A configurable FPGA FEC unit for Tb/s optical communication

Andersen, Jakob Dahl; Larsen, Knud J.; Bering Bøgh, Christian; Forchhammer, Søren; Da Ros, Francesco; Dalgaard, Kjeld; Iqbal, Shajeel

Published in:
Proceedings of 2017 IEEE International Conference on Communications

Link to article, DOI:
10.1109/ICC.2017.7996767

Publication date:
2017

Document Version
Peer reviewed version

A Configurable FPGA FEC Unit for Tb/s Optical Communication

Jakob Dahl Andersen, Knud J. Larsen, Christian Bering Bøgh, Søren Forchhammer,
Francesco Da Ros, Kjeld Dalgaard, Shajeel Iqbal
Department of Photonics Engineering
Technical University of Denmark
Kongens Lyngby, Denmark
e-mail: sofo@fotonik.dtu.dk

***Abstract*—Decoding of FEC (forward error correction) for optical communication beyond 1 Tb/s is investigated. A configurable single FPGA solution is presented having configurations supporting bit-rates in the range from 40 Gb/s to 1.6 Tb/s. The design allows for trade-offs of bit-rate, footprint, and latency within the resources of the FPGA. A proof-of-concept lab experiment at 40 Gb/s was conducted and pre-FEC – post-FEC performance validated with simulated results.**

Keywords—HD-FEC; beyond 1 Tb/s; product codes; optical communication

I. INTRODUCTION

Forward error correction (FEC) plays an important role in today's high speed optical communication systems. The purpose of this paper is to design a FEC unit capable of operating up to 1.6 Tb/s (gross) data rate in a single FPGA. The aim is to use the unit for experimental research into high speed optical communication. The design may of course also be used for future standards of *l* x 100 Gb/s data rates, but additional functions are needed to fit into a specific transmission format.

Many results for optical fiber transmission links at 400 Gb/s have been reported, see e.g. in [1], and application of FEC to such high speed links are also reported, [1], [2]. It must be foreseen that several parallel lanes with a lower bit rate could be used, but combined in a single FEC solution. The IEEE 802.3 400G study group published initial thoughts on a single FEC solution [3]. Though the study is for 400G Ethernet, it shows a trend for future high speed links. In this paper we consider a reconfigurable FPGA solution capable of operating as a single FEC structure initially up to 400 Gb/s net data rate in ordinary FPGAs. The interfaces to the FPGAs are assumed to be aggregates of many lanes since modern FPGAs allow no more than on the order of 25 Gb/s interfaces. For higher data rates, the codes may be interleaved and our goal is to have space for the decoders up to 1600 Gb/s in a single FPGA. In addition to the higher speed, the reference bit error rate (BER) for future systems will be stricter. Today the value is 10^{-15}, and a value of 10^{-17} is one of the proposals for 400G Ethernet [4]. It is characteristic for all iterative decoding algorithms that the performance improves sharply around a certain pre-FEC BER, but the performance is then limited by an error floor for lower pre-FEC error rates. The error floor must then of course also be below 10^{-17}-10^{-15}. Here we shall evaluate the Net Coding Gain (NCG) at 10^{-15}, but aim for an error-floor below 10^{-17}.

Improvements in system performance may also be achieved by applying a FEC frame over two [5] or more wavelengths with different error levels or variations over time, e.g. if the FEC threshold can be calculated as an average over the wavelengths, called gain-sharing, rather than applying a worst-case design. For a fixed size FEC frame the absolute latency due to FEC coding may also be reduced by increasing FEC data rate by coding over a super-channel rather than coding each channel independently.

Overviews of FEC for optical communication can be found in [6] and [7]. Most of the recent research work considers soft-decision FEC especially with use of LDPC codes, [2], [6], [7], [8]. In this paper, we focus on hard decision decoding for two reasons: First, soft decisions require a broader interface to the FPGA and this will limit the data rate due to the limited number of transceivers in the FPGA. Second, soft decision algorithms are more complex and it is difficult to reach the goal of beyond 1 Tb/s for a single FPGA FEC decoder unit. Soft decisions would improve performance since there is 1.1-1.3 dB difference in capacity for a binary channel. For modulation systems of higher order than QPSK, soft decisions may be used to reflect the different reliabilities of the received bits. Soft decisions may be a feasible solution in integrated ASIC design, [9]. Several hard-decision FEC codecs have been proposed for 100 Gb/s, all having a NCG around 9 dB with an overhead slightly less than 7%, e.g. [10], [11], and [12]. The latter presents a product code with a more advanced interleaving scheme reducing the number of bits to take into account at the decoding. Staircase codes [13] may also be seen as product codes with an advanced interleaving scheme and with similar performance. Operating with a larger overhead, hard decision schemes that deliver more than 10 dB NCGs are also product code solutions, see e.g. [1]. The performance of product codes will be discussed in the next section.

In Sec. II, the chosen product code and basic principle of decoding and performance analysis is presented. Sec. III presents a scalable decoder design and overview of VHDL implementation. Synthesis results from 40 Gb/s to 1.6 Tb/s are

This work was supported by the DNRF Research Centre of Excellence, SPOC, ref. DNRF123.

presented in Sec. IV and a proof-of-concept lab-experiment at 40 Gb/s is presented in Sec. V.

II. The Product Code and its Decoding

The component code for the product code used in the present FEC unit is an expurgated BCH-code capable of correcting 3 errors and detecting 4 errors as in [12]. The length of the code, n, is maximum 1023, but in this application, the code is shortened a little depending on the bit rate due to interfacing requirements for the FPGA. The actual code parameters are $(n, k) = (1008, 976)$ to $(960, 928)$, where k is the number of information bits. The number of redundancy bits is always $n-k = 32$ and the minimum distance is 8. The code generator polynomial is

$$g(z) = \left(z^{10}+z^3+1\right)\left(z^{10}+z^3+z^2+z+1\right)\left(z^{10}+z^8+z^3+z^2+1\right)\left(z^2+1\right) \quad (1)$$

where the first 3 factors give a primitive BCH code capable of correcting 3 errors and the last factor expurgates this code to obtain the minimum distance of 8. The decoding operates on syndromes, s_i, calculated in the finite field GF(1024) with primitive element α from the received polynomial $r(z)$:

$$s_i = r\left(\alpha^i\right), i = 1,3,5 \quad (2)$$

The BCH code is applied to both rows and columns in a quadratic product code structure, which is named a frame as illustrated in Fig. 1. Thus the overhead ranges from 6.7% to 7.0%. The BCH decoder implementation follows a suggestion by S. Gravano, [14], who reinvented the decoder structures given by Okano and Imai, [15]. The BCH-code is decoded without the expurgation (i.e. the first three factors in $g(z)$) by first computing syndromes, which are used to directly set up the coefficients of the error locator polynomial.

The error positions are found from a modified error locator polynomial through look-up tables avoiding the search for roots in the polynomial. The decoded result is then qualified by the two parity check bits (one for even positions and one for odd positions) defined by the last factor of $g(z)$.

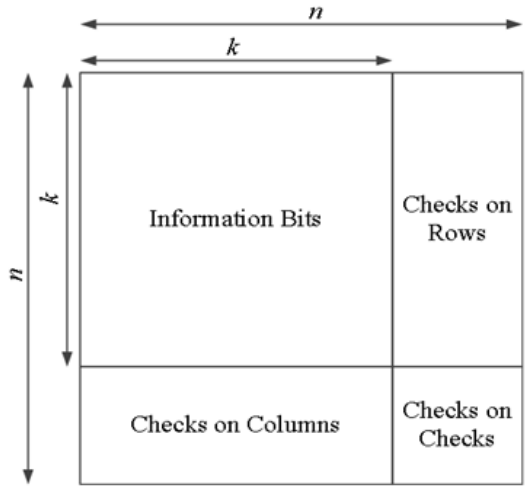

Fig. 1 Frame with product code structure.

Decoding of a frame is performed with hard decisions, first rows, then columns and then rows again and so on. An iteration is defined as decoding the rows followed by decoding the columns. A theory for the performance of such quadratic product codes is found in [16] and [17]. Since the BCH component codes are capable of correcting 3 errors, the theory states that with a high probability, an ideal decoder is successful after many iterations if the average number of pre-FEC bit errors is below 5.14 errors per row or column, i.e. a pre-FEC BER around $5 \cdot 10^{-3}$. This threshold corresponds to an NCG of 9.5 dB, only 0.3 dB less than what is achievable for hard decisions (binary symmetric channel). The actual performance is impeded by a limited number of iterations and the possibility of wrong decoding. If there are more than 3 errors, the BCH decoder may decode to an error pattern corresponding to a codeword with weight 7 or 8. The last factor of the generator polynomial (1) reduces the probability of accepting a wrongly decoded error pattern by 1/4. In [18] the probability for wrong decoding of Reed-Solomon codes is treated and the results also hold for BCH codes. If more than 3 errors occur, the probability of wrong decoding is around 1/3!, and in connection with the two parity checks, the overall probability of wrong decoding is around 1/24 given that more than 3 errors occur. In addition to the threshold where decoding is expected to be successful, an error floor also exists. The error floor could be illustrated by 4×4 errors in a square (also known as a core) which is clearly impossible to decode. These rows and columns of the core do not need to be adjacent, and the error positions could be the intersection of any combination of 4 rows and 4 columns. Thus the probability of frame loss (PFL) due to this type of error constellations at pre-FEC BER p may be estimated by

$$PFL = \binom{n}{4}^2 p^{4\times 4} \quad (3)$$

This may be used as an estimate of the error floor for frame loss at pre-FEC BER p and multiplying by the 16 bit errors gives an estimate the post-FEC error floor BER:

$$Error\_floor(p) = \frac{4^2}{n^2}\binom{n}{4}^2 p^{4\times 4} \quad (4)$$

Errors of a given decoding scheme may also contribute to the error floor. An error floor taking decoding errors into account is given in the Appendix.

III. VHDL Implementation

We have made a VHDL implementation of a decoder for the product code suitable for an FPGA. The implementation is intended not to be vendor or FPGA specific. Only the transceivers and the synchronization part use specific components from the Altera Stratix V FPGA present on our development board. The design is scalable, e.g. both the width of the input and the number of internal BCH decoders working

in parallel can be configured within a number of possible values.

A. Main Decoder Design

First step in decoding of the BCH codes is calculating syndromes (2). Based on the syndromes, the error positions are found by a number of table-look-ups for finding roots in 2nd and 3rd degree polynomials. The approach is described in detail in [13]. The calculations are performed in the finite field GF(1024) based on logarithm and exponential tables. We shall refer to this part of the decoding as a *Gravano decoder*. Such a decoder works on a set of syndromes and the two parity checks described above, and the output is a number (0 to 3) of error positions to be corrected. Corrections are only performed if a valid codeword is found, including the parity checks, and none of the errors are in the part of the codeword that is removed due to shortening the code. The full product code is decoded by iterative decoding, first rows, then columns, and so on. This iterative process goes on as long as the timing restrictions permit.

Although the syndrome calculation is relatively simple it is an advantage only to calculate the syndromes once and then update the syndrome when an error is corrected. For the specific BCH word being corrected, the syndromes are just set to 0, while the syndromes for the codewords in the other dimension have to be read and adjusted according to the error position corrected. Along with the syndromes the actual data frame also has to be corrected. The specific error position has to be read, inverted and written back.

Our decoder design uses a number of Gravano decoders working in parallel. Each decoder receives a stream of syndromes and delivers a stream of corrections. The corrections are placed in a FIFO (first in first out). When there are corrections in the FIFO, these are read and the stored data and syndromes are updated accordingly. The time used to actually correct the errors depends on the number of corrections, but since several row decoders may have corrections to the same column they may also have to wait for each other. It is important that all the corrections from the row decoders are performed before the column decoders are started and vice versa.

B. Latency

The decoder is designed to have a latency of two full frames. The decoder works in frame-slots corresponding to the transmission time for a frame. When the frame is received and stored in RAM, the syndromes for both rows and columns are calculated. In the same frame-slot, the Gravano decoders work on the syndromes for the previous frame making as many corrections as possible in the clock cycles of the frame-slot. Finally, corrected data and parity are read out in a frame-slot. The latency is close to the minimum for the quadratic interleaving used.

C. Scalable Design

High data rates are achieved by receiving a (large) number of input bits in parallel. The decoder is designed such that it can be configured to an input width which is a power of 2. We have used data widths of 128, 256, 512, 1024 and 2048. The output width is identical to the input width. In our design, the input width must divide the full frame size. This is accomplished by shortening the BCH codes slightly. For the above mentioned input widths, the frame sizes are 1008^2, 1008^2, 992^2, 992^2 and 960^2.

The number of iterations for the decoding process depends on the number of clock cycles available and the number of parallel Gravano decoders. With 128-wide input we have $1008^2/128 = 7938$ clock cycles per frame-slot. For 2048 we have 450 clock cycles. The decrease in the numbers of clock cycles for higher data rates can to some extent be compensated by increasing the number of Gravano decoders, which is a parameter in the decoder design. For very high data rates the number of clock cycles may limit the number of iterations to an unacceptable level. As the simulation results show (Fig. 3) at least 3 iterations are required for acceptable performance.

If the required number of iterations cannot be achieved with the present decoder design there are two options. The decoder can be redesigned for a higher latency, i.e. using two frame-slots for the decoding. Alternatively a number of frame decoders with the existing design can be used in parallel. This also gives an increased latency since several encoded frames have to be interleaved before transmission. We aim at the relatively high clock frequency of 312.5 MHz, meaning that 32 bit wide input corresponds to a gross data rate of 10 Gb/s, etc. If this clock frequency is not fully achieved, the decoder will still work, although at a lower data rate.

D. RAM configuration

The main challenge of the implementation is the RAM configuration. The data frame of approximately 1 Mbit must be stored in RAM, but since the bits should be accessed both row-wise and column-wise there is no obvious way to configure the RAM. An obvious solution would be to store one bit at each address, but the full input block has to be stored in one clock cycle.

Our solution is to see the parallel input as a rectangular block within the full two-dimensional codeword. We call such a rectangle a *tile*. For 128 bit input the tile is 8×16, i.e. the first tile is the first 8 bits in the first 16 rows. For larger input data widths the tile sizes are 16×16, 16×32, 32×32 and 32×64. Furthermore, the RAM and part of the control system is divided into *slices* each covering a part of the frame and working in parallel. The number of slices corresponds to the number of columns in a tile. The slices include their share of both the syndrome formers and the syndrome storage. The syndrome formers take a number of bits every clock cycle corresponding to the dimension of the tile. Although the slices provide a simple way of parallelizing the process it still leaves some issues, since no matter how the frame is sliced the corrections will be exchanged between the slices. The bottleneck of this decoder design is thus the distribution of corrections between the RAM slices.

E. Gravano decoder

A fully pipelined Gravano decoder with 18 stages is implemented, i.e. it is capable of receiving a set of syndromes every clock cycle.

F. Synchronization of lanes and frames

To achieve synchronization of both lanes and frames, we include a number of synchronization words. Splitting the connection into a number of lanes really means to have a number of sub-frames transmitted in parallel. Each sub-frame needs frame synchronization which may be achieved by a synchronization word at the top of the sub-frame. Furthermore, since it may not be known which lane is connected to which transceiver the synchronization word is succeeded by a small pointer giving the number of the lane.

When frame synchronization is established for all lanes, the lanes have to be synchronized. This is accomplished by denoting one lane as master and giving this a fixed delay, M. The slave lanes are delayed by FIFOs where the delay can be varied between 0 and $2M$. The delays of these FIFOs are controlled by small state-machines based on the frame start signals from the actual lane and the master lane.

In our lab experiment, we have used 4 lanes with 32 bit synchronization words and 2 bit pointers which adds up to a total of 136 bits. If the number of lanes is increased this can however be a significant number. An obvious solution is only to have synchronization words in some of the frames, although this will increase the initial synchronization time.

IV. Implementation results

We have made the synthesis, place and route etc. from the VHDL code for two Altera FPGAs. The design software used is Quartus Prime Standard Edition software versions 15.1 and 16.0.

In the tables we show the input parameters: the input data width and the number of Gravano decoders. As a result of the synthesis we get the maximum clock frequency, fmax, and the percentage of ALMs (Adaptive Logic Module) and memory blocks used. Based on fmax we obtain the actual maximum gross data rate (including the overhead for parity bits etc.) as the product of fmax and data width. We have also included a suggestion for use of transceivers. The number of iterations depends on the number of clock cycles available, i.e. the data width and the number of Gravano decoders present. The number of iterations is indicated in the last column.

Table I presents selected synthesis results for the Altera Stratix V FPGA present on our development board and used in the lab experiment. The results are for a single frame decoder, but with a varying number of Gravano BCH decoders and without transceivers and synchronization (except for data width 128 which does include synchronization).

Table II presents selected synthesis results for a larger FPGA, this time chosen from Alteras Arria 10 family. These implementations use multiple frame decoders of the type considered in Table I. The maximum clock frequency is not really higher with this FPGA but there is space for several frame decoders working in parallel.

As seen from Table II, we achieve a gross data rate of 1.6 Tb/s with a latency of 10 frames by using 5 parallel frame decoders each having 8 Gravano decoders. For a single frame decoder and 32 Gravano decoders 455 Gb/s is achieved. With the suggested synchronization this becomes 1502 Gb/s and 423 Gb/s, respectively, for the net data rate.

For a single frame decoder as in Table I, the latency is 2 frames, i.e. approx. 2 Mbit. Above 400 Gb/s this is less than 5 µs. At 40 Gb/s, the latency will be 50 µs. Having d frame decoders the latency becomes $2d$ frames, i.e. $2d$ Mbit, but this will be used at higher bit-rates. So for the configurations in Table II, the latency increases slightly from less than 5 µs to approx. 6 µs at 1600 Gb/s. For reference, a latency of 5 µs would be equivalent to the propagation delay introduced by 1 km of standard single-mode fiber. The impact of such a delay would therefore be negligible even for short links.

V. Experiment with 40 G

An experimental set-up for testing the FEC decoder in optical communication using the *Transceiver Signal Integrity Development Kit, Stratix V GX edition* board with the 128 bit input and 2 Gravano decoders configuration from Table I was established in the laboratory as depicted in Fig. 2.

TABLE I. Synthesis results for Altera Stratix V (5SGXEA7N2F40C2 - current board)

Data width	Gravano decoders	Trans-ceivers (in bits)	fmax MHz	Max Gross rate	ALM/mem (%)	No It.
128	2	4*32 bit	339.79	43 G	5/10	7
128	8	4*32 bit	339.21	43 G	8/14	~25
512	8	16*32 bit	322.48	164 G	14/16	6½
1024	8	32*32 bit	323.21	330 G	24/17	3
1024	16	32*32 bit	280.27	286 G	40/23	5½
2048	32	64*32 bit	221.78	452 G	72/34	3

TABLE II. Synthesis results for Altera Arria 10 (10AX115U1F45I1SG)

Data width	Gravano decoders	Trans-ceivers (in bits)	fmax (MHz)	Max Gross rate	ALM/mem (%)	No It.
2048	32	64*32	222.27	455 G	39/32	3
2x2048 (d = 2)	32	64*40 + 32*48	195.47	800 G	79/64	3
3x1024 (d=3)	8	64*48	338.87	1041 G	40/49	3
4x1024 (d=4)	8	64*40 + 32*48	328.19	1344 G	53/65	3
5x1024 (d=5)	8	64*56 + 32*48	313.58	1605 G	69/81	3

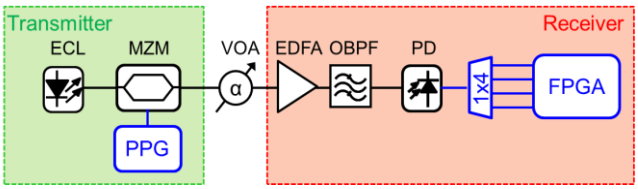

Fig. 2. Experimental Setup for 40 Gb/s.

A continuous wave (CW) external cavity laser (ECL) was modulated at 40 Gb/s in the on-off keying (OOK) modulation format using a Mach-Zehnder modulator (MZM) driven by non-return to zero electrical signals generated by a pulse pattern generator (PPG). The test-frame of 1008x1008 bits was loaded into the PPG as user-defined pattern. The modulated signal was then directly injected into a standard pre-amplified receiver for directed detection and BER measurement. At the receiver input, a noise loading stage based on a variable optical attenuator (VOA) in front of an erbium doped fiber amplifier (EDFA) was used to degrade the optical signal-to-noise ratio (OSNR) of the signal. A 2-nm wide optical bandpass filter (OBPF) was then used to suppress out-of-band amplified spontaneous emission (ASE) noise from the EDFA and a 45-GHz photodiode (PD) provided the optical-to-electrical conversion. The received 40-Gb/s electrical signal was demultiplexed into 4x10-Gb/s subcarriers, each of them input to the FPGA board for decoding and BER measurements.

The received power at the input of the receiver was tuned by varying the attenuation added by the VOA. Thus the signal OSNR was varied, leading to different pre-FEC BER performance. This, in turn, enabled to measure the post-FEC versus pre-FEC BER curve of Fig. 3.

Some of the experimental results showed no frames with errors among the 40000 frames transmitted, and these results are shown as an upper bound on the BER calculated from an upper bound on the PFL with confidence level 95% from Eq. (9-11) in [19]:

$$PFL < 1 - \sqrt[40000]{1-0.95} = 7.5 \cdot 10^{-5} \quad (5)$$

$$BER < \frac{PFL \cdot 305}{1008^2} = 2.25 \cdot 10^{-8} \quad (6)$$

where the factor 305 is the average number of bit errors per frame measured for pre-FEC BER values, where not all frames are in error. This is most likely a conservative, i.e. high, estimate as the average number of bit errors per frame in error tends to decrease as pre-FEC and post-FEC error rates decrease. The results for no errors within the 40000 frames are depicted in Fig. 3, as upper bounds given by (5-6).

Besides the lab experiment results, Fig. 3 depicts the post-FEC vs. pre-FEC performance for the simulations using 2, 3, 7, and 25 iterations in decoding the product code. The performance for the configurable code is only determined by the number of iterations for a given value of pre-FEC errors (assuming independent errors). The 2-Gravano decoder configuration from the lab experiment can make about 7 iterations. Comparing the lab results with the simulated curve at 7 iterations shows a good match. Looking closer, it is seen that many of the experimental points are very close to the curve for 7 iterations, but there are also a fair number of outlier points. These outlier points in Fig. 3 are due to time instabilities in the optical setup resulting in slow drifts of the optimum sampling point within the window of 40000 frames. Nevertheless, tuning the sampling time in the de-mux allowed bringing the performance back on top of the simulated curve.

Based on the simulated curve the NCG was estimated to range from 9.0 dB for 3 iterations to 9.4 dB for 25 iterations at the 10^{-15} BER level, with most of the high gain already achieved with 7 iterations. The estimation of the error-floor (10) depicted is presented in the Appendix. It is seen to be better than 10^{-17} in the range of interest, i.e. for the configurations with 3 – 25 iterations (Fig. 3). Thus performance at 25 iterations is very close to the limit of 9.5 dB for the product code used, [16], [17].

VI. Conclusion

A configurable HD-FEC design was presented demonstrating that constellations from 40Gb/s to 1600Gb/s are feasible on a single FPGA. NCGs from 9.0 - 9.4 dB were estimated based on simulations. The decoder delay may for high data rates be kept in the range of 5-6 µs. The post-FEC vs. pre-FEC BER was experimentally verified in a simple 40-Gb/s back-to-back optical transmission system and good agreement was found with the expected performance.

Appendix

For very small pre-FEC BER p, the error floor is dominated by the probability of a 4×4-core in the received frame and (3-4) gives the probability. Such a core is very unlikely to disappear through the iterative decoding. For larger p, there is a risk that decoding errors may create cores in the iterative process, and in this Appendix, we shall present a more precise description given the described decoding algorithm and a more moderate p.

We assume a quadratic product code with primitive BCH component codes of length n with odd minimum distance, $2T+1$, capable of correcting T errors. The component code may

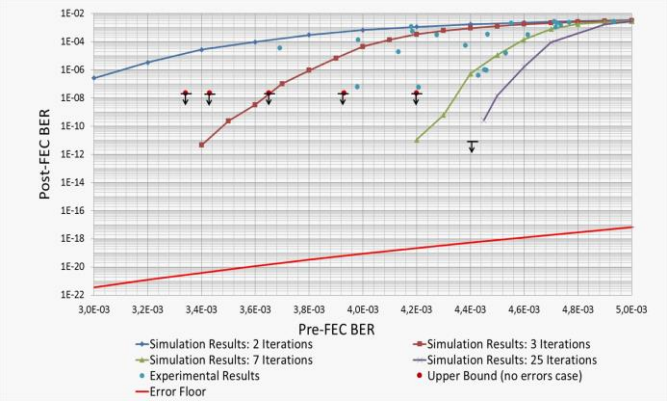

Fig. 3 Simulation results for 2, 3, 7, and 25 iterations and experimental results from 40G experiment.

introduce decoding errors when more than T errors occur in a row/column. It is difficult to specify all error patterns resulting in a core created in the iterative decoding process due to decoding errors, but we conjecture that the most probable are introduced in the first iteration of the row decoding. The code we are using (1) is a primitive BCH code expurgated with a double parity check, allowing only half of the BCH codewords with even weights to be a result of decoding and none of the codewords with odd weights. Thus T+2 errors in a row may decode to a codeword of weight $2T+2$, introducing T new errors. Therefore we shall find out how many codewords with weight $2T+2$ exist in the BCH code and how many of them have ones in the core. Error patterns resulting in codewords with higher even weights are assumed to be so rare that they may be neglected. For the code used here with T = 3, the weight distribution is known, and for higher T we may use an approximate distribution for the number of codewords with weight w, [20]:

$$A_w \approx \binom{n}{w} 2^{-(n-k')} \tag{7}$$

where n is the length of the primitive BCH code and $n-k'$ the number of parity check symbols. (Notice that this number is 2 less than the number for the expurgated code, in our case for T = 3, $n-k'$ = 30). Since the code is cyclic, the number of codewords with weight $w = 2T+2$ to have ones at $T+1$ positions (i.e. in the core) is estimated using the number of codewords with weight $2T+2$, A_{2T+2}, as

$$N_{2T+2} \approx A_{2T+2} \frac{\binom{2T+2}{T+1}}{\binom{n}{T+1}} \tag{8}$$

The code used in the experiments has a double parity check reducing N_{2T+2} to $N_{2T+2}/2$. We may then estimate the probability of frame loss giving the error floor at a pre-FEC BER p as

$$PFL = \binom{n}{T+1}^2 \left(p^{T+1} + \frac{N_{2T+2}}{2} \left(\binom{2T+2}{T+2} - (T+1) \right) p^{T+2} \right)^{T+1} \tag{9}$$

where the subtraction of $T+1$ accounts for error patterns already accounted for in the p^{T+1} term. For small values of p, this converges towards the PFL from (3). The post-FEC BER of the error floor may be estimated using (9)

$$Error\_floor(p) \approx \left(\frac{T+1}{n} \right)^2 PFL \tag{10}$$

An error floor analysis is also found in [17], but with assumption of an ideal bounded distance decoder for the BCH code. The code used in the experiment is shortened a little to n = 1008, but it is believed that the above expressions are still approximately valid. Fig. 3 shows the error floor estimated by (10) for n = 1008 and T = 3. We have used the actual number of codewords with weight $2T+2$ = 8, A_8 = 27734105949, [20].

REFERENCES

[1] B. Li, K.J. Larsen, D. Zibar, and I. Tafur Monroy, "Reconfigurable Forward Error Correction Decoder for Beyond 100 Gb/s High Speed Optical Links," *IEEE Communications Letters*, vol. 19, pp. 119-122, 2015.

[2] K. Cushon, P. Larsson-Edefors, and P. Andrekson, "Low-power 400-Gbps soft-decision LDPC FEC for optical transport networks," *J. of Lightwave Technol*., vol. 34, pp. 4304-4311, September 15, 2016.

[3] IEEE 802.3 study group, "Forward error correction for 400G: initial thoughts," 2013. [Online].
Available:http://www.ieee802.org/3/400GSG/public/adhoc/logic/jun7_13/bates_01_0613_logic.pdf

[4] IEEE 802.3 study group, "BER objective for 400GE," 2013. [Online].
Available:http://www.ieee802.org/3/400GSG/public/13_07/ofelt_400_01_0713.pdf

[5] J. Rahn, e.a. "Transmission improvement through dual-carrier FEC gain sharing," *Proc. OFC*, 2013.

[6] T. Mizuochi, T. Sugihara, Y. Miyata, K. Kubo, K. Onohara, S. Hirano, H. Yoshida, T. Yoshida, and T. Ichikawa, "Evolution and Status of Forward Error Correction," in *OFC/NFOEC 2012*, 2012, pp. OTu2A.6.pdf.

[7] A. Leven and L. Schmalen, "Status and recent advances on forward error correction technologies for lightwave systems," *J. of Lightwave Technol*., vol. 32, pp. 2735-2750, August 15, 2014.

[8] K. Sugihara, Y. Miyata, T. Sugihara, K. Kubo, H. Yoshida, W. Matsumoto, and T. Mizuochi, "A spatially-coupled type LDPC code with an NCG of 12 dB for optical transmission beyond 100 Gb/s," in *OFC 2013*, pp. OM2B.4, 2013.

[9] L. E. Nelson, G. Zhang, M. Birk, C. Skolnick, R. Isaac, Y. Pan, C. Rasmussen, G. Pendock, and B. Mikkelsen, "A robust real-time 100G transceiver with soft-decision forward error correction (Invited)," *J. of Optical Comm. and Networking*, vol. 4, No 11, pp. B131-B141, November 2012.

[10] International Telecommunication Union, "ITU-T Recommendation G.975.1:Forward error correction for high bit-rate DWDM submarine systems," February 2004.

[11] J. Justesen, K.J. Larsen, and L.A. Pedersen, "Error correcting coding for OTN," *IEEE Commun. Mag*., vol. 48, issue 9, pp. 70-75, 2010.

[12] M. Scholten and T. Coe, "Proposed Addition of CI-BCH eFEC to G.975.1," ITU-T Study Group 15 – Contribution 508, September 2009.

[13] B.P. Smith, A. Farhood, A. Hunt, F. R. Kschischang, and J. Lodge, "Staircase Codes: FEC for 100 Gb/s OTN," *J. of Lightwave Technol*., vol. 30, pp. 110-117, January 2012.

[14] S. Gravano, "Decoding the triple-error-correcting (15,5) binary BCH code by the analytic solution of the cubic error-locator polynomial over GF(2^4)," *Int. J. Electronics*, vol. 68, No 2, pp. 175-180, 1990.

[15] H. Okano and H. Imai, "A construction method of high-speed decoders using ROM's for Bose-Chaudhuri-Hocquenghem and Reed-Solomon Codes," *IEEE Trans. Computers*, vol. C-36, pp. 1165-1171, October 1987.

[16] J. Justesen, "Performance of product codes and related structures with iterated decoding," *IEEE Trans. Commun*., vol. 59, pp. 407-415, 2011.

[17] C.Häger, H.D. Pfister, A. Graell i Amat, and F. Brannström, "Density evolution and error floor analysis for staircase and braided codes," in *OFC 2016*, pp. ThA.42, March 2016.

[18] R. McEliece and L. Swanson, "On the error probability for Reed-Solomon codes," *IEEE Trans. Inf. Theory*, vol. 32, pp. 701-703, Sep 1986.

[19] International Telecommunication Union, "ITU-T G Series Recommendation - Supplement 39: Optical system design and engineering considerations," September 2012.

[20] S. Lin and D.J. Costello, *"Error Control Coding, Second Edition"*, Pearson Prentice-Hall, Upper Saddle River, NJ 07458, USA, 2004. ISBN 0-13-017973-6.