Chapter 10
Channel Coding in NR

Yufei Blankenship, Dennis Hui, and Mattias Andersson

To achieve the demanding performance targets of 5G NR, new channel coding techniques are introduced for both the data channels and the control channels. Compared to fourth-generation (4G) long-term evolution (LTE), low-density parity-check (LDPC) codes are introduced for the data channels, replacing the turbo codes of LTE. Similarly, polar codes are introduced for the control channels, replacing the tail-biting convolutional codes (TBCC) of LTE.

1 LDPC Coding in NR

1.1 Introduction

Robert Gallager introduced LDPC codes in his doctoral thesis [1] in 1963. Interest was renewed following the success of turbo codes in the early 1990s, and they were studied by several authors [2, 3]. Since then they have been incorporated in several standards such as DVB-S2, IEEE 802.16e, IEEE 802.11n, etc. LDPC codes were also considered during standardization of 4G LTE more than 10 years ago, though turbo codes with new QPP interleavers were ultimately adopted instead [4].

When channel coding techniques were investigated for 5G NR, it was decided that LDPC codes should be selected to replace the turbo code, considering the

Y. Blankenship
Ericsson Inc., Business Area Networks, Schaumburg, IL, USA

D. Hui (✉)
Ericsson Inc., Ericsson Research, Santa Clara, CA, USA
e-mail: dennis.hui@ericsson.com

M. Andersson
Ericsson AB, Ericsson Research, Stockholm, Sweden

stringent performance requirements of 5G. 5G NR targets very high throughputs with peak data rates of 20 Gbps in the downlink and 10 Gbps in the uplink, as well as ultra-reliable low-latency communication with block error rate (BLER) targets down to 1e-5. Compared to the 4G LTE turbo codes, the 5G NR LDPC codes offer the following advantages:

- Increased throughput, both in terms of area efficiency (as measured by Gbps/mm^2) and peak throughput
- Higher degree of parallelization leading to reduced decoding complexity and latency, especially at high code rates
- Improved performance in the error floor region, with no error floors above BLER 1e-5 regardless of code size or code rate

1.2 Coding Chain of NR Data Channel

In NR, LDPC codes are used in the downlink and uplink data channels (i.e., PDSCH and PUSCH). The NR LDPC coding chain includes code block (CB) segmentation, cyclic redundancy check (CRC) attachment, LDPC encoding, rate matching, and systematic-bit-priority channel interleaving; see Fig. 10.1. Specifically, code block segmentation allows very large transport blocks to be split into multiple smaller-sized code blocks which can be efficiently processed by the LDPC encoder/decoder. CRC bits are then attached to each code block for error detection purposes. Combined with the built-in error detection of LDPC codes through the parity-check equations, very low probability of undetected errors can be achieved for each code block. After applying the CRC attachment, LDPC encoding, rate matching, and bit interleaving steps to each code block individually, the coded bits of all code blocks are concatenated into a single bit sequence for transmission.

CRC Attachment

As shown in Fig. 10.1, two levels of CRC attachment are applied to a transport block: first CRC attachment to the entire transport block and then CRC attachment to each of the code blocks individually after code block segmentation.

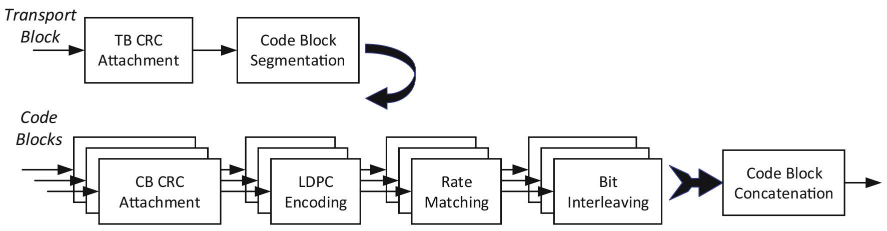


Fig. 10.1 NR LDPC coding chain at transmitter side

At the transport block level, CRC polynomials of two different degrees are used when generating CRC bits for a transport block. To reduce overhead, a CRC polynomial $g_{\mathrm{CRC16}}(D) = [D^{16} + D^{12} + D^5 + 1]$ of degree 16 is used for transport blocks shorter than or equal to 3824 bits. For larger transport blocks, a degree-24 polynomial $g_{\mathrm{CRC24A}}(D) = [D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1]$ is used.

In addition, when a transport block is segmented into two or more code blocks, 24 CRC bits generated using the polynomial $g_{\mathrm{CRC24B}}(D) = [D^{24} + D^{23} + D^6 + D^5 + D + 1]$ are attached to each CB. The addition of CRC bits to individual CBs allows for detailed hybrid automatic repeat request (HARQ) feedback at the CB level, which can be used to save radio resources via partial retransmission of a transport block. For example, if the HARQ feedback indicates that only a subset of the CBs of a given TB is incorrectly decoded, then the scheduler can request retransmissions of incorrectly decoded CBs only, without retransmitting the correctly received CBs belonging to the same TB.

Code Block Segmentation

Code block segmentation is depicted in Fig. 10.2. Let A (bits) be the transport block size generated by higher layer and B (bits) be the size of the transport block after CRC attachment. If B is smaller than the largest code block size K_{cb} for the given base graph, the transport block is not segmented.

Otherwise, the transport block is segmented into $C = \lceil B/(K_{\mathrm{cb}} - 24) \rceil$ equal-sized code blocks[1]. The 24 in the denominator accounts for the 24 CB CRC bits attached to each CB after segmentation. The size $K_{\mathrm{cb}} = 8448$ (bits) for base graph #1, and

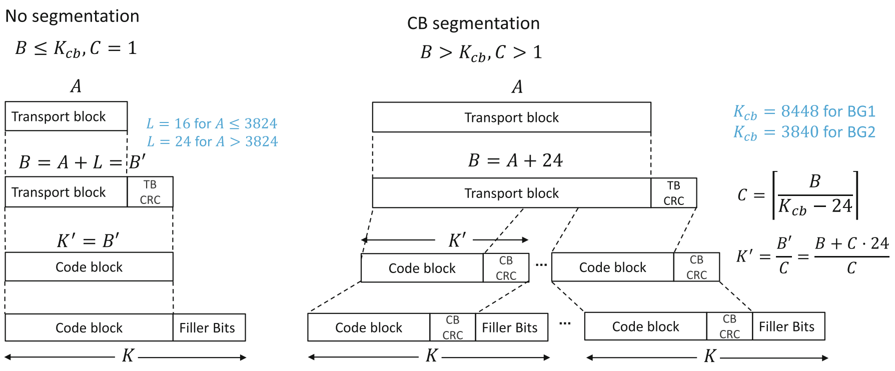


Fig. 10.2 Code block segmentation of NR data channel

[1]The transport block sizes are chosen in such a way that equal-sized code blocks are guaranteed.

$K_{\mathrm{cb}} = 3840$ (bits) for base graph #2. The number of information bits in each CB after segmentation and CB CRC attachment is $K^{'} = B/C + 24$.

NR LDPC Structure

NR LDPC codes are quasi-cyclic codes, where the parity-check matrix (PCM) H is defined by a small bipartite base graph represented by a binary matrix together with a set of shift coefficients [5]. To obtain the parity-check matrix, the base graph is expanded by replacing each entry by a $Z \times Z$ matrix. Each entry with value zero in the base graph is replaced by a $Z \times Z$ zero matrix, and each entry with value one is replaced by a shifted $Z \times Z$ identity matrix. The identity matrix is cyclically shifted to the right by a cyclic shift corresponding to the shift coefficient associated with the entry.

The base graph is represented by a binary matrix with M_{b} rows and N_b columns. The first K_{b} columns, where $K_{\mathrm{b}} = N_{\mathrm{b}} - M_{\mathrm{b}}$, correspond to information bits and are sometimes denoted as systematic columns. It has been noticed that LDPC code performance can be improved by including a small fraction of punctured variable nodes[2] with high degree. In NR, the $2 \times Z$ systematic bits corresponding to the first two columns in the base graph are always punctured.

After expansion, the parity-check matrix H has size of $M_{\mathrm{b}} \times Z$ rows, $N_{\mathrm{b}} \times Z$ columns. Using this H matrix, the number of information bits to be encoded is $K = K_{\mathrm{b}} \times Z$ (bits). After encoding, the number of coded bits available for transmission is $N = (N_{\mathrm{b}} - 2) \times Z$ (bits), since the first $2 \times Z$ systematic bits are always punctured. Thus, without considering further rate-matching operations, the native code rate based on the H matrix is $R = K/N = K_{\mathrm{b}}/(N_{\mathrm{b}} - 2) = (N_{\mathrm{b}} - M_{\mathrm{b}})/(N_{\mathrm{b}} - 2)$. The size of the base graph ($M_{\mathrm{b}} \times N_{\mathrm{b}}$) hence determines the native code rate.

The NR LDPC codes use a large number of degree-one, or single parity check, parity bits as can be seen by the identity sub-matrix of the base graph in the right part of Figs. 10.3 and 10.4, similar to the codes proposed in [6]. Puncturing a different number of these parity bits generates code words of different rates. This is especially useful for communication systems employing HARQ, allowing for the use of incremental redundancy instead of Chase combining for retransmissions. Rows and columns corresponding to punctured degree-one parity bits can be removed from the parity-check matrix when decoding, hence making the decoding complexity and latency smaller for higher code rates. This contrasts with the LTE turbo codes which have constant decoding complexity and latency irrespective of the code rate.

[2]Punctured variable nodes are sometimes referred to as state variable nodes.

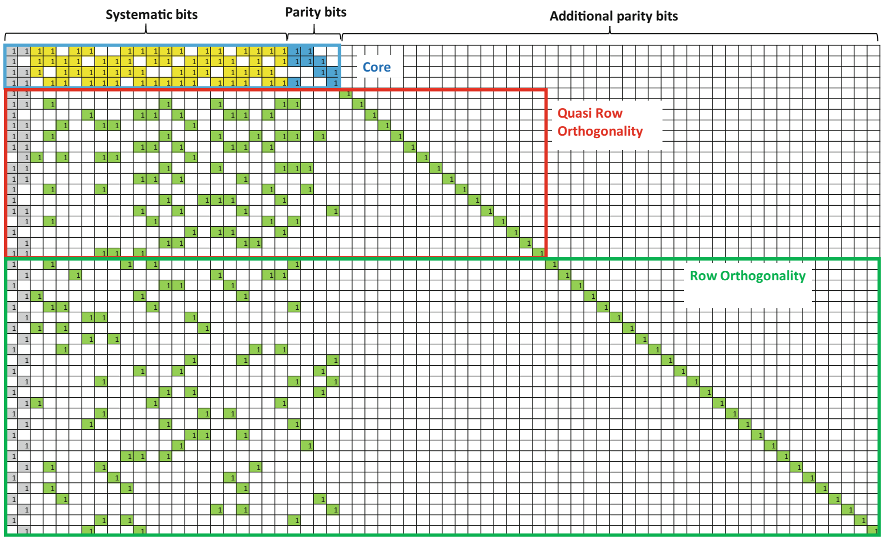


Fig. 10.3 Base graph #1

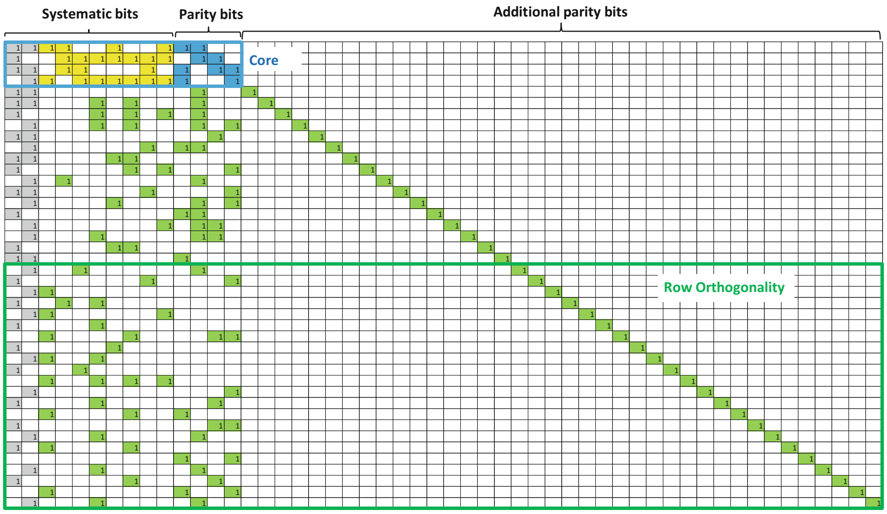


Fig. 10.4 Base graph #2

Two Base Graphs

In order to efficiently support a wide range of use cases with various throughput and reliability requirements, two different base graphs are adopted in NR. This allows NR to efficiently support mobile broadband use cases with large information block

Table 10.1 NR LDPC base graph parameters

Parameter	Base graph #1	Base graph #2
Native code rate of the base graph	1/3	1/5
Base graph size ($M_b \times N_b$)	46×68	42×52
Number of systematic columns (K_b)	22	10
Maximum information block size K ($=K_b \times Z_{max}$)	8448 bits($=22 \times 384$)	3840 bits($=10 \times 384$)
Number of nonzero elements in the base graph	316	197

Table 10.2 Shortening applied for BG2 for small transport block sizes

TB size B (bits) after CRC attachment	K_b	Range of Z
$640 < B$	10	$72 <= Z <= 384$
$560 < B <= 640$	9	$64 <= Z <= 72$
$192 < B <= 560$	8	$26 <= Z <=72$
$B <= 192$	6	$7 <= Z <= 32$

size and high code rate, as well as ultra-reliable low-latency communications with low code rate and small information block size.

Base graph #1 (BG#1) is designed for larger block lengths and higher code rates, while base graph #2 (BG#2) is designed for smaller block lengths and lower code rates. The design parameters of base graph #1 and base graph #2 are given in Table 10.1. Note that different lifting sizes Z as well as additional operations, such as shortening and puncturing, can be applied to obtain many other code rates and code sizes beyond those shown in Table 10.1.

To improve performance and to allow for more parallelization in the decoder, it can be beneficial to use a larger Z together with a smaller K_b for a given information block size. For this reason, shortening is additionally applied when BG#2 is used for small transport block sizes. This is done by choosing a smaller K_b and setting the value of the remaining $(10 - K_b) \times Z$ systematic bits to zero when encoding. These bits are referred to as filler bits and are discarded before transmission. The whole procedure is equivalent to removing the unused systematic columns from the base graph. Since the effect of using shortening to achieve higher level of parallelization is most prominent for the smallest information block sizes, the shortening scheme is applied to BG#2 but not BG#1 (Table 10.2).

Turning now to the BG#1 and BG#2 matrices illustrated in Figs. 10.3 and 10.4, it is observed that the matrices contain several carefully selected design features. These features allow the base graphs to simultaneously achieve the goals of superior code performance, simple encoder and decoder architecture, low decoding complexity, and low decoding latency.

Each base graph contains a very small core matrix of dense connection (i.e., dense ‘1’s), together with extension parts of sparse connection (i.e., sparse ‘1’s). We refer to the sub-matrix containing the top-left four rows and $(K_b + 4)$ columns in BG#1 and BG#2 as the core matrix. The column weight of the parity bits that are not in the core matrix is 1, and the corresponding bottom-right sub-matrix is the identity matrix. This structure implies that the parity-check matrix of the LDPC code obtained by puncturing some of the additional parity bits is given by the sub-matrix that does not include the corresponding punctured parity bits and the parity-check equations that they are involved in. Hence the corresponding decoding operations are no longer necessary. The core matrix of BG#1 is the 4 row-by-26 column sub-matrix at the top-left corner, i.e., $M_{\mathrm{b},0}^{(1)} = 4$, $N_{\mathrm{b},0}^{(1)} = 26$. Hence the highest code rate for which BG#1 can be used without extension[3] is given by:

$$R_{1,\mathrm{core}} = \frac{\left(N_{\mathrm{b},0}^{(1)} - M_{\mathrm{b},0}^{(1)}\right)}{\left(N_{\mathrm{b},0}^{(1)} - 2\right)} = \frac{22}{24} = 0.92.$$

Similarly, the core matrix of BG#2 is the 4 row-by-14 column sub-matrix at the top-left corner, i.e., $M_{\mathrm{b},0}^{(2)} = 4$, $N_{\mathrm{b},0}^{(2)} = 14$. Thus, the highest code rate that BG#2 can be used without extension is given by:

$$R_{2,\mathrm{core}} = \frac{\left(N_{\mathrm{b},0}^{(2)} - M_{\mathrm{b},0}^{(2)}\right)}{\left(N_{\mathrm{b},0}^{(2)} - 2\right)} = \frac{10}{12} = 0.83.$$

Larger sub-matrices are obtained by successively extending the core matrix toward the bottom-right corner, thus achieving lower code rates. For BG#1, if d_b additional number of rows and columns are used, the sub-matrix is composed of the top $\left(M_{\mathrm{b},0}^{(1)} + d_\mathrm{b}\right)$ rows and the left $\left(N_{\mathrm{b},0}^{(1)} + d_\mathrm{b}\right)$ columns of the BG#1 base graph, resulting in the code rate $\frac{\left(N_{\mathrm{b},0}^{(1)} - M_{\mathrm{b},0}^{(1)}\right)}{\left(N_{\mathrm{b},0}^{(1)} + d_\mathrm{b} - 2\right)}$. A similar procedure can be applied to BG#2. Example sub-matrix dimension and code rates are shown in Tables 10.3 and 10.4 for BG#1 and BG#2, respectively, assuming no additional shortening for BG#2. This demonstrates that when higher code rates are used, a smaller sub-matrix is used for encoding and decoding, leading to reduced complexity for encoding and decoding.

The code rates illustrated in Tables 10.3 and 10.4 are achieved by using different sub-matrices of the base graph. In these examples, the number of coded bits is an integer multiple of Z bits, for example, $\left(N_{\mathrm{b},0}^{(1)} + d_\mathrm{b} - 2\right) \times Z$ bits for BG#1. To provide full flexibility for NR, circular buffer-based rate-matching algorithm is applied to the output of LDPC decoder. The rate matcher can provide an arbitrary number of bits for transmission, corresponding to the finest granularity of code rate.

[3]During the code search process, BG#1 and BG#2 were designed for maximum code rates of 8/9 and 2/3, respectively. The sub-matrices corresponding to these code rates are referred to as the kernel of the matrices and have size 5×27 and 7×17 for the two base graphs. Additionally, code rates up to 0.95 can be achieved by puncturing parity bits in the core matrix.

Table 10.3 Example code rates provided by the sub-matrix of BG1 base graph, which is composed of the top $\left(M_{\mathrm{b},0}^{(1)}+d_{\mathrm{b}}\right)$ rows and the left $\left(N_{\mathrm{b},0}^{(1)}+d_{\mathrm{b}}\right)$ columns

d_{b} additional number of rows, columns	Top $\left(M_{\mathrm{b},0}^{(1)}+d_{\mathrm{b}}\right)$ rows	Left $\left(N_{\mathrm{b},0}^{(1)}+d_{\mathrm{b}}\right)$ columns	Code rate of the (sub-)matrix
1	5	27	22/25
9	13	35	2/3
20	24	46	1/2
42	46	68	1/3

Table 10.4 Example code rates provided by the sub-matrix of BG2 base graph, which is composed of the top $\left(M_{\mathrm{b},0}^{(2)}+d_{\mathrm{b}}\right)$ rows and the left $\left(N_{\mathrm{b},0}^{(2)}+d_{\mathrm{b}}\right)$ columns

d_{b} additional number of rows, columns	Top $\left(M_{\mathrm{b},0}^{(2)}+d_{\mathrm{b}}\right)$ rows	Left $\left(N_{\mathrm{b},0}^{(2)}+d_{\mathrm{b}}\right)$ columns	Code rate of the (sub-)matrix
3	7	17	2/3
8	12	22	1/2
18	22	32	1/3
28	32	42	1/4
38	42	52	1/5

Base graph #1, shown in Fig. 10.3, is optimized for larger information block sizes and higher code rates. It is designed for a maximum code rate of 8/9 and may be used for code rates up to $R = 0.95$. As discussed, the systematic bits corresponding to the first two columns (shown in gray) are never transmitted. It is observed that for the top four rows, the 4-by-4 parity block (shown in blue) has a dual-diagonal structure for easy encoding as well as good performance. For parity columns beyond the dual-diagonal structure, the simple diagonal structure (i.e., the green diagonal extending to bottom-right corner) allows for even simpler encoding. For the rows below the core (i.e., row 5 to row 46), the design strives to achieve low connection density (i.e., low density of '1's) and row orthogonality for efficient decoding without compromising the superior BLER performance. Lower connection density implies fewer decoding operations. Row orthogonality allows more parallel decoder processing, hence facilitating higher throughput LDPC decoder implementation. For base graph #1, row 5 to row 20 (shown in red box) have the property of quasi-row orthogonality, i.e., when excluding the first two columns, row i and row $i + 1$ of the base graph are orthogonal, $i = 5, \ldots, 20$. Row 21 to row 46 have the property of row orthogonality, i.e., row i and row $i + 1$ of the base graph are orthogonal (including the first two columns), $i = 21, \ldots 45$.

Base graph #2, shown in Fig. 10.4, is optimized for smaller information block sizes and lower code rates than base graph #1. It is designed with a smaller number of systematic columns than base graph #1, which gives a smaller information block size for the same Z. On the other hand, base graph #2 can reach code rate 1/5 without repeating coded bits. This is significantly lower than base graph #1 and the LTE turbo codes, which have design code rates of 1/3. To reach code rates below the design code rate, repetition is used, resulting in worse performance compared to using a code with lower design code rate. Base graph #2, and its additional coding gain at low code rates, is suitable for use cases that require very high reliability. Similar to base graph #1, the base graph #2 design carefully incorporates several features to achieve easy encoding, high-throughput decoding, as well as optimized code performance. For easy encoding, the parity portion contains the 4-by-4 dual-diagonal structure (shown in blue) as well as the simple diagonal structure beyond (i.e., the green diagonal extending to bottom-right corner). To facilitate high-throughput decoding, for the rows below the core (i.e., row 5 to row 42), row orthogonality is applied as much as possible without degrading code performance. In particular, row 21 to row 42 exhibit row orthogonality, i.e., row i and row $i + 1$ of the base graph are orthogonal, $i = 21, , \ldots 41$.

There is a need to support a large number of different information block sizes for a cellular system to support the wide range of use cases. As discussed, the base graphs are designed to encode information blocks with size $K_b \times Z$ (bits), where K_b is the number of systematic columns and Z is the lifting size. To support arbitrary, smaller, information block sizes than $K_b \times Z$, shortening is applied, where the last information bits are set to zero and not transmitted. In principle, shortening can be used to support any information block size smaller than $K_b \times Z$, but excessive shortening leads to performance degradation. Therefore, 51 different lifting sizes Z are used to define 51 different parity-check matrices for each base graph, with each Z supporting a different information block size. Shortening is still used to support intermediate information block sizes. In total, there are 102 parity-check matrices defined for the NR data channels. For comparison, we note that IEEE 802.11n only specifies 12 PCMs with 4 different code rates and 3 different information block sizes. The lifting sizes are of the form $Z = a \times 2^j$ to facilitate hardware implementation of arbitrary circular shifts of the $Z \times Z$ identity matrix. The possible Z values are listed in Table 10.5, ranging from 2 to 384.

The supported information block size range and code rate range by base graph #1 and #2 overlap, as can be seen in Table 10.1. In the overlapping region, one base graph, from the two candidate base graphs, needs to be selected for actual transmission of a transport block. The selection criteria are largely based on the error-correcting performance of the candidate LDPC codes, including the effect of

Table 10.5 Parameters for lifting sizes of the LDPC codes

		a							
Z		2	3	5	7	9	11	13	15
j	0	2	3	5	7	9	11	13	15
	1	4	6	10	14	18	22	26	30
	2	8	12	20	28	36	44	52	60
	3	16	24	40	56	72	88	104	120
	4	32	48	80	112	144	176	208	240
	5	64	96	160	224	288	352		
	6	128	192	320					
	7	256	384						

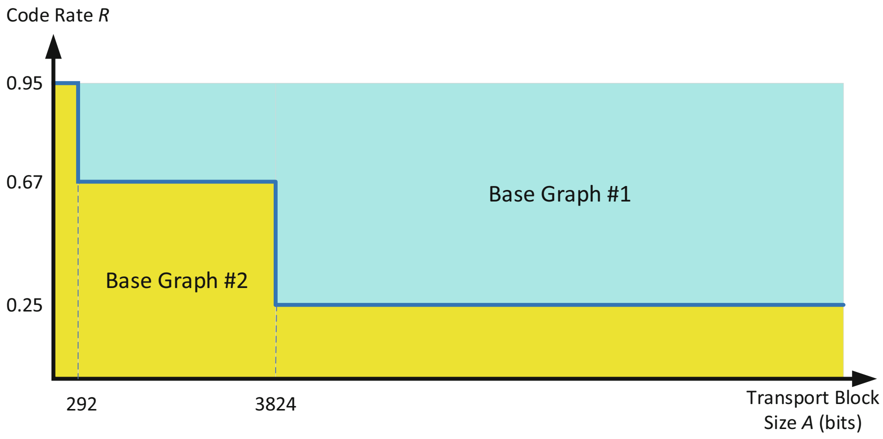


Fig. 10.5 Base graph selection according to transport block size A and code rate

puncturing, repetition, code block segmentation, etc. Overall, the range of transport block sizes A and code rates R covered by each of the two base graphs is shown in Fig. 10.5.

In general, base graph #1 is used for higher code rates and base graph #2 for lower code rates. The information block size is also taken into account when determining the switching point, due to performance differences between the two base graphs for different information block sizes. Base graph #2 is used for all code rates for $A \leq 292$ bits. For $292 < A \leq 3824$, BG#1 is used for code rates above 2/3, with BG#2 used for code rates lower than 2/3. For $A > 3824$ bits, the maximum information block size of $K = 3840$ for BG#2 is reached, when taking into account the 16 TB CRC bits. Hence BG#1 is used for all combinations of $A > 3824$ and $R > 1/4$. BG#2 is used for all cases with $R \leq 1/4$, due to the additional coding gain available from the design rate of 1/5 for BG#2. BG#1 has a design rate of 1/3, but the switching point is at code rate 1/4 due to considerations from code block segmentation. Base graph #1 supports a larger maximum information block size of $K = 8448$, so code block segmentation results in fewer code blocks for a given transport block size, when BG#1 is used. This offsets the additional coding gain from BG#2 for $1/4 < R < 1/3$. That is, for a given transport block size, BG#1 with repetition to reach down to $R = 1/4$ gives better TB-level BLER performance than BG#2, where TB-level decoding success requires all of its code blocks to be successful.

The rate-matching simulation studies show that BG#1 performance is good for 256 QAM up to code rate 0.9375 and good for QPSK up to code rate 0.9565. Hence it was decided that UE can skip decoding with either BG#1 or BG#2 when the effective code rate is greater than 0.95, where the effective code rate refers to the code rate obtained when only actually transmitted coded bits are counted.

Table 10.6 Starting point in the circular buffer for the four RV indices

RV	Starting coded bit index for BG#1	Starting coded bit index for BG#2
0	0	0
1	$17 \times Z$	$13 \times Z$
2	$33 \times Z$	$25 \times Z$
3	$56 \times Z$	$43 \times Z$

1.3 Rate Matching for LDPC Codes

To be able to provide an arbitrary number of coded bits (hence arbitrary code rate) as needed, the circular buffer-based rate-matching algorithm is defined. Coded bits at the output of LDPC encoder are entered into the circular buffer and read out consecutively starting at a predefined point on the circular buffer. The predefined point is according to the redundancy version (RV) index, where the RV takes value of 0–3. In Table 10.6, the predefined starting point for each of the RV indices is shown for both base graphs. Note that the first $2 \times Z$ punctured systematic bits are never entered into the circular buffer; hence the first $2 \times Z$ systematic bits are never transmitted regardless of the code rate.

The full circular buffer size is $66 \times Z$ for BG#1, and $50 \times Z$ for BG#2. RV0, RV1, and RV2 are evenly distributed on the circular buffer (i.e., 0/4, 1/4, 2/4 of the full circular buffer size). The exception is RV3, where RV3 is shifted closer to the starting point of the circular buffer so that a transmission using RV3 is self-decodable even for high code rate since it allows quick wrap around to pick up systematic bits. Thus, together with RV0, two RV indices support self-decodability at high code rate.

The default redundancy version (RV) sequence for HARQ (re-)transmission is {RV0, RV2, RV3, RV1}, which has been shown by simulation to provide the best performance across all block sizes and all code rates. This is illustrated in Fig. 10.6.

1.4 Bit-Level Channel Interleaver for LDPC Codes

The systematic-bit-priority mapping (HSPA-like) was used in the bit-level channel interleaver. The rectangular interleaver improves performance by making systematic bits more reliable than parity bits for the initial transmission of the code blocks. Specifically, the interleaver is realized via a table with the number of rows equal to the modulation order, with the number of columns then determined by the number of coded bits.

The coded bits at the output of the LDPC rate matcher are written into the table row-by-row starting from the top-left corner and read out from the table column-by-column starting from the top-left corner. When the bits are read out and used

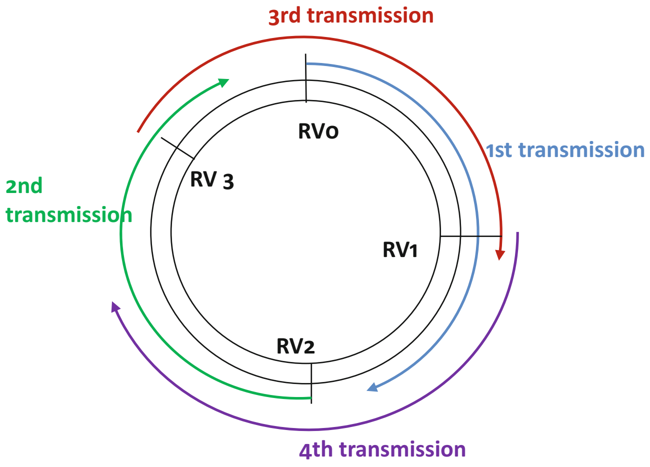


Fig. 10.6 Extraction of coded bits from the circular buffer for transmission assuming the default RV sequence of {RV0, RV2, RV3, RV1}

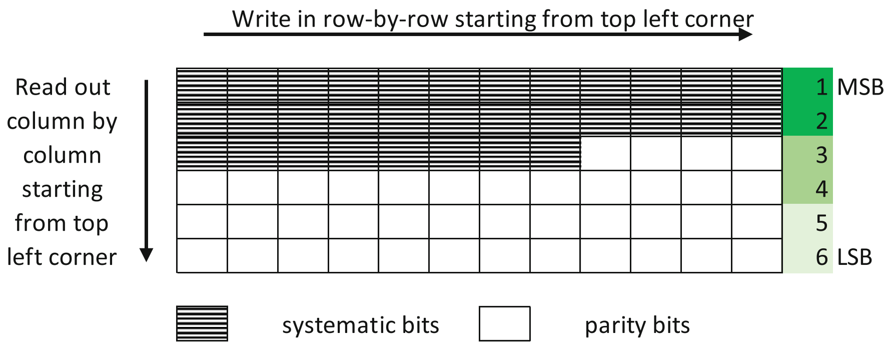

Write in row-by-row starting from top left corner
Read out column by column starting from top left corner
1 MSB
2
3
4
5
6 LSB
systematic bits
parity bits

Fig. 10.7 Illustration of bit-level channel interleaver for LDPC codes assuming modulation of 64-QAM

to generate modulation symbols, the bits at the top row are mapped to the most significant bit (MSB) of the modulation symbols, providing them with the highest bit reliability. Similarly, the bits at the bottom row are mapped to the least significant bit (LSB) of the modulation symbol, providing them with the lowest bit reliability. This effect exists for modulation order higher than QPSK. This tends to improve the LDPC decoding performance since the systematic bits are located toward the MSB when the redundancy version is 0, which the initial transmission of a transport block uses. This is illustrated in Fig. 10.7 assuming 64-QAM.

1.5 *Performance of NR LDPC Codes*

The performance of NR LDPC codes over an AWGN channel has been evaluated using a normalized min-sum decoder, layered scheduling, and a maximum of 20 decoder iterations.

Figure 10.8 shows the SNR required to achieve certain BLER targets as a function of information block size K for code rate 1/2 and QPSK modulation. The results show that NR LDPC codes provide consistently good performance over the full range of block sizes. According to the base graph selection rules, BG#2 is used for $K \leq 3840$ (including CRC bits), and BG# 1 is used for $K > 3840$ (bits). This accounts for the small jump in performance at $K = 3840$.

In Fig. 10.9, the 5G NR LDPC codes are compared with 4G LTE turbo codes for $K = 6144$ (bits), the largest information block size defined for the turbo codes. BG# 2 with two code blocks is used for rate 1/5, and BG#1 is used for the other code rates. The two code families show similar performance, except at high code rates where the LTE turbo codes have a tendency of an error floor. This can be seen in Fig. 10.9 where the LTE turbo code is 0.06 dB worse than the NR LDPC code at BLER = 1e-4. This error floor becomes higher at higher code rates; see [7] for a thorough evaluation of the LTE turbo codes for a large range of code rates and information block sizes.

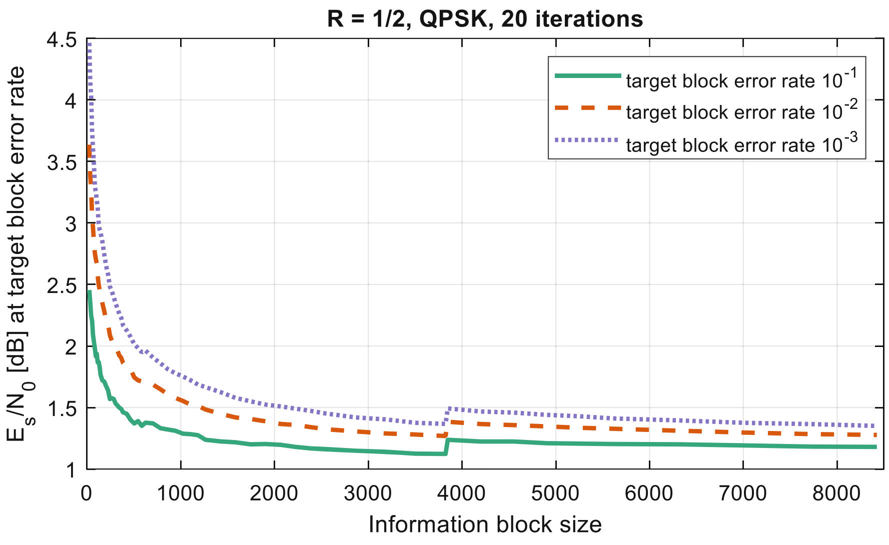


Fig. 10.8 Performance of NR LDPC codes at code rate 1/2 for QPSK modulation

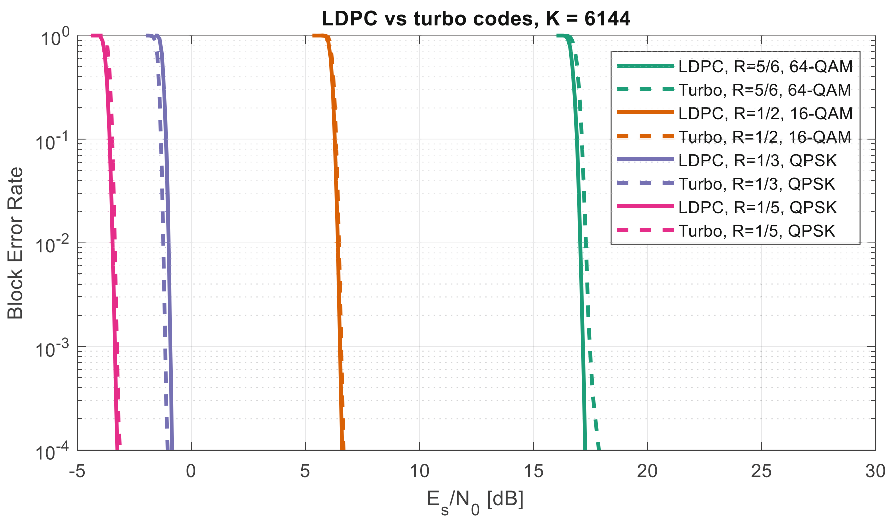


Fig. 10.9 Performance comparison between NR LDPC codes (solid) and LTE turbo codes (dashed)

2 Polar Coding in NR

2.1 Introduction

Polar codes were first introduced by Arıkan [8] who proved constructively that they can achieve the symmetric (Shannon) capacity of binary-input discrete memoryless channel using a low-complexity decoder, namely, a successive cancellation (SC) decoder. However, in practice, polar codes of finite size often exhibit a poor minimum distance property when used alone. As a result, despite being capacity-achieving, they do not perform well at high SNR regime even with exhaustive-search ML decoding [9]. Competitive performance is achieved only in concatenation with an outer code, such as a CRC code (or any parity-check code in general), together with the use of a successive cancellation list (SCL) decoder.

In NR, polar codes are used to protect the downlink control information (DCI), the uplink control information (UCI), as well as the system information in the physical broadcast channel (PBCH). DCI is transmitted over PDCCH, while UCI may be transmitted over PUCCH or PUSCH. As illustrated in the following sections, the NR polar code is significantly more complex than the tail-biting convolutional code (TBCC) used in 4G LTE control channels. The main advantage of polar codes over the TBCC, as well as the turbo codes used in 4G LTE and the NR LDPC codes, is that polar codes with SCL and CRC outer code typically yield better performance at moderate payload sizes (in the order of $K = 250$ bits or less). While not appropriate for data channels, small to moderate payload sizes are typically sufficient for transmitting control information and system information.

Polarization Theory

The core idea of polar coding is based on the transformation of a pair of identical binary-input channels into two distinct binary-input channels of different qualities. One of them is better, while the other is worse, than the original binary-input channel. To understand this transformation, consider the use of a polar code of length $N = 2$ to encode a pair of binary inputs, X_1 and X_2, into two coded bits, Z_1 and Z_2, that are sent over two identical channels with binary input to yield a pair of channel output, Y_1 and Y_2, as shown in Fig. 10.10.

The two inputs, X_1 and X_2, can be viewed as being transmitted individually over two distinct channels when a successive cancellation (SC) decoder is used. On the one hand, since the input X_1 is decoded before the input X_2, the input X_2 is unknown and acts as additional "noise" when the input X_1 is decoded, as depicted in the lower-left figure in Fig. 10.10. Because of this additional "noise," the input X_1 is effectively sent over a channel that has a worse reliability than the original binary-input channel. On the other hand, when decoding the input X_2 with a SC decoder, the input X_1 is already known (and presumably correctly decoded), and its impact can thus be nullified. As a result, the input X_2 can be viewed as being sent over a diversity channel with two outputs, Y_1 and Y_2, as illustrated in the lower-right figure in Fig. 10.10. The diversity channel clearly has a better reliability than the original binary-input channel with a single output, Y_2. Hence, when a SC decoder is used, a polar code of length two essentially "polarizes" two identical binary-input channels into a pair of distinct binary-input channels, commonly referred to as "bit-channels," with two different reliabilities.

Now consider the use of a polar code of length $N = 4$ over four identical binary-input channels, as depicted in the left figure in Fig. 10.11. Using the transformation illustrated above for polar code of length $N = 2$, one can view a polar code of length $N = 4$ as two independent polar codes of length $N = 2$ applied over two kinds of binary-input channel with different reliabilities, as shown in Fig. 10.11. The first polar code of length $N = 2$ has two inputs, X_1 and X_3, and is applied over a worse pair of identical binary-input channels ("channel –"). The second polar code has

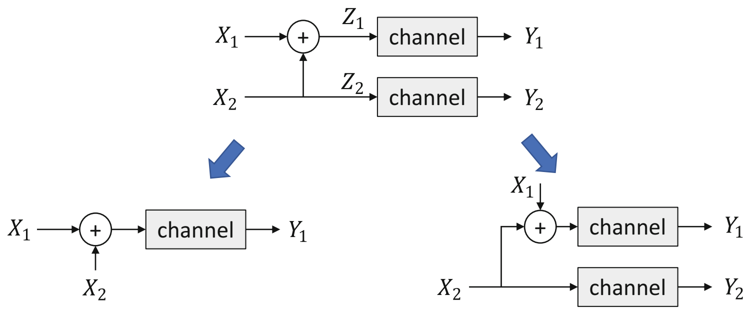


Fig. 10.10 Transformation of two identical channels into a better channel and a worse channel

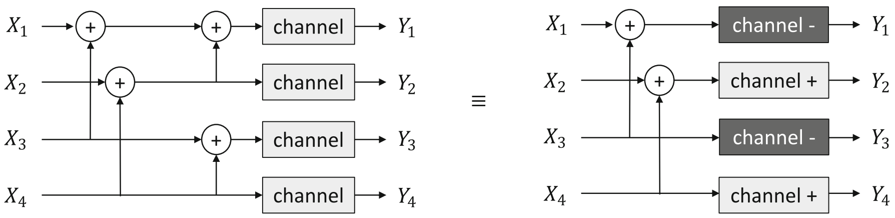
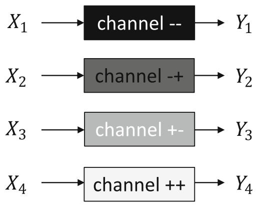


Fig. 10.11 Use of polar code of length four as two uses of polar code of length two, each applied over a different channel

Fig. 10.12 Four bit-channels of different reliabilities transformed from four identical binary-input channels via a length-4 polar code



two inputs, X_2 and X_4, and is applied over a better pair of identical binary-input channel ("channel +"). Applying the transformation again on each of these polar codes of length $N = 2$, one transforms the four original binary-input channels into four distinct bit-channels with four different reliabilities, as depicted in Fig. 10.12.

Such a pairwise polarizing operation can be repeated on a set of $N = 2^n$ independent uses of a binary-input channel for any given integer n to obtain a set of 2^n bit-channels of varying reliabilities. When n increases, some of these bit-channels become nearly perfect (i.e., error-free) while the others become nearly useless (i.e., totally noisy). Arıkan [8] showed that the fraction of nearly perfect bit-channels is exactly equal to the capacity of the original binary-input channel as n approaches infinity. The key idea of polar coding is to transmit data on the nearly perfect channels while fixing the input to the useless channels to fixed or frozen values (e.g., 0) that are known to the receiver. Frozen bits and non-frozen (or information) bits are commonly used to refer to the input bits to the nearly useless and the nearly perfect bit-channels, respectively. Information are only carried on the non-frozen bits. The set of non-frozen bit locations is often referred to as the *information set* and has a direct impact on the performance of a polar code. The number of data bits K to be communicated over the N independent binary-input channels determines the size of the information set.

Since a more reliable bit-channel is clearly more desirable than a less reliable bit-channel in terms of carrying information, a smaller information set (i.e., that for a smaller number of data bits) is always a subset of a larger information set (i.e., that for a larger number of data bits) for the same polar code. Consequently, an efficient way of specifying the information sets for different sizes is a sequence of indices to the bit-channels, termed an *information sequence*, which ranks the bit-channels in an ascending order of reliabilities. The order with which the bit-channels should be used to carry data can be derived from such an information sequence. For instance, the bit-channels corresponding to the last K indices in the information sequence should be used for carrying a given set of K data bits and so forth. Note that puncturing (or shortening) of coded bits can cause some bit-channels to become highly unreliable (i.e., incapable of information-carrying) and thus alter the order with which the bit-channels should be used (as described in more details later). Hence, the puncturing (or shortening) patterns and the information sequence should be designed jointly for polar codes.

2.2 Coding for Downlink Control Information

The core components in coding of DCI in NR are the cyclic-redundancy-check (CRC) encoder, the CRC interleaver, the polar encoding kernel, and the rate matcher, as illustrated in Fig. 10.13. All except the CRC interleaver are also included in the coding chain for uplink control information (UCI) in NR, as described in the next section.

Compared to the polar coding chain for UCI, the polar coding chain for DCI does not have a bit-level channel interleaver. This is due to the existence of interleaver for REG bundles before assigning the DCI coded symbols to the time-frequency resources. Furthermore, since the polar code is not required to support input size larger than 164 bits for DCI, segmentation procedure is not necessary for DCI coding chain.

CRC Encoding for DCI

CRC encoding is an important part of wireless communications. Traditionally, the CRC bits generated by a CRC encoder are used by the receiver to perform error detection in order to maintain a false-detection or false-alarm rate (FAR) below a certain target $P_{\mathrm{FAR}} \approx 2^{-n_{\mathrm{FAR}}}$. Operating with low FAR allows the network to selectively communicate different messages to a large group of different users through blind decoding over a common downlink control channel by scrambling the CRC bits differently. With polar coding, however, the CRC bits are also used for error correction to eliminate decoding paths in the list that fail CRC at the end of SCL decoding. With a target list size $L = 2^{n_L}$ in SCL decoding, the CRC length (i.e., the number of CRC bits) is given by $n_{\mathrm{CRC}} = n_{\mathrm{FAR}} + n_L$. For downlink control channel as well as uplink control channel, the target SCL list size is implicitly assumed to be $L = 8$ (i.e., $n_L = 3$) in the specifications of 5G NR [10].

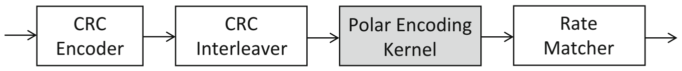


Fig. 10.13 NR coding chain for DCI

Since the number of blind decoding attempts of DCI in NR has been increased compared to LTE, the CRC for DCI has been lengthened to $n_{\mathrm{CRC}} = 24$ bits in NR. Accounting for the additional CRC length of 3 (i.e., $n_L = 3$) required for list decoding, one obtains an improved error detection capability of 2^{-21} (i.e., $n_{\mathrm{FAR}} = 21$) for DCI in NR, in comparison to 2^{-16} of LTE. The CRC polynomial for DCI is $g_{\mathrm{CRC24C}}(D) = [D^{24} + D^{23} + D^{21} + D^{20} + D^{17} + D^{15} + D^{13} + D^{12} + D^{8} + D^{4} + D^{2} + D + 1]$. Note that, among the 24 CRC bits, up to 7 of them are distributed among the information bits, and the remaining CRC bits are clustered at the end. To additionally provide identification (i.e., RNTI) of the UE being targeted by the DCI, 16 RNTI bits are scrambled onto the last 16 CRC bits, such that only the UE with the target identity can successfully pass the CRC check when performing the blind decoding.

The CRC bits are generated as usual by performing a long division of the data polynomial by a CRC polynomial, which is typically implemented by shift registers. When CRC is to be generated for DCI, the input vector is prepended with n_{CRC} ones, which achieves the effect of initializing the shift register by all ones.

CRC Interleaver

As in LTE, all CRC bits are typically clustered together and placed after the corresponding information block [9]. In NR, however, CRC bits are distributed more evenly among the data (non-frozen) and frozen bits using a CRC interleaver in downlink. The main purpose of this interleaver is to facilitate early error detection and the resulting early termination of the decoding process that reduces the latency and the average energy consumption of a polar decoder without performance degradation.

To enable early error detection during SC/SCL decoding, the value of each CRC bit must be computable using only the values of the data bits that come before the CRC bit. Hence, CRC interleaver must be designed under the constraint that each CRC bit is placed after all the data bits that the CRC bit depends on. CRC checks can then be performed at any of these CRC bits during the decoding process so that decoding can be terminated earlier when all surviving paths in the list fails a CRC check. Alternatively, these distributed CRC bits can be used to improve error correction capability. For example, any of these CRC bits can be used as dynamic frozen bits [11] to trim the list of surviving paths during SCL decoding for an improved error-correcting performance. However, each CRC bit can only be used either for early error detection or for improved error correction, but not for both.

CRC interleaving is the only component in the entire NR polar coding chain that is performed only in downlink but not in uplink. The size of the CRC interleaver in NR supports a maximum DCI payload size of 140 bits. Hence the maximum number of information bits at the input of the polar encoder of DCI is 164 bits, including 140 payload or data bits and 24 CRC bits.

Polar Encoding Kernel

Polar encoding kernel performs the basic polar encoding for a given mother code size, $N = 2^n$, that is a power of two and for an information set chosen as in [8] except without the bit-reversal permutation. More precisely, the output of NR polar encoding kernel is given by

$$\boldsymbol{x} = \boldsymbol{G}^{\otimes n}\boldsymbol{u},$$

where $\boldsymbol{G}$ is the 2-by-2 Arıkan kernel matrix given by

$$\boldsymbol{G} = \begin{bmatrix} \mathbf{1}\ \mathbf{0} \\ \mathbf{1}\ \mathbf{1} \end{bmatrix},$$

where $\otimes n$ denotes the n-time Kronecker power and $\boldsymbol{u}$ is the input to polar encoding kernel.

Mother Code Size

In principle, the power index n can simply be chosen such that the mother code size N is just large enough for the desired number of coded bits E (e.g., $n = \lceil \log_2 E \rceil$). In NR, however, a number of additional practical considerations have to be addressed, and the power index n is chosen based not only on E but also on the number of information bits K at the input of polar encoder, where K includes the CRC bits attached to the payload (data) bits, i.e., $K = A + n_{\mathrm{CRC}}$ with A being the payload size. First, when the code rate $R = K/E$ is small, and when E is only slightly higher than a power of two, it is desirable to choose a smaller n (i.e., $n = \lceil \log_2 E \rceil - 1$) and then repeat some of the coded bits, so that a smaller mother code size of 2^n (rather than twice as large, 2^{n+1}) is used for the polar coding kernel. This helps to reduce the latency and complexity of the encoder and decoder. The exact condition under which the smaller n is selected is when $E \leq 9/8 \times 2^{\lceil \log_2 E \rceil - 1}$ and the code rate $R = K/E < 9/16$. Second, at a very low code rate, using a smaller n with repetition of code bits yields similar performance as using a larger n. In NR, a lower bound of $R_{\min} = 1/8$ on code rate is imposed in the determination of the mother code size. Note that $R_{\min}$ is only used in the determination of n but does not limit the choice of code length E. Third, due to hardware limitations, there must be an upper limit on the mother code size. The upper bound on n is $n_{\max} = 9$ (i.e., mother code size $N_{\max} = 512$) for downlink and $n_{\max} = 10$ (i.e., mother code size $N_{\max} = 1024$) for uplink, as a base station can typically afford a higher computational load than a user equipment. Lastly, a lower limit of $n_{\min} = 5$ (i.e., mother code size $N_{\min} = 32$) is also imposed on the mother code size to avoid over-specification with little performance difference.

In summary, the Kronecker power index n is calculated as

$$n = \max\{n_{\min}, \min\{n_1, n_2, n_{\max}\}\}$$

where

$$n_1 = \begin{cases} \lceil \log_2 E \rceil - 1, \text{ if } E \leq \left(\frac{9}{8}\right) 2^{\lceil \log_2 E \rceil - 1} \text{and } \frac{K}{E} < 9/16 \\ \lceil \log_2 E \rceil, \qquad \text{otherwise.} \end{cases}$$

$$n_2 = \lceil \log_2 (K/R_{\min}) \rceil,$$

and where $R_{\min} = 1/8$, $n_{\min} = 5$, and $n_{\max} = 9$ for downlink while $n_{\max} = 10$ for uplink.

Information Sequence

A polar code of mother code size N provides a maximum of N bit-channels with different reliabilities. In NR, the relative reliabilities of bit-channels for $N = 1024$ are captured in an information sequence of length $N = 1024$ that lists the indices (from 0 to 1023) of bit-channels in an ascending order of reliabilities. The corresponding information sequence for a shorter length is nested within this information sequence of length $N = 1024$ in the sense that the sequence of length $N^{'}$, where $N^{'} < N$, can be obtained by removing the indices of values higher or equal to $N^{'}$ from the sequence of length N. Thus, the information sequence of any length smaller than $N = 1024$ can be derived from the information sequence of length $N = 1024$ specified in [10]. For example, the information sequence of length $N = 64$ is given by

0,1,2,4,8,16,32,3,5,9,6,17,10,18,12,33,20,34,24,36,7,11,40,19,13,48,14,21,35,26,37,25,22,38,41,28,42,49,44,50, 15,52,23,56,27,39,29,43,30,45,51,46,53,54,57,58,60,31,47,55,59,61,62,63.

Getting rid of the indices larger than or equal to 32 (in color red) yields the information sequence of length $N = 32$ as

0, 1, 2, 4, 8, 16, 3, 5, 9, 6, 17, 10, 18, 12, 20, 24, 7, 11, 19, 13, 14,

21, 26, 25, 22, 28, 15, 23, 27, 29, 30, 31.

For DCI, the maximum mother code size is $N = 512$. Its information sequence is derived from the $N = 1024$ sequence used for UCI by removing all indices larger than or equal to 512. Note that, in general, the best information sequence for a smaller mother code size may not be nested within that for a larger mother code size. However, it was found that the nested structure simplifies the specification with little performance impact.

The information sequence of length N is used to derive the locations of frozen and non-frozen bits of the polar encoding core of mother code size N. Specifically, the bit-channels corresponding to the last K indices in the information sequence that do not correspond to any punctured or shortened code-bit indices should be used for carrying a given set of K information bits.

Rate Matcher

The rate matcher is used to match the number of coded bits at the input of rate matcher (i.e., a coded bit sequence of length $N = 2^n$) with the amount of available radio resource. It in effect adjusts the code length from N bits to E bits, the number of bits that can be carried by the available radio resource. Three methods of code length adjustment, namely, puncturing, shortening, and repetition, are used in the rate matcher for NR polar codes. Similar to LDPC codes, puncturing refers to discarding coded bits that are generated based on the (unknown) information bits and do not have known values. Hence the punctured coded bits should be treated as unknown by setting the log-likelihood ration (LLR) to zero at the decoder input. Shortening refers to assigning known values (e.g., 0) to a selected set of non-frozen bit positions of the polar code so that a corresponding set of coded bits also have known values (e.g., 0), which are then discarded before transmission. Hence the shortened (i.e., discarded) coded bits should be treated as known (e.g., LLR $= \infty$) at the decoder input. Repetition refers to repeating a selected set of coded bits to arrive at a longer sequence of coded bits. Puncturing typically yields better performance at low code rates (i.e., lower K/E) while shortening performs better at high code rates (i.e., higher K/E). Repetition is used when the desired code length E is longer than the mother code size $N = 2^n$.

A size-N subblock interleaver and a circular buffer, as depicted in Fig. 10.14, are used to implement all three methods of code length adjustment.

The subblock interleaver is used to arrange the coded bits before placing them into the circular buffer so that they are discarded or admitted in a certain desired order. The interleaver simply takes the 5 most significant bits of the binary representation of the indices of the polar encoded bits and permutes them according to a pre-determined integer sequence of length 32 as illustrated in Fig.

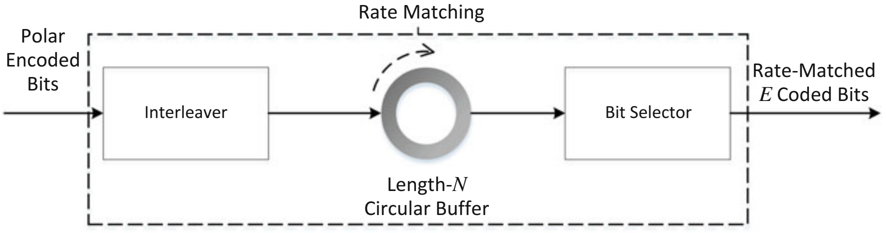


Fig. 10.14 Rate-matching process with interleaver, circular buffer, and bit selector

10.15. Equivalently, such a permutation first divides the N coded bits into 32 subblocks, each of size $N/32$, and then rearranges these subblocks according to the predetermined integer sequence.

The coded bits are selected from the circular buffer according to the relative sizes of E, N, and $A + n_{\mathrm{CRC}}$, as described in Table 10.7 and illustrated in Fig. 10.16, where n_{CRC} is the number of CRC bits and $R_{\mathrm{ps}} = 7/16$ is a threshold for determining whether puncturing or shortening should be used.

As mentioned earlier, discarding coded bits through puncturing or shortening can cause some bit-channels of a polar code to become highly unreliable in carrying information. Hence, depending on the number of discarded coded bits and the method of rate matching, a corresponding set of bit-channel indices needs to be skipped (or so-called pre-frozen) from the information sequence when forming the information set. For example, the indices of those coded bits that are discarded through puncturing or shortening are skipped from the information sequence when determining the information set.

Polar Coding for DCI

For DCI, the number of coded bits is determined by the grid of aggregation levels (AL) for transmitting DCI and can be calculated as follows. A resource element

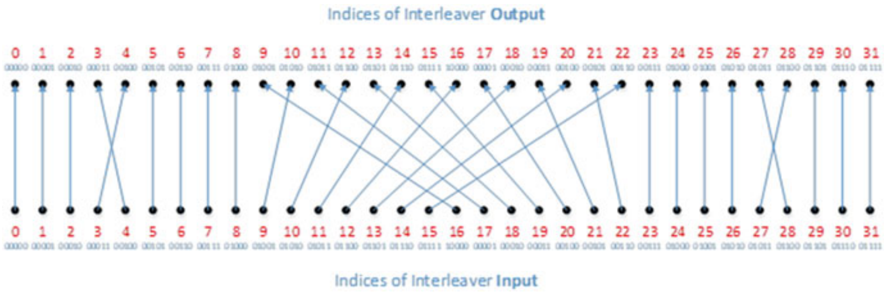


Fig. 10.15 Bipartite graph of rate-matching interleaver

Table 10.7 Bit selection from circular buffer for different rate-matching methods

Method	Condition	Operation
None	$E = N$	Extract all N bits from circular buffer starting from the first position
Puncturing	$E < N$ and $\frac{A+n_{\mathrm{CRC}}}{E} \leq R_{\mathrm{ps}}$	Extract E consecutive bits starting from the $(N - E + 1)$-th position of circular buffer
Shortening	$E < N$ and $\frac{A+n_{\mathrm{CRC}}}{E} > R_{\mathrm{ps}}$	Extract E consecutive bits starting from the first position of circular buffer
Repetition	$E > N$	Extract E consecutive bits starting from the first position of circular buffer and wrapping around when reaching the end of circular buffer

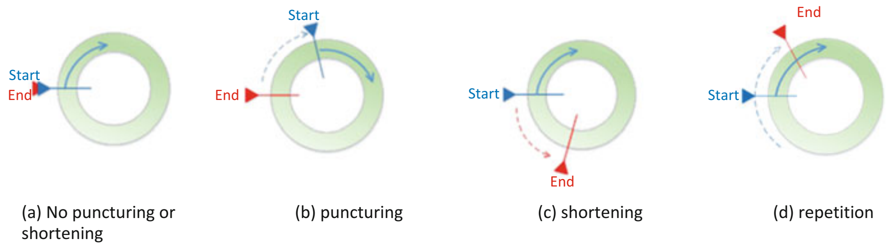


Fig. 10.16 Bit selection from circular buffer

Table 10.8 Number of coded bits and polar code mother size for polar code of DCI

Aggregation level (AL)	Number of coded bits E	Polar code mother code size N	Rate-matching method
1	108	128	Puncturing/shortening
2	216	256	
4	432	512	
8	864	512	Repetition
16	1728	512	

group (REG) consists of 12 resource elements (REs). A quarter of these REs in a REG are occupied by DMRS. Since QPSK modulation is used, the number of coded bits in each REG is thus $(12 \cdot 3/4) \cdot 2 = 18$ bits. For a control channel element (CCE), which is composed of 6 REGs, the number of coded bits is $6 \cdot 18 = 108$ bits. For a DCI candidate of aggregation level AL, a set of AL CCEs are associated with the corresponding time-frequency resources, and the number of coded bits E in the set is $E = 108 \cdot AL$ (bits). Table 10.8 shows the number of coded bits E for the different aggregation levels together with the polar code mother code size N and whether shortening/puncturing or repetition is applied per the rate-matching procedure. Since for DCI transmission $N_{\max} = 2^{n_{\max}} = 512$, repetition is always applied if $E > 512$ is needed.

As mentioned in Sect. 2.2, $R_{\min} = 1/8$ is used in determining the polar code mother code size. For $A_{\min} = 12$ with 24 CRC bits attached, the mother code size corresponding to $R_{\min} = 1/8$ is $N_2 = 2^{n_2} = 512$, where

$$n_2 = \left\lceil \log_2\left((12 + 24)/R_{\min}\right)\right\rceil = 9.$$

Thus, $R_{\min} = 1/8$ is not limiting in the determination of the DCI mother code size.

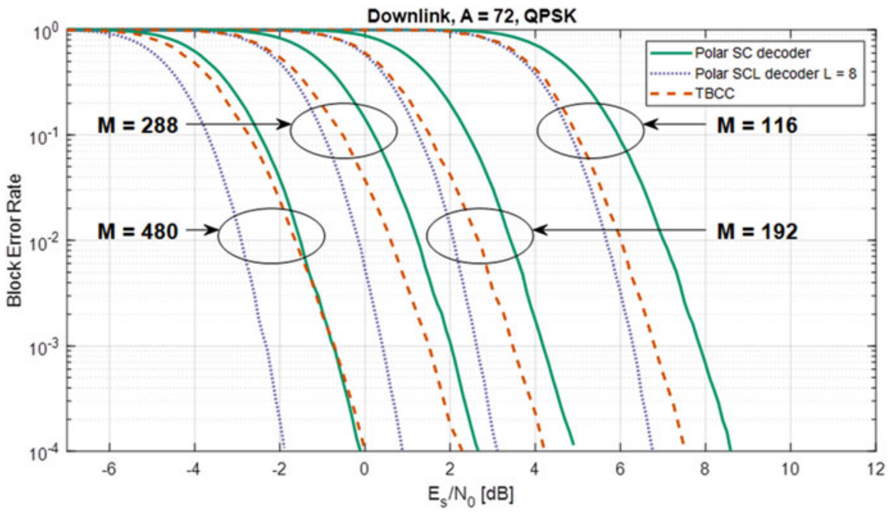


Fig. 10.17 Performance comparison of NR polar code in downlink with 24 CRC bits and LTE TBCC with 21 CRC bits for 72 payload bits

Performance of NR Polar Codes in Downlink

The performance of NR polar code in terms of block error rate (BLER) versus E_S/N_0 is shown in Fig. 10.17 for 72 payload bits in downlink. The performance of LTE TBCC for the same values of A and E is also shown, where the LTE TBCC is used in conjunction with a 21-bit CRC, instead of the 16-bit CRC in LTE DCI, for a fair comparison. NR polar code with SCL decoding outperforms LTE TBCC, especially at low code rates, as shown in the Fig. 10.17.

2.3 Coding for Uplink Control Information

When considering the full range of UCI bits supported in Rel-15, four types of channel coding schemes are used, as illustrated in Fig. 10.18. The parameters are:

- A: number of information bits excluding CRC parity bits
- K: number of information bits including CRC parity bits ($K = A + n_{\mathrm{CRC}}$)
- N: number of coded bits at the output of Polar encoding kernel
- E: rate-matching output size

Polar coding is applied when the number of UCI payload bits A is 12 bits or more. Channel coding schemes as defined in LTE are reused when UCI payload bits is 11 bits or fewer. Specifically, when the UCI payload size is between 3 and 11, the LTE Reed Muller code (32, K) is reused. A repetition code and a simplex code are used for payload size of 1 and 2, respectively.

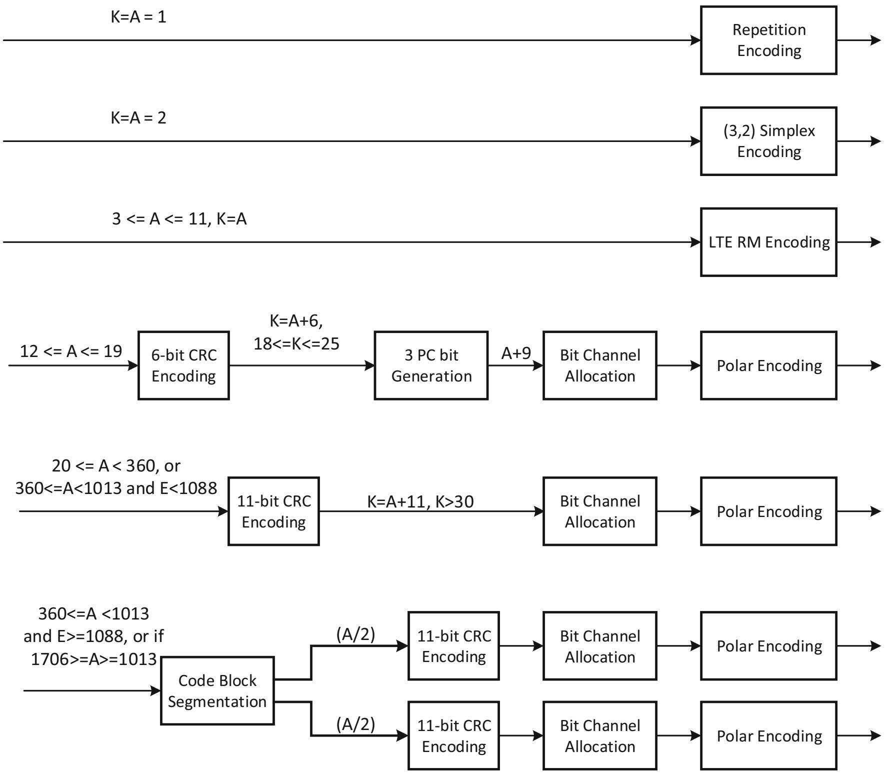


Fig. 10.18 Coding schemes used for UCI

Table 10.9 CRC polynomials for UCI

Range of K	Number of CRC bits	Nominal FAR	CRC Polynomial
$12 \le A \le 19$	6	2^{-3}	$g_{\mathrm{CRC}}\,(D) = D^6 + D^5 + 1$
$A \ge 20$	11	2^{-8}	$g_{\mathrm{CRC}}(D) = D^{11} + D^{10} + D^9 + D^5 + 1$

CRC Encoding for UCI

The CRC bits are generated as usual by performing a long division of the polynomial associated with the data bits by an CRC polynomial, which is typically implemented by shift registers. The CRC polynomials used for UCI for different payload sizes are shown Table 10.9. The shift registers are initialized to all zeros for UCI encoding. Since the target SCL list size of $L = 8$ (i.e., $n_L = 3$) is implicitly assumed, the nominal FAR is $2^{-n_{\mathrm{FAR}}} = 2^{-(n_{\mathrm{CRC}}-3)}$. There is no CRC interleaving in uplink. The CRC bits are appended to the data bits, and they are collectively allocated to the bit-channels of the polar code according to the information set derived from the information sequence.

Parity-Check (PC) Bits

For conventional polar codes, frozen bits typically take a constant value (commonly, 0) that is independent of the data and is known to the decoder. However, by making the frozen bit values dependent on the values of the non-frozen (i.e., information) bits, the error-correcting performance of NR polar code in uplink can be improved slightly in most cases. The use of this kind of data-dependent frozen bits is similar to the use of distributed CRC bits as dynamic frozen bits [11] in downlink as described before. Apart from dynamic frozen bits, they are also referred to as parity-check (PC) frozen bits, or simply PC bits, which is the terminology adopted in 5G NR specifications. The addition of these PC bits essentially forms another layer of outer code between the CRC outer code and the polar code.

To facilitate SCL decoding, the value of these PC bits must be readily determinable during successive decoding from previously decoded bits for each hypothesized decoding path in the list. Hence, each PC bit must be placed, according to the successive decoding order, after all the information bits that the PC bit depends on.

The addition of PC bits in principle should increase the minimum distance of the overall code. However, with successive cancellation decoding, it does not provide significant improvement in error-correcting performance.

Only when UCI has size satisfying $12 \leq A \leq 19$ in 5G NR, $n_{PC} = 3$ PC bits are added at the input of the polar encoding kernel. No PC bits are added outside this range of A.

Placement of PC Bits

In vast majority cases, the PC bits reside in the least reliable positions among the $(A + n_{CRC} + n_{PC})$ most reliable positions at the input of the polar encoder. The only exception occurs when $E - (A + n_{CRC} - n_L) > 192$ or, equivalently, $E > 195 + A$ (since $n_L = 3$ and $n_{CRC} = 6$ when the number of PC bits is nonzero). In this case, two of the three PC bits are placed in the least reliable positions among the $(A + n_{CRC} + n_{PC})$ most reliable positions, while one PC bit is placed elsewhere as specified later.

Replacing the conventional frozen bits with PC (frozen) bits in the least reliable positions among the aggregate of $(A + n_{CRC} + n_{PC})$ ensures no performance degradation in successive cancellation decoders. However, placing a PC bit elsewhere in a higher reliability position pushes another data bit or CRC bit to a lower reliable position and can thus cause performance degradation. In fact, it has been verified that there is a slight performance degradation when $E - A > 195$ precisely due to the fact that one of the three PC bits is not place in the lowest reliable positions among all data-dependent bits at the input of the polar encoder.

Now for the case when $E - A > 195$, two of the three PC bits are placed in the least reliable positions among the $(A + n_{CRC} + n_{PC})$ most reliable positions, while the remaining one bit is placed at the most reliable position among those positions with indices whose binary representations have the smallest weight (i.e.,

Table 10.10 Placement of PC Bits

Range of A and E	Number of PC bits	Placement of PC bits
$12 \leq A \leq 19$ and $E \leq 195 + A$	3	The 3 least reliable positions of the $(A + n_{\text{CRC}} + n_{\text{PC}})$ most reliable positions
$12 \leq A \leq 19$ and $E > 195 + A$	3	The 2 least reliable positions of the $(A + n_{\text{CRC}} + n_{\text{PC}})$ most reliable positions, plus the most reliable position among all positions with indices in the set $I = \{i \in \{0, 1, \cdots, N-1\} : w(i) = w_{\min}(A + n_{\text{CRC}})\}$, where $w(i)$ denotes the weight of the binary representation of index i, and $w_{\min}(m)$ denotes the minimum weight among the binary representations of the m most reliable positions
$A \geq 20$	0	N/A

the number of ones) among the $(A + n_{\text{CRC}})$ most reliable positions at the input of the polar encoder.

The placement of PC bits is summarized in Table 10.10.

Computation of PC Bits

For each PC bit location $k \in I_{\text{PC}}$, where I_{PC} denotes the set of the three PC bit locations if applicable, the value of the PC bit is computed based on the previously decoded information bits, frozen bits, or PC bits as follows:

$$u_k = \sum_{i \in S_k} u_i$$

where

$$S_k = \{i < k : i \notin I_{PC} \text{ and } \text{mod}\,(k - i, 5) = 0\},$$

u_i denotes the value of the bit at position i, and the summation is modulo-2 (i.e., binary addition or XOR). In other words, the PC bit value is simply the modulo-2 cumulative addition of every 5th bit value in front of the PC bit, except for the values of other PC bits. The computation of PC bits can be carried out by a length-5 cyclic shift register.

Polar Coding for UCI

In this subsection, we focus on polar coding only. Without considering the special case of PC bits, the polar coding chain is illustrated in Fig. 10.19.

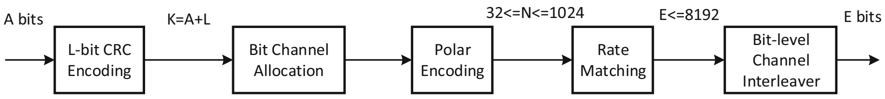


Fig. 10.19 Polar coding for UCI when PC bits are not applied

When the UCI is carried by PUCCH, the maximum UCI payload size supported by polar code is $A_{\max} = 2048 \times \left(\frac{5}{6}\right) \cong 1706$ bits, assuming the highest supported code rate of 5/6. This occurs when code block segmentation is applied, where each of the two segments is polar encoded with code rate 5/6 to $N_{\max} = 1024$ bits.

When UCI is carried by PUSCH, the same set of modulation orders and the same range of modulation symbols apply as UL data. Thus, the number of coded bits E for UCI can be very large, compared to DCI.

For each code block, the maximum number of coded bits after rate matching is $E_{\max} = 8192$ bits. This is due to the need to limit the maximum channel interleaver size, since the complexity of the triangular channel interleaver increases with the number of coded bits E. Considering that two segments are allowed, the total number of coded bits for the entire UCI payload is $2 \times E_{\max} = 2 \times 8192 = 16384$ bits. Since the maximum polar code size is $N_{\max} = 1024$ bits, repetition (part of rate matching) is applied, where the output of polar encoder is repeated eight times to arrive at $E_{\max} = 8192$ bits for each code block.

The supported modulation orders for UCI are the same as uplink data when UCI is carried by PUSCH. When OFDM is used for PUSCH, the modulation order can be QPSK, 16-QAM, or 64-QAM when the maximum modulation order is 64-QAM. The modulation order can be QPSK, 16-QAM, 64-QAM, or 256-QAM when the maximum modulation order is 256-QAM. When transform precoding is used for PUSCH with the maximum modulation order being 64-QAM, pi/2-BPSK is also supported for PUSCH.

Overall, the following applies:

1. When $12 \leq A \leq 19$ bits, 6-bit CRC is applied. Additionally, three PC bits are applied as PC-frozen bits to assist with polar decoding.
2. When $A \geq 20$ bits, 11-bit CRC is applied without PC bits.
   (a) When $20 \leq A < 360$ or $\{306 \leq A < 1013$, and $E < 1088\}$, code block segmentation is not applied. Only a single polar coding chain is applied.
   (b) When $\{360 \leq A < 1013$ and $E \geq 1088\}$ or $1706 \geq A \geq 1013$, code block segmentation is applied. The UCI payload is subdivided into two segments as evenly as possible. Each segment is encoded separately in parallel according to the coding chain illustrated in Fig. 10.20. At the end of channel interleaving, the two segments are concatenated without interlacing.

For polar coding chain of UCI, the rate-matching algorithm is the same as that of DCI. While no bit-level channel interleaver is applied for polar coding chain of DCI, a triangular channel interleaver is applied to polar coding chain of UCI.

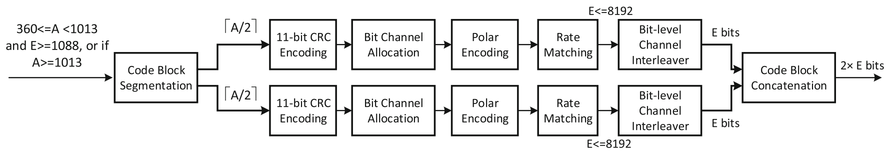


Fig. 10.20 Polar coding for UCI when segmentation is applied

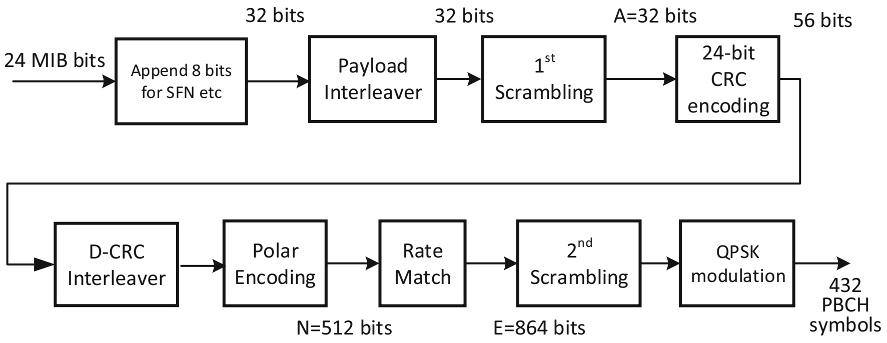


Fig. 10.21 Transmitter procedure for generating the modulation symbols of PBCH

2.4 Polar Coding for PBCH

Apart from DCI and UCI, polar code is also used for encoding of physical broadcast channel (PBCH) payload bits. The same polar code construction as that of physical downlink control channel (PDCCH) is used, with the mother code size of $N_{\max} = 512$ (bits). Similarly, the 24-bit CRC code and the associated interleaver designed for PDCCH is also adopted in PBCH. This allows the UE to implement a single polar decoder to cover both PDCCH and PBCH. The only difference between the encoding procedure of PBCH and PDCCH is that an interleaver is additionally applied to the payload of PBCH. The payload interleaver takes advantage of the unequal bit error probability (or unequal bit-channel reliabilities) provided by the polar code and match that with the unequal error protection needs of the payload bits. For instance, the system frame number (SFN) bits can be known under certain conditions; therefore they are mapped to least reliable information bit positions in the polar encoder and treated as frozen bits (value can be 1 or 0 depending on actual SFN bit values) if known a priori at the polar decoder.

The PBCH payload contains the Master Information Block (MIB). The total number of information bits at the input of encoding procedure is 32 bits. With 24-bit CRC attachment, the number of input bits at the polar encoder is 56 bits. The number of coded bits after polar encoding is 864 bits, which are then used to generate 432 modulation symbols with QPSK. The PBCH is broadcast periodically as a component of the SS/PBCH block. The procedure is illustrated in Fig. 10.21.

Since polar code of $N_{\max} = 512$ is used (the same as that of DCI), repetition is applied to generate the 864 output bits. The approximate polar code rate (including CRC bits) is 56/864 $\cong$ 1/15. Such low code rate is necessary for PBCH because MIB is the most fundamental system information that a device needs to obtain to gain initial access to the system.

References

1. R.G. Gallager, "Low density parity-check codes" (MIT Press, Cambridge, MA, 1963)
2. D. MacKay, R. Neal, "Good codes based on very sparse matrices," Cryptography and coding, 5th IMA Conf., C. Boyd, Ed., Lecture Notes in Computer Science, Oct. 1995
3. N. Alon, M. Luby, A linear time erasure-resilient code with nearly optimal recovery. *IEEE Transactions on Information Theory* **42**, 1732–1736 (1996)
4. ETSI TS 136 212 v12.2.0 (2014-10) LTE Release 12. Available: http://www.etsi.org/deliver/etsi_ts/136200_136299/136212/12.02.00_60/ts_136212v120200p.pdf
5. G. Liva, W.E. Ryan, M. Chiani, Quasi-cyclic generalized LDPC codes with low error floors. *IEEE Transactions on Communications* **56**(1), 49–57 (2008)
6. T.Y. Chen, K. Vakilinia, D. Divsalar, R.D. Wesel, Protograph-based raptor-like LDPC codes. *IEEE Transactions on Communications* **63**(5), 1522–1532 (2015)
7. Ericsson, "Performance evaluation of turbo codes and LDPC codes at higher code rates," 3GPP TSG RAN WG1 Meeting #85, doc. no. R1-164359, Nanjing, China, 23rd–27th May 2016. Available: http://www.3gpp.org/ftp/TSG_RAN/WG1_RL1/TSGR1_85/Docs/R1-164359.zip
8. E. Arıkan, Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on Information Theory* **55**, 3051–3073 (2009)
9. I. Tal, A. Vardy, "List decoding of polar codes," in *Proceedings of IEEE Symposium of Information Theory*, pp. 1–5, 2011
10. 3GPP 5G NR Specification, "Multiplexing and channel coding", TS 38.212, v15.0.0, 2018-01-03, Release15. Available: http://www.3gpp.org/ftp//Specs/archive/38_series/38.212/38212-f20.zip
11. P. Trifonov, V. Miloslavskaya, "Polar codes with dynamic frozen symbols and their decoding by directed search," *Proceedings of IEEE Information Theory Workshop*, pp. 1–5, Sept. 2013