

[PATCH 1/1] IBM PPC EMAC driver:improved support for PHY, resending

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-04/msg10130.html>

- *From:* "Jeff Haran" <jharan@xxxxxxxxxxxx>
 - *Date:* Thu, 26 Apr 2007 18:18:55 -0700
-

From: Jeff Haran <jharan@xxxxxxxxxxxx>

Resending with Outlook patch mangling hopefully corrected (Maybe I should write a HOWTO, this was harder than fixing the driver).

This patch fixes some problems I found while debugging the IBM EMAC driver for PPC32 systems.

The first problem was in the function that configures the PHY for autonegotiation, `genmii_setup_aneg()`. The original code does a read/modify/write of the autonegotiation advertizement register (reg 4), followed by a read/modify/write of the control register (reg 0). While the original code follows the proper procedure as per reading the IEEE specs, what I found is that on at least one PHY model (National DP83843) the read of the control register comes back with the soft reset bit set (bit 15). Because of the read/modify/write operation, this causes the write to write a 1 back to the reset bit, which initiates a software reset of the PHY. This software reset causes the PHY to return to its power up state which advertizes all modes of operation, thus negating the write to the autoneg advertizement register. The modification is to spin reading the control register until the soft reset bit is clear before doing the modify/write. I guess this bit is in reality more of the "device busy" bit on at least some PHYs.

The second problem was in the function that configures the PHY for forced operation, `genmii_setup_forced()`. The original code initiates a software reset operation via a write of a 1 to bit 15 of the control register (reg 0), but then proceeds to do a second write to that same register without waiting until that reset bit is cleared by the PHY itself (which according to the IEEE specs indicates that the PHY reset is complete). This is a violation of how one is supposed to use this software reset feature of these PHYs and I believe was the cause of mysterious, difficult to reproduce link failures that we've observed on some of our systems that use this driver. The fix is to modify the function so that it spins waiting for the reset bit to clear after doing the soft reset and before doing the subsequent write. Since this modification, we haven't seen the mysterious link failures, though they were so rare its difficult to say at this point whether this was the cause.

I also added some error handling and reporting for the abnormal case where the reset bit never clears from the soft reset operation.

[PATCH 1/1] IBM PPC EMAC driver:improved support for PHY, resending

Applied to kernel version 2.6.21.

Signed-off-by: Jeff Haran <jharan@xxxxxxxxxxxx>

--- linux-2.6.21/drivers/net/ibm_emac/ibm_emac_phy.c.orig

2007-04-25 20:08:32.000000000 -0700

+++ linux-2.6.21/drivers/net/ibm_emac/ibm_emac_phy.c 2007-04-26

14:42:09.562996000 -0700

@@ -22,8 +22,12 @@

```
#include <asm/ocp.h>
```

```
+#include "ibm_emac_core.h"
```

```
#include "ibm_emac_phy.h"
```

```
+#define NL "\n"
```

```
+#define PHY_DBG(f,x...) printk("emac" f, ##x)
```

```
+
```

```
static inline int phy_read(struct mii_phy *phy, int reg)
```

```
{
```

```
return phy->mdio_read(phy->dev, phy->address, reg);
```

```
@@ -34,11 +38,34 @@ static inline void phy_write(struct mii_
```

```
phy->mdio_write(phy->dev, phy->address, reg, val);
```

```
}
```

```
-int mii_reset_phy(struct mii_phy *phy)
```

```
+/*
```

```
+ * polls MII_BMCR until BMCR_RESET bit clears or operation times out.
```

```
+ *
```

```
+ * returns:
```

```
+ * >= 0 => success, value in BMCR returned to caller
```

```
+ * -EBUSY => failure, RESET bit never cleared
```

```
+ * otherwise => failure, lower level PHY read failed
```

```
+ */
```

```
+
```

```
+static int mii_spin_reset_complete(struct mii_phy *phy)
```

```
{
```

```
int val;
```

```
int limit = 10000;
```

```
+ while (limit-->0) {
```

```
+ val = phy_read(phy, MII_BMCR);
```

```
+ if ((val >= 0) && ((val & BMCR_RESET) == 0))
```

```
+ return val; /* success */
```

```
+ udelay(10);
```

```
+ }
```

```
+
```

```
+ return (val < 0) ? val : -EBUSY;
```

```
+}
```

```
+
```

```
+int mii_reset_phy(struct mii_phy *phy)
```

```
+{
```

```

+ int val;
+
val = phy_read(phy, MII_BMCR);
val &= ~BMCR_ISOLATE;
val |= BMCR_RESET;
@@ -46,16 +73,17 @@ int mii_reset_phy(struct mii_phy *phy)

udelay(300);

- while (limit--) {
- val = phy_read(phy, MII_BMCR);
- if (val >= 0 && (val & BMCR_RESET) == 0)
- break;
- udelay(10);
+ val = mii_spin_reset_complete(phy);
+
+ if (val < 0) {
+ PHY_DBG("%d: reset_complete failed in reset %d" NL,
+ ((struct ocp_enet_private *)
+ (phy->dev->priv))->def->index, val);
+ } else {
+ if (val & BMCR_ISOLATE)
+ phy_write(phy, MII_BMCR, val & ~BMCR_ISOLATE);
+ }
- if ((val & BMCR_ISOLATE) && limit > 0)
- phy_write(phy, MII_BMCR, val & ~BMCR_ISOLATE);

- return limit <= 0;
+ return val < 0;
+ }

static int genmii_setup_aneg(struct mii_phy *phy, u32 advertise)
@@ -102,9 +130,18 @@ static int genmii_setup_aneg(struct mii_
+ }

/* Start/Restart aneg */
- ctl = phy_read(phy, MII_BMCR);
- ctl |= (BMCR_ANENABLE | BMCR_ANRESTART);
- phy_write(phy, MII_BMCR, ctl);
+ /* on some PHYs (e.g. National DP83843) a write to MII_ADVERTISE
+ * causes BMCR_RESET to be set on the next read of MII_BMCR,
+ which
+ * if not checked for causes the PHY to be reset below */
+ ctl = mii_spin_reset_complete(phy);
+
+ if (ctl < 0) {
+ PHY_DBG("%d: reset_complete failed in setup_aneg %d" NL,
+ ((struct ocp_enet_private *)
+ (phy->dev->priv))->def->index, ctl);
+ } else {
+ ctl |= (BMCR_ANENABLE | BMCR_ANRESTART);

```

[PATCH 1/1] IBM PPC EMAC driver:improved support for PHY, resending

```
+ phy_write(phy, MII_BMCR, ctl);
+ }

return 0;
}
@@ -118,13 +155,13 @@ static int genmii_setup_forced(struct mi
phy->duplex = fd;
phy->pause = phy->asym_pause = 0;

+ /* First reset the PHY */
+ mii_reset_phy(phy);
+
ctl = phy_read(phy, MII_BMCR);
if (ctl < 0)
return ctl;
- ctl &= ~(BMCR_FULLDPLX | BMCR_SPEED100 | BMCR_ANENABLE);
-
- /* First reset the PHY */
- phy_write(phy, MII_BMCR, ctl | BMCR_RESET);
+ ctl &= ~(BMCR_FULLDPLX | BMCR_SPEED100 | BMCR_ANENABLE |
BMCR_SPEED1000);

/* Select speed & duplex */
switch (speed) {
-
```

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>