Hardware-Efficient Polar Decoder for 5G Internet of Things Communication

Jing Guo(✉) and Congduan Li

School of Electronics and Communication Engineering, Sun Yat-sen University, Shenzhen, China
guoj233@mail2.sysu.edu.cn, licongd@mail.sysu.edu.cn

Abstract. Polar codes have aroused extensive attention due to their capacity-achieving property and low encoding and decoding complexity.

With the increasing demand for real-time and high-quality applications, achieving low-latency communication in resource-constrained scenarios such as on Internet of Things (IoT) devices has become essential. This paper proposes a modified semi-parallel decoder for 5G IoT communication, with low decoding latency and high efficiency of hardware resources. 4-bit decoding algorithm and look-ahead approach are used in this work to reduce latency caused by conventional semi-parallel architecture. For a code length of $N = 2^{10}$, the proposed decoder improves latency by 48.64% and 75.19% than the conventional semi-parallel decoder and 2-bit decoder, separately. The significant improvement in hardware utilization rate of processing elements by 68.42% and 119.35% leads to high efficiency of hardware resources.

Keywords: Internet of things · Polar codes · Wireless communication · Hardware architecture · Field programmable gate array

1 Introduction

The evolving of fifth generation (5G) networks is becoming more readily available as a major driver of the growth of Internet of things (IoT) applications. The challenges for Emerging 5G-IoT Scenarios, such as reliability performance, latency and throughput, cost-efficiency, remain significant [1]. Polar code [2], proposed by Arıkan, is the first forward error-correcting code that achieves the channel capacity of binary-input discrete memoryless channels (B-DMCs). In the 5G enhanced mobile broadband (eMBB) scenarios[3], the polar codes are adopted as the coding scheme for the control channel. Since the encoding and decoding complexity of polar codes are relatively low, low-latency and hardware-efficient polar decoder is suitable for 5G-IoT communication.

Successive cancellation (SC) decoding [2], one of main types of polar codes decoding algorithm, achieves low complexity of O(NlogN). Derived from SC

X. Chen et al. (Eds.): IoTaaS 2023, LNICST 585, pp. 58–68, 2025.
https://doi.org/10.1007/978-3-031-70507-6_5

decoding, successive cancellation list (SCL) decoding [4] and successive cancellation flip (SCF) decoding [5], have been broadly studied for better error-correction performance but with higher complexity. Belief propagation (BP) decoding [6] takes the advantage of low latency as well as high throughput, at the cost of high computation complexity, which is not suitable for resource-constrained applications.

In terms of hardware implementation, SC decoding architectures such as line architecture and tree architecture were proposed in [7]. In [8], a semi-parallel architecture was proposed to reduce processing complexity, which consumed less hardware resources than the prior ones. However, less throughput is performed by semi-parallel architecture than other architectures owing to its increased latency. A partial-sum unit with shift register was proposed in [9], which reduces delay in critical path and improve the frequency of the previous partial-sum unit used in [10]. A look-ahead technique was proposed to accelerate SC decoding in [11] by calculating all the possible outcome of the bits that have not been decoded yet ahead of time, and choosing the appropriate one once the corresponding bit is computed. Multi-bit decoding algorithm and architectures were proposed to decode more than one bit in [12,13]. However, the relative increase in hardware resources consumption is more than the relative decline in latency.

In this work, a novel semi-parallel architecture of polar decoder is introduced. The proposed architecture uses 4-bit decoding algorithm and look-ahead approach to reduce latency and improve the efficiency of hardware resource, which is suitable for resource-constrained devices. The main contributions of our work can be summarized as follows: Our design extends the semi-parallel architecture in [10] to remain high efficiency of hardware resource and reduce latency punishment. The proposed decoder reduces hardware resources by yielding better hardware utilization of processing elements with better latency performance compared to [10,14,15].

The remainder of this paper is organized as follows: Sect. 2 provides an introduction to polar codes and SC decoding algorithm. In Sect. 3, the proposed architecture is presented. Then, the performance and comparisons are discussed in Sect. 4. Section 5 concludes the work.

2 Background

2.1 Polar Codes

Polar codes are operated based on channel combination and polarization theory. Consider an (N,K) polar codeword, where K represents the length of information and N=2^n denotes the length of codeword. The information vector (u_0, u_1, u_2, u_3,..., u_{K-1}) is to be encoded and the rest of $N-K$ bits are defined as frozen bits. K information bits are assigned at the most reliable positions and $N-K$ frozen bits, usually set as '0', are assigned at the rest positions. Then, the encoded vector $X_0^{N-1} = (X_0,X_1,X_2,X_3, ..., X_{N-1})$ can be derived from the information vector u_0^{K-1} by the generator matrix G_N as

$$X_0^{N-1} = u_0^{K-1} G_N, \tag{1}$$

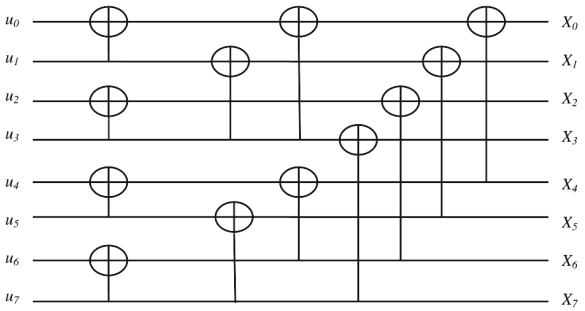

Fig. 1. Encoding graph of polar codes for N=8.

where the generator matrix can be obtained by n times kronecker power of following matrix

$$F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{2}$$

The encoding graph of polar codes for N=8 is shown as Fig. 1.

2.2 Successive Cancellation Decoding Algorithm

SC algorithm decodes the information vector from the log-likelihood ratios (LLRs) $L_{n,i}$. The bit u_i is estimated successively according to the previously decoded bits and LLR values. The process of decoding is illustrated in Fig. 2.

Function f and g can be calculated by the min-sum functions

$$f\left(L_a, L_b\right) \approx \operatorname{sign}\left(L_a\right) \operatorname{sign}\left(L_b\right) \min\left(\left|L_a\right|, \left|L_b\right|\right), \tag{3}$$

$$g\left(\hat{s}, L_a, L_b\right) = L_a(-1)^{\hat{s}} + L_b. \tag{4}$$

Consider $\mathrm{N} = 2^n$ is the polar codes length and let $L_{l,i}$ be the LLRs at each stage. During the decoding process, the decoded vector can be calculated by

$$u_i = \begin{cases} 0 & \text{if } i \epsilon A_c \\ 1 & \text{if } L_{0,i} \geq 0 \\ 0 & \text{otherwise}, \end{cases} \tag{5}$$

where u_i is the decoded bit and $L_{0,i}$ is the LLR of corresponding bit u_i .

3 Proposed Semi-parallel Decoder Architecture

3.1 Over Architecture

The computation part of the proposed decoder mainly consists of processing elements (PEs), which perform g function or f function mentioned before. Due

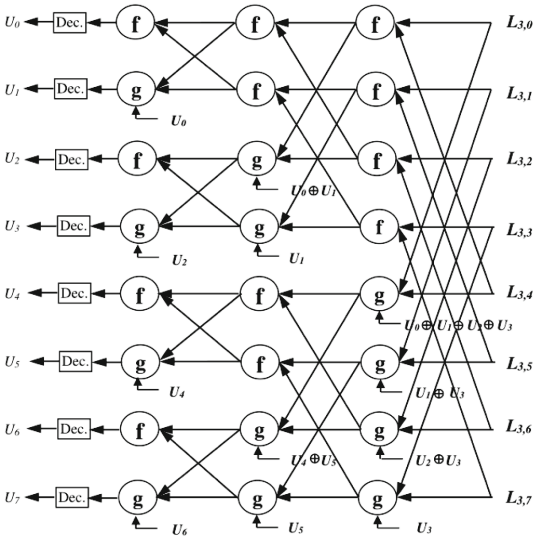

Fig. 2. SC decoding of polar codes for N = 8.

to the sequential nature of SC decoding, some of the PEs are set in idle state during the later stages of decoding, resulting in low utilization of the PEs. The most imperfection that all previous structures confronted is their utilization rate of PEs declines as the code length increases. [10] proposed a semi-parallel architecture to reduce the number of PEs implemented, resulting in higher resource utilization. However, as multiple cycles are required for node updates of LLR values, this leads to increased latency. Complicated multiplexing logic results in complicated control modules between PEs and the rest of the parts, which also increases latency.

In this work, we proposed a novel SC polar decoder with semi-parallel architecture using 4-bit decoding, as shown in Fig. 3. In order to make use of hardware resources, a larger codeword is decoder by PEs lesser than N/2.

The mechanism of the proposed decoder can be summarized as follows. Firstly, the LLRs received from the channel are loaded into the random access memory (RAM), which can store and output data simultaneously, to be further processed. The f node and g node of PEs receive LLRs respectively. Then, The PEs output intermediate LLR values such as minimum, added and subtracted LLRs. Then, these LLRs are put into the corresponding RAMs by multiplexers (MUXs). In the decoding process, f node LLRs and channel LLRs are preserved in one RAM. The rest two RAMs are utilized to preserved added and subtracted LLR values of g node respectively. The LLRs produced by the last but one stage are processed by 4-bit PEs and the output of these PEs are sent to decision unit to output decoded bits. Then, Partial-sum units (PUs) receive decoded bits to output partial sums required for the PEs of g node.

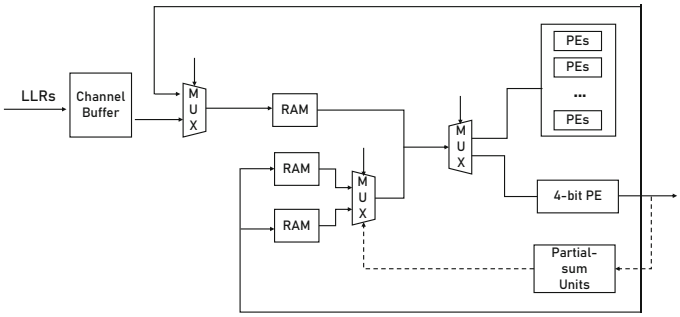

Fig. 3. Overall Architecture of Proposed Decoder.

3.2 Proposed 4-Bit Processing Element

According to Eqs. 3 and 4, 4-input functions can be calculated to be as follows

$$\begin{aligned} ff\left(L_0, L_1, L_2, L_3\right) \approx & \operatorname{sgn}\left(L_0\right) \operatorname{sgn}\left(L_1\right) \operatorname{sgn}\left(L_2\right) \operatorname{sgn}\left(L_3\right) \\ & \cdot \min\left\{\left|L_0\right|, \left|L_1\right|, \left|L_2\right|, \left|L_3\right|\right\}, \end{aligned} \tag{6}$$

$$\begin{aligned} fg\left(L_0, L_1, L_2, L_3, \hat{s}_{00}\right) \approx & \operatorname{sgn}\left(L_1\right) \operatorname{sgn}\left(L_3\right) \cdot \min\left\{\left|L_1\right|, \left|L_3\right|\right\} \\ & +(-1)^{\hat{s}_{00}} \operatorname{sgn}\left(L_0\right) \operatorname{sgn}\left(L_2\right) \cdot \min\left\{\left|L_0\right|, \left|L_2\right|\right\}, \end{aligned} \tag{7}$$

$$\begin{aligned} gf\left(L_0, L_1, L_2, L_3, \hat{s}_{01}, \hat{s}_{11}\right) \approx & \operatorname{sgn}\left(L_0(-1)^{\hat{s}_{01}} + L_2\right) \\ & \cdot \operatorname{sgn}\left(L_1(-1)^{\hat{s}_{11}} + L_3\right) \\ \cdot \min & \left\{\left|\operatorname{sgn}\left(L_0(-1)^{\hat{s}_{01}} + L_2\right)\right|, \left|L_1(-1)^{\hat{s}_{11}} + L_3\right|\right\}, \end{aligned} \tag{8}$$

$$\begin{aligned} & gg\left(L_0, L_1, L_2, L_3, \hat{s}_{10}, \hat{s}_{01}, \hat{s}_{11}\right) = (-1)^{\hat{s}_{10}+\hat{s}_{01}} L_0 \\ & +(-1)^{\hat{s}_{11}} L_1 + (-1)^{\hat{s}_{10}} L_2 + L_3, \end{aligned} \tag{9}$$

where s_{ij} is the i^{th} partial sum located in the j^{th} decoding level.

To reduce the decoding latency, 4-bit PEs are used in the last stage to decode 4 bits simultaneously. The corresponding hardware architecture of 4-bit PEs is illustrated in Fig. 4.

The intermediate nodes can be processed until the computations of corresponding partial sums are done, which is a bottleneck of decoding latency. The partial sum with look-ahead can accelerate the decoding process to compute intermediate nodes before the partial sums are available. This is achieved by calculating all possibilities and choosing the right value at the time the partial sums are produced. The idle PUs can be used for these processes, since not all PUs are occupied during decoding stage. The PU with look-ahead contributes to latency decrease at cost of a little more computation.

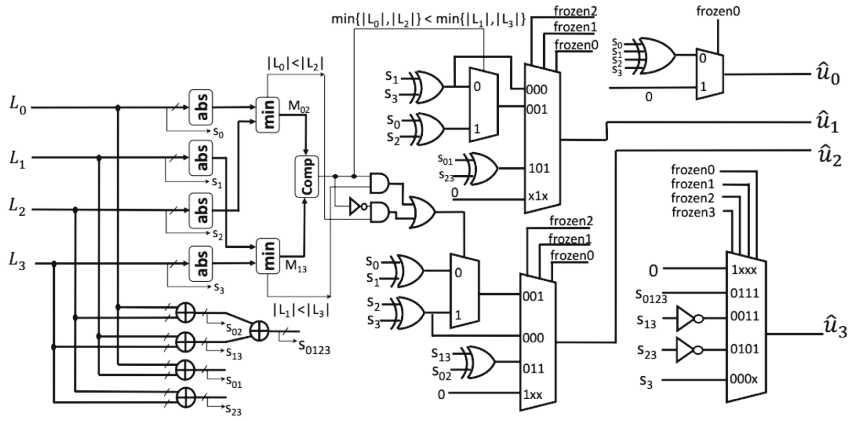

Fig. 4. Architecture of 4-bit PE.

3.3 Pipelined Partial-Sum Unit

Owing to its recursive nature, PU of code length 2^n, as depicted in Fig. 5, consists of 2^{n-1} PUs with extra filp flops (FFs), multiplexers (MUXs) and XOR gates.

Two XOR gates take the critical path of PU, since a XOR is processed 4 bits acquired from 4-bit PE , and the resultant is XOR with previous value $u_0 + u_1$ to generate a partial sum of $u_0 + u_1 + u_2 + u_3$.

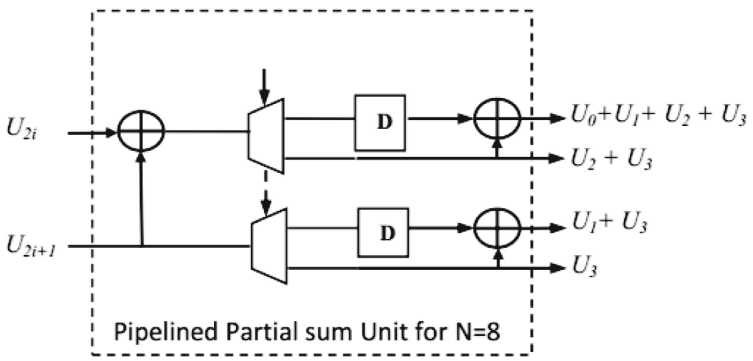

Fig. 5. Partial-sum Unit for N=8.

3.4 Channel Buffer and RAM

Due to limited processing elements used in the proposed architecture, it isn't viable to process all the channel data at the same time for polar codes with

greater length. Thus, the channel LLRs in a gather of P are fed in the RAM, where P is settled as 64 in this work. For a code length of N = 1024, 16 clock cycles are need to load these LLRs into the channel buffer. The RAM is utilized to write the new LLRs and output LLRs to channel buffer at the same time.

3.5 Scheduling Principle

The most imperfection that all previous structures confronted is their utilization rate declines when the code length increases. The utilization rate is defined as the ratio of the whole number of node updates to the product of latency and hardware complexity.

In this work, the utilization rate of PEs can be increased by decreasing the number of PEs. Reduced PEs results in more node updates in each stage and more time to compute intermediate LLRs, resulting in the increase in latency. With the proposed approach, the PEs of last stage are replaced by 4-bit PEs, which decode 4 bits in a single time. Let P be the number of PEs, where P $= 2^p$. Within the stages l where $2^l < P$, the clock cycles is diminished by half, while the rest stages are not influenced. With look-ahead technique applied in partial-sum unit, all the possible values of 4-bit PEs are computed ahead of time. The last stage updated 2^{n-l-3} times when n >3 in a single clock cycle, and the corresponding values are chosen by MUX.

4 Experimental Results

4.1 FER Performance

Error correction performance of the proposed decoder is shown in Fig. 6 for different quantization bits. It is inferred frame error rate (FER) performance is degraded by only 0.25 dB for 5 bit quantization compared to floating point decoder.

The equivalent quantization bits of Q = 5 for the received LLR are given along with necessary inputs to the decoder and the estimated bits are compared with the message vectors for the error correction performance.

4.2 Latency Performance

The decoding latency L of the proposed decoder can be calculated by

$$\begin{aligned} L &= \sum_{l=0} 2^{n-l-2} + \sum_{l=0} 2^{n-l-3} + \sum_{l=1}^{p} 2^{n-l-2} + \sum_{l=p+1}^{n-1} 2^{n-l-2} 2^{l-p} \\ &= \frac{1}{2}N + \frac{N}{4P}(\log_2 \frac{N}{4P} - 1). \end{aligned} \tag{10}$$

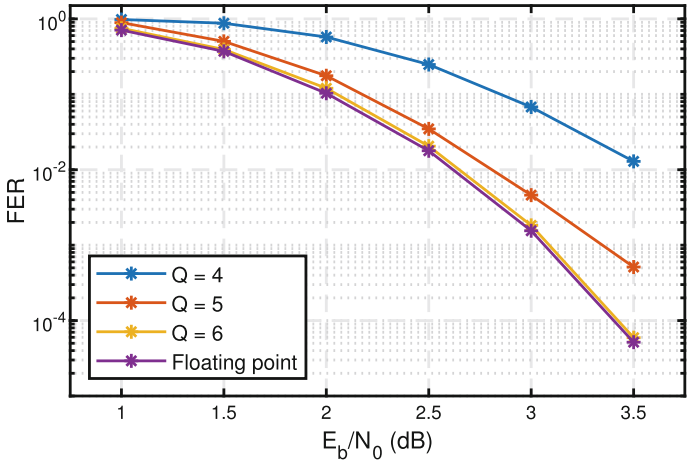

Fig. 6. Effect on FER performance of polar codes.

Figures 7 and 8 present the performance of the proposed decoder and existing SC polar decoders. The conventional semi-parallel SC decoder can be obtained from [10]. [14] presented a scalable SC decoder and [15] presented a 2-bit SC decoder.

According to Fig. 7, it is obvious that our proposed decoder costs much less clock cycles compared to previous decoders. For the length N = 1024, the latency of the proposed decoder is reduced by 48.64% and 75.19% than 2-bit SC decoder and conventional semi-parallel decoder, respectively.

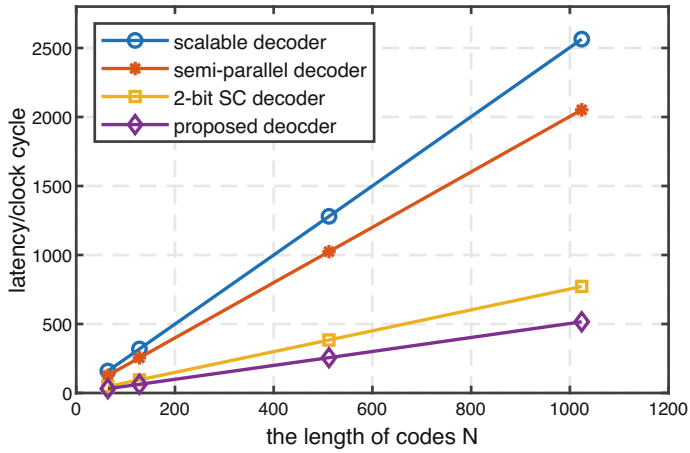

Fig. 7. Latency performance of proposed decoder and existing decoders.

4.3 Hardware Resources Consumption

The computation part of decoder mainly consists of processing elements (PEs). Thus, the efficiency of hardware resources depends on the utilization rate of PEs.

The utilization rate α [10] is defined as the ratio of the whole number of node updates to the product of computation time and computational resource complexity as

$$\alpha \triangleq \frac{\text{total number of node updates}}{\text{computational resource complexity} \times \text{computation time}}. \tag{11}$$

The utilization rate of PEs in this work can be calculated as

$$\alpha = \frac{N \log_2 N}{2P\left(\frac{1}{2}N + \frac{N}{4P}(\log_2 \frac{N}{4P} - 1)\right)} = \frac{2 \log_2 N}{2P + \log_2 \frac{N}{4P} - 1}. \tag{12}$$

Figure 8 indicates that our proposed decoder significantly contributes to the improvement in efficiency of hardware resources. The utilization rate of our decoder increased by 68.42% than semi-parallel decoder and 119.35% than scalable decoder. The utilization rate of 2-bit SC decoder declines with the increase of the length of codes, since the tree architecture used in 2-bit SC decoder and other conventional decoders require more PEs for computation. While semi-parallel and scalable architecture utilize relatively fixed PEs.

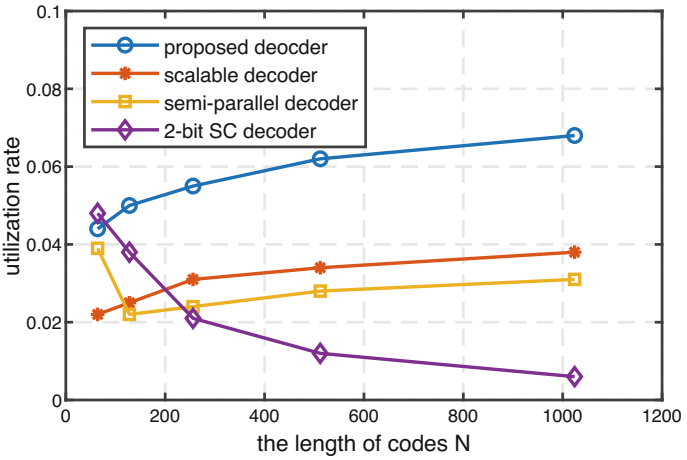

Fig. 8. Utilization rate of the proposed decoder and existing decoders.

All these four polar decoders for a code length of N=1024 with 5-bit quantization are implemented using Xilinx Kintex-7 FPGA. Table 1 shows the hardware resource consumption of conventional semi-parallel decoder, scalable decoder, 2-bit SC decoder and the proposed decoder. LUT refers to Look-up table and

FF denotes flip flop, which are the fundamental resources of FPGA. It can be concluded that the proposed decoder utilizes less hardware resources like LUTs and FFs than existing decoders. While its RAM consumption is higher than previous semi-parallel decoder and 2-bit SC decoder because of extra storage required by look-ahead partial-sum units.

Table 1. The hardware resource consumption of decoders.

Decoder	LUT	FF	RAM(bits)
Conventional semi-parallel decoder	4,130	1,691	15,104
Scalable decoder	2,866	1,304	41,648
2-bit SC decoder	7,432	972	14,324
Proposed decoder	2,544	830	31,104

5 Conclusion

In this paper, we have introduced a hardware-efficient architecture for polar decoder by modifying semi-parallel architecture. With reduced PEs, its hardware utilization rate is improved significantly. The proposed architecture utilizes 4-bit decoding and look-ahead technique to reduces latency. The results show a higher efficiency of hardware resources and lower latency than previous decoders. For a code length of $N = 2^{10}$, the proposed decoder improves latency by 48.64% and 75.19% than the conventional semi-parallel decoder and 2-bit decoder, separately. The significant improvement in hardware utilization rate of processing elements by 68.42% and 119.35% leads to high efficiency of hardware resources.

Acknowledgement. This work was supported by the National Science Foundation of China (NSFC) with grant no. 62271514 and the Science, Technology and Innovation Commission of Shenzhen Municipality with grant no. JCYJ20210324120002007, and ZDSYS20210623091807023.

References

1. Shafique, K., Khawaja, B.A., Sabir, F., Qazi, S., Mustaqim, M.: Internet of things (IoT) for next-generation smart systems: a review of current challenges, future trends and prospects for emerging 5G-IoT scenarios. IEEE Access **8**, 23022–23040 (2020)
2. Arikan, E., Telatar, E.: On the rate of channel polarization. In: 2009 IEEE International Symposium on Information Theory, pp. 1493–1495. IEEE (2009)
3. Notes, C.: 3GPP TSG ran WG1 meeting no. 87, chairmans notes of agenda item 7.1. 5 channel coding and modulation (2016)
4. Tal, I., Vardy, A.: List decoding of polar codes. IEEE Trans. Inf. Theory **61**(5), 2213–2226 (2015)

5. Dai, B., Gao, C., Yan, Z., Liu, R.: Parity check aided SC-flip decoding algorithms for polar codes. IEEE Trans. Veh. Technol. **70**(10), 10359–10368 (2021)
6. Arıkan, E.: Polar codes: a pipelined implementation. In: Proceedings of 4th ISBC vol. 2010, pp. 11–14 (2010)
7. Leroux, C., Tal, I., Vardy, A., Gross, W.J.: Hardware architectures for successive cancellation decoding of polar codes. In: 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1665–1668. IEEE (2011)
8. Pamuk, A.: An FPGA implementation architecture for decoding of polar codes. In: 2011 8th International Symposium on Wireless Communication Systems, pp. 437–441. IEEE (2011)
9. Berhault, G., Leroux, C., Jego, C., Dallet, D.: Partial sums generation architecture for successive cancellation decoding of polar codes. In: SiPS 2013 Proceedings, pp. 407–412. IEEE (2013)
10. Leroux, C., Raymond, A.J., Sarkis, G., Gross, W.J.: A semi-parallel successive-cancellation decoder for polar codes. IEEE Trans. Signal Process. **61**(2), 289–299 (2012)
11. Zhang, C., Yuan, B., Parhi, K.K.: Reduced-latency SC polar decoder architectures. In: 2012 IEEE International Conference on Communications (ICC), pp. 3471–3475. IEEE (2012)
12. Hassan, H.G., Hussien, A.M., Fahmy, H.A.: A simplified radix-4 successive cancellation decoder with partial sum lookahead. AEU-Int. J. Electron. Commun. **96**, 267–272 (2018)
13. Yuan, B., Parhi, K.K.: Low-latency successive-cancellation polar decoder architectures using 2-bit decoding. IEEE Trans. Circuits Syst. I Regul. Pap. **61**(4), 1241–1254 (2013)
14. Raymond, A.J., Gross, W.J.: A scalable successive-cancellation decoder for polar codes. IEEE Trans. Signal Process. **62**(20), 5339–5347 (2014)
15. Babu, G.S., Madala, L.R., Gopalakrishnan, L., Sellathurai, M.: Low-complex processing element architecture for successive cancellation decoder. Integration **66**, 80–87 (2019)