ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 3, Issue 8, August 2016

Polar Code Encoder Based On DCT Architecture

[1] K. Jaya Lakshmi [2] P.Ravi Kumar
[1] M.Tech-VLSID [2] Senior Assistant Professor
[1][2] Department of ECE, Shri Vishnu Engineering College for Women (Autonomous), Bhimavaram, India

Abstract: **-- A family of low-complexity methods that polarize all discrete memoryless processes is introduced. In both data transmission and data compression, codes based on such methods achieve optimal rates, i.e., channel capacity and source entropy, respectively. The error probability behavior of such codes is as in the binary case.Polarizing capabilities of recursive methods are shown to extend beyond memoryless processes: Any construction that polarizes memoryless processes will also polarize a large class of processes with memory. For this DCT operation**

Key words: Polar DCT, polarize

I. INTRODUCTION

Polar code is an encoding/decoding scheme that provably achieves the capacity of the class of symmetric binary memoryless channels. Here addressing two important problems regarding polar codes: the construction of polar codes and performance of polar codes. First, we consider the problem of efficiently constructing polar codes over symmetric binary discrete memoryless channels and provide some algorithms for channel quantization that can be analyzed for complexity and accuracy. In particular, we show that the algorithm can find almost all the "good" channels with computing complexity which is essentially linear in block-length. Next, different methods and algorithms to enhance the performance of the successive cancellation polar decoder. Numerical evidence as well as some mathematical analysis to show that the performance of polar codes are improved by using these algorithms.

Huawei has been one of the leading companies pushing Polar Codes for 5G. The flexibility features of Polar codes make them very appealing for many applications (high-data rate, short-block transmissions, etc) of 5G and recent experiments have shown that they outperform LDPC and Turbo Codes in various regimes.

Polar codes, invented by Arıkan [3], achieve the capacity of arbitrary binary-input symmetric DMCs. Moreover, they have low encoding and decoding complexity and an explicit construction. Following Arıkan's seminal paper [3], his results have been extended in a variety of important ways. In [22], polar codes have been generalized to symmetric DMCs with non-binary input alphabet. In [14], the polarization phenomenon has been studied for arbitrary kernel matrices, rather than Arıkan's original 2×2 polarization kernel, and error exponents were derived for each such kernel. It was shown in [24] that, under list-decoding, polar codes can achieve remarkably good performance at short code lengths. In terms of applications, polar coding has been used with great success in the context of multiple-access channels [2, 23], wiretap channels [16], data compression [1, 4], write-once channels [6], and channels with memory [21]. In this paper, however, we will restrict our attention to the original setting introduced by Arıkan in [3]. Namely, we focus on binary-input, discrete, memoryless, symmetric channels, with the standard 2×2 polarization kernel under standard successive cancellation decoding.

A method for efficiently constructing polar codes is presented and analyzed. Although polar codes are explicitly de- fined, straightforward construction is intractable since the resulting polar bit-channels have an output alphabet that grows exponentially with the code length. Thus the core problem that needs to be solved is that of faithfully approximating a bit-channel with an intractably large alphabet by another channel having a manageable alphabet size. We devise two approximation methods which "sandwich" the original bit-channel between a degraded and an upgraded version thereof. Both approximations can be efficiently computed, and turn out to be extremely close in practice. We also provide theoretical analysis of our construction algorithms, proving that for any fixed $\varepsilon > 0$ and all sufficiently large code lengths n, polar codes whose rate is within ε of channel capacity can be constructed in time and space that are both linear in n.

ISSN (Online) 2394-6849
International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 3, Issue 8, August 2016

In information theory, a polar code is a linear block error correcting code developed by ErdalArıkan. It is the first code with an explicit construction to provably achieve the channel capacity for symmetric binary-input, discrete, memoryless channels (B-DMC) with polynomial dependence on the gap to capacity. Notably, polar codes have encoding and decoding complexity, which makes them practical for many applications.

Channel polarization, introduced by Arıkan [1], is a phenomenon by which, given a binary-input discrete memoryless channel, virtual channels between the bits at the input of a linear encoder and the channel output sequence are created, such that the mutual information in each of these channels converges to either zero or one as the code length tends to infinity; the proportion of channels with mutual information close to one converges to the original channel's mutual information. These virtual channels are created by recursively applying channel combining and splitting steps. Polar codes of rate R = K/N are linear codes whose generator matrix is such that its rows induce the K virtual channels with highest mutual information among all N possible channels.
The scheme behaves as if uncoded bits were sent through these channels. This construction, together with polarization, explicitly gives a code of rate close to the mutual information of the channel with vanishing error probability. We propose a different construction of polar codes. Instead of choosing the best K virtual channels, we choose all channels whose mutual information is above a certain threshold which might depend on the code length. This new construction is shown to preserve the capacity-achieving property of Arıkan's original construction as long as the threshold function is bounded appropriately. This construction induces accurate closed-form upper and lower bounds to the minimum distance of the resulting codes when the design channel is the binary erasure channel (BEC). Our results sharpen existing bounds in the literature on the minimum distance of polar codes [2].

II. LITERATURE SURVEY

In his seminal work of 1948, Shannon had characterized the highest rate (speed of transmission) at which one could reliably communicate over a discrete memoryless channel (a noise model); he called this limit the capacity of the channel. However, he used a probabilistic method in his proof and left open the problem of reaching this capacity with coding schemes of manageable complexities. In the 90's, codes were found (turbo codes and LDPC rediscovered) with promising results in that direction. However, mathematical proofs could only be provided for few specific channel cases (pretty much only for the so-called binary erasure channel). In 2008, ErdalArıkan **at**Bilkent University inventedpolar codes, providing a new mathematical framework to solve this problem.

Besides allowing rigorous proofs for coding theorems, an important attribute of polar codes is, in my opinion, that they bring a new perspective on how to handle randomness (beyond the channel coding problem). Indeed, after a couple of years of digestion of Arıkan's work, it appears that there is a rather general phenomenon underneath the polar coding idea. The technique consist in applying a specific linear transform, constructed from many Kronecker products of a well-chosen small matrix, to a high-dimensional random vector (some assumptions are required on the vector distribution but let's keep a general framework for now). The polarization phenomenon, if it occurs, then says that the transformed vector can be split into two parts (two groups of components): one of maximal randomness and one of minimal one (nearly deterministic). The polarization terminology comes from this antagonism. We will see below a specific example. But a remarkable point is that the separation procedure as well as the algorithm that reconstructs the original vector from the purely random components have low complexities (nearly linear). On the other hand, it is still an open problem to characterize mathematically if a given component belongs to the random or deterministic part. But there exist tractable algorithms to figure this out accurately.

III. EXISTING SCHEME

The fast Fourier transform (FFT) is one of the most important algorithms in the field of digital signal processing. It is used to calculate the discrete Fourier transform (DFT) efficiently. In order to meet the high performance and realtime requirements of modern applications, hardware designers have always tried to implement efficient architectures for the computation of the FFT. In this context, pipelined hardware architectures [1]–[24] are widely used, because they provide high throughputs and low latencies suitable for real time, as well as a reasonably low area and power consumption. There are two main types of pipelined architectures: feedback (FB) and feedforward (FF).

On the one hand, feedback architectures [1]–[14] are characterized by their feedback loops, i.e., some outputs of

ISSN (Online) 2394-6849
International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 3, Issue 8, August 2016

the butterflies are fed back to the memories at the same stage. Feedback architectures can be divided into Single-path Delay Feedback (SDF) [1]–[6], which process a continuous flow of one sample per clock cycle, and Multi-path Delay Feedback (MDF) or parallel feedback [7]– [14], which process several samples in parallel. On the other hand, feedforward architectures [4], [5], [15]–[19], also known as Multi-path Delay Commutator (MDC) [4], do not have feedback loops and each stage passes the processed data to the next stage. These architectures can also process several samples in parallel.

In current real-time applications, the FFT has to be calculated at very high throughput rates, even in the range of GSamples/s. These high-performance requirements appear in applications such as Orthogonal Frequency Division Multiplexing (OFDM) [9]–[12], [22] and Ultra Wideband (UWB) [10]– [13]. In this context two main challenges can be distinguished. The first one is to calculate the FFT of multiple independent data sequences [22], [23]. In this case, all the FFT processors can share the rotation memory in order to reduce the hardware [22]. Designs that manage a variable number of sequences can also be obtained [23].

The second challenge is to calculate the FFT when several samples of the same sequence are received in parallel. This must be done when the required throughput is higher than the clock frequency of the device. In this case it is necessary to resort to FFT architectures that can manage several samples in parallel. As a result, parallel feedback architectures, which had not been considered for several decades, have become very popular in the last few years [8]–[14].

Conversely, not very much attention has been paid to feedforward (MDC) architectures. This paradoxical fact, however, has a simple explanation. Originally, SDF and MDC architectures were proposed for radix-2 [2], [17] and radix-4 [3], [17]. Some years later, radix- 2 2 was presented for the SDF FFT [4] as an improvement on radix-2 and radix-4. Next, radix-2 3 and radix-2 4 , which enable certain complex multipliers to be simplified, were also presented for the SDF FFT. An explanation of radix-2 k SDF architectures can be found in [6]. Finally, the current need for high throughput has been meet by the MDF, which includes multiple interconnected SDF paths in parallel. However, radix- 2 k had not been considered for feedforward architectures until the first radix-2 2 feedforward FFT architectures were introduced a few years ago [24].

Given a 2N point sequence, x(n), and having taken the FFT of x(2n)+jx(2n+1) for

n=0,1,...,N-1, we can now compute:

Given FFT(x(2n)+jx(2n+1)) = A(k)+jF(k),

let X(k) = R(k) + jI(k),

let c(k) = cos(pi*k/N),

let s(k) = sin(pi*k/N),

2R(k) = A(k)+A(N-k) + c(k)(F(k)+F(N-k)) - s(k)(A(k)-A(N-k))

2I(k) = F(k)-F(N-k) - s(k)(F(k)+F(N-k)) - c(k)(A(k)-A(N-k))

This will give us X(k). Notice what happens when we let k'=N-k (k'-> k for convenience):

2R(N-k) = A(k)+A(N-k) - c(k)(F(k)+F(N-k)) + s(k)(A(k)-A(N-k)) (1)

2I(N-k) = -F(k)+F(N-k) - s(k)(F(k)+F(N-k)) - c(k)(A(k)-A(N-k)) (2) because c(N-k) = -c(k) and s(N-k) = s(k).

Since the data needed to compute X(k) is the same as the data needed to compute X(N-k), we decided to compute them simultaneously during each iteration of the conversion stage loop. So the algorithm does the conversion in k, N-k pairs (except for the mid-point).

In order to calculate the inverse, we realize that we have four unknowns and four equations. Thus it is a simple matter to derive:

2A(k) = R(k)+R(N-k) - s(k)(R(k)-R(N-k)) - c(k)(I(k)+I(N-k))

2F(k) = I(k)-I(N-k) + c(k)(R(k)-R(N-k)) - s(k)(I(k)+I(N-k)) (3) and

ISSN (Online) 2394-6849
International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 3, Issue 8, August 2016

2A(N-k) = R(k)+R(N-k) + s(k)(R(k)-R(N-k)) + c(k)(I(k)+I(N-k)) (4)

2F(N-k) = -I(k)+I(N-k) + c(k)(R(k)-R(N-k)) - s(k)(I(k)+I(N-k)) (5)

We can, therefore, calculate A(k)+jF(k) from X(k), run the inverse FFT and regain x(n).

IV. PROPOSING SCHEME

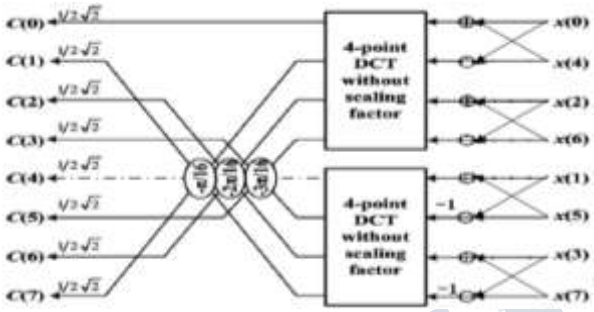

Figure 1: 8 bit DCT configuration

Discrete cosine transform (DCT) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are important to numerous applications in science and engineering, from loss compression of audio and images to spectral methods for the numerical solution of partial differential equations. The use of cosine rather than sine functions is critical in these applications: for compression, it turns out that cosine functions are much more efficient (as described below, fewer functions are needed to approximate a typical signal), whereas for differential equations the cosines express a particular choice of boundary conditions.

In particular, a DCT is a Fourier-related transform similar to the discrete Fourier transform (DFT), but using only real numbers. DCTs are equivalent to DFTs of roughly twice the length, operating on real data with even symmetry (since the Fourier transform of a real and even function is real and even), where in some variants the input and/or output data are shifted by half a sample. There are eight standard DCT variants, of which four are common. The most common variant of discrete cosine transform is the type-II DCT, which is often called simply "the DCT", its inverse, the type-III DCT, is correspondingly often called simply "the inverse DCT" or "the IDCT".

Two related transforms are the discrete sine transforms (DST), which is equivalent to a DFT of real and odd functions, and the modified discrete cosine transforms (MDCT), which is based on a DCT of overlapping data. Like any Fourier-related transform, discrete cosine transforms (DCTs) express a function or a signal in terms of a sum of sinusoids with different frequencies and amplitudes. Like the discrete Fourier transforms (DFT), a DCT operates on a function at a finite number of discrete data points. The obvious distinction between a DCT and a DFT is that the former uses only cosine functions, while the latter uses both cosines and sines (in the form of complex exponentials). However, this visible difference is merely a consequence of a deeper distinction: a DCT implies different boundary conditions than the DFT or other related transforms. The Fourier-related transforms that operate on a function over a finite domain, such as the DFT or DCT or a Fourier series, can be thought of as implicitly defining an extension of that function outside the domain. That is, once you write a function as a sum of sinusoids, you can evaluate that sum at any , even for where the original was not specified. The DFT, like the Fourier series, implies a periodic extension of the original function. A DCT, like a cosine transform, implies an even extension of the original function. DCT, like a cosine transform, implies an even extension of the original function. Illustration of the implicit even/odd extensions of DCT input data, for N=11 data points (red dots), for the four most common types of DCT (types I-IV). However, because DCTs operate on finite, discrete sequences, two issues arise that do not apply for the continuous cosine transform.

Before starting hardware assembling, we use software application to simulate the algorithm. The purpose of using software application for simulation, it is to verify the correctness of the logic function. Skipping this step will result in frustration in finding errors during hardware assembling. It is extremely difficult to locate the error during the VLSI design. The following is the software application using Java language to manipulate the DCT algorithm.

The main function of the program, at first to create a 2-D arrays for substitute the 2-D matrix in the algorithm. The four for loops is for implement the 2 summations in the algorithm. The method shown above performs a DCT algorithm on the 2-D array. Use the output of the array, and

ISSN (Online) 2394-6849
International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 3, Issue 8, August 2016

then perform transposition. The array after transposition will be use for DCT algorithm for row transformation. The output after this operation is the final result after the data compression. However, the transposition method is not complete due insufficient knowledge of how transposition works in the DCT algorithm. In the coming semester, fully understand the purpose of transposition is the first priority. After all the key components figure out, the entire VLSI architecture design will be design in details.

$$\text{Input matrix} = \begin{bmatrix} 48 & 123 & 89 & 175 & 234 & 56 & 78 & 190 \\ 1 & 5 & 67 & 2 & 98 & 3 & 125 & 7 \\ 67 & 89 & 155 & 43 & 27 & 89 & 0 & 129 \\ 57 & 23 & 51 & 93 & 187 & 67 & 189 & 1 \\ 178 & 45 & 39 & 52 & 12 & 78 & 49 & 90 \\ 48 & 56 & 120 & 87 & 65 & 34 & 167 & 203 \\ 23 & 32 & 44 & 45 & 47 & 23 & 90 & 89 \\ 54 & 67 & 90 & 128 & 235 & 50 & 190 & 210 \end{bmatrix}$$

The following is the matrix after the DCT algorithm

$$\text{Output matrix after DCT} = \begin{bmatrix} 663 & -127 & -4 & -66 & 68 & 2 & -54 & 115 \\ -31 & 67 & -69 & 16 & -16 & -44 & 14 & -2 \\ 86 & -76 & -64 & -76 & 63 & -83 & -73 & 37 \\ 17 & -3 & 50 & -35 & 41 & 7 & -7 & -45 \\ 143 & -11 & -57 & 75 & 97 & -9 & -65 & 5 \\ 24 & -6 & -63 & -9 & 6 & -23 & -30 & -19 \\ 165 & -18 & 27 & -110 & 33 & -63 & 51 & -37 \\ -41 & 101 & 83 & -13 & 19 & -49 & 120 & -110 \end{bmatrix}$$

The following matrix is the output after IDCT algorithm

$$\text{Output matrix after IDCT using DCT matrix} = \begin{bmatrix} 48 & 123 & 89 & 175 & 234 & 56 & 78 & 190 \\ 1 & 5 & 67 & 2 & 98 & 3 & 125 & 7 \\ 67 & 89 & 155 & 43 & 27 & 89 & 0 & 129 \\ 57 & 23 & 51 & 93 & 187 & 67 & 189 & 1 \\ 178 & 45 & 39 & 52 & 12 & 78 & 49 & 90 \\ 48 & 56 & 120 & 87 & 65 & 34 & 167 & 203 \\ 23 & 32 & 44 & 45 & 47 & 23 & 90 & 89 \\ 54 & 67 & 90 & 128 & 235 & 50 & 190 & 210 \end{bmatrix}$$

By compare the input matrix and the IDCT matrix, it is confident that the algorithm and software is correct. Because the IDCT matrix is suppose to be the same as the input matrix. IDCT is to use DCT matrix to go back to the original matrix. This way is to double-check the simulation process. However, this software simulation does not include transposition part. According to the output result acquired from Sachidanandan [52]. The simulation includes DCT and transposition. The following is the output result in the paper, which it is the result we are expecting to have.

The coming semester will insert the transposition into the Java code and hopefully the output will be the same as the matrix above.

V. RESULTS

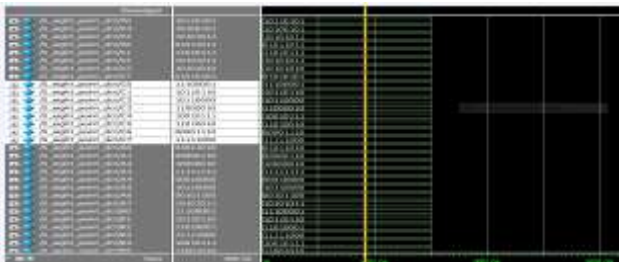

Figure 2: Simulation Result for 32 Bit Data.

Figure 2 results in DCT based 32bit polar codes were developed for encoding data bits based on discrete cosine transform algorithm which possess on both real and complex numbers.

VI. CONCLUSION

A good example makes the advantage apparent: no matter what value the concurrent number M design is able to finish encoding all words before its counterpart (overlapped design) outputs its first decoded word. Another point is the IGC, which is inspired by the DCT processor proposed and can be generated with a nice and easy recurrence relationship, is able to output all control bits required by the multiplexers on the fly. Therefore, no additional clock cycles are needed for computation of , which preserves the advantage of short latency. To the best knowledge of the authors, this is the first detailed design of similar module with such features.

REFERENCES

[1] E. Abbe, “Extracting randomness and dependencies via a matrix polarization,” arXiv:1102.1247v1, 2011.

[2] E. Abbe and E. Telatar, “Polar codes for the m-user MAC and matroids,” arXiv:1002.0777v2, 2010.

[3] E. Arıkan, “Channel polarization: A method for constructing capacityachieving codes for symmetric binary-input memoryless channels,” IEEE Trans. Inform. Theory, vol. 55, pp. 3051–3073, 2009.

[4] E. Arıkan, "Source polarization," arXiv:1001.3087v2, 2010.

[5] E. Arıkan and E. Telatar, "On the rate of channel polarization," in Proc. IEEE Symp. Inform. Theory, Seoul, South Korea, 2009, pp. 1493–1495.

[6] D. Burshtein and A. Strugatski, "Polar write once memory codes," arXiv:1207.0782v2, 2012.

[7] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms, 2nd ed. Cambridge, Massachusetts: The MIT Press, 2001.

[8] T.M. Cover and J.A. Thomas, Elements of Information Theory, 2nd ed. New York: John Wiley, 2006.

[9] R.G. Gallager, Information Theory and Reliable Communications. New York: John Wiley, 1968.

[10] A. Goli, S.H. Hassani, and R. Urbanke, "Universal bounds on the scaling behavior of polar codes," in Proc. IEEE Symp. Inform. Theory, Cambridge, Massachusetts, 2012, pp. 1957–1961.

[11] V. Guruswami and P. Xia, "Polar codes: Speed of polarization and polynomial gap to capacity," http://eccc.hpi-web.de/report/2013/050/, 2013.

[12] S.H. Hassani, R. Mori, T. Tanaka, and R. Urbanke, "Rate-dependent analysis of the asymptotic behavior of channel polarization," arXiv:1110. 0194v2, 2011.

[13] S.B. Korada, "Polar codes for channel and source coding," Ph.D. dissertation, EcolePolytechnique Fed´ erale de Lausanne, 2009.

[14] S.B. Korada, E. S¸ as¸oglu, and R. Urbanke, "Polar codes: Characterization ˘ of exponent, bounds, and constructions," IEEE Trans. Inform. Theory , vol. 56, pp. 6253–6264, 2010.

[15] B.M. Kurkoski and H. Yagi, "Quantization of binary-input discrete memoryless channels with applications to LDPC decoding," arXiv:11107. 5637v1, 2011.

[16] H. Mahdavifar and A. Vardy, "Achieving the secrecy capacity of wiretap channels using polar codes," IEEE Trans. Inform. Theory, vol. 57, pp. 6428–6443, 2011.

[17] R. Mori, "Properties and construction of polar codes," Master's thesis, Kyoto University, arXiv:1002.3521, 2010.

[18] R. Mori and T. Tanaka, "Performance and construction of polar codes on symmetric binary-input memoryless channels," in Proc. IEEE Symp. Inform. Theory, Seoul, South Korea, 2009, pp. 1496–1500.

[19] R. Pedarsani, S.H. Hassani, I. Tal, and E. Telatar, "On the construction of polar codes," in Proc. IEEE Symp. Inform. Theory, Saint Petersburg, Russia, 2011, pp. 11–15.

[20] T. Richardson and R. Urbanke, Modern Coding Theory. Cambridge, UK: Cambridge University Press, 2008.

[21] E. S¸ as¸oglu, "Polarization in the presence of memory," in ˘ Proc. IEEE Symp. Inform. Theory, Saint Petersburg, Russia, 2011, pp. 189–193.

[22] E. S¸ as¸oglu, E. Telatar, and E. Arıkan, "Polarization for arbitrary discrete ˘ memoryless channels," arXiv:0908.0302v1, 2009.

[23] E. S¸ as¸oglu, E. Telatar, and E. Yeh, "Polar codes for the two-user multiple- ˘ access channel," arXiv:1006.4255v1, 2010.

[24] I. Tal and A.Vardy, "List decoding of polar codes," in Proc. IEEE Symp. Inform. Theory, Saint Petersburg, Russia, 2011, pp. 1–5.