**3GPP TSG RAN WG1 Meeting #xx R1-164039**

**Nanjing, China, May 23rd - 27th, 2016**

**Agenda Item: 7.1.5.1**

**Source: Huawei, HiSilicon**

**Title: Polar codes - encoding and decoding**

**Document for: Discussion and Decision**

# Introduction

To meet higher data rates and more diverse requirements of typical NR scenarios including eMBB, mMTC, and URLLC, a powerful and flexible channel coding scheme is one of the fundamental components of the NR access technology. In RAN1#84bis, various candidate schemes were discussed and evaluation assumptions considering convolutional codes, turbo codes, polar codes and LDPC codes were agreed [1].

In this contribution we focus on polar codes and describe encoding and decoding schemes. On the encoder side, we describe a flexible rate-matching scheme as a baseline reference design. Further optimization to reduce complexity is part of our investigation. On the decoder side, we focus on the adaptability of the decoder structure to different scenarios. Complexity and latency of the decoder are addressed in [2].

# Discussion on Polar Codes

## Principles of polar coding

Polar codes [3] can asymptotically (for code length going to infinity) achieve the capacity of any binary input symmetric memoryless channel with encoding and decoding complexity of the order *O(N log N)*, where *N* is the code length. At present they are the only class of channel codes that are provably capacity achieving with an explicit construction.

A Polar code is constructed by recursively applying a linear polarization transform to the binary input symmetric and memoryless channel W, expressed as 2-by-2 matrix . Repeated use of the transform, *n = log2(N)* times, results an N-by-N matrix , denoting the *n*-fold Kronecker product. Polar codes leverage a phenomenon known as channel polarization. Above transform together with a successive cancellation decoder structure turns the *N* available channels (*N* channel uses) in to another set of *N* bit-channels, referred to as synthesised channels, such that the capacities of these bit channels tend to 0 (fully unreliable) or to 1 (fully reliable) when *N* goes to infinity. In fact, the proportion of reliable channels, *K*, tends to the capacity of the original communication channel. Data is communicated by placing information bits on the *K* reliable channels and placing fixed bits, usually zeros, on the *N-K* unreliable channels. These bits on the unreliable channels are referred to as *frozen bits*, and the set of their positions is referred to as the *frozen set F* of size *N-K*. Frozen bits and the frozen set known by both the encoder and the decoder. In this way, a polar code of code length *N*, information word length *K*, and code rate *R=K/N* is constructed.

The original polar code construction allows for different rates by the choice of the size of the frozen set *F*. However, it allows only for lengths that are powers of two, i.e., *N = 2n*. Polar codes of other lengths may be constructed by puncturing or shortening. In puncturing, some code bits are not transmitted, while in shortening, some systematic bits are set to zero and not transmitted. In the following we explain first encoding of mother polar codes, i.e. of polar codes with lengths *N = 2n*. We then explain the constructions of polar cods of any lengths using a puncturing-based method.

## Encoding of punctured polar codes

Assume a mother polar code of length *N = 2n* and information word length *K*. We further assume that a frozen set *F* of size *N-K* is available. The codeword x is obtained by the transformation



where *u* is a vector of length *N*, with zeros for bits within the frozen set *F* (the frozen set) and *K* information bits in the remaining positions; the bit-reversal permutation matrix *Bn* (see below); and the transformation matrix *Fn*. The information word may contain a CRC. This CRC is usually present in most systems, it improves the distance properties of the polar code, and it can be used in the decoder (see under decoding below).

The bit-reversal permutation is known from applied mathematics. For length *N=2n* it is defined as follows. Consider the index sequence *an = (0, 1, …, N-1)*. Now for each element of *a*, the integer value is replaced by its binary representation, this binary representation is reversed (bit reversal), and this bit sequence is replaced by its integer representation. The resulting sequence *bn* is then the bit-reversal permutation vector. The matrix Bn is the corresponding bit-reversal permutation matrix.

With *puncturing-based rate-matching techniques*, a mother polar code of length *N=2n* is trimmed to a shorter code of length *Nt*, *Nt < N*, by puncturing (*N-Nt)* code bits. The index set of these code bits to be punctured is referred to as the puncturing pattern *P,* which has the size *N-Nt*. The information word length *K* is the same as that of the mother code; the size of the frozen set *F* is also equal to that of the mother code, i.e., *N-K*. The code rate of the punctured code is *Rr = K/Nt*. Note that the punctured polar code is defined by the frozen set *F* and the puncturing pattern *P*.

## Quasi-uniform puncturing (QUP)

In this contribution we focus on a specific puncturing scheme, referred to as Quasi-Uniform Puncturing (QUP) [4]. In the QUP scheme, the puncturing pattern is a function of the size of the puncturing pattern, and for each puncturing pattern the frozen set is determined by a density evolution using a Gaussian approximation (DE/GA). The QUP scheme can be considered as the baseline reference design for us to explore Polar Codes evaluation (The simulation results for polar codes based on QUP are shown in [5,6,7].) Further optimization of the rate-matching scheme for memory, latency, and complexity reduction is under the investigation.

**Construction of puncturing pattern and frozen set**

Consider the construction of a punctured polar code of length *Nt* and information word length *K* from a mother polar code of length *N=2n.* The ***puncturing pattern*** *P* of size *N-Nt* is chosen to be the first *N-Nt* of the bit-reversal permutation vector (see above). The ***frozen set*** *F* of size *N-K* is determined by ***density evolution using a Gaussian approximation (DE/GA)****,* tracking the mean values of the L-densities, as follows:

1. Select a design SNR, ; initialize the puncturing vector  such that positions in the puncturing pattern (punctured positions) are ones, and other positions are zeros.
2. Calculate the reliability of each synthesized channels using DE/GA.
3. Determine the frozen set *F* as the set of the *N-K* indices of the synthesized channels with the lowest reliabilities.

The computation of the reliabilities using DE/GA in Step 2 is performed with the following algorithm:

|  |  |  |
| --- | --- | --- |
| 0: |  |  |
| 9:  for  to  do |  |
| 10: |  |
| 1: | 11:      for  to   do |  |
| 2: for  to do | 12:                for  to   do |  |
| 3:         if  do | 13: |  |
| 4: | 14: |  |
| 5:         else | 15: | |
| 6: | 16: |  |
| 7:         end if | 17:                end for |  |
| 8: end for | 18:     end for |  |
|  | 19: |  |
|  | 20:end for |  |

where







With the puncturing pattern and the frozen set obtained, the punctured polar code is encoded as described above. For decoding, the LLRs for the punctured code bits are initialized with zero, and then the polar decoder of the other polar code is applied.

The design SNR cSNR characterizes the distribution of the LLRs obtained from the channel. This design SNR is not the SNR for which the code is eventually used [8] but has to be chosen for the design of the frozen set. According to our simulation results, the performance is not very sensitive to the selection of the value of the design SNR *cSNR*. Therefore, it is recommended to have SNR determined by the combination of *N* and TB size so that DE/GA is a function of *P* and *N* (two dimensions only). The cSNR values used in our simulations are listed in Appendix of [5].

**Complexity analysis of the QUP method**

Step 1: This is a simple initialization.

Step 2: The core of the algorithm is in lines 15 and 16; these lines require the function , its inverse , additions and multiplications; the complexity is dominated by the computation or table-lookup of the functions. In the for loop between lines 9 and 20, the variable *i* goes from 1 to *n*. For *i=1*, lines 15 and 16 are executed once; for *i=2*; they are executed twice; in general, they are executed 2i-1 times. Overall, the lines 15 and 16 are thus executed (2n -1) times (this is the worst case in which all bits are assumed as information bits). Note that lines 11-18 can be executed in parallel. If hardware has enough resource to unroll the lines 11-18, there is only n-stage delay.

Step 3: Determining the *N-K* smallest elements requires (partial) sorting.

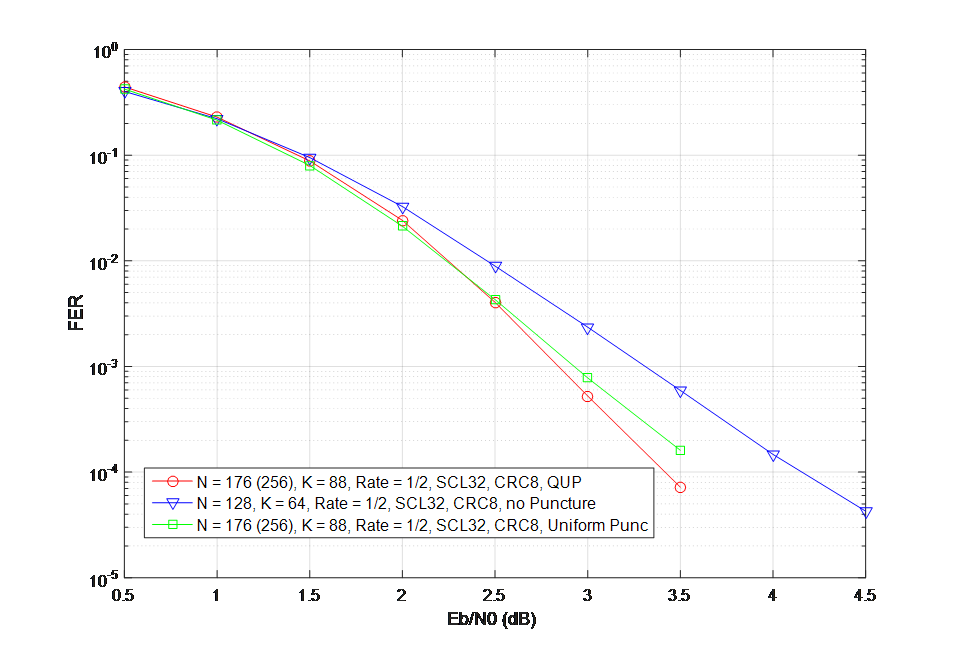
**Further optimization of the QUP scheme**

For the QUP scheme, there is opportunity for further optimization and memory/latency/complexity reduction:

1. The core functions  and  may be stored in a look-up table (LUT, ~4KB with 32-bit precision).
2. Lines 11-18 can be fully parallelized without sacrificing optimality. This is attractive for hardware implementation.
3. The sorting in Step 3 may be replaced by thresholding with pre-defined thresholds stored in a LUT (<15KB 4KB with 32-bit precision to support complete LTE+ combinations of code rates and code lengths) or some empirical formula. This would lead to K times comparison that can be full parallelized.

## Example of Uniform puncturing

QUP defines one example of the puncture pattern. In fact, there’s other possible puncture pattern. Figure 1 compares the performance of an example of uniform puncturing (UP) and QUP methods for rate-matching.



**Figure 1. UP and QUP methods.**

In the figure above, the blue curve is the BLER performance of SCL=32 decoder with K = 64-bit, N = 128-bit, R= 1/2, 8-bit CRC and no puncturing. The red curve is the BLER performance of SCL=32 decoder with K= 88-bit, N = 176-bit, R=1/2, and 8-bit CRC, using the QUP to puncture 80 bits from 256-bit mother code (R=1/3 -> R=1/2). The green curve is the BLER performance of SCL=32 decoder with K= 88-bit, N = 176-bit, R=1/2, and 8-bit CRC using UP scheme to puncture 80 bits from 256-bit mother code (R=1/3 -> R=1/2). We observe that there is no performance gap between the QUP and UP for the BLER at 10-2, while there is a gap for BLER of 10-3.

This example shows that other puncturing patterns or methods can be considered while maintaining a good performance/complexity trade-off.

## Other methods

There are several other methods to construct polar codes. For example random and structured puncturing for a fixed frozen set [9], shortening and subsequent optimisation of the frozen set using DE [10], joint optimisation of shortening and frozen set design [11].

One method is the cross-matrix-based method which pre-freezes some bits to ensure that the punctured bits are only related to the frozen bits. Clearly, the punctured bits will be known by the decoder just according to the related frozen bits. This can be realized by setting frozen bits with the same index of “1” in the “punctured” columns of the coding matrix. As the decoder knows the full information of mother code word, the equivalent code rate for polar construction is not changed by this rate-match scheme.  That is, the reliabilities of bit-channels are not changed for different code lengths if they have the same mother code length. One advantage of this scheme is to avoid the computation of the reliability metrics. The performance comparison for large mother code size of this method and the QUP method described before is illustrated in Figure 1..



**Figure 2. Cross-matrix and QUP schemes.**

## Decoding algorithms

From the channel observations, the LLRs of the code bits are computed. For a punctured polar code, the LLRs of the punctured code bits are set to zero. Then the decoder for the mother polar code is applied.

Three main decoding algorithms are available for polar codes:

*Successive cancellation (SC) decoding* [3]: In SC decoding, LLRs are propagated from right to left and hard decisions are propagated from left to right in the Tanner graph of the polar code. This algorithm has the lowest complexity and is proved to be asymptotically optimal (when the code length tends to infinity). Short and medium length codes suffer a performance loss. The SC decoder does make use of the CRC.

*SC list (SCL) decoding* [12]: The SC decoder performs SC decoding but maintains a list of candidates at each decoding step, where the list size is . During list decoding, the L paths with the best path metric are kept. At the end, the path with the best path metric is chosen as the final decoding result. List decoding has increased complexity but efficiently combats the error propagation in the simple SC decoding and thus leads to superior error-rate performance. Note that no CRC is used.

*CRC-aided SC list (CA-SCL)* [13]: The decoder operates the same way as the SCL decoder to produce the list of candidates. When decoding is finished, the CRC check is applied to the candidates in the list, and among those passing the CRC, the one with the best path metric is chosen. The complexity is the same as that of the SCL decoder. Note that the CRC is used here only for error detection and not for error correction.

The SC decoder can be regarded as a special case of the (CA-)SCL decoder, namely for list size *L=1*. Smaller list sizes lead to lower complexity, but poorer performance; the SC decoder ranges at the extreme end of these options. Therefore the options in polar decoding allow to trade error-rate performance versus decoding complexity. Complexity and latency of the decoding algorithms are analysed in [2].

## Forward compatibility

Polar codes including the decoding algorithms allow in various ways for forward compatibility. CA-SCL decoding with larger list sizes *L* typically leads to lower error rates. And more powerful codes can be obtained by a different choice of the frozen set. Typically SCL decoding with longer lists is then required.

As a result, when technology allows for decoders with larger list sizes, the decoding performance can be improved without any further changes. In that case, other designs of the frozen bits may be applied to increase performance further.

## Considerations on candidate coding schemes

An advantage of the class of Polar Codes is that different types of decoders are possible for the same encoder. Therefore Polar codes are suitable to be used in a wide range of scenarios with diverse requirements. In contrast, other candidate schemes, such as Turbo codes, LDPC codes, and (TB)CCs can not provide such flexibility. The power and area efficiency of Turbo decoders deteriorates very fast when increasing the block size. LDPC codes perform typically well for large blocks and high rates, however performance is poor for rates below ½. It is noted that such code rate range can be the most common scenario in eMBB case. Table 1 summarizes the suitability of the different channel coding schemes. The gray squares indicate that the channel coding scheme cannot meet the requirements for that application.

**Table 1**. Candidate coding schemes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Polar Codes** | **Turbo Codes** | **LDPC** | **(TB)CC** |
| **eMBB:**  **Large Block >8K** | Small-list decoder or SC decoder;  Good power and area efficiency | Low power/area efficiency; | Good power/area efficiency;  Good performance only when code rate is higher than ½. | Not considered |
| **eMBB:**  **1K~8K blocks**  **Fine granularity of code rates and code lengths** | Medium-List decoder;  Good performance for all code rates;  Support any code rates and code lengths | Good performance;  Support a fine granularity of code rates and code lengths; | Poor Performance for lower code rates lower than ½ ;  Difficulty to support fine granularity of code rates and code lengths | Not considered |
| **URLLC/Control-CH**  **/MTC-UL:**  **Small block**  **High reliability** | Large-List decoder;  Very good performance;  Support very low code rate | Poor performance with small block 🡪 not considered;  Error Floor  HARQ required for high reliability | Poor Performance with small block and low code rates  HARQ required for high reliability | VA decoder has poorer performance than SCL Polar  LVA decoder has much higher complexity than Polar List decoder. |
| **MTC-DL:**  **Small block**  **Low Power** | Small-list decoder or SC decoder for low-power realization | Poor Performance with small block 🡪 not considered | Poor Performance with small block 🡪 not considered | Viterbi Decoder for low-power realization;  Performance is worse than SCL decoder |

Furthermore the performance of Polar Codes keeps improving along with increasing list size of the SCL decoder. According to our simulation with list sizes up to 2048, we have not observed any performance saturation so far. In contrast, we observe that the performance of turbo codes and LDPC codes saturates with the number of decoding iterations. As a result, the operators can keep improving the system capacity with newer ASIC technology. For example, when the list size is increased from 32 to 128 (the decoding complexity increases about by a factor of four), the BLER performance will gain another 0.5 ~0.7dB for some control channels. Therefore the system capacity can simply be increased with the availability of new Polar decoding chips.

# Conclusion

In this contribution we have discussed encoding and decoding schemes for polar codes. The QUP method for rate-matching and some approximations have been illustrated. Further optimizations of these and other methods shall be further studied.

# References

1. Chairman’s notes RAN1 #84bis.
2. R1-164039 “On latency and complexity”, Huawei, HiSilicon
3. E. Arıkan, “Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels”, IEEE Transactions on Information Theory, vol. 55, no. 7, July 2009
4. Niu, K., Chen, K., and Lin, J. R., "Beyond turbo codes: rate-compatible punctured polar codes." 2013 IEEE International Conference on Communications, 2013.
5. R1-164375 “Evaluation of polar codes for eMBB scenario”, Huawei, HiSilicon
6. R1-164377 “Performance of channel coding schemes for eMBB scenario”, Huawei, HiSilicon
7. R1-164378 “Performance of channel coding schemes for mMTC and URLLC scenarios”, Huawei, HiSilicon
8. H. Vangala *et al.*, "A comparative study of polar code constructions for the AWGN channel." arXiv preprint arXiv:1501.02473 (2015).
9. A. Eslami and P. N. Hossein, "A practical approach to polar codes," IEEE Int. Symp. on Inf. Theory, 2011.
10. R. Wang and R. Liu, "A novel puncturing scheme for polar codes." IEEE Commun. Letters, 2014.
11. V. Miloslavskaya,"Shortened polar codes" IEEE Trans. on Info. Theory, 2015.
12. I. Tal and A. Vardy, “List decoding of polar codes,” IEEE Trans. Inf. Theory, vol. 61, no. 5, May 2015.
13. K. Niu and K. Chen, “CRC-Aided Decoding of Polar Codes,” IEEE Commun. Letters, vol. 16, no. 10, Oct. 2012