RESEARCH ARTICLE OPEN ACCESS

# Factor Graph Approach for Decoding Reed Muller Codes

Sarah Anjum*
**(Assistant Professor, University Polytechnic, Aligarh Muslim University, Aligarh, India.*

**ABSTRACT**
Factor graphs basically work on the idea of marginalization i.e. sub-dividing a larger function into a number of smaller functions whose contribution to the solution of the problem are considered accordingly. This paper uses Forney factor graph, which is a bipartite graph that models a system into function nodes and variable nodes. The messages are passed to and fro the two sets of nodes via specific message update rules chosen beforehand and the phenomena may be based on sum product or the max product algorithm. The algorithm converges to a decoded output after either a fixed number of iterations or is made to terminate after a threshold is achieved.

***Keywords*:** Factor graphs, Maximum Likelihood Decoding, Max-product algorithm, Reed Muller Codes, Sum-product algorithm.

Date of Submission: 17-12-2017 Date of acceptance: 08-01-2018

## I. INTRODUCTION

Graphical models have always been an important part of the works of an engineer as they are easy to manipulate and self-descriptive. The different graphical models that have been popularly used to describe the events and experiments are circuit diagrams, signal flow graphs; several block diagrams and trellis diagrams. The graph theory[1][6] comprises of the relevant algorithms which allow easy and systematic use of graphical models. In artificial intelligence, statistics and neural networks, Bayesian networks and Markov random fields are have been used to describe the non-probabilistic models. The formulation of Viterbi decoding algorithm is done by a trellis diagram and the Gibbs sampling is required for prediction and projection for Markov fields.

Graphical models have been popular because of the ease with which they provide the solution to a large number of signal processing problems and mainly coding theory from which the factor graph find their origin. The use of graph in solving the signal processing problems results in reduction of complexity. The algorithms involved use minimum knowledge and give approximate results by a simple phenomenon of message passing between the different parts of the factor graph. Factor graphs have been commonly used for decoding Low Density Parity Check codes with the help of a series of steps known as Sum-Product algorithm. The algorithm was invented by Gallager[6][8] in 1970s and is still the standard decoding algorithm for such codes.The two main algorithms operated on factor graphs are sum-product (or belief or probability propagation) algorithm and the min-sum (or max product algorithm). These operations mainly find use in error correcting codes as these codes are mostly used in modern communication systems. The main reason behind the popularity of the Low Density Parity Check codes is the accessibility of the coding scheme round the globe. Previously, the Turbo codes were being used in the communication and transmission systems but they were patented in the western regions and soon got to be outshined by the LDPC codes with the advent of the simple decoding technique introduced.

Another scientist Tanner introduced rather modified the factor graph describing LDPC codes thereby replacing the parity checks with generalized components. He further introduced the min-sum algorithm which is similar to the sum-product algorithm but the cumulative technique is slightly modified which will be described in this paper. The sum product and max product algorithm [4][5] originate from the coding theory. For decoding the turbo codes Trellis diagrams use the BCJR algorithm and the Viterbi algorithm which was the main technique to decode the practical coding schemes. Factor graphs had been reinvented a number of times and with successive input the full potential of the theory was exposed to the coding world. Whether it was the decoding of turbo codes or Low Density Parity Check codes the operations involved take a fixed path of solution i.e. the message passing phenomena over the relevant graph. The Viterbi and BCJR algorithms are examples of the same algorithm and involve message passing over the generalized graph. Berrou[6][8][10] invented the turbo codes and made the method of iterative decoding famous for the coding world. Factor graphs after the ideas of Wiberg could be used for iterative decoding over channels with memory. Moreover, the sum-product algorithm which was earlier being used for cycle free factor graphs could now be used for the factor graphs with cycles.

The basic algorithm for the message passing phenomenon comprises of three different parts [5][7] i.e. the forward-backward algorithm, RLS algorithm and fast Fourier (FFT) algorithm. The forward-backward algorithm and Kalman filtering are found to be the same idea in two different representations. They find use in a variety of applications from sequential processing to iterative operations that involve a number of parameter alteration and yield mostly accurate results with reduced complexity. Earlier a common communication system had sequentially arranged subtasks of synchronization, equalization and decoding which were now designed to interact via multiple feedback loops with the advent of the turbo codes. The state space models [7][11] follow the trends of the Factorial hidden Markov modelsand operated on the product of several state spaces. A variety of signal processing problems can be pictorially represented using a suitable factor graphs and can be solved with the message passing algorithm such as the summary propagation algorithm.Coding out of all the signal processing problems happens to be the most important field which finds its use in the factor graphs along with the use in machine learning, statistics and statistical physics. There have been many algorithms introduced operated by message passing over a generalized factor graph which later were found to be special cases of sum product and max-product algorithm.Factor graph finds use in the functions involving a large number of discrete and/or continuous variables. A large variety of model based detection and estimation problems involve the usage of factor graphs via non trivial algorithm based on local computations that can easily be merged with the building blocks of the system model. The classical Gaussian and gradient techniques over expectation maximization to sequential Monte Carlo [11][12] method which is also known as particle filters, can be easily mixed and matched along with the models based on factor graphs.

## II. FACTOR GRAPHS

A FORNEY FACTOR GRAPH [1] is a diagram as in Figure 1 that represents the factorization of a function of several variables. Assume, for example, that some function *f (u, w, x, y , z )* can be factored as

$$f(a,b,c,d,e) = f_1(a,b,c)\,f_2(c,d,f)\,f_3(e) \qquad (1)$$

This factorization is expressed by the Forney Factor Graph shown in Figure 1. In general, an FFG [18][19]consists of nodes, edges and half edges that are connected only to one node and the Forney Factor graph is defined by the following rules:

- There is a unique node for every factor.
- There is a unique edge or half edge for every variable.
- The node representing some factor g is connected with the edge, or half edge representing some variable *x* if and only if *g* is a function of *x*.

The factors i.e. the sub-functions with smaller number of variables are known as local functions whereas product of all the local functions is known as the global function [13]. The assumption that has been taken care of in here is that the maximum number of factors in which a variable can appear is two. But the assumption can be by-passed to the application comfort. In the following figure the local functions are $f_1$, $f_2$ and $f_3$ while the global function is *f*which is the product of the local functions.

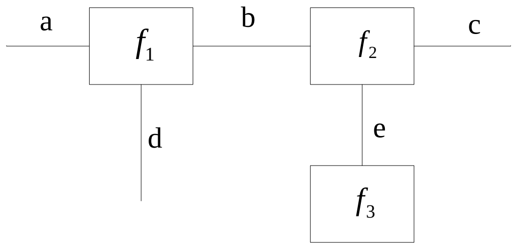

**Fig 1:** Forney Factor Graph

All the variables are assigned particular values before the message passing starts. All the possible configurations are made into the set which make up the domain of the global function f. In Figure 1 the global function represented by the Forney factor graph consists of five variables and thus the configuration set consists of five tuple configurations. If the variables are chosen to be binary then the configuration space is the set {0, 1}. The configuration space consists of non-negative real numbers i.e. $R^5$. A configuration is called valid [6][15] if satisfy the condition to be non-zero. In a fixed configuration, every variable has some definite value denoted by small letters while the functions are denoted by capital letters. If a variable takes some values in a set then it can be denoted as follows:

$$Y:\mu \rightarrow Y:\theta \rightarrow x = X(\theta) \qquad (2)$$

An alphabet is the collective set of any block codes comprising of all the possible code words. Any error correcting code word of block length n is a subset of that alphabet. Let X be some alphabet of the error correcting code word of length n and S be the subset of $A_n$. The code is said to be linear if the alphabet set is a field, usually a finite field and S is a subspace of the vector space $F^n$. If the code is binary then the field is denoted as $F^2$ i.e. the set {0,1} with modulo-2 arithmetic. By elementary linear algebra, any linear code cab be written as

$$C = \{x \varepsilon F n : x^*H^T = 0\} \qquad (3)$$

And as

$$C = \{u^*G : u \in F^{\,k}\} \qquad (4)$$

where*H* and *G* are matrices over *F* and where *k* is the dimension of *C* (as a vector space over *F* ). A matrix *H* as in (5) is called a parity-check matrix [16][17] for *C*, and a *k* × *n* matrix *G* is called a generator matrix for *C*. It is to state that as per the encoding rule a vector u of the information symbols belonging to the field F is mapped to the corresponding code word x.

In case of a binary (7,4,3) Hamming code, the code has length n=7, dimension k=4 and minimum Hamming distance 3. The parity check matrix H of the code may be defined as follows:

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \quad (5)$$

Thus a code word needs to satisfy the following

$$\boldsymbol{F^n} \rightarrow \{0, 1\} : x \rightarrow \begin{Bmatrix} \mathbf{1}, \boldsymbol{if\ x \in C} \\ \mathbf{0},\ \boldsymbol{else} \end{Bmatrix} \qquad (6)$$

From the parity check matrix, the check equations can be written as follows:

$$\begin{aligned} F(x_1, \ldots, x_n) &= \delta(x1 \oplus x2 \oplus x3 \oplus x5) \\ &= \delta(x2 \oplus x3 \oplus x4 \oplus x6) \\ &= \delta(x3 \oplus x4 \oplus x5 \oplus x7) \end{aligned}$$

Here each row of the parity check matrix is depicted by each factor and the symbol stands for modulo 2 additions. The Forney factor graph for (7,4,3) binary Hamming code is shown in figure 2 in which both the check nodes and variable nodes are shown.The connections between the two sets are determined by the parity check matrix.

The summary product algorithm is applicable for cyclic as well as non-cyclic factor graphs. The decoding algorithms of a typical trellis based factor graphs implement sum product rule. In particular, when applied to a trellis, the sum-product algorithm becomes the BCJR algorithm and the max-product algorithm (or the min-sum algorithm applied in the logarithmic domain) becomes a soft-output version of the Viterbi algorithm. The sum-product, the max-product algorithm as well as various guesstimates of these algorithms is used for decoding a typical Low Density Parity Check Codes.

The turbo code encoder consists of two (or more) systematic block codes which share message data via interleavers. In its most conventional realization, the codes are obtained from recursive systematic convolutional (RSC) codes - but other codes can be used as well. Akey development in turbo codes is the iterativedecoding algorithm. In the iterative decoding algorithm, decoders for each constituent encoder take turns operating on the received data. Each decoder produces an estimate of the probabilities of the transmitted symbols. The decoders are thus softoutput decoders. Probabilities of the symbols from one encoder known as extrinsic probabilitiesarepassed to the other decoder (in the symbol order appropriate for the encoder), where they are used as prior probabilities for the other decoder. The decoder thus passes probabilities back and forth between the decoders, with each decoder combining the evidence it receives from the incoming prior probabilities [20][21] with the parity information provided by the code. After some number of iterations, the decoder converges to an estimate of the transmitted code word. Since the output of one decoder is fed to the input of the next decoder, the decoding algorithm is called a turbo decoder.

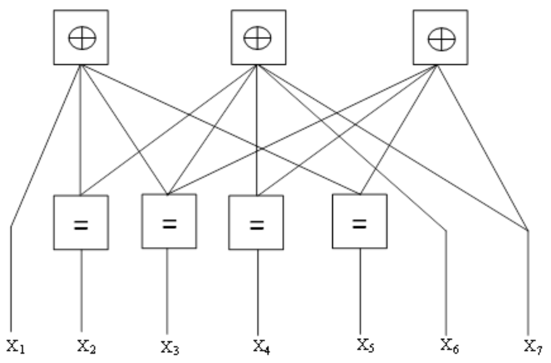

**Fig 2:** An FFG for the (7, 4, 3) binary Hamming code

A channel model is a family *p(y |x)* of probability distributions over a block $y = (y_1, \ldots, y_n)$ of channel output symbols given any block $x = (x_1, \ldots, x_n)$ of channel input symbols. Connecting the factor graph (Tanner graph) of a code *C* with the factor graph of a channel model *p(y | x)* results in a factor graph of the joint likelihood function *p(y | x)I (x)*. If we assume that the code-words are equally likely to be transmitted, we have for any fixed received block *y*

$$p(x \mid y) = \frac{p(y \mid x)\,p(x)}{p(y)} \qquad (7)$$

$$\alpha \quad p(y \mid x) I_C(x)$$

The joint code/channel factor graph thus represents the a posteriori joint probability of the coded symbols $X_1, \ldots, X_n$.

So far, we have freely introduced supplementary variables to obtain nicely structuredgraphical models. Now we will consider the elimination of variables [6][23][39]. For example, for some discrete probability mass function

$f(x_1, \ldots, x_8)$, we might be interested in the marginal probability

$$p(x_4) = \sum_{x_1,x_2,x_3,x_5,x_6,x_7} f(x_1,\ldots,x_8) \quad (8)$$

Or, for some nonnegative function $f(x_1,\ldots,x_8)$, we might be interested in

$$p(x_4) = \max_{x_1,x_2,x_3,x_5,x_6,x_7} f(x_1,\ldots,x_8) \quad (9)$$

The general idea is to get rid of some variables by some "summary operator," and the most popular summary operators are summation (or integration) and maximization (or minimization).
Note that only the valid configurations contribute to a sum as in the above equation, and (assuming that *f* is nonnegative) only the valid configurations contribute to maximization. Now *f* can be written as

$$f(x_1,\ldots,x_8) = (f_1(x_1) f_2(x_2) f_3(x_1,x_2,x_3,x_4))$$
$$(f_4(x_4,x_5,x_6) f_5(x_5)(f_6(x_6,x_7,x_8) f_7(x_7)))$$
$$(10)$$

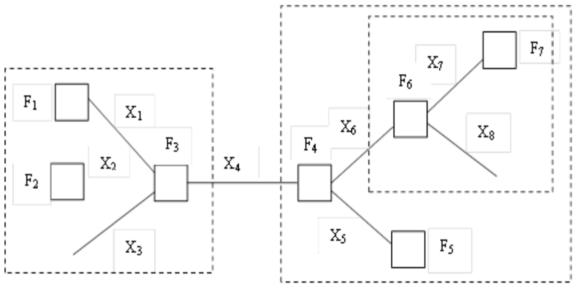

**Fig 3:** Elimination of variables: "closing the box" around subsystems

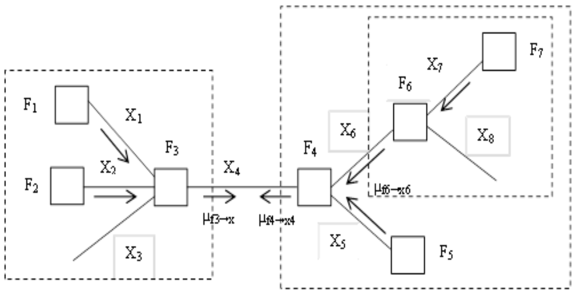

**Fig 4:** ''Summarized" factors as "messages" in the Forney Factor Graph

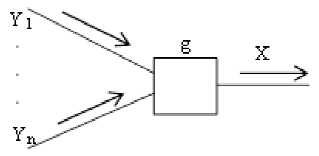

**Fig 5:** Messages along a generic edge

Note that the brackets in (10) correspond to the dashed boxes in Figure 4. Inserting (8) into (9) and applying the distributive law yields (11).

$$p(x_4) = \left( \sum_{x_1} \sum_{x_2} \sum_{x_3} \begin{matrix} f_3(x_1,x_2,x_3,x_4) \\ f_1(x_1) f_2(x_2) \end{matrix} \right) .$$
$$\begin{pmatrix} \sum_{x_5} \sum_{x_6} \boldsymbol{f_4(x_4,x_5,x_6) f_5(x_5)} \\ \left(\sum_{x_7} \sum_{x_8} \boldsymbol{f_6(x_6,x_7,x_8) f_7(x_7)}\right) \end{pmatrix} \quad (11)$$

This expression can be interpreted as enclosing the dashed boxes in Figure 3 by summarizing over their internal variables. The factor $\mu_{f3\to x4}$ is the summary of the big dashed box on the left in Figure 4; it is a function of $x_4$ only. The factor $\mu_{f6\to x6}$ is the summary of the small dashed box on the right in Figure 4; it is a function of $x_6$ only. Finally, the factor $\mu_{f4\to x4}$ is the summary of the big dashed box right in Figure 4; it is a function of $x_4$ only. The resulting expression

$$p(x_4) = \mu_{f3\to x4}(x_4) \cdot \mu_{f4\to x4}(x_4) \quad (12)$$

Corresponds to the FFG of Figure 4 with the dashed boxes closed. Replacing all sums in (11) by maximizations yields an analogous decomposition of (10).

A global marginalization (by summation/integration or by maximizationmay be obtained by successive local marginalization of subsystems. This marginalization makes up an integral stage of the summary product algorithm [24][40]. Towards this end, the summaries i.e., the terms in brackets in (11) are taken to be the messages that are sent out of the corresponding box, as is illustrated in Figure 5. Message out of a terminal node are defined (e.g., $f_1$) as the corresponding function itself [e.g., $f_1(x_1)$]. Open half edges (such as $x_3$) do not carry a message towards the (single) node attached to them; alternatively, they may be thought of as carrying as message a neutral factor 1 i.e. unity. It is then easy to verify that allsummaries/messages in Figure 3 are formed according to the following general rule.

## III. MESSAGE PASSING ALGORITHM

### A. SUM-PRODUCT RULE

In Figure 3 the message out of some node $g(x, y_1, \ldots, y_n)$ along the branch *x* is the function

$$\mu_{g\to x}(x) \triangleq \sum_{y_1} \ldots \sum_{y_n} g(x, y_1, \ldots, y_n) \mu_{y_1\to g}(y_1) \ldots \mu_{y_n\to g}(y_n)$$
$$(13)$$

where$\mu_{yk}\to g$(which is a function of $y_k$) is the message that arrives at *g* along the edge $y_k$. The message out of a factor node $g(x, \ldots)$ along the edge *x* is the product of $g(x, \ldots)$ and all messages towards *g* along all edges except *x* summarized over all variables except *x* .

We have thus seen the following:

- Summaries/marginals such as (10) and (11) can be computed as the product of two messages as in (13).
- Such messages are summaries of the subsystem they are ahead of.
- Computation of messages involves the marginalization of other messages using summary-product algorithm. However, the messages out of terminal nodes do not require such estimations.

It is easy to see that this procedure to compute summaries is not restricted to the example of Figure 3 but applies whenever the factor graph is free of cycles. In its general form, the summary-product algorithm [3][12] computes two messages for each edge in the graph, one in each direction. Each message is computed according to the summary- product rule [typically the sum-product rule (13)]. A sharp distinction divides graphs with cycles from graphs without cycles. If the graph has no cycles, then it is efficient to begin the message computation from the leaves and to successively compute messages as their required incoming messages become available. In this way, each message is computed exactly once. It is then obvious from the previous section that summaries/marginals as in (9) or (10) can be computed as the product of messages as in (11) simultaneously for all variables. The final result is the a posteriori probability $p(x_l|y_1, \ldots, y_4)$ for $i = 1,\ldots,4$. The channel output symbols $Y_l$are binary, and the four nodes in the channel model represent the factors

$$p(y_1 \mid x_1) = \begin{cases} 0.9, \text{if } y_1 = x_1 \\ 0.1, \text{if } y_1 \neq x_1 \end{cases} \quad (14)$$

for$i = 1, \ldots, 4$. If $(Y_1, \ldots, Y_4) = (y_1, \ldots, y_4)$ is known (fixed), the factor graphrepresents the a posteriori probability $p(x_1, \ldots, x_4|y_1, \ldots, y_4)$, up to a scale factor, cf. (12). The messages areas computed according to the sum-product rule (13). The final result is the per-symbol a posteriori probability $p(x_l|y_1, \ldots, y_4)$ for $i = 1, \ldots, 4$; according to (12), this is obtained as (a suitably scaled version of) the product of the two messages alongthe edge $X_l$.

## B. MAX PRODUCT RULE

Assume we wish to maximize some function $f(x_1,\cdots,x_n)$, i.e., we wish to compute

$$(\hat{x}_1,\cdots,\hat{x}_n) = \arg\max_{x_1,\ldots,x_n} f(x_1,\cdots,x_n) \qquad (15)$$

where we assume that f has a maximum. Note that

$$\hat{x}_k = \arg\max_{x_k} \hat{f}_k(x_k) \qquad (16)$$

$$\hat{f}_k(x_k) \triangleq \max_{\substack{x_1,\cdots,x_n \\ \text{except } x_k}} f(x_1,\ldots,x_n) \quad (17)$$

If $f(x_1,\ldots,x_n)$has a cycle-free factor graph [22][24][41], the function (17) can be computed by the max-product algorithm. For example, assume that $f(x_1,\ldots,x_7)$ can be written as

$$f(x_1,\cdots,x_7) = f_1(x_1)f_2(x_2)f_3(x_1,x_2,x_3)$$
$$f_4(x_4)f_5(x_3,x_4,x_5)f_6(x_5,x_6,x_7)f_7(x_7)$$
(18)

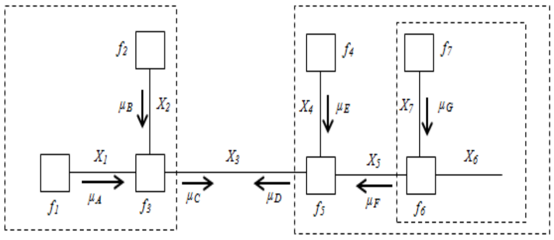

**Fig 6:** Summarised factors as messages in factor graph

Assume that we wish to compute $\hat{f}_3(x_3)$. It is easily verified that

$$\hat{f}_3(x_3) = \mu_C(x_3)\mu_D(x_3) \, (19)$$

With

$$\mu_C(x_3) \triangleq \sum_{x_1,x_2} f_1(x_1)f_2(x_2)f_3(x_1,x_2,x_3) \quad (20)$$

And

$$\mu_D(x_3) \triangleq \sum_{x_4,x_5} f_4(x_4)f_5(x_3,x_4,x_5)\mu_F(x_5) \, (21)$$

With

$$\mu_F(x_5) \triangleq \sum_{x_6,x_7} f_6(x_5,x_6,x_7)f_7(x_7) \, (22)$$

except that summation is everywhere replaced by maximization. The methodology of determining the messages is quite similar for the max-product algorithm and the sum-product algorithm. The method of computation of messages using max product algorithm is as follows. The message out of some node/factor $f_l$ along some edge $X_k$ is formed as the product of $f_l$ and all incoming messages along all edges except $X_k$, maximized over all involved variables except $X_k$. Each message is computed in both the directions which thereby consists of two different messages for all the edges of the factor graph. Thus, the marginal using max product algorithm $\hat{f}_k(x_k)$ is obtained concurrently is determined as the product of these two messages. It is to be noted that the message passing phenomena and the computation of the marginal is quite similar in sum product and max product algorithm. Indeed, the sum-product algorithm can be formulated to operate with abstract addition and multiplication operators "$\otimes$" and"$\otimes$",respectively, and setting "$\otimes$" then yields the max-product algorithm. Translating the max-product algorithm into the logarithmic domain yields the max-sum (or min-sum) algorithm. All the information bits are analysed for the calculation of the a posteriori marginal probability *p(b | y)* of every branch *b* of the factor graph. This technique of sum-product algorithm [21][25] is quite similar to the BCJR rule of message computation.If the max-product rule is used, the forward recursion is essentially identical with the Viterbi algorithm, except that no paths are stored, but all messages (branch metrics) must be stored; the backward recursion is formally identical with the forward recursion; and we can obtain the quantity *p(b |y )* ≜max*ω* ε µ: *b* fixed *p(ω | y )* for every branch *b* of the trellis (and hence for every information bit). The max-product algorithm may thus be viewed as a soft-output Viterbi algorithm, and the Viterbi-algorithm [2][28] itself may be viewed as an efficient hard-decision only version of the max-product algorithm.In the beginning the edges are initialized with unity i.e. *µ(.)* = 1. It is commonly known as neutral factor or neutral message. Iterative decoding algorithm is required for the computation of messages in factor graphs with cycles. All messages are then repeatedly updated, according to some schedule. The computation stops when the available time is over or when some other stopping condition is satisfied (e.g., when a valid codeword was found). This technique is used to determine the marginal in case of a factor graph without cycles for each node while in case of the graph with cycles the summary requires certain approximations. If rule (13) [or (15)] is implemented literally, the values of the messages/functions *µ(.)* typically tend quickly to zero (or sometimes to infinity).

In practice, therefore, the messages often need to be scaled or normalized instead of the message *µ(.)*, a modified message

$$\mu(.) \triangleq \gamma\mu(.) \qquad (23)$$

is computed, where the scale factor *γ* may be chosen freely for everymessage. The final result (21) will then be known only up to a scale factor, which is usually no problem. It is quite popular to write these messages in terms of the single parameters

$$L_X \triangleq \log\frac{\mu X\,(0)}{\mu X\,(1)} \qquad (24)$$

Or$\Delta \triangleq (\mu(0)-\mu(1))/(\mu(0)+\mu(1))$

For the decoding of LDPC codes [29][30] the typicalupdate schedule alternates betweenupdating the messages out of equality constraintnodes and updating the messages outof parity-check nodes.

## IV. REED MULLER CODES

### A. ENCODING

A generalised Boolean function is defined by mapping f:{0,1}$^m$→Z$_q$ of {0,1}- valued variables $x_0, x_1, \ldots, x_{m-1}$. A straightforward counting argument shows that every such function can be written in algebraic normal form as a sum of monomials of the form $x_{j0}x_{j1}\ldots x_{j_{r-1}}$ where $j_0, j_1\ldots, j_{r-1}$ are distinct. With each generalised Boolean function f, a length 2$^m$Z$_q$-valued vector $\lfloor f_0 f_1 \cdots f_{2^m-1}\rfloor$ is defined in which

$$f_i = f(i_0, i_1, \ldots, i_{m-1}) \qquad (25)$$

Where $\left[i_0 i_1 \ldots i_{m-1}\right]$ is the binary expansion of the integer $i=\sum_{j=0}^{m-1} i_j 2^j$. Henceforth, a generalized Boolean function is defined and their corresponding vectors using f to refer to both. The generator matrix to the Reed Muller codes can be obtained using the Hadamard transform. The Hadamard[6][11] matrix is defined for different dimensions, the basic being 2X2 matrix.

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \qquad (26)$$

The Hadamard transform for higher dimensions is obtained by repetition in a pattern as shown below:

$$H_4 = \begin{bmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{bmatrix} \qquad (27)$$

And

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \qquad (28)$$

Also

$$H_8 = \begin{bmatrix} H_4 & H_4 \\ H_4 & -H_4 \end{bmatrix} \qquad (29)$$

The Hadamard matrix of 8X8 dimensions is then obtained as follows:

$$H_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \quad (30)$$

The Hadamard matrix is used to obtain the generator matrix of the Reed Muller Codes. There happen to be eight different rows in the Hadamard matrix out of which the ones with more number of ones are picked.

Thus the generator matrix so obtained consists of eight columns and four rows as the code word length is eight and the data word length is four. Following is the generator matrix for the Reed Muller (1, 3) code:

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \end{bmatrix} \quad (31)$$

It can be written in unipolar form as follows:

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (32)$$

The code word so obtained is a matrix multiplication of the data word and the generator matrix.

$$C = aG \quad (33)$$

The generator matrix can be written in the standard form as follows:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \quad (34)$$

The Reed-Muller codes were first introducedby Muller in 1954. These codes were presented with good distance parameters but efficient method of decoding this group of codes was not announced.

The naïve method of decoding the RM (m; r) code is to enumerate all the code words, compute their distance to the received word and to output the one with the minimum distance. The running time of the naïve decoding algorithm is therefore quasi-polynomial (but not polynomial) in the block length n. however, shortly after introduction of Reed-Muller codes[12][25] the first efficient algorithm for decoding Reed-Muller codes was presented by Muller. Reed's algorithm also corrects up to half the minimum distance (i.e., up to $2^{m-r-1}-1$errors) and further runs in time polynomial in the block length n. This technique of the decoding algorithm involves a majority logic scheme at a very high level.

Let us now give a self-contained argument proving that the distance of the RM (m, r) code is $2^{m-r}$. The distance of binary Reed-Muller codes is at most $2^{m-r}$. Since Reed-Muller codes [30] are linear codes, we can do so by exhibiting a non-zero code word of RM (m; r) with weight $2^{m-r}$. Consider the polynomial

$$f(X_1,\dots,X_m) = X_1X_2\dots X_r$$

The polynomial $f \in F_2[X_1,\dots,X_m]$is a non-zero polynomial of degree r, and clearly $f(\alpha_1,\dots,\alpha_m) \neq 0$ only when $\alpha_1 = \alpha_2 = \dots = \alpha_r = 1$. There are $2^{m-r}$ choices of $\alpha \in F_2^m$ that satisfy this condition, so $wt\left(\langle f(\alpha)\rangle_{\alpha\in F_2^m}\right) = 2^{m-r}$ Let us now show that the distance of binary Reed-Muller codes is at least $2^{m-r}$ by showing that the weight of any non-zero code words in RM (m, r) is at least $2^{m-r}$. Consider any non-zero polynomial f(X1, . . .,Xm) of total degree at most r. We can write f as $f(X_1. . . X_m) = X_1X_2 . . . X_s + g(X_1; : : : ;X_m)$ where $X_1X_2 . . . X_s$is a maximum degree term in f and $s \leq r$. Consider any assignment of values to the variables $X_{s+1}, . . . ,X_m$. The polynomial obtained as a result on $X_1, . . . , X_s$ is a non-zero polynomial, as the cancellation of the term $X_1X_2 . . . X_s$ is not possible. Consequently for each of the $2^{m-s}$likelyallocationof values to the variables $X_{s+1}, . . .,X_m$, the polynomial so obtained is a non-zero polynomial. The non-zero polynomial always exhibits at the minimum of one value applicable to the variables such that the non-zero polynomial cannot be estimated to zero. Therefore, for each assignment $X_{s+1}. . . X_m$ to the variables $X_{s+1}, . . . ,X_m$, there exists at least one assignment of values $\alpha_1,\dots,\alpha_S$ to $X_1, . . ., X_s$ such that $f(\alpha_1,\dots,\alpha_m) \neq 0$ .This implies that with $\left(\left\langle f(\alpha)\right\rangle_{\alpha\in F_2^m}\right) = 2^{m-s} \geq 2^{m-r}$. In summary, when the maximum degree r is constant, binary Reed-Muller codes [13][14] have good distance, but a poor rate ($\approx m^r 2^{-m} \rightarrow 0$ for large m). The code rate r is inversely proportional to the distance of the code. If the code rate is increased to enhance efficiency the distance of the code reduces which in turn compromises the reliability of the coding scheme. Therefore, it can be said that no code can be achieved with constant rate and constant distance.

## B. DECODING

A maximum likelihood, soft-decision (MLSD) algorithm can be obtained by combining the method of super code decoding with the Fast Hadamard Transform (FHT) MLSD decoder [15][29] for RM(1,m). super code decoding is a general method applicable to codes formed from a union of cosets of a base code. Each coset representative is subtracted from the received word in turn, and the best result (in some metric sense) obtained using a decoder for the base code over all these modified words is selected. The corresponding coset representative and this best result determine the final decoded word. The FHT algorithm gives a computationally efficient method for computing the correlations between a received word and all $2^{m+1}$ words of the code RM(1,m).

For a length $2^m$ code formed from l cosets, the complexity of a soft-decision 'supercode+FHT' approach is approximately ln($2^m$)real operations. So this maximum likelihood supercode approach is computationally feasible only when it is relatively small. A convenient method for handling large number of binary cosets is to regard a received word as a code word of the full second-order code (with the addition of noise) and then use the well-known Reed Muller decoding algorithm [32][33]. This approach is guaranteed only to correct errors whose weights are less than half the maximum distance of the second-order Reed Muller code RM(2,m) i.e. whose weights are less than $2^m$. It is not maximum likelihood in general and gives no information when the decoded word happens to lie in a coset of RM(1,m) inside RM(2,m) that is not in the original union of cosets. Moreover, the Reed Algorithm as originally described is a hard decision algorithm. In other words, it operates on an input vector of binary valued components and does not use additional soft information that may be available from the demodulator.

A more efficient approach to decoding for $2^h$-ary codes has been the case of most practical interest. Therefore, coding domain decoders for $RM_2^h(1,m)$ requiring the computation of only h length $2^m$ integer or real FHTs (for hard or soft decision decoding) were given. Consequently, the complexity of these decoders is on the order of hm$2^m$ operations. The algorithms are not maximum likelihood, but a set of error patterns that can be corrected. In particular the algorithms are minimum distance decoders for both hamming and lee matrices (i.e. they can correct all errors of Hamming and Lee weightless [34][37] than half the appropriate minimum distance of the first order code).

The decoding algorithm for quaternary codes (q=4) is a maximum likelihood, hard decision, coding-domain algorithm which makes neat use of the existence of distance-preserving Gray map between the length $2^m$ quaternary code $ZRM_4$(1,m) and the length $2^{m+1}$ binary code $RM_2^h$(1,m+1). Since, $RM_4$(1,4) can be represented as a union of $2^m$cosets of $ZRM_4$(1,m), this map sends any union of l cosets of $RM_4$(1,m) onto a union of $2^m$l cosets of RM(1,m+1) and allows the use of binary decoding technique (for example, binary FHTs) to be applied to a quaternary code. By carefully extending the Gray map to soft values, it is also possible to develop a soft-decision version of this algorithm. On using the standard Gray map

$$\phi: 0 \rightarrow (0,0), 1 \rightarrow (0,1), 2 \rightarrow (1,1), 3 \rightarrow (1,0)$$

then we can extend 0 to a soft Gray map on inputs $r \in [0, 4)$ by writing:

$$\phi(r) = \begin{cases} (0,r) & for 0 \leq r \leq 1 \\ (r-1,1) & for 1 \leq r \leq 2 \\ (1,3-r) & for 2 \leq r \leq 3 \\ (4-r,0) & for 3 \leq r \leq 4 \end{cases} \qquad (35)$$

We can apply this soft $\phi$ to the components of real-valued input vectors and then use real-input FHTs on the resulting length 2m+1 vectors. Unfortunately, this decoder requires the computation of 2m l FHT's and this makes the decoder too intensive in all but the simplest of instances.

## V. ITERATIVE DECODING TECHNIQUE

The factor graph representation of the block codes is a convenient graphical representation that is very useful in the implementation and understanding of the maximum likelihood decoding of the these codes using Viterbi algorithm or symbol by symbol maximum a posteriori decoding using the BCJR algorithm. Representation of codes by more general graphical models is a convenient method in studying the performance of some decoding algorithms. Graph representation [39] is not limited to decoding algorithms but has many applications to signal processing, circuit theory, control theory, networking and probability theory. In this chapter, the design of a general algorithm known as the sum product Algorithm is provided.

The sum product algorithm [14][36][38]was first introduced by Gallager (1963) as a decoding method for Low Density Parity Check (LDPC) codes. Later Tanner (1981) introduced graphical models to describe this class of code. Wiberg et al. (1995) and Wiberg (1996) showed that the Viterbi and BCJR algorithms as well as decoding algorithms for turbo and LDPC codes can be unified in a single algorithm on certain graphs. The idea of graph representation of codes was further developed and generalized by Forney (2001).

The factor graph approach to signal processing involves the following steps.

1. Choose a factor graph to represent the systemmodel.
2. Choose appropriate message passing and estimation rules. The message passing requires use and maintenance of computational rule tables.
3. Choose a message update schedule.

In order to draw the graph of the said code we first need to know the parity check matrix

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

where rows act as function nodes and columns as bit nodes. The rows of the parity check matrix can be written in the form of the following equations:

$$f_1 : c_0 + c_1 + c_2 + c_4$$
$$f_2 : c_0 + c_1 + c_3 + c_5$$
$$f_3 : c_0 + c_2 + c_3 + c_6$$
$$f_4 : c_1 + c_2 + c_3 + c_7$$

These check equations can be represented in the form of a factor graph. The symbol c in equations is analogous to the symbol y in the factor graph.

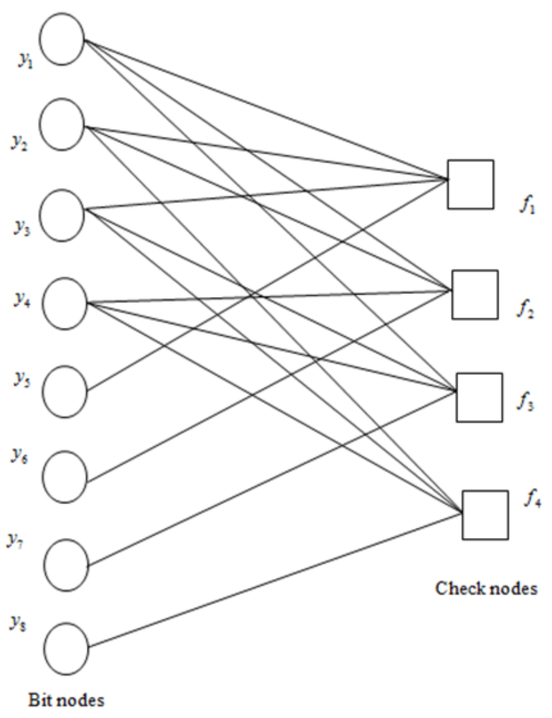

**Fig7:** Factor graph for the code

Instead of flipping bits, the sum-product algorithm (SPA) [31][34][35]propagates soft probabilities of the bits between bit nodes and check nodes through the Tanner graph, thereby refining the confidence that the parity checks provide about the bits. The exchange of the soft probabilities is termed as *messagepassing* or *belief propagation*. When no cycles exist in the Tanner graph, SPA computes the exact probabilities; when cycles are present, it computes only approximate solutions. However, even with cycles, the algorithm, given next, can still decode very effectively.

Initialization:

For bit node *j* with an edge to check node *i*:

Set:

; $$p_j^1 = \frac{1}{1+\exp(\frac{4r_j}{N_O})} \qquad p_j^0 = 1 - p_j^1 \quad (36)$$

And

$$P_{ij}^1 = p_j^1 \quad (37)$$
$$P_{ij}^0 = 1 - P_{ij}^1 \quad (38)$$

where$r_j$is the received bit corrupted by noise, and *s*2 = $N_0/2$ is the variance
of the AWGN channel.

1. From check nodes to bit nodes: For each check node i with an edge to bit node j, compute

$$\Delta P_{ij} = P_{ij}^0 - P_{ij}^1 \qquad (39)$$

Compute $\Delta Q_{ij}$ as a product of $\Delta P_{ij'}$ for all $j' \neq j$, that is

$$\Delta Q_{ij} = \prod_{j'} \Delta P_{ij'} \qquad (40)$$

Also compute,
$$Q_{ij}^1 = \frac{1}{2}\left(1 - \Delta Q_{ij}\right) \qquad (41)$$
$$Q_{ij}^0 = \frac{1}{2}\left(1 + \Delta Q_{ij}\right) \qquad (42)$$

(j'= 1,2,. . .,n)

2. from bit nodes to check nodes: For each bit node j with an edge to check node i:

Compute $P_{ij}^0$ as $p_j^0$ multiplied by the product of $Q_{i'j}^0$ and $P_{ij}^1$ as $p_j^1$ Multiplied by the product of $Q_{i'j}^1$ overall $i' \neq i$ $j' \neq j$ that is:

$$P_{ij}^0 = p_j^0 \prod_{i'} Q_{i'j}^0 \qquad (43)$$
$$P_{ij}^1 = p_j^1 \prod_{i'} Q_{i'j}^1 \qquad (44)$$

And scale $P_{ij}^0$ and $P_{ij}^1$ by a same factor so that
.
$$P_{ij}^1 + P_{ij}^0 = 1 \qquad (45)$$

Compute $P_{ij}^0$ as $p_j^0$ times the product of $Q_{i'j}^0$ and $P_{ij}^1$ multiplied by the product of $Q_{i'j}^1$ overall i, that is $p_j^1$

$$P_j^0 = p_j^0 \prod_i Q_{ij}^0 \qquad (46)$$
$$P_j^0 = p_j^0 \prod_{i'} Q_{ij}^0 \qquad \text{(i=1,2,. . .,m)}$$

And scale $P_j^0$ and $P_j^1$ by a same factor so that
.

$$P_j^0 + P_j^1 = 1 \qquad (47)$$

Decoding and soft outputs: For j = 1,2,. . .,n:

$$c_j = \begin{pmatrix} 0 & if \ \ln(P_j^1 / P_j^0) \geq 0 \\ 1 & otherwise \end{pmatrix} \qquad (48)$$

If $cH^T = 0$, stop and output hard decision c and/or soft likelihood.

$$\ln\left(P_j^1 / P_j^0\right) \qquad (49)$$

Otherwise goto step 1. In the latter case, if the iterations exceed a preset number, declare a decoding failure.

## VI. SIMULATION RESULTS

Bit error rate Vs Signal to noise Ratio curve has been plotted and the results have been compared with existing Maximum Likelihood Decoding technique for the Reed Muller codes.

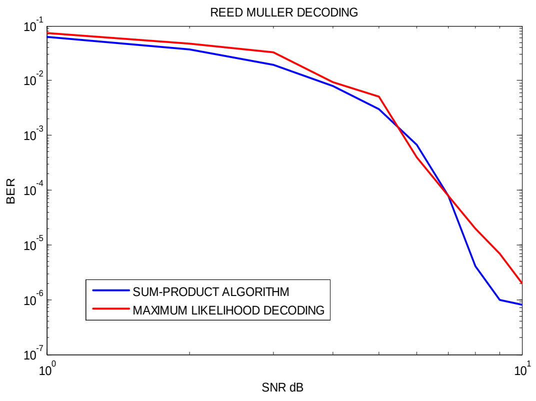

**Fig 8:** Bit error rate Vs SNR curve for reed muller code using sum-product algorithm and maximum likelihood decoding.

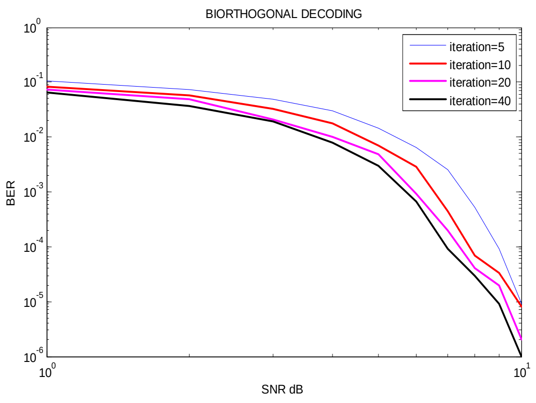

**Fig 9:** Bit error rate Vs SNR curve for biorthogonal code using sum-product algorithm for different iterations.

The plot shows that the performance improves with the growing number of iterations. The sum-product algorithm is an iterative method of decoding. The number of iterations can either be user defined or can be limited by using a desired threshold value. Biorthogonal codes are the first-order Reed-Muller codes RM (1,m) with parameters ($2^m$,m+1,$2^{m-1}$). The hamming distance of Biorthogonal codes is n/2. Similarly the Single Parity Check Codes is also a subcode of Reed Muller Codes with Hamming distance of 2. The parameters of the Single Parity Check codes are ($2^m$,$2^m$,1) where $2^m$ is the block length. For a linear block code, a maximum-likelihood (ML) decoder takes *n* received bits as input and returns the most likely k-bit message among the $2^k$ possible messages. Each message is computed for every edge in both the directions according to the summaryproductrule in a factor graph. A sharp distinction divides graphs with cycles from graphs without cycles. For a cycle-free factor graph the message computation technique is much more efficient. We begin with unity factor nodes or the leaves and successively compute messages as their required "input" messages become available. In this way, each message is computed exactly once. It is then obvious from the previous section that summaries/marginal can be computed as the product of messages simultaneously for all variables.

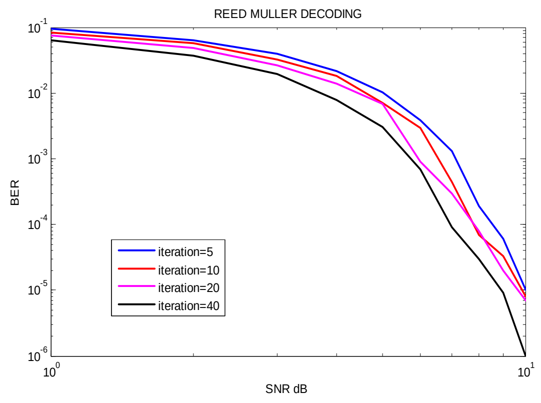

**Fig 10:** Bit error rate Vs SNR curve for Reed Muller code using sum-product algorithm for different iterations.

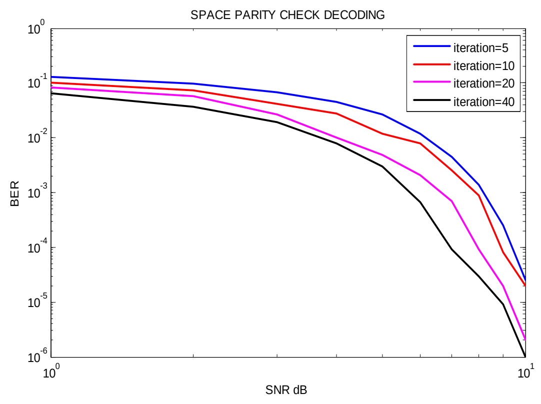

**Fig 11:** Bit error rate Vs SNR curve for Space Parity Check code using sum-product algorithm for different iterations.

## VII. CONCLUSION

In this work, popular signal processing problem such as decoding block codesis carried out using factor graph approach. The graphical theory allows ease of manipulation and interpretation. The coded signals are represented with the help of factor graphs and a network factor graph is created by mapping the vertices onto the network topology. The encoding of the Reed Muller codes is carried out by determining the generator matrix using the Hadamard matrix. Using the similar procedure the generator matrices of the extended Hamming Codes, bi-orthogonal codes and space parity check codes are obtained. These happen to be different categories of the Reed Muller codes. The decoding of the Reed Muller codes is carried out using the sum product algorithm as the message passing technique. The factor graph of the Reed Muller codes is obtained by the parity check matrix, the rows of which represent the check equations. The technique involves the message passing between the bit nodes and the check nodes in the form of a posteriori probabilities. The Bit Error Rate Vs Signal to Noise Ratio curves for the different coding schemes are obtained at the receiver side and significant improvement over SNR is observed. The message passing involves several iterations which are made to terminate after the equation check. As the number of iterations is increased the bit error rate reduces and the plots have been shown.

Factor graph approach can apply to many efficient algorithms.Factor graph is only a simplifying tool to solve the problems suited for hierarchical modeling ("boxes within boxes") and is compatible with standard block diagrams. It has simplest formulation of the summary-product message update rule and high computational efficiency. Instead of dealing with a function as a whole, subsequent decomposition is carried out which decreases the computational complexity and thus the design is simplified. The message update methods may be max-product or sum-product algorithm where the idea of the marginalization remains the same but the method changes from integration to comparison.

Factor graph approach is gradually finding its way into a number of communication and transmission applications.Location-awareness is a key feature of future-generation wireless networks, enabling a multitude of applications in the military (e.g., blue force tracking), public (e.g., search and-rescue), and commercial (e.g., navigation) sectors. Cooperation among nodes has the potential to dramatically improve localization performance

A vast fieldof coding and signal processing problems such as Kalman filtering, recursive least squares, joint decoding and parameter estimation, the iterative decoding of LDPC codes, turbo codes, and similar codes; joint decoding and equalization;hidden-Markov models and more implement factor graph based models. Factor graph based graphical models are found to be can characterize practical detection/estimation algorithms in a wide area of real time applications and other such complex real-world systems. Most good known signal processing techniques—including gradient methods, Kalman filtering, and particle methods—can be used as components of such algorithms. Forney factor graphs involve hierarchical modeling which made it easy to be used in this work. Moreover, compatibility with standard block diagrams provides additional advantage of the decoding and estimation technique.

## REFERENCES

[1]. Loeliger, Hans-Andrea, Justin Dauwels, Junli Hu, SaschaKorl, Li Ping, and Frank R. Kschischang. "The factor graph approach to model-based signal processing." *Proceedings of the IEEE* 95, no. 6 (2007): 1295-1322.

[2]. Loeliger, Hans-Andrea. "An introduction to factor graphs." *Signal Processing Magazine, IEEE* 21, no. 1 (2004): 28-41.

[3]. Moon, Todd K. "Error correction coding." *Mathematical Methods and Algorithms.Jhon Wiley and Son* (2005).

[4]. Arıkan, Erdal. "A performance comparison of polar codes and Reed-Muller codes." *IEEE Commun. Lett* 12, no. 6 (2008): 447-449.

[5]. Alon, Noga, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. "Testing reed-muller codes." *Information Theory, IEEE Transactions on*51, no. 11 (2005): 4032-4039.

[6]. Colavolpe, Giulio, and GianpietroGermi. "On the application of factor graphs and the sum-product algorithm to ISI channels." *Communications, IEEE Transactions on* 53, no. 5 (2005): 818-825.

[7]. Kurkoski, Brian M., Paul H. Siegel, and Jack Keil Wolf. "Joint message-passing decoding of LDPC codes and partial-response channels."*Information Theory, IEEE Transactions on* 48, no. 6 (2002): 1410-1422.

[8]. Drost, Robert J., and Andrew C. Singer. "Factor-graph algorithms for equalization." *Signal Processing, IEEE Transactions on* 55, no. 5 (2007): 2052-2065.

[9]. Baron, Dror, ShriramSarvotham, and Richard G. Baraniuk. "Bayesian compressive sensing via belief propagation." *Signal Processing, IEEE Transactions on* 58, no. 1 (2010): 269-280.

[10]. Worthen, Andrew P., and Wayne E. Stark. "Unified design of iterative receivers using factor graphs." *Information Theory, IEEE Transactions on*47, no. 2 (2001): 843-849.

[11]. Gross, Brandi Nicole. "Input of Factor Graphs into the Detection, Classification, and Localization Chain and Continuous Active SONAR in Undersea Vehicles."PhD diss., Virginia Tech, 2015.

[12]. Wymeersch, H., J. Lien, and M.Z. Win. “Cooperative Localization in Wireless Networks.” Proceedings of the IEEE 97.2 (2009): 427-450. ©2011 IEEE.

[13]. Jiang, Yuan. *A practical guide to error-control coding using Matlab*.Artech House, 2010.

[14]. Shuman, David I., Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains." *IEEE Signal Processing Magazine* 30, no. 3 (2013): 83-98.

[15]. Sandryhaila, Aliaksei, and José MF Moura. "Discrete signal processing on graphs."*IEEE transactions on signal processing* 61, no. 7 (2013): 1644-1656.

[16]. Loeliger, H. Andrea. "Some remarks on factor graphs." In *Proc. 3rd International Symposium on Turbo Codes and Related Topics*, no. 3, pp. 1-5. 2003.

[17]. Korl, Sascha. *A factor graph approach to signal modelling, system identification and filtering*. Diss., TechnischeWissenschaften, EidgenössischeTechnischeHochschule ETH Zürich, Nr. 16170, 2005, 2005.

[18]. Mao, Yongyi, Frank R. Kschischang, Baochun Li, and SubbarayanPasupathy. "A factor graph approach to link loss monitoring in wireless sensor networks." *IEEE Journal on Selected Areas in Communications* 23, no. 4 (2005): 820-829.

[19]. Kschischang, Frank R., Brendan J. Frey, and H-A. Loeliger."Factor graphs and the sum-product algorithm."*IEEE Transactions on information theory* 47, no. 2 (2001): 498-519.

[20]. Mooij, Joris M., and Hilbert J. Kappen. "Sufficient conditions for convergence of the sum–product algorithm." *IEEE Transactions on Information Theory* 53, no. 12 (2007): 4422-4437.

[21]. Papaharalabos, S., and P. T. Mathiopoulos. "Simplified sum-product algorithm for decoding LDPC codes with optimal performance."*Electronics letters* 45, no. 2 (2009): 116-117.

[22]. Wymeersch, Henk. *Iterative receiver design*.Vol. 234. Cambridge: Cambridge University Press, 2007.

[23]. Singla, Naveen, and Joseph A. O'Sullivan. "Joint equalization and decoding for nonlinear two-dimensional intersymbol interference channels."In *Proceedings.International Symposium on Information Theory, 2005. ISIT 2005.*, pp. 1353-1357. IEEE, 2005.

[24]. Jiang, Jing, and Krishna R. Narayanan. "Iterative Soft-Input Soft-Output Decoding of Reed&# 8211; Solomon Codes by Adapting the Parity-Check Matrix."*IEEE Transactions on Information Theory* 52, no. 8 (2006): 3746-3756.

[25]. Guo, Qinghua, and Li Ping. "LMMSE turbo equalization based on factor graphs."*IEEE Journal on Selected Areas in Communications* 26, no. 2 (2008): 311-319.

[26]. Hrycak, Tomasz, D. A. S. Saptarshi, Hans Georg Feichtinger, and Gerald Matz. "Method for channel equalization." U.S. Patent 8,755,429, issued June 17, 2014.

[27]. Bordin, Claudio J., and Marcelo GS Bruno. "Bayesian distributed blind equalization based on density-sum filters." In *Telecommunications Symposium (ITS), 2014 International*, pp. 1-5. IEEE, 2014.

[28]. Howard, S.D., Calderbank, A.R. and Searle, S.J., 2008, March. A fast reconstruction algorithm for deterministic compressive sensing using second order Reed-Muller codes.In *Information Sciences and Systems, 2008.CISS 2008. 42nd Annual Conference on* (pp. 11-15). IEEE.

[29]. Yang, T.Y. and Chen, H., 2015, May. Graph realization of reed-muller codes for data hiding. In *Next-Generation Electronics (ISNE), 2015 International Symposium on* (pp. 1-4).IEEE.

[30]. Peharz, R., 2015. *Foundations of sum-product networks for probabilistic modeling* (Doctoral dissertation, PhD thesis, Aalborg University).

[31]. Kudekar, S., Richardson, T. and Iyengar, A., 2014, June. The effect of saturation on belief propagation decoding of ldpc codes.In *Information Theory (ISIT), 2014 IEEE International Symposium on* (pp. 2604-2608).IEEE.

[32]. Uchoa, A.G., Healy, C.T. and de Lamare, R.C., 2016. Iterative detection and decoding algorithms for MIMO systems in block-fading channels using LDPC codes. *IEEE Transactions on Vehicular Technology*, *65*(4), pp.2735-2741.

[41]. Mondelli, M., Kudekar, S., Kumar, S., Pfister, H.D. and Urbanke, R., 2016. Reed-Muller Codes: Thresholds and Weight Distribution.

[33]. Uchoa, A.G., Healy, C.T. and de Lamare, R.C., 2016. Iterative detection and decoding algorithms for MIMO systems in block-fading channels using LDPC codes. *IEEE Transactions on Vehicular Technology*, *65*(4), pp.2735-2741.

[34]. Liu, X., Zhang, Y. and Cui, R., 2015. Variable-node-based dynamic scheduling strategy for belief-propagation decoding of LDPC codes. *IEEE Communications Letters*, *19*(2), pp.147-150.

[35]. Park, S. and Shin, J., 2014. Max-product belief propagation for linear programming: applications to combinatorial optimization. *arXiv preprint arXiv:1412.4972*.

[36]. Fayyaz, U.U. and Barry, J.R., 2014. Low-complexity soft-output decoding of polar codes. *IEEE Journal on Selected Areas in Communications*, *32*(5), pp.958-966.

[37]. Novak, C., Matz, G. and Hlawatsch, F., 2013. IDMA for the multiuser MIMO-OFDM uplink: A factor graph framework for joint data detection and channel estimation. *IEEE Transactions on Signal Processing*, *61*(16), pp.4051-4066.

[38]. Schlegel, C.B. and Perez, L.C., 2015. *Trellis and Turbo Coding: Iterative and Graph-based Error Control Coding*. John Wiley & Sons.

[39]. Niu, K., Chen, K., Lin, J. and Zhang, Q.T., 2014. Polar codes: Primary concepts and practical decoding algorithms. *IEEE Communications magazine*, *52*(7), pp.192-203.

[40]. Xu, J., Che, T. and Choi, G., 2015, December. XJ-BP: Express journey belief propagation decoding for polar codes. In *Global Communications Conference (GLOBECOM), 2015 IEEE* (pp. 1-6).IEEE.

Sarah Anjum "Factor Graph Approach for Decoding Reed Muller Codes." International Journal of Engineering Research and Applications (IJERA) , vol. 8, no. 1, 2018, pp. 70-82.