

Need help chasing a weird problem

Source: <http://linux.derkeiler.com/Newsgroups/comp.os.linux.development.system/2004-04/0516.html>

From: Kris (qowkolun.eckris_at_sfm.cs.ualberta.ca)

Date: 04/29/04

Date: Thu, 29 Apr 2004 15:15:21 -0600

Dear All:

I am having a weird problem with a somewhat exotic PCI device, whose driver has been ported from Windows (pardon my French). The description may be lengthy, but most of it is just my literary exercise to get the illustrious audience into the mood. The problem has been narrowed down so much that normally it would be trivial to fix ("faulty hardware" would be the obvious diagnosis). The only problem with this conclusion is that the @#\$\$%^ thing does work under Windows (W2K to be precise), yet it consistently refuses to work under Linux.

Summary:

The machine is an IBM x343.

There are 3 identical PCI cards plugged into 3 available high-riser PCI slots. The kernel is 2.4.18-14, but other kernels have been tried (2.2.x, 2.5.x, 2.6.x) yielding the same problem.

Two of the three cards work, the third one (the last card on the bus) gives weird symptoms which appear to be related to some glitch in accessing its I/O ports (or one port to be exact).

Clarification:

While, in my modesty, I have no claims to the "guru" standing (otherwise I wouldn't be seeking your advice, would I), I am well acquainted with the kernel, and I feel that I have done my homework). So suggestions like, "Did you try a different card?", "Did you read the specifications?", and so on, are not going to be extremely helpful.

Details:

Except for the particular selection of the machine (x343), the setup is not new. The cards (I mean, identical and/or exactly the same cards) have been working smoothly in a high-reliability environment

comp.os.linux.development.system: Need help chasing a weird problem

with kernels 2.2.x and 2.4.x on a variety of PC equipment, including SMP and single-CPU kernels. There is a single application, whose specifics are completely irrelevant, and a driver, which has been ported from Windows. The problem occurs at the very beginning (when the cards are initialized), so even the details of the driver are irrelevant. I do have the source code of the Windows driver, and I know perfectly what it does.

The PCI interface of the card is built on AMCC S5920 and, logically, looks as follows (I am listing the driver's dump of the resources allocated to all three cards):

```
CARD 0
=====
BUS ADDR:||||||| c46b5be0
DEVFN:||||||| 40
HDR_TYPE:||||||| 0
ROM_BASE:||||||| 30
DMA_MASK:||||||| ffffffff
CUR_STATE:||||||| 4
IRQ:||||||| 31
NAME:||||||| PCI device 10e8:8328 (Applied Micro
Circuits Corp.)
SLOT:||||||| 01:08.0
COMMAND:||||||| 3
RES [0]:||||||| 2800 287f 101
RES [1]:||||||| feb90000 feb97fff 200
RES [2]:||||||| 0 0 0
RES [3]:||||||| 2880 289f 101
RES [4]:||||||| 28a0 28a3 101
RES [5]:||||||| 0 0 0
RES [6]:||||||| 0 0 0
RES [7]:||||||| 0 0 0
RES [8]:||||||| 0 0 0
RES [9]:||||||| 0 0 0
RES [a]:||||||| 0 0 0
RES [b]:||||||| 0 0 0

CARD 1
=====
BUS ADDR:||||||| c46b5be0
DEVFN:||||||| 48
HDR_TYPE:||||||| 0
ROM_BASE:||||||| 30
DMA_MASK:||||||| ffffffff
CUR_STATE:||||||| 4
IRQ:||||||| 30
NAME:||||||| PCI device 10e8:8328 (Applied Micro
Circuits Corp.)
SLOT:||||||| 01:09.0
COMMAND:||||||| 3
```

```
RES [0]:|||||| 2c00 2c7f 101
RES [1]:|||||| feb80000 feb87fff 200
RES [2]:|||||| 0 0 0
RES [3]:|||||| 28c0 28df 101
RES [4]:|||||| 28a4 28a7 101
RES [5]:|||||| 0 0 0
RES [6]:|||||| 0 0 0
RES [7]:|||||| 0 0 0
RES [8]:|||||| 0 0 0
RES [9]:|||||| 0 0 0
RES [a]:|||||| 0 0 0
RES [b]:|||||| 0 0 0
```

CARD 2

=====

```
BUS_ADDR:|||||| c46b5be0
DEVFN:|||||| 50
HDR_TYPE:|||||| 0
ROM_BASE:|||||| 30
DMA_MASK:|||||| ffffffff
CUR_STATE:|||||| 4
IRQ:|||||| 29
NAME:|||||| PCI device 10e8:8328 (Applied Micro
Circuits Corp.)
SLOT:|||||| 01:0a.0
COMMAND:|||||| 3
RES [0]:|||||| 3000 307f 101
RES [1]:|||||| feb70000 feb77fff 200
RES [2]:|||||| 0 0 0
RES [3]:|||||| 28e0 28ff 101
RES [4]:|||||| 28a8 28ab 101
RES [5]:|||||| 0 0 0
RES [6]:|||||| 0 0 0
RES [7]:|||||| 0 0 0
RES [8]:|||||| 0 0 0
RES [9]:|||||| 0 0 0
RES [a]:|||||| 0 0 0
RES [b]:|||||| 0 0 0
```

Card 2 is the one causing problems. The operation of initializing the card consists in loading an FPGA program via the single-byte port described by RES[4]. This operation involves a trivial, explicitly clocked, extremely tolerant, serial protocol using three pins (three bits of the port). Once the program has been loaded, the other ports (application ports) become responsive, and the card becomes operable.

This does not want to happen for Card 2. Although the port (RES[4]) appears to respond sensibly to the commands (in a way similar to a healthy card), the FPGA program is not loaded correctly, and the card doesn't work.

What's the big deal, I hear you say. The card is broken, or, assuming I have tried putting other cards into the same PCI slot, the PCI slot is broken. Or, perhaps, there is a timing problem with the serial operations needed to load the FPGA program that somehow shows up for the last card on the bus. Or, maybe, there is an unreported resource conflict in the kernel and the port is messed up by something. All these natural suspicions would make perfect sense at this point, but I have also determined that:

1. Under Windows 2000, the driver, which essentially mimics the Linux driver, is able to load the same FPGA program into all cards without a slightest problem.
2. The resources allocated to the cards under Windows (as shown by the device manager) are strictly identical to those allocated under Linux (BIOS takes care of that).
3. Listen carefully to this one. If the card is initialized under Windows and then, without powering the machine down, the system is switched to Linux and the FPGA loading stage of the driver is subsequently skipped, **ALL THREE CARDS WORK**. Repeat: all three cards work under Linux without a further glitch, if the references to port 0x28a8 needed to store the microprogram in the FPGA chip are carried out under Windows. Note that the sole purpose of that port is to provide a means to load the FPGA program, and it is not used during normal operation of the card.

The results of those experiments seem to suggest to me the following three possibilities:

1. There is a difference in the way the two drivers, i.e., the Windows driver and the Linux driver, load the microprogram.
2. There is an unreported resource conflict or something else that Linux kernel does wrong with respect to the initialization of the PCI devices.
3. There is something that Linux misses, i.e., doesn't do while it should. Windows happens to know the trick that is needed to make all the devices work.

Possibility 1 has been eliminated after a lengthy experimental study involving timing adjustments and various modifications to the loading program (slowing down port I/O among other things). The healthy cards have been found to be extremely tolerant to the timing parameters, yet the failing card hasn't been able to respond once. Again, note that exactly the same Linux driver and exactly the same application work with the same three cards on a different PC. There is practically no way to suppose that a bug in the driver or in the application is responsible for the problem.

Possibility 2 has been effectively eliminated by moving the operation of loading the microprogram to the very beginning of the relevant activities in the kernel, i.e., before it starts allocating resources to other devices or even recognizing the PCI configuration. I inserted the code for loading the FPGA program into the kernel BEFORE it even learns that any bus devices are there. The I/O ports are known, so you don't have to be aware of the PCI interface to start talking to the devices. This does work nicely for the healthy cards, but the third card still doesn't work. Note that once the FPGA program is loaded successfully (under Windows), the card works under Linux.

As we have no access to Windows' source code (and, additionally, am not horribly well familiar with that system), I cannot investigate possibility 3, and all I have is circumstantial evidence. Note that the cards are custom made, and none of their idiosyncrasies can be possibly known to W2K; all the wisdom needed to make them work must be confined to the driver.

It appears that the BIOS on the x343 makes it impossible for the kernel to change the PCI resource allocation that was originally done by the BIOS. I tried changing the ports allocated by the BIOS (by writing into the device's configuration space), but such requests were ignored and the new ports were completely not responsive.

For those of you who have managed to read so far, let me offer a reward (a non-virtual drink of your choice in quantities considered decent by reasonably civilized people) for any suggestion that will prove constructive and get me closer to solving this mystery. I have no access to a logic analyzer that would let me capture the complete history of signals on the PCI bus. The problem is not critical enough to warrant a purchase of such equipment. But a few drinks is a completely different matter. So could you please tell me what might happen under Windows that doesn't happen under Linux?

Thanks and best regards,

Kris