FPGA-Based Data Storage System on NAND Flash Memory in RAID 6 Architecture for In-Line Pipeline Inspection Gauges

N. A. Rodríguez-Olivares, *Member, IEEE*, A. Gómez-Hernández, L. Nava-Balanzar, H. Jiménez-Hernández, and J. A. Soto-Cajiga

Abstract—In this manuscript, we present a redundant data storage system based on NAND flash memory chips for in-line Pipeline Inspection Gauges (PIGs). The system is the next step for a technique that reduces data from 1,024 to 37 bytes by 80 transducers used for straight-beam ultrasonic inspection. Each inspection is costly, because PIGs check pipelines up to 100 Km, collecting data every 3 mm and reaching speeds of 2 m/s. These conditions require that the storage system must be redundant, and able to maintain a minimum data flow, thus avoiding bottlenecks. To achieve this, we analyzed the variables that influence the inspection process in real-time, and we structured our Flash Translation Layer (FTL) to eliminate the latencies generated by the computation of the Error Correcting Codes (ECC) and redundancy bytes. Our controller computes the ECC and redundancy bytes while it transfers the information to the cache register of the selected die in the memory chips. At the hardware level, we interleaved 8 NAND flash chips in a Redundant Array of Independent Disks (RAID) type-6 architecture. We tested the storage system considering the incorrect response of up to 2 chips and ensuring a throughput up to 7.28 MB/s. Finally, we expanded the analysis of the data flow, whereby this system is profitable for different pipeline diameters or compression techniques.

Index Terms—Data storage systems, error correction codes (EEC), field programmable gate arrays (FPGA), NAND flash memory, parallel architectures, pipeline inspection gauge (PIG), reed-solomon codes (RS)

1 Introduction

We present the design and experimental results of a data storage system based on a Field Programmable Gate Array (FPGA), and 8 NAND flash memory chips in a RAID-6 architecture useful for in-line Pipeline Inspection Gauges. For pipeline inspection, the one-channel ultrasound system is capable of operating with up to 80 transducers multiplexed. This amount is due to the Pulse Repetition Frequency (PRF) and the necessary data for each transducer [1]. Our storage system keeps up the maximum performance of one-channel ultrasound system, because it ensures a minimum data flow of 80 transducers, avoiding bottlenecks. The system computes the ECC and redundancy bytes while it writes data in the memory chips. Nevertheless, this system is adaptable for a vast Number of Transducers (NT). To achieve this, we analyzed the inspection process in function of the number of Bytes per Transducer (BT), the data burst generated at the maximum Frequency (F), and the Capacity of Storage of the memory chips (CS).

The pipeline inspection allows the prevention of failures which would be more expensive and problematic, should they happen to occur [2]. The inspection involves preparing, inspecting, and reactivating the pipeline operation, which causes high operating costs. Due to this, we consider that the storage system should be redundant to reduce the risk of a second inspection if the storage units are damaged.

The Non-Destructive Test (NDT) based on ultrasound systems checks the correct operation of pipelines for hydrocarbons and water. Ultrasonic techniques allow volumetric and automatic inspection [3]. Automatic in-line inspection of pipelines is done through intelligent PIGs [4], [5], [6]. This equipment has ultrasound systems focused on the detection and sizing of defects, for example, metal loss caused by corrosion. The requirements for an ultrasonic inspection system for PIGs are governed by international standards (see Table 1). For example, to inspect a pipeline up to 100 Km in length, collecting a data set every 3 mm in 203.2 mm (8 inches) diameter pipeline, the PIG must be able to store the information of 80 transducers, and the number of transducers increase with diameter [4], [7], [8].

In our case, each ultrasonic transducer generates a 1,024 data frame of 8 bits from the straight-beam ultrasound technique. The FPGA reduces these signals through 4 steps: Noise reduction, Signal rectification, Envelope detection, and Maxima detection [1]. In this way, the controller only stores the necessary information to determine the thickness of the pipeline. The thickness is estimated using the local maximum of each rectified echo (1 byte) and its temporary location (2 bytes). The reduction technique [1] is adjusted to generate 37 Bytes per Transducer. The first byte identifies the maximum number, and the next 12 groups of 3 bytes represent each one of the detected echoes.

For 203.2 mm diameter pipeline, the PIG inspects with 80 transducers at a speed of up to 2 m/s, at this speed the data flow can reach 1.88 MB/s. Storing data in the NAND flash memory chips could generate a bottleneck under these conditions, because the data storage system should maintain a minimum data flow.

At the hardware level, we connected the NAND Flash memory chips in a particular arrangement, where the memory chips are treated as an improved super chip with two internal targets [9]. Seong et al. [10] proposed the super chip concept, which alludes to interleaving several memory chips through their control signals pins. At the software level, the main advantage of our data storage system against the existing ones is that it does not generate any latency to protect the information. Our controller computes the ECC and redundancy bytes at the same time while it stores the information in the cache register of the selected die in the memory chips (a NAND Flash die is the minimum unit that can independently execute commands and report status [11]). Our key contributions can be summarized as follows:

1) A storage system for raw data that computes the ECC and redundancy bytes at the same time while it writes the information in the cache register of the selected die in the memory chips.
2) We present the analysis to implement this storage system in function of the Number of Transducers, Bytes per Transducer, data burst generated at the maximum Frequency (F), and the Capacity of Storage (CS) of the memory chips.
3) We validated the storage system when the PIG runs at the speed of up 2 m/s, reaching a data flow of up to 14.05 MB/s without failing in any memory, and a minimum data flow of 7.28 MB/s if two memory chips fail in different arrays.

Section 2 presents the preliminaries and the related work relevant to these contributions. Section 3 describes the real-time inspection analysis, the physical connection of the memory chips and the structure of the controller. Section 4 shows the tests applied and the results obtained. Section 5 presents the discussion of possible

• *The authors are with the Department of Energy, Centro de Ingeniería y Desarrollo Industrial (CIDESI), Av. Playa Pie de la Cuesta 702, Desarrollo San Pablo, Queretaro, Qro. 76125, Mexico. E-mail: {noeamir, alex.02.02.88}@gmail.com, {lnava, hugo.jimenez, jsoto}@cidesi.edu.mx.*

Manuscript received 23 June 2017; revised 8 Jan. 2018; accepted 14 Jan. 2018. Date of publication 13 Mar. 2018; date of current version 15 June 2018.
(Corresponding author: N. A. Rodríguez-Olivares.)
Recommended for acceptance by P. Gratz.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2018.2794986

TABLE 1
Typical Requirements for PIGs [4], [7], [8]

Characteristic	value
Inspection distance	100 Km
Inspection speed	2 m/s
Pulse Repetition Frequency (PRF)	600 Hz
Circumferential spatial resolution	8 mm
Axial Spatial Resolution (ASR)	3 mm

applications and performances for different Bytes per Transducer, pipeline diameters, and Number of Transducers. Finally, Section 6 outlines the conclusions of this work.

2 Preliminaries and Related Work

2.1 Typical Data Storage Systems for PIGs

For PIGs, FPGAs and DSPs are the preferred schemes for the processing and reduction of the information, and a computer for controlling Hard Disk Drives (HDDs) for the storage [12]. In other structures, 3 FPGAs are the preferred scheme [5]. The first one for the control, the second for the signal processing and data compression, and the third FPGA for the interface between the second FPGA and an HDD. For our PIG, we discarded an HDD due to mechanical overhead and the need for another hardware element. In this sense, our storage system consists of an FPGA and 8 NAND Flash memory chips, which results in a smaller architecture than that mentioned in [5], [12], allowing small-diameter pipeline inspections.

Solid State Drives (SSDs) have advantages such as a compact size, low power consumption, and physical durability [13]. However, SSDs have a rigid architecture (black-box system) that does not maximize the NAND flash memory performance for storing raw data [14]. File formatting slows down the writing processes; this is the reason why the embedded systems work better with raw data.

2.2 Maintaining a Minimum Data Flow

A NAND flash storage system consists of two principal layers: Memory Technology Device Layer (MTD) and Flash Translation Layer (FTL). The MTD layer generates the operations on NAND flash memory chips such as write, erase, and read. FTL allows users to work with the memory as a block, to control file location, memory degradation, and bad blocks[15].

Different FTL and MTD improvements allow maintaining a minimum data flow and avoiding bottlenecks: Park et al. [16] used a 2-channel, 4-way-interleaving interface scheme with a software architecture adopting a hybrid-mapping algorithm. Seong et al. [10] created the Hydra controller, which achieves high performance through bus-level and chip-level interleaving. To speed up writing request processing, Hydra has an FTL based on block mapping. *Kang* et al. [17] designed a multi-channel architecture called Dumbo. This controller exploits I/O parallelism through striping, interleaving, and pipelining. To achieve the high speed, it uses 4 channels with several memory chips in each channel. These approaches [10], [16], [17] have some drawbacks as complicated controller design if increasing the number of channels, significant consumption of RAM, and several super-chips that consume many FPGA pins. In addition, Gupta et al. [18] proved that a page-level FTL without adjustable parameters can achieve a more straightforward solution with good performance. Because of this, our FTL works in page-level address mapping, where each page is interpreted as an established number of compressed ultrasound signals.

Wu et al. [9] developed a dynamic scheduling strategy to further explore parallelism at the target level in the memory chip. The name of the system is Dysource. It exploits the synchronous interface for NAND flash chips with multiple targets. However, to achieve high write processes, Dysource requires several controllers implemented on the FPGA. Also, Dysource interprets each target as a storage unit; this action would complicate the implementation of the RAID scheme.

Congming et al. [19] enhanced the performance of the controllers by exploiting parallelism through the creation of batches. The host creates an I/O scheduler, which coordinates requests for data input and output. Eui-Young et al. [20] developed a controller that uses a double-data-rate synchronous NAND flash interface to improve reading and writing performance. These controllers require many memory chips and slices in the FPGA to achieve this performance. Nevertheless, in PIGs-subsystems the priority is to optimize the writing process because the reading of the information is done off-line and can be much slower. Unlike the existing improvements in controllers that have been through interleaving, chip upgrading, target level operation, or batch creation. Our controller improves the writing speed performance and eliminates the latencies generated by the processes of information protection, such as ECC and redundancy computation.

2.3 Redundant Controllers

RAID [21] is the standardized scheme for the design of redundant multi-unit systems. The RAID systems have two primary goals, one is to increase performance by increasing the bandwidth, and the other is to enhance fault tolerance through redundancy. Luo et al. [22] implemented a RAID controller within a single SSD board, using Security Digital cards (SD) or Multi Media Cards (MMC) embedded in the same board, showing that the redundant systems become less expensive and having excellent performance in a smaller configuration. The problems with this controller are that it needs a microprocessor and uses SD or MMC devices, generating considerable latencies in the writing process. Alistair and Irfan [23] developed an FTL customized for RAID-based SSD storage. This system offers reliability enhancement under both sequential and random write patterns and works with RAID-4 and RAID-5 systems. The disadvantages of this system [23] are that it does not compute the ECC bits in the same layer, and the redundancy must be computerized sequentially before starting the data writing, which generates a latency in each writing process.

2.4 Criteria for Improving Information Protection

RAID-6 provides a high data reliability, which results in an improved Mean Time to Data Loss (MTDL) over other redundancy schemes. The RAID-6 scheme has two different parity calculations, and there are different techniques to compute them [24]. The criteria for selecting the parity calculations were the limited RAM in the FPGA and the difficulty to implement them. We selected the Reed-Solomon (RS) technique [25], [26] for RAID-6 redundancy. This technique is based on 8-bit Galois finite fields [27]. The advantage of using operations in the Galois field is that fractions and negative numbers do not exist. All mathematical operations result in another value within this same field, which in this case is 8-bit data. Some authors developed RAID-6 techniques only with or-exclusive operations, like EVENODD [28], RDP [29], P-Code [30], Minimum Density [31] and the Liberation Codes [24]. The problem with these techniques is that they require data codified diagonally from past or future moments, increasing the amount of use in RAM and execution times due to the interleaving of the memory chips in our arrangement.

We use Hamming codes [27] as ECC scheme in column mode because ECC codes and redundancy are on the same board, and the Hamming codes are handy if they are in smaller constituent codes along rows and columns. The Hamming codes reach a higher ECC because of the cross-parity checking and present a significant advantage for the lower hardware overload [32].

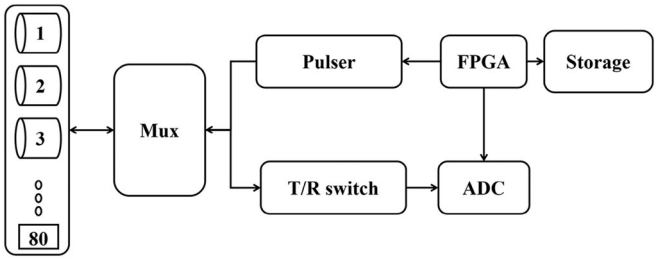

Fig. 1. One-channel ultrasound system that can control up to 80 transducers for this application.

3 Data Storage System Architecture

3.1 Real-Time Inspection Analysis

In pipeline inspection, international standards [4], [7], [8] determine the maximum PRF of each transducer at 600 Hz. PRF is the maximum Frequency (F) at which each transducer generates data burst, and it changes with the type of material and the inspection technique. The inverse of F is the pulse repetition time (t_M), which is the time available to process one ultrasound signal before re-exciting the same transducer, and has the value of 1666 μs.

For carbon-steel pipelines inspection is necessary to acquire data sets of 1,024 elements using an 8-bit Analog to Digital Converter (ADC) with a sampling rate of 50 MHz (20 ns) [1], [4], [5]. The acquisition time (t_a) of the data then takes 20.48 μs. Soto et al. [1] analyzed the ultrasound signal to develop a reduction technique which generates 37 Bytes per Transducer in the time t_a.

If t_M is divided by t_a, then 81 transducers could be excited before re-exciting the same transducer. In this way, the PIG obtains the maximum performing from one-channel ultrasound system, composed of a pulser-receiver circuit, a mux, a T/R switch, and an ADC (see Fig. 1). However, the design should consider the switching times of the mux and the generation of the high voltage pulse; this is the reason why we set this storage system to process 80 transducers (NT). If more transducers are required, then the ultrasound system is duplicated (transducers, ADC, mux, pulser-receiver, and T/R switch). This has the advantage of increasing the diameter of the pipeline to be inspected and therefore the space available.

Equation (1) establishes the total information in GB to store for the system

$$Information = \frac{D}{ASR * 1024^3} * BT * NT. \quad (1)$$

Where D is the total Distance to inspect, ASR is the Axial Spatial Resolution, BT is the Bytes per Transducer, and NT is the Number of Transducers. For example, for a Distance (D) of 100 Km, an ASR of 3 mm, 37 BT, and 80 NT the total information to store is 91.89 GB. The selected memory chip was Micron MT29F128G08 with a CS of 16 GB. The chip specifications are compatible for 8, 16, 32 and 64 GB [11] (see Table 2). We selected this chip because of the availability for purchase it in small quantities and its surface mount type of a 48-TSOP device package. Dividing the total information of 91.89 GB by the CS of 16 GB the total of the memory chips is 6. In Section 5, we analyzed how with this amount of memory chips and this controller it is possible to cover a wide combination of inspection parameters, only by changing CS between the values of 8, 16, 32 and 64 GB.

TABLE 2
Memory Specification [11]

Characteristic	value
Page size	8,640 bytes (8192 + 448 spare)
Block size	256 pages
Plane size	2 plane x 4096 blocks
Data storage capacity	8, 16, 32 or 64 GB
Page program time	2,600 μs (max)

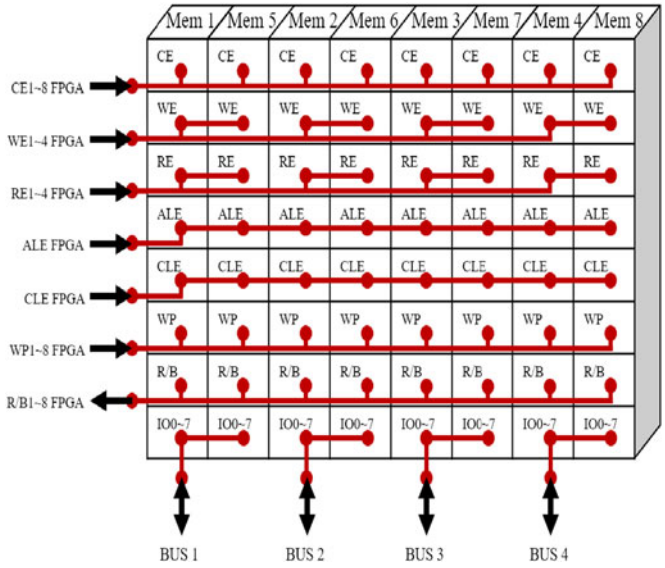

Fig. 2. Physical arrangement of the memory chips.

3.2 Physical Connection

A NAND flash memory has 15 pins for the interface. Eight pins are used to transfer data, commands, and address (IO0 to IO7). The control signals are five pins (CLE, WE, ALE, RE, and CE). Two pins are for hardware write protection (WP) and monitor device status (R/B). Fig. 2 shows the interleaving of the memory chips for our storage system, where six chips (1-3,5-7) are used to store the information corresponding to the ultrasound signals, and two chips (4, 8) stores the redundant information. We connected the chips in 2 arrays of 4 chips. The WE, RE, and 8 DIO pins are shared by pairs of chips (1 with 5, 2 with 6, 3 with 7, and 4 with 8) because they are the highest frequency signals.

Pins CE, WP, and R/B were considered individually for each chip. In this way, interfacing with a single memory chip is possible. ALE, CLE pins are shared by all chips. Of the 120 connection pins that initially would be necessary, with this arrangement only 66 pins were used. This arrangement allows the other pins of the FPGA to be used to carry out other processes such as the control of ultrasonic circuits, or for the telemetry of the PIG. The NAND flash R/B pin provides a hardware method of indicating whether a memory is ready or busy. Utilizing this signal, the controller knows when to start another process. This time depends on the (2) [11]

$$t_{lp} = t_P(lp) + t_P(lp-1) - clt(lp) - alt(lp) - dlt(lp). \quad (2)$$

Where lp is the last page, t_{lp} is the last page programming operation time, t_P is the page programming operation time, clt is the command load time, alt is the address load time, and dlt is the data load time. Due to (2), the programming time of each page is different. The interleaving allows the controller to write in the same page location of the four memory chips, which also keeps the programming times balanced to have similar writing times.

3.3 Memory Technology Device Layer

The seven asynchronous interface bus modes used to control the memory are bus idle, command input, address input, data

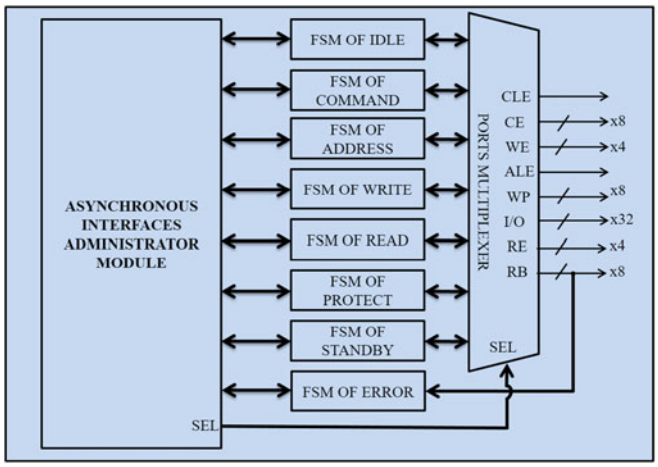

Fig. 3. Memory technology device layer.

output, data input, write protect, and standby mode. The correct combination of these modes allows the corresponding memory chip operation. Fig. 3 shows our MTD Layer. This MTD consists of 8 Finite State Machines (FSMs), where 7 of them are used for the direct interface with memory chips, and the other FSM monitors the 4 R/B outputs of the array.

The Asynchronous Interfaces Administrator Module (AIAM) is responsible for controlling the FSMs according to the operation to be performed. In the case of the page program operation: First, the AIAM activates the communication with the memory chips troughs the FSM of idle. Second, it sends the serial data input command activating the FSM of command. Third, it sends the address of the page to be written by the FSM of address. Fourth, it loads the data into the cache register of the selected memory chips (array 1 or 2) using the FSM of write, in this step, the FSM of write first loads the 8,192 bytes of information, and later the 448 bytes of the spare region of the page. Fifth, the AIAM activates the FSM of command to send the program command. This action initiates the page programming process in the chips. The AIAM can detect the end of a page program operation by monitoring the R/B output. R/B changes logic status from "1" to "0" when the page programming process starts and takes a maximum time t_P of 2600 μs to return to its logical state of "1" [11].

The task that the FSM of error performs is monitoring the 4 R/B outputs of the array with which the AIAM is working. If the program command is activated, and some R/B output does not change from "1" to "0" logic, then the FSM of error automatically discards the timeout for that chip. If the R/B output of a chip changes from "1" to "0" logic but does not rise after the maximum time t_P, then the FSM of error generates a flag for the writing process to continue. This action ensures that storage operation continues even though some memory chips fail during the pipeline inspection.

3.4 Information Protection

Fig. 4 shows how in each page program operation the information is written in a four memory chips array. In the first step, the controller writes the array one. This step includes chip 1 through 4, where the memory chip 4 stores the sum. In the second step, array two is written, which includes chips 5 through 8, where the chip 8 stores the product.

3.4.1 RAID-6 Architecture

In the FPGA, the NAND flash controller operates at 100 MHz. For a page program operation, when the AIAM loads the data into the cache register of the selected memory chips (array 1 or 2) using the FSM of write, the signal that determines the frequency at which

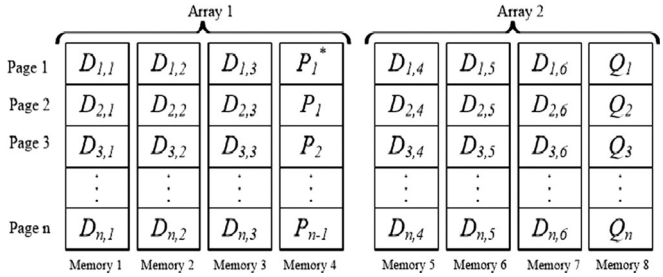

Fig. 4. Memory chips arrangement.

the controller sends the data is WE. The frequency of the WE pin for the memory chip operations is 10 MHz, so the FPGA has five clock cycles (cc) during which the data are placed in the port while information is being transferred to compute the ECC and the redundancy. While the controller transfers the 8,192 bytes of information, it calculates the redundancy P and Q and the ECC bits for the spare area. When the controller has finished sending the 8,192 bytes of information, it already has the ECC bits that it sends to the spare region of the page.

The first redundancy unit is based on xor-operation and is considered as the sum operator. Equation (3) shows this operation, where P refers to the sum of the data of each memory chip, i refers to the page number, and k corresponds to the memory number

$$P_i = \sum_{k=1}^{6}(D_{i,k}) \qquad 1 \leq i \leq n-1. \tag{3}$$

Each $D_{i,k}$ represents the information per page, and n is the total number of pages in memory. Q denotes the second redundancy unit. It refers to the sum of the data of each memory chip multiplied by a power of 2. To achieve this, we used the bit per bit method of operation proposed by Mastrovito [33]. The main advantage of this method is that it does not consume any RAM block of the FPGA and allows the calculation of the product in 1 cc. Equation (4) shows the product

$$Q_i = \sum_{k=1}^{6} 2^{k-1} \otimes (D_{i,k}) \qquad 1 \leq i \leq n. \tag{4}$$

Algorithm 1 shows the procedure performed for the redundancy computation. It has a 1-bit register named *Flat*. This register starts at "0" logic when the controller writes information in the array number 1 (see Fig. 4). The index j indicates the number of data within each page. For example, $p_{2,100}$ indicates that it is the data number 100 of the sum P_2, and $d_{30,5,200}$ indicates that the addressing is the data number 200 in memory 5 on page 30.

Algorithm 1. Procedure for the Computation of the Sum and Product within Each Page of Memory

Ensure: Flat is 1-bit register, $1 \leq j \leq 8191$
1: **if** *Flat* is 0 and falling edge of WE pin **then**
2: Download $p_{i,j-1}$
3: $p_{i,j} = d_{i,1,j} + d_{i,2,j} + d_{i,3,j}$
4: $q_{i,j} = 2^0 \otimes d_{i,1,j} + 2^1 \otimes d_{i,2,j} + 2^2 \otimes d_{i,3,j}$
5: **else if** *Flat* is 1 and falling edge of WE pin **then**
6: $p_{i,j} = d_{i,4,j} + d_{i,5,j} + d_{i,6,j} + p_{i,j}$
7: $q_{i,j} = 2^3 \otimes d_{i,4,j} + 2^4 \otimes d_{i,5,j} + 2^5 \otimes d_{i,6,j} + q_{i,j}$
8: Download $q_{i,j}$
9: **end if**

To perform all redundancy operations in 5 clock cycles, the controller has a dedicated module to work with 2 RAMs named P and

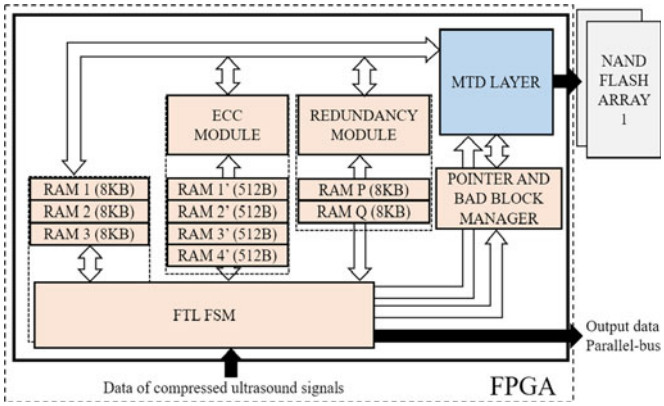

Fig. 5. Block diagram of the FTL designed.

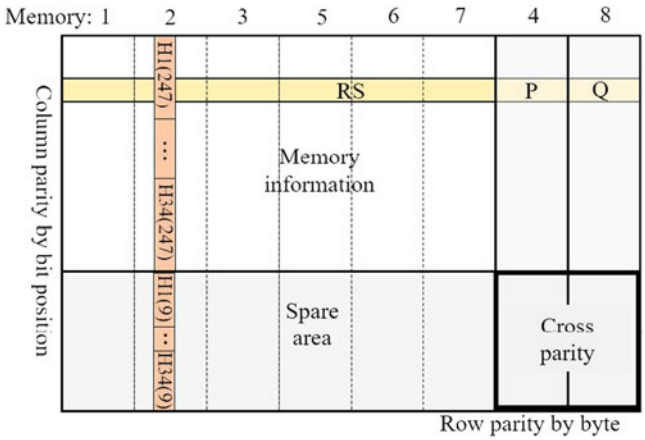

Fig. 6. Structure of data protection by Hamming (256,247) and RAID-6.

Q (see Fig. 5). The RAM P stores the sums and the RAM Q the products. The redundancy module begins operations on the falling edge of WE pin. In the first clock cycle (cc), the controller sets the 8-bit data $p_{i,j-1}$ in the BUS 4 from memory 4 and 8 (see Fig. 2). In the second and third cc, the sum of data from chips 1 to 3 and its product (lines 2 to 4) are calculated in parallel and are stored in the RAMs P and Q respectively.

When the writing of memory chips 5 to 8 begins, the controller works with array number 2, and the *Flat* register changes to value 1. In the first and second cc, the controller computes the total sum and product and stores them in RAMs P and Q respectively. In the third cc, the controller sets the product $q_{i,j}$ in the BUS 4.

We do not save the backup of the sum on the first page of the array 1 (see Fig. 4), so the first P is denoted by $P*$. In the second write process of the array 1, the sum P_1 of the data $D_{1,k}$ is written, and in the second write process of the array 2 the product Q_2 of the data $D_{2,k}$ is written. In this way, the only page that does not have a complete redundancy of the information is the last one to be written (page n).

3.4.2 Hamming Codification

The MT29F128G08 memory must have an error correcting code (ECC) for 24 bits per every 1,080 bytes of data [11]. We used the ECC technique of extended Hamming (256,247) [27]. This technique permits the correction of one error or to detect up to three. Because the Hamming method performs bitwise operations, we developed the encoding of the information vertically. We used an ECC module for each one of the 8 I/O data lines. In this way, 32 Hamming codes are computed in parallel when the controller sends the information to the chips. Fig. 6 shows how we organized the ECC bits and the redundancy bytes in the memory arrangement.

The Algorithm 2 shows the procedure performed in 5 clock cycles for the ECC bits. Our algorithm uses two registers of 8 bits named Reg and u as support. Reg is the counter that determines which bit of the register u computes the xor-operation with the input bit (a memory I/O pin). The u register stores the ECC bits calculated by Hamming. In the first cc, the ECC module increases the Reg register; then it computes the xor-operation of each incoming bit, and the corresponding position bit (lines 3 through 7). In the second cc, the ECC module increases Reg register, and if its value is a power of 2, Reg again is increased. The ECC module repeats this process for each 8-bit data sent to memory until Reg changes to 0 (Reg is an 8-bit register, so the next value of 255 is 0), indicating that 247 bits have been computed. Finally, in the third, fourth and fifth cc the ECC module computes the xor-operation of 8-bits of the register u (line 12) and writes the data to the RAM used for ECC (line 13). For our data storage system will be compatible with others industrial applications, it is necessary that the FPGA will be 10 times faster than the frequency of the WE pin. In this manuscript, the discussion is limited only to showing the controller structure to generate redundancy and ECC codes during data transfer. However, the interaction between ECC by Hamming and RAID-6 by RS is an important topic which is kept open for future research.

Algorithm 2. Parity Bits Calculation x32, Hamming Extended (256,247)

1: $Reg = 3, u = 00000000, sum = 0, bit, j$
Ensure: Reg and u are 8-bit register, $0 \leq j \leq 7$
2: **while** $Reg \neq 0$ **do**
3: $Reg = Reg + 1$
4: $sum = sum$ **xor bit**
5: **if** $Reg(j) == 1$ **then**
6: $u_j = u_j$ **xor bit**
7: **end if**
8: **if** $Reg == 4, 8, 16, 32, 64$ or 128 **then**
9: $Reg = Reg + 1$
10: **end if**
11: **end while**
12: $sum = sum$ **xor** $\mathbf{u_0}$ **xor** $\mathbf{u_1} \ldots$ **xor** $\mathbf{u_7}$
13: Transfer the data to ECC RAM

3.5 Flash Translation Layer

Fig. 5 shows the architecture of our FTL. The FTL-FSM of the controller allows direct access to RAMs 1, 2, and 3. The RAMs 1', 2', 3', 4', P, and Q are not writable directly, only the internal modules can write to them. The connection of the 4 NAND flash chips in parallel increases the bandwidth by 4, so in each page program operation, the information to be stored increases from 8 to 32 KB; however, the inspection process only considers 24 KB of data, since 8 KB corresponds to a redundancy memory. To start a page program operation, it is necessary to fill the RAMs 1, 2 and 3 as much as possible, since the memory MT29F128G08 only allows one programming operation per page before an erase operation [11]. Equation (5) shows the Number of Signals acquired needed to Full a memory-Page ($NSFP$)

$$NSFP = \frac{24576}{BT}. \quad (5)$$

The number 24,576 is due to the sum of the capacity-bytes of the 3 RAMs (1, 2 and 3) that are available for storing data. Considering 37 BT, we have a total of 664 $NSFP$. In this case, 8 bytes

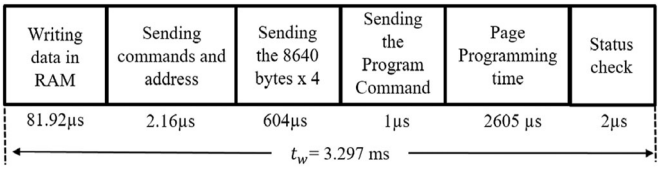

Fig. 7. Acquisition and storing of 24 KB of information.

remain on each page programming process. These 8 bytes store additional inspection data.

4 System Implementation and Performance Evaluation

We implemented the NAND flash controller in the Xilinx Virtex-4 FPGA (XC4VLX25). Through simulation, we determined that the controller take a maximum time of the page program operation (t_w) of 3.297 ms. Fig. 7 shows the time analysis. The time t_w includes 6 steps: 1) Loading the data into RAM (24 KB). 2) Sending the commands and addresses to the NAND flash chips. 3) Sending the 8,640 bytes (8,192 bytes of information and 448 of the ECC). 4) Sending the program command. 5) Waiting for the maximum programming time (t_p). 6) Checking for the status of the operation. The computation of the redundancy and the ECC bits does not appear within the six steps, since the controller computes them internally, eliminating them from the sequential process.

The condition that governs the writing process in real-time requires that $t_w < t_{fram}$, where t_{fram} is the time to fill the 24 KB of information, resulting from the multiplication of $NSFP$ of 664 per t_a of 20.48 μs. Giving a time t_{fram} of 13.598 ms. Therefore, the controller satisfies the condition $t_w < t_{fram}$.

4.1 Comparison of the Data Storage System

We validated the ECC and redundancy modules by comparing the results against the Matlab Communications system Toolbox. Table 3 summarizes the space used in the FPGA of the proposed controller against other existing controllers FPGA-based. The main advantage of the proposed system is that it focuses on the low consumption of slices and LUTs, mainly due to the optimization of the controller to work with raw data.

4.2 Interleaving Validation

We carried out the characterization test of the page program operation in a sequential way. We used the Hantek oscilloscope model DSO-2090 USB for the data capture, with an error per test of +/- 0.8 μs. Fig. 8 shows the controller characterization when writing on the 4 memory chips for 64 consecutive pages. From the test, we observed that none of the 64 writing processes reached 3.297 ms. Also, we corroborated interleaving behavior. Fig. 8 shows that in memory 1, when programming page 48, and in memory 3 when programming page 7, the page program operation time takes less than the expected time. However, this behavior does not affect the controller since the time does not exceed 3.297 ms.

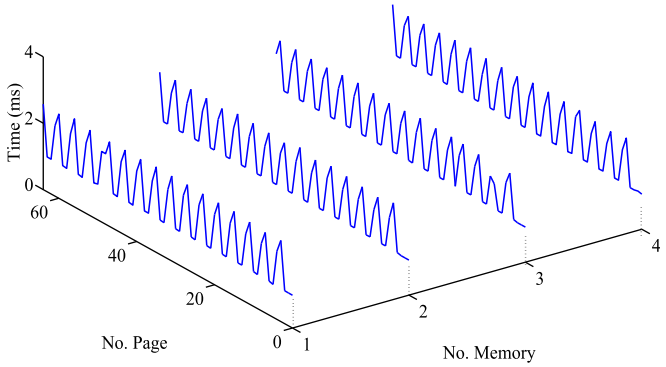

Fig. 8. Test of page program operation time and interleaving validation.

4.3 Data Flow Validation

Equation (6) shows the variables necessary for the data flow estimation of the proposed controller

$$Data flow = \frac{NT * BT * F}{1024^2}. \quad (6)$$

Where NT is the Number of Transducers, BT is the Bytes per Transducer, and F is the maximum Frequency at which data bursts are generated. F can be expressed as the PRF, or the division of the PIG velocity by the ASR. A crucial test was to find the maximum data flow that the controller can reach to store raw data. Fig. 9 shows that the controller can reach a data flow of 14.05 MB/s if there is no fault in the memory chips response. According to Equation (6), with this data flow, if F is equal to 600 Hz, with 37 BT, the controller could store the information of up to 664 transducers (NT). This amount of transducers corresponds to 8 channels of ultrasound working at their maximum performance (8 channels per 80 transducers, according to Fig. 1). Equation (1) gives the distance (D) that the controller could store with the 8 channels.

The worst cases in terms of the time for the page program operation are when the program command is activated, and the R/B pin of the chip changes logic state from "1" to "0" but does not rise after the maximum time t_P. If this incorrect response is presented in one or two memory chips of the same array, the data flow drops to 9.68 MB/s, which is equivalent to 457 transducers, or 5 channels of ultrasound at maximum performance. If the fault occurs in 2 memory chips, one in each array, the data flow drops to 7.28 MB/s, which is equivalent to 343 transducers, or 4 channels of ultrasound at maximum performance.

4.4 Using the Controller with Other Data Compressors

We tested the controller with two other compression techniques widely used in PIGs. The goal was to check the page-level mapping and the controller compatibility with other compressors. The first technique was ALOK [4], [35]. This technique determinates the maximums in a half wave of the ultrasonic signal. The ALOK technique recognizes only those data values which fall within a time window, and are greater than or equal to the predecessor "i" of 8-bit data and which is greater than the successive "k" of 8-bit data

TABLE 3
Comparison of Data Storage System with Existing SSD Structures Based on FPGA

Logic utilization	FPGA	Slices	LUTs	Block RAMS	NAND flash memory chips
FPGA-based SSD [13]	Virtex-5 (XC5VLX330)	18 720	22 960	not specified	40
Ozone (O3) Flash memory controller [34]	Virtex-5 (XC5VFX130T)	6 253	11 453	59	not specified
Hydra SSD [10]	Virtex-4 (XC4VFX100)	25 159	36 299	139	32
Our design	Virtex-4 (XC4VLX25)	1 301	2 465	24	8

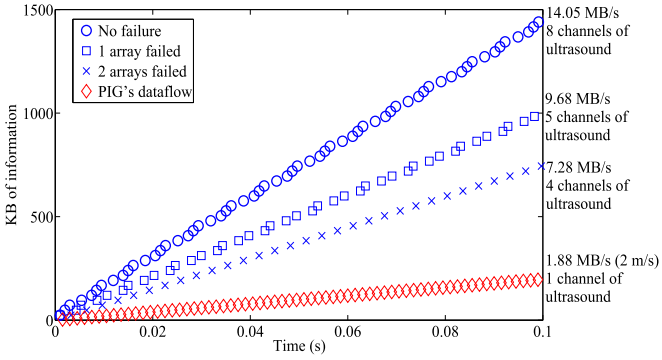

Fig. 9. Data flows for different memory failures in the arrays.

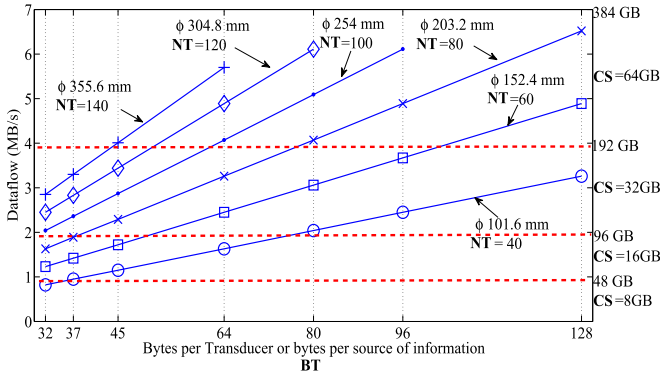

Fig. 10. Graph for the selection of the CS of the memory, depending on data flow and the BT or bytes per source of information.

of the positive half-wave of the signal. This technique has a compression ratio of 7.7 percent. The second technique was that proposed by Lei et al. [5]. This technique stores all values that do not fall within a pre-set threshold of zero. The technique places an 8-bit value of zero and a counter register that indicates the number of zeros that are consecutive. The technique has a compression ratio of 12 percent. Table 4 shows the results obtained using one-channel ultrasound system to its maximum performance, indicating that our storage system is compatible with the most used compressors in PIGs.

5 Discussion

In this work, we set the number of memory chips as 8, where 6 of them are for storage and 2 for redundancy. The CS of the chips can be 8, 16, 32 or 64 GB, giving a total capacity to store information of 48, 96, 192 and 384 GB respectively. Equation (1) allows the calculation of the total capacity during the inspection process. The controller can maintain a data flow up to 7.28 MB/s. Equation (6) gives the maximum data flow generated by an inspection process. Fig. 10 shows a graph that allows selecting the CS of the memory chips according to BT and data flow. In Fig. 10, the diagonal lines facilitate the selection of the CS of the chips according to the diameter of the pipeline to be inspected. The diagonal lines were calculated considering the circumferential resolution of 8 mm, an ASR of 3 mm, a PIG velocity of 2 m/s, and a Distance (D) of 100 Km. For example, if a PIG inspects a 254 mm diameter pipeline with a data compressor that generates 45 BT, then the PIG should have 100 transducers (NT), and the CS of the memory chips should be 32 GB. If the pipeline diameter is greater than 355.6 mm, the inspection system needs to handle more than 140 transducers, so it would be necessary to duplicate the storage system to achieve inspecting 100 Km.

This data storage system can be useful for other applications, where is necessary to store continuous raw data. To achieve this, in (6) the controller interprets NT as the sources of information, BT as the maximum number of bytes for each source of information and F as the maximum frequency at which data bursts will be generated. According to the analysis presented, this controller could be used with another compressor of signals, and for other processes where raw data generation is continuous.

TABLE 4
Controller Compatibility with Other Compressors Considering 600 Hz of PRF, 3 mm of ASR, NT of 80, and a Distance of 100 Km

Compressor	BT	Data flow (MB/s)	$Tfram$	Total information (GB)
Soto et al. [1]	37	1.88	13.59 ms	91.89
ALOK [4], [35]	79	4.01	6.36 ms	196.12
Lei et al. [5]	122	6.20	4.11 ms	303.00

6 Conclusions

The development of adaptable technologies has become an area of interest for PIG-subsystems [2], such as storage systems customizable for these devices, where the goal is to reduce costs. In this sense, we presented and validated the development of a NAND flash controller for a pipeline inspection process, where we used the RAID-6 scheme to protect the integrity of the information. The Reed Solomon redundancy and Hamming parity bits are computed in real-time while the controller writes the information in the cache register of the selected die in the memory chips. The main advantages of this controller are 1) Improved data flow by the elimination of sequential ECC and redundancy computation processes. 2) Usefulness for multiple pipeline diameters and different amounts of bytes per transducer, leading to the application in other techniques of compression and reduction of signals. 3) Storage validation when the velocity of the PIG reach up to 2 m/s. 4) System compatibility for other applications with similar characteristics.

Acknowledgments

This work has been partially funded by the Fondo Sectorial de Investigación para el Desarrollo Aeroportuario y la Navegación Aérea (ASA-CONACYT), through Project Number 242864. All authors would like to thank Joseph Moeller for his help in improving the English manuscript. The first author would also like to thank the anonymous reviewers for their detailed comments which helped to improve the quality of this paper.

References

[1] J. Soto-Cajiga, J. Pedraza-Ortega, C. Rubio-Gonzalez, M. Bandala-Sanchez, and R. D. J. Romero-Troncoso, "FPGA-based architecture for real-time data reduction of ultrasound signals," *Ultrason.*, vol. 52, no. 2, pp. 230–237, 2012.
[2] Z. Liu and Y. Kleiner, "State of the art review of inspection technologies for condition assessment of water pipes," *Meas.*, vol. 46, no. 1, pp. 1–15, 2013.
[3] W. M. Alobaidi, E. A. Alkuam, H. M. Al-Rizzo, and E. Sandgren, "Applications of ultrasonic techniques in oil and gas pipeline industries: A review," *Amer. J. Oper. Res.*, vol. 5, no. 04, 2015, Art. no. 04.
[4] G. Dobmann, O. A. Barbian, and H. Willems, "State of the art of in-line non-destructive weld inspection of pipelines by ultrasonics," *Russian J. Nondestructive Testing*, vol. 43, no. 11, pp. 755–761, 2007.
[5] H. Lei, Z. Huang, W. Liang, Y. Mao, and P. W. Que, "Ultrasonic PIG for submarine oil pipeline corrosion inspection," *Russian J. Nondestructive Testing*, vol. 45, no. 4, pp. 285–291, 2009.
[6] A. S. Hashim, B. Grămescu, and C. Nițu, "State of the art survey on using robots in oil and gas industry," in *Proc. Int. Conf. Mechatron. Cyber-Mixmechatron.*, 2017, pp. 177–185.
[7] C. de Normalización de Petroleos Mexicanos e Instituciones Subsidiarias, "Inspección de ductos de transporte mediante equipos instrumentados," PEMEX-MEXICO, Tech. Rep., 2012.
[8] A. S. of Mechanical Engineers-ASME, "B31.4 pipelines transportation systems for liquid hydrocarbons and other liquids," ASME, Tech. Rep., 2006.
[9] L. Wu, N. Xiao, F. Liu, Y. Du, S. Li, and Y. Ou, "Dysource: A high performance and scalable NAND flash controller architecture based on source synchronous interface," in *Proc. 12th ACM Int. Conf. Comput. Frontiers*, 2015, Art. no. 25.

[10] Y. J. Seong, E. H. Nam, J. H. Yoon, H. Kim, J.-Y. Choi, S. Lee, Y. H. Bae, J. Lee, Y. Cho, and S. L. Min, "Hydra: A block-mapped parallel flash memory solid-state disk architecture," *IEEE Trans. Comput.*, vol. 59, no. 7, pp. 905–921, Jul. 2010.
[11] I. Micron Technology, "L74a 64gb 128gb 256gb asyncsync nand revg," Micron, Tech. Rep., 2009.
[12] M. Balzer and H. Stripf, "Online data reduction with a DSP-FPGA multiprocessor system," in *Proc. 14th Int. Conf. Digit. Signal Process.*, 2002, pp. 819–822.
[13] Y. Cai, E. F. Haratsch, M. McCartney, and K. Mai, ''FPGA-based solid-state drive prototyping platform," in *Proc. IEEE 19th Annu. Int. Symp. Field-Programmable Custom Comput. Mach.*, 2011, pp. 101–104.
[14] Z. Li, F. Wang, J. Liu, D. Feng, Y. Hua, W. Tong, and S. Zhang, "A user-visible solid-state storage system with software-defined fusion methods for PCM and NAND flash," *J. Syst. Archit.*, vol. 71, pp. 44–61, 2016.
[15] Y. Deng and J. Zhou, "Architectures and optimization methods of flash memory based storage systems," *J. Syst. Archit.*, vol. 57, no. 2, pp. 214–227, 2011.
[16] C. Park, P. Talawar, D. Won, M. Jung, J. Im, S. Kim, and Y. Choi, "A high performance controller for NAND flash-based solid state disk (NSSD)," in *Proc. IEEE 21st Non-Volatile Semicond. Memory Workshop*, 2006, pp. 17–20.
[17] J.-U. Kang, J.-S. Kim, C. Park, H. Park, and J. Lee, "A multi-channel architecture for high-performance NAND flash-based storage system," *J. Syst. Archit.*, vol. 53, no. 9, pp. 644–658, 2007.
[18] A. Gupta, Y. Kim, and B. Urgaonkar, ''DFTL: A flash translation layer employing demand-based selective caching of page-level address mappings," *ACM SIGPLAN Notices*, vol. 44, no. 3, pp. 229–240, Mar. 2009. [Online]. Available: http://doi.acm.org/10.1145/1508284.1508271
[19] C. Gao, L. Shi, C. Ji, Y. Di, K. Wu, J. Xue, and E. H.-M. Sha, "Exploiting parallelism for access conflict minimization in flash-based solid state drives," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 1, pp. 168–181, Jan. 2018.
[20] E.-Y. Chung, C.-I. Son, K. Bang, D. Kim, S.-M. Shin, and S. Yoon, "A high-performance solid-state disk with double-data-rate NAND flash memory," arXiv:1502.02239, Feb. 2015.
[21] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," *ACM SIGMOD Rec.*, vol. 17, no. 3, pp. 109–116, 1988.
[22] J. Luo, L. Fan, Z. Chen, and Z. Li, "A solid state drive architecture with memory card modules," *IEEE Trans. Consum. Electron.*, vol. 62, no. 1, pp. 17–22, Feb. 2016.
[23] A. A. McEwan and I. Mir, "An embedded FTL for SSD raid," in *Proc. Euromicro Conf. IEEE Digit. Syst. Des.*, 2015, pp. 575–582.
[24] J. S. Plank,"The RAID-6 liberation code," *Int. J. High Perform. Comput. Appl.*, vol. 23, no. 3, pp. 242–251, 2009.
[25] I. S. Reed and G. Solomon,"Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, 1960.
[26] J. S. Plank, et al., "A tutorial on Reed-Solomon coding for fault-tolerance in raid-like systems," *Softw. Practice Experience*, vol. 27, no. 9, pp. 995–1012, 1997.
[27] Y. Jiang, *A Practical Guide to Error-Control Coding Using Matlab*. Norwood, MA, USA: Artech House, 2010.
[28] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures," *IEEE Trans. Comput.*, vol. 44, no. 2, pp. 192–202, Feb. 1995.
[29] L. Xiang, Y. Xu, J. Lui, Q. Chang, Y. Pan, and R. Li, "A hybrid approach to failed disk recovery using RAID-6 codes: Algorithms and performance evaluation," *ACM Trans. Storage*, vol. 7, no. 3, 2011, Art. no. 11.
[30] C. Jin, H. Jiang, D. Feng, and L. Tian, "P-Code: A new RAID-6 code with optimal properties," in *Proc. 23rd Int. Conf. Supercomput.*, 2009, pp. 360–369.
[31] J. S. Plank, "A new minimum density RAID-6 code with a word size of eight," in *Proc. 7th IEEE Int. Symp. Netw. Comput. Appl.*, 2008, pp. 85–92.
[32] C. Yang, Y. Emre, C. Chakrabarti, and T. Mudge, "Flexible product code-based ECC schemes for MLC NAND flash memories," in *Proc. IEEE Workshop Signal Process. Syst.*, 2011, pp. 255–260.
[33] T. Zhang and K. K. Parhi, "Systematic design approach of mastrovito multipliers over GF (2/sup m/)," in *Proc. IEEE Workshop Signal Process. Syst.*, 2000, pp. 507–516.
[34] E. H. Nam, B. S. J. Kim, H. Eom, and S. L. Min, "Ozone (O3): An out-of-order flash memory controller architecture," *IEEE Trans. Comput.*, vol. 60, no. 5, pp. 653–666, May 2011.
[35] O. Barbian, et al., *IIW Handbook on Automated UT Inspection Systems*. Düsseldorf: DVS Publishing House, 2003.