INTERNATIONAL JOURNAL OF EMINENT ENGINEERING TECHNOLOGIES

Volume 5, Issue 1 AUG 2017

DESIGN AND IMPLEMENTATION OF A PARTIALLY PARALLEL ENCODER FOR LONG POLAR CODES

MSK. Reshma [1] **D.Raghava Reddy**[2]

[1]PG Scholar, Narasaraopeta institute of technology, kotappakonda road (V), narasaraopeta (M), guntur,A.P, 522601[1] ,reshma.msk1992@gmail.com

[2] Assistant Professor,Dept of ECE, Narasaraopeta institute of technology, kotappakonda road (V), narasaraopeta (M), guntur,A.P, 522601[2] ,raghavareddy89@gmail.com

Abstract: Due to the channel achieving property, the polar code has become one of the most favorable error-correcting codes. As the polar code achieves the property asymptotically, however, it should be long enough to have a good error-correcting performance. Although the previous fully parallel encoder is intuitive and easy to implement, it is not suitable for long polar codes because of the huge hardware complexity required. In this brief, we analyze the encoding process in the viewpoint of very-large-scale integration implementation and propose a new efficient encoder architecture that is adequate for long polar codes and effective in alleviating the hardware complexity. As the proposed encoder allows high-throughput encoding with small hardware complexity, it can be systematically applied to the design of any polar code and to any level of parallelism.

Index Terms--Polar codes, polar encoder, very-large-scale integration (VLSI) optimization.

I.Introduction

The polar code is a new class of error correcting codes that provably achieves the capacity of the underlying channels. Though the polar code achieves the underlying channel capacity, the property is asymptotical, since a good error correcting performance is obtained when the code is sufficiently long. The polar code has been regarded as being associated with low complexity, such a long polar code suffers from severe hardware complexity and long latency. Therefore, an architecture that can efficiently deal with long polar codes is necessary to make the VLSI implementation feasible.Among a few manuscripts dealing with the hardware implementation presented a straightforward encoding architecture that processes all the message bits in a fully parallel manner. The fully parallel architecture is intuitive and easy to implement, but it is not suitable for long polar codes due to the excessive hardware complexity. Hence we present the encoding process in the viewpoint of VLSI implementation and proposes a partially parallel architecture. The proposed encoder is highly attractive in implementing a long polar encoder, as it can achieve a high throughput with a small hardware complexity.

II. Polar Encoding

The polar code utilizes the channel polarization phenomenon that each channel approaches either a perfectly reliable or a completely noisy channel as the code length goes to infinity over a combined channel constructed with a set of N identical sub-channels. As the reliability of each sub-channel is known as priori, K most reliable sub-channels are set to predetermined values to construct a polar (N,K) code.

Since the polar code belongs to the class of linear block codes, the encoding process can be characterized by the generator matrix. The generator matrix GN for code length N (2n) is obtained by applying the nth Kronecker power to

the kernel matrix. Given the generator matrix, the code word is computed by x=uGN, where u and x represent information and code word vectors, respectively. Throughout the paper, we assume that information vector u is arranged in a natural order, whereas code word vector x is arranged in a bit reversed order. The encoding complexity of O(NlogN)=19.26 for a polar code of length N=16 and takes n=4 stages when N=2n .

A polar code with a length of 16 bits is implemented with 32 XOR gates and the processed with 4 stages as shown in fig.1. The fully parallel encoder is intuitively designed based on the generator matrix, but implementing such an encoder becomes a significant burden when a long polar code is used to achieve a good error correcting performance. The memory size and the number of XOR gates increase as the code length increases.

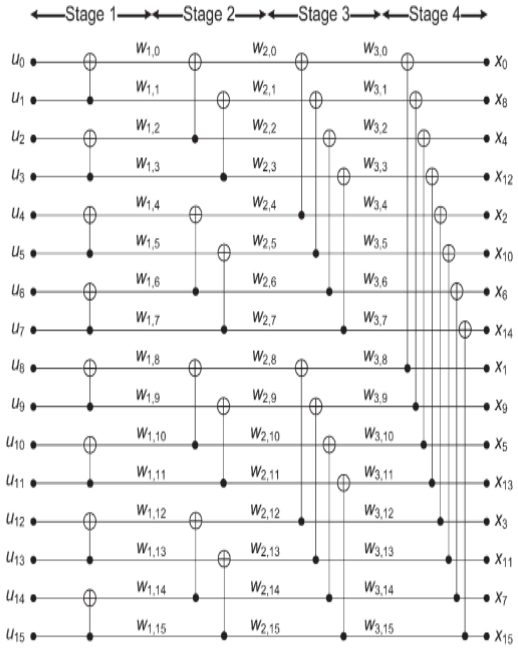


Fig. 1. Fully parallel architecture for encoding a 16-bit polar code.

III. Proposed Polar Encoder

In this section, we propose a partially parallel structure to encode long polar codes efficiently. To clearly show the proposed approach and how to transform the architecture, a 4-parallel encoding architecture for the 16-bit polar code is exemplified in depth. The fully parallel encoding architecture is first transformed to a folded form. Given the 16-bit DFG, the 4-parallel folded architecture that processes 4 bits at a time can be realized with placing two functional units in each stage since the functional unit computes 2 bits at a time.

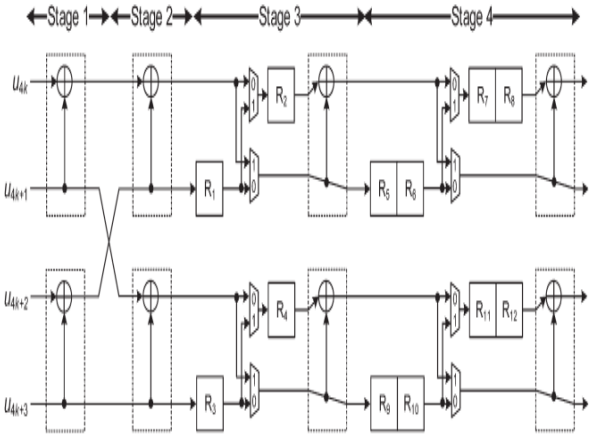


Fig 2. Proposed 4-parallel folded architecture for encoding the polar (16, *K*) codes.

In the folding transformation, determining a folding set, which represents the order of operations to be executed in a functional unit, is the most important design factor.To construct efficient folding sets, all operations in the fully parallel encoding are first classified as separate folding sets. Since the input is in a natural order, it is reasonable to alternatively distribute the operations in the consecutive order. Thus, each stage consists of two folding sets, each of which contains only odd or even operations to be performed by a separate unit.

Considering the four-parallel input sequence in a natural order, stage 1 has two

folding sets of *{A*0, *A*2, *A*4, *A*6*}* and *{A*1, *A*3, *A*5, *A*7*}*. Each folding set contains four elements, and the position of an element represents the operational order in the corresponding functional unit. Two functional units for stage 1 execute *A*0 and *A*1 simultaneously at the beginning and *A*2 and 3 at the next cycle, and so forth. The folding sets of stage 2 have the same order as those of stage 1, i.e., *{ B*0, *B*2, *B*4, *B*6*}* and *{B*1, *B*3, *B*5, *B*7*}*, since the four-parallel input sequence of stage 2 is equal to that of stage 1.

Furthermore, to determine the folding sets of another stage *s*, the property that the functional unit processes a pair of inputs whose indices differ by 2*s*− 1 is exploited. In the case of stage 3, two data whose indices differ by 4 are processed together, which implies that the operational distance of the corresponding data is two as the kernel functional unit computes two data at a time. For instance, *w* 2,0 and *w*2,4 that come from *B*0 and *B*2 are used as the inputs to *C*0. Since both inputs should be valid to be processed in a functional unit, the operations in stage 3 are aligned to the late input data. Cyclic shifting the folding sets right by one, which can be realized by inserting a delay of one time unit, is to enable full utilization of the functional units by overlapping adjacent iterations. As a result, the folding sets of stage 3 are determined to *{C*6, *C*0, *C*2, *C*4*}* and *{C*7, *C*1, *C*3, *C*5*}*, where *C*6 in the current iteration is overlapped with *A*0 and *B*0 in the next iteration. In the same manner, the property that the functional unit processes a pair of inputs whose indices differ by 8 is exploited in stage 4. The folding sets of stage 4 are *{D*2, *D*4, *D*6, *D*0*}* and *{D*3, *D*5, *D*7, *D*1*}*, which are obtained by cyclic shifting the previous folding sets of stage 3 by two.

$$\text{Stage1: } \{A0, A2, A4, A6\}, \{A1, A3, A5, A7\}$$

$$\text{Stage2: } \{B0, B2, B4, B6\}, \{B1, B3, B5, B7\}$$

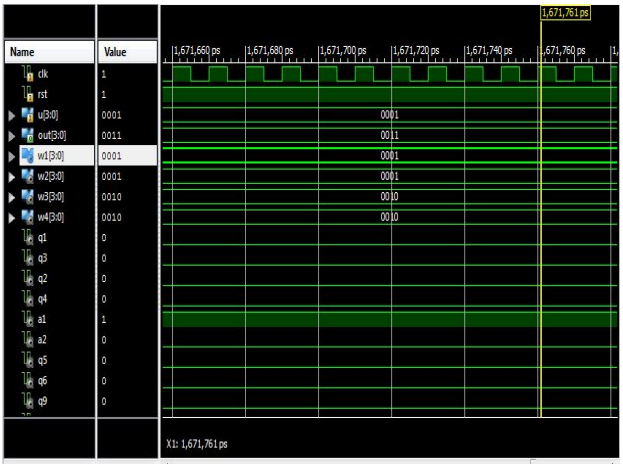
$$\text{Stage3: } \{C6, C0, C2, C4\}, \{C7, C1, C3, C5\}$$

$$\text{Stage4: } \{D2, D4, D6, D0\}, \{D3, D5, D7, D1\}$$

Generally speaking, a stage whose index *s* is less than or equal to log2 *P*, where *P* is the level of parallelism, has the same folding sets determined by evenly interleaving the operations in the consecutive order, and another stage whose index *s* is larger than log2 *P* has the folding sets obtained by cyclic shifting the previous folding sets of stage *s* − 1 right by *s* − log2 *P*.

IV.Results:

Simulation results:



Design summary:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	7	14752	0%
Number of Slice Flip Flops	12	29504	0%
Number of 4 input LUTs	8	29504	0%
Number of bonded IOBs	10	250	4%
Number of GCLKs	1	24	4%

Synthesis report:

```
Data Path: u<3> to out<1>
                                   Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   IBUF:I->O             5   1.106   0.541  u_3_IBUF (u_3_IBUF)
   LUT4:I3->O            3   0.612   0.520  Madd_w2<0>_Madd_xor<0>11 (
   LUT2:I1->O            1   0.612   0.357  Madd_a1_lut<0>1 (Madd_a1_l
   OBUF:I->O                 3.169          out_1_OBUF (out<1>)
  ----------------------------------------
  Total                      6.917ns (5.499ns logic, 1.418ns route)
                                     (79.5% logic, 20.5% route)
```

V. CONCLUSION

This brief has presented a new partially parallel encoder architecture developed for long polar codes. Many optimization techniques have been applied to derive the proposed architecture. Experimental results show that the proposed architecture can save the hardware by up to 73% compared with that of the fully parallel architecture. Finally, the relationship between the hardware complexity and the throughputs is analyzed to select the most suitable architecture for a given application. Therefore, the proposed architecture provides a practical solution for encoding a long polar code.

REFERENCES

[1] E. Arikan, "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, Jul. 2009.

[2] R. Mori and T. Tanaka, "Performance of polar codes with the construction using density evolution,"IEEE Commun. Lett., vol. 13, no. 7, pp. 519– 521, Jul. 2009.

[3] S. B. Korada, E. Sasoglu, and R. Urbanke, "Polar codes: Characterization of exponent, bounds, constructions,"IEEE Trans. Inf. Theory, vol. 56, no. 12, pp. 6253–6264, Dec. 2010.

[4] I. Tal and A. Vardy, "List decoding of polar codes," inProc. IEEE ISIT, 2011, pp. 1–5.

[5] K. Chen, K. Niu, and J. Lin, "Improved successive cancellation decoding of polar codes,"IEEE Trans. Commun., vol. 61, no. 8, pp. 3100–3107, Aug. 2013.

[6] G. Sarkis and W. J. Gross, "Polar codes for data storage applications," in Proc. ICNC, 2013, pp. 840–844.

[7] 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.

[8] G. Berhault, C. Leroux, C. Jego, and D. Dallet, "Partial sums generation architecture for successive cancellation decoding of polar codes," inProc. IEEE Workshop SiPS, Oct. 2013, pp. 407–412.

[9] B. Yuan and K. K. Parhi, "Low-latency successive-cancellation polar decoder architectures using 2-bit decoding,"IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, no. 4, pp. 1241–1254, Apr. 2014.

[10] 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.