Over 10 dB Net Coding Gain Based on 20% Overhead Hard Decision Forward Error Correction in 100G Optical Communication Systems

Li, Bomin; Larsen, Knud J.; Zibar, Darko; Tafur Monroy, Idelfonso

Published in:
Proceedings of the European Conference on Optical Communication (ECOC) 2011

Publication date:
2011

Document Version
Publisher's PDF, also known as Version of record

Citation (APA):
Over 10 dB Net Coding Gain Based on 20% Overhead
Hard Decision Forward Error Correction
in 100G Optical Communication Systems

Bomin Li, Knud J. Larsen, Darko Zibar, and Idelfonso Tafur Monroy
Department of Photonics Engineering, Technical University of Denmark, Lyngby, Denmark
bomli@fotonik.dtu.dk

Abstract: We propose a product code with shortened BCH component codes for 100G optical
communication systems. Simulation result shows that 10 dB net coding gain is promising at post-
FEC BER of 1E-15.

1. Introduction

Ever increasing demands of bandwidth have motivated optical transmission systems to evolve from 10G to 40G to
100G. To compensate the increasing loss over the link, forward error correction (FEC) has been considered as a cost
effective solution. In this paper we propose a product code with shortened BCH component codes for high speed
optical communication systems with 20% overhead. Based on hard decision decoding method, it can achieve
potentially 10 dB Net Coding Gain (NCG) at post-FEC bit error rate (BER) of 10^{-15}.

First generation and second generation FEC studied hard-decision FEC (HD-FEC) with various code
constructions. The best reported performance is from Scholten et al. based on continuously interleaved
BCH(1020,988) codewords of 9.35 dB NCG [1]. Justesen also mentioned around 9.3 dB NCG in [2] based on
product code with BCH(1023,992) component code. Optical Internetworking Forum (OIF) recommended to
increase the overhead to 20% due to the necessity for even higher NCG [3]. The study of HD-FEC is somewhat
neglected afterwards.

Meanwhile, soft decision forward error correction (SD-FEC) gained a lot of interest. FPGA emulations are
carried out for low density parity check (LDPC) code [4,5] and turbo product code (TPC) [6]. The achieved NCG is
between 10.8 dB and 11.3 dB. Various codes are studied, including concatenation of different codes [7] and
combination of codes and modulation formats [8]. Simulation results show a potential NCG between 9.7 dB and
11.38 dB. Soft decision decoding uses not only the conventional decision of value 1/0 as in the hard decision, but
also information on reliabilities of these decisions. With the expense of more complexity, soft decision decoding
method provides more gain than hard decision decoding method. Therefore, more effort is required for real
implementation in Field Programmable Gate Array (FPGA) or Application-Specific Integrated Circuit (ASIC). As
the reliability information provided to SD-FEC is from signal demodulation block, the two blocks have to be
considered together in implementation. Mizuochi et al. demonstrated in [4] that SD-FEC and demodulation should
be implemented in the same ASIC to achieve the best performance and this consequently increases the complexity
of implementation. Power consumption can be another issue due to more information processing.

A difficulty in iterative decoding for HD-FEC is that the introduced decoding errors in the component codes
degrades decoding performance to an unacceptable level with accumulative iterations compared to the ideal case of
no decoding error. We propose a product code with shortened BCH component codes. The decoding error
probability of the component code is reduced to a very low level that has almost no effect on the performance of the
product code. This product code has potentially more than 10 dB NCG at post-FEC BER of 10^{-15}.

2. Code Construction and Decoding Algorithm

We choose the product code with (391,357) shortened BCH component codes as the candidate code. This code has
an overhead of 20% as recommended by OIF. The code structure is shown in Fig. 1. We first construct a product
code with BCH component codes in \(GF(2^{11})\). In a BCH component code, the first 980 bits are fixed to 0s. The
middle 357 bits are information bits and the last 34 bits are parity check bits that enable the BCH code to correct up
to 3 errors. The minimum distance of the BCH code is 8 so that a codeword with 4 errors will not be decoded. The
product code in \(GF(2^{11})\) is then shortened by removing all fixed 0s and this returns a product code with shortened
BCH component codes.

We use iterative decoding for the product code. In the component code decoding stage, each shortened
component code aims to correct up to 3 errors. The principle of the component code decoding algorithm consists of

Tu8A.3.pdf 1
7/27/2011 4:26:06 PM
three steps: to restore the BCH code, to decode BCH code and to use the shortened bits for further check. A row or a column is a BCH codeword in $GF(2^{11})$ by extending it with 980 leading 0s. If there are no more than 3 errors, the correct codeword is returned by the decoding. However, a decoding error may happen when there are more than 3 errors in a codeword and it degrades the performance of the product code heavily after multiple iterations. We therefore propose to include a third step in the decoding algorithm to reduce the decoding error probability. If any bit in the shortened bits has value 1 in the returned BCH codeword, a decoding error is reported.

Use $t$ to denote the maximum number of errors a code can correct. Decoding error probability of BCH code is $1/(t!)$ when there are more than $t$ errors. In the proposed code, a BCH component code can correct up to 3 errors. So we have $t = 3$. The introduced errors from decoding error can be regarded as randomly distributed in the codeword, though there are certain relationships between the error positions and the decoding algorithm. As decoding error with 3 new errors dominates, the theoretical decoding error probability of BCH component code is $(1/(t!))(391/2047)^3 = 0.12\%$. Simulation results match the theoretical prediction.

As the decoding error probability of component codes is very small, the performance of the product code does not degrade much compared to the ideal case of no decoding error.

3. Performance Results

Fig. 2 shows the simulation result of the proposed code in an additive white Gaussian noise Channel. One iteration consists of decoding rows once and decoding columns once. After 8 iterations, the curve drops drastically between input BER $1.2 \cdot 10^{-2}$ and $1.1 \cdot 10^{-2}$. This translates to slightly more than 10 dB NCG.

The error floor is calculated based on the formula $\left(\frac{391}{4}\right)^2 p^{16}$, where $p$ is BER. This means that if the intersecting bits of 4 rows and 4 columns are all wrong, decoding fails. The error floor value is smaller than $10^{-15}$ as shown in Fig. 2 and no further action is required.

Due to limited number of simulated frames, we take 10 iterations into consideration in the hardware implementation to ensure enough margin to get post-FEC BER of $10^{-15}$ at pre-FEC BER of $1.1 \cdot 10^{-2}$.
4. Hardware implementation

For easy hardware implementation in future, we employed the decoding algorithm in [9] as component code decoding algorithm. The algorithm takes advantage of the property that the shift of a BCH codeword is also a codeword. For a codeword of length $n$, it takes $n$ rounds to get the decoding result. In each round, with a pre-setup table, it checks whether there are only 2 or less errors left by flipping the highest bit. If so, the highest bit is flipped. Otherwise the bit keeps its value. The codeword is then shifted by one bit for the next round. After $n$ rounds, the bits in the codeword are back at original positions. Fig. 3 shows the flow of this decoding method. It is also pointed out in [9] that the major circuit area of ROM is $(4^m+1) \cdot 2^m = (4^{11}+1) \cdot 2048 = 90K$. We use this algorithm to find the first error bit. The rest error positions will be deduced from checking syndrome values.

![Fig. 3. Flow of Component Code Decoding](image)

Now we look at the implementation complexity. In the row decoding of first iteration, one lookup table is shared by 4 BCH decoders. In total we need 90K-391/4 = 8.7M ROM. In this step, we will check the first 190 bits in a codeword. As 4 decoders shares one lookup table, we need 190/4=760 clock cycles. The required flip-flops of shift registers are 190-391 = 76K. The result is forwarded to an interleaver of 391-391 = 0.15M bits. In total the ROM and flip-flops in this step is ~7.7M and ~0.22M. Column decoding is similar to row decoding. The first iteration will cost ~17.5M ROM and ~0.45M flip flops. The same process is repeated in the second iteration except that we check the last 190 bits when decoding each component codeword. The third iteration will repeat the process of first iteration and so forth. As most errors are corrected in the first 5 iterations, the major circuit required will be ~17.5M-5 = 87.5M ROM and ~0.45M-5 = 2.25M flip flops. The last 5 iterations only correct errors for a few codewords and the added redundancy is limited. It is feasible to take this implementation out in several state-of-the-art FPGAs.

Each step takes around 760 clock cycles as mentioned above. This equals to 1520 $\mu$s in case that FPGA implementation works under 500 MHz system clock frequency. To receive a frame on a 100G interface takes 391-391-10 $\mu$s = 1529 $\mu$s. So the implementation is fast enough to work for 100G systems.

5. Conclusion

We proposed a product code with shortened BCH component code for high speed optical communication systems. The code provides potentially more than 10 dB NECG at post-FEC BER of $10^{-15}$. The hardware implementation is also investigated and the FPGA verification is feasible.

6. References