Document:	T10/02-320r0	Date:	August 22, 2002
To:	T10 Committee Membership		
From:	Edward A. Gardner, Ophidian Designs		
Subject:	SAS Simplified Arbitration		

This proposal presents an alternate approach to SAS arbitration. It is not a complete proposal with detailed specification changes. Rather it is a tutorial on how the alternate approach would work. If there is interest in pursuing this further, I will prepare a complete detailed proposal (in cooperation with any other interested parties) for the September SAS meeting.

The concept is that arbitration is implemented entirely within expanders. Expanders are solely responsible for fairness, deadlock avoidance, etc. Typically expanders will provide some form of round-robin arbitration. However they are free to modify this (e.g. give priority to certain ports or open requests) to improve performance. The end of this proposal includes some thoughts on this subject.

The PATHWAY BLOCK COUNT, SCALE and ARBITRATION WAIT TIME fields serve no purpose in this proposal. They are eliminated from the OPEN address frame. No SAS port or device need maintain any similar values.

The use of OPEN REJECT(PATHWAY BLOCKED) is closely tied to the current arbitration algorithm. This alternate approach does not use it. If adopted, this proposal will either eliminate that form of OPEN REJECT or redefine it in a substantially changed form. This proposal assumes that OPEN REJECT(PATHWAY BLOCKED) will be eliminated. The unqualified name OPEN REJECT should be interpreted as all current forms of OPEN REJECT except OPEN REJECT(PATHWAY BLOCKED)

1 Open-pending condition

Expanders maintain an open-pending condition flag for each of their links (i.e. each PHY). The open-pending condition indicates that the remote end of the link (e.g. the end port connected to that link) is trying to open a connection. The open-pending condition persists across connections established by opens sent by the expander.

Rules for expanders:

1. An expander shall set the open-pending condition flag associated with one of its links whenever it receives an OPEN request on that link.
2. An expander shall clear the open-pending condition flag associated with one of its links whenever it sends an OPEN-ACCEPT or OPEN-REJECT on that link.
3. An expander should clear the open-pending condition flag associated with one of its links whenever the SAS Expander state machine (figure 60 in SAS-r01) remains in the XL0:Idle state for more than a few microseconds (exact value tbd).

Any device (e.g. an end port) that is trying to open a connection on a link shall send an OPEN request promptly upon the link becoming idle. That is, if an end port is trying to open a connection, it sends an OPEN request immediately upon receiving or sending a CLS or BREAK.

2 Single expander operation (simplified algorithm)

The open-pending condition flags indicate the links (expander PHYs) whose attached devices are attempting to open a connection. The expander selects those links one at a time, either round-robin or whatever other order it chooses. The selected link is given priority for all resources that it needs to satisfy its OPEN request. Requests from other links may be processed only if they do not interfere with satisfying the selected link's request.

An expander follows an algorithm such as the following:

1. Select the next link whose open-pending condition flag is set. The expander shall not send any OPEN requests on the selected link.

2. If an active or pending connection exists on the selected link, wait until it is closed. This circumstance can occur if the remote port on the link sent an OPEN request (setting the open-pending condition flag), then the expander sent an OPEN request on that link (overriding the remote port's OPEN request), and finally the expander selected the link.
3. If an OPEN request has not already been received on the selected link, wait for it to be received. If no OPEN request is received within a few microseconds, clear the link's open-pending condition flag and return to step 1.
4. Reserve whatever expander pathway resources are necessary to process the selected link's OPEN request. Requests from links that would delay completion of the selected link's request shall not be processed.
5. Wait until the destination link is free, then forward the selected link's OPEN request on the destination link.
6. Return to step 1.

When two OPEN requests pass on the same link, the OPEN request sent by the expander overrides the OPEN request sent by the end port. The OPEN request sent by the end port is discarded, although the link's open-pending condition flag remains set. The end port is expected to re-send the same OPEN request when the link becomes idle.

It is easy to show that this is deadlock free. An end port that needs a connection sends an OPEN request when its link becomes idle. This sets the open-pending condition flag. If its OPEN request is overridden by a request from the expander, the end port re-sends its OPEN request following that connection. This ensures that the open-pending condition flag remains set. Eventually the expander selects the link. That guarantees that the expander will not send an overriding request, and will reserve all resources necessary to satisfy the selected end port's request. When the request is satisfied (connected to the destination port), an OPEN-ACCEPT or OPEN-REJECT is returned and the open-pending condition flag is cleared.

Access to expander resources can be as fair or unfair as the expander chooses. Round-robin arbitration is among the simplest that might be implemented. The expander could also choose more complex algorithms, e.g. giving preference to initiators or prioritizing links by activity.

3 Multi-expander operation

Multi-expander configurations operate by extending the algorithm described in the previous section.

Each SAS device knows its local device type as reported in the IDENTIFY address frame:

00 End device

01 Edge expander

02 Fanout expander

Each SAS device also knows the device type attached at the remote end of each of its links. This section assumes that all links connect different device types. That is, the local device type is different from each link's attached device type. The next section extends this to links between the same device types.

As with the single expander case, expanders maintain open-pending condition flags for all links. Each expander selects links whose open-pending condition flag is set in round-robin or some other order. Each expander gives priority for internal pathway resources to the selected link's OPEN request, with a modification.

If two OPEN requests pass on the same link, the request sent by the numerically larger device type overrides the request sent by the numerically smaller device type. OPEN requests sent by expanders override those sent by end devices. OPEN requests sent by a fanout expander override those sent by edge expanders.

The rule that priority is given to the selected link's request is modified as follows. If the selected link's request is overridden, the overriding request is given priority. This may occur if an edge

expander selects a link to an end device. The end device sends an OPEN request that is routed to a fanout expander. A request received from the fanout expander will override the end device's request. Since completion of the overriding request from the fanout expander is necessary before the selected end device's request can be completed, priority is given to the overriding request.

The proof that this is deadlock free is a hierarchical version of the single expander case. Requests that traverse a single expander are the same as the single expander case, so we will only consider requests that traverse multiple expanders. The end device originating an OPEN request will eventually become selected. That request will be sent and re-sent to the fanout expander until the fanout expander selects it. It is then sent to the destination edge expander, where it either becomes selected or overrides a selected request and completes.

4 Virtual expanders

Links between identical device types are handles by, in effect, creating a virtual expander in the middle. The virtual expander acts as if it had two links that it selects alternately.

The devices on a link between identical device types keep track of the direction of the previous OPEN request to complete (a response of OPEN-ACCEPT or OPEN-REJECT). An OPEN request in the opposite direction as the previous request overrides a request in the same direction as the previous request.

5 Extensibility

This technique can be extended to cover an arbitrary depth hierarchical structure. The bottom of the hierarchy contains end devices. Above them are edge expanders, followed by fanout expanders, super expanders, hyper expanders, galactic expanders, etc. The requirement is that all valid pathways consist of a walk up the hierarchy followed by a walk down the hierarchy. During the walk up, each link traversal must be to a device that is higher in the hierarchy, during the walk down each link traversal must be to a device that is lower. A single traversal between equal devices at the top is permitted by the virtual expander trick. Bounces up and down the hierarchy are not allowed.

6 Path blocked handling

Consider an environment where multiple requestors request connections to the same destination. For example, multiple initiators contending for a single target, or multiple targets returning data to the same initiator.

One requestor's connection will be established. Another requestor will be selected and will be connected to the destination next, but not until the first requestor's connection is closed. Other requestors will have to wait in turn for access. The waiting time for a connection to be established could be many milliseconds or longer.

SAS specifies a 1 millisecond timeout for connection requests. In such situations the expander(s) will periodically send an AIP primitive to refresh the timer of the blocked requests.

If any requestor, particularly the selected requestor, were to abandon its request and substitute another, it might lose its place in the arbitration sequence. This could result in a live-lock situation. I believe a similar live-lock exists in the current arbitration algorithm. If needed for performance reasons, we could easily invent a method to substitute requests while maintaining the previous request's position in arbitration. I would prefer to defer this until experience or simulation demonstrates it is necessary.

7 Arbitration preference hints

System performance can often be enhanced through the use of deliberately unfair arbitration in certain circumstances. For example, giving preference to issuing commands, particularly to idle devices. Or giving preference to targets that simply need to return status over those that need to first transfer data and then return status.

In SAS-2 we might wish include some hints in the OPEN address frame that describe the purpose of the OPEN request. Expanders could use this information to give preference to certain requests over others.

The OPEN request already indicates whether it is coming from an initiator or a target. Consider a defining four bits whose meaning is qualified by the initiator / target bit. When sent by an initiator, the bits might indicate:

1. Set if one or more commands will be sent to a LUN on which the initiator has no commands outstanding (i.e. commands sent to a device that is likely to be idle).
2. Set if one or more READ commands will be sent.
3. Set if one or more WRITE commands will be sent.
4. Set if write data will be sent.

When sent by a target, the bits might indicate:

1. Set if one or more RESPONSE frames will be sent.
2. Set if read data will be returned.
3. Set if write data will be requested.
4. Reserved.