Encoding of QC-LDPC Codes of Rank Deficient Parity Matrix

Mohammed Kasim Mohammed Al-Haddad
Lecturer
College of Engineering - University of Baghdad
Email: mkmih12@gmail.com

ABSTRACT

The encoding of long low density parity check (LDPC) codes presents a challenge compared to its decoding. The Quasi Cyclic (QC) LDPC codes offer the advantage for reducing the complexity for both encoding and decoding due to its QC structure. Most QC-LDPC codes have rank deficient parity matrix and this introduces extra complexity over the codes with full rank parity matrix. In this paper an encoding scheme of QC-LDPC codes is presented that is suitable for codes with full rank parity matrix and rank deficient parity matrx. The extra effort required by the codes with rank deficient parity matrix over the codes of full rank parity matrix is investigated.

Key words: Low density Parity Check (LDPC) Codes, Quasi-Cyclic (QC), Circulant Matrix.

ترميز QC-LDPC ذات مصفوفة التطابق ناقصة الرتبة

محمد قاسم محمد الحداد
مدرس
كلية الهندسة – جامعة بغداد
قسم الهندسة الالكترونية

الخلاصة

ان عملية ترميز فحص التطابق ذو الكثافة الواطئة (LDPC) في حال الرموز الطويلة تمثل تحدي مقارنةً بعملية فك الترميز. بينما ترميز الـ LDPC الشبه دائري (QC) له محاسن لتقليل حجم التعقيد الذي تتطلبه كلتا عمايتي الترميز وفك الترميز بسبب بنية الـ QC. معضم حالات ترميز الـ QC-LDPC يكون لها مصفوفة التطابق ذات رتبة ناقصة وهذا يؤدي الى زيادة في التعقيد مقارناً بحالات الترميز التي يكون لها مصفوفة تطابق كاملة الرتبة. في هذا البحث يجري استعراض طريقة لترميز QC-LDPC ملائمة لكلا حالتي مصفوفة التطابق الكاملة الرتبة والناقصة الرتبة. وقد جرى حساب الجهد الاضافي الذي تتطلبه حالات مصفوفة التطابق ذات الرتبة الناقصة على الجهد الذي تتطلبه حالات مصفوفة التطابق ذات الرتبة الكاملة.

1. INTRODUCTION:

Low Density Parity Check (LDPC) codes are subclass of linear block codes, it was first discovered by **Gallager** in 1962, but due to the available technology at that time and the codes computational demand the LDPC codes remained outside the fields of practice and research until it was rediscovered by **McKay, 1999**. The advantage of the LDPC codes over the algebraic codes comes from the fact that their decoding complexity is linear with code length which makes it practical to design long codes that approach Shannon channel capacity limit. The main disadvantage of the LDPC codes is that their encoding complexity is very high. The only competitor of LDPC codes is the turbo codes which are also Shannon limit approaching codes with slightly lower performance than LDPC codes but with better encoding complexity. The research in construction and encoding of LDPC codes is still an open field.

An LDPC code is defined by the $m \times n$ parity check matrix $\boldsymbol{H}$ that should be a sparse matrix from which Tanner graph is constructed. Tanner graph consists of two groups of nodes, the first is the variable nodes (or bit nodes) representing each bit in the code and the check nodes (or constraint nodes) representing the parity-check equations with connections, called edges, between the variable nodes and the check nodes each connection (or edge) represents a 1 in the $\boldsymbol{H}$ matrix. So, if there is a 1 in the i^{th} row and the j^{th} column of $\boldsymbol{H}$ then the i^{th} check node would be connected to the j^{th} variable node as shown in **Fig. 1.** A cycle in a Tanner graph is defined as the number of edges counted starting from a certain node and ending in the same node, the minimum size cycle is called the girth which is an important parameter of the code because the larger the girth the better error correcting capability of the code. A code with no cycles is said to have infinite girth, and the minimum possible value of the girth is 4, **Moreira, and Farrell, 2006**. Codes with girth equal to 4 show degraded performance, so, the cycles of size 4 should be removed from the $\boldsymbol{H}$ matrix in the construction of an LDPC code. These size-4 cycles can be easily located in the $\boldsymbol{H}$ matrix if there are 4 1's forming a rectangle.

LDPC codes fall into two main categories, regular and irregular codes. If the rows and columns of $\boldsymbol{H}$ have a constant weight (number of bits), then the code is regular, if not, it is irregular code. Irregular codes have the advantage of larger girth and hence better bit error correction capability but their lack of structure makes the encoder and decoder implementation more complex. In regular code, the coding rate can be given in terms of the column weight w_c and row weight w_r

$$R \geq 1 - \frac{m}{n} = 1 - \frac{w_c}{w_r} \tag{1}$$

Here the equality holds if $\boldsymbol{H}$ is a full rank matrix. More will be said about rank of parity matrix later.

2. CONSTRUCTIONS OF LDPC CODES:

The construction of LDPC codes is basically the construction of its parity matrix $\boldsymbol{H}$. There is no specific way for constructing LDPC code; instead there are constraints or guidelines. The first constraint is that the $\boldsymbol{H}$ matrix should be sparse, this is important for the decoding complexity reduction. The second is that there should be no two rows (or two columns) that have two 1's in the same positions; this is called the row-column (RC) constraint which makes the LDPC code free of cycles of size 4. Basically there are two main categories of approaches used in construction LDPC codes; the first is the random-like LDPC codes such as **Gallager, 1962** codes and **Mckay, 1999**

codes, progressive-edge-growth (PEG) codes, approximate cycle extrinsic message degree (ACE) codes. The second category is the structured quasi cyclic LDPC codes like protograph codes and codes based on the finite fields and finite geometry mathematics. In general the LDPC code that possesses a structure is preferred over the nonstructured codes because of the reduced complexity in encoding and decoding. The QC-LDPC codes are elegantly structured codes and can be encoded efficiently using shift registers. **Song et al., 2009** prsented a unified approach for constructing QC-LDPC codes based on finite fields the construction is based on primitive elements, additive subgroups, and cyclic subgroups of finite fields.**Tao et al., 2011** presented a search algorithm to first construct a QC-LDPC codes of column-row weigt (k,k) and girth g and then a random LDPC codes was derived with coumn-row weight (2,k) and girth 2g a girth of up to 36 was achieved. **Park et al**., presented a search method based on graph theoretic approach to detect subgraphs that cause short cycles in order to be avoided in the construction of the LDPC code, the design achieved codes of girth greater than 14. **Huang and Lee, 2013** constructed a block circulant RS based LDPC codes which have the advantage over the random RS based LDPC codes interms of decoding complexity. **Wang *et al*., 2013** constructed a high girth hierarchical quasi-cyclic (HQC) LDPC codes in the form of a hierachy of block circulant submatrices instead of the common block circulant based QC-LDPC codes. Since we are interested in the encoding of QC-cyclic LDPC codes an examples of constructing such codes will be provided later.

3. DECODING OF LDPC CODES:

Unlike algebraic codes like Reed-Solomon (RS) codes, LDPC codes can be decoded with complexity linearly proportional to code length and this is a very important feature of LDPC codes making it possible to implement long codes that can approach Shannon capacity limit. The decoding of LDPC codes is based on message-passing iterative algorithms, where messages are passed along the edges of the Tanner graph of the code between the variable nodes and the check nodes. The process is iterated until all parity check equations are satisfied or until a predetermined number of iterations is reached. In case of hard decision decoding the algorithm is called bit flipping where each bit of the code is recalculated using the remaining bits using all the parity check equations and based on the majority logic of the outcomes of the parity check equations, that bit may be flipped or kept as it is. In case of soft decision the information passed along the edges of the Tanner graph represents the probability (or likelihood) that each parity check equation is satisfied according to the possible of the code bit 0 or 1. This algorithm is called belief propagation and has many variations like the sum product algorithm. Although the message passing algorithms are in fact suboptimal algorithms but their impressive performance in error correction gives them significant advantage over algebraic codes.

4. ENCODING OF LDPC CODES:

Although the decoding of long LDPC codes is manageable in term of complexity, the encoding in general presents a challenge. In coding theory, a linear code such as the LDPC codes, the length of the code is $n=k+m$ bits where k is the number of message bits and m is the parity bits, there is an $m\times n$ parity-check matrix $\boldsymbol{H}$ and a $k\times n$ generator matrix $\boldsymbol{G}$ such that $\boldsymbol{GH}^{\mathrm{T}}=\mathbf{0}$, where $\mathbf{0}$ is $k\times m$ zero matrix. If $\boldsymbol{G}$ is in systematic form then $\boldsymbol{G}=[\boldsymbol{I}_k\ \boldsymbol{P}^{\mathrm{T}}]$ where $\boldsymbol{I}_k$ is $k\times k$ identity matrix and $\boldsymbol{P}^{\mathrm{T}}$ is the transpose of the $k\times m$ $\boldsymbol{P}$ matrix. The $\boldsymbol{G}$ matrix can be obtained by transforming the $\boldsymbol{H}$ matrix to the form $\boldsymbol{H}=[\boldsymbol{P}\ \boldsymbol{I}_m]$, where $\boldsymbol{I}_m$ is $m\times m$ identity matrix and $\boldsymbol{GH}^{\mathrm{T}}=\mathbf{0}$ is satisfied. Now, if $\boldsymbol{H}$ is the parity

matrix of an LDPC code then it is a sparse matrix (low density) and by applying the elementary row operation to $\boldsymbol{H}$ in order to get $\boldsymbol{H}=[\boldsymbol{P}\ \boldsymbol{I}_m]$, the result is that $\boldsymbol{H}$ or particularly $\boldsymbol{P}$ is no longer a sparse matrix and that produce a dense $\boldsymbol{G}$ matrix **Moreira, and Farrell, 2006**. Therefore, if traditional brute force implementation of the encoding as $\boldsymbol{v}=\boldsymbol{uG}$, where $\boldsymbol{v}$ is the code vector and $\boldsymbol{u}$ is the message vector, and if we assume that on average that the $\boldsymbol{P}$ matrix has equal number of 1's and 0's, then the number of XOR gates needed for the encoding and the amount of storage required for storage of the $\boldsymbol{P}$ matrix will be proportional to $k\times m$ or $(n-m)\times(n-k)$ that is almost proportional to n^2 and for very long codes this is a very big number that is impractical to implement. The alternative approach is that instead of encoding using the generator matrix $\boldsymbol{G}$, the parity check matrix $\boldsymbol{H}$ is used for encoding, where the parity check equations are solved for the parity bits.

$$\boldsymbol{vH}^{\mathrm{T}}=\boldsymbol{0} \tag{2}$$

The first novel approach for solving the parity equations is by **Richardson and Urbanke** in 2001 where the $\boldsymbol{H}$ matrix is decomposed using column permutation such that

$$\boldsymbol{H}=\begin{bmatrix}\boldsymbol{A} & \boldsymbol{B} & \boldsymbol{T}\\ \boldsymbol{C} & \boldsymbol{D} & \boldsymbol{E}\end{bmatrix} \tag{3}$$

Where $\boldsymbol{T}$ is the largest lower triangular matrix that can be obtained using column permutations with dimensions $(m-g)\times(m-g)$ and $\boldsymbol{E}$ is a $g\times(m-g)$ matrix. The form of the $\boldsymbol{H}$ matrix is called Approximate Lower Triangle (ALT). The algorithm is based on making the value of g (called the gap) as small as possible and this can make the complexity proportional to $n+g^2$. Although Richardson's algorithm works just fine for small values of g, the problem is that not every $\boldsymbol{H}$ matrix of LDPC codes can be decomposed into the form for Eq. (3) with small value of g by simple column permutations especially the quasi cyclic codes. In the case of QC- LDPC codes, the encoding can be performed using shift registers. **Li et al., 2006** presented an elegant approach for encoding QC-LDPC codes by first constructing the generator matrix in systematic form with block circulant structure. The block circulant structure of the generator matrix makes it possible to use shift registers that are loaded with the first row of the generator matrix and by circularly shifting its contents the remaining rows of the circulant block is generated. This saves the amount of the required storage and the number of XOR gates. Li presented two methods for constructing the generator matrix, the first is for a full rank parity matrix where the generator matrix is a block circulant matrix, the complexity in terms of storage, logic gates and number of clocks is provided. The second method is for a rank deficient parity matrix where the generator matrix is not fully block circulant and the complexity is increased but no metric for complexity for this case was provided. **Xia et al., 2008** proposed a method of construction of the $\boldsymbol{H}$ matrix using permutation matrices in a way to further reduce the computations required by Richardson's method. **Freundlich et al. 2008** worked on constructing LDPC codes having ALT with $g=\sqrt{n}$ in order to make encoding complexity proportional to n. **Huang et al. 2014** used Galois Fourier Transform for encoding QC-LDPC codes. **Lu and Moura, 2010** presented the label and decide algorithm suitable for linearly encoding LDPC codes that have tree or pseudo-tree structures.

5. ENCODING OF QC-LDPC CODES

Given that the parity matrix $\boldsymbol{H}$ is a block circulant matrix, which is given in the form

$$\boldsymbol{H}=\begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \cdots & \boldsymbol{A}_{1b} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \cdots & \boldsymbol{A}_{2b} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{a1} & \boldsymbol{A}_{a2} & \boldsymbol{A}_{a3} & \boldsymbol{A}_{ab} \end{bmatrix}=[\boldsymbol{H}_1 \quad \boldsymbol{H}_2] \tag{4}$$

Where $\boldsymbol{H}$ is $a\times b$ array of sparse $q\times q$ circulant matrices $\boldsymbol{A}_{r,s}$ where $1\le r\le a$ and $1\le s\le b$ making $\boldsymbol{H}$ a sparse QC parity matrix of dimension $aq\times bq$. The $\boldsymbol{H}$ matrix is better partitioned as two parts $\boldsymbol{H}=[\boldsymbol{H}_1\ \boldsymbol{H}_2]$, the first part $\boldsymbol{H}_1$ is $(b\text{-}a)\times a$ array of circulants and the second part is $a\times a$ array of circulants and by applying the appropriate matrix operations, the $\boldsymbol{H}_2$ part is converted to the identity matrix and the $\boldsymbol{H}$ matrix is transformed as $\boldsymbol{H}=[\boldsymbol{H}_1\ \boldsymbol{H}_2]\rightarrow[\boldsymbol{P}\ \boldsymbol{I}_{\mathrm{m}}]$ as discussed earlier.

Since the circulant matrix is the building block of the QC-LDPC parity matrix, it is worthwhile at this point to discuss its main properties, **Lu and Moura, 2010**. First the matrix multiplication is commutative, if $\boldsymbol{A}$ and $\boldsymbol{B}$ are circulants then $\boldsymbol{AB}=\boldsymbol{BA}$. Second the circulant matrix is characterized by its first row which is called the generator row, the subsequent rows are simply the circular shift of this generator row. If the generator row consists of one nonzero element, then each row and column will have only one nonzero element and circulant will be a permutation matrix and it is called circulant permutation matrix CPM. If the generator row is represented as a polynomial of x where the powers of x are the position and the coefficients are the elements of the generator row, then there will be a one-to-one mapping between circulants and binary polynomials, and the matrix addition and multiplication is replaced by polynomial addition and multiplication modulo x^q+1. This is called isomorphism where mathematically it is represented by, **Joyner, et al. 2004**

$$\phi(A)+\phi(B)=\phi(A+B) \tag{5a}$$

$$\phi(A)\times\phi(B)=\phi(A\times B) \tag{5b}$$

Where $\phi(\cdot)$ is any one-to-one onto mapping. The polynomial ring is referred to as $GF_2[x]/[1+x^q]$, where GF_2 represent the binary field $\{0,1\}$. In a ring not every element has an inverse, therefore if α and β are elements of a ring and $\alpha\beta=0$, it is possible to have $\alpha\neq0$ and $\beta\neq0$, in this case α and β are called zero-divisors and zero-devisors do not have multiplicative inverse. In our case of binary polynomials $GF_2[x]/[1+x^q]$ there are some examples of zero-divisor polynomials like **Lu and Moura, 2010**

$$\gamma(x)(1+x+x^2+\ldots+x^{q-1})=0 \tag{6}$$

Where $\gamma(1)=0$ which means that $\gamma(x)$ has even number of terms, here both polynomials multiply to zero meaning that both polynomials are zero-divisors and hence have no inverses and the corresponding circulants will be rank deficient. Another example is

$$(1+x^{p}+x^{2p}+\cdots+x^{q-p}) \tag{7}$$

Where p is a divisor of q, here shifting by p positions will generate the same polynomial and the corresponding circulant will have repeated rows and hence will be rank deficient. The fact that rings having elements with no multiplicative inverse makes the equation

$$\alpha x=\beta \tag{8}$$

Does not always have a solution where α and β are elements of a ring. Interestingly Eq. (8) can have a solution even though the element α does not have a multiplicative inverse but for a certain condition. To see this, consider the ring of integers modulo- n (mathematically referred to as Z/Z_n) where n is any integer. Equation (8) can now be written as **Joyner, et al. 2004**

$$\alpha x=\beta \bmod n$$
$$\alpha x-\beta=qn$$
$$\alpha x-qn=\beta$$

The above equation can have a solution if gcd(α,n) divides β **Meyer, 2000**, where gcd(α,n) is the greatest common divisor of α and n. Note, as a special case, if n is a prime then the ring becomes a field and every element has a multiplicative inverse and Eq. (8) always have a solution, on the other hand gcd(α,n)=1 which always divides β. Although this result is shown to apply to ring of integers, it also applies to ring of polynomials and hence circulants and the condition of Eq. (8) to have a solution modifies to gcd($\alpha(x)$,x^r+1) divides $\beta(x)$ where $\alpha(x)$ and $\beta(x)$ are the generator polynomials of the circulants α and β. From the above it can be seen that Eq. (8) can have a solution even if the circulant α is not a full rank matrix, but still there is no guarantee that Eq. (8) always has a solution.
In general it is desirable to transform the $\boldsymbol{H}$ matrix into the form given by Eq. (9) using elementary row operations.

$$\boldsymbol{H}=\begin{bmatrix}\boldsymbol{P} & \boldsymbol{I}_{aq}\end{bmatrix}=\begin{bmatrix}\boldsymbol{P}_{1,1} & \boldsymbol{P}_{1,2} & \cdots & \boldsymbol{P}_{1,(b-a)} & \boldsymbol{I}_q & \boldsymbol{0} & \cdots & \boldsymbol{0}\\ \boldsymbol{P}_{2,1} & \boldsymbol{P}_{2,2} & \cdots & \boldsymbol{P}_{2,(b-a)} & \boldsymbol{0} & \boldsymbol{I}_q & \cdots & \boldsymbol{0}\\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{P}_{a,1} & \boldsymbol{P}_{a,2} & \cdots & \boldsymbol{P}_{a,(b-a)} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I}_q\end{bmatrix} \tag{9}$$

Where $\boldsymbol{P}_{r,s}$ are $q\times q$ block circulant matrices, $1\le r\le a$, $1\le s\le b-a$, $\boldsymbol{I}_q$ is an $q\times q$ identity matrix and $\boldsymbol{0}$ is the all zero matrix, $\boldsymbol{P}$ is an $a\times(b-a)$ array of circulant. It can be seen from the transformed form of $\boldsymbol{H}$ in Eq. (9) that the $\boldsymbol{H}$ matrix is a full rank matrix, i.e., rank=aq. The form of Eq. (9) can only be obtained if the original parity matrix $\boldsymbol{H}$ in Eq. (4) is a full rank matrix **Li *et al.*, 2006**, and this is true

if the partition matrix $\boldsymbol{H}_2$ is invertible and hence $\boldsymbol{P}=\boldsymbol{H}_1\boldsymbol{H}_2^{-1}$. In terms of polynomials if $\boldsymbol{h}_2(x)$ is the matrix of polynomial corresponding to the partition of circulants $\boldsymbol{H}_2$, then $\boldsymbol{H}_2$ is invertible if and only if $\det[\boldsymbol{h}_2(x)]$ is not a zero-divisor, **Lu and Moura, 2010**. An example of such matrix is shown in **Fig. 2** where the 1's are shown as black dots.

The QC-LDPC code can be encoded using Linear Feedback Shift Registers (LFSR) and simple logic circuits. Since for the case of a full rank QC parity check matrix $\boldsymbol{H}$, the $\boldsymbol{P}$ matrix is also QC and this makes it possible to store only the generator row of each $\boldsymbol{P}_{r,s}$ circulant in a shift register (SR) and generating the remaining rows by cyclically shifting the SR contents. Therefore the $\boldsymbol{H}$ matrix in Eq. (9) can be reduced to an $a\times(b\text{-}a)$ array of $1\times q$ vectors by taking the generator rows $\boldsymbol{p}_{r,s}$ of the circulants $\boldsymbol{P}_{r,s}$ and the result is an $r\times(b\text{-}a)q$ parity generator matrix $\boldsymbol{PG}$ as below

$$\boldsymbol{PG}=\begin{bmatrix} \boldsymbol{p}_{1,2} & \boldsymbol{p}_{1,2} & \cdots & \boldsymbol{p}_{1,(b-a)} \\ \boldsymbol{p}_{2,1} & \boldsymbol{p}_{2,2} & \cdots & \boldsymbol{p}_{2,(b-a)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{p}_{a,1} & \boldsymbol{p}_{a,2} & \cdots & \boldsymbol{p}_{a,(b-a)} \end{bmatrix} \tag{10}$$

The remaining rows of the circulants $\boldsymbol{P}_{r,s}$ can be expressed as the cyclic shift of $\boldsymbol{p}_{r,s}$ denoted as $\boldsymbol{p}_{r,s}^{(t)}$ where t is the shift, $0\le t\le q$, and $\boldsymbol{p}_{r,s}^{(0)}=\boldsymbol{p}_{r,s}^{(q)}=\boldsymbol{p}_{r,s}$.

By representing the codeword in systematic form as $\boldsymbol{c}=[\boldsymbol{u}\ \boldsymbol{v}]$ where $\boldsymbol{u}$ is the message bits vector of length $q(b\text{-}a)$ bits and $\boldsymbol{v}$ is the parity bits vector of length aq bits, the problem of encoding reduces to evaluating the $\boldsymbol{v}$ vector. Since each codeword $\boldsymbol{c}$ satisfies the parity equations given by the $\boldsymbol{H}$ matrix, then $\boldsymbol{c}\boldsymbol{H}^{\mathrm{T}}=[\boldsymbol{u}\ \boldsymbol{v}]\ [\boldsymbol{P}\ \boldsymbol{I}_m]^{\mathrm{T}}=\boldsymbol{0}$, so $\boldsymbol{v}=\boldsymbol{u}\boldsymbol{P}^{\mathrm{T}}$. Using the form of the $\boldsymbol{H}$ matrix in Eq. (9), the parity bits v_i, $1\le i\le m=aq$ can be expressed in terms of the message bits u_j, $1\le j\le k=n\text{-}m=(b\text{-}a)q$ as

$$v_i=\sum_{j=1}^{(b-a)q} u_j\,p_{i,j} \quad 1\le i\le aq \tag{11}$$

Where v_i is the i^{th} parity bit of the vector $\boldsymbol{v}$ and $p_{i,j}$ is the matrix element in the i^{th} row and j^{th} column of the matrix $\boldsymbol{P}$. The direct realization of Eq. (11) is referred to as brute force realization, it is inefficient in terms of cost and it is general to any structure of the parity matrix $\boldsymbol{H}$. An efficient realization of Eq. (11) would obviously require the utilization of the QC structure of the parity matrix $\boldsymbol{P}$. This suggests dividing both the message vector $\boldsymbol{u}$ and the parity vector $\boldsymbol{v}$ into blocks of size q resulting in $(b\text{-}a)$ q-size message blocks $\boldsymbol{u}_s$ $1\le s\le b\text{-}a$, $\boldsymbol{u}=[\boldsymbol{u}_1\ \boldsymbol{u}_2\ \ldots\ \boldsymbol{u}_{b\text{-}a}]$ and a q-size parity blocks $\boldsymbol{v}_r$ $1\le r\le a$, $\boldsymbol{v}=[\boldsymbol{v}_1\ \boldsymbol{v}_2\ \ldots\ \boldsymbol{v}_a]$. According to this, Eq. (11) can be rewritten as

$$v_i=\sum_{s=1}^{b-a}\boldsymbol{u}_s[\boldsymbol{p}_{r,s}^{(t)}]^{\mathrm{T}}=\sum_{s=1}^{b-a}w_{r,s,t} \tag{12}$$
$$i=(r-1)q+t+1$$

Where $1 \le r \le a$, $0 \le t \le q-1$, and the notation $[\ ]^T$ is used for transpose. From Eq. (12), the r^{th} block of the parity vector $\boldsymbol{v}_r$ is obtained by fixing r and incrementing the shift index t from 0 to q-1. Note that the product inside the summation is a row-vector product and can be expressed as

$$w_{r,s,t} = \boldsymbol{u}_s [\boldsymbol{p}_{r,s}^{(t)}]^T = \sum_{\ell=1}^{q} u_s^{\ell} p_{r,s}^{(t),\ell} \quad (13)$$

Where u_s^{ℓ} is the ℓ^{th} element of the block vector $\boldsymbol{u}_s$, $1 \le \ell \le q$, and $p_{r,s}^{(t),\ell}$ is the ℓ^{th} element of the generator row $\boldsymbol{p}_{r,s}$ after shifting t positions. The quantity $w_{r,s,t}$ represents the partial sums generated by Eq. (13) which will be summed by Eq. (12) to produce the respective parity bit. The circuit implementation of Eq. (13) is shown in **Fig. 3** and it is the building block of the encoder, it consists of a shift register and modulo-2 adder (XOR), so it is referred to as shift register-adder (SRA) and it is one of the encoder implantation adopted by Li **Li, et al., 2006**. The overall encoder implementation is shown in **Fig. 4**, the number of the SRA's in the encoder will equal to b-a blocks, each block will be parallel loaded a times with the generator rows of the r^{th} row of the $\boldsymbol{PG}$ matrix and shifted q-1 times. The contents are multiplied by the message vector $\boldsymbol{u}$ and the results are added modulo- 2 (XOR) to generate parity bit p_i with every clock cycle, which will be a total of $m=aq$ which is the number of the parity bits. In case of rank deficient $\boldsymbol{H}$ matrix there will be more SRA's as will be shown later.

6. PARITY MATRIX WITH RANK DEFICIENCY

Transforming a QC sparse rank-deficient parity matrix as in **Fig. 5** into the form of Eq. (9) results in a random $\boldsymbol{P}$ matrix and not a block circulant as in **Fig. 6**. In this case, there will be no way of efficient implementation of the encoding of the corresponding LDPC code. The main aim of this paper is to present and algorithm that transforms the $\boldsymbol{H}$ matrix into a form that is as close as possible to the form of Eq. (9) while maintaining the QC structure. This will make it possible to make the encoding using the circuit of **Fig. 4** with some modifications. Specifically speaking, a rank deficient $\boldsymbol{H}$ matrix cannot be diagonalized while keeping the QC structure and since the latter is more important, the goal will be to make $\boldsymbol{H}_2$ matrix a sparse lower triangular matrix, so the parity bits can be evaluated in back substitution manner. The core of the approach of this paper is to work the Gauss-Jordan elimination on circulant blocks rather than on individual bits. This is to guarantee that the $\boldsymbol{H}$ matrix remains in the QC format because the result of addition and multiplication of circulants is also a circulant this is obvious from the circulant-polynomial isomorphism. Since in this case the $\boldsymbol{H}$ matrix is rank deficient, it is expected that there is a number of redundant parity equation and a number of redundant (free) parity bits. The application of Gaussian elimination algorithm on bit level will make the matrix in reduced echelon form **Meyer, 2000** where some of the diagonal elements are zeros and some rows will be an all zero row. The column positions of the zero diagonal elements correspond to the free parity bits or simply free bits and basically can be used as message bits. The all zero rows represent the redundant parity equation.

The application of Gauss-Jordan algorithm on circulant block level to $\boldsymbol{H}$ matrix in order to diagonalize the $\boldsymbol{H}_2$ matrix, two objectives are needed to be accomplished

1- Set every block $A_{r,s}$ above and below the block diagonal of $\boldsymbol{H}_2$ (i.e. $s \ne r+b-a$) to all zero matrix. This is equivalent to solving the matrix equation

$$\boldsymbol{AX} = \boldsymbol{B} \quad (14)$$

for the $\boldsymbol{X}$ matrix where $\boldsymbol{A}$ is a diagonal block of $\boldsymbol{H}_2$ ($\boldsymbol{A}_{r,r+b-a}$) and $\boldsymbol{B}$ is nondiagonal block that is in the same column i.e $\boldsymbol{B}=\boldsymbol{A}_{k,r+b-a}$, $k\neq r$. So, by applying the elementary row operation

$$\boldsymbol{H}_{k,:} = \boldsymbol{H}_{k,:} + \boldsymbol{H}_{r,:}\boldsymbol{X} \tag{15}$$

The $\boldsymbol{A}_{k,r+b-a}$ block will be eliminated. Here the notation $\boldsymbol{H}_{k,:}$ means the k^{th} row of the matrix $\boldsymbol{H}$.
2- Set every block on the diagonal ($\boldsymbol{A}_{r,r+b-a}$) to the identity matrix $\boldsymbol{I}_q$. This is equivalent to solving the equation

$$\boldsymbol{A}\boldsymbol{X} = \boldsymbol{I}_q \tag{16}$$

For the $\boldsymbol{X}$ matrix where $\boldsymbol{A}$ is a diagonal block $\boldsymbol{A}_{r,r+b-a}$. By applying the elementary row operation

$$\boldsymbol{H}_{r,:} = \boldsymbol{H}_{r,:}\boldsymbol{X} \tag{17}$$

The $\boldsymbol{A}_{r,r+b-a}$ block is converted to the identity matrix. Because the elementary row operations involved in the steps above are applied on the block circulant level, the blocks $\boldsymbol{A}_{r,s}$ remain circulants and hence maintain the QC structure of the $\boldsymbol{H}$ matrix. If the solutions, $\boldsymbol{X}$, of Eqs. (14) and (16) exist for each step, the $\boldsymbol{H}$ matrix will be converted to the form of Eq. (9) and the circuit of **Fig. 4** can be used for the encoding. Solving Eqs. (14) and Eq. (16) is performed by augmenting the matrix $\boldsymbol{A}$ and the first column of matrix $\boldsymbol{B}$ and $\boldsymbol{I}_q$ respectively and performing Gauss-Jordan elimination. After diagonlizing matrix $\boldsymbol{A}$, the last column of the augmented matrix will be the first column of matrix $\boldsymbol{X}$ and the remaining columns of matrix $\boldsymbol{X}$ is generated by circular shifts of the first column. This is explained in Eq. (18) below

$$[\boldsymbol{A} \mid \boldsymbol{B}_{:,1}] \rightarrow [\boldsymbol{I}_q \mid \boldsymbol{X}_{:,1}] \tag{18a}$$

$$[\boldsymbol{A} \mid \boldsymbol{I}_{q_{:,1}}] \rightarrow [\boldsymbol{I}_q \mid \boldsymbol{X}_{:,1}] \tag{18b}$$

Unfortunately, for a rank-deficient $\boldsymbol{H}$ matrix, the solution of Eqs. (14) and (16) does not exist for all the steps and hence the $\boldsymbol{H}$ matrix cannot be reduced to the form of Eq. (9). In order to evaluate the parity bits in a cost efficient manner, the $\boldsymbol{H}$ parity matrix will be transformed as below

$$\boldsymbol{H} = [\boldsymbol{H}_1 \quad \boldsymbol{H}_2] \rightarrow [\boldsymbol{P} \quad \boldsymbol{T}] \tag{19}$$

Where $\boldsymbol{P}$ is the dense parity matrix in a block circulant form and $\boldsymbol{T}$ is a sparse lower block triangular matrix also in block circulant form. The parity bits are evaluated in terms of the message bits and the previously evaluated parity bits unlike the work presented by **Li et al., 2006** where the parity pits are evaluated in terms of the message bits only.
In order to evaluate the parity bits, the location of the free parity bits must be determined in advance and this is achieved by applying Gauss-Jordan elimination to the $\boldsymbol{H}$ matrix on the bit level and the free parity bits are identified where there is a zero pivot. The column indices of the matrix $\boldsymbol{T}$ where

the free bits appear are stored in the set FP. The location and number of the free parity bits are used to define two vectors, ***fn*** and ***fl***. The vector ***fn***=[fn_1, fn_2,…,fn_a] is 1×a vector where a is the number of block columns of the ***T*** matrix, the components of ***fn*** are the number of free bits in each block column of the $\boldsymbol{H}_2$ matrix. The vector ***fl***=[fl_1, fl_2,…,fl_{qa}] is 1×aq vector and it is a q fold extension of the ***fn*** vector, i.e. each component of the ***fn*** vector is repeated q times in the ***fl*** vector. To explain this, consider the example where ***H*** is an array of 3x6 circulants each circulant is a 10×10 matrix. This makes the $\boldsymbol{H}_2$ matrix to have 3 block columns and 30 bit columns, so the ***fn*** and ***fl*** vectors will have 3 and 30 components respectively. Assuming that two free bits appeared in the 1st and 11th columns of the $\boldsymbol{H}_2$ matrix, then FP={1,11}, and since q=10, this means that the 1st and 2nd block columns of $\boldsymbol{H}_2$ each has one free bit and the 3rd does not have any, so ***fn***=[1 1 0]. The first 20 components of ***fl***, fl_1-fl_{20}, will be all 1's and the last 10 components, fl_{21}-fl_{30} will be all 0's. It has been mentioned earlier that the Gauss-Jordan elimination is implemented starting from last column instead of the first and this makes the free parity bits to appear in the beginning of each block column. These definitions help in relating the parity bit index to the respective row index of the ***H*** matrix which is actually the equation used to evaluate the parity bit. This is because of the presence of the free bits; the evaluated parity bit will no longer have the same index which is represented as i in Eq (11). So, Eq. (11) will be modified to

$$v_i = y_i + z_i = \sum_{j=1}^{(b-a)r} u_j p_{i',j} + \sum_{k=1}^{i-1} v_k t_{i',k} \quad 1 \le i \le aq \quad i \notin \text{FP} \tag{20}$$

Where

$$i' = i - fl_i \tag{21}$$

Eq. (21) gives a correction factor for relating the evaluated parity bit and the evaluating parity equation, so the example above will have the parity bits evaluated as follows: the parity bits 1 and 11 are not evaluated as they are free, parity bits 2-10 and 12-20 will use equations 1-9 and 11-19 respectively (i=i'-1), equations 10 and 20 are redundant and parity bits 21-30 will use equations 21-30 (i=i'). The temporary variables y_i and z_i correspond to the first and second summation of Eq. (20) respectively and they will help in the encoder description later. The bits components v_k in the second summation are not all parity bits, some of them are the free bits that can either be extra message bits or simply set to zero according to the designer's preference, they are given the same parity bits notation only as a matter of convenience. Because the matrix ***T*** is a sparse matrix as will be seen in the results section, the summation over k actually does not involve many computations as it might seem, since only few entries $t_{i,j}$ are nonzero.

The algorithm of conveting the ***H*** matrix in the form of Eq. (19) is presented by the steps below.

1- Locating the free bits: As explained earlier, the free bits are used as extra message bits, in this case the coding rate will be

$$R = \frac{k}{n} = \frac{(b-a)q + n_{fp}}{bq} = 1 - \frac{a}{b} + \frac{n_{fp}}{bq} \tag{22}$$

Where n_{fp} is the total number of the free parity bits. Gauss-Jordan algorithm is applied to the $\boldsymbol{H}$ matrix on the bit level to diagonlize $\boldsymbol{H}_2$ and after identifying the free bits; FP, ***fn*** and ***fl*** are determined as described earlier. The diagonalizatoin of $\boldsymbol{H}_2$ may not be so straightforward, because rank($\boldsymbol{H}_2$) should be equal to rank($\boldsymbol{H}$), if it is not, then column permutations is performed on the block level to preserve the QC structure of the H matrix. In order to have rank($\boldsymbol{H}_2$)=rank($\boldsymbol{H}$), any all zero row in $\boldsymbol{H}_2$ there should be a corresponding all zero row in $\boldsymbol{H}_1$.

2- Pivoting: For the s^{th} column, $b\text{-}a\text{+}1\leq s\leq b$, find the block circulant $\boldsymbol{A}_{k,s}$, $1\leq k\leq a$, whose rank is maximum and move it to diagonal position by making block row interchange between the k^{th} block row and the r^{th} block row where $r=s\text{-}(b\text{-}a)$.

3- Solve Eq. (14) using Gauss-Jordan elimination: Because of the pivoting performed in the previous step, the possibility for having a solution for Eq. (14) is maximized since in this case the circulant $\boldsymbol{A}$ will have the maximum possible number of linearly independent rows and hence minimum number of linearly dependent rows that will be reduced to all zeros rows when transforming the augmented matrix $[\boldsymbol{A}|\boldsymbol{B}_{:,1}]$ to the reduced echelon form, and it is known that the system $[\boldsymbol{A}|\boldsymbol{B}_{:,1}]$ is consistent if a row of the form, **Meyer, 2000.**

$$[0\ 0\ 0,\ldots.0\mid x],\ x\neq 0 \tag{23}$$

Never appears. If Eq. (14) fails to have a solution for at least one block $A_{r,s}$**,** the s^{th} column of the is permuted with a block column in the left side of $\boldsymbol{H}$. This column permutation is repeated so that Eq. (14) should have a solution for all circulants above the diagonal position in order to have all blocks circulant above the diagonal are eliminated by the next step, this is necessary to transform $\boldsymbol{H}_2$ to the lower block triangular matrix $\boldsymbol{T}$. The column permutation of this step must be performed in accordance with the column permutations in the first step such that if step 1 and step 3 require column permutation at block columns s_1 and s_2 respectively with $s_1>s_2$, the column permutation at s_1 is performed. The column permutation is performed with a left side (lower index) column; the algorithm is stopped and started all over.

5- Elimination of blocks above and below the diagonal: For every column in $\boldsymbol{H}_2$, this process is performed a-1 times using Eq. (15). This step is very much related to the previous step because it depends on the solution of Eq. (14).

6- Transform the diagonal blocks to identity matrix: This is performed by Solving Eq. (16) using Gauss-Jordan elimination. This step is very much like step 3 but with different purpose, here it is required to find the matrix $\boldsymbol{X}$ that is the inverse of the diagonal matrix ($\boldsymbol{A}_{r,r+b-a}$) and by multiplying the rows by the inverses of the diagonal blocks (with the elimination step) the $\boldsymbol{H}_2$ matrix is converted to a diagonal matrix, but that does not happen if $\boldsymbol{H}$ is rank deficient and in this case some of the diagonal blocks will be rank deficient. If a diagonal block is rank deficient, it does not mean that it should be left as it is because it is no longer sparse due to the row operations of step 4. Since we know that there are free bits located in the $\boldsymbol{H}_2$ matrix and these free bits are located by the first step, then the entries of the row generator of the diagonal block corresponding to these free bits need to be set to 1's in order to be taken into account in the evaluation of the parity bit that comes next to these free bits. The polynomial of the generator row of the diagonal of $\boldsymbol{T}$ should be

$$\boldsymbol{t}_{r,r}=\sum_{i=0}^{nf_r} x^i \tag{24}$$

Note that if the r^{th} column of the $\boldsymbol{H}_2$ (or $\boldsymbol{T}$) matrix does not have free bits then nf_r=0 and hence the polynomial in Eq. (24) reduces to the polynomial of the identity matrix which is 1.The augmented column of Eq. (16b) is modified accordingly; this will make Eq. (16) to have a solution.

Due to the rank deficiency of the $\boldsymbol{H}$ matrix and the presence of the rank deficient circulants in the block diagonal positions, it is expected that not all nondiagonal block circulants are eliminated even after exhausting all possible column permutations in step 3, but it is still possible that the $\boldsymbol{H}_2$ matrix can be transformed into the $\boldsymbol{T}$ matrix. Now by looking at **Fig. 7** where the $\boldsymbol{H}$ matrix for two typical different cases is shown, important common features can be observed upon which the circuit implementation shown in **Fig. 8** is made as described below.

1- The $\boldsymbol{P}$ matrix is a dense block circulant matrix and it corresponds to the temporary variable y_i shown in Eq. (20). The implementation of this part is similar to the implementation of the full rank case shown in **Fig. 4**.
2- In the $\boldsymbol{T}$ matrix, the diagonal blocks corresponding to the columns where the there are free bits, there is no identity matrix, instead there are multiple parallel bit level diagonals which are consequence of Eq. (24). The number of these diagonal blocks is denoted by d. For this part, the corresponding temporary variable z_i given by Eq. (20) can be expressed as.

$$z_i = \sum_{k=i-fn_r}^{i-1} v_{(r-1)q+k} \tag{25}$$

Where $i=(r\text{-}1)q+t$, $fn_r+1 \le t \le q\text{-}1$, $1 \le r \le d$. Equation (25) is a reduced form of the second summation in Eq. (20) where the positions of 1's only are taken into account. An important case is that when there is only one free bit per block column, in such case $d=n_{fp}$, $fn_r=1$ and Eq. (25) reduces to

$$z_i = v_{(r-1)q+i-1} \tag{26}$$

The implementation of this part is made by first parallel loading the free bits corresponding to r^{th} block column of $\boldsymbol{T}$ into the first SR and after evaluating the current parity bit $v_i=y_i+z_i$ using the left summator, it will be serially shifted into the SR for the evaluation of the subsequent parity bit and so on until the SR is full. This is repeated d times and the parity bits are linearly shifted continuously into the subsequent SR's until all the d SR's are full, the parity bits will appear in reverse order in the SR's as shown in **Fig. 8**.
3- There are nonzero block circulants below the d rank deficient diagonal blocks mentioned above. These circulants that are not eliminated can be either dense as in **Fig. 7a** or sparse as in **Fig. 6b**. These nonzero block circulants appeared in one row for all the LDPC codes tested by this algorithm. To evaluate the parity bits in the next block column ($r=d+1$), the previously evaluated parity bits will be required which are now all stored in the SR's from the previous step. Noting that the subsequent column has no free bits and the diagonal block is identity matrix, the evaluation of the temporary variable z_i is made by adding the previously evaluated parity bits that correspond to the nonzero pits in these nonzero circulants of $\boldsymbol{T}$ that are not eliminated

$$v_i = y_i + z_i = y_i + \sum_{\substack{k=1 \\ t_{i,k=1}}}^{dq} v_k \quad dq+1 \le i \le (d+1)q \quad i \notin \text{FP} \tag{27}$$

These nonzero positions can be picked up by hardwiring and directly added by XOR gates without using AND gates, these hardwiring are shown as random taps in **Fig. (8)** taken from the SR's to the right summator. The block circulant structure of $\boldsymbol{T}$ is employed such that the contents of the SR's are now circularly shifted in order to generate the subsequent rows of the respective circulants of $\boldsymbol{T}$. A 2-to-1 multiplexers are used in the input of the SR's in order to control the choice of linear shifting and the circular shifting and also to control the evaluation of v_i according to Eq. (25,26) or Eq. (27), all multiplexers have a common control signal x. The complexity of this part is not much affected by the nonzero circulants being sparse or dense as long as they appear in one raw since hardwiring is used. If more than one row appears with non zero circulants in the $\boldsymbol{T}$ matrix, AND gates will be needed as used in the SRA described earlier. The cases that were considered in this work have all resulted in a single nonzero block raw in the $\boldsymbol{T}$ matrix, which suggest the using of hardwiring and no AND gates are needed for the implementation of the z_i part.

4- The next diagonal blocks where $d+2 \le r \le a$ are identity matrices and all circulants to the left are zero circulants, therefore $z_i=0$ and hence $v_i=y_i$ and no further process is needed.

7. RESULTS

Before discussing results, the construction of a QC-LDPC code is presented. In general the construction of QC-LDPC codes is based on finite fields or finite geometry. The codes used in this work are based on finite field and the construction is explained as follows, **Ryan and Lin, 2009**: Let GF(p) be a finite field of p elements where p is a prime number or a power of a prime. Let α be a primitive element of the field such that every nonzero element can be represented as α^i where $0 \le i < q = p-1$ with the special case of $0 = \alpha^{-\infty}$ by convention. Define the $1 \times q$ vector

$$\delta(\alpha^i) = [\, \delta_0 \; \delta_1 \ldots \delta_{q-1}] \tag{28}$$

where $\delta_i=1$ and the remaining q-1 components are all zeros, this is called the location vector of the element α^i and $\delta(0)$ is the all zero $1 \times q$ vector. Generate a $q \times q$ $\boldsymbol{A}$ matrix whose rows are the location vectors of the elements $\alpha^j \alpha^i$ $0 \le j < q$. Note that the first row of $\boldsymbol{A}$ is the location vector of α^i and the subsequent rows are each the right cyclic shift of the row above by one position. Now $\boldsymbol{A}$ is a circulant permutation matrix (CPM) assigned for each element of GF(p) and it is called the dispersion matrix of the element α^i. It is clear that for any two different elements of GF(p) their dispersion matrices are different CPM's over GF(2). Now construct the circulant matrix

$$W=\begin{bmatrix} w_0 \\ w \\ \vdots \\ w_{q-1} \end{bmatrix}=\begin{bmatrix} 0 & \alpha-1 & \alpha^2-1 & \cdots & \alpha^{q-1}-1 \\ \alpha^{q-1}-1 & 0 & \alpha-1 & \cdots & \alpha^{q-2}-1 \\ \alpha^{q-2}-1 & \alpha^{q-1}-1 & 0 & \cdots & \alpha^{q-3}-1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha-1 & \alpha^2-1 & \alpha^3-1 & \cdots & 0 \end{bmatrix} \quad (29)$$

Where the generator row $\mathbf{w}_0=\alpha^i-1$, $0\le i\le q-1$, next replace each element of $\boldsymbol{W}$ by its respective dispersion CPM to obtain the array of CPM's

$$H_{qc,disp}=\begin{bmatrix} 0 & A_1 & A_2 & \cdots & A_{q-1} \\ A_{q-1} & 0 & A_1 & \cdots & A_{q-2} \\ A_{q-2} & A_{q-1} & 0 & \cdots & A_{q-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_1 & A_2 & A_3 & \cdots & 0 \end{bmatrix} \quad (30)$$

Where the CPM $\boldsymbol{A}_i$ is the dispersion matrix of the element α^i-1, $0\le i\le q-1$. $\boldsymbol{H}_{qc,disp}$ is a $q\times q$ array of CPM's each of size $q\times q$ and the diagonal blocks are $q\times q$ matrices. The QC-LDPC code parity matrix $\boldsymbol{H}$ is taken as an $a\times b$ subarray from the $\boldsymbol{H}_{qc,disp}$ matrix where $1\le a<b\le q$, if this subarray is chosen such that it lies either above or below the main diagonal of $\boldsymbol{H}_{qc,disp}$, then it contain no zero submatrix and hence will have a constant row and column weights and therefore will make a regular QC-LDPC code. The constructed QC-LDPC code satisfies the RC-constraint mentioned earlier and hence has a girth of at least 6, **Ryan and Lin, 2009**. The choice of the parameters a and b controls the choice of the code size ($n=bq$) and the code rate which is at least $(b-a)/b$. In this work the $\boldsymbol{H}$ matrix is chosen as the subarray from the bottom left corner of $\boldsymbol{H}_{qc,disp}$, this will impose the condition $a+b\le q$ in order to have a regular code, otherwise the code is irregular. The values of p are all chosen as prime numbers. **Table 1** below shows the chosen code parameters along with the amount of component needed by the circuit of **Fig. 8** in terms of number of FF's and XOR gates. It has been noticed that for all codes used in the result analysis that after the block diagonalization, the free bits are distributed such that there is one free bit in each block column, this means that $d=n_{fp}$. Different codes parameters are used for the analysis, the codes length ranges from n=48 to n=5000 with coding rate of R=0.5 to R=0.72. It has been assumed that the $\boldsymbol{P}$ matrix has an equal number of zeros and 1's and it will be taken as a benchmark for measuring the extra hardware required by the $\boldsymbol{T}$ matrix which is due to the rank deficiency of the matrix $\boldsymbol{H}$. As discussed before there will be extra FF's and XOR gates needed to work on the part of the $\boldsymbol{T}$ matrix where there are some block circulants that are not eliminated. **Table 1** shows both the numbers of these FF's and XOR gates and there percentage relative to the number of FF's and XOR's needed by the $\boldsymbol{P}$ part. This is due to the fact that the effort required to implement the $\boldsymbol{P}$ matrix part is the same whether the $\boldsymbol{H}$ matrix is full rank or rank deficient. The number of FF and XOR gates needed by the $\boldsymbol{P}$ matrix according to the circuit implementation of **Fig. 4** is k and k/2 respectively because all the information bits are needed to be stored and the number of XOR gates equals the number of 1's (which are assume to equal the number of 0's) in the generator rows $\boldsymbol{p}_{r,s}$. The number of FF's and XOR gates required by the $\boldsymbol{T}$

matrix are shown under the column title FF and XOR respectively and their percentages are shown under the column titles δ_F and δ_X.

Table 1. The number of FF's and XOR gates needed to implement the ***T*** matrix part for QC-LDPC codes with different parameters.

Code Parameters						Results Parameters					Code Parameters						Results Parameters				
p	a	b	n	k	R	d	FF	XOR	δ_F	δ_X	p	a	b	n	k	R	d	FF	XOR	δ_F	δ_X
13	4	8	96	48	0.50	3	36	3	75.0	12.5	19	6	13	234	126	0.54	4	72	4	57.1	7.4
13	4	9	108	60	0.56	2	24	2	40.0	8.3	19	6	14	252	144	0.57	3	54	29	37.5	53.7
13	4	10	120	72	0.60	1	12	9	16.7	37.5	31	7	14	420	210	0.50	6	180	6	85.7	5.7
13	5	10	120	60	0.50	1	12	5	20.0	16.7	31	7	15	450	240	0.53	6	180	6	75.0	5.7
17	5	10	160	80	0.50	4	64	4	80.0	10.0	31	7	16	480	270	0.56	6	180	6	66.7	5.7
17	5	11	176	96	0.55	4	64	4	66.7	10.0	31	8	16	480	240	0.50	7	210	7	87.5	5.8
17	5	12	192	112	0.58	3	48	3	42.9	7.5	31	8	17	510	270	0.53	7	210	7	77.8	5.8
17	5	13	208	128	0.62	2	32	14	25.0	35.0	31	8	18	540	300	0.56	7	210	7	70.0	5.8
17	5	14	224	144	0.64	1	16	9	11.1	22.5	31	8	19	570	330	0.58	7	210	7	63.6	5.8
17	6	12	192	96	0.50	3	48	17	50.0	35.4	31	8	20	600	360	0.60	7	210	7	58.3	5.8
17	6	13	208	112	0.54	2	32	16	28.6	33.3	31	8	22	660	420	0.64	7	210	7	50.0	5.8
17	6	14	224	128	0.57	1	16	7	12.5	14.6	31	8	24	720	480	0.67	5	150	71	31.3	59.2
17	7	14	224	112	0.50	1	16	9	14.3	16.1	31	8	28	840	600	0.71	1	30	13	5.0	10.8
19	4	8	144	72	0.50	3	54	3	75.0	8.3	53	10	20	1040	520	0.50	9	468	9	90.0	3.5
19	4	9	162	90	0.56	3	54	3	60.0	8.3	53	15	50	2600	1820	0.70	1	52	21	2.9	5.4
19	5	10	180	90	0.50	4	72	4	80.0	8.9	53	15	30	1560	780	0.50	14	728	14	93.3	3.6
19	5	11	198	108	0.55	4	72	4	66.7	8.9	53	20	40	2080	1040	0.50	11	572	281	55.0	54.0
19	5	12	216	126	0.58	4	72	4	57.1	8.9	73	20	40	2880	1440	0.50	19	1368	19	95.0	2.6
19	5	13	234	144	0.62	4	72	4	50.0	8.9	73	25	50	3600	1800	0.50	21	1512	725	84.0	80.6
19	6	12	216	108	0.50	5	90	5	83.3	9.3	101	25	50	5000	2500	0.50	24	2400	24	96.0	1.9

It can be seen that the number of FF is directly proportional to the value of *d* and actually it equals *dq*, (see **Fig. 8**). The number of XOR gates is not exactly proportional to *d* because as has been mentioned earlier that the block circulants that has not been eliminated in the ***T*** matrix can be either dense or sparse and that affect the number of the XOR gates. A frequency analysis of the value δ_X shows that about 65% of the codes have the value of δ_X below 10%. This means that many codes have their ***T*** matrix with sparse nonzero block circulants which significantly reduces the number of XOR gates.

8. CONCLUSION

An encoding scheme of QC-LDPC codes that have rank defficien H matrix is presented. The low density block circulant H matrix is diagonlized on the block level so each pariy bit is evaluated in terms of the message bits, free parity bits and the previously evaluated parity bits. In this way the parity bits can be evaluated serially. Extra logic is needed for the renk deficient case ove the full rank case which is investigated and the results shows that QC-LDPC codes of certain parameters can have a very small amount of the extra logic compared to the overall required logic.

REFERENCES

- Andrews, K., Dolinar, S., and Thorpe, J., *Encoders for Block-Circulant LDPC codes*, IEEE International Symposium on Information Theory, 2005. ISIT, Australia, 2005.
- Freundlich, S, Burshtein, D., and Litsyn, S., *Approximately Lower Triangular Ensembles of LDPC Codes with Linear Encoding Complexity*, IEEE Transactions on Information Theory, Vol. 53, No. 4, April 2007.
- Gallager, R. G., *Low Density Parity-Check Codes*, IRE Trans. Information Theory, vol. 8, No. 1, pp. 21–28, January 1962.
- Huang, Q., Tang, L., He, S., Xiong, Z., and Wang, Z., *Low-Complexity Encoding of Quasi-Cyclic Codes Based on Galois Fourier Transform*, IEEE Transactions on Communications, Vol. 62, No. 6, June 2014.
- Hwang, S., and Lee, H., *Block-Circulant RS-LDPC Code: Code Construction and Efficient Decoder Design*, IEEE Transactions On Very Large Scale Integration (VLSI) Systems, Vol. 21, No. 7, July 2013.
- Joyner, D., Kreminski, R. and Turisco, J., *Applied Abstract Algebra*, Johns Hopkins Press, 2004.
- Li, Z., Chen, L., Zeng, L., Lin, S., and Fong, W. H., *Efficient Encoding of Quasi-Cyclic Low-Density Parity-Check Codes*, IEEE Trans. Communications, Vol. 54, No. 1, 2006.
- Lu, J. and Moura, J. M. F., *Linear Time Encoding of LDPC Codes*, IEEE Transactions on Information Theory, Vol. 56, No. 1, January 2010.
- MacKay, D., *Good Error Correcting Codes Based on Very Sparse Matrices*, IEEE Trans. Information Theory, Vol. 45, no. 3, pp. 399–431, March 1999.
- Meyer, C. D., *Matrix Analysis and Applied Linear Algebra*, SIAM, 2000.
- Moreira, J. C, and Farrell, P. G., *Essentials of Error-Control Coding*, John Wiley, 2006.
- Park, H., Hong, S., No, J. S. and Shin, D. J., *Protograph Design With Multiple Edges for Regular QC LDPC Codes Having Large Girth*, IEEE International Symposium on Information Theory Proceedings, St. Petersburg, 2011.
- Richardson, T. J. and Urbanke, R., *Efficient Encoding Of Low-Density Parity-Check Codes*, IEEE Trans. Information Theory, vol. 47, no. 2, pp. 638–656, February 2001.
- Ryan, W. E. and Lin, S., *Channel Codes Classical and Modern*, Cambridge University Press, 2009.
- Tao, X., , Zheng, L., Liu, W., and Liu, D., *Recursive Design of High Girth (2,k) LDPC Codes from (k,k) LDPC Codes*, IEEE Communications Letters, Vol. 15, No. 1, January 2011
- Song, S., Zhou, B., Lin, S., and Abdel-Ghaffar, K. *A Unified Approach to the Construction of Binary and Nonbinary Quasi-Cyclic LDPC Codes Based on Finite Fields*, IEEE Transactions on Communications, Vol. 57, No. 1, January 2009.
- Wang, Y., Draper S. C., and Yedidia, J. S, *Hierarchical and High-Girth QC LDPC Codes*, IEEE Transactions On Information Theory, Vol. 59, No. 7, July 2013.
- Xia, D, He, H., Xu, Y. and Cai, Y., *A Novel Construction Scheme with Linear Encoding Complexity for LDPC Codes*, IEEE 4th International Conference on Wireless Communications, Networking and Mobile Computing, 2008, China.

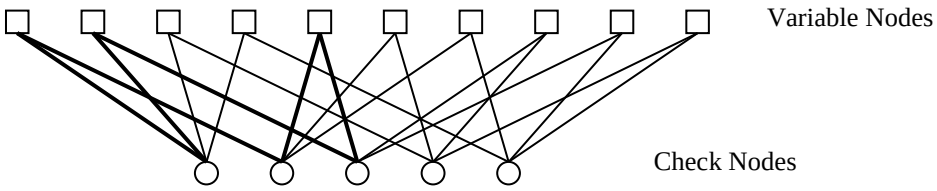


Figure 1. Tanner graph of girth equals to 6.

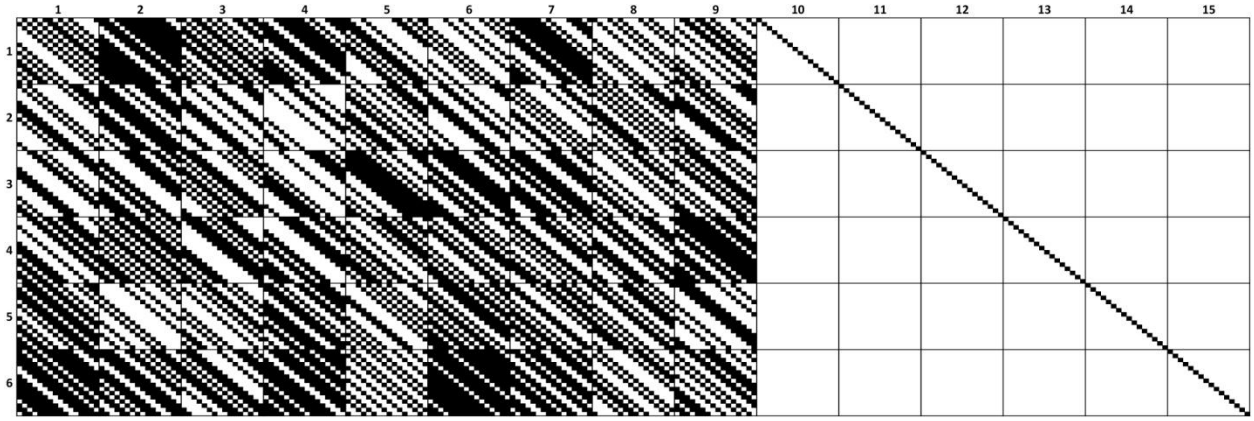


Figure 2. The diagonalizationof a full rank QC parity matrix.

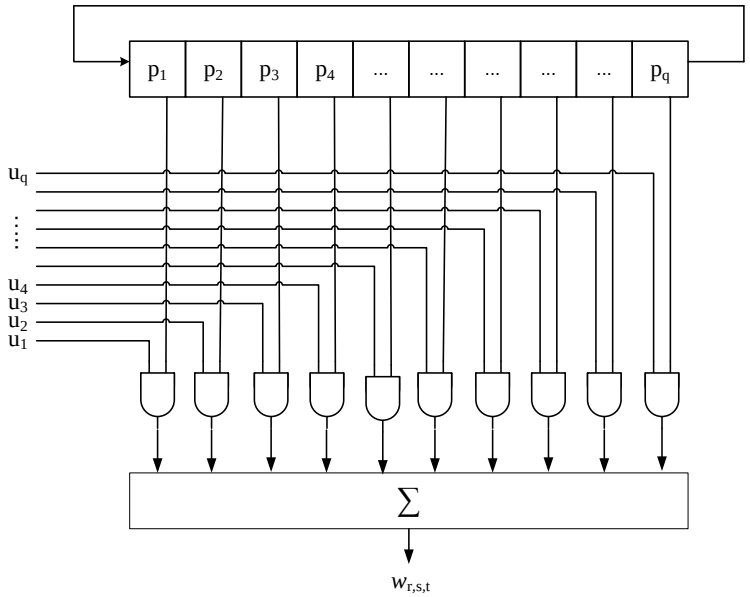


Figure 3. An SRA block of a QC-LDPC encoder.

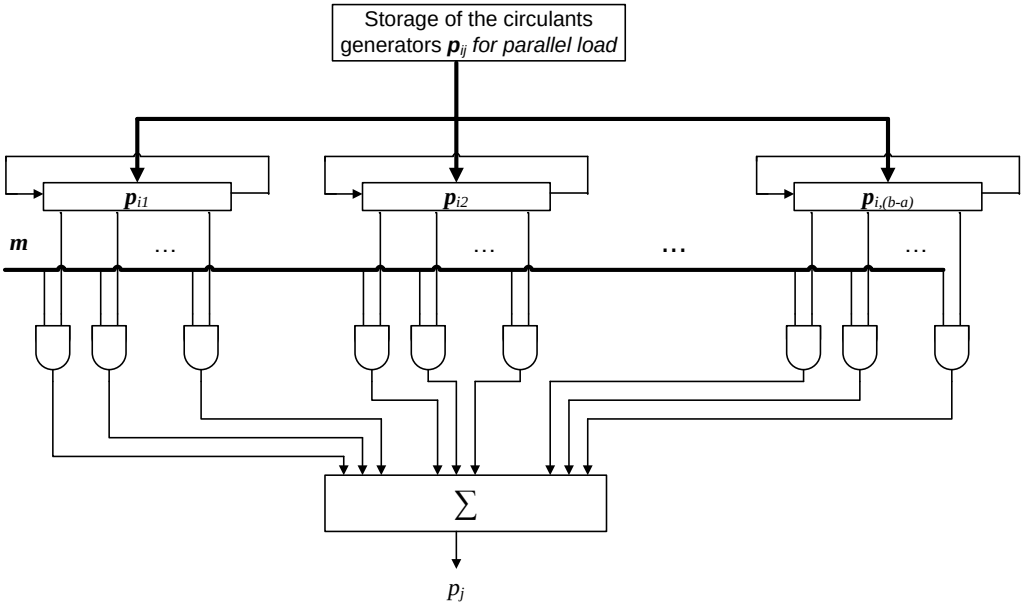


Figure 4. Encoder circuit of QC LDPC code with full rank parity matrix.

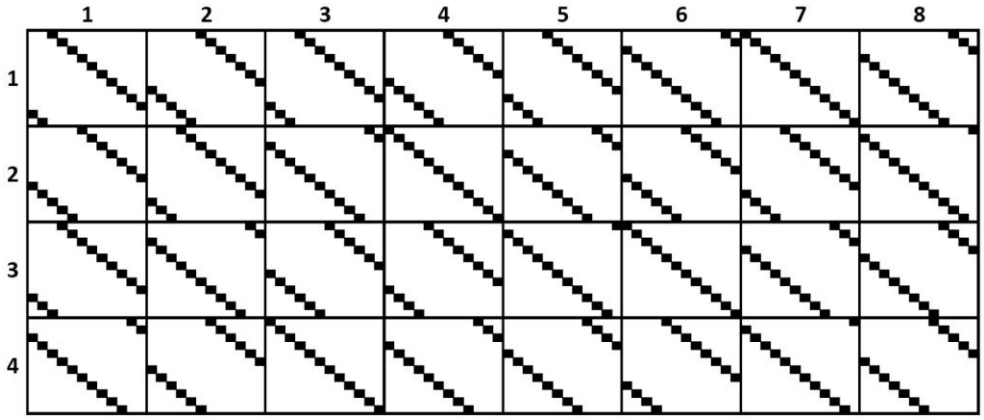


Figure 5. A QC parity matrix with rank deficiency.

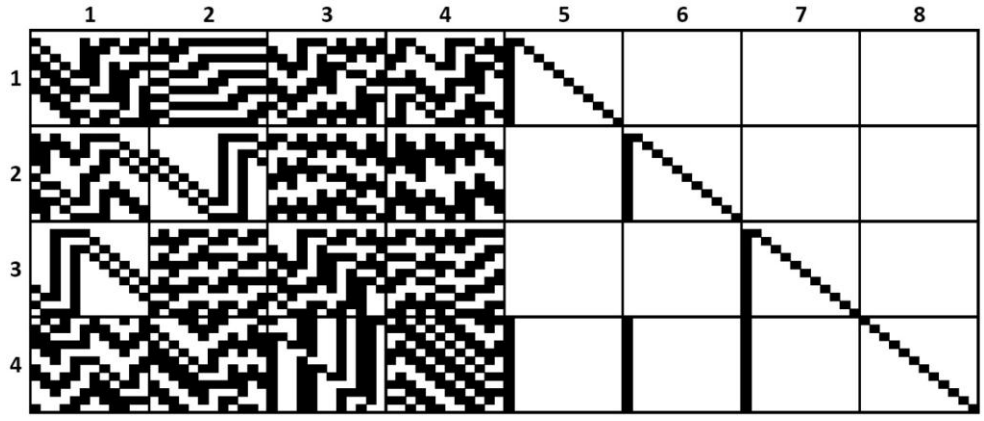
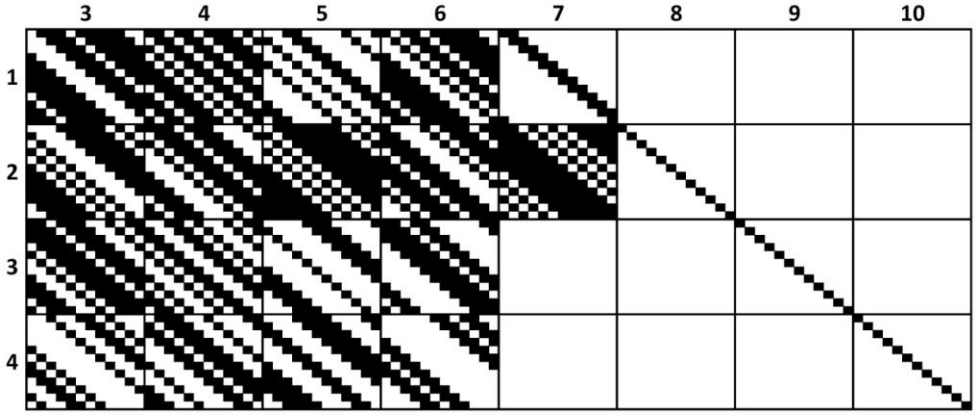
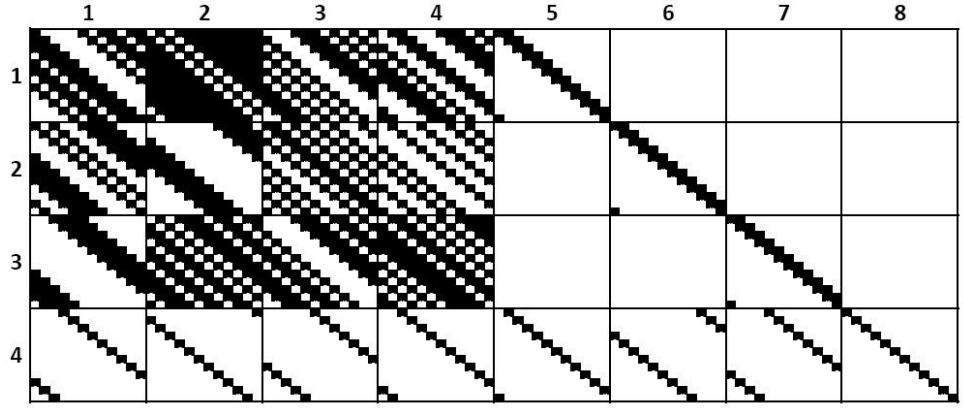


Figure 6. Bit level diagonalization of a rank deficient ***H*** matrix of **Fig. 2**.



(a) ***T*** matrix with nonzero dense circulant



(b) ***T*** matrix with nonzero sparse circulants

Figure 7. Block level diagonalization of a rank deficient ***H*** matrix.

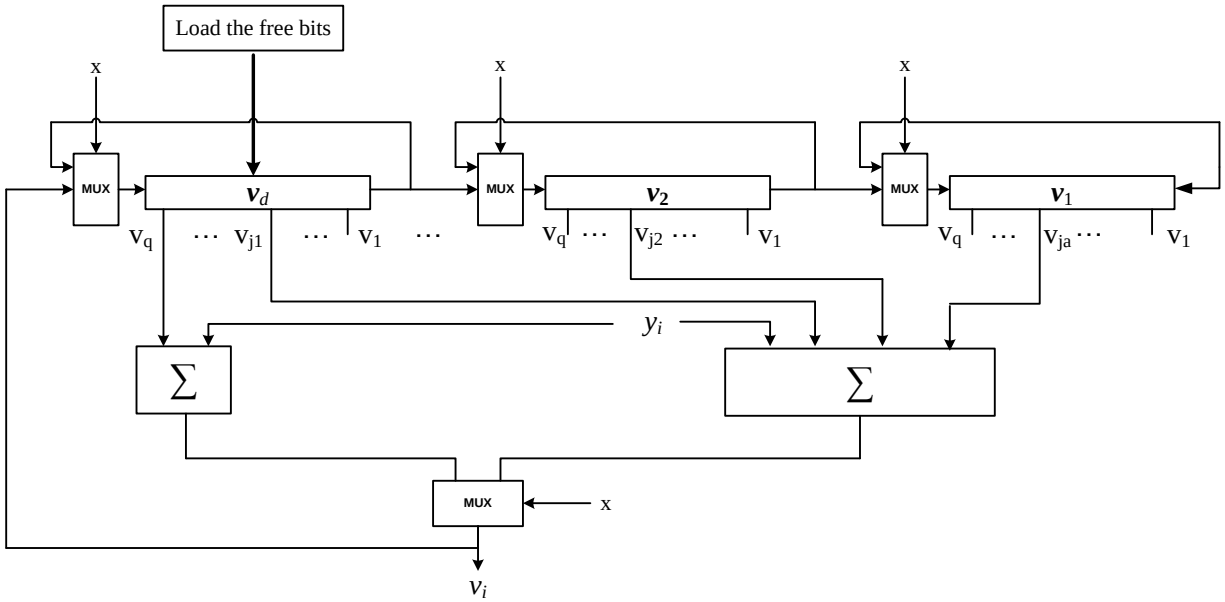


Figure 8. The encoder circuit of QC LDPC code with rank deficient parity matrix.