Dino 3.1 (1FC3-0004) Errata Listing

Revised: September 19, 1997

1) Coalescing of Config Writes

Description: If coalescing is enabled, and a write to the PCI_CONFIG_DATA register is followed by a write to the PCI_IO_DATA register, then these writes may be coalesced together into one two-word write to PCI configuration space.

Fix: There are two ways to avoid this situation. The first solution is to insert another transaction, such as a read, between the two writes. The second solution is to disable coalescing in the BRDG_FEAT register.

2) Dino PCI bus fights in external arbitration mode

Description: If Dino is in external arbitration mode, then Dino may incorrectly assume that it owns the PCI bus one clock after GNT# goes away. If the arbiter deasserts Dino's GNT# coincident with another GNT# on the last cycle of a frame, then Dino may drive fight with the other PCI device.

Fix: Although it is possible to design an external arbiter that works around this problem in Dino, FSL does not plan to support external arbitration mode in Dino.

3) Dino deasserts internal PCI GNT# and asserts another GNT# without 1 state delay on idle PCI bus

Description: If the PCI bus is idle, then the PCI arbiter should not allow one GNT# to deassert coincident with another GNT# being asserted to prevent bus contention on the AD and PAR lines. If Dino is in the address stepping phase of a configuration read or write transaction and a PCI device requests the PCI bus, the arbiter may deassert Dino's GNT# and assert the PCI device GNT# without a 1 state delay. If Dino does I/O or memory transactions, then Dino will not do address stepping and the problem can not happen. If two PCI devices arbitrate for an idle PCI bus, they can not cause this problem to happen. (Even if one of the PCI devices is doing address stepping.)

Fix: PCI services in the kernel must always disable all non-Dino PCI devices in the PAMR register before doing a configuration transaction. After the configuration transaction completes, the PAMR should be restored to its previous value.

4) Dino's PCI Arbiter ignores external device's request after starting a PCI transaction to a coalesable address when the PCI BUS was parked on Dino.

Description: If the PCI BUS is parked on Dino and the next transtion is to a coalesable address the PCI Arbiter's round robin does not update the next pointer to the correct Device.

Fix: Dino's PAPR register needs to programed to 0x7f.

5) Illegal Target Aborts

Description: If Dino is the target of a DMA transaction on PCI and enters fatal mode on the last phase of the DMA transaction, then a target abort is signaled by Dino on PCI even though it is illegal since the transaction has ended. The 'Signaled Target Abort' bit in the PCI status register is not asserted in this case.

Fix: There is no fix for this issue. Since the problem only ocures after Dino has gone fatal, the importance of this issue is judged to be very low.

6) 2, 4, or 8-word GSC reads may generate extra data on PCI

Description: If Dino or Cujo receives a 2, 4, or 8-word read from the GSC bus, then it may generate extra PCI transactions. This problem is caused by a protocol problem with one of the asychronous FIFOs and can only happen when the GSC clock runs slower than the PCI clock. All present systems that use Cujo run the GSC clock fast enough to prevent this problem. There are no GSC hosts that can generate a 4-word or 8-word reads, so we do not need to worry about them. A 2-word read is possible, but no driver is believed to use them. We have only seen this failure in Verilog simulations. We have not been able to produce it in a real system.

Fix: Either keep the GSC clock running faster than the PCI clock or avoid generating 2-word PIO reads to PCI memory space.

7) Parity errors possible while reading the Interrupt Level Register.

Description: Dino may cause a GSC parity error when one or more of the external PCI interrupt lines change state at the same time a PIO read of the Interrupt Level Register (ILR) occurs on the GSC bus. (The ILR is used to read the state of the external PCI interrupt lines.) The ILR is a built out of synchronizer cells to prevent metastability on the PCI interrupt signals. The synchronizer cells outputs are clocked on the opposite phase of the GSC clock then all of the other registers in Dino. This set up a race condition between two paths from the ILR to the GSC pads. One path, called the data path, goes from the ILR to the GSCAD pads, the other path, called the parity path, goes from the ILR through a parity generator to the GSCPAR pad. If the GSC clock slows down or the Cujo part is too fast, then there is a risk that the data WILL make it to the GSCAD pads, but the parity WILL NOT make it to the GSCPAR pad. This will cause a parity error.

Fix: Fast, nominal and slow Dino's at 33MHz GSC clock will not fail. The GSC clock for Dino should never go below 33MHz. A screen for this problem has been added to Dino's ICBD test vectors in case ICBD's process skews beyond fast in the future.

8) Possible deadlock when Delayed Competion is enabled.

Description: If delayed competion is enabled (DCOMP = 1 in the Bridge Feature Enable Register), then it is possible for Dino to deadlock. For the deadlock to occur, Dino must be connected to a PCI device that will retry all PIO read transactions until a pending DMA read completes (this is the case for PCI bridges), and the following must happen:

- The CPU masters a PIO read to a PCI device connected to Dino.
- The PCI device begins a DMA read.
- The DMA read splits the PIO read and sends a read request on the GSC bus.
- A PCI interreupt line asserts before the DMA read returns.

At this point, Dino will not split the PIO read to send the interrupt to the CPU, the PIO read will be retried by the PCI device until the DMA read completes, and the DMA read will not complete until the interrupt is sent to the CPU. This deadlock will cause the PIO read to time-out and cause an HPMC.

Fix: Firmware and PCI services do not currently enable delayed competions. Delayed completion is no longer supported.