Conditionally Cycle-Free Graphical Models for Coset Codes

Thomas R. Halford, Keith M. Chugg, *Fellow, IEEE*, and Marcus T. Urie, *Student Member, IEEE*

***Abstract*—Conditionally cycle-free graphical models (i.e., cyclic graphical models which become cycle-free after conditioning on a subset of the hidden variables) are constructed for coset codes. Following the description of a general construction procedure, examples of a number of families of codes—including first-order Reed-Muller (RM) and the Delsarte-Goethals codes – are provided for which the proposed procedure yields optimal soft-in soft-out (SISO) decoding algorithms that are less complex than the best known trellis-based algorithms. In the case of the first-order RM codes, which have a recursive coset construction, the optimal SISO decoding algorithm that results when the proposed construction is applied repeatedly is denoted recursive coset representation (RCR) decoding. Connections are made between RCR decoding and existing algorithms that exploit fast Hadamard transforms. Finally, the utility of the proposed decoding algorithms are supported by a practically motivated application: the construction of serially concatenated codes that have high rates and low error floors. Extended Hamming codes are proposed as outer codes in such constructions with efficient decoding employing the SISO algorithms developed herein.**

***Index Terms*—Codes on graphs, coset codes, graphical model extraction, soft-in soft-out (SISO) decoding, trellis decoding.**

I. Introduction

TRELLIS representations of block codes were first studied by Bahl, *et al.* in [1], where it was shown that trellises imply optimal soft-in soft-out (SISO) decoding algorithms that can be much more efficient than exhaustive[1] SISO decoding. Wolf described the first systematic trellis construction method and showed that any q-ary length n, dimension k linear block code can be represented by a trellis with at most

$$q^{\min(k,n-k)} \tag{1}$$

states [2]. Ten years later, Forney's *Coset Codes* papers [3], [4] sparked renewed interest in the study of trellis representations. In particular, trellis constructions with fewer states than the bound of (1) were sought (cf., [5]) since the number of states in a trellis is a reasonable indicator for the number of arithmetic operations required by the decoding algorithm it implies.

From the viewpoint of graphical models for codes (cf., [6] and [7]), trellis representations of block codes constitute simple chain models. It is now well-known that *any* cycle-free graphical model for a code implies an optimal SISO decoding algorithm (cf., [6]–[9]). It is thus natural to ask whether there exists a more general cycle-free graphical model for a *given* block code that yields a less complex decoding algorithm than an optimal trellis. This is an example of an extractive graphical modeling problem [10] – i.e., one which is concerned with the obtention, or *extraction*, of a model for a specific code that implies a decoding algorithm with desired complexity and performance characteristics. Kashyap recently established bounds on the decrease in decoding complexity that may be afforded by more general cycle-free models [11], [12].

The present work considers an extractive problem that is closely related to the aforementioned cycle-free model search: graphical models are sought that can be made cycle-free by conditioning on a small number of hidden variables. An optimal SISO decoding algorithm can be obtained from such a *conditionally cycle-free* graphical model by applying the standard decoding algorithm implied by *each* conditional (cycle-free) graph configuration and then marginalizing over the results (cf., [13]). While such variable conditioning has been studied previously in the context of tail-biting codes [14], the focus herein is on a more general class of conditionally cycle-free models. Motivated by Forney's observation that the decomposition of certain codes into the union of cosets of some subcode can lead to efficient trellis realizations [4], the present work extracts conditionally cycle-free graphical models for codes based on such coset decompositions. The term coset SISO decoding is introduced to denote the algorithms implied by the extracted models.

Following a review of the necessary background material in Section II, coset SISO decoding of arbitrary codes is formulated in Section III. In Section IV, this general formulation is applied to first-order Reed-Muller codes in order to derive efficient optimal SISO decoding algorithms, referred to as recursive coset representation (RCR) decoding. Section IV includes a comparison of the RCR decoding algorithm to trellis decoding, Forney's "divide-by-2" algorithms [15] – which are themselves derived from cycle-free graphical models – and an approach based on fast Hadamard transforms similar to that proposed by Ashikhmin and Litsyn [16]. Section V describes coset SISO

Manuscript received October 07, 2008; revised February 19, 2011; accepted August 09, 2011. Date of current version January 06, 2012. The material in this paper was presented in part at the 2007 IEEE International Symposium on Information Theory and at the Hawaii and SITA Joint Conference on Information Theory, Honolulu, HI, May 2007.

T. R. Halford is with TrellisWare Technologies, Inc., San Diego, CA 92127 USA (e-mail: thalford@trellisware.com).

K. M. Chugg and M. T. Urie are with the Communication Sciences Institute, University of Southern California, Los Angeles, CA 90089 USA (e-mail: chugg@usc.edu; urie@usc.edu).

Communicated by H.-A. Loeliger, Associate Editor for Coding Techniques.

Digital Object Identifier 10.1109/TIT.2011.2169540

[1]This denotes a generalization of an exhaustive search to SISO decoding. For example, exhaustive sum-product decoding of a binary length n, dimension k linear block code entails (i) computing the probability that each of 2^k codewords was transmitted (via multiplication) and then (ii) computing the probability that a 0 (respectively, 1) was transmitted at each bit index i by summing the probabilities of each codeword that contains a 0 (respectively, 1) at index i.

decoding algorithms for a number of code families related to first-order RM codes. In Section VI, the utility of coset SISO decoding is supported by a practically motivated application: the construction of serially concatenated codes (SCCs) that have high rates and low error floors (e.g., $< 10^{-6}$ bit error rate). Extended Hamming codes are proposed as outer codes in SCC designs with efficient decoding employing the SISO algorithms described in Section V. Concluding remarks and directions for future work are given in Section VII. Finally, a number of appendices are provided that detail the algorithms against which RCR decoding is compared in Section IV.

II. Background

A. Notation

The field of reals is denoted $\mathbb{R}$ and the nonnegative reals are denoted $\mathbb{R}^+$. The finite field with q elements is denoted $\mathbb{F}_q$ while $\mathbb{Z}_q$ denotes the integers modulo-q. The set of consecutive integers from a to $b > a$ (inclusive) is denoted $[a, b]$.

Bold script letters denote vectors, e.g., $\boldsymbol{v}$ and $\boldsymbol{V}$, and vector elements are indexed starting from 1. Codes and sets are denoted by upper-case script letters, e.g., $\mathcal{C}$ and $\mathcal{S}$. A length n code comprising M codewords and having minimum distance d is described by the triplet (n, M, d). If a code is linear with dimension k then it is also described by the triplet $[n, k, d]$.

B. Coset Codes

Let $\mathcal{C}$ be a length n code over some q-ary alphabet $\mathcal{Q}$ with a defined addition operation. A *coset* of $\mathcal{C}$ is defined as

$$\mathcal{C} + \boldsymbol{l} = \{\boldsymbol{c} + \boldsymbol{l}, \boldsymbol{c} \in \mathcal{C}\} \tag{2}$$

where $\boldsymbol{l} \in \mathcal{Q}^n$ is some length n vector. A *coset code* $\widetilde{\mathcal{C}}$ is simply the union of the cosets of some code $\mathcal{C}$ corresponding to some set of coset representatives $\mathcal{L}$

$$\widetilde{\mathcal{C}} = \bigcup_{\boldsymbol{l} \in \mathcal{L}} (\mathcal{C} + \boldsymbol{l}) \,. \tag{3}$$

If $\mathcal{L}$ contains the all zeros vector $\mathbf{0}$, then $\mathcal{C}$ is a *subcode* of the coset code $\widetilde{\mathcal{C}}$. We focus on codes for which any codeword $\widetilde{\boldsymbol{c}} \in \widetilde{\mathcal{C}}$ has a *unique* decomposition $\widetilde{\boldsymbol{c}} = \boldsymbol{c} + \boldsymbol{l}$ where $\boldsymbol{c} \in \mathcal{C}$ and $\boldsymbol{l} \in \mathcal{L}$, which is the case for coset codes, since $\mathcal{L}$ always contains exactly one representative from each coset it covers.

C. Graphical Models for Codes

In this paper, Forney's normal graph convention is adopted; the reader is refered to [7] and [15] for a full treatment. Briefly, visible variables are represented by half-edges, hidden (generalized state) variables by edges, and local constraints by vertices. Throughout this paper, repetition and single parity-check constraints are represented by vertices labeled with '=' and '+' symbols, respectively. Graphical models imply well-understood SISO decoding algorithms which are optimal for cycle-free models.

D. Semi-Ring Algorithms

In order to properly compare the proposed RCR decoding algorithm to that introduced by Ashikhmin and Litsyn in [16], it is necessary to review the semi-ring algorithm formalism. A *commutative semi-ring* [8] is a set $\mathcal{F}$ together with binary *combining* $\copyright$ and *marginalizing* $\textcircled{m}$ operations satisfying:

S1. $\copyright$ and $\textcircled{m}$ are both commutative and associative on $\mathcal{F}$.

S2. $\exists\, I_c, I_m \in \mathcal{F}$ such that $f \copyright I_c = f$ and $f \textcircled{m} I_m = f$ for all $f \in \mathcal{F}$.

S3. The distributive law holds so that for all $f, g, h \in \mathcal{F}$, $f \copyright (g \textcircled{m} h) = (f \copyright g) \textcircled{m} (f \copyright h)$.

While not a strict requirement of semi-rings, the semi-rings considered in this paper all satisfy the additional property that:

S4. $\copyright$ is a group operation so that there exists a unique inverse f^{-1} for each $f \in \mathcal{F}$ with respect to $\copyright$.

Note that this property does not hold in general for marginalizing operators. For example, neither the $\min$ nor $\max$ operators have well-defined inverses—i.e., if either $\min(x, y) = x$ or $\max(x, y) = x$ is known, y remains undefined.

Semi-rings were first described in the context of SISO decoding by Wiberg [6], and later formalized by Aji and McEliece [8] in order to unify a class of problems that marginalize a product function (MPF). The exhaustive SISO decoding procedure described above is one such MPF problem. It is now well-understood that optimal symbol-wise decoding in the probability domain utilizes the sum-product semi-ring over $\mathbb{R}^+$, while the use of the max-product semi-ring over $\mathbb{R}^+$ yields optimality with respect to codeword error probability. Similarly, in the *metric* domain – where probabilities are replaced by their negative logarithms – $\min^\star$-sum and min-sum processing replace sum-product and max-product processing, respectively.[2]

A *semi-ring algorithm* is any algorithm that solves an MPF problem using only the semi-ring properties of the marginalizing and combining operations [19]. Since it uses only the semi-ring properties, a semi-ring algorithm developed for one particular semi-ring can be immediately converted to an equivalent semi-ring algorithm for any other semi-ring. The forward-backward algorithm (FBA) [1] is an example of a semi-ring algorithm. The Viterbi algorithm [20], however, is not since survivor path selection depends on the property of the min operator ($\textcircled{m}$ in the min-sum semi-ring) that $\min(a, b) \in \{a, b\}$, which is not a semi-ring property.

III. Graphical Models for Coset Codes

A. Optimal SISO Decoding of Coset Codes

Let the soft inputs to a decoding algorithm for a length n binary coset code $\widetilde{\mathcal{C}}$ be $\{\boldsymbol{SI}[x_i]\}_{i=1}^n$, where

$$\boldsymbol{SI}[x_i] = \{\mathrm{SI}[x_i = 0], \mathrm{SI}[x_i = 1]\} \tag{4}$$

[2]Decoding with the $\min^\star$-sum semi-ring, which was introduced in [17], is equivalent to "log-MAP" processing. Formally, $\min^\star(x, y) = -\log(e^{-x} + e^{-y}) = \min(x, y) - \log(1 + e^{-|x-y|})$. A similarly defined $\max^\star$ operator has been described by a number of authors (cf., [18]). Note that $\max^\star$-sum processing operates on logarithms of probabilities ("log-likelihoods") rather than negative logarithms ("metrics").

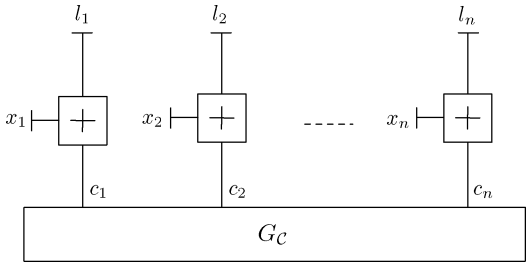

Fig. 1. Generic graphical model for the coset $\mathcal{C} + \boldsymbol{l}$.

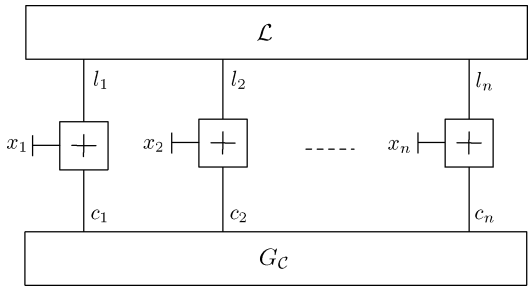

Fig. 2. Generic graphical model for the coset code $\widetilde{\mathcal{C}} = \bigcup_{\boldsymbol{l} \in \mathcal{L}} (\mathcal{C} + \boldsymbol{l})$.

is a vector that provides the metrics (or probabilities) that a 0 or 1 was received at the i^{th} coordinate. An optimal SISO decoding algorithm for $\widetilde{\mathcal{C}}$ computes the soft outputs [19]

$$\mathrm{SO}[x_i = b] = \mathop{\textcircled{m}}_{\boldsymbol{x} \in \widetilde{\mathcal{C}} | x_i = b} \left\{ \mathrm{S}\left[\boldsymbol{x}\right] \right\} \textcircled{c}\, \mathrm{SI}[x_i = b]^{-1} \tag{5}$$

for $i \in [1, n]$ and $b \in \mathbb{F}_2$, and where

$$\mathrm{S}\left[\boldsymbol{x}\right] = \left(\mathop{\textcircled{c}}_{j | x_j = 0} \mathrm{SI}[x_j = 0] \right) \textcircled{c} \left(\mathop{\textcircled{c}}_{j | x_j = 1} \mathrm{SI}[x_j = 1] \right) \tag{6}$$

is the combined input soft information for the codeword $\boldsymbol{x} \in \widetilde{\mathcal{C}}$. Applying a coset decomposition of $\widetilde{\mathcal{C}}$ to (5) yields a second formulation of optimal SISO decoding

$$\mathrm{SO}[x_i = b] = \mathop{\textcircled{m}}_{\boldsymbol{l} \in \mathcal{L}} \left\{ \mathop{\textcircled{m}}_{\substack{\boldsymbol{c} \in \mathcal{C} | \\ c_i + l_i = b}} \left\{ \mathrm{S}\left[\boldsymbol{c} + \boldsymbol{l}\right] \right\} \textcircled{c}\, \mathrm{SI}[x_i = b]^{-1} \right\}. \tag{7}$$

This formulation is readily extended to nonbinary codes. The factorization in (7) implies a class of SISO algorithms for coset codes that marginalize over the results of SISO decoding of each coset; such algorithms are discussed in detail later.

B. A Generic Model

Let $\widetilde{\mathcal{C}}$ be a length n coset code over a q-ary alphabet defined as per (3) and let $G_{\mathcal{C}}$ be a graphical model for the subcode $\mathcal{C}$. Fig. 1 illustrates a generic graphical model for the coset $\mathcal{C} + \boldsymbol{l}$ where $\boldsymbol{l} = (l_1, \ldots, l_n)$. Note that $l_1, \ldots, l_n$ are treated as deterministic visible variables in Fig. 1. If $G_{\mathcal{C}}$ is cycle-free, then the graphical model illustrated in Fig. 1 implies an optimal SISO decoding algorithm for $\mathcal{C} + \boldsymbol{l}$—i.e., the decoding model implied by Fig. 1 then computes

$$\mathrm{SO}[x_i = b | \boldsymbol{l}] = \mathop{\textcircled{m}}_{\substack{\boldsymbol{c} \in \mathcal{C} | \\ c_i + l_i = b}} \left\{ \mathrm{S}\left[\boldsymbol{c} + \boldsymbol{l}\right] \right\} \textcircled{c}\, \mathrm{SI}[x_i = b]^{-1} \tag{8}$$

for each codeword coordinate i and value $b \in \mathbb{F}_q$.

Fig. 2 extends the model of Fig. 1 so that $\boldsymbol{l}$ is constrained not to be a specific element of $\mathcal{L}$, but any element of $\mathcal{L}$. The graphical model illustrated in Fig. 2 constitutes a model for the coset code $\widetilde{\mathcal{C}}$. Irrespective of the cyclic topology of $G_{\mathcal{C}}$, the model illustrated in Fig. 2 contains cycles and thus implies a suboptimal *standard* decoding algorithm.

An optimal SISO decoding algorithm for $\widetilde{\mathcal{C}}$ can be obtained from the model illustrated in Fig. 2 via hidden variable conditioning. Hidden variable conditioning in graphical models is well-understood in the context of tail-biting codes (cf., [14]) and has also been used to slightly improve the performance of decoding algorithms for low-density parity-check (LDPC) and serially concatenated convolutional codes [13]. For example, if a tail-biting trellis contains a hidden (state) variable V with alphabet size q, then optimal SISO decoding can be performed on this trellis by decoding on q conditional trellises (one per possible value of V) and then appropriately marginalizing over the decoding rounds. This approach yields an optimal decoding algorithm because the graphical model that results when V is constrained to a specific value is *conditionally cycle-free*. Returning to Fig. 2, if $G_{\mathcal{C}}$ is cycle-free then an algorithm which computes (7) is obtained be obtained by conditioning on the $|\mathcal{L}|$ valid configurations of $l_1, \ldots, l_n$ (to obtain (8) for each $\boldsymbol{l} \in \mathcal{L}$) and then marginalizing over the results of the decoding rounds.

C. Decoding Details

There are two details of the decoding algorithm for $\widetilde{\mathcal{C}}$ described above that must be addressed before examining complexity. It is first shown that the complexity of optimal SISO decoding of $\mathcal{C} + \boldsymbol{l}$ is identical to that of $\mathcal{C}$. The care that must be exercised when applying metric normalization techniques to coset SISO decoding is then discussed.

Fig. 3 illustrates the messages passed into ($\boldsymbol{SI}[\cdot]$) and out of ($\boldsymbol{SO}[\cdot]$) one of the single parity-check (SPC) constraints of Fig. 1. Suppose first that $\widetilde{\mathcal{C}}$ is a binary code and that decoding is done in the metric domain (i.e., negative logarithms of probabilities). If $l_j = 0$ then $\mathrm{SI}[l_j = 0] = 0$, $\mathrm{SI}[l_j = 1] = \infty$, and the message updates that occur for c_j and x_j at the SPC illustrated in Fig. 3 are

$$\begin{aligned} &\mathrm{SO}[c_j = 0] = \mathrm{SI}[x_j = 0], \quad \mathrm{SO}[c_j = 1] = \mathrm{SI}[x_j = 1], \\ &\mathrm{SO}[x_j = 0] = \mathrm{SI}[c_j = 0], \quad \mathrm{SO}[x_j = 1] = \mathrm{SI}[c_j = 1]. \end{aligned} \tag{9}$$

If, however, $l_j = 1$ then $\mathrm{SI}[l_j = 0] = \infty$, $\mathrm{SI}[l_j = 1] = 0$, and the message updates that occur for c_j and x_j at the SPC illustrated in Fig. 3 are

$$\begin{aligned} &\mathrm{SO}[c_j = 0] = \mathrm{SI}[x_j = 1], \quad \mathrm{SO}[c_j = 1] = \mathrm{SI}[x_j = 0], \\ &\mathrm{SO}[x_j = 0] = \mathrm{SI}[c_j = 1], \quad \mathrm{SO}[x_j = 1] = \mathrm{SI}[c_j = 0]. \end{aligned} \tag{10}$$

More generally, for arbitrary q, the message updates at the SPCs in Fig. 3 entail simply permuting the values of the incoming messages. Thus, the SPCs of Fig. 1 do not impact decoding complexity; optimal SISO decoding of $\mathcal{C} + \boldsymbol{l}$ is therefore identical in complexity to that of $\mathcal{C}$.

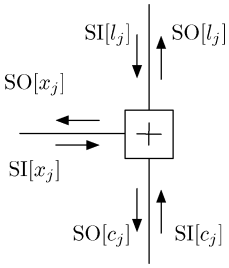

Fig. 3. Coset code single parity-check constraint update.

Metric normalization techniques are widely employed in practical implementations of the decoding algorithms implied by graphical models. For example, it is common practice in LDPC decoding algorithms to replace the *vector* message $\boldsymbol{S}[x] = \{\mathrm{S}[x=0], \mathrm{S}[x=1]\}$ corresponding to a binary variable x by the *scalar* log-likelihood ratio (LLR)

$$\mathrm{S}[x=1] \mathbin{\copyright} \mathrm{S}[x=0]^{-1}. \tag{11}$$

Decoding with LLRs is equivalent to normalizing all metrics so that the all-zero codeword has zero metric. Metric normalization is also used to maintain numerical stability in the Viterbi and forward-backward algorithms (cf., [21]). For example, the forward and backward metrics of a q-ary trellis state variable y may be *rescaled* such that $\min_{b \in \mathbb{F}_q} \mathrm{S}[y=b] = 0$ in the FBA, resulting in soft outputs of the form

$$\widehat{\mathrm{SO}}[x_i = b] = \mathrm{SO}[x_i = b] \mathbin{\copyright} r_i \tag{12}$$

where $\boldsymbol{r}$ is a rescaling vector. Property S1 of semi-rings (see Section II-D) implies that optimal decoding decisions can be obtained from such rescaled soft output metrics.

Care must be exercised in applying metric normalization techniques to coset SISO decoding. For example, suppose that the graphical model $G_{\mathcal{C}}$ for $\mathcal{C}$ in Fig. 2 corresponds to a trellis and that metric rescaling is applied on a per coset basis. The soft outputs for the coset $\mathcal{C} + \boldsymbol{l}$ are then of the form

$$\widehat{\mathrm{SO}}[x_i = b|\boldsymbol{l}] = \mathrm{SO}[x_i = b|\boldsymbol{l}] \mathbin{\copyright} r_i(\boldsymbol{l}) \tag{13}$$

where $\boldsymbol{r}(\boldsymbol{l})$ is a coset-specific rescaling vector. It can be shown that $\mathrm{SO}[x_i = b]$ cannot be *directly* obtained from such information – i.e., unless $\boldsymbol{r}(\boldsymbol{l})$ for all $\boldsymbol{l} \in \mathcal{L}$ are also known.

More generally, any hidden variable metrics that are formed in the optimal SISO decoder for the coset $\mathcal{C} + \boldsymbol{l}$ cannot be normalized independently across cosets. In this paper, the complexity of coset SISO decoding, as well as the methods to which it is compared, is therefore formulated without regard to possible savings resulting from metric normalization. In particular, vector messages for binary variables – rather than LLRs – are assumed throughout.

D. Decoding Complexity

In light of the discussion of Section III-C, the complexity of the optimal SISO decoding algorithm implied by hidden variable conditioning in the graphical model of Fig. 2 is readily determined as follows. Let the number of combining and marginalizing operations required for optimal SISO decoding of $\mathcal{C}$ be $C_{\mathcal{C}}$ and $M_{\mathcal{C}}$, respectively. The number of combining and marginalizing operations required for optimal SISO decoding of the length n coset code $\widetilde{\mathcal{C}}$ over a q-ary alphabet is then

$$C_{\widetilde{\mathcal{C}}} = LC_{\mathcal{C}} \tag{14}$$

$$M_{\widetilde{\mathcal{C}}} = LM_{\mathcal{C}} + (L-1)qn \tag{15}$$

where L is the number of cosets of $\mathcal{C}$ contained in $\widetilde{\mathcal{C}}$. Note that the second term of (15) captures the complexity of marginalizing over the results of each decoding round: for each of the n codeword indices i, and for each of the q possible values for x_i, the outputs of L decoding rounds must be marginalized (at a cost of $L-1$ marginalization operations).

IV. Graphical Models for First-Order RM Codes

A. Model Derivation

Let $\mathcal{RM}(r,m)$ denote the r^{th}-order binary Reed-Muller (RM) code with parameters:

$$n = 2^m, \quad k = \sum_{j=0}^{r} \binom{m}{j}, \quad d = 2^{m-r}. \tag{16}$$

Reed-Muller codes can be defined recursively as [22]

$$\mathcal{RM}(r+1, m+1) = \big\{(\boldsymbol{u}, \boldsymbol{u}+\boldsymbol{v}) \big| \boldsymbol{u} \in \mathcal{RM}(r+1, m), \boldsymbol{v} \in \mathcal{RM}(r, m)\big\}, \tag{17}$$

where $\mathcal{RM}(0,m)$ is a length 2^m repetition code and $\mathcal{RM}(1,1)$ is a length 2 SPC. As noted by Forney, this recursive construction can be reformulated under the coset code rubric [4]. Specifically, define the code

$$\mathcal{RM}(r+1,m)^2 = \big\{(\boldsymbol{u}, \boldsymbol{u}) \big| \boldsymbol{u} \in \mathcal{RM}(r+1, m)\big\} \tag{18}$$

which comprises the concatenation of identical codewords of $\mathcal{RM}(r+1,m)$. The recursive construction of (17) then becomes

$$\mathcal{RM}(r+1, m+1) = \bigcup_{\boldsymbol{v} \in \mathcal{RM}(r,m)} \mathcal{RM}(r+1,m)^2 + (\boldsymbol{0}|\boldsymbol{v}). \tag{19}$$

First-order RM codes have a particularly simple representation as coset codes. Since $\mathcal{RM}(0,m)$ is a length 2^m binary repetition code, (19) becomes

$$\mathcal{RM}(1, m+1) = \bigcup_{\boldsymbol{v} \in \{\boldsymbol{0}, \boldsymbol{1}\}} \mathcal{RM}(1,m)^2 + (\boldsymbol{0}|\boldsymbol{v}). \tag{20}$$

A graphical model for $\mathcal{RM}(1, m+1)$ can thus be constructed as per Section III, provided a graphical model for $\mathcal{RM}(1,m)^2$ is known. Since codewords in $\mathcal{RM}(1,m)^2$ are formed by concatenating identical codewords in $\mathcal{RM}(1,m)$, a model for $\mathcal{RM}(1,m)$ can be used to construct one for $\mathcal{RM}(1,m)^2$. Fig. 4 illustrates the resulting graphical model for $\mathcal{RM}(1, m+1)$. Note that the $\mathcal{L}$ constraint of Fig. 2 has been replaced by a simple binary equality constraint.

As described in Section III, the graphical model illustrated in Fig. 4 implies an optimal SISO decoding algorithm for $\mathcal{RM}(1, m+1)$ if the model for $\mathcal{RM}(1,m)$ is cycle-free (or, more generally, implies an optimal decoding algorithm). Since first-order RM codes are defined recursively, the construction of

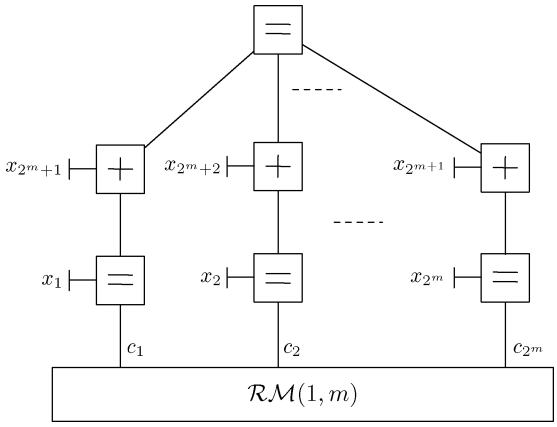

Fig. 4. Graphical model for $\mathcal{RM}(1, m+1)$ based on a coset code representation.

Fig. 4 can be repeated recursively. Provided that optimal SISO decoding is performed at the base of this recursion, an optimal SISO decoding algorithm for $\mathcal{RM}(1, m+1)$ results. As a base for this recursion, consider applying the decomposition of (20) to $\mathcal{RM}(1,3)$. Since $\mathcal{RM}(1,2)$ is simply a length 4 SPC, trellis decoding of this base code is sufficient.

B. Complexity

Let C_{m+1} and M_{m+1}, respectively, denote the number of combining and marginalizing operations required by the optimal SISO decoding algorithm for $\mathcal{RM}(1, m+1)$ presented above. Applying the ideas of Section III to the model illustrated in Fig. 4 yields

$$C_{m+1} = 2C_m + 3 \cdot 2^{m+2} \tag{21}$$
$$M_{m+1} = 2M_m + 2^{m+2}. \tag{22}$$

Note the discrepancy between the form of (14) and (21)—i.e., the $3 \cdot 2^{m+2}$ term. The soft input values on the i^{th} coordinate of $\mathcal{RM}(1,m)$ are formed as

$$\begin{aligned} \mathrm{SI}[c_i = b|\mathbf{0}] &= \mathrm{SI}[x_i = b] \copyright \mathrm{SI}[x_{2^m+i} = b] \\ \mathrm{SI}[c_i = b|\mathbf{1}] &= \mathrm{SI}[x_i = b] \copyright \mathrm{SI}[x_{2^m+i} = b+1] \end{aligned} \tag{23}$$

while the soft output values on x_i and x_{2^m+i} are

$$\begin{aligned} \mathrm{SO}[x_i = b|\mathbf{0}] &= \mathrm{SO}[c_i = b|\mathbf{0}] \copyright \mathrm{SI}[x_{2^m+i} = b] \\ \mathrm{SO}[x_i = b|\mathbf{1}] &= \mathrm{SO}[c_i = b|\mathbf{1}] \copyright \mathrm{SI}[x_{2^m+i} = b+1] \\ \mathrm{SO}[x_{2^m+i} = b|\mathbf{0}] &= \mathrm{SO}[c_i = b|\mathbf{0}] \copyright \mathrm{SI}[x_i = b] \\ \mathrm{SO}[x_{2^m+i} = b|\mathbf{1}] &= \mathrm{SO}[c_i = b+1|\mathbf{1}] \copyright \mathrm{SI}[x_i = b+1] \end{aligned} \tag{24}$$

where, for example, $\mathrm{SO}[c_i = b|\mathbf{0}]$ is the soft-output generated by the decoder for $\mathcal{RM}(1,m)$ for the $\boldsymbol{l} = \mathbf{0}$ coset. The 12 combination operations defined by (23) and (24) are repeated for each of the 2^m inputs to $\mathcal{RM}(1,m)$. Finally, as a base for this recursion, the operations required for SISO trellis decoding of $\mathcal{RM}(1,2)$ are (see Appendix):

$$C_2 = 24, \quad M_2 = 12. \tag{25}$$

TABLE I
COMPLEXITY OF RECURSIVE COSET SISO DECODING VS. TRELLIS AND TREE-BASED METHODS

	Trellis		DB2		RCR	
	©	ⓜ	©	ⓜ	©	ⓜ
$\mathcal{RM}(1,3)$	120	48	96	40	96	40
$\mathcal{RM}(1,4)$	528	192	320	144	288	112
$\mathcal{RM}(1,5)$	2208	768	1152	544	768	288
$\mathcal{RM}(1,6)$	9024	3072	4532	2112	1920	704
$\mathcal{RM}(1,7)$	36480	12288	16896	8320	4608	1664
$\mathcal{RM}(1,8)$	146688	49152	66560	33024	10752	3840

It is apparent from (21) and (22) that the complexity of the optimal SISO decoding algorithms for first-order RM codes implied by this RCR grows as $O(n \log n)$. Table I compares the complexity of RCR decoding to two other optimal SISO techniques: trellis decoding and the tree-based "divide-by-2" (DB2) algorithms described by Forney in [4], [7]. While all three algorithms exploit the recursive structure of first-order RM codes, and therefore exhibit $O(n \log n)$ growth in decoding complexity, the constant coefficients in the rate of growth are smallest for RCR decoding. As detailed in the Appendix, operations are counted in the trellis-based and DB2 decoders assuming the application of *generic* algorithms. For a specific code, complexity savings could likely be obtained for trellis and DB2 decoding by careful application of, for example, the tricks described in [23]. However, as m increases, it becomes increasingly difficult to apply such techniques to decoding algorithms for $\mathcal{RM}(1,m)$ by hand. In this light, RCR decoding can be interpreted as an efficient means of systematically obtaining the complexity savings that can be had via careful optimization of existing trellis- and tree-based decoding algorithms.

C. Comparison to Ashikhmin and Litsyn's SISO

An optimal decoding algorithm for first-order RM codes using fast Hadamard transforms (FHT) was developed in [16]. As noted by a number of authors (cf., [24]), the combination step of exhaustive SISO decoding of first-order RM codes (in the metric or log-likelihood domain) can be performed efficiently via an FHT. Ashikhmin and Litsyn further noted that by converting the resulting codeword metrics to the probability domain, an FHT can also be used for marginalizing in the case of symbol-wise decoding—i.e., combination is performed via the sum operation in the metric or log-likelihood domain ($\min^\star$-sum or max-$\star$-sum) while marginalization is performed via addition in the probability domain (sum-product).

A standard FHT, however, cannot be used for marginalizing in arbitrary semi-rings. The Appendix defines a Generalized FHT (GFHT) that can be used in place of the FHT in Ashikhmin and Litsyn's algorithm as follows. Adopting the notation of the Appendix, the inputs to the combination step of the algorithm described in [16] are

$$\gamma_k = \alpha_k - \beta_k = \mathrm{SI}[x_k = 1] - \mathrm{SI}[x_k = 0] \tag{26}$$

and the outputs are

$$\Gamma_j = \mathrm{S}[\boldsymbol{x}_j] - \mathrm{S}[\boldsymbol{x}_j \oplus \mathbf{1}] \tag{27}$$

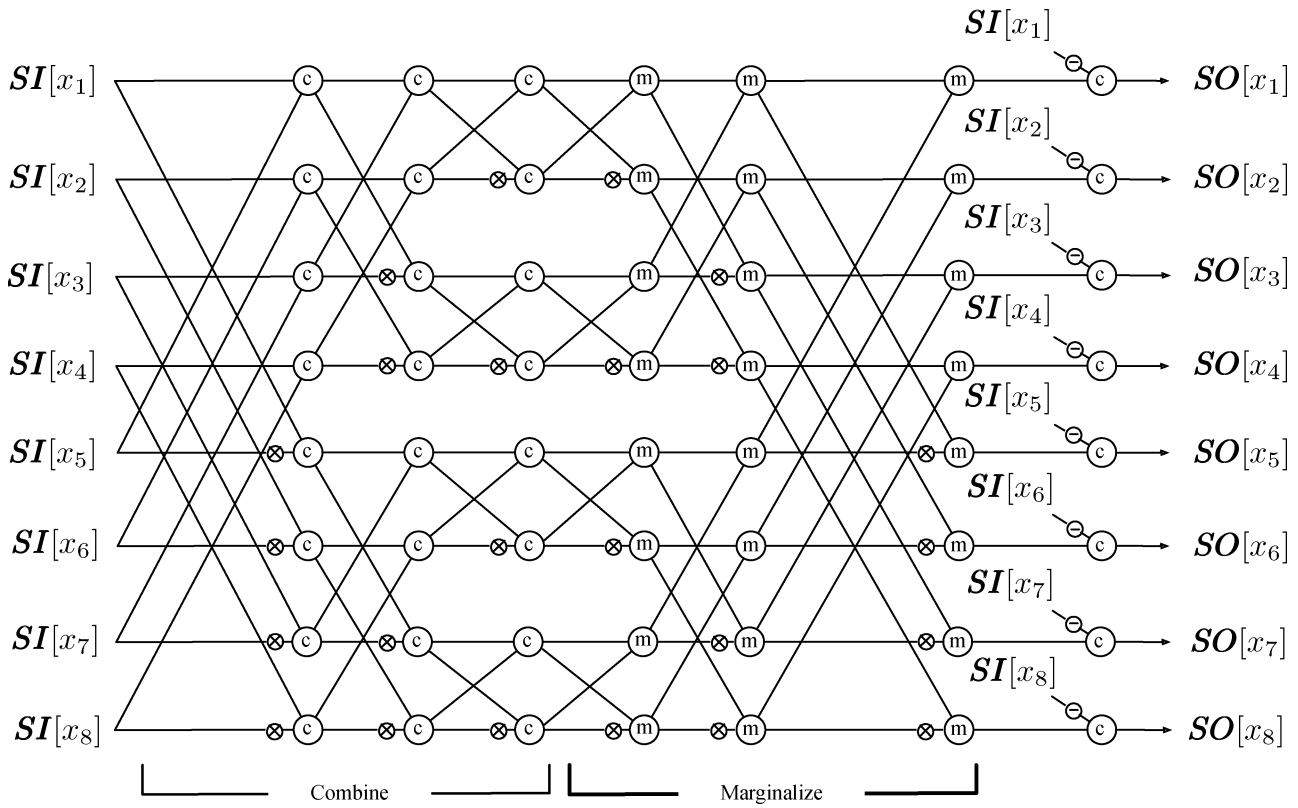

Fig. 5. Data flow of GFHT decoding algorithm for $\mathcal{RM}(1,3)$. Per the discussion in the Appendix, solid lines indicate vector-valued variables so that each illustrated combining (respectively, marginalizing) operation corresponds to 2 binary operations. There are thus a total of 64 binary combining and 48 binary marginalizing operations illustrated in this figure.

where $\mathrm{S}[\boldsymbol{x}_j]$ is metric for the j^{th} codeword. The combining FHT of [16] can therefore readily be replaced by a GFHT that takes vector inputs $\boldsymbol{\gamma}_k = (\alpha_k, \beta_k)$ and produces vector outputs

$$\boldsymbol{\Gamma}_k = (\Gamma_{k,1}, \Gamma_{k,2}) = (\mathrm{S}[\boldsymbol{x}_j], \mathrm{S}[\boldsymbol{x}_j \oplus \mathbf{1}]). \tag{28}$$

The marginalizing step of Ashikhmin and Litsyn's algorithm replaces differences of bit metrics with differences of codeword probabilities as inputs to the GFHT. Since the GFHT is a semi-ring algorithm, the conversion from metric to probability domain is not necessary; the combining step outputs (28) are direct inputs to the marginalizing GFHT.

The advantage of the FHT approach given in [16] is that half as much processing is required in the computation of the standard FHT as in the GFHT, at the expense of some pre- and post-processing (operation count $O(n)$) of the probabilities. Since the operation counts of both the FHT and the GFHT grow as $O(n \log n)$, the use of the FHT provides a slight reduction in the total operation count. The advantage of the GFHT approach given here is that it extends the FHT algorithm to all semi-rings and removes the need to switch domains in the middle of the processing. Additionally, since the GFHT allows for all computation to be done in the metric domain, numerical instability issues inherent to probability domain computations can be avoided. Finally, since the GFHT is a semi-ring algorithm, the $\min$ operator can be used for marginalizing – i.e., optimal decoding with respect to codeword error probability is possible with the GFHT.

One final matter must be considered before the resulting GFHT method can be compared with the RCR model described in Section IV-A. In contrast to standard decoding algorithms implied by graphical models such as the RCR model, the GFHT (and FHT) algorithm inherently outputs intrinsic metrics, rather than extrinsic metrics. When used as a SISO decoding component of a larger modern code, extrinsic metrics are desired at the output of the GFHT SISO. Such metrics can be obtained using postprocessing as follows:

$$\begin{aligned} \mathrm{SO}[c_k = 0] &= \mathrm{S}[c_k = 0] \,\textcircled{c}\, \mathrm{SI}[c_k = 0]^{-1} \\ \mathrm{SO}[c_k = 1] &= \mathrm{S}[c_k = 1] \,\textcircled{c}\, \mathrm{SI}[c_k = 1]^{-1}. \end{aligned} \tag{29}$$

Fig. 5 illustrates the data flow of the resulting GFHT algorithm, including the postprocessing described earlier. The ⓒ and ⓜ symbols denote the combining and marginalizing operators, respectively. As in the Appendix, a $\ominus$ symbol on an input line denotes that the value on that line is replaced by its inverse and a $\otimes$ symbol indicates the swapping of values described in (52) (i.e., the vector $\{x, y\}$ becomes $\{y, x\}$).

From Fig. 5, the complexity of the GFHT algorithm can be readily determined. Let C_{m+1} and M_{m+1}, respectively, denote the number of combining and marginalizing operations required by the optimal GFHT SISO decoding algorithm for $\mathcal{RM}(1, m+1)$. Since the algorithm is based entirely on two consecutive generalized FHTs, the following recursion holds (as seen for $\mathcal{RM}(1,3)$ in Fig. 5):

$$C_{m+1} = 2C_m + 2^{m+2} \tag{30}$$

$$M_{m+1} = 2M_m + 2^{m+2}. \tag{31}$$

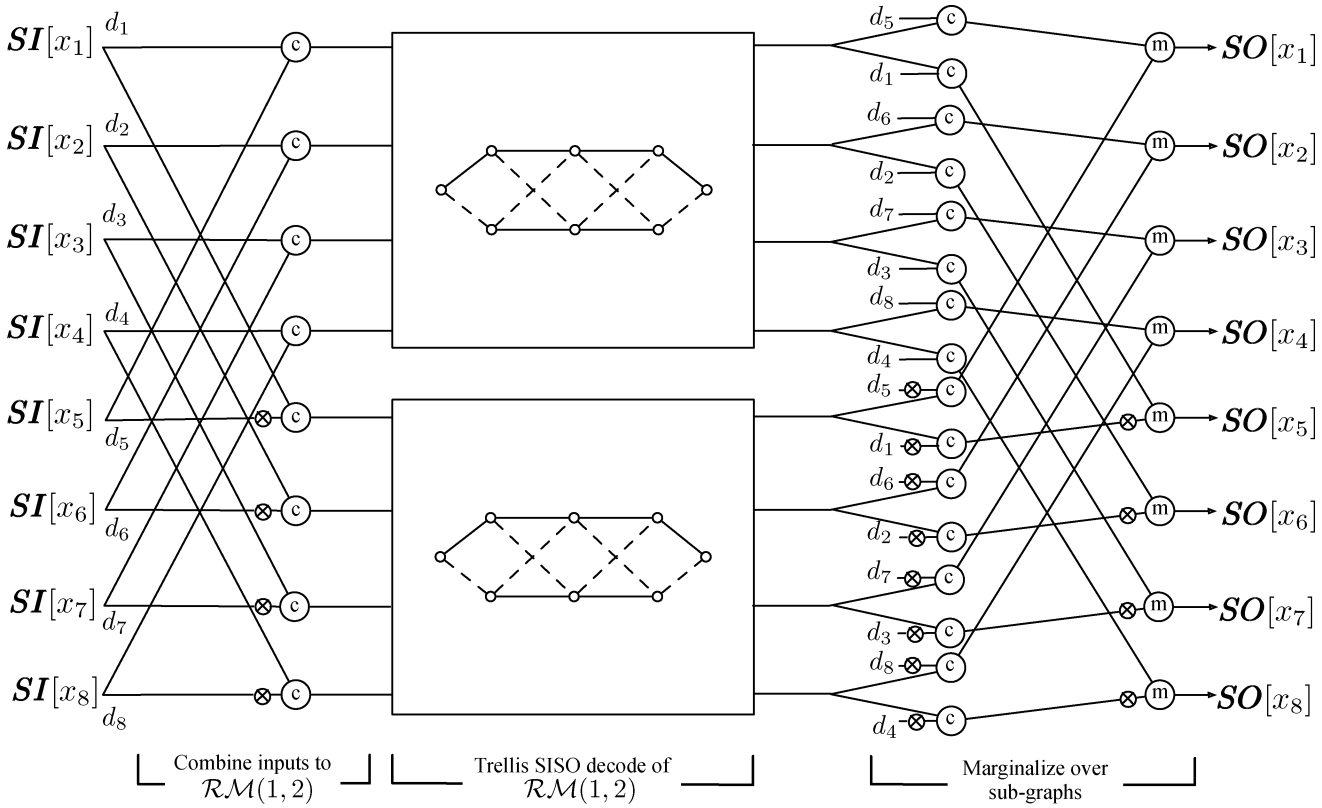

Fig. 6. Data flow of RCR decoding algorithm for $\mathcal{RM}(1,3)$. Each $\mathcal{RM}(1,2)$ trellis entails 24 binary combining and 12 binary marginalizing operations. A total of 96 binary combining and 40 binary marginalizing operations are thus illustrated in this figure.

TABLE II
COMPLEXITY OF RECURSIVE COSET VS. GFHT-BASED DECODING

	RCR		GFHT	
	ⓒ	ⓜ	ⓒ	ⓜ
$\mathcal{RM}(1,3)$	96	40	64	48
$\mathcal{RM}(1,4)$	288	112	160	128
$\mathcal{RM}(1,5)$	768	288	384	320
$\mathcal{RM}(1,6)$	1920	704	896	768
$\mathcal{RM}(1,7)$	4608	1664	2048	1792
$\mathcal{RM}(1,8)$	10752	3840	4608	4096

The operations required for the GFHT decoding algorithm for $\mathcal{RM}(1,2)$ provide a base for this recursion

$$C_2 = 24, \quad M_2 = 16. \tag{32}$$

Equations (30)–(32) are similar to the recursion (21), (22), and (25) for the RCR decoding algorithm. Table II gives specific operation counts for both the GFHT and RCR algorithms for various first-order RM codes. The operation count for the GFHT algorithm includes the postprocessing required to produce extrinsic metrics at the output. RCR decoding requires fewer marginalizing operations than the GFHT algorithm at the expense of more combining operations.

A data flow representation of the RCR algorithm described in Section IV-A similar to that given for the GFHT algorithm in Fig. 5 is illustrated in Fig. 6. A comparison of Figs. 5 and 6 illustrates the distinction between the GFHT and RCR algorithms and demonstrates how the RCR algorithm trades marginalizing operations for combining operations. Trellis decoding of the base $\mathcal{RM}(1,2)$ is more efficient than the GFHT approach (12 vs. 16 marginalizing operations). A less complex algorithm than either RCR or GFHT decoding could thus be constructed by replacing the base 4-input GHFT stages in Fig. 5 by $\mathcal{RM}(1,2)$ trellises.

V. CODES RELATED TO FIRST-ORDER RM CODES

A. $\mathrm{Min}^\star$-*Sum Decoding of Extended Hamming Codes*

Hartmann and Rudolph first described optimal symbol-wise SISO decoding algorithms for linear codes based on trellis representations of their duals in [25]. Forney later generalized this result to more general cycle-free realizations of group codes [7]. To review, in the context of sum-product decoding of the binary code $\mathcal{C}$ with dual $\mathcal{C}^\perp$, suppose that a vector $\boldsymbol{r}$ is received and that the likelihood ratios

$$\widehat{\mathrm{SI}}[c_j] = \frac{\Pr(r_j|c_j=1)}{\Pr(r_j|c_j=0)} \tag{33}$$

are computed for each $j \in [1,n]$. A standard optimal SISO decoding algorithm for $\mathcal{C}$ takes as input these likelihood ratios and returns as output

$$\widehat{\mathrm{SO}}[c_j] = \frac{\Pr(\boldsymbol{r}|c_j=1)}{\Pr(\boldsymbol{r}|c_j=0)} \tag{34}$$

for each $j \in [1,n]$. Define the *reflection coefficients* as

$$\widehat{\mathrm{SI}}[{c_j}^\perp] = \frac{1-\widehat{\mathrm{SI}}[c_j]}{1+\widehat{\mathrm{SI}}[c_j]}, \quad \widehat{\mathrm{SO}}[{c_j}^\perp] = \frac{1-\widehat{\mathrm{SO}}[c_j]}{1+\widehat{\mathrm{SO}}[c_j]} \tag{35}$$

for each $j \in [1,n]$. Forney proved that if $\widehat{\mathrm{SI}}[{c_j}^\perp]$ are used as input to an optimal bit-wise SISO decoding algorithm for $\mathcal{C}^\perp$,

then $\widehat{\mathrm{SO}}[c_j{}^{\perp}]$ are returned. Thus, any optimal bit-wise SISO decoding algorithm for $\mathcal{C}^{\perp}$ can be used to decode $\mathcal{C}$. Forney's proof depends on the orthogonality property of group characters (cf., [26])

$$\sum_{\boldsymbol{c}\in\mathcal{C}} e^{i\pi(\boldsymbol{v}\cdot\boldsymbol{c})} = \begin{cases} 2^k & \text{if } \boldsymbol{v}\in\mathcal{C}^{\perp} \\ 0 & \text{otherwise} \end{cases} \tag{36}$$

which is *not* a semi-ring property. While a result analogous to (36) can be proved for the $\min^{\star}$-sum semi-ring [27], such a result does not hold for the max-product or min-sum semi-rings. Thus, whereas optimal symbol-wise SISO decoding algorithms can be defined using dual codes, optimal algorithms with respect to codeword error probability cannot.

The duals of the first-order Reed-Muller codes are extended Hamming codes. In light of the above discussion, efficient optimal bit-wise SISO decoding algorithms for the extended Hamming codes can be obtained by using Hartmann and Rudolph's dual code procedure in conjunction with the SISO decoding algorithms presented in Section IV. Using other methods, Ashikhmin and Litsyn developed efficient SISO decoding algorithms for extended Hamming codes based on fast Walsh-Hadamard transforms in [16].

B. Generalized First-Order RM Codes

Generalized RM (GRM) codes are extensions of binary RM codes to q-ary alphabets. Different authors have considered constructions over both $\mathbb{F}_q$ (cf., [28]) and $\mathbb{Z}_q$ (cf., [29]). Irrespective of the algebra of the symbol alphabet, the q-ary first-order GRM code $\mathcal{RM}_q(1,m)$ has length q^m and can be defined recursively as

$$\mathcal{RM}_q(1,m) = \big\{(\boldsymbol{u},\boldsymbol{u}+\boldsymbol{v})\big|\boldsymbol{u}\in\mathcal{RM}_q(1,m-1), \boldsymbol{v}\in\mathcal{RM}_q(0,m-1)\big\} \tag{37}$$

where $\mathcal{RM}_q(0,m-1)$ is a q-ary repetition code and $\mathcal{RM}_q(1,1)$ is a length 2 q-ary SPC. As with binary first-order RM codes, the recursive definition of $\mathcal{RM}_q(1,m)$ can be reformulated under the coset code rubric. Specifically, $\mathcal{RM}_q(1,m)$ comprises q cosets of the code defined by concatenating identical codewords of $\mathcal{RM}_q(1,m-1)$.

A number of authors have developed optimal soft-in hard-out (i.e., maximum-likelihood) decoding algorithms for first-order GRM codes (cf., [24], [30]). Ashikhmin and Litsyn used the language of group algebras in order to formulate an FHT-based approach to optimal SISO decoding of first-order GRM codes over $\mathbb{F}_q$ [16]. The recursive coset SISO decoding algorithm for binary first-order RM codes is readily extended to first-order GRM codes over both $\mathbb{F}_q$ and $\mathbb{Z}_q$. The GFHT-based SISO of Section IV-C is similarly readily extended. The complexity of Ashikhmin and Litsyn's SISO as well as the q-ary generalizations of the SISO decoding algorithms described in Section IV grow as $O(q^2 n\log n)$. While minimal trellises for first-order GRM codes are unknown, it was conjectured in [16] that the complexity of trellis decoding of these codes grows as $O(q^2n^2)$.

TABLE III
COMPLEXITY OF TRELLIS VERSUS . COSET SISO DECODING FOR $\mathcal{NR}$

	©	ⓜ
Trellis	2280	1176
RCR	2304	1120
GFHT	1280	1248

C. Nordstrom-Robinson Code (NR)

The NR code $\mathcal{NR}$ is a (16, 256, 6) binary nonlinear code. The NR code is of particular theoretical interest as it has more codewords than any comparable known linear code. It is well known that $\mathcal{NR}$ can be defined as the union of 8 cosets of $\mathcal{RM}(1,4)$. Specifically, under the standard bit-ordering of $\mathcal{RM}(1,4)$ [31],

$$\mathcal{NR} = \bigcup_{i=1}^{8} \mathcal{RM}(1,4) + \boldsymbol{l}_i \tag{38}$$

where

$$\begin{aligned}
\boldsymbol{l}_1 &= (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),\\
\boldsymbol{l}_2 &= (0,0,0,0,0,0,1,1,0,1,1,0,0,1,0,1),\\
\boldsymbol{l}_3 &= (0,0,0,0,0,1,1,0,0,1,0,1,1,1,0,0),\\
\boldsymbol{l}_4 &= (0,0,0,0,0,1,0,1,0,0,1,1,0,1,1,0),\\
\boldsymbol{l}_5 &= (0,0,0,1,0,1,1,1,0,1,0,0,0,0,1,0),\\
\boldsymbol{l}_6 &= (0,0,0,1,0,1,0,0,0,0,0,1,1,0,1,1),\\
\boldsymbol{l}_7 &= (0,0,0,1,0,0,0,1,0,1,1,1,1,0,0,0),\\
\boldsymbol{l}_8 &= (0,0,0,1,0,0,1,0,0,0,1,0,1,1,1,0).
\end{aligned} \tag{39}$$

The technique presented in Section III can thus be used in conjunction with the efficient optimal SISO decoding algorithms for $\mathcal{RM}(1,m)$ presented in Section IV in order to define optimal SISO decoding algorithms for $\mathcal{NR}$.

Table III tabulates the complexity of the optimal SISO decoding algorithms for $\mathcal{NR}$ implied by: (i) a trellis; (ii) the coset technique of Section III using the RCR SISO decoding algorithm from Section IV-A for the $\mathcal{RM}(1,4)$ processing; and (iii) the coset technique of Section III using the GFHT-based SISO decoding algorithm from Section IV-C for the $\mathcal{RM}(1,4)$ processing. Note that the GFHT approach alone does not lead to an optimal decoding technique for $\mathcal{NR}$; it must be combined with the coset construction technique of Section III. Trellis decoding complexity was determined by explicitly counting operations in the least complex known trellis due to Reuven and Be'ery [32]. The complexities of the three algorithms are roughly comparable. Note that while a significant effort was required in order to derive the trellis in [32], the coset SISO decoding algorithms are readily defined. Note also that in contrast to trellis decoding, the coset SISO decoding algorithms naturally afford a parallelism that may be desirable for implementation in hardware [19].

D. Delsarte-Goethals Codes (DG)

The DG codes are a family of binary nonlinear codes consisting of cosets of first-order Reed-Muller codes. Specifically, for even $m\geq 4$ and for $2\leq\delta\leq m/2$, the Delsarte-Goethals code $\mathcal{DG}(m,\delta)$ is a

$$\left(2^m, 2^{(m-1)(m/2-\delta+1)+m+1}, 2^{m-1}-2^{m-1-\delta}\right) \tag{40}$$

nonlinear subcode of $\mathcal{RM}(2,m)$ comprising

$$L(m,\delta) = 2^{(m-1)(m/2-\delta+1)} \tag{41}$$

cosets of $\mathcal{RM}(1,m)$ [22]. The $\delta = m/2$ DS codes are Kerdock codes and, in particular, $\mathcal{DG}(4,2)$ is the Nordstrom-Robinson code studied in Section V-C.

Minimal trellises are, in general, unknown for the Delsarte-Goethals codes. It is, however, known that for $m \geq 6$ and $\delta \geq 3$, the minimum possible value of the maximum number of states in any trellis for $\mathcal{DG}(m,\delta)$ is [33]

$$2^{(m-1)(m/2-\delta+1)+m}. \tag{42}$$

By comparing (41) and (42) and noting that $n = 2^m$ for $\mathcal{DG}(m,\delta)$, it may be seen that the complexity of trellis decoding of $\mathcal{DG}(m,\delta)$ grows as

$$O\left(L(m,\delta)n^2\right). \tag{43}$$

Note that for the Kerdock codes, $L(m,m/2) = n/2$ and the complexity of trellis decoding grows as $O(n^3)$.

The technique presented in Section III can be used in conjunction with the efficient optimal SISO decoding algorithms for $\mathcal{RM}(1,m)$ presented in Section IV in order to define optimal SISO decoding algorithms for $\mathcal{DG}(m,\delta)$. Since the complexity of the proposed decoding algorithms for $\mathcal{RM}(1,m)$ grow as $O(n\log n)$, the complexity of the resulting SISO decoding algorithms for $\mathcal{DG}(m,\delta)$ grow as

$$O\left(L(m,\delta)n\log n\right). \tag{44}$$

Note that for Kerdock codes, the complexity of the proposed coset SISO decoding algorithms grow as $O\left(n^2\log n\right)$.

VI. Application: High-Rate Serially Concatenated Codes

A. Motivation

Very high-rate codes are of great interest for a number of practical applications including data storage and high-speed fiber links. The design of modern codes which simultaneously have very low error floors (e.g., $\lesssim 10^{-10}$ bit error rate) and very high rates (e.g., $\gtrsim 0.9$) is a particularly challenging problem of practical interest. Due to the inherent difficulty of simulating the performance of codes in the very low floor region, the design of such codes often relies on the principles of uniform interleaver analysis (cf., [34]–[37]). To review, on the additive white Gaussian noise (AWGN) channel with binary antipodal signaling, the bit error rate (P_b) and codeword error rate (P_{cw}) of a large class of modern codes vary asymptotically with block size as

$$P_b \sim N^{\alpha_{\max}} \quad P_{cw} \sim N^{\alpha_{\max}+1} \tag{45}$$

where N is the interleaver size and $\alpha_{\max}$ is the maximum exponent of N in an asymptotic union bound approximation. Note that the maximum exponent depends on both the specific code construction and constituent codes used. If the bit (codeword) error rate decays with N, then the code is said to exhibit interleaver gain in bit (codeword) error rate. Designs for codes with low floors require interleaver gain in both bit and codeword error rates and thus require $\alpha_{\max} \leq -2$.

Serially concatenated code constructions (i.e., codes composed of an inner code and outer code separated by a random-like interleaver) are well-suited to low-floor design because, provided the inner code is recursive, the maximum exponent of (45) is [34]

$$\alpha_{\max} = -\left\lfloor \frac{d_o+1}{2} \right\rfloor \tag{46}$$

where d_o is the minimum distance of the outer code. Since the rate of a serially concatenated code (SCC) is equal to the product of the rates of its constituent codes, the design of a high-rate, low-floor SCC requires a high-rate outer code satisfying $d_o \geq 3$. However, it is very challenging to find such outer codes for which there exist low-complexity optimal SISO decoding algorithms. To this end, Graell i Amat, *et al.* introduced a class of high-rate convolutional codes with optimal SISO decoding algorithms of moderate complexity based on their respective dual codes [38], [39].

An alternative approach to the design of high-rate, low-floor codes are the systematic with serially concatenated parity (S-SCP) codes proposed in [40], of which Jin, *et al.*'s systematic repeat accumulate (RA) codes [41] are an example. The S-SCP code structure can be viewed as a systematic code with a parity generating concatenated system that resembles an SCC. It was demonstrated in [40] that S-SCP codes have the same maximum error exponent and design rules as SCCs: codes constructed with a parity generating system composed of a recursive inner parity generator and an outer code satisfying $d_o \geq 3$ achieve interleaver gain in both bit and codeword error rates. In contrast to SCCs, good S-SCPs can be constructed with inner parity generators that have rate greater than 1 so that the rate of outer code can be *lower* than the overall code rate thus alleviating the challenge of finding high-rate, $d_o \geq 3$, outer codes with low-complexity SISO decoding algorithms.

The design of good high-rate, low-floor codes has thus been largely solved for the AWGN channel. However, the S-SCP design philosophy is not directly applicable to the large class of systems which have recursive channels. The term *recursive channel* is introduced to describe systems in which the aggregate of the modulation and (possible) precoding with the channel is recursive. Two examples of such systems are:

1) Continuous phase modulations over AWGN and fading channels.
2) Certain models for precoded magnetic recoding channels (e.g., EPR4 [42]).

In light of the above discussion, Fig. 7 illustrates two potential structures for high-rate, low-floor codes for use in systems with recursive channels. The first structure [Fig. 7(a)] consists of a high-rate modern code (e.g., an SCC or S-SCP) in serial concatenation with the recursive channel while the second structure [Fig. 7(b)] replaces the modern outer code with a classical code (e.g., convolutional or algebraic block code). While both structures have the potential to offer the desired bit and codeword error rate performance (i.e., low floors), the second structure may be more attractive for application in practical systems. Specifically, the use of a classical outer code for which there

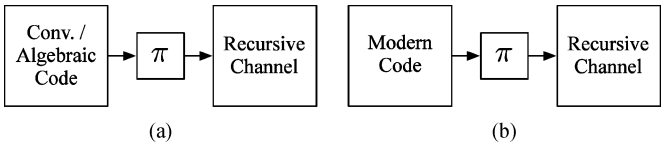

Fig. 7. Potential structures for high-rate, low-floor codes for use in systems with recursive channels. Random-like interleavers are labeled π.

exists a simple noniterative SISO decoding algorithm offers reductions in decoding complexity, decoding latency and required memory with respect to a modern, iteratively decoded, outer code. As with the design of SCCs for the AWGN channel, the design of such codes requires a high-rate, $d_o \geq 3$ outer classical code for which there exists a low-complexity optimal SISO decoding algorithm. The results of Section V-A indicate that the family of extended Hamming codes provide a set of such codes. In the remainder of this section, it is shown that extended Hamming codes offer an attractive alternative to the high-rate convolutional codes studied in [38], [39] for use as outer codes in serial concatenation with recursive inner channels.

B. Simulation Results and Discussion

In [39], Graell i Amat, *et al.* studied a serially concatenated coding scheme consisting of a high-rate convolutional outer code and a recursive rate-1 inner code corresponding to a simplified discrete-time model of a precoded EPR4 magnetic recording channel. Specifically, the recursive channel model comprises a $1/1 \oplus D^2$ precoder followed by a digital recording channel subject to intersymbol interference (ISI) with partial response polynomial $1 + D - D^2 - D^3$ followed finally by an AWGN channel. Note that the precoder and ISI can be jointly decoded on an 8-state trellis [19].

In this section, the performance of this scheme is compared to one which replaces the high-rate convolutional codes with extended Hamming codes. Specifically, 4 convolutional outer codes with input block size 4000 bits and respective rates 8/9, 9/10, 11/12, and 16/17 are compared to four algebraic outer codes composed of the following mixtures of extended Hamming codes:

- The mixture of 3 $\mathcal{RM}(3,5)$ and 69 $\mathcal{RM}(4,6)$ codewords resulting in a code with input block size 4011 bits and rate $4011/4512 = 0.8890 \approx 8/9$.
- The mixture of 56 $\mathcal{RM}(4,6)$ and 7 $\mathcal{RM}(5,7)$ codewords resulting in a code with input block size 4032 bits and rate $4032/4480 = 9/10$.
- The mixture of 30 $\mathcal{RM}(4,6)$ and 19 $\mathcal{RM}(5,7)$ codewords resulting in a code with input block size 3990 bits and rate $3990/4352 = 0.9168 \approx 11/12$.
- The mixture of 2 $\mathcal{RM}(4,6)$, 26 $\mathcal{RM}(5,7)$, and 3 $\mathcal{RM}(6,8)$ codewords resulting in a code with input block size 3975 bits and rate $3975/4224 = 0.9411 \approx 16/17$.

As reported in [39], the serially concatenated codes using convolutional codes utilize s-random interleavers [43]. The codes using extended Hamming outer codes utilize high sum-spread pseudorandom interleavers that were generated using the real-relaxation optimization method described in [44].

Fig. 8 compares the performance of the respective rate 8/9 and 11/12 codes while Fig. 9 compares the performance of the respective rate 10/11 and 16/17 codes. Note that the performance of the serially concatenated codes with convolutional outer codes is reported for 10 decoding iterations while the performance of the codes with mixed extended Hamming outer codes is reported for 10 and 20 decoding iterations. All curves correspond to $\min^\star$-sum processing (or its dual-domain equivalent [27]).

Observe in Figs. 8 and 9 that the codes constructed with mixed extended Hamming outer codes compare favorably in terms of performance to those with convolutional outer codes of similar rates and input block sizes. The mixed extended

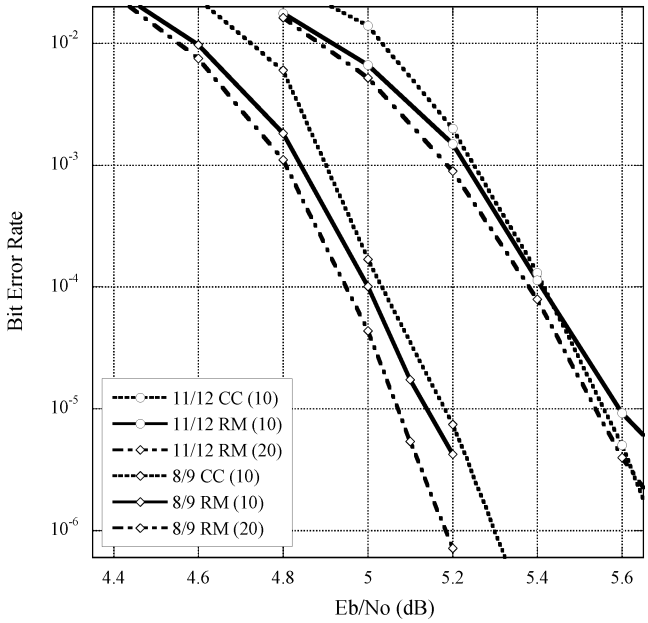

Fig. 8. Bit error rate performance of the respective rate 8/9 and 11/12 serially concatenated codes.

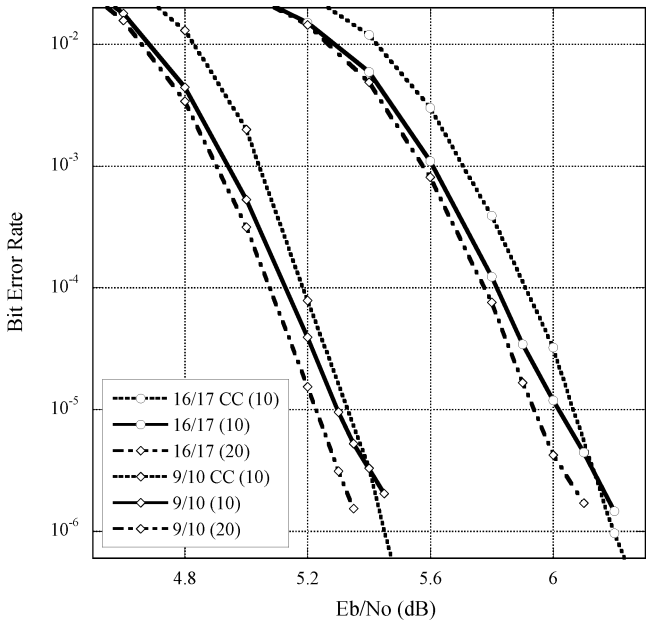

Fig. 9. Bit error rate performance of the respective rate 9/10 and 16/17 serially concatenated codes.

TABLE IV
AVERAGE NUMBER OF OPERATIONS PER DECODING ITERATION, FOR OPTIMAL SISO DECODING OF THE MIXED EXTENDED HAMMING CODES

	GFHT		RCR	
Rate	$+$ / Bit	$\min^\star$ / Bit	$+$ / Bit	$\min^\star$ / Bit
8/9	15.7	13.5	33.6	12.3
9/10	16.0	13.8	34.7	12.7
11/12	16.5	14.3	36.4	13.2
16/17	17.3	15.2	39.2	14.1

Hamming codes compare favorably to their high-rate convolutional code counterparts in terms of complexity also. Table IV tabulates the average number of addition and $\min^\star$ operations per input bit, per decoding iteration, required for optimal SISO decoding (using both the GFHT-based and recursive coset SISO decoding algorithms described in Section IV) for each of the mixed extended Hamming codes. The rate $k/(k+1)$ convolutional outer-codes in [39] were decoded on the 16- (for $k=8,9,10$) and 32- (for $k=16$) state trellises corresponding to their respective rate $1/(1+k)$ duals. Optimal SISO decoding on a rate $1/(1+k)$, 16- (32-) state trellis requires at least 96 (192) additions and 64 (128) comparisons per input bit, per decoding iteration. Thus, if one assumes that an addition and comparison operation have roughly the same complexity,[3] then the complexity of the proposed mixed extended Hamming code SISO decoding algorithms are approximately 5 to 10 times less than that of the respective high-rate convolutional code decoding algorithms proposed in [39].

VII. CONCLUSION AND FUTURE WORK

Motivated by the search for optimal SISO decoding algorithms with complexity less than that of trellis-based algorithms, this paper studied the construction of conditionally cycle-free graphical models for coset codes. It was shown that the proposed approach yields efficient SISO decoding algorithms for first-order RM as well as a number of related codes, e.g., extended Hamming codes. In light of (14) and (15), conditionally cycle-free graphical models defined using the procedure described herein have the potential to be more efficient than trellises when a code $\widetilde{\mathcal{C}}$ can be defined as the union of L cosets of a subcode $\mathcal{C}$ so that: (i) L is not too large; and (ii) an efficient SISO decoding algorithm for $\mathcal{C}$ is known. An interesting direction for future work is thus the search for other codes with these properties.

It was shown that the proposed efficient SISO decoding algorithms for extended Hamming codes are particularly useful in the context of high-rate serially concatenated codes. There are a number of interesting avenues of study concerning the application of the optimal SISO decoder for extended Hamming codes. Specifically, the recursive coset SISO decoding algorithm presented in this paper has a natural tree structure which may lead to high-speed architectures [45].

APPENDIX

1) Trellis Decoding of First-Order RM Codes: Forney demonstrated that the recursive coset representation of RM codes can be used to define efficient trellises in [4]. Fig. 10 illustrates how a trellis for $\mathcal{RM}(1,3)$ can be formed by properly combining two copies of a trellis for $\mathcal{RM}(1,2)$ (i.e., a length-4 SPC). The upper portion of the $\mathcal{RM}(1,3)$ trellis corresponds to $\mathcal{RM}(1,2)^2$ and the lower to $\mathcal{RM}(1,2)^2+(0,0,0,0,1,1,1,1)$. Observe that an unconventional bit ordering is employed in the trellis for $\mathcal{RM}(1,3)$.

Optimal SISO decoding of $\mathcal{RM}(1,2)$ and $\mathcal{RM}(1,3)$ is achieved by applying the forward-backward algorithm (FBA) to the trellises illustrated in Fig. 10. For $\mathcal{RM}(1,2)$, a total of 16 combining and 8 marginalizing operations are required to compute the pertinent forward and backward trellis state metrics. Soft outputs are then computed via a *completion* step (cf., [19]) that entails 8 combining and 4 marginalizing operations, hence (25). Similarly counting operations for $\mathcal{RM}(1,3)$ yields a total of 120 combining and 48 marginalizing operations. The values in Table I were obtained by counting operations in the trellises that result when the construction procedure illustrated in Fig. 10 is applied repeatedly (i.e., for increasing m).

For $\mathcal{RM}(1,3)$ in particular, the RCR decoding algorithm requires 24 less combining and 8 less marginalizing operations than trellis decoding. This complexity saving can be attributed to the latter decoding algorithm exploiting the recursive structure of $\mathcal{RM}(1,3)$ in the *construction* of the trellis diagram, but not in the *application* of the FBA to that trellis. Referring again to the data flow diagram illustrated in Fig. 6, let the metrics that serve as inputs to the $\mathcal{RM}(1,2)$ trellises be denoted

$$\begin{aligned}
\mathrm{SI}[c_i=0|\mathbf{0}] &= \mathrm{SI}[x_i=0] \copyright \mathrm{SI}[x_{i+4}=0] \\
\mathrm{SI}[c_i=1|\mathbf{0}] &= \mathrm{SI}[x_i=1] \copyright \mathrm{SI}[x_{i+4}=1] \\
\mathrm{SI}[c_i=0|\mathbf{1}] &= \mathrm{SI}[x_i=0] \copyright \mathrm{SI}[x_{i+4}=1] \\
\mathrm{SI}[c_i=1|\mathbf{1}] &= \mathrm{SI}[x_i=1] \copyright \mathrm{SI}[x_{i+4}=0] \qquad (47)
\end{aligned}$$

for $i \in [1,4]$. These metrics are computed twice when the FBA is applied to the trellis illustrated in Fig. 10 (once for the forward recursion and a once for the backward), accounting for 16 of the excess combining operations. The remaining 8 combining and marginalizing operations that are saved by the RCR algorithm can be accounted for in the completion step. The trellis-based decoder directly computes $\mathrm{SO}[x_i]$ for bit-level state metrics while the RCR decoder exploits the code structure to first compute $\mathrm{SO}[c_i=b|\boldsymbol{l}]$ (for $b \in [0,1]$ and $\boldsymbol{l} \in \{\mathbf{0},\mathbf{1}\}$), from which soft outputs are then derived via (24).

2) Forney's "Divide-by-Two" Decoding Algorithm: In [7], Forney described cycle-free realizations for first-order RM codes formed by clustering, or "sectionalizing," normal realizations of Hadamard transforms. Fig. 11 illustrates the minimal sectionalized realization for $\mathcal{RM}(1,3)$. Observe that as with the trellis realizations described above and the RCR decoding algorithm, bits are grouped according to the coset representation. Forney noted that these realizations specify the "divide-by-2" (DB2) *maximum likelihood* (i.e., soft-in, hard-out) decoding algorithms proposed in [4]. In this appendix, the optimal SISO decoding algorithms implied by these realizations are instead considered.

The FBA used for trellis decoding can be generalized to an "inward-outward" algorithm that passes messages first into the central 4-ary variable and then back out to the visible variables

[3]This assumption is reasonable provided a table-lookup is used for the correction term in the $\min^\star$ operator.

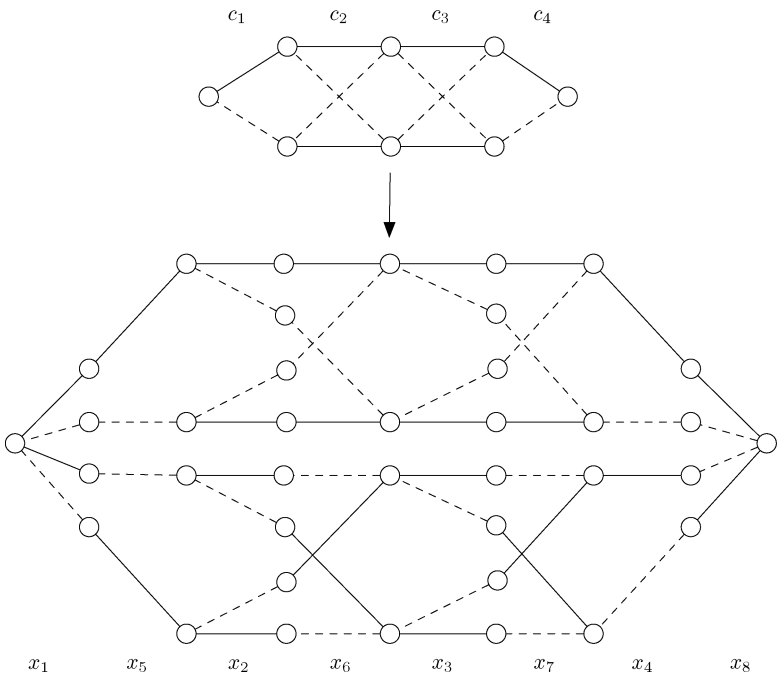

Fig. 10. Constructing a trellis for $\mathcal{RM}(1,3)$ by properly combining copies of a $\mathcal{RM}(1,2)$ trellis. Trellis stages are labeled by codeword coordinates. A solid (respectively, dashed) branch at the i^{th} corresponds to $c_i = 0$ (respectively, 1).

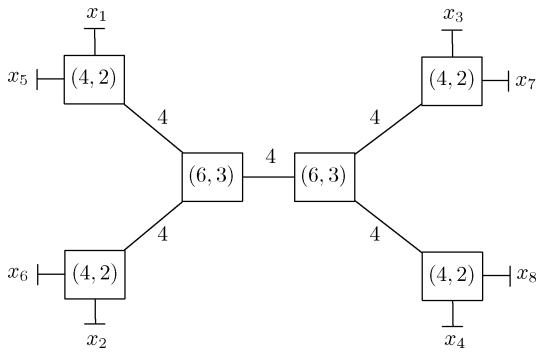

Fig. 11. Forney's "divide-by-two" (DB2) cycle-free graphical model for $\mathcal{RM}(1,3)$. Edges are labeled by the cardinality of the corresponding hidden variable alphabets. Local constraints are labeled by the length and dimension of the corresponding code (over $\mathbb{F}_2$).

that correspond to codeword bits in Fig. 11. Each (4, 2) constraint requires 4 combining operations for the inward computation, and 8 combining and 4 marginalizing operations for the outward computation. Similarly, each (6, 3) constraint requires 8 (respectively, 16) combining and 4 (respectively, 8) marginalizing operations for the inward (respectively, outward) computation. The total complexity of the decoding algorithm implied by the cycle-free graphical model for $\mathcal{RM}(1,3)$ illustrated in Fig. 11 is thus 96 combining and 40 marginalizing operations.

The values in Table I were obtained by counting operations in the DB2 decoding algorithms assuming that the local constraints are decoded via exhaustive combination and marginalization. For the $\mathcal{RM}(1,3)$ code, this yields identical decoding complexity to RCR decoding. However, for $m > 3$, the RCR decoding is less complex than the application of the general "inward-outward" algorithm to the DB2 cycle-free graphical models. Further complexity reductions could be obtained for the DB2 algorithms by exploiting the algebraic structure of the local constraints. In [15], Forney provided such detailed complexity tabulation for soft-in, hard-out decoding.

3) A Marginalizing Generalization of FHTs: The FHT used by Ashikhmin and Litsyn in [16] can be generalized such that it can perform both marginalization and combination. This generalization will be derived in two steps. The first and more straightforward generalization is illustrated in Fig. 12. In this figure, an 8-input FHT is generalized by replacing the standard operation of addition with any general operation $\odot$ for which inverses are defined. Note that a $\ominus$ on an input line denotes that the value on that line is replaced by its inverse. Dashed lines are used to indicate scalar-valued variables. For the standard case when $\odot$ is addition, the inverse of a value is simply its negative. This same procedure can be used to similarly generalize an FHT with any number of inputs.

Now consider an n-input FHT where the inputs x_k are defined in the following manner:

$$\gamma_k = \alpha_k \odot \beta_k^{-1}, \quad 1 \le k \le n. \tag{48}$$

If the operation $\odot$ is both associative and commutative, the outputs of the FHT can be shown to have the following form:

$$\begin{aligned}\Gamma_k = &(\alpha_{i_1} \odot \cdots \odot \alpha_{i_l} \odot \beta_{j_1} \odot \cdots \odot \beta_{j_m}) \odot \\ &(\beta_{i_1} \odot \cdots \odot \beta_{i_l} \odot \alpha_{j_1} \odot \cdots \odot \alpha_{j_m})^{-1}\end{aligned} \tag{49}$$

where $l + m = n$ and each α_k (β_k) appears exactly once within the expression. This forces the following statement to hold: if α_i is in the first parenthesized term, then β_i is required to be in the second parenthesized term and vice versa. Now, given

$$\begin{aligned}\Delta_k = &(\alpha_{i_1} \odot \cdots \odot \alpha_{i_l} \odot \beta_{j_1} \odot \cdots \odot \beta_{j_m}) \odot \\ &(\beta_{i_1} \odot \cdots \odot \beta_{i_l} \odot \alpha_{j_1} \odot \cdots \odot \alpha_{j_m})\end{aligned} \tag{50}$$

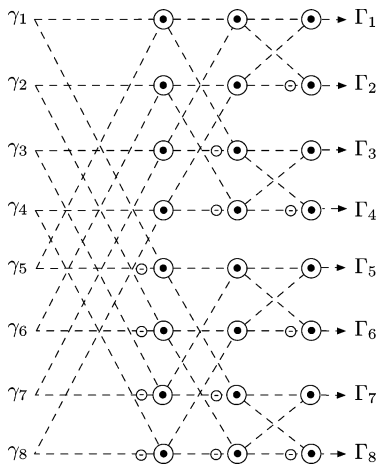

Fig. 12. Data flow of an 8-input generalized FHT.

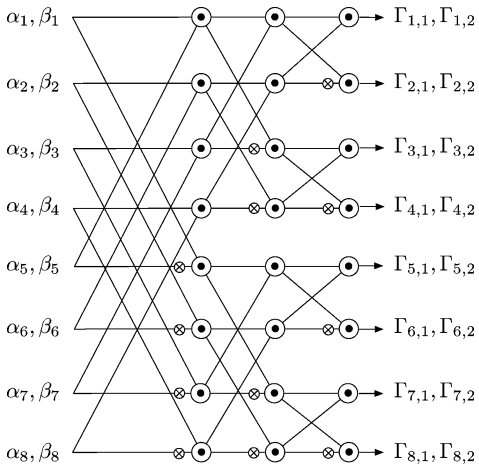

Fig. 13. Data flow of an 8-input GFHT.

the two parenthesized terms of (49) can be extracted from Γ_k to give the following two outputs:

$$\Gamma_{k,1} = (\alpha_{i_1} \odot \cdots \odot \alpha_{i_l} \odot \beta_{j_1} \odot \cdots \odot \beta_{j_m}),$$
$$\Gamma_{k,2} = (\beta_{i_1} \odot \cdots \odot \beta_{i_l} \odot \alpha_{j_1} \odot \cdots \odot \alpha_{j_m}). \quad (51)$$

A problem with this generalization arises when inverses are not defined for the operation $\odot$ (as in the case of many marginalizing operators, e.g., min and max). However, since the desired outputs $\{\Gamma_{k,1}\}$ and $\{\Gamma_{k,2}\}$ do not contain the inverse of any input, they can still be calculated by making a few subtle changes to the FHT shown in Fig. 12. First, the scalar input $\gamma_k = \alpha_k \odot \beta_k^{-1}$ is replaced by a vector $\boldsymbol{\gamma}_k = (\alpha_k, \beta_k)$. It can be verified that by replacing the inverses $\ominus$ in Fig. 12 with an operation $\otimes$, which "swaps" the values α_j and β_j, the desired outputs $\{\Gamma_{k,1}\}$ and $\{\Gamma_{k,2}\}$ are obtained. The swap can be defined as

$$\boldsymbol{\gamma}_j \cdot \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \quad (52)$$

As before, this entire generalization procedure can be applied to similarly generalize an FHT with any number of inputs.

This resulting form of the FHT is referred to throughout this paper as the generalized FHT (GFHT). The data flow of an 8-input GFHT is shown in Fig. 13 where solid lines are used to indicate vector-valued variables. Note that the $\ominus$ symbols have been replaced by $\otimes$ symbols, which indicate the swapping of values that is described in (52). Also note that the GFHT presented here works only when the original inputs are of the form (48) and the desired outputs are of the form (51), which is the case for the FHTs used to decode the first-order RM codes in [16].

REFERENCES

[1] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.
[2] J. K. Wolf, "Efficient maximum-likelihood decoding of linear block codes using a trellis," *IEEE Trans. Inf. Theory*, vol. 24, pp. 76–80, 1978.
[3] G. D. Forney Jr., "Coset codes I: Introduction and geometrical classification," *IEEE Trans. Inf. Theory*, vol. 34, pp. 1123–1151, 1988.
[4] G. D. Forney Jr., "Coset codes II: Binary lattices and related codes," *IEEE Trans. Inf. Theory*, vol. 34, pp. 1152–1187, 1988.
[5] A. Vardy, "Trellis structure of codes," in *Handbook of Coding Theory*, V. S. Pless and W. C. Huffman, Eds. Amsterdam, The Netherlands: Elsevier, 1999.
[6] N. Wiberg, "Codes and Decoding on General Graphs," Ph.D., Linköping Univ., , Sweden, 1996.
[7] G. D. Forney Jr., "Codes on graphs: Normal realizations," *IEEE Trans. Inf. Theory*, pp. 520–548, Feb. 2001.
[8] S. M. Aji and R. J. McEliece, "The generalized distributive law," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 325–343, Mar. 2000.
[9] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, pp. 498–519, Feb. 2001.
[10] T. R. Halford and K. M. Chugg, "The extraction and complexity limits of graphical models for linear codes," *IEEE Trans. Inf. Theory*, vol. 54, no. 9, pp. 3884–3906, Sep. 2008.
[11] N. Kashyap, "On minimal tree realizations of linear codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 8, pp. 3501–3519, Aug. 2009.
[12] N. Kashyap, "Constraint complexity of realizations of linear codes on arbitrary graphs," *IEEE Trans. Inf. Theory*, vol. 55, no. 11, pp. 4864–4877, Nov. 2009.
[13] J. Heo and K. M. Chugg, "Constrained iterative decoding: Performance and convergence analysis," in *Proc. Asilomar Conf. Signals, Systems, Comp.*, Pacific Grove, CA, Nov. 2001, pp. 275–279.
[14] Y. Wang, R. Ramesh, A. Hassan, and H. Koorapaty, "On MAP decoding for tail-biting convolutional codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Ulm, Germany, Jun. 1997, p. 225.
[15] G. D. Forney Jr., "Codes on graphs: Constraint complexity of cycle-free realizations of linear codes," *IEEE Trans. Inf. Theory*, vol. 49, no. 7, pp. 1597–1610, Jul. 2003.
[16] A. Ashikhmin and S. Litsyn, "Simple MAP decoding of first-order Reed-Muller and Hamming codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1812–1818, Aug. 2004.
[17] P. Robertson, E. Villebrum, and P. Hoeher, "A comparison of optimal and suboptimal MAP decoding algorithms operating in the log domain," in *Proc. Int. Conf. Commun.*, Seattle, WA, 1995, pp. 1009–1013.
[18] A. J. Viterbi, "Justification and implementation of the MAP decoder for convolutional codes," *IEEE J. Sel. Areas Commun.*, vol. 16, pp. 260–264, Feb. 1998.
[19] K. M. Chugg, A. Anastasopoulos, and X. Chen, *Iterative Detection: Adaptivity, Complexity Reduction, and Applications*. Boston, MA: Kluwer Academic , 2001.
[20] G. D. Forney Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.
[21] A. P. Hekstra, "An alternative to metric rescaling in viterbi decoders," *IEEE Trans. Commun.*, vol. 37, no. 11, Nov. 1989.
[22] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam: North-Holland, 1978.
[23] A. Lafourcade and A. Vardy, "Optimal sectionalization of a trellis," *IEEE Trans. Inf. Theory*, vol. 42, no. 3, pp. 689–703, May 1996.
[24] A. J. Grant and R. D. V. Nee, "Efficient maximum-likelihood decoding of Q-ary modulated Reed-Muller codes," *IEEE Commun. Lett.*, vol. 2, no. 5, pp. 134–136, May 1998.
[25] C. R. P. Hartmann and L. D. Rudolph, "An optimum symbol-by-symbol decoding rule for linear codes," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 5, pp. 514–517, Sep. 1976.

[26] W. Rudin, *Fourier Analysis on Groups*. New York: Wiley , 1967.
[27] G. Montorsi and S. Benedetto, "An additive version of the SISO algorithm for the dual code," in *Proc. IEEE Int. Symp. Inf. Theory*, Washington, DC, Jun. 2001.
[28] P. Ding and J. K. Key, "Minimum-weight codewords as generators of generalized Reed-Muller codes," *IEEE Trans. Inf. Theory*, no. 6, pp. 2152–2158, Sep. 2000.
[29] K. G. Paterson, "Generalized Reed-Muller codes and power control in OFDM modulation," *IEEE Trans. Inf. Theory*, vol. 46, no. 1, pp. 104–120, Jan. 2000.
[30] K.-U. Schmidt and A. Finger, "Simple maximum-likelihood decoding of generalized first-order Reed-Muller codes," *IEEE Commun. Lett.*, vol. 9, no. 10, pp. 912–914, Oct. 2005.
[31] J.-P. Adoul, "Fast ML decoding for the Nordstrom-Robinson code," *IEEE Trans. Inf. Theory*, vol. 33, no. 6, pp. 931–933, Nov. 1987.
[32] I. Reuven and Y. Be'ery, "Entropy/length profiles, bounds on the minimal covering of bipartite graphs, and trellis complexity of nonlinear codes," *IEEE Trans. Inf. Theory*, vol. 44, no. 2, pp. 580–598, Mar. 1998.
[33] Y. Shany, I. Reuven, and Y. Be'ery, "On the trellis representation of the Delsarte-goethals codes," *IEEE Trans. Inf. Theory*, vol. 44, no. 4, Jul. 1998.
[34] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding," *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 909–926, May 1998.
[35] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Trans. Commun.*, vol. 44, pp. 591–600, May 1996.
[36] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 408–428, Mar. 1996.
[37] D. Divsalar and F. Pollara, "Hybrid Concatenated Codes and Iterative Decoding," JPL-TDA, Tech. Rep. 42–130, 1997.
[38] A. G. i Amat, S. Benedetto, and G. Montorsi, "Optimal high-rate convolutional codes for partial response channels," in *Proc. Globecom Conf.*, Taipei, Taiwan, Nov. 2002, vol. 2, pp. 1031–1036.
[39] A. G. i Amat, G. Montorsi, and S. Benedetto, "New high-rate convolutional codes for concatenated schemes," in *Proc. Int. Conf. Commun.*, New York, May 2002, vol. 3, pp. 1661–1666.
[40] K. M. Chugg, P. Thiennviboon, G. D. Dimou, P. Gray, and J. Melzer, "A new class of turbo-like codes with universally good performance and high-speed decoding," in *Proc. IEEE Military Comm. Conf.*, Atlantic City, NJ, Oct. 2005.
[41] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat-accumulate codes," in *Turbo Code Conf.*, Brest, France, 2000.
[42] J. Li, K. R. Narayanan, E. Kurtas, and C. N. Georghiades, "On the performance of high-rate TPC/SPC codes and LDPC codes over partial response channels," *IEEE Trans. Commun.*, vol. 50, no. 5, pp. 723–734, May 2002.
[43] D. Divsalar and F. Pollara, "Turbo codes for PCS applications," in *Proc. Int. Conf. Commun.*, Seattle, WA, 1995, p. 5459.
[44] S. Crozier, "New high-spread high-distance interleavers for turbo-codes," in *Proc. 20th Biennial Symp. Commun.*, Kingston, ON, Canada, 2000, pp. 3–7.
[45] P. A. Beerel and K. M. Chugg, "A low latency SISO with application to broadband turbo decoding," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 5, pp. 860–870, May 2001.

Thomas R. Halford received the B.A.Sc. degree in engineering physics from Simon Fraser University, Burnaby, Canada, in 2001, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, in 2007.

He spent summer 2005 visiting the Natural Language Processing Group at IBM's T. J. Watson Research Center. In May 2007, he was offered a Director's Postdoctoral Fellowship at Los Alamos National Labs, which he declined in order to join TrellisWare Technologies, Inc., San Diego, CA, where he now leads the Emerging Technologies Group. His current research interests include ad hoc networks, situational awareness, and modulation and coding techniques for next-generation military waveforms.

Keith M. Chugg (S'88-M'95-F'10) received the B.S. degree (high distinction) in engineering from Harvey Mudd College, Claremont, CA, in 1989 and the M.S. and Ph.D. degrees in electrical engineering (EE) from the University of Southern California (USC), Los Angeles, in 1990 and 1995, respectively.

During the 1995–1996 academic year, he was an Assistant Professor with the Electrical and Computer Engineering Department, The University of Arizona, Tucson. In 1996 he joined the EE Department, USC, in 1996 where he is currently a Professor. His research interests are in the general areas of signaling, detection, and estimation for digital communication and data storage systems. He is also interested in architectures for efficient implementation of the resulting algorithms. Along with his former Ph.D. students, A. Anastasopoulos and X. Chen, he is coauthor of the book *Iterative Detection: Adaptivity, Complexity Reduction, and Applications* (New York: Kluwer Academic). He is a co-founder of TrellisWare Technologies, Inc., where he is Chief Scientist.

Dr. Chugg has served as an Associate Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS and was Program Co-Chair for the Communication Theory Symposium at Globecom 2002. He received the Fred W. Ellersick award for the Best Unclassified Paper at MILCOM 2003 and the ASEE Frederick Emmons Terman Award in 2008.

Marcus T. Urie (S'05) received the B. S. degree in electrical engineering from Brigham Young University, Provo, UT, in 2005, and the M. S. degree in electrical engineering from the University of Southern California, Los Angeles, in 2007, where he is currently a doctoral candidate.

In 2010 he joined TrellisWare Technologies, Inc., San Diego, CA as a Communication Systems Engineer. His research interests include high-speed decoding architectures for modern codes, sensor localization, and novel applications of iterative detection in military communications systems.