IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Hardware Design of Concatenated Zigzag Hadamard Encoder/Decoder System with High Throughput

SHENG JIANG[1], FRANCIS C.M. LAU[2], (Senior Member, IEEE), and CHIU-W. SHAM[3], (Senior Member, IEEE)

[1]Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong (e-mail: sheng.jiang@connect.polyu.hk)
[2]Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong (e-mail: francis-cm.lau@polyu.edu.hk)
[3]Department of Computer Science, The University of Auckland, New Zealand (e-mail: b.sham@auckland.ac.nz)

Corresponding author: Francis C.M. Lau (e-mail: francis-cm.lau@polyu.edu.hk).

The work described in this paper was partially supported by a grant from the RGC of the Hong Kong SAR, China (Project No. PolyU 152170/18E).

ABSTRACT Both turbo Hadamard codes and concatenated zigzag Hadamard codes are ultimate-Shannon-limit-approaching channel codes. The former one requires the use of Bahl-Cocke-Jelinek-Raviv (BCJR) in the iterative decoding process, making the decoder structure more complex and limiting its throughput. The latter one, however, does not involve BCJR decoding. Hence its decoder structure can be much simpler and can potentially operate at a much higher throughput. In this paper, we investigate the hardware design of a concatenated zigzag Hadamard encoder/decoder system and implement it onto an FPGA board. We design a decoder capable of decoding multiple codewords at the same time, and the proposed system can operate with a throughput of 1.44 Gbps — an increase of 50% compared with the turbo Hadamard encoder/decoder system. As for the error performance, the encoder/decoder system with a 6-bit quantization achieves a bit error rate of 2×10^{-5} at $E_b/N_0 = -0.2$ dB.

INDEX TERMS concatenated zigzag Hadamard code, hardware design, high throughput, turbo Hadamard code, zigzag Hadamard code.

I. INTRODUCTION

WITH the fast development of communication technologies, the requirements on forward-error-correction (FEC) codes are becoming more and more rigorous. Among the good FEC codes, turbo codes [1–3], low-density parity-check (LDPC) codes [4–7] and polar codes [8–11] have been intensively studied because they can perform close to the capacity limits. In addition, turbo Hadamard codes (THCs) [12], LDPC Hadamard codes [13] and concatenated zigzag Hadamard codes [14] have been shown to perform well even near the ultimate Shannon limit (i.e., -1.59 dB). These ultimate-Shannon-limit codes are applicable to multi-user environments, e.g., code-division multiple-access or interleave-division multiple-access (IDMA) [15] systems. In [16, 17], the hardware design of turbo Hadamard code has been investigated. Since Bahl-Cocke-Jelinek-Raviv (BCJR) decoding is required, the overall decoder design is relatively complex, limiting the throughput to less than 1 Gbps. On the other hand, the concatenated zigzag Hadamard codes do not require BCJR decoding, potentially making the decoder simpler and operating with a higher throughput.

In this paper, we investigate the hardware design of a concatenated zigzag Hadamard encoding/decoding system. We analyze the latency, throughput and utilization rate of the components. We implement the encoding/decoding system and compare the resources utilization and throughput with those of THC systems. The organization of the paper is as follows. Sect. II briefly reviews the structure of Hadamard code, zigzag Hadamard code and concatenated zigzag Hadamard code. Sect. III and Sect. IV present details of the hardware design of the concatenated zigzag Hadamard encoder and decoder, respectively. Sect. V shows the FPGA implementation results, including hardware utilization, throughput and bit error rate performance compared to the THC system. Finally, Sect. VI provides some concluding remarks.

II. CONCATENATED ZIGZAG HADAMARD CODE (CZHC)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3022537, IEEE Access

A. HADAMARD CODE

The codewords of an order-r Hadamard code are directly derived from Hadamard matrices of the same order. For example, Hadamard matrices of order $r = 3$ are given by

$$\pm\boldsymbol{H}_8 = \begin{bmatrix} \pm1 & \pm1 & \pm1 & \pm1 & \pm1 & \pm1 & \pm1 & \pm1 \\ \pm1 & \mp1 & \pm1 & \mp1 & \pm1 & \mp1 & \pm1 & \mp1 \\ \pm1 & \pm1 & \mp1 & \mp1 & \pm1 & \pm1 & \mp1 & \mp1 \\ \pm1 & \mp1 & \mp1 & \pm1 & \pm1 & \mp1 & \mp1 & \pm1 \\ \pm1 & \pm1 & \pm1 & \pm1 & \mp1 & \mp1 & \mp1 & \mp1 \\ \pm1 & \mp1 & \pm1 & \mp1 & \mp1 & \pm1 & \mp1 & \pm1 \\ \pm1 & \pm1 & \mp1 & \mp1 & \mp1 & \mp1 & \pm1 & \pm1 \\ \pm1 & \mp1 & \mp1 & \pm1 & \mp1 & \pm1 & \pm1 & \mp1 \end{bmatrix} \tag{1}$$

and Hadamard matrices of order-r are constructed recursively by

$$\pm\boldsymbol{H}_n = \begin{bmatrix} \pm\boldsymbol{H}_{n/2} & \pm\boldsymbol{H}_{n/2} \\ \pm\boldsymbol{H}_{n/2} & \mp\boldsymbol{H}_{n/2} \end{bmatrix} \tag{2}$$

with $n = 2^r$ and $\pm\boldsymbol{H}_1 = [\pm1]$. The codewords are given by the columns (or rows) of the Hadamard matrices $\pm\boldsymbol{H}_n$. For each codeword of length 2^r, the bit indices $\{0, 1, 2, 4, 8, \ldots, 2^{r-1}\}$ denote the positions of the information bits while the remaining indices are the locations of the parity-check bits.

We suppose an Hadamard codeword $\boldsymbol{c} = (c[0], c[1], c[2], \ldots, c[2^r - 1])$ is transmitted through an additive white Gaussian noise (AWGN) channel with noise mean 0 and variance σ^2. We also denote the noisy observation at the receiver by $\boldsymbol{x} = (x[0], x[1], x[2], \ldots, x[2^r - 1])$. The *a posteriori probability* (APP) logarithm-likelihood ratio (LLR) of the ith bit in the code is obtained by [12, 14]

$$\begin{aligned} L[i] =& \log\frac{\Pr(c[i] = +1|\boldsymbol{x})}{\Pr(c[i] = -1|\boldsymbol{x})} \\ =& \log\frac{\Pr(\boldsymbol{x}|c[i] = +1)\Pr(c[i] = +1)}{\Pr(\boldsymbol{x}|c[i] = -1)\Pr(c[i] = -1)} \\ =& \log\frac{\sum\limits_{c[i]=+1} \Pr(\boldsymbol{x}|\boldsymbol{c})}{\sum\limits_{c[i]=-1} \Pr(\boldsymbol{x}|\boldsymbol{c})}. \end{aligned} \tag{3}$$

Since the codewords are transmitted through an AWGN channel with noise variance σ^2, we have

$$\begin{aligned} L[i] &= \log\frac{\sum\limits_{c[i]=+1} \exp(-\frac{\|\boldsymbol{c}-\boldsymbol{x}\|^2}{2\sigma^2})}{\sum\limits_{c[i]=-1} \exp(-\frac{\|\boldsymbol{c}-\boldsymbol{x}\|^2}{2\sigma^2})} \\ &= \log\frac{\sum\limits_{c[i]=+1} \exp(\frac{<\boldsymbol{cx}>}{\sigma^2})}{\sum\limits_{c[i]=-1} \exp(\frac{<\boldsymbol{cx}>}{\sigma^2})}. \end{aligned} \tag{4}$$

The *a priori* information $\exp(\frac{<\boldsymbol{cx}>}{\sigma^2})$ can be calculated by an r-stage fast-Hadamard transform (FHT). After that, a same order dual-fast-Hadamard transform (DFHT) is applied to calculate (4).

B. ZIGZAG HADAMARD CODE

A zigzag Hadamard code (ZHC) is graphically described in Fig. 1(a) where each segment represents an order-r Hadamard code [14]. The overall code structure is also shown in Fig. 1(b). Assuming an information block $\boldsymbol{D}$ with length $L = rK$ is segmented into K sub-blocks. For the kth segment ($k = 1, 2, \ldots, K$), the information bits $\boldsymbol{d}_k = [d_k(1), d_k(2), \ldots, d_k(r)]$ are represented by blank nodes (area) and the remaining parity-check bits are represented by grey nodes (area). Moreover, the last parity bit of each segment is copied to the first input of the next segment and is denoted as the common bit (black nodes/area in the figures). Note that the first input bit of the first segment is fixed as 0 and is omitted.

Denote the Hadamard codeword in the kth segment as $\boldsymbol{c}_k = [c_k(0), \ldots, c_k(2^r - 1)]$, where $c_k(0) = c_{k-1}(2^r - 1)$ and $c_k(2^{j-1}) = d_k(j), j = 1, 2, \ldots, r$. We also denote the common bit $q_k = c_k(0) = c_{k-1}(2^r - 1)$ and the parity bits $\boldsymbol{p}_k = \{c_k(i), i \neq 0, i \neq 2^{j-1}, j = 1, 2, \ldots, r\}$. The kth segment of a ZHC codeword can then be rewritten as $\boldsymbol{c}_k = (\boldsymbol{d}_k, q_k, \boldsymbol{p}_k)$. The encoding process of ZHC is a Markov process and the correlation between any two consecutive segments depends only on the common bit. To decode the ZHC, a two-way decoding algorithm with two stages can be used [14].

1) Forward recursion: Starting from the first segment to the $(k-1)$th segment, perform FHT and DFHT on the current segment to obtain the APP LLRs of the bits based on the aforementioned discussion; then use the APP LLR of the last bit of the current segment to update the *a priori* LLR of the first bit of the next segment.
2) Backward recursion: Starting from the Kth segment to the first segment, perform FHT and DFHT on on the current segment to obtain the APP LLRs of the bits (including information bits); then use the extrinsic LLR of the first bit of the current segment to update the *a priori* LLR of the last bit of the previous segment.

C. CONCATENATED ZIGZAG HADAMARD CODE

Fig. 2 shows the code structure of a CZHC [14] with M component codes. (When the zigzag Hadamard encoders in Fig. 2 are replaced by convolutional Hadamard encoders, the output codeword becomes a THC [12].) M copies of the same but interleaved information bits are sent to M zigzag Hadamard encoders producing M copies of parity bits. The information $\boldsymbol{D}$ together with the parity bits $\boldsymbol{p}^{(1)}, \boldsymbol{p}^{(2)}, \ldots, \boldsymbol{p}^{(M)}$ are sent to the channel.

The encoder design of CZHC will be discussed in Section III. The decoding of CZHC involves the interleaving and passing of LLRs among different zigzag Hadamard codes (or component codes) and will be explained in Section IV.

III. CZHC ENCODER DESIGN

The data flow of our CZHC encoder/decoder system is shown in Fig. 3. The structure of the CZHC encoder is shown in Fig. 4 where M components of ZHC are encoded in parallel. To generate each CZHC codeword, the following steps are performed.

1) Generate random information bits of length rK using a pseudo random number generator (PRNG), which is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3022537, IEEE Access

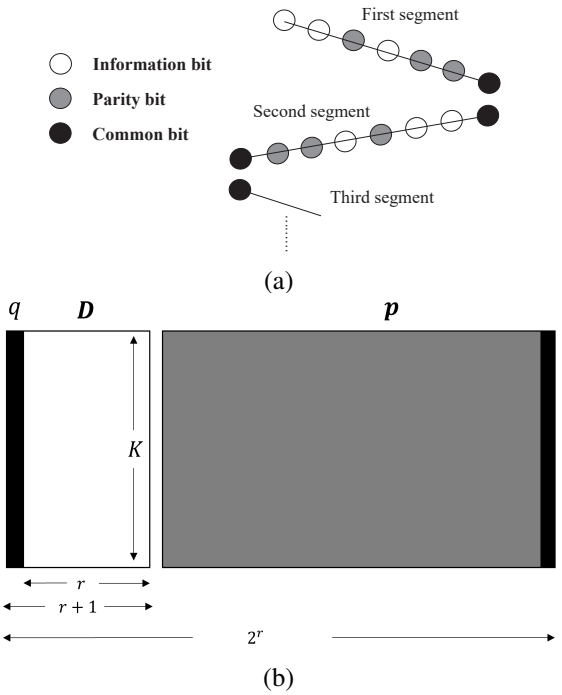


FIGURE 1. A zigzag Hadamard code. (a) Graphical representation and (b) overall structure. White: information bits; grey: parity bits; black: common bits.

realized by the use of linear feedback shift registers (LFSRs). Form the first component code using Step 2) below.

2) Divide the information bits into segments of length r. The r information bits in each segment together with the common bit are then sent to the Hadamard encoder, producing Hadamard codewords. Note that the common bit is the feedback of the Hadamard encoder from the last segment. For the first segment, the common bit is set to 0.
3) For each of the $M-1$ component codes, send the original information bits to the corresponding interleaver denoted by $\prod_1, \prod_2, \ldots, \prod_{M-1}$, respectively; and apply Step 2) above.
4) Send the original information bits together with all the parity-check bits generated from all component encoders to the channel.

IV. CZHC DECODER DESIGN

The structure of the decoder is illustrated in Fig. 5. The decoding process includes:

1) Preparation: (a) The *a priori* LLRs of the information bits, computed by subtracting the corresponding extrinsic LLRs of the information bits produced by the current decoder in the previous iteration from the APP LLRs of the information bits from the *previous* component (ZHC) decoder; and (b) channel LLRs of the parity bits of the current component ZHC code, are input to the decoder.
2) Forward recursion: The *a priori* LLRs are sent to the decoder to perform forward recursion. The forward recursion processor consists of an order-r FHT block and an order-r DFHT block. The *a priori* LLRs (2^r for each segment of ZHC) are directly input to the FHT block where simple addition/subtraction operations are performed to produce 2^r outputs after r stages. Exponential functions are performed to the 2^r outputs and their additive inverse (total 2^{r+1} data) before sending them to the DFHT block, which also produces 2^{r+1} outputs after r stages. Then divisions are performed to the 2^{r+1} outputs to generate 2^r APP LLRs. The above operations are used to realize (4). Note that the exponential functions greatly increase the dynamic range of data in DFHT block and a large number of quantization bits is required to maintain the accuracy of decoding in DFHT block. To avoid the implementation of complicated exponential functions, we use logarithm quantization in the DFHT block. The benefits of the quantization include:
   - turning the exponential functions between two blocks into simple bitwise-NOT functions;
   - reducing the dynamic range of operations in DFHT block and hence the number of bits used to quantize the LLRs;
   - simplifying the decision block from division logics to subtraction logics.

   An illustration of the proposed APP decoder for ZHC with order-2 is shown in Fig. 6.
3) Backward recursion: The backward recursion also consists of an order-r FHT block and an order-r DFHT block. Thus the APP decoding processors in the forward recursion can be reused. The backward recursion processor starts outputting the APP LLR continuously after $2r$ clocks delay. The outputs start from the Kth segment and then all the way to the first segment.
4) The output data from the backward recursion processor are interleaved and passed to the next sub-decoder.
5) The extrinsic LLRs of the information bits in this iteration are generated and stored in the RAMs at the same time.

The FHT/DFHT blocks are implemented in the CZHC decoder to fast calculate the APP LLRs [18]. Each segment in both the forward and backward recursions must wait for the update from the previous (next) segment before continuing decoding. For each of the K segments, the FHT and DFHT processors take a total of $2r$ clocks to complete the computations.

As shown in Fig. 7, only one of the $2r$ stages is working at any time. To better utilize the decoder hardware and to improve the throughput, we decode $2r$ CZHCs at the same time in our design in a pipeline manner. These $2r$ CZHCs are sent into the decoder segment-by-segment, i.e., first segment of the first code is sent to the decoder, followed by the first segment of the second code, and so on. After the first segments of all $2r$ CZHCs are sent, the second segments

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3022537, IEEE Access

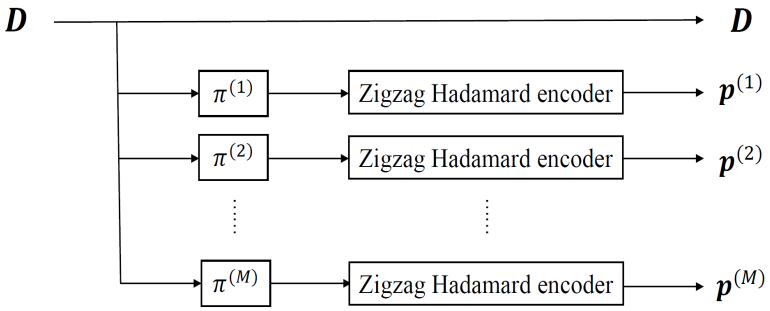


FIGURE 2. Structure of a concatenated zigzag Hadamard code.

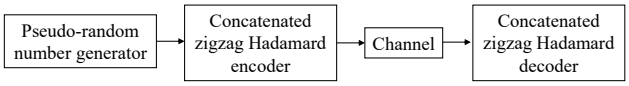


FIGURE 3. Data flow of the CZHC encoder/decoder system.

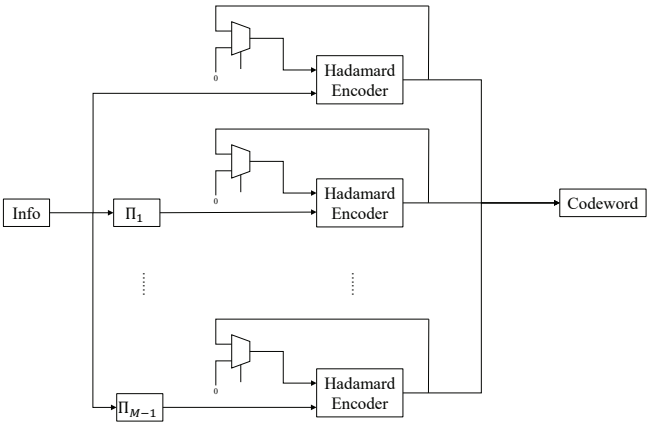


FIGURE 4. Overview of CZHC encoder.

of the $2r$ CZHCs are sent. Note that the time when the DFHT processor finishes computing the APP LLR of the first segments of the $2r$ CZHCs, the *a priori* (AP) LLR of the second segments are arriving at the decoder. Both LLRs will then be sent to the FHT/DFHT processors to compute the forward recursion of the second segment. The operations of the FHT/DFHT processors for $2r$ CZHCs are illustrated in Fig. 8. The utilization rate of the FHT/DFHT processors are therefore greatly improved. Moreover, the latency (time between the last input code bit entering the decoder and the last decoded bit coming out of decoder) of each CZHC codeword is actually the same as that of decoding a single CZHC and the throughput of the decoder is increased by $2r$ times.

Note that the CZHC is a concatenated code with M component codes. Each CZHC codeword needs to go through the decoding process in Fig. 5 M times to complete one iteration. To simplify the control logic and to increase the throughput, we construct M CZHC decoders (each called a sub-decoder) in our decoding system. Hence, M times more CZHC codewords can be decoded simultaneously in a pipeline manner. The usage of control logic and block RAMs between consecutive component code decoders are reduced and the throughput of the decoding system is increased by another M times, i.e., a total of $2rM$ times. Fig. 9 shows the decoding system that consists of M sub-decoders. To decode $2rM$ CZHCs simultaneously, the decoder receives and stores the $2rM$ CZHCs in both the information RAMs and the parity RAMs which are shown in Fig. 10.

Between consecutive sub-decoders, interleavers (omitted in Fig. 9 for simplicity) are needed to shuffle the outputs of the current sub-decoder before inputting them to the next sub-decoder. We use fixed inter-window shuffle (FIWS) interleavers to enable parallel interleaving [17, 19]. The size of the interleaver is $N = 2r \times L = 2Kr^2$ because we need to perform interleaving on the information bits of $2r$ CZHC codes at the same time. The interleaver is divided into r sub-interleavers (also called windows) each with a window size of $2rK$. The windows are designed in such a way that memory contention is avoided when performing parallel interleaving. In other words, the first information bits of all K segments in all $2r$ CZHCs are interleaved/deinterleaved in Window No. 1; the second information bits of all K segments in all $2r$ CZHCs are interleaved/deinterleaved in Window No. 2; etc.

The FIWS interleaver is realized by r width-$(r \times N_{FHT})$ depth-$2rK$ RAMs and a depth-K ROM. The operations are described below and illustrated in Fig. 11.

1) Store the output APP LLRs from the ith sub-decoder ($i = 1, 2, \ldots, M$) to the RAMs in the order that is shown in Fig. 11, i.e., the jth bit of all the K segments of all the $2r$ codes are stored in the jth RAM ($j = 1, 2, \ldots, r$).
2) Read the interleaver patterns of the CZHC code from the ROM. Extract the interleaving information and evaluate the interleaver pattern for all the $2r$ CZHC codes in the r RAMs correspondingly.
3) Read the interleaved APP LLRs from the r different RAMs. Regroup the APP LLRs and send them to the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3022537, IEEE Access

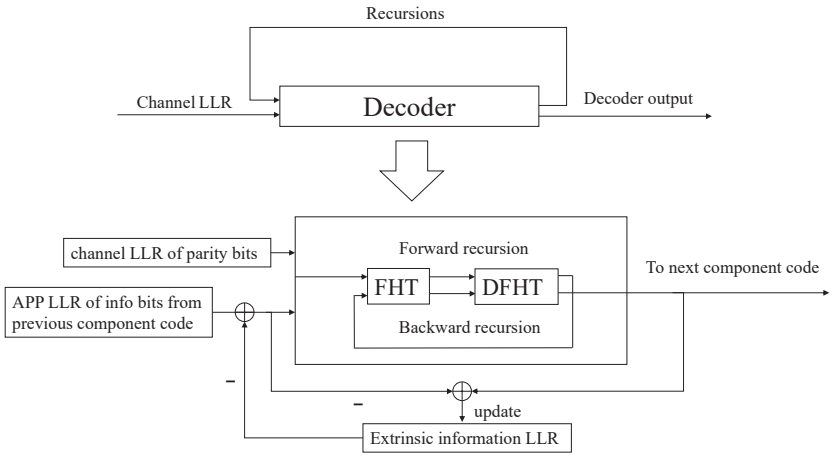


FIGURE 5. Overview of CZHC decoder.

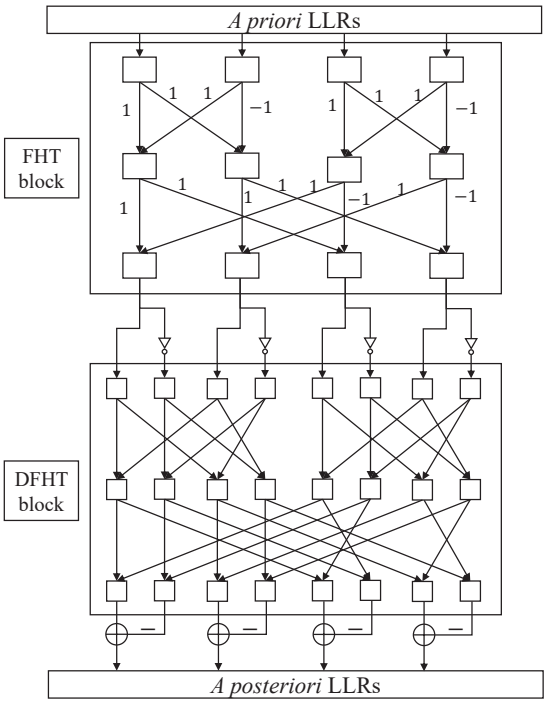


FIGURE 6. Detailed illustration of ZHC APP decoder with order-2.

next sub-decoder as the *a priori* LLRs.

V. IMPLEMENTATION RESULTS AND ANALYSIS

For an order-r CZHC, the code length is $l = rK + MK(2^r - r)$ and the code rate is $r_c = \frac{rK}{rK+MK(2^r-r)} = \frac{r}{r+M(2^r-r)}$. A concatenated zigzag Hadamard encoder/decoder system with the following parameters is implemented.

- Number of component codes $M = 3$
- Each information block $\boldsymbol{D}$ contains $L = 3510$ message bits
- Hadamard order is $r = 6$
- Number of segments per CZHC $K = \frac{L}{r} = 585$
- Number of CZHC codewords decoded in a sub-decoder $n = 2r = 12$
- Code length of one CZHC $l = 105300$
- Code rate $r_c = 0.0333$
- Channel LLRs are quantized by $N_{ch} = 6$ bits
- Inputs to FHT unit are quantized by $N_{FHT} = 10$ bits
- Data in DFHT unit are quantized by $N_{DFHT} = 11$ bits
- $I = 10$ iterations used for decoding each codeword
- Operating frequency $f_c = 150MHz$
- FPGA board Xilinx Virtex UltraScale+ VCU118

Fig. 12 shows the hardware utilization of each unit inside the sub-decoder. The FHT and DFHT units are both utilized in forward and backward recursions. Denoting the hardware utilization rate of the FHT, DFHT and interleaver as U_{FHT}, U_{DFHT} and U_π, respectively. Note that $\frac{1}{K} \to 0$, we have

$$U_{FHT} = U_{DFHT} = \frac{2rK - r + 2rK + r}{4rK + 2r} = \frac{1}{1 + \frac{1}{2K}} \approx 1;$$
$$U_\pi = \frac{2rK}{4rK + 2r} = \frac{1}{2}\frac{1}{1 + \frac{1}{2K}} \approx \frac{1}{2}. \tag{5}$$

The FHT/DFHT units work almost all the time during decoding while the interleavers operate approximately half of the time.

Referring to Fig. 12, it takes $4rK + 2r = 2r(1 + 2K)$ clocks to process one component code in each sub-decoder. Assuming the total number of iterations is I and the number of component codes for each CZHC code is M, it takes $2rIM(1 + 2K)$ clocks to decode the $2r$ CZHC. Also assuming the operating frequency of the sub-decoder is f_c and the CZHC code length is l, the sub-decoder can decode $\frac{2rlf_c}{2rIM(1+2K)}$ bits in one second. Moreover, the decoder consists of M sub-decoders and can decodes $2rM$ CZHC in parallel. The throughput of the whole decoding system is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3022537, IEEE Access

Clock	1	2	3	…	r	$r+1$	$r+2$	…	$2r$	$2r+1$	$2r+2$	…
Stage 1 of FHT	working	idle	idle	…	idle	idle	idle	…	idle	working	idle	…
Stage 2 of FHT	idle	working	idle	…	idle	idle	idle	…	idle	idle	working	…
Stage 3 of FHT	idle	idle	working	…	idle	idle	idle	…	idle	idle	idle	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Stage r of FHT	idle	idle	idle	…	working	idle	idle	…	idle	idle	idle	…
Stage 1 of DFHT	idle	idle	idle	…	idle	working	idle	…	idle	idle	idle	…
Stage 2 of DFHT	idle	idle	idle	…	idle	idle	working	…	idle	idle	idle	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Stage r of DFHT	idle	idle	idle	…	idle	idle	idle	…	working	idle	idle	…

FIGURE 7. Illustration of a forward recursion for a single CZHC code.

Clock	1	2	3	…	r	$r+1$	$r+2$	…	$2r$	$2r+1$	$2r+2$	…
Stage 1 of FHT	code 1	code 2	code 3	…	code r	code $r+1$	code $r+2$	…	code $2r$	code 1	code 2	…
Stage 2 of FHT	idle	code 1	code 2	…	code $r-1$	code r	code $r+1$	…	code $2r-1$	code $2r$	code 1	…
Stage 3 of FHT	idle	idle	code 1	…	code $r-2$	code $r-1$	code r	…	code $2r-2$	code $2r-1$	code $2r$	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Stage r of FHT	idle	idle	idle	…	code 1	code 2	code 3	…	code r	code $r+1$	code $r+2$	…
Stage 1 of DFHT	idle	idle	idle	…	idle	code 1	code 2	…	code $r-1$	code r	code $r+1$	…
Stage 2 of DFHT	idle	idle	idle	…	idle	idle	code 1	…	code $r-2$	code $r-1$	code r	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Stage r of DFHT	idle	idle	idle	…	idle	idle	idle	…	code 1	code 2	code 3	…

FIGURE 8. Forward recursion with $2r$ CZHC codes.

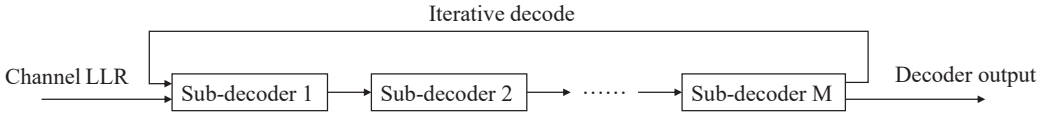


FIGURE 9. M sub-decoders in one decoding system.

Code type	FHT/DFHT	Interleaver utilization rate	BCJR unit utilization rate	Throughput	Approximate latency/sub-decoder
CZHC	1	0.5	N.A.	1.44 Gbps	9.76μs
THC	0.5	0.5	1	0.96 Gbps	1.22μs

TABLE 1. Hardware utilization rate, throughput and latency of CZHC system and THC system.

approximated by

$$T = \frac{M \times 2rlf_c}{2rIM(1+2K)} = \frac{[rK + MK(2^r - r)]f_c}{I(1+2K)} = \frac{[(1-M)r + 2^r M]f_c}{I(2+\frac{1}{K})} \approx \frac{2^{r-1}Mf_c}{I}$$

where the approximation is made because $1/K \ll 1$ and $(M-1)r \ll 2^r M$.

The decoding path in the THC decoder involves going through the FHT block, the BCJR block and then the DFHT block with a latency of approximately $2K$. The decoding path of the CZHC decoder includes the FHT/DFHT block of the forward recursion and then the FHT/DFHT block of the backward recursion with a latency of approximately $4rK$. The latency of CZHC decoder is $2r$ times higher because the forward/backward recursions in CZHC decoder which cannot be performed in pipeline for one single code. However, it is possible to perform pipeline decoding of multiple codes, which is realized in our design.

In Table 1 and Table 2, we compare the FPGA implementation results of the encoder/decoder systems using CZHC and turbo Hadamard code (THC) [17] under the same code rate and code length. Table 1 indicates that BCJR processor is not required in the CZHC decoder. The throughput of the CZHC system is 50% higher than that of the THC system. Table 2 shows that with the same code length and code rate, the look-up tables (LUT), look-up table RAMs and flip-flops used in the CZHC system are, respectively, 73%, 41% and 85% of those in the THC system. The block RAM usage in CZHC system however, is higher than that in the THC system.

Fig. 13 shows the bit-error-rate (BER) results of CZHC and THC. Compared with floating-point CZHC decoder, fixed-point decoder shows a performance loss of about 0.1 dB at BER$= 2 \times 10^{-5}$ when 5-bit or 6-bit quantized channel LLRs are used. For even smaller number of quantization bits, the BER performance is further degraded. Fig. 13 also shows that the BER performances of CZHC and THC are very close. Both codes can achieve BER$= 1.5 \times 10^{-5}$ at

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3022537, IEEE Access

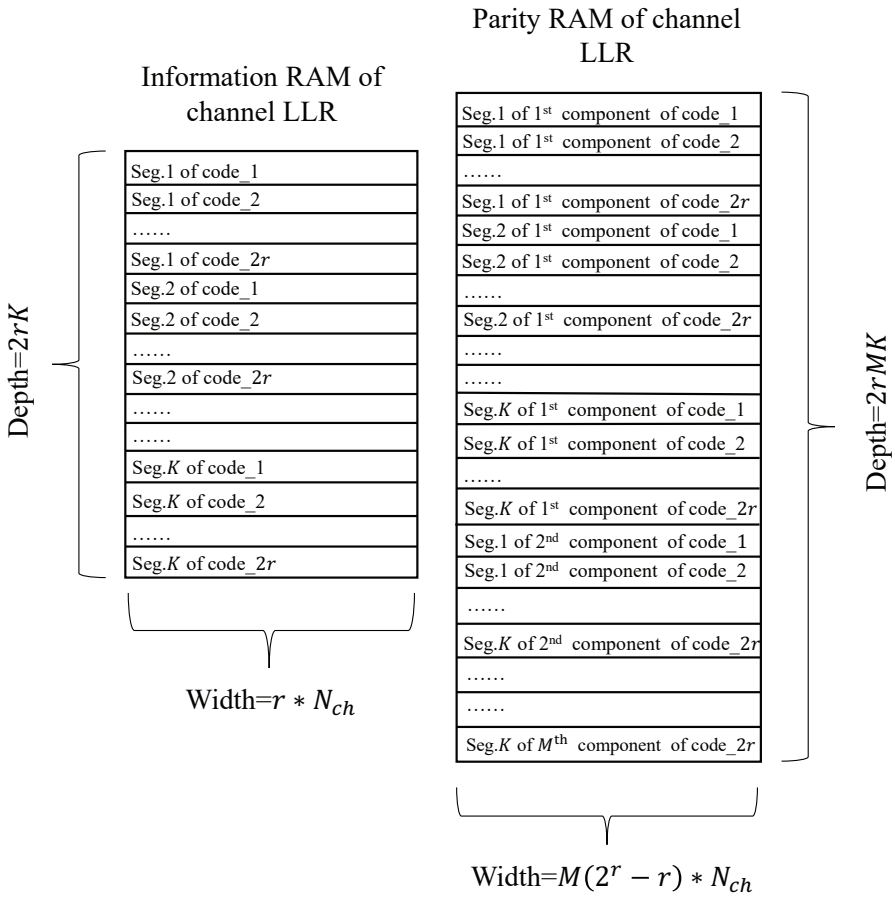


FIGURE 10. Storage order of channel LLRs.

Code type	Code rate	Code length	Look-up Table count (normalized)	Look-up Table RAM count (normalized)	Flip-Flop count (normalized)	Block RAMs count (normalized)	Max. Operating frequency
THC	0.0333	105300	1	1	1	1	100 MHz
CZHC	0.0333	105300	0.737	0.417	0.856	5.180	150 MHz

TABLE 2. Resources utilization of CZHC system compared to THC system. 6-bit quantized channel LLRs are used.

$E_b/N_0 = -0.2$ dB. Fig. 14 shows the BER results under different number of decoding iterations. We observe that THC outperforms CZHC in low E_b/N_0 region, and performs similar in high E_b/N_0 region. Both codes show an error floor at a BER of 10^{-5}.

VI. CONCLUSION

An efficient design of an ultimate-Shannon-limit approaching encoder/decoder system based on CZHC has been explored using FPGA. It can achieve a throughput of 1.44 Gbps at a code rate of 0.0333 and BER$= 1.5 \times 10^{-5}$ at $E_b/N_0 = -0.2$ dB. Compared to the THC system, the CZHC system achieves 1.5 times larger throughput with less complex hardware architecture but more block RAM usage. The main drawback of the CZHC system is a higher decoding latency. Future research work should aim at reducing the latency of the CZHC decoder.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding," in IEEE International Conference on Communications, vol. 2, pp. 1064–1070, May 1993.

[2] D. Chandra, Z. Babar, S. X. Ng, and L. Hanzo, "Near-hashing-bound multiple-rate quantum turbo short-block codes," IEEE Access, vol. 7, pp. 52712–52730, 2019.

[3] K. S. Vishvaksenan and P. T. V. Raj, "Coded downlink multi-user MC-CDMA system using transmitter pre-processing: Performance results," IEEE Access, vol. 4, pp. 4534–4542, 2016.

[4] R. Gallager, "Low-density parity-check codes," IRE Transactions on Information Theory, vol. 8, no. 1, pp. 21–28, 1962.

[5] L. Kong, Y. Liu, H. Liu, and S. Zhao, "Protograph QC-LDPC and rate-adaptive polar codes design for MLC NAND flash memories," IEEE Access, vol. 7, pp. 37131–37140, 2019.

[6] B. Wang, P. Chen, Y. Fang, and F. C. M. Lau, "The design of vertical RS-CRC and LDPC code for ship-based satellite communications on-the-move," IEEE Access, vol. 7, pp. 44977–

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3022537, IEEE Access

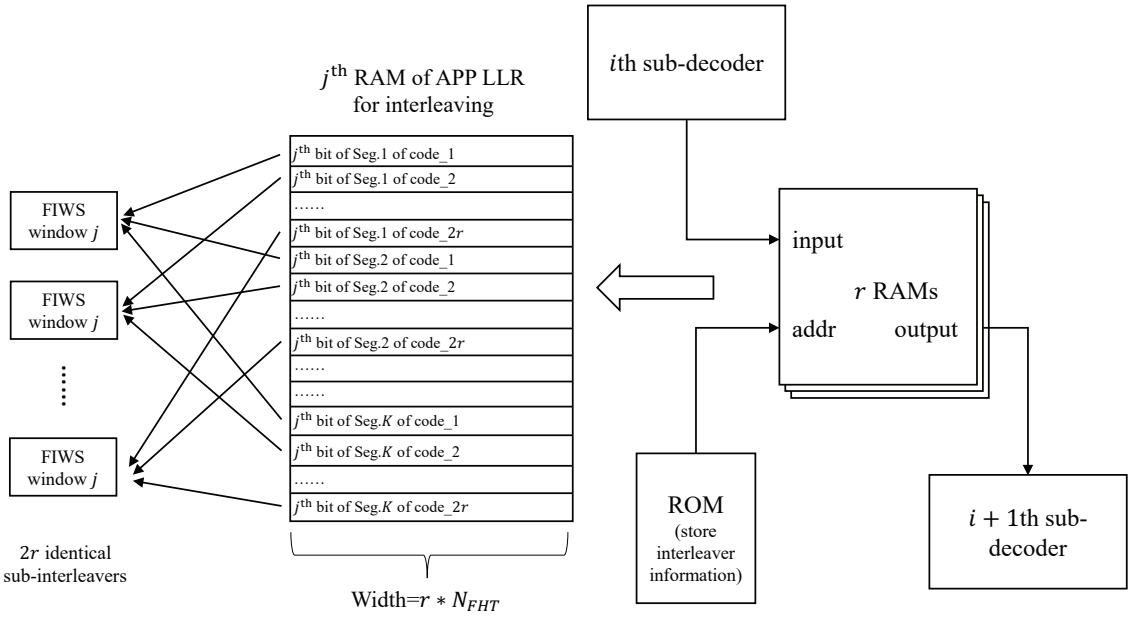


FIGURE 11. Illustration of the FIWS interleaver.

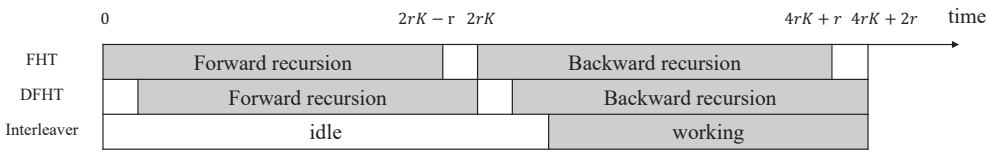


FIGURE 12. Component utilization of each sub-decoder.

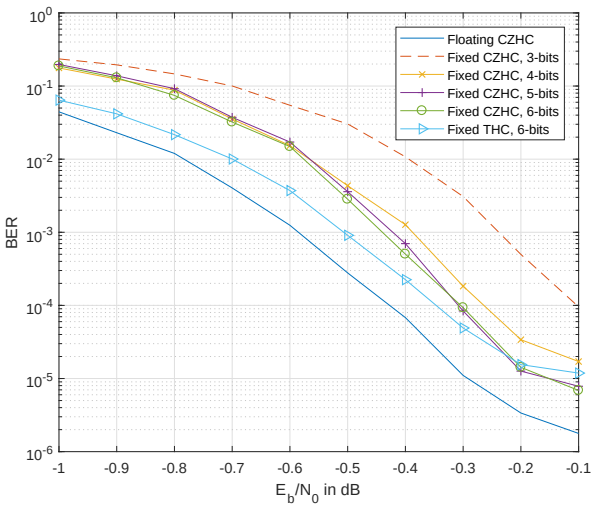


FIGURE 13. BER results of CZHC and THC under different quantization bits.

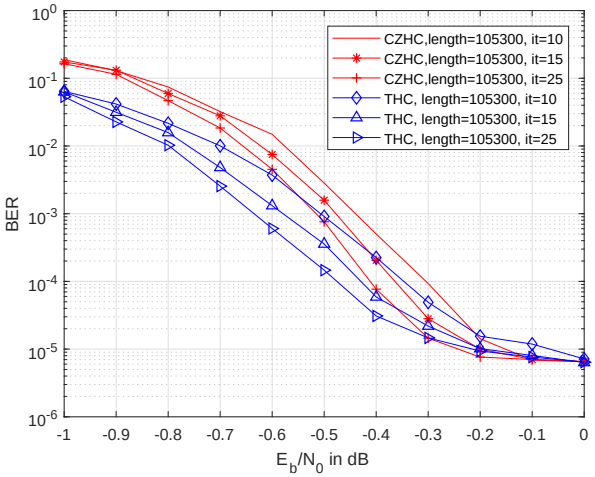


FIGURE 14. BER results of CZHC and THC under different number of iterations.

44986, 2019.

[7] H. D. Vu, T. V. Nguyen, D. N. Nguyen, and H. T. Nguyen, "On design of protograph LDPC codes for large-scale MIMO systems," IEEE Access, vol. 8, pp. 46017–46029, 2020.

[8] V. Guruswami and P. Xia, "Polar codes: Speed of polarization and polynomial gap to capacity," IEEE Transactions on Information Theory, vol. 61, pp. 3–16, Jan 2015.

[9] Q. Yu, Z. Shi, X. Li, J. Du, J. Zhang, and K. M. Rabie, "On the concatenations of polar codes and non-binary LDPC codes,"

IEEE Access, vol. 6, pp. 65088–65097, 2018.

[10] X. Xu, S. Wu, D. Dong, J. Jiao, and Q. Zhang, "High performance short polar codes: A concatenation scheme using spinal codes as the outer code," IEEE Access, vol. 6, pp. 70644–70654, 2018.

[11] P. Chen, B. Bai, Z. Ren, J. Wang, and S. Sun, "Hash-polar codes with application to 5G," IEEE Access, vol. 7, pp. 12441–12455, 2019.

[12] L. Ping, W. K. Leung, and K. Y. Wu, "Low-rate turbo-Hadamard codes," IEEE Transactions on Information Theory, vol. 49, pp. 3213–3224, Dec 2003.

[13] G. Yue, L. Ping, and X. Wang, "Generalized low-density parity-check codes based on Hadamard constraints," IEEE Transactions on Information Theory, vol. 53, pp. 1058–1079, March 2007.

[14] W. K. R. Leung, G. Yue, L. Ping, and X. Wang, "Concatenated zigzag Hadamard codes," IEEE Transactions on Information Theory, vol. 52, pp. 1711–1723, April 2006.

[15] Li Ping, Lihai Liu, Keying Wu, and W. K. Leung, "Interleave division multiple-access," IEEE Transactions on Wireless Communications, vol. 5, pp. 938–947, April 2006.

[16] S. Jiang, P. W. Zhang, F. C. M. Lau, C. W. Sham, and K. Huang, "A turbo-Hadamard encoder/decoder system with hundreds of Mbps throughput," in 2018 IEEE 10th International Symposium on Turbo Codes Iterative Information Processing (ISTC), pp. 1–5, 2018.

[17] K. Y. N. Shimanuki, B. Kurkoski and K. Kobayash, "Improvements and extensions of low-rate turbo-Hadamard codes," in Proceedings of ISITA, Octorber 2006.

[18] L. Ping and S. Chan, "Iterative decoding of concatenated Hadamard codes," in ICC '98. 1998 IEEE International Conference on Communications. Conference Record. Affiliated with SUPERCOMM'98 (Cat. No.98CH36220), vol. 1, pp. 136–140 vol.1, 1998.

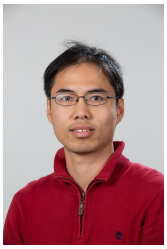
[19] A. Nimbalker, T. K. Blankenship, B. Classon, T. E. Fuja, and D. J. Costello, "Contention-free interleavers for high-throughput turbo decoding," IEEE Transactions on Communications, vol. 56, pp. 1258–1267, August 2008.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3022537, IEEE Access

SHENG JIANG received the Bachelor of Engineering Degree in Microelectronics from Shanghai Jiaotong University, China, and the Master of Engineering Degree in Electronic Engineering from Hong Kong University of Science and Technology, Hong Kong. He is currently pursuing his PhD degree at The Hong Kong Polytechnic University, Hong Kong.

FRANCIS C. M. LAU received the BEng(Hons) degree in electrical and electronic engineering and the PhD degree from King's College London, University of London, UK. He is a Professor at the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong. He is also a Fellow of IET and a Senior Member of IEEE.

He is the co-author of Chaos-Based Digital Communication Systems (Heidelberg: Springer-Verlag, 2003) and Digital Communications with Chaos: Multiple Access Techniques and Performance Evaluation (Oxford: Elsevier, 2007). He is also a co-holder of five US patents and one pending US patent. He has published more than 300 papers. His main research interests include channel coding, cooperative networks, wireless sensor networks, chaos-based digital communications, applications of complex-network theories, and wireless communications. He was the General Co-chair of International Symposium on Turbo Codes & Iterative Information Processing (2018) and the Chair of Technical Committee on Nonlinear Circuits and Systems, IEEE Circuits and Systems Society (2012-13). He served as an associate editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II (2004-2005 and 2015-2019), IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I (2006-2007), and IEEE CIRCUITS AND SYSTEMS MAGAZINE (2012-2015). He has been a guest associate editor of INTERNATIONAL JOURNAL AND BIFURCATION AND CHAOS since 2010.



CHIU-WING SHAM received a Bachelor degree (Computer Engineering) and an MPhil. degree from the Chinese University of Hong Kong in 2000 and 2002 respectively, and received the Ph.D. degree from the same university in 2006. He has worked as an Electronic Engineer on the FPGA applications of the motion-control system and system security with cryptography in ASM Pacific Technology Ltd (HK). During the years at the Hong Kong Polytechnic University, he has have also engaged in various University projects for the commercialization of technology, in particular, a few optical communication projects which are in collaboration with Huawei. He also worked on the physical design of VLSI design automation. He was invited to work at Synopsys, Inc (Shanghai) in the summer of 2005 as a Visiting Research Engineer. He is now working at The University of Auckland as Senior Lecturer. He is also an IEEE Senior Member and the Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II (2017-present).