A Resource Efficient Decoder for Polar Code Based on FPGA

Chenguang Yang[1,*], Huibin Zhang[2]

[1,2]School of Electronic Engineering, Beijing University of Posts and Telecommunications

[1,*]yang545815441@126.com, [2]zhanghuibin@bupt.edu.cn

Abstract. A resource efficient hardware implementation method of polar decode is presented in this paper. It supports the configuration of bit allocation table with various code lengths and rates. The hardware implementation based on Successive Cancellation List Decoding (SCL) scheme makes BER (Bit-Error-Rate) performance more excellent, and the pipe-lined and paralleled design achieved a good balance between performance and complexity, maintaining a high throughput while consuming less resources. The method is implemented on FPGA and has excellent performance. Compared with other hardware implementation methods, the method can achieve the same throughput with less hardware resource.

Keywords: polar code, low resource consumption, successive cancellation list decoding, FPGA implementation

1 Introduction

Polar code is the first coding method that can be strictly proved to reach the Shannon limit. The concept of polar code was first proposed by Arikan in 2009 [1], which is the major research in the field of channel coding in recent years. It has been widely concerned and studied since it was proposed, and has been applied to actual projects [2,3].

With the development of polar code research and the expansion of application scenarios, hardware implementation for polar code has attracted much attention. To perform hardware implementation, resource utilization and throughput are important indicators to measure performance. A polar encoding method with low hardware resource utilization is proposed in [4], but the supported code length is limited to N=8, and the implementation of the decoding method is not described in detail. [5] proposed a hardware implementation method of polar code SCL decoding with large list capacity, which supports long code length of N=4096 and large linked list decoding with L=32. The bit error rate performance is better, but not suitable for other code length, and the latency and throughput are not optimized, the large linked list also leads to large resource occupation. For the common code length format N=1024, [6] proposed an emulator platform, including encoder, channel model and decoder. [8] proposed a method

H. Wu et al. (Eds.): CSIEDE 2022, ACSR 103, pp. 774–781, 2022.
https://doi.org/10.2991/978-94-6463-108-1_86

to support different code length. However, these researches only focus on the implementation and the resource utilization is not further optimized, which leads to the large consumption of hardware resources.

Therefore, this paper presents a hardware implementation of polar decoding with low resource consumption. The paper is structured as followed: decoding theory of polar codes is proposed in section II, the hardware implementation of decoding is followed in section III, section IV underlines hardware results and performances and the comparisons with other literature.

2 Decode Theory for Polar Code

Successive Cancellation (SC) decoding method is the basic decoding method for polar code, also the foundation for Successive Cancellation List (SCL) decoding method. SC can be performed by the following steps.

Polar decode using the soft value log-likelihood ratio (LLR) to perform decoding. According to received LLR, the F function can be performed to calculate and judge information bit u_1 by equation (1) and (2).

$$f(L_1, L_2) = \ln \frac{1+e^{L_1+L_2}}{e^{L_1}+e^{L_2}} \approx \text{sign}(L_1)\text{sign}(L_2)\min\{|L_1|, |L_2|\} \tag{1}$$

$$u_1 = \begin{cases} 0, f(L_1, L_2) \geq 0 \text{ or u is frozen bit} \\ 1, f(L_1, L_2) < 0 \end{cases} \tag{2}$$

The G function can be performed with u_1 as the input value to judge u_2 by equation (3) and (4).

$$g(U_1, L_1, L_2) = \ln \frac{\Pr(Y_1 Y_2 U_1 | U_2 = 0)}{\Pr(Y_1 Y_2 U_1 | U_2 = 1)} = (1 - 2U_1)L_1 + L_2 \tag{3}$$

$$u_2 = \begin{cases} 0, g(U_1, L_1, L_2) \geq 0 \text{ or u is frozen bit} \\ 1, g(U_1, L_1, L_2) < 0 \end{cases} \tag{4}$$

After that, the decoding process for basic decoding unit is done, 2-bit polar code is decoded. For polar code with length 2^n ($n \geq 1$), the decoding process will be divided into different stages, and the partial sum process needs to be done, which means the lower decoded bit needs to be transmit into higher rank for the calculation in the higher stage as the G-function input.

Based on SC, the SCL decoding method is proposed to achieve a better performance. It introduces Path Metric (PM) which refers to the posterior probability of a decoding result. The larger the value, the higher the probability of correct decoding result will be, and the probability of obtaining the final correct decoding result by continuing to use SC decoding method along with this result will be higher. SCL decoder including K SC decoder inside which are placed in parallel. The first decoder performs operations with SC method until ruling out the first information bit, then the Kth decoder will copy all the intermediate data values and LLR values from the first decoder, and the newly decoded information bit for the original SC is 0, for the new SC is 1, which means retaining two kinds of results as different path. Thus, K different decoding results are

generated, and the decoding result with the minimum PM value can be selected as the output of SCL decoder.

3 Hardware Implementation

According to SCL decoding scheme, the basic decoding Processing Unit (PU) of this design is a SC decoding module, which realizes three functions of SC decoder: F function, G function and partial sum function. The structure of PU is shown by the FIG.1.

When fsel=0, the module will perform F function with the current two input LLR (llr_a and llr_b), and output calculated LLR (llr_out) as the input of the next rank. When fsel=1, the control module will give the partial sum (PSUM) value from the low-rank operation module to the current module, which will be used as the input of the post-decision bit, and realize the reuse of the module through the different input.

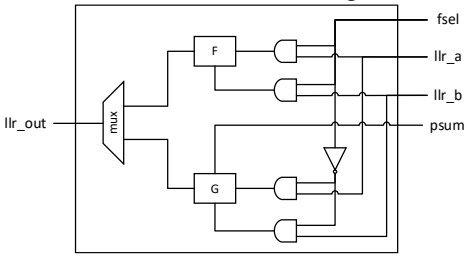

Fig. 1. Processing Unit

Based on PU, the decoder module Process Unit Tree (PUT) is constructed. In order to cope with multiple code lengths and ensure the efficiency, the input will be divided into three layers. A code with length 2^n can be divided into n ranks. The first layer is from the input to the rank 7, the second layer is from the rank 7 to the rank 4, and the third layer is below the rank 4.

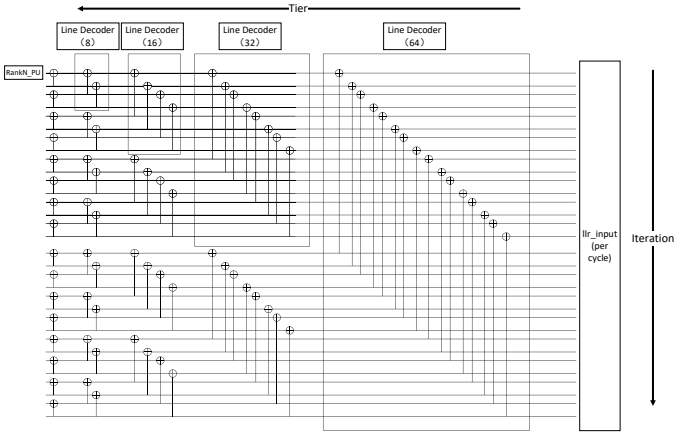

Fig. 2. PUT on layer 1

FIG.2 shows the decoding structure for the first layer. Every 128 bit LLR input from top to bottom is an iteration. From right to left is each tier of the first layer. Each iteration and the operation between each stage need to consume a cycle. When decoding from the first layer to the second layer, the largest line decoder (64 PUs) will start first, which means that if the code length N=2048, the input of the second layer will after the fourth tier; For code length N=1024, it is after the third tier; For code length N=512, it is after the second tier; For the code length N=256, it is after the first tier. In the case of code length N=128, PUT will directly perform the second layer processing.

Since PUT requires multiple operations to fill rank 7 LLR memory with enough LLRs to perform rank 7 to rank 4 processing, iteration operation is necessary. For example, when N=2048, the layer 2 input LLR uses the smallest line decoder to decode, and 8 outputs are generated through 8 decoding units. Therefore, 16 iterations are required to complete the 128 LLRs required for the second layer; When N=1024, the size of the line decoder adopted by the layer 2 input is 16, so 8 iterations are required to meet the needs of the second layer. The smaller the code length is, the fewer iterations are required to reach rank 7.

The calculation of the second layer will be consistent with that of the first layer. The line decoder adopted is the first three layers (64, 32, 16), and then the output results will be sent to the third layer for minimum order decoding operation.

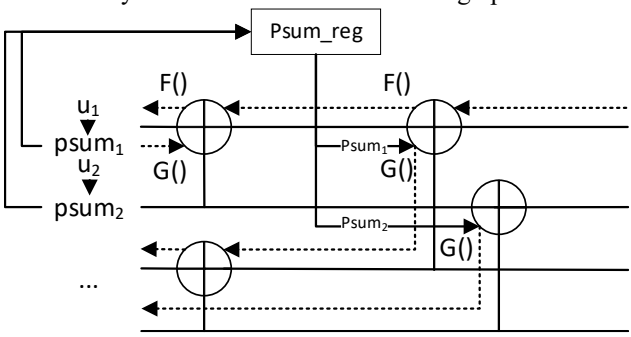

Fig. 3. Basic unit for layer 3

FIG.3 shows the rank 2 structure in layer 3. The upper layer inputs from first rank (rank 4) of F function to the child node (rank 1) to generate the first decoding bit, which will be returned to the current decoder as the input bit of G function, and the PSUM value and the LLR value of intermediate operation will be stored in the register for input to the upper rank SC decoding unit for G function. After the current chuck finishes decoding, the output PSUM can give the next chuck as the input to perform the decoding operation of the next chuck. When the current rank completes the operation, it continues to return the results to the upper layer, and perform the G function and partial sum function by means of back propagation.

With all of the SC results, the SCL decode can be performed. In the decoding structure of the third layer the list size L is 8, so it can be considered that there are 8 SC decoders inside, and all decoders have the same initial value, which is the input value

of the upper layer. The first decoder performs operations and assigns values synchronously to other decoders until the first information bit is decided. After the first bit calculated, all the intermediate data values with PSUM and LLR calculated by the first decoder are given to the eighth decoder, and the information bit obtained by the original decoder path is set to 0, and the information bit of the new path is set to 1, that is, the two results of information 0 and 1 are kept simultaneously. And the two activated decoders will continue to perform the following operations with their current results until the next information bit is decoded, and repeat the above steps respectively, thus producing 8 different decoding results. After all the decoder values are updated the current paths needs to be selected. The newly updated PM value is compared with the original PM value, and the eight paths with the minimum PM value are selected as valid values, and the eight SC decoders are updated based on this. After the decoding of this layer, the path with the minimum PM is selected as the current decoding result, and the result is output to the upper layer as PSUM through the intermediate register. The reverse transmission data has completed the higher-order G function and partial sum function. Finally, the decoding result with the minimum PM can be obtained as the output of the decoding.

In order to improve the generality of the module, the above decoding module is properly encapsulated, and the interface is configured using AXI protocol. For the configuration interface, the AXI-Lite protocol is used for the configuration before data transmission. For the data transmission interface, AXI-Stream is adopted to satisfy high-speed streaming data transmission. The data is input by a block-by-block basis, and the configuration can be performed with the same basis. The corresponding architecture is shown as FIG.4.

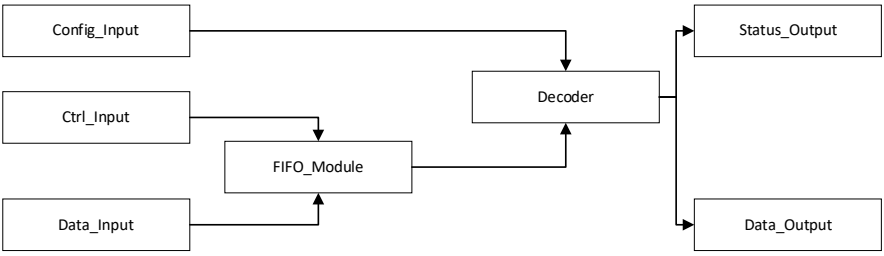

Fig. 4. Encapsulated decoder structure

4 Results and Comparisons

A verification platform for the design is shown by FIG.5, the encoder reads the data from the ROM as input, cache it in the FIFO and input it to the LLR conversion module to complete the conversion from hard-bits to LLR values, and then input the LLR values to the decoder, which outputs the decoding results to the Checker and compares them with the original ROM data to detect the bit error rate.

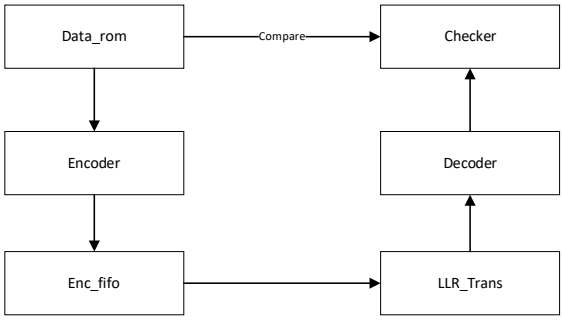

Fig. 5. Verification platform structure

This design is implemented on XC7K325T-2FFG900C FPGA, FIG.6 shows the hardware result with Signal-Noise Ratio (SNR) is 0dB, FIG.7 shows the performance statistics in the case of other code lengths. It can be seen that polar codes have achieved a relatively ideal performance in the current situation.

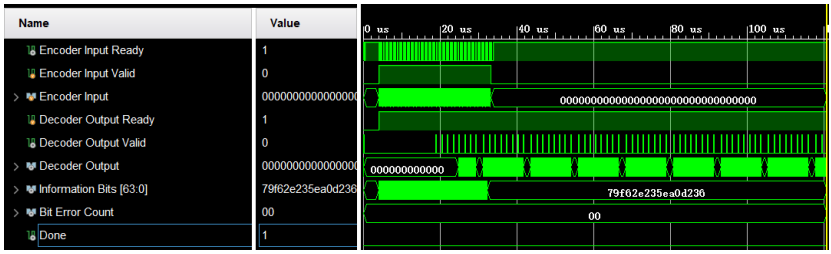

Fig. 6. Hardware result for SNR = 0dB

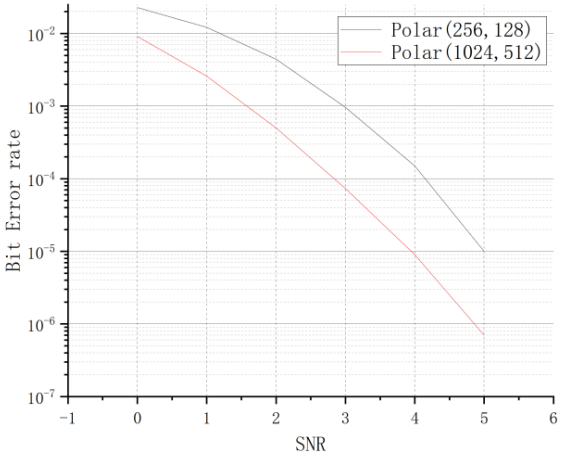

Fig. 7. BER curves of different code length

Table 1 shows the throughput and resource utilization of this design. Compared with [6] and [7], this design not only supports multi-code length configuration, but also consumes resources of 36.5% and 29.5% respectively. Compared with [8], when N=1024, the proposed architecture can achieve almost the same throughput with 49.9% LUT consumption and 43.6% BRAM consumption.

Table 1. Comparison with other design (N=1024)

	[6]	[7]	[8]	This Work
FPGA	Xilinx Kintex 7			
N	1024	1024	32-1024	32-1024
L	4	16	8	8
LUTs	142961	177306	104700	52220
BRAMs	-	130	39	17
Throughput (Mbps)	115	-	160	158

5 Conclusions

In this paper, a resource efficient hardware implementation method is presented. This design support code word length from 32 to 1024, have a maximum throughput of 158 Mbps for N = 1024. Compared with other hardware implementation method, this method can save up to 70% resource cost, which makes it a better construction method for hardware.

References

1. Arikan E. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels [J]. IEEE Transactions on Information Theory, 2009, 55(7): 3051-3073
2. Trifonov P. Effificient design and decoding of polar codes [J]. IEEE Transactions on Communications, 2012, 60(11): 3221-3227
3. H. Afşer and H. Deliç, "On the Channel-Specific Construction of Polar Codes," in IEEE Communications Letters, vol. 19, no. 9, pp. 1480-1483, Sept. 2015, doi: 10.1109/LCOMM.2015.2450213.
4. A. Arpure and S. Gugulothu, "FPGA implementation of polar code based encoder architecture," 2016 International Conference on Communication and Signal Processing (ICCSP), 2016, pp. 0691-0695
5. C. Xia et al., "An implementation of list successive cancellation decoder with large list size for polar codes," 2017 27th International Conference on Field Programmable Logic and Applications (FPL), 2017, pp. 1-4.
6. C. Xiong, Y. Zhong, C. Zhang, and Z. Yan, An FPGA Emulation Platform for Polar Codes, in 2016 IEEE International Workshop on Signal Processing Systems (SiPS), 2016, pp. 148153.

7. http://www.polaran.com/products.html.
8. L. V. Q. Nguyen, J. Dion and N. Gresset, "Polar Decoding Hardware Implementation," 2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2019, pp. 622-625.