LL10GEMAC-IP with AAT Demo Instruction

Rev1.0 11-Oct-21

This document describes how to setup Alveo accelerator card and prepare the test environment for running AAT (Accelerated Algorithmic Trading) demo. The default demo is provided by Xilinx from following link.
https://www.xilinx.com/applications/data-center/financial-technology/accelerated-algorithmic-trading.html

The AAT demo is modified to use LL10GEMAC-IP from Design Gateway instead of 10G/25G Ethernet subsystem to achieve the lower latency time. More details of LL10GEMAC-IP latency time are described in the datasheet.
https://dgway.com/products/IP/Lowlatency-IP/dg_ll10gemacip_data_sheet_xilinx_en.pdf

The AAT demo is run by using Alveo accelerator card plugged into the system that supports to run the accelerator card demo. The accelerator card has QSFP+ connector which supports up to 4x10G Ethernet connection. The demo uses two of 10G Ethernet connection, one for transferring example market data via UDP protocol and another for the order via FIX over TCP. Therefore, the second system must be prepared with integrating two channels of 10G Ethernet connection. In this document, the second system is prepared by setting PC with 10G Ethernet card. “tcpreplay” must be run on the second system to send the example market data. Besides, “TCP port” must be opened to receive the order from the accelerator card via TCP. While the user controls the test operation on the accelerator card via “aat_shell_exe”.

1 Test environment

Before running the test, please prepare following test environment.

- Alveo accelerator card: U50 and U250
- Turnkey accelerator system, TKAS-D2101, for Alveo accelerator card
  https://dgway.com/AcceleratorCards.html
- 10 Gb Ethernet cable: QSFP+ to four SFP+ cable
  https://www.finisar.com/active-optical-cables/fcbn510qe2cxx
- Test PC for 10Gb Ethernet transferring with Turnkey accelerator system
  - Ubuntu 20.04 LTS Server OS
  - Example market data for running AAT demo
  - TCPreplay package for transmitting market data
  - Two ports of 10G Ethernet connection such as Intel X520-DA2
    http://www.intel.com/content/www/us/en/network-adapters/converged-network-adapters/ethernet-x520-server-adapters-brief.html

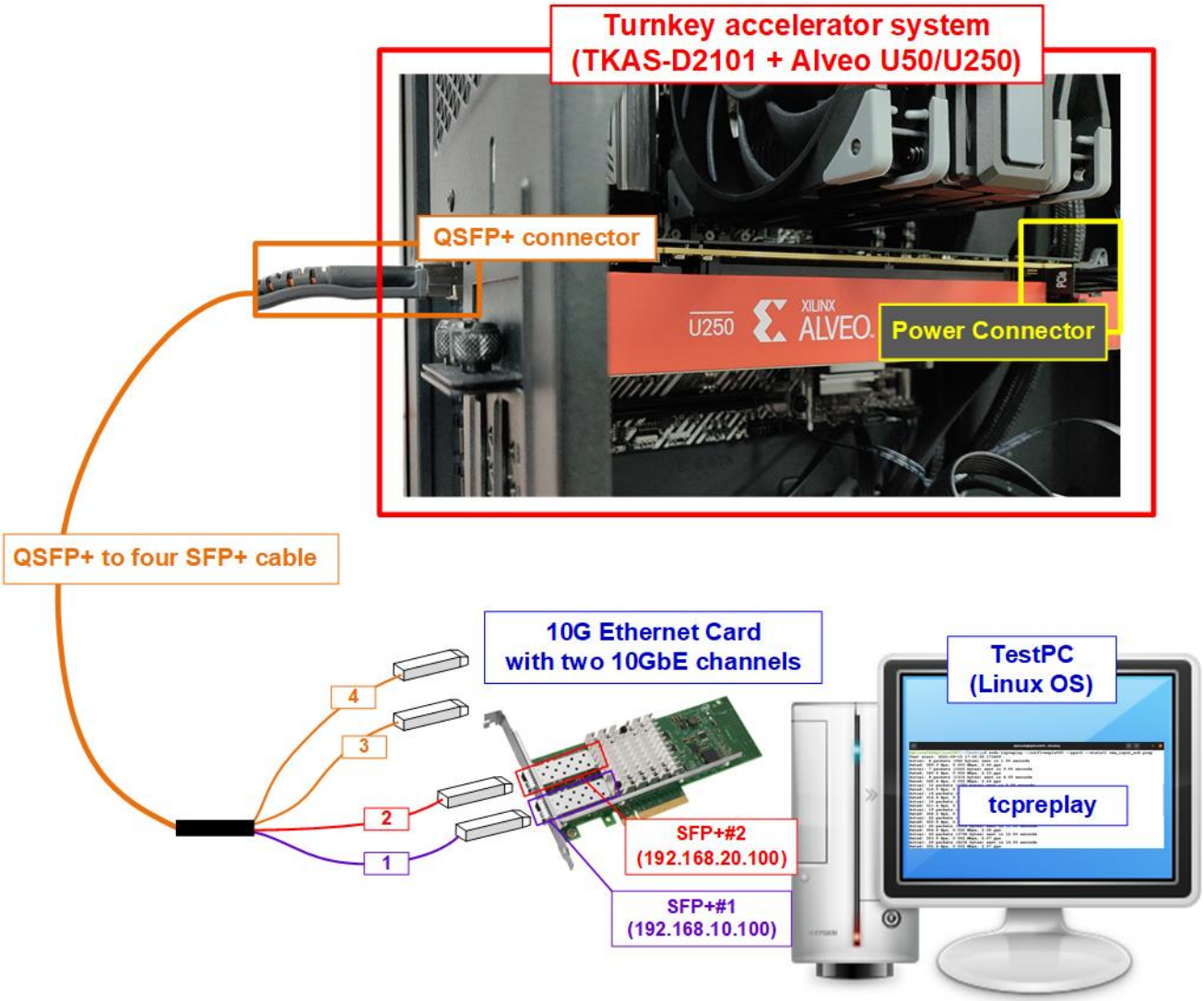

Figure 1-1 LL10GEMACIP with AAT demo (FPGA <-> PC) on Alveo U50/U250 card

2 Test PC setup

This topic shows how to prepare Test PC for transferring market data and the order packet with Alveo accelerator card. The example is the setting on Ubuntu 20.04 LTS Server OS.

2.1 IP Address setting for two ports of 10Gb Ethernet

Please confirm the logical name of Ethernet port that connects to SFP+#1 and SFP+#2 cable. It needs to configure the correct IP address for SFP+#1 and SFP+#2 connection.

1) To list the logical name of 10G Ethernet port on Linux terminal, type “lshw -C network”. Figure 2-1 shows the example result of the logical name of two 10Gb Ethernet connections. “enp1s0f0” is Ethernet port of SFP+#1 while “enp1s0f1” is Ethernet port of SFP+#2.

♦ : Input by user
♦ : Output to user

Test PC Console

```
dglinux02@dglinux02PC:~$ sudo lshw -C network
  *-network:0
       description: Ethernet interface
       product: Ethernet Controller X710 for 10GbE SFP+
       vendor: Intel Corporation
       physical id: 0
       bus info: pci@0000:01:00.0
       logical name: enp1s0f0
       version: 02
       serial: 80:61:5f:07:fa:d6
       width: 64 bits
       clock: 33MHz
       capabilities: pm msi msix pciexpress bus_master cap

  *-network:1
       description: Ethernet interface
       product: Ethernet Controller X710 for 10GbE SFP+
       vendor: Intel Corporation
       physical id: 0.1
       bus info: pci@0000:01:00.1
       logical name: enp1s0f1
       version: 02
       serial: 80:61:5f:07:fa:d7
       width: 64 bits
       clock: 33MHz
       capabilities: pm msi msix pciexpress bus_master cap
```

Display a list of network connection

Logical name of 10G no. 0

Logical name of 10G no. 1

Figure 2-1 Display logical name of 10G Ethernet port

2) Configure IP address of SFP+#1 (enp1s0f0) to “192.168.10.100” and SFP+#2 (enp1s0f1) to “192.168.20.100” respectively by using “ifconfig” command, as shown in Figure 2-2. Besides, the netmask 24 is set by using the same command.

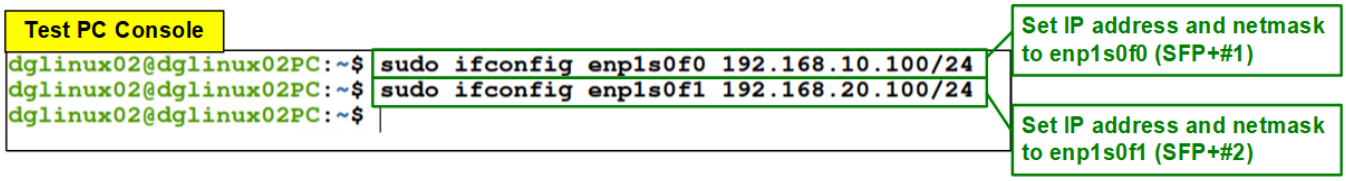

Figure 2-2 Configure IP address and netmask

3) Use “ifconfig” command to confirm IP address and netmask after setting completely.

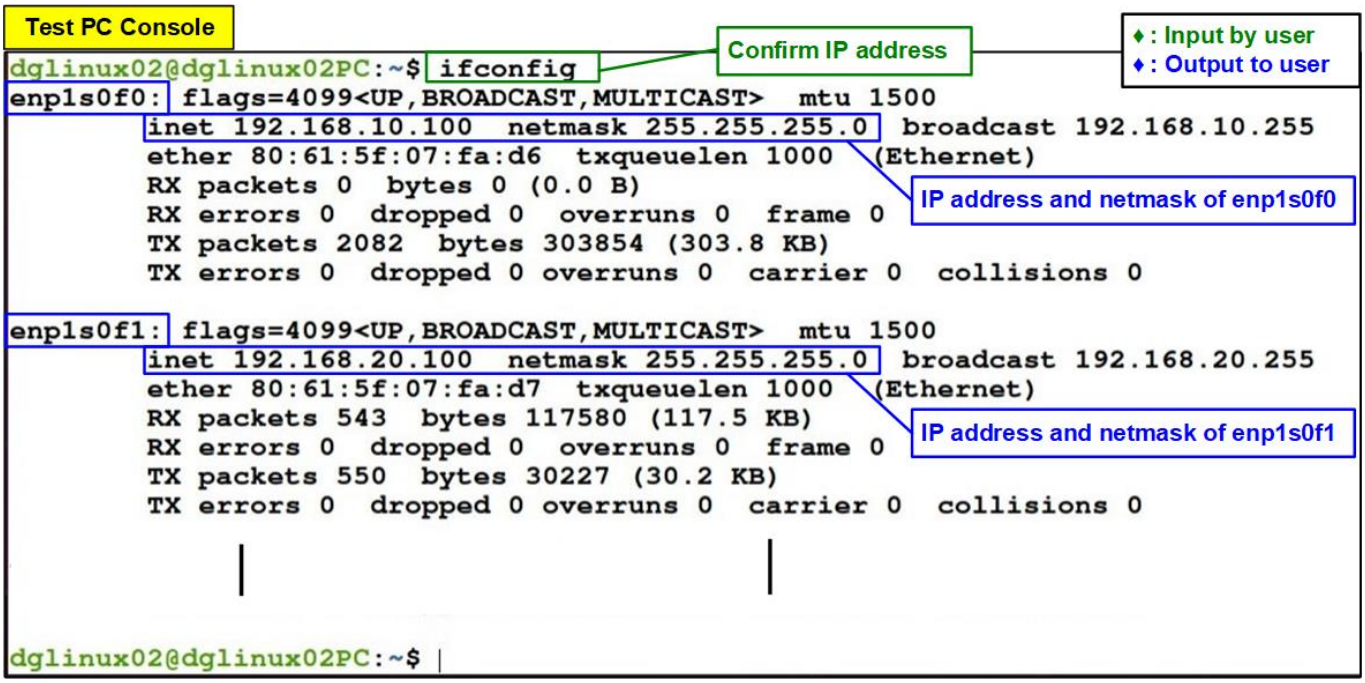

Figure 2-3 Verify IP address and netmask setting

2.2 “tcpreplay” installation

Test PC needs to install tcpreplay for running AAT demo. To install the package, type the command “sudo apt-get install tcpreplay” on the terminal, as shown in Figure 2-4.

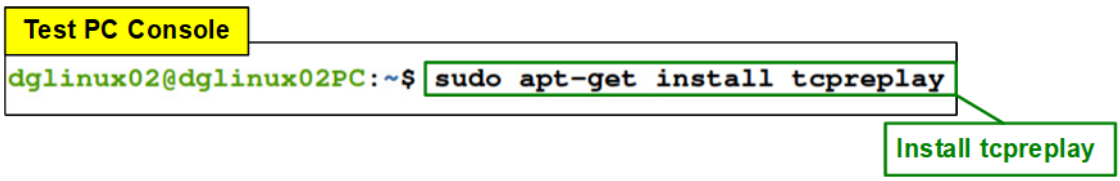

Figure 2-4 Install tcpreplay

3 Test environment setting

This topic shows the steps to prepare Turnkey acceleration system (TKAS-D2101 with U250/U50) to run AAT demo.

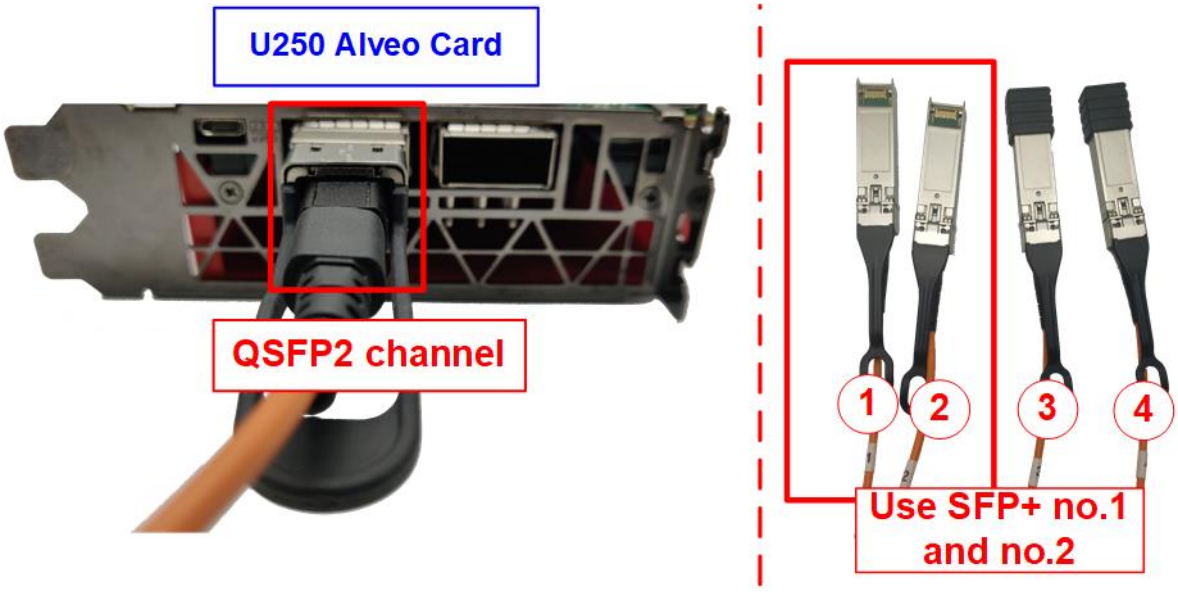

Figure 3-1 QSFP+ channel using on U250 board

1) Connect QSFP+ to four SFP+ cable (4x10G Ethernet cable) between Alveo accelerator card (U50/U250) and Test PC. Two SFP+ connectors (SFP+ no.1 and no.2) are applied.
   i. U250 card has two QSFP+ channel while U50 has one QSFP+ channel. Insert QSFP+ adapter to QSFP+ channel on Alveo accelerator card (QSFP2 channel for U250).
   ii. Connect SFP+ no.1 and SFP+ no.2 to 10Gb Ethernet channel on Test PC.

TKAS-D2101 Console

```
tkas-user@tkas-d2101:~$ source /tools/Xilinx/Vivado/2021.1/settings64.sh
tkas-user@tkas-d2101:~$ source /opt/xilinx/xrt/setup.sh
XILINX_XRT        : /opt/xilinx/xrt
PATH              : /opt/xilinx/xrt/bin:/tools/Xilinx/Vitis_HLS/2021.1/bin:/tools/Xilinx/Model_Composer/2021.1
/bin:/tools/Xilinx/Vitis/2021.1/bin:/tools/Xilinx/Vitis/2021.1/gnu/microblaze/lin/bin:/tools/Xilinx/Vitis/2021
.1/gnu/arm/lin/bin:/tools/Xilinx/Vitis/2021.1/gnu/microblaze/linux_toolchain/lin64_le/bin:/tools/Xilinx/Vitis/
2021.1/gnu/aarch32/lin/gcc-arm-linux-gnueabi/bin:/tools/Xilinx/Vitis/2021.1/gnu/aarch32/lin/gcc-arm-none-eabi/
bin:/tools/Xilinx/Vitis/2021.1/gnu/aarch64/lin/aarch64-linux/bin:/tools/Xilinx/Vitis/2021.1/gnu/aarch64/lin/aa
rch64-none/bin:/tools/Xilinx/Vitis/2021.1/gnu/armr5/lin/gcc-arm-none-eabi/bin:/tools/Xilinx/Vitis/2021.1/tps/l
nx64/cmake-3.3.2/bin:/tools/Xilinx/Vitis/2021.1/aietools/bin:/tools/Xilinx/Vivado/2021.1/bin:/tools/Xilinx/Doc
Nav:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
LD_LIBRARY_PATH   : /opt/xilinx/xrt/lib:
PYTHONPATH        : /opt/xilinx/xrt/python:
tkas-user@tkas-d2101:~$ export PLATFORM_REPO_PATHS='/opt/xilinx/platforms'
tkas-user@tkas-d2101:~$ export XILINX_PLATFORM='xilinx_u250_gen3x16_xdma_3_1_202020_1'
tkas-user@tkas-d2101:~$ export DEVICE=${PLATFORM_REPO_PATHS}/${XILINX_PLATFORM}/${XILINX_PLATFORM}.xpfm
tkas-user@tkas-d2101:~$
```

i. Call Xilinx environment and library

ii. Open Xilinx run time

iii. Set Target accelerator card

Figure 3-2 Xilinx environment setting on TKAS-D2101 for U250

2) Prepare Xilinx environment and library on TKAS-D2101 which connects U50/U250 through PCIe connector.
   i. Input command to setup Xilinx environment and library on terminal
      >> source <install path>/Vivado/2021.1/settings64.sh
   ii. Input command to open Xilinx run time
      >> source /opt/xilinx/xrt/setup.sh
   iii. Input command to set environment for the accelerator card (U50 or U250).
      >> export PLATFORM_REPO_PATHS='/opt/xilinx/platforms'

      For U250 card
      >> export XILINX_PLATFORM='xilinx_u250_gen3x16_xdma_3_1_202020_1'
      For U50 card
      >> export XILINX_PLATFORM='xilinx_u50_gen3x16_xdma_201920_3'

      >> export DEVICE=${PLATFORM_REPO_PATHS}/${XILINX_PLATFORM}/ ${XILINX_PLATFORM}.xpfm

3) For U250 card, before downloading xclbin file to accelerator card, the accelerator card needs to prepare partition for programming. This step can be skipped for U50 card. More details are described in page 32 of UG1301 (V1.9).
https://www.xilinx.com/support/documentation/boards_and_kits/accelerator-cards/1_9/ug1301-getting-started-guide-alveo-accelerator-cards.pdf

i. Detect the accelerator card which connects on TKAS-D2101 to get the details such as Card BDF and shell name by using following command.
>> sudo /opt/Xilinx/xrt/bin/xbmgmt partition --scan

♦ : Input by user
♦ : Output to user

TKAS-D2101 Console

i.Display details of accelerator card

```
tkas-user@tkas-d2101:~$ sudo /opt/xilinx/xrt/bin/xbmgmt partition --scan
-----------------------------------------------------------------------
Deprecation Warning:
    The given legacy sub-command and/or option has been deprecated
    to be obsoleted in the next release.

    Further information regarding the legacy deprecated sub-commands
    and options along with their mappings to the next generation
    sub-commands and options can be found on the Xilinx Runtime (XRT)
    documentation page:

    https://xilinx.github.io/XRT/master/html/xbtools_map.html

    Please update your scripts and tools to use the next generation
    sub-commands and options.
-----------------------------------------------------------------------
Card [0000:01:00.0]
    Partitions running on FPGA:
        xilinx_u250_gen3x16_base_3
            logic-uuid:
            48810c9d17860ef53e9e529e8b14ce39
            interface-uuid:
            695718ec21a232e45e1afcb4e558e11f
        xilinx_u250_gen3x16_xdma_shell_3_1
            logic-uuid:
            bd5fb8abab266c3265918257b5048e88
            interface-uuid:
            f2f6c5e1273e78948f2c4806221462f2
    Partitions installed in system:
        xilinx_u250_gen3x16_xdma_shell_3_1
            logic-uuid:
            bd5fb8abab266c3265918257b5048e88
            interface-uuid:
            f2f6c5e1273e78948f2c4806221462f2

tkas-user@tkas-d2101:~$
```

Card BDF (Board:Device.Function)

Shell name

Figure 3-3 Scan accelerator card

ii. Program partition to accelerator. User needs to know <shell_name> and <card_BDF> which are displayed on the console when running scan command in step 3i).
>> sudo /opt/xilinx/xrt/bin/xbmgmt partition –program –name <shell_name> --card <card_BDF>

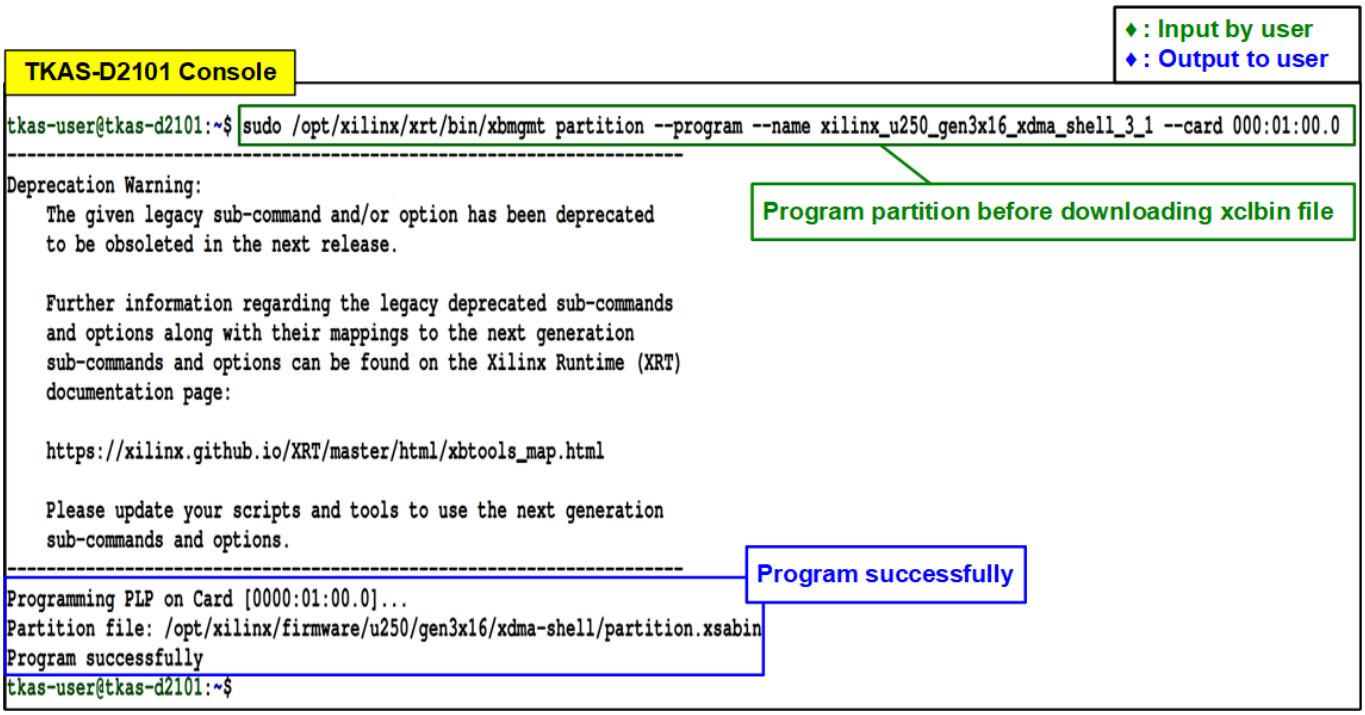

Figure 3-4 Program partition

4) Copy "aat.u250_DGLL10GEMAC.xclbin" file that is provided by DesignGateway to the same directory as "aat_shell_exe". The default path of "aat_shell_exe" in AAT demo design is "../Accelerated_Algorithmic_Trading/build".
Note: If there is no "aat_shell_exe" in build directory, user needs to build it following the description in chapter 7 (Building and Running the AAT) of "UG1067 Accelerated Algorithmic Trading User Guide" document.

5) Browse to directory that includes aat_shell_exe and xclbin file of the demo. After that, run the demo using following command.
>> cd <directory of aat_shell_exe>
>> ./aat_shell_exe
>> download aat.u250_DGLL10GEMAC.xclbin

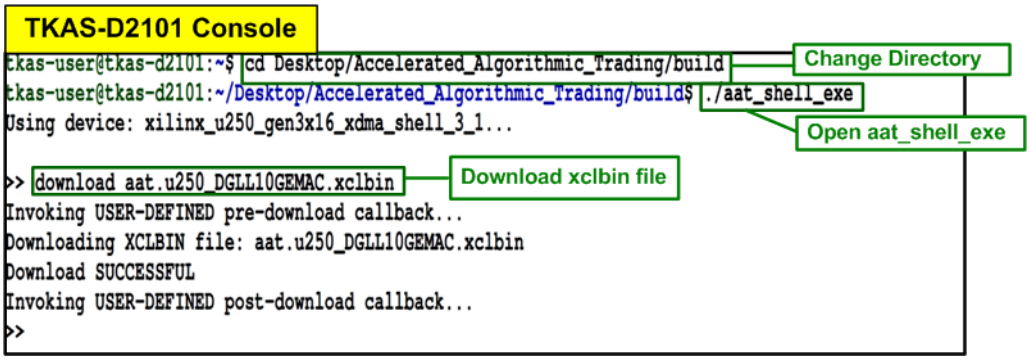

Figure 3-5 Download xclbin file

4 Run AAT Demo

To run the demo, there are three processes for user running. First is the initialization process for preparing the connection and parameter configuration. Next is market data transmission process for start sending market data from Test PC. Last is the test status that is returned by the accelerator card to show the result on the console of TKAS-D2101. More details of each process are described as follows.

4.1 Initialization

To run AAT demo, the user must input the command on Test PC console to listen the specific port that is applied to receive the order packet from accelerator card after finishing processing market data. Similarly, the accelerator card must be configured to set up the parameters for receiving market data and sending the order packet by using script file “demo_setup_cfg”. More details for system initialization are described as follows.

1) On Test PC console, type following command to listen the port no. 12345.
>> nc -l 192.168.20.100 12345 -v

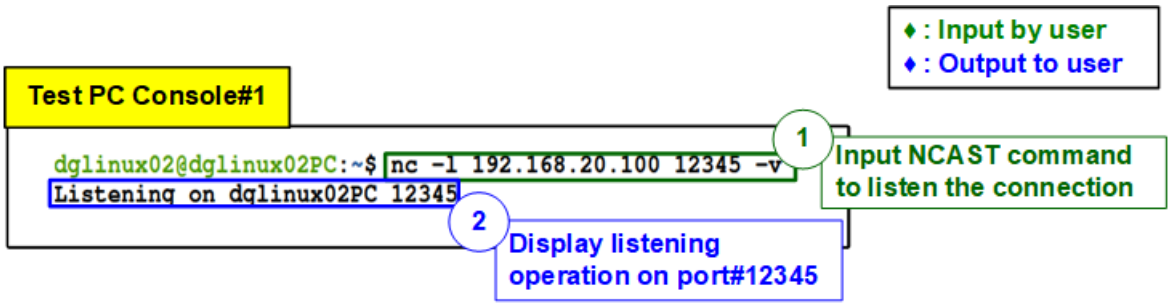

Figure 4-1 Listen TCP port on Test PC

2) After that, the confirmation message (“Listening on <Test PC name> 12345”) is displayed on the console to confirm port is listening, as shown in Figure 4-1.

3) Run script file on TKAS-D2101 to setup the parameters for processing market data by using following command.

>> run support/demo_setup.cfg

Note: demo_setup.cfg can run only one time after downloading xclbin file. To rerun the script file, it needs to re-download xclbin file.

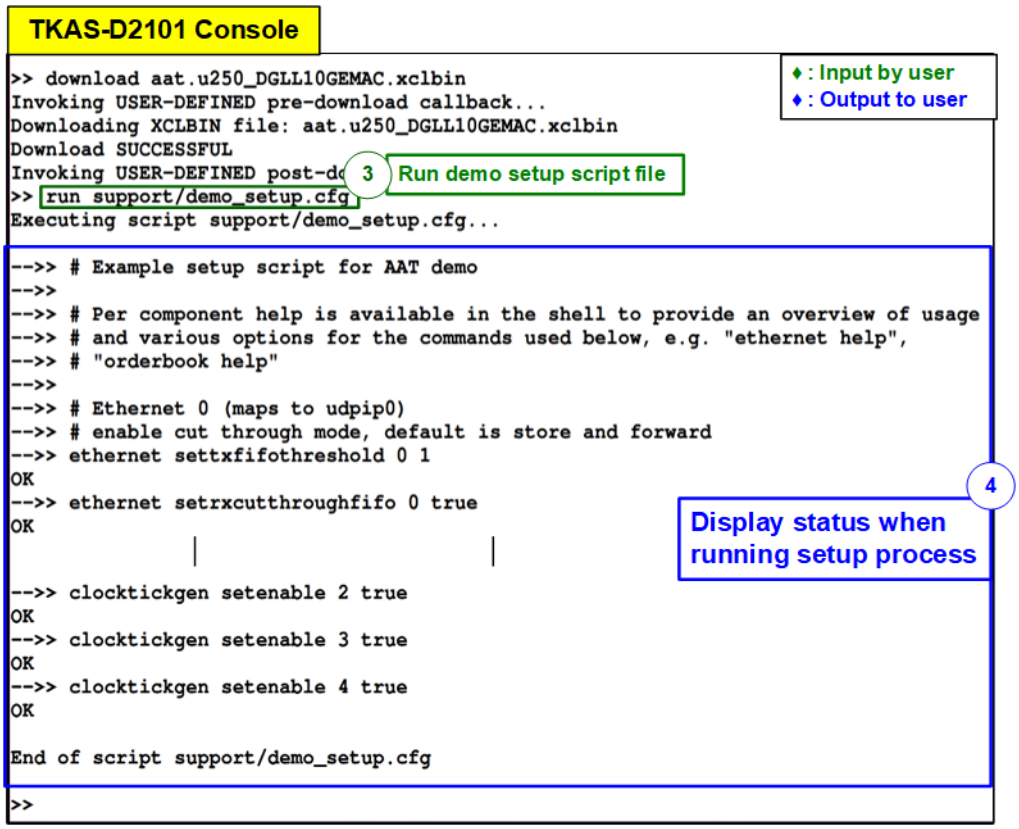

Figure 4-2 Run demo setup script

4) After that, TKAS-D2101 displays the message from setup process, as shown in Figure 4-2.

5) If the parameter is configured successfully, Test PC console displays the message that the port is opened successfully ("Connection received on 192.168.20.200 32768"), as shown in Figure 4-3.

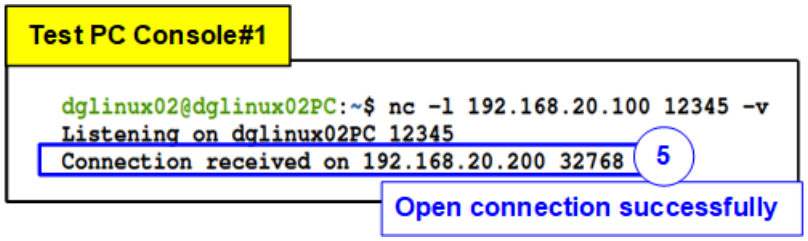

Figure 4-3 Open connection successes

4.2 Market data transmission

This topic shows how to run “tcpreplay” on Test PC to send sample market data. Also, the result on Test PC when the accelerator card returns the order packet is displayed on the listening port. Therefore, the user needs to open two consoles on Test PC, Test PC Console#1 and Test PC Console#2. Test PC Console#1 shows the details of the order packet while Test PC Console#2 is applied to send the sample market data. More details to transmit sample market data are described as follows.

1) Send sample market data (cme_input_arb.pcap which is provided by Xilinx AAT demo) by using “tcpreplay” command. Type following command with four parameters.
   >> sudo tcpreplay –intf1=<eth I/F> --pps=<pac/sec> --stats=<stat period> <replay file>
   i) <eth I/F> : Ethernet interface for sending market data (SFP+#1: enp1s0f0)
   ii) <pac/sec> : Transfer speed by setting number of packets per second
   iii) <stat period> : Set time period time in second unit to display status on console
   iv) <replay file> : File name to transmit the data, cme_input_arb.pcap
   Note: cme_input_arb.pcap is the sample market data, provided by Xilinx AAT demo. Please contact Xilinx to request the sample market data and AAT demo.

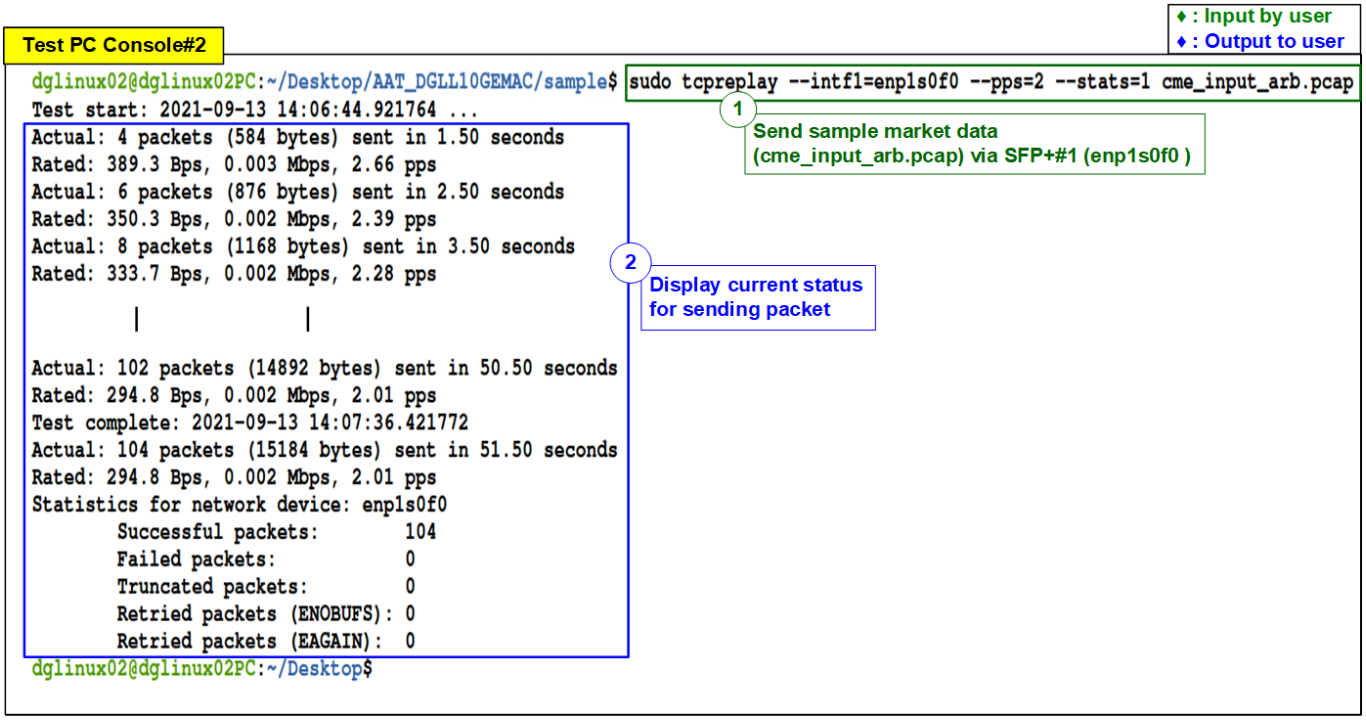

Figure 4-4 Send sample market data by “tcpreplay”

2) After that, the console displays the status to show total number of transmit packets every second.

3) On Test PC console#1 which the port has already connected, the console displays the received data which is the sample order packet, returned by the accelerator card to be the AAT demo result.

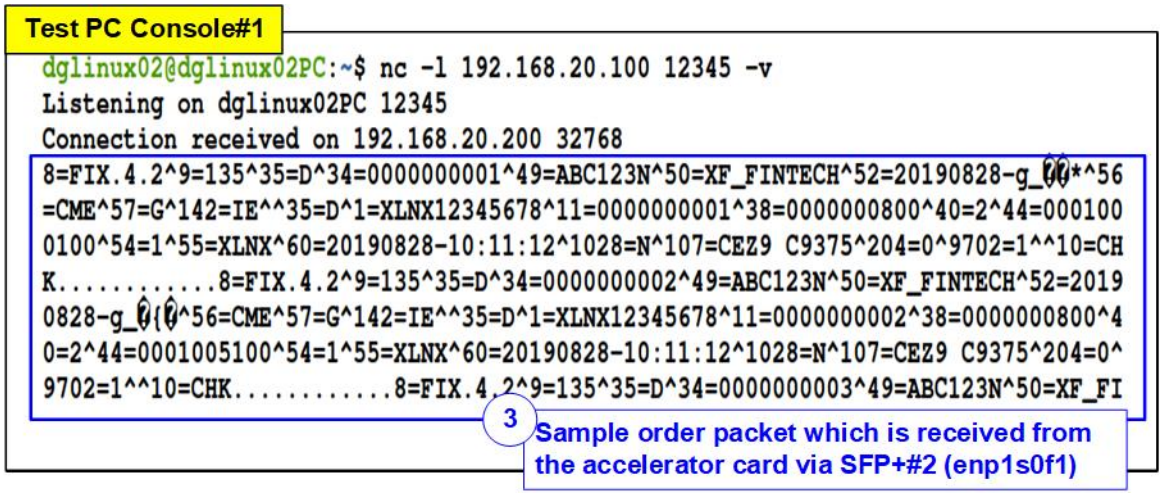

Figure 4-5 The sample data of order packet on SFP+#2 channel

4.3 AAT demo

This topic shows the example result of market data processing on the accelerator card. There are many kernels for processing the sample market data. This topic shows the result of four kernels in AAT demo design, i.e., Ethernet kernel, Feed handler kernel, Order book kernel, and Order entry kernel. More details of the sample results are shown as follows.

4.3.1 Ethernet Kernel

1) User inputs the following command to display the status of Ethernet kernel.
>> ethernet getstatus

```
TKAS-D2101 Console
>> ethernet getstatus                    (1) Display status of all ethernet channels

+-------------------------------+-------------------------+
| CU Index                      |                       1 |
| CU Address                    |   0x0000000001420000    |
+-------------------------------+-------------------------+
| Num Supported Channels (HW)   |                       4 |
+-------------------------------+-------------------------+
                                                          (2)
                                Four ethernet channels
                                are available in AAT demo

                                Channel#0 status
                                (for receiving market data)
+-----------------------+-----------------------------+-----------------------+
| 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 |
+-----------------------+-----------------------------+-----------------------+

♦ : Input by user
♦ : Output to user

                                Channel#1 status
                                (for sending order packet)
+-----------------------+-----------------------------+-----------------------+
| 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 4-6: Ethernet kernel status

2) There are four ethernet channels (channel0 – channel3) on AAT demo system. The document shows the example to use channel#0 for receiving sample market data and channel#1 for returning sample order packet. Please confirm the status of channel#0 and channel#1 are in good status.
   i) Rx Block Lock Status (Live) : LOCKED
   ii) GT Power Good (Live) : true

4.3.2 Feed Handler Kernel

1) User inputs the following command to display the status of Feed handler kernel.

>> feedhandler getstatus

2) The console displays the processed data count in several units such as bytes, packets, and messages. As shown in Figure 4-7, the left window shows the processed data count is equal to zero before starting transmitting sample market data. After finishing transmitting market data, the processed data count is not equal to zero.

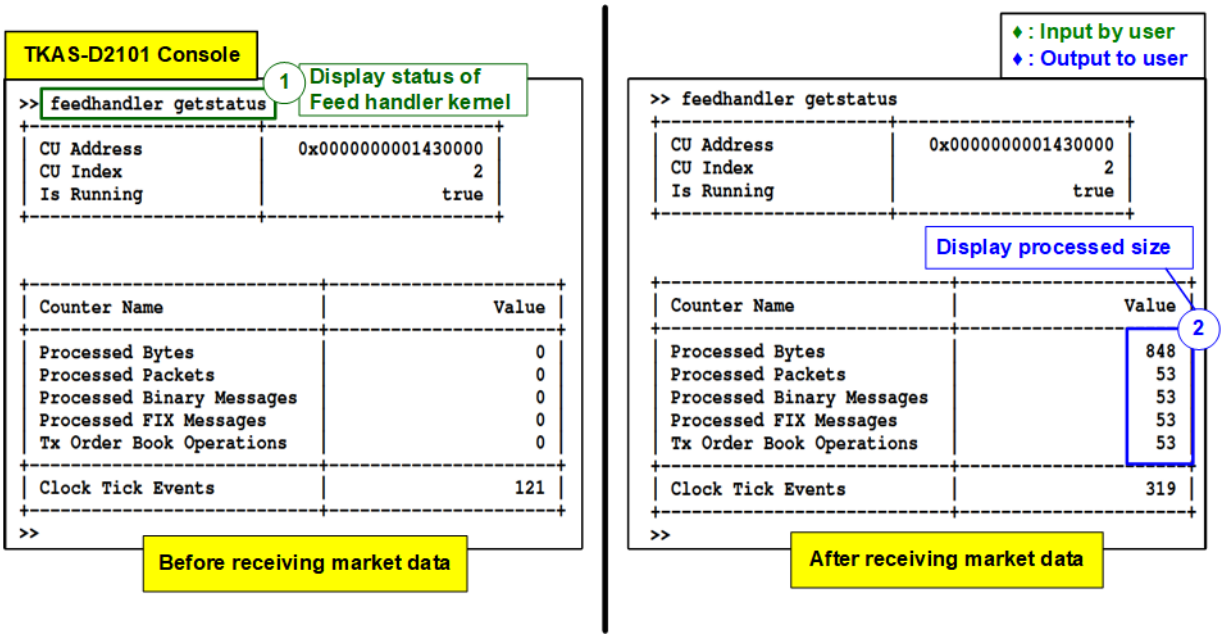

Figure 4-7 Feed handler kernel status

As shown in Figure 4-4, there are 104 packets of sample market data sent by Test PC. However, there are 53 packets that are valid for Feed handler processing. Other packets are rejected by Feed handler.

4.3.3 OrderBook Kernel

1) User inputs the following command to display the order book, the output from Order Book kernel.

>> orderbook readdata

2) The console displays the current value of order book. As shown in Figure 4-8, the left window shows clean status of order book before the system transmits sample market data. While the right window shows the updated order book after transferring all sample market data. The bid/ask quantity and the bid/ask price are updated by OrderBook kernel.

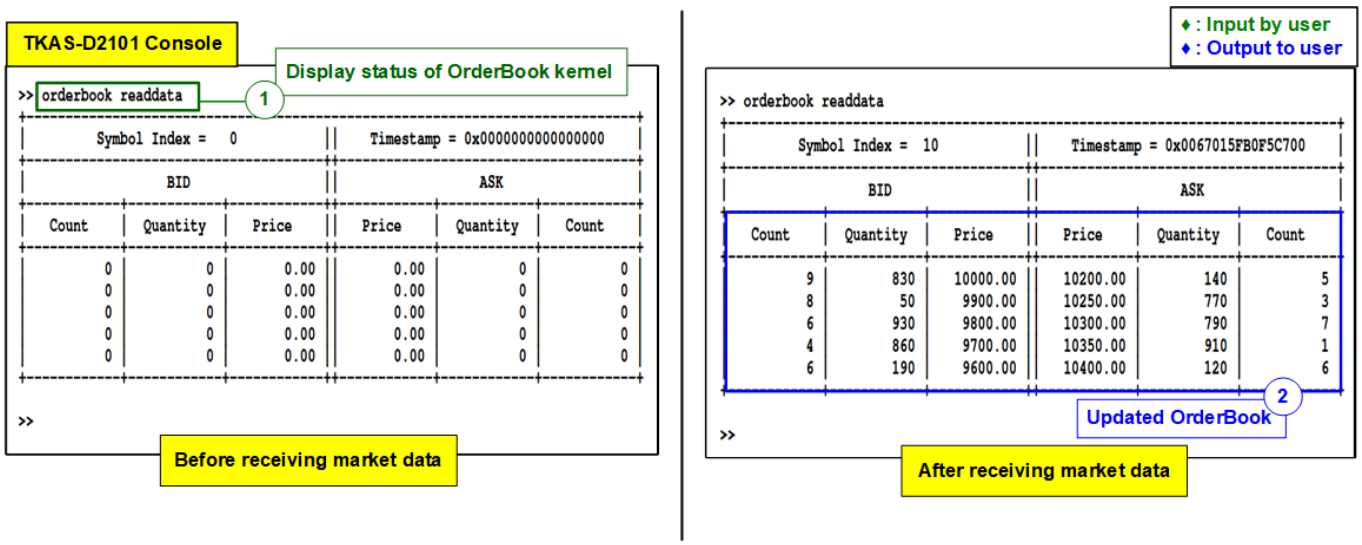

Figure 4-8 Updated OrderBook after finishing processing

4.3.4 Order Entry Kernel

1) User inputs the following command to display the current status of Order Entry kernel.
>> orderentry status
2) After starting AAT demo before starting sending the sample market data, the user can check the status of Ethernet channel#1 from Order Entry kernel status.
   i) Connection Established : true
   ii) Connection Status : SUCCESS

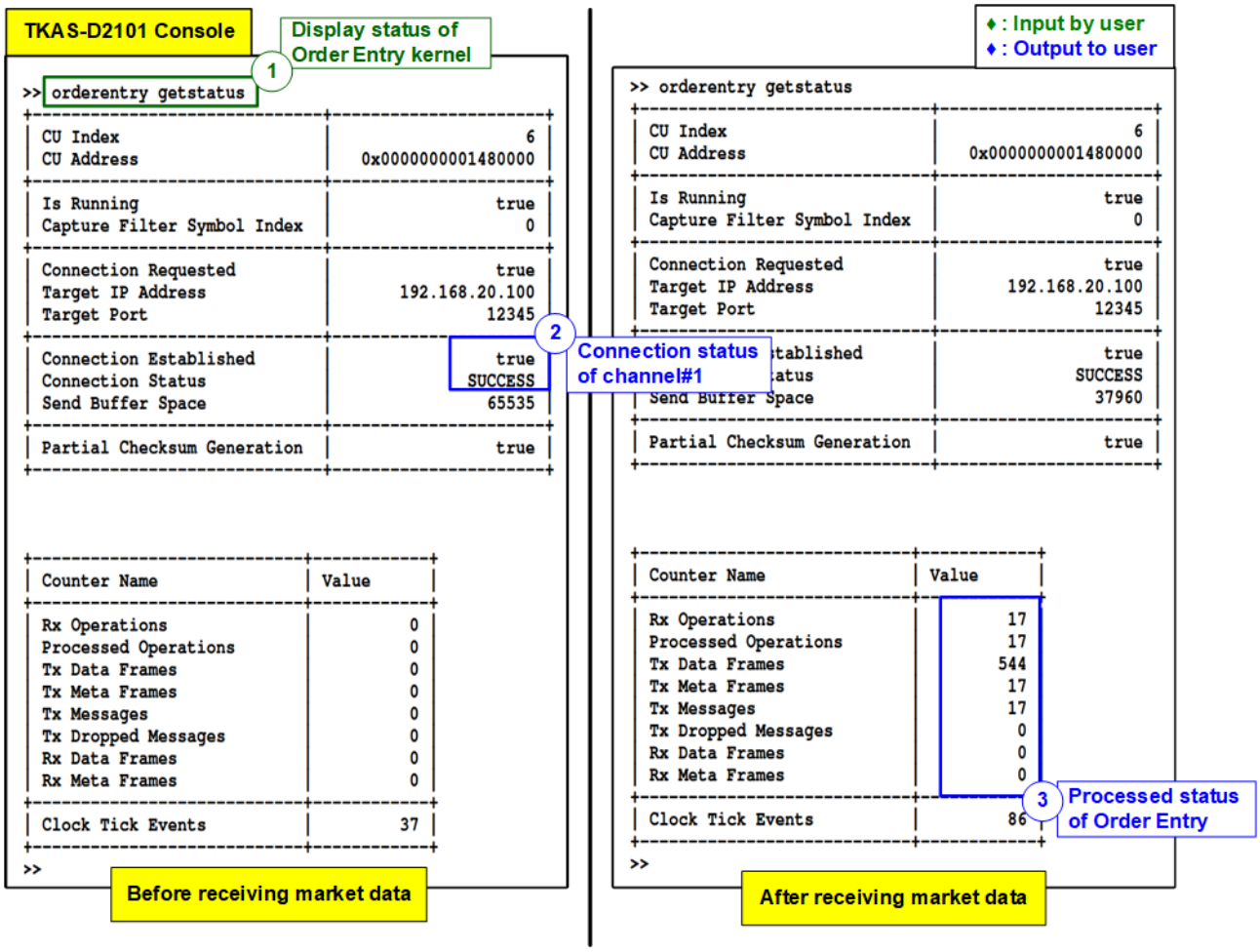

Figure 4-9 Order Entry kernel status

3) After transmitting the market data completely, the packet count of Order Entry is updated from 0 to the new value to show total numbers of message/frames that is processed by Order Entry kernel.

5 Revision History

Revision	Date	Description
1.0	11-Oct-21	Initial version release