International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 18
December2016

# FPGA Implementation of 32-Bit Partially Parallel Encoder Architecture for Long Polar Codes

**[1] Abdul Rameem Sultana & [2] Mr. V. Subba Raju**

1.Pg Scholar,Department Of Ece, Dhanekula Institute Of Engineering Technology, Ganguru, Vijayawada, Krishna Dt, Ap, India.

2. Associate Professor, Department Of Ece, Dhanekula Institute Of Engineering Technology, Ganguru, Vijayawada, Krishna Dt, Ap, India.

**ABSTRACT:**

*Polar code is really a new type of error-fixing codes that provably accomplishes the capability from the underlying channels. Because of the funnel achieving property, the polar code is becoming probably the most favorable error-fixing codes. Because of the funnel capacity achieving property, the polar code has become regarded as a significant breakthrough in coding theory, and also the usefulness from the polar code has been investigated in lots of programs, including data storage products. Because the suggested encoder enables high-throughput encoding with small hardware complexity, it may be methodically put on the style of any polar code and also to any degree of parallelism. Because the polar code accomplishes the home asymptotically, however, it ought to be lengthy enough to possess a good error-fixing performance. Even though the previous fully parallel encoder is intuitive and simple to apply, it's not appropriate for lengthy polar codes due to the large hardware complexity needed. Within this brief, we evaluate the encoding process for 32bits in both techniques within the point of view of very-large-scale integration implementation and propose a brand new efficient encoder architecture that's sufficient for lengthy polar codes and efficient in alleviating the hardware complexity.*

Keywords: Polar codes, polar encoder, very-large-scale integration (VLSI) optimization**.**

## I. INTRODUCTION

Additionally, concrete calculations for creating, encoding, and deciphering the code are developed. Even though the polar code accomplishes the actual funnel capacity, the home is asymptotical since a great error fixing performance is acquired once the code length is sufficiently lengthy. To bond with the funnel capacity, the code length ought to be a minimum of $2^{20}$ bits, and lots of literature works introduced polar codes varying from $2^{10}$ to $2^{15}$ to attain good error-fixing performances used [1]. Because of the funnel capacity achieving property, the polar code has become regarded as a significant breakthrough in coding theory, and also the usefulness from the polar code has been investigated in lots of programs, including data storage products. Polar code is really a new type of error-fixing codes that provably accomplishes the capability from the underlying channels. Additionally, how big a note paid by a mistake-fixing code kept in storage systems is generally 4096 bytes, i.e., 32 768 bits, and it is likely to be lengthened to 8192 bytes or 16 384 bytes soon.

Even though the polar code continues to be considered to be connected with low complexity, this type of lengthy polar code is affected with severe hardware complexity and lengthy latency [2]. Therefore, an architecture that may efficiently cope with lengthy polar codes is essential to help make the very-large-scale integration (VLSI) implementation achievable. Various theoretic facets of the polar code, including code construction and deciphering calculations, happen to be investigated. However, hardware architectures for polar encoding have rarely been talked about. Among a couple of manuscripts coping with hardware implementation, presented an easy encoding architecture that processes all of the message bits inside a fully parallel manner. The fully parallel architecture is intuitive and simple to apply, but it's not appropriate for lengthy polar codes because of excessive hardware complexity. Additionally, the partial sum network (PSN) to have an SC decoder is considered like a polar encoder. Because of the nature of successive deciphering, however, the amount of inputs is seriously restricted within the PSN, one or two bits at any given time. Since a polar encoder typically takes the inputs from the buffer or memory which bit width is a lot bigger, the PSN isn't suitable for creating an over-all polar encoding architecture. The very first time, this brief evaluates the encoding process within the point of view of VLSI implementation and proposes a partly parallel architecture. The suggested encoder is extremely attractive in applying a lengthy polar encoder as it can certainly acquire a high throughput with small hardware complexity.

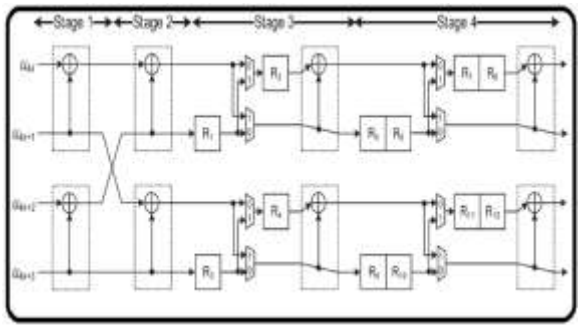



**Fig.1.Proposed 4-parallel folded architecture**

## II. EXISTING ENCODING

The polar code utilizes the funnel polarization phenomenon that every funnel approaches whether perfectly reliable or perhaps a completely noisy funnel because the code length would go to infinity on the combined funnel built with some Corresponding sub channels [3]. The generator matrix GN for code length N An easy fully parallel encoding architecture was presented, that has encoding complexity of O(N logN) for any polar code of length N and takes n stages when N = 2n.Because the longevity of each sub channel is famous a priori, K most dependable sub channels are utilized to transmit information, and also the remaining sub channels are going to predetermined values to create a polar (N, K) code. Because the polar code goes towards the type of straight line block codes, the encoding process could be characterized through the generator matrix.

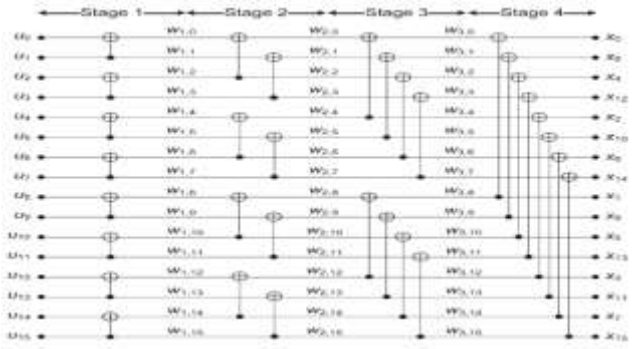



**Fig.2. Fully Parallel Encoder For 16 Bit**

## III. PROPOSED STRUCTURE

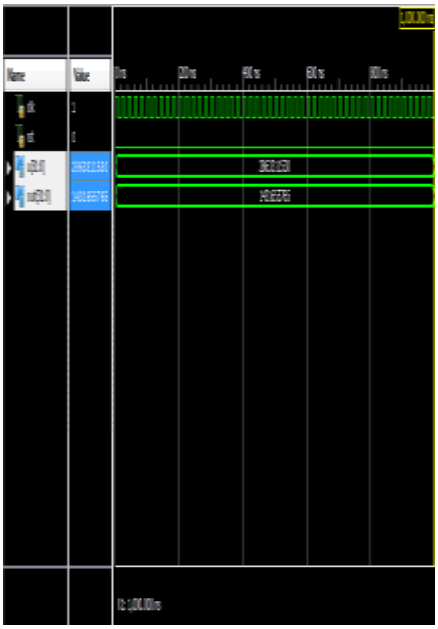
We advise a partially parallel structure to encode lengthy polar codes efficiently. Original delay needs D(wij) and recalculated delay needs D(wij). Straight line lifetime chart for w2j and w3j . Suggested approach and just how to change the architecture, a 4-parallel encoding architecture for that 16-bit polar code is exemplified thorough [4]. The fully parallel encoding architecture is first changed to some folded form, and so the lifetime analysis and register allocation are put on the folded architecture. Lastly, the suggested parallel architecture for lengthy polar codes is described. The folding transformation is broadly accustomed to save hardware sources by time-multiplexing several procedures on the functional unit. An information flow graph (DFG) akin to the fully parallel encoding process for 16-bit polar codes is proven, in which a node signifies the kernel matrix operation F, and wij denotes the jet edge in the it stage. Observe that the DFG from the fully parallel polar encoder is comparable to those of the short Fourier transform with the exception that the polar encoder utilizes the kernel matrix rather than the butterfly operation. Because of the 32-bit DFG, some-parallel folded architecture that processes 4 bits at any given time could be recognized with placing two functional models in every stage because the functional unit computes 2 bits at any given time. Within the folding transformation, figuring out a folding set, which signifies an order of procedures to become performed inside a functional unit, is an essential design factor. To create efficient folding sets, all procedures within the fully parallel encoding are first considered separate folding sets. Because the input is within an all-natural order, it's reasonable to alternatively distribute the procedures within the consecutive order. Thus, each stage includes two folding sets, because both versions consist of only odd or perhaps procedures to become carried out with a separate unit. For that folded architecture to become achievable, the delay needs should be bigger than or comparable to zero for the edges. Pipelining or retiming techniques does apply towards the fully parallel DFG to guarantee that it is folded hardware has nonnegative delays. Four edges with zero delays are specifically marked with negative zeros since additional delays are essential because of the mismatch of the amount of delay elements. The DFG is pipelined by placing delay elements, in which the dashed line signifies the pipeline cut set connected with 12 delay elements. The lifetime analysis is utilized to obtain the minimum quantity of delay elements needed in applying the folded architecture. The duration of every variable is graphically symbolized within the straight line lifetime chart. Consequently, the utmost quantity of live variables is 12, meaning the folded architecture could be implemented with 12 delay elements rather than 48. When the minimum quantity of delay elements continues to be determined, each variable is allotted to some register. For that above example, in this we are implementing the 32 bits to allocate the register allocation. With considering some-parallel processing, variables are carefully allotted to registers inside a forward manner. Finally, the resulting 4-parallel pipelined structure suggested to encode the 32-bit polar code is highlighted, featuring its 8 functional models and 12 delay elements The suggested architecture continuously processes four samples per cycle based on the folding sets and also the register allocation table. Observe that the suggested encoder takes a set of inputs inside a natural order and creates a set of outputs inside a bit-corrected order, as proven. Because the functional unit within the suggested

architecture processes a set of 2 bits at any given time, the suggested architecture keeps the consecutive order in the input side and also the bit reversed order in the output side if a set of consecutive bits is considered like a single entity. Since a practical unit representing the kernel matrix F processes two bits at any given time, each stage necessitates functional models and also the whole structure requires P/2 log2 N functional models as a whole [5]. A set of two functional models takes responsible for one stage, and also the delay elements will be to store variables based on the register allocation table. The hardware structures for stages 1 and a pair of could be straight recognized as no delay elements are essential in individuals stages, whereas for stages 3 and 4, several multiplexers are put before some functional models to configure the inputs from the functional models.

**SIMULATION RESULTS:**

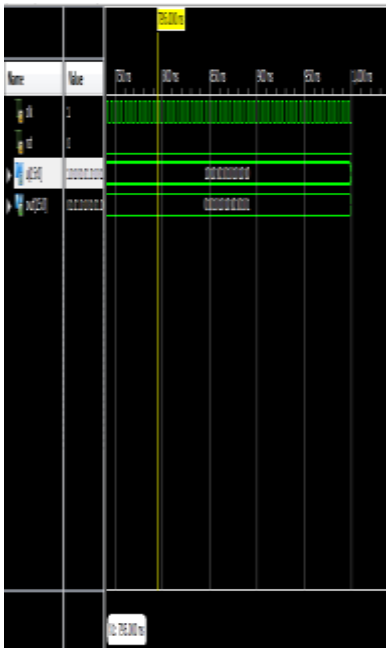
Fig:fully parallel output of 16-bits

Fully parallel 16 bit of four folded parallel

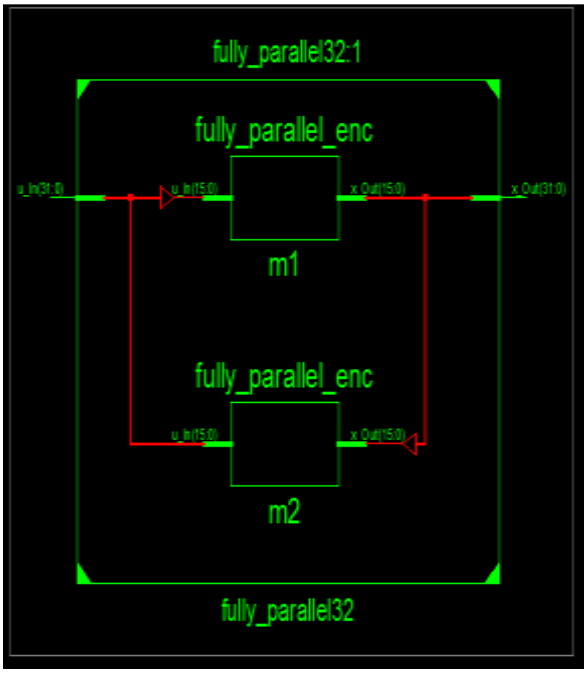


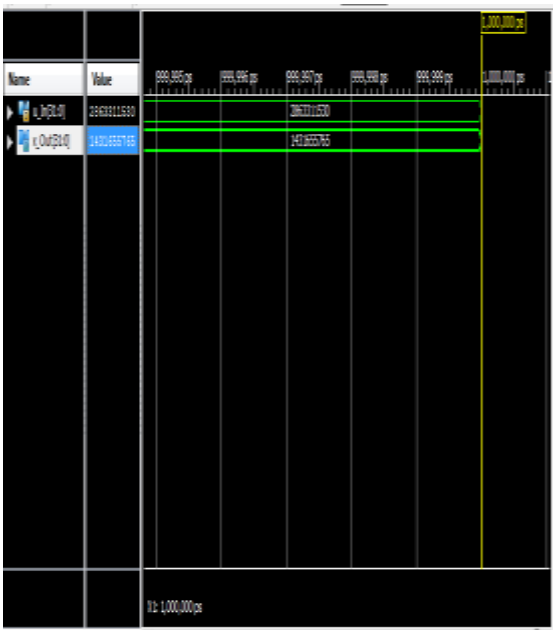
Simulation output of 16-bit proposed

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 18
December2016

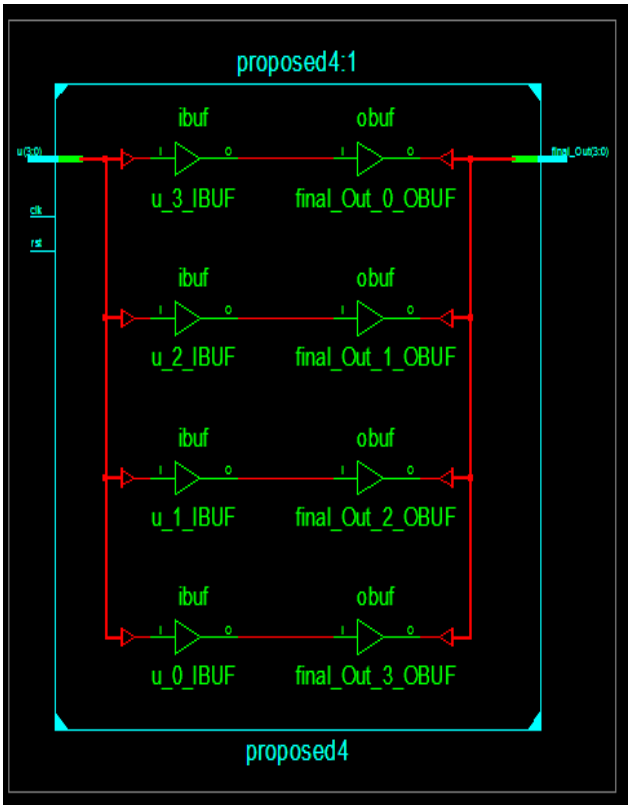


Simulation of 32 bit of four folded

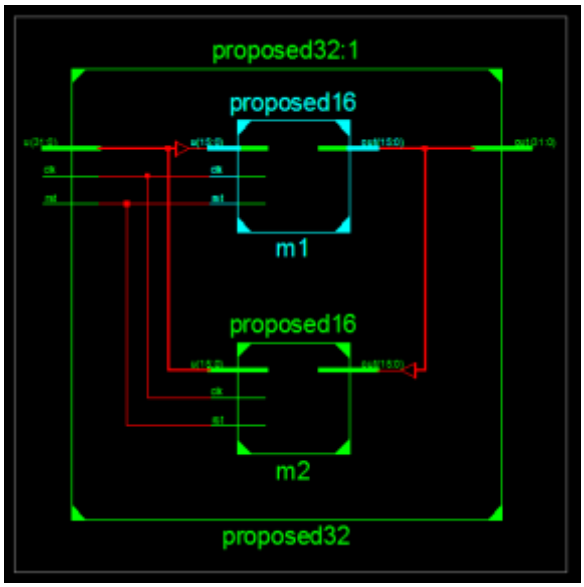


technology view



Fully parallel 16bit encoder
Proposed 32
32 bit fully parallel
32 bit proposed




## IV. CONCLUSION

Many optimization techniques happen to be put on derive the suggested architecture. Experimental results reveal that the suggested architecture can help to save the hardware by as much as 73% in comparison with this from the fully parallel architecture. Within the suggested architecture, the amount of functional models needed within the implementation is dependent around the code length N and the amount of parallelism. This brief has presented a brand new of 32-bits for both fully parallel and four folded partially parallel encoder architecture produced for lengthy polar codes. Finally, the connection between your hardware complexity and also the throughputs is examined to decide on the most appropriate architecture for any given application. Therefore, the suggested architecture supplies a practical solution for encoding a lengthy 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," in *Proc. 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," in *Proc. 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.

[11] A. J. Raymond and W. J. Gross, "Scalable successive-cancellation hardware decoder for polar codes," in *Proc. IEEE GlobalSIP*, Dec. 2013, pp. 1282–1285.

[12] U. U. Fayyaz and J. R. Barry, "Low-complexity soft-output decoding of polar codes," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 958–966, May 2014.

[13] B. Yuan and K. K. Parhi, "Low-latency successive-cancellation list decoders for polar codes with multibit decision," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, DOI: 10.1109/TVLSI.2014.2359793, to be published.

[14] C. Zhang and K. K. Parhi, "Latency analysis and architecture design of simplified SC polar decoders," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 2, pp. 115–119, Feb. 2014.

[15] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. Hoboken, NJ, USA: Wiley, 1999.

[16] K. K. Parhi, "Calculation of minimum number of registers in arbitrary life time chart," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 41, no. 6, pp. 434–436, Jun. 1995.

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

[18] C. Y. Wang, "MARS: A high-level synthesis tool for digital signal processing architecture design," M.S. thesis, Dept. Elect. Eng., University of Minnesota, Minneapolis, MN, USA, 1992.