RF energy harvesting: A contemporary survey," *IEEE Commun. Surv. Tuts.*, vol. 17, no. 2, pp. 757–789, Second Quarter 2015.

[2] L. R. Varshney, "Transporting information and energy simultaneously," in *Proc. IEEE Int. Symp. Inf. Theory*, Toronto, ON, Canada, Jul. 2008, pp. 1612–1616.

[3] A. M. Fouladgar, O. Simeone, and E. Erkip, "Constrained codes for joint energy and information transfer," *IEEE Trans. Commun.*, vol. 62, no. 6, pp. 2121–2131, Jun. 2014.

[4] A. Tandon, M. Motani, and L. R. Varshney, "On code design for simultaneous energy and information transfer," in *Proc. Inf. Theory Appl. Workshop (ITA)*, San Diego, CA, USA, Feb. 2014, pp. 1–6.

[5] R. Zhang and C. K. Ho, "MIMO broadcasting for simultaneous wireless information and power transfer," *IEEE Trans. Wireless Commun.*, vol. 12, no. 5, pp. 1989–2001, May 2013.

[6] X. Zhou, R. Zhang, and C. K. Ho, "Wireless information and power transfer: Architecture design and rate-energy tradeoff," *IEEE Trans. Commun.*, vol. 61, no. 11, pp. 4757–4767, Nov. 2013.

[7] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. New York, NY, USA: Wiley, 2006.

[8] M. Franceschini, G. Ferrari, R. Raheli, and A. Curtoni, "Serial concatenation of LDPC codes and differential modulations," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 9, pp. 1758–1768, Sep. 2005.

[9] M. Griot, A. I. V. Casado, W.-Y. Weng, H. Chan, J. Wang, and R. D. Wesel, "Nonlinear trellis codes for binary-input binary-output multiple-access channels with single-user decoding," *IEEE Trans. Commun.*, vol. 60, no. 2, pp. 364–374, Feb. 2012.

[10] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1995.

[11] J. P. Odenwalder, "Optimal decoding of convolutional codes," Ph.D. dissertation, Dept. Syst. Sci., School Eng. Appl. Sci. Univ. California, Los Angeles, Los Angeles, CA, USA, 1970.

[12] M. Dabirnia and T. M. Duman, "Nonlinear code design for joint energy and information transfer," in *Proc. IEEE Int. Conf. Commun. (ICC)*, London, U.K., Jun. 2015, pp. 4247–4252.

[13] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[14] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[15] Doxygen. (Jul. 23, 2014). *IT++ Documentation*. [Online]. Available: http://itpp.sourceforge.net/4.3.1/

Mehdi Dabirnia (S'15) received the B.S. and M.S. degrees in electrical engineering from the University of Tehran, Tehran, Iran, in 2007 and 2010, respectively. He joined the Electrical and Electronics Engineering Department, Bilkent University, in Spring 2013, where he is currently pursuing the Ph.D. degree. His research interests are in interference channels, joint energy and information transfer, and energy harvesting communications.

Tolga M. Duman (S'95–M'98–SM'03–F'11) received the B.S. degree from Bilkent University, Turkey, in 1993, and the M.S. and Ph.D. degrees from Northeastern University, Boston, in 1995 and 1998, respectively, all in electrical engineering. Prior to joining Bilkent University in 2012, he was with the Electrical Engineering Department, Arizona State University, first as an Assistant Professor (1998–2004), an Associate Professor (2004–2008), and a Professor (2008–2015). He is currently a Professor with the Electrical and Electronics Engineering Department, Bilkent University, and an Adjunct Professor with the School of ECEE, Arizona State University. His current research interests are in systems, with particular focus on communication and signal processing, including wireless and mobile communications, coding/modulation, coding for wireless communications, data storage systems, and underwater acoustic communications.

Dr. Duman is a recipient of the National Science Foundation CAREER Award and the IEEE Third Millennium Medal. He served as an Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS (2003–2008), the IEEE TRANSACTIONS ON COMMUNICATIONS (2007–2012), and the IEEE ONLINE JOURNAL OF SURVEYS AND TUTORIALS (2002–2007). He has been the Coding and Communication Theory Area Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS (2011–present) and an Editor of *Physical Communications* (Elsevier; 2010–present).