A High Throughput List Decoder Architecture For LDPC Codes

Promodini.V[1], Sangeetha.K[2] and Natraj.N. A[3]

[1] *UG Student, Dept. of ECE, Prince Shri Venkateshwara Padmavathy Engineering College, Tamilnadu, India*

[2]*UG Student, Dept. of ECE, Prince Shri Venkateshwara Padmavathy Engineering College, Tamilnadu, India*

[3]*Assistant professor, Dept. of ECE, Prince Shri Venkateshwara Padmavathy Engineering College, Tamilnadu, India*

ABSTRACT

In the field of CMOS technology. LDPC codes are used for the encoding and decoding process. Encoding is the process of putting a sequence of characters (letters, numbers, punctuation, and certain symbols) into a specialized format for efficient transmission or storage. Decoding is the opposite process -- the conversion of an encoded format back into the original sequence of characters. Encoding and decoding are used in data communications, networking, and storage. The term is especially applicable to radio (wireless) communications systems. LDPC codes are especially used to improve its throughput, thereby better performance can be achieved.

Keyword: - *LDPC encoder, LDPC decoder*

1. INTRODUCTION

One of the key underlying technologies in our increasingly connected world is the method for efficiently communicating discretized information over a physical medium such as telephone lines, optical cables, radio links, or magnetic storages. Channel coding plays an integral role in providing a reliable communication method that can overcome signal degradation in practical channels. Turbo codes, invented by Berrou, Glavieux and Thitimajshim in 1993, are the first known capacity approaching error correction code that provides a powerful error correction capability when decoded by an iterative decoding algorithm. More recently, research efforts toward searching for lower complexity codes and iterative decoding led to the rediscovery of low density parity check (LDPC) code, which was originally proposed by Gallager in 1960 and was later generalized as MacKay- Neal code. The LDPC codes have been shown to achieve near-optimal performance in additive white Gaussian noise channels when decoded with the sum-product (SP) algorithm. LDPC codes have several advantages over turbo codes. While it is difficult to apply parallelism in the decoding of turbo code due to the sequential nature of the decoding algorithm, LDPC decoding can be Performed

with a high degree of parallelism to achieve a very high decoding throughput. LDPC codes do not require a long interleaver, which causes a large delay in turbo codes. LDPC codes can be directly designed for a desired code rate while turbo codes, that are based on convolutional codes. The parity check matrix H of a code with quasi-cyclic property can be put into an array of circulant matrices with column and row rearrangements. The LDPC codes with the quasi-cyclic property are called QC-LDPC codes. Because of its practical importance, most of the LDPC decoder architectures in the literature have been designed for QC-LDPC codes or its subclass, in which the

array structure can be exploited for more efficient implementation. Among various array-structured LDPC codes, a class of codes of which the H-matrix is an array

of cyclic permutation matrices, have been of particular interest since the balanced partitioning of 1's into submatrices facilitates the design of a highly parallelized decoders.

The main focus has been done on the decoder side to increase its efficiency as well as to improve its working model by attaining increased throughput. To achieve these above-mentioned specifications, it is required to follow the steps given in the decoder side.

The carryover of this paper is methodized as follows. Related work is reviewed in Part 2. Part 3 brings up the layout of our system. Working is explained in Part 4 and 5. Then the overall block diagram is explained in part 6. The simulated results are viewed at part 7 and its corresponding conclusions are seen at part 8. Finally, references are seen at part 9.

2. RELATED WORK

Hereby there are many encoding and decoding process which are related to the model which has been derived at recent times. Polar codes are a significant breakthrough in a coding theory, since they can achieve the channel capacity of binary-input symmetric memoryless channels and arbitrary discrete memoryless channels. Polar codes of block length N can be efficiently decoded by a successive-cancellation (SC) algorithm with a complexity of $O(N \log N)$. While polar codes of a very large block length approach the capacity of underlying channels under the SC algorithm, for short or moderate polar codes, the error performance of the SC algorithm is worse than turbo or low-density parity-check codes. Lots of efforts have already been devoted to the improvement of error performance of polar codes with short or moderate lengths. An SC list (SCL) decoding algorithm performs better than the SC algorithm. In the cyclic redundancy check (CRC) is used to pick the output codeword from L candidates, where L is the list size. The CRC-aided SCL (CA-SCL) decoding algorithm performs much better than the SCL decoding algorithm at the expense of negligible loss in a code rate. Despite its significantly improved error performance, the hardware implementations of SC-based list decoders still suffer from long decoding latency and limited throughput due to the serial decoding schedule. In order to reduce the decoding latency of an SC-based list decoder, M $(M > 1)$ bits are decoded in parallel, where the decoding speed can be improved by M times ideally. However, for the hardware implementations of the algorithms in the actual decoding speed improvement is less than M times due to extra decoding cycles on finding the L most reliable paths among $2ML$ candidates, where L is the list size. A software adaptive simplified SC (SSC)-list-CRC decoder was proposed. For a (2048, 1723) polar + CRC-32 code, the SSC-list-CRC decoder with $L = 32$ was shown to be about seven times faster than an SC-based list decoder. However, it is unclear whether the list decoder in is suitable for hardware implementation. RLLD algorithm is proposed to reduce the decoding latency of SC list decoding for polar codes. For a node v, let Iv denote the total number of leaf nodes that are associated with information bits the RLLD algorithm performs the SC-based list decoding on Gn and follows the node activation schedule, except when a certain type of nodes is activated. These nodes calculate and return the codewords to their parent nodes while updating the decoding paths and their metrics, without activating their child nodes. the channel message memory (CMEM) stores the received channel LLRs, and

the internal LLR message memory (IMEM) stores the LLRs generated during the SC computation process. With the concatenation and split method, the IMEM is implemented with area efficient memories, such as RF or SRAM.

3. SYSTEM OVERVIEW

The system we proposed is a personalized itinerary format to perform the encoding and decoding process. For the given word read from a memory protected with one step MLD EG-LDPC codes, and affected by up to four bit-flips, all errors can be detected upto twenty decoding cycles errors affecting more than five bits were detected with a probability very close to one. The probability of undetected errors was also found to decrease as the code block length increased. This may be sufficient for some application with very little additional circuitry as the decoding circuitry is also used for error detection. low density parity check encoding can be implemented serially with simple hardware but requires a large decoding time. For memory applications, this increases the memory access time. Whenever a data get corrupted or lost due to transient problem using the high efficient LDPC decode algorithm we can retrieve the original information to be stored into the memory using parity bits.

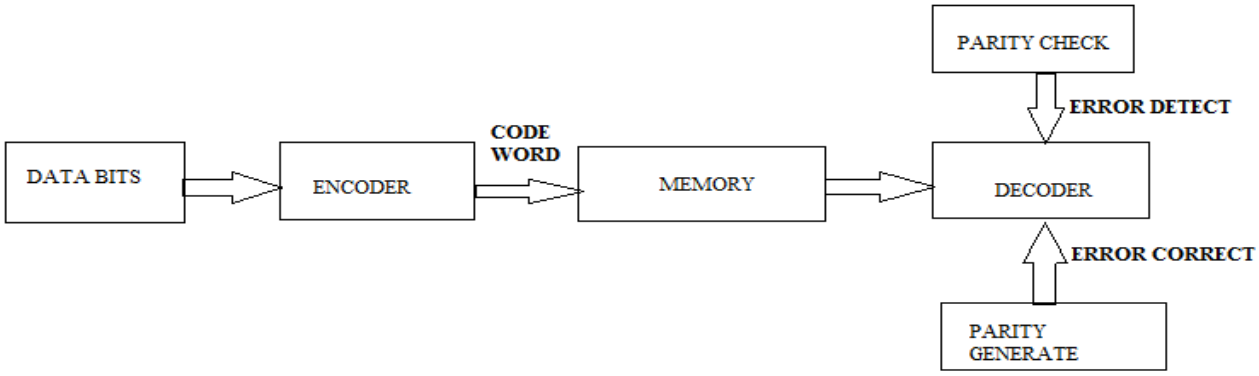

Fig-1: Overall block Diagram

4. LDPC ENCODER

A low-density parity-check (LDPC) code is defined by a parity- check matrix that is sparse. A regular (n,k) LDPC code is defined by an (n- k) × n parity check matrix with n- block length of the code and k information bits generated by the binary source. There are kinds of LDPC codes regular and irregular, irregular performs better than regular but regular codes are easy to implement. Take a special case of LDPC codes as cyclic codes which is used to construct the parity check code and study the behaviour.

Construction of parity check matrix is the important part of Encoding process. The block-circulant LDPC code construction is used for creating parity check matrix.

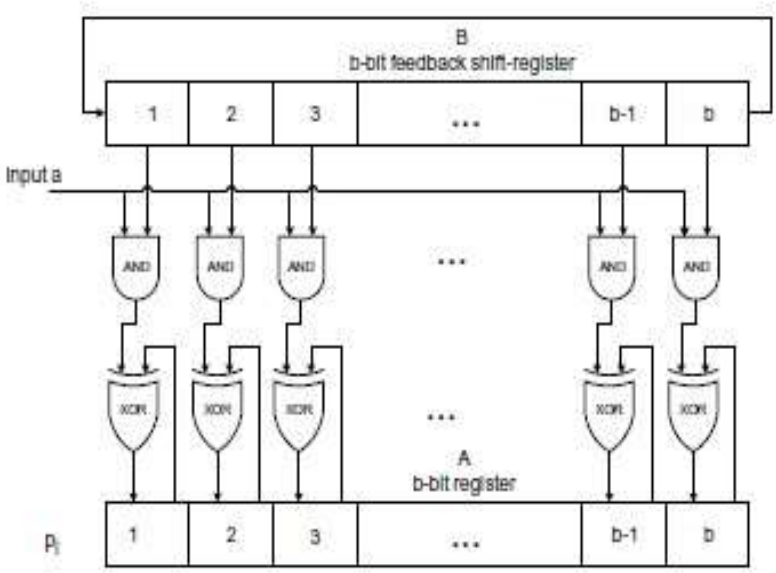

Fig-2: Encoding Process

There are three methods for constructing the generator matrices of QCLDPC codes in systematic-circulant form from their parity-check matrices for two different cases. The first case is that the rank of the parity-check matrix H_{qc}, denoted r, is equal to the number cb of rows of H_{qc}, i.e., r = cb. The second case is that r < cb. encoding process and the expression given by the j^{th} parity check section p_j can be formed with a shift-register-adder-accumulator (SRAA)- circuit. At the beginning of the first step, $g(0)_{1,j} = g_{1,j}$ is stored in the feedback shift register B and the content of register A (accumulator) is set to zero. When the information bit a1 is shifted into the encoder and the channel, the product $a_1g(01_{,j}$ is formed at the outputs of the AND-gates and is added to the content stored in the register A (zero at this time). The sum is then stored in the register A. The feedback register B is shifted once to the right. The new content in B is $g(1)_{1,j}$. When the next information bit a2 is shifted into the encoder, the product $a_2g(1)_{1,j}$ is formed at the outputs of the AND-gates. This product is then added to the sum $a_1g(0)$ 1,j in the accumulator register A. The sum $a_1g(0)_{1,j} + a_2g(1)_{1,j}$ is then stored in A. The above shift-add store process continues. When the information bit ab has been shifted into the encoder, register A stores the partial sum $a_1G_{1,j}$, which is the contribution to the parity section pj from the information section a1. At this time, the generator $g_{2,j}$ of the circulant $G_{2,j}$ is loaded into B. The shift add-store process repeats. When the information section a2 has been completely shifted into the encoder, register A contains the accumulated sum $a_1G_{1,j} + a_2G_{2,j}$, which is the contribution to the parity section pj from the first two information sections, a1 and a2. The above process repeats until the entire information sequence a has been shifted into the encoder. The next stage is to form the c parity sections based on $p_{Tj} = B_j y_T$. This can be done with another c banks of XOR-gates. If the parity-check bits of each parity section are generated serially one bit at a time, simply cyclically shift the buffer registers, BR_1, . . ., $BR_{c,b}$ times (left shift). The parity-check bits are generated in the same manner as the bits of y-vector.

Consider the case for which the rank r of the c×c array D is equal to the rank of H_{qc} given by (1) and r < cb Based on the generator matrix G_{qc} given by, an encoder with two sub-encoders can be implemented. The first sub-encoder is

implemented based on the submatrix G_{qc} and the second one is implemented based on the submatrix Q. An information sequence a of tb − r bits are divided into two parts, a(1) and a(2), where a(1) consists of the first (t −c)b information bits and a(2) consists of the last cb−r information bits. The first sub encoder encodes a(1) into a codeword in the subcode generated by G_{qc} and the second sub- encoder encodes a(2) into a codeword in the subcode generated by Q. Adding the outputs of the two sub-encoders, obtain the codeword for the information sequence a. The first sub encoder can be implemented in the same way as described for previous case. The second sub-encoder can be implemented as a conventional encoder for a linear block code.

5. LDPC DECODER

While the powerful error correction capability of the LDPC codes has drawn a lot of research interests in the aspect of code performance, the availability of the highly parallelizable decoding algorithm has brought as much interest to the design of efficient hardware implementation. Besides the use of the decoding hardware for the deployment in practical systems, another important usage is to evaluate a given code, usually as part of the design process. Although there exists an approximate analytical method called density evolution to predict the performance of LDPC code with iterative decoding algorithm, it has to rely on the assumption that the code word length tends to infinity to make the graph essentially cycle-free .

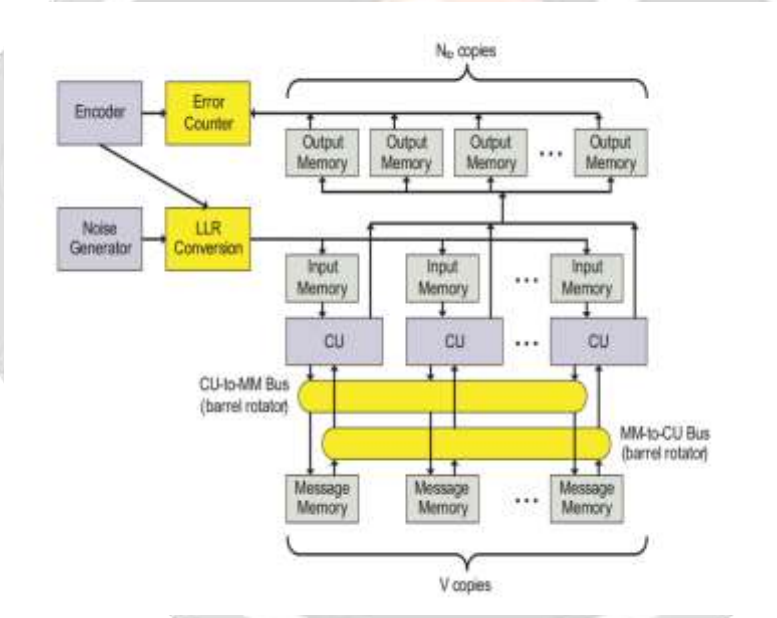

Fig-3: The overall block diagram of decoder.

The decoder consists of the input memory to store the input LLR data, computation units to perform bit/check node operations, the message memory to store the messages, the output memory to store the decoder output, and the error counter. In addition, there is a register block for communication with an external device. In our FPGA implementation, use an embedded processor to give commands and fetch results through the register block. The overall block diagram is shown in Figure 4.1. The pre-synthesis parameters that determine the amount of logic and memory used for the decoder are as follows.

• Algorithm: the SP or MS/MMS algorithm

• Data precision for message representation (integer part and fractional part)

• The parallelization factor V (even)

• The maximum possible values for N_c, N_b, P, j, and k

• The maximum possible value for⌈P/V⌉+ 1

• The maximum possible value for codeword length N

• The maximum possible value for the number of N_{ph}, where N_{ph} = max (largest column weight) $*N_b$, (largest row weight) $*N_c$

• The maximum possible value for the number of check equations M

• The maximum possible value for the number of nonnegative elements in the extended S Matrix N_s

• The depth of each input and output memory blocks IOM_d

• The depth of each message memory blocks MM_d

• The maximum number of message bit errors per codeword that can be counted N_{err}

• The maximum number of iterations N_{iter}

• Total number of intermediate checkpoints N_{cp} (to see the decoding results at different number of iterations) While the pre-synthesis parameters define a range of code parameters that can be supported by the synthesized hardware, the actual code parameters can be reprogrammed by writing to the register block. The post-synthesis parameters that can be modified at run time are as follows.

• The number of iterations

• Whether to enable early stopping when a valid codeword is found

• Code parameters N_c, N_b, p, j and k

• The shift values for the extended S-matrix

• The coefficients of the look-up table for the F() transform function block (for SP only).

• Scaling factor for the check node output (for MMS algorithm only)

• A list of intermediate checkpoints

In the decoder, however, each check or bit node operation is performed in a sequential manner, i.e., one operand is processed at a time. This scheme makes the design of the computation units independent from the actual code parameters since any larger weight can be supported by increasing the number of clock cycles for each bit or check node operation. Also, since the computation unit is not designed for a specific number of operands, it is possible to design a shared bit/check computation unit that works as a BCU during the first half iteration and as a CCU during the second half iteration.

6.OVERALL WORKING OF LDC CODES

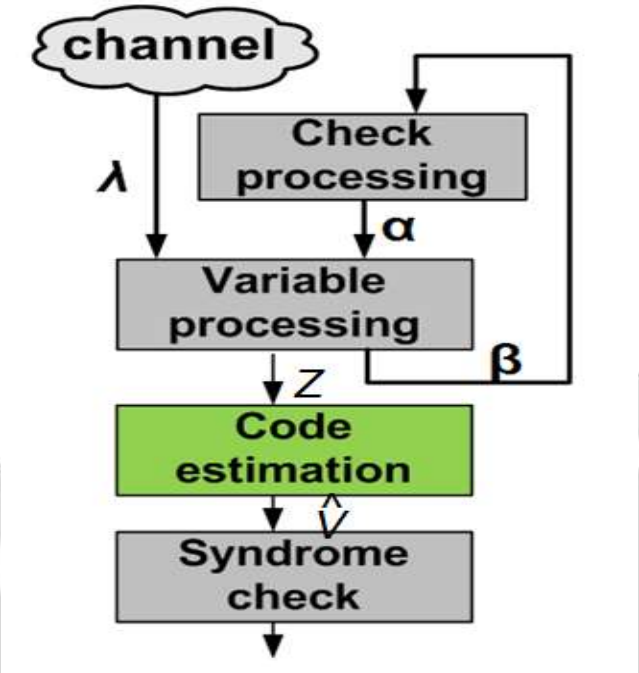

Fig-4: Flow diagram of the decoding process.

Figure 4 describes the operation done by the decoding part. It consists of four blocks, they are, (i) the variable processing block, (ii) the check processing block, (iii) the code estimation block and (iv) the syndrome check block. The encoded information along with the redundant bits i.e., noise from the channel, λ is sent for variable processing which gets another input α from the check processing block. Initially the output from the check processing is null, it gets assigned as the iteration increases. The output from the variable processing block β is sent to the check processing operation and the final result Z, the decoded output.

7.SIMULATED RESULTS

The simulated results of the information bits run in Model Sim software. A sixteen-bit information is sent in order to perform the LDPC decoding operation. The program is coded in such a way that iteration i.e., syndrome check is performed for twenty times, after which it gets out of the loop indicating that the coded information bit could not obtained within twenty iterations. As parallel decoder is used the number of cycles used is comparatively less. Hence throughput have been increased from 3.8 to 8.2 times better than polar codes using RLLD algorithm. In this encoding and decoding operation to ensure proper result condition are given and checked, the conditions are

- The sixteen bit 01s are encoded and, error bits are added at the channel and it is corrected in two iterations.
- The sixteen bit 110 is encoded and sent through the channel which takes more than twenty iterations and hence the program is not able to

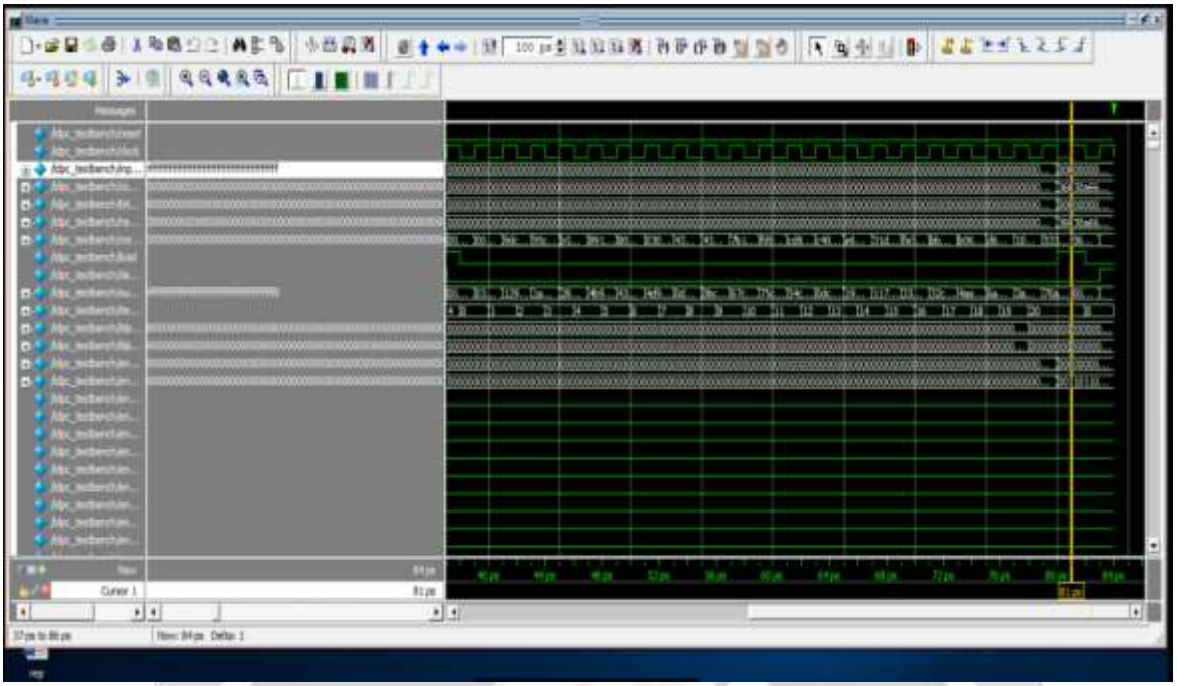

Figure 5 : Result

Determine the original value which is encoded, so it comes out of loop.

8.CONCLUSIONS

The iterative decoding approach is already used in turbo codes but the structure of LDPC codes give even better results. In many cases, they allow a higher code rate and also a lower error floor rate. The inherent parallelism in decoding LDPC codes suggests their use in high data rate systems. They provide a performance which is very close to the capacity for a lot of different channels and linear time complex algorithms for decoding. Furthermore, are they suited for implementations that make heavy us of parallelism. Quasi-cyclic (QC) low-density parity-check (LDPC) codes form an important subclass of LDPC codes. These codes have encoding advantage over the other types of LDPC codes. Also, well designed QC-LDPC codes perform as well as computer generated random LDPC codes in terms of bit-error performance, block-error performance, error-floor, and rate of iterative decoding convergence, collectively. Thereby a sustainable amount of throughput can be achieved using LDPC codes when compared with the RLLD algorithm , which can increase the performance of the encoding and decoding process.

9. 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, Jul. 2009.

[2] E. Sasoglu, I. E. Teltar, and E. Arıkan, "Polarization for arbitrary discrete memoryless channels," in Proc. IEEE Inf. Theory Workshop, Seoul, Korea, Oct. 2009, pp. 144–148.

[3] 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.

[4] I. Tal and A. Vardy, "List decoding of polar codes," IEEE Trans. Inf. Theory, vol. 61, no. 5, pp. 2213–2226, May 2015.

[5] K. Niu and K. Chen, "CRC-aided decoding of polar codes," IEEE Commun. Lett., vol. 16, no. 10, pp. 1668–1671, Oct. 2012.

[6] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," IEEE Commun. Lett., vol. 16, no. 12, pp. 2044–2047, Dec. 2012.

[7] A. Balatsoukas-Stimming, A. J. Raymond, W. J. Gross, and A. Burg, "Hardware architecture for list successive cancellation decoding of polar codes," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 61, no. 8, pp. 609–613, Aug. 2014.

[8] J. Lin and Z. Yan, "An efficient list decoder architecture for polar codes," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 11, pp. 2508–2518, Nov. 2015.

[9] B. Yuan and K. K. Parhi, "Successive cancellation list polar decoder using log-likelihood ratios," in Proc. 48th IEEE Asilomar Conf. Signal, Syst. Comput., Nov. 2014, pp. 548–552.

[10] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," IEEE Trans. Signal Process., vol. 63, no. 19, pp. 5165–5179, Oct. 2015. [Online].

[11] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP), Florence, Italy, May 2014, pp. 3903–3907.

[12] B. Li, H. Shen, and D. Tse. (Sep. 2013). "Parallel decoders of polar codes." [Online].

[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., vol. 23, no. 10, pp. 2268–2280, Oct. 2015.

[14] C. Xiong, J. Lin, and Z. Yan, "Symbol-decision successive cancellation list decoder for polar codes," IEEE Trans. Signal Process., vol. PP, no. 99. DOI: 10.1109/TSP.2015.2486750

[15] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Increasing the speed of polar list decoders," in Proc. IEEE Workshop Signal Process. Syst. (SiPS), Belfast, U.K., Oct. 2014, pp. 1–6.

[16] H. Yoo and I.-C. Park, "Partially parallel encoder architecture for long polar codes," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 62, no. 3, pp. 306–310, Mar. 2015.

[17] Y. Huo, X. Li, W. Wang, and D. Liu, "High performance table-based architecture for parallel CRC calculation," in Proc. IEEE Int. Workshop Local Metropolitan Area Netw. (LANMAN), Apr. 2015, pp. 1–6.

Promodini.V is currently pursuing her B.E degree in Prince Shri Venkateshwara Padmavathy Engineering College. She is interested in Very large scale industry and Computer Networks. She has authored or coauthored papers.

Sangeetha.K is currently pursuing her B.E degree in Prince Shri Venkateshwara Padmavathy Engineering College. She is interested in Very Large scale integration. She has authored or coauthored papers.

.