U25N documentation

Release 1.0

Xilinx, Inc.

Mar 31, 2022

Table of Contents

Introduction

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: **Alveo U25N SmartNIC**

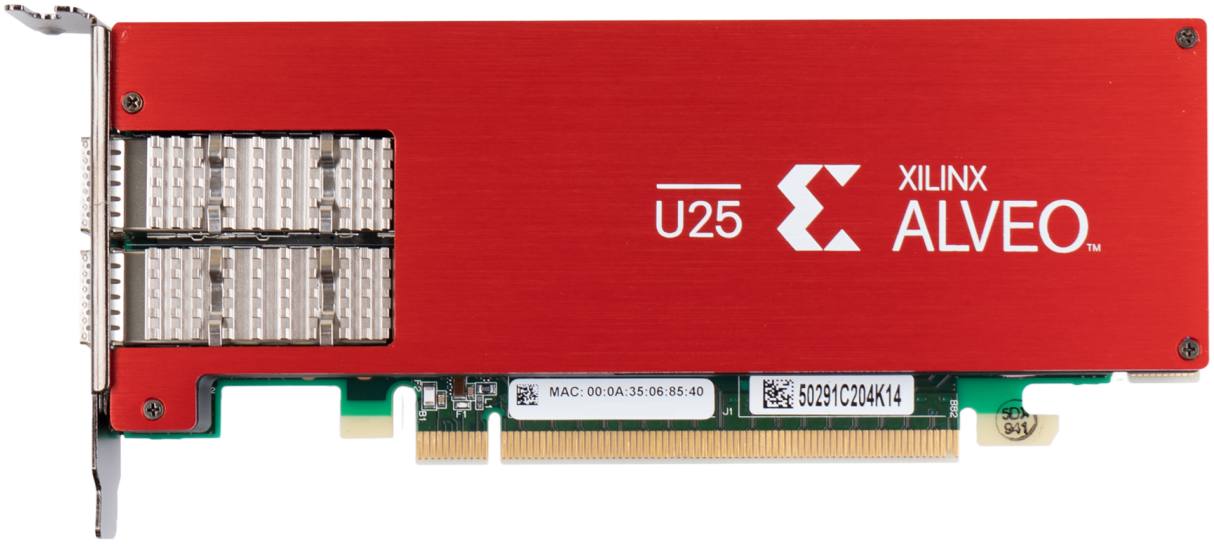
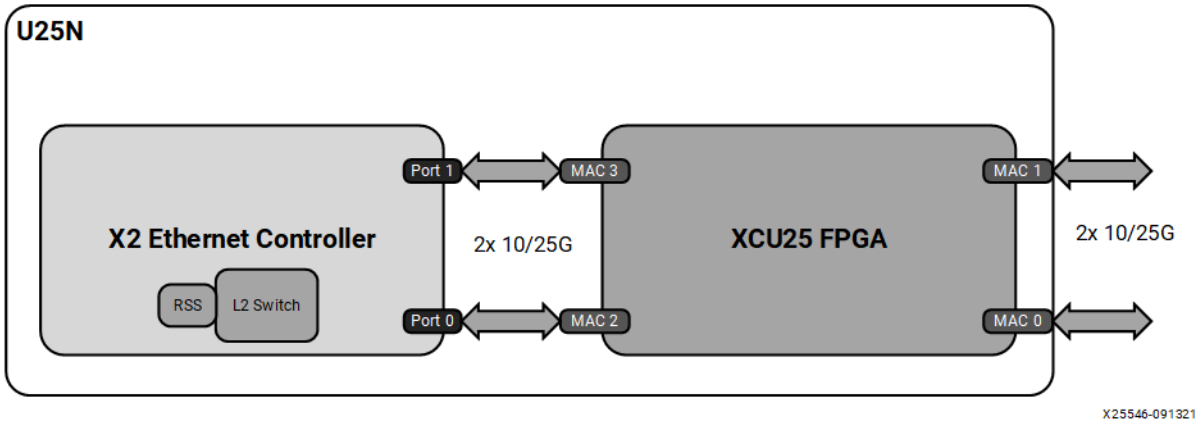


Figure 2: **U25N Architecture**

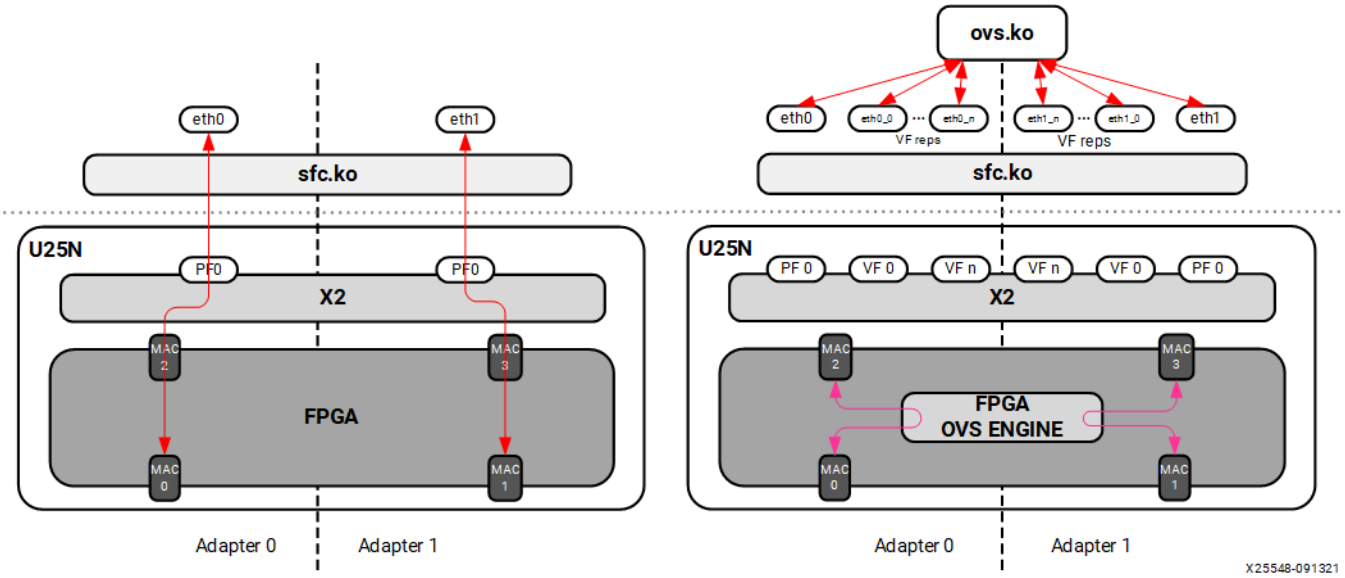

U25N
X2 Ethernet Controller
RSS
L2 Switch
Port 1
Port 0
2x 10/25G
MAC 3
MAC 2
XCU25 FPGA
MAC 1
MAC 0
2x 10/25G
X25546-091321

Supported Services

2.1 U25N Modes: Legacy and Switchdev Mode

The U25N SmartNIC supports two different modes of operation. One is a switchdev mode that enables Open vSwitch (OVS), and the other is a legacy mode without OVS. The basic operation is shown in the following figure, and more details are provided in Legacy and Switchdev Modes.

Figure 3: **U25N Modes of Operation**



2.2 Open vSwitch

OVS is a multilayer software switch licensed under the open source Apache 2 license. The Alveo U25N SmartNIC implements a production quality Open vSwitch platform that supports standard management interfaces and offloads the forwarding functions to hardware. For more information about Open vSwitch, refer to https://docs.openvswitch.org/en/latest/intro/what-is-ovs.

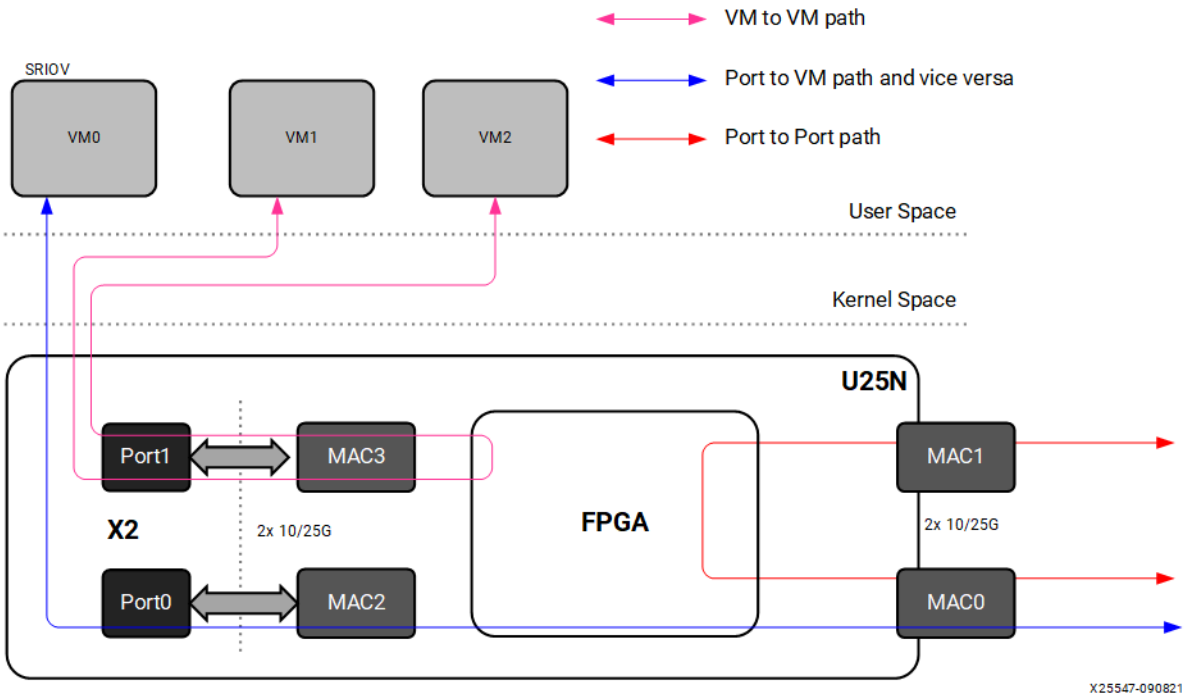
OVS supports various dataflows—port to VM, VM to VM, and port to port. Details of these dataflows are presented in the following sections.

2.2.1 Port to Port

In this scenario, traffic flows from one external network port to another external port, and vice versa. The OVS dataplane is offloaded to the SmartNIC, and the control plane is on the x86. If packets arrive

without a proper flow/rule entry in the OVS dataplane, the initial few packets get forwarded to the x86. Then the OVS control plane adds an entry in the OVS dataplane of the SmartNIC based on the policy programmed. Subsequent packet switching is done by the SmartNIC OVS.

Figure 4: **OVS Offload Scenario**



2.2.2 Port to VM or VM to Port

In this scenario, traffic flows from the host or VMs to any external ports, and vice versa. Create a virtual function (VF) corresponding to the physical function (PF) interface using the sfboot command (refer to the Solarflare user guide at https://support-nic.xilinx.com/wp/drivers). Here, the packet switching is done by OVS.

2.2.3 VM to VM

In this scenario, traffic flows from a VM to another VM via OVS in the SmartNIC. Create two VFs corresponding to the PF interface using the sfboot command. Here, the packet switching is done by OVS.

2.2.4 L2GRE

OVS supports layer 2 over generic routing encapsulation (L2GRE). L2GRE is a mechanism that encapsulates the entire packet. The solution supports L2GRE encapsulation and decapsulation on U25N hardware.

2.2.5 VXLAN

The solution supports VXLAN tunneling/de-tunneling. Tunnel interfaces can be created using OVS. The tunnel process is offloaded on the U25N hardware.

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. Control packets are forwarded to the CPU and handled by an open source application called

strongSwan (see https://www.strongswan.org). Security association (SA) and security policy (SP) are added to the U25N hardware as flows using a plugin library linked to strongSwan through ioctls exposed from the U25N driver. 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.

2.4 Stateless Firewall

Stateless firewall should be added on the ingress of ports so that filtering based on nftables rules can be applied. 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.

2.5 DPDK on U25N

Data Plane Development Kit (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.

2.6 Statistics Counter

Statistics data is updated periodically from U25N hardware. Counters are available for each service separately.

2.7 Debug

A debug command is provided to collect the debug logs from the processor subsystem (PS) on the FPGA and send them to the x86. Debug commands for verifying the default rules and offloaded rules from the x86 are provided.

U25N Installation

3.1 Basic Requirements and Component Versions Supported

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

  ***Note*:** Xilinx 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: 5.3.3.1003 (minimum).
  ***Note*:** To install the latest U25N driver, refer to U25N Driver.
- X2 firmware version: v7.8.17.1004 (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.

3.2 U25N Driver

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

After the server is booted with U25N hardware, check whether the U25N SmartNIC is detected using the lspci command:

```
lspci | grep Solarflare
```

Two PCI IDs corresponding to the U25N are displayed:

```
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)
```

3.2.1 Utility

The Solarflare Linux Utilities package (SF-107601-LS) is available from https://support-nic.xilinx.com/wp/drivers?sd=SF-107601-LS-71&pe=1945.

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 sfupdate, sfkey, sfctool, and sfboot utilities are available on the server.

3.2.2 U25N Driver Installation

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

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

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

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

   This should give the output as version 5.3.3.1003.
   ***Note*:** If the driver version is at least 5.3.3.1003, ignore the following step.
2. If the Debian package already exists, remove the already existing one before installing the latest Debian 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 example:

   ```
   dpkg -i sfc-dkms_5.3.3.1003_all.deb
   ```

3.2.3 Updating U25N Firmware

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

   ```
   lsmod | grep sfc
   ```

   For example:

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

2. Execute the following step to update the firmware:

   ```
   sfupdate -i <X2_interface> --image <firmware image> --write --force --yes
   ```

   ***Note*:** Use the PF0 interface. No reboot is required.

3. Confirm the firmware version using the sfupdate command. The version should be 1004.

```
sfupdate | grep Bundle
```

Bundle type: U25 (bundle type for production SmartNICs is U25N) Bundle version: v7.8.17.1004 (minimum)

3.2.4 sfboot Configuration

***Note*:** Ignore this section if the U25N mode is SR-IOV and the required VF counts are already assigned. This can be verified by running the sfboot command as root.

Refer to the Solarflare server adapter user guide at https://support-nic.xilinx.com/wp/drivers. X2 switch mode should be in SR-IOV. Based on the number of VF required on each U25N PF interface, execute the following command. The maximum vf-count could be 120:

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

For example, here 120 VFs have been assigned for each U25N PF interface:

```
sudo sfboot switch-mode=sriov vf-count=120
```

***Note*:** Powercycle is required to update the configuration. The configuration is retained even after cold or warm reboot.

3.3 U25N Shell

By default, the U25N shell is programmed with the golden image. Follow the steps in the following sections to verify the shell. If this fails, refer to U25N Shell Programming.

3.3.1 Shell Version Check

***Note*:** If the U25N driver version is 5.3.3.1003, ignore step 1.

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

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

This should give an output of version of 5.3.3.1003.
***Note*:** To install the latest sfc driver, refer to the U25N Driver section.

2. The u25n_update utility can be used to read the U25N shell version. Version reading is done at the U25N PF0 network interface using the u25n_update utility.
    1. Reading of version is performed using a CLI command:

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

For example:

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

Golden shell version: 0x21081600.

    2. After the versions are read successfully, the log would be:

```
STATUS: SUCCESSFUL
```

***Note*:** If u25n_update status shows failed, the shell is not programmed with the golden image. Refer to U25N Shell Programming to flash the golden image to the U25N shell.

Follow the sections below if needed to check the golden image functionality. For flashing deployment

image to U25N shell, refer to the Deployment Image Flashing section.

3.3.2 Golden Image Functionality

- U25N shell with Golden image includes Image Upgrade and Basic NIC functions. Offloaded features are not supported.
- U25N hardware in legacy mode [MAC0 TO MAC2 AND MAC1 TO MAC3].

Checking Basic NIC Functionality

1. Run DPDK testpmd at the X2 PF interface.
2. The packet sending to an external MAC is received at the testpmd application corresponding to PF bound.
3. Connect the external MAC to the traffic generator or any peer NIC device.

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

3.3.3 Deployment Image Flashing

The U25N SmartNICs are shipped with the golden image. Refer to Shell Version Check to check the shell version. If the shell has a golden image, continue with the following steps.

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

1. If the U25N driver version is 5.3.3.1003, ignore this step. Check the driver version of the U25N interface using the following command:

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

   This should give an output of version 5.3.3.1003.
   ***Note*:** To install the latest sfc driver, refer to U25N Driver.

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

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

   The output of the above command is:

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

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

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

3. Image flashing is done via the U25N PF0 network interface using the u25n_update utility.
   1. The image is flashed using a CLI command.
      ***Note*:** The interface should be the PF0 interface of the target U25N SmartNIC. The file must be a .bin file. Make sure the path to the image to be flashed is given correctly, as follows:

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

      For example:

      ```
      ./u25n_update upgrade BOOT.bin u25eth0
      ```

   2. Flash operations like ERASE, WRITE, and VERIFY are displayed in the CLI based on the procedures done by the u25n_update utility.
   3. After the image is flashed successfully, the following log is shown:

```
STATUS: SUCCESSFUL
```

4. Reset the U25N hardware to boot from the updated deployment image:
   For example:
   After the reset is done successfully, the following log is shown:
5. The u25n_update utility can be used to get the U25N deployment image version:

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

   For example:

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

   Deployment shell version: 0x21011800.
   After the versions are read successfully, the following log is shown:

   ```
   STATUS: SUCCESSFUL
   ```

3.3.4 Loading Partial Bitstream into Deployment Shell

***Note*:** The deployment image must be flashed to the U25N shell before following the steps below. Refer to Deployment Image Flashing to flash the deployment image to the U25N shell.

1. If the U25N driver version is 5.3.3.1003, ignore this step. Check the driver version of the U25N interface using the following command:

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

   This should give an output of version 5.3.3.1003.
   ***Note*:** To install the latest sfc driver, refer to 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.
2. The U25N SmartNIC should be in legacy mode. Use the following command to verify this:

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

   The output of the above command is:

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

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

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

   ***Note*:** The interface should be the PF0 interface of the target U25N SmartNIC. The file must be a `.bit` file. Make sure the path to the image to be flashed is given correctly.
3. Image flashing is done via the U25N PF0 network interface using the u25n_update utility.
   1. The image is flashed using a CLI command:

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

      For example:

      ```
      ./u25n_update upgrade fpga.bit u25eth0
      ```

   2. After the image is flashed successfully, the following log is shown:

      ```
      STATUS: Successful
      ```

4. The u25n_update utility can be used to get the bitstream version. Version reading is done at the U25N PF0 network interface using the u25n_update utility:

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

For example:

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

Bitstream version: 0xA00D10D1
After the versions are read successfully, the following log is shown:

```
STATUS: SUCCESSFUL
```

3.4 Updating XCU25 Linux Kernel Image

This section is required when the file system needs to be updated for the existing deployment image. The deployment image must be flashed to the U25N shell before following the steps below. Refer to Deployment Image Flashing to flash the deployment image.

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

1. If the U25N driver version is 5.3.3.1003, ignore this step. Check the driver version of the U25N interface using the following command:

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

This should give an output of version 5.3.3.1003.
***Note*:** To install the latest sfc driver, refer to U25N Driver.

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

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

The output of the above command is:

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

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

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

3. Image flashing is done via the U25N PF0 network interface using the u25n_update utility.
***Note*:** The interface should be the PF0 interface of the target U25N SmartNIC. The file must be a `.ub` file. Make sure the path to the image to be flashed is given correctly.
    1. The image is flashed using a CLI command:

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

    For example:

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

    2. Flash operations like ERASE, WRITE, and VERIFY are displayed in the CLI based on the procedures done by the u25n_update utility.
    3. After the image is flashed successfully, the following log is shown:

    ```
    STATUS: SUCCESSFUL
    ```

4. Reset the U25N hardware to boot with the new kernel image:

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

For example:

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

After the image is flashed successfully, the following log is shown:

```
STATUS: SUCCESSFUL
```

3.5 Reverting U25N SmartNIC to Golden Image

Reverting to 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 where the deployment image is unresponsive even after performing multiple resets using the u25n_update utility.

***Note*:** After the factory reset is performed, the U25N loaded is the golden image. It has the essential provision to upgrade the deployment image but no provision to upgrade bitstream.

1. If the U25N driver version is 5.3.3.1003, ignore this step. Check the driver version of the U25N interface using the following command:

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

This should give an output of version 5.3.3.1003.
***Note*:** To install the latest sfc driver, refer to 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.

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

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

The output of the above command is:

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

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

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

3. Factory reset is done via the U25N PF0 network interface.
    1. Factory reset is performed using a CLI command:

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

For example:

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

    1. After the factory reset is done successfully, the following log is shown:

```
STATUS: SUCCESSFUL
```

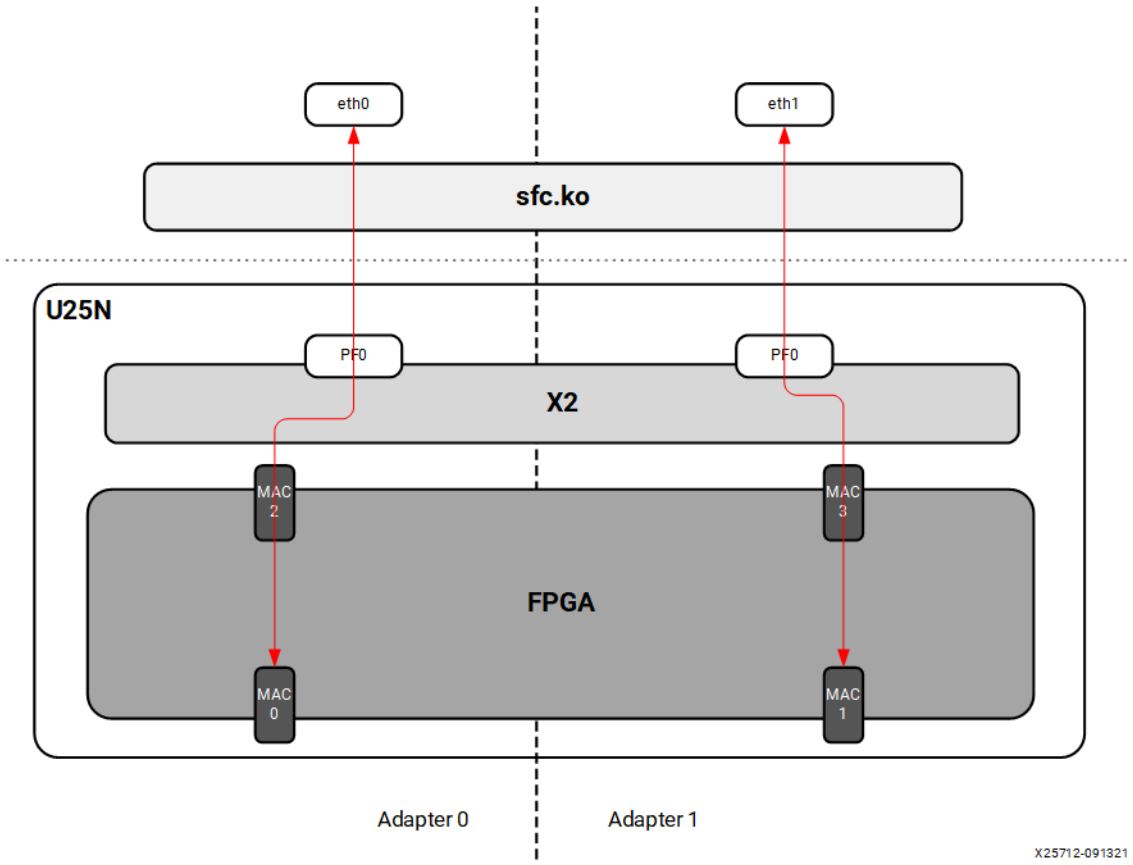
Detailed Applications Description

4.1 Legacy and Switchdev Modes

4.1.1 Legacy NIC (Default)

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.

Figure 5: **Legacy Mode**

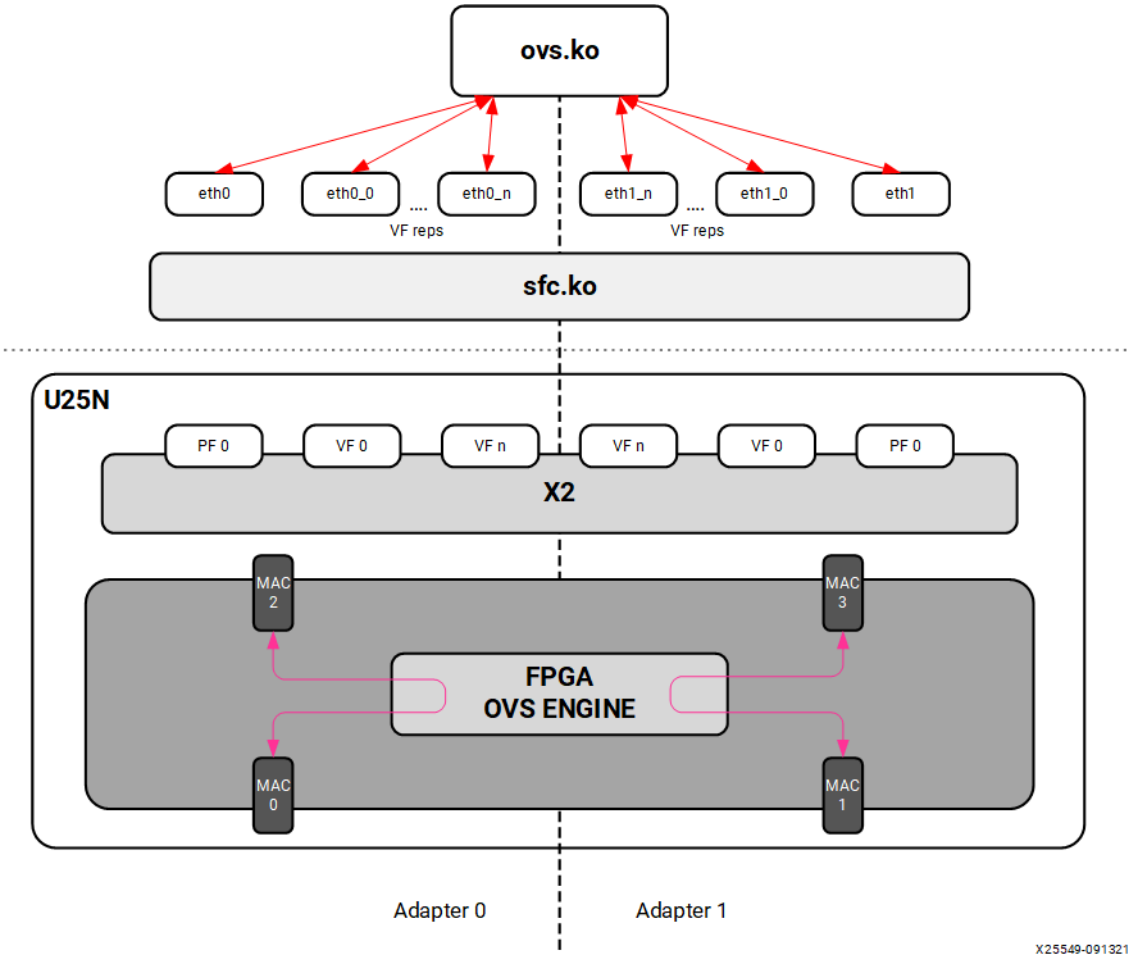


4.1.2 Switchdev Mode

When changed to 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**



4.2 OVS

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 VM environments. Carry out the following step to install OVS:

1. The OVS source code is its Git repository, which you can clone into a directory named ovs with the git clone command (see https://github.com/openvswitch/ovs.git).
2. After it has been cloned, the ovs directory will be in the current directory path. Move inside the ovs directory using the cd command. For example:

```
cd ovs
```

3. Execute the following commands one by one as the root user:

```
./boot.sh
./configure
make -j8
make install
```

4. Export the 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: 8k

4.2.2 Classification Fields (Matches)

Key

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

Action

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

4.2.3 Port to Port

The U25N PF is added to the OVS bridge. Packets are sent at an external MAC, and OVS does the switching based on the packet received.

1. Refer to Basic Requirements and Component Versions Supported for the required OS/ software version.
2. If the U25N driver version is 5.3.3.1003, ignore this step. Check the driver version of the U25N interface using the following command:

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

This should give the output as version 5.3.3.1003.
***Note*:** To install the latest sfc driver, refer to U25N Driver.

3. Make both PF interfaces up:
    1. List the interfaces using the `ifconfig -a` command.
    2. Find the U25N interface using the `ethtool -i <interface_name>` command:

    ```
    ifconfig <u25_interface> up
    ```

    For example:

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

4. Make the PF interfaces into switchdev mode.
***Note*:** Make sure the PF 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
    devlink dev eswitch set pci/0000:af:00.1 mode switchdev
    ```

5. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, continue to the next step.
6. Add external ports to the OVS bridge:

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

    For example:

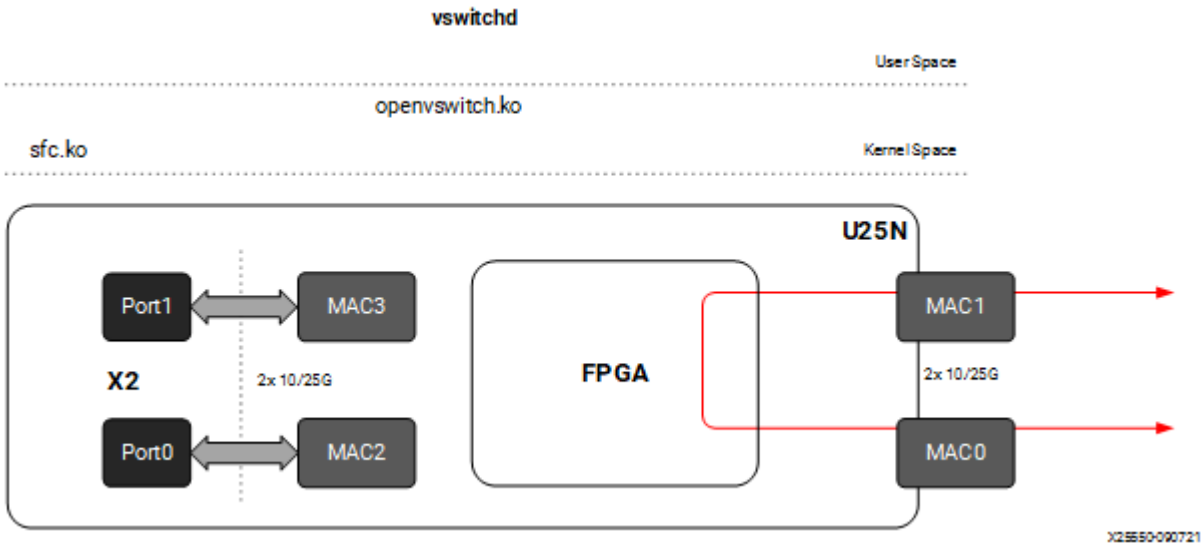
    ```
    ovs-vsctl add-port br0 U25_eth0
    ovs-vsctl add-port br0 U25_eth1
    ```

7. Print a brief overview of the database contents:

    ```
    ovs-vsctl show
    ```

    Refer to Functionality Check to check the OVS functionality.

Figure 7: **Port to Port**



4.2.4 Port to VM or VM to Port

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

1. Refer to Basic Requirements and Component Versions Supported for the required OS/software version. For VM use cases, VFs need to be created at 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.

2. Check the driver version of the U25N interface using the following command:
   ***Note*:** Ignore this step if the U25N driver version is 5.3.3.1003.

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

   This should give the output as version 5.3.3.1003.
   ***Note*:** To install the latest sfc driver, refer to U25N Driver.
   ***Note*:** Refer to Deployment Image Flashing for flashing images to check OVS functionality.

3. Make the U25N PF interface up:
   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: 5.3.3.1003

   ```
   ifconfig <U25_interface> up
   ```

   For example:

   ```
   ifconfig U25eth0 up
   ```

4. Allocate the number of VF to PF. Here, a single VF is allocated to the PF0 interface:
   ***Note*:** The VF could also be created in the PF1 interface based on use case. The sriov_numvfs count should be less than or equal to the VF count given in the sfboot command. The sriov_numvfs should be done only in legacy mode. To check the U25N mode, excecute 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
```

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 the above command is executed, a VF PCIe ID and VF interface are created. The VF PCIe device ID can be listed with the command `lspci -d 1924:1b03`. An example of the device ID is 86:00.2 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.

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.
6. Make the VF interface up:

```
ifconfig <vf_interface> up
```

7. Make the PF interfaces into switchdev mode:
***Note*:** Make sure 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`.

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*:** Here, the number of VF representor interfaces 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 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
```

9. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed to the next step.
10. Add PF interfaces as ports to the OVS bridge:

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

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

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

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

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

12. Make the OVS bridge up:

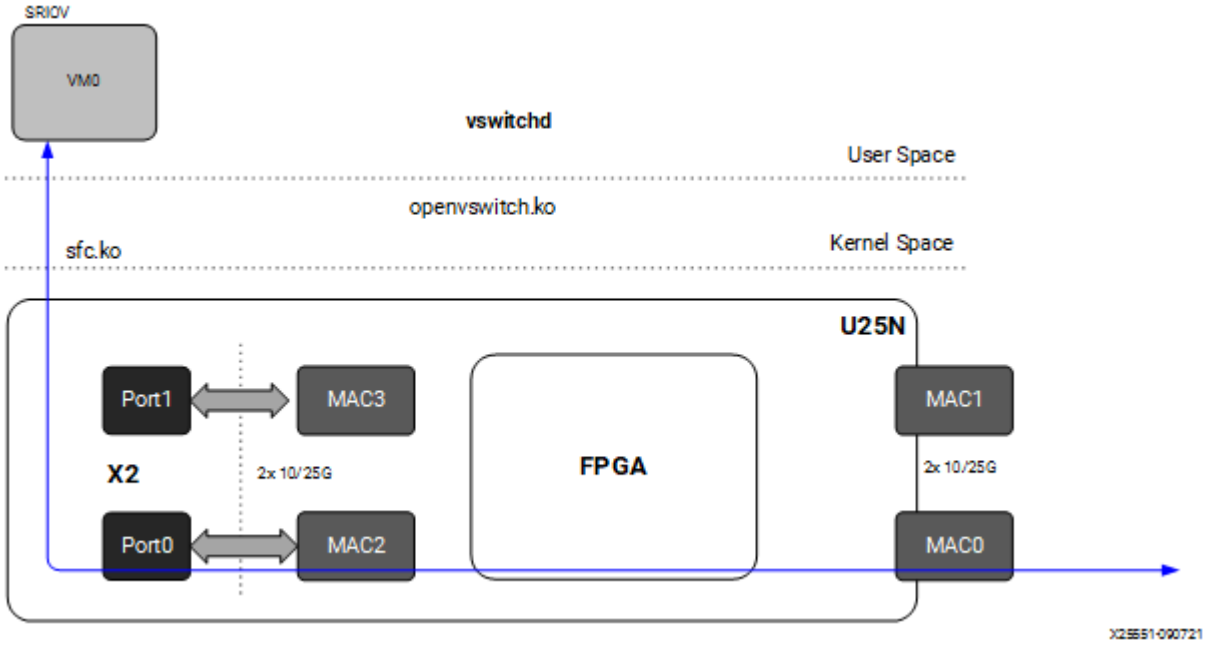
```
ifconfig <bridge-name> up
```

13. Print a brief overview of the database contents:

```
ovs-vsctl show
```

14. Refer to VM Installation to make the VM up. After the VM is up, proceed to the next step to check functionality.
15. Refer to Functionality Check to check OVS functionality.

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



4.2.5 VM to VM

***Note*:** SR-IOV must be enabled in BIOS. For a VM to VM case, a tunnel could be created with two server setups. Here the tunnel can be VXLAN or L2GRE.

1. Refer to U25N Driver for the required OS/software version. For VM use cases, VFs need to be created at the corresponding PF for binding to the VM. The number of VF counts should be configured in the sfboot command and in sriov_numvfs. Offload will occur only between VMs created using same the PF's VF.
   ***Note*:** For more information, refer to sfboot Configuration.
2. Check the driver version of the U25N interface using the following command:
   ***Note*:** Ignore this step if the U25N driver version is 5.3.3.1003.

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

   This should give the output as version 5.3.3.1003.
   ***Note*:** To install the latest sfc driver, refer to U25N Driver.
   ***Note*:** Refer to Deployment Image Flashing for flashing images to check OVS functionality.
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: 5.3.3.1003

```
ifconfig <U25_interface> up
```
For example:

```
ifconfig U25eth0 up
```

4. Allocate the number of VF to PF. Here, two VFs are allocated to the PF0 interface:
   ***Note*:** The VF could also be created in the PF1 interface based on the use case. The sriov_numvfs count should be less than or equal to the VF count given in the sfboot command. The sriov_numvfs should be done only in legacy mode. To check 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
   ```

   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 2 > /sys/class/net/<interface>/device/sriov_numvfs
   ```

   For example:

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

   ***Note*:** After the above command is executed, 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 would be used for binding VFs to VMs.

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.

6. Make the two VF interfaces up:

   ```
   ifconfig <vf_interfaceup>
   ```

7. Make the PF interfaces into switchdev mode.
   ***Note*:** Make sure 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`.

8. Running the above command creates two VF representor interfaces. The VF representor interface name will be the PF interface name and `_0` for the first VF representor and `_1` for the second VF representor.
   ***Note*:** Here, the number of VF representor interfaces created is 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
```

9. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed to the next step.
10. Add two VF representator interfaces to the OVS bridge:

```
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
```

11. Make the OVS bridge up:

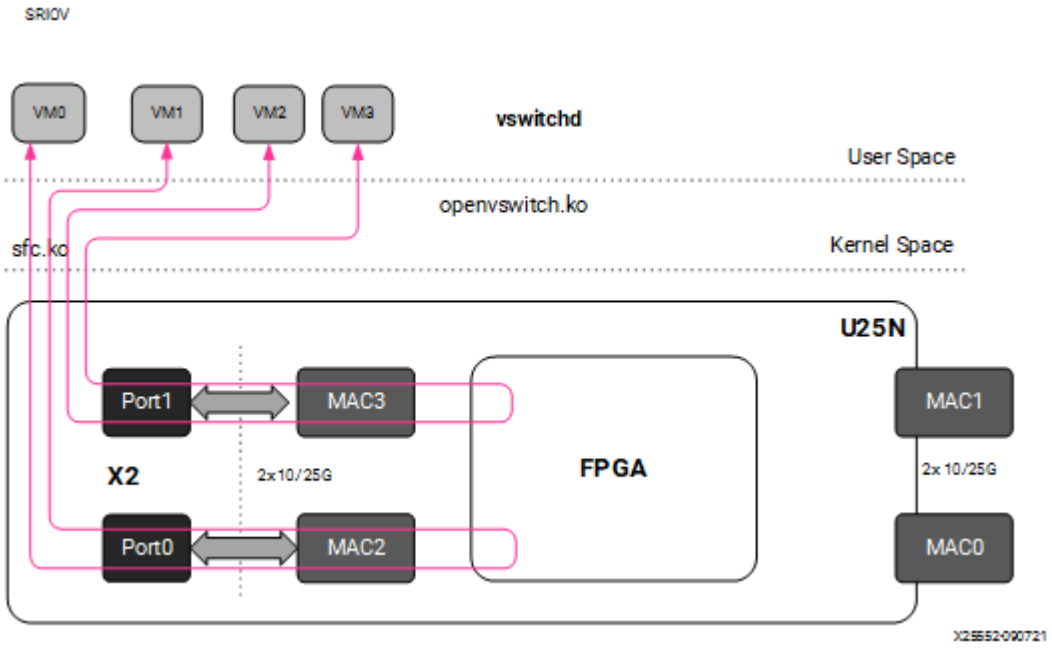
```
ifconfig <bridge-name> up
```

12. Print a brief overview of the database contents:

```
ovs-vsctl show
```

13. Refer to VM Installation to make the VM up. After the VM is up, proceed to the next step to check functionality.
14. Refer to [Functionality Check(link)] to check OVS functionality.

Figure 9: **VM to VM**



4.2.6 Tunnels (Encapsulation/Decapsulation)

U25N hardware supports offloading of tunnels using encapsulation and decapsulation actions.

- **Encapsulation:** Pushing of tunnel header is supported on TX
- **Decapsulation:** Popping of 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 setups. Here the tunnel can be L2GRE.

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

Refer to 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 where the tunnel needs to be created.

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

Server 1 Configuration

1. Check the driver version of the U25N interface using this command:
   ***Note*:** Ignore this step if the U25N driver version is 5.3.3.1003.

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

   This should give the output as version 5.3.3.1003
   ***Note*:** To install the latest sfc driver, refer to U25N Driver.
   ***Note*:** Refer to Deployment Image Flashing for flashing images to check OVS functionality.

2. Make the U25N PF interfaces 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: 5.3.3.1003

   ```
   ifconfig <U25_interface> up
   ```

   For example:

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

3. Assign tunnel IP to PF0 interface:

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

   For example, ifconfig U25eth0 10.16.0.2/24 up.

4. Make PF0 interface into 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
   ```

5. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed to the next 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=10.16.0.2 options:remote_ip=10.16.0.1
```

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
```

8. Make the bridge up:

```
ifconfig <bridge_name> up
```

For example:

```
ifconfig br0 up
```

9. Print a brief overview of the database contents:

```
ovs-vsctl show
```

Server 2 Configuration

1. Check the driver version of the U25N interface using this command:
   ***Note*:** Ignore this step if the U25N X2 driver version is 5.3.3.1003.

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

   This should give the output as version 5.3.3.1003.
   ***Note*:** To install the latest sfc driver, refer to U25N Driver.
2. Make the U25N PF interfaces 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:
3. Assign tunnel IP to PF0 interface:

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

   For example:

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

4. Make PF0 interface into 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
```

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

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
```

8. Make the bridge up:

```
ifconfig <bridge_name> up
```

For example:

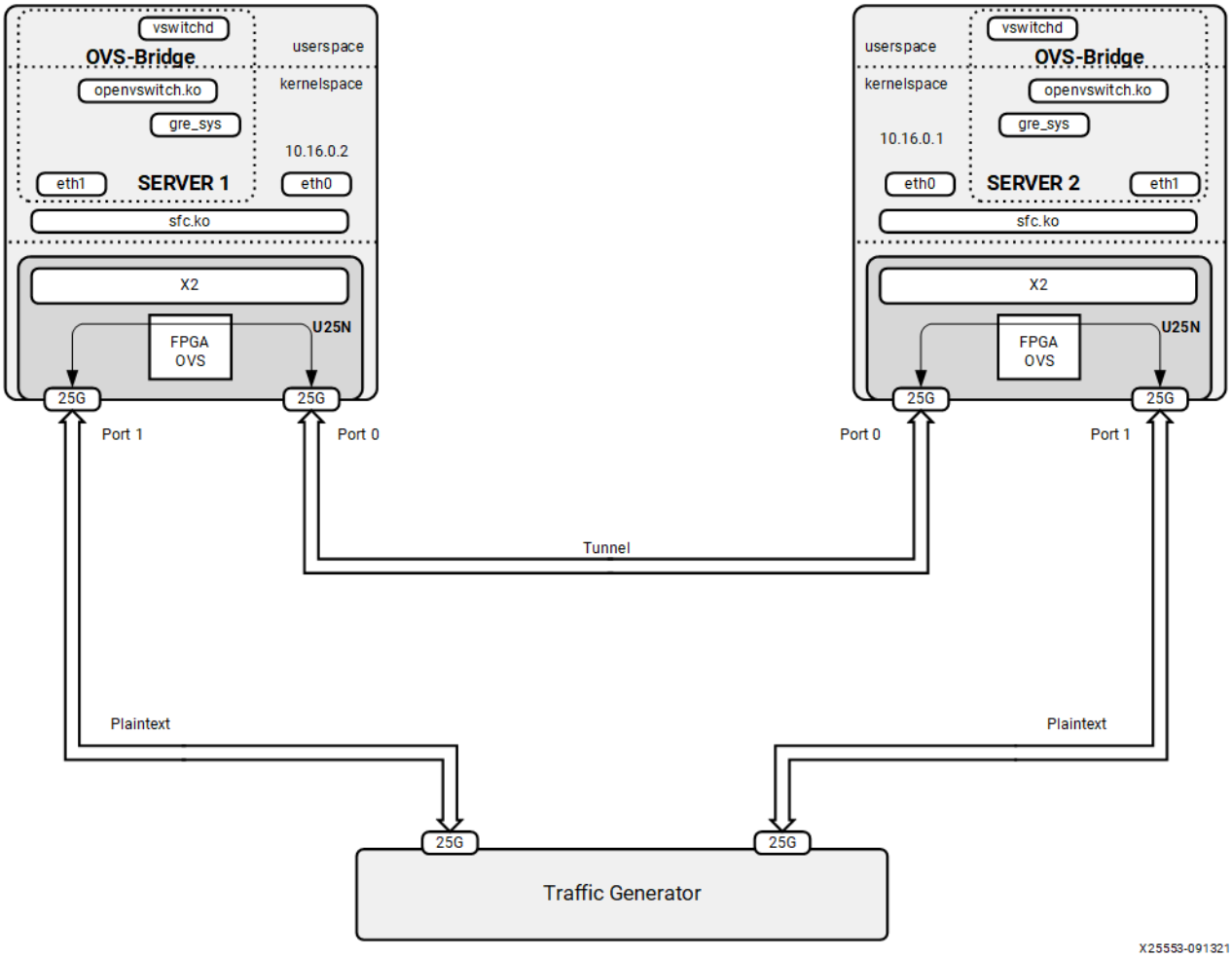
```
ifconfig br0 up
```

9. Print a brief overview of the database contents:

```
ovs-vsctl show
```

10. Refer to Functionality Check to check OVS functionality.

Figure 10: **L2GRE**



VXLAN

***Note*:** For Port to VM or VM to Port or VM to VM case, a tunnel could be created with two server setups. Here the tunnel can be VXLAN.

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

Refer to 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.

Server 1 Configuration

1. Check the driver version of the U25N interface using this command:
   ***Note*:** Ignore this step if the U25N X2 driver version is 5.3.3.1003.

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

   This should give the output as version 5.3.3.1003.
   ***Note*:** To install the latest sfc driver, refer to U25N Driver.
   ***Note*:** Refer to Deployment Image Flashing for flashing images to check OVS functionality.
2. Make the U25N PF interfaces 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: 5.3.3.1003

```
ifconfig <U25_interface> up
```

For example:

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

3. Assign tunnel IP to PF0 interface:

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

For example:

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

4. Make PF0 interface into 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`.

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

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`.

8. Make the bridge up:

```
ifconfig <bridge_name> up
```

For example:

```
ifconfig br0 up
```

9. Print a brief overview of the database contents:

```
ovs-vsctl show
```

Server 2 Configuration

1. Check the driver version of the U25N X2 interface using this command:
***Note*:** Ignore this step if the U25N driver version is 5.3.3.1003.

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

This should give the output as version 5.3.3.1003.
***Note*:** To install the latest sfc driver, refer to U25N Driver.

2. Make the U25N PF interfaces 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: 5.3.3.1003

```
ifconfig <U25_interface> up
```

   For example:

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

3. Assign tunnel IP to PF0 interface:

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

   For example:

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

4. Make PF0 interface into 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
```

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

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
```

8. Make the bridge up:

```
ifconfig <bridge_name> up
```

   For example:

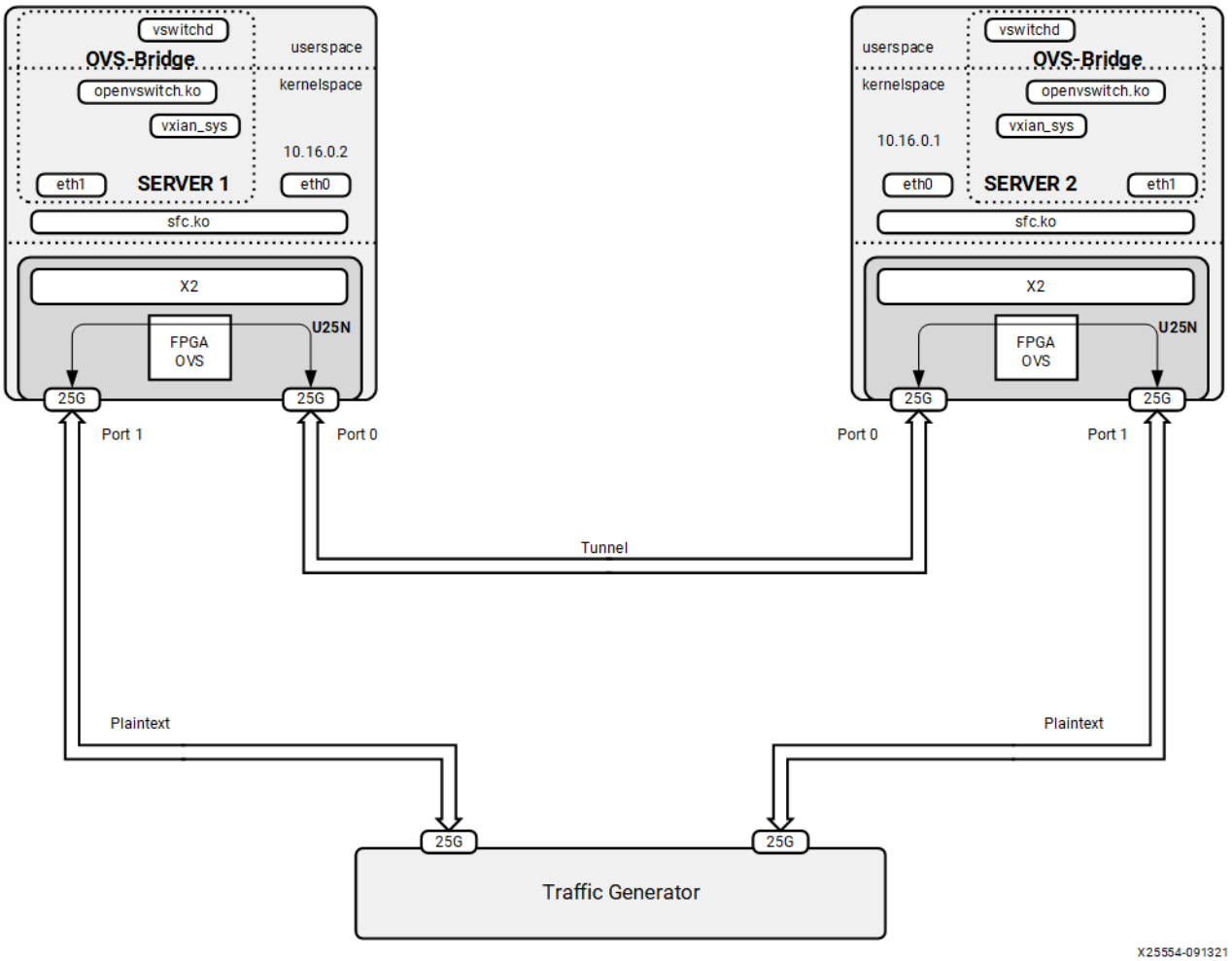
```
ifconfig br0 up
```

9. Print a brief overview of the database contents:

```
ovs-vsctl show
```

10. Refer to Functionality Check to check OVS functionality.

Figure 11: **VXLAN**



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 formed between two servers. Tunnel endpoint IP should be added to the PF interface where the tunnel needs to be created.

Server 1 Configuration

1. Check the driver version of the U25N interface using this command:
   ***Note*:** Ignore this step if the U25N X2 driver version is 5.3.3.1003.

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

   This should give the output as version 5.3.3.1003.
   ***Note*:** To install the latest sfc driver, refer to U25N Driver.
   ***Note*:** Refer to Deployment Image Flashing for flashing images to check OVS functionality.
2. Make the U25N PF interfaces 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: 5.3.3.1003

```
ifconfig <U25_interface> up
```

For example:

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

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
```

4. Allocate the number of VF to PF. Here, one VF is allocated to the PF0 and PF1 interfaces:
***Note*:** The sriov_numvfs count should be less than or equal to VF count given in sfboot command. The sriov_numvfs should be done only in legacy mode. To check 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 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*:** After the above command is executed, a VF PCIe ID and a VF interface are 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.

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.

6. Make the two VF interfaces up:

```
ifconfig <vf_interface> up
```

7. Make the PF0 and PF1 interfaces into 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
```

8. Running the above command creates a VF representor interface corresponding to each PF interface. The VF representor interface name will be the PF interface name along with _0.
   ***Note*:** Here, the 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>`.

```
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
```

9. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed to the next 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=10.16.0.2 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
```

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

```
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
```

12. Make the two bridges up:

```
ifconfig <bridge_name> up
```

For example:

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

13. Print a brief overview of the database contents:

```
ovs-vsctl show
```

14. Refer to VM Installation to make the virtual machine up.

Server 2 Configuration

1. Check the driver version of the U25N interface using this command:
   ***Note*:** Ignore this step if the U25N X2 driver version is 5.3.3.1003.

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

   This should give the output as version 5.3.3.1003.
   ***Note*:** To install the latest sfc driver, refer to U25N Driver.
   ***Note*:** Refer to Deployment Image Flashing for flashing images to check OVS functionality.

2. Make the U25N PF interfaces 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: 5.3.3.1003

```
ifconfig <U25_interface> up
```

   For example:

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

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
```

4. Allocate the number of VF to PF. Here, one VF is allocated to the PF0 and PF1 interfaces:
   ***Note*:** The sriov_numvfs count should be less than or equal to VF count given in sfboot command. The sriov_numvfs should be done only in legacy mode. To check 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 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*:** After the above command is executed, a VF PCIe ID and a VF interface are 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.

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.
6. Make the two VF interfaces up:

```
ifconfig <vf_interface> up
```

7. Make the PF0 and PF1 interfaces into 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
```

8. Running the above command creates a VF representor interface corresponding to each PF interface. The VF representor interface name will be the PF interface name along with `_0`.
   ***Note*:** Here, the 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.
   Now make the VF representor interfaces up using the ifconfig command:

```
ifconfig <vf_rep_interface> up
```

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

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

```
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
```

12. Make the two bridges up:

```
ifconfig <bridge_name> up
```

For example:

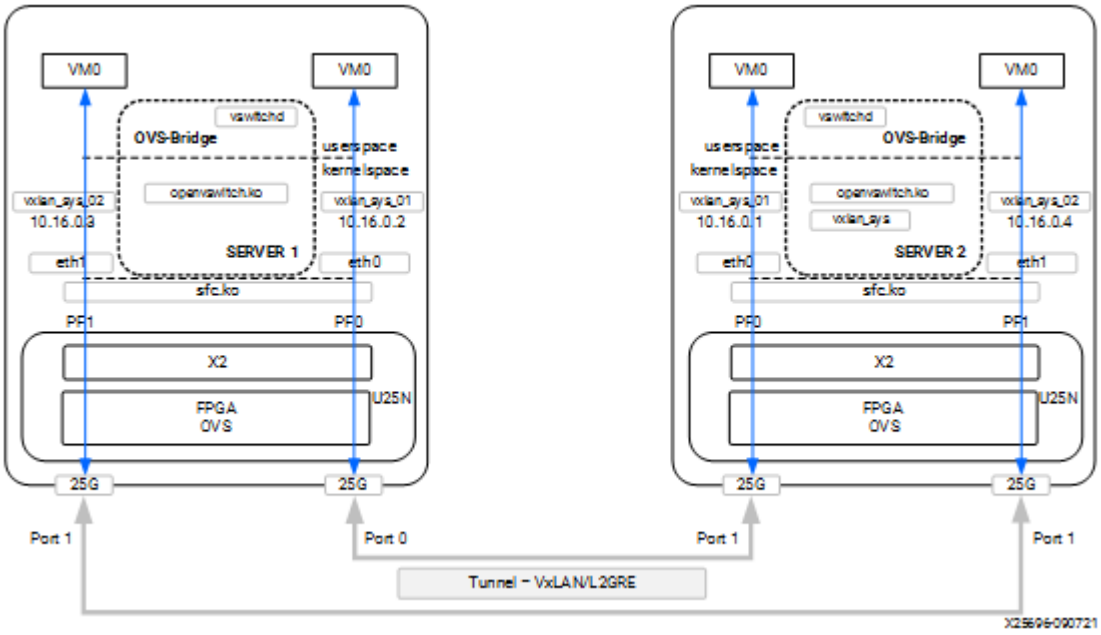
```
ifconfig br0 up
ifconfig br1 up
```

13. Print a brief overview of the database contents:

```
ovs-vsctl show
```

14. Refer to VM Installation to make the virtual machine up. After the VM is up, do refer to Functionality Check check functionality.

Figure 12: **Tunneling/Detunneling**



4.2.7 OVS Configuration

1. Export the OVS path:

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

2. Stop OVS and remove the database for removing old configurations:

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

3. Start OVS:

```
ovs-ctl start
```

4. Enable hardware offload:

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

5. After adding the policy, restart OVS:

```
ovs-ctl restart
```

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
```

7. Obtain 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
```

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.

9. Add bridge to OVS:

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

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

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

```
ping <remote_ip>
```

2. After the ping occurs, do an iperf:
   ***Note*:** For VXLAN and L2GRE, set the MTU size to 1400 before running iperf or pktgen on a particular interface.

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

3. Run iperf3 -s on the host device [iperf server].
4. Run iperf3 -c on a remote device [iperf client].

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

4.2.9 Statistics

Refer to Statistics to get commands for statistics.

4.3 IPsec

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 userspace. 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

4.3.2 Classification Fields (Matches)

Encryption

Key

1. IPv4 source address
2. IPv4 destination address
3. IP4 protocol Action

Action

1. Action flag
2. SPI
3. Key
4. IV

Decryption

Key

1. IPv4 source address
2. IPv4 destination address
3. SPI Action

Action

1. Decryption key
2. IV

4.3.3 strongSwan Installation

Do the following 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.

1. Refer to Basic Requirements and Component Versions Supported for the required OS/software version.
2. Check the version of the strongSwan Debian package. If it shows the version as strongswan_5.8.4-1, ignore this step.
   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
```

3. After installation of the strongSwan package, create a CA certificate. For this, create a CA certificate in one server and copy the same to another server.

Server 1 Configuration

1. Generate a self-sign 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
```

2. After the key and certificate is generated in server 1, copy it to server 2 in the same path.
   a. Copy the file `strongswanKey.pem` in the path `/etc/ipsec.d/private/` in the first server to the second server in the same path.
   b. Copy the file `strongswanCert.pem` in the path `/etc/ipsec.d/cacerts/strongswanCert.pem` in the first server to the second server in the same path.
   After finishing the above, create a key pair and certificate for each server separately as root.
3. Generate the key pair and certificate in server 1 as root:

```
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
```

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: White space is present between `:` and `RSA`.

Server 2 Configuration

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
```

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: White space is present between `:` and `RSA`.

Server 1: Steps to Run IPsec

1. Check 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 5.3.3.1003.
This should give the output as version: 5.3.3.1003.
***Note*:** To install the latest sfc driver, refer to U25N Driver.

2. Make the PF1 interface up:

```
ifconfig <interface_2> up
```

For example, `ifconfig U25eth1 up`.

3. Make the PF0 interface up and assign a tunnel endpoint IP to it:

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

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

4. Make the PF0 interface into 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`.

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

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
```

7. Add the PF1 interface OVS bridge:

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

8. Print a brief overview of the database contents:

```
ovs-vsctl show
```

9. Make the bridge up:

```
ifconfig <bridge_name> up
```

For example, `ifconfig br0 up`.

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`.

11. Start IPsec:

```
sudo ipsec restart
```

Server 2: Steps to Run IPsec

1. Check 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 5.3.3.1003.
This should give the output as version 5.3.3.1003.
***Note*:** To install the latest sfc driver, refer to U25N Driver.

2. Make the PF1 interface up:

```
ifconfig <interface_2> up
```

For example, `ifconfig U25eth1 up`.

3. Make the PF0 interface up and assign a tunnel IP to it:

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

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

4. Make the PF0 interface 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`.

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

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
```

7. Add PF1 interface OVS bridge:

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

8. Print a brief overview of the database contents:

```
ovs-vsctl show
```

9. Make the bridge up.

```
ifconfig <bridge_name> up
```

For example, `ifconfig br0 up`.

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`.

11. Start IPsec:

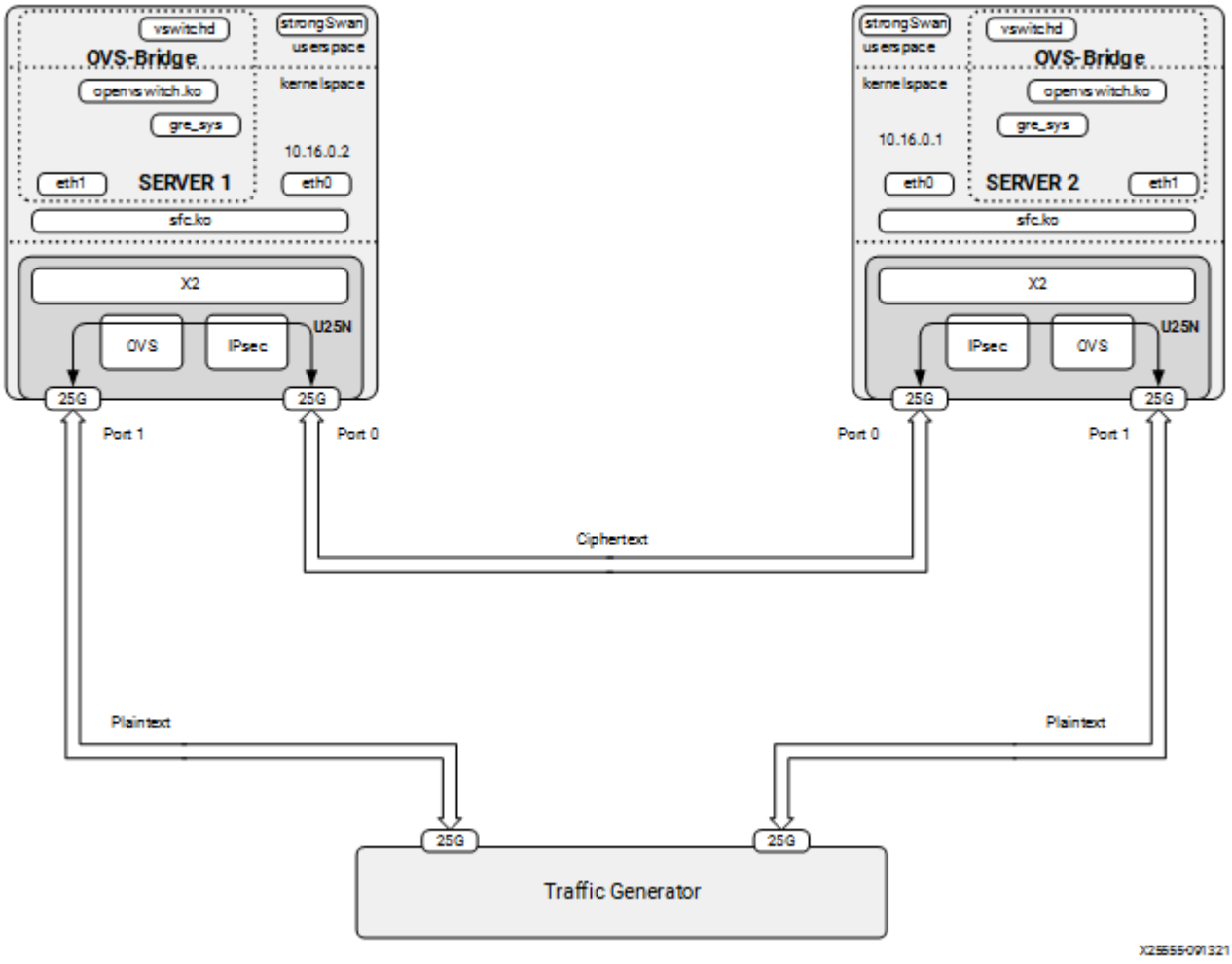
```
sudo ipsec restart
```

Refer to Functionality Check to check OVS functionality.

4.3.4 Statistics

Refer to Statistics to get IPsec statistics.

Figure 13: **IPsec + OVS**



4.4 Stateless Firewall

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

4.4.1 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
```

4.4.2 nftables

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

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
```

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

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

2. Refer to 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.

4.4.3 Classification Fields (Matches)

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

Key:

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

Action

1. drop
2. accept

Driver Installation

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

The prerequisites for the installation are as follows:

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

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.

2. Create a table

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

   For example, `nft add table netdev filter`.

3. Create 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.

4. Add 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`.

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
```

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>
```

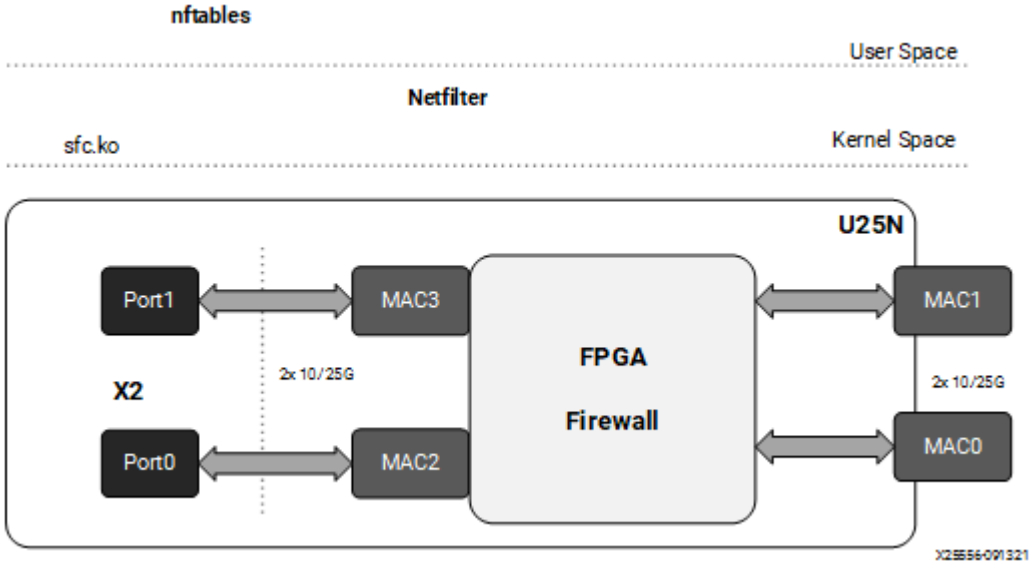
   For example, `nft delete chain netdev filter input1`.
   c. Deleting a specific rule with a handle:

```
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.

Figure 14: **Firewall**



4.5 Statistics

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

4.5.1 OVS Commands

- To print a brief overview of the database contents:

```
ovs-vsctl show
```

- To show the datapath flow entries:

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

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

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

- To show the OVS datapath flows:

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

- To show which flows are offloaded or not:

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

4.5.2 MAE Rules

- To show offloaded rules now 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`.

- To show default rules now present in the MAE:

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

4.5.3 IPsec Statistics

- `sudo swanctl --stats`
- Use the `ip xfrm show` command to display IPsec offload security association.

4.5.4 External MAC counter

TBD

4.6 Debug Commands

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

```
dmesg
```

2. Logs from the U25N hardware are saved to a file. Reading a file from the X86 host produces logs. The logs are collected from the internal processing subsystem and exported to the host at intervals. These logs are populated when switchdev mode is enabled. The path to read the logs in host is `/var/log/ps_dmesg.txt`.
3. sfreport: This is a command line utility that generates a diagnostic log file providing diagnostic data about the server and Solarflare adapters.

Chapter 5

DPDK

Describes the DPDK compilation steps:

1. Download the dpdk git repositories from http://www.dpdk.org/browse.
2. Uncompress 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. After compilation is successful, 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.
   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. 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. Make sure the VF interface has been made down and unbinded 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
```

5. 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
```

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.

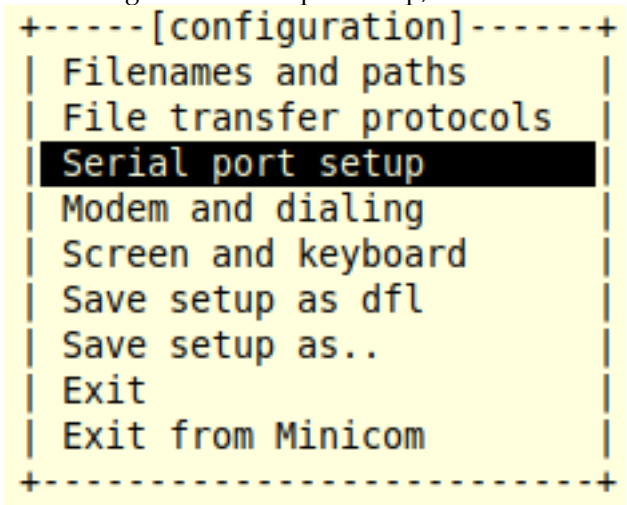
6.1 Entering into U-Boot Mode

***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 shown 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.

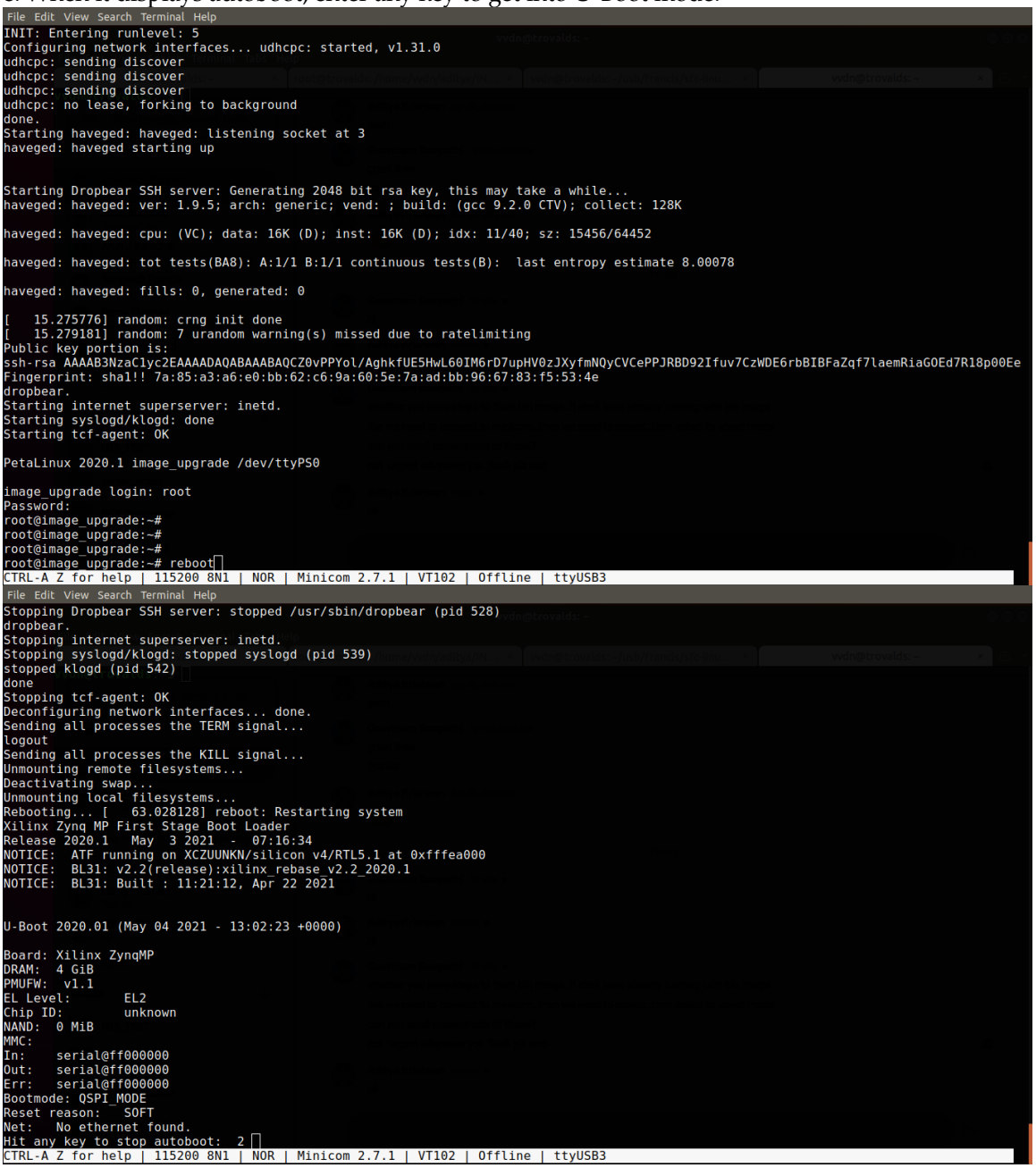
```
sudo minicom


Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Aug 13 2017, 15:25:34.
Port /dev/ttyUSB3, 14:58:39

Press CTRL-A Z for help on special keys
```

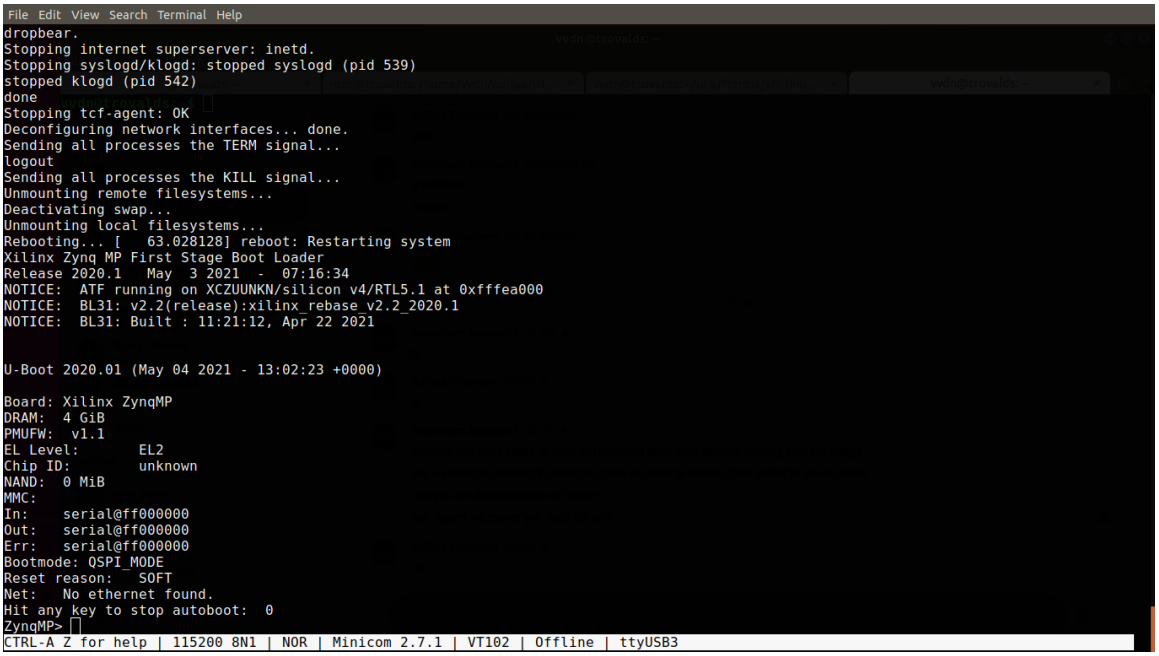
c. When prompted for a login and password, enter the following:

- login: root
- password: root

```
File  Edit  View  Search  Terminal  Help
[    5.319665] CONTROLLER application
Configuring packages on first boot....
 (This may take several minutes. Please do not power off the machine.)
Running postinst /etc/rpm-postinsts/100-sysvinit-inittab...
update-rc.d: /etc/init.d/run-postinsts exists during rc.d purge (continuing)
INIT: Entering runlevel: 5
Configuring network interfaces... udhcpc: started, v1.31.0
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending discover
udhcpc: no lease, forking to background
done.
Starting haveged: haveged: listening socket at 3
haveged: haveged starting up


Starting Dropbear SSH server: Generating 2048 bit rsa key, this may take a while...
haveged: haveged: ver: 1.9.5; arch: generic; vend: ; build: (gcc 9.2.0 CTV); collect: 128K

haveged: haveged: cpu: (VC); data: 16K (D); inst: 16K (D); idx: 11/40; sz: 15456/64452

haveged: haveged: tot tests(BA8): A:1/1 B:1/1 continuous tests(B):  last entropy estimate 8.00078

haveged: haveged: fills: 0, generated: 0

[   15.275776] random: crng init done
[   15.279181] random: 7 urandom warning(s) missed due to ratelimiting
Public key portion is:
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCZ0vPPYol/AghkfUE5HwL60IM6rD7upHV0zJXyfmNQyCVCePPJRBD92Ifuv7CzWDE6rbBIBFaZqf7laemRiaGOEd7R18p00Ee
Fingerprint: sha1!! 7a:85:a3:a6:e0:bb:62:c6:9a:60:5e:7a:ad:bb:96:67:83:f5:53:4e
dropbear.
Starting internet superserver: inetd.
Starting syslogd/klogd: done
Starting tcf-agent: OK

PetaLinux 2020.1 image_upgrade /dev/ttyPS0

image_upgrade login:
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB3
```

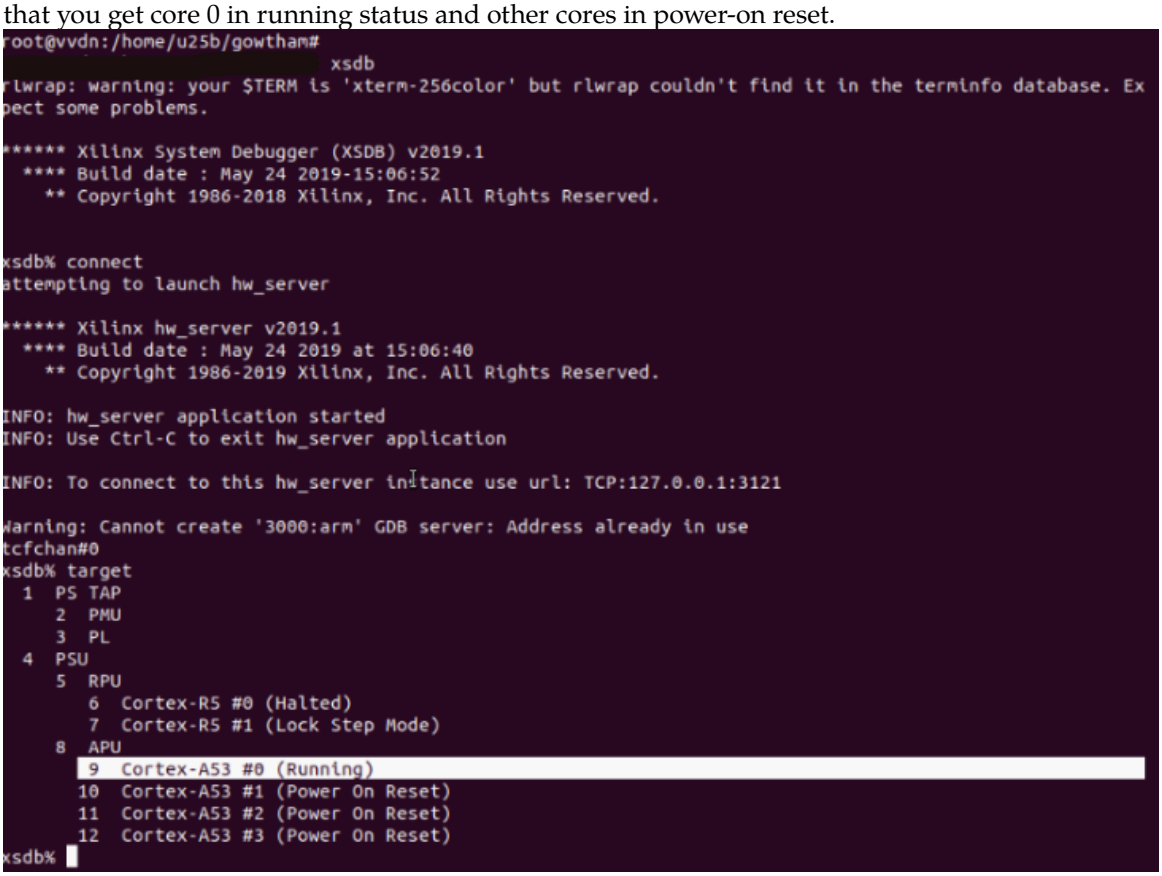
d. After logging in, do a reboot.
The system starts executing the reboot command.
e. When it displays autoboot, enter any key to get into U-Boot mode.





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.



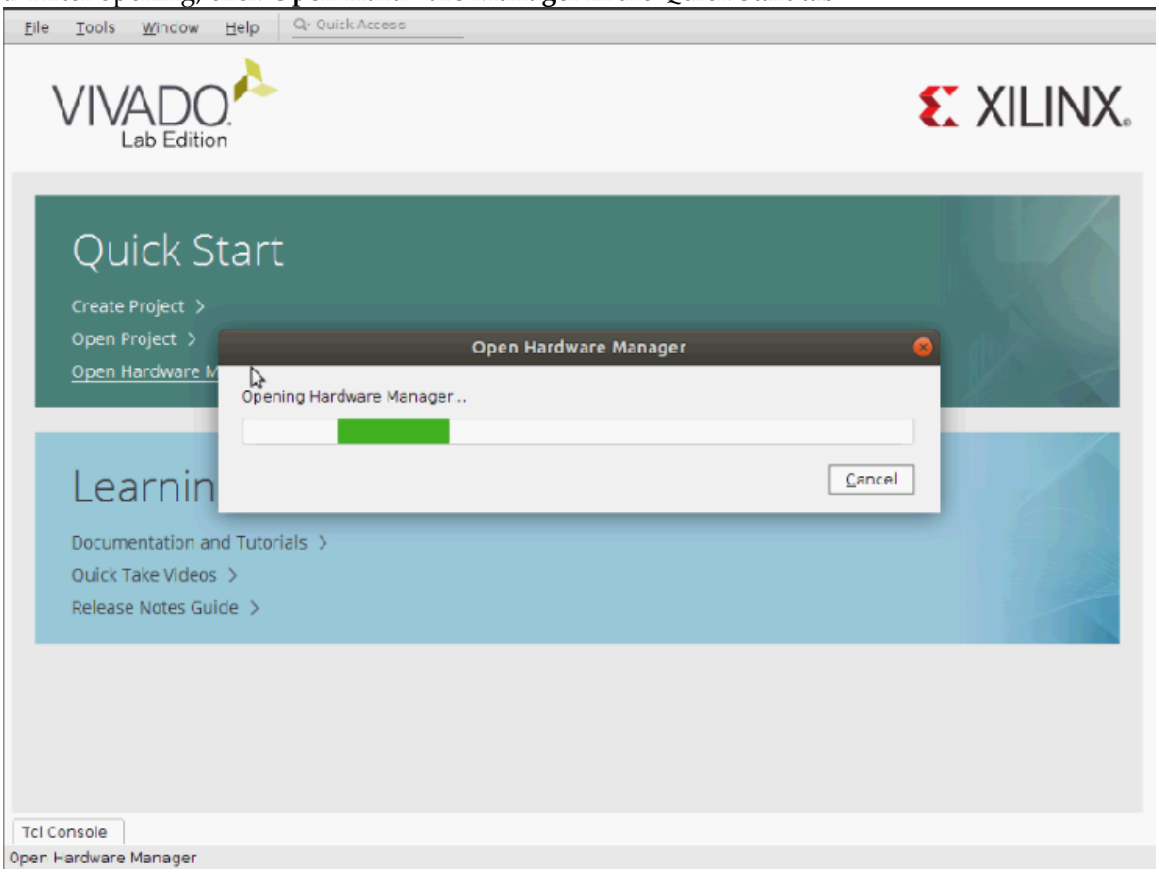
   b. Exit xsdb by entering **exit** in the xsdb console.

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

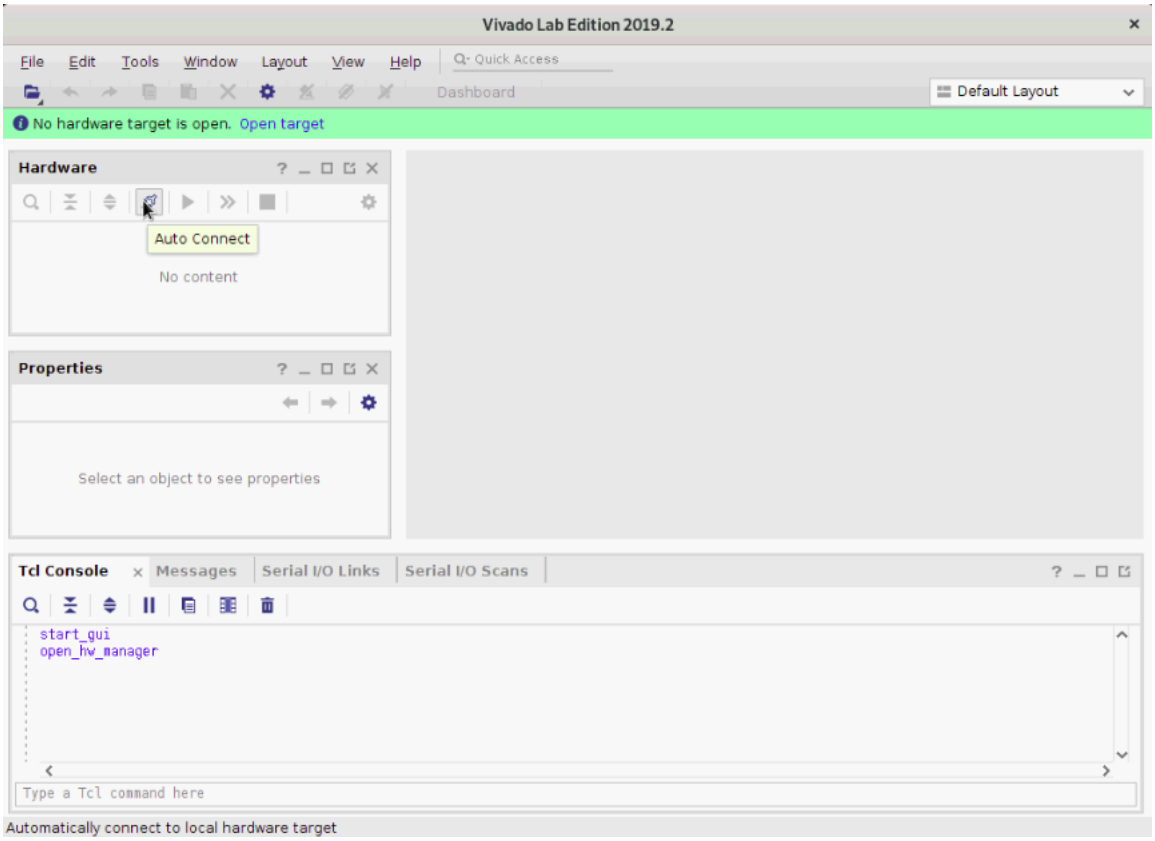
```
sudo <path>Vivado_Lab/2019.2/bin/vivado_lab
```

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

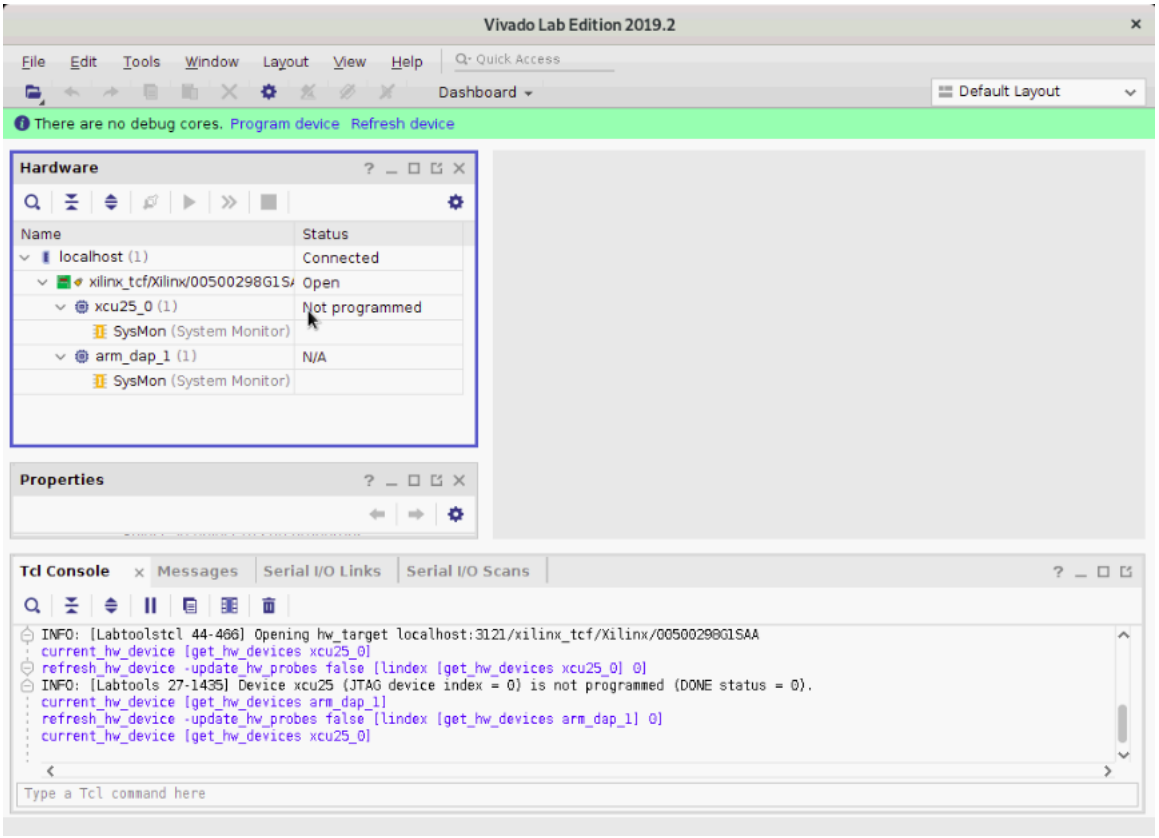
ware opens, carry out the following steps:
a. After opening, click **Open Hardware Manager** in the Quick Start tab.



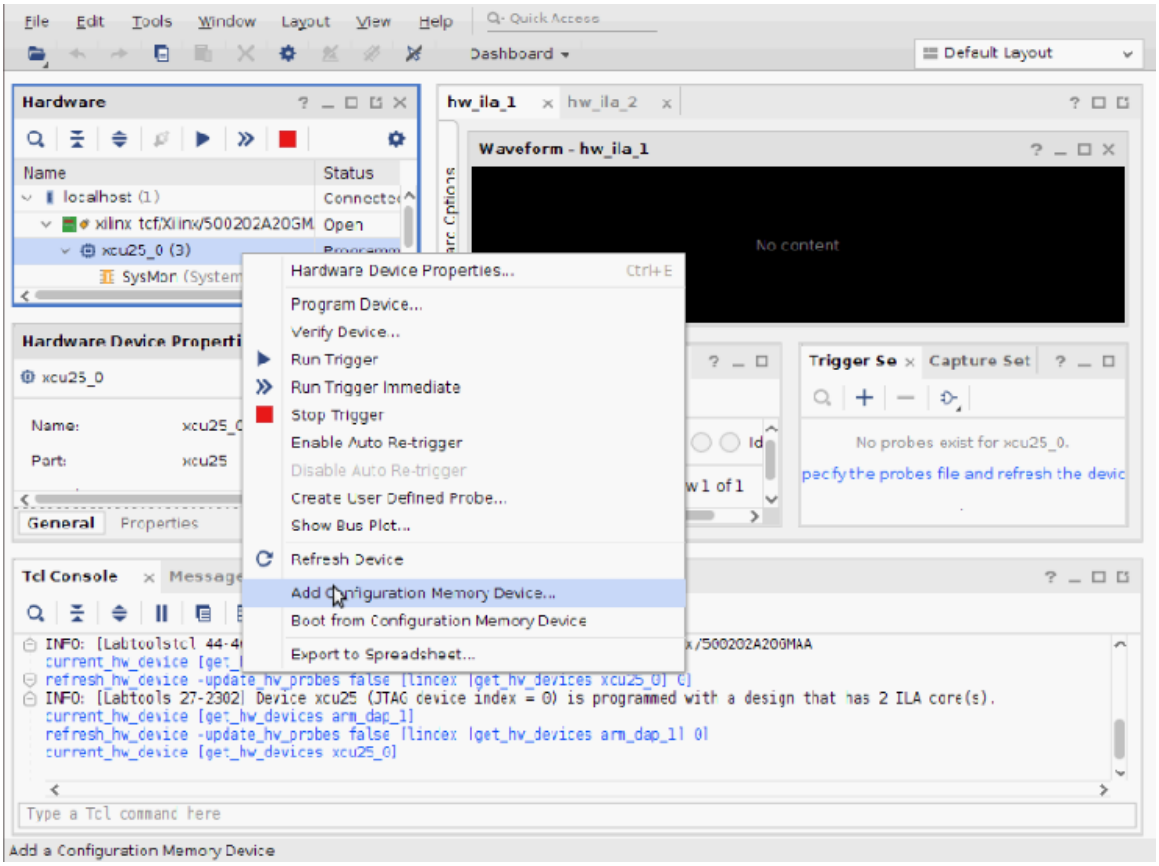
b. Click **Auto Connect**.



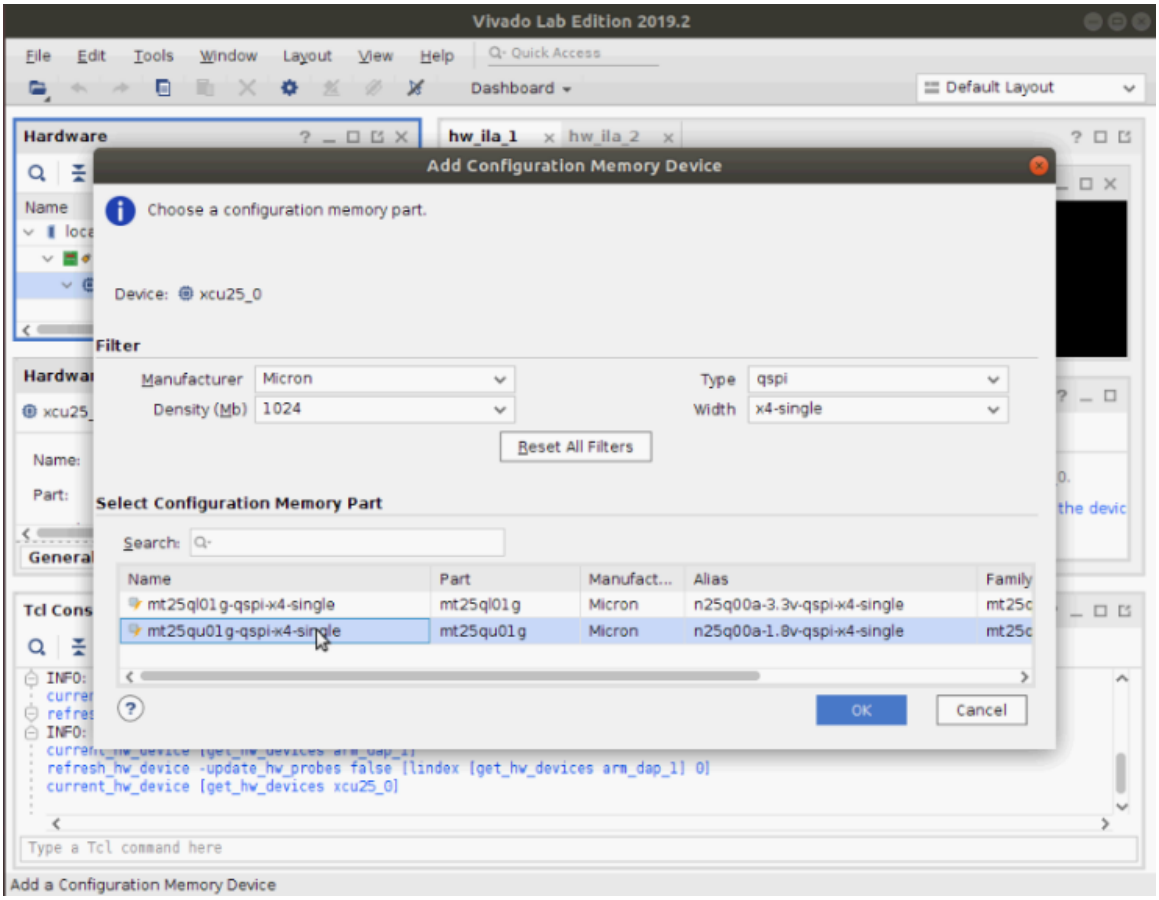
c. Verify whether the U25N SmartNIC is detected properly.



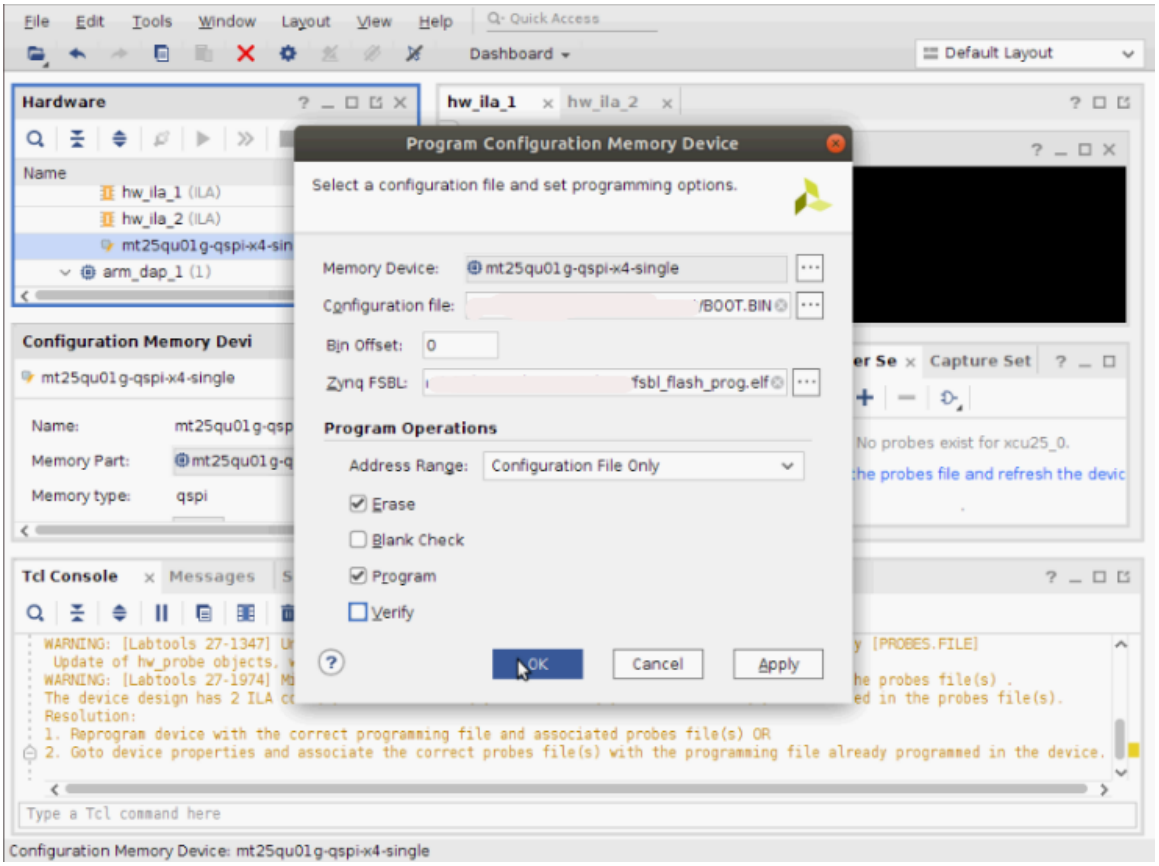
d. Select the SmartNIC as shown below, right-click, and select **Add Configuration Memory Device**.



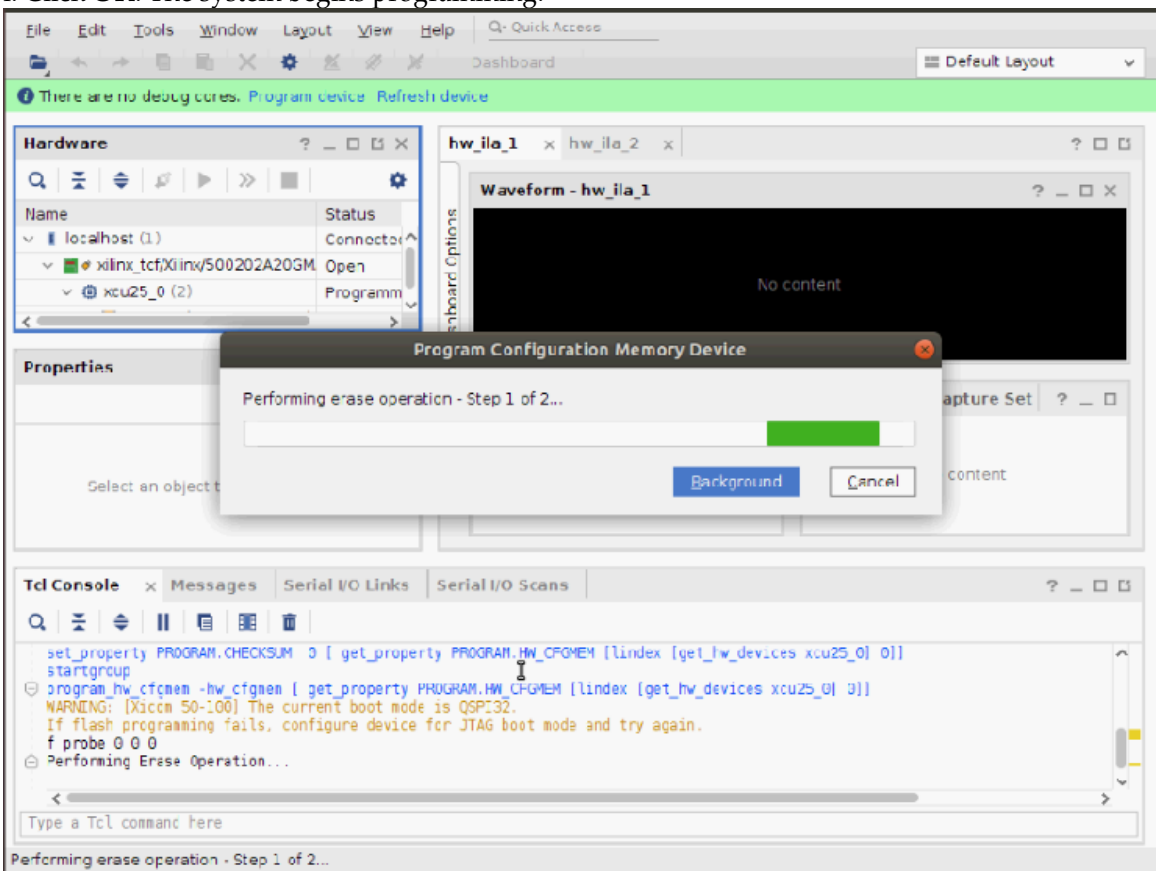
e. Search by clicking **mt25qu01g-qspi-x4-single** in the Search tab.



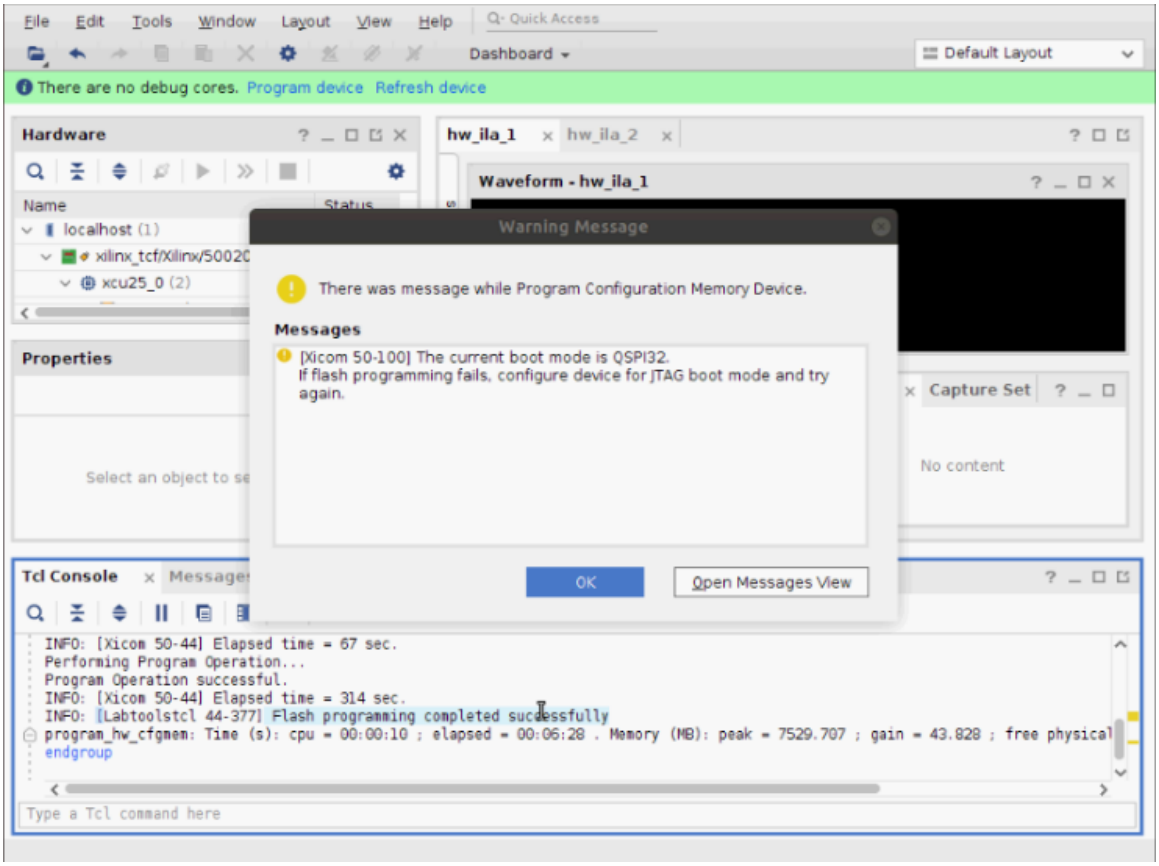
f. After making the above selections, a dialog box appears. Click **OK** and continue.
g. Select the configuration file (`BOOT.BIN`), and first stage bootloader file (`fsbl_flash_prog.elf`).
h. Uncheck Verify under Program operations. It will be time consuming otherwise.



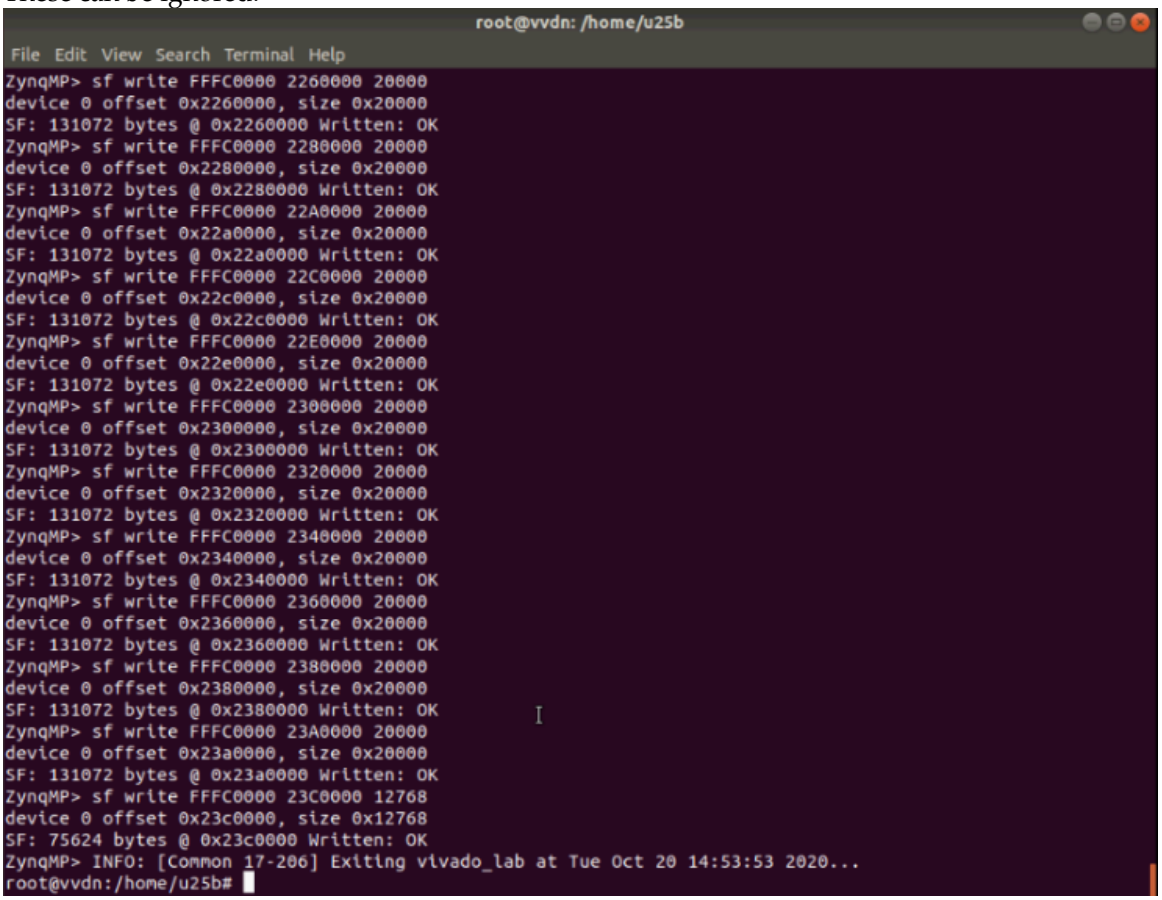
i. Click **OK**. The system begins programming.



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).

VM Installation

***Note*:** This section only needs to be done for VM cases. The following configuration is 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
```