ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 5, Issue 1, January 2016

An Advanced Architecture for 16-bit Polar Codes using Partially Parallel Encoder

Yamuna devi S[1](Asst. Professor), Magdalinjoenita G[2](Asst.Professor), Revathi V[3](Asst.Professor)

[1]*Electronics & Communication Engg.,Aksheyaa College of Engg., Madhuranthagam, 603314, India*
[2]*Electronics & Communication Engg.,Aksheyaa College of Engg., Madhuranthagam, 603314, India*
[3]*Electronics & Communication Engg.,Aksheyaa College of Engg., Madhuranthagam, 603314, India*

***Abstract*— The most favourable error correcting code in use, today is the polar code, due to its channel achieving property. This property is achieved by the polar code in a asymptotic manner, even though, in order to have a good performance over error correcting, it must be longer.We use a partially parallel encoder in this paper since the already available fully parallel encoder does not suit for long coded due to its complexity in the hardware. Here, the process of encoding has been analysed for VLSI implementation and an advanced architecture required for long polar codes has been proposed with reduced hardware complexity. This proposed architecture has high throughput so it can be designed for higher levels of parallelism too.**

***Index Terms*— Partially parallel encoder, long polar codes, VLSI implementation.**

I. INTRODUCTION

The polar code is a new class of error correcting codes that provably achieves the capacity of the underlying channels [1]. 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 [2]. 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 [1] 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 [1]. 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 G_N for code length N (2^n) is obtained by applying the nth Kronecker power to the kernel matrix [1]. Given the generator matrix, the code word is computed by $x=uG_N$, 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 $N=2^n$. 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.

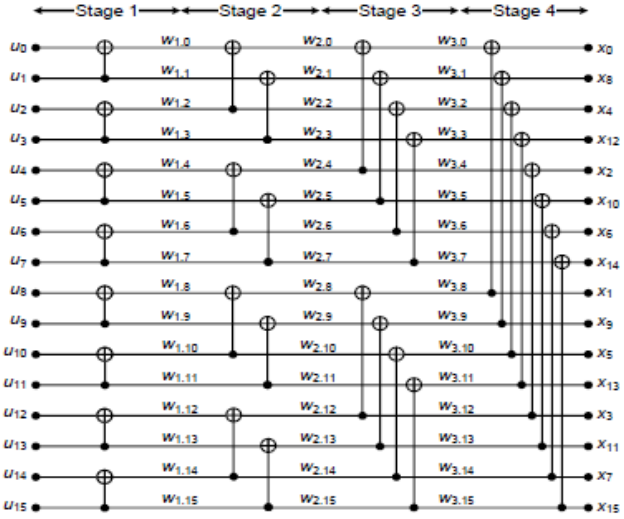

Fig.1 Fully parallel architecture for encoding a 16-bit polar code

III. PROPOSED POLAR ENCODER

A data flow graph (DFG) corresponding to the fully parallel encoding process for 16-bit polar codes is shown in Fig. 2, where a node represents the kernel matrix operation $F=\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}$, and w_{ij} denotes the j^{th} edge at the i^{th} stage. Given the 16-bit DFG, the 4-paralell 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. 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 [3]. 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.

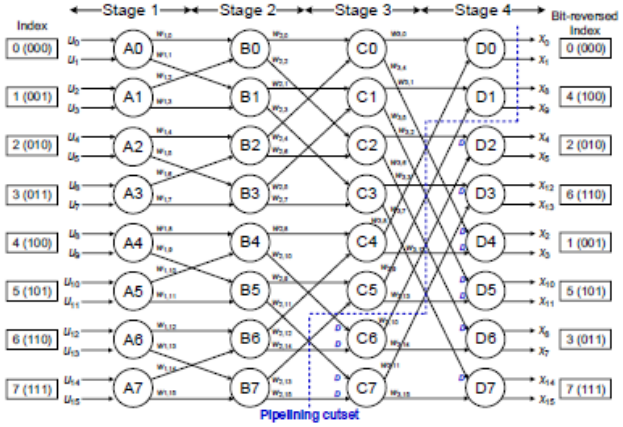

Fig.2 Data flow graph of 16-bit polar encoding

Considering the four parallel input sequence in a natural order, stage 1 has two folding sets of {A0, A2, A4, A6} and {A1, A3, A5, A7}. 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 A0 and A1 simultaneously at the beginning, and A2 and A3 at the next cycle, and so forth. The folding sets of stage 2 have the same order as those of stage 1, that is {B0, B2, B4, B6} and {B1, B3, B5, B7}, 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}. In 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 B0 and B2 are used as the inputs to C0. 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 {C6,C0,C2,C4} and {C7,C1,C3,C5}, where C6 in the current iteration is overlapped with A0 and B0 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 {D2, D4, D6, D0} and {D3, D5, D7, D1}, which are obtained by cyclic shifting the previous folding sets of stage 3 by two. When an edge w_{ij} from functional unit S to functional unit T has a delay of d, the delay requirements for w_{ij} in the F-folded architecture can be calculated as

$$D(w_{ij})=Fd+t-s$$

where t and s denote the position in the folding set corresponding to T and S, respectively. Here the kernel functional unit is not pipelined. The DFG is pipelined by inserting delay elements as shown in Fig. 2, where the dashed line indicates the pipeline cutset associated with 12 delay elements. The delay requirements of the pipelined DFG, $D'(w_{ij})$, are recalculated based on above equation. As a result, 8 functional units and 48 delay elements in total are enough to implement the 4-parallel 4-folded encoding architecture based on the folding sets.

Once the minimum number of delay elements has beendetermined, each variable is allocated to a register. Finally, the resulting 4-parallel pipelined structure proposed to encode the 16-bit polar code is illustrated in Fig.3, which consists of 8 functional units and 12 delay elements. A pair of two functional units takes in charge of one stage, and the delay elements are to store variables according to the register allocation table. The hardware structures for stages 1 and 2 can be straightforwardly realized as no delay elements are necessary in those stages, whereas for stages 3 and 4 several multiplexers are placed in front of some functional units to configure the inputs of the functional units. The proposed architecture continuously processes four samples per cycle according to the folding sets and the register allocation table. Note that the proposed encoder takes a pair of inputs in a natural order and generates a pair of outputs in a bit-reversed order as shown in Fig 2. As the functional unit in the proposed architecture processes a pair of 2 bits at a time, the proposed architecture maintains the consecutive order at the input side and the bit-reversed order at the output side if a pair of consecutive bits is regarded as a single entity.

IV. VLSI IMPLEMENTATIONS

Discussed methods were physically realized on a FPGAbased rapid prototyping system for various register sizes andtested using on-chip hardware-in-the-loop co-simulation. Thearchitectures were designed for digital realization within theXilinx with synthesis options set to generic Verilog HDL generation.This was necessary because the auto-generated register transferlanguage (RTL) hardware descriptions are targeted on FPGAs.

Selected Device: 3s250epq208-5

Number of Slices	8 out of 2448 0%
Number of Slice Flip Flops	12 out of 4896 0%
Number of 4 input LUTs	13 out of 4896 0%
Number of IOs	12
Number of bonded IOBs	12 out of 158 7%
Number of GCLKs	1 out of 24 4%

Timing Summary:

Minimum period: 2.826ns (Maximum Frequency: 353.870MHz)

V. CONCLUSION

In this paper, we proposed a new partially parallel encoder for 16-bit polar codes. The proposed architecture can save the hardware compared to that of the fully

parallel architecture. Therefore, the proposed architecture provides a practical solution for encoding a long polar code.

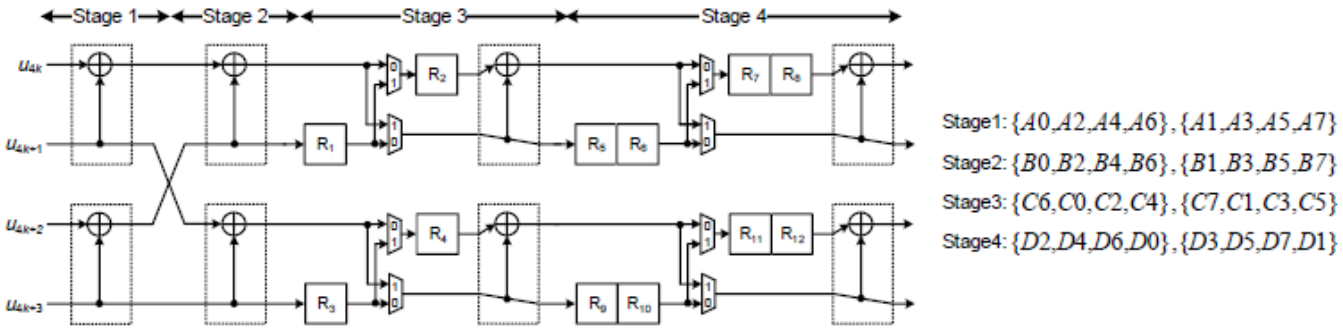

Fig.3 4-parallel folded architecture for encoding the polar (16, K) codes

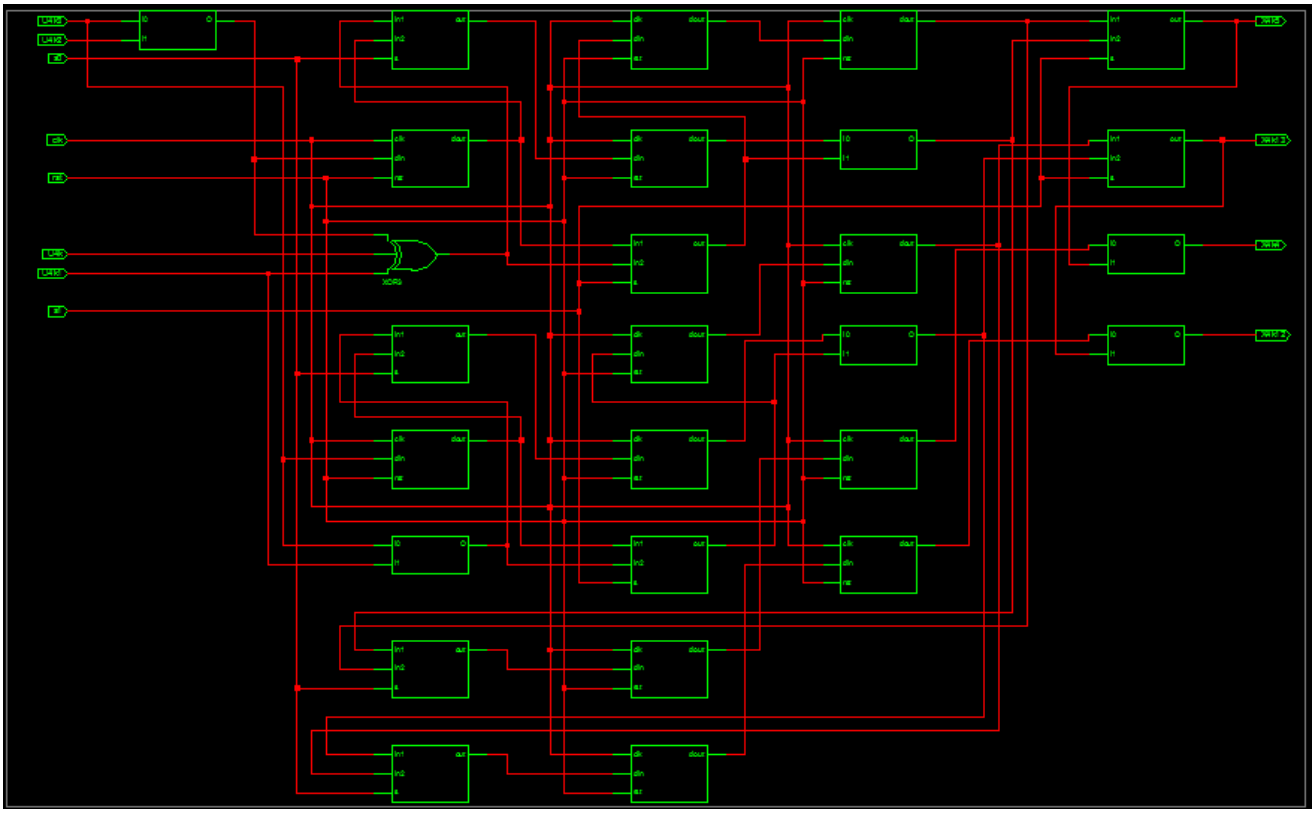

Fig.4 RTL Schematic

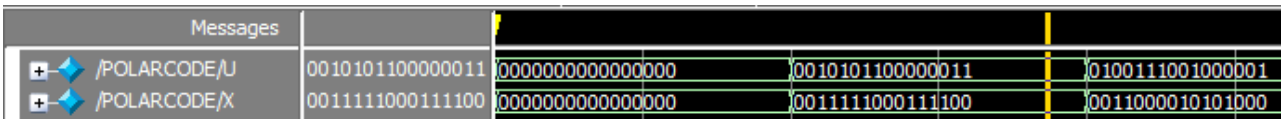

Fig.5 Simulation result

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] HoyoungYoo, and In-Cheol Park, 'Partial parallel encoder architecture for long polar codes', IEEE Transactions on Circuits and Systems II, Oct. 21, 2014

[3] K. K. Parhi, 'VLSI Digital Signal Processing Systems: Design and Implementation', Hoboken, NJ: Wiley, 1999.

[4] www.xlinix.com

[5] www.altera.com

[6] 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.

[7] C. Wang and K. K. Parhi, "High-level DSP synthesis using concurrent transformations, scheduling, and allocation," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 14, no. 3, pp. 274–295, Mar 1995.