FPGA Tool Basic

2023.07.19

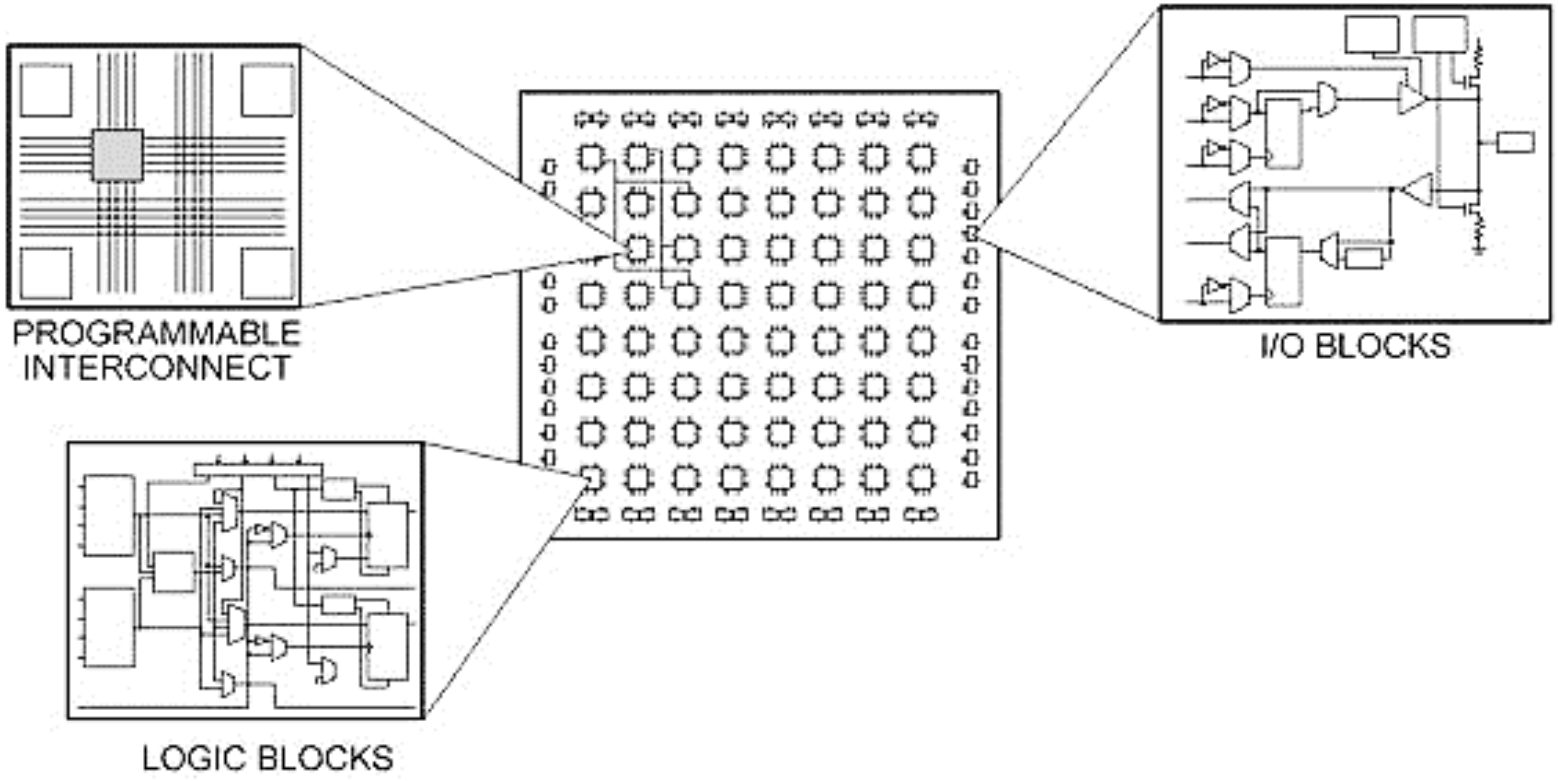
Yi Chen

CastLab

What is FPGA?

- **FPGA(Field Programmable Gate Array)**

  : Integrated Circuit designed to be configured by a customer or a designer after manufacturing.



CastLab

What is FPGA?

- **FPGA**
  - Hardware Design Flow Capable with Single Tool (e.g. Xilinx or Quartus)
  - Simple Design Revision
  - More accessibility to IP
  - Fixed Amount of Resource
  - Limited Customization During Place & Route

- **ASIC**
  - Multiple Tool for Each Design Steps (e.g. Cadence and Synopsys)
  - High Cost for Hardware Revision
  - Large Time Investment for Development
  - Implementation of Desired Floorplan
  - Full Customization and Control of Constraints

FPGA Board Structure

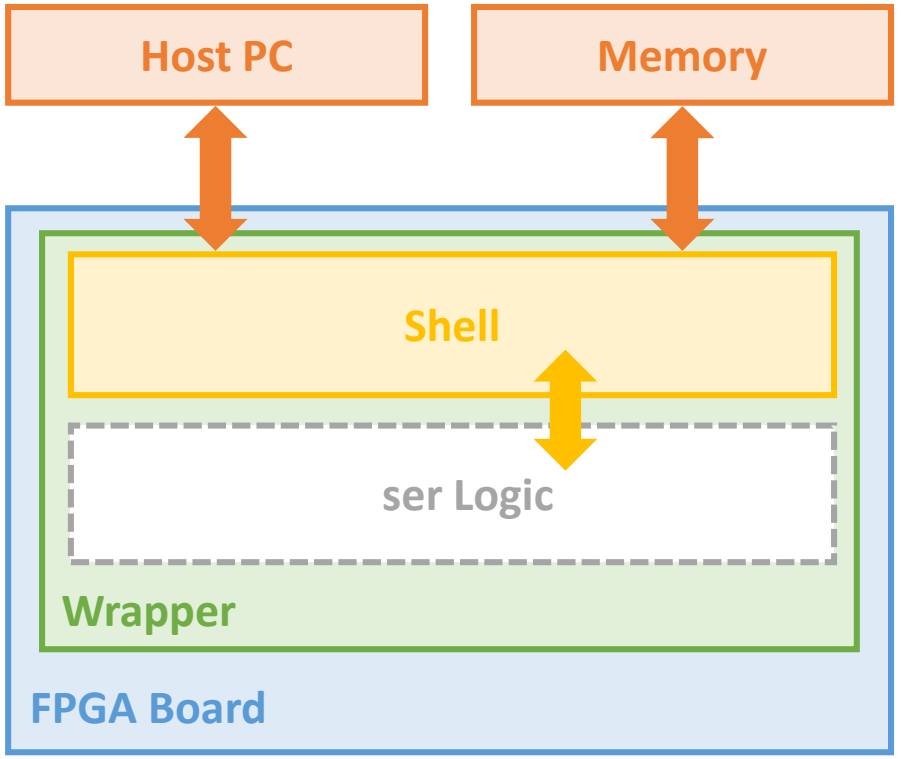
- **FPGA Shell**

  : System-level component which administrates the basic operations of device such as host communication, external memory access

  : Contains PCIe controller, Memory controller, Bus, Clock manager

- **Top level Wrapper**

  : Verilog module which connects Shell & User Logic



CastLab

How to use Vivado?

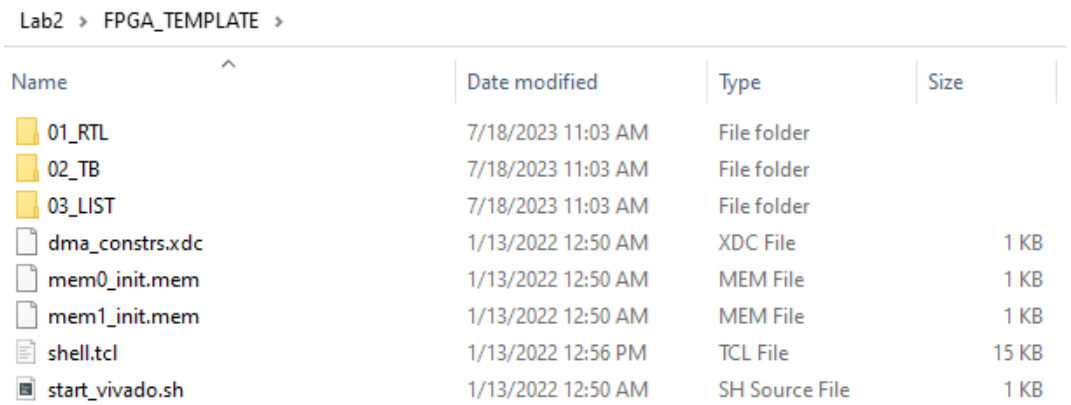
- **Required files for synthesis and implementation**

  : Sources (Designed RTL files, IPs)

  : Constraints (Timing, Pin mapping, Pblock Setting, … in .xdc files)

- **Steps to set up the project**

  1. Unzip the zipped file
  2. Move all of your working source codes of lab1 to 01_RTL folder
  3. Modify the vlist.f file accordingly to include all of your source codes.



CastLab

How to use Vivado?

- **Use the script to create the project, add constraints and pre-designed source**

  4. Script *shell.tcl* for generating pre-designed source (ex. Block Design) automatically

  Make Tcl Script for customized Block Design using ‘Export Block Design’

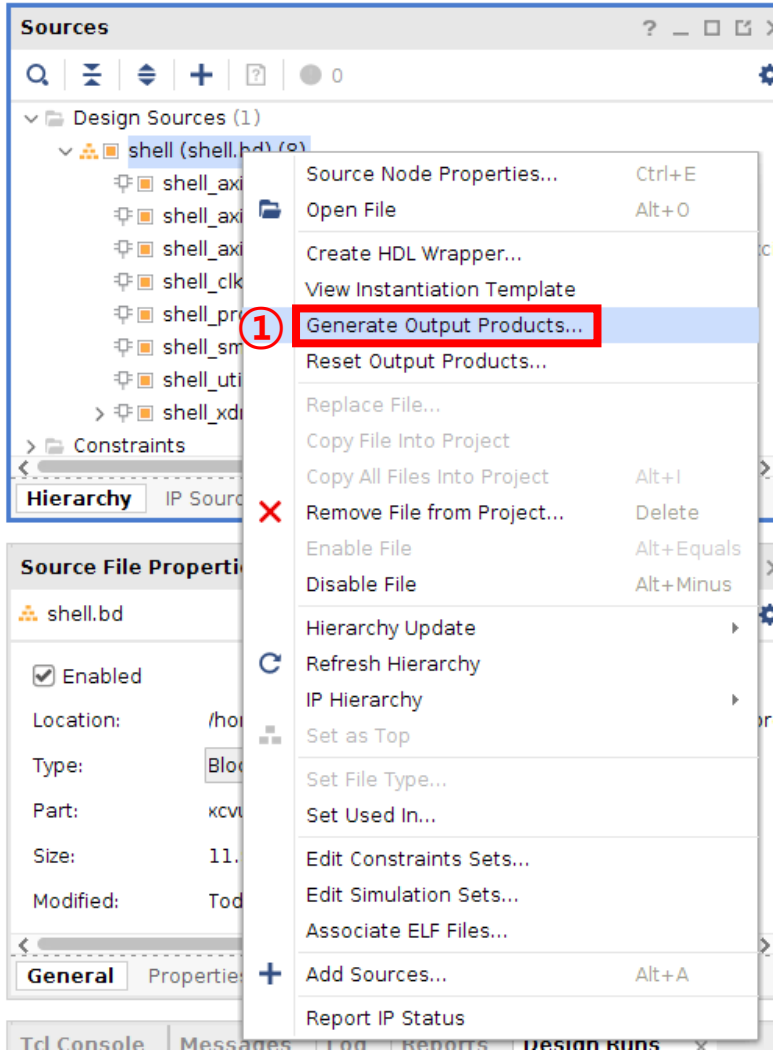
```
undergraduate@u50-server4:~/Yi$ ls
01_RTL  02_TB  03_LIST  dma_constrs.xdc  mem0_init.mem  mem1_init.mem  shell.tcl  start_vivado.sh
undergraduate@u50-server4:~/Yi$ ./start_vivado.sh
-bash: ./start_vivado.sh: Permission denied
undergraduate@u50-server4:~/Yi$ chmod 777 start_vivado.sh
undergraduate@u50-server4:~/Yi$ ./start_vivado.sh
undergraduate@u50-server4:~/Yi$
****** Vivado v2020.2 (64-bit)
  **** SW Build 3064766 on Wed Nov 18 09:12:47 MST 2020
  **** IP Build 3064653 on Wed Nov 18 14:17:31 MST 2020
    ** Copyright 1986-2020 Xilinx, Inc. All Rights Reserved.

start_gui
```

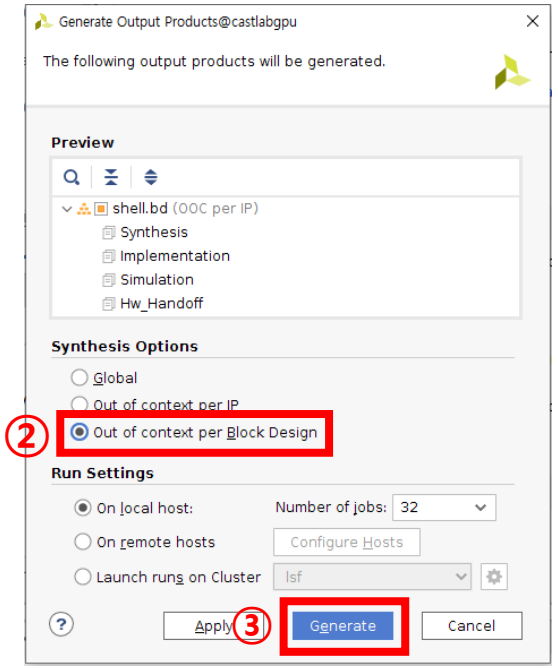
How to use Vivado?

- **Generate Output Products**

  : Convert Block Design to Verilog File

  : Generate Pre-Synthesized Module

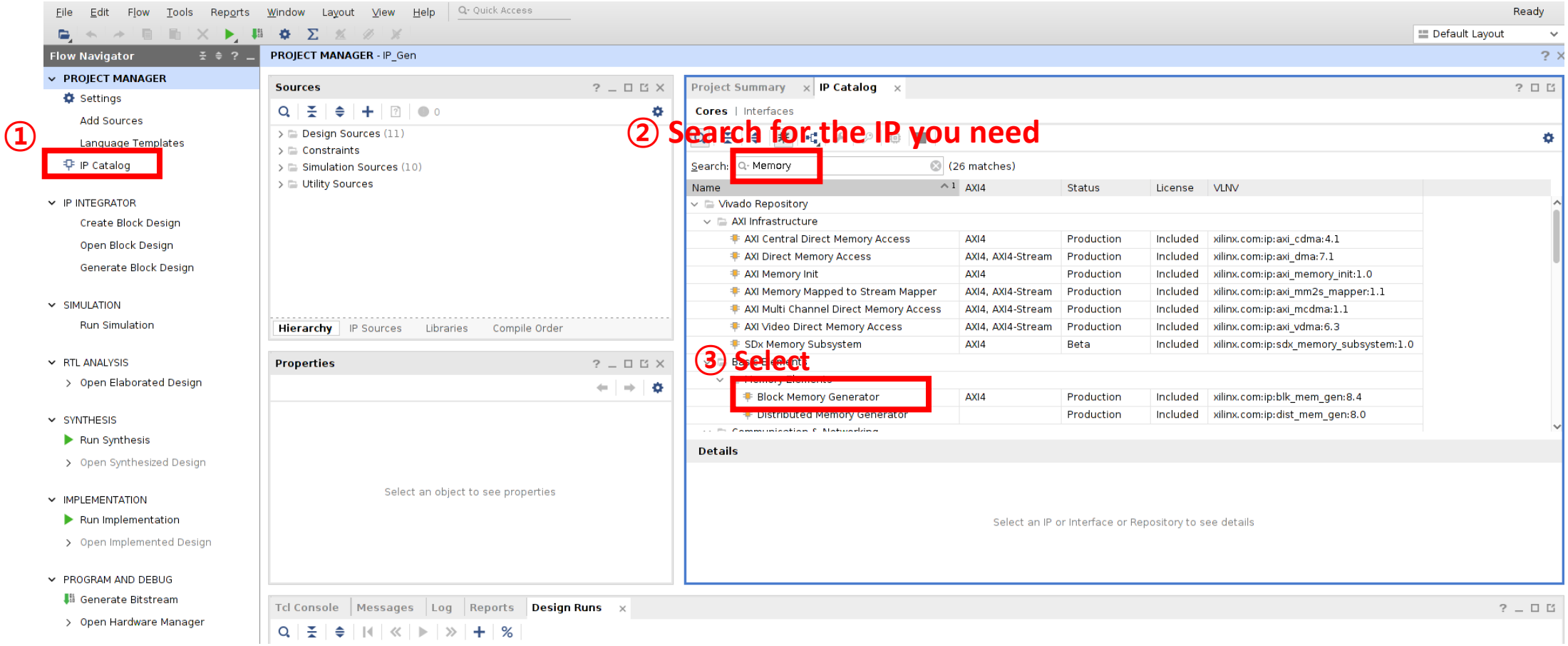




CastLab

How to use Vivado?

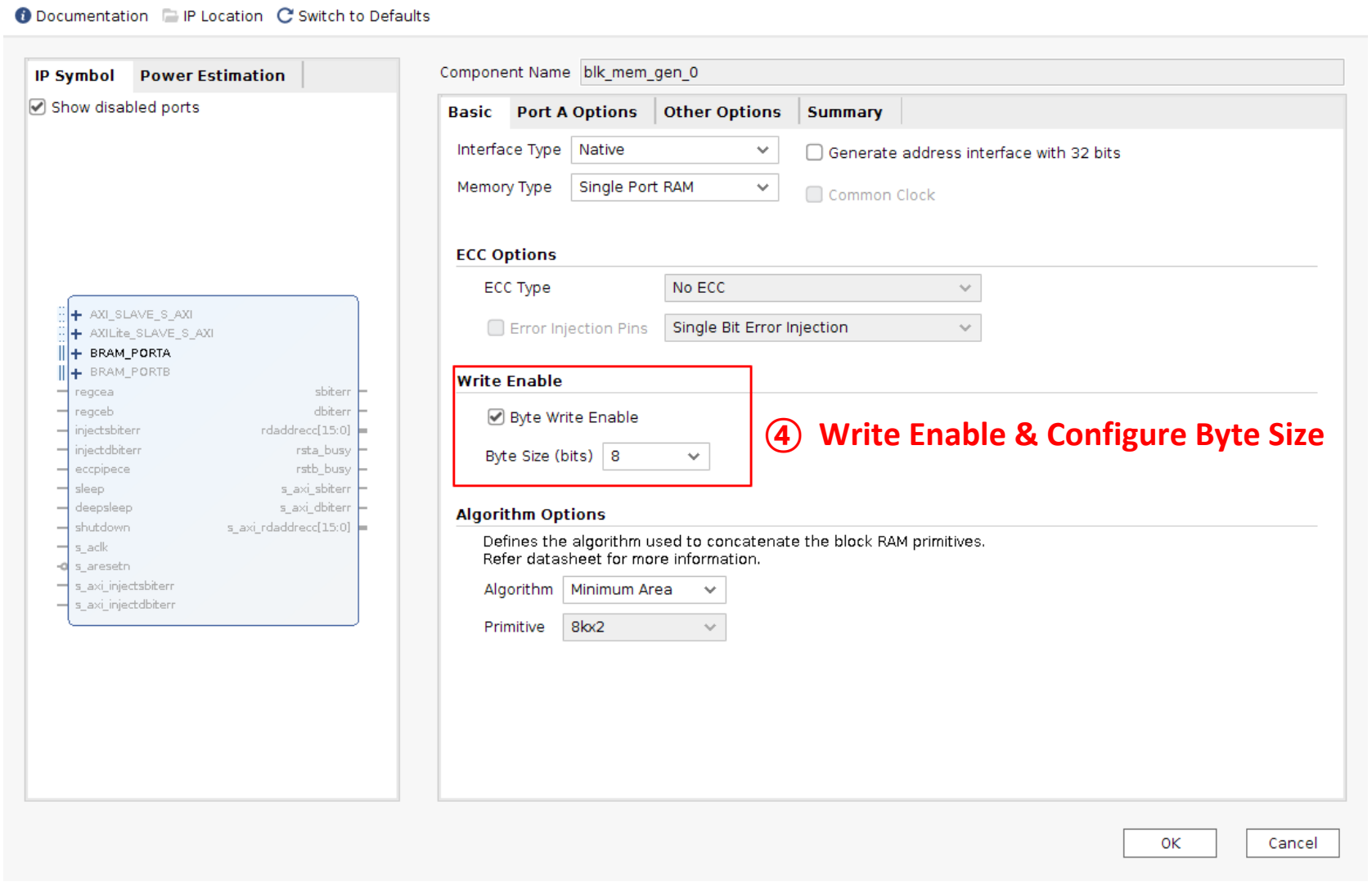
- **IP Generation**



CastLab

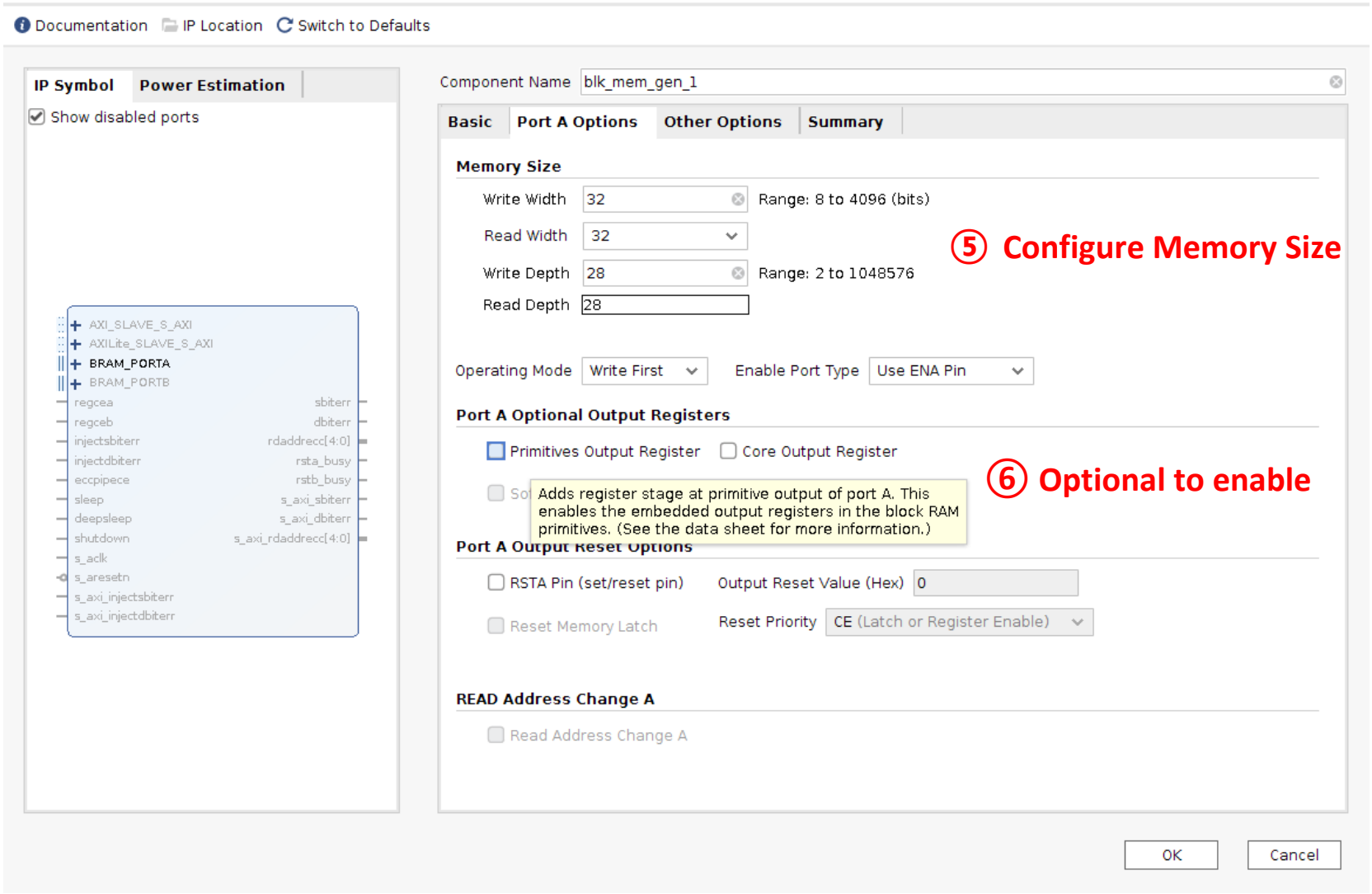
How to use Vivado?

- **IP Generation**



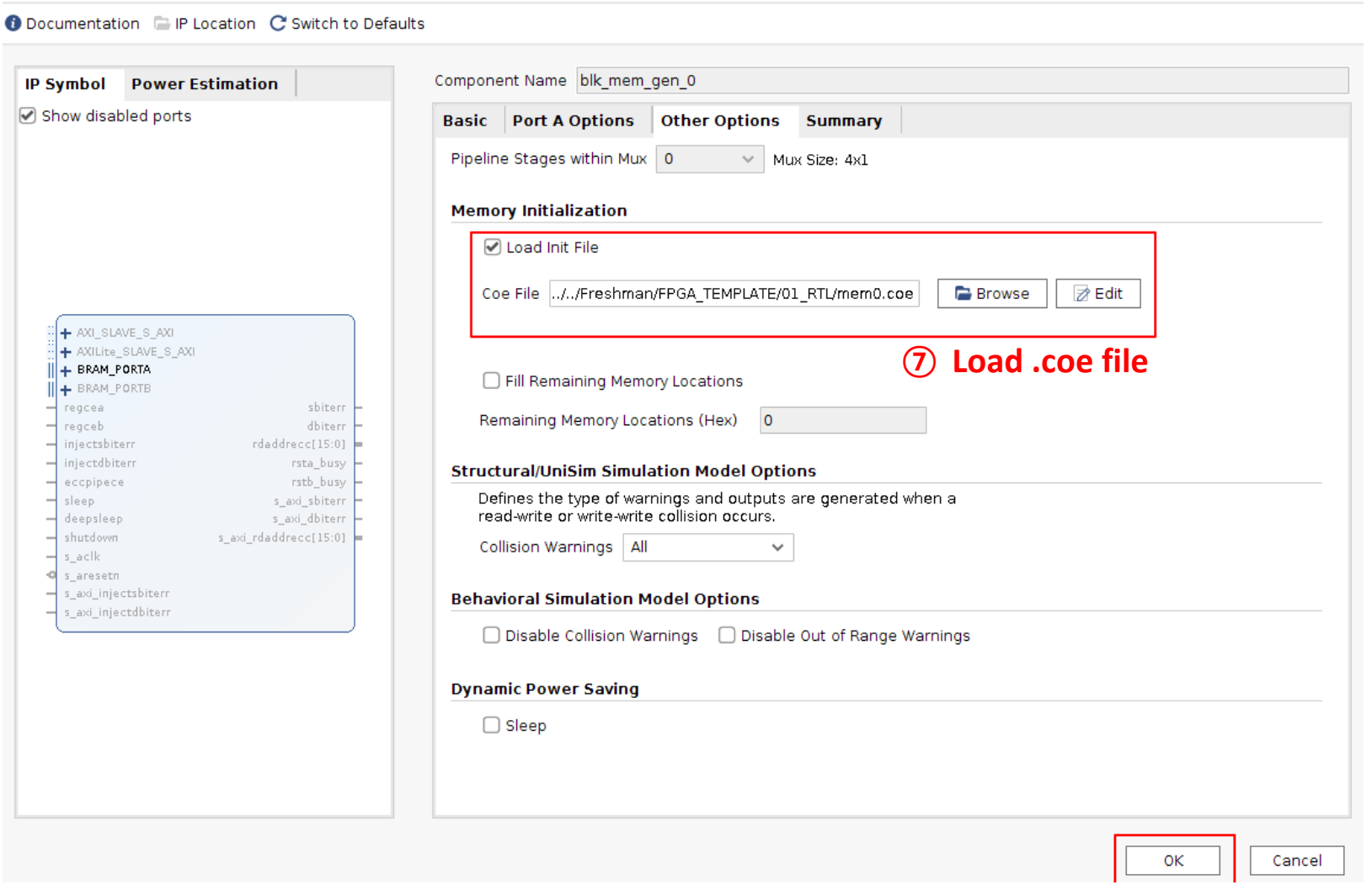
How to use Vivado?

- **IP Generation**



How to use Vivado?

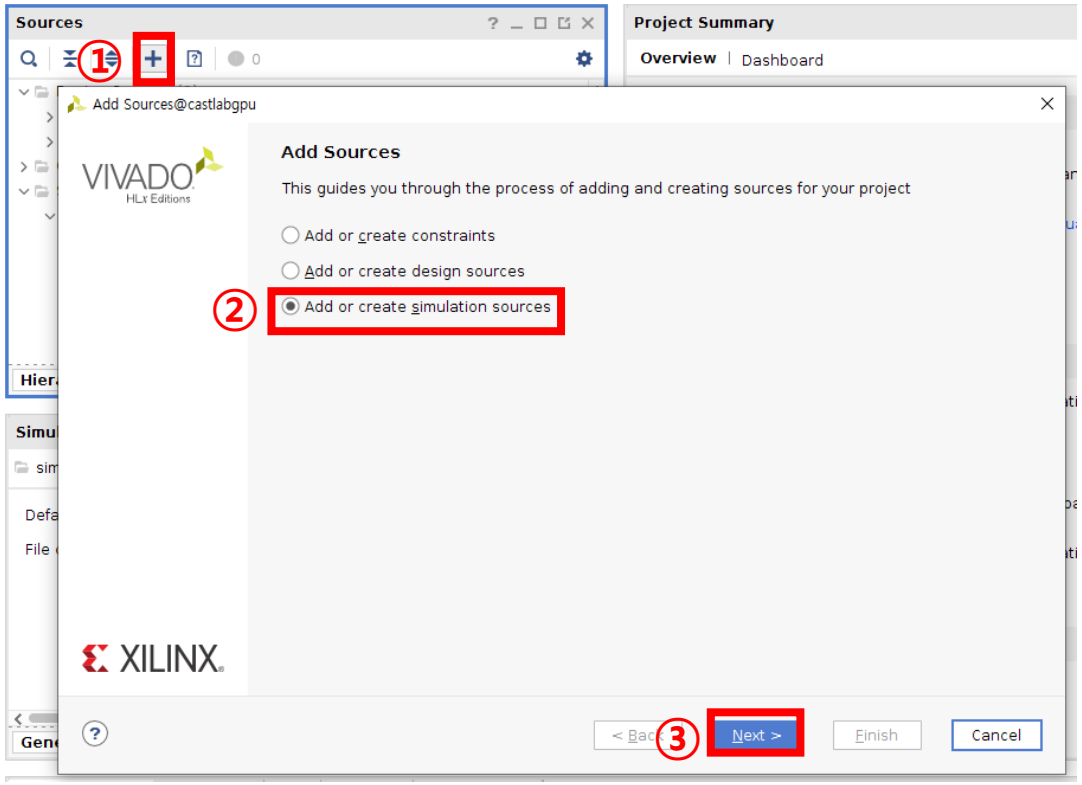
- **IP Generation**

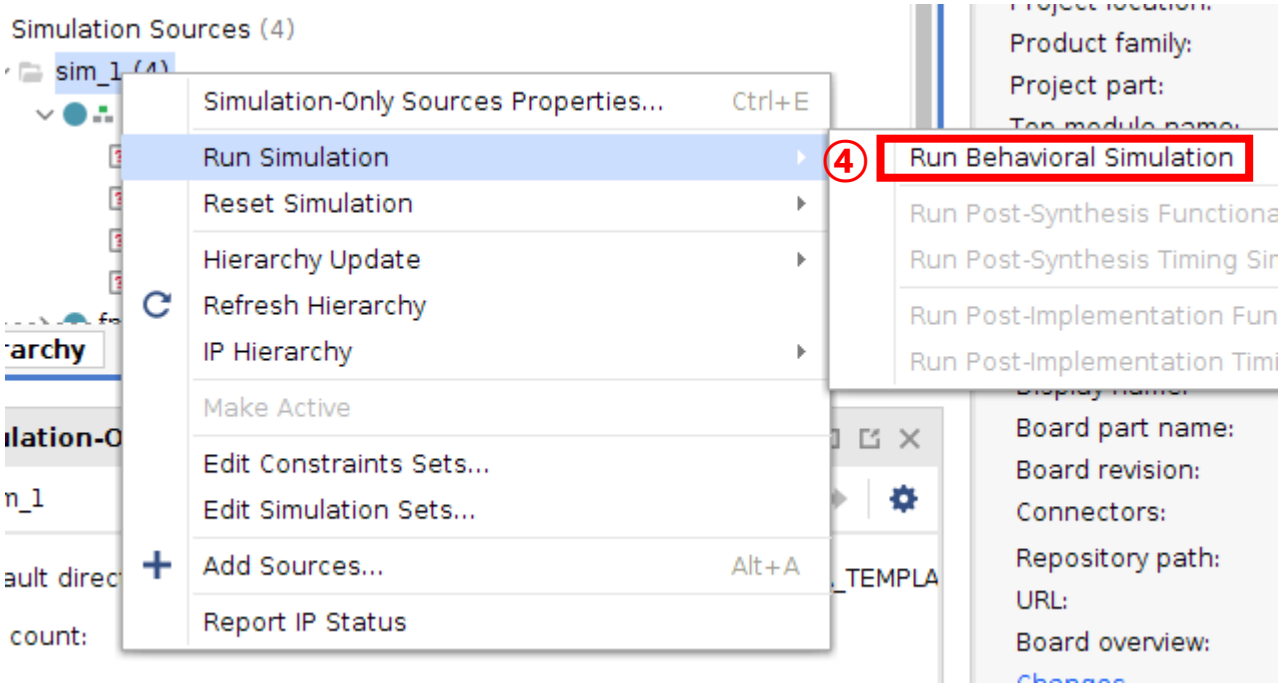


How to use Vivado?

- **Functional Simulation**

  : Simulate Functional Logic through customized Testbench

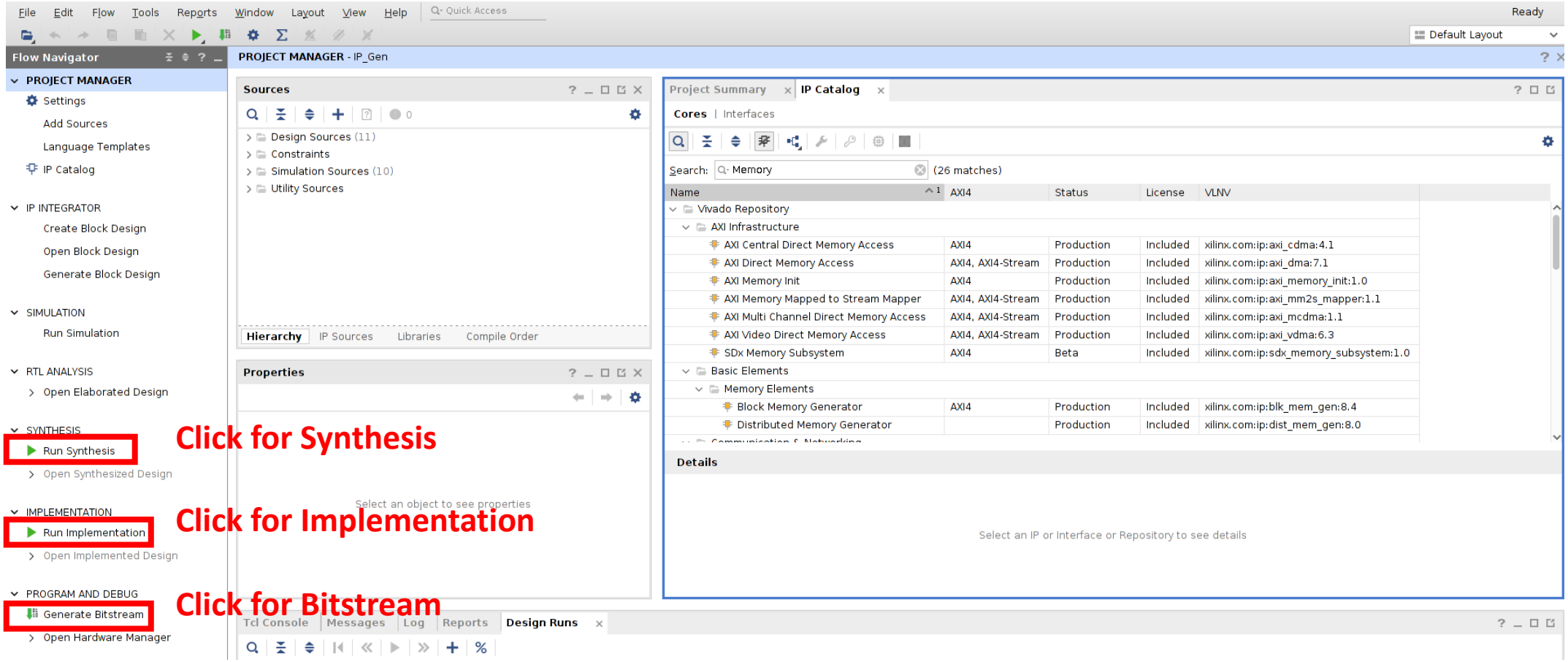




CastLab

How to use Vivado?

- **Synthesis & Implementation & Generate Bitstream**



CastLab

Integrated Logic Analyzer

- **ILA(Integrated Logic Analyer)**

  : The customizable Integrated Logic Analyzer (ILA) IP core is a logic analyzer that can be used to monitor the internal signals of a design.

  : The ILA core includes many advanced features of modern logic analyzers, including boolean trigger equations and edge transition triggers. Because the ILA core is synchronous to the design being monitored, all design clock constraints that are applied to your design are also applied to the components of the ILA core.

How to make ILA?

- **Before Synthesis**

  : Write (* MARK_DEBUG="true" *) ahead of input/output port which you want to check @ RTL

  : Clock & Reset signals are not allowed to make ILA

```
// Memory 0 (Base Address : 0x0010_0000)
output logic                                mem0_en,
output logic [MEM_STRB_WIDTH-1:0]           mem0_we,
output logic [MEM_ADDR_WIDTH-1:0]           mem0_addr,
output logic [MEM_DATA_WIDTH-1:0]           mem0_wdata,
(* MARK_DEBUG="true" *) input  logic [MEM_DATA_WIDTH-1:0]           mem0_rdata,

// Memory 1 (Base Address : 0x0020_0000)
output logic                                mem1_en,
output logic [MEM_STRB_WIDTH-1:0]           mem1_we,
output logic [MEM_ADDR_WIDTH-1:0]           mem1_addr,
output logic [MEM_DATA_WIDTH-1:0]           mem1_wdata,
input  logic [MEM_DATA_WIDTH-1:0]           mem1_rdata,

(* MARK_DEBUG="true" *) output logic [1:0]                          led
```

→ **Probe**

→ **Probe**

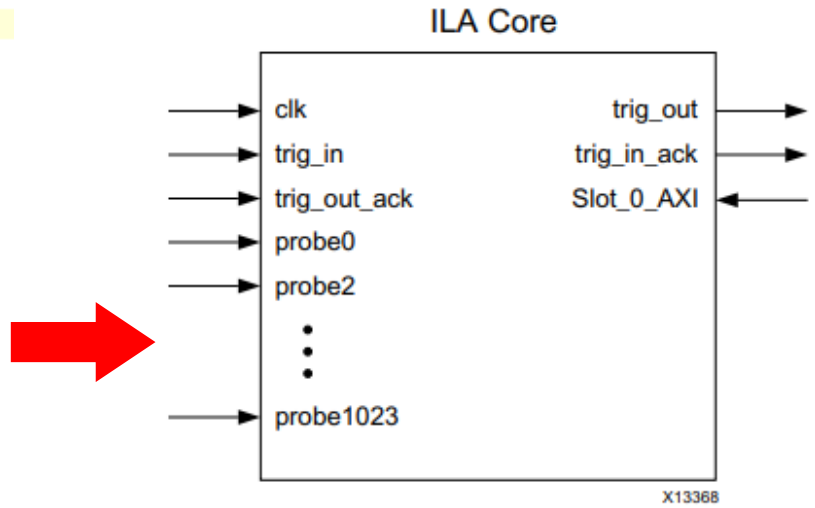
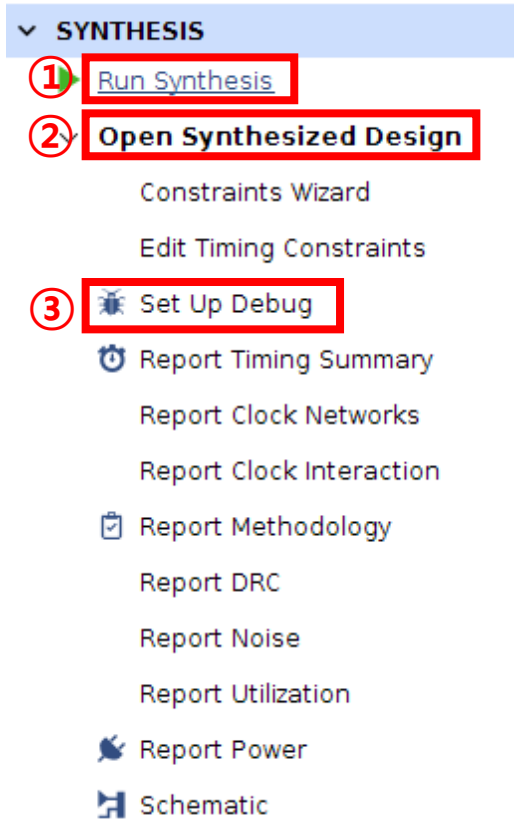


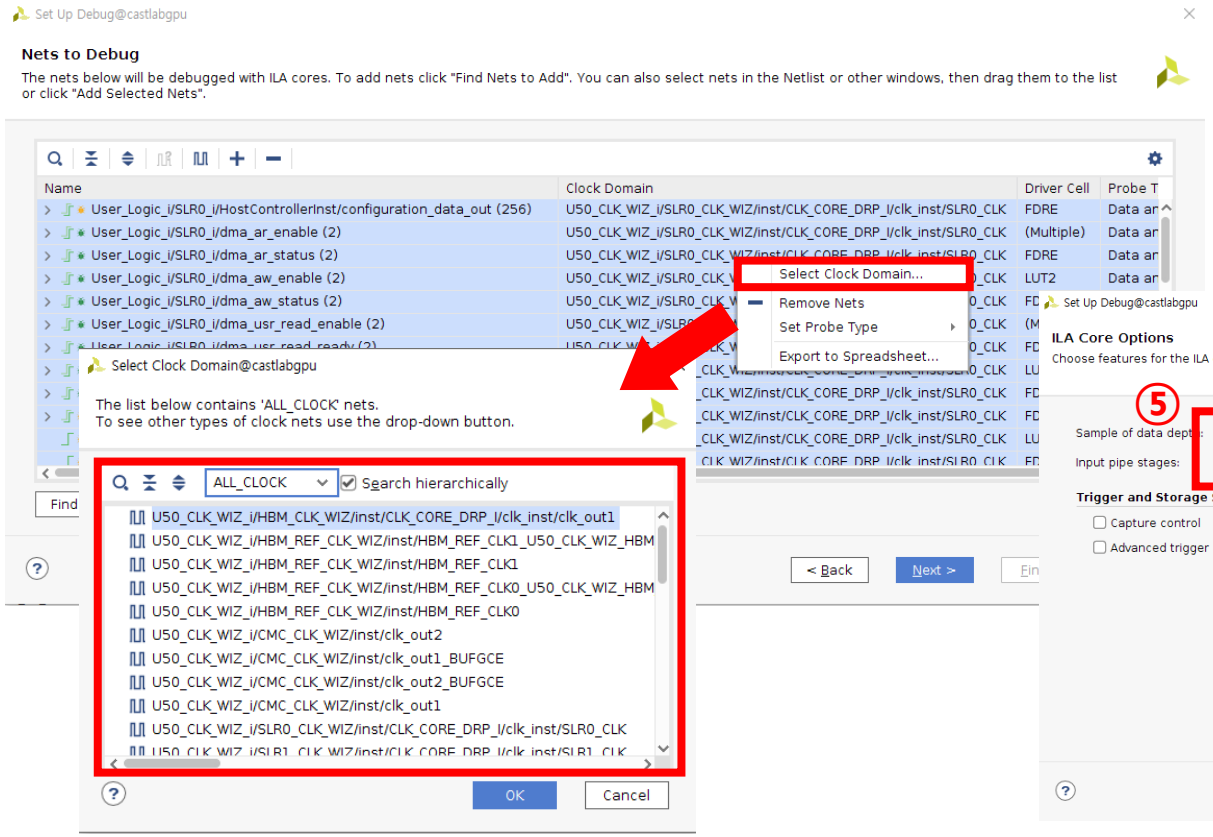
Figure 1-1: **ILA Core Symbol**

CastLab

How to make ILA?

- **After Synthesis**





④ Select proper Clock Domain

⑥ Impl. & generate Bitstream

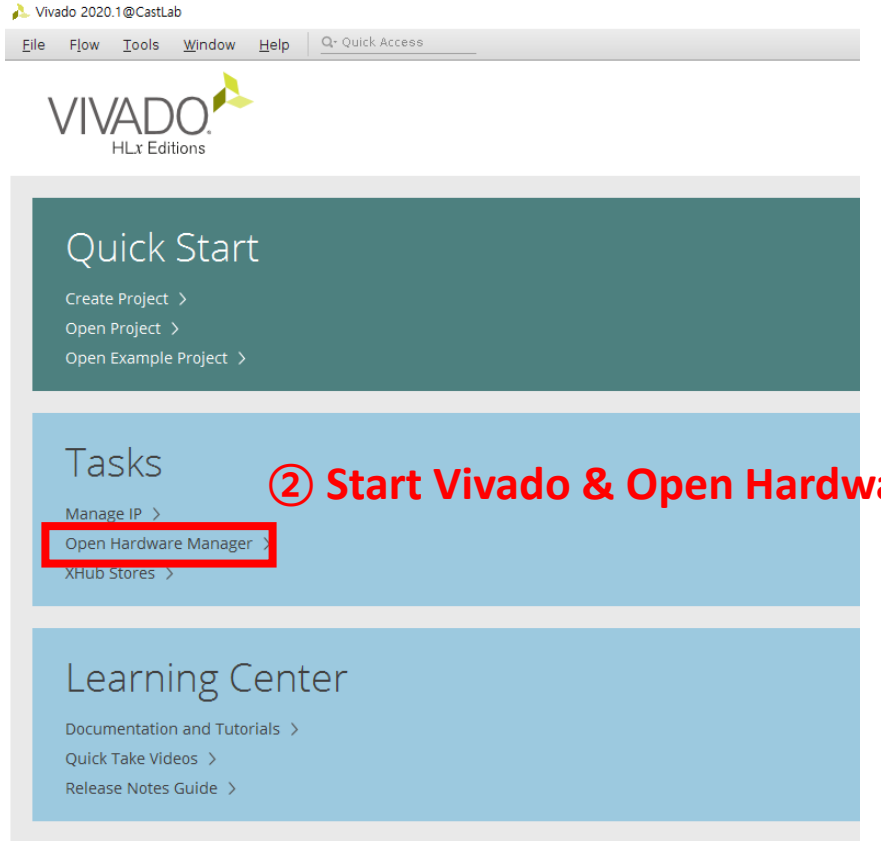
CastLab

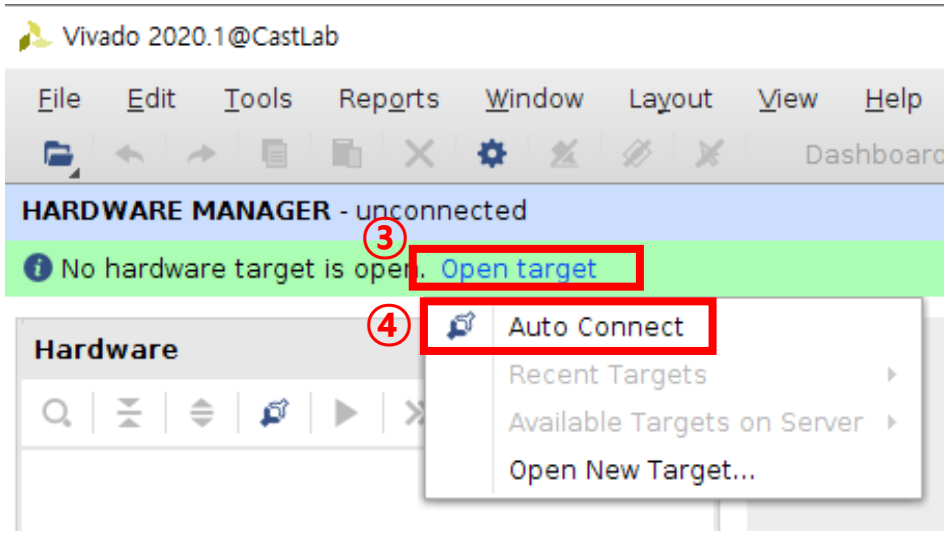
How to do Board Verification?

- **Program Device**

② Move your own bitstream (.bit) to test PC server (ILA (.ltx) file if necessary)

- File Location : $project_folder/$project_name/$project_name.runs/impl_1

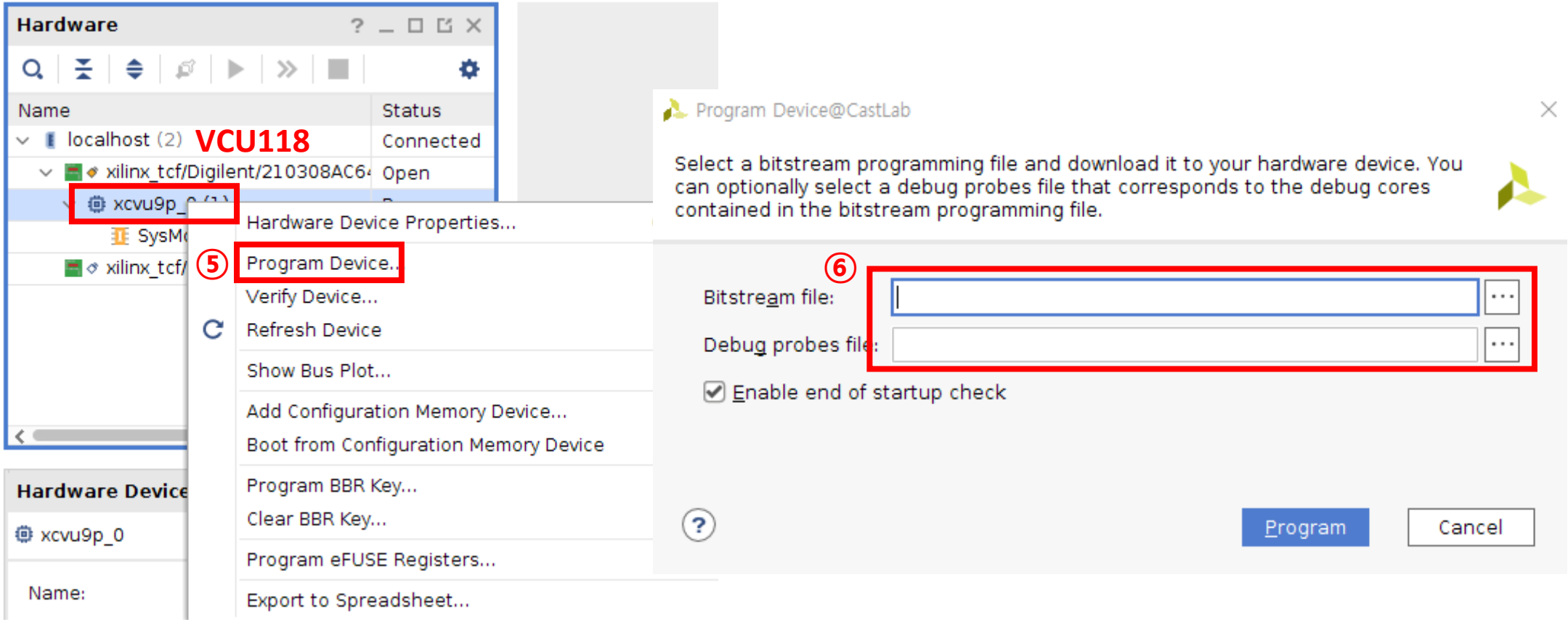




CastLab

How to do Board Verification?

- **Program Device**



CastLab

How to do Board Verification?

- **Test PC setting**

  : Connect Host PC to Programmed FPGA Device

  - File Location : $test PC server(ID : freshman)/Workplace/Freshman-Curriculum/test/driver_command.txt

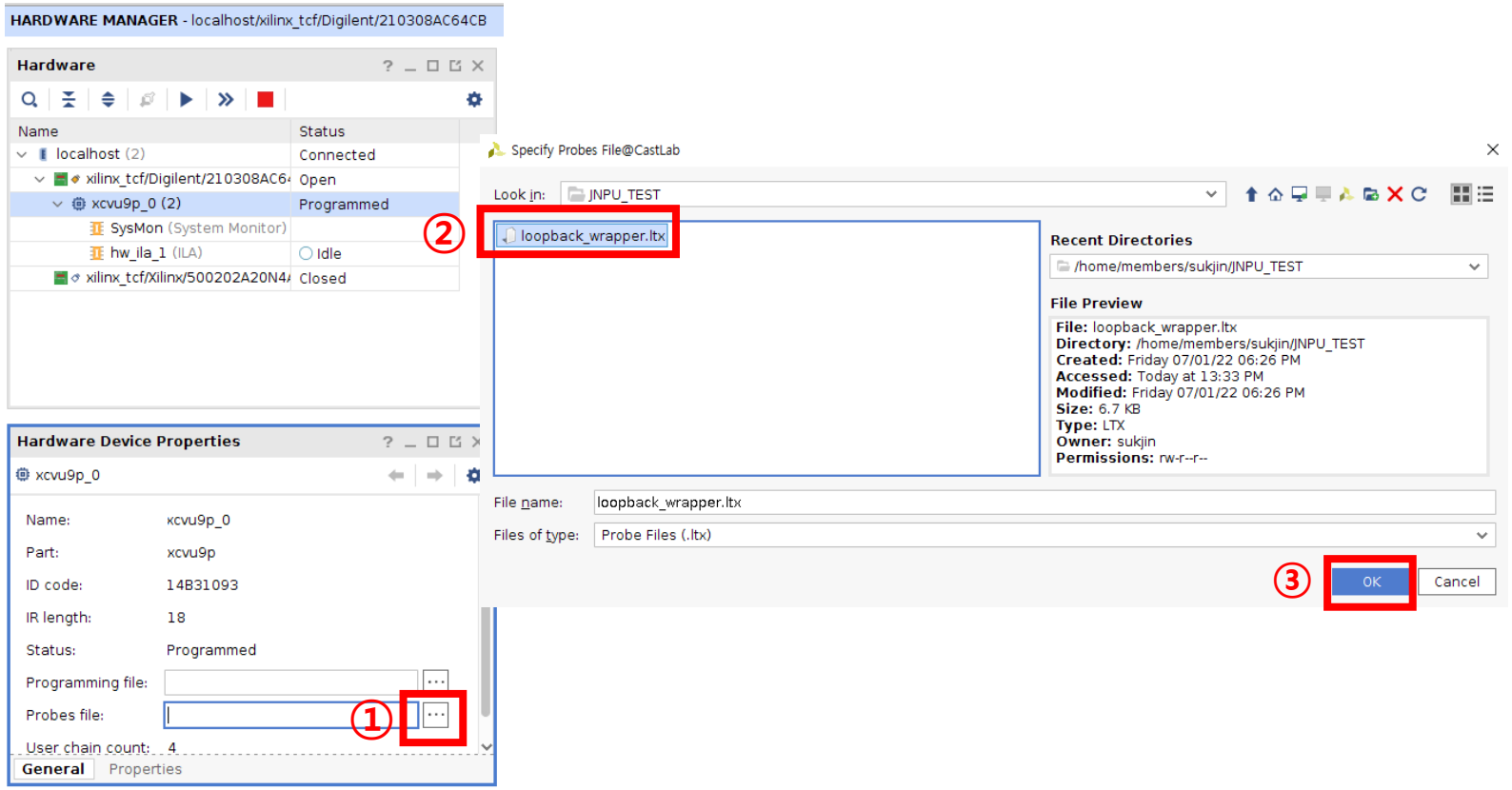
```
cd /home/members/freshman/Workplace/XilinxAR65444/Linux/Xilinx_Answer_65444_Linux_Files/tests
sudo make clean

cd /home/members/freshman/Workplace/XilinxAR65444/Linux/
sudo ./build-install-driver-linux.sh


sudo reboot


cd /home/members/freshman/Workplace/XilinxAR65444/Linux/Xilinx_Answer_65444_Linux_Files/tests
sudo ./load_driver.sh
```

How to do Board Verification?

- **Set ILA waveform**

  : After setting up test PC, Open Vivado and Do Program Device process 2,3,4 again (See previous Slide for Program Device)

  : Load ILA file(.ltx) to ILA waveform



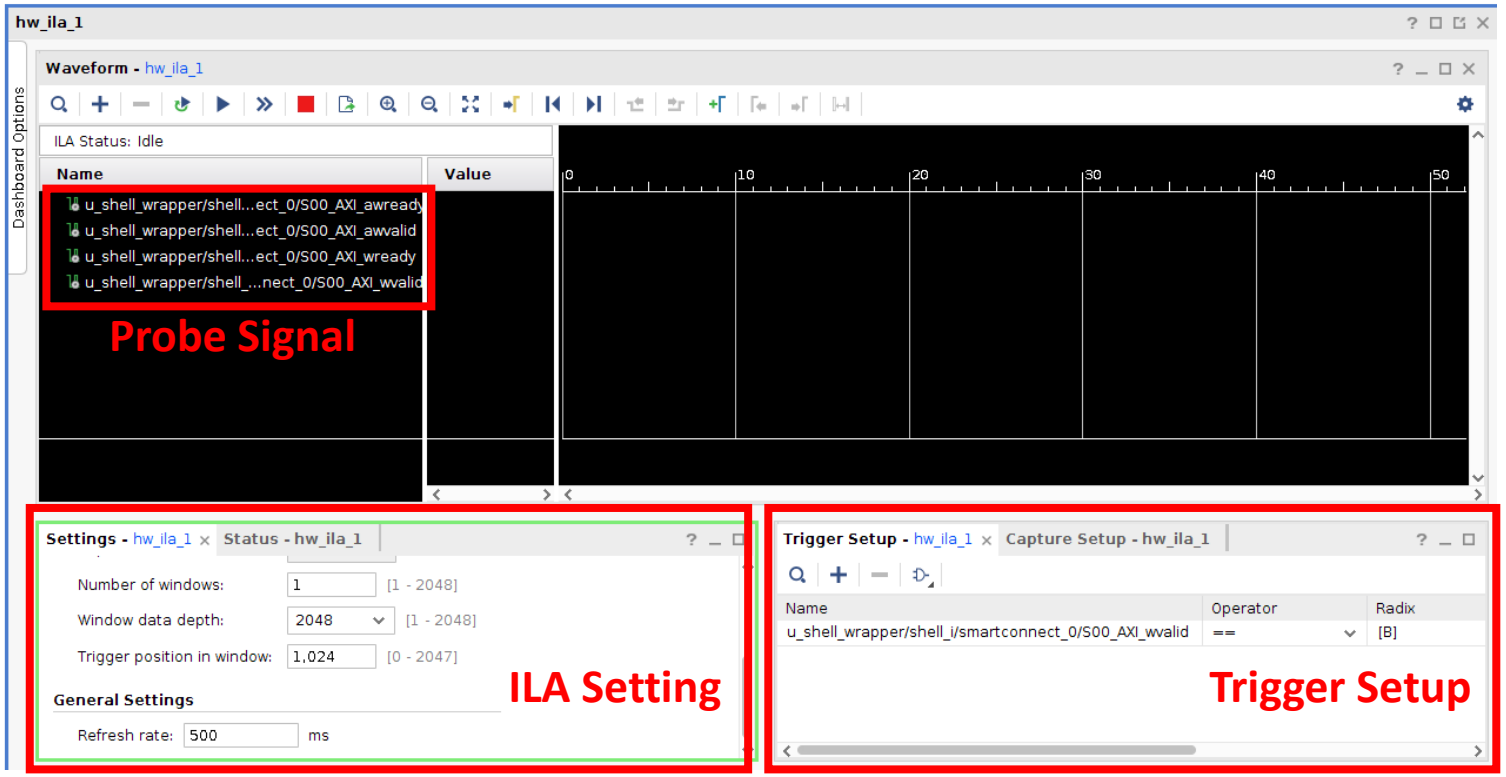
CastLab

How to do Board Verification?

- **ILA waveform**

  : ILA Setting – Choose number & position of samples you want

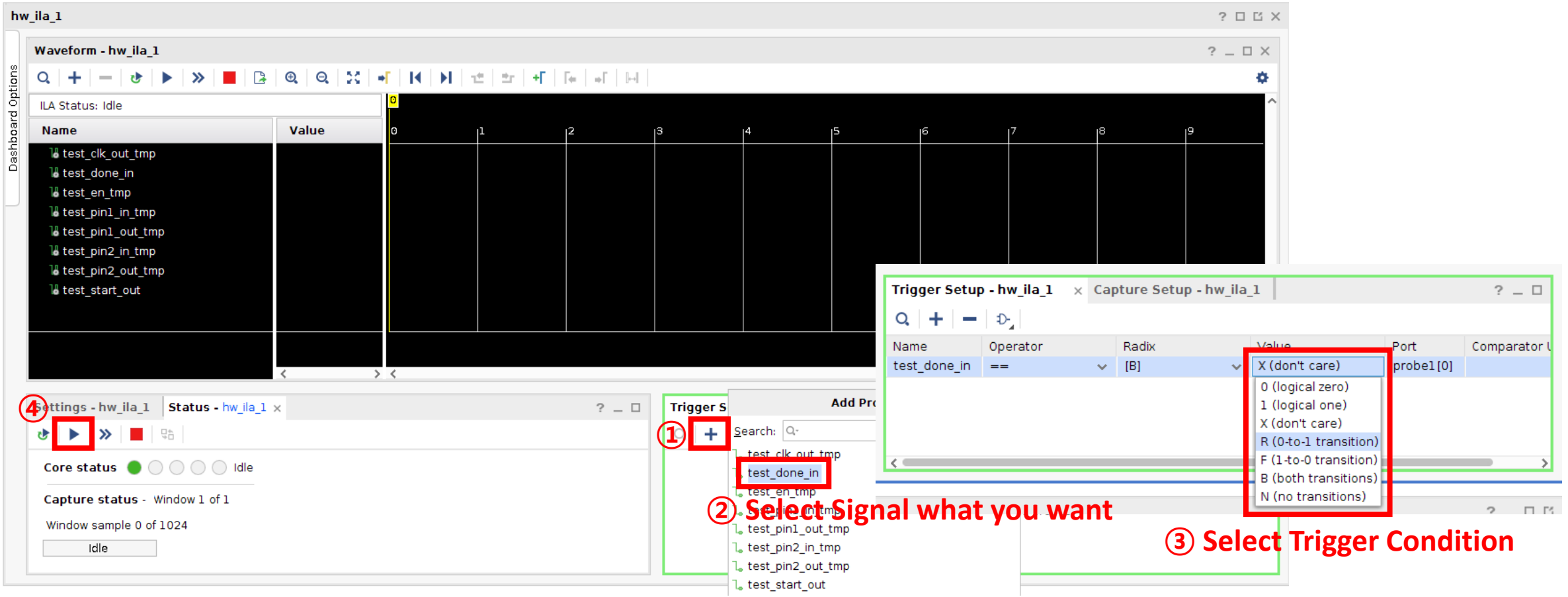
  : Trigger Setup – Set the specific conditions which trigger to get samples



CastLab

How to do Board Verification?

- **ILA waveform**



⑤ Write Command at Host PC

CastLab

How to do Board Verification?

- **Command for Host PC**

  : Command to write & read data through AXI_LITE & AXI

  - File Location : $test PC server(ID : freshman)/Workplace/Freshman-Curriculum/test/driver_command.txt

```
reg_rw          program
xdma1_user      xdma name, 0 for u50, 1 for vcu118
0x00000000      address
w               r/w
0x00000007      value, if none, read

./reg_rw /dev/xdma1_user 0x00000000 w 0x00000007

dma_to_device
dma_from_device         program
-d                      xdma name
xdma1_h2c_0
xdma1_c2h_0
-f                      file
-s                      size
-a                      address

./dma_to_device -d /dev/xdma1_h2c_0 -f data/in_helloworld.bin -s 0x00000100 -a 0x00100000 -c 1
./dma_from_device -d /dev/xdma1_c2h_0 -f result/tmp_test_hi.bin -s 0x00000010 -a 0x00100000 -c 1
```

① AXI_LITE Read/Write Command

② AXI Read/Write Command

CastLab

Reference

- **For detailed information, please refer to the '*XilinxDesignFlow_v1*' file on DB server**

Revision History

- **Version 1.0 : 21/01/19, Sukbin Lim**
- **Version 1.1 : 21/10/01, Juyeong Yoon**
- **Version 2.0 : 22/07/12, Sukjin Lee**
- **Version 2.1 : 23/07/11, Yi Chen**