(54) Title: HIGH PERFORMANCE RAID-6 SYSTEM ARCHITECTURE WITH PATTERN MATCHING

(57) Abstract: An acceleration unit offloads computationally intensive tasks from a processor. The acceleration unit includes two data processing paths each having an Arithmetic Logical Unit and sharing a single multiplier unit. Each data processing path may perform configurable operations in parallel on the same data. Special multiplexer paths and instructions are provided to allow P and Q type syndromes to be computed on a stripe in a single-pass of the data through the acceleration unit.
Published:
— with international search report
— before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments
HIGH PERFORMANCE RAID-6 SYSTEM ARCHITECTURE WITH PATTERN MATCHING

FIELD

[0001] This disclosure relates to redundant array of independent disks (RAID) systems and in particular to acceleration of computations for a RAID-6 system.

BACKGROUND

[0002] A Redundant Array of Independent Disks (RAID) combines a plurality of physical hard disk drives into a logical drive for purposes of reliability, capacity, or performance. Thus, instead of multiple physical hard disk drives, an operating system sees the single logical drive. As is well known to those skilled in the art, there are many standard methods referred to as RAID levels for distributing data across the physical hard disk drives in a RAID system.

[0003] For example, in a level 0 RAID system the data is striped across a physical array of hard disk drives by breaking the data into blocks and writing each block to a separate hard disk drive. Input/Output (I/O) performance is improved by spreading the load across many hard disk drives. Although a level 0 RAID improves I/O performance, it does not provide redundancy because if one hard disk drive fails, all of the data is lost.

[0004] A level 5 RAID system provides a high level of redundancy by striping both data and parity information across at least three hard disk drives. Data striping is combined with distributed parity to provide a recovery path in case of failure. A level 6 RAID system provides an even higher level of redundancy than a level 5 RAID system by allowing recovery from double disk failures. In a level 6 RAID system, two syndromes referred to as the P syndrome and the Q syndrome are generated for the data and stored on hard disk drives in the RAID system. The P syndrome is generated by simply computing parity information for the data in a stripe (data blocks (strips), P syndrome block and Q syndrome block). The generation of the Q syndrome requires Galois Field multiplications and is complex in the event of a disk drive failure. The regeneration scheme to recover data and/or P and/or Q syndromes performed during disk recovery operations requires both Galois multiplication and inverse operations.
The regeneration is typically performed using lookup tables for computation or through the use of a plurality of Galois-field multipliers which are limited to a specific polynomial.

5 BRIEF DESCRIPTION OF THE DRAWINGS

[0006] Features of embodiments of the claimed subject matter will become apparent as the following detailed description proceeds, and upon reference to the drawings, in which like numerals depict like parts, and in which:

[0007] Fig. 1 is a block diagram illustrating an embodiment of a RAID-6 array showing a plurality of stripes with each stripe including data blocks (strips) and P and Q syndromes striped across an array of hard disks;

[0008] Fig. 2 is a block diagram of a system that includes an embodiment of a system that includes an acceleration unit to accelerate computations according to the principles of the present invention;

[0009] Fig. 3 is a block diagram of an embodiment of the data processing unit in the acceleration unit shown in Fig. 2;

[0010] Fig. 4 is a block diagram of an embodiment of a configuration register for any one of the data processing paths in the data processing unit shown in Fig. 3;

[0011] Fig. 5 is a block diagram of an embodiment of a status register for any one of the data processing paths in the data processing unit shown in Fig. 3;

[0012] Fig. 6 illustrates an embodiment of a method for computing P and Q syndromes in parallel using the acceleration unit shown in Fig. 2 and the data processing unit shown in Fig. 3.

[0013] Fig. 7 illustrates an embodiment of a method for performing partial P and Q syndrome computation in the RAID-6 array shown in Fig. 1;

[0014] Fig. 8 illustrates an embodiment of a method for recovering from the loss of two data disks in the RAID-6 array shown in Fig. 1;

[0015] Fig. 9 illustrates an embodiment of a method for performing a compare check that may be performed in the acceleration unit shown in Fig. 2;

[0016] Fig. 10 illustrates an embodiment of a method for performing a result non-zero check that may be performed in the acceleration unit shown in Fig. 2; and
[0017] Fig. 11 is a block diagram of a system that includes an embodiment of an acceleration unit to accelerate RAID-6 computations according to the principles of the present invention.

[0018] Although the following Detailed Description will proceed with reference being made to illustrative embodiments of the claimed subject matter, many alternatives, modifications, and variations thereof will be apparent to those skilled in the art. Accordingly, it is intended that the claimed subject matter be viewed broadly, and be defined only as set forth in the accompanying claims.

DETAILED DESCRIPTION

[0019] Fig. 1 is a block diagram illustrating an embodiment of a RAID-6 array 100 showing a plurality of stripes with each stripe including data blocks (strips) and P and Q syndromes striped across an array of hard disks 150. In the embodiment shown, the RAID array has five hard disks 150: three data disks and 2 syndrome (P, Q) disks. Data is written to the RAID-6 array using block-level striping with P and Q syndromes distributed across the member hard disks in a round robin fashion. Sequential data, for example, a file segmented into blocks may be distributed across a stripe, for example, horizontal stripe 0, with one of the blocks stored in data blocks 102, 104, 106 on three of the data disks 102. A P and a Q syndrome computed for the data blocks 102, 104, 106 in horizontal stripe 0 are stored in a respective P block 130 and Q block 132. P and Q syndrome blocks are stored on different hard disks 150 in each stripe. In one embodiment, there are 512 bytes in each block in a stripe.

[0020] The P syndrome may be generated by performing an exclusive OR (XOR) operation. XOR is a logical operation on two operands that results in a logical value of '1', if only one of the operands has a logical value of '1'. For example, the XOR of a first operand having a value '11001010' and a second operand having a value '1000001' provides a result having a value '01001001'. If the hard drive that stores the first operand fails, the first operand may be recovered by performing an XOR operation on the second operand and the result.

[0021] The P syndrome is the simple parity of data (D) computed across a stripe using Θ(XOR) operations. In a system with n data disks, the generation of the P syndrome is represented by equation 1 below:

\[ P = D_0 \Theta D_1 \Theta D_2 \ldots \Theta D_{n-1} \] (Equation 1)
The computation of the Q syndrome requires multiplication (*) using a Galois Field polynomial (g). Arithmetic operations are performed on 8-bit (byte) Galois-field polynomials at very high performance. A polynomial is an expression in which a finite number of constants and variables are combined using only addition, subtraction, multiplication and non-negative whole number exponents. One primitive polynomial is $x^8 + x^4 + x^3 + x^2 + 1$ which may be denoted in hexadecimal notation by $\text{ID}$. The Galois Field (GF) operations on polynomials are also referred to as GF($2^8$) arithmetic. In a system with n data disks, the generation of the Q syndrome is represented by equation 2 below:

$$Q = g^0 \Theta g^1 \Theta g^2 \Theta g^3 \ldots \Theta g^{n-1} \Theta D_i^1$$  (Equation 2)

Byte-wise Galois-field operations are performed on a stripe basis, where each byte in the block is computationally independent from the other bytes. Byte-wise Galois-Field operations can accommodate as many as 255 ($2^8$-1) data disks.

Fig. 2 is a block diagram of a system that includes an embodiment of an acceleration unit 200 to accelerate computations according to the principles of the present invention.

The system includes a memory 202 for storing work requests 210 and a state machine (micro engine) 204 which processes the work requests 210. The state machine 204 issues instructions to an acceleration unit 208 through a shared memory 206. Based on instructions executed from the state machine 204, the acceleration unit 208 may perform arithmetic operations on 8-bit Galois-Field polynomials.

In one embodiment the state machine 204 may be one of a plurality of micro engines in a processor, for example, an IXP® 2400 processor available from Intel Corporation. The acceleration unit 208 offloads the computation of 8-bit Galois-Field polynomials from the state machine (micro engine) 204.

The memory 202 and the shared memory 206 may be Dynamic Random Access Memory (DRAM), Static Random Access Memory (SRAM), Synchronized Dynamic Random Access Memory (SDRAM), Double Data Rate 2 (DDR2) RAM or Rambus Dynamic Random Access Memory (RDRAM) or any other type of memory.
The acceleration unit 208 includes a data processing unit 218 to perform operations on the data. Based on a work request 210 stored in memory, the state machine 204 may offload computation for a RAID array operation to the acceleration unit 208. In one embodiment, the RAID array may be the RAID-6 array 100 shown. The acceleration unit 208 also includes control registers 212, status registers 214 and an acceleration unit state machine 216.

Fig. 3 is a block diagram of an embodiment of the data processing unit 218 in the acceleration unit 208 shown in Fig. 2. In the embodiment shown, the data processing unit 218 includes two Arithmetic Logical Units (ALUs) 314, 316. The ALUs perform arithmetic and logical operations on operands. The logical or arithmetic operation that is performed by each independent ALU 314, 316 is configurable through the configuration registers 212 (Fig. 2) associated with the ALUs 314, 316.

In an embodiment, each ALU 314, 316 may perform byte-wise arithmetic and logical operations. Each ALU 314, 316 may include an XOR array and compare and zero-check functions and may perform a P or Q syndrome calculation independently in a single-pass of data through the data processing unit 218.

In an embodiment, each ALU 314, 316 has a 64-bit (8-byte) data path and includes 8 configurable Galois Field (GF)-Byte slices. Each GF-Byte slice operates on one byte in the 8-byte data path. Each of the ALUs 314, 316 operates on 8 bytes concurrently in the 64-bit data path. This allows disk block operations to be performed in parallel using the appropriate bytes from each disk block. Each ALU 314, 316 has an respective independent accumulate buffer 306, 308 that may store up to 512 Bytes. The ALUs 314, 316 share a GF(8) byte-wise multiplier 322. The accumulate buffers 314, 316 may be used to store intermediate results of operations in the ALUs 314, 316 and multiplier 322 in the processing unit 218.

The data processing unit 218 has two separate data processing paths. One data processing path includes accumulate buffer 306 and ALU 314. The other data processing path includes accumulate buffer 308 and ALU 316.

Both data processing paths share multiplier 322. If both data processing paths need to perform general multiply-accumulate functions at the same time, the use of multiplier 322 is toggled between the ALUs 314, 316. Although throughput is reduced, the acceleration unit 208 may allow double-disk recovery to be performed.
with a single general multiply-accumulate with a single pass of the data from shared memory 206.

[0034] The acceleration unit 208 uses the single multiplier 322 efficiently with little loss of performance in storage applications, for example, computing P and Q syndromes and performing data recovery. However, the acceleration unit 208 is not limited for use in storage applications. The acceleration unit 208 may be used for computations involving any 8-bit polynomial that generates an 8-bit field. Furthermore, each ALU 314, 316 may work independently on its own polynomial.

[0035] The data processing unit 218 includes a plurality of multiplexers. Each multiplexer encodes (or multiplexes) information for two or more inputs into a single output based on the state of a selector. For example, multiplexers 302, 304, allow selection of the source of data to be stored in each accumulate buffer 306, 308. The data source may be push data received from shared memory 206 (Fig. 2) or intermediate results output from the ALUs 314, 316 or multiplier 322. Multiplexers 310, 312 allow selection of the source for one of the operands to the respective ALU 314, 316. The operand source may be either from accumulate buffer 306 or accumulate buffer 308.

[0036] Multiplexer 318 allows selection of one of the operands for multiplier 322. The operand source is either the result of an operation in ALU 314 or ALU 316, push data received from shared memory, or data that is stored in accumulate buffer 306 or accumulate buffer 308.

[0037] Multiplexer 320 allows selection of one of a plurality of polynomials (POLY1, POLY2) to be used for performing a multiply operation by multiplier 322. The polynomials are programmable and each processing path through a respective ALU 314, 316 may have a different programmable polynomial. For example, in one embodiment POLY1 may be the programmable polynomial associated with ALU 314 and POLY2 may be the programmable polynomial associated with ALU 316. Multiplexer 324 allows selection of one of a plurality of multiplier coefficients (M1, M2) to be used for performing a multiply operation by multiplier 322. For example, multiplier coefficient M1 may be associated with the processing path through ALU 314 and multiplier coefficient M2 may be associated with the processing path through ALU 316.
Multiplexers 328, 332 allows selection of either the result of the current operation output from the respective ALU 314, 316 or XOR 330 from multiplier 322. The result of one of the data processing paths (including ALU 314 or ALU 316) is either the final result to be returned to shared memory or an intermediate result to be stored in the respective accumulate buffer 306, 308.

The 64-bit XOR 330 and multiplexer 326 allows computations in the data processing path to include a multiply operation and an XOR operation to be performed in the data processing path. Based on the selection of outputs through the various multiplexers, each data processing path may include ALU operations (ALUs 314, 316), multiply operations (multiplier 322) and/or Exclusive Or (XOR) operations (XOR 330).

The configuration registers 212 discussed in conjunction with Fig. 2 include a configuration register for each separate data processing path shown in Fig. 3.

Fig. 4 is a block diagram of an embodiment of a configuration register 400 for any one of the data processing paths in the data processing unit 218 shown in Fig. 3.

In one embodiment the configuration register 400 has 32-bits that may be used to manage the data processing paths, that is, management of acceleration functions. In the embodiment shown, the configuration register 400 may include one or more reserved bits.

The configuration register 400 includes a preserve status field 404 which may include one bit indicating whether the status register contents are preserved or cleared after an operation. For example, the preserve status field 404 may be a single bit that may be set to logical '1' to preserve the status or cleared, that is, set to logical '0' to clear the contents of the status register prior to the next operation, so that the status may be updated by the next operation.

The source field 406 in the configuration register 400 may be configured to select the source of data for ALUs 314, 316 through multiplexers 310, 312. For example, the source of data for an operation to be performed in ALU 314 may be accumulate buffer 306 or accumulate buffer 308. In one embodiment, the source field 406 has one bit which is set or cleared to select the source of the ALU operand through multiplexers 310, 312.
The accumulate field 408 in the configuration register 400 selects whether the result for a data processing path through the data processing unit 218 is to be directed back to the respective accumulate buffer.

The calculation mode field 410 in the configuration register 400 allows each data processing path to be configured to perform different operations. The state of the calculation mode field 410 may be used to control the data path through multiplexers 328, 332 to the accumulate buffers 306, 308 and/or multiplexer 334. In one embodiment, the calculation mode field 410 has 3 bits allowing one of 8 operations to be selected. These operations are shown below in Table 1 with the associated calculation mode and the result:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>NULL</td>
<td>Data</td>
</tr>
<tr>
<td>001</td>
<td>Basic XOR,</td>
<td>Data XOR ACC</td>
</tr>
<tr>
<td>010</td>
<td>GF8 XOR,</td>
<td>Data GF8XOR ACC</td>
</tr>
<tr>
<td>011</td>
<td>Find M Byte</td>
<td>(Data == M)? 0x01 : 0</td>
</tr>
<tr>
<td>100</td>
<td>GF8 mul,</td>
<td>M • Data</td>
</tr>
<tr>
<td>101</td>
<td>mul data after XOR</td>
<td>M • (Data XOR ACC)</td>
</tr>
<tr>
<td>110</td>
<td>mul data before XOR</td>
<td>(M • Data) XOR Ace</td>
</tr>
<tr>
<td>111</td>
<td>mul Ace before XOR</td>
<td>(M • Ace) XOR Data</td>
</tr>
</tbody>
</table>

Table 1

XOR operations for modes 100, 101, 110 and 111 are performed by the 64-bit XOR 330. For example, for mode 100, multiplexer 326 selects 64'hO, and the data processing path performs a GF8 multiply operation in multiplier 322 with the result of the operation (M • Data) output from 64-bit XOR 330. The other modes listed in Table 1 will be described in greater detail later.

The configuration register 400 allows each data processing path to configure a different multiplier coefficient and Galois Field (GF) polynomial for performing a multiply operation in multiplier 322. The path through the multiplier 332 is shared by both data processing paths, that is, the path through ALU 314 and the path through ALU 316. Multiplexers 320, 324 allow the multiplier to be used for the current operation to be selected for one of the processing paths.
The multiplier coefficient field 412 allows one of a plurality of multiplier coefficients (M) to be configured. The multiplier coefficient (M1, M2) that is configured for the respective data processing path is input through multiplexer 324 for the respective data processing path that is using the multiplier 322 for the current operation. In one embodiment, the multiplier coefficient field has 8-bits.

The GF polynomial field 414 allows one of a plurality of polynomials to be configured for the data processing path. In one embodiment, the GF polynomial field has 8-bits. The GF polynomial (POLY1, POLY2) that is configured for the respective data processing path is input through multiplexer 320 for the respective data processing path that is currently using the multiplier 322 for the current operation. In one embodiment, the GF polynomial field 414 is set to ‘ID’ (hexadecimal (hex)). However, the GF polynomial field 414 allows other GF polynomials to be configured.

The acceleration unit 208 may operate on any GF (2^8) polynomial field and is optimized to work in a power-efficient manner on the generator ID (hex) that is commonly used. The acceleration unit 208 permits a fast implementation of multiplication with arbitrary coefficients by a single multiplier 322 that processes 8 bytes per cycle. Throughput is increased by performing multiplication on 8 bytes in parallel.

Special multiplexer paths discussed in conjunction with Fig. 3 and instructions enable the acceleration unit 208 to provide high performance on partial (random) writes by overlapping movement of data in and out of the acceleration unit 208 from/to shared memory 206 data movements.

The multiple paths also reduce wasteful multi-pass of data, that is, passing the same data from shared memory 206 to the data processing unit 218 to perform different operations.

The acceleration unit 208 provides high performance without the need to add significant additional logic. In one embodiment, the accumulate buffers 306, 308 are chosen to provide the minimum memory size. The single large multiplier 322 is used efficiently without significant loss of performance in storage applications. The acceleration unit 208 is also flexible in terms of supporting any 8-bit polynomial that generates an 8-bit field. Furthermore, each ALU 314, 316 may work independently on a different polynomial.
Fig. 5 is a block diagram of an embodiment of a status register 500 for any one of the data processing paths in the data processing unit 218 shown in Fig. 3. Each data processing path through a respective ALU 314, 316 in the data processing unit 218 includes a respective status register 500. In the embodiment shown, the status register 500 includes a result zero check field 502 and a byte address field 504 for storing the byte address of the first non-zero result byte. The result zero check field 502 may be one bit field that is set or cleared dependent on the result of an operation. The byte address field 504 may have 9-bits for storing an address of a byte within a 512-byte data block.

In one embodiment, the acceleration unit 208 shown in Fig. 2 accelerates the computation of the P and Q syndromes for a RAID 6 stripe write operation. The P and Q syndromes are computed in parallel in the acceleration unit 208 on data blocks for the stripe that are stored in the shared memory 206. The data blocks for the stripe are received from shared memory 206 by the acceleration unit 208. One data processing path has its respective ALU 314, 316 configured to compute the P syndrome for the stripe and the other data processing path has its respective ALU configured to compute the Q syndrome for the same stripe.

As discussed previously, the P syndrome is computed by performing an XOR operation on the data blocks in the stripe to be stored across n data disk drives and is represented by equation 1 below:

\[ P = D_0 \Theta \Theta \Theta D_2 \ldots \Theta D_{n,i} \] (Equation 1)

Referring to Fig. 1, a stripe includes a group of equal sized data blocks 102, 104, 106 which may be referred to as strips. For each stripe there are two checksums, the P and Q syndromes 130, 132. The P and Q syndromes are computed across a data byte in the data blocks each stripe.

The full-stripe Q syndrome may be calculated using factorization. Each coefficient \( g^1 \) is an 8-bit polynomial. The general multiplication can also be performed efficiently using the acceleration unit 206 in a single pass with a 1 cycle throughput (8 Bytes). The Q syndrome is computed by multiplying (*) data blocks to be stored across n disk drives by a Galois Field (GF) polynomial (g) in multiplier 322 and
performing an XOR operation in XOR on the results of the multiplication operation as shown below in equation 2.

\[
Q = g^{a \ast D_0} \Theta g^{1 \ast D_1} \Theta g^{2 \ast D_2} \ldots \Theta g^{n-1 \ast O_{n-1}} \quad \text{(Equation 2)}
\]

[0060] In an embodiment of the invention, the P and Q syndromes are computed on the same data in parallel. With both computations performed in a single-pass of the data through the acceleration unit 208, data movement is optimized and system bus bandwidth (that is, to/from shared memory 206) is conserved. Thus, the acceleration unit 208 improves performance of the computation of the P and Q syndromes.

[0061] Fig. 6 illustrates an embodiment of a method for computing P and Q syndromes in parallel using the acceleration unit 208 shown in Fig. 2 and the data processing unit 218 shown in Fig. 3.

[0062] As previously discussed, the P and Q syndromes are computed for data blocks in a stripe across a plurality of hard disks. In one embodiment, each block of data (strip) in the stripe may store 512 bytes.

[0063] At block 600, each data processing path is configured by storing configuration data in the respective configuration register 400 in the acceleration unit 208. A first data processing path through ALU 314 may be configured to compute the P syndrome and a second data processing path through ALU 316 may be configured to compute the Q syndrome in parallel on the same data.

[0064] The respective acceleration unit configuration register 400 for each data processing path is initially configured to move data from shared memory 206 (Fig. 2) to the respective accumulate buffers 306, 308. The calculation mode field 410 is set to NULL, the accumulate field 408 is set to '1' to accumulate the push data in the accumulate buffer 306, 308 and the preserve status bit set to '1' to preserve the contents of the respective status register 500. All of the other fields in the respective configuration register 400 are set to the default values, with the GF polynomial field 414 set to the default value of ID (hex), the multiplier co-efficient field 412 is set to the default value of '0' and the accumulate buffer source field 408 set to the respective accumulate buffer 306, 308 associated with the ALU 314, 316.
In one embodiment an instruction to load the acceleration unit configuration register 400 with a configuration register stored in shared memory 206 is executed by the accelerator state machine 216. Processing continues with block 602.

At block 602, each data processing path has been configured through the respective configuration registers 400 to store a block of data from shared memory 206 (Fig. 2) in the respective accumulate buffers 306, 308. A write instruction is issued to both data processing paths by the accelerator state machine 216 to move data from shared memory 206 and store it in the accumulate buffers 306, 308. The data stored in the accumulate buffers 306, 308 is the first data block for the first disk drive in a stripe. Based on the state of the configuration register 400, push data received from shared memory 206 enters ALUs 314, 316. No operation is performed in ALUs 314, 316 because the calculation mode field 410 in the configuration register 400 is NULL. The unmodified data flows out of the ALUs 314, 316 through multiplexers 328, 332 to return through multiplexers 302, 304 to be stored in the accumulate buffers 306, 308. Processing continues with block 604.

At block 604, after the first block of data for the stripe has been stored in the accumulate buffers 306, 308, the configuration registers 400 for the respective data processing paths are configured to perform P and Q syndrome computation. In one embodiment ALU 314 is configured to perform P syndrome computation on the stripe and ALU 316 is configured to perform Q syndrome operation on the stripe based on the state of the configuration registers.

ALU 314 is configured to compute the P syndrome with the calculation mode field 410 set to "XOR", the accumulate field 408 set to accumulate and the preserve status field 404 set to 'preserve' in the associated configuration register 400. All other fields in the configuration register 400 are set to the default values as discussed previously.

ALU 316 is configured to compute the Q syndrome with the calculation mode field 410 set to "mul data before XOR" the accumulate field 408 set to accumulate and the preserve status field 404 set to preserve status in the associated configuration register 400. Processing continues with block 606.

At block 606, each ALU 314, 316 performs the configured operation on two operands based on the state of the respective configuration register 400. The
first operand is stored in the respective accumulate buffer 306, 308. The other operand is streamed from shared memory 206 through push data. The result of the operation performed by the respective ALU 314, 316 is stored in the respective accumulate buffer 306, 308.

[0071] For example, to compute the P syndrome, an XOR operation is performed in ALU 314 on data stored in accumulate buffer 306 received from the accumulate buffer 306 through multiplexer 310 and push data retrieved from shared memory 206. The data (operand) stored in the accumulate buffer 306 may be a first strip for the stripe from a first disk drive. The other data (operand) streamed from shared memory 206 may be the second strip for the stripe from a second disk drive. As the accumulate field 408 in the configuration register 400 is set to accumulate, the result of the operation in ALU 314 is stored back in the accumulate buffer 306 through multiplexers 328 and 302. Processing continues with block 608.

[0072] At block 608, if there is another strip (other than the last strip) in the stripe requiring further P or Q computation for the stripe to be performed, processing continues with block 606. If not, processing continues with block 610.

[0073] At block 610, both the P and Q syndromes have been computed for all but the last strip in the stripe. The P syndrome and the Q syndrome are computed for the stripe with data from the last strip as one of the operands. The other operand is the result of the operation performed on the other strips that is stored in the accumulate buffers 306, 308. The result of the P syndrome computation is sent through multiplexers 328 and 334 to a "pull" First In First Out (FIFO) memory (not shown) that is shared by both data processing paths. The result of the Q operation is stored in accumulate buffer 308. Processing continues with block 612.

[0074] At block 612, the P syndrome computed for the stripe that is stored in the pull FIFO is moved to shared memory 206. Processing continues with block 614.

[0075] At block 614, the accumulate buffer 308 associated with the ALU 316 computing the Q syndrome is flushed to the pull FIFO. Processing continues with block 616.

[0076] At block 616, the Q syndrome stored in the pull FIFO is moved to shared memory 206. Both the Q syndrome and the P syndrome for the stripe have been computed in a single pass of the data received from shared memory 206 for the stripe. The results (P and Q) have been returned to shared memory 206.
[0077] Fig. 7 illustrates an embodiment of a method for performing partial P and Q computation in the RAID-6 array 100 shown in Fig. 1. Partial P and Q computation is performed when data in a stripe has only changed on a single block of data (D) stored on one data disk for the stripe. The inputs are therefore the old values of P, Q and D for the stripe. The outputs are based on the old values of D, P and Q and the new values of D. The following equations define the operations for computing the new values for the P and Q syndromes.

\[
P_{\text{new}} = P_{\text{old}} \oplus (D_{\text{old}} \oplus D_{\text{new}}) \quad \text{(Equation 3)}
\]

\[
Q_{\text{new}} = Q_{\text{old}} \oplus M \cdot (D_{\text{old}} \oplus D_{\text{new}}) \quad \text{(Equation 4)}
\]

[0078] The \((D_{\text{old}} \oplus D_{\text{new}})\) expression is used in both equation 3 and equation 4 to compute the new P syndrome and the new Q syndrome. Thus, this expression may be computed once for \(P_{\text{new}}\) and then used to compute \(Q_{\text{new}}\). The acceleration unit 206 may be configured to perform this computation through the configuration registers 400.

[0079] At block 700, both data processing paths are configured to store old values of the data block D in the respective accumulate buffers 306, 308 by setting appropriate bits in the respective configuration registers 400. The calculation mode field 410 is set to NULL, the accumulate field 408 is set to '1' to store the result back in the accumulate buffers 306, 308 and the preserve status bit set to '1' to preserve the contents of the status register 214. All of the other fields in the configuration register 400 for each data processing path are set to the default values, with the GF polynomial field 414 set to the default value of ID (hex), the multiplier co-efficient field 412 is set to the default value of '0' and the accumulator source field 408 set to the accumulate buffer 306, 308 associated with the respective ALU 314, 316. An instruction is executed by the state machine 216 to load a configuration register stored in shared memory 206 to initialize the fields in each configuration register 400. Processing continues with block 702.

[0080] At block 702, each data processing path has been configured through the respective configuration register 400 to store a block of data from shared memory 206 in the respective accumulate buffers 306, 308. A write instruction is issued to both data processing paths by the accelerator state machine 216 to move data from
shared memory 206 and store it in the accumulate buffers 306, 308. The data stored in the accumulate buffers 306, 308 is the old data block. Based on the state of the configuration register 400, push data received from shared memory 206 enters ALUs 314, 316. No operation is performed in ALUs 314, 316 because the calculation mode field 410 in the configuration register 400 is NULL. The unmodified data flows out of the ALUS 314, 316 through multiplexers 328, 332 to return through multiplexers 302, 304 to be stored in the accumulate buffers 306, 308. Processing continues with block 704.

At block 704, the configuration register 400 for the respective data processing paths are reconfigured to compute a new P and Q syndrome for the new data block. In one embodiment, based on the state of each configuration register 400, ALU 314 is configured to compute the P syndrome and ALU 316 is configured to compute the Q syndrome.

ALU 314 is configured to compute a partial result for the new P syndrome with fields in the respective configuration register 400 set as follows: the calculation mode field 410 set to "XOR", the accumulate field 408 set to accumulate and the preserve status field 404 set to preserve. All other fields in the configuration register 400 are set to the default values as discussed previously.

ALU 316 is configured to compute a partial result for the new Q syndrome with fields in the respective configuration register 400 set as follows: the calculation mode field 410 set to "mul data after XOR", the accumulate field 408 set to accumulate, the preserve status field 404 set to preserve status and the multiplier co-efficient field 412 storing the multiplier co-efficient for the multiply operation to be performed by ALU 316. Processing continues with block 706.

At block 706, each ALU 314, 316 performs the configured operation based on the state of the respective configuration register 400 on two data inputs. The first data input (old data) is stored in the respective accumulate buffer 306, 308, the other data input (new data) is streamed into the ALUs 314, 316 from shared memory 206. The result of the operation performed by the respective ALU 314, 316 is stored in the respective accumulate buffer 306, 308. ALU 314 performs an XOR operation on old data stored in accumulate buffer 306 and new data streamed in through 'push data'. ALU 316 also performs an XOR operation on the old data stored in accumulate buffer 308 and new data streamed in through 'push data' from shared
memory 206. The multiplier 322 multiplies the result of the XOR operation received through multiplexer 318 from ALU 316 by the multiplier co-efficient received from the output of multiplexer 324. After performing the configured operations, accumulate buffer 306 stores the result of \((D_{old} \Theta D_{new})\) and accumulate buffer 308 stores the result of \(M \times (D_{old} \Theta D_{new})\). Processing continues with block 708.

At block 708, the configuration register 400 associated with the computation of \(P\) is configured to compute the new \(P\) syndrome \((P_{new})\) using the partial results stored in accumulate buffer 306. To compute \(P_{new}\), the fields in the configuration register 400 are configured as follows: the calculation mode field 410 set to XOR, the accumulate field 408 set to '1' to accumulate and the preserve status bit set to '1' to preserve the status.

The fields in configuration register 400 for the other data processing path that includes ALU 316 are configured as follows: the calculation mode field 410 set to NULL, the accumulate field 408 set to '0' to turn accumulate off, and the preserve status bit set to '1' to preserve the contents of the status register. Processing continues with block 710.

At block 710, the old \(P\) is streamed from shared memory 206 into ALUs 314, 316. The old \(P\) and the result of the prior operation stored in the accumulate buffers 306, 308 are input to ALUs 314, 316. The result of the operation in ALU 314, that is, \(P_{old} \Theta\) (contents of accumulate buffer 306) is the new \(P\) \((P_{new})\) which is forwarded through multiplexer 334 to the Pull FIFO and also sent back to accumulate buffer 306. As the NULL operation is performed in ALU 316 and the result returned to accumulate buffer 308, the data stored in accumulate buffer 308 is unchanged. Processing continues with block 712.

At block 712, the new \(P\) \((P_{new})\) stored in the Pull FIFO is stored in shared memory 206. Processing continues with block 714.

At block 714, the configuration register 400 associated with the data processing path that includes ALU 316 is configured to compute the new \(Q\) \((Q_{new})\) using the partial results stored in accumulate buffer 308. The configuration register 400 associated with computing the \(P\) syndrome is configured as follows: the calculation mode field 410 set to NULL, the accumulate field 408 set to '0' to not accumulate, and the preserve status bit set to '1' to preserve the contents of the status register. The configuration register 400 associated with computing the \(Q\)
syndrome is configured to compute the new Q syndrome with fields configured as follows: the calculation mode field 410 set to XOR, the accumulate field 408 set to '1' to accumulate and the preserve status bit set to '1' to preserve the status. Processing continues with block 716.

At block 716, the old Q is streamed through Push data from shared memory 206. The old Q and the result of the prior operation stored in the accumulate buffers 306, 308 are input to ALUs 314, 316. The result of the operation in ALU 316, that is QoId Θ (contents of accumulate buffer 308) is the new Q (Qnew) which is forwarded to the Pull FIFO and back to accumulate buffer 308. As the NULL operation is performed in ALU 314 and the result returned to accumulate buffer 306, the data stored in accumulate buffer 306 is unchanged. After block 714 executes, the data transfers in blocks 712 and 716 run concurrently. Thus, the new P is moved to shared memory 206 while the new Q is being computed. This reduces the time to output the result of the operation from the acceleration unit. Processing continues with block 718.

At block 718, the new Q (Qnew) stored in the Pull FIFO is read from the FIFO and stored in shared memory 206.

Fig. 8 is an embodiment of a method for recovering from the loss of two data disks in the RAID-6 array shown in Fig. 1. For example, if there are n data disks and data disks D0 and D1 have failed, data disks D0 and D1 may be recovered by performing the following computations on the remaining disks:

\[ D_0 = a^2 \Theta a^3 \Theta D_3 \ldots \Theta a^n \Theta P \Theta a^*Q \quad (\text{Equation 5}) \]
\[ D_1 = b^2 \Theta b^3 \Theta D_3 \ldots \Theta b^n \Theta b^*P \Theta b^*Q \quad (\text{Equation 6}) \]

Typically, D0 is computed first. D1 may be computed using D0 as follows:

\[ D_1 = D_0 \Theta D_2 \Theta D_3 \ldots \Theta D_n \Theta P \quad (\text{Equation 7}) \]

However, this requires two passes of the data. As shown above, both D0 and D1 computation using equations 5 and 6 above require the use of a multiplier. As the probability that two disks will fail simultaneously is low, the additional cost of adding a second multiplier 322 to the acceleration unit 208 may not be justified for the small performance gain. Thus, to allow for recovery of two data disks with a
single pass of the data, the single multiplier is shared by the two data processing paths. The multiplier may be shared via the selection of the polynomial (P) through multiplexer 320 and the multiplication coefficient (M) through multiplexer 324 for the particular data processing path that is using the multiplier 322.

Using the data processing unit 218 shown in Fig. 3, in an embodiment in which there are six data disks D0-D5 and D4 and D5 have failed, D4 and D5 may be recovered in a single pass of the data using D0-D3, P and Q.

D4 may be computed as a linear combination of the set \{D0, D1, D2, D3, P, Q\} with multiplier coefficients \{mθ, ml, m2, m3, mP, mQ\}. In an embodiment, each data block in the set may have 512 bytes and each multiplier coefficient is one byte and is applied to the entire data block. Thus, the following operations are performed to recover D4 and D5:

\[
D4 = (mO*D0 \oplus Hi*I*D1 \oplus m2*D2 \oplus m3*D3 \oplus mP*P) \oplus mQ*Q
\]

\[
D5 = (D_6 \oplus D_i \oplus D_2 \oplus D_3 \oplus P) \oplus D_4
\]

At block 800, the ALUS 314, 316 for each data processing path are configured. The configuration register 400 associated with ALU 314 is configured as follows: the calculation mode field 410 is set to NULL and the accumulate field 408 is set to accumulate. The configuration register 400 associated with ALU 316 in the other data processing path is configured as follows: the calculation mode field 410 is set to GF8 MUL and the accumulate field 408 is set to accumulate the results of the GF8 MUL operation. Processing continues with block 802.

At block 802, each data processing path performs the configured operations on a block of data received from shared memory. The NULL operation in ALU 314 results in the received block of data being stored in accumulate buffer 306. The result of the GF8 multiply on the block of data is stored in accumulate buffer 308. Processing continues with block 804.

At block 804, the multiplier co-efficient (M) for the next data block is stored in the multiplier co-efficient field 412 in the configuration register 400 for ALU 316. The operation stored in both configuration registers 400 (XOR in ALU
314 and mul data before XOR in ALU 316) is performed on the result of the previous operation stored in the respective accumulate buffers 306, 308 and the new block streamed in from shared memory 206. Processing continues with block 806.

[0099] At block 806, if there is another data block to be streamed in, processing continues with block 804. If not, processing continues with block 808.

[0100] At block 808, accumulate buffers 306, 308 store the results of the following operations:

\[
\text{Accumulate buffer 308} = (m_0 \cdot D_0 \Theta H_i \cdot D_1 \Theta m_2 \cdot D_2 \Theta m_3 \cdot D_3)
\]
\[
\text{Accumulate buffer 306} = (D_0 \Theta D_i \Theta D_2 \Theta D_3)
\]

[0101] The configuration register 400 associated with the data processing path that includes ALU 314 is configured with the multiplier coefficient for P. The P block is streamed into both ALUs 314, 316. The accumulate buffers 306, 308 store the results of the following operations:

\[
\text{Accumulate buffer 308} = (m_0 \cdot D_0 \Theta H_i \cdot D_1 \Theta m_2 \cdot D_2 \Theta m_3 \cdot D_3 \Theta mP \cdot P)
\]
\[
\text{Accumulate buffer 306} = (D_0 \Theta D_i \Theta D_2 \Theta D_3 \Theta P)
\]

[0102] At block 810, the configuration register 400 associated with the data processing path that includes ALU 316 is configured with the multiplier coefficient for Q; the configuration register 400 associated with the data processing path that includes ALU 314 is configured to not accumulate the results of this operation. The Q data is streamed into both ALUs 314, 316. The result of the operation in ALU 316, that is mQ*Q \Theta (contents of accumulate buffer 308) is the recovered D4 which is forwarded to the Pull FIFO and back to accumulate buffer 308. After the operation configured in each ALU has been performed on the Q block, the accumulate buffers 306, 308 store the result of the following operations.

\[
\text{Accumulate buffer 308} = D_4 = ((m_0 \cdot D_0 \Theta H_i \cdot D_1 \Theta m_2 \cdot D_2 \Theta m_3 \cdot D_3 \Theta mP \cdot P) \Theta mQ \cdot Q)
\]
\[
\text{Accumulate buffer 306} = (D_0 \Theta D_i \Theta D_2 \Theta D_3 \Theta P)
\]
Processing continues with block 812.

At block 812, the recovered data for D4 is stored back to shared memory 206.

Processing continues with block 814.

[0103] At block 814, the configuration register 400 associated with the data processing path that includes ALU 314 is configured to XOR and accumulate; the configuration register 400 associated with the data processing path that includes ALU 316 is configured to disable the results from that path from being returned to the accumulate buffer. The D4 data is streamed into both ALUS 314, 316. The result of the operation in ALU 314, that is D4 \( \Theta \) (contents of accumulate buffer 306), is the recovered D5 which is forwarded to the Pull FIFO and back to accumulate buffer 306. That is, the following operation is performed:

\[
D5 = (D_0 \Theta D_1 \Theta D_2 \Theta D_3 \Theta P) \Theta D_4
\]

The result is stored in accumulate buffer 306.

At block 816, the recovered data for data disk D5 is returned to shared memory.

[0104] In addition to the methods for performing recovery from a double disk failure, partial P and Q update and computation of P and Q syndrome described in conjunction with Figs. 6-8, the acceleration unit may also be used to accelerate the recovery from a data disk and P disk double failure or a data disk and Q disk double failure by configuring the configuration registers to perform the recovery operation. In the case of a failure of both the P and Q disks, the P and Q disks may be recovered using the method described in conjunction with Fig. 7.

[0105] The acceleration unit may be used to accelerate data processing operations other than those involving recovery of P, Q and data disks and computation of P and Q syndromes for a RAID-6 system.

[0106] Fig. 9 illustrates an embodiment of a method for performing a compare check that may be performed in the acceleration unit shown in Fig. 2. The compare function compares two data blocks (A and B) and checks that data block A is equal to data block B. Fig. 9 will be described in conjunction with Figs. 2 and 3.

[0107] At block 900, the configuration register 400 for ALU 314 is configured to perform the compare operation. The calculation mode field 410 of the configuration register 400 for ALU 314 is set to XOR and the accumulate field 408 is configured to accumulate the results. The calculation mode field 410 of the configuration register
400 for ALU 316 is set to NULL with no accumulate of the results. Processing continues with block 902.

[0108] At block 902, data from a first data block (the 'A data') is loaded into accumulate buffer 306. Processing continues with block 904.

[0109] At block 904, data from a second data block (the 'B data') is streamed into the acceleration unit 208 from the shared memory 206 and an XOR operation is performed on a byte-by-byte basis with the A data stored in the accumulate buffer 306. Processing continues with block 906.

[0110] At block 906, after the B data has been streamed through ALU 314, the status register 500 associated with ALU 314 is read. The state of the result zero check field 502 in the status register 500 associated with ALU 314 indicates whether a miscompare was detected. Also, the byte address of the first miscompare byte in the data block may be stored in the byte address field 504 in the status register 500.

[0111] Fig. 10 is an embodiment of a method for performing a result non-zero check that may be performed in the acceleration unit 208 shown in Fig. 2.

[0112] At block 1000, the configuration register 400 for ALU 314 is configured to perform the result non-zero check operation. The calculation mode field 410 of the configuration register 400 for ALU 314 is set to NULL, the preserve status field 404 is set to update status, and the accumulate field 408 is configured not to accumulate the results. The calculation mode field 410 of the configuration register 400 for ALU 316 is also set to NULL with no accumulate of the results. Processing continues with block 1002.

[0113] At block 1002, data from a data block is streamed into both ALUS 314, 316 from the shared memory and a NULL operation is performed on a byte-by-byte basis with no data stored in the accumulate buffer 306.

[0114] At block 1004, after the data has been streamed through the ALUs 314, 316, the status register 500 associated with ALU 314 is read. The state of the result zero check field 502 in the status register 500 associated with ALU 314 indicates whether a result non-zero was detected. Also, the byte address of the first non-zero result byte in the block may be stored in the byte address field 504 in the status register 500.

[0115] In another embodiment, the ALU in one data processing path may be configured to perform a result non-zero check and the ALU in the other data
processing path configured to perform a compare check. The two operations may be performed in parallel on the same block of data.

[0116] Enhanced Byte-search functions are useful in Public Key Cryptography Systems (PKCS) and Random Number Generators (RNG), for example, to search for Zero bytes or to find a variable pattern at the start of a message of the form 0*0x01*. In the latter case, the position of the variable pattern indicates the start of a well-formed message body. The search for the start of the well-formed message body is very slow and requires a large amount of code space when performed in the micro engine 204.

[0117] Fig. 11 is a block diagram of a system that includes an embodiment of an acceleration unit to accelerate RAID-6 computations according to the principles of the present invention.

[0118] The system 1100 includes a host Central Processing Unit (CPU) 1101, a Memory Controller Hub (MCH) 1102 and an I/O Controller Hub (ICH) 1104. The MCH 1102 includes a memory controller 1106 that controls communication between the CPU 1101 and memory 202. The CPU 1101 and MCH 1102 communicate over a system bus 1116.

[0119] The Host Central Processing Unit (CPU) 1101 may be any one of a plurality of processors such as a single core Intel® Pentium IV® processor, a single core Intel Celeron processor, an® XScale processor or a multi-core processor such as Intel® Pentium D, Intel® Xeon® processor, or Intel® Core® Duo processor or any other type of processor.

[0120] The memory 202 may be Dynamic Random Access Memory (DRAM), Static Random Access Memory (SRAM), Synchronized Dynamic Random Access Memory (SDRAM), Double Data Rate 2 (DDR2) RAM or Rambus Dynamic Random Access Memory (RDRAM) or any other type of memory.

[0121] The ICH 1104 may be coupled to the MCH 1102 using a high speed chip-to-chip interconnect 1114 such as Direct Media Interface (DMI). DMI supports 2 Gigabit/second concurrent transfer rates via two unidirectional lanes.

[0122] The ICH 1104 may include an acceleration unit 208 for accelerating RAID computations for a RAID array 100 coupled to the ICH 1104. The ICH 1104 may communicate with the RAID array 100 over a storage protocol interconnect 1118 using a serial storage protocol. Commands, data and status information encapsulated in frames
may be exchanged between the ICH 1104 and the RAID array 100, over the storage protocol interconnect 1118 using standard serial attached storage protocol suites.


It will be apparent to those of ordinary skill in the art that methods involved in embodiments of the present invention may be embodied in a computer program product that includes a computer usable medium. For example, such a computer usable medium may consist of a read only memory device, such as a Compact Disk Read Only Memory (CD ROM) disk or conventional ROM devices, or a computer diskette, having a computer readable program code stored thereon.

While embodiments of the invention have been particularly shown and described with references to embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the scope of embodiments of the invention encompassed by the appended claims.
CLAIMS

1. An apparatus comprising:
   a first data processing path to receive a block of data, the first data
   processing path including a first accumulate buffer to store the block of data and a
   first arithmetic logical unit to perform a first operation on the block of data; and
   a second data processing path to receive the block of data, the second data
   processing path including a second accumulate buffer to store the block of data and a
   second arithmetic logical unit to perform a second operation on the block of data,
   the first data processing path and the second data processing path sharing a
   multiplier, the multiplier to perform a multiply operation on the block of data, each
   of the data processing paths to process the block of data in parallel to provide a
   first result block of data and a second result block of data in a single pass of the
   block of data through the data processing paths.

2. The apparatus of claim 1, wherein the block of data is a data block in a Redundant
   Array of Independent Disks (RAID) stripe, the first operation to compute a P
   syndrome for the stripe and the second operation to compute a Q syndrome for the
   stripe.

3. The apparatus of claim 1, wherein the block of data is a data block in a Redundant
   Array of Independent Disks (RAID) stripe, the RAID stripe includes a P syndrome
   block and a Q syndrome block, the first operation to recover a first data block and
   the second operation to recover a second block in the RAID stripe using the P
   syndrome block, the Q syndrome block and other data blocks in the RAID stripe.

4. The apparatus of claim 1, wherein the block of data is a data block in a Redundant
   Array of Independent Disks (RAID) stripe, the RAID stripe is updated for a new
   data block, the first operation to compute a new P syndrome for the stripe using an
   old P syndrome block, an old data block and a new data block for the stripe.
5. The apparatus of claim 4, wherein the second operation to compute a new Q syndrome using an old Q syndrome block, the old data block and the new data block for the stripe.

6. The apparatus of claim 5, wherein the new P syndrome is moved to a shared memory while the new Q syndrome is computed.

7. The apparatus of claim 1, wherein the first arithmetic logical unit has a first programmable polynomial and the second arithmetic logical unit has a second programmable polynomial.

8. The apparatus of claim 1, wherein the first operation to perform on the first block of data and a second block, the first block of data and the second block included in a Redundant Array of Independent Disks (RAID) level 6 stripe, the result to be stored in the first accumulate buffer for a subsequent operation with another block.

9. A method comprising:
   storing a block of data in a first accumulate buffer in a first data processing path;
   storing the block of data in a second accumulate buffer in a second processing path, the first processing path and the second processing path sharing a multiplier;
   performing, by a first arithmetic unit a first operation on the block of data; and
   performing, by a second arithmetic unit a second operation on the block of data, the first data processing path and the second data processing path processing the block of data in parallel to provide a first result block of data and a second result block of data in a single pass of the block of data through the data processing paths.

10. The method of claim 9, wherein the block of data is a data block in a Redundant Array of Independent Disks (RAID) stripe, the first operation computes a P
syndrome for the stripe and the second operation computes a Q syndrome for the stripe.

11. The method of claim 9, wherein the block of data is a data block in a Redundant Array of Independent Disks (RAID) stripe, the RAID stripe includes a P syndrome block and a Q syndrome block, the first operation to recover a first data block and the second operation to recover a second block in the RAID stripe using the P syndrome block, the Q syndrome block and other data blocks in the RAID stripe.

12. The method of claim 9, wherein the block of data is a data block in a Redundant Array of Independent Disks (RAID) stripe, the RAID stripe is updated for a new data block, the first operation computing a new P syndrome for the stripe using an old P syndrome block, an old data block and a new data block for the stripe.

13. The method of claim 12, wherein the second operation computes a new Q syndrome using the old Q syndrome block, the old data block and the new data block for the stripe.

14. The method of claim 13, wherein the new P syndrome is moved to a shared memory while the new Q syndrome is computed.

15. The method of claim 9, wherein the first arithmetic logical unit has a first programmable polynomial and the second arithmetic logical unit has a second programmable polynomial.

16. The method of claim 9, wherein the first operation to perform a pattern match on the block of data stored in the first accumulate buffer and a second block of data received by the first data processing path.

17. The method of claim 9, wherein the first operation to perform a non zero result operation on the received block of data.
18. The method of claim 9, wherein the first operation to perform on the first block of data and a second block, the first block of data and the second block included in a Redundant Array of Independent Disks (RAID) level 6 stripe, the result to be stored in the first accumulate buffer for a subsequent operation with another block.

19. An article including a machine-accessible medium having associated information, wherein the information, when accessed, results in a machine performing:

- storing a block of data in a first accumulate buffer in a first data processing path;
- storing the block of data in a second accumulate buffer in a second processing path, the first processing path and the second processing path sharing a multiplier;
- performing, by a first arithmetic unit a first operation on the block of data; and
- performing, by a second arithmetic unit a second operation on the block of data, the first data processing path and the second data processing path processing the block of data in parallel to provide a first result block of data and a second result block of data in a single pass of the block of data through the data processing paths.

20. A system comprising:

- a redundant array of independent disks (RAID) system having a plurality of Serial Attached Small Computer System Interface (SAS) disk drives; and
- an acceleration unit to process data for the RAID system, the acceleration unit comprising:

  a first data processing path to receive a block of data, the first data processing path including a first accumulate buffer to store the block of data and a first arithmetic logical unit to perform a first operation on the block of data; and

  a second data processing path to receive the block of data, the second data processing path including a second accumulate buffer to store the block of data and a second arithmetic logical unit to perform a second operation on the block of data, the first data processing path and the second
data processing path sharing a multiplier, the multiplier to perform a multiply operation on the block of data, each of the data processing paths to process the block of data in parallel to provide a first result block of data and a second result block of data in a single pass of the block of data through the data processing paths.
FIG. 1
FIG. 2
FIG. 5

RESULT ZERO CHECK

BYTE ADDRESS OF FIRST NON-ZERO RESULT BYTE

502 504
FIG. 6

600 CONFIGURE ACCELERATION UNIT TO STORE FIRST BLOCK

602 STORE FIRST BLOCK IN ACCUMULATE BUFFERS

604 CONFIGURE ACCELERATION UNIT TO COMPUTE P AND Q

606 COMPUTE P AND Q

608 ANOTHER BLOCK?

610 COMPUTE P AND Q WITH LAST BLOCK AND STORE P IN FIFO

612 RETURN P

614 FLUSH ACCUMULATE BUFFER STORING Q

616 RETURN Q
FIG. 7
FIG. 8

800  CONFIGURE OPERATION IN ACCELERATION UNIT

802  PERFORM CONFIGURED OPERATION

804  CONFIGURE MULTIPLY COEFFICIENT FOR BLOCK FROM NEXT DATA DISK & PERFORM CONFIGURED OPERATIONS

806  YES  ANOTHER DATA DISK ?

808  NO  CONFIGURE MULTIPLY COEFFICIENT FOR P & PERFORM CONFIGURED OPERATIONS

810  CONFIGURE MULTIPLY COEFFICIENT FOR Q & PERFORM CONFIGURED OPERATIONS

812  RETURN RECOVERED DATA FOR FIRST FAILED DISK

814  CONFIGURE ACCELERATION UNIT TO COMPUTE RECOVERED DATA FOR SECOND FAILED DISK

816  RETURN RECOVERED DATA FOR SECOND DISK
FIG. 9
FIG. 10

1000
CONFIGURE ACCELERATION UNIT

1002
STREAM DATA INTO ACCELERATION UNIT AND CHECK FOR NON-ZERO

1004
READ STATUS REGISTER
INTERNATIONAL SEARCH REPORT

INTERNATIONAL SEARCH REPORT
PCT/US2007/086672

A. CLASSIFICATION OF SUBJECT MATTER

G06F 3/06(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 8 G06F, G1B, H03M

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean Utility Models and applications for Utility Models since 1975
Japanese Utility Models and applications for Utility Models since 1975

Electronic database consulted during the international search (name of data base and, where practicable, search terms used)
eKIPASS/KIPO internal) 'RAID', 'syndrome', 'path', 'multiply', 'parallel'

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>US 2003/0070042 A1 (JAMES BYRD et al.) 10 April 2003 See the abstract, figures 1-8, paragraphs [0017] - [0018] and claims 1-18</td>
<td>1-20</td>
</tr>
<tr>
<td>A</td>
<td>JP 2000/259359 A (TOSHIBA CORP.) 22 September 2000 See the abstract, figures 1-8, paragraphs [0029] - [0073] and claims 1-7</td>
<td>1-20</td>
</tr>
<tr>
<td>A</td>
<td>US 2006/0156211 A1 (SAMANTHA J EDIRISOORIYA et al.) 13 July 2006 See the abstract, figures 1-4, paragraphs [0009] - [0037] and claims 1-24</td>
<td>1-20</td>
</tr>
<tr>
<td>A</td>
<td>US 2005/0050384 A1 (ROBERT L HORN) 03 March 2005 See the abstract, figures 1-3, paragraphs [0018] - [0041] and claims 1-15</td>
<td>1-20</td>
</tr>
<tr>
<td>A</td>
<td>US 2003/0066010 A1 (JOHN D ACTON) 03 April 2003 See the abstract, figures 1-7, paragraphs [0018] - [0046] and claims 1-32</td>
<td>1-20</td>
</tr>
<tr>
<td>A</td>
<td>US 2002/0157044 A1 (JAMES M BYJAD) 24 October 2002 See the abstract, figures 1-10, paragraphs [0011] - [0019] and claims 1-35</td>
<td>1-20</td>
</tr>
<tr>
<td>A</td>
<td>US 6247157 B1 (SAMANTHA J W EDIRISOORIYA) 12 June 2001 See the abstract, figures 1, 3-4, column 1, line 27 - line 49 and claims 1-36</td>
<td>1-20</td>
</tr>
</tbody>
</table>

□ Further documents are listed in the continuation of Box C × See patent family annex

* Special categories of cited documents
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)
"D" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance, the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance, the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search
13 MAY 2008 (13 05 2008)

Date of mailing of the international search report
13 MAY 2008 (13.05.2008)

Name and mailing address of the ISA/KR
Korean Intellectual Property Office
Government Complex-Daejeon, 139 Seomsa-ro, Seogu, Daejeon 302-701, Republic of Korea

Authorized officer
LEE, Jung Ho
Telephone No 82-42-481-5704

Facsimile No 82-42-472-7140

Form PCT/ISA/210 (second sheet) (April 2007)
<table>
<thead>
<tr>
<th>Patent document cited in search report</th>
<th>Publication date</th>
<th>Patent family member(s)</th>
<th>Publication date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JP 2000-259359 A</td>
<td>22.09.2000</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 07343546 B2</td>
<td>11.03.2008</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 2006071837 A2</td>
<td>06.07.2006</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>US 2005/0050384 A1</td>
<td>03.03.2005</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2003066010 A1</td>
<td>03.04.2003</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2002157044 A1</td>
<td>24.10.2002</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>US 6247157 B1</td>
<td>12.06.2001</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>