Fast SD-Hamming Decoding in FPGA for High-Speed Concatenated FEC for Optical Communication

Zhang, Can; Forchhammer, Søren; Andersen, Jakob Dahl; Mehmood, Tayyab; Yankov, Metodi Plamenov; Larsen, Knud J.

Published in:
Proceedings of 2020 IEEE Global Communications Conference

Link to article, DOI:
10.1109/GLOBECOM42002.2020.9322436

Publication date:
2021

Document Version
Peer reviewed version

Fast SD-Hamming Decoding in FPGA for High-Speed Concatenated FEC for Optical Communication

Can Zhang, Søren Forchhammer, Jakob Dahl Andersen, Tayyab Mehmood, Metodi P. Yankov, Knud J. Larsen
Department of Photonics Engineering
Technical University of Denmark
Kongens Lyngby, Denmark
Email: canzh, sofo@fotonik.dtu.dk

***Abstract*—In this paper, we consider fast decoding of soft-decision (SD) Hamming codes as inner codes in concatenated forward error-correction (FEC) schemes for high-speed optical communication. The goal is single FPGA implementations at speeds of 400 Gb/s and beyond. A low complexity maximum *a posteriori* (MAP) probability decoding is applied to a (128,120) Hamming code. Chase decoding of a (128,119) Hamming code is also implemented. The VHDL designs for both decoding schemes are presented. The FEC performance and FPGA resource utilization are investigated and compared. Synthesis results indicate that, both the Chase and the MAP decoder leave sufficient resources available to also accommodate a powerful outer hard decision code, on a single FPGA. Furthermore, MAP decoding of (128,120) Hamming code features lower hardware complexity and provides a higher data throughput.**

***Index Terms*—Soft-decision, Hamming codes, Chase decoding, concatenated coding, MAP decoding, VHDL, FPGA.**

I. Introduction

For fast and reliable high-speed connections between data centers, forward error correction (FEC) is applied. In optical communication, FEC provides the required bit error rate (BER), which is often specified at 10^{-15} [1], [2]. Highly efficient hard-decision (HD) FECs are deployed in optical communication using product codes [3], [4] or staircase codes [5], [6].

To improve the performance further, soft-decision (SD) FECs are studied and proposed, at the expense of increased complexity. One approach is concatenated coding using a SD inner code and a HD outer code [7], as in the 400G ZR implementation, recently specified by the Optical Interconnect Forum (OIF) [8], [9]. This system combines an inner SD-Hamming code with a HD outer staircase code in a 16 Quadrature Amplitude Modulation (QAM) optical communication system targeted at Data Center Interconnects (DCIs).

In this paper, we consider fast decoding of soft-decision Hamming codes. We shall refer to this setting as SD-Hamming. We consider the case where the SD-Hamming code is intended as the inner code in a concatenated coding scheme (Fig. 1), as in OIF 400G ZR [8], [9], where the outer code is a HD staircase code.

This work was supported by the IFD INCOM project (8057-00059B) and DNRF Research CoE SPOC, ref. DNRF123.

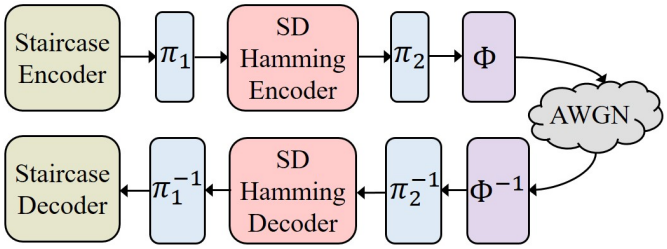

Fig. 1. Block diagram of concatenated FEC coding and modulation over AWGN channel with an inner SD-Hamming code and the outer staircase code. π denotes interleaving and ϕ the mapping of the modulation format.

We set the 4.5×10^{-3} output BER of the inner code as our target to match the 10^{-15} output BER of an outer staircase code, as is specified in [1] at the overhead $\sim 6.7\%$. Higher BER can also be relevant, if relaxing the BER requirement after outer code [1] or allowing a higher overhead [10]. An implementation of 400G ZR based on 50 FPGAs was presented in [9] for verification and investigation. This implementation had a throughput of 200 Gb/s.

The goal here is fast decoding of the FEC enabling a 400 Gb/s concatenated decoding implementation on a single FPGA. We also investigate the feasibility of having SD-Hamming decoding at 800 Gb/s on the FPGA. We focus on a fast maximum *a posteriori* (MAP) probability decoding scheme for SD-Hamming. For comparison we implement and compare with fast decoding based on a Chase algorithm. The FEC solution presented in this work can be deployed to potential ZR+ applications, with extended capabilities of 400G ZR.

Sec. II presents related work on different approaches for SD-Hamming decoding, with a particular focus on fast MAP. In Sec. III, the Hamming codes considered are presented together with fast soft-decision decoding using both fast MAP decoding and Chase decoding. In Sec. IV, a VHDL design is presented and in Sec. V, VHDL synthesis results towards 400G/s and 800G/s are presented. Sec. VI discusses the concatenation of SD-Hamming with the HD outer code.

II. Related Works

Chase decoding [11] and fast MAP decoding [12] have been considered and deployed in different coding schemes. A low-complexity serial code concatenation for high-speed

978-1-7281-8298-8/20/$31.00 ©2020 IEEE

optical communication systems was proposed in [13], where the inner code is a Turbo Product Code (TPC) based on two extended Hamming codes, with parameters (128,120) and (64,57). The fast MAP algorithm is used among other techniques, but no performance results were reported for the fast MAP in [13]. I. B. Djordjevic *et al.* proposed a reverse concatenated BCH-LDPC code [14]. Fast MAP is applied to decode an extended BCH (128,120) code, which is the same as an extended Hamming (128,120) code. The decoding performance outperforms its TPC counterpart, using Chase and BCJR for decoding, respectively. In [15] Generalized LDPC code with Reed-Muller (RM) and BCH codes as component code was considered. After decomposing the RM(4,6) to several first-order RM codes, 64 parallel low-complexity MAP decoders are used for decoding. It has been shown that the RM(4,6)-based GLDPC code features a much lower decoding complexity, compared with the TPC based on BCH(128,113) × BCH(256,239), which requires 239 parallel Chase decoders. Chase decoding for TPC constructed by extended Hamming code (64,57) was considered in [16], and the synthesis results on a Xilinx Artix-7 FPGA were presented. In [9], Chase decoding of the 400G ZR (128,119) Hamming code was implemented. We design and implement a Chase decoder for the same code.

It is noted that MAP decoding for SD-Hamming (128,120) is only seen in [13] and [14] as part of different coding schemes. [13]–[15] deployed fast MAP decoding in simulations for extended Hamming/BCH and RM codes due to its low-complexity, whose structure is also suitable for FPGA/ASIC implementation. However, to the best of our knowledge, hardware implementation results for fast MAP decoding have not been reported in the literature.

III. Hamming codes and SD Decoding

We consider Hamming codes as the inner code in a concatenated coding scheme as in OIF 400G ZR [8], where the outer code is an HD staircase code. Thus, the Hamming code serves as an error-reducing code utilizing the soft information in the received signal. The outer code serves the purpose of bringing the BER to the low values required, e.g. 10^{-15}. This means a combined design, where the BER output of the SD-Hamming decoding is required to be better than, e.g. 4.5×10^{-3} [1], depending on the channel and the parameters and performance of decoding the outer code.

A. Hamming codes

Hamming codes were the first error-correcting code to be defined. The $(n, k) = (2^m - 1,\ 2^m - m - 1)$ code is specified from the $(2^m - 1) \times m$ parity check matrix having all non-zero m-tuples as rows in some order. The order may be chosen to ease implementation or to provide some specific properties. Hamming codes have a minimum distance of 3, and thus are always able to correct one error. The codes may be extended by an overall parity check to have length 2^m and minimum distance 4. We shall consider the (128,120) extended Hamming code below. Error performance could be improved by expurgating some codewords away by converting an information symbol into an extra parity check which is specified by an extra column in the parity check matrix. We shall consider such a (128,119) code as specified by OIF in [8]. The extra column in the parity check matrix excludes codewords where a selection of positions (64) has odd weight.

For SD decoding of the Hamming codes, we assume that the demodulator provides the Log-Likelihood-Ratio (LLR) for each received value y_j given transmitted binary c_j at time j by

$$L_j = \log \frac{\Pr\left(y_j | c_j = 1\right)}{\Pr\left(y_j | c_j = 0\right)} \tag{1}$$

where $\Pr(y_j|c_j)$ denotes the conditional probability of y_j given c_j. The value L_j is easily calculated from the received values, both for BPSK (e.g. extracted from QPSK) and 16QAM modulation. We shall consider Chase decoding [8] for the (128,119) code and MAP decoding [12] for the (128,120) code. Chase decoding of (128,120) has also been investigated, but performance is inferior to MAP. Another option for SD decoding is based on a trellis formulation for the block code facilitating a trellis decoding algorithm like Viterbi or BCJR MAP [17]. However, the trellis for this length of codes seems too complex, compared to the fast Chase and MAP decoding considered.

B. MAP Decoding of (128,120) Hamming code

The algorithm is based on MAP decoding of a linear code by using its dual code [12]. The fast computation of the soft decisions is achieved by using fast Hadamard transforms (FHT) of size n×n [12], where n corresponds to the code length. The coefficients of FHT are all +1 or -1, therefore the computation is reduced to only addition and subtraction operations. The number of operations is proportional to $n\log_2 n$. Other steps in the fast MAP algorithm all have linear complexity. The overall complexity of fast MAP is $O\left(n \log_2 n\right)$, while applying a BCJR algorithm for Hamming codes has complexity $O(n^2)$ [12]. The extended Hamming code (128,120) can be seen as the dual code of the first order Reed-Muller code $\mathcal{RM}(1, 7)$ if a particular order of rows in the parity check matrix for the extended Hamming code is used. The key steps in the fast MAP algorithm is briefly outlined as follows. Firstly, the terms ρ_j and τ_j are computed from the soft-values in (1).

$$\rho_j = \frac{1 - \frac{\Pr(y_j|1)}{\Pr(y_j|0)}}{1 + \frac{\Pr(y_j|1)}{\Pr(y_j|0)}} = \Pr\left(c_j = 0 | y_j\right) - \Pr\left(c_j = 1 | y_j\right) \tag{2}$$

$$\tau_j = \ln\left|\rho_j\right|, \quad j = 0, \ldots, n-1 \tag{3}$$

The first part computes the product of probabilities $h_i^{(0)}$ and $h_i^{(1)}$ by the following expression

$$h_i^{(0)} = \prod_{j \in \mathcal{I}_i^{(0)}} \rho_j \text{ and } h_i^{(1)} = \prod_{j \in \mathcal{I}_i^{(1)}} \rho_j \quad i = 0, \ldots, n-1 \tag{4}$$

where the sets $\mathcal{I}_i^{(0)}$ and $\mathcal{I}_i^{(1)}$ refers to the positions where the ith codeword of $\mathcal{RM}(1, 7)$ has 0's and 1's, respectively.

The fast computation separates the sign and absolute value of (4). We first compute the sum of logarithms of probabilities

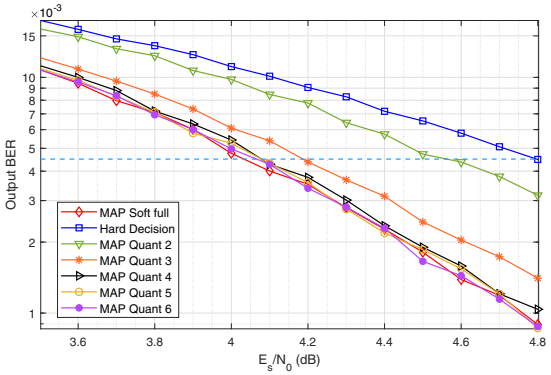

Fig. 2. MAP decoding performance of (128,120) Hamming code for different levels of quantization of input values.

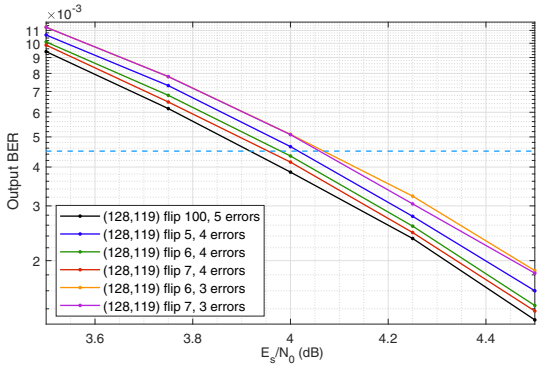

Fig. 3. Chase decoding performance of (128,119) Hamming code for different choice of Chase parameters: Number of candidates for flipping (flip) and maximum number of errors corrected.

$\sum_{j\in\mathcal{I}_i^{()}} \ln|\rho_j|$ (5). This term is obtained by applying FHT to the vector $(\tau_0, \tau_1, \ldots, \tau_{n-1})$. The computation performs the steps in (6) $\log_2 n$ times.

$$s_j = \tau_{2j}+\tau_{2j+1}, \quad s_{n/2+j} = \tau_{2j}-\tau_{2j+1} \quad j = 0,\ldots,n/2-1$$
$$\tau_j = s_j \quad j = 0,\ldots,n-1 \tag{6}$$

To determine the signs of (4), the signs of ρ_j are grouped to a new vector and a modified Walsh-Hadamard transform over GF(2) is used. Finally (4) can be reconstructed with the help of exponentiation.

In the second part, let $x_i = h_i^{(0)} - h_i^{(1)}$. By applying FHT again to $(x_0, x_1, \ldots, x_{n-1})$, and after some auxiliary computations, the soft decision for all coded bits $\Pr\left(c_j = 0|\underline{y}\right) - \Pr\left(c_j = 1|\underline{y}\right)$ is obtained. Detailed description and proof of the MAP decoding algorithm is given in [12].

Considering different input quantization levels, the MAP decoding performance of the Hamming (128,120) code is investigated. The random bits are generated from the function `randi` in MATLAB. We have simulated 10^4 blocks of 128 bits. For BPSK, the received signal is quantized to 1 sign bit, 1 integer bit and N fractional bits. All values use saturation at ± 1 before scaling. We have tested with N = 2, 3, 4, 5 and 6, respectively, and compared with the performance when using full precision soft bits. As is indicated in Fig. 2, using 6 fractional bits for the soft information achieves almost the full potential of MAP decoding. Using 4 or 5 fractional bits still achieves an acceptable performance. The N = 3 quantization setting corresponds to a ca. 0.2 dB loss in a BPSK channel at the target output BER threshold of 4.5×10^{-3}.

C. Chase decoding of (128,119) Hamming code

The basic idea of Chase decoding [11] is simply to find multiple possible error patterns for the received word by a combination of bit-flipping and hard decision decoding. For each error pattern, the distance to the actual received word is calculated based on the soft input values. Finally, the error pattern with the best distance (maximum likelihood) is chosen as the most likely error pattern. Chase decoding can and has been applied to a wide variety of codes designed for HD decoding, extending them to SD decoders.

There exist many variations of the algorithm, but basically there are two parameters: how many positions are candidates for flipping and how many bits can be flipped simultaneously. The candidates for flipping are chosen among the positions with the most unreliable received values. Including many patterns increases the decoder complexity, since the number of hard-decision decodings increases rapidly. While increasing the number of patterns, the performance of the decoding approaches that of true maximum likelihood decoding.

We have investigated the performance for various settings of the two parameters for the (128,119) code of interest and conclude that flipping up to 3 positions out of the 6 most unreliable gives a performance within 0.1 dB of that of a maximum likelihood decoding and at the same time a reasonable implementation complexity. This choice implies 42 hard-decision decoding instances of the Hamming code, i.e. 1 for flipping 0 positions, 6 for flipping 1 position, 15 for flipping 2 positions and 20 for flipping 3 positions. However, as we shall see the number of decoders can be reduced for the specific (128,119) code. The performance is depicted in Fig. 3 as a function of the signal-to-noise ratio (E_s/N_0), and it is in line with the performance reported in [5]. In Fig. 3, the performance for different combinations of the number of candidates (flip) and the maximum number of errors corrected which is given by the maximum number flipped plus the 1 error corrected by the Hamming code.

D. Comparison of MAP and Chase decoding

The output BER performance from the SD-Hamming decoder is investigated for different settings of Chase parameters and MAP quantization levels, which is depicted in Fig. 4 as a function of (E_b/N_0). When using 6 fractional bits for input soft values and full precision for other internal values (MAP Quant 6,floating point), the output BER of MAP decoding is the same as that of finding 6 candidates for flipping in the Chase decoding (flip 6) and identical to that of using full precision ("MAP soft full" in Fig.2). For fast implementations, the internal values in the MAP are quantized to finite precisions. Thus the output BER performance is shown in Fig. 4 if input values are quantized to 6 and 4 fractional bits, respectively. A light version of the Chase (flip 5 reduced) is achieved when finding 5 flip candidates, flipping each of these and 4 combinations of 2 flips.

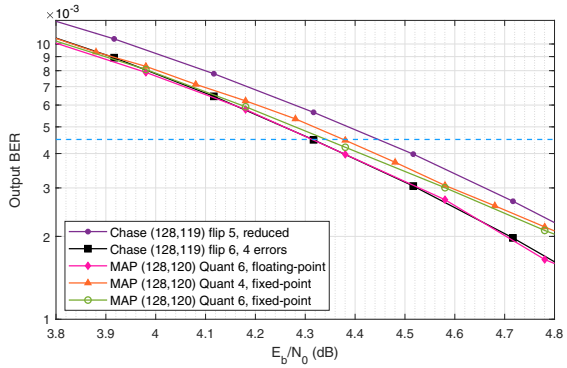

Fig. 4. Comparison of decoding performance for MAP and Chase decoding.

The MAP decoder has a higher code rate of 120/128 vs. a code rate of 119/128 for the Chase decoder. The (128,120) code with MAP decoding provides ca. a 1% higher information throughput. This is incorporated in Fig. 4 by using E_b/N_0. The MAP decoder can furthermore output soft values (which we have also implemented with a marginal increase in decoding resources.)

IV. SD-Hamming VHDL Design

The VHDL design of the SD-Hamming decoders studied differ significantly. The transforms are applied in the MAP decoder, whereas many parallel decoding instances are performed based on potential error patterns in the Chase decoder.

A. MAP decoding of (128,120) Hamming code

1) Fixed-point computations: Although the simulated results in Fig. 2 demonstrate the MAP decoder performance with quantization on the soft values, the computation of soft decisions, in principle, here still uses full precision of the floating point numbers in all steps. When it comes to VHDL design, the fixed-point arithmetic is performed for computing all the temporary values. To further reduce the hardware complexity and allow for fast implementation, exponentiation and logarithm operations in the MAP decoding algorithm have been substituted by pre-computed values in look-up tables, based on the knowledge that input values have been quantized to a given precision.

In two's complement, each value is quantized to a finite precision, represented by one sign bit, L integer bits and N fraction bits. The quantization error is accumulated and propagated in each computational step, and the actual effective number of soft information bits becomes smaller, which degrades the decoding performance. Hence, finding the minimum requirement of bit-width to store the primary variables within the MAP decoder is a critical part in the VHDL design.

A rough estimate can be firstly achieved by simulations, gathering enough statistics for all internal values at different SNRs. This involves finding the maximum absolute value to determine the number of bits to allocate, and finding the minimum absolute value and minimum fraction for each parameter to determine the number of fractional bits to allocate. Since the inner code serves as an error-reducing code, a slight loss in performance can be tolerated using fewer fractional bits.

From the detailed description presented in [12], the value of ρ is only used in the last step for computing the soft decision. By comparison, FHT is performed on τ (3) to compute the sum of logarithms of probabilities in (5). Hence quantization errors in τ are more likely to accumulate and the precision of τ plays a dominant role in the MAP decoder. By comparing different bit width settings, the desired decoding performance can be achieved, when allocating 5 and 8 fractional bits to ρ (2) and τ (3), respectively. For all other temporary values, they are set to 16-bit width, which appears to be enough by conducting behavioral simulations for the VHDL design.

2) Fast Hadamard transform: Among all the computations of temporary values in the fast MAP algorithm, the twofold use of FHT (6) requires the largest number of add/subtract operations. Accordingly, an area-efficient and high-speed implementation of FHT is our main focus in the VHDL design.

Due to the recursive property of the Hadamard transform, the FHT can be implemented in a fully pipelined way. The size 4×4 transform is considered as a kernel module and computed in two stages. Each stage consists of 2 adders and 2 subtractors. Then the size 8×8 transform is further implemented in three stages, and the result is obtained from two 4-point transform modules and an additional stage of add/subtract operations. Similarly, the 128×128 transform in the MAP decoder can be easily computed from these short length modules, and cascaded in a 7-stage pipeline. This design can execute FHT in a minimum number of clock cycles, at the expense of requiring more FPGA resources. Alternatively, to allow for efficient hardware resource reuse, an iterative FHT architecture is considered as in (6). The 128×128 FHT design block in FPGA is constructed by one stage consisting of 64 adders and 64 substractors. The termination of computation is controlled by a counter and extra logic units, which switch the register values out of the FHT back to the FHT input multiplexer, and finally outputs the FHT result after 7 iterations. Both of these two ways have been implemented and performance results are given in Sec. V-C1.

B. Chase decoding of (128,119) Hamming code

First, our Chase decoder does a partial sorting of the soft-decision input values to identify unreliable bits. Then, parallel hard-decision decoders will decode the received word with selected combinations of bits flipped, and finally the best one is selected. To optimize resources, an initial syndrome is calculated and thereafter modified according to the flip-patterns.

1) Sorting the soft-values: Each soft input value is received as one sign bit and 4 soft decision bits. A partial sorting of the 128 values is first performed to find the 6 minimum absolute LLR values. All the 128 values of one code word arrive in every clock cycle. The solution we chose for sorting the 128 values, is based on the idea of merging two sorted lists of 6 elements into one sorted list of the 6 minimum elements.

First the 128 values are inserted into 22 sorted lists of length 6. Then these lists are merged in a tree structure to one final list of the 6 minimum values.

2) Syndrome calculation: The syndrome is calculated by multiplication of the received 128 bit sign vector and the 128×9 parity check matrix - modulo 2 [8]. Each syndrome bit is an XOR of the sign vector bits at the positions in the column where the parity matrix is 1.

Some syndromes indicate that 1 error is the most likely while others indicate more than 1 error. We only make a correction when the syndrome indicates 1 error. Since the last syndrome bit is just an XOR of all 128 positions, all syndromes indicating 1 error have this bit as a 1. This means that some of the flipping patterns can be excluded based on this syndrome bit. If it is already 1, we will flip an even number of positions (0 and 2) otherwise we will flip an odd number of bits (1 and 3). Based on this parity bit, the maximum number of Hamming decoders required is reduced from 42 to 26.

The basic syndrome calculated based on the received bits is modified to match the various flip-patterns. This is found by a table look-up and the modifications are XOR'ed to the basic syndrome for each flip-pattern.

3) Hamming decoder: After the syndrome calculation, the error position is found by a look-up table with 256 entries. The fast implementation uses 26 Hamming decoders in parallel.

V. FPGA Synthesis results

A. FPGA device and synthesis tool

The SD-Hamming decoder VHDL design has been synthesized for a Xilinx Virtex UltraScale+ family device: XCVU9P-L2FLGA2104. The synthesis, place and route use all the default strategies in Vivado v2019.1.3. In the target UltraScale architecture, each Configurable Logic Block (CLB) provides 8×6-input look-up table (LUT)s and 16 flip flops [18]. 2160 on-chip block RAM blocks are available in this VU9P device. Each block stores up to 36 Kb data, which can be configured as either two independent 18 Kb RAMs (RAMB18E2), or one 36 Kb RAM (RAMB36E2).

B. Performance and complexity metrics

After the place and route process, the maximum operating frequency (F_{max}) is achieved, based on which the information data rate is calculated. The hardware complexity is depicted by the logic utilization (CLB, CLB LUTs, CLB Registers) and the number of occupied block RAMs [19]. Dedicated multipliers on the FPGA are not used. The numbers for resource consumption and their percentage of maximum available amount are both presented in Tables I-II.

C. Analysis of results

1) MAP decoder: The MAP decoder is synthesized under different settings of input quantization levels (Fig. 2). We have investigated 8, 6 and 4 bits, respectively. The logic utilization is slightly lower when using fewer bits. This is attributed to the reduced size of look-up tables to store ρ (2) and τ (3). In this work, we focus on the 8 bits (6 fraction bits) setting, which gives a better decoding performance.

As discussed in Sec. IV-A, we've considered two different architectures to implement the FHT block. Both of these VHDL designs enable a data rate at 400 Gb/s and 800 Gb/s, using 9 and 20 parallel MAP decoders, respectively. As is shown in Table I, MAP decoding based on the iterative reuse FHT architecture requires 15.5% and 33.6% of CLBs for 400 Gb/s and 800 Gb/s, respectively. Specifically, the two FHT blocks occupy approximately 60% of the CLB LUTs used in the MAP decoder (9250 out of 14716). By comparison, the fully pipelined FHT architecture requires 33.4% and 70.1% of CLBs for 400 Gb/s and 800 Gb/s, respectively. Accordingly, the iterative reuse FHT architecture greatly lowers the requirement of FPGA resource utilization and reduces the overall complexity.

It is worth mentioning that, such an iterative architecture could possibly lead to inferior timing closure performance when the combined design gets larger. The increased routing delay lies in the path between the FHT output and FHT input multiplexer, i.e., the path to perform each iteration. According to our experiments, there is still a good timing margin in this path when targeting at 400G and 800G implementations, since parallelizing 9 and 20 MAP decoders also relaxes the minimum speed requirement of each decoder.

2) Chase decoder: Table II shows the synthesis results of a single Chase decoder, which is configured to find 6 candidates for flipping and correcting up to 4 errors, as is illustrated in Sec. III-C and performance given by the green curve shown in Fig. 3. By using 8 and 20 parallel Chase decoders, the achieved information rates out of the SD-Hamming decoder are 428.4 and 854.4 Gb/s, respectively. This allows for redundancy required by an outer code for 400 and 800 Gb/s information rate out of a concatenated coding scheme. A lower hardware complexity of the Chase decoder can be achieved by considering fewer flipping candidates. When finding 5 candidates for flipping (see "flip 5 reduced" in Fig. 4), synthesis results show that the CLB utilization is reduced to 1.7%, at the cost of 0.1 dB in performance compared with the Chase decoder above.

VI. Concatenated coding

The target application of the SD-Hamming code is, as mentioned, to concatenate it with a HD outer code for a 400G (and beyond) FEC solution. We have evaluated the performance of (128,119) Chase decoding and (128,120) MAP decoding respectively in a concatenated staircase and Hamming code (CSHC) setting, as in 400G ZR [8]. In a CSHC set-up with 16QAM modulation, the MAP decoder achieved a 0.1 dB improvement over the Chase in terms of gap to constrained capacity for 16QAM modulation (without constellation shaping) at an overall output BER of 10^{-9} [20]. As shown in Sec. V-C1, the MAP decoder requires 15.5% and 33.6% of the CLBs on the UltraScale+ FPGA, for 400Gb/s and 800Gb/s respectively. This leaves sufficient resources for an efficient outer HD code. As an example,

TABLE I
SYNTHESIS RESULTS FOR MAP DECODING OF SD-HAMMING ON VIRTEX ULTRASCALE+ (XCVU9P-L2FLGA2104)

Input data width	Inf. data width	MAP decoders	Est. Fmax (MHz)	Inf. data rate (Gb/s)	CLBs	CLB LUTs	CLB Registers	RAMB36E2	RAMB18E2
128×8	120	1	529	63.5	2475(1.7%)	14716(1.2%)	18987(0.8%)	2(0.1%)	0
9×128×8	9×120	9	419	452.5	22930(15.5%)	123644(10.5%)	171420(7.2%)	18(0.8%)	0
20×128×8	20×120	20	371	890.4	49643(33.6%)	275205(23.3%)	378635(16.0%)	40(1.9%)	0

TABLE II
SYNTHESIS RESULTS FOR CHASE DECODING OF SD-HAMMING ON VIRTEX ULTRASCALE+ (XCVU9P-L2FLGA2104)

Input data width	Inf. data width	Chase decoders	Est. Fmax (MHz)	Inf. data rate (Gb/s)	CLBs	CLB LUTs	CLB Registers	RAMB36E2	RAMB18E2
128×5	119	1	503	59.9	5971(4.0%)	34217(2.9%)	44433(1.9%)	0	29(0.7%)
8×128×5	8×119	8	450	428.4	42384(28.7%)	259457(21.9%)	332719(14.1%)	0	232(5.4%)
20×128×5	20×119	20	359	854.4	93110(63.0%)	548568(46.4%)	822115(34.8%)	0	580(13.4%)

the HD product code considered in [3] was synthesized on an Altera/Intel Arria 10 FPGA. The parallel implemented decoders occupy 39% and 79% of ALMs (Adaptive Logic Module) in the Arria 10 to achieve max gross rate at 455Gb/s and 800Gb/s, respectively. This product code is based on BCH (after Bose, Chaudhuri, and Hocquenghem) component codes decoded over GF (2^{10}) capable of correcting $t = 3$ errors. Concatenation of the MAP SD-Hamming and the BCH $t = 3$ product code should be quite feasible on the more powerful UltraScale+ considered for 400Gb/s and most likely also for 800Gb/s. The staircase code used in 400G ZR is also based on BCH $t = 3$ component codes. Comparing the number of BCH decodings, the product code is comparable to a staircase code with window size 4 and 1 iteration in terms of complexity. Since the MAP and the Chase decoder, on the more powerful UltraScale+ FPGA, uses around 16% and 29% of the CLB resources, we conjecture that there will also be space for a staircase code implementation at 400Gb/s, also with window size more than 4 and/or more iterations. Thus, if we aim for a fast low complexity set-up of a staircase code in terms of window size and number of iterations, it should also be feasible to implement along with especially the MAP but also the Chase SD-Hamming on a single FPGA.

VII. Conclusion

In this paper, two approaches to decoding SD-Hamming codes were presented, based on the fast MAP algorithm and Chase algorithm, respectively. In both cases, VHDL design and synthesis results towards 400 and 800 Gb/s on a single Xilinx UltraScale+ FPGA were presented. Compared with the Chase decoder for SD-Hamming (128,119), the MAP decoder for SD-Hamming (128,120) features a lower hardware complexity for FPGA implementation and achieves a higher information data throughput. By parallelizing 9 and 20 MAP decoders, 15.5% and 33.6% of CLBs are utilized to achieve a maximum net data rate of 452 Gb/s and 890 Gb/s, respectively. Furthermore, a slightly higher error correction performance is obtained using MAP decoding of the (128,120) Hamming code. It is further argued that, concatenating the MAP or Chase based decoding of SD-Hamming with a HD product code or a low complexity HD staircase code, both based on BCH component codes, decoding of a combined concatenated code at 400 and 800 Gb/s on a single Xilinx UltraScale+ FPGA is feasible.

References

[1] ITU-T, OTU4 long-reach interface, Rec. G.709.2/Y.1331.2, July 2018.

[2] D. A. Morero et al., "Design Tradeoffs and Challenges in Practical Coherent Optical Transceiver Implementations," in J. Lightwave Technol, vol. 34, no. 1, pp. 121-136, 1 Jan.1, 2016.

[3] J. Dahl Andersen et al., "A configurable FPGA FEC unit for Tb/s optical communication", IEEE Int. Conf. Commun. (ICC '17), 2017, pp. 1-6.

[4] ITU, Telecommunication Standardization Sector: Forward error correction for high bit-rate DWDM submarine systems. Rec. 975.1, 2004

[5] B. P. Smith et al., "Leveraging 400G ZR FEC technology," http://www.ieee802.org/3/B10K/public/17_11/lyubomirsky_b10k_01_1117.pdf

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

[7] B. P. Smith and F. R. Kschischang, "A Pragmatic Coded Modulation Scheme for High-Spectral-Efficiency Fiber-Optic Communications," in J. Lightwave Technol, vol. 30, no. 13, pp. 2047-2053, July. 2012.

[8] OIF Optical Internetworking Forum, "Implementation agreement 400ZR", IA # OIF-400ZR 0.10-Draft.

[9] Y. Cai et al., "FPGA Investigation on Error-Flare Performance of a Concatenated Staircase and Hamming FEC Code for 400G Inter-Data Center Interconnect," in J. Lightwave Technol, vol. 37, no. 1, pp. 188-195, 1 Jan.1, 2019.

[10] L. M. Zhang and F. R. Kschischang, "Staircase codes with 6% to 33% overhead," J. Lightwave Technol, 32(10), 1999-2002, 2014.

[11] D. Chase, "A Class of Algorithms for Decoding Block Codes with Channel Measurement Information,", IEEE Transactions on Information Theory, vol. IT-18, no. 1, pp. 170 – 182, Jan 1972.

[12] A. Ashikhmin and S. Litsyn, "Simple MAP decoding of first-order Reed-Muller and Hamming codes", IEEE Transactions on Information Theory, IT-50, August 2004, pp 1812-1818.

[13] D. Morero and M. Hueda, "Novel serial code concatenation strategies for error floor mitigation of low-density parity-check and turbo product codes," in Canadian Journal of Electrical and Computer Engineering, vol. 36, no. 2, pp. 52-59, Spring 2013.

[14] I. B. Djordjevic, L. Xu and T. Wang, "On the reverse concatenated coded-modulation for ultra-high-speed optical transport," Opt. Fiber Commun. Conf. Expo. Natl. Fiber Opt. Eng. Conf. (OFC/NFOEC'11), 2011, paper OWF3.

[15] I. B. Djordjevic et al., "GLDPC Codes with Reed-Muller Component Codes Suitable for Optical Communications," in IEEE Communications Letters, vol. 12, no. 9, pp. 684-686, Sept. 2008.

[16] W. Kuang, R. Zhao and Z. Juan, "FPGA implementation of a modified turbo product code decoder," 2017 IEEE 9th International Conference on Communication Software and Networks, 2017, pp. 71-74.

[17] R. J. McEliece, "On the BCJR Trellis for Linear Block Codes," IEEE Trans. Inf. Theory, vol. IT-42, pp. 1072-1092, July 1996.

[18] UltraScale Architecture Configurable Logic Block User Guide (UG574) https://www.xilinx.com/support/documentation/user_guides/ug574-ultrascale-clb.pdf

[19] I. B. Djordjevic and D. Zou, "FPGA-based rate-adaptive LDPC-coded modulation for the next generation of optical communication systems," 2017 19th International Conference on Transparent Optical Networks (ICTON), Girona, 2017, pp. 1-6.

[20] A. Bisplinghoff, S. Langenbach and T. Kupfer, "Low-Power, Phase-Slip Tolerant, Multilevel Coding for M-QAM," in J. Lightwave Technol, vol. 35, no. 4, pp. 1006-1014, 15 Feb.15, 2017.