

Flexible Port Partitioning Configuration Guide

**Intel® Ethernet CNA X710 & XL710 on Red Hat*
Enterprise Linux 7***

Technical Brief

Networking Division (ND)

December 2014

Revision 1.0
331598-001



LEGAL

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors which may cause deviations from published specifications.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

* Other names and brands may be claimed as the property of others.

© 2014 Intel Corporation.

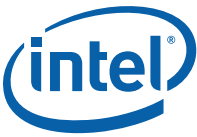


Revision History

Revision	Date	Comments
1.0	December 1, 2014	Initial release (Intel Public).



NOTE: This page intentionally left blank.



Contents

- 1.0 Introduction** 7
 - 1.1 Intel and the Ethernet 7
 - 1.2 Intel® Ethernet Controller XL710 8
 - 1.3 I/O Virtualization 9
 - 1.3.1 Hardware Requirements 9
 - 1.3.2 Software Requirements 9
- 2.0 Installation and Configuration** 9
 - 2.1 Server Setup..... 9
 - 2.2 Application Binding16
 - 2.3 Process Binding20
 - 2.4 Virtual Machine Binding21
- 3.0 Summary**25
- 4.0 Customer Support**26
- 5.0 Product Information**26



NOTE: This page intentionally left blank.



1.0 Introduction

In the past, systems administrators employed multiple 1 GbE network ports to segregate or partition various types of outbound network traffic from servers. 10 GbE networks are becoming the standard, and data centers have started deploying 40 GbE networks. System administrators are constantly challenged to manage these resources effectively, and need mechanisms to control, shape, and partition/segregate network traffic.

Intel Flexible Port Partitioning (FPP) uses SR-IOV technology to create Virtual Functions (VFs) on Intel 10 GbE and 40 GbE Converged Network Adapters. These VFs can be assigned to dedicated services, processes, and applications running on Linux servers. Systems administrators can also enforce Quality of Service (QoS) and transmit rate limits to further fine-tune network resources.

This document is a follow-on to [An Introduction to Intel Flexible Port Partitioning Using SR-IOV Technology](#) technical brief. This document shows how to partition an Ethernet network interface using Intel® Ethernet CNA X710 & XL710 Virtual Functions in Red Hat* Enterprise Linux* version 7.

1.1 Intel and the Ethernet

Since its inception in 1973, Intel has been vital to the development of the Ethernet, and continues to be the Industry leader. For over 40 years, the Ethernet has been growing to accommodate increasing bandwidth needs for complex multi-media; streaming video, music and voice data, for example. Beginning with One Gigabit Ethernet, expanding to 10 Gigabit, and now introducing 40 Gigabit Ethernet, computing and storage resource needs continue to grow.

Following is a brief history of Intel and the Ethernet:

- 1994: Intel ships the world's first 10/100 Mb/s Network Interface Card (NIC).
- 1997: Intel ships the first single-chip 10/100 Mb/s controller.
- 2001: Intel ships the first single-chip 10/100/1000 Mb/s controller.
- 2002: Intel ships the first XPAK Multimode Optical Transceiver, delivering 10-Gigabit Ethernet (GbE) and 10-Gigabit Fibre Channel transport for storage systems at half the cost, a third less power consumption, and a third of the size of earlier solutions.
- 2003: Intel ships the world's first 10 Gigabit Ethernet NIC.
- 2006: Intel introduces the first low-profile quad-port Ethernet NIC. By incorporating 4-Gigabit Ethernet connections in a low-profile PCI Express slot, it improved server throughput and rack density at the same time.
- 2007: Intel releases first "initiator" source code to enable Linux implementations of Fibre Channel over Ethernet (FCoE). By allowing fiber channel SAN traffic to run over Gigabit Ethernet networks, FCoE enables consolidation of storage area network (SAN) and LAN traffic, simplifying network infrastructure in data centers.



1.2 Intel® Ethernet Controller XL710

The 40 Gigabit XL710 Controller is designed for flexibility, with configurable port speeds of up to 2 x 40 GbE, or 4 x 10 GbE, ensuring a smooth transition to 40 GbE. It also provides a 222% increase in Gigabits per Watt in adapter power for approximately half the power cost when compared to using two previous generation dual-port adapters.

The XL710 offers the following features:

- 10/40 GbE Controller (Dual and Single 40 GbE, Quad and Dual 10 GbE configurations).
- PCI Express* (PCIe) 3.0, x8 including Direct I/O optimizations via TLP Processing Hints (TPH).
- Intelligent Off-load to enable high-performance with Intel® Xeon® servers.
- Network Virtualization off-loads including VXLAN and NVGRE.
- Industry-leading I/O virtualization innovations and performance with broad hypervisor and standards support.
- Intel® Ethernet Flow Director (for hardware application traffic steering).
- Excellent small packet performance for network appliances and NFV.
- Intel® Data Plane Developer Kit Optimize.
- Unified Networking providing a single wire for LAN and storage: NAS(SMB,NFS) and SAN (iSCSI, FCoE).

The following are the Intel 40 Gigabit XL710 Controller-based Dual and Quad Adapter offerings:

Note: These boards do NOT ship with optics installed. Optics must be purchased separately.

- Intel® Ethernet Converged Network Adapter X710-DA4
 - X710DA4FH, XL710DA4FHBLK (Retail, Quad Port Full Height)
 - X710DA4FHG1P5 (OEM Gen, Quad Port Full Height)
 - X710DA4G1P5 (OEM Gen, Quad Port Low Profile)
- Intel® Ethernet Converged Network Adapter X710-DA2
 - X710DA2, XL710DA2BLK (Retail, Dual Port)
 - X710DA2G1P5 (OEM Gen, Dual Port)
- Intel® Ethernet Converged Network Adapter XL710-QDA2
 - XL710QDA2, XL710QDA2BLK (Retail, Dual Port)
 - XL710QDA2G1P5 (OEM Gen, Dual Port)
- Intel® Ethernet Converged Network Adapter XL710-QDA1
 - XL710QDA1, XL710QDA1BLK (Retail, Single Port)
 - XL710QDA1G1P5 (OEM Gen, Single Port)

Power efficiency is critical to IT specialists as energy consumption is a real concern in data center operations. The Intel Ethernet Controller provides a low-power interface to eliminate the need for additional power. It also offers the manageability IT personnel require for remote control and alerting.

This controller provides multiple interface options, a smaller footprint for reduced infrastructure and cabling costs, lower power consumption, and intelligent off-loads that do not require disabling key features and flow direction to balance high volume traffic flows.



1.3 I/O Virtualization

The Intel® Ethernet Server Adapter X710 & XL710 family of adapters delivers numerous industry-leading features that are helping data center administrators implement innovative solutions for difficult and challenging connectivity problems. I/O Virtualization is one of the fastest growing usage models within the data center.

The X710 & XL710 family of adapters provides the ability to create Virtual Functions (VFs) that are identical instantiations of the Physical Functions (PFs). VFs are capable of providing up to 10 GbE or 40 GbE connectivity to Virtual Machines (VMs) within a virtualized operating system framework. The Intel® Ethernet Server Adapter X710 (Quad Port) supports up to 32 VFs per port, for a total of up to 128 VFs per adapter. The Intel® Ethernet Server Adapter XL710 (Dual Port) supports up to 64 VFs per port, for a total of up to 128 VFs per adapter.

1.3.1 Hardware Requirements

- An Intel® Ethernet Converged Network Adapter X710 or XL710 (codename Fortville).
- A server platform that supports Intel® Virtualization Technology for Directed I/O (VT-d) and the PCI-SIG* Single Root I/O Virtualization and Sharing (SR-IOV) specification.
- A server platform with an available PCI Express*: x8 5.0Gb/s (Gen2) or x8 8.0Gb/s (Gen3) slot.

1.3.2 Software Requirements

- Red Hat Enterprise Linux Version 7.0.
- Intel® Ethernet Converged Network Adapter X710 or XL710 Linux Drivers for PF and VF (<http://sourceforge.net/projects/e1000/files/>).

2.0 Installation and Configuration

2.1 Server Setup

This section shows various setup and configuration steps for enabling SR-IOV on the X710 or XL710 server adapters.

1. Install the X710 or XL710 server adapter in an available PCI-Express x8 slot. (Ensure that the x8 slot is electrically connected as x8, some slots are physically x8 but electrically support only x4. Verify this with your server manufacturer or system documentation.)
2. Power up the server.
3. Enter the server's BIOS setup and make sure the virtualization technology and Intel® VT-d features are enabled.
4. Install Red Hat Enterprise Linux 7.0 on the server.
5. Make sure all Linux KVM modules, libraries, user tools, and utilities have been installed during the operation system installation.
6. The Red Hat Enterprise Linux installation process may require a server reboot upon successful operating system install.
7. Log in to the newly-installed Red Hat Enterprise Linux operating system and use **sudo** to run commands in superuser or root mode.

8. I/O Memory Management Unit (IOMMU) support is not enabled by default in Red Hat Enterprise Linux 7.0 distribution. IOMMU support is required for a VF to function properly when assigned to a VM. The following kernel boot parameter is required to enable IOMMU support for Linux kernels:

```
intel_iommu=on
```

This parameter can be appended to the `GRUB_CMDLINE_LINUX` entry in `/etc/default/grub` configuration file, as shown in [Figure 1](#).



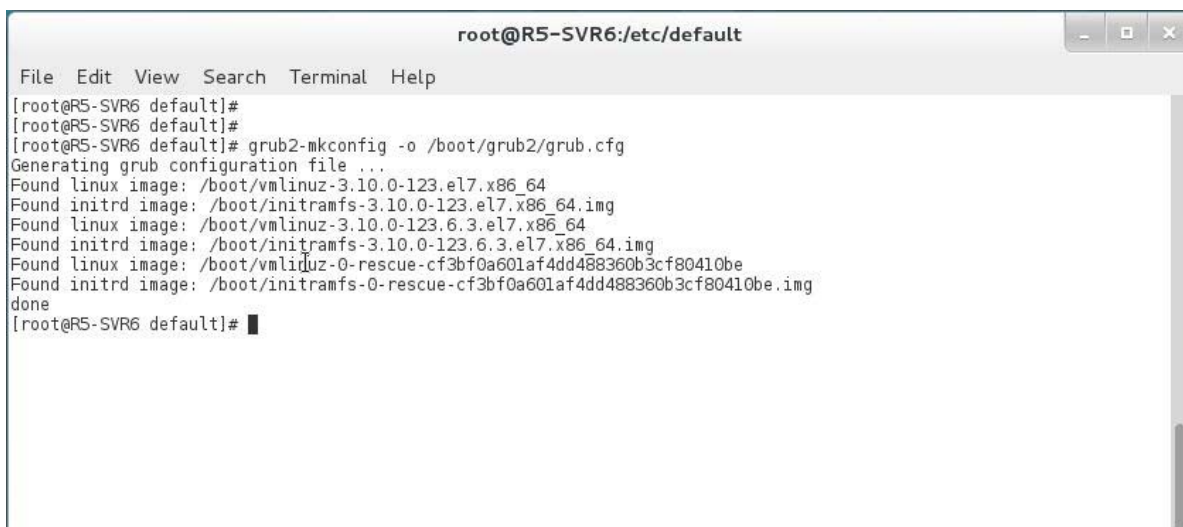
```

root@R5-SVR6:/etc/default
File Edit View Search Terminal Help
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=rhel_r5-svr6/root vconsole.font=latarcyrheb-sun16 vconsole.keymap=us
rd.lvm.lv=rhel_r5-svr6/swap rhgb quiet intel_iommu=on"
GRUB_DISABLE_RECOVERY="true"
~
~
~
~
~
~
~
~
~
~

```

Figure 1. GRUB Configuration File

9. Update grub configuration using the **grub2-mkconfig** command, as shown in [Figure 2](#).



```

root@R5-SVR6:/etc/default
File Edit View Search Terminal Help
[root@R5-SVR6 default]#
[root@R5-SVR6 default]#
[root@R5-SVR6 default]# grub2-mkconfig -o /boot/grub2/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.10.0-123.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-123.el7.x86_64.img
Found linux image: /boot/vmlinuz-3.10.0-123.6.3.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-123.6.3.el7.x86_64.img
Found linux image: /boot/vmlinuz-0-rescue-cf3bf0a601af4dd488360b3cf80410be
Found initrd image: /boot/initramfs-0-rescue-cf3bf0a601af4dd488360b3cf80410be.img
done
[root@R5-SVR6 default]#

```

Figure 2. GRUB Boot Loader Update Process Output



10. Reboot the server for the `iommu` change to take effect.
11. PF and VF drivers for the X710 and XL710 server adapters are included in Red Hat Enterprise Linux 7.0 distribution and are named as `i40e` and `i40evf` respectively. Newer versions of these drivers are available at Intel's Open Source Linux driver site. Using latest available drivers is strongly recommended.
12. The Red Hat Enterprise Linux 7.0 installation does not create VF by default. The X710 server adapter supports up to 32 VFs per port. The XL710 server adapter supports up to 64 VFs per port. There are two methods to create VFs depending on the Linux Kernel installed:
 - a. Linux Kernel version 3.7.x and below — VFs can be created by using the `i40e` driver load time parameter called `max_vfs`.

```
#modprobe i40e max_vfs=4,4
```

The example in [Figure 3](#) shows the creation of four VFs per port.

```
root@R7-SVR7:~  
File Edit View Search Terminal Help  
[root@R7-SVR7 ~]#  
[root@R7-SVR7 ~]#  
[root@R7-SVR7 ~]# modprobe i40e max_vfs=4,4  
[root@R7-SVR7 ~]#  
[root@R7-SVR7 ~]#  
[root@R7-SVR7 ~]#
```

Figure 3. i40e Driver Load Example

- b. Linux Kernel version 3.8.x and above — VF can be created by writing an appropriate value to the `sriov_numvfs` parameter via `sysfs` interface.

```
#echo 4 > /sys/class/net/<device_name>/device/sriov_numvfs
```

The example in [Figure 4](#) shows the creation of four VFs per port.

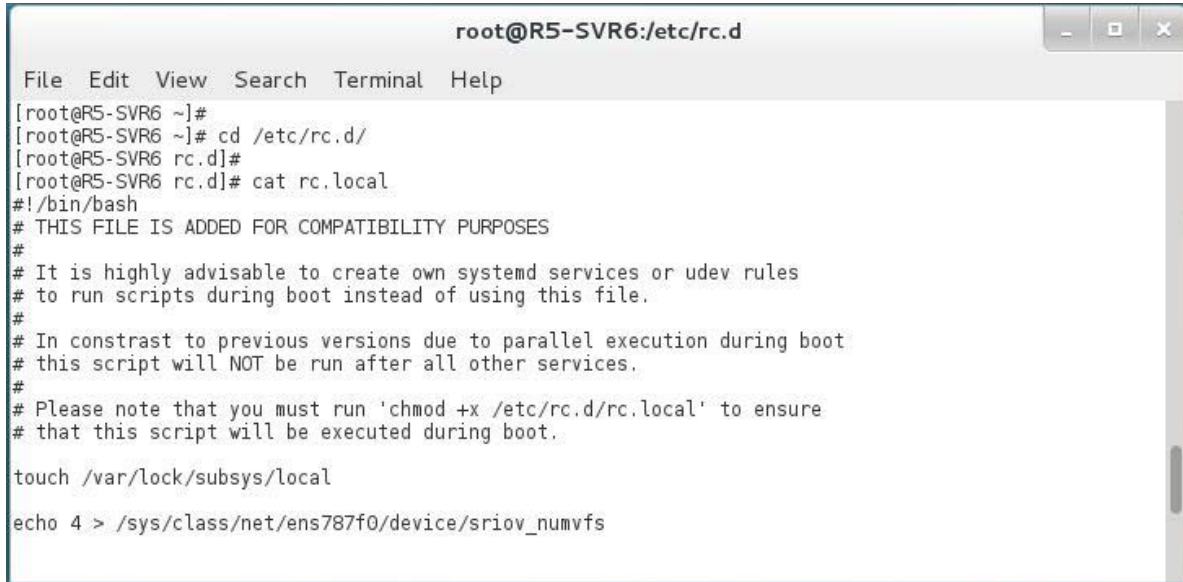
```
root@R5-SVR6:~  
File Edit View Search Terminal Help  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]# echo 4 > /sys/class/net/ens787f0/device/sriov_numvfs  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]# cat /sys/class/net/ens787f0/device/sriov_numvfs  
4  
[root@R5-SVR6 ~]#
```

Figure 4. VF Creation via SysFS

The example in [Figure 4](#) shows four VFs being created on device name `ens787f0`, the device name assigned to XL710 server adapter port 0 by the Linux operating system. Device name for the XL710 server adapter ports on your system may be different.

The second command in the example above queries the `sriov_numvfs` parameter to verify the four VFs are successfully created.

13. Module options are not persistent from one boot to the next. To ensure that the desired number of VFs are created each time the server is power-cycled, append the above command to the *rc.local* file, which is located in the */etc/rc.d/* directory. The Linux OS executes the *rc.local* script at the end of the boot process. The example in [Figure 5](#) shows contents of *rc.local* file.



```

root@R5-SVR6:/etc/rc.d
File Edit View Search Terminal Help
[root@R5-SVR6 ~]#
[root@R5-SVR6 ~]# cd /etc/rc.d/
[root@R5-SVR6 rc.d]#
[root@R5-SVR6 rc.d]# cat rc.local
#!/bin/bash
# THIS FILE IS ADDED FOR COMPATIBILITY PURPOSES
#
# It is highly advisable to create own systemd services or udev rules
# to run scripts during boot instead of using this file.
#
# In contrast to previous versions due to parallel execution during boot
# this script will NOT be run after all other services.
#
# Please note that you must run 'chmod +x /etc/rc.d/rc.local' to ensure
# that this script will be executed during boot.

touch /var/lock/subsys/local

echo 4 > /sys/class/net/ens787f0/device/sriov_numvfs
  
```

Figure 5. *rc.local* File Contents

Warning: Errors and informational messages during *i40e* driver load are logged in the */var/log/messages* file. It is good practice to review this file to confirm that the driver loaded successfully without warnings or errors.

14. On Linux Kernel version 3.8.x and above, VF can be destroyed or disabled by writing the value 0 to the *sriov_numvfs* parameter via *sysfs* interface.

```
#echo 0 > /sys/class/net/<device_name>/device/sriov_numvfs
```

The example in [Figure 6](#) shows disabling SR-IOV on a given port.



```

root@R5-SVR6:~
File Edit View Search Terminal Help
[3;J
[root@R5-SVR6 ~]#
[root@R5-SVR6 ~]# cat /sys/class/net/ens787f0/device/sriov_numvfs
4
[root@R5-SVR6 ~]#
[root@R5-SVR6 ~]# echo 0 > /sys/class/net/ens787f0/device/sriov_numvfs
[root@R5-SVR6 ~]#
[root@R5-SVR6 ~]# cat /sys/class/net/ens787f0/device/sriov_numvfs
0
[root@R5-SVR6 ~]# █
  
```

Figure 6. Disabling/Destroying VFs Example



15. On Linux Kernel version 3.8.x and above, the maximum number of VFs supported by the adapter can be queried by reading the `sriov_totalvfs` parameter via `sysfs` interface.

```
#cat /sys/class/net/<device_name>/device/sriov_totalvfs
```

The example in [Figure 7](#) shows the maximum number of VFs supported by a given port.

```
root@R5-SVR6:~  
File Edit View Search Terminal Help  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]# cat /sys/class/net/ens787f0/device/sriov_totalvfs  
64  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]#
```

Figure 7. Total VF Supported Query Output

16. Use the `lspci` command to confirm that the VF was successfully created.

```
#lspci | grep 'Ether'
```

The example in [Figure 8](#) shows the result of this command.

```
root@R7-SVR7:~  
File Edit View Search Terminal Help  
[root@R7-SVR7 ~]#  
[root@R7-SVR7 ~]# lspci | grep Ether  
02:00.0 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 01)  
02:00.1 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 01)  
02:02.0 Ethernet controller: Intel Corporation XL710/X710 Virtual Function (rev 01)  
02:02.1 Ethernet controller: Intel Corporation XL710/X710 Virtual Function (rev 01)  
02:02.2 Ethernet controller: Intel Corporation XL710/X710 Virtual Function (rev 01)  
02:02.3 Ethernet controller: Intel Corporation XL710/X710 Virtual Function (rev 01)  
02:0a.0 Ethernet controller: Intel Corporation XL710/X710 Virtual Function (rev 01)  
02:0a.1 Ethernet controller: Intel Corporation XL710/X710 Virtual Function (rev 01)  
02:0a.2 Ethernet controller: Intel Corporation XL710/X710 Virtual Function (rev 01)  
02:0a.3 Ethernet controller: Intel Corporation XL710/X710 Virtual Function (rev 01)  
04:00.0 Ethernet controller: Intel Corporation Ethernet Controller 10-Gigabit X540-AT2 (rev 01)  
04:00.1 Ethernet controller: Intel Corporation Ethernet Controller 10-Gigabit X540-AT2 (rev 01)  
81:00.0 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 01)  
81:00.1 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 01)  
[root@R7-SVR7 ~]#
```

Figure 8. lspci Output

[Figure 8](#) shows four VFs each for the physical port 0 and port 1 of the XL710 server adapter. Each VF is identified by a unique bus, device, and function number. In the example, the first VF is assigned Bus #81, Device #02, and Function #0.

VFs with low device number belong to PF 0, which is port 0. In the example above VF designated by 02:02:1 belongs to PF0. VFs with high device number belong to PF 1, which is port 1. VF designated by 02:0a:1 belongs to PF1.

17. Module options are not persistent from one boot to the next. On Linux Kernel version 3.7.x and below, create the *i40e.conf* file in the */etc/modprobe.d/* folder to ensure the user-defined number of VFs are created during server boot time, as shown in [Figure 9](#).



```

root@R7-SVR7:~
File Edit View Search Terminal Help
[root@R7-SVR7 ~]#
[root@R7-SVR7 ~]# cat /etc/modprobe.d/i40e.conf

options i40e max_vfs=4,4
[root@R7-SVR7 ~]#
[root@R7-SVR7 ~]#
[root@R7-SVR7 ~]#
[root@R7-SVR7 ~]#

```

Figure 9. *i40e.conf* Driver Configuration File Contents

Upon successful VF creation, the Linux operating system automatically loads the *i40vf* driver.

18. The VF driver automatically loads in the host operating system as soon as the VFs are created by the PF driver. The VF driver claims newly-created VFs, and these VFs are available for assignment to various services, processes or applications.
19. During the creation of a user-defined number of VFs, the *i40e* driver assigns MAC address 00:00:00:00:00:00 to each VF. An application such as LibVirt, VF driver, or Virtual Machine Manager assigns a valid MAC address to the VF before use. The Intel *i40e* driver has a built-in security feature that allows system administrators to assign a valid MAC address to a VF from within the host operating system. Make sure each VF is assigned a unique MAC address; duplicate MAC addresses cause loss of communication on the network. Use the following command to set a MAC address for each VF, if necessary.

```
#ip link set ens787f0 vf 0 mac aa:bb:cc:dd:ee:ff
```

20. Use the following command to confirm that the VF MAC address assignment was completed successfully.

```
#ip link show ens787f0
```

[Figure 10](#) shows an example of the results of this command.



```
root@R5-SVR6:~  
File Edit View Search Terminal Help  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]# ip link set ens787f0 vf 0 mac aa:bb:cc:dd:ee:ff  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]# ip link show ens787f0  
4: ens787f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT qlen 1000  
    link/ether 68:05:ca:26:8a:58 brd ff:ff:ff:ff:ff:ff  
        vf 0 MAC aa:bb:cc:dd:ee:ff, link-state auto  
        vf 1 MAC 00:00:00:00:00:00, link-state auto  
        vf 2 MAC 00:00:00:00:00:00, link-state auto  
        vf 3 MAC 00:00:00:00:00:00, link-state auto  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]#
```

Figure 10. VF MAC Address Query Result

21. To ensure each VF carries the same MAC address assignment from one boot to the next, the commands from the previous step can be appended to the *rc.local* file, which is located in the */etc/rc.d/* directory. The Linux OS executes the *rc.local* script at the end of the boot process, as shown in Figure 11.

```
root@R5-SVR6:/etc/rc.d  
File Edit View Search Terminal Help  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]#  
[root@R5-SVR6 ~]# cd /etc/rc.d  
[root@R5-SVR6 rc.d]#  
[root@R5-SVR6 rc.d]#  
[root@R5-SVR6 rc.d]# ls  
init.d rc0.d rc1.d rc2.d rc3.d rc4.d rc5.d rc6.d rc.local  
[root@R5-SVR6 rc.d]#  
[root@R5-SVR6 rc.d]# cat rc.local  
#!/bin/bash  
# THIS FILE IS ADDED FOR COMPATIBILITY PURPOSES  
#  
# It is highly advisable to create own systemd services or udev rules  
# to run scripts during boot instead of using this file.  
#  
# In contrast to previous versions due to parallel execution during boot  
# this script will NOT be run after all other services.  
#  
# Please note that you must run 'chmod +x /etc/rc.d/rc.local' to ensure  
# that this script will be executed during boot.  
  
touch /var/lock/subsys/local  
  
ip link set ens787f0 vf 0 mac aa:bb:cc:dd:ee:00  
ip link set ens787f0 vf 1 mac aa:bb:cc:dd:ee:01  
ip link set ens787f0 vf 2 mac aa:bb:cc:dd:ee:02  
ip link set ens787f0 vf 3 mac aa:bb:cc:dd:ee:03  
[root@R5-SVR6 rc.d]#
```

Figure 11. *rc.local* File Contents

2.2 Application Binding

In this example, the Apache web server application is bound to a specific network interface. The network interface is a channel bond created using Linux Channel Bonding driver in mode=1 (active-backup), also known as failover mode, which employs two VFs. Each VF is associated with a physical network port of the Intel® Ethernet Converged Network Adapter XL710, delivering up to 40 GbE Network connectivity per physical port. An example is shown in [Figure 12](#).

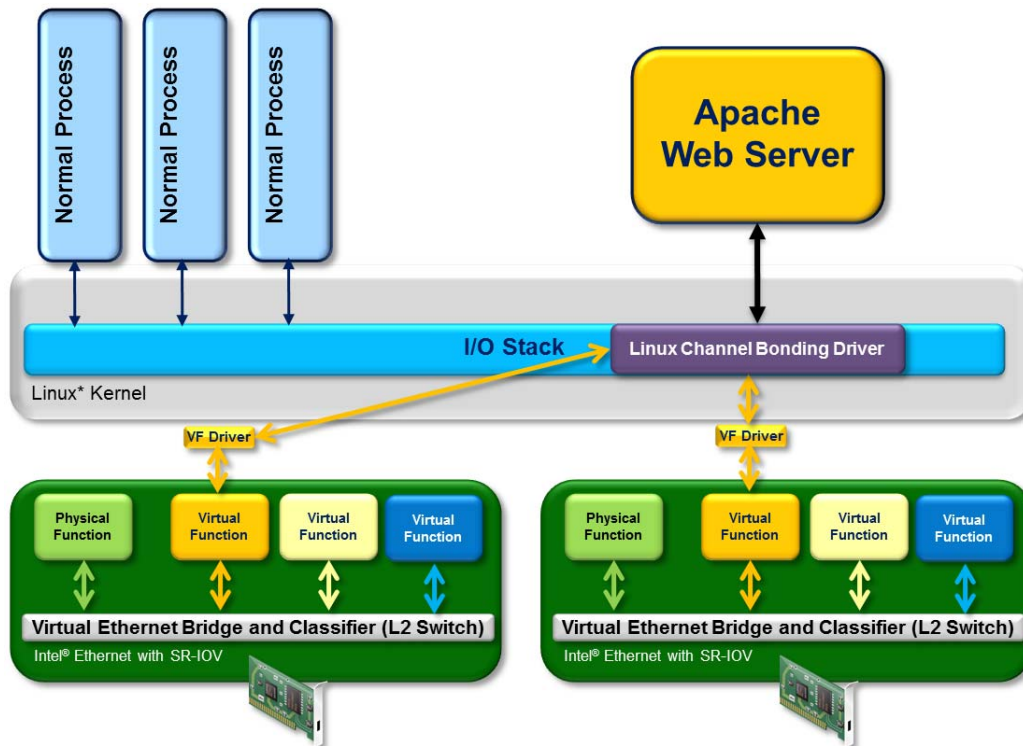


Figure 12. Application Binding Setup Diagram

1. Create VFs on the X710 or XL710 server adapter as prescribe in [Section 2.1, “Server Setup”](#).
 2. Execute the **modinfo bonding** command to verify Linux Bonding driver’s module and usage information for various supported Linux Channel Bonding driver module parameters.
 3. Execute the **modprobe --first-time bonding** command at Linux shell prompt. This command loads the Linux Channel Bonding driver.
 4. Execute the **lsmod bonding** command to confirm that the Linux Channel Bonding driver is loaded.
- Note:** Network Manager service may try to configure network interface automatically. Consider stopping and disabling Network Manager Service to avoid automatic network interface configuration.
5. Linux operating system loads the VF driver as soon as the VFs are created. Linux network stack also creates the network interfaces and assigns names. In this example, the Linux operating system has assigned `enp129s2` and `enp129s10` to the VFs.
 6. Create the `ifcfg-enp129s2` file for the VF in the `/etc/sysconfig/network-scripts/` directory. The contents of this file is shown in [Figure 13](#).



```
root@R5-SVR6:/etc/sysconfig/network-scripts
File Edit View Search Terminal Help

[root@R5-SVR6 network-scripts]#
[root@R5-SVR6 network-scripts]# pwd
/etc/sysconfig/network-scripts
[root@R5-SVR6 network-scripts]#
[root@R5-SVR6 network-scripts]# cat ifcfg-enp129s2
DEVICE="enp129s2"
TYPE=Ethernet
ONBOOT="yes"
BOOTPROTO="none"
USERCTL=no
MASTER=bond0
SLAVE=yes
[root@R5-SVR6 network-scripts]#
```

Figure 13. VF Interface Configuration File

The example file shows `enp129s2` network interface configuration. The network interface is configured as a Slave interface, and its master is `Bond0` interface.

7. Create the `ifcfg-enp129s10` file for the VF in the `/etc/sysconfig/network-scripts/` directory. The contents of this file is shown in [Figure 14](#).



```
root@R5-SVR6:/etc/sysconfig/network-scripts
File Edit View Search Terminal Help

[root@R5-SVR6 network-scripts]#
[root@R5-SVR6 network-scripts]# cat ifcfg-enp129s10
DEVICE="enp129s10"
TYPE=Ethernet
ONBOOT="yes"
BOOTPROTO="none"
USERCTL=no
MASTER=bond0
SLAVE=yes
[root@R5-SVR6 network-scripts]#
```

Figure 14. VF Interface Configuration File

The example file shows `enp129s2` network interface configuration. The network interface is configured as a Slave interface, and its master is `Bond0` interface.

8. Create the `ifcfg-bond0` file for the NIC team (bonding) in the `/etc/sysconfig/network-scripts/` directory. The contents of this file is shown in [Figure 15](#).

```

root@R5-SVR6:/etc/sysconfig/network-scripts
File Edit View Search Terminal Help
[root@R5-SVR6 network-scripts]#
[root@R5-SVR6 network-scripts]# cat ifcfg-bond0
DEVICE=bond0
NAME=bond0
TYPE=Bond
BONDING_MASTER=yes
IPADDR=172.16.45.11
NETMASK=255.255.255.0
ONBOOT=yes
BOOTPROTO=none
BONDING_OPTS="mode=1 miimon=100"
[root@R5-SVR6 network-scripts]#

```

Figure 15. Bond0 interface Configuration File

The example file shows Bond0 network interface configuration. The network interface is configured as a Master interface, Bond interface type, and is in Mode 1 (Active-Backup). Bond0 network interface is configured for IP Address 172.15.45.11.

In this example, static IP Address scheme is used. Alternatively, Bond0 can be configured for DHCP.

9. Execute the **ifup bond0** command to bring up the bonded network interface.

Figure 16 shows contents of `/proc/net/bonding/bond0`.

```

root@R5-SVR6:/etc/sysconfig/network-scripts
File Edit View Search Terminal Help
[root@R5-SVR6 network-scripts]# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: None
Currently Active Slave: enp129s10
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: enp129s10
MII Status: up
Speed: Unknown
Duplex: Unknown
Link Failure Count: 0
Permanent HW addr: 52:e3:42:13:96:c6
Slave queue ID: 0

Slave Interface: enp129s2
MII Status: up
Speed: Unknown
Duplex: Unknown
Link Failure Count: 0
Permanent HW addr: 2e:2a:6f:c0:c5:bd
Slave queue ID: 0
[root@R5-SVR6 network-scripts]#

```

Figure 16. Bond0 Configuration



10. Use the **ping 172.16.45.11** command from another server on the network to confirm connectivity to the bonding interface.
11. Install and configure Apache Web Server using the **yum install httpd** command from a Linux shell prompt.
12. Update the `/etc/httpd/conf/httpd.conf` file with Bond0 network interface IP Address for Apache Web Server.

Figure 17 shows contents of the `httpd.conf` file. Apache Web Server is configured to listen on Port 80, and is bound to IP Address 172.16.45.11.

```
root@R5-SVR6:/etc/httpd/conf
File Edit View Search Terminal Help
#
ServerRoot "/etc/httpd"
#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, instead of the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
Listen 172.16.45.11:80
#Listen 80
#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding 'LoadModule' lines at this location so the
# directives contained in it are actually available _before_ they are used.
# Statically compiled modules (those listed by 'httpd -l') do not need
# to be loaded here.
#
```

Figure 17. Apache Web Server Configuration file

Note: Follow your Linux Distribution's best known method to properly configure and secure Apache Web Server.

2.3 Process Binding

Extending the example setup introduced in [Section 2.2, “Application Binding”](#), the Linux backup process (rsync) is bound to a specific network interface. The network interface is a VF associated with a physical network port of the Intel® Ethernet Converged Network Adapter XL710, delivering up to 40 GbE Network connectivity per physical port. An example is shown in [Figure 12](#).

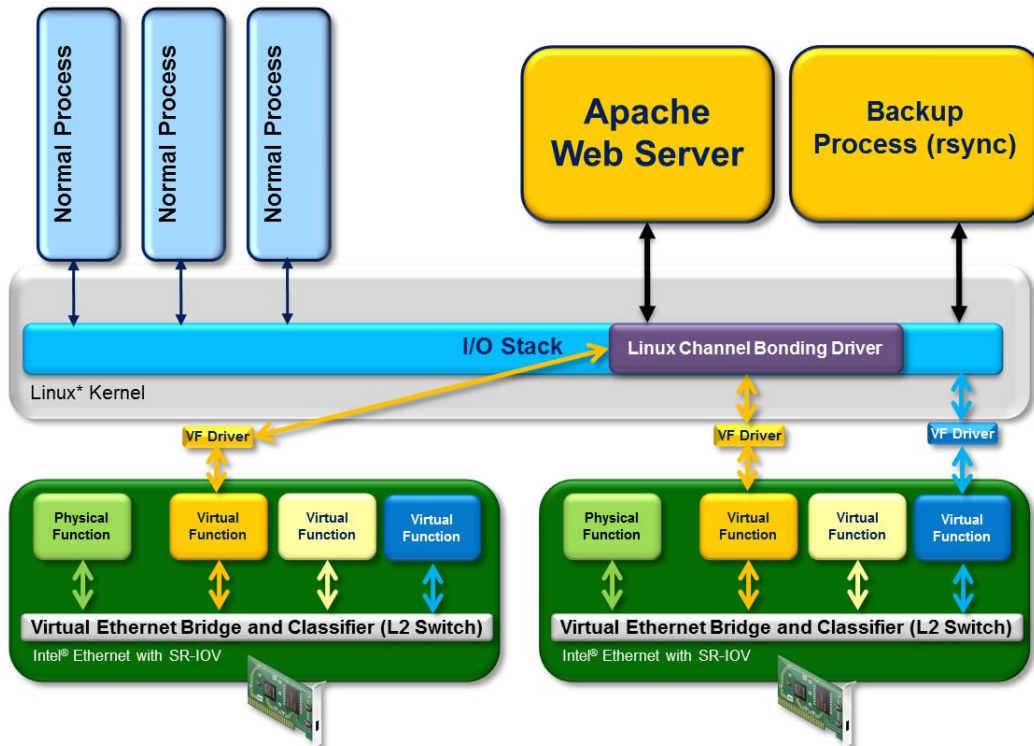


Figure 18. Process Binding Setup Diagram

Rsync can run in standalone and daemon mode. Standalone mode is used for this example.

1. Select an unused VF and assign an IP address to it. In this example, IP Address 172.16.123.45 and Subnet Mask 255.255.255.0 are assigned to the VF.
2. Configure **rsync** to perform backup. In this example, the rsync process is bound to IP Address 172.16.123.45. This binding ensures that the rsync process is using VF for backups.

```

root@R5-SVR6:~/Desktop
File Edit View Search Terminal Help
[root@R5-SVR6 Desktop]#
[root@R5-SVR6 Desktop]#
[root@R5-SVR6 Desktop]# rsync -avzhe ssh /home/paelab/ paelab@172.16.123.200:/backups --address=172.16.123.45

```

Figure 19. rsync Process Command Line Example

The example in Figure 19 shows the **rsync** process command configured to back up the contents of the `/home/paelab/` directory to the `/backups/` folder on a server with the IP Address 172.16.123.200, and the process is using a local network interface with the IP Address 172.16.123.45.

3. System Administrators can further fine tune the **rsync** process by limiting bandwidth utilizing the X710 and XL710 adapter's Transmit Rate Limit feature. The example in Figure 20 shows how to configure Transmit Rate Limit for limiting VF # 2 to utilize maximum of 500 MB of available bandwidth.

```

root@R5-SVR6:~/Desktop
File Edit View Search Terminal Help
[root@R5-SVR6 Desktop]#
[root@R5-SVR6 Desktop]#
[root@R5-SVR6 Desktop]# ip link set ens787f0 vf 2 rate 500
[root@R5-SVR6 Desktop]#
[root@R5-SVR6 Desktop]#
[root@R5-SVR6 Desktop]#

```

Figure 20. Link Transmit Rate Limit Example

2.4 Virtual Machine Binding

Building on the example setups introduced in Section 2.2 and Section 2.3, one or more VFs can be assigned to Virtual Machines. For example, an existing VM running Ubuntu 14.04 is assigned a VF. An example is shown in Figure 21.

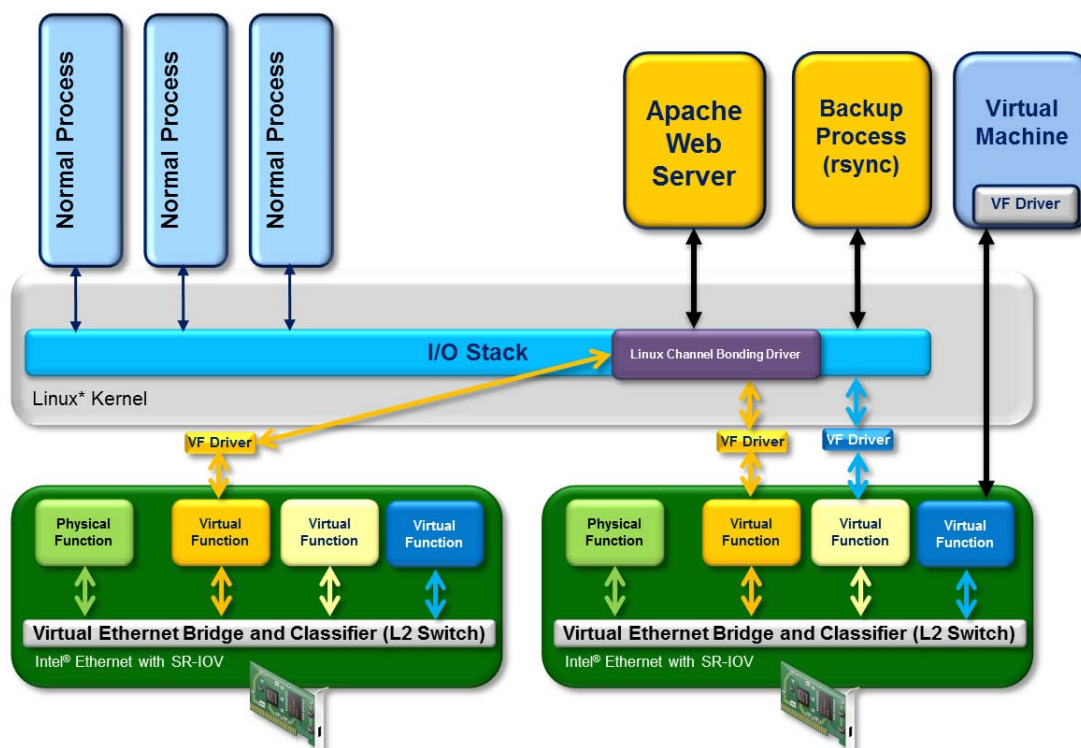


Figure 21. Virtual Machine Binding Setup Diagram

1. Select one or more VFs that are to be assigned to VMs. Remember the bus, device, and function numbers for these VFs, as this information will be used later. The VF driver claims all VFs visible on the PCIe bus when it is loaded within the Linux host. To assign VFs to one or more VMs, VFs should be unbound from the VF driver. The example in [Figure 22](#) shows how to release VFs from VF driver.




```

root@R5-SVR6:~/Desktop
File Edit View Search Terminal Help
[root@R5-SVR6 Desktop]#
[root@R5-SVR6 Desktop]#
[root@R5-SVR6 Desktop]#
[root@R5-SVR6 Desktop]#
[root@R5-SVR6 Desktop]# echo "0000:81:02.1" > /sys/bus/pci/devices/0000\:81\:02.1/driver/unbind
[root@R5-SVR6 Desktop]#
[root@R5-SVR6 Desktop]#
[root@R5-SVR6 Desktop]#
[root@R5-SVR6 Desktop]#

```

Figure 22. Unbinding VF from Driver

The example in [Figure 22](#) shows the VF identified by 81:02.1 is being unbound from the VF driver by writing to the `sysfs`. Once the VF driver releases the VF, it can be assigned to a VM.

2. Use virt-manager to create a VM.
3. Click on the  icon to edit the VM properties.
4. Click on the **Add Hardware** icon to display the **Add New Virtual Hardware** wizard, as shown in [Figure 23](#).

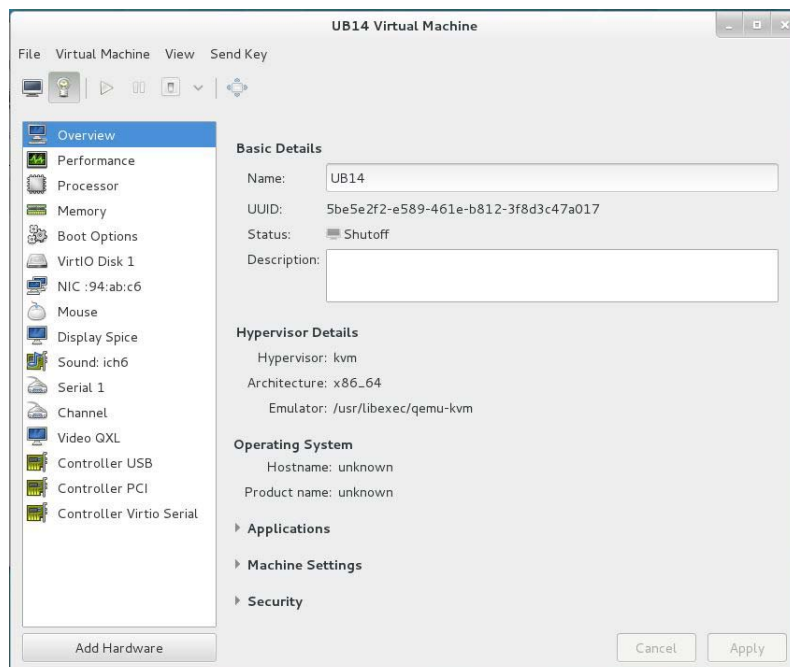


Figure 23. Virtual Machine Configuration Page

5. Click **PCI Host Device** to display the **Add New Virtual Hardware** window, as shown in [Figure 24](#).

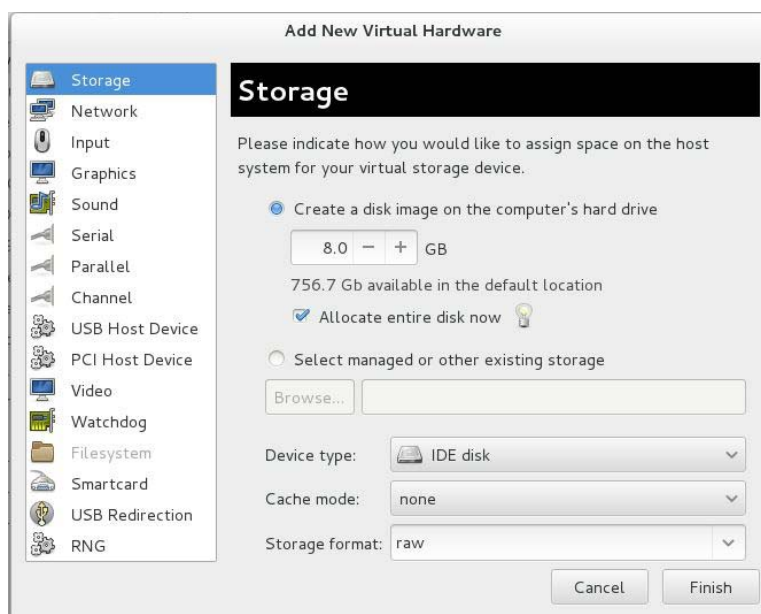


Figure 24. Add New Virtual Hardware Page

6. Select an **XL710 X710 Virtual Function** and click the **Finish**.

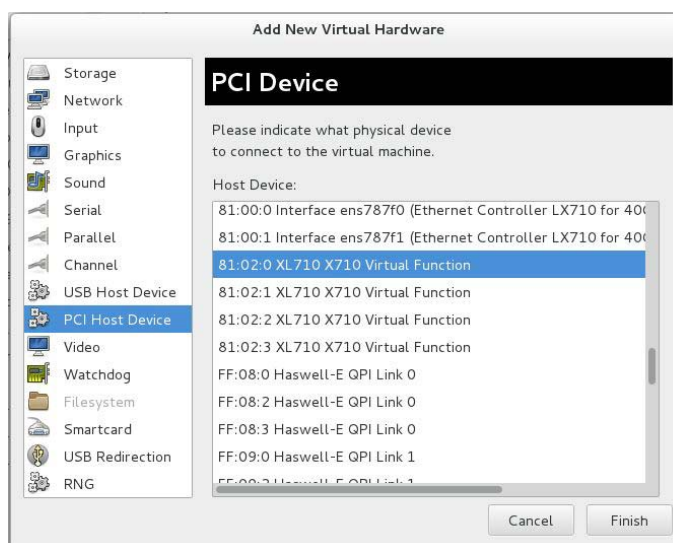
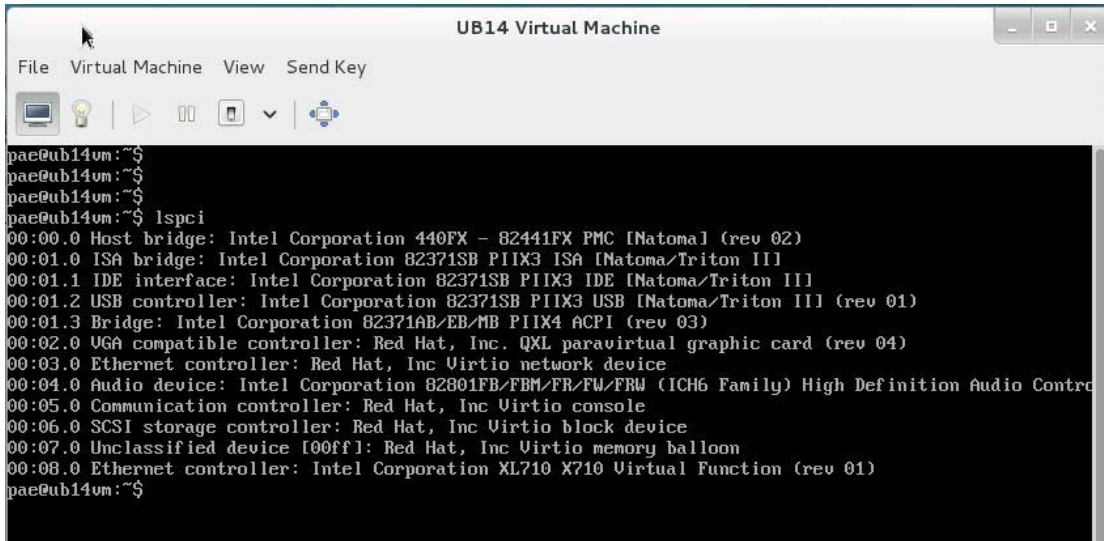


Figure 25. PCI Device Selection Page

In [Figure 25](#), the Intel Ethernet XL710 or X710 Virtual Functions are listed as "XL710 X710 Virtual Function". One or more VFs can be assigned to a VM. Upon successful assignment, the VM is ready to use.

7. Power up the Ubuntu 14.04 VM. Log into the VM using the credentials created during the VM installation process.
8. At the Linux Console, use the Linux **lspci** utility to confirm that the assigned VF is shown within the VM's PCIe hierarchy, as shown in [Figure 26](#).

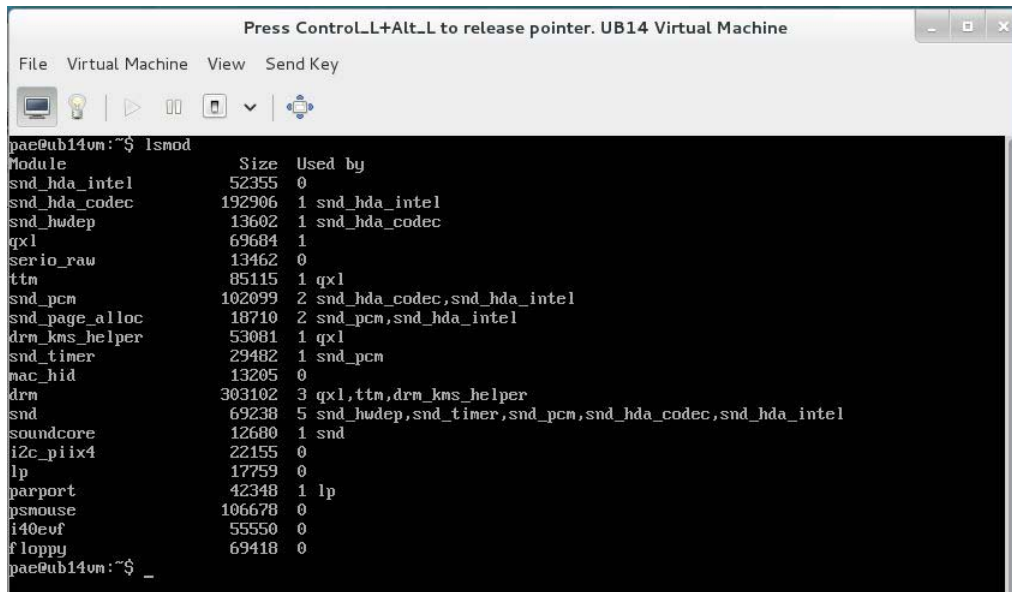


```

UB14 Virtual Machine
File Virtual Machine View Send Key
pae@ub14vm:~$
pae@ub14vm:~$
pae@ub14vm:~$
pae@ub14vm:~$ lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton III]
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton III]
00:01.2 USB controller: Intel Corporation 82371SB PIIX3 USB [Natoma/Triton III] (rev 01)
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)
00:02.0 VGA compatible controller: Red Hat, Inc. QXL paravirtual graphic card (rev 04)
00:03.0 Ethernet controller: Red Hat, Inc Virtio network device
00:04.0 Audio device: Intel Corporation 82801FB/EB/FR/FW/FRW (ICH6 Family) High Definition Audio Controller
00:05.0 Communication controller: Red Hat, Inc Virtio console
00:06.0 SCSI storage controller: Red Hat, Inc Virtio block device
00:07.0 Unclassified device [00ff]: Red Hat, Inc Virtio memory balloon
00:08.0 Ethernet controller: Intel Corporation XL710 X710 Virtual Function (rev 01)
pae@ub14vm:~$
  
```

Figure 26. "lspci" Output of the VM

9. Use the Linux **lsmod** utility to confirm that *i40evf* driver for the VF has loaded successfully, as shown in [Figure 27](#).



```

Press Control_L+Alt_L to release pointer. UB14 Virtual Machine
File Virtual Machine View Send Key
pae@ub14vm:~$ lsmod
Module                  Size  Used by
snd_hda_intel           52355  0
snd_hda_codec          192906  1 snd_hda_intel
snd_hudep               13602  1 snd_hda_codec
qxl                     69684  1
serio_raw              13462  0
ttm                     85115  1 qxl
snd_pcm                102099  2 snd_hda_codec,snd_hda_intel
snd_page_alloc         18710  2 snd_pcm,snd_hda_intel
drm_kms_helper         53081  1 qxl
snd_timer              29482  1 snd_pcm
mac_hid                13205  0
drm                   303102  3 qxl,ttm,drm_kms_helper
snd                    69238  5 snd_hudep,snd_timer,snd_pcm,snd_hda_codec,snd_hda_intel
soundcore              12680  1 snd
i2c_piix4              22155  0
lp                     17759  0
parport                42348  1 lp
psmouse               106678  0
i40evf                 55550  0
floppy                 69418  0
pae@ub14vm:~$ _
  
```

Figure 27. "lsmod" Output



10. Use the Linux **ifconfig** utility to confirm that the newly assigned VF is ready for use, as shown in Figure 28.

```
pa@ub14vm:~$  
pa@ub14vm:~$  
pa@ub14vm:~$ ifconfig  
eth1      Link encap:Ethernet  HWaddr 7e:8c:b9:5e:55:52  
          inet addr:192.168.122.10  Bcast:192.168.122.255  Mask:255.255.255.0  
          inet6 addr: fe80::7c8c:b9ff:fe5e:5552/64 Scope:Link  
          UP BROADCAST MULTICAST  MTU:1500  Metric:1  
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)  
  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:65536  Metric:1  
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0  
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)  
  
pa@ub14vm:~$ [ 3818.456135] i40evf 0000:00:08.0: Reset was not detected
```

Figure 28. “ifconfig” Output

11. The VF can be configured for DHCP or static IP address assignment. The VF is ready to communicate once it has an IP address assigned.

3.0 Summary

Intel Flexible Port Partitioning (FPP) technology utilizes industry standard PCI SIG SR-IOV to efficiently divide a physical Ethernet device into multiple virtual devices. Intel FPP provides an instant Quality of Service by ensuring each process is assigned to a VF and is provided a fair share of available bandwidth.

Intel FPP requires an SR-IOV-enabled platform, OS, and an Intel SR-IOV-capable Ethernet device. There are no requirements for special hardware outside of the system. Intel FPP is available today on several Linux* based operating systems. Intel continues to work with the Linux community to add additional features to the kernel. Intel FPP uses SR-IOV virtual functions for kernel processes as well as for virtual machines.

Intel's best-of-breed 40 GbE solutions are now available with I/O Virtualization capabilities. Customers get world-class Ethernet support along with I/O virtualization support in mainstream Linux distributions in a single adapter.



4.0 Customer Support

Intel® Customer Support Services offers a broad selection of programs, including phone support and warranty service. For more information, contact us as the following:

[support.intel.com/support/go/network/ adapter/home.htm](https://support.intel.com/support/go/network/adapter/home.htm)

support.intel.com/support/go/network/adapter/home.htm

Service and availability may vary by country.

5.0 Product Information

To see the full line of Intel Network Adapters for PCI Express*, visit www.intel.com/go/ethernet.

To speak to a customer service representative regarding Intel products, please call 1-800-538-3373 (U.S. and Canada) or visit support.intel.com/support/go/network/contact.htm for the telephone number in your area.