Priority Queue VLSI Architecture for Sequential Decoder of Polar Codes

Aleksei E. Krylov, Andrey V. Rashich, Dmitrii K. Fadeev, and Kirill A. Sinjutin

Abstract—The VLSI architectures for stack or priority queue (PQ) are required in the implementation of stack or sequential decoders of polar codes. Such type of decoders provide good BER performance keeping complexity low. Extracting the best and the worst paths from PQ is the most complex operation in terms of both latency and complexity, because this operation requires full search along priority queue. In this work we propose a low latency and low complexity parallel hardware architecture for PQ, which is based on the systolic sorter and simplified sorting primitives. The simulation results show that just small BER degradation is introduced compared to ideal full sorting networks. Proposed PQ architecture is implemented in FPGA, the synthesis results are presented for all components of PQ.

Index Terms—Polar codes, polar codes decoding, priority queue, stack, sequential decoder, stack decoder, hardware architecture.

I. INTRODUCTION

Polar codes were proposed by Arıkan [1] in 2009 and are already utilized in 5G NR standard as a FEC scheme for control channels [2]. The main reasons for such rapid adoption in industry are that polar codes are first to achieve channel capacity and still have explicit algorithms for code construction, encoding and decoding.

Paper [1] also proposed Successive Cancellation (SC) decoder that exploits polarization transform features to find solution. The main drawbacks of this decoding algorithm are low correction capability for codes of practical lengths and sequential nature of decoding process. The latter one causes high decoding latency that linearly depends on codeword length N , since SC decoder constructs solution codeword bit-by-bit by traversing one path in a binary decoding tree.

In order to override first drawback, Successive Cancellation List (SCL, [3]) and Successive Cancellation Stack (SCS, [4], [5], [6]) decoders were proposed. SCL decoder constructs L best paths in parallel and tend to maximum likelihood decoding while L increase. The cost is higher computational complexity. SCS decoder uses the same best path search width as SCL, but uses a sorted stack in order to find and continue only the one best path in the stack during each iteration. This approach results in lower computational complexity with error correction capability being the same, but greatly increases decoding latency, compared to SCL. The sorter is main bottleneck of SCS decoder, since it may become quite complex.

The research was funded by the Ministry of Science and Higher Education of the Russian Federation as part of World-class Research Center program: Advanced Digital Technologies (contract No. 075-15-2020-934 dated 17 November 2020).

The authors are with the Peter the Great St.Petersburg Polytechnic University, 195251, Russia, St. Petersburg, Polytechnic st. 29.

There are several other proposals that aim SCL decoder complexity reduction by unnecessary calculations elimination. For example in [7] authors propose a hybrid scheme, where many separate SC decoder cores work in parallel, but in case of failed CRC-check in one of them this decoder bank switch to SCL mode. To flatten decoder throughput, extra input buffer is used. Paper [8] presents an input distribution-aware SCL decoder, that disables some of SC processing cores for “good” codewords decoding, thus dynamically changing the list size in order to reduce energy consumption. Other papers [9], [10], [11] also present SCL decoders that adjust list size, in order to have good error correction capability of SCL on one hand and lower average computational complexity on the other.

All these tricks do reduce either average complexity and/or energy consumption, but they are not capable to reduce chip area, and similar to SCS decoder, result in higher decoding latency. Also these solutions give less flexible architectures and require extra control logic to switch between list sizes and modes.

Sequential (SQ) decoder [12], [13] could be seen as a modification of SCS decoder that uses several ingenious improvements in order to reduce decoding latency. The basic hardware architecture of sequential decoder is proposed in [14]. One of the most important blocks in SQ decoder is the Priority Queue (PQ) or stack in terms of SCS decoder. This block contains partial paths, and during each decoder iteration it outputs one best path and thus this block delay is an essential part of overall decoding delay.

To the best of our knowledge, no papers about SCS decoder provide detailed algorithm of stack processing or hardware architecture, especially for the best path extraction operation. In this paper we propose a new hardware architecture for PQ, which is based on the folded systolic sorter, and provide its detailed description and analysis in terms of complexity and decoder BER performance. The PQ sorter is based on the systolic sorting algorithm presented in [15].

The rest of this paper is structured as follows. The SQ decoding algorithm and general SQ decoder structure are described in Section II. The details of proposed PQ architecture are given in Section III. Sequential decoder performance analysis is shown in Section IV. Section V deals with FPGA implementation. Finally, the conclusions are presented in Section V.

II. POLAR CODES SEQUENTIAL DECODER

A. Sequential Decoding Algorithm

SC decoder was proposed by Arıkan in [1]. In logarithmic domain this algorithm uses two functions – namely Q -function

and P function (Eq. 1) to traverse the decoding tree [14]. Decoding tree leafs are referred to codeword bits, and thus traversing the tree correspond to bit-by-bit successive decoding. In (Eq. 1) a and b are LLR values, taken from other functions' outputs or from channel, ps is the partial sum, calculated from decoded bits. Each decoder iteration it makes one-bit step along the codeword, obtaining this bit estimation and incrementing the decoding phase ϕ . In this case only one path (partial solution codeword) is considered and no memory is required. In case if error occurs in one bit, it won't be corrected and could result in errors in some following bits in current codeword.

$$\begin{aligned} Q(a, b) &= \text{sgn}(a)\text{sgn}(b)\max(|a|, |b|); \\ P(ps, a, b) &= (-1)^{ps} \cdot a + b. \end{aligned} \quad (1)$$

Sequential decoder, proposed in [12], keeps N_{PQ} paths with different lengths in PQ. During each iteration PQ outputs the best path in terms of metric and this path is extended using operations in Eq. 1. If current phase ϕ refers to a frozen bit, the path does not split and goes back to PQ after metric update. In case of non-frozen bit the path has two possible continuations, metrics are calculated for both of them, and then they are stored in PQ.

Path metric function $M(v_0^\phi, y_0^{N-1})$ (Eq. 2) includes precalculated bias function $\Psi(\phi)$ [16] and the total penalty of path $v_0^{\phi-1}$, consisting of penalties τ (Eq. 3) from all ϕ decoded bits.

$$M(v_0^\phi, y_0^{N-1}) = \sum_{i=0}^{\phi-1} \tau(S_{m,i}(v_0^{i-1}|y_0^{N-1}), v_i) - \Psi(\phi). \quad (2)$$

$$\tau(S, v) = \begin{cases} 0, & \text{sgn}(S) = (-1)^v; \\ -|S|, & \text{otherwise.} \end{cases} \quad (3)$$

In order to bound the computational delay, both SCS and SQ decoders use the search width parameter L . Decoder sets counters q_ϕ , which indicate how many paths have passed phase ϕ . If some counter q_i becomes equal to L , than all paths with phase less than i have to be deleted from PQ. This parameter has the same sense, as the list size in SCL decoder, but in SQ decoder it limits the computational delay instead of decoder area.

In SQ decoder, as well as in the stack one, the number of decoding iterations depends on the input LLR vector and could vary in vast range. The main cause is that longer paths on average always have worse metric compared to shorter ones, just because they had more opportunities for metric decrease through decoding. Thus decoder tends to return to shorter paths prolonging unreasonably often. The bias function $\Psi(\phi)$ from Eq. 2 was suggested in [17] in order to solve this imperfection by increasing longer path metrics and reducing the mean number of iterations.

As a result, with bias function mean number of iterations tend to that value for SCL decoder, but distinct values of this parameter may be several times higher. The bias function values are obtained through modelling and depend on the signal-to-noise ratio.

B. General SQ Decoder Structure

In most general view the SQ decoder consists of two main subblocks, the first one is Metric Processor (MP), which implements operations from Eq. 1 to calculate path metrics on each iterations. MP in [14] uses fully parallel structure in order to reduce the latency of one iteration and to avoid intermediate result storage that is proposed in [13].

The second subblock is Priority Queue that stores and sorts candidate paths and provide input data to MP during every iteration. PQ stores all needed information about each path, sort the paths according to their metrics and output one best path each iteration. PQ delay for sorting strongly affects the overall decoding delay, because MP latency is usually less than that of sorting operation.

In this paper we do not dive into MP implementation details, but we could formulate several demands that MP poses to PQ:

- MP and PQ operate in parallel;
- each iteration PQ has to provide the best and the worst paths in terms of metric for continuation and deletion respectively;
- each iteration MP processes the best path from PQ;
- each iteration one or two paths enter the PQ, and new search is needed for the next iteration to find the path with the best metric;
- PQ stores N_{PQ} paths themselves, their metrics and current phases;
- PQ has to delete paths, shorter than ϕ , if some counter q_ϕ becomes equal to L .

III. PROPOSED PRIORITY QUEUE ARCHITECTURE

Since the paths themselves are binary vectors of length up to N , it is preferable to store them in RAM. In such case the PQ splits to several blocks, presented in Fig. 1.

Fig. 1. Sequential decoder PQ structure.

Path RAM stores paths – binary vectors that represent parts of possible solution codewords and has classical dual port RAM architecture. q_ϕ Counters block consists of a set of counters. It outputs ϕ_{min} value, all paths with phase less than ϕ_{min} are deleted from Path RAM and Sorter blocks.

Early output (EO) block is inserted in order to reduce PQ delay. PQ finds overall best path among those stored in RAM and those two paths that just arrived from MP. Since it is quite often that the best path is one of the newly arrived, PQ does not need to save it to the RAM, but just forward it to the MP for further processing. This situation is very common for “good” words, which are weakly corrupted with noise. Thus this block compares three paths to find the best one, and forward it to MP. Two other paths are forwarded into the RAM (one of them could be already there).

Sorter block is the most crucial one in PQ, since its performance and complexity are highly significant for those of whole decoder. In this paper we use sorter structure based on the algorithm from [15] which results in parallel architecture, depicted in Fig. 2. The main advantage of this algorithm and the architecture is very low delay for contents resorting if only two elements change their values at a time and only one best element is needed after resorting.

Fig. 2. Proposed PQ sorter architecture.

In case of SQ decoder hardware implementation, Metric Processor block requires several clock cycles for metric calculation and during its processing the PQ needs to perform one iteration of sorting and provide result to EO block. Two layers of Compare and Select blocks from Fig. 2 require only one or two clock cycles for processing, due to their simple structure. Together with reading data from RAM, this means that Fig. 2 direct implementation requires 2 (or 3) clock cycles for processing. For codes of length 512 and higher this delay is less than that of MP block.

Fig. 3. CAS3 block structure.

Proposed architecture uses a set of universal 3-input sorting blocks (*CAS3*, depicted in Fig. 3, which rearrange 3 inputs into a sorted set of 3 outputs using metrics. Since the *CAS3* output must have maximum metric input on the first output, we propose to let other two outputs remain unsorted. Such simplification greatly reduces the number of resources for *CAS3* block and increases its scalability, while BER performance keeps almost unchanged (Figs. 5, 6). *CAS2* blocks have trivial structure of one metric comparator and multiplexers for all bus elements.

Phase check blocks (PCB) from Fig. 4 are used on each sorter input to reset sorter contents when new word decoding starts and to compare path phases with current ϕ_{min} value and delete the paths with lesser phases, reducing the average number of decoder iterations. In order to reduce complexity the latter function is performed on N_{PCB} upper sorter inputs. In Section IV we provide results for various N_{PCB} values and show that in most cases, this value could be much less than N_{PQ} .

The path deletion is done by forcing the path metric to minus infinity, i.e. $-\infty$ to minimum possible negative value for the chosen fixed-point bit widths. In case of sign-magnitude arithmetics forcing is done by setting maximum possible positive value. Since paths after deletion remain in sorter until they get to its bottom, decoder performance degradation occurs, compared to full sorter output.

The systolic sorter requires a number of iterations to find the best and the worst paths, whilst these outputs are to be provided on every iteration. Not true solution may be output at first several iterations after paths deletion using phase counters mechanism. However, decoder performance degradation, caused by this errors, is negligible (Fig. 5).

In order to further reduce resources consumption we propose the folded sorter (Fig. 4), which uses only half of *CAS3* blocks from original unfolded one (Fig. 2). This one layer of *CAS3* blocks is used twice for one iteration of sorting. This leads up to 1 clock cycle delay increase, but as long as the overall PQ delays is lower than SP delay, this does not reduce decoder throughput.

Fig. 4. Proposed hardware PQ sorter architecture

One could further reduce the number of *CAS3* blocks, folding it vertically, but this will require much more complex multiplexing and control logic and thus resources consumption gain will be lower. Another option is using one-layer sorter and giving extra clock cycles for its processing to make more than one sorting iteration during one decoding iteration. In case of q_i phase pass counters usage, this will reduce the performance degradation compared to ideal sorter.

IV. PERFORMANCE ANALYSIS

In this section we provide the simulation results for performance of sequential decoder equipped with the proposed PQ. The code under consideration is the polar subcode from [18] with $N = 512$, $R = 1/2$ and 24 dynamic frozen bits. The

size of PQ is 256. The performance is analyzed in terms of BER, FER and average number of decoder iterations. Also we compare the performance of the proposed PQ with the one of PQ, which uses ideal sorter. Here "ideal" means that the full sorting of all paths in PQ is done on each iteration, and this operation does not require any clock cycles.

The BER and FER performance of sequential decoder with various modifications of proposed PQ is shown on Fig. 5. A number of cases is considered:

- the performance of decoder with an ideal sorter in PQ for cases, when heuristic function is applied or not applied (for reference);
- the performance of decoder with the proposed PQ, which uses the systolic sorter for various number of PCB blocks.

There is quite small BER and FER degradation of less than 0.05 dB of decoder with proposed PQ architecture for high and low SNR values compared to the case of PQ with ideal sorter. Also the error rate performance does not depend greatly on the number of PCBs for the considered range $N_{PCB} = 6 \dots 256$.

Fig. 5. Sequential decoder BER and FER performance

The Fig. 6 shows the average number of iterations for different variants of proposed PQ. First of all we note dramatic decrease of iterations number, when the heuristic function is applied, as it was introduced in [16]. Moreover the proposed PQ architecture introduces approximately the same results as for the PQ with ideal sorter. Only little degradation ($< 1\%$) in iterations number can be seen for low SNR values. An important fact here is that the number of iterations does not depend on the number of PCB blocks for the considered range. This allows to use just 6 PCB blocks, reducing the complexity of PQ.

Note that for high SNR values the average number of iterations is less than the code length 512. We propose to start decoding process not from phase 0, but from the phase, which corresponds to the first non frozen element. This allows to slightly reduce the average number of iterations.

V. HARDWARE IMPLEMENTATION OF PROPOSED PQ ARCHITECTURE

The proposed PQ architecture was implemented in RTL using VHDL language. The main target for development is ASIC prototyping. Thus no registers were used for additional

Fig. 6. Average number of iterations

pipelining, as it is done in FPGAs. Also the preliminary decoder synthesis was made in FPGA Xilinx Ultrascale XCKU060 for verification and testing. The Vivado synthesis was done for clock frequency 250 MHz. The synthesis results for various number of PCB blocks and path metric bit widths $BW_M = 8, 10$ are shown in Table I.

TABLE I
PQ SYNTHESIS RESULTS IN XILINX ULTRASCALE+ XCKU15P

PCBs	$BW_M = 8$			$BW_M = 10$		
	LUT	FF	BRAM	LUT	FF	BRAM
6	13671	6739	8	18690	7257	8
60	14294	6739	8	19662	7257	8
128	15103	6739	8	20887	7257	8
255	16630	6739	8	20930	7257	8

In general the total resource usage depends on the PQ length. The block memory consumption does not depend on the complexity of sorter and is defined only by the PQ size. The block memory stores only the decoded bits of each path, whilst paths phases and metrics are stored inside the systolic sorter. The number of flip-flops depends on the PQ size and path metric bit width as the registers are placed at the Systolic Sorter outputs. The main logic consumption is introduced by the systolic sorter's Max Units. There number is proportional to PQ size. The PCB blocks also require additional logic but their number could be quite small (up to 6), still providing very small BER degradation (Fig. 5).

CONCLUSION

This paper presents the hardware architecture of priority queue, which consists of early output logic, RAM and systolic sorter. It ensures low latency to output the best and the worst path at each decoder iteration. The folded systolic sorting approach combined with simplified basic sorting units provide low resources consumption. The simplification of basic sorting units causes BER degradation less than 0.05 dB for BER level 10^{-4} , and the average number of iterations is decreased by less than 1%. The proposed architecture can be used in stack or sequential decoders of polar codes.

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] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; NR; Multiplexing and channel coding, document TS 38.212, V16.05.0, 3GPP, mar 2021.
- [3] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.
- [4] K. Niu and K. Chen, "Stack decoding of polar codes," *Electronics Letters*, vol. 48, no. 12, pp. 695–697, June 2012.
- [5] K. Niu and K. Chen, "Crc-aided decoding of polar codes," *IEEE Communication Letters*, vol. 16, no. 10, May 2012.
- [6] W. Song, H. Zhou, K. Niu, Z. Zhang, L. Li, X. You, and C. Zhang, "Efficient successive cancellation stack decoder for polar codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2608–2619, 2019.
- [7] Y. Wang, Q. Wang, Y. Zhang, S. Qiu, and Z. Xing, "An area-efficient hybrid polar decoder with pipelined architecture," *IEEE Access*, vol. 8, pp. 68 068–68 082, 2020.
- [8] C. Condo, "Input-distribution-aware successive cancellation list decoding of polar codes," *IEEE Communications Letters*, pp. 1–1, 2021.
- [9] P. Giard, A. Balatsoukas-Stimming, T. C. Müller, A. Bonetti, C. Thibault, W. J. Gross, P. Flatresse, and A. Burg, "Polarbear: A 28-nm fd-soi asic for decoding of polar codes," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 7, no. 4, pp. 616–629, 2017.
- [10] X. Liu, Q. Zhang, P. Qiu, J. Tong, H. Zhang, C. Zhao, and J. Wang, "A 5.16gbps decoder asic for polar code in 16nm finfet," in *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*, 2018, pp. 1–5.
- [11] J. Tong, H. Zhang, L. Huang, X. Liu, and J. Wang, "An asymmetric adaptive scl decoder hardware for ultra-low-error-rate polar codes," in *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*, 2019, pp. 532–536.
- [12] V. Miloslavskaya and P. Trifonov, "Sequential decoding of polar codes," *IEEE Communication Letters*, vol. 18, no. 7, pp. 1127–1130, July 2014.
- [13] G. Trofimiuk, N. Iakuba, S. Rets, K. Ivanov, and P. Trifonov, "Fast block sequential decoding of polar codes," 2020.
- [14] A. Krylov, A. Rashich, A. Gelgor, and D. Fadeev, "Polar codes sequential decoder hardware architecture," in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, 2019, pp. 261–264.
- [15] P. Lavoie, D. Haccoun, and Y. Savaria, "A systolic architecture for fast stack sequential decoders," *Communications, IEEE Transactions on*, vol. 42, pp. 324 – 335, 03 1994.
- [16] P. Trifonov, "A score function for sequential decoding of polar codes," in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 1470–1474.
- [17] G. Trofimiuk, N. Iakuba, S. Rets, K. Ivanov, and P. Trifonov, "Block sequential decoding techniques for polar subcodes," *CoRR*, vol. abs/1808.07268, 2018. [Online]. Available: <http://arxiv.org/abs/1808.07268>
- [18] P. Trifonov and V. Miloslavskaya, "Polar subcodes," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 254–266, Feb 2016.