OpenSSD Tutorial 2022

CRZ Technology

OpenSSD Introduction

OpenSSD Motivation

Need a SSD platform

- to develop a new firmware algorithm

- to explore hardware architecture and organization

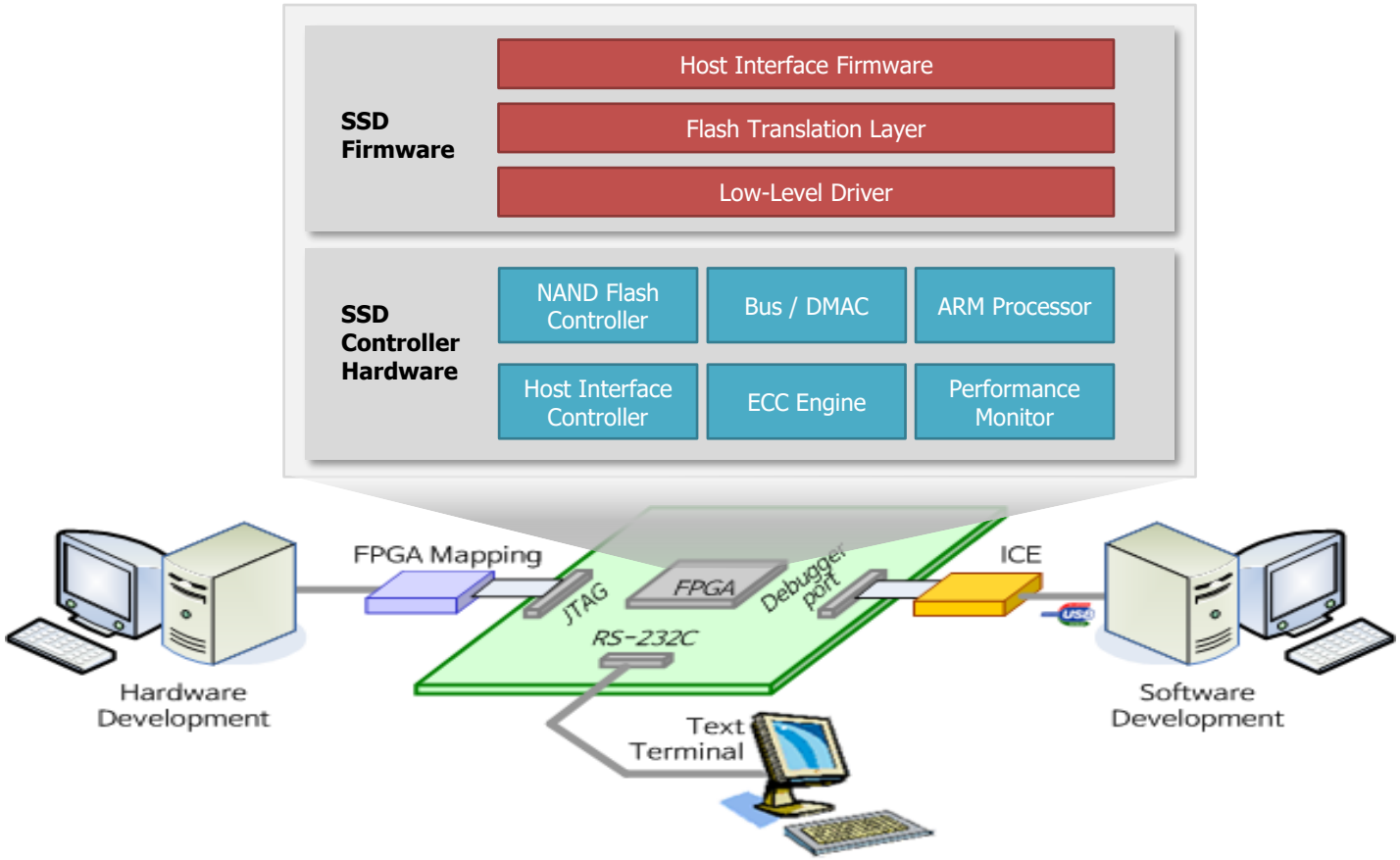
Use a commercial product as a platform?

- little information on HW/SW

- no way to change controller SoC

What's the OpenSSD Project?

Open source SSD design used for research and education



OpenSSD Project History

- **Open-source SSD platforms**
  - Jasmine OpenSSD (2011)
  - Cosmos OpenSSD (2014)
  - Cosmos+ OpenSSD (2016)
  - Daisy+ OpenSSD (2022)

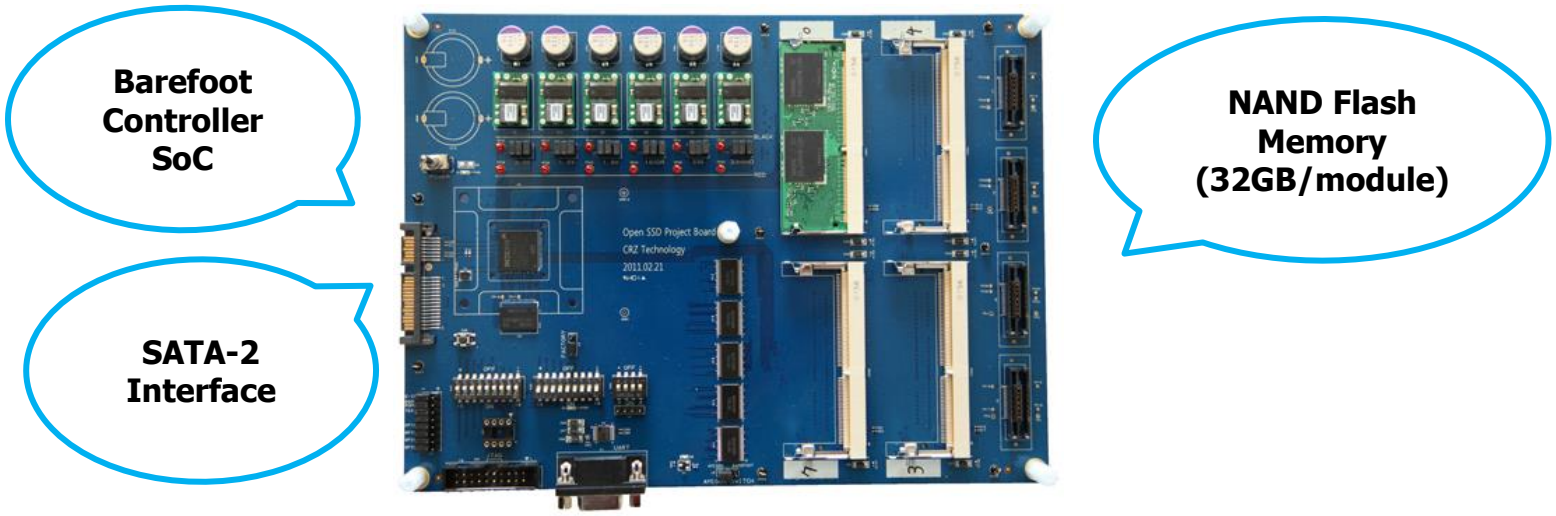
- **Daisy+ OpenSSD: FPGA-based platform**
  - Could modify SSD controller and firmware
  - Could add new hardware and software functionality

Why OpenSSD

- **Realistic research platform**
  - Solve your problem in a real system running host applications
  - Design your own SSD controller (hardware and firmware), if possible
- **Information exchange**
  - Share your solution with people in society
- **Community contribution**
  - Open your own solution to public
- **Expensive custom-made storage system**
  - Unique
- **Play for fun**

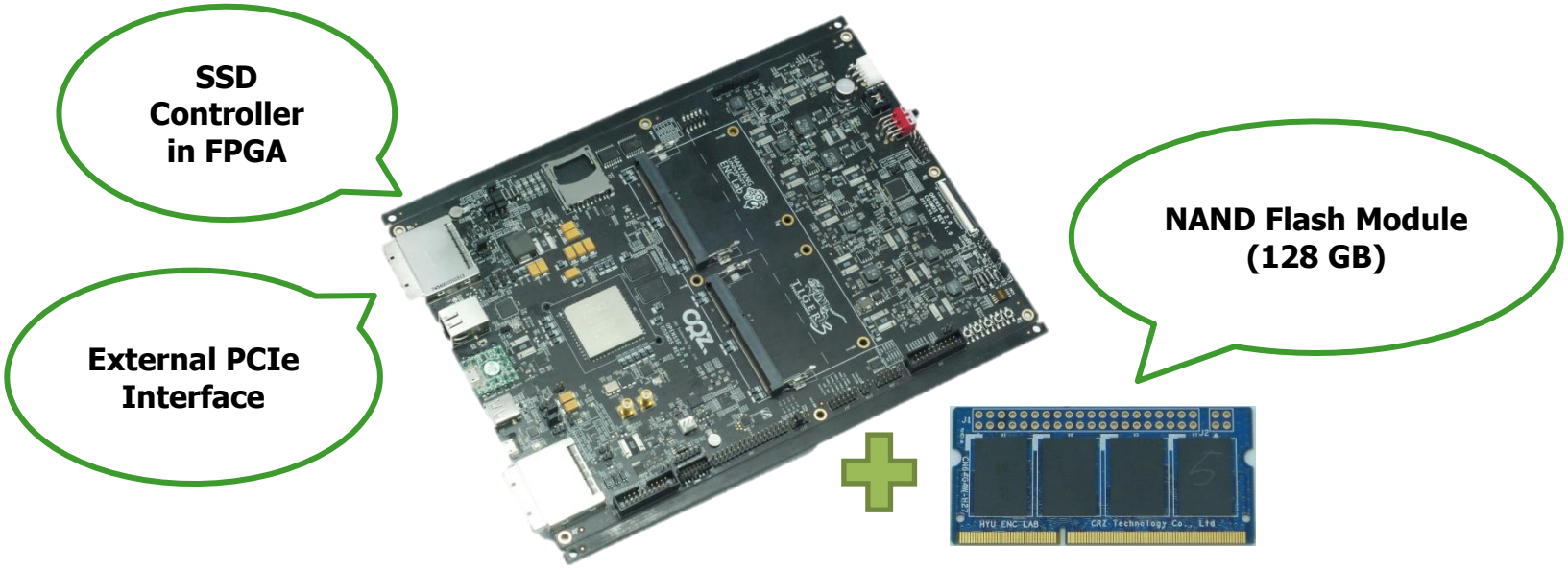
1st OpenSSD (Indilinx)

- **Jasmine OpenSSD (2011)**
  - SSD controller: Indilinx Barefoot (SoC w/SATA2)
  - Firmware: SKKU VLDB Lab
  - Users from 10+ countries



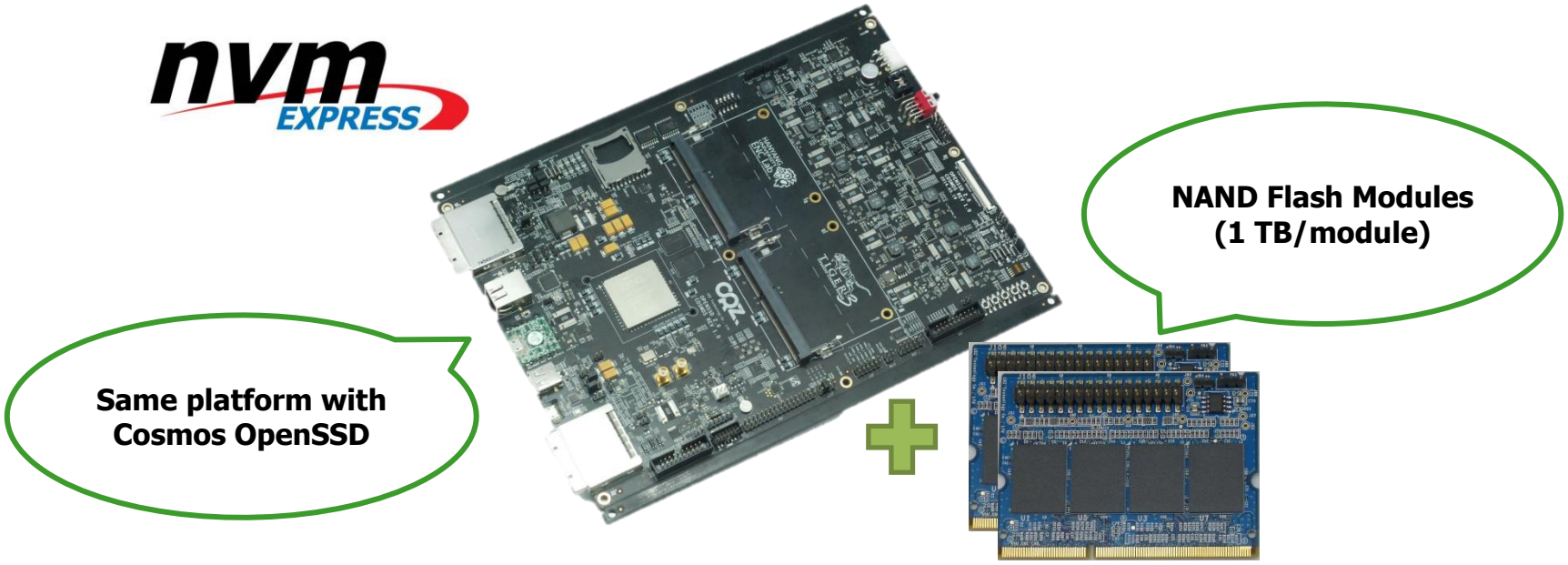
2nd OpenSSD (Hanyang University)

- **Cosmos OpenSSD (2014)**
  - SSD controller: HYU Tiger 3 (FPGA w/PCIe Gen2)
  - Firmware: HYU ENC Lab
  - Users from 5 countries (mostly in USA)



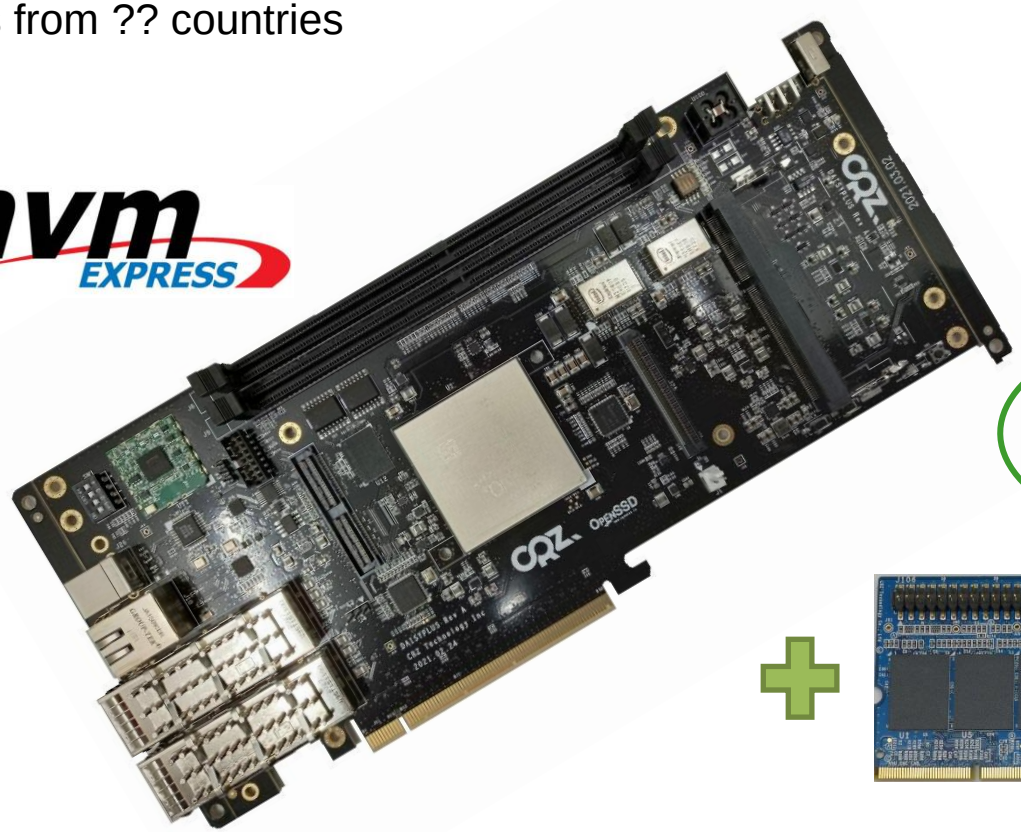
3rd OpenSSD (Hanyang University)

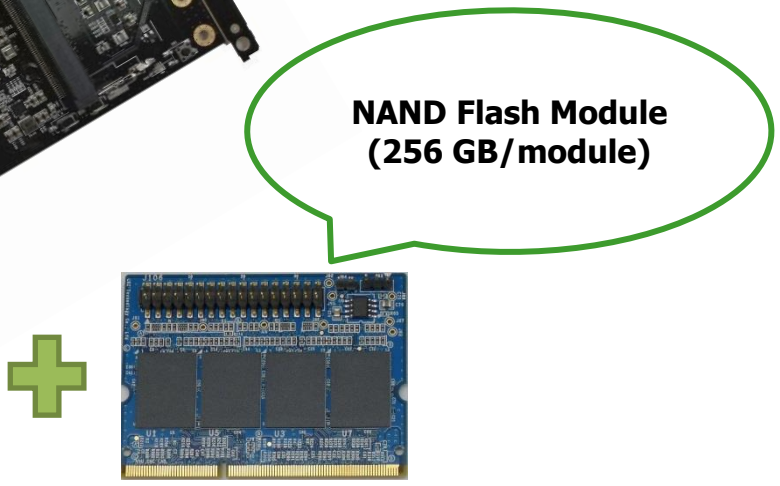
- **Cosmos+ OpenSSD (2016)**
  - SSD controller: HYU Tiger 4 (FPGA w/NVMe over PCIe Gen2)
  - Same main board with different memory modules
  - Firmware: HYU ENC Lab
  - Users from ?? countries



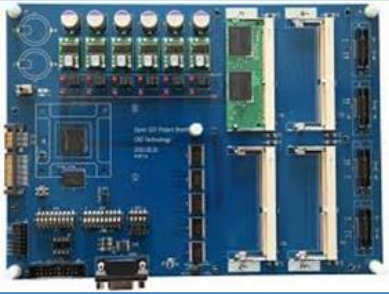
4th OpenSSD (CRZ Technology)

- **Daisy+ OpenSSD (2022)**
  - SSD controller: Daisy+ (FPGA w/NVMe over PCIe Gen3)
  - Firmware: CRZ Technology
  - Users from ?? countries





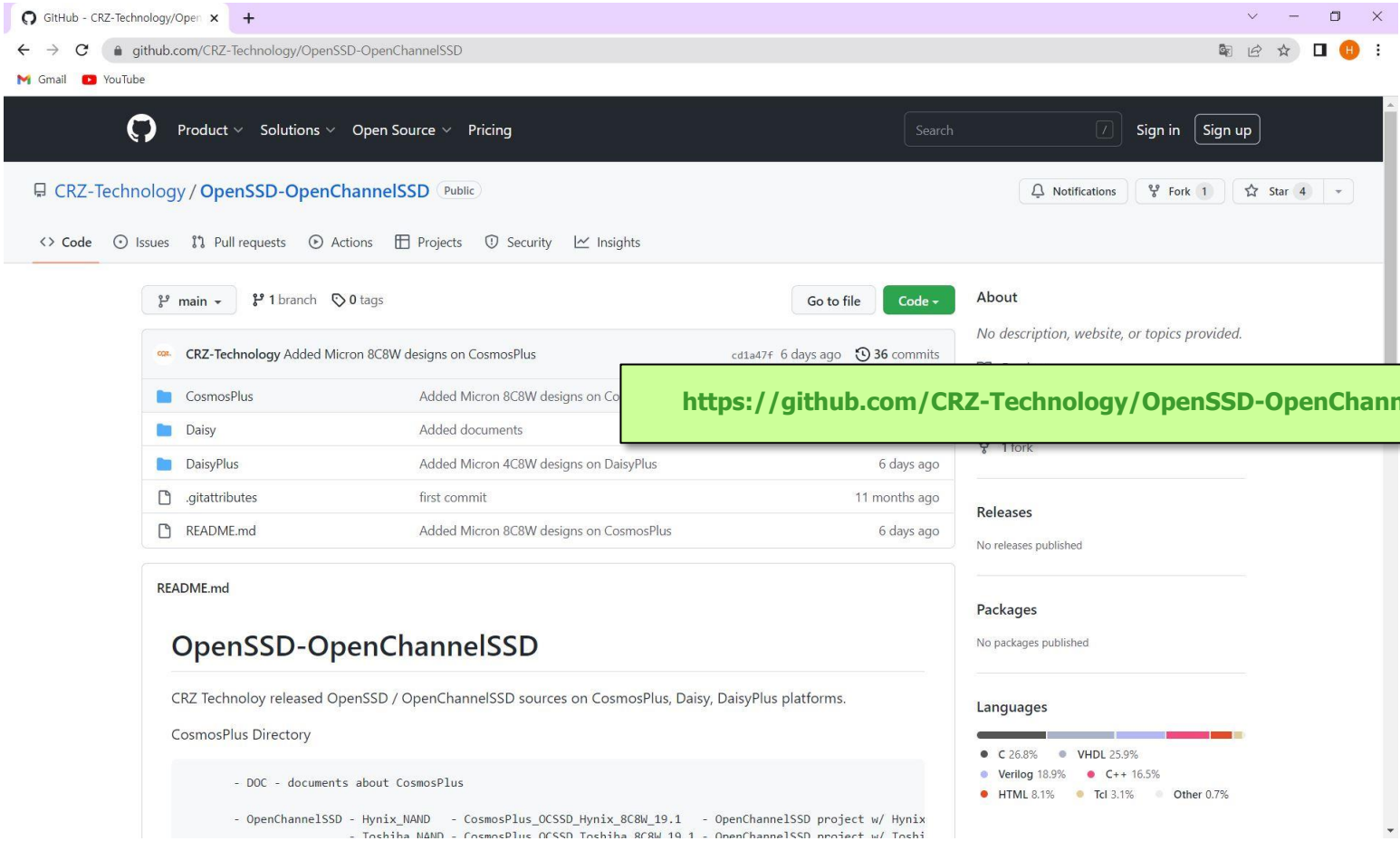
Platform Comparison

	Jasmine OpenSSD	Cosmos OpenSSD	Cosmos+ OpenSSD
Released in	2011	2014	2016
Main Board			
SSD Controller	Indilinx Barefoot (SoC)	HYU Tiger3 (FPGA)	HYU Tiger4 (FPGA)
Host Interface	SATA2	PCIe Gen2 4-lane (AHCI)	PCIe Gen2 8-lane (NVMe)
Maximum Capacity	128 GB (32 GB/module)	256 GB (128 GB/module)	2 TB (1 TB/module)
NAND Data Interface	SDR (Asynchronous)	NVDDR (Synchronous)	NVDDR2 (Toggle)
ECC Type and Strength	BCH, 16 bits/512 B	BCH, 32 bits/2 KB	BCH, 26 bits/512 B

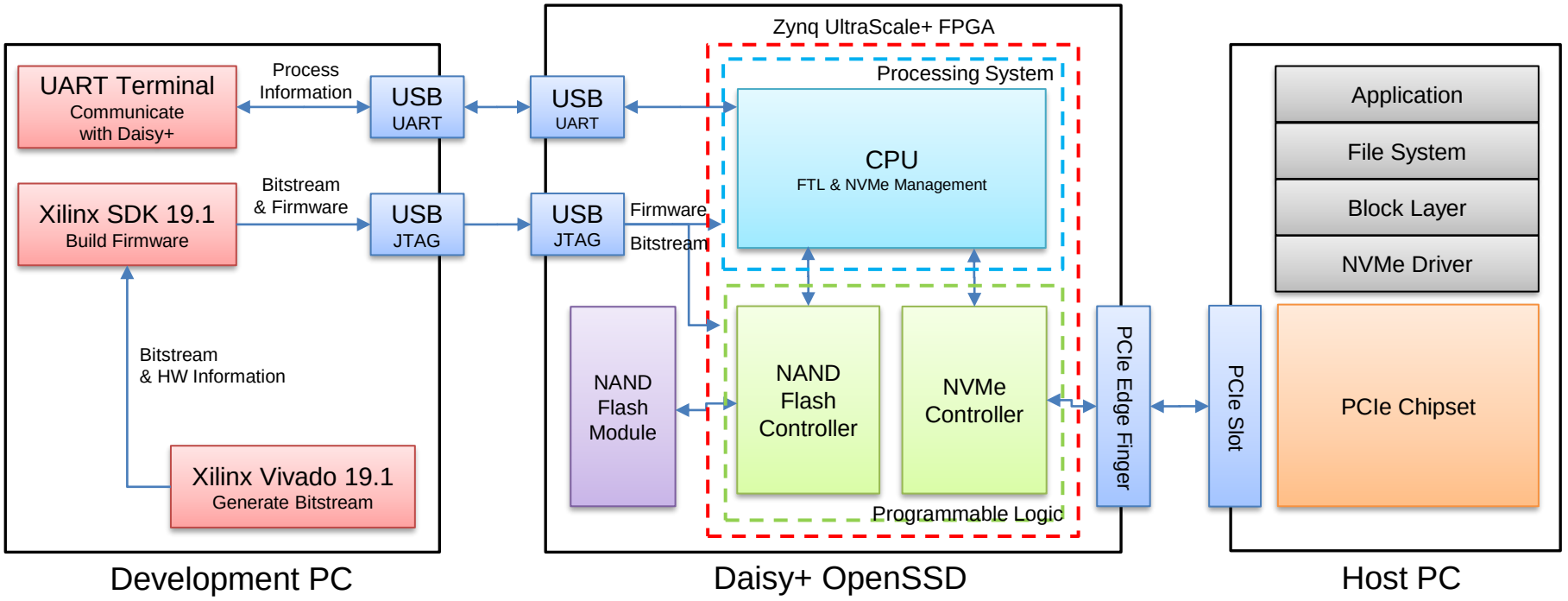
Platform Comparison

	Daisy+ OpenSSD
Released in	2022
Main Board	
SSD Controller	CRZ(FPGA)
Host Interface	PCIe Gen3 16-lane (NVMe)
Maximum Capacity	256 GB (256 GB/module)
NAND Data Interface	NVDDR2 (Toggle)
ECC Type and Strength	BCH, 26 bits/512 B

OpenSSD Git Hub

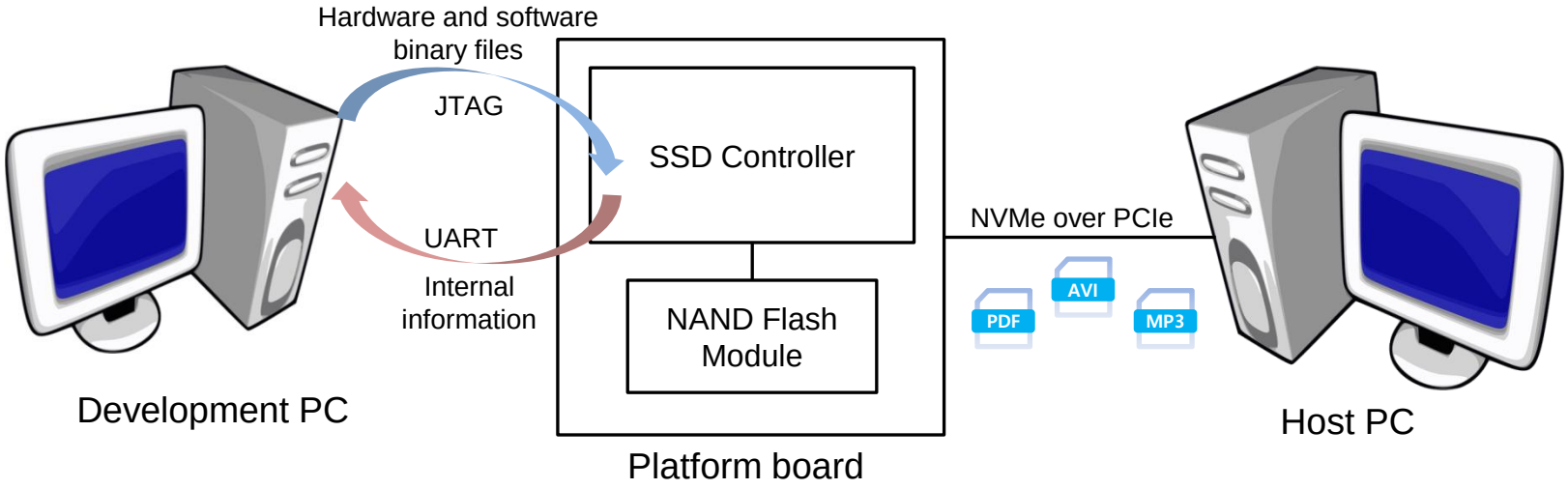


Daisy+ OpenSSD Overview



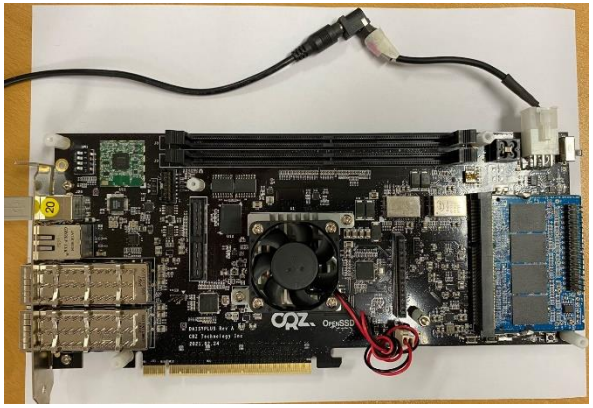
Daisy+ OpenSSD Environment

- **1 Development PC**
  - Downloading hardware/software design (JTAG)
  - Monitoring Daisy+ OpenSSD internals (UART)
- **1 Host PC**
  - Executing applications such as a benchmark (PCIe)
- **1 Platform board with 1 NAND flash module installed**
  - Working as a storage device to the host PC

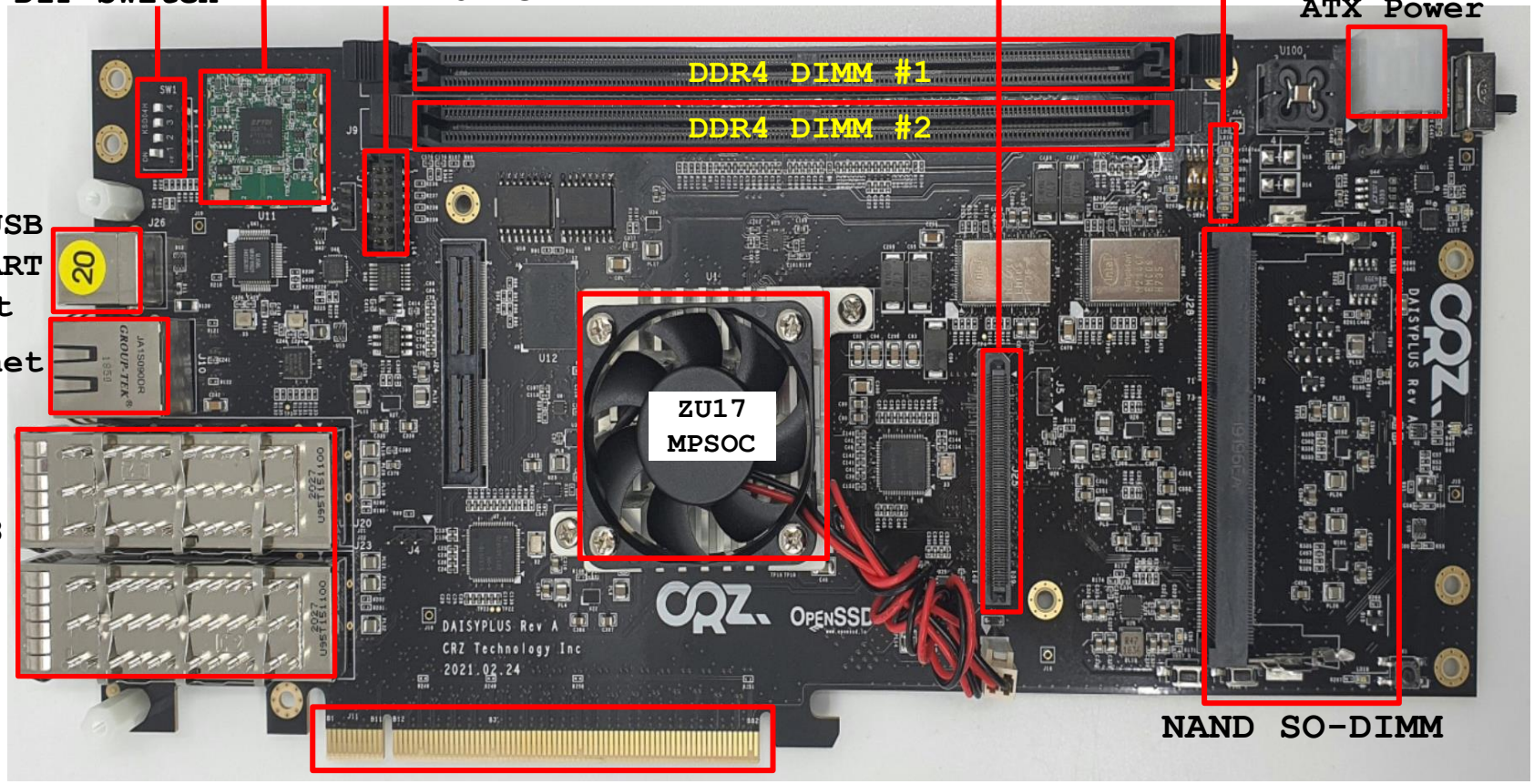


Hardware Components

- **Daisy+ OpenSSD platform board**
  - Consists of a UltraScale+ FPGA and other peripherals
- **NAND flash module**
  - Configured as multi-channel and multi-way flash array
  - Inserted into Daisy+ OpenSSD platform board
- **PCIe Edge Finger**
  - Connected with host PC through PCIe slot
- **USB cable for JTAG and UART**
  - Connected with development PC
- **Power cable and adapter**
  - 12V supply voltage



Daisy+ OpenSSD Platform Board

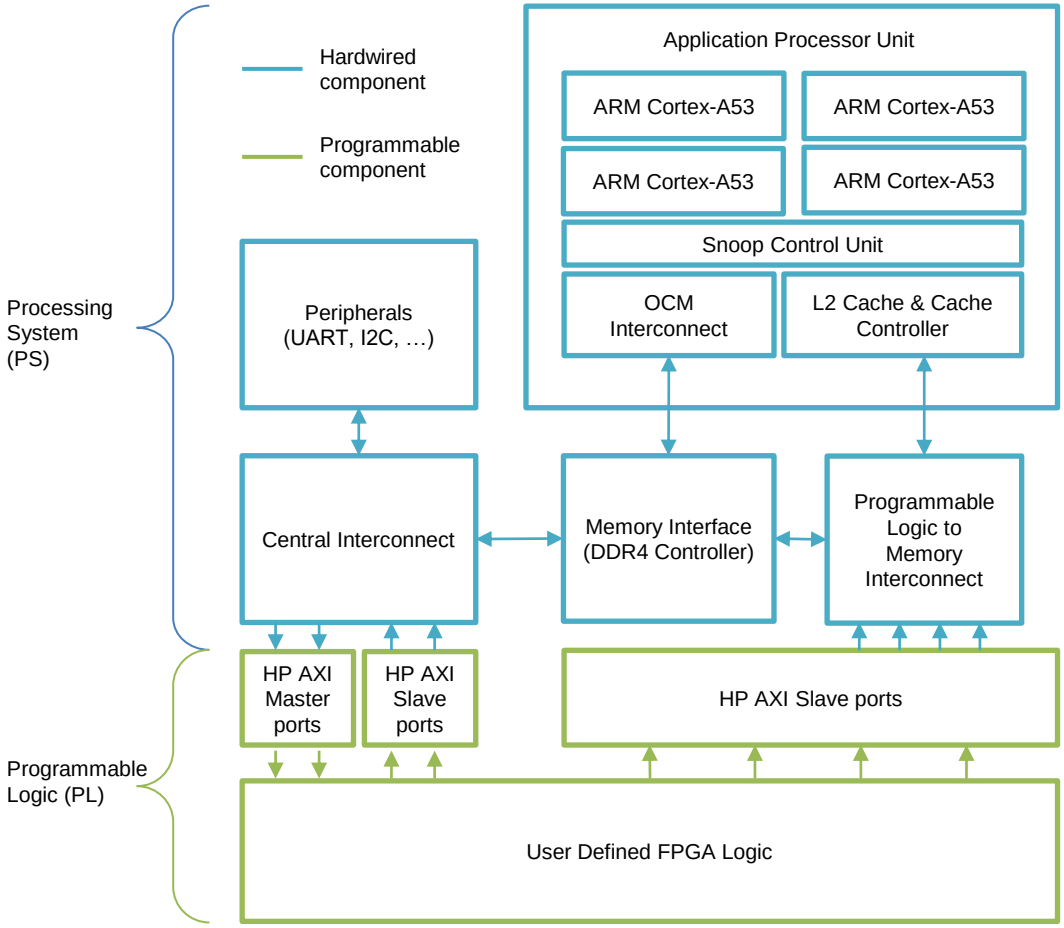


Primary Details

FPGA		Xilinx Zynq UltraScale+ ZU17EG FPGA (XCZU17EG-2FFVC1760-E)
Logic cells		926K (~ 5.2M ASIC gates)
CPU	**Type**	Quad-Core ARM Cortex™- A53 / Dual-Core ARM Cortex™- R5
	Clock frequency	Up to 1.5GHz(A53) / Up to 600MHz(R5)
Storage	**Total capacity**	Up to 256 GB
	Organization	Up to 4-channel 8-way
DRAM	**Device interface**	LP-DDR4 2400
	Total capacity	2 GB
Bus	**System**	AXI-Lite (bus width: 32 bits)
	Storage data	AXI (bus width: 64 bits, burst length: 16)
SRAM		256 KB (FPGA internal)

Zynq-UltraScale+ FPGA Architecture

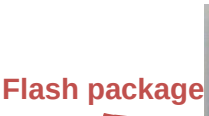
- **Xilinx's embedded SoC**
- **Two regions**
  - Processing System (PS)
    - Hardwired components
    - Executes the firmware program
  - Programmable Logic (PL)
    - Programmable components (FPGA region)
    - NAND flash controller (NFC) and NVMe controller reside in PL
- **Benefits of Using Zynq UltraScale+**
  - CPU is more faster than soft core (such as MicroBlaze)
  - No need to worry about organizing hardware memory controller, and some other peripherals (such as UART)
  - Xilinx supports BSP (Board Support Package)

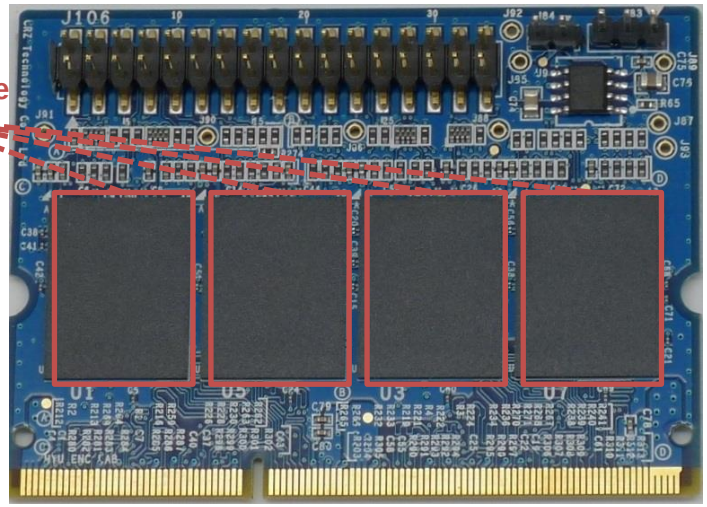


Zynq-UltraScale+ architecture overview

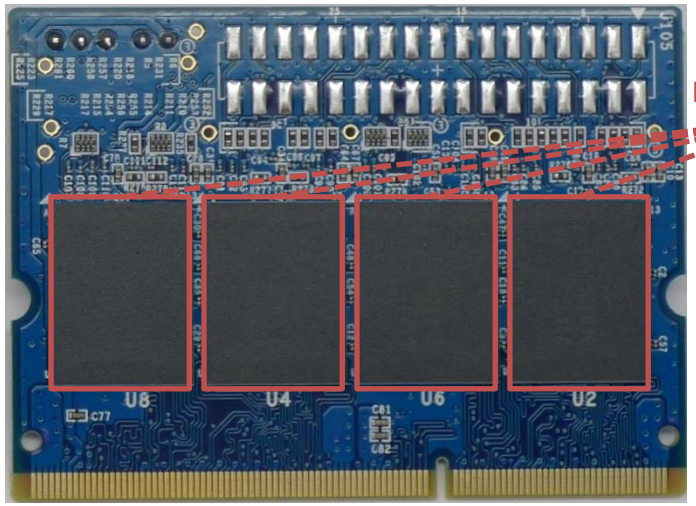
Daisy+ OpenSSD NAND Module

- **Each module has 8 flash packages**
  - One flash package
    - Capacity: 32 GB
    - Page size: 18048 Bytes (spare area: 1664 Bytes)
  - Toggle NAND
- **Used with CRZ Controller**





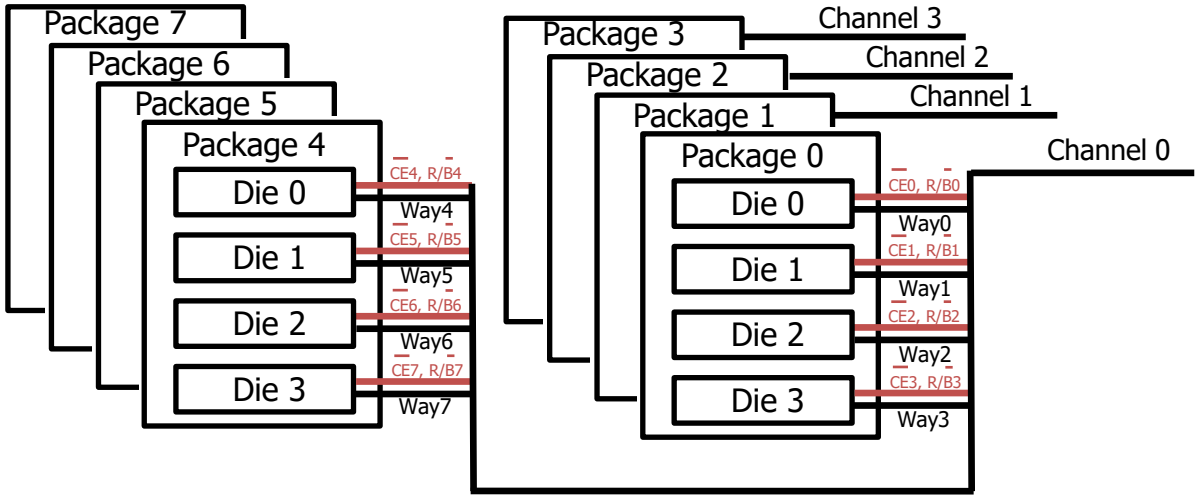
Front side



Rear side

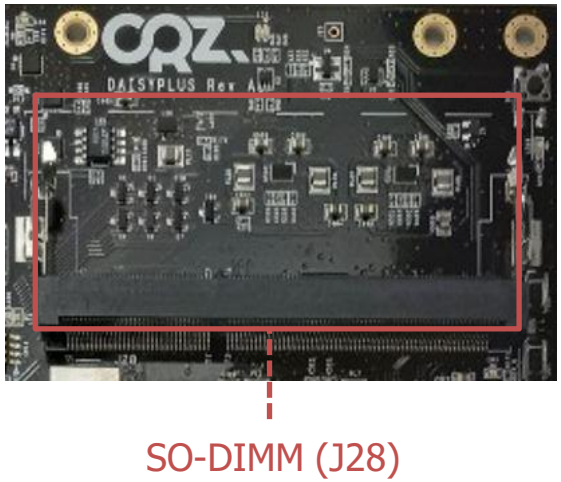
Logical Organization of NAND Flash Module

- **Module configuration**
  - 4-channels/module and 8-ways/channel
- **Shared signals within a channel (a package)**
  - Dies in the same package share the I/O channel
  - Dies in the same package share command signals except Chip Enable (CE)
  - Each die has own Ready/Busy (R/B) signal



NAND Module Setup

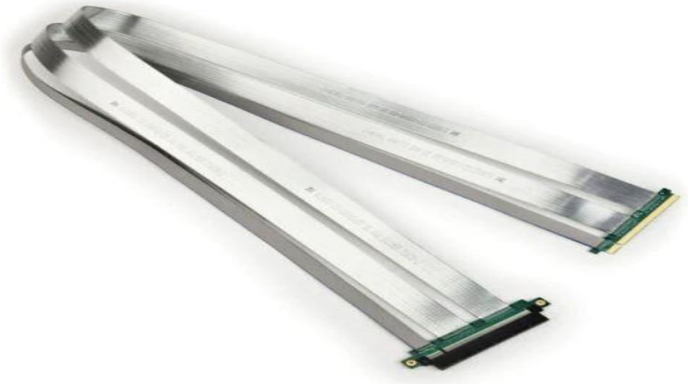
- **Daisy+ OpenSSD**
  - Supports one flash module slot (J28)



- **Caution**
  - Daisy+ OpenSSD flash module slots have custom pin maps
  - You should not insert any SDRAM module into this slot

PCIe Extender

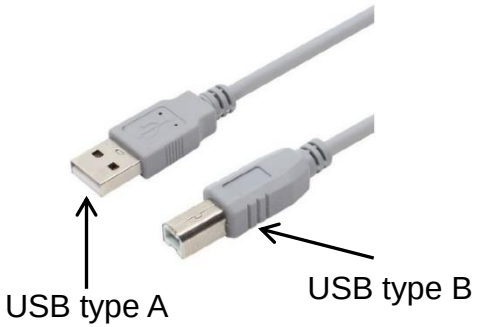
- **Expand PCIe Slot of host PC to connect external device**
  - 3M™ Twin Axial PCI Express X16 Extender Assemblies Gen 3.0
  - 8KC3-0726-0250
- **External PCIe connector (16-lane) to PCIe Edge Finger**
  - 2.5 GT/s for a Gen1, 5.0 GT/s for a Gen2, 8.0 GT/s for a Gen3
  - Connected with high data rate serial transceiver in FPGA



PCIe Extender

Connection with Development PC

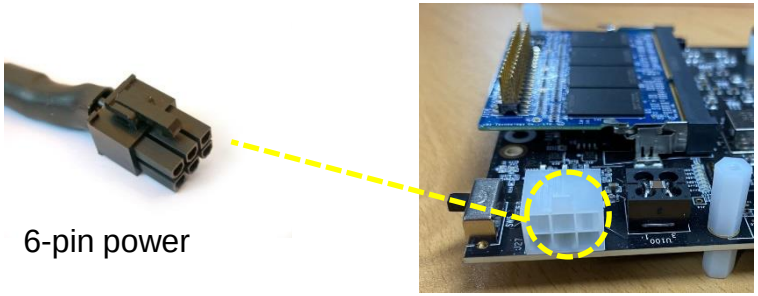
- **USB cable**
  - Used for downloading hardware and software binary files
  - Used for monitoring internal processes of Daisy+ OpenSSD
  - USB type A to USB type B cable



USB cable

Power Connection

- **Single-source of power to the platform board**
  - 6-pin power connector (J27)



- **The 6-pin connector looks similar to the regular PC 6-pin PCIe connector**

  Note: Difference in pin assignment between two connectors

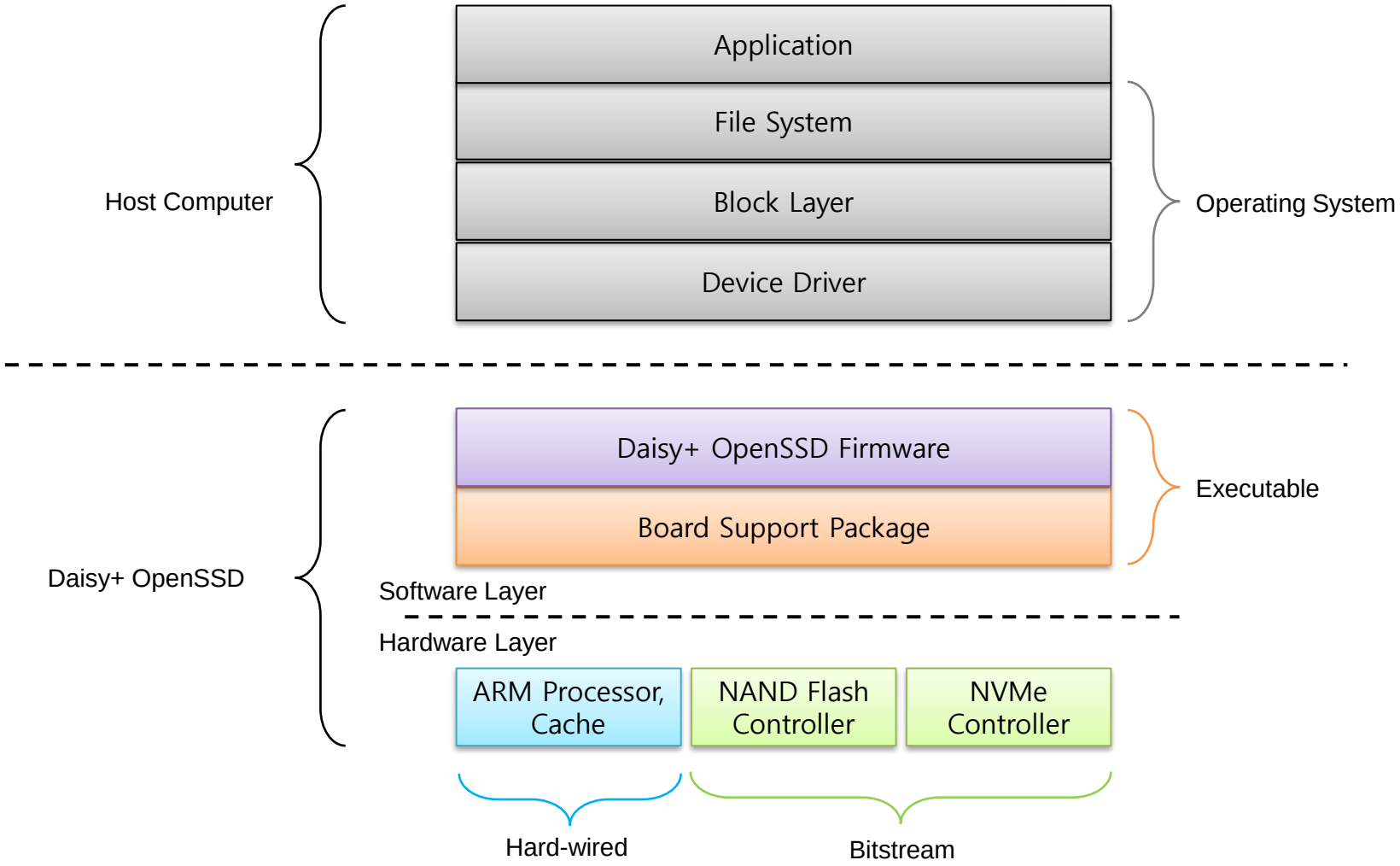
Connector	Pin map					
	1	2	3	4	5	6
Platform board 6-pin power	12V	NC	12V	GND	NC	GND
PC 6-pin PCIe power	GND	GND	GND	12V	12V	12V

- **Caution**
  - Do not plug PC 6-pin PCIe power cable in platform board 6-pin power connector (J27)

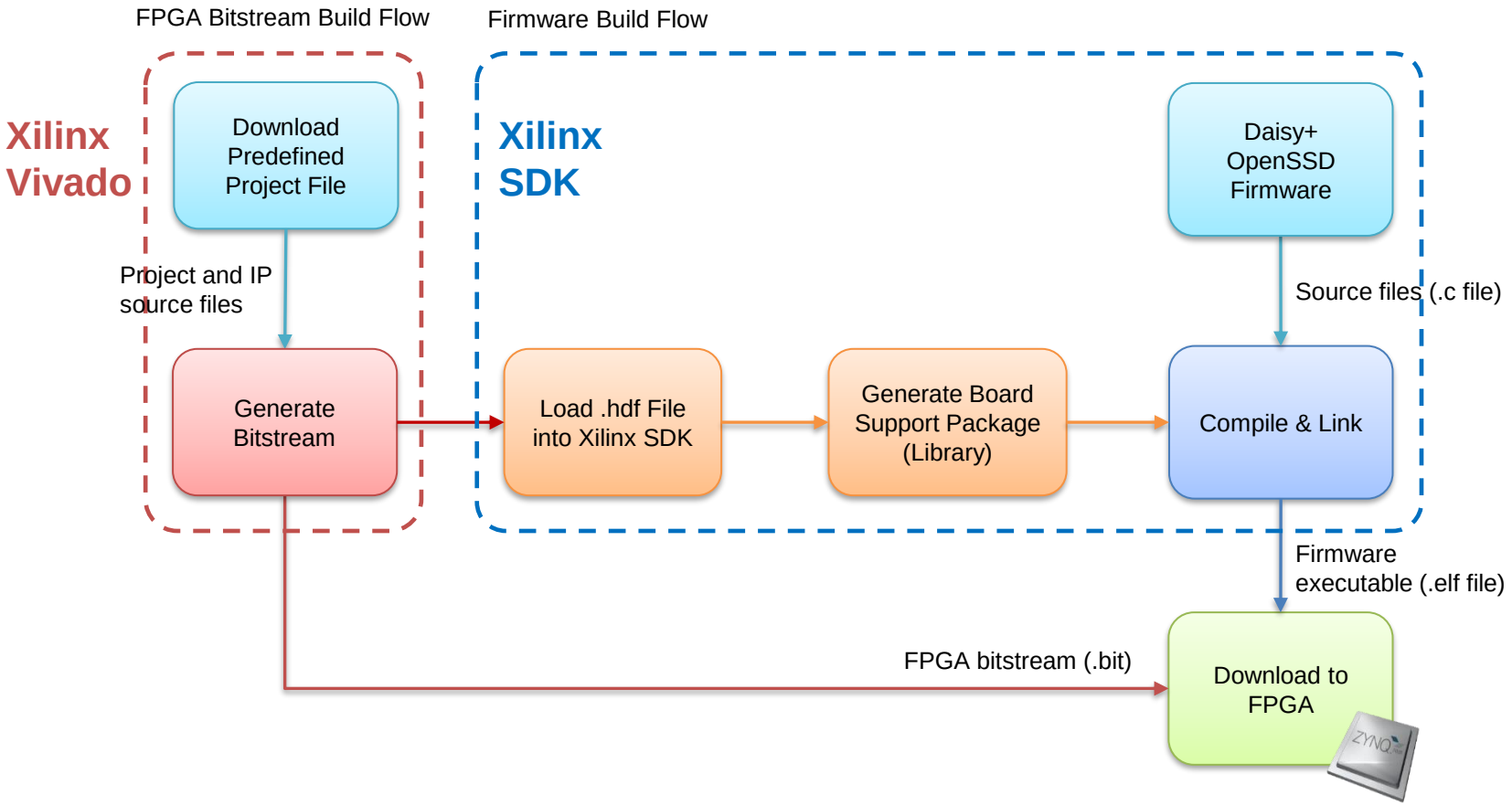
Development Software Components

- **Xilinx Vivado**
  - Generates a FPGA bitstream
  - Exports the generated FPGA bitstream to Xilinx SDK
- **Xilinx SDK**
  - Builds a SSD controller firmware
  - Downloads a FPGA bitstream and a firmware to the Zynq UltraScale+ FPGA
- **FPGA bitstream**
  - Used to configure the programmable logic side of Zynq UltraScale+ FPGA
- **Firmware**
  - Manages the NAND flash array
  - Handles NVMe commands

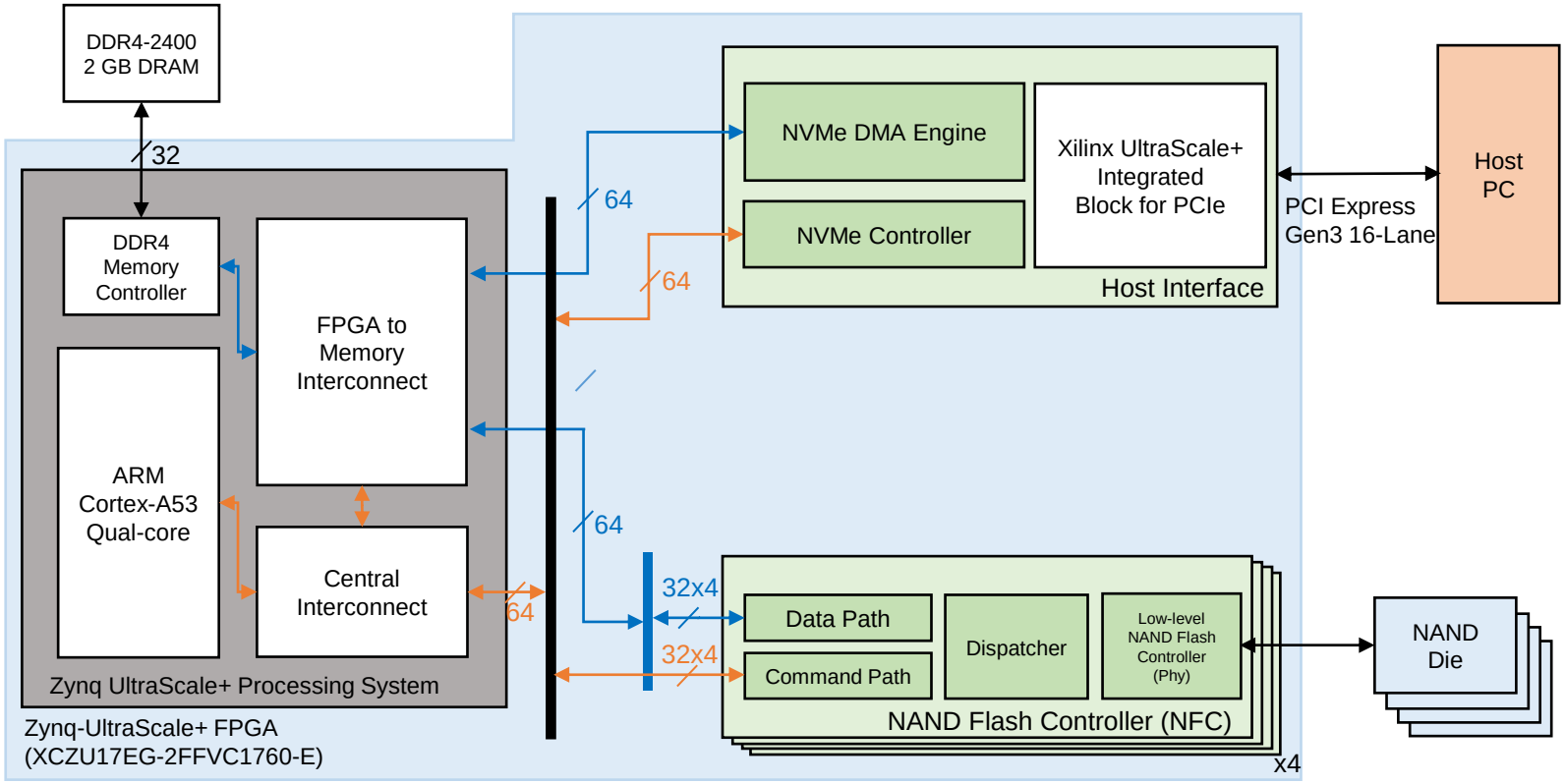
Host System Software Architecture



Software Porting Flow

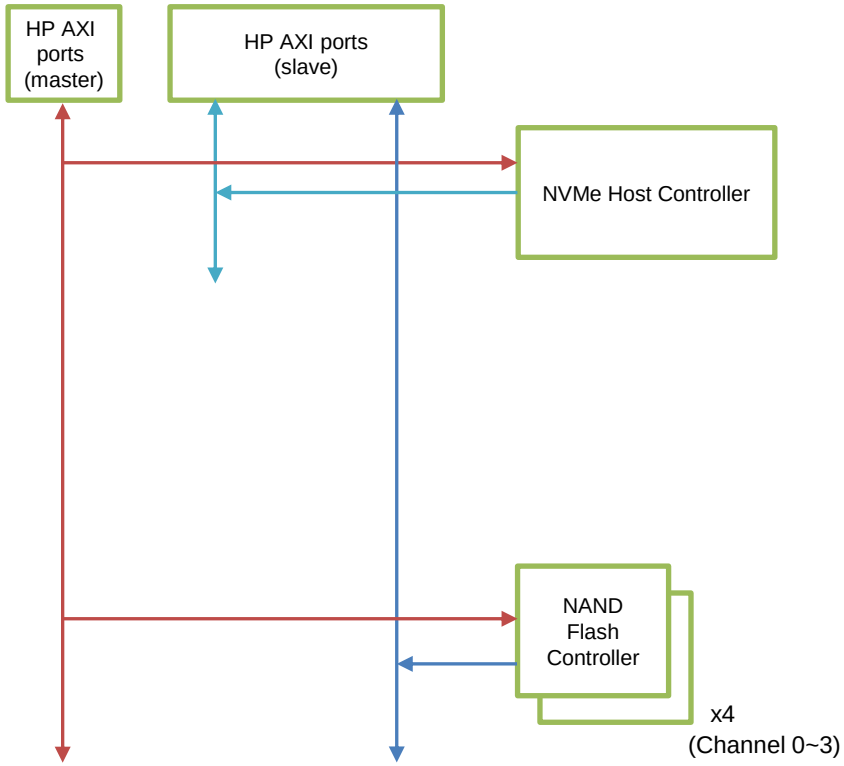


Daisy+ OpenSSD Internal System Overview

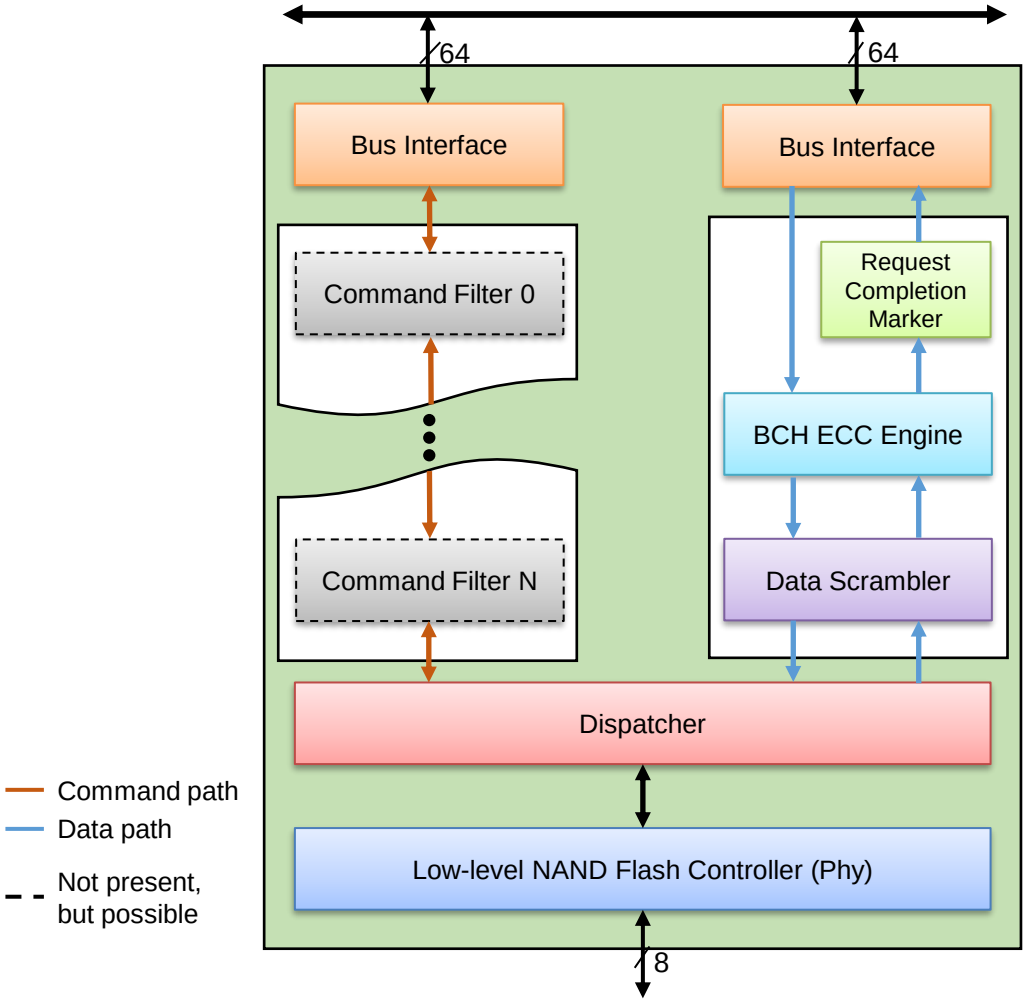


System Bus Structure

- **High Performance (HP) AXI4 bus**
  - 64bits interface
  - Used for control
  - Operates @ 200MHz
- **High Performance (HP) AXI4 bus**
  - 64bits interface
  - Used for Direct Memory Access (DMA)
  - Operates @ 250 MHz

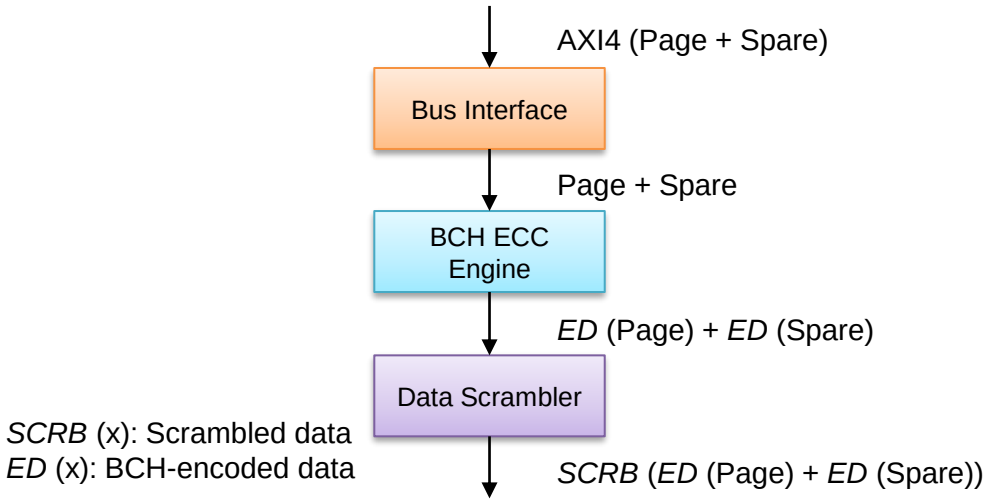


NAND Flash Controller Overview



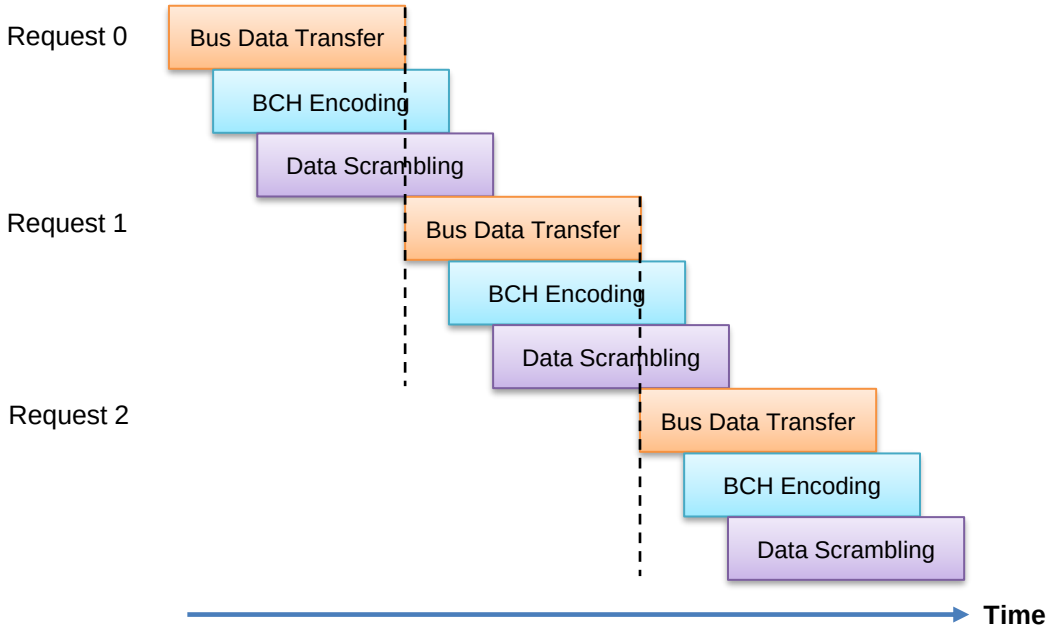
Layered and Modular NFC Design

- **Commands and data streams are encapsulated or decapsulated throughout modules in a layer**
- **Users can insert or remove modules more easily**



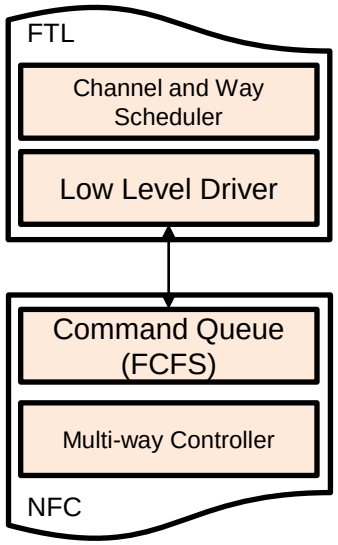
Pipelined NFC Operation

- **Data transfers throughout a layer from DRAM to NAND flash or from NAND flash to DRAM are all pipelined**
- **Page buffer is not required in channel controller**



Software-controlled NAND Flash Scheduler

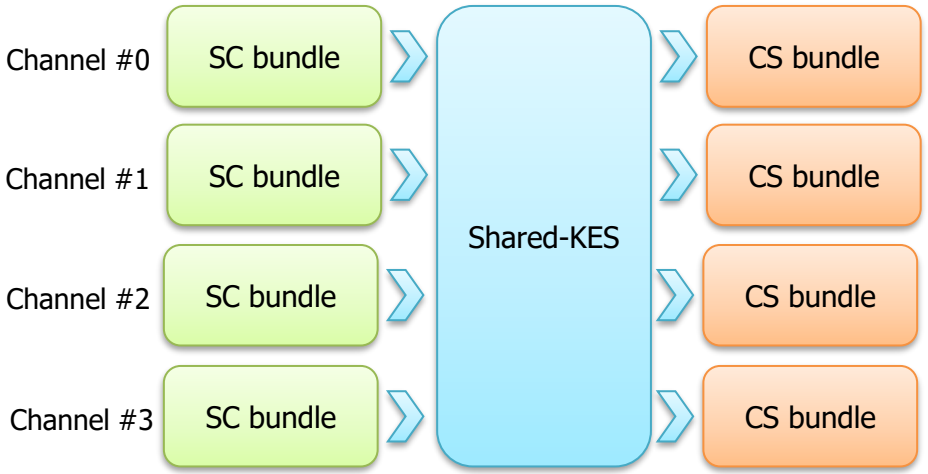
- **FTL is responsible for channel and way scheduling**
- **This enables flexible scheduling policy**



NFC in Daisy+ OpenSSD

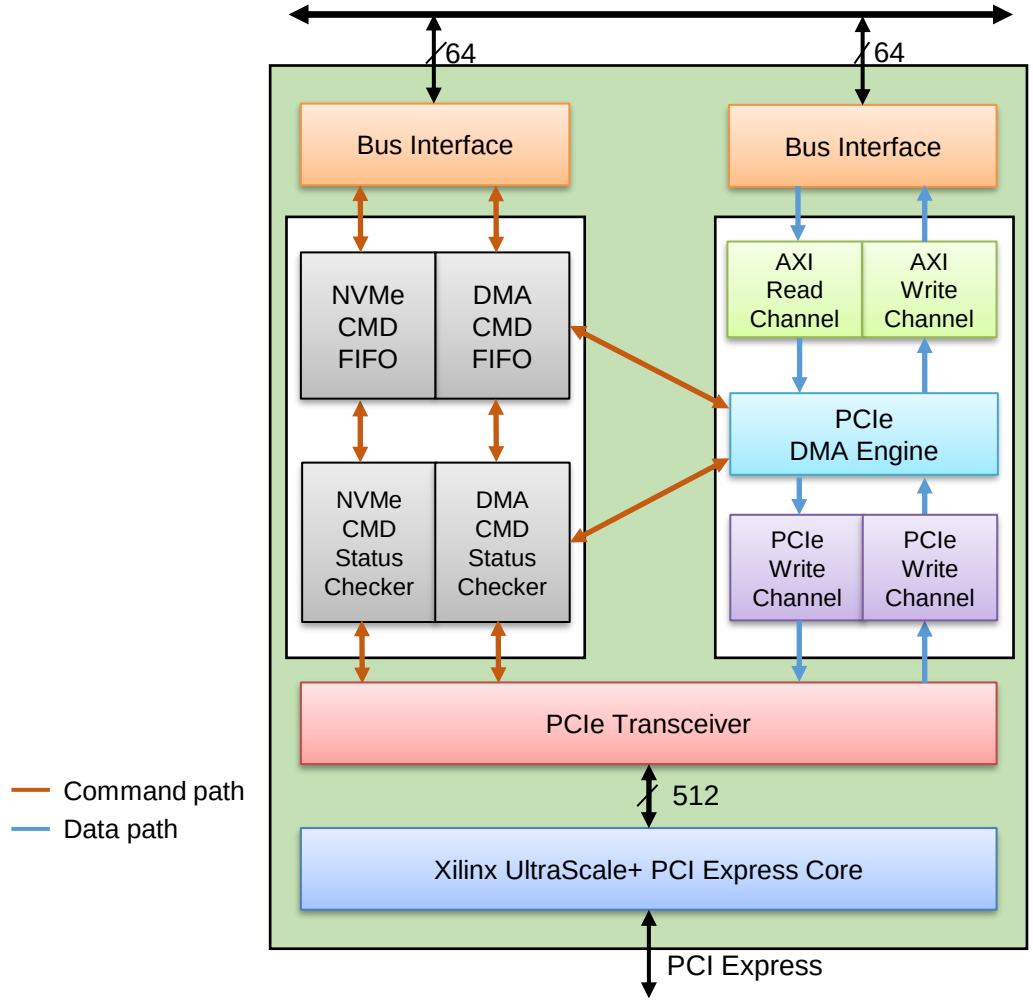
BCH ECC Engine with Shared-KES

- **Key equation solver (KES) used more (≥50 %) of logic cells than syndrome calculator and chien searcher**
- **Shared-KES saves 40 % of logic cells used in a BCH ECC decoder**
- **Short BCH code parallelization is applied for high utilization of hardware resources**



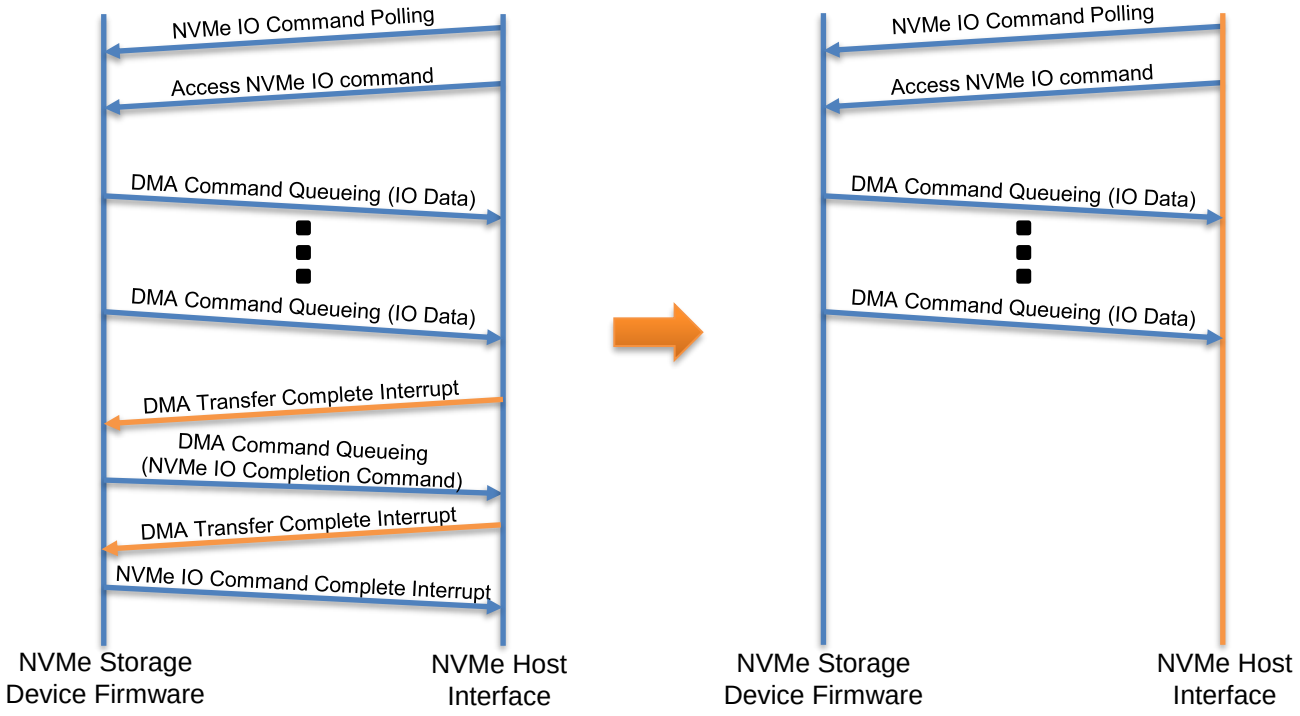
* (256 B x 2) bundle, (13 x 2) bit error correction, (8 bit x 2) parallel level

NVMe Host Interface Overview



Automated NVMe IO Command Completion

- The NVMe host interface completes NVMe IO commands automatically
- The FTL does not need to be involved in the completion process



NVMe Host Interface Specification

- **NVMe specification 1.1/1.2 compliant**
  - Up to 8 IO submission/completion queues - 256 entries each
  - 512B and 4KB sector size
  - Physical region page (PRP) data transfer mechanism
  - Native device driver for Windows 8/8.1 and Linux kernel>=4.18
  - OpenFabrics Alliance (OFA) NVMe driver for Windows 7 and later

NVMe Interface Performance (DRAM Disk)

Workload	Read	Write
Random 4KB	300K IOPS	300K IOPS
128KB	1.7 GB/s	1.7 GB/s

Firmware FTL Features

LRU data buffer management

- Data transfer between host system and NAND flash memory via data buffer
- Eviction of LRU buffer entry

Pure page-level mapping (16 KB page)

- Static mapping
- Channel/way interleaving

Greedy garbage collection

- On-demand garbage collection
- Greedy selection of GC victims

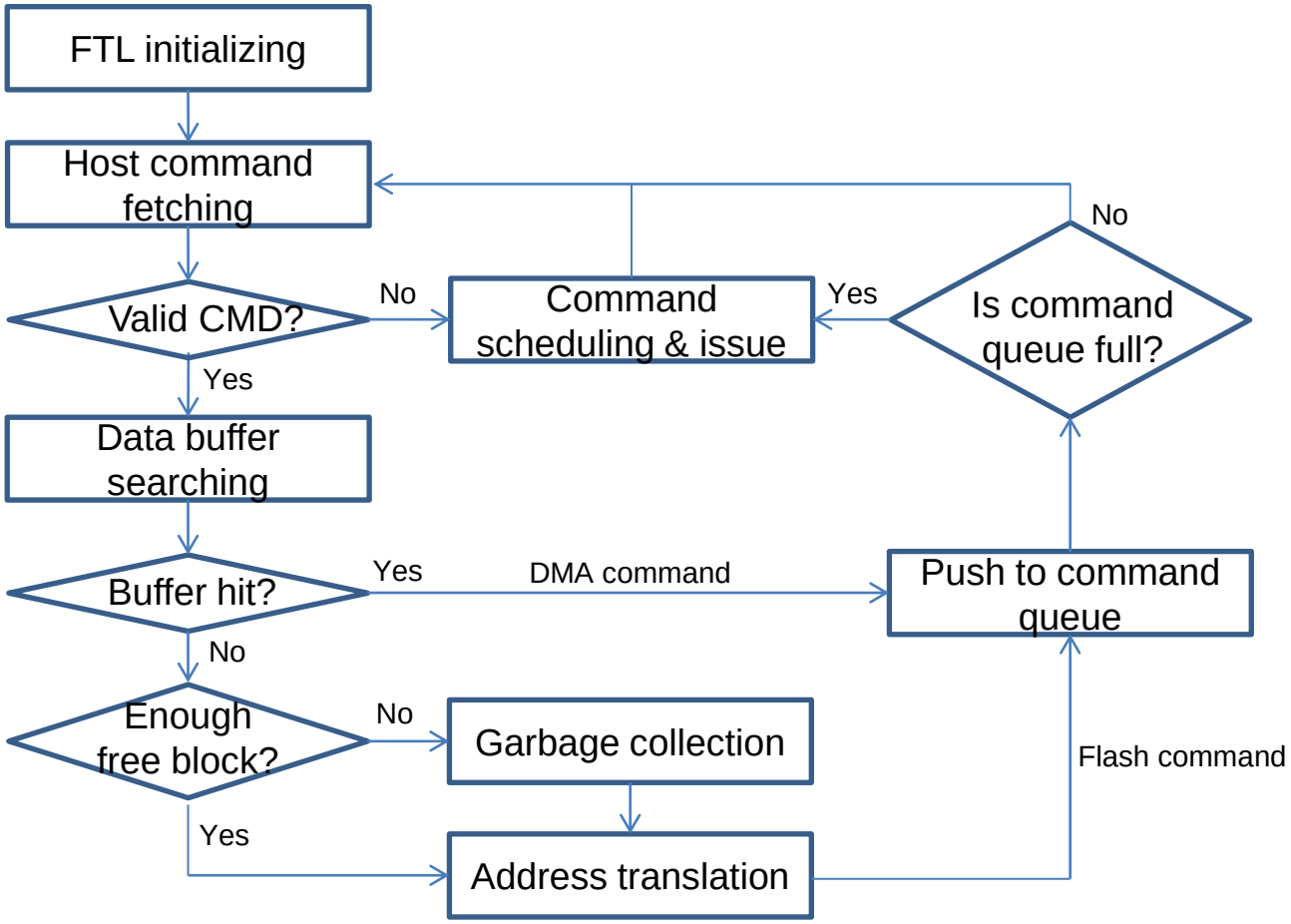
Command Set

- Single plane flash commands
- DMA commands for data transfer between host system and SSD

Priority-based scheduling

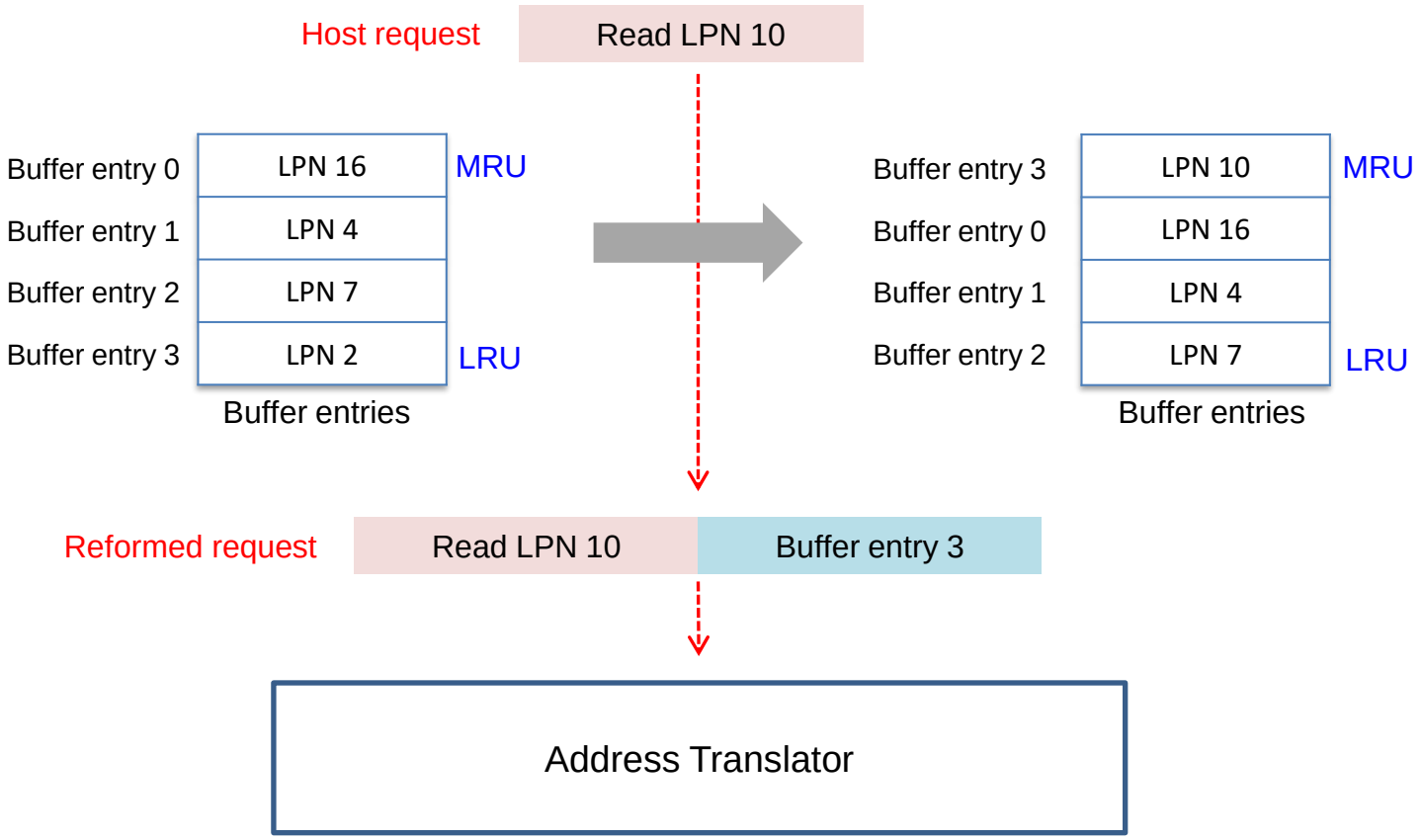
- Predetermined priority between DMA commands and flash commands
- Out of order execution between commands accessing different flash dies

Firmware Overall Sequence



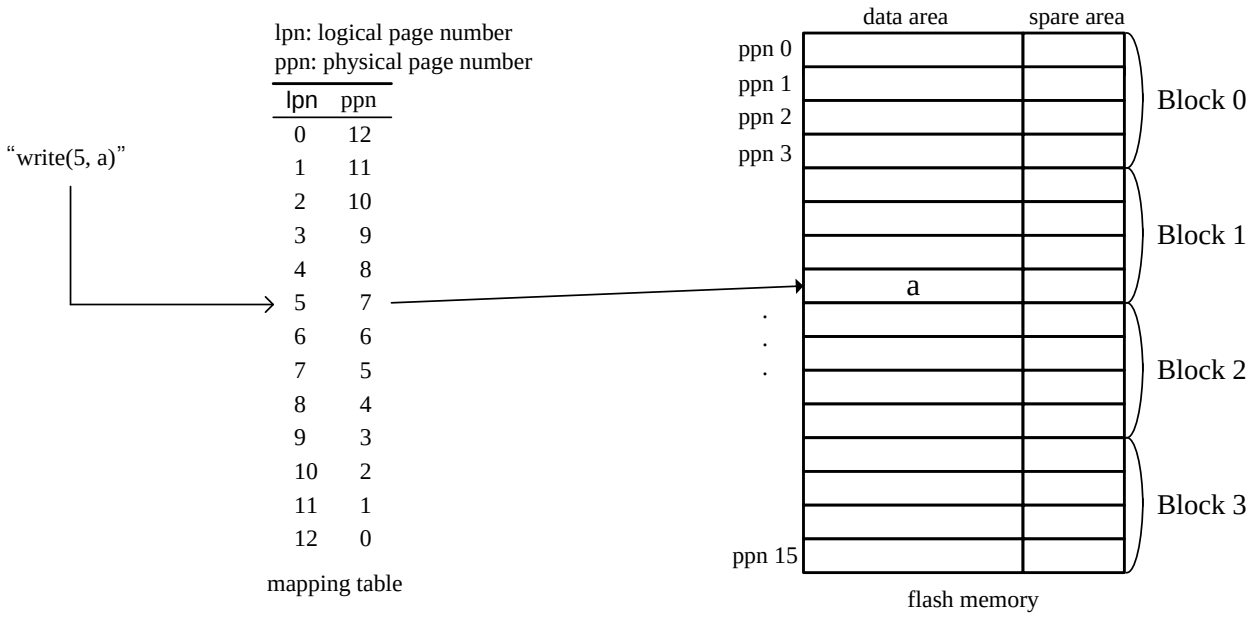
LRU Data Buffer Management

- **Buffer entry eviction**
  - LRU buffer entry is evicted to allocate a buffer entry for a new request



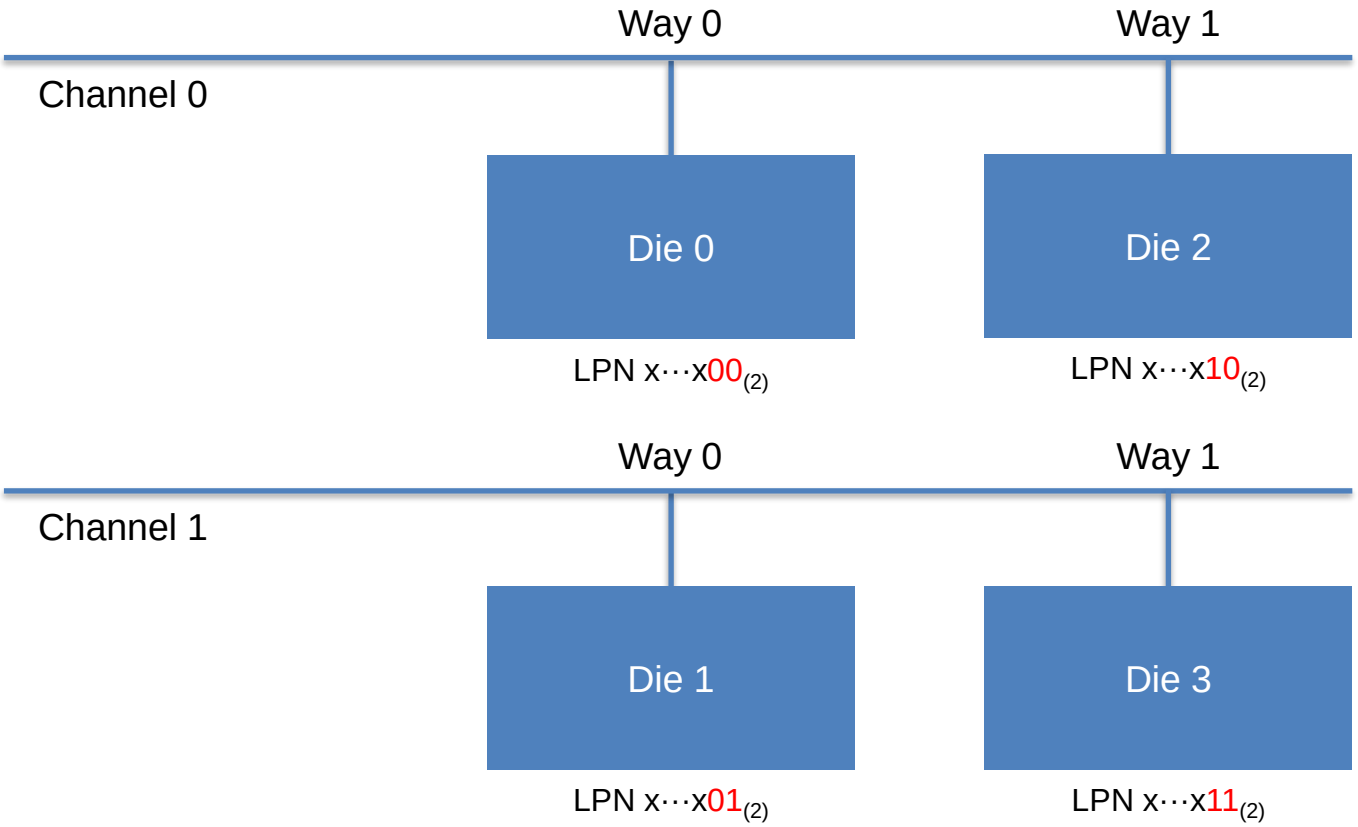
Page-level Mapping

- **Main Idea**
  - Every logical page is mapped to a corresponding physical page
- **Advantage**
  - Better performance over random write than block-level mapping
- **Disadvantage**
  - Huge amount of memory space requirement for the mapping table



Static Mapping

- **Mapping tables are managed within a die**
  - Simple channel/way interleaving for sequential logical access



Each LPN is deterministically mapped to specific die (ex. 2-channel, 2-way)

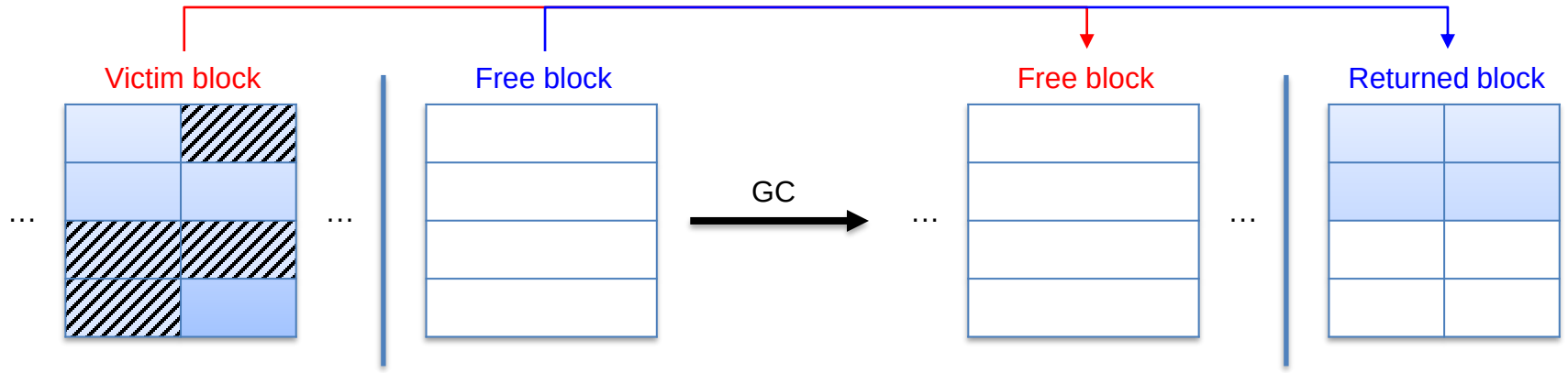
LPN: Logical Page Number

Concept of Garbage Collection

- **Why is garbage collection needed**
  - To reclaim new free blocks for future write requests
    - Invalid data occupy storage space before GC
- **What is garbage collection**
  - Copies the valid data into a new free block and erases the original invalid data
  - Basic operations involved in GC are the following
    - 1. The victim blocks meeting the conditions are selected for erasure
    - 2. The valid physical pages are copied into a free block
    - 3. The selected physical blocks are erased
- **What is important in GC**
  - Victim block selection
    - GC time depends on the status of victim block

On-demand GC

- **GC Trigger**
  - Each GC is triggered independently of other dies
  - GC is triggered when there is no free user block of each die
- **Blocks in GC**
  - One block per die is overprovisioned
  - Single victim block is a target of GC



Valid pages in victim block are copied to free block and the role of two blocks are swapped

Source file: pagemap.c

Firmware Command Set

Commands for NVMe DMA engine

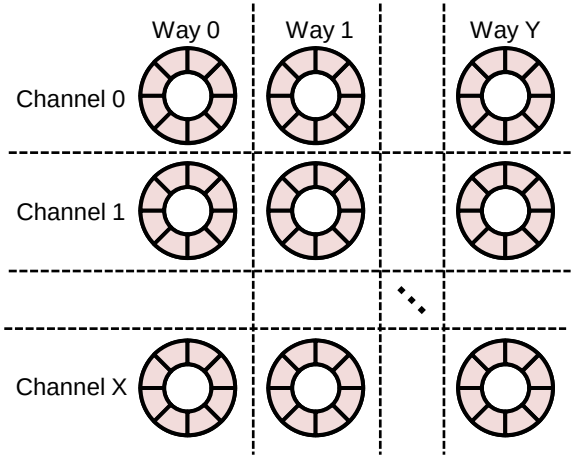
LLSCommand_RxDMA
▶ Transfer data from host system to data buffer
LLSCommand_TxDMA
▶ Transfer data from data buffer to host system

Commands for NAND flash controller

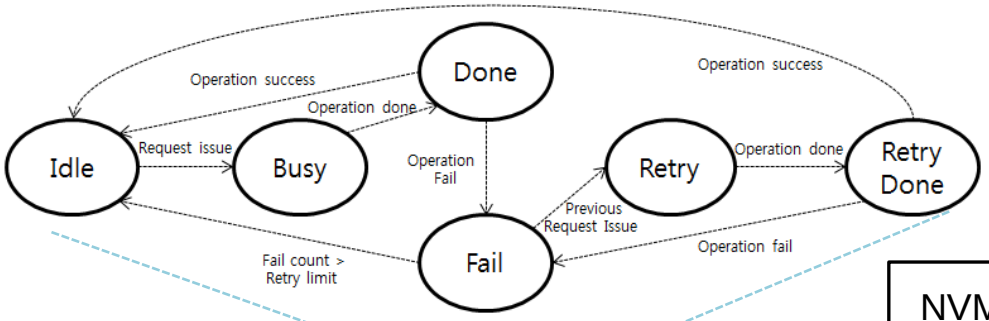
V2FCommand_ReadPageTrigger
▶ Read data of a flash page ▶ Store data to register of the flash die
V2FCommand_ReadPageTransfer
▶ Transfer data from a flash die to data buffer ▶ Inform bit error information to FTL
V2FCommand_ProgramPage
▶ Transfer data from data buffer to a flash die ▶ Program data to a flash page
V2FCommand_BlockErase
▶ Erase a flash block
V2FCommand_StatusCheck
▶ Check a previous command execution result

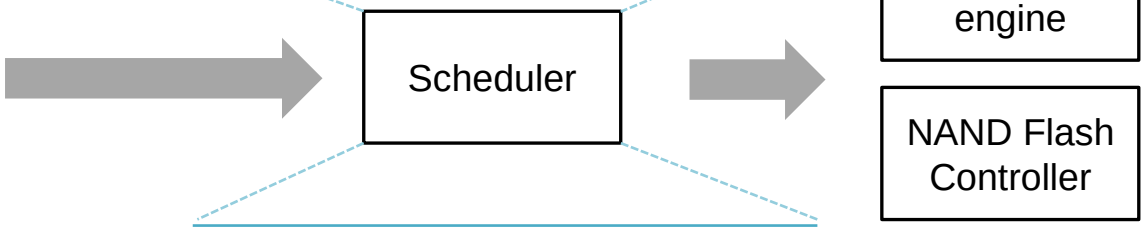
Priority-based Scheduling

- **Waiting commands are issued by scheduler**
  - Scheduler checks the state of flash memory controller and host interface controller
  - Priority of flash commands enhance multi channel, way parallelism



Command queues





Command	Priority
LLSCommand_RxDMA	0
LLSCommand_TxDMA	0
V2FCommand_StatusCheck	1
V2FCommand_ReadPageTrigger	2
V2FCommand_BlockErase	3
V2FCommand_ProgramPage	4
V2FCommand_ReadPageTransfer	5

Known Restrictions (1/3)

- **Firmware**
  - Supports
    - Buffer management (LRU)
    - Static page mapping
    - Garbage collection (On-demand)
  - Not supports
    - Meta flush
    - Wear leveling
  - Notice
    - I / O performance can be degraded when performing garbage collection
    - The number of usable blocks is limited when the MLC NAND array is used in the 4-channel 8-way structure
    - The latest firmware in SLC mode accesses only LSB pages of MLC NAND
    - Accessing to MSB pages may cause data errors not able to be corrected by ECC

MSB Page Data Error Issue

- The bit error rate increases if MSB pages of NAND flash are accessed
- Increased bit errors might not be corrected by BCH error correction engine in the current version of NAND flash controller
- For now, the firmware runs in SLC mode in order to reduce the error rate due to this reason

Known Restrictions (2/3)

- **PCIe-NVMe**
  - Supports
    - Up to PCIe Gen3.0 x16 lanes
    - Mendatary NVMe commands
    - PRP data transfer mechanism and out-of-order data transfer in PRP list
    - 1 namespce (can be extended by updating firmware)
    - Up to 8 NVMe IO submission queues and 8 NVMe IO completion queues with 256 depths
    - Up to 256 depths internal NVMe command table
    - MSI interrupt with 8 interrupt vectors
    - x86/x64 Ubuntu 14.04 and Windows 8.1
  - Not supports
    - 4 byte addressing yet (on debugging)
    - Optional NVMe commands (can be supported by updating firmware)
    - SGL data transfer mechanism
    - Power management (can be supported by updating firmware)
    - MSI-X interrupt
    - Virtualization and sharing features

Known Restrictions (3/3)

- **NAND flash controller**
  - Supports
    - Channel can be configured up to 4
    - Maximum bandwidth of NAND flash bus 200 MT
  - Not supports
    - Additional advanced commands are not supported (e.g. multi-plane operation)

Get Started with Daisy+ OpenSSD

Overall Steps

- **Preparing development environment**
  - Host computer
  - Platform board
  - Development tools
- **Building materials**
  - FPGA bitstream
  - Firmware
- **Operating Daisy+ OpenSSD**
  - Bitstream and firmware download to the FPGA
  - Host computer boot and SSD recognition check
  - SSD format
  - SSD performance evaluation and analysis

Tested Host PC Mainboard Compositions

Mainboard	BIOS Ver.	Result	Comment
Asrock Z77 Extream 6	P2.40	Working	
ASUS H87-Pro	0806x64	Working	
Gigabyte H97-Gaming 3	F5	Working	
Gigabyte Z97X-UD5H	F8	Working	
	F10c	Not working	4-byte addressing problems in Daisy+ PCIe DMA engine

Tested Host PC Operating System

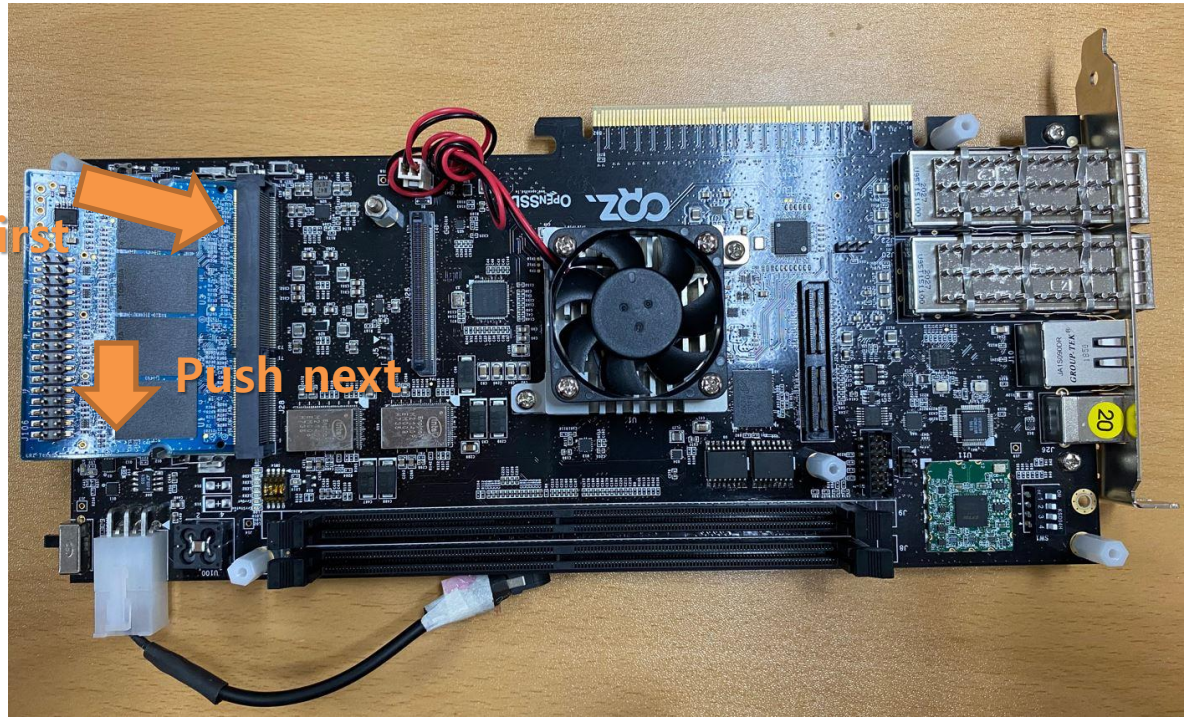
OS	x86/x64	Result	Comment
Windows 7	x64	Working	with OFA driver
Windows 8.1	x64	Working	
Windows 10	x64	Not working	4-byte addressing problems in D aisy+ PCIe DMA engine
Ubuntu 14.04 LTS or above	x64	Working	Kernel version 3.13 or above

Preparing the Platform Board

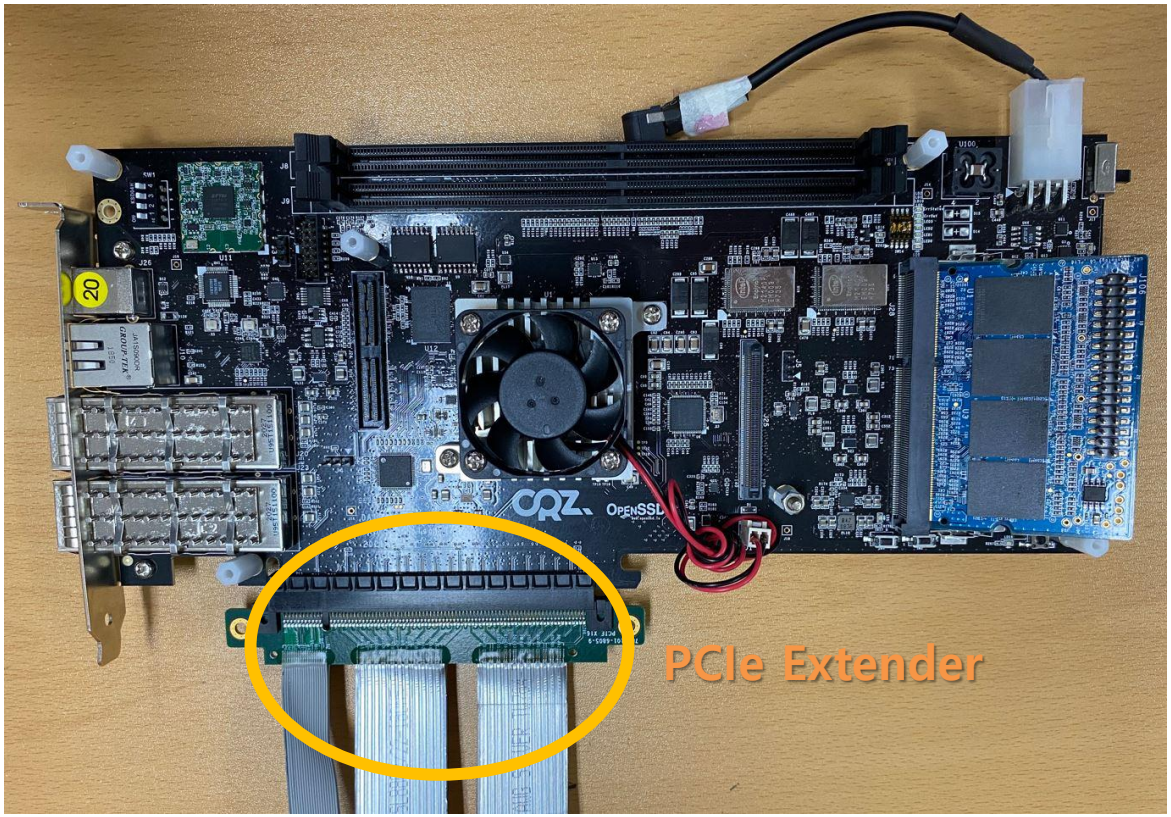
- Insert NAND flash module
- Connect the PCIe extender
- Connect the USB cable for jtag / UART
- Connect the power cable

Insert NAND Flash Module

- **A single NAND flash module can support up to 4-channel configuration**

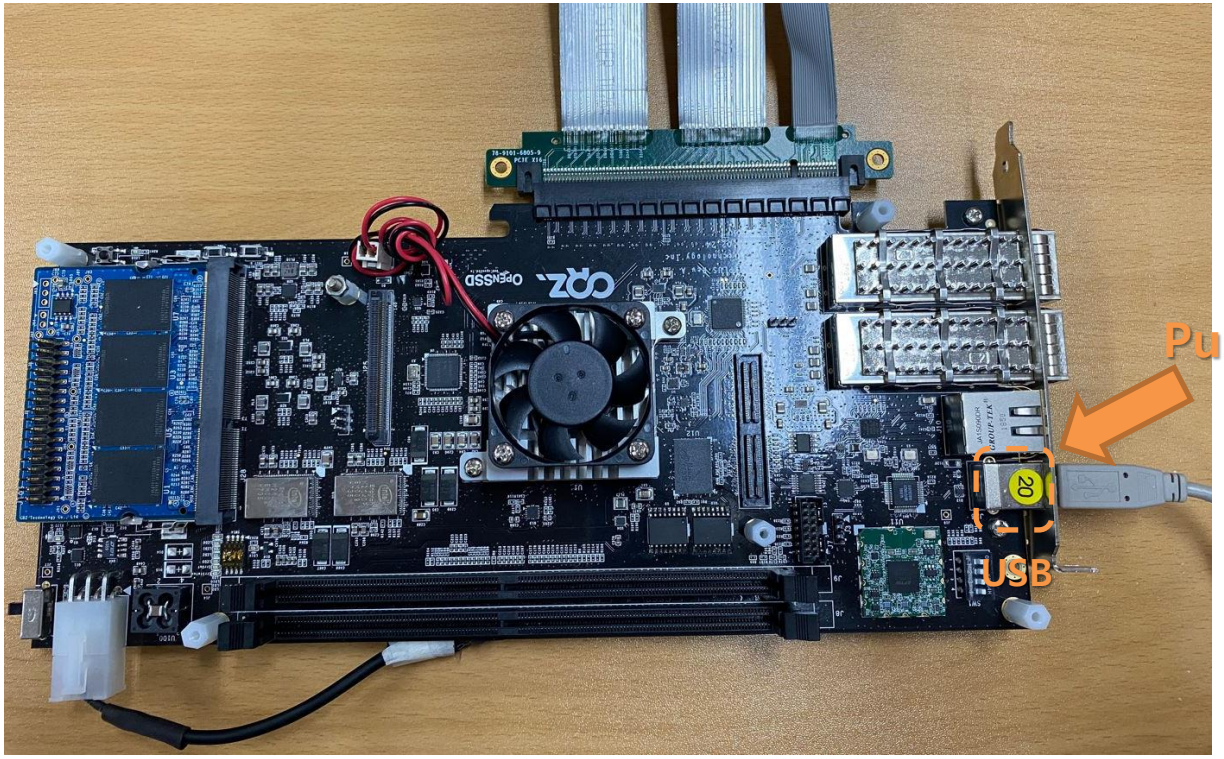


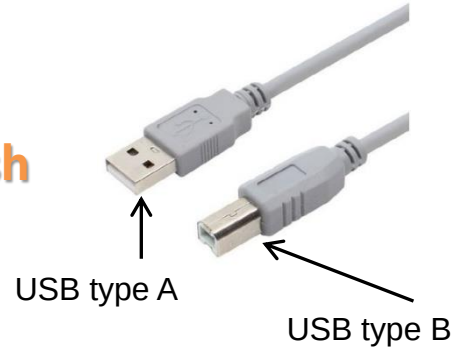
Connect PCIe Extender



Connect USB Cable

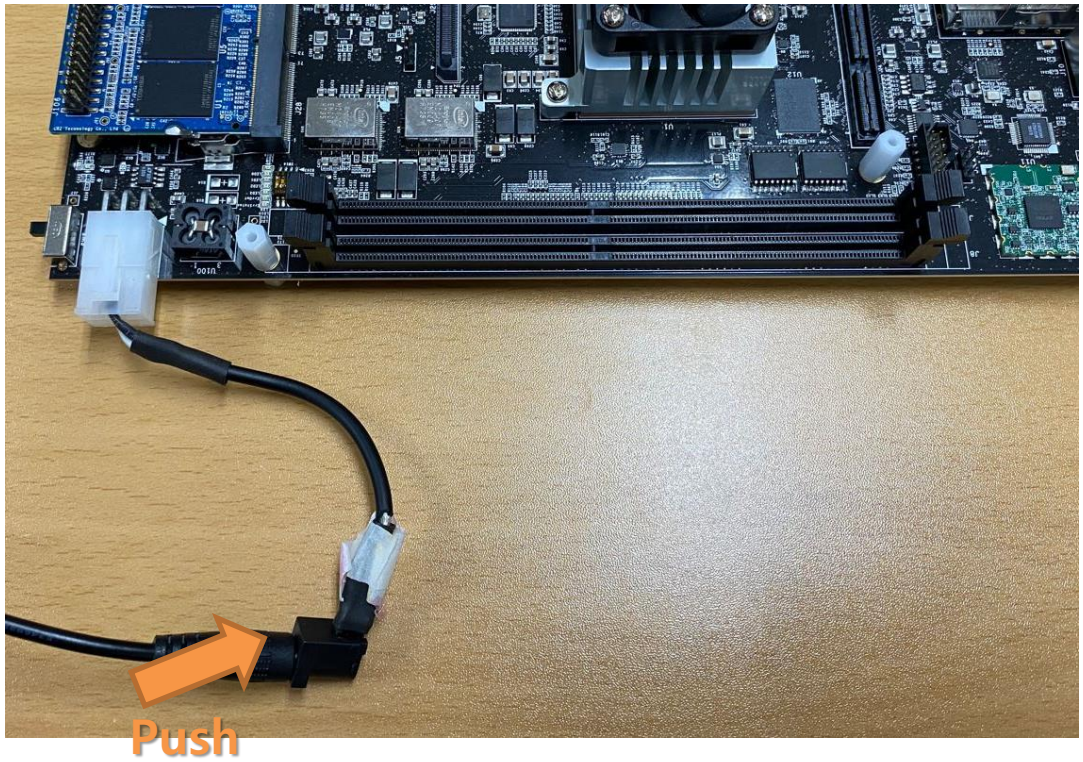
- **USB requires a USB type B (male) to USB type A (male) cable**





Connect Power Cable

- **Connect the power cable to the 5.5 mm power connector**



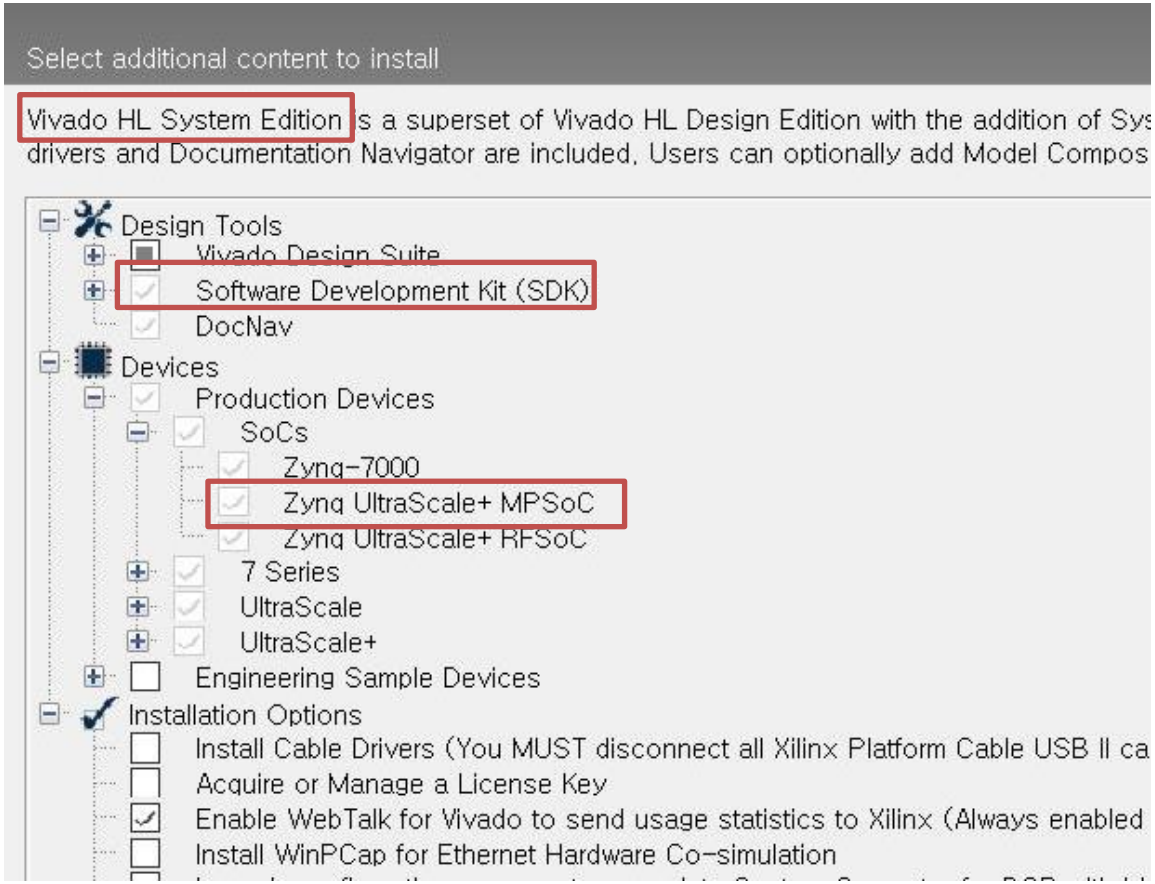
Preparing Software for Development PC

- **Download materials**
  - Pre-defined Vivado project for manual FPGA bitstream generation
  - Firmware source code

- **Install Xilinx Vivado Design Suite: System Edition 2019.1**
  - Xilinx Vivado 2019.1
  - Xilinx SDK 2019.1

Install Xilinx Vivado Design Suite

- **Make sure that Vivado is HL System Edition and that “Software Development Kit” and “Zynq UltraScale+ MPSoc” are checked**

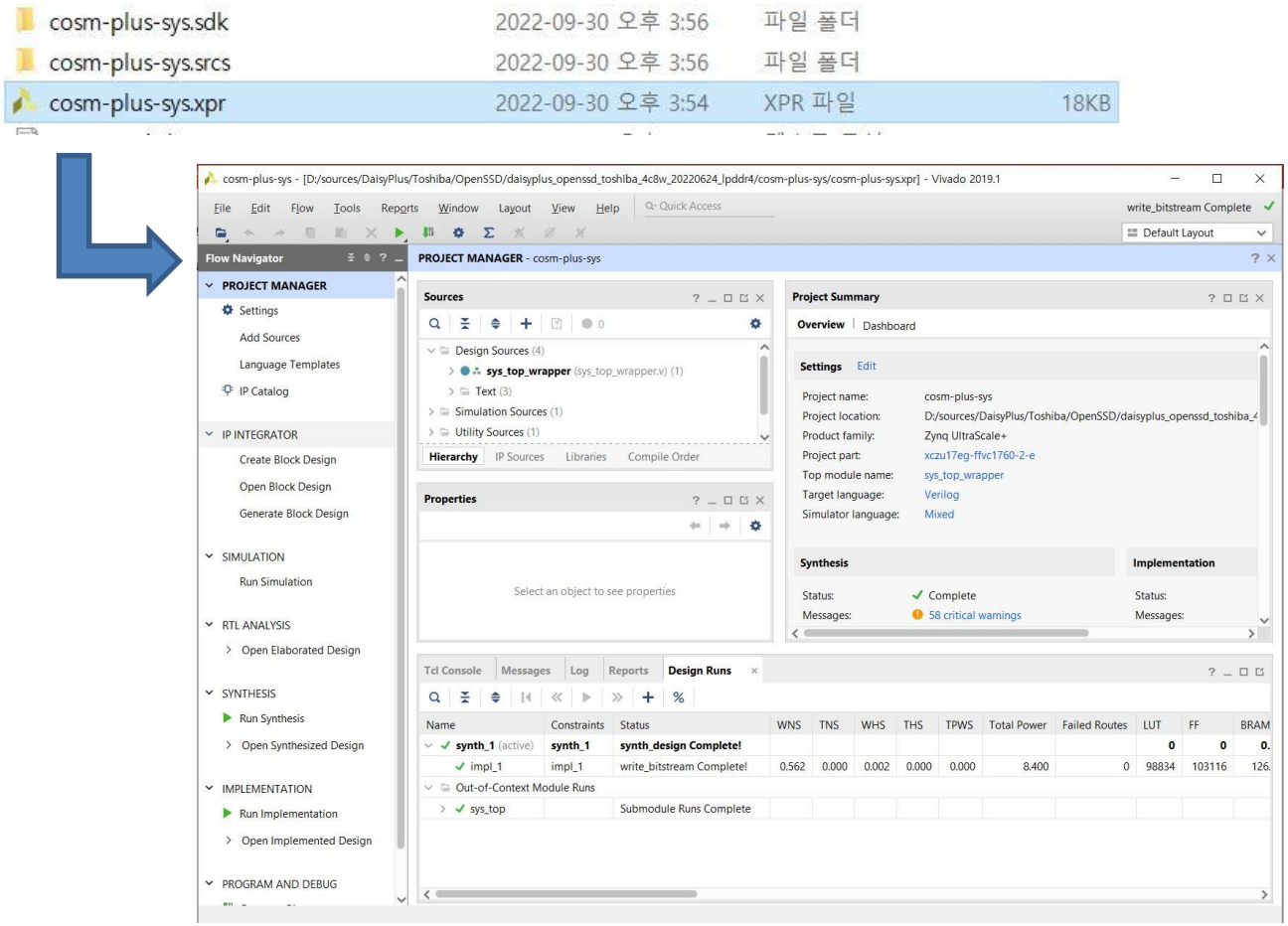


Generating FPGA Bitstream for Pre-defined Project

1. **Run synthesis**
2. **Run implementation**
3. **Generate bitstream**
4. **Export hardware**

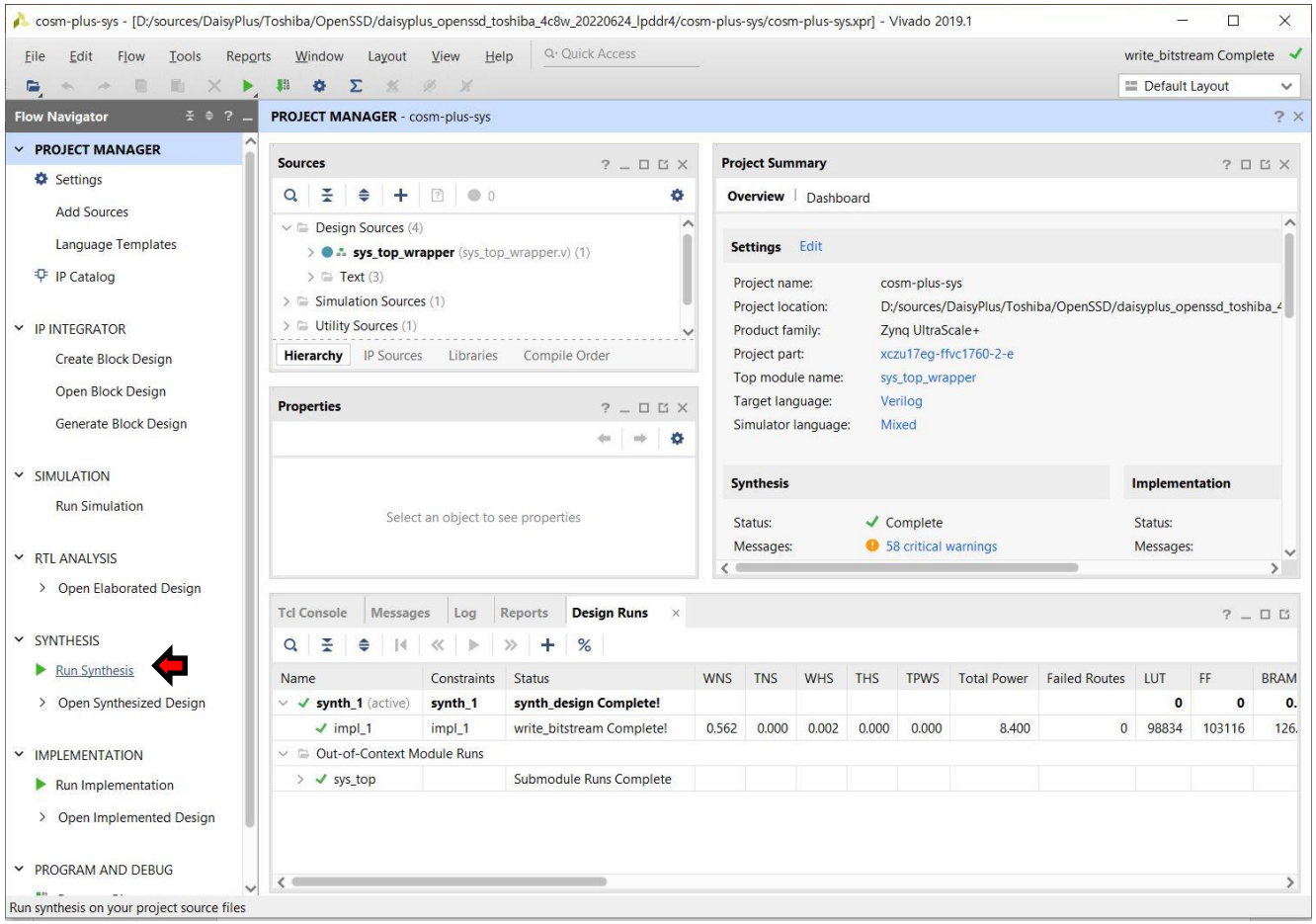
Launch Xilinx Vivado

- **Open project included in daisyplus_openssd_toshiba_4c8w_lpddr4**



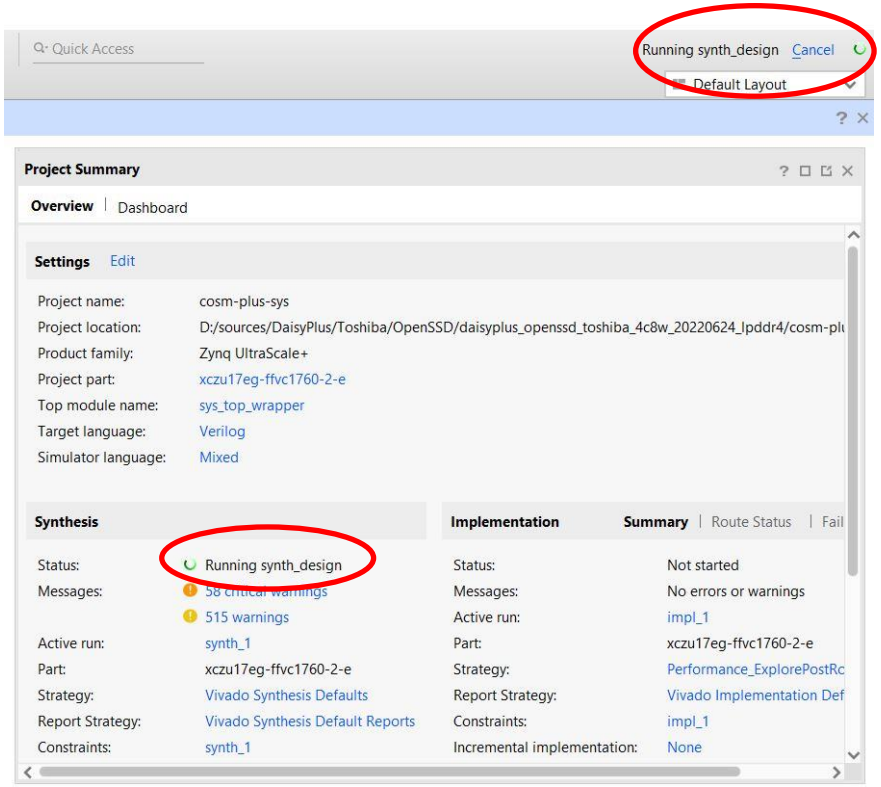
Run Synthesis (1 / 2)

- **Click “Run Synthesis”**



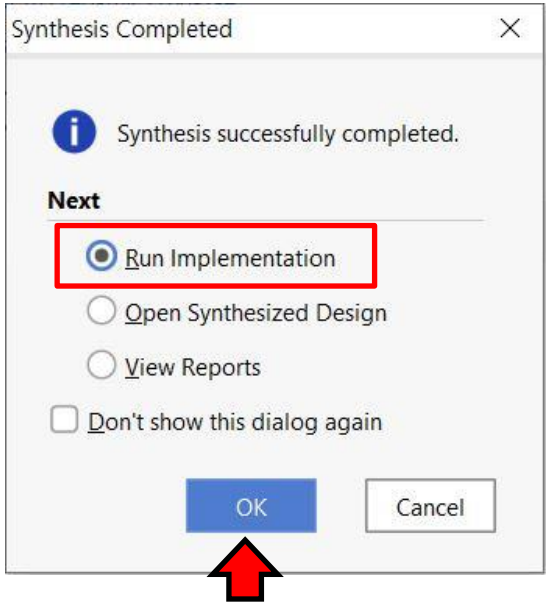
Run Synthesis (2 / 2)

- **Synthesis is running…**



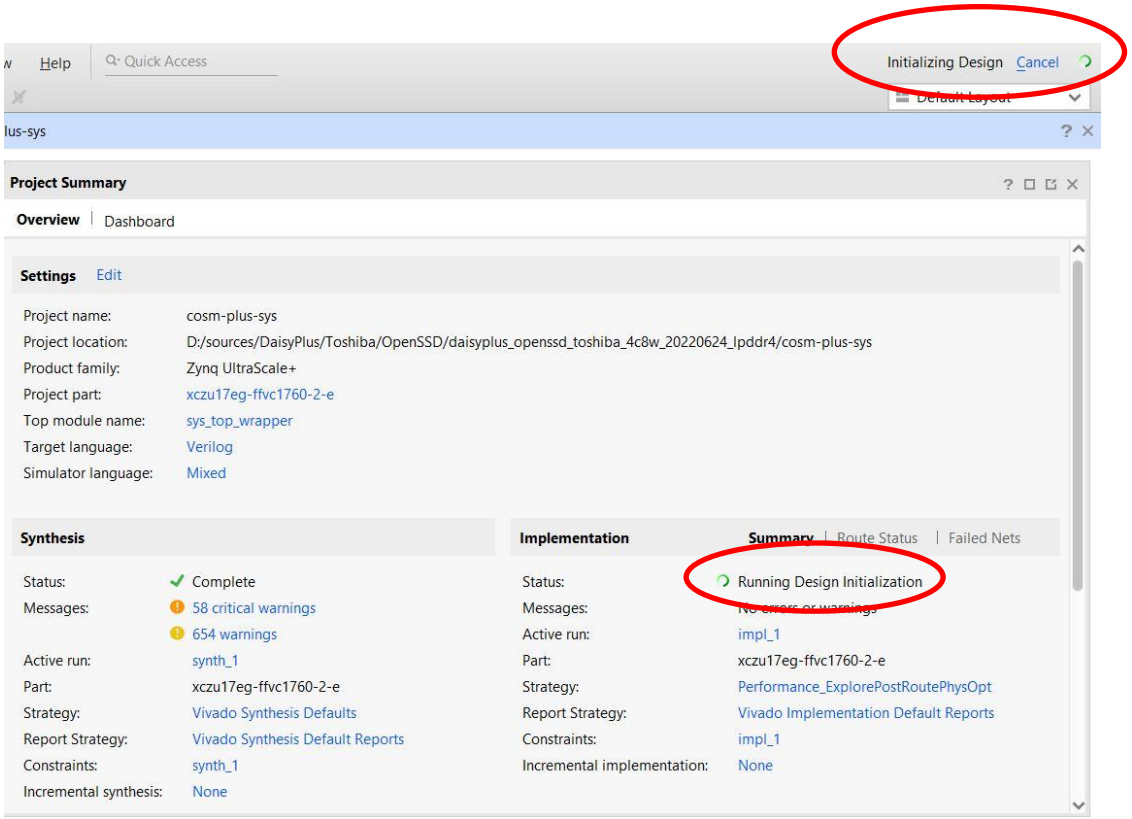
Synthesis Complete

- **Select “Run Implementation” and click OK**
  - If you want to see the synthesized results, choose “Open Synthesized Design” or “View Reports”



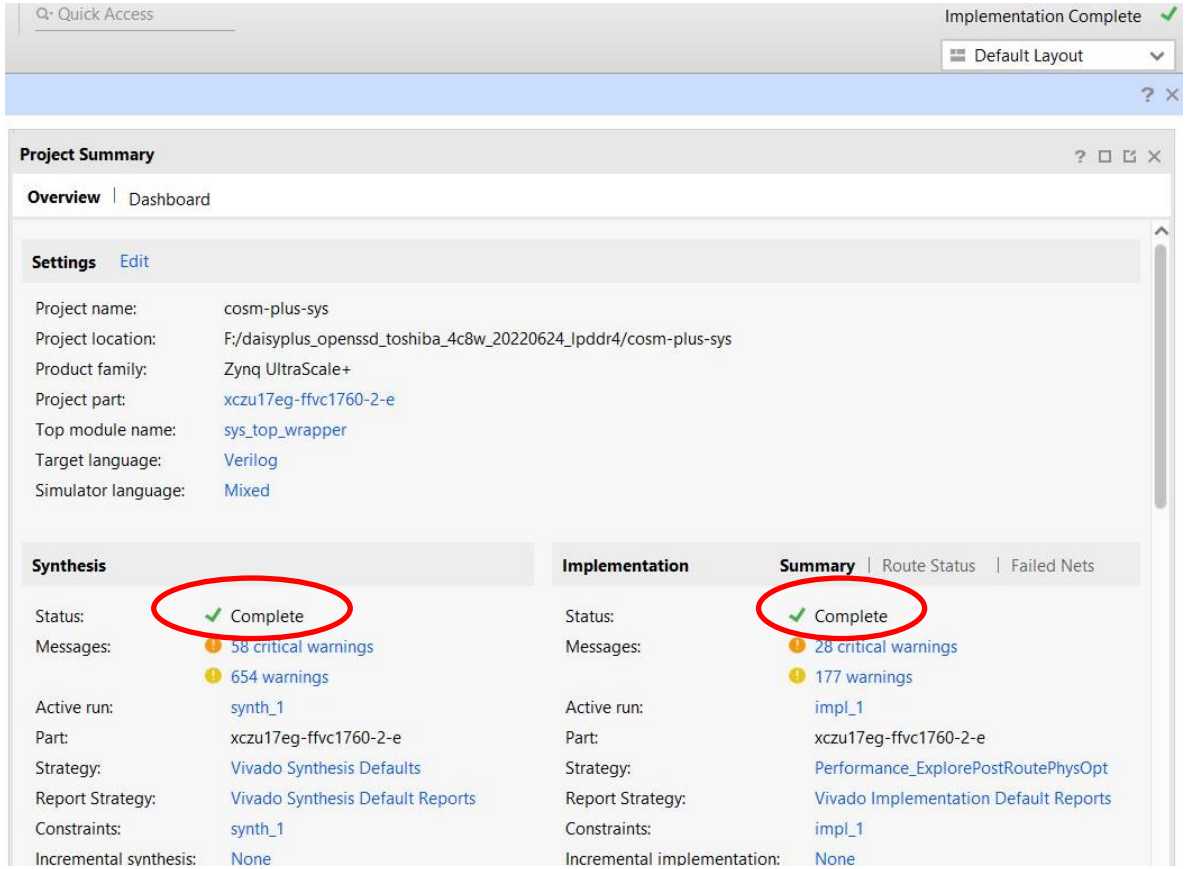
Run Implementation

- **Implementation is running…**



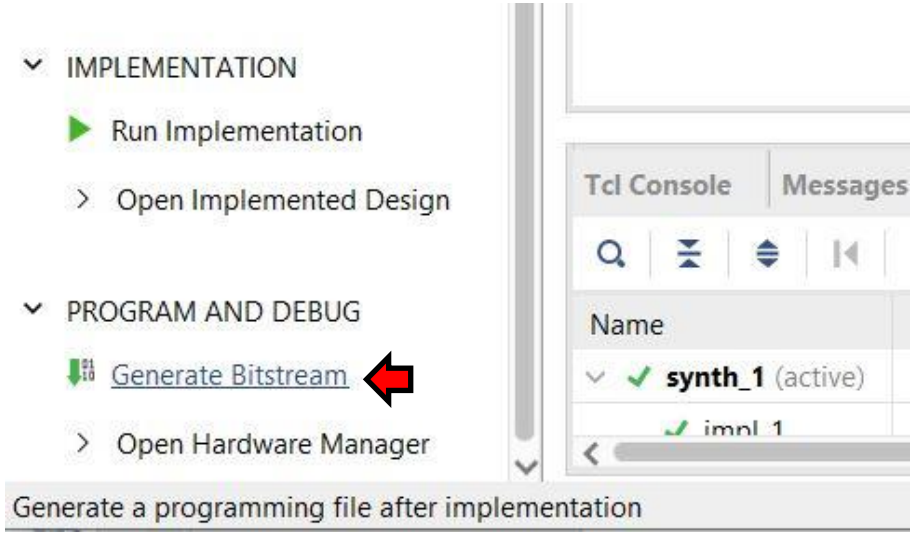
Implementation Complete

- **Check the status of synthesis and implementation**



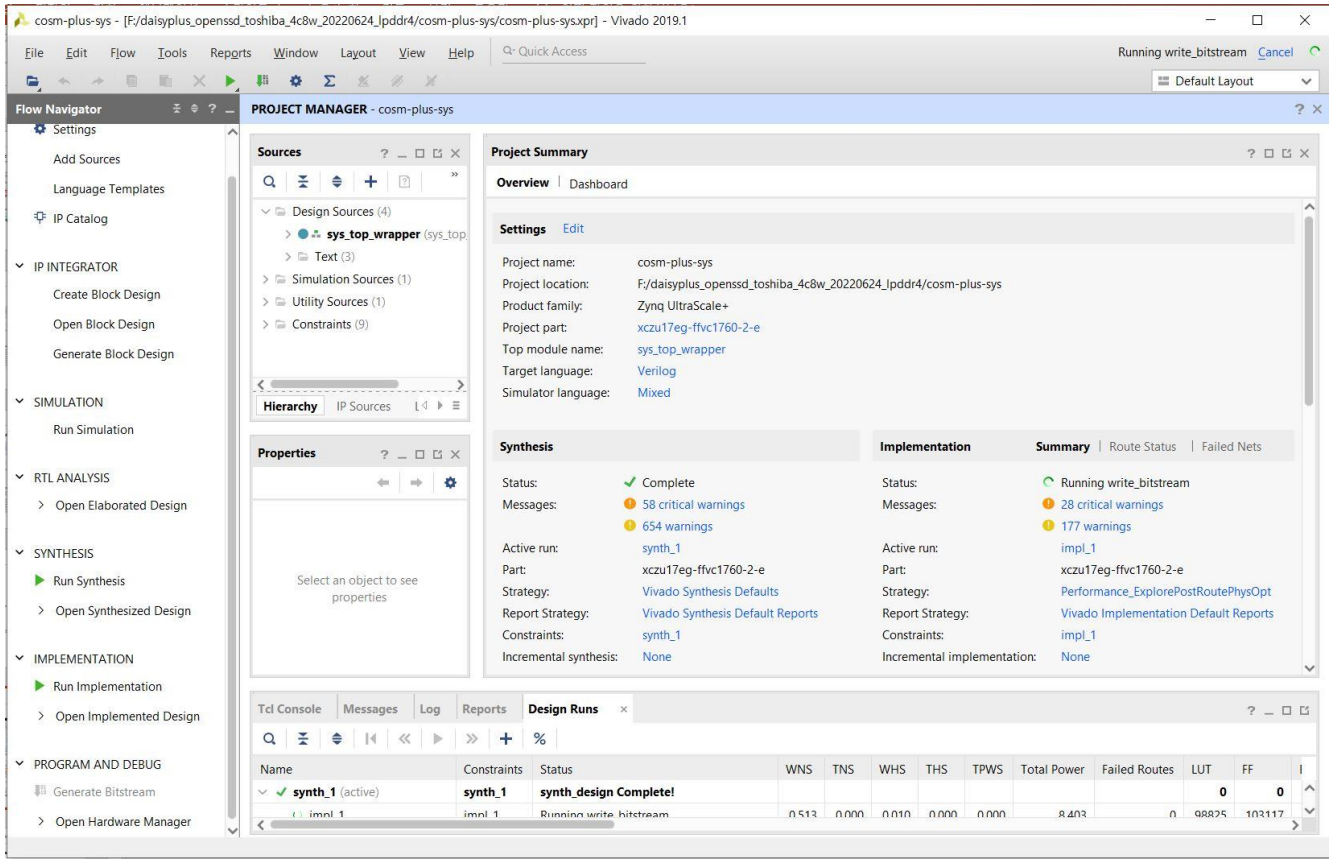
Generate Bitstream (1 / 2)

- **Click “Generate Bitstream”**



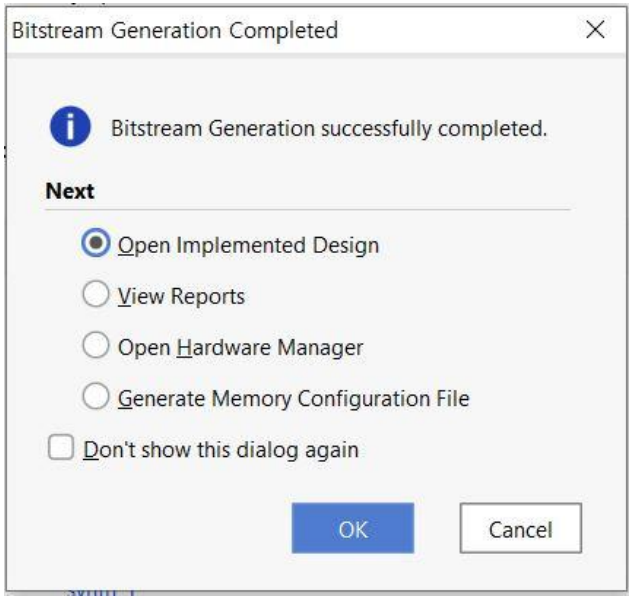
Generate Bitstream (2 / 2)

- **Generate bitstream is running…**



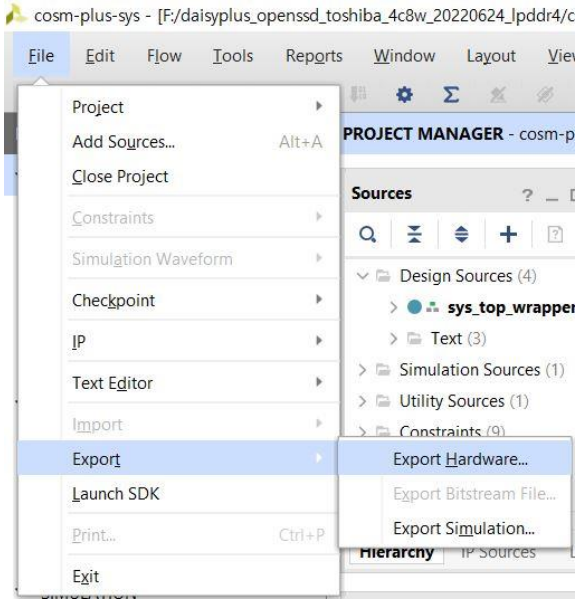
Bitstream Generation Complete

- **If you want to see the implemented design, select open implemented design and click the OK button**



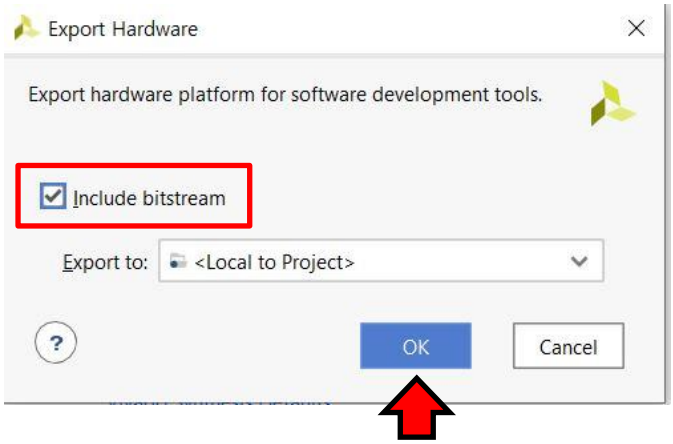
Export hardware (1 / 2)

- Go to File -> Export and click “Export Hardware”



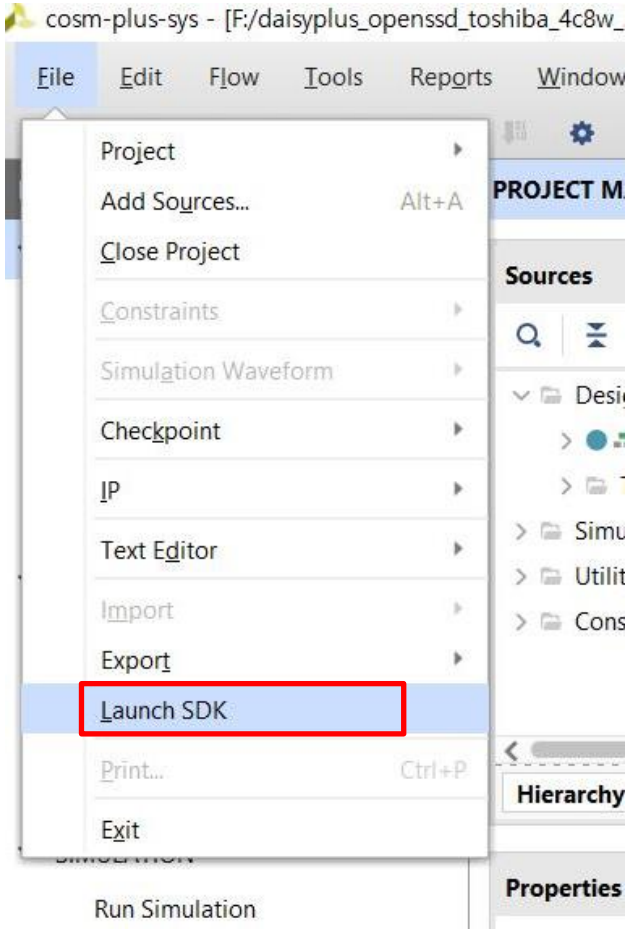
Export hardware (2 / 2)

- **Select the “Include bitstream” and click OK**



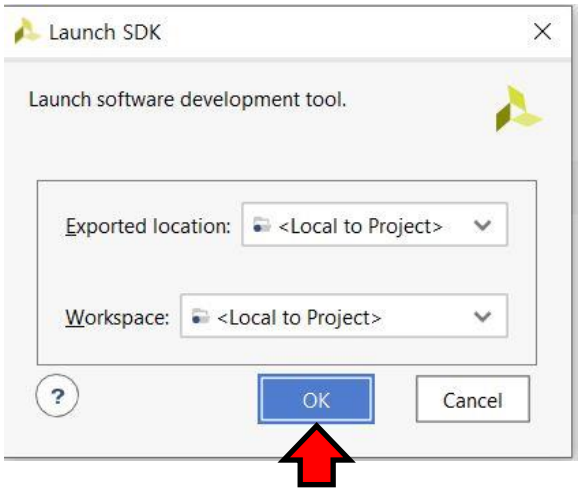
Launch SDK (1 / 4)

- **Go to File -> Launch SDK**



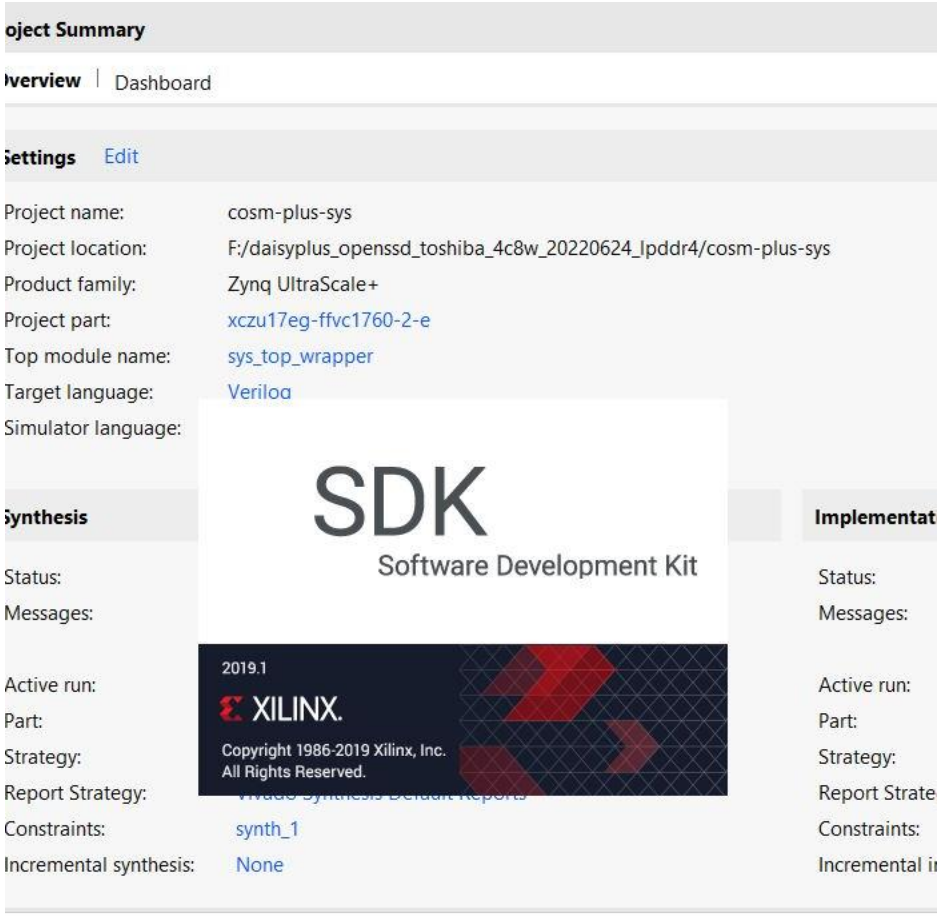
Launch SDK (2 / 4)

- **Click the OK button**



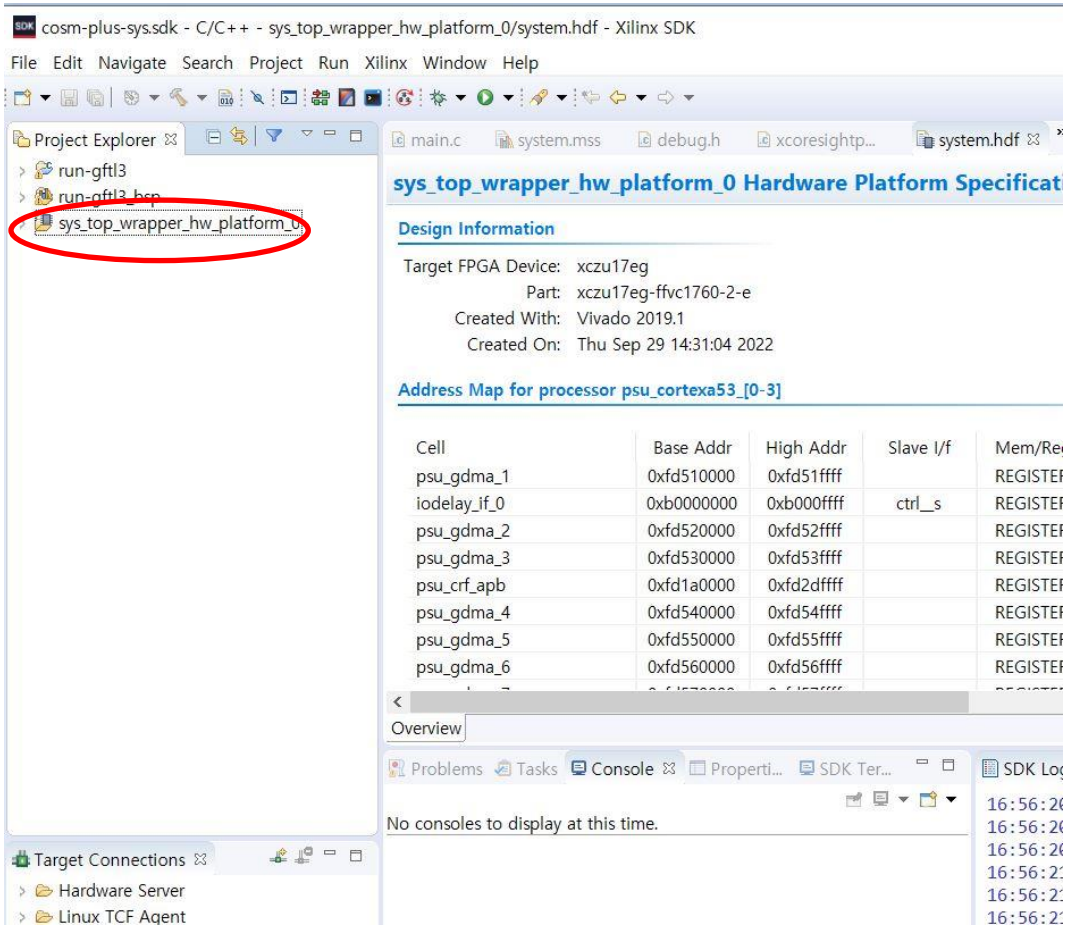
Launch SDK (3 / 4)

- **Then, SDK is launched**



Launch SDK (4 / 4)

- **As shown below, exported hardware platform is set as target hardware**

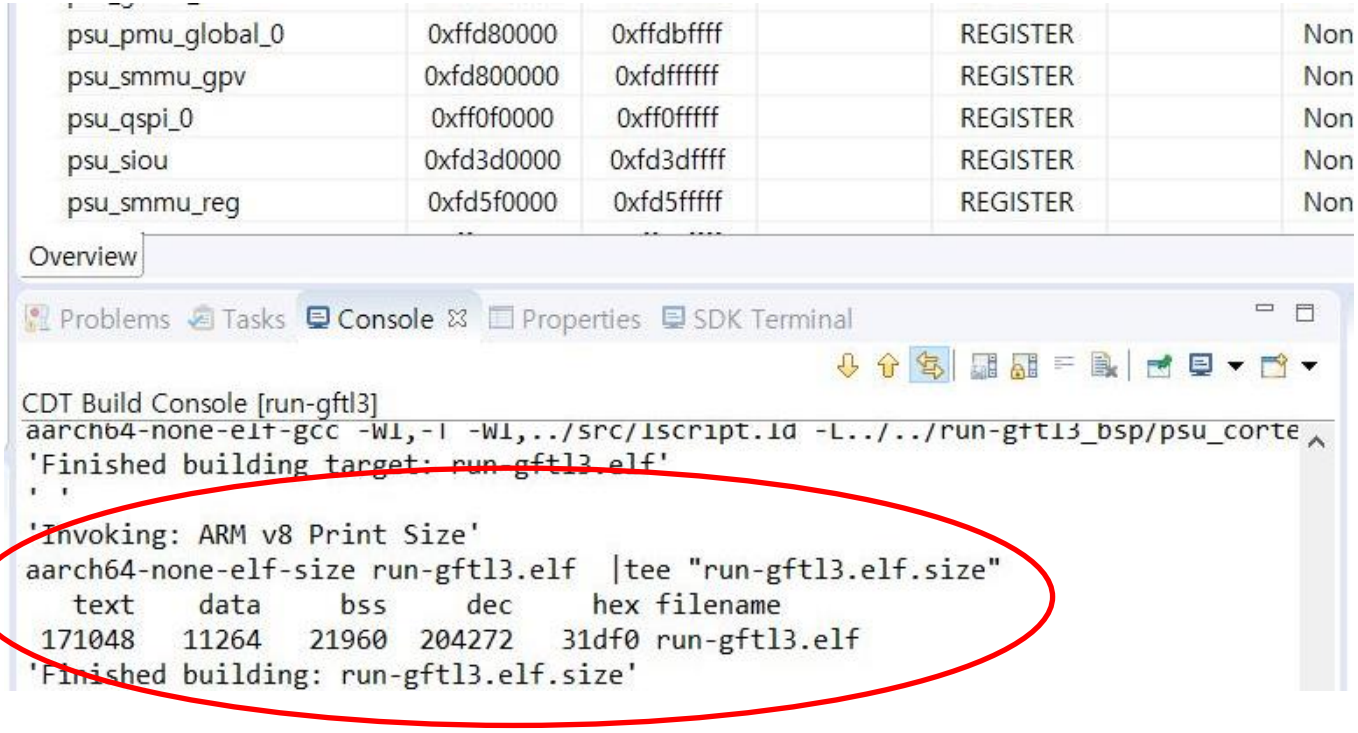


Building Firmware for Project

1. **Build firmware source codes**

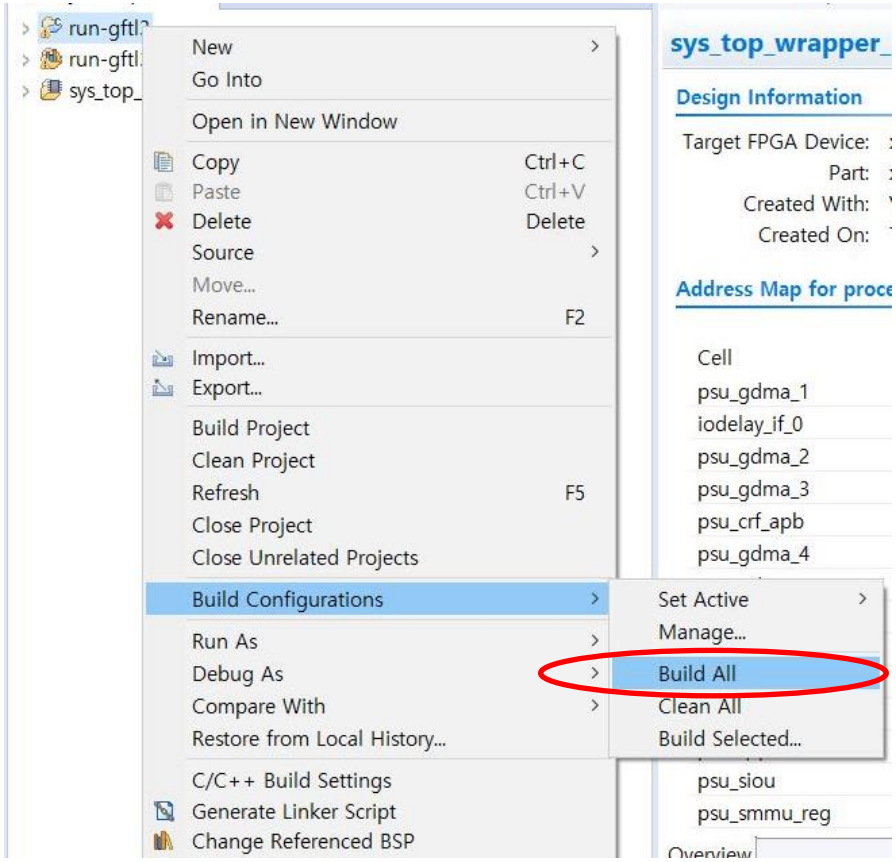
Build Firmware (1 / 2)

- **If everything goes well, the automatic build process should finish successfully**



Build Firmware (2 / 2)

- **Click “Build All” to make both debug and release executables**

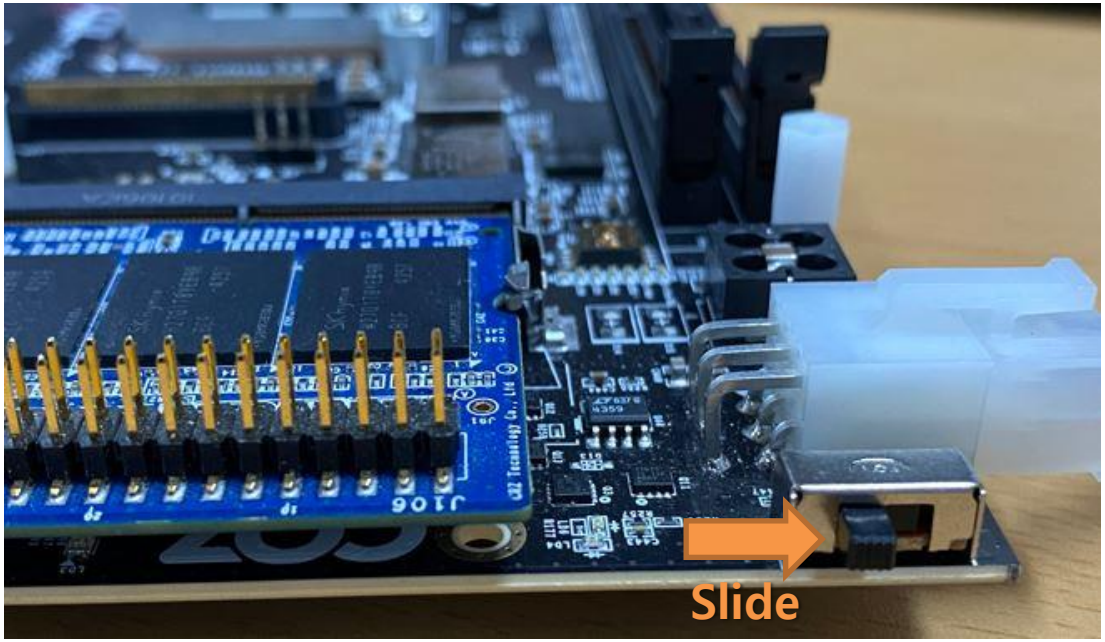


Preparing for Operating Daisy+ OpenSSD

1. **Power on the platform board**
2. **Configure UART**
3. **Execute firmware**

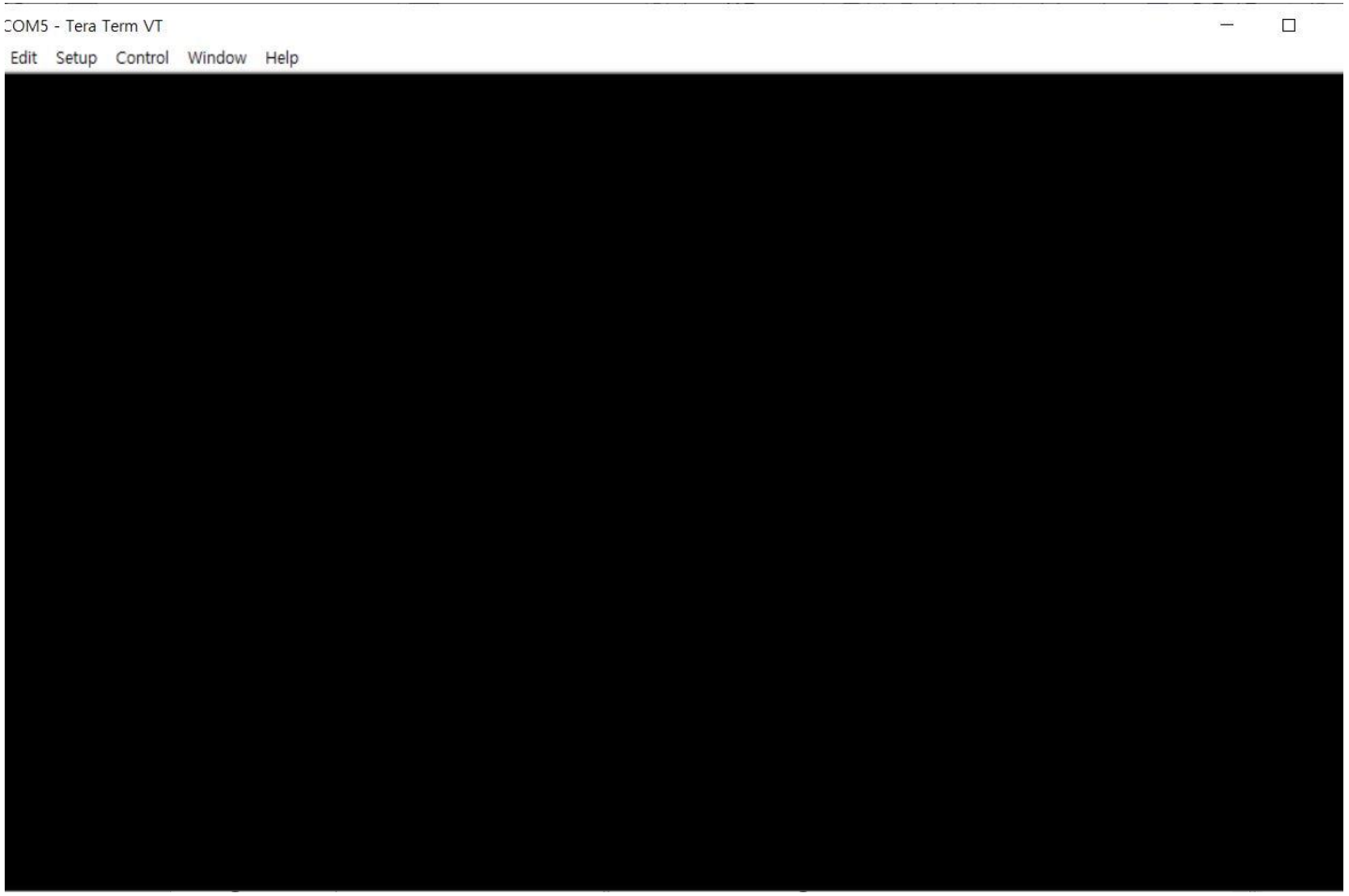
Power on the Platform Board

- **Before you power on the board, make sure that your host computer is powered off**



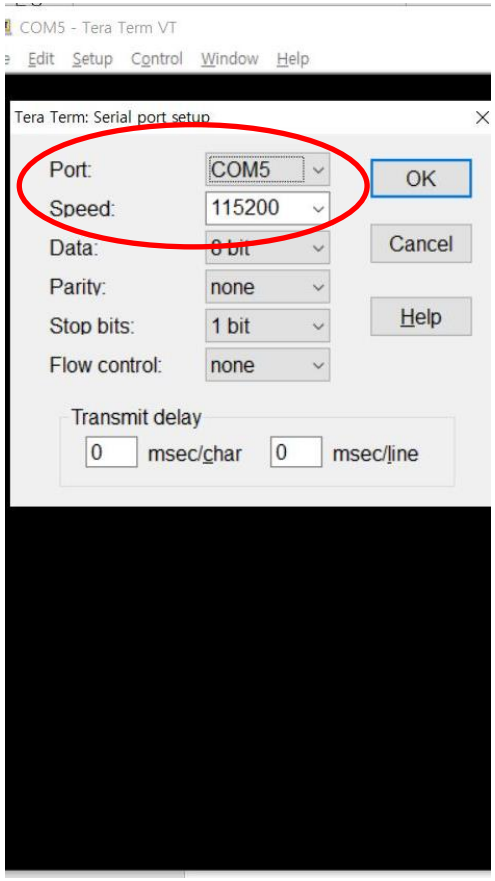
Configure UART

- **Open terminal program.**



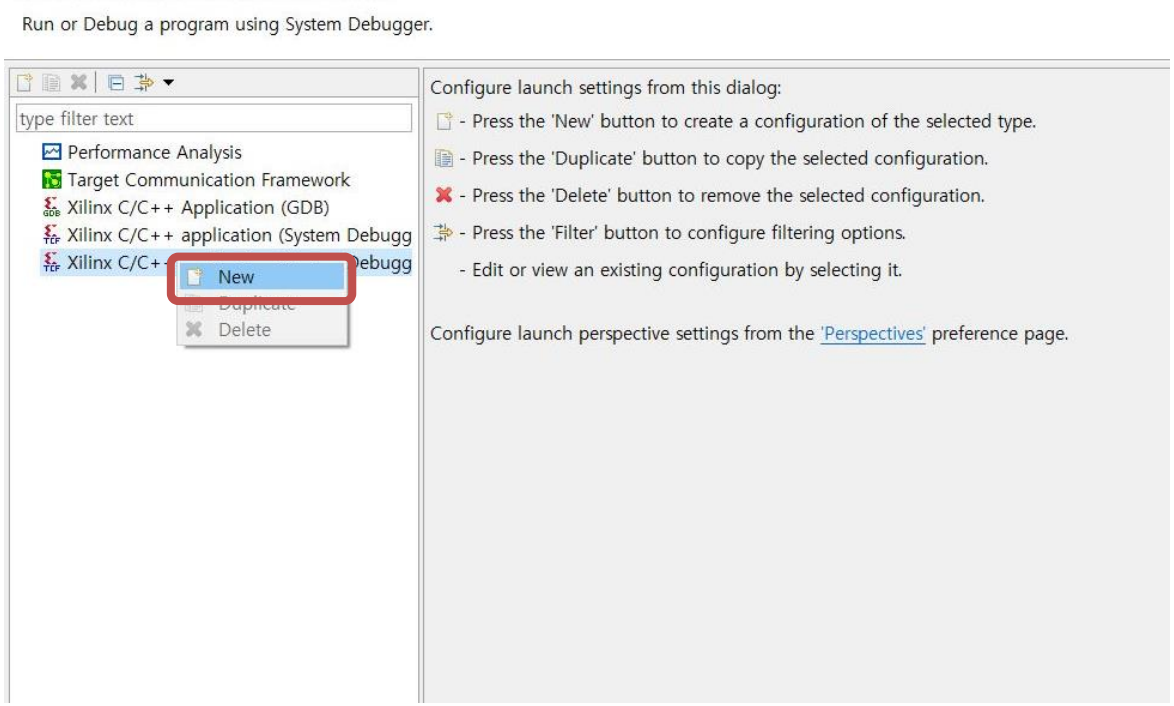
Configure UART

- On Terminal program, set “Port” and “Baud Rate” to 115200, respectively



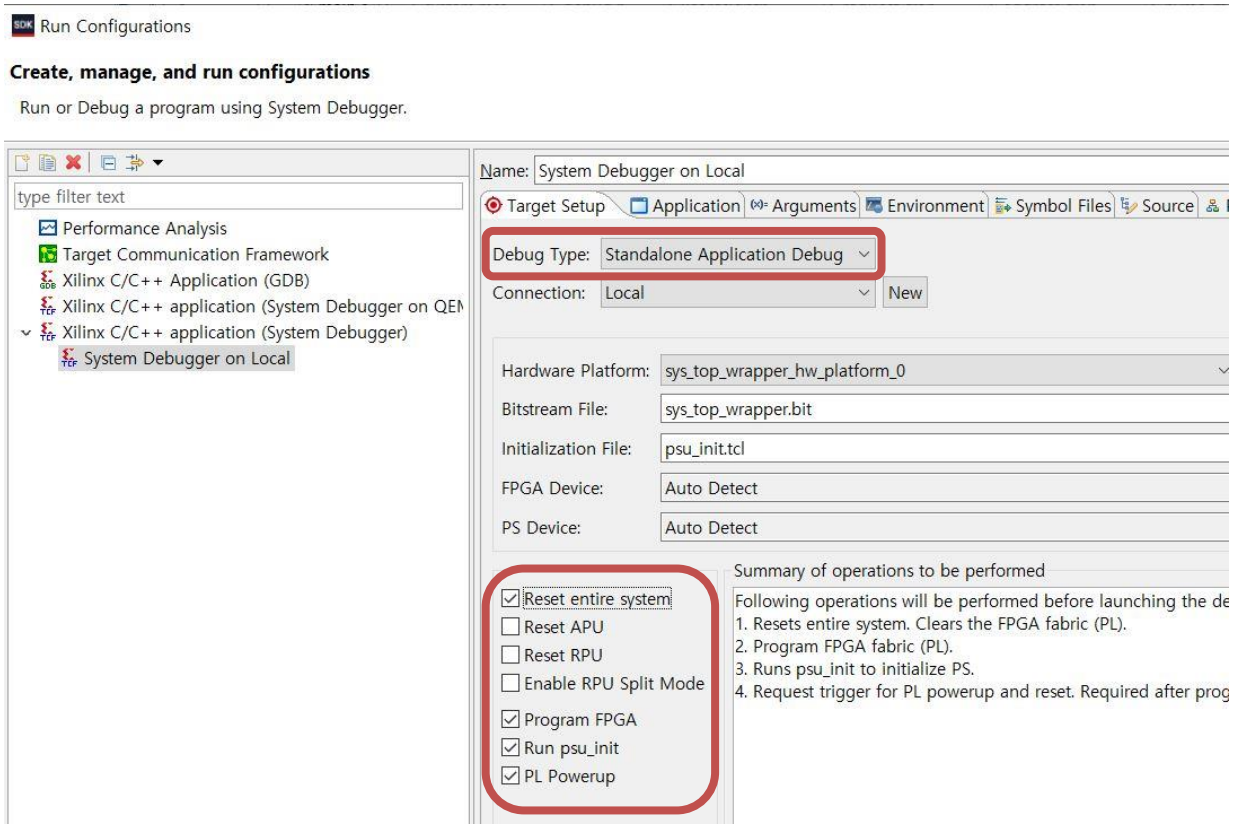
Execute Firmware (1 / 4)

- **Click “Run” -> “Run Configurations” and right-click on “Xilinx C/C++ application(System Debugger) and click “New”.**



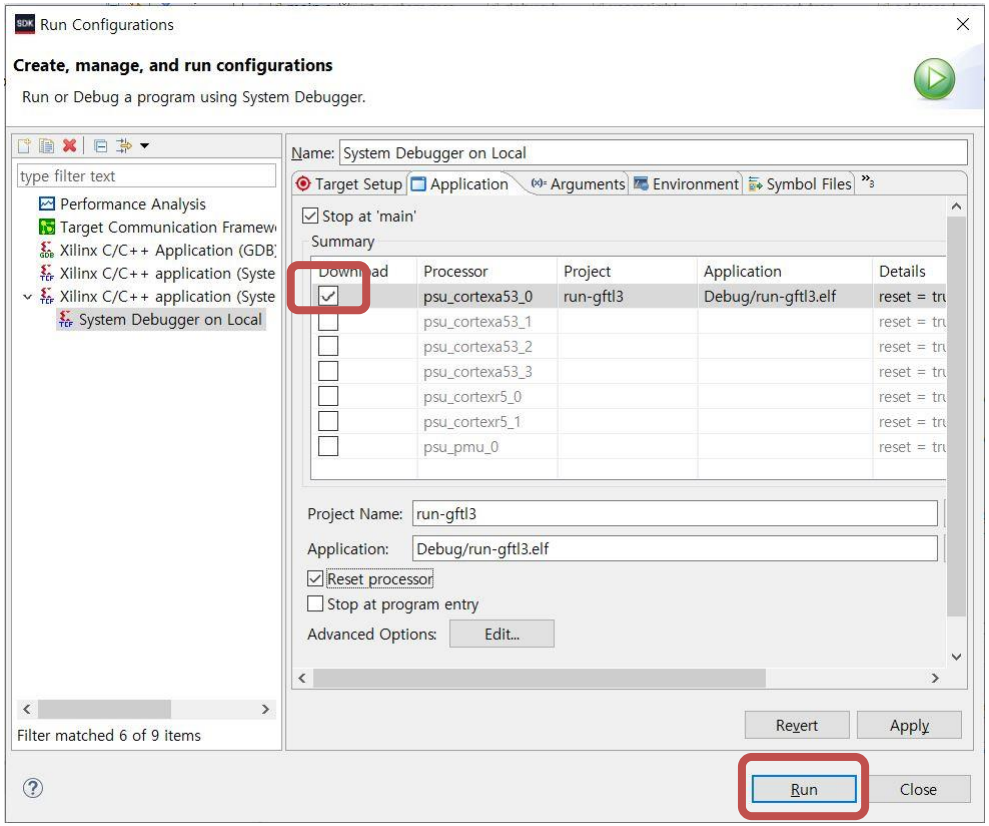
Execute Firmware (2 / 4)

- **On “Target Setup” tab, select “Standalone Application Debug” as “Debug Type” and check “Reset entire system”.**



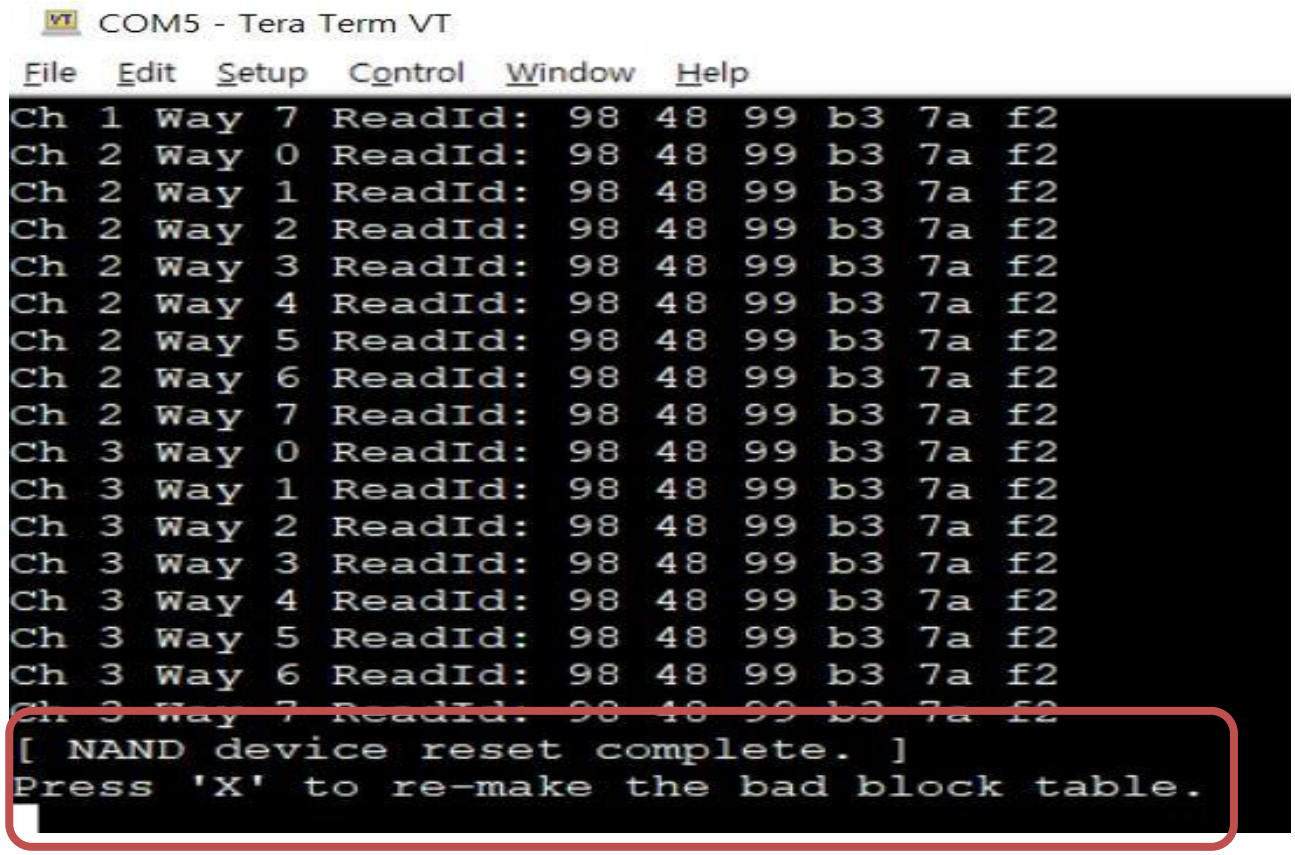
Execute Firmware (3 / 4)

- On “Application” tab, check “psu_cortexa53_0” and click “Run”.



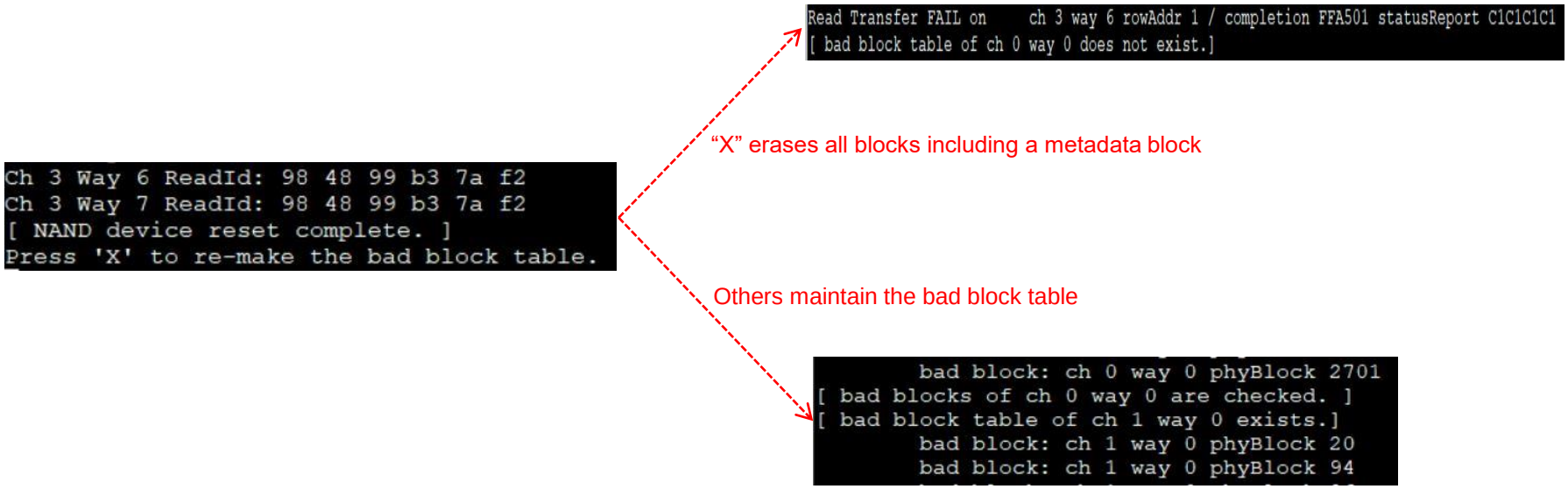
Execute Firmware (4 / 4)

- **Press ‘n’ to maintain the bad block table**



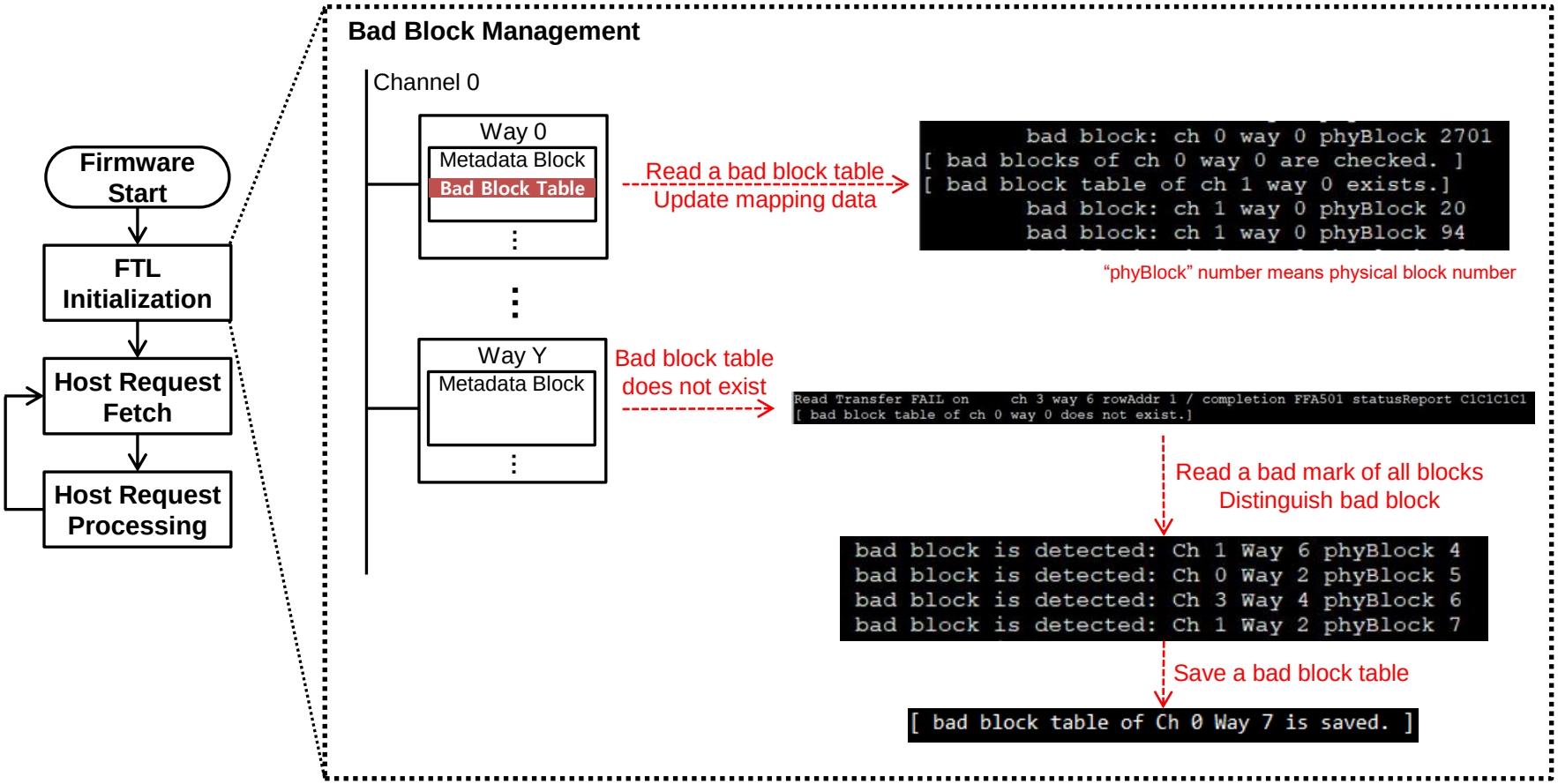
Bad Block Management (1 / 2)

- **Choose whether remake the bad block table in FTL initialization step**
  - If you want to remake the bad block table, press “X” on UART terminal
    - Damaged bad block table can be recovered



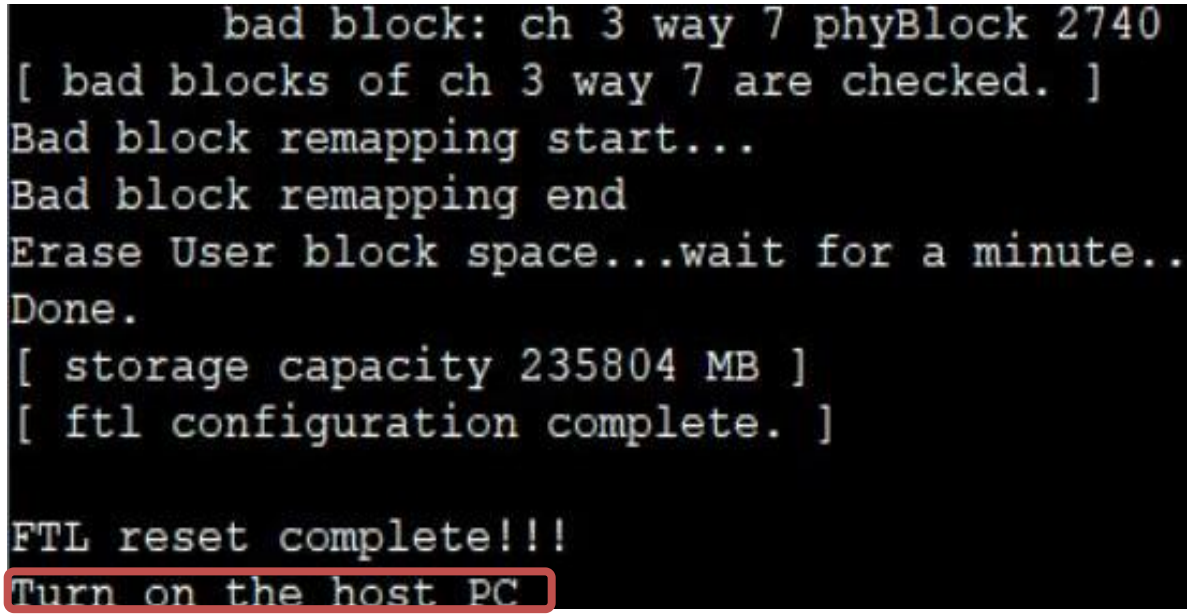
Bad Block Management (2 / 2)

- **Bad blocks are detected in FTL initialization step**



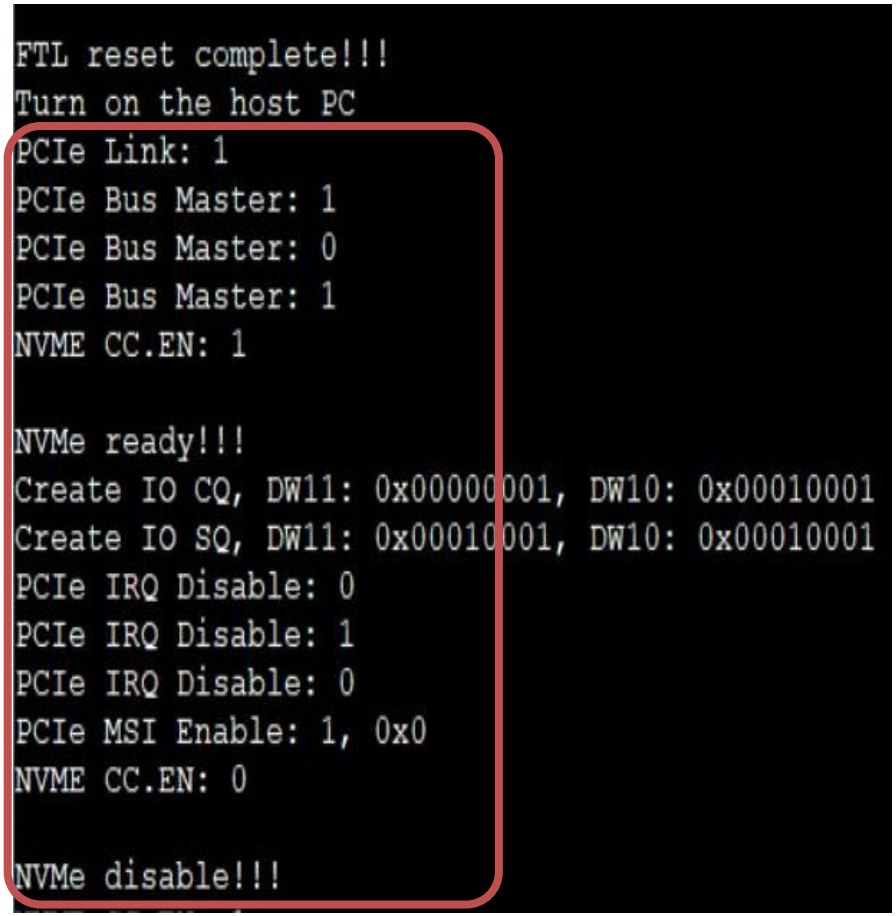
Turn on the Host PC

- **Turn on the host PC when the firmware reset is done**



UART Messages While Host Computer is Booting up

- **NVMe SSD initialization steps are on going**

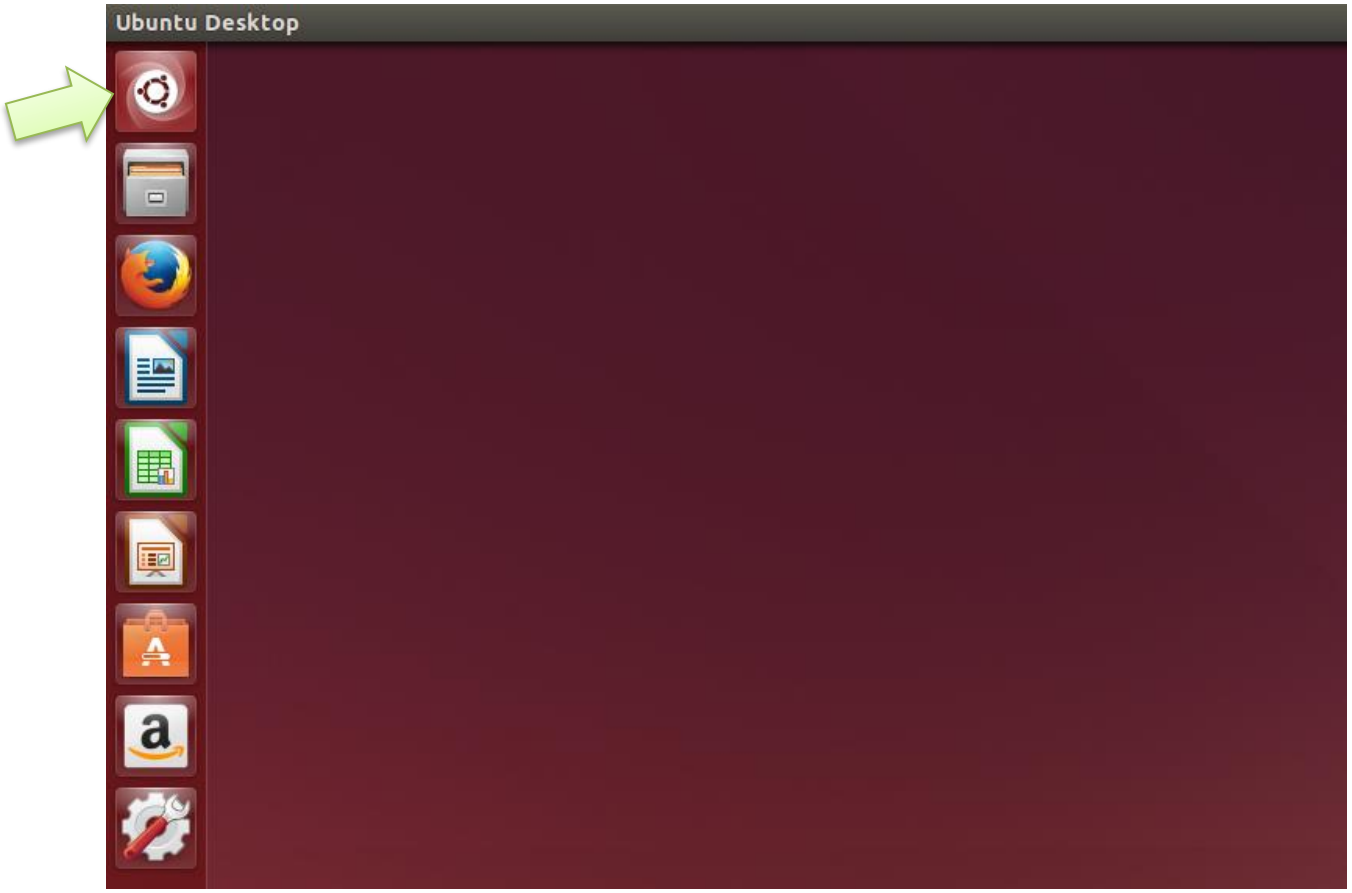


Operating Daisy+ OpenSSD (Linux)

1. **Check device recognition**
2. **Create a partition**
3. **Check the created partition**
4. **Format the partition**
5. **Create a mount point**
6. **Mount the partition**
7. **Check the mounted partition**

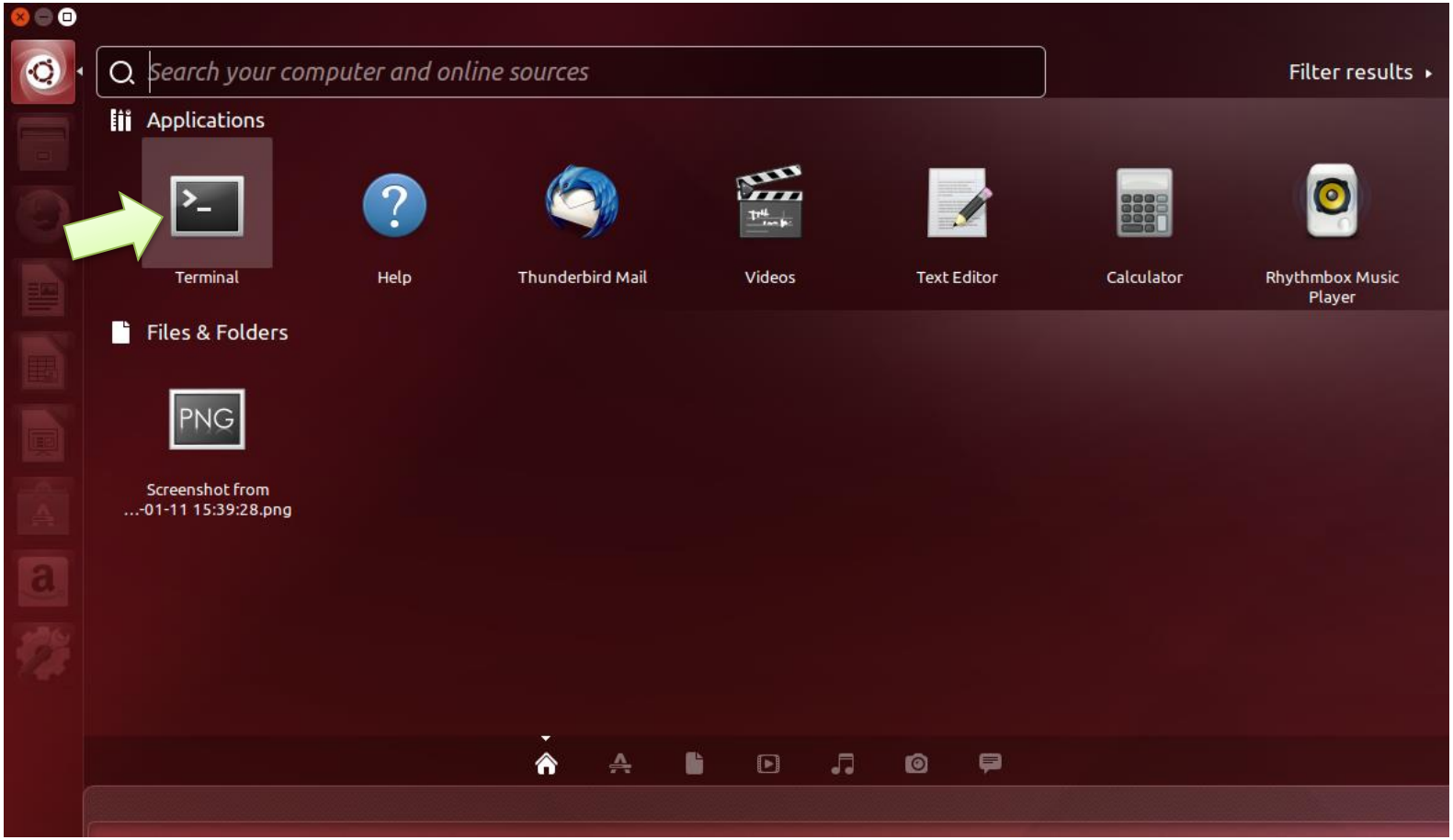
Open a Terminal (1 / 2)

- **Click the pointed icon**



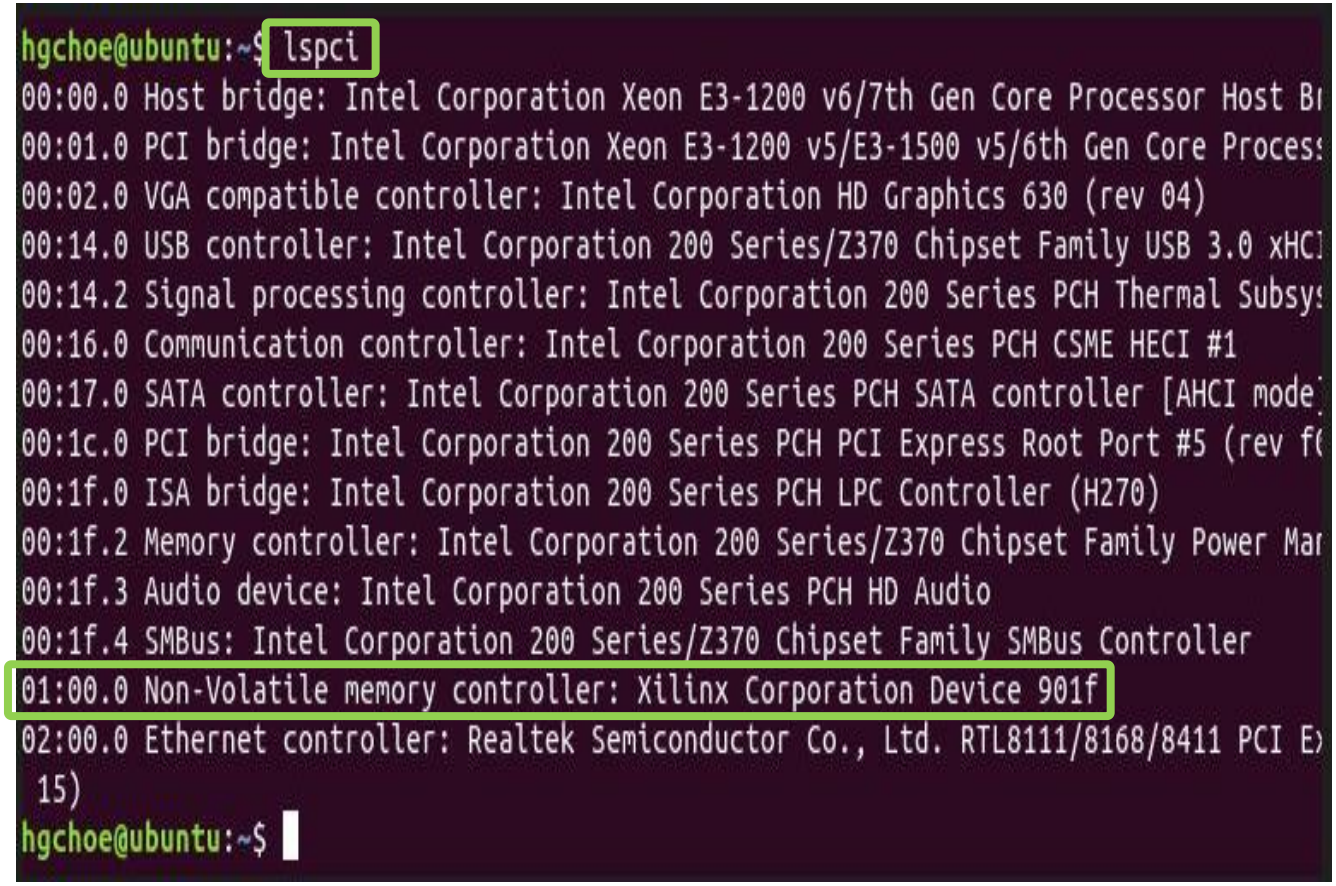
Open a Terminal (2 / 2)

- **Click the terminal icon**



Check Device Recognition (1 / 2)

- **Types “lspci” -> press ENTER -> check “Non-Volatile memory controller: Xilinx Corporation Device 901f” on the PCI device list**



Check Device Recognition (2 / 2)

- Types “ls /dev” -> press ENTER -> check “nvme0n1” on the device list

Create a Partition

- **Type “sudo fdisk /dev/nvme0n1”, press ENTER -> type your password, press ENTER -> type “n”, press ENTER->press ENTER->press ENTER, press ENTER -> type “w”, press ENTER**

```
hgchoe@ubuntu:~$ sudo fdisk /dev/nvme0n1
[sudo] password for hgchoe:

Welcome to fdisk (util-linux 2.31.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0xbf7cc307.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p):

Using default response p.
Partition number (1-4, default 1):
First sector (256-60365823, default 256):
Last sector, +sectors or +size{K,M,G,T,P} (256-60365823, default 60365823):

Created a new partition 1 of type 'Linux' and of size 230.3 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

hgchoe@ubuntu:~$
```

Check the Created Partition

- Types “ls /dev” -> press ENTER -> check “nvme0n1p1” on the device list

Format the Partition

- Type “sudo mkfs -t ext4 / dev/nvme0n1p1”, press ENTER

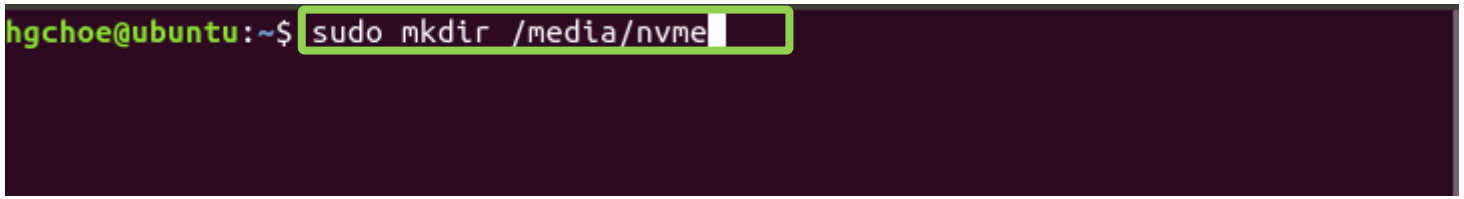
```
hgchoe@ubuntu:~$ sudo mkfs -t ext4 /dev/nvme0n1p1
mke2fs 1.44.1 (24-Mar-2018)
Creating filesystem with 60365568 4k blocks and 15097856 inodes
Filesystem UUID: 4c5f1175-5c68-459d-b516-558811bc19ca
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (262144 blocks): done
Writing superblocks and filesystem accounting information: done

hgchoe@ubuntu:~$
```

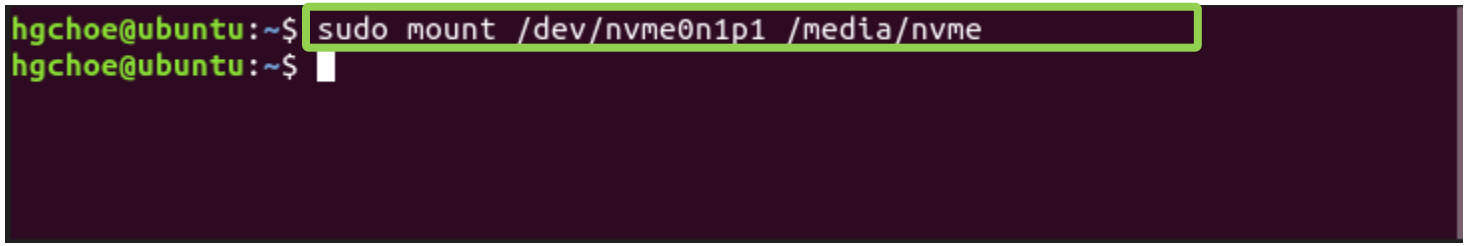
Create a Mount Point

- **Type “sudo mkdir /media/nvme”, press ENTER**



Mount the Partition

- Type “sudo mount /dev/nvme0n1p1 /media/nvme”, press ENTER



Check the Mounted Partition (1 / 2)

- **Type “lsblk”, press ENTER -> check the mounted partition on the block device list**

```
hgchoe@ubuntu:~$ lsblk
NAME          MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0           7:0    0  70.4M  1 loop /snap/core22/275
loop1           7:1    0   556K  1 loop /snap/gnome-logs/112
loop2           7:2    0   1.5M  1 loop /snap/gnome-system-monitor/181
loop3           7:3    0   476K  1 loop /snap/gnome-characters/781
loop4           7:4    0    48M  1 loop /snap/snapd/17029
loop5           7:5    0    48M  1 loop /snap/snapd/17336
loop6           7:6    0   219M  1 loop /snap/gnome-3-34-1804/72
loop7           7:7    0  63.2M  1 loop /snap/core20/1623
loop8           7:8    0  55.6M  1 loop /snap/core18/2560
loop9           7:9    0 346.3M  1 loop /snap/gnome-3-38-2004/115
loop10          7:10   0   2.6M  1 loop /snap/gnome-calculator/920
loop11          7:11   0  81.3M  1 loop /snap/gtk-common-themes/1534
loop12          7:12   0   704K  1 loop /snap/gnome-characters/741
loop13          7:13   0 346.3M  1 loop /snap/gnome-3-38-2004/119
loop14          7:14   0     4K  1 loop /snap/bare/5
loop15          7:15   0   696K  1 loop /snap/gnome-logs/115
loop16          7:16   0   219M  1 loop /snap/gnome-3-34-1804/77
loop17          7:17   0 414.4M  1 loop /snap/gnome-42-2204/29
loop18          7:18   0   2.6M  1 loop /snap/gnome-system-monitor/178
loop19          7:19   0   2.5M  1 loop /snap/gnome-calculator/884
loop20          7:20   0  55.6M  1 loop /snap/core18/2566
loop21          7:21   0  91.7M  1 loop /snap/gtk-common-themes/1535
loop22          7:22   0    62M  1 loop /snap/core20/1611
sda             8:0    0 953.9G  0 disk
├─sda1          8:1    0   499M  0 part
├─sda2          8:2    0   100M  0 part /boot/efi
├─sda3          8:3    0    16M  0 part
├─sda4          8:4    0 952.7G  0 part
└─sda5          8:5    0   568M  0 part
sdb             8:16   0 931.5G  0 disk
├─sdb1          8:17   0   512M  0 part
├─sdb2          8:18   0 899.1G  0 part /
└─sdb3          8:19   0  31.9G  0 part
sr0            11:0    1  1024M  0 rom
nvme0n1       259:0    0 230.3G  0 disk
└─nvme0n1p1   259:2    0 230.3G  0 part /media/nvme
hgchoe@ubuntu:~$
```

Check the Mounted Partition (2 / 2)

- **Type “df -h”, press ENTER -> check the mounted partition on the storage list**

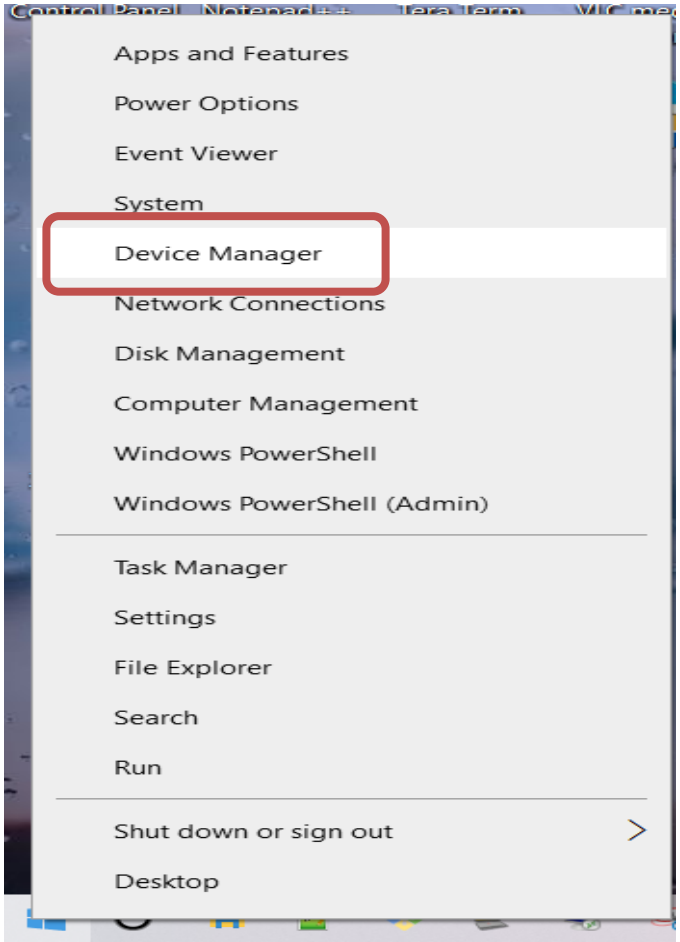
```
hgchoe@ubuntu:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev             16G     0   16G   0% /dev
tmpfs           3.2G  3.6M  3.2G   1% /run
/dev/sdb2       884G  452G  388G  54% /
tmpfs            16G     0   16G   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs            16G     0   16G   0% /sys/fs/cgroup
/dev/loop0       71M   71M     0 100% /snap/core22/275
/dev/loop1      640K  640K     0 100% /snap/gnome-logs/112
/dev/loop2      1.5M  1.5M     0 100% /snap/gnome-system-monitor/181
/dev/loop3      512K  512K     0 100% /snap/gnome-characters/781
/dev/loop4       48M   48M     0 100% /snap/snapd/17029
/dev/loop6      219M  219M     0 100% /snap/gnome-3-34-1804/72
/dev/loop5       48M   48M     0 100% /snap/snapd/17336
/dev/loop8       56M   56M     0 100% /snap/core18/2560
/dev/loop7       64M   64M     0 100% /snap/core20/1623
/dev/loop10     2.7M  2.7M     0 100% /snap/gnome-calculator/920
/dev/loop12     768K  768K     0 100% /snap/gnome-characters/741
/dev/loop15     768K  768K     0 100% /snap/gnome-logs/115
/dev/loop11      82M   82M     0 100% /snap/gtk-common-themes/1534
/dev/loop16     219M  219M     0 100% /snap/gnome-3-34-1804/77
/dev/loop13     347M  347M     0 100% /snap/gnome-3-38-2004/119
/dev/loop14     128K  128K     0 100% /snap/bare/5
/dev/loop17     415M  415M     0 100% /snap/gnome-42-2204/29
/dev/loop9      347M  347M     0 100% /snap/gnome-3-38-2004/115
/dev/loop18     2.7M  2.7M     0 100% /snap/gnome-system-monitor/178
/dev/loop19     2.5M  2.5M     0 100% /snap/gnome-calculator/884
/dev/loop20      56M   56M     0 100% /snap/core18/2566
/dev/loop21      92M   92M     0 100% /snap/gtk-common-themes/1535
/dev/loop22      62M   62M     0 100% /snap/core20/1611
/dev/sda2        96M   52M   45M  55% /boot/efi
tmpfs           3.2G   16K  3.2G   1% /run/user/121
tmpfs           3.2G     0  3.2G   0% /run/user/1000
/dev/nvme0n1p1  226G   28K  215G   1% /media/nvme
hgchoe@ubuntu:~$
```

Operating Daisy+ OpenSSD (Windows)

1. **Check device recognition**
2. **Create a partition**
3. **Format the partition**

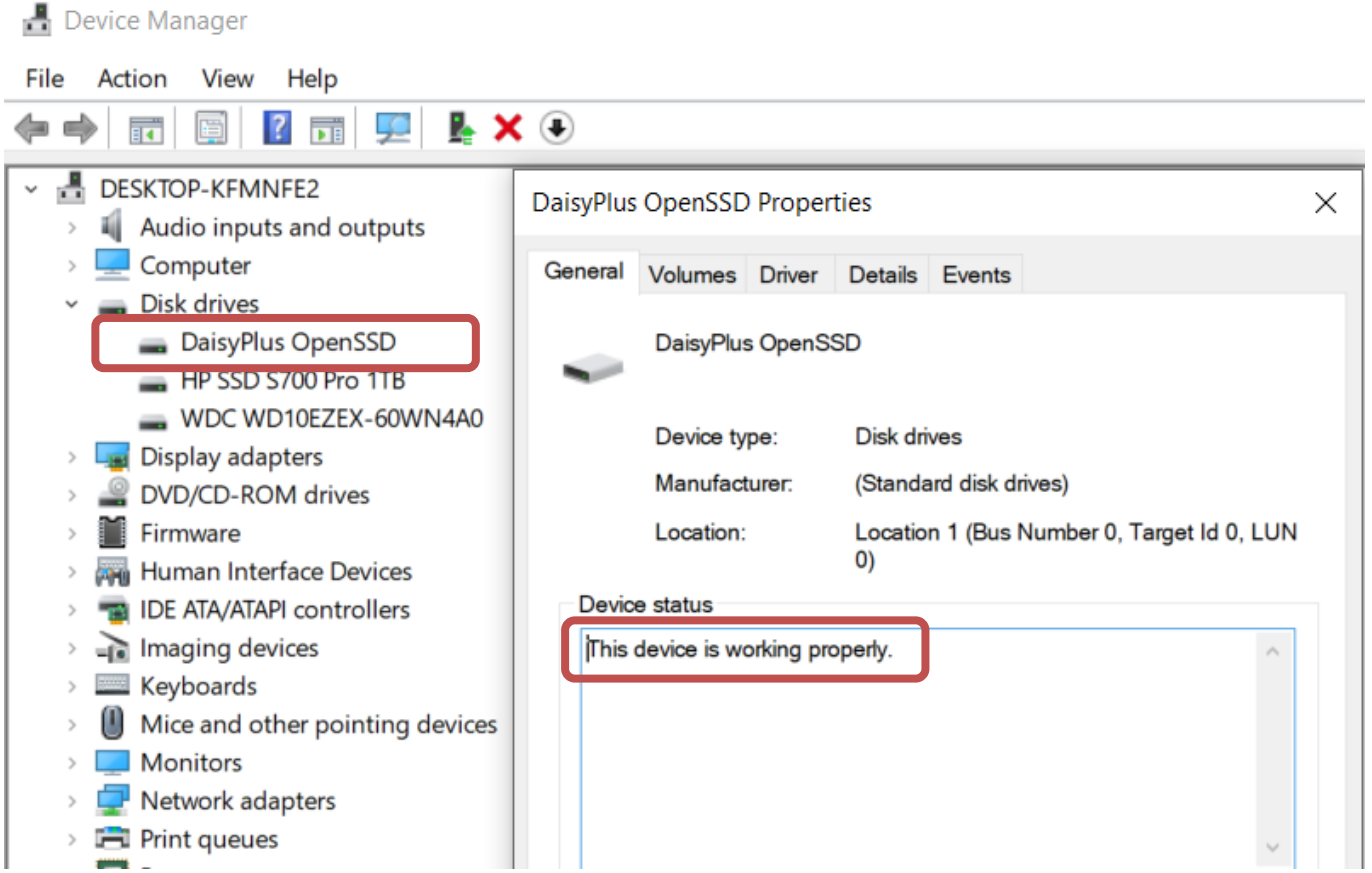
Check Device Recognition (1 / 2)

- **Right-click “Start” menu → click “Device Manager”**



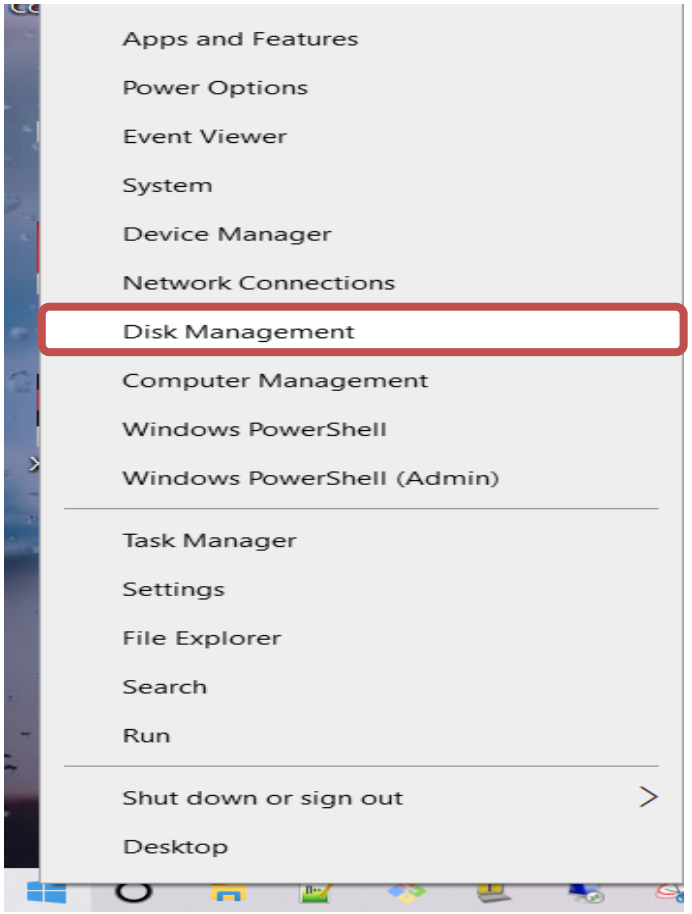
Check Device Recognition (2 / 2)

- **Disk drives → double-click “DaisyPlus OpenSSD”**



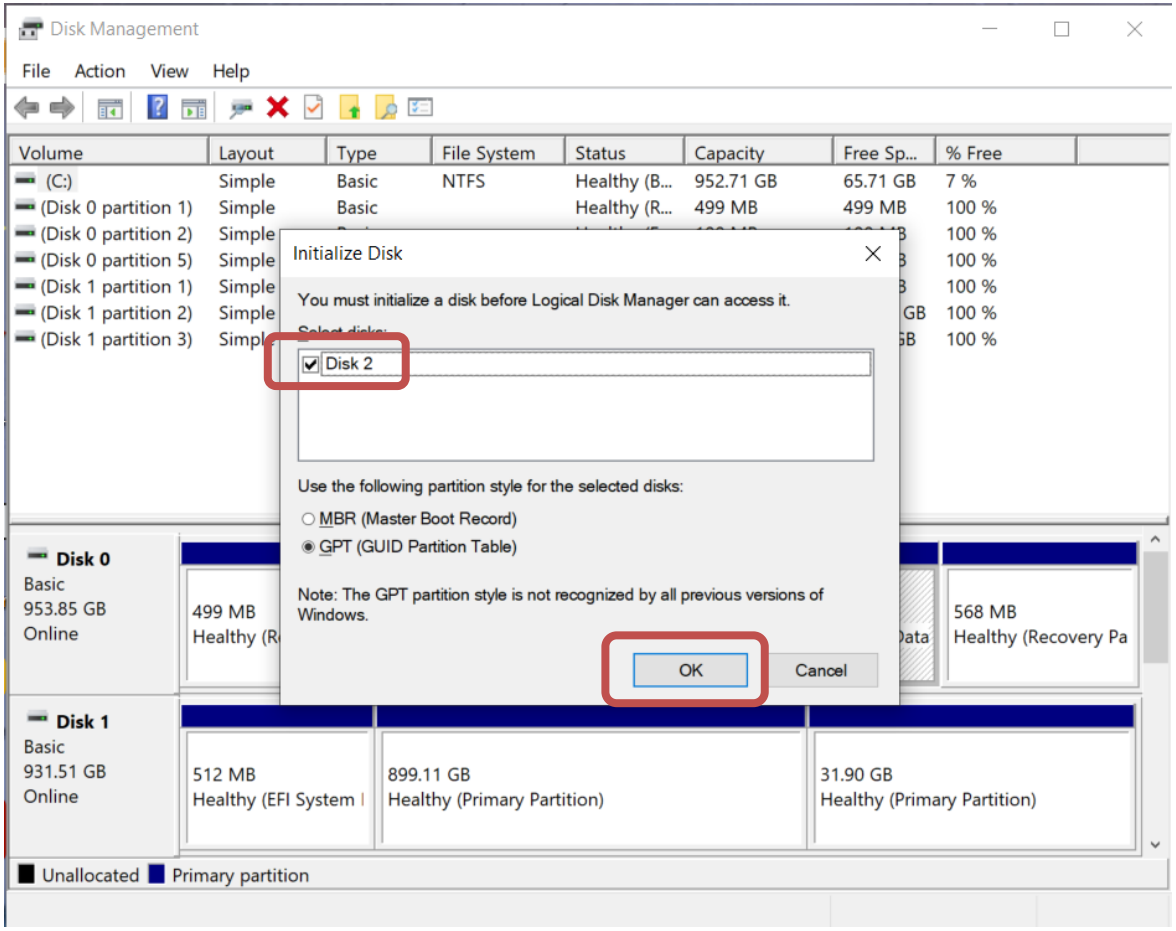
Create a Partition (1 / 4)

- **Right-click “Start” menu → click “Disk Management”**



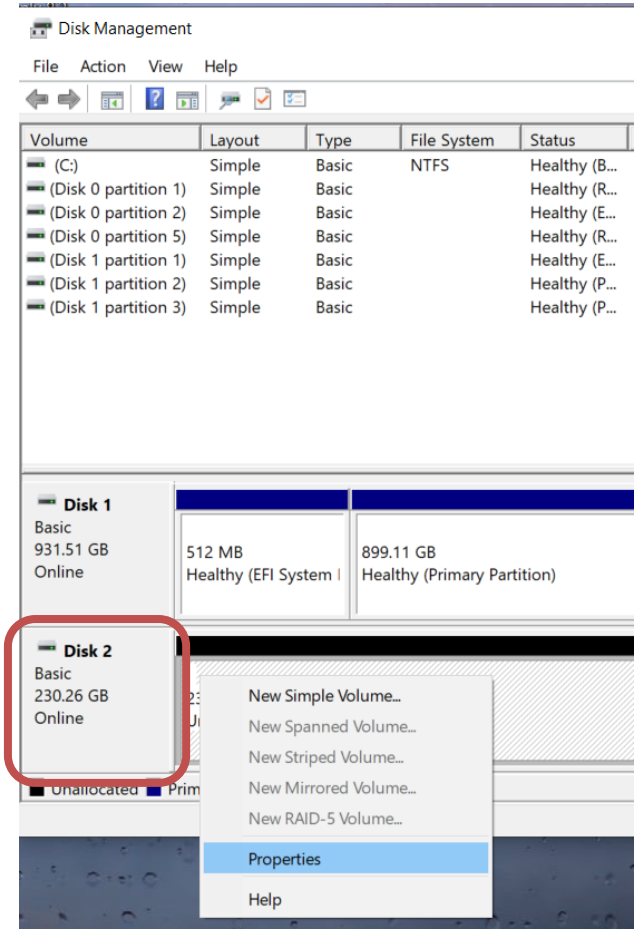
Create a Partition (2 / 4)

- **Click “OK” to confirm disk initialization**



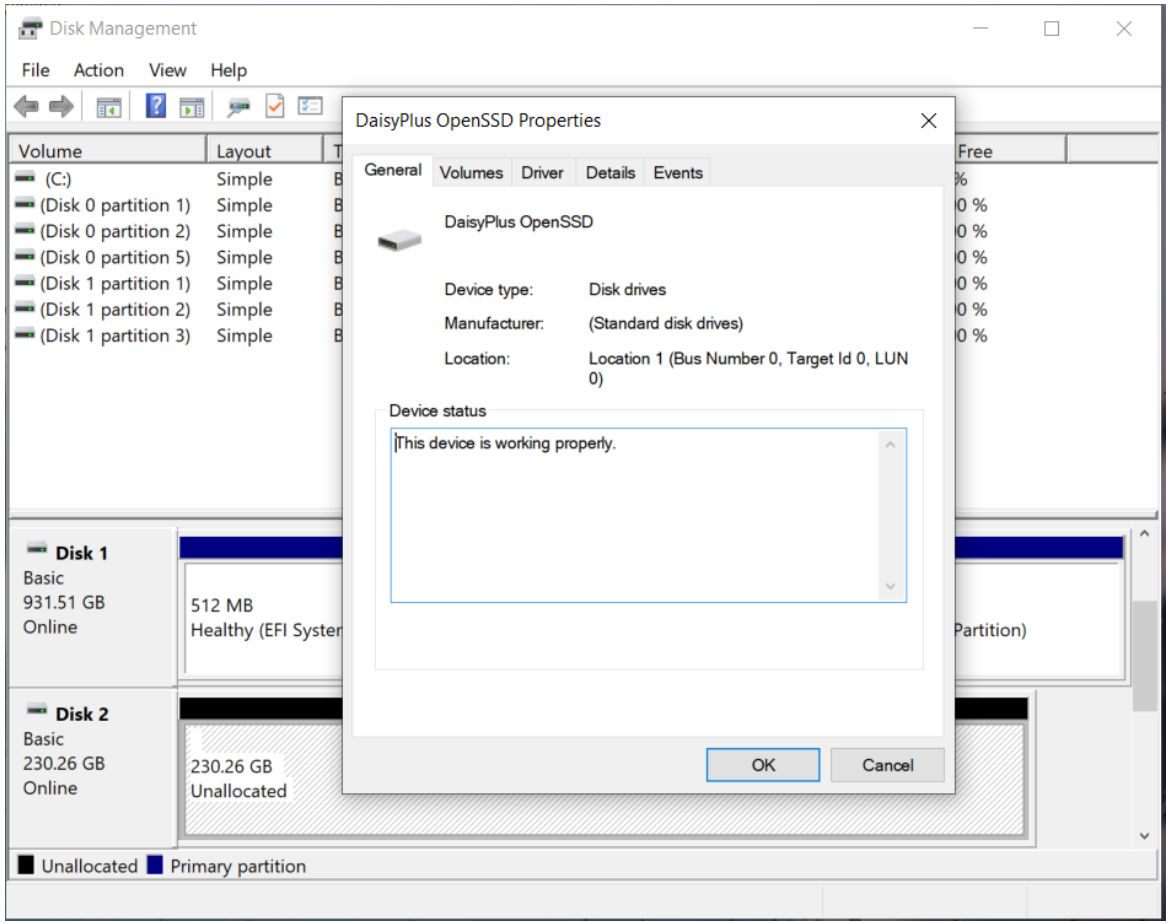
Create a Partition (3 / 4)

- **Click right mouse button on “Disk 2” which was shown in 2nd step → click “Properties”**



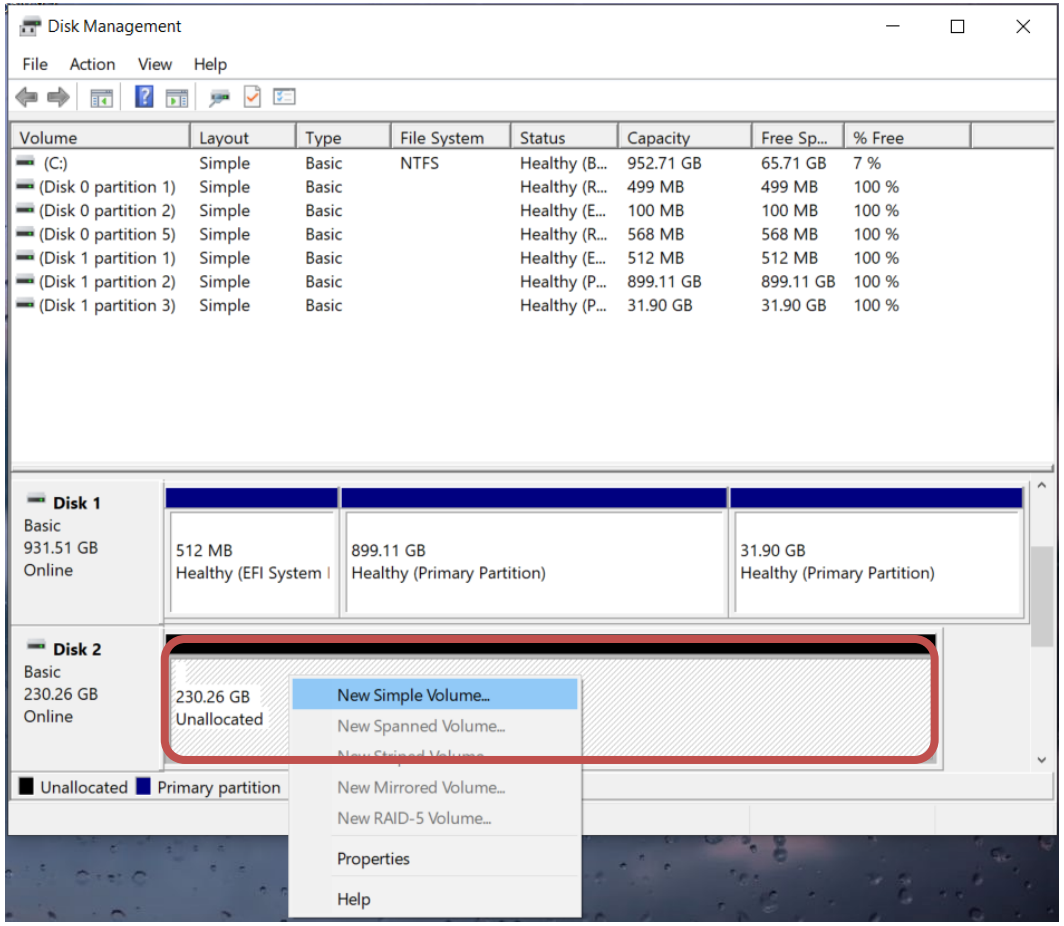
Create a Partition (4 / 4)

- **Make sure that the “Disk 2” is DaisyPlus OpenSSD before you proceed to the next step**



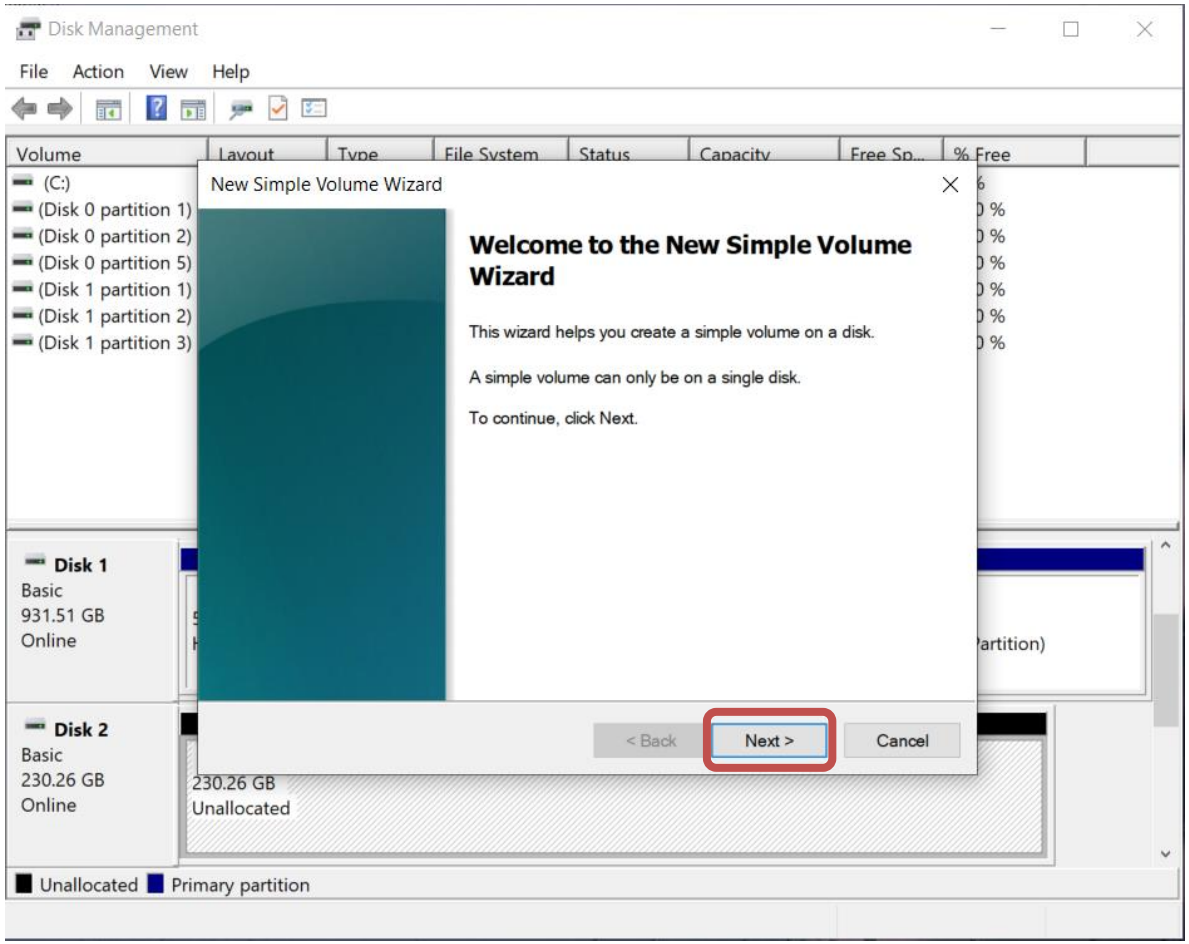
Format the Partition (1 / 8)

- **Click right mouse button on the right part of “Disk 2” → click “New Simple Volume”**



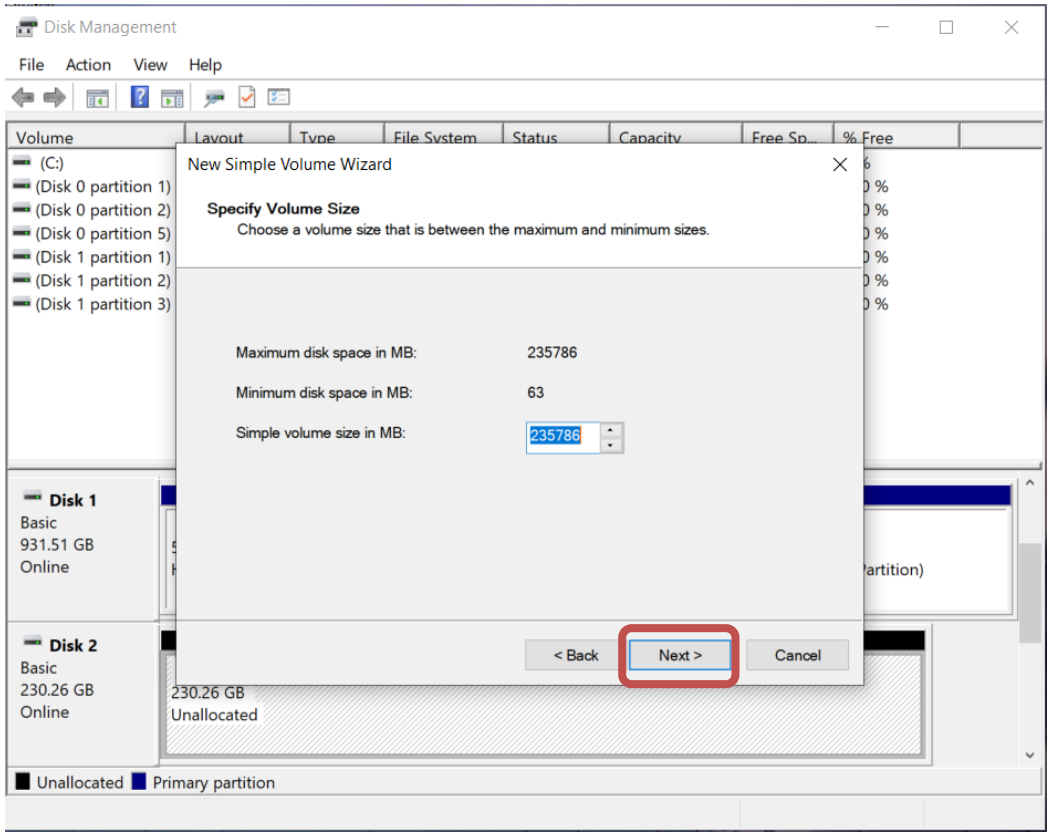
Format the Partition (2 / 8)

- **Click “Next”**



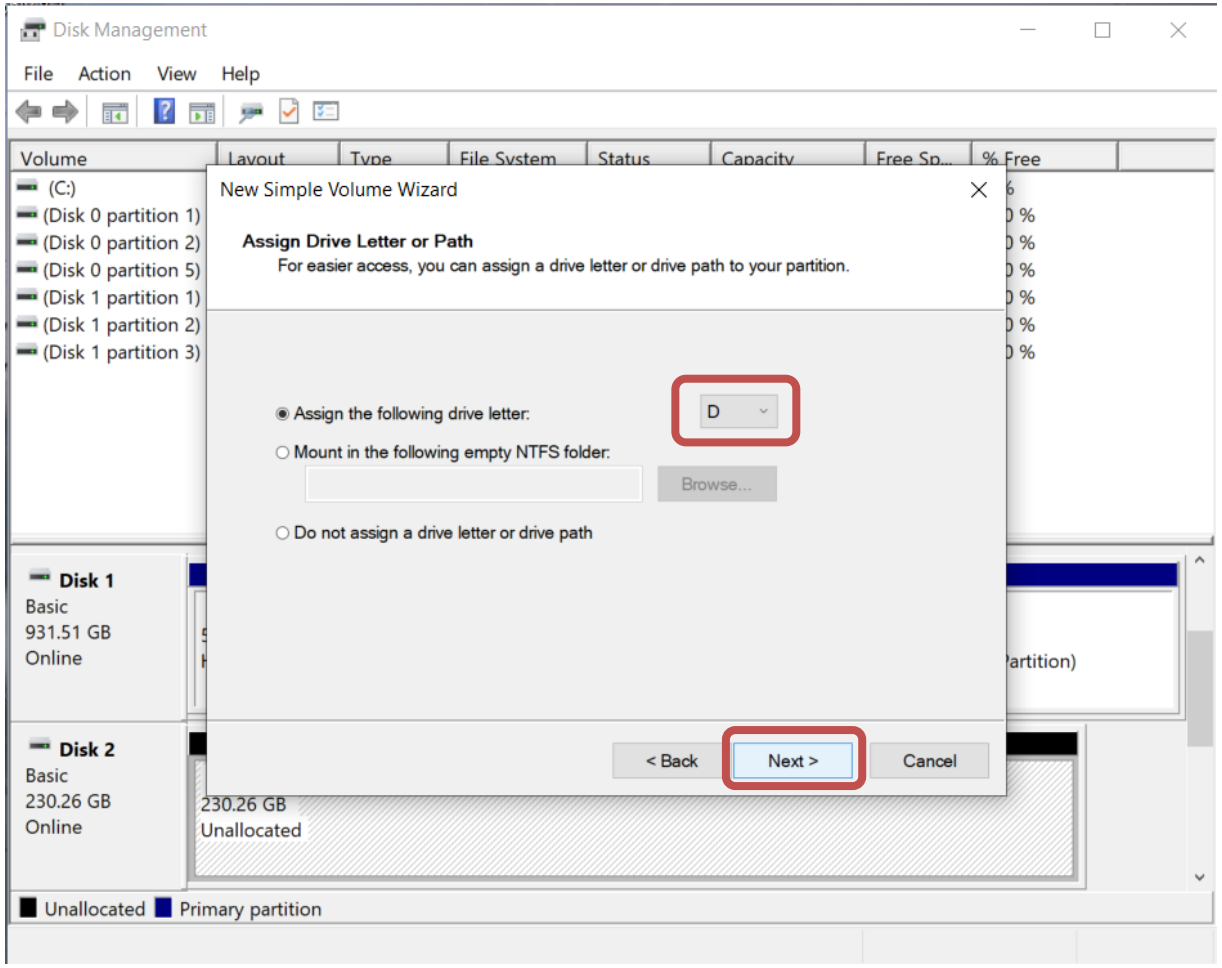
Format the Partition (3 / 8)

- **Click “Next”**



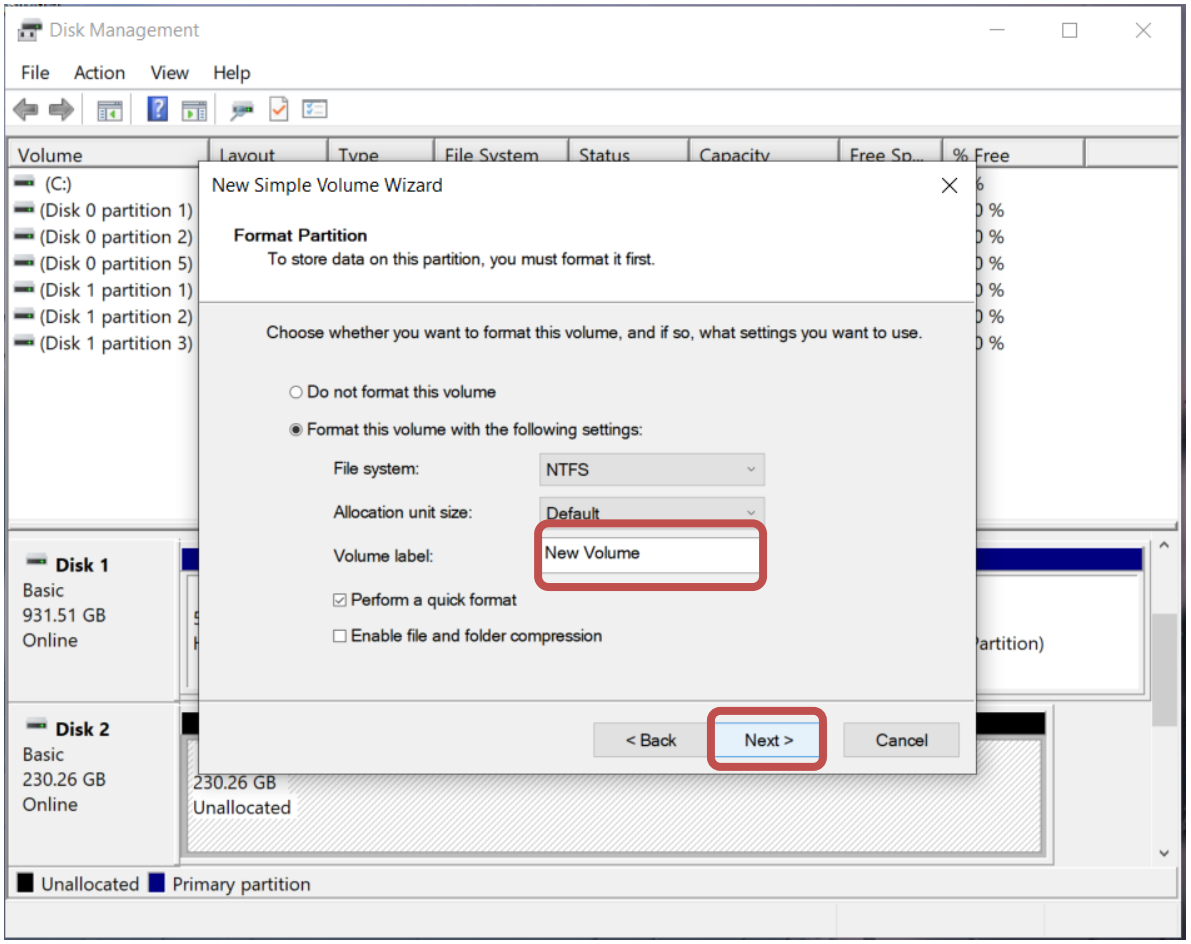
Format the Partition (4 / 8)

- **Select desired drive letter → Click “Next”**



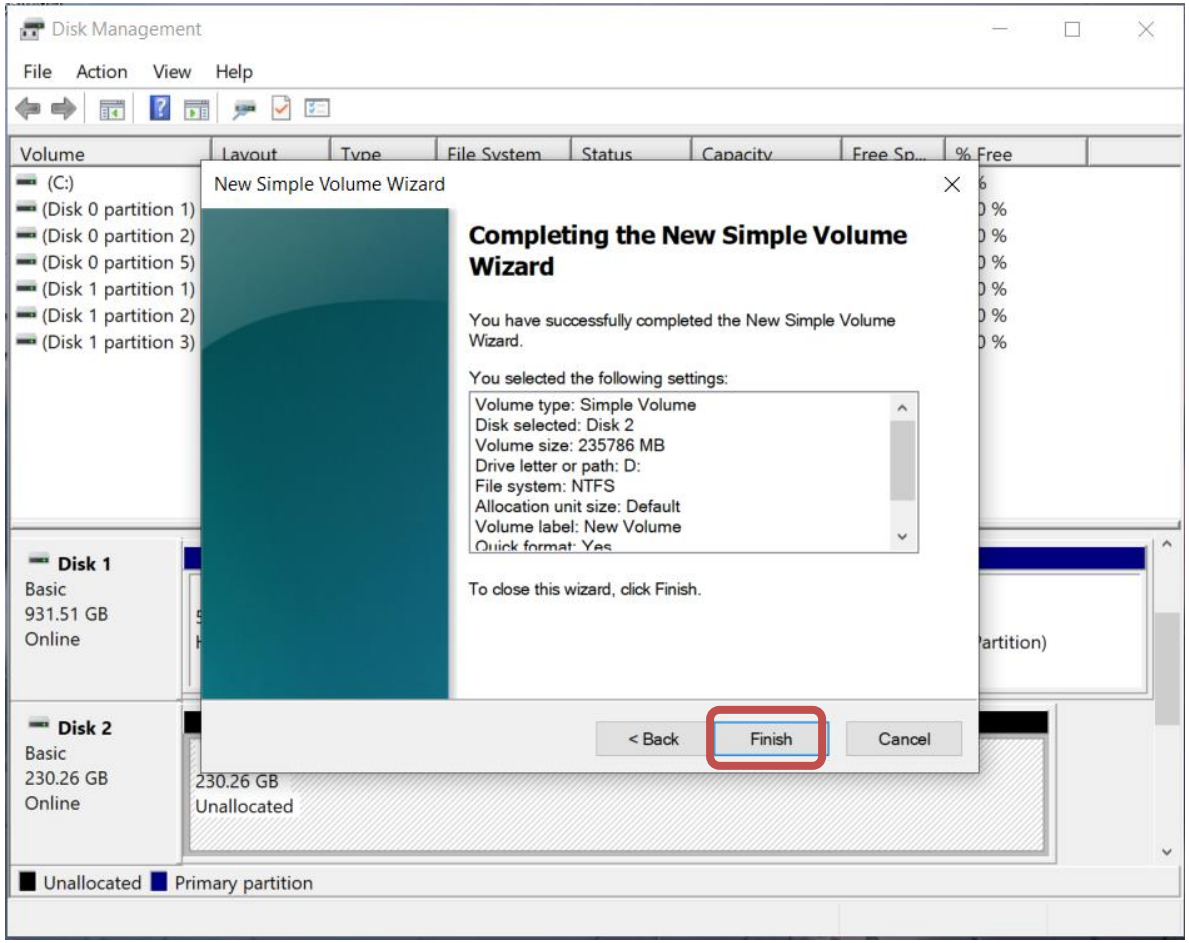
Format the Partition (5 / 8)

- **Type desired volume label → Click “Next”**



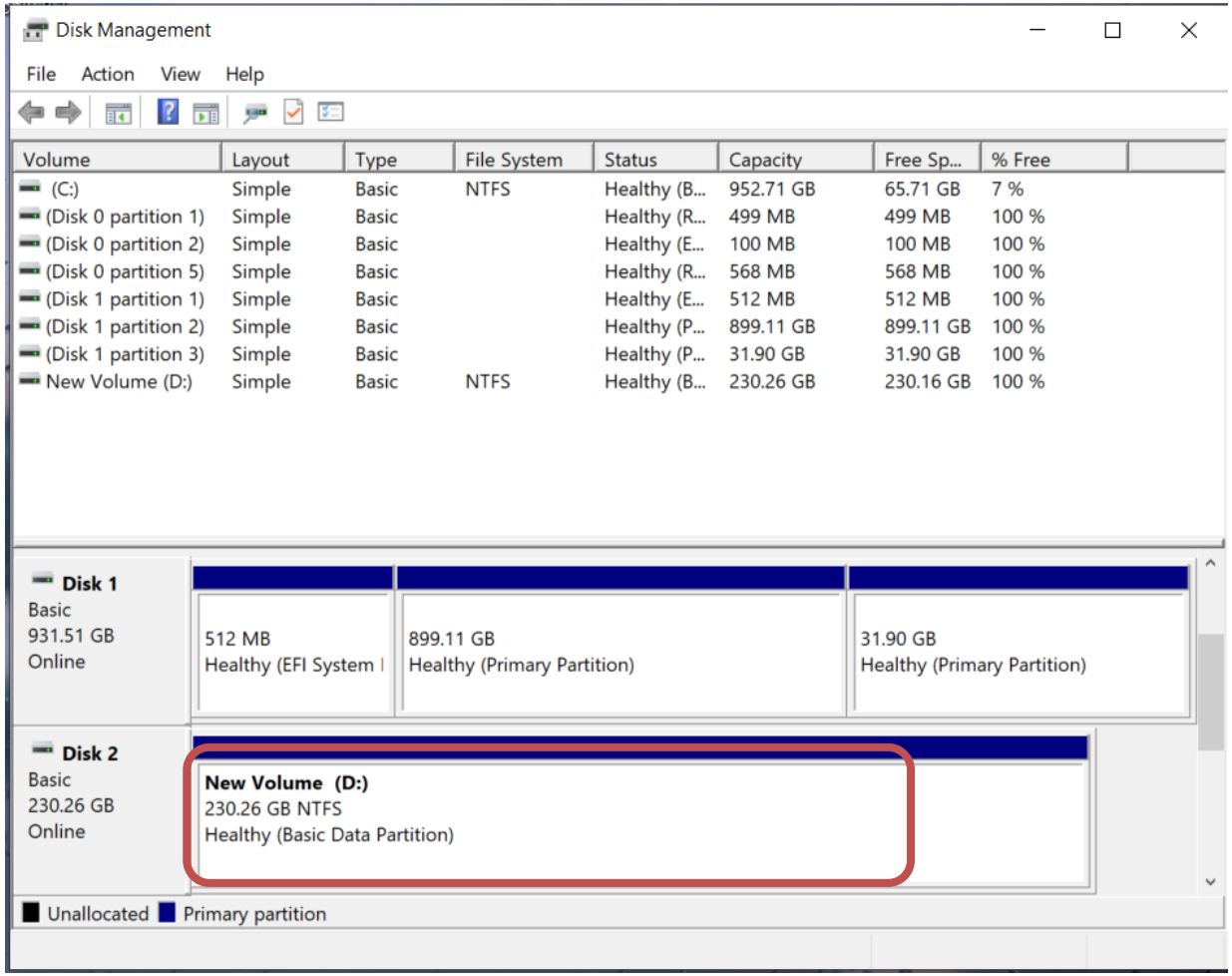
Format the Partition (6 / 8)

- **Click “Finish”**



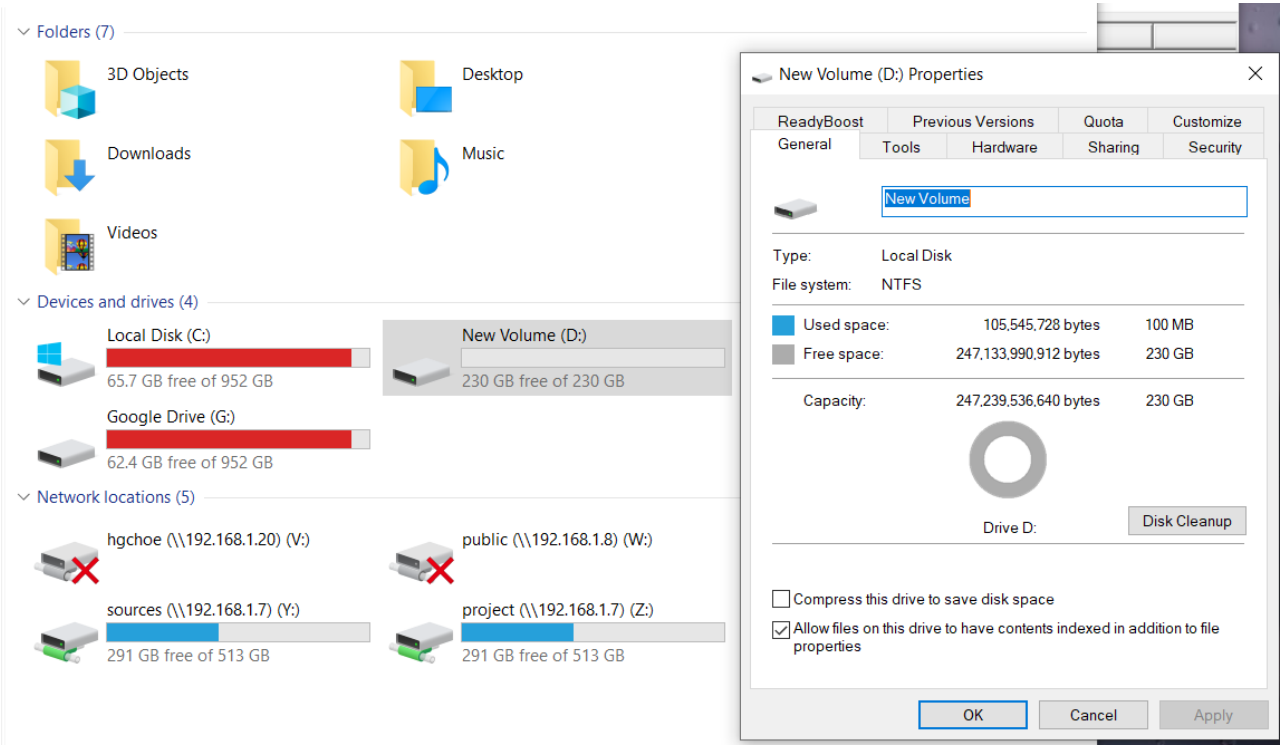
Format the Partition (7 / 8)

- **Formatting is now finished**



Format the Partition (8 / 8)

- **Now you can find the formatted Daisy+ OpenSSD at “This PC”**

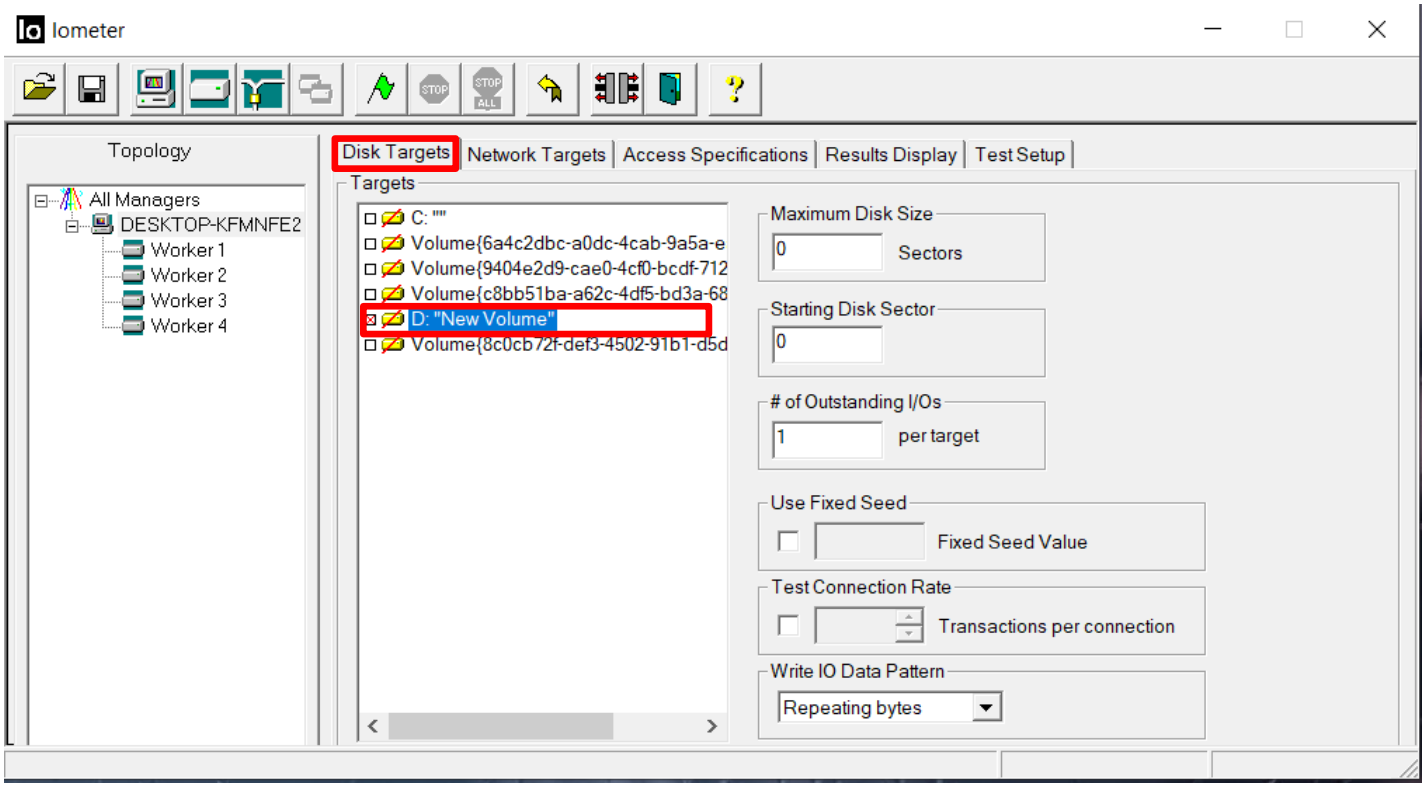


Evaluating Daisy+ OpenSSD Performance

1. **Install benchmark application (Iometer)**
2. **Disconnect workers except one worker**
3. **Generate a access specification**
4. **Set the sufficient number of outstanding I/Os**
5. **Assign a access specification**
6. **Run an evaluation**
7. **Check evaluation results**

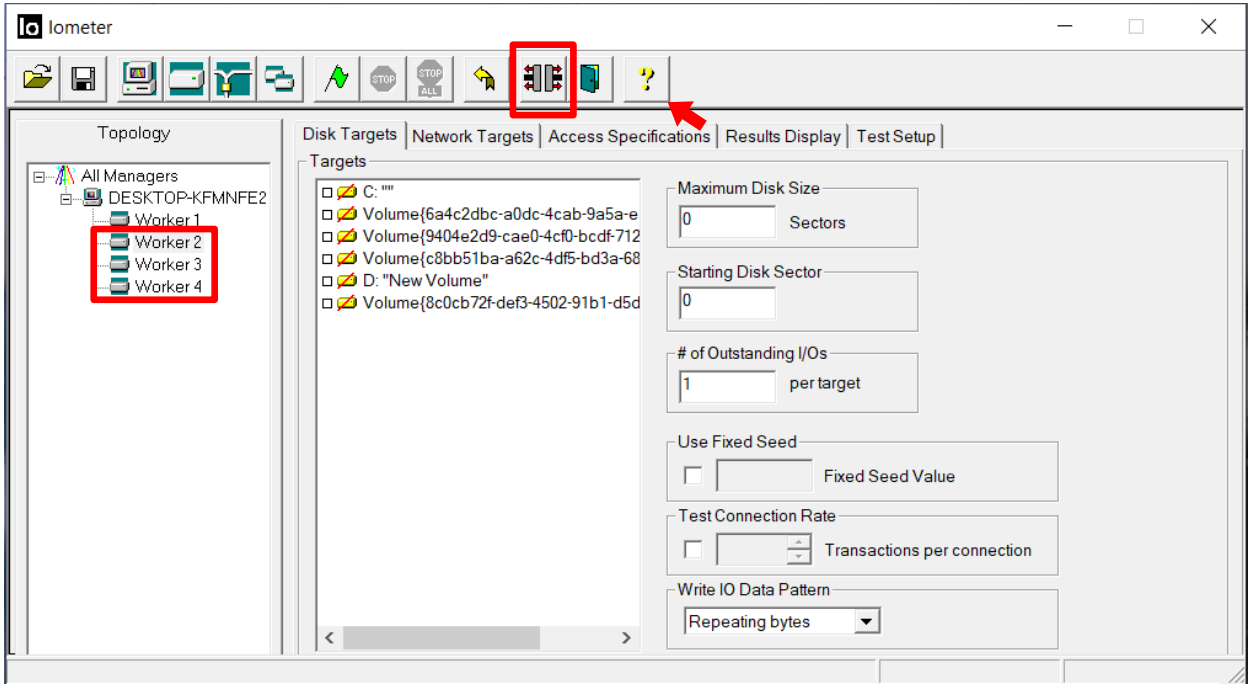
Install Benchmark Application

- **Iometer 1.1.0 (http://www.iometer.org/doc/downloads.html)**
  - Daisy+ OpenSSD is recognized as NVMe storage device



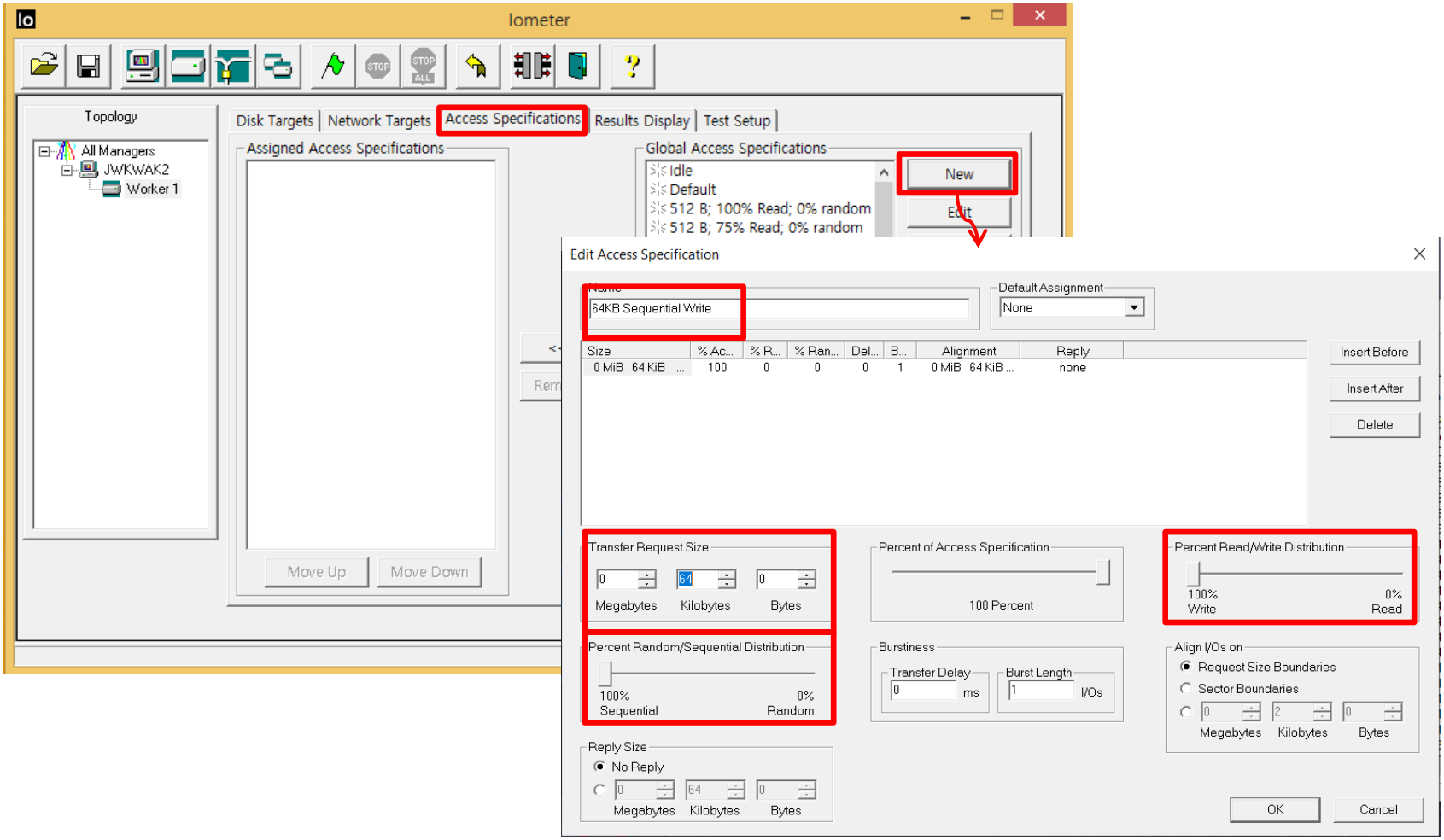
Disconnect workers except one worker

- **Avoid Workers having a same access specifications**
  - Workers can access the same logical address almost the same time
    - Increase the data buffer hit ratio
  - Performance can be measured higher than real performance



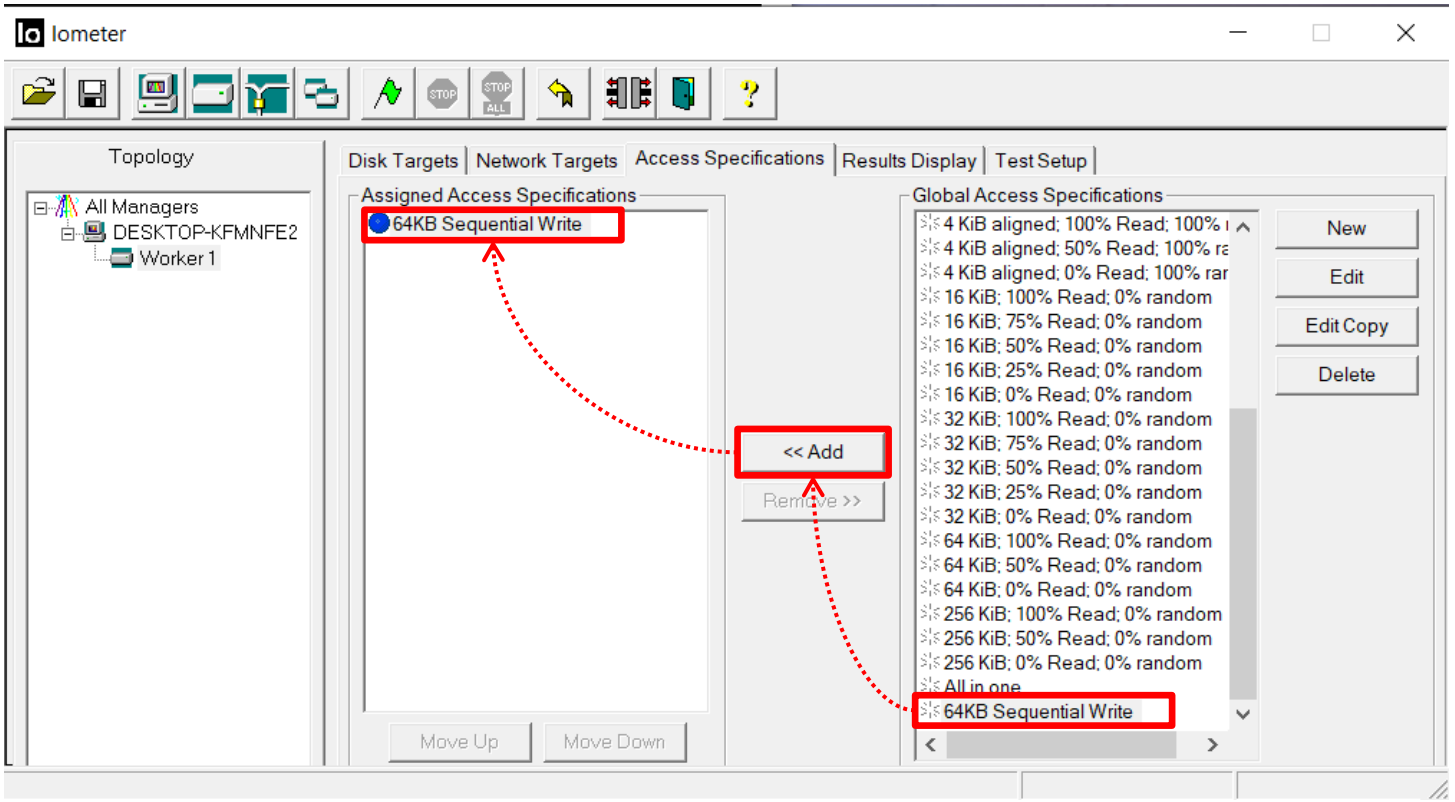
Generate a access specification

- **User can define a access specification**



Assign a access specification

- **Select a desired access specification and click “Add” button**

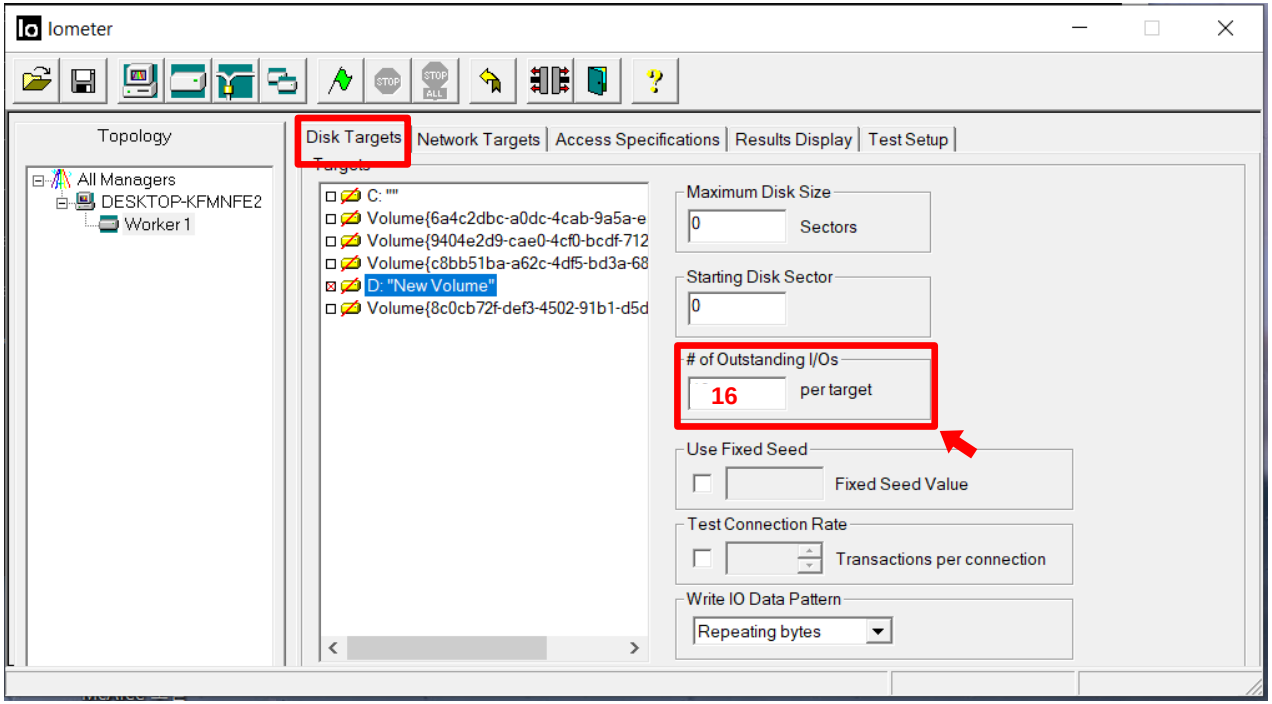


Set the Sufficient Number of Outstanding I/Os

- **X channel – Y way flash array needs "X * Y" outstanding flash requests at least for utilizing multi channel/way parallelism**
  - In case of a Daisy+ OpenSSD configuration (4 channel – 8 way, 16KB page size), "64KB sequential write" access specification needs 8 outstanding I/Os at least

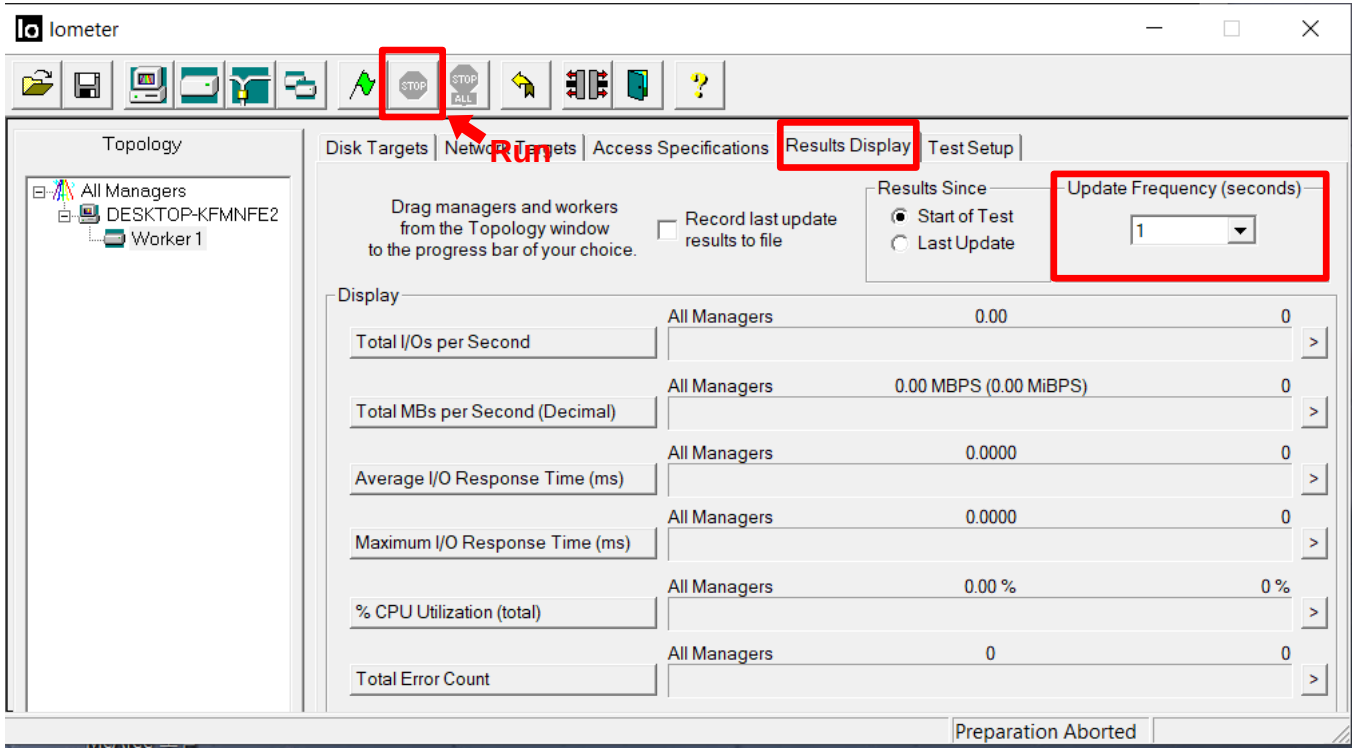
32 (64KB/16KB * 8) outstanding flash requests

  - Recommend the environment generating 2 * X * Y outstanding flash requests



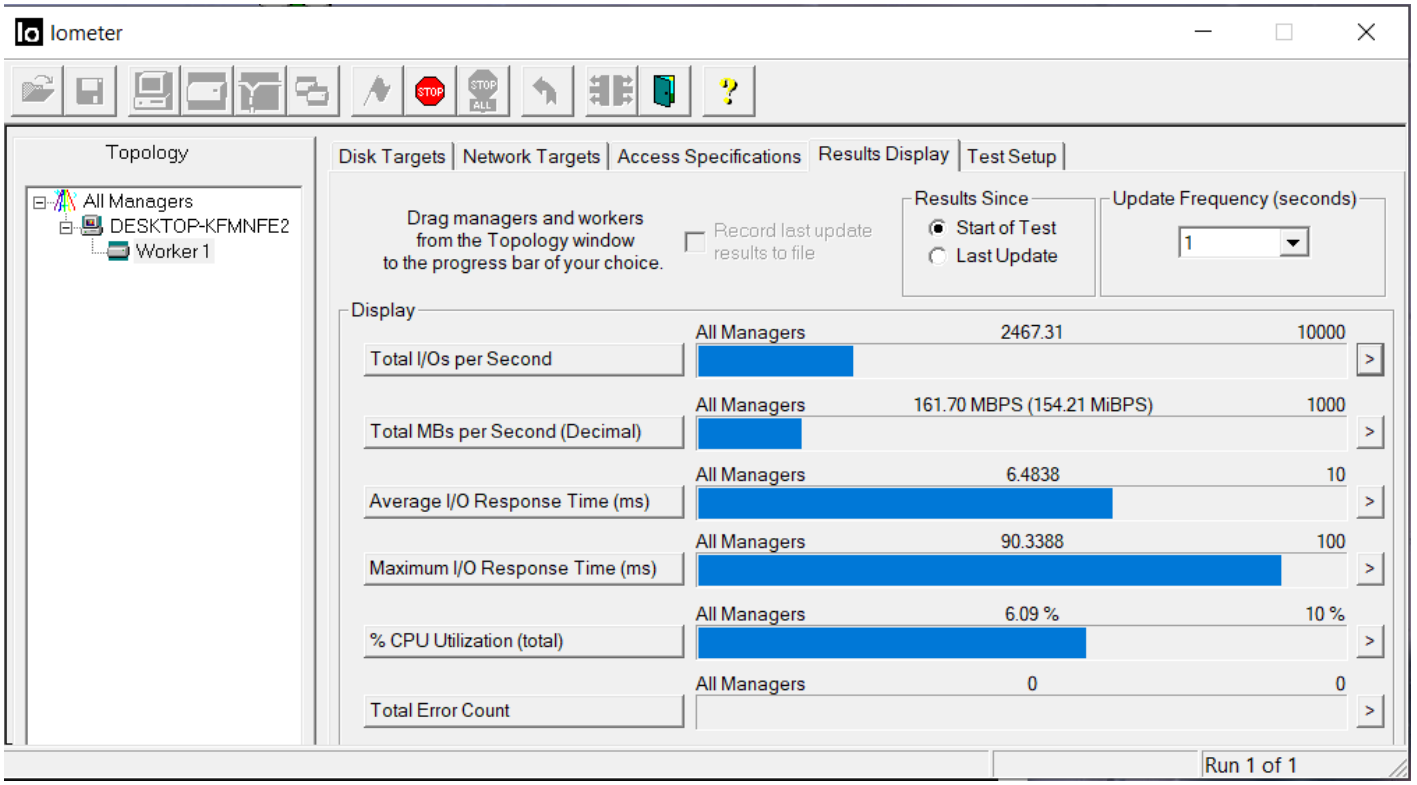
Run an Evaluation

- **Set the update frequency and click “Run” button**



Check evaluation results

- **“Results display” tab shows the performance evaluation results**
  - IOPs, throughput, average/maximum response time



Evaluation Guideline

- **Perform pre-fill process before the read performance evaluation**
  - There are no mapping information for unwritten data
- **Set the number of outstanding I/Os equal or less than 256**
  - Unknown problem of host interface
- **Set the write request size equal or larger than the page size**
  - Read-modify-write process can degrade the performance
    - In case of “4KB random write”, IOPs can be decreased as the experiment progresses

Thank You