Variable-Rate FEC Decoder VLSI Architecture for 400G Rate-Adaptive Optical Communication

Citation for the original published paper (version of record):
Jain, V., Fougstedt, C., Larsson-Edefors, P. (2019)
Variable-Rate FEC Decoder VLSI Architecture for 400G Rate-Adaptive Optical Communication
Proc. of IEEE Int. Conf. on Electronics, Circuits and Systems: 45-48
http://dx.doi.org/10.1109/ICECS46596.2019.8964930

(article starts on next page)

Variable-Rate FEC Decoder VLSI Architecture for 400G Rate-Adaptive Optical Communication

Vikram Jain, Christoffer Fougstedt, and Per Larsson-Edefors
Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden
vikram.jain@kuleuven.be, chrfou@chalmers.se, perla@chalmers.se

***Abstract*—Optical communication systems rely on forward error correction (FEC) to decrease the error rate of the received data. Since the properties of the optical channel will vary over time, a variable FEC coding gain would be useful. For example, if the channel conditions are benign, lower code overhead can be used, effectively increasing the code rate. We introduce a variable-rate FEC decoder architecture that can operate in several different modes, where each mode is linked to code rate and decoding iterations. We demonstrate a decoder implementation that provides a net coding gain range of 9.96–10.38 dB at a post-FEC bit-error rate of 10^{-15}. For this range, a decoder implemented in a 28-nm process technology offers throughputs in excess of 400 Gbps, decoding latencies below 53 ns and a power dissipation of less than 0.95 W (or 1.3 pJ/information bit).**

I. Introduction

Legacy wireline and wireless communication systems have traditionally been designed to transmit data at a fixed information bit rate, which entailed the use of fixed forward error correction (FEC) code, modulation scheme, and transmission power. In recent years, however, the majority of wireless communication systems being developed are capable of adapting transmission parameters based on channel conditions. While optical links have followed the old approach to use fixed data rates on account of a static transmission medium, there has recently been a gradual shift to a more dynamic approach which can handle varying bandwidth demands in optical communication [1]. At the backbone of this approach are flexible transceivers that have the ability to adapt to the current channel characteristics in order to maximize the spectral efficiency. Constellation shaping, time-domain hybrid modulation formats, and variable-rate FEC codes are examples of approaches used to enable transceiver flexibility [2]. However, there have been concerns that variable-rate FEC codes would need one decoder unit for each supported code [3], [4] causing transceiver cost to increase significantly with flexibility.

One key feature of digital circuits is that we can “program” their functionality simply by changing the state of some logic signals. In this work, we present a VLSI architecture of a high-throughput FEC decoder which *standalone* can flexibly support several different code rates, allowing us to avoid the area-wasting replication of several fixed-rate decoders. The mode of the decoder can be changed instantaneously, with buffering/unbuffering of data blocks being the only source of latency. By increasing the code overhead, the coding gain, which is the improvement in signal-to-noise ratio (SNR) over an uncoded transmission for a certain bit-error rate (BER), can be improved, at the cost of a reduced information throughput.

The VLSI decoder implementation that we will demonstrate comprises twelve modes, which are obtained by varying the overhead from 21.9 to 40 %. Our architecture is based on hard-decision product codes, which are amenable to high-throughput implementations [5]. While the overheads and decoder iterations were selected taking throughput for 400 Gbps and above optical systems into consideration, our VLSI architecture can be extended to other decoder configurations. To the best of our knowledge, this is the first VLSI implementation of variable-rate FEC decoders for very high throughput operation.

II. Background

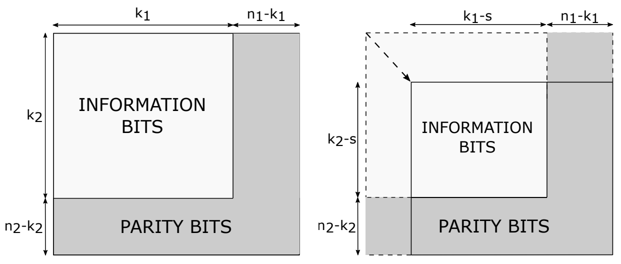
By combining smaller *component* codes to form codes that can provide higher error-correction capability, *product* codes are constructed by encoding information bits row wise, using a row component code, followed by column wise encoding, using column component codes, as shown in Fig. 1a. The twofold encoding over both data and parity gives a resulting minimum distance of $d_1 \cdot d_2$. Product decoding can be implemented using a memory which is iteratively decoded by component decoders. Incoming data bits are loaded into the memory after which decoding and error correction of row and column data take place in an iterative fashion.

Bose-Chaudhuri-Hocquenghem (BCH) codes are a class of random error-correcting cyclic codes expressed by the set of parameters BCH(n, k, t), where n is the block length, k is the number of information bits, and t is the number of errors that can be corrected. Primitive narrow-sense binary BCH codes are defined using a primitive element α of a Galois field, $GF(2^m)$, where m is a positive integer. Here, parameters are related as $n = 2^m - 1$ and $n - k = m \cdot t$, allowing us to define the code rate $R = \frac{k}{n}$ and the code overhead OH $= \frac{n}{k} - 1$.

To vary the code overhead or code rate, we use code shortening, which is the process of substituting zeros for some information bit positions at the encoder; bits which are never transmitted. By shortening our original code, which we call the *mother code*, we increase the overhead and, thus, the coding gain. (Puncturing can be used to increase the code rate [6], but this is not explored in this work.) The shortened codes used here are denoted $n_s = n - s$ and $k_s = k - s$, where s is the number of bits shortened.

A product code can be designed by concatenating two component codes, BCH(n_1, k_1, t_1) and BCH(n_2, k_2, t_2). The product code formed is a $n_1 \times n_2$ matrix, with information bits forming a $k_1 \times k_2$ matrix inside it, as shown in Fig. 1a. The code rate of the resulting product code is the product of the code rate of individual component codes, $R = R_1 \cdot R_2 = \frac{k_1}{n_1} \cdot \frac{k_2}{n_2}$, the overhead is OH $= \frac{n_1 \cdot n_2}{k_1 \cdot k_2} - 1$, and the error-correction capability is $t_1 \cdot t_2$. When using shortened BCH codes to construct the product code, the memory becomes a

$(n_1 - s) \times (n_2 - s)$ matrix as the enclosed information is reduced to a $(k_1 - s) \times (k_2 - s)$ matrix (Fig. 1b).



(a) Baseline memory [7] (b) Shortened-code memory

Fig. 1. Product code memory.

III. Variable-Rate Product Decoder Architecture

A simplified schematic of our variable-rate product decoder architecture is shown in Fig. 2. SYND represents the syndrome calculation unit, KES represents the key-equation solving unit, while CHIEN handles Chien search. The control logic (CONTROL) handles the different modes of decoder operation: We control the code rate and the coding gain by varying only two parameters, viz. the code overhead and the number of iterations. (Varying component code block length is not as good a design option as this would lead to large unused memory portions for modes with small block lengths and a need for varying the degree of the finite-field arithmetic when solving key equations.) The VLSI architecture of the baseline (fixed-rate) product decoder is inspired by our recently published product and staircase decoders [5]. Since our component decoders are fully parallel and have no feedback loops, the introduction of variable-rate features have a limited impact on system throughput.

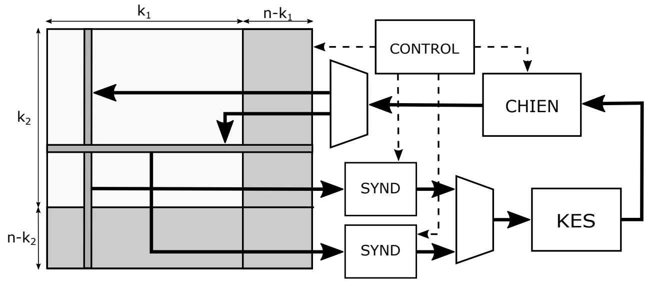


Fig. 2. Our variable-rate product decoder architecture.

A. Product Code Memory

The product code memory stores the received data bits into a matrix and as the code is shortened, correspondingly the memory matrix also reduces as shown in Fig. 1. The bits that are shortened are required to be flushed and gated to avoid interference with downstream decoder computations; this also helps in providing a power-efficient design as no logic signals toggle at these bit positions.

A bit mask of size n is used for gating the memory, with the bits of the mask set to 1 or 0 based on the mode selected, as shown in Fig. 3. The incoming data bits are then ANDed with the bit mask and stored into the product memory as rows and columns. This ensures that the shortened bits are set to 0, preventing power-dissipating toggling of logic signals. Also, the presence of gating of an incoming signal to be stored into the product memory flushes the shortened part of memory when switching between different modes, especially when moving from longer component codes to shorter.

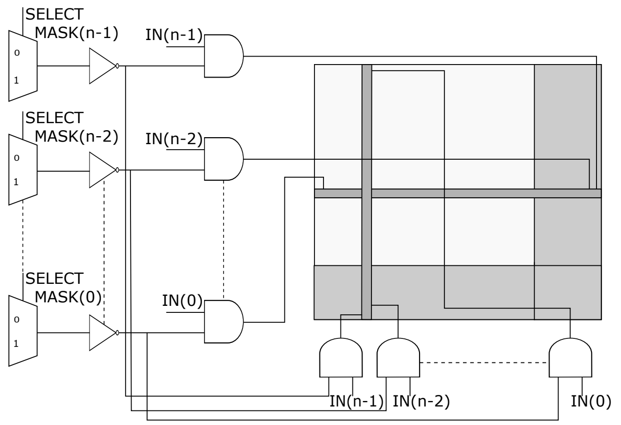


Fig. 3. Product code memory with reconfigurability.

B. Syndrome Calculation

Syndromes are calculated using $S = u \cdot H^T$, where S is the syndrome, u is the received codeword and H is the parity check matrix. The equation can be simplified to form an XOR tree such that each syndrome is a set of XOR operations of the codeword bits at positions where the parity check matrix is 1. When shortening is applied, a number of lower XOR operations (equal to the number of shortened bits) may be removed from hardware. Here, however, the full XOR tree is retained and when shortening is applied, the codeword is shifted to the most significant bits (MSBs). This is achieved by using a set of multiplexers in the SYND unit (Fig. 2). The incoming codeword bits are selected such that the MSB of the shortened code $(n-1-s)$ is always at the MSB of the mother code, while the least significant bits (LSBs) (equal to s) are set to 0 to prohibit signal toggling. The value of s represents the number of shortened bits and $s = 0, s1, s2$ and $s3$ for each of the four modes. The resulting set of bits is passed to the XOR tree to generate a set of $2mt$ syndromes.

C. Chien Search

The error-locator polynomial generated in the KES unit (Fig. 2) using the direct-solution Peterson approach [8] is forwarded to the CHIEN unit where the polynomial is evaluated at all α^x. This is done by using finite-field multipliers (FFMs) to multiply the coefficients with α^x, after which the resulting values are XORed together to form the value of the polynomial. A value of zero for the polynomial represents an error at that position.

The CHIEN unit is designed to be fully parallel to achieve high throughput. Due to its parallel nature, n FFMs are used for every component decoder. To support all modes of operation, all n FFMs are available in the hardware making it necessary to utilize gating to prevent unnecessary computation and power dissipation. (While other less area-consuming schemes to identify roots of error-locator polynomials have been proposed [9], [10], they result in long timing paths which need to be pipelined, degrading throughput and increasing latency.) At the shortened bit (s) positions starting from the LSB, for which computations are not required, the KES coefficients from KES unit are ANDed with enable signals, as shown in Fig. 4. The enable signals are set to 0 or 1 depending on the selected mode of operation. In this way, the inputs to the FFMs are set to 0 and any unnecessary computation is prevented. Finally, the resulting error signal is shifted back to the LSB using the set of multiplexers shown.

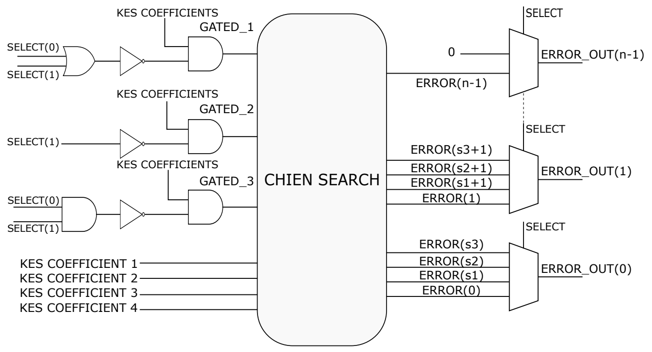


Fig. 4. Reconfigurable Chien search.

IV. 400-Gbps Decoder Modes

To fulfil the combined requirement of high coding gain and operation at 400 Gbps and above, a product code with BCH(255,231,3) component codes is selected as mother code. We use $t = 3$ as this has been shown to provide coding gain above 10 dB [7] at reasonable area and power costs. As coherent optical systems have evolved, the code overhead (OH) has gradually increased, from 7% in early systems, to 15-25% in 100-Gbps systems, and to even higher OHs in published variable-rate schemes [11]. However, considering the very high decoder throughput target and the energy waste of performing digital signal processing on parity information [12], we opt for a limited base OH of 21.9 % ($n = 255$, $k = 231$), 25 % OH ($n = 227$, $k = 203$), 33.1 % OH ($n = 180$, $k = 156$), and 40 % OH ($n = 155$, $k = 131$).

The number of decoder modes can be extended further by varying the number of iterations. The design tradeoff is such that the more iterations, the higher the coding gain and the lower the throughput. For example, more iterations can be combined with a lower-rate code to yield higher coding gain. However, since it has previously been reported that more than five iterations may yield diminishing coding gain returns [7] so we do not consider more iterations than five.

V. VLSI Evaluation Framework

In the context of an application-specific integrated circuit (ASIC), the decoder was implemented using VHDL and simulated for functional verification using Cadence Incisive. One VHDL testbench was used to generate uniformly-distributed data which were encoded using a product encoder. Here, two random number generators (RNGs) based on the uniform procedure in VHDL were used; one for generating uniformly-distributed data, and one to generate bit flips with a probability corresponding to the input SNR. The RNGs have a repetition period of approximately $2.3 \cdot 10^{18}$ for each set of seed values. For BER analysis, we used a VHDL testbench that transmits the all-zero codeword and adds errors with a probability given by the input SNR. The data are then decoded and errors are detected. For each SNR, the testbench runs until 50 erroneous blocks have been found. A common target post-FEC BER is 10^{-15} and, thus, we use this threshold to define the *net coding gain (NCG)*. The error floor, the region of degrading performance where the BER plot does not follow the waterfall model, was estimated using the method proposed by Justesen [13].

The design was synthesized in Cadence Genus to a low-leakage cell library of a 28-nm 0.9-V fully-depleted silicon-on-insulator process technology, assuming slow conditions. Based on an architectural analysis, the target clock rate was set to 610 MHz; stricter timing constraints increase area and power dissipation significantly. Higher clock rate can be easily achieved by switching to a high-performance cell library, however, at the cost of increasing leakage. Using the functional testbench in Cadence Incisive and the synthesized netlist, signal toggling activity information was generated and then back-annotated to the generated netlist in Cadence Genus for the power analysis, under typical conditions. In addition, clock-tree power was estimated using Cadence Genus. The power and energy metrics were evaluated at the same post-FEC BER used in the NCG analysis. Since low-leakage cells are used, leakage is negligible [5].

VI. Results

Fig. 5 shows the output BER as a function of E_b/N_0 for the variable-rate decoder modes that represent the NCG extremes, i.e., 21.9 % and 40 % OH. The estimated coding gain ranges are 0.31, 0.33, and 0.38 dB for three, four, and five iterations (#IT), respectively.

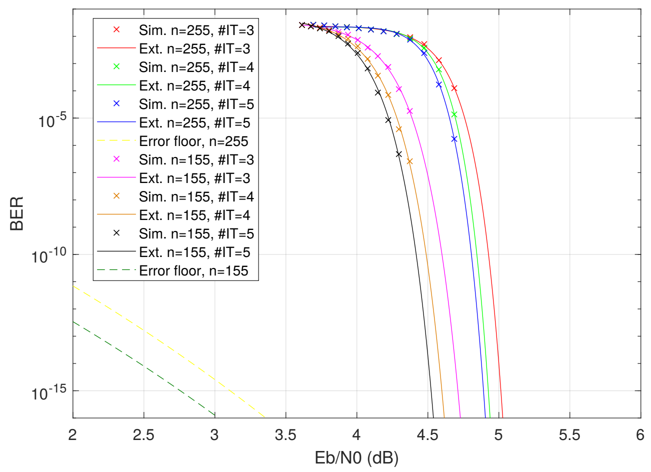


Fig. 5. Output BER as function of E_b/N_0. Simulated data (Sim.) are used for extrapolation (Ext.) in MATLAB with the *berfit* function. E_b/N_0 for an uncoded system stands at 14.99 dB for an output BER of 10^{-15}.

A wider range of 0.5 dB can be achieved if the decoder is operated with three iterations for the base OH (21.9 %) and with five iterations for 40 % OH. However, five iterations with

40 % OH cannot attain the targeted throughput of 400 Gbps; instead assuming the mode of four iteration with 40 % OH, the coding range becomes 0.42 dB.

The coding gain range depends on the block length of the component codes. In our architecture, the minimum coding gain is limited by the overhead of the component code, i.e., BCH(255,231), which is 21.9 %. Conversely, the upper limit of coding gain, i.e., the highest OH, is limited by the constraint of achieving throughputs in excess of 400 Gbps. The coding range could be extended further by utilizing a higher OH, but this has as consequence a reduced throughput. Another alternative to increasing the coding gain range is to utilize longer component codes, e.g., BCH(511,484), whose product code has a base OH of 11.5 %. This, however, would lead to a more complex decoder with higher energy dissipation. Increasing the error-correction capability t would increase the coding gains of individual modes with a skew in the range, at a significant area and power dissipation cost.

Table I presents our post-synthesis netlist results, with latencies below 53 ns and throughputs as high as 1.6 Tbps. Moreover, owing to the use of clock gating, the VLSI implementations are highly energy efficient with a maximum energy per information bit being as low as 1.29 pJ/bit. Variation in iteration count (#IT) provides a very resource-efficient alternative to regulating coding gains at fixed code rates, however, it cannot provide as large range in coding gain obtainable by varying code overhead. As shown, a combination of OH and iteration variation can provide a wider range of operation.

TABLE I
VLSI EVALUATION RESULTS OF VARIABLE-RATE DECODERS

	#IT	Overhead (OH)			
		21.9 %	**25 %**	**33.1 %**	**40 %**
Cell area (mm^2)		8.78			
Code rate, R		0.82	0.80	0.75	0.71
Throughput (Gbps)	3	1628	1257	742	523
	4	1252	967	571	402
	5	1017	785	464	327
Block decoding latency (ns)	3	32.78			
	4	42.61			
	5	52.45			
NCG @ BER 10^{-15} (dB)	3	9.96	10.05	10.16	10.27
	4	10.06	10.14	10.24	10.38
	5	10.08	10.23	10.35	10.46
Power @ BER 10^{-15} (mW)	3	941	822	599	524
	4	830	768	525	461
	5	770	710	479	422
Energy @ BER 10^{-15} (pJ/info-bit)	3	0.58	0.65	0.81	1.00
	4	0.66	0.79	0.92	1.14
	5	0.76	0.90	1.03	1.29

Extra logic circuits are required for a decoder unit to handle several different codes. The variable-rate decoder area is 31 % larger than the baseline (fixed-rate) 21.9 %-OH version [5], which occupies 6.69 mm^2. Comparing decoders for e.g. #IT=4, the variable-rate decoder is, however, found to not dissipate significantly more than baseline decoder's 788 mW.

We also implemented reference decoders based on conventional iterative Berlekamp-Massey (BM) algorithms. The variable-rate decoder area is only 16 % larger than a 21.9 %-OH decoder based on the simplified inverse-free BM scheme [14]–[16], suggesting that the choice of decoder algorithm has a larger impact on the VLSI implementation than the decision to opt for fixed or variable-rate decoders.

VII. CONCLUSION

We have introduced a VLSI architecture for high-throughput variable-rate FEC decoders based on product codes. Using a 28-nm process technology, we presented a decoder implementation that achieves an estimated net coding gain range from 9.96 to 10.38 dB, with a minimum throughput of 400 Gbps and a maximum decoding latency of 53 ns, showing the viability of flexible, yet energy-efficient decoders for high-throughput systems. We found that the circuit overhead, in terms of area and power dissipation, required to handle different modes is limited; replicating several fixed-rate decoders would have a much more adverse effect on FEC circuits.

ACKNOWLEDGMENT

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

REFERENCES

[1] O. Gerstel, M. Jinno, A. Lord, and S. J. B. Yoo, "Elastic optical networking: a new dawn for the optical layer?" *IEEE Commun. Mag.*, vol. 50, no. 2, pp. s12–s20, Feb. 2012.
[2] G. Bosco, "Advanced modulation techniques for flexible optical transceivers: The rate/reach tradeoff," *IEEE J. Lightw. Technol.*, vol. 37, no. 1, pp. 36–49, Jan. 2019.
[3] X. Zhou, L. E. Nelson, and P. Magill, "Rate-adaptable optics for next generation long-haul transport networks," *IEEE Comm. Magazine*, vol. 51, no. 3, pp. 41–49, Mar. 2013.
[4] D. A. A. Mello, A. N. Barreto, T. C. de Lima *et al.*, "Optical networking with variable-code-rate transceivers," *IEEE J. Lightw. Technol.*, vol. 32, no. 2, pp. 257–266, Jan. 2014.
[5] C. Fougstedt and P. Larsson-Edefors, "Energy-efficient high-throughput VLSI architectures for product-like codes," *IEEE J. Lightw. Technol.*, vol. 37, no. 2, pp. 477–485, Jan. 2019.
[6] G. Gho, L. Klak, and J. M. Kahn, "Rate-adaptive coding for optical fiber transmission systems," *IEEE J. Lightw. Technol.*, vol. 29, no. 2, pp. 222–233, Jan. 2011.
[7] 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, p. Tu.6.A.3.
[8] 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.
[9] X. Zhang and M. O'Sullivan, "Ultra-compressed three-error-correcting BCH decoder," in *IEEE Int. Conf. Circuits Syst. (ISCAS)*, May 2018.
[10] D. Kim, I. Yoo, and I. Park, "Fast low-complexity triple-error-correcting BCH decoding architecture," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 6, pp. 764–768, June 2018.
[11] A. L. N. Souza, E. J. M. Ruiz, J. D. Reis *et al.*, "Parameter selection in optical networks with variable-code-rate superchannels," *IEEE J. Opt. Commun. Netw.*, vol. 8, no. 7, pp. A152–A161, July 2016.
[12] P. Larsson-Edefors, C. Fougstedt, and K. Cushon, "Implementation challenges for energy-efficient error correction in optical communication systems," in *Advanced Photonics 2018*, July 2018, p. SpTh4F.2.
[13] J. Justesen, "Performance of product codes and related structures with iterated decoding," *IEEE Trans. Commun.*, vol. 59, no. 2, pp. 407–415, Feb. 2011.
[14] E. Hurtic and H. Lillmaa, "Hard-decision staircase decoder in 28-nm fully-depleted silicon-on-insulator," Master's thesis, Chalmers University of Technology, 2016.
[15] 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 on Signal Processing Systems*, Oct. 2006, pp. 303–308.
[16] M. Yin, M. Xie, and B. Yi, "Optimized algorithms for binary BCH codes," in *IEEE Int. Symp. on Circuits and Systems*, May 2013, pp. 1552–1555.