D3.3

Report on link-level simulation performance final results

Project number:	760150
Project acronym:	EPIC
Project title:	EPIC: Enabling Practical Wireless Tb/s Communications with Next Generation Channel Coding
Start date of the project:	1st September, 2017
Duration:	36 months
Programme:	H2020-ICT-2016-2

Deliverable type:	Report
Deliverable reference number:	ICT-760150 / D3.3 / 1.0
Work package contributing to the deliverable:	WP3
Due date:	Aug. 2020 – M36
Actual submission date:	1st September 2020

Responsible organisation:	TB
Editor:	Charbel Abdel Nour
Dissemination level:	PU
Revision:	V1.0

Abstract:	Report on link-level simulation performance final results and on communication performance comparison between the different families of FEC codes under study.
Keywords:	Turbo codes, LDPC codes, Polar codes, simulation results, comparisons.

The EPIC project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 760150.

Editor

Charbel Abdel Nour (TB)

Contributors (ordered according to beneficiary numbers)

Matthias Herrmann, Claus Kestel, Norbert Wehn (TUKL)

Meng Li (IMEC)

Erdal Arıkan, Ertuğrul Kolağasıoğlu, Altuğ Süral, Yiğit Ertuğrul (POL)

Charbel Abdel Nour, Rami Klaimi, Stefan Weithoffer (TB)

Onur Sahin (IDCC)

Disclaimer

Executive Summary

The current D3.3 reports final performance results on link-level simulations performed by using the selected algorithms and architectural parameters in D2.3 including any further refinements included for implementation and reported in D2.4. For Turbo codes, these include new protograph-based interleaver and local SOVA decoding algorithm. For LDPC codes, these include new block-based construction with fast-convergence and spatially coupled one. For Polar code, they are based on 2 new variants: one adapted for low complexity SC decoding and the other adapted to the more performant list decoding, both applying proposed multi-bit decision and adaptive quantization. The effect of different parameters, specific to each code family was evaluated first, then a scenario was set for comparing the 3 code families under varying channel conditions. Obtained results show that for any fair comparison between the code families the constraints of the target application should be clearly taken into account, especially regarding frame size and its flexibility, code rate and its flexibility, target throughput and coding gain and the transmission channel. These results can serve as a first step for performance comparison between the different families of FEC codes and when considered jointly with results of other work packages, the comparison can be extended to span hardware related aspects.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

The performed simulations in the context of WP3 provide input to WP2 and WP4 to refine code and decoder designs on the one side and help find best trade-offs between performance and complexity on the other side. Moreover, obtained results in WP3 represent a first step to perform a comparison between different Forward Error Correction (FEC) designs.

A first step, where a common link-level simulation template and a calibration plan were described and agreed upon between project partners, was provided in D3.1. Then, simulators were developed, followed by a calibration campaign using reference simulation scenarios provided for each code family. Afterwards, initial simulation results were obtained for some of the most promising techniques reported in D2.2 and identified after the design space exploration of D2.1. In addition to these results, the features supported by developed simulators were reported in D3.2, serving as a first feedback to WP2 and to WP4 for the refinement of these promising techniques.

A necessary period of a tight and continuous interaction between WP3, WP2 and WP4 was made. During this period, WP3 provided the necessary feedback for evaluating different code designs, for the impact of the different introduced simplifications to decoding algorithms and for the parameter selection procedure dedicated to hardware architectures. In the following, best designs relative to each code family were selected for implementation. The selection process is reported in D2.4.

D3.3 includes the results corresponding to the final selection of algorithms/architectures reported in D2.4. It also includes any potential developments/refinements introduced since for final implementation. Once made, this selection provided a complete design for each code family that served as a basis for performance comparison under varying channel conditions also reported in D3.3.

Chapter 2 Simulator description and impact of important parameters or additional features

This section is intended to provide all the necessary details regarding the final selection of the code design, decoding algorithm and architecture parameters to be able to simulate and provide corresponding error rate results. It is also intended to provide some insight related to the impact of the most important parameters per code family. It is divided into a sub-section per code family. Each sub-section is also organized in 3 parts:

- In the first part, a description regarding the choices and definitions for the code itself, decoding algorithm and the architecture refinements are provided. These are listed and briefly recalled where necessary. For more details, links are provided to the deliverables of WP2 and WP4.
- In the second part, results related to the impact of important parameters or additional features are provided. These features are code family dependent and therefore their number and corresponding results are specific to the considered code family. Such aspects may include quantization effects, number of iterations, assessment of different decoding architectures/algorithms, of different code structures, etc.
- In the 3rd and final part, a conclusion regarding achieved improvements of each code family with respect to previous state of the art is provided.

2.1 Final Turbo code proposal for EPIC: parameters and features

This section presents the final Turbo code [1] design proposed in WP2 and implemented in WP4. It includes new interleaver and puncturing patterns on the code design side, a new decoding algorithm on the decoding side and a new fully pipelined architecture on the hardware implementation side. In the following, we will briefly provide the parameters of the complete design. These are required to be able to provide the corresponding simulation results.

2.1.1 Turbo code parameters

The component code is the classical 8-state convolutional code with octal polynomials $(1, 15/13)_8$ used also as component code for the Long-Term Evolution (LTE) Turbo code. The interleaver is a protograph-based [2] Almost Regular Permutation (ARP) structure [3] derived from the regular interleaver structure with the introduction of a degree of disorder through a vector of shifts $\boldsymbol{S}$**.** The address i in the interleaved frame corresponds to the address $\Pi(i)$ in the natural-order frame following**:**

$$\Pi(i) = \left(P \times i + \boldsymbol{S}(i \bmod Q)\right) \bmod K$$

where P is the interleaver period that must be relatively prime to the information frame size K. The shift vector $\boldsymbol{S}$ has length Q and to guarantee the bijectivity of the ARP interleaver function, Q must be a divisor of K. The same regular M periodic puncturing was applied on the output parity of each component code. The turbo code was designed for a frame size of $K = 850$ bits and a target code rate of $R = 5/6$. The interleaver and puncturing parameters are as follows:

$Q = 10$, $P = 467$, $S(Q) = \{6, 631, 660, 458, 614, 602, 658, 766, 625, 320\}$

$M(Q) = \{0,1,0,0,0,0,0,0,0,0\}$ where a zero designates a punctured position.

2.1.2 Turbo decoder parameters

The proposed local Soft Output Viterbi Algorithm (SOVA) decoding algorithm with the additional simplifications detailed in D2.4 was used. Different window sizes were evaluated for the decoder. The decoding performance with different number of iterations was also assessed.

2.1.3 Hardware architecture-related parameters

The simulation results correspond to the fully pipelined unrolled architecture proposed in D2.3 [4] [5] and refined further in D2.4 [6] with the local SOVA [7] computation units. 6 bits input quantization are implemented. Radix-4 and radix-8 decoding units were used. Note that the choice of the radix order does not impact performance. A summary of the required parameters is provided in Table 1.

2.1.4 Simulation results

Table 1: Common simulation parameters for Turbo codes

Parameter	Value
channel	AWGN
mapping	Gray
modulation	BPSK
decoder_type	Local SOVA
max_frame_errors	Vary based on BER
max_frame_errors	50 @ lowest point
LLR quantization	6 bits
Window size	34/50/85
Max half iterations	8

Effect of window size

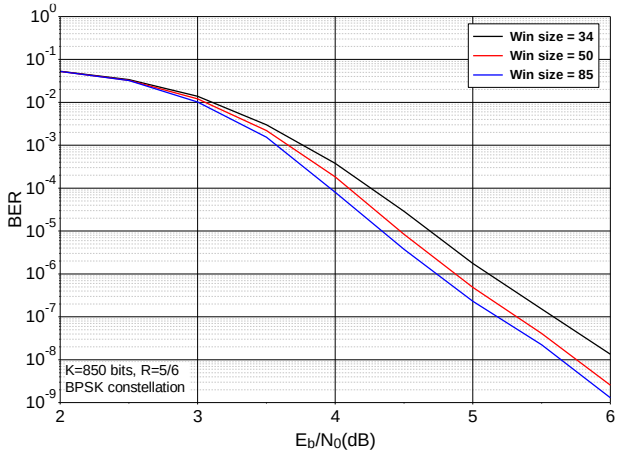

Figure 1: BER simulation results for different window sizes. AWGN channel, BPSK constellation, 850-bit information size, R=5/6 and 4 iterations.

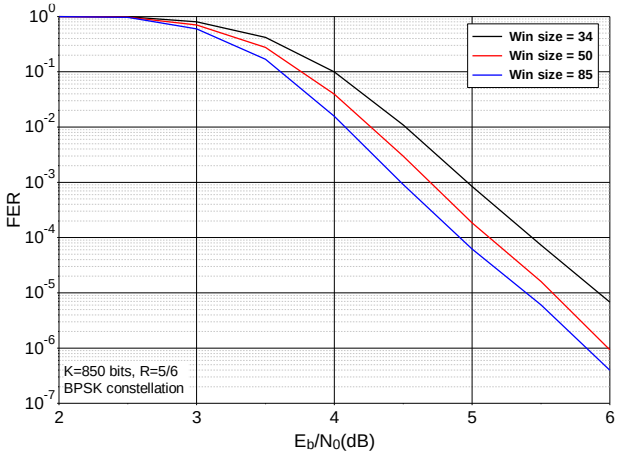

Figure 2: FER simulation results for different window sizes. AWGN channel, BPSK constellation, 850-bit information size, R=5/6 and 4 iterations.

Figure 1 shows a Bit Error Rate (BER) comparison between the performance results of the 850-bit frame size Turbo code for different window sizes, namely 34, 50 and 85 trellis sections long. A difference around 0.3 dB can be observed between window sizes 34 and 85. Note that most of the gain is achieved when moving from a window size of 34 to 50.

This difference is larger in the Frame Error Rate (FER) curves of Figure 2, suggesting the appearance of a small amount of residual errors in each window when these get smaller. The gaps remain unchanged when simulating over fast fading Rayleigh channels.

Effect of the number of iterations

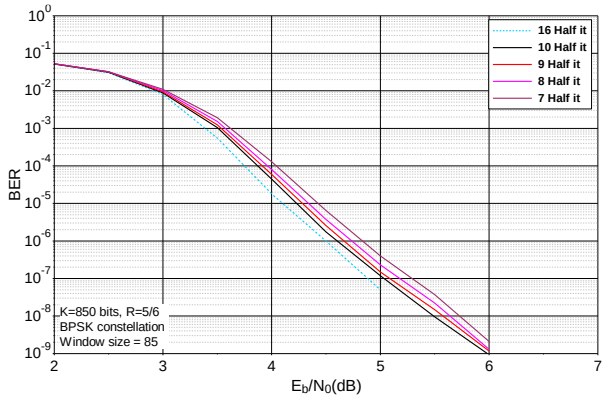

Figure 3: BER simulation results for different number of iterations. AWGN channel, BPSK constellation, 850-bit information size, R=5/6 and window size 85.

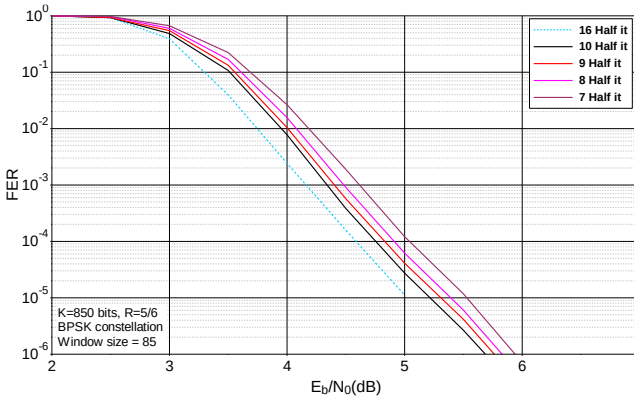

Figure 4: FER simulation results for different number of iterations. AWGN channel, BPSK constellation, 850-bit information size, R=5/6 and window size 85.

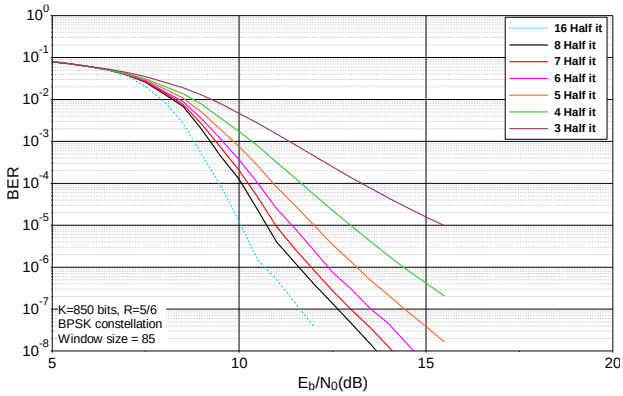

Figure 5: BER simulation results for different number of iterations. Rayleigh channel, BPSK constellation, 850-bit information size, R=5/6 and window size 85.

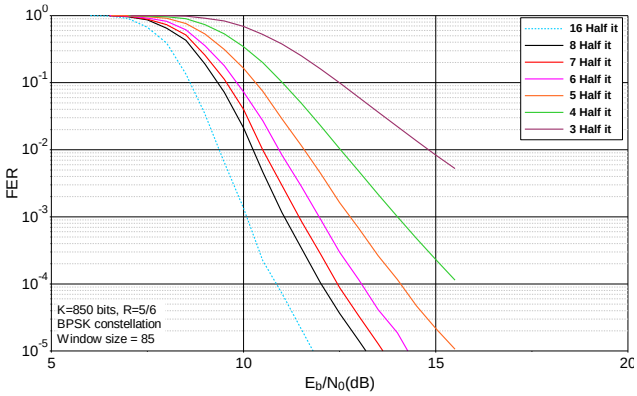

Figure 6: FER simulation results for different number of iterations. Rayleigh channel, BPSK constellation, 850-bit information size, R=5/6 and window size 85.

Figure 3 and Figure 4 show the BER and FER simulation results over an AWGN channel for different number of iterations for the same set of parameters of the Turbo code. At most, a gain of around 0.5 dB can be achieved when moving from 7 to 16 half iterations. However, this gain spans 3 dB over a Rayleigh fading channel as we can see from Figure 5 and Figure 6. We can then conclude that the effect of number of iterations is much larger over fading channels.

2.1.5 Conclusion

The different parameters of the final design of the Turbo code within EPIC are described in this section. It applies the 3 main proposals related to Turbo codes: the new protograph ARP-based interleaver and puncturing patterns, the new local SOVA decoding algorithm and the new fully pipelined architecture. The adoption of these proposals enabled:

- to significantly lower error floors, especially for high rates and/or short frame sizes,
- to introduce regularity into the interleaver, largely simplifying parallel architectures,
- to support frame size flexibility for pipelined architectures in addition to the inherent code rate flexibility,
- to reduce by around 50% the complexity of the computation units required for decoding,
- to improve by more than an order of magnitude (from a few Gb/s to more than 400 Gb/s) the achieved throughput at a reasonable hardware cost.

Assessment of high impact parameters such as window size and number of iterations were performed. Their impact is not limited to performance. Complexity is a deciding factor in the final choices. These latter were dictated based on simulation parameters and/or to achieve the best compromise between performance and complexity for the targeted throughput level in EPIC.

2.2 Final LDPC code proposal for EPIC: parameters and features

This section presents the final selection of the LDPC and SC-LDPC codes and simulation results of the implemented SC-LDPC and LDPC (block) decoders.

2.2.1 Parameters of the considered LDPC block codes

Two LDPC block codes were designed in WP2 and eventually considered for implementation. For a detailed description on the design of the EPIC codes we refer to D2.2. The lifted H-matrices of the codes are specified as follows:

LDPC code 1: 3×13, $Z = 79$, $N = 1027$

```
Hbc1 = [
34  9 23 39  3 28 60 25 66 26 27 -1 -1;
25 34  1 31  6  9  5 14  4 10 29 20 -1;
67 12 74  8 45 64 59 16 76 60 33 12 34]
```

LDPC code 2: 4×24, $Z = 43$, $N = 1032$

```
Hbc2 = [
27 22 31 12 40 35 17 31 11 15 28 33 26 26 42  8 21  1 41 16 22 -1 -1 -1;
38 12 41 10  6 10 21 26 11 16 14  4 10 20 21  5  6 18 10 40 41  3 -1 -1;
36 15 42  7 22 23 31  0 16  7 28  0 42  1  3 28 15 33 39 23 28 14 36 -1;
 0 30 27 25  0  3 21 32  9  7 42 39 11  4  1  1 16 27 22 28  9 25  1 17]
```

2.2.2 Parameters of the considered SC-LDPC code

The EPIC SC-LDPC code was designed with particular focus on a simplified encoding and decoding procedure. The basic structure of the H-matrix H_{SC} is depicted in **Fehler! Verweisquelle konnte nicht gefunden werden.**. The sub-matrices H_0 and H_1 were each constructed from an 8×40 protograph lifted using progressive edge growth with $\mathrm{Z} = 16$. The lifted matrices H_0 and H_1 are:

```
H0 = [
12 -1 -1 -1 -1 -1 -1  7 11 -1 -1 -1  3 -1 -1 -1 14 -1 -1 -1 -1 -1  6 -1 -1 12  3 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1;
-1  2 -1 -1 -1 15 -1 -1 -1 -1  9 -1 -1 -1 -1  3 -1 -1 10 -1 -1 12 -1 -1 -1 -1 -1  5 -1 -1 -1  2 -1  0 -1 -1 -1 -1 -1 -1;
-1 -1 -1  3  8 -1 -1 -1 -1  2 -1  7 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1  7  4 -1 -1  9 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1;
-1  1 -1 -1 -1 -1 -1 13 -1 -1 -1 -1 -1  2  6 -1  5 -1 -1 -1  7 -1 -1 -1  1 -1 -1 -1 -1 14 -1 -1 -1 -1 -1 -1  0 -1 -1 -1;
-1 -1 -1 -1 -1  4  5 -1 -1 -1 -1  4  4 -1 -1 -1 -1  4 -1 -1 -1  5 -1 -1 -1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1;
-1 -1 -1  9  9 -1 -1 -1  1 -1 -1 -1  3 -1 -1 -1 -1 15  5 -1 -1 -1 -1 -1 -1 -1 -1 -1  8 -1  5 -1 -1 -1 -1 -1 -1  0 -1 -1;
13 -1 15 -1 -1 -1 -1 -1 -1 11 -1 -1 -1 -1  5 -1 -1 -1 -1 -1 -1 -1 11 13 -1 -1 -1 -1  7 -1  6 -1 -1 -1 -1 -1 -1 -1  0 -1;
-1 -1  0 -1 -1 -1 15 -1 -1 -1  2 -1 -1 -1 -1 11 -1 -1 -1 -1 15  9 -1 -1 -1 -1 -1 -1 -1  8 -1  8 -1 -1 -1 -1 -1 -1 -1 -1  0]

H1 = [
-1 -1 -1 -1 15 -1  4 -1 -1 -1 -1 -1 10 10 -1 -1 -1 -1 10  5 -1 -1 -1 -1 -1 -1 -1 14 -1 -1 -1  4 -1  3 -1 -1 -1 10 -1  9;
-1 -1 -1 -1  6 -1  5 -1 -1 -1 -1 -1 -1 -1 15 10 -1 10 -1 -1 -1  1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1  7 -1  8  9 -1 -1 -1 -1  1;
 3 -1 -1 -1 -1 -1 -1 -1  4 -1 -1  0 -1 -1 -1 -1  4 -1  6 -1 -1 -1 -1 -1  9 -1 -1 10 12 -1 -1 -1 -1 -1 -1  2 -1 -1  2 12 -1 -1 -1;
-1 -1 -1 -1 -1 -1  7 -1  7 -1 -1 -1 -1  4 -1 -1  0 -1 -1 13 -1 -1 -1 -1 10 -1 -1  9 -1 -1  2 -1 -1  3 -1 -1 -1 -1  5  0 -1;
12 -1 -1  2 -1 -1 -1 -1  6 -1 -1  0 -1 -1 -1 -1 12 -1 -1 -1 -1 -1 -1  2 -1 10 -1 -1  4 -1 -1 -1 -1 -1 12  2  4 -1 -1 -1 -1 -1;
-1 -1  8 15 -1 -1 -1 -1 -1 -1  9 -1 10 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  4 -1 -1 15 -1 -1  1 -1 -1 -1 15 -1 -1 -1 -1 -1 -1 -1  4  3 14;
-1  2 -1 -1 -1 12 -1 -1 -1 -1  4 -1 -1  7 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 12 -1 -1 -1 -1 -1 -1  4 -1  8 -1 -1 -1 -1 -1  9 15 -1  6 -1;
-1 11 11 -1 -1 -1 -1 -1 -1  5  5 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  3 10 -1 -1 -1 -1 -1 -1 -1 -1 -1 15 -1  1 -1 -1 -1  6  5  3 -1 -1 -1]
```

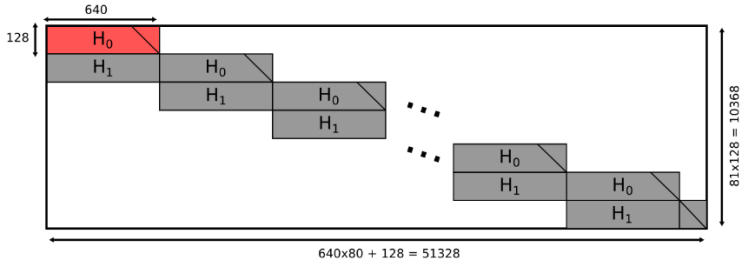

Figure 7: H-matrix structure of the EPIC SC-LDPC code

The rightmost part of H_0 is an 8×8 identity matrix to enable simplified systematic encoding [8]. The resulting sub-block size is $c = 640$ and the number of parity check equations for each sub-matrix is $(c - b) = 128$. A full code block is composed of 80 sub-blocks plus 128 tail-bits and has, thus, length $N = 51328$. The code rate is $R \approx 0.8$.

2.2.3 Simulation results

Partial-parallel Frame Interleaved Layered Decoder

In this subsection, we present the performance of the partially parallel frame interleaved block decoders for the EPIC LDPC codes that has been designed and implemented in WP2 and WP4.

Table 2: Common simulation parameters for LDPC codes

Parameter	Value
channel	AWGN
mapping	Gray
modulation	BPSK
decoder_type	Offset min-sum
max_frame_errors	Vary based on BER
max_frame_errors	20 @ BER 10^{-6}
LLR quantization	4/5 bits
Max iterations	2/3/4/5

Two LDPC codes with a coded size *N* around 1000 bits and rate 5/6 were designed in WP2. BER versus Eb/No performance and FER versus Eb/No performance of the EPIC LDPC code with = 1027 bits are shown in Figure 8 and Figure 9, respectively. BER versus Eb/No performance and FER versus Eb/No performance of the EPIC LDPC code with *N*=1032 bits are shown in Figure 10 and Figure 11, respectively. For the LDPC (1027,856) code, at FER @ 10^{-5}, the 5 bit quantization has 0.1 dB and 0.15 dB performance degradation compared with floating point at maximum 3 and maximum 4 iterations, respectively. The 4 bit quantization has 0.4 dB and 0.5 dB performance degradation compared with floating point at maximum 3 and maximum 4 iterations, respectively. For the LDPC (1032,860) code, at FER @ 10^{-5}, the 5 bit quantization variant has 0.05 dB and almost no performance degradation compared with floating point at maximum 2 and maximum 3 iterations, respectively. The 4 bit quantization has 0.1 dB and almost no performance degradation compared with floating point at maximum 2 and maximum 3 iterations, respectively.

For each figure shown below, both the floating-point simulation and 4 bits / 5 bits fixed-point simulation are carried out with different maximum number of iterations. The 5 bits quantization has limited performance degradation when compared with floating-point performance, however, 4 bits quantization has around 1 dB performance degradation at a BER of 10^{-6} when compared with floating-point performance. The 5 bits quantization variant has limited performance degradation when compared with floating-point performance, however, 4 bits quantization has around 0.5 dB performance degradation at BER a of 10^{-6} when compared with floating-point performance.

The LDPC code (1027,856) has only 3 layers in the parity check matrix and the LDPC code (1032,860) has 4 layers in the parity check matrix. The throughput of the partially parallel frame interleaved decoder scales as the inverse of the number of layers, hence the throughput of LDPC code (1027,856) is 1.3 times the one of LDPC code (1032,860). However, in terms of BER & FER performance, LDPC code (1032,860) outperforms the LDPC code (1027,856). For fixed point quantization, the decoder for codes (1032,860) with 2 maximum iterations has similar performance as the decoder for codes (1027,856). Hence, in WP4, an 8-core architecture for code (1032,860) with maximum 2 iterations is better than a 12-core architecture for code (1027,856) with maximum 4 iterations.

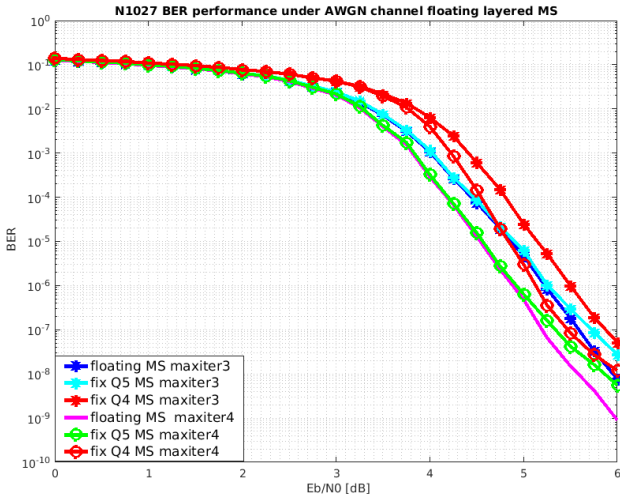

Figure 8: BER vs. Eb/No (dB) performance of LDPC (1027,856) with BPSK modulation under AWGN channel.

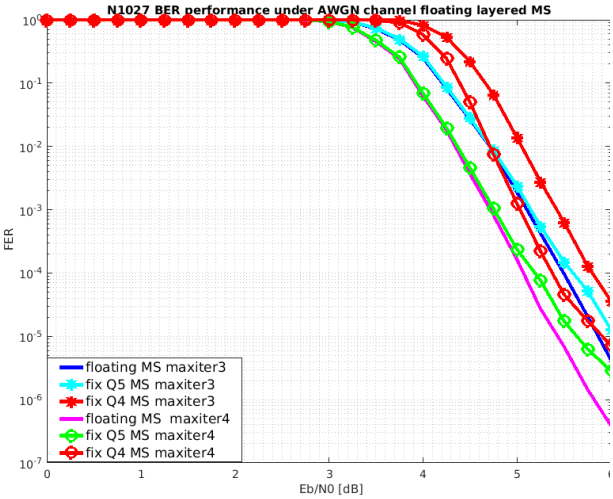

Figure 9: FER vs. Eb/No (dB) performance of LDPC (1027,856) with BPSK modulation under AWGN channel.

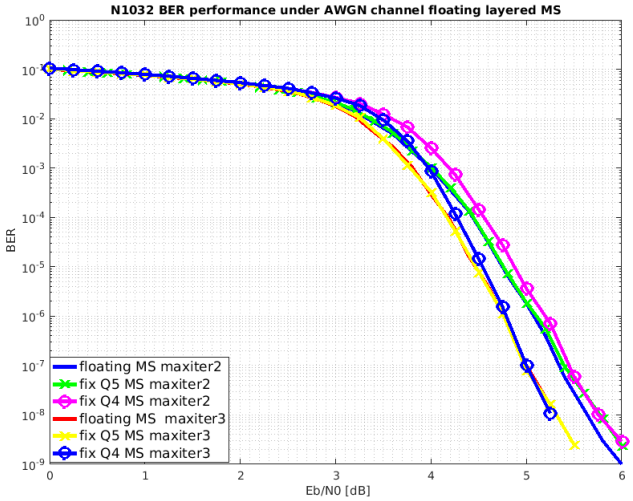

Figure 10 : BER vs. Eb/No (dB) performance LDPC (1032,860) with BPSK modulation under AWGN channel.

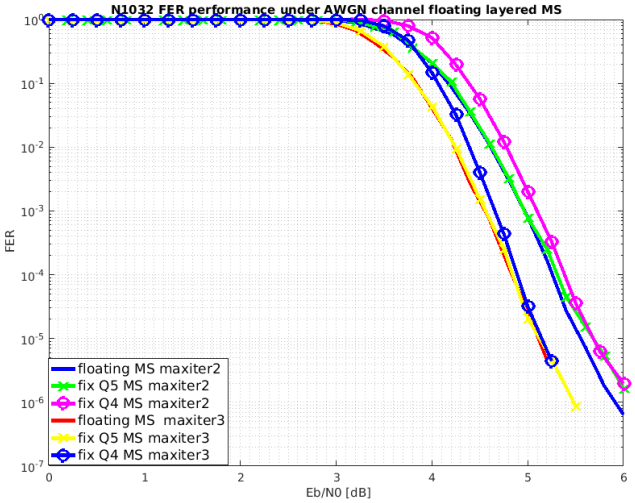

Figure 11: FER vs. Eb/No (dB) performance of LDPC (1032,860) with BPSK modulation under AWGN channel.

Unrolled Block Decoders

In this section, we present the error correction performance of the EPIC unrolled block decoders (see D2.4):

- The Unrolled Two-Phase (TP) Decoder
- The Unrolled Layered Decoder

Error correction performance is assessed with FER over SNR simulations. Each SNR point was simulated with 10^7 blocks transmitted over an AWGN channel using BPSK modulation. Figure 12 shows the performance of layered decoding and two-phase decoding for the (1032, 860) EPIC code for various iterations. The decoding is performed with the Normalized Min-Sum (NMS) algorithm with extrinsic scaling factor $\gamma = 0.75$. It can be observed that two-phase decoding requires about twice the number of iterations to achieve the performance as layered decoding. Figure 13 shows the performance of the implemented unrolled decoders (see D4.3). Both decoders use 4bit for the quantization of channel values and extrinsic messages. Up to a FER of 10^{-5} the loss due to quantization is less than 0.25dB.

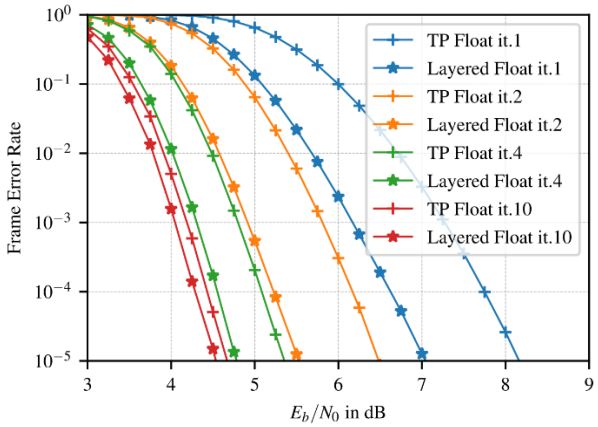

Figure 12 : Performance comparison of layered and two-phase scheduling for the (1032, 860) EPIC LDPC code (float), various iterations of NMS.

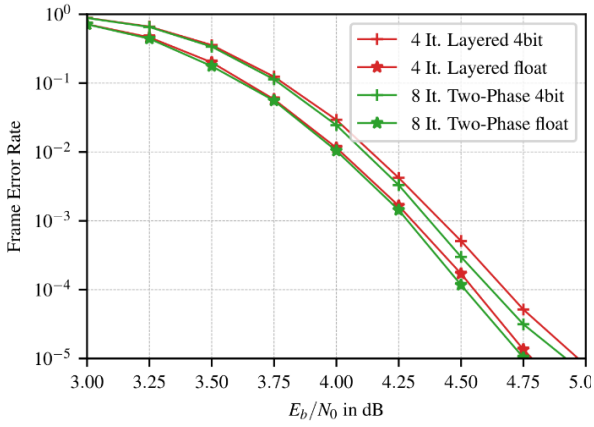

Figure 13 : Performance comparison of 4-iteration layered and 8-iteration two-phase decoders, float vs. 4bit fixed point.

SC-LDPC Window Decoders

In this section, we present the error correction performance of the SC-LDPC window decoders initially described in D2.4:

- The Layered Window Decoder
- The Unrolled Window Decoder
- The Unrolled Window Decoder with pipeline stage

Error correction performance is assessed with BER over SNR simulations. Each SNR point was simulated with 10^6 blocks (corresponding to $80 \cdot 10^6$ sub-blocks) transmitted over an AWGN channel with BPSK modulation. Figure 14 shows the floating point performance of the Layered Window Decoder and the Unrolled Window Decoder for different iterations. The decoders use the NMS algorithm with extrinsic scaling factor $\gamma = 0.75$ and $\gamma = 0.875$ respectively. Figure 15 shows the impact of the additional pipeline stage in the Unrolled Window Decoder (see description in D2.4). The performances of the implemented SC-LDPC decoders (see D4.3) are compared to each other and to the respective floating point performance in Figure 16. Finally, Figure 17 compares the performances of the SC-LDPC decoders with the unrolled LDPC block decoders.

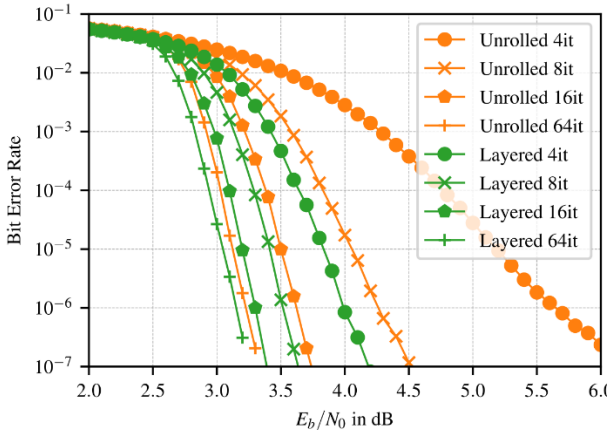

Figure 14 : Performance of the Unrolled Window Decoder and the Layered Window Decoder for various number of iterations.

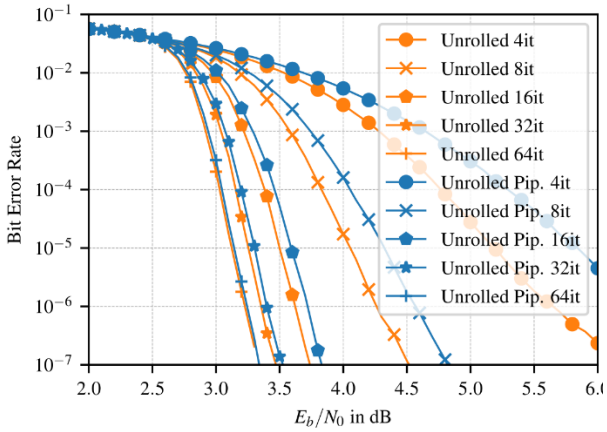

Figure 15 : Performance comparison of the Unrolled Window Decoder and the Unrolled Window Decoder with additional pipeline stage.

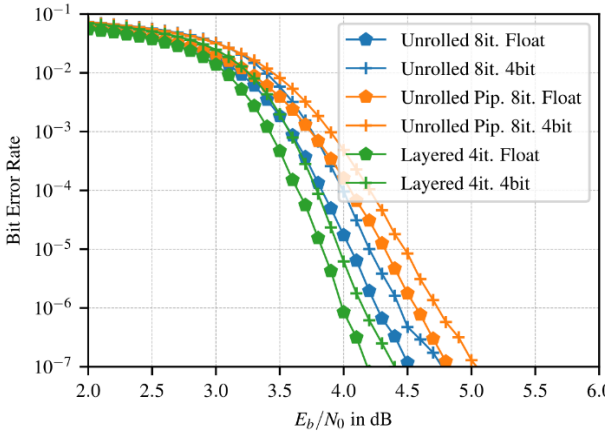

Figure 16 : Performance comparison of 4-iteration Layered Window Decoder, 8-iteration Unrolled Window Decoder and 8-iteration Unrolled Window Decoder with additional pipeline stage.

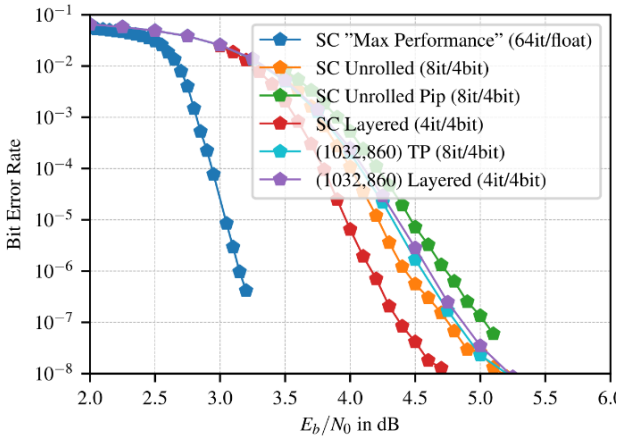

Figure 17 : Performance comparison of the SC-LDPC decoders and the unrolled LDPC block decoders.

2.2.4 Conclusion

Decoders for LDPC and SC-LDPC codes are developed and optimized for highest throughput in WP2 and implemented in WP4. The respective simulation results are presented in this deliverable. For the EPIC LDPC block codes (length ~1000 bit), 8 iterations of two-phase and 4 iterations of layered decoding provide a good trade-off between error correction performance and computational complexity. The EPIC SC-LDPC code (length ~50k bit) requires more iterations, e.g. 8 iterations for the Layered Window Decoder and 16 iterations for the Unrolled Window Decoder to perform close to the maximum performance (at 64 iterations). However, already 4 and 8 iterations provide a gain over the LDPC block decoders (0.4dB for the layered window decoder and 0.1dB for the unrolled window decoder).

2.3 Final Polar code proposal for EPIC: parameters and features

This section presents the two polar code designs developed in WP2 and implemented in WP4. The first design is a (1024,854) polar code with a low-complexity Successive Cancellation (SC) decoding algorithm [9] and described in Section 2.3.1. The second design is a (1024,888) polar code with SC List (SCL), list size two, decoding algorithm [10] and described in Section 2.3.2. The first design has been shown to achieve 1 Tb/s throughput in WP4. The second design achieves better performance compared to the first but is more complex. Common features of the two polar code designs are listed below.

- BER performance is improved with systematic polar code [11].
- Encoding and decoding complexity is minimized with binary 2x2 kernel.
- BER/FER optimization is achieved with density-evolution based polar code design [12].
- Hardware complexity is reduced with min-sum decoding.
- Decoding of polar code segments are simplified by multi-bit parallel SC decoding with shortcuts [13].

It is worth mentioning that low-complexity polar encoder utilizes only a series of XOR operations between bit channels. In addition, bit reversed ordering of LLR values are used.

2.3.1 Polar Code Design with SC Decoding

The first code design is polar (1024,854) with SC decoding algorithm. Polar code with block length equals to 1024 bits and payload equals to 854 bits. Code rate equals to 5/6. Density evolution code construction method is used to determine the set of free bit channels at 6.5 dB SNR. Among 1024-bit channels, most reliable 854 of them are identified as free bit channels which convey information.

Remaining 170-bit channels are named as frozen bit channels which do not carry any information and are set to a predetermined value. The set of frozen bit channels are listed below.

[1,2,3,4,5,6,7,9,10,11,13,17,18,19,21,25,33,34,35,37,41,45,49,53,57,65,66,67,69,73,77,81,83,85,89,97,99,101,105,113,129,130,131,133,137,141,145,147,149,153,161,163,165,169,177,193,194,195,197,201,209,225,241,257,258,259,261,265,267,269,273,275,277,281,289,290,291,293,297,305,321,322,323,325,329,337,353,361,369,385,386,387,389,393,401,417,425,433,449,453,457,465,481,513,514,515,517,521,523,525,529,530,531,533,537,545,546,547,549,553,561,577,578,579,581,585,593,609,617,625,641,642,643,645,649,657,665,673,681,689,705,709,713,721,737,769,770,771,773,777,785,793,801,805,809,817,833,837,841,849,865,897,899,901,905,913,929,961,977,993]

SC decoding algorithm is a depth-first search algorithm with $O(NlogN)$ asymptotic complexity where N denotes the block length. Decoding of the first polar code requires 2046 time-steps. Multi-bit decision functions are used to decrease the required time-steps to 150. Complexity of the multi-bit decision functions are limited with segment size 64. 5 bits LLR precision is used at the decoder input to limit the communications performance loss. Average LLR precision of the decoder is reduced by 15.1% using adaptive quantization technique [14]. Bit precision for each code segment is marked in Figure 18. For example, node between code segment (512,493) and (256,255) is marked as three, meaning 3-bit quantization is enough to decode (256,255) code segment under negligible communications performance loss.

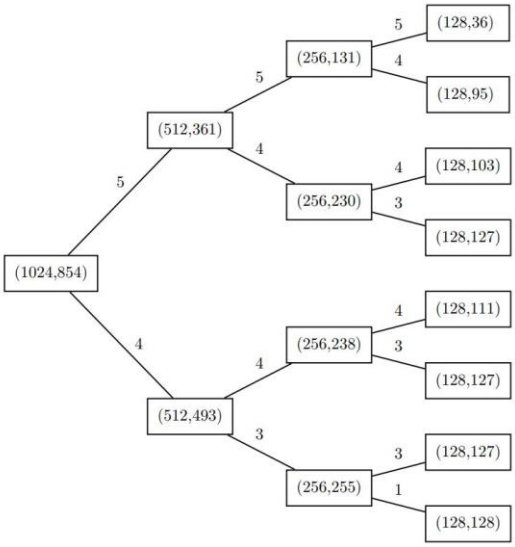

Figure 18: Bit precision of each code segment under polar (1024, 854) code.

2.3.2 Polar Code Design with SCL Decoding

Another code that has been studied is polar (1024, 888) under SC with a list of two decoding. Block length equals to 1024 bits and payload equals to 888 bits. Code rate equals to 13/15. 4 Cyclic Redundancy Check (CRC) bits are appended to 888 information bits. Density evolution code construction method is used to determine the set of free bit channels at 8 dB SNR. Among 1024-bit channels, most reliable 892 (888+4) of them are identified as free bit channels. Remaining 132-bit channels are named as frozen bit channels which do not carry any information and are set to a predetermined value. The set of frozen bit channels are listed below.

[1,2,3,4,5,6,7,9,10,11,13,17,18,19,21,25,33,34,35,37,41,49,57,65,66,67,69,73,81,85,89,97,101,105,113,129,130,131,133,137,145,149,153,161,163,165,169,177,193,195,197,201,209,225,257,258,259,261,265,269,273,277,281,289,291,293,297,305,321,323,325,329,337,353,385,386,387,389,393,401,417,449,465,481,513,514,515,517,521,525,529,531,533,537,545,547,549,553,561,577,578,579,581,585,593,609,641,642,643,645,649,657,673,705,721,737,769,770,771,773,777,785,801,817,833,849,865,897,905,913,929,961]

SCL decoding algorithm has asymptotic complexity $O(LNlogN)$ where N denotes the block length and L denotes the list size. It employs M-algorithm a breadth-first sorting algorithm such that 2L paths are compared at any depth [15]. Each path has an associated metric which could be thought of the history of that path. CRC bits are used to select among L candidate codewords. CRC polynomial $g(x) = 1 + x + x^2 + x^3 + x^4$ is determined by software simulations. CRC bits are placed into the most reliable bit channels among 892 free bit channels.

A special sorter for list size two is used in this decoder where at any depth we deal with an almost sorted 4 paths. Optimization of the sorter diminishes the complexity of the decoder significantly since it's used multiple times throughout the decoding.

Multi-bit decision functions are used to decrease the required time steps for decoding from 2046 to 181. Pipeline depth of the decoder is reduced to 78 from 181 by using register reduction/balancing technique. 5 bits LLR precision is used to limit the communications performance loss.

2.3.3 Simulation results

This subsection presents simulation studies related to the EPIC polar codes that has designed and implemented in WP2 and WP4. Software simulations have been carried out to verify the communications performance of codes. Software simulator capabilities regarding polar codes are described in the previous deliverable. Software simulation results are shared according to agreed JSON format. Common parameters for each simulation are shown in Table 3.

Table 3: Common simulation parameters for polar codes

Parameter	Value
channel	AWGN
mapping	Gray
modulation	BPSK
seed_rnd	2
max_frame_errors	300
max_trials	1000000
target_FER	10^{-5}
rate_matching	None
code_design_type	Density Evolution
decoder_type	min-sum
systematic	yes

BER versus Eb/No performance of the first polar code design is shown in Figure 19 where BPSK modulation is used under AWGN channel. Dashed purple curve is the theoretical BER performance of uncoded transmission with BPSK modulation. Blue curve is the software simulation of the min-sum SC decoder with floating point representation of the data. Brown curve is the FPGA implementation of min-sum SC decoder with 5 bits LLR precision throughout the decoder. Yellow curve is the FPGA implementation of the developed polar code which has 6.21 dB coding gain at 10^{-12} BER compared to uncoded transmission. FER versus Eb/No performance of the first polar code design is shown in Figure 20 where BPSK modulation is used under AWGN channel.

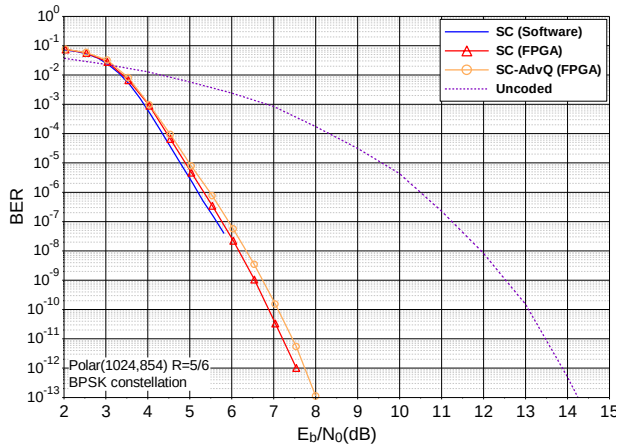

Figure 19: BER vs. Eb/No (dB) performance of Polar (1024,854) with BPSK modulation under AWGN channel.

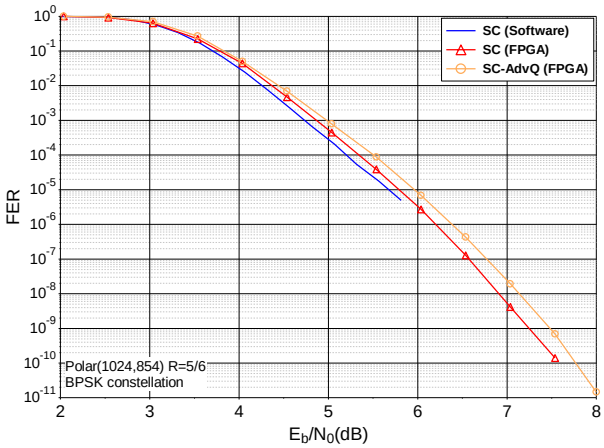

Figure 20: FER vs. Eb/No (dB) performance of Polar (1024,854) with BPSK modulation under AWGN channel.

BER versus Eb/No performance of the polar (1024,888) is shown in Figure 21 where BPSK modulation is used under AWGN channel. Dashed yellow curve is the theoretical BER performance of uncoded transmission with BPSK modulation. Blue curve is the software performance of the min-sum SC List2 decoder with floating point representation of the data. Brown curve is the min-sum SC List2 decoder with 5 bits LLR precision throughout the decoder. Simulations show that the second polar code design has 7 dB coding gain at 10^{-12} BER compared to uncoded transmission. FER versus Eb/No performance of the second polar code design is shown in Figure 22 where BPSK modulation is used under AWGN channel.

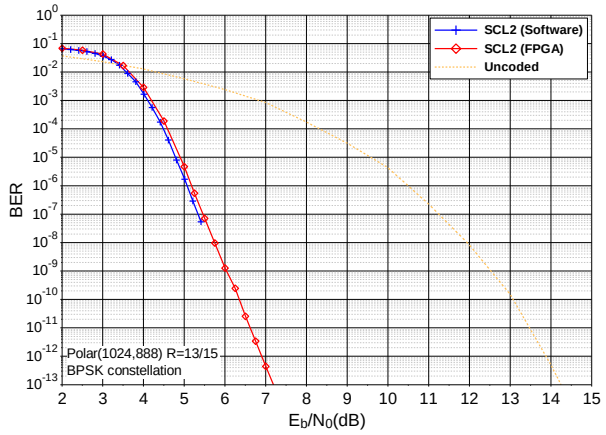

Figure 21: BER vs. Eb/No (dB) performance of Polar (1024,888) with BPSK modulation under AWGN channel.

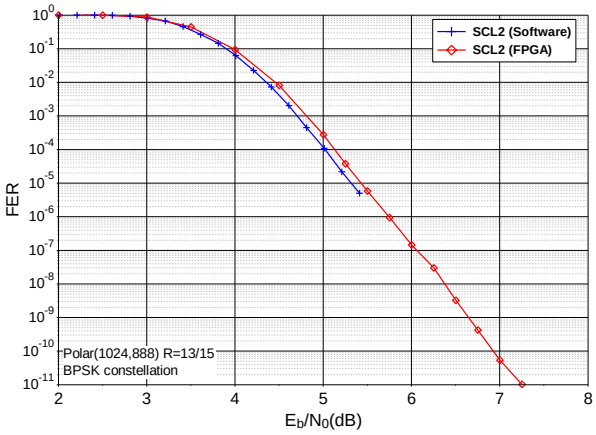

Figure 22: FER vs. Eb/No (dB) performance of Polar (1024,888) with BPSK modulation under AWGN channel.

2.3.4 Conclusion

Two polar code designs with decoding algorithm and architecture pairs are developed in WP2 and implemented in WP4. In both designs, systematic polar coding and density evolution code construction method is preferred to improve BER performance. Multi-bit decision functions are used to reduce the required time steps for decoding. 1 Tb/s throughput is achieved with the first polar code design under low-complexity SC decoding algorithm. A more complex decoder is used in the second design to provide better communications performance.

Chapter 3 Comparison of simulation results

Intended to provide an insight on the performance comparison of the designed EPIC codes, this section starts by defining agreed upon simulation conditions.

3.1 Simulation parameters

With the throughput levels targeted in EPIC and taking into account some of the target applications studied in WP1, simplified decoding algorithms were used, introducing a considerable impact on the achieved coding gain. Taking this fact into account, a way to harmonize results from the different code families had to be proposed. Since SC decoding was used for Polar codes, the corresponding error rate results were used as a target benchmark to be matched by the remaining code families through adapting their design parameters. Then the set of Turbo and LDPC code parameters matching Polar SC decoding performance would be used as input for WP4 to provide an insight into the comparison of the complexity of the decoders.

With the input provided from WP4 considering implementation constraints, we have clearly identified that the adopted fully pipelined architectures for the code families have a considerable impact on the supported frame size and code rate flexibility. Consequently, dictated by implementation constraints, comparison parameters have been set to a coded frame size of around 1024 bits with a coding rate of R=5/6.

3.2 Simulation results

3.2.1 Over AWGN channel

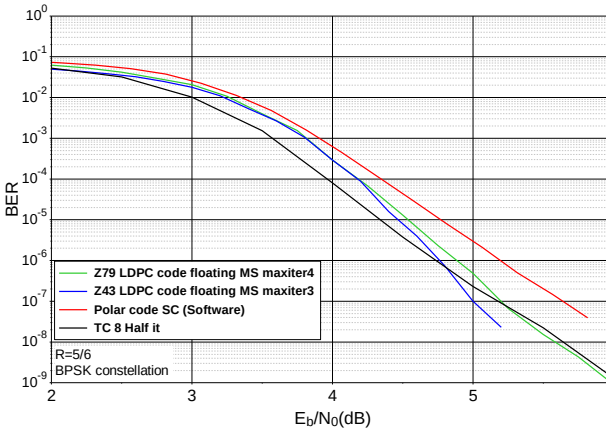

Figure 23: BER vs. Eb/No (dB) performance comparison of the considered codes with BPSK modulation under AWGN channel and floating-point precision for Polar and LDPC codes.

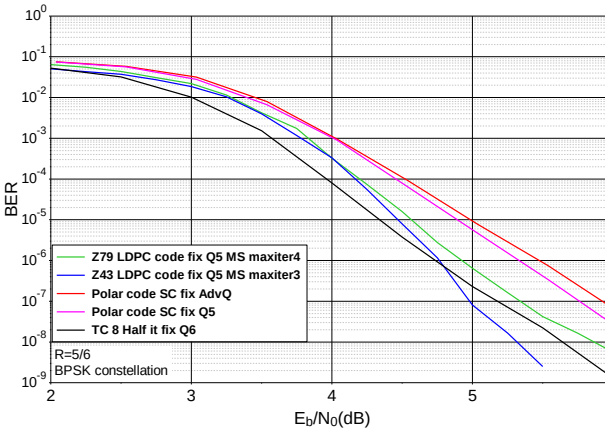

Figure 24: BER vs. Eb/No (dB) performance comparison of the considered codes with BPSK modulation under AWGN channel and fixed-point precision.

Figure 23 and Figure 24 show performance comparison of the considered codes over an AWGN channel with floating and fixed-point precision respectively. We can see that the Turbo code matches or slightly outperforms Polar code's performance for 8 Half iterations (4 iterations). The same number of layered iterations is required for the Z79 LDPC code while only 3 layered iterations are required for the Z43 LDPC code in the floating point case. A slight penalty is observed for the quantized case as expected, depending on the number of quantization bits. With 6 bits precision, the Turbo decoder does not suffer from any penalty. LDPC decoders with 5 bits quantization only suffer from a slight penalty. The quantized performance of the Polar code depends on the quantization method used.

3.2.2 Over Rayleigh fading

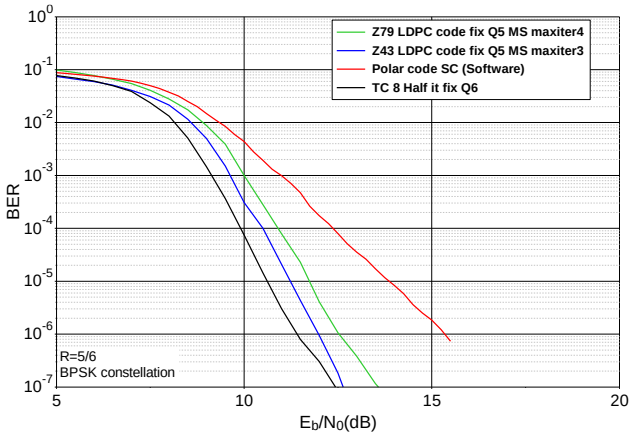

Figure 25: BER vs. Eb/No (dB) performance comparison of the considered codes with BPSK modulation under Rayleigh fading channel and floating-point precision.

Figure 25 shows the simulation results for the same set of parameters, especially in number of iterations over a Rayleigh fading channel where the channel gain is independent and identically distributed from a channel use to the next. This is equivalent to having infinite interleaving depth. We can conclude that the codes do not provide the same level of robustness with varying channel conditions.

Chapter 4 Summary and Conclusion

This deliverable is intended to provide final simulation results for the different code families including the proposals at the level of the code design, decoding algorithms and the impact of implementation. It can also be used to provide a glimpse at a comparison in performance between the different code families.

For the different proposals targeting Turbo codes, the effect of the window size and number of iterations were investigated. For LDPC codes, several designs were evaluated with different levels of trade-offs between performance and complexity. These include block-based LDPC codes and spatially coupled ones. The effect of number of iterations was also studied for these structures. For Polar codes, two designs with decoding algorithm and architecture pairs were investigated. The first is SC-based and targets high throughput while the second is SCL-based and is directed towards better performance and coding gain.

A way to compare codes was set, aligned to SC Polar performance for a target case of coded frame size of around N=1024 and R=5/6 dictated by implementation constraints. The sets of parameters of Turbo and LDPC codes matching this performance were identified over an AWGN channel. Additional simulations were provided for these sets of parameters over a Rayleigh fading channel. These seem to indicate that all codes do not provide the same robustness to varying channel conditions.

From all these results, we can conclude that for any fair comparison between the code families the constraints of the target application should be clearly taken into account, especially regarding frame size and its flexibility, code rate and its flexibility, target throughput and coding gain and transmission channel. Indeed, the comparison conclusions are expected to vary greatly with these parameters. Moreover, for each code family, there are often several sets of parameters that lead to similar BER/FER performance. Therefore, obtained performance should always be considered jointly with the set of parameters and their impact on complexity and throughput through a cross referencing with obtained results from WP4 and provided namely in D4.3.

Chapter 5 List of Abbreviations

Abbreviation	Translation
ARP	Almost Regular Permutation
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
CRC	Cyclic Redundancy Check
FEC	Forward Error Correction
FER	Frame Error Rate
FPGA	Field Programmable Gate Array
LDPC	Low Density Parity Check
LLR	Log Likelihood Ratio
LTE	Long Term Evolution
NMS	Normalized Min Sum
SC	Successive Cancellation
SC-LDPC	Spatially Coupled LDPC
SCL	Successive Cancellation List
SNR	Signal to Noise Ratio
SOVA	Soft Output Viterbi Algorithm
TP	Two-Phase

Chapter 6 Bibliography

[1] C. Berrou, A. Glavieux and P. Thitimajshima, “Near Shannon limit error-correcting coding and decoding: Turbo-codes,” in *IEEE International Conference on Communications, (ICC’93)*, Geneva, Switzerland, May 1993.

[2] R. Garzón-Bohórquez, C. Abdel Nour and C. Douillard, “Protograph-based interleavers for punctured turbo codes,” *IEEE Transactions on Communications,* vol. 66, no. 5, pp. 1833-1844, 2018.

[3] R. Garzón-Bohórquez, C. Abdel Nour and C. Douillard, “Improving Turbo Codes for 5G with parity puncture-constrained interleavers,” in *9th Int. Symp. on Turbo Codes and Iterative Information Processing (ISTC)*, Brest, France, Sep. 2016.

[4] S. Weithoffer, C. Abdel Nour, N. Wehn, C. Douillard and C. Berrou, “25 Years of Turbo Codes: From Mb/s to beyond 100 Gb/s,” in *IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC)*, Hong Kong, China, Dec. 2018.

[5] S. Weithoffer, O. Griebel, R. Klaimi, C. Abdel Nour and N. Wehn, “Advanced Hardware Architectures for Turbo Code Decoding Beyond 100 Gb/s,” in *Submitted to IEEE Wireless Commun. and Networking Conf. (WCNC), Available online at https://hal.archives-ouvertes.fr/hal-02319732*, Seoul, Korea, April 2020.

[6] S. Weithoffer, R. Klaimi, C. Abdel Nour, W. Wehn and C. Douillard, “Fully Pipelined Iteration Unrolled Decoders - The Road to Tb/s Turbo Decoding,” in *Submitted to the 45th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Barcelona, Spain, May 2020.

[7] V. H. Son Le, C. Abdel Nour, E. Boutillon and D. C., “Revisiting the Max-Log-Map algorithm with SOVA updates rules: new simplifications for high-radix SISO decoders,” *Under revision for publication in IEEE Transactions on Communications available online at https://hal.archives-ouvertes.fr/hal-02332503,* 2019.

[8] J. Chen, A. Dholakia, E. Eleftheriou and M. P. C. Fossorier, “Reduced-complexity decoding of LDPC codes,” *IEEE Transactions on Communications, 53, 8,* pp. 1288-1299, 2005.

[9] 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, pp. 3051-3073, 2009.

[10] I. Tal and A. Vardy, “List Decoding of Polar Codes,” *IEEE Transactions on Information Theory,* vol. 61, no. 5, pp. 2213-2226, 2015.

[11] E. Arıkan, “Systematic Polar Coding,” *IEEE Communications Letters,* vol. 15, no. 8, pp. 860-862, 2011.

[12] I. Tal and A. Vardy, “How to Construct Polar Codes,” *IEEE Transactions on Information Theory,* vol. 59, no. 10, pp. 6562-6582, 2013.

[13] G. Sarkis, 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, 2014.

[14] A. Süral, E. G. Sezer, Y. Ertuğrul, O. Arıkan and E. Arıkan, “Trabits-per-Second Throughput for Polar Codes,” *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC),* 2019.

[15] J. B. Anderson and S. Mohan, “Sequential Coding Algorithms: A Survey and Cost Analysis,” *IEEE Transactions on Communications ,* vol. 32, no. 2, pp. 169-176, 1984.