Digital VLSI Architectures for Advanced Channel Decoders

POLITECNICO DI TORINO

SCUOLA DI DOTTORATO
Dottorato in Ingegneria Elettronica – XXVIII ciclo

Tesi di Dottorato

Digital VLSI Architectures for Advanced Channel Decoders

Ing. Andrea Dario Giancarlo Biroli

Tutore
Prof. Guido Masera

27 May 2016

To my beloved parents

Contents

List of Figures

Chapter 1

Summary

Every modern digital communication and storage system makes strong use of error-correcting codes. Typical applications are wireless communications, sensor networks, optical communications, computer hard drives, flash memories and space probes. The demand of codes with better error-correcting capability for new and emerging applications as well as the design and implementation of those high-gain error-correcting codes are open challenges. The direct mapping of these algorithms to hardware implementation often leads to very high complexity architecture because they usually involve complex mathematical computations so lots of research is performed to reduce complexity while enhancing decoding performance.
This work aims to focus on Polar codes, which are a recent class of channel codes with the proven ability to reduce decoding error probability arbitrarily small as the block-length is increased, provided that the code rate is less than the capacity of the channel. This property and the recursive code-construction of this algorithms attracted wide interest from the communications community.
Hardware architectures with reduced complexity can efficiently implement a polar codes decoder using either successive cancellation approximation or belief propagation algorithms. The latter offers higher throughput at high signal-to-noise ratio thanks to the inherently parallel decision-making capability of such decoder type.
In Chapter 2 is proposed an overview on technological improvements from dawn of channel coding techniques and related forefront applications to actual solutions. In the same chapter general characteristics of error correcting block codes and iterative decoding properties are presented. Effects of Additive White Gaussian Noise are explored in relation to different channel modulations.
At Chapter 3 the polar codes theory is deeply analyzed. Channel polarization, effects of channel combining and splitting are described. Coding and decoding methods, approximations and performance evaluation are presented. A novel inspection on scheduling algorithms for Belief Propagation is given. Drawn observation enable to focus the research from simple maximum achievable throughput to throughput over

area efficiency ratio. Property of polar codes graph representation like uniformity or trellis redundancy are evaluated.
The model used to simulate different algorithms and hardware architectures is introduced in the fourth chapter. Simulation results are commented and general observation are drawn.
In Chapter 5 the state-of-art background for belief polar code decoders is presented. Chapter 6 describes implemented architectures in relation with proposed best scheduling algorithms discovered. Synthesis results of processing elements and overall structures are presented. Researched objectives in terms of absolute throughput and area efficiency are met.
Last chapter reports conclusion and compare achieved results with best architectures from literature. It also reports suggested future works in the direction of further improve obtained results.

Chapter 2

Introduction

2.1 Brief history of information and coding theory

The most fundamental parameter regarding a communication channel is its capacity C, a concept introduced by Claude E. Shannon in 1948 [5] that showed how to calculate the highest rate at which information can be transmitted reliably over the channel. This cut-off rate is considered the be the "practical coding limit" and is also known as "Shannon limit".
For the next half century, channel coding central objective was to find practical coding schemes that could approach channel capacity on well-understood channels such as the additive white Gaussian noise (AWGN) channel. This goal proved to be challenging, but not impossible. [6]
For the first couple of decades, algebraic coding dominated the field of channel coding. The principal objective of algebraic coding theory is to maximize the minimum Hamming distance for a given code to maximize error correction power. To improve error detection and correction, binary linear block codes were invented: Hamming code [7], Golay [8] codes and Reed-Muller codes [9] [10].
Even though algebraic codes were clearly limited to a few, channel models, their development appeared in the early sixties to be an area of considerable potential payoff, in applied as well as theoretical directions, beginning with the algebraic representation [11] of all linear error-correcting codes, and culminating at the end of the decade with the important class of Bose-Chaudhuri-Hocquenghem (BCH) codes [12], [13] and following Reed–Solomon [14] which are a class of non-binary BCH codes.
On the road to modern capacity-approaching codes, an essential step has been to replace hard-decision with soft-decision decoding (decoding that takes into account the reliability of received channel outputs). The earliest soft-decision decoding algorithm in literature is Wagner decoding, described in [15].

An alternative line of development that was more directly inspired by Shannon's

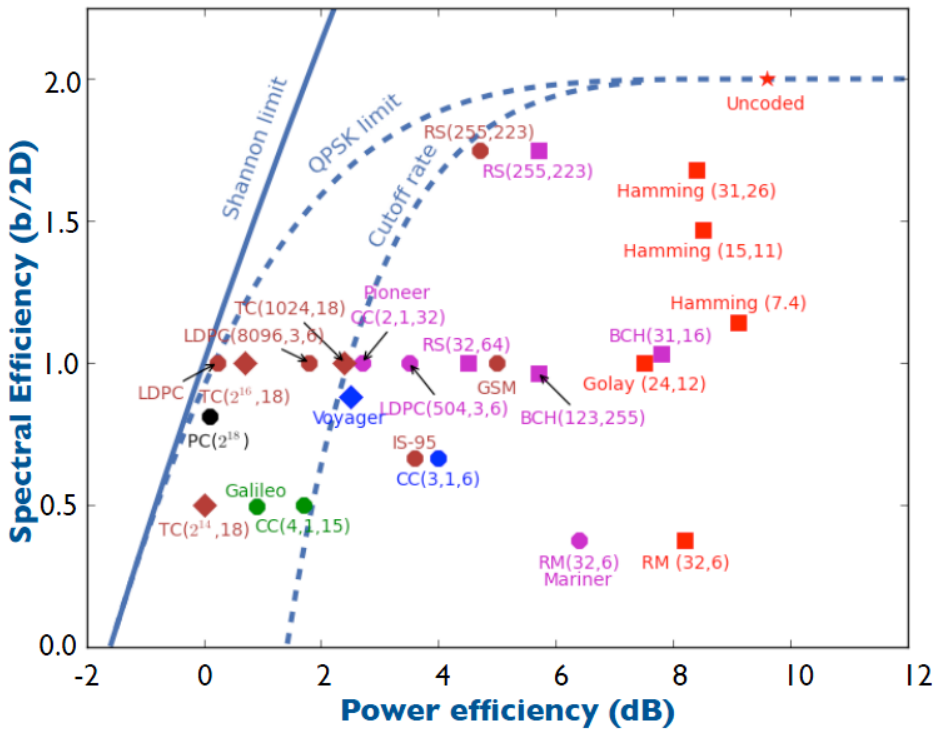


Figure 2.1: *Performance of codes example for Hamming, Golay, Reed-Muller (RM), concatenated (CC), BHC, Reed-Solomon (RS), LDPC, turbo (TC) and polar (PC) codes. It is also reported practical implementations of codes for deep-space probes: Mariner (RM) 1969, Pioneer (CC) 1968, Voyager(RS) 1977, Galileo (Viterbi) 1989. And for cellular applications: GSM (Viterbi) 1987, IS-95 standard (RM)1995. Figure from [1].*

probabilistic approach to coding is named "Probabilistic coding". It is more concerned with finding classes of codes that optimize average performance as a function of coding and decoding complexity. Following coding schemes fall into this class.
Convolutional codes and product codes were invented by Peter Elias in 1954-1955 [16] [17]. In 1962 Gallager developed Low-density parity-check code (LDPC code) [18] but his work was not taken into account until nineties by David MacKay.
Concatenated codes were introduced by Dave Forney in 1966 [19]. It is based on an inner and an outer code that can be relatively short and easy to encode and decode, while the resulting concatenated code is powerful and much longer.
To decode convolutional codes, Andy Viterbi in 1967 proposed an "asymptotically

optimal" algorithm (known as Viterbi Algorithm) which performs a maximum-likelihood decoding through the search of the closest sequence on a trellis with finite number of states. In nineties, we can date the main developments in codes approaching the Shannon's bound: Turbo codes (1993) [20], the rediscovery of LDPC (1996) [21],[22] and Polar codes (2008) [23].
In Figure 2.1 a performance comparison among cited codes and forefront applications of these codes can be observed. Notice that from the start in 1948, coding research has been oriented in closing the gap between practical achievable performance (with proper code structure influenced by technological constrains) and the Shannon bound ($\frac{E_b}{N_0} > \frac{2^r-1}{r} > \ln 2 \simeq -1.6dB$).

2.2 Near-Capacity Channel Codes

2.2.1 Linear block codes

In Figure 2.2 a generic numerical transmission classical point to point scheme is presented, even if nowadays a part of research, especially in multimedia applications, study the possibility of combining source and channel coding, in order to optimize transmissive resources. Block codes are channel codes which allow to reveal and correct errors induced by transmission channel. It is briefly reported fundamental

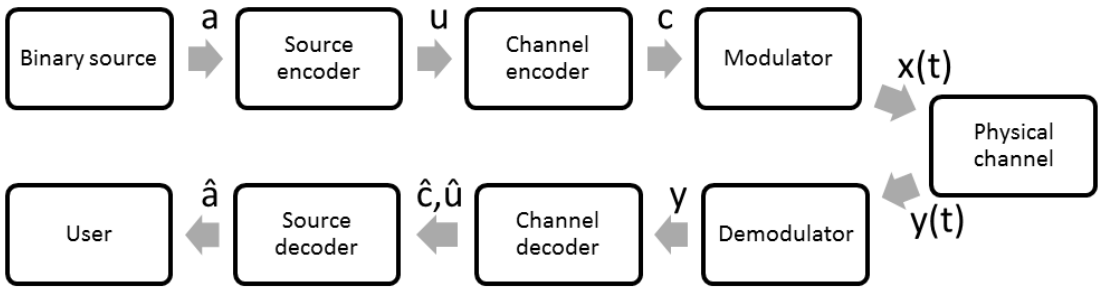


Figure 2.2: *Scheme representing transmitter, transmission channel and receiver.*

concept on linear block codes.
Let $\mathbb{F} = GF(2)$ be a Galois Field of order 2 and $\mathbb{F}^N$ a vector space of dimension N on GF(2). A block code $\mathcal{C} \subseteq \mathbb{F}^N$ is an application $\mathcal{C} : \mathbb{F}^K \mapsto \mathbb{F}^N$. It is linear if:

- $\forall \lambda \in \mathbb{F},\ u \in \mathcal{C}$, also $\lambda u \in \mathcal{C}$;
- $\forall u, v \in \mathcal{C}$, also $u \oplus v \in \mathcal{C}$.

$\mathcal{C}$ is characterized by (N, K, d) where d is the *minimal distance* of the code, defined as in 2.1.

$$d \triangleq \min_{u \neq v} d_H(u, v) \tag{2.1}$$

where d_H is the *Hamming distance*. Hence information bitstream is divided in vectors u, of dimension 1xK and mapped on codeword x of dimension 1xN, through linear transformation 2.2.

$$x = uG \tag{2.2}$$

The ratio $R = K/N$ is called code *rate*. A code is called *systematic* if x can be rewritten as $x = [u|c]$. Matrix G^T of dimension KxN is called generating matrix and it is in systematical form if it respect following three properties:

- Leftmost not-zero value of every row is a 1;
- Every column containing the leftmost 1 has all other input equal to 0;
- If the leftmost not-zero value of the row i is in column t_i, then $t_1 < t_2 ... < t_r$

So a particular case occurs when the matrix is built as follows.

$$G = [I_K|P_{N-K}] \tag{2.3}$$

where I_K is an identity matrix KxK and P_{N-K} of dimension Kx$(N-K)$.
So the coding for a code $\mathcal{C}$ has the desirable characteristic that information symbols appear clearly in the *codeword* x; more in general, the symbol x_i will appear in position t_i in the codeword $x = uG$, if the leftmost value of row i-th of G happens at column t_i.
Every code (linear or not), that verify the property that exist a rule of coding so that information symbols appear clearly is called *systematic*. It can be proven that every linear code can be made systematic.
In a digital transmission channel, sent *codeword* x can be corrupted by noisy channel and the received word y can be expressed as in 2.4.

$$y = x + e \tag{2.4}$$

where e is the error vector that represent channel effects. Knowing the received signal y, the decoder tries to evaluate an estimate of $\widehat{x}$ of the transmitted information word u. Estimated *codeword* $\widehat{x}$ must be calculated to be the closest word (inside the codeword set $\mathcal{C}$) to the transmitted one. To do so, it is chosen a method based on *codeword* distance (either Hamming or Euclidean distance). To perform di evaluation, the decoder should compute $(N - K)$ parity bits and compare them with original parity bits of x, if equal, the received word is correct. So an error occur if at least one parity check fails.

2.2.2 Coding

For simplicity of exposition, it will be considered in the following paragraph that message bits u are placed at the end of the codeword. The codeword can be divided in M check bits followed by K message bits, so the codeword is divisible in M control bits followed by K message bits u.

$$x = [c|u] \tag{2.5}$$

Thanks to the previous assumption, H parity matrix can be divided in a A matrix MxM which occupy first M column of H and a B matrix MxK which occupy remaining columns of H.

$$H = [A|B] \tag{2.6}$$

So the requirement that a codeword comply with all parity checks ($Hx = 0$) can be written as:

$$Ac + Bu = 0 \tag{2.7}$$

Assuming A is not singular, follows:

$$c = A^{-1}Bu \tag{2.8}$$

A can be singular for some choices so codeword bits are message bits, however it always exist a choice of A not singular if H rows are linearly independent. Anyway it is possible that H rows are not linearly independent (i.e. some rows are redundant). This particular case is not very interesting.
Equation 2.8 defines which should be control bits, however the way those bits are evaluated can be fulfilled in different ways.
The developed software does not involve the use of a matrix representation in sparse form since the high number of ones in the generator polar code matrix, equal to $3^{log_2(N)}$ where N is the codeword length. The explicit representation of the generator matrix G can be expressed for the systematic form as in 2.9.

$$G = [A^{-1}B|I_K] \tag{2.9}$$

This form, expressed for a generic generator matrix, does not take into account specific recursive structure of polar codes that will be discussed in detail in chapter 3.3.

2.3 Iterative Decoding

Actual high performance decoders work with data represented as information called “soft-information”.

Block codes can be decoded through two different strategies which depend on input data format at the decoder. When decoder inputs are made by bits (0 or 1), it is called "Hard" decoding; while if inputs are real values (i.e. 0.784) it is considered "Soft" decoding.
Recalling Figure 2.2, channel decoder inputs are digital demodulator outputs, so if this component which receives channel signal, represent it as real values, then following decoder will be "soft". Instead if the digital demodulator takes its decision also on transmitted data choosing if the sent bit was a 0 or 1, it will be called "hard" decoder. Moreover, since in soft decoding, real values are involved, the metric for the chosen distance is the Euclidean, while for hard decoding Hamming distance is used.
On decoding performance point of view, it is well known in information theory that Soft Decoding overcame Hard Decoding because it is able to correct a larger number of errors. Decoding process becomes quite difficult when a word is searched in the space of all possible messages, which minimize the probability of having an error. Hard Decoding limits this process in the search into the subspace of binary words which are a limited set. Instead Soft Decoding "extends" the search by considering all real values, this means that the research space is considerably augmented. However the probability of finding transmitted codeword is also increased; and also adopting same error correction code, soft decoding can achieve better gains compared to hard decoding. From these remarks appear that high performance channel decoders use Soft decoding strategy even if decoder architectures are more complex.
For the specific case of Polar code decoders, they do not work on raw soft data received from the channel, but on a "reliability" measure obtained from the data. These metrics represent how received signal is close to bit 0 or 1 of the transmitted signal; they are called *likelihood-ratio* (λ_d) and defined as in 2.10

$$\lambda_d = \frac{P(y_d|x_d = 0)}{P(y_d|x_d = 1)} = \frac{1 - P_d}{P_d} \tag{2.10}$$

Where y_d is the d-th received bit, x_d is d-th transmitted bit and P_d indicate the probability of receiving a bit 1 in position d given that bit 1 has been transmitted in position d. Let us remark that the conditional probability evaluated on input signal needs the knowledge on the transmitted information, that is if it has been sent a 1 or a 0 from transmitter.
It is also defined the *Log-likelihood-ratio* (LLR) as in 2.11.

$$\Lambda_d \triangleq ln(\lambda_d) = ln\frac{P(y_d|x_d = 0)}{P(y_d|x_d = 1)} = ln\frac{1 - P_d}{P_d} \tag{2.11}$$

The decoder purpose is to determine the transmitted correct value, starting from the knowledge of the received information, iteratively updating *likelihood-ratio* metrics,

that means to determine the a posteriori probability on received data. Since these metrics are probabilities, the iterative update will be multiplication among them and resulting probability will be interpreted as an information on how close the received message is to a codeword. The choice of using logarithms is therefore motivated by the easiness to implement an algorithm which presents sums instead of multiplications, in fact the use of multiplicative blocks is more expensive in terms of hardware and less stable form numerical representation point of view compared to sum blocks. Moreover the introduction of the logarithmic domain does not modify the overall behaviour of the decoder.
For the AWGN channel and BPSK modulation some simplification occur to the evaluation formula of LLRs as presented in the following chapter 2.4, while for QAM modulation a detailed overview is given in 2.5.

2.3.1 Optimum decoding

The purpose of decoders is to find the codeword $\widehat{c}$ with the highest probability of having been sent over the channel, given the channel output y and the knowledge of the used code:

$$\widehat{c} = \underset{c' \in \mathcal{C}}{arg\max}\, P(c = c'|y) \tag{2.12}$$

Such kind of decoding is called *word maximum a posteriori* (W-MAP). The code knowledge appears in the conditioned probability. Using Bayes theorem the a posteriori probability $P(c = c'|y)$ can be expressed as:

$$P(c|y) = \frac{P(y|c)P(c)}{P(y)} = \frac{P(y|c)P(c)}{\sum\limits_{c \in \mathcal{C}} P(y|c)P(c)} \tag{2.13}$$

If the a priori probability $P(c)$ is equal for every c (equiprobable source), since $P(y)$ does not depend from c then 2.12 can be expressed as:

$$\widehat{c} = \underset{c' \in \mathcal{C}}{arg\max}\, P(y|c = c') \tag{2.14}$$

This criterion is known as *word maximum likelihood* (W-ML). $P(y|c)$ is the likelihood function when expressed for a particular y and for a given c it represents the density probability function.
If the channel is memoryless, it is also valid:

$$P(y|c = c') = \prod_{j=0}^{N-1} P(y_j|c_j = c'_j) \tag{2.15}$$

So the W-ML criterion becomes:

$$\widehat{c_j} = \underset{c' \in \{0,1\}}{arg\ \max} P(y_j | c_j = c') \tag{2.16}$$

The only way to obtain an optimal W-MAP is to test every memory word, which means 2^K combinations if a binary source is considered. W-MAP and W-ML decoders are optimum equivalent decoders if the source is equiprobable.

2.4 Log-Likelihood ratio for AWGN channel and BPSK modulation

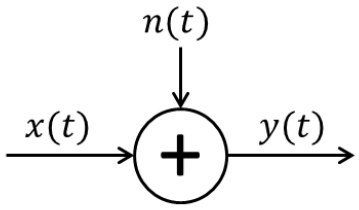


Figure 2.3: *Scheme of AWGN channel.*

Supposing to represent the transmission medium as an AWGN (Additive White Gaussian Noise) channel with $\mathcal{N}(0, \sigma^2 = \frac{N_0}{2})$, the received signal $y(t)$ can be written as $y(t) = x(t) + n(t)$, where $x(t)$ is the transmitted signal, so it is considered a constellation of only two symbols as for binary phase shift keying (BPSK) modulation, and $n(t)$ represent the noise component (Figure 2.3).
Phase shift keying (PSK) is widely used in the communication industry and BPSK is the simplest implementation with just two signals with different phases.
Typically these two phases are 0 and π, the signals are:

$$\begin{aligned} s_1(t) &= Acos(2\pi f_c t), \quad 0 \leq t \leq T, \quad for\ 1 \\ s_2(t) &= -Acos(2\pi f_c t), \quad 0 \leq t \leq T, \quad for\ 0 \end{aligned} \tag{2.17}$$

These signals are called *antipodal* and they can also be graphically represented by a *signal constellation* (Figure 2.4) in the coordinate system with

$$\Phi_1(t) = \sqrt{\frac{2}{T}} cos(2\pi f_c t)\ for\ 0 \leq t \leq T \tag{2.18}$$

The energy of a transmitted signal is therefore:

$$E = A^2 \frac{T}{2} \tag{2.19}$$

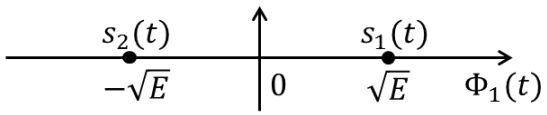


Figure 2.4: *BPSK signal constellation.*

Let us remind that for AWGN channel the density probability function is defined

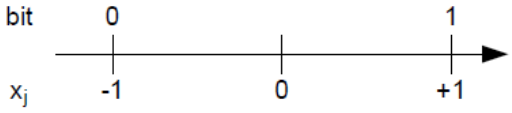


Figure 2.5: *Used Mapping.*

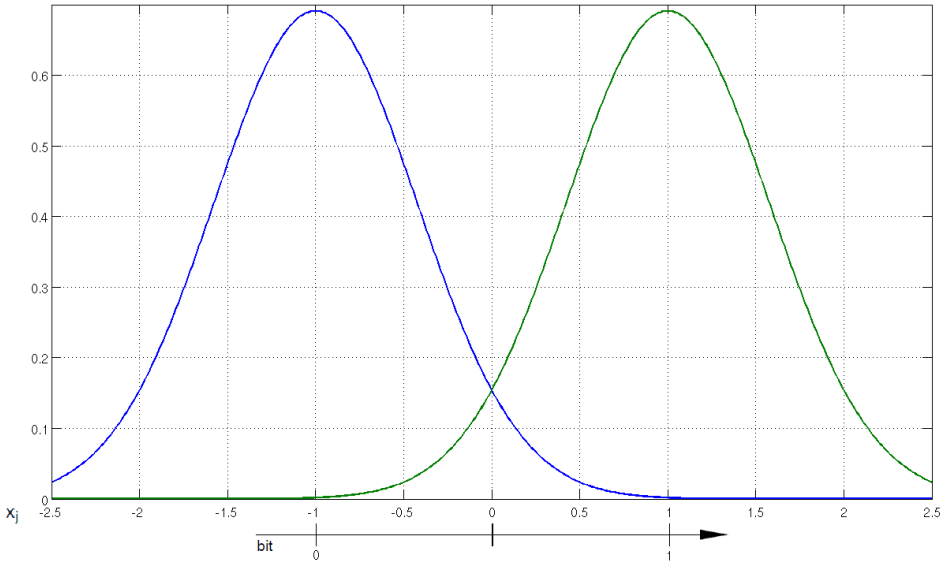


Figure 2.6: *Probability density for the AWGN channel.*

by 2.20

$$f(r_i|b_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(r_i - (1-2b_i))^2}{2\sigma^2}} \tag{2.20}$$

where b_i is the bit that is transmitted and is mapped as $b_i = 0$ for symbol $x = s_2$ and $b_i = 1$ for symbol $x = s_1$ with unit energy that is normalized (as in Figure 2.5). r_i is the received symbol. So for the Gaussian channel with variance σ is given by $\frac{E_S}{N_0} = \frac{1}{2\sigma^2}$. As LLR was previously defined (2.11) it can be written the following.

$$\begin{aligned}
\Lambda_d = ln\frac{P(y_d|x_d = s_2)}{P(y_d|x_d = s_1)} = \quad & ln\frac{f(r_d|b_d = 0)}{f(r_d|b_d = 1)} \quad = \\
& ln\frac{e^{-\frac{(r_d-1)^2}{2\sigma^2}}}{e^{-\frac{(r_d+1)^2}{2\sigma^2}}} \quad = \\
& ln\ e^{-\frac{(-2r_d-2r_d)}{2\sigma^2}} \quad = \\
& 2\frac{r_d}{\sigma^2} \quad = \\
& 4\frac{E_S}{N_0}r_d
\end{aligned} \tag{2.21}$$

The final result of 2.21 will be used to compute simulations probability of the AWGN transmission channel with BPSK signal mapping of sent data for the software implementation.

2.5 QAM modulation

Quadrature amplitude modulation (QAM) is a class of nonconstant envelope schemes that can achieve higher bandwidth efficiency than M-ary Phase Shift Keying (MPSK) with the same average signal power. The first QAM scheme was proposed by C. R. Cahn in 1960 [24]. He extended phase modulation to a multi-amplitude phase modulation. This means that more than one amplitude is associated to every allowed phase. Many different types of constellation have been presented in literature over the years. Adopting the description of QAM constellation in [25] let us introduce three most significative types .
Type I constellation is a fixed number of signal points (or phasors) equally spaced on N circles, where N is the number of amplitude levels (Figure 2.7(a)). This type of constellation suffers of the problem that the points on the inner ring are closest together in distance so are most vulnerable to errors. To mitigate this problem, type II constellation was proposed by Hancock and Lucky a few months later [26] (Figure

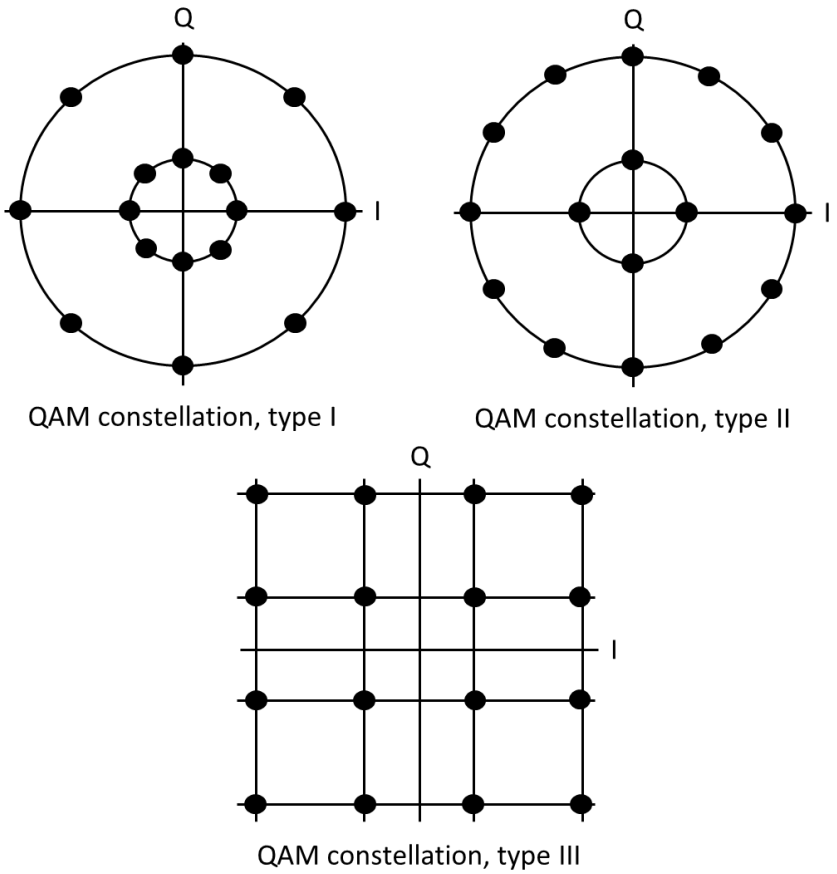


Figure 2.7: *Examples of type I, II, and III QAM constellations.*

2.7(b)). Signal points of type II constellation are still on circles, but the number of points on the inner circle is lower than the number of points on the outer circle. This makes the distance between two adjacent points on the inner circle approximately equal to that on the outer circle.
In 1962 Campopiano and Glazer [27] proposed the square QAM constellation shown in Figure 2.7(c). This III type of constellation does not offer a big improvement in performance over the type II system, but its implementation is considerably simpler than that of type I and II. It can be easily generated by two M-ary Amplitude Shift Keying (MASK) signals sent on two phase-quadrature carriers and demodulated to yield the two quadrature components. Few of the other constellations offer slightly better error performance for AWGN channels, but with more complicated system implementation. Due to this, the type III constellation has been the most widely used system. QAM is used for instance in modems designed for telephone channels.

QAM schemes starting from uncoded 16QAM to trellis coded 128QAM are used in CCITT telephone circuit modem standards V.29 to V.33. A very active research on QAM applications are also satellite systems, point-to-point wireless systems, and mobile cellular telephone systems.[25]

2.5.1 Square QAM

Considering both amplitude and phase modulation in a scheme, a general QAM signal can be written as:

$$s_i(t) = A_i cos(2\pi f_c t + \theta_i), \ for\ i = 1,2,...M \tag{2.22}$$

where A_i is the amplitude and θ_i is the phase of the i-th signal in the M-ary signal set. The pulse shaping parameter $p(t)$ which multiply $s_i(t)$ has been neglected because there is no particular interest to discuss the improvement of the spectrum and the intersymbol interference (ISI) for QAM current description for simulation, but it is good to know for a practical implementation. In fact even if pulse shaping is not desired, it inevitably occurs due to the limit of bandwidth for the considered system. Pulse shaping is usually achieved through filtering so $P(f) = H_T(f)H_C(f)H_R(f)$ (or equivalently $p(t) = h_T(t) * h_C(t) * h_R(t)$) where $H_T(f)$,$H_C(f)$ and $H_R(f)$ are the spectral responses of the transmitter filter, channel, and receiver filter. A common choice of $P(f)$ is the raised-cosine (or its approximated delayed version due to causality issues), whose time domain function $p(t)$ has zero values at sampling instants except at $t = 0$, thus $p(t)$ incurs no ISI.
Equation 2.22 can be written as

$$\begin{aligned} s_i(t) &= I_i\sqrt{\frac{E_0}{T}}cos(2\pi f_c t) + Q_i\sqrt{\frac{E_0}{T}}sin(2\pi f_c t) \ = \\ &= I_i\sqrt{\frac{E_0}{2}}\Phi_1(t) + Q_i\sqrt{\frac{E_0}{2}}\Phi_2(t) \end{aligned} \tag{2.23}$$

where (I_i, Q_i) are a pair of independent integers which determine the location of the signal point in the constellation. E_0 is the energy of the signal with the lowest amplitude, or equivalently the average energy of the signal when (I_i, Q_i) are normalized. The two orthonormal function in 2.23 are:

$$\begin{aligned} \Phi_1(t) &= \sqrt{\frac{2}{T}}cos(2\pi f_c t) \ , \ for\ 0 \le t \le T \\ \Phi_2(t) &= \sqrt{\frac{2}{T}}sin(2\pi f_c t) \ , \ for\ 0 \le t \le T \end{aligned} \tag{2.24}$$

The pair (I_i, Q_i) is an element of an LxL matrix with all possible combinations. For instance a 16QAM matrix has $L = 4$ and is represented by:

$$[I_i, Q_i] = \begin{bmatrix} (-3,\ 3) & (-1,\ 3) & (1,\ 3) & (3,\ 3) \\ (-3,\ 1) & (-1,\ 1) & (1,\ 1) & (3,\ 1) \\ (-3,-1) & (-1,-1) & (1,-1) & (3,-1) \\ (-3,-3) & (-1,-3) & (1,-3) & (3,-3) \end{bmatrix} \tag{2.25}$$

The generalized M-QAM matrix, where $M = 4^n$, n = 1,2,3, ..., and $L = \sqrt{M}$, can be written as:

$$[I_i, Q_i] = \begin{bmatrix} (-(L-1),(L-1)) & (-(L-3),(L-1)) & \cdots & ((L-1),(L-1)) \\ (-(L-1),(L-3)) & (-(L-3),(L-3)) & \cdots & ((L-1),(L-3)) \\ \vdots & \vdots & & \vdots \\ (-(L-1),-(L-1)) & (-(L-3),-(L-1)) & \cdots & ((L-1),-(L-1)) \end{bmatrix} \tag{2.26}$$

The constellation can be conveniently expressed in terms of (I_i, Q_i), but the phasor for the square QAM is

$$s_i = \left(I_i \sqrt{\frac{E_0}{2}}, Q_i \sqrt{\frac{E_0}{2}} \right) \tag{2.27}$$

And consequently the average energy is given by:

$$E_{avg} = E\left\{ \frac{E_0}{2} \left(I_i^2 + Q_i^2 \right) \right\} = \frac{E_0}{2} \left[E\left\{ I_i^2 \right\} + E\left\{ Q_i^2 \right\} \right] = E_0 E\left\{ I_i^2 \right\} \tag{2.28}$$

where

$$\begin{aligned} E\left\{ I_i^2 \right\} &= \frac{1}{L} \left[(-(L-1))^2 + (-(L-3))^2 + \cdots + (L-3)^2 + (L-1)^2 \right] = \\ &= \frac{2}{L} \left[1^2 + 3^2 + \cdots + (L-1)^2 \right] = \frac{2}{L} \left[\sum_{i=1}^{L/2} (2i-1)^2 \right] = \\ &= \tfrac{1}{3}(L^2 - 1) = \tfrac{1}{3}(M-1) \end{aligned} \tag{2.29}$$

Thus the average power is

$$P_{avg} = \frac{E_0}{3T}(M-1) = \frac{P_0}{3}(M-1) \tag{2.30}$$

Where P_0 is the power associated to the smallest signal. The average transmitted power required to achieve a given minimum distance is only slightly greater than the average power required for the best M-ary QAM signal constellation. For these reasons, rectangular M-ary QAM signals are most frequently used in practice.

Error probability

Square QAM signal constellations have the distinct advantage of being easily generated as two MASK signals impressed on the in-phase and quadrature carriers, each having $L = \sqrt{M}$ signal points. An error occurs if the additive Gaussian noise either n_1 or n_2 on the orthonormal bases Φ_1, Φ_2 is large enough to cause an error in one of the two MASK signals. A QAM symbol is detected correctly only when two MASK symbols are detected correctly. Thus the probability of correct detection of a

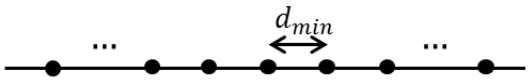


Figure 2.8: *The MASK constellation.*

QAM symbol is the product of correct decision probabilities for constituent MASK systems.

$$P_{c,M-QAM} = P^2_{c,\sqrt{M}-ASK} = (1 - P_{e,\sqrt{M}-ASK})^2 \tag{2.31}$$

And consequently the error probability is

$$\begin{aligned} P_{e,M-QAM} &= 1 - (1 - P_{e,\sqrt{M}-ASK})^2 = \\ &= 2P_{e,\sqrt{M}-ASK} - P^2_{e,\sqrt{M}-ASK} \end{aligned} \tag{2.32}$$

where $P_{e,\sqrt{M}-MASK}$ is symbol error probability of a single MASK with one-half the average power of the QAM signal.
Recalling that the symbol error probability of a MASK signal is given by

$$\begin{aligned} P_{e,M-ASK} &= \frac{1}{M}\sum_{m=1}^{M} P(err|m) \\ &= \frac{1}{M}\left[2(M-2)Q\left(\frac{d_{min}}{\sqrt{2N_0}}\right) + 2Q\left(\frac{d_{min}}{\sqrt{2N_0}}\right)\right] = \\ &= \frac{2(M-1)}{M}Q\left(\frac{d_{min}}{\sqrt{2N_0}}\right) = \\ &= 2\left(1 - \frac{1}{M}\right)Q\left(\sqrt{\frac{6log_2M}{M^2-1}\frac{E_{b,avg}}{N_0}}\right) \end{aligned} \tag{2.33}$$

where d_{min} is the distance between two signal of the constellation (as in Figure 2.9) and $Q(x) = \frac{1}{2} erfc\left(\frac{x}{\sqrt{2}}\right)$.[2] It comes that the 2.32 becomes:

$$P_{e,\sqrt{M}-ASK} = 2\left(1 - \frac{1}{\sqrt{M}}\right) Q\left(\sqrt{\frac{3log_2 M}{M-1}\frac{E_{b,avg}}{N_0}}\right)$$

$$P_{c,M-QAM} = 4\left(1 - \frac{1}{\sqrt{M}}\right) Q\left(\sqrt{\frac{3log_2 M}{M-1}\frac{E_{b,avg}}{N_0}}\right) - \left(2\left(1 - \frac{1}{\sqrt{M}}\right) Q\left(\sqrt{\frac{3log_2 M}{M-1}\frac{E_{b,avg}}{N_0}}\right)\right)^2 \quad (2.34)$$

That is an exact expression for the symbol error probability of a M-ary QAM signal.

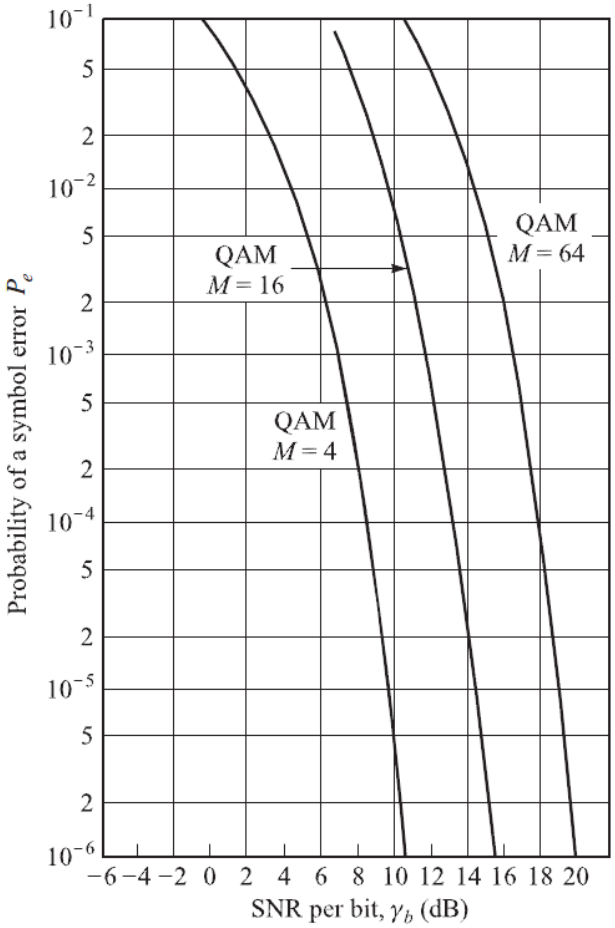


Figure 2.9: *The QAM symbol probability. From [2].*

Gray coding

To minimize the bit error of n-tuples of QAM points, Gray coding is typically used for mapping these data. In fact the Grey code is constructed imposing that adjacent symbols differ only for one bit. In Figure 2.10 an example of Gray coding is given

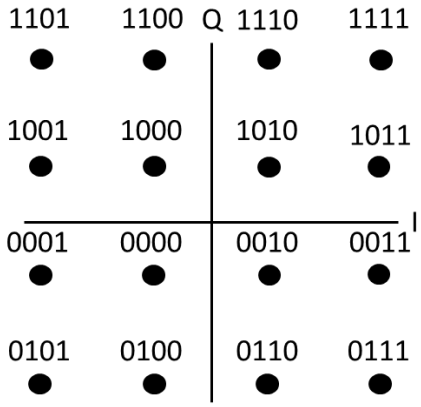


Figure 2.10: *The 16-QAM constellation with Gray coding.*

for 16-QAM constellation. Observing that square QAM can implement perfectly Gray code, the bit error probability can be obtained from symbol error probability as

$$P_b \cong \frac{P_S}{\log_2 M} \tag{2.35}$$

supposing that there is only one bit of difference between close symbols.

Chapter 3

Polar code theory

Polar coding discover came out by studying a technique to improve the cutoff rate of sequential decoding of a concatenated decoding scheme. Starting from a vector channel and splitting it into multiple correlated subchannels, it was possible to use a different sequential decoder on each subchannel. Polar coding was originally designed as a simple recursive operation to be used as low complexity inner code that implemented this behaviour. But it was noticed that polar coding performance was so good that no outer convolutional code was needed to increment the cutoff rate to channel capacity.[28]

3.1 Preliminary definitions

3.1.1 Channel models and channel coding

In order to specify a mathematical model for a channel, we shall specify:

1. the set of possible inputs to the channel,
2. the set of possible outputs,
3. for each input, a probability measure on the set of outputs.

Discrete memoryless channels (DMC) are a simple class of channel models and can be defined as follows: the input is a sequence of letters from a finite alphabet $\mathcal{X} = \{a_1, \ldots, a_K\}$, and the output is a sequence of letters from the same or a different alphabet $\mathcal{Y} = \{b_1, \ldots, b_J\}$. Each letter in the output sequence is statistically dependent only on the letter in the corresponding position of the input sequence and is determined by a fixed conditional probability assignment $P(b_j|a_k)$ defined for each letter a_k in the input alphabet and each letter b_j in the output alphabet. For

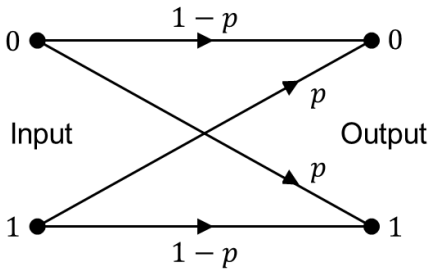


Figure 3.1: *Binary symmetric channel (BSC).*

example, the binary symmetric channel (BSC) (see Figure 3.1) is a discrete memoryless channel (DMC) with binary input and output sequence where each digit in the input sequence is reproduced correctly at the channel output with some fixed probability $1-p$ and is altered by noise into the opposite digit with probability p. In general, for discrete memoryless channels, the transition probability assignment tells us everything that we have to know about how the noise combines with the channel input to produce the channel output. Another class of channel models which bears a more immediate resemblance to physical channels is the class where the set of inputs and set of outputs are each a set of time functions (waveforms), and for each input waveform the output is a random process. A particular model in this class which is of great theoretical and practical importance (particularly in space communication) is the additive Gaussian noise channel. The set of inputs for this model is the set of time functions with a given upper limit on power and the output is the sum of the input plus withe Gaussian noise.[29]
The binary-input discrete memoryless channel (B-DMC) can be defined with $\mathcal{W}: \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X} = \{0,1\}$ is the input alphabet, $\mathcal{Y}$ is the output alphabet, and $\mathcal{W}(y|x)$ are the transition probabilities for every $x \in \mathcal{X}; y \in \mathcal{Y}$. The output alphabet and the transition probabilities may be arbitrary.
Given a B-DMC , there are two channel parameters of primary interest for polar codes: the symmetric capacity and the Bhattacharyya parameter. The symmetric capacity is also known as maximum of the average mutual information, that is the mean of the mutual information, which is a random variable defined as in 3.1.

$$I_{X;Y}(a_k, b_j) = log\frac{P_{X|Y}(a_k|b_j)}{P_X(a_k)} = log\frac{P_{Y|X}(b_j|a_k)}{P_Y(b_j)} = I_{Y;X}(b_j, a_k) \tag{3.1}$$

where $\{a_1, \ldots, a_K\}$ is the X sample space, $\{b_1, \ldots, b_J\}$ is the Y sample space and XY is the joint ensemble with the probability assignment $P_{XY}(a_k, b_j)$. An event $x = a_k$ might be interpreted as the input letter into a noisy discrete channel and $y = b_j$ its output so 3.1 gives the information provided about the event x by the

occurrence of the event y.
The base of the logarithm defines the numerical scale used to measure information. For base 2 logarithms, the numerical value is called the number of *bits* (binary digits) of information, while for base e (natural logarithms) is the number of *nats* (natural units).
The average mutual information between input and output is given by 3.2.

$$I(W) = I(X;Y) \triangleq \sum_{k=1}^{K} \sum_{j=1}^{J} P_{XY}(a_k, b_j) log \frac{P_{X|Y}(a_k|b_j)}{P_X(a_k)} \tag{3.2}$$

For a DMC where $Q(k)$ is the probability to measure to the input integer k and $P(j|k)$ is the transition probability of receiving integer j given k at the channel input, 3.2 can be written as

$$I(W) = I(X;Y) = \sum_{k=1}^{K} \sum_{j=1}^{J} Q(k)P(j|k) log \frac{P(j|k)}{\sum_{i=1}^{K} Q(i)P(j|i)} \tag{3.3}$$

The capacity C of a discrete memoryless channel (DMC) can be written as

$$C \triangleq \max_{Q(0),\cdots,Q(K-1)} \sum_{k,j} Q(k)P(j|k) log \frac{P(j|k)}{\sum_{i} Q(i)P(j|i)} \tag{3.4}$$

Notice that $I(X;Y)$ is a function of both the channel and the input assignment, while C is a function only of the channel.
The evaluation of C involves a maximization over K variables with following constraints: $Q(k) \geq 0$ and $\sum Q(k) = 1$. Since the function is continuous and the maximization is over a closed bounded region of vector space, the maximum value must exist.
The average mutual information for a DMC with independent identical distributed (i.i.d.) inputs has been proved to be the symmetric capacity as in [29, Sec. 4.5] and for a B-DMC can be written as in 3.5

$$I(W) = \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} \frac{1}{2} W(y|x) log \frac{W(y|x)}{\frac{1}{2}W(y|0) + \frac{1}{2}W(y|1)} \tag{3.5}$$

Another important information is given by the Bhattacharyya parameter of W,

$$Z(W) \triangleq \sum_{y \in \mathcal{Y}} \sqrt{W(y|0) + W(y|1)} \tag{3.6}$$

that is the upper bound on the probability of maximum-likelihood decision error when W is used only once to transmit 0 or 1. The Bhattacharyya measure has a

simple geometric interpretation as the cosine of the angle between the K-dimensional vectors $(\sqrt{W(y_1|0)}, \ldots, \sqrt{W(y_k|0)})$ and $(\sqrt{W(y_1|1)}, \ldots, \sqrt{W(y_k|1)})$.[30]
The parameter in 3.6 will be used instead of 3.5, to select the information set of good channels $W_N^{(i)}$. Intuitively the relation between $I(W)$ and $Z(W)$ are:

$$\begin{aligned} I(W) \approx 1 \Leftrightarrow Z(W) \approx 0 \\ I(W) \approx 0 \Leftrightarrow Z(W) \approx 1 \end{aligned} \tag{3.7}$$

Considering the cutoff symmetric channel it can be written as:

$$E_0(\rho, Q) = -log \sum_{j=0}^{J-1} \left[\sum_{k=0}^{K-1} Q(k) P(j|k)^{1/(1+\rho)} \right]^{1+\rho} \tag{3.8}$$

It is proven in [29, Sec. 5.6] that $I(W) \geq E_0(1, Q)$ so it can be rewritten as

$$\begin{aligned} E_0(1, Q) &= -log \sum_{j=0}^{J-1} \left[\sum_{k=0}^{K-1} Q(k) P(j|k)^{1/2} \right]^2 = \\ &= -log \sum_{j=0}^{J-1} \left[Q(0) P(j|0)^{\frac{1}{2}} + Q(1) P(j|1)^{\frac{1}{2}} \right]^2 \\ &= -log \sum_{j=0}^{J-1} \left[\frac{1}{2} \sqrt{P(j|0)} + \frac{1}{2} \sqrt{P(j|1)} \right]^2 \\ &= log \frac{1}{\sum_{j=0}^{J-1} \left[\frac{1}{4} P(j|0) + \frac{1}{4} P(j|1) + \frac{1}{2} \sqrt{P(j|0)} \sqrt{P(j|1)} \right]} \\ &= log \frac{2}{\sum_{j=0}^{J-1} \left[\frac{1}{2} P(j|0) + \frac{1}{2} P(j|1) \right] + \sum_{j=0}^{J-1} \left[\sqrt{P(j|0) P(j|1)} \right]} \\ &= log \frac{2}{1 + Z(W)} \end{aligned} \tag{3.9}$$

The information of 3.7 is then partially given by the following inequality.

$$I(W) \geq log \frac{2}{1 + Z(W)} \tag{3.10}$$

To impose an upper limit to $I(W)$ for a B-DMC, firstly define variation distance $d(W)$ as

$$d(W) \triangleq \frac{1}{2}\sum_{y\in\mathcal{Y}} |W(y|0) - W(y|1)| \tag{3.11}$$

Let us consider 3.5; it can be explicitly written for both inputs as

$$I(W) = \sum_{y\in\mathcal{Y}} \frac{1}{2}\left[W(y|0) log \frac{W(y|0)}{\frac{1}{2}W(y|0) + \frac{1}{2}W(y|1)} + W(y|1) log \frac{W(y|1)}{\frac{1}{2}W(y|0) + \frac{1}{2}W(y|1)} \right] \tag{3.12}$$

The ith term in brackets is given by

$$f(x) = x\ log\frac{x}{x+\delta} + (x+2\delta) log \frac{x+2\delta}{x+\delta} \tag{3.13}$$

where $x = min\{W(y|0), W(y|1)\}$ and $\delta = \frac{1}{2}|W(y|0) - W(y|1)|$. To maximize $f(x)$ over $0 \leq x \leq 1-2\delta$ it is computed:

$$\frac{df(x)}{dx} = log\left(\frac{x(x+2\delta)}{(x+\delta)^2}\right) = \frac{1}{2} log\left(\frac{\sqrt{x(x+2\delta)}}{(x+\delta)}\right) \tag{3.14}$$

It appears $\sqrt{x(x+2\delta)}$ the geometric mean and $(x+\delta)$ the arithmetic mean of x and $(x+2\delta)$, consequently $\frac{df}{dx} \leq 0$ and $f(x)$ is maximum when $x = 0$. In conclusion $f(x)|_{x=0} = 2\delta$ so $f(x) \leq 2\delta$ and substituting in 3.12 it follows:

$$I(W) \leq \sum_{y\in\mathcal{Y}} \frac{1}{2}|W(y|0) - W(y|1)| = d(W) \tag{3.15}$$

Let $R_i \triangleq (W(y_i|0) + W(y_i|1))/2$ and $\delta_i \triangleq \frac{1}{2}|W(y_i|0) - W(y_i|1)|$, then $Z(W)$ can be rewritten as:

$$Z(W) = \sum_{y_i\in\mathcal{Y}} \sqrt{(R_i+\delta_i)(R_i-\delta_i)} = \sum_{i=1}^{J} \sqrt{(R_i^2 - \delta_i^2)} \tag{3.16}$$

To carry out the maximization of 3.16 over δ_i subject to $0 \leq \delta_i \leq R_i$ and $i = 1, \ldots, J$, partial derivatives are computed:

$$\begin{aligned} \frac{dZ(W)}{d\delta_i} &= -\frac{\delta_i}{\sqrt{R_i^2 - \delta_i^2}} \\ \frac{d^2Z(W)}{d\delta_i^2} &= -\frac{R_i^2}{(R_i^2 - \delta_i^2)^{\frac{3}{2}}} \end{aligned} \tag{3.17}$$

Z(W) is therefore is decreasing over all its domain and it is a concave function of δ_i for each i. The maximum occurs at the solution of the set of i equations $\frac{dZ(W)}{d\delta_i} = k$, where k is a constant, in other words for $\delta_i = R_i\sqrt{\frac{k^2}{(1+k^2)}}$.

Impose that $d(W) = \sum_{i=1}^{J} \delta_i = \delta$ and notice the fact that $\sum_{i=1}^{J} R_i = 1$, we find $\sqrt{\frac{k^2}{(1+k^2)}} = \delta$.

So the maximum occurs at $\delta_i = \delta R_i$ and 3.16 has the value $\sum_{i=1}^{J} \sqrt{R_i^2 - \delta^2 R_i^2} = \sqrt{1-\delta^2}$. We have thus shown that $Z(W) \leq \sqrt{1-d(W)^2}$, which is equivalent to $d(W) \leq \sqrt{1-Z(W)^2}$.
In conclusion following inequality which relates $I(W)$ and $Z(W)$ can be written.

$$I(W) \leq d(W) \leq \sqrt{1-Z(W)^2} \tag{3.18}$$

An important channel considered to obtain specific results (later discussed) for polar codes theory is the B-DMC called binary erasure channel (BEC). As represented in

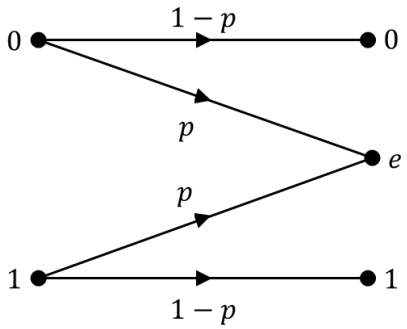


Figure 3.2: *Binary Erasure Channel (BEC).*

Figure 3.2 this channel has binary input $\mathcal{X} = \{0,1\}$ and ternary outputs $\mathcal{Y} = \{0,1,e\}$, where the inputs are received unaltered with fixed probability $1-p$ otherwise inputs are completely lost (erased) with probability p, so the symbol e is received with probability p, called *erasure probability*.
Another important definition is the *Kronecker product* of a matrix $A = [A_{i,j}]$ of

dimension mxn and matrix $B = [B_{i,j}]$ of dimension rxs as:

$$A \otimes B = \begin{bmatrix} A_{1,1}B & \cdots & A_{1,n}B \\ \vdots & \ddots & \vdots \\ A_{m,1}B & \cdots & A_{m,n}B \end{bmatrix} \tag{3.19}$$

which is an mrxns matrix. Starting from this definition it can be given the *Kronecker power* $A^{\otimes n}$ which is defined with a recursion for all $n \geq 1$ as:

$$A^{\otimes n} = A \otimes A^{\otimes(n-1)} \tag{3.20}$$

It is also given by convention that $A^{\otimes 0} \triangleq [1]$.

3.2 Channel polarization effect

The channel polarization effect described in [23] is given by the creation of a vector of N synthetic channels $\{W_N^{(i)} : 1 \leq i \leq N\}$ from N independent copies of a given B-DMC channel that causes as N becomes large, the symmetric capacity terms $\{I(W_N^{(i)})\}$ tend towards 0 or 1 for all channels, with exception of a vanishing fraction of indices i. This effect can be achieved through channel splitting and channel combining operations based on [31].

3.2.1 Channel combining

Starting from identical copies of a given B-DMC channel W, with a recursive method is produced the synthetic channel vector $W_N : \mathcal{X}^N \to \mathcal{Y}^N$, where $N = 2^n$, $n \geq 0$. For $n = 0$ the synthetic channel vector equals the given channel $W_1 \triangleq W$, so no

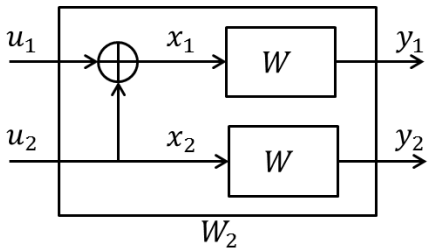


Figure 3.3: W_2 *channel.*

modification are introduced. For the second step of recursion ($n = 1$) two copies of W_1 are used to obtain channel $W_2 : \mathcal{X}^2 \to \mathcal{Y}^2$ with the transition probability

$$W_2(y_1, y_2|u_1, u_2) = W(y_1|u_1 \oplus u_2)W(y_2|u_2) \tag{3.21}$$

In Figure 3.3 is given a graphical representation of how two channels are combined to obtain the synthetic channel vector for $n = 1$. The recursion then follows by using the new synthetic channels and combine again in the same way of the previous step, so since W_2 has two channels, two copy of it will be used to generate $W_4 : \mathcal{X}^4 \to \mathcal{Y}^4$ with transition probability

$$W(y_1^4|u_1^4) = W_2(y_1^2|u_1 \oplus u_2, u_3 \oplus u_4)W_2(y_3^4|u_2, u_4) \tag{3.22}$$

where it is used the general notation for row vectors a_i^j as $(a_1, \ldots, a_N)$ with $1 \leq i$, $j \leq N$. If $j < i$, a_i^j is considered void. In Figure 3.4 graphical representation

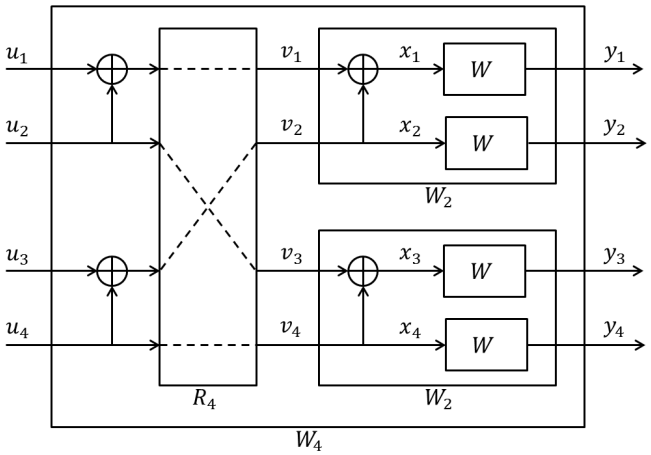


Figure 3.4: W_4 *channel obtained by recursions of* W_2 *and* W.

of interconnection for channel W_4 is shown, it is also pointed out the permutation operation R_4 that maps an input $s_1^4 = (s_1, s_2, s_3, s_4)$ to $v_1^4 = (s_1, s_3, s_2, s_4)$. The generating matrix that maps $u_1^4 \mapsto x_1^4$ form the inputs of W_4 to the inputs of every W (this set will be called W^4) can be written as

$$x_1^4 = u_1^4 G_4 = u_1^4 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.23}$$

So it is generated the relation between the transition probabilities of W_4 and those of W^4.

$$W_4(y_1^4|u_1^4) = W^4(y_1^4|u_1^4 G_4) \tag{3.24}$$

The recursion for creating the channel vector is generalized in Figure 3.5 where two

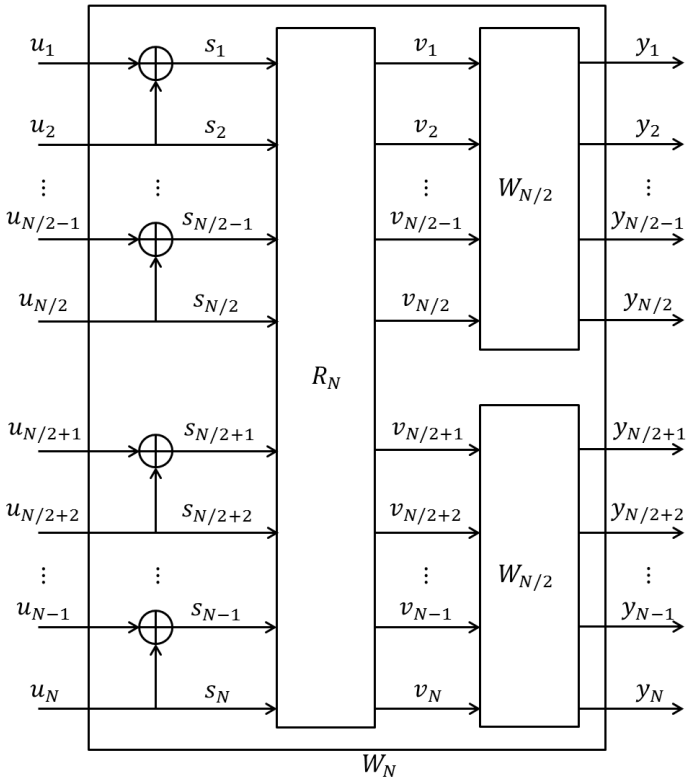


Figure 3.5: *Generalized W_N channel obtained by recursion of two $W_{\frac{N}{2}}$.*

independent copies of $W_{\frac{N}{2}}$ are combined to produce the channel W_N. The input vector u_1^N is transformed according to:

$$\begin{cases} s_{2i-1} &= u_{2i-1} \oplus u_{2i} \\ s_{2i} &= u_{2i} \end{cases} \tag{3.25}$$

for $1 \leq i \leq \frac{N}{2}$. The operator R_N in figure is a *reverse shuffle* permutation operation over input s_1^N to produce $v_1^N = (s_1, s_3, \ldots, s_N - 1, s_2, s_4, \ldots, s_N)$. v_1^N is the input of the two copies of $W_{\frac{N}{2}}$.

The overall mapping $u_1^N \mapsto x_1^N$ from inputs to the inputs of raw channels W^N, is linear thanks the fact that every mapping $u_1^N \mapsto v_1^N$ in the recursion is linear over $GF(2)$. So the final mapping can be represented by the *generator matrix* G_N of size NxN, where

$$x_1^N = u_1^N G_N \tag{3.26}$$

The transition probabilities of the two channels W_N and W^N are related by

$$W_N(y_1^N|u_1^N) = W^N(y_1^N|u_1^N G_N) \tag{3.27}$$

for every $y_1^N \in \mathcal{Y}^N,\ u_1^N \in \mathcal{X}^N$.
In Section 3.3 is shown that $G_N = B_N F^{\otimes n}$ where $N = 2^n,\ n \geq 0,\ B_N$ is the $bit-reversal$ permutation matrix and $F \triangleq \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$.
As presented, the channel combining operation is fully specified by F, and G_N and $F^{\otimes n}$ differ only by the (bit-reversed) order of rows.

3.2.2 Channel splitting

The process of splitting the vector channel W_N into N binary-input independent copies of channel W is called *channel splitting*. Formally $W_N^{(i)} : \mathcal{X} \to \mathcal{Y}^N \mathrm{x} \mathcal{X}^{i-1}$, for $1 \leq i \leq N$, defined by the transition probabilities

$$W_N^{(i)}(y_1^N, u_1^{i-1}|u_i) \triangleq \sum_{u_{i+1}^N \in \mathcal{X}^{N-i}} \frac{1}{2^{N-1}} W_N(y_1^N|u_1^N) \tag{3.28}$$

where u_i is the given input and (y_1^N, u_1^{i-1}) is the output of $W_N^{(i)}$. To have an intuitive understanding of the channels $\{W_N^{(i)}\}$, consider to use a genie-aided decoder in which the i-th decision of u_i is taken after observing y_i^N and previous channel inputs u_1^{i-1} supplied correctly by the genie regardless of any decision errors at early stages. If u_1^N is *a-priori* uniform on $\mathcal{X}^N$, then $W_N^{(i)}$ is the effective channel seen by the ith decision element.
For example in the case of synthetic channel vector W_2, it can be split in $W_2^{(1)}$ and $W_2^{(2)}$, the transition probabilities of $W_2^{(1)} : \mathcal{X} \to \mathcal{Y}^2$ and $W_2^{(2)} : \mathcal{X} \to \mathcal{Y}^2 \mathrm{x} \mathcal{X}$ can be written as:

$$\begin{aligned} W_2^{(1)}(y_1^2|u_1) &= \tfrac{1}{2} \sum_{u_2} W_2(y_1^2|u_1^2) = \tfrac{1}{2} \sum_{u_2} W(y_1|u_1 \oplus u_2) W(y_2|u_2) \\ W_2^{(2)}(y_1^2, u_1|u_2) &= \tfrac{1}{2} W_2(y_1^2|u_1^2) = \tfrac{1}{2} W(y_1|u_1 \oplus u_2) W(y_2|u_2) \end{aligned} \tag{3.29}$$

where 3.25 channel combining information has been used. For the $\{W_N^{(i)}\}$ channels the Bhattacharyya parameter 3.6 becomes

$$Z(W_N^{(i)}) = \sum_{y_1^N \in \mathcal{Y}^N} \sum_{u_1^{i-1} \in \mathcal{X}^{i-1}} \sqrt{W_N^{(i)}(y_1^N, u_1^{i-1}|0) W_N^{(i)}(y_1^N, u_1^{i-1}|1)} \tag{3.30}$$

3.2.3 Operations on recursive synthetic channels

The goal of this section is to show that the operations 3.27 and 3.28 obtained for entire synthetic channel $W_N^{(i)}$ originates from recursive single-step transformation of same operation on channels which created it.

Considering a pair of binary-input channels $W' : \mathcal{X} \to \tilde{\mathcal{Y}}$ and $W'' : \mathcal{X} \to \tilde{\mathcal{Y}}\mathrm{x}\mathcal{X}$ are obtained by a single step-step transform of two independent copies of a binary-input channel $W : \mathcal{X} \to \mathcal{Y}$ if exist a one-to-one mapping $f : \mathcal{Y}^2 \to \mathcal{Y}$ such that

$$\begin{aligned} W'(f(y_1, y_2)|u_1) &= \sum_{u_2'} \tfrac{1}{2} W(y_1|u_1 \oplus u_2') W(y_2|u_2') \\ W''(f(y_1, y_2), u_1|u_2) &= \tfrac{1}{2} W(y_1|u_1 \oplus u_2) W(y_2|u_2) \end{aligned} \tag{3.31}$$

for every $u_1, u_2 \in \mathcal{X}, y_1, y_2 \in \mathcal{Y}$. If it is true we can write:

$$(W, W) \mapsto (W', W'') \tag{3.32}$$

Let us recall 3.28 for $2N$ channels and $1 \leq i \leq N$, we write

$$\begin{aligned} W_{2N}^{(2i-1)}(y_1^{2N}, u_1^{2i-2}|u_{2i-1}) &= \sum_{u_{2i}^{2N}} \frac{1}{2^{2N-1}} W_{2N}(y_1^{2N}|u_1^{2N}) \\ W_{2N}^{(2i)}(y_1^{2N}, u_1^{2i-1}|u_{2i}) &= \sum_{u_{2i+1}^{2N}} \frac{1}{2^{2N-1}} W_{2N}(y_1^{2N}|u_1^{2N}) \end{aligned} \tag{3.33}$$

so considering the first equation and introducing subscript o and e on a vector that respectively represent a subvector of the initial one composed by elements with only odd or even indices, follows:

$$\begin{aligned} &W_{2N}^{(2i-1)}(y_1^{2N}, u_1^{2i-2}|u_{2i-1}) = \sum_{u_{2i,o}^{2N}, u_{2i,e}^{2N}} \frac{1}{2^{2N-1}} W_{2N}(y_1^{2N}|u_1^{2N}) = \\ &= \sum_{u_{2i,o}^{2N}, u_{2i,e}^{2N}} \frac{1}{2^{2N-1}} W_N(y_1^N|u_{1,o}^{2N} \oplus u_{1,e}^{2N}) W_N(y_{N+1}^{2N}|u_{1,e}^{2N}) = \\ &= \sum_{u_{2i}} \frac{1}{2} \sum_{u_{2i+1,e}^{2N}} \frac{1}{2^{N-1}} W_N(y_{N+1}^{2N}|u_{1,e}^{2N}) \sum_{u_{2i+1,o}^{2N}} \frac{1}{2^{N-1}} W_N(y_1^N|u_{1,o}^{2N} \oplus u_{1,e}^{2N}) = \\ &= \sum_{u_{2i}} \frac{1}{2} \sum_{u_{2i+1,e}^{2N}} \frac{1}{2^{N-1}} W_N(y_{N+1}^{2N}|u_{1,e}^{2N}) W_N^{(i)}(y_1^N, u_{1,o}^{2N} \oplus u_{1,e}^{2N}|u_{2i-1} \oplus u_{2i}) = \\ &= \frac{1}{2} \sum_{u_{2i}} W_N^{(i)}(y_{N+1}^{2N}, u_{1,e}^{2i-2}|u_{2i,e}) W_N^{(i)}(y_1^N, u_{1,o}^{2i-2} \oplus u_{1,e}^{2i-2}|u_{2i-1} \oplus u_{2i}) \end{aligned} \tag{3.34}$$

because, as $u_{2i+1,o}^{2N}$ rages over $\mathcal{X}^{N-i}$, $u_{2i+1,o}^{2N} \oplus u_{2i+1,e}^{2N}$ ranges also over $\mathcal{X}^{N-i}$. Similarly for second equation of 3.31 we can write

$$\begin{aligned} &W_{2N}^{(2i)}(y_1^{2N}, u_1^{2i-1}|u_{2i}) = \sum_{u_{2i+1}^{2N}} \frac{1}{2^{2N-1}} W_{2N}(y_1^{2N}|u_1^{2N}) = \\ &= \tfrac{1}{2} \sum_{u_{2i+1,e}^{2N}} \frac{1}{2^{N-1}} W_N(y_{N+1}^{2N}|u_{1,e}^{2N}) \sum_{u_{2i+1}^{2N}} \frac{1}{2^{N-1}} W_N(y_1^N|u_{1,o}^{2N} \oplus u_{1,e}^{2N}) = \\ &= \tfrac{1}{2} W_N^{(i)}(y_1^N, u_{1,o}^{2i-2} \oplus u_{1,e}^{2i-2}|u_{2i-1} \oplus u_{2i}) W_N^{(i)}(y_{N+1}^{2N}, u_{1,e}^{2i-2}|u_{2i}) \end{aligned} \tag{3.35}$$

Considering following substitutions in 3.34 and 3.35 is possible to prove that these equations are equal to 3.31.

$$\begin{aligned}
&W_N^{(i)} \to W, \\
&W_{2N}^{(2i-1)} \to W', \\
&W_{2N}^{(2i)} \to W'', \\
&u_{2i-1} \to u_1, \\
&u_{2i} \to u_2, \\
&(y_1^N, u_{1,o}^{2i-2} \oplus u_{1,e}^{2i-2}) \to y_1, \\
&(y_N^{2N}, u_{1,e}^{2i-2}) \to y_2, \\
&(y_1^{2N}, u_1^{2i-2}) \to f(y_1, y_2).
\end{aligned} \tag{3.36}$$

So the recursive mapping used can be written in general form as

$$(W_N^{(i)}, W_N^{(i)}) \mapsto (W_{2N}^{(2i-1)}, W_{2N}^{(2i)}) \tag{3.37}$$

This shows that channel transformation from W_N to $(W_N^{(1)}, \ldots, W_N^{(N)})$ can be broken into single-step channel transformation.

3.2.4 Channel polarization

In the previous sections, using the channel combining and splitting operation, we have created N new channels $W_N^{(i)}$. Let us consider the core recursion for two identical channels and to use a simplified case of example this can be thought to be equal to the case $n = 1$ recursion which generates 3.29. In section 3.2.3 the validity of this approach has been proven.
This result can be exploited to evaluate how rate $I(W_N^{(i)})$ and reliability $Z(W_N^{(i)})$ parameters in function of the single-step transformation.
Suppose $(W, W) \mapsto (W^-, W^+)$, where $W : \mathcal{X} \to \mathcal{Y}$, $W^- : \mathcal{X} \to \tilde{\mathcal{Y}}$, $W^+ : \mathcal{X} \to \tilde{\mathcal{Y}}\mathrm{x}\mathcal{X}$ and there is a one-to-one function $f : \mathcal{Y} \to \tilde{\mathcal{Y}}$ such that 3.31 are verified. Let us

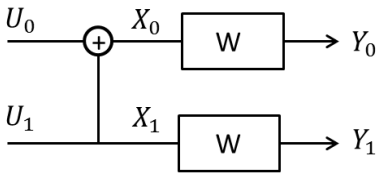


Figure 3.6: *Channel variables relationship.*

consider the pairs of random variables (U_0, U_1) uniformly distributed over $\mathcal{X}^2$, they

generate $(X_0, X_1) = (U_0 \oplus U_1, U_1)$ (Figure 3.6). The transition probability becomes $P_{Y_0,Y_1|X_0,X_1} = W(y_0|x_0)W(y_1|x_1)$, and define $\tilde{Y} = f(Y_0, Y_1)$. It is possible to write

$$\begin{aligned} W^-(\tilde{y}|u_0) &= P_{\tilde{Y}|U_0}(\tilde{y}|u_0) \\ W^+(\tilde{y}, u_0|u_1) &= P_{\tilde{Y}U_0|U_1}(\tilde{y}, u_0|u_1) \end{aligned} \tag{3.38}$$

Considering also the fact that $(Y_0, Y_1) \mapsto \tilde{Y}$ is invertible, we get

$$\begin{aligned} I(W^-) &= I(U_0; \tilde{Y}) = I(U_0; Y_0Y_1) \\ I(W^+) &= I(U_1; \tilde{Y}U_0) = I(U_1; Y_0Y_1U_0) \end{aligned} \tag{3.39}$$

Since U_0 and U_1 are independent, $I(U_1; Y_0Y_1U_0) = I(U_1; Y_0Y_1|U_0)$, using the chain rule of mutual information is possible to write

$$\begin{aligned} I(W^-) + I(W^+) &= I(U_0; Y_0Y_1) + I(U_1; Y_0Y_1U_0) = \\ &= I(U_0; Y_0Y_1) + I(U_1; Y_0Y_1|U_0) = \\ &= I(U_0U_1; Y_0Y_1) = \\ &= I(X_0X_1; Y_0Y_1) \end{aligned} \tag{3.40}$$

where the one-to-one relation between (X_0, X_1) and (U_0, U_1) has been used. From Figure 3.6 can be observed that

$$I(X_0X_1; Y_0Y_1) = I(X_0; Y_0) + I(X_1; Y_1) = 2I(W) \tag{3.41}$$

Starting from this result is possible to prove $I(W^+) \geq I(W)$ by noticing

$$\begin{aligned} I(W^+) &= I(U_1; Y_0Y_1U_0) = \\ &= I(U_1; Y_1) + I(U_1; Y_0U_0|Y_1) = \\ &= I(W) + I(U_1; Y_0U_0|Y_1) \end{aligned} \tag{3.42}$$

In conclusion is possible to write

$$\begin{aligned} I(W^-) + I(W^+) &= 2I(W) \\ I(W^-) \leq I(W) &\leq I(W^+) \end{aligned} \tag{3.43}$$

The first equality in 3.43 means that the single-step channel transform preserves the symmetric capacity. The second inequality becomes $I(W^-) = I(W) = I(W^+)$ if and only if W is a perfect noiseless channel ($(W) = 1$) or a completely noisy channel ($I(W) = 0$), otherwise the single-step transform moves symmetric capacity as $I(W^-) < I(W) < I(W^+)$, thus generating polarization. The last sentence is proven by studying when $I(U_1; Y_0U_0|Y_1) = 0$, this can be rewritten equivalently as

$$P_{U_0,U_1,Y_0|Y_1}(u_0, u_1, y_0|y_1) = P_{U_0,Y_0|Y_1}(u_0, y_0|y_1)P_{U_1|Y_1}(u_1|y_1) \tag{3.44}$$

for all (u_0, u_1, y_0, y_1) such that $P_{Y_1}(y_1) > 0$, or equivalently

$$P_{Y_0,Y_1|U_0,U_1}(y_0, y_1|u_0, u_1)P_{Y_1}(y_1) = P_{Y_0,Y_1|U_0}(y_0, y_1|u_0)P_{Y_1|U_1}(y_1|u_1) \tag{3.45}$$

for all (u_0, u_1, y_0, y_1). Since $P_{Y_0,Y_1|U_0,U_1}(y_0, y_1|u_0, u_1) = W(y_0|u_0 \oplus u_1)W(y_1|u_1)$, it is possible to write 3.45 as

$$W(y_1|u_1)[W(y_0|u_0 \oplus u_1)P_{Y_1}(y_1) - P_{Y_0,Y_1|U_0}(y_0, y_1|u_0)] = 0 \tag{3.46}$$

By construction $P_{Y_1}(y_1) = \frac{1}{2}W(y_1|u_1) + \frac{1}{2}W(y_1|u_1 \oplus 1)$ and $P_{Y_0,Y_1|U_0}(y_0, y_1|u_0) = \frac{1}{2}W(y_0|u_0 \oplus u_1)W(y_1|u_1) + \frac{1}{2}W(y_0|u_0 \oplus u_1 \oplus 1)W(y_1|u_1 \oplus 1)$, simplifying is possible to obtain:

$$W(y_1|u_1)W(y_1|u_1 \oplus 1)[W(y_0|u_0 \oplus u_1) - W(y_0|u_0 \oplus u_1 \oplus 1)] = 0 \tag{3.47}$$

choosing $(u_0, u_1) = (0,0)$, (but is equal for all four realizations) 3.48 becomes:

$$W(y_1|0)W(y_1|1)[W(y_0|0) - W(y_0|1)] = 0 \tag{3.48}$$

so if $W(y_0|0) = W(y_0|1)$ implies $I(W) = 0$ or exist no y_1 such that $W(y_1|0)W(y_1|1) > 0$ which means $I(W) = 1$.

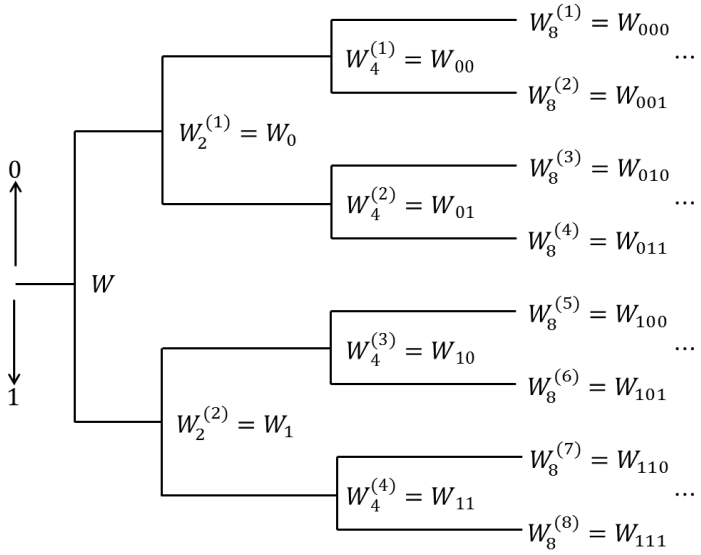


Figure 3.7: *Binary tree for the recursive construction of synthetic channels.*

In Figure 3.7 is shown the process of recursive construction of channels by a binary tree graph. Starting from the root node which is associated to channel W, two children channels $W_2^{(1)}$ and $W_2^{(2)}$ are generated. According to previous result

3.43, the second channel has better transmission property. This better channel is then used in the recursion to produce other two channels ($W_4^{(3)}$ and $W_4^{(4)}$), which again will be subject to 3.43 and so it will generate another better channel in terms of capacity than the originating one and so on. This concept can be applied to all generated channels both good and bad.
The final step is to proof that the recursion produces overall channels which are more and more noiseless or noisy with the following theorem.

Theorem

For any B-DMC W, the channels $\{W_N^{(i)}\}$ polarize in the sense that, for any fixed $\delta \in (0,1)$, as N goes to infinity through powers of two, the fraction of indices $i \in \{1,\dots,N\}$ for which $I(W_N^{(i)}) \in (1-\delta,1]$ goes to $I(W)$ and the fraction for which $I(W_N^{(i)}) \in [0,\delta)$ goes to $1-I(W)$.

Proof

Let us consider the stochastic convergence property of the random sequence $\{I_n\}$, where $I_n = I(H_n)$ is a random process obtained by the random channel process $K_n = W_{b_1\dots,b_n}$.
$\{b_1\dots,b_n\}$ is a binary label of a channel $W_{2^n}^{(i)}$ (as also reported in Figure 3.7).
Calling Ω the space of all binary sequences $(b_1\dots,b_n) \in 0,1^\infty$, $\mathfrak{F}$ the Borel field (BF) generated by the cylindrical set of coordinates $S(b_1\dots,b_n) \triangleq \{\omega_1^n \in \Omega : \omega_1 = b_1,\dots,\omega_n = b_n\}$. So is possible to define $\mathfrak{F}_0$ as the trivial BF of the null set, from which follows $\mathfrak{F}_0 \subset \mathfrak{F}_1 \subset \cdots \mathfrak{F}_n$.
The sequence of random variable and BF $\{I_n, \mathfrak{F}_n; n \geq 0\}$ is a martingale if:

1. $\mathfrak{F}_n \subset \mathfrak{F}_{n+1}$ and I_n is $\mathfrak{F}_n$ measurable
2. $E\{|I_n|\} < \infty$
3. $I_n = E\{I_{n+1}|\mathfrak{F}_n\}$

The first condition is true by construction. The second statement is verified considering the fact that $0 \leq I_n \leq 1$. The last condition is verified by writing

$$E\{I_{n+1}|S(b_1,\dots,b_n)\} = \frac{1}{2}I(W_{b_1,\dots,b_{n-1},0}) + \frac{1}{2}I(W_{b_1,\dots,b_{n-1},1}) = I(W_{b_1,\dots,b_n}) \quad (3.49)$$

where the first equation of 3.43 is used and $I(W_{b_1,\dots,b_n})$ is the value of I_n on $S(b_1,\dots,b_n)$. So it follows that the sequence $\{I_n; n \geq 0\}$ converges almost everywhere to a random variable I_∞, such that $E\{I_\infty\} = I_0$ by the general convergence results about uniformly integrable martingales.

The limit of RV I_∞ so will take values in $\{0,1\}$ thanks to the transformation $(W, W) \mapsto (W_2^-, W_2^+)$ as determined by the equality condition of second equation in 3.43.

In Figure 3.8 is shown the polarization effect for a BEC channel at the increasing

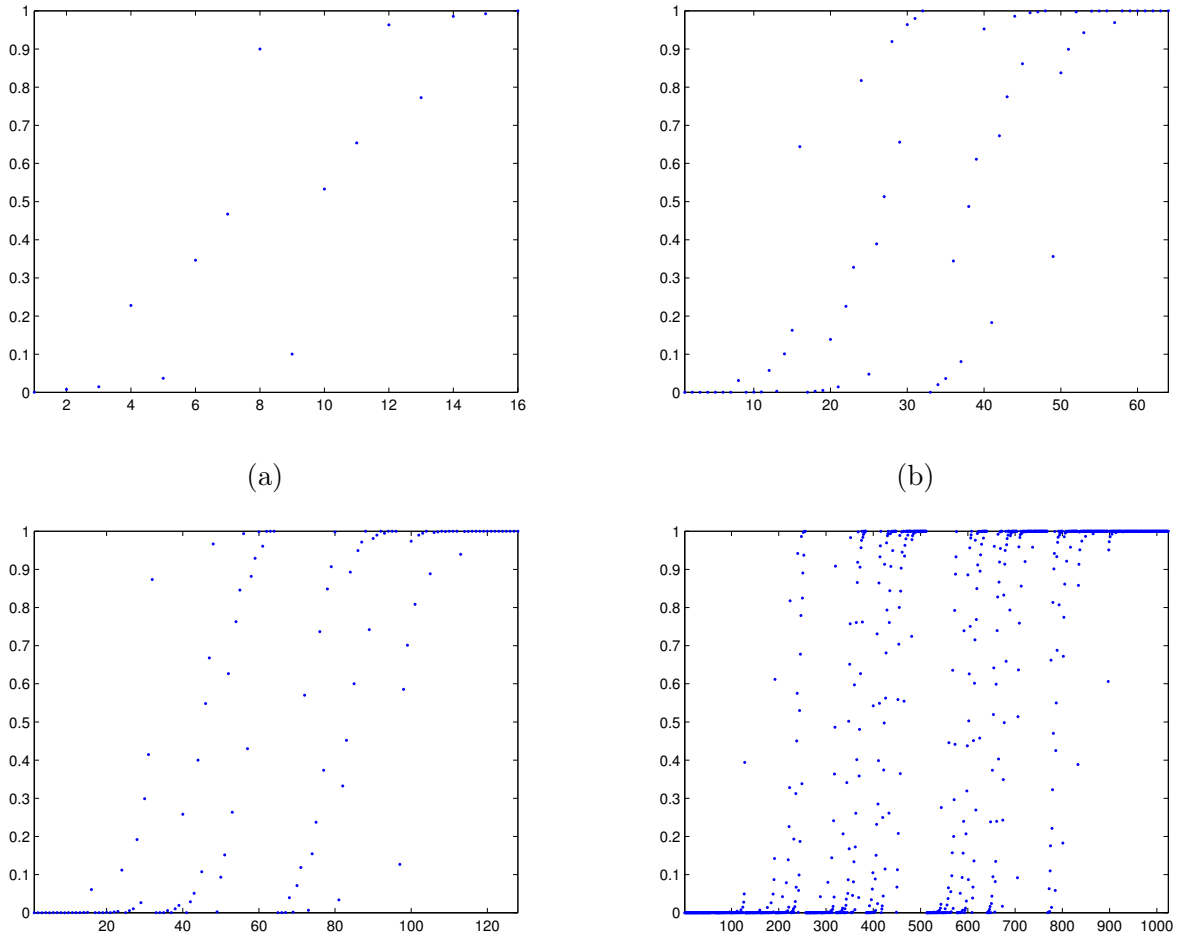


Figure 3.8: Symmetric capacity $I(W_N^{(i)})$ for N BEC identical channels with erasure probability $\epsilon = 0.5$. a)$N = 16$ b)$N = 64$ c)$N = 128$ d)$N = 1024$.

of codeword length N, in particular it can be noticed that $I(W_N^{(i)})$ tends to be near 0 for small i and near 1 for $i \to N$.

3.3 Encoding and decoding of Polar Codes

The polarization effect can be exploited to construct codes that achieve the symmetric channel capacity $I(W)$ by a method that is called *polar coding*. The fundamental idea of polar coding is to synthesize, out of N independent identical copies of a given B-DMC channel, a second set of N binary-input channels $W_N^{(i)}$ which can be individually accessed and send data only through those for which $Z(W_N^{(i)}) \simeq 0$ to achieve $I(W_N^{(i)}) \simeq 1$ i.e. almost noiseless channel.
The general linear block code is expressed by

$$x_1^N = u_1^N G_N \tag{3.50}$$

where G_N is the generator matrix of order N as defined in 3.26. Considering a subset $\mathcal{A} \subset \{1, \cdots, N\}$ is possible to rewrite it as

$$x_1^N = u_{\mathcal{A}} G_N(\mathcal{A}) \oplus u_{\mathcal{A}^c} G_N(\mathcal{A}^c) \tag{3.51}$$

where $G_N(\mathcal{A})$ is the submatrix of G_N created by the rows with index in $\mathcal{A}$. The set $\mathcal{A}$ is called the $information\ set$ and $u_{\mathcal{A}^c} \in \mathcal{X}^{N-K}$ is the $frozen\ bit$ vector. A mapping from the source blocks $u_{\mathcal{A}}$ to the codeword blocks x_1^N is obtained by fixing $\mathcal{A}$ and $u_{\mathcal{A}^c}$ and leaving $u_{\mathcal{A}}$ as a free variable. This class of codes is called $G_N - coset\ codes$, identified by a vector $(N, K, \mathcal{A}, u_{\mathcal{A}^c})$ where N is the size of original G_N matrix, K is the size of $\mathcal{A}$, so K/N is the *code rate*.
The following example shows the encoder mapping for a code (4,2, {2,4}, (0,0))

$$\begin{aligned} x_1^4 &= u_1^4 G_4 = \\ &= (u_2, u_4) \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} + (0,0) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \end{aligned} \tag{3.52}$$

So given a realization of the source block $(u_2, u_4) = (1,1)$, the coded block becomes $x_1^4 = (0,1,0,1)$. Therefore in polar codes is of particular importance the choice of the frozen bits to achieve best coding performance as previously discussed.
The initial set of data u_1^N , composed by information and frozen bits, is encoded into codeword x_1^N, the coded information is sent over W_N vector channel and it is received the channel output y_1^N. The decoder goal is to correctly estimate $\hat{u}_1^N$ of u_1^N, given the knowledge of y_1^N, $\mathcal{A}$ and $u_{\mathcal{A}^c}$. So the decoder only needs to estimate $\hat{u}_{\mathcal{A}}$ of $u_{\mathcal{A}}$ because the decoding of frozen information is already known.
First proposed decoding method for decoding polar codes was *Successive Cancellation* (SC), where given any $(N, K, \mathcal{A}, u_{\mathcal{A}^c})$ code, decision of $\hat{u}_1^N$ are taken by computing:

$$\hat{u}_i \triangleq \begin{cases} u_i, & \text{if } i \in \mathcal{A}^c \\ h_i(y_1^N, \hat{u}_1^{i-1}), & \text{if } i \in \mathcal{A} \end{cases} \tag{3.53}$$

where i goes in crescent order from 1 to N and the *decision functions* $h_i : \mathcal{Y}^N \mathrm{x} \mathcal{X}^{i-1} \rightarrow \mathcal{X}, i \in \mathcal{A}$ are defined as:

$$h_i \triangleq \begin{cases} 0, & \text{if } \dfrac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|0)}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|1)} \geq 1 \\ 1, & \text{otherwise} \end{cases} \tag{3.54}$$

for all $y_1^N \in \mathcal{Y}^N, \hat{u}_1^{i-1} \in \mathcal{X}^{i-1}$. A *block error* occurs if $\hat{u}_1^N \neq u_1^N$ or equivalently $\hat{u}_\mathcal{A} \neq u_\mathcal{A}$. Decision functions $\{h_i\}$ can be efficiently computed using recursive formulas, but as drawback they are not maximum-likelihood (ML) decision because "future" frozen bits $(u_j : j > i, j \in \mathcal{A}^c)$ are considered random variables instead of known bits. The exact computation of a-posteriori probability with all a-priori information is performed by belief propagation algorithm, so better decoding performance are expected at the cost of the increase of computational complexity.

3.4 Belief propagation in polar codes

Polar codes are similar to Reed-Muller codes as discussed in [32], so they can be decoded using Belief Propagation on Forney-style factor graphs, as described in [33] for Reed-Muller codes. An implementation for polar codes was conceptually presented in [34]. It was based on the direct mapping of the factor graph representation (as in Figure 3.9 for codeword length $N = 8$) into a fully-parallel implementation architecture. Each node of the factor graph can be identified by a couple of integers (i, j), $1 \leq i \leq n + 1$, $1 \leq j \leq N$ where the polar code length is $N = 2^n$, so the factor graph contains $(n + 1)$ levels. The rightmost nodes $(n + 1, j)$ are associated with codeword x, that is the channel output y if considered for the decoding scheme with noise of the channel, while leftmost nodes $(1, j)$ elaborate source word u, that are searched information.

General BP algorithm allows to determine codeword $\widehat{u}$, by implementing 3.55, that is the ratio of the a posteriori probabilities of having received a 0 rather than a 1 in position j and evaluated as in 2.13.

$$\frac{P(x_j = 0|y)}{P(x_j = 1|y)} = \frac{P(y_j|x_j = 0)}{P(y_j|x_j = 1)} \frac{P(x_j = 0)}{P(x_j = 1)} \tag{3.55}$$

Value $\widehat{u}$ is determined bit per bit in the following way:

$$\begin{aligned} \widehat{x_j} = 0 \text{ if } P(x_j = 0|y) > P(x_j = 1|y) \Rightarrow \frac{P(x_j = 0|y)}{P(x_j = 1|y)} > 1 \\ \widehat{x_j} = 1 \text{ if } P(x_j = 0|y) < P(x_j = 1|y) \Rightarrow \frac{P(x_j = 0|y)}{P(x_j = 1|y)} < 1 \end{aligned} \tag{3.56}$$

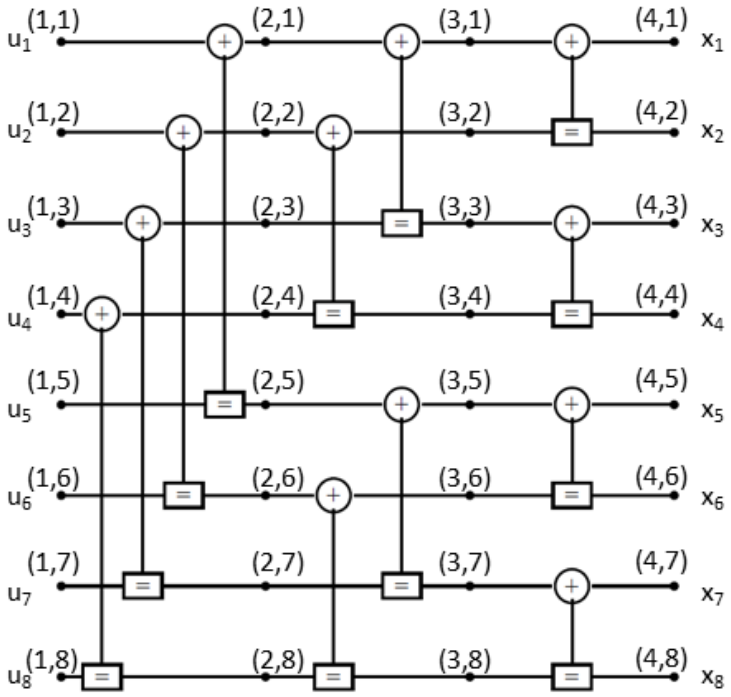


Figure 3.9: *Forney-style graph representation of a polar code of length eight.*

The final probability on $\widehat{x_j}$ used in 3.56 can be computed by evaluating probabilities associated to each node (i, j) of the propagation graph, exploiting the recursive structure of polar codes already discussed. So it is possible to study the single BP computational block composed the coding part plus the channels which may represent either external or synthetic channels.

Messages inside each block are called left-propagating or right propagating accord-

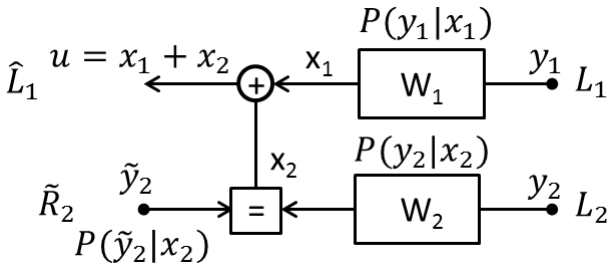


Figure 3.10: *Basic computational block for BP polar code decoding. Pointed out messages for the evaluation of* $\widehat{L_1}$ *information.*

ing to the direction which they cross the block. In Figure 3.10 is presented the case for the evaluation of upper left-going probability ratio, that will be called $\hat{L}_1$, given the information available on all possible source of data (left and right propagating messages). It is possible to rewrite left hand part of 3.55 as:

$$\begin{aligned}\widehat{L_1} &= \frac{P(u=0|y_1,y_2,\widetilde{y_2})}{P(u=1|y_1,y_2,\widetilde{y_2})} = \\ &= \frac{P_{x_1,x_2,y_1,y_2,\widetilde{y_2}}(x_1,x_1\oplus 0,y_1,y_2,\widetilde{y_2})}{P_{x_1,x_2,y_1,y_2,\widetilde{y_2}}(x_1,x_1\oplus 1,y_1,y_2,\widetilde{y_2})}\end{aligned}$$

where has been introduced support variables x_1, x_2. Imposing the coding property $u = x_1 \oplus x_2$, and by setting u equal to the correct value in the numerator or denominator, x_2 only depends by the value of x_1. So using Bayes theorem as right hand part of 3.55 follows:

$$\begin{aligned}\widehat{L_1} &= \frac{\sum_{x_1=0}^{1} P(x_1)\cdot P(x_2)\cdot P(y_2|x_2)\cdot P(\widetilde{y_2}|x_2)\cdot P(y_1|x_1)}{\sum_{x_1=0}^{1} P(x_1)\cdot P(x_2)\cdot P(y_2|x_2)\cdot P(\widetilde{y_2}|x_2)\cdot P(y_1|x_1)} = \\ &= \frac{\sum_{x_1=0}^{1} P(x_1)\cdot P(x_1)\cdot P(y_2|x_1)\cdot P(\widetilde{y_2}|x_1)\cdot P(y_1|x_1)}{\sum_{x_1=0}^{1} P(x_1)\cdot P(x_1\oplus 1)\cdot P(y_2|x_1\oplus 1)\cdot P(\widetilde{y_2}|x_1\oplus 1)\cdot P(y_1|x_1)}\end{aligned}$$

Then explicitly writing last equation, supposing equiprobable information data x_1 and x_2 and performing some algebraic operations as reported, it is possible to obtain result 3.57.

$$\begin{aligned}\widehat{L_1} &= \frac{P_{x_1}(0)P_{x_2}(0)P(y_2|0)P(\widetilde{y_2}|0)P(y_1|0)+P_{x_1}(1)P_{x_2}(1)P(y_2|1)P(\widetilde{y_2}|1)P(y_1|1)}{P_{x_1}(0)P_{x_2}(1)P(y_2|1)P(\widetilde{y_2}|1)P(y_1|0)+P_{x_1}(1)P_{x_2}(0)P(y_2|0)P(\widetilde{y_2}|0)P(y_1|1)} = \\ &= \frac{\dfrac{P_{x_1}(0)}{P_{x_1}(1)}\dfrac{P_{x_2}(0)}{P_{x_2}(1)}\dfrac{P(y_2|0)}{P(y_2|1)}\dfrac{P(\widetilde{y_2}|0)}{P(\widetilde{y_2}|1)}\dfrac{P(y_1|0)}{P(y_1|1)}+1}{\dfrac{P_{x_1}(0)}{P_{x_1}(1)}\dfrac{P_{x_2}(1)}{P_{x_2}(1)}\dfrac{P(y_2|1)}{P(y_2|1)}\dfrac{P(\widetilde{y_2}|1)}{P(\widetilde{y_2}|1)}\dfrac{P(y_1|0)}{P(y_1|1)}+\dfrac{P_{x_1}(1)}{P_{x_1}(1)}\dfrac{P_{x_2}(0)}{P_{x_2}(1)}\dfrac{P(y_2|0)}{P(y_2|1)}\dfrac{P(\widetilde{y_2}|0)}{P(\widetilde{y_2}|1)}\dfrac{P(y_1|1)}{P(y_1|1)}} = \\ &= \frac{\dfrac{P(y_2|0)}{P(y_2|1)}\dfrac{P(\widetilde{y_2}|0)}{P(\widetilde{y_2}|1)}\dfrac{P(y_1|0)}{P(y_1|1)}+1}{\dfrac{P(y_1|0)}{P(y_1|1)}+\dfrac{P(y_2|0)}{P(y_2|1)}\dfrac{P(\widetilde{y_2}|0)}{P(\widetilde{y_2}|1)}} =\end{aligned}$$

Observing that $L_1 = \dfrac{P(y_1|0)}{P(y_1|1)}$, $L_2 = \dfrac{P(y_2|0)}{P(y_2|1)}$ and $\tilde{R}_2 = \dfrac{P(\widetilde{y_2}|0)}{P(\widetilde{y_2}|1)}$ we get:

$$\widehat{L_1} = \frac{L_2\widetilde{R_2}L_1+1}{L_1+\widetilde{R_2}L_2} \tag{3.57}$$

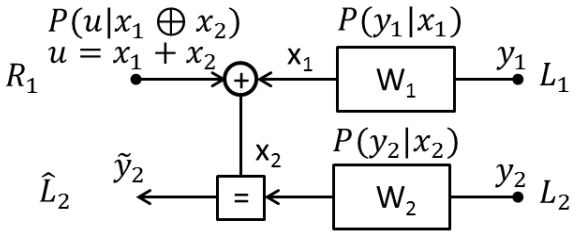


Figure 3.11: *Basic computational block for BP polar code decoding. Pointed out messages for the evaluation of $\widehat{L_2}$ information.*

With similar process, with reference to Figure 3.11, it is possible to compute the lower left propagating message $\widehat{L_2}$ as

$$\widehat{L_2} = \frac{P(\widetilde{y_2}=0|u,y_1,y_2)}{P(\widetilde{y_2}=1|u,y_1,y_2)} =$$

$$= \frac{P_{x_1,x_2,y_1,y_2,u}(x_1, u=x_1, y_1, y_2, u=x_1\oplus x_2)}{P_{x_1,x_2,y_1,y_2,u}(x_1, u=x_1\oplus 1, y_1, y_2, u=x_1\oplus x_2)} =$$

where the variable x_2 is imposed equal 0 in the numerator and equal 1 in denominator.

$$\widehat{L_2} = \frac{\sum_{x_1=0}^{1} P(x_1)\cdot P(y_1|x_1)\cdot P(u|x_1)\cdot P(y_2|0)\cdot P_{x_2}(0)}{\sum_{x_1=0}^{1} P(x_1)\cdot P(y_1|x_1)\cdot P(u|x_1\oplus 1)\cdot P(y_2|1)\cdot P_{x_2}(1)} =$$

$$= \frac{P_{x_1}(0)P(y_1|0)P(u|0)P(y_2|0)P_{x_2}(0) + P_{x_1}(1)P(y_1|1)P(u|1)P(y_2|0)P_{x_2}(0)}{P_{x_1}(0)P(y_1|0)P(u|1)P(y_2|1)P_{x_2}(1) + P_{x_1}(1)P(y_1|1)P(u|0)P(y_2|1)P_{x_2}(1)} =$$

$$= \frac{\frac{P_{x_1}(0)}{P_{x_1}(1)}\frac{P(y_1|0)}{P(y_1|1)}\frac{P(u|0)}{P(u|1)}\frac{P(y_2|0)}{P(y_2|0)}\frac{P_{x_2}(0)}{P_{x_2}(0)} + 1}{\frac{P_{x_1}(0)}{P_{x_1}(1)}\frac{P(y_1|0)}{P(y_1|1)}\frac{P(u|1)}{P(u|1)}\frac{P(y_2|1)}{P(y_2|0)}\frac{P_{x_2}(1)}{P_{x_2}(0)} + \frac{P_{x_1}(1)}{P_{x_1}(1)}\frac{P(y_1|1)}{P(y_1|1)}\frac{P(u|0)}{P(u|1)}\frac{P(y_2|1)}{P(y_2|0)}\frac{P_{x_2}(1)}{P_{x_2}(0)}} =$$

$$= \frac{\frac{P(y_1|0)}{P(y_1|1)}\frac{P(u|0)}{P(u|1)} + 1}{\frac{P(y_1|0)}{P(y_1|1)}\frac{P(y_2|1)}{P(y_2|0)} + \frac{P(u|0)}{P(u|1)}\frac{P(y_2|1)}{P(y_2|0)}} =$$

$$\widehat{L_2} = \frac{L_1R_1+1}{L_1\frac{1}{L_2}+R_1\frac{1}{L_2}} = L_2\frac{R_1L_1+1}{L_1+R_1} \tag{3.58}$$

So the result for the $\widehat{L_2}$ probability ratio is given by 3.58.

Comparing Figure 3.11 with Figure 3.12, it easy to note that by considering

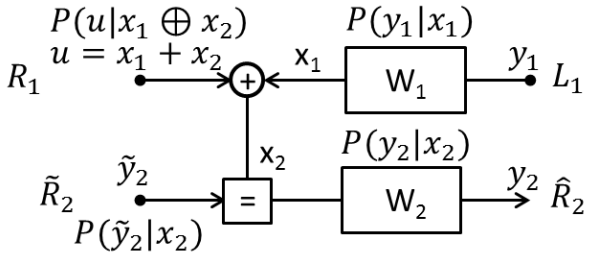


Figure 3.12: *Basic computational block for BP polar code decoding. Pointed out messages for the evaluation of $\widehat{R_2}$ information.*

$P(\widetilde{y_2}|x_2)$ instead of $P(y_2|x_2)$, equations should remain the same. So it is possible to substitute L_2 with $\widetilde{R_2}$ to obtain the result 3.59.

$$\widehat{L_2} = L_2 \frac{R_1 L_1 + 1}{L_1 + R_1} \quad \Rightarrow \quad \widehat{R_2} = \widetilde{R_2} \frac{R_1 L_1 + 1}{L_1 + R_1} \tag{3.59}$$

Last result for upper right propagating message $\widehat{R_1}$ with reference of Figure 3.13

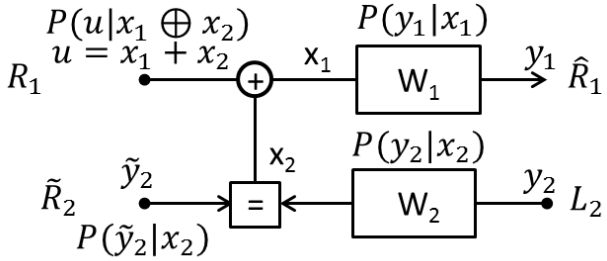


Figure 3.13: *Basic computational block for BP polar code decoding. Pointed out messages for the evaluation of $\widehat{R_1}$ information.*

can be computed similarly to 3.57 by

$$\begin{aligned}
\widehat{R_1} &= \frac{P(y_1=0|u,y_2,\widetilde{y_2})}{P(y_1=1|u,y_2,\widetilde{y_2})} = \\
&= \frac{P_{x_1,x_2,y_2,\widetilde{y_2},u}(0,x_2,y_2,\widetilde{y_2},x_2)}{P_{x_1,x_2,y_2,\widetilde{y_2},u}(1,x_2,y_2,\widetilde{y_2},x_2\oplus 1)} = \\
&= \frac{\sum_{x_2=0}^{1} P_{x_1}(0)\cdot P(x_2)\cdot P(y_2|x_2)\cdot P(\widetilde{y_2}|x_2)\cdot P(u|x_2)}{\sum_{x_2=0}^{1} P_{x_1}(1)\cdot P(x_2)\cdot P(y_2|x_2)\cdot P(\widetilde{y_2}|x_2)\cdot P(u|x_2\oplus 1)} = \\
&= \frac{P_{x_1}(0)P_{x_2}(0)P(y_2|0)P(\widetilde{y_2}|0)P(u|0)+P_{x_1}(0)P_{x_2}(1)P(y_2|1)P(\widetilde{y_2}|1)P(u|1)}{P_{x_1}(1)P_{x_2}(0)P(y_2|0)P(\widetilde{y_2}|0)P(u|1)+P_{x_1}(1)P_{x_2}(1)P(y_2|1)P(\widetilde{y_2}|1)P(u|0)} = \\
&= \frac{\frac{P_{x_1}(0)}{P_{x_1}(0)}\frac{P_{x_2}(0)}{P_{x_2}(1)}\frac{P(y_2|0)}{P(y_2|1)}\frac{P(\widetilde{y_2}|0)}{P(\widetilde{y_2}|1)}\frac{P(u|0)}{P(u|1)}+1}{\frac{P_{x_1}(1)}{P_{x_1}(0)}\frac{P_{x_2}(0)}{P_{x_2}(1)}\frac{P(y_2|0)}{P(y_2|1)}\frac{P(\widetilde{y_2}|0)}{P(\widetilde{y_2}|1)}\frac{P(u|1)}{P(u|1)}+\frac{P_{x_1}(1)}{P_{x_1}(0)}\frac{P_{x_2}(1)}{P_{x_2}(1)}\frac{P(y_2|1)}{P(y_2|1)}\frac{P(\widetilde{y_2}|1)}{P(\widetilde{y_2}|1)}\frac{P(u|0)}{P(u|1)}} = \\
&= \frac{\frac{P(y_2|0)}{P(y_2|1)}\frac{P(\widetilde{y_2}|0)}{P(\widetilde{y_2}|1)}\frac{P(u|0)}{P(u|1)}+1}{\frac{P(y_2|0)}{P(y_2|1)}\frac{P(\widetilde{y_2}|0)}{P(\widetilde{y_2}|1)}+\frac{P(u|0)}{P(u|1)}} = \\
\end{aligned}$$

$$\widehat{R_1} = \frac{\widetilde{R_2}L_2R_1+1}{\widetilde{R_2}L_2+R_1} \tag{3.60}$$

Logarithmic representation allows to convert multiplications into sums, so in order to simplify final equations to implement, LR messages are converted to log-likelihood ratio messages (LLRs) according to 3.61.

$$\begin{aligned}
\ell_d &\triangleq \ln L_d \\
r_d &\triangleq \ln R_d
\end{aligned} \tag{3.61}$$

where d is the considered message index. To implement all equations, with reference to general graph representation (ex. Figure 3.9), equations 3.57, 3.58, 3.60 and 3.59 can be respectively written as

$$\ell_{i,j}^{(t+1)} = f\left(\ell_{i+1,j}^{(t)},\, \ell_{i+1,j+N_i}^{(t)} + r_{i,j+N_i}^{(t)}\right) \tag{3.62}$$

$$\ell_{i,j+N_i}^{(t+1)} = \ell_{i+1,j+N_i}^{(t)} + f\left(\ell_{i+1,j}^{(t)},\, r_{i,j}^{(t)}\right) \tag{3.63}$$

$$r_{i+1,j}^{(t+1)} = f\left(r_{i,j}^{(t)},\, \ell_{i+1,j+N_i}^{(t)} + r_{i,j+N_i}^{(t)}\right) \tag{3.64}$$

$$r_{i+1,j+N_i}^{(t+1)} = r_{i,j+N_i}^{(t)} + f\left(r_{i,j}^{(t)},\, \ell_{i+1,j}^{(t)}\right) \tag{3.65}$$

where log-likelihood ratio (LLR) of left propagating messages $\ell_{i,j}^{(t)}$ are considered for the (i,j) node at time (or equivalently iteration) index $t = 0,1,\ldots$, and for the same node also exist a LLR of right propagating message $r_{i,j}^{(t)}$. These equations are implemented in processing element (PE) to perform the iterative update of LLR messages. The function f is given by

$$f(x,y) = \ln\left[\frac{(1+e^{(x+y)})}{(e^x+e^y)}\right] \tag{3.66}$$

for any two LLRs x, y. This function f can be replaced by min-sum approximation as presented in section 3.8. LLR messages that correspond to initialization are

$$\ell_{n+1,j}^{(0)} = \ln\left(\frac{P(y_j|x_j=0)}{P(y_j|x_j=1)}\right) = \ln\left(\frac{W(y_j|0)}{W(y_j|1)}\right) \tag{3.67}$$

$$r_{1,j}^{(0)} = \begin{cases} 0 & \text{if } j \text{ is a data index} \\ \infty & \text{if } j \text{ is a frozen-bit index.} \end{cases} \tag{3.68}$$

because by setting $r_{1,j}^{(0)} = \infty$ for a frozen-bit it means that the associated probability to be a zero is 1 and consequently the probability of being a one goes to 0, while by setting a 0 it corresponds (in logarithms) of having equal probability to have value zero or one in position j. Every other $r_{i,j}^{(0)}$ and $\ell_{i,j}^{(0)}$ will be equal 0 at iteration $t = 0$. As previously presented every PE is composed of four connection nodes, two for each side of the same level, so for a general graph for polar decoding of length N, the direct mapping implementation will be composed of $\frac{1}{2}N\log_2 N$ PEs.
The decision about the final result at the end of t iteration for every variable node (i,j) is taken by computing $\ell_{i,j}^{(t)} + r_{i,j}^{(t)}$ but for practical BP decoding only first level should be analyzed in order to obtain uncoded estimated vector $\widehat{u}$ as in 3.69

$$\begin{aligned} \widehat{u_j} &= 0 \text{ if } \ell_{1,j}^{(t)} + r_{1,j}^{(t)} \geq 0 \\ \widehat{u_j} &= 1 \text{ if } \ell_{1,j}^{(t)} + r_{1,j}^{(t)} < 0 \end{aligned} \tag{3.69}$$

3.5 Belief propagation scheduling

The scheduling of a belief propagation algorithm is the order in which messages are propagated in the graph. For the case of a code without cycles, the scheduling does not affect the algorithm convergence. For practical codes with cycles the order in

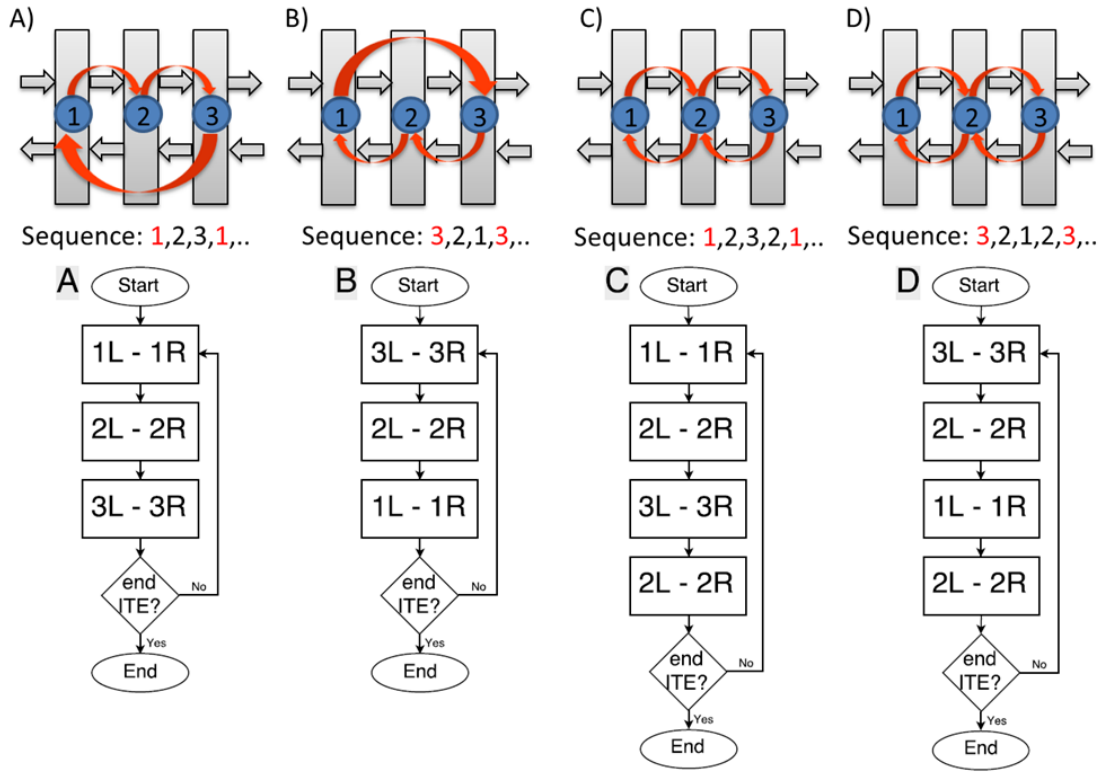


Figure 3.14: *Bidirectional scheduling algorithms: A)Linear LR. B)Linear RL. C)Circular LR. D)Circular RL.*

which processing elements elaborate messages and in which are activated (so what we refer when talking about scheduling) will affect the convergence due to the presence of cycles in the graph. In literature the scheduling problem has not been faced systematically for polar codes. A brief discussions about this topic is given in [35], where Hussami et. al. state that for general BDMC the chosen scheduling directly affects decoding performance and only present their used scheduling empirically found (considered later in this work as scheduling C). Also in [4] this scheduling is used and it is stated to led to better performance among other tested schedules by authors.
Implicitly in [36] another scheduling is considered, it is described below as scheduling G, and it allows to significantly reduce implemented hardware resources and as consequence improve critical path delay.
In order to describe different scheduling algorithms for BP polar decoding, it has been considered the general graph structure, every level (column of PEs) has been considered a single block that could evaluate left or right LLR messages. Eleven BP

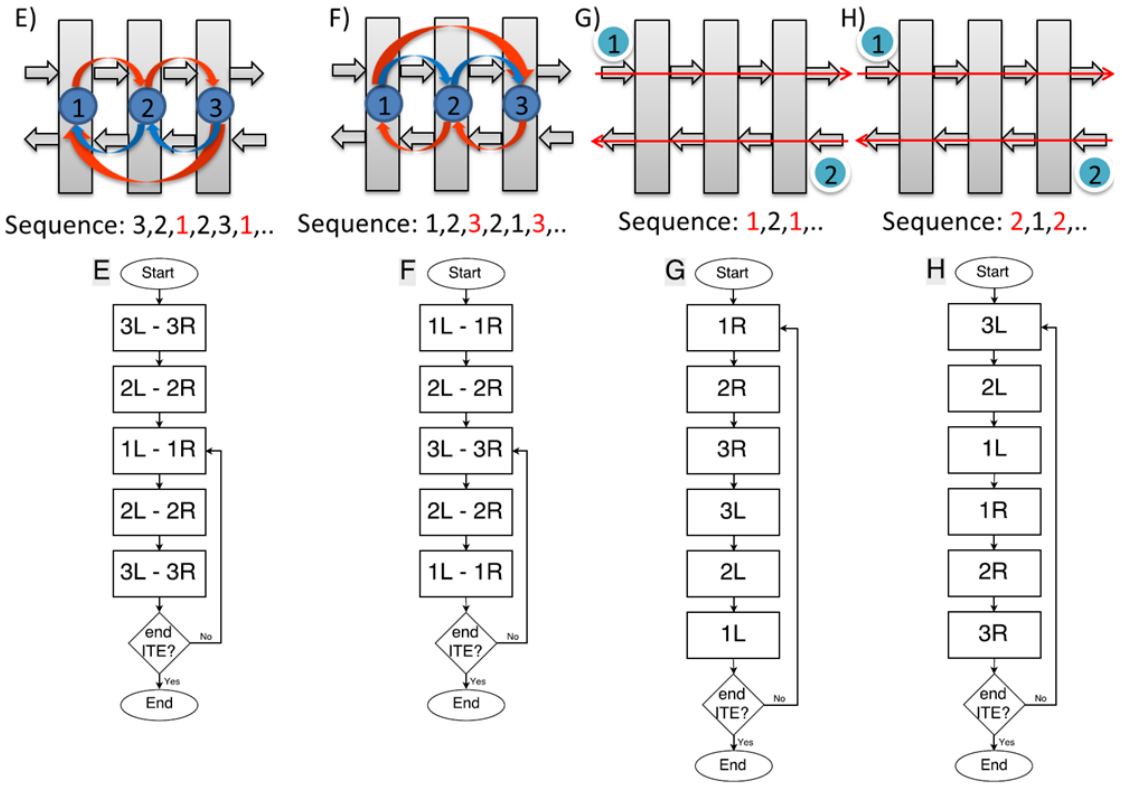


Figure 3.15: *Bidirectional scheduling algorithms: A)Linear LR, start R. B)Linear RL, start L. Unidirectional scheduling algorithms: G)Circular LR. H)Circular RL.*

scheduling have been considered, they have been identified by capital letters and an intuitive name as

A Linear bidirectional Left to Right. Every iteration starts from left-most stage and ends at the right-most one; LLRs updates are bidirectional.

B Linear bidirectional Right to Left. Every iteration starts from right-most stage and ends at the left-most one; LLRs updates are bidirectional.

C Circular bidirectional Left to Right. One iteration is composed of bidirectional updates of each stage from left-most to right-most and back from right-most to left-most.

D Circular bidirectional Right to Left. Bidirectional updates in one iteration are performed from right-most to left-most and back from left-most to right-most.

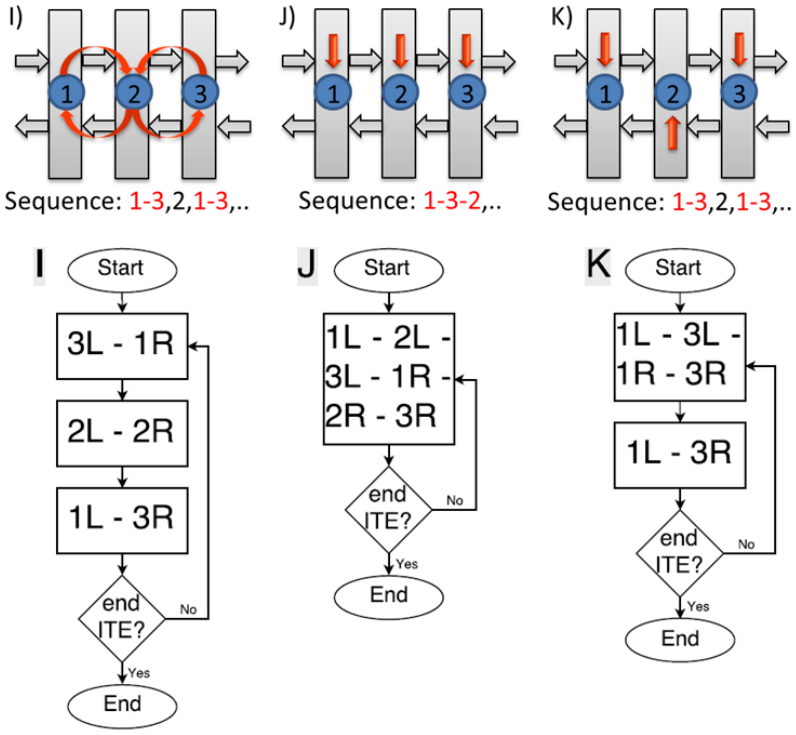


Figure 3.16: *Scheduling algorithms: I)Circular double-wave. J)All-on. K)Odd-Even.*

E Linear bidirectional Left to Right, Right start. The initial iteration is performed as B scheduling, then following iterations are like A scheduling.

F Linear bidirectional Right to Left, Left start. The initial iteration is performed as A scheduling, then following iterations are like B scheduling.

G Circular unidirectional Left to Right. During one iteration stages are updated as C scheduling, but when from left-most to right-most updates only right messages are propagated, then from right-most to left-most only left messages are updated.

H Circular unidirectional Right to Left. During one iteration stages are updated as D scheduling, but when from right-most to left-most updates only left messages are propagated, then from left-most to right-most only right messages are updated.

I Circular double-wave. Both scheduling G and H are executed in parallel.

J All-on. All stages are bidirectionally updated in parallel at every iteration.

K Odd-Even. During one iteration only odd numbered stages are enabled, then in the following step only even numbered stages are updated. LLRs updates are bidirectional.

All presented scheduling algorithms (SAs) have been depicted in Figures 3.14, 3.15 and 3.16 for the case of a 3 stage decoder (equivalent to code block length of $N = 8$), together with their respective flow diagram for a better comprehension. Every stage i may update $\ell_{i,j}^{(t)}$ or $r_{i+1,j}^{(t)}$ for every $1 \leq j \leq N$ according to equation from 3.62 to 3.65 for a given t. In the presented flow graphs $\ell_{i,j}$ computations are indicated by iL and similarly $r_{i+1,j}$ updates as iR.
A scheduling is called *bidirectional* if in the same operation the entire column of PEs which compose one stage outputs both left and right messages at once. On other hand if in the same state only left or right updates are carried out, the scheduling is named *unidirectional*. With reference to the Figures from 3.14 to 3.16, SAs can be easily categorized by the observation of its related flow chart, if inside the same state iL and iR appear, it is *bidirectional* scheduling while if only one of them appear it is called *unidirectional*.
The minimum sequence of different operations that have to be performed, is called *iteration*. In Figures from 3.62 to 3.65 it is also shown a sample sequence of activated stages for each scheduling. The red numbers identify the initial stage labels of every iteration.
The specific target is to find best scheduling for high-throughput BP decoding, where all LLRs are in parallel. In fact it is easy to see from equations 3.62, 3.63, 3.64 and 3.65 that there is no inputs dependence for a single processing stage. So it is granted that PEs maximum delay will be the smallest possible by exploiting intrinsic hardware parallelism. Therefore this solution avoids of assigning an order of computation of LLRs as in [35] and [4], but follows the classical belief propagation philosophy of "flooding", that means that messages from the same node are all updated together, so inside all PEs all messages are parallel.

In order to fairly compare the SAs, the iteration parameter is not a good candidate, because the number of operations change from one scheduling to the other, so at parity of iteration it is not considered the delay to achieve output results, which for a practical implementation means time delay. The choice of the scheduling have to point out the best solution in terms of convergence of the algorithm compared to the effort to obtain it. To quantify the effort it has been considered the number of states of the considered scheduling and the number of parallel operation for each state. In Table 3.1 is presented the general result for each schedule in terms of number of states (S), parallel operation (PO) and the derived parameter operation frequency $f_{OP} = \frac{1}{S \cdot PO}$, which describes the "weight" of single operation compared to a full iteration of the scheduling.

Scheduling	# of states S	parallel operations PO	operation frequency f_{OP}
A E	n	2	$\frac{1}{2n}$
B F	n	2	$\frac{1}{2n}$
C	$2n-2$	2	$\frac{1}{4(n-1)}$
D	$2n-2$	2	$\frac{1}{4(n-1)}$
G	$2n-2$	1	$\frac{1}{2(n-1)}$
H	$2n-2$	1	$\frac{1}{2(n-1)}$
I	$n-1$	2	$\frac{1}{2(n-1)}$
J	1	n	$\frac{1}{n}$
K	2	$n/2$	$\frac{1}{n}$

Table 3.1: *Scheduling properties for codeword length* $N = 2^n$.

Nevertheless to perform some simulation on different codes, the number of iteration is still needed. So we decided to assign a relatively small fixed number of maximum operation, to evaluate comparable number of iterations for the different SAs presented. The choice of this number of total operation has been evaluated iteratively, where at least one algorithm behaves well and differentiation among the obtained performance of the SAs is good. With the implementation prospective, the number of total operation can be thought as the allowed number of clock cycles for the final architecture, so the number of states S becomes the number of clock period per iteration and the product of number of operation and operation frequency becomes the number of iteration of the architecture design.
The simulation result of different SAs performance for the code length of $N = 2^{10}$ is presented in Figure 3.17. The number of iterations used for 200 operations are: A, B, E, F 10; G, H, I 11; J, K 20. In the Figure presented, (A, E) and (B, F) SAs have been regrouped because their result are superposed, while C and D are not drawn because for the same number of iteration they have performance respectively equal to G and H, as proven in [37]. Note that the presence of the Successive Cancellation (SC) decoding curve, below other curves grants that by increasing the number of iterations, these curves would get closer and even overstep it. The min-sum approximation (discussed in Section 3.8) for both BP SAs and SC decoding has been used

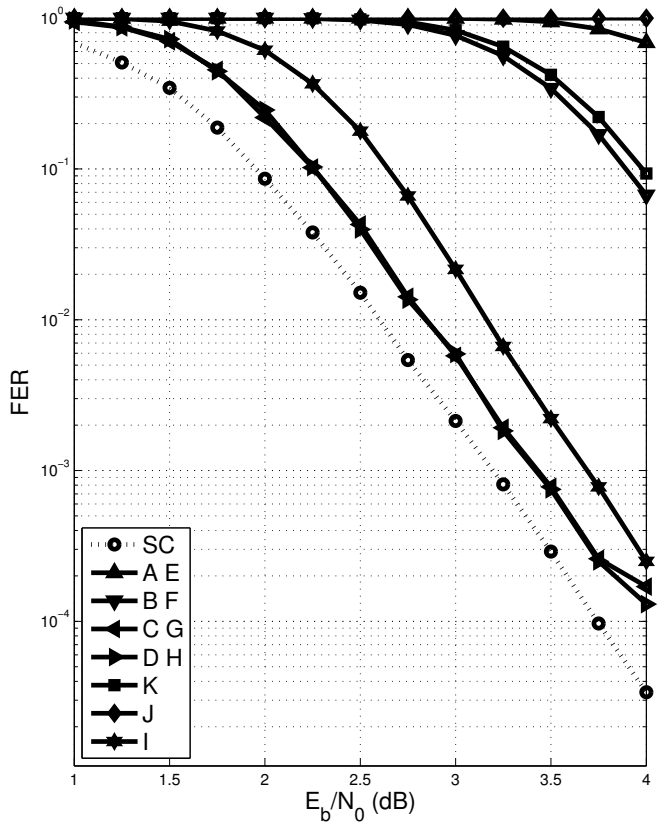


Figure 3.17: *Scheduling algorithms comparison for* $N = 1024$*, rate* $\frac{1}{2}$.

in order to consider also the approximation effects on scheduling algorithms.
Most efficient algorithms appear to be G and H, where only one operation per state is used, however another interesting schedule is I, because if we unconstrained the number of parallel operation (which for the implementation becomes number of PE stages), same performances can be achieved in half iterations. The number of parallel operation so is clearly related to the area used in the implementation, therefore for other SAs the efficiency drops and the comparable number of parallel operations makes clear that it would be possible to achieve better throughput to area ratio performance just by replicating a simpler architecture with one of the previous scheduling analyzed.

3.6 Uniform Belief Propagation Decoder Structure

The factor graph for polar coding, as represented in Figure 3.9, could be implemented into encoder and decoder architectures (both software or hardware) as is. However the non-uniform structure of the graph from one level to the next makes more difficult to reuse processing modules. It is possible to give a more uniform architecture design for the graph by introducing an operator which reorders the index of input vector.[34]

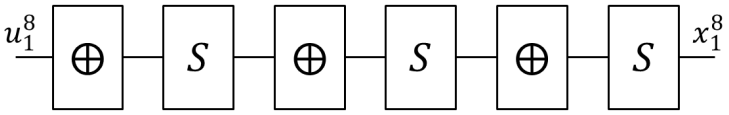


Figure 3.18: *Uniform graph representation with $shuffle$ operators for $F^{\otimes 3}$*

It is possible to obtain an equivalent graph architecture introducing a $shuffle$ operator S which maps an input vector v_1^N of length $N = 2^n$, into the vector $(v_1, v_{\frac{N}{2}+1}, v_2, v_{\frac{N}{2}+2}, \ldots, v_i, v_{\frac{N}{2}+i}, \ldots, v_{\frac{N}{2}}, v_N)$, where $1 \leq i \leq \frac{N}{2}$.
The block $\bigoplus$ is the module-2 addition operation that transform the vector v_1^N into $(v_1 \oplus v_2, v_2, v_3 \oplus v_4, v_4, \ldots, v_{2i-1} \oplus v_{2i}, v_{2i}, \ldots, v_{N-1} \oplus v_N, v_N)$, where $1 \leq i \leq \frac{N}{2}$. This $\bigoplus$ block is equivalent to the last level of Figure 3.9. In Figure 3.18 the new uniform structure graph is presented. Note that the uniform structure is end-to-end equivalent to the non-uniform case and only internal interconnections have been rearranged. The number of introduced S and $\bigoplus$ blocks is equal to n.

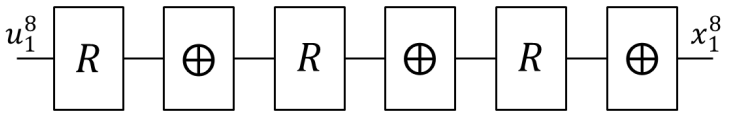
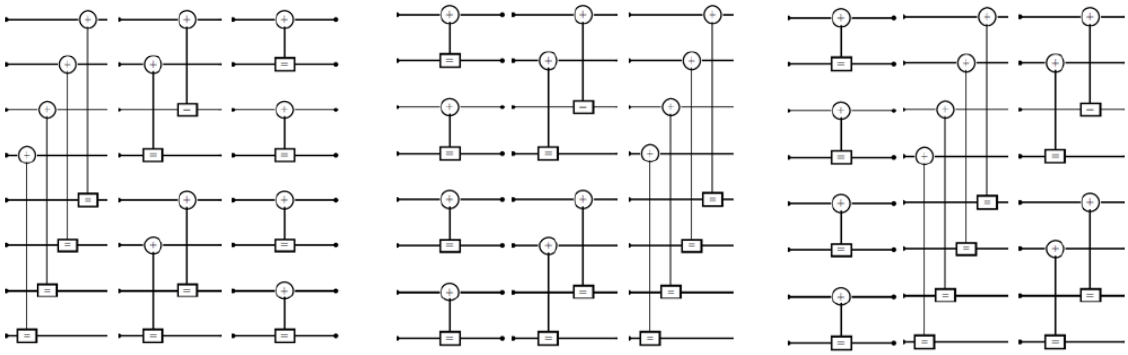


Figure 3.19: *Uniform graph representation with $reverse - shuffle$ operators for $F^{\otimes 3}$*

Another possible uniform graph representation is given in Figure 3.19, where R is the $reverse - shuffle$ operator that transforms vector v_1^N of even length N into $(v_1, v_3, \ldots, v_{i_{odd}}, \ldots, v_{N-1}, v_2, v_4, \ldots, v_{i_{even}}, \ldots, v_N)$, for $1 \leq i \leq N$. This graph implements the inverse of the previous uniform structure, so since the inverse of the generating matrix of polar codes is the matrix itself, also this realization is correct. Due to the particular simple structure of $\bigoplus$ block the previous two uniform realizations have been presented, nevertheless there are many other uniform factor graphs.

Hardware implementations can benefit from these uniform realizations because the same block can be reused without changing interconnections.

3.7 Graph Representation: Redundant Trellises



(a) (b) (c)

Figure 3.20: Example of three redundant trellises.

As observed in [35], the generating matrix of polar codes may be described correctly by many redundant representations. In Figure 3.20 three valid representation are given for a code block length $N = 8$. For a generic code block length $N = 2^n$ the number of possible different representations are $n!$. Decoding performance of these overcomplete representation are different even for other conditions being equal.
Since the problem was open it has been decided to systematically investigate property of redundant trellises for a dimension of code of practical use. It was selected $N = 2^{10}$, which leads to about $3.6 \cdot 10^6$ different structures. To perform this task a modified version of the simulation C code has been used. Starting from known architecture as defined in literature [34], to every stage has been assigned a number (P 0123456789), then all possible permutation of that sequence was used to represent different interconnections.
The list of structures under test has been realized as a linked list, in order to eliminate bad candidates and let only good structures to be further tested. In fact due to the big amount of candidates the simple moving on a list and checking a variable (for good or bad candidate) would require too much time.
The optimization search has been conducted for D scheduling algorithm for BP decoding with 15 iterations at 1dB SNR, in order to need few frames, but still observing FER variations on performance. Achieved results has pointed out that classical structure is among top 6 best trellises. Top 10 results have been also studied from 0 to 4dB, FER simulations are so close that can be considered practically

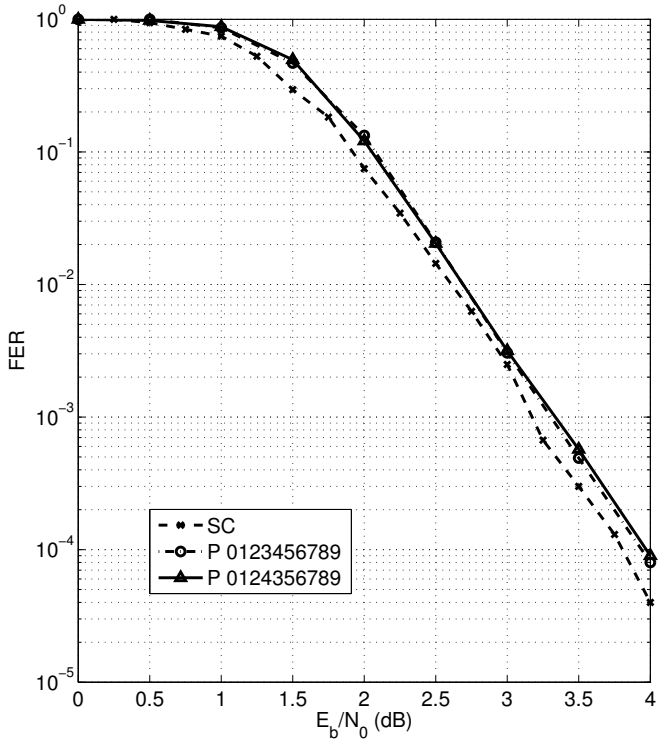


Figure 3.21: *Comparison result of FER performance of two redundant graph representations for 15 iterations.*

superposed. In Figure 3.21 the best architecture discovered (P 0124356789) is compared with the classical implementation.
Since the classical architecture has the good property of being an uniform structure it has been proven to be not rewarding the choice of a different trellis structure.

3.8 Min-Sum Approximation

The Min-sum approximation has been introduced in order to simplify the evaluation of the function $\Psi(x) = ln\ tgh\frac{x}{2} = ln\ \frac{e^x-1}{e^x+1}$ for the algorithm implementation of Belief Propagation LDPC codes. With the finite precision of an hardware implementation this function shows numerical stability problems, so the introduction of a problem relaxation (with small losses < 0.5dB) is acceptable considered that losses would occur in any case. Other solutions, different from the one here exposed, are realized by *Look up* tables or piecewise linear approximations of the function Ψ. But in

every case exists the problem of numerical stability.
This simplification firstly appeared in [38] named λ-Min Algorithm. It represents a good compromise between achieved result and implementation complexity. The use of this solution for polar codes has been proposed in [3].
Let us consider the module of the probability ratio for the general parity check φ verified by expression 3.70.

$$\Lambda_{\varphi} = 2tgh^{-1}\left(\prod_{i=1}^{n} tgh\left(\frac{|\Lambda_i|}{2}\right)\right) \tag{3.70}$$

When $n = 2$ it can be equivalently written as

$$\Omega(x,y) = 2tgh^{-1}\left(tgh\frac{x}{2}tgh\frac{y}{2}\right) = 2sgn(x)sgn(y)tgh^{-1}\left(tgh\frac{|x|}{2}tgh\frac{|y|}{2}\right) \tag{3.71}$$

Where function $\Omega(x,y)$ is proven to be an associative operator, for example if it is given $\varphi = c_1 \oplus c_2 \oplus c_3$, $\Lambda_{12} = \Omega(\Lambda_1, \Lambda_2)$ and $\Lambda_{23} = \Omega(\Lambda_2, \Lambda_3)$ then $\Omega(\Lambda_{12}, \Lambda_3) = \Omega(\Lambda_1, \Lambda_{23})$, that means that it does not care the order in which they are evaluated. In Figure 3.22 is reported a graphical representation of function $\Omega(x,y)$.
Using the identity $tgh^{-1}(x) = \frac{1}{2}ln\frac{1+x}{1-x}$ and Eulero formulas through following algebraic passages is possible to obtain an expression in terms of exponentials and logarithms.

$$2tgh^{-1}\left(tgh\frac{|x|}{2}tgh\frac{|y|}{2}\right) =$$

$$= ln\frac{1 + tgh\frac{|x|}{2}tgh\frac{|y|}{2}}{1 - tgh\frac{|x|}{2}tgh\frac{|y|}{2}} =$$

$$= ln\frac{1 + \frac{e^{\frac{|x|}{2}} - e^{-\frac{|x|}{2}}}{e^{\frac{|x|}{2}} + e^{-\frac{|x|}{2}}}\frac{e^{\frac{|y|}{2}} - e^{-\frac{|y|}{2}}}{e^{\frac{|y|}{2}} + e^{-\frac{|y|}{2}}}}{1 - \frac{e^{\frac{|x|}{2}} - e^{-\frac{|x|}{2}}}{e^{\frac{|x|}{2}} + e^{-\frac{|x|}{2}}}\frac{e^{\frac{|y|}{2}} - e^{-\frac{|y|}{2}}}{e^{\frac{|y|}{2}} + e^{-\frac{|y|}{2}}}} =$$

$$= ln\frac{\left(e^{\frac{|x|}{2}}+e^{-\frac{|x|}{2}}\right)\left(e^{\frac{|y|}{2}}+e^{-\frac{|y|}{2}}\right)+\left(e^{\frac{|x|}{2}}-e^{-\frac{|x|}{2}}\right)\left(e^{\frac{|y|}{2}}-e^{-\frac{|y|}{2}}\right)}{\left(e^{\frac{|x|}{2}}+e^{-\frac{|x|}{2}}\right)\left(e^{\frac{|y|}{2}}+e^{-\frac{|y|}{2}}\right)-\left(e^{\frac{|x|}{2}}-e^{-\frac{|x|}{2}}\right)\left(e^{\frac{|y|}{2}}-e^{-\frac{|y|}{2}}\right)} =$$

$$= ln\frac{2e^{\frac{|x|+|y|}{2}}+2e^{\frac{-|x|-|y|}{2}}}{2e^{\frac{|x|-|y|}{2}}+2e^{\frac{-|x|+|y|}{2}}} =$$

$$= ln\left(\frac{e^{\frac{|x|+|y|}{2}}}{e^{\frac{|x|+|y|}{2}}}\frac{1+e^{-|x|-|y|}}{e^{-|x|}+e^{-|y|}}\right) =$$

$$ln\left(1+e^{-|x|-|y|}\right)-ln\left(e^{-|x|}+e^{-|y|}\right) \tag{3.72}$$

Notice that same process to obtain 3.72 without considering the module of inputs give as results 3.66.
It is also valid following property:

$$ln\left(e^{a}+e^{b}\right)=\max(a,b)+ln\left(1+e^{-|a-b|}\right) \tag{3.73}$$

Demonstration :

Let $a > b$, means that $b-a<0$, $ln\left(e^{a}+e^{b}\right) = ln\left[e^{a}\left(1+e^{b-a}\right)\right] = a + ln\left(1+e^{-|a-b|}\right)$.
Let $b > a$, $a-b<0$ so: $ln\left(e^{a}+e^{b}\right) = ln\left[e^{b}\left(1+e^{a-b}\right)\right] = b + ln\left(1+e^{-|a-b|}\right)$.
That means 3.73 is valid.

As consequence 3.72 can be rewritten as follows:

$$\begin{array}{ll} ln\left(1+e^{-|x|-|y|}\right)-\max(-|x|,-|y|)-ln\left(1+e^{-|-|x|+|y||}\right) & = \\ ln\left(1+e^{-|x|-|y|}\right)+\min(|x|,|y|)-ln\left(1+e^{-||x|-|y||}\right) & = \end{array}$$

So 3.71 becomes:

$$\Omega(x,y)=sgn(x)sgn(y)\left(\min(|x|,|y|)+ln\frac{1+e^{-|x|-|y|}}{1+e^{-||x|-|y||}}\right) \tag{3.74}$$

This exact result can be approximated neglecting the second term in parenthesis. This operation is correct when $|x|, |y| \gg 0$ and $||x| - |y|| \gg 0$, moreover this second term is exactly equal to 0 when $x = 0$ or $y = 0$. In Figure 3.24 is shown the error introduced by using the approximation compared to the use of function $\Omega(x, y)$.

In general the loss of performance introduced by the use of min-sum approximation

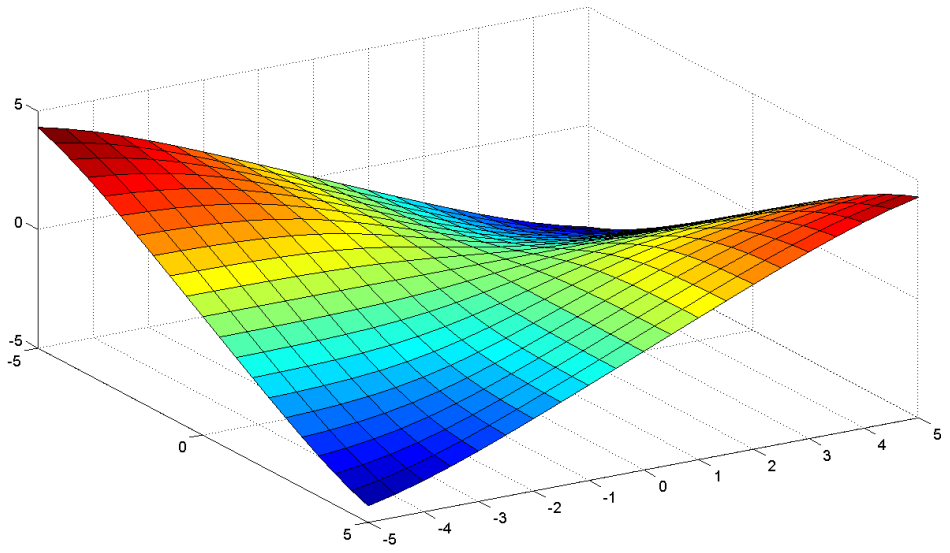


Figure 3.22: *Function* $\Omega(x, y)$.

compared to the use of exact formula is partially recovered, through two correction methods:

- Offset
- Normalization

In the first case, the equation of the processing node becomes:

$$\tilde{\Omega}(x, y) = sign(x) \cdot sign(y) \cdot \max\left(\min(|x|, |y|) - \beta, 0\right) \tag{3.75}$$

where β is an optimized offset for the quantization scheme used in the architecture. Typical values of β are not explored in literature for polar codes (to the best of our knowledge) but referring to LDPC we expect $\beta \leq 0.25$.
In the second case the equation becomes:

$$\tilde{\Omega}(x, y) = sign(x) \cdot sign(y) \cdot \alpha \cdot \min(|x|, |y|) \tag{3.76}$$

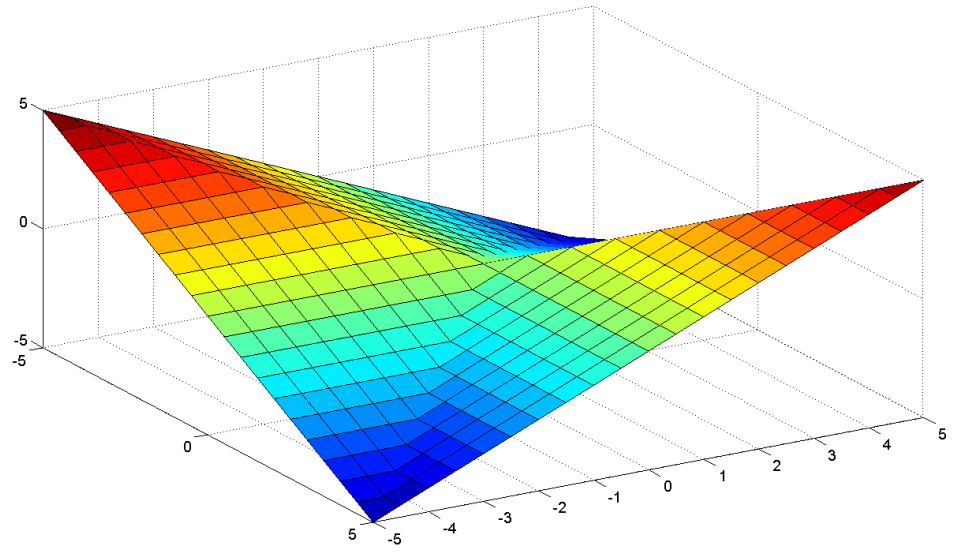


Figure 3.23: *Min-Sum approximation of function* $\Omega(x, y)$.

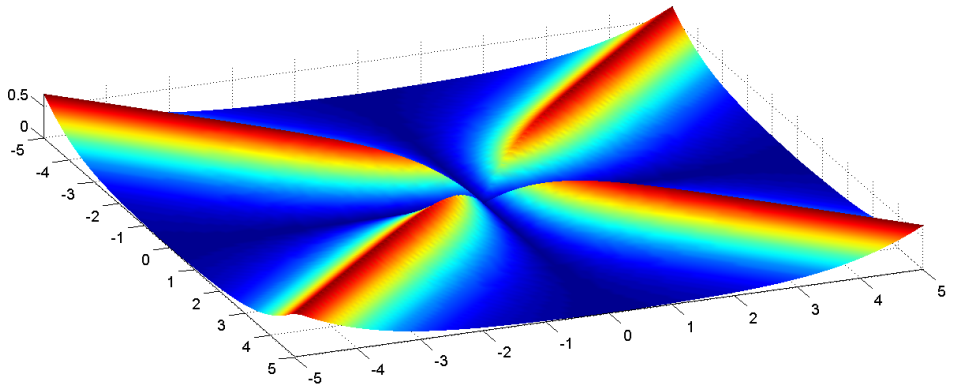


Figure 3.24: *Approximation error of using Min-sum compared to function* $\Omega(x, y)$.

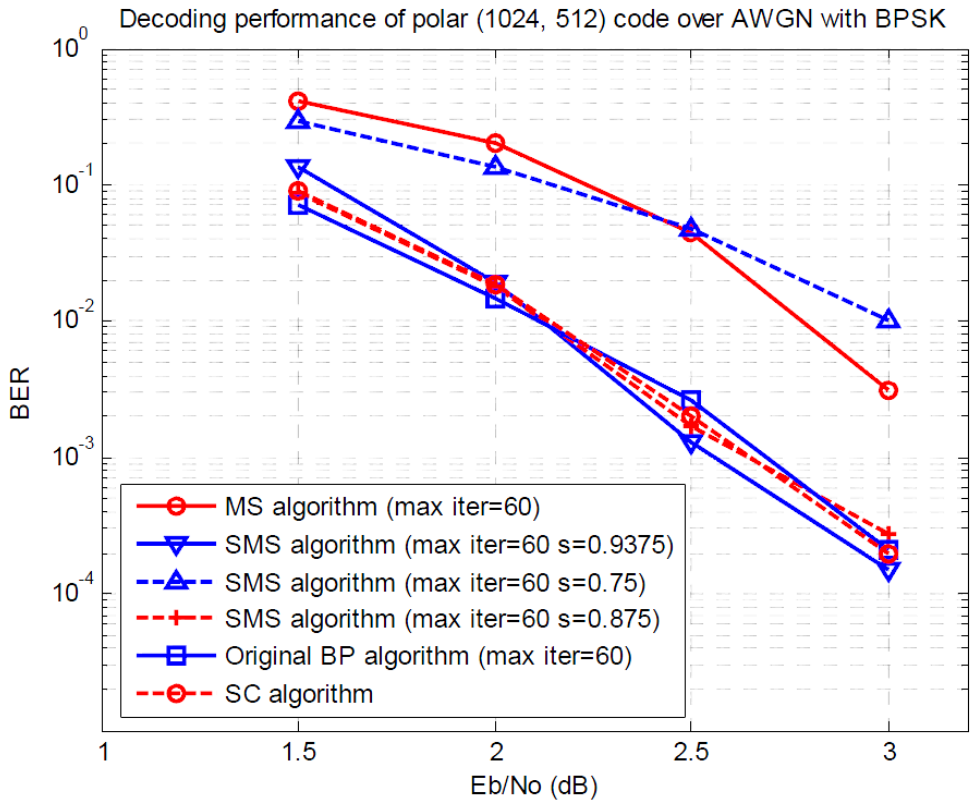


Figure 3.25: *Performance comparison of approximations for polar decoding. From [3].*

$\alpha < 1$ is the normalization factor for the processing node, its value is typically $\alpha \geq 0.8$. This case has been explored in [3] with the name of Scaled Min-Sum (SMS). In Figure 3.25 with a normalization factor of $\alpha = 0.9375$ for the case of BPSK modulation over AWGN channel, codeword length 1024 and rate $\frac{1}{2}$ is shown that the loss introduced by the min-sum approximation is compensated by the scaling factor α. Performance are therefore overlapped to original Belief Propagation algorithm (BP) at 60 iterations and Successive Cancellation (SC) algorithm. The choice of $\alpha = 0.9375 = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} = 1 - 2^{-4}$ is made because of the easiness of hardware implementation, in fact it can be realized with just a shift and a sum stage. This choice has also been adopted in the current work for hardware implementations.
In principle even complex normalization algorithms or different corrections of offset and normalization could be combined to achieve better results, but good proven

results with simple SMS architecture make the choice of introducing complex correction an inadvisable path with the final hardware implementation purpose. The study of variation of codes performances due to the variation of introduced correction also depends on bit quantization and the decoding algorithm used, these studies are called *Density Evolution Analysis*.

3.9 Performance evaluation

Usually communication system with uncoded messages, compared to system with coded data behaves like in Figure3.26. For E_b/N_0 values lower than a certain quan-

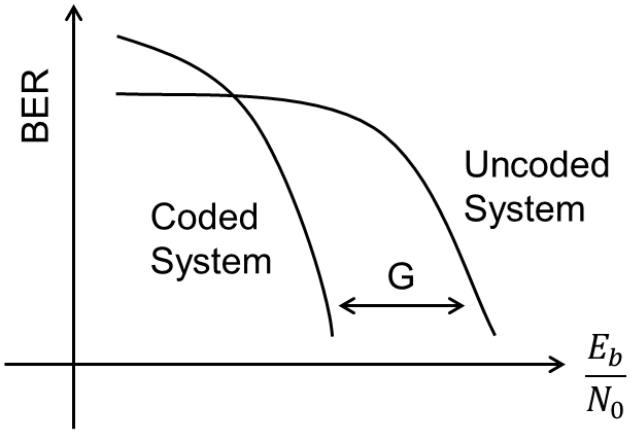


Figure 3.26: *Typical digital coded and uncoded communication system performance.*

tity (known as convergence abscissa), coded system has usually lower performance compared to uncoded ones: since some redundancy is introduced, more data are transmitted into the channel and so more errors may occur. Excluding this region where also coded communications are not reliable, coded systems allow the transmitted power reduction because the same error probability (*Bit Error Rate*) can be achieved with a lower signal to noise ratio E_b/N_0.
Fixed a certain BER value, it is called *Coding Gain* of a coded system, the difference in dB of the signal to noise ratio E_b/N_0 needed to the uncoded system compared to the coded one to achieve same desired BER.
The error ratio of iteratively decoded codes has a typical form as schematized in Figure 3.27 for two cases. For the solid line three different regions can be distinguished:

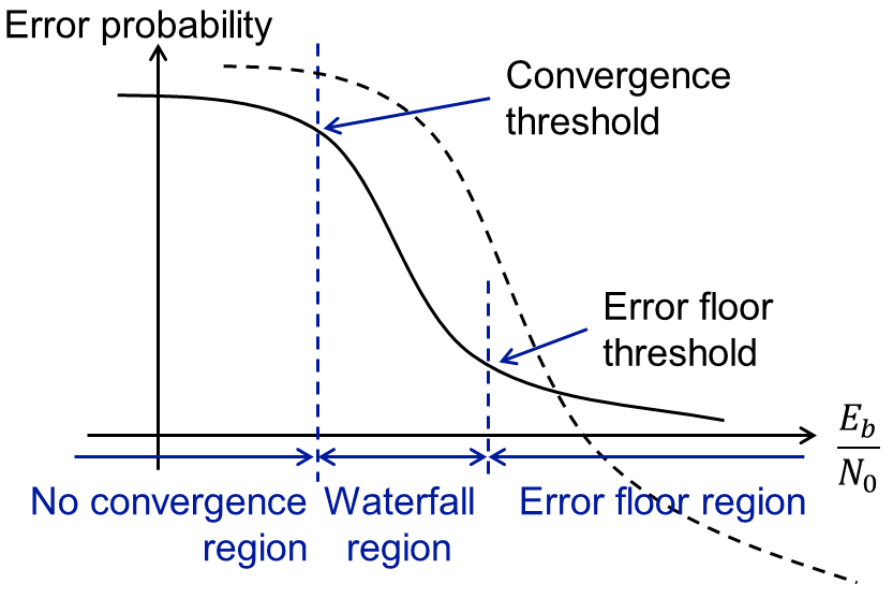


Figure 3.27: *Typical regions in an error probability curve for iterative decoding algorithms: the solid line identify the waterfall region and the error floor region. The trade-off between the two regions is illustrated by the second curve (dashed line) which has lower error floor at the expense of higher convergence threshold.*

- First region is where the code is inefficient, under the convergence threshold. Even if the number of iteration is increased there are no performance improvements.
- The *waterfall* region where the error rate has a sharp fall, which increases with the number of iterations.
- The *error floor* region is where the descent of the error rate is lowest than the waterfall region. The error floor is given to the minimum Hamming distance for the code.

As illustrated by the dashed curve, often exist a trade-off between code performance in the waterfall region and those in error floor region. For the specific case of polar codes is stated in [39] and [4] that BER performance of polar codes under belief propagation decoding shows no sign of error floor up to very low BER for the AWGN channel ($< 10^{-9}$ for $N = 2^{13}$, rate $\frac{1}{2}$). It is proven that this behaviour is due to the high minimum distance (equal to the *stopping distance*) achieved in Polar Codes and the *girth* of its related trellis graph.
It is called g_n the girth of a variable node vn_i, that is the length (number of paths) of

the smallest cycle which crosses vn_i. Being the *girth* of a graph $g_{min} = \min_{n\in\{(1,\dots N^2)\}}(g_n)$ the smallest number of steps that avoid auto-confirmation of information among all possible nodes. A *stopping set* is a non-empty set of nodes such that for every given set of nodes on the left-most side of the graph, every node that is connected to the right of the initial nodes is in the set, for this new set, apply same rule up to the right-most nodes. It is called *stopping tree* a stopping set with initial set of only one node that contains an information bit. The *stopping distance* is defined as the size of a *stopping set*.

In Figure 3.28 an example of cycles for polar codes is given. An interesting consid-

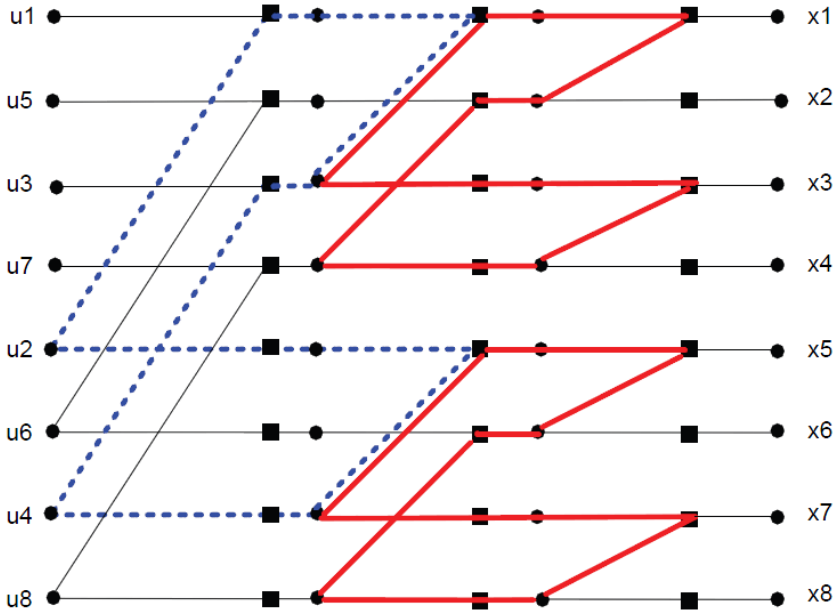


Figure 3.28: *Cycles example on polar code factor graph for the case of* $N = 8$*, girth* $g_{min} = 12$*. From [4].*

eration about those cycles is that since a polar code graph of code length $N = 2^n$ is always contained in a graph of code length $N = 2^{n+1}$, also cycles in smaller codes are always present in bigger ones. Moreover first order cycles appears in the graph for a code length of $N = 4$, a second order cycles which connects the first order cycles appears for $N = 8$, it clearly appears that successive cycles for increasing code lengths connects two halves of previous graph size.

Chapter 4

Belief Propagation Polar Decoder software model

4.1 Introduction to C model

The software model has been realized with the purpose of allowing to simulate polar codes performance under different conditions and also to evaluate the behave of hardware decoders. The general block structure is presented in Figure 4.1. The

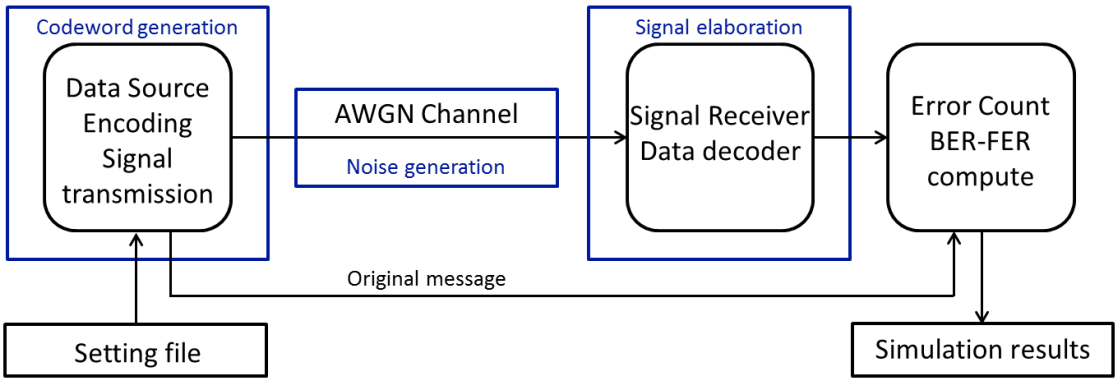


Figure 4.1: *Block description of C model.*

code start expecting as input a setting file, inside the file specific information are given to perform the simulation analysis. The first block is in charge to generate uniformly distributed random binary inputs as uncoded information. These data are then coded according to specified polar matrix to obtain coded bits sequences. Every coded sequence generated is a frame that is tested by sending it through the channel. The purpose of the block channel is to simulate effects of sending signals

through an AWGN channel and receiving its output. After this block, the channel received signals must be converted into soft-information to feed the described polar decoder. The polar decoder carry out its duty by iterating multiple times the decoding algorithm. When decoder iterations are completed a final hard decision is taken based on final soft-updates. As last step for the single frame the decoded bitstream is compared with original message and error statistics are updated. Multiple frames are processed in order to statistically estimate performance of the decoding structure through updates on error statistics.

4.1.1 Setting file

The setting file is a text file passed as parameter to the main function of the software. To give a clear insight of the developed code let us comment information contained in the setting file. The information contained in the setting file are:

Output file Name of the result output file.

Check results period Number of seconds to display an output. This output gives the number of processed frames out of total and an estimation of the time needed to complete decoding simulation. This value is also used for the periodic save of updated error statistics on a log file.

uncoded frame size Number of uncoded bits randomly generated

Bhattacharyya epsilon This variable gives the initial erasure probability of the BEC channel to compute the symmetric capacity of generated synthetic channels. This parameter is ignored if it is selected (inside the code) to use specific Bhattacharyya parameters evaluated through simulations for the AWGN channel.

Max number of iterations It is the maximum number of iteration to perform the BP decoding.

Max number of frame simulated It is the maximum number of simulated frames if not enough error occurs.

Eb/N_0 It states the minimum SNR, the increase step and the maximum SNR.

lfsr_seed First number it is the seed for the random noise generating function of uncoded bits, the second parameter specify which random function should be used for the generation. '0' calls random generation obtained through the use of C primitive *rand*(), while '1' is used to invoke a Box-Müller algorithm implemented for C code.

unif_seed It uses as input parameters equal to previous lfsr_seed. It is used to set the seed for the Gaussian noise evaluation.

intrinsic information fractional bits, minimum and maximum value These parameters are respectively the number of fractional bits used by the decoder, the minimum and the maximum representable value inside the decoding architecture.

extrinsic information minimum and maximum value It is the data representation of values from the channel.

scaling factor for min-sum algorithm It is the normalization factor for the min sum approximation.

Starting frame number It is the number to start resumed simulation, if equal 1 it is a new simulation.

Decoding type method Every decoding scheduling described in the software is associated with a number. This parameter specify which decoder to use.

BPSK=0 or 16-64-256QAM, BICM=0,1->off,on The first parameter select which modulation to use, the second one enables or disables bit interleaved coded modulations (BICM).

So the setting file is a card which describes the simulation that is processed.

4.1.2 Encoding

Once the uncoded data u are generated, the encoding is performed as a generic block code. So the generating matrix G is computed by presented recursive construction, then coded information x is computed as $x^T = G^T u^T$. In order to save time of execution of the code the G^T matrix is computed only before the start of frame iterations. Also to reduce the memory occupation G matrix is deleted and memory freed.
Once the bits are coded they may be interleaved if the BICM option is selected. This functionality emulates practical coding schemes used in real application environments.

4.1.3 Channel simulation and soft information evaluation

For BPSK modulation over AWGN channel, the simulation of Gaussian noise addition is evaluated by 2.21. For QAM symbols the scheme of AWGN channel is

expressed by $Y = X_k + N_k$, where N_k is a complex Gaussian noise with $\mathcal{N}(0, \frac{N_0}{2})$. So it is possible to write the probability of having received Y given that signal $X_k \in S$ was transmitted as

$$P(Y|X_k) = \frac{1}{2\pi\sigma^2} e^{\frac{-|Y - X_k|^2}{2\sigma^2}} \tag{4.1}$$

and consequently

$$\begin{aligned} f(Y|X_k) &= e^{\frac{-|Y - X_k|^2}{2\sigma^2}} \\ \ln f(Y|X_k) &= \frac{-|Y - X_k|^2}{2\sigma^2} = \frac{-E_S|Y - X_k|^2}{N_0} \end{aligned} \tag{4.2}$$

where the average energy per symbol is $E\{|x|^2\} = E_S$. As can be seen in Figure 4.2 the coded frame is mapped into a list of signals. The list of symbols of the

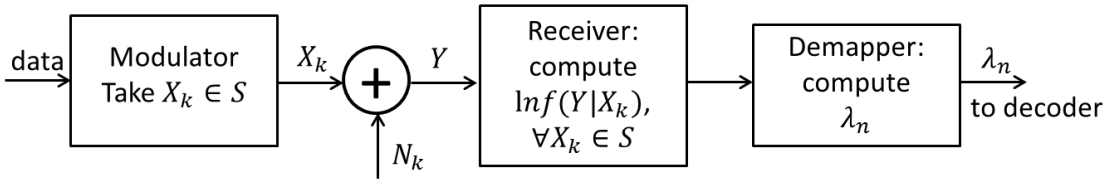


Figure 4.2: *Structure of LLRs message allocation for C model.*

frame is sent over the AWGN channel, so for each signal in S, the receiver computes $f(Y|X_k)$. The LLR estimation on n-th bit is given by 4.3.

$$\begin{aligned} \lambda_n &= \ln \frac{P(d_n = 0)}{P(d_n = 1)} = \ln \frac{\sum\limits_{X_k \in S_n^{(0)}} f(Y|X_k)}{\sum\limits_{X_k \in S_n^{(1)}} f(Y|X_k)} \\ &= \ln \sum_{X_k \in S_n^{(0)}} f(Y|X_k) - \ln \sum_{X_k \in S_n^{(1)}} f(Y|X_k) \\ &= \max_{X_k \in S_n^{(0)}}{}^{*} (f(Y|X_k)) - \max_{X_k \in S_n^{(1)}}{}^{*} (f(Y|X_k)) \end{aligned} \tag{4.3}$$

where $S_n^{(0)}$ represents the set of symbols with n-th bit equal to 0 and $S_n^{(1)}$ is the set of symbols with n-th bit equal to 1. In Figure 4.3 an example of $S_3^{(1)}$ is given.

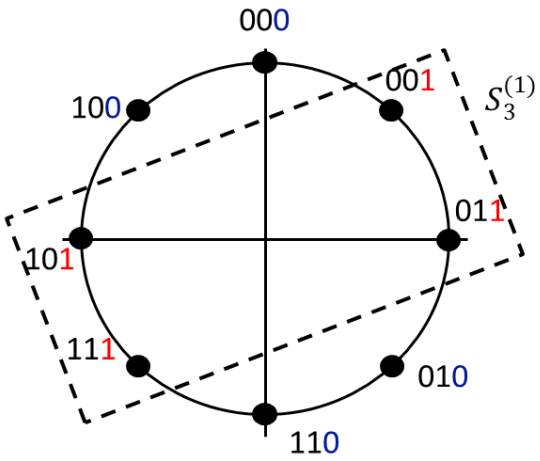


Figure 4.3: *Example of* $S_3^{(1)}$.

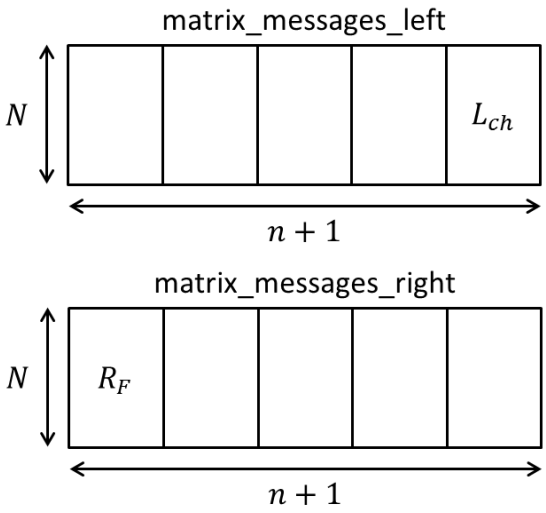


Figure 4.4: *Structure of LLRs message allocation for C model.*

4.1.4 Graph description and Decoding

The LLR messages that are computed during the simulation are stored into two memory of size Nx$(n+1)$. Each matrix is used to store only left-going or right-going messages. In Figure 4.4 is depicted the memory representation, it is also possible to see the initialization of memories according to channel LLR estimations for $matrix\_messages\_left$ and of a-priori information R_F in $matrix\_messages\_right$. All other cells are set to zero at start of decoding.

This memory representation stores the update messages in the same order of the graph description for polar codes as in Figure 3.9. To reduce the complexity of the code, the final version of the simulation software does not implement explicit node interconnections, but it exploits the uniform structure of the decoder. The reverse-shuffle function called *sorter* and its inverse $back - sorter$ are used to appropriately remap data of the columns interested. The scheduling algorithm is defined with a structure composed of one integer and six pointers.

```
typedef struct generic_fun_struct {int tot;
                                   int *vector_col;
                                   int *vector_upd;
                                   int *calc_left;
                                   int *calc_right;
                                   int *left_temp;
                                   int *right_temp;
                                  } generic_f;
```

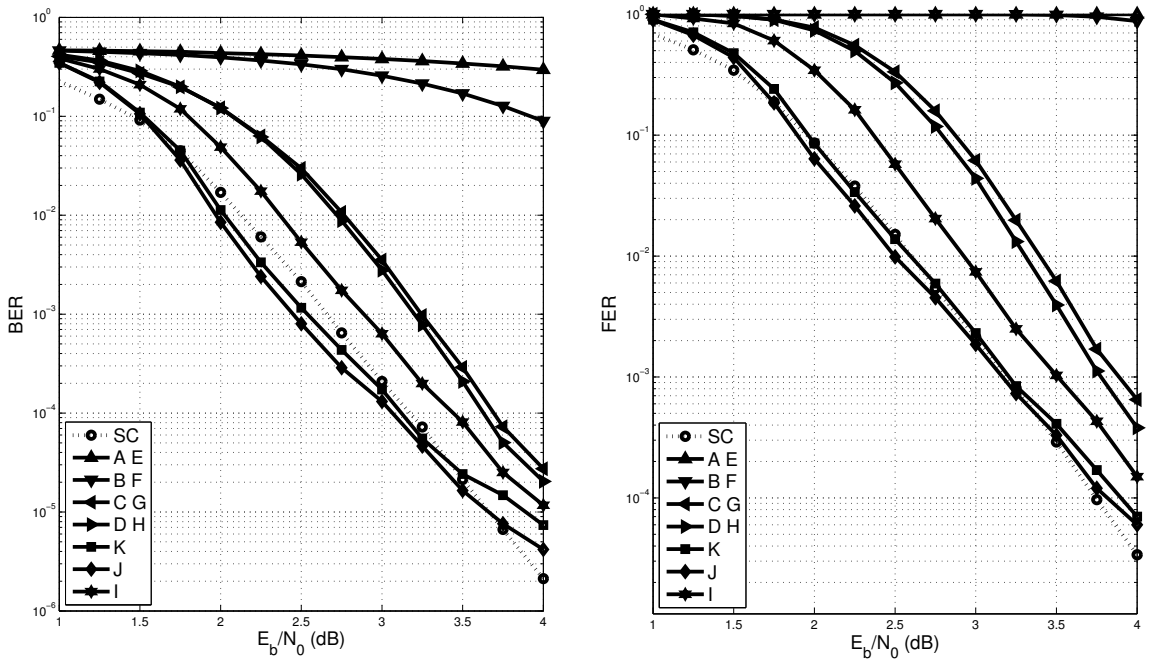
The integer *tot* defines the total number of elements in the $vector\_col$, which is the ordered list of the stages that are executed by the SA. The $calc\_left$ and $calc\_right$ store for the stage pointed by $vector\_col$ if it must be performed left propagation or right propagation (or both). $left\_temp$ and $right\_temp$ are pointers to two memory locations of the same size of $matrix\_messages\_left$. Before any operation on each stage, data from $matrix\_messages\_left$ and $matrix\_messages\_right$ are copied into these memories. The results of propagated messages are stored inside these support memories. Only when $vector\_upd$ has a 1 in the corresponding position of the column, after evaluation of propagation messages, the results in the support memories are copied back to correspondent $matrix\_messages\_left$ and $matrix\_messages\_right$. This solution allows to simulate hardware behaviour when on the same clock multiple operations are performed.

4.2 Simulation Results

Computed results in terms of BER and FER for every step of iteration are saved into the result text file. With simple Matlab scripts, this file can be elaborated and plotted to obtain following curves. On practical point of view, to design and compare decoder architectures, the favourite representation is given by FER results which defines if the packet received was completely decoded correctly or not.
Since the length of the iteration is different among Scheduling Algorithms(SAs), to obtain an information of performance given a certain fixed time is possible to assign a given number of states of the flow graph. In Figure 4.5 a comparison among presented SAs for same number of states is given. It is possible to see that most

performing scheduling at the same states condition are J, followed by K and I, which outperform also SC decoding.
Comparing scheduling to a fixed number of states, does not give a reliable comparison because, as can see in their flow chart, the number of operation which compose a state may be different.

To obtain an information of the speed of convergence for a scheduling instead of



(a) (b)

Figure 4.5: Performance for BP decoding with different schedules, BPSK modulation, floating point decoding, rate $\frac{1}{2}$, codeword length $N = 1024$, 150 states a)BER. b)FER.

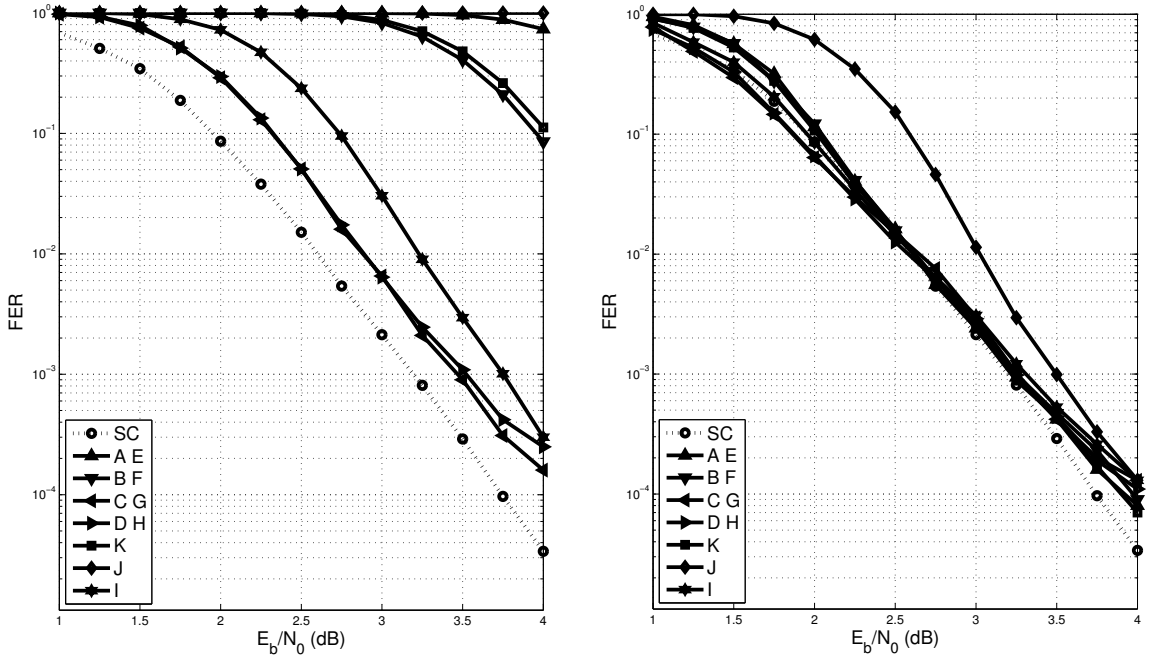
consider the number of state, is possible to consider the number of operations to achieve performance. The result anticipated in Figure 3.17 reports FER results for BPSK modulation over an AWGN channel with floating point precision of decoding and 200 cycles of maximum decoding operations. It is possible to compare this result with finite precision result for a 9 bit quantized messages of the final architecture of Figure 4.6a. This comparison shows that the precision chosen does not modify the curves plotted. This quantization parameter has been chosen for the final hardware implementation, however by decreasing the number of bits and accepting a small loss to decoding performance it is possible to reduce the area of final hardware implementation.

A deep investigation has been made to point out properties to compare these scheduling. Main effect analyzed are listed below:

- differences among scheduling policies tend to increase when the block size is increased;
- differences among scheduling policies tend to increase when the code rate is reduced;
- differences among scheduling policies tend to increase when the number of iterations is reduced.

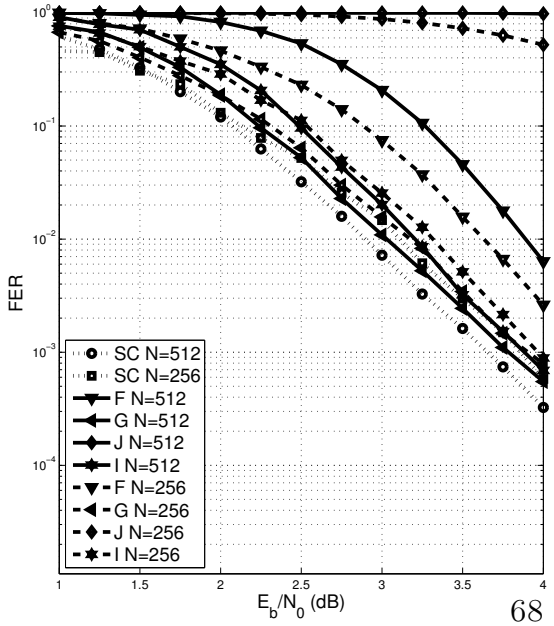
As can be seen observing first and second graph of Figure 4.6, the effect of different scheduling is present both at an high number of operations, that can be considered the convergence of the decoding (650 operations), and at an early number of iteration (200 operations). The effect of scheduling becomes more evident as the decoding time is reduced and it is appreciable from comparing these two graphs. In the third graph of Figure 4.6 the codeword length N directly affects Frame Error Rate (FER) in concordance with theory, but the scheduling effect is also present. Comparing G to I scheduling, difference are small and vary from about 0,3dB for $N = 256$ to 0.4dB for N = 1024. While for scheduling comparison like I to F difference are of about 1dB, in the same range.
Performance related to SA also depend on rate. As shown in the fourth graph of Figure 4.6, the decrease of the rate offer better performance compliant with theory. For the case of small decoding time, as before, the difference between G and I becomes more and more significant as the code rate reduces. In the presented figure, rate $\frac{3}{4}$ gives a difference of about 0.5dB that becomes 2.5dB for rate $\frac{1}{4}$. The reduction of the rate is however not always feasible in practical application because it reduces the effective information sent every frame. In the first to third graph of Figure 4.7, effects of channel modulation, for Additive White Gaussian Noise (AWGN) channel and Quadrature Amplitude Modulation (16-QAM, 64-QAM, 256-QAM) on all scheduling and SC decoding algorithm with min-sum approximation are analyzed. All QAM modulation use Gray code bit assignment. The effect of the modulation on channel is that differences among scheduling policies tend to slightly reduce when the order of modulation is increased. For example considering G to I scheduling, the gain to move from 16-QAM to 256QAM is about 0.25dB.

Figure 4.6: FER performance for BP decoding with different schedules, BPSK modulation, 200 operations. a)finite precision decoding (9 bit), rate $\frac{1}{2}$, codeword length $N = 1024$. b)floating point decoding, rate $\frac{1}{2}$, codeword length $N = 1024$, 650 operations. c)floating point decoding, rate $\frac{1}{2}$, variable codeword length ($N = 512,\ 256$). d)floating point decoding, variable rate ($\frac{1}{4}$, $\frac{3}{4}$), codeword length $N = 1024$.

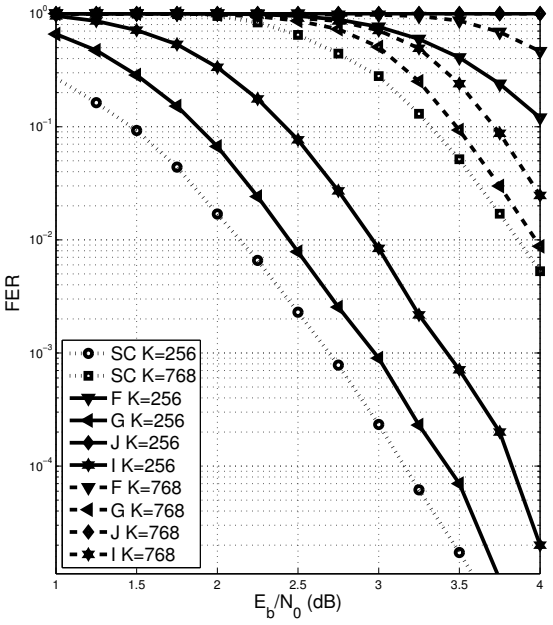


(a)

(b)

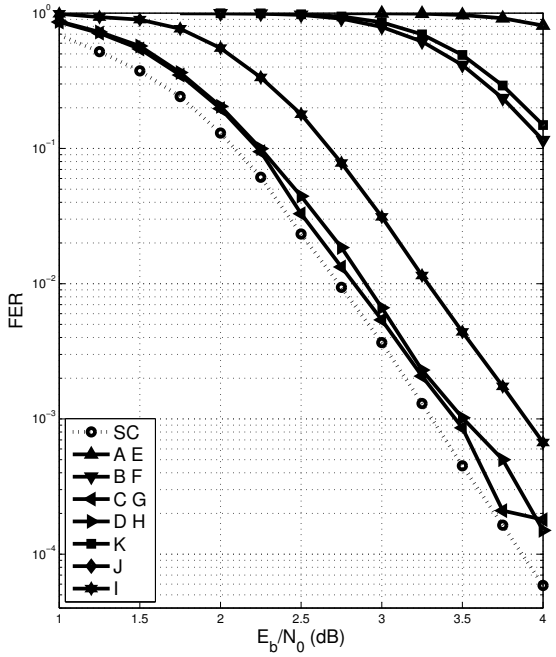


(c)

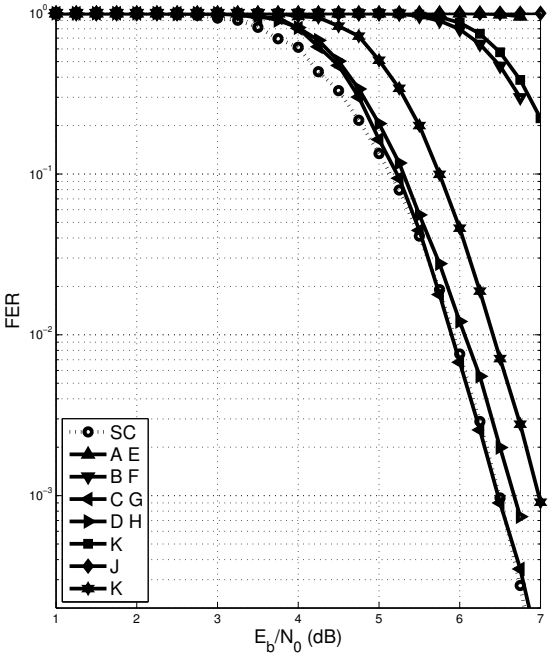


(d)

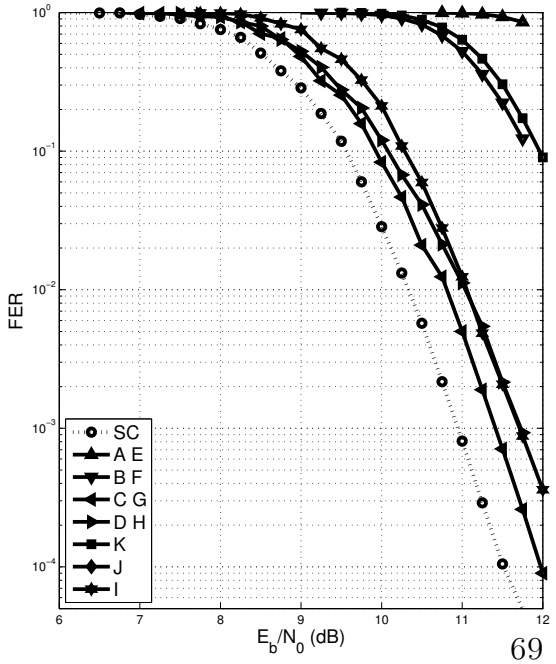
Figure 4.7: FER performance for BP decoding with different schedules, floating point decoding, rate $\frac{1}{2}$, codeword length $N = 1024$. a) 200 operations, 16-QAM modulation. c) 200 operations, 64-QAM modulation. d) 200 operations, 256-QAM modulation.



(a)



(b)



(c)

Chapter 5

State of art for polar codes decoder implementations

Initial description of equations and feasibility of Belief Propagation (BP) decoding was given in [34] in 2010. This article lead the way to different BP architectures in the following years. Here is presented a review of published implemented solutions to give a research context background. At the end of this chapter Table 5.1 summarize main performance results and hardware costs.

2011

First BP decoding architecture attempt found in literature implemented BP algorithm on an FPGA.[40] This implementation has been created with the aim to give maximum flexibility in term of possible decoding graphs. The use of FPGA allows trading the number of processing units with the control part by reprogramming it so achievable throughput change according to these choices. The code rate can instead be changed at runtime, by setting correct frozen data. By adding an extra address generator it is possible to decode also graphs smaller than maximum codeword length N without reprogramming. Used FPGA is a Xilinx XC5VLX85, which allows to implement maximum block size of $N = 8192$, with 16 PEs. Highest reported throughput rate is $53Mbps$ for a (256,192) polar code and 5 iterations, however decoding performance results are given for 50 iterations.

2012

In [41] is presented a GPU implementation to explore the parallel computing capability of this solution. The NVIDIA Fermi Architecture is used, to prove the feasibility of implementing a BP decoder and exploit multiprocessors, on chip registers, shared and global memory to efficiently improve the occupancy of available resources for

parallel computing. Operation used are in floating point base and computed messages are likelihood ratios (LRs). Parallel threading programming handle processors to evaluate LR update equations, data are exchanged among each other through the global memory, the address related to data also need to be interchanged in order to perform shuffle and its inverse among data. The shared memory (faster than the previous one) is used to copy left and right messages and perform operation. At the end of the iteration, results are updated in the global memory. The used card is an NVIDIA GTX 560 Ti, with 384 cores with working frequency of 1.66GHz, which is presented to implement up to block size $N = 2048$, and for a small block length $N = 256$ it reach 57.20Mbps of throughput at 10 iterations.

2013

As described in Section 3.8, in [3] was first suggested to use min-sum approximation for BP polar decoding. It was also proposed a particular PE (divided in two parts, for upper and lower messages) in which data representations are changed between sign-magnitude to two's complement to perform additions and then converted in sign and magnitude for comparison, scaling and as output. It is evaluated that for the introduction of the scaling block, the overall path delay is augmented by about $\frac{1}{5}$. A critical path reduction is introduced by merging sum modulus and the two conversion modulus inside the PE. The overall BP architecture is then used as a systolic architecture by inserting between each stage a pipeline register to store update messages. Only one stage is activated at a time, so during every iteration $n - 1$ stages are not working, where $N = 2^n$ is the codeword length.

2014

In June 2014, the work of Park was announced in [36]. It presented a BP parallel decoder architecture that at 4dB achieve 4.68Gbps of "coded throughput", which means that no frozen bits are used for the transmission. The main improvement was the reduction of unnecessary operation for what is called scheduling C in this work so obtaining scheduling G (more details in Chapter 6.1).[37] The obtained architecture needs 45kb of memory to store data and it is composed by a pipelined double stage in order to mitigate the memory access delay to data. The structure also implements an early termination rule of 3 successive agreement on hard decoding decision to achieve declared performance as coded throughput (P1). Without the early termination the maximum coded throughput is 2.02Gbps, to decode a codeword length $N = 1024$ at nominal supply of $1V$. This architecture has been fabricated with TSMC 65nm CMOS process in a chip, where the decoder core area occupy $1.8mm$ x $0.82mm$, relevant measurements of the chip characteristics are given in Table 5.1, where also a low frequency low voltage condition is reported (P2).

The architecture described in [42], aim to reduce the complexity of BP decoder by eliminating calculation of frozen bit LLRs. So the resulting architecture is strictly related to the specific polar code implemented. Four different type of PE are implemented, to take care of all possible case of presence of frozen data. This simplified BP decoder reduce the number of operation (sums and probability update) of about $20 - 25\%$ for code blocklengths ranging from $N = 128$ to $N = 32768$.

The work in [43] optimize their previous architecture [3] with the classical analysis of operation dependency over time, retiming and pipelining for different decoding messages. This approach allows to increase the hardware utilization up to 100%. For the initial implementation, four solution are presented:
-*Overlapping at iteration level*, that means to reorder operation according to data dependencies to reduce latency. It is done by filling unused PEs stages with following task (rescheduling).
-*Overlapping at codeword level*, that is common message pipelining. This choice force to implement n times the initial memory of the architecture where the codeword length is $N = 2^n$, to store all messages.
-*Joint overlapping of codeword and iteration level*, that combines previous approaches. The overlapping at iteration level is pipelined, this reduces the total amount of needed memory to just double than initial architecture.
-*Folded architecture*, by implementing the architecture only with a fixed number of stages. This folding factor can grow from 1 (P1)to $n/2$ (P2), where the code block length is $N = 2^n$. Furthermore to decrease latency is used overlapping at iteration level (P2).

The introduction of early stopping criteria for BP decoding have been studied in [44]. Two possible solutions are presented, the first one evaluates estimation of uncoded $\hat{u}$ and coded $\hat{x}$ bits, and use the generating matrix G of polar codes to verify if $\hat{x} = \hat{u}G$. This method is called G-matrix-based detection type. The second method is named $(minLLR)$-based detection type, it define the decoding process complete if the minimum absolute value among all LLR is above a certain threshold. This method also uses an estimation to the channel condition in order to set the correct value of the threshold. It evaluates the Hamming distance between $\hat{x}$ and $\hat{u}G$ to compute this estimation. The implemented architecture with first early stopping criterion (P1), is not influenced in terms of critical path (PE has bigger critical path for codes with common lengths $N \leq 10000$) while for the second criterion (P2), a pipeline stage is necessary to reduce the critical path of the Hamming distance measurement block for high speed designs. G-matrix solution can reduce the number of iterations up to 42.5%, while adaptive $(minLLR)$-based reach 23.2% at 3.5dB. The

base BP decoder solution remains unchanged as in [43].

In [45] is proposed an hybrid Belief Propagation-Successive Cancellation decoder. The main idea is to use a the BP decoder with early termination of [44] and if the codeword is not decoded after the maximum number of 60 iterations, a SC decoding is performed on updated LLRs from the BP decoder. It is shown for the case study of a (1024,512) polar code that this hybrid solution shows performance lower than BP decoder of about 0.2dB in the entire SNR region. Compared to SC decoder, for medium SNR (around 2.5dB) the hybrid decoding reach 0.2dB of gain, and for higher and lower SNR the behaviour converge to SC performance (that for high SNR is better than BP). From the hardware point of view the SC and BP computation are carried out by same PEs, because BP decoding is a generalized case of SC, so with the proper input signals and the introduction of some multiplexer the BP PE has been generalized.

2015

To reduce the amount of total memory used to implement polar codes BP decoders, the work in [46] propose to combine four adjacent 2 by 2 PEs of the polar code graph into one 4 by 4 block suitably interconnected. By doing so the total amount of memory and stages is halved while the critical path is increased from $2.05ns$ of typical solution (implemented by authors) to $5.07ns$. The decoder area at growing codeword length becomes advantageous after $N = 1024$, in fact as $N = 2^{12}$,2^{14},2^{16},2^{18}, the area is reduced by 18.4%,45.5%,51.5%,52.7% compared to typical implementation. This solution is interesting because works in the region where polar codes perform better. The drawback for presented results is that the throughput ratio between this solution and conventional solution also decrease with N due to the critical path.

The architecture presented in [47] implements a Soft-output Cancellation (SCAN) decoder, it uses polar BP update equation to decode the trellis graph in the specific SC order. This solution force the exploration of the graph as a binary tree, but this exploration is iterated multiple times as in BP. This choice of scheduling allows to reduce the overall memory form $N(n+1)$ of the classical trellis graph to $\frac{nN}{2}+\frac{N}{2}-2$ as presented in [48] for the codeword length $N = 2^n$. This architecture has been implemented on XC5VLX85 FPGA device with a reduced number of PE. Given decoding performance are compared with lower than average performance BP decoder, but with normal BP decoding are comparable or worst. Obtained throughput is comparable to [40] FPGA implementation. The number of implemented Look Up Tables grows linearly from +18% for small $N = 256$ up to +43% for $N = 8192$. However for same explored cases it reduces the number of Flip-flops of approximately one

order of magnitude.

To reduce the energy dissipation of BP decoding and to mitigate the required latency introduced by multiple iterations, in [49] is presented a method called sub-factor-graph freezing, which avoids to calculate updates for already converged portions of the graphs during subsequent iterations. The presented decoding method offers very small improvement (less than 0.1dB) compared with classical BP decoding. Through this method is therefore possible to reduce the average number of iterations of about $40-46\%$ if compared to generic BP decoder as in [34] and 17% if G-matrix-based method is implemented as in [44] at SNR of 3dB. The lowered average number of computations grants to lower power consumption and it is estimated by 65% and $30-46\%$ respectively for same comparisons. An effective hardware implementation of this method is still not existing in literature.

In [50] a modified PE is introduced; the evaluation of left-propagating upper LLR message is evaluated only through left-going LLRs data (except for last stage). The other equations for polar codes BP probability updates remain unchanged. This simplification reduce the amount of total LLRs message to be stored to $5N-3$. The graphs exploration evolves like SCAN algorithm and presents similar performance in terms of Frame error rate with negligible degradation. The implemented decoder area is about 66% smaller than [36] but the compared throughput is about 20%. With comparison at [44], the decoder dimensions are reduced by 83% while the throughput is 11% of the other architecture.

The proposed architecture in [51] exploits the simplification of PE nodes, so four new types of PE are used: -*All frozen node*, for the part directly connected to frozen leaf, it is not necessary to compute LLRs.
-*All information node*, all leaf node are information. The right propagating LLRs have 0 information and it will not be updated so they can be removed.
-*Repetition node*, where there is only a single information leaf on the last node. The information bit is just copied multiple times, so it is possible to avoid the message passing in multiple stages.
-*Single parity check node*, where there is only a single leaf frozen node. It can be substituted by a single parity check node to evaluate belief information.
It is used the proposed scheduling of [36] and the general early termination rule for block codes given by the equation $\hat{x}H = 0$ where H is the parity matrix and $\hat{x}$ is the hard estimation of the LLRs information updated at right-most of the decoding graph. Form both Frame error rate performance and average number of iteration for decoding points of view it is proved that the architecture in [44] and the proposed one have practical same performance. The reduction of computation of the proposed architecture due to the nodes is instead up to 40% for low SNR compared to [36].

Table 5.1: *Implemented BP architectures in literature.*

Property	Unit	[40]		[41]		[3]	[36]P1	[36]P2	[43]P1	[43]P2
Decoding type						A	G		A	
Block length		1024	256	2048	256	1024	1024		1024	
Rate		0.5	0.75	0.5		0.5			0.5	
Technology		FPGA		GPU			CMOS		CMOS	
Process	nm						65		45	
Core area	mm^2						1.476			
Supply	V						1	0.475		
Frequency	MHz	160		1660			300	50	500	
Power	mW						477.5	18.6		
Iterations		5		35	10		6.57[c]	6.57[c]	60	
Throughput	Mb/s	27.83	53.33	1.23	57.20	37	4676[a]	779.3[a]	426	2000
Energy efficiency	pJ/bit						102.1	23.8		
Energy eff. per iter.	pJ/b/iter						15.54	3.63		
Area efficiency	$Mb/s/mm^2$						3168	528.0		

Property	Unit	[44]	[44]P1	[44]P2	[46]	[47]	[50]	
Decoding type			A		C	SCAN	SCAN	
Block length			1024		1024	1024	1024	32768
Rate			0.5			0.5	0.5	0.9
Technology			CMOS		CMOS	FPGA	CMOS	CMOS
Process	nm		45		45		90	90
Core area	mm^2				0.747		0.97	4.734
Supply	V							
Frequency	MHz		500		197	90	571	518
Power	mW							
Iterations		40	23[b]	30.7[b]	15	5	1[b]	1[b]
Throughput	Mb/s	2900	4500	3500	1683[a]	17.56	958[a]	2208[a]
Energy efficiency	pJ/bit	328	220	291				
Energy eff. per iter.	pJ/b/iter							
Area efficiency	$Mb/s/mm^2$						987	466

[a]Coded throughput; [b]Early termination at 3.5dB; [c]Early termination at 4dB

Chapter 6

Belief Propagation Decoder hardware implementation

This chapter describes developed architectures for the evaluation of hardware implementations performance.
Top down approach is used to present designed structures. An initial decision that has been taken for the developed designs is to avoid to constrain the architectures as in [42],[51]. These solutions allows to reduce hardware complexity and increase working frequency by focusing on a specific target code and consequently optimize the hardware, but obtained results lose the capability to decode different polar codes. In fact the frozen bit information is fixed in hardware and cannot be changed while working. For practical application this force to redesign the decoder architecture if polar codes improvements are discovered. Moreover this decision force to implement in the final application a specific decoder for every single polar code used for transmission. This lack of flexibility has been judged unacceptable for the develop of real exploitable designs.

6.1 Effects of scheduling algorithms on architectures

Scheduling Algorithms (SA) directly affects properties of implemented architectures. From parallelism point of view, the architecture design may be classified in

- Fully serial,
- Partially parallel,
- Fully parallel.

Serial implementation is clearly the simplest and compact but the limited computational resource cause to be the slowest. This solution could be sufficient to grant

moderate throughput requirements. On the contrary the fully parallel architecture can reach highest speed but it may face two fundamental problems:

- Lack of flexibility;
- Conflicts in memory access.

For the specific purpose of studying achievable performance of polar codes decoders, fully parallel implementations are the most liked because they do not penalize throughput or increase latency, however if dimensions of codes becomes too large it is necessary to realize partially parallel architectures.
Previous consideration does not affect the possibility to reduce hardware structure due to different scheduling while achieving maximum performance.
Following descriptions on SAs are related to the ones that have been implemented

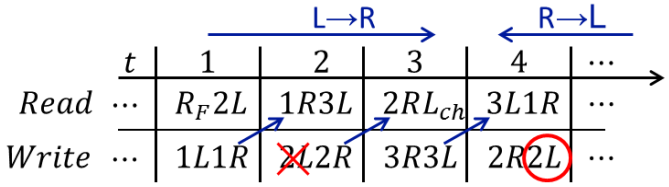


Figure 6.1: *Scheduling C message update example.*

in an hardware design. For scheduling C, if all update messages are observed, when Left to Right ($L \rightarrow R$) bidirectional updates are performed, evaluated left propagating messages are then rewritten at Right to Left ($R \rightarrow L$) stage updates, so the first update is lost without having been used. Generalizing this consideration allows to move from scheduling C to scheduling G without loss of performance. Moreover the PE will only need to calculate half of previous information, becoming unidirectional.
In Figure 6.1 the time evolution of updates is presented for a 4 state C scheduling (or equivalently a 3 stage decoder) with same name convention given in Section 3.5 (R_F are a-priori information and L_{ch} are channel LLRs). It is possible to see that in this example the LLR $2L$ is calculated in ($L \rightarrow R$) but never used, so it is possible to avoid to save that value in memory. Moreover the updated information is used only in the following time period, this observation grants that it is possible to replace these messages in the data storing block of the implemented design.
These properties are exploited to reduce hardware components in the final architecture. Identical considerations lead from scheduling D to H. The architecture which implements previous observations has been called Reduced Complexity Architecture (RC).
By similar considerations to implement scheduling I is possible to simplify the required hardware without loss of performance. Only two operation are needed each

state, this will be executed by Bidirectional Wave Architecture (BI-W).
To implement J scheduling the number of stages must be equal to the number of operation needed in one state. It can be considered as the upper bound of implemented resources for previous SAs. For this algorithm Fully Parallel (FP) Architecture will be used.

6.2 Hardware Implementation

6.2.1 Main Entity

Fully Parallel Architecture

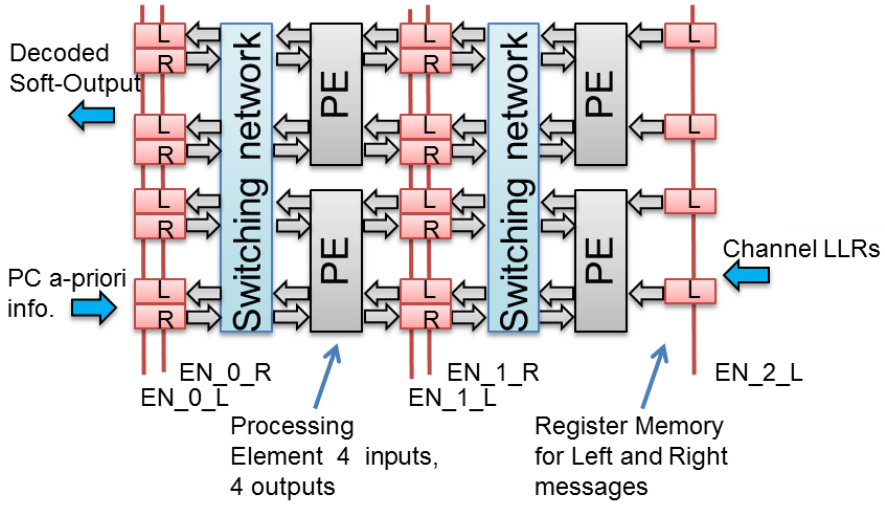


Figure 6.2: *Fully parallel architecture for a code of length* $N = 4$.

Conceptually, the simplest approach to implement the BP decoder is to map the factor graph of Figure 3.9 directly to hardware as shown in Figure 6.2. This fully parallel (FP) architecture uses $\frac{N}{2}\log_2(N)$ PEs. A supplementary feature in Figure 6.2 is the presence of *switching networks*, which are circuits for implementing certain permutation operations, such as *perfect shuffle* as described in 3.6. The decoder contains banks of memory registers. Those registers marked "L" hold left-going LLR messages, while those marked "R" hold right-going LLR messages. The R-registers at level 1 are initialized in accordance with (3.68) and the L-registers at level $n + 1$ are initialized in accordance with (3.67). All remaining registers are initialized to zero. Note that there is no need to have any R registers at the channel

level $n+1$. Following the initialization, the registers are updated in accordance with formulas (3.62)–(3.65) and in accordance with an update schedule. In order to conserve energy, the update schedule should avoid executing an update when there is no change in the messages coming from the neighbours.
This architecture opportunely controlled is able to perform any scheduling algorithm by construction, moreover a-priori information of inner memory register may be preloaded to save computations and reduce dynamic energy consumption.

Reduced Complexity Architecture

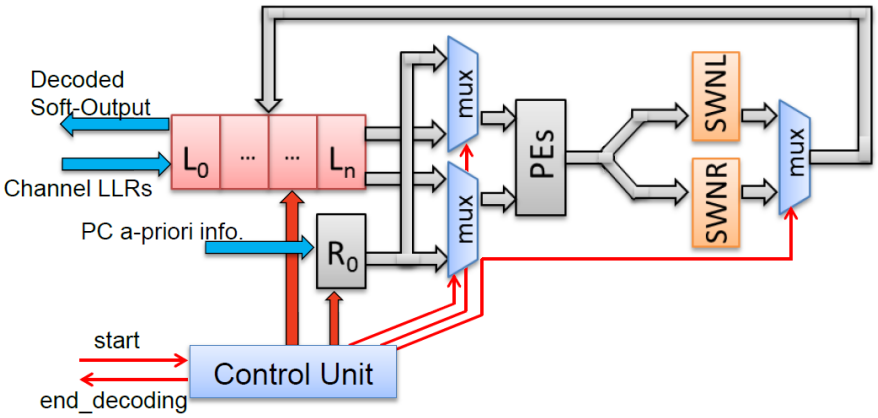


Figure 6.3: *Reduced complexity architecture.*

The reduced complexity (RC) architecture depicted in Figure 6.3 is composed of a single stage of PEs that receive the LLR values either from a register memory block or from a RAM (or ROM if the polar code is fixed) memory R_0 at the start of each iteration. R_0 stores the a-priori information of frozen bits, the memory block stores $(n+1) \times (N)$ LLR messages. The information from the channel is stored in the rightmost L_n memory column, the final soft output is stored in the leftmost memory column L_0 and the other memories are used to save temporary information of both left and right messages. This kind of architecture memory reduction has been proposed earlier in [37]. The switching networks SWNL and SWNR permute the LLR messages back and forth as $reverse - shuffle$ in 3.6. This choice allows to store LLR messages from channel in the memory block in their original order. The control unit is deputed to send correct signal and addresses in order to select correct outputs and store data updates.
To save initial a-priori computations, fixed results (allowed by the implemented G scheduling) are preloaded to register memory block.

Bidirectional Wave Architecture

The Bidirectional Wave Architecture (BI-W) can be seen as an extension of previous RC architecture. As can be seen in Figure 6.4 the number of processing PE stages

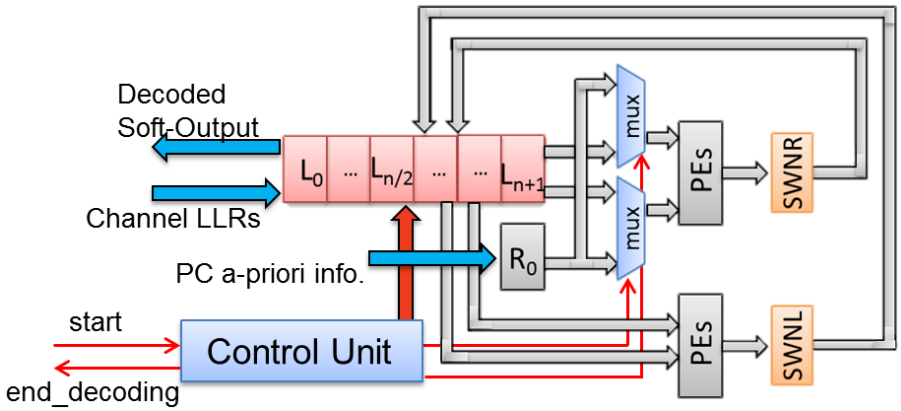


Figure 6.4: *Bidirectional wave architecture.*

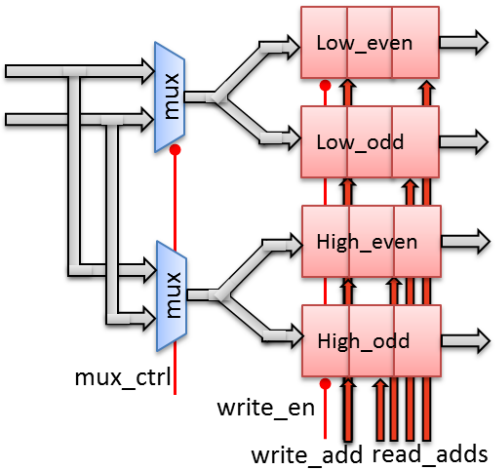


Figure 6.5: *Bidirectional wave register memory block detail.*

is doubled, so it is possible to calculate two LLR set update in parallel. According to observations on I scheduling the register memory bock has size $(n+2) \times (N)$. As can be seen from Figure 6.5 assigning a specific memory organization, by separating even and odd index messages of the lower or upper half, it is possible to reduce the hardware complexity to four single double port memory register accordingly interconnected. This simplification is possible due to message order specified by

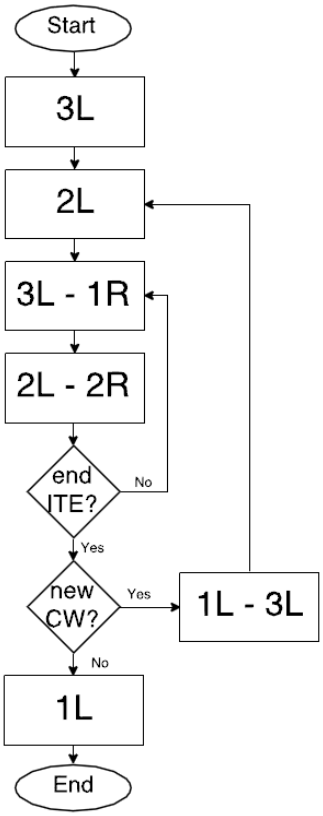


Figure 6.6: *Bidirectional wave architecture refined I scheduling.*

I scheduling. R_0 RAM memory and switching networks SWNL and SWNR are unchanged. To enhance performance of the final design a modified I scheduling has effectively been implemented, as reported in Figure 6.6. From computation point of view no changes are inserted in the new presented scheduling, however this new flow map explicitly point out effective operations. Initial iR messages are precomputed and loaded in the correct memory positions for a-priori LLRs, so at first iteration only half PE stages are used. At the last iteration, only the left-going updates for last second half of operations of the iteration are necessary to compute final estimated decoded message. These observations combined allows to start a new codeword decoding if available at the last iteration of the previous one saving half of total number of operations for first iteration.

6.2.2 Processing Element

The basic PE for the proposed BP decoding architectures is a component that has four inputs and four outputs, implementing two sets of belief updates: a left-wave of updates given by (3.62) and (3.63) and a right update given by (3.64) and (3.65). One implementation option is to define a *full* PE that can carry out the left- and

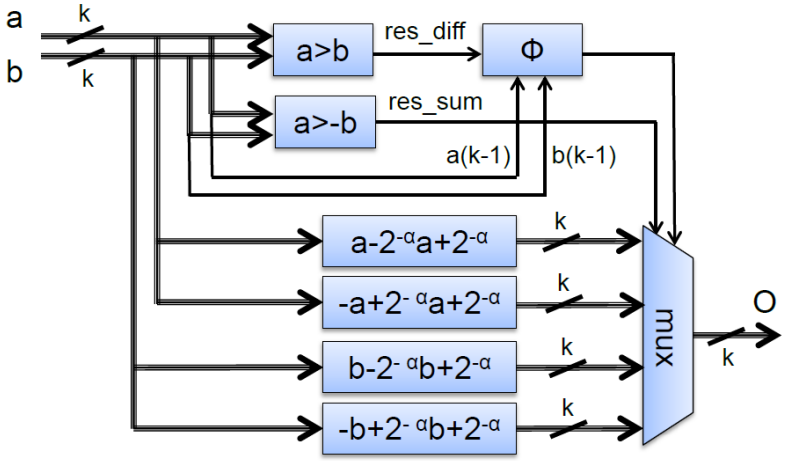


Figure 6.7: *Normalized min-sum block for two's complement data representation.*

right-wave updates simultaneously. One may conserve hardware by noting the congruent nature of the two updates and defining a *half* PE that can carry out either the left or the right update at any given time.
With proper scheduling this half-size PE does not increase latency while it reduces complexity significantly. The normalized min-sum approximation is used to evaluate part of the four outputs of the general PE component. For higher clock speed, every half-size PE is implemented using sign and modulus representation. This result has been achieved comparing two's complement (2'c) and sign and modulus (SM) solutions.

Since in literature the only detailed architecture structure of polar code PEs is given in [3], presentation of both 2'c and SM basic PE component is given to point out architectural improvements and prove how synthesis results have been achieved, in particular no conversion between 2'c and SM are performed in the same architecture, saving conversion blocks delays.
For the 2'c implementation, the sum is the classical sum block, and it is necessary a min-sum component specialized for 2'c. The min-sum component presented in Figure 6.7, has been designed to present the best performance achievable in terms of minimum combinatory delay.
The basic structure is made by four sum stages which evaluates $(1-2^{-\alpha})a$, $(1-2^{-\alpha})b$

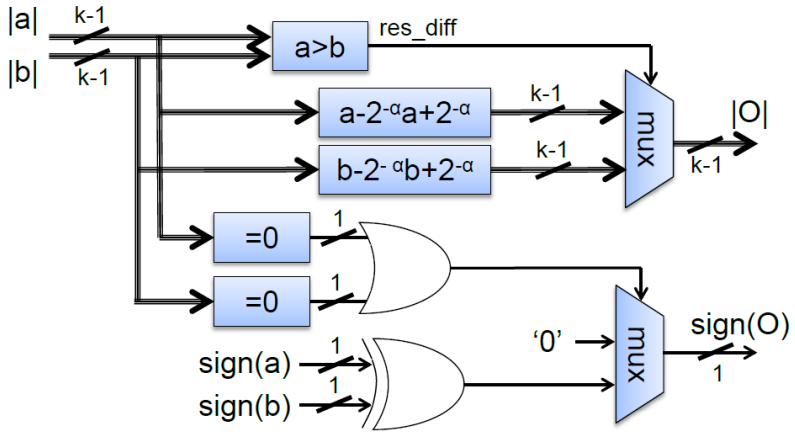


Figure 6.8: *Normalized min-sum block for modulus and sign data representation.*

or their opposite values in parallel. A multiplexer selects the correct output from each stage. Signals res_sum and res_dif are the carry/remainder of the sum or subtraction operation on $k+1$ bits of a and b; Φ is computed as $\Phi = a_{k-1}\overline{b_{k-1}} + \overline{b_{k-1}}(a > b)_k + a_{k-1}(a > b)_k$.
The normalization used is $(1 - 2^{-\alpha}) = 0.9375$ so the scaling operation can be implemented with a simple shift-addition circuit as in [3]. In order to reduce rounding errors for truncation, in sum stage α decimal bits are used to represent data and an additional $2^{-\alpha}$ is summed to approximate the round function. This approximation work as the real round for positive numbers, while for negative values, only for -0.5 decimal, result is rounded up instead of down, and for all other cases round function is preserved. This relaxation allows to just sum $2^{-\alpha}$ as approximation both for positive and negative 2'c numbers to obtain the correct behaviour.

The sign and modulus (SM) solution allows a simple comparison of the modulus to implement the min-sum approximation. As can be seen in Figure 6.8, the min-sum component reduces to a single comparison block, which directly computes the sign of difference between the two modulus, a multiplexer to select the smallest and normalization blocks, similar to 2'c case of normalization of two inputs (in this case modulus are always considered as positive inputs). For the generation of the sign output an xor gate evaluates the sign of the solution except for the case that at least one of the two inputs is a zero, that is checked by a simple comparison to zero module and the correct output is selected through a multiplexer.
The sum modulus for the case SM becomes more complex. As represented in Figure 6.9, the simple sum component of 2'c case is replaced by a sum and two subtractor modules, to perform all possible combination of operation between the two input. This component decompose operation $A+B$ in all possible combinations to evaluate

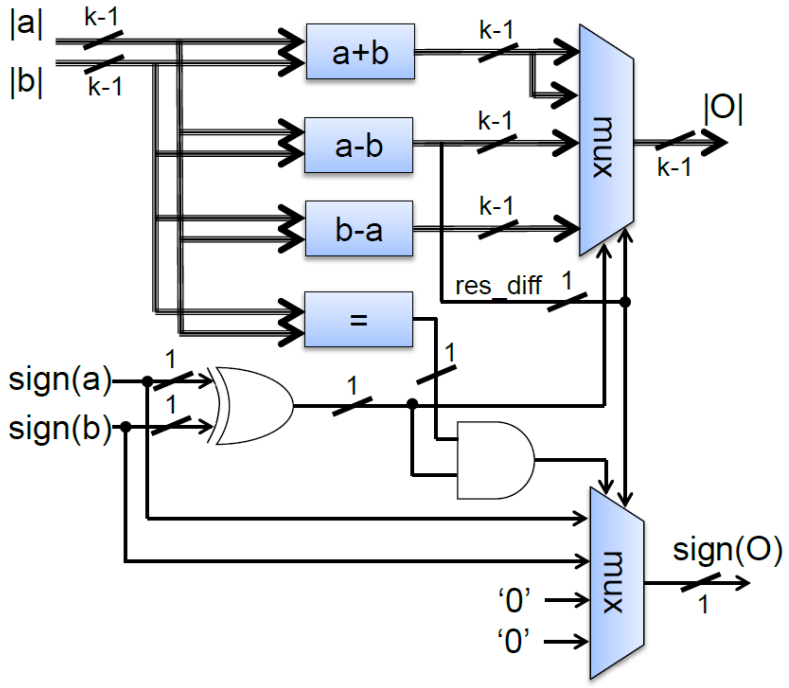


Figure 6.9: *Sum block for modulus and sign data representation.*

the modulus output as:

$$|O| = \begin{cases} |A| + |B| & if\ (A > 0)(B > 0) + (A < 0)(B < 0) \\ |A| - |B| & if\ (A > B)((A > 0)(B < 0) + (A < 0)(B > 0)) \\ |B| - |A| & if\ (A < B)((A > 0)(B < 0) + (A < 0)(B > 0)) \end{cases}$$

The sign concordance, given by the xor of sign inputs, select the sum module as output of the multiplexer, or the remainder res_dif of first subtractor module output is used to select between the two subtractor modules.
To evaluate the sign of the sum operation, res_dif gives the information of the highest value, and selects accordingly the output on the multiplexer. If both input have same module but different sign, the sign output must be zero. To perform this check a compare module is added and its results goes inside an and gate with xor result of sign inputs to the first control signal of the multiplexer for sign output.

6.3 Synthesis results

Both SM and 2'c solution for PE have been implemented in VHDL, validated and synthesized using Synopsys Design Compiler (version Z-2007.03-SP1) and STMicroelectronics standard cell library for 45 nm CMOS VLSI design. Obtained result for 2'c PE and SM PE are presented in Table 6.1. To the best of our knowledge,

Property	Unit	SM	2'c
Critical Path	ns	0.95	1.05
Combinational Area	μm^2	2873.91	1482.82
Switching power	mW	0.4905	0.2117
Total power	mW	0.8941	0.3829

Table 6.1: Comparison of developed PEs

presented critical path results are the best results compared to available ones in literature. The fast solution proposed uses 10% less time to compute outputs, as drawback the combinational area is increased of 94% also the switching and total power are increased by 131% and 133% for the single PE.
Both structures for PE have been used to implement proposed decoder architectures. The main features of implemented designs are reported in Table 6.2.
As expected, the maximum throughput achievable is obtained by sign and modulus (SM) Fully Parallel (FP) architecture.
The speed-up introduced by selecting the best data representation for both FP and RC architectures in terms of frequency and throughput is only 1%. This phenomenon is attributable to close PE results in terms of critical delay.
Notice that an unexpected result appears, as can be seen from core area computations the synthesis tool performs better both in terms of maximum achievable frequency and core area for larger SM structures of a constant ratio above 11%.
This result has been exploited by implementing only for MS case the Bidirectional Wave architecture (BI-W). For BI-W architecture the throughput is computed for the single codeword decoding. In this worst case condition it is possible to observe that the throughput over area ratio is better than RC solution of about 33%
The energy estimation has been evaluated by propagating the switching activity from input ports with uniform distribution through all internal nets of the architecture. It appears that BI-W uses 15% more energy per bit compared to RC solution.

Property	Unit	FP	RC	FP	RC	BI-W
Data representation		2's complement		sign-modulus		
Decoding Scheduling		BP J	BP G	BP J	BP G	BP
Block length				1024		
Rate				0.5		
CMOS Process	nm			45		
Core area	mm^2	12.46	1.65	10.89	1.48	1.93
Supply	V	1	1	1	1	1
Frequency	MHz	606	555	625	588	571
Power	mW	2056.5	328.4	1846.7	331.0	638.14
Iterations		65	9	65	9	10
Throughput	Mb/s	4773	1754	4923	1858	3215
Energy efficiency	pJ/bit	430.82	187.22	375.11	178.09	198.48
Energy eff. per iter.	pJ/b/iter	6.63	20.80	5.77	19.79	19.85
Area efficiency	$Mb/s/mm^2$	383.10	1063.08	452.14	1253.96	1665.84

Table 6.2: Comparison of developed polar decoders

Chapter 7

Conclusions

7.1 Achieved results

In Table 7.1 designed architectures are compared with main Belief Propagation (BP) solutions available in literature. As expected, the Fully Parallel (FP) architecture achieves the maximum throughput of almost 5Gbps, so about 2 times the best throughput design in literature. The cost of this performance is the occupied area to implement all PEs. To overcome this problem the research has been oriented to optimize the area efficiency parameter. Best two scheduling candidates have been implemented. The Reduced Complexity (RC) architecture reaches comparable performance with existing solutions, while Bidirectional-Wave (BI-W) architecture improves of more than 33% the throughput over area ratio compared to second candidate (and literature). This qualifies BI-W architecture as best decoder as throughput over area design.
From the energy efficiency point of view, achieved results show that RC architecture improves of about 46% the best result in literature while BI-W architecture reduces this gain of only 6%.

7.2 Future work

As future development of the current work many paths are open:

Polar codes algorithmic improvement The developed C software for simulation allows to inspect many aspects of polar codes, in particular the Frozen Bit Set can improve performance decoding without any architectural modification. Frozen bits depend and can be optimized for the considered channel but also for the decoding algorithm. Present used sets come from BEC best set or SC optimized set for AWGN channel at 0 dB of SNR. The C software

could be used as a starting point to optimize via Monte Carlo simulations (or other kind of optimizations) for specific BP decoding and target channel.

Evaluation of early stopping rules Starting from already existing early stopping rules it is left open to implement those rules to enhance maximum achievable throughput for presented architectures by reducing the total number of iteration for good message SNR conditions. Moreover in [44] is claimed that H-matrix-based approach similar to the one used for most block codes (like LDPC codes) cannot be exploited because the update of the channel message $\hat{x}$ to test if $\hat{x}H^T = 0$ where H is the parity matrix. However this statement is incorrect (as confirmed in [51]), in fact it is possible to compute $\hat{x}_j = \ell_{n+1,j} + r_{n+1,j}$, where $1 \leq j \leq N$ with $N = 2^n$ codeword length. Usually this information is not used (and not computed), because it is exploited the polar codes property that the trellis graph give already the uncoded result. So it would be interesting to compare their proposed G-matrix-based solution with the previous one in terms of hardware complexity.

ASIC realization Best presented BP architectures could be further studied to achieve an effective ASIC chip that targets a gate level technology. Constraining the architecture to the available hardware allows to evaluate real performance and compare them with post synthesis results.

Property	Unit	[36]	[43] P2	[44]	[46]	[50]	FP	RC	BI-W
Decoding Scheduling		BP G	BP A	BP A	BP C	SCAN	BP J	BP G	BP I
Block length		1024	1024	1024	1024	1024	1024	1024	1024
Rate		0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Technology		CMOS	CMOS	CMOS	CMOS	CMOS	CMOS	CMOS	CMOS
Process	nm	65	45	45	45	90	45	45	45
Core area	mm^2	1.48	4.80†	4.80†	0.75	0.97	10.89	1.48	1.93
Supply	V	1					1	1	1
Frequency	MHz	300	500	500	197	571	625	588	571
Power	mW	477.5					1846.7	331.0	638.14
Iterations		15	60	40	15	1	65	9	10
Throughput*	Mb/s	1024	2000	2900	841	479	4923	1858	3215
Energy efficiency	pJ/bit	466.31	328	328			375.11	178.09	198.48
Energy eff. per iter.	pJ/b/iter	31.09	5.47	8.20			5.77	19.79	19.85
Area efficiency	$Mb/s/mm^2$	693.77	619.17	897.80	1126.50	493.81	452.14	1253.96	1665.84
Normalized to 45 nm according to ITRS roadmap									
Throughput*	Mb/s	1263	2000	2900	841	652.42	4923	1858	3215
Energy efficiency	pJ/b	683.09	328	328			375.11	178.09	198.48
Area efficiency	$Mb/s/mm^2$	1250.21	619.17	897.80	1126.50	1195.74	452.14	1253.96	1665.84

∗ Throughput obtained by disabling the BP early-stopping rules for fair comparison.
† Estimation from gate count.

Table 7.1: *Comparison of implemented architectures with literature.*

Bibliography

[1] Matthieu Bloch. A (very) brief hystory of coding theory. http://users.ece.gatech.edu/mbloch/sp10_ece6606/coding_history.pdf.

[2] Masoud Salehi John G. Proakis. *Digital communications*. McGraw-Hill Higher Education, Boston, 5 edition, 2008.

[3] Bo Yuan and K.K. Parhi. Architecture optimizations for BP polar decoders. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2654–2658, May 2013.

[4] A. Eslami and H. Pishro-nik. On Finite-Length Performance of Polar Codes: Stopping Sets, Error Floor, and Concatenated Design. *IEEE Transactions on Communications*, 61(3):919–929, March 2013.

[5] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948.

[6] D. J. Costello and G. D. Forney. Channel coding: The road to channel capacity. *Proceedings of the IEEE*, 95(6):1150–1177, June 2007.

[7] R. W. Hamming. Berror detecting and error correcting codes. *Bell Syst. Tech. J.*, 29:147–160, 1950.

[8] M. J. E. Golay. Notes on digital coding. *Proc. IRE*, 37:657, June 1949.

[9] D. E. Muller. Application of boolean algebra to switching circuit design and to error detection. *IRE Trans. Electron. Comput.*, EC-3:6–12, September 1954.

[10] I. S. Reed. A class of multiple-errorcorrecting codes and the decoding scheme. *IRE Trans. Inform. Theory*, IT-4:38–49, September 1954.

[11] D. Slepian. A class of binary signal alphabets. *Bell Syst. Tech. J.*, 35:203–234, 1956.

[12] A. Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2:147–156, 1959.

[13] R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Inform. Contr.*, 3:68–79, 1960.

[14] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *J. SIAM*, 8:300–304, June 1960.

[15] R. A. Silverman and M. Balser. Coding for constant-data-rate systems. *IRE Trans. Inform. Theory*, PGIT-4:50–63, September 1954.

[16] P. Elias. Coding for noisy channels. *IRE Conv. Rec.*, 4:37–46, March 1955.

[17] P. Elias. Error-free coding. *IRE Trans. Inform. Theory*, IT-4:29–37, September 1954.

[18] R. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, January 1962.

[19] Jr. G. D. Forney. *Concatenated Codes*. MA: MIT Press, Cambridge, 1966.

[20] A. Glavieux C. Berrou and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo codes. In *Proc. 1993 Int. Conf. Commun.*, page 1064–1070, Geneva, Switzerland, May 1993.

[21] D. J. C. MacKay and R. M. Neal. Good codes on very sparse matrices. In C. Boyd, editor, *Proc. Cryptography Coding. 5th IMA Conf.*, page 100–111, Berlin, Germany, 1995. Springer.

[22] D. J. C. MacKay and R. M. Neal. Near shannon limit performance of low-density parity-check codes. *Elect. Lett.*, 32:1645–1646, August 1996.

[23] E. Arikan. Channel polarization: A method for constructing capacity-achieving codes. In *IEEE International Symposium on Information Theory, 2008. ISIT 2008*, pages 1173–1177, July 2008.

[24] C. Cahn. Combined Digital Phase and Amplitude Modulation Communication Systems. *IRE Transactions on Communications Systems*, 8(3):150–155, September 1960.

[25] Fuqin Xiong. *Digital Modulation Techniques*. Boston : Artech House, 2 edition, 2006.

[26] J. Hancock and R. Lucky. Performance of Combined Amplitude and Phase-Modulated Communication Systems. *IRE Transactions on Communications Systems*, 8(4):232–237, December 1960.

[27] C. Campopiano and B. Glazer. A Coherent Digital Amplitude and Phase Modulation Scheme. *IRE Transactions on Communications Systems*, 10(1):90–95, March 1962.

[28] E. Arıkan. On the Origin of Polar Coding. *IEEE Journal on Selected Areas in Communications*, 34(2):209–223, February 2016.

[29] Robert G. Gallager. *Information Theory and Reliable Communication*. Wiley, 1968.

[30] Peter IRockett Frank J. Aherne, Neil A.Thacker. The bhattacharyya metric as an absolute similarity measure for frequency coded data. *Kybernetika*, 34(4):[363]–368, 1998.

[31] E. Arikan. Channel combining and splitting for cutoff rate improvement. *IEEE Transactions on Information Theory*, 52(2):628–639, February 2006.

[32] E. Arıkan. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on Information Theory*, 55(7):3051–3073, July 2009.

[33] G.D. Forney Jr. Codes on graphs: normal realizations. *IEEE Transactions on Information Theory*, 47(2):520–548, February 2001.

[34] E. Arıkan. Polar codes: A pipelined implementations. In *Proc. 4th Int. Symp. Broadband Communication*, Melaka, Malaysia, 11-14 July 2010.

[35] N. Hussami, R. Urbanke, and S.B. Korada. Performance of polar codes for channel and source coding. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, pages 1488 –1492, 28 2009-july 3 2009.

[36] Youn Sung Park, Yaoyu Tao, Shuanghong Sun, and Zhengya Zhang. A 4.68gb/s belief propagation polar decoder with bit-splitting register file. In *2014 Symposium on VLSI Circuits Digest of Technical Papers*, pages 1–2, June 2014.

[37] Youn Sung Park. Energy-Efficient Decoders of Near-Capacity Channel Codes. 2014. PhD.

[38] E. Boutillon F. Guilloud and J.L. Danger. λ-min decoding algorithm of regular and irregular ldpc codes. In *International Symposium on Turbo Codes and Related Topics (ISTC)*, pages 451–454. IEEE, September 2003.

[39] A. Eslami and H. Pishro-Nik. On bit error rate performance of polar codes in finite regime. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 188–194, September 2010.

[40] A Pamuk. An FPGA implementation architecture for decoding of polar codes. In *2011 8th International Symposium on Wireless Communication Systems (ISWCS)*, pages 437–441, November 2011.

[41] R.L. Bharath Kumar and N. Chandrachoodan. A GPU implementation of belief propagation decoder for polar codes. In *2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pages 1272–1276, November 2012.

[42] Y. Zhang, Q. Zhang, X. Pan, Z. Ye, and C. Gong. A simplified belief propagation decoder for polar codes. In *Wireless Symposium (IWS), 2014 IEEE International*, pages 1–4, March 2014.

[43] B. Yuan and K. K. Parhi. Architectures for polar BP decoders using folding. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 205–208, June 2014.

[44] Bo Yuan and K.K. Parhi. Early Stopping Criteria for Energy-Efficient Low-Latency Belief-Propagation Polar Code Decoders. *IEEE Transactions on Signal Processing*, 62(24):6496–6506, December 2014.

[45] Bo Yuan and K.K. Parhi. Algorithm and architecture for hybrid decoding of polar codes. In *2014 48th Asilomar Conference on Signals, Systems and Computers*, pages 2050–2053, November 2014.

[46] J. Sha, X. Liu, Z. Wang, and X. Zeng. A memory efficient belief propagation decoder for polar codes. *China Communications*, 12(5):34–41, May 2015.

[47] G. Berhault, C. Leroux, C. Jego, and D. Dallet. Hardware implementation of a soft cancellation decoder for polar codes. In *2015 Conference on Design and Architectures for Signal and Image Processing (DASIP)*, pages 1–8, September 2015.

[48] C. Leroux, A. J. Raymond, G. Sarkis, and W. J. Gross. A Semi-Parallel Successive-Cancellation Decoder for Polar Codes. *IEEE Transactions on Signal Processing*, 61(2):289–299, January 2013.

[49] S. M. Abbas, Y. Fan, J. Chen, and C. Y. Tsui. Low complexity belief propagation polar code decoder. In *2015 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 1–6, October 2015.

[50] J. Lin, C. Xiong, and Z. Yan. Reduced complexity belief propagation decoders for polar codes. In *2015 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 1–6, October 2015.

[51] J. Xu, T. Che, and G. Choi. XJ-BP: Express Journey Belief Propagation Decoding for Polar Codes. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, December 2015.

[52] Jiming Chen Yan Zhang, Laurence T. Yang. *RFID and sensor networks: architectures, protocols, security and integrations*. CRC Press, 2010.

[53] A. Viterbi. Information theory in the sixties. *IEEE Transactions on Information Theory*, 19(3):257–262, May 1973.

[54] P Vontobel R. Koetter. Graph-covers and iterative decoding of finite length codes. In *3rd International Symposium on Turbo Codes and related topics*, 1-5 Sept. 2003.

[55] Michael S. Postol David J.C. MacKay. Weaknesses of margulis and ramanujan-margulis low-density parity-check codes. *Electronic Notes in Theoretical Computer Science*, 74, 2003.

[56] K.K. Parhi D. Oh. Min-sum decoder architectures with reduced word length for ldpc codes. *IEEE Transactions on circuits and systems*, 57(1):105–115, January 2010.