To: T10 Technical Committee
From: Rob Elliott, HP (elliott@hp.com)
Date: 6 December 2007
Subject: 08-009r0 SAS-2 Elasticity buffer clarifications

Revision history
Revision 0 (6 December 2007) First revision

Related documents
sas2r13 - Serial Attached SCSI - 2 (SAS-2) revision 13

Overview
This proposal clarifies issues regarding deletable primitives and the elasticity buffer.

1. Based on feedback from a SAS-2 editing session, the elasticity buffer figure is redrawn to show more detail about how the elasticity buffer interacts of the state machines.

2. SAS-2 should recommended (at the “should” level) that phys detect NOTIFYs before rather than after the elasticity buffer. It is highly desirable that phys detect all NOTIFY (ENABLE SPINUP) or NOTIFY (POWER LOSS EXPECTED) primitives and not miss them. Since there are only 4 unique NOTIFYs to detect, and since the precise number of each of them and the order in which they are received is unimportant, this could be implemented using four flag bits per phy (or four small counters) with a clearing mechanism.

3. Deletable primitives should be grouped into their own section in 7.2 since they are described in separate tables in 7.2.2 and 7.2.3. That provides a better cross-reference for uses of the phrase than pointing to 7.2 or 7.3.

Suggested changes

4.3.3 Receive data path

The SP_DWS receiver (see 6.9.2) establishes dword synchronization and sends dwords to the SP_DWS state machine (see 6.9) and to the link layer state machine receivers.

If multiplexing (see 6.10) is enabled (see table 92 in 6.7.4.2.3.3), the SP_DWS receiver uses incoming MUX primitives to determine the logical link numbers and route the dwords to the appropriate link layer receivers.

Figure 36 shows the receive data path in a SAS phy, showing the relationship between:

a) the SP_DWS state machine (see 6.9) and the SP_DWS receiver (see 6.9.2);
b) the SL_IR state machines (see 7.9.4) and the SL_IR receiver (see 7.9.4.2);
c) the SL state machines (see 7.14) and the SL receiver (see 7.14.2); and
d) the SSP receiver (see 7.16.8.2), SMP receiver (see 7.18.5.2), and STP receiver (see 7.17.8).

See figure 156 in 7.3.1 for more information about the elasticity buffer, which is not shown in figure 36.

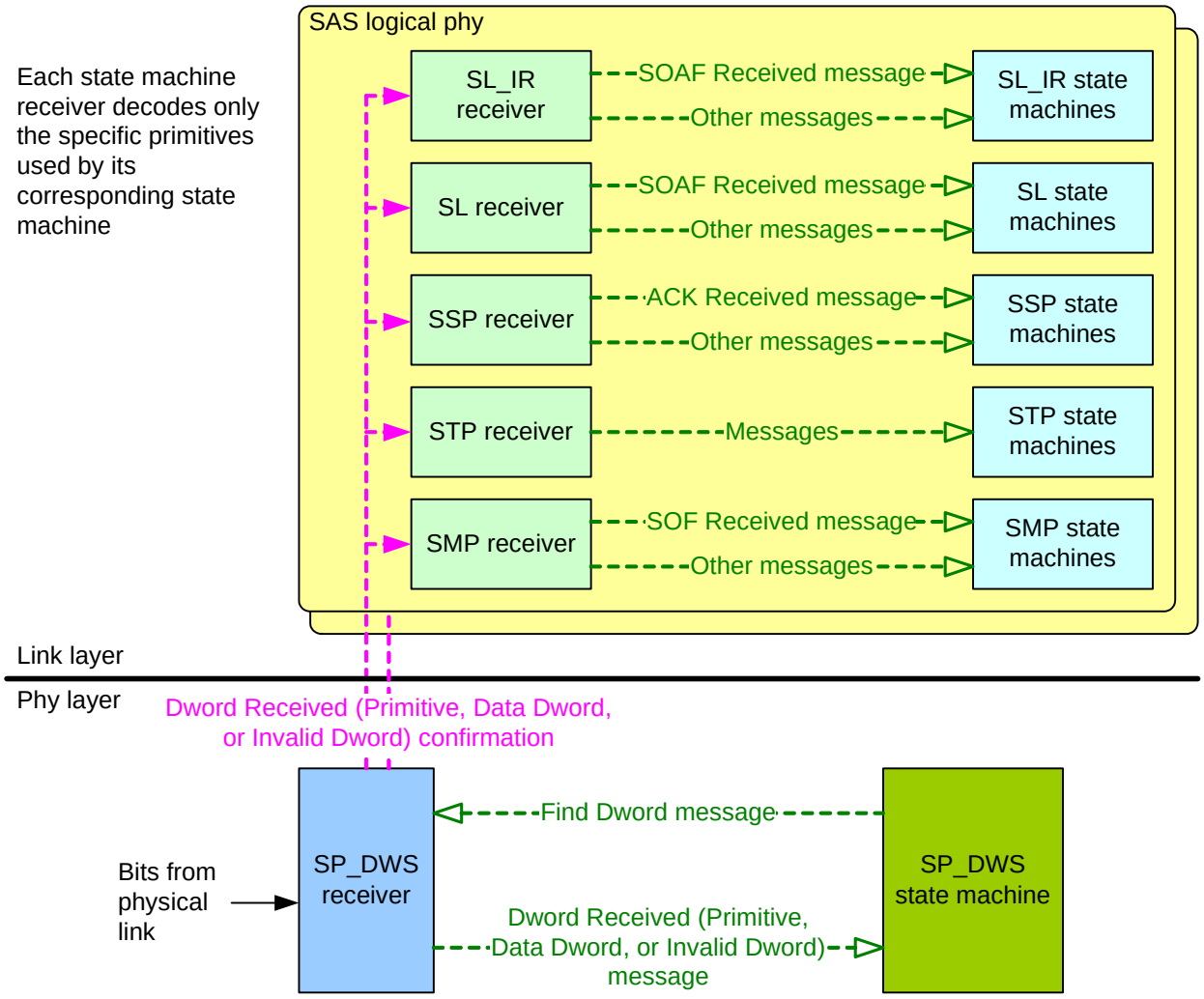

Figure 36 — Receive data path in a SAS phy

Figure 37 shows the receive data path in an expander phy showing the relationship between:

a) the SP_DWS state machine (see 6.9) and the SP_DWS receiver (see 6.9.2);
b) the SL_IR state machines (see 7.9.4) and the SL_IR receiver (see 7.9.4.2); and
c) the XL state machines (see 7.15) and the XL receiver (see 7.15.2).

See figure 156 in 7.3.1 for more information about the elasticity buffer, which is not shown in figure 37.

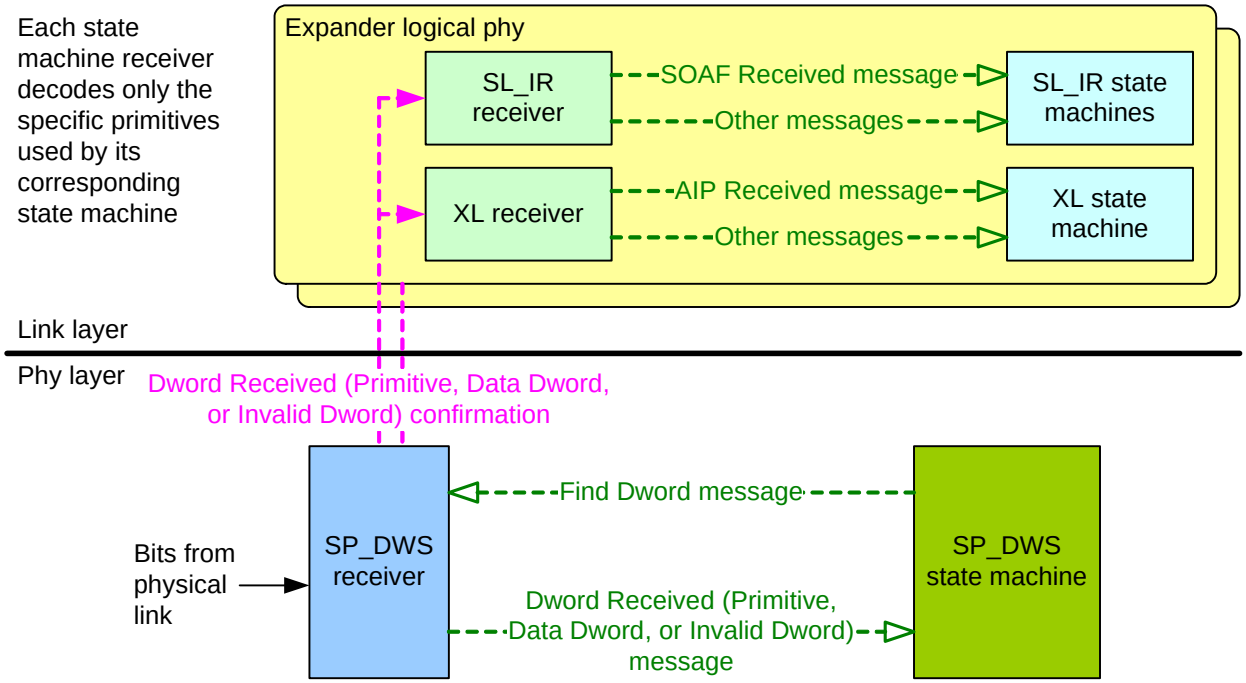

Figure 37 — Receive data path in an expander phy

6.7.4.3 Multiplexing sequence

If SNW-3 indicates multiplexing (see 6.10) is enabled (see table 92 in 6.7.4.2.3.3), the phy shall transmit the multiplexing sequence immediately after the speed negotiation sequence.

The multiplexing sequence is:

1) MUX (LOGICAL LINK 0);
2) MUX (LOGICAL LINK 1);
3) MUX (LOGICAL LINK 0);
4) MUX (LOGICAL LINK 1);
5) MUX (LOGICAL LINK 0); and
6) MUX (LOGICAL LINK 1).

The phy shall not transmit deletable primitives for physical link rate tolerance management (see 7.3) during the multiplexing sequence.

If SNW-3 indicates multiplexing is not enabled, the phy shall not transmit the multiplexing sequence.

The phy shall assign the incoming logical links to its logical phys based on the first MUX primitive it receives:

a) MUX (LOGICAL LINK 0) indicates the position of logical link 0 and indicates the next dword is in logical link 1; or
b) MUX (LOGICAL LINK 1) indicates the position of logical link 1 and indicates the next dword is in logical link 0.

The phy shall process MUX primitives ~~in logic running off the received clock without using an elasticity buffer~~in logic using the clock derived from the bits in the physical link (see 7.3) rather than logic after the elasticity buffer, because they are not accompanied by additional deletable primitives (e.g., ALIGNs and/or NOTIFYs).

The phy shall handle errors during the multiplexing sequence (i.e., after receiving the first MUX primitive) as follows:

a) If the phy loses dword synchronization, it shall restart the link reset sequence rather than attempt to reestablish dword synchronization;

b) If the phy receives a dword that is not a MUX primitive before receiving the MUX primitive expected in that position, it shall discard the dword;
c) If the phy receives an invalid dword, it shall discard the dword; and
d) If the phy receives a MUX primitive that does not match the MUX primitive expected in that position (i.e., it receives MUX (LOGICAL LINK 1) on logical link 0 or receives MUX (LOGICAL LINK 0) on logical link 1), it shall restart the link reset sequence.

7.2 Primitives

7.2.1 Primitives overview

Primitives are dwords whose first character is a K28.3 or K28.5. Primitives are not considered big-endian or little-endian; they are just interpreted as first, second, third, and last characters. Table 98 defines the primitive format. []

7.2.2 Primitive summary

Table 99 defines the deletable primitives. []

Table 100 defines the primitives not specific to the type of connection.[]

Table 101 defines the primitives used only inside SSP and SMP connections. []

Table 102 lists the primitives used only inside STP connections and on SATA physical links. []

7.2.3 Primitive encodings

Table 103 defines the primitive encoding for deletable primitives. []

Table 104 defines the primitive encoding for primitives not specific to type of connection. []

Table 105 defines the primitive encodings for primitives used only inside SSP and SMP connections. []

Table 106 lists the primitive encodings for primitives used only inside STP connections and on SATA physical links. []

7.2.4 Primitive sequences

7.2.4.1 Primitive sequences overview

Table 107 summarizes the types of primitive sequences. []

Any number of deletable primitives may be sent inside primitive sequences without affecting the count or breaking the consecutiveness requirements. Rate matching deletable primitives shall be sent inside primitive sequences inside of connections if rate matching is enabled (see 7.13).

7.2.4.2 Single primitive sequence

Primitives labeled as single primitive sequences (e.g., RRDY, SATA_SOF) shall be transmitted one time to form a single primitive sequence.

Receivers count each primitive received that is labeled as a single primitive sequence as a distinct single primitive sequence.

ALIGNs, NOTIFYs, and MUXs are called deletable primitives (see 7.3).

7.2.4.3 Repeated primitive sequence

Primitives that form repeated primitive sequences (e.g., SATA_PMACK) shall be transmitted one or more times. Only STP primitives form repeated primitive sequences. Any number of deletable primitives may be sent inside repeated primitive sequences as described in 7.2.4.1.

Figure 151 shows an example of transmitting a repeated primitive sequence. []

Receivers do not count the number of times a repeated primitive is received (i.e., receivers are simply in the state of receiving the primitive). An expander device forwarding a repeated primitive sequence may transmit more repeated primitives than it receives (i.e., expand) or transmit fewer repeated primitives than it receives (i.e, contract).

Figure 152 shows an example of receiving a repeated primitive sequence. []

7.2.4.4 Continued primitive sequence

Primitives that form continued primitive sequences (e.g., SATA_HOLD) shall be transmitted as specified in 7.17.3. Any number of deletable primitives may be sent inside continued primitive sequences as described in 7.2.4.1.

7.2.4.5 Extended primitive sequence

Primitives that form extended primitive sequences (e.g., AIP) shall be transmitted three times consecutively. Any number of deletable primitives may be sent inside primitive sequences as described in 7.2.4.1.

A receiver shall detect an extended primitive sequence after the primitive is received one time. The receiver shall process an extended primitive sequence the same as a single primitive sequence (see 7.2.4.2).

Figure 153 shows examples of extended primitive sequences. []

7.2.4.6 Triple primitive sequence

Primitives that form triple primitive sequences (e.g., CLOSE (NORMAL)) shall be sent three times consecutively. Any number of deletable primitives may be sent inside primitive sequences as described in 7.2.4.1.

Receivers shall detect a triple primitive sequence after the identical primitive is received in three consecutive dwords. After detecting a triple primitive sequence, a receiver shall not detect a second instance of the same triple primitive sequence until it has received three consecutive dwords that are not any of the following:

a) the original primitive; or
b) a deletable primitive.

Figure 154 shows examples of triple primitive sequences. []

7.2.4.7 Redundant primitive sequence

Primitives that form redundant primitive sequences (e.g., BROADCAST (CHANGE)) shall be sent six times consecutively. Any number of deletable primitives may be sent inside primitive sequences as described in 7.2.4.1.

A receiver shall detect a redundant primitive sequence after the identical primitive is received in any three of six consecutive dwords. After detecting a redundant primitive sequence, a receiver shall not detect a second instance of the same redundant primitive sequence until it has received six consecutive dwords that are not any of the following:

a) the original primitive; or
b) a deletable primitive.

Figure 155 shows examples of redundant primitive sequences. []

7.2.5 Deletable primitives [new subclause]

7.2.5.1 ALIGN [subclause moved here]

ALIGNs are used for:

a) OOB signals (see 6.6);
b) character and dword alignment during the speed negotiation sequence (see 6.7.4.2);
c) physical link rate tolerance management after the phy reset sequence (see 7.3); and
d) rate matching during connections (see 7.13).

ALIGNs are deletable primitives (see 7.3).

Table 109 defines the different versions of ALIGN primitives.

Table 109 — ALIGN primitives

Primitive	Description
ALIGN (0)	Used for OOB signals, the speed negotiation sequence, physical link rate tolerance management, and rate matching.
ALIGN (1)	Used for the speed negotiation sequence, physical link rate tolerance management, and rate matching.
ALIGN (2)	Used for physical link rate tolerance management and rate matching.
ALIGN (3)	

Phys may use ALIGN (0) to construct OOB signals as described in 6.6. Phys use ALIGN (0) and ALIGN (1) during the speed negotiation sequence as described in 6.7.4.2. Phys shall rotate through ALIGN (0), ALIGN (1), ALIGN (2), and ALIGN (3) for all ALIGNs sent after the phy reset sequence.

Phys receiving ALIGNs after the phy reset sequence shall not verify the rotation and shall accept any of the ALIGNs at any time.

Phys shall only detect an ALIGN after decoding all four characters in the primitive.

NOTE 1 - SATA devices are allowed to decode every dword starting with a K28.5 as an ALIGN, since ALIGN is the only primitive defined starting with K28.5.

For physical link rate tolerance management and rate matching, ALIGNs may be replaced by NOTIFYs (see 7.2.6.11) or MUXs (see 7.2.6.10). ALIGNs shall not be replaced by NOTIFYs or MUXs during OOB signals or speed negotiation.

7.2.5.2 MUX (Multiplex) [subclause moved here]

MUX is used if multiplexing (see 6.10) is enabled (see table 92 in 6.7.4.2.3.3) as follows:

a) transmitted during the multiplexing sequence (see 6.7.4.3); and
b) substituted for an ALIGN (see 7.2.6.2) being transmitted for physical link rate tolerance management (see 7.3) or rate matching (see 7.13) to confirm the logical link number.

Substitution of a MUX for an ALIGN may or may not affect the ALIGN rotation (i.e., the MUX may take the place of one of the ALIGNs in the rotation through ALIGN (0), ALIGN (1), ALIGN (2), and ALIGN (3), or it may delay the rotation).

MUXs are deletable primitives (see 7.3).

The versions of MUX are defined in table 112.

Table 112 — MUX primitives

Primitive	Description
MUX (LOGICAL LINK 0)	Establishes the position of dwords in logical link 0.
MUX (LOGICAL LINK 1)	Establishes the position of dwords in logical link 1.

See 6.10 for details on multiplexing.

7.2.5.3 NOTIFY [subclause moved here]

7.2.5.3.1 NOTIFY overview

NOTIFY may be transmitted in place of any ALIGN (see 7.2.6.2) being transmitted for physical link rate tolerance management (see 7.3) and rate matching (see 7.13). Substitution of a NOTIFY for an ALIGN may or may not affect the ALIGN rotation (i.e., the NOTIFY may take the place of one of the ALIGNs in the rotation through ALIGN (0), ALIGN (1), ALIGN (2), and ALIGN (3), or it may delay the rotation). A specific NOTIFY shall not be transmitted in more than three consecutive dwords until at least three other dwords have been transmitted.

NOTIFYs are deletable primitives (see 7.3). If a phy supports a specific NOTIFY primitive, it should decode it in logic using the clock derived from the bits in the physical link (see 7.3) rather than logic after the elasticity buffer to avoid missing detection of important information.

NOTIFY shall not be forwarded through expander devices. Expander devices shall substitute an ALIGN for a NOTIFY if necessary.

SAS target devices are not required to detect every transmitted NOTIFY.

The versions of NOTIFY representing different reasons are defined in table 113.

Table 113 — NOTIFY primitives

Primitive	Description	Reference
NOTIFY (ENABLE SPINUP)	Specify to a SAS target device that it may temporarily consume additional power while transitioning into the active or idle power condition state.	7.2.5.11.2
NOTIFY (POWER LOSS EXPECTED)	Specify to a SAS target device that power loss may occur within the time specified by the POWER LOSS TIMEOUT field in the Shared Port Control mode page (see 10.2.7.6).	7.2.5.11.3
NOTIFY (RESERVED 1)	Reserved.	
NOTIFY (RESERVED 2)		

NOTIFY (RESERVED 1) and NOTIFY (RESERVED 2) shall be ignored by all devices.

7.2.5.3.2 NOTIFY (ENABLE SPINUP)

NOTIFY (ENABLE SPINUP) is transmitted by a SAS initiator port or expander port and is used to specify to an SAS target device that it may temporarily consume additional power (e.g., while spinning-up rotating media) while transitioning into the active or idle power condition state. The length of time the SAS target device consumes additional power and the amount of additional power is vendor specific. NOTIFY (ENABLE SPINUP) shall interact with the device's power condition state transitions, controlled by the Power Conditions mode page (see SPC-4) and/or the START STOP UNIT command (see SBC-3), as described in 10.2.10.

SAS initiator devices and expander devices shall use NOTIFY (ENABLE SPINUP) while attached to SSP target devices (i.e., devices that report SSP target port support in their IDENTIFY address frames). They shall transmit one NOTIFY (ENABLE SPINUP) after power on when the enclosure is ready for initial spin-up. After the initial NOTIFY (ENABLE SPINUP), they shall transmit NOTIFY (ENABLE SPINUP) periodically. Otherwise, the selection of when and how often to transmit NOTIFY (ENABLE SPINUP) is outside the scope of this standard.

NOTE 2 - The SAS initiator device or expander device uses NOTIFY (ENABLE SPINUP) to avoid exceeding enclosure power supply capabilities during spin-up of multiple SAS target devices. It may choose to rotate transmitting NOTIFY (ENABLE SPINUP) across all of its ports, distributing it to N ports at a time if the enclosure power supply is capable of powering N SAS target devices spinning up at a time. An expander

device may allow this timing to be configured by an NVRAM programmed with enclosure-specific sequencing patterns, or may employ more complex, dynamic interaction with the enclosure power supply.

NOTE 3 - NOTIFY (ENABLE SPINUP) should be transmitted as frequently as possible to avoid incurring application layer timeouts.

I_T nexus loss, logical unit reset, and hard reset shall not cause a SAS target device to spin-up automatically on receipt of NOTIFY (ENABLE SPINUP).

A SAS target device with multiple SAS target ports shall honor NOTIFY (ENABLE SPINUP) from all its SAS target ports equivalently (e.g., if a SAS target device contains two SSP target ports A and B, powers on in the Stopped state, and receives a START STOP UNIT command with the START bit set to one through SSP target port A, then a NOTIFY (ENABLE SPINUP) received on SSP target port B causes the SAS target device to spin up its rotating media (see 10.2.10)).

7.2.5.3.3 NOTIFY (POWER LOSS EXPECTED)

NOTIFY (POWER LOSS EXPECTED) is transmitted by a SAS initiator port or expander port and is used to specify to a SAS target device that power loss may occur within the time specified in the POWER LOSS TIMEOUT field in the Shared Port Control mode page (see 10.2.7.6).

NOTIFY (POWER LOSS EXPECTED) shall be transmitted at least three times by the SAS initiator port or expander port.

If a SAS target device supports NOTIFY (POWER LOSS EXPECTED) and receives NOTIFY (POWER LOSS EXPECTED) on an SSP target port, then the device server for each logical unit to which the SSP target port has access shall:

1) stop writing data to the media on a block boundary (e.g., all write activity shall continue until a block boundary is reached then all writing shall stop);
2) clear all task sets as defined in SAM-4; and
3) establish a unit attention condition for the initiator port associated with every I_T nexus as defined in SAM-4 (e.g., with the additional sense code set to COMMANDS CLEARED BY POWER LOSS NOTIFICATION).

If a SAS target device supports NOTIFY (POWER LOSS EXPECTED) and receives NOTIFY (POWER LOSS EXPECTED) on an SSP target port, then the SAS target device shall:

a) on each phy that receives NOTIFY (POWER LOSS EXPECTED), if there is an SSP connection, then transmit a BREAK on that connection; and
b) on each phy that does not receive NOTIFY (POWER LOSS EXPECTED), if there is an SSP connection, then transmit a BREAK or a CLOSE on that connection.

The SCSI application layer that receives a Power Loss Expected event shall:

a) start the power loss timer;
b) send an Accept_Reject OPENs (Reject SSP) request to all ST_T state machines (i.e., all SSP connection requests result in OPEN_REJECT (RETRY));
c) if a SCSI Command Received transport protocol service indication is received, then the device server shall abort that command and send an Accept_Reject OPENs (Reject SSP) request to the ST_T state machine on which the SCSI Command Received transport protocol service indication was received; and
d) if the power loss timeout timer expires, then the SCSI application layer shall send an Accept_Reject OPENs (Accept SSP) request to all ST_T state machines.

If any frames are received by the SAS target device after receiving NOTIFY (POWER LOSS EXPECTED) before a connection is closed, then the SAS target device shall discard the received frames.

After power on, the power loss timeout timer shall be initialized and stopped until a NOTIFY (POWER LOSS EXPECTED) is received.

7.2.6 Primitives not specific to type of connections

7.2.6.1 AIP (Arbitration in progress)

~~7.2.6.2 ALIGN~~ [moving to new location]

7.2.6.3 BREAK

7.2.6.4 BREAK_REPLY

7.2.6.5 BROADCAST

7.2.6.6 CLOSE

7.2.6.7 EOAF (End of address frame)

7.2.6.8 ERROR

7.2.6.9 HARD_RESET

~~7.2.6.10 MUX (Multiplex)~~ [moving to new location]

~~7.2.6.11 NOTIFY~~ [moving to new location]

7.2.6.12 OPEN_ACCEPT

7.2.6.13 OPEN_REJECT

7.2.6.14 SOAF (Start of address frame)

7.2.6.15 TRAIN

7.2.6.16 TRAIN_DONE

7.2.7 Primitives used only inside SSP and SMP connections

7.2.8 Primitives used only inside STP connections and on SATA physical links

7.3 Physical link rate tolerance management

7.3.1 Physical link rate tolerance management overview

The internal clock for a ~~device~~phy is typically based on a PLL with its own clock generator and is used when transmitting dwords on the ~~logical~~physical link. When receiving, however, dwords need to be latched based on a clock derived from the input bit stream itself. Although the input clock is nominally a fixed frequency, it may differ slightly from the internal clock frequency up to the physical link rate tolerance defined in table 58 (see 5.3.3). ~~Over time, if the input clock is faster than the internal clock, the phy receiver may receive a dword and not be able to forward it to an internal buffer; this is called an overrun. If the input clock is slower than the internal clock, the phy receiver may not have a dword when needed in an internal buffer; this is called an underrun.~~ Over time:

a) if the input clock is faster than the internal clock, the phy receiver may receive a dword and not be able to forward it to an internal buffer; this is called an overrun; and
b) if the input clock is slower than the internal clock, the phy receiver may not have a dword when needed in an internal buffer; this is called an underrun.

To solve this problem, phy transmitters insert deletable primitives (see 7.2.x) in the dword stream. Phy receivers may pass deletable primitives through to their internal buffers, or may strip them out when an

overrun occurs. Phy receivers add deletable primitives when an underrun occurs. The internal logic shall ignore all deletable primitives that arrive in the internal buffers.

Elasticity buffer circuitry, as shown in figure 156, is required to absorb the slight differences in frequencies between the phys. The frequency tolerance for a phy is specified in 5.3.3. The depth of the elasticity buffer is vendor-specific but shall accommodate the physical link rate tolerance management deletable primitive insertion requirements in table 119 (see 7.3.2).

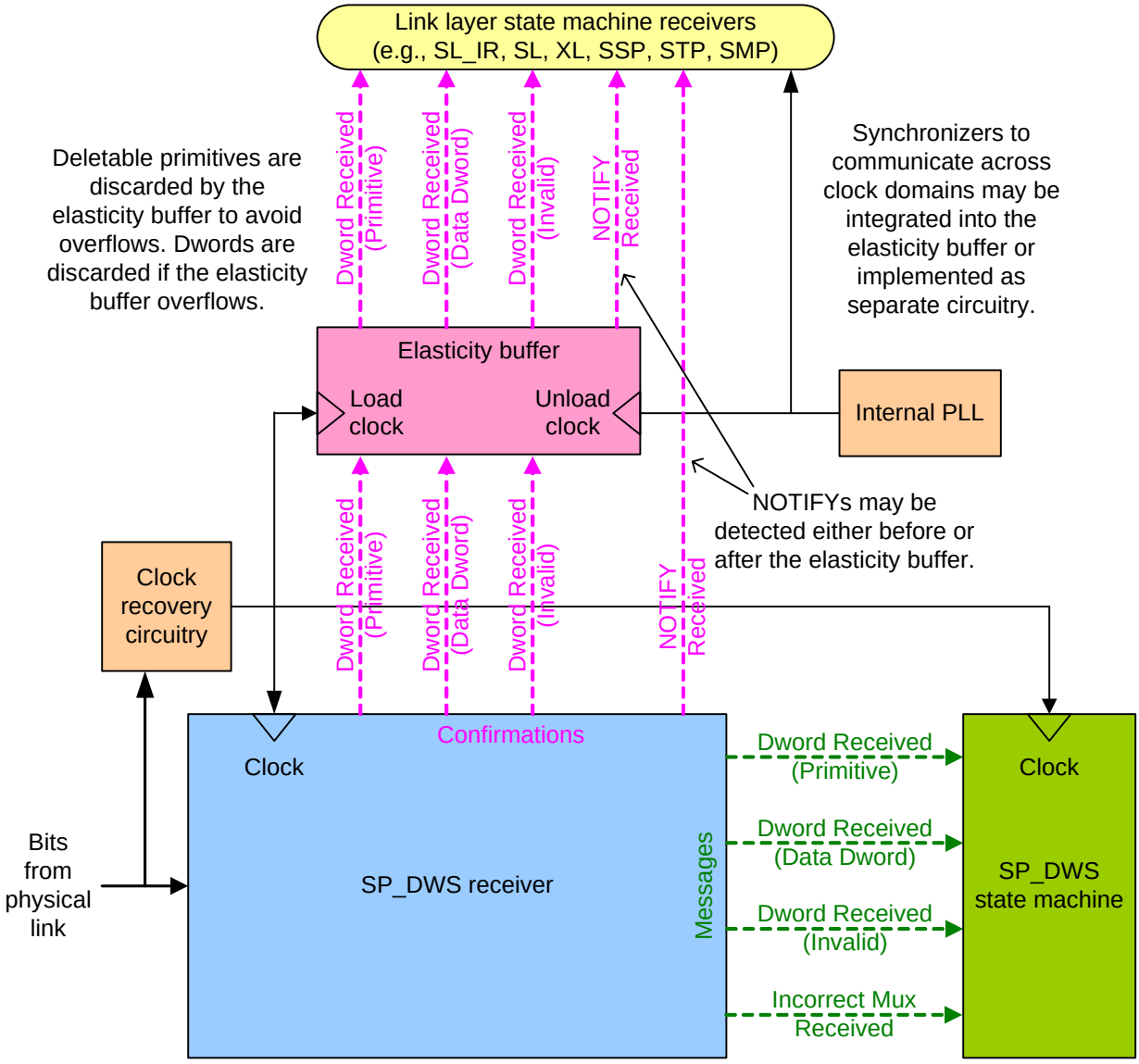

Figure 156 — Elasticity buffers