A Fast Hardware Implementation of an LDPC Codec For Packet Erasure Channels

Chad Cole
ATG, LLC

***Abstract*—Erasure codes are a form of Forward Error Correction (FEC) that is used to reconstruct messages that are sent over an erasure channel. The erasure channel is a popular way to model a lossy packet network. Erasure codes are typically employed at a higher layer in the communication stack, such as at the application layer. Erasure codes are not nearly as ubiquitous in communications systems as the error correction codes used at the physical layer. However, there are certain situations, such as when sending a real-time, high-data rate stream of digital intermediate frequency I/Q samples over a potentially lossy packet network, where erasure coding can offer great benefits. This paper compares two popular types of well-known erasure codes - LDPC codes and Reed Solomon codes - and recommends an LDPC code for use in the nascent DIFI [1] standard, because while both codes offer similar erasure correction performance, the LDPC code has a simple parallel hardware implementation that provides multi-Gbps throughput.**

***Index Terms*—Erasure Codes, LDPC Codes, Reed Solomon**

I. Introduction

Erasure codes are a form of Forward Error Correction (FEC) that are used to reconstruct messages that are sent over an erasure channel [2], such as a lossy packet network. The erasure channel has the property in which message symbols either arrive at the destination correctly or are 'erased' and the receiver knows the symbol is missing. Forward error correction (FEC) erasure codes have been a popular research topic at various times over the last few decades; however, to the author's knowledge, they have never made much of an impact in real world communication systems. The reasons for this lack of interest in practical communication systems are varied. One common competing higher-layer method to avoid packet loss is to employ some sort of retransmission scheme, such as what is done in the transport layer with Transmission Control Protocol (TCP) [3]. To ensure reliable delivery in TCP, packets that are missing at the destination are simply requested again from the source. As long as there is a suitable feedback channel with a relatively short round-trip time between source and destination, then a retransmission scheme works well.

However, there are situations where retransmission is either not feasible or unwieldy. The obvious case where FEC holds a big advantage over the simple retransmission scheme is when there is no feedback channel in which to request missing packets. Another important scenario where FEC is preferred over retransmission is in a one-to-many or broadcast scenario [4]. The scenario of sending high-speed real-time digital IF data from an edge collection node to a cloud processing node over the internet, as proposed in the DIFI standard, is a good application for packet FEC.

This research was funded internally by ATG, LLC.

There are many codes that can be used as erasure codes. Codes that are good candidates for physical layer error correction - because they may have good codeword distance properties or efficient encoding and decoding algorithms - are also typically good candidates as codes for the binary erasure channel. The classic example of a good erasure code is the class of Reed Solomon (RS) codes [5], which have the property of being Maximum Distance Separable (MDS). An MDS code always has a minimum code distance of $n-k+1$, where k is the number of message symbols being sent and n (where $n > k$) is the coded message symbol length with added redundancy. The MDS codes have the best distance property that a code can possess, as these codes meet the Singleton Bound [6].

Low-Density Parity-Check (LDPC) codes are a popular linear block coding technique that has been around for over 50 years [7]. While they have been used as physical layer error correcting codes for many years, it is only more recently that they have been considered as candidates for codes over an erasure channel. LDPC codes are characterized by a sparse parity check matrix, $\mathbf{H}$. An LDPC code has k source symbols and $n-k$ parity symbols, for a total of n symbols in a codeword. These two parameters are usually written as an (n, k) pair, and the code rate is defined as $r = k/n$, which describes the amount of redundancy in the code. LDPC codes are popular because there is a simple 'message passing algorithm' (MPA) which can be utilized to decode the noisy received codewords. The MPA uses the sparsity of $\mathbf{H}$ to iteratively pass messages about bit and parity check probabilities across all bits in the code. While the MPA is suboptimal, it has good performance for well designed LDPC codes and is a great choice for a practical codec implementation. The efficient encoding and decoding algorithms available for LDPC codes allow for code designs to employ large block lengths and arbitrary code rates.

Another popular class of erasure codes is Raptor codes [8], which are similar to LDPC codes and share many of their strengths and weaknesses. For larger block lengths and over a higher order Galois field, Raptor codes have similar performance to RS codes, but have lower encoding and decoding complexity. A good performance comparison between LDPC codes and Raptor codes was illustrated in [9], where the conclusion was made that suitably constructed LDPC codes using a maximum likelihood (ML) decoder had similar performance to Raptor codes, but without the Intellectual

Property concerns.

A packet erasure channel constitutes a burst erasure channel relative to a transmitted binary stream. Use of conventional FEC codes can combat this burstiness via interleaving, as shown in Figure 1. There we assume packets of length S bits are sent over the binary channel, and we instantiate S encoders and decoders in parallel. Thus a single packet erasure presents single bit erasures to each decoder; powerful codes are capable of repairing multiple packet erasures with this technique.

This approach simultaneously offers a simple parallel hardware architecture for encoding and decoding. This provides throughput enhancements, at the cost of hardware resources.

In the following, we study the use of moderate length ($n \approx 2000$) LDPC codes capable of fast encoding and erasure processing. S is selected to be 8192, corresponding to a packet size of roughly 1 kilobyte.

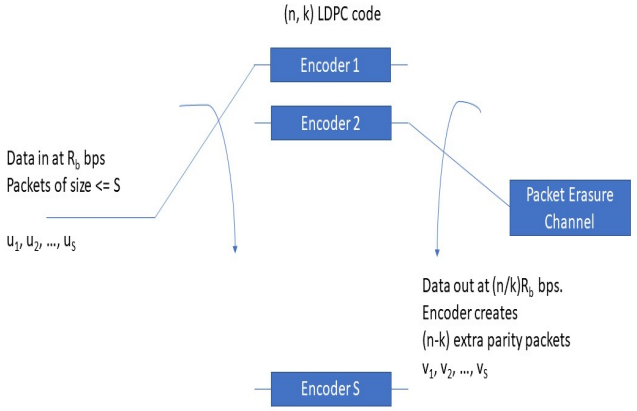

Fig. 1. Packet Erasure System Model

Section II introduces an LDPC code construction that has good decoding performance at higher SNR, or, equivalently stated, at low raw Packet Erasure Rates (PER). Next, Section III explains some of the practical considerations in implementing RS and LDPC codecs. Section IV provides simulation results showing the performance of RS versus LDPC codes for several block lengths and code rates. Section V describes the OpenCL design and shows some high SNR performance results of the LDPC code that were made possible because of the fast hardware implementation of the decoder. The final section summarizes the paper.

II. Random Code Construction With Girth Eight

The LDPC codes created for this paper were designed with three major concerns in mind:

- First, the codes need to be systematic.
- Next, the codes should allow for a simple implementation of both the encoder and the decoder.
- Finally, the codes must have a relatively low error floor, since it is in this low PER region where these codes will typically be utilized.

There are various ways to design LDPC codes to ensure certain desirable conditions are met for encoding and decoding. One popular constraint for erasure codes is to create a 'systematic' code where the first k symbols of a codeword are the original source symbols that are being transmitted. A systematic code has the nice property that if no erasures occur on the original k source symbols, then no effort must be expended in decoding the received codeword. The $\mathbf{H}$ matrix structure in Figure 2 holds this systematic encoding property. The 'lower-triangular' format that allows systematic encoding is evident in the right half of the matrix in the figure.

To illustrate the systematic encoding process, consider the simple example $\mathbf{H}$ matrix, though not low-density in nature, that is shown in Equation 2. The codeword $\mathbf{v}$ in Equation 1, consists of labels to show which symbols are associated with source versus parity symbols. The first $k = 3$ symbols are source symbols and the last $(n - k) = 3$ symbols are parity symbols. The parity check matrix in Equation 2 forms a $(6, 3)$ rate $\frac{1}{2}$ code in triangular format. Remember that in a linear block code, for a vector to be in the code, the constraint $\mathbf{vH^T} = \mathbf{0}$ must be met. To generate the first parity symbol, each of the source symbols at the positions of non-zero columns in the first row of $\mathbf{H}$ must be XORed together to form p_1. For this first row, s_2 is the only non-zero column, so $p_1 = s_2$. Setting p_1 in this way ensures that the first position of $\mathbf{vH^T} = \mathbf{0}$. In a similar fashion, to ensure that the second position of $\mathbf{vH^T} = \mathbf{0}$, p_2 can be found to equal $s_1 XOR s_3$. Notice that the calculation of p_2 does not affect our previously calculated p_1 because there is a '0' in the p_2 column for the first row. Finally, $p_3 = s_3 XOR p_1$. Again, due to the triangle format of $\mathbf{H}$, the rows above in column p_3 are zero and neither of the previously calculated p_1 and p_2 are affected by p_3.

$$\mathbf{v} = \begin{pmatrix} s_1 & s_2 & s_3 & p_1 & p_2 & p_3 \end{pmatrix} \tag{1}$$

$$\mathbf{H} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \tag{2}$$

Most LDPC code constructions satisfy the easy implementation constraint above, but to further ease hardware implementation, a regular parity check degree was utilized. Since the message passing decoding algorithm used on the erasure channel performs its operations on the rows of $\mathbf{H}$, in other words the parity check constraints of the code, it makes a parallel implementation more efficient if the check node degree (the number of ones in each row of $\mathbf{H}$) is about the same, i.e. 'regular.'

To help achieve the third bullet above, a low error floor, the codes were constructed in such a way that the Tanner graph [10] of the code contains no 4-cycles or 6-cycles and thus the graph is guaranteed to possess a girth of at least eight. As the girth of a code increases, so does the minimum distance of the code [11], as well as the small weight stopping set profile which allows the MPA to perform closer to an ML decoder [12].

The specific code construction method utilized in this work is similar to the 'Bit-filling' algorithm of [13]. However, since the algorithm in this work was independently conceived, it does have a few differences. One difference is that the row or column of $\mathbf{H}$ to which one's are added is based on a probabilistic rule in which rows and columns are given a probability to be used based on the inverse of their current weight in $\mathbf{H}$. For example, these codes were constructed in a row-wise manner, so for each row, the column chosen for an additional one in $\mathbf{H}$ would be chosen with a probability that is uniformly inversely proportional to it's current weight in $\mathbf{H}$. As each one is added to $\mathbf{H}$, the girth eight constraint is checked and if it fails, then a new one position is randomly chosen among the set of columns of $\mathbf{H}$ still under consideration. The number of one's in each row of $\mathbf{H}$ is kept regular and the degree distribution of the columns of $\mathbf{H}$ can be constrained if desired.

III. Implementation Comparison

In this section we will compare the difficulty of encoding and decoding RS and LDPC codes over the erasure channel. First, a block of k source symbols, $\mathbf{u}$, is encoded into a block of n coded symbols, $\mathbf{v}$, by the linear matrix operation of $\mathbf{v} = \mathbf{uG}$. $\mathbf{G}$ is the $k \times n$ generator matrix of the linear block code. Next, $\mathbf{v}$ is sent over an erasure channel yielding $\mathbf{y} = \mathbf{v} + \mathbf{e}$ at the receiver, where $\mathbf{e}$ is the vector of erasures, where if e_i is 1 then an erasure occurs at the i^{th} symbol in $\mathbf{v}$. The goal of a decoder is to try to reconstruct $\mathbf{u}$ from $\mathbf{y}$.

A. Reed Solomon Codes

RS codes are typically implemented in systematic form and thus have a generator matrix of the following form:

$\mathbf{G} = \begin{pmatrix} \mathbf{I} & \mathbf{P} \end{pmatrix}$

$\mathbf{I}$ is a $k \times k$ identity matrix and $\mathbf{P}$ is a $k \times (n-k)$ dense matrix over a higher-order Galois field, such as the $GF(2^8)$ used in the RS code in the TIA-5041 standard [14]. The implementation of RS codecs presents two major difficulties. First, all matrix operations must be performed over the Galois field order of the RS code, which gets unwieldy for large field sizes. For smaller field sizes, such as 256 and under, a simple lookup table approach can be used for arithmetic operations. Second, the generator matrix is *dense*, which means that the matrix multiplication necessary for encoding will require $\mathcal{O}(k(n-k))$ operations over the Galois field. Decoding linear codes over an erasure channel involves finding the solution to $\mathbf{u} = \mathbf{y_c}{\mathbf{G_c}}^{-1}$, where $\mathbf{G_c}$ is the subset of columns of $\mathbf{G}$ that were received (i.e. not erased). These correctly received positions of $\mathbf{y}$ are designated by $\mathbf{y_c}$. Since the first k symbols are sent without processing, any of these that are received will lower the decoding complexity. Call the number of systematic symbols received in $\mathbf{y_c}$ as N_{sys}. The Gaussian elimination process to obtain $\mathbf{G_c^{-1}}$ has a complexity of $\mathcal{O}(k(k-N_{sys})^2)$. The RS codes have the property that any k columns of $\mathbf{G}$ are guaranteed to be full rank and thus lead to a unique solution for $\mathbf{u}$. Similarly to the system model of Figure 1 for binary LDPC codes, the RS codec over GF(256) operates on sequential groups of 8 bits in parallel. For example, if $S = 8192$, then there would be $8192/8 = 1024$ RS codecs operating in parallel.

B. LDPC Codes

The encoding complexity of these systematic lower-triangular form LDPC codes is $\mathcal{O}((n-k)w_r)$, where w_r is the row weight of $\mathbf{H}$. This can be seen from following the simple encoding procedure described with an example in Section II. The decoding principle for LDPC codes sent over the erasure channel is the same as for the RS codes described above; however, the decoding is explained from a $\mathbf{H}$ perspective instead of a $\mathbf{G}$ perspective, since LDPC codes are characterized by their $\mathbf{H}$. It is true that for any linear block code, $\mathbf{Hv} = \mathbf{0}$. Thus, over the binary field, $\mathbf{H_c v_c} = \mathbf{H_e v_e}$, where $\mathbf{H_c}$ and $\mathbf{v_c}$ are the subsets of columns of $\mathbf{H}$ and $\mathbf{v}$ respectively where an erasure does not occur. Similarly, $\mathbf{H_e}$ and $\mathbf{v_e}$ are the column subsets where an erasure *does* occur. To find any erased systematic symbols from $\mathbf{v_e}$, $\mathbf{H_e^{-1}}$ can be computed if it is full rank. An ML decoder would perform this inverse matrix operation, which requires $\mathcal{O}((n-k)^3)$ complexity for dense matrices, but for sparse $\mathbf{H}$ requires fewer computations. The LDPC message passing algorithm gives a suboptimal approximation to this matrix inverse at a slight cost of erasure correction performance, but with a computational complexity of only $\mathcal{O}((n-k)w_r N_{iter})$, where N_{iter} is the number of message passing iterations, which is typically around 10-20. Thus, for a long code where $(n-k) >> w_r$ and $(n-k) >> N_{iter}$, this leads to a decoding complexity that is linear in $n-k$.

IV. Performance Comparison

Since RS codes have the MDS property, their erasure correction performance cannot be beat for a given block size and code rate. The block size of RS codes must be less than their field size, so larger block sizes necessitate operations over larger Galois fields. Because of this requirement, it is not practical for the RS block size to be much larger than 511. Also, the $\mathcal{O}(k(k-N_{sys})^2)$ complexity of the RS decoding is a distinct disadvantage for larger k. However, if a larger latency can be tolerated, it is fair to compare a longer block size LDPC code against a smaller block size RS code. For example, a $(2040, 1530)$ LDPC code with $r = 3/4$ could be compared against a sequence of eight $(255, 192)$ RS codes. A carefully designed irregular binary LDPC code can outperform the RS code in this case. As the block size of the LDPC code increases, this performance advantage over the RS code will also grow. The reason a RS code of size $(255, 192)$ over GF(256) is chosen in this paper is because the TIA-5041 specification [14] describes it as a candidate code for application layer FEC and thus serves as a good performance comparison point against the LDPC codes proposed herein.

The degree distribution of the LDPC code has a large impact on both the decoding threshold of the code and the error floor behavior of the code [15]. The decoding threshold is the signal-to-noise (SNR) level at which the coded

block error rate drops precipitously. The error floor region is the high-SNR region where a small number of minimum distance codewords or codeword-like graph structures such as stopping sets or trapping sets will confuse an iterative, non-ML decoder [16]. The variable node, $V(x)$, and check node, $C(x)$, degree distributions are typically given by the equations $V(x) = \sum_{i=1}^{d_v} V_i x^i$ and $C(x) = \sum_{i=1}^{d_c} C_i x^i$, where d_v and d_c are the maximum variable and check node degrees, respectively. Two LDPC codes were analyzed in depth for this paper. Both are of moderate length and represent two code rates that are useful in different loss scenarios. The $r = 1/2$ codes such as the $(2000, 1000)$ code used in this paper have an almost regular degree profile given by the degree polynomials of roughly:

$$V(x) = 0.0005x + 0.005x^2 + 0.9895x^3 + 0.005x^4$$
$$C(x) = 0.997x^6 + 0.003x^7$$

The $0.0005x$ term in $V(x)$ represents the final column of $\mathbf{H}$ where the triangle ends. The lower-triangular form of $\mathbf{H}$ used in this work is similar to the double identity parity structure in the $\mathbf{H}$ of irregular repeat-accumulate codes [17], but it does not suffer from the constraint that all of the $n-k$ parity columns of $\mathbf{H}$ are degree two. A surfeit of degree-two variable nodes leads to small stopping sets and codewords, and thus a relatively high error floor.

In this paper we also look at two $r = 3/4$ codes of $k = 3060$ and $k = 1530$, which have degree polynomials of roughly:

$$V(x) = 0.0002x + 0.0324x^2 + 0.3311x^3 + 0.4995x^4 + 0.1279x^5 + 0.0081x^6 + 0.0007x^7$$
$$C(x) = 0.99x^{15} + 0.01x^{16}$$

A lower error floor can be obtained by going to a non-binary LDPC code [18]. This would involve simply taking a binary $\mathbf{H}$ matrix and for each position where there is currently a 1, randomly substitute in a non-zero value from GF(m). However, non-binary codes cannot be decoded by the simple MPA and instead an ML decoding procedure must be employed. The ML decoding of non-binary LDPC codes suffers from only one of the two drawbacks of the RS decoding described above in section III-A. The operations on the matrix to find the inverse of the received symbol matrix must be performed over the Galois field that is used in the LDPC code design. However, the second drawback of RS codes, namely the density of the received matrix that must be inverted, is sparse in the LDPC code. Applying ML decoding to binary LDPC codes after the MPA has failed is another way to increase the erasure correction performance of these codes.

The LDPC codes used in this work, although binary, still have a relatively good error floor performance due to their careful construction that avoids small cycles in the Tanner graph [11].

The block error performance of a $(2040, 1530)$ code is compared to a $(255, 192)$ RS code in Figure 3. In this Monte Carlo simulation, 1000000 random code blocks are sent through the binary erasure channel with the raw (i.e. uncoded) PER given by the value in the horizontal axis. Since both of these codes have $r = 3/4$, as the raw PER gets close to 25%, the coded block error rate approaches one for both codes. Note in the figure that the LDPC code with the simple MPA outperforms the RS code at PER below 18%. The full range of PER's tested in this scenario is between 14-22%. The curve marked 'LDPC MP/Maximum Likelihood Decoder' outperforms the RS code at all PER's. For this case, the decoder starts with the simple MPA and after 10 iterations, if the MPA hasn't corrected all of the erasures, then the remaining erasures are removed by using a Gaussian elimination process to find the inverse of the submatrix formed by the columns of $\mathbf{H}$ where there are remaining erasures after the MPA. Although the ML decoder is more complicated to implement than the MPA, Figure 3 shows that it offers erasure correction performance benefits over the simple MPA applied to LDPC codes. In this work, the ML decoder was implemented in Matlab and only the simple MPA was converted to OpenCL.

One seemingly straightforward argument in favor of RS codes is that the latency required for longer LDPC codes is proportionately larger compared to the shorter RS codes. For example, a $(2000, 1000)$ LDPC code will require 8 times as long to collect and encode/decode symbols as a $(250, 125)$ RS code. This argument is true when the packet erasures are completely random and independent. However, for a bursty channel, which is common in packet networks where erasure codes are commonly used, the performance of short RS codes will not stand up to longer LDPC codes unless an interleaver is used to combine symbols across multiple RS code blocks. Interleaving across multiple RS codes, to give them the same time diversity that the long LDPC codes enjoy, will introduce more latency, thus negating any argument that short RS codes have an advantage by providing a lower latency.

V. OpenCL Implementation of LDPC Codec

The simulations for this paper were first performed in Matlab, and it took considerable time to obtain LDPC code performance results at low PER. OpenCL is a high-level hardware design language that resembles C code. To speed up simulation time at lower PER, an implementation of the LDPC codec was done in OpenCL. While it is convenient to write code in a higher-level language such as OpenCL, the code generated from the OpenCL compiler is typically not as efficient as manually-designed hardware design language (HDL) code. However, due to the inherent simplicity of the LDPC codec design, even the OpenCL implementation of the codec produced a hardware design that allowed multi-Gbps throughput for both the encoder and decoder.

The simple systematic encoding scheme and message passing decoding algorithm of LDPC erasure codes is a natural candidate for hardware implementation. The heart of both the encoder and decoder is simple XOR accumulations that are highly performant in a parallel hardware architecture. The Intel Quartus Prime Pro 21.2 toolset was used to compile the OpenCL code into a bit file, which was run on a Stratix 10 FPGA.

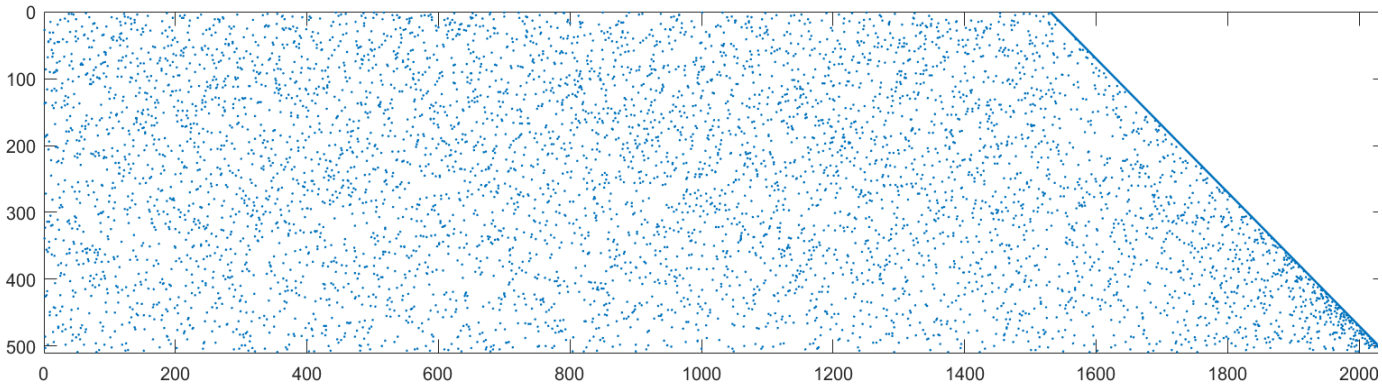

Fig. 2. Triangular Form (2040, 1530) LDPC Code Parity Check Matrix

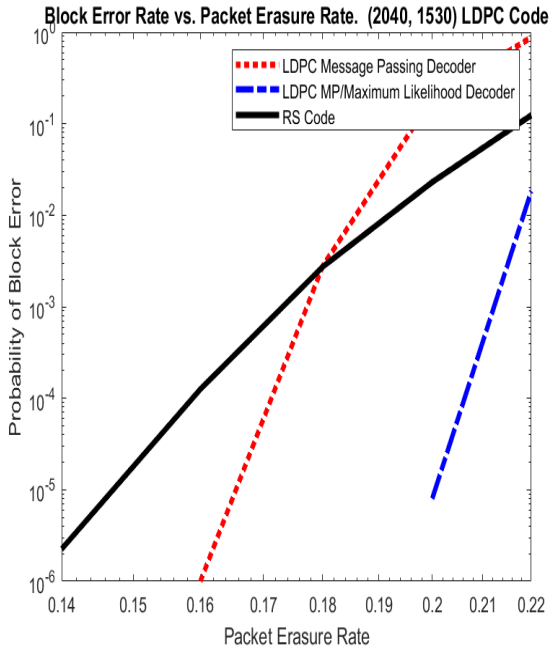

Fig. 3.

The code symbol size in the OpenCL design was specified to be an array of 128 64-bit long integers. The OpenCL compiler was able to efficiently design the parallel operations performed across the $S = (128)(64) = 8192$ bits in each symbol. However, the compiler was not able to automatically find efficient ways to parallelize the $n - k$ parity check operations performed in each decoding iteration, such as by running some of the parity check constraints in parallel. So, a manual partitioning of the parity check constraint equations can be performed to increase throughput. For example, a simple way to speed up the decoder by a factor of two is to perform the MPA on the first $(n - k)/2$ parity constraints in parallel with another MPA operating on the other half of the parity check constraints. The results from the two instances of MPA's are combined by taking the separate instances of the codeword that the two MPA's were operating on and performing a union of the non-erasure positions in the two codewords. If either of the two codewords has a non-erasure value in a position, then that is used as the correct value. If both instances of the codeword are erasures, then so is the combined final codeword in that position. This is a simple way to trade additional hardware resources for an almost doubling of throughput. This method can scale up if more hardware resources are available for more parallelization.

A simple throughput test showed speeds of 36.3 Gbps for the $(2040, 1530)$ code at 14.06% raw PER as seen in the final row of Table I. The throughput in information bits per second is calculated as $(SN_T k)/T_{sim}$. The throughput numbers will be highly dependent on the code rate and the raw PER. A higher code rate will lead to higher throughput, because there are fewer check node constraints per information symbol. A lower raw PER leads to higher throughput because it requires fewer MPA iterations to reach a valid codeword on average, because there is an early stopping criterion built into the decoder. If there are no erasures present in the codeblock after a decoding iteration, then the decoder stops and returns the codeblock that the algorithm converged upon. Thus, if the maximum number of iterations in the decoder is set at 10, then there can be an almost 10 times variation in decoder throughput between very low PER, where the average number of iterations may be 1 or 2, to a high PER behavior, where the maximum number of decoder iterations is frequently reached. Note that this test system was not actually sending packets over a real network that would encounter delay, jitter, and out-of-order packet delivery - so these throughput numbers can be looked at as an upper bound on a real communication system.

When considering these high throughput numbers for an erasure decoder, remember that there are fundamental differences between erasure codes and physical layer error correction codes. One difference is that the number of bits in each of these erasure codes is much higher than a typical physical layer code. With $k = 1000$ and each symbol equaling a typical packet size, which is set to 1024 bytes in this research, the number of bits per erasure code block is $(1000)(1024)(8) = 8192000$. So, even a decoder that only processes one erasure

code block per second is still achieving an effective throughput of 8 Mbps. Another key reason that throughput is higher with these erasure codes compared to physical layer codes is that so much of the decoding operation is a simple parallel XORing of the 8192 bits per symbol. In hardware implementations, it is simple to perform these XORs in parallel, which greatly increases throughput. Most physical layer decoding techniques must operate on soft values of probabilities about a single bit likelihood, which even when implemented with a fixed point value must still incorporate at least 8 bits or so per soft value. Thus the storage and calculation logic for a physical layer error correction decoder is at least a factor of 8 greater per bit of decoding throughput compared to an erasure decoder that only needs to perform XOR operations in parallel to do its decoding.

TABLE I
BLOCK ERROR RATES AT HIGH SNR

Raw PER	LDPC Code BLER	RS BLER	N_T	T_{sim} (sec)
$(2000, 1000)$ **LDPC Code, Iterative Decoding**				
0.375	$2.2 * 10^{-5}$	$2.1 * 10^{-5}$	$2 * 10^6$	2061
0.3594	0	$2 * 10^{-6}$	10^7	4013
0.3438	0	0	$2 * 10^8$	70397
$(2040, 1530)$ **LDPC Code, Iterative Decoding**				
0.1875	0.02	$7.3 * 10^{-3}$	10^6	812
0.1719	$1.3 * 10^{-4}$	$9.3 * 10^{-4}$	10^6	525
0.1562	0	$6.3 * 10^{-5}$	10^7	4143
0.1406	0	$2 * 10^{-6}$	10^8	34492

The statistics in Table I refer to the LDPC code *Block* error rate (BLER). If a block contains k symbols, and approximately 10% of the symbols are still erased after iterative decoding, then the *Symbol* erasure rate is approximately an order of magnitude lower than the BLER column values shown in the table. The number of LDPC code frames sent through the Monte Carlo simulation for a given raw PER is indicated by N_T. The comparable RS code shown in the table for the $(2000, 1000)$ LDPC code is of size $(250, 125)$. So, it is of the same rate as the LDPC code and exactly 8 times shorter. Since the RS code is MDS, it is not necessary to actually implement the RS decoder in the simulation and instead it is sufficient to assume a block error occurs if the number of erased symbols exceeds $n - k$. The number of RS block errors is designated as N_{err}^{RS}, so the RS BER is calculated as $N_{err}^{RS}/(8N_T)$. The comparable RS code shown in the table for the $(2040, 1530)$ LDPC code is of size $(255, 192)$, which is also one eighth the size and approximately the same code rate.

The raw PER values in the first column of Table I are chosen as a multiple of $1/64$ because the random number generator in the hardware creates a random 64-bit long integer, and the bottom 6 bits are used as the random variable to determine whether a packet erasure occurs or not.

VI. SUMMARY

This paper compares the performance of a girth-constrained, lower-triangular, moderate-length binary LDPC code versus a comparable rate RS code. The block erasure rate performance is similar between the two codes, as long as an LDPC code of at least eight times the block length of the comparable RS code is utilized. The reduced encoding and decoding complexity of the LDPC code gives it an advantage over RS codes as a candidate for an FEC erasure code in a digital IF standard such as DIFI. Although only a small number of LDPC code block lengths and code rates were presented in this paper, similar LDPC code constructions can be created with a wide variety of block lengths and code rates.

VII. FUTURE WORK

Testing the codec over an internet link, which can introduce delay, bursty error patterns and out-of-order packet reception, is the ultimate goal of this research. If the codec performs well in practice, then we would attempt to increase the erasure correction performance of the codes by moving to the ML decoding algorithm and also explore non-binary LDPC codes.

ACKNOWLEDGMENT

The author thanks professor Steve Wilson and ATG colleagues for reviewing the paper and offering useful input.

REFERENCES

[1] DIFI Consortium, "Digital IF interoperability standard v1.0," *IEEE Standard 4900-2021*, 2021. [Online]. Available: https://bit.ly/3gwVlnw

[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY: Wiley, 1991.

[3] V. G. Cerf and R. E. Kahn, "A protocol for packet network intercommunication," *IEEE Trans. Commun.*, vol. 22, no. 5, p. 637–648, 1974.

[4] M. G. Luby, T. Gasiba, T. Stockhammer, and M. Watson, "Reliable multimedia download delivery in cellular broadcast network," *IEEE Trans. on Broadcasting*, vol. 53, no. 1, pp. 235–246, Mar 2007.

[5] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[6] R. Singleton, "Maximum distance q-nary codes," *IEEE Trans. Inf. Theory*, vol. 10, no. 2, p. 116–118, April 1964.

[7] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, pp. 21–28, Jan 1962.

[8] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, June 2006.

[9] V. Roca, M. Cunche, C. Thienot, J. Detchart, and J. Lacan, "RS + LDPC-staircase codes for the erasure channel: Standards, usage and performance," in *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, October 2013, pp. 638–644.

[10] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, Sept 1981.

[11] ——, "Minimum-distance bounds by graph analysis," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 808–821, Feb 2001.

[12] R.G.Gallager, *Low-Density Parity-Check Codes*. MIT Press, 1963.

[13] J. Campello, D. Modha, and S. Rajagopalan, "Designing LDPC codes using bit-filling," in *Proc. IEEE Intl. Conf. Commun. (ICC)*, vol. 1, June 2001, pp. 55–59.

[14] "Future advanced SATCOM technologies (FAST) open standard digital-IF interface (OSDI) for SATCOM systems," https://tiaonline.org/, May 2016.

[15] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 585–598, Feb 2001.

[16] C. A. Cole, S. G. Wilson, E. K. Hall, and T. R. Giallorenzi, "A general method for finding low error rates of LDPC codes," *Available at http://arxiv.org/abs/cs.IT/0605051*, 2006.

[17] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat-accumulate codes," *Proc. 2nd. Int. Symp. Turbo Codes and Related Topics*, pp. 1–8, Sept 2000.

[18] G. Garrammone and B. Matuz, "Short erasure correcting LDPC IRA codes over GF(q)," *IEEE GlobeComm 2010*, pp. 995–1001, Dec 2010.