

[PATCH 04/17] Add RapidIO multi mport support.

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-03/msg04313.html>

- From: Zhang Wei <wei.zhang@xxxxxxxxxxxxxx>
- Date: Tue, 11 Mar 2008 17:07:45 +0800

The original RapidIO driver suppose there is only one mpc85xx RIO controller in system. So, some data structures are defined as mpc85xx_rio global, such as 'regs_win', 'dbell_ring', 'msg_tx_ring'. Now, I changed them to mport's private members. And you can define multi RIO OF-nodes in dts file for multi RapidIO controller in one processor, such as PCI/PCI-Ex host controllers in Freescale's silicon. And the mport operation function declaration should be changed to know which RapidIO controller is target.

Signed-off-by: Zhang Wei <wei.zhang@xxxxxxxxxxxxxx>

```

arch/powerpc/sysdev/fsl_rio.c | 395 ++++++
drivers/rapidio/rio-access.c | 10 +-
include/linux/rio.h | 18 +-
3 files changed, 238 insertions(+), 185 deletions(-)

```

```

diff --git a/arch/powerpc/sysdev/fsl_rio.c b/arch/powerpc/sysdev/fsl_rio.c
index 659a560..80acc79 100644

```

```

--- a/arch/powerpc/sysdev/fsl_rio.c
+++ b/arch/powerpc/sysdev/fsl_rio.c
@@ -1,6 +1,9 @@
/*

```

```

* Freescale MPC85xx/MPC86xx RapidIO support
*
+ * Copyright (C) 2007, 2008 Freescale Semiconductor, Inc.
+ * Zhang Wei <wei.zhang@xxxxxxxxxxxxxx>
+ *
* Copyright 2005 MontaVista Software, Inc.
* Matt Porter <mporter@xxxxxxxxxxxxxx>
*
@@ -20,6 +23,11 @@

```

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

```

+/* RapidIO definition irq, which read from OF-tree */
+#define IRQ_RIO_BELL(m) (((struct rio_priv *) (m->priv))->bellirq)
+#define IRQ_RIO_TX(m) (((struct rio_priv *) (m->priv))->txirq)
+#define IRQ_RIO_RX(m) (((struct rio_priv *) (m->priv))->rxirq)
+

```

[PATCH 04/17] Add RapidIO multi mport support.

```
#define RIO_REGS_BASE (CCSRBAR + 0xc0000)
#define RIO_ATMU_REGS_OFFSET 0x10c00
#define RIO_MSG_REGS_OFFSET 0x11000
@@ -112,20 +120,12 @@ struct rio_tx_desc {
u32 res4;
};

-static u32 regs_win;
-static struct rio_atmu_regs *atmu_regs;
-static struct rio_atmu_regs *maint_atmu_regs;
-static struct rio_atmu_regs *dbell_atmu_regs;
-static u32 dbell_win;
-static u32 maint_win;
-static struct rio_msg_regs *msg_regs;
-
-static struct rio_dbell_ring {
+struct rio_dbell_ring {
void *virt;
dma_addr_t phys;
-} dbell_ring;
+};

-static struct rio_msg_tx_ring {
+struct rio_msg_tx_ring {
void *virt;
dma_addr_t phys;
void *virt_buffer[RIO_MAX_TX_RING_SIZE];
@@ -133,16 +133,32 @@ static struct rio_msg_tx_ring {
int tx_slot;
int size;
void *dev_id;
-} msg_tx_ring;
+};

-static struct rio_msg_rx_ring {
+struct rio_msg_rx_ring {
void *virt;
dma_addr_t phys;
void *virt_buffer[RIO_MAX_RX_RING_SIZE];
int rx_slot;
int size;
void *dev_id;
-} msg_rx_ring;
+};
+
+struct rio_priv {
+ void __iomem *regs_win;
+ struct rio_atmu_regs __iomem *atmu_regs;
+ struct rio_atmu_regs __iomem *maint_atmu_regs;
+ struct rio_atmu_regs __iomem *dbell_atmu_regs;
+ void __iomem *dbell_win;
```

[PATCH 04/17] Add RapidIO multi mport support.

[PATCH 04/17] Add RapidIO multi mport support.

```
+ void __iomem *maint_win;
+ struct rio_msg_regs __iomem *msg_regs;
+ struct rio_dbell_ring dbell_ring;
+ struct rio_msg_tx_ring msg_tx_ring;
+ struct rio_msg_rx_ring msg_rx_ring;
+ int bellirq;
+ int txirq;
+ int rxirq;
+};

/**
 * fsl_rio_doorbell_send – Send a MPC85xx doorbell message
 @@ -153,12 +169,14 @@ static struct rio_msg_rx_ring {
 * Sends a MPC85xx doorbell message. Returns %0 on success or
 * %-EINVAL on failure.
 */
-static int fsl_rio_doorbell_send(int index, u16 destid, u16 data)
+static int fsl_rio_doorbell_send(struct rio_mport *mport,
+ int index, u16 destid, u16 data)
{
+ struct rio_priv *priv = mport->priv;
pr_debug("fsl_doorbell_send: index %d destid %4.4x data %4.4x\n",
index, destid, data);
- out_be32((void *)&dbell_atmu_regs->rowtar, destid << 22);
- out_be16((void *)&dbell_win, data);
+ out_be32(&priv->dbell_atmu_regs->rowtar, destid << 22);
+ out_be16(priv->dbell_win, data);

return 0;
}
@@ -173,11 +191,13 @@ static int fsl_rio_doorbell_send(int index, u16 destid, u16 data)
 * Generates a MPC85xx local configuration space read. Returns %0 on
 * success or %-EINVAL on failure.
 */
-static int fsl_local_config_read(int index, u32 offset, int len, u32 *data)
+static int fsl_local_config_read(struct rio_mport *mport,
+ int index, u32 offset, int len, u32 *data)
{
+ struct rio_priv *priv = mport->priv;
pr_debug("fsl_local_config_read: index %d offset %8.8x\n", index,
offset);
- *data = in_be32((void *)&regs_win + offset);
+ *data = in_be32(priv->regs_win + offset);

return 0;
}
@@ -192,12 +212,14 @@ static int fsl_local_config_read(int index, u32 offset, int len, u32 *data)
 * Generates a MPC85xx local configuration space write. Returns %0 on
 * success or %-EINVAL on failure.
 */
-static int fsl_local_config_write(int index, u32 offset, int len, u32 data)
```

[PATCH 04/17] Add RapidIO multi mport support.

```
+static int fsl_local_config_write(struct rio_mport *mport,
+ int index, u32 offset, int len, u32 data)
{
+ struct rio_priv *priv = mport->priv;
pr_debug
("fsl_local_config_write: index %d offset %8.8x data %8.8x\n",
index, offset, data);
- out_be32((void *) (regs_win + offset), data);
+ out_be32(priv->regs_win + offset, data);

return 0;
}
@@ -215,18 +237,19 @@ static int fsl_local_config_write(int index, u32 offset, int len, u32 data)
* success or %-EINVAL on failure.
*/
static int
-fsl_rio_config_read(int index, u16 destid, u8 hopcount, u32 offset, int len,
- u32 * val)
+fsl_rio_config_read(struct rio_mport *mport, int index, u16 destid,
+ u8 hopcount, u32 offset, int len, u32 *val)
{
+ struct rio_priv *priv = mport->priv;
u8 *data;

pr_debug
("fsl_rio_config_read: index %d destid %d hopcount %d offset %8.8x len %d\n",
index, destid, hopcount, offset, len);
- out_be32((void *)&maint_atmu_regs->rowtar,
+ out_be32(&priv->maint_atmu_regs->rowtar,
(destid << 22) | (hopcount << 12) | ((offset & ~0x3) >> 9));

- data = (u8 *) maint_win + offset;
+ data = (u8 *) priv->maint_win + offset;
switch (len) {
case 1:
*val = in_8((u8 *) data);
@@ -255,17 +278,18 @@ fsl_rio_config_read(int index, u16 destid, u8 hopcount, u32 offset, int len,
* success or %-EINVAL on failure.
*/
static int
-fsl_rio_config_write(int index, u16 destid, u8 hopcount, u32 offset,
- int len, u32 val)
+fsl_rio_config_write(struct rio_mport *mport, int index, u16 destid,
+ u8 hopcount, u32 offset, int len, u32 val)
{
+ struct rio_priv *priv = mport->priv;
u8 *data;
pr_debug
("fsl_rio_config_write: index %d destid %d hopcount %d offset %8.8x len %d val %8.8x\n",
index, destid, hopcount, offset, len, val);
- out_be32((void *)&maint_atmu_regs->rowtar,
```

[PATCH 04/17] Add RapidIO multi mport support.

```
+ out_be32(&priv->maint_atmu_regs->rowtar,
(destid << 22) | (hopcount << 12) | ((offset & ~0x3) >> 9));

- data = (u8 *) maint_win + offset;
+ data = (u8 *) priv->maint_win + offset;
switch (len) {
case 1:
out_8((u8 *) data, val);
@@ -296,9 +320,10 @@ int
rio_hw_add_outb_message(struct rio_mport *mport, struct rio_dev *rdev, int mbox,
void *buffer, size_t len)
{
+ struct rio_priv *priv = mport->priv;
u32 omr;
- struct rio_tx_desc *desc =
- (struct rio_tx_desc *)msg_tx_ring.virt + msg_tx_ring.tx_slot;
+ struct rio_tx_desc *desc = (struct rio_tx_desc *)priv->msg_tx_ring.virt
+ + priv->msg_tx_ring.tx_slot;
int ret = 0;

pr_debug
@@ -311,11 +336,11 @@ rio_hw_add_outb_message(struct rio_mport *mport, struct rio_dev *rdev, int
mbox,
}

/* Copy and clear rest of buffer */
- memcpy(msg_tx_ring.virt_buffer[msg_tx_ring.tx_slot], buffer, len);
+ memcpy(priv->msg_tx_ring.virt_buffer[priv->msg_tx_ring.tx_slot], buffer,
+ len);
if (len < (RIO_MAX_MSG_SIZE - 4))
- memset((void *)((u32) msg_tx_ring.
- virt_buffer[msg_tx_ring.tx_slot] + len), 0,
- RIO_MAX_MSG_SIZE - len);
+ memset(priv->msg_tx_ring.virt_buffer[priv->msg_tx_ring.tx_slot]
+ + len, 0, RIO_MAX_MSG_SIZE - len);

/* Set mbox field for message */
desc->dport = mbox & 0x3;
@@ -327,15 +352,16 @@ rio_hw_add_outb_message(struct rio_mport *mport, struct rio_dev *rdev, int
mbox,
desc->dwcnt = is_power_of_2(len) ? len : 1 << get_bitmask_order(len);

/* Set snooping and source buffer address */
- desc->saddr = 0x00000004 | msg_tx_ring.phys_buffer[msg_tx_ring.tx_slot];
+ desc->saddr = 0x00000004
+ | priv->msg_tx_ring.phys_buffer[priv->msg_tx_ring.tx_slot];

/* Increment enqueue pointer */
- omr = in_be32((void *)&msg_regs->omr);
- out_be32((void *)&msg_regs->omr, omr | RIO_MSG_OMR_MUI);
+ omr = in_be32(&priv->msg_regs->omr);
```

[PATCH 04/17] Add RapidIO multi mport support.

[PATCH 04/17] Add RapidIO multi mport support.

```
+ out_be32(&priv->msg_regs->omr, omr | RIO_MSG_OMR_MUI);

/* Go to next descriptor */
- if (++msg_tx_ring.tx_slot == msg_tx_ring.size)
- msg_tx_ring.tx_slot = 0;
+ if (++priv->msg_tx_ring.tx_slot == priv->msg_tx_ring.size)
+ priv->msg_tx_ring.tx_slot = 0;

out:
return ret;
@@ -356,28 +382,30 @@ fsl_rio_tx_handler(int irq, void *dev_instance)
{
int osr;
struct rio_mport *port = (struct rio_mport *)dev_instance;
+ struct rio_priv *priv = port->priv;

- osr = in_be32((void *)&msg_regs->osr);
+ osr = in_be32(&priv->msg_regs->osr);

if (osr & RIO_MSG_OSR_TE) {
pr_info("RIO: outbound message transmission error\n");
- out_be32((void *)&msg_regs->osr, RIO_MSG_OSR_TE);
+ out_be32(&priv->msg_regs->osr, RIO_MSG_OSR_TE);
goto out;
}

if (osr & RIO_MSG_OSR_QOI) {
pr_info("RIO: outbound message queue overflow\n");
- out_be32((void *)&msg_regs->osr, RIO_MSG_OSR_QOI);
+ out_be32(&priv->msg_regs->osr, RIO_MSG_OSR_QOI);
goto out;
}

if (osr & RIO_MSG_OSR_EOMI) {
- u32 dqp = in_be32((void *)&msg_regs->odqdp);
- int slot = (dqp - msg_tx_ring.phys) >> 5;
- port->outb_msg[0].mcbk(port, msg_tx_ring.dev_id, -1, slot);
+ u32 dqp = in_be32(&priv->msg_regs->odqdp);
+ int slot = (dqp - priv->msg_tx_ring.phys) >> 5;
+ port->outb_msg[0].mcbk(port, priv->msg_tx_ring.dev_id, -1,
+ slot);

/* Ack the end-of-message interrupt */
- out_be32((void *)&msg_regs->osr, RIO_MSG_OSR_EOMI);
+ out_be32(&priv->msg_regs->osr, RIO_MSG_OSR_EOMI);
}

out:
@@ -398,6 +426,7 @@ fsl_rio_tx_handler(int irq, void *dev_instance)
int rio_open_outb_mbox(struct rio_mport *mport, void *dev_id, int mbox, int entries)
{
```

[PATCH 04/17] Add RapidIO multi mport support.

```
int i, j, rc = 0;
+ struct rio_priv *priv = mport->priv;

if ((entries < RIO_MIN_TX_RING_SIZE) ||
(entries > RIO_MAX_TX_RING_SIZE) || (!is_power_of_2(entries))) {
@@ -406,54 +435,53 @@ int rio_open_outb_mbox(struct rio_mport *mport, void *dev_id, int mbox, int
entr
}

/* Initialize shadow copy ring */
- msg_tx_ring.dev_id = dev_id;
- msg_tx_ring.size = entries;
-
- for (i = 0; i < msg_tx_ring.size; i++) {
- if (!
- (msg_tx_ring.virt_buffer[i] =
- dma_alloc_coherent(NULL, RIO_MSG_BUFFER_SIZE,
- &msg_tx_ring.phys_buffer[i],
- GFP_KERNEL))) {
+ priv->msg_tx_ring.dev_id = dev_id;
+ priv->msg_tx_ring.size = entries;
+
+ for (i = 0; i < priv->msg_tx_ring.size; i++) {
+ priv->msg_tx_ring.virt_buffer[i] =
+ dma_alloc_coherent(NULL, RIO_MSG_BUFFER_SIZE,
+ &priv->msg_tx_ring.phys_buffer[i], GFP_KERNEL);
+ if (!priv->msg_tx_ring.virt_buffer[i]) {
rc = -ENOMEM;
- for (j = 0; j < msg_tx_ring.size; j++)
- if (msg_tx_ring.virt_buffer[j])
+ for (j = 0; j < priv->msg_tx_ring.size; j++)
+ if (priv->msg_tx_ring.virt_buffer[j])
dma_free_coherent(NULL,
- RIO_MSG_BUFFER_SIZE,
- msg_tx_ring.
- virt_buffer[j],
- msg_tx_ring.
- phys_buffer[j]);
+ RIO_MSG_BUFFER_SIZE,
+ priv->msg_tx_ring.
+ virt_buffer[j],
+ priv->msg_tx_ring.
+ phys_buffer[j]);
goto out;
}
}

/* Initialize outbound message descriptor ring */
- if (!(msg_tx_ring.virt = dma_alloc_coherent(NULL,
- msg_tx_ring.size *
- RIO_MSG_DESC_SIZE,
```

[PATCH 04/17] Add RapidIO multi mport support.

```
- &msg_tx_ring.phys,
- GFP_KERNEL))) {
+ priv->msg_tx_ring.virt = dma_alloc_coherent(NULL,
+ priv->msg_tx_ring.size * RIO_MSG_DESC_SIZE,
+ &priv->msg_tx_ring.phys, GFP_KERNEL);
+ if (!priv->msg_tx_ring.virt) {
rc = -ENOMEM;
goto out_dma;
}
- memset(msg_tx_ring.virt, 0, msg_tx_ring.size * RIO_MSG_DESC_SIZE);
- msg_tx_ring.tx_slot = 0;
+ memset(priv->msg_tx_ring.virt, 0,
+ priv->msg_tx_ring.size * RIO_MSG_DESC_SIZE);
+ priv->msg_tx_ring.tx_slot = 0;

/* Point dequeue/enqueue pointers at first entry in ring */
- out_be32((void *)&msg_regs->odqpar, msg_tx_ring.phys);
- out_be32((void *)&msg_regs->odqepar, msg_tx_ring.phys);
+ out_be32(&priv->msg_regs->odqpar, priv->msg_tx_ring.phys);
+ out_be32(&priv->msg_regs->odqepar, priv->msg_tx_ring.phys);

/* Configure for snooping */
- out_be32((void *)&msg_regs->osar, 0x00000004);
+ out_be32(&priv->msg_regs->osar, 0x00000004);

/* Clear interrupt status */
- out_be32((void *)&msg_regs->osr, 0x000000b3);
+ out_be32(&priv->msg_regs->osr, 0x000000b3);

/* Hook up outbound message handler */
- if ((rc =
- request_irq(MPC85xx_IRQ_RIO_TX, fsl_rio_tx_handler, 0,
- "msg_tx", (void *)mport)) < 0)
+ rc = request_irq(IRQ_RIO_TX(mport), fsl_rio_tx_handler, 0,
+ "msg_tx", (void *)mport);
+ if (rc < 0)
goto out_irq;

/*
@@ -463,28 +491,28 @@ int rio_open_outb_mbox(struct rio_mport *mport, void *dev_id, int mbox, int
entr
* Chaining mode
* Disable
*/
- out_be32((void *)&msg_regs->omr, 0x00100220);
+ out_be32(&priv->msg_regs->omr, 0x00100220);

/* Set number of entries */
- out_be32((void *)&msg_regs->omr,
- in_be32((void *)&msg_regs->omr) |
+ out_be32(&priv->msg_regs->omr,
```

[PATCH 04/17] Add RapidIO multi mport support.

```
+ in_be32(&priv->msg_regs->omr) |
((get_bitmask_order(entries) - 2) << 12));

/* Now enable the unit */
- out_be32((void *)&msg_regs->omr, in_be32((void *)&msg_regs->omr) | 0x1);
+ out_be32(&priv->msg_regs->omr, in_be32(&priv->msg_regs->omr) | 0x1);

out:
return rc;

out_irq:
- dma_free_coherent(NULL, msg_tx_ring.size * RIO_MSG_DESC_SIZE,
- msg_tx_ring.virt, msg_tx_ring.phys);
+ dma_free_coherent(NULL, priv->msg_tx_ring.size * RIO_MSG_DESC_SIZE,
+ priv->msg_tx_ring.virt, priv->msg_tx_ring.phys);

out_dma:
- for (i = 0; i < msg_tx_ring.size; i++)
+ for (i = 0; i < priv->msg_tx_ring.size; i++)
dma_free_coherent(NULL, RIO_MSG_BUFFER_SIZE,
- msg_tx_ring.virt_buffer[i],
- msg_tx_ring.phys_buffer[i]);
+ priv->msg_tx_ring.virt_buffer[i],
+ priv->msg_tx_ring.phys_buffer[i]);

return rc;
}
@@ -499,15 +527,16 @@ int rio_open_outb_mbox(struct rio_mport *mport, void *dev_id, int mbox, int
entr
*/
void rio_close_outb_mbox(struct rio_mport *mport, int mbox)
{
+ struct rio_priv *priv = mport->priv;
/* Disable inbound message unit */
- out_be32((void *)&msg_regs->omr, 0);
+ out_be32(&priv->msg_regs->omr, 0);

/* Free ring */
- dma_free_coherent(NULL, msg_tx_ring.size * RIO_MSG_DESC_SIZE,
- msg_tx_ring.virt, msg_tx_ring.phys);
+ dma_free_coherent(NULL, priv->msg_tx_ring.size * RIO_MSG_DESC_SIZE,
+ priv->msg_tx_ring.virt, priv->msg_tx_ring.phys);

/* Free interrupt */
- free_irq(MPC85xx_IRQ_RIO_TX, (void *)mport);
+ free_irq(IRQ_RIO_TX(mport), (void *)mport);
}

/**
@@ -523,12 +552,13 @@ fsl_rio_rx_handler(int irq, void *dev_instance)
{
```

[PATCH 04/17] Add RapidIO multi mport support.

```
int isr;
struct rio_mport *port = (struct rio_mport *)dev_instance;
+ struct rio_priv *priv = port->priv;

- isr = in_be32((void *)&msg_regs->isr);
+ isr = in_be32(&priv->msg_regs->isr);

if (isr & RIO_MSG_ISR_TE) {
pr_info("RIO: inbound message reception error\n");
- out_be32((void *)&msg_regs->isr, RIO_MSG_ISR_TE);
+ out_be32((void *)&priv->msg_regs->isr, RIO_MSG_ISR_TE);
goto out;
}

@@ -540,10 +570,10 @@ fsl_rio_rx_handler(int irq, void *dev_instance)
* make the callback with an unknown/invalid mailbox number
* argument.
*/
- port->inb_msg[0].mcbkback(port, msg_rx_ring.dev_id, -1, -1);
+ port->inb_msg[0].mcbkback(port, priv->msg_rx_ring.dev_id, -1, -1);

/* Ack the queueing interrupt */
- out_be32((void *)&msg_regs->isr, RIO_MSG_ISR_DIQI);
+ out_be32(&priv->msg_regs->isr, RIO_MSG_ISR_DIQI);
}

out:
@@ -564,6 +594,7 @@ fsl_rio_rx_handler(int irq, void *dev_instance)
int rio_open_inb_mbox(struct rio_mport *mport, void *dev_id, int mbox, int entries)
{
int i, rc = 0;
+ struct rio_priv *priv = mport->priv;

if ((entries < RIO_MIN_RX_RING_SIZE) ||
(entries > RIO_MAX_RX_RING_SIZE) || (!is_power_of_2(entries))) {
@@ -572,36 +603,35 @@ int rio_open_inb_mbox(struct rio_mport *mport, void *dev_id, int mbox, int entri
}

/* Initialize client buffer ring */
- msg_rx_ring.dev_id = dev_id;
- msg_rx_ring.size = entries;
- msg_rx_ring.rx_slot = 0;
- for (i = 0; i < msg_rx_ring.size; i++)
- msg_rx_ring.virt_buffer[i] = NULL;
+ priv->msg_rx_ring.dev_id = dev_id;
+ priv->msg_rx_ring.size = entries;
+ priv->msg_rx_ring.rx_slot = 0;
+ for (i = 0; i < priv->msg_rx_ring.size; i++)
+ priv->msg_rx_ring.virt_buffer[i] = NULL;

/* Initialize inbound message ring */
```

[PATCH 04/17] Add RapidIO multi mport support.

```
- if (!msg_rx_ring.virt = dma_alloc_coherent(NULL,
- msg_rx_ring.size *
- RIO_MAX_MSG_SIZE,
- &msg_rx_ring.phys,
- GFP_KERNEL))) {
+ priv->msg_rx_ring.virt = dma_alloc_coherent(NULL,
+ priv->msg_rx_ring.size * RIO_MAX_MSG_SIZE,
+ &priv->msg_rx_ring.phys, GFP_KERNEL);
+ if (!priv->msg_rx_ring.virt) {
rc = -ENOMEM;
goto out;
}

/* Point dequeue/enqueue pointers at first entry in ring */
- out_be32((void *)&msg_regs->ifqdparr, (u32) msg_rx_ring.phys);
- out_be32((void *)&msg_regs->ifqparr, (u32) msg_rx_ring.phys);
+ out_be32(&priv->msg_regs->ifqdparr, (u32) priv->msg_rx_ring.phys);
+ out_be32(&priv->msg_regs->ifqparr, (u32) priv->msg_rx_ring.phys);

/* Clear interrupt status */
- out_be32((void *)&msg_regs->isr, 0x00000091);
+ out_be32(&priv->msg_regs->isr, 0x00000091);

/* Hook up inbound message handler */
- if ((rc =
- request_irq(MPC85xx_IRQ_RIO_RX, fsl_rio_rx_handler, 0,
- "msg_rx", (void *)mport) < 0) {
+ rc = request_irq(IRQ_RIO_RX(mport), fsl_rio_rx_handler, 0,
+ "msg_rx", (void *)mport);
+ if (rc < 0) {
dma_free_coherent(NULL, RIO_MSG_BUFFER_SIZE,
- msg_tx_ring.virt_buffer[i],
- msg_tx_ring.phys_buffer[i]);
+ priv->msg_tx_ring.virt_buffer[i],
+ priv->msg_tx_ring.phys_buffer[i]);
goto out;
}

@@ -612,15 +642,13 @@ int rio_open_inb_mbox(struct rio_mport *mport, void *dev_id, int mbox, int entries)
* Unmask all interrupt sources
* Disable
*/
- out_be32((void *)&msg_regs->imr, 0x001b0060);
+ out_be32(&priv->msg_regs->imr, 0x001b0060);

/* Set number of queue entries */
- out_be32((void *)&msg_regs->imr,
- in_be32((void *)&msg_regs->imr) |
- ((get_bitmask_order(entries) - 2) << 12));
+ setbits32(&priv->msg_regs->imr, (get_bitmask_order(entries) - 2) << 12);
```

[PATCH 04/17] Add RapidIO multi mport support.

```
/* Now enable the unit */
- out_be32((void *)&msg_regs->imr, in_be32((void *)&msg_regs->imr) | 0x1);
+ setbits32(&priv->msg_regs->imr, 0x1);

out:
return rc;
@@ -636,15 +664,16 @@ int rio_open_inb_mbox(struct rio_mport *mport, void *dev_id, int mbox, int entries)
*/
void rio_close_inb_mbox(struct rio_mport *mport, int mbox)
{
+ struct rio_priv *priv = mport->priv;
/* Disable inbound message unit */
- out_be32((void *)&msg_regs->imr, 0);
+ out_be32(&priv->msg_regs->imr, 0);

/* Free ring */
- dma_free_coherent(NULL, msg_rx_ring.size * RIO_MAX_MSG_SIZE,
- msg_rx_ring.virt, msg_rx_ring.phys);
+ dma_free_coherent(NULL, priv->msg_rx_ring.size * RIO_MAX_MSG_SIZE,
+ priv->msg_rx_ring.virt, priv->msg_rx_ring.phys);

/* Free interrupt */
- free_irq(MPC85xx_IRQ_RIO_RX, (void *)mport);
+ free_irq(IRQ_RIO_RX(mport), (void *)mport);
}

/**
@@ -659,21 +688,22 @@ void rio_close_inb_mbox(struct rio_mport *mport, int mbox)
int rio_hw_add_inb_buffer(struct rio_mport *mport, int mbox, void *buf)
{
int rc = 0;
+ struct rio_priv *priv = mport->priv;

pr_debug("RIO: rio_hw_add_inb_buffer(), msg_rx_ring.rx_slot %d\n",
- msg_rx_ring.rx_slot);
+ priv->msg_rx_ring.rx_slot);

- if (msg_rx_ring.virt_buffer[msg_rx_ring.rx_slot]) {
+ if (priv->msg_rx_ring.virt_buffer[priv->msg_rx_ring.rx_slot]) {
printk(KERN_ERR
"RIO: error adding inbound buffer %d, buffer exists\n",
- msg_rx_ring.rx_slot);
+ priv->msg_rx_ring.rx_slot);
rc = -EINVAL;
goto out;
}

- msg_rx_ring.virt_buffer[msg_rx_ring.rx_slot] = buf;
- if (++msg_rx_ring.rx_slot == msg_rx_ring.size)
- msg_rx_ring.rx_slot = 0;
+ priv->msg_rx_ring.virt_buffer[priv->msg_rx_ring.rx_slot] = buf;
```

[PATCH 04/17] Add RapidIO multi mport support.

```
+ if (++priv->msg_rx_ring.rx_slot == priv->msg_rx_ring.size)
+ priv->msg_rx_ring.rx_slot = 0;

out:
return rc;
@@ -691,20 +721,21 @@ EXPORT_SYMBOL_GPL(rio_hw_add_inb_buffer);
*/
void *rio_hw_get_inb_message(struct rio_mport *mport, int mbox)
{
- u32 imr;
+ struct rio_priv *priv = mport->priv;
u32 phys_buf, virt_buf;
void *buf = NULL;
int buf_idx;

- phys_buf = in_be32((void *)&msg_regs->ifqdparr);
+ phys_buf = in_be32(&priv->msg_regs->ifqdparr);

/* If no more messages, then bail out */
- if (phys_buf == in_be32((void *)&msg_regs->ifqeparr))
+ if (phys_buf == in_be32(&priv->msg_regs->ifqeparr))
goto out2;

- virt_buf = (u32) msg_rx_ring.virt + (phys_buf - msg_rx_ring.phys);
- buf_idx = (phys_buf - msg_rx_ring.phys) / RIO_MAX_MSG_SIZE;
- buf = msg_rx_ring.virt_buffer[buf_idx];
+ virt_buf = (u32) priv->msg_rx_ring.virt + (phys_buf
+ - priv->msg_rx_ring.phys);
+ buf_idx = (phys_buf - priv->msg_rx_ring.phys) / RIO_MAX_MSG_SIZE;
+ buf = priv->msg_rx_ring.virt_buffer[buf_idx];

if (!buf) {
printk(KERN_ERR
@@ -716,11 +747,10 @@ void *rio_hw_get_inb_message(struct rio_mport *mport, int mbox)
memcpy(buf, (void *)virt_buf, RIO_MAX_MSG_SIZE);

/* Clear the available buffer */
- msg_rx_ring.virt_buffer[buf_idx] = NULL;
+ priv->msg_rx_ring.virt_buffer[buf_idx] = NULL;

out1:
- imr = in_be32((void *)&msg_regs->imr);
- out_be32((void *)&msg_regs->imr, imr | RIO_MSG_IMR_MI);
+ setbits32(&priv->msg_regs->imr, RIO_MSG_IMR_MI);

out2:
return buf;
@@ -741,27 +771,27 @@ fsl_rio_dbell_handler(int irq, void *dev_instance)
{
int dsr;
struct rio_mport *port = (struct rio_mport *)dev_instance;
```

[PATCH 04/17] Add RapidIO multi mport support.

```
+ struct rio_priv *priv = port->priv;

- dsr = in_be32((void *)&msg_regs->dsr);
+ dsr = in_be32(&priv->msg_regs->dsr);

if (dsr & DOORBELL_DSR_TE) {
pr_info("RIO: doorbell reception error\n");
- out_be32((void *)&msg_regs->dsr, DOORBELL_DSR_TE);
+ out_be32(&priv->msg_regs->dsr, DOORBELL_DSR_TE);
goto out;
}

if (dsr & DOORBELL_DSR_QFI) {
pr_info("RIO: doorbell queue full\n");
- out_be32((void *)&msg_regs->dsr, DOORBELL_DSR_QFI);
+ out_be32(&priv->msg_regs->dsr, DOORBELL_DSR_QFI);
goto out;
}

/* XXX Need to check/dispatch until queue empty */
if (dsr & DOORBELL_DSR_DIQI) {
u32 dmsg =
- (u32) dbell_ring.virt +
- (in_be32((void *)&msg_regs->dqdpar) & 0xfff);
- u32 dmr;
+ (u32) priv->dbell_ring.virt +
+ (in_be32(&priv->msg_regs->dqdpar) & 0xfff);
struct rio_dbell *dbell;
int found = 0;

@@ -784,9 +814,8 @@ fsl_rio_dbell_handler(int irq, void *dev_instance)
("RIO: spurious doorbell, sid %2.2x tid %2.2x info %4.4x\n",
DBELL_SID(dmsg), DBELL_TID(dmsg), DBELL_INF(dmsg));
}
- dmr = in_be32((void *)&msg_regs->dmr);
- out_be32((void *)&msg_regs->dmr, dmr | DOORBELL_DMR_DI);
- out_be32((void *)&msg_regs->dsr, DOORBELL_DSR_DIQI);
+ setbits32(&priv->msg_regs->dmr, DOORBELL_DMR_DI);
+ out_be32(&priv->msg_regs->dsr, DOORBELL_DSR_DIQI);
}

out:
@@ -803,12 +832,13 @@ fsl_rio_dbell_handler(int irq, void *dev_instance)
*/
static int fsl_rio_doorbell_init(struct rio_mport *mport)
{
+ struct rio_priv *priv = mport->priv;
int rc = 0;

/* Map outbound doorbell window immediately after maintenance window */
- if (!(dbell_win =
```

[PATCH 04/17] Add RapidIO multi mport support.

```
- (u32) ioremap(mport->iores.start + RIO_MAINT_WIN_SIZE,
- RIO_DBELL_WIN_SIZE))) {
+ priv->dbell_win = ioremap(mport->iores.start + RIO_MAINT_WIN_SIZE,
+ RIO_DBELL_WIN_SIZE);
+ if (!priv->dbell_win) {
printk(KERN_ERR
"RIO: unable to map outbound doorbell window\n");
rc = -ENOMEM;
@@ -816,37 +846,36 @@ static int fsl_rio_doorbell_init(struct rio_mport *mport)
}

/* Initialize inbound doorbells */
- if (!(dbell_ring.virt = dma_alloc_coherent(NULL,
- 512 * DOORBELL_MESSAGE_SIZE,
- &dbell_ring.phys,
- GFP_KERNEL))) {
+ priv->dbell_ring.virt = dma_alloc_coherent(NULL, 512 *
+ DOORBELL_MESSAGE_SIZE, &priv->dbell_ring.phys, GFP_KERNEL);
+ if (!priv->dbell_ring.virt) {
printk(KERN_ERR "RIO: unable allocate inbound doorbell ring\n");
rc = -ENOMEM;
- iounmap((void *)dbell_win);
+ iounmap(priv->dbell_win);
goto out;
}

/* Point dequeue/enqueue pointers at first entry in ring */
- out_be32((void *)&msg_regs->dqdpar, (u32) dbell_ring.phys);
- out_be32((void *)&msg_regs->dqepar, (u32) dbell_ring.phys);
+ out_be32(&priv->msg_regs->dqdpar, (u32) priv->dbell_ring.phys);
+ out_be32(&priv->msg_regs->dqepar, (u32) priv->dbell_ring.phys);

/* Clear interrupt status */
- out_be32((void *)&msg_regs->dsr, 0x00000091);
+ out_be32(&priv->msg_regs->dsr, 0x00000091);

/* Hook up doorbell handler */
- if ((rc =
- request_irq(MPC85xx_IRQ_RIO_BELL, fsl_rio_dbell_handler, 0,
- "dbell_rx", (void *)mport) < 0)) {
- iounmap((void *)dbell_win);
+ rc = request_irq(IRQ_RIO_BELL(mport), fsl_rio_dbell_handler, 0,
+ "dbell_rx", (void *)mport);
+ if (rc < 0) {
+ iounmap(priv->dbell_win);
dma_free_coherent(NULL, 512 * DOORBELL_MESSAGE_SIZE,
- dbell_ring.virt, dbell_ring.phys);
+ priv->dbell_ring.virt, priv->dbell_ring.phys);
printk(KERN_ERR
"MPC85xx RIO: unable to request inbound doorbell irq");
goto out;
```

[PATCH 04/17] Add RapidIO multi mport support.

```
}

/* Configure doorbells for snooping, 512 entries, and enable */
- out_be32((void *)&msg_regs->dmr, 0x00108161);
+ out_be32(&priv->msg_regs->dmr, 0x00108161);

out:
return rc;
@@ -887,6 +916,8 @@ void fsl_rio_setup(int law_start, int law_size)
{
struct rio_ops *ops;
struct rio_mport *port;
+ struct rio_priv *priv = NULL;
+ int rc;

ops = kmalloc(sizeof(struct rio_ops), GFP_KERNEL);
ops->lcread = fsl_local_config_read;
@@ -895,9 +926,17 @@ void fsl_rio_setup(int law_start, int law_size)
ops->cwrite = fsl_rio_config_write;
ops->dsend = fsl_rio_doorbell_send;

- port = kmalloc(sizeof(struct rio_mport), GFP_KERNEL);
+ port = kzalloc(sizeof(struct rio_mport), GFP_KERNEL);
port->id = 0;
port->index = 0;
+
+ priv = kzalloc(sizeof(struct rio_priv), GFP_KERNEL);
+ if (!priv) {
+ printk(KERN_ERR "Can't alloc memory for 'priv'\n");
+ rc = -ENOMEM;
+ goto err;
+ }
+
INIT_LIST_HEAD(&port->dbells);
port->iores.start = law_start;
port->iores.end = law_start + law_size;
@@ -911,22 +950,32 @@ void fsl_rio_setup(int law_start, int law_size)
port->ops = ops;
port->host_deviceid = fsl_rio_get_hdid(port->id);

+ port->priv = priv;
rio_register_mport(port);

- regs_win = (u32) ioremap(RIO_REGS_BASE, 0x20000);
- atmu_regs = (struct rio_atmu_regs *) (regs_win + RIO_ATMU_REGS_OFFSET);
- maint_atmu_regs = atmu_regs + 1;
- dbell_atmu_regs = atmu_regs + 2;
- msg_regs = (struct rio_msg_regs *) (regs_win + RIO_MSG_REGS_OFFSET);
+ priv->regs_win = ioremap(RIO_REGS_BASE, 0x20000);
+ priv->atmu_regs = (struct rio_atmu_regs *) (priv->regs_win
+ + RIO_ATMU_REGS_OFFSET);
```

[PATCH 04/17] Add RapidIO multi mport support.

```
+ priv->maint_atmu_regs = priv->atmu_regs + 1;
+ priv->dbell_atmu_regs = priv->atmu_regs + 2;
+ priv->msg_regs = (struct rio_msg_regs*)(priv->regs_win
+ + RIO_MSG_REGS_OFFSET);

/* Configure maintenance transaction window */
- out_be32((void*)&maint_atmu_regs->rowbar, 0x000c0000);
- out_be32((void*)&maint_atmu_regs->rowar, 0x80077015);
+ out_be32(&priv->maint_atmu_regs->rowbar, 0x000c0000);
+ out_be32(&priv->maint_atmu_regs->rowar, 0x80077015);

- maint_win = (u32) ioremap(law_start, RIO_MAINT_WIN_SIZE);
+ priv->maint_win = ioremap(law_start, RIO_MAINT_WIN_SIZE);

/* Configure outbound doorbell window */
- out_be32((void*)&dbell_atmu_regs->rowbar, 0x000c0400);
- out_be32((void*)&dbell_atmu_regs->rowar, 0x8004200b);
+ out_be32(&priv->dbell_atmu_regs->rowbar, 0x000c0400);
+ out_be32(&priv->dbell_atmu_regs->rowar, 0x8004200b);
fsl_rio_doorbell_init(port);
+
+ return;
+err:
+ if (priv)
+ iounmap(priv->regs_win);
+ kfree(priv);
+ kfree(port);
}
diff --git a/drivers/rapidio/rio-access.c b/drivers/rapidio/rio-access.c
index 8b56bbd..a3824ba 100644
--- a/drivers/rapidio/rio-access.c
+++ b/drivers/rapidio/rio-access.c
@@ -48,7 +48,7 @@ int __rio_local_read_config_##size \
u32 data = 0; \
if (RIO_###size##_BAD) return RIO_BAD_SIZE; \
spin_lock_irqsave(&rio_config_lock, flags); \
- res = mport->ops->lcread(mport->id, offset, len, &data); \
+ res = mport->ops->lcread(mport, mport->id, offset, len, &data); \
*value = (type)data; \
spin_unlock_irqrestore(&rio_config_lock, flags); \
return res; \
@@ -71,7 +71,7 @@ int __rio_local_write_config_##size \
unsigned long flags; \
if (RIO_###size##_BAD) return RIO_BAD_SIZE; \
spin_lock_irqsave(&rio_config_lock, flags); \
- res = mport->ops->lcwrite(mport->id, offset, len, value); \
+ res = mport->ops->lcwrite(mport, mport->id, offset, len, value); \
spin_unlock_irqrestore(&rio_config_lock, flags); \
return res; \
}
@@ -108,7 +108,7 @@ int rio_mport_read_config_##size \
```

[PATCH 04/17] Add RapidIO multi mport support.

```
u32 data = 0; \
if (RIO_###size##_BAD) return RIO_BAD_SIZE; \
spin_lock_irqsave(&rio_config_lock, flags); \
- res = mport->ops->cread(mport->id, destid, hopcount, offset, len, &data); \
+ res = mport->ops->cread(mport, mport->id, destid, hopcount, offset, len, &data); \
*value = (type)data; \
spin_unlock_irqrestore(&rio_config_lock, flags); \
return res; \
@@ -131,7 +131,7 @@ int rio_mport_write_config_###size \
unsigned long flags; \
if (RIO_###size##_BAD) return RIO_BAD_SIZE; \
spin_lock_irqsave(&rio_config_lock, flags); \
- res = mport->ops->cwrite(mport->id, destid, hopcount, offset, len, value); \
+ res = mport->ops->cwrite(mport, mport->id, destid, hopcount, offset, len, value); \
spin_unlock_irqrestore(&rio_config_lock, flags); \
return res; \
}
@@ -166,7 +166,7 @@ int rio_mport_send_doorbell(struct rio_mport *mport, u16 destid, u16 data)
unsigned long flags;

spin_lock_irqsave(&rio_doorbell_lock, flags);
- res = mport->ops->dsend(mport->id, destid, data);
+ res = mport->ops->dsend(mport, mport->id, destid, data);
spin_unlock_irqrestore(&rio_doorbell_lock, flags);

return res;
diff --git a/include/linux/rio.h b/include/linux/rio.h
index 68e3f68..258c453 100644
--- a/include/linux/rio.h
+++ b/include/linux/rio.h
@@ -163,6 +163,7 @@ struct rio_dbell {
* @id: Port ID, unique among all ports
* @index: Port index, unique among all port interfaces of the same type
* @name: Port name string
+ * @priv: Master port private data
*/

struct rio_mport {
struct list_head dbells; /* list of doorbell events */
@@ -178,6 +179,7 @@ struct rio_mport {
unsigned char index; /* port index, unique among all port
interfaces of the same type */
unsigned char name[40];
+ void *priv; /* Master port private data */
};

/**
@@ -229,13 +231,15 @@ struct rio_switch {
* @dsend: Callback to send a doorbell message.
*/

struct rio_ops {
- int (*lread) (int index, u32 offset, int len, u32 * data);
```

[PATCH 04/17] Add RapidIO multi mport support.

```
- int (*lwrite) (int index, u32 offset, int len, u32 data);
- int (*cread) (int index, u16 destid, u8 hopcount, u32 offset, int len,
- u32 * data);
- int (*cwrite) (int index, u16 destid, u8 hopcount, u32 offset, int len,
- u32 data);
- int (*dsend) (int index, u16 destid, u16 data);
+ int (*lcread) (struct rio_mport *mport, int index, u32 offset, int len,
+ u32 *data);
+ int (*lcwrite) (struct rio_mport *mport, int index, u32 offset, int len,
+ u32 data);
+ int (*cread) (struct rio_mport *mport, int index, u16 destid,
+ u8 hopcount, u32 offset, int len, u32 *data);
+ int (*cwrite) (struct rio_mport *mport, int index, u16 destid,
+ u8 hopcount, u32 offset, int len, u32 data);
+ int (*dsend) (struct rio_mport *mport, int index, u16 destid, u16 data);
};
```

```
#define RIO_RESOURCE_MEM 0x00000100
```

```
--
```

1.5.4

```
--
```

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/>