



Cosmos+ OpenSSD Tutorial 2017

Prof. Yong Ho Song

Department of Electronic Engineering, Hanyang University





OpenSSD Introduction



Need a SSD platform

- to develop a new firmware algorithm
- to explore hardware architecture and organization

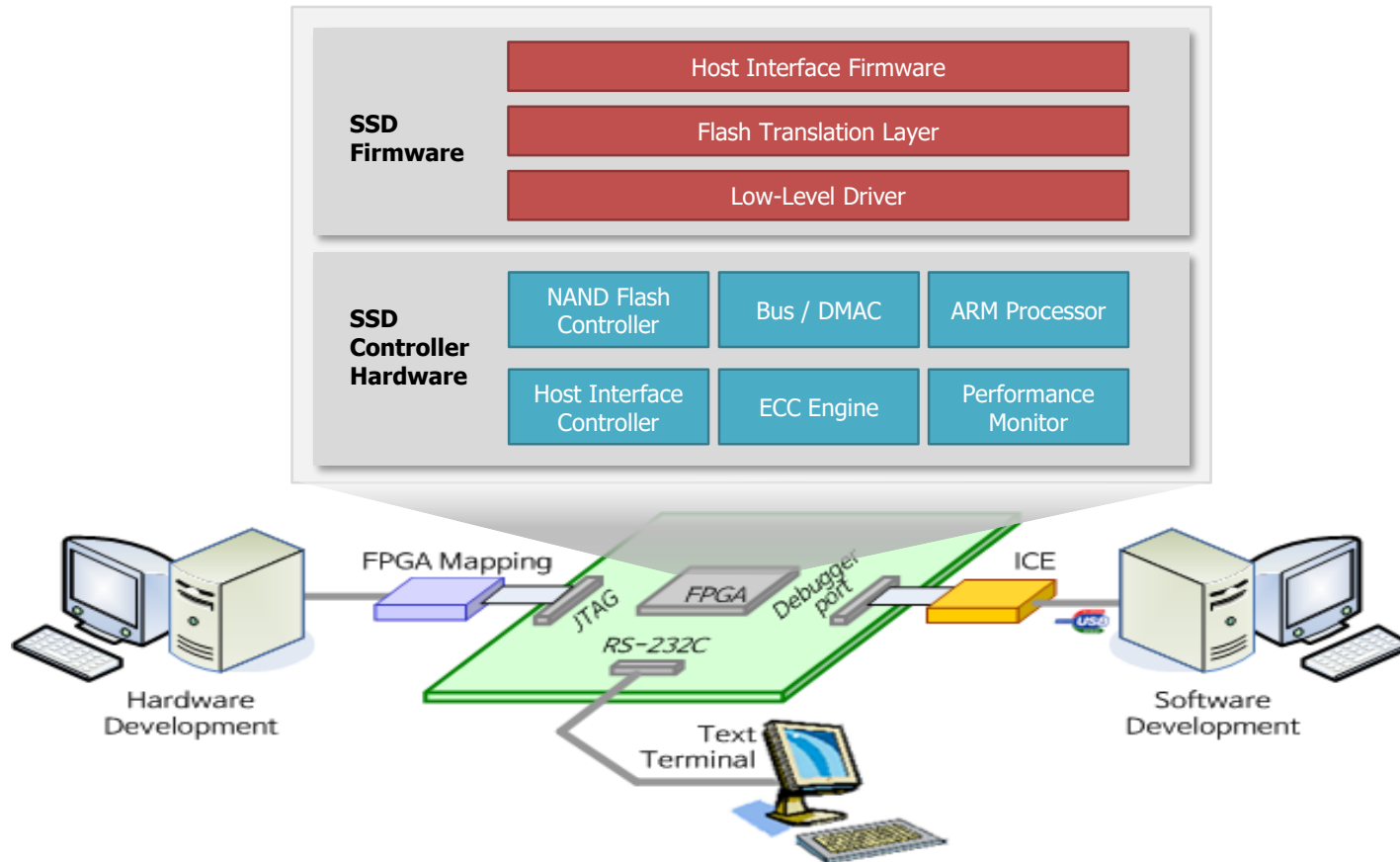


Use a commercial product as a platform?

- 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)

■ Cosmos/Cosmos+ 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

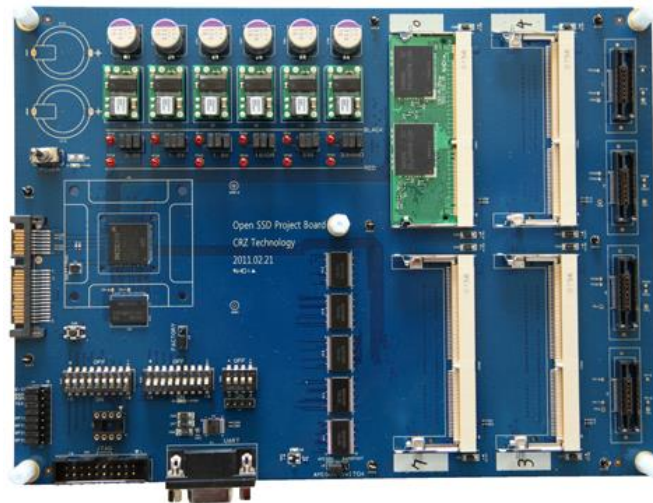
1st OpenSSD (Indilinx)

■ Jasmine OpenSSD (2011)

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

**Barefoot
Controller
SoC**

**SATA-2
Interface**

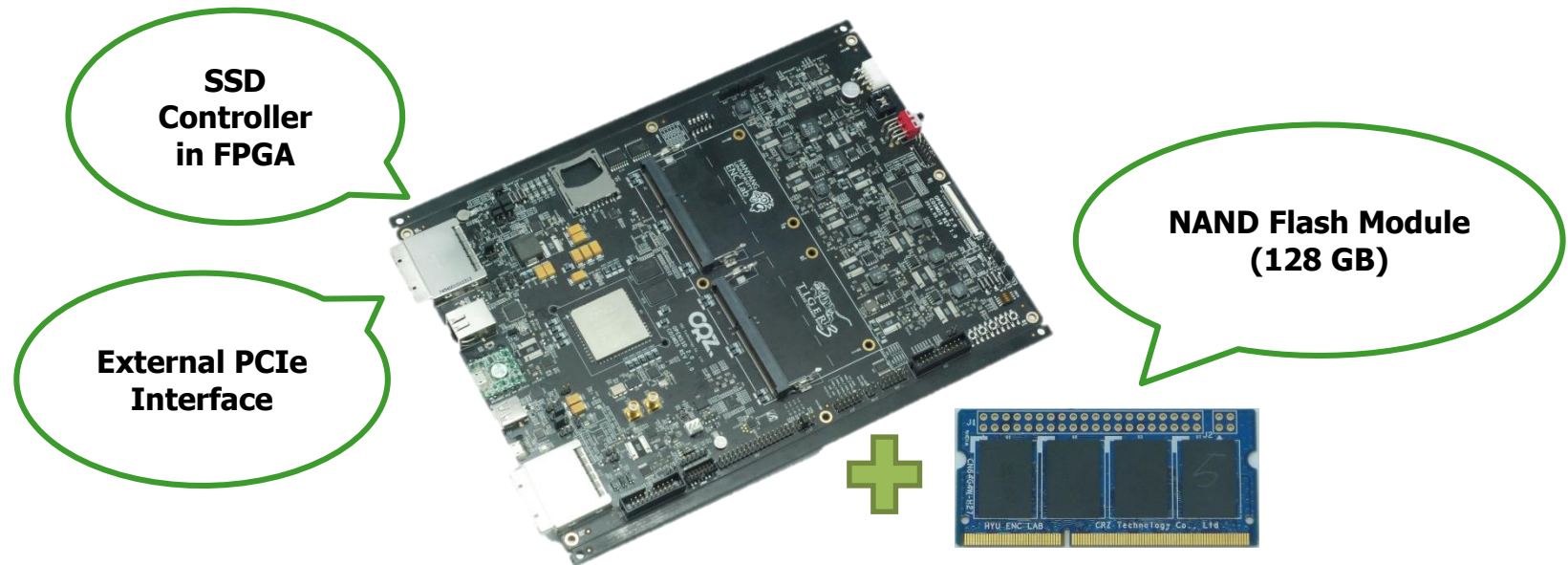


**NAND Flash
Memory
(32GB/module)**

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)

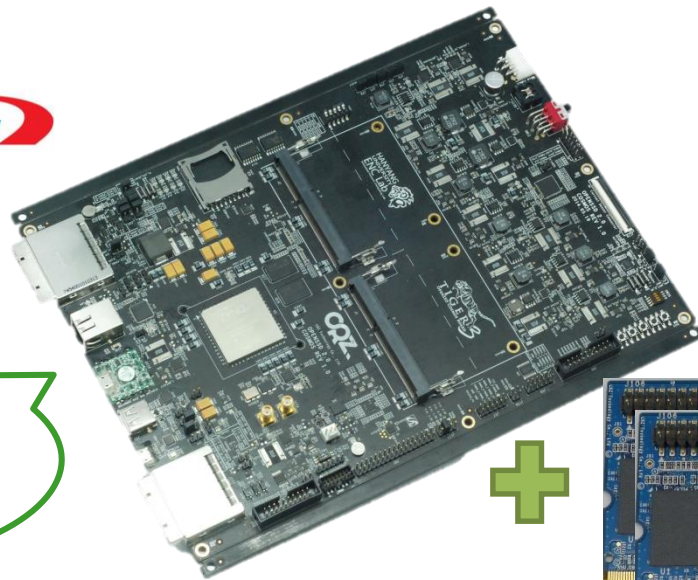


3rd OpenSSD (Hanyang University)

■ 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

nvm
EXPRESS

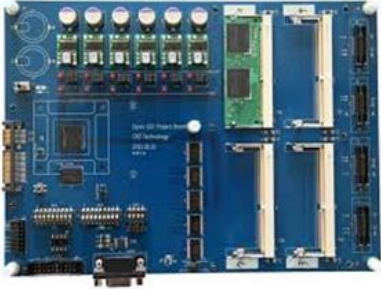



Same platform with
Cosmos OpenSSD



NAND Flash Modules
(1 TB/module)

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 |

OpenSSD Project Homepage

The OpenSSD Project

www.openssd.io/

OPENSSD

OPEN-SOURCE SOLID-STATE DRIVE PROJECT FOR RESEARCH AND EDUCATION

HOME PUBLICATIONS RESOURCES FORUM PEOPLE RELATED LINKS

IT'S OPEN-SOURCE

All materials including documents, firmwares, hardware RTL sources, and platform board schematics are fully accessible to the public.

IT'S MODULAR

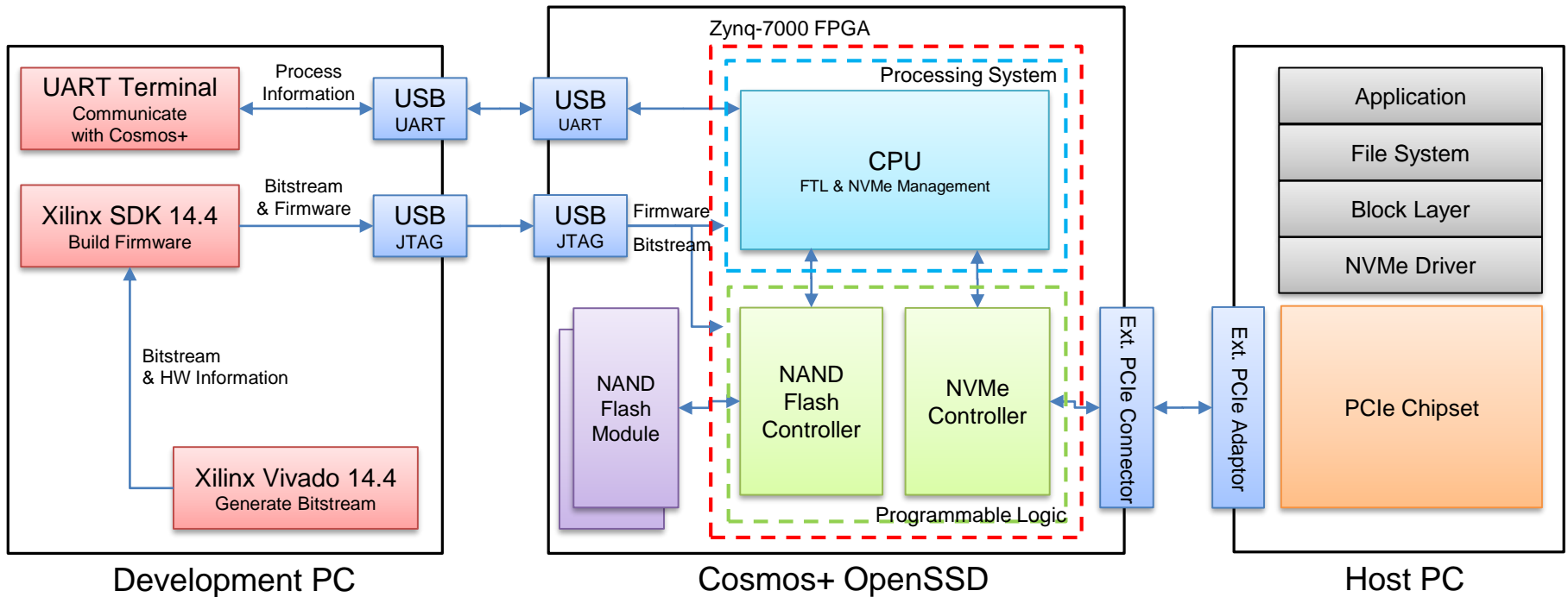
All software and hardware designs consist of many replaceable modules and offer flexibility in their re-design.

IT'S REAL

A real hardware board, called Cosmos OpenSSD platform, is a part of Cosmos OpenSSD system. It operates as a real solid-state drive.

<http://www.openssd.io>

Cosmos+ OpenSSD Overview



Cosmos+ OpenSSD Environment

■ 1 Development PC

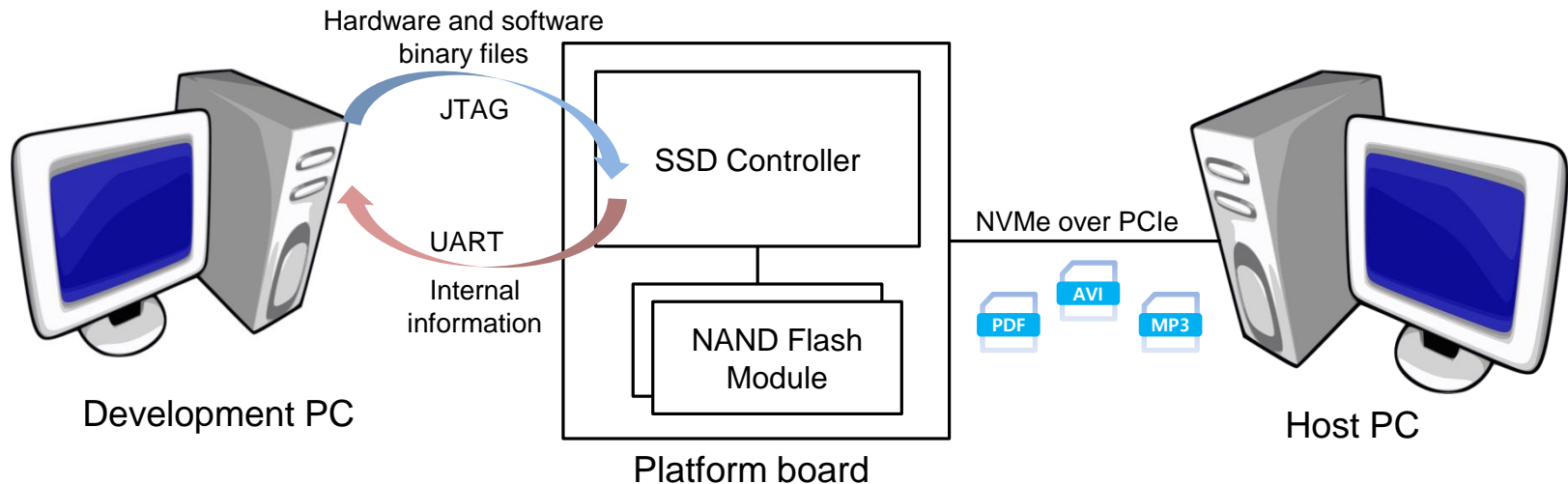
- Downloading hardware/software design (JTAG)
- Monitoring Cosmos+ OpenSSD internals (UART)

■ 1 Host PC

- Executing applications such as a benchmark (PCIe)

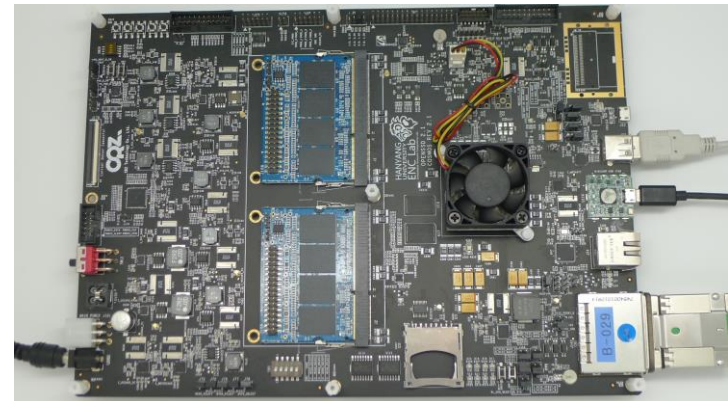
■ 1 Platform board with 1+ NAND flash modules installed

- Working as a storage device to the host PC

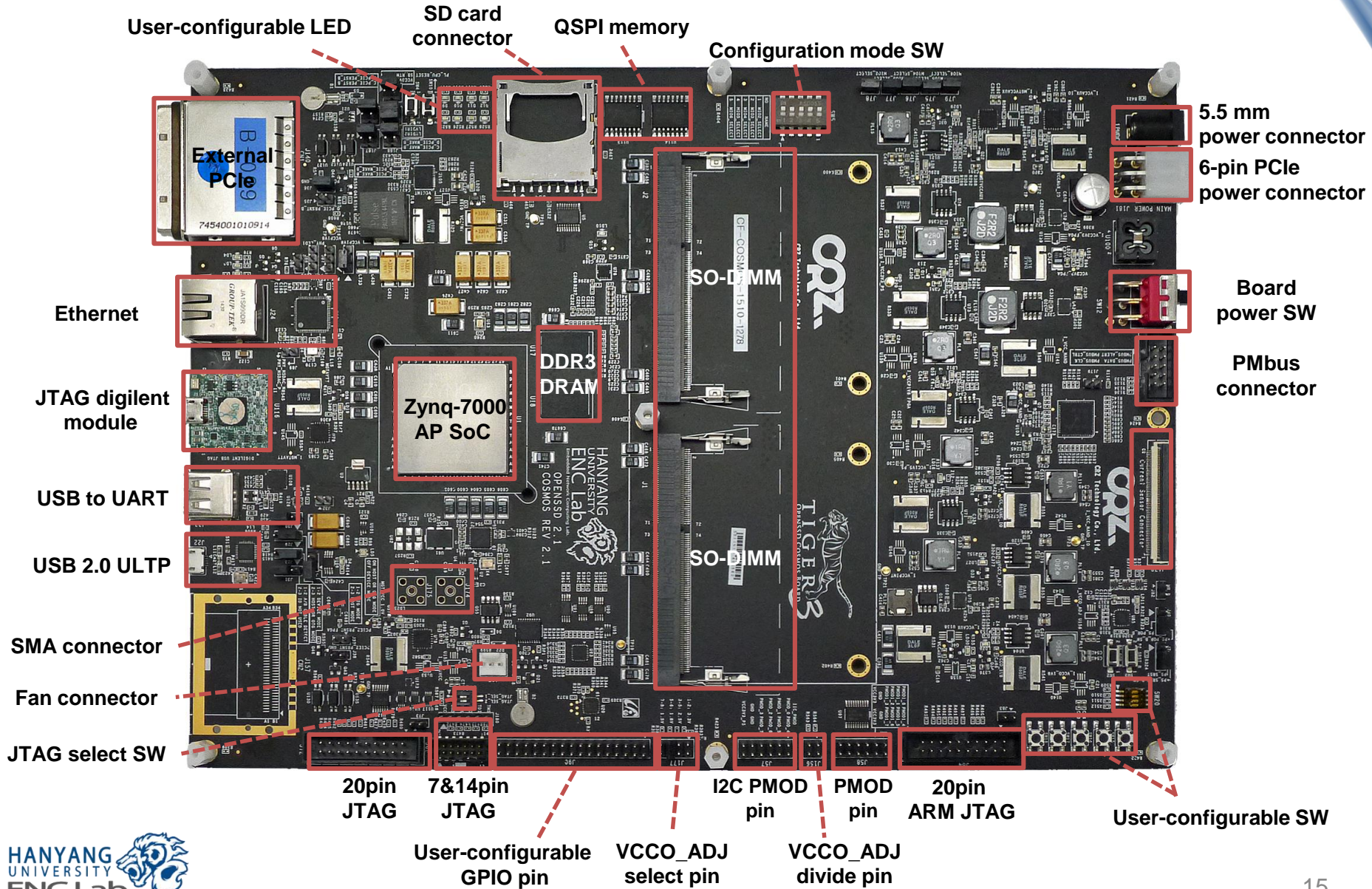


Hardware Components

- **Cosmos+ OpenSSD platform board**
 - Consists of a Zynq FPGA and other peripherals
- **NAND flash modules**
 - Configured as multi-channel and multi-way flash array
 - Inserted into Cosmos+ OpenSSD platform board
- **External PCIe adapter and cable**
 - Connected with host PC
- **USB cables for JTAG and UART**
 - Connected with development PC
- **Power cable and adapter**
 - 12V supply voltage



Cosmos+ OpenSSD Platform Board



Primary Details

| | | |
|--------------------|-------------------------|--|
| FPGA | | Xilinx Zynq-7000 AP SoC (XC7Z045-FFG900-3) |
| Logic cells | | 350K (~ 5.2M ASIC gates) |
| CPU | Type | Dual-Core ARM Cortex™- A9 |
| | Clock frequency | Up to 1000 MHz |
| Storage | Total capacity | Up to 2 TB (MLC) |
| | Organization | Up to 8-channel 8-way |
| DRAM | Device interface | DDR3 1066 |
| | Total capacity | 1 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-7000 FPGA Architecture

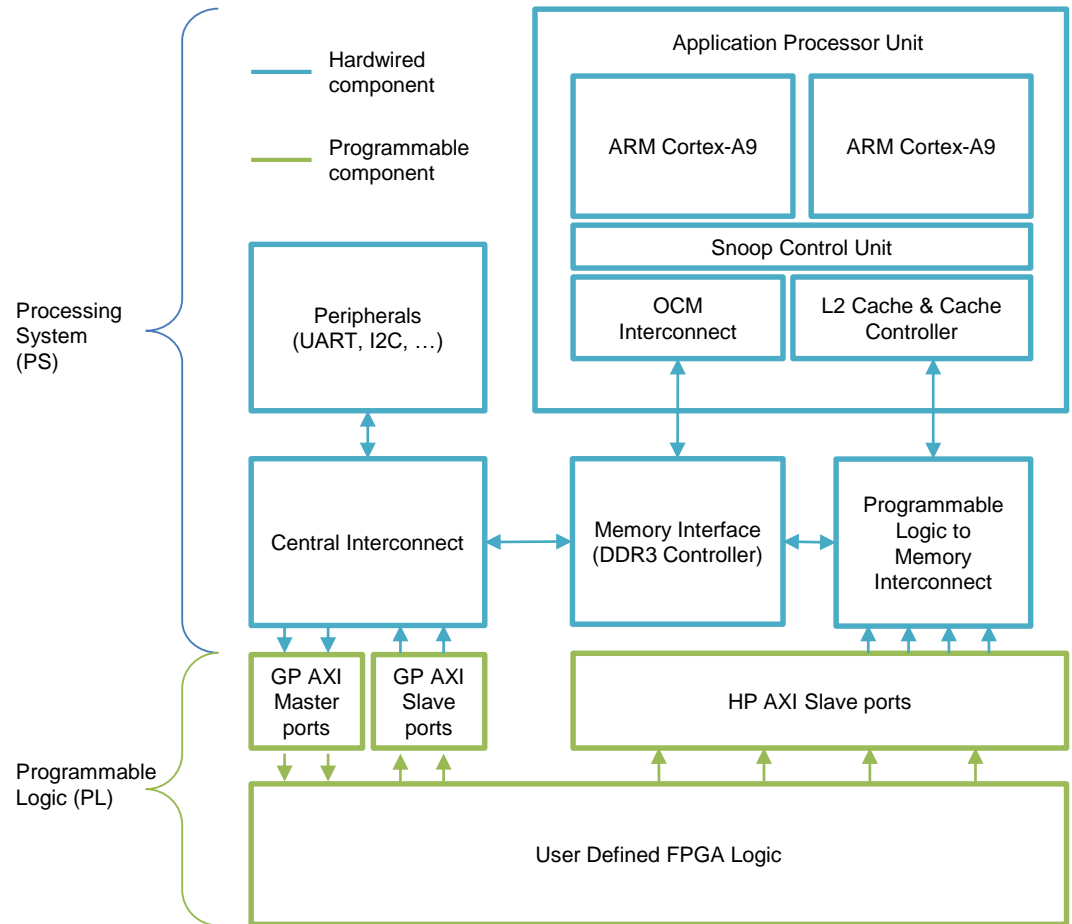
■ Xilinx's embedded SoC

■ 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

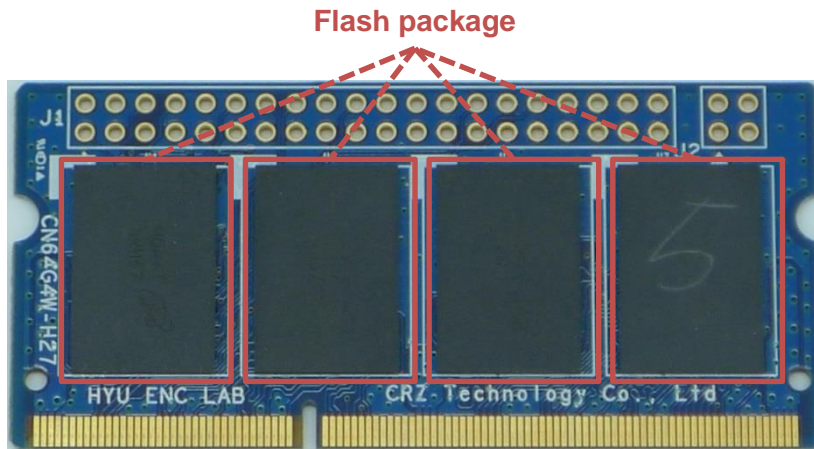
- CPU is more faster than soft core (such as MicroBlaze)
- No need to worry about organizing hardware memory controller, and some other peripherals (such as UART)
- Xilinx supports BSP (Board Support Package)



Zynq-7000 architecture overview

Cosmos OpenSSD NAND Module

- **Each module has 4 flash packages**
 - One flash package
 - Capacity: 32 GB
 - Page size: 8640 Bytes (spare area: 448 Bytes)
 - Synchronous NAND
- **Used with Tiger3 Controller**



Front side



Rear side

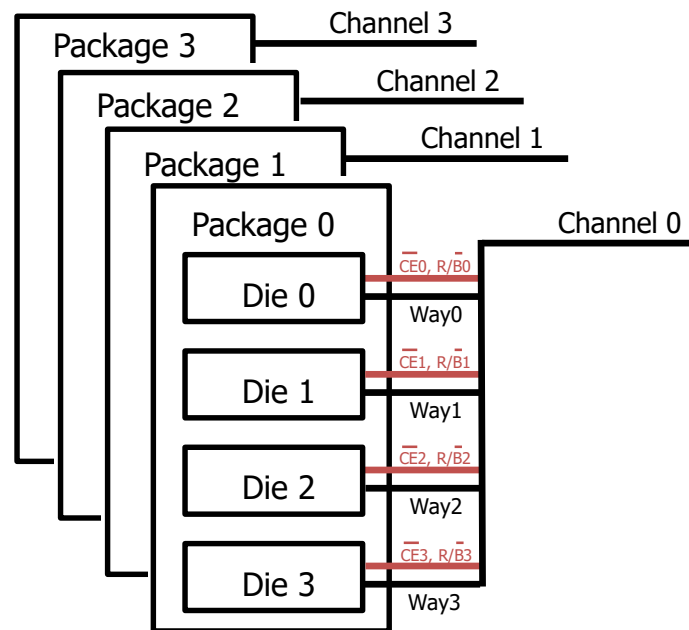
Logical Organization of Flash module

■ Module configuration

- 4 channels/module and 4 ways/channel

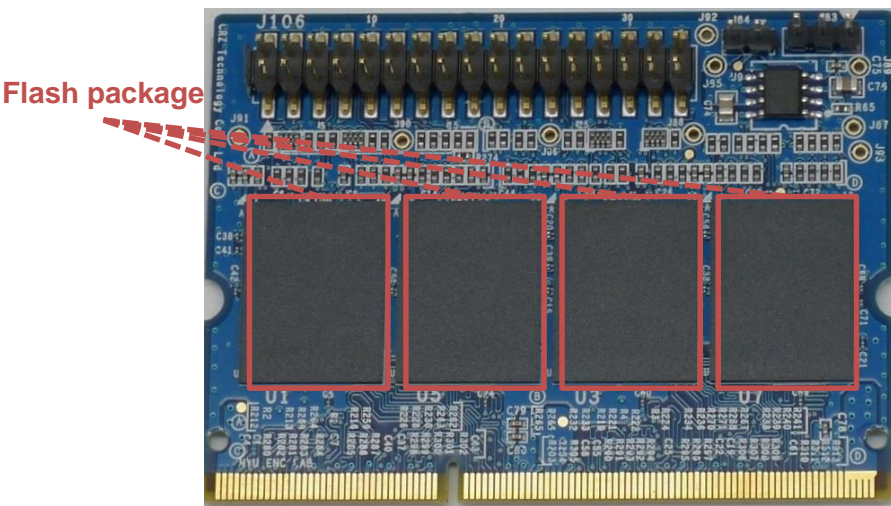
■ Shared signals within a channel (a package)

- 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

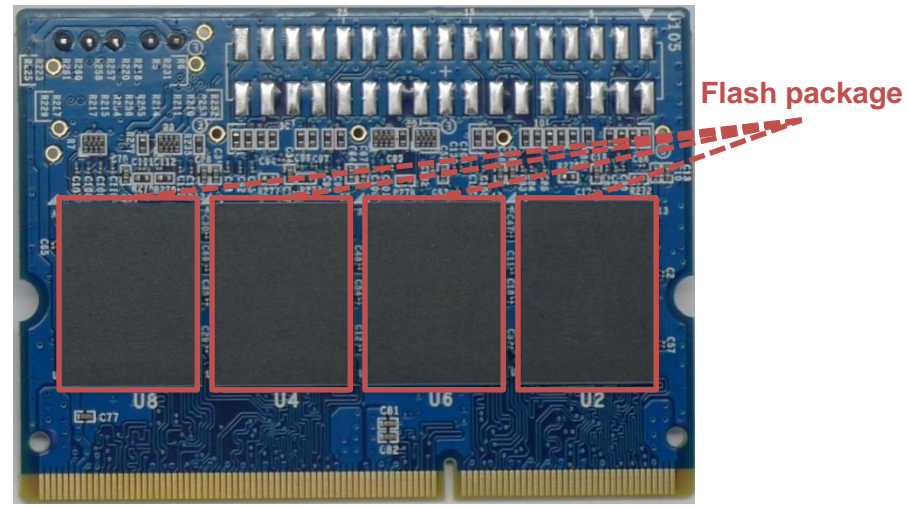


Cosmos+ OpenSSD NAND Module

- Each module has 8 flash packages
 - One flash package
 - Capacity: 128 GB
 - Page size: 18048 Bytes (spare area: 1664 Bytes)
 - Toggle NAND
- Used with Tiger4 Controller



Front side



Rear side

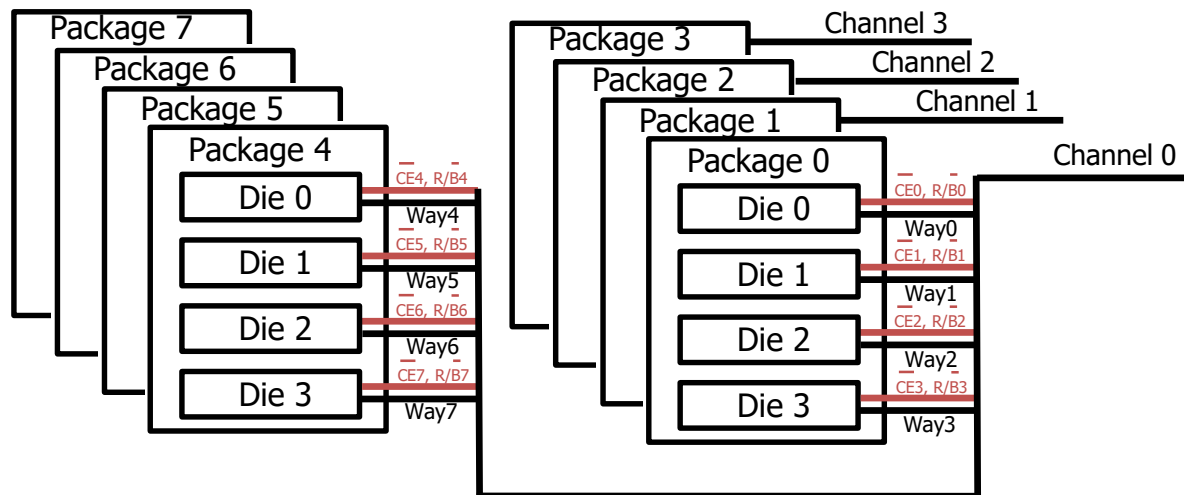
Logical Organization of NAND Flash Module

■ 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



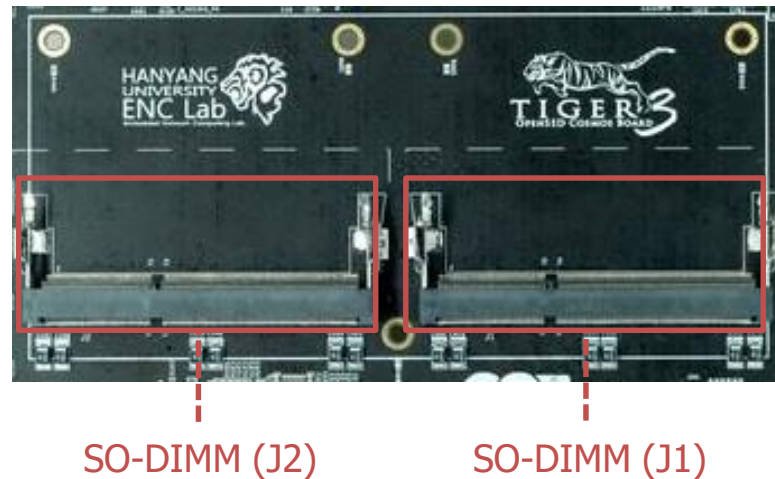
NAND Module Setup

■ Cosmos OpenSSD

- Supports only one flash module slot (J1)

■ Cosmos+ OpenSSD

- Supports both flash module slots (J1, J2)



■ Caution

- Cosmos/Cosmos+ OpenSSD flash module slots have custom pin maps
- You should not insert any SDRAM module into this slot

External PCIe

- **Expand PCIe Slot of host PC to connect external device**
- **Adapter card**
 - Installed on host PC
 - Provide a high-performance and low latency solution for expanding PCIe
- **External PCIe cable (8-lane)**
- **External PCIe connector (8-lane) on platform board**
 - 2.5 GT/s for a Gen1, 5.0 GT/s for a Gen2
 - Connected with high data rate serial transceiver in FPGA



External PCIe adapter



External PCIe cable

Connection with Development PC

■ JTAG cable

- Used for downloading hardware and software binary files
- Available cable types
 - USB type A to USB type micro B cable
 - Emulator, JTAG N pin cable (N: 7, 14, 20)

■ UART cable

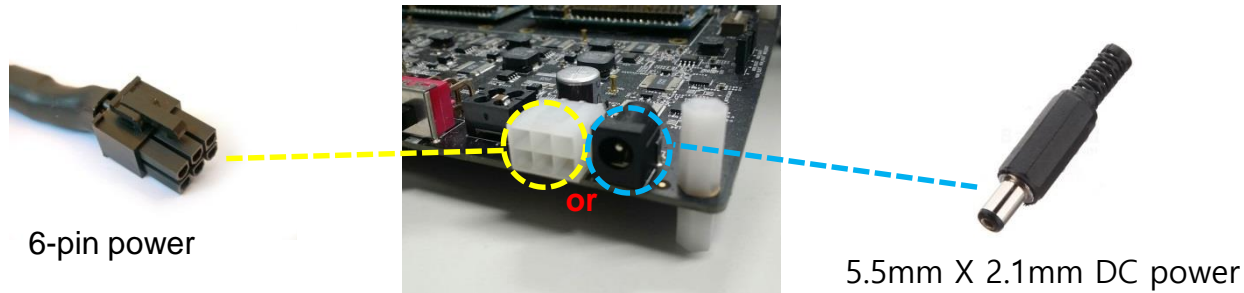
- Used for monitoring internal processes of Cosmos+ OpenSSD
- USB type A to USB type A cable



Power Connection

■ Single-source of power to the platform board

- 6-pin power connector (J181) or 5.5mm X 2.1mm DC power plug (J182)



■ 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 | 12V | NC | NC | GND | 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 (J181)

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 FPGA

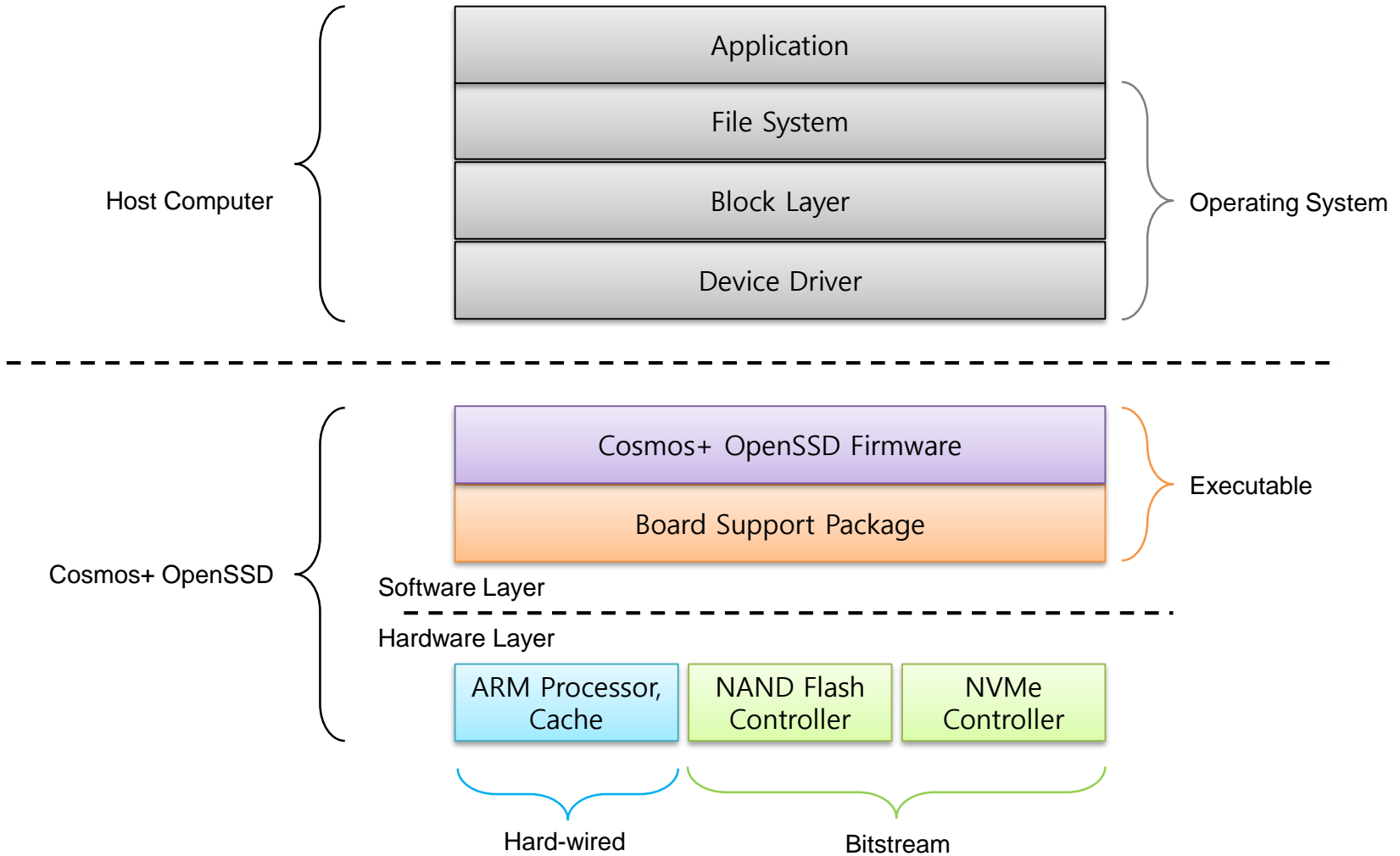
■ FPGA bitstream

- Used to configure the programmable logic side of Zynq FPGA

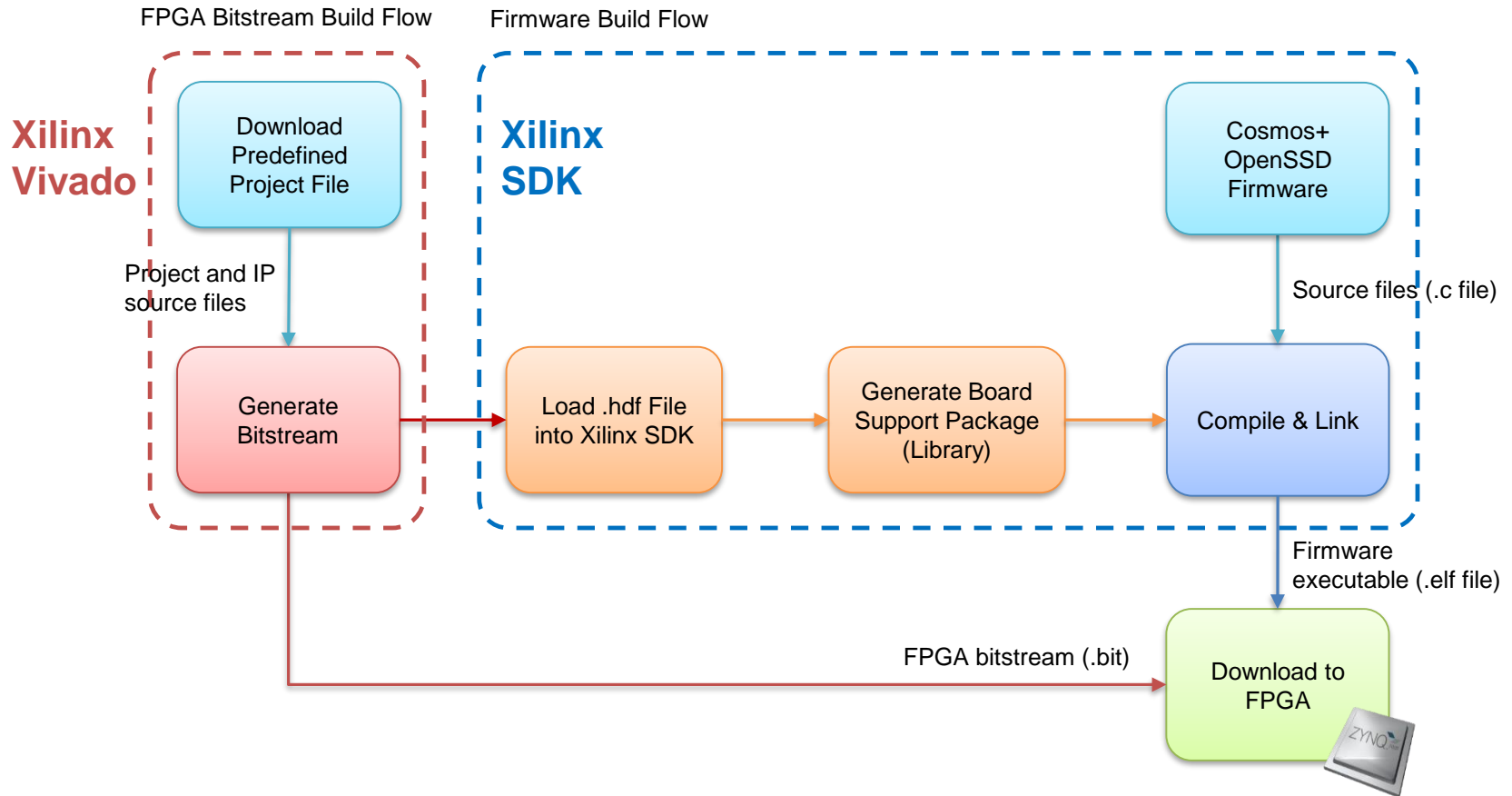
■ Firmware

- Manages the NAND flash array
- Handles NVMe commands

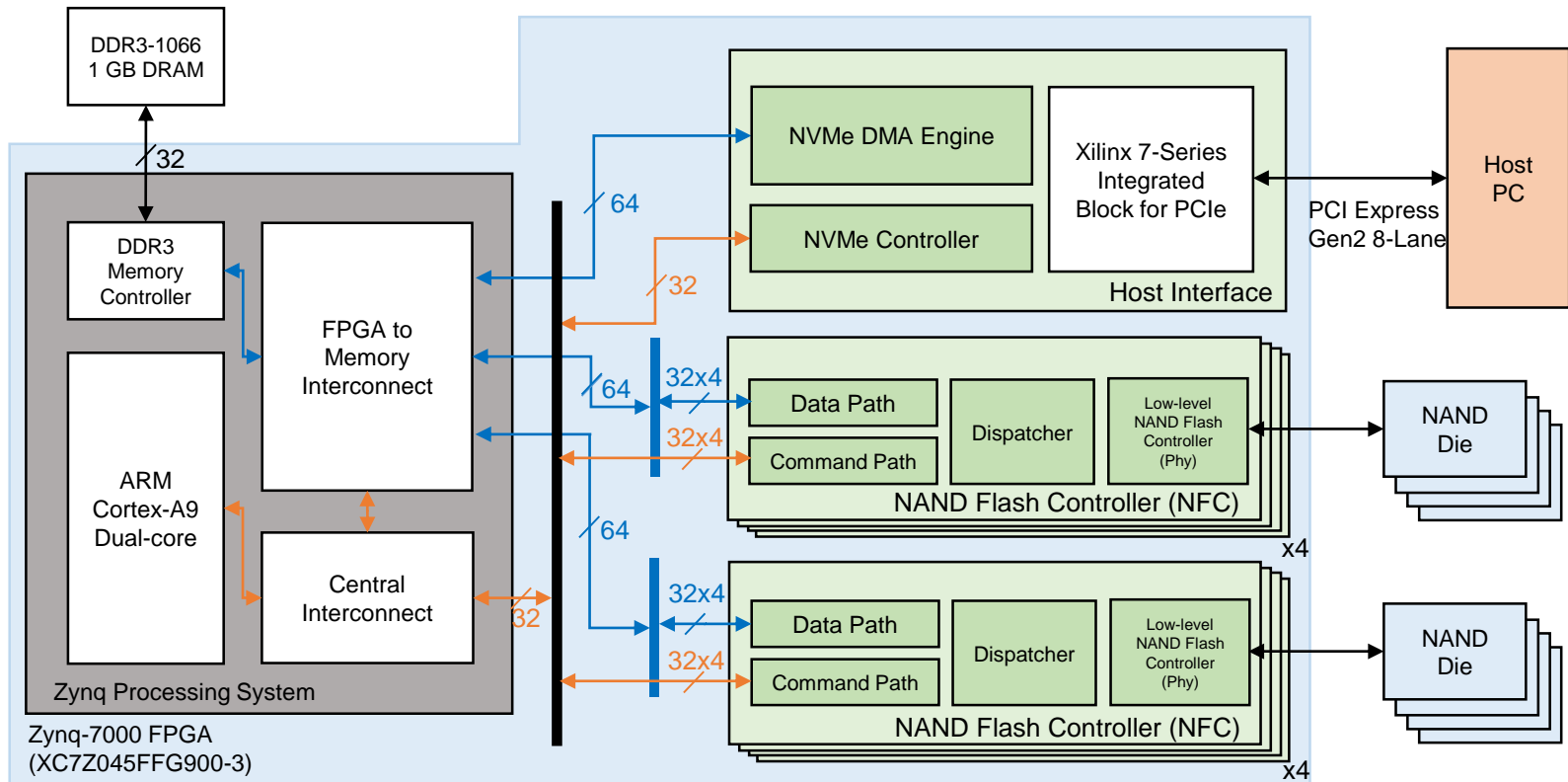
Host System Software Architecture



Software Porting Flow



Cosmos+ OpenSSD Internal System Overview



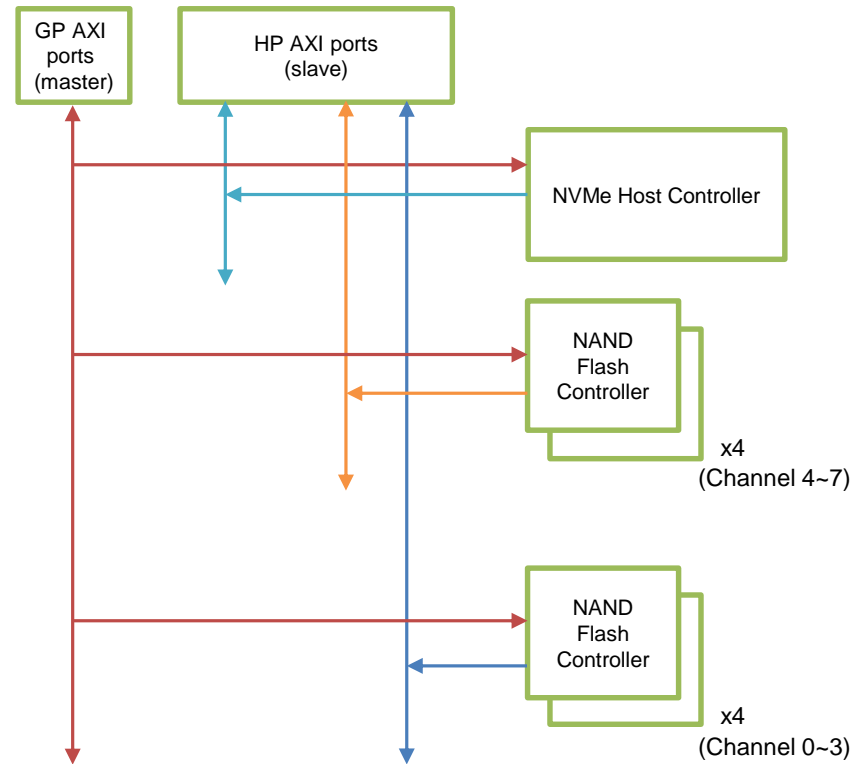
System Bus Structure

■ General Purpose (GP) AXI4 Lite bus

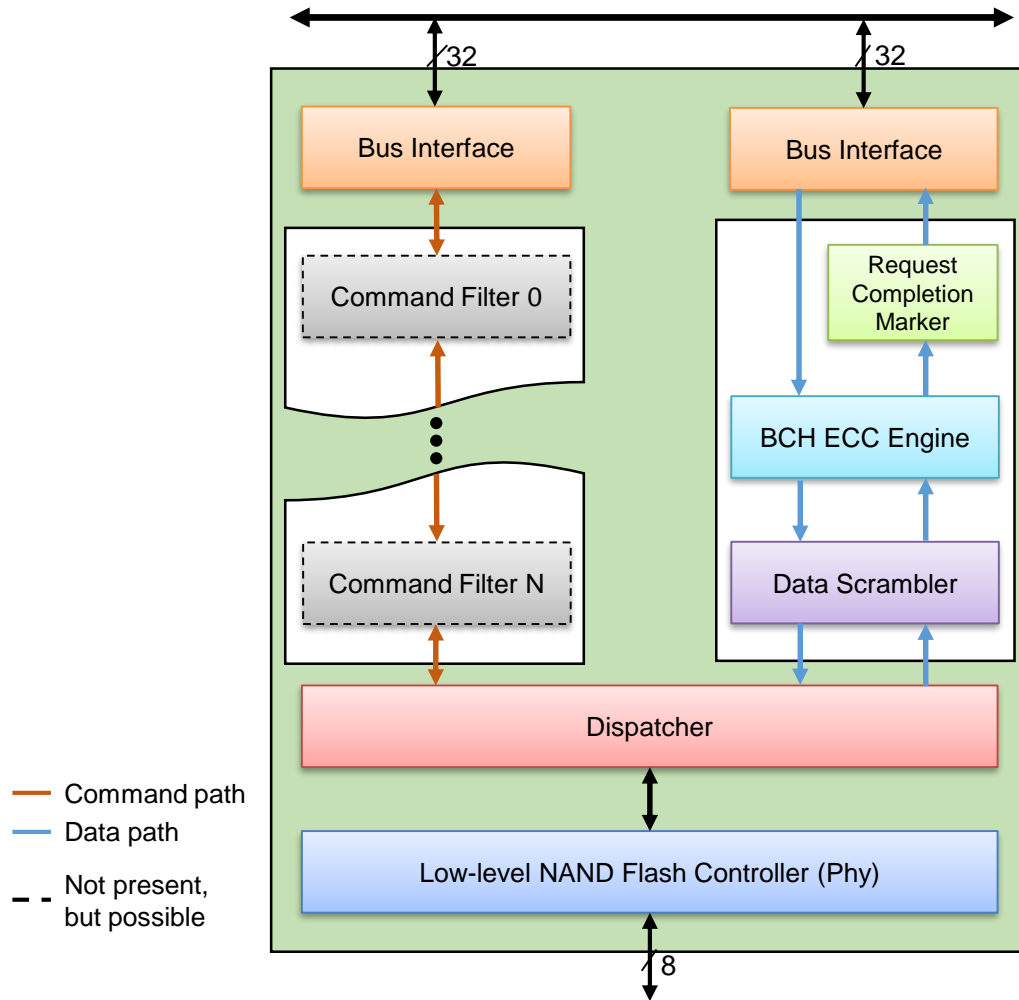
- 32bits interface
- Used for control
- Operates @ 100MHz

■ High Performance (HP) AXI4 bus

- 64bits interface
- Used for Direct Memory Access (DMA)
- Operates @ 250 MHz

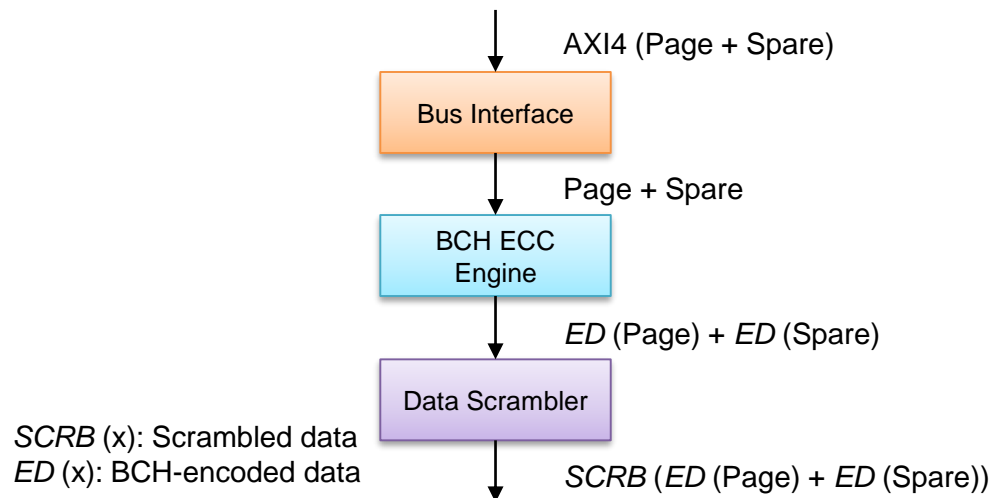


NAND Flash Controller Overview



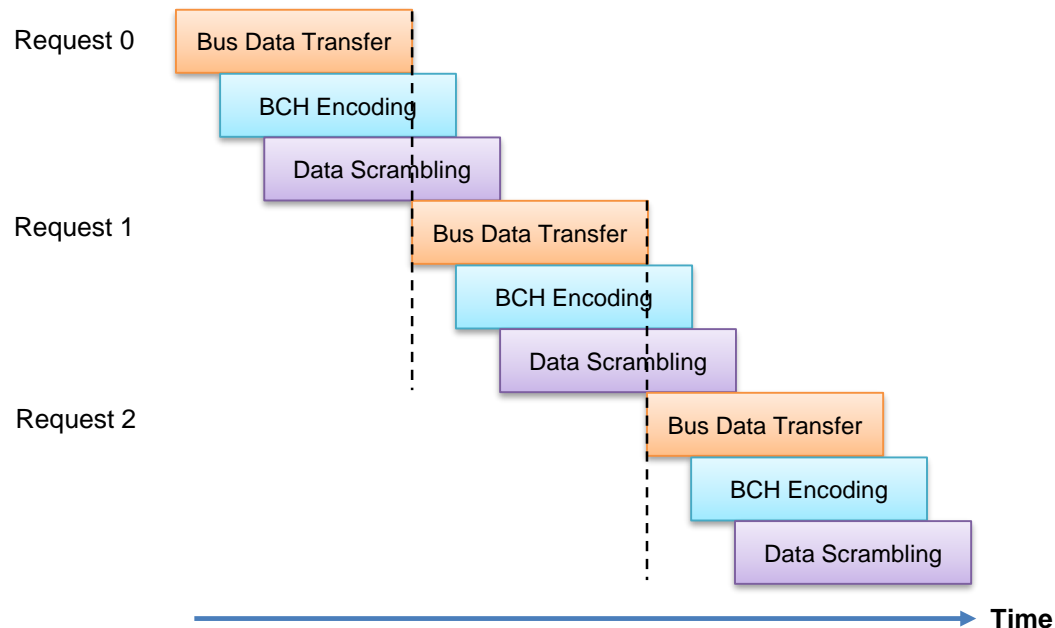
Layered and Modular NFC Design

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



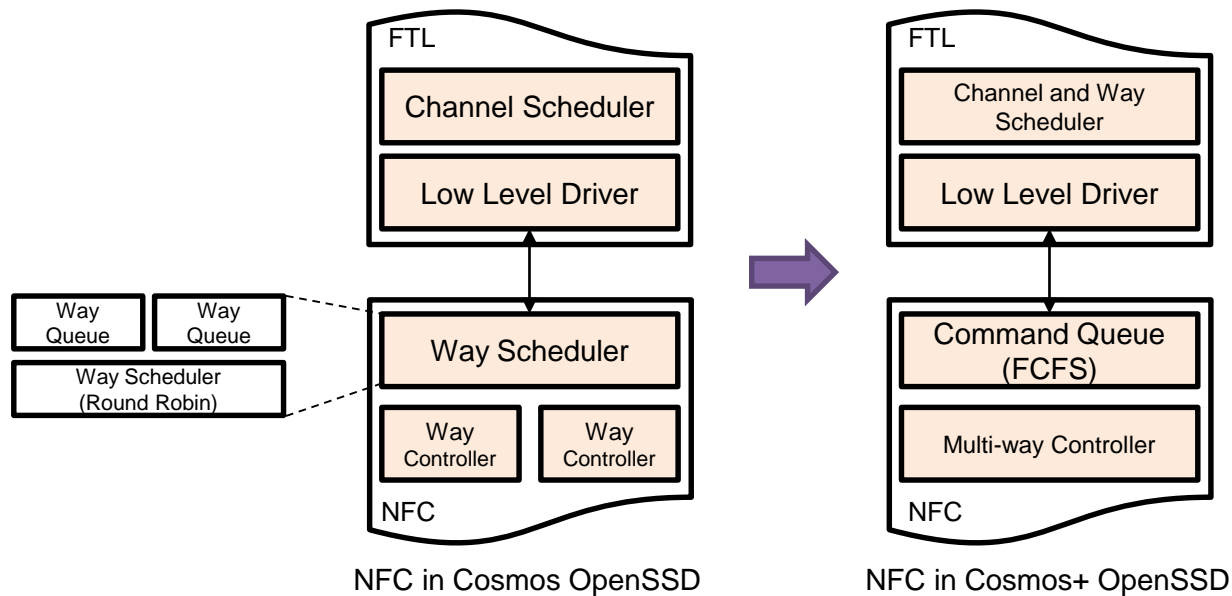
Pipelined NFC Operation

- 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



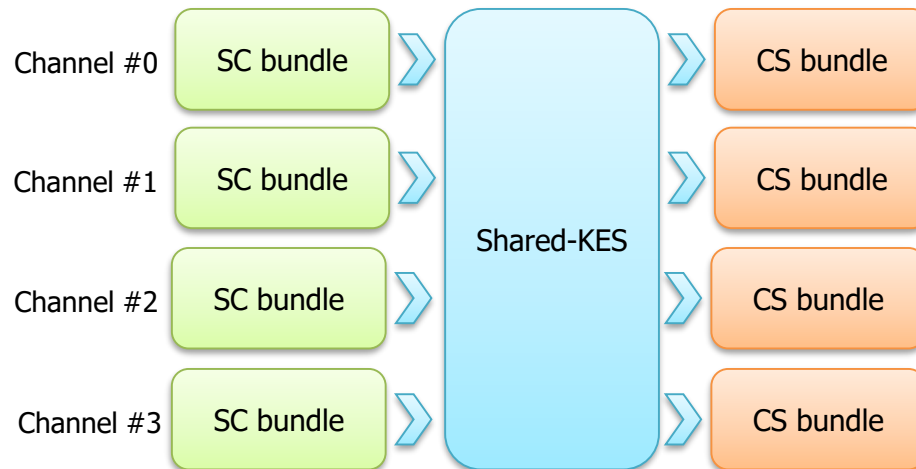
Software-controlled NAND Flash Scheduler

- Hardware-level way scheduler of NFC in Cosmos OpenSSD is removed
- FTL is now responsible for channel and way scheduling
- This enables more flexible scheduling policy



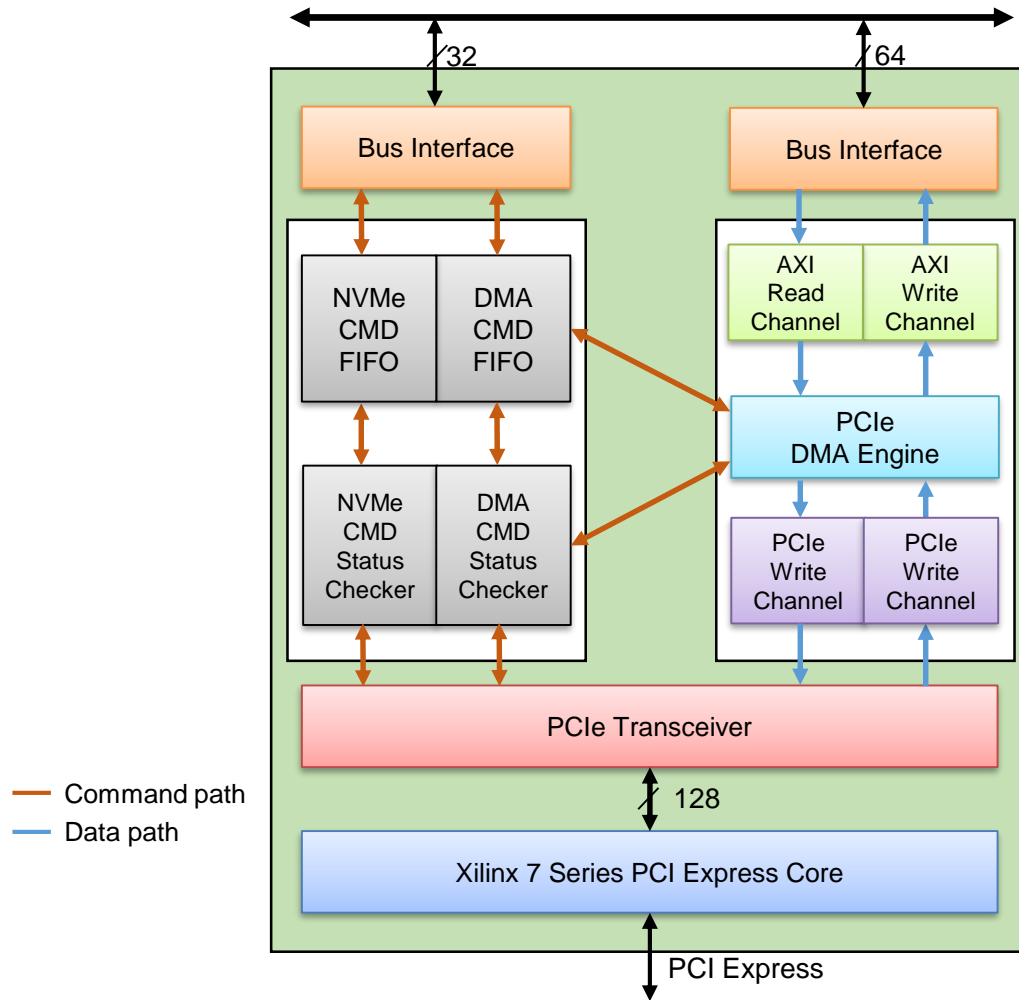
BCH ECC Engine with Shared-KES

- Key equation solver (KES) used more ($\geq 50\%$) 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



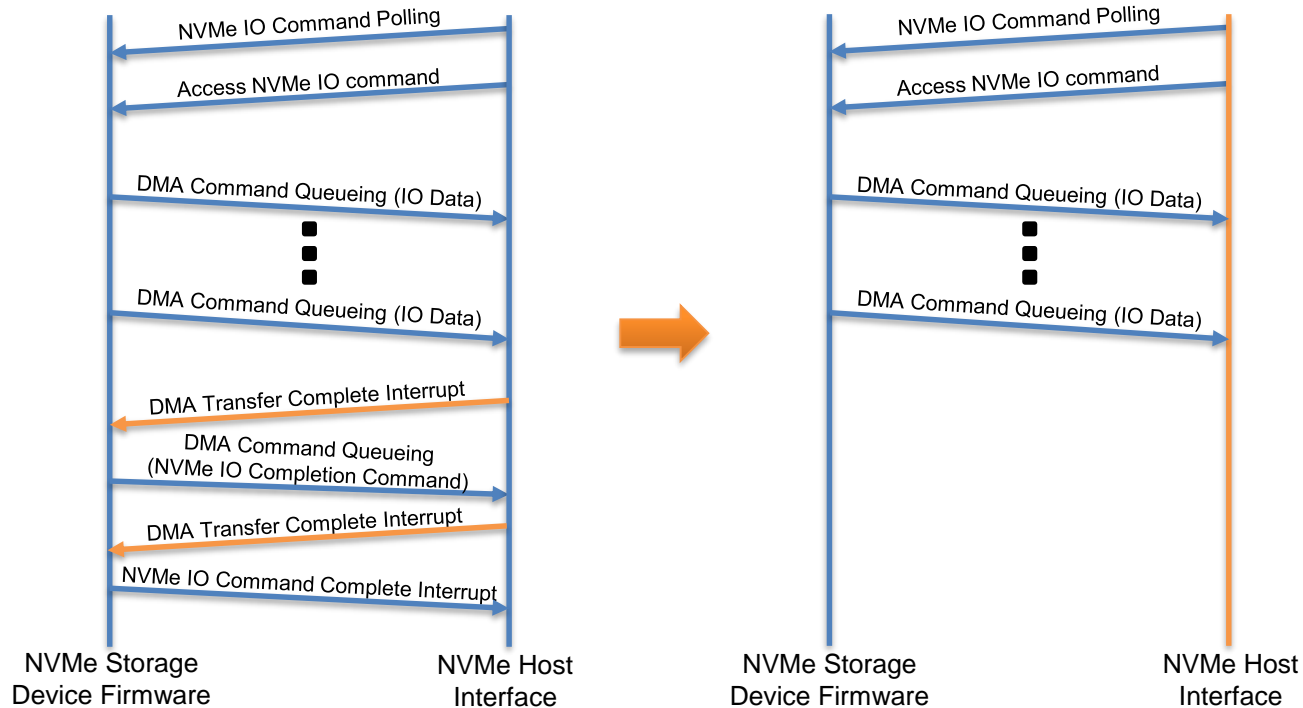
* (256 B x 2) bundle, (13 x 2) bit error correction, (8 bit x 2) parallel level

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 \geq 3.3
- 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

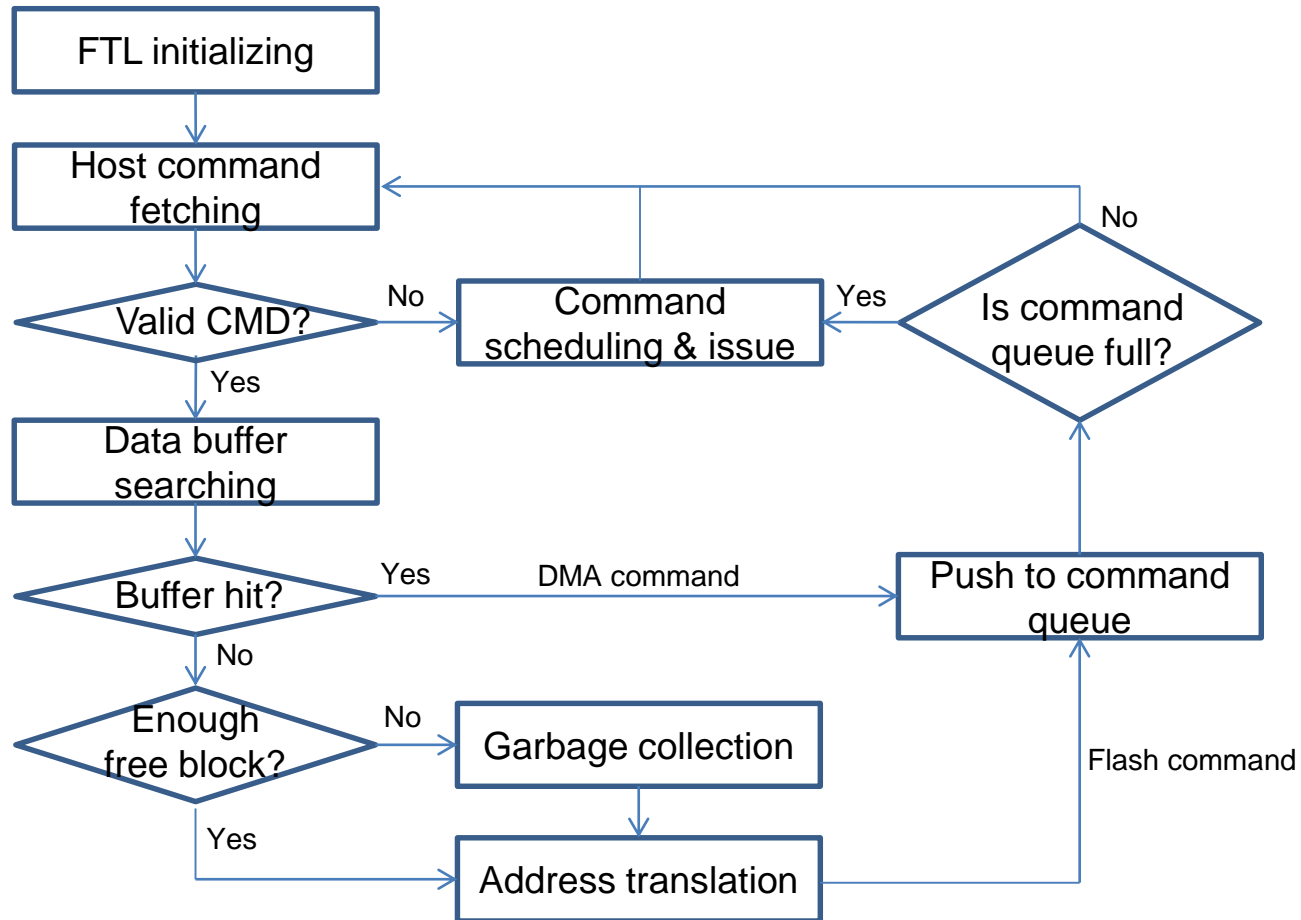
Command Set

- 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

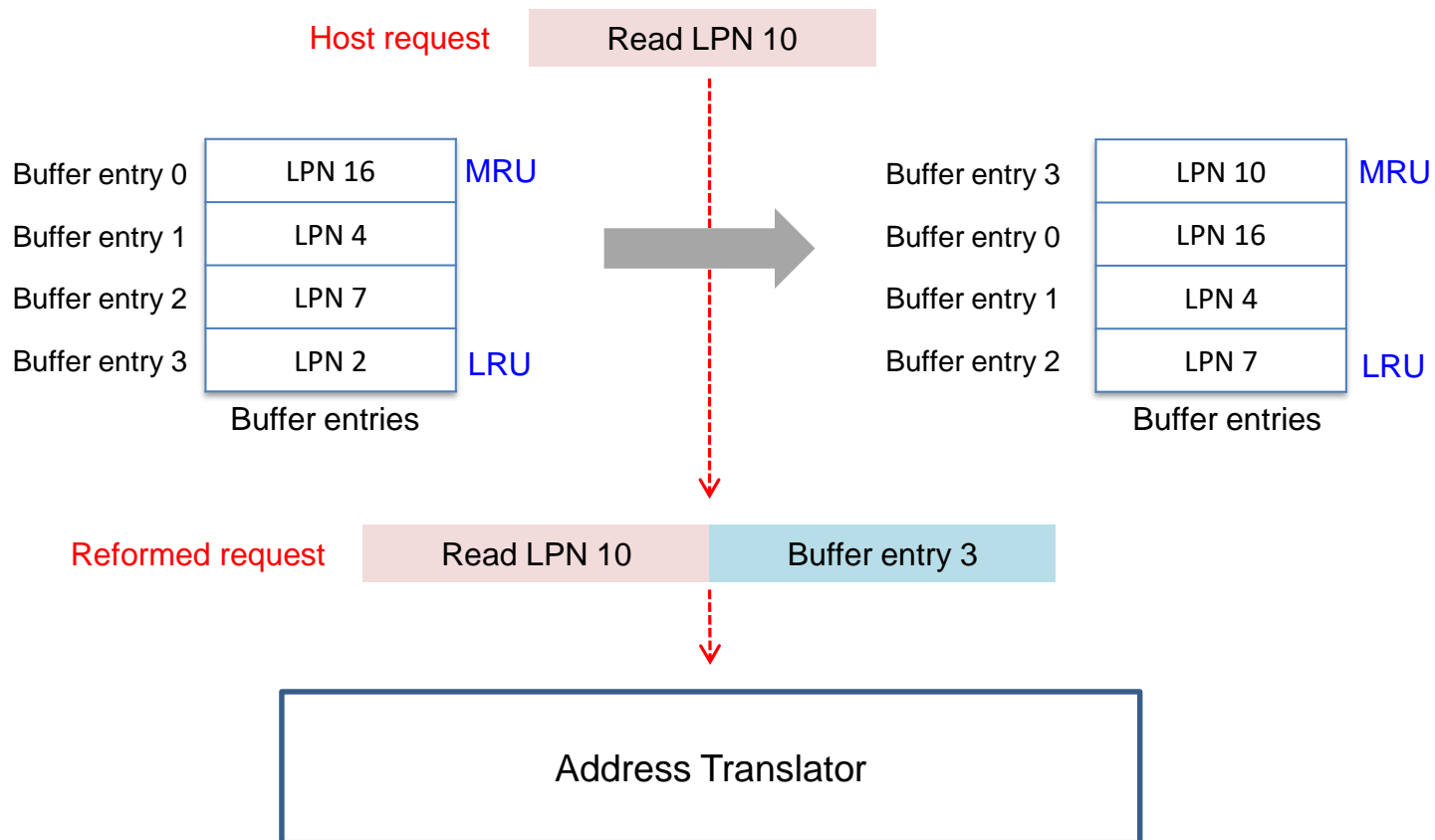
Firmware Overall Sequence



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

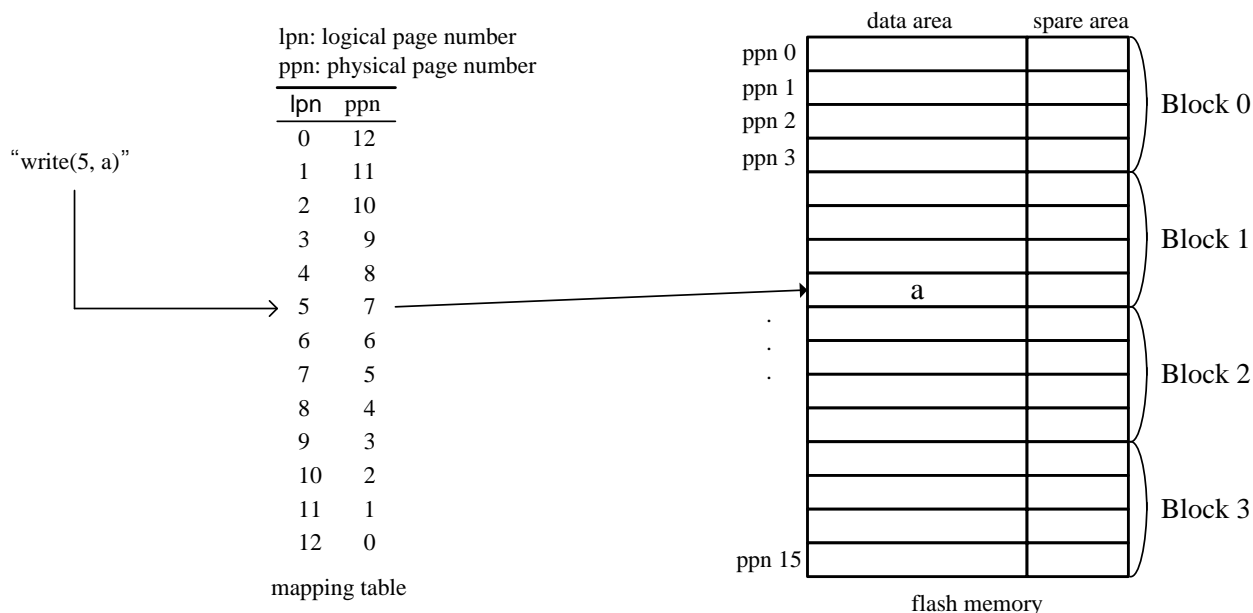
- Every logical page is mapped to a corresponding physical page

■ Advantage

- Better performance over random write than block-level mapping

■ Disadvantage

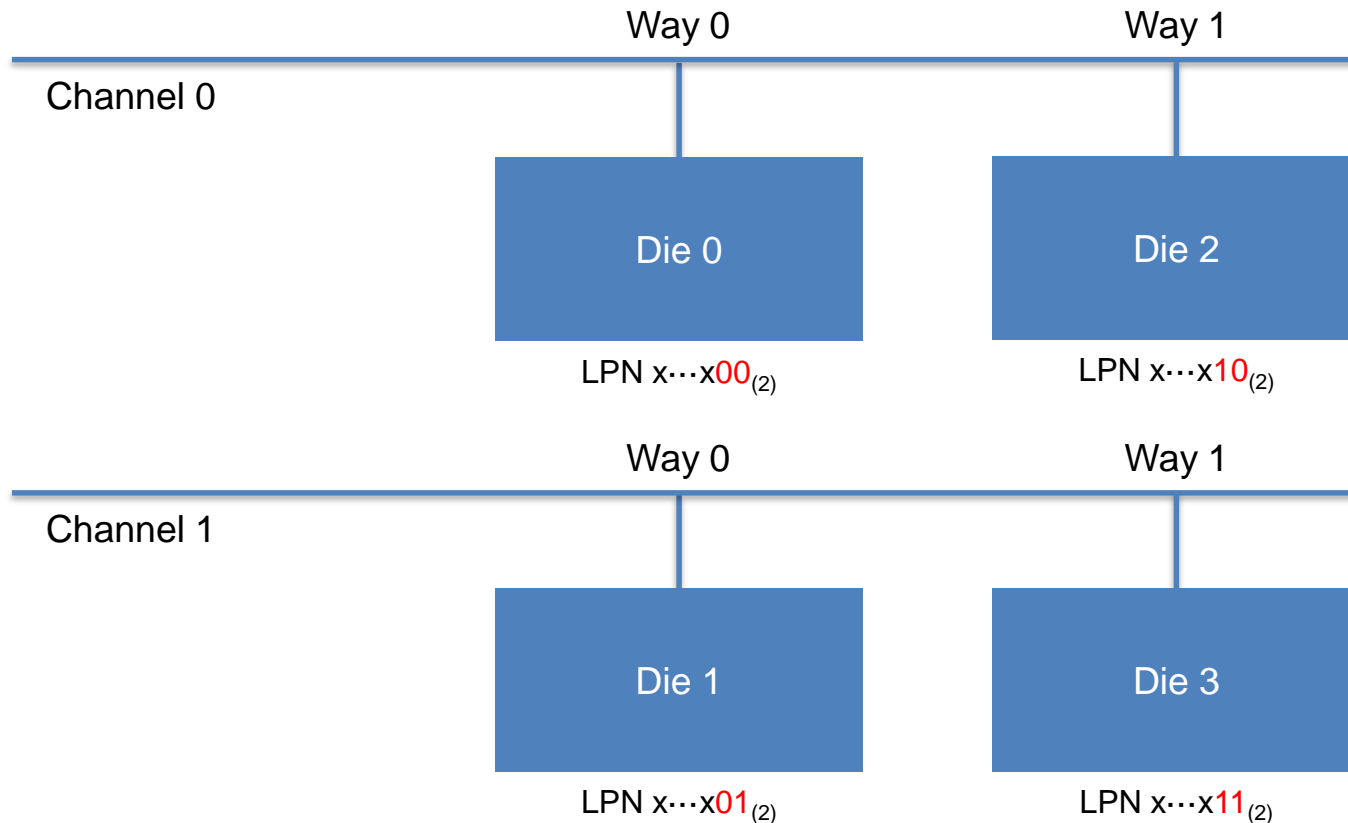
- Huge amount of memory space requirement for the mapping table



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)

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

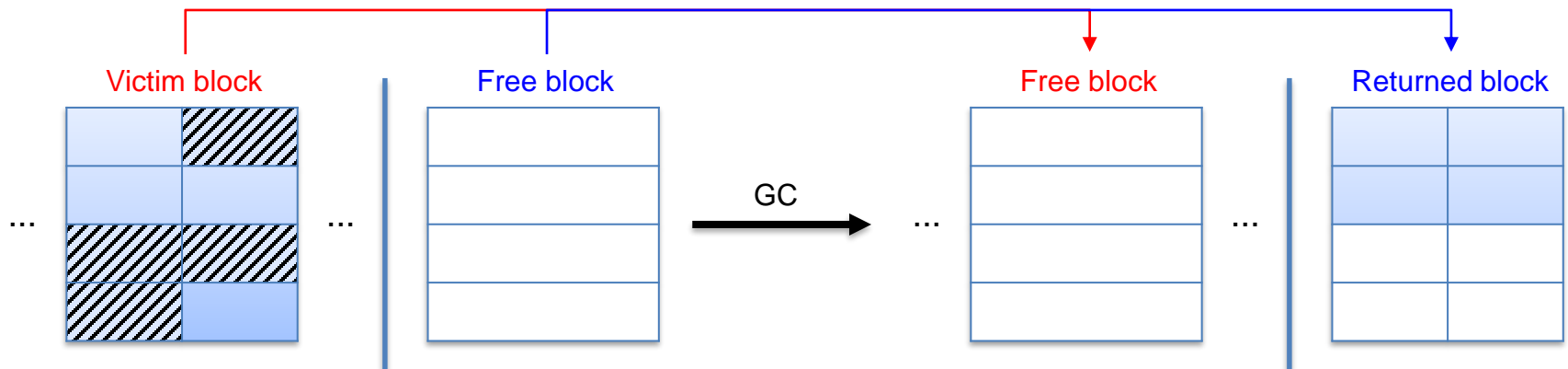
On-demand GC

■ 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

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

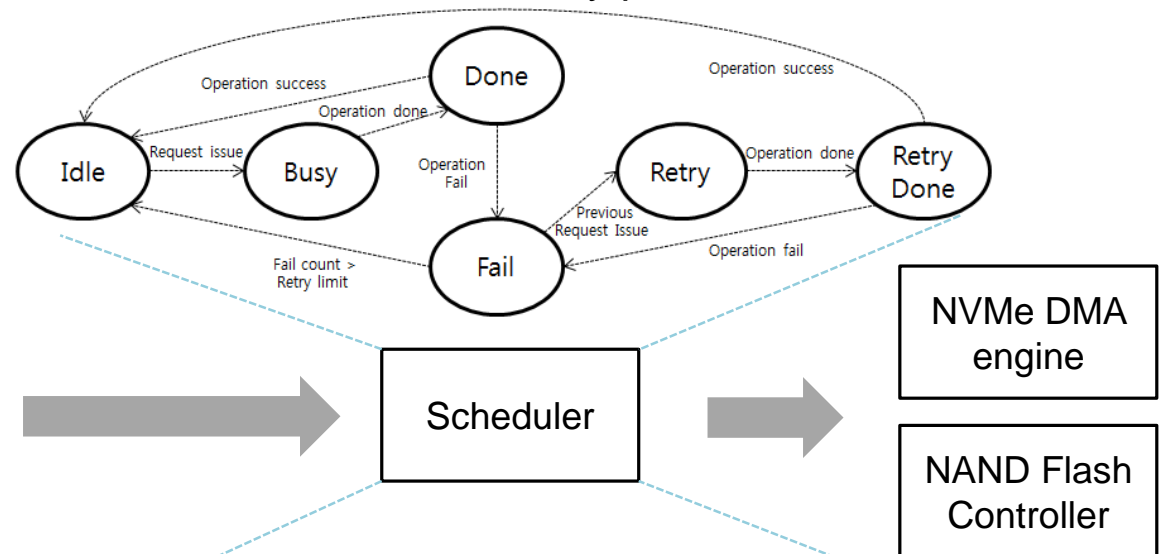
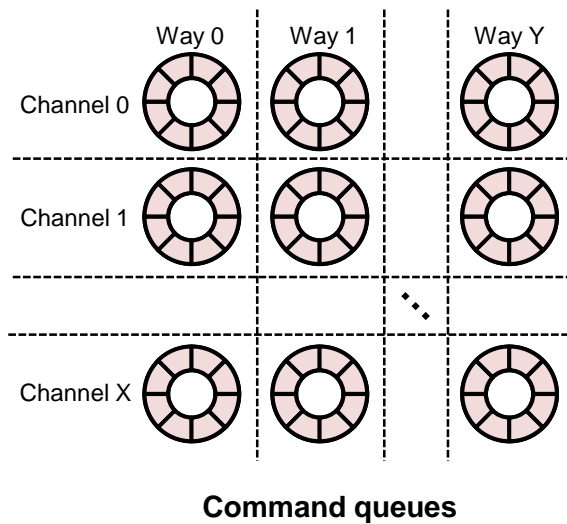
V2FCommand_StatusCheck

- ▶ 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 | 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 8-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**

SLC Mode of Firmware

- Currently, MLC to SLC mode transition command of NAND flash is not supported
- Accessing only LSB pages achieves similar characteristics to real SLC NAND flash

| Paired page address | |
|---------------------|-----------|
| LSB pages | MSB pages |
| 00h | 02h |
| 01h | 04h |
| | ⋮ |
| FDh | FFh |

Known Restrictions (2/3)

■ PCIe-NVMe

● Supports

- Up to PCIe Gen2.0 x8 lanes
- Mandatory NVMe commands
- PRP data transfer mechanism and out-of-order data transfer in PRP list
- 1 namespace (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 8
 - Maximum bandwidth of NAND flash bus 200 MT
- Not supports
 - Additional advanced commands are not supported (e.g. multi-plane operation)



Get Started with Cosmos+ OpenSSD

Overall Steps

■ Preparing development environment

- Host computer
- Platform board
- Development tools

■ Building materials

- FPGA bitstream
- Firmware

■ Operating Cosmos+ 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 Cosmos+ PCIe DMA engine |

Tested Host PC Operating System

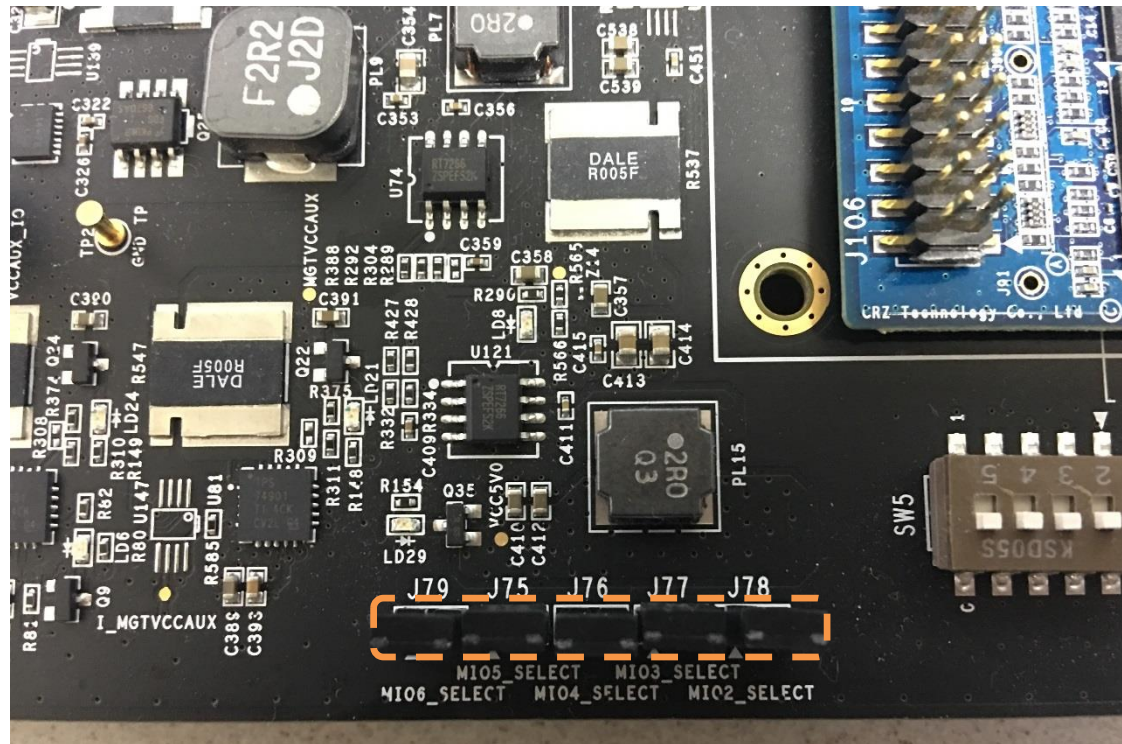
| OS | x86/x64 | Result | Comment |
|---------------------------|---------|-------------|--|
| Windows 7 | x64 | Working | with OFA driver |
| Windows 8.1 | x64 | Working | |
| Windows 10 | x64 | Not working | 4-byte addressing problems in C osmos+ PCIe DMA engine |
| Ubuntu 14.04 LTS or above | x64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

Preparing the Platform Board

- **Check jumper pins of the platform board**
- **Insert NAND flash module(s)**
- **Connect the external PCIe cable**
- **Connect the USB cable for jtag**
- **Connect the USB cable for UART**
- **Connect the power cable**

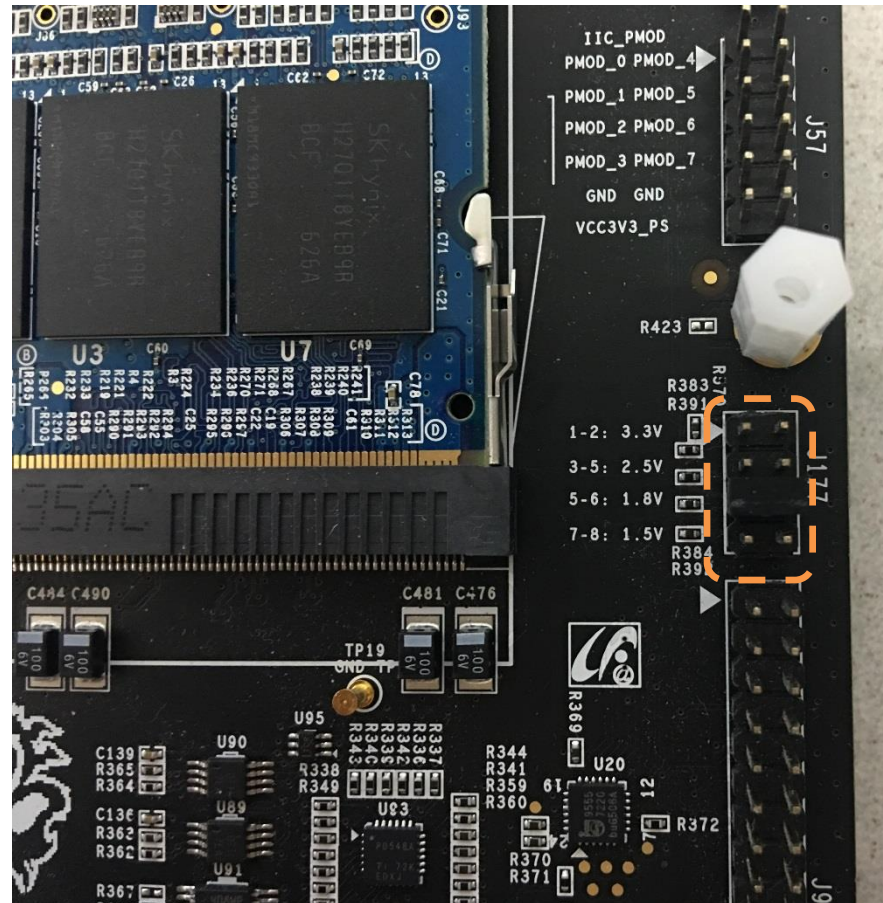
Check Platform Board Jumper Pins (1 / 5)

- Make sure that jumper pins on board are set as default below



Check Platform Board Jumper Pins (2 / 5)

- Make sure that jumper pins on board are set as default below

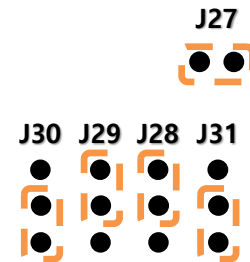
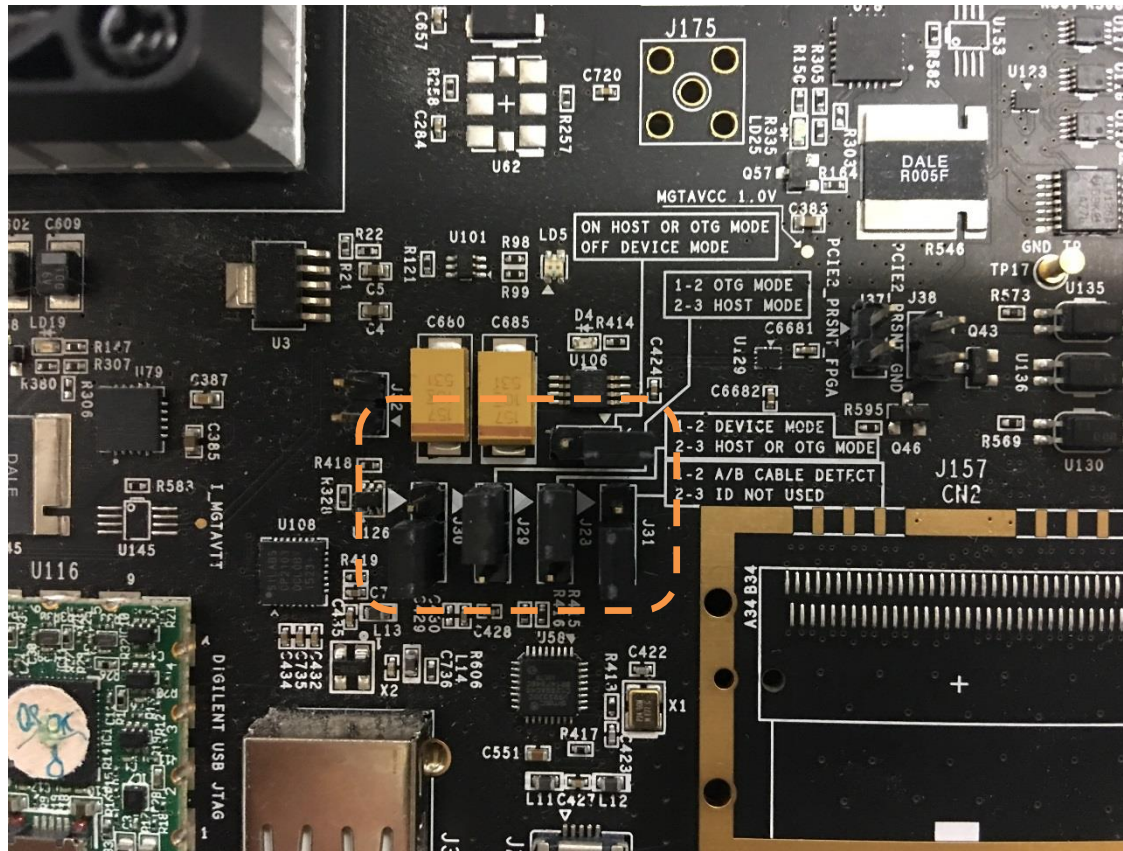


J177



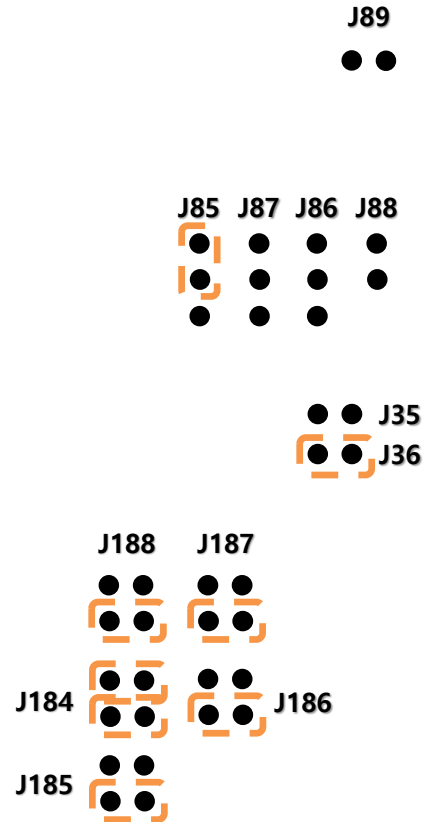
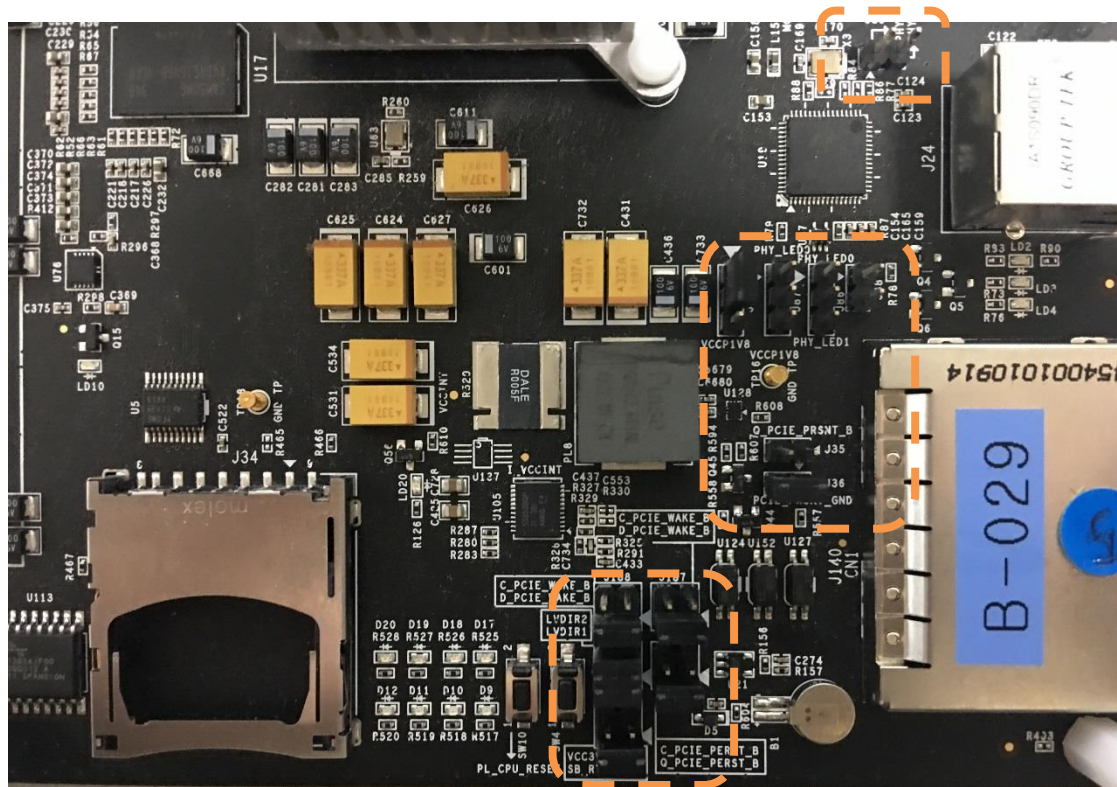
Check Platform Board Jumper Pins (3 / 5)

- Make sure that jumper pins on board are set as default below



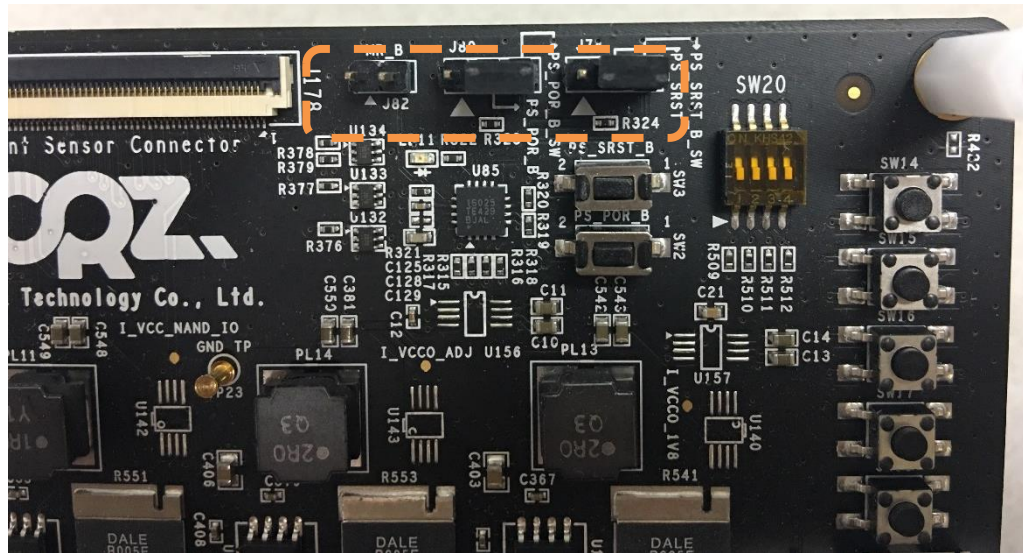
Check Platform Board Jumper Pins (4 / 5)

- Make sure that jumper pins on board are set as default below



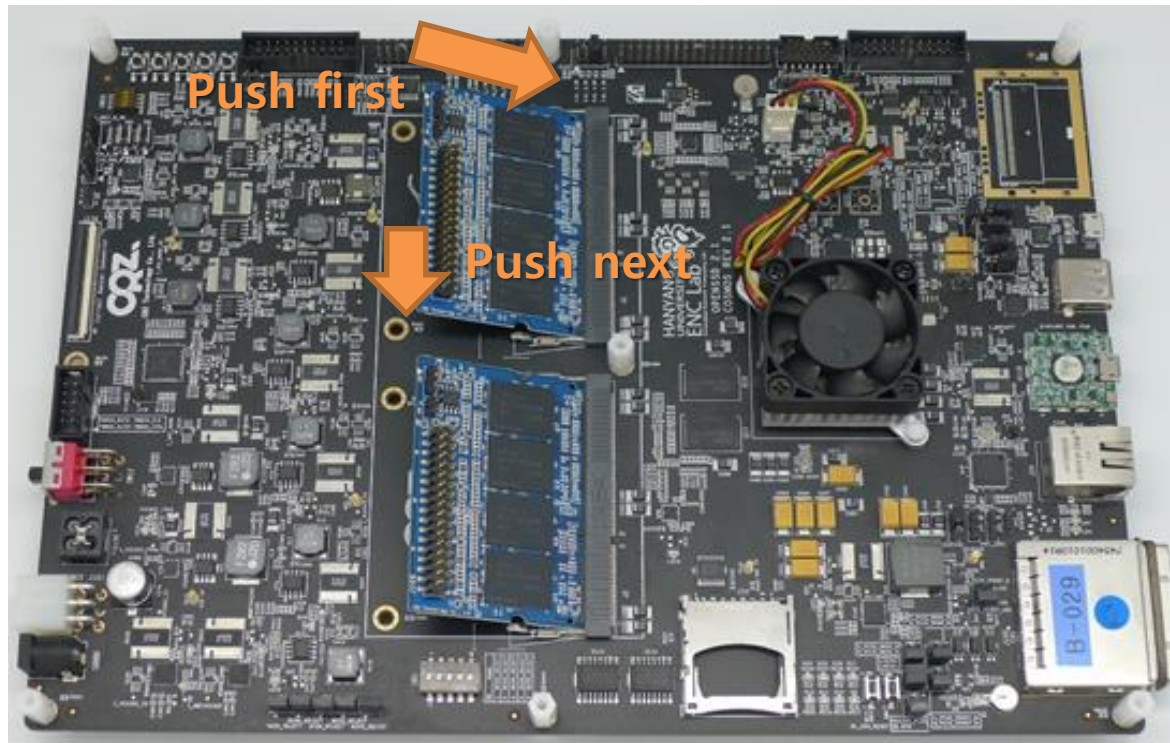
Check Platform Board Jumper Pins (5 / 5)

- Make sure that jumper pins on board are set as default below



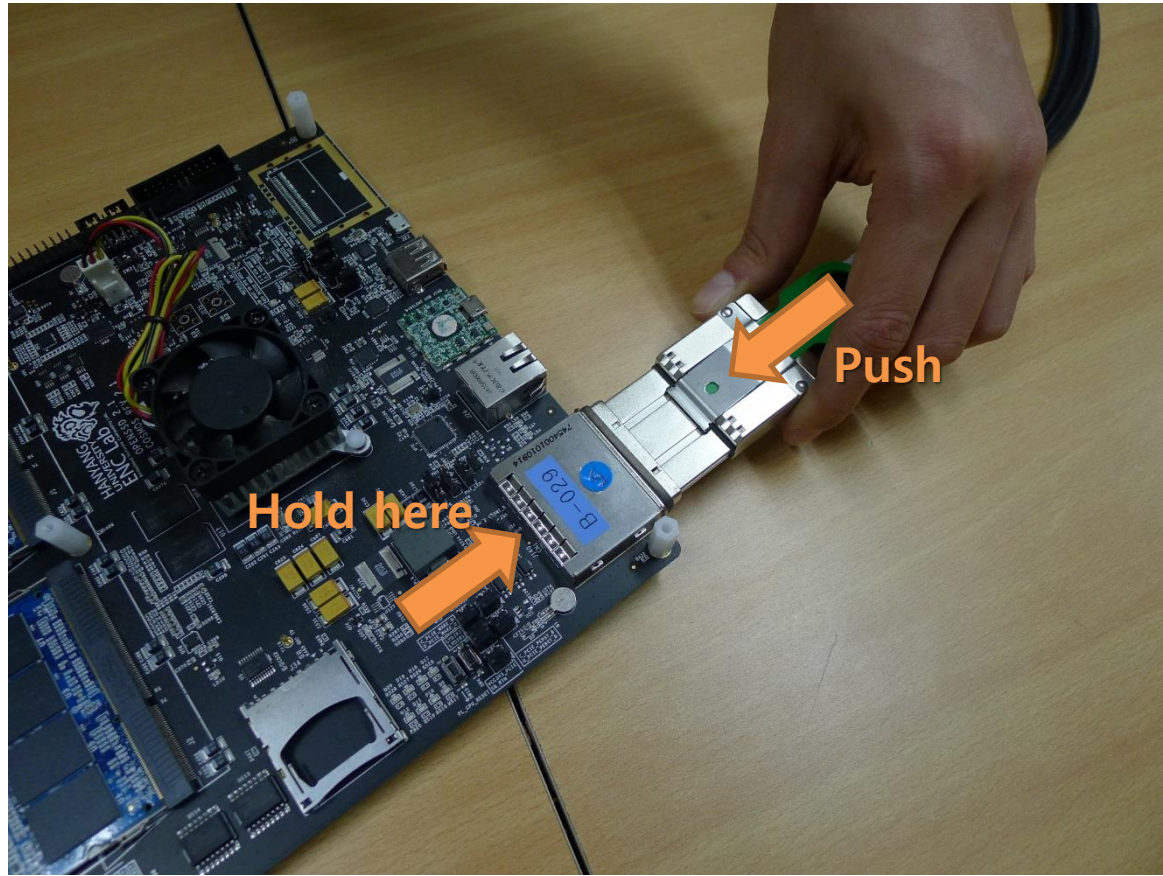
Insert NAND Flash Modules

- **A single NAND flash module can support up to 4-channel configuration**
 - For prebuild 3.0.0, two NAND flash modules are required
 - For predefined project 1.0.0, one NAND flash module is required



Connect External PCIe Cable (1 / 2)

- Hold external PCIe connector and push the cable in it



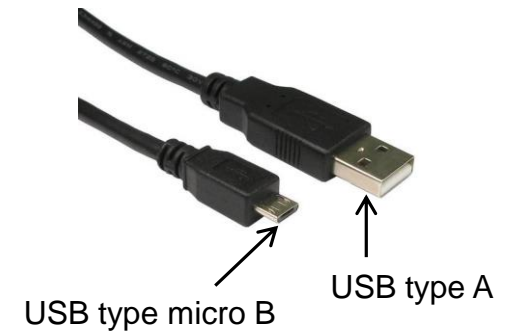
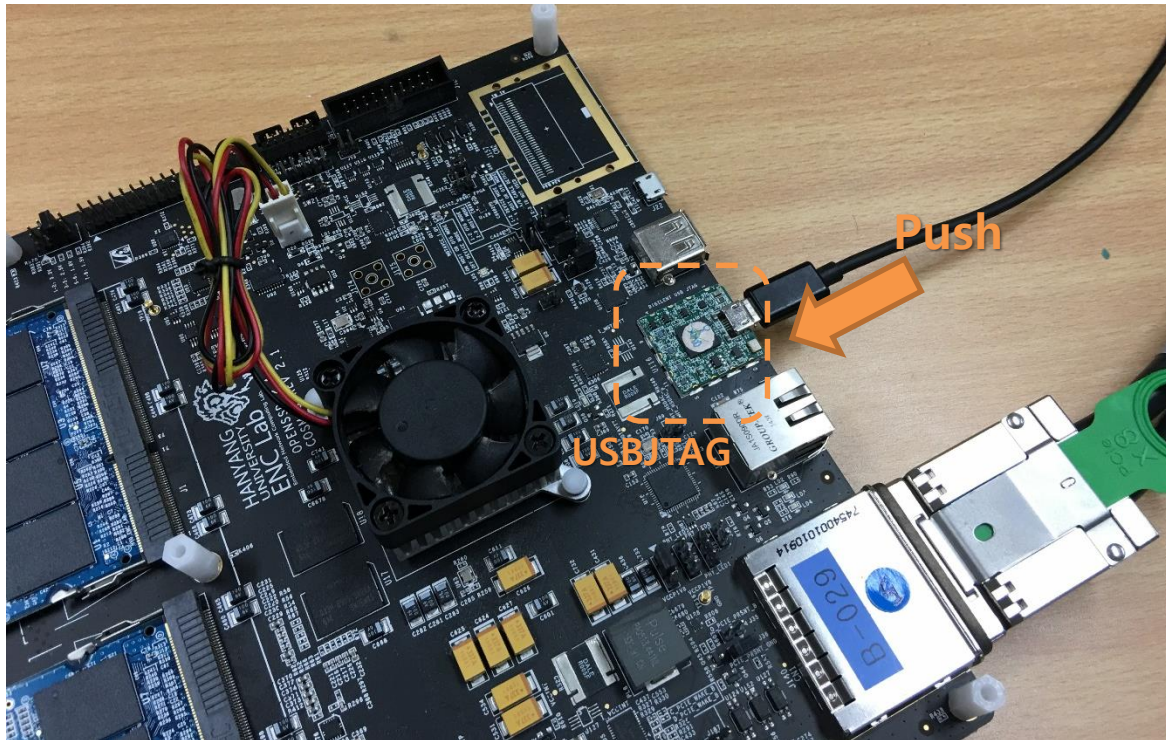
Connect External PCIe Cable (2 / 2)

- Make sure that the cable is fixed tightly



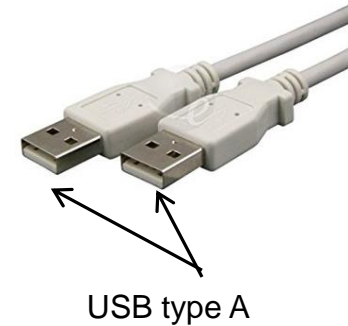
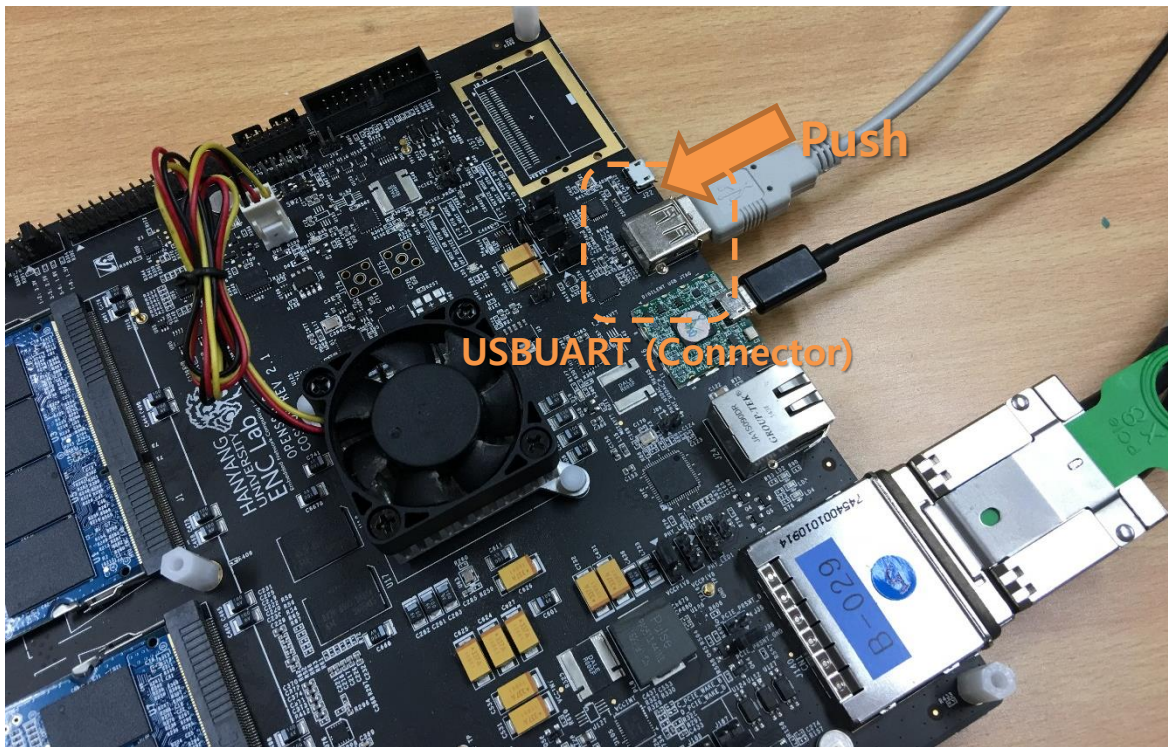
Connect USBTAG Cable

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



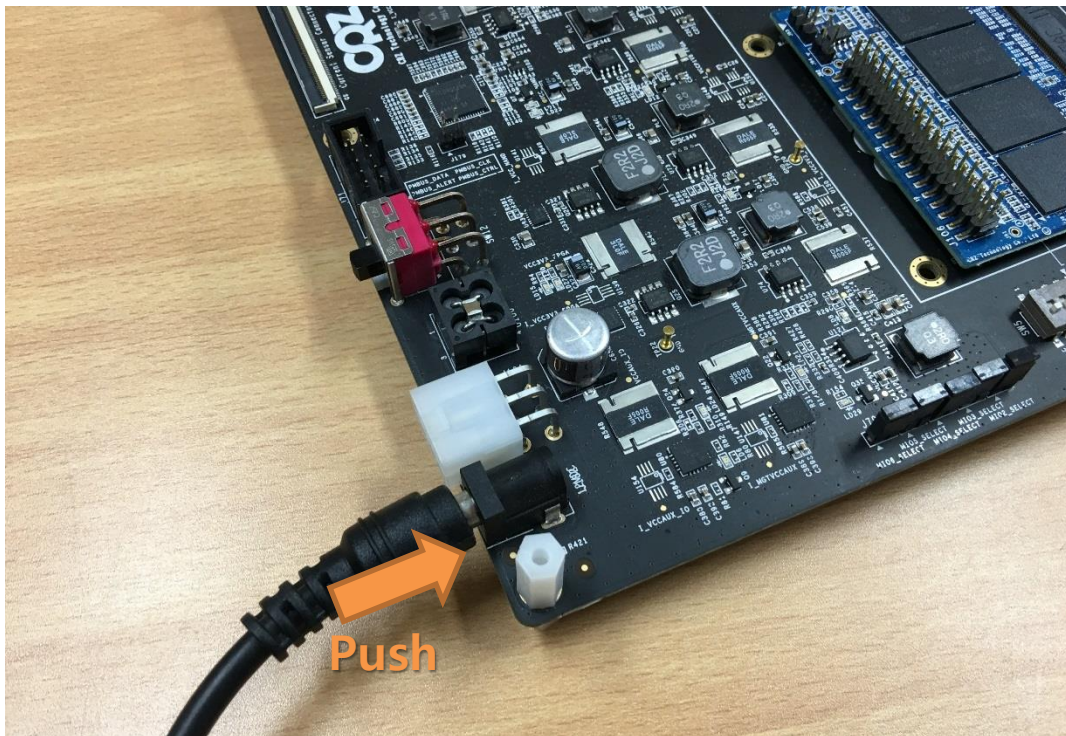
Connect USBUART Serial Communication Cable

- USBUART requires a USB type A (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

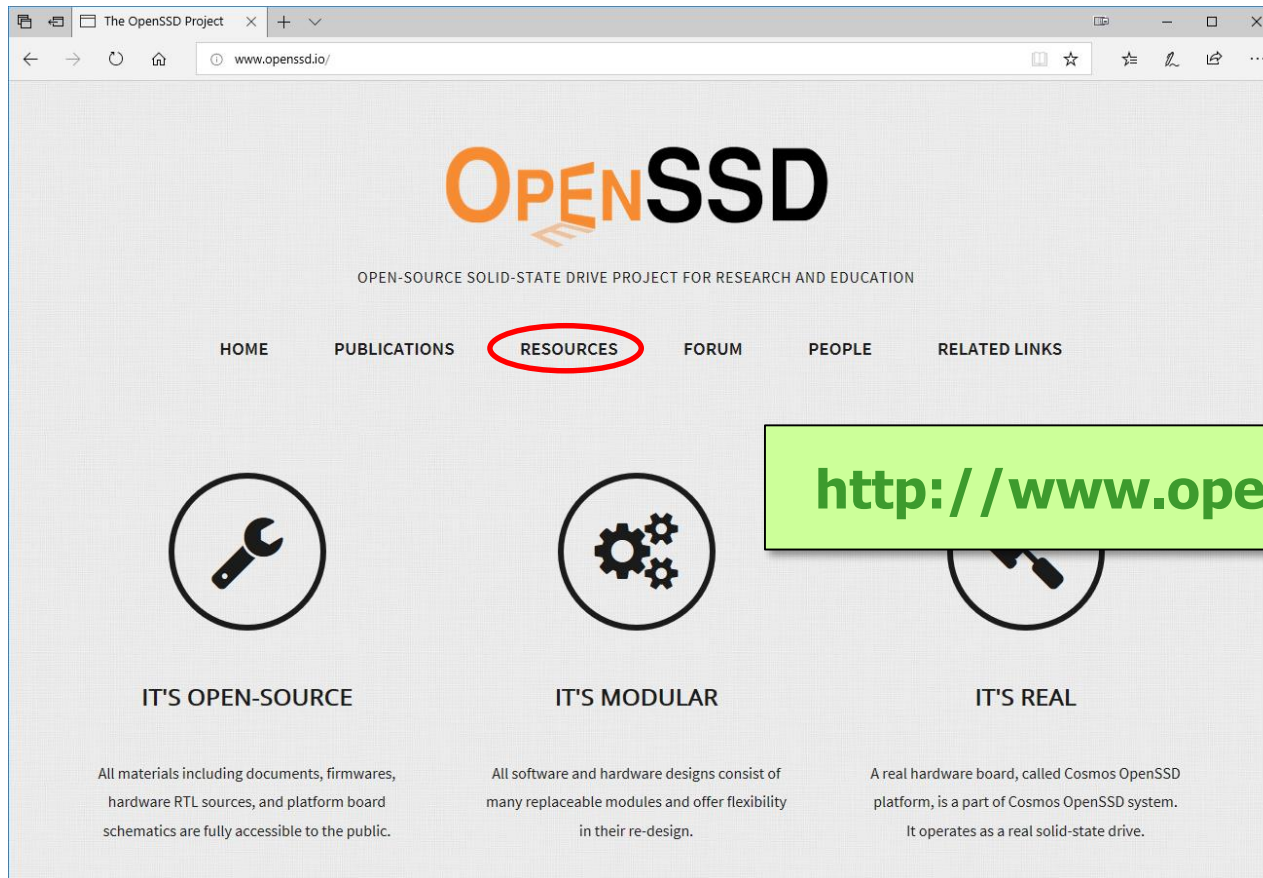
- Prebuilt FPGA bitstream
- Pre-defined Vivado project for manual FPGA bitstream generation
- Firmware source code

■ Install Xilinx Vivado Design Suite: System Edition 2014.4

- Xilinx Vivado 2014.4
- Xilinx SDK 2014.4

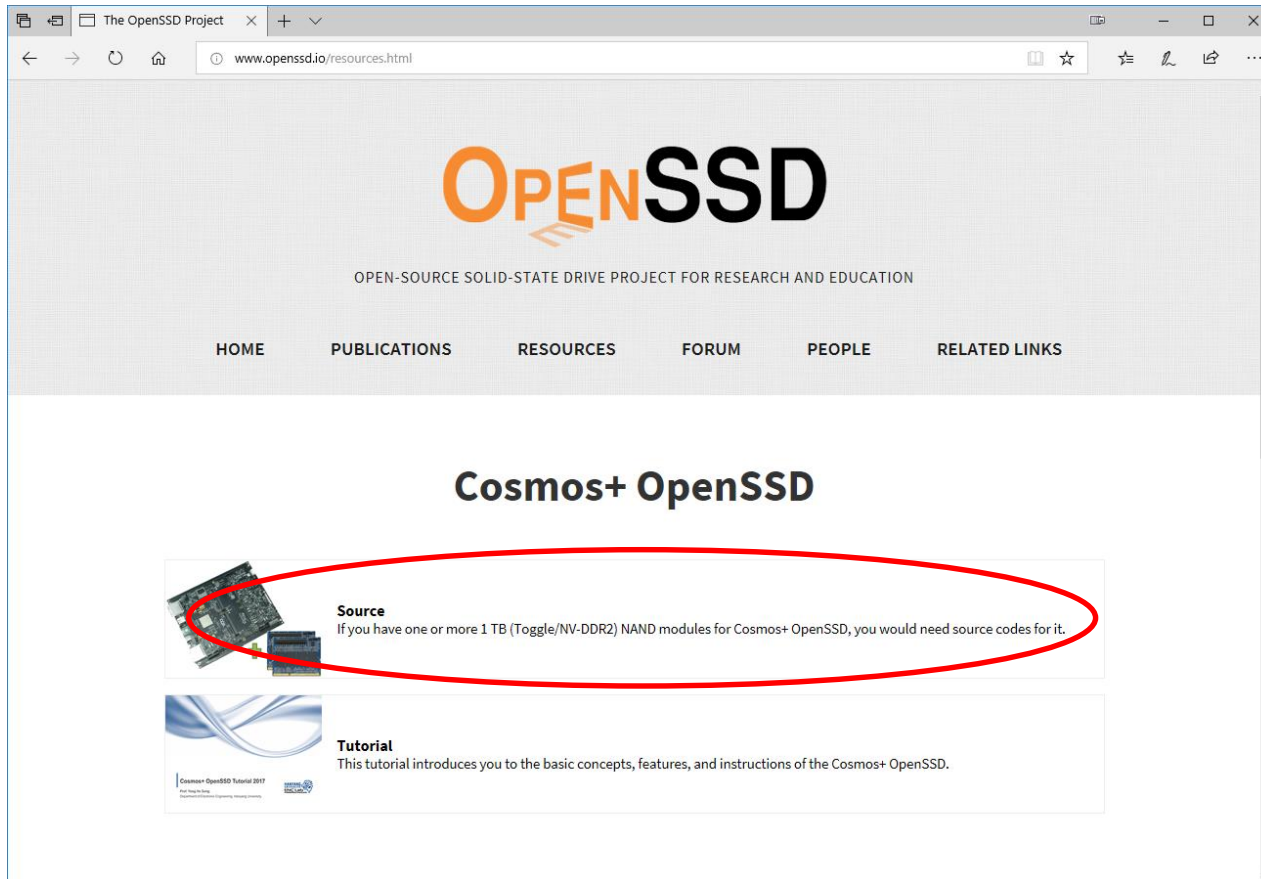
Download Materials (1 / 3)

- Go to the OpenSSD project site, and click “Resources”



Download Materials (2 / 3)

- Click “Source”



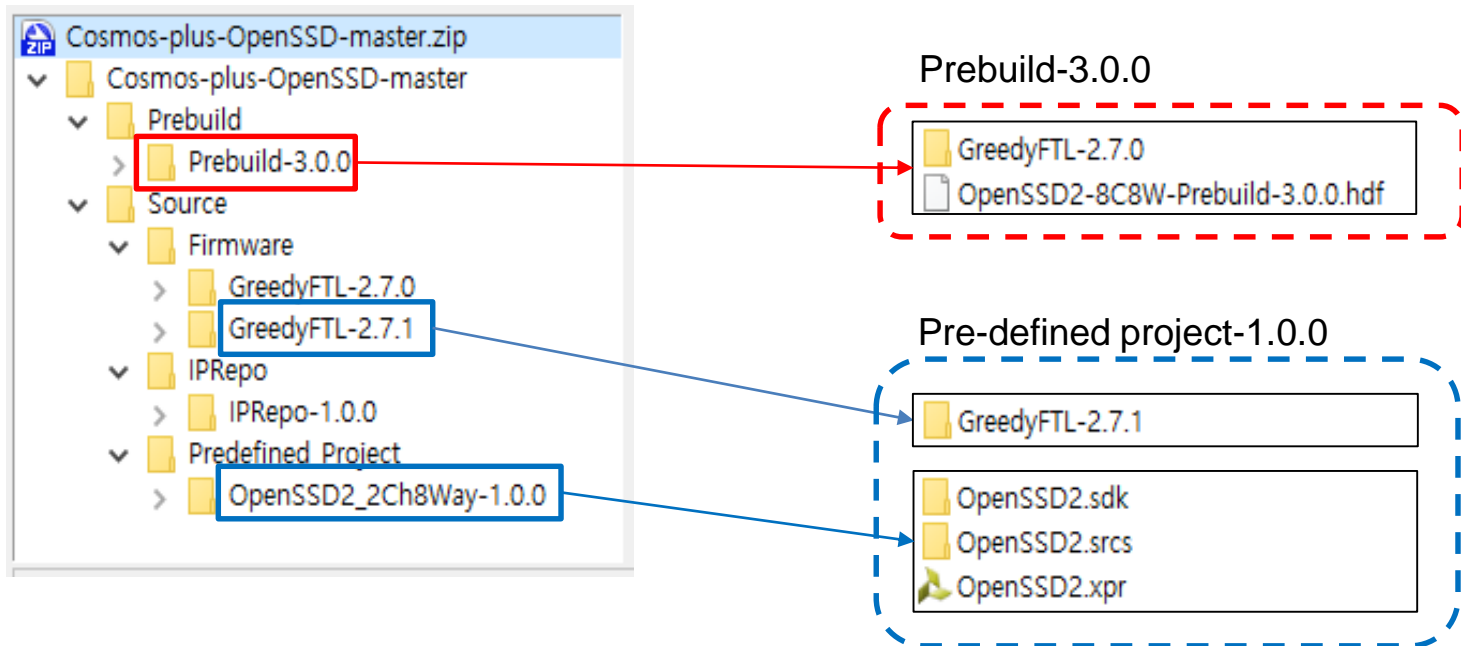
Download Materials (3 / 3)

- Click “Clone or download” -> “Download ZIP”

The screenshot shows the GitHub repository page for `Cosmos-OpenSSD / Cosmos-plus-OpenSSD`. The repository is under the `master` branch. A dropdown menu is open, showing options to clone or download the repository. The `Clone or download` button is circled in red, and a red arrow points to it. The `Download ZIP` button in the dropdown menu is also circled in red. The repository description is "Cosmos OpenSSD + Hardware and Software source distribution". The repository has 19 commits, 2 branches, 0 releases, 2 contributors, and is licensed under GPL-3.0. The repository files include `doc`, `project`, `source`, `support`, `LICENSE`, and `README.md`. The `README.md` file content is visible at the bottom of the page, showing the repository name and description.

Directory Tree of Downloaded Materials

- Materials include a prebuilt bitstream, a pre-defined project, and a firmware source code



Type of Bitstream and Firmware

| Bitstream Type | Ver. | Channel | Way | Bits / cell | Capacity |
|----------------|-------|---------|-----|-------------|-----------------|
| ● Prebuild | 3.0.0 | 8 | 8 | SLC / MLC | 1 TB / 2 TB |
| ● Predefined | 1.0.0 | 2 | 8 | SLC / MLC | 256 GB / 512 GB |

| Firmware Type | Ver. | Channel | Way | Bits / cell | Capacity |
|---------------|-------|---------|-----|-------------|----------|
| ● GreedyFTL | 2.5.0 | 8 | 8 | SLC | 1 TB |
| | 2.6.0 | | | | |
| | 2.7.0 | | | | |
| ● GreedyFTL | 2.7.1 | 2 | 8 | SLC | 256 GB |

Remarks on the Type of Bitstream

■ Prebuild type

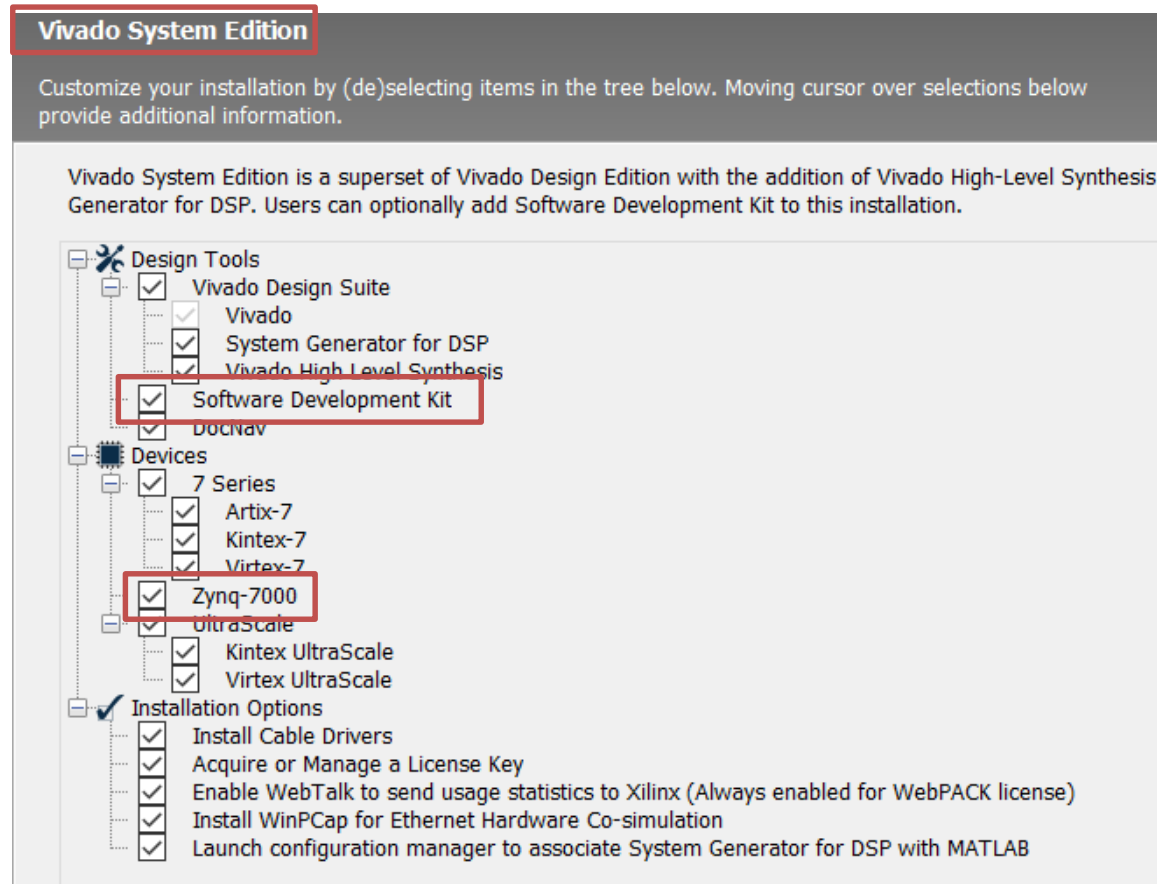
- A prebuilt bitstream is included, so you can skip bitstream generation steps
- Prebuild type is distributed as a hardware description file (.hdf) which consists of a FPGA bitstream, bitstream information, and an initialization code for CPU in Zynq FPGA

■ Pre-defined type

- bitstream is not included, so you should follow bitstream generation steps
- Pre-defined type is distributed as a vivado project file with register transfer level (RTL) source codes of intellectual properties (IPs) such as NVMe controller

Install Xilinx Vivado Design Suite

- Make sure that Vivado is system edition and that “Software Development Kit” and “Zynq-7000” are checked



Vivado System Edition

Customize your installation by (de)selecting items in the tree below. Moving cursor over selections below provide additional information.

Vivado System Edition is a superset of Vivado Design Edition with the addition of Vivado High-Level Synthesis Generator for DSP. Users can optionally add Software Development Kit to this installation.

- Design Tools
 - Vivado Design Suite
 - Vivado
 - System Generator for DSP
 - Vivado High Level Synthesis
 - Software Development Kit
 - DocNav
- Devices
 - 7 Series
 - Artix-7
 - Kintex-7
 - Virtex-7
 - Zynq-7000
 - UltraScale
 - Kintex UltraScale
 - Virtex UltraScale
- Installation Options
 - Install Cable Drivers
 - Acquire or Manage a License Key
 - Enable WebTalk to send usage statistics to Xilinx (Always enabled for WebPACK license)
 - Install WinPCap for Ethernet Hardware Co-simulation
 - Launch configuration manager to associate System Generator for DSP with MATLAB

Generating FPGA Bitstream for Pre-defined Project

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

Launch Xilinx Vivado

■ Open the predefined project included in “OpenSSD2_2Ch8Way-1.0.0”

| | | | |
|---------------|------------------|---------------------|------|
| OpenSSD2.sdk | 2017-01-09 오후... | 파일 폴더 | |
| OpenSSD2.srcs | 2017-01-09 오후... | 파일 폴더 | |
| OpenSSD2 | 2017-01-09 오후... | Vivado Project File | 13KB |



OpenSSD2 - [I:/Projects/OpenSSD2_2Ch8Way_Release/OpenSSD2.xpr] - Vivado 2014.4.1

Project Manager OpenSSD2

Project Summary

Project Settings

Project name: OpenSSD2
Project location: I:/Projects/OpenSSD2_2Ch8Way_Release
Product family: Zynq-7000
Project part: xc7z045ffg900-3
Top module name: OpenSSD2

Synthesis

Status: Not started
Messages: No errors or warnings
Part: xc7z045ffg900-3
Strategy: Vivado Synthesis Defaults
Constraints: synthesis

Implementation

Status: Not started
Messages: No errors or warnings
Part: xc7z045ffg900-3
Strategy: Performance_ExplorePostRoutePhysOpt
Constraints: implementation
Incremental comple: None

DRC Violations

DRC information is not available because it hasn't been run

Timing

Timing information is not available because it hasn't been run

Utilization

Utilization information is not available because it hasn't been run

Power

Power information is not available because it hasn't been run

Design Runs

| Name | Constraints | WNS | TNS | WHS | THS | TPWS | Failed Routes | LUT | FF | BRAM | DSP | Start | Elapsed | Status |
|---------|----------------|-----|-----|-----|-----|------|---------------|-----|----|------|-----|-------|---------|--------|
| synth_1 | synthesis | | | | | | | | | | | | | Nc |
| impl_1 | implementation | | | | | | | | | | | | | Nc |

Run Synthesis (1 / 2)

Click “Run Synthesis”

The screenshot shows the Vivado IDE interface for a project named 'OpenSSD2'. The 'Project Manager' window is open, displaying the 'Synthesis' section. The 'Run Synthesis' button is highlighted with a red arrow. A tooltip is visible over the button, stating: 'Run Synthesis Run synthesis on your project source files.'

The 'Project Summary' window is also open, showing the following details:

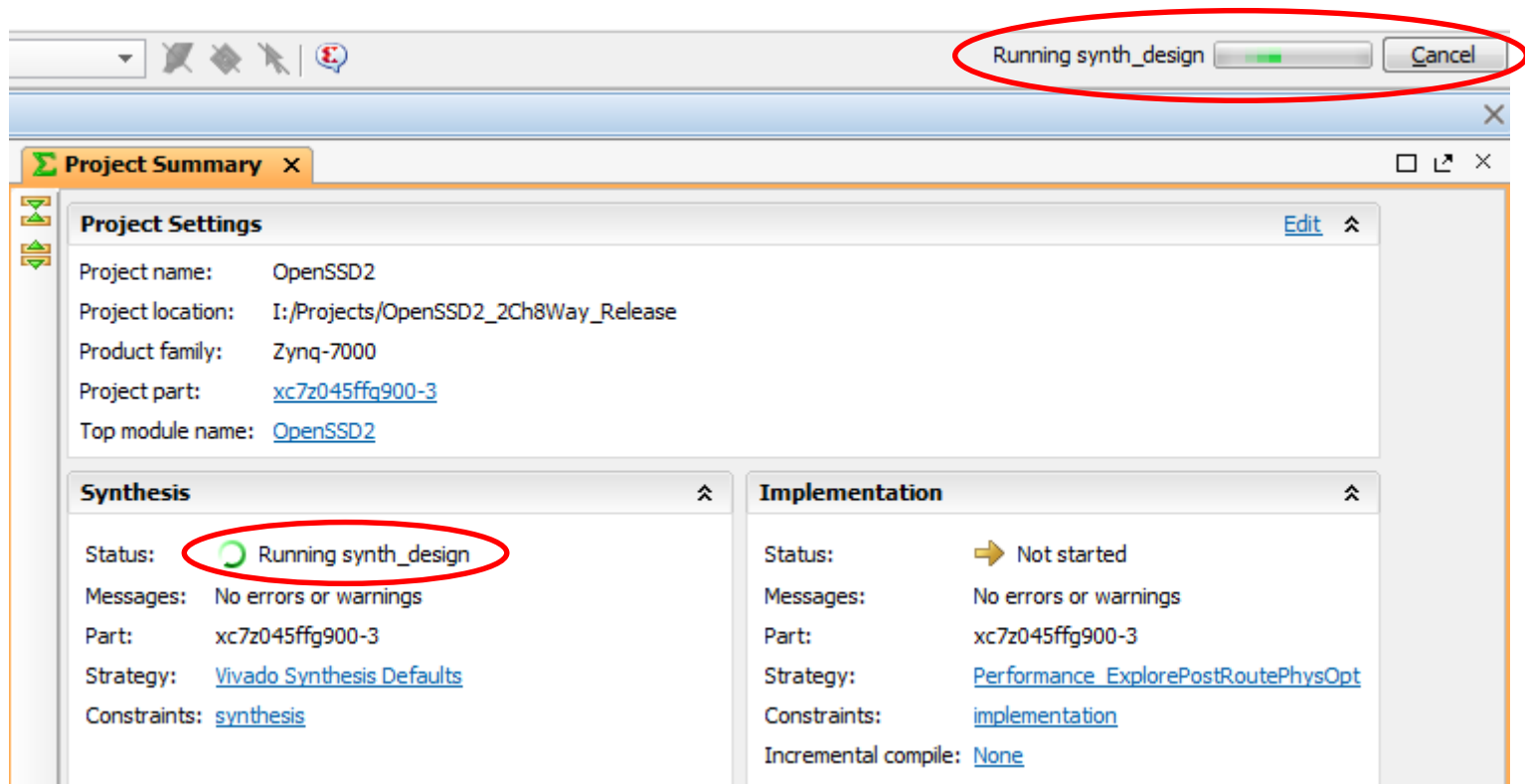
- Project Settings:** Project name: OpenSSD2, Project location: I:/Projects/OpenSSD2_2Ch8Way_Release, Product family: Zynq-7000, Project part: xc7z045ffg900-3, Top module name: OpenSSD2.
- Synthesis:** Status: Not started, Messages: No errors or warnings, Part: xc7z045ffg900-3, Strategy: Vivado Synthesis Defaults, Constraints: synthesis.
- Implementation:** Status: Not started, Messages: No errors or warnings, Part: xc7z045ffg900-3, Strategy: Performance_ExplorePostRoutePhysOpt, Constraints: implementation, Incremental compile: None.
- DRC Violations:** DRC information is not available because it hasn't been run.
- Timing:** Timing information is not available because it hasn't been run.
- Utilization:** Utilization information is not available because it hasn't been run.
- Power:** Power information is not available because it hasn't been run.

The 'Design Runs' table at the bottom shows the following data:

| Name | Constraints | WNS | TNS | WHS | THS | TPWS | Failed Routes | LUT | FF | BRAM | DSP | Start | Elapsed | Status |
|---------|----------------|-----|-----|-----|-----|------|---------------|-----|----|------|-----|-------|---------|-------------|
| synth_1 | synthesis | | | | | | | | | | | | | Not started |
| impl_1 | implementation | | | | | | | | | | | | | Not started |

Run Synthesis (2 / 2)

■ Synthesis is running...




The screenshot displays the Vivado IDE interface. At the top, a progress bar indicates the synthesis process is running, with the text "Running synth_design" and a green progress indicator. A red circle highlights this progress bar and the "Cancel" button. Below the progress bar, the "Project Summary" window is open, showing project settings and synthesis/implementation status.


Project Settings

- Project name: OpenSSD2
- Project location: I:/Projects/OpenSSD2_2Ch8Way_Release
- Product family: Zynq-7000
- Project part: xc7z045ffg900-3
- Top module name: OpenSSD2

Synthesis

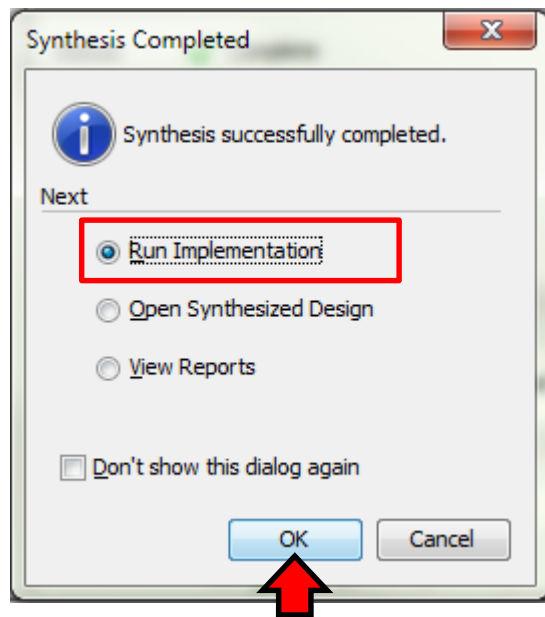
- Status:  Running synth_design
- Messages: No errors or warnings
- Part: xc7z045ffg900-3
- Strategy: [Vivado Synthesis Defaults](#)
- Constraints: [synthesis](#)

Implementation

- Status:  Not started
- Messages: No errors or warnings
- Part: xc7z045ffg900-3
- Strategy: [Performance_ExplorePostRoutePhysOpt](#)
- Constraints: [implementation](#)
- Incremental compile: [None](#)

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

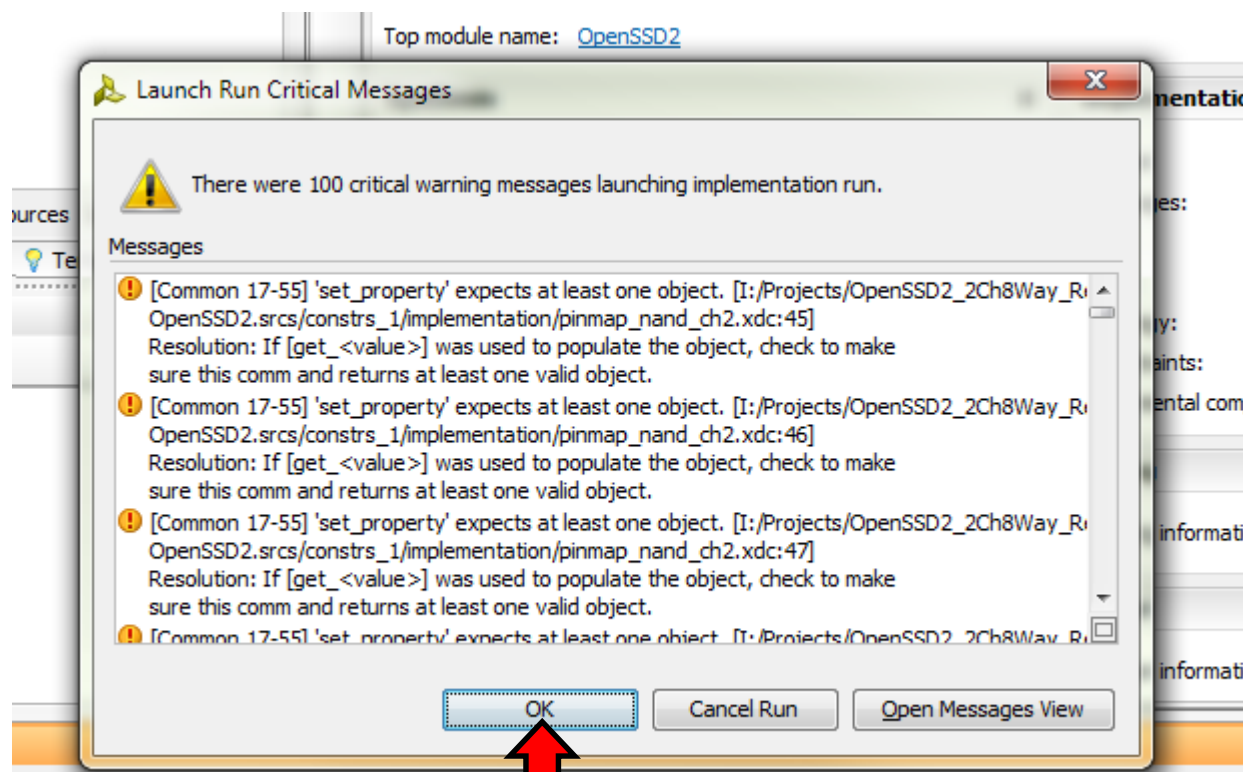
■ Implementation is running...

The screenshot displays the Vivado IDE interface during the implementation phase. At the top right, a progress bar labeled "Initializing Design" is shown with a green bar indicating progress, and a "Cancel" button is visible. Below this, the "Project Summary" window is open, showing the following details:

- Project Settings:**
 - Project name: OpenSSD2
 - Project location: I:/Projects/OpenSSD2_2Ch8Way_Release
 - Product family: Zynq-7000
 - Project part: [xc7z045ffg900-3](#)
 - Top module name: [OpenSSD2](#)
- Synthesis:**
 - Status: Complete
 - Messages: [6 critical warnings](#)
 [595 warnings](#)
 - Part: xc7z045ffg900-3
 - Strategy: [Vivado Synthesis Defaults](#)
 - Constraints: [synthesis](#)
- Implementation:**
 - Status: **Running Design Initialization**
 - Messages: No errors or warnings
 - Part: xc7z045ffg900-3
 - Strategy: [Performance ExplorePostRoutePhysOpt](#)
 - Constraints: [implementation](#)
 - Incremental compile: [None](#)

Warning Message

- The following critical messages appear when implementation is running, but you can ignore it



3-27] Running DRC with 2 threads

Implementation Complete

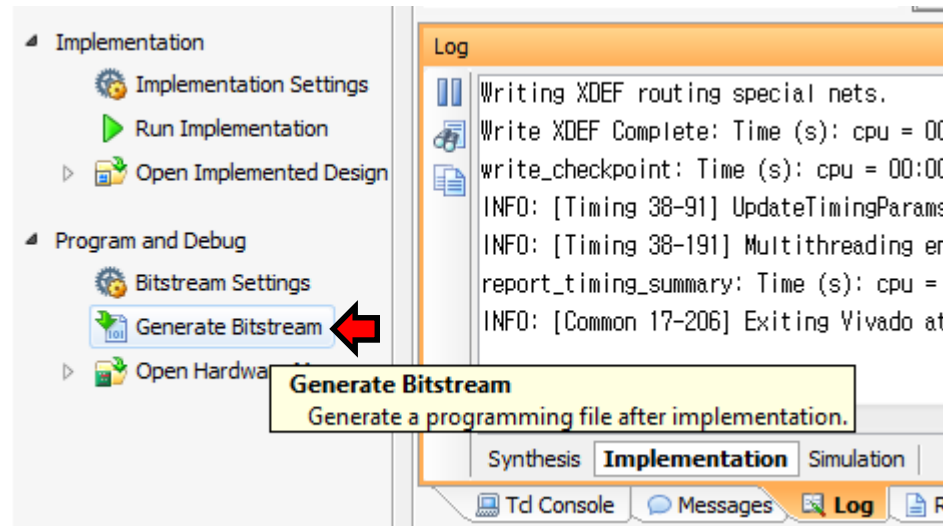
- Check the status of synthesis and implementation

The screenshot displays the 'Implementation Complete' window, which is divided into several sections:

- Project Settings:** Project name: OpenSSD2, Project location: I:/Projects/OpenSSD2_2Ch8Way_Release, Product family: Zynq-7000, Project part: xc7z045ffg900-3, Top module name: OpenSSD2.
- Synthesis:** Status: **Complete** (indicated by a green checkmark and circled in red). Messages: 0 critical warnings, 595 warnings. Part: xc7z045ffg900-3. Strategy: Vivado Synthesis Defaults. Constraints: synthesis.
- Implementation:** Status: **Complete** (indicated by a green checkmark and circled in red). Messages: 100 critical warnings, 304 warnings. Part: xc7z045ffg900-3. Strategy: Performance ExplorePostRoutePhysOpt. Constraints: implementation. Incremental compile: None. Summary button is visible.
- DRC Violations:** Summary: 0 errors, 0 critical warnings, 88 warnings.
- Timing - Post-Implementation:** Worst Negative Slack (WNS): 0.051 ns, Total Negative Slack (TNS): 0 ns, Number of Failing Endpoints: 0.

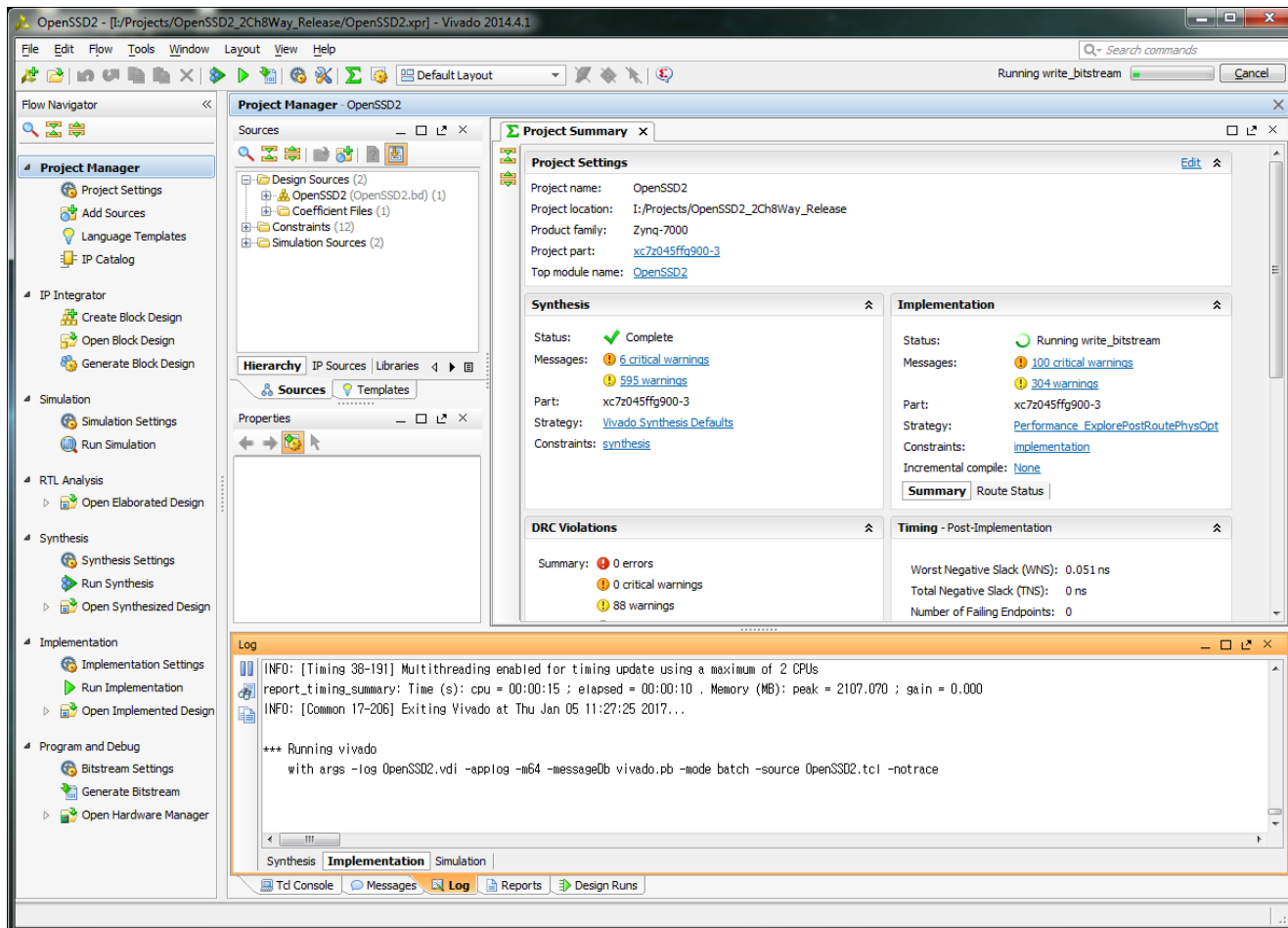
Generate Bitstream (1 / 2)

- Click “Generate Bitstream”



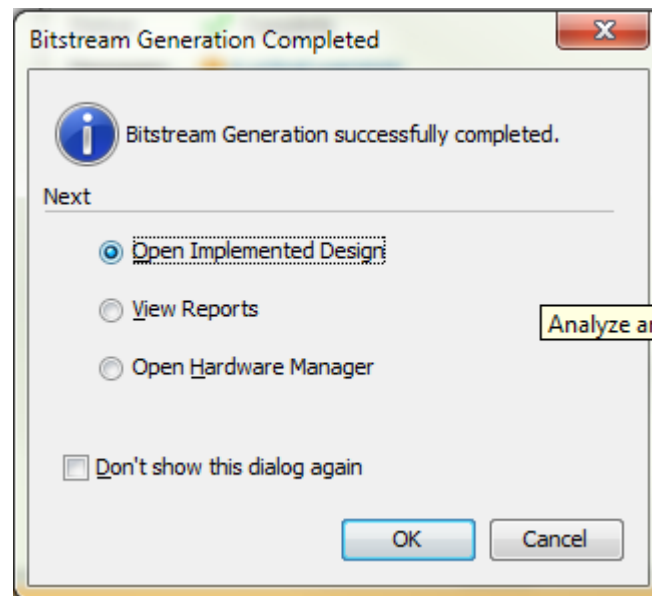
Generate Bitstream (2 / 2)

■ Generate bitstream is running...



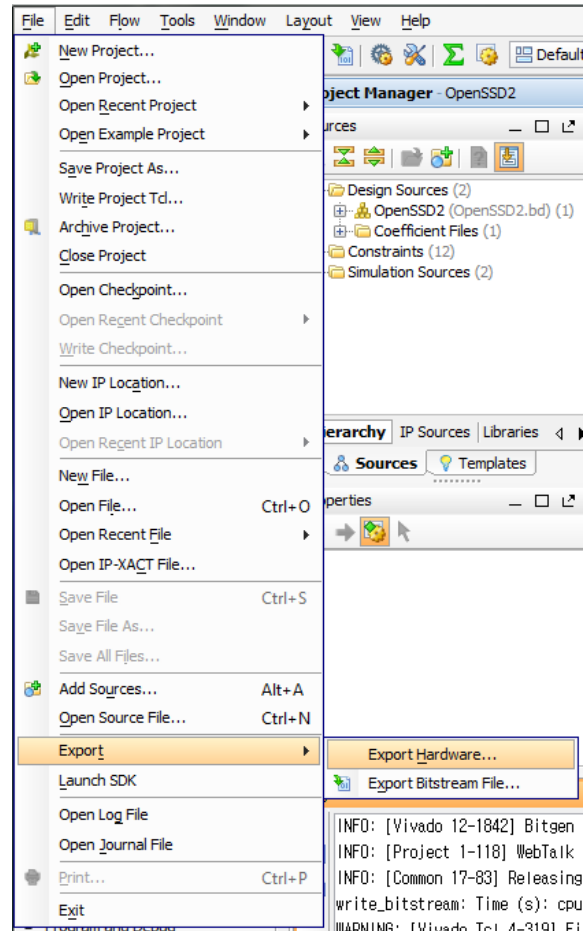
Bitstream Generation Complete

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



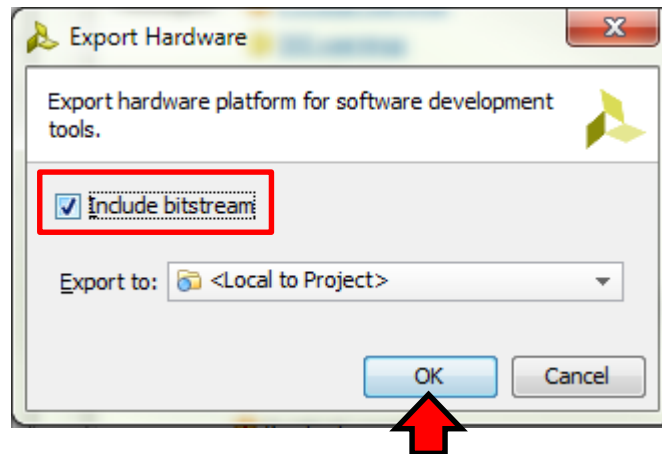
Export hardware (1 / 2)

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



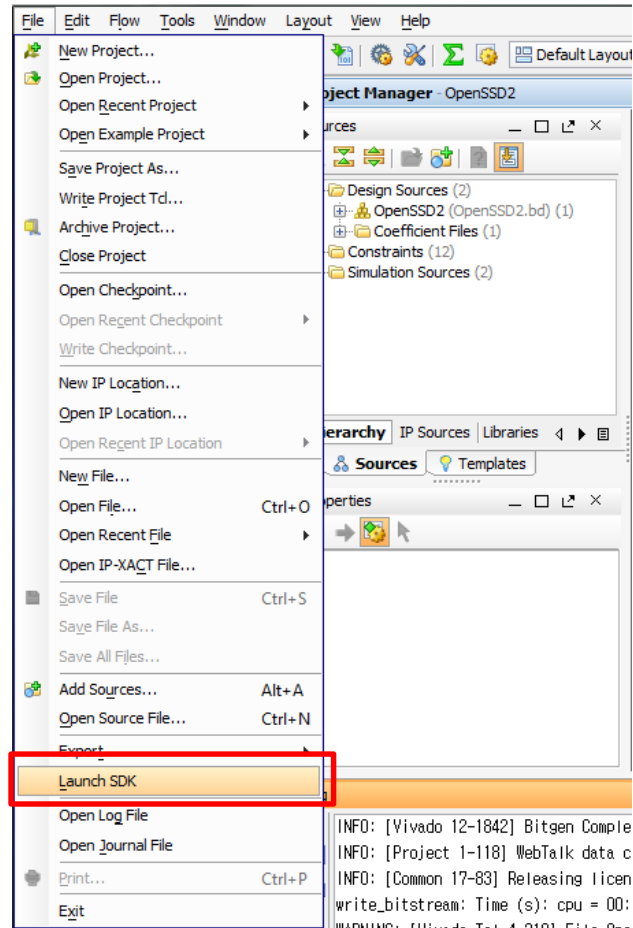
Export hardware (2 / 2)

- Select the “Include bitstream” and click OK



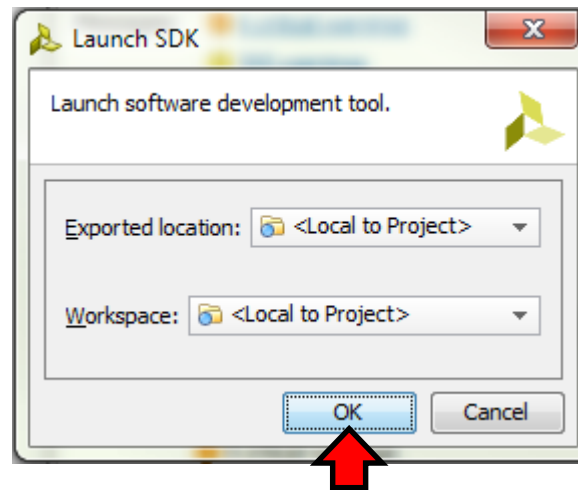
Launch SDK (1 / 4)

■ Go to File -> Launch SDK



Launch SDK (2 / 4)

- Click the OK button



Launch SDK (3 / 4)

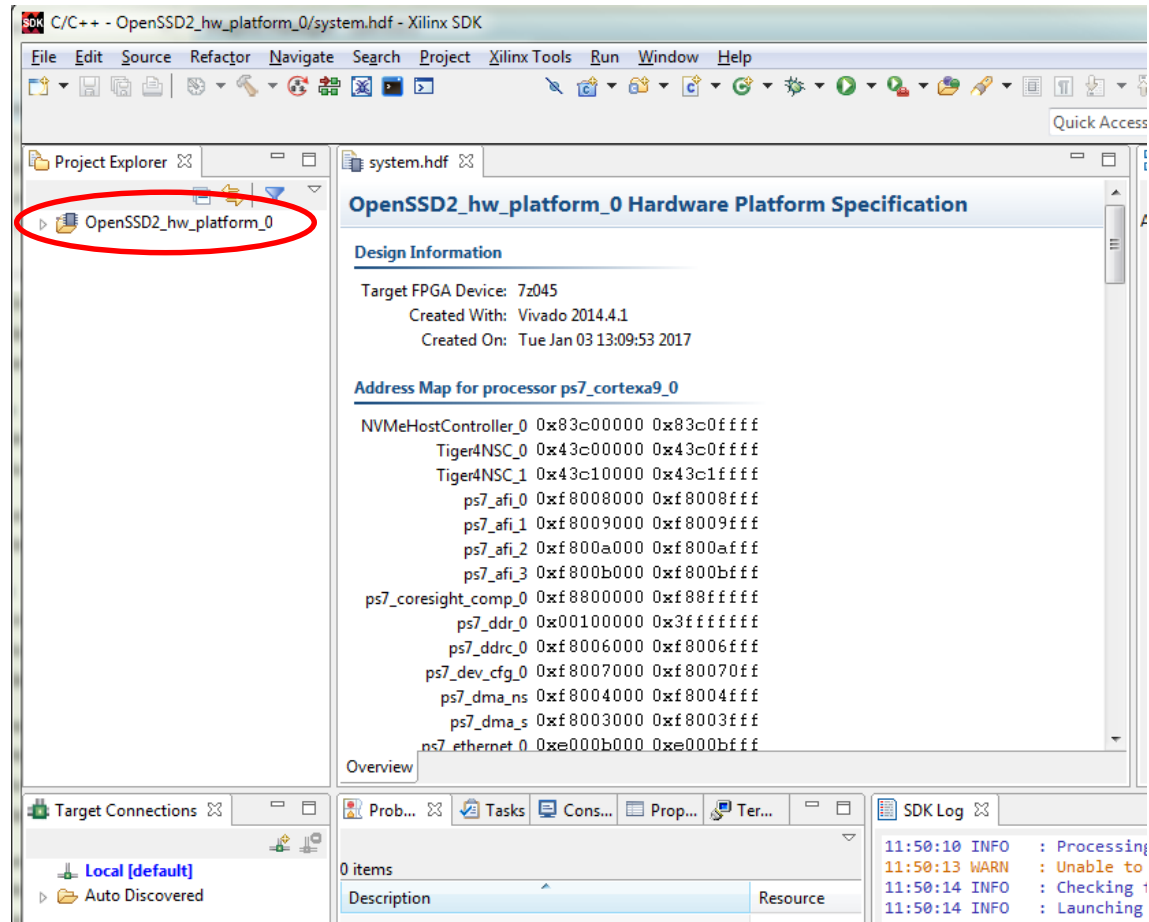
■ Then, SDK is launched



```
[Vivado 12-1842] Bitgen Completed Successfully.  
[Project 1-118] WebTalk data collection is enabled (User setting is ON. Install Setting is ON.).  
[Common 17-83] Releasing license: Implementation  
gitstream: Time (s): cpu = 00:02:43 ; elapsed = 00:02:43 . Memory (MB): peak = 2003.164 ; gain = 491.707  
3: [Vivado_Tcl 4-319] File OpenSSD2.mmi does not exist  
I:/Projects/OpenSSD2_2Ch8Way_Release/OpenSSD2.runs/impl_1/.Xil/Vivado-499416-aCentauri/HWH/OpenSSD2_bd.  
[Common 17-206] Exiting Vivado at Thu Jan 05 11:43:52 2017...
```

Launch SDK (4 / 4)

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

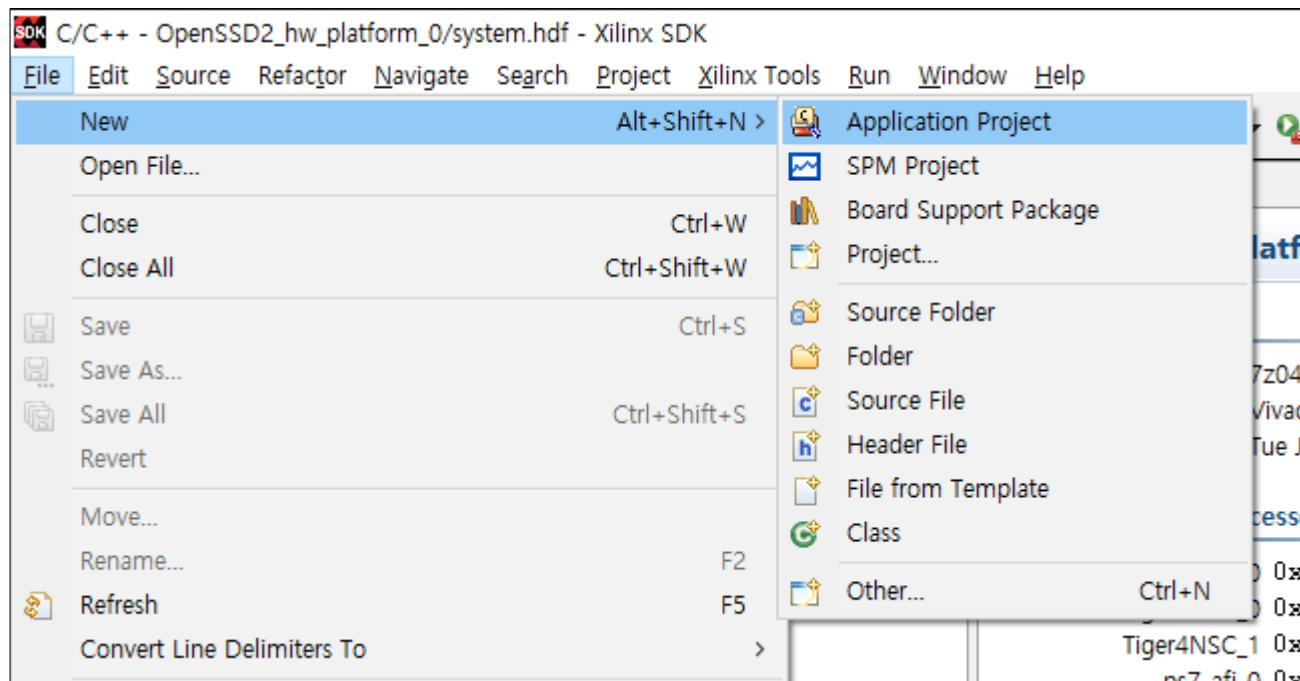


Building Firmware for Pre-defined Project

1. **Create a new application project**
2. **Add source codes**
3. **Build firmware source codes**

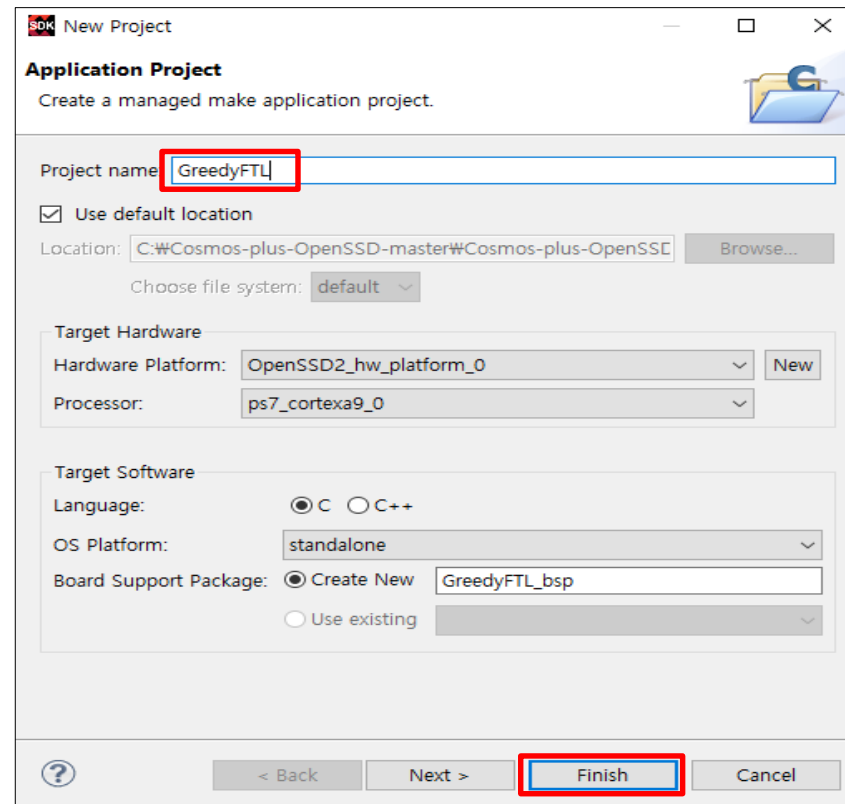
Create a New Application Project

- Go to File -> New -> Application Project



Specify the Project Name

- Fill in the project name and click “Next”



SDK New Project

Application Project
Create a managed make application project.

Project name:

Use default location

Location:

Choose file system:

Target Hardware

Hardware Platform:

Processor:

Target Software

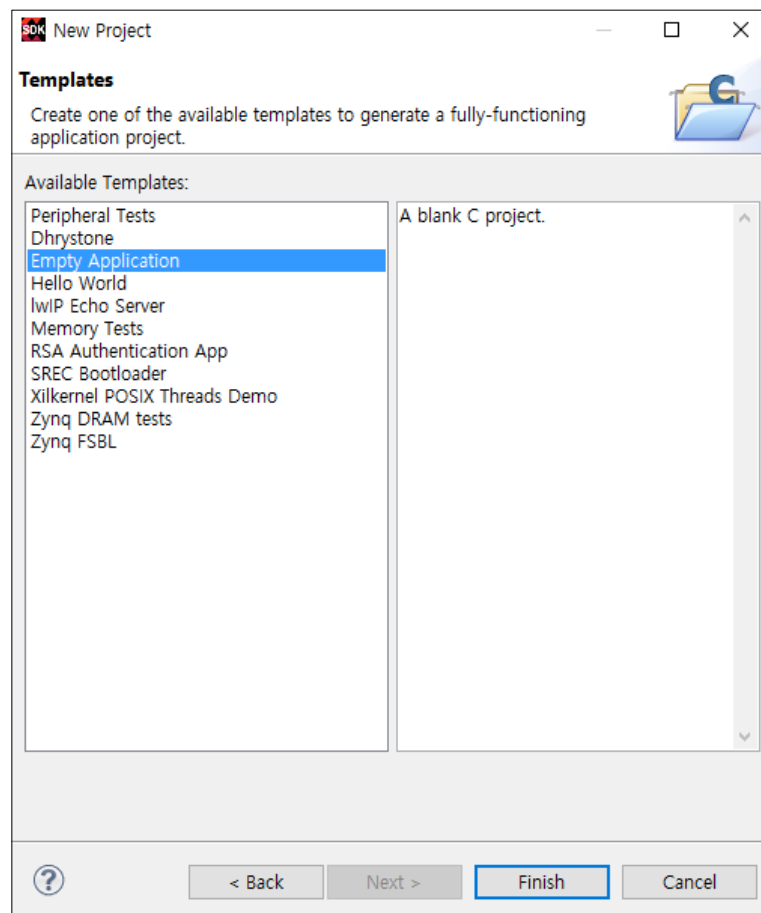
Language: C C++

OS Platform:

Board Support Package: Create New Use existing

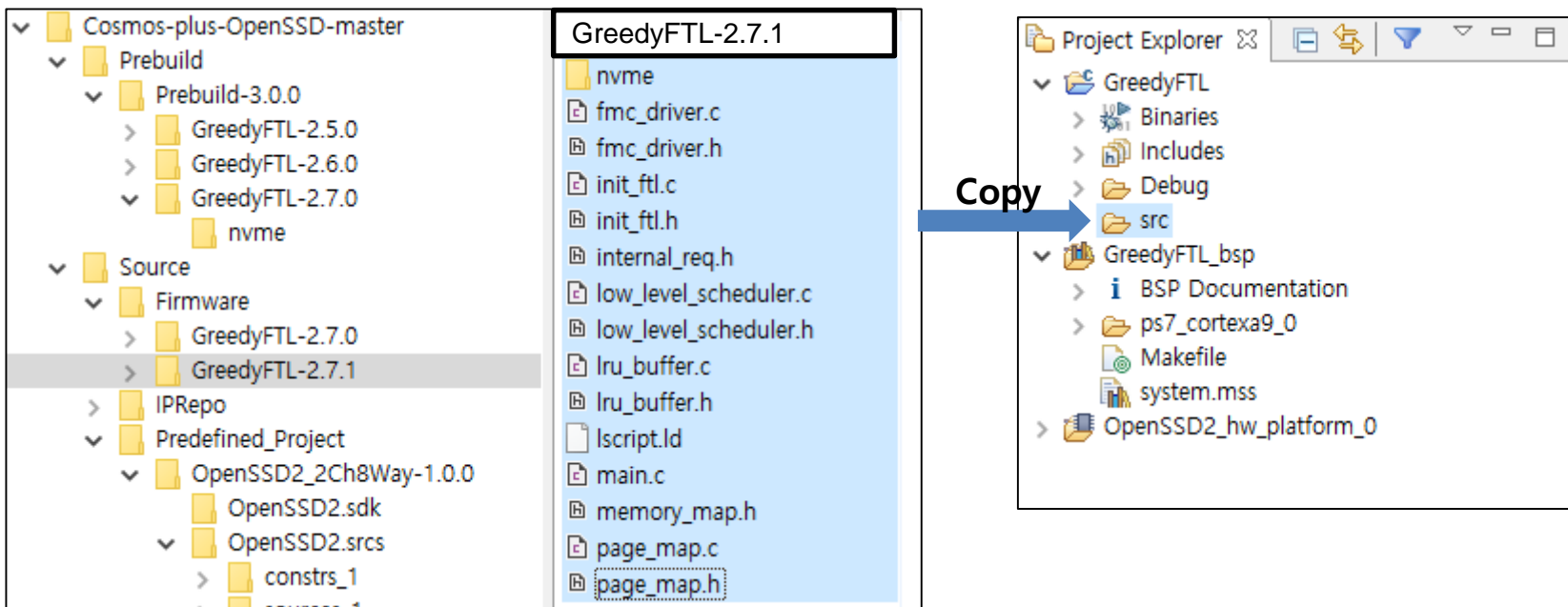
Select a Project Template

- Select an empty application and finish this template wizard



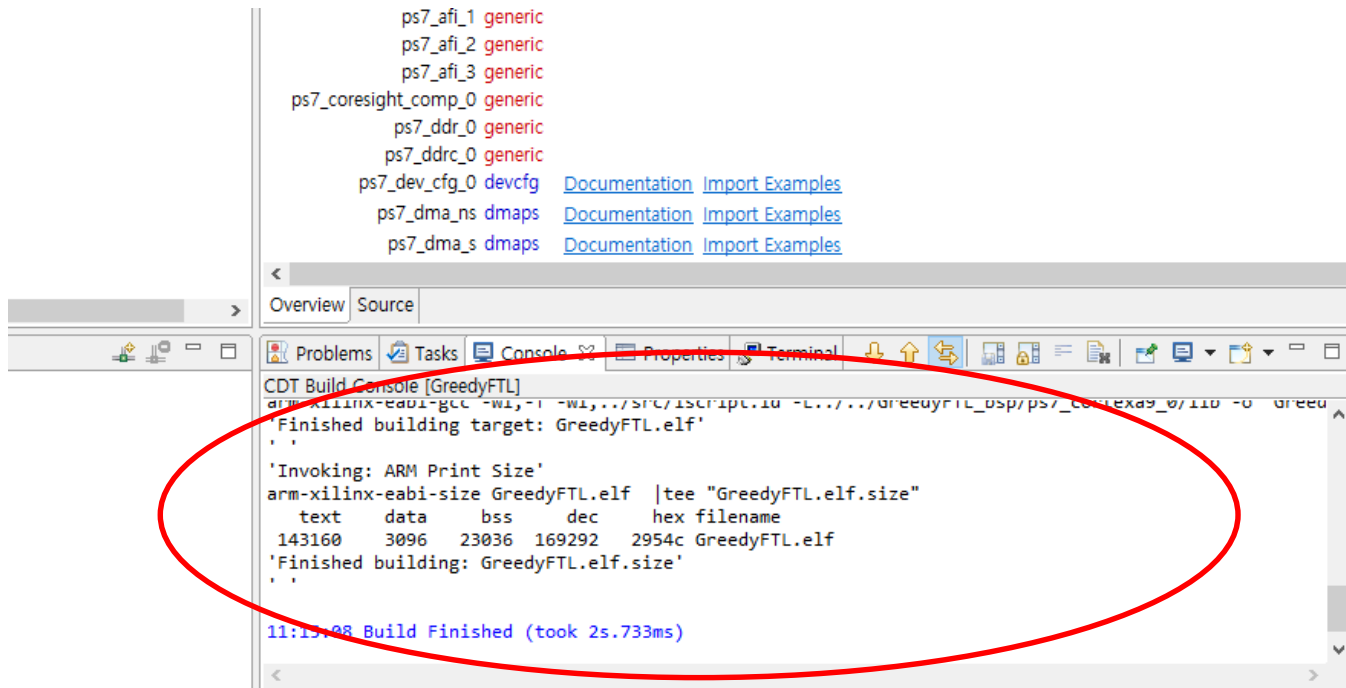
Add Source Code

- Copy GreedyFTL source files to “src” folder in project explorer



Build Firmware (1 / 2)

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



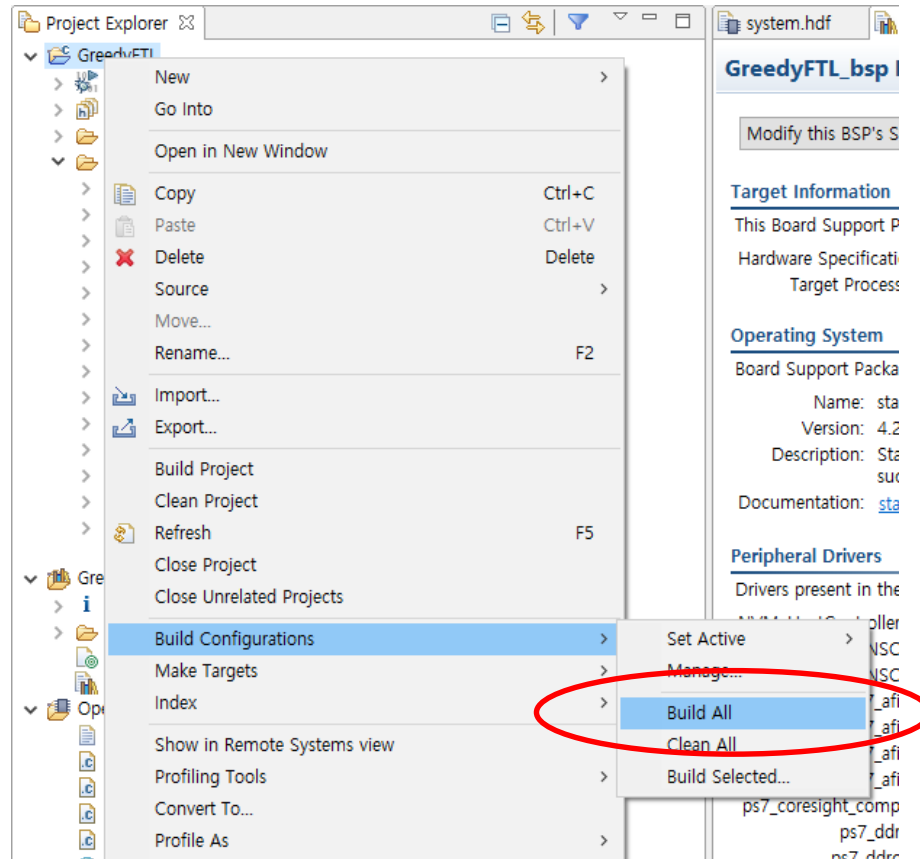
```
ps7_afi_1 generic
ps7_afi_2 generic
ps7_afi_3 generic
ps7_coresight_comp_0 generic
ps7_ddr_0 generic
ps7_ddrc_0 generic
ps7_dev_cfg_0 devcfg Documentation Import Examples
ps7_dma_ns dmaps Documentation Import Examples
ps7_dma_s dmaps Documentation Import Examples

CDT Build Console [GreedyFTL]
arm-xilinx-eabi-gcc -wi,-l-wi,../src/iscrpt.ld -L../greedyFTL_bsp/ps7_cortexa9_0/110 -o GreedyFTL.elf
Finished building target: GreedyFTL.elf
'Invoking: ARM Print Size'
arm-xilinx-eabi-size GreedyFTL.elf |tee "GreedyFTL.elf.size"
  text  data  bss  dec  hex filename
143160  3096  23036 169292 2954c GreedyFTL.elf
'Finished building: GreedyFTL.elf.size'

11:15:08 Build Finished (took 2s.733ms)
```

Build Firmware (2 / 2)

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

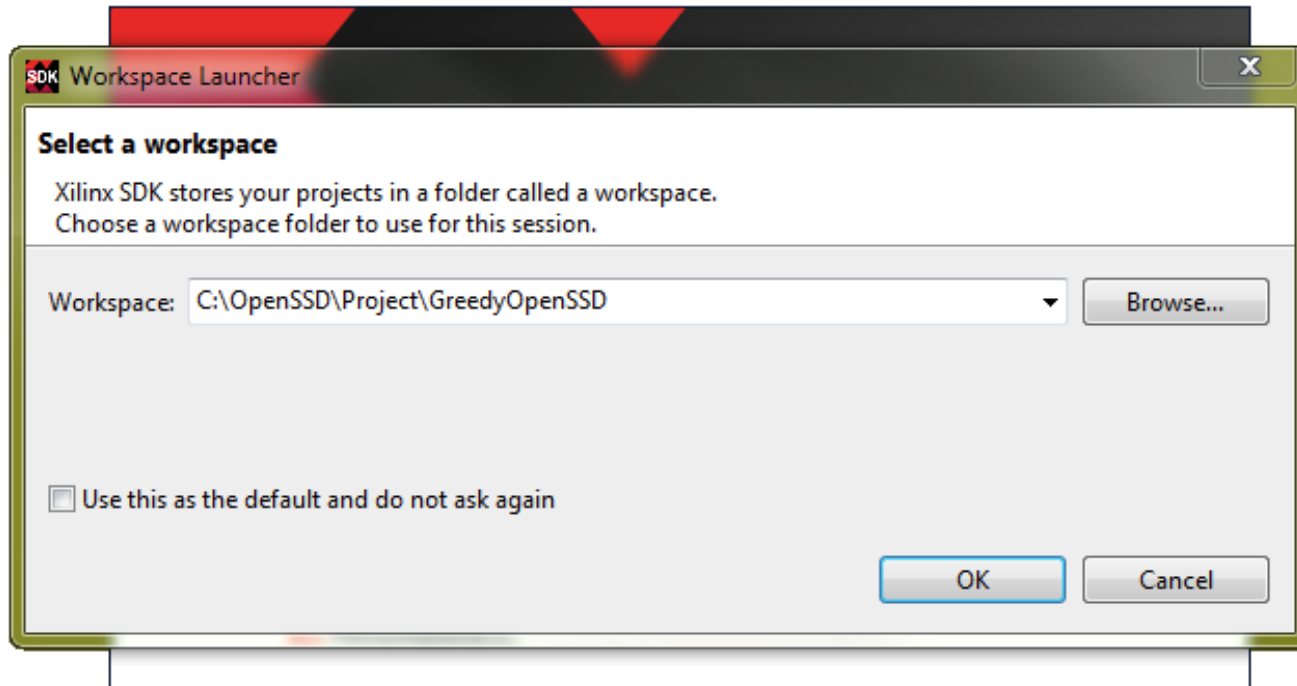


Building Firmware for Prebuild Bitstream

- 1. Create a workspace directory and a new application project**
- 2. Set a hardware platform**
- 3. Add source codes**
- 4. Build firmware source codes**

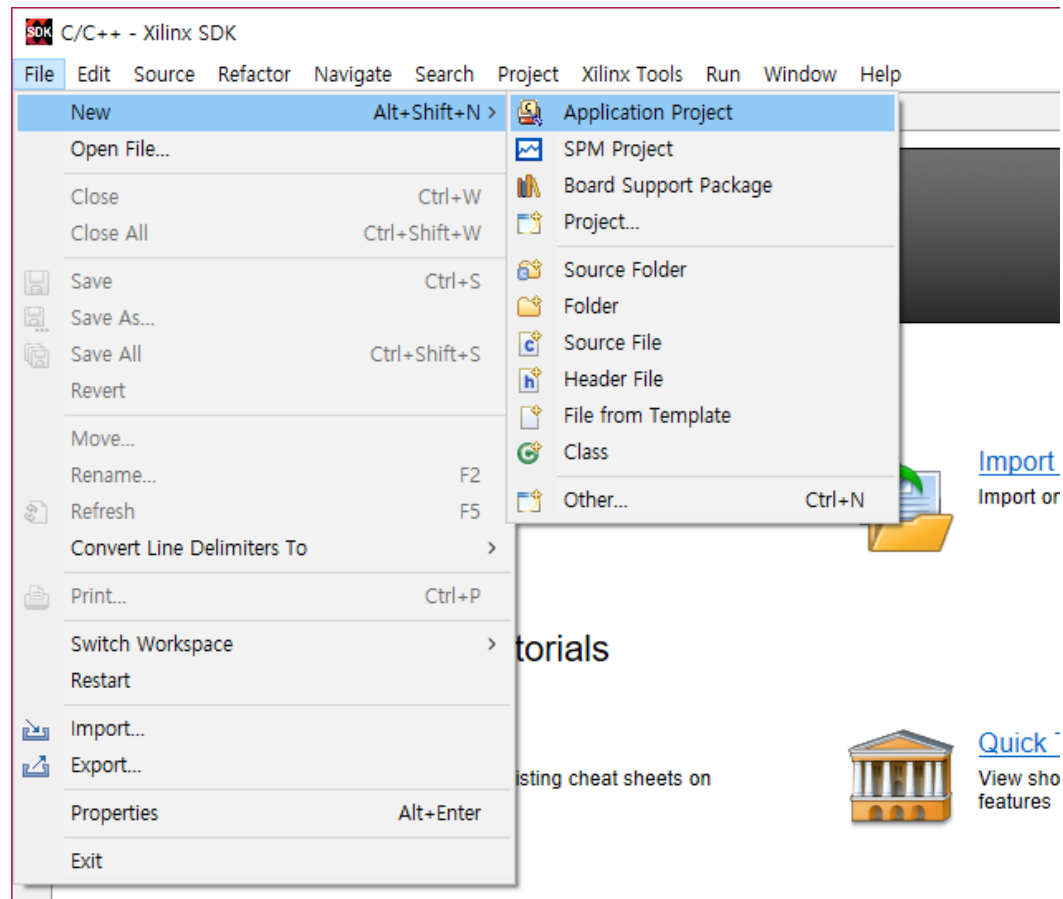
Create a Workspace Directory

- Launch Xilinx SDK and designate the workspace



Create a New Application Project

- Go to File -> New -> Application Project



Import the Prebuild Bitstream from HDF (1 / 2)

- Press “New” to register the hardware description file (HDF)

SDK New Project

Application Project
Project name must be specified

Project name:

Use default location
Location:
Choose file system:

Target Hardware
Hardware Platform:
Processor:

Target Software
Language: C C++
OS Platform:
Board Support Package: Create New
 Use existing

Import the Prebuild Bitstream from HDF (2 / 2)

- Name the hardware project and specify the path of the HDF

SDK New Hardware Project

New Hardware Project
Create a new Hardware Project.

Project name:

Use default location

Location:

Choose file system: ▾

Target Hardware Specification
Provide the path to the hardware specification file exported from Vivado.
This file usually resides in SDK/SDK_Export/hw folder relative to the Vivado project location.
The specification file and associated bitstream content will be copied into the workspace.

Finish the Project Wizard

■ Name the application project and finish this project wizard

Application Project
Create a managed make application project.

Project name:

Use default location

Location:

Choose file system:

Target Hardware

Hardware Platform:

Processor:

Target Software

Language: C C++

OS Platform:

Board Support Package: Create New Use existing



Templates
Create one of the available templates to generate a fully-functioning application project.

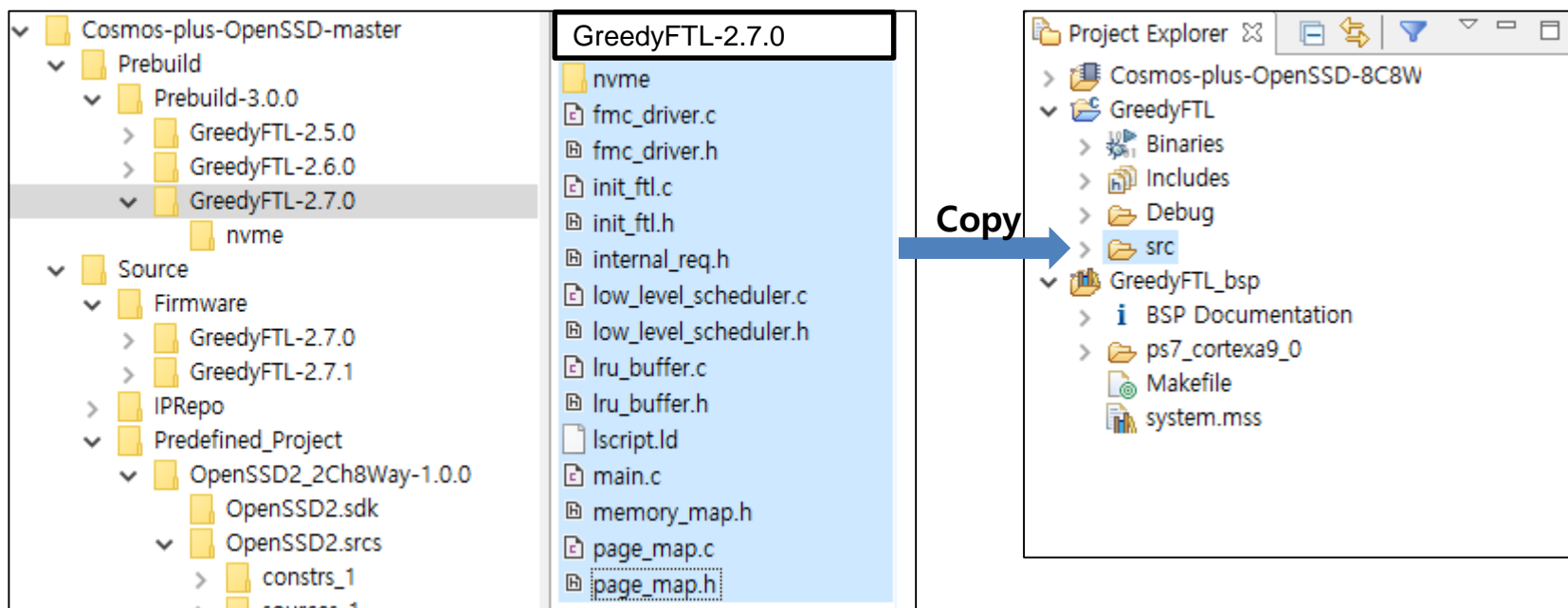
Available Templates:

- Peripheral Tests
- Dhrystone
- Empty Application**
- Hello World
- IwIP Echo Server
- Memory Tests
- RSA Authentication App
- SREC Bootloader
- Xilkernel POSIX Threads Demo
- Zynq DRAM tests
- Zynq FSBL

A blank C project.

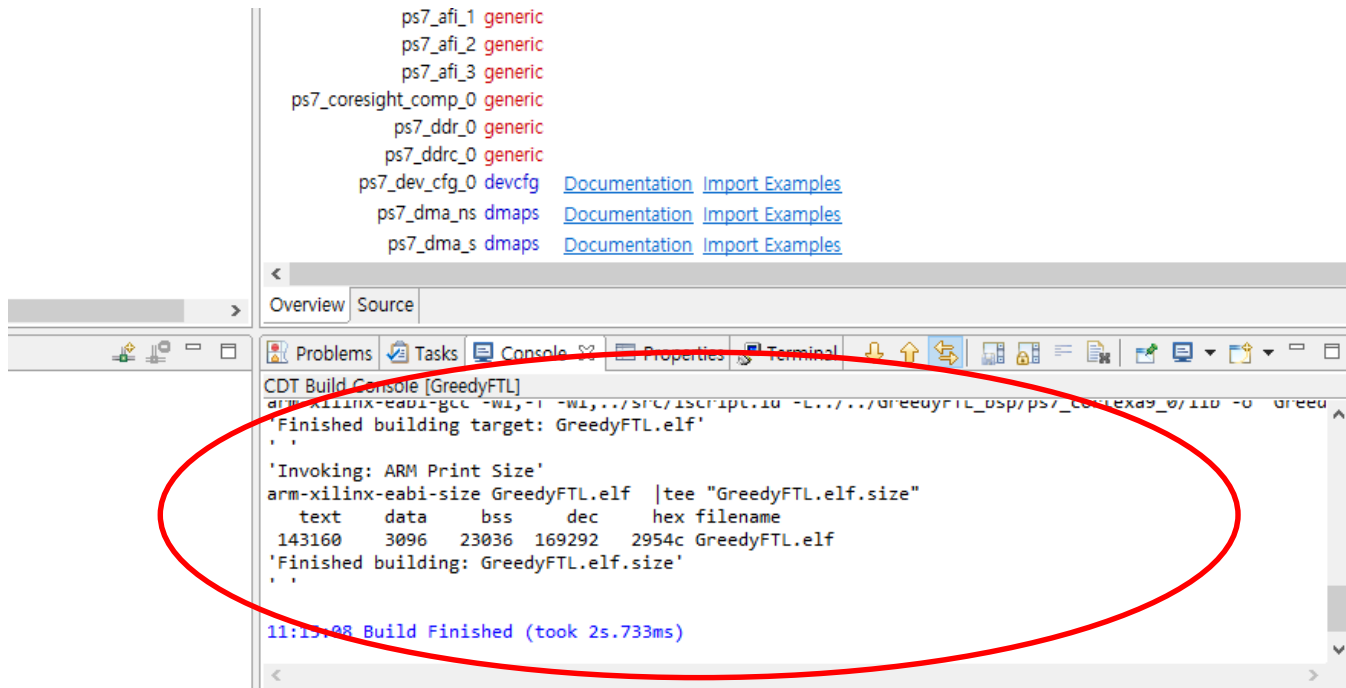
Add Source Code

- Copy GreedyFTL source files to “src” folder in project explorer



Build Firmware (1 / 2)

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



```
ps7_afi_1 generic
ps7_afi_2 generic
ps7_afi_3 generic
ps7_coresight_comp_0 generic
ps7_ddr_0 generic
ps7_ddrc_0 generic
ps7_dev_cfg_0 devcfg Documentation Import Examples
ps7_dma_ns dmaps Documentation Import Examples
ps7_dma_s dmaps Documentation Import Examples
```

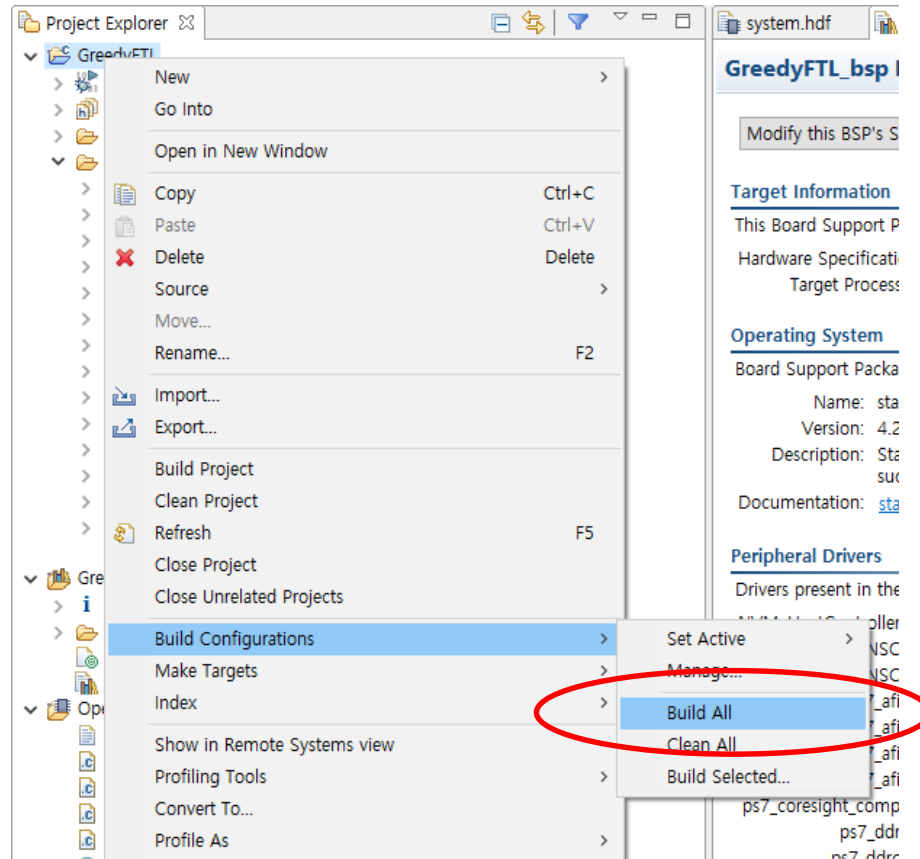
Overview Source

Problems Tasks Console Properties Terminal

CDT Build Console [GreedyFTL]
arm-xilinx-eabi-gcc -wi,-l-wi,../src/iscrpt.ld -L../greedyFTL_bsp/ps7_cortexa9_0/110 -o GreedyFTL.elf
'Finished building target: GreedyFTL.elf'
'
'Invoking: ARM Print Size'
arm-xilinx-eabi-size GreedyFTL.elf |tee "GreedyFTL.elf.size"
text data bss dec hex filename
143160 3096 23036 169292 2954c GreedyFTL.elf
'Finished building: GreedyFTL.elf.size'
'
11:15:08 Build Finished (took 2s.733ms)

Build Firmware (2 / 2)

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

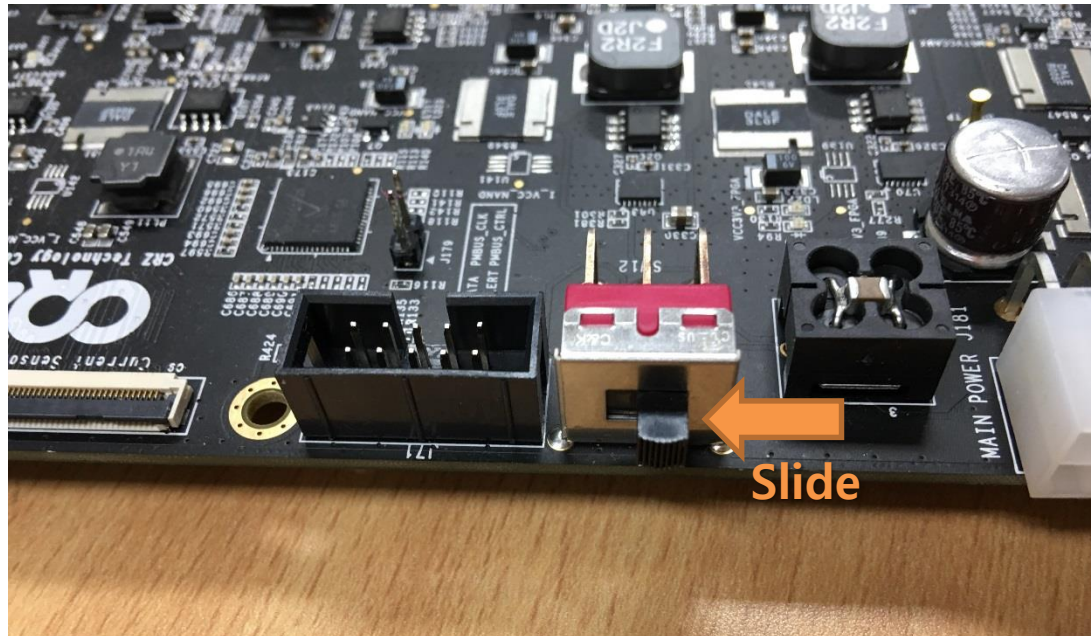


Preparing for Operating Cosmos+ OpenSSD

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

Power on the Platform Board

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



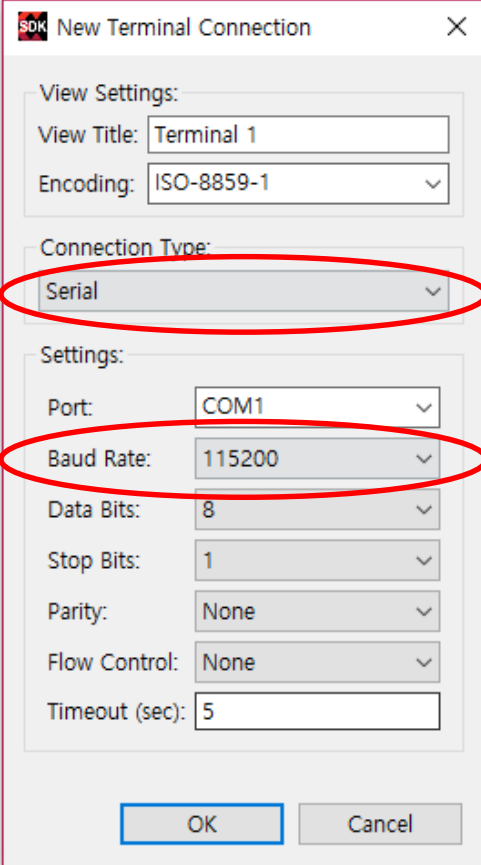
Configure UART

- In SDK, go to Terminal -> New Terminal Connection as shown below

The screenshot displays the SDK IDE interface. On the left, the file explorer shows a project structure with folders like 'src' and 'GreedyFTL_bsp'. Below it, the 'Target Connections' panel shows 'Local [default]' and 'Auto Discovered'. The main workspace is divided into two panes. The top pane shows the 'Board Support Package' information, including the hardware specification, target processor, operating system details (standalone v4.2), and a list of peripheral drivers (NVMeHostController_0, Tiger4NSC_0-4). The bottom pane shows the 'Terminal 1' view, which is currently empty. A context menu is open over the 'Terminal 1' tab, with two red arrows pointing to the 'New Terminal Connection in Current View...' and 'New Terminal View' options.

Configure UART

- Set “Connection Type” and “Baud Rate” to serial and 115200, respectively



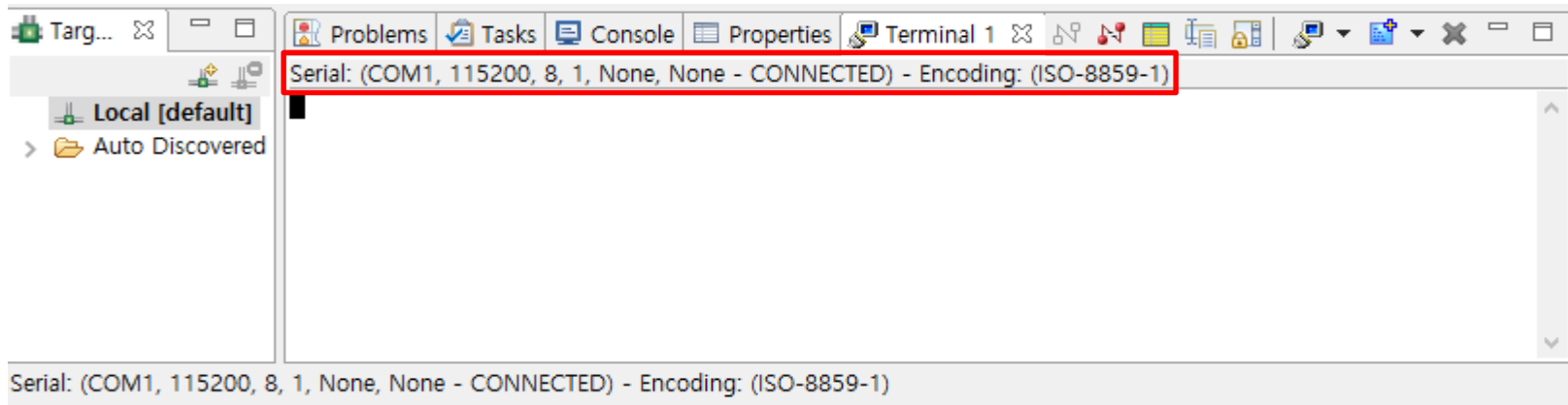
The screenshot shows a dialog box titled "New Terminal Connection" with the following settings:

- View Settings:
 - View Title: Terminal 1
 - Encoding: ISO-8859-1
- Connection Type: Serial
- Settings:
 - Port: COM1
 - Baud Rate: 115200
 - Data Bits: 8
 - Stop Bits: 1
 - Parity: None
 - Flow Control: None
 - Timeout (sec): 5

Buttons: OK, Cancel

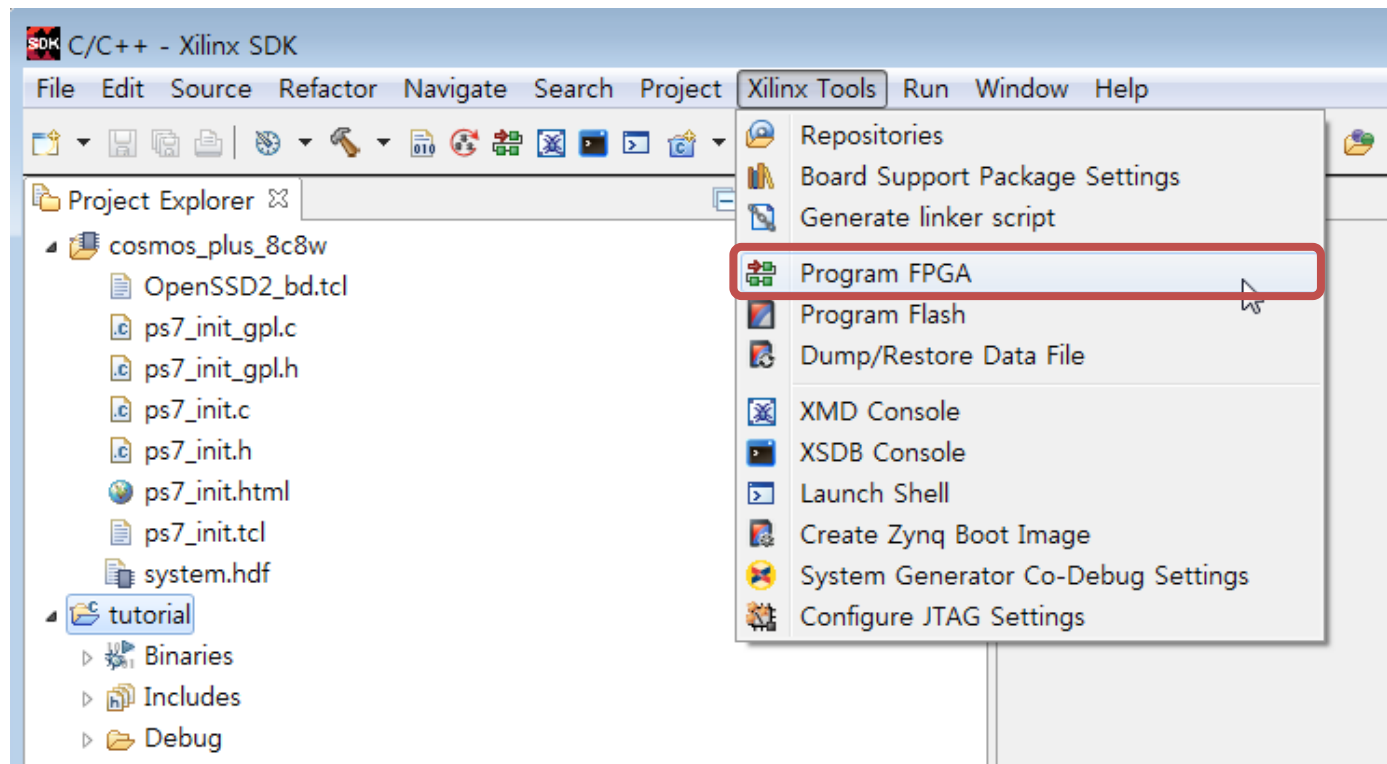
Configure UART

- If then, UART is connected as shown below



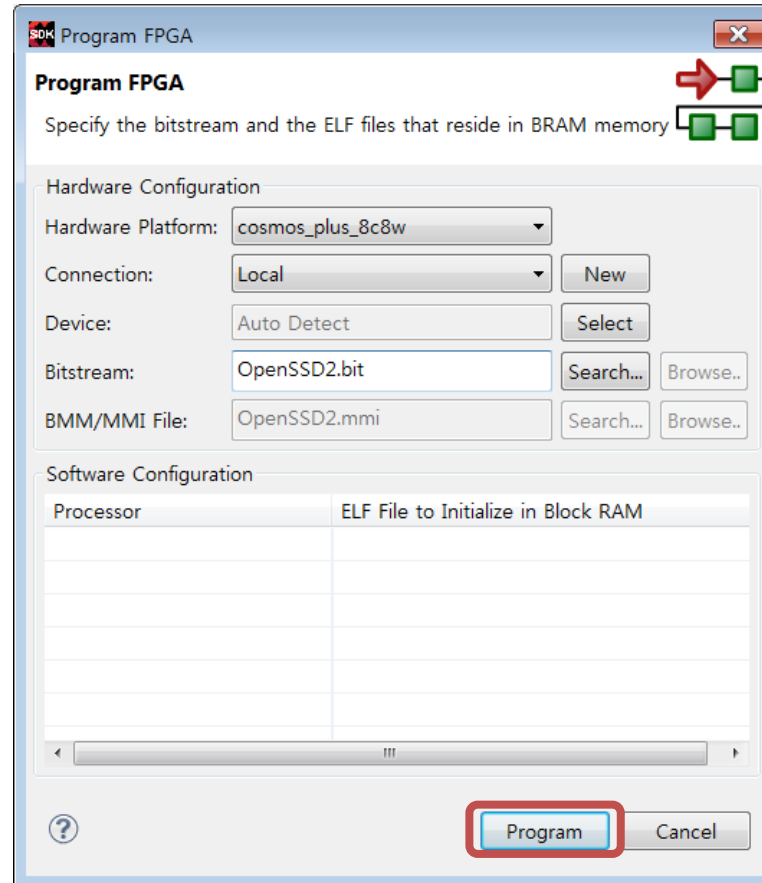
Program FPGA (1 / 4)

- Click “Xilinx Tools” -> click “Program FPGA”



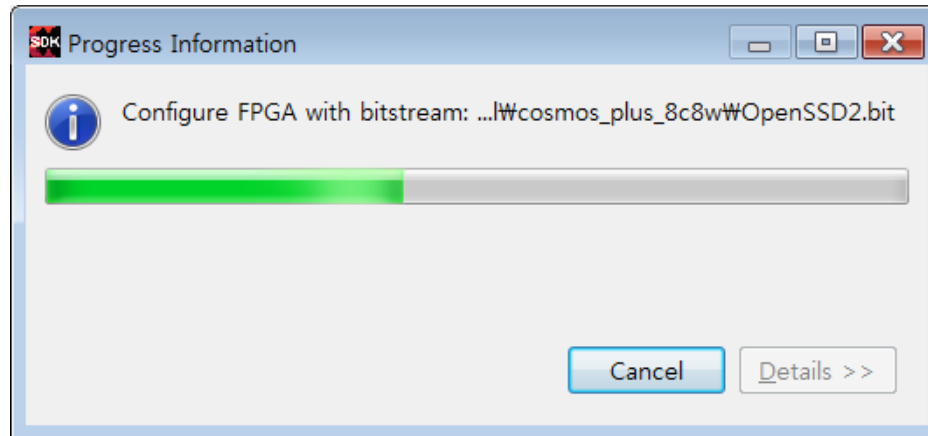
Program FPGA (2 / 4)

- Click “Program” to program FPGA



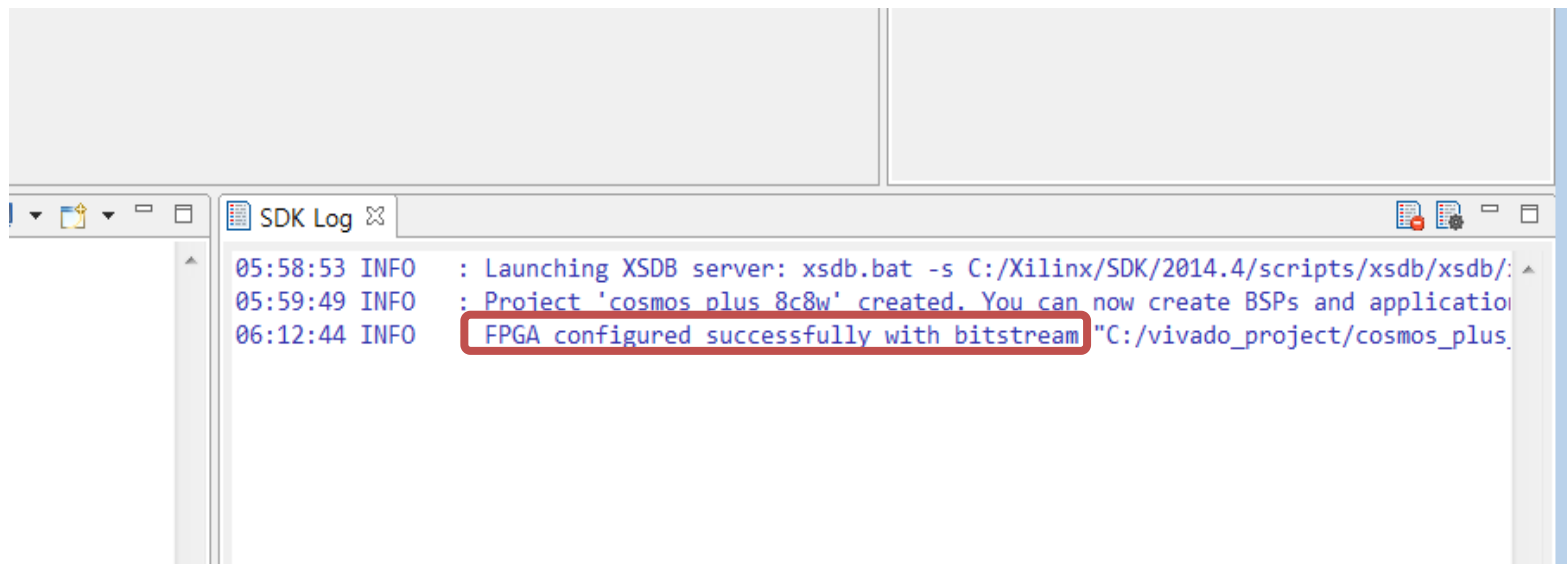
Program FPGA (3 / 4)

- Hang on a second



Program FPGA (4 / 4)

- Check FPGA programming done successfully



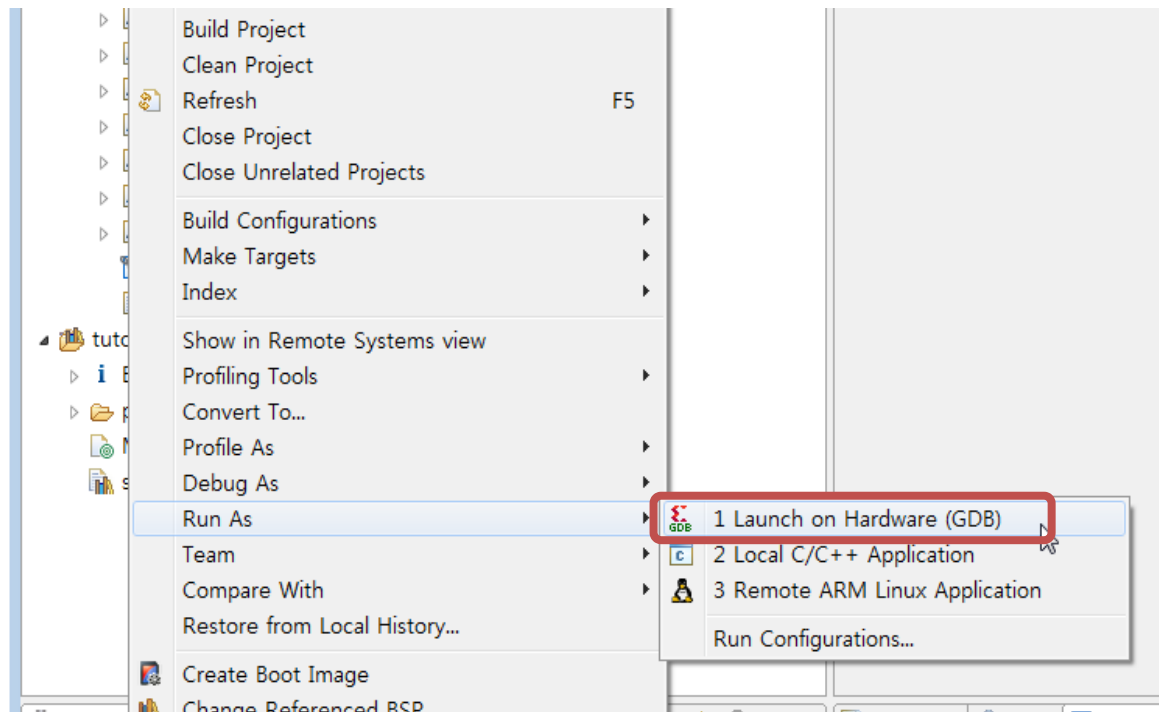
The screenshot shows a window titled "SDK Log" with the following log entries:

```
05:58:53 INFO : Launching XSDB server: xsdb.bat -s C:/Xilinx/SDK/2014.4/scripts/xsdb/xsdb/
05:59:49 INFO : Project 'cosmos plus 8c8w' created. You can now create BSPs and application
06:12:44 INFO : FPGA configured successfully with bitstream "C:/vivado_project/cosmos_plus
```

The text "FPGA configured successfully with bitstream" is highlighted with a red box.

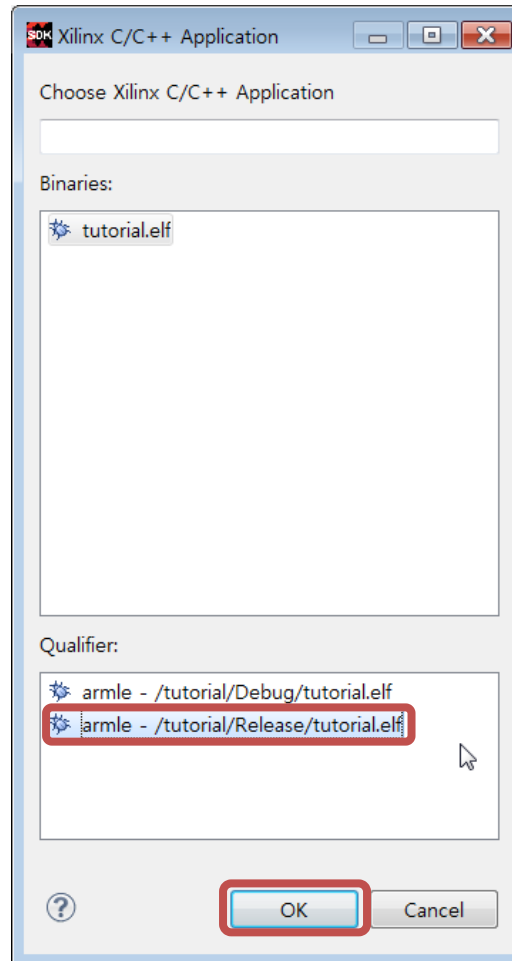
Execute Firmware (1 / 3)

- Right click on the application project -> “Run As” -> click “1 Launch on Hardware (GDB)”



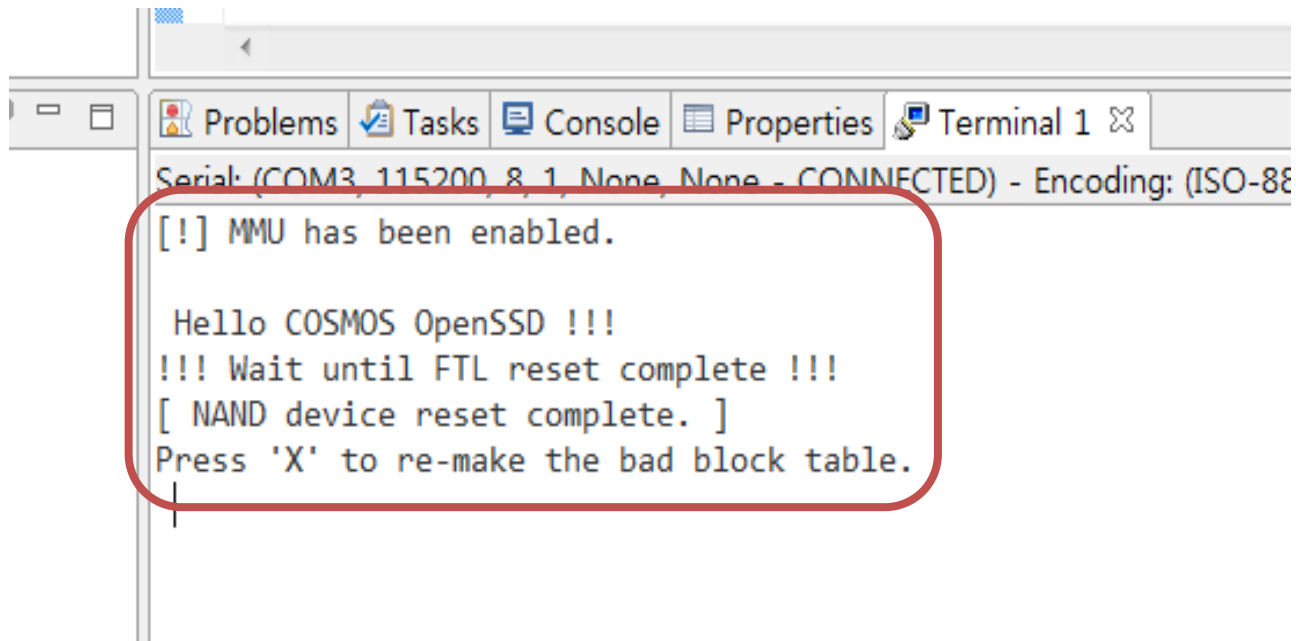
Execute Firmware (2 / 3)

- Click the firmware to execute -> click “OK” -> wait UART message



Execute Firmware (3 / 3)

- Press 'n' to maintain the bad block table



The screenshot shows a terminal window with a toolbar containing 'Problems', 'Tasks', 'Console', 'Properties', and 'Terminal 1'. The terminal title bar indicates the connection is 'Serial: (COM3, 115200, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)'. The output text is as follows:

```
[!] MMU has been enabled.  
  
Hello COSMOS OpenSSD !!!  
!!! Wait until FTL reset complete !!!  
[ NAND device reset complete. ]  
Press 'X' to re-make 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
 - Bad block table format of greedy FTL v2.7.0 is different from the previous versions
 - Damaged bad block table can be recovered

```
[!] MMU has been enabled.  
Hello COSMOS OpenSSD !!!  
!!! Wait until FTL reset complete !!!  
[ NAND device reset complete. ]  
Press 'X' to re-make the bad block table.
```

```
DS_SUB_REEXE Request 18 Fail - ch 0 way 7 rowAddr 1 / status A5000001  
[ bad block table of ch 0 way 7 does not exist.]
```

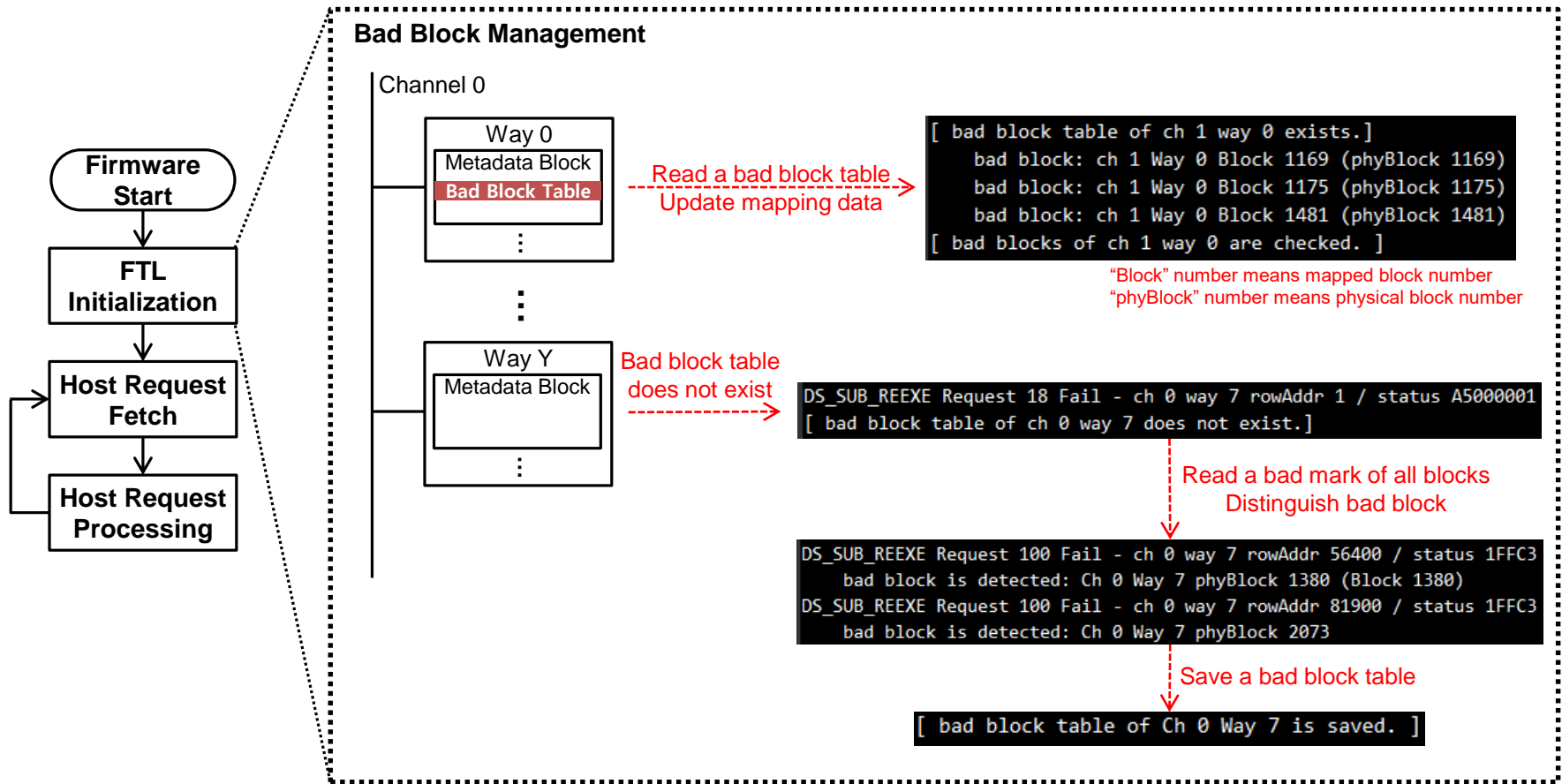
“X” erases all blocks including a metadata block

Others maintain the bad block table

```
[ bad block table of ch 1 way 0 exists.]  
bad block: ch 1 Way 0 Block 1169 (phyBlock 1169)  
bad block: ch 1 Way 0 Block 1175 (phyBlock 1175)  
bad block: ch 1 Way 0 Block 1481 (phyBlock 1481)  
[ bad blocks of ch 1 way 0 are checked. ]
```

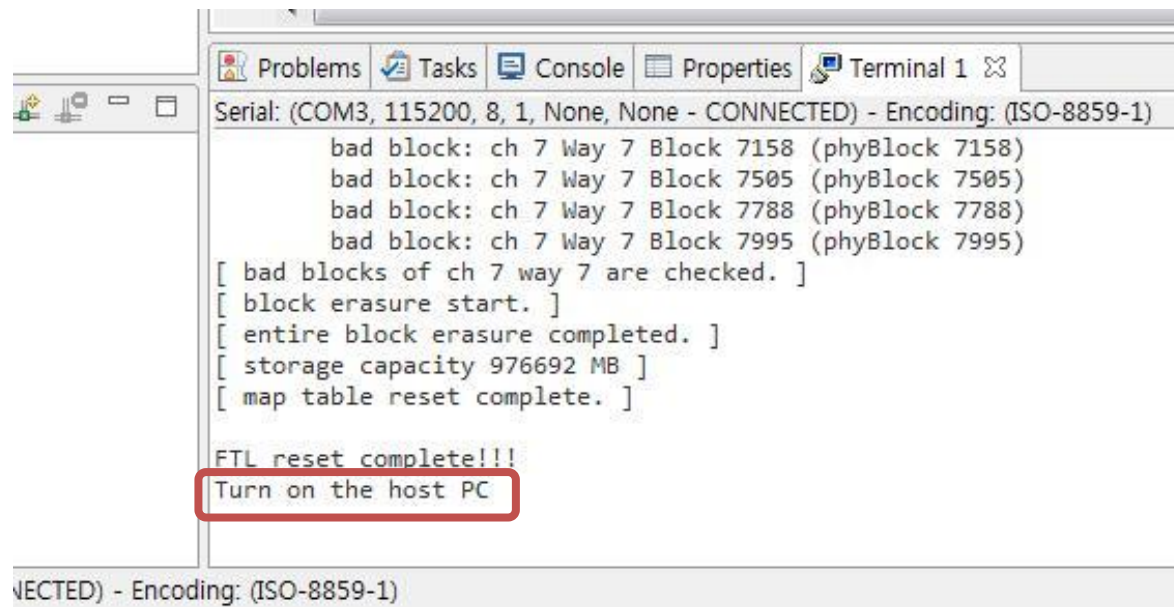
Bad Block Management (2 / 2)

Bad blocks are detected in FTL initialization step



Turn on the Host PC

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



```
Serial: (COM3, 115200, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)
    bad block: ch 7 Way 7 Block 7158 (phyBlock 7158)
    bad block: ch 7 Way 7 Block 7505 (phyBlock 7505)
    bad block: ch 7 Way 7 Block 7788 (phyBlock 7788)
    bad block: ch 7 Way 7 Block 7995 (phyBlock 7995)
[ bad blocks of ch 7 way 7 are checked. ]
[ block erasure start. ]
[ entire block erasure completed. ]
[ storage capacity 976692 MB ]
[ map table reset complete. ]

FTL reset complete!!!
Turn on the host PC
```

CONNECTED) - Encoding: (ISO-8859-1)

UART Messages While Host Computer is Booting up

- NVMe SSD initialization steps are on going

```
Serial: (COM3, 115200, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)
[ storage capacity 977588 MB ]
[ map table reset complete. ]

FTL reset complete!!!
Turn on the host PC
PCIe Link: 1
PCIe Bus Master: 1
PCIe Bus Master: 0
PCIe Bus Master: 1
NVME CC.EN: 1

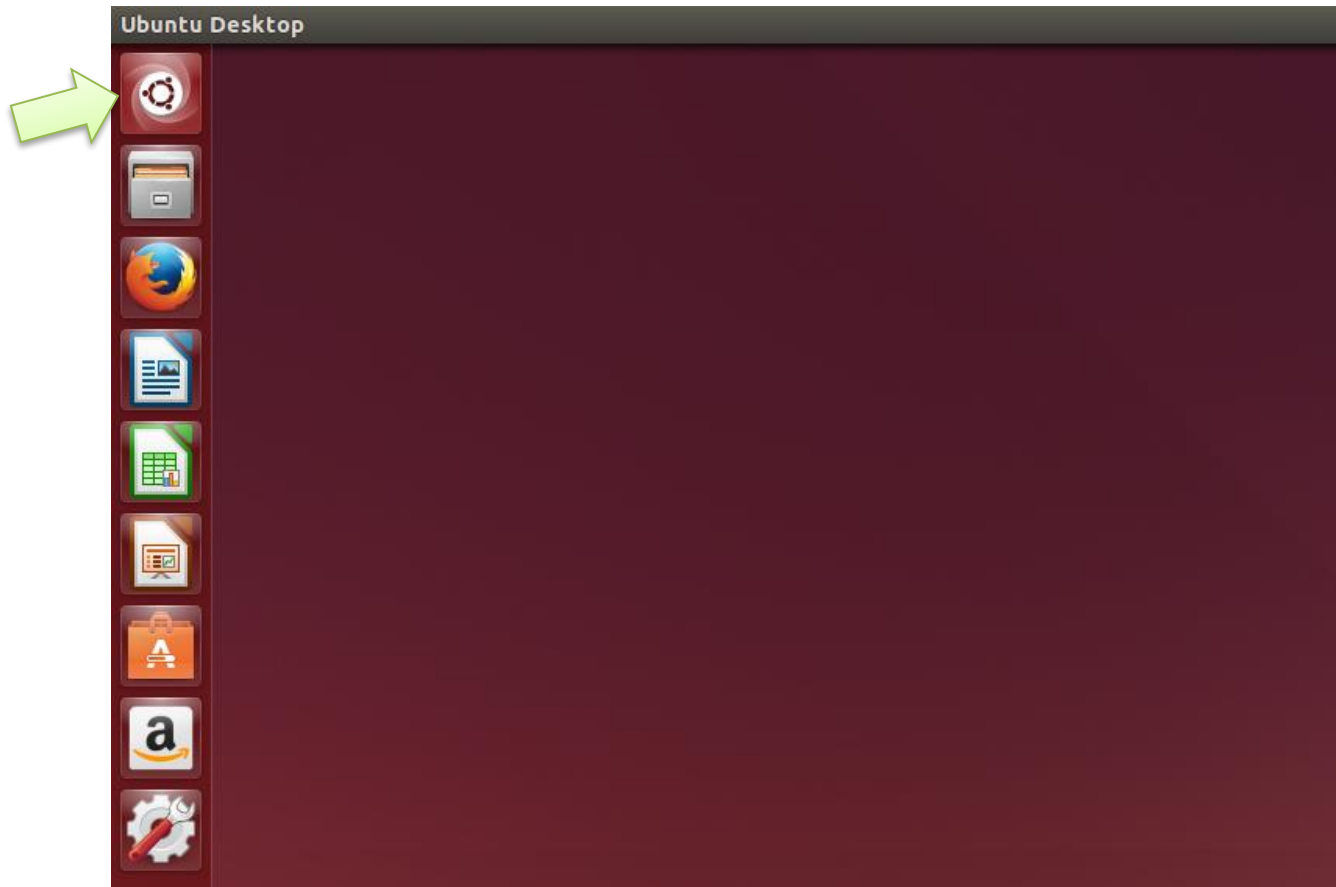
NVMe ready!!!
Done Admin Command OPC: 6
num_of_queue 30003
Set Feature FID:7
PCIe IRQ Disable: 0
PCIe MSI Enable: 1, 0x0
Done Admin Command OPC: 9
create cq: 0x00000003, 0x00FF0001
Done Admin Command OPC: 5
create sq: 0x00010005, 0x00FF0001
Done Admin Command OPC: 1
Done Admin Command OPC: 6
Done Admin Command OPC: C
Done Admin Command OPC: 6
Done Admin Command OPC: 6
Done Admin Command OPC: 6
```

Operating Cosmos+ 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**

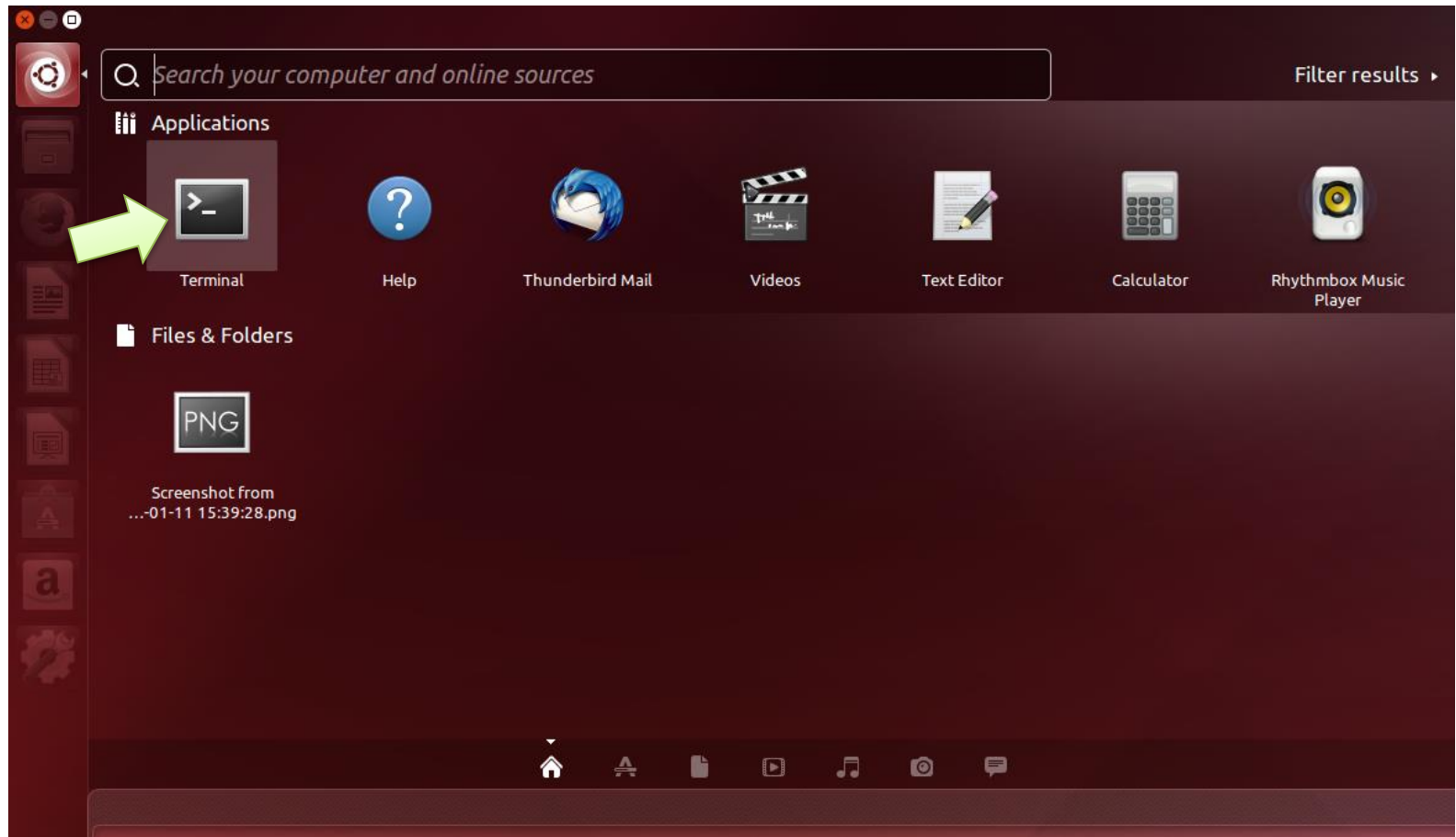
Open a Terminal (1 / 2)

- Click the pointed icon



Open a Terminal (2 / 2)

- Click the terminal icon



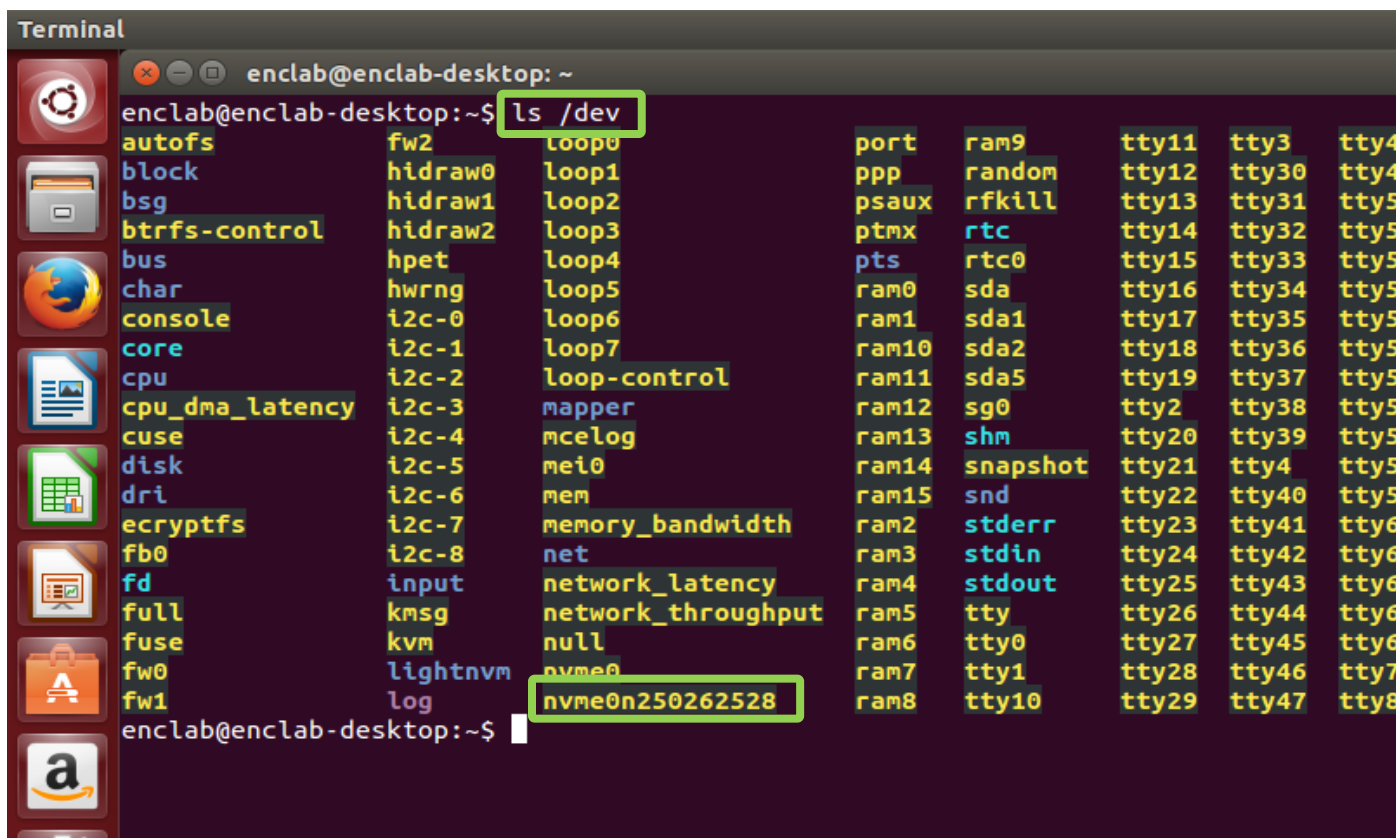
Check Device Recognition (1 / 2)

- Types “lspci” -> press ENTER -> check “Non-Volatile memory controller: Xilinx Corporation Device 7028” on the PCI device list

```
Terminal
enclab@enclab-desktop: ~
enclab@enclab-desktop:~$ lspci
00:00.0 Host bridge: Intel Corporation Xeon E3-1200 v2/3rd Gen Core processor DRAM Controller (rev 09)
00:01.0 PCI bridge: Intel Corporation Xeon E3-1200 v2/3rd Gen Core processor PCI Express Root Port (rev 09)
00:02.0 VGA compatible controller: Intel Corporation Xeon E3-1200 v2/3rd Gen Core processor Graphics Controller (rev 09)
00:14.0 USB controller: Intel Corporation 7 Series/C210 Series Chipset Family USB xHCI Host Controller (rev 04)
00:16.0 Communication controller: Intel Corporation 7 Series/C210 Series Chipset Family MEI Controller #1 (rev 04)
00:1a.0 USB controller: Intel Corporation 7 Series/C210 Series Chipset Family USB Enhanced Host Controller #2 (rev 04)
00:1b.0 Audio device: Intel Corporation 7 Series/C210 Series Chipset Family High Definition Audio Controller (rev 04)
00:1c.0 PCI bridge: Intel Corporation 7 Series/C210 Series Chipset Family PCI Express Root Port 1 (rev c4)
00:1c.4 PCI bridge: Intel Corporation 7 Series/C210 Series Chipset Family PCI Express Root Port 5 (rev c4)
00:1c.5 PCI bridge: Intel Corporation 7 Series/C210 Series Chipset Family PCI Express Root Port 6 (rev c4)
00:1c.6 PCI bridge: Intel Corporation 7 Series/C210 Series Chipset Family PCI Express Root Port 7 (rev c4)
00:1c.7 PCI bridge: Intel Corporation 7 Series/C210 Series Chipset Family PCI Express Root Port 8 (rev c4)
00:1d.0 USB controller: Intel Corporation 7 Series/C210 Series Chipset Family USB Enhanced Host Controller #1 (rev 04)
00:1f.0 ISA bridge: Intel Corporation Z77 Express Chipset LPC Controller (rev 04)
00:1f.2 SATA controller: Intel Corporation 7 Series/C210 Series Chipset Family 6-port SATA Controller [AHCI mode] (rev 04)
00:1f.3 SMBus: Intel Corporation 7 Series/C210 Series Chipset Family SMBus Controller (rev 04)
01:00.0 PCI bridge: PLX Technology, Inc. Device 8749 (rev ca)
01:00.1 System peripheral: PLX Technology, Inc. Device 87d0 (rev ca)
01:00.2 System peripheral: PLX Technology, Inc. Device 87d0 (rev ca)
01:00.3 System peripheral: PLX Technology, Inc. Device 87d0 (rev ca)
01:00.4 System peripheral: PLX Technology, Inc. Device 87d0 (rev ca)
02:08.0 PCI bridge: PLX Technology, Inc. Device 8749 (rev ca)
02:09.0 PCI bridge: PLX Technology, Inc. Device 8749 (rev ca)
04:00.0 Non-Volatile memory controller: Xilinx Corporation Device 7028
05:00.0 FireWire (IEEE 1394): VIA Technologies, Inc. VT6306/7/8 [Fire II(M)] IEEE 1394 OHCI Controller (rev 01)
06:00.0 SATA controller: ASMedia Technology Inc. ASM1062 Serial ATA Controller (rev 01)
07:00.0 Ethernet controller: Broadcom Corporation NetLink BCM57781 Gigabit Ethernet PCIe (rev 10)
08:00.0 USB controller: Etron Technology, Inc. EJ168 USB 3.0 Host Controller (rev 01)
09:00.0 PCI bridge: PLX Technology, Inc. PEX 8605 PCI Express 4-port Gen2 Switch (rev aa)
0a:01.0 PCI bridge: PLX Technology, Inc. PEX 8605 PCI Express 4-port Gen2 Switch (rev aa)
0a:02.0 PCI bridge: PLX Technology, Inc. PEX 8605 PCI Express 4-port Gen2 Switch (rev aa)
0a:03.0 PCI bridge: PLX Technology, Inc. PEX 8605 PCI Express 4-port Gen2 Switch (rev aa)
0d:00.0 PCI bridge: ASMedia Technology Inc. ASM1083/1085 PCIe to PCI Bridge (rev 03)
0e:02.0 FireWire (IEEE 1394): VIA Technologies, Inc. VT6306/7/8 [Fire II(M)] IEEE 1394 OHCI Controller (rev c0)
enclab@enclab-desktop:~$
```

Check Device Recognition (2 / 2)

- Types “ls /dev” -> press ENTER -> check “nvme0nxxxx” on the device list



```
enclab@enclab-desktop: ~  
enclab@enclab-desktop:~$ ls /dev  
autofs          fw2             loop0           port            ram9            tty11           tty3            tty4  
block           hidraw0         loop1           ppp             random          tty12           tty30           tty4  
bsg             hidraw1         loop2           psaux           rfcill          tty13           tty31           tty5  
btrfs-control  hidraw2         loop3           ptmx            rtc             tty14           tty32           tty5  
bus             hpet           loop4           pts             rtc0            tty15           tty33           tty5  
char            hwrng          loop5           ram0            sda             tty16           tty34           tty5  
console         i2c-0           loop6           ram1            sda1            tty17           tty35           tty5  
core            i2c-1           loop7           ram10           sda2            tty18           tty36           tty5  
cpu             i2c-2           loop-control    ram11           sda5            tty19           tty37           tty5  
cpu_dma_latency i2c-3           mapper          ram12           sg0             tty2            tty38           tty5  
cuse            i2c-4           mcelog          ram13           shm             tty20           tty39           tty5  
disk            i2c-5           mei0            ram14           snapshot        tty21           tty4            tty5  
dri             i2c-6           mem             ram15           snd             tty22           tty40           tty5  
ecryptfs        i2c-7           memory_bandwidth ram2            stderr          tty23           tty41           tty6  
fb0             i2c-8           net             ram3            stdin           tty24           tty42           tty6  
fd              input           network_latency ram4            stdout          tty25           tty43           tty6  
full            kmsg           network_throughput ram5            tty             tty26           tty44           tty6  
fuse            kvm            null            ram6            tty0            tty27           tty45           tty6  
fw0             lightnvme       nvme0           ram7            tty1            tty28           tty46           tty7  
fw1             log             nvme0n250262528 ram8            tty10           tty29           tty47           tty8  
enclab@enclab-desktop:~$
```

Create a Partition

- Type “sudo fdisk /dev/nvme0nxxxx”, press ENTER -> type your password, press ENTER -> type “n”, press ENTER -> type “p”, press ENTER -> type “1”, press ENTER -> type “4096”, press ENTER

```
Terminal
enclab@enclab-desktop: ~
enclab@enclab-desktop:~$ sudo fdisk /dev/nvme0n250262528
[sudo password for enclab:
Note: sector size is 4096 (not 512)
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF
Building a new DOS disklabel with disk identifier 0xfdd4e220.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (256-250262527, default 256): 4096
```

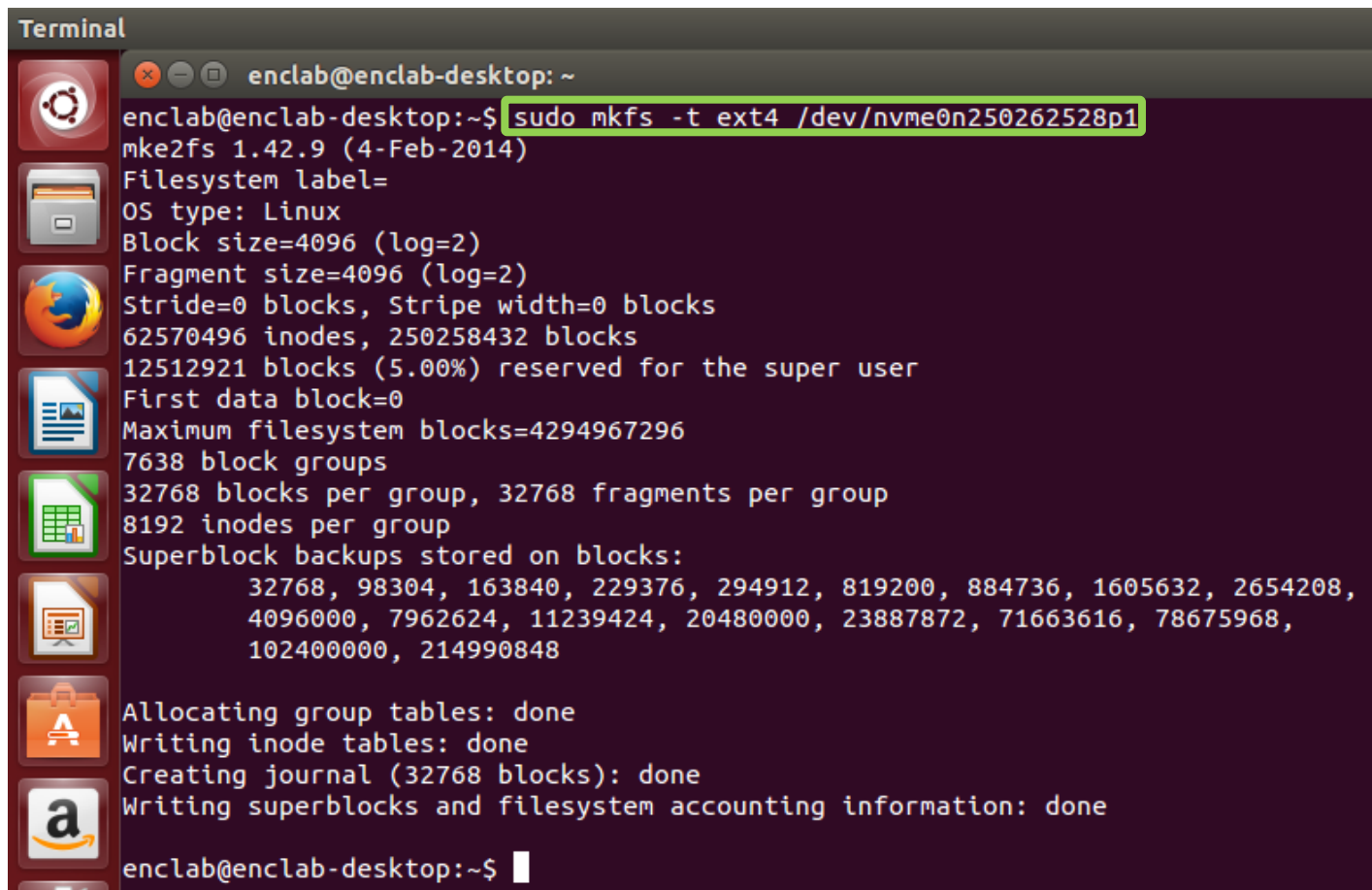

Check the Created Partition

- Types “ls /dev” -> press ENTER -> check “nvme0nxxxxp1” on the device list

```
Terminal
enclab@enclab-desktop: ~
enclab@enclab-desktop:~$ ls /dev
autofs          hidraw1        loop4          ram0           sda2           tty2           tty4           tty6           ttyS20        usb
block           hidraw2        loop5          ram1           sda5           tty20          tty40          tty60          ttyS21        userio
bsg             hpet          loop6          ram10          sg0            tty21          tty41          tty61          ttyS22        vcs
btrfs-control  hwrng         loop7          ram11          shm            tty22          tty42          tty62          ttyS23        vcs1
bus            i2c-0         loop-control   ram12          snapshot       tty23          tty43          tty63          ttyS24        vcs2
char           i2c-1         mapper         ram13          snd            tty24          tty44          tty7           ttyS25        vcs3
console        i2c-2         mcelog         ram14          stderr         tty25          tty45          tty8           ttyS26        vcs4
core           i2c-3         mei0           ram15          stdout         tty26          tty46          tty9           ttyS27        vcs5
cpu            i2c-4         mem            ram2           ttdout         tty27          tty47          ttyprintk      ttyS28        vcs6
cpu_dma_latency i2c-5         memory_bandwidth ram3           tty            tty28          tty48          ttyS0          ttyS29        vcsa
cuse          i2c-6         net            ram4           tty0           tty29          tty49          ttyS1          ttyS3         vcsa1
disk         i2c-7         network_latency ram5           tty1           tty3           tty5           ttyS10         ttyS30        vcsa2
dri          i2c-8         network_throughput ram6           tty10          tty30          tty50          ttyS11         ttyS31        vcsa3
ecryptfs     input         null           ram7           tty11          tty31          tty51          ttyS12         ttyS4         vcsa4
fb0          kmsg         nvme0          ram8           tty12          tty32          tty52          ttyS13         ttyS5         vcsa5
fd           kvm          nvme0n250262528 ram9           tty13          tty33          tty53          ttyS14         ttyS6         vcsa6
full        lightningvm  nvme0n250262528p1 random         tty14          tty34          tty54          ttyS15         ttyS7         vfio
fuse         log          port           rfc            tty15          tty35          tty55          ttyS16         ttyS8         vga_arbiter
fw0          loop0        ppp            rtc            tty16          tty36          tty56          ttyS17         ttyS9         vhci
fw1          loop1        psaux          rtc0           tty17          tty37          tty57          ttyS18         uhid          vhost-net
fw2          loop2        ptmx           sda            tty18          tty38          tty58          ttyS19         uinput       zero
hidraw0     loop3        pts            sda1           tty19          tty39          tty59          ttyS2          urandom
```


Format the Partition

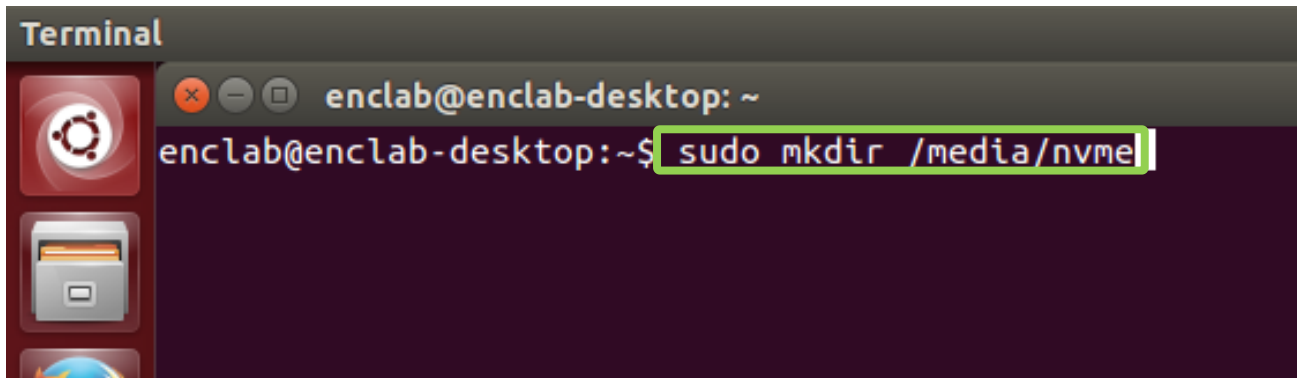
- Type “mkfs -t ext4 / dev/nvme0nxxxxp1”, press ENTER

A terminal window titled "Terminal" showing the execution of the command "sudo mkfs -t ext4 /dev/nvme0n250262528p1". The command is highlighted with a green box. The output shows the progress of formatting the partition, including details like "mke2fs 1.42.9 (4-Feb-2014)", "Filesystem label=", "OS type: Linux", "Block size=4096 (log=2)", "Fragment size=4096 (log=2)", "Stride=0 blocks, Stripe width=0 blocks", "62570496 inodes, 250258432 blocks", "12512921 blocks (5.00%) reserved for the super user", "First data block=0", "Maximum filesystem blocks=4294967296", "7638 block groups", "32768 blocks per group, 32768 fragments per group", "8192 inodes per group", "Superblock backups stored on blocks:", "32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208, 4096000, 7962624, 11239424, 20480000, 23887872, 71663616, 78675968, 102400000, 214990848", "Allocating group tables: done", "Writing inode tables: done", "Creating journal (32768 blocks): done", "Writing superblocks and filesystem accounting information: done", and the prompt "enclab@enclab-desktop:~\$".

```
Terminal
enclab@enclab-desktop: ~
enclab@enclab-desktop:~$ sudo mkfs -t ext4 /dev/nvme0n250262528p1
mke2fs 1.42.9 (4-Feb-2014)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
62570496 inodes, 250258432 blocks
12512921 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
7638 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000, 7962624, 11239424, 20480000, 23887872, 71663616, 78675968,
    102400000, 214990848
Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
enclab@enclab-desktop:~$
```

Create a Mount Point

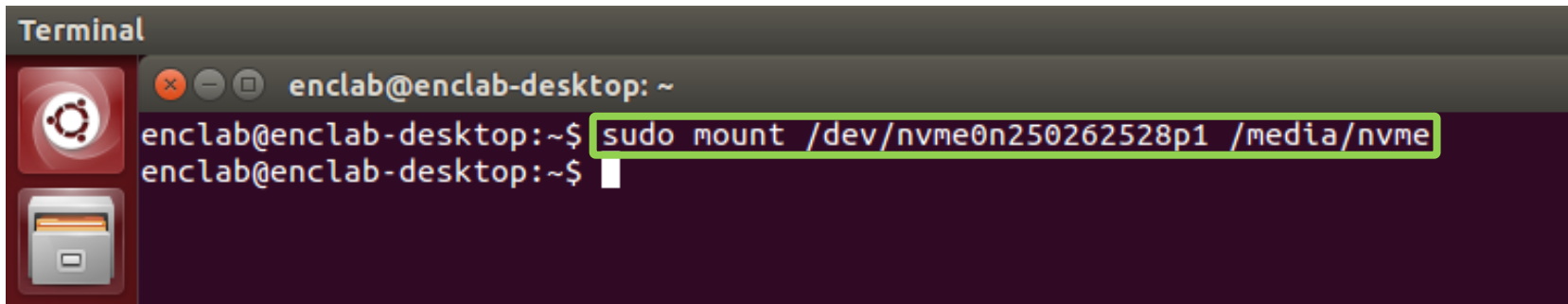
- Type “`sudo mkdir /media/nvme`”, press ENTER



A terminal window titled "Terminal" is shown. The window title bar includes standard Linux window controls (close, minimize, maximize) and the text "enclab@enclab-desktop: ~". The terminal content shows the prompt "enclab@enclab-desktop:~\$" followed by the command "sudo mkdir /media/nvme" which is highlighted with a green box. The terminal background is dark purple. On the left side of the terminal window, there is a vertical sidebar with three icons: the Ubuntu logo, a storage device icon, and a globe icon.

Mount the Partition

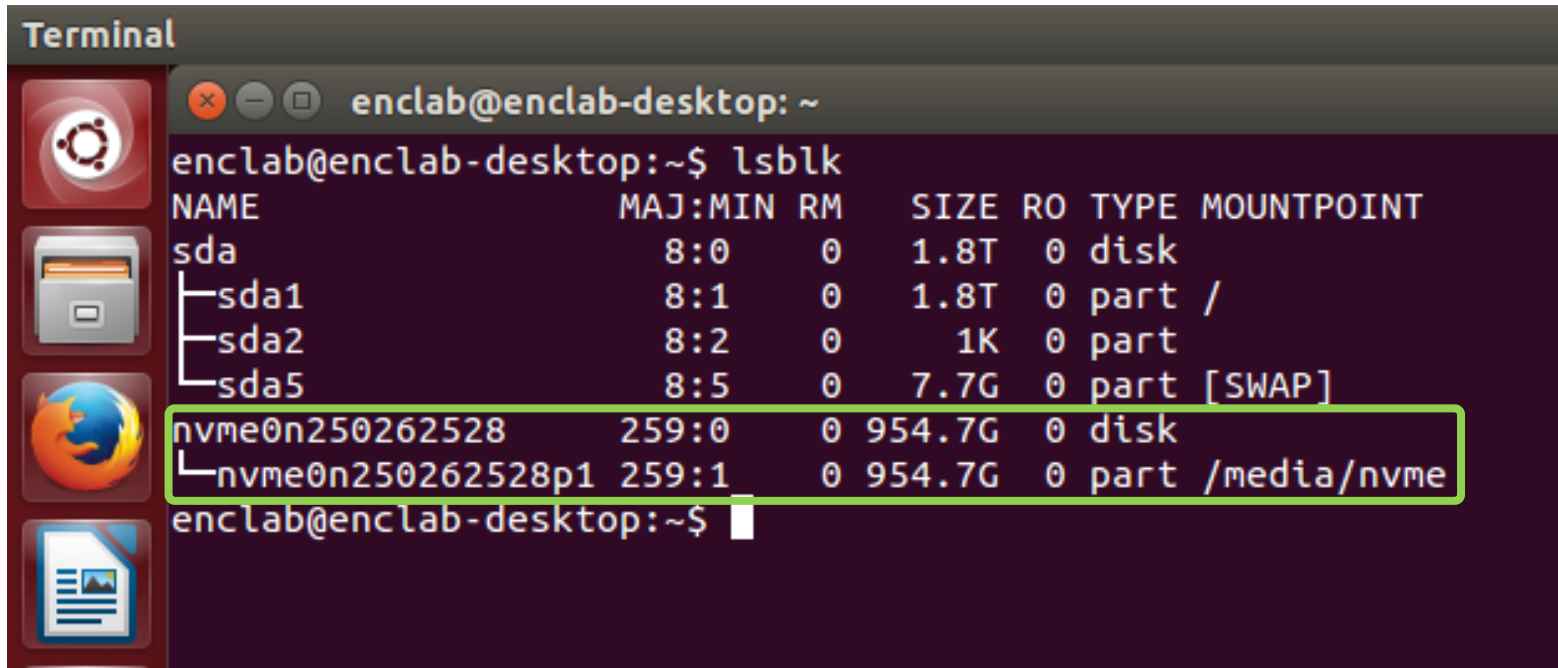
- Type “`sudo mount /dev/nvme0nxxxxp1 /media/nvme`”, press ENTER



```
Terminal
enclab@enclab-desktop: ~
enclab@enclab-desktop:~$ sudo mount /dev/nvme0n250262528p1 /media/nvme
enclab@enclab-desktop:~$
```

Check the Mounted Partition (1 / 2)

- Type "lsblk", press ENTER -> check the mounted partition on the block device list



```
Terminal
enclab@enclab-desktop: ~
enclab@enclab-desktop:~$ lsblk
NAME                                MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                                  8:0    0  1.8T  0 disk
├─sda1                               8:1    0  1.8T  0 part /
├─sda2                               8:2    0    1K  0 part
└─sda5                               8:5    0   7.7G  0 part [SWAP]
nvme0n250262528                    259:0   0 954.7G  0 disk
└─nvme0n250262528p1                259:1   0 954.7G  0 part /media/nvme
enclab@enclab-desktop:~$
```

Check the Mounted Partition (2 / 2)

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

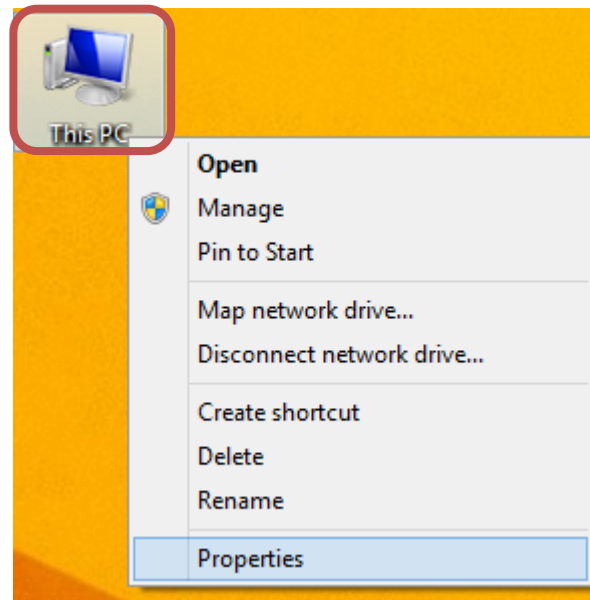
```
Terminal
enclab@enclab-desktop: ~
enclab@enclab-desktop:~$ df -h
Filesystem                Size      Used Avail Use% Mounted on
udev                     3.8G       8.0K  3.8G   1% /dev
tmpfs                    769M       1.3M  767M   1% /run
/dev/sda1                 1.8T       4.4G  1.7T   1% /
none                      4.0K         0   4.0K   0% /sys/fs/cgroup
none                      5.0M         0   5.0M   0% /run/lock
none                      3.8G       176K  3.8G   1% /run/shm
none                      100M        28K  100M   1% /run/user
/dev/nvme0n250262528p1    940G       72M  892G   1% /media/nvme
enclab@enclab-desktop:~$
```

Operating Cosmos+ OpenSSD (Windows)

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

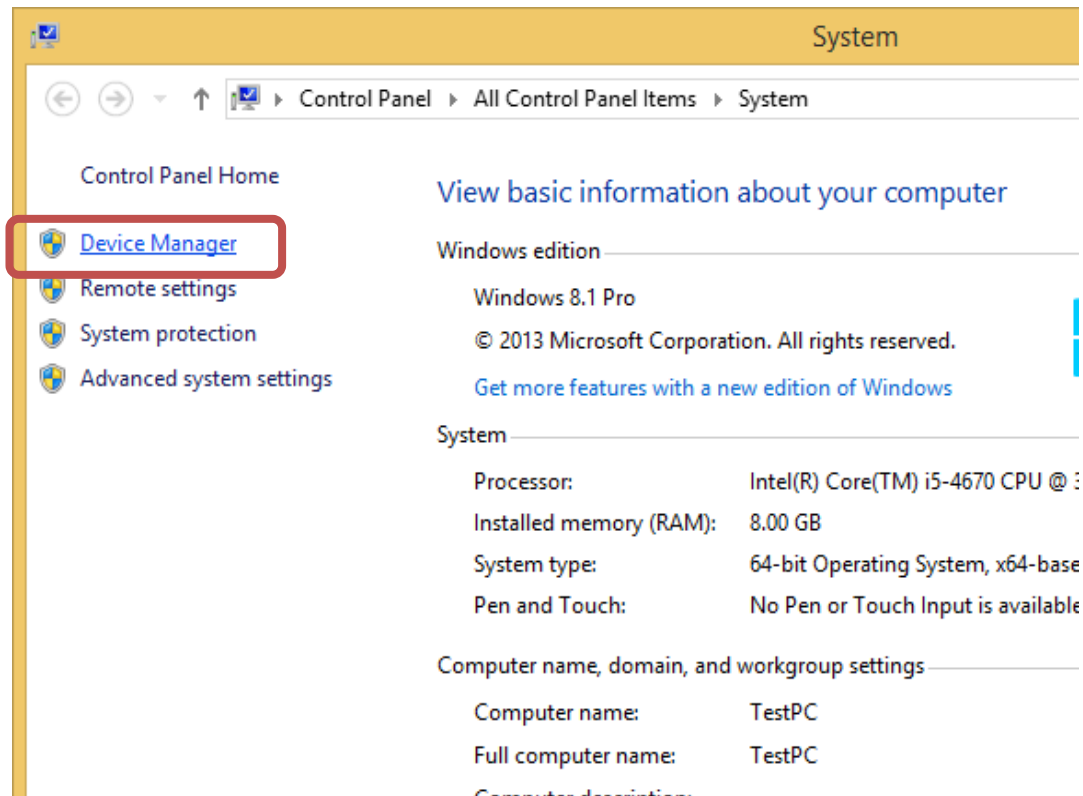
Check Device Recognition (1 / 3)

- **This PC** → click left mouse button → click “Properties”



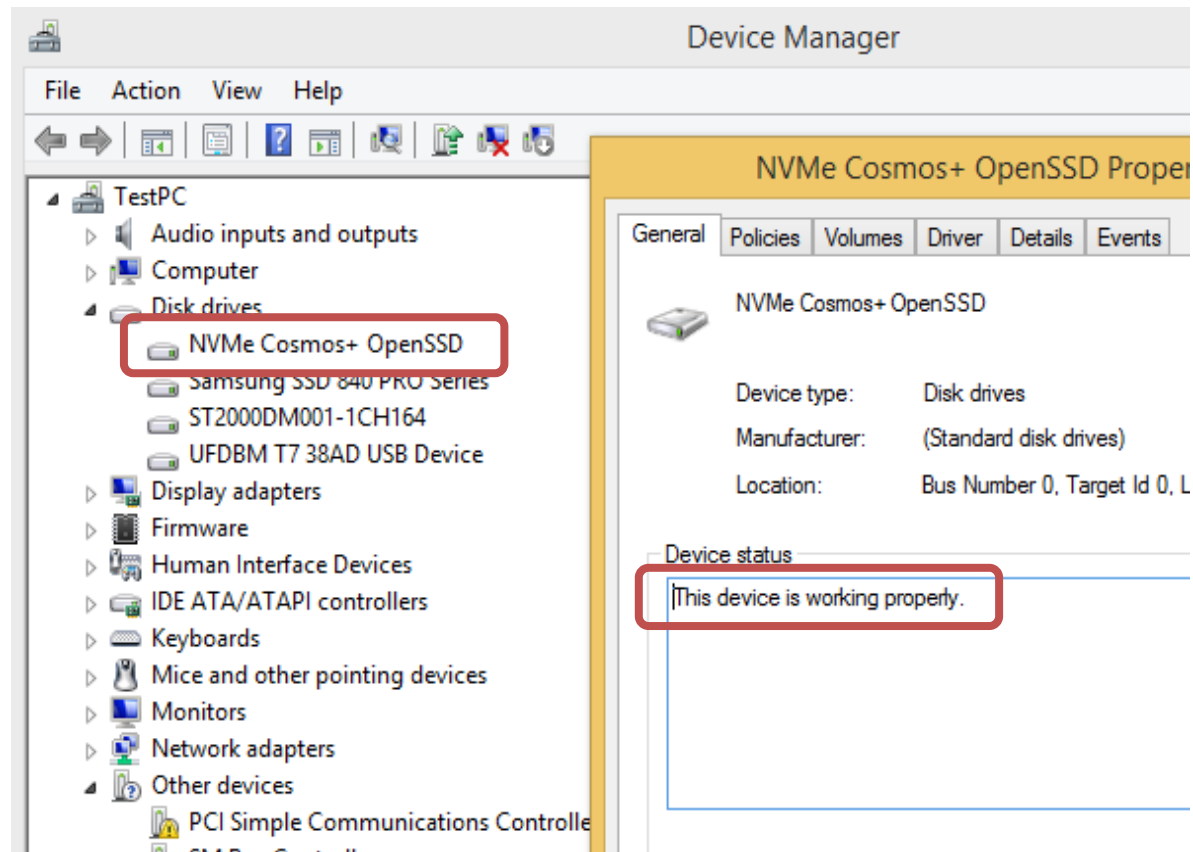
Check Device Recognition (2 / 3)

- System → click “Device Manager”



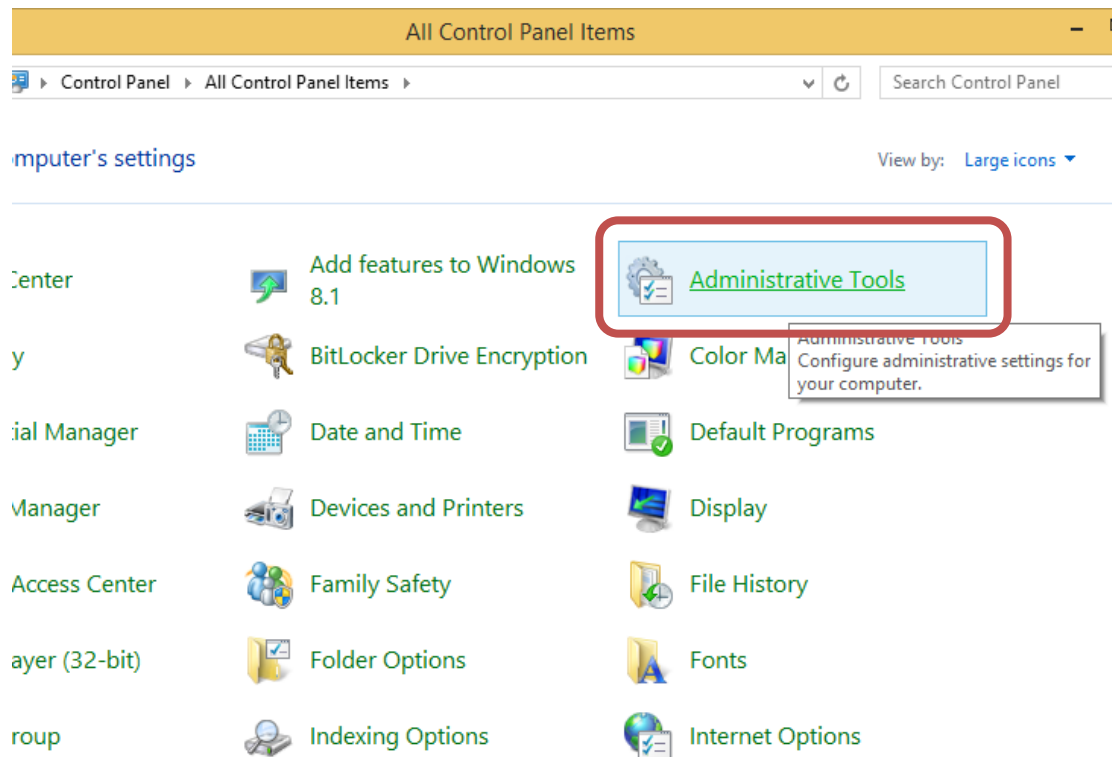
Check Device Recognition (3 / 3)

- Disk drives → double-click “NVMe Cosmos+ OpenSSD”



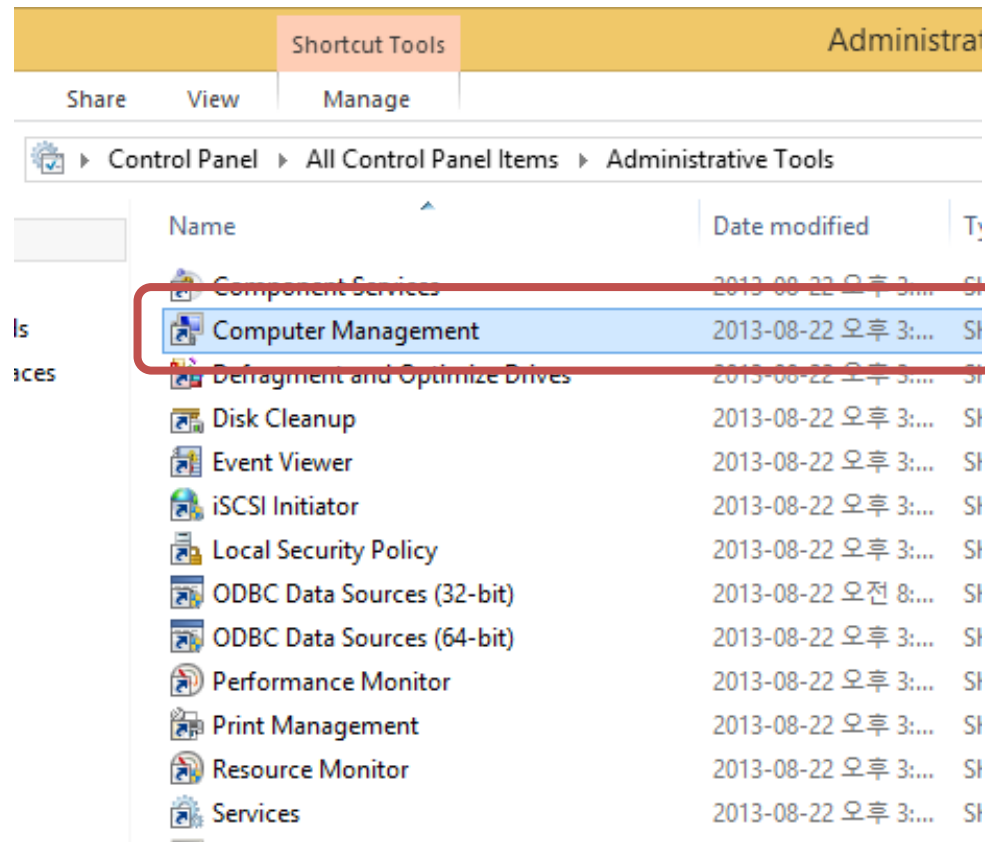
Create a Partition (1 / 5)

- Control panel → click “Administrative Tools”



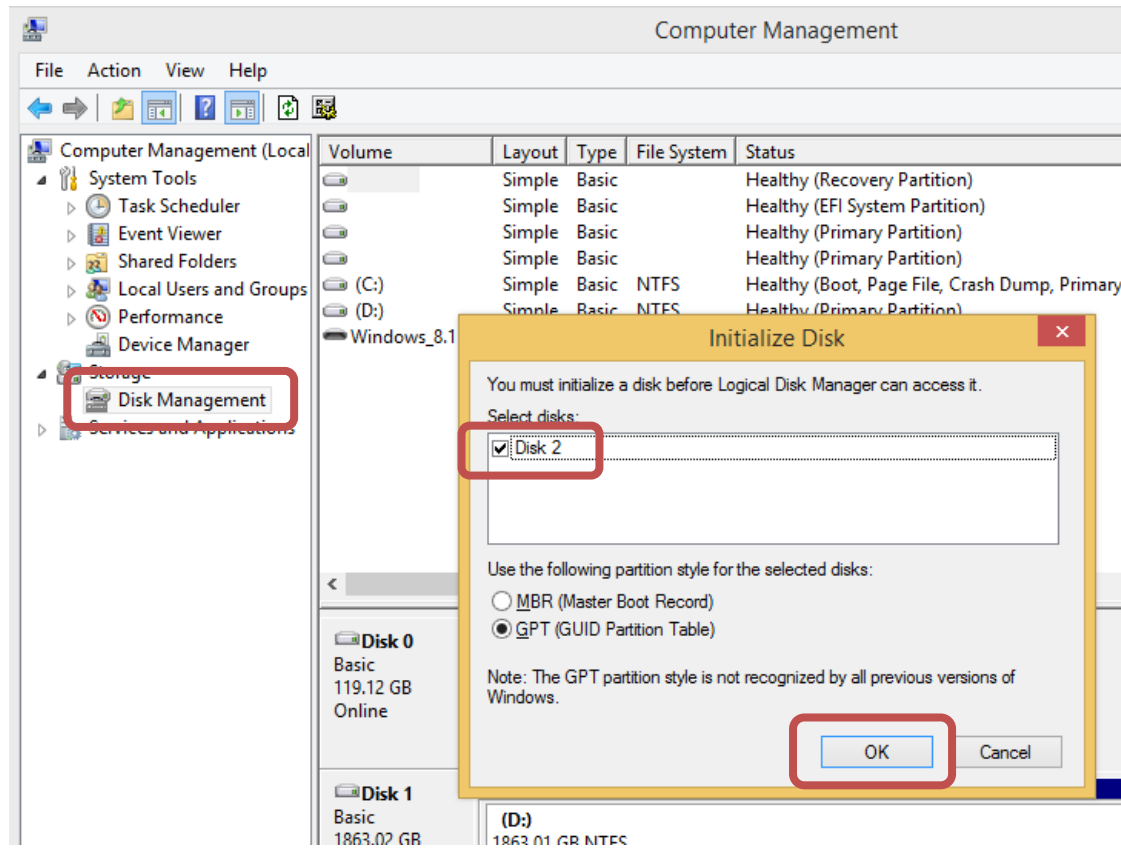
Create a Partition (2 / 5)

- Administrative tools → double-click “Computer Management”



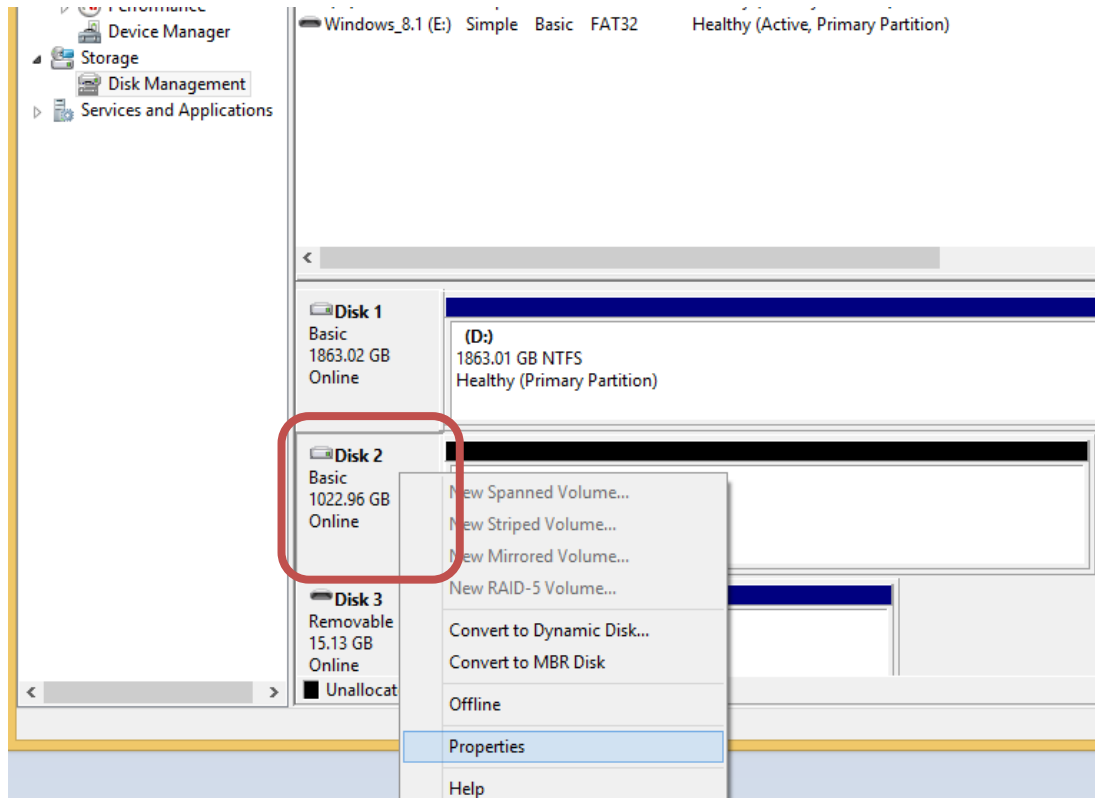
Create a Partition (3 / 5)

- Computer management → click “Disk Management” → click “OK” to confirm disk initialization



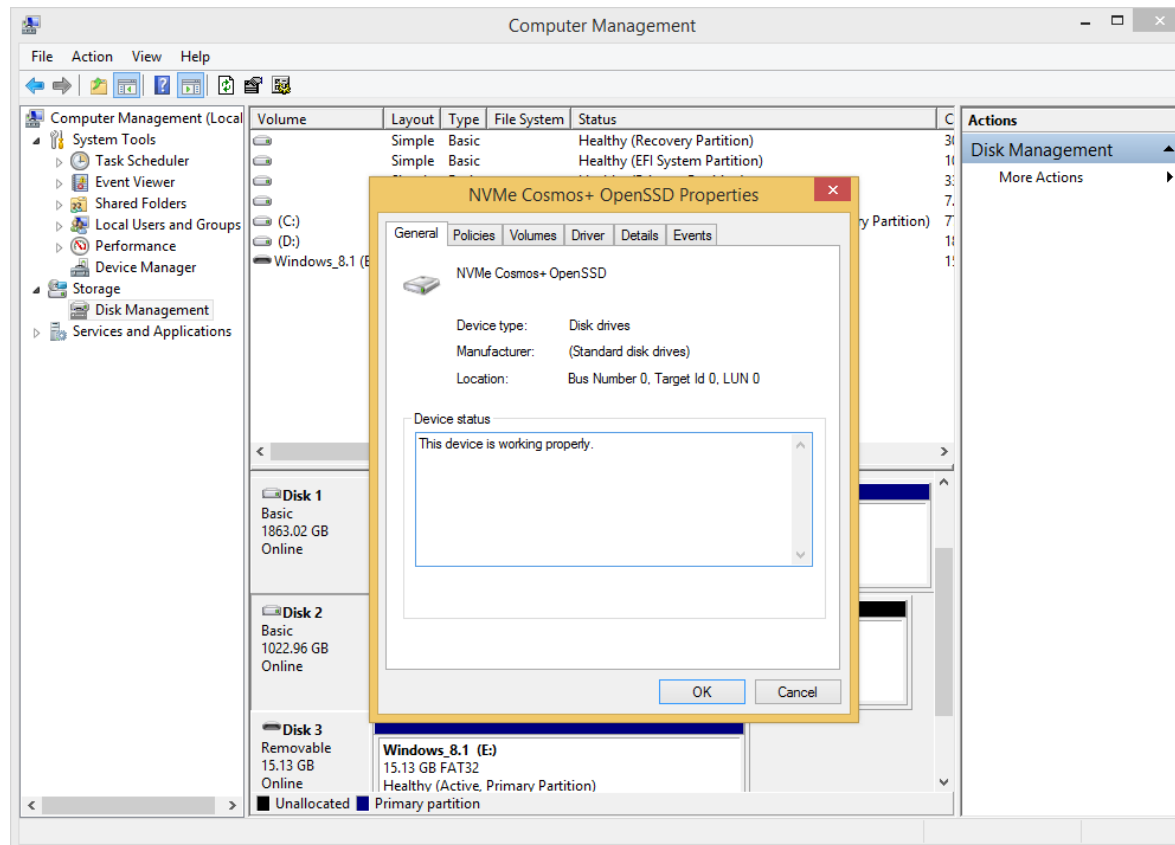
Create a Partition (4 / 5)

- Click right mouse button on “Disk 2” which was shown in 3rd step → click “Properties”



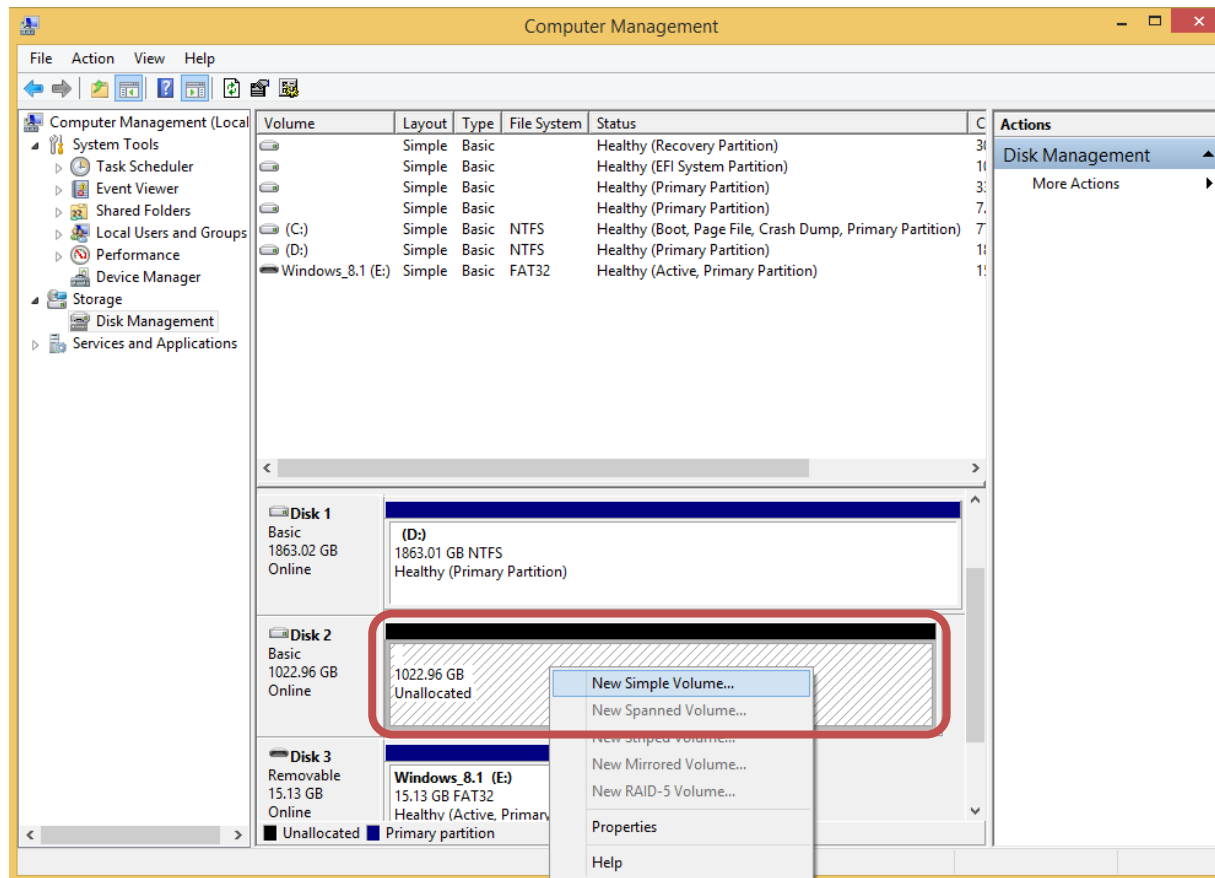
Create a Partition (5 / 5)

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



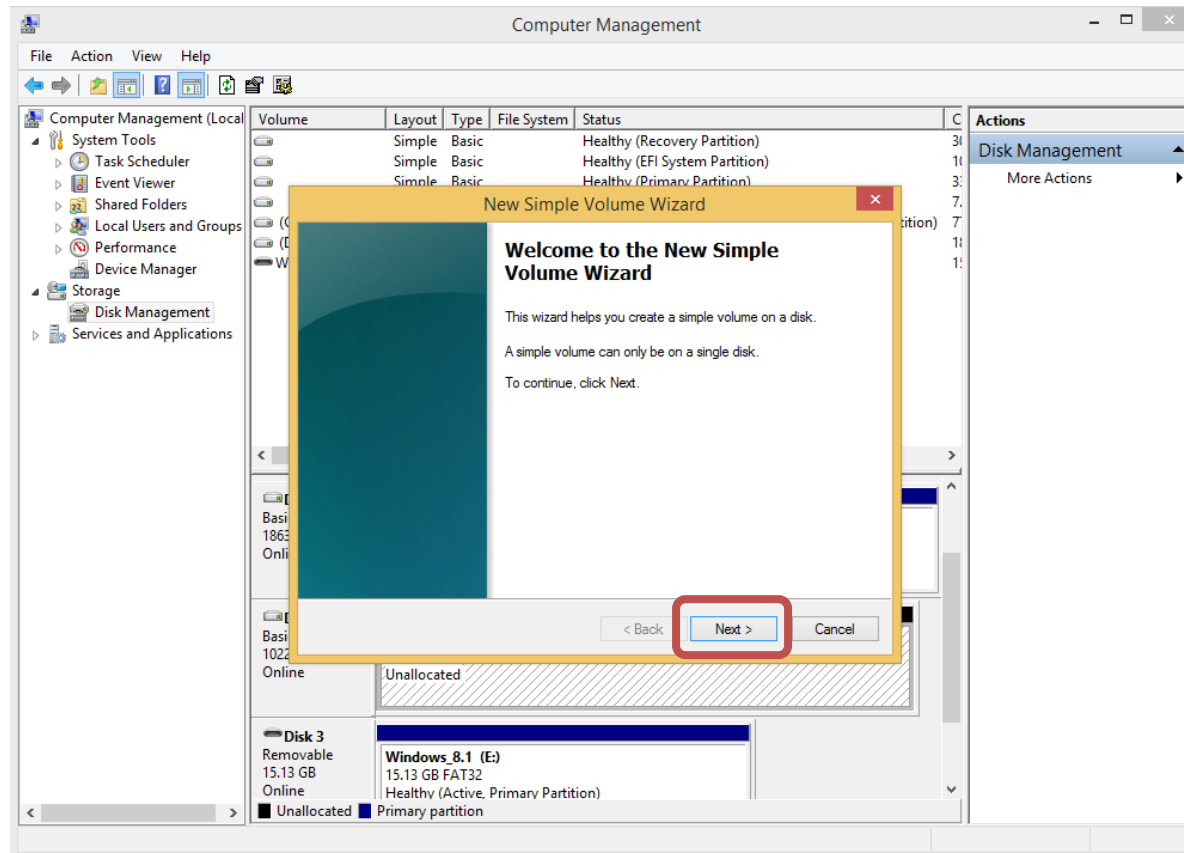
Format the Partition (1 / 8)

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



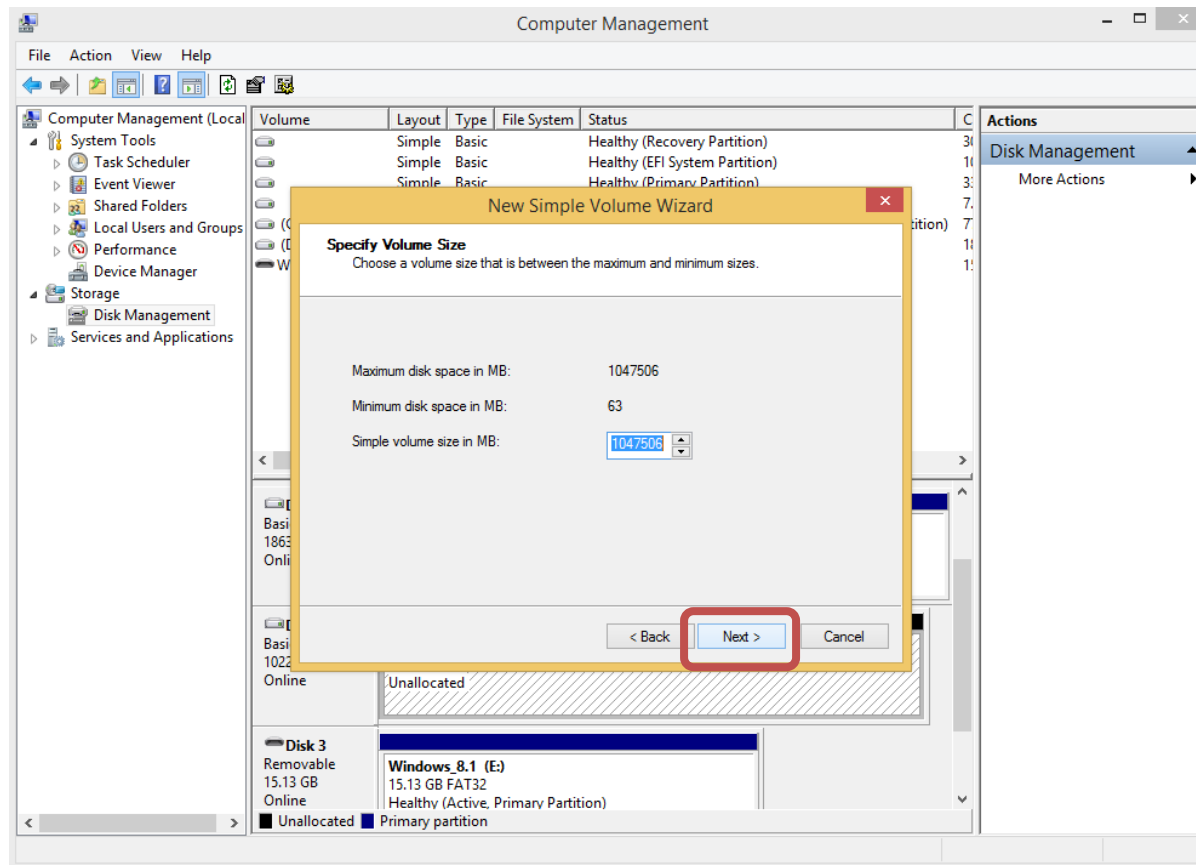
Format the Partition (2 / 8)

- Click “Next”



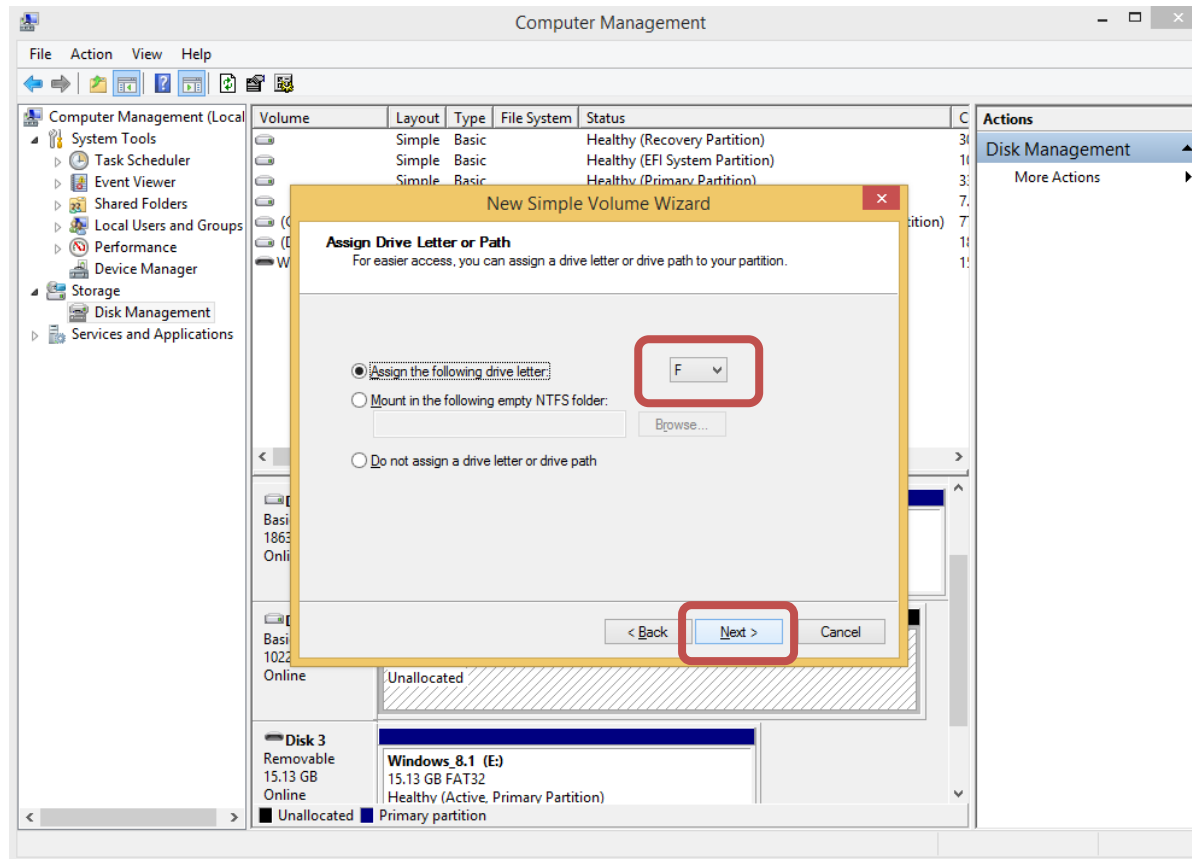
Format the Partition (3 / 8)

- Click “Next”



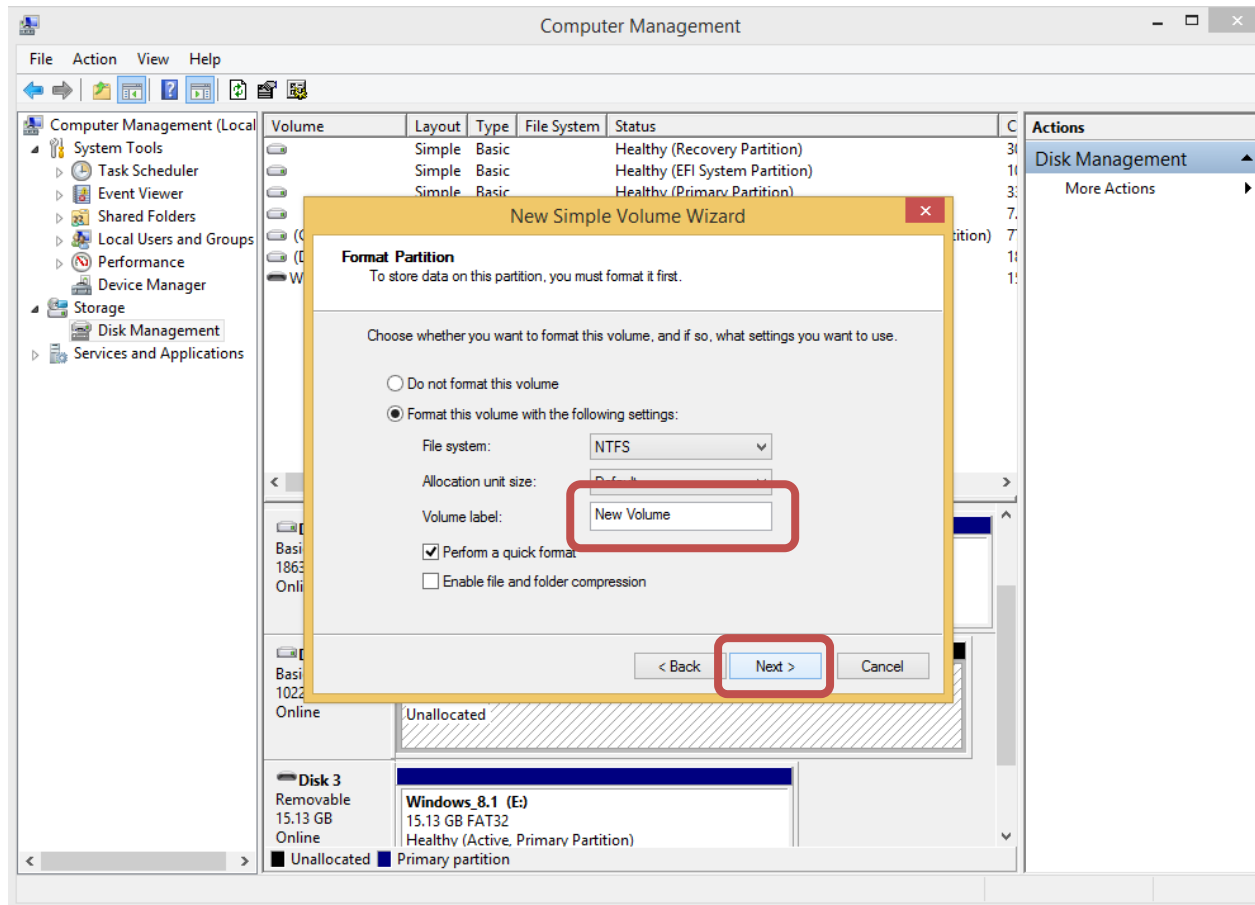
Format the Partition (4 / 8)

- Select desired drive letter → Click “Next”



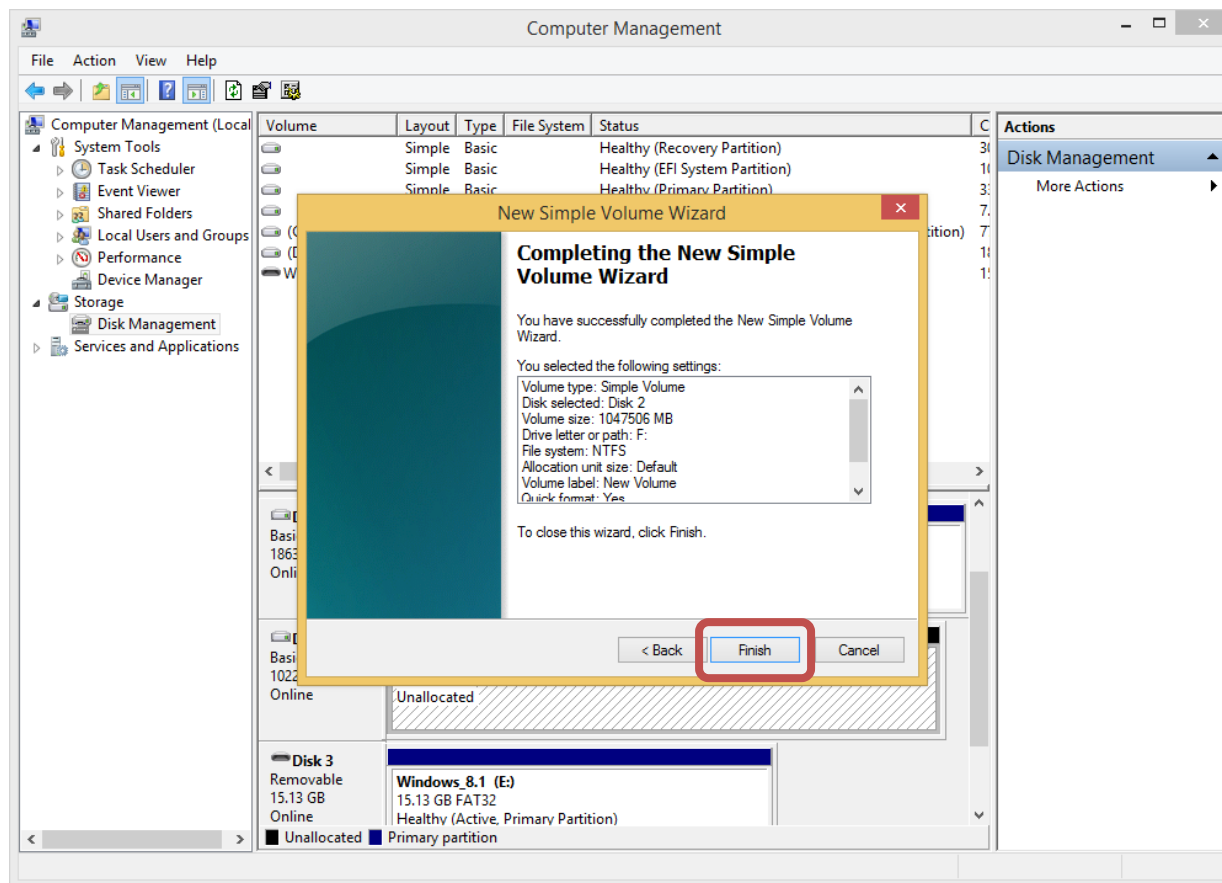
Format the Partition (5 / 8)

- Type desired volume label → Click “Next”



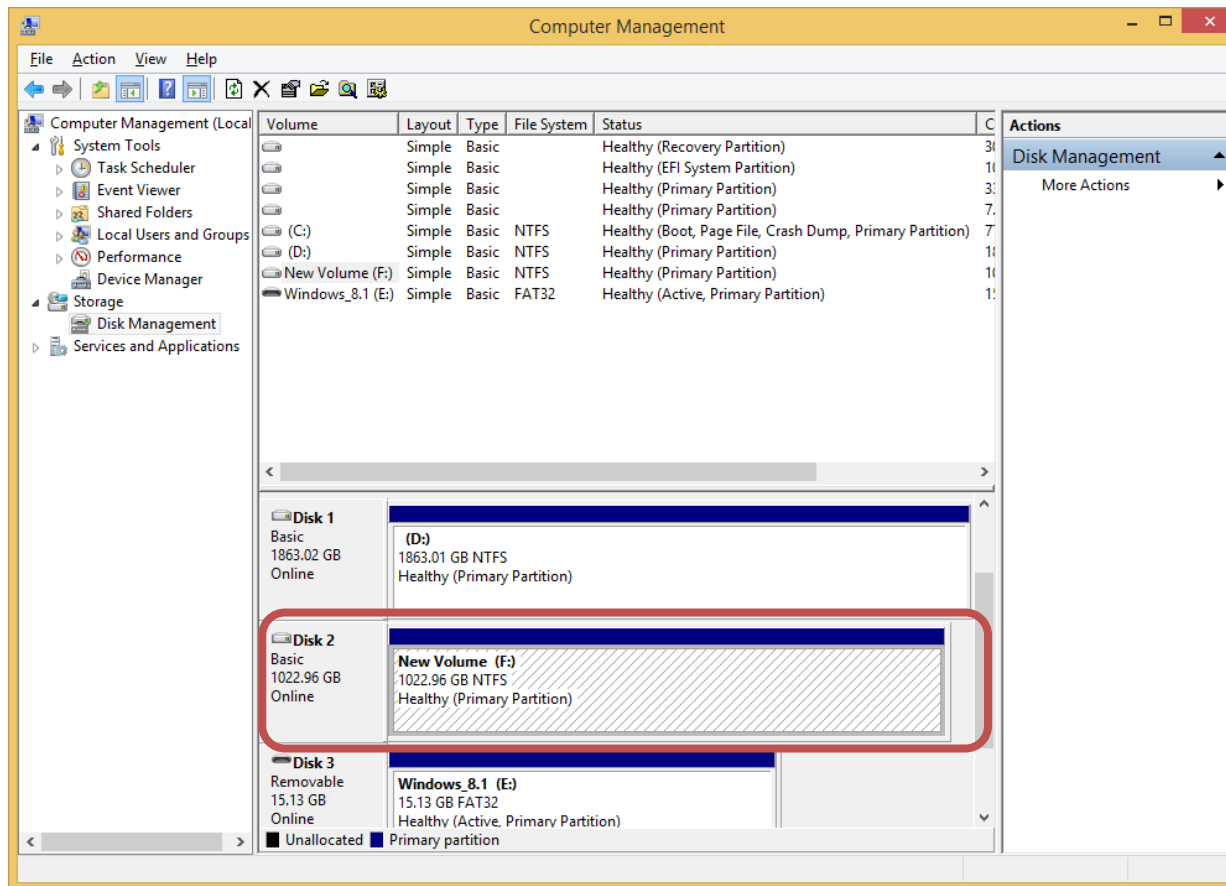
Format the Partition (6 / 8)

- Click “Finish”



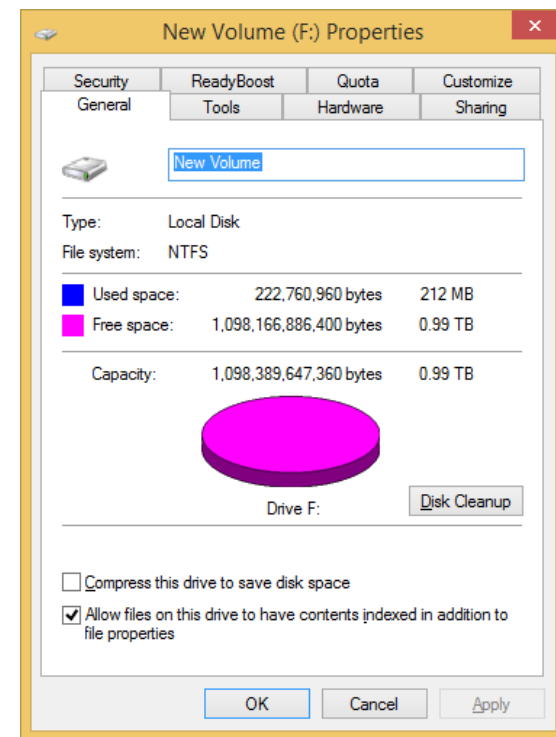
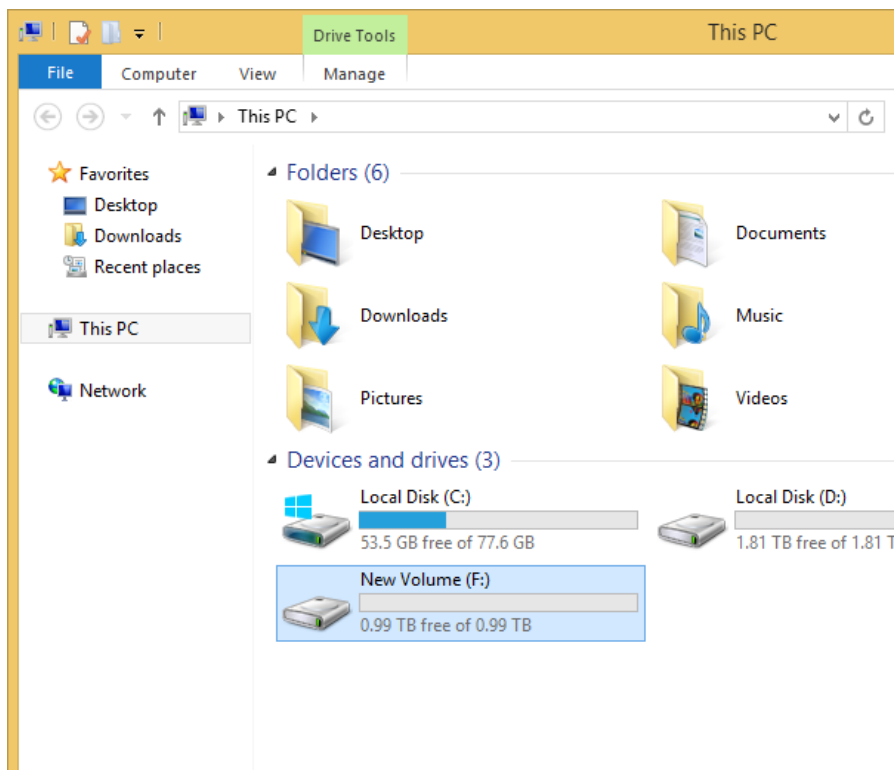
Format the Partition (7 / 8)

■ Formatting is now finished



Format the Partition (8 / 8)

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

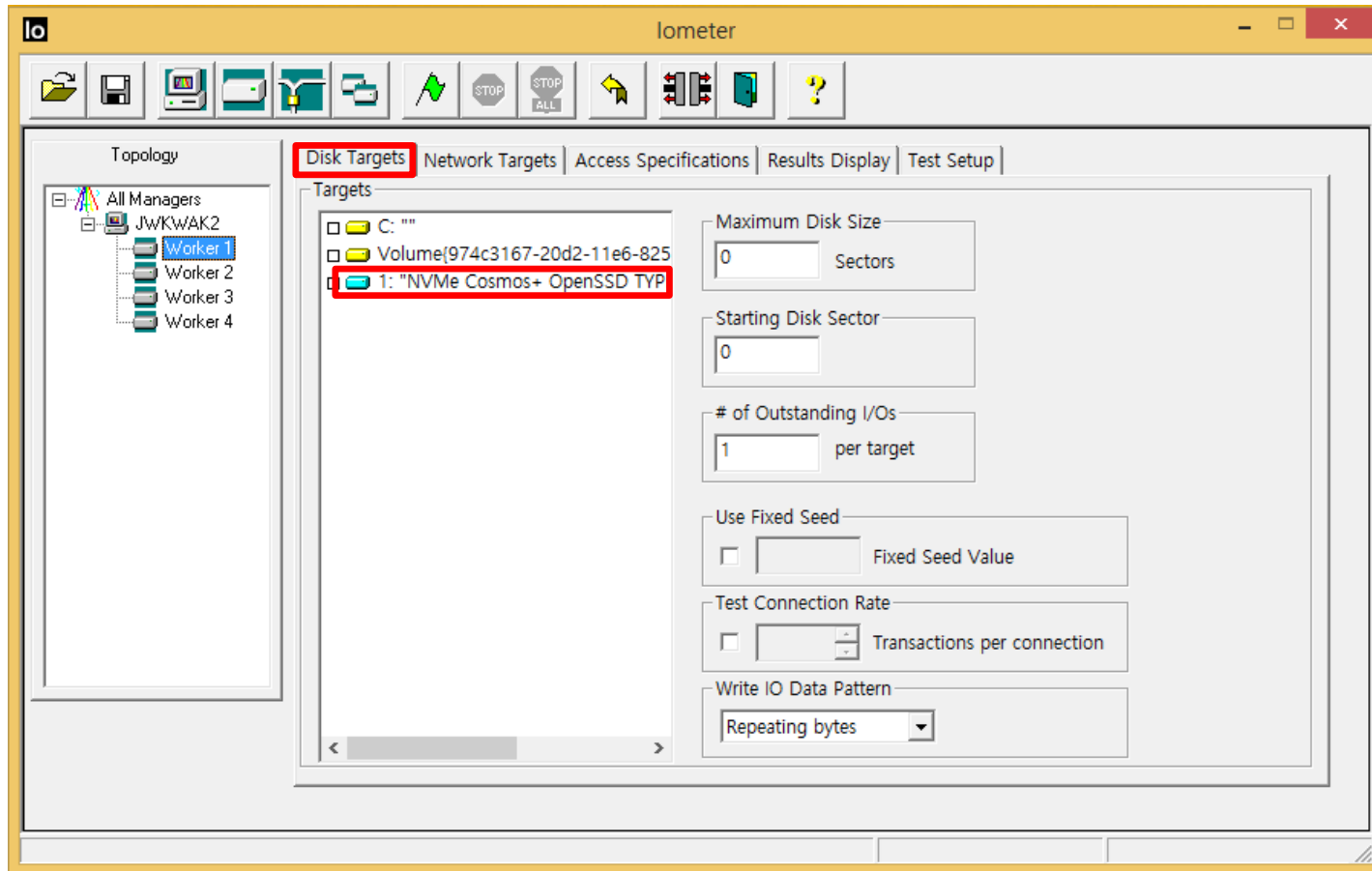


Evaluating Cosmos+ OpenSSD Performance

- 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

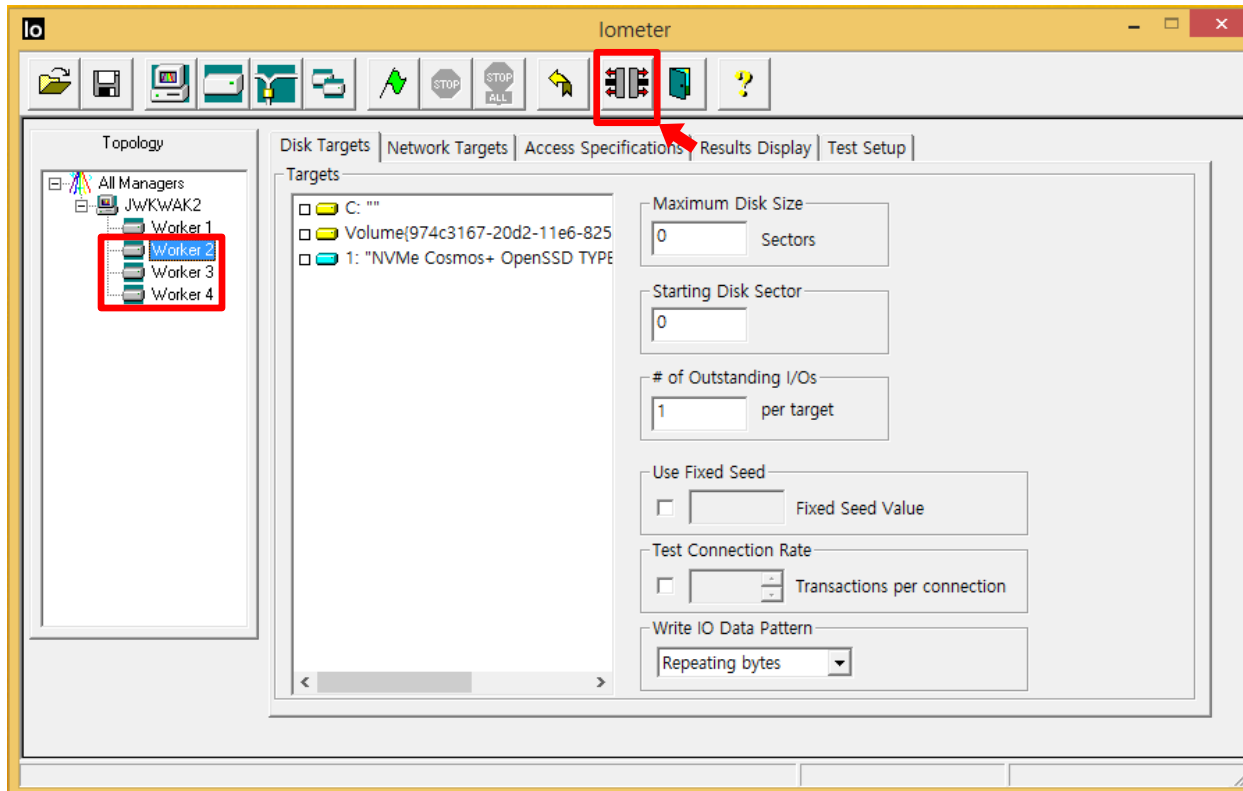
- **iometer 1.1.0** (<http://www.iometer.org/doc/downloads.html>)
 - Cosmos+ OpenSSD is recognized as NVMe storage device



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

The screenshot displays the Iometer application window. The 'Access Specifications' tab is selected, showing a list of 'Global Access Specifications' with a 'New' button highlighted. The 'Edit Access Specification' dialog box is open, showing the configuration for a new specification named '128KB Sequential Write'. The dialog includes a table of parameters and several control panels for fine-tuning the specification.

| Size | % Access | % Read | % Random | Delay | Burst | Alignment | Reply |
|-------------------|----------|--------|----------|-------|-------|-----------------|-------|
| 0 MiB 128 KiB 0 B | 100 | 0 | 0 | 0 | 1 | 0 MiB 4 KiB 0 B | none |

Transfer Request Size: 0 Megabytes, 128 Kilobytes, 0 Bytes

Percent of Access Specification: 100 Percent

Percent Read/Write Distribution: 100% Write, 0% Read

Percent Random/Sequential Distribution: 100% Sequential, 0% Random

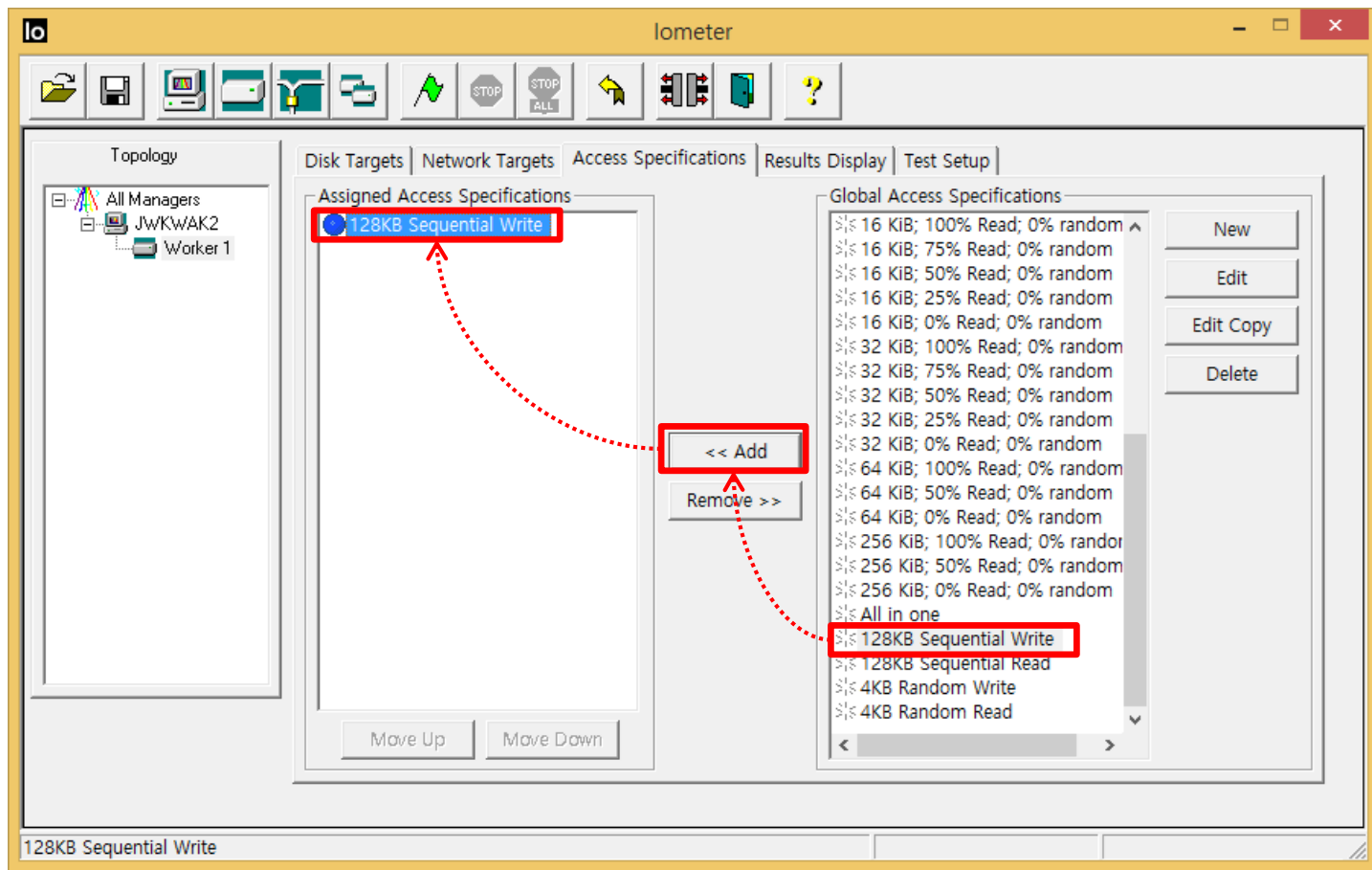
Reply Size: No Reply (selected), 0 Megabytes, 128 Kilobytes, 0 Bytes

Burstiness: Transfer Delay: 0 ms, Burst Length: 1 I/Os

Align I/Os on: Request Size Boundaries (selected), Sector Boundaries, 0 Megabytes, 4 Kilobytes, 0 Bytes

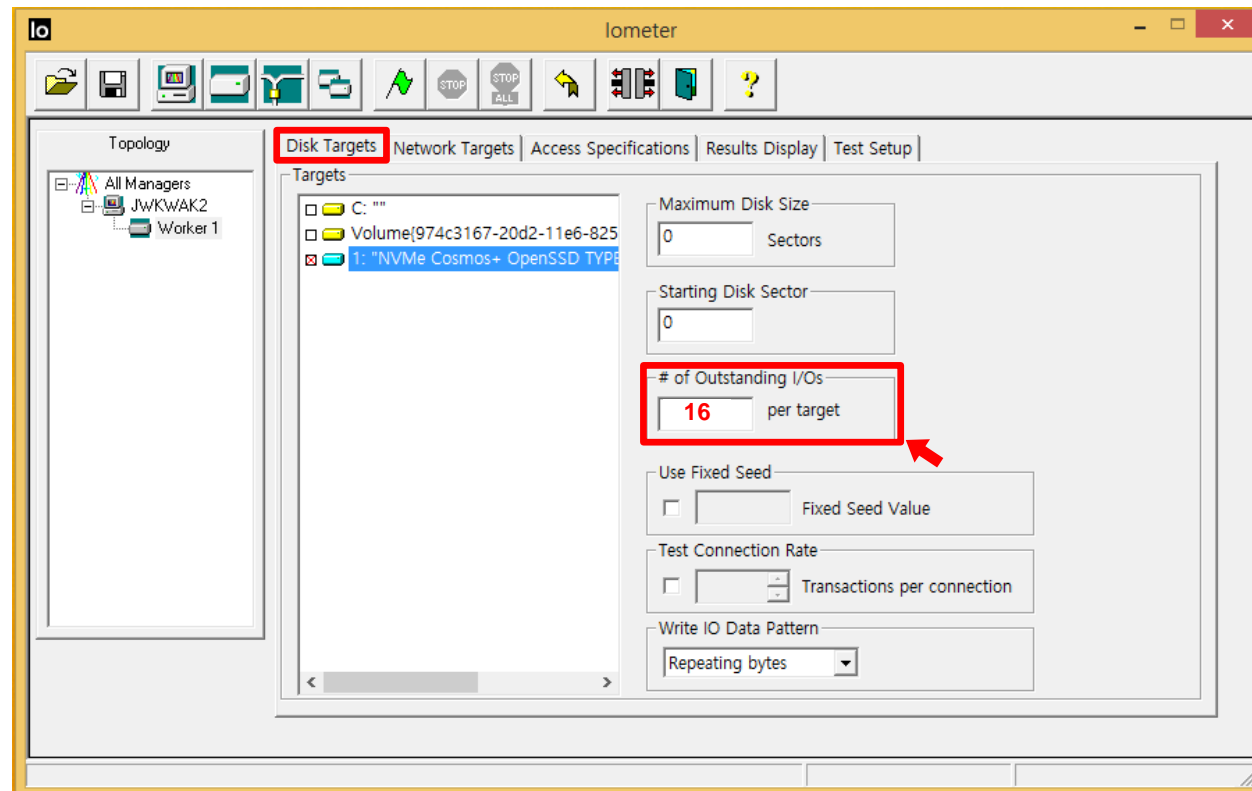
Assign a access specification

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



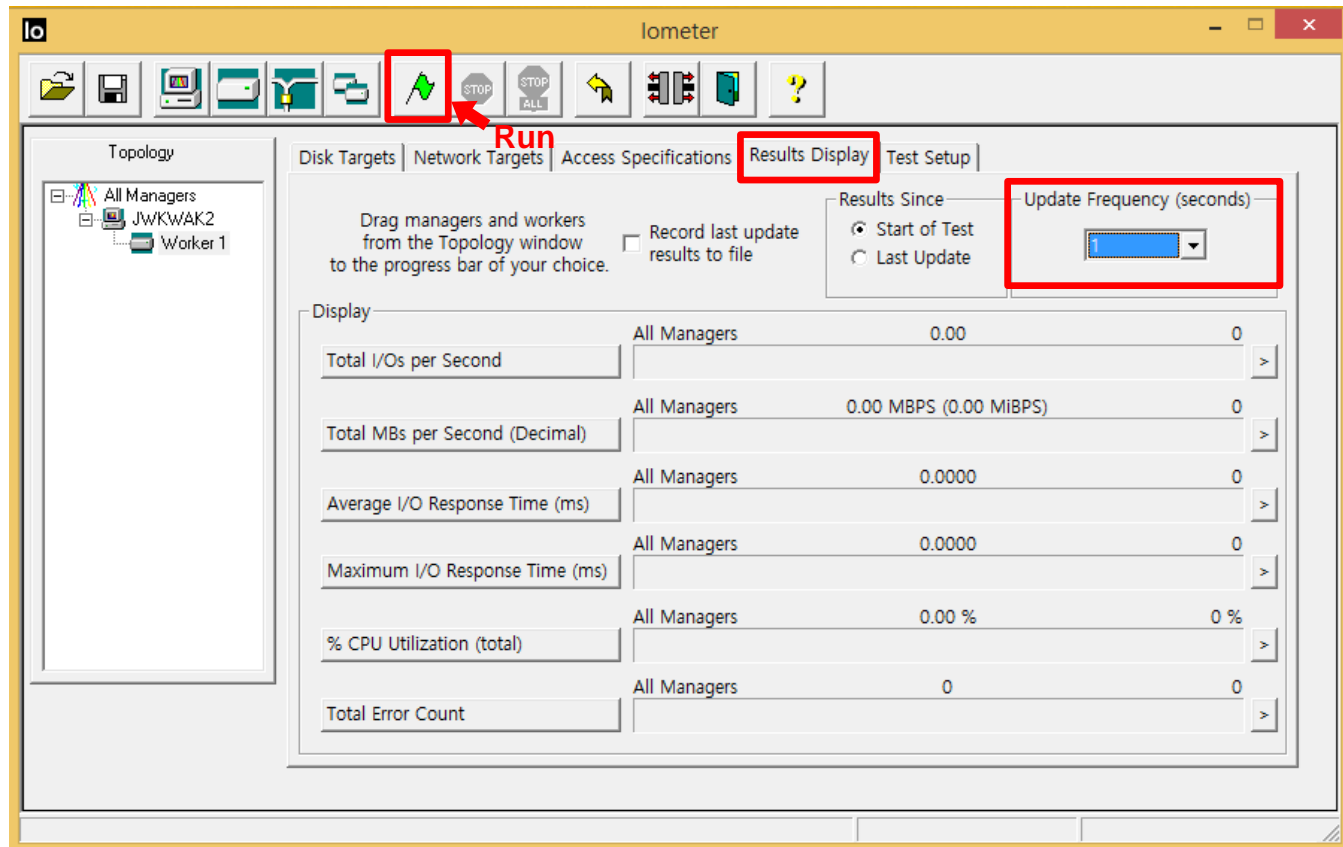
Set the Sufficient Number of Outstanding I/Os

- **X channel – Y way flash array needs “X * Y” outstanding flash requests at least for utilizing multi channel/way parallelism**
 - In case of a Cosmos+ OpenSSD configuration (8 channel – 8 way, 16KB page size), “128KB sequential write” access specification needs 8 outstanding I/Os at least
 - 64 (128KB/16KB * 8) outstanding flash requests
 - Recommend the environment generating $2 * X * Y$ outstanding flash requests



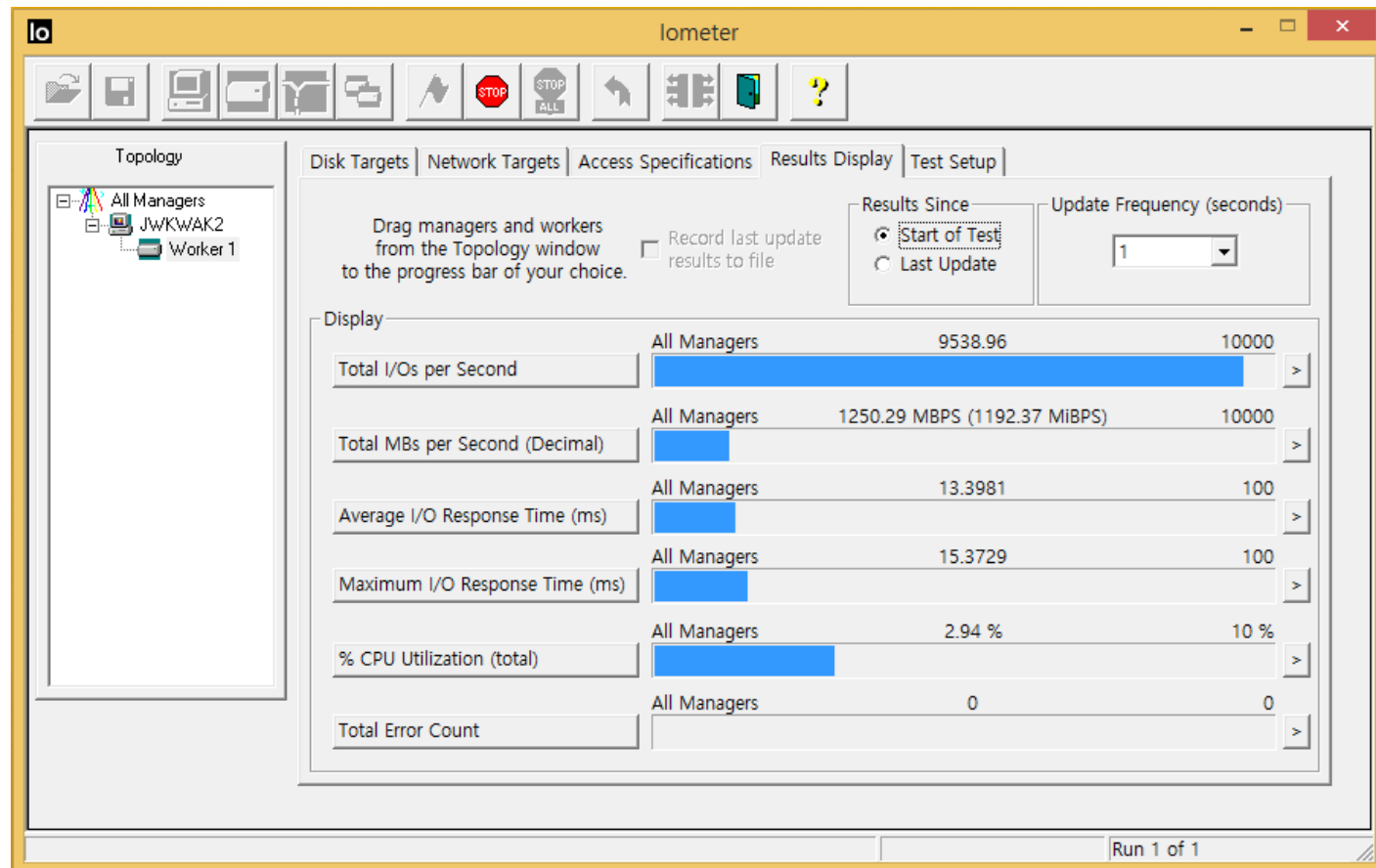
Run an Evaluation

- 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

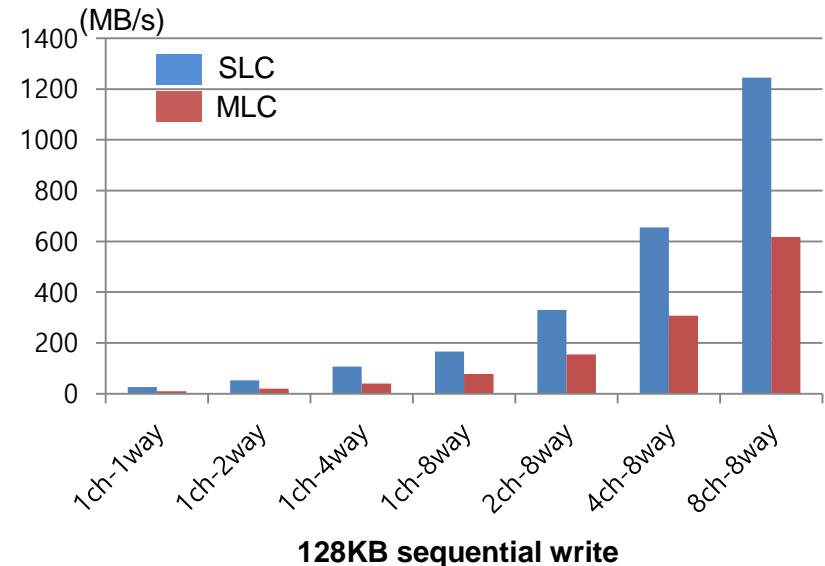
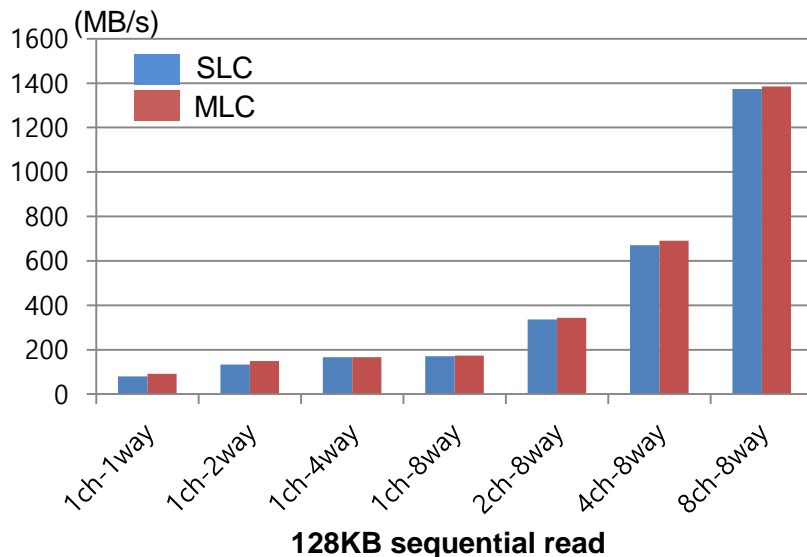
Evaluation Results (1 / 3)

■ Maximum throughput/channel \cong 173 MB/s

- 100Mhz DDR flash bus (bit width: 8) \rightarrow 200MB/s
- 16,384 + 1,664(spare) byte page \rightarrow 90% (16,384/18048) of 200MB/s = 181MB/s
- Overhead of flash memory controller \rightarrow 173 MB/s

■ Measured throughput/channel of 8channel-8way configuration

- Sequential read: 99% of maximum throughput
- Sequential write: 45~90% of maximum throughput



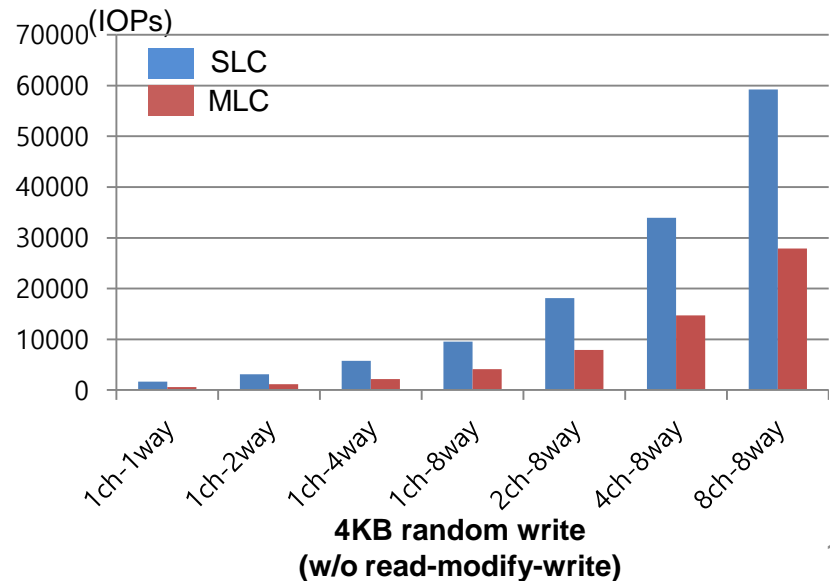
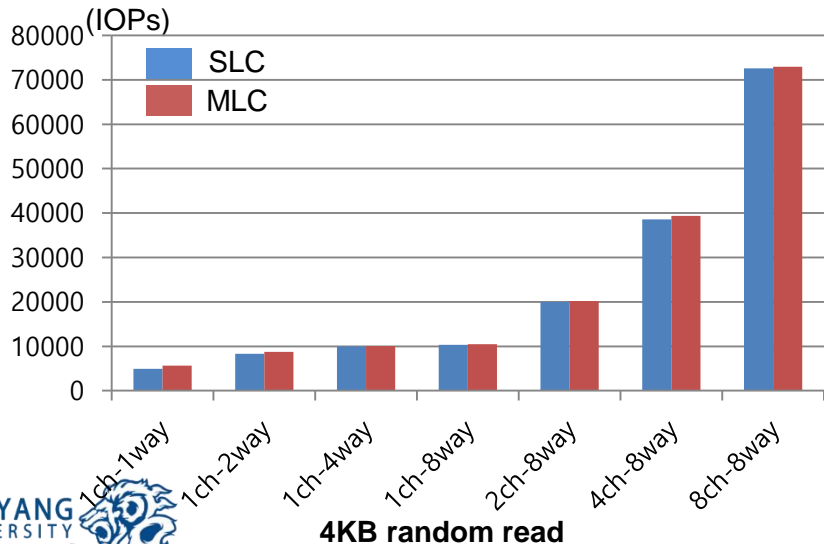
Evaluation Results (2 / 3)

■ Maximum 4KB IOPs/channel \approx 10812 IOPs

- Page mapping \rightarrow a page is accessed in order to access 4KB data
- $173\text{MB/s}(\text{Maximum throughput/channel}) \div 16\text{KB}(\text{page size}) = 10812 \text{ IOPs}$

■ Measured throughput/channel

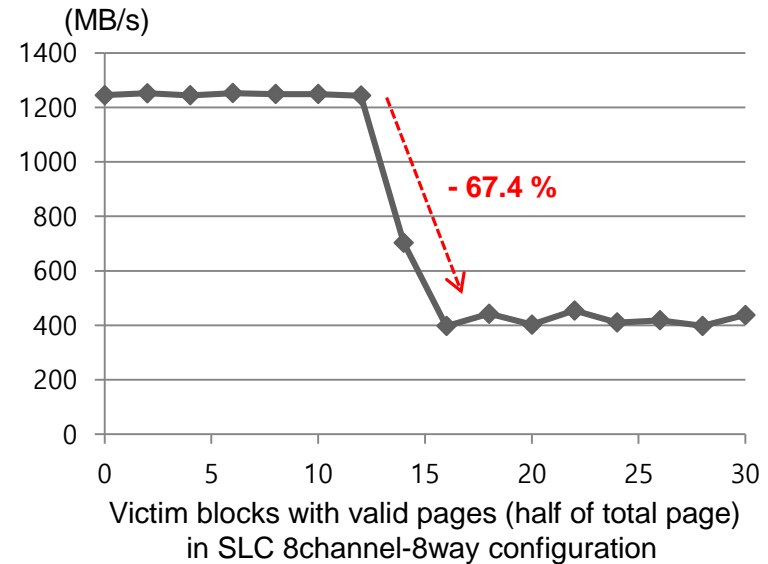
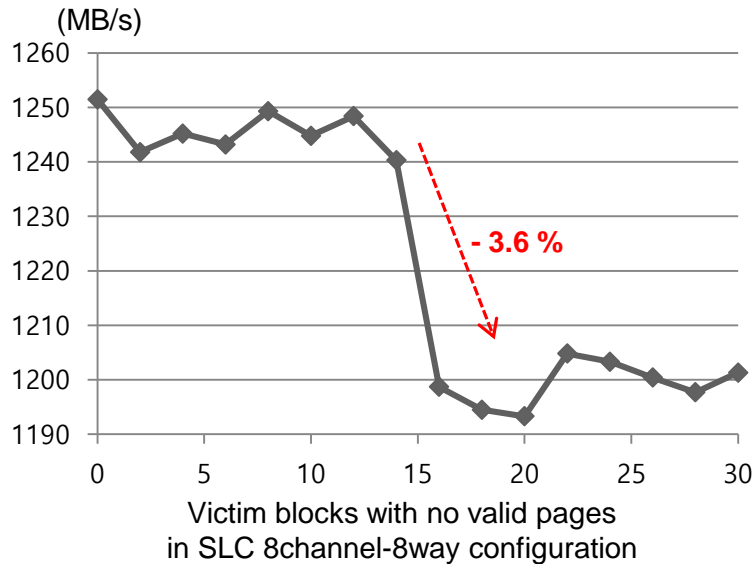
- 1channel-8way configuration
 - Random 4KB read: 96% of maximum 4KB IOPs
 - Random 4KB write: 38~88% of maximum 4KB IOPs
- 8channel-8way configuration
 - SW-based scheduling has a larger latency in many channel/way configuration
 - Scheduling latency can increase the idle time of hardware controllers



Evaluation Results (3 / 3)

■ Performance degradation by on-demand garbage collection

- After all available blocks are used, garbage collection is triggered steadily
- Effect of performance degradation varies depending on copy operation overhead
 - Copy operation overhead depends on the number of valid page belong to victim blocks



Thank You

