Abstract—In theory, Polar codes do not exhibit an error floor under successive-cancellation (SC) decoding. In practice, frame error rate (FER) down to $10^{-12}$ has not been reported with a real SC list (SCL) decoder hardware. This paper presents an asymmetric adaptive SCL (A2SCL) decoder, implemented in real hardware, for high-throughput and ultra-reliable communications. We propose to concatenate multiple SC decoders with an SCL decoder, in which the numbers of SC/SCL decoders are balanced with respect to their area and latency. In addition, a novel unequal-quantization technique is adopted. The two optimizations are crucial for improving SCL throughput within limited chip area. As an application, we build a link-level FPGA emulation platform to measure ultra-low FERs of 3GPP NR Polar codes (with parity-check and CRC bits). It is flexible to support all list sizes up to 8, code lengths up to 1024 and arbitrary code rates. With the proposed hardware, decoding speed is 7000 times faster than a CPU core. For the first time, FER as low as $10^{-12}$ is measured and quantization effect is analyzed.

Index Terms—Polar codes, A2SCL, Emulation platform, FER.

I. INTRODUCTION

Polar codes, proposed by Arikan [1], has been selected by the 5G standards. Polar codes with successive-cancellation (SC) decoding theoretically achieve channel capacity in the asymptotic sense. To improve error-correction performance at short or moderate lengths, SC list (SCL) decoding is proposed by keeping $L$ codeword candidates. Concatenated with cyclic redundancy check (CRC) [2] or parity check (PC) [3] bits, the error-correction performance can be further improved.

One advantage of polar codes is that it does not exhibit an error floor when decoded by the SC and SCL algorithms. This makes Polar codes suitable for applications with stringent error performance requirements. For some industrial and medical applications, FER is required to be smaller than $10^{-10}$. However, an efficient hardware solution designed for this purpose has not been reported yet.

It is not easy to achieve this goal in an efficient way, because both decoding latency and throughput should be highly optimized within limited chip area. To our best knowledge, an ultra-low FER below $10^{-10}$ has not been reported from a real hardware. Although many efforts have been made to optimize the decoder hardware of Polar codes [4]–[11], the lowest FER reported in a real hardware is $10^{-6}$ (not fulfilling the $<10^{-10}$ requirement). An FPGA emulation platform is designed for ultra-reliable communications [12], but does not present any hardware-measured FER results.

A. Motivation and Contribution

To achieve ultra-reliable and high-throughput decoding, we adopt the adaptive SCL decoder framework in [13]. To further improve throughput, we propose an asymmetric adaptive SCL (A2SCL) decoder, based on the observation that SC and SCL decoders exhibit huge differences in terms of the area & latency, as well as quantization precision. A2SCL mainly adopts the following two techniques:

1) Asymmetric deployment: the number of SC and SCL decoders are no longer 1:1 as in the original design, but carefully chosen to reflect their significant difference in terms of area and latency.
2) Asymmetric quantization: The different demands for data precision between SC and SCL decoders are also exploited to pack as many SC decoders for parallel decoding, yet without FER loss.

In addition, we provide a reference design through an efficient emulation platform in an FPGA, and evaluate the ultra-low FER performance of Polar codes to demonstrate its practical value. The proposed A2SCL decoder not only achieves FER $\approx 10^{-12}$, but also supports list sizes 1, 2, 4, 8 with maximum code length $N_{max} = 2^{10}$. The emulation platform has the following features:

- **Integrity:** All modules in the link-level emulation such as source vector generator, encoder, modulator, AWGN channel and decoder are executed in the FPGA. The server is only responsible for the code length/rate configuration and results collection.
- **Efficiency:** The emulation platform dramatically improves evaluation speed. One FPGA board can be up to 7000 times faster than a CPU core.
- **Flexibility:** The emulation platform supports CA-Polar (up to 24 CRC bits), PC-Polar [3] (as specified by 3GPP), various rate-matching schemes, list sizes, code lengths and code rates. All these can be configured by the server on the fly.
- **Scalability:** A server can manage one or more FPGAs to speed up the emulation. Servers can also form a cluster to further speed up the emulation.

With the emulation platform, ultra-low FER performance of Polar codes is measured and the error-correction performance of 3GPP NR Polar codes is evaluated.
II. POLAR CODES

An \((N, K)\) polar codes has \(N\) coded bits and \(K\) information bits. The code rate is \(R = K/N\). The information bits are assigned to the \(K\) most reliable sub-channels, and frozen bits, typically zeros, are assigned to the remaining ones. The encoding of Polar code is \(c = u F^\otimes n\), where \(u\) is the information vector (including information and frozen bits), \(F^\otimes n = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^\otimes n\) is the transformation matrix, where \(\otimes\) denotes Kronecker power, and \(n = \log_2 N\).

A. SC-based Decoders

The decoding graph of SC decoder is shown in Fig. 1. The soft bits propagate from right to left and the hard bits propagate from left to right. A hardware-friendly version of soft value updating is carried out in log-likelihood ratio (LLR) domain \([8]\). Two incoming LLRs \((L_{in1} \text{ and } L_{in2})\) are combined to produce \(L_{out}\) with the following f-function

\[
L_{out} = \text{sign}(L_{in1} \cdot L_{in2}) \cdot \min(|L_{in1}|, |L_{in2}|). \tag{1}
\]

or g-function

\[
L_{out} = L_{in1} + (-1)^{\hat{s}} \cdot L_{in2}, \tag{2}
\]

where \(\hat{s}\) is the modulo-2 sum of previously decoded bits and is called partial sum (PS).

For an SCL decoder, the decoding process is similar to SC decoder except that it keeps \(L\) paths. When making hard decision for each bit, \(L\) paths split into \(2L\) paths, and the ones with smallest path metric (PM) are kept. For the \(i_{th}\) path and bit \(u_i\), the LLR of stage 0 is denoted by \(L^i_{0,i}\) and its hard decision is denoted by \(\beta^i_i\). The PMs update according to

\[
PM^i_i = \begin{cases} 
PM^i_{i-1}, & \text{if } u^i_i = \beta^i_i \\
PM^i_{i-1} + |L^i_{0,i}|, & \text{otherwise}
\end{cases} \tag{3}
\]

After all bits are decoded, the path with the smallest PM is selected as the decoding output.

For CRC aided SCL (CA-SCL), the most reliable path that passes CRC check is selected as the decoding output. For parity-check SCL (PC-SCL), each parity bit is decided by its parity function rather than by its LLR. A PC-CA-SCL decoder combines the features of both, if both CRC bits and PC bits are employed. Throughout this work, we implement CA-SCL and PC-CA-SCL decoders.

III. ASYMMETrIC ADAPTiVe SCL (A2SCL) DECODER

The original adaptive SCL decoder \([13]\) progressively increases the list size until a packet is successively decoded or a maximum list size \(L_{\text{max}}\) is reached. Our implementation is built upon a simplified version of \([13]\) that has only two decoders, i.e., an SC and an SCL with a given list size. The algorithm is described in Algorithm 1.

Although the software implementation of Algorithm 1 is rather straightforward, its hardware implementation is different. One has to take into account the huge difference between an SC decoder and an SCL decoder in terms of hardware resource, as well as their work load at a target signal-to-noise ratio (SNR).

The chip area & decoding latency comparison between an SC and an SCL decoder \((L = 8)\) is shown in Table II. The (normalized) measurements are based on our reference ASIC implementations in \([11]\), with both SC and SCL decoders optimized to their best efficiency (see details in \([11]\)). According to the measurements, both the area and latency of an SCL decoder \((L = 8)\) is up to 6 times of an SC decoder with the same quantization and code rate. If we implement many SCL decoders with different list sizes, both the area efficiency and time efficiency will be very low.

The work load comparison between an SC and an SCL decoder is given through a case study of \((N = 1024, K = 512, 24\) CRC bits) Polar codes. As shown in Fig. 2 the required SNR for CA-SCL with \(L = 8\) to achieve ultra-reliable communications \((\text{FER} \leq 10^{-8})\) is around 3.5 dB. In such a high SNR region, an SC decoder already exhibits very small FER \((\sim 10^{-4})\), i.e., only loses one or two packets in 10,000.

That means, while SC needs to process all packets, only a small fraction of packets need to be processed by the SCL decoder. This is a huge difference in terms of work load.

Considering the above, a direct implementation of \([13]\) would incur very low hardware utilization efficiency. To address this, we propose an asymmetric adaptive SCL (A2SCL) decoder to overcome the above mentioned issues.
TABLE I
Comparison of area and latency between an SC and SCL decoder with \( \text{L} = 8 \)

| Decoder      | Area / Quantization | Latency / Code Rate |
|--------------|---------------------|---------------------|
|             | 6 bit | 8 bit | 1/8 | 1/4 | 1/2 | 3/4 |
| SC          | 1.00  | 1.27  | 1.00 | 1.43 | 2.04 | 2.34 |
| SCL with \( \text{L} = 8 \) | 5.06  | 6.32  | 4.69 | 7.39 | 9.94 | 12.1 |

A. Asymmetric deployment

To increase throughput, an A2SCL decoder deploys as many SC decoders as possible. To improve efficiency, A2SCL implements only one SCL decoder (e.g., \( \text{L}_{\text{max}} = 8 \)) instead of many SCL decoders with different list sizes (e.g., \( \text{L} = 2, 4, 8 \)).

A scheduler with a MUX is used to collect the CRC-failed packets from the SC decoders, and send them to the SCL decoder. Fig. 3 shows the hardware architecture of the A2SCL decoder. We refer to the different number of SC decoders and SCL decoder as "asymmetric deployment".

![Fig. 3. A2SCL decoder hardware architecture](image)

The SC decoder Core and SCL decoder Core have the similar architecture as described in [11], which summarizes some state-of-the-art optimizations over SC and SCL decoders. Both decoders only store intermediate LLRs for every two neighboring stages in the trellis graph shown in Fig. 1. The "double-packet mode" and "decoded-bit recovery" features are enabled to reduce the number of LUT/BRAM/FF modules. The hardware-friendly "syndrome-check" [?] and "decision-aided" [9] approaches are adopted to increase the throughput of SC and SCL decoders, respectively.

Assume the work target is \( \text{FER} < 10^{-9} \), in almost all cases, SC decoder’s \( \text{FER} < 10^{-3} \) under the target SNR. According to simulation results and real hardware test results [11], the SC decoder and SCL decoder’s throughput ratio is 5:1. Thus, the SCL decoder can process the failed packets of 200 SC decoders at \( \text{FER} < 10^{-9} \).

The LLR buffer size of the SCL decoder should be larger than those of SC decoders, in case that many SC decoders generate failed packets at the same time. The following formula evaluates the probability that, during one SCL decoding, the SC decoders have failed \( e \) packets.

\[
P(e) = \left( \frac{c}{e} \right) \times \text{FER}^e \times (1 - \text{FER})^{c-e},
\]

where \( c = \frac{N_{\text{SC}} \times 2 \times T_{\text{SCL}}}{T_{\text{SC}}} \) is the total number of packets processed by the SC decoder during one SCL decoding, \( N_{\text{SC}} \) is the number of SC decoders in the A2SCL, \( T_{\text{SCL}} \) and \( T_{\text{SC}} \) are the decoding time of SCL and SC decoders, respectively.

In our final design, \( N_{\text{SC}} = 18 \) SC decoders are implemented in the A2SCL decoder. As mentioned above, \( T_{\text{SCL}} / T_{\text{SC}} = 5 \). Assume the SC decoders work at \( \text{FER} \approx 10^{-3} \), Table II shows the probabilities when the number of SC-failed packets \( e \) increases from 0 to 4. According to the table, the probability that \( e < 3 \) is 99.9%. Thus, we set the LLR buffer size of the SCL decoder to be 2048 (two packets at maximum), while larger sizes are also allowed.

B. Asymmetric quantization

In a real hardware design, all LLRs are quantized. But the number of quantization bits should be carefully chosen. More quantization bits improves decoding performance, but requires the extra hardware resource. Therefore, we choose the fewest number of bits that incurs negligible performance loss. Moreover, since SCL decoder and SC decoder take different roles in the A2SCL, we propose to employ different quantization widths for them. The scheme is called asymmetric quantization.

First, an SC decoder should be as fast as possible. For an A2SCL decoder, its SC decoding performance can be relaxed to some extent, because the SCL decoder will take care of the failed packets. Typically, longer codes require more quantization bits than short ones. As shown in Fig. 4 the FER curves of \( N = 1024, K = \lfloor 1/8, 7/8 \rfloor \) Polar codes with quantization bits \( [6, 8, 12] \) are almost the same under SC decoding. Accordingly, 6-bits or 8-bits quantization is sufficient for SC decoders.
Second, the SCL decoder should yield almost the same performance as a floating-point decoder. We plot the FER curves of $N = 1024$, $K = [1/8, 7/8]$ Polar codes under SCL decoding ($L = 8$) as reference to show the influence of different quantization bits. As shown in Fig. 5, 8-bits quantization incurs 0.1db loss at maximum, and 12-bits quantization yields the same performance as a floating-point decoder. Accordingly, we adopt 12-bits quantization.

IV. Emulation Platform

An overview of our platform is shown in Fig. 6. A server can manage one or more FPGA boards via the PCI-E slots. When multiple FPGA boards (constrained by the number of PCI-E slots) are employed, the decoding throughput can be further increased.

A Xilinx xc7vx690t is integrated in the FPGA board. The server is the controller of the platform. The code construction (information sub-channel positions) can be configured by the server to evaluate different code constructions. In addition, code and channel parameters are also configured at the server. According to these configurations, frames are generated, encoded, passed through the AWGN channel and decoded in the A2SCL decoder. The number of decoded frames and frame errors are counted in the FPGA and collected by the server. Finally, the FER curve is displayed on the server.

A. Encoder

A random bit stream of length $K$ is generated. As shown in Fig. 7 the frozen bits are set to zero, the CRC and parity check bits are inserted. The pre-coded bits are then fed into a polar encoder.

To achieve high-throughput encoding, every 32 bits are processed in parallel. Specifically, a polar code of length $N$ is split into $N/32$ short codes with length 32. The encoder consists of two parts. At first, a short Polar code of length 32 is encoded and stored into memory. Then, $N/32$ short codes are encoded iteratively with the memory, buffer and the XOR logic. Intermediate results are stored in the buffer. The size of the buffer is half of $N_{\text{max}}$. At most three frames can be stored in the memory. Therefore, frames can be encoded with this pipelined fashion.

B. AWGN channel

The AWGN noise sequence is generated by converting a uniform distributed sequence in the range $[0,1]$ using the inverse cumulative distribution function (ICDF) [14]. A 32-bit hardware random number generator is designed based on a 43-bits linear feedback shift register (LFSR), and a 37-bits cellular automata shift register (CASR) [15]. The cycle length of the combined generator is close to $2^{80}$. 

---

**Fig. 4. SC FER@Different Quantization Bits**

**Fig. 5. SCL ($L = 8$) FER @ Different Quantization Bits**

**Fig. 6. The architecture of the A2SCL emulation platform.**

**Fig. 7. The encoder architecture**
To reduce the mapping table between white uniform noise and white Gaussian noise, we employ 128 line segments to approximate the ICDF. Taking advantage of the symmetry of the ICDF, the mapping table is reduced to 64 starting points, terminal points and slopes. In addition, one multiplier and one adder are required to rebuild the ICDF.

As shown in Fig. 8, 16 AWGN generators with different seeds are combined to provide high throughput. The noise is quantized to 16 bits. Experiment results in Section IV-D show that the quantized noise has negligible effect on FER performance.

C. Run time balancing

In the link-level emulation, different modules have different run time to process one packet. Balancing the run time among different modules will benefit the overall operating efficiency. Table III shows the running cycles required by each module in the link-level emulation platform for different \((N, K)\) case.

![Fig. 8. The Gaussian noise generator architecture](image)

| \((N, K)\) | Encoder | AWGN | SC Decoder | SCL Decoder |
|-----------|---------|------|------------|-------------|
| (1024,512) | 97      | 76   | 221        | 1073        |
| (1024,128) | 97      | 76   | 108        | 506         |
| (512,256)  | 40      | 44   | 105        | 498         |
| (256,128)  | 21      | 28   | 66         | 261         |

![TABLE III](image)

Fig. 9. The emulation time of the FPGA platform vs software implementation.

D. Hardware vs software implementations

To justify the A2SCL hardware platform, its simulation speed and FER performance are compared with a software counterpart. The hardware platform utilizes only one FPGA board. The software implementation is written by C language, and runs on a server that contains 4 Intel Xeon(R) E5-4627 v2@3.30GHz CPUs with 12 cores and 256 GB RAM. For fairness, the data type of the LLR in the software decoder is short.

1) Simulation speed: The emulation time of \(10^7\) frames by the FPGA platform and software implementations are plotted and compared in the Fig. 9. We evaluated code lengths \(N = [256, 512, 1024]\), and rates \(R = [1/8, 1/4, 1/3, 1/2, 2/3, 3/4]\). As seen, the emulation time of the FPGA platform is much shorter than the software implementation on CPUs. When \(N = 1024\) and \(R = 3/4\), the emulation time of the FPGA platform is about 1.275 seconds. However, the CPUs requires 767.9 seconds.

Define the speed ratio (SR) as the emulation time of 12 CPU cores divided by that of one FPGA board, also plotted in Fig. 9. The highest SR is 611, which means that one FPGA board is 611 times faster than 12 CPU cores. Converted to one encoder resource includes the 9 encoder modules. AWGN resource includes the 9 AWGN modules. A2SCL decoder resource includes the 18 8-bit-quantization-SC decoder cores, one 12-bit-quantization-SCL decoder core and the scheduler/mux units that connect them.
CPU core, a FPGA board is 7332 times faster. As shown, the emulation platform can greatly reduce emulation time.

2) FER performance: Based on 5G Polar codes with lengths $N = 1024$ and $N = 800$, we compare the floating-point results from software and fixed-point results from the FPGA platform. Due to the very time-consuming floating-point simulation, software results for $FER > 10^{-6}$ are measured. As shown in Fig. 10, the FER results of the FPGA platform perfectly match the floating-point results under various code lengths and code rates. Note that no error floor is observed from the FPGA platform even when FER results are below $10^{-6}$.

V. PERFORMANCE OF 5G POLAR CODES IN 5G eMBB

Thanks to the FPGA platform, we can now quickly evaluate error-correction performance of 5G Polar codes at FER below $10^{-11}$.

The typical cases of downlink control information (DCI) are evaluated. For $K = 64$, PDCCH aggregation levels $[1, 2, 4, 8]$, the measured FER results are shown in Fig. 11. And we also measured $K = [96, 128, 164]$, PDCCH aggregation levels $[1, 2, 4, 8]$ the results are shown in the Fig. 12, Fig. 13 and Fig. 14.

VI. CONCLUSION

In this paper, we present an asymmetric adaptive SCL decoder in real hardware. Equipped with asymmetric deployment and asymmetric quantization, the decoder can provide much higher decoding throughput in a resource-limited FPGA/AISC chip. The A2SCL algorithm, along with all the required link-level modules, is implemented in an FPGA platform. The platform is efficient, flexible and scalable. The emulation speed of one FPGA board is 611 times as fast as 12 CPU cores; converted to one CPU core, a FPGA board is 7332 times faster. Ultra-low FER performance as low as $10^{-12}$ is measured for

---

Footnote: For $K = 64$, aggregation level 1, the rate matching is shortening; for aggregation level $[2, 4]$, the rate matching is puncturing; for aggregation level 8, the rate matching is repetition.
5G Polar codes for the first time in real hardware.

REFERENCES

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

[2] K. Niu and K. Chen, “Crc-aided decoding of polar codes,” IEEE Communications Letters, vol. 16, no. 10, pp. 1668–1671, October 2012.

[3] H. Zhang, R. Li, J. Wang, S. Dai, G. Zhang, Y. Chen, H. Luo, and J. Wang, “Parity-check polar coding for 5g and beyond,” 2018 International Conference on Communications (ICC), pp. 1–6, May 2018.

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

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

[6] J. Lin and Z. Yan, “An efficient list decoder architecture for polar codes,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 23, no. 11, pp. 2508–2518, Nov 2015.

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

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

[9] B. Li, H. Shen, and K. Chen, “A decision-aided parallel sc-list decoder for polar codes,” arXiv preprint arXiv:1506.02955 (2015), 2015.

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

[11] X. Liu, Q. Zhang, P. Qu, J. Tong, H. Zhang, C. Zhao, and J. Wang, “A 5.16gbps decoder asic for polar code in 16nm finfet,” 2018 15th International Symposium on Wireless Communication Systems (ISWCS), Sep 2018.

[12] V. K. L. Huang, Z. Pang, C. A. Chen, and K. F. Tsang, “New trends in the practical deployment of industrial wireless: From noncritical to critical use cases,” IEEE Industrial Electronics Magazine, vol. 12, no. 2, pp. 50–58, June 2018.

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

[14] R. C. Cheung, D.-U. Lee, W. Luk, and J. D. Villasenor, “Hardware generation of arbitrary random number distributions from uniform distributions via the inversion method,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 15, no. 8, pp. 952–962, 2007.

[15] T. E. Tkacik, “A hardware random number generator,” in International Workshop on Cryptographic hardware and embedded systems. Springer, 2002, pp. 450–453.