U25N documentation

Release 1.1

Xilinx, Inc.

Apr 14, 2022

List of Figures **iii**

1 2 Supported Services **1**
1.1 2.1 U25N Modes: Legacy and Switchdev Mode . . . 1
1.2 2.2 Open vSwitch . . . 1
1.3 2.3 IPsec . . . 3
1.4 2.4 Stateless Firewall . . . 3
1.5 2.5 DPDK on U25N . . . 3

2 3. U25N Installation **5**
2.1 3.1 Basic Requirements and Component Versions Supported . . . 5
2.2 3.2 U25N Driver . . . 5
2.3 3.3 U25N Shell . . . 7
2.4 3.5 Reverting the U25N SmartNIC to Golden Image . . . 12

3 4 Detailed Applications Description **13**
3.1 4.1 Legacy and Switchdev Modes . . . 13
3.2 4.2 OVS . . . 14
3.3 IPsec . . . 36
3.4 4.4 Stateless Firewall . . . 43
3.5 4.5 Statistics . . . 46
3.6 4.6 Debug Commands . . . 47

4 DPDK **51**

5 APPENDIX A U25N Shell Programming **53**
5.1 Entering into U-Boot Mode . . . 53

6 APPENDIX B VM Installation **63**

Figure 1: *Figure 1:* **Alveo U25N SmartNIC** 65
Figure 2: *Figure 2:* **U25N Architecture** 65

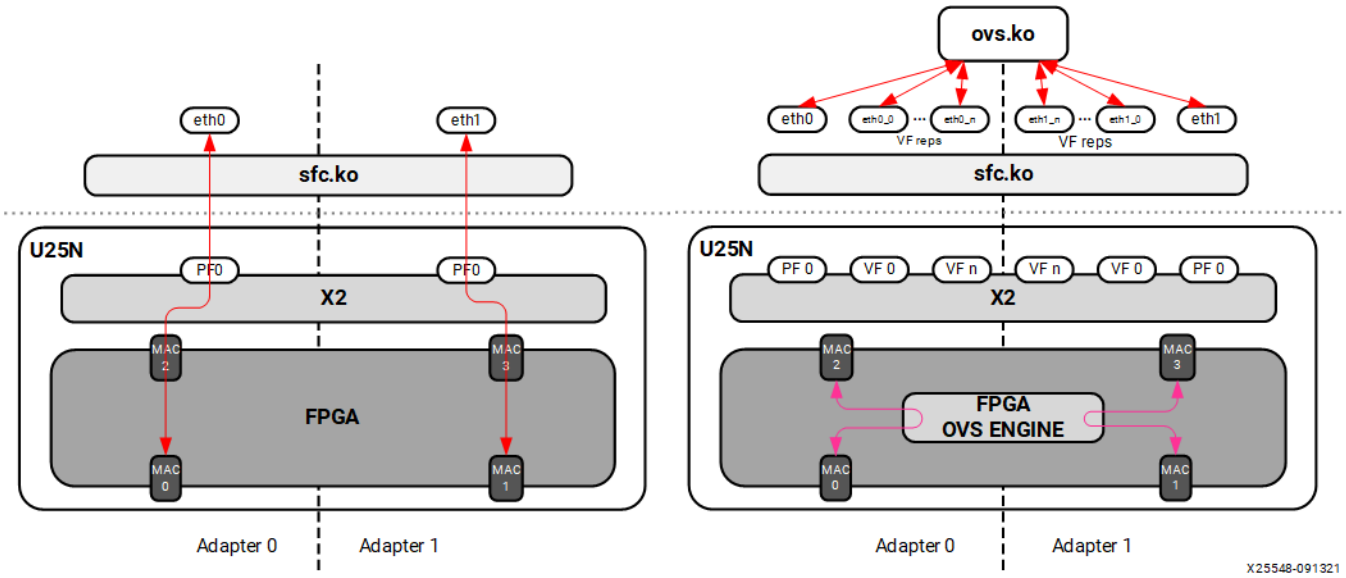
2 Supported Services

1.1 2.1 U25N Modes: Legacy and Switchdev Mode

The U25N SmartNIC can operate in two modes:

Switchdev Mode with Open vSwitch (OVS): This mode allows offloading the Linux kernel forwarding data plane to U25N SmartNIC. Legacy Mode without OVS: In this mode, packets are forwarded in and out of the U25N SmartNIC without any modifications.

Figure 3: **U25N Modes of Operation**



1.2 2.2 Open vSwitch

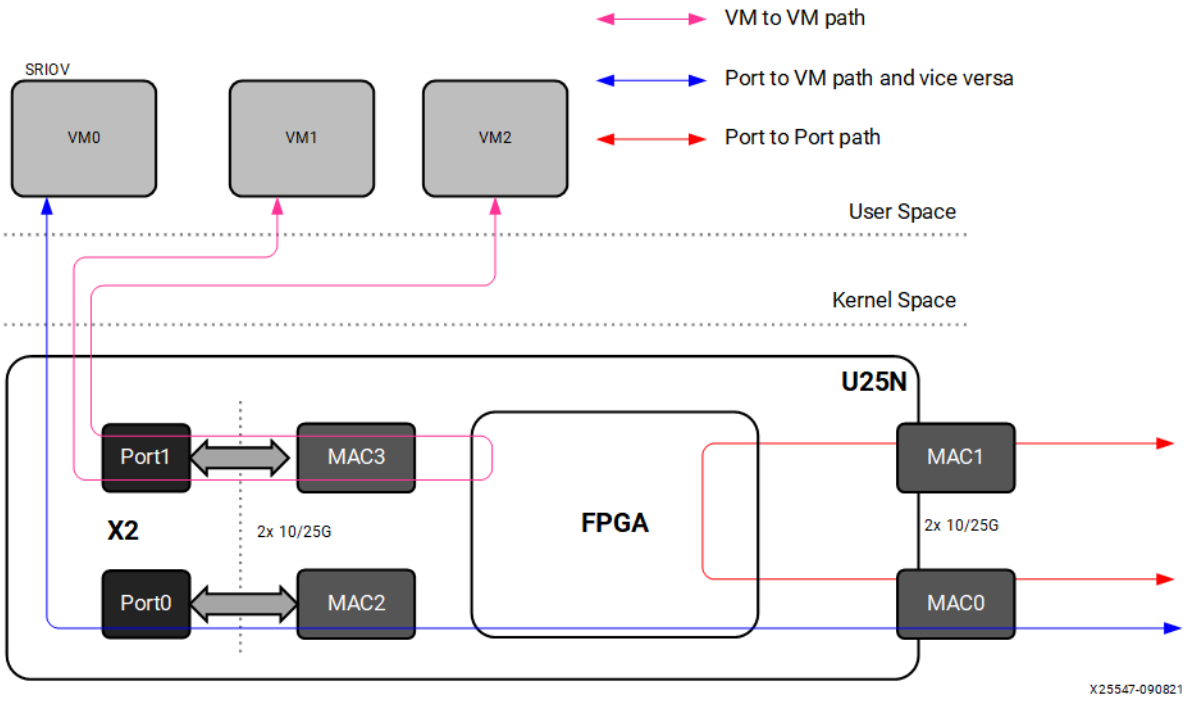
Open vSwitch (OVS) is a software-defined virtual switch that a hypervisor uses to manage traffic between virtual instances. It performs not only classic layer-2 switching, but also supports arbitrarily complex rules matching on any packet field, and performing any transformation. OVS groups packet flows requiring the same switching behavior into megaflows that can require filtering at layer 2, layer 3, or layer 4 (potentially changing over time) of the Network stack. Deployments can require very large numbers of simultaneous megaflows to be supported. The downside of OVS is that it is resource-intensive. Cloud providers have to dedicate a significant number of cores to run this infrastructure, instead of renting those cores to their customers for running business applications.The U25N SmartNIC can offload the virtual switching off of the host processor and hence improve CPU efficiency .

OVS supports various dataflows: port to VM, VM to VM, and port to port. The following section contains detailed information about these dataflows.

1.2.1 2.2.1 Port to Port

In this configuration, traffic flows from one physical port to the other physical port of the U25N SmartNIC and vice versa . The OVS datapath is offloaded to the SmartNIC, and the control path runs on the host. Initial incoming packets with no flow/entry in the OVS dataplane are forwarded up to the host(where vSwitchd daemon runs) for flow/rule learning. Subsequently, the flow/rules are propagated to the OVS dataplane present in the SmartNIC which handles further packet switching.

Figure 4: **OVS Offload Scenario**



1.2.2 2.2.2 Port to VM or VM to Port

In this configuration, traffic flows from the host or VMs to U25N physical ports, and vice versa. This configuration is achieved by creating virtual functions (VFs) corresponding to a physical function (PFs) using sfboot command (for more information refer to the Solarflare Server Adapter user guide at https://support-nic.xilinx.com/wp/drivers). In this configuration packet switching is done by OVS.

1.2.3 2.2.3 VM to VM

In this configuration, traffic flows from a VM to another VM via OvS kernel module present in the SmartNIC. Packet switching is done by OvS in this configuration.

1.2.4 2.2.5 Supported Overlay Network protocols

2.2.5.1 L2GRE

OVS supports layer 2 over generic routing encapsulation (L2GRE). L2GRE tunnel bridges two discrete LAN segments over the internet by creating a tunnel. At the origin of the tunnel, packets are encapsulated and at the terminating end, packets are decapsulated. The constant encapsulation and decapsulation puts a tremendous load on the host CPU. With the help of hardware offload, packet encapsulation and decapsulation is offloaded to the U25N SmartNIC and does not consume valuable host CPU cycles.

2.2.5.1 VXLAN

OVS supports Virtual eXtensible Local Area Network or VXLAN. VXLAN is an overlay network that transports an L2 network over an existing L3 network. For more information on VXLAN, please see RFC 7348. Like L2GRE the VXLAN tunnel creation and packet encapsulation/decapsulation process can be offloaded to the U25N Smart NIC, thus saving valuable CPU cycles.

1.3 2.3 IPsec

Internet protocol security (IPsec) is a secure network protocol suite that authenticates and encrypts packets to provide a secure channel between two endpoints over an internet protocol network. Encrypted packets are forwarded to the CPU and handled by an open-source application called strongSwan (see https://www.strongswan.org). Due to the computing complexity of encryption and decryption of packets, handling IPsec in the host CPU consumes significant compute cycles. Offloading this operation to U25N hardware easily increases system-level throughput and lowers CPU utilization.

Currently, IPsec transport mode is supported in the gateway mode, i.e., traffic from one port is encrypted and sent to the other port.

1.4 2.4 Stateless Firewall

Stateless firewalls make use of a packet's source, destination, and other parameters to figure out whether it presents a threat. Stateless firewall should be added on the ingress of ports so that filtering based on nftables rules can be applied. nftables is a subsystem of the Linux kernel providing filtering and classification of network packets/datagrams/frames. First-level filtering is done based upon nftables rules. The hardware offload is available for nftables through the netdev family and the ingress hook. This also includes base chain hardware offloading.

1.5 2.5 DPDK on U25N

Data Plane Development Kit or DPDK is a kernel bypass technique to accelerate packet processing. DPDK enables the applications running in user-space to interact directly with NIC, bypassing the Linux kernel space. DPDK uses Poll Mode Driver (PMD) to interact with the underlying NIC. DPDK can be run on the X2 PF or VF. To run the testpmd/pktgen application on the U25N, refer to Solarflare libefx-based Poll Mode DriverDPDK. For more information about how to run DPDK on the U25N, refer to DPDK.

3. U25N Installation

2.1 3.1 Basic Requirements and Component Versions Supported

- OS requirement: Ubuntu 18.04 or 20.04.
  ***Note*:** We recommend using the LTS version of Ubuntu.
- Kernel requirements:
  - OVS Functionality ≥ 4.15.
  - Stateless Firewall Functionality ≥ 5.5.

  ***Note*:** We recommends the default kernel 5.8 that comes with Ubuntu 20.04.02 LTS to support all features of the U25N hardware.
- PCIe® Gen3 x16 slot.
- Requires passive airflow. More details can be found in *Alveo U25N SmartNIC Data Sheet* (DS1005).
- U25N driver version: 5.3.3.1008.3 (minimum).
- X2 firmware version: v7.8.17.1011 (minimum).
  ***Note*:** To install the latest firmware, refer to the Updating U25N Firmware section.
- OVS version: 2.12 and above. For more information about OVS and its installation, refer to https://docs.openvswitch.org/en/latest/intro/install/general.

2.2 3.2 U25N Driver

***Note*:** Root user privileges are required to execute following steps and commands.

Once the server is successfully booted with U25N Smart NIC, validate that the U25N Smart NIC is detected using the lspci command:

```
lspci | grep Solarflare
```

It will list two PCIe id's corresponding to U25N, similar to the following sample output:

```
af:00.0 Ethernet controller: Solarflare Communications XtremeScale SFC9250
10/25/40/50/100G Ethernet Controller (rev 01)
af:00.1 Ethernet controller: Solarflare Communications XtremeScale SFC9250
10/25/40/50/100G Ethernet Controller (rev 01)
```

2.2.1 Solarflare Linux Utilities Installation

The Solarflare Linux Utilities package can be found from the U25N software package downloaded

from the lounge.

1. Download and unzip the package on the target server.
   ***Note*:** Alien package must be downloaded using the command `sudo apt install alien`.
2. Create the `.deb` file:

   ```
   sudo alien sfutils-<version>.x86_64.rpm
   ```

   This command generates the `sfutils_<version>_amd64.deb` file.
3. Install the `.deb` file:

   ```
   sudo dpkg -i sfutils_<version>_amd64.deb
   ```

The server should have sfupdate, sfkey, sfctool and sfboot utilities now.

2.2.2 3.2.1 U25N Driver Installation

***Note*:** Install the dkms package with the following command if not available:

```
sudo apt-get install dkms
```

***Note*:** If the driver version is 5.3.3.1008.3, please ignore Step 1. Check the driver version of U25N interface using the following command:

```
ethtool -i U25N_eth0 | grep version #U25N_eth0 is the first port of U25N.
```

Step 1 : If the debian package already exists, before installing the latest debian package, remove the already existing package.

```
modprobe mtd  [for first time alone]

modprobe mdio  [for first time alone]

rmmod sfc

dpkg -r sfc-dkms

dpkg -i sfc-dkms_x.x.x.x_all.deb

modprobe sfc
```

For Eg:

```
dpkg -i sfc-dkms_5.3.3.1008.3_all.deb
```

2.2.3 3.2.2 Updating U25N Firmware

Step 1: Make sure U25N driver is loaded using `lsmod` command

```
lsmod | grep sfc
```

For Eg :

```
lsmod | grep sfc
sfc                   626688  0
```

Step 2: Execute the following step to update firmware

```
sfboot --list # This should output U25N interfaces
sfupdate #This should work or do the below step
sfupdate -i <U25N_eth0> --write --force --yes # U25N_eth0 is the PF0 interface of
```

```
U25N
```

Note: Please use the PF0 interface. No reboot is required.

Step 3: Confirm the firmware version using the command `sfupdate`. Version should be 1011.

Eg:

```
sfupdate | grep Bundle
```

Bundle type: U25N Bundle version: v7.8.17.1011

2.2.4 3.2.3 sfboot Configuration

***Note*:** Ignore the following steps if the U25N is operating in SR-IOV mode and the required VF are already assigned. Available VFs and SR-IOV mode can be verified by running the sfboot command as a root user.

For more information on enabling VFs and SR-IOV, refer to the solarflare user guide

X2 switch mode must be configured in SR-IOV mode for enabling VFs. Each U25N PF can have up to 120 VFs. The following command demonstrates how to switch to the SR-IOV mode and enable VFs.

```
sudo sfboot switch-mode=sriov vf-count=<No. of VF required>
```

***Note*:** Perform a power cycle to update the configuration. Updated configuration will be retained post cold boot.

2.3 3.3 U25N Shell

The U25N shell is programmed on the U25N with a known good image which is called "golden image" out of the factory. The Following sections demonstrate how to verify the shell. If the validation fails, refer to U25N Shell Programming.

2.3.1 3.3.1 Shell Version Check

Step 1 : Check the driver version of U25N interface using the following command:

***Note*:** Ignore Step 1, if the U25N driver version is 5.3.3.1008.3.

```
ethtool -i U25N_eth0 | grep version
```

***Note*:** To install the latest sfc driver, refer to U25N Driver.

Step 2 : Use the u25n_update utility (found inside RELEASE_DIR/utils directory) to find out the U25N shell version. Execute the following command to read the U25N shell version.

```
./u25n_update get-version <U25N_eth0>
```

Eg:

```
./u25n_update get-version u25eth0

Sample output:
[u25n_update] - Image Upgrade/Erase Utility V3.1

Reading version from the hardware

X2 Firmware Version : 7.8.17.1011
PS Firmware Version : 5.00
Deployment shell Version : 0x0812210D # Timestamp of Shell
Bitstream Version : 0xA00E60D3 # Timestamp of offload
Features supported:  MAE IPSEC # Represent NIC by default
```

```
Timestamp : 08-12-2021 16:30:45


STATUS: SUCCESSFUL
```

You should get following prompt post successful U25N version read.

```
STATUS: SUCCESSFUL
```

***Note*:** If the u25n_update command returns status as failed, this implies that the shell is not programmed with the Golden image. Please contact support. Refer to U25N Shell Programming to flash the Golden image to the U25N shell.

The following section demonstrates how to validate the Golden Image Functionality. For flashing deployment image to the U25N shell refer to Deployment Image Flashing.

***Note*:** If the Golden image version is same as the Deployment image version, flashing is not required.

2.3.2 3.3.2 Testing Golden Image Functionality

1. The U25N shell with Golden image includes Image Upgrade and Basic NIC functions.
2. U25N hardware in legacy mode where the acceleration logic is simple passthrough from ethernet ports to host driver.

3.3.2.1 Checking Basic NIC Functionality

Connect the U25N physical port to a traffic generator or any peer NIC device.

```
Run ping after assigning IP addresses on source and remote ports.
Run iperf server and client on any ports to verify traffic.
Run DPDK testpmd at the X2 PF interface. Packets sent to the physical port will be
 received at the testpmd application corresponding to the bound PF.
```

***Note*:** Refer to DPDK on U25N to run dpdk-testpmd.

2.3.3 3.3.3 Deployment Image Flashing

***Note*:** U25N must have the Golden Image before performing any of the following steps. Refer to Shell Version Check to check the Shell version. Ignore the flashing if Golden image version is same as deployment version in the release.

***Note*:** Before flashing the image ensure the U25N card is in legacy mode with no OVS software application running. Also, remove all associated OVS databases(if any)

```
To check the mode of operation:(Legacy/Switchdev modes)
devlink dev eswitch show pci/0000:<pci_id> #pci_id is the BDF of U25N Device
```

Step 1 : In case U25N driver version is 5.3.3.1008.3, this step can be ignored. The following command can be used to check the driver version:

```
ethtool -i U25N_eth0 | grep version
```

***Note*:** To install the latest sfc driver, please refer to U25N Driver.

Step 2 : Ensure that the U25N card is in legacy mode. Below is the command to validate:

```
devlink dev eswitch show pci/0000:<pci_id>
```

The command mentioned above should return the following sample output:

```
pci/0000:<pci_id>: mode legacy
```

Use the following command to switch mode in case the card is not already in the legacy mode:

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode legacy
```

Step 3 : The u25n_update utility uses U25N PF0 network interface to flash the image. Execute the following command to commence image flash. ***Note*:** The interface must be the PF0 interface of the target U25N card. The file must be a `.bin` file. Ensure the image path to be flashed is correct.

```
./u25n_update upgrade <path_to_bin_or_ub_file_with_extension> <PF0_interface> #
The bin file should be in the shell directory of the release.
```

Eg:

```
#From Release_dir/utils:
./u25n_update upgrade ../shell/BOOT.bin u25eth0
#The name of the bin file may change to represent month and day like
U25N_Shell_Dec13.bin
```

Post successful image flash you will the following prompt:

```
STATUS: SUCCESSFUL
```

Step 4 : Reset the U25N hardware to boot from the updated deployment image.

```
./u25n_update reset <U25N_eth0>
```

Eg:

```
./u25n_update reset u25eth0
```

Post a successful reset the following prompt will be shown:

```
STATUS: SUCCESSFUL # The above command takes a few seconds to complete to ensure
proper reset sequence is validated.
```

Step 5 : To retrieve U25N Deployment image version details, use the following commands:

```
./u25n_update get-version <U25N_eth0>
```

Eg:

```
./u25n_update get-version u25eth0
#Sample Output:
[u25n_update] - Image Upgrade/Erase Utility V3.1

Reading version from the hardware

X2 Firmware Version : 7.8.17.1011
PS Firmware Version : 5.00
Deployment shell Version : 0x0812210D
Bitstream Version : 0xA00E60D3
Features supported:  No Features supported
Timestamp : 08-12-2021 16:30:45


STATUS: SUCCESSFUL
```

The following prompt will be shown, post a successful retrieval

```
STATUS: SUCCESSFUL
```

2.3.4 3.3.3.1 Loading Partial Bitstream into Deployment Shell

***Note*:** The U25N must be flashed with the Deployment image before running the following

commands. Refer to the [Deployment Image Flashing](./ug1534-installation.md#deployment-image-flashing) to flash the deployment image to the U25N shell.

***Note*:** In case partial Bitstream already exists in the U25N card and/or if a warm reboot has already been done, ensure a fpga reset is performed using u25n_update application before following the steps mentioned below. Use the following command to reset the FPGA.

```
./u25n_update reset <U25N_eth0>
```

Eg:

```
./u25n_update reset u25eth0
```

Use the PF0 interface of U25N card to initiate the FPGA reset.

Step 1. In case U25N driver version is 5.3.3.1008, this step can be ignored. Driver version of the U25N can be validated using the following command:

```
ethtool -i U25_eth0 | grep version
```

***Note*:** To install the latest sfc driver, refer to U25N Driver.

***Note*:** Before flashing the image, make sure the U25N Smart NIC is in legacy mode, with OVS software application running, and all OVS databases are removed.

Step 2. The U25N Smart NIC should be in legacy mode. Use the following command to verify this:

```
devlink dev eswitch show pci/0000:<pci_id>
```

Sample output of the above command:

```
pci/0000:<pci_id>: mode legacy
```

If the U25N hardware is not in legacy mode, change it to legacy mode using the following command:

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode legacy
```

***Note*:** The interface used in the above command should be the PF0 interface of the target U25N SmartNIC. The file must be a `.bit` file. Ensure the path to the image to be flashed is correct.

Step 3. Image flashing happens through the U25N PF0 network interface using the u25n_update utility.

1. The image is flashed using the following CLI command:

***Note*:** The interface used in the above command should be the PF0 interface of the target U25N SmartNIC. The file must be a `.bit` file. Ensure the path to the image to be flashed is correct.

```
./u25n_update upgrade <path_to_bit_file_with_extension> <PF0_interface>
# the path to bit files is Release_dir/bits
```

E.g:

```
./u25n_update upgrade ../bits/ovs_*.bit u25eth0
```

1. After the image is flashed successfully, the following message is displayed:

   ```
   STATUS: Successful
   ```

2. The u25n_update utility can be used to retrieve the bitstream version. Version retrieval happens at the U25N PF0 network interface using the u25n_update utility. Use the following command to retrieve the image version:

```
./u25n_update get-version <U25N_eth0>
```

E.g:

```
./u25n_update get-version u25eth0
```

Bitstream version: 0xA00D10D1

Post a successful image retrieval, the following message is displayed:

```
STATUS: SUCCESSFUL
```

2.3.5 3.4 Steps to change MPSoC Linux kernel image (For advanced users)

Follow this Section only when the file system needs to be updated for the existing Deployment image.

Deployment image must be flashed to the U25N shell before following the below mentioned steps. Refer to [Deployment Image Flashing](./ug1534-installation.md#deployment-image-flashing) to flash the Deployment image.

***Note*:** Before flashing the image to the shell ensure the card is in legacy mode with no OVS software application running and all OVS databases have been removed.

Step 1 : If the U25N driver version is 5.3.3.1008.3, this step can be ignored. Driver version of the U25N can be validated by using the following command:

```
ethtool -i U25_eth0 | grep version
```

***Note*:** To install the latest sfc driver, refer Section 3.2.

Step 2 : U25N Cards should be in legacy mode. Use the following command to validate:

```
devlink dev eswitch show pci/0000:<pci_id>
```

Sample output of above command:

```
pci/0000:<pci_id>: mode legacy
```

In case U25N hardware is not in legacy mode, change it to legacy mode using the following command:

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode legacy
```

Step 3 : Image flashing is performed through the U25N PF0 network interface using the u25n_update utility.

Image is flashed using a CLI command

***Note*:** The interface should be the PF0 interface of the target U25N card. The file must be a `.ub` file. Make sure the path to the image to be flashed is given correctly.

```
./u25n_update upgrade <path_to_bin_or_ub_file_with_extension> <PF0_interface>
```

Eg:

```
./u25n_update upgrade image.ub <u25eth0>
```

Post successful image flash, the following message will be displayed:

```
STATUS: SUCCESSFUL
```

Step 4 : Reset the U25N hardware to boot with the new kernel image using the following command:

```
./u25n_update reset <PF0_interface>
```

Eg:

```
./u25n_update reset <u25eth0>
```

After successful image flash, the following message will be displayed:

```
STATUS: SUCCESSFUL
```

2.4 3.5 Reverting the U25N SmartNIC to Golden Image

Reverting to the factory (or golden) image is recommended in the following cases:

- Preparing to flash a different shell onto the SmartNIC.
- The SmartNIC no longer appears on lspci after programming a custom image onto the SmartNIC.
- To recover from the state when the deployment image has become unresponsive even after performing multiple resets using the u25n_update utility.

***Note*:** After the factory reset is performed, the U25N loaded with the golden image (A known good image). It has the essential provision to upgrade the deployment image.

Step 1. In case the U25N driver version is 5.3.3.1008, this step can be ignored. Driver version of the U25N interface can be validated using the following command:

```
ethtool -i U25_eth0 | grep version
```

***Note*:** To install the latest sfc driver, refer to the U25N Driver.

***Note*:** Before flashing the image, make sure the SmartNIC is in legacy mode, no OVS software application is running, and OVS databases are removed.

Step 2. The U25N SmartNIC should be in legacy mode. Use the following command to verify:

```
devlink dev eswitch show pci/0000:<pci_id>
```

Sample output of the above command:

```
pci/0000:<pci_id>: mode legacy
```

If the U25N hardware is not in the legacy mode, change it to legacy mode using the following command:

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode legacy
```

Step 3. Perform a factory reset via PF0 network interface of U25N

1. Factory reset is performed using the following CLI command:

```
./u25n_update factory-reset <PF0_interface>
```

E.g: `./u25n_update factory-reset <u25eth0>`

1. After a successful factory reset, the following message is displayed:

```
STATUS: SUCCESSFUL
```

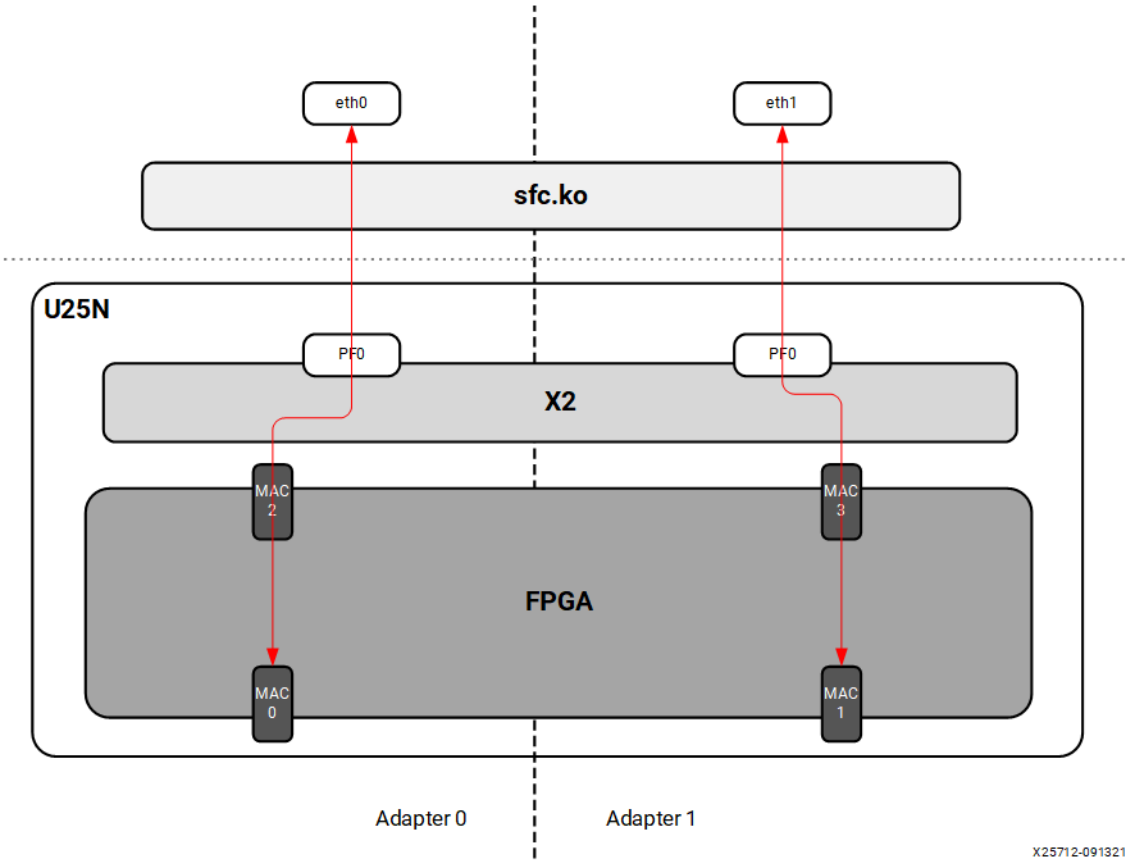
4 Detailed Applications Description

3.1 4.1 Legacy and Switchdev Modes

3.1.1 4.1.1 Legacy NIC (Default)

In legacy mode packets from the external MAC0 are forwarded to the internal MAC2 without any modifications on flow entry miss, and vice versa. Similarly, packets from the external MAC1 are forwarded to the internal MAC3 without any modifications on flow entry miss, and vice versa. OVS is not supported in this configuration.

Figure 5: **Legacy Mode**

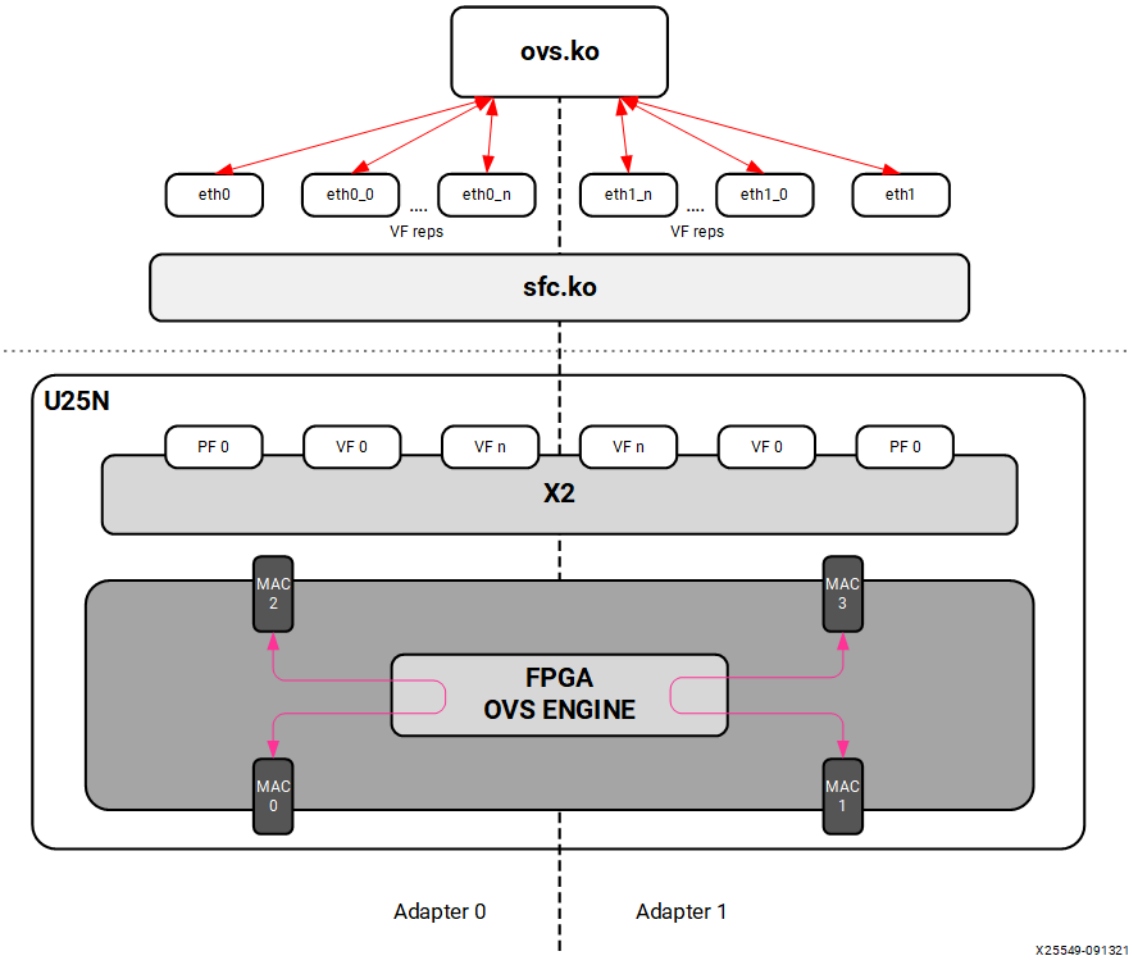


3.1.2 4.1.2 Switchdev Mode

When changed to the switchdev mode, the U25N can support OVS switching. Devlink features are

added to the PF0 interface in each adapter to support the switch mode. Switchdev mode can be added for a single adapter or both. A new representor network interface comes up for each VF when a VM is connected to the VF via SR-IOV virtual ports provided by X2 VNICs.

Figure 6: **Switchdev Mode**



3.2 4.2 OVS

3.2.1 4.2.1 Installing OVS

OVS is a multilayer software switch licensed under the open source Apache 2 license. It implements a production quality switch platform that supports standard management interfaces and opens the forwarding functions to programmatic extension and control. OVS is well suited to function as a virtual switch in virtualized environments. Follow the below-mentioned steps to install OVS.

1. The OVS source code is available in its Git repository, which can be cloned into a directory named "ovs" using the git clone command (see https://github.com/openvswitch/ovs.git).
2. After cloning, the ovs directory will be in the current directory path. "cd" to the ovs directory as mentioned below:

```
cd ovs
```

3. Execute the following commands sequentially as the root user:

```
./boot.sh
./configure
```

```
make -j8
make install
```

4. Export the OVS path:

```
export PATH=$PATH:/usr/local/share/openvswitch/scripts
```

5. Perform a version check:

```
ovs-vswitchd --version
```

***Note*:** Version 2.12 and 2.14 have been tested.

Maximum flows supported and tested: 8k

3.2.2 4.2.2 Classification Fields (Matches)

Supported Keys and Actions

```
Keys

1. ipv4/ipv6 src_ip

2. ipv4/ipv6 dst_ip

3. ip_tos

4. ip_proto

5. ovlan Outer

6. ivlan Inner

7. ether_type

8. tcp/udp src_port

9. tcp/udp dst_port

10. src_mac

11. dst_mac

12. vni

13. Ingress port

Actions

1. do_decap

2. do_decr_ip_ttl

3. do_src_mac

4. do_dst_mac

5. do_vlan_pop

6. do_vlan_push

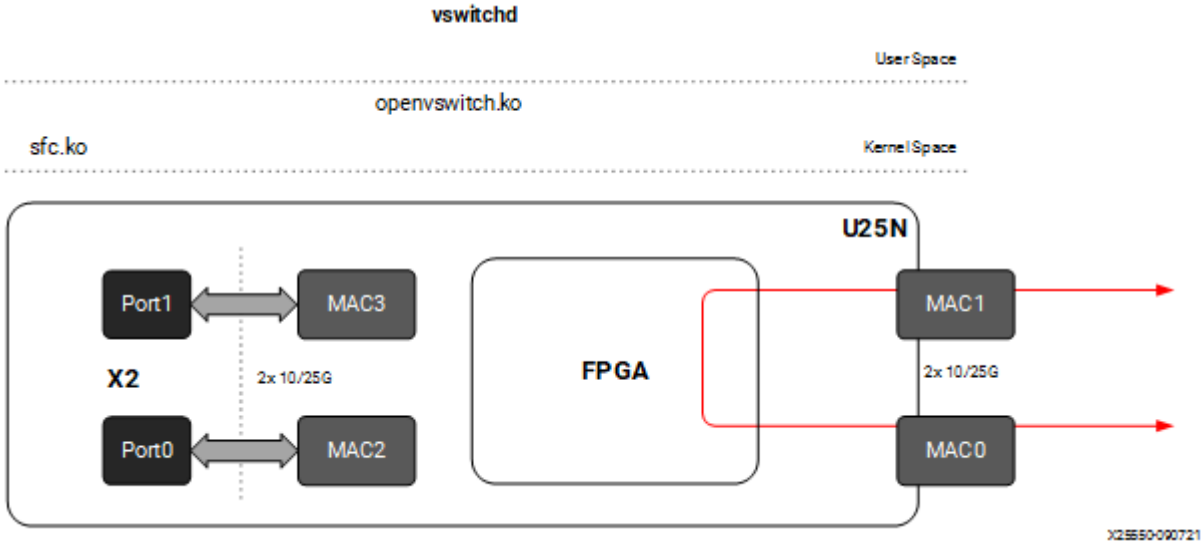
7. do_encap
```

```
8. do_deliver
```

3.2.3 4.2.3 Port to Port

In this configuration the U25N PF is added to the OVS bridge as an interface. Packets are sent to an external MAC, and OVS performs the switching based on the packets received.

Figure 7: **Port to Port setup**



Step 1. Refer to Basic Requirements and Component Versions Supported for the required OS/software version.

Step 2. In case the U25N driver version is 5.3.3.1008.3,this step can be ignored. Driver version of the U25N interface can be validated using the following command:

```
ethtool -i U25_eth0 | grep version
```

***Note*:** To install the latest sfc driver, refer to U25N Driver installation section.

Step 3. Put both U25N PF interfaces in ready state by executing the following command:

1. List the interfaces using the `ifconfig -a` command.
2. Search for U25N interfaces using the `ethtool -i <interface_name>` command:

   ```
   sfboot --list # shows the U25n interfaces
   ifconfig <u25n_interface> up
   ```

   Example Output:

   ```
   ifconfig U25_eth0 up
   ifconfig U25_eth1 up
   ```

Step 4. Put the U25N PF interfaces into switchdev mode by executing the following command:

***Note*:** Ensure that the PF interface link is up before doing switchdev mode.

The `lspci | grep Sol` command gives us the PCIe® device bus ID required to execute the following command.

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode switchdev
```

Example Output:

```
devlink dev eswitch set pci/0000:af:00.0 mode switchdev
```

```
devlink dev eswitch set pci/0000:af:00.1 mode switchdev
```

Step 5. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, continue with the next step.

Step 6. Add external ports to the OVS bridge by executing the following commands:

```
ovs-vsctl add-port br0 <PF interface>
```

For example:

```
ovs-vsctl add-port br0 U25_eth0
ovs-vsctl add-port br0 U25_eth1
```

Step 7. Prints a brief overview of the database contents using the following command:

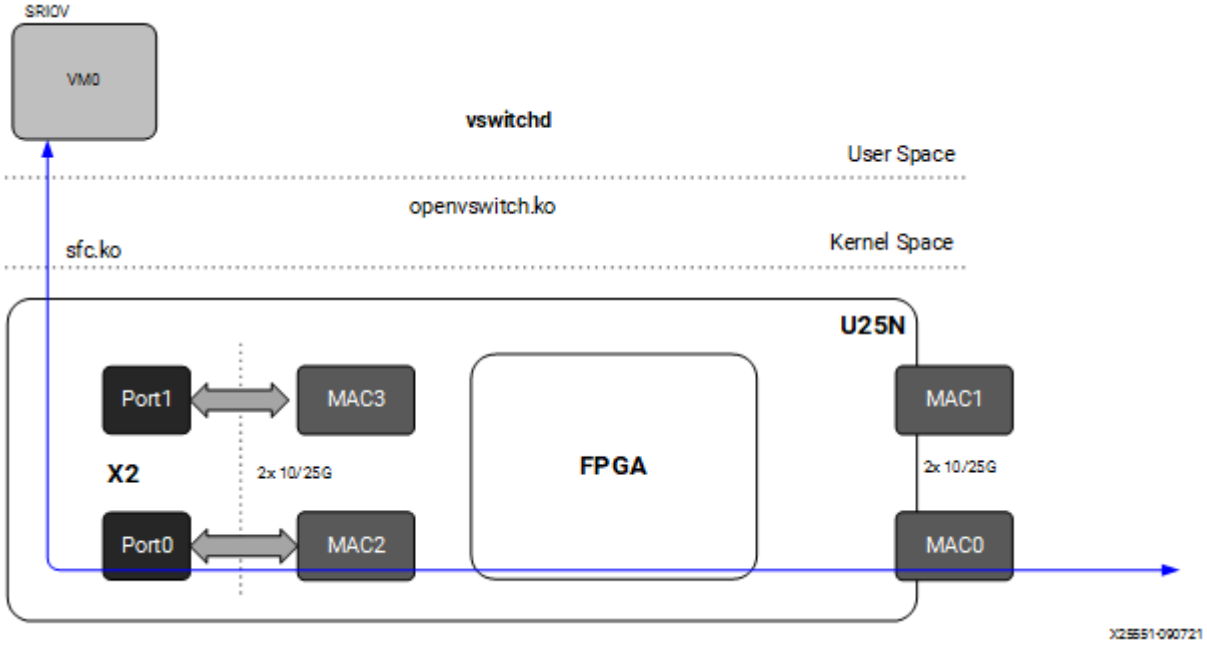
```
ovs-vsctl show
```

Refer to Functionality Check to check the OVS functionality.

3.2.4 4.2.4 Port to VM or VM to Port

***Note*:** To have this configuration SR-IOV must be enabled in BIOS. For the Port to VM or VM to Port configuration, a tunnel L2GRE or VXLAN could be created with two server setups.

Figure 8: **Port to VM or VM to Port setup**



Step 1. Refer to Basic Requirements and Component Versions Supported for the required OS/software version. For VM use cases, VFs need to be created for the corresponding PF for binding to the VM. The number of VF counts should be configured in the sfboot command and in sriov_numvfs.

***Note*:** For more information, refer to sfboot Configuration.

Step 2. Check the driver version of the U25N interface using the following command:

***Note*:** Ignore this step if the U25N driver version is 5.3.3.1008.3

```
ethtool -i U25_eth0 | grep version
```

***Note*:** To install the latest sfc driver, refer to U25N Driver.

***Note*:** Refer to Deployment Image Flashing for flashing images to check OVS functionality.

Step 3. Ensure that U25N PF interfaces are up by running the following commands:

a. List the PF interface using the `ifconfig -a` command.

b. Find the U25N PF interface using the `ethtool -i <interface_name>` command.

For example:

```
ethtool -i <U25_eth0>
```

driver: sfc

version: 5.3.3.1008.3

```
ifconfig <U25_interface> up
```

For example:

```
ifconfig U25eth0 up
```

Step 4. Enable desired VFs to the U25N PF. In the following command, a single VF is enabled on the PF0 interface:

***Note*:** The VF could also be created on the PF1 interface based on the use case. The sriov_numvfs count should be less than or equal to the VF count specified in the sfboot command. The sriov_numvfs should be enabled only in legacy mode. To check the U25N mode, excecute the following steps.

```
devlink dev eswitch show pci/0000:<pci_id>
```

Example Output:

```
pci/0000:af:00.0 mode legacy
```

If not in legacy mode, change to legacy mode using the following command:

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode legacy
echo 1 > /sys/class/net/<interface>/device/sriov_numvfs
```

For example:

```
echo 1 > sys/class/net/U25_eth0/device/sriov_numvfs
```

***Note*:** After executing above mentioned command, a VF PCIe ID and VF interface will get created. The VF PCIe device ID can be listed with the command `lspci -d 1924:1b03`.

An example of the device ID is

```
af:00.2 Ethernet controller: Solarflare Communications XtremeScale SFC9250
10/25/40/50/100G Ethernet Controller (Virtual Function) (rev 01).
This VF PCIe ID is used for binding the VF to a VM.
```

Step 5. The VF interface can be found using the `ifconfig -a` command. To differentiate VF from PF, use the ip link show command. This gives the VF interface ID and VF interface mac address under the PF interface.

Step 6. Ensure that the VF interface is up by executing the following command:

```
   ifconfig <vf_interface> up
```

Step 7. Ensure that the PF interface is in switchdev mode by executing the following command:

***Note*:** Ensure the PF interface link is up before doing switchdev mode.

The `lspci | grep Sol` command gives the PCIe device bus ID.

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode switchdev
```

For example, `devlink dev eswitch set pci/0000:af:00.0 mode switchdev`.

Step 8. Running the above command creates a VF representor interface. The VF representor interface name will be the PF interface name followed by `_0` for the first VF representor and `_1` for the second V representor, and so on.

Note: The total number of VF representor interfaces created are based on the sriov_numvfs value configured.

```
ip link show | grep <PF_interface>
```

For example, `ip link show | grep <u25eth0>`.

***Note*:** Here u25eth0 is the PF interface and u25eth0_0 is the VF representor interface.

Now make the VF representor interface up using the ifconfig command:

```
ifconfig <vf_rep_interface> up
```

Step 9. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed to the next step.

Step 10. Add PF interfaces as ports to the OVS bridge:

```
```bash
ovs-vsctl add-port br0 <x2 interface>
```

For example, `ovs-vsctl add-port br0 U25eth0`.
```

Step 11. Add a VF representor interface as a port to the OVS bridge:

```
```bash
ovs-vsctl add-port <bridge-name> <VF rep interface>
```

For example, `ovs-vsctl add-port br0 U25eth0_0`.
```

Step 12. Ensure that the the OVS bridge is up by running the following command:

```
```bash
ifconfig <bridge-name> up
```
```

Step 13. Print a brief overview of the database contents:

```
```bash
ovs-vsctl show
```
```

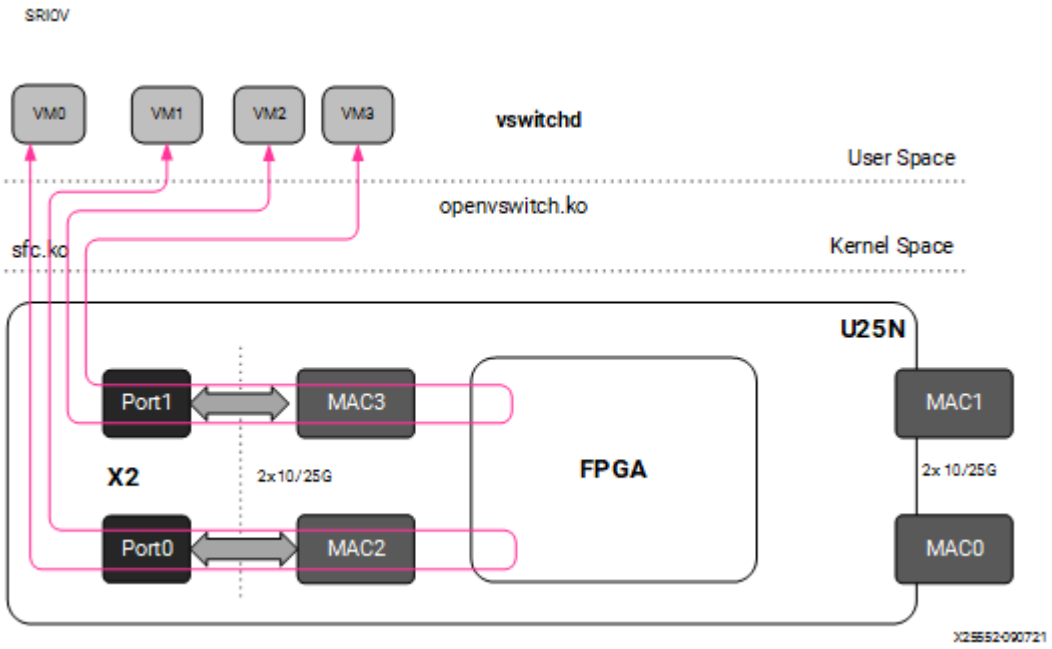
Step 14. Refer VM Installation to instantiate the VM. Post successful instantiation follow the Step 15 to validate its functionality.

Step 15. Refer to Functionality Check to check OVS functionality.

3.2.5 VM to VM

***Note*:** To have this configuration SR-IOV must be enabled in BIOS. For a VM to VM configuration, a tunnel- VXLAN or L2GRE could be created with two server setups.

Figure 9: **VM to VM setup**



Step 1. Refer to U25N Driver for the required OS/software version. For VM use cases, VFs need to be enabled on the corresponding PF and should be bound to the VM. The desired number VFs counts should be configured using sfboot command and in sriov_numvfs. Offload will happen between VMs created using same the PF's VF.

***Note*:** For more information, refer to sfboot Configuration.

Step 2. Validate the driver version of the U25N interface using the following command:

***Note*:** Ignore this step if the U25N driver version is 5.3.3.1008.3

```
ethtool -i U25_eth0 | grep version
```

***Note*:** To install the latest sfc driver, refer to U25N Driver.

***Note*:** Refer Deployment Image Flashing for flashing images to check OVS functionality.

Step 3. Make the U25N X2 PF interface up:

1. List the PF interface using the ifconfig -a command.
2. Find the U25N PF interface using the `ethtool -i <interface_name>` command.
   For example:

   ```
   ethtool -i <U25_eth0>
   ```

   driver: sfc
   version: 5.3.3.1008.3
   `ifconfig <U25_interface> up`
   For example:

   ```
   ifconfig U25eth0 up
   ```

Step 4. Enable the desired VFs on the corresponding PF. In this following commands, two VFs are enabled the PF0 interface:

***Note*:** The VF could also be enabled in the PF1 interface based on the use case. The sriov_numvfs count should be less than or equal to the VF count specified in the sfboot command. The sriov_numvfs should be enabled only in legacy mode. To validate the U25N mode, execute the following steps.

```
devlink dev eswitch show pci/0000:<pci_id>
```

The output of the above command would be:

```
pci/0000:af:00.0 mode legacy
```

In case it is not in legacy mode, change it to legacy mode using the following command:

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode legacy
echo 2 > /sys/class/net/<interface>/device/sriov_numvfs
```

For example:

```
echo 1 /sys/class/net/U25_eth0/device/sriov_numvfs
```

***Note*:** Post executing the above mentioned command, two VF PCIe IDs and two VF interfaces are created. The VF PCIe device ID can be listed with the `lspci -d 1924:1b03` command. An example of the device ID is

```
   af:00.2 Ethernet controller: Solarflare Communications XtremeScale SFC9250
10/25/40/50/100G Ethernet Controller (Virtual Function) (rev 01)* and *af:00.3
Ethernet controller: Solarflare Communications XtremeScale SFC9250
10/25/40/50/100G Ethernet Controller (Virtual Function) (rev 01)*.
These VF PCIe IDs will be used for binding VFs to VMs.
```

Step 5. The two VF interfaces can be found using the `ifconfig -a` command. To differentiate VF from PF, use the `ip link show` command. This gives the VF interface ID and VF interface mac address under the PF interface.

Step 6. Ensure that the two VF interfaces are up by executing the following command:

```
ifconfig <vf_interfaceup>
```

Step 7. Ensure that the PF interfaces are in switchdev mode by executing the following command.

***Note*:** Ensure the PF interface link is up before doing switchdev mode.

The `lspci | grep Sol` command gives the PCIe device bus ID:

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode switchdev
```

For example, `devlink dev eswitch set pci/0000:af:00.0 mode switchdev`.

Step 8. Execution of above mentioned command results in the creation of two VF representor interfaces. The VF representor interface name will have PF interface name followed by `_0` for the first VF representor and `_1` for the second VF representor and so on and so forth.

***Note*:** The number of VF representor interfaces created are based on the sriov_numvfs value configured.

ip link show | grep <PF_interface>

For example, `<ip link show | grep <u25eth0>`.

```
u25eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
mode DEFAULT group default qlen 1000
u25eth0_0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
master ovs-system state UP mode DEFAULT group default qlen 1000
u25eth0_1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
master ovs-system state UP mode DEFAULT group default qlen 1000
```

***Note*:** Here u25eth0 is the PF interface, and u25eth0_0 and u25eth0_1 are the VF representor interfaces.

Now make the VF representor interfaces up using the ifconfig command:

```
ifconfig <vf_rep_interface> up
```

Step 9. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed to the next step.

Step 10. Add two VF representor interfaces to the OVS bridge:

```
```bash
ovs-vsctl add-port <bridge-name> <VF rep interface 1>
ovs-vsctl add-port <bridge-name> <VF rep interface 2>
```

For example:

```
ovs-vsctl add-port br0 u25eth0_0
ovs-vsctl add-port br0 u25eth0_1
```
```

Step 11. Make the OVS bridge up:

```
```bash
ifconfig <bridge-name> up
```
```

Step 12. Print the brief overview of the database contents:

```
```bash
ovs-vsctl show
```
```

Step 13. Refer VM Installation to instantiate the VM. Post VM instantiation, proceed to the next step to validate the functionality.

Step 14. Refer [Functionality Check(link)] to validate OVS functionality.

3.2.6 4.2.6 Tunnels (Encapsulation/Decapsulation)

U25N Smart NIC supports offloading of tunnels using encapsulation and decapsulation actions.

- **Encapsulation:** Pushing tunnel header is supported on TX
- **Decapsulation:** Stripping tunnel header is supported on RX

Supported tunnels:

- VXLAN
- L2GRE

L2GRE

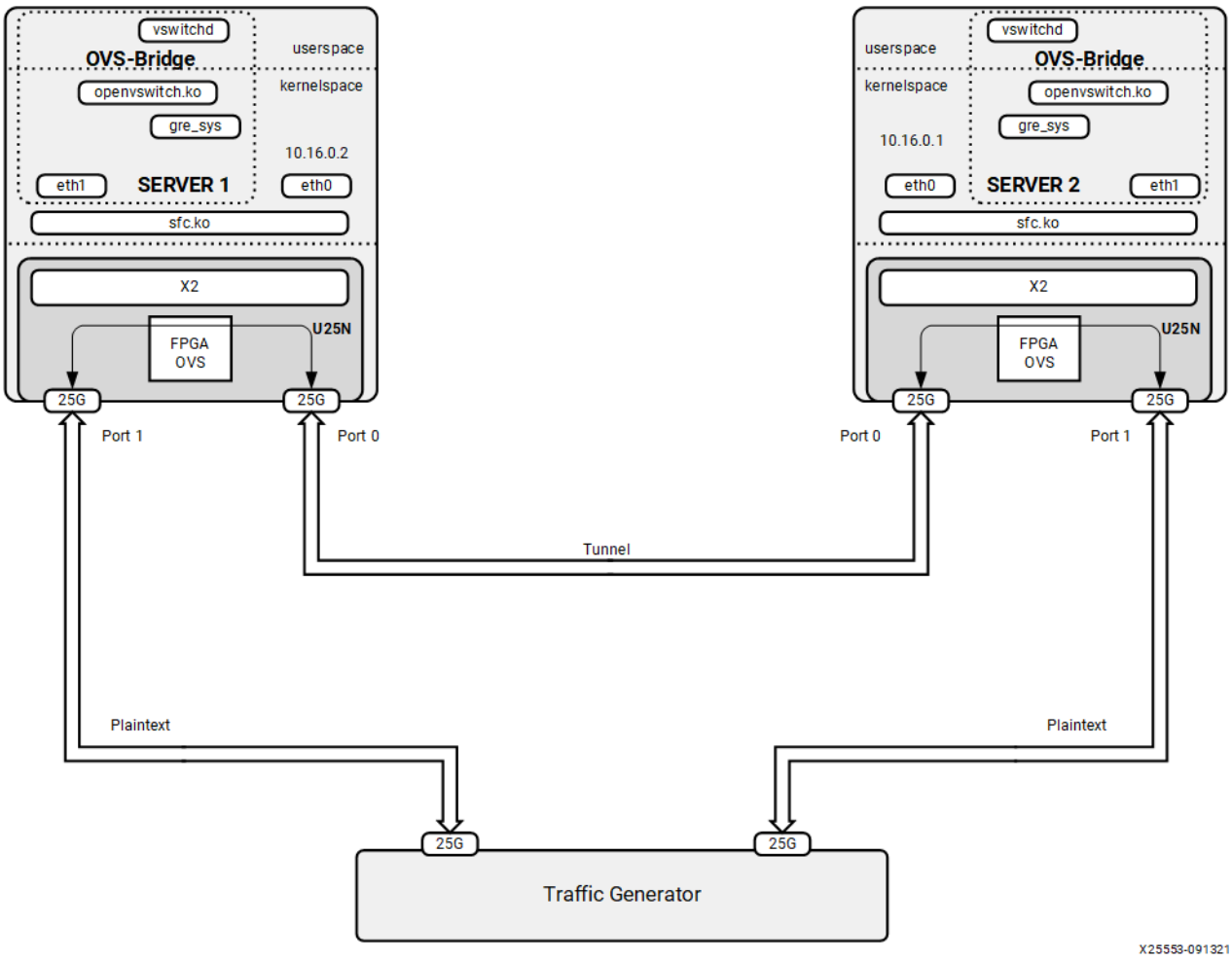
***Note*:** For Port to VM or VM to Port or VM to VM case, a tunnel could be created with two server setup. The following section demonstrates creating a L2GRE based tunnel.

- Maximum tunnel support = 1K
- Maximum supported flows = 8K
- Maximum MTU size = 1400

Refer Basic Requirements and Component Versions Supported for the required OS/software version. An L2GRE tunnel can be formed between two servers. Tunnel endpoint IP should be added to the PF interface which acts as a origin of the tunnel.

***Note*:** Two tunnels could be formed between two PFs of U25N SmartNICs for two different servers for the VM to VM case.

Figure 10: **L2GRE End to End setup with OVS functionality**



Server 1 Configuration

Step 1. Validate the driver version of the U25N interface by executing the following command:

***Note*:** Ignore this step if the U25N driver version is 5.3.3.1008.3.

```
ethtool -i U25_eth0 | grep version
```

***Note*:** To install the latest sfc driver, refer to U25N Driver Installation section.

***Note*:** Refer Deployment Image Flashing for flashing images and to validate OVS functionality.

Step 2. Ensure that the U25N PF interfaces are up:

List the PF interfaces using the `ifconfig -a` command. Find the U25N PF interface using the `ethtool -i <interface_name>` command.

For example:

```
ethtool -i <U25_eth0>
```

driver: sfc

version: 5.3.3.1008.3

```
ifconfig <U25_interface> up
```

For example:

```
ifconfig U25eth0 up
ifconfig U25eth1 up
```

Step 3. Assign tunnel IP to PF0 interface by executing following command:

```
ifconfig <interface_1> <ip> up
```

For example, ifconfig U25eth0 10.16.0.2/24 up.

Step 4. Ensure that the PF0 interface is in switchdev mode.

***Note*:** Make sure the PF0 interface link is up before doing switchdev mode.

The `lspci | grep Sol` command gives the PCIe device bus ID.

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode switchdev
```

For example:

```
devlink dev eswitch set pci/0000:af:00.0 mode switchdev
```

Step 5. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed to the next step.

Step 6. Create GRE interfaces bby executing following commands:

```
ovs-vsctl add-port <bridge_name> gre0 -- set interface gre0 type=gre
options:local_ip=<ip_address> options:remote_ip=<ip_address>
```

For example:

```
ovs-vsctl add-port br0 gre0 -- set interface gre0 type=gre
options:local_ip=10.16.0.2 options:remote_ip=10.16.0.1
```

Step 7. Add a PF1 interface as a port to the OVS bridge:

```
ovs-vsctl add-port <bridge-name> <U25N interface_2>
```

For example:

```
ovs-vsctl add-port br0 U25eth1
```

Step 8. Ensue the OVS bridge is up:

```
ifconfig <bridge_name> up
```

For example:

```
ifconfig br0 up
```

Step 9. Print a brief overview of the database contents:

```
ovs-vsctl show
```

Server 2 Configuration

Step 1. Check the driver version of the U25N interface using this command:

***Note*:** Ignore this step if the U25N X2 driver version is 5.3.3.1008.3

```
ethtool -i U25_eth0 | grep version
```

***Note*:** To install the latest sfc driver, refer to U25N Driver installation section.

Step 2. Ensure that U25N PF interfaces are up:

List the PF interfaces using the `ifconfig -a` command. Find the U25N PF interface using the `ethtool -i <interface_name>` command.

For example:

```
ethtool -i <U25_eth0>
```

driver: sfc

version: 5.3.3.1000

```
ifconfig <U25_interface> up
```

For example:

Step 3. Assign tunnel IP to PF0 interface by executing the following command:

```
ifconfig <interface_1> <ip> up
```

For example:

```
ifconfig U25eth0 10.16.0.1/24 up
```

Step 4. Ensure that PF0 interface is in switchdev mode.

***Note*:** Make sure the PF0 interface link is up before doing switchdev mode.

The `lspci | grep Sol` command gives the PCIe device bus ID.

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode switchdev
```

For example:

```
devlink dev eswitch set pci/0000:af:00.0 mode switchdev
```

Step 5. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed to the next step.

Step 6. Create GRE interfaces:

```
ovs-vsctl add-port <bridge_name> gre0 -- set interface gre0 type=gre
options:local_ip=<ip_address> options:remote_ip=<ip_address>
```

For example:

```
ovs-vsctl add-port br0 gre0 -- set interface gre0 type=gre
options:local_ip=10.16.0.1 options:remote_ip=10.16.0.2
```

Step 7. Add a PF1 interface as a port to the OVS bridge:

```
ovs-vsctl add-port <bridge_name> <U25N interface_2>
```

For example:

```
ovs-vsctl add-port br0 U25eth1
```

Step 8. Ensure that the bridge up:

```
ifconfig <bridge_name> up
```

For example:

```
ifconfig br0 up
```

STep 9. Print a brief overview of the database contents:

```
ovs-vsctl show
```

Step 10. Refer Functionality Check to check OVS functionality.

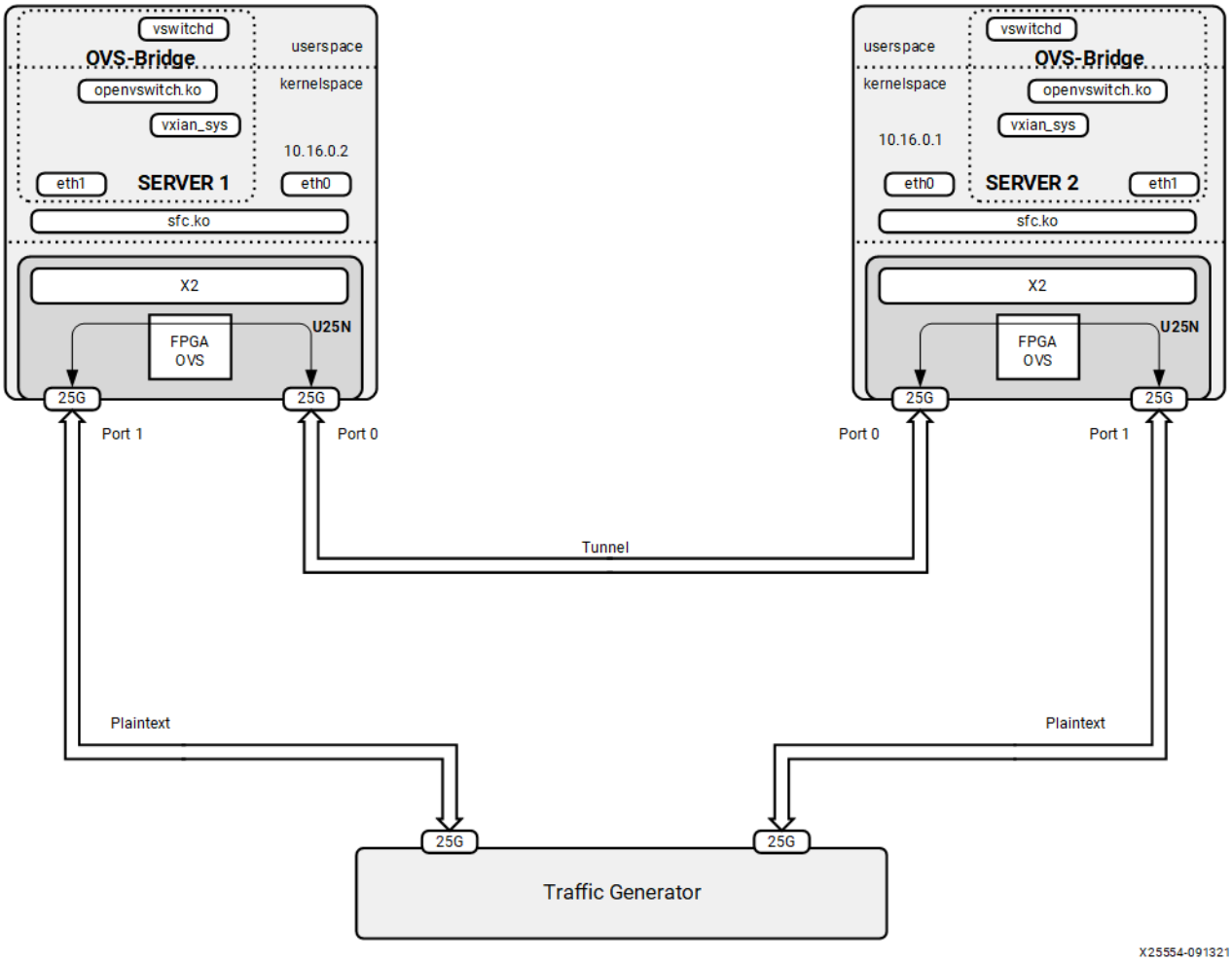
4.2.6.2 VXLAN

***Note*:** For Port to VM or VM to Port or VM to VM configuration, a tunnel could be created with two server setups. The following section demonstrates creating a VXLAN based tunnel.

- Maximum tunnel support = 1K
- Maximum supported flows = 8K
- Maximum MTU size = 1400

Refer Basic Requirements and Component Versions Supported for the required OS/software version. A VXLAN tunnel can be formed between two servers. Tunnel endpoint IP should be added to the PF interface where the tunnel needs to be created.

Figure 11: **VXLAN**



Server 1 Configuration

Step 1. Validate the driver version of the U25N interface using this command:

***Note*:** Ignore this step if the U25N X2 driver version is 5.3.3.1008.3

```
ethtool -i U25_eth0 | grep version
```

***Note*:** To install the latest sfc driver, refer U25N Driver.

***Note*:** Refer to Deployment Image Flashing for flashing images to check OVS functionality.

Step 2. Ensure that the the U25N PF interfaces are up:

List the PF interfaces using the ifconfig -a command. Find the U25N PF interface using the `ethtool -i <interface_name>` command.

For example:

```
ethtool -i <U25_eth0>
```

driver: sfc

version: 5.3.3.1008.3

```
ifconfig <U25_interface> up
```

For example:

```
ifconfig U25eth0 up
ifconfig U25eth1 up
```

Step 3. Assign tunnel IP to PF0 interface:

```
ifconfig <interface_1> <ip> up
```

For example:

```
ifconfig U25eth0 10.16.0.2/24 up
```

Step 4. Ensure that PF0 interface is in switchdev mode.

***Note*:** Make sure the PF0 interface link is up before doing switchdev mode.

The `lspci | grep Sol` command gives the PCIe device bus ID.

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode switchdev
```

For example, `devlink dev eswitch set pci/0000:af:00.0 mode switchdev`.

Step 5. Follow the steps mentioned in OVS Configuration to create an OVS bridge. Post OVS bridge creation, proceed to the next step.

Step 6. Create VXLAN interfaces:

```
ovs-vsctl add-port br0 vxlan0 -- set interface vxlan0 type=vxlan
options:local_ip=<ip_address> options:remote_ip=<ip_address>
options:key=<key_id>
ovs-vsctl add-port br0 vxlan0 -- set interface vxlan0 type=vxlan
options:local_ip=10.16.0.2 options:remote_ip=10.16.0.1 options:key=123
```

Step 7. Add a PF1 interface as a port to the OVS bridge:

```
ovs-vsctl add-port <bridge-name> <U25N interface_2>
```

For example, `ovs-vsctl add-port br0 U25eth1`.

Step 8. Ensure that the bridge is up:

```
ifconfig <bridge_name> up
```

For example:

```
ifconfig br0 up
```

Step 9. Print a brief overview of the database contents:

```
ovs-vsctl show
```

Server 2 Configuration

Step 1. Check the driver version of the U25N X2 interface using this command:

***Note*:** Ignore this step if the U25N driver version is 5.3.3.1008.3

```
ethtool -i U25_eth0 | grep version
```

This should give the output as version 5.3.3.1008.3

***Note*:** To install the latest sfc driver, refer to U25N Driver.

Step 2. Ensure that the U25N PF interfaces are up:

List the PF interfaces using the `ifconfig -a` command. Find the U25N PF interface using the `ethtool -i <interface_name>` command.

For example:

```
ethtool -i <U25_eth0>
```

driver: sfc

version: 5.3.3.1008.3

```
ifconfig <U25_interface> up
```

For example:

```
ifconfig U25eth0 up
ifconfig U25eth1 up
```

Step 3. Assign tunnel IP to PF0 interface:

```
ifconfig <interface_1> <ip> up
```

For example:

```
ifconfig U25eth0 10.16.0.1/24 up
```

Step 4. Ensure that PF0 interface is in switchdev mode.

***Note*:** Make sure the PF0 interface link is up before doing switchdev mode.

The `lspci | grep Sol` command gives the PCIe device bus ID.

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode switchdev
```

For example:

```
devlink dev eswitch set pci/0000:af:00.0 mode switchdev
```

Step 5. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed to the next step.

Step 6. Create VXLAN interfaces:

```
ovs-vsctl add-port br0 vxlan0 -- set interface vxlan0 type=vxlan
options:local_ip=<ip_address> options:remote_ip=<ip_address>
options:key=<key_id>
ovs-vsctl add-port br0 vxlan0 -- set interface vxlan0 type=vxlan
options:local_ip=10.16.0.1 options:remote_ip=10.16.0.2 options:key=123
```

Step 7. Add a PF1 interface as a port to the OVS bridge:

```
ovs-vsctl add-port <bridge_name> <U25N interface_2>
```

For example:

```
ovs-vsctl add-port br0 U25eth1
```

Step 8. Ensure the bridge is up:

```
ifconfig <bridge_name> up
```

For example:

```
ifconfig br0 up
```

Step 9. Print a brief overview of the database contents:

```
ovs-vsctl show
```

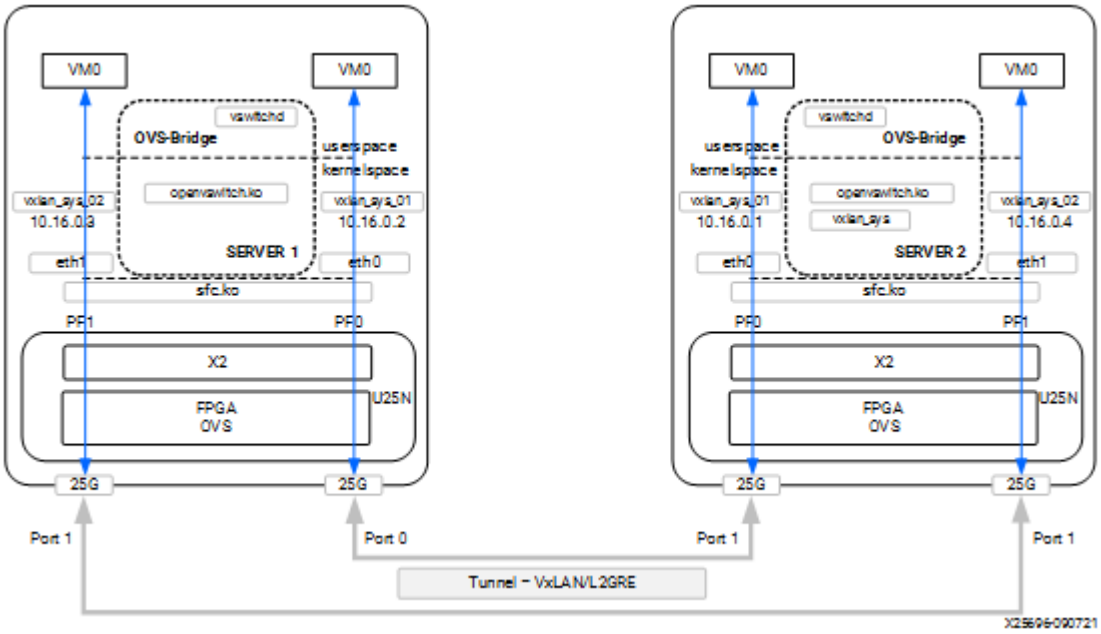
STep 10. Refer to Functionality Check to check OVS functionality.

4.2.6.3 VM to VM or VM to Port or Port to VM Tunnel

- Maximum tunnel support = 1K
- Maximum supported flows = 8K
- Maximum MTU size = 1400

Refer to Basic Requirements and Component Versions Supported. A VXLAN tunnel can be created between two servers. Tunnel endpoint IP should be added to the PF interface whcih acts as the starting point of the tunnel.

Figure 12: **Tunneling/Detunneling setup**



Server 1 Configuration

Step 1. Check the driver version of the U25N interface using this command:

***Note*:** Ignore this step if the U25N X2 driver version is 5.3.3.1008.3

```
ethtool -i U25_eth0 | grep version
```

***Note*:** To install the latest sfc driver, refer to U25N Driver.

***Note*:** Refer to Deployment Image Flashing for flashing images to check OVS functionality.

Step 2. Ensure that the U25N PF interfaces are up:

List the PF interfaces using the `ifconfig -a` command. Find the U25N PF interface using the `ethtool -i <interface_name>` command.

For example:

```
ethtool -i <U25_eth0>
```

Driver: sfc

Version: 5.3.3.1008.3

```
ifconfig <U25_interface> up
```

For example:

```
ifconfig U25eth0 up
ifconfig U25eth1 up
```

Step 3. Assign tunnel IP to PF0 and PF1 interfaces:

```
ifconfig <interface_1> <ip> up
ifconfig <interface_2> <ip> up
```

For example:

```
ifconfig <interface_1> <ip> up
ifconfig <interface_2> <ip> up
```

Step 4. Enable the desired the number of VF on the corresponding PF. Here, a single VF is enabled on each PF0 and PF1 interfaces:

***Note*:** The sriov_numvfs count should be less than or equal to VF count specified in sfboot command. The sriov_numvfs should be enabled only in legacy mode. To validate U25N mode, please follow the below steps.

```
devlink dev eswitch show pci/0000:<pci_id>
```

The output of the above command would be:

```
pci/0000:af:00.0 mode legacy
```

If not in legacy mode, change it to legacy mode using the following command:

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode legacy
echo 1 > /sys/class/net/<interface>/device/sriov_numvfs
```

For example:

```
echo 1 > /sys/class/net/U25_eth0/device/sriov_numvfs
echo 1 > /sys/class/net/U25_eth1/device/sriov_numvfs
```

***Note*:** Post executing the above mentioned command, a VF PCIe ID and a VF interface will get created corresponding to each PF0 and PF1. The VF PCIe device ID can be listed with the `lspci -d 1924:1b03` command. An example of the device ID is `bash af:00.2 Ethernet controller: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet Controller (Virtual Function) (rev 01) and af:00.6 Ethernet controller: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet Controller (Virtual Function) (rev 01). These VF PCIe IDs would be used for`

binding VFs to VMs. bash

Step 5. The two VF interfaces can be found using the ifconfig -a command. To differentiate VF from PF, use the ip link show command. This gives the VF interface ID and VF interface mac address under the PF interface.

Step 6. Ensure that the two VF interfaces up:

```
ifconfig <vf_interface> up
```

Step 7. Ensure that the PF0 and PF1 interfaces are in switchdev mode.

***Note*:** Make sure the PF0 and PF1 interface link is up before doing switchdev mode.

The lspci | grep Sol command gives the PCIe device bus ID:

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode switchdev
```

For example:

```
devlink dev eswitch set pci/0000:af:00.0 mode switchdev
devlink dev eswitch set pci/0000:af:00.1 mode switchdev
```

Step 8. Post execution of the above mentioned command a VF representor interface corresponding to each PF interface will get created. The VF representor interface name will have PF interface name followed by _0.

***Note*:** Here, the number of VF representors created are based on the sriov_numvfs value configured.

```
ip link show | grep <PF_interface>
```

For example, ip link show | grep <u25eth0>.

```
u25eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
mode DEFAULT group default qlen 1000
u25eth0_0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
master ovs-system state UP mode DEFAULT group default qlen 1000
ip link show | grep <u25eth1>
u25eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
mode DEFAULT group default qlen 1000
u25eth1_0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
master ovs-system state UP mode DEFAULT group default qlen 1000
```

***Note*:** Here u25eth0 and u25eth1 are the PF interfaces, and u25eth0_0 and u25eth1_0 are the VF representor interfaces.

Now make the VF representor interfaces up using the ifconfig command:

```
ifconfig <vf_rep_interface> up
```

Step 9. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed to the next step.

Step 10. Creating VXLAN/GRE interfaces:

```
***Note*:** The following configuration is for the VXLAN. Similarly, the GRE
tunnel could also be used.

```bash
ovs-vsctl add-port br0 vxlan0 -- set interface vxlan0 type=vxlan
options:local_ip=<ip_address> options:remote_ip=<ip_address>
options:key=<key_id>
ovs-vsctl add-port br1 vxlan1 -- set interface vxlan0 type=vxlan
options:local_ip=<ip_address> options:remote_ip=<ip_address>
options:key=<key_id>
```
```

```
For example:

```bash
ovs-vsctl add-port br0 vxlan0 -- set interface vxlan0 type=vxlan
options:local_ip=10.16.0.2 options:remote_ip=10.16.0.1 options:key=123
ovs-vsctl add-port br1 vxlan1 -- set interface vxlan type=vxlan
options:local_ip=10.16.0.3 options:remote_ip=10.16.0.4 options:key=456
```
```

Step 11. Execute the following command to add VF representor enabled on each PF interface to a separate OVS bridge.

```
```bash
ovs-vsctl add-port <bridge-name_0> <VF rep interface 1>
ovs-vsctl add-port <bridge-name_1> <VF rep interface 2>
```

For example:

```bash
ovs-vsctl add-port br0 u25eth0_0
ovs-vsctl add-port br1 u25eth1_0
```
```

Step 12. Ensure the the two bridges are up:

```
```bash
ifconfig <bridge_name> up
```

For example:

```bash
ifconfig br0 up
ifconfig br1 up
```
```

Step 13. Print a brief overview of the database contents:

```
```bash
ovs-vsctl show
```
```

Step 14. Refer VM Installation to instantiate the VM.

Server 2 Configuration

Step 1. Check the driver version of the U25N interface using this command:

***Note*:** Ignore this step if the U25N X2 driver version is 5.3.3.1008.3

```
ethtool -i U25_eth0 | grep version
```

This should give the output as version 5.3.3.1008.3

***Note*:** To install the latest sfc driver, refer to U25N Driver.

***Note*:** Refer Deployment Image Flashing for flashing images to check OVS functionality.

Step 2. Ensure that U25N PF interfaces are up:

List the PF interfaces using the `ifconfig -a` command. Find the U25N PF interface using the `ethtool -i <interface_name>`command.

For example:

```
ethtool -i <U25_eth0>
```

Driver: sfc

Version: 5.3.3.1008.3

```
ifconfig <U25_interface> up
```

For example:

```
ifconfig U25eth0 up
ifconfig U25eth1 up
```

Step 3. Assign tunnel IP to PF0 and PF1 interfaces:

```
ifconfig <interface_1> <ip> up
ifconfig <interface_2> <ip> up
```

For example:

```
ifconfig U25eth0 10.16.0.1/24 up
ifconfig U25eth1 10.16.0.4/24 up
```

Step 4. Enable the desired number of VFs to the corresponding PF. Here, one VF is enabled on each PF0 and PF1 interface:

***Note*:** The sriov_numvfs count should be less than or equal to VF count assigned in sfboot command. The sriov_numvfs should be enabled only in legacy mode. To validate mode, please follow the below steps.

```
devlink dev eswitch show pci/0000:<pci_id>
```

The output of the above command would be:

```
pci/0000:af:00.0 mode legacy
```

In case not in legacy mode, change it to legacy mode using the following command:

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode legacy
echo 1 > /sys/class/net/<interface>/device/sriov_numvfs
```

For example:

```
echo 1 > /sys/class/net/U25_eth0/device/sriov_numvfs
echo 1 > /sys/class/net/U25_eth1/device/sriov_numvfs
```

***Note*:** Post execution of the above mentioned command, a VF PCIe ID and a VF interface will get created corresponding to each PF0 and PF1. The VF PCIe device ID can be listed with the `lspci -d 1924:1b03` command. An example of the device ID is af:00.2 Ethernet controller: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet Controller (Virtual Function) (rev 01) and af:00.6 Ethernet controller: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet Controller (Virtual Function) (rev 01). These VF PCIe IDs would be used for binding VFs to VMs.

Step 5. The two VF interfaces can be found using the `ifconfig -a` command. To differentiate VF from PF, use the `ip link show` command. This gives the VF interface ID and VF interface mac address under the PF interface.

Step 6. Ensure that two VF interfaces are up:

```
ifconfig <vf_interface> up
```

Step 7. Ensure that PF0 and PF1 interfaces are in switchdev mode.

***Note*:** Make sure the PF0 and PF1 interface link is up before doing switchdev mode.

The `lspci | grep Sol` command gives the PCIe device bus ID:

```
devlink dev eswitch set pci/0000:<PCIe device bus idm ode switchdev
```

For example:

```
devlink dev eswitch set pci/0000:af:00.0 mode switchdev
devlink dev eswitch set pci/0000:af:00.1 mode switchdev
```

Step 8. Running the above command creates a VF representor interface corresponding to each PF interface. The VF representor interface name will have the PF interface name followed by `_0`.

***Note*:** The total number of VF representor created is based on the sriov_numvfs value configured.

```
ip link show | grep <PF_interface>
```

For example, `ip link show | grep <u25eth0>`.

***Note*:** Here u25eth0 and u25eth1 are the PF interfaces, and u25eth0_0 and u25eth1_0 are the VF representor interfaces.

Ensure the VF representor interfaces are up using the ifconfig command:

```
ifconfig <vf_rep_interface> up
```

Step 9. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed to the next step.

Step 10. Create VXLAN/GRE interfaces:

```
***Note*:** The following configuration is for the VXLAN. Similarly, the GRE
tunnel could also be used.

```
ovs-vsctl add-port br0 vxlan0 -- set interface vxlan0 type=vxlan
options:local_ip=<ip_address> options:remote_ip=<ip_address>
options:key=<key_id>
ovs-vsctl add-port br1 vxlan1 -- set interface vxlan0 type=vxlan
options:local_ip=<ip_address> options:remote_ip=<ip_address>
options:key=<key_id>
```

For example:

```
ovs-vsctl add-port br0 vxlan0 -- set interface vxlan0 type=vxlan
options:local_ip=10.16.0.1 options:remote_ip=10.16.0.2 options:key=123
ovs-vsctl add-port br1 vxlan1 -- set interface vxlan type=vxlan
options:local_ip=10.16.0.4 options:remote_ip=10.16.0.3 options:key=456
```
```

Step 11. Adding VF representor of each PF interface to separate OVS bridges.

```
```bash
ovs-vsctl add-port <bridge-name_0> <VF rep interface 1>
ovs-vsctl add-port <bridge-name_1> <VF rep interface 2>
```

For example:

```
ovs-vsctl add-port br0 u25eth0_0
ovs-vsctl add-port br1 u25eth1_0
```
```

```
```

Step 12. Ensure that the two bridges are up:

```
```bash
ifconfig <bridge_name> up
```

For example:

```
ifconfig br0 up
ifconfig br1 up
```
```

Step 13. Print a brief overview of the database contents:

```
```bash
ovs-vsctl show
```
```

Step 14. Refer to VM Installation to instantiate the VM. Post successful instantiation refer Functionality Check to validate functionality.

3.2.7 4.2.7 OVS Configuration

Step 1. Export the OVS path:

```
export PATH=$PATH:/usr/local/share/openvswitch/scripts
export PATH=$PATH:/usr/local/bin
```

Step 2. Stop OVS and remove any database to get rid of old configurations:

```
ovs-ctl stop
rm /usr/local/etc/openvswitch/conf.db
```

Step 3. Start OVS:

```
ovs-ctl start
```

Step 4. Enable hardware offload:

```
ovs-vsctl set Open_vSwitch . other_config:hw-offload=true
ovs-vsctl set Open_vSwitch . other_config:tc-policy=none
```

Step 5. After adding the hardware offload policy, restart OVS:

```
ovs-ctl restart
```

Step 6. Set OVS log levels (for debug purpose only, if needed):

```
ovs-appctl vlog/set ANY:ANY:dbg
ovs-appctl vlog/set poll_loop:ANY:OFF
ovs-appctl vlog/set netlink_socket:ANY:OFF
```

Step 7. Set the maximum time (in ms) that idle flows remain cached in the datapath:

```
ovs-vsctl set open_vswitch $(ovs-vsctl list open_vswitch | grep _uuid |
cut -f2 -d ":" | tr -d ' ') other_config:max-idle=30000000g
```

Step 8. Print a brief overview of the database contents:

```
ovs-vsctl show
```

The output should be:

```
<git_version>
ovs_version: "<ovs_version>"
```

***Note*:** OVS versions 2.12 and 2.14 have been tested.

Step 9. Adding bridge to OVS:

```
ovs-vsctl add-br <bridge-name>

For Example
ovs-vsctl add-br br0
```

***Note*:** For VM to VM or VM to Port or Port to VM Tunnel alone create two OVS bridges. For example, `ovs-vsctl add-br br0` and `ovs-vsctl add-br br1`.

3.2.8 4.2.8 Functionality Check

After adding the U25N network interfaces to the OVS bridge, the functionality can be verified using ping, iperf, and dpdk network performance tools.

Ping Test

Step 1. Assign the IP address to the respective interface and do a ping using the following command:

```
ping <remote_ip>
```

Step 2. After a successful ping test, iperf can be used:

***Note*:** For VXLAN and L2GRE, set the MTU size to 1400 before running iperf3 or pktgen on a particular interface.

```
ifconfig <interfacemtu 1400 [as root]
```

Step 3. Run iperf3 -s on the host device [iperf server].

Step 4. Run iperf3 -c on a remote device [iperf client].

***Note*:** Refer to DPDK on U25N to run dpdk-testpmd.

3.3 IPsec

3.3.1 4.3.1 Supported XFRM Parameters

IPsec tunnels are created between two servers. Because IPsec is in *transport mode*, L2GRE is used to create tunnels. The strongSwan application runs in user space. The charon plugin of strongSwan is used to offload rules on the U25N. Packets reaching the IPsec module should be L2GRE encapsulated.

- Encryption algorithm: AES-GCM 256 encryption/decryption
- IPsec mode: Transport mode.
- Maximum IPsec tunnel supported: 32

3.3.2 4.3.2 Classification Fields (Matches)

4.3.2.1 Encryption

```
Key

1. IPv4 source address
```

```
2. IPv4 destination address

3. IP4 protocol Action

Actions

1. Action flag

2. SPI

3. Key

4. IV
```

4.3.2.2 Decryption

```
Keys

1. IPv4 source address

2. IPv4 destination address

3. SPI (Security Parameter Index)

Actions

1. Decryption key

2. IV
```

3.3.3 4.3.3 strongSwan Installation

Execute the following commands as a root user:

```
- apt-get install aptitude

- aptitude install opensc

- aptitude install libgmp10

- aptitude install libgmp-dev

- apt-get install libssl-dev
```

***Note*:** Before installing the Debian package for strongSwan, make sure all the dependencies are installed.

Step 1. Refer Basic Requirements and Component Versions Supported for the required OS/software version.

Step 2. Use the following command to validate the version of the strongSwan Debian package. Ignore this step if it shows the version as strongswan_5.8.4

The version can be found using the command `sudo swanctl --version`.

Remove the already installed package before installing the latest one:

```
dpkg -r strongswan_5.8.4-1_amd64
dpkg -i strongswan_5.8.4-1_amd64.deb
```

Step 3. After installing the strongSwan package, create a CA certificate. CA certificate can be created in one server and can be copied to the other server.

4.3.3.1 Server 1 Configuration

Step 1. Generating a self-signed CA certificate using the PKI utility of strongSwan:

```
cd /etc/ipsec.d
ipsec pki --gen --type rsa --size 4096 --outform pem > private/
strongswanKey.pem
ipsec pki --self --ca --lifetime 3650 --in private/strongswanKey.pem --
type rsa --dn "C=CH, O=strongSwan, CN=Root CA" --outform pem > cacerts/
strongswanCert.pem
```

Step 2. After the key and certificate are generated in server 1, copy them to server 2 in the same directory.

```
    a. Copy the file `strongswanKey.pem` present in `/etc/ipsec.d/private/` from
the first server to the second server at the same location.

    b. Copy the file `strongswanCert.pem` presenti n
`/etc/ipsec.d/cacerts/strongswanCert.pem` from the first server to the second
server at the same location.

   After finishing the above, create a key pair and certificate for each server
separately as root.
```

Step 3. Generate the key pair and certificate in server 1 as a root user:

```
cd /etc/ipsec.d
ipsec pki --gen --type rsa --size 2048 --outform pem > private/
client1Key.pem
chmod 600 private/client1Key.pem
ipsec pki --pub --in private/client1Key.pem --type rsa | ipsec pki --
issue --lifetime 730 --cacert cacerts/strongswanCert.pem --cakey private/
strongswanKey.pem --dn "C=CH, O=strongSwan, CN=device1" --san device1 --
flag serverAuth --flag ikeIntermediate --outform pem > certs/
client1Cert.pem
```

Step 4. Configure the conf file and secret file in server 1:

```
sudo vim /etc/ipsec.conf
conn hw_offload #
left=10.16.0.2
right=10.16.0.1
ike=aes256gcm16-sha256-modp2048
esp=aes256gcm16-modp2048
keyingtries=%forever
ikelifetime=8h
lifetime=8h
dpddelay=1h
dpdtimeout=1h
dpdaction=restart
auto=route
keyexchange=ikev2
type=transport
leftcert=client1Cert.pem
leftsendcert=always
hw_offload=yes
leftid="C=CH, O=strongSwan, CN=device1"
rightid="C=CH, O=strongSwan, CN=device2"
leftprotoport=gre
rightprotoport=gre
sudo vim /etc/ipsec.secrets
: RSA client1Key.pem
```

***Note*:** There is a white space, present between : and RSA.

4.3.3.2 Server 2 Configuration

Step 1. Generate the key pair and certificate in server 2 as root:

```
cd /etc/ipsec.d
ipsec pki --gen --type rsa --size 2048 --outform pem > private/
client2Key.pem
chmod 600 private/client2Key.pem
ipsec pki --pub --in private/client2Key.pem --type rsa | ipsec pki --
issue --lifetime 730 --cacert cacerts/strongswanCert.pem --cakey private/
strongswanKey.pem --dn "C=CH, O=strongSwan, CN=device2" --san device2 --
flag serverAuth --flag ikeIntermediate --outform pem > certs/
client2Cert.pem
```

Step 2. Configure the conf file and secret file in server 2:

```
sudo vim /etc/ipsec.conf
conn hw_offload #
left=10.16.0.1
right=10.16.0.2
ike=aes256gcm16-sha256-modp2048
esp=aes256gcm16-modp2048
keyingtries=%forever
ikelifetime=8h
lifetime=8h
dpddelay=1h
dpdtimeout=1h
dpdaction=restart
auto=route
keyexchange=ikev2
type=transport
leftcert=client2Cert.pem
leftsendcert=always
hw_offload=yes
leftid="C=CH, O=strongSwan, CN=device2"
rightid="C=CH, O=strongSwan, CN=device1"
leftprotoport=gre
rightprotoport=gre
sudo vim /etc/ipsec.secrets
: RSA client2Key.pem
```

***Note*:** There is a white space, present between : and RSA.

4.3.3.3 Server 1: Steps to Run IPsec

Step 1. Validate the driver version of the U25N interface using the following command:

```
ethtool -i U25_eth0 | grep version
```

***Note*:** Ignore this step if the U25N driver version is 5.3.3.1008.3

***Note*:** To install the latest sfc driver, refer to U25N Driver. *Note:** Either PF0 or PF1 interface could act as a Tunnel endpoint . In the following example we have assumed PF0 as a tunnel endpoint.

Step 2. Ensure that the PF1 interface is up:

```
ifconfig <interface_2> up
```

For example, `ifconfig U25eth1 up`.

Step 3. Esnure that the PF0 interface is up and assign a tunnel endpoint IP to it:

```
ifconfig <interface_1> <ip> up
```

For example, `ifconfig U25eth0 10.16.0.2/24 up`.

Step 4. Esnure that PF0 interface is in switchdev mode.

***Note*:** Make sure the PF0 interface link is up before doing switchdev mode.

The lspci | grep Sol command gives the PCIe device bus ID.

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode switchdev
```

For example, `devlink dev eswitch set pci/0000:af:00.0 mode switchdev`.

Step 5. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed with the following steps.

Step 6. Create GRE interfaces:

```
ovs-vsctl add-port br0 gre0 -- set interface gre0 type=gre
options:local_ip=<ip_address> options:remote_ip=<ip_address>
ovs-vsctl add-port br0 gre0 -- set interface gre0 type=gre
options:local_ip=10.16.0.2 options:remote_ip=10.16.0.1
```

Step 7. Add the PF1 interface OVS bridge:

```
ovs-vsctl add-port br0 <U25N interface_2>
eg:ovs-vsctl add-port br0 U25eth1
```

Step 8. Print a brief overview of the database contents:

```
ovs-vsctl show
```

Step 9. Ensure that the bridge is up:

```
ifconfig <bridge_name> up
```

For example, `ifconfig br0 up`.

Step 10. Enable ipsec offload in the driver:

```
```
echo 1 >> /sys/class/net/<U25N_interface_1>/device/ipsec_enable
```

For example, `echo 1 >> /sys/class/net/U25eth0/device/ipsec_enable`.
```

Step 11. Start IPsec:

```
```
sudo ipsec restart
```
```

4.3.3.4 Server 2: Steps to Run IPsec

Step 1. Validate the driver version of the U25N interface using the following command:

```
ethtool -i U25_eth0 | grep version
```

***Note*:** Ignore this step if the U25N driver version is 5.3.3.1008.3

***Note*:** To install the latest sfc driver, refer to U25N Driver.

Step 2. Ensure that the PF1 interface up:

```
ifconfig <interface_2> up
```

For example, `ifconfig U25eth1 up`.

Step 3. Ensure that the PF0 interface is up and assign a tunnel IP to it:

```
ifconfig <interface_1> <ip> up
```

For example, `ifconfig U25_eth0 10.16.0.1/24up`.

Step 4. Ensure that the PF0 interface is into switchdev mode:

***Note*:** Make sure the PF0 interface link is up before doing switchdev mode.

The `lspci | grep Sol` command gives us the PCIe device bus ID.

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode switchdev
```

For example, `devlink dev eswitch set pci/0000:af:00.0 mode switchdev`.

Step 5. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed with the following steps.

Steo 6. Create GRE interfaces:

```
ovs-vsctl add-port br0 gre0 -- set interface gre0 type=gre
options:local_ip=<ip_address> options:remote_ip=<ip_address>
ovs-vsctl add-port br0 gre0 -- set interface gre0 type=gre
options:local_ip=10.16.0.1 options:remote_ip=10.16.0.2
```

Step 7. Add PF1 interface OVS bridge:

```
ovs-vsctl add-port br0 <U25N interface_2>
eg:ovs-vsctl add-port br0 U25eth1
```

Step 8. Print a brief overview of the database contents:

```
ovs-vsctl show
```

Step 9. Ensure that the bridge is up.

```
ifconfig <bridge_name> up
```

For example, `ifconfig br0 up`.

Step 10. Enable ipsec offload in the driver:

```
```
echo 1 >> /sys/class/net/<U25N_interface_1>/device/ipsec_enable
```

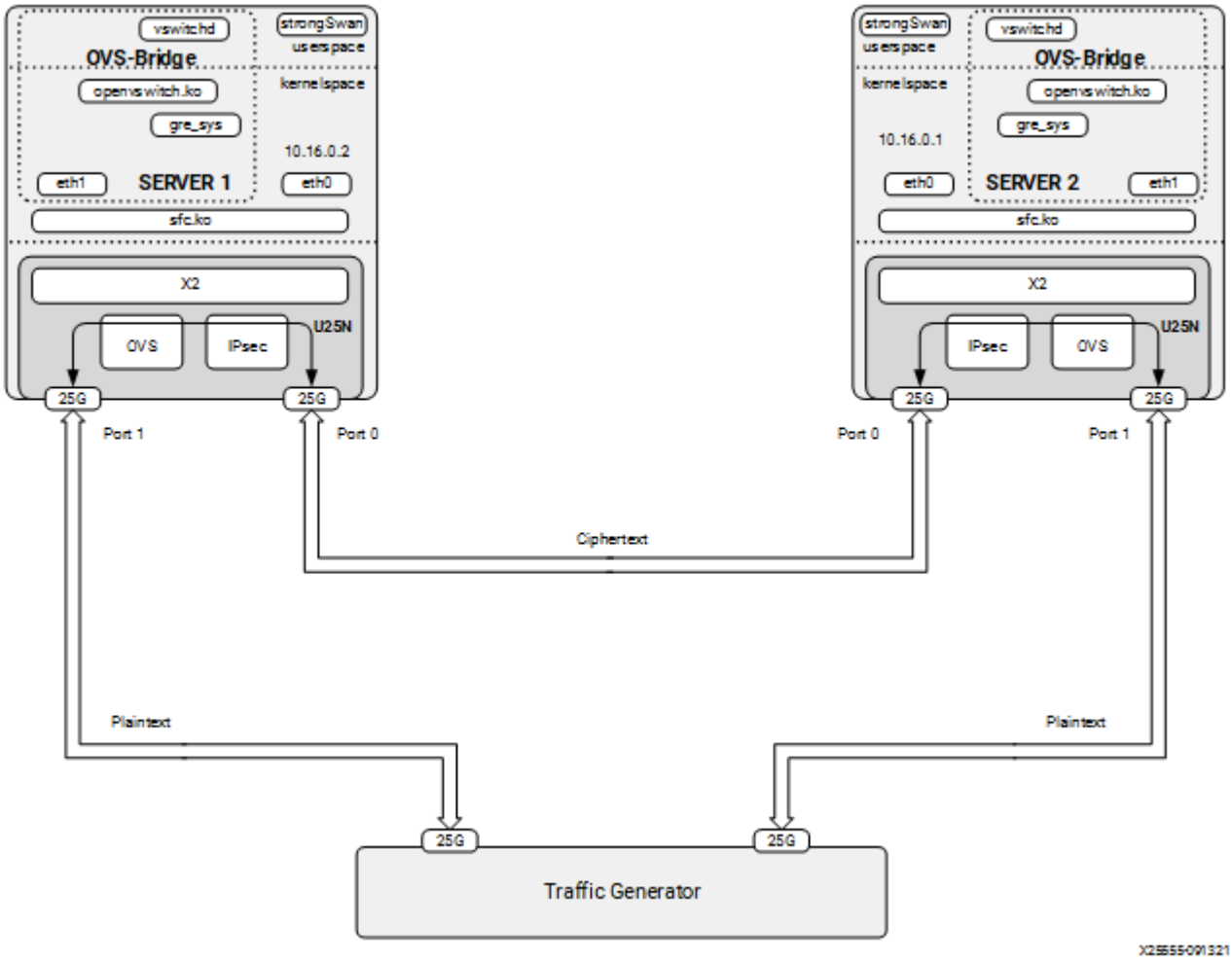
For example, `echo 1 >> /sys/class/net/U25eth0/device/ipsec_enable`.
```

Step 11. Start IPsec:

```
```
sudo ipsec restart
```

Refer [Functionality
Check](./ug1534-detailedappsdescriptions.md#functionality-check) to validate OVS
functionality.
```

Figure 13: **IPsec + OVS End to End setup Diagram**



3.3.4 4.3.5 Changing IPsec Tunnel Endpoints

***Note*:** In the below case, we assumed PF0 interface had previously acted as tunnel endpoint and now we are configuring PF1 to act as tunnel endpoint.

***Note*:** Login as root user before proceeding below steps.

Step 1: Stop the incoming and outgoing traffic.

Step 2: Stop IPsec running on both servers using the below mentioned command.

```
ipsec stop
```

Step 3: Disable ipsec offload in PF interface using the below command.

```
echo 0 > /sys/class/net/<U25N_interface_1>/device/ipsec_enable
```

Eg: `echo 0 > /sys/class/net/U25eth0/device/ipsec_enable`

Step 4: Now enable the IPsec offload in the PF which needs to act as Tunnel Endpoint.

```
echo 1 > /sys/class/net/<U25N_interface_2>/device/ipsec_enable
```

Eg: `echo 1 > /sys/class/net/U25eth1/device/ipsec_enable`

Step 5: Start the IPsec on both servers.

```
ipsec start
```

3.4 4.4 Stateless Firewall

This section is about stateless firewall prerequisites, installation steps, and supported rules.

3.4.1 Kernel Upgrade

Run the command `uname -r` to get the kernel version. If the kernel version is v5.5 or higher, ignore the following steps:

1. Download the Debian packages for the kernel upgrade.
2. Install the dpkg utility using the following command:

```
sudo apt-get update -y
sudo apt-get install -y dpkg
```

3. Unzip the folder using the following command:

```
unzip <folder_name>.zip
```

4. Move inside the folder using the cd command:

```
cd <folder_name>
```

5. Run the following command to install the Debian packages:

```
sudo dpkg -i *.deb
```

6. After the command is executed with no error, do a reboot:

```
sudo reboot
```

3.4.2 nftables

***Note*:** Refer Deployment Image Flashing for flashing images to check firewall functionality.

Step 1. The nftables only work in legacy mode. Ensure the SmartNIC is in legacy mode using the following command:

```
devlink dev eswitch show pci/0000:<pci_id>
```

The output of the above command should be:

```
pci/0000:<pci_id>: mode legacy
```

In case not in legacy mode, change it to legacy mode using the following command:

```
devlink dev eswitch set pci/0000:<PCIe device bus id> mode legacy
```

Step 2. Refer Basic Requirements and Component Versions Supported for the required OS/software version.

The nftables version can be found using the command `nft -v`.

nftables version ≥ v0.9.6

***Note*:** Tested version 0.9.6 and 0.9.8.

3.4.3 4.4.3 Classification Fields (Matches)

Maximum rules supported: 1K for each U25N PF interface.

```
Keys:

1. IPv4/IPv6 source address

2. IPv4/IPv6 destination address

3. Protocol (TCP/UDP)

4. Src_port

5. Dst_port

6. Chain

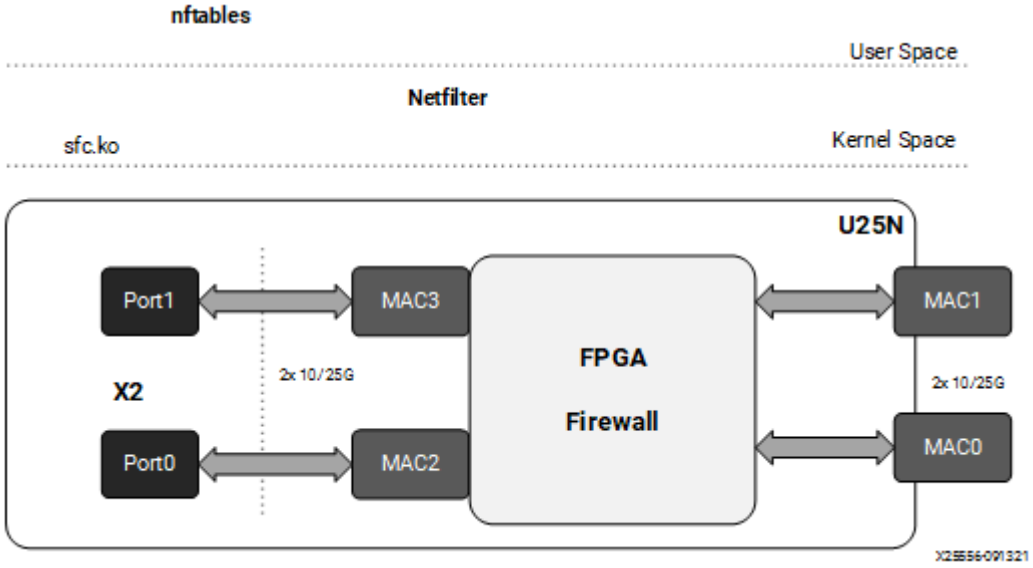
7. Interface Action

Actions

1. drop

2. accept
```

Figure 14: **Firewall**



Driver Installation

***Note*:** Log in as a root user before proceeding with the following steps.

The prerequisites for the driver installation are as follows:

- modprobe mtd (first time only)
- modprobe mdio (first time only)

Step 1. If the debian package already exists, remove the existing package before installing the latest debian package:

```
rmmod sfc
dpkg -r sfc-dkms
dpkg -i sfc-dkms_5.3.3.2000_all.deb
modprobe sfc
```

***Note*:** Login as root user before proceeding with the following steps.

Step 2. Creating a table

```
nft add table <family> <name>
```

For example, `nft add table netdev filter`.

Step 3. Creating a chain:

For example, `nft add chain netdev filter input1 { type filter hook ingress device ens7f1np1 priority 1; flags offload ;}`. Adding a chain without specifying the policy leads to the default policy Accept.

Step 4. Adding rules to the chain:

```
nft add rule <family> <table name> <chain name> ip saddr <ip> drop
```

For example, `nft add rule netdev filter input1 ip saddr 1.1.1.1 drop`.

Step 5. Commands for listing tables, chain, and rules:

a. Listing a table of a netdev family:

```
  ```
  nft list tables <family>
  ```

  For example, `nft list tables netdev`.
```

b. Listing a particular chain from a table:

```
  ```
  nft list chain <family> <table name> <chain_name>
  ```

  For example, `nft list chain netdev filter input1`.
```

c. Listing a chain along with a handle:

```
  ```
  nft -a list chain <family> <table name> <chain_name>
  ```

  For example, `nft -a list chain netdev filter input1`.
```

d. Listing all tables, chains, and rules with handle:

```
  ```
  nft -a list ruleset
  ```
```

Step 6. Commands for deleting tables, chains and rules:

a. Deleting a table:

```
  ```
  nft delete table <family> <name>
  ```

  For example, `nft delete table netdev filter`.
```

b. Deleting a chain:

```
  ```
  nft delete chain <family> <table name> <chain name>
  ```
```

```
  For example, `nft delete chain netdev filter input1`.
```

c. Deleting a specific rule with a handle:

```
  ```
  nft delete rule <family> <table name> <chain_name> handle <handle_no>
  ```

  For example, `nft delete rule netdev filter input1 handle 3`.

  ***Note*:** Here the handle number for a specific rule could be found using the
`nft -a list ruleset` command.
```

3.5 4.5 Statistics

This section outlines the commands used by different modules to check the statistics and packet counters.

3.5.1 OVS Commands

1. To print a brief overview of the database contents:

```
ovs-vsctl show
```

1. To show the datapath flow entries:

```
ovs-ofctl dump-flows <bridge_name>
```

1. To show the full OpenFlow flow table, including hidden flows, on the bridge:

```
ovs-appctl dpctl/dump-flows type=offloaded
```

1. To show the OVS datapath flows:

```
ovs-dpctl dump-flows
ovs-appctl dpctl/dump-flows
```

1. To show which flows are offloaded or not:

```
tc filter show dev <iface_name> ingress
```

3.5.2 4.5.2 MAE Rules

1. Use the following command to display the rules present in the match-action engine (MAE):

```
cat /sys/kernel/debug/sfc/<if_iface>/mae_rules
```

***Note*:** Here `if_iface` should be the corresponding PF interface.

For example, `cat /sys/kernel/debug/sfc/if_u25eth0/mae_rules`.

1. Use the following command to display the default rules present in the MAE:

```
cat /sys/kernel/debug/sfc/<iface/mae_default_rules
```

3.5.3 4.5.3 IPsec Statistics

1. IPsec stats can be obtained by executing the following command:

- `sudo swanctl --stats`

1. Use the `ip xfrm show` command to display IPsec offload security association.

3.6 4.6 Debug Commands

***Note*:** The output of the following commands should be saved for debug purposes.

1. lsmod - It displays which kernel modules are currently loaded. Whether the sfc driver is inserted or not can be verified by the below command.

```
lsmod | grep sfc
```

```
Expected result:
```

sfc 704512 0 sfc_driverlink 16384 1 sfc virtual_bus 20480 1 sfc mtd 65536 14 cmdlinepart,sfc mdio 16384 1 sfc

```

```

1. To get kernel logs, enter the following command.

```
dmesg
```

This command will help to understand all the actions performed by the driver and if there are any crashes happening due to the driver.

```
lspci
```

To display information about all PCI buses and devices in the system. It will also show the network cards inserted in the system with details like driver in use, pci id etc. For Example:

1. lspci | grep Solarflare - Lists the info regarding solarflare devices in the system lspci -k | grep Solarflare - Lists the subsystem information also.lspci -vvv -s

Expected result:

```
lspci | grep Solarflare:
3b:00.0 Ethernet controller: Solarflare Communications XtremeScale SFC9250
10/25/40/50/100G Ethernet Controller (rev 01)

3b:00.1 Ethernet controller: Solarflare Communications XtremeScale SFC9250
10/25/40/50/100G Ethernet Controller (rev 01)

lspci -k | grep Solarflare:

3b:00.0 Ethernet controller: Solarflare Communications XtremeScale SFC9250
10/25/40/50/100G Ethernet Controller (rev 01)
Subsystem: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet
```

```
 Controller

3b:00.1 Ethernet controller: Solarflare Communications XtremeScale SFC9250
10/25/40/50/100G Ethernet Controller (rev 01)
Subsystem: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet
 Controller
```

1. Logs generated by U25N hardware are saved to a file.The logs are collected from the internal processing subsystem and exported to the host at frequent intervals. Currently these logs are populated when switchdev mode is enabled.

Path to read the logs in host : `/var/log/ps_dmesg.txt`

1. `sfreport` - A command line utility that generates a diagnostic log file providing diagnostic data about the server and Solarflare adapters. Please refer to SF-103837-CD Solarflare Server Adapter User Guide chapter 5.20 for more details.
2. `top` - This command can be used to show the linux processes or threads. It provides a real time view of the running system. It can be used to detect memory leaks. The file in the linux path /proc/meminfo can also be used for detecting memory leaks.

Watch out for the total memory already in use. Memory leak can be identified by running the command multiple times and checking whether the memory usage keeps on increasing.

1. `ps` - This command displays relevant information about active processes.Example: ps -aux | grep sfc
2. `ethtool` - This command can be used to understand the driver related information such as version, firmware info and the enabled features. ethtool -i <interface_name>Example Usage: ethtool -i enp59s0f0np0

```
Expected result:
driver: sfc
version: 5.3.3.2000.1
firmware-version: 7.8.7.1005 rx0 tx0
expansion-rom-version:
bus-info: 0000:3b:00.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: no
supports-register-dump: yes
supports-priv-flags: yes
```

To get the information of the state of protocol offload and other features ethtool -k <interface_name> Example Usage: ethtool -k enp59s0f1np1

```
Expected result:
Features for enp59s0f1np1:
rx-checksumming: on
tx-checksumming: on
    tx-checksum-ipv4: on
    tx-checksum-ip-generic: off [fixed]
    tx-checksum-ipv6: on
    tx-checksum-fcoe-crc: off [fixed]
    tx-checksum-sctp: off [fixed]
scatter-gather: on
    tx-scatter-gather: on
    tx-scatter-gather-fraglist: off [fixed]
tcp-segmentation-offload: on
    tx-tcp-segmentation: on
    tx-tcp-ecn-segmentation: on
    tx-tcp-mangleid-segmentation: off
```

```
   tx-tcp6-segmentation: on
generic-segmentation-offload: on
generic-receive-offload: on
large-receive-offload: off
-------------------------------------and so on.
```

To change the offload parameters and other features of the network device sudo ethtool -K <interface_name> <on/off>

To get information about NIC Statistics sudo ethtool -S <interface_name> Example usage: sudo ethtool -S enp59s0f1np1

```
Expected result:
NIC statistics:
     rx_noskb_drops: 0
     rx_nodesc_trunc: 0
     port_tx_bytes: 59052
     port_tx_packets: 353
     port_tx_pause: 0
     port_tx_control: 0
     port_tx_unicast: 0
     port_tx_multicast: 273
     port_tx_broadcast: 80
     port_tx_lt64: 0
     port_tx_64: 0
     port_tx_65_to_127: 229
     port_tx_128_to_255: 44
     port_tx_256_to_511: 80
     port_tx_512_to_1023: 0
     port_tx_1024_to_15xx: 0
     port_tx_15xx_to_jumbo: 0
     port_rx_bytes: 0
     port_rx_good_bytes: 0
     port_rx_bad_bytes: 0
     port_rx_packets: 0
--------------------------------------- and so on.
```

NOTE: ethtool functionalities for sfc driver can also be realised using the sfctool utility also. Eg: sudo sfctool -S <interface_name> Example Usage: sudo sfctool -S enp59s0f1np1

1. u25n_update Application: u25n_update utility can be used to read the U25N shell version. Eg:

```
./utils/u25n_update get-version <PF0_interface>
```

10.MCDI Logging - Mcdi request and response data will be visible in dmesg if we activate mcdi logs. To activate the logs in dmesg:echo 1 >> /sys/class/net/<interface_name>/device/mcdi_logging

1. OVS log levels - It can be turned on/off using the ovs-appctl commands. The ovs-vswitchd accepts the option –log-file[=file] to enable logging to a specific file. The file argument is actually optional, so if it is specified, it is used as the exact name for the log file. The default is used if the file is not specified. Usually the default is /usr/local/var/log/openvswitch/ovs-vswitchd.log Setting OVS Log levels:

```
ovs-appctl vlog/set ANY:ANY:dbg
ovs-appctl vlog/set poll_loop:ANY:OFF
ovs-appctl vlog/set netlink_socket:ANY:OFF
```

1. Mae Rules - To see the offloaded rules available in the MAE, check the below file:

```
cat /sys/kernel/debug/sfc/if_<interface_name>/mae_rules
```

To check the default rules in MAE, check the below file:

```
cat /sys/kernel/debug/sfc/if_<interface_name>/mae_default_rules
```

1. iperf3 - Iperf is a tool for network performance measurement and tuning. It is a cross-platform tool that can produce standardized performance measurements for any network. It has client and server functionality, and can create data streams to measure the throughput between the interfaces. After setting proper ip addresses: Server: iperf3 -s Client : iperf -c <ip_addr_interface>
2. tcpdump - Tcpdump is a packet sniffing and packet analyzing tool meant for System Administrators to troubleshoot connectivity issues in Linux. For example it can capture the packets coming to the network interface using the below command.

```
tcpdump –i <interface_name>
```

DPDK

DPDK compilation steps

1. Download the dpdk git repositories from http://www.dpdk.org/browse.
2. Extract the downloaded dpdk files and download the required packages:

```
apt-get install libnuma-dev liblua5.3-dev libpcap-dev [for ubuntu]
```

3. Create the DPDK build tree:
   Follow the steps mentioned at Compilation Steps to do DPDK compilation.
4. Post successful compilation, allocate hugepage and bind the PCI device to run testpmd:

```
   a. `mkdir /mnt/huge`

   b. `mount -t hugetlbfs nodev /mnt/huge`
```

```
  You can use VFIO or UIO [uio_pci_generic or igb_uio]. Refer to the following
link to install the Linux drivers for DPDK:
[https://doc.dpdk.org/guides/linux_gsg/linux_drivers.html](https://doc.dpdk.org/guides/linux_gsg

  Added below the example commands with igb_uio driver. For the igb_uio driver,
the path for dpdk-kmod can be found at
[http://git.dpdk.org/dpdk-kmods](http://git.dpdk.org/dpdk-kmods). After
downloading, go to the following path and compile:

  ```
  cd <dpdk-kmod>/src
  ```
```

c.

```
`modprobe uio`
```

d.

```
`insmod /<dpdk-kmod>/src/igb_uio.ko`
```

e. Ensure that the VF interface is down and unbound from sfc before binding it to the Linux driver:

```
  ```
  ifconfig <U25eth0_VF> down
  . /usertools/dpdk-devbind.py -u <pci_id> [inside dpdk directory]
  ```
```

f. Run the following command:

```
```
. /usertools/dpdk-devbind.py -b igb_uio <pci_id> [inside dpdk directory]
```

For example:

```
./usertools/dpdk-devbind.py -b igb_uio af:00.0
```
```

1. Command to run testpmd:
   Refer to https://doc.dpdk.org/guides/testpmd_app_ug/run_app.html for more information about the testpmd runtime command. The following is an example command to run testpmd:
   ***Note*:** Make sure the dpdk-testpmd for specific PCI devices is running on the same NUMA node. Use the following command to get the numa node for a specific PCI device.

   ```
   cat /sys/bus/pci/devices/0000\:<pci device id>/numa_node
   ```

   The cores for the specific numa node could be found using the following command:

   ```
   lscpu | grep NUMA
   ```

   For example, `lscpu | grep NUMA`.

   ```
   NUMA node(s): 2
   NUMA node0 CPU(s): 0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30
   NUMA node1 CPU(s): 1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31
   ```

APPENDIX A U25N Shell Programming

To program the U25N flash, an Alveo™ programming cable is required. Refer to *Alveo Programming Cable User Guide* (UG1377) to connect the Alveo programming cable to the U25N maintenance connector. The Vivado® tools must be installed on the server to which the Alveo programming cable's USB port is connected. For more information, see Vivado ML Overview.

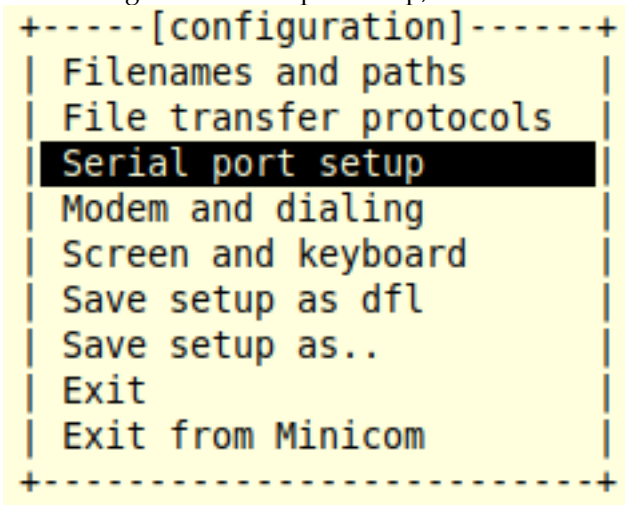
5.1 Entering into U-Boot Mode

***Note*:** There is no need to performing the first step when flashing the shell image for the first time.

1. Connect a USB cable to the SmartNIC.
   a. Open a command line terminal and run the minicom command with sudo.
   b. Before running minicom, verify the serial port setup configuration using the following steps:
   i. Pull up the settings using the -s option.

   ```
   sudo minicom -s
   ```

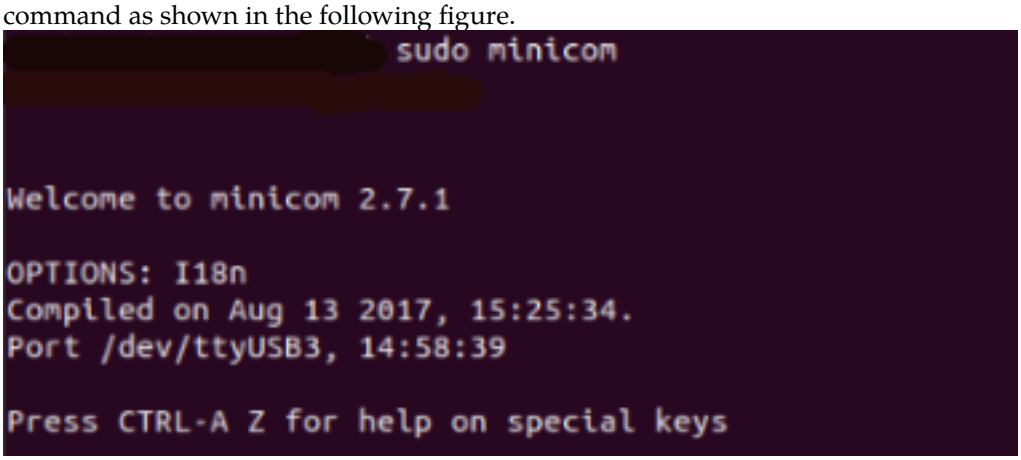
   This should bring up a colorful display listing the different settings.
   ii. To configure the serial port setup, arrow down to **Serial port setup** and press **Enter**.

   iii. To modify the different configurations, press the key corresponding to the setting. For example, press **A** to modify the path to the serial device. Press **Enter** to save the parameters for the setting.

```
+-----------------------------------------------------------------------
| A -    Serial Device      : /dev/ttyUSB3
| B - Lockfile Location     : /var/lock
| C -   Callin Program      :
| D -  Callout Program      :
| E -    Bps/Par/Bits       : 115200 8N1
| F - Hardware Flow Control : No
| G - Software Flow Control : No
|
|    Change which setting?
+-----------------------------------------------------------------------
```

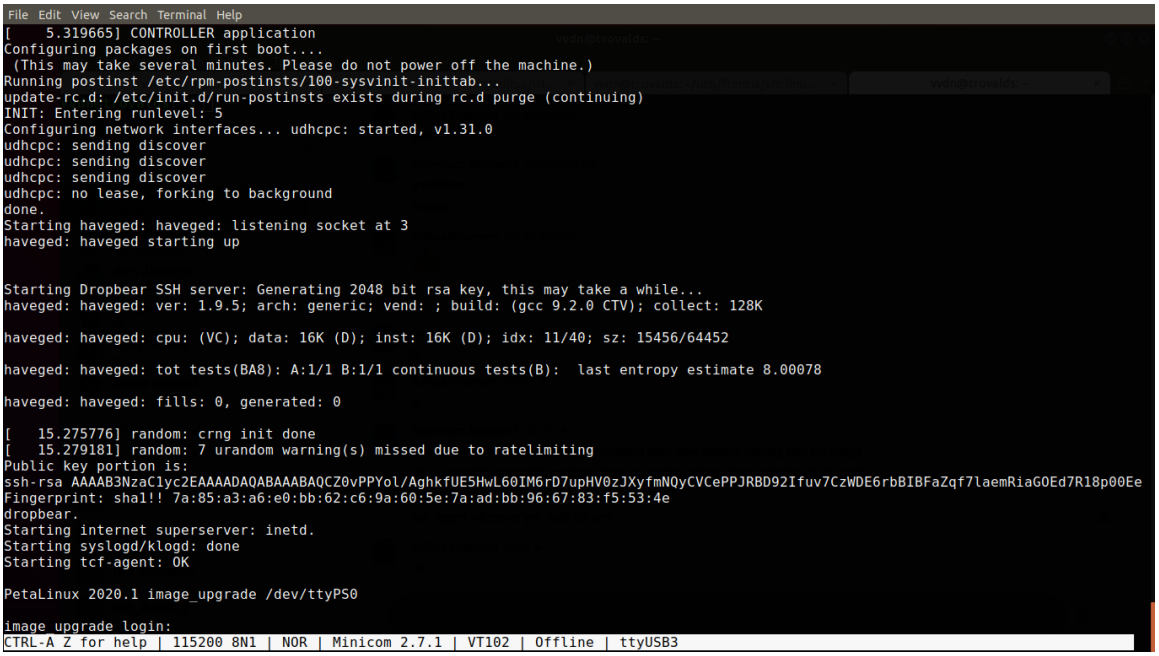
Make sure the above mentioned configuration is applied for E,F,G fields.
iv. After you have saved the configuration, arrow down and select **Exit** to exit from minicom.
v. Some logs will appear after executing the minicom command. These can be ignored. Run the command as shown in the following figure.



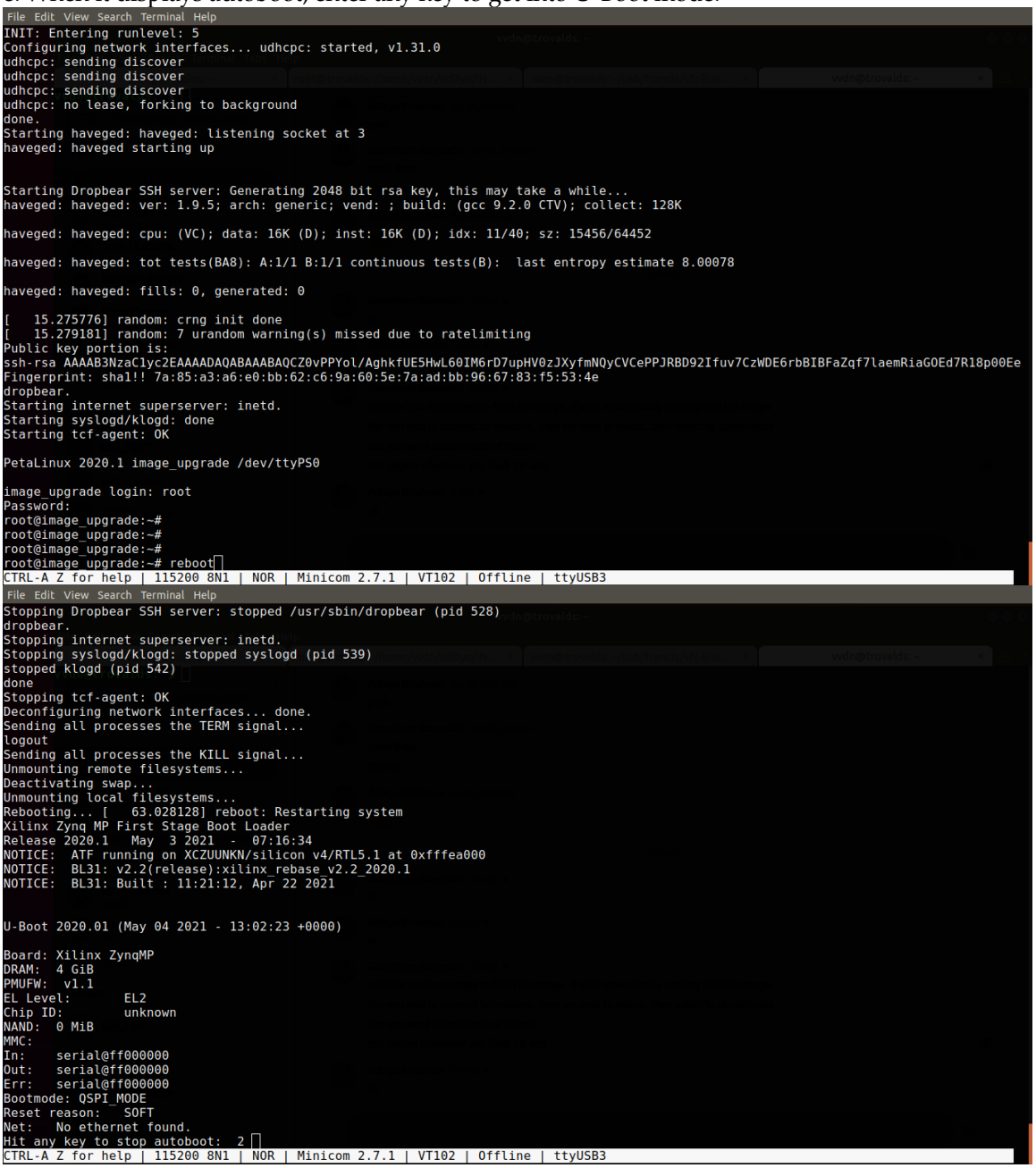
c. When prompted for a login and password, enter the following:

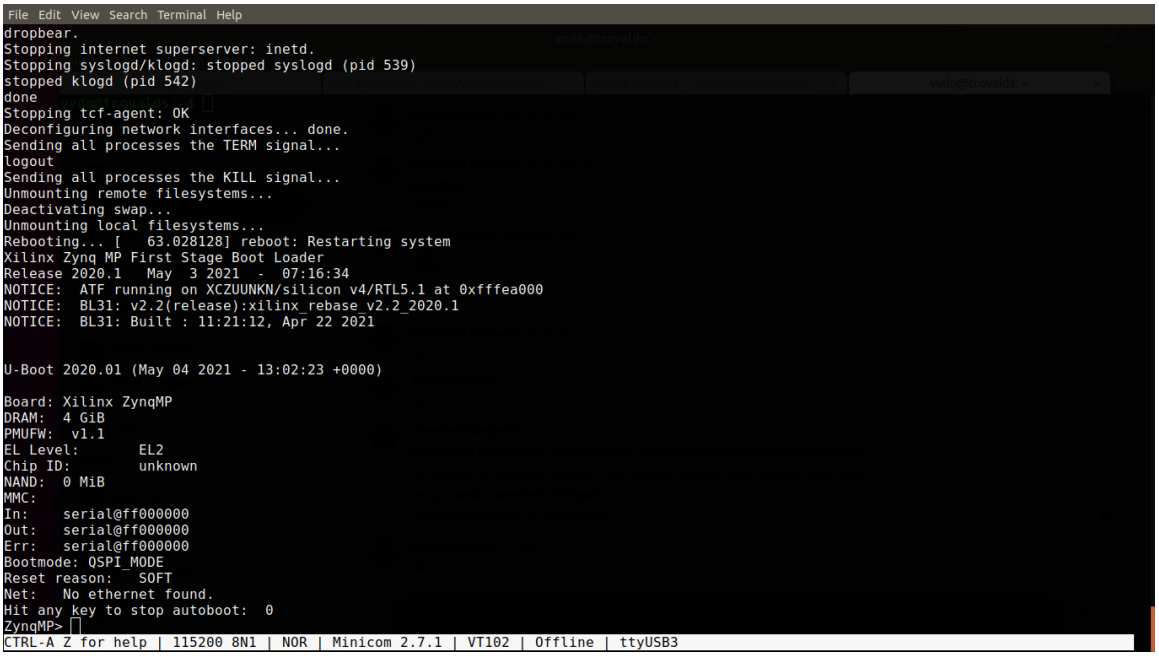
```
- login: root

- password: root
```



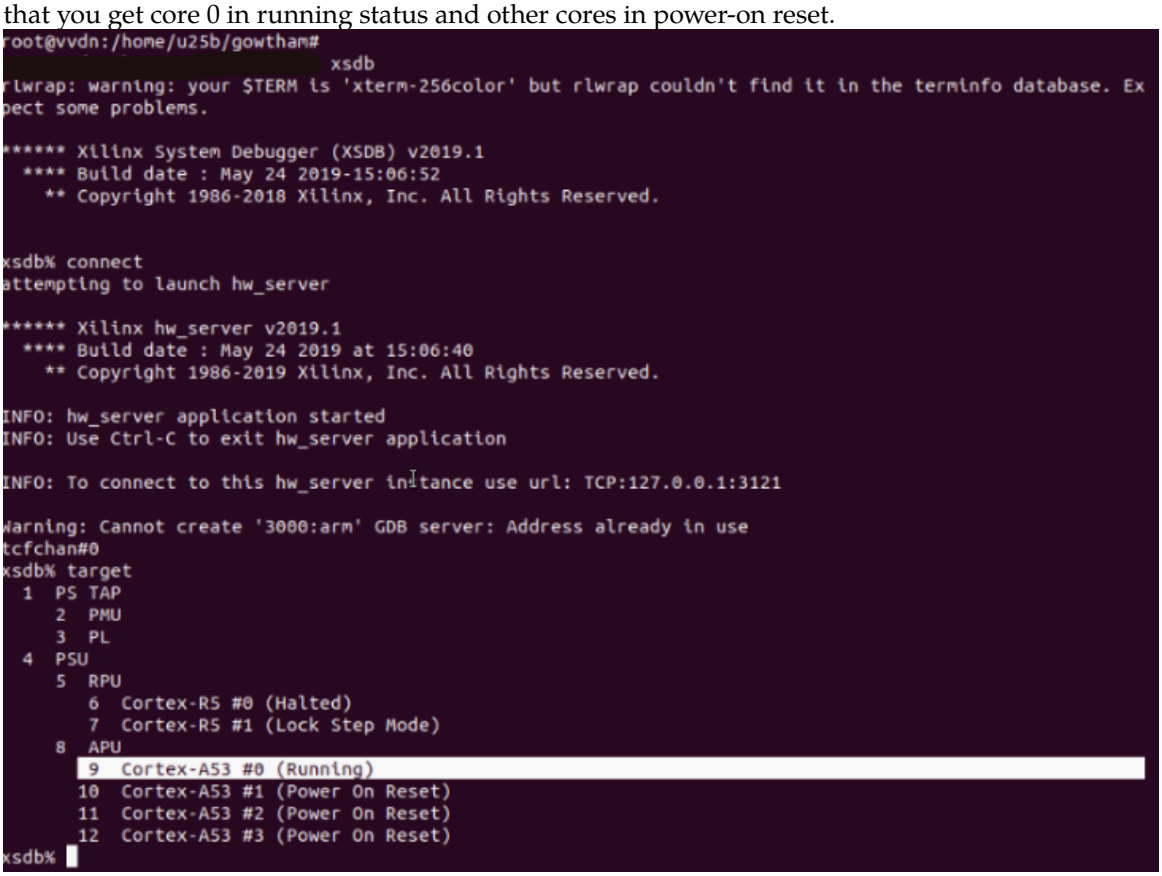
d. After logging in, do a reboot.
The system starts executing the reboot command.
e. When it displays autoboot, enter any key to get into U-Boot mode.





Close the terminal after the above state occurs.

2. As a root, run the xsdb command from the Vivado tools directory in the <install_directory/Vivado_Lab/2019.2/bin> path.
   a. Type the `connect` command and check the target using the `target` command. Make sure that you get core 0 in running status and other cores in power-on reset.



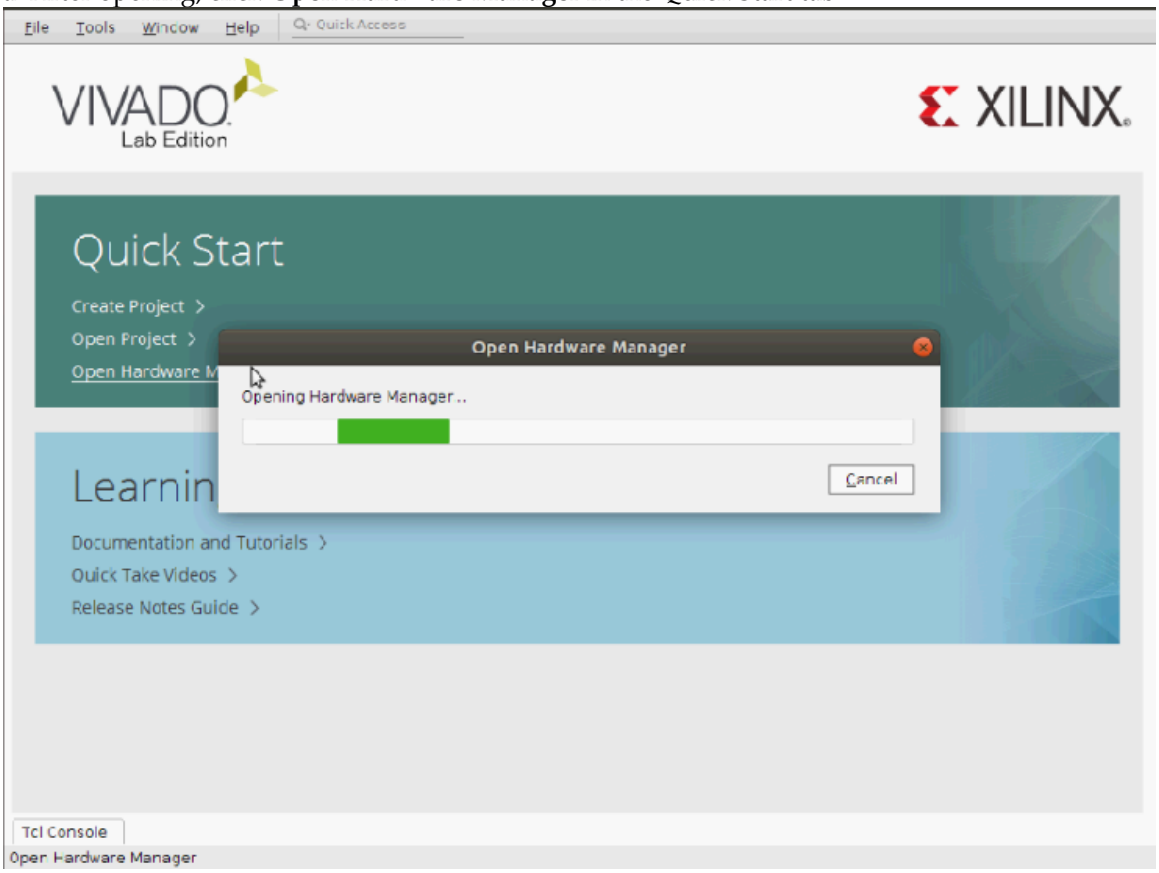
   b. Exit xsdb by entering **exit** in the xsdb console.

3. Launch the Vivado tools. For example, you can enter the following:

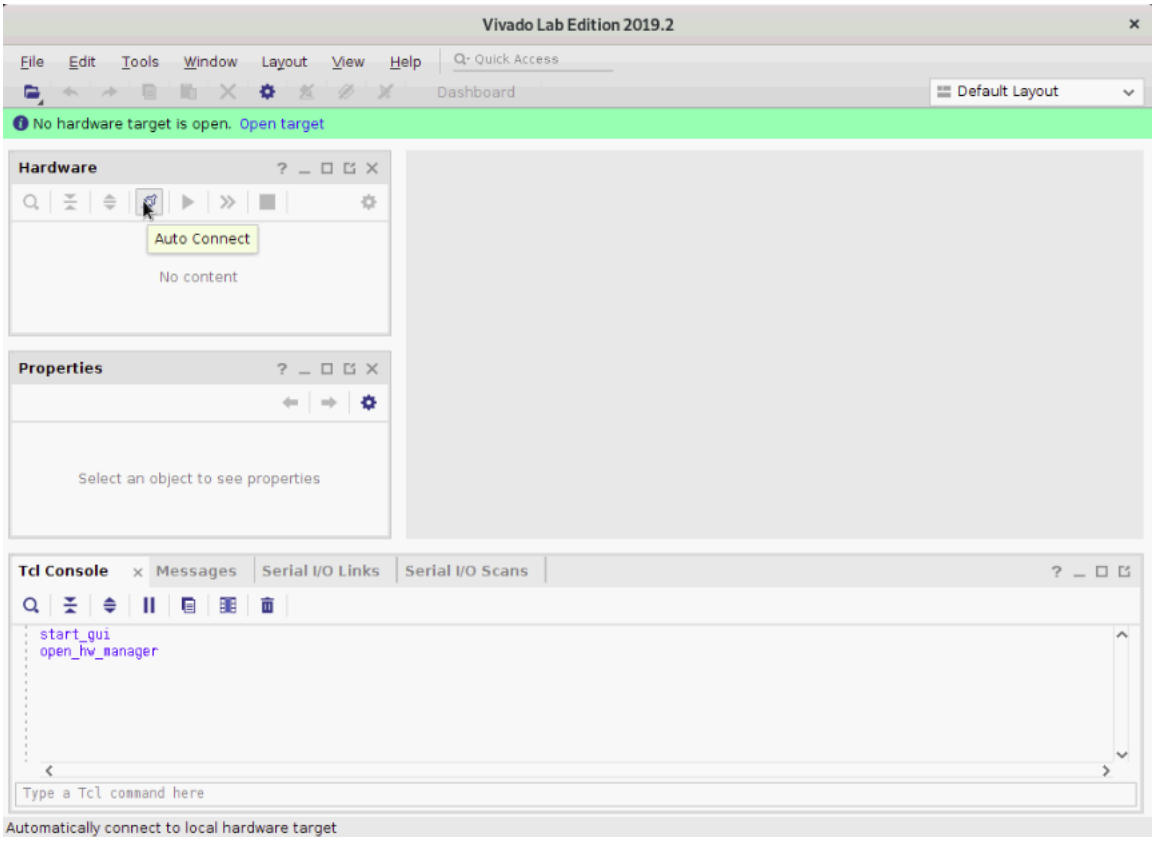
```
sudo <path>Vivado_Lab/2019.2/bin/vivado_lab
```

This launches the Vivado software which is used to flash the FPGA image. After the Vivado soft-

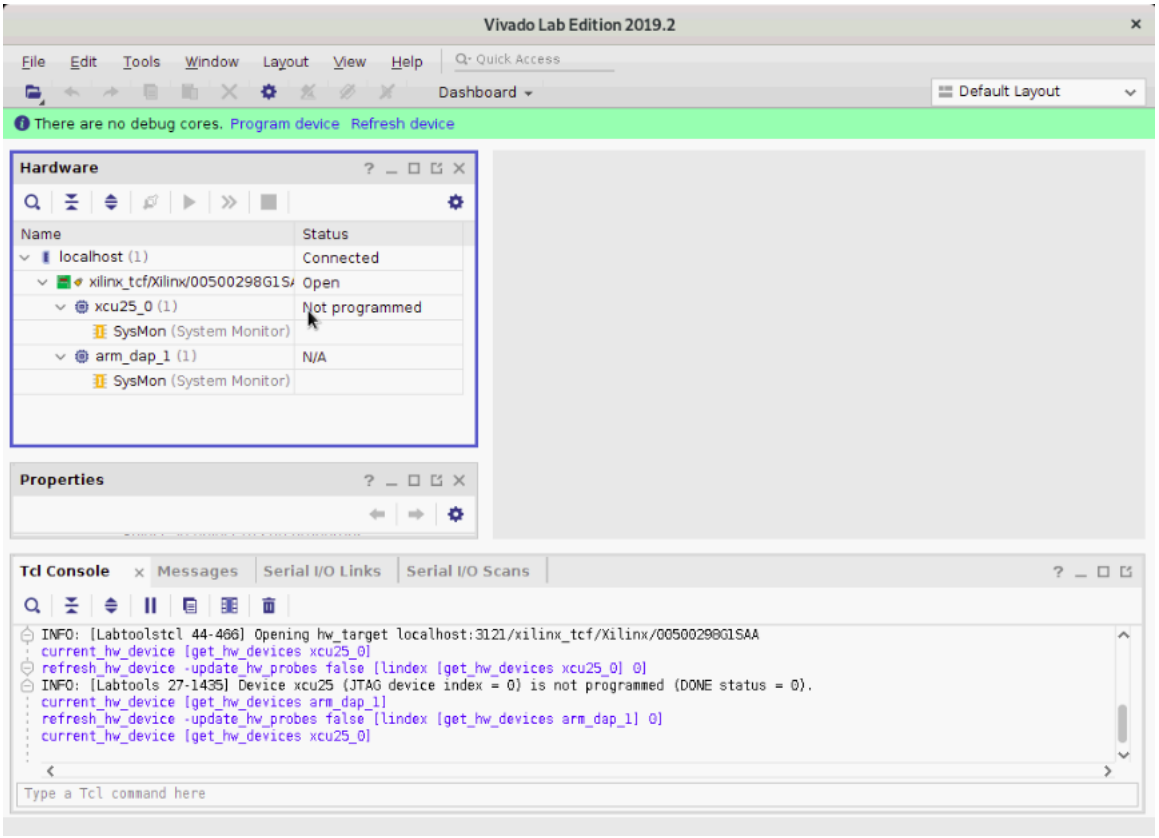
ware opens, carry out the following steps:
a. After opening, click **Open Hardware Manager** in the Quick Start tab.



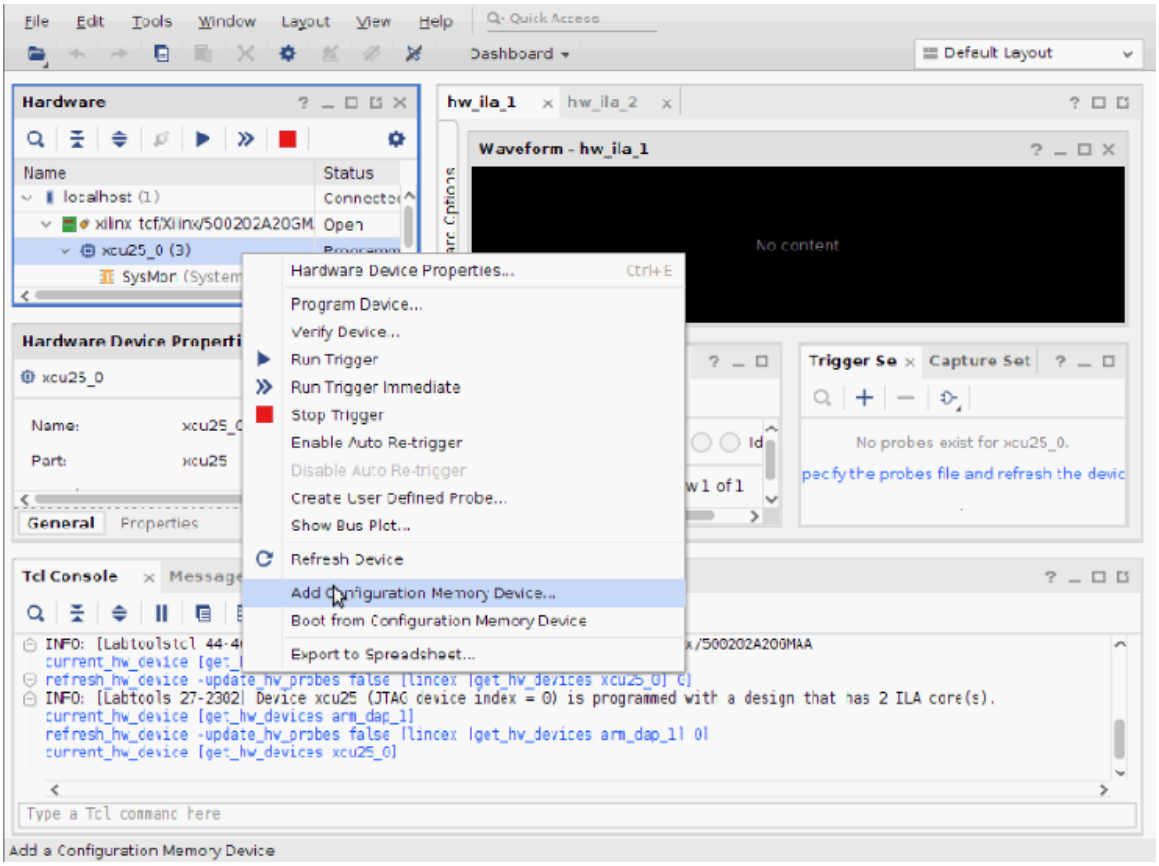
b. Click **Auto Connect**.



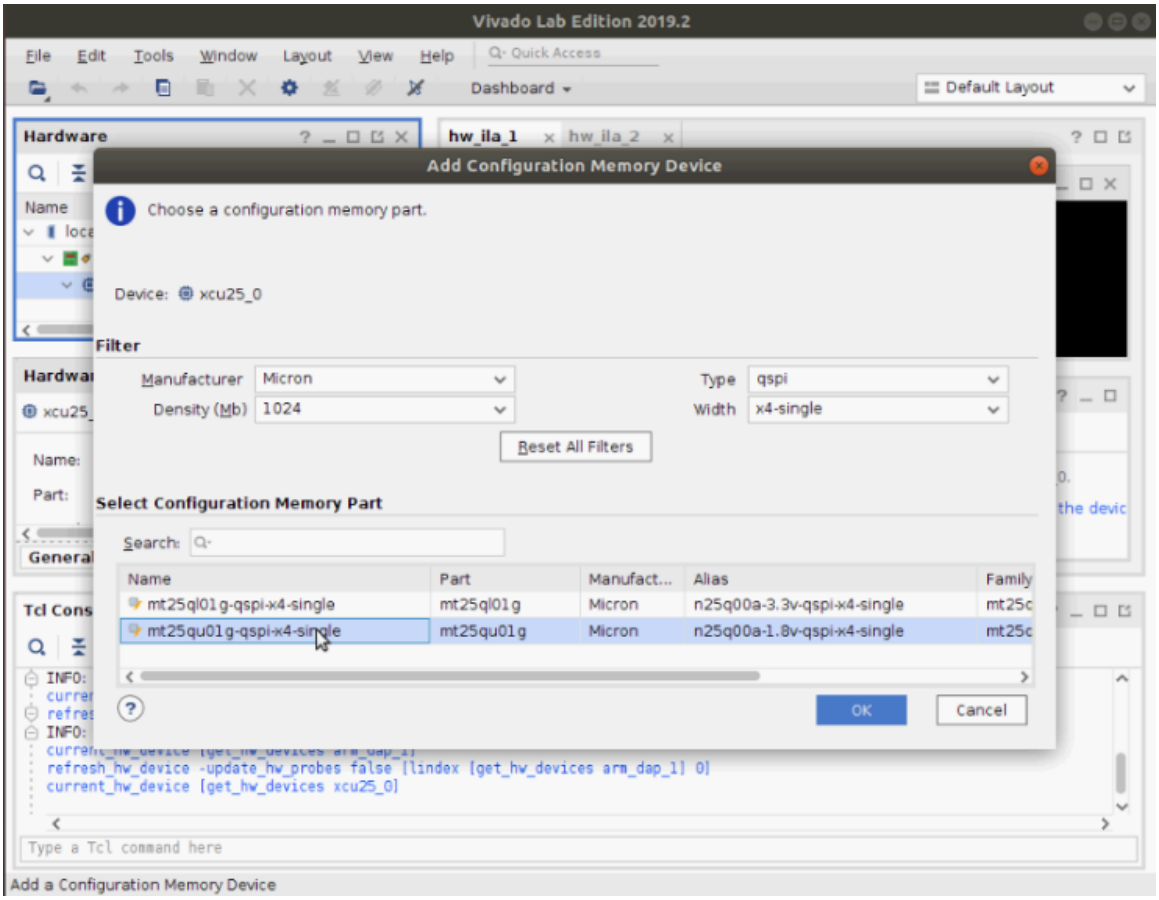
c. Verify whether the U25N SmartNIC is detected properly.



d. Select the SmartNIC as shown below, right-click, and select **Add Configuration Memory Device**.



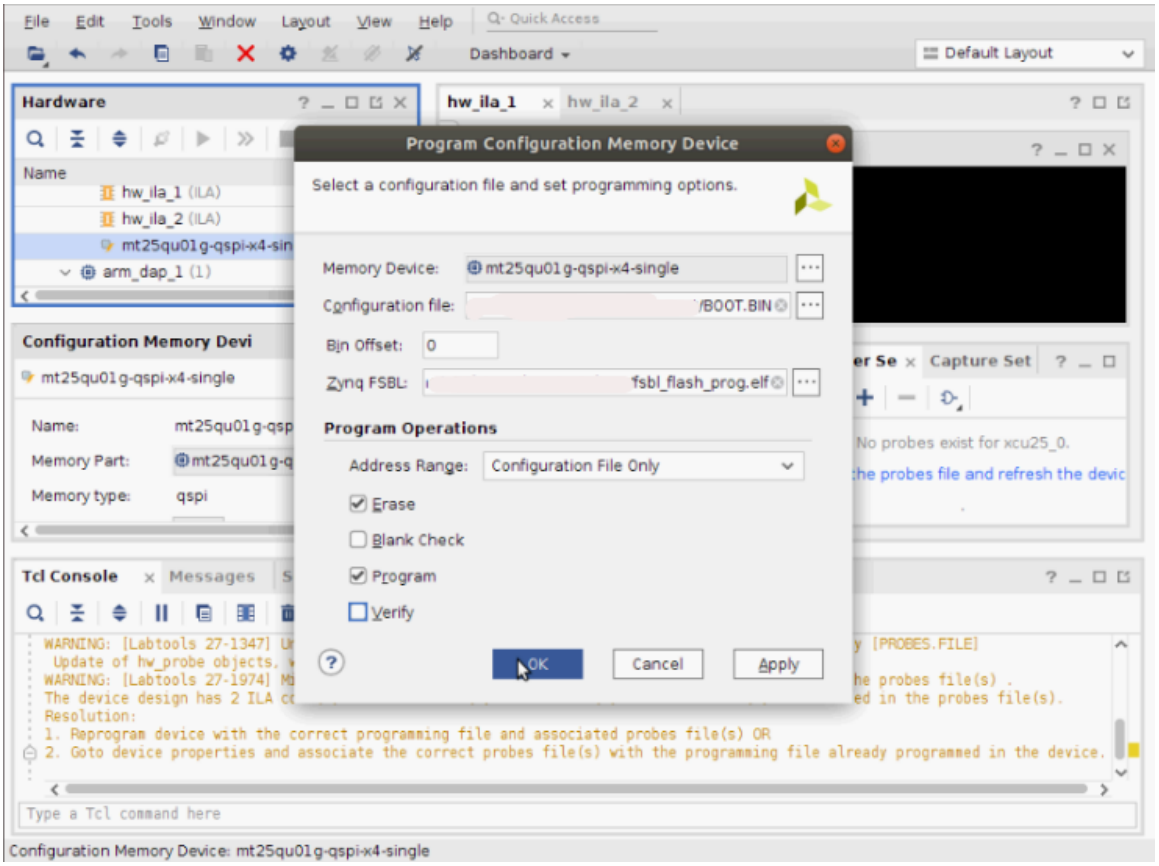
e. Search by clicking **mt25qu01g-qspi-x4-single** in the Search tab.



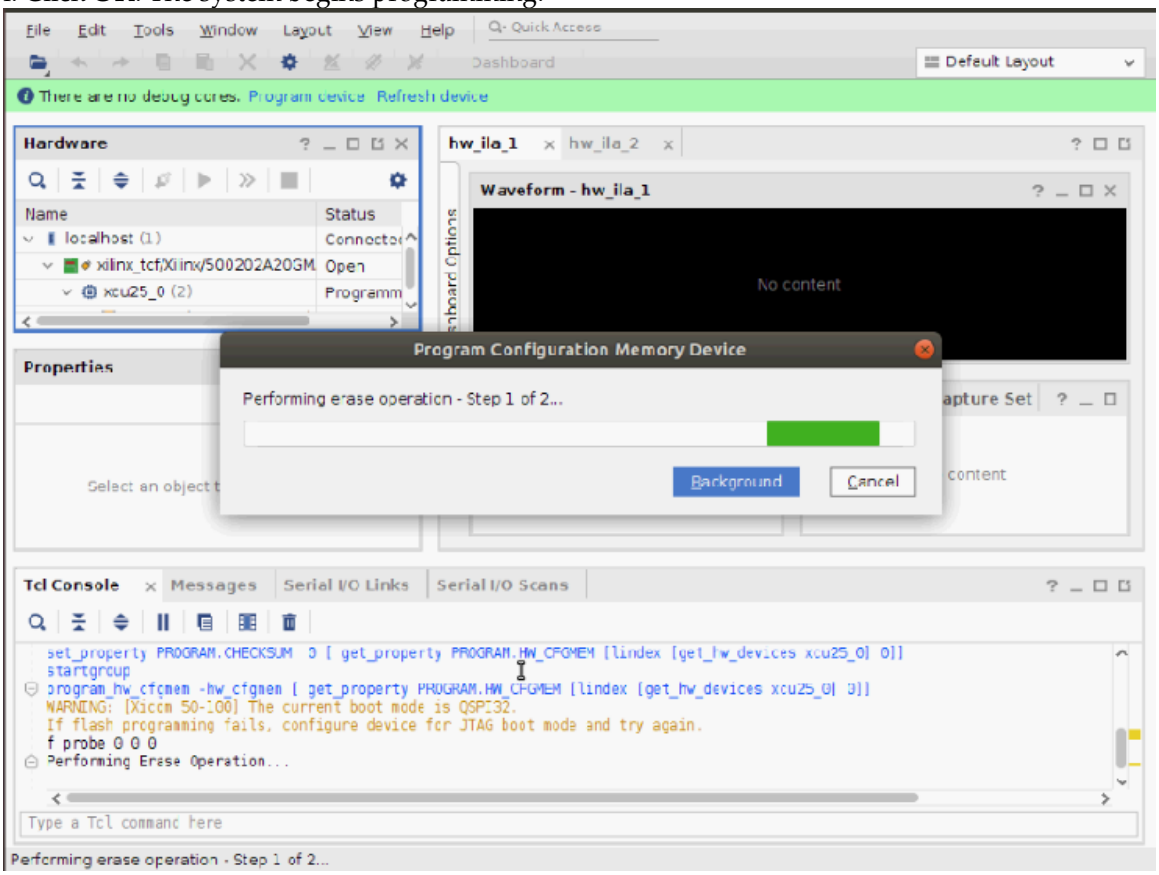
f. After making the above selections, a dialog box appears. Click **OK** and continue.

g. Select the configuration file (`BOOT.BIN`), and first stage bootloader file (`fsbl_flash_prog.elf`).

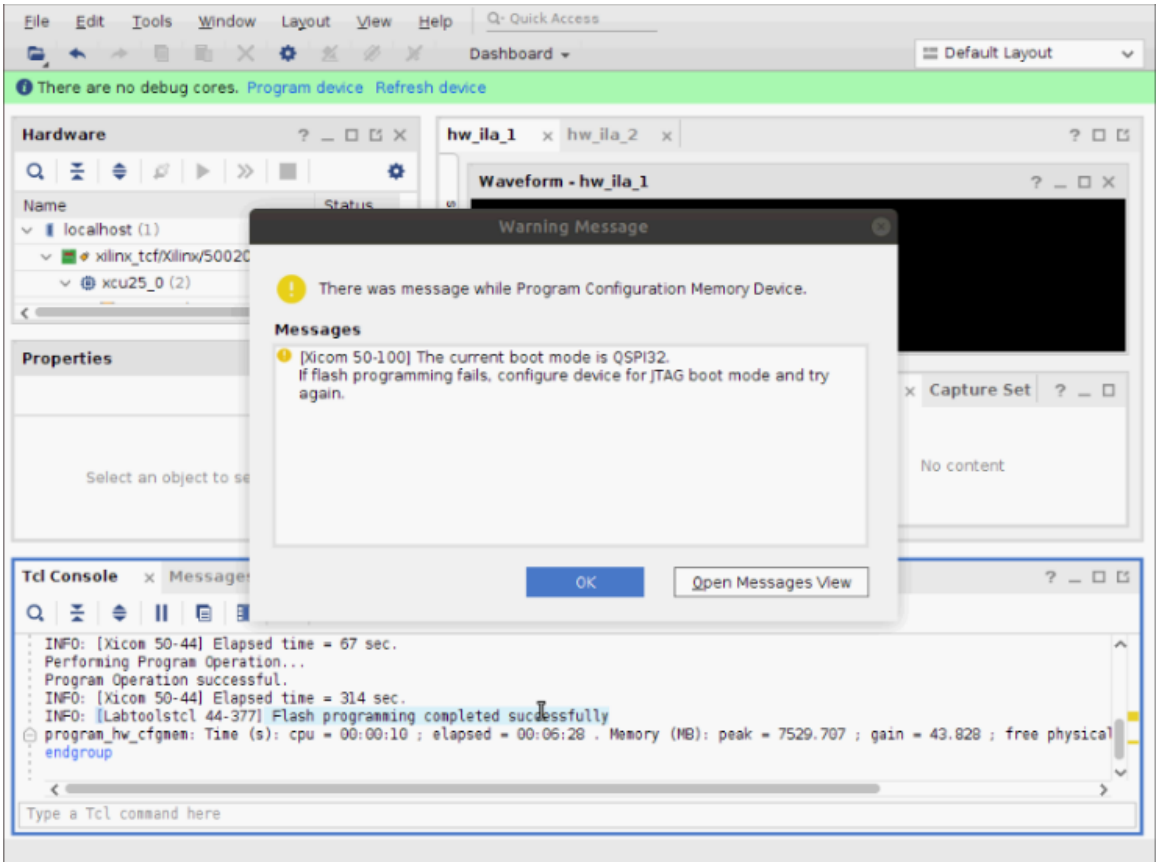
h. Uncheck Verify under Program operations. It will be time consuming otherwise.



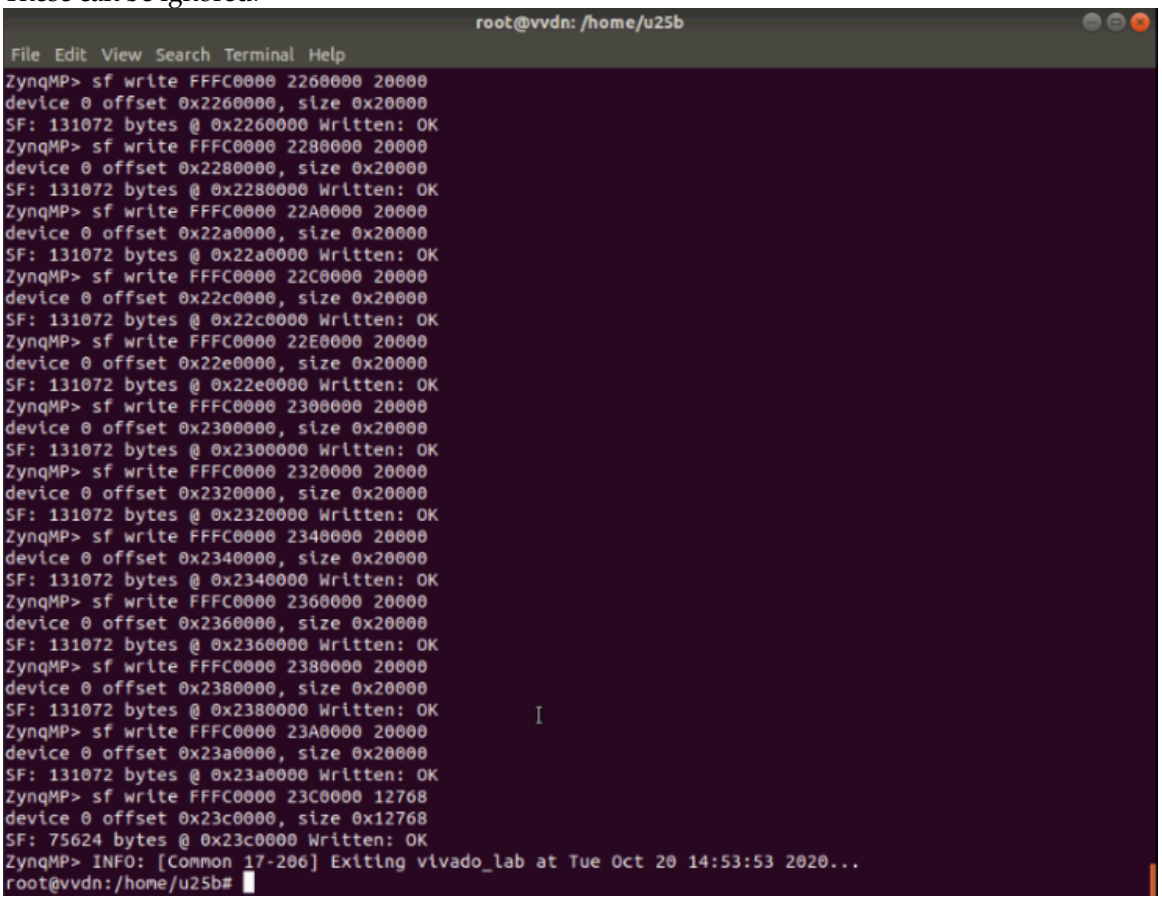
i. Click **OK**. The system begins programming.



j. The programming takes 5 to 10 minutes. Upon successful completion of programming, the following output log appears. Click **OK**.



k. If you exit the Vivado tools after the flash programming completes, the following logs appear. These can be ignored.



4. Power cycle the server (Power OFF and Power ON).

APPENDIX B VM Installation

***Note*:** This steps mentioned in this section needs to be performed only for VM cases. The following configuration steps are for Ubuntu OS.

Installation of libraries and dependencies is required for running VMs

1. Install qemu (IN HOST):

```
sudo apt-get install qemu-system-x86
```

2. Update the following command in the path `/etc/sysctl.conf` for huge page configuration:

```
sudo vim /etc/sysctl.conf
kernel.shmmax = 25769803776 [ it allocates 24G 1G hugepages ]
vm.hugetlb_shm_group = 0
```

3. Update Host Grub with the following command:

```
sudo vim /etc/default/grub
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash hugepagesz=1GB hugepages=24
default_hugepagesz=1GB iommu=pt intel_iommu=on pci=realloc"
```

4. After doing changes in Host Grub, update the Grub using the following command:

```
sudo update-grub
sudo reboot
```

5. Allocate the hugepage of size 8 Gb for a VM using this command:

```
mkdir /mnt/huge_vm
mount -t hugetlbfs -o size=8G none /mnt/huge_vm
```

6. Insert the VFIO driver using this command:

```
sudo modprobe vfio-pci
```

   ***Note*:** Make sure the next step is done only after the VF representor interfaces are added to the OVS bridge.

7. Unbind the VF PCIe® id from sfc before binding to the vfio-pci driver.
   ***Note*:** The VF PCIe device ID can be listed with the `lspci -d 1924:1b03` command. An example of the device ID is *af:00.2 Ethernet controller: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet Controller (Virtual Function) (rev 01)*.

```
echo '0000:<VF PCIe id>' > /sys/bus/pci/drivers/sfc/unbind
```

   For example:

```
echo '0000:af:00.2' > /sys/bus/pci/drivers/sfc/unbind
```

8. Bind the Vendor ID and device ID of the U25N X2 to VFIO driver:

```
echo "1924 1b03" > /sys/bus/pci/drivers/vfio-pci/new_id
```

9. Bind the respective PCIe device ID to the driver:

```
echo '0000:<VF PCIe id>' > /sys/bus/pci/drivers/vfio-pci/bind
```

For example:

```
echo '0000:af:00.2' > /sys/bus/pci/drivers/vfio-pci/bind
```

10. QEMU command for launching the VMs from terminal are:

```
qemu-system-x86_64 -cpu host -enable-kvm -m <Memory> -mem-prealloc -mempath
/mnt/huge_vm -smp sockets=<cpu_socket>,cores=<no_of_cores> -hda
<path_to_qcow2_imagr> -device e1000,netdev=net0 -netdev
user,id=net0,hostfwd=tcp:<port>-:22 -device vfio-pci,host=<VF PCIe id> -
vnc :<port> &
```

For example:

```
qemu-system-x86_64 -cpu host -enable-kvm -m 8192 -mem-prealloc -mempath
/mnt/huge_vm -smp sockets=1,cores=4 -hda ubuntu.qcow2 -device
e1000,netdev=net0 -netdev user,id=net0,hostfwd=tcp:5556-:22 -device vfiopci,
host=af:00.2 -vnc :5556 &
```

***Note*:** In the above command, 8 GB memory has been allocated. Four cores and qcow2 images are named ubuntu.qcow2.

11. After the above command gets executed, run the VM using the following command in different terminals:

```
ssh -p <port> <user_name>@127.0.0.1
```

For example:

```
ssh -p 5556 vm@127.0.0.1
```

The Alveo™ U25N is a 2x10/25G SmartNIC. The half-height, half-length (HHHL) Alveo U25N SmartNIC is compliant with the PCI Express® Gen3 x8 (x16 connector). It features the Zynq® UltraScale+™ XCU25 MPSoC and XtremeScale X2 Ethernet controller. The Alveo U25N SmartNIC platform is based on a powerful FPGA, enabling hardware acceleration and offload to happen inline with maximum efficiency while avoiding unnecessary data movements and CPU processing. It is composed of multiple software modules that contain Ethernet drivers and control demons such as vswitchd and strongSwan.

This user guide describes installation, configuration, and operation of the Alveo U25N SmartNIC, as well as its features, performance, and diagnostic tools. For the U25N SmartNIC feature list, refer to the Alveo U25N page.

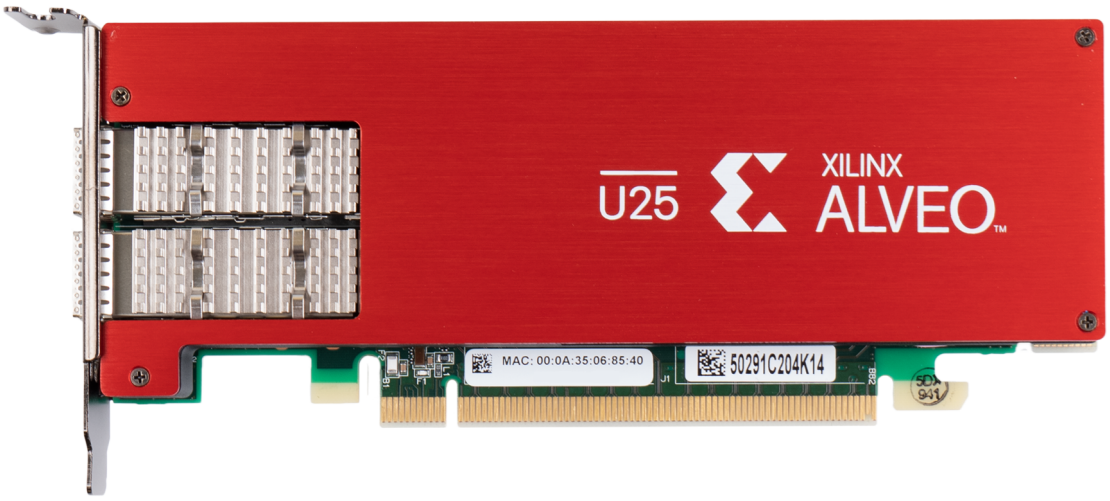


Figure 1. *Figure 1:* **Alveo U25N SmartNIC**

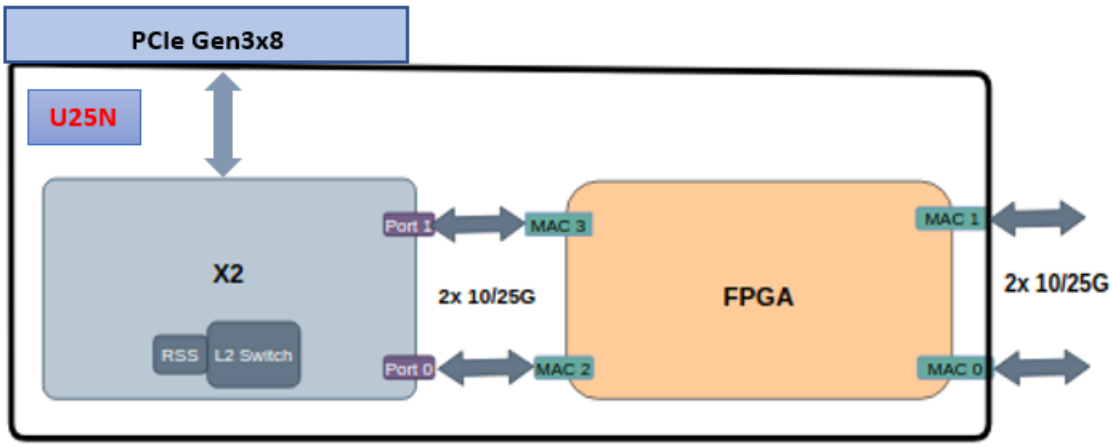


Figure 2. *Figure 2:* **U25N Architecture**