NAME

bxe - QLogic NetXtreme II Ethernet 10Gb PCIe adapter driver

SYNOPSIS

To compile this driver into the kernel, place the following lines in your kernel configuration file:

```
device bxe
```

Alternatively, to load the driver as a module at boot time, place the following line in loader.conf(5):

```
if_bxe_load="YES"
```

DESCRIPTION

The **bxe** driver provides support for PCIe 10Gb Ethernet adapters based on the QLogic NetXtreme II family of 10Gb chips. The driver supports Jumbo Frames, VLAN tagging, checksum offload (IPv4, TCP, UDP, IPv6-TCP, IPv6-UDP), MSI-X interrupts, TCP Segmentation Offload (TSO), Large Receive Offload (LRO), and Receive Side Scaling (RSS).

HARDWARE

The **bxe** driver provides support for various NICs based on the QLogic NetXtreme II family of 10Gb Ethernet controller chips, including the following:

- QLogic NetXtreme II BCM57710 10Gb
- QLogic NetXtreme II BCM57711 10Gb
- QLogic NetXtreme II BCM57711E 10Gb
- QLogic NetXtreme II BCM57712 10Gb
- QLogic NetXtreme II BCM57712-MF 10Gb
- QLogic NetXtreme II BCM57800 10Gb
- QLogic NetXtreme II BCM57800-MF 10Gb
- QLogic NetXtreme II BCM57810 10Gb
- QLogic NetXtreme II BCM57810-MF 10Gb
- QLogic NetXtreme II BCM57840 10Gb / 20Gb
- QLogic NetXtreme II BCM57840-MF 10Gb

CONFIGURATION

There a number of configuration parameters that can be set to tweak the driver’s behavior. These parameters can be set via the loader.conf(5) file to take effect during the next system boot. The following parameters affect ALL instances of the driver.

hw.bxe.debug

DEFAULT = 0
Sets the default logging level of the driver. See the Diagnostics and Debugging section below for more details.

hw.bxe.interrupt_mode
DEFAULT = 2
Sets the default interrupt mode: 0=IRQ, 1=MSI, 2=MSIX. If set to MSIX and allocation fails, the driver will roll back and attempt MSI allocation. If MSI allocation fails, the driver will roll back and attempt fixed level IRQ allocation. If IRQ allocation fails, then the driver load fails. With MSI/MSIX, the driver attempts to allocate a vector for each queue in addition to one more for default processing.

hw.bxe.queue_count
DEFAULT = 4
Sets the default number of fast path packet processing queues. Note that one MSI/MSIX interrupt vector is allocated per-queue.

hw.bxe.max_rx_bufs
DEFAULT = 0
Sets the maximum number of receive buffers to allocate per-queue. Zero(0) means to allocate a receive buffer for every buffer descriptor. By default this equates to 4080 buffers per-queue which is the maximum value for this config parameter.

hw.bxe.hc_rx_ticks
DEFAULT = 25
Sets the number of ticks for host interrupt coalescing in the receive path.

hw.bxe.hc_tx_ticks
DEFAULT = 50
Sets the number of ticks for host interrupt coalescing in the transmit path.

hw.bxe.rx_budget
DEFAULT = 0xffffffff
Sets the maximum number of receive packets to process in an interrupt. If the budget is reached then the remaining/pending packets will be processed in a scheduled taskqueue.

hw.bxe.max_aggregation_size
DEFAULT = 32768
Sets the maximum LRO aggregration byte size. The higher the value the more packets the hardware will aggregate. Maximum is 65K.

hw.bxe.mrrs
DEFAULT = -1
Sets the PCI MRRS: -1=Auto, 0=128B, 1=256B, 2=512B, 3=1KB

hw.bxe.autogreeen
DEFAULT = 0
Set AutoGrEEEN: 0=HW_DEFAULT, 1=FORCE_ON, 2=FORCE_OFF

hw.bxe.udp_rss
DEFAULT = 0
Enable/Disable 4-tuple RSS for UDP: 0=DISABLED, 1=ENABLED

Special care must be taken when modifying the number of queues and receive buffers.
FreeBSD imposes a limit on the maximum number of mbuf(9) allocations. If buffer allocations fail, the interface initialization will fail and the interface will not be usable. The driver does not make a best effort for buffer allocations. It is an all or nothing effort.

You can tweak the mbuf(9) allocation limit using sysctl(8) and view the current usage with netstat(1) as follows:

```
# netstat -m
# sysctl kern.ipc.nmbclusters
# sysctl kern.ipc.nmbclusters=<#>
```

There are additional configuration parameters that can be set on a per-instance basis to dynamically override the default configuration. The '#' below must be replaced with the driver instance / interface unit number:

dev.bxe.#.debug
DEFAULT = 0
Sets the default logging level of the driver instance. See *hw.bxe.debug* above and the Diagnostics and Debugging section below for more details.

dev.bxe.#.rx_budget
DEFAULT = 0xffffffff
Sets the maximum number of receive packets to process in an interrupt for the driver instance. See *hw.bxe.rx_budget* above for more details.

Additional items can be configured using ifconfig(8):

MTU - Maximum Transmission Unit
DEFAULT = 1500
RANGE = 46-9184
# ifconfig bxe# mtu <n>

Promiscuous Mode
DEFAULT = OFF
# ifconfig bxe# [promisc | -promisc]

Rx/Tx Checksum Offload
DEFAULT = RX/TX CSUM ON
Note that the Rx and Tx settings are not independent.
ifconfig bxe# [rxcsum | -rxcsum | txcsum | -txcsum]

TSO - TCP Segmentation Offload
DEFAULT = ON
# ifconfig bxe# [tso | -tso | tso6 | -tso6]

LRO - TCP Large Receive Offload
DEFAULT = ON
# ifconfig bxe# [lro | -lro]

DIAGNOSTICS AND DEBUGGING

There are many statistics exposed by **bxe** via sysctl(8).

To dump the default driver configuration:

```
# sysctl -a | grep hw.bxe
```

To dump every instance’s configuration and detailed statistics:

```
# sysctl -a | grep dev.bxe
```

To dump information for a single instance (replace the ’#’ with the driver instance / interface unit number):

```
# sysctl -a | grep dev.bxe.#
```

To dump information for all the queues of a single instance:

```
# sysctl -a | grep dev.bxe.#.queue
```

To dump information for a single queue of a single instance (replace the additional '#' with the queue number):

```
# sysctl -a | grep dev.bxe.#.queue.#
```

The **bxe** driver has the ability to dump a ton of debug messages to the system log. The default level of logging can be set with the *hw.bxe.debug* sysctl(8). Take care with this setting as it can result in too many logs being dumped. Since this parameter is the default one, it affects every instance and will dramatically change the timing in the driver. A better alternative to aid in debugging is to dynamically change the debug level of a specific instance with the *dev.bxe.#.debug* sysctl(8). This allows you to turn on/off logging of various debug groups on-the-fly.

The different debug groups that can be toggled are:

```
DBG_LOAD   0x00000001 /* load and unload    */
DBG_INTR   0x00000002 /* interrupt handling */
DBG_SP     0x00000004 /* slowpath handling  */
DBG_STATS  0x00000008 /* stats updates      */
DBG_TX     0x00000010 /* packet transmit    */
DBG_RX     0x00000020 /* packet receive     */
DBG_PHY    0x00000040 /* phy/link handling  */
DBG_IOCTL  0x00000080 /* ioctl handling     */
DBG_MBUF   0x00000100 /* dumping mbuf info  */
DBG_REGS   0x00000200 /* register access    */
DBG_LRO    0x00000400 /* lro processing     */
DBG_ASSERT 0x80000000 /* debug assert       */
DBG_ALL    0xFFFFFFFF /* flying monkeys     */
```

For example, to debug an issue in the receive path on bxe0:

```
# sysctl dev.bxe.0.debug=0x22
```

When finished turn the logging back off:

```
# sysctl dev.bxe.0.debug=0
```

SUPPORT

For support questions please contact your QLogic approved reseller or QLogic Technical Support at

http://support.qlogic.com, or by E-mail at <*support@qlogic.com*>.

SEE ALSO

netstat(1), altq(4), arp(4), netintro(4), ng_ether(4), vlan(4), ifconfig(8)

HISTORY

The **bxe** device driver first appeared in FreeBSD 9.0.

AUTHORS

The **bxe** driver was written by Eric Davis <*edavis@broadcom.com*>,
David Christensen <*davidch@broadcom.com*>, and
Gary Zambrano <*zambrano@broadcom.com*>.