

[PATCH 2 of 18] ipath – core driver header files

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-03/msg08350.html>

- *From:* Bryan O'Sullivan <bos@xxxxxxxxxxxxxx>
 - *Date:* Thu, 23 Mar 2006 20:41:34 -0800
-

ipath_common.h and ips_common.h contain definitions shared between userspace and the kernel.

ipath_kernel.h is the core driver header file.

ipath_debug.h contains mask values used for controlling driver debugging.

ipath_registers.h contains bitmask definitions used in chip registers.

Signed-off-by: Bryan O'Sullivan <bos@xxxxxxxxxxxxxx>

```
diff -r 747da2c0243d -r 4b2debbae33 drivers/infiniband/hw/ipath/ipath_common.h
--- /dev/null Thu Jan 1 00:00:00 1970 +0000
+++ b/drivers/infiniband/hw/ipath/ipath_common.h Thu Mar 23 20:27:44 2006 -0800
@@ -0,0 +1,616 @@
+/*
+ * Copyright (c) 2003, 2004, 2005, 2006 PathScale, Inc. All rights reserved.
+ *
+ * This software is available to you under a choice of one of two
+ * licenses. You may choose to be licensed under the terms of the GNU
+ * General Public License (GPL) Version 2, available from the file
+ * COPYING in the main directory of this source tree, or the
+ * OpenIB.org BSD license below:
+ *
+ * Redistribution and use in source and binary forms, with or
+ * without modification, are permitted provided that the following
+ * conditions are met:
+ *
+ * - Redistributions of source code must retain the above
+ * copyright notice, this list of conditions and the following
+ * disclaimer.
+ *
+ * - Redistributions in binary form must reproduce the above
+ * copyright notice, this list of conditions and the following
+ * disclaimer in the documentation and/or other materials
+ * provided with the distribution.
+ *
+ * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
+ * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
```


[PATCH 2 of 18] ipath – core driver header files

```
+ #define IPATH_IB_LINKINIT 3
+ #define IPATH_IB_LINKDOWN_SLEEP 4
+ #define IPATH_IB_LINKDOWN_DISABLE 5
+
+ /*
+  * stats maintained by the driver. For now, at least, this is global
+  * to all minor devices.
+  */
+ struct infinipath_stats {
+  /* number of interrupts taken */
+  __u64 sps_ints;
+  /* number of interrupts for errors */
+  __u64 sps_errints;
+  /* number of errors from chip (not incl. packet errors or CRC) */
+  __u64 sps_errs;
+  /* number of packet errors from chip other than CRC */
+  __u64 sps_pkterrs;
+  /* number of packets with CRC errors (ICRC and VCRC) */
+  __u64 sps_crcerrs;
+  /* number of hardware errors reported (parity, etc.) */
+  __u64 sps_hwerrs;
+  /* number of times IB link changed state unexpectedly */
+  __u64 sps_iblink;
+  /* no longer used; left for compatibility */
+  __u64 sps_unused3;
+  /* number of kernel (port0) packets received */
+  __u64 sps_port0pkts;
+  /* number of "ethernet" packets sent by driver */
+  __u64 sps_ether_spkts;
+  /* number of "ethernet" packets received by driver */
+  __u64 sps_ether_rpkts;
+  /* number of SMA packets sent by driver */
+  __u64 sps_sma_spkts;
+  /* number of SMA packets received by driver */
+  __u64 sps_sma_rpkts;
+  /* number of times all ports rcvhdrq was full and packet dropped */
+  __u64 sps_hdrqfull;
+  /* number of times all ports egrtid was full and packet dropped */
+  __u64 sps_etidfull;
+  /*
+  * number of times we tried to send from driver, but no pio buffers
+  * avail
+  */
+  __u64 sps_nopiobufs;
+  /* number of ports currently open */
+  __u64 sps_ports;
+  /* list of pkeys (other than default) accepted (0 means not set) */
+  __u16 sps_pkeys[4];
+  /* lids for up to 4 infinipaths, indexed by infinipath # */
+  __u16 sps_lid[4];
+  /* number of user ports per chip (not IB ports) */
```

[PATCH 2 of 18] ipath – core driver header files

```
+ __u32 sps_nports;
+ /* not our interrupt, or already handled */
+ __u32 sps_nullintr;
+ /* max number of packets handled per receive call */
+ __u32 sps_maxpkts_call;
+ /* avg number of packets handled per receive call */
+ __u32 sps_avgpkts_call;
+ /* total number of pages locked */
+ __u64 sps_pagelocks;
+ /* total number of pages unlocked */
+ __u64 sps_pageunlocks;
+ /*
+ * Number of packets dropped in kernel other than errors (ether
+ * packets if ipath not configured, sma/mad, etc.)
+ */
+ __u64 sps_krdrops;
+ /* mlids for up to 4 infinipaths, indexed by infinipath # */
+ __u16 sps_mlid[4];
+ /* pad for future growth */
+ __u64 __sps_pad[45];
+};
+
+/*
+ * These are the status bits readable (in ascii form, 64bit value)
+ * from the "status" sysfs file.
+ */
+#define IPATH_STATUS_INITTED 0x1 /* basic initialization done */
+#define IPATH_STATUS_DISABLED 0x2 /* hardware disabled */
+/* Device has been disabled via admin request */
+#define IPATH_STATUS_ADMIN_DISABLED 0x4
+#define IPATH_STATUS_OIB_SMA 0x8 /* ipath_mad kernel SMA running */
+#define IPATH_STATUS_SMA 0x10 /* user SMA running */
+/* Chip has been found and initted */
+#define IPATH_STATUS_CHIP_PRESENT 0x20
+/* IB link is at ACTIVE, usable for data traffic */
+#define IPATH_STATUS_IB_READY 0x40
+/* link is configured, LID, MTU, etc. have been set */
+#define IPATH_STATUS_IB_CONF 0x80
+/* no link established, probably no cable */
+#define IPATH_STATUS_IB_NOCABLE 0x100
+/* A Fatal hardware error has occurred. */
+#define IPATH_STATUS_HWERROR 0x200
+
+/*
+ * The list of usermode accessible registers. Also see Reg_* later in file.
+ */
+typedef enum _ipath_ureg {
+ /* (RO) DMA RcvHdr to be used next. */
+ ur_rcvhdrtail = 0,
+ /* (RW) RcvHdr entry to be processed next by host. */
+ ur_rcvhdrhead = 1,
```

[PATCH 2 of 18] ipath – core driver header files

```
+ /* (RO) Index of next Eager index to use. */
+ ur_rcvegrindextail = 2,
+ /* (RW) Eager TID to be processed next */
+ ur_rcvegrindexhead = 3,
+ /* For internal use only; max register number. */
+ _IPATH_UregMax
+ } ipath_ureg;
+
+ /* bit values for spi_runtime_flags */
+ #define IPATH_RUNTIME_HT 0x1
+ #define IPATH_RUNTIME_PCIE 0x2
+ #define IPATH_RUNTIME_FORCE_WC_ORDER 0x4
+ #define IPATH_RUNTIME_RCVHDR_COPY 0x8
+
+ /*
+  * This structure is returned by ipath_userinit() immediately after
+  * open to get implementation-specific info, and info specific to this
+  * instance.
+  *
+  * This struct must have explicit pad fields where type sizes
+  * may result in different alignments between 32 and 64 bit
+  * programs, since the 64 bit * bit kernel requires the user code
+  * to have matching offsets
+  */
+ struct ipath_base_info {
+ /* version of hardware, for feature checking. */
+ __u32 spi_hw_version;
+ /* version of software, for feature checking. */
+ __u32 spi_sw_version;
+ /* InfiniPath port assigned, goes into sent packets */
+ __u32 spi_port;
+ /*
+  * IB MTU, packets IB data must be less than this.
+  * The MTU is in bytes, and will be a multiple of 4 bytes.
+  */
+ __u32 spi_mtu;
+ /*
+  * Size of a PIO buffer. Any given packet's total size must be less
+  * than this (in words). Included is the starting control word, so
+  * if 513 is returned, then total pkt size is 512 words or less.
+  */
+ __u32 spi_piosize;
+ /* size of the TID cache in infinipath, in entries */
+ __u32 spi_tidcnt;
+ /* size of the TID Eager list in infinipath, in entries */
+ __u32 spi_tidegrcnt;
+ /* size of a single receive header queue entry. */
+ __u32 spi_rcvhdrnt_size;
+ /*
+  * Count of receive header queue entries allocated.
+  * This may be less than the spu_rcvhdrnt passed in!.
+  */
+ };
```

[PATCH 2 of 18] ipath – core driver header files

```
+ */
+ __u32 spi_rcvhdr_cnt;
+
+ /* per-chip and other runtime features bitmap (IPATH_RUNTIME_*) */
+ __u32 spi_runtime_flags;
+
+ /* address where receive buffer queue is mapped into */
+ __u64 spi_rcvhdr_base;
+
+ /* user program. */
+
+ /* base address of eager TID receive buffers. */
+ __u64 spi_rcv_egrbufs;
+
+ /* Allocated by initialization code, not by protocol. */
+
+ /*
+ * Size of each TID buffer in host memory, starting at
+ * spi_rcv_egrbufs. The buffers are virtually contiguous.
+ */
+ __u32 spi_rcv_egrbufsize;
+ /*
+ * The special QP (queue pair) value that identifies an infinipath
+ * protocol packet from standard IB packets. More, probably much
+ * more, to be added.
+ */
+ __u32 spi_qpair;
+
+ /*
+ * User register base for init code, not to be used directly by
+ * protocol or applications.
+ */
+ __u64 __spi_uregbase;
+ /*
+ * Maximum buffer size in bytes that can be used in a single TID
+ * entry (assuming the buffer is aligned to this boundary). This is
+ * the minimum of what the hardware and software support Guaranteed
+ * to be a power of 2.
+ */
+ __u32 spi_tid_maxsize;
+ /*
+ * alignment of each pio send buffer (byte count
+ * to add to spi_piobufbase to get to second buffer)
+ */
+ __u32 spi_piobalign;
+ /*
+ * The index of the first pio buffer available to this process;
+ * needed to do lookup in spi_piobavailaddr; not added to
+ * spi_piobufbase.
+ */
+ __u32 spi_pioidex;
```

[PATCH 2 of 18] ipath – core driver header files

```
+ /* number of buffers mapped for this process */
+ __u32 spi_piocnt;
+
+ /*
+ * Base address of writeonly pio buffers for this process.
+ * Each buffer has spi_piosize words, and is aligned on spi_pioalign
+ * boundaries. spi_piocnt buffers are mapped from this address
+ */
+ __u64 spi_piobufbase;
+
+ /*
+ * Base address of readonly memory copy of the pioavail registers.
+ * There are 2 bits for each buffer.
+ */
+ __u64 spi_pioavailaddr;
+
+ /*
+ * Address where driver updates a copy of the interface and driver
+ * status (IPATH_STATUS_*) as a 64 bit value. It's followed by a
+ * string indicating hardware error, if there was one.
+ */
+ __u64 spi_status;
+
+ /* number of chip ports available to user processes */
+ __u32 spi_nports;
+ /* unit number of chip we are using */
+ __u32 spi_unit;
+ /* num bufs in each contiguous set */
+ __u32 spi_rcv_egrperchunk;
+ /* size in bytes of each contiguous set */
+ __u32 spi_rcv_egrchunksz;
+ /* total size of mmap to cover full revegrbuffers */
+ __u32 spi_rcv_egrbufstotlen;
+} __attribute__((aligned(8)));
+
+ /*
+ * This version number is given to the driver by the user code during
+ * initialization in the spu_userversion field of ipath_user_info, so
+ * the driver can check for compatibility with user code.
+ *
+ * The major version changes when data structures
+ * change in an incompatible way. The driver must be the same or higher
+ * for initialization to succeed. In some cases, a higher version
+ * driver will not interoperate with older software, and initialization
+ * will return an error.
+ */
+#define IPATH_USER_SWMAJOR 1
+
+ /*
+ * Minor version differences are always compatible
```


[PATCH 2 of 18] ipath – core driver header files

```
+ /*
+ * cache line aligned (64 byte) user address to
+ * which the rcvhdrtail register will be written by infinipath
+ * whenever it changes, so that no chip registers are read in
+ * the performance path.
+ */
+ __u64 spu_rcvhdraddr;
+
+ /*
+ * address of struct base_info to write to
+ */
+ __u64 spu_base_info;
+
+} __attribute__((aligned(8)));
+
+/* User commands. */
+
+#define IPATH_CMD_MIN 16
+
+#define IPATH_CMD_USER_INIT 16 /* set up userspace */
+#define IPATH_CMD_PORT_INFO 17 /* find out what resources we got */
+#define IPATH_CMD_RECV_CTRL 18 /* control receipt of packets */
+#define IPATH_CMD_TID_UPDATE 19 /* update expected TID entries */
+#define IPATH_CMD_TID_FREE 20 /* free expected TID entries */
+#define IPATH_CMD_SET_PART_KEY 21 /* add partition key */
+
+#define IPATH_CMD_MAX 21
+
+struct ipath_port_info {
+ __u32 num_active; /* number of active units */
+ __u32 unit; /* unit (chip) assigned to caller */
+ __u32 port; /* port on unit assigned to caller */
+};
+
+struct ipath_tid_info {
+ __u32 tidcnt;
+ /* make structure same size in 32 and 64 bit */
+ __u32 tid_unused;
+ /* virtual address of first page in transfer */
+ __u64 tidvaddr;
+ /* pointer (same size 32/64 bit) to __u16 tid array */
+ __u64 tidlist;
+
+ /*
+ * pointer (same size 32/64 bit) to bitmap of TIDs used
+ * for this call; checked for being large enough at open
+ */
+ __u64 tidmap;
+};
+
+struct ipath_cmd {
```

[PATCH 2 of 18] ipath – core driver header files

```
+ __u32 type; /* command type */
+ union {
+ struct ipath_tid_info tid_info;
+ struct ipath_user_info user_info;
+ /* address in userspace of struct ipath_port_info to
+ write result to */
+ __u64 port_info;
+ /* enable/disable receipt of packets */
+ __u32 recv_ctrl;
+ /* partition key to set */
+ __u16 part_key;
+ } cmd;
+};
+
+struct ipath_iovec {
+ /* Pointer to data, but same size 32 and 64 bit */
+ __u64 iov_base;
+
+ /*
+ * Length of data; don't need 64 bits, but want
+ * ipath_sendpkt to remain same size as before 32 bit changes, so...
+ */
+ __u64 iov_len;
+};
+
+ /*
+ * Describes a single packet for send. Each packet can have one or more
+ * buffers, but the total length (exclusive of IB headers) must be less
+ * than the MTU, and if using the PIO method, entire packet length,
+ * including IB headers, must be less than the ipath_piosize value (words).
+ * Use of this necessitates including sys/uio.h
+ */
+struct __ipath_sendpkt {
+ __u32 sps_flags; /* flags for packet (TBD) */
+ __u32 sps_cnt; /* number of entries to use in sps_iov */
+ /* array of iov's describing packet. TEMPORARY */
+ struct ipath_iovec sps_iov[4];
+};
+
+ /* Passed into SMA special file's ->read and ->write methods. */
+struct ipath_sma_pkt
+{
+ __u32 unit; /* unit on which to send packet */
+ __u64 data; /* address of payload in userspace */
+ __u32 len; /* length of payload */
+};
+
+ /*
+ * Data layout in I2C flash (for GUID, etc.)
+ * All fields are little-endian binary unless otherwise stated
+ */
```

[PATCH 2 of 18] ipath – core driver header files

```
+ #define IPATH_FLASH_VERSION 1
+ struct ipath_flash {
+ /* flash layout version (IPATH_FLASH_VERSION) */
+ __u8 if_fversion;
+ /* checksum protecting if_length bytes */
+ __u8 if_csum;
+ /*
+ * valid length (in use, protected by if_csum), including
+ * if_fversion and if_sum themselves)
+ */
+ __u8 if_length;
+ /* the GUID, in network order */
+ __u8 if_guid[8];
+ /* number of GUIDs to use, starting from if_guid */
+ __u8 if_numguid;
+ /* the board serial number, in ASCII */
+ char if_serial[12];
+ /* board mfg date (YYYYMMDD ASCII) */
+ char if_mfgdate[8];
+ /* last board rework/test date (YYYYMMDD ASCII) */
+ char if_testdate[8];
+ /* logging of error counts, TBD */
+ __u8 if_errcntp[4];
+ /* powered on hours, updated at driver unload */
+ __u8 if_powerhour[2];
+ /* ASCII free-form comment field */
+ char if_comment[32];
+ /* 78 bytes used, min flash size is 128 bytes */
+ __u8 if_future[50];
+ };
+
+ /*
+ * These are the counters implemented in the chip, and are listed in order.
+ * The InterCaps naming is taken straight from the chip spec.
+ */
+ struct infinipath_counters {
+ __u64 LBIntCnt;
+ __u64 LBFlowStallCnt;
+ __u64 Reserved1;
+ __u64 TxUnsupVLErrCnt;
+ __u64 TxDataPktCnt;
+ __u64 TxFlowPktCnt;
+ __u64 TxDwordCnt;
+ __u64 TxLenErrCnt;
+ __u64 TxMaxMinLenErrCnt;
+ __u64 TxUnderrunCnt;
+ __u64 TxFlowStallCnt;
+ __u64 TxDroppedPktCnt;
+ __u64 RxDroppedPktCnt;
+ __u64 RxDataPktCnt;
+ __u64 RxFlowPktCnt;
```

[PATCH 2 of 18] ipath – core driver header files

```
+ __u64 RxDwordCnt;
+ __u64 RxLenErrCnt;
+ __u64 RxMaxMinLenErrCnt;
+ __u64 RxICRCERRcnt;
+ __u64 RxVCRCErrCnt;
+ __u64 RxFlowCtrlErrCnt;
+ __u64 RxBadFormatCnt;
+ __u64 RxLinkProblemCnt;
+ __u64 RxEBPCnt;
+ __u64 RxLPCRCERRcnt;
+ __u64 RxBufOvflCnt;
+ __u64 RxTIDFullErrCnt;
+ __u64 RxTIDValidErrCnt;
+ __u64 RxPKeyMismatchCnt;
+ __u64 RxP0HdrEgrOvflCnt;
+ __u64 RxP1HdrEgrOvflCnt;
+ __u64 RxP2HdrEgrOvflCnt;
+ __u64 RxP3HdrEgrOvflCnt;
+ __u64 RxP4HdrEgrOvflCnt;
+ __u64 RxP5HdrEgrOvflCnt;
+ __u64 RxP6HdrEgrOvflCnt;
+ __u64 RxP7HdrEgrOvflCnt;
+ __u64 RxP8HdrEgrOvflCnt;
+ __u64 Reserved6;
+ __u64 Reserved7;
+ __u64 IBStatusChangeCnt;
+ __u64 IBLinkErrRecoveryCnt;
+ __u64 IBLinkDownedCnt;
+ __u64 IBSymbolErrCnt;
+ };
+
+ /*
+ * The next set of defines are for packet headers, and chip register
+ * and memory bits that are visible to and/or used by user-mode software
+ * The other bits that are used only by the driver or diags are in
+ * ipath_registers.h
+ */
+
+ /* RcvHdrFlags bits */
+#define INFINIPATH_RHF_LENGTH_MASK 0x7FF
+#define INFINIPATH_RHF_LENGTH_SHIFT 0
+#define INFINIPATH_RHF_RCVTYPE_MASK 0x7
+#define INFINIPATH_RHF_RCVTYPE_SHIFT 11
+#define INFINIPATH_RHF_EGRINDEX_MASK 0x7FF
+#define INFINIPATH_RHF_EGRINDEX_SHIFT 16
+#define INFINIPATH_RHF_H_ICRCERR 0x80000000
+#define INFINIPATH_RHF_H_VCRCERR 0x40000000
+#define INFINIPATH_RHF_H_PARITYERR 0x20000000
+#define INFINIPATH_RHF_H_LENERR 0x10000000
+#define INFINIPATH_RHF_H_MTUERR 0x08000000
+#define INFINIPATH_RHF_H_IHDRERR 0x04000000
```

[PATCH 2 of 18] ipath – core driver header files

```
+#define INFINIPATH_RHF_H_TIDERR 0x02000000
+#define INFINIPATH_RHF_H_MKERR 0x01000000
+#define INFINIPATH_RHF_H_IBERR 0x00800000
+#define INFINIPATH_RHF_L_SWA 0x00008000
+#define INFINIPATH_RHF_L_SWB 0x00004000
+
+/* infinipath header fields */
+#define INFINIPATH_I_VERS_MASK 0xF
+#define INFINIPATH_I_VERS_SHIFT 28
+#define INFINIPATH_I_PORT_MASK 0xF
+#define INFINIPATH_I_PORT_SHIFT 24
+#define INFINIPATH_I_TID_MASK 0x7FF
+#define INFINIPATH_I_TID_SHIFT 13
+#define INFINIPATH_I_OFFSET_MASK 0x1FFF
+#define INFINIPATH_I_OFFSET_SHIFT 0
+
+/* K_PktFlags bits */
+#define INFINIPATH_KPF_INTR 0x1
+
+/* SendPIO per-buffer control */
+#define INFINIPATH_SP_LENGTHP1_MASK 0x3FF
+#define INFINIPATH_SP_LENGTHP1_SHIFT 0
+#define INFINIPATH_SP_INTR 0x80000000
+#define INFINIPATH_SP_TEST 0x40000000
+#define INFINIPATH_SP_TESTEBP 0x20000000
+
+/* SendPIOAvail bits */
+#define INFINIPATH_SENDPIOAVAIL_BUSY_SHIFT 1
+#define INFINIPATH_SENDPIOAVAIL_CHECK_SHIFT 0
+
+#endif /* _IPATH_COMMON_H */
diff -r 747da2c0243d -r 4b2debbcae33 drivers/infiniband/hw/ipath/ipath_debug.h
--- /dev/null Thu Jan 1 00:00:00 1970 +0000
+++ b/drivers/infiniband/hw/ipath/ipath_debug.h Thu Mar 23 20:27:44 2006 -0800
@@ -0,0 +1,96 @@
+/*
+ * Copyright (c) 2003, 2004, 2005, 2006 PathScale, Inc. All rights reserved.
+ *
+ * This software is available to you under a choice of one of two
+ * licenses. You may choose to be licensed under the terms of the GNU
+ * General Public License (GPL) Version 2, available from the file
+ * COPYING in the main directory of this source tree, or the
+ * OpenIB.org BSD license below:
+ *
+ * Redistribution and use in source and binary forms, with or
+ * without modification, are permitted provided that the following
+ * conditions are met:
+ *
+ * - Redistributions of source code must retain the above
+ * copyright notice, this list of conditions and the following
+ * disclaimer.
```

[PATCH 2 of 18] ipath – core driver header files

```
+ *
+ * – Redistributions in binary form must reproduce the above
+ * copyright notice, this list of conditions and the following
+ * disclaimer in the documentation and/or other materials
+ * provided with the distribution.
+ *
+ * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
+ * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
+ * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
+ * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
+ * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
+ * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
+ * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
+ * SOFTWARE.
+ */
+
+#ifndef _IPATH_DEBUG_H
+#define _IPATH_DEBUG_H
+
+#ifndef _IPATH_DEBUGGING /* debugging enabled or not */
+#define _IPATH_DEBUGGING 1
+#endif
+
+#if _IPATH_DEBUGGING
+
+/*
+ * Mask values for debugging. The scheme allows us to compile out any
+ * of the debug tracing stuff, and if compiled in, to enable or disable
+ * dynamically. This can be set at modprobe time also:
+ * modprobe infinipath.ko infinipath_debug=7
+ */
+
+#define __IPATH_INFO 0x1 /* generic low verbosity stuff */
+#define __IPATH_DBG 0x2 /* generic debug */
+#define __IPATH_TRSAMPLE 0x8 /* generate trace buffer sample entries */
+/* leave some low verbosity spots open */
+#define __IPATH_VERBDBG 0x40 /* very verbose debug */
+#define __IPATH_PKTDBG 0x80 /* print packet data */
+/* print process startup (init)/exit messages */
+#define __IPATH_PROCDBG 0x100
+/* print mmap/nopage stuff, not using VDBG any more */
+#define __IPATH_MMDBG 0x200
+#define __IPATH_USER_SEND 0x1000 /* use user mode send */
+#define __IPATH_KERNEL_SEND 0x2000 /* use kernel mode send */
+#define __IPATH_EPKTDBG 0x4000 /* print ethernet packet data */
+#define __IPATH_SMADBG 0x8000 /* sma packet debug */
+#define __IPATH_IPATHDBG 0x10000 /* Ethernet (IPATH) general debug on */
+#define __IPATH_IPATHWARN 0x20000 /* Ethernet (IPATH) warnings on */
+#define __IPATH_IPATHERR 0x40000 /* Ethernet (IPATH) errors on */
+#define __IPATH_IPATHPD 0x80000 /* Ethernet (IPATH) packet dump on */
+#define __IPATH_IPATHTABLE 0x100000 /* Ethernet (IPATH) table dump on */
```

[PATCH 2 of 18] ipath – core driver header files

```
+
+##else /* _IPATH_DEBUGGING */
+
+/*
+ * define all of these even with debugging off, for the few places that do
+ * if(infinipath_debug & _IPATH_xyzzzy), but in a way that will make the
+ * compiler eliminate the code
+ */
+
+##define __IPATH_INFO 0x0 /* generic low verbosity stuff */
+##define __IPATH_DBG 0x0 /* generic debug */
+##define __IPATH_TRSAMPLE 0x0 /* generate trace buffer sample entries */
+##define __IPATH_VERBDBG 0x0 /* very verbose debug */
+##define __IPATH_PKTDBG 0x0 /* print packet data */
+##define __IPATH_PROCDBG 0x0 /* print process startup (init)/exit messages */
+/* print mmap/nopage stuff, not using VDBG any more */
+##define __IPATH_MMDBG 0x0
+##define __IPATH_EPKTDBG 0x0 /* print ethernet packet data */
+##define __IPATH_SMADBG 0x0 /* print process startup (init)/exit messages */##define
__IPATH_IPATHDBG 0x0 /* Ethernet (IPATH) table dump on */
+##define __IPATH_IPATHWARN 0x0 /* Ethernet (IPATH) warnings on */
+##define __IPATH_IPATHERR 0x0 /* Ethernet (IPATH) errors on */
+##define __IPATH_IPATHPD 0x0 /* Ethernet (IPATH) packet dump on */
+##define __IPATH_IPATHTABLE 0x0 /* Ethernet (IPATH) packet dump on */
+
+##endif /* _IPATH_DEBUGGING */
+
+##define __IPATH_VERBOSEDBG __IPATH_VERBDBG
+
+##endif /* _IPATH_DEBUG_H */
diff -r 747da2c0243d -r 4b2debbcae33 drivers/infiniband/hw/ipath/ipath_kernel.h
---- /dev/null Thu Jan 1 00:00:00 1970 +0000
+++ b/drivers/infiniband/hw/ipath/ipath_kernel.h Thu Mar 23 20:27:44 2006 -0800
@@ -0,0 +1,908 @@
+##ifndef _IPATH_KERNEL_H
+##define _IPATH_KERNEL_H
+/*
+ * Copyright (c) 2003, 2004, 2005, 2006 PathScale, Inc. All rights reserved.
+ *
+ * This software is available to you under a choice of one of two
+ * licenses. You may choose to be licensed under the terms of the GNU
+ * General Public License (GPL) Version 2, available from the file
+ * COPYING in the main directory of this source tree, or the
+ * OpenIB.org BSD license below:
+ *
+ * Redistribution and use in source and binary forms, with or
+ * without modification, are permitted provided that the following
+ * conditions are met:
+ *
+ * - Redistributions of source code must retain the above
+ * copyright notice, this list of conditions and the following
```

[PATCH 2 of 18] ipath – core driver header files

```
+ * disclaimer.
+ *
+ * – Redistributions in binary form must reproduce the above
+ * copyright notice, this list of conditions and the following
+ * disclaimer in the documentation and/or other materials
+ * provided with the distribution.
+ *
+ * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
+ * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
+ * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
+ * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
+ * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
+ * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
+ * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
+ * SOFTWARE.
+ */
+
+/*
+ * This header file is the base header file for infinipath kernel code
+ * ipath_user.h serves a similar purpose for user code.
+ */
+
+#include <linux/interrupt.h>
+#include <asm/io.h>
+
+#include "ipath_common.h"
+#include "ipath_debug.h"
+#include "ipath_registers.h"
+
+/* only s/w major version of InfiniPath we can handle */
+#define IPATH_CHIP_VERS_MAJ 2U
+
+/* don't care about this except printing */
+#define IPATH_CHIP_VERS_MIN 0U
+
+/* temporary, maybe always */
+extern struct infinipath_stats ipath_stats;
+
+#define IPATH_CHIP_SWVERSION IPATH_CHIP_VERS_MAJ
+
+struct ipath_portdata {
+ void **port_rcvegrbuf;
+ dma_addr_t *port_rcvegrbuf_phys;
+ /* rcvhdrq base, needs mmap before useful */
+ void *port_rcvhdrq;
+ /* kernel virtual address where hdrqtail is updated */
+ u64 *port_rcvhdrtail_kvaddr;
+ /* page * used for uaddr */
+ struct page *port_rcvhdrtail_pagep;
+ /*
+ * temp buffer for expected send setup, allocated at open, instead
```

[PATCH 2 of 18] ipath – core driver header files

```
+ * of each setup call
+ */
+ void *port_tid_pg_list;
+ /* when waiting for rcv or pioavail */
+ wait_queue_head_t port_wait;
+ /*
+ * rvegr bufs base, physical, must fit
+ * in 44 bits so 32 bit programs mmap64 44 bit works)
+ */
+ dma_addr_t port_rvegr_phys;
+ /* mmap of hdrq, must fit in 44 bits */
+ dma_addr_t port_rcvhdrq_phys;
+ /*
+ * the actual user address that we ipath_mlock'ed, so we can
+ * ipath_munlock it at close
+ */
+ unsigned long port_rcvhdrtail_uaddr;
+ /*
+ * number of opens on this instance (0 or 1; ignoring forks, dup,
+ * etc. for now)
+ */
+ int port_cnt;
+ /*
+ * how much space to leave at start of eager TID entries for
+ * protocol use, on each TID
+ */
+ /* instead of calculating it */
+ unsigned port_port;
+ /* chip offset of PIO buffers for this port */
+ u32 port_piobufs;
+ /* how many alloc_pages() chunks in port_rvegrbuf_pages */
+ u32 port_rvegrbuf_chunks;
+ /* how many egrbufs per chunk */
+ u32 port_rvegrbufs_perchunk;
+ /* order for port_rvegrbuf_pages */
+ size_t port_rvegrbuf_size;
+ /* rcvhdrq size (for freeing) */
+ size_t port_rcvhdrq_size;
+ /* next expected TID to check when looking for free */
+ u32 port_tidcursor;
+ /* next expected TID to check */
+ unsigned long port_flag;
+ /* WAIT_RCV that timed out, no interrupt */
+ u32 port_rcvwait_to;
+ /* WAIT_PIO that timed out, no interrupt */
+ u32 port_piowait_to;
+ /* WAIT_RCV already happened, no wait */
+ u32 port_rcvnowait;
+ /* WAIT_PIO already happened, no wait */
+ u32 port_pionowait;
+ /* total number of rcvhdrqfull errors */
```

[PATCH 2 of 18] ipath – core driver header files

```
+ u32 port_hdrqfull;
+ /* pid of process using this port */
+ pid_t port_pid;
+ /* same size as task_struct .comm[] */
+ char port_comm[16];
+ /* pkeys set by this use of this port */
+ u16 port_pkeys[4];
+ /* so file ops can get at unit */
+ struct ipath_devdata *port_dd;
+};
+
+struct sk_buff;
+
+/*
+ * control information for layered drivers
+ */
+struct _ipath_layer {
+ void *l_arg;
+};
+
+/* Verbs layer interface */
+struct _verbs_layer {
+ void *l_arg;
+ struct timer_list l_timer;
+};
+
+typedef u64 __bitwise ipath_err_t;
+
+struct ipath_devdata {
+ struct list_head ipath_list;
+
+ struct ipath_kregs const *ipath_kregs;
+ struct ipath_cregs const *ipath_cregs;
+
+ /* mem-mapped pointer to base of chip regs */
+ u64 __iomem *ipath_kregbase;
+ /* end of mem-mapped chip space; range checking */
+ u64 __iomem *ipath_kregend;
+ /* physical address of chip for io_remap, etc. */
+ unsigned long ipath_physaddr;
+ /* base of memory allocated for ipath_kregbase, for free */
+ u64 *ipath_kregalloc;
+ /*
+ * version of kregbase that doesn't have high bits set (for 32 bit
+ * programs, so mmap64 44 bit works)
+ */
+ u64 __iomem *ipath_kregvirt;
+ /*
+ * virtual address where port0 rcvhdrqtail updated for this unit.
+ * only written to by the chip, not the driver.
+ */
+};
```

[PATCH 2 of 18] ipath – core driver header files

```
+ volatile __le64 *ipath_hdrqtailptr;
+ dma_addr_t ipath_dma_addr;
+ /* ipath_cfgports pointers */
+ struct ipath_portdata **ipath_pd;
+ /* sk_buffs used by port 0 eager receive queue */
+ struct sk_buff **ipath_port0_skbs;
+ /* kvirt address of 1st 2k pio buffer */
+ void __iomem *ipath_pio2kbase;
+ /* kvirt address of 1st 4k pio buffer */
+ void __iomem *ipath_pio4kbase;
+ /*
+ * points to area where PIOavail registers will be DMA'ed.
+ * Has to be on a page of it's own, because the page will be
+ * mapped into user program space. This copy is *ONLY* ever
+ * written by DMA, not by the driver! Need a copy per device
+ * when we get to multiple devices
+ */
+ volatile __le64 *ipath_pioavailregs_dma;
+ /* physical address where updates occur */
+ dma_addr_t ipath_pioavailregs_phys;
+ struct _ipath_layer ipath_layer;
+ /* setup intr */
+ int (*ipath_f_intrsetup)(struct ipath_devdata *);
+ /* setup on-chip bus config */
+ int (*ipath_f_bus)(struct ipath_devdata *, struct pci_dev *);
+ /* hard reset chip */
+ int (*ipath_f_reset)(struct ipath_devdata *);
+ int (*ipath_f_get_boardname)(struct ipath_devdata *, char *,
+ size_t);
+ void (*ipath_f_init_hwerrors)(struct ipath_devdata *);
+ void (*ipath_f_handle_hwerrors)(struct ipath_devdata *, char *,
+ size_t);
+ void (*ipath_f_quiet_serdes)(struct ipath_devdata *);
+ int (*ipath_f_bringup_serdes)(struct ipath_devdata *);
+ int (*ipath_f_early_init)(struct ipath_devdata *);
+ void (*ipath_f_clear_tids)(struct ipath_devdata *, unsigned);
+ void (*ipath_f_put_tid)(struct ipath_devdata *, u64 __iomem*,
+ u32, unsigned long);
+ void (*ipath_f_tidtemplate)(struct ipath_devdata *);
+ void (*ipath_f_cleanup)(struct ipath_devdata *);
+ void (*ipath_f_setextled)(struct ipath_devdata *, u64, u64);
+ /* fill out chip-specific fields */
+ int (*ipath_f_get_base_info)(struct ipath_portdata *, void *);
+ struct _verbs_layer verbs_layer;
+ /* total dwords sent (summed from counter) */
+ u64 ipath_sword;
+ /* total dwords rcvd (summed from counter) */
+ u64 ipath_rword;
+ /* total packets sent (summed from counter) */
+ u64 ipath_spkts;
+ /* total packets rcvd (summed from counter) */
```

[PATCH 2 of 18] ipath – core driver header files

```
+ u64 ipath_rpkts;
+ /* ipath_statusp initially points to this. */
+ u64 _ipath_status;
+ /* GUID for this interface, in network order */
+ u64 ipath_guid;
+ /*
+ * aggregate of error bits reported since last cleared, for
+ * limiting of error reporting
+ */
+ ipath_err_t ipath_lasterror;
+ /*
+ * aggregate of error bits reported since last cleared, for
+ * limiting of hwerror reporting
+ */
+ ipath_err_t ipath_lasthwerror;
+ /*
+ * errors masked because they occur too fast, also includes errors
+ * that are always ignored (ipath_ignorederrs)
+ */
+ ipath_err_t ipath_maskederrs;
+ /* time in jiffies at which to re-enable maskederrs */
+ unsigned long ipath_unmasktime;
+ /*
+ * errors always ignored (masked), at least for a given
+ * chip/device, because they are wrong or not useful
+ */
+ ipath_err_t ipath_ignorederrs;
+ /* count of egrfull errors, combined for all ports */
+ u64 ipath_last_tidfull;
+ /* for ipath_qcheck() */
+ u64 ipath_lastport0rcv_cnt;
+ /* template for writing TIDs */
+ u64 ipath_tidtemplate;
+ /* value to write to free TIDs */
+ u64 ipath_tidinvalid;
+ /* PE-800 rcv interrupt setup */
+ u64 ipath_rhdrhead_intr_off;
+
+ /* size of memory at ipath_kregbase */
+ u32 ipath_kregsize;
+ /* number of registers used for pioavail */
+ u32 ipath_pioavregs;
+ /* IPATH_POLL, etc. */
+ u32 ipath_flags;
+ /* ipath_flags sma is waiting for */
+ u32 ipath_sma_state_wanted;
+ /* last buffer for user use, first buf for kernel use is this
+ * index. */
+ u32 ipath_lastport_piobuf;
+ /* is a stats timer active */
+ u32 ipath_stats_timer_active;
```

[PATCH 2 of 18] ipath – core driver header files

```
+ /* dwords sent read from counter */
+ u32 ipath_lastsword;
+ /* dwords received read from counter */
+ u32 ipath_lastrword;
+ /* sent packets read from counter */
+ u32 ipath_lastspkts;
+ /* received packets read from counter */
+ u32 ipath_lastrpkts;
+ /* pio bufs allocated per port */
+ u32 ipath_pbufsport;
+ /*
+ * number of ports configured as max; zero is set to number chip
+ * supports, less gives more pio bufs/port, etc.
+ */
+ u32 ipath_cfgports;
+ /* port0 rcvhdrq head offset */
+ u32 ipath_port0head;
+ /* count of port 0 hdrqfull errors */
+ u32 ipath_p0_hdrqfull;
+
+ /*
+ * (*cfgports) used to suppress multiple instances of same
+ * port staying stuck at same point
+ */
+ u32 *ipath_lastrcvhdrqtails;
+ /*
+ * (*cfgports) used to suppress multiple instances of same
+ * port staying stuck at same point
+ */
+ u32 *ipath_lastegrheads;
+ /*
+ * index of last piobuffer we used. Speeds up searching, by
+ * starting at this point. Doesn't matter if multiple cpu's use and
+ * update, last updater is only write that matters. Whenever it
+ * wraps, we update shadow copies. Need a copy per device when we
+ * get to multiple devices
+ */
+ u32 ipath_lastpioindex;
+ /* max length of freezemsg */
+ u32 ipath_freezelen;
+ /*
+ * consecutive times we wanted a PIO buffer but were unable to
+ * get one
+ */
+ u32 ipath_consec_nopiobuf;
+ /*
+ * hint that we should update ipath_pioavailshadow before
+ * looking for a PIO buffer
+ */
+ u32 ipath_upd_pio_shadow;
+ /* so we can rewrite it after a chip reset */
```

[PATCH 2 of 18] ipath – core driver header files

```
+ u32 ipath_pcibar0;
+ /* so we can rewrite it after a chip reset */
+ u32 ipath_pcibar1;
+ /* sequential tries for SMA send and no bufs */
+ u32 ipath_nosma_bufs;
+ /* duration (seconds) ipath_nosma_bufs set */
+ u32 ipath_nosma_secs;
+
+ /* HT/PCI Vendor ID (here for NodeInfo) */
+ u16 ipath_vendorid;
+ /* HT/PCI Device ID (here for NodeInfo) */
+ u16 ipath_deviceid;
+ /* offset in HT config space of slave/primary interface block */
+ u8 ipath_ht_slave_off;
+ /* for write combining settings */
+ unsigned long ipath_wc_cookie;
+ /* ref count for each pkey */
+ atomic_t ipath_pkeyrefs[4];
+ /* shadow copy of all exptids physaddr; used only by funcsim */
+ u64 *ipath_tidsimshadow;
+ /* shadow copy of struct page *'s for exp tid pages */
+ struct page **ipath_pageshadow;
+ /* lock to workaround chip bug 9437 */
+ spinlock_t ipath_tid_lock;
+
+ /*
+ * IPATH_STATUS_*,
+ * this address is mapped readonly into user processes so they can
+ * get status cheaply, whenever they want.
+ */
+ u64 *ipath_statusp;
+ /* freeze msg if hw error put chip in freeze */
+ char *ipath_freezmsg;
+ /* pci access data structure */
+ struct pci_dev *pcidev;
+ struct cdev *cdev;
+ struct class_device *class_dev;
+ /* timer used to prevent stats overflow, error throttling, etc. */
+ struct timer_list ipath_stats_timer;
+ /* check for stale messages in rcv queue */
+ /* only allow one intr at a time. */
+ unsigned long ipath_rcv_pending;
+
+ /*
+ * Shadow copies of registers; size indicates read access size.
+ * Most of them are readonly, but some are write-only register,
+ * where we manipulate the bits in the shadow copy, and then write
+ * the shadow copy to infinipath.
+ *
+ * We deliberately make most of these 32 bits, since they have
+ * restricted range. For any that we read, we won't to generate 32
```

[PATCH 2 of 18] ipath – core driver header files

```
+ * bit accesses, since Opteron will generate 2 separate 32 bit HT
+ * transactions for a 64 bit read, and we want to avoid unnecessary
+ * HT transactions.
+ */
+
+ /* This is the 64 bit group */
+
+ /*
+ * shadow of pioavail, check to be sure it's large enough at
+ * init time.
+ */
+ unsigned long ipath_pioavailshadow[8];
+ /* shadow of kr_gpio_out, for rmw ops */
+ u64 ipath_gpio_out;
+ /* kr_revision shadow */
+ u64 ipath_revision;
+ /*
+ * shadow of ibcctrl, for interrupt handling of link changes,
+ * etc.
+ */
+ u64 ipath_ibcctrl;
+ /*
+ * last ibcstatus, to suppress "duplicate" status change messages,
+ * mostly from 2 to 3
+ */
+ u64 ipath_lastibcstat;
+ /* hwerrmask shadow */
+ ipath_err_t ipath_hwerrmask;
+ /* interrupt config reg shadow */
+ u64 ipath_intconfig;
+ /* kr_sendpiobufbase value */
+ u64 ipath_piobufbase;
+
+ /* these are the "32 bit" regs */
+
+ /*
+ * number of GUIDs in the flash for this interface; may need some
+ * rethinking for setting on other ifaces
+ */
+ u32 ipath_nguid;
+ /*
+ * the following two are 32-bit bitmasks, but {test,clear,set}_bit
+ * all expect bit fields to be "unsigned long"
+ */
+ /* shadow kr_rcvctrl */
+ unsigned long ipath_rcvctrl;
+ /* shadow kr_sendctrl */
+ unsigned long ipath_sendctrl;
+
+ /* value we put in kr_rcvhdrcnt */
+ u32 ipath_rcvhdrcnt;
```

[PATCH 2 of 18] ipath – core driver header files

```
+ /* value we put in kr_rcvhdrsize */
+ u32 ipath_rcvhdrsize;
+ /* value we put in kr_rcvhdrentsize */
+ u32 ipath_rcvhdrentsize;
+ /* offset of last entry in rcvhdrq */
+ u32 ipath_hdrqlast;
+ /* kr_portcnt value */
+ u32 ipath_portcnt;
+ /* kr_pagealign value */
+ u32 ipath_palign;
+ /* number of "2KB" PIO buffers */
+ u32 ipath_piobcnt2k;
+ /* size in bytes of "2KB" PIO buffers */
+ u32 ipath_piosize2k;
+ /* number of "4KB" PIO buffers */
+ u32 ipath_piobcnt4k;
+ /* size in bytes of "4KB" PIO buffers */
+ u32 ipath_piosize4k;
+ /* kr_rcvegrbase value */
+ u32 ipath_rcvegrbase;
+ /* kr_rcvegrcnt value */
+ u32 ipath_rcvegrcnt;
+ /* kr_rcvtidbase value */
+ u32 ipath_rcvtidbase;
+ /* kr_rcvtidcnt value */
+ u32 ipath_rcvtidcnt;
+ /* kr_sendregbase */
+ u32 ipath_sregbase;
+ /* kr_userregbase */
+ u32 ipath_uregbase;
+ /* kr_counterregbase */
+ u32 ipath_cregbase;
+ /* shadow the control register contents */
+ u32 ipath_control;
+ /* shadow the gpio output contents */
+ u32 ipath_extctrl;
+ /* PCI revision register (HTC rev on FPGA) */
+ u32 ipath_pcirev;
+
+ /* chip address space used by 4k pio buffers */
+ u32 ipath_4kalign;
+ /* The MTU programmed for this unit */
+ u32 ipath_ibmtu;
+ /*
+ * The max size IB packet, included IB headers that we can send.
+ * Starts same as ipath_piosize, but is affected when ibmtu is
+ * changed, or by size of eager buffers
+ */
+ u32 ipath_ibmaxlen;
+ /*
+ * ibmaxlen at init time, limited by chip and by receive buffer
```

[PATCH 2 of 18] ipath – core driver header files

```
+ * size. Not changed after init.
+ */
+ u32 ipath_init_ibmaxlen;
+ /* size of each rcvegrbuffer */
+ u32 ipath_rcvegrbufsize;
+ /* width (2,4,8,16,32) from HT config reg */
+ u32 ipath_htwidth;
+ /* HT speed (200,400,800,1000) from HT config */
+ u32 ipath_htspeed;
+ /* ports waiting for PIOavail intr */
+ unsigned long ipath_portpiowait;
+ /*
+ * number of sequential ibcstatus change for polling active/quiet
+ * (i.e., link not coming up).
+ */
+ u32 ipath_ibpollcnt;
+ /* low and high portions of MSI capability/vector */
+ u32 ipath_msi_lo;
+ /* saved after PCIe init for restore after reset */
+ u32 ipath_msi_hi;
+ /* MSI data (vector) saved for restore */
+ u16 ipath_msi_data;
+ /* MLID programmed for this instance */
+ u16 ipath_mlid;
+ /* LID programmed for this instance */
+ u16 ipath_lid;
+ /* list of pkeys programmed; 0 if not set */
+ u16 ipath_pkeys[4];
+ /* ASCII serial number, from flash */
+ u8 ipath_serial[12];
+ /* human readable board version */
+ u8 ipath_boardversion[80];
+ /* chip major rev, from ipath_revision */
+ u8 ipath_majrev;
+ /* chip minor rev, from ipath_revision */
+ u8 ipath_minrev;
+ /* board rev, from ipath_revision */
+ u8 ipath_boardrev;
+ /* unit # of this chip, if present */
+ int ipath_unit;
+ /* saved for restore after reset */
+ u8 ipath_pci_cacheline;
+ /* LID mask control */
+ u8 ipath_lmc;
+};
+
+extern u64 *ipath_port0_rcvhdrtail;
+extern dma_addr_t ipath_port0_rcvhdrtail_dma;
+
+#define IPATH_PORT0_RCVHDRTAIL_SIZE PAGE_SIZE
+
```

[PATCH 2 of 18] ipath – core driver header files

```
+extern struct list_head ipath_dev_list;
+extern spinlock_t ipath_devs_lock;
+extern struct ipath_devdata *ipath_lookup(int unit);
+
+extern u16 ipath_layer_rcv_opcode;
+extern int ipath_verbs_registered;
+extern int __ipath_layer_intr(struct ipath_devdata *, u32);
+extern int ipath_layer_intr(struct ipath_devdata *, u32);
+extern int __ipath_layer_rcv(struct ipath_devdata *, void *,
+ struct sk_buff *);
+extern int __ipath_layer_rcv_lid(struct ipath_devdata *, void *);
+extern int __ipath_verbs_piobufavail(struct ipath_devdata *);
+extern int __ipath_verbs_rcv(struct ipath_devdata *, void *, void *, u32);
+
+void ipath_layer_add(struct ipath_devdata *);
+void ipath_layer_del(struct ipath_devdata *);
+
+int ipath_init_chip(struct ipath_devdata *, int);
+int ipath_enable_wc(struct ipath_devdata *dd);
+void ipath_disable_wc(struct ipath_devdata *dd);
+int ipath_count_units(int *npresentp, int *nupp, u32 *maxportsp);
+void ipath_shutdown_device(struct ipath_devdata *);
+
+struct file_operations;
+int ipath_cdev_init(int minor, char *name, struct file_operations *fops,
+ struct cdev **cdevp, struct class_device **class_devp);
+void ipath_cdev_cleanup(struct cdev **cdevp,
+ struct class_device **class_devp);
+
+int ipath_diag_init(void);
+void ipath_diag_cleanup(void);
+void ipath_diag_bringup_link(struct ipath_devdata *);
+
+int ipath_sma_init(void);
+void ipath_sma_cleanup(void);
+
+int ipath_user_add(struct ipath_devdata *dd);
+void ipath_user_del(struct ipath_devdata *dd);
+
+struct sk_buff *ipath_alloc_skb(struct ipath_devdata *dd, gfp_t);
+
+extern int ipath_diag_inuse;
+
+irqreturn_t ipath_intr(int irq, void *devid, struct pt_regs *regs);
+void ipath_decode_err(char *buf, size_t blen, ipath_err_t err);
+#if __IPATH_INFO || __IPATH_DBG
+extern const char *ipath_ibcstatus_str[];
+#endif
+
+/* clean up any per-chip chip-specific stuff */
+void ipath_chip_cleanup(struct ipath_devdata *);
```

[PATCH 2 of 18] ipath – core driver header files

```
+/* clean up any chip type-specific stuff */
+void ipath_chip_done(void);
+
+/* check to see if we have to force ordering for write combining */
+int ipath_unordered_wc(void);
+
+void ipath_disarm_piobufs(struct ipath_devdata *, unsigned first,
+ unsigned cnt);
+
+#define IPATH_SMA_HDRSZ (8+12+8) /* LRH+BTH+DETH */
+#define IPATH_SMA_MAX_PKTSZ (IPATH_SMA_HDRSZ+256)
+#define IPATH_NUM_SMA_PKTS 16
+
+int ipath_create_rcvhdrq(struct ipath_devdata *, struct ipath_portdata *);
+void ipath_free_pddata(struct ipath_devdata *, u32, int);
+
+int ipath_parse_ushort(const char *str, unsigned short *valp);
+
+extern unsigned ipath_sma_first;
+extern unsigned ipath_sma_next;
+extern int ipath_sma_inuse;
+extern spinlock_t ipath_sma_lock;
+extern u8 ipath_sma_data_bufs[IPATH_NUM_SMA_PKTS + 1][IPATH_SMA_MAX_PKTSZ];
+extern u8 *ipath_sma_data_spare;
+extern wait_queue_head_t ipath_sma_wait;
+extern wait_queue_head_t ipath_sma_state_wait;
+
+extern struct _ipath_sma_rpkt {
+ /* length of received packet; non-zero if queued */
+ u32 len;
+ /* unit number of interface packet was received from */
+ u32 unit;
+ u8 *buf;
+} ipath_sma_data[IPATH_NUM_SMA_PKTS];
+
+int ipath_wait_linkstate(struct ipath_devdata *, u32, int);
+void ipath_set_ib_lstate(struct ipath_devdata *, int);
+void ipath_kreceive(struct ipath_devdata *);
+int ipath_setrcvhdrsize(struct ipath_devdata *, unsigned);
+int ipath_reset_device(int);
+void ipath_get_faststats(unsigned long);
+
+/* for use in system calls, where we want to know device type, etc. */
+#define port_fp(fp) ((struct ipath_portdata *) (fp)->private_data)
+
+/*
+ * values for ipath_flags
+ */
+/* The chip is up and initted */
+#define IPATH_INITTED 0x2
+/* set if any user code has set kr_rcvhdrsize */
```

[PATCH 2 of 18] ipath – core driver header files

```
+#define IPATH_RCVHDRSZ_SET 0x4
+ /* The chip is present and valid for accesses */
+#define IPATH_PRESENT 0x8
+ /* HT link0 is only 8 bits wide, ignore upper byte crc
+ * errors, etc. */
+#define IPATH_8BIT_IN_HT0 0x10
+ /* HT link1 is only 8 bits wide, ignore upper byte crc
+ * errors, etc. */
+#define IPATH_8BIT_IN_HT1 0x20
+ /* The link is down */
+#define IPATH_LINKDOWN 0x40
+ /* The link level is up (0x11) */
+#define IPATH_LINKINIT 0x80
+ /* The link is in the armed (0x21) state */
+#define IPATH_LINKARMED 0x100
+ /* The link is in the active (0x31) state */
+#define IPATH_LINKACTIVE 0x200
+ /* link current state is unknown */
+#define IPATH_LINKUNK 0x400
+ /* no IB cable, or no device on IB cable */
+#define IPATH_NOCABLE 0x4000
+ /* Supports port zero per packet receive interrupts via
+ * GPIO */
+#define IPATH_GPIO_INTR 0x8000
+ /* uses the coded 4byte TID, not 8 byte */
+#define IPATH_4BYTE_TID 0x10000
+ /* packet/word counters are 32 bit, else those 4 counters
+ * are 64bit */
+#define IPATH_32BITCOUNTERS 0x20000
+ /* can miss port0 rx interrupts */
+#define IPATH_POLL_RX_INTR 0x40000
+#define IPATH_DISABLED 0x80000 /* administratively disabled */
+
+ /* portdata flag bit offsets */
+ /* waiting for a packet to arrive */
+#define IPATH_PORT_WAITING_RCV 2
+ /* waiting for a PIO buffer to be available */
+#define IPATH_PORT_WAITING_PIO 3
+
+ /* free up any allocated data at closes */
+void ipath_free_data(struct ipath_portdata *dd);
+int ipath_waitfor_mdio_cmdready(struct ipath_devdata *);
+int ipath_waitfor_complete(struct ipath_devdata *, ipath_kreg, u64, u64 *);
+u32 __iomem *ipath_getpiobuf(struct ipath_devdata *, u32 *);
+ /* init PE-800-specific func */
+void ipath_init_pe800_funcs(struct ipath_devdata *);
+ /* init HT-400-specific func */
+void ipath_init_ht400_funcs(struct ipath_devdata *);
+void ipath_get_guid(struct ipath_devdata *);
+u64 ipath_snap_cntr(struct ipath_devdata *, ipath_creg);
+
```

[PATCH 2 of 18] ipath – core driver header files

```
+/*
+ * number of words used for protocol header if not set by ipath_userinit();
+ */
+#define IPATH_DFLT_RCVHDRSIZE 9
+
+#define IPATH_MDIO_CMD_WRITE 1
+#define IPATH_MDIO_CMD_READ 2
+#define IPATH_MDIO_CLD_DIV 25 /* to get 2.5 Mhz mdio clock */
+#define IPATH_MDIO_CMDVALID 0x40000000 /* bit 30 */
+#define IPATH_MDIO_DATAVALID 0x80000000 /* bit 31 */
+#define IPATH_MDIO_CTRL_STD 0x0
+
+static inline u64 ipath_mdio_req(int cmd, int dev, int reg, int data)
+{
+ return (((u64) IPATH_MDIO_CLD_DIV) << 32) |
+ (cmd << 26) |
+ (dev << 21) |
+ (reg << 16) |
+ (data & 0xFFFF);
+}
+
+ /* signal and fifo status, in bank 31 */
+#define IPATH_MDIO_CTRL_XGXS_REG_8 0x8
+ /* controls loopback, redundancy */
+#define IPATH_MDIO_CTRL_8355_REG_1 0x10
+ /* premp, encdec, etc. */
+#define IPATH_MDIO_CTRL_8355_REG_2 0x11
+ /* Kchars, etc. */
+#define IPATH_MDIO_CTRL_8355_REG_6 0x15
+#define IPATH_MDIO_CTRL_8355_REG_9 0x18
+#define IPATH_MDIO_CTRL_8355_REG_10 0x1D
+
+int ipath_get_user_pages(unsigned long, size_t, struct page **);
+int ipath_get_user_pages_nocopy(unsigned long, struct page **);
+void ipath_release_user_pages(struct page **, size_t);
+void ipath_release_user_pages_on_close(struct page **, size_t);
+int ipath_eeprom_read(struct ipath_devdata *, u8, void *, int);
+int ipath_eeprom_write(struct ipath_devdata *, u8, const void *, int);
+
+ /* these are used for the registers that vary with port */
+void ipath_write_kreg_port(const struct ipath_devdata *, ipath_kreg,
+ unsigned, u64);
+u64 ipath_read_kreg64_port(const struct ipath_devdata *, ipath_kreg,
+ unsigned);
+
+ /*
+ * We could have a single register get/put routine, that takes a group type,
+ * but this is somewhat clearer and cleaner. It also gives us some error
+ * checking. 64 bit register reads should always work, but are inefficient
+ * on opteron (the northbridge always generates 2 separate HT 32 bit reads),
+ * so we use kreg32 wherever possible. User register and counter register
```

[PATCH 2 of 18] ipath – core driver header files

```
+ * reads are always 32 bit reads, so only one form of those routines.
+ */
+
+/*
+ * At the moment, none of the s-registers are writable, so no
+ * ipath_write_sreg(), and none of the c-registers are writable, so no
+ * ipath_write_creg().
+ */
+
+/**
+ * ipath_read_ureg32 – read 32-bit virtualized per-port register
+ * @dd: device
+ * @regno: register number
+ * @port: port number
+ *
+ * Return the contents of a register that is virtualized to be per port.
+ * Prints a debug message and returns -1 on errors (not distinguishable from
+ * valid contents at runtime; we may add a separate error variable at some
+ * point).
+ *
+ * This is normally not used by the kernel, but may be for debugging, and
+ * has a different implementation than user mode, which is why it's not in
+ * _common.h.
+ */
+static inline u32 ipath_read_ureg32(const struct ipath_devdata *dd,
+ ipath_ureg regno, int port)
+{
+ if (!dd->ipath_kregbase)
+ return 0;
+
+ return readl(regno + (u64 __iomem *)
+ (dd->ipath_uregbase +
+ (char __iomem *)dd->ipath_kregbase +
+ dd->ipath_palign * port));
+}
+
+/**
+ * ipath_write_ureg – write 32-bit virtualized per-port register
+ * @dd: device
+ * @regno: register number
+ * @value: value
+ * @port: port
+ *
+ * Write the contents of a register that is virtualized to be per port.
+ */
+static inline void ipath_write_ureg(const struct ipath_devdata *dd,
+ ipath_ureg regno, u64 value, int port)
+{
+ u64 __iomem *ubase = (u64 __iomem *)
+ (dd->ipath_uregbase + (char __iomem *) dd->ipath_kregbase +
+ dd->ipath_palign * port);
```

[PATCH 2 of 18] ipath – core driver header files

```
+ if (dd->ipath_kregbase)
+ writeq(value, &ubase[regno]);
+}
+
+static inline u32 ipath_read_kreg32(const struct ipath_devdata *dd,
+ ipath_kreg regno)
+{
+ if (!dd->ipath_kregbase)
+ return -1;
+ return readl((u32 __iomem *) & dd->ipath_kregbase[regno]);
+}
+
+static inline u64 ipath_read_kreg64(const struct ipath_devdata *dd,
+ ipath_kreg regno)
+{
+ if (!dd->ipath_kregbase)
+ return -1;
+
+ return readq(&dd->ipath_kregbase[regno]);
+}
+
+static inline void ipath_write_kreg(const struct ipath_devdata *dd,
+ ipath_kreg regno, u64 value)
+{
+ if (dd->ipath_kregbase)
+ writeq(value, &dd->ipath_kregbase[regno]);
+}
+
+static inline u64 ipath_read_creg(const struct ipath_devdata *dd,
+ ipath_sreg regno)
+{
+ if (!dd->ipath_kregbase)
+ return 0;
+
+ return readq(regno + (u64 __iomem *)
+ (dd->ipath_cregbase +
+ (char __iomem *)dd->ipath_kregbase));
+}
+
+static inline u32 ipath_read_creg32(const struct ipath_devdata *dd,
+ ipath_sreg regno)
+{
+ if (!dd->ipath_kregbase)
+ return 0;
+ return readl(regno + (u64 __iomem *)
+ (dd->ipath_cregbase +
+ (char __iomem *)dd->ipath_kregbase));
+}
+
+/*
+ * sysfs interface.
```

[PATCH 2 of 18] ipath – core driver header files

```

+ */
+
+struct device_driver;
+
+extern const char ipath_core_version[];
+
+int ipath_driver_create_group(struct device_driver *);
+void ipath_driver_remove_group(struct device_driver *);
+
+int ipath_device_create_group(struct device *, struct ipath_devdata *);
+void ipath_device_remove_group(struct device *, struct ipath_devdata *);
+int ipath_expose_reset(struct device *);
+
+int ipath_init_ipathfs(void);
+void ipath_exit_ipathfs(void);
+int ipathfs_add_device(struct ipath_devdata *);
+int ipathfs_remove_device(struct ipath_devdata *);
+
+/*
+ * Flush write combining store buffers (if present) and perform a write
+ * barrier.
+ */
+#if defined(CONFIG_X86_64)
+#define ipath_flush_wc() asm volatile("sfence" ::: "memory")
+#else
+#define ipath_flush_wc() wmb()
+#endif
+
+extern unsigned ipath_debug; /* debugging bit mask */
+
+const char *ipath_get_unit_name(int unit);
+
+extern struct mutex ipath_mutex;
+
+#define IPATH_DRV_NAME "ipath_core"
+#define IPATH_MAJOR 233
+#define IPATH_SMA_MINOR 128
+#define IPATH_DIAG_MINOR 129
+#define IPATH_NMINORS 130
+
+#define ipath_dev_err(dd,fmt,...) \
+ do { \
+ const struct ipath_devdata * __dd = (dd); \
+ if (__dd->pcidev) \
+ dev_err(&__dd->pcidev->dev, "%s: " fmt, \
+ ipath_get_unit_name(__dd->ipath_unit), \
+ ##__VA_ARGS__); \
+ else \
+ printk(KERN_ERR IPATH_DRV_NAME ": %s: " fmt, \
+ ipath_get_unit_name(__dd->ipath_unit), \
+ ##__VA_ARGS__); \

```

[PATCH 2 of 18] ipath – core driver header files

```
+ } while (0)
+
+#if _IPATH_DEBUGGING
+
+# define __IPATH_DBG_WHICH(which,fmt,...) \
+ do { \
+ if(unlikely(ipath_debug&(which))) \
+ printk(KERN_DEBUG IPATH_DRV_NAME ": %s: " fmt, \
+ __func__,##__VA_ARGS__); \
+ } while(0)
+
+# define ipath_dbg(fmt,...) \
+ __IPATH_DBG_WHICH(__IPATH_DBG,fmt,##__VA_ARGS__)
+# define ipath_cdbg(which,fmt,...) \
+ __IPATH_DBG_WHICH(__IPATH_##which##DBG,fmt,##__VA_ARGS__)
+
+#else /* !_IPATH_DEBUGGING */
+
+# define ipath_dbg(fmt,...)
+# define ipath_cdbg(which,fmt,...)
+
+#endif /* _IPATH_DEBUGGING */
+
+#endif /* _IPATH_KERNEL_H */
diff -r 747da2c0243d -r 4b2debbcae33 drivers/infiniband/hw/ipath/ipath_registers.h
--- /dev/null Thu Jan 1 00:00:00 1970 +0000
+++ b/drivers/infiniband/hw/ipath/ipath_registers.h Thu Mar 23 20:27:44 2006 -0800
@@ -0,0 +1,444 @@
+/*
+ * Copyright (c) 2003, 2004, 2005, 2006 PathScale, Inc. All rights reserved.
+ *
+ * This software is available to you under a choice of one of two
+ * licenses. You may choose to be licensed under the terms of the GNU
+ * General Public License (GPL) Version 2, available from the file
+ * COPYING in the main directory of this source tree, or the
+ * OpenIB.org BSD license below:
+ *
+ * Redistribution and use in source and binary forms, with or
+ * without modification, are permitted provided that the following
+ * conditions are met:
+ *
+ * - Redistributions of source code must retain the above
+ * copyright notice, this list of conditions and the following
+ * disclaimer.
+ *
+ * - Redistributions in binary form must reproduce the above
+ * copyright notice, this list of conditions and the following
+ * disclaimer in the documentation and/or other materials
+ * provided with the distribution.
+ *
+ * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
```

[PATCH 2 of 18] ipath – core driver header files

```
+ * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
+ * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
+ * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
+ * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
+ * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
+ * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
+ * SOFTWARE.
+ */
+
+#ifndef _IPATH_REGISTERS_H
+#define _IPATH_REGISTERS_H
+
+/*
+ * This file should only be included by kernel source, and by the diags.
+ * It defines the registers, and their contents, for the InfiniPath HT-400 chip
+ */
+
+/*
+ * These are the InfiniPath register and buffer bit definitions,
+ * that are visible to software, and needed only by the kernel
+ * and diag code. A few, that are visible to protocol and user
+ * code are in ipath_common.h. Some bits are specific
+ * to a given chip implementation, and have been moved to the
+ * chip-specific source file
+ */
+
+/* kr_revision bits */
+#define INFINIPATH_R_CHIPREVMINOR_MASK 0xFF
+#define INFINIPATH_R_CHIPREVMINOR_SHIFT 0
+#define INFINIPATH_R_CHIPREVMAJOR_MASK 0xFF
+#define INFINIPATH_R_CHIPREVMAJOR_SHIFT 8
+#define INFINIPATH_R_ARCH_MASK 0xFF
+#define INFINIPATH_R_ARCH_SHIFT 16
+#define INFINIPATH_R_SOFTWARE_MASK 0xFF
+#define INFINIPATH_R_SOFTWARE_SHIFT 24
+#define INFINIPATH_R_BOARDID_MASK 0xFF
+#define INFINIPATH_R_BOARDID_SHIFT 32
+
+/* kr_control bits */
+#define INFINIPATH_C_FREEZEMODE 0x00000002
+#define INFINIPATH_C_LINKENABLE 0x00000004
+#define INFINIPATH_C_RESET 0x00000001
+
+/* kr_sendctrl bits */
+#define INFINIPATH_S_DISARMPIOBUF_SHIFT 16
+
+#define IPATH_S_ABORT 0
+#define IPATH_S_PIOINTBUFAVAIL 1
+#define IPATH_S_PIOBUFAVAILUPD 2
+#define IPATH_S_PIOENABLE 3
+#define IPATH_S_DISARM 31
```

[PATCH 2 of 18] ipath – core driver header files

```
+
+#define INFINIPATH_S_ABORT (1U << IPATH_S_ABORT)
+#define INFINIPATH_S_POINTBUFAVAIL (1U << IPATH_S_POINTBUFAVAIL)
+#define INFINIPATH_S_POIOBUFAVAILUPD (1U << IPATH_S_POIOBUFAVAILUPD)
+#define INFINIPATH_S_PIOENABLE (1U << IPATH_S_PIOENABLE)
+#define INFINIPATH_S_DISARM (1U << IPATH_S_DISARM)
+
+/* kr_rcvctrl bits */
+#define INFINIPATH_R_PORTENABLE_SHIFT 0
+#define INFINIPATH_R_INTRAVAIL_SHIFT 16
+#define INFINIPATH_R_TAILUPD 0x80000000
+
+/* kr_intstatus, kr_intclear, kr_intmask bits */
+#define INFINIPATH_I_RCVURG_SHIFT 0
+#define INFINIPATH_I_RCVAVAIL_SHIFT 12
+#define INFINIPATH_I_ERROR 0x80000000
+#define INFINIPATH_I_SPIOSENT 0x40000000
+#define INFINIPATH_I_SPOIOBUFAVAIL 0x20000000
+#define INFINIPATH_I_GPIO 0x10000000
+
+/* kr_errorstatus, kr_errorclear, kr_errormask bits */
+#define INFINIPATH_E_RFORMATERR 0x0000000000000001ULL
+#define INFINIPATH_E_RVCRC 0x0000000000000002ULL
+#define INFINIPATH_E_RICRC 0x0000000000000004ULL
+#define INFINIPATH_E_RMINPKTLEN 0x0000000000000008ULL
+#define INFINIPATH_E_RMAXPKTLEN 0x0000000000000010ULL
+#define INFINIPATH_E_RLONGPKTLEN 0x0000000000000020ULL
+#define INFINIPATH_E_RSHORTPKTLEN 0x0000000000000040ULL
+#define INFINIPATH_E_RUNEXPCHAR 0x0000000000000080ULL
+#define INFINIPATH_E_RUNSUPVL 0x0000000000000100ULL
+#define INFINIPATH_E_REBP 0x0000000000000200ULL
+#define INFINIPATH_E_RIBFLOW 0x0000000000000400ULL
+#define INFINIPATH_E_RBADVERSION 0x0000000000000800ULL
+#define INFINIPATH_E_RRCVEGRFULL 0x0000000000001000ULL
+#define INFINIPATH_E_RRCVHDRFULL 0x0000000000002000ULL
+#define INFINIPATH_E_RBADTID 0x0000000000004000ULL
+#define INFINIPATH_E_RHDRLLEN 0x0000000000008000ULL
+#define INFINIPATH_E_RHDR 0x0000000000010000ULL
+#define INFINIPATH_E_RIBLOSTLINK 0x0000000000020000ULL
+#define INFINIPATH_E_SMINPKTLEN 0x0000000020000000ULL
+#define INFINIPATH_E_SMAXPKTLEN 0x0000000040000000ULL
+#define INFINIPATH_E_SUNDERRUN 0x0000000080000000ULL
+#define INFINIPATH_E_SPKTLEN 0x0000000100000000ULL
+#define INFINIPATH_E_SDRAPPEDSMPPKT 0x0000000200000000ULL
+#define INFINIPATH_E_SDRAPPEDDATAPKT 0x0000000400000000ULL
+#define INFINIPATH_E_SPIOARMLAUNCH 0x0000000800000000ULL
+#define INFINIPATH_E_SUNEXPERRPKTNUM 0x0000001000000000ULL
+#define INFINIPATH_E_SUNSUPVL 0x0000002000000000ULL
+#define INFINIPATH_E_IBSTATUSCHANGED 0x0001000000000000ULL
+#define INFINIPATH_E_INVALIDADDR 0x0002000000000000ULL
+#define INFINIPATH_E_
```