September 4, 2002

To: T10 Technical Committee
From: Brian Day
LSI Logic Corporation
4420 Arrowswest Drive
Colorado Springs, CO 80907

Re: Allowing STP Initiators to close connections

Currently in the SAS specification, section 7.15.3 requires that an STP initiator port shall not originate closing an STP connection. I believe this rule was intended to simplify the expander’s role in managing connections, and would eliminate any need for the expander to handle the case where the connection is closed while the SATA target is sending a FIS to the initiator.

However, this restriction can be a problem for an initiator that is connected to both SSP and STP targets. Since connections are opened and closed very frequently in SSP protocol, it will be common for initiators to allocate resources (such as DMA engines, data buffers, etc) for an SSP task, and attempt to reestablish a connection to a SSP target. A conflict can arise when an overriding (based on AWT) STP connection is established that also requires the same resources, after they have already been allocated for another purpose. I believe the main conflict is for resources that will result in the transmission of frames.

The following changes would allow an STP initiator to close a connection **only** after a condition in Serial ATA protocol where the target is waiting for a frame from the initiator. This would allow the initiator to have control of its resources in a similar fashion as SSP traffic, yet still prohibits cases where the SATA target would be sending frames to the initiator.

Proposed changes to the SAS rev 1 specification are:

Section 7.11.7 Closing a connection – Add a reference to section 9.3.1 in the second paragraph.

Section 7.15.3 Preparing to close an STP connection – Remove the first sentence restriction.

Section 9.3.1 SATA tunneling – Add the following paragraph:

An STP initiator device may originate the closing of a connection only after one of the following:

a) reception of a PIO Setup FIS with the D bit indicating a write to the target device;
b) reception of a DMA Activate FIS;
c) reception of any FIS from the target device that results in BSY set to zero, DRQ set to zero, and SERV set to one; or
d) reception of any FIS from the target device that results in the BSY set to zero, DRQ set to zero, and REL set to one.

(Need to work in task file reference into c and d)

Sata Reference Sequences

The following tables are intended for information only. They do not contain pertinent material relative to changes in the SAS specification.

All examples show two DATA FISes per command. The BSY and DRQ bits are from the controller's perspective of the shadow task file.

PIO Data Read

FIS	BSY	DRQ
	0	0
Reg H2D	1	0
PIO Setup D2H (D bit = read)	1	0
Data D2H	0	1
After Data completes	1	0
PIO Setup D2H	1	0
Data D2H	0	1
After Data completes	0	0

PIO Data Write

FIS	BSY	DRQ
	0	0
Reg H2D	1	0
PIO Setup D2H (D bit = write)	0	1
Data H2D	0	1
After Data completes	1	0
PIO Setup D2H	0	1
Data H2D	0	1
After Data completes	0?	0
Reg D2H	0	0

Non-Queued DMA Read

FIS	BSY	DRQ
	0	0
Reg H2D	1	0
Data D2H	1	0
After Data completes	1	0
Data D2H	1	0
After Data completes	1	0
Reg D2H	0	0

Non-Queued DMA Write

FIS	BSY	DRQ
	0	0
Reg H2D	1	0
DMA Activate D2H	1	0
Data H2D	1	0
After Data completes	1	0
DMA Activate D2H	1	0
Data H2D	1	0
After Data completes	1	0
Reg D2H	0	0

Queued DMA Read (two commands of two DATA FISes each, received out-of-order)

FIS	BSY	DRQ
	0	0
Reg H2D (command #1)	1	0
Reg D2H (release)	0	0
Reg H2D (command #2)	1	0
Reg D2H (release)	0	0
SetDevBits D2H (service bit)	0	0
Reg H2D (SERVICE command)	1	0
Reg D2H (tag #2)	0	1
Data D2H #2	0	1
After Data completes	0	1
Data D2H #2	0	1
After Data completes	0	1
Reg D2H (status)	0	0
Optional SetDevBits D2H (service bit)	0	0
Reg H2D (SERVICE command)	1	0
Reg D2H (tag #1)	0	1
Data D2H #1	0	1
After Data completes	0	1
Data D2H #1	0	1
After Data completes	0	1
Reg D2H (status)	0	0

Queued DMA Write (two commands of two DATA FISes each, received out-of-order)

FIS	BSY	DRQ
	0	0
Reg H2D (command #1)	1	0
Reg D2H (release)	0	0
Reg H2D (command #2)	1	0
Reg D2H (release)	0	0
SetDevBits D2H (service bit)	0	0
Reg H2D (SERVICE command)	1	0
Reg D2H (tag #2)	0	1
DMA Activate D2H	0	1
Data H2D #2	0	1
After Data completes	0	1
DMA Activate D2H	0	1
Data H2D #2	0	1
After Data completes	0	1
Reg D2H (status)	0	0
Optional SetDevBits D2H (service bit)	0	0
Reg H2D (SERVICE command)	1	0
Reg D2H (tag #1)	0	1
DMA Activate D2H	0	1
Data H2D #1	0	1
After Data completes	0	1
DMA Activate D2H	0	1
Data H2D #1	0	1
After Data completes	0	1
Reg D2H (status)	0	0

ATAPI PIO Read

FIS	BSY	DRQ
	0	0
Reg H2D (PACKET command)	1	0
PIO Setup D2H (D bit = write)	0	1
Data H2D (command packet)	0	1
After Data completes	1	0
PIO Setup D2H (D bit = read)	1	0
Data D2H	0	1
After Data completes	0	0
PIO Setup D2H (D bit = read)	1	0
Data D2H	0	1
After Data completes	0	0

ATAPI PIO Write

FIS	BSY	DRQ
	0	0
Reg H2D (PACKET command)	1	0
PIO Setup D2H (D bit = write)	0	1
Data H2D (command packet)	0	1
After Data completes	1	0
PIO Setup D2H (D bit = write)	0	1
Data H2D	0	1
After Data completes	1	0
PIO Setup D2H (D bit = write)	0	1
Data H2D	0	1
After Data completes	0?	0
Reg D2H	0	0

ATAPI Non-Queued DMA Read

FIS	BSY	DRQ
	0	0
Reg H2D	1	0
PIO Setup D2H (D bit = write)	0	1
Data H2D (command packet)	0	1
After Data completes	1	0
Data D2H	1	0
After Data completes	1	0
Data D2H	1	0
After Data completes	1	0
Reg D2H	0	0

ATAPI Non-Queued DMA Write

FIS	BSY	DRQ
	0	0
Reg H2D	1	0
PIO Setup D2H (D bit = write)	0	1
Data H2D (command packet)	0	1
After Data completes	1	0
DMA Activate D2H	1	0
Data H2D	1	0
After Data completes	1	0
DMA Activate D2H	1	0
Data H2D	1	0
After Data completes	1	0
Reg D2H	0	0

ATAPI Queued PIO Read
ATAPI Queued PIO Write
ATAPI Queued DMA Read
ATAPI Queued DMA Write