Additional SAS-1.1 Expander Issues

September 28, 2005

Issue 5 added 10/21/05

Revised 10/28/05 after discussions with Tim Hoglund

Authors:
Alexander Amezquita (alex@expertio.com) of Expert I/O
Craig Stoops (craig@expertio.com) of Expert I/O
www.expertio.com

Issue 1: SAS Expander additional qualification in Arb Won

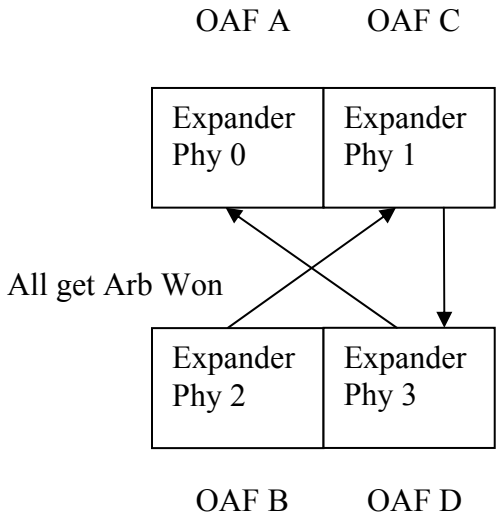
After discussion with Tim Hoglund, we determined this is not an issue pending resolution for additional qualification in Arb Lost described in Issue 4.

Problem

An additional qualification is necessary for an expander link to receive an Arb Won message from the expander function. Consider the following scenario.

Expander Phys 0, 1, 2, and 3 receive open address frames (OAF) A, C, B, and D respectively, where by arbitration rules D > C > B > A. Oaf A received on expander phy 0 requests a connection to expander phy 2, Oaf C received on expander phy 1 requests a connection to expander phy 3. OAF B is destined for expander phy 1 and OAF D is destined for expander phy 0.

Example Summary:
OAF A received phy 0 destined for phy 2
OAF C received phy 1 destined for phy 3
OAF B received phy 2 destined for phy 1
OAF D received phy 3 destined for phy 0



The current revision of the SAS-1.1 specification (1r10) details an Arb Won is issued as follows:

The ECM shall generate the Arb Won confirmation when all of the following conditions are met:
a) the connection request maps to an expander phy that:
A) supports the connection rate; and

B) is not reporting a Phy Status (Partial Pathway), Phy Status (Blocked Partial Pathway), or Phy Status (Connection) response, unless that expander phy is arbitrating for the expander phy making this connection request;

b) there are sufficient routing resources to complete the connection request;
c) no higher priority connection requests are present with this expander phy as the destination;and
d) the connection request is chosen as the highest priority connection request in the expander device mapping to the specified destination expander phy.

Assuming there are sufficient routing resources and the connection rates are valid this scenario proves to be problematic. The expander function receives four request path messages from the respective expander phys. The value of phy status during this phase of the request path is such that Condition a) is satisfied for all ports. Condition b) is satisfied based on the assumption that the expander is designed accordingly. Condition c) rules out expander phy 0 to receive an Arb Won message since OAF D > A. Condition d) does not create any further restrictions since each OAF is destined to a unique expander phy.

Thus, since all conditions are met, an Arb Won message is issued to expander phy 3 with OAF D, expander phy 2 with OAF B, and expander phy 1 with OAF C. There are two error cases that need to be considered:

1) Expander phy 1 should not receive an Arb Won as it will cause a collision in the connection between expander phy 0 and 3
2) Expander phy 2 should not receive an Arb Won since OAF B sent to an end device attached to expander phy 1 will be ignored due to C > B. Further, expander phy 1 may have issued AIP due to receiving OAF C, which in turn could result in a hung connection attempt (i.e. an Open Timeout would not occur).

Solution
A recommendation to solve this problem is to add the following clause to the Arb Won qualification.

e) the connection request of this expander phy is higher priority than an outstanding connection request of the destination expander phy.

Issue 2: Phy Status Clarification in XL1:Request_Path

After comments by Robert Elliot and conversation with Tim Hoglund, we understand that the absence of a defined Phy Status, that Phy Status (Ilde) is assumed. Would prefer that a NOTE or comment be added making this assumed behavior more obvious.

Problem
Additional rules need to be included for the value of Phy Status while in the XL1:Request_Path state. Particularly when transitioning from XL3:Open_Confirm_Wait or XL6:Open_Response_Wait.

Consider the following example:

Expander Phy 0 receives OAF A destined for expander phy 1. Expander Phy 0 is given an Arb Won message and forwards OAF A. Shortly thereafter, Expander Phy 1 receives OAF B where B > A in terms of arbitration priority. OAF B is destined for expander phy 2 which causes a backoff retry condition. The Phy Status of both Expander Phy 0 and Expander Phy 1 just prior to the BackOff Retry message was Partial Pathway due to being in XL3:Open_Confirm_Wait and XL6:Open_Response_Wait respectively. Upon transmission of the BackOff Retry message and the reception of the BackOff Retry message, each expander phy transitions to XL1:Request_Path. There is no mechanism described in XL1:Request_Path, XL3:Open_Confirm_Wait, or XL6:Open_Response_Wait to allow the Phy Status to change from Phy Status (Partial Pathway) to Phy Status (Idle).

If we continue with the example and consider an OAF C being received on expander Phy 2 destined for expander Phy 0, we can see we have reached a deadlock. This is because an Arb Won message cannot be issued to expander phy 2 since the Phy Status of expander phy 0 is Partial Pathway. Further, if we assume OAF C > OAF B in terms of arbitration priority, an Arb Won message cannot be delivered to expander phy 1 (See first issue in this document for proof). Thus we have a standoff.

Solution
Upon entrance to XL1:Request_Path, the Phy Status is changed from Phy Status (Partial Pathway/Blocked Partial Pathway) to Phy Status (Idle). Further, the specification doesn't officially declare a Phy Status (Idle), however, by reason of deduction, one must exist. It would be helpful to define this value for Phy Status to aid the reader.

Issue 3: Specification Clarification for Phy Status (Partial Pathway)

Problem
There is text in Section 4.6.6.3 ECM Interface in Table 11 under the row for Phy Status (Partial Pathway) that is misleading. In case a) the text is as follows:

is being used for an unblocked partial pathway (i.e., the expander phy is in the XL3:Open_Confirm_Wait state or XL6:Open_Response_Wait state and the last AIP transmitted or received was not AIP (WAITING ON PARTIAL)); or

~~I made the assumption that the act of transmitting or receiving AIP (WAITING_ON_PARTIAL) applied to the expander link while in XL3:Open_Confirm_Wait state. Based on experience and comments in Section 7 regarding the XL Expander link definition, transmitting and receiving does not affect the value of Phy Status while in XL3:Open_Confirm_Wait state.~~

Solution
~~Reword clause after i.e. to make it clear that transmitting/receiving AIP is only applicable to XL6:Open_Response_Wait. This also applies to the wording in the Phy Status (Blocked Partial Pathway) row of the same table.~~

Reword clause to make the OR and AND combinations more obvious in the text. By XL rules, it seems that it could only be possible to transmit AIP while in XL3 and receive AIP while in XL6.

Issue 4: Expander – 7.15.4.5 Transition XL1:Request_Path to XL5:Forward_Open

This issue was originally part of our SAS 1.1 letter ballot comments but has not been resolved yet. We are including it here for completeness of all our unresolved issues with the expander architecture.

Problem

The specification indicates that if a forward open message is received after an arbitrating (NORMAL) message has been received, the forward open message is ignored. We believe there is a flaw in the statement or perhaps overall in the expander function handling. It is illustrated in the following case.

Expander Port 2 wins arbitration to open Expander Port 0. Expander Port 1 receives an Open Address Frame (OAF) to Open Expander Port 2 but has to hold off as it waits for the connection to try to open between 2 and 0. The expander function sends an arbitrating (NORMAL) message to expander Port 1 to acknowledge the receipt of the request path message. Expander Port 0 transmits an OAF at the same time an OAF resolving to port 1 is received from the device connected to Expander Port 0. The OAF received by expander Port 0 wins over the outgoing OAF by arbitration rules. Expander Port 0 sends a backoff retry message to Expander Port 2 and also a request path message to the expander function requesting a connection to port 1. The request path message from Expander Port 0 wins by arbitration rules, that is, it is more significant than the outstanding request message by Expander Port 1. As a result, Expander Port 0 issues a forward open message to Expander Port 1. However, since Expander Port 1 had already received an arbitrating (NORMAL) message while in the XL1:Request_Path state, the forward open message is ignored and a stall occurs.

Solution

After discussion with Tim H. and comments from Robert Elliot, an additional qualification to Arb Lost is required. Currently the specification only considers the source and destination expander phy in the Arb Lost generation. The algorithm should include other expander phys with outstanding request path messages as well.

~~Removing the restriction of transitioning to XL5:Forward_Open if an Arbitrating (Normal) message has been seen alleviates the problem. There are two cases to consider in determining that this is a valid solution. Following the example described above:~~

~~The first case is that the OAF that is forwarded to Expander Port 1 wins via arbitration rules over the OAF received by Expander Port 1. The second case being the OAF that is forwarded to Expander Port 1 loses via arbitration rules over the OAF received by Expander Port 1.~~

~~In the case that the forwarded OAF wins, with the restriction removed, the OAF will be received by the port connected and will discard the OAF it sent. This mechanism is already described in both the expander and link layer specification.~~

~~The second case can not occur because a forward open message will not be generated by Expander Port 0 destined for Expander Port 1 since Expander Port 0 did not win arbitration in the Expander Function.~~

Issue 5: End Device Connection Rate matching algorithm causes expander overflow

Consider the case where device 0 is running 3G, and connected at 1.5G through an Expander to device 1 running at a link speed of 1.5G. Device 0 is the first to send CLOSE.

In SAS r10, Section 7.13, the rules for rate matching is that the end device stops rate matching when the 1st dword of a close sequence is transmitted. Therefore there will be a stream of IDLEs following the CLOSE.

In Section 7.15.10, the XL inbound phy continues to forward dwords received even after a Close Received Message arrives. The incoming dwords from the end device will contain mostly IDLEs and a few ALIGN / NOTIFIES. The IDLEs will be forward to the EF and onto the destination phy.

Operational problem:

What happens in this situation is that immediately following the receipt of the CLOSE in the expander phy connected to device 0, the IDLE stream will no longer be rate matched, and hence the 3G stream will be forwarded to the expander phy running at 1.5G link, and an overflow of the outbound fifo will occur regardless of how deep it is.

Error Case problem:

Another, perhaps more serious, problem arises when the just one of the CLOSE primitives is damaged / lost on the wire resulting in no CLOSE sequence recognized by the remote device. In this case the source expander phy will be forwarding a 3G un-rate matched stream into a 1.5G destination Phy.

Solutions:

1) Unfortunately, due to the fact that it affects the end device, the most architecturally proper / consistent, is to change section 7.13 to only discontinue rate matching after both the LAST dword of CLOSE is sent AND a CLOSE sequence has been received. This avoids the problem totally and is consistent with the notion outlined in that section that we are rate matching even at the end of the EOAF to make the expander's job easy. This is a one term change to most state machine implementations and avoids both the operational and error case problems.

2) Another possible fix which addresses the operational problem, and lessens, but does not eliminate, the error case problem, is to change section 7.15.10 to remember that a Forward Close Message has been sent to the EF, and upon this event cease forwarding all incoming dwords to the EF. The impact of this is that no additional primitives will be forwarded. BREAK is the exception, as there is a special clause to forward breaks already. As mentioned this does not eliminate the error case of a damaged close sequence, but at least isolates it to a single link rather than the entire fabric of expanders.

Issue 6: Backoff Reverse Path Open Address Frame Connection Rate Checking

Issue

Currently when a backoff reverse path message is issued, the received open address frame (OAF) that caused the message is automatically forwarded. If the connection rate in the received OAF is invalid (e.g. 3G connection rate on 1.5G link), rate matching problems may occur. After the transmission of the OAF on the destination expander phy of the reverse path message, establishing rate matching in this case is undefined and inappropriate.

Solution

Similar to how the Expander Function checks the content of a request path message for valid content, the connection rate should be checked for a reverse path message. If the reverse path message has an invalid connection rate as determined by the expander function, the associated expander phys should go through the same procedure as in the backoff retry case. This will cause the involved expander phys to resissue their respective request path messages and go through arbitration again. An Arb_Reject could then be issued for the invalid connection rate.

Issue 7: Receiving Forward Open while in XL4:Open_Reject

Issue

If a request path message yields an Arb_Reject message, the XL state machine transitions to XL4:Open_Reject. The XL state machine remains in XL4:Open_Reject until an Open Reject message is sent to the XL Transmitter. However, the act of sending the Open Reject message is implementation specific which may or may not yield a delay before transmission.

Consider the case where expander phy X is issued an Arb Reject, and expander phy Y issued a request path message destined to expander phy X. It is possible pending design implementation, that a forward open message may be received by expander phy X while still in XL4:Open_Reject. Per specification, the message would be missed.

Solution

Similar to cases in the XL state machine and other state machines in the specification, the forward open message should be carried through XL4:Open_Reject into the XL0:Idle state. This will allow the XL state machine to transitions to the XL5:Forward_Open state appropriately independent of design implementation.