RESEARCH INVENTY: International Journal of Engineering and Science
ISSN: 2278-4721, Vol. 1, Issue 7 (November 2012), PP 36-41
www.researchinventy.com

Design of LDPC Decoder Using FPGA: Review of Flexibility

[1]Asisa Kumar Panigrahi, [2]Ajit Kumar Panda
[1,2] *(Department of Electronics & Communication Engg*
National Institute of Science & Technology
Brahmapur, India)

Abstract - *Low Density Parity Check (LDPC) codes have become very popular now-a-days because of their shannon limit approaching error correcting capability and hence have been used in many applications. This paper demonstrates a flexible Low Density Parity Check (LDPC) decoder which is an improvement over other existing work on a general LDPC decoder. In this paper we have presented a fully flexible LDPC decoder design in FPGA which supports different codes and data rates. This decoder finds application in many communication standards such as Wireless LAN (IEEE 802.lln), WIMAX (IEEE 801.16e) and DVB-S2, on a single hardware platform which makes the transition from theory to practice a much easier one. Finally, this paper proposes a method of designing of fully flexible LDPC Decoder in Spartan6 - xc6slx16-3 FPGA hardware.*

Keywords *- Error-correcting code, FPGA hardware, LDPC, ROM Configuration* Shannon *capacity*

I. INTRODUCTION

Error correction coding is a means which deals with detection and correction of errors which are introduced into a communication system by the receiver. Shannon published a paper in 1948 that revolutionized communications. He found that there is a fundamental limit to how much information can be reliably sent over a channel (its capacity) with respect to the noise present in the system [1]. Although he defined a limit, it only showed how good the best possible code would perform, hence the proof was non-constructive. Low Density Parity Check (LDPC) Codes are a class of block code that satisfies both randomness and increased length. These codes are found to have the best performance till date, by achieving a capacity within 0.0045 dB of the Shannon limit [2].A (n, k) block code takes k bits (message bits) at a time and produces n bits (code bits). Normally, Block codes introduce redundancy for error correction. Let u be the collection of k message bits which is denoted as message word, and c be the collection of n encoded bits which is denoted as a codeword.

$$u = [u_0 \quad u_1 \ \dots\dots\dots\dots\dots\dots u_{k-1}]$$
$$c = [c_0 \quad c_1 \ \dots\dots\dots\dots\dots\dots c_{n-1}]$$

An important parameter of a block code is its code rate (r). It is a measure of the amount of information (message) bits sent per codeword. The more the information sent, the lesser the redundancy and fewer the error correction by the code. Conversely if the amount of redundancy introduced is high the transmitter will spend less time sending information.

The code words are encoded through a *generator matrix G* as given by the following identity:

$$c = u * G$$

Where G is a (n, k) binary matrix.

For every generator matrix *G*, there exists many *parity check matrices (H)* that satisfy the equation:

$$G * H^T = 0$$

The parity check matrix has the following important property i.e. c is a codeword if and only if it satisfies:

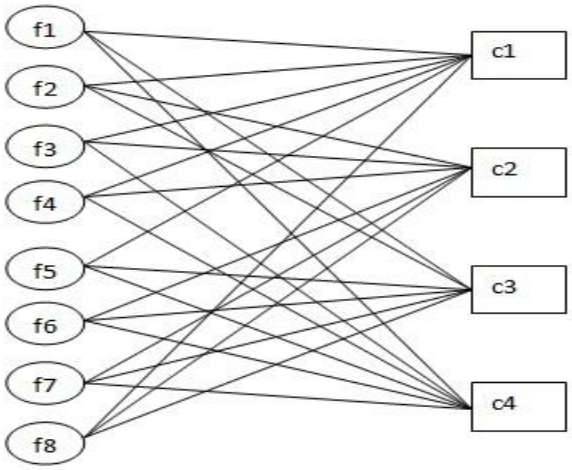
$$C * H^T = 0$$

i.e. all the code words lie in the null space of H.

Low Density Parity Check (LDPC) Codes are a form of forward error correction codes and was first discovered by Gallager in 1962 [3]. These are nothing but block codes whose parity-check matrix (H) contains very few non zero entities.

$$H=\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

(a)



(b)

Fig 1. LDPC code example. (a) Parity check matrix (b) Tanner graph

The Tanner graph shown above is a graphical form of representing the parity check matrix. A Tanner graph consists of two sets of vertices i.e. n vertices for the codeword bits (bit nodes), and m vertices for the parity-check equations (check nodes). A bit node gets joined to a check node by an edge, if that bit is included in the corresponding parity-check equation. The number of edges in the tanner graph is equal to the number of ones in the parity check matrix. Fig 1(b) shows a Tanner graph representation of the parity check matrix H as shown in Fig 1(a).

1.1 DECODING-MESSAGE PASSING:

The most typical decoding procedure implemented in several designs is the message passing algorithm, also known as "sum-product" or "belief propagation" (BP) Algorithm [4]. This algorithm implements the iterative exchange of node-generated messages along the edges of the graph. Most of the implementation challenges come from the communication structure to support message exchange among Variable Nodes (VNs) and Check Nodes (CNs) even if the nodes require additional computational capabilities. In addition, modern applications often need adaptation capabilities in a number of system parameters as block size and code rate impose tight flexibility requirements on hardware implementation of some designs.

II. LITERATURE REVIEW ON FLEXIBLE LDPC DECODER

Several researches have been conducted on the LDPC decoder till date, but there is very less reported work on flexible LDPC decoders. Some of the literature surveys done are summarized as follows:

2.1 SERIAL DECODER

It consists of a single check node, a single variable node and memory. The variable nodes can be updated once at a time and then the check nodes are updated in the serial manner. Serial implementation has a added advantage that it is very flexible thereby supporting different block sizes and code rates with only a new parity matrix being loaded into the memory. Unfortunately, this approach is too slow for any operation except simulations [5]. The number of clock cycles required for each iteration of the serial decoder is approximately twice the number of edges present.

2.2 ANALOG IMPLEMENTATION

While majority of the papers presented the use of digital signals to implement the decoder architecture, a number of papers have also used analog methods for implementation of their proposed designs. While it is not feasible to implement an analog decoder on an FPGA (FPGA devices can only process digital signals), it is nonetheless interesting to see analog methods being applied to solve a purely digital problem especially when their performance is comparable to their digital counterparts. There were two approaches of implementing the sum product algorithm. One of the two approaches was to use the non-linearities of charging a Field Effect Transistor (FET) [6, 7], while the other approach was to use a combination of resistors and capacitors, thereby resulting in RC differential equations [8]. The main disadvantage of these two approaches is that they provide more delay during decoding and are not feasible to implement on FPGA.

2.3 PARALLEL DECODER

A fully-parallel decoder completes the update check message step in one clock cycle and then in the next cycle completes the update bit message step. This allows a marked increase in decoding speed as compared to the serial decoder. The main disadvantage of using the decoder is that none of the reviewed parallel decoder implementations [9, 10, 11, and 12] have any flexibility in view of code design. The routing between check and bit nodes need redesigning for another code realization, while a different code rate or degree distribution may not be possible without a complete redesign.

2.4 PARTIALLY PARALLEL DECODER

In a partially parallel decoder design, some check and bit nodes are realized in hardware which involves sharing as in the serial decoder case. There is a trade-off between speed and complexity which results in increase in the number of check and bit nodes when realized in hardware which in turn increases the speed of each decoder iteration. This in turn increases the complexity of the design. Serial and parallel architectures could be considered subclasses of the partially parallel decoder (serial implementation has only one bit and check node realized in hardware while parallel implementation would have all bit and check nodes realized in hardware). The major disadvantage in this case is memory collision. If two nodes try to access the same memory during the same clock cycle a collision occurs and one of the nodes has to stall for a cycle which in turn decreases the throughput of the decoder.

III. VARIOUS PROPOSED METHODS FOR FLEXIBLE LDPC DECODER IMPLEMENTATION

A number of authors have recommended the use of partially parallel method for designing a fully flexible LDPC Decoder.Masera et al., (2007) proposed flexible LDPC decoder with the help of Low-Traffic BP (LTBP) algorithm [5]. The LTBP algorithm is of real help in the implementation of LDPC decoders only if a communication network supports parallel Multicast. A simple structure to implement such a network consists of a crossbar switch known as 'Bens Network' and a multiplexer bank. The memory collision problem which occurs due to the usage of partially parallel decoder is somehow reduced by this architecture. Unfortunately, the design is highly complex and not so mature as far as speed is concerned. In another research, Lee et al., (2008) have also used the Benes network in their partially parallel LDPC decoder design [13]. In the proposed architecture, Benes network is used to implement the interconnection network that can be configured according to a parity-check matrix. Compared to the previous flexible architecture, the proposed decoder shows a better Throughput-to-Area Ratio (TAR). Unfortunately the design being highly complex and deeply pipelined is suitable for no application other than Application Specific Integrated Circuit (ASIC) implementation. Chuan Zhang et al. (2010) proposed a flexible LDPC decoder design for mbps applications [14]. The Benes network is employed to implement the configurable interconnections between the Variable Node Processing Unit (VNPU) and Check Node Processing Unit (CNPU) arrays which in turn brings sufficient flexibility for multiple code rates and code lengths. Compared to the previous flexible architecture, the proposed decoder shows better TAR. By adopting better optimized pipelining scheme higher throughput can be achieved in this architecture.

IV. FULLY FLEXIBLE DECODER ARCHITECTURE

The design is based on scheduling as described in [15] by Masera et al. The proposed design implementation is based on both partially parallel implementation and unicast messages while the design proposed by Masera et al is based on multicast messages. In this design, the interleaver is more complex while the bit and check node processors as well as the scheduling are simpler.

The main design goal is universality i.e. the decoder should be capable of implementing any given code (up to maximum design constraints) without changing the design of the decoder. The architecture consists of a number of Processors (P), a Message Permutation Block (MPB) and a Control Logic Block (CLB), as shown in Fig 2.

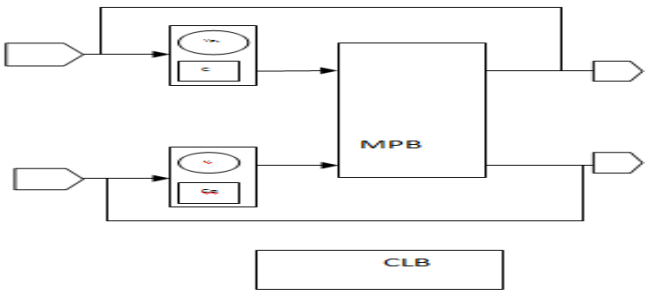


Fig 2. Top block diagram for decoder

The decoding process follows four distinct steps as shown in Fig. 3.

4.1. INITIALIZATION

During initialization, channel measurements are loaded into the bit processor blocks.

4.2. BIT TO CHECK

During the bit to check half iteration, the bit node processor performs the bit node function as described in the equation below [16].

$$Q_{ij} = \lambda_j + \sum_{\alpha \in c[j]\, \alpha \neq i} R_{\alpha j}\ [k-1]$$

In the first half iteration there are no incoming messages and the processor outputs the channel measurement. The messages flow from the bit node processors through the message permuting block and then to the corresponding check node processor in order.

4.3. CHECK TO BIT

During the bit to check half iteration, the check node processor performs the check node function as described in the equation below [16].

$$R_{ij} = 2\tanh^{-1} \prod_{\alpha \in V[j] \alpha \neq i} \tanh\ \left(\frac{Q_{\alpha j}}{2}\right)$$

The messages flow from the check node processors through the message permuting block and then to the corresponding bit node processor in order.

4.4. OUTPUT

After a number of iterations of bit to check and check to bit cycles the decoder moves to the output stage.

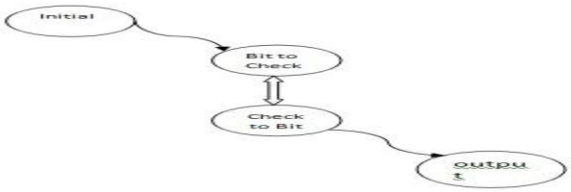


Fig 3. State machine for decoder

The MPB decides the flexibility of the decoder. Its purpose is to connect the bit node processor to the check node processor (and vice versa), so that the check node processor receives the incoming messages in a order as depicted in the Tanner graph. The control unit is responsible for implementing the state machine. The configuration for the code is stored in a ROM block within the control unit which contains information about the time varying settings for the Benes network and the interleaver banks in the message permutation unit. It also assigns the appropriate bit and check nodes to the respective processors.

V. DESIGN APPROACH

System generator is very flexible in integrating different design approaches namely Simulink models and VHDL codes. Simulink provide a much higher level of abstraction than VHDL. Therefore, our design approach utilizes Simulink library blocks to design various modules of the proposed decoder architecture and the VHDL code for the same is generated automatically using system generator. The auto-generated VHDL code can also be used in Xilinx ISE (Integrated Software Environment) for synthesis and implementation on desired FPGA (Field Programmable Gate Array). Automatic VHDL code generation reduces design efforts hence increases design productivity.

VI. DECODER DESIGN

Our decoder design architecture mainly consists of two Processors i.e. Check Node Unit Processor and Variable Node Unit Processor. A generic Simulink model was first developed for the variable node and check node.

6.1. VARIABLE NODE DESIGN USING SIMULINK

The variable node, shown in Fig 4, has been designed in Simulink using the basic addsub and mux Xilinx blocks from the Simulink library. As shown in Fig 4, the total sum is computed from three messages coming in from different check nodes as well as the channel input data. Three different outgoing messages are formed where each outgoing message has each corresponding message from the check node subtracted from it.

These outgoing messages are sent to the appropriate node connection as designated by the parity check matrix. At the start of the first iteration, the variable nodes will simply bypass the sum operation and send the input noisy channel data directly to the check nodes as there are no return messages from the check node in the beginning.

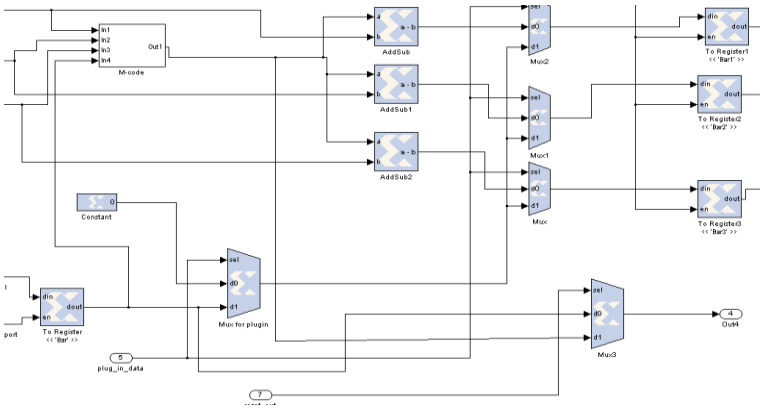


Fig 4. Variable node design in Simulink

6.2. CHECK NODE DESIGN USING SIMULINK

The Simulink model of the check node is as shown in Fig 5

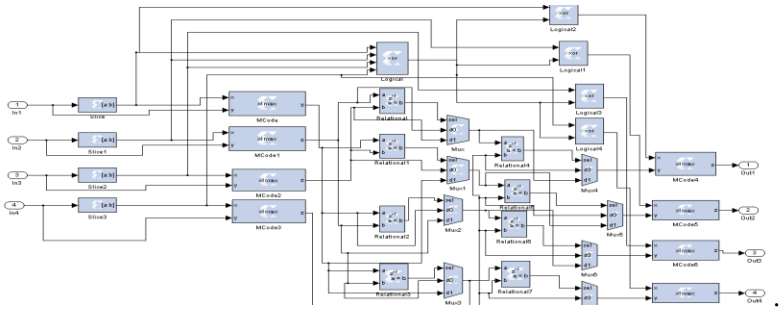


Fig 5. Check node design in Simulink

It uses blocks from Simulink library (absolute and bitwise XOR Xilinx blocks). The check node finds the minimum of all the inputs and performs the parity checks. Finally, the fully flexible LDPC decoder design using Variable node and Check node is shown in Fig 6 below.

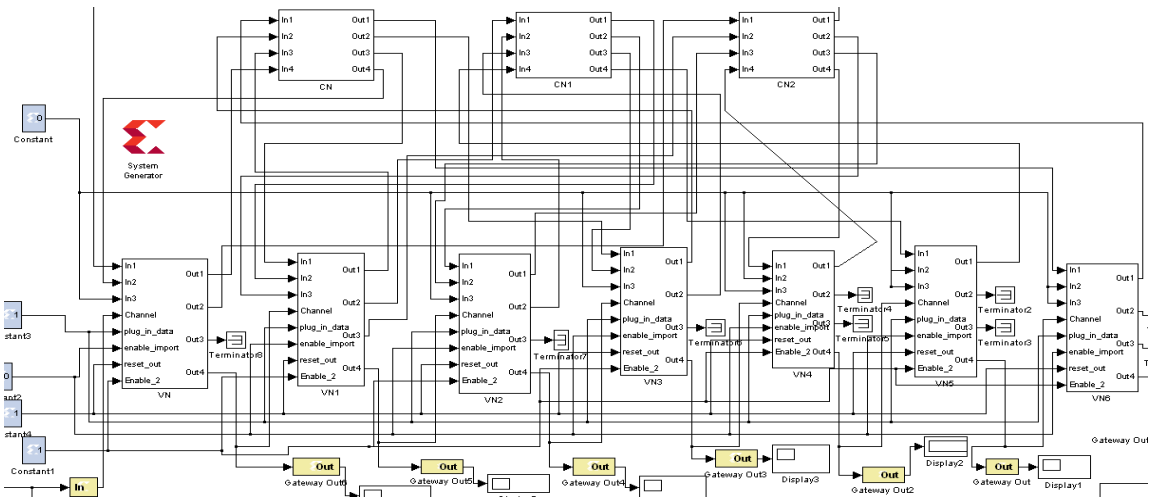


Fig 6. LDPC Decoder design in Simulink

Once the Simulink models are developed for the LDPC Decoder, the VHDL code can be generated automatically using the system generator.

VII. FPGA IMPLEMENTATION AND SIMULATION RESULTS

We implement this LDPC decoder in Xilinx Hardware Description Language and synthesize it using ISE (Ver. 13.4). The code is downloaded onto the xc6slx16 device of Spartan 6 family using csg324 package. The board was NEXYS 3 from DIGILENT spartan6. The table below summarizes the design statistic results of FPGA for the desired LDPC decoder.

Logic Components(Resources)	Used
Slices	765
Flip flops	646
Block RAMs	19
Maximum Frequency of Operation	89.29MHz

VIII. CONCLUSION

This paper has undergone an extensive review of the existing research works on flexible LDPC decoder and flexible LDPC decoders. We have designed a fully flexible LDPC decoder by using partially parallel implementation and the decoder is found to run at a clock speed of 11.2ns (Maximum frequency of operation being 89.29MHz). Based on these results, we conclude that the proposed architecture provides better solution for high speed decoding and also takes lesser area. Hence TAR increases which is better than the previous architectures.

IX. ACKNOWLEDGMENT

The authors acknowledge TIFAC-CORE for 3G/4G Communication Technologies at National Institute of Science & Technology, Brahmapur, Odisha.

REFERENCES

Journal Papers:

[1] C. Shannon, "A mathematical theory of communication," Bell System Technical Journal, vol. 27, no. 1, pp. 379-423, 1948.
[2] S.-Y. Chung, J. Forney, G.D., T. Richardson, and R. Urbanke, "On the design of low density parity check codes within 0.0045 db of the shannon limit," Communications Letters, IEEE, vol. 5, no. 2, pp. 58-60, Feb 2001.
[3] R. Gallager, "Low-density parity-check codes," Information Theory, IEEE Transactions on, vol. 8, no. 1, pp. 21-28, Jan 1962.
[4] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," IEEE Trans. Inf. Theory, vol. 45, no. 2, pp. 399–431, Mar. 1999.
[5] G. Masera, F. Quaglio, and F. Vacca, "Implementation of a flexible LDPC decoder,"Circuits and Systems II: Express Briefs, IEEE Transactions on, vol. 54, no. 6, pp. 542-546, June 2007.
[6] V. G. Chris Winstead, Nhan Nguyen and C. Schlegel, "Low-voltage cmos circuits for analog iterative decoders," Circuits and Systems, IEEE Transactions on, vol. 53, Apr. 2006.
[7] C. Kong and S. Chakrabartty, "Analog iterative ldpc decoder based on margin propagation," Circuits and Systems, IEEE Transactions on, vol..54, pp.1140-1144, Dec.2007.
[8] S. Hemati and A. Banihashemi, "Dynamics and performance analysis of analog iterative decoding for low-density parity-check (ldpc) codes," Communications, IEEE Transactions on, vol. 54, pp. 61-70, Jan. 2006.
[9] A. Blanksby and C. Howland, "A 690-mw 1-gb/s 1024-b, rate-1/2 low-density parity-check code decoder," Solid-State Circuits, IEEE Journal of, vol.37, no.3, pp.404-412, Mar2002.
[10] G. C. Luca Fanucci, Pasquale Ciao, "Design of a fully-parallel high-throughput decoder for turbo gallager codes," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, no.7, pp.976-1986.
[11] L. Zhou, C. Wakayama, and C.-J. Shi, "Cascade: A standard supercell design methodology with congestion-driven placement for three-dimensional interconnect-heavy very large-scale integrated circuits," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 26, no. 7, pp. 1270-1282, July 2007.
[12] V. Nagarajan, S. Laendner, N. Jayakumar, O. Milenkovic, and S. P. Khatri, "High- throughput vlsi implementations of iterative decoders and related code construction problems," J. VLSI Signal Process. Syst., vol. 49, no. 1, pp. 185-206, 2007.
[13] J.-Y. Lee and H.-J. Ryu, "A 1-Gb/s flexible ldpc decoder supporting multiple code rates and block lengths," Consumer Electronics, IEEE Transactions on, vol. 54, pp. 417-424, May 2008.
[14] Chuan Zhang, Zhongfeng Wang, Jin Sha, Li Li, and Jun Lin ," Flexible ldpc decoder design for multigigabit-per-second applications," Circuits and Systems, IEEE Transactions on, vol. 57, pp. 116-124, Jan. 2010.
[15] A. Tarable, S. Benedetto, and G. Montorsi, "Mapping interleaving laws to parallel turbo and LDPC decoder architectures," Information Theory, IEEE Transactions on, vol. 50, no. 9, pp. 2002-2009, Sept. 2004.
[16] S. Johnson, "Introduction to ldpc codes," in ACoRN Spring School on Coding, Multiple User Communications and Random Matrix Theory, 2006.