LL10GEMAC-IP with AAT QDMA Demo Instruction

LL10GEMAC-IP with AAT QDMA Demo Instruction

Rev1.00 2-Dec-2024

1 Overview

This document provides detailed instructions for configuring the Alveo accelerator card and setting up the test environment to run the customized Accelerated Algorithmic Trading (AAT) demo from Design Gateway – DG AAT QDMA demo. This demo is a modified version of AMD's original AAT demo, where the development platform has been migrated from Vitis to Vivado. Furthermore, the DG AAT QDMA demo is specifically designed to achieve lower latency by utilizing the LL10GEMAC-IP from Design Gateway, replacing the 10G/25G Ethernet subsystem.

Detailed information about the DG AAT QDMA demo is available in the reference design document:

https://dgway.com/products/IP/Lowlatency-IP/ll10gemac-ip-aat-qdma-refdesign-amd/

Latency details for the LL10GMEAC-IP can be found in its datasheet:

https://dgway.com/products/IP/Lowlatency-IP/dg_ll10gemacip_data_sheet_xilinx_en/

The demo operates on the Alveo accelerator card and demonstrates performance on a 10G Ethernet connection. The Gigabit Ethernet connectors available on each Alveo card vary depending on the model's capabilities.

- U250, U55C : Equipped with two QSFP28 ports, supporting up to four 10G Ethernet channels per QSFP28.
- U50 : Equipped with one QSFP28 port, supporting up to four 10G Ethernet channels.
- X3522 : Equipped with two DSFP28 ports, supporting up to two 10G Ethernet channels per DSFP28.

In this demo, two 10G Ethernet channels are required:

1) Market Data Transmission: Transmits sample market data using the UDP protocol.
2) Order Transmission: Handles order data transmission using FIX over TCP.

To set up the system, a target PC with two 10G Ethernet ports is required. The sample market data is transmitted using the "tcpreplay" tool, while order reception is monitored by opening a TCP port on the target system. The demo on the Alveo accelerator is initiated by executing the "aat_qdma_exe" application.

The following test environment was configured to produce the results presented in this document.

1) Supported Alveo cards: U50, U250, U55C, and X3522.
2) Host system for Alveo accelerator card: Turnkey accelerator system (TKAS-D2101). Detailed specifications are available at https://dgway.com/solutions.html.
3) The Vivado Design Suite installed on the host system to program the Alveo card.
4) 10G Ethernet cable
   - U50/U250/U55C card: QSFP+ to four SFP+ breakout cable: https://www.sfpcables.com/5-meter-40g-qsfp-to-4-sfp-aoc-cable-om3-mmf-cisco-oem-compatible.
   - X3522 card: 2xSFP+ Active Optical Cable (AOC): https://www.10gtek.com/10gsfp+aoc.
5) Programming cable
   - U250/U55C card : Micro-USB cable.
   - U50 card : Alveo programming cable.
   - X3522 card : Alveo Debug Kit (ADK2).
6) Target system is configured with the following specifications.
   - Operation System: Ubuntu 20.04 LTS Server.
   - Market Data: Sample market data file (cme_input_arb.pcap).
   - Packet Replay: "tcpreplay" package for transmitting market data.
   - Ethernet Ports: Two 10G Ethernet ports using Intel X710-DA2 adapters. https://ark.intel.com/content/www/us/en/ark/products/83964/intel-ethernet-converged-network-adapter-x710da2.html

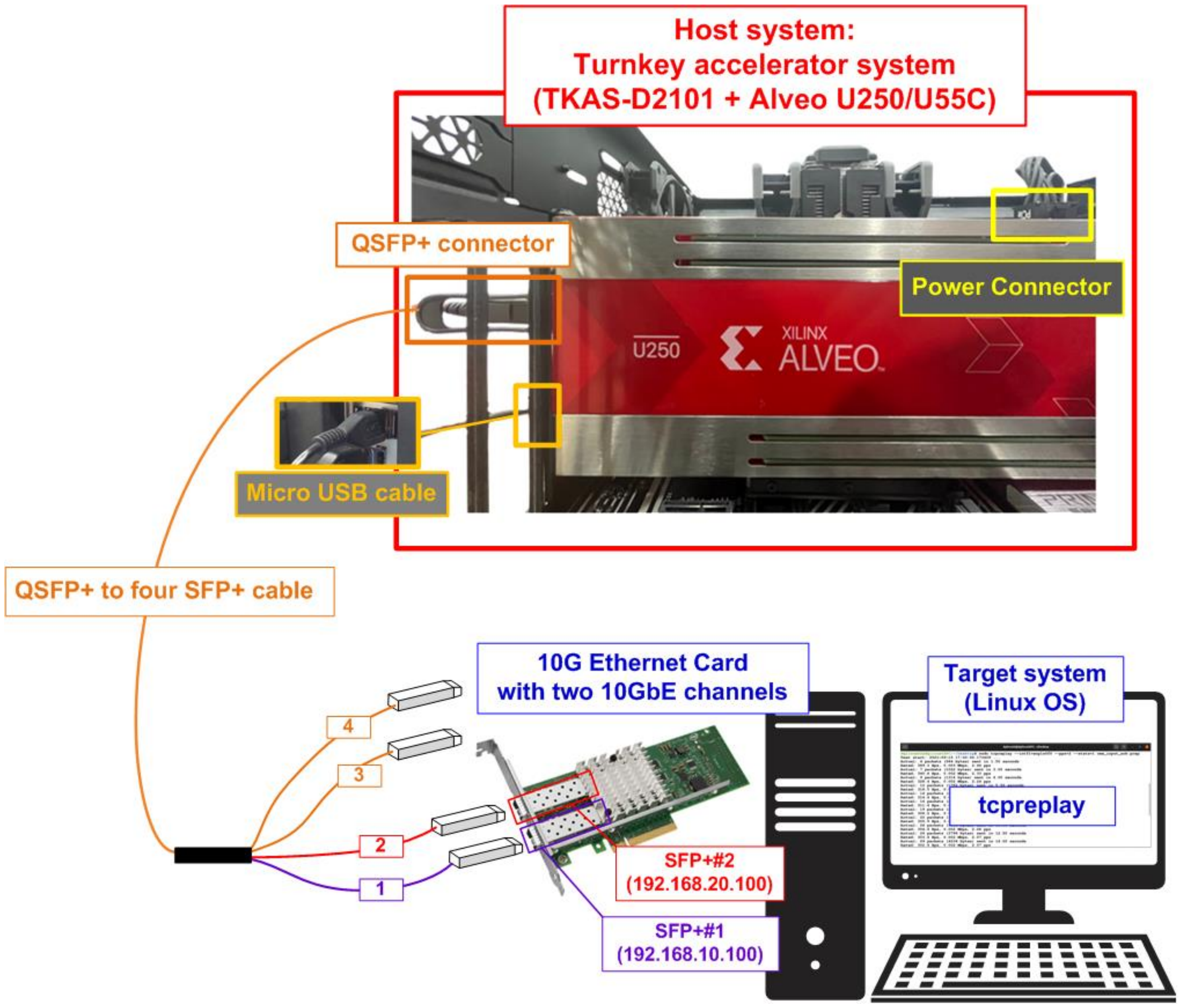


Figure 1 LL10GEMAC-IP with AAT QDMA Demo using Alveo U250/U55C Card

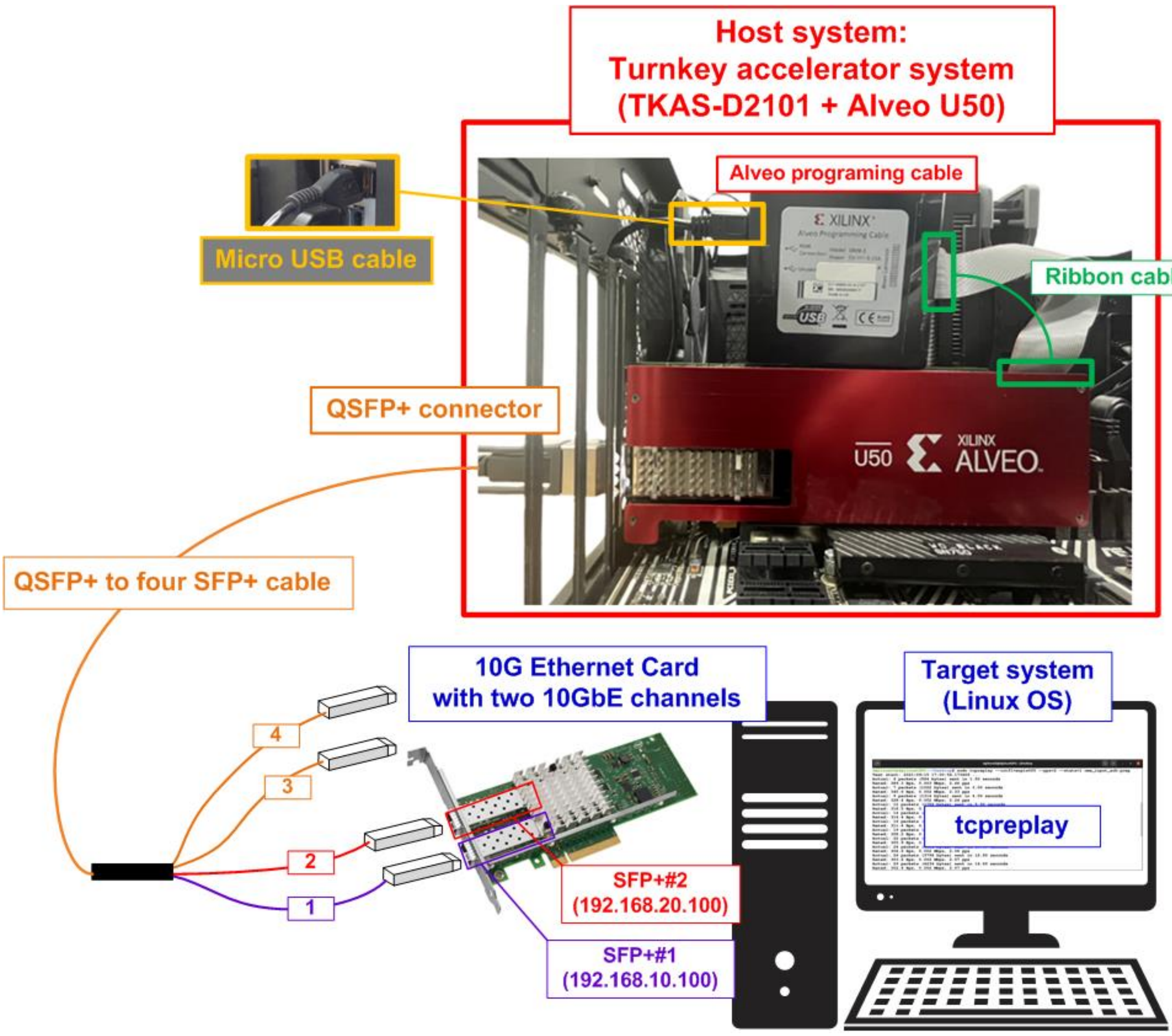


Figure 2 LL10GEMAC-IP with AAT QDMA Demo using Alveo U50 Card

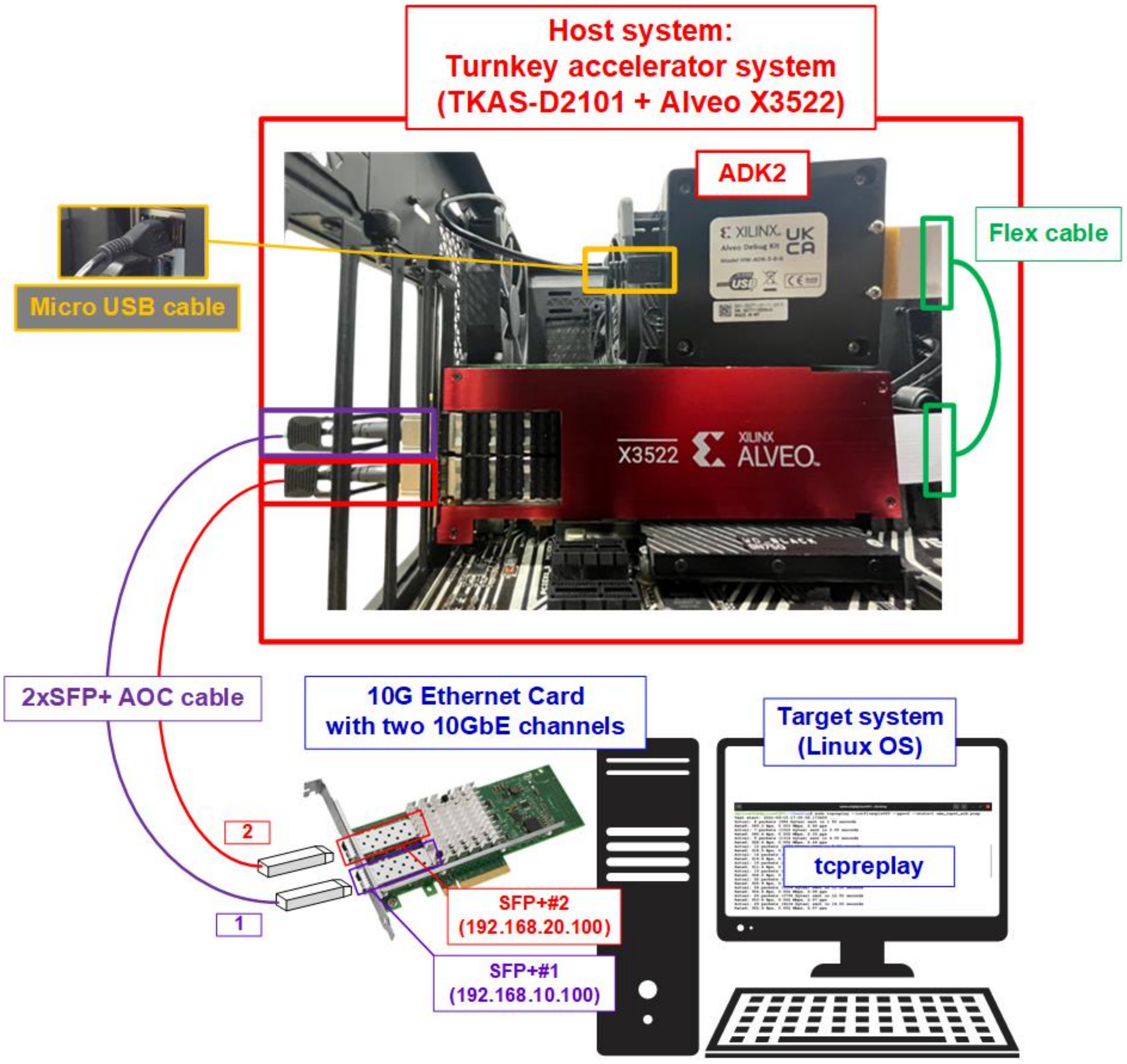


Figure 3 LL10GEMAC-IP with AAT QDMA Demo using Alveo X3522 Card

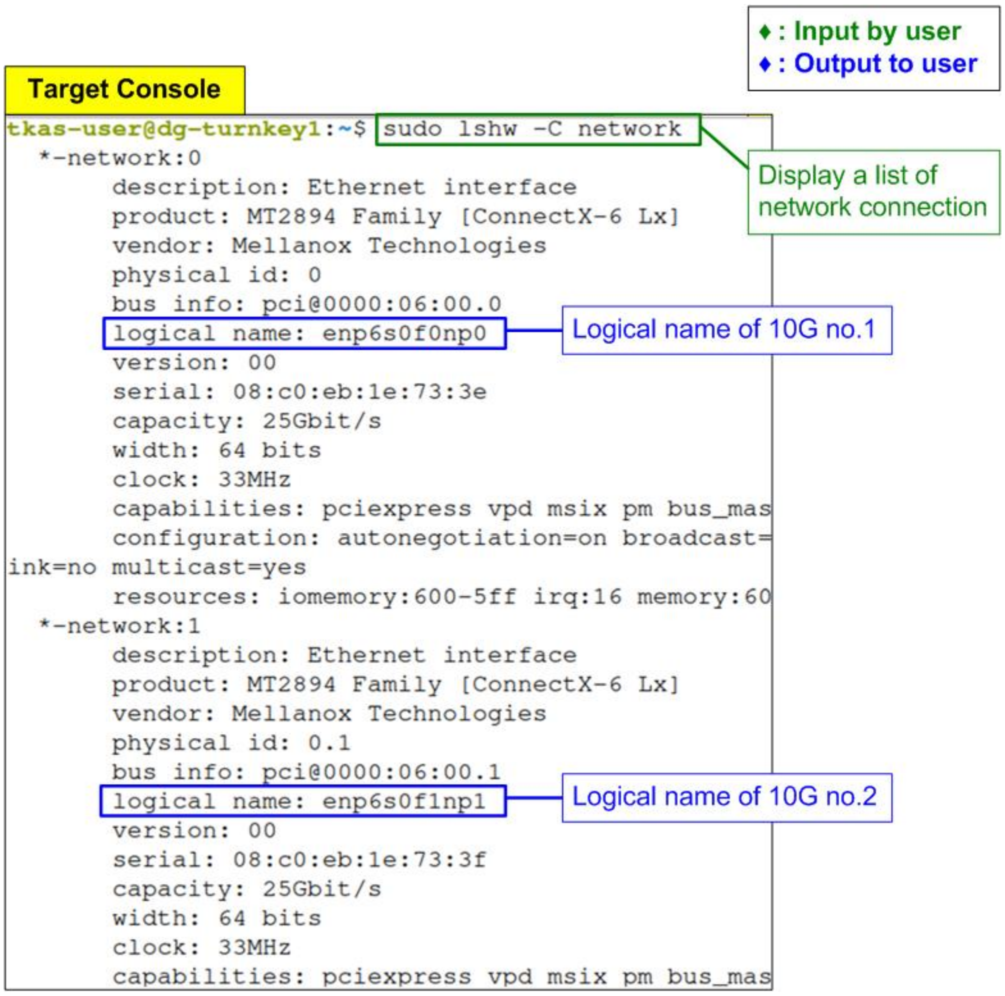
2 Target System Setup

This section provides step-by-step instructions for preparing the target system, equipped with two 10G Ethernet ports, to transfer market data and order packets with the Alveo accelerator card. The system runs Ubuntu 20.04 LTS Server OS.

2.1 IP Address Configuration for Two 10G Ethernet Ports

First, identify the logical names of the two 10G Ethernet ports, which connect to the SFP+#1 and SFP+#2 cables. These logical names may vary based on your test environment, so it is important to configure the correct IP address for the SFP+#1 and SFP+#2 connections.

1) Open a Linux terminal and use the following command to list the logical names of the 10G Ethernet ports: “lshw -C network”.



```
tkas-user@dg-turnkey1:~$ sudo lshw -C network
  *-network:0
       description: Ethernet interface
       product: MT2894 Family [ConnectX-6 Lx]
       vendor: Mellanox Technologies
       physical id: 0
       bus info: pci@0000:06:00.0
       logical name: enp6s0f0np0
       version: 00
       serial: 08:c0:eb:1e:73:3e
       capacity: 25Gbit/s
       width: 64 bits
       clock: 33MHz
       capabilities: pciexpress vpd msix pm bus_mas
       configuration: autonegotiation=on broadcast=
ink=no multicast=yes
       resources: iomemory:600-5ff irq:16 memory:60
  *-network:1
       description: Ethernet interface
       product: MT2894 Family [ConnectX-6 Lx]
       vendor: Mellanox Technologies
       physical id: 0.1
       bus info: pci@0000:06:00.1
       logical name: enp6s0f1np1
       version: 00
       serial: 08:c0:eb:1e:73:3f
       capacity: 25Gbit/s
       width: 64 bits
       clock: 33MHz
       capabilities: pciexpress vpd msix pm bus_mas
```

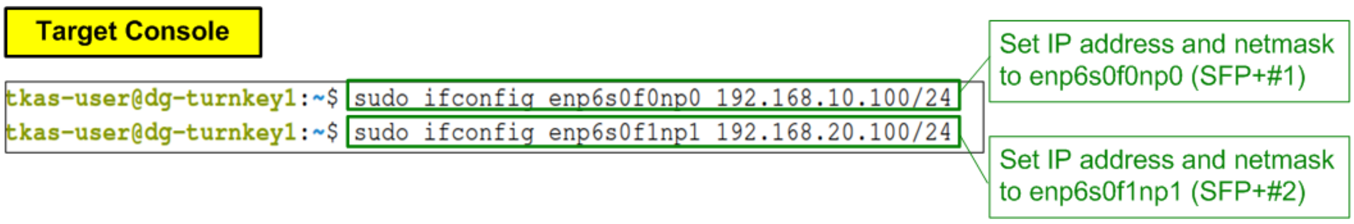
Figure 4 Display Logical Name of 10G Ethernet Ports

The output will display information about the network interfaces. For example, Figure 4 shows logical names such as “enp6s0f0np0” for SFP+#1 and “enp6s0f1np1” for SFP+#2.

2) Configure the IP address for each port using the "ifconfig" command.

- Set SFP+#1 (enp6s0f0np0) to "192.168.10.100".
- Set SFP+#2 (enp6s0f1np1) to "192.168.20.100".

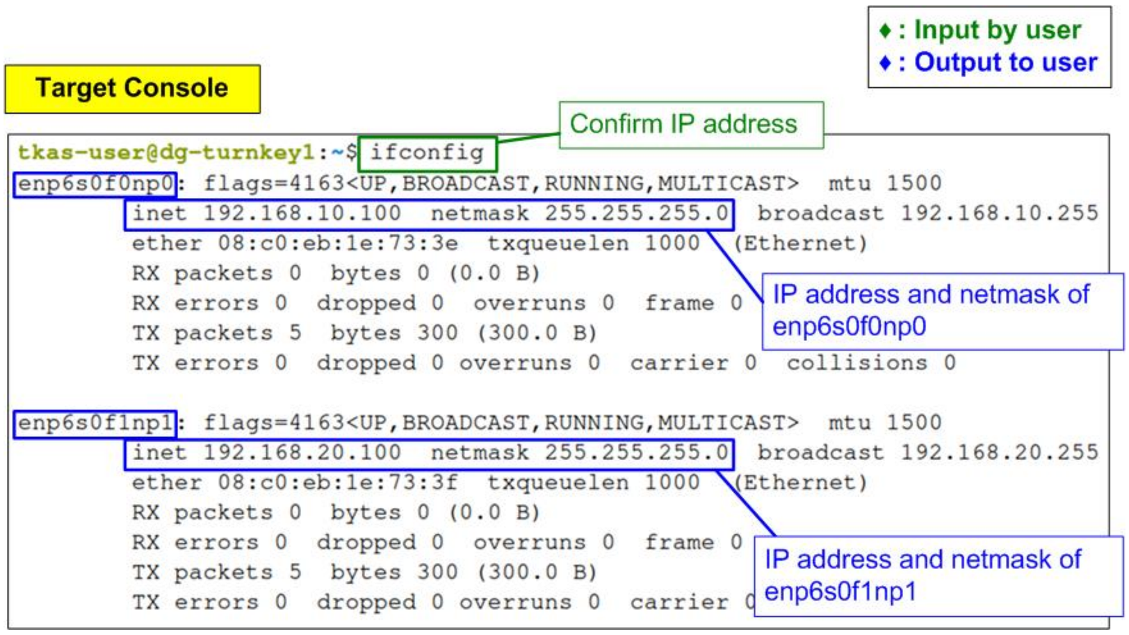
Additionally, configure the netmask to 255.255.255.0 (i.e., /24 subnet). The command format is as follows.


Target Console
Set IP address and netmask to enp6s0f0np0 (SFP+#1)
Set IP address and netmask to enp6s0f1np1 (SFP+#2)

```
tkas-user@dg-turnkey1:~$ sudo ifconfig enp6s0f0np0 192.168.10.100/24
tkas-user@dg-turnkey1:~$ sudo ifconfig enp6s0f1np1 192.168.20.100/24
```

Figure 5 Configure IP Address and Netmask

3) After configuring the IP addresses and netmask, verify the settings using the "ifconfig" command.


♦ : Input by user
♦ : Output to user
Target Console
Confirm IP address
IP address and netmask of enp6s0f0np0
IP address and netmask of enp6s0f1np1

```
tkas-user@dg-turnkey1:~$ ifconfig
enp6s0f0np0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.100  netmask 255.255.255.0  broadcast 192.168.10.255
        ether 08:c0:eb:1e:73:3e  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5  bytes 300 (300.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp6s0f1np1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.20.100  netmask 255.255.255.0  broadcast 192.168.20.255
        ether 08:c0:eb:1e:73:3f  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5  bytes 300 (300.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0
```

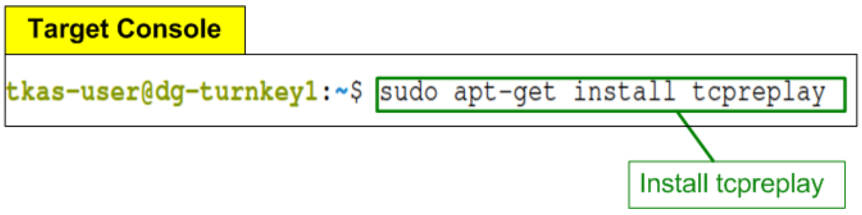
Figure 6 Verify IP Address and Netmask Setting

Ensure that both Ethernet ports are correctly assigned with their respective IP addresses and netmask values.

2.2 Installation of “tcpreplay”

To run the AAT QDMA demo, the target PC must have the “tcpreplay” tool installed, which is used to replay packet capture files over a network interface. Run the following command to install the “tcpreplay” package:

>> sudo apt-get install tcpreplay



```
tkas-user@dg-turnkey1:~$ sudo apt-get install tcpreplay
```

Figure 7 “tcpreplay” Installation

This command will install “tcpreplay”, as illustrated in Figure 7. Once the installation is completed, “tcpreplay” will be ready for use in the demo.

3 Host System Setup

This section outlines the steps to prepare the Turnkey Acceleration System (TKAS-D2101 with an Alveo card), which is the host system for running the AAT QDMA demo.

1) Install QDMA DPDK driver. Since DG AAT QDMA demo requires the packet transfer through PCIe for the Alveo card configuration, QDMA DPDK driver must be installed on the host system. The installation guide can be found on the AMD website, "QDMA DPDK Driver" on the topic of "Building QDMA DPDK Software":

   https://xilinx.github.io/dma_ip_drivers/master/QDMA/DPDK/html/build.html

2) Connect the Ethernet and programming cable between Alveo card and the target system. The detail is slightly different depending on the Alveo card installed on the host system.

   Alveo U250/U55C

   i) Insert a QSFP+ transceiver into the QSFP+ connector on the Alveo accelerator card. There are two QSFP+ channels on the U250/U55C card. Make sure to utilize the QSFP1 channel.
   ii) Connect SFP+ no.1 (IP: 192.168.10.100) and SFP+ no.2 (IP: 192.168.20.100) to the 10G Ethernet ports on the target system.
   iii) For programming the card, connect a Micro USB cable from the Alveo accelerator card to the target system.

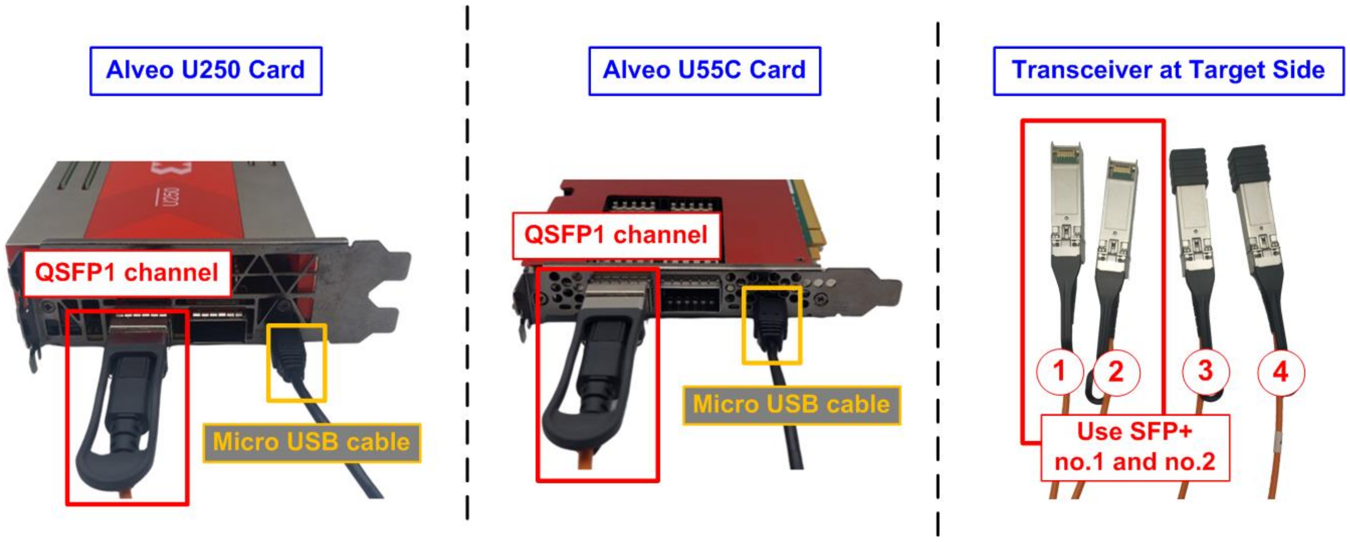


Figure 8 QSFP+ and Micro USB Cable Connection on U250/U55C

Alveo U50

i) Insert a QSFP+ transceiver into the QSFP+ connector on the Alveo accelerator card.
ii) Connect SFP+ no.1 (IP: 192.168.10.100) and SFP+ no.2 (IP: 192.168.20.100) to the 10G Ethernet ports on the target system.
iii) For programming the card, connect the Ribbon cable from the Alveo accelerator card to the Alveo Programming Module.

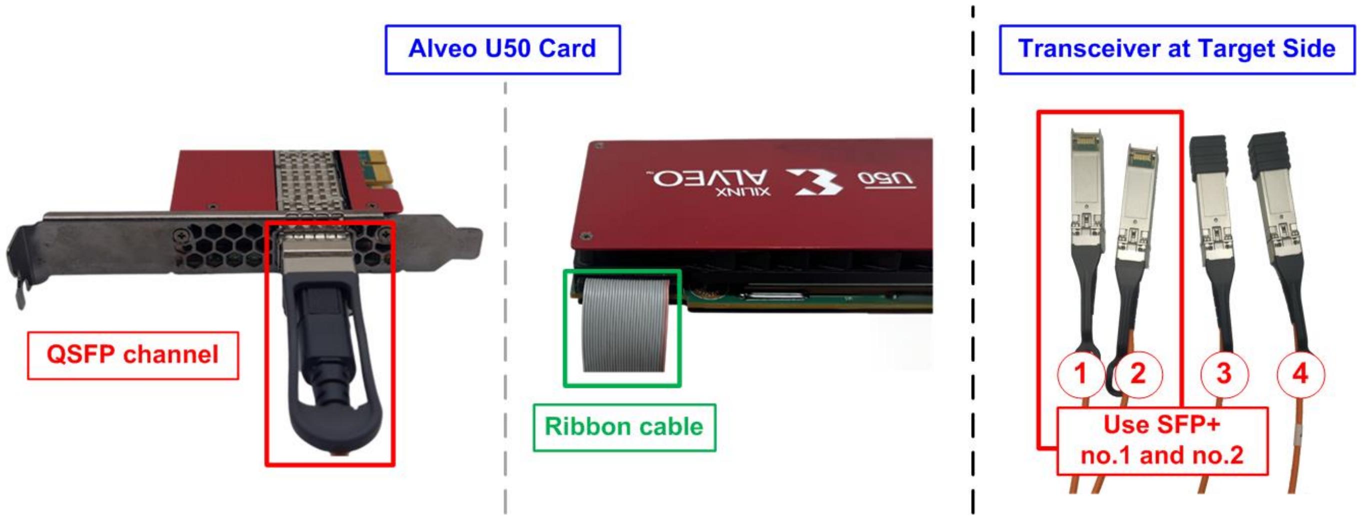


Figure 9 QSFP+ and Ribbon Cable Connection on U50

Alveo X3522

i) Insert two SFP+ transceivers into the SFP+ connectors on the Alveo accelerator card.
ii) Connect SFP+ no.1 (IP: 192.168.10.100) and SFP+ no.2 (IP: 192.168.20.100) to the 10G Ethernet ports on the target system.
iii) For programming the card, connect the Flex cable from the Alveo accelerator card to the Alveo Debug kit (ADK2). Make sure the Flex cable is connected firmly.

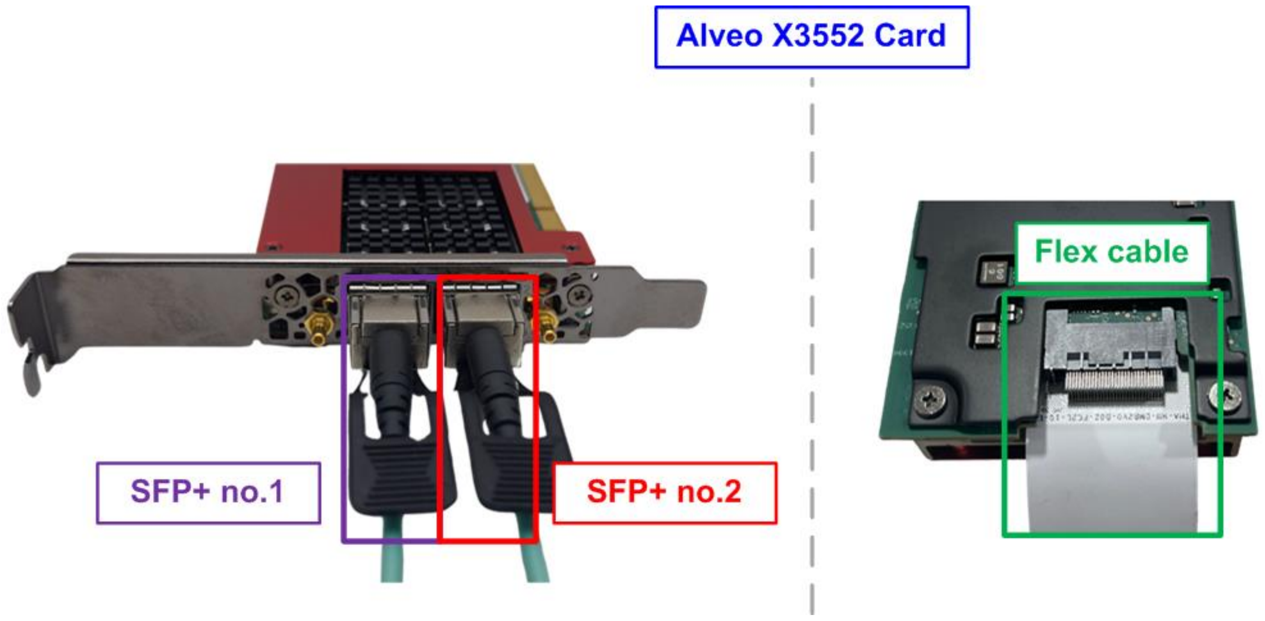


Figure 10 SFP+ and Flex Cable Connection on X3522

3) Utilize the Vivado Hardware Manager to program the Alveo card. Open the Vivado Hardware Manager and program the board with the required bit file as illustrated in Figure 11.

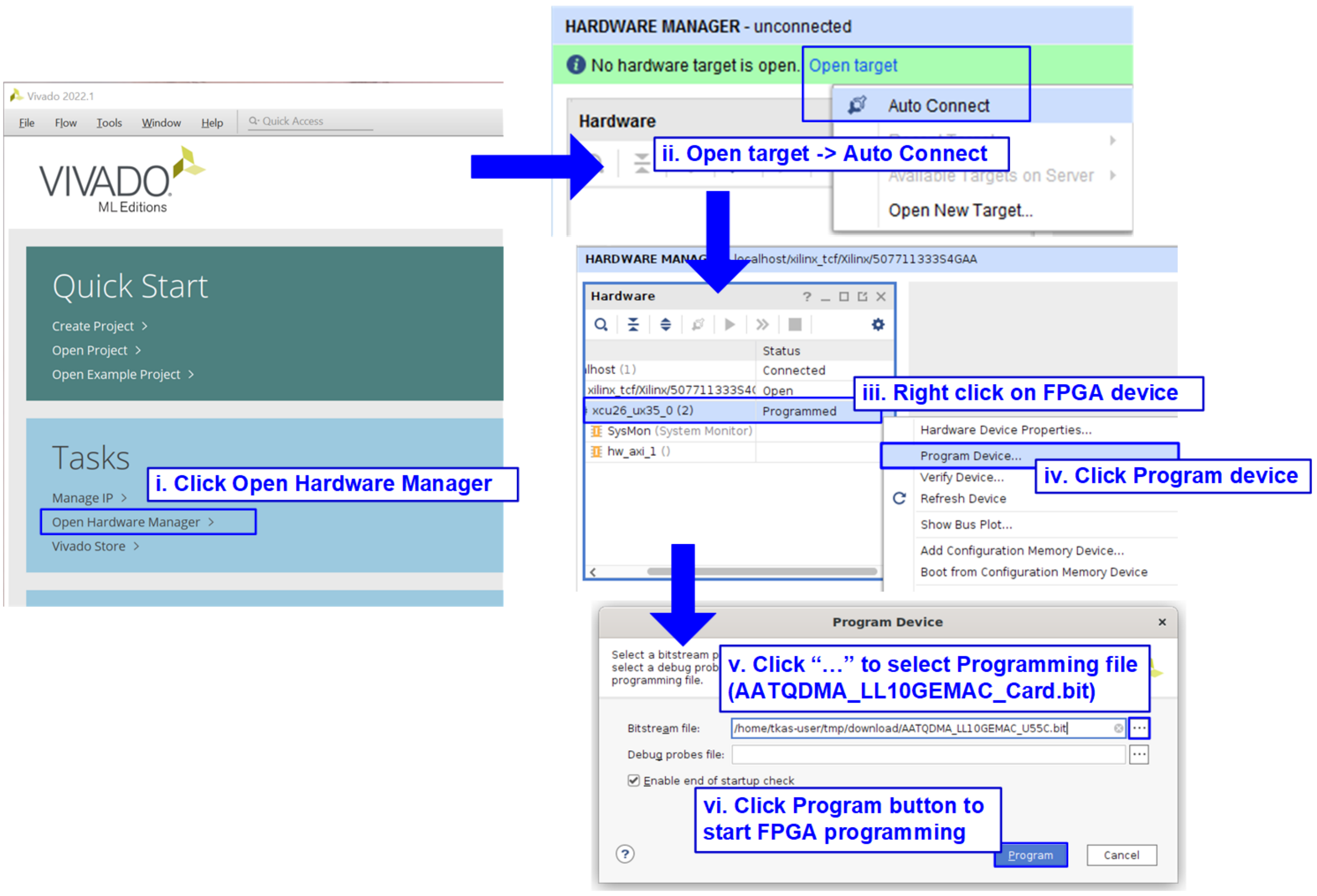
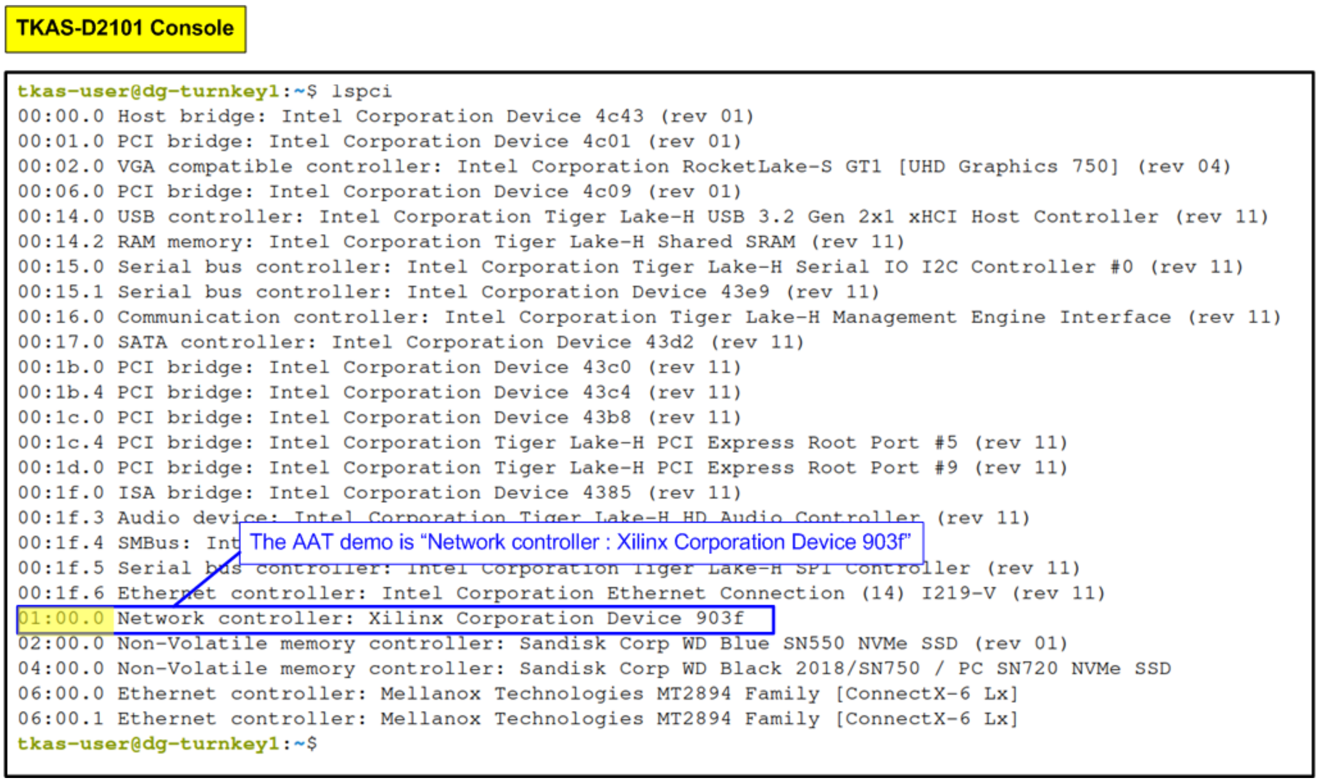

i. Click Open Hardware Manager
ii. Open target -> Auto Connect
iii. Right click on FPGA device
iv. Click Program device
v. Click "…" to select Programming file (AATQDMA_LL10GEMAC_Card.bit)
vi. Click Program button to start FPGA programming
Quick Start
Create Project
Open Project
Open Example Project
Tasks
Manage IP
Open Hardware Manager
Vivado Store
HARDWARE MANAGER - unconnected
No hardware target is open. Open target
Auto Connect
Open New Target...
Hardware Device Properties...
Program Device...
Verify Device...
Refresh Device
Show Bus Plot...
Add Configuration Memory Device...
Boot from Configuration Memory Device
Program Device
Bitstream file: /home/tkas-user/tmp/download/AATQDMA_LL10GEMAC_U55C.bit
Debug probes file:
Enable end of startup check
Program
Cancel

Figure 11 Program Alveo by Vivado Tool

4) Warm reboot the system and confirm that the hardware is implemented on the card using the "lspci" command. The console must display "Network controller: Xilinx Corporation Device 903f" as shown in Figure 12.


TKAS-D2101 Console
The AAT demo is "Network controller : Xilinx Corporation Device 903f"

```
tkas-user@dg-turnkey1:~$ lspci
00:00.0 Host bridge: Intel Corporation Device 4c43 (rev 01)
00:01.0 PCI bridge: Intel Corporation Device 4c01 (rev 01)
00:02.0 VGA compatible controller: Intel Corporation RocketLake-S GT1 [UHD Graphics 750] (rev 04)
00:06.0 PCI bridge: Intel Corporation Device 4c09 (rev 01)
00:14.0 USB controller: Intel Corporation Tiger Lake-H USB 3.2 Gen 2x1 xHCI Host Controller (rev 11)
00:14.2 RAM memory: Intel Corporation Tiger Lake-H Shared SRAM (rev 11)
00:15.0 Serial bus controller: Intel Corporation Tiger Lake-H Serial IO I2C Controller #0 (rev 11)
00:15.1 Serial bus controller: Intel Corporation Device 43e9 (rev 11)
00:16.0 Communication controller: Intel Corporation Tiger Lake-H Management Engine Interface (rev 11)
00:17.0 SATA controller: Intel Corporation Device 43d2 (rev 11)
00:1b.0 PCI bridge: Intel Corporation Device 43c0 (rev 11)
00:1b.4 PCI bridge: Intel Corporation Device 43c4 (rev 11)
00:1c.0 PCI bridge: Intel Corporation Device 43b8 (rev 11)
00:1c.4 PCI bridge: Intel Corporation Tiger Lake-H PCI Express Root Port #5 (rev 11)
00:1d.0 PCI bridge: Intel Corporation Tiger Lake-H PCI Express Root Port #9 (rev 11)
00:1f.0 ISA bridge: Intel Corporation Device 4385 (rev 11)
00:1f.3 Audio device: Intel Corporation Tiger Lake-H HD Audio Controller (rev 11)
00:1f.4 SMBus: Int
00:1f.5 Serial bus controller: Intel Corporation Tiger Lake-H SPI Controller (rev 11)
00:1f.6 Ethernet controller: Intel Corporation Ethernet Connection (14) I219-V (rev 11)
01:00.0 Network controller: Xilinx Corporation Device 903f
02:00.0 Non-Volatile memory controller: Sandisk Corp WD Blue SN550 NVMe SSD (rev 01)
04:00.0 Non-Volatile memory controller: Sandisk Corp WD Black 2018/SN750 / PC SN720 NVMe SSD
06:00.0 Ethernet controller: Mellanox Technologies MT2894 Family [ConnectX-6 Lx]
06:00.1 Ethernet controller: Mellanox Technologies MT2894 Family [ConnectX-6 Lx]
tkas-user@dg-turnkey1:~$
```

Figure 12 Output of the "lspci" Command After Programming the Alveo Card

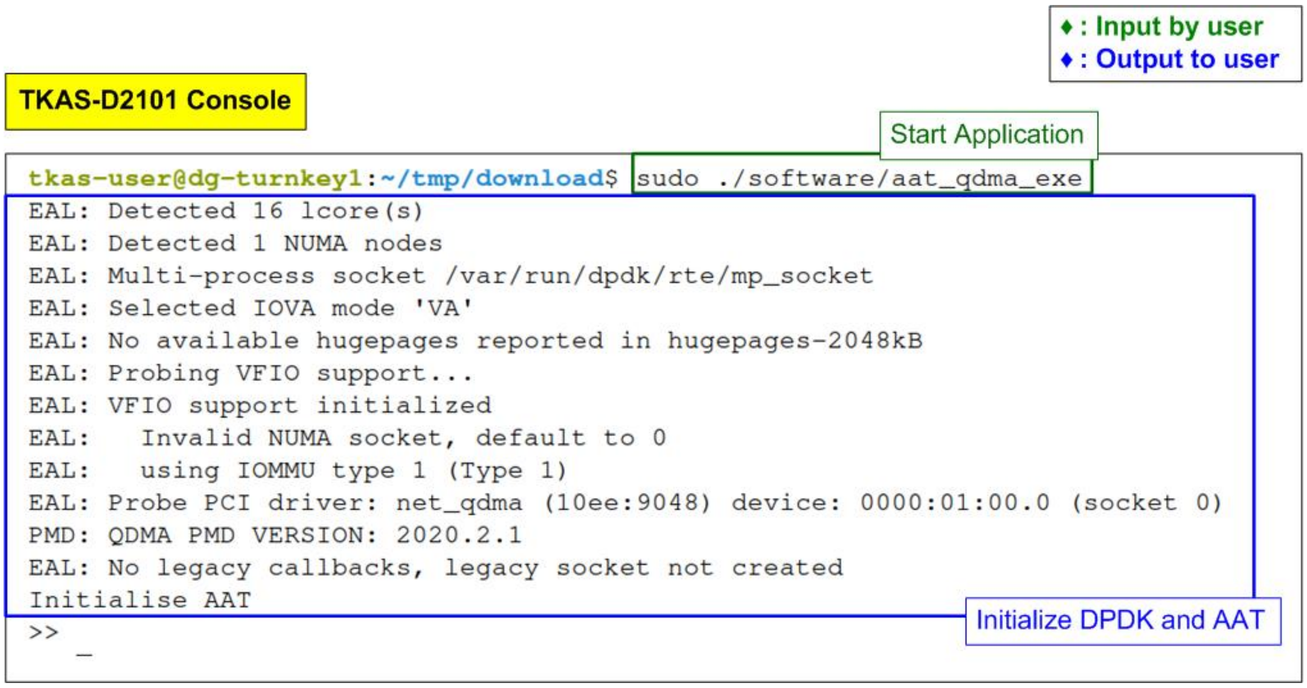
5) Bind the previously installed DPDK driver from step (1), with the Alveo card on the host system.

   i) Navigate to the "usertools" folder within the DPDK directory:
   >> cd <DPDK directory>/dpdk-20.11/usertools
   ii) User the following command to bind the vfio-pci driver to the Alveo card:
   >> sudo ./dpdk-devbind.py -b vfio-pci 01:00.0

6) Boost the AAT QDMA demo on the Alveo card by executing the "aat_qdma_exe" application.

   i) Navigate to the "download" folder:
   >> cd <download directory>
   ii) Execute the application:
   >> sudo ./software/aat_qdma_exe

   After execution, the DPDK and AAT applications will initialize, as shown in Figure 13.



```
tkas-user@dg-turnkey1:~/tmp/download$ sudo ./software/aat_qdma_exe
EAL: Detected 16 lcore(s)
EAL: Detected 1 NUMA nodes
EAL: Multi-process socket /var/run/dpdk/rte/mp_socket
EAL: Selected IOVA mode 'VA'
EAL: No available hugepages reported in hugepages-2048kB
EAL: Probing VFIO support...
EAL: VFIO support initialized
EAL:   Invalid NUMA socket, default to 0
EAL:   using IOMMU type 1 (Type 1)
EAL: Probe PCI driver: net_qdma (10ee:9048) device: 0000:01:00.0 (socket 0)
PMD: QDMA PMD VERSION: 2020.2.1
EAL: No legacy callbacks, legacy socket not created
Initialise AAT
>>
  _
```

Figure 13 Status Displayed After Executing the Demo Application on the Host System

4 Run AAT QDMA Demo

To execute the demo, the user must follow three key steps: Initialization, market data transmission, and test result display. The initialization process establishes the connection and configure the necessary parameters. The market data transmission process involves sending market data from the target system, while the results are received from the Alveo card and displayed on the TKAS-D2101 console. Detailed instructions for each process are provided below.

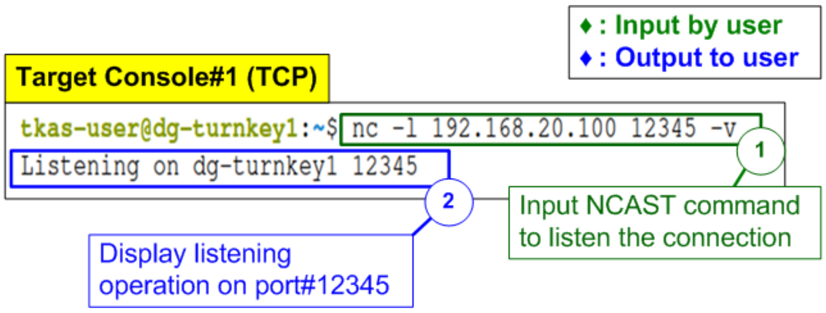
4.1 Initialization

To run the AAT QDMA demo, both the target system and the Alveo card need to be properly configured. The target must listen on a specific port to receive order packets from the Alveo card, once it has completed processing the market data. Similarly, the Alveo card must be configured to process the market data and send the order packet. This is done using "demo_setup.cfg" or "demo_setup_with_datamover.cfg" script file. Follow the steps below for system initialization.

1) On the target system's console, enter the following command to listen on port 12345.

   >> nc -l 192.168.20.100 12345 -v

   This will configure the target to listen for incoming packets on the specified IP and port.



```
tkas-user@dg-turnkey1:~$ nc -l 192.168.20.100 12345 -v
Listening on dg-turnkey1 12345
```

♦ : Input by user
♦ : Output to user
Target Console#1 (TCP)
1
2
Input NCAST command to listen the connection
Display listening operation on port#12345

Figure 14 Listen TCP Port on Target PC

2) After entering the command, you should see a confirmation message on the console "Listening on <Target PC name> 12345" indicating that the port is listening, as shown in Figure 14.

3) On the TKAS-D2101 console, run the script file to configure the parameters for processing market data. There are two options for the configuration depending on the implementation of Pricing Engine.

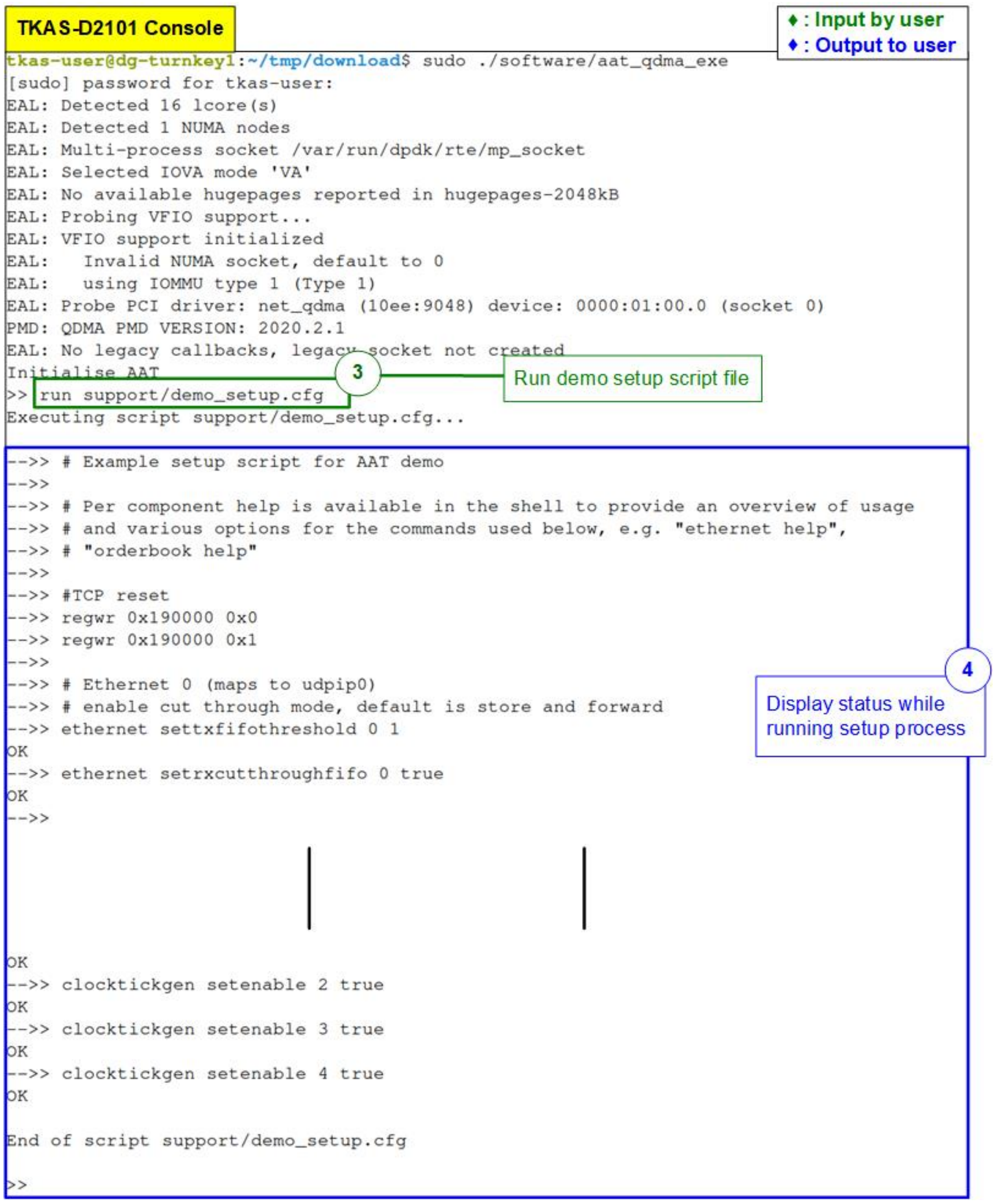
   - Configure to implement Pricing Engine on the Alveo card by using "demo_setup.cfg" script file. This method offers more latency reduction for market data processing.

     >> run support/demo_setup.cfg

   - Configure to implement Pricing Engine on the host software by using "demo_setup_with_datamover.cfg" script file. This method suits for more complicated algorithm for market data processing.

   >> run support/ demo_setup_with_datamover.cfg

   Note: The script file can only be executed once after the demo has already been boosted up (execute application in step (5) of section 3-Host System Setup). If user requires to rerun the script file, please use "exit" command to cease the demo application and repeat from step (5) of section 3-Host System Setup.

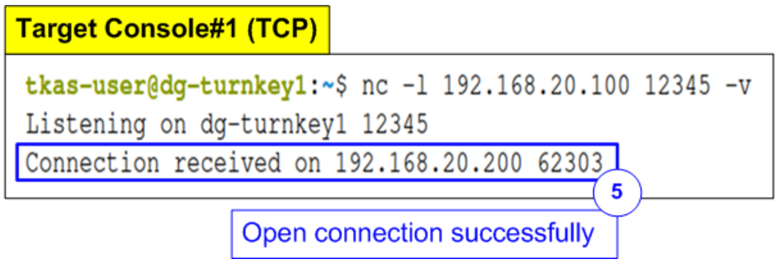

TKAS-D2101 Console
♦ : Input by user
♦ : Output to user
3
Run demo setup script file
4
Display status while running setup process

```
tkas-user@dg-turnkey1:~/tmp/download$ sudo ./software/aat_qdma_exe
[sudo] password for tkas-user:
EAL: Detected 16 lcore(s)
EAL: Detected 1 NUMA nodes
EAL: Multi-process socket /var/run/dpdk/rte/mp_socket
EAL: Selected IOVA mode 'VA'
EAL: No available hugepages reported in hugepages-2048kB
EAL: Probing VFIO support...
EAL: VFIO support initialized
EAL:   Invalid NUMA socket, default to 0
EAL:   using IOMMU type 1 (Type 1)
EAL: Probe PCI driver: net_qdma (10ee:9048) device: 0000:01:00.0 (socket 0)
PMD: QDMA PMD VERSION: 2020.2.1
EAL: No legacy callbacks, legacy socket not created
Initialise AAT
>> run support/demo_setup.cfg
Executing script support/demo_setup.cfg...

-->> # Example setup script for AAT demo
-->>
-->> # Per component help is available in the shell to provide an overview of usage
-->> # and various options for the commands used below, e.g. "ethernet help",
-->> # "orderbook help"
-->>
-->> #TCP reset
-->> regwr 0x190000 0x0
-->> regwr 0x190000 0x1
-->>
-->> # Ethernet 0 (maps to udpip0)
-->> # enable cut through mode, default is store and forward
-->> ethernet settxfifothreshold 0 1
OK
-->> ethernet setrxcutthroughfifo 0 true
OK
-->>

OK
-->> clocktickgen setenable 2 true
OK
-->> clocktickgen setenable 3 true
OK
-->> clocktickgen setenable 4 true
OK

End of script support/demo_setup.cfg

>>
```

Figure 15 Run Demo Configuration Script (using demo_setup.cfg)

4) TKAS-D2101 will display messages during the setup process, as shown in Figure 15. These messages will indicate the progress and status of the configuration process.

5) If the parameter configuration is successful, the target console will display a message indicating that the port has been opened successfully, such as “Connection received on 192.168.20.200 62303”, as shown in Figure 16.



```
tkas-user@dg-turnkey1:~$ nc -l 192.168.20.100 12345 -v
Listening on dg-turnkey1 12345
Connection received on 192.168.20.200 62303
```

Figure 16 Port Opened Success

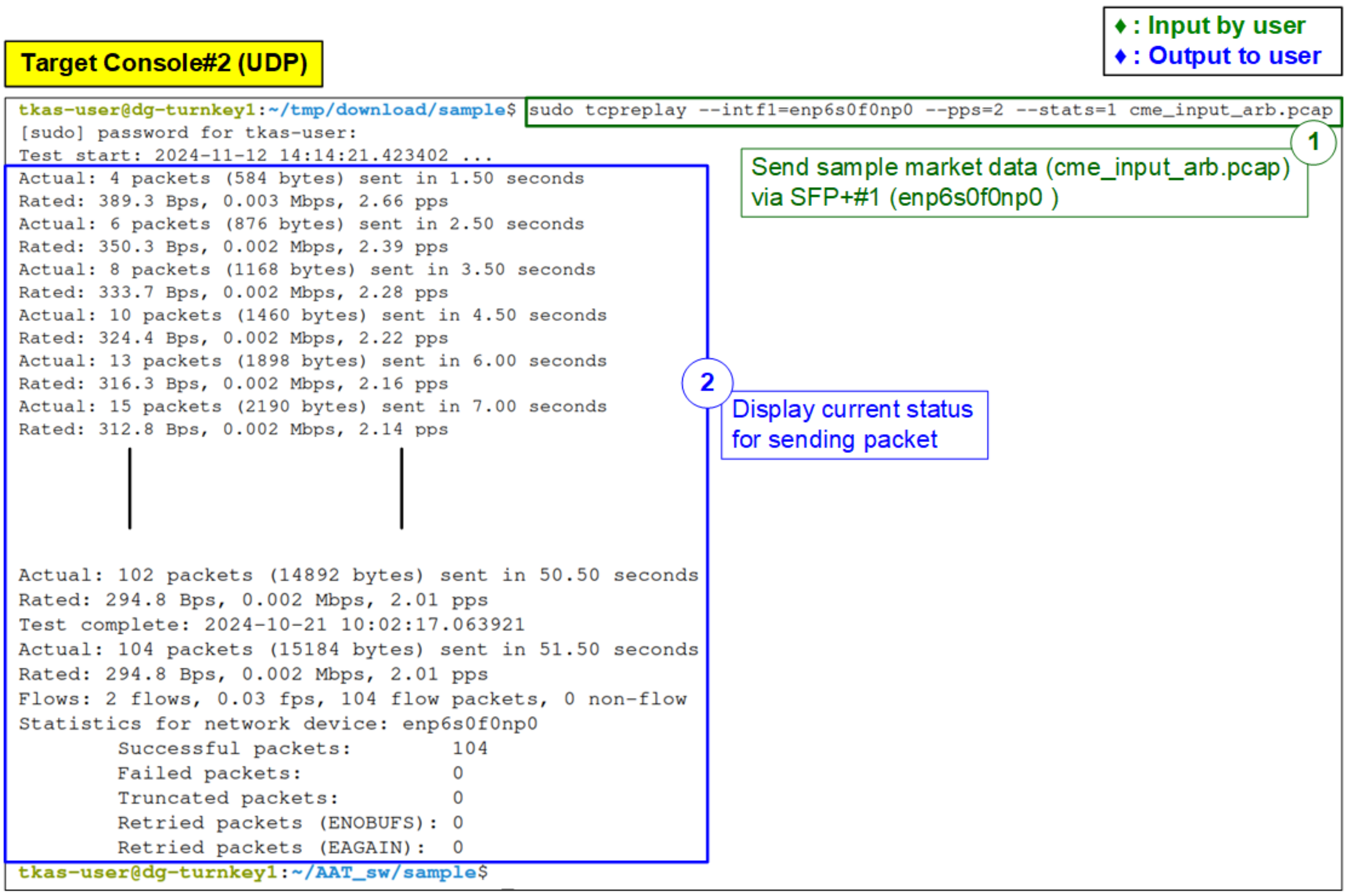
4.2 Market Data Transmission

To transmit sample market data, follow these steps using the “tcpreplay” on the target PC. You will need to open two terminal windows on the target PC: Target Console#1 and Target Console#2. Target Console#1 will display the details of the received order packet via TCP protocol (through SFP+#2: enp6s0f1np1), while Target Console#2 will be used to send the sample market data via UDP protocol (through SFP+#1: enp6s0f0np0). Follow the steps below.

1) Send the sample market data. Use the “tcpreplay” command to send the provided sample market data file from AMD AAT demo (cme_input_arb.pcap). Execute the following command on Target Console#2 with the four parameters.

   >> sudo tcpreplay –intf1=<eth I/F> --pps=<pac/sec> --stats=<stat period> <replay file>

i)	<eth I/F>	: Ethernet interface used to send market data (SFP+#1: enp6s0f0np0).
ii)	<pac/sec>	: Transfer speed, defined as the number of packets per second.
iii)	<stat period>	: Time interval (in seconds) to display the transmission status on the console.
iv)	<replay file>	: File name of the market data to be transmitted (e.g., cme_input_arb.pcap).

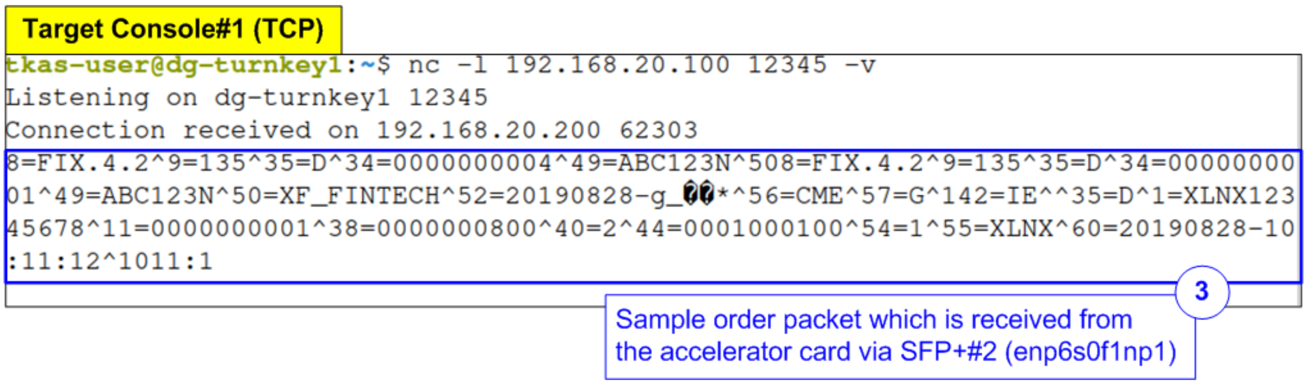

♦ : Input by user
♦ : Output to user
Target Console#2 (UDP)
1
Send sample market data (cme_input_arb.pcap) via SFP+#1 (enp6s0f0np0)
2
Display current status for sending packet

```
tkas-user@dg-turnkey1:~/tmp/download/sample$ sudo tcpreplay --intf1=enp6s0f0np0 --pps=2 --stats=1 cme_input_arb.pcap
[sudo] password for tkas-user:
Test start: 2024-11-12 14:14:21.423402 ...
Actual: 4 packets (584 bytes) sent in 1.50 seconds
Rated: 389.3 Bps, 0.003 Mbps, 2.66 pps
Actual: 6 packets (876 bytes) sent in 2.50 seconds
Rated: 350.3 Bps, 0.002 Mbps, 2.39 pps
Actual: 8 packets (1168 bytes) sent in 3.50 seconds
Rated: 333.7 Bps, 0.002 Mbps, 2.28 pps
Actual: 10 packets (1460 bytes) sent in 4.50 seconds
Rated: 324.4 Bps, 0.002 Mbps, 2.22 pps
Actual: 13 packets (1898 bytes) sent in 6.00 seconds
Rated: 316.3 Bps, 0.002 Mbps, 2.16 pps
Actual: 15 packets (2190 bytes) sent in 7.00 seconds
Rated: 312.8 Bps, 0.002 Mbps, 2.14 pps


Actual: 102 packets (14892 bytes) sent in 50.50 seconds
Rated: 294.8 Bps, 0.002 Mbps, 2.01 pps
Test complete: 2024-10-21 10:02:17.063921
Actual: 104 packets (15184 bytes) sent in 51.50 seconds
Rated: 294.8 Bps, 0.002 Mbps, 2.01 pps
Flows: 2 flows, 0.03 fps, 104 flow packets, 0 non-flow
Statistics for network device: enp6s0f0np0
        Successful packets:        104
        Failed packets:            0
        Truncated packets:         0
        Retried packets (ENOBUFS): 0
        Retried packets (EAGAIN):  0
tkas-user@dg-turnkey1:~/AAT_sw/sample$
```

Figure 17 Sample Market Data Transmission by “tcpreplay”

2) As the data is transmitted, the console will display the status every second, showing the total number of packets transmitted.

3) On Target Console#1, which is already connected to the listening port, the console will display the received data. The data represents the sample order packet returned by the Alveo card and serves as the result of the AAT QDMA demo.


Target Console#1 (TCP)
3
Sample order packet which is received from the accelerator card via SFP+#2 (enp6s0f1np1)

```
tkas-user@dg-turnkey1:~$ nc -l 192.168.20.100 12345 -v
Listening on dg-turnkey1 12345
Connection received on 192.168.20.200 62303
8=FIX.4.2^9=135^35=D^34=0000000004^49=ABC123N^508=FIX.4.2^9=135^35=D^34=00000000
01^49=ABC123N^50=XF_FINTECH^52=20190828-g_��*^56=CME^57=G^142=IE^^35=D^1=XLNX123
45678^11=0000000001^38=0000000800^40=2^44=0001000100^54=1^55=XLNX^60=20190828-10
:11:12^1011:1
```

Figure 18 The Sample Data of Order Packet on SFP+#2 Channel

4.3 AAT QDMA Demo

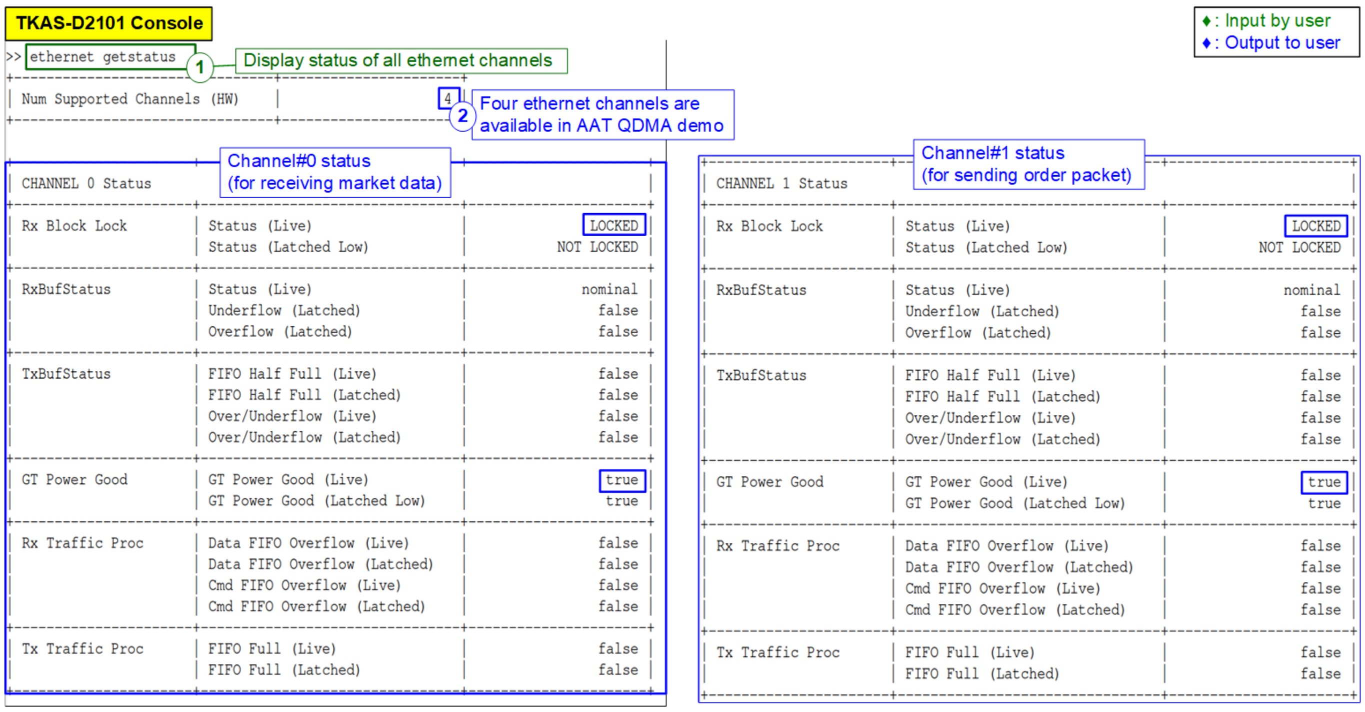
This section provides example results from market data processing on the Alveo card. The AAT QDMA demo design includes the subsystem and multiple submodules responsible for processing the sample market data. This section focuses on six key components: Ethernet subsystem, Feed Handler submodule, Order Book submodule, Data Mover submodule, Pricing Engine submodule, and Order Entry submodule. The following sections describe the sample results obtained from these components within the demo design.

4.3.1 Ethernet Subsystem

To check the status of the Ethernet subsystem in the AAT QDMA demo system, follow these steps.

1) Enter the following command in the terminal to display the status of Ethernet subsystem.

   >> ethernet getstatus


TKAS-D2101 Console
♦: Input by user
♦: Output to user
1
Display status of all ethernet channels
2
Four ethernet channels are available in AAT QDMA demo
Channel#0 status (for receiving market data)
Channel#1 status (for sending order packet)

```
>>ethernet getstatus
+-------------------------------+--------------------------+
| Num Supported Channels (HW)   |                        4 |
+-------------------------------+--------------------------+

+-----------------------+----------------------------------+---------------------+
| CHANNEL 0 Status                                                               |
+-----------------------+----------------------------------+---------------------+
| Rx Block Lock         | Status (Live)                    |              LOCKED |
|                       | Status (Latched Low)             |          NOT LOCKED |
+-----------------------+----------------------------------+---------------------+
| RxBufStatus           | Status (Live)                    |             nominal |
|                       | Underflow (Latched)              |               false |
|                       | Overflow (Latched)               |               false |
+-----------------------+----------------------------------+---------------------+
| TxBufStatus           | FIFO Half Full (Live)            |               false |
|                       | FIFO Half Full (Latched)         |               false |
|                       | Over/Underflow (Live)            |               false |
|                       | Over/Underflow (Latched)         |               false |
+-----------------------+----------------------------------+---------------------+
| GT Power Good         | GT Power Good (Live)             |                true |
|                       | GT Power Good (Latched Low)      |                true |
+-----------------------+----------------------------------+---------------------+
| Rx Traffic Proc       | Data FIFO Overflow (Live)        |               false |
|                       | Data FIFO Overflow (Latched)     |               false |
|                       | Cmd FIFO Overflow (Live)         |               false |
|                       | Cmd FIFO Overflow (Latched)      |               false |
+-----------------------+----------------------------------+---------------------+
| Tx Traffic Proc       | FIFO Full (Live)                 |               false |
|                       | FIFO Full (Latched)              |               false |
+-----------------------+----------------------------------+---------------------+

+-----------------------+----------------------------------+---------------------+
| CHANNEL 1 Status                                                               |
+-----------------------+----------------------------------+---------------------+
| Rx Block Lock         | Status (Live)                    |              LOCKED |
|                       | Status (Latched Low)             |          NOT LOCKED |
+-----------------------+----------------------------------+---------------------+
| RxBufStatus           | Status (Live)                    |             nominal |
|                       | Underflow (Latched)              |               false |
|                       | Overflow (Latched)               |               false |
+-----------------------+----------------------------------+---------------------+
| TxBufStatus           | FIFO Half Full (Live)            |               false |
|                       | FIFO Half Full (Latched)         |               false |
|                       | Over/Underflow (Live)            |               false |
|                       | Over/Underflow (Latched)         |               false |
+-----------------------+----------------------------------+---------------------+
| GT Power Good         | GT Power Good (Live)             |                true |
|                       | GT Power Good (Latched Low)      |                true |
+-----------------------+----------------------------------+---------------------+
| Rx Traffic Proc       | Data FIFO Overflow (Live)        |               false |
|                       | Data FIFO Overflow (Latched)     |               false |
|                       | Cmd FIFO Overflow (Live)         |               false |
|                       | Cmd FIFO Overflow (Latched)      |               false |
+-----------------------+----------------------------------+---------------------+
| Tx Traffic Proc       | FIFO Full (Live)                 |               false |
|                       | FIFO Full (Latched)              |               false |
+-----------------------+----------------------------------+---------------------+
```

Figure 19: Ethernet Submodule Status

2) The AAT QDMA demo system uses four Ethernet channels: channel0 to channel3. In this example, channel#0 is used for receiving the sample market data, and channel#1 is used for returning the sample order packet. Check the status of channel#0 and channel#1 to ensure they are operating correctly. The following parameters should be verified.

• Rx Block Lock Status (Live)	: LOCKED
• GT Power Good (Live)	: true

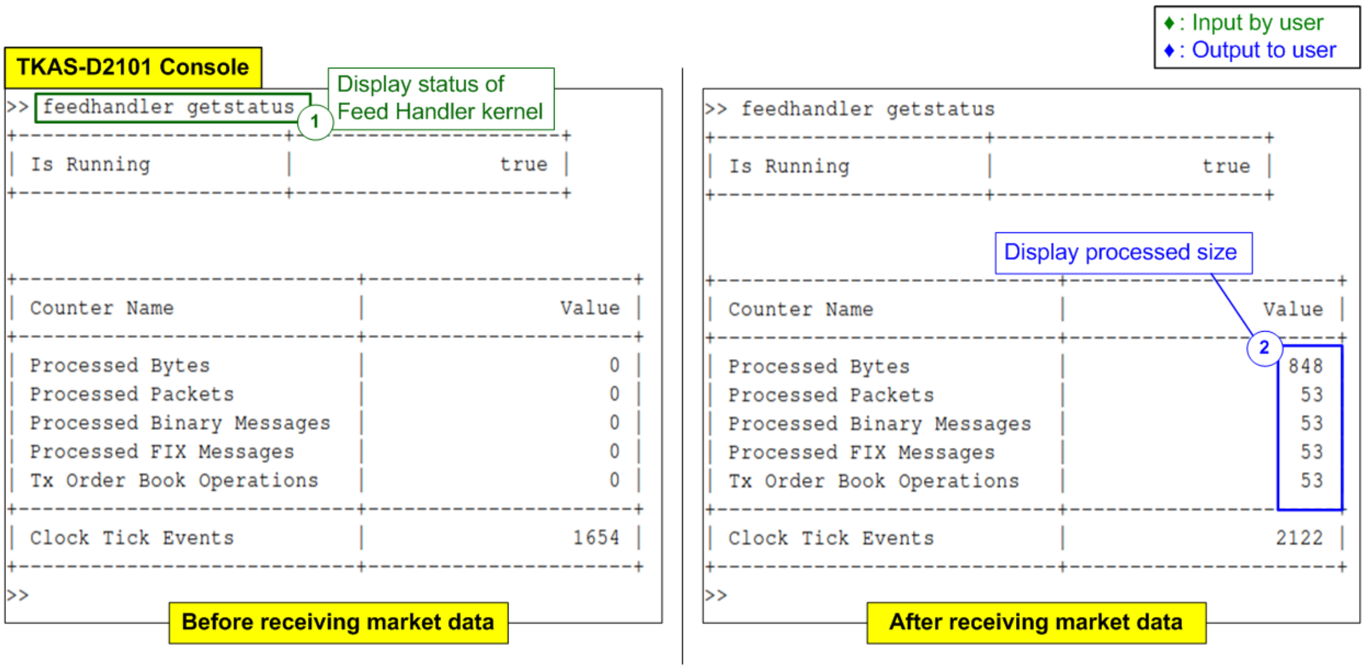
4.3.2 Feed Handler Submodule

To check the status of the Feed Handler submodule in the AAT QDMA demo system, follow these steps.

1) Enter the following command in the terminal to display the status of the Feed Handler submodule.

   >> feedhandler getstatus

2) The console will display the count of processed data in various units, such as bytes, packets, and messages. Before transmitting the sample market data, the processed data count will be zero. After the market data transmission, this count will increase, indicating that the Feed Handler submodule has begun processing the market data.


♦ : Input by user
♦ : Output to user
TKAS-D2101 Console
Display status of Feed Handler kernel
1
Display processed size
2
Before receiving market data
After receiving market data

```
>> feedhandler getstatus
+----------------------+----------------------+
| Is Running           |                 true |
+----------------------+----------------------+


+----------------------------+----------------------+
| Counter Name               |                Value |
+----------------------------+----------------------+
| Processed Bytes            |                    0 |
| Processed Packets          |                    0 |
| Processed Binary Messages  |                    0 |
| Processed FIX Messages     |                    0 |
| Tx Order Book Operations   |                    0 |
+----------------------------+----------------------+
| Clock Tick Events          |                 1654 |
+----------------------------+----------------------+
>>
```

```
>> feedhandler getstatus
+----------------------+----------------------+
| Is Running           |                 true |
+----------------------+----------------------+


+----------------------------+----------------------+
| Counter Name               |                Value |
+----------------------------+----------------------+
| Processed Bytes            |                  848 |
| Processed Packets          |                   53 |
| Processed Binary Messages  |                   53 |
| Processed FIX Messages     |                   53 |
| Tx Order Book Operations   |                   53 |
+----------------------------+----------------------+
| Clock Tick Events          |                 2122 |
+----------------------------+----------------------+
>>
```

Figure 20 Feed Handler Submodule Status

For example, in Figure 20, the left window shows that the processed data count is initially zero. After transmitting the sample market data, the count increases, confirming the submodule is processing the data.

In Figure 17, an example is shown where 104 packets of sample market data were sent by the target PC. Out of these packets, 53 packets were processed by the Feed Handler submodule, while the remaining packets were rejected.

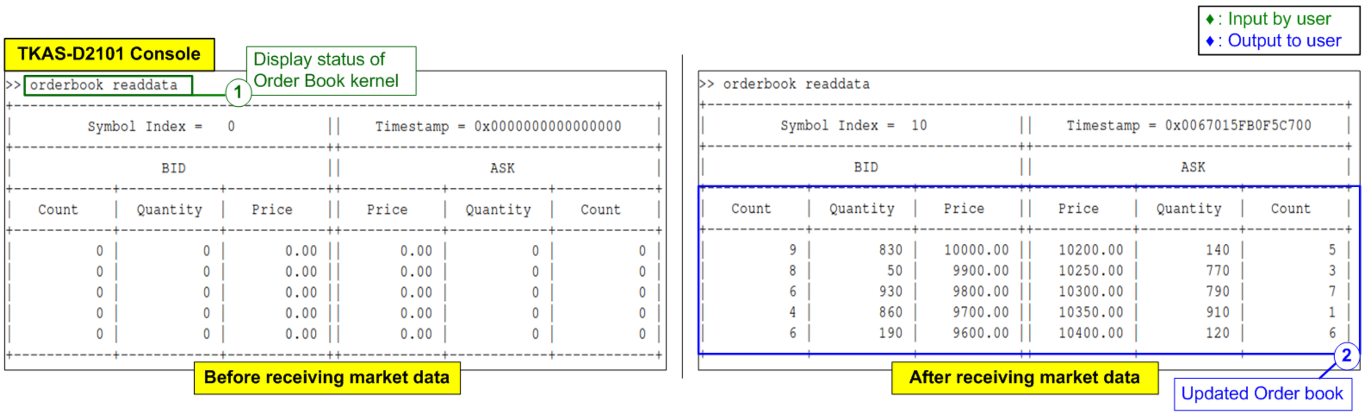
4.3.3 Order Book Submodule

To check the status of the Order Book submodule in the AAT QDMA demo system, follow these steps.

1) Enter the following command to read and display the current order book output from the Order Book submodule.

   >> orderbook readdata

2) The console will show the current values in the order book. Initially, before transmitting any sample market data, the order book will be in a clean state. However, after the transmission of all sample market data, the Order Book submodule will update the bid/ask quantities and prices in the order book to reflect the changes in market conditions.


♦ : Input by user
♦ : Output to user
TKAS-D2101 Console
Display status of Order Book kernel
1
Before receiving market data
After receiving market data
2
Updated Order book

```
>> orderbook readdata
+-------------------------------------------++-------------------------------------------+
|         Symbol Index =   0                ||     Timestamp = 0x0000000000000000        |
+-------------------------------------------++-------------------------------------------+
|                   BID                     ||                   ASK                     |
+-------------+-------------+---------------++-------------+-------------+---------------+
|    Count    |  Quantity   |    Price      ||    Price    |  Quantity   |    Count      |
+-------------+-------------+---------------++-------------+-------------+---------------+
|           0 |           0 |        0.00   ||        0.00 |           0 |            0  |
|           0 |           0 |        0.00   ||        0.00 |           0 |            0  |
|           0 |           0 |        0.00   ||        0.00 |           0 |            0  |
|           0 |           0 |        0.00   ||        0.00 |           0 |            0  |
|           0 |           0 |        0.00   ||        0.00 |           0 |            0  |
+-------------+-------------+---------------++-------------+-------------+---------------+
```

```
>> orderbook readdata
+-------------------------------------------++-------------------------------------------+
|         Symbol Index =   10               ||     Timestamp = 0x0067015FB0F5C700        |
+-------------------------------------------++-------------------------------------------+
|                   BID                     ||                   ASK                     |
+-------------+-------------+---------------++-------------+-------------+---------------+
|    Count    |  Quantity   |    Price      ||    Price    |  Quantity   |    Count      |
+-------------+-------------+---------------++-------------+-------------+---------------+
|           9 |         830 |    10000.00   ||    10200.00 |         140 |            5  |
|           8 |          50 |     9900.00   ||    10250.00 |         770 |            3  |
|           6 |         930 |     9800.00   ||    10300.00 |         790 |            7  |
|           4 |         860 |     9700.00   ||    10350.00 |         910 |            1  |
|           6 |         190 |     9600.00   ||    10400.00 |         120 |            6  |
+-------------+-------------+---------------++-------------+-------------+---------------+
```

Figure 21 Updated Order Book upon the Processing Completion

For instance, in Figure 21, the left window displays the clean status of the order book before the market data is transmitted. The right window shows the updated order book after all the sample market data has been processed. The bid/ask quantities and prices are adjusted based on the market data, as updated by the Order Book submodule.

4.3.4 Data Mover Submodule

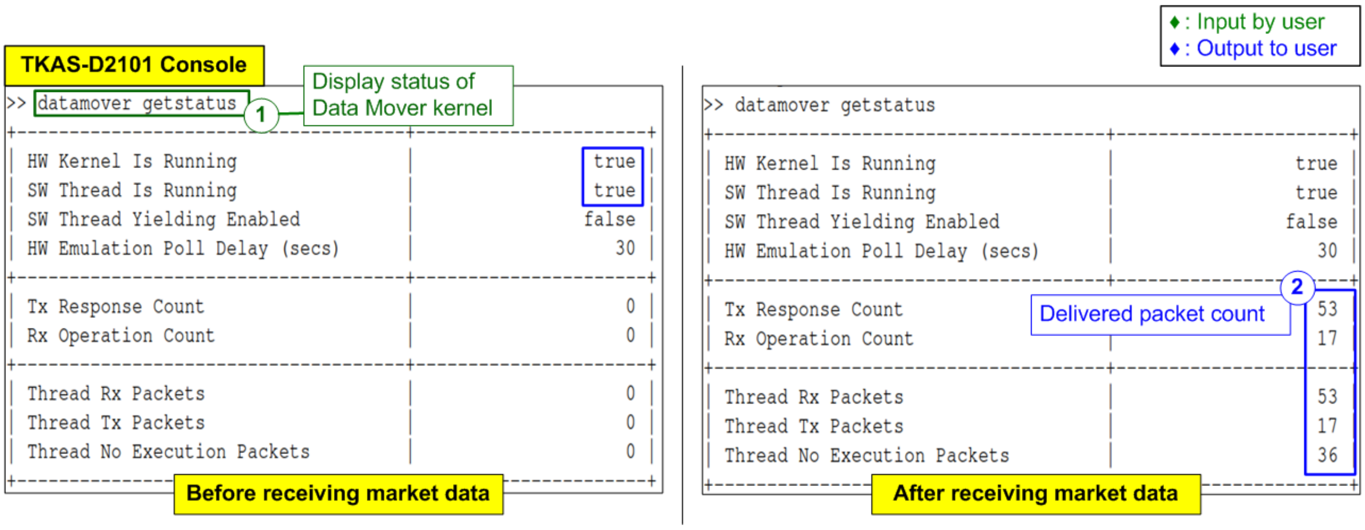
To check the status of the Data Mover submodule in the AAT QDMA demo system, follow these steps.

1) Enter the following command in the terminal to display the status of Data Mover submodule.

   >> datamover getstatus

2) The console will display the current status of Data Mover submodule. This Data Mover is active only when user selects to implement Pricing Engine by host software (utilized "demo_setup_with_datamover.cfg" file in step (3) of section 4.1-Initialization). Before transmitting the sample market data, the processed data count will be zero. After the market data transmission, this count will increase, indicating that the Data Mover submodule has begun delivering the market data. Before transmitting the sample market data, make sure that the software thread and hardware submodule (hardware kernel) is running:

   - HW Kernel Is Running : true
   - SW Thread Is Running : true


♦ : Input by user
♦ : Output to user
TKAS-D2101 Console
Display status of Data Mover kernel
1
2
Delivered packet count
Before receiving market data
After receiving market data

```
>> datamover getstatus
+-----------------------------------+---------------------+
| HW Kernel Is Running              |                true |
| SW Thread Is Running              |                true |
| SW Thread Yielding Enabled        |               false |
| HW Emulation Poll Delay (secs)    |                  30 |
+-----------------------------------+---------------------+
| Tx Response Count                 |                   0 |
| Rx Operation Count                |                   0 |
+-----------------------------------+---------------------+
| Thread Rx Packets                 |                   0 |
| Thread Tx Packets                 |                   0 |
| Thread No Execution Packets       |                   0 |
+-----------------------------------+---------------------+
```

```
>> datamover getstatus
+--------------------------------------+----------------------+
| HW Kernel Is Running                 |                 true |
| SW Thread Is Running                 |                 true |
| SW Thread Yielding Enabled           |                false |
| HW Emulation Poll Delay (secs)       |                   30 |
+--------------------------------------+----------------------+
| Tx Response Count                    |                   53 |
| Rx Operation Count                   |                   17 |
+--------------------------------------+----------------------+
| Thread Rx Packets                    |                   53 |
| Thread Tx Packets                    |                   17 |
| Thread No Execution Packets          |                   36 |
+--------------------------------------+----------------------+
```

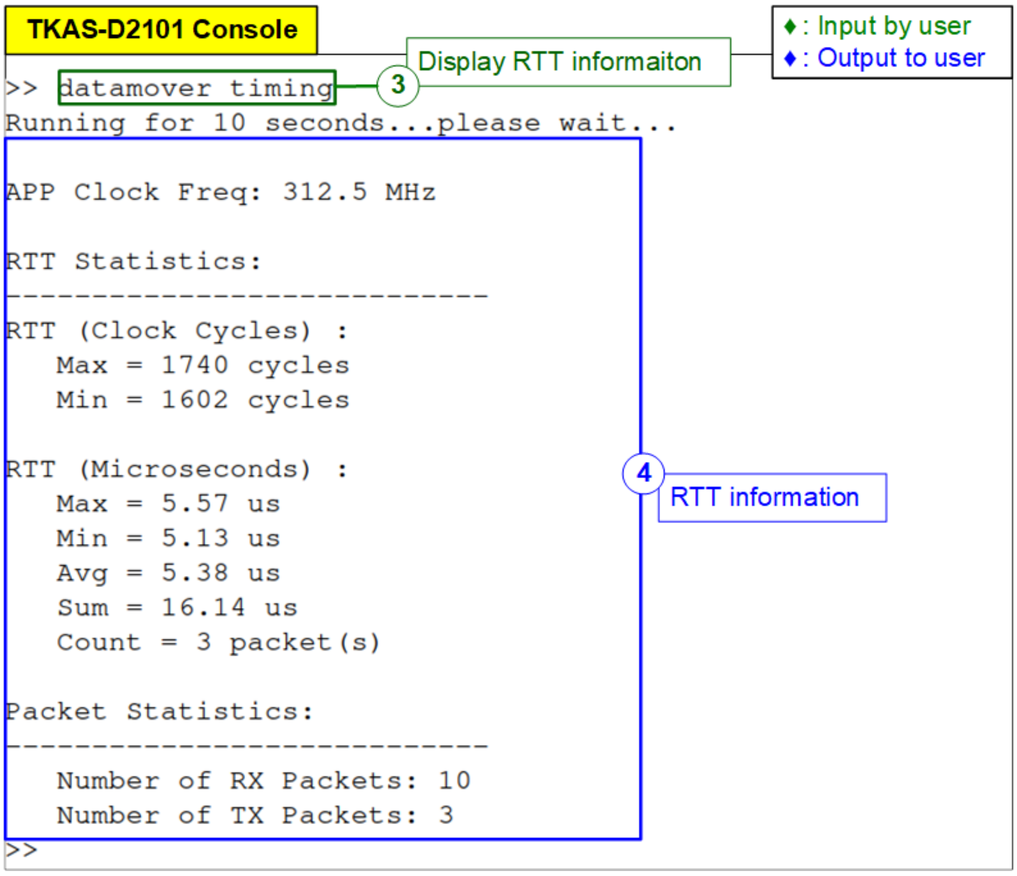
Figure 22 Data Mover Submodule Status

In Figure 22, the total number of packets moved to Pricing Engine on the host software is 53 packets. After the host Pricing engine completes the processing, the Data Mover receives 17 packets representing the orders to be delivered to the Order Entry submodule. The remaining 36 packets are not executed by the Pricing Engine.

3) To measure the Round-Trip Time (RTT) – measured once packet is out from Data Mover submodule, be processed in host Pricing Engine, and order packet returns back to Data Mover submodule, execute the following command while transmitting the sample market data.

   >> datamover timing

4) After entering the command, the measurement is operated for 10 seconds.



```
>> datamover timing
Running for 10 seconds...please wait...

APP Clock Freq: 312.5 MHz

RTT Statistics:
----------------------------
RTT (Clock Cycles) :
   Max = 1740 cycles
   Min = 1602 cycles

RTT (Microseconds) :
   Max = 5.57 us
   Min = 5.13 us
   Avg = 5.38 us
   Sum = 16.14 us
   Count = 3 packet(s)

Packet Statistics:
----------------------------
   Number of RX Packets: 10
   Number of TX Packets: 3
>>
```

Figure 23 Data Mover RTT Result

Figure 23 shows the RTT measurement result including the maximum, minimum, average values, and the total number of collected sample market data packets transmitted over a 10-second interval.

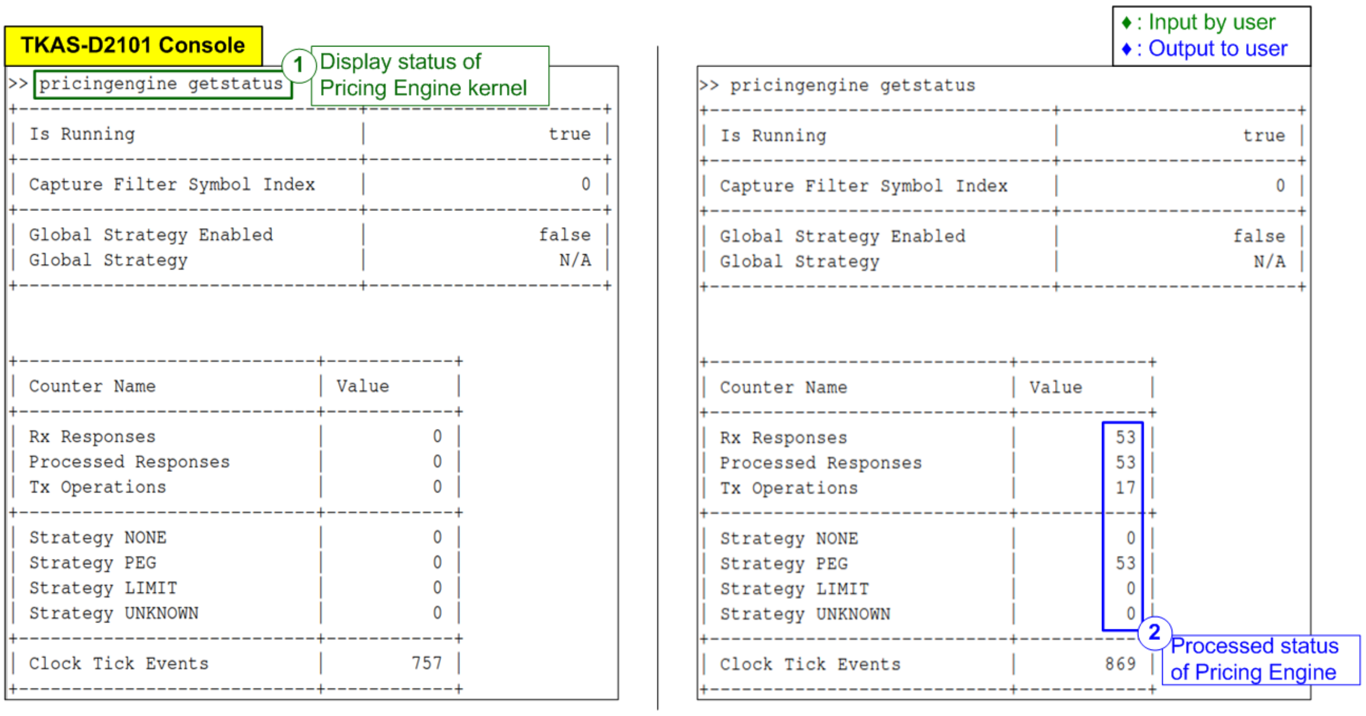
4.3.5 Pricing Engine Submodule

To check the status of the Pricing Engine submodule in the AAT QDMA demo system, follow these steps.

1) Enter the following command in the terminal to display the status of Pricing Engine submodule.

   >> pricingengine getstatus

2) Pricing Engine submodule, operating on the Alveo card, is activated only when user selects to implement Pricing Engine on the FPGA (selected “demo_setup.cfg” file in step (3) of section 4.1-Initialization). Similar to the other components, before transmitting any sample market data, the Pricing Engine will be in a clean state.


TKAS-D2101 Console
1
Display status of Pricing Engine kernel
♦ : Input by user
♦ : Output to user
2
Processed status of Pricing Engine

```
>> pricingengine getstatus
+-----------------------------------+----------------------+
| Is Running                        |                 true |
+-----------------------------------+----------------------+
| Capture Filter Symbol Index       |                    0 |
+-----------------------------------+----------------------+
| Global Strategy Enabled           |                false |
| Global Strategy                   |                  N/A |
+-----------------------------------+----------------------+


+------------------------------+------------+
| Counter Name                 | Value      |
+------------------------------+------------+
| Rx Responses                 |          0 |
| Processed Responses          |          0 |
| Tx Operations                |          0 |
+------------------------------+------------+
| Strategy NONE                |          0 |
| Strategy PEG                 |          0 |
| Strategy LIMIT               |          0 |
| Strategy UNKNOWN             |          0 |
+------------------------------+------------+
| Clock Tick Events            |        757 |
+------------------------------+------------+
```

```
>> pricingengine getstatus
+-----------------------------------+----------------------+
| Is Running                        |                 true |
+-----------------------------------+----------------------+
| Capture Filter Symbol Index       |                    0 |
+-----------------------------------+----------------------+
| Global Strategy Enabled           |                false |
| Global Strategy                   |                  N/A |
+-----------------------------------+----------------------+


+------------------------------+------------+
| Counter Name                 | Value      |
+------------------------------+------------+
| Rx Responses                 |         53 |
| Processed Responses          |         53 |
| Tx Operations                |         17 |
+------------------------------+------------+
| Strategy NONE                |          0 |
| Strategy PEG                 |         53 |
| Strategy LIMIT               |          0 |
| Strategy UNKNOWN             |          0 |
+------------------------------+------------+
| Clock Tick Events            |        869 |
+------------------------------+------------+
```

Figure 24 Pricing Engine Submodule Status

As shown in Figure 24, after market data is transmitted and processed, Pricing Engine submodule receives 53 data sets from the updated order book. As the consequences of computation and algorithm of Pricing Engine, 17 data sets match the conditions and require trading actions, which is informed to Order Entry as the next step.

4.3.6 Order Entry Submodule

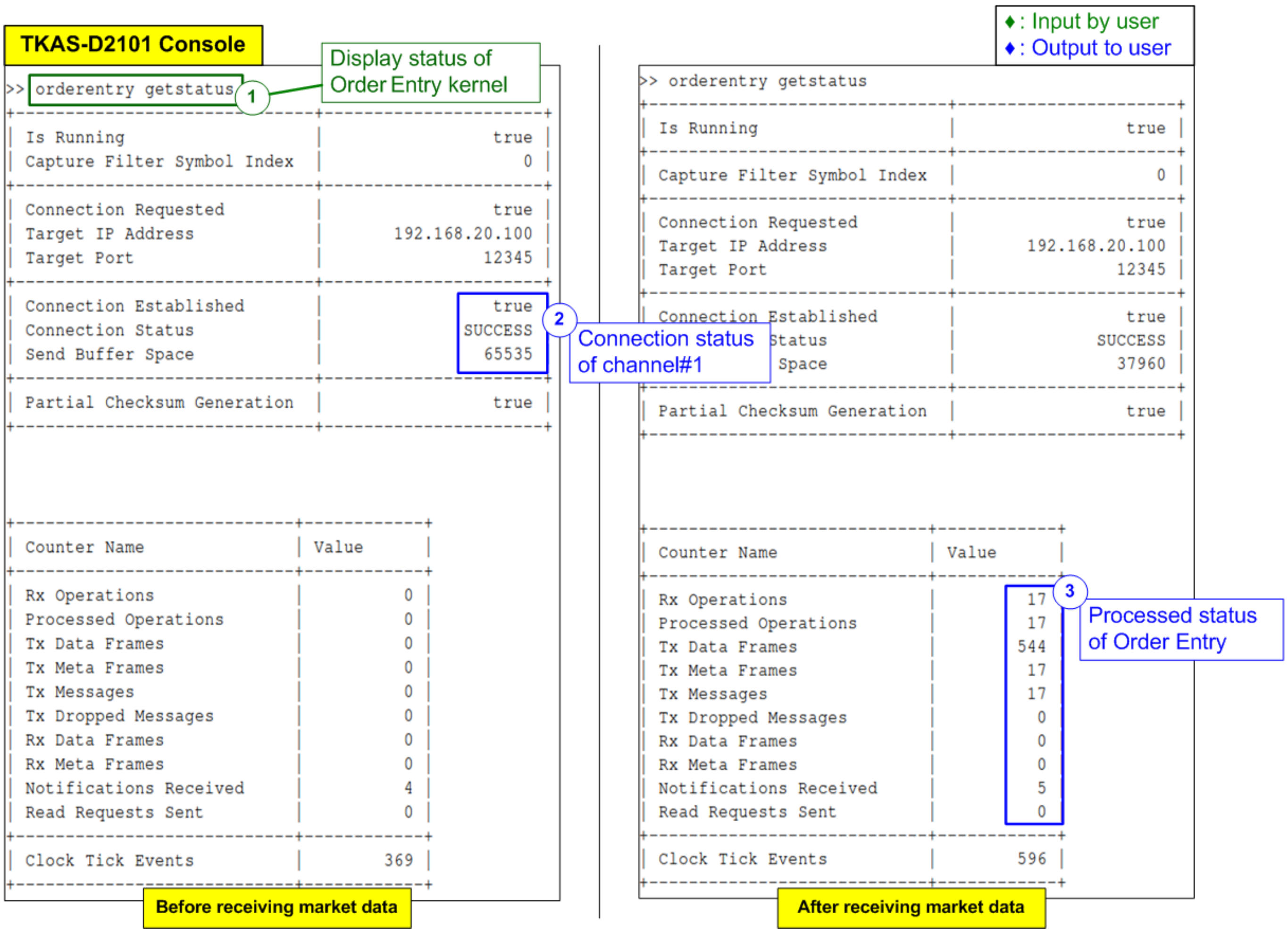
To check the current status of the Order Entry submodule in the AAT demo system, follow these steps.

1) Enter the following command to display the current status of the Order Entry submodule.

   >> orderentry getstatus

2) Before transmitting the sample market data, check the connection status of Ethernet channel#1 in the Order Entry submodule status. Two key indicators to confirm are as follows.

   - Connection Established : true
   - Connection Status : SUCCESS



TKAS-D2101 Console
Display status of Order Entry kernel
1
2
Connection status of channel#1
3
Processed status of Order Entry
♦: Input by user
♦: Output to user
Before receiving market data
After receiving market data

```
>> orderentry getstatus
+----------------------------------+------------------------+
| Is Running                       |                   true |
| Capture Filter Symbol Index      |                      0 |
+----------------------------------+------------------------+
| Connection Requested             |                   true |
| Target IP Address                |         192.168.20.100 |
| Target Port                      |                  12345 |
+----------------------------------+------------------------+
| Connection Established           |                   true |
| Connection Status                |                SUCCESS |
| Send Buffer Space                |                  65535 |
+----------------------------------+------------------------+
| Partial Checksum Generation      |                   true |
+----------------------------------+------------------------+

+------------------------------+--------------+
| Counter Name                 | Value        |
+------------------------------+--------------+
| Rx Operations                |            0 |
| Processed Operations         |            0 |
| Tx Data Frames               |            0 |
| Tx Meta Frames               |            0 |
| Tx Messages                  |            0 |
| Tx Dropped Messages          |            0 |
| Rx Data Frames               |            0 |
| Rx Meta Frames               |            0 |
| Notifications Received       |            4 |
| Read Requests Sent           |            0 |
+------------------------------+--------------+
| Clock Tick Events            |          369 |
+------------------------------+--------------+
```

```
>> orderentry getstatus
+----------------------------------+------------------------+
| Is Running                       |                   true |
+----------------------------------+------------------------+
| Capture Filter Symbol Index      |                      0 |
+----------------------------------+------------------------+
| Connection Requested             |                   true |
| Target IP Address                |         192.168.20.100 |
| Target Port                      |                  12345 |
+----------------------------------+------------------------+
| Connection Established           |                   true |
|            Status                |                SUCCESS |
|            Space                 |                  37960 |
+----------------------------------+------------------------+
| Partial Checksum Generation      |                   true |
+----------------------------------+------------------------+

+------------------------------+--------------+
| Counter Name                 | Value        |
+------------------------------+--------------+
| Rx Operations                |           17 |
| Processed Operations         |           17 |
| Tx Data Frames               |          544 |
| Tx Meta Frames               |           17 |
| Tx Messages                  |           17 |
| Tx Dropped Messages          |            0 |
| Rx Data Frames               |            0 |
| Rx Meta Frames               |            0 |
| Notifications Received       |            5 |
| Read Requests Sent           |            0 |
+------------------------------+--------------+
| Clock Tick Events            |          596 |
+------------------------------+--------------+
```

Figure 25 Order Entry Submodule Status

3) After transmitting the market data completely, the packet count in the Order Entry submodule will update from 0 to reflect the total number of messages or frames processed by the Order Entry submodule.

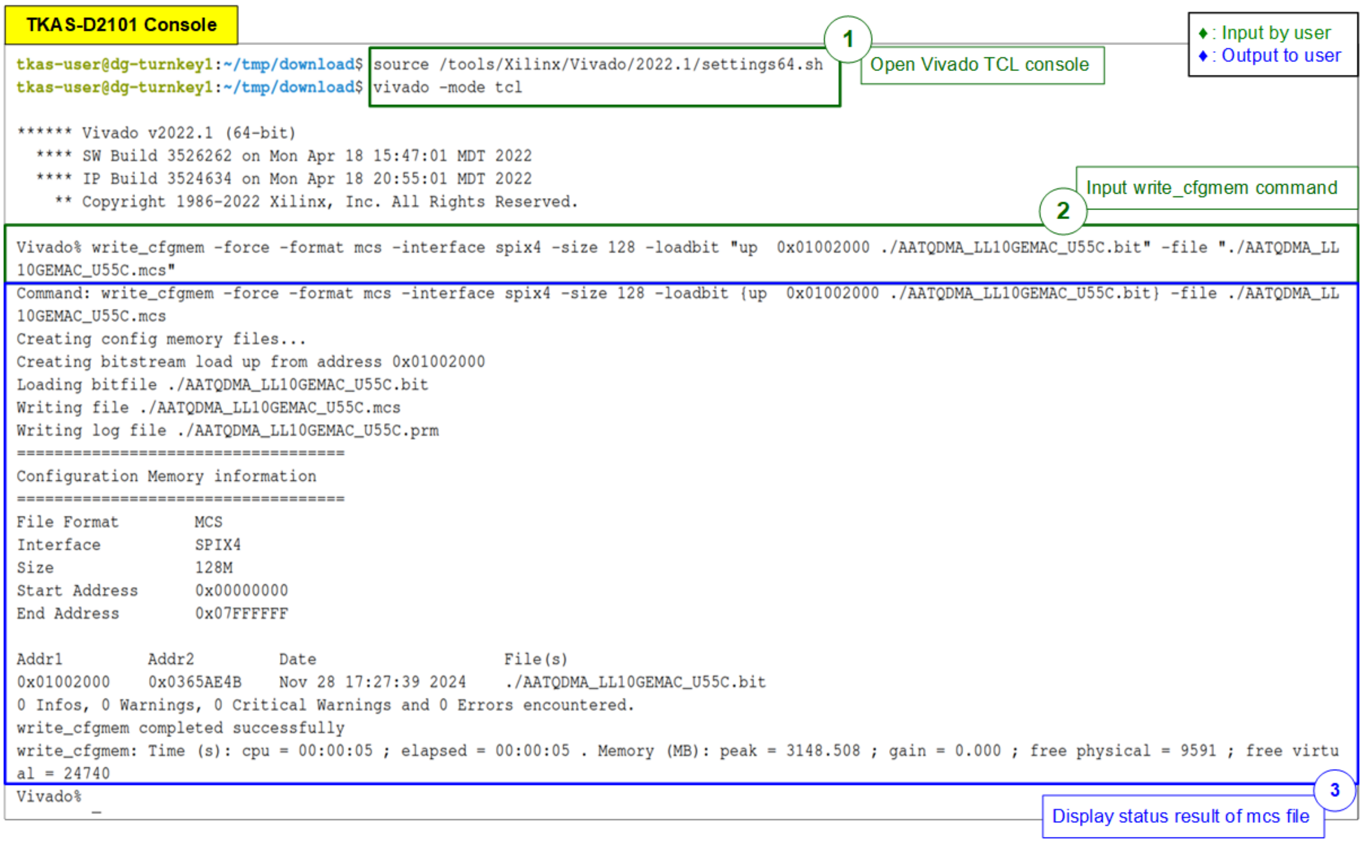
5 Update Hardware via PCIe

In certain environments, remote hardware updates are necessary. The AAT-QDMA includes a hardware module for supporting hardware updates via PCIe, eliminating the need for a programming cable. This section outlines two steps: mcs file creation and downloading.

For downloading the mcs file via PCIe, the "xbflash2" utility is required. Users can download and install "xbflash2" by following the link below:

https://www.amd.com/en/products/accelerators/alveo/u250/a-u250-a64g-pq-g.html#tabs-ca1f6f6dc7-item-2760c1c14c-tab

5.1 Create the MCS file



```
tkas-user@dg-turnkey1:~/tmp/download$ source /tools/Xilinx/Vivado/2022.1/settings64.sh
tkas-user@dg-turnkey1:~/tmp/download$ vivado -mode tcl

****** Vivado v2022.1 (64-bit)
  **** SW Build 3526262 on Mon Apr 18 15:47:01 MDT 2022
  **** IP Build 3524634 on Mon Apr 18 20:55:01 MDT 2022
    ** Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.

Vivado% write_cfgmem -force -format mcs -interface spix4 -size 128 -loadbit "up  0x01002000 ./AATQDMA_LL10GEMAC_U55C.bit" -file "./AATQDMA_LL
10GEMAC_U55C.mcs"
Command: write_cfgmem -force -format mcs -interface spix4 -size 128 -loadbit {up  0x01002000 ./AATQDMA_LL10GEMAC_U55C.bit} -file ./AATQDMA_LL
10GEMAC_U55C.mcs
Creating config memory files...
Creating bitstream load up from address 0x01002000
Loading bitfile ./AATQDMA_LL10GEMAC_U55C.bit
Writing file ./AATQDMA_LL10GEMAC_U55C.mcs
Writing log file ./AATQDMA_LL10GEMAC_U55C.prm
===================================
Configuration Memory information
===================================
File Format        MCS
Interface          SPIX4
Size               128M
Start Address      0x00000000
End Address        0x07FFFFFF

Addr1         Addr2         Date                    File(s)
0x01002000    0x0365AE4B    Nov 28 17:27:39 2024    ./AATQDMA_LL10GEMAC_U55C.bit
0 Infos, 0 Warnings, 0 Critical Warnings and 0 Errors encountered.
write_cfgmem completed successfully
write_cfgmem: Time (s): cpu = 00:00:05 ; elapsed = 00:00:05 . Memory (MB): peak = 3148.508 ; gain = 0.000 ; free physical = 9591 ; free virtu
al = 24740
Vivado%
```

Figure 26 MCS File Creation

1) Open the Vivado tcl console by using the following command: “vivado -mode tcl”.

2) Execute the following command to generate the mcs file:

   >> write_cfgmem -force -format mcs -interface <interface_type> -size <size> -loadbit “up <user_config_region_offset> <input_file.bit>” -file “output_file.mcs”

   Note: The values for “interface_type”, “size”, and “user_config_region_offset” depend on the Alveo card model, as detailed in Table 1.

Table 1 Parameters for MCS File Creation

Alveo card model	Interface_type	size	user_config_region
X3522	spix4	256	0x01002000
U250/U50/U55C		128	

3) The operation result will be displayed upon completion, confirming the successful creation of the mcs file.

5.2 Download the MCS file via PCIe

1) Execute the “xbflash2” utility with root permissions using the following command:

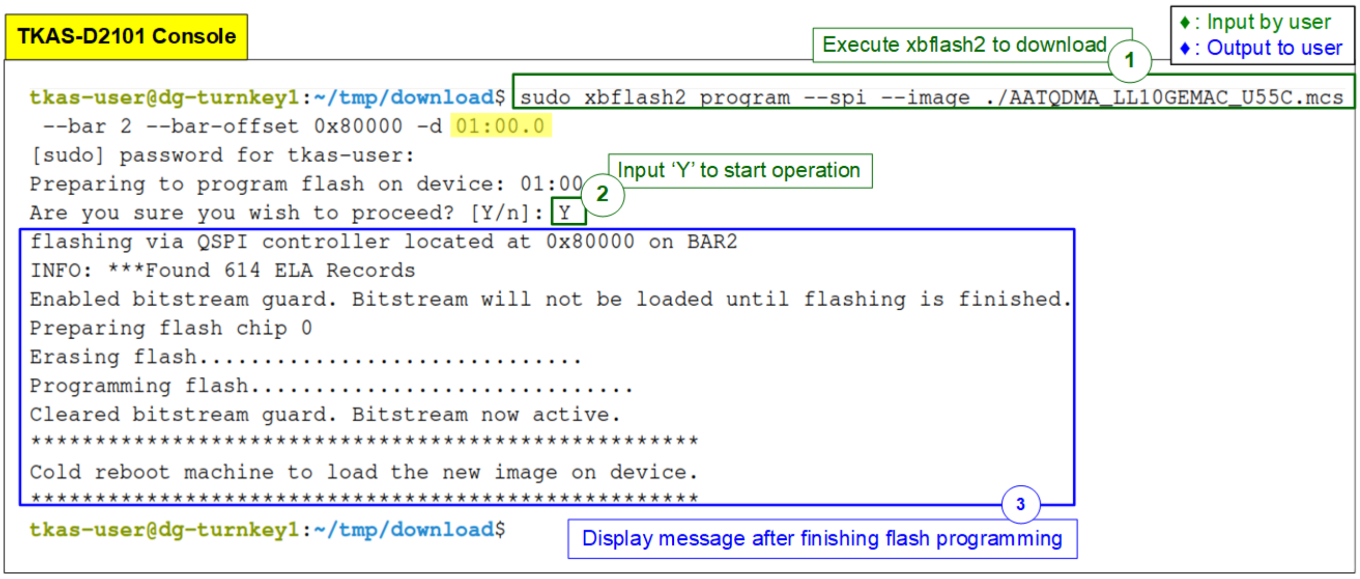
   >> sudo xbflash2 program --spi --image <target_mcs_file>.mcs --bar 2 --bar-offset 0x80000 -d <BDF>

   Note: The parameters “bar” and “bar-offset” are specific to the default AAT QDMA demo. If the current hardware on the card is not configured by the default AAT QDMA demo, these parameters may need to be adjusted accordingly.

2) Input ‘Y’ to confirm the operation.

3) Wait for the following message to appear on the console “Cold reboot machine to load the new image on device”.

4) Perform a cold reboot of the system. After rebooting, the new hardware configuration will be permanently implemented on the card.


TKAS-D2101 Console
♦: Input by user
♦: Output to user
Execute xbflash2 to download
1
Input ‘Y’ to start operation
2
3
Display message after finishing flash programming

```
tkas-user@dg-turnkey1:~/tmp/download$ sudo xbflash2 program --spi --image ./AATQDMA_LL10GEMAC_U55C.mcs
 --bar 2 --bar-offset 0x80000 -d 01:00.0
[sudo] password for tkas-user:
Preparing to program flash on device: 01:00
Are you sure you wish to proceed? [Y/n]: Y
flashing via QSPI controller located at 0x80000 on BAR2
INFO: ***Found 614 ELA Records
Enabled bitstream guard. Bitstream will not be loaded until flashing is finished.
Preparing flash chip 0
Erasing flash............................
Programming flash............................
Cleared bitstream guard. Bitstream now active.
****************************************************
Cold reboot machine to load the new image on device.
****************************************************
tkas-user@dg-turnkey1:~/tmp/download$
```

Figure 27 “xbflash2” Programming

6 Revision History

Revision	Date (D-M-Y)	Description
1.00	2-Dec-24	Initial version release