# Some Results and Challenges on Codes and Iterative Decoding with Non-equal Symbol Probabilities

Bowen Dai, Lei Wei

Department of Electrical and Computer Engineering,
University of Central Florida,
Orlando, FL 32816, email: lei@ee.ucf.edu.

***Abstract*—Modern telecommunication and error control coding are largely designed to handle equiprobable symbols. In this paper, we present challenges and several results in designing codes and iterative decoding when we extend to non-equiprobable symbols, which may lead to better understanding of bio-signal processing in future. We first demonstrate the limitation of XOR operation in dealing with non-equiprobable symbols. We then present two possible ways to mitigate the limitation: quasi-XOR operation and intermediate transformation layer. We also show how to construct codes for non-equiprobable symbols using quasi-XOR operation and prove the codes are very poor in terms of error correction capability. We further compute probabilistic messages for sum-product algorithm using XOR, AND, and OR operations with non-equiprobable symbols. We outline further challenges to design codes for non-equiprobable symbols.**

## I. Introduction

Shannon's separation principle [1] has played a key role in design of telecommunication systems, which recommends to design the source and channel coders in a system independently. This is because whatever performance is achievable with a jointly designed source/channel coding system, it is also achievable with a source coder, designed and optimized solely with regard to the source description, and a channel coder, designed and optimized solely with regard to the channel description. When we design the channel coding and modulation, we always assume that the data has been compressed by an ideal source encoder, so it produces an independent, identically distributed (i.i.d) sequence of equiprobable bits at the output. Both modulation, capacity, and error control coding have been well studied under these considerations [2] - [5].

Recently, these settings have been extended in various directions to accommodate the practical needs, not only in telecommunication system design [7]-[11], but also in the understanding of biological systems [13]-[16]. In this paper, we move away from the equiprobable case by one small step. We assume that information source symbols have a fixed, but non-equal prior probability. We wish to design an error control encoder which produces the output symbols with the same prior probability. Later on, we can generalize the results to more complicated cases. When we examine the key operation, the XOR operation, used in error control coding, we note that it does not fit well with the non-equiprobable symbol case. In other words, when we apply XOR on two equiprobable symbols (say $x, y$), the result ($z = x \oplus y$, where $\oplus$ denotes XOR) is an equiprobable symbol. However, when we apply XOR on two non-equiprobable symbols (say, $P(x = 0) = P(y = 0) = 0.3$, where $P(.)$ denotes probability), the output symbol does not have the same prior probability ($P(z = 0) = 0.58$). This implies that we can not apply XOR directly to symbol sequences with non-equiprobability. Furthermore, when studying biological systems using likelihood detection theory [13], source compressing called sparse coding [14], and iterative turbo-like joint estimation and decoding called factorized decoder [15], we may need to go beyond the conventional wisdom. The ultimate goal is to study how to design source and channel coding mechanisms for the systems with imperfections, which may lead to better understanding of signal processing mechanism used in bio-systems and a new type of intelligent machine called GPRAM [16] [17].

Our work aims to extend the possible linkage between advanced coding theory and brain functionalities, highlighted in [18]-[21]. To do so, we need to extend error control coding theory beyond current application.

In this paper we will present challenges and some results on how to construct codes and iterative decoding algorithms for the non-equiprobable case. The rest of the paper is organized as follows. In Section II, we illustrate the limitation of XOR operation in the conventional error control codes to deal with the non-equiprobable case. In Section III, we present two possible ways to mitigate the limitation: quasi-XOR operation and intermediate transformation layer. In Section IV, we show how to construct codes for non-equiprobable symbols using quasi-XOR operation and how to compute probabilistic messages for sum-product algorithm using XOR, AND, and OR operations. In Section V, we conclude and outline further challenges to design codes for non-equiprobable symbols.

## II. Limitation of XOR Operation

Let us consider a simple (2,3) parity check code, in which the two message bits are $(u_0, u_1)$ and three codeword bits are $(v_0, v_1, v_2) = (u_0, u_1, u_0 \oplus u_1)$, where $\oplus$ denotes XOR operation. If the message bits are equiprobable, i.e., $u_i \in \{0, 1\}$ with $P(u_i = 0) = P(u_i = 1) = 0.5$, then symbol probabilities for the codeword bits are also equiprobable. After encoding, we can transmit codeword bits $v_i \in \{0, 1\}$) using equal-spaced constellation (i.e., $x_i \in \{\pm 1\}$).

The XOR operation has several nice properties: (a) reversible (i.e., if $u_0 \oplus u_1 = u_2$, then $u_2 \oplus u_1 = u_0$ and $u_0 \oplus u_2 = u_1$; (b) symmetric (i.e., $u_0 \oplus u_1 = u_1 \oplus u_0$); (c) scalable (i.e., $u_0 \oplus u_1 \oplus u_2 = (u_0 \oplus u_1) \oplus u_2$).

However, if the message bits are non-equiprobable (say $P(u_i = 0) = 0.3$), then we have non-equiprobable codeword bits, i.e., $P(v_0 = 0) = P(v_1 = 0) = 0.3$ and $P(v_2 = 0) = 0.58$. Clearly, message passing from $v_2$ and $v_1$ to $v_0$ (i.e., $v_1 \oplus v_2 = v_0$), will result in a mismatching in prior probability, since $P(v_1 \oplus v_2 = 0) = 0.468 \neq P(v_0 = 0) = 0.3$. Furthermore, it is difficult to match up optimal constellations between message and codeword bits.

The optimal constellations for message bits satisfy $x_i \in \{w_0, w_1\}$ and $0.3w_0 + 0.7w_1 = 0$, where $w_j$ denotes amplitude of rectangular pulse waveform when $u = j$ is transmitted, [12]. If we select $x_i \in \{\pm 1\}$, then it costs 0.72 dB reduction in $E_b/N_0$ capacity. No coding strategy can recover this loss. If we express constellation in two consecutive bits, then we have four symbols, $m_0 = (u_0 = 0, u_1 = 0) = (0, 0)$, $m_1 = (0, 1)$, $m_2 = (1, 0)$, $m_3 = (1, 1)$, with probabilities (0.09, 0.21, 0.21, and 0.49), and constellation $(\{w_0, w_0\}, \{w_0, w_1\}, \{w_1, w_0\}, \{w_1, w_1\})$, respectively. Now, how to assign the constellations to codeword bits? Conventionally, we use the same constellation $x_i \in \{w_0, w_1\}$ for codeword bits but each has a duration of $2/3$ of the message bit duration. However, this is not optimal since $P(v_2 = 0) = 0.58$, not 0.3.

Therefore, we need to replace XOR operation so that we have prior probability of each codeword bit matched to the prior probability of each message bit. Here we introduce two novel solutions. The first one is to use quasi-XOR operation and the second one is to use intermediate transformation to obtain symbols with prior probabilities suitable for XOR operation.

## III. quasi-XOR Operation and Intermediate Transformation Layer

In this section, we illustrate two methods to solve the problem. The first method is to find alternative operation to replace XOR and the second method is to transfer non-equiprobable symbols to equiprobable symbols.

### *A. quasi-XOR Operation*

We introduce quasi-XOR operation which produces output with arbitrary prior probability ($0 \leq p \leq 1$) identical to that of input. In Fig. 1, we show Karnaugh maps for XOR operation (a), and quasi-XOR (QXOR) operations with three inputs (c) and four inputs (d), respectively. We obtain QXOR operation as follows. First, we extend XOR operation by introducing one additional input (i.e. $I_3$) and then label 1 for all slots with $I_3 = 1$ (see (b)). We then move some of "1" to other places with the same Hamming weight. For example, "1" at the bottom left (in (b)) is moved to "1" at the top right (in Fig. 1(c)). By doing so, we have six cases: $O_1 = \bar{I}_2 I_1 + I_2 I_3$ (shown in Fig. 1(c)), $O_2 = \bar{I}_1 I_3 + I_2 I_1$, $O_3 = \bar{I}_3 I_2 + I_1 I_3$, $O_4 = \bar{I}_2 I_3 + I_1 I_2$, $O_5 = \bar{I}_1 I_2 + I_1 I_3$, and $O_6 = \bar{I}_3 I_1 + I_2 I_3$. We will select the first three, since they form three feedback paths from $O_1, O_2$, and $O_3$ to $I_1, I_2$, and $I_3$ as follows: $I_1 = \bar{O}_2 O_1 + O_2 O_3$, $I_2 = \bar{O}_1 O_3 + O_1 O_2$, $I_3 = \bar{O}_3 O_2 + O_3 O_1$, i.e., by swapping $I_1, I_2, I_3$ with $O_1, O_2, O_3$, respectively. We define this operation as $QXOR_3$.

For four inputs, the construction is as follows. $O_1 = I_1 I_2 I_3 + I_2 \bar{I}_1 \bar{I}_4 + I_2 I_4 \bar{I}_3 + I_3 I_4 \bar{I}_2$ (see Fig. 1 (d)); $O_2 = I_1 I_2 I_4 + I_1 \bar{I}_2 \bar{I}_3 + I_3 I_4 \bar{I}_1 + I_1 I_3 \bar{I}_4$; $O_3 = I_1 I_3 I_4 + I_3 \bar{I}_2 \bar{I}_4 + I_1 I_2 \bar{I}_3 + I_2 I_3 \bar{I}_1$; $O_4 = I_2 I_3 I_4 + I_4 \bar{I}_1 \bar{I}_3 + I_1 I_2 \bar{I}_4 + I_1 I_4 \bar{I}_2$. The reversing operations can be obtained by swapping $I_1, I_2, I_3$, and $I_4$ with $O_1, O_2, O_3$, and $O_4$ respectively. We define this operation as $QXOR_4$.

We obtain many different sets of similar operations. Here, we list one of them as $O_1 = I_4 \bar{I}_1 \bar{I}_3 + I_2 I_3 I_4 + I_1 I_2 \bar{I}_4 + I_1 I_4 \bar{I}_2$; $O_2 = I_2 \bar{I}_1 \bar{I}_3 + I_1 I_2 I_4 + I_3 I_4 \bar{I}_2 + I_2 I_3 \bar{I}_4$; $O_3 = I_3 \bar{I}_1 \bar{I}_4 + I_2 I_3 I_4 + I_1 I_3 \bar{I}_2 + I_1 I_2 \bar{I}_3$; $O_4 = I_1 \bar{I}_2 \bar{I}_3 + I_1 I_2 I_4 + I_1 I_3 \bar{I}_4 + I_3 I_4 \bar{I}_1$. and reversing operations as $I_1 = O_1 O_4 \bar{O}_3 + O_2 O_3 O_4 + O_1 O_3 \bar{O}_4 + O_4 \bar{O}_1 \bar{O}_2$; $I_2 = O_2 \bar{O}_3 \bar{O}_4 + O_1 O_2 O_4 + O_2 O_3 \bar{O}_1 + O_1 O_3 \bar{O}_2$; $I_3 = O_3 \bar{O}_1 \bar{O}_2 + O_1 O_3 O_4 + O_2 O_3 \bar{O}_4 + O_2 O_4 \bar{O}_3$; $I_4 = O_1 \bar{O}_3 \bar{O}_4 + O_1 O_2 O_3 + O_2 O_4 \bar{O}_1 + O_1 O_4 \bar{O}_2$.

By combining three and four input operations, we can obtain operations with any number (no less than 3) of inputs.

### *B. Intermediate Transformation Layer*

First, let us look at how symbol prior probability changes after XOR operation. Let us define $M$ input XOR operation (i.e., $I_1 \oplus I_2 \oplus I_3, ..., \oplus I_M$ as $M$-XOR. If the prior probability of symbol $I_i$ is $p_i$, then we can compute the prior probability of output symbol (denoted as $p_o$) recursively as follows. Setting, $y(2) = p_1 p_2 + (1 - p_1)(1 - p_2)$, For i=3, ..., M, compute

$$y(i) = p_i y(i-1) + (1 - p_i)(1 - y(i-1)) \qquad (1)$$

and finally, we have $p_o = y(M)$. In Fig. 5, we plot $p_o$ as a function of $p_i$ for $M$-XOR. It shows that (a) in order to generate symbols with all possible prior probabilities between 0 and 1, we need to stay at the left hand side, i.e., small $p_i$. That is, the probability of satisfying parity check constraint must be small. If we can use NOT gates to invert symbols, then we can stay at the right hand side as well. But, in either case, we must stay away from the central point ($p_i = 0.5$). (b) We have the value of $p_o$ close to 0.5 over a large range of $p_i$ values around of 0.5. The larger the value of $M$ is, the broader this range becomes. (c) From (1), we have the following results. If $p_i < 0.5$, then $y(i) = p_i y(i-1) + (1-p_i)(1-y(i-1)) = p_i + (1 - 2p_i)(1 - y(i-1)) \geq p_i$; and if $p_i > 0.5$, then $y(i) = p_i - (2p_i - 1)(1 - y(i-1)) \leq p_i$. (d) Furthermore, we have $p_o = 0.5$ if $p_i = 0.5$ for any $i \in (1, 2, ..., M)$.

Secondly, in order to impose code constraint, the prior probabilities must match each other, for example, if $I_1 \oplus I_2 \oplus I_3 = I_4$, then $p_o$ of $I_1 \oplus I_2 \oplus I_3$ must match up with $p_i$ of $I_4$. We do not use $I_1 \oplus I_2 \oplus I_3 \oplus I_4 = 0$ since reverse operations are often not true due to mismatch between prior probabilities for non-equiprobable cases. Therefore, code constraint becomes directional, rather than bi-directional.

There are three ways to use intermediate transformation layer (ITL) to deal with non-equiprobable symbols: (a) to apply ITL first, then follow by $M$-XOR. For example, we can use logic gates to convert symbols with $p_i = 0.3$ to symbols with $p = 0.1316$, then apply 3-XOR operation to impose code constraint directly, which produce output symbols with $p_o = 0.3$. We call this the front ITL (F-ITL) (see Fig. A). (b) To apply $M$-XOR operations to non-equiprobable symbols with $p_i$s, then use the transformation to match up $p_o$ of the output symbol with $p_i$ of the constrained symbol. We call it the End ITL (E-ITL). (c) To convert non-equiprobable symbols to equiprobable symbols, then to operate using conventional codes, and finally, to convert equiprobable symbols back to non-equiprobable symbols. We call this Half ITL (H-ITL).

Let us look F-ITL using an example $p = 0.3$. If we combine two inputs together as a group, then we have three basic probabilities ($p^2 = 0.09$ for $I_1 = 0$ and $I_2 = 0$, $p(1-p) = 0.21$, and $(1-p)^2 = 0.49$). For each two inputs (say $I_{i,1}$ and $I_{i,2}$, where subscribe $i$ denote the $i^{th}$ group of two inputs), F-ITL will have three outputs $P(y_{i,1} = 0) = P(I_{i,1} = 0 \bigcap I_{i,2} = 0) = 0.09$, $P(y_{i,2} = 0) = P(I_{i,1} = 0 \bigcap I_{i,2} = 1) = 0.21$, $P(y_{i,3} = 0) = P(I_{i,1} = 1 \bigcap I_{i,2} = 0) = 0.21$), i.e., $y_{i,1} = I_{i,1} + I_{i,2}$, $y_{i,2} = I_{i,1} + \bar{I}_{i,2}$, and $y_{i,3} = \bar{I}_{i,1} + I_{i,2}$. 3-XOR with $y_{i,1}$ (i.e., $P(y_{i,1} = 0) = 0.09$ for the $i^{th}$ group), $y_{i+1,1}$ (i.e., $P(y_{i+1,1} = 0) = 0.09$ for the $i+1^{th}$ group) and $y_{i+2,2}$ (i.e., $P(y_{i+2,2} = 0) = 0.21$ for the $i+2^{th}$ group) will produce an output symbol with $p_o = 0.305$, which approximates $p = 0.3$ (see Fig. A).

If we combine three inputs as a group, then we have four basic probabilities ($p^3 = 0.027$, $p^2(1-p) = 0.063$, $p(1-p)^2 = 0.147$, and $(1-p)^3 = 0.343$). We have seven outputs, $y_{i,1} = I_{i,1} + I_{i,2} + I_{i,3}$, $y_{i,2} = \bar{I}_{i,1} + I_{i,2} + I_{i,3}$, $y_{i,3} = I_{i,1} + \bar{I}_{i,2} + I_{i,3}$, $y_{i,4} = I_{i,1} + I_{i,2} + \bar{I}_{i,3}$, $y_{i,5} = \bar{I}_{i,1} + \bar{I}_{i,2} + I_{i,3}$, $y_{i,6} = \bar{I}_{i,1} + I_{i,2} + \bar{I}_{i,3}$, $y_{i,7} = I_{i,1} + \bar{I}_{i,2} + \bar{I}_{i,3}$. Let $z_{i,1} = y_{i,1} y_{i+1,1} y_{i+2,1} y_{i+3,1} y_{i+4,1}$, $z_{i,2} = y_{i,2} y_{i+1,2}$, and $z_{i,3} = y_{i,5}$. 3-XOR with $z_{i,1}$, $z_{i+5,1}$ and $z_{i+10,2}$, (or $z_{i,2}$, $z_{i+2,2}$, and $z_{i+4,3}$), can produce an output symbol with $p_o = 0.291$ (or $p_o = 0.299$, respectively). If we have $w_{i,1} = y_{i,1} y_{i+1,1} y_{i+2,1}$, $w_{i,2} = y_{i,2}$, and $w_{i,3} = y_{i,1} y_{i+1,5}$, then 4-XOR with $w_{i,1}$, $w_{i+3,1}$, $w_{i+6,2}$, and $w_{i+7,3}$ will produce an output symbol with $p_o = 0.705$. Inverting this symbol will get a symbol with a prior probability of 0.2954.

It becomes clear that F-ITL will generate symbols with low probabilities first then XOR these low probable symbols will produce symbols approximately matched to the required prior probability.

We consider E-ITL, again using the example with $p = 0.3$. According to Fig. 5, if we apply $M$-XOR with $M \geq 4$, then $p_o \approx 0.5$. Let $O$s denote output symbols. We can construct $y_i(m) = O_i + O_{i+1} + ... + O_{i+m}$ and the prior probability of $y_i(m)$ is $p_y(m) = 2^{-(m+1)}$. Searching for combination of $p_y(m)$ approximately equal to $p = 0.3$, we have $p_y(1) + p_y(4) + p_y(6) = 0.296$. Now, we can build $z_i = y_i(2) y_{i+3}(5) y_{i+9}(6)$ which has prior probability of $p = 0.296$. This is far more simpler than F-ITL.

Finally, we consider H-ITL. We can use $M$-XOR to construct the front-layer which transform prior probability from 0.3 to 0.5. Then construct the end-layer similar to E-ITL. Between the front-layer and the end-layer, we can use conventional error control codes.

## IV. Code Construction and Probabilistic Messages for Non-equiprobable Symbols

In this section, we study how to construct codes using QXOR. We call them neither source codes nor error control codes, since they may perform functionalities in applications or help us understanding bio-signal process beyond conventional source and channel coding in modern telecommunication systems (see [16]).

### A. Tree Codes with QXOR

In this subsection, we construct two simple codes with QXOR operations (Fig. 6(2) and (4)) and compare them with codes based on XOR operations (Fig. 6 (1) and (3)). Currently, we are working on how to construct codes using Intermediate Transformation Layer.

Codes (1) and (3) are constructed as follows: Code (1) with $O_1 = I_1 \oplus I_2 = \bar{I}_2 I_1 + I_2 \bar{I}_1$, $O_2 = I_1 \oplus I_3$, $O_3 = I_2 \oplus I_3$ (defined as $XOR_3$, which is different from $M$-XOR); Code (3) with $O_1 = I_1 \oplus I_2 \oplus I_3$, $O_2 = I_1 \oplus I_3 \oplus I_4$, $O_3 = I_1 \oplus I_2 \oplus I_4$, and $O_4 = I_2 \oplus I_3 \oplus I_4$ (defined as $XOR_4$). Both codes are linear with Hamming distances of 3 and 4 for Code (1), and 4 and 7 for Code (3), respectively. Furthermore, many new constraints exist in both codes, for example, $O_1 \oplus O_2 \oplus O_3 = 0$ for Code (1) and $I_1 = O_1 \oplus O_2 \oplus O_3$ for Code (3). These constraints will form mesh networks when we merge more operations together.

Codes (2) and (4) in Fig. 6 are constructed based on $QXOR_3$ and $QXOR_4$ respectively. Both are non-linear codes with Hamming distance (2,4,6) and (2,4,6,8) respectively. Both are similar to repetition codes in term of distances. Input bits in the repetition codes are independent each other, while input bits in Codes (2) and (4) are dependent each other through constraints of $O$ bits. Both are reversible for non-equiprobable symbols.

*Property 1:* The minimum Hamming distance of codes based on $QXOR_M$ is no greater than 2 and multiplicity is $M$.

*Proof:* Codes based on $QXOR_M$ contain two parts ($I$s and $O$s, see Fig. 6) and Hamming weight of each codeword is equal to the sum of Hamming weights in $I$s and $O$s. According to Karnaugh maps (see Fig. 1), each of $M-$ $O$ bits with $O = 1$ in codes based on $QXOR_M$ must contain one and only one codeword with Hamming weight of 1 in $I$s in order to maintain the same prior probability. For example, see Fig. 1 (d), in 8 places which $O_1 = 1$, only one place has Hamming weight of 1, i.e., $I_1 I_3 I_3 I_4 = 0100$. There are a total of $M$ codewords with Hamming weight of 1 in $I$s. If one of such codewords has a Hamming weight of 2 or more in $O$s, then there must be at least one codeword with Hamming weight of 0 in $O$s in these $M$ codewords. Consequently, the minimum Hamming weight is 1. If each of $M$ codewords has a Hamming weight of 1 in $O$s, then we have codes with minimum Hamming distance of 2 and multiplicity is $M$. ■

The above proposition simply rules out the possible to construct a good error control code using this method, since it produces neither large minimum distance nor a favorable distribution of multiplicities.

## *B. Probabilistic messages for non-equiprobable symbols with XOR and other logic operations*

It has been well known how to compute probabilistic messages for sum-product algorithm with XOR operation and equiprobable symbols (see [5] for summary). Here we derive probabilistic operations for XOR, AND, and OR gates for non-equiprobable symbols.

Let us focus basic two-inputs and one-output operations. For XOR, we have $(v_0, v_1, v_2)$ and $(v_2 = v_0 \oplus v_1)$. Then, symbol $v_i$ with prior probabilities $(p_i = Pr(v_i = 0))$ is mapped to constellation $x_i$ for transmission, where $i = 0, 1, 2$. At the receiver, we obtain a noise corrupted signal $r_i = x_i + n_i$, where $n_i$ is a white Gaussian variable with zero mean and variance $\sigma^2$. Let us define $L_i = \ln\left(\frac{P(v_i=0|r_i)}{P(v_i=1|r_i)}\right)$ for $i = 0$ and 1, $R_2 = \ln\left(\frac{P(r_2|v_2=0)}{P(r_2|v_2=1)}\right)$ $U_i = \ln\left(\frac{P(v_i=0|r_{j_1},r_{j_2})}{P(v_i=1|r_{j_1},r_{j_2})}\right)$, where $i = 0, 1, 2$, $j_1 = i + 1 \mod 3$, and $j_2 = i + 2 \mod 3$, $\mod$ denotes modulo operation. To simplify expression, we denote $L_2 = R_2$. Now, we obtain

$$\tanh\left(\frac{U_i}{2}\right) = \tanh\left(\frac{L_{j_1}}{2}\right)\tanh\left(\frac{L_{j_2}}{2}\right) \tag{2}$$

or

$$e^{U_i} = \frac{1 + e^{L_{j_1}}e^{L_{j_2}}}{e^{L_{j_1}} + e^{L_{j_2}}} \tag{3}$$

$$\ln\left(\frac{P(v_i = 0|r_1, r_2, r_3)}{P(v_i = 1|r_1, r_2, r_3)}\right) = U_i + L_i \tag{4}$$

where $i = 0, 1, 2$.

For AND, we replace (3) by

$$e^{U_i} = \frac{e^{L_{j_1}} + e^{L_{j_2}}}{1 + e^{L_{j_1}}e^{L_{j_2}}} \tag{5}$$

for i=0, 1, and

$$e^{U_2} = (1 + e^{L_1})(1 + e^{L_2}) - 1 \tag{6}$$

For OR, we replace (3) by

$$e^{U_i} = \frac{1 + e^{L_2}e^{L_{j_3}}}{1 + e^{L_{j_3}}} \tag{7}$$

for i=0, 1, and $j_3 = i + 1 \mod 2$, and

$$e^{U_2} = \frac{e^{L_0}e^{L_1}}{1 + e^{L_0} + e^{L_1}} \tag{8}$$

## V. Conclude and Challenges

In this paper, we presented challenges and several results in designing codes and iterative decoding for non-equiprobable symbols. We first demonstrated the limitation of XOR operation in dealing with non-equiprobable symbols. We then presented two possible ways to mitigate the limitation: quasi-XOR operation and intermediate transformation layer. We showed how to construct codes for non-equiprobable symbols using quasi-XOR operation and prove the codes are very poor in term of error correction capability. We further computed probabilistic messages for sum-product algorithm using XOR, AND, and OR operations with non-equiprobable symbols.

We now outline further challenges to design codes for non-equiprobable symbols.

1 **beyond XOR:** To understand bio-systems, we may need to extend the basic operation beyond XOR, for example, including AND and OR gates, or the combination of these gates. These will increase the complication of design.

3 **Code construction using intermediate transformation layer:** Up to now, we could not find a way to match the targeted prior probabilities precisely with those in intermediate transformation layer. We can only approximately match them up.

4 **Encoder:** It has been a well-known problem to construct encoder from parity check matrix of general LDPC codes. For codes with XOR, AND, and OR gates with non-equiprobable, the problem gets much worse.

5 **Application of these codes:** It is an open question how to apply these codes in GPRAM design (see [17]).

## References

[1] C. E. Shannon, "A Mathematical theory of communication," *Bell Syst. Tech. J.*, Vol. 27, pp.370-423, July 1948

[2] J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering*. John Wiley & Sons,Inc., New York-London -Sydney, 1965.

[3] M. K. Simon, M. K. Hinedi, andW. C. Lindsey, *Digital Communication* Techniques. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[4] J. B. Proakis and M. Salehi, *Digital Communications*, 5th ed. McGraw-Hill, 2008

[5] S. Lin and D. Costello, *Error control coding: fundamentals and applications*, 2nd ed. Pearson-Prentice Hall, 2004.

[6] F. Alajaji, N. Phamdo, and T. Fuja, "Channel codes that exploit the residual redundancy in CELP-encoded speech," *IEEE Trans. Speech Audio Processing*, vol. 4, pp. 325-336, Sept. 1996.

[7] W. Xu, J. Hagenauer, and J. Hollmann, "Joint source-channel decoding using the residual redundancy in compressed images," *in Proc. Int. Conf. Communications*, Dallas, TX, June 1996.

[8] H. Kuai, F. Alajaji, and G. Takahara, "Tight error bounds for nonuniform signaling over AWGN channels," *IEEE Trans. Inform. Theory*, vol. 46, pp. 2712-2718, Nov. 2000.

[9] I. Korn, J. P. Fonseka, and S. Xing, "Optimal binary communication with nonequal probabilities," *IEEE Trans. Commun.*, vol. 51, no. 9, pp. 1435-1438, Sep. 2003.

[10] L. Wei and I. Korn, "Optimal M-ASK/QASK with Non-equal Symbol Probabilities," *IET Communications*, Vol. 5, No.6, April 2011, pp.745-752.

[11] L. Wei, "Optimal M-ary Orthogonal Signaling with Non-equal Symbol Probabilities," IEEE ISIT 2012, submitted.

[12] L. Wei, "Channel Capacity and Constellation Optimization of MASK Input AWGN with Non-equal Symbol Probabilities," in preparation

[13] L. Wei, D. M. Levi, R. Li, S. Klein "Feasibility Study on a Hyperacuity Device with Motion Uncertainty: Two-point Stimuli," *IEEE Trans. on Systems. Man, and Cybernetics,* April 2007, pp. 385-97.

[14] B. A. Olshausen and D. J. Field: Emergence of simple cell receptive field properties by learning a sparse code for nature images, *Nature*, **381** (1997) 607-609.

[15] Burak Y, Rokni U, Meister M, Sompolinsky H "Bayesian Model of dynamic image stabilization in the visual system," *Proc Natl Acad Sci,* USA, 107: 19525-19530, 2010.

[16] L. Wei, "General Purpose Representation and Association Machine, Part 1: Introduction and Illustrations," and "General Purpose Representation and Association Machine, Part 2: Biological Implications," IEEE Southeastcon, Mar. 2012, Orlando, USA.

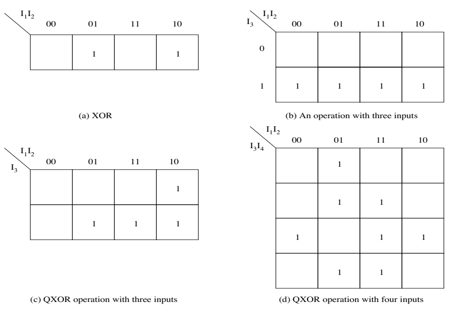
[17] H. Li, B. Dai, S. Schultz, and L. Wei, "General Purpose Representation and Association Machine, Part 3: Prototype Study using LDPC codes" IEEE ISIT, July, 2012, MIT, USA.

[18] R. J. McEliece, D. J. C. MacKay, Jung-Fu Cheng Turbo decoding as an instance of Pearls belief propagation algorithm, *IEEE JASC* Volume: 16 Issue: 2 , Feb. 1998, Page(s): 140 -152

[19] F. R. Kschischang, B. J. Frey and H. A. Loeliger, Factor graphs and the sum-product algorithm, IEEE Trans. Inform. Theory, Vol. 47, pp.498-519, Feb. 2001.

[20] J. Hagenauer "Analog decoding and Beyond," *ITW2001,* Cairns, Australia, Sept. 2-7, 2001, p.126-127.

[21] C. Mead, *Analog VLSI and Neural Systems,* Addison-Wesley, 1989.



(a) XOR
(b) An operation with three inputs
(c) QXOR operation with three inputs
(d) QXOR operation with four inputs

Fig. 1. Karnaugh maps for XOR and QXOR operations.

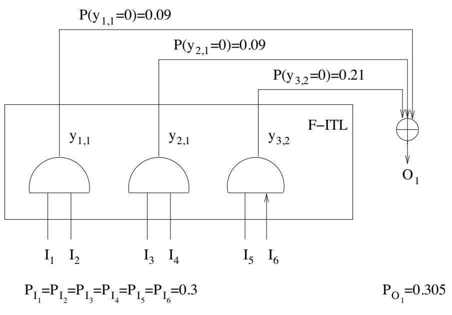


Fig. 2. Code Structure with F-ITL

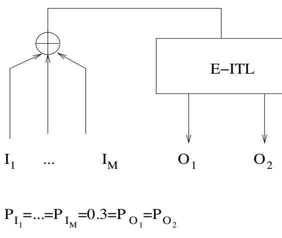
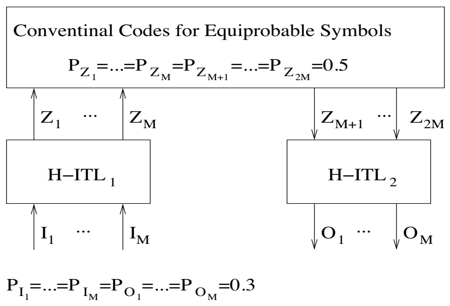


Fig. 3. Code Structure with E-ITL



Conventinal Codes for Equiprobable Symbols
P_Z1=...=P_ZM=P_ZM+1=...=P_Z2M=0.5
H−ITL 1
H−ITL 2
P_I1=...=P_IM=P_O1=...=P_OM=0.3

Fig. 4. Code Structure with H-ITL

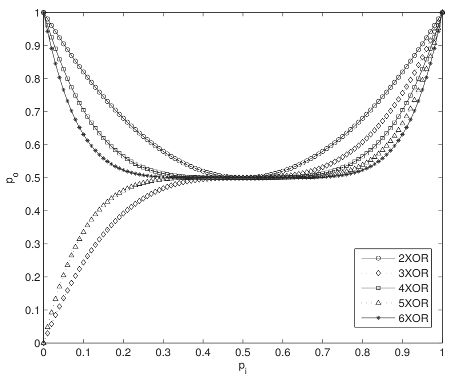
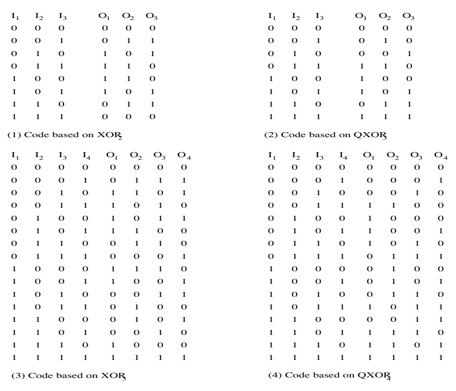


Fig. 5. Probabilities of output symbols versus input symbols



| $I_1$ | $I_2$ | $I_3$ | $O_1$ | $O_2$ | $O_3$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |

(1) Code based on $XOR_2$

| $I_1$ | $I_2$ | $I_3$ | $O_1$ | $O_2$ | $O_3$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

(2) Code based on $QXOR_3$

| $I_1$ | $I_2$ | $I_3$ | $I_4$ | $O_1$ | $O_2$ | $O_3$ | $O_4$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

(3) Code based on $XOR_3$

| $I_1$ | $I_2$ | $I_3$ | $I_4$ | $O_1$ | $O_2$ | $O_3$ | $O_4$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

(4) Code based on $QXOR_4$

Fig. 6. Codes with XOR and QXOR operations.