Polar Codes and Their Quantum-Domain Counterparts

Zunaira Babar, Zeynep B. Kaykac Egilmez, Luping Xiang, Daryus Chandra, Robert G. Maunder, Soon Xin Ng and Lajos Hanzo

Abstract—Arikan's polar codes are capable of approaching Shannon's capacity at a low encoding and decoding complexity, while conveniently supporting rate adaptation. By virtue of these attractive features, polar codes have found their way into the 5G New Radio (NR). Hence, in this paper we provide a comprehensive survey of polar codes, highlighting the major milestones achieved in the last decade. Furthermore, we also provide tutorial insights into the operation of the polar encoder, decoders as well as their code construction methods. We also extend our discussions to quantum polar codes with an emphasis on the underlying quantum-to-classical isomorphism and the syndrome-based quantum polar decoders.

Keywords—*channel coding, channel polarization, capacity, polar codes, quantum error correction.*

ACRONYMS

ASIC	Application Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BCH	Bose-Chaudhuri-Hocquenghem
B-DMC	Binary-Input Discrete Memoryless Channel
BEC	Binary Erasure Channel
BER	Bit Error Rate
BICM	Bit-Interleaved Coded Modulation
BICM-ID	BICM with Iterative Decoding
BLER	BLock Error Ratio
B-MC	Binary-input Memoryless Channel
BP	Blief Propagation
BSC	Binary Symmetric Channel
BP	Belief Propagation
CA-SCL	Cyclic Redundancy Check-Aided SCL
CA-SCS	Cyclic Redundancy Check-Aided SCS
CSS	Calderbank-Shor-Steane
CNOT	Controlled NOT
Cz	Controlled-Z
CRC	Cyclic Redundancy Check
DE	Density Evolution
DMC	Discrete Memoryless Channel
eMBB	enhanced Mobile BroadBand
FEC	Forward Error Correction

FPGA	Field Programmable Gate Array
GA	Gaussian Approximation
HARQ	Hybrid Automatic Repeat Request
IRCC	IRregular Convolutional Coded
LDPC	Low Density Parity Check
LLR	Log-Likelihood Ratio
LR	Likelihood Ratio
MERA	Multi-scale Entanglement Renormalization Ansatz
ML	Maximum Likelihood
ML-SSC	Maximum Likelihood Simplified Successive Cancellation
mMTC	massive Machine Type Communication
NR	New Radio
PBCH	Physical Broadcast CHannel
PC	Parity Check
QECC	Quantum Error Correction Code
QPSK	Quadrature Phase Shift Keying
QSCD	Quantum Successive Cancellation Decoder
RM	Reed-Muller
RRNS	Redundant Residue Number System
RS	Reed-Solomon
SC	Successive Cancellation
SCAN	Soft CANcellation
SCH	Successive Cancellation Hybrid
SCL	Successive Cancellation List
SCS	Successive Cancellation Stack
SNR	Signal-to-Noise Ratio
SSC	Simplified Successive Cancellation
SSCL	Simplified Successive Cancellation List
TCM	Trellis Coded Modulation
TTCM	Turbo Trellis Coded Modulation
URC	Unity Rate Code
URLLC	Ultra-Reliable Low-Latency Communication
XOR	eXclusive-OR
3GPP	Third Generation Partnership Project

I. INTRODUCTION

Information is the resolution of uncertainty,
Claude Shannon.

The inception of classical coding theory dates back to 1948 [1], when Claude Shannon introduced the notion of 'channel capacity'. Explicitly, Shannon predicted in his seminal paper [1] that virtually error-free transmission over noisy

channels can be achieved by invoking error correction codes having a coding rate R lower than the channel capacity C and having an infinitely long codeword length. The capacity of an Additive White Gaussian Noise (AWGN) channel having the bandwidth B (Hz) and the noise power spectral density of $N_0/2$ (Watts/Hz) per dimension is quantified by the Shannon-Hartley theorem as follows:

$$C = B \log_2 \left(1 + \frac{S}{N_0 B} \right), \qquad (1)$$

when the average transmitted power is S Watts. Hence, the maximum permissible coding rate of an error correction code operating close to capacity is limited by the Signal-to-Noise Ratio (SNR) ($\frac{S}{N_0 B}$) and the bandwidth (B) under the assumption of tolerating infinite implementation complexity and transmission delay. Similarly, the capacities of a Binary Symmetric Channel (BSC) or a Binary Erasure Channel (BEC) are specified by their respective channel characteristics, i.e. by the state cross-over probability of the BSC and the erasure probability of the BEC, again assuming infinite processing and time resources. However, practical systems can neither afford an infinite implementation complexity nor can they tolerate an infinite transmission delay. So, we need optimized codes, which perform close to Shannon's capacity limit, while guaranteeing the desired target performance metrics, as illustrated in Fig. 1.

Shannon quantified the capacity limit and proved the existence of 'capacity-achieving' codes based on the 'random-coding' argument. However, he did not give any recipes for constructing such capacity-achieving codes. Over the last seven decades, researchers have endeavored to design optimum codes, which are capable of operating close to the capacity limit, while also striking a compelling trade-off amongst the desired performance metrics of Fig. 1. Broadly speaking, this
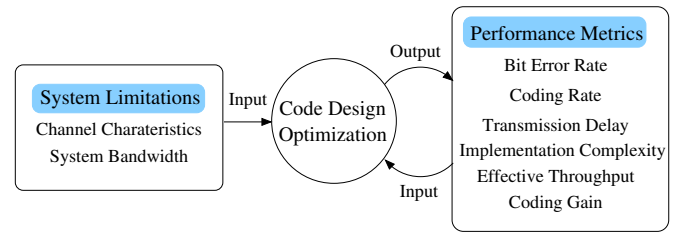

Fig. 1: Factors driving the code design optimization.

quest has pursued two avenues: the algebraic coding avenue and the probabilistic coding avenue, as portrayed in the stylized road map of Fig. 2. Algebraic coding was the main research avenue for the first few decades. The aim of this coding paradigm is to design powerful codes by exploiting finite-field arithmetic to maximize the minimum Hamming distance[1] between the codewords for a given coding rate, or more specifically for the given information word length k and codeword length n. This has given rise to a range of popular coding families, which includes for example Hamming codes [2], Reed-Muller (RM) codes [3], [4], Bose-Chaudhuri-Hocquenghem (BCH) codes [5], [6], Reed-Solomon (RS) codes [7] and Redundant Residue Number System (RRNS) codes [8], [9]. Unfortunately, maximizing the minimum distance of codewords along the algebraic avenue of Fig. 2 does not promise a 'capacity-achieving' design. Nonetheless, algebraic codes have found their way into practical applications by virtue of their strong error correction capabilities (or equivalently low Bit Error Rate (BER) floors). Explicitly, these codes are useful, when the

---

[1]The Hamming distance between two vectors is equal to the number of positions at which the corresponding elements (bits or symbols) differ.
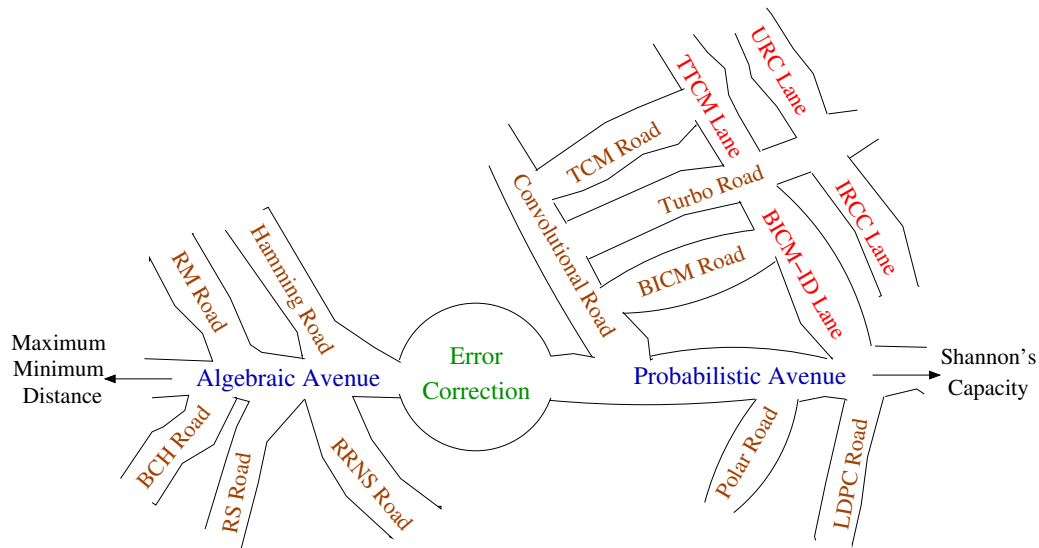

Fig. 2: Road map portraying the evolution of classical channel coding theory. The algebraic avenue aims at maximizing the minimum distance, while the probabilistic avenue leads to capacity approaching designs.

received information is in the form of hard decisions. For example, RS codes are used in magnetic tape and disk storage as well as in several standardized systems, such as the deep-space coding standard [10], where they typically constitute an outer layer[2] of error correction (known as the outer code) to reduce the BER floor resulting from the failure of the inner layer of error correction (called the inner code).

In contrast to the algebraic coding avenue, the probabilistic coding avenue has succeeded in paving the way to capacity. Explicitly, the probabilistic coding avenue of Fig. 2 is inspired by Shannon's random coding philosophy and strives for achieving a reasonable trade-off between the performance and the complexity. This design avenue has led to the construction of convolutional codes [11], Low Density Parity Check (LDPC) codes [12]–[14], turbo codes [15], [16] as well as polar codes [17]. The probabilistic coding paradigm also includes various 'turbo-like' iterative coding schemes, for example turbo BCH codes [18], turbo Hamming codes [19], and the Unity Rate Code (URC)-assisted and IRregular Convolutional Coded (IRCC) concatenated schemes of [20], [21], as well as the coded modulation schemes, including Trellis Coded Modulation (TCM) [22]–[24], Bit-Interleaved Coded Modulation (BICM) [25], [26], BICM with Iterative Decoding (BICM-ID) [27], and Turbo Trellis Coded Modulation (TTCM) [28]. In particular, the turbo and LDPC codes made it possible to operate arbitrarily close to the Shannon limit, while the polar codes finally managed to provably approach the capacity, albeit at infinitely long codeword lengths. Despite being a relatively immature coding scheme, polar coding has proved to be a fierce competitor of turbo and LDPC codes, both of which have been ruling for over two decades. Polar codes have already found their way into the 5G New Radio (NR) for the control channels of the enhanced Mobile BroadBand (eMBB) and the Ultra-Reliable Low-Latency Communication (URLLC) use-cases as well as for the Physical Broadcast CHannel (PBCH). Polar codes have also been identified as potential candidates for the data and control channels of the massive Machine Type Communication (mMTC) use-cases.

Polar codes emerged in at once, when the apprehension that 'coding is dead' started looming again[3]. Hence, the discovery of polar codes re-energized the coding community and equipped them with a radically different approach for approaching Shannon's capacity. In addition to its influence on the classical coding theory, polar codes have also attracted considerable attention within the quantum research community. *Motivated by the growing interest in polar codes, in this paper we provide a comprehensive survey of both the classical as well as quantum polar codes, taking the readers through the major milestones achieved and providing a slow-paced tutorial on the related encoding and decoding algorithms.* This tutorial

paper sets the necessary background for understanding the operation of the Third Generation Partnership Project (3GPP) 5G NR polar codes, which have been surveyed in [29], [30]. It is pertinent to mention here that, to the best of authors' knowledge, only two survey papers [31], [32] exist on polar codes at the time of writing. In [31], the author's have provided a succinct overview of the fundamental concepts pertaining to polar codes, including the encoding, decoding and construction methods, while [32] focuses on the Application Specific Integrated Circuit (ASIC) implementation of polar decoders. By contrast, this paper has a broader scope, since we provide in-depth tutorial insights (with explicit examples) as well as a comprehensive survey on polar encoders, decoders as well as polar code construction methods.

Fig. 3 provides the overview of the paper. We commence our discourse in Section II, where we discuss Arikan's channel polarization philosophy, which is the key to approaching Shannon's capacity. We then survey the polar encoding and decoding algorithms in Section III and Section IV, respectively, with detailed tutorial insights. Continuing further our discussions, we present the polar code design principles and guidelines in Section V. In Section VI, we detail the transition from the classical to the quantum channel polarization by identifying the underlying isomorphism. Based on the isomorphism of Section VI, we proceed to quantum polar codes in Section VII. Finally, we conclude our discussions in Section VIII.

II. THE PHILOSOPHY OF CHANNEL POLARIZATION

Polar codes rely on the phenomenon of channel polarization, which is the key contributor towards approaching Shannon's capacity. Channel polarization is basically the process of redistributing channel capacities among the various instances, or more precisely uses, of a transmission channel without affecting, while conserving the total capacity, as encapsulated in Fig. 4. Explicitly, channel polarization implies that a set of given channels is polarized into perfect and useless (or completely random) channels, having capacities of 1 and 0, respectively. This in turn makes the channel coding problem trivial, since the perfect channels may be used for transmitting uncoded information without any errors, while the useless channels can be discarded.

Let us consider N uses of a B-DMC W, each having a capacity of $I(W)$, as exemplified in Fig. 4. In the asymptotic region, i.e. when N is infinitely large, channel polarization results in $k = N \times I(W)$ perfect channels having near-1 capacity and $(N-k)$ useless channels having near-0 capacity. Thereafter, k information bits are sent uncoded (rate-1) through the induced perfect channels, while the $(N-k)$ inputs to the induced useless channels are frozen, implying that known redundant bits are sent across these channels (rate-0)[4]. Hence, the resultant coding rate is equivalent to the channel capacity,

---

[2]Outer layer is with respect to the channel. The inner layer is closer to the channel.

[3]The notion that 'coding is dead' first officially surfaced in the IEEE Communication Theory Workshop held in St. Petersburg, Florida, in April 1971, where a group of coding theorists concluded that there was nothing more to do in coding theory. This workshop became famous as the 'coding is dead' workshop.

[4]An induced bit-channel is a hypothetical end-to-end channel, which consists of the encoder, the real channel and the decoder, as discussed further in Section II-A. Hence, the information sent across the real channels is encoded. However, the resulting induced bit-channels may be viewed as rate-1 and rate-0 channels, since they exhibit a capacity of 0 and 1, respectively.
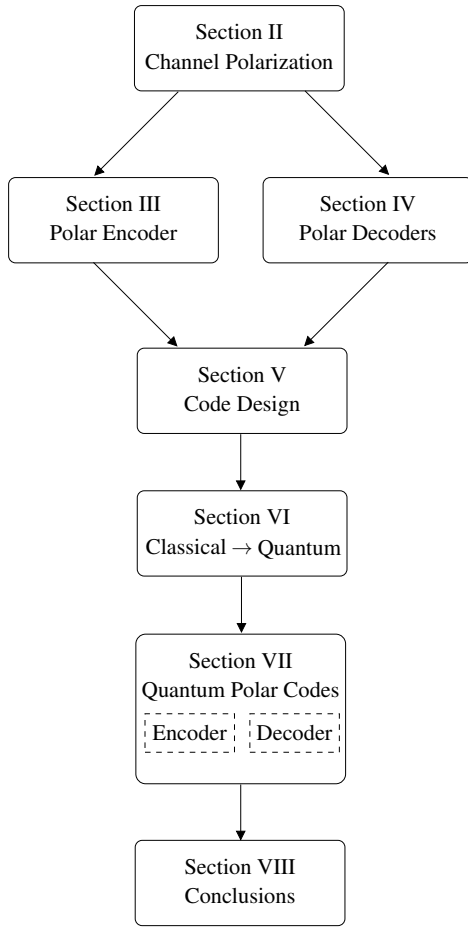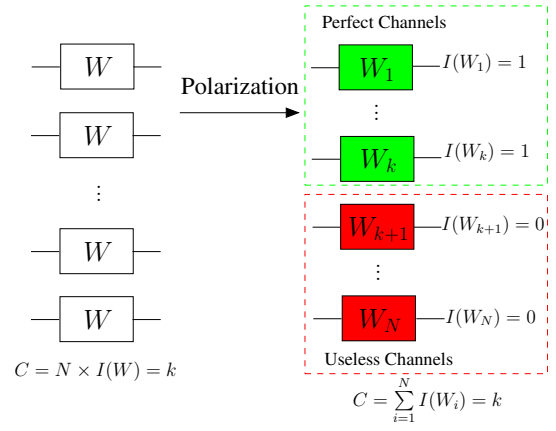
Fig. 3: Paper structure.

Fig. 4: The philosophy of channel polarization: the total capacity of N B-DMCs is redistributed, resulting in $k = N \times I(W)$ perfect channels having unit capacity each and $(N - k)$ useless channels having zero capacity, without affecting the total capacity. Perfect channels are used for transmitting k information bits, while the useless channels are frozen.

For simplicity, let W be a BEC having an erasure probability of ϵ. Hence, the total capacity of the compound channel W^2 is:

$$C(W^2) = 2 \times I(W) = 2 \times (1 - \epsilon). \quad (4)$$

According to the chain rule [33], Eq. (3) may also be expressed as follows:

$$\begin{aligned} C(W^2) &= I(u_1, u_2; y_1, y_2) \\ &= I(u_1; y_1, y_2) + I(u_2; y_1, y_2 | u_1) \\ &= I(u_1; y_1, y_2) + I(u_2; y_1, y_2, u_1), \quad (5) \end{aligned}$$

since u is an independently and identically distributed random variable, u_1 and u_2 are independent. Eq. (5) implies that the compound channel W^2 may be split into the two single-bit channels W_1 and W_2 of Fig. 5b and Fig. 5c, respectively, which are defined as follows:

$$W_1 \triangleq u_1 \rightarrow (y_1, y_2) \quad (6)$$
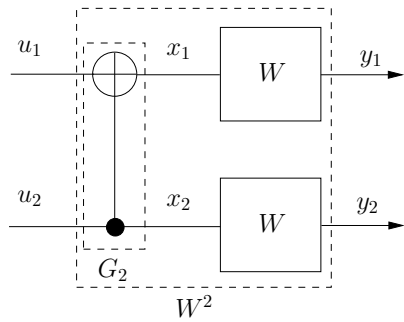$$W_2 \triangleq u_2 \rightarrow (y_1, y_2, u_1). \quad (7)$$

This splitting process, which occurs at the decoder, redistributes the total capacity between the two induced bit channels W_1 and W_2, so that one of the channels gets better, while the other gets worse.

Resuming our example of a BEC, let us calculate the capacities of the two induced channels. The first bit-channel W_1 of Fig. 5b has no information about u_2. So, the receiver estimates u_1 based on the received bits y_1 and y_2 as follows:

$$\hat{u}_1 = y_1 \oplus y_2, \quad (8)$$

where $\oplus$ denotes modulo-2 addition. Hence, u_1 can be decoded only when neither y_1 nor y_2 is erased. Hence, the erasure probability of the induced bit-channel W_1 is:

$$\epsilon_1 = 1 - (1 - \epsilon)^2 = 2\epsilon - \epsilon^2, \quad (9)$$

i.e. we have:

$$R = \frac{k}{N} = \frac{NI(W)}{N} = I(W). \quad (2)$$

But how do we achieve channel polarization? This is where Arikan's polar codes come in [17], which inspired researchers to develop more sophisticated coding schemes for achieving channel polarization. More explicitly, polar encoders convert the N inputs into bits that can be sent across the real channels at Shannon's capacity.

A. Arikan's Polar Codes

Arikan's polar codes [17] achieve channel polarization by recursively invoking the simple 2-bit encoding kernel of Fig. 5a. More specifically, two uses of the B-DMC are combined using a single eXclusive-OR (XOR) gate at the encoder. This step is termed as 'channel combining' and the resultant compound channel is denoted by W^2, which has a capacity of:

$$C(W^2) = I(u_1, u_2; y_1, y_2) = I(x_1, x_2; y_1, y_2) = 2 \times I(W). \quad (3)$$

(a) Channel combining.

(b) Channel splitting: first bit channel.

$W_1 : u_1 \rightarrow (y_1, y_2)$

(c) Channel splitting: second bit channel.

$W_2 : u_2 \rightarrow (y_1, y_2, u_1)$

Fig. 5: Arikan's 2-bit polar code, relying on channel combining and channel splitting for channel polarization.
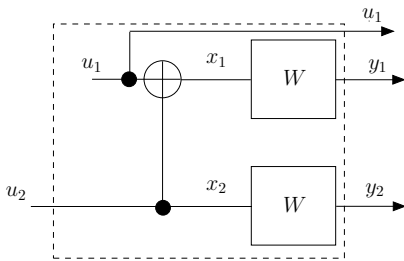
which is worse than that of the original channel W. Consequently, W_1 is the worse channel, also denoted as W^-, having a reduced capacity of:

$$I(W^-) = 1 - \epsilon_1 = 1 - 2\epsilon + \epsilon^2. \qquad (10)$$

The second bit channel of Fig. 5c outputs u_1 as well as y_1 and y_2. Explicitly, the availability of u_1 implies that we have a 'genie' decoder or a side-information channel, which reveals the value of u_1. Consequently, u_2 can be decoded as long as neither y_1 nor y_2 is erased. Hence, the erasure probability of the induced channel W_2 is:

$$\epsilon_2 = \epsilon^2, \qquad (11)$$

which is lower than that of the original channel W. This implies that W_2 is the better channel, also denoted as W^+,

which has a capacity of:

$$I(W^+) = 1 - \epsilon_2 = 1 - \epsilon^2. \qquad (12)$$

Based on Eq. (10) and Eq. (12), we may conclude that:

$$I(W^-) \leq I(W) \leq I(W^+), \qquad (13)$$

which implies that W^- (or equivalently W_1) tends to polarize towards zero-capacity, while W^+ (or equivalently W_2) tends to polarize towards unit capacity. It is pertinent to mention here that the equality in Eq. (13) holds only when W is an extreme channel having a capacity of either 0 or 1. Furthermore, the total capacity remains unaffected, since we have:

$$I(W^-) + I(W^-) = 2 \times I(W). \qquad (14)$$

To elaborate further, let us assume that $\epsilon = 0.5$, hence we have:

$$I(W) = 0.5$$
$$I(W_1) = I(W^-) = 0.25$$
$$I(W_2) = I(W^+) = 0.75, \qquad (15)$$

where we may observe that the value of $I(W^-)$ gets closer to zero, while that of $I(W^+)$ gets closer to one. The impact of channel polarization may be enhanced by recursively invoking the basic encoding kernel of Fig. 5a, which is labeled 'G_2'. More specifically, as exemplified in Fig. 6, a compound channel W^4 can be constructed by using two copies of the compound channel W^2. In the first layer of polarization, two independent copies of W^2 are invoked, inducing two good channels W^+ and two bad channels W^-, which have the capacities of 0.25 and 0.75, respectively, for $\epsilon = 0.5$. In the second layer of polarization, the bad channels are coupled using the encoding kernel G_2 of Fig. 5a, which is simply a XOR gate marked in red in Fig. 6. Recall that the encoding transformation of Fig. 5a yields a bad and a good channel, whose capacities may be calculated using Eq. (10) and Eq. (12), respectively. Consequently, the red XOR gate of Fig. 6 polarizes the two W^- channels into a worse channel W^{--} and a better channel W^{-+}, having capacities of 0.0625 and 0.4375, respectively. We may observe here that the second layer of polarization polarizes the first channel more towards the zero capacity, while the capacity of the other channel tends to increase towards one.

Similarly, the second layer of polarization couples the two good channels W^+ using the XOR gate marked in green in Fig. 6. This second XOR gate induces the channels W^{+-} and W^{++} having capacities of 0.5625 and 0.9375, respectively. Hence, a two-layered polarization yields two strongly polarized channels W^{--} and W^{++}, which exhibit a higher degree of polarization than the channels W^- and W^+. It is important to point out that the bits in Fig. 6 are indexed according to their decoding order, which will be discussed further in Section IV. Furthermore, it may also be observed that the bits in Fig. 6 follow a bit-reversed indexing. Explicitly, bit reversing implies that a number $i \in \{1, N\}$ having the n-bit binary representation $i_1 i_2 \ldots i_n$, for $n = \log_2 N$, is mapped onto its bit-reversed counterpart having
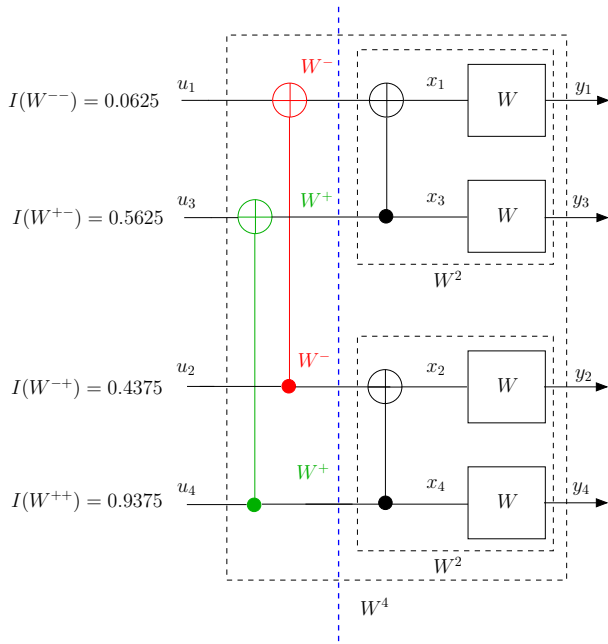
Fig. 6: 4-bit compound channel W^4 (bit-reversed indexing): it is constructed from two 2-bit channels W^2 by coupling the bad channels W^- using the red XOR gate and the good channels W^+ using the green XOR gate (viewing the schematic from right to left). This enhances the strength of polarization yielding two strongly polarized channels W^{--} and W^{++}.

Fig. 7: 4-bit compound channel W^4 (natural indexing), which is obtained by re-wiring the inputs of Fig. 6.

the binary representation $i_n i_{n-1} \ldots i_1$. Consequently, the four inputs of Fig. 6 $\{1, 2, 3, 4\}$ having the binary representations $\{00, 01, 10, 11\}$ are indexed as $\{1, 3, 2, 4\}$ corresponding to their bit-reversed binary representations $\{00, 10, 01, 11\}$. This bit-reversed order of the bits may also be obtained from the labels $\{W^{--}, W^{+-}, W^{-+}, W^{++}\}$ of the induced channels by mapping $+$ and $-$ onto 1 and 0, respectively. Hence, the sequence $\{W^{--}, W^{+-}, W^{-+}, W^{++}\}$ yields the bit-reversed order $\{00, 10, 01, 11\}$. The inputs of Fig. 6 may be re-wired to follow the natural indexing order, as shown in Fig. 7. Bit-reversed indexing of Fig. 6 facilitates hardware implementations, while the natural indexing of Fig. 7 is more apt for software implementations. In this paper, we will follow the the natural indexing of Fig. 7.

The polarization of the induced bit-channels of the compound channel W^4 may be enhanced by invoking a third layer of polarization using two independent copies of W^4, resulting in the compound channel W^8. The process may be repeated recursively using the generalized channel combining transformation of Fig. 8 until either the desired length N or the desired polarization strength is achieved. Fig. 8 shows the schematic of G_N for constructing an N-bit compound channel W^N. The associated capacities of the induced bit-channels can
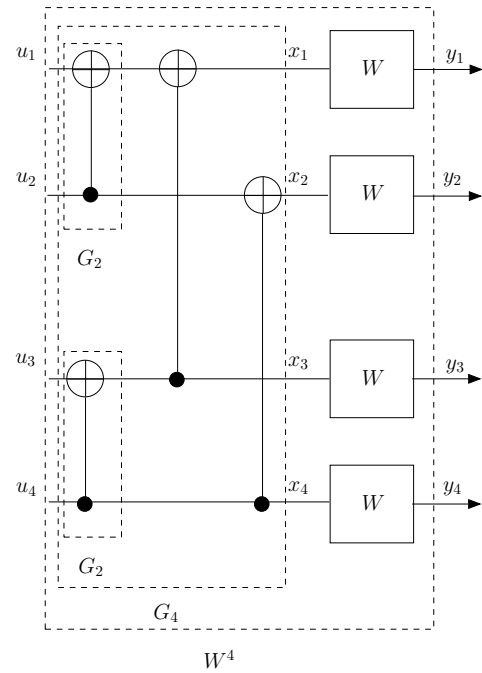
be recursively calculated as follows:

$$I\left(W_{2i-1}^N\right) = I\left(W_i^{N/2}\right)^2$$
$$I\left(W_{2i}^N\right) = 2I\left(W_i^{N/2}\right) - I\left(W_i^{N/2}\right)^2, \qquad (16)$$

for $1 \leq i \leq N/2$, where W_i^N denotes the ith induced bit-channel of an N-bit compound channel W^N. Please note that the recursive calculation of capacities given in Eq. (16) is only valid for BECs.

The encoding transformation of Fig. 8 embeds n layers of polarization, when the codeword length is $N = 2^n$. Each layer of polarization makes the bad channels worse and the good channels better, which is typically referred to as the 'Matthew effect', and demonstrated using the polarization martingale of Fig. 9, which converges to 0 and 1, when N is infinitely long. To elaborate further, Fig. 10 shows the bit-wise mutual information intensity map for increasing codeword length. We may observe in Fig. 10 that the proportion of near-0 and near-1 bit-channels increases as the codeword length is increased from 32 to 4096. Please note that the bit-channel indices in Fig. 10 are sorted based on their mutual information values. Hence, polar codes do not polarize the channels completely at finite codeword lengths. Explicitly, in contrast to the perfect and useless channels of Fig. 4 having capacities of 1 and 0, respectively, finite-length polar codes polarize the underlying channels into the set of good and bad channels, which tend to polarize towards the absolute 1 and 0 capacity, when the codeword length is infinitely long.
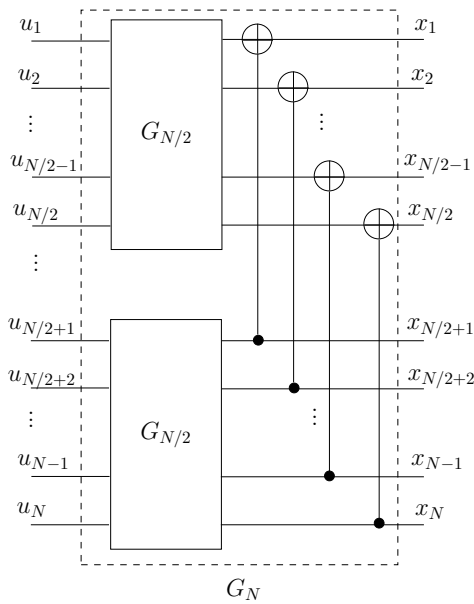
Fig. 8: Channel combining transformation G_N for constructing an N-bit compound channel W^N. It is constructed from two $(N/2)$-bit transformations by coupling their corresponding outputs using a XOR gate (viewing the schematic from left to right).

B. Non-Arikan Polar Codes

Channel polarization is a general technique and is hence not restricted to Arikan's polar code [34], [35]. The channel combining and channel splitting procedures of Fig. 5 may be generalized for an arbitrary encoder G_N, as shown in Fig. 11 and Fig. 12, respectively. More specifically, the encoder G_N of Fig. 11 is a one-to-one N-bit mapper, which constructs a compound channel W^N by coupling N uses of a B-DMC. The total capacity of the resultant channel W^N is:

$$C(W^N) = I(u_1^N; y_1^N) = I(x_1^N; y_1^N) = N \times I(W), \quad (17)$$

where a_i^j denotes the vector $(a_i, a_{i+1}, \ldots, a_j)$. Analogous to Eq. (5), the chain rule may be applied to Eq. (17), as follows:

$$
\begin{aligned}
C(W^N) &= I(u_1^N; y_1^N)) \\
&= I(u_1; y_1^N) + I(u_2; y_1^N|u_1) + \ldots I(u_N; y_1^N|u_1^{N-1}) \\
&= \sum_{i=1}^{N} I(u_i; y_1^N|u_1^{i-1}) \\
&= \sum_{i=1}^{N} I(u_i; y_1^N, u_1^{i-1}). \quad (18)
\end{aligned}
$$

Hence, the compound channel W^N may be split into N induced bit-channels, so that the ith bit-channel is defined as:

$$W_i \triangleq u_i \rightarrow \left(y_1^N, u_1^{i-1}\right), \quad (19)$$

which is also illustrated in Fig. 12. More specifically, the induced bit-channel W_i takes the input $u_i \in \mathcal{X}$, yielding the
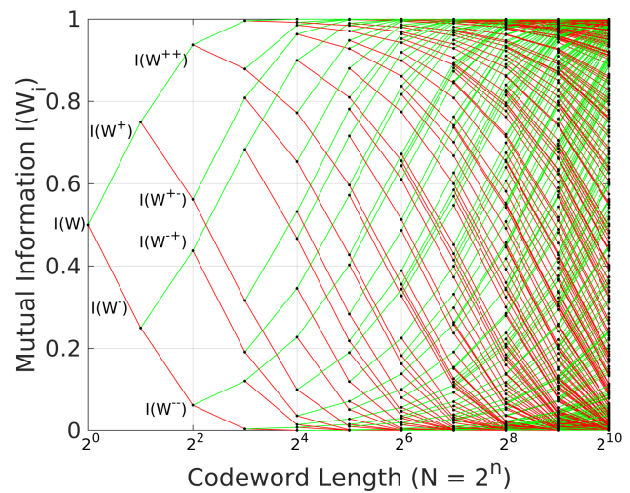
Fig. 9: $[0, 1]$-bounded polarization martingale for a BEC $I(W)$ having $\epsilon = 0.5$. The mutual information of the induced bit-channels W_i converges to 0 or 1 upon increasing the codeword length N.
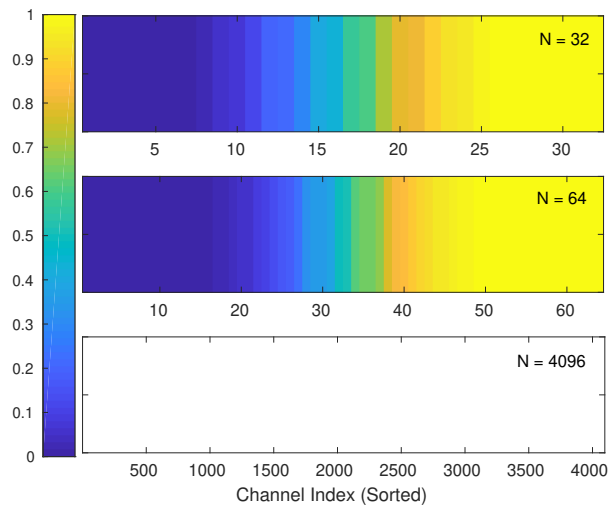
Fig. 10: Mutual information intensity map for a BEC at $\epsilon = 0.5$ and codeword lengths of 32, 64 and 4096. The strength of polarization enhances upon increasing the codeword length.

output $(y_1^N, u_1^{i-1}) \in \mathcal{Y}^N \times \mathcal{X}^{i-1}$, where $\mathcal{X}$ and $\mathcal{Y}$ denote the input and output alphabets of the channel W. Any random N-bit permutation G_N may exhibit good polarization characteristics, as identified by Shannon's random coding argument. However, Arikan's polar codes bring with them the benefit of a simple recursive structure, which is easy to implement.

Arikan's polar codes, relying on the 2-bit kernel G_2 of Fig. 5a, exhibit an error exponent of $1/2$. This implies that the net error probability of the resultant polar code decays

Fig. 11: Channel combining: the encoder G_N combines the N uses of a B-DMC to construct a compound channel W^N.

Fig. 12: Channel splitting: the ith induced bit-channel W_i.

exponentially with the square root of the codeword length N, when N is sufficiently large. For the sake of achieving a higher error exponent, higher-dimensional kernels, relying on more than 2 bits, were used by Korada *et 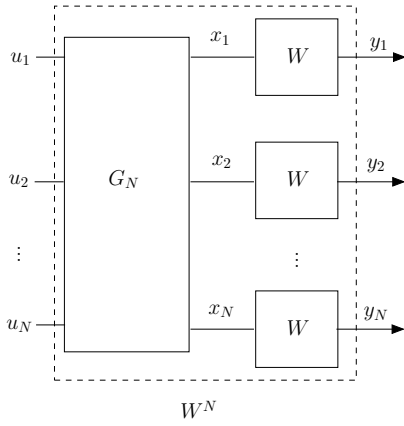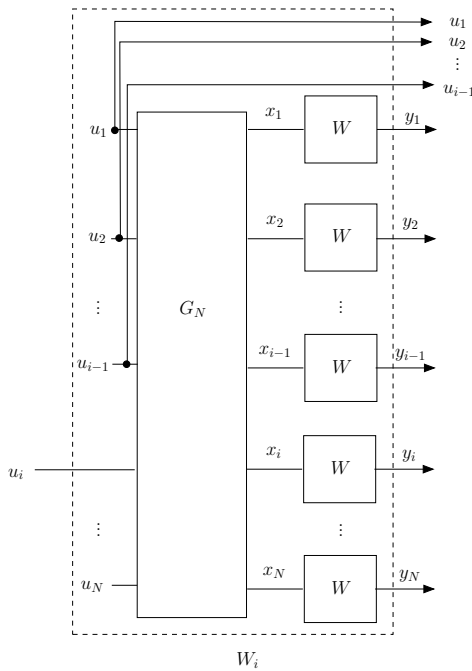al.* [36] for polarizing B-DMCs. Korada *et al.* [36] also conceived a general formalism for designing BCH-based l-bit kernels, which achieve error exponents close to 1 for large values of l. The notion of code decomposition (also called code nesting) was used in [37], [38] for increasing the error exponents of higher-dimensional kernels, while BCH, RS and Golay code based kernels were explored in [39]–[42] for the sake of increasing the minimum distance of polar codes. Furthermore,

the phenomenon of channel polarization was generalized to non-binary channels in [43], [44], while RS-based non-binary constructions were investigated in [45]–[47]. Finally, the idea of mixed kernels, consisting of sub-kernels of arbitrary non-binary alphabet sizes, was investigated in [48], [49]. The resultant polar codes provided attractive benefits both in terms of the error correction performance as well as the decoding complexity. However, the codeword length was limited to $N = l^n$. For the sake of obtaining arbitrary codeword lengths, puncturing and shorterning techniques were exploited in [50]–[52], while multi-kernel polar codes, invoking multiple l-bit kernels, were conceived in [53], [54]. Nonetheless, in this paper, we will focus on Arikan's polar codes.

III. POLAR ENCODERS

Recall from Section II-A that polar codes induce a set of good and bad channels, so that information bits can be transmitted through the induced good channels, while the input to the induced bad channels are frozen. When the codeword length N is infinitely long, the channels are polarized into k good channels (or the so-called perfect channels) and $(N-k)$ bad channels (or the so-called useless channels). Hence, a polar code encodes k information bits into N coded bits using $(N-k)$ redundant bits, which are called the 'frozen bits'. It is characterized by the parameters $(N, k, \mathcal{F}, u_{\mathcal{F}})$, where $\mathcal{F} \subset \{1, 2, \ldots, N\}$ specifies the location of frozen bits, while $u_{\mathcal{F}}$ is an $(N-k)$-bit vector of frozen bits, which are known to the decoder. The performance of polar codes rely on the parameters N, k as well as $\mathcal{F}$. In particular, $\mathcal{F}$ is channel specific and must be optimized for the channel under consideration, as discussed further in Section V. However, the performance of polar codes is unaffected by the value of frozen bits, more precisely the vector $u_{\mathcal{F}}$, if the channel is symmetric. Generally, $u_{\mathcal{F}}$ is assumed to be an all-zero vector. It is important to note here that polar codes are intrinsically rate compatible, since the coding rate can be varied by merely changing the number of frozen bits, while using the same encoder G_N. The frozen bits' locations $\mathcal{F}$ can be selected using a sequence that reads the locations. The best locations for one coding rate are typically a subset of the best locations for any lower coding rate.

The polar encoding process, which was generalized in Fig. 11 for an arbitrary encoder G_N, may be represented as:

$$x_1^N = u_1^N G_N. \tag{20}$$

For Arikan's polar code, G_N is the nth Kronecker product of the (2×2) kernel matrix, which characterizes the encoding transformation G_2 of Fig. 5a. More specifically, Arikan's kernel G_2 may be represented in matrix form as follows:

$$G_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \tag{21}$$

while the N-bit encoder G_N is defined recursively as:

$$G_N = G_2^{\otimes n} = \begin{pmatrix} G_{N/2} & 0 \\ G_{N/2} & G_{N/2} \end{pmatrix}. \tag{22}$$

Hence, Arikan's polar code has a recursive structure invoking $n = \log_2 N$ layers of polarization and each layer of polarization uses $N/2$ XOR gates, as previously illustrated in Fig. 8. Hence, the encoding operation of Eq. (22) imposes a complexity of $O(N \log_2 N)$. We may also notice from Eq. (22) that polar codes assume a non-systematic structure. Later, systematic polar codes were derived in [55]–[58], which outperformed the classic non-systematic polar codes in terms of the BER, while retaining the same BLock Error Ratio (BLER) and encoding, decoding complexity. We restrict our discussions to the classic non-systematic polar codes in this paper.

The 8-bit polar encoder may be formulated as follows:

$$G_8 = G_2^{\otimes 3} = \begin{pmatrix} G_2 & 0 & 0 & 0 \\ G_2 & G_2 & 0 & 0 \\ G_2 & 0 & G_2 & 0 \\ G_2 & G_2 & G_2 & G_2 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}. \quad (23)$$

Let us consider an 8-bit polar code having $k = 4$, $\mathcal{F} = \{1, 2, 3, 5\}$, $u_{\mathcal{F}} = (0\ 0\ 0\ 0)$ and an information bit sequence $u_{\mathcal{F}_c} = (1\ 0\ 0\ 1)$. Then the encoded output can be computed as follows:

$$\begin{aligned} x_1^8 &= (\ 0\quad 0\quad 0\quad u_4\quad 0\quad u_6\quad u_7\quad u_8\ ) \cdot G_8 \\ &= (\ 0\quad 0\quad 0\quad 1\quad 0\quad 0\quad 0\quad 1\ ) \cdot G_8 \\ &= (\ 0\quad 0\quad 0\quad 0\quad 1\quad 1\quad 1\quad 1\ ). \end{aligned} \quad (24)$$

Please note that $\mathcal{F}_c$ denotes the complementary set of $\mathcal{F}$, which specifies the location of information bits. The codeword of Eq. (24) may also be directly worked out from the encoding circuit, as exemplified in Fig. 13.

The encoding process of Eq. (20) may be reformulated as:

$$x_1^N = u_{\mathcal{F}_c} G_N(\mathcal{F}_c) + u_{\mathcal{F}} G_N(\mathcal{F}), \quad (25)$$

where $G_N(\mathcal{F})$ is a submatrix of G_N containing only the rows with indices in $\mathcal{F}$. When $u_{\mathcal{F}}$ is set to an all-zero bit sequence, Eq. (25) reduces to:

$$x_1^N = u_{\mathcal{F}_c} G_N(\mathcal{F}_c), \quad (26)$$

where $G_N(\mathcal{F}_c)$ is a $(k \times N)$ generator matrix. Hence, polar codes are equivalent to linear block codes having the generator matrix $G_N(\mathcal{F}_c)$. The encoder of polar codes brings with it the additional benefit of scalability, both in terms of the codeword length as well as the coding rate. More specifically, the feature of length scalability comes from the recursive nature of G_N, while the rate can be modified by only changing the number of frozen bits. It may also be noticed in Eq. (25) that when the frozen bits are not set to an all-zero sequence, then the resulting code is a coset of the linear block code having the
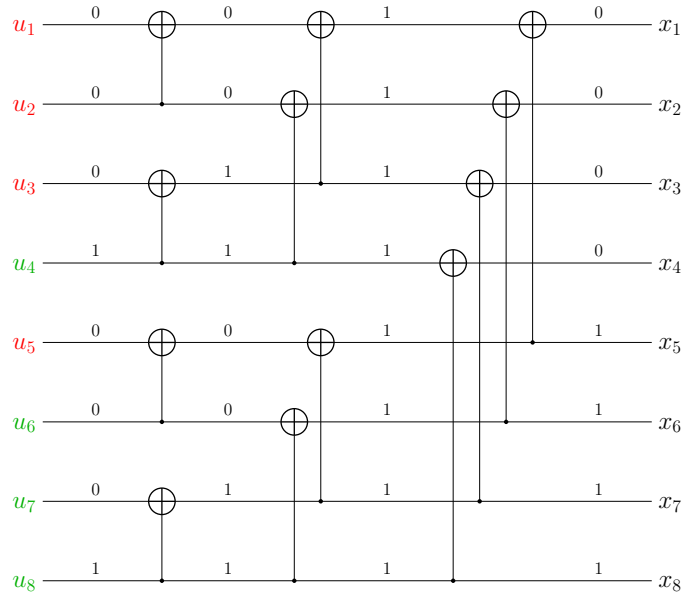

Fig. 13: Example of the polar encoding process: An $N = 8$ polar code having $k = 4$, $\mathcal{F} = \{1, 2, 3, 5\}$ and $u_{\mathcal{F}} = (0\ 0\ 0\ 0)$ is used to encode the information bit sequence $u_{\mathcal{F}_c} = (1\ 0\ 0\ 1)$. The resulting codeword is $x_1^8 = (0\ 0\ 0\ 0\ 1\ 1\ 1\ 1)$, as also shown in Eq. (24).

generator matrix $G_N(\mathcal{F}_c)$ and the coset is determined by the vector $u_{\mathcal{F}} G_N(\mathcal{F})$.

Polar codes are closely related to the family of RM codes [59]–[61], since both rely on the encoder G_N of Eq. (22). Explicitly, given a pair of integers $0 \leq r \leq n$, there exists an RM code of codeword length $N = 2^n$ and information word length $k = \sum_{i=0}^{r} \binom{n}{i}$, whose generator matrix G_{RM} is a submatrix of G_N. This property is analogous to that of polar codes. However, while the generator matrix $G_N(\mathcal{F}_c)$ of a polar code corresponds to the most reliable rows of G_N, the generator matrix G_{RM} of an RM code consists of rows of G_N having Hamming weights $\geq 2^{m-r}$. Equivalently, we may say that we freeze the least reliable channels in polar codes, while we freeze the the lowest Hamming weight channels in RM codes. Consequently, polar codes approach the capacity, while the RM codes exhibit a high minimum distance. It is interesting to point out that the reliability-based selection of frozen bit-channels over a BEC coincides with the lowest Hamming weight channels for $n = 3$ and $n = 4$. Hence, the polar code of Eq. (24) is equivalent to the $(8, 4, 4)$ RM code. However, the benefits of polar codes begin to emerge as n increases [59].

IV. POLAR DECODERS

Since the inception of polar codes, intensive research efforts have been invested in improving the polar decoding algorithms from the algorithmic perspective as well as from the perspective of hardware implementations. Major contributions

Algorithmic Developments	Year	Hardware Implementations
Belief Propagation (BP) and Successive Cancellation (SC) decoder [17], [62]	2009	
BP decoding improved by exploiting overcomplete factor graphs [63]		
Trellis-based Maximum Likelihood (ML) decoder for short polar codes [64]		
Linear programming based polar decoder for BEC [65]	2010	
Log Likelihood Ratio (LLR)-based SC decoder [66], [67]	2011	Hardware implementations relying on improved scheduling for (LLR)-based SC decoder [66], [67]
Simplified Successive Cancellation (SSC) decoder [68]		
Successive Cancellation List (SCL) decoder [69]–[71]		
Successive Cancellation Stack (SCS) decoder [72]	2012	Pre-computed look-ahead scheduling for reducing the latency of SC decoder [73]
Cyclic Redundancy Check (CRC)-aided SCL (CA-SCL) and SCS (CA-SCS) decoders [70], [74]		
Adaptive CA-SCL decoder [75]		
Multistage polar decoder [76]		
ML sphere decoder for short polar codes [77]		
Successive Cancellation Hybrid (SCH) decoder [78]	2013	Semi-parallel architecture for SC decoder [79]
Maximum Likelihood Simplified Successive Cancellation (ML-SSC) decoder [80]		Scalable semi-parallel implementation of SC decoder [81], [82]
Soft CANcellation (SCAN) decoder [83], [84]		Field Programmable Gate Array (FPGA) implementation of BP decoder [85]
		Two-phase SC decoder architecture having a lower complexity and memory utilization, and a higher throughput [86]
		Overlapped SC decoder architecture [87]
ML-SSC improved [88]	2014	Flexible and high throughput architecture of improved ML-SSC decoder [88]
Modified BP [89], [90]		Hardware implementation of SCL decoder [91]
Early terminated BP decoder [92]:		Efficient partial-sum network architecture for semi-parallel SC decoder [93]
Low-latency CA-SCL decoder [94]		Architecture of SSC decoder [95]
Symbol-based SC and SCL decoders [96], [97]		Architecture of 2-bit SC decoder [98]
SC flip decoder [99]		
Error exponent of SCL investigated [100]	2015	Hardware architecture of LLR-based SCL decoder [101]
LLR-based SCL decoder [101]		Architecture of a low-latency multi-bit SCL decoder [102]
		Metric sorter architecture for SCL decoder [103]
		Low-latency SCL decoder relying on double thresholding based list pruning [104], [105]
Reduced-complexity early terminated BP decoder [106]	2016	Hardware Architecture of low-latency CA-SCL decoder [107] relying on [94]
The reduced latency ideas of [88] extended to the SCL decoder [108]		Implementation of adaptive throughput-area efficient SCL decoder invoking approximate ML decoding components [109]
Tree pruning for low-latency SCL decoding [110]		Sphere decoding based architecture of SCL decoder [111]
Reduced-complexity SCL decoder [112]		Improved metric sorter architecture for SCL decoder [113]
Simplified Successive Cancellation List (SSCL) decoder avoiding redundant calculations [114]		Hardware implementation of CA-SCL based on distributed sorting [115]
Parity-check-aided polar decoding [116]		
Reduced latency SSCL decoder [117]	2017	Hardware implementation of SSCL decoder [117]
Unsorted SCL decoder [118]		Hardware implementation of low-latency BP decoder [119]
Syndrome-based SC decoder [120]		Two-step metric sorter architecture for parallel SCL decoder [121]
Soft SCL decoder for systematic codes [122], [123]		Memory efficient architectures for SC and SCL decoders [124]
Reduced-latency SSC flip decoder [125]	2018	
BP list decoder [126]		Hardware architecture of multi-bit double thresholding SCL decoder relying on pre-computed look-ahead scheduling [127]

TABLE I: Major contributions to the polar decoding paradigm.

in this context are chronologically summarized in Table I. In this section, we will review the achievements of Table I with an emphasis on the major decoding algorithms identified in Fig. 14.

A. Successive Cancellation Decoders

Arikan's seminal paper [17] proposed a Likelihood Ratio (LR) based Successive Cancellation (SC) polar decoder, which was later modified by Leroux *et al.* [66], [67] to carry out operations in the logarithmic domain; hence reducing the associated computational complexity. Recall from Fig. 12 that the compound channel W^N may be split into N polarized bit-channels such that the ith bit-channel W_i takes the input
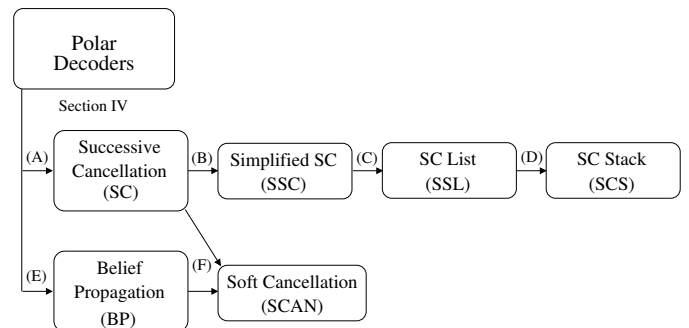
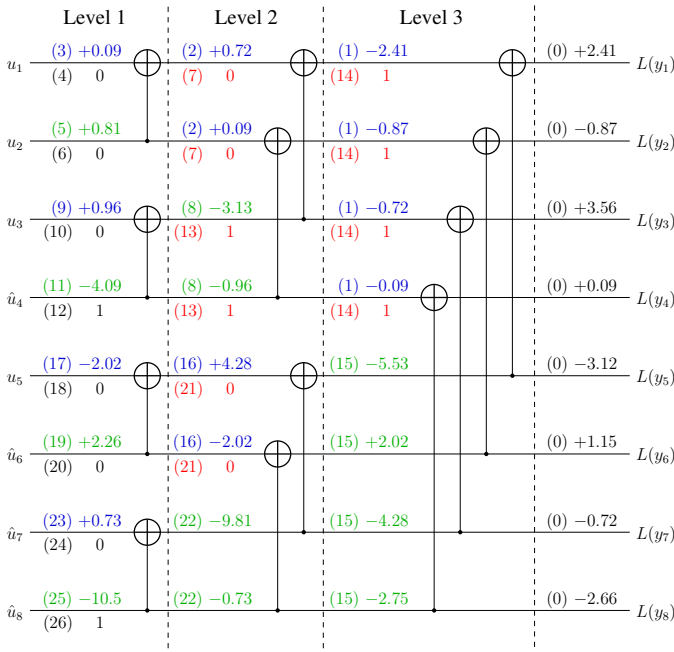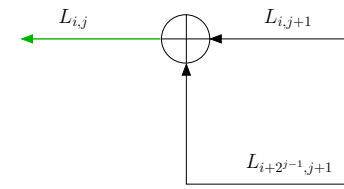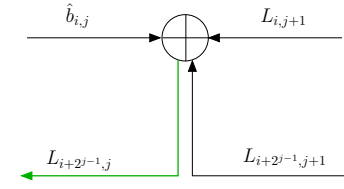

Fig. 14: Polar decoding algorithms.

Fig. 15: Example of the SC decoding process: An $N = 8$ polar code having $k = 4$, $\mathcal{F} = \{1, 2, 3, 5\}$ and $u_{\mathcal{F}} = (0\ 0\ 0\ 0)$ is used to decode the received encoded LLRs $L(y_i)$ into the $k = 4$ recovered information bits $\hat{u}_{\mathcal{F}_c} = (1\ 0\ 0\ 1)$. The LLRs obtained using the f and g functions of Eq. (27) and Eq. (30) are shown above each connection in blue and green, respectively. The bits obtained using the partial sum computations of Eq. (31) and Eq. (32) are shown below each connection in red. The accompanying numbers in parenthesis identify the step of the SC decoding process where the corresponding LLR or bit becomes available.

(a) Function $f\left(L_{i,j+1}, L_{i+2^{j-1},j+1}\right)$: LLRs propagate from right-to-left.

(b) Function $g(L_{i,j+1}, L_{i+2^{j-1},j+1}, \hat{b}_{i,j})$: switch from propagating bits to propagating LLRs.
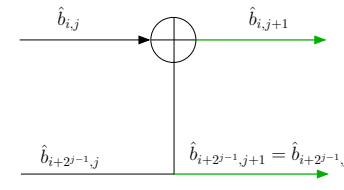
(c) Partial sum calculation XOR $(\hat{u}_1, \hat{u}_2)$: bits propagate from left-to-right.

Fig. 16: The three computations that can be performed for an XOR in the polar decoder. $L_{i,j}$ is the LLR pertaining to the bit $b_{i,j}$, while $\hat{b}_{i,j}$ is the estimation of bit $b_{i,j}$.

u_i and yields the output (y_1^N, u_i^{i-1}). The associated channel transition probabilities are denoted by $P_i(y_1^N, u_1^{i-1}|u_i)$, which may be estimated using an SC decoder. To elaborate, the channel combining process of Fig. 11 couples together the input bits u_i^N. So, an SC decoder reverses this process at the receiver by removing the contribution, or more precisely interference, of the bits u_1^{i-1} from the received coded bits y_1^N, hence revealing the value of the ith bit u_i.

An SC decoder operates on the same circuit as that of the encoder, as exemplified in Fig. 15 for the polar encoder of Fig. 13. However, while an encoder always processes the bits from left to right, an SC decoder operates from right to left as well as from left to right. To elaborate, an SC decoder performs computations pertaining to the XORs in the circuit according to a sequence that is dictated by the availability of data on the left and right hand side of the XOR, which introduces data dependencies in the decoding process. Hence, the functionality of each XOR in the decoding circuit varies, when performing operations on LLRs at different steps in the

SC decoding process[5]. There are three types of computations that can be performed by a particular XOR in the decoding circuit, depending on the availability of LLRs provided on the connections on its right-hand side, as well as upon the availability of bits provided on the connections on its left-hand side. Let us exemplify this by considering the 2-bit elementary kernel of Fig. 15, which operates on the ith and $(i + 2^{j-1})$th bits, where $j \in [1, n]$ denotes the level index.

The first occasion when an XOR can contribute to the SC decoding process is when an LLR has been provided by each of the connections on its right-hand side, as shown in Fig. 16a. Since the XOR connects the ith and $(i + 2^{j-1})$th bits, we refer to the first and second of these two LLRs as $L_{i,j+1}$ and $L_{i+2^{j-1},j+1}$, respectively. More specifically, $L_{i,j+1}$ and $L_{i+2^{j-1},j+1}$ provide soft-information pertaining to the bits $b_{i,j+1}$ and $b_{i+2^{j-1},j+1}$, respectively. These LLRs may be generated either by the soft demodulator (for $j = n$) or by the other XORs in the circuit (for $j < n$). Based on the input

---

[5]The LLR $L(b)$ pertaining to the bit b is defined as:

$$L(b) = \log\left(\frac{P(b = 0)}{P(b = 1)}\right),$$

where $P(.)$ denotes the probability of occurrence.

$L_{i,j+1}$ and $L_{i+2^{j-1},j+1}$, the XOR of Fig. 16a computes the LLR $L_{i,j}$ for the first of the two connections on its left-hand side, as follows:

$$L_{i,j} = f\left(L_{i,j+1}, L_{i+2^{j-1},j+1}\right)$$
$$= L_{i,j+1} \boxplus L_{i+2^{j-1},j+1}, \qquad (27)$$

where the box-plus operator is defined as [128]:

$$L(b_1) \boxplus L(b_2)$$
$$= L(b_1 \oplus b_2)$$
$$= \ln \frac{1 + e^{L(b_1)} e^{L(b_2)}}{e^{L(b_1)} + e^{L(b_2)}}$$
$$= 2 \tanh^{-1}(\tanh(L(b_1)/2)\tanh(L(b_2)/2)) \qquad (28)$$
$$= \text{sign}(L(b_1))\text{sign}(L(b_2)) \min(|L(b_1)|, |L(b_2)|)$$
$$+ \log\left(1 + e^{-|L(b_1)+L(b_2)|}\right) - \log\left(1 + e^{-|L(b_1)-L(b_2)|}\right)$$
$$\approx \text{sign}(L(b_1))\text{sign}(L(b_2)) \min(|L(b_1)|, |L(b_2)|). \qquad (29)$$

Here, $L(b_1)$ and $L(b_2)$ are the LLRs pertaining to the bits b_1 and b_2, respectively. The $\text{sign}(\cdot)$ of Eq. (29) returns -1 if its argument is negative and $+1$ if its argument if positive. Here, Eq. (29) is referred to as the min-sum approximation.

Later in the SC decoding process, the estimated bit $\hat{b}_{i,j}$ is provided on the first of the connections on the left-hand side of the XOR, as shown in Fig. 16b. Together with the LLRs $L_{i,j+1}$ and $L_{i+2^{j-1},j+1}$ that were previously provided using the connections on the right-hand side, this enables the XOR to compute the LLR $L_{i+2^{j-1},j}$ for the second of the two connections on its left-hand side, according to the g function as follows:

$$L_{i+2^{j-1},j} = g(L_{i,j+1}, L_{i+2^{j-1},j+1}, \hat{b}_{i,j})$$
$$= (-1)^{\hat{b}_{i,j}} L_{i,j+1} + L_{i+2^{j-1},j+1}. \qquad (30)$$

We may observe in Eq. (30) that the g function is analogous to the decoding operation of a repetition node, since the two LLR values are summed together. This is because the information pertaining to the bit $b_{i+2^{j-1},j}$ is contained in $L_{i+2^{j-1},j+1}$ as well as in $L_{i,j+1}$. Furthermore, the sign of $L_{i,j+1}$ is flipped when $\hat{b}_{i,j} = 1$, since we have $b_{i+2^{j-1},j} = b_{i,j+1} \oplus b_{i,j}$.

Later still, the bit $\hat{b}_{i+2^{j-1},j}$ will be provided on the second of the connections on the left-hand side of the XOR, as shown in Fig. 16c. Together with the bit $\hat{b}_{i,j}$ that was previously provided using the first of the connections on the left-hand side, this enables the partial sum computation of bits $\hat{b}_{i,j+1}$ and $\hat{b}_{i+2^{j-1},j+1}$ for the first and second connections on the right-hand side of the XOR, where

$$\hat{b}_{i,j+1} = \text{XOR}(\hat{b}_{i,j}, \hat{b}_{i+2^{j-1},j}), \qquad (31)$$
$$\hat{b}_{i+2^{j-1},j+1} = \hat{b}_{i+2^{j-1},j}. \qquad (32)$$

As may be appreciated from the discussions above, the f function of Eq. (27) may be used to propagate LLRs from right-to-left within the SC decoder, while the partial sum computations of Eq. (31) and Eq. (32) may be used to propagate bits from left-to-right and the g function of Eq. (30)

may be used to switch from propagating bits from left-to-right to propagating LLRs from right-to-left. The SC decoding process begins by processing LLRs from right to left. However, in order that LLRs can be propagated from right to left, it is necessary to provide LLRs on the connections on the right-hand edge of the circuit, i.e. right-hand connections at level 3 of Fig. 15. In the example of Fig. 15, this is performed at the start of the SC decoding process by providing successive LLRs from a soft demodulator on successive connections on the right-hand edge of the circuit. We may also call them channel LLRs, since they provide soft information pertaining to the channel outputs. The SC decoding process then begins by using the f function of Eq. (27) to propagate LLRs from the right hand edge of the decoding circuit to the top connection on the left-hand edge, allowing the first bit to be recovered (steps (0) to (4) in Fig. 15). Explicitly, if the first bit is an information bit, then a hard-decision is made based on the resulting LLR $L_{1,1}$. By contrast, if the first bit is a frozen bit, then it is set equivalent to the known frozen bit. Then the g function of Eq. (30) is used to compute the LLR pertaining to the second bit, hence revealing its value (steps (5) and (6) in Fig. 15). Following this, each successive bit from top to bottom is recovered by using the partial sum computations of Eq. (31) and Eq. (32) to propagate bits from left to right, then using the g function of Eq. (30) for a particular XOR to switch from bit propagation to LLR propagation, before using the f function to propagate LLRs to the next connection on the left-hand edge of the circuit, allowing the corresponding bit to be recovered. It is pertinent to mention here that if bit on the left-hand edge is a frozen bit, then the associated the LLR is ignored and the value of the bit is set to the known value of the frozen bit. The complexity of this decoding process is on the order of $O(N \log_2 N)$, since there are $n = \log_2 N$ levels and each level invokes $N/2$ XOR gates. Furthermore, a straightforward implementation of the SC decoder also has a chip-area proportional to $O(N \log_2 N)$, which was reduced to $O(N)$ in [70] by exploiting the recursive nature of polar codes in conjunction with 'lazy-copy' algorithmic techniques.

The SC decoding process of Fig. 15 may also be visualized over a decoding tree, as shown in Fig. 17. The decoding tree of Fig. 17 consists of $n = 3$ levels and each level is composed of 2^{n-i} parent nodes and 2^{n-i+1} child nodes; hence resulting in $2^3 = 8$ leaf nodes at level 1, which correspond to the bits u. To elaborate, the number of child nodes at each level corresponds to the number of distinct polarized channels created at that level. Recall from Section II that we get two types of polarized channels W^- and W^+ at level 3, which are further polarized into W^{--}, W^{-+}, W^{+-} and W^{++} at level 2 and then into eight types at level 1. This results in 2^{n-i+1} types of polarized channels at each level. Furthermore, the decoding tree at level 3 starts with a length N parent node, whose length reduces by half at every child node, hence adopting a recursive divide-and-conquer approach.

Let us now elaborate on the flow of LLRs and bits through the decoding tree of Fig. 17, where each node acts as a local decoder executing the XOR operations of Fig. 16. The SC decoding process begins with the parent node at Level 3, which

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2937923, IEEE Communications Surveys & Tutorials
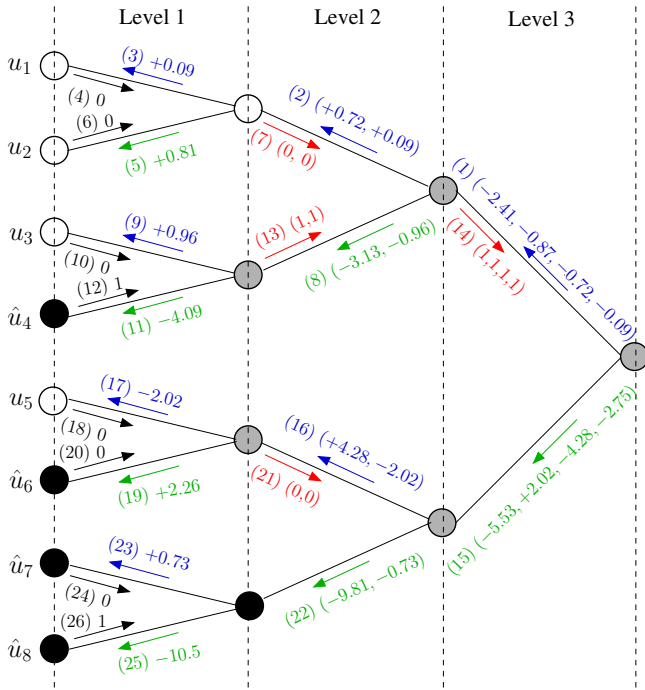
13

Fig. 17: SC decoding tree for the SC decoding circuit of Fig. 15. Rate-0 and Rate-1 nodes are shown in white and black color, respectively. The LLRs obtained using the f and g functions of Eq. (27) and Eq. (30) are shown in blue and green, respectively, while the bits obtained using the partial sum computations of Eq. (31) and Eq. (32) are shown in red. The accompanying numbers in parenthesis identify the step of the SC decoding process where the corresponding LLR or bit becomes available.

has the channel LLRs from the soft demodulator. This parent node generates LLRs for the top child node using the function f of Fig. 16a. It then waits until it receives the hard-decoded bits from the top child node and transfers the parental control to the top child node, which now acts as the next parent node. This process continues in a recursive manner until we reach the leaf nodes at level 1. At this point, hard decision is made pertaining to the uncoded bits u and sent to the parent node at level 1. On receiving the hard-decoded bits, the parent node at level 1 calculates the LLRs for the bottom child node using the g function of Fig. 16b and waits for the hard-decoded bits from the bottom child node. On receiving the bits, the parent node at level 1 executes the XOR operation of Fig. 16c and sends the resulting bits to its parent node at Level 2. The process continues recursively until all the uncoded bits u have been recovered.

SC decoders are favored for having a low decoding complexity. However, this is achieved at the cost of a high decoding latency, since there are several data dependencies associated

with the SC decoding process. Explicitly, the f operations have to wait for the LLRs to be made available on their right-hand connections, while the g operations have to wait for the availability of the estimated bit values on their left-hand connections, in addition to the availability of the LLRs on their right-hand connections. Similarly, it is necessary to provide bits on the left-hand in order to facilitate the propagation of bits from left to right. Hence, owing to these data dependencies, the information bits on the left-hand edge of the circuit are serially recovered from top to bottom. This in turn makes the hardware implementation of SC decoders challenging. More specifically, the data dependencies allow different numbers of operations to be completed in parallel at different times, as illustrated in the example of Fig. 15. In order to minimise the number of steps required to complete the decoding process, a large amount of hardware may be used so that a single processing step is sufficient to complete the largest number of parallel operations that are supported by the decoder data dependencies. However, the data dependencies will prevent much of this hardware from being used throughout the rest of the decoding process, which may motivate the use of a smaller amount of hardware and a greater number of steps. However, either way, the ratio of hardware resource usage to the latency required to complete the decoding process may be unfavorable, unless sophisticated alternative techniques can be developed and utilised. In this context, Leroux et al. [66], [67] proposed hardware architectures for SC decoders, which rely on improved scheduling for enhancing resource sharing and memory management. In [73], pre-computed look-ahead techniques were invoked for reducing the latency of SC decoding process, while semi-parallel implementations of SC decoders were presented in [79], [81], [82]. A two-phase SC decoder architecture was conceived in [86], which exhibits lower complexity, memory utilization and latency, while an overlapped SC decoder architecture was presented in [87] for the sake of reducing the latency. Furthermore, Fan et al. [93] developed an efficient partial-sum network architecture for semi-parallel SC decoders. In the spirit of further reducing the latency, Yuan et al. [98] proposed a 2-bit decoding architecture for SC decoders, which concurrently processes two bits during the last stage of the SC decoding process. Look-ahead techniques were also invoked in [98] and recently in [129], while memory-efficient hardware implementations were presented in [124], [130].

The main characteristics of an SC polar decoder are summarized in Table II.

B. Simplified Successive Cancellation Decoders

The SC decoding process of Section IV-A consists of some redundant calculations, which may be discarded without compromising the BER or BLER performance. Based on this notion, a Simplified Successive Cancellation (SSC) decoder was conceived in [68]. As compared to the classic SC decoder, the SSC decoder provides significant reduction in the computational complexity as well as the latency, while maintaining the same BER or BLER performance. Quantitatively, it was demonstrated in [68] that the SSC decoder reduces the number

Complexity	• Time complexity = $O(N \log_2 N)$ • Space complexity = $O(N)$
Advantages	• Low decoding complexity • Asymptotically capacity achieving
Disadvantages	• Sub-optimal finite-length performance • Serial processing, resulting in high latency (or low throughput) • Fully-parallel implementation not feasible

TABLE II: Main characteristics of an SC polar decoder.

Complexity	Depends on the underlying polar code
Advantages	• Lower complexity and latency than SC • Asymptotically capacity achieving
Disadvantages	• Sub-optimal finite-length performance • Serial processing, resulting in high latency (or low throughput) • Fully-parallel implementation not feasible

TABLE III: Main characteristics of an SSC polar decoder.

of computationally intensive box-plus operators of Eq. (28) by around 20% to 50%, while the decoding latency is reduced by around 75% to 95%.

The SSC decoder exploits the tree structure of Fig. 17 for discarding redundant computations. Explicitly, the nodes of Fig. 17 may be classified into three types, as follows:

1) Rate-0 nodes, whose descendants are all frozen bits (white nodes in Fig. 17);
2) Rate-1 nodes, whose descendants are all information bits (black nodes in Fig. 17);
3) Rate-R nodes, whose descendants are a mix of frozen bits and information bits (gray nodes in Fig. 17).

Recall that the value of frozen bits may be directly used at the decoder rather than estimating it based on the computed LLRs. Consequently, the SSC decoder de-activates the f and g operations at Rate-0 nodes of Fig. 17 without affecting the decoder's performance. Explicitly, u_1 and u_2 are initialized to the known value of frozen bits, i.e. 0, in the SSC decoder, and steps (3) to (6) are not required. Similarly, the SSC decoder further reduces the complexity at the Rate-1 nodes by making a hard-decision based on the input LLRs and passing the hard-decoded bits to the child nodes, rather than computing the f and g functions. In the context of Fig. 17, an SSC decoder makes hard-decision based on the LLRs computed in step (22) revealing $\hat{u}_8$, discards steps (23) to (26), and estimates $\hat{u}_7$ based on the hard-decision values of step (22). Since all nodes connected to the Rate-1 nodes are information bits, this simplification does not affect the performance of the decoder. Improved versions of the SSC decoder were proposed in [80], [88] for the sake of further reducing the associated latencies, while the hardware architectures of SSC decoder were presented in [88], [95].

The main characteristics of an SSC polar decoder are summarized in Table III.

C. Successive Cancellation List (SCL) Decoders

The SC decoder provably achieves the capacity of a B-DMC, when the codeword length is infinitely long. However, it does not exhibit good performance at finite codeword lengths, because the channels are not sufficiently polarized. More specifically, in the SC decoding process, the value of each

recovered information bit $\hat{u}_i$ depends on all the previous recovered information bits $\hat{u}_1^{i-1}$. Consequently, if a bit is incorrectly decoded, it will often catastrophically propagate the errors to all subsequent bits. The selection of an incorrect value for an information bit may be detected with consideration of the subsequent frozen bits, since the decoder knows that these bits should have values of 0. More specifically, if the corresponding LLR has a sign that would imply a value of 1 for a frozen bit, then this suggests that an error may have been made during the decoding of one of the preceding information bits. However, in the SC decoding process, there is no opportunity to consider alternative values for the preceding information bits. Once a value has been selected for an information bit, the SC decoding process moves on and the decision is final.

Inspired by the recursive list decoding of RM codes [131], Tal and Vardy proposed an LR-based SCL decoder [69], [70], whose LLR-based counterpart was presented in [101]. In contrast to an SC decoder, an SCL decoder considers a list of alternative values for the information bits, hence improving the finite-length performance of polar codes. More explicitly, as the decoding process progresses, an SCL decoder considers both options for the value of each successive information bit, rather than making a hard-decision based on the associated LLR value. This is achieved by maintaining a list of candidate information bits, where the list is built up as the SCL decoding proceeds. At the start of the process, the list is empty. Whenever the decoding process reaches a frozen bit, a bit value of 0 is appended to the list. However, whenever the decoding process reaches an information bit, two replicas of the list are created. Here, the bit value of 0 is appended to the first replica, while the bit value of 1 is appended to the second replica. Hence, the number of lists, or more specifically the number of candidate decoding paths, doubles whenever an information bit is encountered. This continues until the number of decoding paths reaches a limit L, which is known as the list size and is typically chosen as a power of two. From this point onwards, each time the number of decoding paths is doubled when considering an information bit, the worst L amongst the $2L$ candidate paths are identified and pruned from the list. In this way, the size of the list is maintained at L until the SCL decoding process completes. A straightforward implementation of the SCL algorithm incurs a complexity polynomial in the codeword length N. However, Tal and Vardy [70] exploited the recursive nature of polar codes
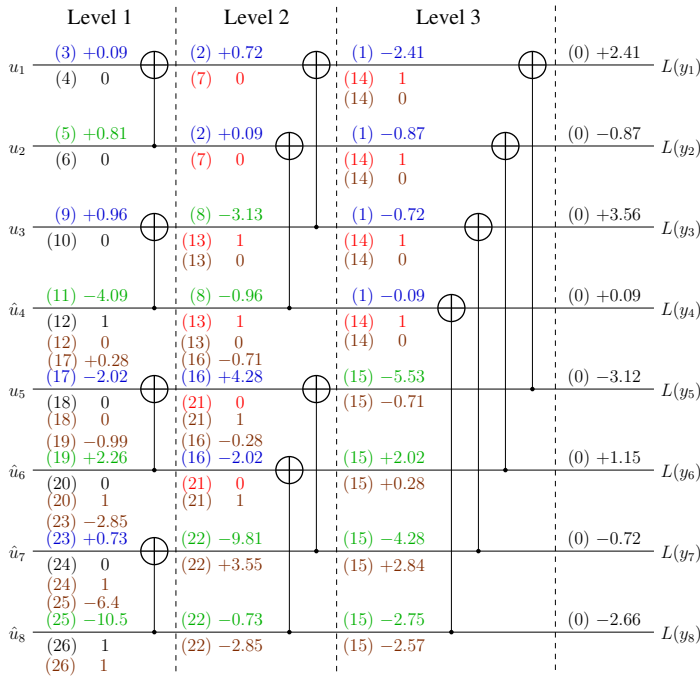
(a) An SCL ($L = 2$) decoding circuit. For the first candidate path, the LLRs obtained using the f and g functions of Eq. (27) and Eq. (30) are shown above each connection in blue and green, respectively, while the bits obtained using the partial sum computations of Eq. (31) and Eq. (32) are shown below each connection in red. All the LLRs and bits pertaining to the second candidate path are shown in brown. The accompanying numbers in parenthesis identify the step of the SC decoding process where the corresponding LLR or bit becomes available.
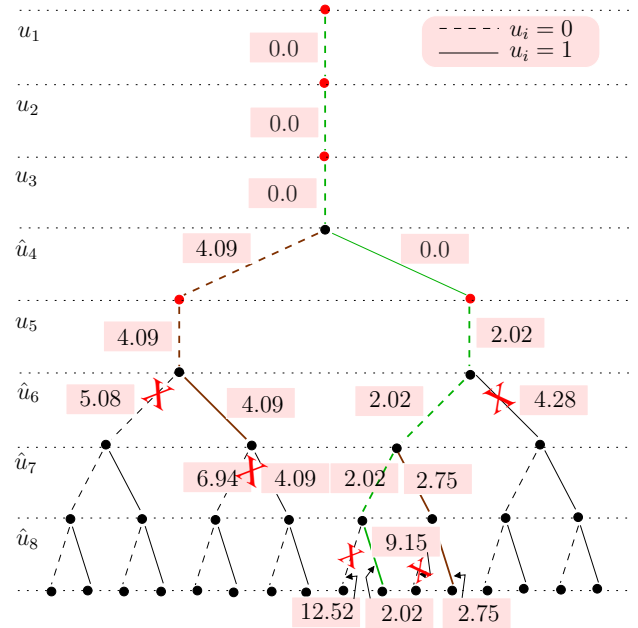
(b) SCL decoding over a binary tree. Frozen and information nodes are marked in red and black, respectively, the two candidate paths are shown in green and brown, and the pruned paths are marked with a red cross. The values next to the branches are the associated path metrics computed using Eq. (34). Please note that the green path is the same as the SC decoding path.

Fig. 18: Example of an SCL decoding process ($L = 2$): An $N = 8$ polar code having $k = 4$, $\mathcal{F} = \{1, 2, 3, 5\}$ and $u_{\mathcal{F}} = (0\ 0\ 0\ 0)$ is used to decode the received encoded LLRs $L(y_i)$ into the $k = 4$ recovered information bits $\hat{u}_{\mathcal{F}_c} = (1\ 0\ 0\ 1)$.

together with 'lazy-copy' algorithmic techniques to reduce the time complexity to $O(LN \log_2 N)$ and the space complexity to $O(LN)$, both of which are L times the complexities of the classic SC decoder.

The SCL decoding process may be viewed as a path search in a binary tree of depth N, as illustrated in Fig. 18b for the decoding circuit of Fig. 18a. The input bits u_i may be successively recovered as we move down the binary tree. Explicitly, the nodes of the binary tree (except for the leaf nodes) of Fig. 18b correspond to the input u_i, while the branches represent the possible values 0 and 1 of the bit u_i. Consequently, all the information nodes (marked in black in Fig. 18) have two branches, while the frozen nodes (marked in red in Fig. 18) have a single branch, since their values are known to the decoder. The lth branch at depth $i \in [1, N]$ is identified with a path metric $\phi_{l,i}$, which is calculated as

follows:

$$\phi_{l,i} = \phi_{l,i-1} + \ln(1 + e^{-(1-2\hat{u}_{l,i})L(u_{l,i})}) \tag{33}$$

$$\approx \begin{cases} \phi_{l,i-1} & \text{if } \hat{u}_{l,i} = \frac{1}{2}(1 - \text{sign}(L(u_{l,i}))) \\ \phi_{l,i-1} + |L(u_{l,i})| & \text{otherwise} \end{cases}, \tag{34}$$

where $\phi_{l,i-1}$ is the parent path's metric at depth $(i-1)$, $\hat{u}_{l,i}$ is the value of u_i associated with the lth branch and $L(u_{l,i})$ denotes the corresponding LLR $L_{i,1}$ obtained on the left-hand edge of the polar decoding circuit of Fig. 18a. These LLRs are obtained throughout the SCL decoding process by using separate replicas of the partial sum computations of Eq. (31) and Eq. (32) to propagate the bits of each candidate path from left to right. Similarly, separate replicas of the f and g computations of Eq. (27) and Eq. (30) are used to propagate corresponding replicas of the LLRs, as shown in Fig. 18a in brown color for the second candidate path. Here, Eq. (34) is referred to as the min-sum approximation. Intuitively, Eq. (34) implies that if the value of bit $\hat{u}_{l,i}$ corresponding to the branch
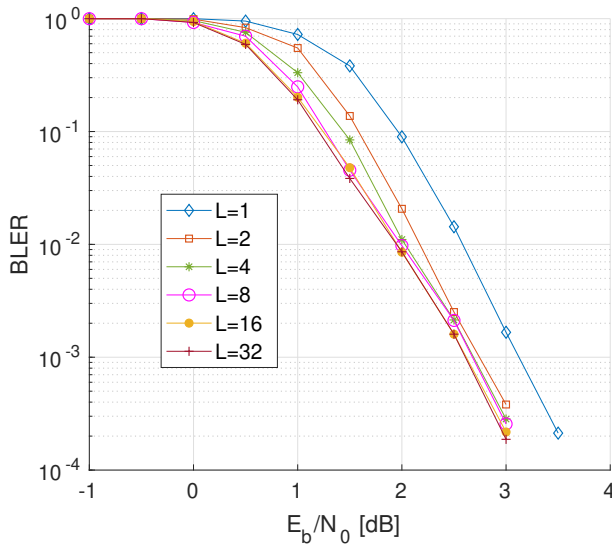
Fig. 19: The achievable BLER performance of the SCL decoder for variable list sizes. A $(1024, 512)$ polar code relying on the 5G frozen bit sequence was used in conjunction with QPSK transmission over an AWGN channel. Min-sum approximation of Eq. (29) was invoked for calculating box-plus operations.

at depth i complies with the LLR $L(u_{l,i})$, then the path metric is left unchanged, otherwise it is penalized by $|L(u_{l,i})|$. Furthermore, Eq. (34) accumulates across all bit positions $i \in [1, N]$. So, it must be calculated for all L candidate paths whenever a frozen bit value of 0 is appended, as well as for all $2L$ candidates when both possible values of an information bit are considered. In the latter case, the $2L$ metrics are sorted and the L candidates having the highest values are identified as being the worst and are pruned from the list, as shown in Fig. 18b for $L = 2$. Following the completion of the SCL decoding process, the candidate path having the lowest metric may be selected as the most likely decoding path. It is pertinent to mention here that an SCL decoder having $L = 1$ is the same as an SC decoder and it becomes equivalent to a Maximum Likelihood (ML) decoder, when $L = 2^k$, implying a search through all possible 2^k codewords.

Fig. 19 records the BLER performance of the SCL decoder for variable list sizes L. We may observe in Fig. 19 that the BLER performance improves upon increasing the value of L, providing significant improvement over the SC decoder ($L = 1$). However, the performance improves with diminishing returns at higher values of L. More precisely, the performance improvement is only marginal for $L > 8$. Hence, a list size of 8 is deemed sufficient, while a list size of 32 brings the performance arbitrary close to that of the ML decoder, as demonstrated in [70].

Despite the improved performance of the SCL decoder, polar codes failed to outperform the state-of-the-art LDPC and turbo codes [70], which may be attributed to their poor

minimum distance. More specifically, the authors of [74] observed that when the SCL decoder failed to output the correct information sequence, the correct information sequence was usually present in its list of candidate paths, but with a smaller path metric. This motivated the development of the Cyclic Redundancy Check-Aided SCL (CA-SCL) decoder [70], [74], while an adaptive CA-SCL decoding scheme was developed in [75]. The Cyclic Redundancy Check (CRC)-aided approach concatenates a CRC to the polar code as an outer code, so that a portion of the information bits is utilized for carrying the CRC bits. More specifically, $(k - r)$ information bits are first encoded using a CRC code, which appends an r-bit CRC to the $(k - r)$ information bits. This CRC code can simply be a small random systematic linear block code. The resulting CRC encoded bits are then passed to the inner polar code. At the CA-SCL decoder, all candidate paths that do not satisfy the CRC are pruned during the last stage of the classic SCL decoding process, before the candidate having the lowest metric is selected as the most likely decoding path. The CA-SCL decoder improved the performance of polar codes to be on par with the state-of-the-art turbo and LDPC codes, while retaining the complexity of the classic SCL decoder. However, this is achieved at the cost of a small loss in the coding rate, since the coding rate of the resultant concatenated code is $(k - r)/N$. The notion of CRC-aided concatenation was generalized to parity check codes in [116].

In pursuit of approaching the perfromance of the SCL decoder with the SC decoder's space requirements, bit-flip decoding strategy was adopted in [99]. The proposed SC flip decoder of [99] enhances the performance of the classic SC decoder by exploiting CRC in conjunction with the bit-flipping algorithm. More specifically, the SC flip decoder commences the decoding process by running a single iteration of the classic SC decoder for estimating the codeword $\hat{u}$, as exemplified in Fig. 15. If the CRC passes with the estimated codeword $\hat{u}$, the decoding process is terminated. Otherwise, upto T additional SC decoding iterations are invoked, sequentially flipping the estimated value of one of the T least reliable bits in each iteration. The process is repeated until the CRC passes or the maximum number of attempts T is reached. While the SC flip decoder preserves the space requirements of the classic SC decoder as well as improves the decoding performance, it fails to outperform the SCL decoder, since the bit-flip algorithm is only capable of correcting a single additional error. Furthermore, the computational complexity is substantially high at low SNRs and approaches that of the SC decoder at high SNRs. Improvements to the SC flip decoder were proposed in [132]–[134], while the use of the bit-flip algorithm was extended to the SSC and the SCL decoders in [125], [135] and [136].

Similarly to the classic SC decoder, the SCL decoding scheme also suffers from a high latency due to the data dependencies associated with the decoding process. In this context, various low-latency SCL decoding schemes have been presented in [94], [104], [105], [108], [110], [111], [117]. Furthermore, the complexity of an SCL decoder is L times higher than that of a classic SC decoder, since it processes L

Complexity	• Time complexity = $O(LN \log_2 N)$ • Space complexity = $O(LN)$
Advantages	• Capable of achieving the ML performance • Performance of CA-SCL is on par with turbo and LDPC codes
Disadvantages	• Serial processing, resulting in high latency (or low throughput) • Fully-parallel implementation not feasible • Higher complexity than SC • Higher latency than SC due to the 'metric sorting' operation

TABLE IV: Main characteristics of an SCL polar decoder.

candidate paths. For the sake of reducing the complexity of SCL decoders, reduced-complexity techniques were explored in [112], [114]. Another key challenge in the SCL decoding process is imposed by metric sorting, which is required to identify the worst L candidate paths among the merged list of $2L$ paths for pruning. Simultaneously comparing all the $2L$ path metrics requires large amount of hardware resources, while successively comparing the paths increases the latency. Hence, a suitable compromise must be reached between the required hardware resources and the imposed latency. Hardware implementations of the SCL decoder have been discussed in [91], [94], [101]–[103], [109], [113], [115], [117], [121], [124], [127], while an unsorted SCL decoder was conceived in [118].

The main characteristics of an SCL polar decoder are summarized in Table IV.

D. Successive Cancellation Stack (SCS) Decoders

The SC decoding may be viewed as a greedy search over the binary tree of Fig. 18b, since it only considers the path with the lowest metric, hence making a bit-by-bit decision. By contrast, the SCL decoding adopts a breadth-first approach, doubling the number of paths for each information bit and selecting the best L paths for further processing. The breadth-first approach of SCL provides attractive performance benefits at the cost of an increased computational complexity, because L paths have to be processed in contrast to the single decoding path of SC. For the sake of achieving a reasonable compromise between the performance and computational complexity, Successive Cancellation Stack (SCS) decoding was proposed in [72] , inspired by the stack decoding of convolutional codes [137] and RM codes [138].

Similarly to SCL decoding, SCS decoding also operates over the binary tree of Fig. 18b. However, while an SCL decoder records the L best candidate paths (having the lowest path metric) of the same length, an SCS decoder records the D best candidate paths of variable lengths in an ordered stack. Furthermore, in contrast to the SCL decoding, which processes all the L candidate paths in parallel, SCS decoding only processes the path at the top of the stack at a time, hence reducing the computational complexity. Let us elaborate on

this by revisiting the decoding example of Fig. 18 with the SSC algorithm.

Analogous to SCL decoding, the SCS decoding algorithm begins from the root node at depth $i = 1$ of the binary tree of Fig. 18b, computing the path metric $\phi_{l,1}$ according to Eq. (34) for all branches at depth 1. The resulting path, which is identified by the estimated information bit $\hat{u}_1$ and the path metric $\phi_{l,1}$, is stored in a stack, as shown in the first column of Table V. The stack is sorted in the order of increasing path metrics. However, since we only have a single path at time instant $t = 1$ in our example, sorting is not required. Thereafter, the algorithm moves along the binary tree of Fig. 18b by recursively invoking the following operations:

- **Pulling:** The top path from the stack, having the lowest path metric, is pulled out for further processing.
- **Extension:** The pulled path, having the estimated information bits $\hat{u}_1^i$ and the path metric $\phi_{l,i}$, is extended along the binary tree of Fig. 18b to include the next bit. If the next bit is a frozen bit, we get a single extended path by appending a 0 to the original path $\hat{u}_1^i$. By contrast, if the next bit is an information bit, we get two extended paths $\hat{u}_1^{i+1}$: one is obtained by appending a 0 to $\hat{u}_1^i$, while the other is obtained by appending a 1 to $\hat{u}_1^i$. Path metrics are calculated for the extended paths according to Eq. (34).
- **Deletion:** A stack can store at most D paths. Consequently, if $(D - 1)$ paths are already stored in the stack, the path at the bottom of the stack is deleted in order to make space for the two extended paths. This step may be omitted, if there is a single extended path.
- **Pushing:** The extended paths $\hat{u}_1^{i+1}$ are pushed in the stack along with their path metrics $\phi_{l,i+1}$. Please note that, consistent with the notation of Section IV-C, l denotes the lth branch in the binary tree of Fig. 18b.
- **Sorting:** The stack is sorted in order of increasing path metrics, so that the most reliable path having the lowest path metric appears at the top.

The aforementioned five operations are repeated until the top path of the stack reaches the leaf node of the binary tree of Fig. 18b. The resulting stack outputs are recorded in Table V for a stack size of $D = 4$.

The time and space complexities of an SCS decoder are $O(DN \log_2 N)$ and $O(DN)$, respectively. However, the actual complexity of an SCS decoder depends on the channel SNR. More specifically, at high SNRs, when the received information is less noisy, the SCS decoder converges faster, approaching the complexity of the classic SC decoder. By contrast, at low SNRs, the complexity of an SCS decoder approaches that of an SCL decoder having $L = D$. Nonetheless, SCS decoders are shown to have a lower complexity than the SCL decoder at the desired BLER of 10^{-3} [72], but this is achieved at the cost of high space complexity. More specifically, a significantly high value of D is required to match the performance of the SCL decoder of a given list size L. For example, a depth size of $D = 100$ was used in [72] for the SCS decoder to match the performance of an SCL decoder having $L = 20$. In pursuit of

$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$
$\hat{u}_1$ $\phi_{1,1}=0.0$	$\hat{u}_1^2$ $\phi_{1,2}=0.0$	$\hat{u}_1^3$ $\phi_{1,3}=0.0$	$\hat{u}_1^4$ $\phi_{2,4}=0.0$	$\hat{u}_1^5$ $\phi_{2,5}=2.02$	$\hat{u}_1^6$ $\phi_{3,6}=2.02$	$\hat{u}_1^7$ $\phi_{5,7}=2.02$	$\hat{u}_1^8$ $\phi_{10,8}=2.02$
			$\hat{u}_1^4$ $\phi_{1,4}=4.09$	$\hat{u}_1^4$ $\phi_{1,4}=4.09$	$\hat{u}_1^4$ $\phi_{1,4}=4.09$	$\hat{u}_1^7$ $\phi_{6,7}=2.75$	$\hat{u}_1^7$ $\phi_{6,7}=2.75$
					$\hat{u}_1^6$ $\phi_{4,6}=4.28$	$\hat{u}_1^4$ $\phi_{1,4}=4.09$	$\hat{u}_1^4$ $\phi_{1,4}=4.09$
							$\hat{u}_1^8$ $\phi_{9,8}=12.52$

TABLE V: An example of the SCS decoding process ($D = 4$) corresponding to the SCL decoding example of Fig. 18. Each column records the stack outputs at time instant t. The deleted path is highlighted in red, while the final optimal path is marked in green.

Complexity	• Time complexity = $O(DN \log_2 N)$ • Space complexity = $O(DN)$
Advantages	• Capable of achieving the ML performance when $D = 2^k$ • Performance of CA-SCS is on par with turbo and LDPC codes • Lower complexity than SCL at moderate and high SNRs
Disadvantages	• Serial processing, resulting in high latency (or low throughput) • Fully-parallel implementation not feasible • High time complexity at low SNRs • High space complexity • Higher latency than SC due to the 'metric sorting' operation

TABLE VI: Main characteristics of an SCS polar decoder.

combining the benefits of the SCL and SCS decoding schemes, a hybrid scheme called Successive Cancellation Hybrid (SCH) was proposed in [78]. The notion of CRC-aided decoding is also readily applicable to SCS decoding, resulting in the Cyclic Redundancy Check-Aided SCS (CA-SCS) decoder [74]. Furthermore, an efficient software implementation of the SCS decoder, relying on the LLRs, was presented in [139], which imposes a reduced time and space complexity. The decoding performance of the LLR-based SCS decoder of [139] was further improved in [140]. In particular, the improved SCS decoder of [140] incurs a substantially lower time complexity, while maintaining the same space complexity.

The main characteristics of an SCS polar decoder are summarized in Table VI.

E. Belief Propagation

All the above polar decoding schemes are derived from the classic SC decoder of Section IV-A, and suffer from the issue of serial processing, which imposes a high latency. Furthermore, all these coding schemes yield a hard decision output. Hence, these decoders are not suitable for iterative joint detection and decoding schemes, which require soft-in soft-out decoders. Fortunately, polar codes may also be represented using a factor graph, which permits soft-in soft-out Belief Propagation (BP) decoding [17]. Furthermore, the BP decoding algorithm is more amenable to parallel implementation.

Fig. 20 shows the factor graph representation of the polar circuit of Fig. 15. We may notice that Fig. 20 is obtained from Fig. 15 by replacing each 2-bit elementary kernel's circuit by
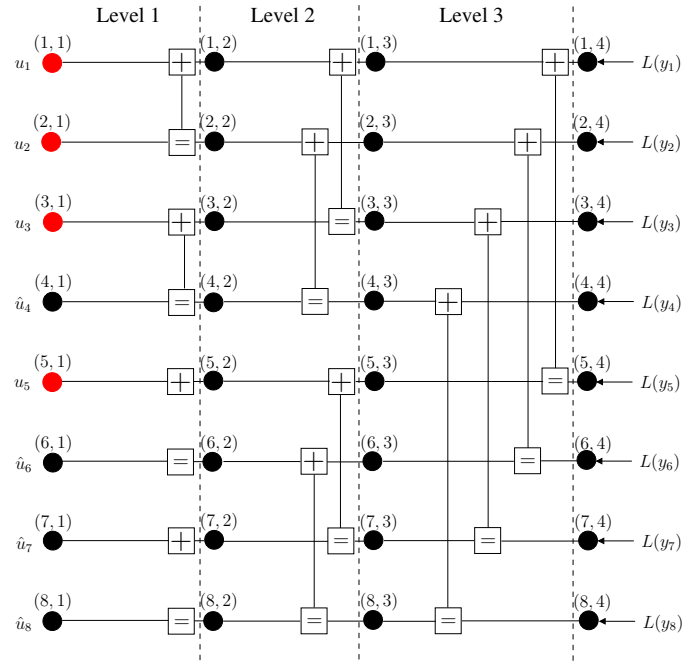
Fig. 20: Factor graph of an $N = 8$ polar code having $k = 4$ and $\mathcal{F} = \{1, 2, 3, 5\}$. Variable nodes and check nodes are denoted by circle and square, respectively. The frozen variable nodes on the left-hand edge of the factor graph are drawn in red.
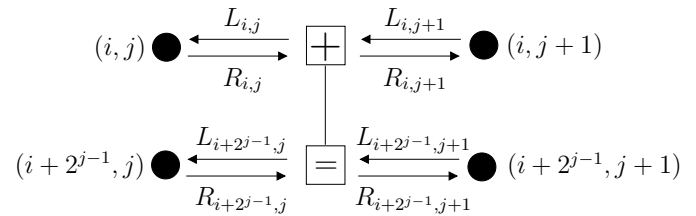
Fig. 21: Factor graph of the 2-bit elementary kernel of Fig. 20 at the jth level.

its factor graph, which is shown in Fig. 21. The resulting factor graph consists of $n = \log_2 N$ levels and $N \times (n + 1)$ variable nodes. Furthermore, each variable node is identified by the index (i, j) where i and j denote the bit index and level index, respectively.

The BP algorithm iteratively exchanges messages over the factor graph of Fig. 20 until the maximum number of decoding iterations $I_{\max}$ is reached. Explicitly, as shown in Fig. 21, two types of messages flow through the factor graph: the right-to-left (left) LLRs $L_{i,j}$ flowing towards the left of the factor graph and the left-to-right (right) LLRs $R_{i,j}$ flowing towards the right of the factor graph. These messages are computed during the BP algorithm as follows:

- **Initialization:** The left LLRs $L_{i,n+1}^t$, pertaining to the variable nodes on the right-hand end of the factor graph, are set to be equivalent to the channel LLRs for all decoding iterations $t \in [1, I_{\max}]$. Furthermore, all right LLRs $R_{i,j}^0$ are initialized to zero for the first decoding iteration, except for the LLRs pertaining to the frozen nodes on the left-hand edge of the factor graph ($j = 1$), which are set to infinity.

- **Right-to-left message exchange:** The BP algorithm processes the left LLRs $L_{i,j}^t$ from right to left, starting from the level $j = 3$. Explicitly, the left messages are computed as follows:

$$L_{i,j}^t = f(L_{i,j+1}^t, L_{i+2^{j-1},j+1}^t + R_{i+2^{j-1},j}^{t-1}) \qquad (35)$$

$$L_{i+2^{j-1},j}^t = L_{i+2^{j-1},j+1}^t + f(L_{i,j+1}^t, R_{i,j}^{t-1}), \qquad (36)$$

where $f(.)$ is given by Eq. (27), for all nodes at the jth level and the tth decoding iteration. The operation of Eq. (35) and Eq. (36) is encapsulated in Fig. 22a and Fig. 22b, respectively. We may notice that Eq. (35) reduces to the XOR operation of Fig. 16a, when $R_{i+2^{j-1},j}^{t-1}$ is set to zero, i.e. when we don't have any a-priori information about the bits $b_{i+2^{j-1},j}$. Similarly, Eq. (36) becomes equivalent to the g operation of Fig. 16b, when $R_{i,j}^{t-1}$ of Eq. (36) is replaced by its hard-decision value of plus or minus infinity corresponding to the recovered bit $\hat{b}_{i,j}$.

- **Left-to-right message exchange:** Following the right-to-left message exchange, the BP algorithm computes the right LLRs $R_{i,j}^t$ from left-to-right of the factor graph as follows:

$$R_{i,j+1}^t = f(R_{i,j}^t, L_{i+2^{j-1},j+1}^t + R_{i+2^{j-1},j}^t) \qquad (37)$$

$$R_{i+2^{j-1},j+1}^t = R_{i+2^{j-1},j}^t + f(R_{i,j}^t, L_{i,j+1}^t), \qquad (38)$$

for all nodes at the jth level and the tth decoding iteration, starting from $j = 1$. The operation of Eq. (37) and Eq. (38) is encapsulated in Fig. 22c and Fig. 22d, respectively, which is analogous to the partial sum calculation of Fig. 16a. To elaborate, the partial sum calculation of Fig. 16a operates on the hard-decision values, while Eq. (37) and Eq. (38) carry out the same operation on the corresponding LLRs. It is also pertinent to mention here that the values of $R_{i,1}^t = R_{i,1}^0$ for all decoding iterations, since they are not updated as the iterations proceed.

- **Iterative message exchange:** One pass of the aforementioned 'right-to-left' and 'left-to-right' message exchange procedures constitute one round of BP decoding
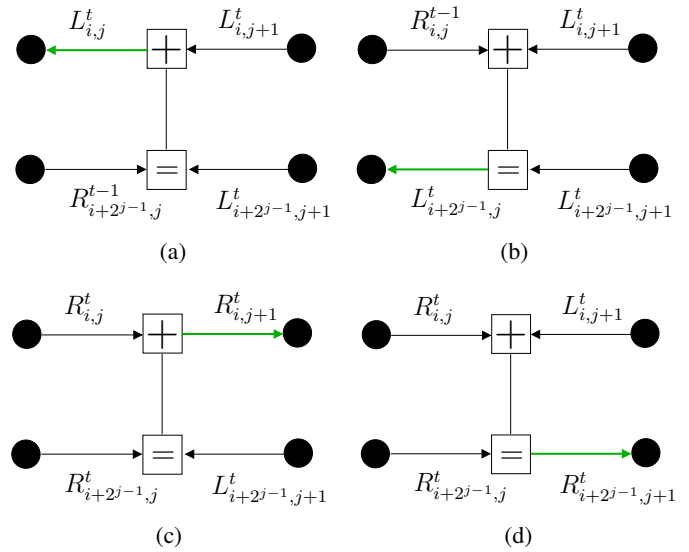

Fig. 22: Computation of LLRs at the jth level of the factor graph during the tth iteration of BP algorithm: (a) The function $f(L_{i,j+1}^t, L_{i+2^{j-1},j+1}^t + R_{i+2^{j-1},j}^{t-1})$ for calculating the left message $L_{i,j}^t$. (b) The function $L_{i+2^{j-1},j+1}^t + f(L_{i,j+1}^t, R_{i,j}^{t-1})$ for calculating the left message $L_{i+2^{j-1},j}^t$. (c) The function $f(R_{i,j}^t, L_{i+2^{j-1},j+1}^t + R_{i+2^{j-1},j}^t)$ for calculating the right message $R_{i,j+1}^t$. (d) The function $R_{i+2^{j-1},j}^t + f(R_{i,j}^t, L_{i,j+1}^t)$ for calculating the right message $R_{i+2^{j-1},j+1}^t$.

iteration. These procedures are repeatedly invoked during each decoding iteration until the maximum number of iterations is reached.

- **Hard-decision:** Finally, the information bits are estimated based on the hard-decision values of the LLRs $L_{i,1}^{I_{\max}}$ at the left-hand edge of the factor graph, as follows:

$$\hat{u}_i = \begin{cases} 0 & \text{if } L_{i,1}^{I_{\max}} > 0 \\ 1 & \text{otherwise}, \end{cases} \qquad (39)$$

for $i \in \mathcal{F}_c$.

The performance of the BP decoder is comparable to that of the classic SC decoder. However, it imposes a higher computational complexity due to the large number of iterations required for achieving the convergence and higher memory requirements. The time complexity of the BP algorithm is $O(I_{\max} N \log_2 N)$, while the space complexity is $O(N \log_2 N)$. Efforts were made in [90], [92] to reduce the computational complexity, while the hardware implementations of the BP decoder were presented in [85], [119]. Furthermore, improvements were proposed in [63], [89], [141] for the sake of enhancing the performance of the BP decoding algorithm. More specifically, a different decoding schedule for BP was investigated in [63], which first completes the right-to-left message exchange over the polar decoding circuit

Complexity	• Time complexity = $O(I_{max} N \log_2 N)$ • Space complexity = $O(N \log_2 N)$
Advantages	• Soft output, hence compatible with iterative detection • Fully-parallel implementation
Disadvantages	• Sub-optimal performance (better than SC) • High complexity • High memory requirements

TABLE VII: Main characteristics of a BP polar decoder.

Complexity	• Time complexity = $O(I_{max} N \log_2 N)$ • Space complexity = $O(N \log_2 N)$
Advantages	• Soft output, hence compatible with iterative detection • Much lower complexity than BP • Much lower memory requirements than BP
Disadvantages	• Sub-optimal performance (better than SC) • Serial processing, resulting in high latency • Fully-parallel implementation not feasible

TABLE VIII: Main characteristics of a SCAN polar decoder.

and then proceeds with the left-to-right message exchange. An alternative technique was also investigated in [63], called the overcomplete representation (also called permuted factor-graph), which permutes the different levels of the polar factor graph of Fig. 20 for achieving decoding convergence. In [89], the reliability of BP messages was improved by incorporating the knowledge of frozen bits, while a CRC was invoked in [141] together with the overcomplete representation of [63] for achieving decoding convergence. Despite these efforts, the BP algorithm did not outperform the SCL or SCS decoders. Recently, a BP list decoding algorithm was conceived in [126], which combines the benefits of the classic BP and SCL decoders. Explicitly, the BP list decoder benefits from the good error correction capabilities of the SCL decoder and the soft-in soft-out nature of the classic BP algorithm. Furthermore, it also lends itself to parallel implementation, hence there is a possibility of achieving a lower latency. However, the BP list decoder imposes a high computational complexity and its compatibility with CRC-aided polar codes needs to be investigated.

The main characteristics of a BP polar decoder are summarized in Table VII.

F. Soft Cancellation (SCAN)

The SC decoder of Section IV-A is a soft-in hard-out decoder, since hard-decisions are made pertaining to the bits $b_{i,j}$ during the decoding process. In the spirit of extracting soft output from the SC decoder, a Soft CANcellation (SCAN) decoder was proposed [83], [84], which is in essence a combination of the SC decoder of Section IV-A and the BP decoder of Section IV-E. Recall that the left-to-right message exchange procedure of BP is a soft-valued version of the f and g operations of an SC decoder. Similarly, the right-to-left message exchange of BP is a soft-valued counterpart of the partial sum calculation of SC. Hence, we may conclude that all operations of BP are soft-valued versions of the operations of SC, but they follow a different processing schedule. A SCAN decoder combines the attributes of SC and BP decoding algorithms by using the BP operations of Fig. 22 in combination with the SC processing schedule. This in turn facilitates faster convergence of the resulting decoding algorithm, hence drastically reducing the computational complexity. Quantitatively, it
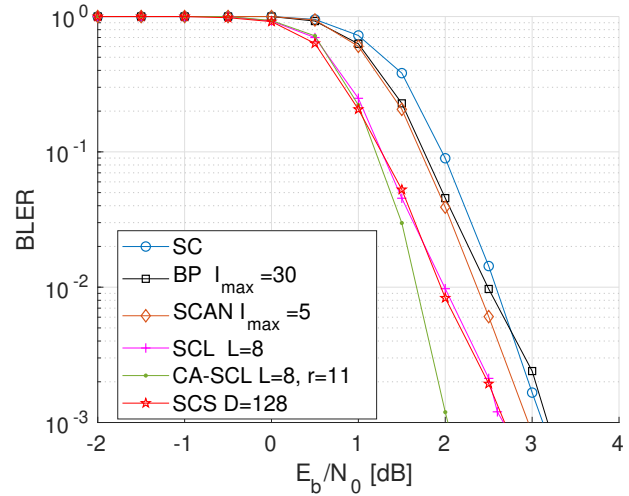
Fig. 23: Comparison of the achievable BLER performance of the polar decoders. A $(1024, 512)$ polar code relying on the 5G NR frozen bit sequence was used in conjunction with QPSK transmission over an AWGN channel. Min-sum approximation of Eq. (29) was invoked for calculating the box-plus operations in all decoders, except in the BP decoder, for which the exact computation of Eq. (28) was used.

was demonstrated in [84] that the computational complexity of the SCAN decoder is only 4% of the complexity of the classic BP decoder. Another notable contribution in the context of soft-in soft-out polar decoders is the soft counterpart of the SCL decoder [122], [123], which yields soft information for the iterative decoding of concatenated codes. Explicitly, similar to the classic SCL decoder, the soft SCL decoder maintains a list of L candidate decoding paths. Hence, it outperforms the the BP as well as the SCAN decoders. However, the soft SCL decoder is only applicable to systematic polar codes.

The main characteristics of a SCAN polar decoder are summarized in Table VIII.
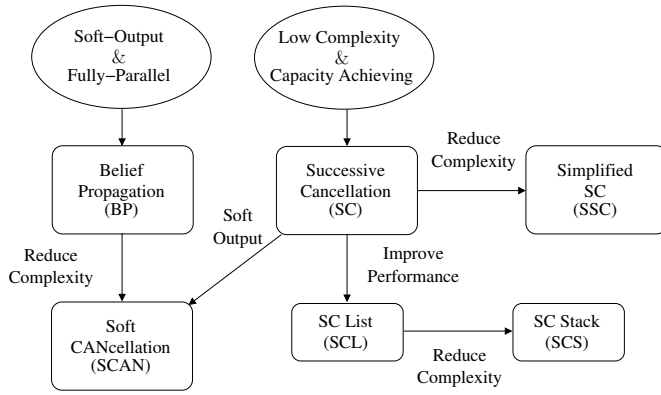
Fig. 24: Rationale for developing the polar decoders of Sections IV-A to IV-F.

Decoder	Complexity	Space Req.	Performance	Fully-parallel
SC	low	low	sub-optimal	No
SSC	very low	low	sub-optimal	No
SCL	medium	low	ML	No
CA-SCL	medium	low	outperform ML	No
SCS	low-medium	high	ML	No
BP	high	high	sub-optimal	Yes
SCAN	medium	medium	sub-optimal	No

TABLE IX: Comparison of the polar decoders discussed in Sections IV-A to IV-F.

G. Comparison of Polar Decoders

In Fig. 23, we have compared the BLER performance of the various polar decoders. We have used a $1/2$-rate polar code having a codeword length of 1024 and having the frozen bit sequence specified for the 5G NR [142]. For the sake of ensuring a fair comparison, the r CRC bits of the CA-SCL scheme have been taken from the parity bits of the constituent polar code, rather than from the information bits. More explicitly, the constituent polar code used in the CA-SCL scheme has a coding rate of $(k+r)/N = 523/1024$; hence, the coding rate of the resultant CRC-aided concatenated polar code is $k/N = 1/2$. Furthermore, we have invoked QPSK modulation and AWGN channel for transmission, in order to allow for comparison with the work that was completed during the development of the 5G NR standard. We may observe in Fig. 23 that the SC decoder exhibits the worst performance, while the CA-SCL decoder has the best performance and that of the BP, SCAN, SCL and SCS lies in between. Furthermore, the performance of SCS with a stack size of $D = 128$ is the same as that of SCL with a list-size of $L = 8$. As discussed in Sections IV-A to IV-F, the performance improvement comes at the cost of complexity. In Fig. 24, we have summarized the rationale for developing the various polar decoders, while their main characteristics are compared in Table IX.

V. POLAR CODE CONSTRUCTION: DESIGN PRINCIPLES, GUIDELINES & EXAMPLES

Recall from Fig. 9 and Fig. 10 that polar codes do not completely polarize at finite block lengths. Hence, a reliability metric is required to identify the least reliable bit-channels for transmitting frozen bits. This selection of frozen bit-channels is a very important step in the design of polar codes, because it directly dictates the resultant BLER. Explicitly, the BLER of a polar code is upper bounded by the sum of the BERs of the individual good bit-channels. So, if a bad channel is inadvertently not frozen, it will deteriorate the performance of the resultant polar code. Unfortunately, the set of frozen bit-channels is channel specific and is hence not universal. This makes the design of polar codes challenging, since it has to be optimized for the channel under consideration (or equivalently for a specific channel noise level). However, it must be acknowledged that the capacity changes with changing channel conditions. So, the required optimization is a natural consequence of any changes in the channel conditions.

Hence the design objective of polar code construction is as follows:

Design Objective: *For a given codeword length (N) and channel characteristics, for example SNR of an AWGN channel, determine the ($N-k$) least reliable bit-channels (or equivalently the k most reliable bit-channels) - the so-called bad channels.*

The polar code design process relies on the selection of a suitable metric for quantifying the reliability of the induced bit-channels and an accurate reliability estimation method, which we will discuss in Sections V-A and V-B, respectively. Once the reliability of each bit-channel is quantified, the ($N-k$) least reliable bit-channels may be selected as the frozen channels. Alternatively, a threshold may be defined for rate compatible codes, so that all bit-channels having a reliability below the threshold are frozen. The overall polar code design process is summarized in Fig. 25.

A. Reliability Metrics

The reliability of bit-channels can be quantified in terms of the mutual information $I(W_i)$, or more specifically the capacity, of the induced bit-channels W_i. A set of frozen bit-channels $\mathcal{F} \subset \{1, 2, \ldots, N\}$ can then be selected based on the mutual information for ensuring that we have:

$$I(W_i) \leq I(W_j) \quad \forall i \in \mathcal{F}, j \in \mathcal{F}_c. \qquad (40)$$

Alternatively, as proposed in Arikan's seminal paper [17], reliability can also be calculated using the Bhattacharyya parameter, since Bhattacharyya parameter gives an upper bound on the ML decision error and is hence a more accurate representative of the BLER. The Bhattacharyya parameter is defined as follows:

$$Z(W_i) = \sum_{y_1^N, u_1^{i-1}} \sqrt{P_i\left(y_1^N, u_1^{i-1}|u_i=0\right) P_i\left(y_1^N, u_1^{i-1}|u_i=1\right)}. \qquad (41)$$
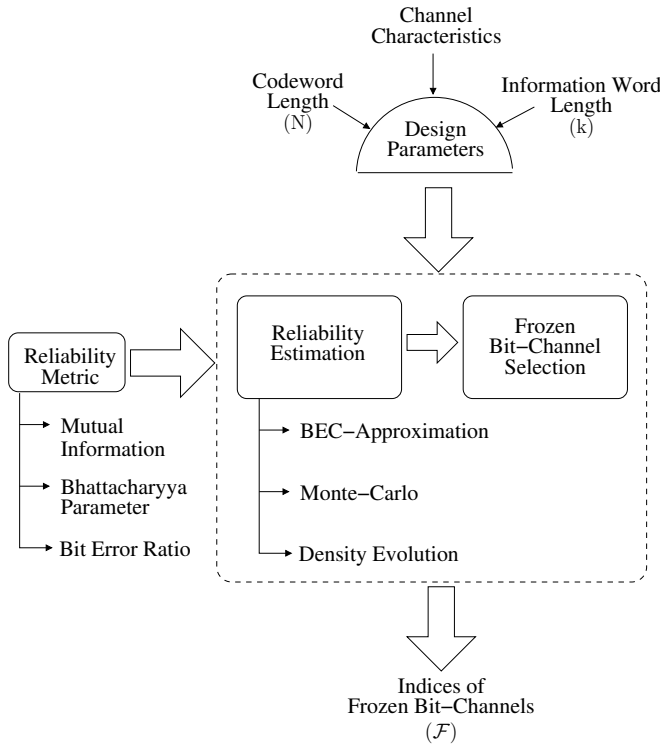
Fig. 25: Polar code design process.

A higher value of the Bhattacharyya parameter indicates a lower reliability, i.e. $Z(W_i) \to 1$, $I(W_i) \to 0$, and vice versa. Hence, a set of frozen bit-channels $\mathcal{F} \subset \{1, 2, \ldots, N\}$ can be selected based on the Bhattacharyya parameter for ensuring that we have:

$$Z(W_i) \geq Z(W_j) \quad \forall i \in \mathcal{F}, j \in \mathcal{F}_c. \tag{42}$$

Reliability can also be directly calculated in terms of the BER of the induced bit-channels. All these three metrics, i.e. the mutual information, the Bhattacharyya parameter and the BER, are interchangeably used in the literature as reliability metrics for the selection of frozen bit-channels.

B. Reliability Estimation Methods

The reliability metrics of Section V-A rely on the accurate computation of the transition probabilities $P_i(y_1^N, u_1^{i-1}|x_i)$ of the induced bit-channels. Explicitly, as illustrated in Fig. 12, the ith induced bit-channel maps the input $u_i \in \mathcal{X}$ onto the output $(y_1^N, u_1^{i-1}) \in \mathcal{Y}^N \times \mathcal{X}^{i-1}$, where $\mathcal{X}$ and $\mathcal{Y}$ denote the input and output alphabets of the channel W. Hence, the cardinality of the output alphabet of the induced channel is $|\mathcal{Y}^N| \times |\mathcal{X}^{i-1}|$, which grows exponentially with the codeword length N. This in turn implies that the complexity of computing the exact channel transition probabilities grows exponentially with the codeword length N. This is also evident from Eq. (41), since the summation in Eq. (41) is carried out over all possible values of y_1^N as well as u_1^{i-1}. Therefore, frozen bit-channel

selection is deemed intractable. However, due to the recursive nature of polar codes, the Bhattacharyya parameter as well as the mutual information may be efficiently computed for BECs. The recursive mutual information calculations were shown in Eq. (16), while the Bhattacharyya parameter may be recursively calculated as follows:

$$Z\left(W_{2i-1}^N\right) = 2Z\left(W_i^{N/2}\right) - Z\left(W_i^{N/2}\right)^2$$
$$Z\left(W_{2i}^N\right) = Z\left(W_i^{N/2}\right)^2, \tag{43}$$

with $Z(W_1) = \epsilon$, for the worse and better channels, respectively, where ϵ is the erasure probability of BEC. This algorithm incurs a complexity cost of $O(N \log_2 N)$.

Since it is hard to track the exact mutual information, the Bhattacharyya parameter or the BER for channels other than BECs, various approximation methods have been proposed over the years. In [17], Monte-Carlo simulations were invoked for estimating the bit-channel reliabilities. More specifically, recall from Section IV-A that the SC decoder yields the channel transitions probabilities $P_i(y_1^N, u_1^{i-1}|x_i)$. Consequently, the Monte-Carlo based method operates by generating random information and noise sequences for the given channel characteristics, and then estimating the Bhattacharyya parameter or the mutual information using the output probabilities $P_i(y_1^N, u_1^{i-1}|x_i)$ of the SC decoder, assuming that the decoder knows the bits u_1^{i-1}. Equivalently, the BER of the induced channels can be estimated based on the output of the SC decoder. This process incurs a complexity of $O(MN \log_2 N)$ for M rounds of Monte-Carlo simulations. However, the complexity of the individual SC operations can be reduced for symmetric channels by assuming an all-zero input. Explicitly, since the SC decoder is assumed to know correctly the bits u_1^{i-1} for the purpose of channel reliability estimation, the g operation of Eq. (30) reduces to:

$$L_{i+2^{j-1},j} = L_{i,j+1} + L_{i+2^{j-1},j+1}. \tag{44}$$

when the input is an all-zero sequence. This in turn implies that we do not need to carry out the partial sum operations of Eq. (31). Hence, the f and g operations of the SC decoder can be computed in parallel, which significantly speeds up the process. Nonetheless, this approach becomes computationally intensive for long codeword, in particular at high SNRs, because M must be large enough to get reliable estimates.

Polar codes for arbitrary binary-input channels can also be heuristically designed by considering a BEC of equivalent capacity [59] or Bhattacharyya parameter [143], while Density Evolution (DE) was invoked in [144], [145] to calculate the bit-channel reliabilities with a linear complexity of $O(N)$. The polar codes of [144], [145], which were customized for BSC and AWGN channel, outperformed the designs of [59]. However, the DE-based construction method of [144], [145] invokes convolution operations, whose exact implementation imposes exponentially increasing memory requirements. Alternatively, the memory requirements can be reduced by approximating the convolutional operations using quantization (also called binning). But this in turn leads to quantization

errors. Hence, a suitable compromise has to be reached between the implementation complexity and the accuracy. Tal and Vardy addressed this issue in [146] by introducing two channel approximations, called the degraded and the upgraded quantizations, which provide a lower and upper bound, respectively, on the error probability of the underlying channel. Both these approximation methods reduce the cardinality of the channel output based on the parameter μ, so that the channel outputs become tractable. The complexity of the resulting method is $O(N\mu^2 \log_2 \mu)$, where the typical value of μ is 256. These ideas were further investigated in [147] by exploiting alternate methods for approximating a degraded channel, and generalized to non-binary channels in [148].

Inspired by the low-complexity Gaussian Approximation (GA)-based DE of LDPC codes, Trifonov [76] used GA-DE for designing polar codes for AWGN channels. GA-DE tracks the mean value of the LLRs over the polar decoding circuit, assuming that the LLRs at all the nodes conform to a Gaussian distribution. More explicitly, given that an all-zero codeword is transmitted over an AWGN channel, the channel LLRs $L(y)$ exhibit a Gaussian distribution with a mean of $2/\sigma^2$ and a variance of $4/\sigma^2$, where σ^2 denotes the noise variance per dimension. The mean of the LLRs of the polar decoding circuit can then be approximated as follows:

$$\mathrm{E}\left[L_{2i-1}^{N}\right] = \phi^{-1}\left(1 - \left(1 - \phi\left(\mathrm{E}\left[L_i^{N/2}\right]\right)\right)^2\right)$$
$$\mathrm{E}\left[L_{2i}^{N}\right] = 2\,\mathrm{E}\left[L_i^{N/2}\right], \tag{45}$$

for the worse and better channels, respectively. In Eq. (45), E denotes the expectation (or equivalently mean) operation and the function ϕ is defined as follows:

$$\phi(x) = \begin{cases} 1 - \frac{1}{4\pi x}\int_{-\infty}^{\infty} \tanh\frac{u}{2}\, e^{-\frac{(u-x)^2}{4x}}\, du & \text{if } x > 0 \\ 1 & \text{if } x = 0, \end{cases} \tag{46}$$

which may be approximated as [149], [150]:

$$\phi(x) \approx \begin{cases} e^{-\alpha x^2 + bx} & \text{for } 0 \le x < c \\ e^{-\alpha x^{\gamma} + \beta} & \text{for } c \le x < 10 \\ \sqrt{\frac{\pi}{x}}e^{-\frac{x}{4}}\left(1 - \frac{10}{7x}\right) & \text{for } 10 \le x, \end{cases} \tag{47}$$

where $a = -0.0564$, $b = 0.48560$, $c = 0.867861$, $\alpha = -0.4527$, $\beta = 0.0218$ and $\gamma = 0.86$. Furthermore, the inverse function ϕ^{-1} of Eq. (45) can be estimated using numerical analysis techniques, for example the bisection method or the Newton-Raphson method. The mean LLRs of Eq. (45) may then be used for approximating the BER of the corresponding bit-channels using:

$$\mathrm{BER}_i \approx Q\left(\sqrt{\mathrm{E}\left[L_i^N\right]}\right). \tag{48}$$

Alternatively, the mean LLRs may be used for calculating the mutual information or the Bhattacharyya parameter.

The aforementioned polar code construction methods rely on the SC decoder. Vangala *et al.* [143] compared these construction methods and demonstrated that all perform equally

Design Parameters	Codeword Length (N)
	Information word Length (k)
	Channel characteristics
Reliability Metric	Mutual Information
	Bhattacharyya Parameter
	Bit Error Rate
Estimation Method	Monte-Carlo [17]
	BEC-approximation [59], [143]
	Density Evolution [144], [145]
	Degraded/Upgraded Quantization [146]
	Gaussian Approximation Density Evolution [76]
	Decoder-specific methods [151]–[155]

TABLE X: Frozen bit-channel selection procedure for polar codes.

well provided that the design SNR is carefully chosen. Qin *et al.* [151] proposed an improved reliability estimation method tailored for the BP decoder, which tracks the evolution LLRs during the BP decoding process. However, only a marginal improvement of upto 0.5 dB was reported. In [152], the polar code was heuristically optimized for the SCL decoder. Inspired by these decoder-specific polar code designs, reinforcement learning techniques and genetic algorithms were invoked in [153] and [154], [155] to customize the code design for the given polar decoder.

We have summarized the discussions of this section in Table X.

C. Design Examples

In this section, we will compare the different reliability estimation methods of Fig. 25 by designing a $1/2$-rate polar code having a codeword length of $N = 1024$ for an AWGN channel. In particular, we compare the BEC-approximation of [59], [143], the GA-DE of [76] and the Monte Carlo based method [17]. Furthermore, we chose the Bhattacharyya parameter for quantifying reliability using the BEC-approximation method, while the BER was used for the other two methods. Fig. 26a records the resulting intensity maps for the three methods at $E_b/N_0 = 0$ dB. Explicitly, a value of 0 in the intensity map of Fig. 26a corresponds to the maximum reliability, while a value of 1 denotes the lowest reliability. We may notice that there are only slight variations in the three intensity maps, with the GA-DE and Monte Carlo methods being very similar. We next classify the 512 least reliable bit-channels as frozen bit-channels and plot the frozen bit-channel patterns in Fig. 26b, where the frozen bit-channels are colored in black, while the information bit-channels are in white. Again, the frozen bit-channel patterns are similar for the three methods.

In Fig. 27, we compare the achievable BLER performance of the $(1024, 512)$ polar codes constructed using the BEC-approximation based Bhattacharyya parameter, GA-DE based BER and the Monte Carlo based BER at different design SNRs per bit. In line with Fig. 26, the performance of the polar codes constructed using the Monte Carlo and the GA-DE methods is similar, with the latter being slightly better, while that of the polar codes designed using the BEC-approximation
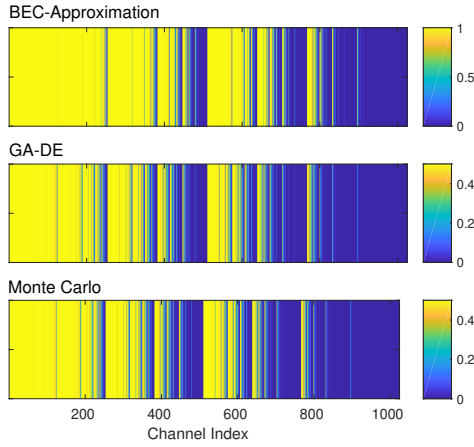
(a) Reliability intensity. Please note that a value of 0 implies high reliability, since the reliability metric is the Bhattacharyya parameter for BEC-approximation and the BER for GA-DE and Monte Carlo.
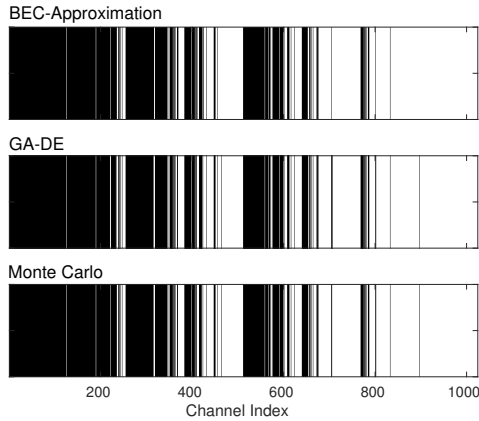
(b) Frozen bit-channel patterns. The black regions denote the frozen indices, while the white one is for information indices.

Fig. 26: Comparison of polar code construction methods at $\frac{E_b}{N_0} = 0$ dB: BEC-approximation based Bhattacharyya parameter, GA-DE based BER and Monte Carlo based BER.

method is worse. Furthermore, Fig. 27 shows that the BLER performance of the Monte Carlo and the GA-DE methods is only sightly affected, when the design SNR per bit is increased from 0 dB to 2 dB. It is obvious that if we choose a very low or high value of the design SNR per bit, then the performance will get worse, as demonstrated in [143]. Hence, the polar code construction method is not very sensitive to reasonable discrepancies between the design SNR and the operating SNR. We have also benchmarked the designed polar codes against the 3GPP 5G NR polar code in Fig. 27. It may be observed that the performance of the Monte Carlo and the GA-DE based polar codes approaches that of the 3GPP 5G NR when the design $E_b/N_0 = 2$ dB.
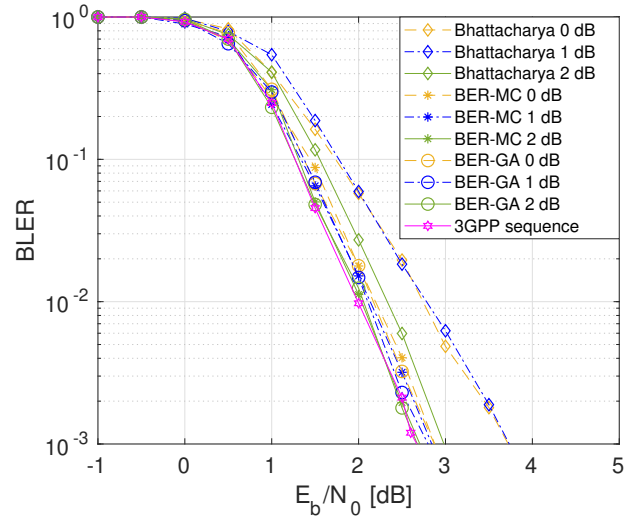
Fig. 27: Comparison of the achievable BLER performance of the $(1024, 512)$ polar codes constructed using the various reliability estimation methods of Section V-B at different design SNRs per bit. The performance is benchmarked against the 3GPP 5G NR frozen bit-channel sequence. An SCL decoder having $L = 8$ and QPSK transmission over an AWGN channel was used.

VI. QUANTUM-TO-CLASSICAL ISOMORPHISM

In contrast to a classical bit, which can be either 0 or 1 at any particular instant, a quantum bit (qubit)[6] exists in superposition of the orthogonal basis states $|0\rangle$ and $|1\rangle$. This superimposed state of the qubit is generally described using the state vector as follows:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \qquad (49)$$

where $|\cdot\rangle$ is called Ket or Dirac notation [157], and α and β are complex coefficients conforming to:

$$|\alpha|^2 + |\beta|^2 = 1. \qquad (50)$$

Furthermore, unlike a classical bit, which can be 'observed' (or 'measured') without disturbing its value, any observation of the qubit perturbs its superimposed state of Eq. (49). To elaborate, if a qubit is observed in the computational basis[7], it may collapse to the state $|0\rangle$ with a probability of $|\alpha|^2$ and the state $|1\rangle$ with a probability of $|\beta|^2$. The quantum superposition of Eq. (49) makes quantum processing (or computation) systems inherently parallel, while the observation property together with the quantum no-cloning theorem[8] makes quantum transmissions absolutely secure. However, these unusual quantum characteristics, which have no counterpart in the classical

---

[6]Please refer to [156] for deeper insights into the duality of the classical and the quantum regime.

[7]The pair of orthogonal basis states $|0\rangle$ and $|1\rangle$ is called the computational basis.

[8]According to the quantum no-cloning theorem, arbitrary quantum states cannot be cloned (or copied) [158].

domain, make it challenging to design Quantum Error Correction Codes (QECCs). Nonetheless, there exists an underlying isomorphism between the classical and quantum paradigms, which can be exploited for designing efficient QECCs from the known classical codes [156], [159], [160].

Environmental decoherence is a major source of noise in quantum systems. It can be modeled using a depolarizing channel, which is considered the 'worst-case scenario' [161]. Explicitly, a quantum depolarizing channel characterized by the depolarizing probability p independently inflicts an error on each qubit, such that a qubit may experience a bit-flip (Pauli-$\mathbf{X}$), a phase-flip (Pauli-$\mathbf{Z}$) or a bit-and-phase-flip (Pauli-$\mathbf{Y}$) error with a probability of $p/3$ each[9]. Hence, in contrast to the classical channels, which only inflict bit-flip errors, the quantum depolarizing channel imposes both bit-flip as well as phase-flip errors. An interesting point to notice here is that a quantum depolarizing channel may also be viewed as a pair of correlated BSCs inflicting bit-flips and phase-flips respectively. Fortunately, classical polar codes are capable of concurrently polarizing both the bit-flip as well as the phase-flip channels, when the classical XOR gates are replaced by the quantum Controlled NOT (CNOT) gates[10]; hence achieving the quantum channel capacity, as demonstrated in [162], [163]. This is because polar codes merely rely on CNOT gates for channel polarization, which are capable of concurrently polarizing both the bit-flip as well as the phase-flip channels, but in opposite directions, as further discussed below.

QECCs exploit the computational basis for correcting bit-flip errors, while the Hadamard basis[11] is used for phase-flip correction. The action of CNOT on the Hadamard basis is analogous to that on the computational basis with the role of control and target qubits swapped. Explicitly, let us consider the two-qubit state $|+-\rangle$ in Hadamard basis, which is equivalent to:

$$|+-\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right)$$
$$= \frac{1}{2}\left(|00\rangle - |01\rangle + |10\rangle - |11\rangle\right), \quad (51)$$

in the computational basis. When a CNOT gate is applied to

---

[9]The $\mathbf{I}$, $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ are single-qubit Pauli operators (or gates) defined as:

$$\mathbf{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \ \mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \ \mathbf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \ \mathbf{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}.$$

[10]The CNOT gate is a two-qubit gate, having a control qubit and a target qubit. When the control qubit is in state $|1\rangle$, the target qubit is flipped; otherwise, the target qubit is left unchanged, as encapsulated below:

$$\text{CNOT}\left(|\psi_0\rangle, |\psi_1\rangle\right) = |\psi_0\rangle \otimes |\psi_0 \oplus \psi_1\rangle,$$

where $|\psi_0\rangle$ is the control qubit, while $|\psi_1\rangle$ is the target qubit. Hence, CNOT gate is the quantum analogue of the classical XOR gate.

[11]The pair of orthogonal basis states $|+\rangle$ and $|-\rangle$ is called the Hadamard basis, where the basis states $|+\rangle$ and $|-\rangle$ are defined as:

$$|+\rangle \triangleq \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right), \ |-\rangle \triangleq \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right).$$
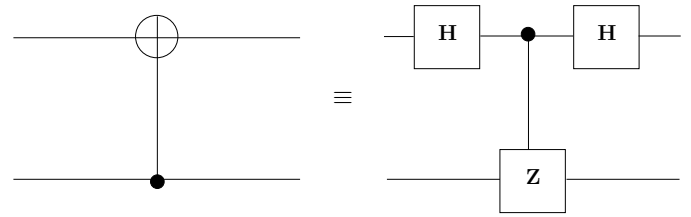

Fig. 28: A CNOT gate is equivalent to a Controlled-$\mathbf{Z}$ (Cz) gate, with the control and target qubits swapped, when Hadamard gates are invoked at the input and output. The circuit to the left flips the top qubit in computational basis ($|0\rangle$ and $|1\rangle$), when the bottom qubit is in the state $|1\rangle$, while the circuit to the right flips the botton qubit in the Hadamard basis ($|+\rangle$ and $|-\rangle$), when the top qubit is in the state $|-\rangle$.
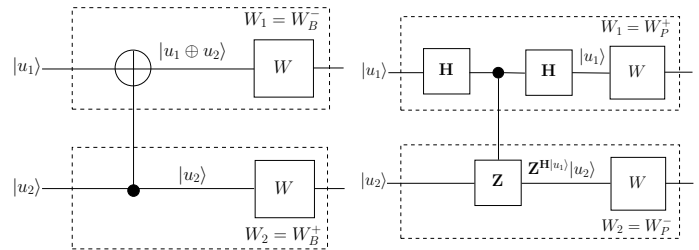
(a) Polarization in the computational basis, resulting in polarized bit-flip channels.

(b) Polarization in the Hadamard basis, resulting in polarized phase-flip channels.

Fig. 29: The 2-qubit kernel of a quantum polar code in the computational and Hadamard bases.

the second qubit controlled by the first qubit, we get:

$$|+-\rangle \xrightarrow{\text{CNOT}(1,2)} \frac{1}{2}\left(|00\rangle - |01\rangle + |11\rangle - |10\rangle\right) = |--\rangle. \quad (52)$$

We may observe in Eq. (52) that the operation of $\text{CNOT}(1,2)$ on the computational basis is equivalent to that of $\overline{\text{CNOT}}(2,1)$ on the Hadamard basis. More explicitly, the classic $\text{CNOT}(i,j)$ acting on the computational basis flips the jth qubit (target) between $|0\rangle$ and $|1\rangle$, when the ith qubit (control) is in the state $|1\rangle$. This operation is analogous to that of $\overline{\text{CNOT}}(j,i)$ acting on the Hadamard basis, which flips the ith qubit (target) between $|+\rangle$ and $|-\rangle$, when the jth qubit (control) is in the state $|-\rangle$. More specifically, a CNOT gate may also be implemented using a Controlled-$\mathbf{Z}$ (Cz) gate, with the control and target qubits swapped, as shown in Fig. 28.

Let us now consider the 2-qubit kernel of polar code given in Fig. 29a, which is the quantum-domain equivalent of Arikan's kernel of Fig. 5a. From the perspective of the computational basis, the encoder of Fig. 29a combines the information of the two channels in the computational basis, so that the computational basis information of the $|u_2\rangle$ is redundantly added to that of $|u_1\rangle$ ($\oplus$ denotes modulo-2 addition in the

computational basis). Consequently, the second qubit-channel W_2 becomes more robust against bit-flip errors at the cost of deteriorating the robustness of the first qubit-channel W_1. In other words, W_2 tends to polarize towards a better bit-flip channel W_B^+, while W_1 tends to polarize towards a worse bit-flip channel W_B^-. However, the total capacity of the two channels is conserved.

Fig. 29b shows the polar encoder from the perspective of the Hadamard basis; hence, the CNOT gate is replaced by the equivalent circuit of Fig. 28, which invokes the Cz gate. More explicitly, in Fig. 29b, the Hadamard basis information of the first qubit $|u_1\rangle$ is redundantly incorporated in the Hadamard basis information of the second qubit. Consequently, the information carrying capacity of the first qubit-channel W_1 is enhanced in the phase basis, while that of the second qubit-channel W_2 degrades in the phase basis. Hence, the first qubit-channel W_1 tends to polarize towards a better phase-flip channel, while the second qubit-channel W_2 tends to polarize towards a worse phase-flip channel, denoted by W_P^+ and W_P^-, respectively. Therefore, the elementary kernal of a polar code is capable of polarizing both the bit-flip as well as phase-flip channels, but the direction of polarization is opposite, as illustrated in Fig. 29.

Based on the above discussions, a quantum polar code induces four sets of channels, covering both the bit-flip as well as the phase-flip errors. Explicitly, the polarized channels may belong to one of the following four sets:

1) **Good bit-and-phase channels** ($\mathcal{F}^c$): These induced channels exhibit high information carrying capacity in the computational basis (bit-flip) as well as in the Hadamard basis (phase-flip). Consequently, these induced channels are used for transmitting the uncoded qubits.

2) **Good bit-only channels** ($\mathcal{F}_P$): These channels have high information carrying capacity in the computational basis (bit-flip), but low capacity in the Hadamard basis (phase-flip). Hence, these channels are frozen in the Hadamard basis by transmitting the Hadamard basis states $|+\rangle$ or $|-\rangle$, which are known to the receiver.

3) **Good phase-only channels** ($\mathcal{F}_B$): These channels have low information carrying capacity in the computational basis (bit-flip), but high capacity in the Hadamard basis (phase-flip). Consequently, they are frozen in the computational basis, hence transmitting the computational basis states $|0\rangle$ or $|1\rangle$, which are known to the receiver.

4) **Bad bit-and-phase channels** ($\mathcal{F}_{BP}$): These channels have low information carrying capacity in both the computational basis (bit-flip) as well as the Hadamard basis (phase-flip). Consequently, these channels are frozen in both the computational as well as the Hadamard bases.

This is achieved by exploiting pre-shared entangled[12] qubits, which are referred to as ebits. Explicitly, ebits are created in the Bell state $|\phi^+\rangle$, expressed as:

$$|\phi^+\rangle = \frac{|00\rangle^{T_X R_X} + |11\rangle^{T_X R_X}}{\sqrt{2}}, \qquad (53)$$

so that the first qubit is retained at the transmitter (T_X), while the associated entangled qubit is sent to the receiver (R_X) before actual transmission commences, for example during off-peak hours, when the channels are under-utilized. It is generally assumed that the pre-sharing of ebits takes place over a noiseless quantum channel. Hence, quantum polar codes intrinsically belong to the family of entanglement-assisted QECCs [165], [166].

VII. QUANTUM POLAR CODES

Inspired by the provably capacity achieving nature of classical polar codes as well as their efficient encoding and decoding structures, Wilde and Guha [167] were the first to demonstrate the existence of the channel polarization phenomenon for classical-quantum channels, which transmit classical information over quantum channels. These ideas were later extended to the transmission of quantum information in [162]. The quantum polar codes of [162], [167] exploit the same encoder as Arikan's polar codes, except that the classical XOR gates are replaced by the quantum CNOT gates. Consequently, the quantum polar encoders of [162], [167] inherently benefit from the low encoding complexity of Arikan's classic polar codes. Furthermore, a quantum counterpart of the classical SC decoder, named Quantum Successive Cancellation Decoder (QSCD), was conceived in [162], [167], which makes collective measurement on all channel uses. This is achieved by exploiting quantum hypothesis testing [168], [169] in conjunction with Sen's noncommutative union bound [170]. The QSCD of [162], [167] failed to match the low decoding complexity of the classical SC decoder. This issue was addressed by Renes *et al.* in [171], where an efficient implementation of a quantum polar decoder was given for quantum Pauli[13] and erasure channels. Finally, Wilde and Renes combined their efforts in [163], [172] to present an efficient QSCD for arbitrary quantum channels. The quantum polar codes of [162], [167], [171] rely on the sharing of noiseless ebits between the transmitter and the receiver. In this context, the first unassisted quantum polar codes, consisting of concatenated bit-flip (computational basis) and phase-flip (Hadamard basis) correction

---

[12]'Entanglement', which Einstein termed as a 'spooky action at a distance' [164], is the mysterious, correlation-like property between two or more qubits, which implies that the entangled qubits cannot be expressed as the tensor product of the individual qubits. Furthermore, a strange relationship exists between the two entangled qubits, which entails that measuring one of them also reveals the value of the other, even if they are geographically separated.

[13]A quantum Pauli channel independently inflicts an error on each qubit, such that a qubit may experience a bit-flip (Pauli-**X**), a phase-flip (Pauli-**Z**) and a bit-and-phase-flip (Pauli-**Y**) error with a probability of p_x, p_z and p_y, respectively. A Quantum depolarizing channel is a special case of a Pauli channel having $p_x = p_z = p_y = p/3$.

polar transformations, were recently conceived in [173]. The authors of [173] also presented efficient encoding and decoding implementations for Pauli and erasure channels. In the midst of these advancements, Ferris and Poulin [174] developed a new family of QECCs based on tensor networks, called the branching Multi-scale Entanglement Renormalization Ansatz (MERA) codes, which is a generalization of quantum polar codes. In contrast to the quantum polar codes of [162], [163], [167], [171], [172], the tensor network based quantum polar codes of [174] invoke syndrome-based classical decoding for estimating channel errors encountered over the quantum Pauli or erasure channels.

A. Quantum Polar Encoders

Recall from Section VI that a quantum depolarizing channel is equivalent to two correlated classical BSCs. The correlation between these two channels can be ignored in the spirit of simplifying the design process, while compromising on the achievable performance. Hence, a quantum depolarizing channel having a depolarizing probability of p may be modeled using two independent BSCs having a cross-over probability of $2p/3$. This model is widely used for constructing Calderbank-Shor-Steane (CSS)-type quantum codes [175]–[177], which are designed to independently correct bit-flip and phase-flip errors. Consequently, qubits frozen in the computational and in the Hadamard basis can be independently determined by finding qubit locations which yield the highest mutual information for the bit-flip and phase-flip channels, respectively. Furthermore, since the polarization of a quantum bit-flip channel is identical to that of a classical channel, as illustrated in Fig. 29a, qubits frozen in the computational basis can be selected by running the classical frozen bit-channel selection procedure for a BSC having a cross-over probability of $2p/3$ (given that the design depolarizing channel probability is p). By contrast, we observed in Fig. 29, that the polarization of a quantum phase-flip channel is the same as the classical channel polarization (or equivalently quantum bit-flip channel polarization), but the direction of polarization is opposite. This in turn implies that the pattern of quantum phase-flip polarization can be obtained by swapping the control and target qubits of the 2-bit kernel of a classical polar code. The resulting 2-bit kernel for phase-flip channel is given by:

$$\tilde{G}_2 = G_2^T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \tag{54}$$

and the equivalent N-bit encoder $\tilde{G}_N$ is:

$$\tilde{G}_N = (G_2^T)^{\otimes n} = G_N^T. \tag{55}$$

Hence, the location of qubits frozen in the computational basis may be determined by invoking the encoder G_N, while the same in the Hadamard basis may be selected by using the encoder G_N^T.

Fig. 30 shows the encoders G and G^T for selecting frozen channel indices in the computational and in the Hadamard basis, respectively, when the codeword length is 4. We may observe that the ith bit of the encoder G is equivalent to the
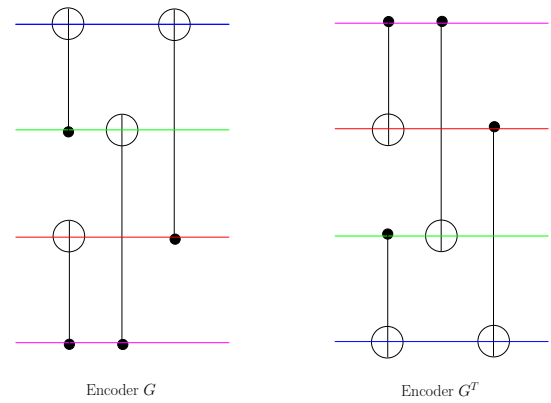

Encoder G Encoder G^T

Fig. 30: Encoders G and G^T used for selecting frozen channel indices for quantum bit-flip and phase-flip channels, respectively.

$(4-i+1)$th bit of the encoder G^T. This in turn implies that the ith bit of the encoder G is equivalent to the $(N-i+1)$th bit of the encoder G^T, where $1 \leq i \leq N$. Therefore, if we freeze the ith qubit in the computational basis for the bit-flip channel, then we also freeze the $(N-i+1)$th qubit in the Hadamard basis for the phase-flip channel.

Fig. 31 shows the mutual information intensity maps for the bit-flip and the phase-flip channels at a depolarizing probability of $p = 0.06$, when a polar code of length 64 is used. Explicitly, the colormap in Fig. 31 represents the mutual information, and the qubit indices are sorted based on the mutual information of the bit-flip channel. As discussed in the context of Fig. 30, the mutual information intensity map for the phase-flip channel is the same as that of the bit-flip channel, but it is flipped from right to left.

B. Quantum Polar Decoders

Fig. 32 shows the general schematic of a quantum communication system relying on a quantum polar code for protection against environmental decoherence. Unlike the QSCD based polar codes of [162], [163], [167], [171], [172], the system of Fig. 32 invokes the syndrome-based classical decoding approach of [174]. It is pertinent to mention here that quantum polar codes with syndrome-based classical decoding may not be capacity achieving for arbitrary quantum channels, which require collective measurement over all channel uses. Nonetheless, we have adopted this approach because it is a more direct application of the classical polar codes. Recall from Section VI that a polar code polarizes the input quantum channels into four sets, which may be denoted as:

- $\mathcal{F}^c$: good bit-and-phase channels for transmitting arbitrary quantum information $|\psi\rangle$;
- $\mathcal{F}_P$: good bit-only channels for transmitting known Hadamard basis states $|z\rangle$, where $z \in \{+, -\}$;
- $\mathcal{F}_B$: good phase-only channels for transmitting known computational basis states $|x\rangle$, where $x \in \{0, 1\}$;
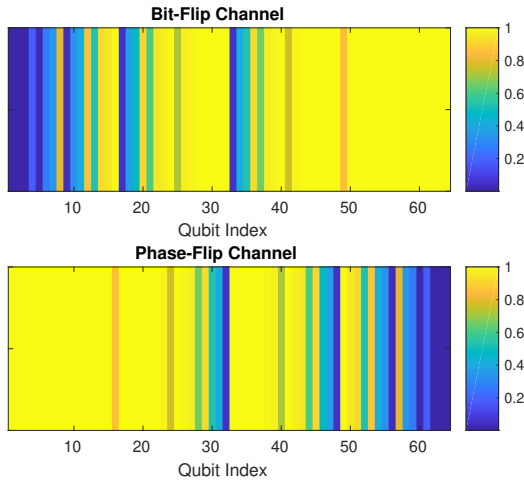
Fig. 31: Mutual information intensity map for bit-flip and phase-flip channels at a depolarizing probability of $p = 0.06$, when a polar code of length 64 is used.

- $\mathcal{F}_{BP}$: bad bit-and-phase channels for transmitting ebits.

Consequently, an $[N, k, c]^{14}$ polar code specified by the encoder $\mathcal{V}$ and the codespace $\mathcal{C}$ takes as input a k-qubit information word (logical qubits) $|\psi\rangle$ and maps it onto an n-qubit codeword (physical qubits) with the aid of $(N - k - c)$ frozen qubits initialized to the known computational and Hadamard basis states and c ebits, whose one qubit is pre-shared with the receiver. This may be mathematically expressed as:

$$\mathcal{C} = \{|\overline{\psi}\rangle = \mathcal{V}\left(|\psi\rangle_{\mathcal{F}^c} \otimes |x\rangle_{\mathcal{F}_B} \otimes |z\rangle_{\mathcal{F}_P} \otimes |\phi\rangle^+_{\mathcal{F}_{BP}}\right)\}. \quad (56)$$

The resulting encoded qubits $|\overline{\psi}\rangle$ are sent over a quantum depolarizing channel, which may inflict bit-flip, phase-flip or bit-and-phase-flip errors, each with a probability of $p/3$. The received quantum information may be represented as:

$$|\widetilde{\overline{\psi}}\rangle = \mathcal{P}|\overline{\psi}\rangle, \quad (57)$$

where $\mathcal{P}$ denotes the n-qubit Pauli error inflicted by the quantum depolarizing channel. At the receiver, the received information of Eq. (57) is passed through the inverse encoder $\mathcal{V}^\dagger$, which yields:

$$\begin{aligned} \mathcal{V}^\dagger |\widetilde{\overline{\psi}}\rangle = \mathcal{V}^\dagger \mathcal{P}|\overline{\psi}\rangle = \mathcal{V}^\dagger \mathcal{P}\mathcal{V}\left(|\psi\rangle_{\mathcal{F}^c} \otimes |x\rangle_{\mathcal{F}_B} \otimes |z\rangle_{\mathcal{F}_P} \otimes |\phi\rangle^+_{\mathcal{F}_{BP}}\right) \\ = \mathcal{L}|\psi\rangle_{\mathcal{F}^c} \otimes \mathcal{S}_x |x\rangle_{\mathcal{F}_B} \otimes \mathcal{S}_z |z\rangle_{\mathcal{F}_P} \otimes \mathcal{S}_y |\phi\rangle^+_{\mathcal{F}_{BP}} \\ = |\tilde{\psi}\rangle_{\mathcal{F}^c} \otimes |\tilde{x}\rangle_{\mathcal{F}_B} \otimes |\tilde{z}\rangle_{\mathcal{F}_P} \otimes |\tilde{\phi}\rangle^+_{\mathcal{F}_{BP}}. \quad (58) \end{aligned}$$

where $\mathcal{V}^\dagger \mathcal{P} \mathcal{V} \equiv (\mathcal{L} \otimes \mathcal{S}_x \otimes \mathcal{S}_z \otimes \mathcal{S}_y)$ and $\mathcal{L}$ denotes the k-qubit error inflicted on the information word $|\psi\rangle$, while $\mathcal{S}_x$, $\mathcal{S}_z$ and $\mathcal{S}_y$ represent the errors imposed on the frozen qubits, i.e. the computational, the Hadamard and the entangled qubits, respectively. As mentioned earlier, we assume that only

---

[14]We consistently use round brackets (.) for classical codes, while the square brackets [.] are used for quantum codes.

the first ebit of the Bell states experiences noise, while the second ebit is pre-shared over a noiseless channel. Finally, the corrupted computational and Hadamard basis states, i.e. $|\hat{x}\rangle$ and $|\hat{z}\rangle$, are measured in the computational and in the Hadamard basis, respectively, for the sake of determining the errors inflicted on these qubits. Explicitly, the former reveals information about any bit-flips imposed on the frozen states $|x\rangle$, while the latter provides information about phase-flips inflicted on the frozen states $|z\rangle$. Let us denote the outcomes as s_x and s_z, respectively, which are classical bits. Next, we have to determine the error imposed on the ebits transmitted over the quantum channel. Recall that ebits are frozen in both the computational and the Hadamard bases, since the corresponding channels are bad from the perspective of bit-flips as well as phase-flips. Consequently, we have to find the bit-and-phase-flip errors imposed on the ebits. This may be achieved by using the Pauli operators $g_x = \mathbf{XX}$ and $g_z = \mathbf{ZZ}$, where the first Pauli operator acts on the ebit transmitted over the quantum channel, while the second Pauli operator acts on the pre-shared noiseless ebit. Since ebits were created in the Bell states of Eq. (53), both Pauli operators g_x and g_z constitute the stabilizers. Furthermore, a bit-flip error on the first ebit will yield an eigenvalue of -1 for the stabilizer g_z, while a phase-flip error will yield an eigenvalue of -1 for g_x. Hence, both bit-flip as well as phase-flip errors acting on the first ebit can be determined, which may be denoted as s_y. The error patterns s_x, s_z and s_y acting on the frozen qubits are then fed to a syndrome-based polar decoder for the sake of estimating the logical error $\hat{\mathcal{L}}$ experienced by the information word $|\psi\rangle$. Finally, a recovery operation $\mathcal{R}$ is applied to $|\hat{\psi}\rangle$ based on the estimated error pattern $\hat{\mathcal{L}}$, hence recovering the transmitted information.

Let us now elaborate on the syndrome decoding block of Fig. 32, which takes as input the errors experienced by the frozen qubits and estimates the error $\hat{\mathcal{L}}$ imposed on the logical qubits. The bit-flip and phase-flip errors constituting $\hat{\mathcal{L}}$ can be estimated independently as shown in Fig. 33, assuming the quantum depolarizing channel is approximated as two independent BSCs. More specifically, the quantum polar decoder of Fig. 33 consists of two independent classical syndrome-based polar decoders used for estimating bit-flip and phase-flip errors, denoted as $\hat{L}_x$ and $\hat{L}_z$, respectively. The former decoder exploits the classical polar encoder G_N, while the latter relies on the encoder G_N^T. It is pertinent to mention here that the syndrome-based polar decoder already exists in the classical regime in the context of distributed source coding [178], [179], where it is used to find the source information. However, when exploited from the perspective of channel coding, the syndrome-based polar decoder finds the most likely error on the information word. This is in contrast to the the conventional polar decoder, referred to as the codeword-based polar decoder, which aims to find the most likely information word. Such a syndrome-based polar decoder is obtained from the codeword-based polar decoder of Fig. 15 by setting the values of the frozen bits according to the observed syndrome values, rather than the actual values of the frozen bits. Similarly, the channel LLRs are replaced by the channel error LLRs, i.e.
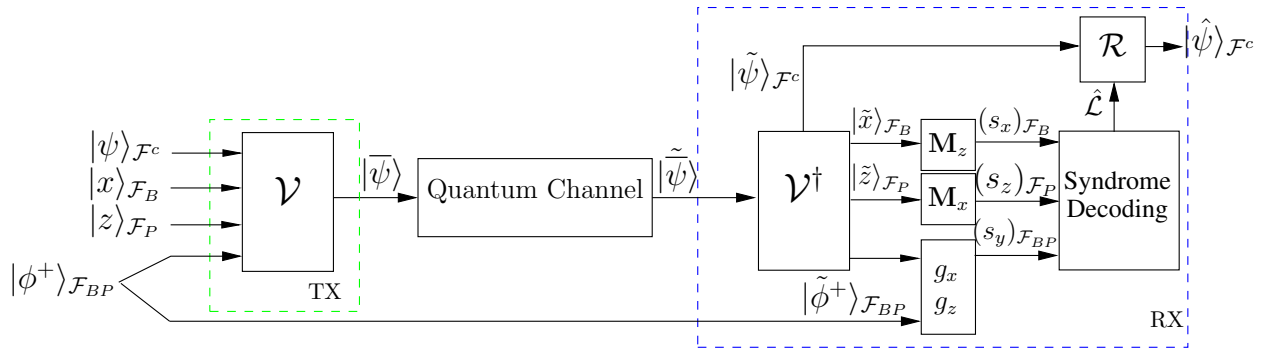
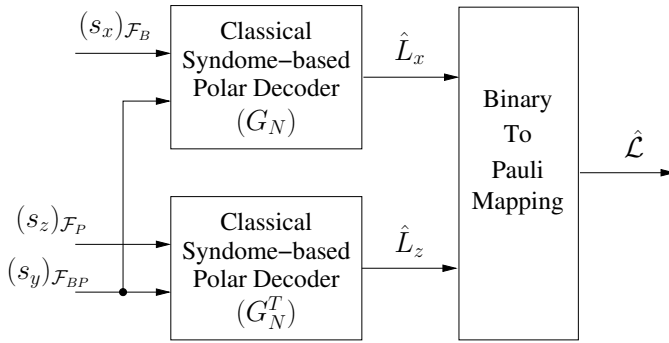Fig. 32: Schematic of a quantum communication system relying on a syndrome decoding based quantum polar code.

Fig. 33: Schematic of quantum polar decoder.

$(\hat{L}_z, \hat{L}_x)$	$\hat{\mathcal{L}}$	Estimated Error
$(0,0)$	**I**	No Error
$(0,1)$	**X**	Bit-flip Error
$(1,0)$	**Z**	Phase-flip Error
$(1,1)$	**Y**	Bit-and-phase-flip Error

TABLE XI: Binary to Pauli mapping for estimating the Pauli error.

the probability of encountering channel error on the coded sequence. In case of a depolarizing channel, or equivalently a BSC channel, the channel error LLRs are set according to the marginalized depolarizing probability $2p/3$. The rest of the decoding process is the same as that of the codeword-based decoding. Finally, the estimated error patterns $\hat{L}_x$ and $\hat{L}_z$ are mapped onto the corresponding Pauli operators using the binary-to-Pauli mapping of Table XI, hence yielding the Pauli error $\hat{\mathcal{L}}$.

VIII. CONCLUSIONS & FUTURE DIRECTIONS

To conclude, Arikan's polarization phenomenon of Section II has paved the way to provably approach the Shannon's capacity at low encoding and decoding complexity. In particular, polar codes have structured encoders and decoders, as discussed in Section III and Section IV, respectively, hence dispelling the notion that randomized coding structures are more apt for approaching Shannon's capacity. Furthermore, polar codes intrinsically support rate-adaptation, since the coding rate can be varied by only changing the number of frozen bits, while retaining the same encoder and decoder.

The attractive features of polar codes have stimulated a wave of interest in the research community as well as in the industry. Consequently, within a decade of their inception, polar codes have replaced their classic counterparts in the 5G NR for the control channels of the eMBB use-case. However, the high decoding latency of polar codes is a major concern restricting the widespread application of polar codes, as discussed in Section IV, where we have reviewed the major polar decoders with slow-paced tutorial examples. More specifically, recall from Table IX that the different polar decoding schemes, namely the SC of Section IV-A, the SSC of Section IV-B, the SCL of Section IV-C, the SCS of Section IV-D, the BP of Section IV-E and the SCAN of Section IV-F, entail a compromise between the imposed complexity, space requirements, performance and the tendency for fully-parallel implementation.

Another concern related to polar codes is their channel-specific nature, which necessitates the optimization of polar codes for the channel under consideration. Explicitly, the optimization of polar codes entails selecting the right indices for the frozen bit-channels. In Section V, we detail the polar code design principles with particular emphasis on the BEC-approximation, Monte-Carlo and GA-DE methods for estimating the reliabilities of the bit-channels. Furthermore, we construct frozen bit-channel sequences for the AWGN channel using these three methods and compare their performance. It is demonstrated that the sequences constructed using the Monte-Carlo and GA-DE are equally good, while those designed using the BEC-approximation have inferior performance. We surmise that, this is because a BEC does not accurately model transmission over an AWGN channel. We further demonstrate that, while the SNR is an important design parameter, limited variations in the design SNR only slightly affect the BLER performance.

Polar codes have also been warmly welcomed by the quan-

tum coding community, since the notion of channel polarization readily extends to the quantum channels. Interestingly, Arikan's polar encoder is capable of concurrently polarizing the bit-flip and phase-flip quantum channels, when the XOR gates are replaced by quantum CNOT gates, as exemplified in Section VI. However, the QSCD is not a direct extension of the classical SC decoder. The relevant contributions in this context are briefly summarized in Section VII. Nonetheless, there exist quantum polar codes designed for quantum Pauli channels, which are more directly linked to the classical polar codes, since they invoke the classical syndrome based polar decoders. The encoder and decoder of this class of quantum polar codes are reviewed in Section VII-A and Section VII-B, respectively.

As surveyed in this paper, intensive research efforts have been invested in the polar coding paradigm over the last decade for the sake of bringing it on par with its contemporaries, namely the turbo and LDPC codes. Nonetheless, there is a great potential to explore this coding paradigm further, since it is still in its infancy. Some of the potential research directions are discussed below:

1) **Non-Arikan Polar Codes:** As discussed in Section II-B, efforts have been made to design multi-dimensional as well as non-binary kernels for polar codes. However, these kernels have not been able to replace the Arikan's kernel, owing to the associated encoding and decoding complexities. This is still an open research area. In particular, the non-binary kernels are important from the perspective of source coding.

2) **Low-Latency, Power-Resource-Efficient & Flexible Polar Decoders:** Polar decoder continues to be a major concern; hence preventing the adoption of polar codes for the data channel of 5G NR. These include a range of open research problems both from the algorithmic perspective as well as from the implementation. In particular, the existing polar decoders incur a high latency, which is a primary concern for URLLC applications. Recall from Table IX that only the BP decoder supports fully-parallel implementation. However, this is achieved at cost both in terms of the performance, and its complexity as well as space requirement. substantial efforts have been made to partially parallelize the other polar decoders, for example in [79], [81], [82], [93], [121], but their latency still remains higher than that of the turbo and LDPC codes, which lend themselves to a convenient fully-parallel implementation [180]–[182]. Furthermore, there is a need to explore more practical polar decoders as well as to develop further the hardware implementations of the existing polar decoders to bring them at par with the turbo and LDPC codes, particularly from the perspective of latency (or throughput), power efficiency, resource efficiency as well as flexibility. From the algorithmic perspective, soft-in soft-out polar decoding is also a promising research avenue, since the existing soft SCL decoder is only applicable to systematic polar codes, while the performance of the SCAN decoder is not at par with

that of the SCL decoder. So, a more general soft SCL decoder is required for concatenated frameworks, for example concatenated coding schemes or joint detection and decoding schemes, invoking iterative decoding.

3) **Stochastic Polar Decoders:** Stochastic LDPC and turbo decoders are known to provide attractive benefits both in terms of fault-tolerance (to timing errors) as well as latency (or throughput) [183], [184]. However, stochastic polar decoders have not attracted much attention, apart from the contributions in [185]–[190]. Since the latency of polar decoders is already a prime concern, in this context it is worth investigating the area of stochastic polar decoders.

4) **Universal Polar Codes:** Perhaps another limitation of polar codes is their channel-specific nature, which necessitates code optimization for the required channel characteristics. The impact of this is not very significant in practical scenarios. Hence, despite the channel-specific nature of polar codes, they have been adopted for the control channels of the 5G NR. However, this still remains a concern, especially from the theoretical perspective. More specifically, polar codes with SC decoding incur a capacity loss over compound channels[15], as demonstrated in [191]. This capacity loss is due to the sub-optimal SC decoding and may be alleviated by invoking the optimal ML decoding, as theoretically shown in [35]. Explicitly, it was demonstrated in [35] that a polar code optimized for a BSC is universal under ML decoding and hence is optimal for any channel of equivalent capacity. Unfortunately, ML decoding of polar codes is not feasible. For the sake of overcoming this issue, universal polar codes were conceived in [192], [193], which essentially retain the low encoding and decoding complexity (per bit) of classic polar codes. However, this universality is achieved by increasing the length of polar codes, which in turn imply a higher overall complexity and longer latencies (or equivalently lower throughputs). This is a promising research avenue and must be explored further to find improved approaches for achieving universality for binary and non-binary classical polar codes as well as for quantum polar codes.

REFERENCES

[1] C. Shannon, "A mathematical theory of communication, bell system technical journal 27: 379-423 and 623–656," *Mathematical Reviews (MathSciNet): MR10, 133e*, 1948.

[2] R. W. Hamming, "Error detecting and error correcting codes," *Bell Labs Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[3] I. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 38–49, September 1954.

[4] D. E. Muller, "Application of boolean algebra to switching circuit design and to error detection," *Transactions of the IRE Professional Group on Electronic Computers*, no. 3, pp. 6–12, 1954.

---

[15]Compound channels model the transmission scenario where the exact channel is unknown and only a set of channels to which the actual channel belongs is known.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2937923, IEEE Communications Surveys & Tutorials

31

[5] A. Hocquenghem, "Codes Correcteurs d'Erreurs," *Chiffres (Paris)*, vol. 2, pp. 147–156, Sept. 1959.

[6] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and control*, vol. 3, no. 1, pp. 68–79, 1960.

[7] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[8] R. W. Watson and C. W. Hastings, "Self-checked computation using residue arithmetic," *Proceedings of the IEEE*, vol. 54, no. 12, pp. 1920–1931, 1966.

[9] N. S. Szabo and R. I. Tanaka, *Residue arithmetic and its applications to computer technology*. McGraw-Hill, 1967.

[10] *Blue Book: Recommendations for Space Data System Standards: Telemetry Channel Coding*. Consultative Committee for Space Data Systems, May 1984.

[11] P. Elias, "Coding for noisy channels," in *IRE International Convention Record*, 1955, pp. 37–46.

[12] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, January 1962.

[13] D. MacKay and R. M. Neal, "Good codes based on very sparse matrices," *Cryptography and Coding*, pp. 100–111, 1995.

[14] D. J. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics letters*, vol. 32, no. 18, p. 1645, 1996.

[15] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Technical Program of the IEEE International Conference on Communications, ICC '93 Geneva*, vol. 2, May 1993, pp. 1064–1070 vol.2.

[16] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Transactions on communications*, vol. 44, no. 10, pp. 1261–1271, 1996.

[17] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, July 2009.

[18] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on information theory*, vol. 42, no. 2, pp. 429–445, 1996.

[19] H. Nickl, J. Hagenauer, and F. Burkert, "Approaching Shannon's capacity limit by 0.2 dB using simple Hamming codes," *IEEE Communications Letters*, vol. 1, no. 5, pp. 130–132, 1997.

[20] D. Divsalar, S. Dolinar and F. Pollara, "Serial concatenated trellis coded modulation with rate-1 inner code," in *Proc. IEEE Global Telecommun. Conf.*, San Francisco, CA, Nov 2000, pp. 777–782.

[21] M. Tüchler and J. Hagenauer, "EXIT charts of irregular codes," in *Proceedings of Conference on Information Science and Systems*, Princeton University, 20-22 March 2002, pp. 748–753.

[22] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Transactions on Information Theory*, vol. 28, no. 1, pp. 55 – 67, Jan. 1982.

[23] ——, "Trellis-coded modulation with redundant signal sets part I: Introduction," *IEEE Communications Magazine*, vol. 25, no. 2, pp. 5–11, February 1987.

[24] ——, "Trellis-coded modulation with redundant signal sets part II: State of the art," *IEEE Communications Magazine*, vol. 25, no. 2, pp. 12–21, February 1987.

[25] E. Zevahi, "8-PSK trellis codes for a Rayleigh fading channel," *IEEE Transactions on Communications*, vol. 40, pp. 873–883, 1992.

[26] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation," *IEEE transactions on information theory*, vol. 44, no. 3, pp. 927–946, 1998.

[27] X. Li and J. A. Ritcey, "Bit-interleaved coded modulation with iterative decoding," *IEEE Communications Letters*, vol. 1, no. 6, pp. 169–171, 1997.

[28] P. Robertson and T. Worz, "Bandwidth-efficient turbo trellis-coded modulation using punctured component codes," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 206 – 218, Feb. 1998.

[29] V. Bioglio, C. Condo, and I. Land, "Design of polar codes in 5g new radio," *arXiv preprint arXiv:1804.04389*, 2018.

[30] Z. B. K. Egilmez, L. Xiang, R. G. Maunder, and L. Hanzo, "The development, design and performance of the 5G polar code," *IEEE Communications Surveys & Tutorials (submitted)*.

[31] K. Niu, K. Chen, J. Lin, and Q. T. Zhang, "Polar codes: Primary concepts and practical decoding algorithms," *IEEE Communications Magazine*, vol. 52, no. 7, pp. 192–203, July 2014.

[32] S. Shao, P. Hailes, T. Wang, J. Wu, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "Survey of turbo, LDPC and polar decoder ASIC implementations," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2019.

[33] A. Wyner, "A definition of conditional mutual information for arbitrary ensembles," *Information and Control*, vol. 38, no. 1, pp. 51 – 59, 1978. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0019995878900268

[34] E. Arikan, "A short course on polar coding (theory and applications)," *Lecture Notes, Center for Wireless Communications, University of Oulu*, May 2016.

[35] E. Sasoglu, "Polar coding theorems for discrete systems," 2011.

[36] S. B. Korada, E. Sasoglu, and R. Urbanke, "Polar codes: Characterization of exponent, bounds, and constructions," *IEEE Transactions on Information Theory*, vol. 56, no. 12, pp. 6253–6264, 2010.

[37] N. Presman, O. Shapira, and S. Litsyn, "Binary polar code kernels from code decompositions," in *2011 IEEE International Symposium on Information Theory Proceedings*, July 2011, pp. 179–183.

[38] N. Presman, O. Shapira, S. Litsyn, T. Etzion, and A. Vardy, "Binary polarization kernels from code decompositions," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2227–2239, May 2015.

[39] P. Trifonov and V. Miloslavskaya, "Twisted polar codes," in *International Symposium on Information Theory and its Applications*, Oct 2014, pp. 443–447.

[40] ——, "Polar codes with dynamic frozen symbols and their decoding by directed search," in *IEEE Information Theory Workshop (ITW)*, Sept 2013, pp. 1–5.

[41] ——, "Polar subcodes," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 254–266, Feb 2016.

[42] V. Bioglio and I. Land, "Polar-code construction of golay codes," *IEEE Communications Letters*, vol. 22, no. 3, pp. 466–469, March 2018.

[43] E. Şaşoğlu, E. Telatar, and E. Arikan, "Polarization for arbitrary discrete memoryless channels," in *IEEE Information Theory Workshop*, 2009, pp. 144–148.

[44] E. Sasoglu, "Polar codes for discrete alphabets," in *IEEE International Symposium on Information Theory Proceedings*, July 2012, pp. 2137–2141.

[45] R. Mori and T. Tanaka, "Channel polarization on q-ary discrete memoryless channels by arbitrary kernels," in *IEEE International Symposium on Information Theory*, June 2010, pp. 894–898.

[46] ——, "Non-binary polar codes using reed-solomon codes and algebraic geometry codes," in *IEEE Information Theory Workshop (ITW)*, 2010, pp. 1–5.

[47] ——, "Source and channel polarization over finite fields and Reed-Solomon matrices," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2720–2736, May 2014.

[48] N. Presman, O. Shapira, and S. Litsyn, "Polar codes with mixed kernels," in *IEEE International Symposium on Information Theory Proceedings*, July 2011, pp. 6–10.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2937923, IEEE Communications Surveys & Tutorials

32

[49] ——, "Mixed-kernels constructions of polar codes," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 239–253, Feb 2016.

[50] K. Niu, K. Chen, and J. Lin, "Beyond turbo codes: Rate-compatible punctured polar codes," in *IEEE International Conference on Communications (ICC)*, June 2013, pp. 3423–3427.

[51] R. Wang and R. Liu, "A novel puncturing scheme for polar codes," *IEEE Communications Letters*, vol. 18, no. 12, pp. 2081–2084, Dec 2014.

[52] V. Bioglio, F. Gabry, and I. Land, "Low-complexity puncturing and shortening of polar codes," in *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, March 2017, pp. 1–6.

[53] F. Gabry, V. Bioglio, I. Land, and J. Belfiore, "Multi-kernel construction of polar codes," in *IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2017, pp. 761–765.

[54] M. Benammar, V. Bioglio, F. Gabry, and I. Land, "Multi-kernel polar codes: Proof of polarization and error exponents," in *IEEE Information Theory Workshop (ITW)*, Nov 2017, pp. 101–105.

[55] E. Arikan, "Systematic polar coding," *IEEE Communications Letters*, vol. 15, no. 8, pp. 860–862, August 2011.

[56] H. Vangala, Y. Hong, and E. Viterbo, "Efficient algorithms for systematic polar encoding," *IEEE communications letters*, vol. 20, no. 1, pp. 17–20, 2016.

[57] G. Sarkis, I. Tal, and P. Giard, "Flexible and low-complexity encoding and decoding of systematic polar codes," *IEEE Transactions on Communications*, vol. 64, p. 7, 2016.

[58] G. T. Chen, Z. Zhang, C. Zhong, and L. Zhang, "A low complexity encoding algorithm for systematic polar codes," *IEEE Communications Letters*, vol. 20, no. 7, pp. 1277–1280, 2016.

[59] E. Arikan, "A performance comparison of polar codes and reed-muller codes," *IEEE Communications Letters*, vol. 12, no. 6, pp. 447–449, June 2008.

[60] ——, "A survey of reed-muller codes from polar coding perspective," in *IEEE Information Theory Workshop on Information Theory (ITW 2010, Cairo)*. IEEE, 2010, pp. 1–5.

[61] M. Mondelli, S. H. Hassani, and R. Urbanke, "From polar to reed-muller codes: A technique to improve the finite-length performance," in *IEEE International Symposium on Information Theory (ISIT)*. Ieee, 2014, pp. 131–135.

[62] E. Arikan and E. Telatar, "On the rate of channel polarization," in *IEEE International Symposium on Information Theory*, June 2009, pp. 1493–1495.

[63] N. Hussami, S. B. Korada, and R. Urbanke, "Performance of polar codes for channel and source coding," in *IEEE International Symposium on Information Theory*, 2009, pp. 1488–1492.

[64] E. Arıkan, H. Kim, G. Markarian, U. Ozgur, and E. Poyraz, "Performance of short polar codes under ML decoding," *Proc. ICT MobileSummit,(Santander, Spain)*, pp. 10–12, 2009.

[65] N. Goela, S. B. Korada, and M. Gastpar, "On LP decoding of polar codes," in *IEEE Information Theory Workshop (ITW)*, 2010, pp. 1–5.

[66] C. Leroux, I. Tal, A. Vardy, and W. J. Gross, "Hardware architectures for successive cancellation decoding of polar codes," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 1665–1668.

[67] C. Leroux, A. J. Raymond, G. Sarkis, I. Tal, A. Vardy, and W. J. Gross, "Hardware implementation of successive-cancellation decoders for polar codes," *Journal of Signal Processing Systems*, vol. 69, no. 3, pp. 305–315, 2012.

[68] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," *IEEE Communications Letters*, vol. 15, no. 12, pp. 1378–1380, December 2011.

[69] I. Tal and A. Vardy, "List decoding of polar codes," in *IEEE International Symposium on Information Theory Proceedings*, July 2011, pp. 1–5.

[70] ——, "List decoding of polar codes," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.

[71] K. Chen, K. Niu, and J. R. Lin, "List successive cancellation decoding of polar codes," *Electronics Letters*, vol. 48, no. 9, pp. 500–501, April 2012.

[72] K. Niu and K. Chen, "Stack decoding of polar codes," *Electronics Letters*, vol. 48, no. 12, pp. 695 –697, June 2012.

[73] C. Zhang, B. Yuan, and K. K. Parhi, "Reduced-latency SC polar decoder architectures," in *IEEE International Conference on Communications (ICC)*, June 2012, pp. 3471–3475.

[74] K. Niu and K. Chen, "CRC-aided decoding of polar codes," *IEEE Communications Letters*, vol. 16, no. 10, pp. 1668–1671, October 2012.

[75] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," *IEEE Communications Letters*, vol. 16, no. 12, pp. 2044–2047, December 2012.

[76] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Transactions on Communications*, vol. 60, no. 11, pp. 3221–3227, November 2012.

[77] S. Kahraman and M. E. Celebi, "Code based efficient maximum-likelihood decoding of short polar codes," in *IEEE International Symposium on Information Theory Proceedings*, July 2012, pp. 1967–1971.

[78] K. Chen, K. Niu, and J. Lin, "Improved successive cancellation decoding of polar codes," *IEEE Transactions on Communications*, vol. 61, no. 8, pp. 3100–3107, August 2013.

[79] 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*, vol. 61, no. 2, pp. 289–299, Jan 2013.

[80] G. Sarkis and W. J. Gross, "Increasing the throughput of polar decoders," *IEEE Communications Letters*, vol. 17, no. 4, pp. 725–728, April 2013.

[81] A. J. Raymond and W. J. Gross, "Scalable successive-cancellation hardware decoder for polar codes," in *IEEE Global Conference on Signal and Information Processing*, Dec 2013, pp. 1282–1285.

[82] ——, "A scalable successive-cancellation decoder for polar codes," *IEEE Transactions on Signal Processing*, vol. 62, no. 20, pp. 5339–5347, Oct 2014.

[83] U. U. Fayyaz and J. R. Barry, "Polar codes for partial response channels," in *IEEE International Conference on Communications (ICC)*. IEEE, 2013, pp. 4337–4341.

[84] ——, "Low-complexity soft-output decoding of polar codes," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 958–966, 2014.

[85] A. Pamuk, "An FPGA implementation architecture for decoding of polar codes," in *Wireless Communication Systems (ISWCS), 2011 8th International Symposium on*. IEEE, 2011, pp. 437–441.

[86] A. Pamuk and E. Arikan, "A two phase successive cancellation decoder architecture for polar codes," in *IEEE International Symposium on Information Theory*, July 2013, pp. 957–961.

[87] C. Zhang and K. K. Parhi, "Low-latency sequential and overlapped architectures for successive cancellation polar decoder," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2429–2441, May 2013.

[88] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast polar decoders: Algorithm and implementation," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 946–957, May 2014.

[89] Y. Zhang, A. Liu, X. Pan, Z. Ye, and C. Gong, "A modified belief propagation polar decoder," *IEEE Communications Letters*, vol. 18, no. 7, pp. 1091–1094, July 2014.

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2937923, IEEE Communications Surveys & Tutorials

33

[91] A. Balatsoukas-Stimming, A. J. Raymond, W. J. Gross, and A. Burg, "Hardware architecture for list successive cancellation decoding of polar codes," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 8, pp. 609–613, Aug 2014.

[92] B. Yuan and K. K. Parhi, "Early stopping criteria for energy-efficient low-latency belief-propagation polar code decoders," *IEEE Transactions on Signal Processing*, vol. 62, no. 24, pp. 6496–6506, Dec 2014.

[93] Y. Fan and C. Tsui, "An efficient partial-sum network architecture for semi-parallel polar codes decoder implementation," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3165–3179, June 2014.

[94] J. Lin, C. Xiong, and Z. Yan, "A reduced latency list decoding algorithm for polar codes," in *IEEE Workshop on Signal Processing Systems (SiPS)*, Oct 2014, pp. 1–6.

[95] C. Zhang and K. K. Parhi, "Latency analysis and architecture design of simplified SC polar decoders," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 2, pp. 115–119, Feb 2014.

[96] C. Xiong, J. Lin, and Z. Yan, "Symbol-based successive cancellation list decoder for polar codes," in *IEEE Workshop on Signal Processing Systems (SiPS)*, Oct 2014, pp. 1–6.

[97] ——, "Symbol-decision successive cancellation list decoder for polar codes," *IEEE Transactions on Signal Processing*, vol. 64, no. 3, pp. 675–687, Feb 2016.

[98] B. Yuan and K. K. Parhi, "Low-latency successive-cancellation polar decoder architectures using 2-bit decoding," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 4, pp. 1241–1254, April 2014.

[99] O. Afisiadis, A. Balatsoukas-Stimming, and A. Burg, "A low-complexity improved successive cancellation decoder for polar codes," in *48th Asilomar Conference on Signals, Systems and Computers*, Nov 2014, pp. 2116–2120.

[100] M. Mondelli, S. H. Hassani, and R. L. Urbanke, "Scaling exponent of list decoders with applications to polar codes," *IEEE Transactions on Information Theory*, vol. 61, no. 9, pp. 4838–4851, Sept 2015.

[101] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," *IEEE Transactions on Signal Processing*, vol. 63, no. 19, pp. 5165–5179, Oct 2015.

[102] B. Yuan and K. K. Parhi, "Low-latency successive-cancellation list decoders for polar codes with multibit decision," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 10, pp. 2268–2280, Oct 2015.

[103] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "On metric sorting for successive cancellation list decoding of polar codes," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 1993–1996.

[104] Y. Fan, J. Chen, C. Xia, C. Tsui, J. Jin, H. Shen, and B. Li, "Low-latency list decoding of polar codes with double thresholding," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 1042–1046.

[105] Y. Fan, C. Xia, J. Chen, C. Tsui, J. Jin, H. Shen, and B. Li, "A low-latency list successive-cancellation decoding implementation for polar codes," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 303–317, Feb 2016.

[106] C. Simsek and K. Turk, "Simplified early stopping criterion for belief-propagation polar code decoders," *IEEE Communications Letters*, vol. 20, no. 8, pp. 1515–1518, Aug 2016.

[107] J. Lin, C. Xiong, and Z. Yan, "A high throughput list decoder architecture for polar codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2378–2391, June 2016.

[108] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast list decoders for polar codes," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 318–328, Feb 2016.

[109] C. Xiong, J. Lin, and Z. Yan, "A multimode area-efficient SCL polar decoder," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 12, pp. 3499–3512, Dec 2016.

[110] K. Chen, B. Li, H. Shen, J. Jin, and D. Tse, "Reduce the complexity of list decoding of polar codes by tree-pruning," *IEEE Communications Letters*, vol. 20, no. 2, pp. 204–207, Feb 2016.

[111] S. A. Hashemi, C. Condo, and W. J. Gross, "A fast polar code list decoder architecture based on sphere decoding," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, pp. 2368–2380, Dec 2016.

[112] Z. Zhang, L. Zhang, X. Wang, C. Zhong, and H. V. Poor, "A split-reduced successive cancellation list decoder for polar codes," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 292–302, Feb 2016.

[113] B. Y. Kong, H. Yoo, and I. Park, "Efficient sorting architecture for successive-cancellation-list decoding of polar codes," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 7, pp. 673–677, July 2016.

[114] S. A. Hashemi, C. Condo, and W. J. Gross, "Simplified successive-cancellation list decoding of polar codes," in *IEEE International Symposium on Information Theory (ISIT)*, July 2016, pp. 815–819.

[115] X. Liang, J. Yang, C. Zhang, W. Song, and X. You, "Hardware efficient and low-latency CA-SCL decoder based on distributed sorting," in *IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[116] T. Wang, D. Qu, and T. Jiang, "Parity-check-concatenated polar codes." *IEEE Communications Letters*, vol. 20, no. 12, pp. 2342–2345, 2016.

[117] S. A. Hashemi, C. Condo, and W. J. Gross, "Fast and flexible successive-cancellation list decoders for polar codes," *IEEE Transactions on Signal Processing*, vol. 65, no. 21, pp. 5756–5769, Nov 2017.

[118] S. Shi, B. Han, J. Gao, and Y. Wang, "Enhanced successive cancellation list decoding of polar codes," *IEEE Communications Letters*, vol. 21, no. 6, pp. 1233–1236, June 2017.

[119] S. M. Abbas, Y. Fan, J. Chen, and C. Tsui, "High-throughput and energy-efficient belief propagation polar code decoder," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 3, pp. 1098–1111, March 2017.

[120] J. Choi and I. Park, "Improved successive-cancellation decoding of polar codes based on recursive syndrome decomposition," *IEEE Communications Letters*, vol. 21, no. 11, pp. 2344–2347, Nov 2017.

[121] V. Bioglio, F. Gabry, L. Godard, and I. Land, "Two-step metric sorting for parallel successive cancellation list decoding of polar codes," *IEEE Communications Letters*, vol. 21, no. 3, pp. 456–459, March 2017.

[122] Z. Liu, K. Niu, and J. Lin, "Parallel concatenated systematic polar code based on soft successive cancellation list decoding," in *International Symposium on Wireless Personal Multimedia Communications (WPMC)*. IEEE, 2017, pp. 181–184.

[123] L. Zhou, B. Feng, J. Jiao, K. Liang, S. Wu, and Q. Zhang, "Performance analysis of soft decoding algorithms for polar-staircase coding scheme," in *International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2018, pp. 1–6.

[124] S. A. Hashemi, C. Condo, F. Ercan, and W. J. Gross, "Memory-efficient polar decoders," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 7, no. 4, pp. 604–615, Dec 2017.

[125] P. Giard and A. Burg, "Fast-SSC-flip decoding of polar codes," in *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, April 2018, pp. 73–77.

[126] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Belief propagation list decoding of polar codes," *IEEE Communications Letters*, vol. 22, no. 8, pp. 1536–1539, 2018.

[127] C. Xia, J. Chen, Y. Fan, C. Tsui, J. Jin, H. Shen, and B. Li, "A high-throughput architecture of list successive cancellation polar codes decoder with large list size," *IEEE Transactions on Signal Processing*, vol. 66, no. 14, pp. 3859–3874, July 2018.

[128] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, March 1996.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2937923, IEEE Communications Surveys & Tutorials

34

[129] H. G. H. Hassan, A. M. A. Hussien, and H. A. H. Fahmy, "Radix-4 successive cancellation decoding of polar codes with partial sum lookahead," in *International Conference on Microelectronics (ICM)*, Dec 2017, pp. 1–4.

[130] F. Ercan, C. Condo, and W. J. Gross, "Reduced-memory high-throughput fast-SSC polar code decoder architecture," in *IEEE International Workshop on Signal Processing Systems (SiPS)*, Oct 2017, pp. 1–6.

[131] I. Dumer and K. Shabunov, "Soft-decision decoding of reed-muller codes: recursive lists," *IEEE Transactions on information theory*, vol. 52, no. 3, pp. 1260–1266, 2006.

[132] C. Condo, F. Ercan, and W. J. Gross, "Improved successive cancellation flip decoding of polar codes based on error distribution," in *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, April 2018, pp. 19–24.

[133] L. Chandesris, V. Savin, and D. Declercq, "Dynamic-SCFlip decoding of polar codes," *IEEE Transactions on Communications*, vol. 66, no. 6, pp. 2333–2345, June 2018.

[134] F. Ercan, C. Condo, and W. J. Gross, "Improved bit-flipping algorithm for successive cancellation decoding of polar codes," *IEEE Transactions on Communications*, vol. 67, no. 1, pp. 61–72, Jan 2019.

[135] Y. Zhou, J. Lin, and Z. Wang, "Improved fast-SSC-flip decoding of polar codes," *IEEE Communications Letters*, vol. 23, no. 6, pp. 950–953, June 2019.

[136] F. Cheng, A. Liu, Y. Zhang, and J. Ren, "Bit-flip algorithm for successive cancellation list decoder of polar codes," *IEEE Access*, vol. 7, pp. 58 346–58 352, 2019.

[137] F. Jelinek, "Fast sequential decoding algorithm using a stack," *IBM journal of research and development*, vol. 13, no. 6, pp. 675–685, 1969.

[138] N. Stolte, U. Sorger, and G. Sessler, "Sequential stack decoding of binary reed-muller codes," *ITG FACHBERICHT*, pp. 63–70, 2000.

[139] H. Aurora, C. Condo, and W. J. Gross, "Low-complexity software stack decoding of polar codes," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5.

[140] L. Xiang, Z. B. K. Egilmez, R. G. Maunder, and L. Hanzo, "CRC-aided logarithmic stack decoding of polar codes for ultra reliable low latency communication in 3GPP new radio," *IEEE Access*, 2019.

[141] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Belief propagation decoding of polar codes on permuted factor graphs," in *IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.

[142] 3GPP TS 38.212 V15.1.1, "NR multiplexing and channel coding," *3rd Generation Partnership Project Std. 3GPP*, 2018.

[143] H. Vangala, E. Viterbo, and Y. Hong, "A comparative study of polar code constructions for the AWGN channel," *arXiv preprint arXiv:1501.02473*, 2015.

[144] R. Mori and T. Tanaka, "Performance and construction of polar codes on symmetric binary-input memoryless channels," in *IEEE International Symposium on Information Theory*, 2009, pp. 1496–1500.

[145] ——, "Performance of polar codes with the construction using density evolution," *IEEE Communications Letters*, vol. 13, no. 7, pp. 519–521, July 2009.

[146] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6562–6582, 2013.

[147] R. Pedarsani, S. H. Hassani, I. Tal, and E. Telatar, "On the construction of polar codes," in *IEEE International Symposium on Information Theory Proceedings*, 2011, pp. 11–15.

[148] T. C. Gulcu, M. Ye, and A. Barg, "Construction of polar codes for arbitrary discrete memoryless channels," *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 309–321, 2018.

[149] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation," *IEEE Transactions on Information theory*, vol. 47, no. 2, pp. 657–670, 2001.

[150] J. Ha, J. Kim, and S. W. McLaughlin, "Rate-compatible puncturing of low-density parity-check codes," *IEEE Transactions on information Theory*, vol. 50, no. 11, pp. 2824–2836, 2004.

[151] M. Qin, J. Guo, A. Bhatia, A. G. i FÀ bregas, and P. H. Siegel, "Polar code constructions based on LLR evolution," *IEEE Communications Letters*, vol. 21, no. 6, pp. 1221–1224, June 2017.

[152] P. Yuan, T. Prinz, and G. Böcherer, "Polar code construction for list decoding," *CoRR*, vol. abs/1707.09753, 2017. [Online]. Available: http://arxiv.org/abs/1707.09753

[153] L. Huang, H. Zhang, R. Li, Y. Ge, and J. Wang, "Reinforcement learning for nested polar code construction," *CoRR*, vol. abs/1904.07511, 2019. [Online]. Available: http://arxiv.org/abs/1904.07511

[154] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Genetic algorithm-based polar code construction for the AWGN channel," *CoRR*, vol. abs/1901.06444, 2019. [Online]. Available: http://arxiv.org/abs/1901.06444

[155] ——, "Decoder-tailored polar code design using the genetic algorithm," *CoRR*, vol. abs/1901.10464, 2019. [Online]. Available: http://arxiv.org/abs/1901.10464

[156] Z. Babar, D. Chandra, H. V. Nguyen, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, "Duality of quantum and classical error correction codes: Design principles & examples," *IEEE Communications Surveys & Tutorials*, 2018.

[157] P. A. Dirac, *The Principles of Quantum Mechanics*. Oxford University Press, 1982.

[158] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, no. 5886, pp. 802–803, Oct. 1982. [Online]. Available: http://dx.doi.org/10.1038/299802a0

[159] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, "The road from classical to quantum codes: A hashing bound approaching design procedure," *IEEE Access*, vol. 3, pp. 146–176, 2015.

[160] ——, "Fifteen years of quantum LDPC coding and improved decoding strategies," *IEEE Access*, vol. 3, pp. 2492–2519, 2015.

[161] M. M. Wilde, *Quantum Information Theory*. Cambridge University Press, May 2013. [Online]. Available: http://arxiv.org/abs/1106.1445

[162] M. Wilde and S. Guha, "Polar codes for degradable quantum channels," *IEEE Transactions on Information Theory*, vol. 59, no. 7, pp. 4718–4729, July 2013.

[163] J. Renes and M. Wilde, "Polar codes for private and quantum communication over arbitrary channels," *IEEE Transactions on Information Theory*, vol. 60, no. 6, pp. 3090–3103, June 2014.

[164] M. Born, *The Born-Einstein letters*. Walker, 1971.

[165] G. Bowen, "Entanglement required in achieving entanglement-assisted channel capacities," *Phys. Rev. A*, vol. 66, p. 052313, Nov 2002. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevA.66.052313

[166] T. A. Brun, I. Devetak, and M.-H. Hsieh, "Correcting quantum errors with entanglement," *Science*, vol. 314, no. 5798, oct. 2006.

[167] M. Wilde and S. Guha, "Polar codes for classical-quantum channels," *IEEE Transactions on Information Theory*, vol. 59, no. 2, pp. 1175–1187, feb. 2013.

[168] C. W. Helstrom, "Quantum detection theory," 1971.

[169] A. S. Holevo, "Analog of a theory of statistical decisions in a noncommutative theory of probability," *Tr. Mosk. Mat. Obs.*, vol. 26, pp. 133–149, 1972.

[170] P. Sen, "Achieving the han-kobayashi inner bound for the quantum interference channel by sequential decoding. september 2011," *arXiv preprint arXiv:1109.0802*.

[171] J. M. Renes, F. Dupuis, and R. Renner, "Efficient polar coding of quantum information," *Phys. Rev. Lett.*, vol. 109, p. 050504, Aug 2012. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.109.050504

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2019.2937923, IEEE Communications Surveys & Tutorials

35

[172] M. Wilde and J. Renes, "Quantum polar codes for arbitrary channels," in *IEEE International Symposium on Information Theory*, july 2012, pp. 334 –338.

[173] J. M. Renes, D. Sutter, F. Dupuis, and R. Renner, "Efficient quantum polar codes requiring no preshared entanglement," *IEEE Transactions on Information Theory*, vol. 61, no. 11, pp. 6395–6414, 2015.

[174] A. J. Ferris and D. Poulin, "Tensor networks and quantum error correction," *Physical review letters*, vol. 113, no. 3, p. 030501, 2014.

[175] A. Steane, "Multiple-particle interference and quantum error correction," *Royal Society of London Proceedings Series A*, vol. 452, pp. 2551–2577, Nov. 1995.

[176] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Phys. Rev. A*, vol. 54, no. 2, pp. 1098–1105, Aug 1996.

[177] A. M. Steane, "Error correcting codes in quantum theory," *Phys. Rev. Lett.*, vol. 77, no. 5, pp. 793–797, Jul 1996.

[178] S. B. Korada and R. L. Urbanke, "Polar codes are optimal for lossy source coding," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1751–1768, 2010.

[179] S. Onay, "Polar codes for nonasymmetric slepian-wolf coding," *arXiv preprint arXiv:1208.3056*, 2012.

[180] R. Maunder, "A fully-parallel turbo decoding algorithm," *IEEE Transactions on Communications*, vol. 63, no. 8, pp. 2762–2775, Aug 2015.

[181] L. Xiang, M. F. Brejza, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "Arbitrarily parallel turbo decoding for ultra-reliable low latency communication in 3GPP LTE," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 4, pp. 826–838, April 2019.

[182] V. A. Chandrasetty and S. M. Aziz, "FPGA implementation of a LDPC decoder using a reduced complexity message passing algorithm," *Journal of Networks*, vol. 6, no. 1, p. 36, 2011.

[183] X. Zuo, I. Perez-Andrade, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "Improving the tolerance of stochastic ldpc decoders to overclocking-induced timing errors: A tutorial and a design example," *IEEE Access*, vol. 4, pp. 1607–1629, 2016.

[184] I. Perez-Andrade, S. Zhong, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "Stochastic computing improves the timing-error tolerance and latency of turbo decoders: Design guidelines and tradeoffs," *IEEE Access*, vol. 4, pp. 1008–1038, 2016.

[185] Z. Xu and K. Niu, "Successive cancellation decoders of polar codes based on stochastic computation," in *IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*. IEEE, 2014, pp. 908–912.

[186] B. Yuan and K. K. Parhi, "Successive cancellation decoding of polar codes using stochastic computing," in *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2015, pp. 3040–3043.

[187] X. Liang, C. Zhang, M. Xu, S. Zhang, and X. You, "Efficient stochastic list successive cancellation decoder for polar codes," in *28th IEEE International System-on-Chip Conference (SOCC)*. IEEE, 2015, pp. 421–426.

[188] B. Yuan and K. K. Parhi, "Belief propagation decoding of polar codes using stochastic computing," in *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2016, pp. 157–160.

[189] M. Xu, X. Liang, C. Zhang, Z. Wu, and X. You, "Stochastic BP polar decoding and architecture with efficient re-randomization and directive register," in *IEEE International Workshop on Signal Processing Systems (SiPS)*. IEEE, 2016, pp. 315–320.

[190] K. Han, J. Wang, and W. J. Gross, "Bit-wise iterative decoding of polar codes using stochastic computing," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*. ACM, 2018, pp. 409–414.

[191] S. H. Hassani, S. B. Korada, and R. Urbanke, "The compound capacity of polar codes," in *47th Annual Allerton Conference o Communication, Control, and Computing*. IEEE, 2009, pp. 16–21.

[192] S. H. Hassani and R. Urbanke, "Universal polar codes," in *IEEE International Symposium on Information Theory*. Ieee, 2014, pp. 1451–1455.

[193] E. Şaşoğlu and L. Wang, "Universal polarization," *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 2937–2946, 2016.