**DPDK and QDMA Training Document**

1. First, we should have to learn Basic Networking Topics like,

* IP Address
* MAC Address
* Ethernet Frame Format
* Ether Types
* Network Protocol
* NIC
* Switches
* Routers
* DNS
* DHCP
* ARP
* TCP/IP
* UDP etc..,

2) We have to learn Dpdk User Guide and should have an idea on Libraries and Drivers, EAL etc.

3) We Should have to know about DPDK and its installation Procedure and to run basic Application like Helloworld in x86.

**Tasks we should know to work with DPDK :**

**I . Steps for DPDK Installation and to run the Basic Example Application like helloworld :**

(i) DPDK Download (choose whatever version which you want)

(ii) $tar xvf dpdk-<version>.tar.gz

(iii) $cd dpdk-stable-<version>

(iv) Add Hugepages and its size in the below path

$vi /etc/default/grub

GRUB\_CMDLINE\_LINUX="default\_hugepagesz=1GB hugepagesz=1G hugepages=1”

(v) $make config T=x86\_64-native-linux-gcc

Then build folder will be generated for the Target.

(vi) $cd build

(vii) $make

All the Libraries and drivers will be generated in the build folder which is useful for the Target to link all the Libraries.

**II. Steps to run the DPDK Application in User space:**

(i) $export RTE\_SDK=<dpdk-stable-version>

(ii) $export RTE\_TARGET=build

(iii) $make

It will scan and link all the libraries and dependencies from the build (Target) folder.

(x) $ sudo ./<executable>

* Parallelly, we have to learn about QDMA User(Product) Guide and its working and how do we apply and using it on DPDK.
* After Running a predefined Application like Hello world, we’ll get a brief idea about DPDK and its Functionality. So, next we have to create our own customized Application by using the DPDK Libraries and its Pre-defined functions.

1) Write a dpdk application which runs three lcores particularly on 3,4,5 cores and print core numbers by using flags.

2) Write a dpdk application which runs three lcores 3,4,5 particularly without using a flags in command line.

a)3rd core assign memory (using malloc) and enqueue to 4th core through ring.

b)4th core dequeue from the 3rd core and enqueue to 5th core.

c)5th dequeue from the 4th core and print the data.

3) Do question 2 with memory allocating from mempools

After doing the 2nd and 3rd questions by using DPDK, we’ll get a complete idea on how to use the preferred functions like Rings , Mbuf, mempool.

1. Know About skeleton Application and how to do binding and unbinding the driver and to know and what makes the use of using bbdev.
2. Before going to bbdev we should have a knowledge on Memory Management, Uplink and Downlink Channels.Then we can move onto bbdev Application and how to make use of it.

**Procedure for Customized Skeleton Application :**

* To build a Customized Skeleton Application, we should have to export the SDK and Target by giving the SDK path and Target Path.

Eg : export RTE\_SDK=<dpdk-path>

export RTE\_TARGET=build

* Next, we have to create a hugepages for our Application.Otherwise ,it gives error ( No hugepages were created).

$mkdir mkdir /mnt/huge/

$ sudo su

$ sudo mount -t hugetlbfs nodev /mnt/huge/

$ sudo echo 1024 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr\_hugepages

* T1-card should have to conect to PCIe slot and JTAG Connection port from T1-card should have to connect to our CPU.
* Open vivado to flash the bitstream file.
* By default a predefined bitstream file is present. Whenever our bitstream file is flashed for the Customized Skeleton Application, previous bitstream file is override. If not override to new bitstream file, try the reboot the pc and check PCIe device location is came or not by giving the below command.

$ lspci | grep Xilinx

(Eg : 46:00.0 : Ethernet Controller)

* If it not showing the status of the Ethernet Controller again, try to hardboot the Board. Then again check the status of the Ethernet Controller by giving the same above (lspci)command.
* Next, we have to modprobe the uio module.

$ sudo modprobe uio

* Next, insert the .ko file by giving the below command.

$ sudo insmod igb\_uio.ko (dpdk<version>/build/kmod)

* Next, we have to bind the igb\_uio module to PCIe device Location (46:00.0) by giving the below commnad.

$ sudo ./dpdk-devbind.py --force –b igb\_uio 46:00.0

* To check wheather igb\_uio is binded to the PCIe device location or not by using the below command.

$ sudo ./dpdk-devbind.py --status

* Finally we have to run our Binary or Executable file of our Customized Skeleton Application.
* If we want to unbind the PCIe device location, use the below command.

$ sudo ./dpdk-devbind.py -u --force <46:00.0>