D4.4

FPGA Verification Report

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 / D4.4/ 1.0
Work package contributing to the deliverable:	WP4
Due date:	August 2020 – M36
Actual submission date:	1st September 2020

Responsible organisation:	CRE
Editor:	Jhon Jimenez
Dissemination level:	PU
Revision:	V1.0

Abstract:	This report will describe FPGA verification of selected encoders/decoders which are best performing and prone to be commercial FEC IPs. It will report the description of simulation chain, implementation results of HW complexity and BER performance.
Keywords:	FPGA, verification, encoders, decoders, FEC

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

Editor

Jhon Jimenez (CRE)

Contributors (ordered according to beneficiary numbers)

Evren Goksu Sezer (POL)

Disclaimer

The information in this document is provided “as is”, and no guarantee or warranty is given that the information is fit for any particular purpose. The content of this document reflects only the author`s view – the European Commission is not responsible for any use that may be made of the information it contains. The users use the information at their sole risk and liability.

Executive Summary

In WP2, an exhaustive design space exploration for each code family was performed, from which the most promising combinations of design parameters were identified in D2.1 [1] and D2.2 [2] to meet the EPIC requirements. Based on these, promising architectural templates were also implemented at register-transfer level (RTL) and thoroughly described in D2.3 [3]. On the other hand, while most of WP4 focuses on the validation and implementation feasibility of these architectures in different advanced technology nodes, it is also important to validate and verify them in terms of their error correction performance. For this purpose, WP4 has laid out a verification approach based on field-programmable gate arrays (FPGA), which allows the measurement of very low error rates with a low computation time cost.

This work is therefore the result of the approach mentioned above and describes the mapping of a few selected architectures onto FPGA, enabling bit error rate (BER) measurements down to 10^{-10} and lower. In order to achieve this, the selected pairs of encoder and decoder have been integrated into a complete hardware simulation chain, creating a fast and efficient environment to carry out testing and measurement of their performance. The presented work also complements the communications performance assessments of WP3.

Several aspects had to be considered prior to the execution of this task, which included the actual selection of architectures for verification. Due to constrained workload resources, it was deemed unfeasible to implement all the architectures investigated in WP2, since the development of an adequate hardware simulation environment is already a rather laborious endeavour for one target architecture. Therefore, the partners involved in the actual implementation process agreed upon a few decoder designs, after preliminary discussions took place. The selected architectures include the low-density parity-check convolutional codes (LDPC-CC) unrolled window decoder, whose verification was undertaken by the partner CRE. Likewise, two polar decoder architectures had also been selected, whose implementation was carried out by the partner POL.

The following document lays out a description of this work and the developed simulation chains in Chapter 2 and Chapter 3. The verification results show a close correspondence with those from preliminary software simulations, while also complementing the latter with further simulation of lower error rates at higher signal-to-noise ratios (SNR). As a consequence, the hardware simulations, and thus the performance assessments carried out in this work are able to reach beyond the scope of what is feasible with software simulations, as initially intended.

Additionally, the FPGA verification systems developed in this work have also served an additional purpose in WP5 during the second half of the project as dissemination tools in various relevant industry- and academic-related events. These served as FPGA demonstrators, showcasing some of the preliminary results of the project in a physical medium that the general public could also appreciate. These dissemination activities have been documented in D5.5 [4]. The FPGA intellectual property (IP) cores developed in this task will also serve as a basis for several high-performance stand-alone encoder/decoder products for the FPGA segment of the forward error correction (FEC) market.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

The bit error rate (BER) is one fundamental performance metric of the design and implementation of practical and efficient FEC solutions. Its significance is directly tied to the increasing demand for higher throughput, as errors can occur more often, the more data is transmitted in a given unit of time, thus creating more stringent BER requirements. For example, assuming an average of one erroneous bit every second as acceptable, the maximum permitted BER approaches 10^{-12} at throughputs approaching the intended 1Tb/s of EPIC. Therefore, it is important to verify the developed EPIC designs' error correction performance at those levels, in order to assess their feasibility under these requirements. However, this poses a practicality problem for software-based simulations, being the standard method of verification, as these error levels require non-practical amount of time to compute. Furthermore, to get BER measurements as low as 10^{-12}, more than 10^{12} bits must be at least simulated just for one simulation scenario, which can take several weeks or even months of computation.

For these reasons, the motivation arises for using FPGA verification as a mean to simulating low BERs in a feasible amount of time. With this approach, parallel and concurrent behaviour can also be implemented and exploited to drastically speed up simulations, thereby reducing simulation times by several orders of magnitude, as well as facilitating measurements of BERs in the order of 10^{-10} and lower. In this way, FPGAs become a powerful prototyping and verification tool, closing the gap between application-specific integrated circuits (ASIC) and software-based verification.

This report describes the development and implementation of several of these FPGA simulation platforms, with the intent of verifying some of the EPIC designs for Polar and LDPC codes. Likewise, this endeavour has also served as a tool for showcasing the results and achievements obtained throughout the project's results, in sundry venues of academic and industrial relevance. These developments also have the underlying purpose of validating the feasibility of EPIC designs as viable and marketable products in the IP core and FPGA industry.

Chapter 2 LDPC Code

2.1 Simulation Chain

The simulation chain aims to verify the error correction performance of the FEC architecture, enabling BER measurements down to 10^{-10}, while also assessing aspects such as throughput and latency under an actual hardware environment. As the targeted architecture exhibits a high degree of parallelism and throughput, so do the rest of the chain's components, to ensure these don't experience any bottlenecks that might affect the performance assessments during simulations. To achieve this, all chain components are fully pipelined and mapped onto the same FPGA chip, to avoid throughput and/or latency limitations with any FPGA interfaces.

This implementation is mapped onto a Xilinx Virtex Ultrascale+ FPGA, which has the necessary hardware resources to accommodate the entire chain. Configuration and reading of the performance measurements from the FPGA is done with a universal asynchronous receiver transmitter (UART) interface, through which a computer controls the execution.

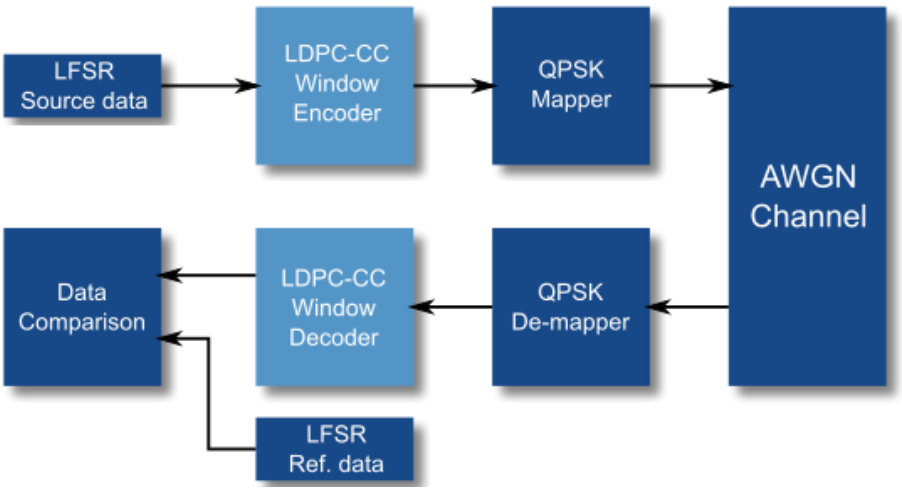

Figure 1. Simulation chain of the LDPC-CC window decoder

Figure 1 is a high-level representation of the simulation chain implemented onto the FPGA. The set-up is comprised of the minimum components necessary to model a transmission chain and therefore it does not contain any interleaving, hybrid automatic repeat request (HARQ) components nor similar. This simplified approach removes the influence of these FEC enhancing components, and puts the focus onto the decoder itself, as this is the design whose performance is to be verified in the first place. Furthermore, this simplification drastically reduces the implementation complexity. A more detailed description of the simulation chain's component is presented in the following sub-sections.

2.1.1 Test Data Generation

The simulation chain employs a hardware-wise low-cost method for generating random test data, which consists of a Galois Linear Feedback Shift Register (LFSR) of 256 bits, initialized with a unique seed. To ensure that the produced sequence of random numbers is long enough, a feedback polynomial has been chosen [5], which theoretically maximizes the period length to 2^n-1 values, with n being the number of bits. Under these parameters, and assuming a clock frequency of 100MHz, the generated sequence of pseudo-random numbers has a period of approximately 1.1×10^{77} clock cycles or 3.6×10^{61} years. This is clearly enough for the purposes of this application.

2.1.2 Encoder and Decoder

The center piece of the simulation chain is the unrolled window-based LDPC-CC decoder (also referred to as spatially-coupled LDPC (SC-LDPC) decoder hereafter), whose architecture and decoding algorithm has been described in [2] and [3]. An implementation of this decoder has been optimized for virtual silicon tape-out and subsequently synthesized in 22nm fully depleted silicon on insulator (FD-SOI) technology featuring a window size, i.e. number of decoding iterations, fixed to 5 with a code-length of 80 sub-blocks of 1280 bits and a code-rate of R=4/5. The corresponding synthesis results were reported accordingly in the RTL implementation report [6]. However, it was unfeasible to map this design with the rest of the simulation chain onto one FPGA, due to several reasons.

In the first place, the intrinsic routing complexity and amount of parallelism made placing and routing the design onto the fixed layout of the FPGA a challenging task, which ultimately exhausted all available hardware resources, leaving no room for the rest of the chain. Furthermore, the amount of parallelism would have also impacted the rest of the chain components in a similar way, further increasing the required hardware resources, as it is explained in the following subsections. Therefore, a more conservative design was selected for this work, which featured the same decoding algorithm and architecture as the aforementioned implementation. The code construction method, as well as the number of decoding iterations (window side) have likewise remained the same. On the other hand, the code- length and rate were reduced for this implementation, which drastically reduced the amount of hardware resources, and allowed the entire simulation chain to be mapped onto a single FPGA. The code-rate is then equal to R = 1/2, while the code-length is equal to 40960 bits, divided into 80 sub-blocks of 512 bits entering the decoder every clock cycle.

The encoder and decoder work in a stream-like fashion, whereby data is expected to be continuously transmitted, as it is needed for spatially coupling it along the stream, as SC-LDPC suggests. Therefore, every clock cycle, each sub-block within a frame carries information, which is partly utilized for decoding of the next one, hence each sub-block is “coupled” with its successor. This characteristic enables the implementation of large code-lengths, such as those mentioned before. On the other hand, this also implies the exchange of information between consecutive frames for the same purpose, which the encoder transmits on the last sub-block of the frame. Consequently, the encoder defines the frame boundaries using an additional signal to flag the last of the 80 sub-blocks, indicating the start of a new frame thereafter. This signal is thus propagated along the chain’s pipeline to be used by the decoder, as well as the data comparison component, to determine the number of simulated frames.

2.1.3 Mapper and De-mapper

The mapping and de-mapping process plays an important role in the development of the complete on-chip simulation chain, not only in terms of performance, but also in computational complexity, hardware resources, and thus implementation feasibility. Arguably the most conspicuous feature of the FEC architectures in this simulation chain is the high level of parallelism. This translates into the mapping of a proportional number of parallel symbols, processed by the additive white Gaussian noise (AWGN) channel and later by the de-mapper.

Higher modulation schemes would mean a fewer number of parallel symbols. However, these are more computationally intensive, thus potentially compromising the overall speed and performance of the chain. An analogue compromise also occurs in the opposite way, whereby a lower modulation scheme would be less computationally intensive, nevertheless with a larger number of parallel symbols. This, on the other hand, would mean a larger amount of hardware resources, not only for the mapping and de-mapping components but also for the AWGN channel, resources which would otherwise be needed to allocate the decoder design. Taking these criteria into account, the quadrature phase shift keying (QPSK) modulation scheme has been selected. This serves as a good compromise for this chain’s implementation, whereby mapping and de-mapping is not too complex, while still halving the number of symbols required, compared to binary phase shift keying (BSPK) and its variants.

The optimum quantization of the mapped and de-mapped symbols proved to be a decisive aspect of the chain's overall performance, particularly the number of fractional bits at each of these ends. The final quantization parameters for this chain were chosen as a good compromise between hardware constraints and accuracy after an initial assessment of these in preliminary software simulations. Therefore, the encoded bits were mapped into blocks of 256 symbols, whose in-phase/quadrature (I/Q) components consist of 12-bits two's complement integers with 11 fractional bits. Likewise, the input symbols of the de-mapper consist of 12-bit I/Q components with 8 fractional bits. In this case, additional bits were added to the integer part with respect to the mapped symbols to account for the added noise from the channel. Finally, the log-likelihood ratio (LLR) values coming from the de-mapper are represented using 6-bit two's complement integers with 2 fractional bits.

2.1.4 Channel (AWGN)

An AWGN channel is used as the source of noise for the data in the chain. Since practical assessment of performance at very low error rates is one of the main motivations for this hardware-accelerated simulation chain, the channel must exhibit a good degree of accuracy. On the other hand, a parallel and efficient implementation of the channel is also required, due to the high level of parallelism and throughput mentioned earlier. Together with fixed-point integer representation of data, these contrasting aspects converge into an implementation challenge, which seeks an efficient low-complexity design with little loss of accuracy.

This implementation uses a proprietary IP core from the partner Creonic GmbH, consisting of an AWGN channel designed using independent noise generators for each input symbol. These have been fully pipelined to meet the desired throughput, while keeping a reasonable amount of hardware resources. Similar to the test data, the noise generation starts with LFSRs of 64 bits initialized with a unique random seed, which create samples with a uniform distribution. Likewise, the feedback polynomial has been chosen to achieve the maximum period length of 2^{64}-1 samples, which suffices for this application. These samples are subsequently processed to obtain the noise with the desired normal distribution, using an implementation based on the Box-Müller algorithm [7]. The resulting noise is normalized according to the SNR configuration and then added to the corresponding symbol. The channel supports a wide range of SNRs with a resolution of 0.1dB (E_s/N_0). The SNR value can be configured at any moment during run-time. Alternatively, the channel can also be switched between normal operation and bypass mode, where in the latter no noise is added to the original data. Much like the mapper and de-mapper, the input and output symbol quantization of the channel is also fully parametrizable at design-time.

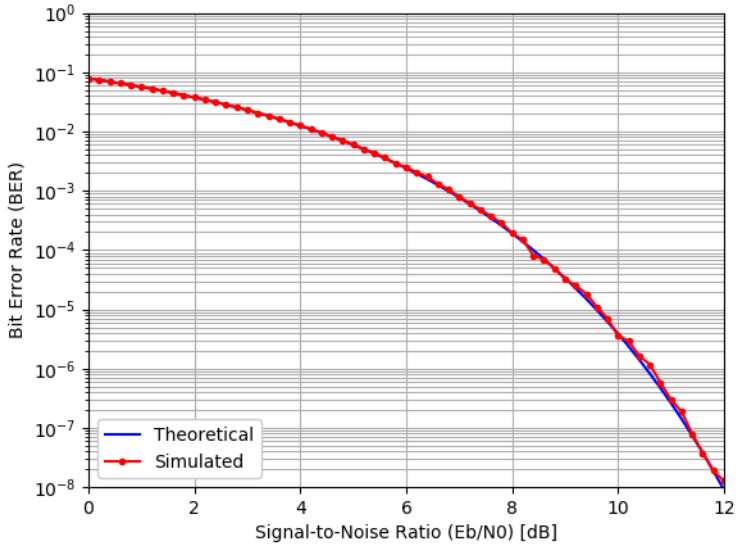

Figure 2. BER curves for uncoded transmission with AWGN channel and QPSK modulation

The previous figure shows a comparison between the theoretical bit error probability for QPSK [8] modulation and the measured BER of the AWGN channel implementation used in this work, across a range of SNR values. For these measurements, the encoder and decoder were removed from the simulation chain, leaving the mapper, channel and de-mapper as the only active components in it. Additionally, the channel configured for processing 256 symbols in parallel, with the quantization described in section 2.1.3. The results show that these components exhibit the expected behavior and a high degree of accuracy under these parameters, which means little additional error is introduced into the final system and thus into the final performance assessment of the decoder.

2.1.5 Data Comparison and Performance Measurements

Finally, the end-point of the chain receives the decoded data, which is compared against a reference in order to get the frame error rate (FER) and BER measurements. Just like the other end of the chain, the reference data is also generated with a Galois LFSR, which has been initialized with a seed identical to that of the source. However, the reference Galois LFSR is only activated every time a new block of decoded data has arrived, hence keeping decoded and reference data synchronized. Having these two pseudo-random number generators with identical seeds is advantageous, as it eliminates the need of a pipeline-long first-in first-out (FIFO) memory to forward the source data to the end of the chain, and gives to the synthesis tool a little more freedom to place and route the design onto the chip.

The reference and decoded date are compared on a bit-by-bit basis, whereby the number of erroneous bits is incremented for every bit-mismatch in the comparison. This is done for every incoming sub-block of data, every clock cycle. On the other hand, since one frame is comprised of several sub-blocks of bits, the number of erroneous frames is only incremented if there have been any bit errors during the current frame by the time the last block has been compared. Adjacently, the number of simulated frames is also updated for every incoming frame, regardless of bit-errors.

An execution control mechanism has also been built into this component, where the current execution of a certain configuration scenario is stopped upon reaching a maximum number of erroneous frames. To do so, the maximum number of simulated frames, as well as a threshold of erroneous frames are configured prior to execution, with the latter being considerably lower. With these parameters, the simulation doesn't have to run through the maximum number of frames, if enough erroneous ones have been recorded. This aids to reduce unnecessary simulation time, at lower SNR values, where errors can occur more often.

2.2 Results

The following table offers a short summary of the parametrization of the components present in the simulation chain. This offers a hint of the influence of these parameters on the overall accuracy of this set-up, from the perspective of the communication's performance. Notice that the type and quantization of the mapper, channel and de-mapper are decisive in this regard.

Table 1. Summary of the simulation chain's components

	Data source	Encoder	Mapper	AWGN Channel	De-mapper	Decoder
Type	LFSR	SC-LDPC	QPSK	Box-Müller	QPSK	SC-LDPC
Parallelism	256 (bits)	512 (bits)	256 (symbols)		512 (LLRs)	256 (bits)
Quantization	1		12 (11fract. bits)	12 (8 fract. bits)	6 (2 fract. bits)	1

2.2.1 Implementation Results

Table 2 contains the results of synthesis, place and route of the encoder and decoder on the selected FPGA. In terms of complexity, there is a clear absence of more specialized hardware resources, such as block RAMs (BRAMs) or multipliers, in both designs' implementations, only utilizing look-up tables (LUTs) and flip-flops (FFs) as their building components. Although this could give the impression of a mostly combinational architecture with low complexity, the greater source of complexity comes not only from the amount of resources but from the routing congestion, as indicated in Section 4.3 of [6]. Unlike an ASIC implementation, this is a particularly more complex task on FPGAs, since the architecture's components must be placed and routed onto a fixed FPGA layout. For this reason, the routing congestion is also prominent in this work, as there is no absolute freedom for placing and routing hardware resources, without incurring into longer critical paths and timing violations. As a consequence of these factors, the working frequency had to be drastically reduced to 100MHz, affecting the throughput as well.

Table 2. Complexity of LDPC-CC encoder/decoder at 100MHz on xcvu9p-flga2104-2L-e

	LUTs	FFs
Encoder	1198 (0.1%)	2858 (0.12%)
Decoder	412935 (34.92%)	198401 (8.39%)

On the other hand, the RTL description of the designs used in this work features no FPGA-specific optimizations, since it is primarily intended for an ASIC implementation. Furthermore, this work has sought primarily to aid in the verification of decoder architectures for silicon tape-out, and tinkering with it in this matter would essentially change the target design, invalidating the verification to some extent. Despite all this, the results show great potential for this design to become a viable IP block for FPGA applications. With this objective in mind, factors such as long critical path and timing constraints could be significantly improved with an RTL description that better fits a FPGA architecture and its available components.

2.2.2 Error Correction Performance

Figure 3 offers a look into the simulation results of both the software- and hardware-based chains, as well as a comparison thereof. For this work, a preliminary software simulation was carried out as the baseline for comparing and validating the FPGA implementation. The software simulation features the same chain set-up depicted in Figure 1, as well as an almost identical parametrization as its FPGA equivalent. Nevertheless, several minor details are still different, such as the type of source data and random Gaussian noise generators, although they have the same behaviour as their hardware counterpart (see Section 2.1.4).

Error rate measurements took place within a SNR range of 0 dB to 12 dB, with increments of 0.2 dB between every consecutive measurement. For each of these measurements, the chain was configured to simulate a maximum of 100,000,000 frames, with a threshold of 100 erroneous frames as an early-stop condition. As explained in Section 2.1.5, this drastically reduces the computation time, when simulating at lower SNR values, where errors occur more often. On the other hand, the software simulation exhibits the same configuration, but spans a shorter SNR range of 8 dB. Simulating beyond this value would have meant protracted computational times, spared by the FPGA set-up, as intended with this work. As an additional remark, the hardware-based simulation featured a total runtime of about 10 minutes.

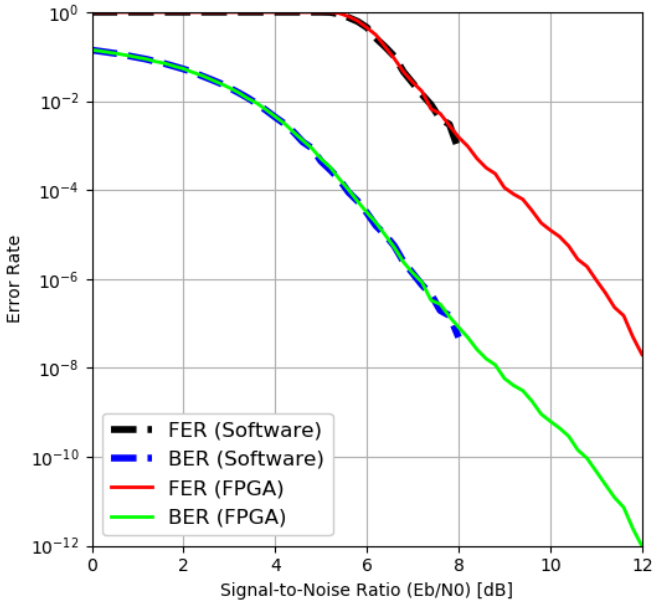

Figure 3. Error correction performance of the LDPC-CC window decoder with N=40960, R=1/2

The comparison shows a close correspondence between the software and hardware simulation's results, as evidenced by the overlap of both sets of curves. The results also show that the window decoder achieves a BER of 10^{-12} at just shy of 12 dB, which is congruent with the initial EPIC goals. On the other hand, it is important to note that, despite the decreasing BER, the FER remains equal to 1 until about the 5 dB mark. This can be explained while considering the actual frame length. Although the decoder outputs 256 bits every clock cycle, the actual frame is comprised of 80 of these sub-blocks. However, the end-point of the chain counts one frame error, if at least one bit out of the 80 sub-blocks is erroneous. Therefore, it is more likely to count frame errors during the simulation, the longer the frame becomes. Hence this is one of the characteristics of the convolutional nature of the SC-LDPC.

Chapter 3 Polar Code

The EPIC project aims to develop a power efficient high throughput FEC solution for wireless systems. Although the final output of the project is going to be on ASIC; FPGA test systems are required to verify very low error rates. For testing the polar code systems, an end-to-end simulation chain is developed and used with two different polar decoders, successive cancellation (SC) for block length *N=1024* and data length *K =854* (SC (1024,854)); and successive cancellation list (SCL) of two for block length *N=1024* and data length *K=888* (SCL2 (1024,888)) aided with 4 bits of cyclic redundancy check (CRC). The components of the simulation chain are explained in Section 3.1. Results of the solutions, both in terms of complexity and communication performance are presented in Section 3.2.

3.1 Simulation Chain

The purpose of the simulation chain is to check the correctness of the system and, if the system is working properly, measure the communication performance of the decoder at very low error rates. Speed of the simulation chain is important for two main reasons. First, if the throughput is higher, measuring the communication performance of the system for very low FER/BER values takes relatively less time. More importantly, there is a high correlation between the achievable throughput on FPGA and ASIC; an architecture with high throughput on FPGA translates to high throughput on ASIC.

The FPGA simulation chain for polar code is implemented on Xilinx Virtex Ultrascale+ FPGA chip and reaches 200 Gb/s net throughput. Simulation chain is fully pipelined, and every component of the chain implemented on chip, otherwise reaching such high throughputs would be near impossible due to limited throughput at the FPGA interface. The components of the simulation chain and their sequential order can be seen in Figure 4. Following sections explain these components.

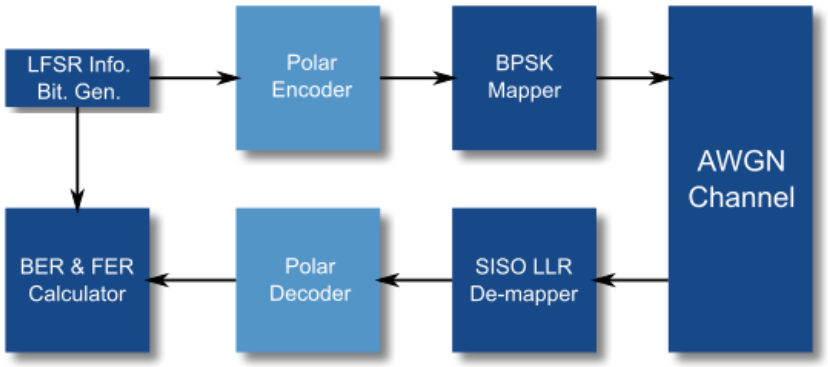

Figure 4. Simulation chain of the SC/SCL polar decoders

3.1.1 Test Data Generation

First step of the simulation chain is creating random test inputs which is done by 32-bit LFSRs. We use $X^{32}+X^{22}+X^{2}+X^{1}+1$ [9] feedback polynomial for achieving maximum length feedback. In this case maximum length is 4294967295 (2^{32}-1), after 4294967295 clock cycles LFSR starts to repeat itself. Although this can occur within a simulation point at very low error rates, it has no effect on performance since polar codes are linear and consequently performance is data independent. These randomly generated inputs are fed into the polar encoder and written in a FIFO memory at

the same time. After the delay in the actual data path, the FIFO output is compared with the output of the system to calculate FER and BER.

3.1.2 Encoder and Decoder

The polar code solutions in EPIC employ a systematic encoder [10], as these achieve a better communication performance compared to non-systematic encoder without any significant increase of hardware complexity. The polar encoder is a small circuitry which can be easily implemented in single clock cycle. We use an unrolled architecture to simplify and speed up the encoder. If the system is aided with CRC, one clock cycle is also required for encoding the CRC bits. After CRC bits are encoded, all the data bits and CRC bits are encoded with the systematic polar encoder. The SCL2 (1024, 888) polar decoder uses 4 bits of CRC, meaning 892 bits are encoded with polar encoder. CRC polynomial is chosen as $x^4+x^3+x^2+x+1$ [11].

On the other hand, the decoder is the main output of the EPIC project and the most complex module of a FEC solution. Other components of the simulation chain are present to aid in measuring the performance of the decoder. Naturally, most of the engineering effort is used toward researching and developing (R&D) the algorithms and architectures of the decoders. After this R&D study has matured, an automated tool was developed for writing the VHDL code of the decoders. This tool can easily and quickly generate SC and SCL2 decoders with different block lengths, code rates and designs. This work presents two example decoders from this vast array of possible solutions

3.1.2.1 SC (1024, 854)

SC is a very low complexity decoding algorithm for polar codes. It has a sequential structure which promotes the use of pipelining to reach high throughput. The latency of the original SC algorithm is equal to 2*N-2 in fully pipelined form, thanks to its sequential structure. A higher latency also increases the amount of memory required for buffering as well. However, simple arithmetic and clear flow of the decoder allow the optimization of this and many other aspects. These optimizations include parallel decision shortcuts [12], [13], [14], which helped reducing both complexity and latency and thus memory consumption, translating into a significant improvement. To reduce the complexity even more, advanced quantization and register balancing methods were also developed [15]. The latency of the decoder dropped to 60 from original latency of 2046 clock cycles after applying these, which made the complexity of the decoder small enough, so that whole simulation chain could fit into the FPGA chip.

3.1.2.2 SCL2 (1024, 888) aided with CRC

SCL decoders [16] offer better communication performance compared to SC. Especially if SCL is aided with CRC, it performs considerably better [16]. However, this increment in performance comes with a cost of increased implementation complexity. SCL has very similar structure and architecture to SC and thus exhibits high latency and potentially high memory consumption as well, which can be solved in similar manners. Usage of parallel bit decision short cuts stays the same, but algorithms used for shortcuts differ [17], [18]. Likewise, this work utilizes register balancing for SCL as well. After applying these optimizations, the latency of the decoder dropped to 64 from original latency of 2046 clock cycles.

3.1.3 Mapper and De-mapper

Encoded bits are mapped to BPSK data symbols as follows. An encoded “0” bit is mapped to a 16-bit “+1” symbol with 8-bit fractional values using fixed point arithmetic. Similarly, an encoded “1” bit is mapped to a “-1” symbol. The encoded bits are corrupted with noise while going through the AWGN channel. In order to achieve best performance, we de-map the noisy encoded bits with soft quantization, which means they are quantized with more than one bit. Our simulation chain uses only 5 bits to minimize the performance loss because of quantization while keeping the complexity in check. The de-mapper truncates the noisy encoded bits to 5-bit LLR values with 3 fractional bits, represented in signed magnitude format.

3.1.4 Channel (AWGN)

The AWGN channel is implemented on the chip like the rest of the simulation chain to satisfy the throughput requirements. One of the aims of the simulation chain is to test the performance of the decoder at very low FER/BER (e.g. 10^{-15} BER). To be able to test the decoder at such low error rates, tails of Gaussian noise should exhibit a high degree of accuracy in order to simulate high SNR values. Therefore, the Gaussian number generator (GNG) of this simulation chain is developed by combining three 32-bit LFSR systems. The generated noise is represented with 16 bits; 11 of which are fractional. The standard deviation of the Gaussian distribution is ±9,1. Lastly, the generated Gaussian noise is scaled according to the desired SNR and added to the encoded and mapped bits.

Figure 5 illustrates the performance curves for uncoded transmission, showing two different GNGs implemented on two different FPGAs. The number after the GNG represents the parallelism level of the GNG e.g. GNG-16 means there are 16 GNG are employed in parallel starting from calculated different initial seeds to increase the throughput. The BER performance curves show that the performance of GNG does not deteriorate as parallelism level increases. The simulation chains of the SC (1024, 854) and SCL2 (1024, 888) utilizes GNG-1024 in order to cope with the throughput of the system.

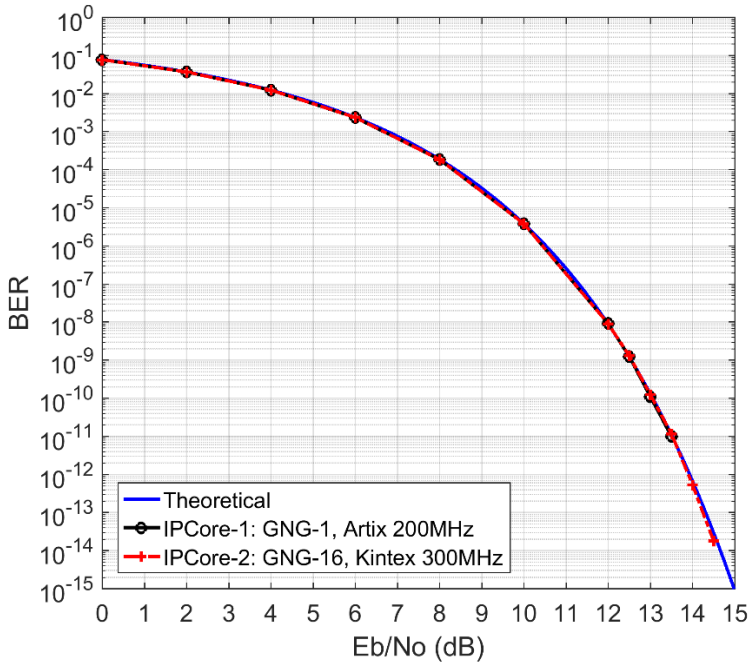

Figure 5. BER curves for uncoded transmission with AWGN channel and BPSK modulation

3.1.5 Data Comparison and Performance Measurements

The generated input data stored in a FIFO memory is used in this step. The output of the decoder is compared bit by bit with the corresponding input values. If they are not the same, the frame error number is increased by one and the bit error number increased by the number of bits that are different between the input and output data.

3.2 Results

3.2.1 Implementation Results

Table 3, Table 4 and Table 5 present the implementation results in terms of LUTs, FFs and BRAMs. As it can be seen from the tables, any of the decoders consumes at most the 16% of the FPGA, therefore they can easily fit into the Xilinx Virtex Ultrscale+ (xcvu37p-fsch2892-2L-e-es1) or even smaller FPGAs. Much smaller FPGAs can also be used for SC (1024,854) as its complexity

is much lower than that of the SCL2 (1024, 888) decoder. However, as it will be presented in the next section, SCL2 (1024, 888) has better error correction performance.

Complexity values are presented for two different implementations of SC (1024, 854) in Table 3 and Table 4. It can be seen from the tables that, there is a trade-off between complexity of the decoder and the maximum achievable frequency. It is possible to increase the operating frequency of the simulation chain by adjusting the parameters of the shortcuts and register balancing technique.

Table 3. Complexity of SC (1024,854) running at 120 MHz on xcvu37p-fsch2892-2L-e-es1

	LUT	FF	BRAM
Encoder	1,842 (0.14%)	1,825 (0.07%)	-
Decoder	53,247 (4.08%)	35,192 (1.35%)	136 (6.75%)

Table 4. Complexity of SC (1024,854) running at 250 MHz on xcvu37p-fsch2892-2L-e-es1

	LUT	FF	BRAM
Encoder	1,842 (0.14%)	1,825 (0.07%)	-
Decoder	95,653 (7.34%)	50,700 (1.95%)	72 (3.55%)

Table 5. Complexity of SCL2 (1024,888) aided with 4 bits CRC on xcvu37p-fsch2892-2L-e-es1

	LUT	FF	BRAM
Encoder	1,884 (0.16%)	1,918 (0.08%)	-
Decoder	190,084 (16.08%)	63,528 (2.69%)	108 (4.98%)

As shown in Table 3 and Table 5, even though SCL2(1024,888) is more complex than the SC(1024,854); the number of BRAMs used in SCL2 is less than in SC. Using BRAMs on FPGAs increases the routing complexity, as already mentioned SCL2(1024,888) is a more complex design and thus routing for SCL2(1024,888) is harder. Therefore, using the FFs instead of BRAMs makes routing the design easier. Thus, utilizing BRAMs or FFs is a design choice which does not necessarily shows the complexity of the design.

3.2.2 Error Correction Performance

At the end of the simulation chain, the output of the decoder is compared with the corresponding input bit by bit. Result of every comparison is saved and stored. This comparison is performed repetitively for each SNR value of interest. This Monte Carlo simulation provides us with communication performance of the decoder for different SNR values. Communication performance of both decoders SC(1024,854) and SCL2(1024,888) aided with 4 bits CRC is given in Figure 6 and Figure 7 respectively.

Error correction performance of the decoders on software are also provided in the figures to show the performance loss due to quantization.

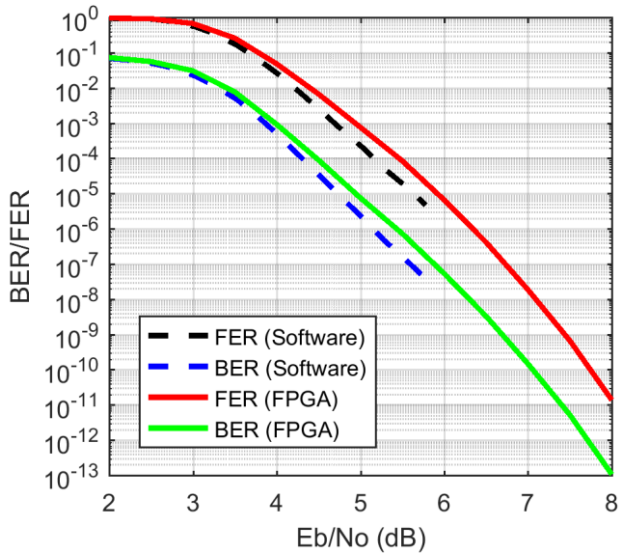

Figure 6. Error correction performance of SC (1024,854)

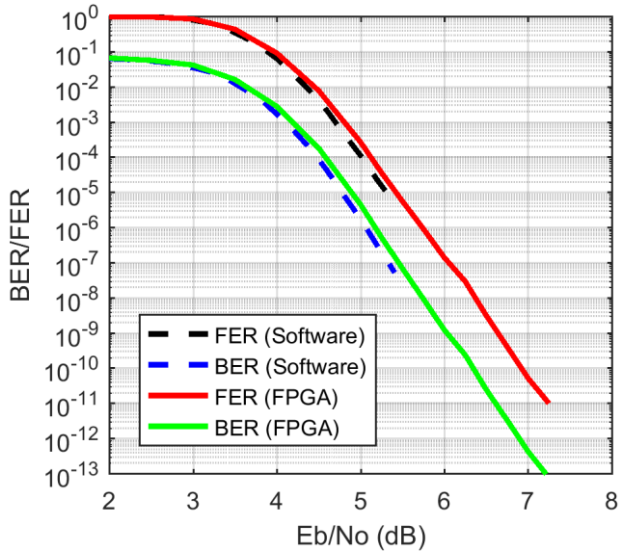

Figure 7. Error correction performance of SCL2 (1024,888) aided with 4 bits CRC

To sum up, polar code simulation chain consists of polar encoder, polar decoder and AWGN channel IP cores and supporting components such as mapper, de-mapper etc. Simulation chain can reach 200 Gb/s net throughput on Xilinx Virtex Ultrascale+ FPGA. Our automated tool can create polar encoder and decoder designs that are compatible with the simulation chain with various block sizes and code rates. These designs reach very high throughputs with low hardware complexity.

Chapter 4 Summary and Conclusion

This work has presented the implementation of various FPGA simulation environments for some of the decoder designs and architectures investigated during the EPIC project. These simulation environments allowed a rapid and efficient verification of the selected designs, as well as the assessment of their communication performance at lower error rates than it is currently practical and feasible with just traditional software simulations. All the implemented FPGA simulation chains in this work exhibited a significant reduction of the simulation time by several orders of magnitude, compared to software counterparts. The loss of error correction performance is due to limited quantization of LLR values, which will be the actual case for an ASIC implementation as well. This has made them powerful prototyping and validation tools for FEC designs, closing the gap between ASIC and software verification.

With respect to the LDPC-CC window decoder, the results indicate a strong correlation between the FPGA and software simulation results, validating the use of the hardware simulation chain. Furthermore, the simulation results indicate that the window decoder achieves a BER of 10^{-12} at a SNR of 12 dB, which agrees with the conservative assumption of 1 erroneous bit every second at 1Tb/s as acceptable. The implementation results also look promising, despite the reduced frequency that was achieved. This, however, was not addressed in this work, as the focus lay primarily on using the FPGA as a tool for the validation and verification of the actual ASIC design Nevertheless, the obtained results still indicate great potential and room for future improvements and optimizations to fully leverage the design as an IP block for FPGA platforms.

Likewise, the results obtained from the polar codes' simulation chains were satisfactory, achieving a throughput up to 200 Gb/s, while still exhibiting a good communication performance. The simulation results indicate that the SC and SCL decoders achieve a BER of 10^{-12} at a SNR of 7.6 and 6.9 dB respectively. For this approach, the decoder designs underwent several optimization techniques to leverage their full performance and validate them as viable IP blocks for FPGA. Additionally, the implementation of these hardware simulation chains was supported with an automated framework for generating the polar decoder designs.

Chapter 5 List of Abbreviations

Abbreviation	Translation
ASIC	Application-Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
BRAM	Block Random Access Memory
CRC	Cyclic Redundancy Check
FF	Flip-Flop
FEC	Forward Error Correction
FER	Frame Error Rate
FIFO	First-In First-Out
FPGA	Field-Programmable Gate Array
GNG	Gaussian Number Generator
HARQ	Hybrid Acknowledge-Request
IP	Intellectual Property
LDPC	Low Density Parity Check
LDPC-CC	Low Density Parity Check – Convolutional Codes
LFSR	Linear Feedback Shift Register
LLR	Log Likelihood Ratio
LUT	Look-Up Table
QPSK	Quadrature Phase Shift Keying
RTL	Register Transfer Level
SC	Successive Cancellation
SCL	Successive Cancellation List
SC-LDPC	Spatially-Coupled Low Density Parity Check
SNR	Signal-to-Noise Ratio
UART	Universal Asynchronous Receiver-Transmitter

Chapter 6 Bibliography

[1] EPIC, „D2.1 - First Design Report," 2018.

[2] EPIC, „D2.2 - Second Design Report," 2019.

[3] EPIC, „D2.3 - Third Design Report," 2019.

[4] EPIC, „D5.5 - Communication and Dissemination Report V2," 2019.

[5] R. T. M. Ward, *Table of linear feedback shift registers,* Department of Physics, University of Otago, 2007.

[6] EPIC, „D4.2 - RTL Implementation Report," 2019.

[7] G. E. P. Box und M. E. Muller, „A Note on the Generation of Random Normal Deviates," The Annals of Mathematical Statistics. 29 (2): 610–611, 1958.

[8] M. S. John G. Proakis, Communication Systems Engineering, Prentice Hall, 1994.

[9] A. K. Panda, P. Rajput und B. Shukla, FPGA Implementation of 8, 16 and 32 Bit LFSR with Maximum Length Feedback Polynomial using VHDL, IEEE International Conference on Communication Systems and Network Technologies, Volume: 2, 2012.

[10] A. Erdal, Channel Polarizataion: A Method for Constructing Capacisty-Achieving Codes for Symmetric Binary-Input Memoryless Channels, IEEE Transactions on Information Theory, vol. 55, no. 7, pp. 3051-3073, 2009.

[11] P. Koopman und T. Chakravarty, Cyclic redundancy code (CRC) polynomial selection for embedded networks, International Conference on Dependable Systems and Networks, 2004.

[12] A. Alamdar-Yazdi und F. R. Kschischang, A simplified successive-cancellation decoder for polar codes, IEEE Communication Letters, vol. 15, no. 12, pp. 1378-1380, 2011.

[13] G. Sarkis, P. Giard und A. Vardy, Fast polar decoders: algorithm and implementation, IEEE Journal on Selected Areas in Communications, vol. 32, no. 5, pp. 946-957, 2014.

[14] M. Hanif und M. Ardakani, Fast Successive-Cancellation Decoding of Polar Codes: Identification and Decoding of New Nodes, IEEE Communications Letters, vol. 21, no. 11, pp. 2360-2363, 2017.

[15] A. Sural, E. G. Sezer, Y. Ertugrul, O. Arikan und E. Arikan, Terabits-per-Second Throughput for Polar Codes, IEEE 30th International Sysmposium on Personal, Indoor and Mobile Radio Communications (PIMRC Workshops), 2019.

[16] I. Tal und A. Vardy, List Decoding of Polar Codes, IEEE Transactions on Information Theory vol. 61, no. 5, pp. 2213-2226, 2015.

[17] S. A. Hashemi, C. Condo und W. Gross, Fast Simplified Successive-Cancellation List Decoding of Polar Codes, 2017 IEEE Wireless Communications and Networking Conference Workshops, 2017.

[18] G. Sarkis, P. Giard, A. Vardy, C. Thibeault und W. J. Gross, Fast List Decoders for Polar Codes, Bd. 34, IEEE Journal on Selected Areas in Communications , Feb. 2016, pp. 318-328.