FROM: Jerry Hauck, jerry_hauck@ccm.sc.intel.com

TO: P1394a Working Group

DATE: September 19, 1997

RE: Minimum cycle_offset in cycle start packet

In preparation for discussion on Agenda Item 4.11 in Natick, here is some background information and a proposal for specifying a minimum cycle_offset value in the cycle start packet.

cycleInconsistent Background

OHCI link designs (and I imagine some others) apply a few consistency checks on incoming cycle start packets. One of the checks compares the cycle_count and second_count fields of the cycle start packet with the link's internal CYCLE_TIME register. If these fields are not equal, a cycleInconsistent interrupt is asserted.

In the case of OHCI, the cycleInconsistent interrupt confirms that the cycle master's concept of time has changed (probably due to a new cycle master being selected after a bus reset). One effect of a cycleInconsistent is that isochronous transmit or receive streams which were programmed to start in a specific isoch interval may now have to be rescheduled. (Due to host bus latencies, OHCI implementations typically schedule work up to two isoch intervals ahead of "real" time with the assumption that cycle time can be accurately predicted. A cycleInconsistent then temporarily invalidates the assumption.)

In summary, without the cycleInconsistent mechanism, it would be difficult if not impossible for software to ensure that a specific packet arrive at a target during a specific isoch interval. This capability is near essential for isoch protocols which require an absolute cycle number to be encoded in each packet.

Problem Statement

As outlined above, the cycleInconsistent test relies on cycle slaves seeing their internal cycle sync event before receiving the corresponding cycle start packet from the master. If the order of these events are exchanged during normal operation, false cycleInconsistents will regularly occur.

Ideally, the exchange of cycle syncs and cycle starts never occurs since the CYCLE_TIME registers in the master and slaves are synchronized and differ by a fixed offset representing the propagation delay between the master and each slave. When a cycle master detects an internal cycle sync #N, a slave will observe its internal cycle sync #N a propagation delay later. In response to cycle sync #N, the master will generate cycle start #N. A slave then observes this cycle start #N a propagation delay later. Thus, cycle slaves ideally always see cycle sync #N followed by cycle start #N.

However, jitter in delivery of cycle start packets must also be considered. For example, if cycle start N arrives at a slave with worst case propagation delay and cycle start N+1 arrives with best case delay, then the difference in time between cycle sync N+1 and cycle start N+1 will be shorter at the slave than at the master. A similar scenario exists in which the difference in time appears greater at the slave than at the master. In the limiting cases when the cycle master transmits the cycle start packet too close to it's own cycle sync, the order of the events can become interchanged when observed at a given cycle slave.

A Possible "Fix"

The exchange of events at the cycle slave can be completely avoided if the cycle master can guarantee that it's internal cycle sync and cycle start generation are always separated by at least the worst case delivery jitter of the bus. Specifically,

a) The minimum cycle_offset contained in the cycle start packet must be larger than the delivery jitter

b) The maximum cycle_offset contained in the cycle start packet must be less than (3071 - delivery jitter).

In turn, delivery jitter consists of PHY delay jitter and clock drift between the master and slave. Looking at the clock drift component first, I propose we consider the maximum drift which can occur over 2 isoch intervals (protecting us even in the event a bus reset clobbers one cycle start packet). With a maximum difference of 200 ppm between master and slave, the maximum drift over 250 uSec is 50 nSec.

In considering PHY jitter, a traditional 1394 analysis would consider a 20 nSec per PHY jitter multiplied by a maximum hop count of 16 for a total PHY jitter of 320 nSec. However, P1394a's greater independence from a specified maximum hop count and the possibility of PHY's with larger propagation jitter suggests we consider a larger cumulative jitter. I'll propose 640 nSec.

Combining PHY jitter and clock drift, the worst case delivery jitter for cycle start packets is 640 + 50 = 690 nSec. In units of the cycle_offset field, 690 nSec ~= 17. So our fix becomes:

a) The minimum cycle_offset = 18.

b) The maximum cycle_offset = 3053.

Except for some extreme scenarios in which a bus reset interrupts cycle start generation, I believe rules a) and b) above are already met by today's links. Consequently, I believe this issue becomes one of specmanship and should not need to affect current silicon designs.

Rule a) should be practically met if you consider the time it takes to issue a PriReq and the duration of DATA_PREFIX the PHY enforces before granting the bus to the link. For completeness we should specify a minimum (in the anticipation of integrated PHY/LINKs.)

Rule b) should be practically met considering the prioritized arbitration the root possesses. Cycle starts can normally be delayed at most by a maximum sized asynchronous packet, so the cycle_offset should be no where near the maximum value. The only scenario which challenges this is a bus reset which ends near an upcoming cycle sync event. I willing to accept the low probability of receiving a false cycleInconsistent in this scenario.

Specific P1394a Proposal

I propose some text along the lines of "If bus arbitration for and transmission of the cycle start packet communicating the start of interval N both begin within isochronous interval N, then the minimum cycle_offset specified by the cycle start packet shall be 17." I think a note would also be appropriate to indicated that "Due to the asynchronous occurrence of bus resets, it is possible in rare circumstances for transmission of a cycle start packets to be delayed into the interval subsequent to the one in which arbitration was requested. That is, a second cycle sync occurs before the first cycle Start packet is formed and transmitted. In such rare circumstances, there is no requirement for the link to enforce the minimum cycle_offset value."

The intent of my cumbersome wording is to free link designers from additional hardware enforcement of the minimum cycle_offset under all possible conditions. I think our goal is to ensure that, during normal operating conditions in which cycle syncs and cycle starts enjoy a matched 1:1 ratio, false cycleInconsistents are eliminated.

Hardware could be designed in a manner which guaranteed the min and max values at all times, but I don't think the effort is justified and I don't feel the specification should require it.