1. Device installation

Some motherboards have certain PCI Express slots connected directly to the processor and others connected to the chipset. For optimal performance, install the ExaNIC in a slot directly connected to the processor.

Install SFP+ modules into the ExaNIC as required. There are no limitations on the type or manufacturer of SFP+ modules that can be used; we supply and recommend Finisar modules.

2. Software installation

There are a number of different ways of installing the ExaNIC software: using the pre-built packages provided by Exablaze, building your own packages, or installing from source.

Pre-built packages

Pre-built packages are available for some Linux distributions. The easiest way to access these is to add the Exablaze repository to your yum or apt sources, or you can manually download the desired packages directly from the repository.

The yum repository containing .rpm packages for Redhat/CentOS is located at:

http://exablaze.com/downloads/yum/

The apt repository containing .deb packages for Debian/Ubuntu is located at:

http://exablaze.com/downloads/apt/

In both cases there is a README.txt file which explains how to add the repository, what distributions are currently supported and what packages are available.

Build your own packages

You can also build your own packages using the packaging provided by Exablaze.

To build your own .rpm packages, download the latest source rpm from http://exablaze.com/downloads/yum/redhat/el6/SRPMS/ and run `rpmbuild --rebuild exanic-*.src.rpm` . Packages will be output to `~/rpmbuild/RPMS`.

To build your own .deb packages, download and unpack the source from the support website as per the first step below, then run `dpkg-buildpackage` from within the source directory. Packages will be output to the parent directory.

Install from source

If binary packages are not available for your distribution, the ExaNIC software is also easy to build and install from source. To start, download the ExaNIC source package from the support website and unpack it.

```
$ tar xzf exanic-1.8.0.tar.gz
$ cd exanic-1.8.0
```

Running the following commands should build and install the driver, library and utilities:

```
$ make
$ sudo make install
```

By default, the library will be installed to `/usr/local/lib/libexanic.a`, include files to `/usr/local/include/exanic` and utilities to `/usr/local/bin/exanic-*`. (You can add a `PREFIX=` argument to the `make install` command to use a prefix other than `/usr/local`.) Additionally the driver (`exanic.ko`) will be installed into `/lib/modules/`uname -r`/extra`.

If the driver fails to build, ensure that you have the kernel headers package for your kernel installed (this is typically named `kernel-devel` [RedHat/CentOS] or `linux-headers` [Debian/Ubuntu]).

3. After installation

Assuming installation completed successfully, load the driver and verify that the `exanic-config` utility works:

```
$ sudo modprobe exanic
$ sudo exanic-config exanic0
```

If either of these commands fail, run `dmesg` and check for errors.

The `exanic-config` output will show the Linux interfaces corresponding to each port of the ExaNIC. You can verify that packets are being received by bringing up an interface and running `tcpdump`. For example:

```
$ sudo ifconfig eth7 up
$ sudo tcpdump -n -i eth7
```

If desired you can also build and run the benchmarking utilities (see section 12).

4. Configuration

Since the ExaNIC appears to Linux as a normal network card, most configuration can be performed through the normal Linux mechanisms. For example, in Redhat-based distributions, configuration is generally performed using files in the `/etc/sysconfig/network-scripts` directory. Alternatively, the interface can be configured temporarily by issuing a command such as `ifconfig eth7 192.168.1.100`.

If the interface is being used through the low-level userspace API (libexanic) only, it is sufficient to bring the interface up without assigning an IP address. This can be done through either `ifconfig` (e.g. `ifconfig eth7 up`) or `exanic-config` (e.g. `exanic-config exanic0:0 up`).

5. The exanic-config utility

The exanic-config utility can be used to inspect diagnostic information about the card and SFP modules. For example:

```
$ exanic-config
Device exanic0:
 Hardware type: ExaNIC X4
 Board ID: 0x02
 Temperature: 53.2 C   VCCint: 1.00 V   VCCaux: 1.81 V
 Fan speed: 6987 RPM
 Function: network interface
 Firmware date: Thu Nov 28 20:54:56 2013
 Bridging: off
 Port 0:
   Interface: eth7
   Port speed: 10000 Mbps
    Port status: enabled, SFP present, signal detected, link active
    Mirroring: off
    Promiscuous mode: off
    Bypass-only mode: off
 MAC address: 64:3f:5f:01:13:18
 RX packets: 6  ignored: 0  error: 0
 TX packets: 6
 ...

$ exanic-config exanic0:0 sfp status
Device exanic0 port 0 SFP status:
 Vendor: FINISAR CORP.    PN: FTLX8571D3BCL     rev: A
                          SN: ALF0QE7          date: 111006
 Wavelength: 850 nm
 Nominal bit rate: 10300 Mbps
 Rx power: -2.5 dBm (0.56 mW)
```

```
Tx power: -1.9 dBm (0.65 mW)
Temperature: 33.8 C
```

The `exanic-config` utility will either accept the Linux interface name (e.g. `eth7`) or the device name of the ExaNIC device (e.g. `exanic0` for the first ExaNIC in the system by PCI ID, and `exanic0:0` for the first port of that card).

6. Setting port speed

Normally the ExaNIC operates in 10 Gigabit Ethernet (only) mode. To set a port to a different speed, use the speed command to either `ethtool` or `exanic-config`:

```
$ ethtool -s eth7 speed 1000
```

Or:

```
$ exanic-config exanic0:0 speed 1000
```

Supported values are 10000, 1000 and 100.

1 Gigabit Ethernet support requires firmware dated 20140206 or later. 100 Mbit Ethernet support requires firmware version 20150327 or later, and one of the following SFP modules: Finisar FCLF-8520-3/FCLF-8521-3/FCMJ-8520-3/FCMJ-8521-3 or Avago ABCU57xxxxxZ. For compatibility with other SFP modules, inquire with Exablaze.

7. Port mirroring and bridging

To configure card-specific functionality such as port mirroring and bridging functionality, you can use either `ethtool` or `exanic-config`. With ethtool, use `ethtool --show-priv-flags` and `ethtool --set-priv-flags`. For example:

```
$ sudo ethtool --show-priv-flags eth7
Private flags for eth7:
bypass_only: off
mirror_rx: off
mirror_tx: off
bridging: off
```

```
$ sudo ethtool --set-priv-flags eth7 bridging on
```

The same settings can also be configured through `exanic-config`:

```
$ sudo exanic-config exanic0 bridging on
exanic0: bridging on (ports 0 and 1)
```

```
$ sudo exanic-config exanic0:0 mirror-rx on
exanic0:0: RX mirroring on
```

Enabling receive port mirroring (`mirror_rx`) on a port causes the received data to be copied to port 3 (the last port; in this document the ports are numbered from 0). Enabling transmit port mirroring (`mirror_tx`) on a port causes the transmitted data to be copied to port 3. Enabling bridging (on either ports 0 or 1) causes any data received on port 0 that is not destined for the host to be forwarded to port 1, and vice versa. Port mirroring and bridging settings are stored in non-volatile memory and are persistent across reboot and power off.

Note that port mirroring and bridging are not currently supported on second generation cards (X10/X40). This may change in a future firmware update.

8. Bypass-only mode

If the interface is being used through the userspace API only and kernel CPU load on high traffic connections is a concern, the `bypass-only` setting in `exanic-config` (or `bypass_only` in `ethtool`) can be used to detach the kernel driver from the interface. This means that the kernel driver will not receive any packets. Tools such as `tcpdump` will not function correctly while this is enabled. Additionally, this setting is not currently persistent across reboots so, if required, should be added to post-up scripts for the interface.

Note that `bypass-only` mode should *not* be enabled for exasock to function correctly.

9. Setting permissions

If the ExaNIC is to be used in kernel bypass mode, it may be useful to set permissions on the ExaNIC device files (`/dev/exanic0`, etc, and the `/dev/exasock` device file used by ExaNIC Sockets) so that they can be accessed by certain non-root applications. Permissions can be set with `chown/chgrp/chmod` as normal, however this will not persist across reboots. A persistent configuration can be achieved by appropriate configuration of the `udev` daemon. For example, to force the device files to be accessible by a group called `exanic`, create a `/etc/udev/rules.d/exanic.rules` file as follows:

```
KERNEL=="exanic*", GROUP="exanic", MODE="0660"
KERNEL=="exasock", GROUP="exanic", MODE="0660"
```

In the current software release the ExaNIC device files also expose the device control registers so only trusted users should be given access to the ExaNIC in this way.

10. Clock synchronization

ExaNICs support hardware timestamping. To use the ExaNIC for timestamping, it is necessary to synchronize the card's hardware clock. This can be done in two different ways: using PTP or using the `exanic-clock-sync` daemon.

10.1 PTP

The Precision Time Protocol (PTP) is a protocol that allows for very accurate time synchronization between a master and a number of client (slave) systems. The ExaNIC supports the Linux SO_TIMESTAMPING API, so standard PTP daemons such as ptp4l (LinuxPTP) or ptpd can be used to synchronize the card's clock. For example, ptp4l can be run with a command such as the following:

```
$ ptp4l -i eth4
```

Note that at the time of writing, the current release of ptpd (2.3.1) does not contain the required hardware timestamping code, so you need to either use the hardware timestamping branch described at http://nwtime.org/new-features-for-ptpd/ or to additionally run `exanic-clock-sync` to synchronize the time from the host (see below). LinuxPTP works out of the box.

Note also that the underlying time standard used by PTP is the International Atomic Time (TAI) standard, which at the time of writing is 36 seconds ahead of the Universal Coordinated Time (UTC) standard. LinuxPTP will set and keep the ExaNIC hardware clock in TAI time, not UTC time. Applications that make use of ExaNIC timestamps will need to be aware of this. Recent kernels are aware of the UTC-TAI offset, and this can be queried using `adjtimex()`.

10.2 Host or PPS or second ExaNIC (exanic-clock-sync)

If PTP is not being used, or to synchronize additional ExaNIC cards to a primary, the `exanic-clock-sync` daemon can be used to keep the ExaNIC card's clock synchronized. Each argument to `exanic-clock-sync` specifies a `<device>:<synchronization-source>` pair, with the synchronization source being one of `host` (host clock), `pps` (pulse-per-second input) or `exanicN` (another ExaNIC device). For example:

```
$ exanic-clock-sync exanic0:host exanic1:exanic0
```

If desired this can be placed in system startup scripts. A sample `init.d` script is provided in `configs/exanic-clock-sync`.

By default, the SMA input on ExaNIC cards has standard 50 ohm termination. To turn termination off, specify `pps-no-term` instead of `pps`. This allows for daisy chaining the PPS inputs of several ExaNIC cards together. The intermediate cards should be configured with `pps-no-term` and the final card with `pps`.

11. Pulse-per-second output

ExaNICs with a SMA connector (ExaNIC X10 and onwards) can be configured to drive a PPS out using the following command:

```
$ exanic-config exanic0 pps-out on
```

12. Running benchmarking utilities

A number of benchmarking utilities, both for the ExaNIC and for other cards, are located in the perf-test directory provided with the distribution. To build the benchmarks for ExaNIC:

```
$ cd perf-test
$ make exanic
```

The simplest benchmark to run is `exanic_loopback`, which sends packets out one port and receives them on another. Connect a cable from port 0 to port 1 of the device, bring the respective interfaces up, and run, for example (to obtain 100 samples at 64 byte frame size):

```
$ ./exanic_loopback exanic0 0 1 64 100
```

13. Updating firmware

The firmware version currently running on the ExaNIC card is shown under 'Firmware date' in `exanic-config`. If there is a newer version available on the website, you can update your firmware as follows. First, uncompress the downloaded firmware image:

```
$ gunzip exanic_x4_20131128.fw.gz
```

Now run `exanic-fwupdate`:

```
$ exanic-fwupdate exanic_x4_20131128.fw
```

Note that there are different firmware files for each type of ExaNIC (X4, X10, etc.). The utility will prevent you from installing firmware incompatible with your hardware.

14. Uninstallation

If you installed from packages then you should use the appropriate yum or apt command (`sudo yum remove 'exanic*'` or `sudo apt-get remove 'exanic*'`).

If you installed from source the following command should remove everything that was installed by `make install`:

```
$ sudo make uninstall
```

Also remove any files that were manually created while following these instructions (`/etc/udev/rules.d/exanic.rules` and any scripts that run `exanic-clock-sync`).

15. Troubleshooting performance

1. Make sure the ExaNIC is plugged into a PCIe x8 Gen 3 slot and is running @ 8.0 GT/s per lane (for systems that support PCIe Gen3) . This can be identified by running the `lspci` command and looking at the `LnkSta` (link status) output.

```
$ sudo lspci -d 1ce4:* -vvv |grep LnkSta:
    LnkSta: Speed 8GT/s, Width x8, TrErr- Train- SlotClk+ DLActive-
BWMgmt- ABWMgmt-
```

2. Make sure the ExaNIC is plugged into a PCIe slot directly connected to a CPU. The server or motherboard documentation should indicate which slots are connected to CPUs and which are connected to the chipset. If unsure, the following procedure can be used. First determine the bus number of the ExaNIC from `lspci`:

```
$ sudo lspci -d 1ce4:*
02:00.0 Ethernet controller: Exablaze ExaNIC X10
```

In this case, the bus number is `02`. Now search for the device that has `secondary=02` in the output of `lspci -v`, for example:

```
$ sudo lspci -v
...
00:01.1 PCI bridge: Intel Corporation Xeon E3-1200 v2/3rd Gen Core
processor PCI Express Root Port (rev 09) (prog-if 00 [Normal decode])
    Flags: bus master, fast devsel, latency 0
    Bus: primary=00, secondary=02, subordinate=02, sec-latency=0

...
```

For optimal performance, this should be a processor root port (in this case, "Intel Corporation Xeon E3-1200 v2/3rd Gen Core processor PCI Express Root Port").

3. For initial latency testing, test only the ExaNIC card, try to remove external switches or other NICs from the test setup.

4. When testing software performance, pin the process to an isolated core, e.g.

```
$ sudo taskset -c 2 ./exanic_loopback exanic0 0 0 64 1000000
```

5. If hardware timestamps appear to be out by 36 seconds, this is likely a TAI/UTC mismatch. See the note in the Clock Synchronization using PTP section (Section 10.1).

FCC Compliance Notice

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation.

This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.