The Implementation of Triple-Disk-Failure Tolerant RAID

[1]Ming-Haw Jing, [2]Yao-Tsu Chang, [3]Chong-Dao Lee, and [1]Kai Su

[1]Department of Information Engineering, I-Shou University
[2]Department of Applied Mathematics, I-Shou University
[3]Department of Communication Engineering, I-Shou University
Kaohsiung, Taiwan

mhjing@isu.edu.tw, ytchang@isu.edu.tw,
chongdao@isu.edu.tw, isu10103004m@isu.edu.tw

Abstract – **A brand new algorithm of RAID for tolerating triple disk failures is presented in this paper. A prototyping system is built on Altera FPGA DE2-115 board. To build this system, the PC and FPGA board are integrated to perform as the system development platform to increase the visualization of details inside the system. Also, this platform is used to transfer the designs from the original simulation software into engineering system using HW and SW co-design concept. The advantage of this platform is that the developer can do the algorithm proof, component/module design, testing and system integration efficiently. In the future, such triple-failure recovery technique should be widely used in storage system, network and cloud computing.**

Keywords**—RAID; Fault-tolerance; HDD failure; FPGA; platform**

I. Introduction

In recent years, the function of web services and data center grows sharply. Many Internet/cloud service providers offer huge online storage services to the users. How to restore the lost data, increase access speed, and increase security to allow users to have confidence in using cloud storage, is an important subject. The RAID (Redundant Arrays of Inexpensive Disks) technology for storage reliability can offer an effective way to large online storage services.

The size of RAID may combine from multiple hard disks to ultra large number of hard drives. In internet application, the concept of RAID can have data blocks spread into multiple hard drives or domain in network which may increase the reading speed and tolerate failures. In most RAID, restoration algorithms are used to generate redundant data and having the ability to recover data from failure drives.

Nowadays there are many algorithms used to tolerate two disk failures, such as RAID 6. Because today's cloud service providers have the storage demand growing abruptly with the requirement for data integrity. So, tolerating more data errors is a trend. The RAID 7.3 design can provide simultaneous three-fault tolerance or may extend higher demands. This paper proposes a RAID 7.3 algorithm and attempts to establish a RAID prototype system. This algorithm uses the math of finite field. The operation of finite field is easily realized in the software, but the software version of RAID 7.3 system will consume large amounts of CPU time and power consumption. So this project will develop hardware modules to reduce these problems relating to speed and efficiency. Fortunately, the FPGA (Field-Programmable Gate Array) has high advantage to implement the prototyping system with the requirement of the hardware software design. In FPGA, one can replace slow software modules with hardware and then it can lower the power consumption of the modules. Also, FPGA has the advantage of low cost, good developing efficiency, long-term maintainability, online update capabilities and other features. Therefore, in view of the above reason, we choose to use Altera DE2-115.

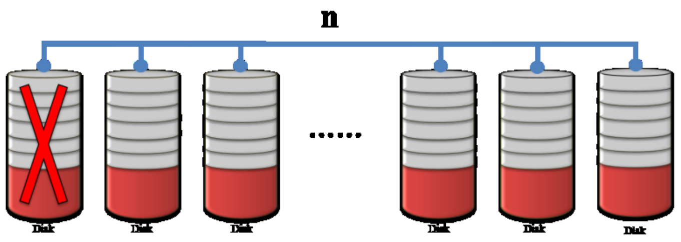

Fig. 1A The RAID has single disk failure

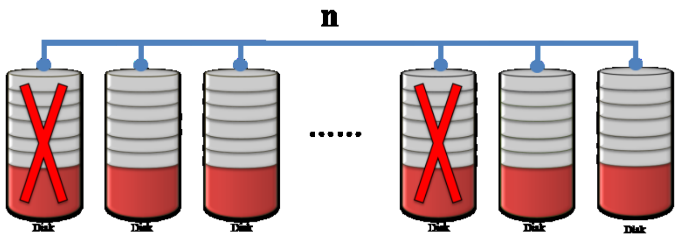

Fig. 1B The RAID has double disk failures

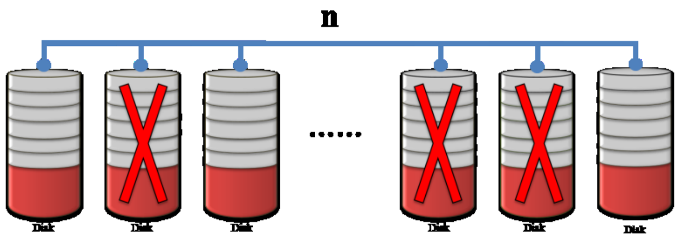

Fig. 1C The RAID has triple disk failures

The RAID technology for two erasures or less, has used for many years, however, we can develop a new solutions of triple disk failures and may even more expanded to multi-disk failures [1]. The RAID 7.3 can tolerate triple disk failures, as shown in Fig. 1A, 1B, 1C. It not only can only be applied to hard disk, but also may apply to channels in internet such as hard drive, RAID, networking, cloud, supercomputers, etc.

The future wireless Internet bandwidth has been increased and magnetic hard disk price is cheaper, huge storage will be used. If there are high fault tolerances in network storage, it will bring users to have a higher level of confidence to internet storage. In future the network storage will have not only the size of storage, but also the have requirement in expanding performance and reliability of the storage in server. If most customers choose to store data in the network, the future market of reliable storage will be very big.

II. Theoretical Background

Theoretical background of this paper we will be briefly covered here.

A. Finite Field：

The size of Galois Field GF(2^8), as Anvin had mentioned [2], can be small or large in this application, however, a smaller field would limit the number of drives possible, and a larger field would require extremely large hardware/tables. Therefore, we adopt the most efficient GF(2^8) in here.

GF(2^8) allows up to a number of 255(2^8-1) hard drives used in a data storage. It's because GF(2^8) is a generally used cyclic field. Here we use GF(2^8)= $x^8 + x^4 + x^3 + x^2 + 1$ as irreducible polynomial in this application.

Galois field has several characteristics in following explanation.

- The addition field operator (+) is represented by XOR.
- The 0 is represented by {00} in hexadecimal.
- A + A = A - A = {00}.
- $A^n \cdot A^{-n}$ = {01}.
- If S= $a_7x^7 + a_6x^6 + a_6x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a^0$， x={02}，S multiply （・) by x is implemented by the following relations：
  $S \cdot x^0 = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0$
  $S \cdot x^1 = a_6x^7 + a_5x^6 + a_4x^5 + (a_7+a_3)x^4 + (a_7+a_2)x^3 + (a_7+a_1)x^2 + a_0x^1 + a_7$
  $S \cdot x^2 = a_5x^7 + a_4x^6 + (a_7+a_3)x^5 + (a_7+a_2+a_6)x^4 + (a_7+a_1+a_6)x^3 + (a_0+a_6)x^2 + a_7x^1 + a_6$

The above presentation is considered as a Linear Feedback Shift Registers (LFSR) by hardware engineer as shown in Fig. 2, however, the mathematicians are considered that is a Boolean polynomial multiplication modulo the irreducible polynomial. The polynomial, $x^8 = x^4 + x^3 + x^2 + 1$={1D}, is used in RAID 6 core algorithms.

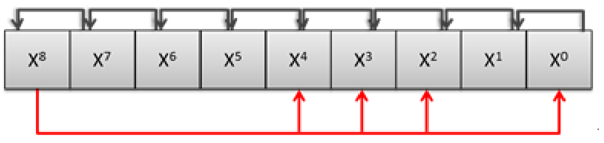

Fig. 2 LFSR

As an example, the power of {02} is presented in Table I. From Table I, if we use $x^8 = x^4 + x^3 + x^2 + 1$={1D} as the irreducible polynomial, there are all 255 elements can be created.

TABLE I. The presentation of {02^n}

n	0	1	2	3	4
{02^n}	{02^0}	{02^1}	{02^2}	{02^3}	{02^4}
results	{01}	{02}	{04}	{08}	{10}
n	5	6	7	8	9
{02^n}	{02^5}	{02^6}	{02^7}	{02^8}	{02^9}
results	{20}	{40}	{80}	{1D}	{3A}
n	10	11	12	13	14
{02^n}	{02^{10}}	{02^{11}}	{02^{12}}	{02^{13}}	{02^{14}}
results	{74}	{E8}	{CD}	{87}	{13}

⋮

n	251	252	253	254	255
{02^n}	{02^{251}}	{02^{252}}	{02^{253}}	{02^{254}}	{02^{255}}
results	{D8}	{AD}	{47}	{8E}	{01}

B. Parity Check：

In general checking application, the commercial RAID uses simple parity checking mostly, as shown in Fig. 3A [3]. In encoding procedure, we firstly XOR the original information to get a redundant data as called parity, we have 8 Bytes in this example. This redundant parity is stored with the information together and is called codeword which is expanded from 8 Bytes into 9 Bytes.

After received this information, as shown in Fig. 3B, the received data is also XORed to get a new parity. We check the new parity with the received one for error checking. These checking operations may only limit to the error detection function.

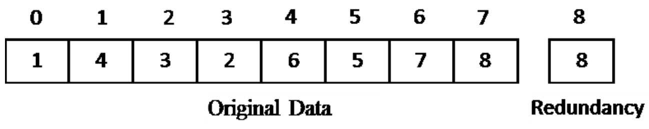

Fig. 3A A group parity check

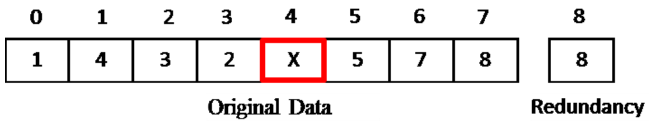

Fig. 3B An erasure

In this situation, if one wants to correct the error and the error position is known, it is called erasure. Taking Fig. 3B as an example, the error position is known so that one may collect all other data as well as the redundant parity to do simple XOR operation to recover the data, it is 6. Since there is only a redundant parity, one cannot correct the error with unknown position.

From above, there is boundary for error correction with respect to the redundant information. One redundancy may only resolve one set of information for correction because one error correction needs two set of information, including position and value of this data in protected array.

About the single erasure as above, we assume that we know the position of the fail disk. In case of one more disk failed, one redundancy will not be able to recover this disk and is called a crash of the storage system.

C. *RAID 6* ：

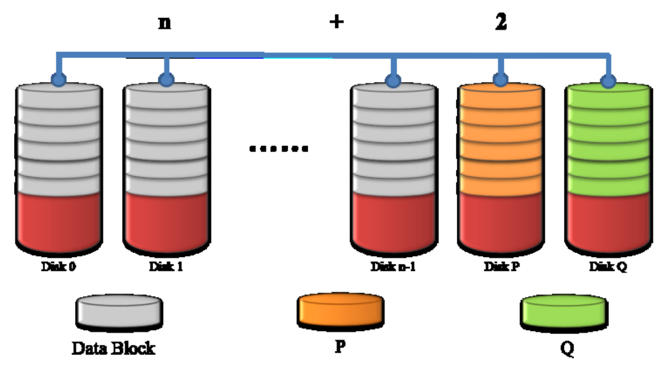

Fig. 4A The parity P and Q are stored in independent disks in RAID 6

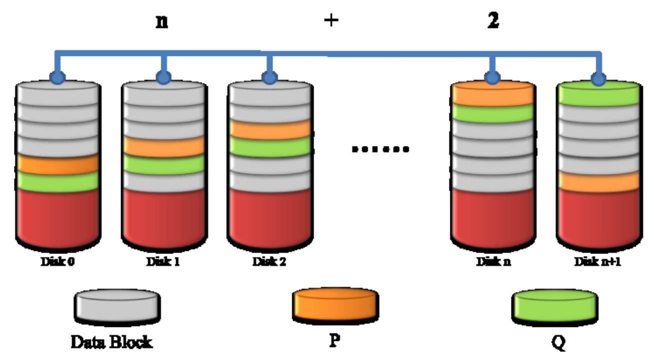

Fig. 4B The parity P and Q are spread into all disks in RAID 6

RAID 6 has one more parity checker, Q, out of RAID 5, it is enhanced to tolerate double disk failures. For the development of RAID 6, there are many researches using different coding methods to achieve higher performance with different algorithms.

In general, there are two designs of data array in RAID 6, the horizontal codes as shown in Fig. 4A and vertical codes as shown in Fig. 4B [5]. The main differences between those two are that the arrangement of parity checkers is in different directions, such as in row, diagonal or column direction of data array. Different codes may have various designs and usages; they include EVENODD [6], RDP [7], Liberation Code [8], Anvin, B-Code [9], X-Code [10], P-Code [11], etc.

D. *RAID 7* ：

In design of RAID 7 [4], there is no unique standard in names. Recently, RAID 7.1 is denoted for single disk failure, like RAID 5; RAID7.2 is for simultaneous double disk failures, like RAID 6; RAID 7.3 is for simultaneous triple disk failures.

III. RAID 7.3

In this section, RAID7.3 algorithm is explained with the definition, the structure is as Fig. 5.

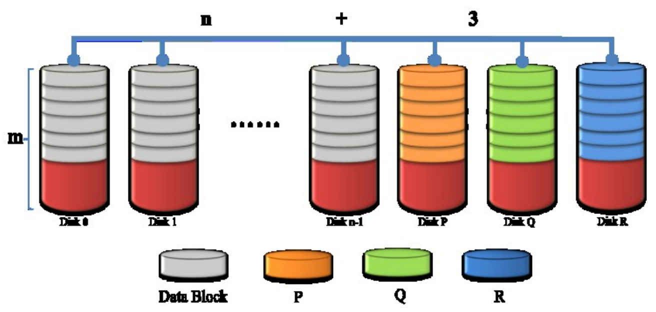

Fig. 5 The structure of RAID 7.3

Firstly, we define parameters, *D*, is the data in data disks, *n* is the number of data disks, *m* is the number of the row array in the whole set of data disks, and P, Q, R are 3 independent parities in each row array. Here the whole three set of P, Q, R are assumed storing in three parity disks. The P, Q, R in each row array are created by one set of functions $\{g_0 , g_1 , g_2 \}$ using the math of finite field. The definition of P, Q, and R is defined as follow:

$$P = g_0^0 \cdot D_0 + g_0^1 \cdot D_1 + g_0^2 \cdot D_2 + \cdots + g_0^{n-1} \cdot D_{n-1}$$

$$Q = g_1^0 \cdot D_0 + g_1^1 \cdot D_1 + g_1^2 \cdot D_2 + \cdots + g_1^{n-1} \cdot D_{n-1}$$

$$R = g_2^0 \cdot D_0 + g_2^1 \cdot D_1 + g_1^2 \cdot D_2 + \cdots + g_2^{n-1} \cdot D_{n-1}$$

The above equation may also be present as a matrix form as:

$$\begin{bmatrix} g_0^0 & g_0^1 & \cdots & g_0^n \\ g_1^0 & g_1^1 & \cdots & g_1^n \\ g_2^0 & g_2^1 & \cdots & g_2^n \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \\ \vdots \\ D_n \end{bmatrix} = \begin{bmatrix} P \\ Q \\ R \end{bmatrix}$$

Here, $\{g_0 , g_1 , g_2 \}$ is actually represented as $\{1, 2, 4\}$. The choice of parameter g affects the value of *n* or the maximum number of disks in the RAID, and in this case, the use of {1, 2, 4} supports the size of n≦255.

To correct erasure(s), there are three situations including single, double, or triple disk failure(s). There are corresponding methods of various situations in solving the erasure(s), as shown in Table II.

TABLE III A.
Solving Equation for single disk failure

Single disk failure	
Error location	Solving Equation
D_x	$D_x = P + \sum_{i=0,i\neq x}^{n-1} g_0^i \cdot D_i$
P	$P = \sum_{i=0}^{n-1} g_0^i \cdot D_i$
Q	$Q = \sum_{i=0}^{n-1} g_1^i \cdot D_i$
R	$R = \sum_{i=0}^{n-1} g_2^i \cdot D_i$

TABLE IIII B.
Solving Equation for double disk failures

Double disk failures	
Error location	Solving Equation
P & Q	$P = \sum_{i=0}^{n-1} g_0^i \cdot D_i$ $Q = \sum_{i=0}^{n-1} g_1^i \cdot D_i$
P & R	$P = \sum_{i=0}^{n-1} g_0^i \cdot D_i$ $R = \sum_{i=0}^{n-1} g_2^i \cdot D_i$
Q & R	$Q = \sum_{i=0}^{n-1} g_1^i \cdot D_i$ $R = \sum_{i=0}^{n-1} g_2^i \cdot D_i$
D_x & P	1.Get D_x $D_x = \frac{Q+\sum_{i=0,i\neq x}^{n-1} g_1^i \cdot D_i}{g_1^x}$, or $D_x = \frac{R+\sum_{i=0,i\neq x}^{n-1} g_2^i \cdot D_i}{g_2^x}$ 2.Then $P = \sum_{i=0}^{n-1} g_0^i \cdot D_i$
D_x & Q	1.Get D_x $D_x = P + \sum_{i=0,i\neq x}^{n-1} g_0^i \cdot D_i$, or $D_x = \frac{R+\sum_{i=0,i\neq x}^{n-1} g_2^i \cdot D_i}{g_2^x}$ 2.Then $Q = \sum_{i=0}^{n-1} g_1^i \cdot D_i$
D_x & R	1.Get D_x $D_x = P + \sum_{i=0,i\neq x}^{n-1} g_1^i \cdot D_i$, or $D_x = \frac{Q+\sum_{i=0,i\neq x}^{n-1} g_1^i \cdot D_i}{g_1^x}$ 2.Then $R = \sum_{i=0}^{n-1} g_2^i \cdot D_i$
D_x & D_y	1.Get P_{xy}&Q_{xy} $P_{xy} = \sum_{i=0,i\neq x,i\neq y}^{n-1} g_0^i \cdot D_i$ $Q_{xy} = \sum_{i=0,i\neq x,i\neq y}^{n-1} g_1^i \cdot D_i$ 2.Then $D_x = \frac{g^{-x} \cdot (Q+Q_{xy}) + g^{y-x} \cdot (P+P_{xy})}{g^{y-x}+\{01\}}$ $D_y = P + P_{xy} + D_x$

TABLE IVI C.
Solving Equation for Triple disk failures

Triple disk failures	
Error location	Solving Equation
P & Q & R	$P = \sum_{i=0}^{n-1} g_0^i \cdot D_i$ $Q = \sum_{i=0}^{n-1} g_1^i \cdot D_i$ $R = \sum_{i=0}^{n-1} g_2^i \cdot D_i$
D_x & P & Q	$D_x = \frac{R+R_x}{g_2^x}, R_x = \sum_{i=0,i\neq x}^{n-1} g_2^i \cdot D_i$ $P = \sum_{i=0}^{n-1} g_0^i \cdot D_i$ $Q = \sum_{i=0}^{n-1} g_1^i \cdot D_i$
D_x & P & R	$D_x = \frac{Q+Q_x}{g_1^x}, Q_x = \sum_{i=0,i\neq x}^{n-1} g_1^i \cdot D_i$ $P = \sum_{i=0}^{n-1} g_0^i \cdot D_i$ $R = \sum_{i=0}^{n-1} g_2^i \cdot D_i$
D_x & Q & R	$D_x = \frac{P+P_x}{g_0^x}, P_x = \sum_{i=0,i\neq x}^{n-1} g_0^i \cdot D_i$ $Q = \sum_{i=0}^{n-1} g_1^i \cdot D_i$ $R = \sum_{i=0}^{n-1} g_2^i \cdot D_i$
D_x & D_y & P	$D_x = \frac{g_2^y \cdot (Q+Q_{xy}) + g_1^y \cdot (R+R_{xy})}{(g_1^y \cdot g_2^x + g_1^x \cdot g_2^y)}$ $D_y = \frac{(Q+Q_{xy})+D_x}{g_1^x}$ $P = \sum_{i=0}^{n-1} g_0^i \cdot D_i$
D_x & D_y & Q	$D_x = \frac{g_2^y \cdot (P+P_{xy}) + g_0^y \cdot (R+R_{xy})}{(g_2^y \cdot g_0^x + g_2^x \cdot g_0^y)}$ $D_y = \frac{(P+P_{xy})+D_x}{g_0^x}$ $Q = \sum_{i=0}^{n-1} g_1^i \cdot D_i$
D_x & D_y & R	$D_x = \frac{g_1^y \cdot (P+P_{xy}) + g_0^y \cdot (Q+Q_{xy})}{(g_1^y \cdot g_0^x + g_1^x \cdot g_0^y)}$ $D_y = \frac{(P+P_{xy})+D_x}{g_0^x}$ $R = \sum_{i=0}^{n-1} g_2^i \cdot D_i$
D_x & D_y & D_z	$D_z =$ $\frac{\begin{pmatrix} (g_1^y+g_1^x g_0^{y-x})[(R+R_{xyz})+g_2^x g_0^{-x}(P+P_{xyz})] \\ + \\ (g_2^y+g_2^x g_0^{y-x})[(Q+Q_{xyz})+g_1^x g_0^{-x}(P+P_{xyz})] \end{pmatrix}}{\begin{pmatrix} (g_2^z+g_2^x g_0^{z-x})(g_1^y+g_1^x g_0^{y-x}) \\ + \\ (g_2^y+g_2^x g_0^{y-x})(g_1^z+g_1^x g_0^{z-x}) \end{pmatrix}}$ $D_y = \left[\frac{(Q+Q_{xyz})+g_1^x g_0^{-x}(P+P_{xyz})}{(g_1^y+g_1^x g_0^{y-x})}\right] +$ $D_z \left[\frac{g_1^z+g_1^x g_0^{z-x}}{g_1^y+g_1^x g_0^{y-x}}\right]$ $D_x = g_0^{-x}(P + P_{xyz}) + g_0^{z-x}(D_z) +$ $g_0^{y-x}(D_y)$

IV. SYSTEM DESIGN

A. The environment

Before system design, one should consider the requirements of building basic components, construct the modules, doing integration, algorithm proof and system analysis. Based on these requirement as well as the considerations of efficient development and system reliability, the developing environment should be considered in the first stage. From the experiences, a concept of developing and analyzing platform using hardware/software co-design is applied to this project. This concept will use all the facilities which are easy/ready to get such FPGA, communication interfaces and PC.

B. Steps of Development

- Man/machine interface: One should provide useful data or information on the monitoring on PC. The information should include test and functional data, commands, error injection data for supporting algorithm proof.

- Basic components/modules: Developing the computation units of finite field arithmetic and the encoder/decoder of RAID 7.3 by using VHDL or C.
- The platform: Converting the simulation program into test/integration units
- Communication: The design of the transfer protocol for the RS232/USB.

C. *Architecture design of System platform*

In the stage of initial algorithm proof, one is normally doing simulation on PC using C, including the program and testing files. For the hardware design, such simulation program should be reused to help the developing of components or modules in testing and integration. These software will be a part of the platform for the purposes of the supports of converting into modules and comparison tools in testing. In FPGA, the major parts, from component to system modules, are constructed gradually in the developing platform. Afterwards, in system view, the major modules and functions are integrated into the system through the communication link on PC. The PC will provide better visualization of dynamic operations using JTAG or in-house link to support the software development and system verification efficiently. The architecture of this platform is shown in Fig. 5.

D. *Communication link*

The communication link of the platform may use JTAG in FPGA, but this project uses extra link by RS232/USB interface for mass data transfer. This should have programing on both NIOS and PC sides to support the hardware test, functional examination, and system verification.

V. CONCLUSIONS

Considering a system performance on speed, fault tolerance, flexibility and scalability, the RAID 7.3 is better than the current design of RAID. Firstly, this paper proposes a triple error fault-tolerant RAID with the decoding on rows for fast encode/decode. In the comparison of decoding speed with others, the proposed design uses blocks of data from received data buffer on rows access only so that the decoder can use burst mode in faster access speed than the other designs [12]. This brings this RAID system having advantage on access speed and system flexibility.

The proposed RAID 7.3 using the math modules of finite field to decode the errors/erasures, as refer to Table II, is assumed to use a hardware computing unit to speed up the decoding function. This will not just increase the speed of decoder but also decrees the power consumption of the system. In this project, the method to efficient developing the decoder is our important goal. To achieve this goal, several developing methodologies are applied, such as embedded system design, software and hardware co-design, and the use of developing platform.

On this platform, the FPGA present as the core of the developing system with great usages, such as testbed of developing components/modules, processor developer like NIOS, the verifier of the software, and as the center interface unit of the peripheral and computer systems. Finally, this project uses FPGA, communication link, and PC as a test and developing platform to design and develop RAID 7.3 system efficiently and successfully. We think this system is not just having a new product, but also bring the higher reliability to the future storage world.

The FPGA provides students with high design flexible environment and speed up hardware development. In the developing of system development platform, the developer has experienced the skills from the building of operational core, hardware and software interfaces, the OS driving interface, transport protocol, system refinement, and the entire system integration. Also this project provides student valuable experiences to enjoy the competition and learn mutual cooperation in teamwork.

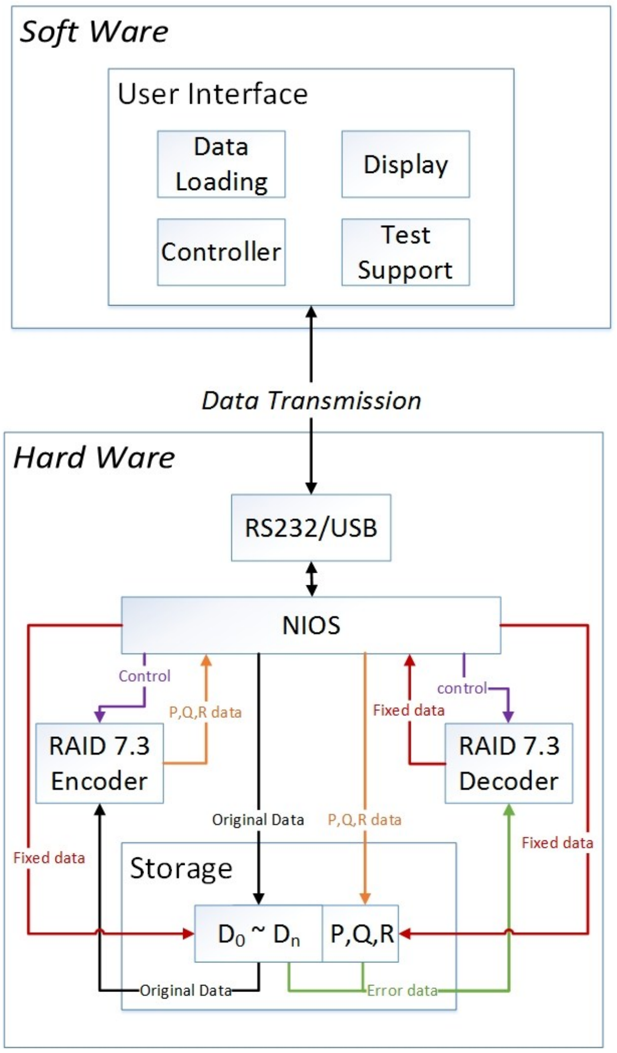

Fig. 5 Architecture of System platform

ACKNOWLEDGMENT

This project is support by the NSC101-2221-E-214-058, NSC102-2221-E-214-028,102-2815-C-214-011-E and

ISU101-1-10 from National Science Council and I-Shou University.

REFERENCES

[1] Minghaw Jing, Chong-Dao Lee, Yaotsu Chang, and Kai Su, “Tolerating Triple Disk Failures in Triple-Parity RAID”, ISNE, Kaohsiung, Taiwan, 2013

[2] H.P. Anvin. The mathematics of RAID-6. 2011. Available at http://kernel.org/pub/linux/kernel/ people/hpa/raid6.pdf

[3] 袁國元袁國元,“利用 RS-code 編解碼積體電路實現高可靠度磁碟陣列之研究”, 義守大學資訊工程研究所碩士論文, 中華民國八十九年六月

[4] Adam Leventhal, Sun Microsystems,“Triple-Parity RAID and Beyond ”, Acmqueue, File Systems and storage, Vol.7 No. 11, 2009,12,1

[5] Chao Jin, Dan Feng, Hong Jiang, Lei Tian, “A Comprehensive Study on RAID-6 Codes: Horizontal vs. Vertical”, 2011 Sixth IEEE International Conference on Networking, Architecture, and Storage

[6] Blaum M, Brady J, Bruck J. EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures. IEEE Transactions on Computers, 1995, 44(2):192-202.

[7] Corbett P, English B, Goel A. Row-Diagonal Parity for Double Disk Failure Correction. in: Proceedings of the 3rd USENIX Conference on File and Storage Technologies (FAST'04), San Francisco, CA, 2004, 1-14.

[8] Plank J S. The RAID-6 Liberation Codes. in: Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST'08), San Jose, CA, 2008, 97-110.

[9] Xu L, Bohossian J, Bruck J, et al. Low-density MDS codes and factors of complete graphs. IEEE Transactions on Information Theory, 1999, 45(6):1817-1826.

[10] Xu L, Bruck J. X-Code: MDS array codes with optimal encoding. IEEE Transactions on Information Theory, 1999, 45(1):272-276.

[11] Jin C, Jiang H, Feng D, et al. P-Code: A New RAID-6 Code with Optimal Properties. in: Proceedings of the 23rd ACM International Conference on Supercomputing (ICS'09), New York, NY, 2009, 360-369.

[12] 陶志行,“磁碟陣列系統之容錯演算”, 國立成功大學工程研究所博士論文, 95 年 1 月 19 日