Appendix B

Challenge DMA with Multiple IO4 Boards

In late 1995 a subtle hardware problem was identified in the IO4 board that is the primary I/O interface subsystem to systems using the Challenge/Onyx architecture. The problem can be prevented with a software fix. The software fix is included in all device drivers distributed with IRIX 6.2 and a software patch is available for IRIX 5.3. However, some third-party device drivers also need to incorporate the sofware fix. This appendix explains the IO4 problem as it affects device drivers produced outside Silicon Graphics.

The issue in a nutshell: if you are responsible for a kernel-level device driver for a DMA device for the Challenge/Onyx architecture, you probably need to insert a function call in the driver interrupt handler.

The IO4 Problem

The IO4 hardware problem involves a subtle interaction between two IO4 boards when they perform DMA to the identical cache line of memory. If one IO4 performs a partial update of a 128-byte cache line, and another IO4 accesses the same cache line for DMA between partial updates, the second IO4 can suffer a change to a different, unrelated cache line in its on-board cache. That modified cache line may not be used again, but if it is used, invalid data can be transferred.

It is important to note that the IO4 problem is specific to interactions between multiple IO4 boards. It does not affect memory interactions between CPUs, or between CPUs and IO4s. Cache coherency is properly maintained in these cases.

An unusual coincidence is required to trigger the modification of the IO4 cache memory; then the modified cache line must be used for output before the error has any effect. The right combinations are sufficiently rare that many systems with multiple IO4 boards have never enountered it. For example, the problem has occurred on a system that acted as a network gateway between ATM and FDDI network, with ATM and FDDI adapters on different IO4 boards; and it has been seen when “raw” (not filesystem) disk input was copied to a tape on a different IO4.

Software Fix

The software solution involves a number of behind-the-scenes changes to kernel functions that manage I/O mapping. However, for third-party device drivers, the fix to the IO4 problem consists of ensuring that any IO4 doing DMA input (when a device uses DMA to write to memory) flushes its cache on any interrupt. This change has been made in IRIX 6.2 to all device drivers supplied by Silicon Graphics.

A patch containing all necessary fixes is available for IRIX 5.3. Contact the Silicon Graphics technical support line for the current patch number for a particular system.

Software Not Affected

As a result of hardware design and software fixes, none of the following kinds of software are affected by the problem:

- Code using PIO to manage a device.

  The IO4 problem cannot be triggered by PIO, either at the user or kernel level.
- User-level code using the udmalib library or the dslib SCSI library.

  These libraries for user-level DMA contain the fix, or use kernel functions that are fixed.
- User-level code based on user-level interrupts (ULI) or external interrupts.

  These facilities are not relevant to the IO4 problem.

Among kernel-level drivers, only drivers that directly program DMA can be affected. STREAMS drivers are not affected; nor are pseudo-device drivers; nor are drivers that use only PIO and memory mapping. Drivers that are not used on Challenge-architecture machines are not affected; for example an EISA-bus driver cannot be affected.

SCSI drivers that use the host adapter interface (see "Host Adapter Facilities" on page 354) are also not affected. Silicon Graphics host adapter drivers contain the fix. Host adapter drivers from third parties may need to be fixed, but this does not affect drivers that rely on the host adapter interface.

Drivers that do only block-mode I/O for the filesystem, and do not implement a character I/O interface (or do not support the character I/O interface using DMA) are not affected. This is because the filesystem always requests I/O in cache-line-sized multiples to buffers that are cache-aligned.

Fixing the IO4 Problem

A kernel-level device driver for a device that uses DMA in a Challenge-architecture system probably needs to make one change to guard against the IO4 problem.

In order to preclude any chance of data corruption, drivers that are affected must ensure that the IO4 flushes its cache following any DMA write to memory (input from a device). This is done by calling a new kernel function, **io4_flush_cache()**, in the interrupt routine immediately following completion of any DMA.

The prototype of **io4_flush_cache()** is

```
int io4_flush_cache(caddr_t any_PIO_mapaddr);
```

The argument to the function is any value returned by **pio_mapaddr()** that is related to the device doing the DMA. The kernel uses this address to locate the IO4 involved. The returned value is 0 when the operation is successful (or was not needed). It is 1 when the argument is not a valid address returned by **pio_mapaddr()**.

The function should be called immediately after the completion of a DMA input to memory. Typically the device produces an interrupt at the end of a DMA, and the function can be called from the interrupt hander. However, some devices can complete more than one DMA transaction per interrupt, and **io4_flush_cache()** should be called when each DMA completes. Put another way, if it is possible that a data transfer completed after an interrupt, then the driver should call **io4_flush_cache()** before marking the transaction as complete.

The **io4_flush_cache()** function does nothing and returns immediately in a machine that has only one IO4 board, and in a machine in which all IO4 boards have the hardware fix.

The kernel's VME interrupt handler calls **io4_flush_cache()** once on each VME interrupt. Thus a VME device driver only needs to call **io4_flush_cache()** in the event that it handles the completion of more than DMA transaction per interrupt. For example, a VME-based network driver that handles multiple packets per interrupt should call **io4_flush_cache()** once for each packet that completes.

Since this problem only affects Challenge/Onyx systems (including POWER Challenge, POWER Onyx, and POWER Challenge R10000), the software fix can and should be conditionally compiled on the compiler variable EVEREST, which is set by */var/sysgen/Makefile.kernio* for the affected machines (see "Using /var/sysgen/Makefile.kernio" on page 228).

The following is a skeletal example of fix code for a hypothetical driver:

```
#ifdef EVEREST
extern void io4_flush_cache(void* anyPIOMapAddr);
#endif
caddr_t some_PIO_map_addr;
hypothetical_edtinit(...)
{
...
   some_PIO_map_addr = pio_mapaddr(my_piomap, some_dev_addr)
...
}
hypothetical_intr(...)
{
...
#ifdef EVEREST
   io4_flush_cache(some_PIO_map_addr);
#endif
...
}
```

For another example, see the code of the example VME device driver under “Sample VME Device Driver” on page 334.