X3T10/94-111r9

XOR Commands on SCSI Disk Drives

Gerry Houlder
Seagate Technology
8001 E. Bloomington Freeway
Bloomington, MN 55420-1094
TEL: (612) 844-5869
FAX: (612) 844-5708
Gerry_Houlder@notes.seagate.com

Jay Elrod (Technical Editor)
Seagate Technology
8001 E. Bloomington Freeway
Bloomington, MN 55420-1094
TEL: (612) 844-5978
FAX: (612) 844-5708
Jay_Elrod@notes.seagate.com

Mike Miller
Seagate Technology
8001 E. Bloomington Freeway
Bloomington, MN 55420-1094
TEL: (612) 844-5924
FAX:
Mike_Miller@notes.seagate.com

1. Model for xor commands

In storage arrays, an array controller organizes a group of storage devices into a redundancy group. Some areas within the address space of the storage array are used for check data. The check data is generated by performing a cumulative exclusive-or (xor) operation with the data from other areas within the address space of the storage array known as protected data. This xor operation can be performed by the array controller or by the storage device.

Performing the xor operation in the storage device may result in a reduced number of data transfers across the interconnect. For example, when the xor operation is done within the array controller four data transfer operations are needed for a typical update write sequence: a read transfer from the device containing protected data, a write transfer to the device containing protected data, a read transfer from the device containing check data, and a write transfer to the device containing check data. The array controller also does two internal xor operations in this sequence. In contrast, during *array controller supervised* xor operations (see clause 1.1) only three data transfer operations are needed: a write transfer to the device containing protected data, a read transfer from the device containing protected data, and a write transfer to the device containing check data. Note that the array controller doesn't do any internal xor operations. In further contrast, during *third party* xor operations (see clause 1.2) only two data transfer operations are needed: a write transfer from the array controller to the device containing protected data and a write transfer from the device containing protected data to the device containing check data. Note that the array controller only issues one command and does no xor operations.

Performing the xor operation in the device eliminates the need for the array controller to perform any xor operations. An array controller performs three basic operations that require xor functionality. These are the update write, regenerate, and rebuild operations. A command sequence for each of these operations is defined for the following operating modes. The command sequences use the device to perform the xor functions needed for the major operations.

1.1 Array Controller Supervised Xor Operations

Three xor commands are needed to implement array controller supervised xor operations: XDWRITE, XPWRITE, and XDREAD. The array controller will also use READ and WRITE commands for certain operations. This technique may be used when all of the devices are in the same domain, when all devices are in separate domains, or any combination thereof, as long as the domains are accessible by the array controller.

1.1.1 The Update Write operation

The update write operation is used to write new data to a device containing protected data and update the parity information on the device containing check data.

1) An XDWRITE command is sent to the device containing protected data. This transfers the new write data to that device. The device reads the old data, performs an xor operation on the old data and the new data, retains the xor result in its buffer, and writes the new data to the medium.
2) An XDREAD command is sent to the device containing protected data. This transfers the retained xor data from that device to the array controller.
3) An XPWRITE command is sent to the device containing check data. This transfers the xor data (received in the previous XDREAD command) to the device containing check data. The device reads the old data, performs an xor operation on the old data and the new data, and writes the xor result to the medium.

1.1.2 The Regenerate operation

The regenerate operation is used to recreate a data block that cannot be read from a data device. This is done by reading the associated data block from each of the other devices within the redundancy group and performing an xor operation with each of these data blocks. The last xor result is the data that should have been present on the unreadable device. The number of steps is dependent on the number of devices in the redundancy group, but the sequence is as follows:

1) A READ command is sent to the first device. This transfers the old data from the device to the array controller.
2) An XDWRITE command with the disable write bit set is sent to the next device. This transfers the data from the previous read operation to the device. The device reads its old data, performs an xor operation on the old data and the new data, and retains the xor result in its buffer.
3) An XDREAD command is sent to the same device as in step 2. This transfers the retained xor data from the device to the array controller.
4) Steps 2 and 3 are repeated until all devices (except the failed device) in the redundancy group have been accessed. The xor data returned by the last XDREAD command is the regenerated data for the failed device.

1.1.3 The Rebuild operation

The rebuild operation is similar to the regenerate operation, except that the last xor result is written to the replacement device. This is used when a failed device is replaced and the array controller is writing the rebuilt data to that replacement device. This sequence is as follows:

1) A READ command is sent to the first device. This transfers the old data from the device to the array controller.
2) An XDWRITE command with disable write bit set is sent to the next device. This transfers the data from the previous read operation to the device. The device reads its old data, performs an xor operation on the old data and the new data, and retains the xor result in its buffer.
3) An XDREAD command is sent to the same device as in step 2. This transfers the retained xor data from the device to the array controller.
4) Steps 2 and 3 are repeated until all devices (except the replacement device) in the redundancy group have been accessed. The xor data returned by the last XDREAD command is the regenerated data for the replacement device.
5) A WRITE command is sent to the replacement device. This transfers the final xor result data from step 4 to the replacement device. The device writes this data to the medium.

1.2 Third Party Xor Operations

Five xor commands are needed to implement the third party xor operations: XDWRITE EXTENDED, XPWRITE, XDREAD, REGENERATE, and REBUILD. The array controller will also use READ and WRITE for certain operations. Third party xor operations are restricted to systems where all devices involved in the commands are located in the same domain since interaction between those devices is required.

1.2.1 The Update Write operation

The update write operation is used to write new data to a device containing protected data and update the parity information on the device containing check data.

1) An XDWRITE EXTENDED command is sent to the device containing protected data. This transfers the new write data to that device. The device reads its old data, performs an xor operation on the old data and the new data, temporarily stores the xor result in its buffer, and writes the new data to the medium.
2) The device containing protected data becomes a temporary initiator and sends an XPWRITE command to the device containing check data. This transfers the resulting xor data from the device containing protected data to the device containing check data. The device containing check data reads its old data, performs an xor operation on the old data and the new data, and writes the xor result to the medium.
3) After the device containing protected data receives status for its XPWRITE command, it returns ending status for the XDWRITE EXTENDED command to the array controller. This indicates that the operations on both the device containing protected data and the device containing check data have completed.

1.2.2 The Regenerate operation

The regenerate operation is used to recreate a data block that cannot be read from a data device. This is done by reading the associated data block from each of the other devices within the redundancy group and performing an xor operation with each of these data blocks. The last xor result is the data that should have been present on the unreadable device. The number of steps is dependent on the number of devices in the redundancy group, but the sequence is as follows (note: since xor operands are commutable the xor order is irrelevant, and the order of steps 2 and 3 is interchangable):

1) A REGENERATE command is sent to a valid device in the redundancy group (a valid device is any device other than the failed device). This transfers the regenerate parameter list from the array controller to the device.
2) The device reads the requested data from its own medium. The device retains this data for a future xor operation.
3) The device becomes a temporary initiator and sends a READ command to another device included in the regenerate parameter list. That target device transfers the requested data to the temporary initiator. The temporary initiator performs an xor operation between the read data from the previous step and the read data received in this step. The resulting xor data is retained for the next step.
4) Step 3 is repeated until all devices listed in the regenerate parameter list have been accessed. The last xor data result is retained in the temporary initiator's buffer. The temporary initiator returns completion status for the REGENERATE command to the array controller when this data is available.
5) An XDREAD command is sent from the array controller to the temporary initiator device. This transfers the regenerated data from the device to the array controller.

1.2.3 The Rebuild operation

The rebuild operation is similar to the regenerate operation, except that the last xor result is written to the replacement device. This is used when a failed device is replaced and rebuilt data is written to that replacement device. This sequence is as follows:

1) A REBUILD command is sent to the replacement device in the redundancy group (the device that replaces a failed device). This transfers the rebuild parameter list from the array controller to the device.
2) The device becomes a temporary initiator and sends a READ command to a device included in the rebuild parameter list. That target device transfers the requested data to the temporary initiator. The temporary initiator retains this data for a future xor operation.
3) The temporary initiator sends a READ command to another device included in the rebuild parameter list. That device transfers the requested data to the temporary initiator. The temporary initiator performs an xor operation between the read data from the previous step and the read data received in this step. The resulting xor data is retained for the next step.
4) Step 3 is repeated until all devices listed in the rebuild parameter list have been accessed. The last xor data result is written to the replacement device's medium, then the device returns completion status for the REBUILD command to the array controller.

1.3 Hybrid Subsystem Xor Operations

A hybrid subsystem is one in which the redundancy group is divided between two or more domains (see 1.1) and at least one of those domains contains two or more of the devices in the redundancy group. Such a system could do its xor operations as described in clause 1.1 (Array Controller Supervised xor operations) but it may choose to use third party xor commands for parts of the xor operation where all the involved devices are in the same domain. This clause describes use of the third party xor operations on a hybrid system with two domains. The redundancy group has six devices, with three devices in each domain.

1.3.1 The Update Write operation

When the update write operation involves two devices that are in different domains, the array controller must use the technique described in array controller supervised xor operations. When the update write operation involves two devices that are in the same domain the array controller may use either the array controller supervised or the third party technique.

1.3.2 The Regenerate operation

The regenerate operation will always involve five devices (all but the failed device) where three devices are in one domain (referred to as domain A) and two devices in the other domain (referred to as domain B). The sequence is as follows (note: since xor operands are commutable the xor order is irrelevant, and the order of steps 2 and 3 is interchangable; likewise, the order of steps 7 and 8 is interchangable):

1) A REGENERATE command is sent to a device in domain A. This transfers the regenerate parameter list (containing the other two valid devices and data extents) from the array controller to the device.
2) The device reads the requested data from its own medium. The device retains this data for a future xor operation.
3) The device becomes a temporary initiator and sends a READ command to another device included in the regenerate parameter list. That device transfers the requested data to the temporary initiator. The temporary initiator performs an xor operation between the read data from the previous step and the read data received in this step. The resulting xor data is retained for the next step.
4) Step 3 is repeated until all devices listed in the regenerate parameter list have been accessed. The last xor data result is retained in the temporary initiator's buffer. The temporary initiator returns completion status for the REGENERATE command to the array controller when this data is available.
5) An XDREAD command is sent from the array controller to the temporary initiator device. This transfers the partially regenerated data from the device to the array controller.
6) A REGENERATE command with the intermediate data bit set is sent to a device in domain B. This transfers the regenerate parameter list (containing the other valid device and data extent) from the array controller to the device. The xor data received in step 5 is sent as the intermediate data.
7) The device reads the requested data from its own medium. The device performs an xor operation between the intermediate data and its own data. The resulting xor data is retained for the next step.
8) The device becomes a temporary initiator and sends a READ command to the other device included in the regenerate parameter list. That target device transfers the requested data to the temporary initiator. The temporary initiator performs an xor operation between the xor data from step 7 and the read data received in this step. The last xor data result is retained in the temporary initiator's buffer. The temporary initiator returns completion status for the REGENERATE command to the array controller when this data is available.
9) An XDREAD command is sent from the array controller to the temporary initiator device. This transfers the regenerated data from the device to the array controller.

1.3.3 The Rebuild operation

The rebuild operation will always involve five valid devices and the replacement device where three valid devices are in one domain (referred to as domain A) and two valid devices and the replacement device are in the other domain (referred to as domain B). The sequence is as follows (note: since xor operands are commutable the xor order is irrelevant, and the order of steps 2 and 3 is interchangable):

1) A REGENERATE command is sent to a device in domain A. This transfers the regenerate parameter list (containing the other two valid devices and data extents) from the array controller to the device.
2) The device reads the requested data from its own medium. The device retains this data for a future xor operation.
3) The device becomes a temporary initiator and sends a READ command to another device included in the regenerate parameter list. That device transfers the requested data to the temporary initiator. The temporary initiator performs an xor operation between the read data from the previous step and the read data received in this step. The resulting xor data is retained for the next step.
4) Step 3 is repeated until all devices listed in the regenerate parameter list have been accessed. The last xor data result is retained in the temporary initiator's buffer. The temporary initiator returns completion status for the REGENERATE command to the array controller when this data is available.
5) An XDREAD command is sent from the array controller to the temporary initiator device. This transfers the partially regenerated data from the device to the array controller.
6) A REBUILD command with the intermediate data bit set is sent to the replacement device in domain B. This transfers the rebuild parameter list (containing the two valid devices and data extents) from the array controller to the device. The xor data received in step 5 is sent as the intermediate data.
7) The device becomes a temporary initiator and sends a READ command to a device included in the rebuild parameter list. That target device transfers the requested data to the temporary initiator. The temporary initiator performs an xor operation between the intermediate data and the read data received in this step. The resulting xor data is retained for the next step.
8) The temporary initiator sends a READ command to the other device included in the rebuild parameter list. That device transfers the requested data to the temporary initiator. The temporary initiator performs an xor operation between the read data from the previous step and the read data received in this step.
9) The last xor data result is written to the replacement device's medium, then the device returns completion status for the REBUILD command to the array controller.

1.4 Additional Array Subsystem Considerations

This clause lists considerations that apply to any array subsystem, but describes how use of the xor commands may affect handling of those situations.

1.4.1 Buffer Full status handling

When the array controller sends an XDWRITE or REGENERATE command to a device, the device has an obligation to retain the resulting xor data until the array controller issues a matching XDREAD command to retrieve the data. This locks up part or all (depending on the size of the device's buffer and the size of the xor data block) of the device's buffer space. When all of the device's buffer is allocated for xor data, it may not be able to accept new media access commands other than valid XDREAD commands and it may not even be able to begin execution of commands that are already in the task set.

When the device cannot accept a new command because there is not enough space in the buffer, the device shall terminate that command with a CHECK CONDITION. The sense data shall be set to ILLEGAL REQUEST : BUFFER FULL. When an array controller receives this status, it may issue any matching XDREAD commands needed to satisfy any previous XDWRITE or REGENERATE commands. This will result in buffer space being freed for other commands. If it is a multi-initiator system and the array controller has no XDREAD commands to send, the array controller may assume the buffer space has been allocated to another initiator. The array controller may retry the command in the same manner that a command ending with BUSY status would be retried.

The array controller may use command linking to avoid a Buffer Full condition. For example, an array controller supervised update write operation would consist of an XDWRITE command linked to an XDREAD command. This prevents the device from working on any other commands until after the xor data has been transferred to the array controller.

1.4.2 Access to an inconsistent stripe

When the array controller begins an update write to a device, the data in that device has been updated when the status is returned for the command. Until the device containing check data has been updated, however, the associated stripe in the redundancy group is not consistent (e.g. performing an xor operation on the protected data does not produce the check data). The array controller must keep track of this window of inconsistency and make sure that a regenerate or rebuild operation for any data extent within the stripe is not attempted until after the device containing check data has been updated (making the stripe consistent again). For multi-initiator systems, this problem may be more severe because each array controller must make sure that the other array controller is not writing to a stripe that it is regenerating or rebuilding. This coordination between array controllers is system specific and is not addressed by this standard. The following list identifies cases where an array controller must be careful to prevent data corruption due to a temporarily inconsistent stripe.

a) When an XDWRITE command has been issued and completed, the device containing protected data has been updated but the device containing check data has not. The stripe will be inconsistent until the XPWRITE command to the device containing check data returns completion status.
b) When an XDWRITE EXTENDED command has been issued, the device containing protected data and the device containing check data will be updated at different times during the course of the command. The stripe should be treated as inconsistent between the time the array controller issues the XDWRITE EXTENDED command and the completion status for the command is received.

c) Any time a regenerate or rebuild operation is in progress for a given stripe, update writes to that stripe should be avoided.

1.4.3 Error handling considerations

If any of the xor commands end with CHECK CONDITION status and an unrecovered error is indicated, an inconsistent stripe may result. It is up to the array controller to identify the failing device and extent of the failure, then limit access to the inconsistent stripe accordingly. In the case of third party xor operations the failing device may be a device containing protected data or a device containing check data. The array controller may be able to identify the failing device from the resulting sense data, or it may have to access the devices directly to determine the health of the affected devices. The recovery procedures that the array controller may implement are not addressed by this standard.

1.4.4 Xor data retention requirements

The following shall remove the requirement of a target to retain data awaiting retrieval by an XDREAD command: XDREAD command, Target Reset, power cycle, Clear Task Set, Abort Task, Abort Task Set.

2. Xor Operation SCSI Commands and Mode Pages

2.1 REBUILD Command

The REBUILD command (see table 1) requests that the target write to the medium the xor data generated from the specified source devices. The target, acting as a temporary initiator, issues READ commands to retrieve the specified data.

Table 1 - REBUILD COMMAND

Byte/Bit	7	6	5	4	3	2	1	0
0	Operation code (81h)							
1	Reserved			DPO	FUA	IntData	Port control	
2	MSB							
3	Logical block address							
4								
5								LSB
6	MSB							
7	Rebuild length							
8								
9								LSB
10	MSB							
11	Parameter list length							
12								
13								LSB
14	Reserved							
15	Control							

Note: The target that receives the REBUILD command is not one of the source devices. If only one source is specified, then an xor operation does not occur. This case can occur in disk mirroring applications.

If the command terminates with CHECK CONDITION status the sense data shall contain the logical block address of the failed block with the lowest logical block address. All logical blocks affected by the command and having a logical block address lower than that of the reported failing block shall have been rebuilt and written to the medium.

If the intermediate data (IntData) bit is set to zero, then intermediate data is not sent with the rebuild parameter list. If the bit is set to one, the rebuild parameter list includes intermediate data. The length of the intermediate data can be calculated by multiplying the rebuild length times the block size. This data shall be treated as an additional source, and an xor operation performed with it and the data from the specified sources.

The logical block address field specifies the starting logical block address at which the target shall write the xor result data on its own medium.

The rebuild length field specifies the number of blocks to be written to the medium. It also specifies the number of blocks that are read from each source.

The parameter list length field specifies the length in bytes of the parameter list that shall be transferred from the initiator to the target (see table 3).

The port control field is defined in table 2. If the port control field has a value of 01b and the target is not a multiple port device the command shall be terminated with a CHECK CONDITION status. The sense data shall be set to ILLEGAL REQUEST : INVALID FIELD IN CDB.

Table 2 - Port control field.

Value	Description
00	The target transfers the data using the same port that received the command.
01	The target transfers the data using a different port than that which received the command.
10	The target transfers the data using one port of the targets choice.
11	The target transfers the data using one or more ports of the targets choice.

The REBUILD parameter data is described in table 3.

Table 3 - REBUILD and REGENERATE parameter data

Byte/Bit	7	6	5	4	3	2	1	0
0	Number of source descriptors (x)							
1	Reserved							
2	Source descriptor/pad length (MSB)							
3	Source descriptor/pad length (LSB)							
	Source descriptor(s) (if any)							
4 19	Source descriptor (first)							
	. . .							
16x - 12 16x + 3	Source descriptor (last)							
16x + 4	Pad, if any (Length y)							
16x+y+3								
16x+y+4	MSB							
	Intermediate data, if any (Length z)							
16x+y+z+3								LSB

The number of source descriptors field indicates the number of source descriptors in the parameter data.

The source descriptor/pad length specifies the sum of the lengths in bytes of all of the source descriptors and the pad.

The source descriptors identify the source device target identifiers and starting logical block addresses on those devices for the regenerate or rebuild operation. See table 4 for the source descriptor format.

The pad field contains invalid data and shall be ignored.

Note: The pad field is included to accommodate initiators which require the intermediate data to be aligned on a particular memory boundary.

The intermediate data field contains data that shall be used in the xor operation with the data from the specified source devices. The length of the data is equal to the rebuild/regenerate length multiplied by the block size.

Table 4 - Source descriptor format

Byte/Bit	7	6	5	4	3	2	1	0
0	MSB							
				Source device address				
7								LSB
8	MSB							
				Reserved				
11								LSB
12	MSB							
				Source starting logical block address				
15								LSB

The source device address field specifies a SAM compliant target identifier of a device that is a data source.

The source starting logical block address field indicates the starting logical block address to use when reading data from the source specified in the source device address field.

2.2 REGENERATE command

The REGENERATE command (see table 5) requests that the target write to the buffer the xor data generated from its own medium and the specified source devices. The target, acting as a temporary initiator, issues READ commands to retrieve the specified data.

Table 5 - REGENERATE command

Byte/Bit	7	6	5	4	3	2	1	0
0	Operation code (82h)							
1	Reserved			DPO	FUA	IntData	Port control	
2	MSB							
3	Logical block address							
4								
5								LSB
6	MSB							
7	Regenerate length							
8								
9								LSB
10	MSB							
11	Parameter list length							
12								
13								LSB
14	Reserved							
15	Control							

The resulting xor data is retained in the target's buffer until it is retrieved by an XDREAD command with a starting logical block address and transfer length that match the logical block address and regenerate length of this command.

See 2.1 for a definition of the IntData bit.

See table 2 for a definition of the port control field.

The logical block address field specifies the starting logical block address for the target to read data from its own medium. This data is a source for the regenerate operation.

The regenerate length field indicates the length in logical blocks of the resulting xor data. It also specifies the length in logical blocks that is transferred from each of the specified sources.

The parameter data for the REGENERATE command is defined in table 3. This parameter data describes the other devices that will be sources for the regenerate operation. The target receiving the REGENERATE command is implicitly a source, and is not included in the parameter data.

2.3 XDREAD command

The XDREAD command (see table 6) requests that the target transfer to the initiator the xor data generated by an XDWRITE or REGENERATE command.

Table 6 - XDREAD command

Byte/Bit	7	6	5	4	3	2	1	0
0	Operation code (52h)							
1	Reserved							
2	MSB							
3	Logical block address							
4								
5								LSB
6	Reserved							
7	Transfer length							
8								
9	Control							

The xor data transferred is identified by logical block address and transfer length that are the same as those specified in a prior XDWRITE or REGENERATE command. If a match is not found the command is terminated with a CHECK CONDITION status. The sense data is set to ILLEGAL REQUEST : INVALID FIELD IN CDB.

2.4 XDWRITE command

The XDWRITE command (see table 7) requests that the target xor the data transferred with the data on the medium. The resulting xor data is stored in the target's buffer. The disposition of the data transferred from the initiator is controlled by the disable write bit.

Table 7 - XDWRITE command

Byte/Bit	7	6	5	4	3	2	1	0
0	Operation code (50h)							
1	Reserved	Reserved	Reserved	DPO	FUA	Disable write	Reserved	
2	MSB							
3	Logical block address							
4								
5								LSB
6	Reserved							
7	Transfer length							
8								
9	Control							

The resulting xor data is retained in the target's buffer until it is retrieved by an XDREAD command with starting logical block address and transfer length fields that match the starting logical block address and transfer length of this command.

The field descriptions are the same as in the XDWRITE EXTENDED command.

2.5 XDWRITE EXTENDED command

The XDWRITE EXTENDED command (see table 8) requests that the target xor the data transferred with disable write bit. The resulting xor data is sent to a secondary device using an XPWRITE command.

Table 8 - XDWRITE EXTENDED command

	7	6		4	3		1	0
0	Operation Code (80h)							
1	Table address	Reserved	Reserved	DPO	FUA	Disable write	Port control	
2	MSB							
3	Logical Block Address							
4								
5								LSB
6	MSB							
7	Secondary logical block address							
8								
9								LSB
10	MSB							
11	Transfer length							
12								
13								LSB
14	Secondary address							
15	Control							

A table address bit of zero indicates that the secondary address field contains the target identifier of the target to which the xor data is transferred. The LUN of the secondary target shall be zero.

Note: If the protocol requires more than one byte for the target identifier and the table address bit is set to zero, the secondary address field specifies the least significant byte of the secondary target identifier - the upper bytes of the secondary target identifier are assumed to be equal to the upper bytes of the target identifier of the XDWRITE EXTENDED target.

A table address bit of one indicates that the secondary address field contains a pointer to a look up table of SAM compliant target identifiers. This look up table is reserved for future definition.

A disable write bit of zero indicates that the data transferred from the initiator shall be written to the medium after the xor operation is complete. A disable write bit of one indicates that the data shall not be written to the medium.

The port control field is defined in table 2. If the port control field has a value of 01b and the target is not a multiple port device the command shall be terminated with a CHECK CONDITION status. The sense data shall be set to ILLEGAL REQUEST : INVALID FIELD IN CDB.

The transfer length field specifies the number of logical blocks that shall be transferred to the XDWRITE EXTENDED target, and to the XPWRITE target.

The xor data transfer to the secondary target is performed using an XPWRITE command. The XPWRITE command is sent to the device specified in the secondary address field. The secondary logical block address field value is placed in the logical block address field of the XPWRITE command. The transfer length field value is placed in the transfer length field of the XPWRITE command. The completion status of the XDWRITE EXTENDED command shall not be returned to the initiator until the completion status of the XPWRITE command has been received.

Note: The xor data transfer to the secondary target may be broken into multiple XPWRITE commands. If this is done, the XDWRITE EXTENDED target will need to calculate the logical block addresses and transfer lengths for the individual XPWRITE commands. Also, the completion status

of the XDWRITE EXTENDED command shall not be returned to the initiator until the completion status of all XPWRITE commands have been received.

RECOVERED ERROR the XDWRITE EXTENDED command shall return CHECK CONDITION status.

XPWRITE command

the medium and then writes the xor data to the medium.

Table 9 - XPWRITE command

	7	6		4	3		1	0
0	Operation code (51h)							
1	Reserved			DPO	FUA	Reserved		
2	MSB							
	Logical block address							
4								
								LSB
6								
7	Transfer length							
9	Control							

data from its medium. It also specifies the starting logical block address at which the xor result data is to be written to the medium.

number of blocks to be written to the medium.

2.7 Xor control mode page

The xor control mode page (see table 10) provides the initiator with the means to obtain or modify certain xor operating parameters of the target.

Table 10 - Xor control mode page

Byte/Bit	7	6	5	4	3	2	1	0
0		Reserved	Page code (10h)					
	Page length (16h)							
2							XorDis	Reserved
	Reserved							
4	Maximum xor write size							
5								
6								
7								
8	MSB							
	Reserved							
10								
								LSB
12	Maximum regenerate size							
13								
15								LSB
	MSB							
17								
18								
19								
20	Reserved							
22	Rebuild delay							

An xor disable (XorDis) bit of zero enables the xor operations within a device. An XorDis bit of one disables the xor operations within a device. If the XorDis bit is set to one and an xor command is sent to

set to ILLEGAL REQUEST: INVALID COMMAND OPERATION CODE.

The maximum xor write size field specifies the maximum transfer length in blocks that the target accepts

The maximum regenerate size field specifies the maximum regenerate length in blocks that the target accepts for the REGENERATE command.

use for READ commands during a rebuild operation. This field does not limit the rebuild size.

The rebuild delay field specifies the minimum time in milliseconds between successive READ

Errors during third party operations

causes the target of that command to generate another command to a secondary device is referred to as a “primary” command (this is a temporary definition for this clause only, and should not be associated

primary command, as a result of that primary command, is referred to as a “secondary” command. (e.g. The target of an XDWRITE EXTENDED primary command generates an XPWRITE secondary

commands.) Two classes of exception conditions may occur during these commands. One class consists

secondary command. Either or both of these classes of exception may occur during a third party

2.8.1 Primary errors - errors resulting directly from the primary command

the primary command (primary target) and are not due to the failure of a resulting secondary command.

the primary target to continue operating, and parity errors while transferring the primary command, data,

1)
2)

2.8.2 Secondary errors - errors resulting from the secondary command

command. The sense data for such errors shall be passed to the initiator of the primary command in the

If the primary target detects the exception (i.e. by some means other than receiving CHECK

1) terminate the primary command with CHECK CONDITION status.
2) set the sense key to ABORTED COMMAND if there are no primary errors to report. Otherwise, the sense key shall be set according to the primary error;
3) set the first byte of the command specific information field of the sense data to the starting byte number, relative to the first byte of sense data, of an area that contains the primary target’s sense data for the secondary error. A zero value in this byte indicates no secondary error has been detected by the primary target. The secondary sense data shall be built in the standard sense data format as defined for the REQUEST SENSE command;
4) in the case of a REBUILD or REGENERATE (primary) command, set the third byte of the command specific information field of the sense data to an index value indicating the target identifier of the failing secondary target. This value shall be an index into the source descriptor entries of the parameter data of the primary command, and shall point to the entry containing the target identifier of the failing device; 0 points to the first entry, 1 points to the second entry, etc. This byte is invalid and shall be ignored if the primary command is not a REBUILD or REGENERATE.

If the secondary target detects the exception, the primary target receives CHECK CONDITION status from the secondary target. The primary target shall recover the sense data associated with the exception condition, clear any ACA associated with the CHECK CONDITION status, and shall:

1) terminate the primary command with CHECK CONDITION status;
2) set the sense key to ABORTED COMMAND if there are no primary errors to report. Otherwise, the sense key shall be set according to the primary error;
3) set the second byte of the command specific information field of the sense data to the starting byte number, relative to the first byte of sense data, of an area that contains (unchanged) the

secondary target's status byte followed by its sense data. A zero value in this byte indicates no secondary error has been reported by the secondary target;

4) in the case of a REBUILD or REGENERATE (primary) command, set the third byte of the command specific information field of the sense data to an index value indicating the target identifier of the failing secondary target. This value shall be an index into the source descriptor entries of the parameter data of the primary command, and shall point to the entry containing the target identifier of the failing device; 0 points to the first entry, 1 points to the second entry, etc. This byte is invalid and shall be ignored if the primary command is not a REBUILD or REGENERATE.

For a given primary command, if errors are generated by more than one secondary command, the sense data shall contain error information for the secondary error first realized by the primary target.

IMPLEMENTOR'S NOTE: Since, for secondary errors, the sense key is set to ABORTED COMMAND only if there are no primary errors to report (see items 2 above), the first and second bytes of the command specific information field should always be checked, even when the sense key is set to a value other than ABORTED COMMAND, to determine if any secondary errors have occurred.

Note: It may be possible for all three of the above error types to occur during the same third party operation. If this happens, there will be three unique pieces of error information contained in the sense data - one for the primary error (starting at byte 0), and two for the secondary errors (in the additional sense bytes).

Annex A
(informative)
Xor command examples

This annex provides xor command examples in various redundancy group configurations.

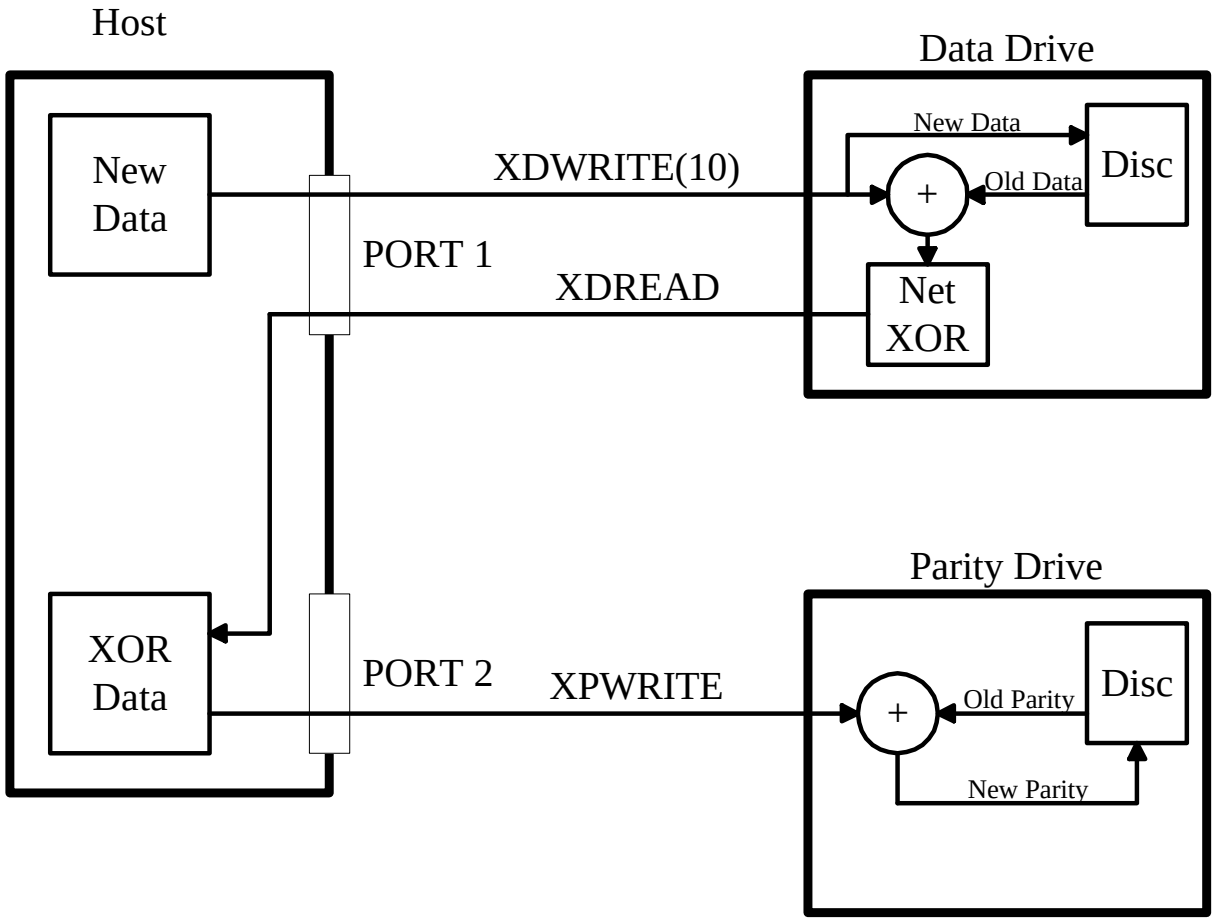
Array Controller Supervised Xor Operations

Update Write operation

This example illustrates an array controller supervised Read-Modify-Write operation. A host, a data disk drive (protected data), and a parity disk drive (check data) are used. The data and parity drives are on separate SCSI busses, and incapable of peer to peer interaction. Three SCSI commands are used: XDWRITE, XDREAD, and XPWRITE.

The host begins by sending the data drive new data using an XDWRITE command. It also sends the parity drive the XPWRITE command (command phase only at this point - this gets the parity drive started reading old parity from its disk into its buffer). The data drive reads old data from its disk, xors the old data with the new data from the host, stores the xor'd data in its buffer, and writes the new data from the host to its disk. The host reads the xor'd data by sending the data drive an XDREAD command.

The host sends the xor'd data to the parity drive during the data phase of the already issued XPWRITE command. The parity drive xors this data with the old parity data in its buffer. The resulting new parity data is written to the disk.

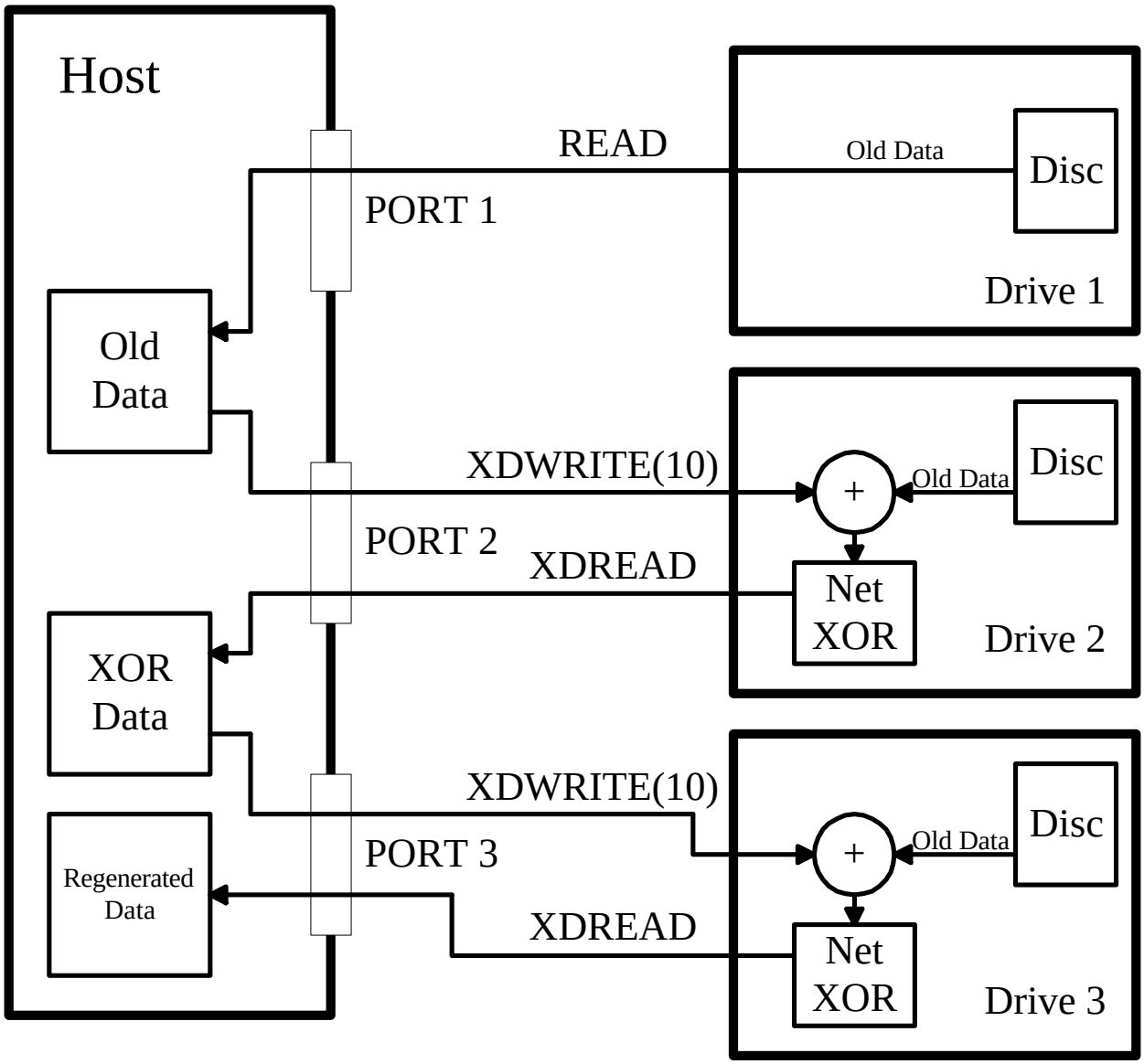


Regenerate operation

This example illustrates an array controller supervised regenerate operation. A host serves as the array controller, and three disk drives as the source devices. All three drives are on separate SCSI busses, and incapable of peer to peer interaction. Three SCSI commands are used: READ, XDWRITE, and XDREAD.

The host begins by sending drive 1 a READ command. The data received from this drive is sent by the host to drive 2 using an XDWRITE command with the disable write bit set. Drive 2 reads old data from its disk, xors this old data with the data sent from the host, and stores the result in its buffer. The host retrieves the xor'd data by sending drive 2 an XDREAD command. The host issues the XDWRITE and XDREAD commands in a likewise manner to drive 3.

The xor'd data from drive 3 is the regenerated data.

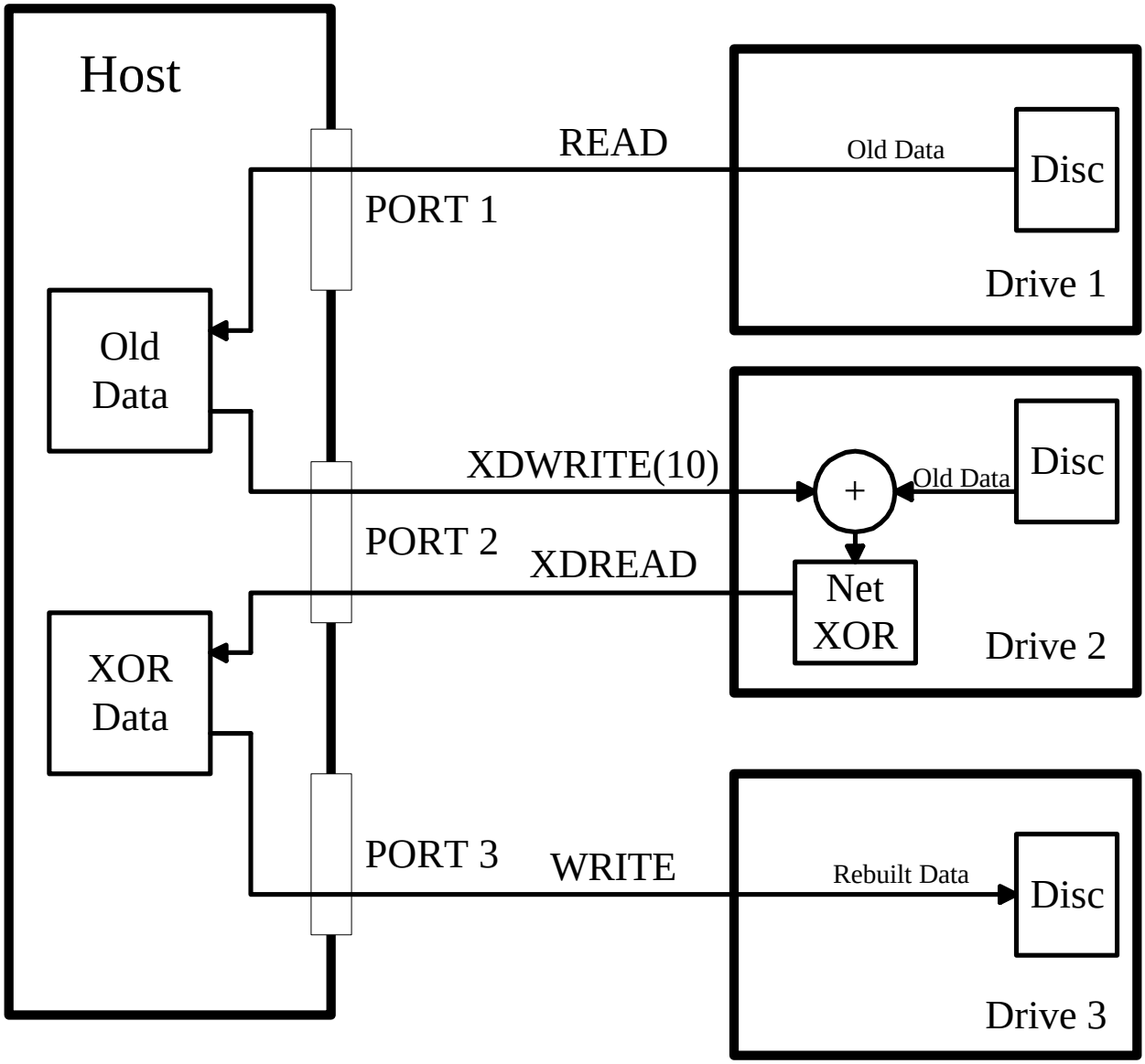


Rebuild operation

This example illustrates an array controller supervised rebuild operation. A host serves as the array controller, two disk drives as the source devices, and one disk drive as the rebuild device. All three drives are on separate SCSI busses, and incapable of peer to peer interaction. Four SCSI commands are used: READ, XDWRITE, XDREAD, and WRITE.

The host begins by sending the first source drive (drive 1) a READ command. The data received from this drive is sent by the host to the second source drive (drive 2) using an XDWRITE command with the disable write bit set. The second source drive reads old data from its disk, xors this old data with the data sent from the host, and stores the result in its buffer. The host retrieves this xor'd data by sending an XDREAD command to the second source drive.

The xor'd data from the second source drive is the "Rebuilt" data, and is sent to the rebuild drive (drive 3) using a WRITE command.



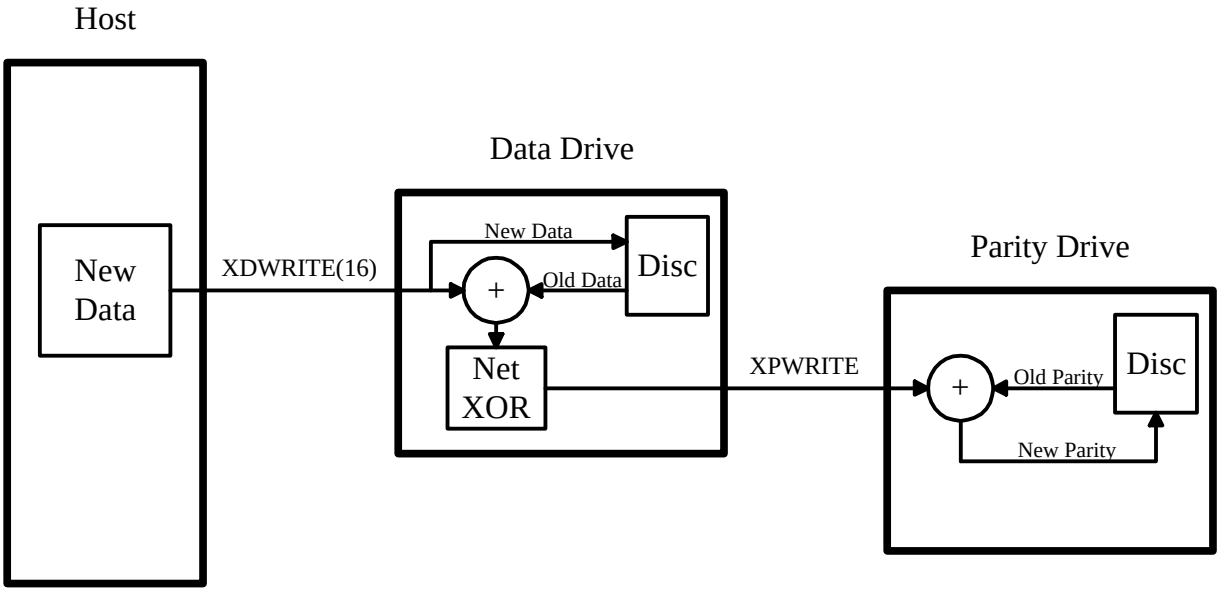
Third Party Xor Operations

Update Write operation

This example illustrates a third party Read-Modify-Write operation. A host, a data disk drive (protected data), and a parity disk drive (check data) are used. The data and parity drives are on the same loop, and are capable of peer to peer interaction. Two SCSI commands are used: XDWRITE EXTENDED and XPWRITE.

The host begins by sending the data drive new data using an XDWRITE EXTENDED command. The data drive takes on the role of initiator and sends an XPWRITE command to the parity drive (command portion only, no data at this point - this gets the parity drive started reading old parity from the disk into its buffer). The data drive reads old data from its own disk, xors this old data with the new data from the host, sends the xor'd data to the parity drive (data transfer portion of XPWRITE), and writes the new data from the host to its disk.

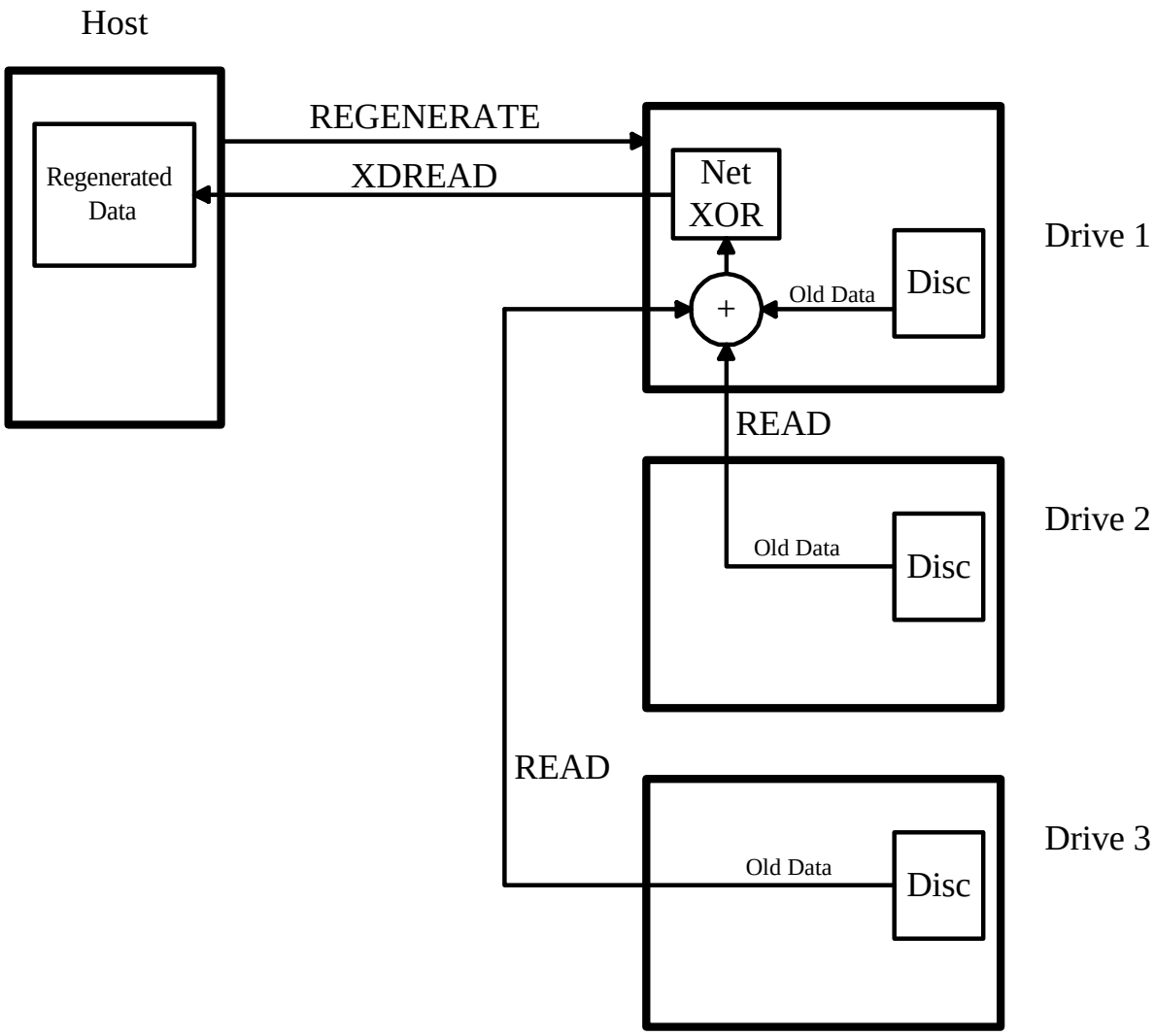
The parity drive receives the xor'd data from the data drive and xors this data with the old parity data in its buffer. The resulting new parity is written to the disk.



Regenerate operation

This example illustrates a third party regenerate operation. A host is used, along with three disk drives as source devices. All three drives are on the same loop, and are capable of peer to peer interaction. Three SCSI commands are used: REGENERATE, READ, and XDREAD.

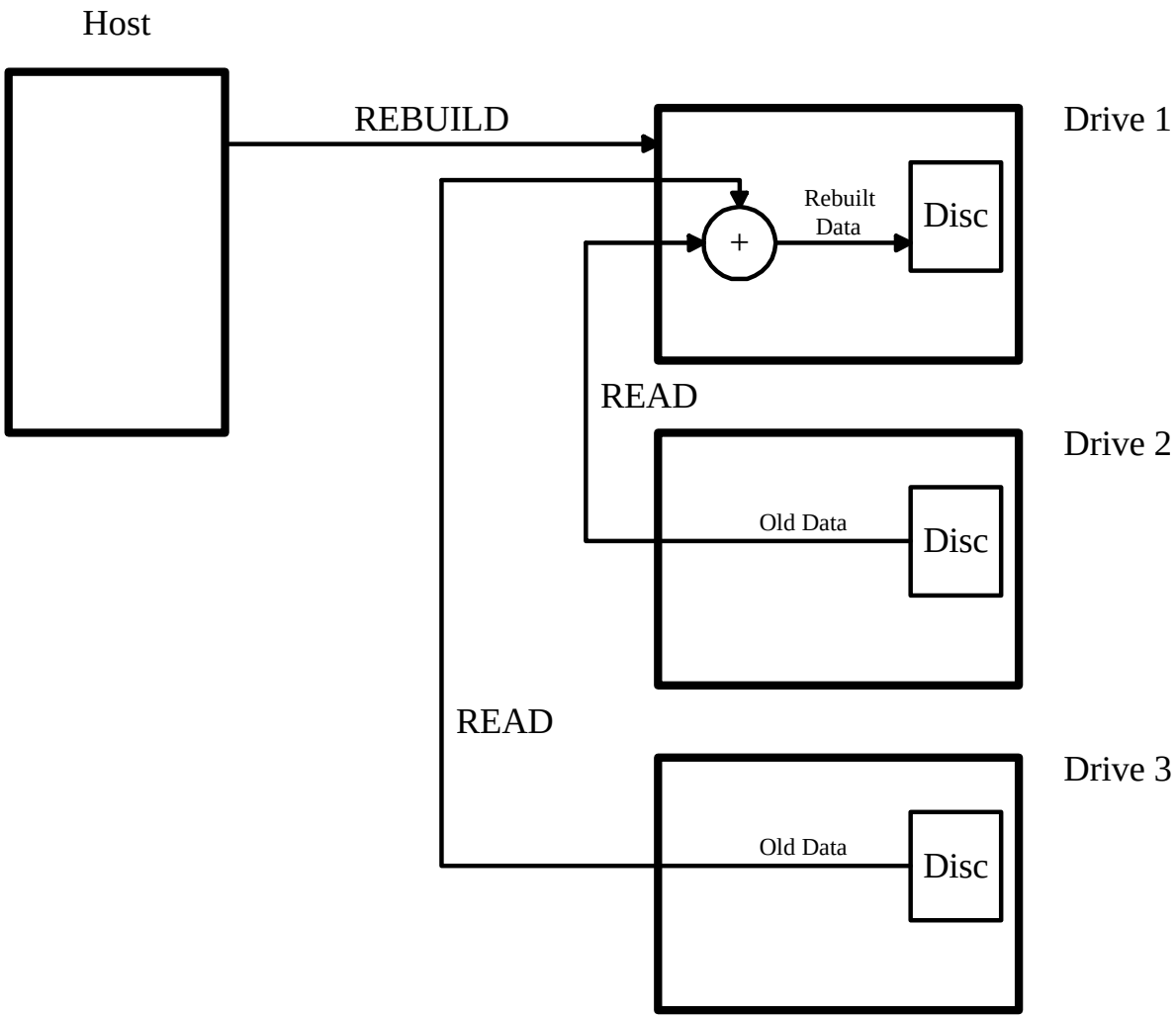
The host begins by sending drive 1 a REGENERATE command. Drive 1 takes on the role of initiator and sends READ commands to drives 2 and 3. It also concurrently reads data from its own disk. The data from all three drives is xor'd and written to the buffer in drive 1. The host retrieves this regenerated data by sending drive 1 an XDREAD command.



Rebuild operation

This example illustrates a third party rebuild operation. The example uses a host, two disk drives as the source devices, and one disk drive as the rebuild device. All three drives are on the same loop, and are capable of peer to peer interaction. Two SCSI commands are used: REBUILD and READ.

The host begins by sending the REBUILD command to the rebuild drive (drive 1). That drive takes on the role of initiator and issues READ commands to the source drives (drives 2 and 3). The data from the source drives is xor'd in the rebuild drive and written to its disk.



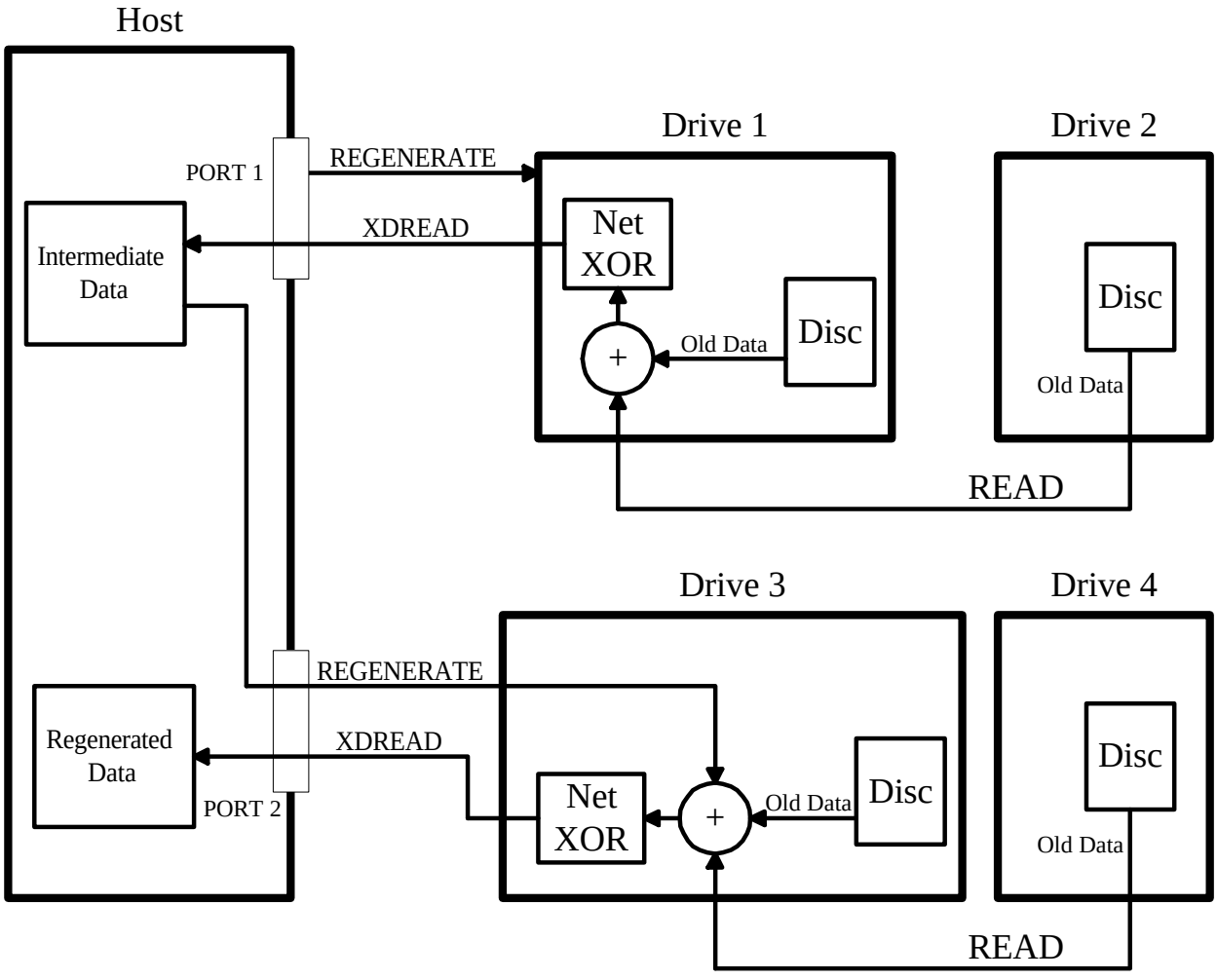
Hybrid Subsystem Xor Operations

Regenerate operation

This example illustrates a hybrid subsystem regenerate operation. A host is used, along with four disk drives as source devices. Two of the drives are on one loop, and are capable of peer to peer interaction between themselves. The other two drives are on a different loop and are also capable of peer to peer interaction between themselves. Three SCSI commands are used: REGENERATE, READ, and XDREAD.

The host begins by sending drive 1 a REGENERATE command. Drive 1 takes on the role of initiator and sends a READ command to drive 2. It also concurrently reads data from its own disk. The data from the two drives is xor'd and written to the buffer in drive 1. The host retrieves this intermediate data by sending drive 1 an XDREAD command.

The host then sends drive 3 a REGENERATE command with the intermediate data obtained from drive 1. Drive 3 takes on the role of initiator and sends a READ command to drive 4. It also concurrently reads data from its own disk. The drive 3 data is xor'd with the drive 4 data, and the result xor'd with the intermediate data. The final xor data is written to the buffer in drive 3. The host retrieves this regenerated data by sending drive 3 an XDREAD command.

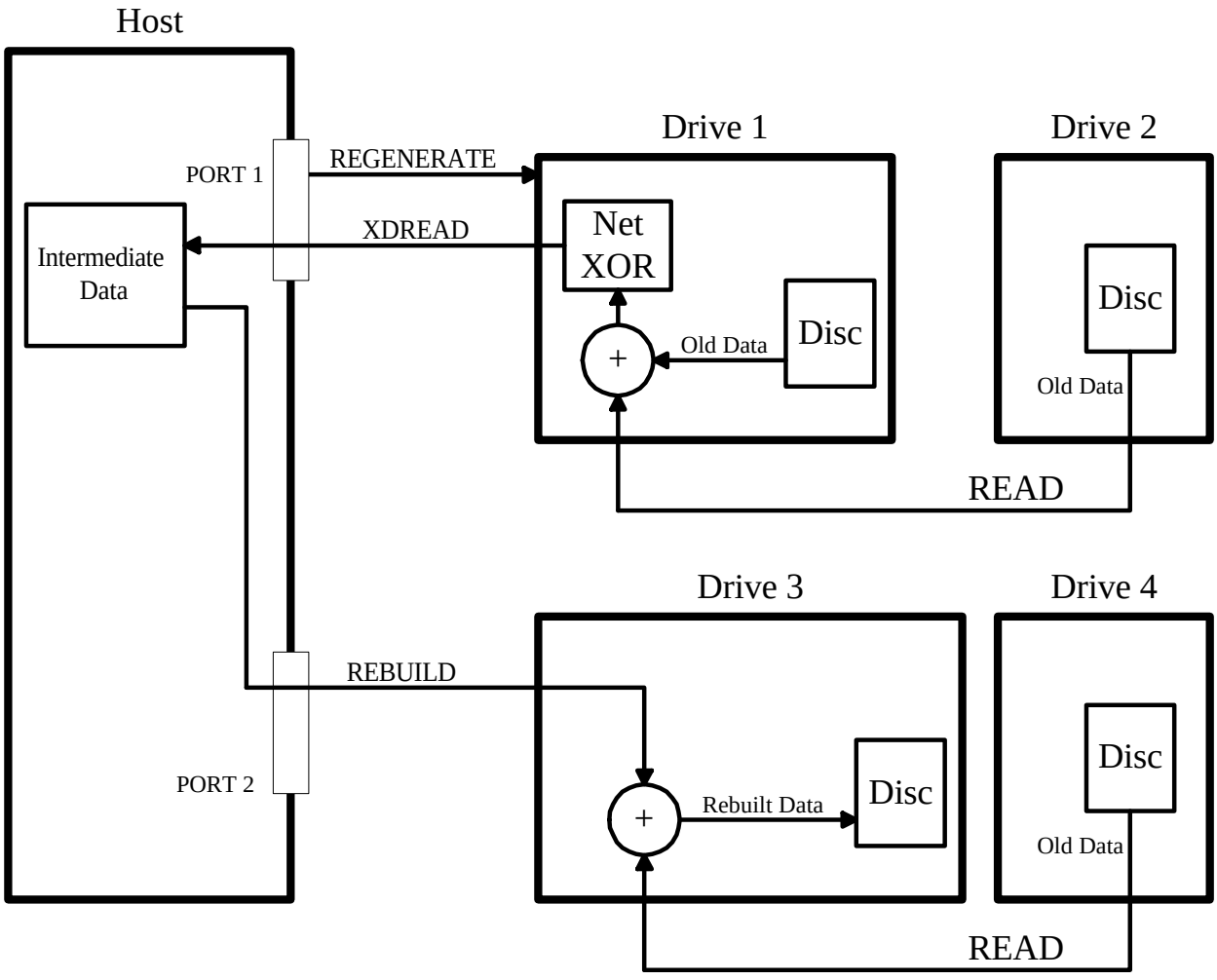


Rebuild operation

This example illustrates a hybrid subsystem rebuild operation. A host is used, along with three disk drives as source devices, and one disk drive as the rebuild device. Two of the drives are on one loop, and are capable of peer to peer interaction between themselves. The other two drives are on a different loop and are also capable of peer to peer interaction between themselves. Four SCSI commands are used: REGENERATE, READ, XDREAD, and REBUILD.

The host begins by sending drive 1 (a source device) a REGENERATE command. Drive 1 takes on the role of initiator and sends a READ command to drive 2 (a source device). It also concurrently reads data from its own disk. The data from the two drives is xor'd and written to the buffer in drive 1. The host retrieves this intermediate data by sending drive 1 an XDREAD command.

The host then sends drive 3 (rebuild device) a REBUILD command with the intermediate data obtained from drive 1. Drive 3 takes on the role of initiator and sends a READ command to drive 4 (a source device). The intermediate data is xor'd with the data from drive 4. The resulting rebuilt data is written to the disk in drive 3.



Annex B
(informative)
Definitions

This annex provides definitions for terms used in this document.

check data: Information contained within a redundancy group that allows lost or destroyed user data to be recreated. (See SCC document)

domain: An I/O system consisting of a set of SCSI devices that interact with one another by means of a service delivery subsystem. (See SAM document)

protected space: The portion of a redundancy group that does not contain check data. (See SCC document)