Low-Power 400-Gbps Soft-Decision LDPC FEC for Optical Transport Networks

Kevin Cushon, *Member, IEEE*, Per Larsson-Edefors, *Senior Member, IEEE*, and Peter Andrekson, *Fellow, IEEE*

***Abstract*—We present forward error correction systems based on soft-decision low-density parity check (LDPC) codes for applications in 100–400-Gbps optical transport networks. These systems are based on the low-complexity "adaptive degeneration" decoding algorithm, which we introduce in this paper, along with randomly-structured LDPC codes with block lengths from 30 000 to 60 000 bits and overhead (OH) from 6.7% to 33%. We also construct a 3600-bit prototype LDPC code with 20% overhead, and experimentally show that it has no error floor above a bit error rate (BER) of 10^{-15} using a field-programmable gate array (FPGA)-based hardware emulator. The projected net coding gain at a BER of 10^{-15} ranges from 9.6 dB at 6.7% OH to 11.2 dB at 33% OH. We also present application-specific integrated circuit synthesis results for these decoders in 28 nm fully depleted silicon on insulator technology, which show that they are capable of 400-Gbps operation with energy consumption of under 3 pJ per information bit.**

***Index Terms*—Application-specific integrated circuit (ASIC) synthesis, forward error correction (FEC), low-density parity-check (LDPC) codes, low power.**

I. Introduction

Forward error correction (FEC) is a critical part of modern high-performance communication systems. For optical communication in particular, there has been a great deal of development in soft-decision FEC systems since the introduction of coherent transmission. Because soft-decision FEC can make use of soft probability information for received symbols, it can achieve superior error correction performance compared to hard-decision FEC. This improvement is very appealing for long-haul optical transport network (OTN) applications, which place very high demands on FEC performance. Typically proposed requirements for these applications include throughputs of 100 Gbps or multiples thereof, low power consumption, coding gain approaching the theoretical limit, and special adaptations for optical channels [1].

Low-density parity-check (LDPC) codes are a popular choice for such systems, as these codes can readily achieve the required coding gain and throughput with current application-specific integrated circuit (ASIC) fabrication technology. For example, Onohara *et al.* [2] proposes an FEC system for 100 Gbps OTN with a soft-decision LDPC inner code, concatenated with a hard-decision Reed–Solomon outer code to suppress the LDPC code's error floor [3]. An alternative approach in [4] uses a non-concatenated LDPC code with post-processing. Another proposed 100 Gbps OTN decoder in [5] makes use of a convolutional LDPC code concatenated with a Bose-Chaudhuri-Hocquenghem (BCH) outer code. Other recent papers have proposed spatially coupled LDPC (SC-LDPC) codes, which provide even greater coding gain than conventional block LDPC codes [1], [6].

However, the iterative soft-decision message-passing algorithms used to decode LDPC codes, such as the normalized min-sum algorithm (NMSA), are very costly in terms of silicon area and energy consumption, especially when they must meet the aforementioned performance goals. In [7], it is estimated that the LDPC decoder from [4] and corresponding encoder respectively consume 15.8% and 6.6% of the total energy in a 100 Gbps DP-16-QAM link over 1100 km of fiber, and 10.1% and 4.2% of the total energy in a DP-QPSK link over 2400 km. Thus, the FEC components should be a high priority for energy reduction efforts.

Many previous LDPC decoder designs have tried to improve hardware and energy efficiency, often by trading off a small amount of error correction performance. In particular, decoders for 10 Gbps Ethernet applications are quite promising for adaptation to OTN, since they share the requirements of high throughput and low error floors. Bit-serial min-sum [8], split-row min-sum [9], and simplified variable weight min-sum [10] are simplified min-sum variants that achieve very good trade-offs between error correction capability and efficiency. In addition to these, there are binary message passing decoders, in which all messages are single bits. This results in them having much simpler computational units compared to min-sum, but usually at a larger penalty in bit error rate (BER) performance. Stochastic [11] and relaxed half-stochastic [12] decoders exchange messages as probabilistic bit streams. They achieve good BER performance, but require a high number of iterations, which may be troublesome in latency-sensitive OTN. Gradient descent bit flipping [13] and noisy gradient descent bit flipping [14] are recent variants on weighted bit flipping (WBF) decoding that decode more quickly than previous WBF variants - these could become candidates for OTN systems as well. Finally, the improved differential binary (IDB) algorithm [15] is another soft bit-flipping algorithm that decodes using accumulated sums of binary messages.

Manuscript received April 6, 2016; revised June 7, 2016; accepted July 25, 2016. Date of publication August 10, 2016; date of current version September 2, 2016. This work was presented in part at the 2015 European Conference on Optical Communication, Valencia, Spain, September 2015. This work was supported by the Knut and Alice Wallenberg Foundation.

K. Cushon and P. Larsson-Edefors are with the Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg SE-41296, Sweden (e-mail: cushon@chalmers.se; perla@chalmers.se).

P. Andrekson is with the Photonics Laboratory, Chalmers University of Technology, Gothenburg SE-41296, Sweden (e-mail: peter.andrekson@chalmers.se).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JLT.2016.2598440

In our earlier conference paper [16], we introduced soft-decision LDPC FEC systems based on IDB as a decoding algorithm as a less complex and more energy-efficient alternative for 100+ Gbps OTN FEC. In Section II of this follow-up, we present a revised version of IDB called the *adaptive degeneration* (AD) algorithm. In Section III, we describe the design of the LDPC codes used in this work, and show that they have low-complexity encoders. We present implementation results and performance data for AD decoders in Section IV. We provide experimental proof that these systems have no error floor above a BER of 10^{-15}, obtained using an FPGA-based hardware emulator. We evaluate the error rate performance of the AD algorithm using 30 000- and 60 000-bit blocklength codes with varying overhead (OH), and present ASIC synthesis results and power estimations for these decoders using 28 nm fully depleted silicon on insulator (FD-SOI) fabrication technology. Finally, we conclude this work in Section V.

II. The AD Decoding Algorithm

After briefly introducing LDPC codes in Section II-A, we will give an overview of the AD decoding algorithm in Section II-B after which we will discuss important algorithmic parameters in Section II-C.

A. Introduction to LDPC Codes

An LDPC code is characterized by a sparse parity check matrix $\mathbf{H}$ with dimensions $m \times n$, where n is the number of bits in the block and m is the number of parity checks. An (n, k) LDPC code has k information bits per block. If $\mathbf{H}$ is full rank, $k = n - m$. A frame $\mathbf{x}$ with length n is a valid codeword iff $\mathbf{H}\mathbf{x}^T = \mathbf{0}^T$. Equivalently, an LDPC code may be represented by a Tanner graph, where variable nodes (VNs) v_i represent the columns of $\mathbf{H}$, and check nodes (CNs) c_j represent the rows. An edge exists between v_i and c_j iff $\mathbf{H}_{j,i} = 1$. The degree of a node (d_v for VNs and d_c for CNs) is equal to the number of edges connecting to it. The set of VNs neighboring CN c is represented by V_c, and the set of CNs neighboring VN v is C_v.

B. Algorithmic Description

The AD algorithm for decoding LDPC codes is a variant of the IDB algorithm [15]. It is described in Algorithm 1.

During initialization, the VN memories M_v are set to the log-likelihood ratios (LLRs) of the received bits, with x_v being the transmitted bits and y_v what is received from the channel. The iteration count index i is set to 0. Each M_v (and LLR) is quantized using q bits. The vector u stores the number of unsatisfied parity check equations for the current iteration (u_0) and the previous ℓ iterations (u_1 through u_ℓ). These are all initialized to $m + 1$, which is equal to one greater than the number of parity checks in the LDPC code.

In lines 12–15, the VN-to-CN messages $b_{v\to c}$ and hard-decision bits h_v are computed. These are both equal to the sign bit of M_v. Note that only one message is generated per VN, and that this same message is sent to all neighboring CNs.

Algorithm 1: The AD Decoding Algorithm.

```
1  // Description of parameters:
2  // $\gamma_0, \gamma_1$: Possible values of $\delta$, $0 \le \gamma_0 < \gamma_1$.
3  // $\ell$: Number of previous iterations to consider
       when calculating $\delta$.
4  // $T_u$: Threshold of unsatisfied parity checks for
       setting $\delta = \gamma_1$.
5  // $s$: CN-to-VN message scaling factor, $0 < s \le 1$.
6  for $v \in [0 \,..\, n-1]$ do                    // Initialization
7      $M_v \leftarrow LLR_v = \ln\left[\frac{P(y_v|x_v = 0)}{P(y_v|x_v = 1)}\right]$
8  end
9  $i \leftarrow 0$
10 $u_\ell, u_{\ell-1}, \dots, u_0 \leftarrow m + 1$
11 repeat                                            // Decoding
12     for $v \in [0 \,..\, n-1]$ do
13         $b_{v\to c} \leftarrow \begin{cases} 0, & \text{if } M_v \ge 0 \\ 1, & \text{if otherwise} \end{cases}$
14         $h_v \leftarrow b_{v\to c}$
15     end
16     for $c \in [0 \,..\, m-1]$ do
17         $p_c = \bigoplus_{v \in V_c} (b_{v\to c})$
18         for $v \in V_c$ do
19             $b_{c\to v} = p_c \oplus b_{v\to c}$
20         end
21     end
22     $u \leftarrow u \ll 1$
23     $u_0 \leftarrow \sum p$
24     if $u_0 = 0$ then
25         Declare decoding successful, output $h$
26     end
27     if $u_0 = \max(u)$ and $u_0 < T_u$ and $i < i_m - \ell$ then
28         $\delta \leftarrow \gamma_1$
29         $u_\ell, u_{\ell-1}, \dots, u_0 \leftarrow m + 1$
30     else
31         $\delta \leftarrow \gamma_0$
32     end
33     for $v \in [0 \,..\, n-1]$ do
34         $\sigma \leftarrow d_v - 2 \cdot \sum_{c \in C_v} b_{c\to v}$
35         $M_v \leftarrow \begin{cases} M_v + s \cdot \sigma - \delta, & \text{if } M_v \ge 0 \\ M_v + s \cdot \sigma + \delta, & \text{if otherwise} \end{cases}$
36     end
37     $i \leftarrow i + 1$
38 until $i = i_m$
39 Declare decoding failed, output $h$
```

Lines 16–21 describe the computations performed in the CNs. These are the parities p and CN-to-VN messages $b_{c\to v}$. The $\oplus$ symbol indicates modulo-2 addition.

Next, in lines 22–26, the elements of u are shifted one place to the left ($u_\ell \leftarrow u_{\ell-1}, u_{\ell-1} \leftarrow u_{\ell-2}$, and so on). The new number of unsatisfied parity checks is computed and stored in u_0. If $u_0 = 0$, then decoding has converged on a valid codeword. The algorithm stops immediately and outputs the hard-decision bit vector h.

Lines 27–32 determine the value of the degeneration factor δ. The first condition on line 27 determines if decoding has made progress (in terms of reducing the number of unsatisfied parity checks) within the last ℓ iterations. The second determines if the number of unsatisfied parity checks is below a threshold T_u, which is a free parameter. The third determines if there are at least ℓ more iterations before the maximum iteration limit i_m is reached. If all three conditions are met, then a high magnitude degeneration factor γ_1 is used. In addition, all elements of u are reset to $m + 1$, which ensures that γ_1 cannot be used again in the following ℓ iterations. If the conditions are unmet, then a small magnitude degeneration factor γ_0 is applied instead. The possible values of δ, γ_0 and γ_1, are free parameters as well. They are positive numbers with $\gamma_1 > \gamma_0$.

In lines 33–36, the VN memories M_v are updated. For each VN, all incoming messages $b_{c \to v}$ are added, with each '0' message assigned a value of $+1$ and each '1' assigned a value of -1. This sum is scaled by a factor s that allows the weight of incoming messages to be adjusted for improved performance (s is analogous to the scaling factor α in the NMSA). Finally, the degeneration factor δ is added or subtracted depending on the sign of M_v.

At this point, the iteration count index i is incremented, and if it is equal to the maximum number of iterations i_m, failure is declared and decoding stops. If not, then the next iteration begins.

C. Error Floor Performance of the AD Algorithm

The differences between AD and IDB are that AD uses different values for δ depending on the decoding state, and does not use the IDB relaunching technique. Relaunching (in which the decoder restarts and uses different parameters or applies perturbations to the initial LLRs) is not practical for use in OTN applications because it requires too many decoding iterations.

The motivation for varying δ is to improve decoding performance in the error floor region. LDPC codes are known to exhibit error floors, where the slope of the BER curve diminishes significantly below a certain BER. Error floors are primarily caused by *trapping sets*, which are defined as subgraphs of the Tanner graph of an LDPC code containing a small number of incorrect symbols that reinforce one another through wrongly satisfied parity checks [3], [17].

Since OTN applications require very low BERs (i.e., BER $< 10^{-15}$), these error floors cause a significant loss in coding gain. Many previously proposed LDPC-based FEC systems include measures to mitigate or suppress the error floor, such as using a concatenated outer code (as in [2] and [5]), or modifying the decoding algorithm to avoid or break out of trapping sets (as in [4], [18], and [10]).

In the IDB and AD algorithms, degeneration can correct trapping sets by changing the signs of erroneous VNs, even if a majority of inputs agree with their current values. Specifically, degeneration will cause the magnitude of M_v to decrease and eventually change sign if

$$c_{\mathrm{ws}} - c_u < \frac{\delta}{s}, \tag{1}$$

where c_{ws} and c_u are respectively the number of wrongly satisfied and unsatisfied CNs connected to the VN. However, degeneration can also wrongly cause correct VNs to change sign, as it is impossible to distinguish between a correctly satisfied parity check and a wrongly satisfied one. If δ is constant, as in IDB, then $\delta \le s$ (if d_v is odd) or $\delta \le 2s$ (if d_v is even) is required to achieve good decoding performance, because otherwise degeneration will overpower legitimate majority inputs and propagate errors throughout the graph [15]. Hence, IDB can only correct trapping sets where participating VNs have $c_{\mathrm{ws}} = c_u$. If δ is variable, as in AD, then it becomes possible to have a small δ during normal decoding, and then use a larger δ when the decoder detects the presence of a trapping set. Using this technique, the AD algorithm can correct trapping sets in which all VNs have $c_{\mathrm{ws}} > c_u$. More specifically, a VN in a trapping set satisfying the condition

$$c_{\mathrm{ws}} - c_u \le \frac{\gamma_0}{s} \tag{2}$$

will eventually change sign due to degeneration, because the average value of δ is greater than γ_0. However, it is not guaranteed that all such trapping sets are correctable, because they may collapse to a smaller trapping set in which none of the participating VNs satisfy Equation 2. Furthermore, the possibility remains for large values of δ to cause correct VNs to wrongly change sign.

Thus, while large values of δ can be useful for correcting trapping sets, a balance is needed to prevent introducing additional errors. To achieve this balance, we set three separate conditions that must all be met in order to use a large δ, which are shown in line 24 of Algorithm 1. The first condition, $u_0 = \max(u)$, indicates that decoding has not made any progress (as measured by the number of unsatisfied parity check equations) within the last ℓ iterations. The second, $u_0 < T_u$, places a maximum threshold T_u on the number of unsatisfied checks. This is necessary because a large number of unsatisfied checks implies the existence of many erroneous bits and low-magnitude values of M_v. Using a large degeneration factor under these conditions can flip hundreds of correct bits and cause an unrecoverable error. Finally, the third condition, $i < i_m - \ell$, prevents γ_1 from being used in the final ℓ iterations. Similarly to the previous condition, a large degeneration factor can cause many correct but low-magnitude M_vs to change sign. If this occurs near the end of decoding, they will not have time to recover, and the frame will have more bit errors as a result. We note that satisfaction of these conditions does not prove that the decoder is stuck in a trapping set - only that decoding has not progressed for the last ℓ iterations.

Determining the value of δ for a given iteration requires actively monitoring the number of unsatisfied parity check equations, which requires an m-input 1-bit addition. This is a significant source of additional circuitry and critical path delay as compared to the IDB algorithm. It is possible to avoid this by controlling δ using passive techniques, such as applying γ_1 only at certain values of i, but simulations of this variation determined that long "recovery periods" between successive applications of γ_1 are necessary to avoid causing unrecoverable errors. As a result, 2 to 3 times as many iterations are necessary to achieve the same level of BER performance compared to the active method, which makes it difficult to meet the latency and

TABLE I
ENCODING COMPLEXITY

LDPC code	Complexity (2-input XORs)	Area (mm^2)
(30 000, 26 786)	$3.74 \cdot 10^5$	0.326
(30 000, 25 000)	$3.86 \cdot 10^5$	0.336
(60 000, 56 235)	$5.57 \cdot 10^5$	0.485
(60 000, 53 570)	$5.75 \cdot 10^5$	0.500
(60 000, 50 000)	$5.99 \cdot 10^5$	0.521
(60 000, 48 000)	$6.12 \cdot 10^5$	0.533
(60 000, 45 000)	$6.33 \cdot 10^5$	0.551

throughput requirements of OTN applications. Our simulations and ASIC synthesis results (see Section IV) determined that the active method is necessary to achieve throughputs greater than 100 Gbps, and that the m-input 1-bit adder this necessitates is not a critical problem.

III. Code Design

While the IDB algorithm, and by extension the AD algorithm, require LDPC codes with $d_v \geq 6$ to decode effectively [15], the low circuit and wiring complexity makes it practical to implement fully parallel decoders using LDPC codes with long block lengths and irregular structures. Since (constrained) randomly structured codes are known to approach capacity at long block lengths [19], we selected these for implementation, and found that they perform very well under AD decoding.

We implement several LDPC codes to investigate the performance of the AD algorithm over a range of block lengths and code rates. The first two are 30 000-bit block length codes with 12% and 20% OH, which correspond to (30 000, 26 786) and (30 000, 25 000) respectively in (n, k) notation. The other codes have 60 000-bit block lengths and OHs of 6.7%, 12%, 20%, 25%, and 33.3%, respectively corresponding to (60 000, 56 235), (60 000, 53 570), (60 000, 50 000), (60 000, 48 000) and (60 000, 45 000). Finally, we also have constructed a (3600, 3000) prototype code for use in FPGA decoder implementations.

All of our codes have column weight and VN degree $d_v \approx 6$, and are randomly constructed with certain constraints in place to improve code quality, such as avoiding length-4 cycles [20]. To permit efficient encoding using the Richardson–Urbanke (RU) encoding algorithm [21], these codes are generated in approximate upper-triangular form with the gap size $g = 600$. This gap size was chosen to minimize the appearance of trapping sets in the codes. Trapping set searches on these codes using a heuristic algorithm found nothing, suggesting that these codes have very good error floor performance [22]. Since it is impossible for a regular code with even d_v to be full rank, each code was generated with $d_v = 6$ and 1 redundant row, which was then deleted. As a result, all codes are full rank but slightly irregular, with a small number of VNs having $d_v = 5$. The reason we do this is that the RU algorithm requires the parity-check matrix to be full rank. Adaptation to matrices with redundant rows is possible, but requires non-trivial modifications.

Table I shows encoding complexity measured in terms of the number of 2-input binary XOR operations required to encode a block for the LDPC codes used in this work, as well as the estimated area of the encoder in 28 nm FD-SOI. Compared to encoding with the $\mathbf{G}$ matrix, RU encoding reduces complexity by factors of about 100 to 500. In [7], it is estimated that $\mathbf{G}$ matrix encoding of a (24 576, 20 482) LDPC code consumes 36 pJ per information bit with an ASIC encoder implemented in 40 nm complementary metal-oxide-semiconductor (CMOS), which accounts for 4.2–6.6% of the total link energy in a 100 Gbps coherent optical fiber system. From this result, we can conclude that encoding for our codes will consume an insignificant amount of energy.

However, these encoders require a relatively large area compared to encoders for certain types of structured codes. Specialized code constructions, such as irregular repeat-accumulate (IRA) and extended IRA codes [23] have very simple encoders, but these are unsuitable for use with the AD algorithm because they have a large number of degree-2 VNs. Quasi-cyclic (QC) codes, which are one of the most commonly used classes of LDPC code, have serial and partially-parallel encoders with highly compact circuit implementations [24]. However, these encoders require a large number of combinational operations, which leads to high energy consumption in spite of their low area. QC-LDPC codes with long block lengths, such as those proposed for use in OTN applications, require on the order of millions to tens of millions of 2-input XOR operations using the two-stage method described in [24]. In comparison, the RU encoders listed in Table I have fewer total operations, but they must all be implemented in parallel.

In this work, we investigate only randomly structured LDPC codes, since they are easily constructed, have good error correction performance, appear to lack harmful trapping sets, and have energy-efficient (if not necessarily area-efficient) encoders.

IV. Implementation Results

We have implemented AD decoders for the seven aforementioned LDPC codes with 30 000-bit and 60 000-bit blocklengths in software to characterize their error correction performance. We have also implemented decoders for the $(3600, 3000)$ prototype code in software and on an FPGA, which we use to measure the BER performance of this code to below 10^{-15}, and thus prove that it has no error floor above this level. Finally, we present ASIC synthesis results using 28-nm FD-SOI fabrication technology and standard cells from STMicroelectronics, use these to estimate the silicon area, throughput, and power consumption of our decoders, and compare our results with those from previous works.

A. Parameter Selection

While the AD algorithm has many parameters that must be set for implementation, some are tightly constrained by practical considerations, while the rest can be determined empirically. To begin, we set the number of quantization bits for input LLRs and M_v to $q = 5$, as any lower value results in seriously degraded performance. Using a conventional fixed-point number format with 1 sign bit, 3 integer bits, and 1 fractional bit, the smallest representable positive number is 0.5, which is thus

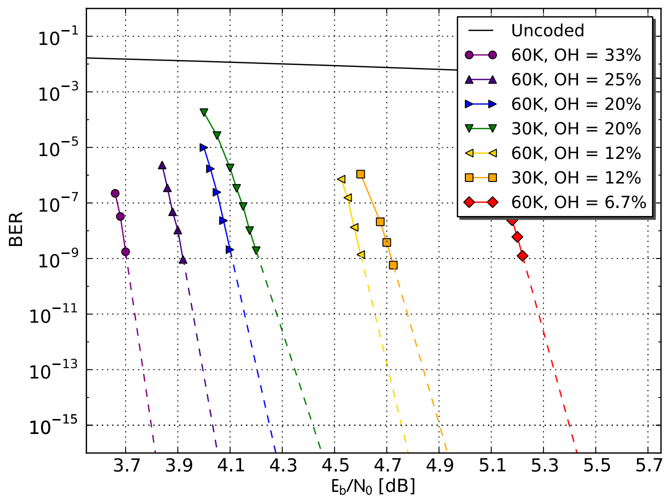

Fig. 1. BER performance of 30 K and 60 K-bit blocklength codes with AD decoding, measured using Monte Carlo simulations in software (solid lines) and linear projections (dashed lines).

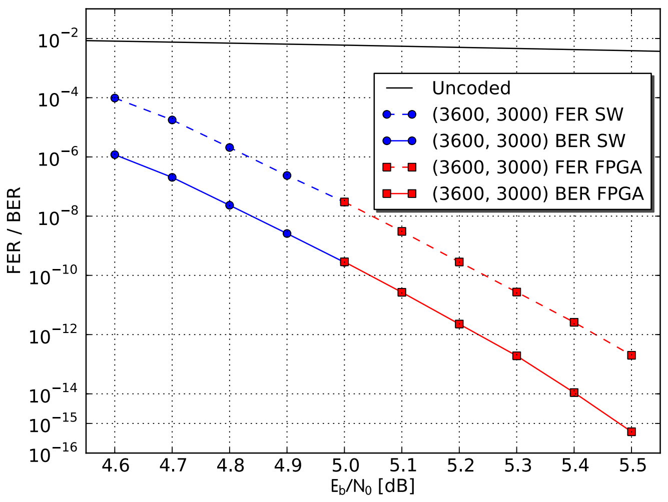

Fig. 2. Error rate performance of the (3600, 3000) prototype code in software and FPGA.

the optimal value of γ_0 [15]. This constrains the value of s to $0.25 \leq s \leq 0.5$. Based on Equation 2, s should be as small as possible to achieve the best error floor performance, so we set $s = 0.25$. The maximum number of iterations i_m is determined by the desired throughput and latency. To achieve 400 Gbps in ASIC implementations with a reasonable clock frequency, $i_m = 49$ is chosen (with one extra iteration reserved for loading and unloading the decoder). The values of γ_1 and ℓ were optimized via a brute-force search, with $\gamma_1 = \ell = 3$ determined to be optimal or negligibly different from optimal for all codes. Finally, the threshold T_u was determined by removing it, running simulations, and observing the values of u_0 when unrecoverable mass sign flips of M_v occurred. These were observed to occur only when $u_0 > 200$ during extensive simulations. This is many times greater than the expected values of u_0 produced by small trapping sets, so we can set $T_u = 64$ to provide a margin of safety. Every implemented decoder, including the FPGA prototype, uses these same parameter values.

B. *Error Rate Performance*

Fig. 1 shows plots of BER performance for each of the 30 000- and 60 000-bit blocklength decoders, while Fig. 2 plots the BER and frame error rate (FER) performance of the (3600, 3000) prototype code. All plots assume binary phase-shift keying modulation and additive white Gaussian noise.

The software results were obtained using multi-threaded C programs, with which it is feasible to simulate at BERs of roughly 10^{-9} in a reasonable amount of time. At least 20 frame errors were recorded for the lowest data points, with more errors (generally 100) collected for higher data points. However, it is not computationally feasible to perform software simulations down to the OTN application target BER of 10^{-15}. It is also not possible to implement these codes on currently available FPGAs, both because of their long blocklengths, and because they do not have any regular structure that can be exploited to produce area-efficient partially parallel implementations, as can be done with quasi-cyclic codes such as in [4] and [18].

Thus, to obtain experimental results for AD decoding at low BERs, we constructed a (3600, 3000) prototype code using the same method and column weight as the longer codes. We then implemented an AD decoder for this code on a Xilinx Virtex-7 XC7VX690T FPGA [25]. The FER and BER of this system are plotted in Fig. 2. The FPGA system includes XORshift random number generators [26] to generate the input LLRs on-chip. The uniform random numbers produced by these generators are mapped to one of the 32 possible LLRs (given $q = 5$) using a binary decision tree. The system also includes a logger module that keeps track of the number of completed blocks, block errors, bit errors, and various other information. At high SNR, this system achieves an information throughput of 34 Gbps. Even at this rate, we observed only 5 frame errors for the lowest data point in Fig. 2 after 30 days of operation, and 3 of those were correctable with more decoding iterations. Despite the small sample size for that data point, these results prove through direct emulation that there is no error floor above a BER of 10^{-15} for this decoder.

To estimate the net coding gain (NCG) of the larger codes at a BER of 10^{-15}, we plot linear projections of the lowest two data points, which are shown as dashed lines in Fig. 1. This requires assuming that none of our codes will have an error floor above 10^{-15}. However, we believe that this assumption is well justified. As mentioned in Section III, we performed trapping set searches on our codes using importance sampling techniques [22], but these found no failure cases. This technique was successful in finding trapping sets in codes constructed with smaller values of g, however. Furthermore, this technique was also used in [4] to discover the dominant trapping sets of a (24 576, 20 482) LDPC code and accurately predict its error floor performance. Thus, the fact that our searches found no trapping sets is evidence of absence of trapping sets that could produce an impact at BERs above 10^{-15}.

TABLE II
ASIC SYNTHESIS RESULTS AND DECODER PERFORMANCE SUMMARY

	(30 000, 26 786)	(30 000, 25 000)	(60 000, 56 235)	(60 000, 53 570)	(60 000, 50 000)	(60 000, 48 000)	(60 000, 45 000)
Area (mm^2)	4.54	4.53	9.04	9.09	9.25	9.49	9.62
Area (core only) (mm^2)	3.73	3.71	7.41	7.46	7.62	7.87	8.00
Clock frequency (MHz)	373	400	356	373	400	417	444
Max. iterations	49	49	49	49	49	49	49
Info. throughput (Gbps)	200	200	400	400	400	400	400
Latency (ns)	134	125	141	134	125	120	113
Coding OH (%)	12	20	6.7	12	20	25	33.3
Estimated NCG @ BER = 10^{-15} (dB)	10.1	10.55	9.6	10.25	10.75	10.95	11.2
Gap to capacity (dB)	1.9	2.1	1.5	1.7	1.9	2.0	2.2
Power (mW)	369	418	688	760	941	974	1134
Power (core only) (mW)	301	346	558	624	759	824	975
Energy (pJ/info. bit)	1.85	2.09	1.72	1.90	2.35	2.44	2.85
Energy (core only) (pJ/info. bit)	1.50	1.73	1.40	1.56	1.90	2.06	2.44

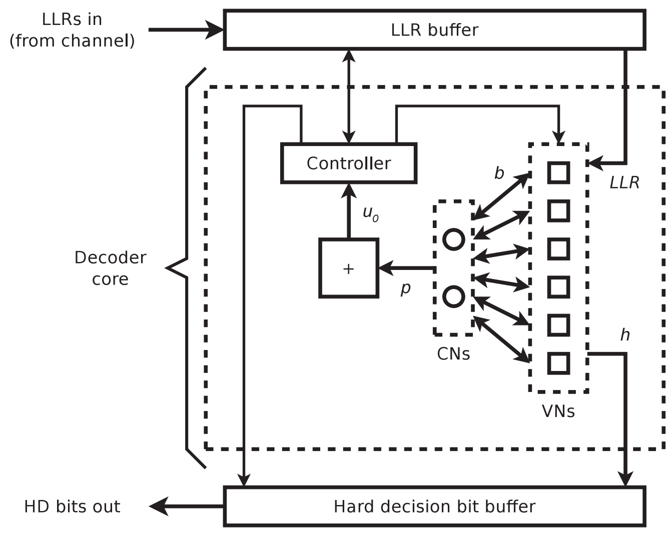

Fig. 3. Top level block diagram of the decoder circuit implementations showing the major components. Labels in *italics* correspond to variable names in Algorithm 1.

The BER performance of the FPGA prototype, as plotted in Fig. 2, provides further evidence. These results prove that there is no error floor above 10^{-15} for the prototype code. We can infer that this applies to the longer codes as well, since they were constructed using the same algorithm and same column weight. While some of the longer codes have lower OH than the prototype code, they have lower density due to their much longer blocklengths. We can therefore expect a lower incidence of intersecting short cycles that cause trapping sets [22], even for the codes with lower OH.

C. ASIC Synthesis and Power Estimation

We performed ASIC synthesis for all seven codes with 30 000- or 60 000-bit blocklengths previously described in this work. The results are summarized in Table II, alongside the estimated NCG and power consumption of each decoder.

Fig. 3 shows a top level block diagram of our hardware decoder design. All decoders are fully parallel, and include n VNs and k CNs. In addition to the VNs and CNs, the decoder core also consists of a controller state machine, and a k-input 1-bit adder to determine the number of unsatisfied parity checks u_0. The complete system includes an $n \times q$-bit buffer for the input LLRs and a k-bit buffer for the hard-decision bits. These are implemented as addressable register banks rather than shift registers to reduce their dynamic power consumption. Clock gating is used extensively in both the decoder core and the buffers.

Synthesis was performed with Cadence RTL Compiler version 14.11 using 28-nm FD-SOI technology from STMicroelectronics. We provide silicon area results for both the complete system, as shown in Fig. 3, as well as for the decoder core only (i.e., excluding the I/O buffers). The clock frequencies were chosen to achieve exactly 200 Gbps information throughput with the 30 000-bit blocklength decoders, and 400 Gbps with the 60 000-bit blocklength decoders. In addition, all designs have a minimum of 400 ps of timing slack for later phases of ASIC implementation. All of these results were obtained using worst-case process and operating conditions (slow process, 0.8 V supply voltage, 125 °C junction temperature). Due to the low wiring complexity of the AD algorithm, we do not anticipate routing congestion or excessively low silicon area utilization. We note that the IDB decoder of [15], which has identical inter-node wiring complexity, achieved 95% utilization in a 65-nm process with 7 metal layers, albeit with a much smaller $(2048, 1723)$ LDPC code. The 28-nm process used in this work is available with up to 10 metal layers, and even more layers are available in other modern processes. Thus, we do not expect these designs to have serious routing problems, even with the larger codes.

The power and energy estimations were performed using Synopsys PrimeTime version 2011.6, with post-synthesis data for wiring parasitics and switching activity. Likewise as for silicon area, we provide results both including and excluding the I/O buffers. The switching activity data was obtained via simulations of 100 randomly generated blocks, with E_b/N_0 set to the estimated BER $= 10^{-15}$ point of each decoder. During these simulations, data was streamed continuously in and out of the decoder, at a rate of $n/(i_m + 1)$ LLRs input and $k/(i_m + 1)$ hard-decision bits output per clock cycle, in order to simulate operation in a practical setting. These power and energy estimates use typical process and operating conditions (typical process, 0.9 V supply voltage, 25 °C junction temperature). Static power consumption is estimated at approximately 60 mW for the 30 000-bit blocklength decoders, and 120 mW for the 60 000-bit blocklength decoders. These estimates were generated with Synopsys PrimeTime and are included in the power figures listed in Table II.

D. Comparison With Previous Works

Compared with previously reported soft-decision FEC schemes for OTN applications, AD decoders achieve much higher throughput and lower energy consumption. To the best of our knowledge, these are the first soft-decision FEC decoders capable of 400 Gbps operation in the literature. However, they provide lower NCG than several higher-complexity FEC systems, such as LDPC codes decoded using the NMSA [4], Turbo product codes (TPCs) [27], and SC-LDPC codes [6]. This makes AD decoding an attractive option for applications that do not require the maximum attainable NCG [28].

While it is not an OTN decoder, [29] provides a useful reference as a layered offset min-sum LDPC decoder. It uses a (2048, 1723) LDPC code with similar column weight ($d_v = 6$) and OH ($\sim 20\%$) to codes used in this work. It also employs post-processing to suppress the error floor, and uses an aggressively pipelined architecture that would be well suited to high-throughput OTN applications. Implemented in 65-nm CMOS, this decoder has a cell area of 4.52 mm^2, minimum guaranteed information throughput of 7.18 Gbps, and energy consumption of 69.8 pJ per information bit. We then scale the area and throughput linearly to a code size of 30 000, and apply scaling factors of 1.6 to clock frequency, 0.4 to area, and 0.3 to energy to estimate this design's performance in 28-nm FD-SOI (these are the scaling factors we observed while migrating our designs from 65-nm CMOS in [16] to this work). This results in an area of 26.5 mm^2, throughput of 166.7 Gbps, and energy of 20.9 pJ/bit. Our (30 000, 25 000) AD decoder is 6 times smaller, 20% faster in terms of absolute throughput, 7 times faster in terms of throughput per unit area, and 10 times more energy efficient. Despite having a maximum iteration limit of 14 and having a much higher clock frequency than the AD decoder, the layered decoder achieves lower throughput due to requiring 12 clock cycles to complete an iteration. In contrast, the AD decoders in this work all have maximum iteration limits of 49 and complete an iteration in a single clock cycle.

The NMSA-based LDPC decoder proposed in [4] achieves NCG of 11.3 dB using a (24 576, 20 482) LDPC code with 20% OH, and in [7] it is estimated this decoder would consume 60 pJ per information bit in 28-nm CMOS. This is much higher than the estimated energy consumption for the layered decoder detailed above. However, we note that the estimates in [7] assume a conventional parallel (i.e., non-layered) decoder, and also take the higher wiring parasitics of a large decoder circuit into account. Unfortunately, there are no ASIC area estimates for this decoder. The (60 000, 50 000) AD decoder in this work achieves an estimated 0.55 dB lower coding gain, and a factor of 25 reduction in energy consumption. Due to the lower complexity of the AD algorithm, we seek to make up some of the BER performance gap with NMSA by using longer blocklengths; hence we use the 60 000-bit blocklength decoder in this comparison.

A TPC decoder for OTN is proposed in [27]. Synthesized in 40-nm CMOS, it has NCG of 11 dB at 15% OH, an area of 23 mm^2, and power consumption of 7 W (and thus energy consumption of 70 pJ/bit, as it operates at 100 Gbps). Scaling to 28-nm gives an area of 11.5 mm^2 and energy consumption of 49 pJ/bit, using power scaling of 30% per process generation at constant frequency [7]. This is a considerable improvement over the layered LDPC decoder in terms of area and throughput per unit area, though energy consumption is higher. Our (60 000, 50 000) AD decoder achieves 5 times higher throughput per unit area, and 21 times lower energy consumption. The interpolated NCG of our decoders at 15% OH is 10.44 dB, so the performance gap is similar to that between AD and NMSA.

SC-LDPC codes have achieved estimated NCG as high as 12.18 dB at 25% OH in FPGA implementations [6], but no ASIC implementation or power consumption figures are available for these codes at present. They can be reasonably assumed to be comparable or slightly higher than the figures for conventional block LDPC codes.

Another low-power soft-decision LDPC decoder for 100 Gbps optical applications is presented in [5]. This design uses a convolutional LDPC decoder with a high-rate BCH outer code. It achieves 10.2 dB NCG with 12% OH, and 100 Gbps throughput in 2 W in 28-nm CMOS, giving an energy per bit of 20 pJ. The 12% OH AD decoders presented in this work achieve similar NCG (10.1–10.25 dB) with approximately an order of magnitude lower energy consumption.

Staircase codes are a class of hard-decision codes that achieve NCG approaching that of soft-decision AD decoding, so a comparison with them is also apt. The NCG of practical staircase codes ranges from 9.54 dB at 6.7% OH, to 10.41 dB at 20%, to 10.70 dB at 33.3% [30]. Their maximum achievable performance with infinite blocklengths is 9.56 dB at 6.7% OH, 10.64 dB at 20%, and 11.03 dB at 33.3% [31]. In comparison, the AD decoders presented in this work achieve estimated NCG of 9.6 dB at 6.7% OH, 10.75 dB at 20%, and 11.2 dB at 33.3%. Thus, AD decoding outstrips both the practical and theoretical maximum performance of staircase decoding, particularly at higher OHs. Unfortunately, no ASIC implementation or power consumption figures are available for staircase decoders, nor are they available for the component BCH decoders in modern fabrication technologies. However, we note that staircase codes require much longer blocklengths to achieve this level of error correction performance (on the order of 400 000 bits).

V. Conclusion

In this paper, we presented soft-decision FEC systems for OTN applications. These systems are based on randomly-constructed LDPC codes in conjunction with the reduced complexity AD decoding algorithm, which allows practical fully-parallel ASIC implementations of codes with long block lengths. The estimated NCG of these decoders ranges from 9.6 dB at 6.7% OH to 11.2 dB at 33.3% OH. Since the LDPC codes used in this work do not need to have regular structure, it is possible to construct them to have low complexity encoders. ASIC synthesis results show that these decoders easily achieve 200–400 Gbps information throughput, with low circuit complexity, and energy consumption of less than 3 pJ per information bit, which is 10 to 25 times lower than previously reported soft-decision decoders for OTN. AD decoding achieves a very good trade-off between throughput, energy consumption, and error correction performance for applications that do not require maximal coding gain.

REFERENCES

[1] A. Leven and L. Schmalen, "Status and recent advances on forward error correction technologies for lightwave systems," *J. Lightw. Technol.*, vol. 32, no. 16, pp. 2735–2750, Aug. 2014.

[2] K. Onohara *et al.*, "Soft-decision-based forward error correction for 100 Gb/s transport systems," *IEEE J. Sel. Topics Quantum Electron.*, vol. 16, no. 5, pp. 1258–1267, Sep. 2010.

[3] T. J. Richardson, "Error-floors of LDPC codes," in *Proc. 41st Annu. Allerton Conf. Commun. Control Comput.*, Oct. 2003, pp. 1426–1435.

[4] D. A. Morero, M. A. Castrillon, F. A. Ramos, T. A. Goette, O. E. Agazzi, and M. R. Hueda, "Non-concatenated FEC codes for ultra-high speed optical transport networks," in *Proc. IEEE Global Telecommun. Conf.*, Dec. 2011, pp. 1–5.

[5] M. Li *et al.*, "Low-overhead low-power-consumption LDPC-based FEC solution for next-generation high-speed optical systems," in *Proc. Opt. Fiber Commun. Conf. Exhib.*, Mar. 2015, pp. 1–3.

[6] L. Schmalen, V. Aref, Junho Cho, D. Suikat, D. Rosener, and A. Leven, "Spatially coupled soft-decision error correction for future lightwave systems," *J. Lightw. Technol.*, vol. 33, no. 5, pp. 1109–1116, Mar. 2015.

[7] B. S. G. Pillai *et al.*, "End-to-end energy modeling and analysis of long-haul coherent transmission systems," *J. Lightw. Technol.*, vol. 32, no. 18, pp. 3093–3111, Sep. 2014.

[8] A. Darabiha, A. Chan Carusone, and F. R. Kschischang, "Power reduction techniques for LDPC decoders," *IEEE J. Solid-State Circuits*, vol. 43, no. 8, pp. 1835–1845, Aug. 2008.

[9] T. Mohsenin, D. N. Truong, and B. M. Baas, "A low-complexity message-passing algorithm for reduced routing congestion in LDPC decoders," *IEEE Trans. Circuits Syst. I Reg. Papers*, vol. 57, no. 5, pp. 1048–1061, May 2010.

[10] F. Angarita, J. Valls, V. Almenar, and V. Torres, "Reduced-complexity min-sum algorithm for decoding LDPC codes with low error-floor," *IEEE Trans. Circuits Syst. I*, vol. 61, no. 7, pp. 2150–2158, Jul. 2014.

[11] S. Sharifi Tehrani, A. Naderi, G.-A. Kamendje, S. Hemati, S. Mannor, and W. J. Gross, "Majority-based tracking forecast memories for stochastic LDPC decoding," *IEEE Trans. Signal Process.*, vol. 58, no. 9, pp. 4883–4896, Sep. 2010.

[12] F. Leduc-Primeau, A. J. Raymond, P. Giard, K. Cushon, C. Thibeault, and W. J. Gross, "High-throughput LDPC decoding using the RHS algorithm," in *Proc. Conf. Des. Archit. Signal Image Process.*, Oct. 2012, pp. 1–6.

[13] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, "Gradient descent bit flipping algorithms for decoding LDPC codes," *IEEE Trans. Commun.*, vol. 58, no. 6, pp. 1610–1614, Jun. 2010.

[14] G. Sundararajan, C. Winstead, and E. Boutillon, "Noisy gradient descent bit-flip decoding for LDPC codes," *IEEE Trans. Commun.*, vol. 62, no. 10, pp. 3385–3400, Oct. 2014.

[15] K. Cushon, S. Hemati, C. Leroux, S. Mannor, and W. J. Gross, "High-throughput energy-efficient LDPC decoders using differential binary message passing," *IEEE Trans. Signal Process.*, vol. 62, no. 3, pp. 619–631, Feb. 2014.

[16] K. Cushon, P. Larsson-Edefors, and P. Andrekson, "Energy-efficient soft-decision LDPC FEC for long-haul optical communication," in *Proc. Eur. Conf. Opt. Commun.*, Sep. 2015, pp. 1–3.

[17] L. Dolecek, Zhengya Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolic, "Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 181–201, Jan. 2010.

[18] D. Chang *et al.*, "FPGA verification of a single QC-LDPC code for 100 Gb/s optical systems without error floor down to BER of 10^{-15}," in *Proc. Opt. Fiber Commun. Conf. Expo. Nat. Fiber Opt. Eng. Conf.*, Mar. 2011, pp. 1–3.

[19] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[20] D. J. C. MacKay. (2002) "Gallager code resources," [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/CodesFiles.html

[21] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 638–656, Feb. 2001.

[22] E. Cavus, C. L. Haymes, and B. Daneshrad, "Low BER performance estimation of LDPC codes via application of importance sampling to trapping sets," *IEEE Trans. Commun.*, vol. 57, no. 7, pp. 1886–1888, Jul. 2009.

[23] M. Yang, W. E. Ryan, and Yan Li, "Design of efficiently encodable moderate-length high-rate irregular LDPC codes," *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 564–571, Apr. 2004.

[24] Zongwang Li, Lei Chen, Lingqi Zeng, S. Lin, and W. H. Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," *IEEE Trans. Commun.*, vol. 54, no. 1, pp. 71–81, Jan. 2006.

[25] Xilinx Inc., *7 Series FPGAs Overview*, May 2015, v1.17.

[26] G. Marsaglia, "Xorshift RNGs," *J. Statist. Softw.*, vol. 8, no. 1, pp. 1–6, 2003.

[27] S. Dave, L. Esker, Fan Mo, W. Thesling, J. Keszenheimer, and R. Fuerst, "Soft-decision forward error correction in a 40-nm ASIC for 100-Gbps OTN applications," in *Proc. Opt. Fiber Commun. Conf. Expo. Nat. Fiber Opt. Eng. Conf.*, Mar. 2011, pp. 1–3.

[28] D. A. Morero, M. A. Castrillon, A. Aguirre, M. R. Hueda, and O. E. Agazzi, "Design tradeoffs and challenges in practical coherent optical transceiver implementations," *J. Lightw. Technol.*, vol. 34, no. 1, pp. 121–136, Jan. 2016.

[29] Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolic, "An efficient 10GBASE-T ethernet LDPC decoder design with low error floors," *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 843–855, Apr. 2010.

[30] L. M. Zhang and F. R. Kschischang, "Staircase codes with 6% to 33% overhead," *J. Lightw. Technol.*, vol. 32, no. 10, pp. 1999–2002, May 2014.

[31] C. Häger, A. Graell i Amat, H. D. Pfister, A. Alvarado, F. Brännström, and E. Agrell, "On parameter optimization for staircase codes," in *Proc. Opt. Fiber Commun. Conf. Exhib.*, Mar. 2015, pp. 1–3.

Authors' biographies not available at the time of publication.