INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2020, VOL. 66, NO. 3, PP. 545–553
Manuscript received March 2, 2020; revised July, 2020. DOI: 10.24425/ijet.2020.134011

Fixed-latency System for High-speed Serial Transmission Between FPGA Devices with Forward Error Correction

Michał Kruszewski, and Wojciech Marek Zabołotny

Abstract—**This paper presents the design of a compact protocol for fixed-latency, high-speed, reliable, serial transmission between simple field-programmable gate arrays (FPGA) devices. Implementation of the project aims to delineate word boundaries, provide randomness to the electromagnetic interference (EMI) generated by the electrical transitions, allow for clock recovery and maintain direct current (DC) balance. An orthogonal concatenated coding scheme is used for correcting transmission errors using modified Bose–Chaudhuri–Hocquenghem (BCH) code capable of correcting all single bit errors and most of the double-adjacent errors. As a result all burst errors of a length up to 31 bits, and some of the longer group errors, are corrected within 256 bits long packet. The efficiency of the proposed solution equals 46.48%, as 119 out of 256 bits are fully available to the user. The design has been implemented and tested on Xilinx Kintex UltraScale+ KCU116 Evaluation Kit with a data rate of 28.2 Gbps. Sample latency analysis has also been performed so that user could easily carry out calculations for different transmission speed. The main advancement of the work is the use of modified BCH(15, 11) code that leads to high error correction capabilities for burst errors and user friendly packet length.**

Keywords—**data transmission, fixed-latency transmission, forward error correction, orthogonal concatenated coding, FPGA**

I. Introduction

IN modern, complex electronic systems inter-chip and inter-board communication are often trivialized. However, it is one of the fundamental modules of advanced FPGA based solutions, which can potentially be the bottleneck for data processing. As a proof, one can look at the area of application that includes: artificial intelligence, machine learning, image processing, 5G mobile network, advanced driver assistance systems (ADAS), radar communication, high energy physics experiments (HEP). The most challenging applications that work in a harsh environment, are space explorations, satellite communications, and HEP experiments. Due to omnipresent, high volume radiation, the probability of transmission errors and data corruption is at an unacceptable level, especially in case of high-speed transmission. In order to achieve reliable and efficient communication it is necessary to merge multiple protection mechanisms.

A typical solution providing the reliable data transfer is the acknowledgment and retransmission system, where the receiver must acknowledge each data frame transmitted by the sender after a successful and error-free reception. If the receiver receives a corrupted frame, it sends the negative acknowledgment, triggering immediate retransmission. If the frame is corrupted so badly that it is lost, the acknowledgment is never sent. Therefore the transmitter must retransmit all not acknowledged frames after a certain timeout. To avoid stalling the transmission before a frame is acknowledged, usually, a certain amount of next frames is transmitted without waiting for the acknowledgment [1], [2]. The described mode of operation increases the complexity of the transmission and has following disadvantages:

- The transmission requires bidirectional communication link.
- The transmission is not time-deterministic, because re-transmission of the particular frame (especially after the timeout) results in a significant delay.
- The transmitter must buffer all the frames that are transmitted and not yet acknowledged. That increases the memory consumption, which in FPGA is a scarce resource.
- The receiver may obtain the frames out of order. Therefore, if the data should be further processed as an ordered data stream, it is necessary to buffer the received frames at the receiver side, until all previous frames are successfully delivered.

The above properties of the reliable data transfer based on acknowledgment and retransmission may be unacceptable in certain applications. If we need “almost reliable” data transport with time-deterministic latency and a possibility to operate via a unidirectional link, then the forward error correction based system is the right solution.

A. State of the art

FPGA vendors offer intellectual property (IP) blocks for the forward error correction as well as high-speed transmission protocols. While error correction IP blocks can be used as building blocks, protocol IP blocks are theoretically ready to use. One of the most popular is the Interlaken protocol [3], which is based on SPI4.2 and XAUI. It utilizes simple but very efficient ways for delineating word boundaries, providing randomness to the EMI, maintaining DC balance and clock recovery. The most important are synchronous scrambler and 64b/67b encoding. Xilinx Interlaken IP core enables transmission speeds from 10 Gbps up to 150 Gbps when multiple

The work has been partially supported by the statutory funds of the Warsaw University of Technology the Institute of Electronic Systems.

M. Kruszewski and W.M. Zabołotny are with Warsaw University of Technology, the Institute of Electronic Systems, Warsaw, Poland (e-mail: mkruszew@elektron.elka.pw.edu.pl, wzab@ise.pw.edu.pl).

serial lanes are used. The main drawback of the Interlaken protocol is lack of the forward error correction mechanism. It is also not so easy to add any because these IP blocks are tightly integrated with GT (Gigabit Transceiver) blocks and any modifications (improvements) require interference into IP blocks.

Probably the most popular solution in HEP experiments is the GBTX link interface [4]. It offers three different frame formats for data transmission. Only the default one (the so-called "GBT frame format") uses the forward error correction. The frame is 120-bits long with 84 bits for user data. Error correction scheme is built by interleaving two Reed-Solomon encoded words with 4-bit symbols, each capable of correcting a double symbol error. This in practice means that a sequence of up to 16 consecutive corrupted bits can be corrected. This correction technique consumes 32 bits out of 120-bits long frame.

Authors in [5] propose high-speed error resilient communication protocol intended to be used in HEP experiments. They use orthogonal concatenation of error correcting codes to protect from transmission errors and CRC sum in order to detect invalid packets. Two very simple error correction codes are used: BCH(15, 11) and Hamming(7, 4). Unfortunately, the result of such a choice is a non-standard length of a single data packet, which equals 44 bits before encoding and 93 bits after encoding (only user data is orthogonally encoded that is why encoded block length is 93 bits, not $15 \times 7 = 105$). Size of the frame was adjusted to the size of the data generated for single collision detection (i.e. energy, timestamp and the position at which the particle hit the detector). Use of dedicated gigabit transceivers, available in Xilinx chips, required the addition of 3 bits so that the final block length equals 96 bits. The efficiency of error correcting scheme equals $\frac{44}{96} \cdot 100\% = 45,83\%$. Synchronous scrambler is used for minimizing the probability of long sequences of '0' or '1' value bits. From the point of view of general use, the main drawback is quite short, non-standard block size. Authors have achieved the maximum data rate of 5 Gbps. The publication lacks complete analysis of error correcting capabilities, especially for group errors.

In paper [6] authors propose serial link with forward error correction for JESD204B standard. This solution is customized for 16 bits words returned by analog-to-digital converters (ADC) and is not able to correct long burst errors. However, it is very efficient in case of frequent short burst errors (up to 4 bits but not more often than every 24 bits).

[7] introduces unified communication framework that uses 8/10-b encoding without any error correction. It also describes the network layer, while it is sometimes desired to leave user with full choice in terms of network layer architecture.

Paper [8] proposes an optical link between two computing nodes with peripheral component interconnect express (PCIe) interface. Authors have achieved a data rate of 8.5 Gbps with hard PCIe IP block available in Stratix IV FPGA board. The solution does not introduce any error correction scheme.

In [9] author introduces a simple and interesting solution for error multiplication problem in synchronous scramblers on 64/66 bit encoded links. The method is based on a calculation of four CRC16 sums, each of them is calculated for two adjacent 64-bit blocks. The syndrome table is then used to correct a single error, even if the error gets multiplied in the next data block. Unfortunately, the proposed scheme is weak for handling group errors. What is more, the algorithm is protected by the patents.

Authors in [10] propose a flexible FPGA-to-FPGA communication system capable of transmitting different bus protocols (PICe, Ethernet and SRIO). Although such additional transmission layer is useful and provides unified physical layer, it leads to increase FPGA resource utilization. It also does not introduce any extra error correction mechanisms.

Authors of paper [11] suggest to use Aurora protocol with DDR3 SDRAM. The RAM memory serves as a large FIFO, that prevents data loss if the user data rate is larger than the Aurora transmission rate. Authors do not mention how transmission errors should be handled.

Paper [12] analyzes high-speed serial communication physical layer protocols that could be used for high-performance computing clusters (HPCC) based on FPGAs. Authors have choosen ten-gigabit attachment unit interface (XAUI) as the most appropriate in terms of latency. The main drawback is that XAUI does not offer error correction.

II. Proposed design

This section describes in detail different levels of implementation and explains all taken design decisions.

A general block diagram of the proposed design is shown in Fig.1. It shows only crucial modules that are necessary for achieving link synchronization and user data transfer. It lacks cyclic redundancy check (CRC16) and delay blocks for control signals. RX FIFO can be placed at the input (as shown in the figure) or at the output of the receiver block, depending on the desired system behavior on flow control signals. When RX FIFO is at the input, the actual stop of data reception is delayed because flow control information needs to be delivered to the opposite node. Placing RX FIFO at the output leads to immediate suspension of the reception at the expense of incomplete FIFO emptying.

The basic data block in the physical transmission channel is a frame with a length of 256 bits. This decision was motivated by two factors. The first one is the width of user data accepted by serializer/deserializer (SerDes) modules available in FPGA chips. In case of encoding implemented outside of the dedicated blocks, these modules accept data of width being multiples of 8 bits (32, 40, 64, 80, 128, 160), the adopted length does not lead to an artificial reduction in the coding efficiency. There is no empty padding of data blocks. If there is no hardware SerDes module available in FPGA, user can implement custom one using VHDL or SystemVerilog language, utilizing SERDES primitives [13]–[15] or native I/O interfaces [16]. The second factor is the trade-off between the efficiency of the forward error correction and demand for resources (encoders and decoders complexity). The applied orthogonal concatenation uses the same correction code in the horizontal and vertical direction so the length of a single data block needs to be a square of a natural number. The efficiency of orthogonal concatenation decreases as the block length decreases. On the other hand, long data blocks lead to a rapid increase in resource utilization, especially for error correction decoders. The drawback of the 256 bits long frame is a nondivisibility of the user message content length by

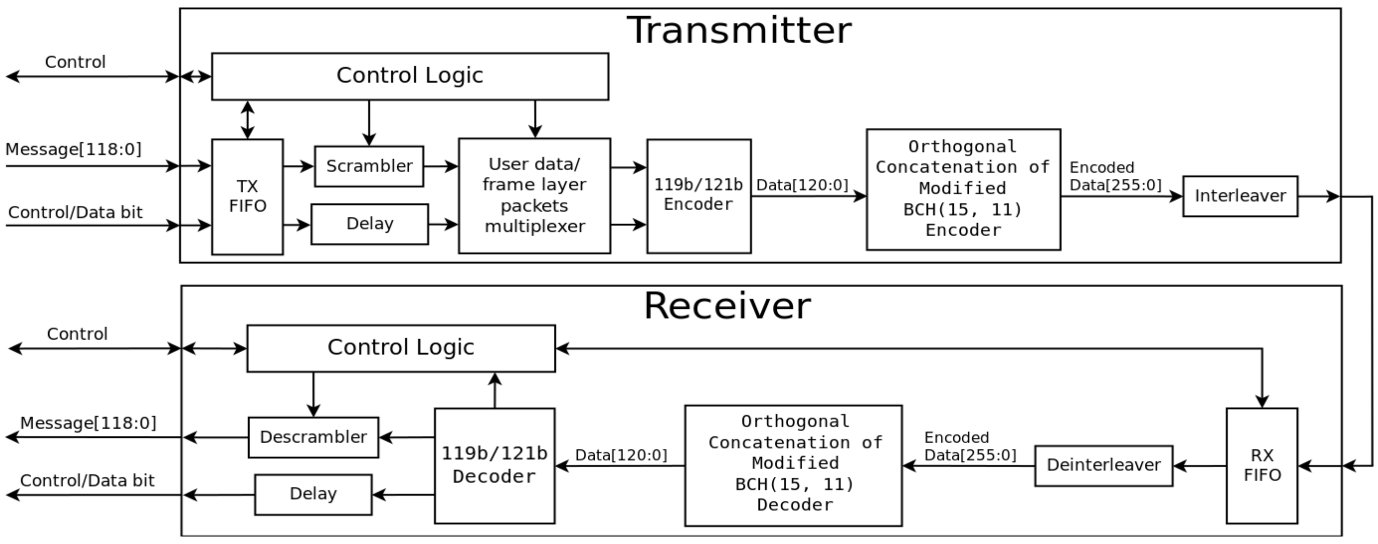

Fig. 1. General block diagram of proposed design

8. However, it is not a significant shortcoming because bits remaining after splitting the message into 8 bit bytes can be easily used in data flow control mechanisms (for example for communication channels that enable directing data to specific application modules).

A. Scrambler

The role of the scrambler as a component of transmitter and receiver is to increase the accuracy of clock recovery, maintain the DC balance and provide randomness to the EMI generated by output drivers.

Synchronous scrambler is used in the design, which in contrast to the self-synchronous scrambler does not multiplicate errors (because of no feedback loop). Scrambler can be implemented as parallel or serial. Parallel architecture has been used in the implementation in order to reduce the datapath latency. Theoretically, once the synchronous scrambler has synchronized, it should never become unsynchronized, even if there are transmission errors. Two possible scenarios when synchronous scrambler is not synchronized are the invalid initialization of the internal state or ions/electromagnetic radiation causing a change of the state in scrambler flip-flops. [1]

The disadvantage of the synchronous scrambler is the need for explicit initialization. The opposite node has to send its internal scrambler state at the beginning of the transmission, right after locking to word boundary and before user data transmission. It is not a big excess though.

$x^{16}+x^{12}+x^{3}+x^{1}+1$ has been chosen as a primary polynomial. Results of synthesis with different polynomials have shown, that longer polynomials lead to higher resource utilization as shown in Table I.

There is no unequivocal answer to the question if the chosen polynomial length is sufficient for 119 bits long user data block. The shorter polynomial has a shorter repetition period, however, much depends on the data sent by the user. If the application data also shows periodicity, and that period is close to the period of applied polynomial, then the efficiency of the scrambling process might significantly decrease (this is not always the truth because it also depends on the phase difference between the scrambler and the data). For example, in Interlaken protocol, the $x^{58}+x^{39}+1$ prime polynomial is applied to 64 bits long data block, whereas the $x^{16}+x^{15}+x^{13}+x^{4}+1$ polynomial is used in SATA technology for data blocks that might be up to 8 kB long [20]. It also facilitates the implementation when the scrambler state is short enough to be sent within a single framing layer packet.

[1]Of course such unexpected change of the state would lead to massive error propagation. However, protecting chips and boards against the radiation is a different topic, and it is outside the scope of this paper (one of the possible solutions is a redundancy of modules [17]–[19]).

TABLE I
RESOURCE UTILIZATION FOR PARALLEL SYNCHRONOUS SCRAMBLER FOR 64 BITS LONG WORD AS A FUNCTION OF PRIME POLYNOMIAL (RESULTS FOR XILINX KINTEX ULTRASCALE+ FAMILY).

Polynomial	Slice LUTs	Number of flip-flops
x^9+x^4+1	45	74
$x^{16}+x^{12}+x^3+x^1+1$	94	81
$x^{31}+x^3+1$	107	96
$x^{58}+x^{39}+1$	125	123

B. 119b/121b Encoding

119b/121b encoding is required to delineate word boundaries and distinguish framing layer control packets from user data packets. The applied encoding scheme is just a simple modification of a 64b/66b encoding used in the IEEE 802.3ae standard for 10 Gigabit Ethernet over fiber [21]. Each 119 bits long data block, after randomization in the scrambler module, gets extended by 2 bits, the so-called synchronization bits. Value ”01” indicates user data packet, while value ”10” is used for framing layer control packets. Both values ”00” and ”11” are invalid what allows achieving correct link synchronization and failures detection, especially the loss of synchronization. The process of locking to word boundary is greatly inspired by the 64b/67b word boundary lock in the Interlaken protocol. The synchronization is achieved after correct detection of 64 consecutive suffix patterns. The flow diagram for achieving and maintaining 119b/121b encoding synchronization is shown in Fig.2. Rx slide mechanism, example shown in Figure 3 for packets of length 2, must be supported by the SerDes module that the design interact with. Rx slide allows shifting parallel data by one bit. Maximum synchronization time for 119b/121b boundary lock (assuming no errors after FEC decodings) depends on the hardware platform and used SerDes block, as such blocks have different timings for rx slide mechanism. The overhead associated with the 119b/121b encoding equals 1.65%.

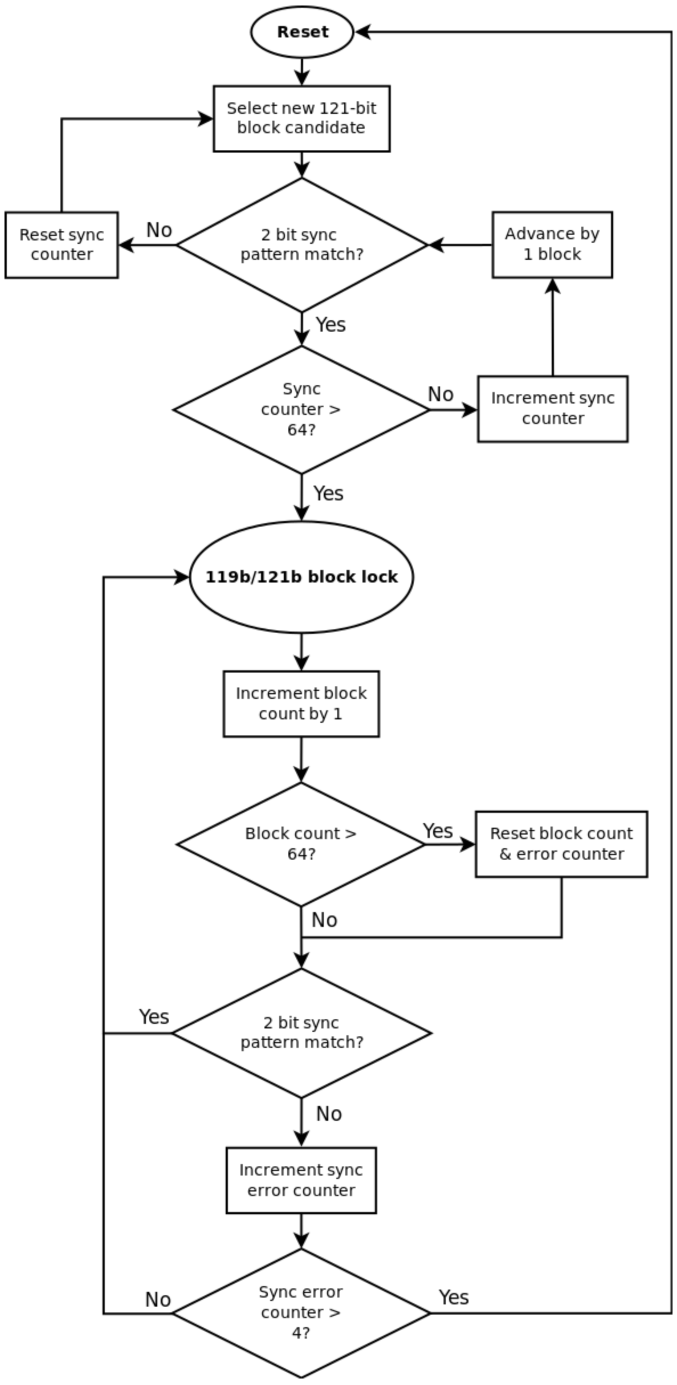
Reset
Select new 121-bit block candidate
Reset sync counter
No
2 bit sync pattern match?
Advance by 1 block
Yes
Sync counter > 64?
No
Increment sync counter
Yes
119b/121b block lock
Increment block count by 1
Block count > 64?
Yes
Reset block count & error counter
No
Yes
2 bit sync pattern match?
No
Increment sync error counter
No
Sync error counter > 4?
Yes

Fig. 2. 119b/121b word boundary lock

An important part of the 119b/121b encoding is the ability to define and distinguish framing layer control packets, which

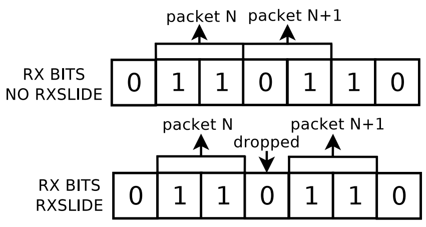

Fig. 3. Rx slide mechanism

are used for monitoring and managing the transmission. The format of the applied control packet is presented in Table II.

Control words of framing layer are recognized on the basis of 'Framing layer control packet' flag value. Having such a separate flag allows utilizing 119b/121b encoding control packets also by upper layers protocols.

There are 2 types of framing layer control packets: idle and control. One of the functions of the control packet is to allow descrambler synchronization. The current state of the scrambler is of course sent without any extra randomization. Framing layer control packet of type control is sent periodically every 8192 blocks. However, there are 3 following exceptions to this rule:

- The control packet is always sent, with 'Framing layer control packet request' flag set, after locking to the 119b/121b word boundaries in order to achieve descrambler synchronization as soon as possible.
- The control packet is sent after each change of the flow control flag.
- The control packet is sent when the node receives a control packet with 'Framing layer control packet request' flag set.

An idle frame is sent when there is no user data in TX FIFO. The main purpose of the idle frame is to keep the link constantly synchronized. The only difference between idle and control frame is that in the case of the control frame the scrambler block is suspended for a single clock tick, which makes it possible to synchronize the descrambler in the opposite node. Suspending the scrambler module for idle frames would lead to no randomization of the packets. If there were no user data packets in the TX FIFO for a longer period of time, this could increase the probability of the DC wander.

C. Forward Error Correction

Forward error correction algorithm utilizes orthogonal concatenation, which is shown in Fig.4. First, input data is encoded in the horizontal direction (green squares symbolizes redundancy bits of this coding). Then input data is encoded in the vertical direction (red squares). Vertical encoding also applies to redundant bits from horizontal coding (yellow squares).

The base for the forward error correction, both in rows and columns, is modified BCH(15, 11) with $x^4 + x + 1$ as a prime polynomial, which got extended by 1 parity bit.

256 bits long data frames imply a 16 bits codeword. That is not a standard length for commonly used error correction codes so the decision has been taken to extend ideal code such as BCH(15, 11).

TABLE II
FRAMING LAYER CONTROL PACKET FORMAT

Bits	Start value	Function	Description
120	'1'	Framing layer control packet	Flag used to distinguish between framing layer control packets and user layer control packets. Value '1' indicates control packet of framing layer. Value '0' can be used by user for upper layers control packets.
119	'1'	Idle/Control frame	Flag indicating if the packet of framing layer is idle or control packet. Value '1' indicates idle packets.
118	'0'	Framing layer control packet request	Request to the opposite node for framing layer control packet.
117	'0'	Flow control	Flow control information. Value '1' indicates that the node is capable of data reception.
116:101	Variable	CRC16	Checksum of the received packet. It includes all the fields except the 2-bits suffix from 119b/121b encoding.
100:19	Variable	Unused/ For future use	Not used in current implementation. For framing layer packets its value is generated by the scrambler.
18:2	Variable	Current scrambler state	Current scrambler state of the transmitting node. In case of idle frame the value is pseudo-random.
1:0	'10'	Control/User data frame (119b/121b)	The suffix from the 119b/121b encoding. Value '01' indicates user data frame. Value '10' is used for control packet of framing layer or control packet of upper user layers (depending on the value of 'Framing layer control packet' field).

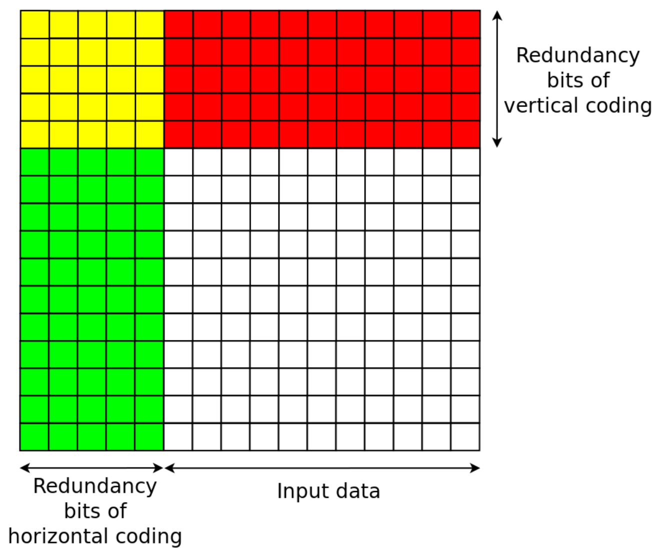
Redundancy bits of vertical coding
Redundancy bits of horizontal coding
Input data

Fig. 4. Orthogonal concatenation of forward error correction codes

The domination of extrinsic noises in the work environment usually results in the occurrence of groups errors. Therefore, it seems reasonable to improve the code so that it is capable of correcting double-adjacent errors. By analyzing recent publications in the field of error correcting codes, one can see a gain in popularity of codes capable of correcting double-adjacent errors or even triple-adjacent errors [22]–[26]. In 1959 N. M. Abramson in [27] has proposed a simple and effective method for constructing error correction codes capable of correcting all single errors and all double-adjacent errors. However, the so-called Abramson codes are characterized by the fact that the number of available syndrome values is greater than the number of possible single errors and double-adjacent errors, so they should not be considered as perfect codes. This fact results from the assumption that an error on the first and last position of the codeword is considered by Abramson as a double-adjacent error.

The code with parameters (16, 11) provides $2^5-1 = 31$ non-zero values of syndromes. Theoretically, such code should be capable of correcting all single errors (there are 16 single-bit errors in 16 bits long codeword) and all double-adjacent errors (there are 15 double-adjacent errors in 16 bits long codeword). If a code with required properties exists, then its parity test matrix (H) must meet the following conditions [23]:

- All columns in H are different and non-zero.
- The sums of two adjacent columns in H are all different and also different from all columns and non-zero.

An analytic proof that there exists 5x16 matrix that meets above conditions is not so trivial. Applying a numerical method to search all matrices with such dimensions would be too time-consuming. Therefore, a Python script has been implemented to check if there are any 3x4 or 4x8 matrices meeting the necessary conditions, which could be parity test matrices for error correction codes with a similar structure but a shorter codeword. Checking the whole sets of 3x4 and 4x8 matrices showed that there are none that would meet the required conditions. Based on this it can be assumed, with very high probability, that there is also no 5x16 matrix that would meet these conditions.

In the tested set, there were matrices, that indicate the existence of error correction codes with given parameters capable of correcting all single errors and all double-adjacent errors except one. Analogical matrix has been found for 5x16 dimensions. The uncorrectable configuration of double-adjacent error spans on one parity bit and one message bit. It means that after decoding there is only a single bit error (bit number 10). This scenario is shown in Fig.5, the green square represents an extra parity bit added to standard BCH(15, 11) code.

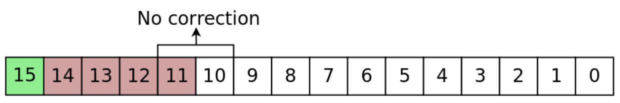

Fig. 5. Codeword for modified BCH(15, 11) code

The generator matrix for applied error correction code has the following form:

$$G_{11,16} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

BCH codes have one more advantage in terms of hardware design. They are very simple for implementation in the hardware description languages. Both encoding and decoding of single message requires zero clock cycles. Listing 1 shows implementation for the modified BCH(15, 11) code.

Listing 1. Modified BCH(15, 11) encoder implementation in VHDL.

```
library ieee;
use ieee.std_logic_1164.all;

entity bch_16_11_enc is
  port(
    msg: in std_ulogic_vector(10 downto 0);
    code_word: out std_ulogic_vector(15 downto 0)
  );
end;

architecture rtl of bch_16_11_enc is

begin
  code_word(10 downto 0) <= msg(10 downto 0);

  code_word(11) <= msg(0) xor msg(3) xor msg(4)
                   xor msg(6) xor msg(8) xor msg(9)
                   xor msg(10);
  code_word(12) <= msg(0) xor msg(1) xor msg(3)
                   xor msg(5) xor msg(6) xor msg(7)
                   xor msg(8);
  code_word(13) <= msg(1) xor msg(2) xor msg(4)
                   xor msg(6) xor msg(7) xor msg(8)
                   xor msg(9);
  code_word(14) <= msg(2) xor msg(3) xor msg(5)
                   xor msg(7) xor msg(8) xor msg(9)
                   xor msg(10);
  code_word(15) <= msg(0) xor msg(1) xor msg(2)
                   xor msg(4) xor msg(5) xor msg(8)
                   xor msg(10);
end rtl;
```

D. Interleaver

Interleaving is an operation of data reordering before transmission, performed in order to reduce the effect of burst errors. The most common solutions are the block interleaving and the helical interleaving [28]. In the proposed design, helical interleaving is used as it works very well with orthogonal concatenation. Due to specific properties of applied error correction code (no correction for double adjacent error spanned on one parity bit and one message bit), two implementations of helical interleaving have been checked. The first one with the vertical axis of the helix (data column direction) and the second one with the horizontal axis (data row direction). Both variants are shown in Fig.6. The numbers inside the squares indicate the order of bit sending. The performance against burst errors handling is presented in Table III.

Helical interleaving greatly improves the efficiency of orthogonal concatenation with modified BCH(15, 11) up to a certain length of a burst error. After exceeding this length,

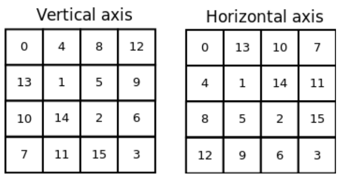

Fig. 6. Orthogonal implementations of helical interleaving

TABLE III
NUMBER OF UNCORRECTABLE POSITIONS OF GROUP ERRORS FOR DIFFERENT IMPLEMENTATIONS OF INTERLEAVING.

Burst error length	Number of possible positions	Number of uncorrectable positions		
		No interleaving	Interleaving vertical axis	Interleaving horizontal axis
16	241	0	0	0
18	239	1	0	0
24	233	19	0	0
30	227	25	0	0
31	226	26	15	0
32	225	27	45	1
33	224	28	155	43
34	223	50	168	92
38	219	219	219	202
39	218	218	218	218

the efficiency decreases. The crossing point is close to the length of two single codewords, 31 for helical interleaving with the vertical axis and 32 for the one with the horizontal axis. This result was expected since the base error correcting code is capable of correcting all single errors and almost all double-adjacent errors except one. In case of no interleaving, correction problems begin already for burst error with the length of 18 bits. That is because of the error concentration in the area where modified BCH(15, 11) is not able to correct double-adjacent errors (borders where squares change color).

Different results obtained for vertical and horizontal axes might seem surprising at first. The inconsistency starts from the burst error of 31 bits length. This can be easily explained when one looks in Fig.6. Assume that 31 bits long error spans on the squares with numbers from 2 to 32. In case of the vertical axis, such scenario leads to a triple-adjacent error in the single codeword in one of the columns (squares 2, 17, 32). For implementation with the horizontal axis, the situation is similar, but a triple error occurs in one of the rows. As the decoding of orthogonal concatenation starts with columns decoding, the implementation with the horizontal axis handles this scenario because it does not see any triple error. This error occurs in the rows direction but is corrected in the first stage of columns decoding. Of course, the final implementation of the design uses helical interleaving with the horizontal axis.

III. RESULTS AND PERFORMANCE ANALYSIS

A widely accepted quality indicator of transmission systems is bit error rate (BER), which represents the probability of receiving wrong bit value. Common requirements for serial links are generally in the range of 10^{-15} to 10^{-6} [29]. For example, for 10 Gigabit Ethernet, it was recommended that $BER \leq 10^{-13}$ [30].

The error correcting capabilities have been estimated for both random errors and burst errors. Matlab software has been used for simulations. Note that both encoding and decoding

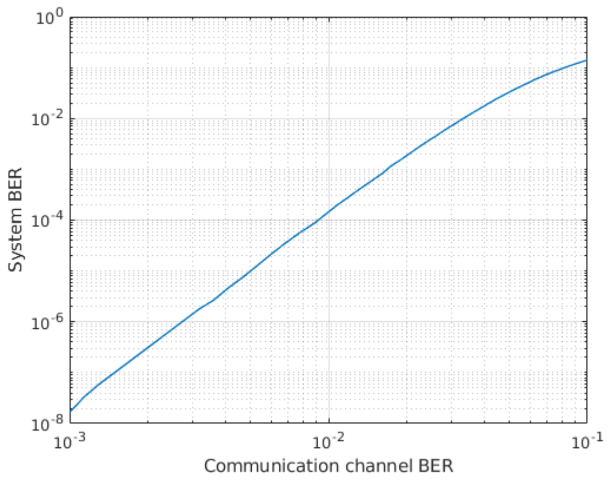

Fig. 7. System BER characteristic as a function of communication channel BER.

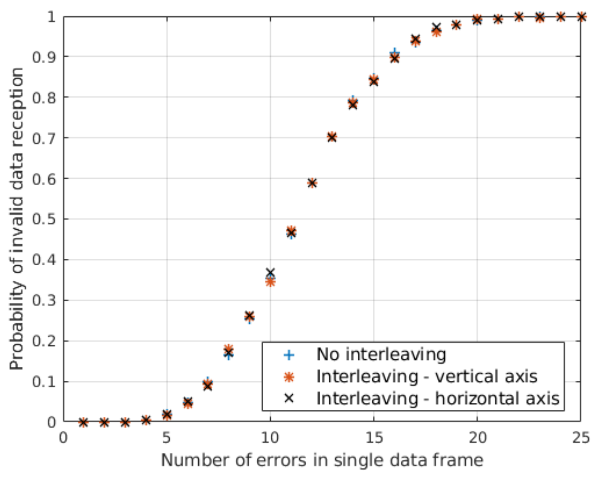

Fig. 8. Impact of helical interleaving on random errors correction capabilities.

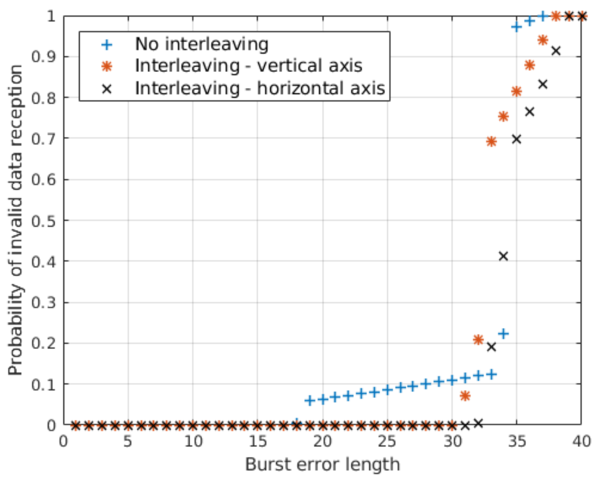

Fig. 9. Probability of invalid data reception as a function of burst error length.

algorithms are fully deterministic, so the probabilities obtained from the simulations are not estimations but the exact values.

A. Random errors

Usually, the system BER characteristic as a function of SNR (Signal Noise Ratio) is used to present system performance against random errors. However, this characteristic depends also on applied signal modulation in a physical transmission channel. If modulation is not defined, it is more useful and convenient to present the system BER as a function of the communication channel BER. Then, having both characteristics, it is easy to determine the system BER characteristic as a function of SNR.

Fig.7 shows the performance of the proposed design against random errors. It has been created with *BinarySymmetricChannel* model from *Communications System Toolbox Version 6.3 (R2016b) 25-Aug-2016*. This model assumes that the error probability for both ’0’ and ’1’ value bits is the same. Simulations have been carried out only for the range of two decades of communication channel BER due to long simulation times for values less than 10^{-3}. In real links, the communication channel BER is much smaller. For example, in the telecom optical link applications the maximum physical BER requirement is around 10^{-9}, whereas in the datacom applications, it is in the range of 10^{-15} to 10^{-12} [31].

Fig.8 shows how helical interleaving impacts on random error correction capabilities. The number of errors in a single data frame refers to the number of errors in 256 bits long frame transmitted in a physical channel. As one can see, helical interleaving does not play any role in case of random errors.

B. Burst errors

Some of the results for burst error correction capabilities can be found in Table III. Fig.9 presents the probability of invalid data reception as a function of burst error length. To obtain the points for the figure, simulations have been carried out in Matlab. Burst errors of diferent length have been injected into the encoded packet on all possible positions. For example, there are 256 possible positions for burst error of length 1, and 255 possible positions for burst error of length 2. For each burst error length, the probability is the ratio of the number of incorrectly decoded packets to the number of all possible positions for this particular burst error length. It also shows the influence of the helical interleaving. A significant improvement of burst error correction capabilities is achieved with the helical interleaving, especially with the horizontal axis.

Fig.10 sums up the performance of the proposed error correction scheme. It presents the probability of invalid user data reception (119 bits long block) as a function of the number of errors in the 256 bits long frame transmitted in a physical channel for both random and burst errors.

C. Latency analysis

Table IV sums up latency achieved during tests with 28.2 Gbps data rate. Xilinx GTY IP block was configured with 128 bits user data width and 64 bits internal data width. GTY TX and GTY RX are not shown in Fig.1. If they were, they should

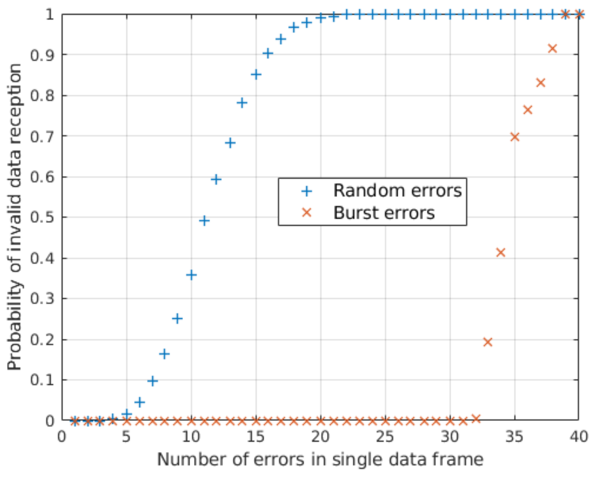

Fig. 10. Summary of error correction capabilities.

TABLE IV
SUMMARY OF OVERALL TRANSMISSION LATENCY.

Module	Number of clock cycles	Clock period [ns]	Module latency [ns]
TX FIFO	2	9.0780	18.1560
Scrambler	1	9.0780	9.0780
Packet multiplexer	1	9.0780	9.0780
119b/121b encoder	1	9.0780	9.0780
BCH encoder	2	9.0780	18.1560
Interleaver	0	0	0
GTY TX wrapper	1	9.0780	9.0780
GTY TX	407.2	0.0355	14.4556
Total TX	-	-	87.0796
GTY RX	721.7	0.0355	25.6204
GTY RX wrapper	1	9.0780	9.0780
RX FIFO	2	9.0780	18.1560
Deinterleaver	0	0	0
BCH decoder	2	9.0780	18.1560
119b/121b decoder	1	9.0780	9.0780
Descrambler	1	9.0780	9.0780
Total RX	-	-	89.1664
Total	-	-	176.2460

be respectively placed after the interleaver and before RX FIFO. 28.2 Gbps data rate and 256 bits packet length results with data path being clocked with 110.15625 MHz (approx 9.078 ns clock period). As GTY was configured to work with the widest possible internal data width (64), this frequency was synthesized by dividing TXUSRCLK and RXUSRCLK by 4 in BUFG_GT buffers [32]. GTY wrappers are necessary for splitting 256 bits long blocks into 2 128 bits blocks as Xilinx GTY transceiver does not support 256 bits user data width. Latency of the GTY TX and RX has been estimated based on [33]. The overall latency (from the time user data is sampled on the rising edge at the TX FIFO input, to the time user data is available at the descrambler output) is around 176.2460 ns. Final latency of course depends on the adopted transmission speed and length of a physical connection.

Further optimization could be applied to the solution to decrease overall latency. For example, if flow control is not necessary both TX FIFO and RX FIFO could be removed (minus 4 data path clock cycles). One can even try to clock data path several times faster than the minimum for the applied data rate, as long as timings are still met.

IV. CONCLUSION

In this paper, the protocol for high-speed serial transmission between FPGA devices has been proposed and evaluated. It can be used in a harsh environment as it is capable of correcting long burst errors. The basis for communication are the encoded 256 bits long data blocks, that can contain up to 119 bits of user data. The total efficiency of the solution equals 46.48%.

Thanks to the orthogonal concatenation of the modified BCH(15, 11) code and the helical interleaving, the design is capable of correcting all group errors of length up to 31 bits.

The solution has also been implemented and verified on the Xilinx KCU116 evaluation board with a data rate of 28.2 Gbps, which is the upper limit for GT blocks provided on this board. Tables V and VI summarize resource utilization for a single instance synthesized with strategy *Vivado Synthesis Defaults (Vivado Synthesis 2017)*. Utilization of resources not mentioned in the table equals 0. Both tables refer to the Xilinx Kintex UltraScale+ family - xcku5p-ffvb676-2-e part. Total On-Chip Power, with a single GTY transceiver working, equals 1.294 W, wherein 0.739 W (57%) is dynamic power and 0.555 W (43%) is static power.

TABLE V
RESOURCE UTILIZATION FOR A SINGLE INSTANCE OF PROPOSED DESIGN.

CLB LUTs	CLB Registers	CLB	Block RAM Tile
1278(0.59%)	1338(0.31%)	252(0.01%)	6(1.25%)

TABLE VI
RESOURCE UTILIZATION FOR A SINGLE INSTANCE OF PROPOSED DESIGN PLUS A SINGLE GTY TRANSCEIVER.

CLB LUTs	CLB Registers	CLB	Block RAM Tile
1485(0.68%)	2016(0.46%)	315(0.01%)	6(1.25%)

The proposed protocol can be easily modified and extended, for instance by utilizing the 82 unused bits in the framing layer control packet. Among the particularly interesting directions of development one can distinguish:

- Enabling the performance scaling with increasing the number of transmission lanes.
- Addition of clock compensation mechanism for a repeater function, so that the protocol may be electrically relayed across an intermediary device between two clock domains.
- Extending the protocol with the functionality of communication channels (the answer to the question which layer is suitable for such a mechanism is not obvious and depends on the application).

REFERENCES

[1] J. Postel, “Transmission Control Protocol,” Tech. Rep., 1981. [Online]. Available: https://www.rfc-editor.org/info/rfc0793

[2] G. Fairhurst and L. Wood, “Advice to link designers on link Automatic Repeat reQuest (ARQ),” RFC Editor, Tech. Rep. RFC3366, Aug. 2002. [Online]. Available: https://www.rfc-editor.org/info/rfc3366

[3] C. Systems and C. Systems, “Interlaken Protocol Definition,” Tech. Rep., 2008.

[4] G. specifications group, “GBTX Manual,” Tech. Rep., Oct. 2016.

[5] S. Mandal, J. Saini, W. M. Zabołotny, S. Sau, A. Chakrabarti, and S. Chattopadhyay, “An FPGA-Based High-Speed Error Resilient Data Aggregation and Control for High Energy Physics Experiment,” *IEEE Transactions on Nuclear Science*, vol. 64, no. 3, pp. 933–944, Mar. 2017.

[6] R. Giordano, V. Izzo, S. Perrella, and A. Aloisio, “A JESD204b-Compliant Architecture for Remote and Deterministic-Latency Operation,” *IEEE Transactions on Nuclear Science*, vol. 64, no. 6, pp. 1225–1231, Jun. 2017.

[7] D. Gaisbauer, Y. Bai, S. Huber, I. Konorov, D. Levit, S. Paul, and D. Steffen, “Unified Communication Framework,” *IEEE Transactions on Nuclear Science*, vol. 64, no. 10, pp. 2761–2764, Oct. 2017.

[8] E. Kadric, N. Manjikian, and Z. Zilic, “An FPGA implementation for a high-speed optical link with a PCIe interface,” in *2012 IEEE International SOC Conference*, Sep. 2012, pp. 83–87.

[9] B. Raahemi, “Error correction on 64/66 bit encoded links,” in *Canadian Conference on Electrical and Computer Engineering, 2005.*, May 2005, pp. 412–416.

[10] A. Wu, X. Jin, X. Du, and S. Guo, “A flexible FPGA-to-FPGA communication system,” in *2017 19th International Conference on Advanced Communication Technology (ICACT)*, Feb. 2017, pp. 836–843.

[11] J. Zhang, Q. Lin, Y. Zhang, and Z. Chen, “Design and Implementation of High-Speed Data Transmission Scheme Between FPGA Boards Based on Virtex -7 Series,” in *2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC)*, May 2018, pp. 444–448.

[12] R. Sanchez Correa and J. P. David, “Ultra-low latency communication channels for FPGA-based HPC cluster,” *Integration*, vol. 63, pp. 41–55, Sep. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167926017303966

[13] Xilinx, “Source-Synchronous Serialization and Deserialization (up to 1050 Mb/s),” Tech. Rep., 2013.

[14] ——, “LVDS Source Synchronous 7:1 Serialization and Deserialization Using Clock Multiplication,” Tech. Rep., 2018.

[15] I. Corporation, “Intel® Stratix® 10 High-Speed LVDS I/O User Guide,” Tech. Rep., 2019.

[16] Xilinx, “Native High-Speed I/O Interfaces,” Tech. Rep., 2017. [Online]. Available: https://www.xilinx.com/support/documentation/application_notes/xapp1274-native-high-speed-io-interfaces.pdf

[17] P. Samudrala, J. Ramos, and S. Katkoori, “Selective triple Modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs,” *IEEE Transactions on Nuclear Science*, vol. 51, no. 5, pp. 2957–2969, Oct. 2004.

[18] K. S. Morgan, D. L. McMurtrey, B. H. Pratt, and M. J. Wirthlin, “A Comparison of TMR With Alternative Fault-Tolerant Design Techniques for FPGAs,” *IEEE Transactions on Nuclear Science*, vol. 54, no. 6, pp. 2065–2072, Dec. 2007.

[19] W. M. Zabolotny, I. M. Kudla, K. T. Pozniak, K. Bunkowski, K. Kierzkowski, G. Wrochna, and J. Krolikowski, “Radiation tolerant design of RLBCS system for RPC detector in LHC experiment,” in *Photonics Applications in Industry and Research IV*, vol. 5948. International Society for Optics and Photonics, Oct. 2005, p. 59481E. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/5948/59481E/Radiation-tolerant-design-of-RLBCS-system-for-RPC-detector-in/10.1117/12.622864.short

[20] I. APT Technologies, D. C. Corporation, I. Corporation, I. Corporation, M. Corporation, and S. Technology, “Serial ATA: High Speed Serialized AT Attachment,” Aug. 2001.

[21] S. Minami, J. Hoffmann, N. Kurz, and W. Ott, “Design and implementation of a data transfer protocol via optical fiber - IEEE Conference Publication.” [Online]. Available: https://ieeexplore.ieee.org/document/5750447/

[22] S. Cha and H. Yoon, “Single-Error-Correction and Double-Adjacent-Error-Correction Code for Simultaneous Testing of Data Bit and Check Bit Arrays in Memories,” *IEEE Transactions on Device and Materials Reliability*, vol. 14, no. 1, pp. 529–535, Mar. 2014.

[23] P. Reviriego, J. Martínez, S. Pontarelli, and J. A. Maestro, “A Method to Design SEC-DED-DAEC Codes With Optimized Decoding,” *IEEE Transactions on Device and Materials Reliability*, vol. 14, no. 3, pp. 884–889, Sep. 2014.

[24] C. Badack, T. Kern, and M. Gössel, “Modified DEC BCH codes for parallel correction of 3-bit errors comprising a pair of adjacent errors,” in *2014 IEEE 20th International On-Line Testing Symposium (IOLTS)*, Jul. 2014, pp. 116–121.

[25] X. She, N. Li, and D. W. Jensen, “SEU Tolerant Memory Using Error Correction Code,” *IEEE Transactions on Nuclear Science*, vol. 59, no. 1, pp. 205–210, Feb. 2012.

[26] P. Reviriego, S. Pontarelli, A. Evans, and J. A. Maestro, “A Class of SEC-DED-DAEC Codes Derived From Orthogonal Latin Square Codes,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 5, pp. 968–972, May 2015.

[27] N. Abramson, “A class of systematic codes for non-independent errors,” *IRE Transactions on Information Theory*, vol. 5, no. 4, pp. 150–157, Dec. 1959.

[28] K.-M. Cheung and L. Swanson, “A performance comparison between block interleaved and helically interleaved concatenated coding systems,” Aug. 1989. [Online]. Available: http://adsabs.harvard.edu/abs/1989tdar.nasa...95C

[29] S. Cypress, “AN1047 - Understanding Bit-Error-Rate with HOTLink®,” Aug. 2017. [Online]. Available: http://www.cypress.com/documentation/application-notes/an1047-understanding-bit-error-rate-hotlinkr

[30] E. S. Chang and R. Taborek, “Recommendation of 10 ˆ-13 Bit Error Rate for 10 Gigabit Ethernet,” Tech. Rep., Jul. 1999.

[31] R. Dahlgren and B. Dahlgren, “NOISE IN FIBER OPTIC COMMUNICATION LINKS,” Tech. Rep.

[32] Xilinx, “UltraScale Architecture GTY Transceivers User Guide (UG578),” Tech. Rep., 2017.

[33] ——, “AR# 69011: UltraScale+ GTY Transceiver: TX and RX Latency Values.” [Online]. Available: https://www.xilinx.com/support/answers/69011.html