To: X3T9.2 Committee Membership

From: Edward A. Gardner, Digital Equipment Corporation

Subject: Clarifications to SCAM revision 5 (X3T9.2/93-109r5)

From the questions I have received in the past several weeks, it is clear that some sections of SCAM (document X3T9.2/93-109r5) are not as clear as they ought to be. The following is an attempt to clarify the more problematic areas.

Section 4.4. Transfer Cycles

During a transfer cycle, described in section 4.4, all devices that are participating in the SCAM protocol must assert and release DB5-7. It makes no difference whether a node is sending data, receiving data, or ignoring DB0-4, all participating nodes must perform the handshake on DB5-7. In particular:

1. After a master node initiates or responds to SCAM selection, it shall perform the handshake on DB5-7 until such time as that master choses to exit or terminate the SCAM protocol (at which time it releases all signals).
2. After a slave node initiates or responds to SCAM selection, it shall perform the handshake on DB5-7 until it is assigned an ID, it determines that no master nodes are participating (by detecting that C/D is released), or it internally resets (perhaps due to an error). In all three cases it exits the SCAM protocol and releases all signals.

Briefly, why this is necessary, this maintains the property that every participating device is always asserting at least one of the three handshake signals (DB5-7). If a participating device did not perform the handshake, there would be some time interval when it was asserting no signals. During that time interval the other participating devices (assuming they are sufficiently fast) could perform an arbitrary number of transfer cycles without the first device having time to notice the cycles. This violates the requirement that transfer cycles be fully interlocked and asynchronous. Having every participating device always assert at least one handshake signal provides an interlock or synchronization point when that signal is released.

In the next revision of SCAM I plan to make the following changes to the description of the nine transfer cycle steps. I welcome suggestions on further wording changes that would be helpful.

1. Place data on DB0-4, if the device is sending data. All devices assert DB5.
2. All devices release DB7.
3. Wait until DB7 is released by all other devices, using wired-or glitch filtering.
4. Read and latch data from DB0-4. All devices assert DB6.
5. All devices release DB5.
6. Wait until DB5 is released by all other devices, using wired-or glitch filtering.
7. Release or change DB0-4. All devices assert DB7.
8. All devices release DB6.
9. Wait until DB6 is released by all other devices, using wired-or glitch filtering.

Software Wired-or Glitch Filtering

Several people have found this obvious, others have not. I welcome suggestions whether the following ought to be included in the SCAM document.

The SCAM protocol specifies in many places that a device shall wait until a signal is released, using wired-or glitch filtering. The wired-or glitch filtering can be performed with hardware or through a software polling loop.

Hardware wired-or glitch filtering is essentially identical to the hardware used to detect bus free conditions. Borrowing the wording for bus free, a device may determine that a signal is released, using wired-or glitch filtering, by observing that the signal is continuously false for at least a bus settle delay. Note that continuous observation of the signal requires dedicated hardware, it cannot be done with a software polling loop.

Alternately, SCAM devices may perform wired-or glitch filtering using a software polling loop. This is considerably slower than hardware, but has adequate performance for the SCAM protocol. I expect that all SCAM implementations for the foreseeable future will use software polling for wired-or glitch filtering. This is certainly true for all SCAM implementations based on existing hardware.

The basic approach is to construct a polling loop that repeatedly checks if a signal is released. If the signal is observed to be released for a sufficient number of consecutive iterations, the device may conclude that the signal has indeed been released by all devices. If the signal is ever observed to be asserted, then the iteration count is reset and the polling loop restarted.

The following iteration count calculation uses a simplified pessimistic model of wired-or glitches. As such it may calculate an unnecessarily large iteration count. However, its performance is adequate for SCAM, and the pessimistic assumptions should lead to more robust implementations.

The iteration count calculation is based on two assumptions. First, a single device releasing a wired-or signal may cause a wired-or glitch (that is, may cause the signal to falsely appear released) whose duration is at most a bus settle delay. That implies that if the signal is observed to be released at two samples taken more than a bus settle delay apart, then either the signal has truly been released by all devices, or the two samples occurred during two independent wired-or glitches caused by two separate devices releasing the signal. The second assumption is that the bus is electrically limited to a maximum of 32 devices, so at most 32 wired-or glitches may occur for each signal release.

Given these two assumptions, the iteration count is calculated from the minimum time between consecutive samples, the bus settle delay, and the maximum number of devices on the bus (32). One of the following two cases applies:

1. If the minimum time between samples is greater than the bus settle delay, then 32 consecutive samples are sufficient. Each wired-or glitch can affect at most one sample.
2. If the minimum time between samples is less than or equal to the bus settle delay, calculate:

   N = (bus settle delay) / (minimum sample interval)

   rounding N up to the next higher integer. N*32 consecutive samples are sufficient. Each wired-or glitch can affect at most N samples.

End of document Clarifications to SCAM revision 5 (X3T9.2/93-109r5), X3T9.2/93-173r0.