User Guide

For

QDMA Ethernet Platform

(DPDK Driver)

Project Number: 2000-0158

Author: Pankaj Darak

CONTENTS

LIST OF TABLES

1 Introduction

1.1 Document Overview

This User Guide provides the setup procedure and software usage instructions of the DPDK driver for Network PF of QDMA Ethernet Platform.

1.2 Document References

Document References	Version
[1] QDMA Subsystem for PCI Express (PG302)	3.0
[2] UltraScale+ Devices Integrated 100G Ethernet Subsystem (PG203)	2.6

Table 1-1: Document References

1.3 Glossary

Acronym / Term	Description
BAR	Base Address Register
C2H	Card to Host
DMA	Direct Memory Access
DPDK	Data Plane Development Kit
DSA	Device Support Archive
H2C	Host to Card
HW	Hardware
IP	Intellectual Property
PCIe	Peripheral Component Interconnect Express
PF	Physical Function
PMD	Poll Mode Driver
QDMA	Multi Queue Direct Memory Access
QEP	QDMA Ethernet Platform
STM	Streaming Traffic Manager
STM-N	Streaming Traffic Manager for Networking

Table 1-2: Glossary

2 System Overview

The design adds Ethernet support to Streaming DSA (QDMA based) platform. The Ethernet Subsystem is added to the static region of the shell. The platform has three physical functions, two physical functions for device management (PF0) and compute acceleration (PF1), and one physical function (PF2) for Network acceleration. The Ethernet subsystem is accessible to the host via PF2.

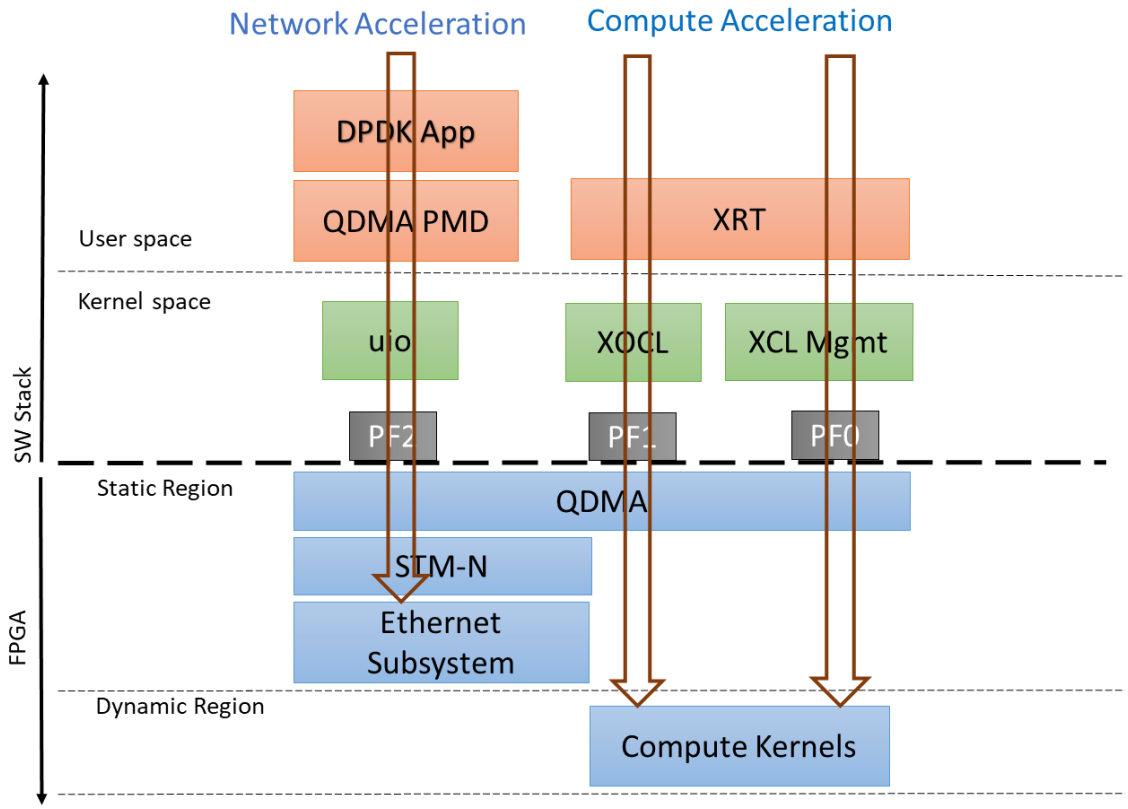

Figure 1: Software Architecture

3 QEP DPDK Driver

This User Guide document describes the following for QEP DPDK Driver that will be generally available for customers:

- System Setup
- Compiling and loading the driver
- Sample commands to use the driver and test application

3.1 System Setup

This release was tested with the following system configuration.

Host System Configuration	Description
Operating System	Ubuntu 18.04.1 LTS
Linux Kernel	4.15.0-54-generic
RAM	32GB on local NUMA node

Table 3-1: System Configuration

The following modifications must be made to the /boot/grub/grub.cfg on the Host system:

- Add hugepages for DPDK
    - Add following parameters to /etc/default/grub file

      ```
      GRUB_CMDLINE_LINUX="default_hugepagesz=1GB hugepagesz=1G hugepages=6"
      ```

      The example above adds six 1GB hugepages, which are required to support 256 queues, with descriptor ring of 2048 entries, and each descriptor supporting buffer size of 4KB.

      The number of hugepages required should be changed if the above configuration (queues, ring size, buffer size) changes.
- Execute the following command to modify the /boot/grub/grub.cfg with the configuration set in the above steps and permanently add them to the kernel command line:
    - `update-grub`

Reboot host after making the above modifications.

3.2 Environment

To execute the QEP DPDK driver and testpmd application on the example design, following system requirements are to be met:

1. Host System with at least one Gen3 x16 PCIe slot and minimum 16GB RAM on same CPU node.
2. Execution of the steps mentioned in section 3.1
3. U250 Board
4. USB digilent cables to connect the U250 board to the Host System.

5. Xilinx 2019.1 Vivado tools for programming the FPGA.
6. SDAccel Platform DSA with networking subsystem support

Refer to https://www.xilinx.com/products/boards-and-kits/alveo/u250.html#gettingStarted on setting up the board and programming the FPGA.

3.3 Modifying the driver for your own PCIe device ID

The current driver is designed to recognize the PCIe Device ID that gets generated with the example design. If the PCIe Device ID is modified during IP customization, one needs to modify QEP PMD to recognize this new ID.

User can also remove PCIe Device IDs that will not be used by the end solution or add new device IDs used for network PF. To modify the PCIe Device ID in the driver,

Update `struct rte_pci_id qdma_pci_id_tbl[]` in the file `drivers/net/qep/qdma_ethdev.c`.

Also add the device IDs in `usertools/dpdk-devbind.py` in `xilinx_qep_pf` as specified in section 3.4.

Once modified, the driver and application must be recompiled.

3.4 Building the QEP DPDK Software

DPDK requires certain packages to be installed on the host system. For a full list, refer to the official DPDK documentation:

https://doc.dpdk.org/guides/linux_gsg/sys_reqs.html.

Note: If the NUMA library is missing, it should be installed. For example:

```
For ubuntu:> sudo apt-get install libnuma-dev
```

Below Table describes the DPDK software database structure and its contents on the Xilinx GitHub https://github.com/Xilinx/qep-drivers, subdirectory DPDK.

Directory	Description
drivers/net/qep	Xilinx QEP DPDK Poll Mode Driver
ug03-2000-0158.pdf	This document (User guide)
RELEASE.txt	Release Notes

Table 3-2: DPDK software database content

3.4.1 Setup: Download and modifications

The driver code requires DPDK version 18.11. Follow the steps below to download the proper version of DPDK and apply driver code supplied on the GitHub.

Extract the DPDK driver software database from GitHub to the host system where U250 card is installed. Henceforth, this area is referred to as `<dpdk_sw_database>`.

Create a directory to download the DPDK source on the host system where the U250 card is installed and move to this directory.

```
mkdir <host_dir>/<dpdk_test_area>
cd <host_dir>/<dpdk_test_area>
git clone http://dpdk.org/git/dpdk-stable
cd dpdk-stable
git checkout v18.11
cp -r <dpdk_sw_database>/drivers/net/qep ./drivers/net/
```

Additionally, make below changes to the DPDK 18.11 tree to build QEP driver, and populate Xilinx devices for binding.

1. To build QEP driver

   a. Add below lines to `./config/common_base` in DPDK 18.11 tree

   ```
   #
   #Complie Xilinx QEP PMD driver
   #
   CONFIG_RTE_LIBRTE_QEP_PMD=y
   CONFIG_RTE_LIBRTE_QDMA_DEBUG_DRIVER=n
   ```

   To enable driver debug logs, set

   ```
   CONFIG_RTE_LIBRTE_QDMA_DEBUG_DRIVER=y
   ```

   b. Add below lines to `drivers/net/Makefile`, where PMDs are added

   ```
   DIRS-$(CONFIG_RTE_LIBRTE_QEP_PMD) += qep
   ```

   c. Add below lines to `mk/rte.app.mk`, where PMDs are added

   ```
   _LDLIBS-$(CONFIG_RTE_LIBRTE_QEP_PMD) += -lrte_pmd_qep
   ```

2. To add Xilinx devices for device binding, add below lines to `./usertools/dpdk-devbind.py` after `cavium_pkx` class, where PCI base class for devices are listed.

   ```
   xilinx_qep_pf = {'Class':  '05', 'Vendor': '10ee',
   'Device': '7002',
                   'SVendor': None, 'SDevice': None}
   ```

   Update entries in network devices class in `./usertools/dpdk-devbind.py` to add Xilinx devices

   ```
   network_devices = [network_class, cavium_pkx,
   xilinx_qep_pf]
   ```

3.4.2 Setup: Huge Pages

DPDK requires that hugepages are setup on the server. Perform steps outlined in section 3.1 to reserve hugepages.

3.4.3 Setup: Make Commands

Execute the following to compile the driver:

```
cd <host_dir>/<dpdk_test_area>/dpdk-stable
make config T=x86_64-native-linuxapp-gcc install
```

```
#In the make output, verify that the QEP files are being
built. Below figure shows the QEP files that are built as part
of make.
```

```
== Build drivers/net/qep
  SYMLINK-FILE include/rte_pmd_qep.h
  CC qdma_ethdev.o
  PMDINFO qdma_ethdev.o.pmd.c
  CC qdma_ethdev.o.pmd.o
  LD qdma_ethdev.o
  CC qdma_vf_ethdev.o
  PMDINFO qdma_vf_ethdev.o.pmd.c
  CC qdma_vf_ethdev.o.pmd.o
  LD qdma_vf_ethdev.o
  CC qdma_devops.o
  CC qdma_common.o
  CC qdma_rxtx.o
  CC qdma_xdebug.o
  CC qdma_access.o
  CC qdma_list.o
  CC qdma_resource_mgmt.o
  CC qdma_mbox_protocol.o
  CC qdma_mbox.o
  CC qdma_platform.o
  CC rte_pmd_qdma.o
  CC qdma_flow.o
  CC xcmac.o
  CC stmn.o
  CC xcmac_rsfec.o
  CC qdma_xcmac.o
  CC rte_pmd_qep.o
  AR librte_pmd_qep.a
  INSTALL-LIB librte_pmd_qep.a
```

```
#The following should appear when make completes:

      Build complete [x86_64-native-linuxapp-gcc]

#Verify that librte_pmd_qep.a is installed in ./x86_64-native-
linuxapp-gcc/lib/ directory.
```

```
#*NOTE: If any of above steps are missed or require code
modifications, perform 'make clean' before re-running make. For
driver related modifications, perform 'make clean' from inside
x86_64-native-linuxapp-gcc directory
```

3.5 Running the DPDK testpmd application

The below steps describe the step by step procedure to run the DPDK testpmd application.

1. Navigate to DPDK root directory.

   ```
   # cd <host_dir>/<dpdk_test_area>/dpdk-stable/
   ```

2. Run the 'lspci' command on the console and verify that the PFs are detected as shown below. Here, '81' is the PCIe bus number on which Xilinx device is installed.

   ```
   # lspci | grep Xilinx
   81:00.0 Processing Accelerators: Xilinx Corporation Device 7000
   81:00.1 Processing Accelerators: Xilinx Corporation Device 7001
   81:00.2 Ethernet controller: Xilinx Corporation Device 7002
   ```

3. Execute the following commands required for running the DPDK application:

   ```
   # mkdir /mnt/huge
   # mount -t hugetlbfs nodev /mnt/huge
   # modprobe uio
   # insmod x86_64-native-linuxapp-gcc/kmod/igb_uio.ko
   ```

4. Bind PF ports to the igb_uio module as shown below:

   ```
   # usertools/dpdk-devbind.py -b igb_uio 81:00.2
   ```

5. Run the testpmd using the following command:

   ```
   # ./x86_64-native-linuxapp-gcc/app/testpmd -c3ff -n4 -w 81:00.2 -
   - -i --nb-cores=9 --rxq=8 --txq=8 --rxd=2048 --txd=2048 --
   burst=64 --mbuf-size=4224 --max-pkt-len=9000
   ```

   Refer testpmd application user guide (https://fast.dpdk.org/doc/pdf-guides-18.11/testpmd_app_ug-18.11.pdf) for details on testpmd execution.

3.6 Configuring the driver

3.6.1 Supported Device arguments (module parameters)

Device specific parameters can be passed to a device by using the '-w' EAL option. Xilinx supports following device arguments to configure QEP DPDK Driver.

Devargs options	Description
queue_base	Starting HW queue id for the given PCIe function. User needs to make sure that the queues belonging to different PCIe functions do not share same HW queue id. Default value of queue_base is 0. Example usage: `./x86_64-native-linuxapp-gcc/app/testpmd -c 0x1f -n 4 -w 81:00.2,queue_base=64` In this example, the device "81:00.2" uses absolute queue id starting from 64.
rsfec	Specifies whether to enable or disable RS-FEC. Default is set to 1 i.e. enable in the driver. Example usage: `./x86_64-native-linuxapp-gcc/app/testpmd -c 0x1f -n 4 -w 81:00.2,queue_base=64,rsfec=0` This example disables RS-FEC on the MAC for device "81:00.2".

Table 3-3: Device arguments supported by DPDK driver

4 Document Revision History

Version	Date	Description	State
4	12-Jul-2019	QEP DPDK driver user guide for 0.1.0 release. Reviewed and Approved by Srikanth Chatla	Released

Table 4-1: Document Review History