Serial Concatenation of Reed Muller and LDPC Codes with Low Error Floor

Xin Xiao¹, Mona Nasseri², Bane Vasić ³ and Shu Lin⁴

Abstract—In this paper, we propose a concatenated coding scheme involving an outer Reed-Muller (RM) code and an inner finite field low-density parity check (LDPC) code of medium length and high rate. It lowers the error floor of inner finite field LDPC code. This concatenation scheme offers flexibility in design and is easy to implement. In addition, the decoding works in a serial turbo manner and has no harmful trapping sets of size smaller than the minimum distance of the outer code. The simulation results indicate that the proposed serial concatenation can eliminate the dominant trapping sets of the inner finite field LDPC code.

I. INTRODUCTION

Long LDPC codes with Belief-Propagation (BP) based iterative decoding algorithms, such as Sum Product Algorithm (SPA) [1] and Min Sum Algorithm (MSA) [2], have been shown to approach Shannon capacity [3][4] in various channels including those used to realistically model data storage and communications systems. However, LDPC codes of medium length have sub-optimal performance under BP algorithms due to the presence of short cycles in Tanner graph and the nature of iterative decoding. There have been many research efforts on LDPC code design to get good error performance, either on algebraic construction [5][6], or graphical construction [7][8]. The Finite Field LDPC (FF-LDPC) codes [6] are one type of quasi-cyclic (QC) LDPC codes designed based on finite field, which have good threshold performance under AWGN channel. Furthermore, their constructions are flexible with regards to code length and rate. However, the Tanner graphs of FF-LDPC codes contain many small “trapping sets” if the column weights of their parity-check matrices are too small [9]. These harmful substructures will generate error floor in high SNR region and limit their applications that require high reliability. One practical way to reduce error floor is to utilize concatenation.

Concatenations of LDPC codes with different error correction codes have attracted significant attention recently. Eslami et al [10] concatenated a long polar code of length 32768 as outer code on a long inner LDPC code of length 34493 for Optical Transport Network, where the outer polar code played a role to lower the error floor. Nasseri et al [11] proposed a concatenation of a Finite Geometry (FG-) LDPC code of length 1057 as outer code and a FF-LDPC code of length 8000 as inner code, where the outer FG-LDPC code acted to eliminate the trapping sets of size smaller than its large minimum distance. In addition, short LDPC codes have been considered as outer code in concatenation with a finite length inner polar code. For finite length polar codes, the bit-channels are not completely polarized, thus there exist intermediate channels that are neither fully “noiseless” nor fully “noisy”. Considering the existence of intermediate channels, Guo et al [12] suggested to concatenate an inner polar code of length 4096 with an outer Tanner code of length 155, where the outer Tanner code was applied on the intermediate channels to provide extra protection.

Different from previous concatenated schemes where short or long length LDPC codes were considered, in this paper, we investigate the concatenation of a short RM code [13] and a FF-LDPC code of medium length. It has been shown that RM code can approach capacity of an erasure channel [14]. Furthermore, Arikan revealed the close relationship of polar code and RM code in [15]. By numerical simulation results and theoretical analysis, Hussami et al showed that RM codes perform worse under successive cancellation and iterative decoding than polar codes, but they outperform polar codes under MAP decoding due to their better minimum distance [16]. Moreover, the MAP decoding has neither trapping set nor error floor problem. Inspired by the excellent “error floor free” performance of the RM codes under MAP decoding, for high rate, medium length FF-LDPC codes, we propose a serial concatenation to lower the error floor of FF-LDPC codes under BP algorithms. The inner code is a FF-LDPC code of high rate and medium length, and the outer code is a short RM code. We encode the outer RM code in a cyclic way [17], which can be easily implemented by Linear Feedback Shift Registers (LFSR). By shortening or extending some outer RM blocks, the encoding of inner FF-LDPC code is fairly flexible. Another difference is the decoding process, where all mentioned concatenation schemes apply BP algorithm to decode both inner and outer codes. We still use the MSA to decode the inner FF-LDPC code for global errors, but apply BCJR algorithm [18] to decode the outer RM code to break trapping sets of inner FF-LDPC code. These two decoders exchange extrinsic messages in a serial turbo fashion. As
the BCJR decoder brings about additional computational complexity, to minimize the complexity cost, we propose a recursive algorithm to reform the generator matrix of the outer RM code, whose trellis has the minimal state space dimension. Furthermore, since the size of raw error pattern is small in the error floor region of FF-LDPC codes, few BCJR decodings are needed. Therefore, there is flexibility to trade the computational decoding complexity off against the latency by deciding the number of BCJR decoders. The simulation results show that the serial turbo concatenation with RM code efficiently eliminates the dominant trapping sets and lowers the error floor of FF-LDPC code.

The rest of the paper is organized as follows. Section II introduces the structures of RM code and FF-LDPC code. Section III presents the proposed serial turbo concatenation of RM code and FF-LDPC code. Section VI shows an example of proposed RM-FF-LDPC concatenation and the simulation results in terms of FER performance. Conclusions are finally given in Section V.

II. STRUCTURE OF CONCATENATION COMPONENTS

In this section, we give a brief description of the structures of the outer RM code and inner FF-LDPC code. We also introduce the Trellis-Oriented Form (TOF) [19] of generator matrix of RM code, whose trellis has the minimal state space dimension. In the end, we propose a recursive algorithm to generate the TOF generator matrix of the outer RM code. Up to our best knowledge, this is the first algorithm designed for a RM code to find its TOF generator matrix.

A. Reed Muller code

Reed Muller Codes are one class of linear block codes that can correct multiple errors. Given integer $m$ and $r$ where $0 \leq r \leq m$, a binary $r$th order RM code, denoted by $\text{RM}(r,m)$, has the following parameters:

1) Code length: $n = 2^m$
2) Dimension: $k(r,m) = \sum_{i=0}^{r} \binom{m}{i}$
3) Minimum distance: $d_{\text{min}} = 2^{m-r}$.

We use $(\mathbf{u} | \mathbf{u} + \mathbf{v})$ construction to demonstrate the structure of RM codes. For given length $2^m$, $m \geq 2$, the $r$th order RM code can be constructed by following formula recursively:

$$\text{RM}(r,m) = \left\{ (\mathbf{u} | \mathbf{u} + \mathbf{v}) : \mathbf{u} \in \text{RM}(r,m-1), \mathbf{v} \in \text{RM}(r-1,m-1) \right\}$$

Hence the generator matrix of $\text{RM}(r,m)$ is given by the following recursive formula:

$$\mathbf{G}(r,m) = \begin{bmatrix}
\mathbf{G}(r,m-1) & \mathbf{G}(r,m-1) \\
0 & \mathbf{G}(r-1,m-1)
\end{bmatrix}$$  \hspace{1cm} (1)

Where $\mathbf{G}(0,m) = [1, 1, \ldots, 1]$, $\mathbf{G}(m,m) = \begin{bmatrix} \mathbf{G}(m-1,m) \end{bmatrix}$, and $\mathbf{g} = [0, \ldots, 0, 1]$. The parity check matrix of $\text{RM}(r,m)$ is $\mathbf{H}_{rm}(r,m) = \mathbf{G}(m-r-1,m)$. We decode $\text{RM}(r,m)$ code by BCJR decoding algorithm on trellis diagrams. To minimize the state space dimension of a trellis, we put $\mathbf{G}(r,m)$ in Trellis-Oriented Form, based on the recursive structure of $\mathbf{G}(r,m)$. The TOF is defined by the following two properties:

1) The leading ‘1’ of each row appears in a column before the leading ‘1’ of any row below it.
2) No two rows have their trailing ‘1’s in the same column.

where the leading ‘1’ and trailing ‘1’ denote the first and last nonzero components in a vector, respectively. Based on the following two lemmas, we propose a recursive construction of TOF of $\mathbf{G}(r,m)$ in Algorithm 1, where $(\mathbf{M})_{U,V}$ is the submatrix of matrix $\mathbf{M}$, whose rows are indexed by $U$ and columns are indexed by $V$.

**Lemma 1:** The generator matrix derived by $(\mathbf{u}|\mathbf{u} + \mathbf{v})$ construction satisfies the first property of TOF.

**Proof:** We use induction on the number $k = r + m$. For $k = 1$, $r = 0$, $m = 1$, $\mathbf{G}(r,m) = \mathbf{G}(0,1) = [1,1]$ satisfies the first property. Supposed that for all $k \leq s, s \geq 1$, any generator matrix $\mathbf{G}(r,m)$ derived by $(\mathbf{u}|\mathbf{u} + \mathbf{v})$ construction with $r, m$ (where $r + m = k$) satisfies the first property. Then for $r + m = k = s + 1$,

$$\mathbf{G}(r,m) = \mathbf{G}(s + 1 - m, m) = \begin{bmatrix} \mathbf{G}(s + 1 - m, m - 1) & \mathbf{G}(s + 1 - m, m - 1) \\
0 & \mathbf{G}(s - m, m - 1)\end{bmatrix}$$

Both $\mathbf{G}(s + 1 - m, m - 1)$ and $\mathbf{G}(s - m, m - 1)$ satisfy the first property since $s + 1 - m + 1 - m = s$ and $s - m + m - 1 = s - 1$. Therefore, $\mathbf{G}(r,m) = \mathbf{G}(s + 1 - m, m)$ also satisfies the first property. By induction, for $\forall r,m$, $\mathbf{G}(r,m)$ satisfies the first property.

**Algorithm 1** Recursive construction of TOF of $\mathbf{G}(r,m)$ based on $(\mathbf{u}|\mathbf{u} + \mathbf{v})$ structure

1: Input: $r,m$
2: Construct $\mathbf{G}(r,m)$ by (1). Let $\mathbf{G}_1 = \mathbf{G}(r,m-1), \mathbf{G}_2 = \mathbf{G}(r-1,m-1)$, in $\mathbf{G}(r,m)$, respectively.
3: Find the indices of rows (denoted by $S$) corresponding to $\mathbf{G}_2$ in $\mathbf{G}_1$.
4: Set $(\mathbf{G}_1)_{S,[1:2^{m-1}-1]} = \mathbf{G}_2$, $(\mathbf{G}_1)_{S,[2^{m-1}+1:2^{m}]} = 0$.
5: Put the two $\mathbf{G}_2$‘s into TOF by Algorithm 1.
6: Use Gaussian Elimination to update the rows indexed by $\{k(r,m-1) - 1 + 1, \ldots, k(r,m-1)\}$ based on the positions of trailing ‘1’.
7: Output: TOF of $\mathbf{G}(r,m)$

**Lemma 2:** Algorithm 1 generates the TOF of $\mathbf{G}(r,m)$.

**Proof:** Based on Lemma 1, the order of rows in $\mathbf{G}(r,m)$ is fixed. It is sufficient to update the rows based...
on the positions of trailing ‘1’. Since
\[ G(r, m - 1) = \begin{bmatrix} G(r - 1, m - 1) \\ \Delta_{(r/r-1, m-1)} \end{bmatrix} \]
where \( \Delta_{(r/r-1, m-1)} \) consists of rows in \( G(r, m - 1) \) but not in \( G(r - 1, m - 1) \), it is easy to check that Algorithm 1 generates the TOF of \( G(r, m) \).

**B. Finite Field LDPC code**

There are lots of methods to design a FF-LDPC code. For completeness, here we introduce a *Latin Square based adaptable construction* [6].

Let \( q \) be a prime and consider the prime field \( GF(q) \) with \( q \) elements. Let \( \alpha \) be a primitive element of \( GF(q) \), then \( \alpha^0 = \alpha^{-q-1} = 1, \alpha, \alpha^2, \ldots, \alpha^{q-2} \) give all the nonzero elements of \( GF(q) \). Let \( \alpha^{-\infty} \) denote the zero element in \( GF(q) \). We consider a \( q \times q \) Latin Square over \( GF(q) \) with all zero entries on the main diagonal as follows:

\[
B = \begin{bmatrix}
\alpha^{-\infty} & \alpha^{-\infty} & \alpha^0 & \alpha^{-\infty} & \vdots & \alpha^{q-2} & \alpha^{-\infty} \\
\alpha^{-\infty} & \alpha^0 & \alpha^{-\infty} & \alpha^0 & \alpha^{-\infty} & \vdots & \alpha^{q-2} \\
\alpha^{-\infty} & \alpha^{-1} & \alpha^{-1} & \alpha^{0} & \alpha^{q-2} & \alpha^{-1} & \alpha^{q-2} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\alpha^{-\infty} & \alpha^{q-2} & \alpha^0 & \alpha^{-q-2} & \alpha^{q-2} & \alpha^{-q-2} & \alpha^{-2} \end{bmatrix}
\]

(2)

This matrix has the following structural properties: each \( 2 \times 2 \) submatrix of \( B \) is non-singular (this is called \( 2 \times 2 \) submatrix (SM) constraints); all the entries in a row (or a column) are distinct elements of \( GF(q) \). Each row (or each column) contains exactly one zero element. Let \( \gamma, \rho \) be two positive integers with \( 1 \leq \gamma, \rho \leq q - 1 \). Take a \( \gamma \times \rho \) submatrix from \( B \) and denote it by \( B(\gamma, \rho) \). Then, \( B(\gamma, \rho) \) will satisfy \( 2 \times 2 \) SM constraint as well. This \( B(\gamma, \rho) \) will be used as the base matrix, with each entry dispersed by its *circulant permutation matrix* (CPM).

For \( 0 \leq i < q - 1 \), the CPM of the nonzero element \( \alpha^i \) in \( GF(q) \) is a matrix representation with size \( (q - 1) \times (q - 1) \), denoted by \( A(\alpha^i) \), whose top row has a single 1-component at the position \( i \). For zero element in \( GF(q) \), the CPM is a zero matrix of size \( (q - 1) \times (q - 1) \) and denoted by \( A(\alpha^{-\infty}) \). The mapping defined by CPM is bijective. Each entry in \( B(\gamma, \rho) \) is \( [b_{i,j}]_{1 \leq i, j \leq \rho} \) is dispersed into its corresponding CPM, i.e., if \( b_{i,j} = \alpha^i \), \( b_{i,j} \) is replaced by \( A(\alpha^i) \). This generates \( H(\gamma, \rho) \), a \( (q - 1) \gamma \times (q - 1) \rho \) matrix over \( GF(2) \) with column and row weights \( \gamma \) and \( \rho \), respectively.

In this matrix product, the nonzero entries in \( B(\gamma, \rho) \) at the locations corresponding to zero entries in \( Z(\gamma, \rho) \) are replaced (or masked) by zeros. The replacement of \( \mu \) nonzero entries in \( B(\gamma, \rho) \) is equivalent to replace \( \mu \) CPMs by \( \mu \) zero matrices at the corresponding locations in \( H(\gamma, \rho) \), i.e., removing \((q-1)\mu \) edges from its Tanner graph, thus \( H(\gamma, \rho) \) may have a smaller number of short cycles, or a larger girth. The updated masked matrix \( B_{mask}(\gamma, \rho) \) is then used as the base matrix.

With different choices of submatrix \( B(\gamma, \rho) \) and masking matrix \( Z(\gamma, \rho) \) of size \( \gamma \times \rho \), it is possible to construct a family of QC-FF-LDPC codes of various lengths and rates whose Tanner graphs have girth at least 6. Hence, the construction is very flexible.

**III. SERIAL TURBO CONCATENATION OF RM CODE AND LDPC CODE**

In this section, we present a serial turbo concatenated coding scheme for FF-LDPC codes of high rate and medium length.

In proposed concatenation, the outer code is \( r \)-th order RM code \( RM(r, m) \). The inner code is FF-LDPC code \( C_{FF} \) given by the null space of a parity-check matrix \( H_{FF} \), which is constructed based on Latin Square over a finite field \( GF(q) \), as described in Section II. Fig. 1 shows the serial turbo concatenation framework of RM code and FF-LDPC code.

**A. Encoding**

![Fig. 1. Serial Turbo Concatenation of RM code and FF-LDPC code](image)

Let \( t = \left\lfloor \frac{(r-\gamma)(q-1)}{2^m} + \frac{1}{2} \right\rfloor \) and \( u \) be a sequence of \( tk(r, m) \) binary information symbols, where \( \lfloor z \rfloor \) is the largest integer no greater than \( z \). We encode this information sequence in two stages. In the first stage, we divide \( u \) into \( t \) subsequences, denoted by \( u_0, u_1, \ldots, u_{t-1} \), each consisting of \( k(r, m) \) binary information symbols. For \( 0 \leq i \leq t-1 \), each subsequence \( u_i \) is encoded into a codeword in \( RM(r, m) \). In particular, as shown in Fig. 1, instead of directly using \( G(r, m) \), we use the cyclic version of \( G(r, m) \) as the generator matrix (denoted by \( G_{cyc}(r, m) \)) to simplify the encoding implementation [17]. For each \( u_i \), the encoded sequence \( c_i \) is a permutation of a codeword in original \( RM(r, m) \). Thus applying an interleaver \( P \) on \( c_i \) produces a codeword \( w_i \) in \( RM(r, m) \), where \( P \) satisfies \( P(G_{cyc}(r, m)) = G(r, m) \).
More specifically, $G_{cyc}(r, m)$ can be generated by adding one overall parity check column to the generator matrix $G_0(r, m)$ of a cyclic code as follows:

$$G_{cyc}(r, m) = [0, 1, \ldots, 1_{2^m-1}] = [u_0 | G_0(r, m)]$$

where

$$G_0(r, m) = \left[ \begin{array}{c} v_1^T \\
\vdots \\
v_{\delta_0}^T v_{\delta_1}^T \ldots v_{\delta_{m-1}}^T \\
\end{array} \right],$$

$$v_i = (1, 1, \ldots, 1), \quad v_i = (a_{i, 0}, a_{i, 1}, \ldots, a_{i, 2^m-2}), \quad \alpha^i = \sum_{i=0}^{m-1} a_{i, j} \alpha^i$$

$a_{i, j} \in \{0, 1\}, 0 \leq i \leq m - 1, 0 \leq j \leq 2^m - 2$.

$I_i$ is the $i$-th column in $G_{cyc}(r, m)$, $u_0$ is the overall parity check column and $\alpha$ is the primitive element of $GF(2^m)$. $G_0(r, m)$'s generator polynomial $g(X)$ can be easily found by the following theorem:

**Theorem 1:** Let $h = \delta_0 + \delta_1 \cdot 2 + \ldots + \delta_{m-1} \cdot 2^{m-1}, W_2(h) = \sum_{k=0}^{m-1} \delta_k$. The generator polynomial $g(X)$ of $G_0(r, m)$ of length $2^m - 1$ has $\alpha^h$ as a root if and only if [17]

$$0 < W_2(h) \leq m - r - 1. \quad (3)$$

Once we obtain $g(X)$, we can simply use LFSR to encode $u$ into $c = (c_0, c_1, \ldots, c_{2^m-1})$. It is clear that the subspace consisting of first $m + 1$ rows of $G_0(r, m)$ is in the form of

$$\left[ \begin{array}{cccc} 1 & 1 & 1 & \cdots \\
\alpha & (\alpha^2) & (\alpha^3) & \cdots (\alpha^{2^m-2}) \\
\end{array} \right],$$

where $(\alpha^j) = (a_{0, j}, a_{1, j}, \ldots, a_{m-1, j})^T$, thus $P$ can be derived based on the one-to-one correspondence of $j$ and $\alpha^j$ as follows:

$$P (l_0) = l_0, P (l_i) = 1_{\alpha^{i-1}} + 1 \leq i \leq 2^m - 1. \quad (4)$$

Applying $P$ on $c$ symbolwise generates a codeword sequence $w = (w_0, w_1, \ldots, w_{t-1})$ of length $t \cdot 2^m$.

In the second stage, $w$ is used as the information sequence and encoded into a codeword $v$ of length $\rho(q - 1)$ by the systematic form of generator matrix of $C_{FF}$. To fit the dimension of $C_{FF}$, we extend $w$ by adding zeros if $t \cdot 2^m \leq (\rho' - \gamma)(q - 1)$, otherwise we shorten some blocks of $w$. This completes the second encoding stage of the concatenation of the outer RM code and inner FF-LDPC code. Especially, when $t \cdot 2^m = (\rho' - \gamma)(q - 1)$, the equivalent parity check matrix of serial Concatenation $H_{con}$ has the following form:

$$H_{con} = \begin{bmatrix} \text{diag} (H(r, m), H(r, m), \ldots, H(r, m)) & 0 \\
H_{FF, sys, I} & H_{FF, sys, P} \end{bmatrix}$$

In $H_{con}$, $O$ is the zero matrix and $\text{diag}(H(r, m), H(r, m), \ldots, H(r, m))$ is the parity check matrix of first stage encoding. It is a diagonal array of size $t \times t$ with $t$ copies of $H(r, m)$ lying on its main diagonal and zero elsewhere. The lower submatrix of $H_{con}$ is the systematic form of the parity-check matrix $H_{FF}$, where $H_{FF, sys, I}$ and $H_{FF, sys, P}$ correspond to the information part and parity part of $C_{FF}$, respectively.

**B. Decoding**

The decoding of serial concatenation of RM code and FF-LDPC code is carried out in two phases. Supposed that the received vector from channel is $r$. Let $I_{max}$ be the maximum number of iterations of inner decoding algorithm, $I_I$ be the maximum number of iterations of turbo decoding, $\hat{x}$ be the decoded vector, $S_{FF}$ and $S_{rm}$ be the syndrome of inner and outer decoding algorithm, respectively. Denote the extrinsic Log Likelihood Ratio (LLR) from inner (outer) decoding algorithm as outer (inner) decoding algorithm by $LLR_{inn}$ ($LLR_{out}$). Let the posteriori values of inner (outer) decoding algorithm be $LLR_{inn}$ ($LLR_{out}$). Present the priori values of inner (out) decoding algorithm by $LLR_{inn}$ ($LLR_{out}$).

In the initialization, $LLR_{inn} = 0$ as we assume that the information bits are equally likely.

In the first decoding phase, $r$ is decoded based on the parity-check matrix $H_{FF}$ by an iterative message passing algorithm, such as SPA or MSA, which takes the $r$ and $LLR_{inn}$ as inputs. At the end of each iteration, based on $\hat{x}$ and $H_{FF}$, we check whether the syndrome $S_{FF}$ is equal to zero. If $S_{FF} = 0$, then $\hat{x}$ is a codeword in $C_{FF}$. If $S_{FF} \neq 0$, we continue the inner iterative decoding until either $S_{FF} = 0$ or $I_{max}$ is reached. When the inner iterative decoding algorithm completes, $\hat{x}$ and $LLR_{inn}$ are passed to outer code decoder. The decoded sequence of information symbols $\hat{w}$ is extracted from $\hat{x}$ and partitioned into $t$ vectors: $\hat{w}_0, \hat{w}_1, \ldots, \hat{w}_{t-1}$. For each $\hat{w}_i, 0 \leq i \leq t - 1$, we check whether its syndrome $S_{rm}$ of $H(r, m)$ is zero or not. If $S_{FF}$ and all $t$ syndromes are zero, then $\hat{w}_0, \hat{w}_1, \ldots, \hat{w}_{t-1}$ are codewords in outer RM($r, m$). The outer code decoding will be skipped and the serial concatenation decoding is done.

If the syndrome $S_{rm}$ of any of $\hat{w}_0, \hat{w}_1, \ldots, \hat{w}_{t-1}$ is not zero, the second decoding phase begins and the outer code decoder carries out on the decoded vectors $\hat{w}_i$ whose syndromes $S_{rm}$ are not zero; If $S_{FF} \neq 0$ but all $t$ syndromes of $\hat{w}_0, \hat{w}_1, \ldots, \hat{w}_{t-1}$ are zero, the outer code decoder runs over all $t$ vectors. The outer code decoder is chosen to be bitwise BCJR decoder, which takes the received vector $r$ as channel values to compute branch metrics and $LLR_{inn}$ from inner code decoder as priori values. To further reduce the computational complexity, the BCJR decoder performs in a log domain and over a sectionalized TOF trellis diagram [20]. When the outer BCJR decoder completes, it outputs the maximum posteriori values $LLR_{inn}$ and re-calculates $S_{FF}$ and $S_{rm}$ based on updated $\hat{x}$ and $\hat{w}$, respectively. Similarly, if $S_{FF}$ and all $S_{rm}$ are zero, the turbo decoding will be terminated, otherwise $LLR_{out}$ will be passed to inner code decoder as priori information and a new iteration of turbo decoding will

![Fig. 2. Serial Concatenation Turbo Decoder](image-url)
begin. This proposed concatenation decoding will continue until \( S_{FF} \) and all \( S_{RM} \)s are zero or the preset \( I_f \) is reached.

The turbo decoder outputs the \( \hat{w} \) when the decoding process is completed. We then apply the inverse interleaver \( P^{-1} \) to sectionwise rearrange \( \hat{w} \) into \( \hat{c} \). After removing all parity check symbols in \( \hat{c} \), we deliver \( \hat{u} \), the \( t \cdot 2^m \) decoded information symbols to user. Fig. 2 demonstrates this serial concatenation turbo decoding.

C. Trapping sets of serial turbo concatenation

Let \( d_0 \) be the minimum distance of outer RM code, then the minimum distance of serial turbo concatenation is \( d_0 + \delta \), where \( \delta \) is contributed by the parity symbols formed by \( H_{FF} \). With the turbo decoding, no trapping set of size \( d_0 - 1 \) or smaller will cause an outer decoding failure, as the BCJR decoder is not affected by the cycles in Tanner graph. Moreover, if a trapping set of size \( d_0 - 1 \) or smaller makes the inner decoder fail, it will be broken by the outer decoder. Therefore, the turbo decoder has no harmful trapping sets of size \( d_0 - 1 \) or less.

IV. LOWERING ERROR FLOOR BY CONCATENATION

In this section, we show the design of serial turbo concatenation of RM code and FF-LDPC codes of high rate and medium length.

A. Example

This serial concatenation has code length of 2400 and code rate of 0.525, where the outer RM code is \( RM(3, 6) \) and inner FF-LDPC code \( C_{FF} \) has code length of 2400 and code rate of 0.8.

An information sequence of 1260 bits \( u \) is first divided into 30 blocks \( u_0, u_1, ..., u_{29} \), each consisting of 42 bits. As described in Section III, we use a cyclic Reed Muller code encoder and an interleaver \( P \) to encode each block into a codeword of \( RM(3, 6) \). By theorem 1, the generator polynomial \( g(X) \) of \( G_0(r, m) \) has roots:

\[
\begin{align*}
\alpha, & \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^{10}, \alpha^{16}, \\
\alpha^{20}, & \alpha^{34}, \alpha^{36}, \alpha^{38}, \alpha^{40}.
\end{align*}
\]

forming \( g(X) \) by \( \alpha \).

Next, \( w \) is used as the information sequence and encoded into a codeword \( x \) in \( C_{FF} \). \( C_{FF} \) is constructed by the method introduced in Section II. Specifically, let \( \alpha \) be a primitive element of \( GF(61) \).

We take a \( 8 \times 40 \) submatrix \( B(8, 40) \) from \( 2 \) as the base matrix with column weight 8 and row weight 40. We design a \( 8 \times 40 \) masking matrix \( Z(8, 40) = [Z_0, Z_1, Z_2, Z_3, Z_4] \), where each \( Z_i \) is a \( 8 \times 8 \) circulant and has a top row of weight 3 or 4. Masking \( B(8, 40) \) with \( Z(8, 40) \) generates a masked base matrix \( B_{mask} \) of \( G(61) \) with column weight 3 or 4 and row weight 17. The CPM dispersion of \( B_{mask} \) gives a \( 480 \times 2400 \) matrix \( H(8, 40) \) over \( G(2) \) with column weight 3 or 4 and row weight 17 which satisfies the RC-constraint. The null space of \( H(8, 40) \) is \( C_{FF} \), an irregular \((2400, 1920)\) QC FF-LDPC code with rate 0.8 and girth at least 8. Using the systematic form of generator matrix of \( C_{FF} \) to encode \( w \) completes the encoding of serial concatenation of \( RM(3, 6) \) and \( C_{FF} \), and generates a codeword \( x \) in \( (2400, 1260) \) serial concatenated code with rate 0.525.

To decode the received vector \( r \), we first use MSA based on the parity check matrix \( H(8, 40) \) of \((2400, 1920)\) FF-LDPC inner code \( C_{FF} \). At the end of each iteration, we check the syndrome \( S_{FF} \) of decoded vector \( \hat{x} \). If \( S_{FF} = 0 \), \( \hat{x} \) is a codeword in \( C_{FF} \), otherwise the decoding is continued until either \( \hat{x} \) is a codeword in \( C_{FF} \) or \( I_{max} \) is reached. Then, \( \hat{x} \) and the extrinsic information \( LLR_{in} \), of MSA are passed to BCJR decoder. Before the second decoding phase begins, the decoded sequence of information symbols \( \hat{w} \) in \( \hat{x} \) is partitioned into 30 vectors: \( \hat{w}_0, \hat{w}_1, ..., \hat{w}_{29} \). If \( S_{FF} \) and all 30 syndromes of \( \hat{w}_i \) are zero, the serial concatenation decoding is finished, otherwise BCJR decoder is activated to decode over either all \( \hat{w}_i \) (under the condition that \( S_{FF} \neq 0 \) and all 30 syndromes of \( \hat{w}_i \) are zero), or the \( \hat{w}_i \) whose syndromes are not zero. Once the BCJR decoder completes, it re-calculates \( S_{FF} \) and all \( s_{RM} \) based on updated \( \hat{x} \) and \( \hat{w} \). The serial turbo decoding will start a new iteration with extrinsic information \( LLR_{out} \) of BCJR decoder fed into MSA as priori values until \( S_{FF} \) and all 30 syndromes of \( \hat{w}_i \) are zero or the preset \( I_f \) is reached.

Once the serial turbo decoding is completed, we apply the inverse interleaver \( P^{-1} \) to rearrange each \( \hat{w}_i \) into \( \hat{c}_i, 0 \leq i \leq 29 \). After removing all parity bits, we deliver the 1260 decoded bits to user. Since the outer RM code has minimum distance 8, no trapping set of size 7 or smaller will cause a decoding failure.

B. Numerical Results

The FER performance of this \((2400, 1260)\) serial concatenated code over AWGN channel is shown in Fig. 3, together with the FER performance of outer code \( RM(3, 6) \) under BCJR decoding algorithm and FER performance of inner code \( C_{FF} \) under MSA. BPSK Modulation is considered in all three cases. The BCJR decoding algorithm runs over a 8-sectionized trellis diagram in log domain, and the MSA runs with a scaling factor 0.5 in concatenated decoding and 0.7 in individual \( C_{FF} \) decoding. The FER performance of \( C_{FF} \) under MSA has \( I_{max} \) equal to 25 and 50, respectively, both of which suggest that \( C_{FF} \) suffers from error floor at FER of \( 10^{-7} \) under MSA. The comparison of serial concatenated code with different \( I_f \) indicates that the turbo decoding converges vary fast. The simulation results show that no trapping sets of size 7 or smaller have been found in turbo decoding. The turbo decoding of serial concatenated code with both \( I_{max} \) and \( I_f \)
equal to 5 outperforms the MSA of $C_{FF}$ with $I_{max}$ equal to 25 around 0.8 dB, with no error floor observed at FER of $10^{-7}$. Similarly, with $I_{max}$ equal to 10, it has coding gain around 1 dB over the MSA of $C_{FF}$ with $I_{max}$ equal to 50.

![Fig. 3. FER performance of (2400,1260) serial concatenated code over AWGN channel](image)

**V. CONCLUSIONS**

In this paper, we proposed a serial concatenation of short RM codes and FF-LDPC codes of medium length and high rate. We introduced the cyclic RM encoding for outer RM code and the Latin Square based construction of inner FF-LDPC code, which are easy to be implemented. We also presented the design and decoding for proposed serial concatenated code, with the decoding working in a serial-turbo fashion. We proposed a recursive algorithm to generate the TOF of generator matrix of outer code to reduce complexity. We showed that the turbo decoding scheme has no harmful trapping sets of size smaller than the minimum distance of outer code. The simulation results indicate that the proposed serial concatenation under turbo decoding can lower the error floor and achieve a large coding gain over inner FF-LDPC code.

**ACKNOWLEDGMENT**

This work is funded by the NSF under grant ECCS-1500170, and is supported by Indo-US Science and Technology Forum (IUSSTF) through the Joint Networked Center for Data Storage Research (JC-16-2014-US).

**REFERENCES**


[11] M. Nasseri, T. Wang, and S. Lin, “Concatenated finite geometry and high rate. We introduced the cyclic RM encoding for outer RM code and the Latin Square based construction of inner FF-LDPC code, which are easy to be implemented. We also presented the design and decoding for proposed serial concatenated code, with the decoding working in a serial-turbo fashion. We proposed a recursive algorithm to generate the TOF of generator matrix of outer code to reduce complexity. We showed that the turbo decoding scheme has no harmful trapping sets of size smaller than the minimum distance of outer code. The simulation results indicate that the proposed serial concatenation under turbo decoding can lower the error floor and achieve a large coding gain over inner FF-LDPC code.

**ACKNOWLEDGMENT**

This work is funded by the NSF under grant ECCS-1500170, and is supported by Indo-US Science and Technology Forum (IUSSTF) through the Joint Networked Center for Data Storage Research (JC-16-2014-US).

**REFERENCES**