October 25, 2002

To: T10 Technical Committee

From: Timothy Hoglund (LSI Logic)

Subject: SAS Expander Annex (informative)

This proposal seeks to provide additional insight into how SAS expanders process connection requests in the form of an informative annex. Information provided hereby is meant as a supplement to relevant normative sections within the SAS draft specification, i.e. this proposed annex contains examples but not in a comprehensive manner.

Revision 0
- initial draft

Revision 1
- incorporate feedback from September SAS Protocol working group meeting (Minneapolis)

Revision 2
- incorporate feedback from October SAS Protocol working group meeting (Las Vegas)
  - made use of color consistent
  - made STP examples use same format as SAS general examples
  - added STP expander initiated CLOSE example
  - miscellaneous editorial adjustments…

X SAS Expander Annex (informative)

The following examples are intended to provide additional insight into how SAS Expanders process connection requests.

X.1 SAS connection examples

Figure 1 shows the topology used by the examples in this sub-clause.

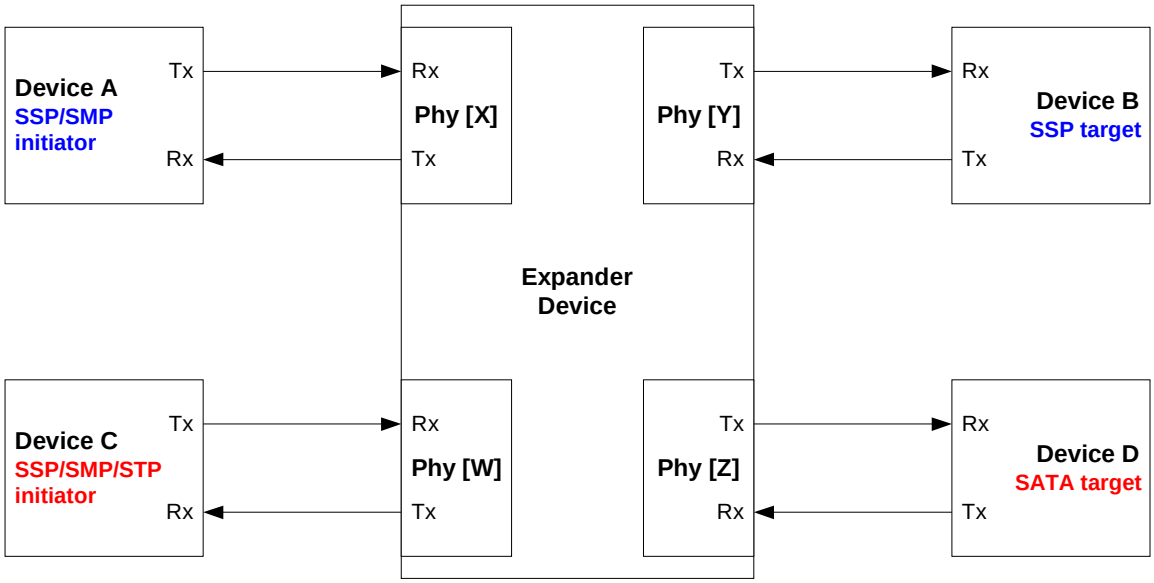

Figure 1: Example Topology

Table 1 defines the columns used within the figures contained within this sub-clause.

Column Header	Description
Phy [X] RX	Phy [X] receive from Device A
Phy [X] TX	Phy [X] transmit to Device A
Phy [X] XL state	Phy [X] XL state
Phy [X] XL req/rsp	Phy [X] XL request and response interface signals
Phy [X] cnf/ind	Phy [X] XL confirm and indicate interface signals
Phy [Y] cnf/ind	Phy [Y] XL confirm and indicate interface signals
Phy [Y] XL req/rsp	Phy [Y] XL request and response interface signals
Phy [Y] XL state	Phy [Y] XL state
Phy [Y] TX	Phy [Y] transmit to Device B
Phy [Y] RX	Phy [Y] receive from Device B

Table 1: Column descriptions for SAS connection examples

X.1.1 Connection Request – Open Accept

Figure 2 shows the establishment of a successful connection between two end devices.

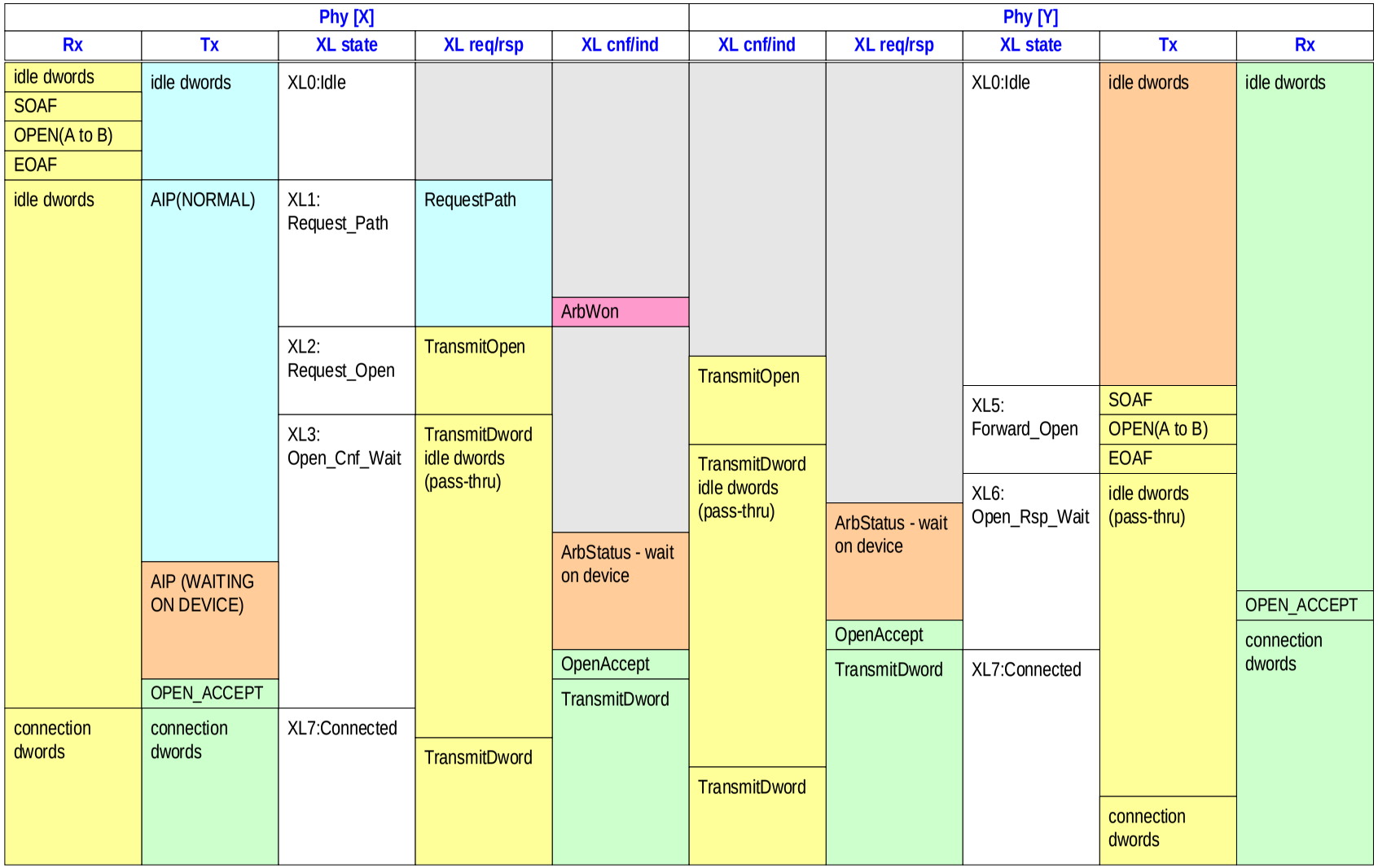

Figure 2. Open Accept

X.1.2 Connection Request – Open Reject by end device

Figure 3 shows failure to establish a connection due to rejection of the connection request by an end device.

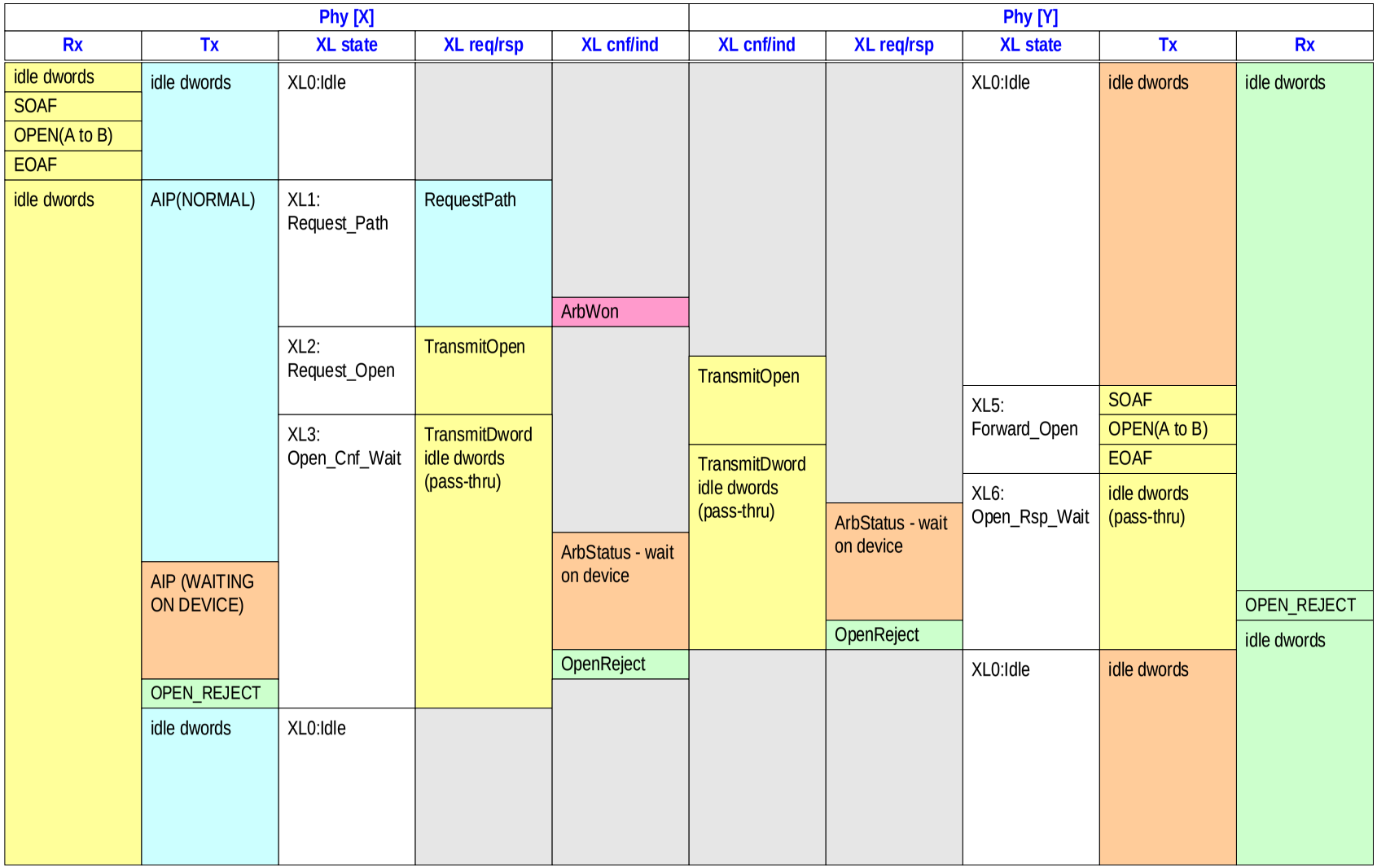

Figure 3. Open Reject by end device

X.1.3 Connection Request – Open Reject by expander device

Figure 4 shows failure to establish a connection due to rejection of the connection request by an expander.

Phy [X]					Phy [Y]				
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx
idle dwords	idle dwords	XL0:Idle					XL0:Idle	idle dwords	idle dwords
SOAF									
OPEN(A to B)									
EOAF									
idle dwords	AIP(NORMAL)	XL1: Request_Path	RequestPath						
				ArbReject					
	OPEN_REJECT	XL4:Open_Reject							
	idle dwords	XL0:Idle							

Figure 4. Open Reject by expander device

X.1.4 Connection Request – Arbitration Lost

Figure 5 shows two end devices attempting to establish a connection at the same time. This example assumes that the OPEN(A to B) address frame has higher priority than the OPEN(B to A) address frame and therefore device A wins arbitration and device B loses arbitration.

Figure 5. Arbitration Lost

X.1.5 Connection Request – Backoff and Retry

Figure 6 shows the condition which occurs when a higher priority OPEN address frame (B to C) is received by a phy which has previously forwarded an OPEN address frame to a different destination (A to B). In this case Phy[X] is required to back off and retry path arbitration.

Figure 6. Backoff and Retry

X.1.6 Connection Request – Backoff and Reverse Path

Figure 7 shows the condition which occurs when a higher priority OPEN address frame (B to A) is received by a phy which has previously forwarded an OPEN address frame to the same destination (A to B). In this case Phy[Y] forwards the higher priority OPEN to Phy[X].

Figure 7. Backoff and Reverse Path

X.1.7 Connection Close – single step

Figure 8 shows an end device initiating the closing of a connection by transmitting CLOSE, followed by another end device responding with CLOSE at a later time.

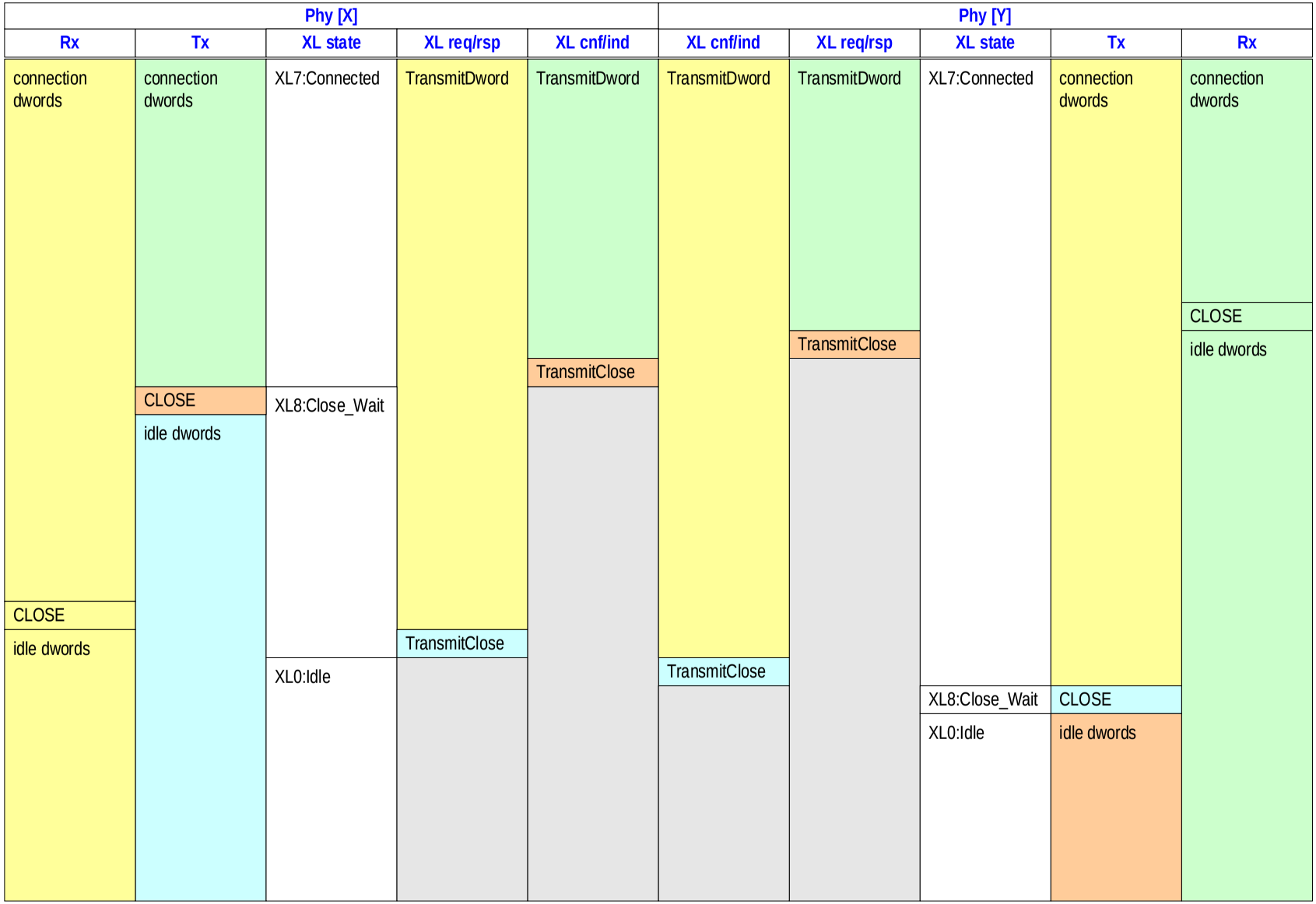

Figure 8. Connection Close – single step

X.1.8 Connection Close – simultaneous

Figure 9 shows two end devices simultaneously transmitting CLOSE to each other.

Phy [X]					Phy [Y]				
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx
connection dwords	connection dwords	XL7:Connected	TransmitDword	TransmitDword	TransmitDword	TransmitDword	XL7:Connected	connection dwords	connection dwords
CLOSE									CLOSE
idle dwords			TransmitClose			TransmitClose			idle dwords
				TransmitClose	TransmitClose				
	CLOSE	XL8:Close_Wait					XL8:Close_Wait	CLOSE	
	idle dwords	XL0:Idle					XL0:Idle	idle dwords	

Figure 9. Connection Close - simultaneous

X.1.9 Break handling during path arbitration

Figure 10 shows an expander device responding to the reception of a BREAK primitive sequence during path arbitration.

Phy [X]					Phy [Y]				
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx
idle dwords	idle dwords	XL0:Idle					XL0:Idle	idle dwords	idle dwords
SOAF									
OPEN(A to B)									
EOAF									
idle dwords	AIP(NORMAL)	XL1: Request_Path	RequestPath						
BREAK									
	BREAK	XL9:Break							
idle dwords	idle dwords	XL0:Idle							

Figure 10. Break handling during arbitration

X.1.10 Break handling during connection

Figure 11 shows an expander device responding to the reception of a BREAK primitive sequence during a connection.

Figure 11. Break handling during connection

X.1.11 STP Connection – originated by STP initiator

Figure 12 shows an STP initiator originating a connection to a SATA target.

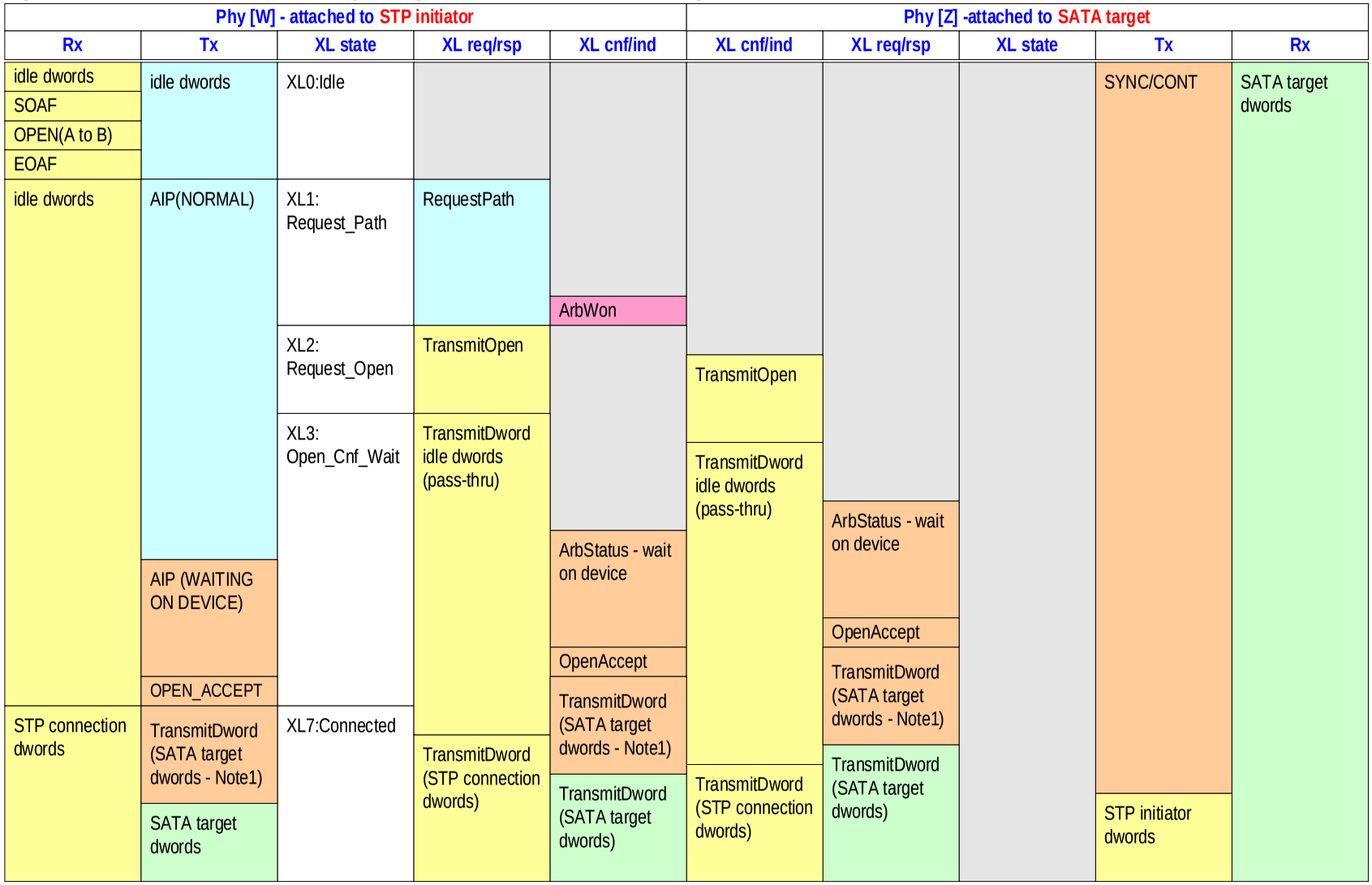

Figure 12. STP Connection – originated by STP initiator

Notes:

1. Expander required to duplicate the dword stream which is being received from the SATA target before forwarding dwords – this is to insure that a continued SATA primitive is correctly forwarded back to STP initiator.

X.1.12 STP Connection – originated by expander on behalf of SATA target

Figure 13 shows an expander originating a connection on behalf of a SATA target which is requesting to transmit a frame.

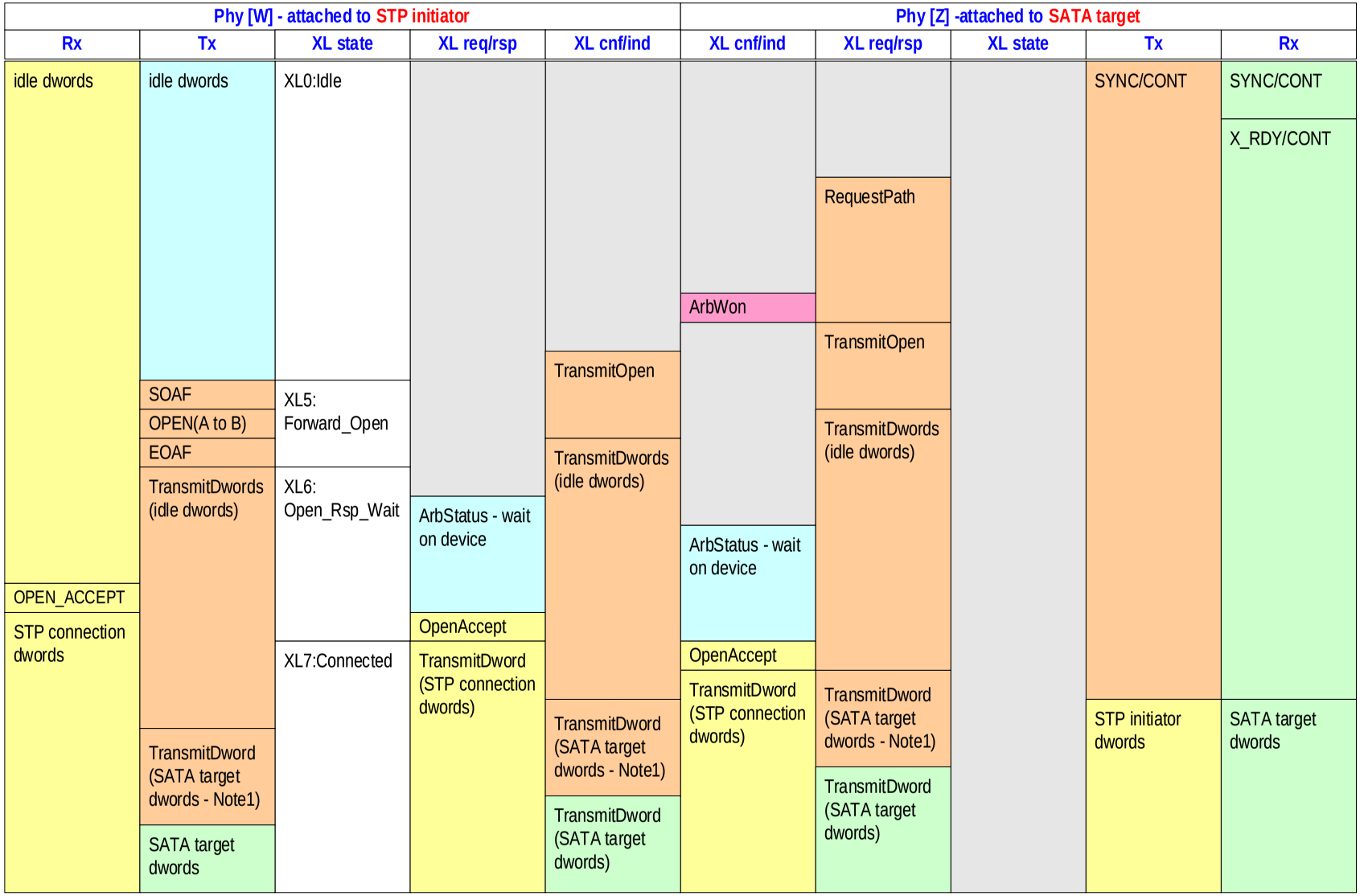

Figure 13. STP Connection – originated by expander on behalf of SATA target

Notes:

1. Expander required to duplicate the dword stream which is being received from the SATA target before forwarding dwords – this is to insure that a continued SATA primitive is correctly forwarded back to STP initiator.

X.1.13 STP Connection – CLOSE originated by STP initiator

Figure 14 shows an STP initiator closing a connection to a SATA target.

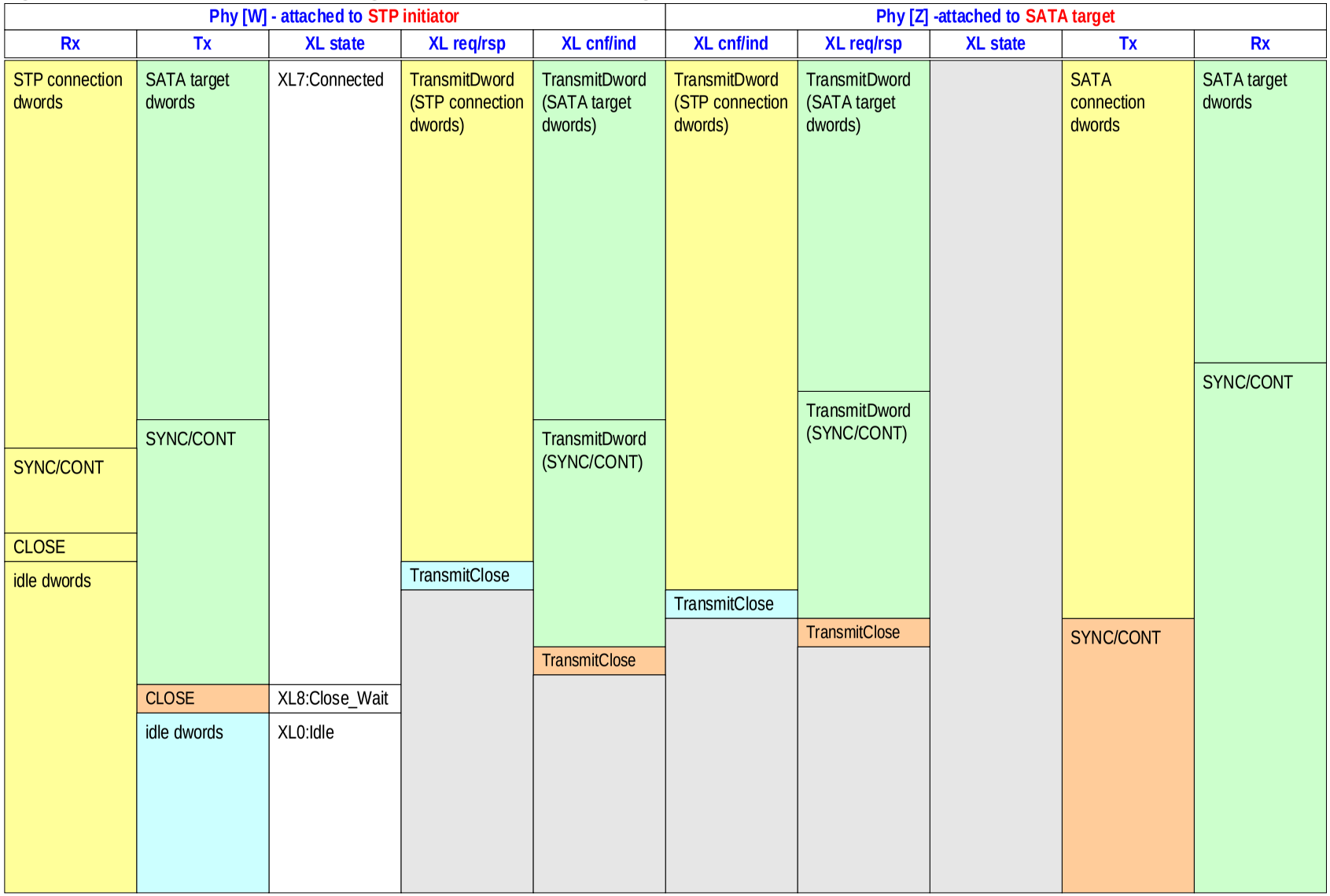

Figure 14. STP Connection – CLOSE originated by STP initiator

X.1.14 STP Connection – CLOSE originated by expander

Figure 14 shows an expander closing an STP connection.

Phy [W] - attached to STP initiator					Phy [Z] -attached to SATA target				
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx
STP connection dwords	SATA target dwords	XL7:Connected	TransmitDword (STP connection dwords)	TransmitDword (SATA target dwords)	TransmitDword (STP connection dwords)	TransmitDword (SATA target dwords)		SATA connection dwords	SATA target dwords
	SYNC/CONT		TransmitDword (SYNC/CONT)	TransmitDword (SYNC/CONT)		TransmitDword (SYNC/CONT)			SYNC/CONT
SYNC/CONT					TransmitDword (SYNC/CONT)				
				TransmitClose		TransmitClose		SYNC/CONT	
	CLOSE	XL8:Close_Wait							
CLOSE	idle dwords	XL0:Idle							
idle dwords			TransmitClose		TransmitClose				

Figure 15. STP Connection – CLOSE originated by STP initiator

X.2 Pathway blocked and recovery example

Figure 16 shows the topology used to illustrate pathway recovery:

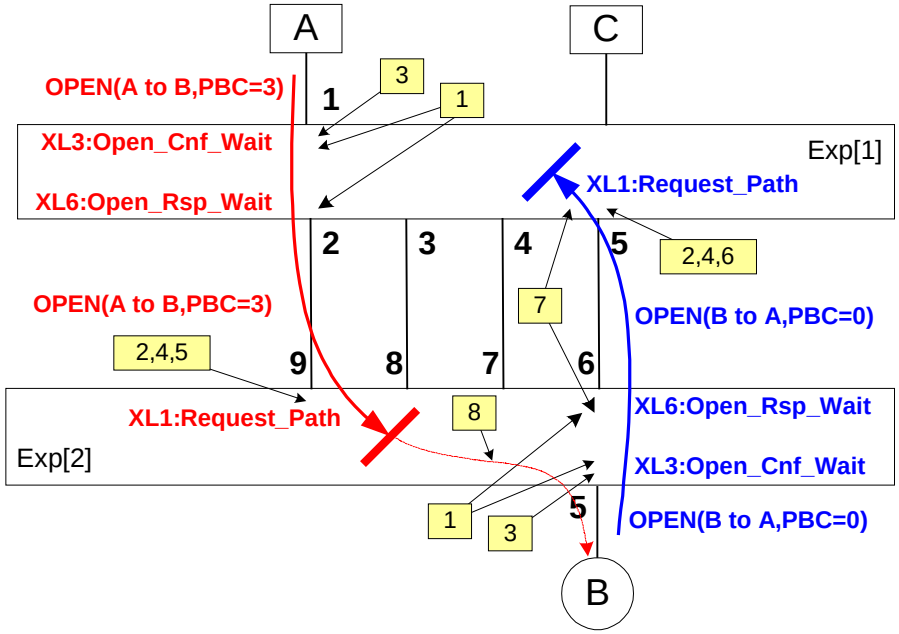

Figure 16. Partial Pathway Recovery

The sequence of events used to identify pathway blockage and to perform pathway recovery are as follows:

1. Exp[1].Phy[1,2] and Exp[2].Phy[5,6] send Phy Status (Partial Pathway) to Connection Manager to indicate that they contain partial pathways;
2. Exp[1].Phy[5] and Exp[2].Phy[9] receive Arbitrating (Waiting On Partial) from Connection Manager which cause them to transmit AIP(WAITING ON PARTIAL);
3. AIP(WAITING ON PARTIAL) received by Exp[1].Phy[2] and Exp[2].Phy[6] then forwarded to Exp[1].Phy[1] and Exp[2].Phy[5] as Arb Status (Waiting On Partial). Exp[1].Phy[1] and Exp[2].Phy[5] send Phy Status (Blocked On Partial) to Connection Manager as confirmation that they are blocked waiting on a partial pathway in another expander;
4. Exp[1].Phy[5] and Exp[2].Phy[9] receive Arbitrating (Blocked On Partial) from Connection Manager while all destination phys contain a Phy Status of Blocked On Partial which cause them to run their Partial Pathway Timeout timers;
5. PPT expires in Exp[2].Phy[9] – this causes a request to the Connection Manager to resolve pathway blockage. Pathway recovery priority for this phy is not lower than all phys within the destination port which are also blocked. Connection Manager does not provide Arb Reject (Pathway Blocked) to Exp[2].Phy[9] so this phy waits for pathway resolution to occur elsewhere in the topology;
6. PPT expires in Exp[1].Phy[5] – this causes a request to the Connection Manager to resolve pathway blockage. Pathway recovery priority for this phy is lower than all phys within the destination port which are also blocked. Connection Manager provides Arb Reject (Pathway Blocked) to Exp[1].Phy[5] which instructs this phy to reject the connection request using OPEN_REJECT(PATHWAY BLOCKED);
7. OPEN_REJECT(PATHWAY BLOCKED) tears down partial pathway all the way to the originating end device (Device B); and
8. Partial pathway established between Exp[2].Phy[9] and Exp[2].Phy[5] and OPEN(A to B) is delivered to Device B.