

Re: [RFC] MMIO accessors & barriers documentation

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-09/msg03291.html>

- *From:* Benjamin Herrenschmidt <benh@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 13 Sep 2006 07:22:05 +1000
-

Or you do the sane thing and just not allow two threads of execution access to the same I/O device at the same time.

Why ? Some devices are designed to be able to handle that...

Now compare this with the similar scenario for "normal" MMIO, where we do store;sync (or sync;store or even sync;store;sync) for every writel() — exactly the same problem.

What problem ? "Normal" MMIO doesn't get combined, thus there is no problem. Of course there is no guarantee of ordering of the stores from the 2 CPUs unless there is a spinlock etc etc... but we are talking about a case where that is acceptable here. However, combining is not.

Better lock at a higher level than just per instruction.

Some devices that want to support multiple clients at the same time have multiple identical "register files", one for each client, to prevent this and other problems (and it's useful anyway).

Yes, they do, and what happen if those register "files" happen to be consecutive in the address space and the CPU suddenly combines a store to the last register of one "file" and an unrelated store from another thread to the first register of the other ?

This is a very specific problem that has nothing to do with your "grand general case" which means that at least on Cell, you cannot use explicit barriers to guarantee the absence of write combining. That's as simple as that. All I need to figure out now is if that problem is specific to one CPU implementation or more general, in which case, we'll have to figure out some way to provide an interface.