Sensors and Materials, Vol. 35, No. 5 (2023) 1519–1531
MYU Tokyo

S & M 3265

Efficient Reed–Solomon Redundant Arrays of Independent Disks 6 Exclusive OR Accelerator with Scaling Scheme

Yaping Yue,[1,2] Ruizhen Wu,[1,2*] Ronghui Hou,[1] Yan Wu,[2] and Lin Wang[2]

[1]School of Cyber Engineering, Xidian University, Xi'an 710071, China
[2]United Micro Technology (Xi'an) Co. Ltd., Xi'an 710076, China

(Received June 28, 2022; accepted December 2, 2022)

Keywords: RAID, RAID 6, RS-RAID 6, codec

An efficient Reed–Solomon redundant arrays of independent disks 6 (RS-RAID 6) exclusive OR (XOR) accelerator with a scaling scheme is proposed in this paper. The accelerator first considers the hardware working characteristic to reduce the complexity of codec computation. To increase efficiency, the parameter calculation of the 8-bit Galois field [GF(8)] is converted to a GF(2) calculation, where two additions, one multiplication, and one division are optimized in every 8 bits of calculation. A new scaling scheme is also proposed to guarantee the input/output (I/O) balance. The proposed Reed–Solomon (RS) codec hardware architecture is fully verified at the Real Time Logistics (RTL) level and synthesized with a GF 12 nm technology library to compare the gate cost and input/output operations per second (IOPS) performance with those of other classic solutions. Separate simulations are performed for different codec algorithms and scaling schemes. The IOPS performance of the proposed RS-RAID 6 XOR accelerator is improved by a factor of 2–3 at the cost of 47% more gates.

1. Introduction

Redundant arrays of independent disks (RAID) technology is widely adopted and almost ubiquitous in modern storage systems, especially in large disk arrays, because of its ability to exploit input/output (I/O) parallelism and provide data protection. RAID systems are classified from levels 0 to 6, which are different levels having different advantages. RAID 6 has become increasingly popular in modern storage applications, especially in large-scale storage systems (e.g., cloud storage systems) that consist of many less reliable (but more economical) disks such as Serial Advanced Technology Attachment (SATA) (vs Small Computer System Interface, SCSI). It can provide a large I/O bandwidth via parallel I/O operations and tolerate two failures by maintaining dual parity.[1–3]

Many different RAID 6 systems have been proposed, each with different advantages. The most popular codes used are EVENODD[1–3] and Row-Diagonal Parity (RDP) codes.[4–6] However, none of these representative codes has emerged as a clear overall winner as each code has its own limitations. For example, EVENODD and RDP both have to update three (on

*Corresponding author: e-mail: wuruizhen_1985@163.com
https://doi.org/10.18494/SAM4002

ISSN 0914-4935

average) parity blocks whenever a data block is changed; this number is 1.5 times the theoretical lower bound.[7–10]

RAID systems have received increasing attention with the ever increasing probability of multiple disk failures.[11,12] RAID-based architectures are also used in clusters and large-scale storage systems. Because of the ever increasing demand for storage capabilities, applications often require a large storage capacity and high performance. This is normally achieved by adding new disks to existing RAID systems;[13–16] this is known as RAID scaling.

To regain load balance after RAID 6 scaling, data must be redistributed evenly among all disks whether old or new. In current server environments, the cost of downtime is extremely high.[17,18] Therefore, RAID 6 scaling requires an efficient approach to redistributing the data online that meets the following requirements. (1) Data redistribution should be completed in a short time, which means that as little data as possible should be redistributed. (2) The impact of data redistribution on application performance should be low. An efficient approach to Reed–Solomon (RS)-RAID 6 scaling based on the characteristics of its coding method should be designed. Thus, in this paper, we first propose an efficient RAID scaling scheme for RS-RAID 6.

Regarding the codec part, the RS code is a popular error control code that has been studied in various applications, especially communication systems. We propose an algorithm based on the RS code for implementation in an advanced RAID system. The proposed algorithm can meet the current needs for RAID 6 coding, while providing many other advantages. First, the number of data drives and the fault tolerance capability are flexible. Second, all computations can be completed by XOR. Furthermore, the proposed algorithm is implemented using about 47% more gates than in EVENODD and RDP RAID 6, but the IOPS performance is improved by a factor of 2–3.

The rest of this paper is organized as follows. In Sect. 2, we first present the round-robin (RR) and semi-round-robin (semi-RR) scaling schemes followed by their related codec. The details of the proposed scaling scheme and the new proposed prototype RAID 6 and its algorithms are introduced in Sect. 3. The experimental evaluation and results are shown in Sect. 4, and conclusions are given in Sect. 5.

2. Typical RAID 6 System

Broadly, the term RAID 6 represents any form of RAID that can tolerate any two concurrent disk failures.[1] Nevertheless, the de facto standard form of RAID 6 has been the so-called P + Q redundancy because of its compatibility with the widely deployed RAID 5.[19] There are two keys to constructing a RAID 6 system, the codec and the scaling scheme, both of which affect the performance of the system. RR and semi-RR are two widely used RAID scaling schemes with EVENODD or RDP used as the codec.

2.1 Scaling scheme

Typical RAID scaling approaches preserve the RR and semi-RR data distributions after adding disks.[18–21] Although RR and semi-RR are simple approaches to implementation on RAID 6, the resulting overhead is high.

RR scaling can achieve an I/O load balance regardless of the number of added disks. We take a four-disk RAID 6 system as an example. There are two data and P + Q parities of each stripe in the system. When two new disks are added, the RR scaling saves the first stripe's data and redistributes all the other data, as shown in Fig. 1.

From the left scheme in Fig. 1, we find that both the D and P + Q data are evenly balanced (all disks have almost the same number of P + Q data) among all disks. When disks 5 and 6 are added, the first stripe remains the same (grey area on the right scheme in Fig. 1), but all other data must migrate with the RR sealing scheme to fill the newly added disks.

The semi-RR scaling scheme is proposed to decrease the high migration cost in RR scaling. Semi-RR scaling only moves part of the data, and the migration overhead is less than that of the RR scaling scheme. Unfortunately, semi-RR scaling also has the flaw that it does not ensure a uniform data distribution on all disks after multiple scaling. The data migration process of semi-RR scaling based on RAID 6 corresponding to Fig. 1 is shown in Fig. 2.

The data migration in semi-RR scaling affects only half the data of the original RAID system. It can be clearly seen from Fig. 2 that the number of data that must be redistributed is reduced, but the I/O balance cannot be easily achieved. With the growing need for RAID systems, an efficient scaling scheme focusing on both the I/O balance and the migration cost is important.

2.2 RAID 6 codec

EVENODD and RDP are the most used RAID 6 codec algorithms in industry because of their straightforward calculation in both encoding and decoding.

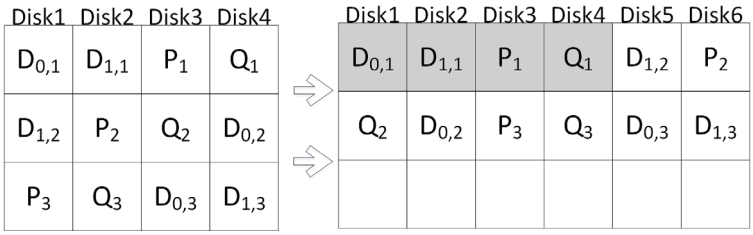

Fig. 1. RR scaling of RAID 6 from four to six disks.

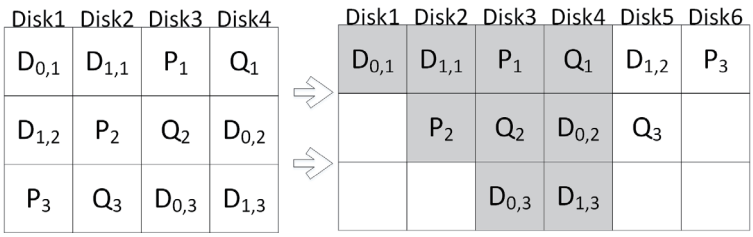

Fig. 2. Semi-RR scaling of RAID 6 from four to six disks.

EVENODD is one of the most widely studied RAID 6 codes. EVENODD encoding considers a $(p - 1) \times (p + 2)$ 2D array with prime number $p > 2$. Columns p and $p + 1$ are two parity columns, whereas the other p columns are information (data under codec) columns.

The two parity columns p and $p + 1$ are also named the row and diagonal parity columns in EVENODD encoding. The row parity symbol $d_{i,p}$ in column p is computed as the XOR sum of all information symbols in row i. The XOR sum of symbols along diagonal $p - 1$ is computed as

$$h = \oplus_{j=1}^{p-1} d_{p-1-j,j}, \tag{1}$$

where h is called the EVENODD adjuster, which is used to generate each parity symbol in the diagonal parity column $p + 1$. Figure 3 shows a particular EVENODD code in a stripe of $p = 5$.[1]

RDP is one of the most important RAID 6 codes and achieves optimality in both computation efficiency and I/O efficiency.[2] RDP encoding takes a $(p - 1) \times (p + 1)$ 2D array (or stripe) with prime number $p > 2$. The first $p - 1$ columns in the array are information columns and the last two columns are parity columns.

The two parity columns in the array, named the row and diagonal parity columns, ensure that all pieces of information are recoverable when there are no more than two disk failures. A row parity symbol is generated by XOR summing all the information symbols in that row, and a diagonal parity symbol is generated by XOR summing all symbols along the same diagonal. Figure 4 shows the construction mechanism of the RDP code when $p = 5$, where $d_{i,j}$ is the i-th symbol in column j.

In a RAID 6 system, when one or two disks fail, to reconstruct the failed data, the corresponding information and parity data should be read from surviving disks to calculate the related failed stripes as soon as possible. Thus, a good RAID 6 algorithm has a simple codec operation and the same complexity for different combinations of failed disks.

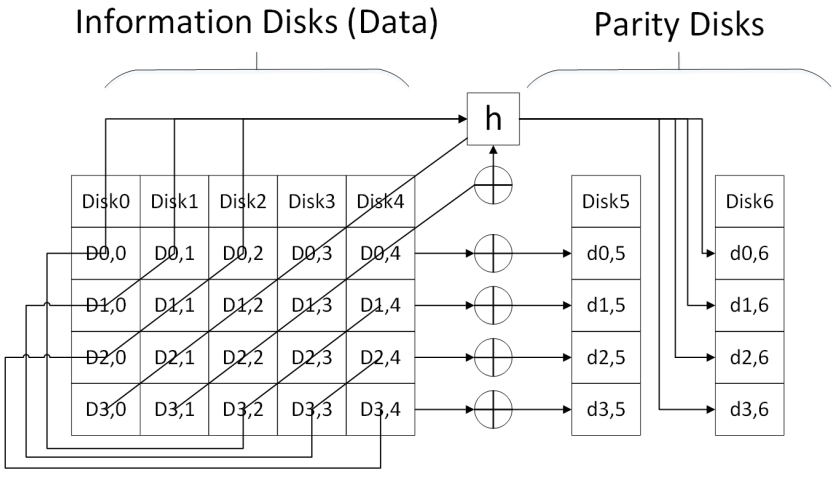

Fig. 3. EVENODD coding in a stripe of $p = 5$.

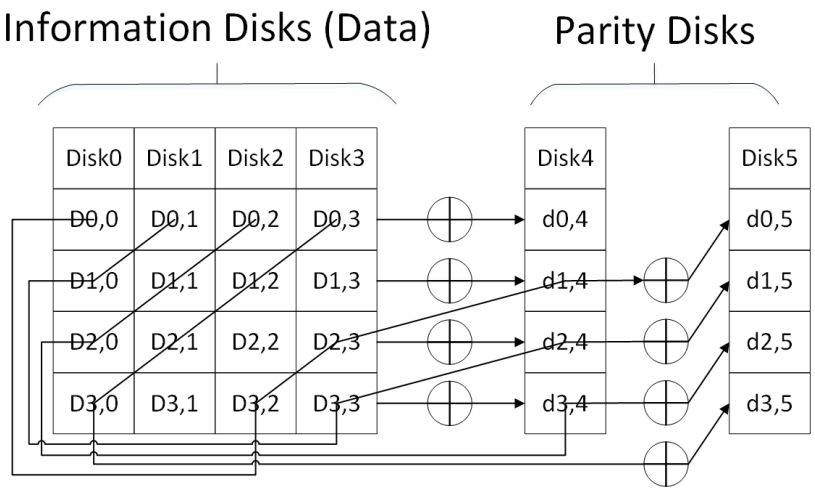

Fig. 4. RDP coding in a stripe of $p = 4$.

3. Proposed RAID 6 XOR Accelerator with Scaling Scheme

The work in this paper has two parts: scaling followed by the XOR operation for parity updates. We select an RS codec algorithm for use with RAID 6, which we name RS-RAID 6. An efficient RAID scaling scheme based on the characteristics of its coding method should be designed. Here, we briefly introduce the RS-RAID 6 coding method.

The RAID 6 construction must consider the RAID level migration, which means that when one disk fails, we can still use RAID 5 for any disk's failure. In RAID 6, we have two sets of parity, P and Q, which can be on any two disks. These two sets of parity are calculated from two different independent equations; the equations must be independent so that any two elements in the equations can always be solved in terms of the other elements. To meet these requirements, from among the RS codes, we select the Vandermonde matrix to determine the coefficients. The two equations selected are

$$\begin{cases} d_0 \oplus d_1 \oplus \ldots \oplus d_n \oplus p \oplus q = 0, \\ \alpha^0 d_0 \oplus \alpha^1 d_1 \oplus \ldots \oplus \alpha^n d_n \oplus \alpha^{n+1} p \oplus \alpha^{n+2} q = 0. \end{cases} \tag{2}$$

The first equation is the same as that in RAID 5, which ensures recovery when only one disk fails. When two disks fail, both equations must be used in the recovery.

A RAID system normally has three working scenarios: 1. encoding to obtain the parity data; 2. updating of parity data with part of the data changed; 3. decoding one or two errors. These three scenarios in the proposed RS-RAID 6 are next introduced.

3.1 Encoding to obtain parity data

RS-RAID 6 introduces parameters that fit the RS code (Vandermonde matrix here) to obtain the parity data p and q. As shown by Eq. (2), p and q are placed in the final two positions, but

they can be placed anywhere with different parameters. For the easy calculation of p and q, we place them at the first two positions as follows:

$$\begin{aligned} &\alpha^0 p \oplus \alpha^0 q \oplus \alpha^0 d_0 \oplus \alpha^0 d_1 \oplus ... \oplus \alpha^0 d_{n-2} = 0, \\ &\alpha^0 p \oplus \alpha^1 q \oplus \alpha^2 d_0 \oplus \alpha^3 d_1 \oplus ... \oplus \alpha^n d_{n-2} = 0. \end{aligned} \tag{3}$$

Then, we can obtain equations for p and q:

$$\begin{aligned} p &= \frac{\alpha^1 \square \alpha^2}{\alpha^0 \oplus \alpha^1} d_0 \oplus \frac{\alpha^1 \quad \alpha^3}{\alpha^0 \oplus \alpha^1} d_1 \oplus ... \oplus \frac{\alpha^1 \quad \alpha^n}{\alpha^0 \oplus \alpha^1} d_{n-2}, \\ q &= \frac{\alpha^0 \oplus \alpha^2}{\alpha^0 \oplus \alpha^1} d_0 \oplus \frac{\alpha^0 \oplus \alpha^3}{\alpha^0 \oplus \alpha^1} d_1 \oplus ... \oplus \frac{\alpha^0 \oplus \alpha^n}{\alpha^0 \oplus \alpha^1} d_{n-2}. \end{aligned} \tag{4}$$

The calculation of p and q depends on the user data d_n and its disk location.

3.2 Updating of parity data

In practice, users sometimes need to update one or several pieces of information but not rewrite all parities. RAID systems require an efficient way to update data. The RS-RAID 6 system can meet this requirement. From the upper equation of Eq. (4), we obtain

$$\begin{cases} a^x d_x \oplus a^p p \oplus a^q q = a^x d'_x \oplus a^p p' \oplus a^q q', \\ d_x \oplus p \oplus q = d'_x \oplus p' \oplus q', \end{cases} \tag{5}$$

where dx is an original datum currently in the RAID system and dx' is its updated form. We set $d_x \oplus d'_x = \Delta$. Then, from the upper equation of Eq. (5), we derive

$$q' = q + \frac{(\alpha^p + \alpha^x)}{(\alpha^p + \alpha^q)} \Delta . \tag{6}$$

We derive a similar result for p'.

3.3 Decoding

When there is only one error, the following decoding is used in the RAID 5 calculation:

$$d_n = d_0 \oplus ... \oplus d_{n-1} \oplus ... d_{n+1} \oplus ... \oplus p \oplus q . \tag{7}$$

For two errors, the same operation of Eq. (4) is used, but "p,q" is changed to "d_m,d_n" and its related parameters:

$$\begin{aligned} d_m &= \frac{\alpha^{n-2} \oplus \alpha^0}{\alpha^{n-2} \oplus \alpha^{m-2}} p \oplus ... \oplus \frac{\alpha^{n-2} \oplus \alpha^n}{\alpha^{n-2} \oplus \alpha^{m-2}} d_{n-2}, \\ d_n &= \frac{\alpha^{m-2} \oplus \alpha^0}{\alpha^{n-2} \oplus \alpha^{m-2}} p \oplus ... \oplus \frac{\alpha^{m-2} \oplus \alpha^n}{\alpha^{n-2} \oplus \alpha^{m-2}} d_{n-2}, \end{aligned} \tag{8}$$

where m and n are the positions of error data. For different errors, only the error multiplication in Eq. (8) needs to be replaced.

To implement our proposed RS-RAID 6, we must first determine the data format. An XOR hardware accelerator can improve the operation speed of software. Thus, the data format is set to 8 bits for each input for easy hardware implementation. However, the data in RS encoding requires many multiplications, which require more than 8 bits. Thus, we require a set of numbers that follow the rules of mathematics while remaining within 8 bits. The solution is to use a finite field based on RS encoding that is generated by a primitive polynomial. Here, we use the Galois field GF(8) for our number set. This field and the operators we define on it are used in all equations specific to our RS-RAID 6 system.

The primitive polynomial we choose is

$$GF(8) = \alpha^8 + \alpha^4 + \alpha^3 + \alpha^2 + 1 . \tag{9}$$

Using the polynomial equation, our field here is made up of powers from α^0 to α^{254}, where α^0 is used to represent the hexadecimal numbers x01 and α^{254} in the xFF. The calculations are all based on Eq. (9) in the Galois field.

Given two 8-bit operands A and B, we assume that each bit position represents the order of a term in a polynomial. For example, a high-order bit of A is A_7. Then, A can be expressed as the sum of products:

$$A_7\alpha^7 + A_6\alpha^6 + ... + A_1\alpha^1 + A_0\alpha^0 . \tag{10}$$

The multiplication of $A * B$ can be expressed as

$$\begin{aligned} A * B &= (A_7\alpha^7 + ... + A_0\alpha^0) * (B_7\alpha^7 + ... + B_0\alpha^0) \\ &= (A_7B_7\alpha^{14}) + (A_6B_7 + A_7B_6)\alpha^{13} + ... + A_0B_0\alpha^0 . \end{aligned} \tag{11}$$

The powers of α^{14} and α^{13} in Eq. (11) are larger than 7. Thus, we need to perform polynomial division with Eq. (9) to obtain its expression with $\alpha^0 \sim \alpha^7$. Taking α^8 as an example, we can obtain $\alpha^8 = 29 = \alpha^4 + \alpha^3 + \alpha^2 + \alpha^1$.

As defined in Eq. (5), we set the user data after scaling as d'_x and its original d_x, Δ as $d_x \oplus d'_x = \Delta$. The result of Δ multiplied by 1 is set as Δ_d, and that multiplied by α^x is set as Δ'_d. Then, we can obtain

$$f(\alpha)=\begin{cases} p\oplus q\oplus \Delta_d, \\ p\oplus \alpha q\oplus \Delta'_d. \end{cases} \tag{12}$$

Replace α in Eq. (12) with x and approximate to obtain $f(x)$ using Eq. (12). The primitive polynomial of GF(2) is obtained using $f(x)$; then, GF(8) in Eq. (9) is used to obtain $p(x)$.

$$\begin{aligned} f(x) &\approx c_1 + c_2 x \\ p(x) &= x^2 + (1+\alpha)x + \alpha \end{aligned} \tag{13}$$

Use $f(x)$ in Eq. (13) to replace all the parameter calculations in GF(8). The RS-RAID 6 calculation will be simplified to c_1 and c_2 multiplication.

Taking d_0 in Eq. (8) as an example, the parameter for d_0 is α^2. Then, use $p(x)$ from Eq. (13) to perform the derivation as follows.

$$\begin{array}{r} 1 \\ x^2+(\alpha+1)+\alpha \overline{)\, x^2} \\ x^2+(\alpha+1)x+\alpha \\ \text{----------} \\ (\alpha+1)x+\alpha \end{array} \tag{14}$$

From Eq. (14), we can derive the c_1 and c_2 relationship as $c_1 = \alpha$, $c_2 = \alpha + 1$. Then, c_2 can be transformed to α^{25}. By the same derivation, we can calculate each dn in GF(2) for GF(8).

To verify the correctness of d_0, c_1 is taken as α and c_2 as α^{25}. Then, p_0, the parameter of d_0, is α in Eq. (8) and can be derived as

$$p_0 = \frac{\alpha^1 \oplus \alpha^2}{\alpha^0 \oplus \alpha^1}, \tag{15}$$

$$\frac{\alpha^1 \oplus \alpha^2}{\alpha^0 \oplus \alpha^1} = \frac{\alpha^{26}}{\alpha^{25}} = \alpha. \tag{16}$$

Performing the same calculation for q, we can obtain the result of q with c_1, c_2 *as* α^{25}.

As described previously, decoding in RS-RAID 6 in fact has the same mathematical relationship as encoding but with a different unknown position.

From Table 1, whether coding for p and q or any two data errors, we can obtain different tables by changing the p and q locations. The calculation can be performed as a search of the parameters in the table to multiply with data, followed by XOR on all results. With the proposed algorithm, all codecs can be simplified to a multiplication; this saves a considerable amount of the calculation cost of the RS-RAID 6 system.

Table 1
c_1 and c_2 in GF(2) for GF(8).

d position	$m(x)$	c_1	c_2
d_0	x^2	α	α^{25}
d_1	x^3	α^{26}	α^{198}
d_2	x^4	α^{199}	α^{75}
d_3	x^5	α^{76}	α^{100}
⋮	⋮	⋮	⋮

The proposed hardware is shown in Fig. 5. Figure 5 shows an example of the updating mode of the RS-RAID 6 system. The new information data is d_{n-2} and the data being updated are Data (0) and Data (1), which are the parity data p and q. The operation is as follows, as marked in Fig. 5.

1. When d_{n-2} arrives, it is temporarily stored in a buffer or First In First Out (FIFO), for example, and then sent to its disk location—which is Data (n−2) here—with a suitable response. On the basis of the location information, the parameter C to be multiplied is read from the table and sent to the XOR module at the same time. Data to be updated are all exclusive OR operated with d_{n-2} at the XOR module.
2. The new data are used to update p and q [Data (0) and (1) here, respectively]. Then, the next input for the next update is awaited.

As shown in Fig. 5, the proposed RS-RAID 6 system does not require complicated hardware for the GF operation and only requires the parameter look-up table (Table 1). The multiplication is performed using different parameters. With the fixed p and q locations in Eq. (8), a different amount of input information data only affects the newly added parameters but not the old parameters. Therefore, for different working environments, the same parameter table can suit all requirements (the table is set to the maximum input number), which results in easy implementation and simplified operation.

By the above-described introduction of the RS-RAID 6 codec, we find that the coding mostly depends on the location of the data stripe (which is used to determine the parameter to multiply). The RR scaling shown in Fig. 1 has too many data migrations that change the location of the data stripe, so the proposed RS-RAID 6 system is severely complex.

The advantages of RS-RAID 6, which are also due to its coding, are only affected by data in the same stripe. Hence, for the semi-RR shown in Fig. 2, the redistribution of P and Q is disadvantageous for our RS-RAID 6.

Existing scaling schemes are insufficient to achieve the advantages of RS-RAID 6 because of too much migration and stripe structure destruction. This motivates us to propose a new scaling scheme for the proposed RS-RAID 6 system.

RS-RAID 6 encoding only relies on the data in the stripe of each row, but no column data are involved. Therefore, the intention of the proposed RS-RAID 6 scaling scheme first is only to migrate the data in the row. To construct an efficient scaling scheme, we must also consider the location relationship of RS-RAID 6, that is, the fewer changes of the original locations of data the better. With these two purposes, the proposed RS-RAID 6 scaling scheme can be described as below.

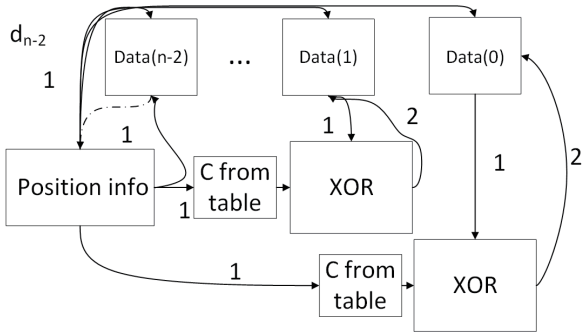

Fig. 5. Schematic of hardware of proposed RS-RAID 6 in updating mode.

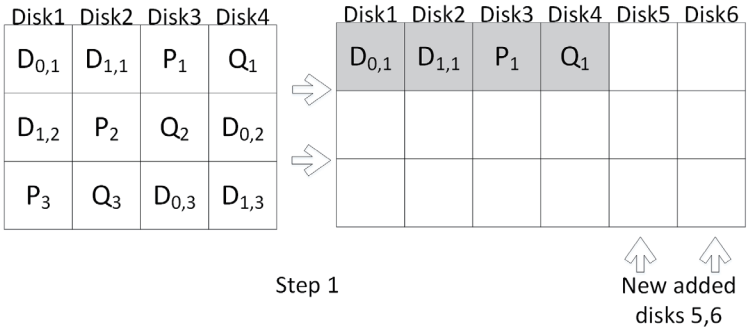

Fig. 6. Step 1 of proposed RS-RAID 6 scaling scheme.

1. Add new disks into the original RAID system and retain all the data and parity of the first stripe. Keep the empty spaces of the first stripe of the added disks. Refer to the examples in Figs. 1 and 2 for an understanding of Fig. 6.
2. The second row's data need to move to the new added disks. There are two new disks so we pick Q_2 and $D_{0,2}$ (which are both next to the new disks) to move to the new disks. The data migration only involves the location change of data described in Fig. 7. As shown in Fig. 7, the circle indicates the data location. Compare the data in the circles with those in the original RAID system and transfer Q_2 and $D_{0,2}$ to new locations. The grey parts in the figure indicate the data that do not require redistribution.
3. Repeat step 2 until all stripes are redistributed. The final scaling scheme is shown in Fig. 8. The I/O load here is composed of two data values on each disk, which realizes load balance. Less data than in the original RAID system must be redistributed. With the proposed scheme, the RS-RAID coding relationship in each stripe is the same as in the original RAID system.

4. Simulation and Results

The proposed RS codec hardware architecture is fully verified at the Real Time Logistics (RTL) level and synthesized with the GF 12 nm technology library. During the hardware synthesis, time delays have been considered by restricting the longest path delay of the codec. As

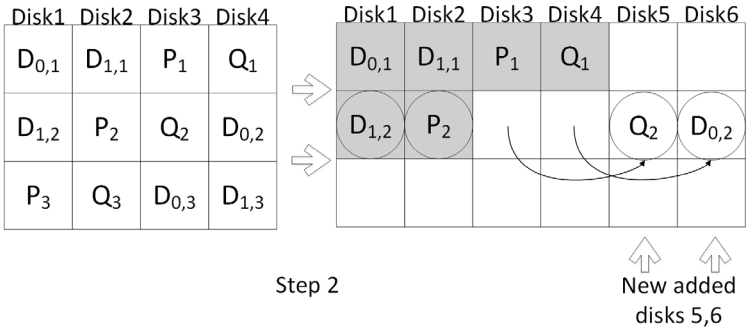

Fig. 7. Step 2 of proposed RS-RAID 6 scaling scheme.

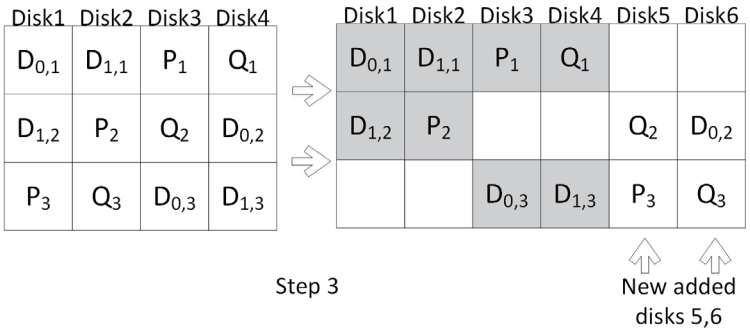

Fig. 8. Step 3 of proposed RS-RAID 6 scaling scheme.

seen in Fig. 9, our approach indeed needs fewer XOR operators than the conventional RS codes, but more than RDP and EVENODD. The RS code proposed in this paper can process data in parallel. This is because the encoding process of the proposed RS code can calculate the effects of each data drive on the respective checksum drive.

The total number of gates is determined using the synthesized results excluding the FIFO memories. The new proposed architecture was implemented with about 21000 gates when the information data number was 29, which means that the RAID 6 system needs a total of 32 disks including two parity disks. The number of gates is reduced by about 62.3% compared with that in the case of RS-RAID 6 without the proposed improvement method. However, this is still about 47% more gates than in EVENODD and RDP because RS-RAID needs to use RS to perform complex codec calculations. The FIFO memory sizes have been reduced since the latency has been reduced. We have verified the correctness of the designed RS codec by extensive simulation with randomly generated data and errors.

The IOPS performance characteristics of the proposed codec, RS-RAID 6, EVENODD, and RDP are compared using the INSPUR_SHUYU server, which has CPU_I_E5-2650v3-Xeon2.3_9.6G_ 25M_10C and Samsung 6G_DDR4_2133ER_2R4_D36F for the 8 × 2.5 inch 12G SAS HDD or an 8 × 3.5 inch 6G SATA SDD. The read or write set is 256 kB for each stripe. The results are shown in Fig. 10.

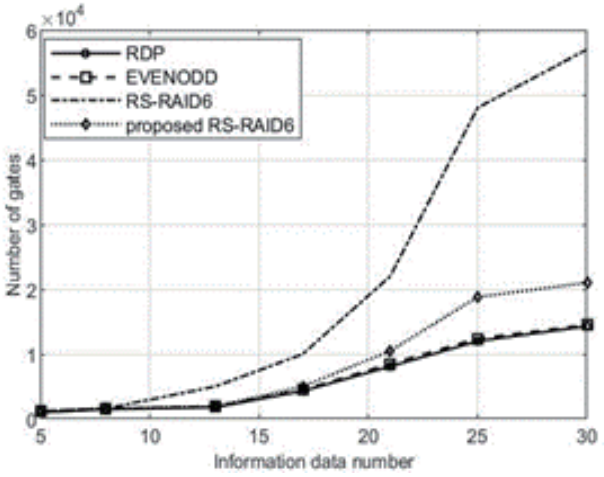

Fig. 9. Number of gates used in proposed codec, EVENODD, RDP, and RS-RAID 6.

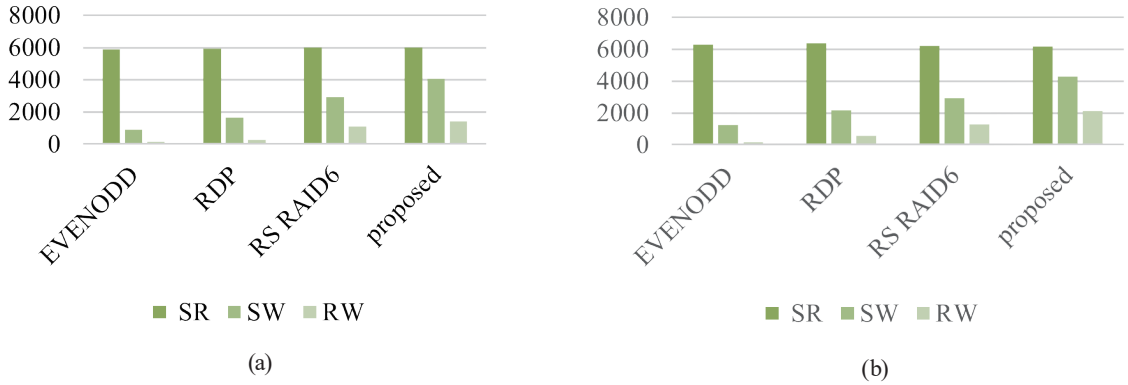

Fig. 10. (Color online) Comparison of IOPS results. (a) IOPS comparison with 8*2.5 inch 12G SAS HDD. (b) IOPS comparison with 8*2.5 inch 6G SATA SDD.

SR means stripe read, SW stripe write, and RW read/write with 50% of the bandwidth for stripe read and 50% for stripe write. IOPS results are slightly a higher for SDD than for HDD. The general tendency is that RS-RAID 6 system has better IOPS performance than EVENODD and RDP. The proposed RS-RAID 6 with the improvement algorithm has more IOPS than the conventional RS-RAID 6. The IOPS result of the proposed RS-RAID system 6 is 2–3 times higher than those of the EVENODD and RDP RAID 6 systems.

5. Conclusion

An efficient RS-RAID 6 XOR accelerator with a scaling scheme was proposed in this paper. First, parity codes were generated by the RS algorithm Vandermonde and optimized by adjusting the hardware working characteristics. The calculations of parameter generation were reduced from two GF additions and one GF division to only one GF multiplication. From the results of the simulation, we found that the proposed system cost 47% more gates to achieve an IOPS

improvement of 2–3 times those of the EVENODD and RDP RAID 6 systems. The scheme was improved by guaranteeing an IO balance to reduce the complexity of scaling redistribution and using exclusive OR operations in accordance with the characteristics of the RS-RAID 6 coding method. Finally, a comparison of the simulation results for three classic working scenarios showed that the proposed scheme has a higher performance. Increasing the number of added disks reduced the advantages of the proposed system, but it was still better than RR and semi-RR.

References

1 L. Xiang, Y. Xu, J. C. S. Lui, Q. Chang, Y. Pan, and R. Li: ACM Trans. Storage **7** (2011) Article 11. http://doi.org/10.1145/2027066.2027071
2 Y. Lan, H. Hou, and P. Zhang: 2020 IEEE 20th Int. Conf. Communication Technology (IEEE, 2020) 447–453.
3 H. Hou, Y. S. Han, K. W. Shum, and L. Hui: IEEE Trans. Commun. **66** (2018) 5053. https://doi.org/10.1109/TCOMM.2018.2859956
4 B. Li, Y. Meng, S. Mohajer, W. Qian, and D. J. Lilja: 2018 IEEE Int. Conf. Networking, Architecture and Storage (NAS) (IEEE, 2018) 1–10.
5 G. Zhang, K. Li, J. Wang, and W. Zheng: IEEE Trans. Comput. **64** (2014) 32. http://doi.org/10.1109/TC.2013.210
6 X. Chen and X. Ma: IEEE Commun. Lett. **22** (2018) 2443. http://doi.org/10.1109/LCOMM.2018.2875468
7 Z. Huang, H. Jiang, Z. Shen, H. Che, and N. Li: 2020 IEEE Int. Parallel and Distributed Processing Symposium (IPDPS) (IEEE, 2020) 708–717.
8 Z. Qiao, S. Liang, S. Fu, H. B. Chen, and B. Settlemyer: 2019 49th Annu. IEEE/IFIP Int. Conf. Dependable Systems and Networks – Industry Track (IEEE, 2019) 17–20.
9 L. Han, S. Tan, B. Xiao, C. Ma, and Z. Shao: 2019 IEEE Non-Volatile Memory Systems and Applications Symp. (NVMSA) (IEEE, 2019) 1–6.
10 Y. Hu, P. Xie, M. Y. Li, D. Wang, and S. Geng: 2020 IEEE 3rd Int. Conf. Information Communication and Signal Processing (ICICSP) (IEEE, 2020) 481–485.
11 E. Pinheiro, W. D. Weber, and L. A. Barroso: File and Storage Technologies (ACM, 2007) 17–29.
12 B. Schroeder and G. A. Gibson: 5th USENIX Conf. File Storage Technol (ACM, 2006) 1–16.
13 G. Zhang, G. Wu, L. Yu, W. Jie, and W. Zheng: IEEE Trans. Parallel and Distrib. Syst. **27** (2016) 3687. http://doi.org/10.1109/tpds.2016.2542806
14 Y. Saito, S. Frolund, A. Veitch, A. Merchant, and S. Spence: Acm Sigarch Comput. Archit. News **38** (2004) 48. http://doi.org/10.1145/1037949.1024400
15 Y. Xiang: USENIX Association (ACM, 2000) 17–32.
16 S. Ghandeharizadeh and D. Kim: Int. Conf. Database and Expert Systems Applications (ACM, 1996) 751–768.
17 D. A. Patterson: Large Installation System Administration (ACM, 2002) 185–188.
18 P. Jin, P. Xie, Z. Yuan, Y. Hu, Y. Gao, and J. Ma: 2019 IEEE 5th Int. Conf. Computer and Communications (ICCC) (IEEE, 2020) 545–549.
19 C. Tian, Y. Li, S. Wu, J. Chen, and Y. Xu: IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **39** (2019) 2911. http://doi.org/10.1109/TCAD.2019.2930580
20 X. Zhong, Z. Yuan, Y. Hu, and P. Xie: 2019 IEEE 2nd Int. Conf. Computer and Communication Engineering Technology (CCET) (IEEE, 2019) 105–108.
21 G. Zhang, W. Zheng, and K. Li: IEEE Trans. Comput. **63** (2014) 2816. http://doi.org/10.1109/TC.2013.143/