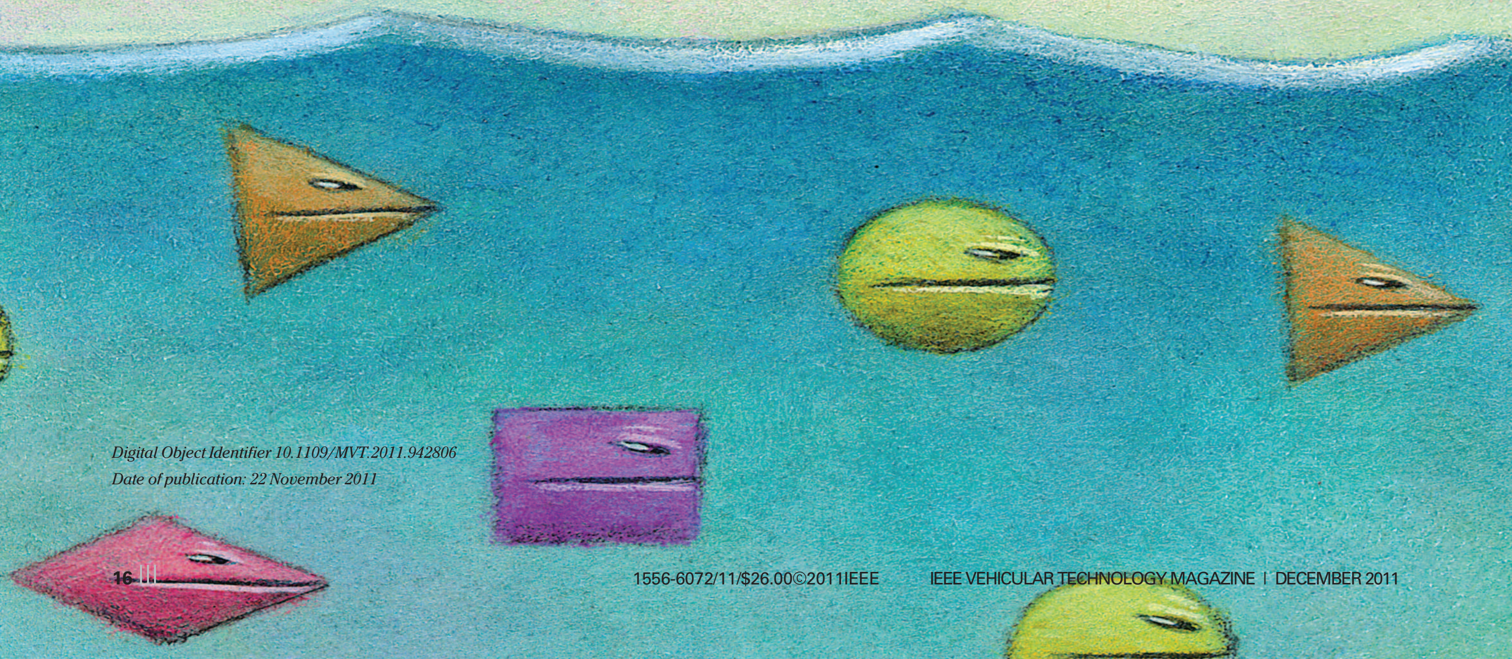
DESIGN OF LOW-DENSITY PARITY-CHECK CODES

An Overview

Nicholas Bonello, Sheng Chen, and Lajos Hanzo

This article provides an overview of the conflicting design tradeoffs of low-density parity-check (LDPC) codes and thus advocates a more holistic approach to their design for wireless channels. We reveal some of the intricate interdependencies of the LDPC code parameters and hence recommend designing codes that strike an attractive tradeoff concerning a number of desirable attributes, rather than simply designing codes that closely approach capacity but possess less-attractive hardware implementations.

The birth of information and coding theory is marked by Shannon's seminal article, "A Mathematical Theory of Communication" [1], published in 1948, in which it was demonstrated that arbitrarily reliable communication of information over an unreliable channel is possible provided that the transmission rate, i.e., the amount of information-carrying bits transmitted over the channel per second is less than the channel capacity. The fundamental idea is that of incorporating a channel code that imposes redundancy on the data bits before their transmission. Therefore, according to Shannon, it is the channel capacity that determines the minimum possible amount of redundancy that has to be incorporated to be able to correct the errors imposed by the channel. However, Shannon's theory proved the existence of capacity-approaching codes but refrained from suggesting any specific practical coding schemes that can achieve this aim. This naturally triggered a widespread endeavor by the research community leading to diverse extensions, deeper interpretations, and further pragmatic realizations of Shannon's original work. For decades, the principal aim of code designers was to reduce the distance, measured in decibels, from this ultimate channel capacity bound promised by Shannon, and thus the channel code's performance was typically assessed by means of plots of the bit error ratio (BER) or block error ratio (BLER) versus the channel's signal-to-noise ratio (SNR) or the ratio of the energy per uncoded bit to the noise power spectral density, commonly denoted by E_b/N_0. This race toward capacity reached its pinnacle by the discovery of the iterative decoding principle, which was proposed in the now-classic turbo [2] and LDPC [3] codes, both of which provided



Digital Object Identifier 10.1109/MVT.2011.942806
Date of publication: 22 November 2011

the most satisfying solutions to Shannon's challenge up to this point in time.

Following these notable developments, one can argue that the original research problem posed by Shannon must now be reassessed and tackled from a wider perspective. Let us commence our discussions by posing the following two questions:

- In terms of which particular performance metric should a channel code be evaluated? Should code designers focus solely on the attainable BER/BLER performance or should other attributes be also taken into consideration?
- What is the complexity penalty as well as the implementation-related ramification of attaining a BER/BLER performance that is within a minute margin of the channel's capacity bound?

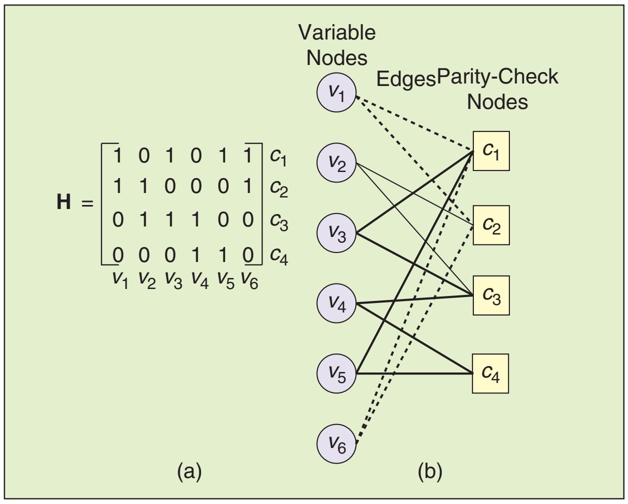
Preliminaries

LDPC codes form part of a larger family of codes, which are typically referred to as *linear block codes*. A code is termed a *block code* if the original information bit sequence can be segmented into fixed-length message blocks, hereby denoted by $\mathbf{u}$, each having K information digits. This implies that there are 2^K possible distinct message blocks. For the sake of simplicity, we will only be considering binary LDPC codes, i.e., the codes associated with the logical symbols/bits of $\{1, 0\}$. The LDPC encoder is then capable of transforming each input message block $\mathbf{u}$ according to a predefined set of rules into a distinct N-tuple (N-bit sequence) $\mathbf{z}$, which is typically referred to as the *code word*. The code word length N, where $N > K$, is then referred to as the *block length*.

The number of nonzero symbols of a code word $\mathbf{z}$ is called the *weight*, whereas the number of bit positions in which two code words differ is termed the *distance*. (Code words that have a low number of binary ones are referred to as *low-weight* code words.) For instance, the distance between the code words $\mathbf{z}_1 = (1101001)$ and $\mathbf{z}_2 = (0100101)$ is equal to three. The minimum distance, hereby denoted by $d_{\min}$, corresponds to the minimum distance between two of its code words, and for a linear code, $d_{\min}$ is determined by the weight of the nonzero code word that has the minimum weight.

The unique and distinctive nature of the code words implies that there is a one-to-one mapping between a K-bit information sequence $\mathbf{u}$ and the corresponding N-bit code word $\mathbf{z}$ described by the set of rules of the encoder. Clearly, if both K and N are small, then the 2^K distinct message blocks and the corresponding code words can be stored in a look-up table (LUT). However, for large K, the 2^K-entry LUT encoder will be prohibitively complex. This complexity is significantly reduced by the fact that LDPC codes are linear codes, and therefore, the modulo-2 sum (eXclusive OR) of any two or more code words is another code word. In fact, the code word $\mathbf{z}$ can be calculated by multiplying the input message sequence $\mathbf{u}$ with a $(K \times N)$-element matrix $\mathbf{G}$, which is referred to as the *generator matrix* (GM). So, if we consider the simple example of having a four-bit input message sequence $\mathbf{u}$ and assume that the ith column of $\mathbf{G}$ is given by $[1101]$, then the ith bit of the code word $\mathbf{z}$ will be equal to the modulo-2 sum of the first, second, and fourth bits of $\mathbf{u}$.

However, there is another useful matrix associated with a linear block code. This matrix is referred to as the *parity-check matrix* (PCM), which is typically denoted by $\mathbf{H}$ and contains $M \geq (N - K)$ rows and N columns. A characteristic of the PCM of LDPC codes is that it is sparse, i.e., there are fewer ones than there are zeros. As a result, their PCM is said to have a low density—hence, the terminology of LDPC codes. The code rate R of an LDPC code will then satisfy $R \geq 1 - M/N$, where the equality holds when all the rows of the underlying PCM are linearly independent. (In this specific case, the PCM has $M = N - K$ rows.) The PCM can also be represented graphically by what is known as a *bipartite graph*, as exemplified in Figure 1. Let us consider, as an example, the LDPC code having $N = 6$, associated with the PCM shown in Figure 1(a). The corresponding graph is then illustrated in Figure 1(b). It can be observed that this graph can be divided in two parts (hence, the terminology bipartite), whereby the right-hand side of the graph shows the parity-check nodes, which correspond to a row of the PCM $\mathbf{H}$, whereas the left-hand side (LHS) contains the variable nodes, which relate to the columns of the PCM $\mathbf{H}$. A variable node is, essentially, a transmitted bit in the code word $\mathbf{z}$. The ones in the PCM $\mathbf{H}$ of Figure 1(a) represent the edges that interconnect the parity-check nodes and variable nodes located on the graph of Figure 1(b). For example, one can observe from Figure 1(b) that the first parity-check node c_1 is checking the result of the modulo-2 sum (called the *parity*) of v_1, v_3, v_5, and v_6, which is also seen in the first row of the corresponding PCM; i.e., if the transmitted bits represented by v_1, v_3, v_5, and v_6 are received correct, then the value of $v_1 \oplus v_3 \oplus v_5 \oplus v_6 \oplus c_1 = 0$. Here, we denote the modulo-2



$$\mathbf{H} = \begin{array}{cccccccc} & 1 & 0 & 1 & 0 & 1 & 1 & c_1 \\ & 1 & 1 & 0 & 0 & 0 & 1 & c_2 \\ & 0 & 1 & 1 & 1 & 0 & 0 & c_3 \\ & 0 & 0 & 0 & 1 & 1 & 0 & c_4 \\ & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & \end{array}$$

FIGURE 1 (a) A PCM. (b) The bipartite graph having girth of four and corresponding to the PCM of (a). A cycle of six (represented by the continuous bold lines) and a cycle of four (represented by dashed bold lines) are shown.

operation by $\oplus$. We also remark that the edge interconnections between the variable and check node essentially constitute an interleaver, and hence LDPC-coded systems can dispose of the channel interleaver when transmitting over Rayleigh channels.

Adopting a Holistic Approach to the LDPC Code Design

Previously, we have briefly touched upon adopting a more holistic approach to the design of LDPC codes. We hinted that the performance attributes of codes, in this case those of LDPC codes, must be viewed from a wider perspective that also takes into account other factors, such as the practicality of the code as well as the ease/difficulty of implementation. In this respect we have attempted to summarize in Figure 2 the most important tradeoffs associated with the LDPC code design. More specifically, we divided these tradeoffs into four categories, namely, the BER/BLER performance metrics, the construction attributes, and the encoder and decoder implementation-related characteristics, all of which will be described in more detail in the forthcoming sections. It will soon become evident that trying to find a good balance between these design attributes is definitely not a simple task since most of the parameters are conflicting, i.e., favoring one particular attribute will almost certainly result in attaining inferior features for one or more of the remaining parameters.

BER/BLER Performance Metrics

The overall BER/BLER versus SNR performance of an LDPC code is generally described by two different regions and a threshold. The first region is commonly referred to as the *waterfall region*, which corresponds to the low-to-medium SNR region of the BER/BLER versus SNR plot. By contrast, the error floor [4] is located at the bottom of the waterfall-shaped curve where it can be observed that the BER/BLER no longer exhibits the rapid improvement as in the waterfall region. More often than not, the error floor is not explicitly visible in the corresponding BER/BLER plot since it is below the BERs readily generated by the simulation performed. There is the parlance of turbo cliff, above which the BER/BLER performance improves rapidly upon increasing the SNR. The performance of an LDPC code in the different regions as well as the actual value of the turbo-cliff SNR depend on various code construction attributes, which will be detailed in the next section.

Construction Attributes

The construction attributes of LDPC codes are best understood in the context of their associated PCM or the corresponding bipartite graph; hence; we will build upon the preliminary concepts and definitions introduced in the "Preliminaries" section.

Regular Versus Irregular Parity-Check Matrices

One of the first dilemmas faced when designing LDPC codes is that of choosing between a regular or an irregular construction. To understand the difference between these two attributes, we will introduce further basic definitions related to graph theory. We will start by defining the degree of a node in a graph as the number of edges emerging from that node. Hence, a parity-check node's degree, hereby denoted by ρ, is the number of ones in the relevant row of the associated PCM, and a variable node's degree γ is the number of ones in the corresponding column of the PCM $\mathbf{H}$. The parameters ρ and γ are also referred to as the *row and column weights* of the $\mathbf{H}$ matrix. Subsequently, an LDPC code is said to be regular if every parity-check node contained in its underlying graph is connected to ρ variable nodes, whilst every variable node is connected to γ parity-check nodes. If this is not the case, the code and its associated graph are termed to be irregular. For example, the LDPC code represented by the graph of Figure 1 is irregular or, more accurately, left regular since it is only the variable nodes located on the LHS of the graph that have the same degree of two. Carefully designed irregular LDPC codes can attain a lower turbo-cliff SNR than regular codes of the same rate; i.e., their exhibited BER/BLER starts to rapidly

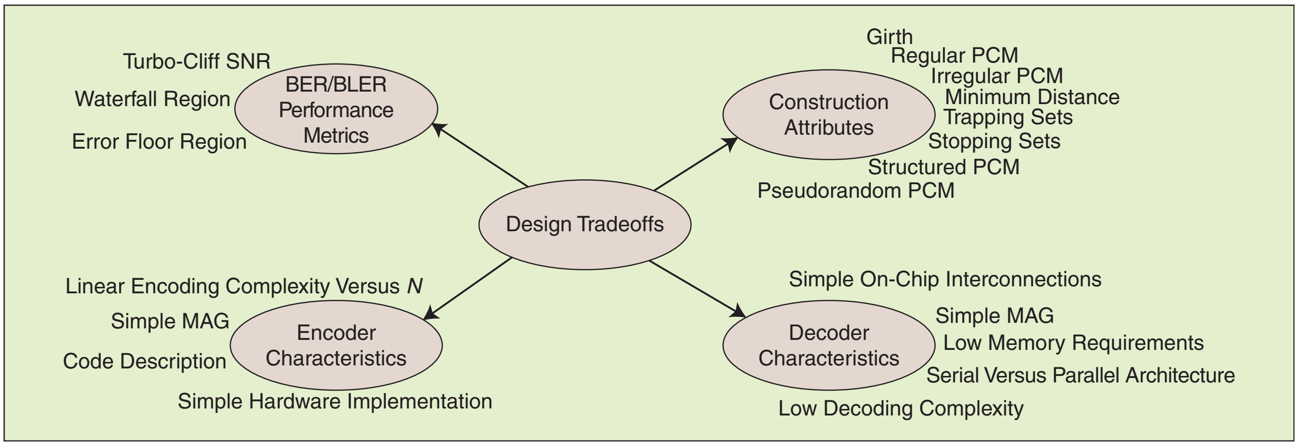


FIGURE 2 Conflicting design factors related to LDPC code construction. PCM: parity check matrix. MAG: memory address generation.

decrease at a lower SNR value, and hence, their BER/BLER performance is superior in the waterfall region. The reason for this phenomenon lies in the conflicting (ideal) requirements of the variable and parity-check nodes, whereby the variable nodes benefit from having large degrees, strongly protecting them. By contrast, a parity-check node should have a low degree to prevent error propagation when it is corrupted. However, we note that the superior BER/BLER performance of irregular LDPC codes is achieved at the expense of a potentially increased implemental complexity, since it is slightly more challenging to design a hardware architecture that is sufficiently flexible to support an implementation having different row and column degrees.

Previously, we have emphasized that irregular LDPC codes must be carefully designed for two main reasons. First, the design of irregular codes necessitates the use of sophisticated techniques, such as density evolution [5] or extrinsic information transfer (EXIT) charts [6], both of which can predict the value of the turbo-cliff SNR. Both density evolution and EXIT charts can provide actual (nonuniform) distributions for the row and column weights of the irregular PCM. However, we note that both of these techniques assume that the LDPC code has a high block length (in the region of $N \geq 10,000$ b) and will be decoded by a decoder that can afford a high number of independent iterations. (This is related to what is referred to as the *girth* of the underlying graph, which will be defined in the "The Girth of the Underlying Graph" section.) Second, the BER/BLER performance exhibited by irregular LDPC codes is inferior to that exhibited by regular LDPC codes in the error floor region, unless specific techniques are employed at the PCM design stage. These specific techniques are referred to, in parlance, as *conditioning* and will be described in the "Code-Construction Attributes Affecting the Error Floor" section.

LDPC codes are decoded using the sum–product algorithm (SPA) where messages are iteratively exchanged between the nodes residing at both sides of the bipartite graph. The girth influences the achievable BER/BLER in the waterfall region because short cycles prevent the decoder from gleaning independent parity-check information. Therefore, the higher the girth, the faster the iteration-aided BER/BLER improvement, and this is why many construction techniques (collectively referred to by the term *girth conditioning*) attempt to maximize the girth of the underlying graph (see [7] and the references therein). Specifically, Gallager [3] demonstrated that the number of independent iterations T for an LDPC code having a girth of g, i.e., the iterations that provide valuable extrinsic information, is bounded by $T < g/4 \leq T + 1$. Clearly, for the girth to be high, the block length also has to be sufficiently high. It was, in fact, proven in [3] that the maximum girth of the bipartite graph is bounded by a logarithmic function of the block length as well as the row and column weights of the underlying PCM. Gallager [3] also provided the loose lower bound on the required block length for achieving a specific girth, which is shown in Figure 3(a) for girths of 6, 8, 10, and 12 and assuming regular LDPC code constructions associated with a PCM having a column weight of $\gamma = 3$. For example, it can be observed from Figure 3(a) that a rate $R = 0.75$ regular LDPC code associated with a PCM having $\gamma = 3$ must have a block length of at least $N = 6,084$ b to have a girth of 12. In Figure 3(b), we have sought to portray the relationship among the PCM column weight, girth of the underlying graph, and Gallager's lower bound on the block length N. For instance, it can be verified from Figure 3(b) that a block length of at least $N = 12,195$ b will be required to realize a similar rate $R = 0.75$ regular LDPC but having a PCM associated with $\gamma = 4$; i.e., increasing the column weight by one will require at least twice the block

The Girth of the Underlying Graph

Let us once again focus our attention on the bipartite graph illustrated in Figure 1(b). A cycle in a graph refers to a particular chain of nodes forming a closed loop where the initial and final nodes are same and no edge is used more than once. The number of edges in a cycle is then called the *length* of the cycle, and the shortest cycle length of the graph corresponds to the girth. The girth in a bipartite graph is always even and its smallest value is four. The graph depicted in Figure 1(b) has a girth of four, and the corresponding cycle of four is shown by the dashed bold edges. A cycle of six is shown by the continuous bold edges.

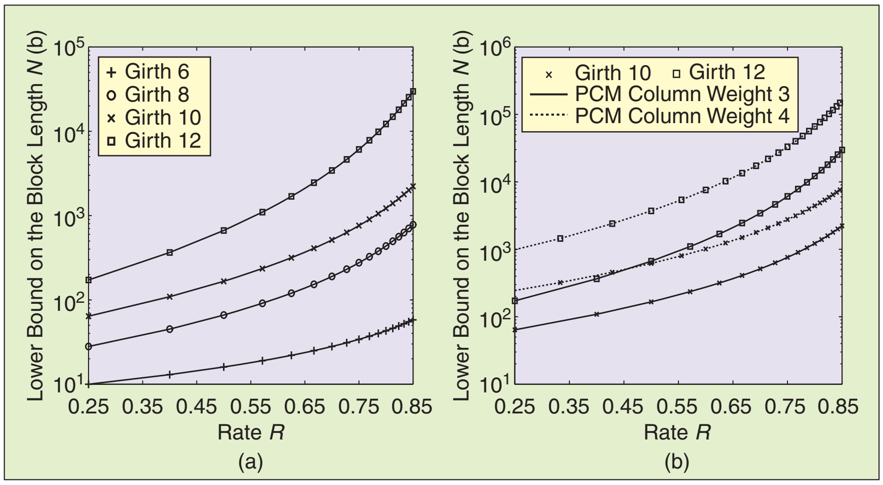


FIGURE 3 (a) Gallager's loose lower bound [3] on the value of the block length N that is required to realize an LDPC code having a code rate of R and girths of 6, 8, 10, and 12. These bounds have been calculated for regular LDPC codes that are associated with a PCM having a column weight of $\gamma = 3$. (b) A comparison of Gallager's loose lower bound [3] on the value of the block length N that is required to realize a regular LDPC code associated with a PCM having column weights of $\gamma = 3$ and $\gamma = 4$ and girths of 10 and 12.

length to realize an LDPC code having the same girth. As will be discussed in the "Code-Construction Attributes Affecting the Error Floor" section, increasing the column weight γ and, subsequently, the row weight ρ may be beneficial in some other aspect; however, at this stage we can reasonably conclude that increasing the PCM weights degrades both the girth of the corresponding graph as well its BER/BLER performance in the waterfall region.

Code-Construction Attributes Affecting the Error Floor

The performance in the error floor region depends on three main factors, namely 1) on $d_{\min}$ as well as the presence of particular graphical structures in the underlying graph, which are referred to as 2) *stopping sets* and 3) *trapping sets*. We note that whilst having a large $d_{\min}$ is always beneficial, the performance of the LDPC codes in the error floor region is largely dependent on the presence of stopping and trapping sets. We further notice that stopping sets characterize the performance of the code on erasure channels, whereas trapping sets play an analogous role for noisy channels, such as the additive white Gaussian noise (AWGN) and binary symmetric channels. (In addition to the attributes mentioned in this article, contemporary research is also focusing on the effects of pseudocode words, instanstons, and absorbing sets; however, these issues go beyond the scope of our contribution. The interested reader is referred to [8] for relevant pointers to the related literature.) We will continue our discourse by discussing each of these factors in more detail.

Classical coding theory has always placed strong emphasis on trying to design codes that have a large $d_{\min}$, which is clearly justified when one recalls the fact that a code can correct up to $\lfloor (d_{\min}-1)/2 \rfloor$ errors when using a bounded distance decoder, where $\lfloor x \rfloor$ denotes the floor function that gives the largest integer less than or equal to x. Tanner [9] derived the lower bounds on the achievable $d_{\min}$ of an LDPC code and demonstrated that these increase with both the PCM column weight as well as girth of the underlying graph. (According to these bounds, a regular LDPC code having a girth of 10 and with a $\gamma = 3$ will attain a $d_{\min} \geq 10$ whereas that code having the same girth but with a $\gamma = 4$ will attain a $d_{\min} \geq 17$. Moreover, a regular LDPC code having the same $\gamma = 4$ but with a higher girth of 12 will achieve a $d_{\min} \geq 26$.) However, the relationship between these parameters is quite intricate. Whilst increasing the girth or column weight of the associated PCM improves the minimum distance, we have seen in the "The Girth of the Underlying Graph" section that an increase in the column weight (for a given constant N) will degrade the girth. Hence, if we consider two LDPC codes having the same rate but different column weights, the code having the higher column weight will exhibit a lower error floor owing to its higher $d_{\min}$ (if it was carefully designed) but a worse BER/BLER in the waterfall region due to its lower girth. Another interesting point to make is that whilst a code having a high girth is always preferred, ironically, completely cycle-free codes constitute bad codes due to their low minimum distance [10]. Another deficient family of LDPC codes is constituted by those having a PCM with $\gamma = 2$. It was established in [9] that such codes have $d_{\min} = g/2$. Moreover, Gallager [3] also indicated that the PCM of LDPC codes must have $\gamma \geq 3$ for the minimum distance to increase linearly (rather than logarithmically) with the block length, albeit we will discuss later in the "Structured versus Pseudorandom Parity-Check Matrices" section that this statement is not applicable for some structured LDPC codes.

In the "Preliminaries" section, we argued that a code having a small $d_{\min}$ is characterized by the presence of low-weight code words. These will cause undetected errors since the decoding process will find a code word that is not the originally transmitted one. However, given the fact that $d_{\min}$ of most LDPC codes increases linearly with N, undetected errors are relatively uncommon, unless the block length is short (less than a few hundred bits) or the code rate is high. Nonetheless, it is was shown in [11] that it is computationally complex to directly design codes having a high $d_{\min}$.

An indirect way of increasing $d_{\min}$ is to increase the girth of the bipartite graph. However, rather than using the aforementioned conventional girth conditioning techniques, which only focus on increasing the shortest cycle length, Tian et al. [11] revealed that it is also important to consider the specific connectivity of the cycles with the other parts of the bipartite graph, rather than only the length of the cycles. This is because not all cycles are equally harmful. Those that are well connected to the rest of the graph are acceptable whereas poorly connected long cycles may be more detrimental. This technique, which is referred to as *cycle conditioning*—as opposed to girth conditioning—requires the identification of stopping sets, which are a particular group of variable nodes that is connected to a group of neighboring parity-check nodes more than once. One example of a stopping set exemplified in Figure 1(b) is constituted by the variable nodes v_2, v_3, and v_6 because all the neighboring parity-check nodes c_1, c_2, and c_3 are connected to this variable node set twice. By means of avoiding small stopping sets, the technique of Tian et al. [11] succeeded in significantly reducing the error floor of irregular LDPC codes whilst only suffering from a slight BER degradation in the waterfall region.

The trapping sets have a direct influence on the error floor of LDPC codes. A trapping set (a, b) refers to that particular set of a variable nodes in the associated bipartite graph that induces a subgraph that contains b odd-degree and an arbitrary number of even-degree parity-check nodes. For example, a trapping set $(5, 4)$ can be observed in the bipartite graph of Figure 1(b) constituted by the variable nodes v_1, v_2, v_3, v_4, and v_6 and the parity-check nodes c_1, c_2, c_3, and c_4. When the values of a and b are relatively small, the variable nodes in the trapping set are not well connected to the rest of the graph, and therefore, the corresponding bits have a weak protection. In some research literature, trapping sets are described as *near code words* because when the parameters a and b are relatively small, an incorrectly decoded code word may only be slightly different from the transmitted code word. We emphasize that the errors resulting from the presence of small trapping as well as stopping sets are detected by the decoder; i.e., the decoder will be aware that the no legitimate code word was found owing to having some unsatisfied (nonzero-valued) parity-check nodes after the affordable maximum number of decoding iterations. The problems that arise from the presence of trapping sets/near-code words can be mitigated by either altering the PCM (without changing the actual code) or modifying the decoder [12].

Structured Versus Pseudorandom Parity-Check Matrices

Another dilemma that code designers face is that of having a pseudorandom or a structured PCM. The pseudorandom allocation of the logical one values in the PCM was considered to be an important feature in the LDPC design since it was demonstrated in [13] that these codes exhibit excellent error correction capabilities. On the other hand, structured LDPC code constructions [7] impose additional constraints on their PCM, and hence, they may potentially suffer from some BER/BLER performance degradation. In respect of Figure 2 and our discussions at the beginning of this section, where we have attempted to show the intricate tradeoffs among the block length, girth, minimum distance, and PCM weights, it is also worth mentioning that some structured constructions may further constrain the aforementioned attributes. For example, the minimum distance of the structured construction proposed in [14] does not increase linearly with the block length as in other pseudorandom constructions, instead it is bounded by $(\gamma + 1)!$, where $(\cdot)!$ denotes the factorial operator. This implies that the only way to increase $d_{\min}$ is to increase the column weight, which comes at the expense of reducing the associated girth and increasing the complexity. (The relationship between the PCM weights and decoding complexity will be explained in the "Decoder Characteristics" section.) Furthermore, it was also demonstrated in [14] that the maximum girth achieved by these constructions is at most 12. There are other designs, such as [15] (amongst others), that overcome these limitations. Given their practical encoder and decoder implementations, it is not surprising that structured codes managed to make their way into a number of standards [e.g., digital video broadcast by satellite (DVB-S2), DVB-T2, IEEE WiMAX and IEEE 802.11n [19]–[22]].

Encoder Characteristics

The standard encoding operation requires the calculation of the GM $\mathbf{G}$ from the PCM $\mathbf{H}$ using Gaussian elimination and, finally, multiplying the input message sequence with the calculated $\mathbf{G}$. An important aspect to point out in this context is that whilst the PCM of an LDPC code is sparse, its GM $\mathbf{G}$ is not. For this reason, this encoding process becomes significantly more complex than that of the competing turbo codes, considering that the latter have a low encoding complexity—which increases linearly with the block length. Several complexity reduction measures have been proposed to address this issue. One of the frequently used techniques rearranges the PCM in the approximate lower triangular form, which reduces the encoding complexity to $(N + \underline{g}^2)$, where $\underline{g}$ is referred to as the *gap* [16]. (For example, a regular LDPC code associated with a PCM having $\gamma = 3$ and $\rho = 6$ has a normalized gap of $\underline{g}' = \underline{g}/N = 0.017$ [16]. Note that this definition of the gap assumes normalization to N. So, for this specific regular code with $N = 10{,}000$ b, a gap of approximately 170 will result.) Nevertheless, the matrices used for encoding do not possess any strict internal structure, and therefore, the location of all the logical ones in the matrices must be enumerated.

The family of structured codes, which were introduced in the "Structured versus Pseudorandom Parity-Check Matrices" section, also benefits from efficient encoder implementations. Typically, such structured codes are either cyclic or quasi-cyclic (QC) and thus possess a PCM and a GM that is composed from a number of circulant matrices. (A QC code is defined as a code in which any cyclic shift of a constituent code word by x number of bits is also a code word. For a cyclic code, we have $x = 1$. We define a circulant matrix as a binary-valued square matrix where each row is constructed from a single right cyclic shift of the previous row, and the first row is obtained by a single right cyclic shift of the last row.) This distinctive characteristic provides two main benefits. First, it will significantly reduce the amount of memory required to store the GM as well as simplify the memory address generation (MAG) since only the first row of each circulant matrix will be stored and memory shifts corresponding to the cyclic/QC structure will be used to address the messages. Second, the encoding can be carried out by means of linear shift registers—thus simplifying the required hardware—whilst attaining a linearly increasing block-length-dependent encoding complexity. For a number of QC LDPC codes, such as the codes proposed for the 802.11n and the 802.16e [22] standards, encoding may be realized using the PCM through back substitution, thus eliminating the need to store the GM.

Decoder Characteristics

The first challenge to be tackled when implementing the SPA in hardware is that of effectively managing the exchange of extrinsic messages between the check and variable nodes. In this regard, there is always a tradeoff between choosing a parallel or a serial implementation, whereby the former offers a higher throughput at the expense of an increase in the required silicon area whereas the latter requires a smaller chip area but attains slower decoding speeds. The inherent suitability of the SPA to

TABLE 1 The effect on the BER/BLER performance or the girth, minimum distance, and the complexity if certain adjustments are made.

	Adjustments				
Effect on the	*Regular PCM*	*Irregular PCM*	*Increasing N*	*Increasing g*	*Increasing γ*
Waterfall performance	Generally worse than irregular counterparts	Generally better than regular counterparts	Improves	Improves	Improves (if code is carefully designed)
Error floor performance	Generally better than irregular counterparts	Generally worse than regular counterparts[1]	Improves	Improves[2]	Improves (if code is carefully designed)
Girth	—	—	Increases, except for some QC codes (e.g., [14])	—	Decreases
Minimum distance	—	—	Increases, except for some QC codes (e.g., [14])	Increases	Increases (if code is carefully designed)
Computational complexity[3]	—	—	Increases	Decreases (fewer iterations required)	Increases
Hardware complexity	Generally easier to implement than irregular counterparts	Generally more difficult to implement that regular counterparts	Increases	—	Increases

— Stands for inconclusive/not applicable/no change.
[1] Much of the contemporary research is focused on improving the error floor of irregular LDPC codes (cf. [8]).
[2] Whilst increasing the girth is certainly beneficial, the error floor performance is more dependent on stopping sets and trapping sets (cf. the ''Code-Construction Attributes Affecting the Error Floor'' section).
[3] This is dependent on the girth and the total number of edges of the underlying graph.

parallel architectures enables LDPC codes to achieve a higher degree of parallelism. Some constructions, such as the structured or QC codes, possess a PCM that allows for the simultaneous update of a large number of parity-check nodes. However, one must also be aware that the BER/BLER performance may be degraded if the code's structure is severely restricted so as to increase the degree of parallelism. Other codes, such as those based on protographs [17], [18], have a construction that can favor both serial as well as parallel architectures (referred to as *semiparallel*) and thus strive to achieve a better compromise between the decoding speed versus area tradeoff.

Our second challenge is constituted by the memory requirements, which are dependent on both the type of information to be stored and the number of bits used to represent them (i.e., the numerical precision) as well as the block length. A considerable amount of memory is needed to store the information messages that are exchanged between the nodes located at the opposite sides of the bipartite graph at every iteration of the decoding process. These memory requirements can be significantly reduced using suboptimum algorithms, such as the min–sum algorithm, or by passing reduced-precision messages, although this might slightly degrade the achievable BER/BLER performance. We also have to store the code's description, which defines the specific interconnections of the edges between the nodes. Whilst it can never be denied that pseudorandom codes, such as the classic regular MacKay LDPC codes [13] and conditioned irregular codes [11], exhibit an excellent BER/BLER performance, the random selection of the connections between their parity-check and variable nodes makes it, particularly, hard to create a convenient description for the code. Hence, their implementation often results in either inflexible hard-wired interconnections or large inefficient LUTs. On the other hand, structured codes benefit from simplified descriptions as well as from facilitating efficient read and write operations from/to memory.

Furthermore, we must not ignore the complexity imposed by the variable and parity-check nodes, when computing the messages to be exchanged. Specifically, the number of additions processed by each variable node in the graph is twice the PCM column weight. The number of additions and multiplications required by the parity-check nodes is a linear function of the PCM row weight, where the specific function depends on whether we employ optimal or suboptimal decoding algorithms. We have demonstrated in the ''Code-Construction Attributes Affecting the Error Floor'' section that increasing the PCM weights increases the minimum distance of the code, but it is now also clear that this consequently increases the computational complexity. However, this decoding complexity can be reduced by increasing the girth, since this will decrease the number of iterations required to arrive at a code word (please refer to the ''The Girth of the Underlying Graph'' section).

Conclusions

In this article, we have characterized the interplay of parameters influencing the design of LDPC codes as transpires from Table 1. Naturally, the BER/BLER performance, the construction attributes, as well as the complexity are closely coupled, and beneficial designs maintain a good BER/BLER performance when transmitting over both AWGN and Rayleigh channels, whilst still having hardware friendly implementations. The ideas presented may also be extended to the design of arbitrarily wireless transceivers using diverse modulation schemes, which constitutes our future research objective.

Acknowledgments

The financial support of both the Engineering and Physical Sciences Research Council (EPSRC), United Kingdom, and the EU under the auspices of the Optimix project is gratefully acknowledged.

Author Information

Nicholas Bonello (nb06r@ecs.soton.ac.uk) received his B.Eng. (Hons.) degree in electrical engineering from the University of Malta in 2004, his M.Sc. degree in radio frequency communications systems from the University of Southampton, United Kingdom, in 2006, and his Ph.D. degree in wireless communications from the University of Southampton, United Kingdom, in 2009. His research interests include fixed-rate as well as rateless error correction techniques, signal processing, and statistics.

Sheng Chen (sqc@ecs.soton.ac.uk) received his B.Eng. degree from the East China Petroleum Institute, Dongying, China, in 1982, and his Ph.D. degree from City University, London, in 1986, both in control engineering. Since 1999, he has been with the School of Electronics and Computer Science, University of Southampton, United Kingdom. In 2005, he received a D.Sc. degree from the University of Southampton, United Kingdom. He has held research and academic appointments at the universities of Sheffield, Edinburgh and Portsmouth in the United Kingdom. He has published more than 280 research papers. He is on the list of highly cited researchers in the engineering category. His research works include adaptive signal processing, wireless communications, modeling and identification of nonlinear systems, neural network and machine learning, finite-precision digital controller design, evolutionary computation methods, and optimization.

Lajos Hanzo (lh@ecs.soton.ac.uk) received his degree in electronics in 1976 and his doctorate degree in 1983. Since 1986, he has been with the School of Electronics and Computer Science, University of Southampton, United Kingdom, where he was the chair in telecommunications. From 2008, he was the editor-in-chief of IEEE Press, and from 2009, a chair professor at Tsinghua University, Beijing. In 2009, he was awarded the honorary doctorate Doctor Honaris Causa by the Technical University of Budapest. In 2012, he became one of four European Signal Processing Association fellows. He has held various research and academic posts in Hungary, Germany, and the United Kingdom. He has supervised more than 70 Ph.D. students, coauthored 20 Wiley/IEEE Press books on mobile radio communications, published more than 1,200 research entries at IEEE *Xplore*, acted both as Technical Program Committee and general chair of IEEE conferences, presented keynote lecture, and has been awarded a number of distinctions. Currently, he is directing an academic research team that works on a range of research projects in the field of wireless multimedia communications sponsored by industry, EPSRC, United Kingdom, the European IST Programme, and the Mobile Virtual Centre of Excellence, United Kingdom. He is an enthusiastic supporter of industrial and academic liaison, and he offers a range of industrial courses. He is also a governor of the IEEE Vehicular Technology Society.

References

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 1948.
[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE Int. Conf. Communications, Geneva Tech. Program*, Geneva, Switzerland, May 23–26, 1993, vol. 2. pp. 1064–1070.
[3] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. 45, pp. 21–28, Jan. 1962.
[4] T. J. Richardson, "Error floors of LDPC codes," in *Proc. 41st Annu. Allerton Conf. Communications, Control and Computing*, Urbana-Champaign, IL, Oct. 2003, pp. 1426–1435.
[5] T. J. Richardson and R. L. Urbanke, *Modern Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, Mar. 2008.
[6] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, pp. 1727–1737, Oct. 2001.
[7] J. M. F. Moura, J. Lu, and H. Zhang, "Structured low-density parity-check codes," *IEEE Signal Processing Mag.*, vol. 21, pp. 42–55, Jan. 2004.
[8] P. O. Vontobel. Tech. Rep. [Online]. Available: http://www.hpl.hp.com/personal/Pascal_Vontobel/pseudocodewords/
[9] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, pp. 533–547, Sept. 1981.
[10] T. Etzion, A. Trachtenberg, and A. Vardy, "Which codes have cycle-free Tanner graphs?" *IEEE Trans. Inform. Theory*, vol. 45, pp. 2173–2181, Sept. 1999.
[11] T. Tian, C. R. Jones, J. D. Villasenor, and R. D. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Trans. Commun.*, vol. 52, pp. 1242–1247, Aug. 2004.
[12] Y. Han and W. Ryan, "Low-floor decoders for LDPC codes," *IEEE Trans. Commu.*, vol. 57, pp. 1663–1673, June 2009.
[13] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Mar. 1999.
[14] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inform. Theory*, vol. 50, pp. 1788–1793, Aug. 2004.
[15] M. Esmaeili and M. Gholami, "Structured quasi-cyclic LDPC codes with girth 18 and column-weight $J \geq 3$," *Int. J. Electron. Commun. (AEÜ)*, vol. 64, pp. 202–217, Mar. 2010.
[16] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity check codes," *IEEE Trans. Commun.*, vol. 47, pp. 808–821, Feb. 2001.
[17] K. Andrews, S. Dolinar, D. Divsalar, and J. Thorpe. (2004, Nov.). Design of low-density parity-check codes LDPC codes for deep-space applications, IPN Progress Rep. 42-159, Jet Propulsion Lab. [Online]. Available: http://ipnpr.jpl.nasa.gov/progress_report/42-159/159K.pdf
[18] N. Bonello, S. Chen, and L. Hanzo, "Construction of regular quasi-cyclic protograph LDPC codes based on Vandermonde matrices," *IEEE Trans. Veh. Technol.*, vol. 57, pp. 2583–2588, July 2008.
[19] *IEEE Standard for Local and Metropolitan Area Networks—Part 16: Air Interface for Fixed Broadband Wireless Access Systems (Revision of 802.16-2001)*, IEEE Standard 802.16d-2004.
[20] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-Speed Physical Layer in the 5 GHz Band*, IEEE Standard 802.11a 1999.
[21] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11g, 2003.
[22] *Draft IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications—Amendment 4: Enhancements for Higher Throughput*, IEEE Standard 802.11n-d4-2008. ***VT***