Syndrome-aided Pruning Architecture for Successive-Cancellation Decoding of Polar Codes

Dohyun Ryu, Hyeonkyu Kim, and Hoyoung Yoo
Department of Electronics Engineering,
Chungnam National University,
Daejeon, 34134, Republic of Korea
dhryu.cas@gmail.com; hkkim.cas@gamil.com; hyyoo@cnu.ac.kr;

Abstract

This paper presents a new successive-cancellation decoding architecture for long polar codes. In the proposed decoder, unnecessary recursive computation is eliminated when syndrome check is satisfied for corresponding constituent codes. Due to a simple syndrome check, a large number of node computations can be saved by replacing recursive computations with a simple calculation. The proposed decoding architecture to check syndromes is presented to prohibit an increase of latency and critical path delay.

Keywords: polar codes, error-correction codes, pruning method, optimization.

1. Introduction

Polar codes have recently gather significant attentions due to its channel achieving property, and many applications including 5G communication systems and massive solid-state drive systems are considering its applicability [1]. Although the polar codes can provide a good error correction capability, traditional successive-cancellation decoding [1] and its list variant [2] algorithms suffer from a long latency problem. Since the traditional algorithms adopt divide-and-conquer approach, a long latency problem is inevitable, and it might lead a huge amount of power consumptions. Recently, many literatures are proposed to mitigate this problem [3]-[5], and up to our knowledge [6] shows the best pruning performance. In this manuscript, we presents a syndrome-based decoding architecture based on [6] and analyze its practical implementation.

2. Review of Syndrome-Check Pruning

The polar decoding can be considered as tree structure. Let us suppose that a node n receives soft information α_n from the parent node n_p. First, the node n calculates the left soft information α_{nl} using min-sum approximation as

$$\alpha_{nl}[i]=\mathrm{s}(\alpha_n[2i-1])\times\mathrm{s}(\alpha_n[2i])\times\min(|\alpha_n[2i-1]|,|\alpha_n[2i]|) \quad (1)$$

Next, the decoder moves the left child node n_l. After receiving the left hard information β_{nl} from the left child node, the decoder moves to the right child node n_r accompanying with the right soft information

$$\alpha_{nr}[i]=(1-2\beta_{nl}[i])\alpha_n[2i-1]+\alpha_n[2i] \quad (2)$$

When receiving the right hard information β_{nr} from the right child node, the decoder calculates the node n hard information β_n as (3), and it sends the hard information to the parent node n_p.

$$\beta_n[2i-1]=\beta_{nl}[i]\oplus\beta_{nr}[i],\ \beta_n[2i]=\beta_{nr}[i] \quad (3)$$

where $\oplus$ denotes the XOR operation.

Syndrome-check pruning method [6] eliminates unnecessary soft information computations of (1) and (2) by checking syndromes for frozen bit positions as

$$SYN_n[\,frozen\ bit]=h(\alpha_n)G_n \quad (4)$$

where G_n denotes a generator matrix of node n.

When a decoder satisfies the syndrome check, the proposed method eliminates the recursive computations, and it is replaced with a simple calculation as (5)

$$\begin{aligned}&\beta_{nl}=h(\alpha_{nl}),\ \beta_{nr}=h(\alpha_{nr}),\\&\hat{u}_{nl}=h(\alpha_{nl})G_n,\ \hat{u}_{nr}=h(\alpha_{nr})G_n\end{aligned} \quad (5)$$

where $h(x)$ is a hard decision function and $\hat{u}_n$ is estimated bits for node n.

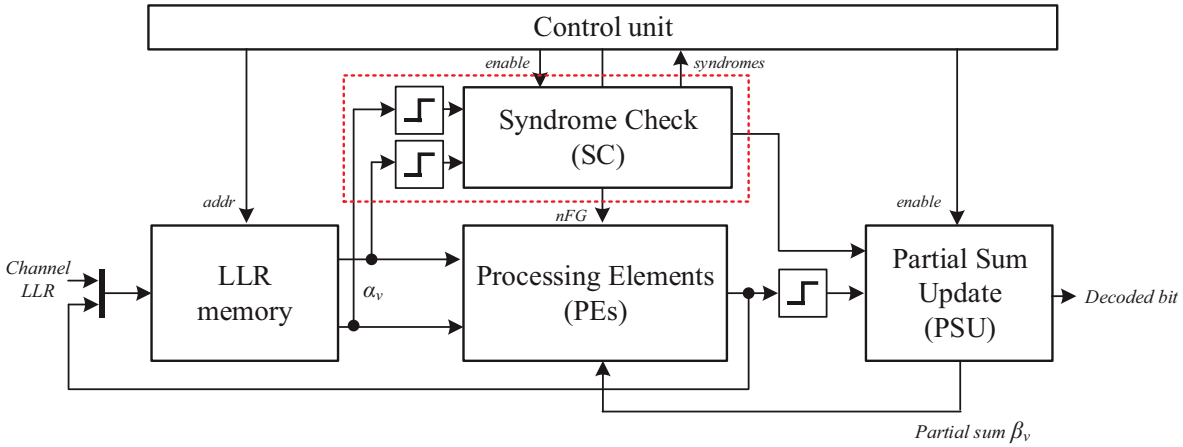

Fig. 1. The proposed Syndrome-check Pruning Architecture.

3. Hardware Architecture

Figure 1 presents syndrome-check based decoding architecture based on [6] by employing the semi-parallel structure [7]. It consists of LLR memory, syndrome check unit, processing elements, and partial sum update unit. The LLR memory contains the channel LLRs and soft information. The syndrome check unit identifies prunable codes based on (4), and the processing elements selectively calculate soft information of (2) and (3). The partial sum unit computes partial sums of hard information, which is needed in the processing units.

It is important to decide when to perform the syndrome check. If the syndrome check is activated before the traditional SC decoding is initiated, it may increase the latency drastically. As the syndrome check and the traditional SC decoding can be carried out simultaneously in Fig. 1, we decide that the syndrome is checked while the soft information of the traditional SC decoding is being calculated so as to achieve the benefit of the syndrome check algorithm [6]. Under this architecture, no additional latency is demanded when the syndrome check fails, since the traditional SC decoding can continue seamlessly. When the syndrome check is satisfied, on the other hand, the latency analysis should take into account the time to check the syndromes and obtain the outputs. Fortunately, the results of the estimated bit can be directly used to obtain the syndrome check, since the parity check matrix is a part of the generator matrix according to (4) and (5), and thus the time to compute a single matrix multiplication is only taken at the expense of pruning a node.

Furthermore, the syndrome check block can be realized with combinational logics. More precisely, the matrix multiplication can be realized by a combination of XOR gates. Therefore, the proposed algorithm can significantly improve the decoding latency at the expense of a small hardware increase.

4. Conclusion

A new decoding architecture is proposed to adopt syndrome-check pruning method without introducing an increase of latency and critical path delay. The proposed architecture can provide a practical realization for long polar codes by mitigating long latency.

Acknowledgement

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2013-0-00405, Development of Device Collaborative Giga-Level Smart Cloudlet Technology)

References

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

[2] I. Tal and A. Vardy, “List decoding of polar codes,” in *Proc. IEEE Int. Symp. Inform. Theory*, 2011, pp. 1–5.

[3] A. Alamdar-Yazdi and F. R. Kschischang, “A simplified successive- cancellation decoder for polar codes,” *IEEE Commun. Lett.,* vol. 15, no. 12, pp. 1378–1380, Dec. 2011.

[4] G. Sarkis and W. J. Gross, “Increasing the throughput of polar decoders,” *IEEE Commun. Lett.,* vol. 17, no. 4, pp. 725–728, April 2013.

[5] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W.J. Gross. “Fast polar decoders: algorithm and implementation,” *IEEE J. Sel. Areas Commun.,* vol. 32, no. 5, pp. 946−957, May 2014.

[6] H. Yoo and I.-C. Park, “Efficient Pruning for Successive-Cancellation Decoding of Polar Codes,” *IEEE Communications Letters,* vol. 20, no. 12, pp. 2362-2365, Dec. 2016.

[7] C. Leroux, A. J. Raymond, G. Sarkis, and W. J. Gross, “A semi-parallel successive-cancellation decoder for polar codes,” *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 289–299, Jan. 2013.