CHALMERS
UNIVERSITY OF TECHNOLOGY

Energy-Efficient High-Throughput VLSI Architectures for Product-Like Codes

Citation for the original published paper (version of record):
Fougstedt, C., Larsson-Edefors, P. (2019). Energy-Efficient High-Throughput VLSI Architectures for Product-Like Codes. Journal of Lightwave Technology, 37(2): 477-485.
http://dx.doi.org/10.1109/JLT.2019.2893039

(article starts on next page)

Energy-Efficient High-Throughput VLSI Architectures for Product-Like Codes

Christoffer Fougstedt, *Student Member, IEEE* and Per Larsson-Edefors, *Senior Member, IEEE*

***Abstract*—Implementing forward error correction (FEC) for modern long-haul fiber-optic communication systems is a challenge, since these high-throughput systems require FEC circuits that can combine high coding gains and energy-efficient operation. We present VLSI decoder architectures for product-like codes for systems with strict throughput and power dissipation requirements. To reduce energy dissipation, our architectures are designed to minimize data transfers in and out of memory blocks, and to use parallel non-iterative component decoders. Using a mature 28-nm VLSI process technology node, we showcase different product and staircase decoder implementations that have the capacity to exceed 1-Tb/s information throughputs with energy efficiencies of around 2 pJ/bit.**

I. Introduction

FORWARD error correction (FEC) is indispensable to meet the increasing capacity demands in fiber-optic communication systems [1]. With throughput demands approaching 1 Tb/s, power and energy dissipation aspects are becoming critical to FEC implementations, especially when error-correction performance in terms of coding gain needs to be high. In a recent roadmap [2], the strict power constraints imposed by future systems for 400G and beyond are expected to be a major challenge in the implementation of FEC circuitry. It is well known that the highest coding gain is obtained using soft-decision (SD) decoding in which all channel information is harnessed. SD decoding, however, does not as naturally lend itself to high-throughput very-large-scale integration (VLSI) implementations as hard-decision (HD) decoding does [3]. In this paper, we address the challenge to implement VLSI systems for very-high-throughput decoders and, as a consequence, we choose to focus on HD decoding.

For fundamental reasons HD decoding is somewhat limited in terms of coding gain in comparison to SD decoding, but there are HD schemes that can deliver relatively high coding gains [4]. Consider, e.g., product codes [5] for which two component codes are combined into one longer block-length code, which has higher error-correcting capability and which is decoded iteratively: In each iteration, decoding is successively done for each of the two shorter component codes, and as we increase the number of iterations we improve coding gain. If the coding gain offered by product codes is not sufficient, we can instead consider staircase codes [6], which are a spatially-coupled generalization of product codes. By virtue of the iterative windowed decoding of the connected component codes, the coding gain can be substantially increased over product codes, at a cost in VLSI implementation complexity.

Since throughput aspects are central in this work, it is worthwhile to note that the structure of product and staircase codes lends itself to block-parallel high-throughput decoding. Because of this property, staircase codes have recently received significant attention within the fiber-optic communication research community as a promising alternative for power-constrained applications: Prior art includes staircase-code optimization, both as stand-alone codes [7], [8] and in concatenated schemes [9], [10], but to the best of our knowledge, no VLSI implementations have been published in the open literature, save for our own previous work [11].

At the core of our product and staircase decoders we find non-iterative component decoders, which realize bounded-distance decoding of shortened binary Bose-Chaudhuri-Hocquenghem (BCH) codes [12] with error-correction capabilities in the range of 2–4. These component decoders are fully parallel and strictly feed-forward, which means that internal state registers can be avoided. As will be shown, this is key to high throughput and low latency. We will, e.g., showcase VLSI implementations of a number of staircase codes that are relevant for fiber-optic communication systems [7]. The implementations are capable of achieving in excess of 1-Tb/s information throughput, which is significantly higher than those of currently published state-of-the-art FEC implementations [13]–[17]. While high throughput requirements typically make power dissipation a serious design concern, this is not the case for our decoders, which dissipate only 1.3–2.4 W (or around 2 pJ/bit), depending on configuration and assumed input bit-error rate.

II. BCH, Product and Staircase Codes

Before we describe the VLSI architectures and implementations of product and staircase decoders, we will briefly introduce concepts pertaining to the FEC codes used: As component code, we use BCH(n, k, t), where n is the block length, k is the number of useful information bits, and t is the number of bit errors that the code can correct. Given that $GF(2^m)$ is the Galois field in which computations are performed, the BCH code parameters are related as $n = 2^m - 1$ and $n - k = mt$. In addition to the definitions above, the code rate, which is the proportion of a block that contains useful information, is defined as $R = \frac{k}{n}$, while the code overhead is defined as $\mathrm{OH} = \frac{n-k}{k} = \frac{1}{R} - 1$.

This work was financially supported by the Knut and Alice Wallenberg Foundation.

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

Part of this work was presented at the Optical Fiber Communication Conference, San Diego, CA, USA, 2018.

TABLE I
PRODUCT CODE PARAMETERS

	Product code overhead		
	20 %	**25 %**	**33 %**
$(n_s, k_s, t = 2)$	-	-	$(120, 104)$
$(n_s, k_s, t = 3)$	$(309, 282)$	$(255, 228)$	$(180, 156)$
$(n_s, k_s, t = 4)$	$(412, 376)$	$(340, 304)$	$(240, 208)$

BCH codes can be shortened to allow for more flexibility in terms of code rate, i.e., this is a tradeoff between coding gain and information throughout. Shortening means a number of information-bit positions are fixed to zero and never transmitted, with the result that the code overhead is increased from the initial non-shortened BCH code. Hereon, we denote the block length and the number of information bits in the shortened codes as $n_s = n - s$ and $k_s = k - s$, respectively, where s is the number of removed bits.

In the general case, the product code is constructed out of BCH(n_{s1}, k_{s1}, t_1) and BCH(n_{s2}, k_{s2}, t_2). The code rate of the product code is defined as $R = \frac{k_{s1}}{n_{s1}} \cdot \frac{k_{s2}}{n_{s2}}$, and the resulting minimum distance is $t_1 \cdot t_2$. In this paper, each product code uses two shortened BCH codes that are identical. We consider decoder implementations with OH=20–25 %, which is achieved by varying the shortening, s. Table I shows the component-code parameters used for our product decoders. The 33 %-OH product codes are based on shortened 255-bit BCH codes, whereas the 25 % and 20 % codes are based on shortened 511-bit BCH codes. Note that the $t = 2$ code, due to its excessively high error floor (see Section VI-A), is only considered as reference for the 33 % case.

For the staircase decoders, we use BCH$(324, 297, 3)$ and BCH$(432, 396, 4)$ component codes. In contrast to the less complex product decoders, we will not vary the code overhead in our explorations of staircase decoders because of excessive design and simulation run times. With each block in the staircase representing an $n/2$-by-$n/2$ matrix, where n is the block length of the component code, we can define the staircase code rate as $R = 1 - \frac{k}{n/2}$ [6]. Here, the shortened BCH codes presented above give an overall staircase code rate $R = 0.83$, which corresponds to an overhead of 20 %.

III. COMPONENT DECODERS

Since the component decoders are central to efficient VLSI implementation of product-like codes, we will first introduce the BCH component decoders, with emphasis on the key equation solver. Variants of these decoders have been used in our previous work: We described 1- and 2-error correcting decoders in [18], while more advanced 3- and 4-error correcting implementations were used (but not described) in [11].

A typical BCH decoder employs syndrome calculation, the Berlekamp-Massey (BM) algorithm (to find the error-location polynomial), and Chien search (to find the errors). Different optimizations of the BM algorithm, such as the simplified inverse-free BM (SiBM) algorithm [19], can improve the implementation, however, a fundamental problem is that conventional BM implementations are iterative and require at least t clock cycles to complete their operation.

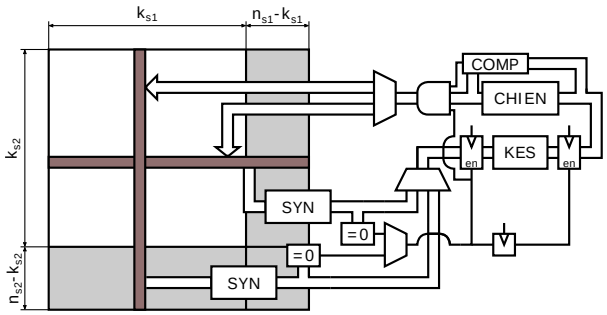


Fig. 1. BCH component decoders used in a product-decoder architecture.

Aiming for high throughput and low latency, we have developed fully parallel BCH component decoders based on the direct-solution Peterson algorithm [20]. Fig. 1 shows a product-decoder architecture in which the BCH component decoders—comprising SYN, KES (see Section III-A) and CHIEN units—are integrated. (The memory block can just as well belong to a staircase decoder.) The component decoders are non-iterative and strictly feed-forward, thus, simplifying state-machine design and allowing for synchronous clock gating of a component decoder's pipeline. The decoders are implemented using bit-parallel polynomial-base $GF(2^m)$ multipliers [21].

Error-free component data are detected after the first stage in the decoder pipeline, i.e., the syndrome calculation stage. To save power, we use several techniques: If all syndromes are zero, the pipeline is gated sequentially and a flag is set to allow for memory-block gating. Each column and row in the product/staircase code uses separate syndrome calculation units to reduce logic signal switching (see Section IV), while the KES and Chien search units are shared between a row/column pair.

The component codes are shortened to yield the desired overall code overhead and since all the removed bits are fixed to zero, the corresponding hardware in the fully parallel syndrome calculation and Chien search can be removed. The number of found errors after the Chien search is compared—in the COMP unit—with the degree of the error-locator polynomial, and the result is discarded if not equal. Component-code miscorrections may induce errors in the non-existing part of the code (which was removed during shortening); this comparison, thus, reduces the probability of miscorrections, improving overall bit-error rate (BER) performance.

A. Key-Equation Solver (KES)

The key-equation solver (KES) unit calculates the error-locator polynomial from the syndromes obtained in the SYN unit. The error-locator polynomial can be expressed as

$$\Lambda(x) = \Lambda_t x^t + \cdots + \Lambda_2 x^2 + \Lambda_1 x + \Lambda_0 \qquad (1)$$

where Λ_n is the error-locator polynomial coefficients calculated in the KES unit. For the case of $t = 2$, the polynomial is given as [18]

$$\Lambda(x) = (S_3 + S_1^3)x^2 + S_1^2 x + S_1 \qquad (2)$$

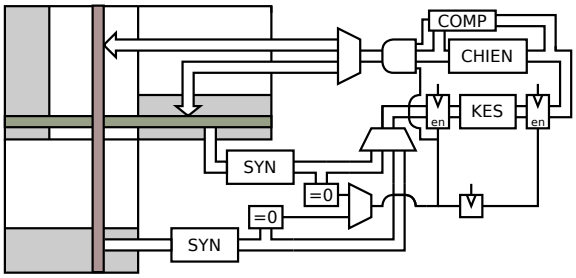


Fig. 2. A 3-block staircase memory array supported by component decoders.

where S_i are the syndromes.

For the cases of $t = 3, 4$, we modify the equations given in [22] to remove inversions [23]. The resulting polynomial coefficients for $t = 3$ are

$$\begin{aligned} \Lambda_0 &= S_1^3 + S_3 \\ \Lambda_1 &= \Lambda_0 S_1 \\ \Lambda_2 &= S_1^2 S_3 + S_5 \\ \Lambda_3 &= \Lambda_0^2 + S_1 \Lambda_2. \end{aligned} \tag{3}$$

If all the resulting coefficients are 0, the equation is incorrect, and at most one error has occurred. In this case, the equation is replaced with

$$\Lambda(x) = S_1 + 1. \tag{4}$$

For $t = 4$, the resulting polynomial coefficients are

$$\begin{aligned} \Lambda_0 &= S_1(S_1^5 + S_5) + S_3(S_1^3 + S_3) \\ \Lambda_1 &= S_1 \Lambda_0 \\ \Lambda_2 &= S_1(S_1^7 + S_7) + S_3(S_1^5 + S_5) \\ \Lambda_3 &= S_1^4 S_5 + S_1^2 S_7 + S_3(S_1^6 + S_3^2) \\ \Lambda_4 &= S_1^3(S_1^7 + S_7) + S_3(S_1^7 + S_1 S_3^2 + S_7) \\ &\quad + S_5(S_1^5 + S_1^2 S_3 + S_5). \end{aligned} \tag{5}$$

In this case, if the coefficients are all 0, two possibilities need to be addressed: If $S_1 = 0$, then no errors have occurred. If not, the equation is replaced with Eq. 2.

IV. Decoder Architecture Overview

After initially moving the received bits into the memory block closest to the channel, product and staircase codes are decoded using an iterative scheme. For product decoders, an iteration consists of two phases: With reference to Fig. 1, first all rows are decoded and errors are corrected, after which all columns are decoded and errors are corrected. This procedure is straightforwardly repeated for a given number of iterations.

In staircase decoding, the iteration scheme is more complex in that each component code covers two spatially coupled blocks, as shown in the simplified decoder configuration in Fig. 2 which operates on a window of 3 blocks. Additional component decoders can be chained over the decoder window, as shown for the more useful 5-block decoder configuration in Fig. 3. Similar to the product decoding above, the staircase algorithm [6] entails decoding of rows followed by columns: After channel data have been moved into memory block 1, parallel decoding of all rows and columns of that block is

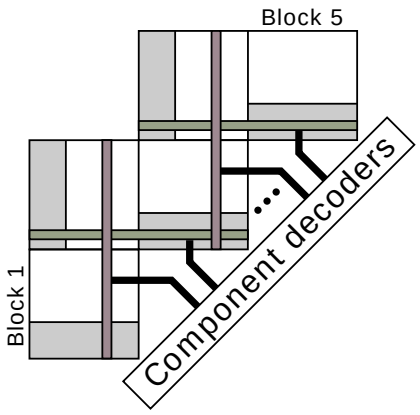


Fig. 3. A 5-block staircase memory array with attached component decoders. The lower block (Block 1) is the memory block closest to the channel.

followed by decoding of all rows and column in the *next memory block*. Successively all memory blocks get decoded and once this is completed, a new decoding phase commences in memory block 1. This process is repeated for a given number of iterations and once all iterations are completed, the data of memory block 1 are moved to memory block 2 and new channel data are moved into memory block 1, thus, shifting the decoding window. We now perform syndrome recomputation in order to avoid separate storage and multiplexers to move syndromes with the data block. (Overall, syndrome computation contributes to at most 10 % of total power dissipation in all considered configurations.)

Since there are no inter-column dependencies nor inter-row dependencies during each half iteration, our staircase decoder architecture first iterates over all rows *in the entire window*, then over all columns in the window. This reduces iteration time and increases throughput compared to decoders that operate on memory blocks sequentially in the decoder window. No significant difference in error-correction performance of staircase codes was found when comparing two MATLAB reference implementations.

As switching power dissipation is proportional to the signal switching, we use replicated syndrome computation units for all decoders. By assigning one syndrome computation unit to rows and another unit to columns, fewer signals switch, since at most t bits are flipped in each row/column per iteration—the majority of the XOR-gates in the syndrome computation are thus kept static—reducing switching power dissipation. Consider the case of correcting one bit: One flipped bit causes at most $\lceil \log_2(n) \rceil$ toggles in each of the XOR-trees for syndrome calculation. The KES and Chien search (the decoder back-end) units are shared between rows and columns.

Row and column syndromes indicate presence of believed errors in corresponding rows and columns. In the product decoder, the codeword is believed to be correct once all syndromes are zero. If this is the case, the memory block is clock gated to save power. The state-based clock gating is somewhat more complex in the staircase memory array: Beside gating each memory block, if no errors are found within that block during component-code decoding, the whole staircase array is gated during component-code decoding and

only clocked on write-back or window shifting.

Since each full row or column is read out and decoded fully block parallel in these decoder architectures, we obtain decoders with very high throughput and low processing latency. Another advantage of using component decoders without any feedback loops is that the control state-machine implementation is straightforward.

The throughput (given a fixed clock rate) of a fully parallel decoder scales quadratically with component-code length, while the area scales slightly faster due to the arithmetic units. A fully parallel design is thus primarily suitable for shorter block-length, higher-OH codes. Although we only consider fully parallel implementations in this work and, thus, focus on higher overhead codes, it should be noted that the strictly feed-forward decoder pipeline structure is beneficial for decoder sharing between component codes, which is a must for lower overhead codes. Since each step is non-iterative, no pipeline stalling is required and, thus, subsequent decoder back-end sharing between L rows (or columns) adds $L-1$ cycles per half iteration. However, while the number of component decoder back-ends required is significantly reduced, a more complicated control state machine and multiple-cycle write-back to the decoder memory is required, which would increase power dissipation.

V. Evaluation Methodology

The decoders were implemented using a hardware description language (VHDL) and a 28-nm fully depleted silicon-on-insulator (FD-SOI) low-leakage cell library, at the slow process corner, a supply voltage of 0.9 V and an operating temperature of 125°C. Cadence Genus [24] was used to synthesize the VHDL implementations using physical wire models. The target clock rate varies somewhat with decoder configuration, from 500 MHz to 600 MHz (see Section VI). The synthesized gate netlists of the decoders were logically verified using simulation in Cadence Incisive with a VHDL testbench generating encoded data transmitted over a binary-symmetric channel.

Post-FEC BER was estimated using the implemented VHDL decoders, simulated using a binary-symmetric channel implemented in VHDL in order to estimate error-correction performance. The obtained post-FEC BER was extrapolated down to 10^{-15} using `berfit` in MATLAB. We want to stress that the obtained net coding gain (NCG) should be seen as *estimations*, since excessive logic simulation run times limit the accuracy of the obtained low-BER extrapolations, especially for complex staircase decoders. However, the resulting estimations are consistent with earlier published results [7], [8], considering algorithmic differences.

A. Decoder Power Dissipation

Energy in VLSI circuits based on CMOS technology is expended when logic signals are switching state. In FEC decoders, power dissipation will therefore be a function of the probability of an error being corrected by each component decoder, giving rise to a significant dependency on input (pre-FEC) BER. In addition, for staircase decoders, component

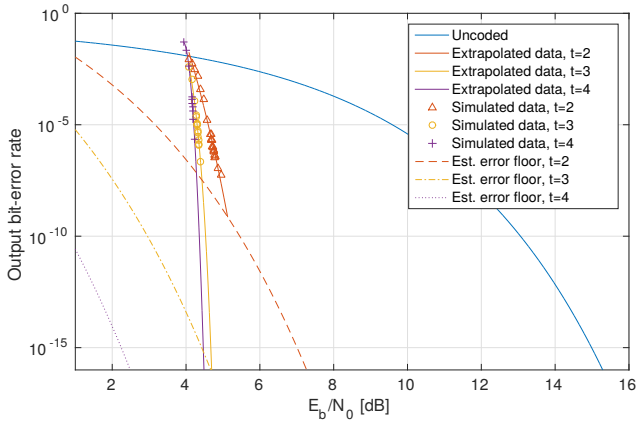


Fig. 4. BER as a function of E_b/N_0 for 33 %-OH product decoders, with simulation data from VHDL implementations.

decoder power dissipation will depend on the spatial location in the decoding window. This dependence on temporal and spatial aspects of logic signal switching makes algorithm complexity a poor metric for power dissipation in VLSI circuits and, thus, rigorous netlist-based simulations are necessary.

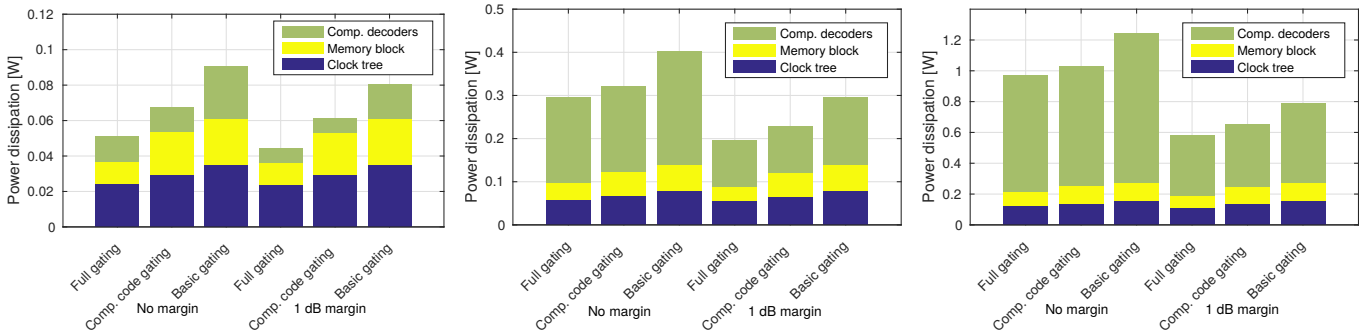
In our evaluation, we used delay-annotated netlists which, e.g., enable us to capture power-dissipating signal glitches that are caused by delay imbalances. The netlists were simulated using a VHDL testbench that generates uniformly-distributed encoded data and can emulate varying pre-FEC BERs, ensuring realistic input-signal statistics. The resulting internal switching activity was backannotated to the netlist and power dissipation was estimated using Cadence Genus, at the typical process corner at a temperature of 25°C and using physical wire models. Since the topology of pre-FEC and post-FEC buffers would be dependent on the overall DSP architecture, our decoders do not have any separate I/O buffers but they use block-parallel inputs and a separate output-block register. Depending on architectural requirements, additional multiplexing/demultiplexing buffers that will increase power dissipation, may be required.

By virtue of the non-iterative component decoders, the proposed architectures become intrinsically very fast. This, in turn, enables us to use low-leakage cell libraries for which static power dissipation is next to negligible. For example, leakage power makes up for less than 1.2 % of the overall power dissipation in a 7-block staircase decoder, which uses $t = 4$ component decoders and performs 6 iterations at a pre-FEC BER of 10^{-2}. With this in mind, we can focus our implementation and analysis on switching power dissipation,

$$P_{sw} = f C_\alpha {V_{DD}}^2, \tag{6}$$

where f is the clock rate, C_α is the switched capacitance, and V_{DD} is the supply voltage.

During normal operation of a general FEC circuit, the actual correction of an error is, with respect to the processing of an information block, relatively rarely performed and this has repercussions on P_{sw}. In the front-end region of a decoder circuit, `FR`, the signal switching activity α tends to be high as blocks of erroneous data are moved inside the VLSI circuit and decoding commences. This makes the switched capacitance $C_\alpha = \sum_{i \in \mathtt{FR}} C_i \alpha_i$ high for the circuit nodes of `FR`. However,



(a) $t = 2$ component code (b) $t = 3$ component code (c) $t = 4$ component code

Fig. 5. Power dissipation of the 33 %-OH decoders at the 10^{-15} threshold and at a 1-dB margin. Note the dramatically varying scales of the y-axes.

the back-end of the decoder is only active if errors are found and need to be corrected and switching activity is, thus, lower. Similarly, for iterative decoders, switching activities are higher for the first iterations than for later iterations, because errors are being successively corrected.

Considering that switching activities impact total power dissipation, stopping the clock—referred to as clock gating—for back-end circuit regions can be a very useful technique. This has an effect both on the clock, which is otherwise toggling in every cycle with $\alpha = 1$, and the logic signals, and is effective for designing energy-efficient FEC circuits. Clock power was estimated using the clock-tree power estimation feature in Cadence Genus as well as using Synopsys PrimeTime [25].

VI. Results

Although the decoder memory architecture is somewhat different for our product and staircase decoders, they employ component decoders in the same manner. We will therefore first investigate design and implementation aspects of product decoders, with a focus on variations brought about by choice of component codes. Then we will extend the scope of our exploration to staircase decoders and design aspects of such windowed decoders.

A. Product Decoder Results

We will first present an analysis geared towards VLSI implementation aspects, such as clock gating, for 33 %-OH product decoders using different component decoders ($t = 2$–4). Then we will extend the analysis to the system level and consider three levels of code overhead (20 %, 25 %, and 33 %) for the configurations with the highest coding gains ($t = 3, 4$).

1) Varying error-correction capability (t): Fig. 4 shows the estimated BER performance, extrapolated from VHDL simulation of 33 %-OH product decoder implementations. The error floors of the codes are estimated as in [26]. While the $t = 2$ code performs relatively well in the waterfall region, it suffers from a high error floor.

Table II summarizes the implementation data for the 33 %-OH product decoders. The NCG parameter is defined as the improvement in signal-to-noise ratio (SNR) over an uncoded transmission, to achieve a threshold which is equal to a post-FEC BER of 10^{-15}. For $t = 3, 4$, the extrapolated BER is used, whereas in the case of $t = 2$, the estimated error floor in Fig. 4 is used to determine the 10^{-15} threshold. Since the decoders are fully block parallel, all implementations have the same block-decoding latency.

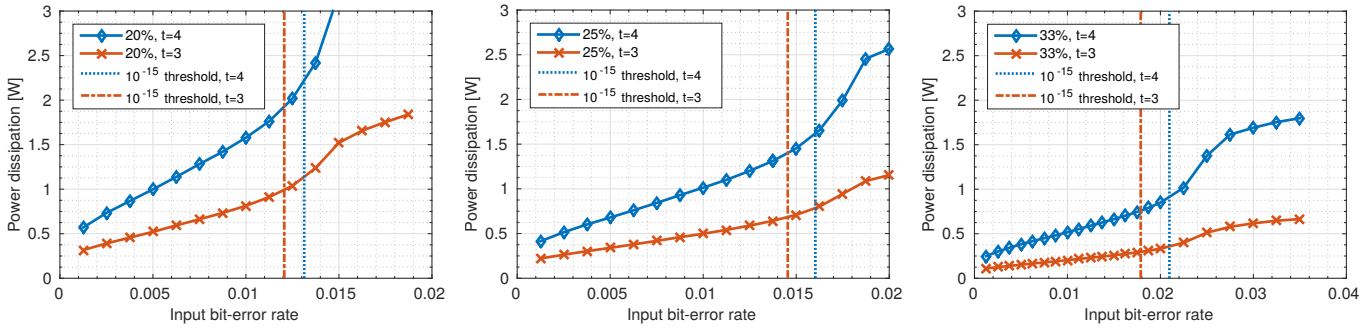
TABLE II
Implementation Data, 33 %-OH Product Codes

	$t = 2$	$t = 3$	$t = 4$
Cell area (mm^2)	0.70	2.23	5.38
Throughput (Gb/s)	203	456	811
Block-decoding latency (ns)	53.3	53.3	53.3
Estimated net coding gain (NCG) (dB)	(8.0)	10.3	10.6
Power dissipation @ threshold (mW)	51.2	297.0	916.4
Power dissipation @ 1-dB margin (mW)	44.3	196.0	551.5

Since the input (pre-FEC) BER impacts the switching events of the decoder (as discussed in Section V-A), power was estimated for two different cases in Table II: Either the product decoders operate at the estimated 10^{-15} post-FEC threshold or with a 1-dB margin to this threshold. The power dissipation is clearly affected by the input BER, especially in the case of the high-NCG $t = 3, 4$ implementations.

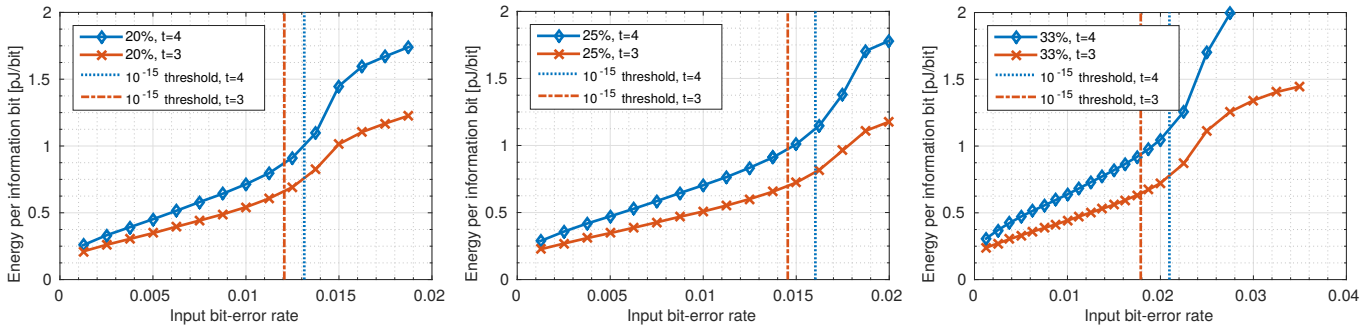
Fig. 5 presents a power dissipation breakdown for three types of clock gating (see Section V-A). All decoders employ *Basic gating* for which non-active registers are clock gated based on the current product-decoder state. We also consider the case of synchronously-gated component decoders (*Comp. code gating*) (see Section III) as well as *Full gating* where, in addition to the component-decoder gating, the memory block (Fig. 1) is gated if all syndromes are zero. As shown in Fig. 5, both memory-block gating and synchronous decoder-pipeline gating are effective in reducing power dissipation. Employing both gives a significant reduction in overall power dissipation and since the different gating schemes entail similar circuit overheads, we will consequently from now on use *Full gating* for all implementations.

2) Varying code overhead: Fig. 6 and 7 show the power dissipation and energy dissipation per information bit as a function of input BER, for the considered 20 %-, 25 %-, and 33 %-OH implementations. The product decoders based on $t = 3$ achieve sub-pJ/bit operation when the decoder input



(a) 20 % OH (b) 25 % OH (c) 33 % OH

Fig. 6. Power dissipation as a function of input BER for product decoders with 20–33 % code overhead.



(a) 20 % OH (b) 25 % OH (c) 33 % OH

Fig. 7. Energy dissipation per information bit as a function of input BER for product decoders with 20–33 % code overhead.

BER is below the decoder 10^{-15} post-FEC threshold. The implementations using $t = 4$ provide higher NCG, at a slight increase in energy dissipation. All product decoders are capable of sub-pJ/bit operation when operating at a 0.15-dB margin to the 10^{-15} threshold. Comparing the power dissipation for the 20 %-33 % decoders (Fig. 6a–6c) to the energy per bit (Fig. 7a–7c), it is clear that while power dissipation decreases with the overhead of the codes, the decrease in energy per bit is not as significant; this is because the information throughput reduces with an increasing code overhead.

It is also interesting to note that compared to the $t = 3$ implementations, the power dissipation of the $t = 4$ implementations is a more sensitive function of the input BER. This is due to the higher energy dissipation *per correction* of the more complex $t = 4$ component decoders.

Table III shows the implementation data for the considered $t = 3, 4$ codes. The $t = 4$ 20 %- and 25 %-OH implementations together with the $t = 3$ 20 %-OH implementation all easily attain a throughput in excess of 1 Tb/s. In addition, all product-decoder implementations—even the ones with lower code overhead—have an NCG of > 10 dB.

TABLE III
IMPLEMENTATION DATA, 20–33 %-OH PRODUCT CODES

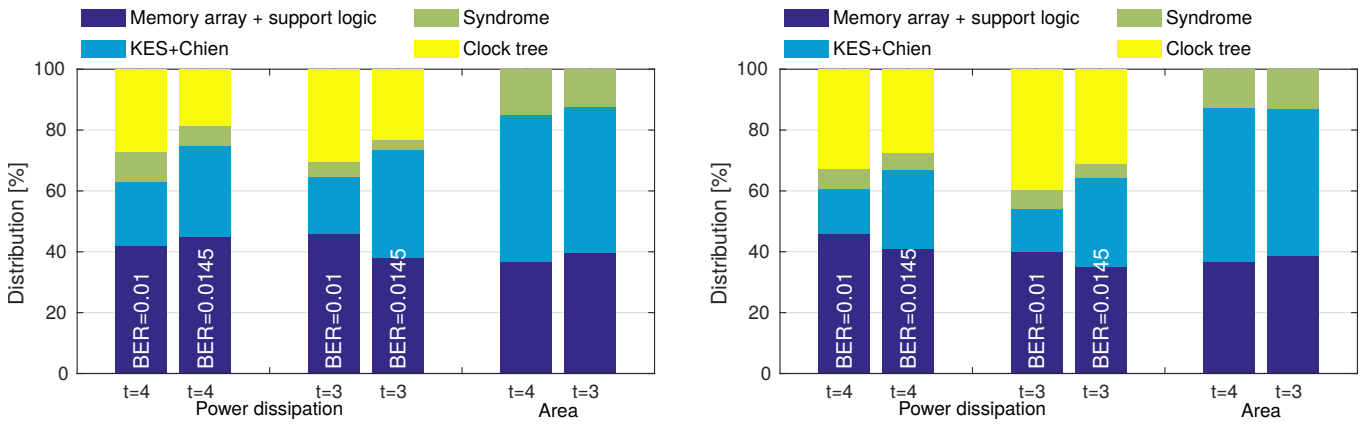
	20 % OH		25 % OH		33 % OH	
	$t = 3$	$t = 4$	$t = 3$	$t = 4$	$t = 3$	$t = 4$
Clock rate (MHz)	600	500	600	500	600	600
Cell area (mm^2)	7.0	16.5	4.86	11.0	2.23	5.38
Throughput (Tb/s)	1.5	2.21	0.98	1.44	0.46	0.81
Latency (ns)	53.3	64	53.3	64	53.3	53.3
Estimated NCG (dB)	10.1	10.3	10.2	10.4	10.3	10.6

B. Staircase Decoder Results

Since we will present data for different staircase decoder configurations, we will from now on use a "shorthand" notation to define how many blocks (bl) there are in the array/window, what error-correcting capability (t) the component codes have, and how many iterations (it) are used. For example, a decoder employing $t = 4$ component codes, performing 6 iterations in a 7-block window will be referred to as a 7-bl t=4 6-it decoder. All implementations have an overall code overhead of 20 % and use a clock rate of 550 MHz.

1) Power dissipation distribution: Fig. 8a and Fig. 8b show the distribution of power dissipation for the considered decoder configurations at an input BER of either $1 \cdot 10^{-2}$ or $1.45 \cdot 10^{-2}$ (the latter corresponds approximately to the 10^{-15} threshold). It is clear that the power dissipated and the VLSI circuit area used by the staircase decoder sub-units are not very correlated. A large proportion of the power dissipation can be attributed to the decoder memory array and the clock tree. Estimations indicate that, e.g., in a 5-bl t=3 5-it decoder, approximately 70 % of all power dissipation in memory elements is caused by clocking.

The syndrome computation units contribute only to a small part of overall power dissipation; syndrome recomputation with block shifting is, thus, not of any concern to overall



(a) 5-bl 5-it staircase decoders

(b) 7-bl 5-it staircase decoders

Fig. 8. Power dissipation and area distributions of 5- and 7-block window staircase decoders performing 5 decoding iterations.

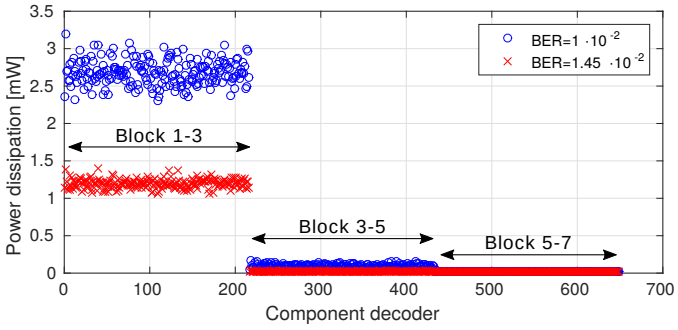


Fig. 9. Power dissipation of component decoder back-ends (KES+Chien) as a function of spatial placement in a staircase window and input BER.

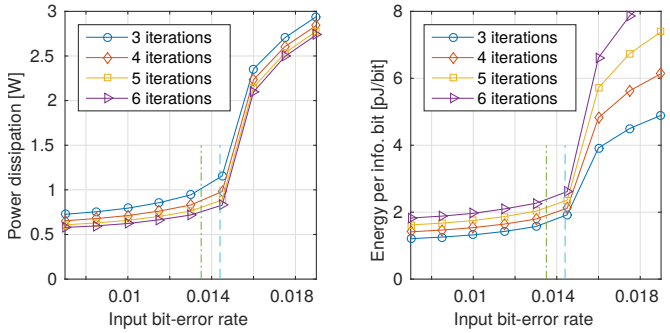


Fig. 10. Power and energy dissipation as a function of input BER for a 7-bl t=3 staircase decoder. Note the order of the curves. The dashed lines show the approximate threshold range for the considered decoder configurations.

power efficiency. The clock tree is estimated to contribute a significant amount of overall power dissipation as it is constantly toggling and, thus, causes large switching power dissipation. The early clock-tree power estimation in Cadence Genus does not provide any area estimate of the clock tree. However, clock buffers in the tree represent less than 1% of the total number of cells even in the design with the largest fraction of clock-power dissipation (which is the 7-bl t=3 configuration), and thus the area contribution of the clock tree can be assumed to be insignificant.

2) Spatial coupling effects: In contrast to the product decoders, the error rate of a processed block in a staircase decoder is not only a function of the input BER and the number of previously performed iterations, but also of its current spatial placement in the staircase window. Being hardware units located far from the channel input, the KES and Chien search units of the component decoders constitute the staircase decoder *back-end*. Showing the back-end power dissipation for all component decoders, sorted after spatial placement in a 7-bl t=4 4-it decoder, Fig. 9 illustrates clearly that power dissipation is a sensitive function of where a component decoder is placed and when it is used. The essence of this figure is that the power dissipation is highly dependent on the number of errors occurring in a memory array block: The major part of the power is dissipated in the component decoders that operate on the blocks that are located closest to the channel input. In addition, there is a strong dependency on input BER.

The probability of decoding an error decreases quickly as data blocks are moved through the decoding window. Accordingly, the power dissipation of the component decoders located deeper in the window decreases. It should be noted that the probability of an error propagating all the way through to the final memory array block is vanishingly small. Thus, sharing of component decoder back-ends deeper in the window may significantly reduce the overall area. This design option, however, is not further explored in this paper.

3) Energy efficiency and power dissipation: Since the component decoder's switching power depends strongly on input BER, the total decoder power dissipation is also heavily dependent on input BER. Fig. 10 shows the power and energy dissipation of a 7-block staircase decoder as a function of input BER. Here, the *average* power dissipation decreases as the number of iterations increases, since each iteration reduces the number of errors in the blocks; errors which would otherwise activate the component decoders.

Energy and power are terms that sometimes are used interchangeably, but the information in Fig. 10 is a good illustration on how energy efficiency and average power dissipation trends differ. As the information throughput decreases with an increasing number of iterations, the energy per information

TABLE IV
EVALUATION RESULTS FOR 5-BLOCK STAIRCASE DECODERS

		$t=3$				$t=4$			
Iterations		3	4	5	6	3	4	5	6
Cell area (mm^2)		7.37				18.37			
Throughput (Gb/s)		601	463	376	317	1069	823	668	563
Block-decoding latency (ns)		181.8	236.3	290.9	345.4	181.8	236.3	290.9	345.4
Estimated NCG (dB)		10.0	10.1	10.3	10.3	10.4	10.5	>10.5	>10.5
Power dissipation (W)	BER=$1 \cdot 10^{-2}$	0.601	0.534	0.491	0.463	1.298	1.132	1.028	0.955
	BER=$1.45 \cdot 10^{-2}$	1.007	0.818	0.729	0.676	1.870	1.573	1.407	1.294
Energy per info. bit (pJ/bit)	BER=$1 \cdot 10^{-2}$	1.00	1.15	1.31	1.46	1.21	1.38	1.54	1.70
	BER=$1.45 \cdot 10^{-2}$	1.67	1.77	1.94	2.13	1.75	1.91	2.10	2.30

TABLE V
EVALUATION RESULTS FOR 7-BLOCK STAIRCASE DECODERS

		$t=3$				$t=4$			
Iterations		3	4	5	6	3	4	5	6
Cell area (mm^2)		11.00				26.40			
Throughput (Gb/s)		601	463	376	317	1069	823	668	563
Block-decoding latency (ns)		254.5	330.8	407.3	483.6	254.5	330.8	407.3	483.6
Estimated NCG (dB)		10.3	10.3	10.4	10.4	10.5	>10.5	>10.5	>10.5
Power dissipation (W)	BER=$1 \cdot 10^{-2}$	0.794	0.712	0.658	0.623	1.819	1.629	1.507	1.425
	BER=$1.45 \cdot 10^{-2}$	1.155	0.980	0.887	0.829	2.385	2.068	1.885	1.76
Energy per info. bit (pJ/bit)	BER=$1 \cdot 10^{-2}$	1.32	1.54	1.75	1.97	1.70	1.98	2.26	2.53
	BER=$1.45 \cdot 10^{-2}$	1.92	2.12	2.36	2.62	2.23	2.51	2.82	3.13

bit increases, since the loss of throughput dominates over the slightly decreasing power dissipation.

Fig. 10 also shows how power dissipation increases rapidly if the input BER is too high to maintain the decoding wave-front within the staircase window. When the wave-front is lost, the entire decoder memory array is filled with data containing errors, causing a rapid increase in switching activity as component decoders futilely attempt to perform decoding operations throughout all iterations.

Table IV and Table V present the evaluation results for the 5-block and 7-block staircase decoder, respectively. An information throughput in excess of 1 Tb/s is achieved by both 5- and 7-block decoders when performing three iterations, with an estimated NCG of 10.4 and 10.5 dB, respectively. The block-decoding latency ranges from 181.8–483.6 ns. Considering that the refractive index of silica is roughly 1.46, the longest latency, thus, corresponds to adding 210 m of fiber to the link. The tables also show that the two design dimensions to consider regarding the iterative decoding process—the number of in-place iterations and the number of blocks in the window which are iterated over—give vastly different results in terms of implementation. Since the clock rate is kept constant, increasing the number of blocks in the memory array (and thus the length of the decoded window) increases the NCG without reducing the throughput. The area usage, power dissipation, and the latency are however increased, since the number of memory elements, as well as the in-memory switching activity due to block movements, increase. Increasing the number of performed iterations increases NCG, at the expense of reduced throughput, increased latency and energy per bit, while area remains constant.

The $t = 3$ staircase decoders achieve a higher energy efficiency than the $t = 4$ decoders, however, at the expense of a lower NCG. The $t = 4$ decoders offer, in our opinion, an interesting trade-off in terms of energy dissipation and NCG (assuming that the VLSI area budget is generous), especially considering the throughput.

VII. DISCUSSION

The implemented hard-decision product and staircase decoders can provide very high information throughput at low energy dissipation and are, thus, suitable for future energy-constrained high-throughput systems. A key enabler is the feed-forward component decoders that allow for high throughput, while iteratively operating on a largely static decoder memory. This allows us to avoid the switching activity caused by the data movement in an iteration-unrolled architecture, where the data are constantly moved during processing.

For the configurations and clock rates considered in this work, the staircase decoders can achieve up to > 1 Tb/s information throughput, while the product decoders achieve up to > 2 Tb/s. While we do not explicitly consider supply voltage scaling in this work, it should be noted that switching power

has a quadratic dependency on supply voltage (see Eq. 6) and can be significantly lowered if the supply voltage is reduced. However, this would be at the expense of reducing the maximum clock rate of the circuit and, thus, the throughput.

The decoders are estimated to achieve a coding gain of 10.0–10.6 dB, depending on configuration. Focusing on the estimated NCG of the 20%-OH implementations, the presented staircase decoders are estimated to perform as well as the best performing HD-decoded codes listed in [1], while our product decoders tie for second best. Note again that our NCG estimations should be seen as approximate as low-BER statistics are limited due to long VHDL simulation run times.

There are very few high-throughput decoders published in the open literature. Compared to a recently published hard-decision product decoder [14], our product decoders achieve more than an order of magnitude higher throughput and much lower latency, at the expense of larger area. In comparison to a recently published high-throughput LDPC decoder [15] (soft decision, 18 % OH), our 20%-OH product decoder can achieve more than three times the throughput at comparable areas (assuming a 70 % area utilization of library cells), at a *higher* coding gain ($E_\mathrm{b}/N_0 = 4.6\,\mathrm{dB}$ at a post-FEC BER of 10^{-7} for our 20%-OH product decoder, compared to 4.95 dB [15]).

The spatial coupling of staircase codes enhances the coding gain compared to product codes, but this comes at a cost of an increase in area, latency, and power dissipation. Nonetheless, our staircase decoders dissipate as little as 1–3 pJ/bit, which is making them highly energy efficient.

VIII. Conclusion

We have implemented energy-efficient high-throughput VLSI decoders for product and staircase codes, suitable for future 400G+ power-constrained fiber-optic communication systems. The decoders have been implemented and evaluated using synthesized gate netlists in a 28-nm VLSI process technology, allowing us to consider aspects of energy efficiency and related tradeoffs. Our decoders achieve more than 10-dB net coding gain and can reach more than 1- and 2-Tb/s information throughput, for staircase and product decoders, respectively. The staircase decoders have a block-decoding latency of <483.6 ns, which corresponds to adding 210 m of fiber to the link, while the product decoder latencies are <64 ns. Effective use of clock gating to inhibit signals from switching is shown to significantly reduce energy dissipation of iterative decoders, both in memory blocks and in component decoders. All considered product and staircase decoders are estimated to dissipate less than 2.4 W, demonstrating the viability of high-throughput hard-decision product and staircase decoders.

References

[1] G. Tzimpragos, C. Kachris, I. B. Djordjevic, M. Cvijetic, D. Soudris, and I. Tomkos, "A survey on FEC codes for 100 G and beyond optical networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 209–221, Firstquarter 2016.

[2] F. Kschischang, "Roadmap of optical communications," in *J. Optics*, E. Agrell and M. Karlsson, Eds. IOP Publishing, 2016, ch. 11. Forward error correction, pp. 23–24.

[3] P. Larsson-Edefors, C. Fougstedt, and K. Cushon, "Implementation challenges for energy-efficient error correction in optical communication systems," in *Advanced Photonics 2018*, 2018, p. SpTh4F.2.

[4] B. Li, K. J. Larsen, D. Zibar, and I. T. Monroy, "Over 10 dB net coding gain based on 20% overhead hard decision forward error correction in 100G optical communication systems," in *Eur. Conf. Opt. Commun. (ECOC)*, Sept. 2011.

[5] P. Elias, "Error-free coding," *IRE Trans. Inf. Theory*, vol. 4, no. 4, pp. 29–37, Sept. 1954.

[6] B. P. Smith, A. Farhood, A. Hunt, F. R. Kschischang, and J. Lodge, "Staircase codes: FEC for 100 Gb/s OTN," *IEEE J. Lightw. Technol.*, vol. 30, no. 1, pp. 110–117, Jan. 2012.

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

[8] C. Häger, A. Graell i Amat, H. D. Pfister, A. Alvarado, F. Brännström, and E. Agrell, "On parameter optimization for staircase codes," in *Opt. Fiber Commun. Conf. (OFC)*, Mar. 2015, p. Th3E.3.

[9] L. M. Zhang and F. R. Kschischang, "Low-complexity soft-decision concatenated LDGM-staircase FEC for high-bit-rate fiber-optic communication," *IEEE J. Lightw. Technol.*, vol. 35, no. 18, pp. 3991–3999, Sept 2017.

[10] M. Barakatain and F. R. Kschischang, "Low-complexity concatenated LDPC-staircase codes," *IEEE J. Lightw. Technol.*, vol. 36, no. 12, pp. 2443–2449, June 2018.

[11] C. Fougstedt and P. Larsson-Edefors, "Energy-efficient high-throughput staircase decoders," in *Opt. Fiber Commun. Conf. (OFC)*, Mar. 2018, p. Tu3C.6.

[12] R. Bose and D. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Inf. Control*, vol. 3, no. 1, pp. 68–79, 1960.

[13] F. Paludi, D. A. Morero, T. Goette, M. Schnidrig, F. Ramos, and M. R. Hueda, "Low-complexity turbo product code for high-speed fiber-optic systems based on expurgated BCH codes," in *IEEE Int. Conf. Circuits Syst. (ISCAS)*, May 2016, pp. 429–432.

[14] C. Condo, P. Giard, F. Leduc-Primeau, G. Sarkis, and W. J. Gross, "A 9.52 dB NCG FEC scheme and 162 b/cycle low-complexity product decoder architecture," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 4, pp. 1420–1431, Apr. 2018.

[15] R. Ghanaatian, A. Balatsoukas-Stimming, T. C. Müller, M. Meidlinger, G. Matz, A. Teman, and A. Burg, "A 588-Gb/s LDPC decoder based on finite-alphabet message passing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 2, pp. 329–340, Feb. 2018.

[16] K. Cushon, P. Larsson-Edefors, and P. Andrekson, "Low-power 400-Gbps soft-decision LDPC FEC for optical transport networks," *IEEE J. Lightw. Technol.*, vol. 34, no. 18, pp. 4304–4311, Sept. 2016.

[17] ——, "Improved low-power LDPC FEC for coherent optical systems," in *Eur. Conf. Opt. Commun. (ECOC)*, Sept. 2017, p. M.1.D.5.

[18] C. Fougstedt, K. Szczerba, and P. Larsson-Edefors, "Low-power low-latency BCH decoders for energy-efficient optical interconnects," *IEEE J. Lightw. Technol.*, vol. 35, no. 23, pp. 5201–5207, Dec. 2017.

[19] W. Liu, J. Rho, and W. Sung, "Low-power high-throughput BCH error correction VLSI design for multi-level cell NAND flash memories," in *IEEE Workshop Signal Processing Systems Design and Implementation*, Oct. 2006, pp. 303–308.

[20] W. Peterson, "Encoding and error-correction procedures for the Bose-Chaudhuri codes," *IRE Trans. Inf. Theory*, vol. 6, no. 4, pp. 459–470, Sept. 1960.

[21] A. Reyhani-Masoleh and M. A. Hasan, "Low complexity bit parallel architectures for polynomial basis multiplication over GF(2m)," *IEEE Trans. Comput.*, vol. 53, no. 8, pp. 945–959, Aug. 2004.

[22] H. Okano, "Decoding of triple and quadruple error-correcting BCH codes by direct solving of error-locator polynomials," *Electron. Commun. in Japan (Part I: Communications)*, vol. 64, no. 2, pp. 22–29, 1981.

[23] S. An, H. Tang, and J. Park, "A inversion-less Peterson algorithm based shared KES architecture for concatenated BCH decoder," in *Int. SoC Design Conf. (ISOCC)*, Nov. 2015, pp. 281–282.

[24] *Cadence® Genus®, v. 16.22-s033_1*, Cadence Design Systems, Inc., 2017.

[25] *Synopsys® PrimeTime®, v. M-2017.06-SP3*, Synopsys, Inc., 2017.

[26] J. Justesen, "Performance of product codes and related structures with iterated decoding," *IEEE Trans. Commun.*, vol. 59, no. 2, pp. 407–415, Feb. 2011.