FPGA IMPLEMENTATION OF LINEAR LDPC ENCODER

Chetna N. Kharkar[1], M. M. Jadhav[2], A. M. Sapkal[3]

[1]*Sr.Design Engineer, Qualitat Systems,Pune,Maharashtra,India,**chetnakharkar@gmail.com***
[2]*Asso. Professor, E&TC, Sinhagad college of Engineering, Pune, Maharashtra,India,**makj123@yahoo.com***
[3]*HOD, E&TC, Government College of Engineering, Pune, Maharashtra, India, **hod.extc@coep.ac.in***

Abstract

In this paper, a FPGA implementation of linear time LDPC encoder is presented. This encoder implementation can handle large size of input message. Linear Time encoder hardware architecture reduces the Complexity and area of encoder than generator matrix based encoder techniques. This encoder is simulated on different platform which includes Matlab & High level languages for 1/2 rate & up to 4096 code length. FPGA implementation of the encoder is done on Xilinx Spartan 3E Starter Kit. The result shows the speed & area comparison for different FPGA platform.

Keywords*— LDPC codes, dual-diagonal, Linear encoding, Generator matrix complexity, FPGA Implementation*

--***--

1. INTRODUCTION

As their name suggests, LDPC codes are block codes with parity-check matrices that contain only a very small number of non-zero entries proposed by Gallager in 1962 [1] and has gained popularity due to their capacity-approaching error correcting performance [2].

In LDPC codes sparseness of H guarantees both a decoding complexity and minimum distance which increases only linearly with the code length, however, finding a sparse parity-check matrix for an existing code is not practical. Instead LDPC codes are designed by constructing a sparse parity-check matrix first and then determining a generator matrix for the code afterwards.

In order to reduce encoding complexity, LDPC codes with dual diagonal structure is adopted by the latest next-generation wireless LAN standard, IEEE 802.11n [3]. The LDPC encoding algorithm used is near-linear time proposed by [4] & [5].An LDPC code parity-check matrix is called *(*wc,wr*)-regular* if each code bit is contained in a fixed number, wc, of parity checks and each parity-check equation contains a fixed number, wr, of code bits. An efficient encoding algorithm [6] is used to reduce the encoding complexity.

In this paper we have implemented the low complexity Encoder algorithm on hardware platform on Xilinx Spartan 3E FPGA & simulated using Matlab 2012, Modelsim & c code. The Synthesis results shows the area & speed comparison on different FPGA platform. The encoded codeword is decoded using belief propagation algorithm [7] & results are verified using Matlab program.

2. LDPC CONSTRUCTION

The construction of binary LDPC codes involves assigning a small number of the values in an all-zero matrix to be 1 so that the rows and columns have the required degree distribution.

The original LDPC codes presented by Gallager are regular and defined by a banded structure in H. The rows of Gallager's parity-check matrices are divided into wc sets with M/wc rows in each set. The first set of rows contains wr consecutive ones ordered from left to right across the columns. (i.e. for $i \leq M/wc$, the i-th row has non zero entries in the $((i - 1)K + 1)$-th to i-th columns). Every other set of rows is a randomly chosen column permutation of this first set. Consequently every column of H has a '1' entry once in every one of the wc sets. Since LDPC codes are often constructed pseudo-randomly we often talk about the set (or ensemble) of all possible codes with certain parameters (for example a certain degree distribution) rather than about a particular choice of parity-check matrix with those parameters. LDPC codes are often represented in graphical form by a Tanner graph.

The Tanner graph as shows in figure-1, consists of two sets of vertices: n vertices for the code word bits (called bit nodes), and m vertices for the parity-check equations (called check nodes). An edge joins a bit node to a check node if that bit is included in the corresponding parity-check equation and so the number of edges in the Tanner graph is equal to the number of ones in the parity-check matrix.

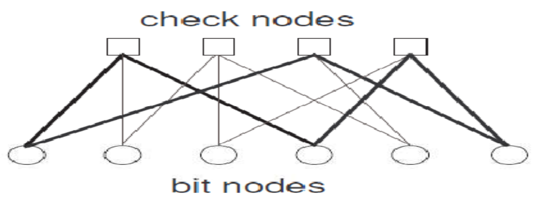


Fig 1: The Tanner graph representation of the parity-check a 6-cycle is shown in bold.

A *cycle* in a Tanner graph is a sequence of connected vertices which start and end at the same vertex in the graph, and which contain other vertices no more than once. The length of a cycle is the number of edges it contains, and the *girth* of a graph is the size of its smallest cycle. The Mackay Neal construction method for LDPC codes can be adapted to avoid cycles of length 4, called 4-cycles, by checking each pair of columns in H to see if they overlap in two places. The construction of 4-cycle free codes using this method is given in Algorithm 1. Input is the code length n, rate r, and column and row degree distributions v and h. The vector α is a length n vector which contains an entry i for each column in H of weight i and the vector β is a length m vector which contains an entry i for each row in H of weight i.

Algorithm 1: H Matrix Generation

```
Procedure
MNCONSTRUCTION (n, r, v, h) ⊲Required length, rate and degree distributions
H = all zero n(1 − r) × n matrix    ⊲ Initialization
α = [];
for i = 1 : max(v) do
  for j = 1 : vi × n do
    α = [α, i]
  end for
end for
β = []
for i = 1 : max(h) do
  for j = 1 : hi × m do
    β = [β, i]
  end for
end for
for i = 1 : n do                    ⊲ Construction
  c = random subset of β, of size αi
  for j = 1 : αi do
    H(cj , i) = 1
  end for
  α = α − c
end for
repeat
  for i = 1 : n − 1 do              ⊲ Remove 4-cycles
    for j = i + 1 : n do
      if |H(:, i)
S
H(:, j)| > 1 then
        permute the entries in the j-th column
      end if
    end for
  end for
until cycles removed
end procedure
```

3. ENCODING USING GENERATOR MATRIX

For a linear block code, the sum of any two code words results in another code word. LDPC code construction is also done in similar way of linear block code. From a given parity check matrix, H, a generator matrix, G is derived. Data, m = m_1, m_2…..m_n is encoded by multiplying it with the generator matrix, c = mG where m is a string of information bits. It has to be noted that putting H in systematic form, H= $[P^T | I_M]$, no longer has fixed column or row weights and P is very likely to be dense. The denseness of P determines the encoder computational complexity. A dense generator matrix requires a large number of operations when doing the matrix multiplication with the data to be sent. The encoding complexity could be reduced for some codes by parity check matrix pre-processing. An efficient encoding technique has been developed to reduce the encoding complexity by rearranging the parity check matrix before encoding. The encoding complexity also depends on the structure of the code

The construction of LDPC codes is categorized mainly into two: Random constructions and structured constructions. The type of construction is determined by the connections between check nodes and variable nodes in Tanner graph. Each type of constructions has their advantages over the other. Random constructions refer to the unstructured row-column connections in the parity check matrix with no predefined pattern. Random codes have better performance compared to structured codes in case of long codes. They are used in cases we want to increase the girth or rate of a given size. But longer length random LDPC codes require large memory storage in practical implementation which affects the computational efficiency of the code. The uncertainty of guaranteeing an asymptotically optimum performance in random constructions leads to the use of structured construction of LDPC codes. Structured construction method put constraints on row – column connections to get a desired or predefined connection pattern that is easier to implement in hardware.

3.1 LDPC Encoding Example

$$H = \begin{pmatrix} 0\,1\,0\,1\,1\,0\,0\,1 \\ 1\,1\,1\,0\,0\,1\,0\,0 \\ 0\,0\,1\,0\,0\,1\,1\,1 \\ 1\,0\,0\,1\,1\,0\,1\,0 \end{pmatrix}$$

The $H_{M\times N}$ parity check matrix defines a rate $R=K/N$, (N,K) code where $K=N-M$.

Code word is said to be valid if it satisfies the syndrome calculation:

$$z=c.H^T=0$$

We can generate the code word in by multiplying message ***m*** with generator matrix ***G***

$$c=m.G$$

We can obtain the generator matrix ***G*** from parity check matrix ***H*** by:

1. Arranging the parity check matrix in systematic form using row and column operations

$$H_{sys}=\left[I_M\middle|P_{M\times K}\right]$$

$$H_{sys}=\begin{bmatrix}1&0&0&1&0&1\\0&1&0&1&1&1\\0&0&1&0&1&1\end{bmatrix}$$

2. Rearranging the systematic parity check matrix

$$G=\left[P^T_{K\times M}\middle|I_K\right],$$

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

3. We can verify our results as $G.H^T=0$

4. LINEAR-TIME ENCODING FOR LDPC CODES

Instead of finding a generator matrix for H, an LDPC code can be encoded using the parity-check matrix directly by transforming it into upper triangular form and using back substitution. The idea is to do as much of the transformation as possible using only row and column permutations so as to keep as much of H as possible sparse.

Firstly, using only row and column permutations, the parity-check matrix is put into *approximate lower triangular form*:

$$H_t=\begin{pmatrix}A & B & T\\ C & D & E\end{pmatrix}$$

Where the matrix T is a lower triangular matrix (that is T has ones on the diagonal from left to right and all entries above the diagonal zero) of size

$$(m-g)\times(m-g)$$

If H_t is full rank the matrix B is size

$$m-g\times g$$

And A is size

$$m-g\times k$$

The g rows of H left in C, D, and E are called the *gap* of the approximate representation and the smaller g the lower the encoding complexity for the LDPC code.

Step 1

Instead of putting H into reduced row-echelon form we put it into approximate lower triangular form using only row and column swaps. For this H we swap the 2-nd and 3-rd rows and 6-th and 10-th columns to obtain:

$$H_t=\begin{pmatrix}1&1&0&1&1&0&0&1&0&0\\0&0&0&1&0&1&0&1&1&0\\0&1&1&0&1&0&1&0&0&1\\1&1&0&0&0&0&1&0&1&1\\0&0&1&0&0&1&0&1&0&1\end{pmatrix}$$

With a gap of two

Once in upper triangular format, Gauss-Jordan elimination is applied to clear E which is equivalent to multiplying H_t by

$$\begin{pmatrix}I_{m-g} & 0\\ -ET^{-1} & I_g\end{pmatrix},$$

To give

$$\tilde{H}=\begin{pmatrix}I_{m-g} & 0\\ -ET^{-1} & I_g\end{pmatrix}\qquad H_t=\begin{pmatrix}A & B & T\\ \tilde{C} & \tilde{D} & 0\end{pmatrix}$$

where

$$\tilde{C}=-ET^{-1}A+C$$

And

$$\tilde{D}=-ET^{-1}B+D$$

From Step 1

Step 2

$$T^{-1}= \begin{pmatrix} 1&0&0 \\ 1&1&0 \\ 0&0&1 \end{pmatrix}$$

And

$$\begin{pmatrix} I_{m-g} & 0 \\ -ET^{-1} & I_g \end{pmatrix} = \begin{pmatrix} 1&0&0&0&0 \\ 0&1&0&0&0 \\ 0&0&1&0&0 \\ 1&1&1&1&0 \\ 1&0&1&0&1 \end{pmatrix},$$

To give

$$\tilde{H} = \left(\begin{array}{cccccc|cc|cc} 1&1&0&1&1&0&0&1&0&0 \\ 0&0&0&1&0&1&0&1&1&0 \\ 0&1&1&0&1&0&1&0&0&1 \\ \hline 0&1&1&0&0&1&0&0&0&0 \\ 1&0&0&1&0&1&1&0&0&0 \end{array}\right)$$

When applying Gauss-Jordan elimination to clear E only $\tilde{C}$ and $\tilde{D}$ are affected, the rest of the parity-check matrix remains sparse. Finally, to encode using H the code word

$$c = [c1c2, \ldots, cn]$$

Is divided into three parts,

$$c = [u, p1, p2],$$

Where

$$u = [u1, u2, \ldots, uk]$$

Is the k-bit message

$$p1 = [p11 , p12 , \ldots, p1g],$$

Holds the first g parity bits and

$$p_2 = [p2_1 , p2_2 , \ldots, p2_{m-g}]$$

Holds the remaining parity bits

The code word

$$\tilde{c} = [u, p_1, p_2]$$

Must satisfy the parity-check equation $c\,\tilde{H}^T = 0$ and so

$$Au + Bp1 + Tp2 = 0, \qquad \text{---- (1)}$$

And

$$\tilde{C}u + \tilde{D}p1 + 0p2 = 0. \qquad \text{---- (2)}$$

Since E has been cleared, the parity bits in p1 depend only on the message bits, and so can be calculated independently of the parity bits in p_2. If D is invertible, p1 can be found from $\tilde{E}$quation (2)

$$p1 = \tilde{D}{-1}\, \tilde{C}u. \qquad \text{---- (3)}$$

If $\tilde{D}$ is not invertible the columns of H can be permuted until it is. By keeping g as small as possible the added complexity burden of the matrix multiplication in Equation (3), which is (g2), is kept low. Once p_1 is known p_2 can be found from Equation (1)

$$p2 = -T-1(Au + Bp1), \qquad \text{---- (4)}$$

Where the sparseness of A, B and T can be employed to keep the complexity of this operation low and, as T is upper triangular, p_2 can be found using back substitution.

From Step 2 we partition the length 10 codeword c = [c1, c2, . . . , c10] as c = [u, p1, p2] where p1 = [c6, c7] and p2 = [c8, c9, c10]. The parity bits in p1 are calculated from the message using Equation 3

Step 3

$$p_1= \tilde{D}^{-1}\tilde{C} = \begin{pmatrix} 1&0 \\ 1&1 \end{pmatrix} \begin{pmatrix} 0&1&1&0&0 \\ 1&0&0&1&0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1&0 \end{pmatrix}$$

As T is upper-triangular the bits in p_2 can then be calculated using back substitution

$$p2_1 = u_1 \oplus u_2 \oplus u_4 \oplus u_5 = 1 \oplus 1 \oplus 0 \oplus 1 = 1$$
$$p2_2 = u_4 \oplus p1_1 \oplus p2_1 = 0 \oplus 1 \oplus 1 = 0$$
$$p2_3 = u_2 \oplus u_3 \oplus u_5 \oplus p1_2 = 1 \oplus 0 \oplus 1 \oplus 0 = 0$$

and the code word is $c = \begin{pmatrix} 1&1&0&0&1&1&0&1&0&0 \end{pmatrix}$

Again column permutations were used to obtain H_t from H and so either H_t, or H with the same column permutation applied, will be used at the decoder.

5. ENCODER DESIGN & IMPLEMENTATION

Hardware implementation of Encoder is done on Xilinx Spartan 3E FPGA starter kit. Figure 2 shows the flow diagram for encoder implementation. We have implemented the ½ rate encoder for different matrix size 4X8, 16X32, 32X64, 64X128 on FPGA.We have used the Xilinx Vivado high level synthesis tool for design of FPGA based encoder. This tool supports the High level synthesis feature, using this feature we have done synthesis of our high level program. The Encoder design is synthesized on different FPGA & results are compared in terms of area & speed.

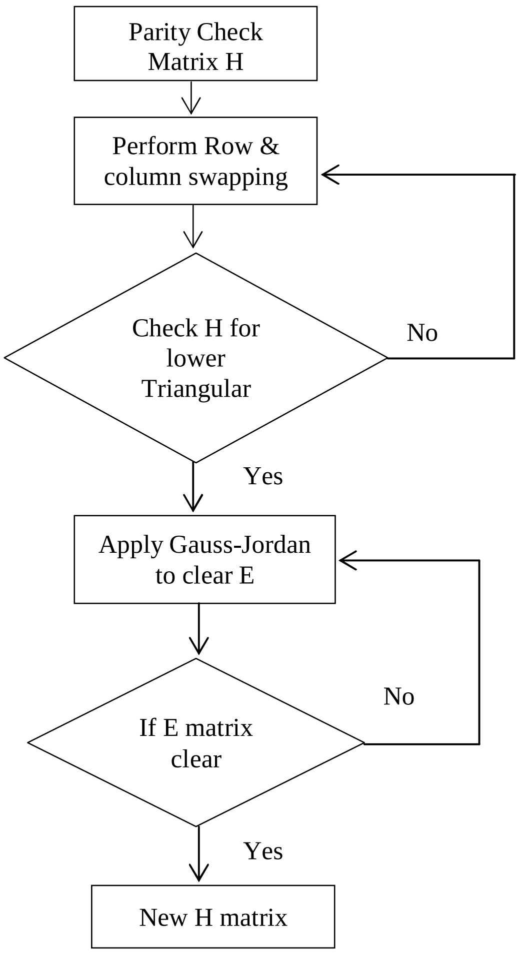
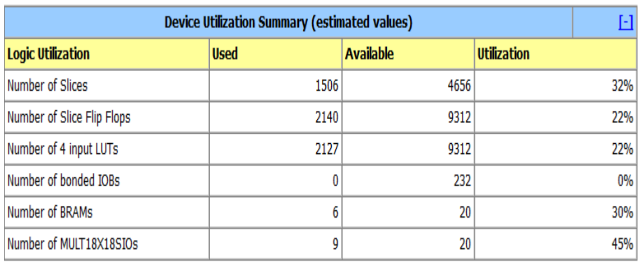

Parity Check Matrix H
Perform Row & column swapping
Check H for lower Triangular
No
Yes
Apply Gauss-Jordan to clear E
If E matrix clear
No
Yes
New H matrix

Fig-2 Encoder Implementation flow

6. IMPLEMENTATION RESULTS

We have simulated the ldpc encoder & log domain decoder algorithm in Matlab & results are verified both in simulation & implementation. Figure-4 shows Matlab simulation results. We have implemented linear-time Encoder for LDPC codes. This algorithm is implemented on Xilinx Spartan 3E board using ISE 13.1 & Xilinx High Level synthesis vivado HLS tool.

The synthesis results for Spartan 3E FPGA are shown in Figure 3.



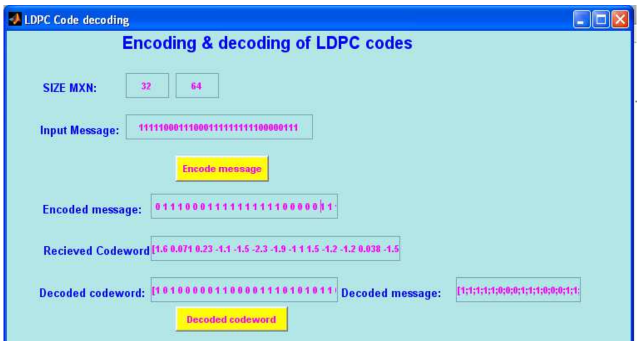
Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	1506	4656	32%
Number of Slice Flip Flops	2140	9312	22%
Number of 4 input LUTs	2127	9312	22%
Number of bonded IOBs	0	232	0%
Number of BRAMs	6	20	30%
Number of MULT18X18SIOs	9	20	45%

Fig -3 Device Utilization for Spartan 3E FPGA

Table-1 Comparison of Area & speed

Selected Device	Number of Slice Registers	Clock Frequency
3s500efg320-4	1506 (out of 4656)	71.782MHz
6slx4tqg144-3	2321 (out of 4800)	117.427MHz
7a30tcsg324-3	1918 (out of 42000)	187.337MHz

Encoder performance is verified on different FPGA platform table 1 shows the comparison of area & speed, from the table it is clear that Xilinx 7a30tcsg device Supports
Faster design speed.



LDPC Code decoding

Encoding & decoding of LDPC codes

SIZE MXN: 32 64

Input Message: 11111000111000111111111100000111

Encoded message: 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1

Recieved Codeword [1.6 0.071 0.23 -1.1 -1.5 -2.3 -1.9 -1 1 1.5 -1.2 -1.2 0.038 -1.5

Decoded codeword: [1 0 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 1 0 1 0 1 1

Decoded message: [1;1;1;1;1;0;0;0;1;1;1;0;0;0;1;1;

Fig-4 Result of Encoding & Decoding in Matlab simulation

CONCLUSIONS

We have implemented the linear time encoder in simulation & synthesis is done using Xilinx Tool. Xilinx Spartan 3E starter board is used for hardware implementation. The algorithm accepts the inputs as a input Message, H-matrix size & generates the Encoded codeword as a output.

This algorithm we have simulated on various platform including Matlab, C code, ISE13.2 & Modelsim .Building the Encoded codeword using Generator matrix is complicated for large size of parity check matrix .This Linear Time encoder algorithm provides an alternative for generator matrix creation & suitable for large parity check matrix .We have simulated & implemented LDPC encoder algorithm for smaller as well as larger codeword.

REFERENCES

[1]. R. G. Gallager, “Low density parity check codes,”*IRE Trans. Inform. Theory*, vol. IT-8, pp. 21–28, Jan. 1962.
[2]. D. J. C. MacKay and R. M. Neal, “Near Shannon limit performance of low density parity check codes,” *Electron. Lett.*, vol. 32, p. 1645, 1996.
[3]. T.J. Richardson and R.L. Urbanke, “Efficient encoding of low density parity-check codes,” *IEEE Trans. Inform. Theory*, vol.47, no.2 pp. 638- 656, Feb. 2001
[4]. D. J. C. MacKay, “Good error-correcting codes based on very sparse matrices,”*IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 399–431, March
1999.
[5]. C. Yoon, J.-E. Oh, M. Cheong Cheong, and S.-K. Lee, “A hardware efficient LDPC encoding scheme for exploiting decoder structure and resources,” *IEEE Vehicular Technology Conference (VTC2007-Spring)*, pp. 2445-2449, April 2007.
[6]. Susmitha Remmanapudi,and Balaji Bandaru, “An FPGA Implementation of low density Parity-check codes construction & Decoding” Devices ,Circuits & Systems International conference, pp .216-220,April 2012.

BIOGRAPHIES

Chetna N. Kharkar, She has completed her Graduation in Electronics & telecommunication Engineering. Currently working as a Sr. Design Engineer at Qualitat systems, Pune (India).She has a 6+ years of experience in Embedded domain. Perusing Master of Engineering from Sinhagad college of Engineering, Pune

M. M. Jadhav, he has Completed his graduation & Post graduation from Government college of Engineering, Pune. Presently perusing PHD from Government college of Engineering, Pune in the field of communication He has guided graduates & more than ten Post graduates students, His area of Interest is communication system

Dr. A. M. Sapkal, HOD of E&TC Department at government college of Engineering, Pune.He has 23 years of experience in teaching, 3 years Industrial & 14 years of Research experience. He has guided More than 100 post graduates students. He is a Chairman of Central Library of College of Engineering Pune.He is a member of THE IET U.K., IEEE, MIETE India.