

# Practice 10

- Custom IP

Computing Memory Architecture Lab.

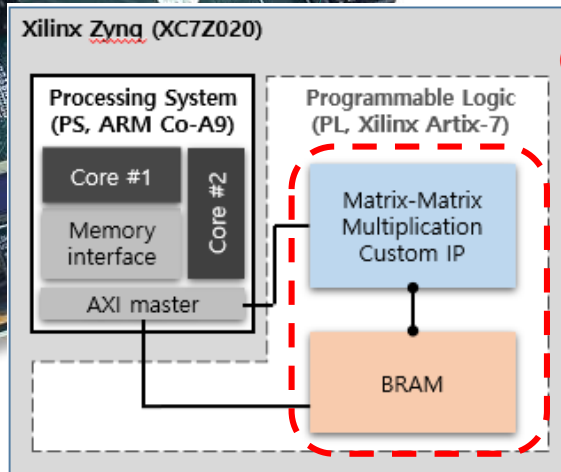
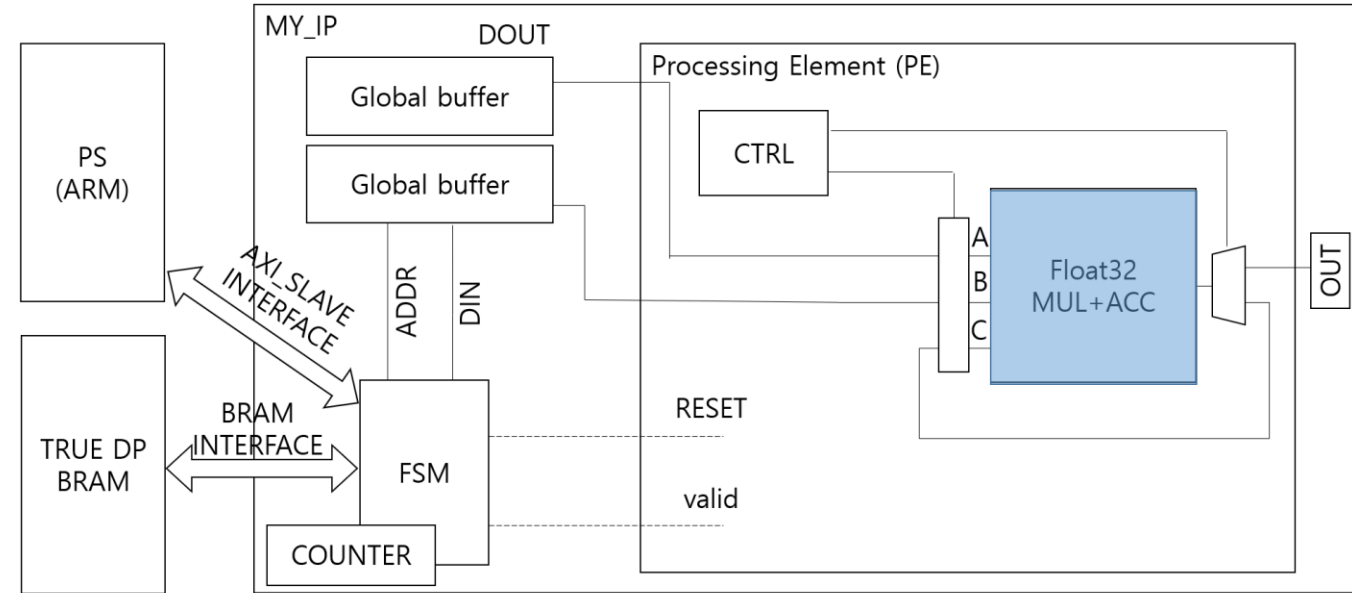
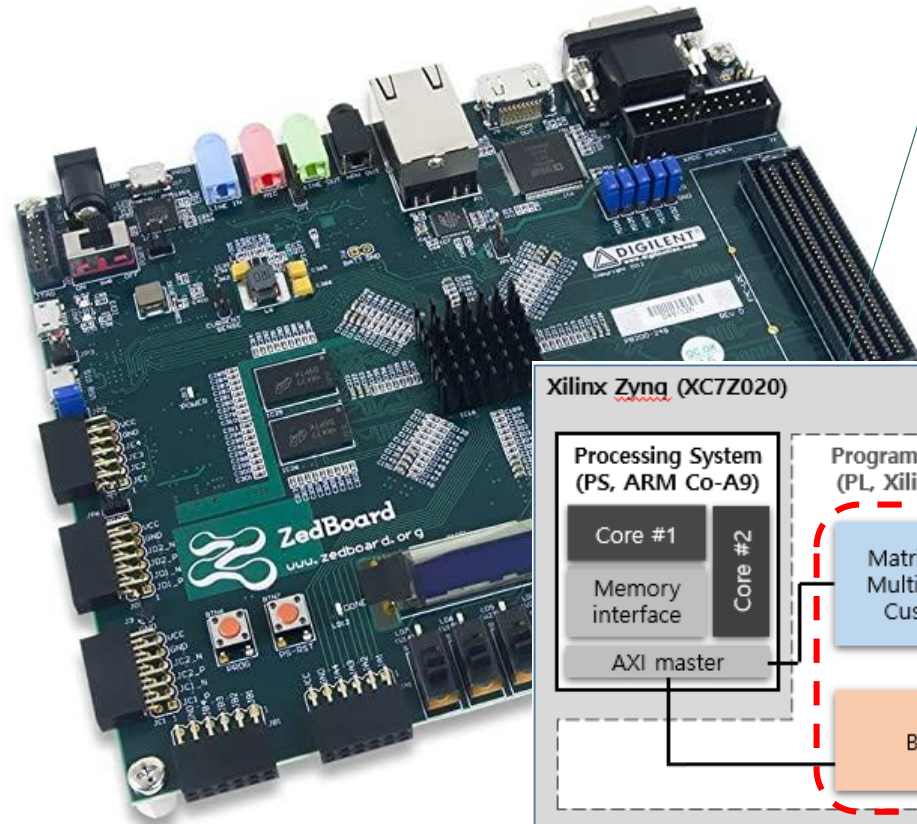
# Overview

---

## ■ Custom IP Tutorial

- Processing System + BRAM + Connectivity + Custom IP (shifter)
  - Note) Term project = PS + BRAM + Connectivity + Custom IP (M\*M multiplier)

# Final Project Overview: Matrix Multiplication IP



Custom IP on zedboard system

# Custom IP Tutorial

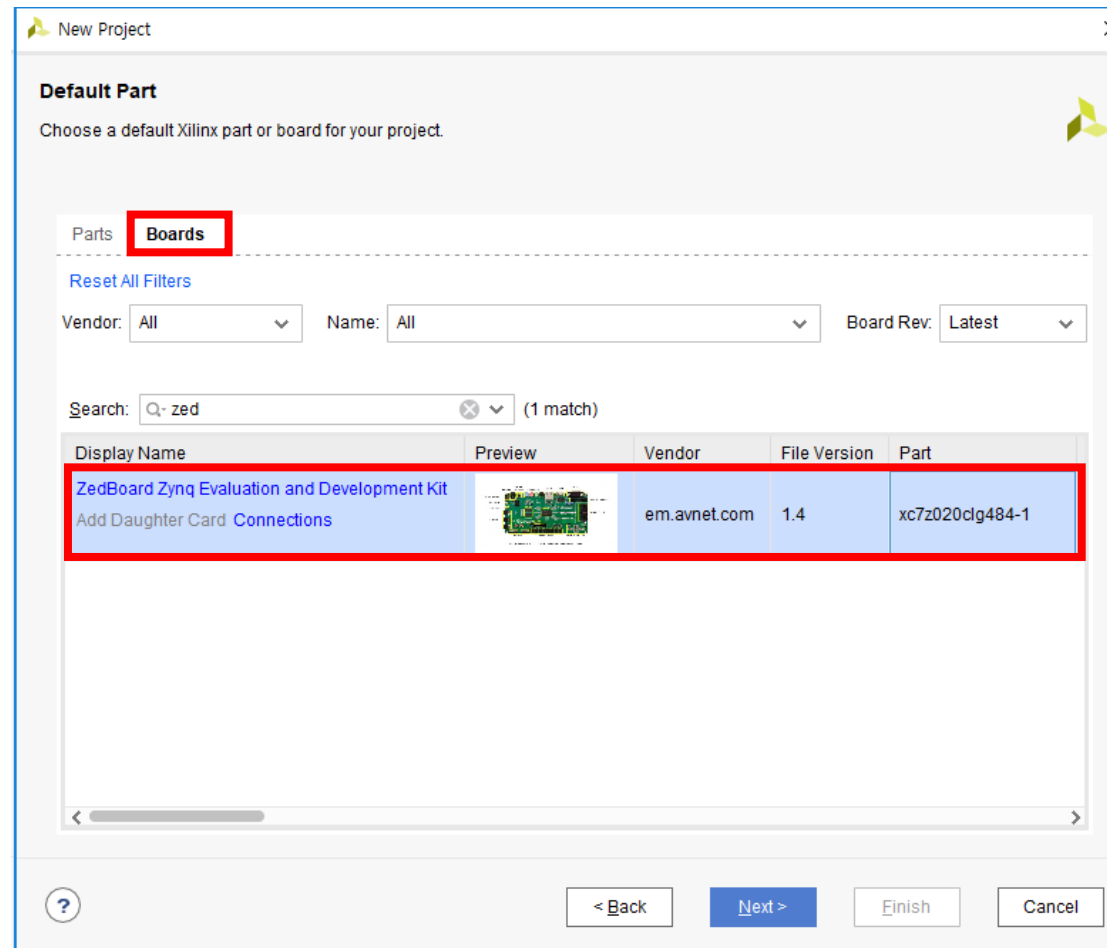
# Notations

---

- HOST\$ XXX
  - Type XXX @ the terminal of your PC
- BOARD\$ YYY
  - Type YYY @ the terminal of ZedBoard
    - Which is equivalent to the after-terminal of below command
      - HOST\$ minicom -D /dev/ttyACM0
- TCL\_Console\$ ZZZ
  - Type ZZZ @ tcl console of Vivado (details are in the following slides)

# Vivado project creation

- Choose part or board
  - We are going to use ZedBoard



# Example IP - Shifter

- Download an example IP repository(lab10\_ip\_repo.tar)

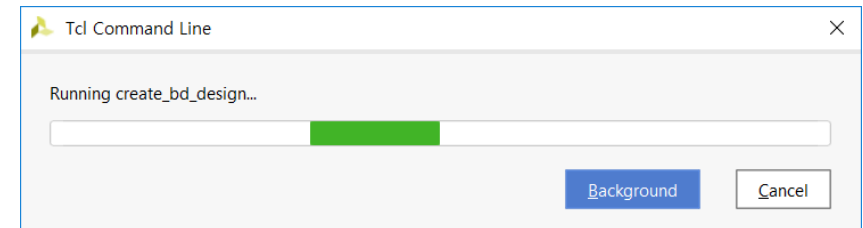
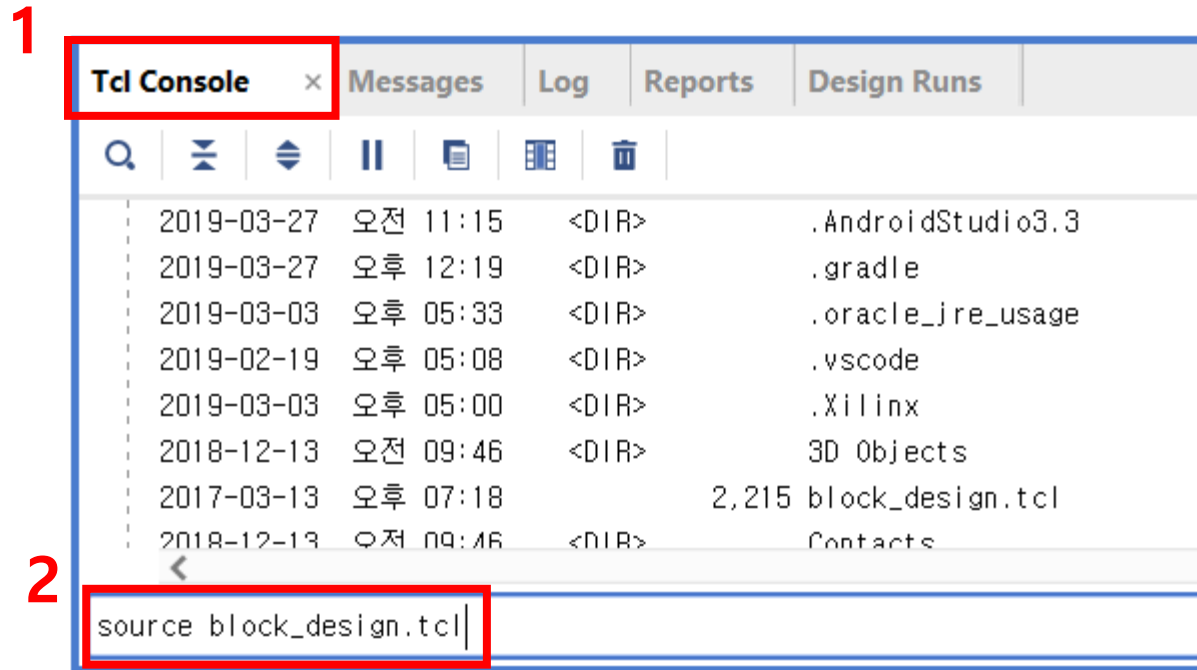
The screenshot illustrates the process of adding an IP repository in the IP Integrator software. The interface is divided into several sections, with numbered red boxes highlighting the steps:

- Flow Navigator:** The 'PROJECT MANAGER' section is expanded, and the 'Settings' option is selected.
- Settings:** The 'IP' category is selected in the left sidebar, and the 'Repository' sub-option is chosen.
- IP Repositories:** The '+' button is clicked to add a new repository.
- IP > Repository:** The 'Add directories to the additional IP to a selected tool-tip will alert you' message is displayed.
- IP Repositories:** The 'Directory' field is set to 'C:\lab10\lab10\_ip\_repo'.
- Add Repository:** The 'OK' button is clicked to confirm the addition of the repository.
- Settings:** The 'OK' button is clicked to confirm the changes in the Settings dialog.

The 'Add Repository' dialog shows a message: '1 repository was added to the project'. The 'Repository' section lists the added repository: 'C:\lab10\lab10\_ip\_repo' with 'IPs (1)' listed below it.

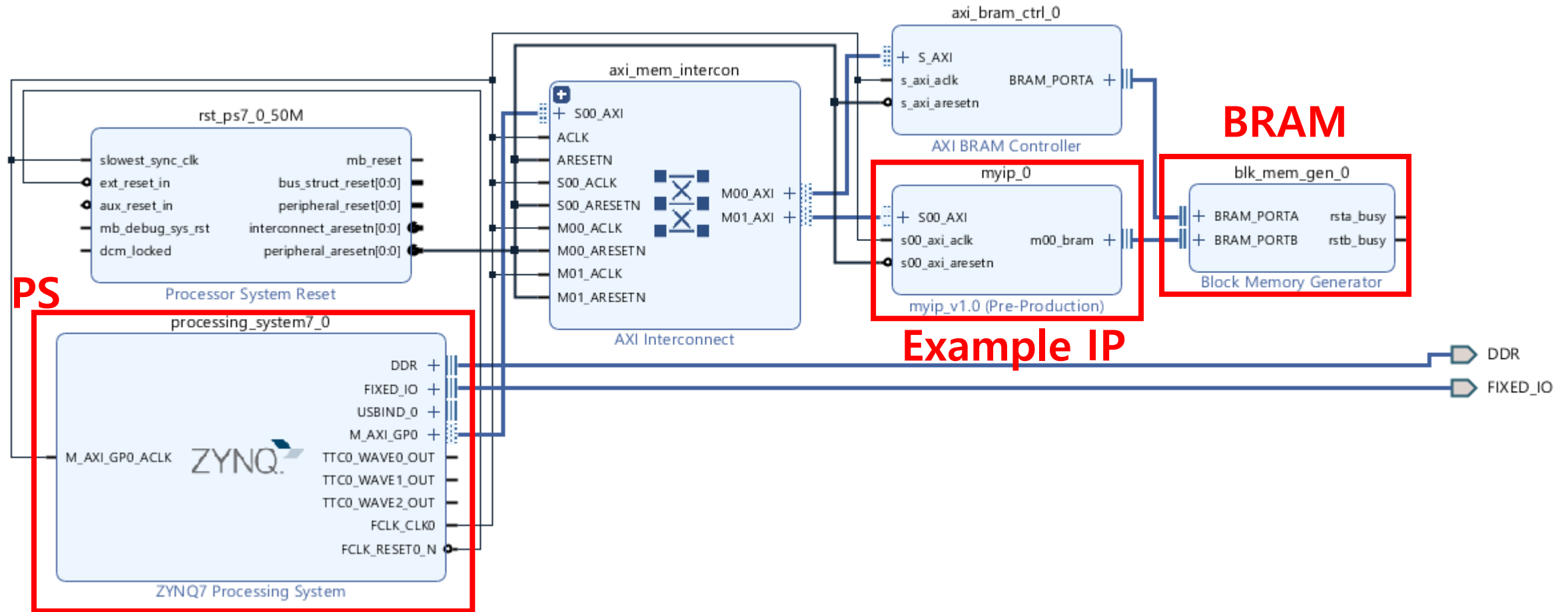
# Example IP - Shifter

- Execute TCL scripts of the example design
  - TCL\_Console\$ cd \${DOWNLOAD\_LAB10\_DIR}
  - TCL\_Console\$ source block\_design.tcl





# Example IP - Shifter



# Example IP - Shifter

- BRAM is @ address 0x4000\_0000 ~ 0x4000\_1FFF
- Shifter is @ address 0x43C0\_0000 ~ 0x43C0\_FFFF

Diagram	Address Editor				
Cell	Slave Interface	Base Name	Offset Address	Range	High Address
processing_system7_0					
Data (32 address bits : 0x40000000 [ 1G ])					
S_AXI	Mem0	0x4000_0000	8K		0x4000_1FFF
S00_AXI	S00_AXI_reg	0x43C0_0000	64K		0x43C0_FFFF

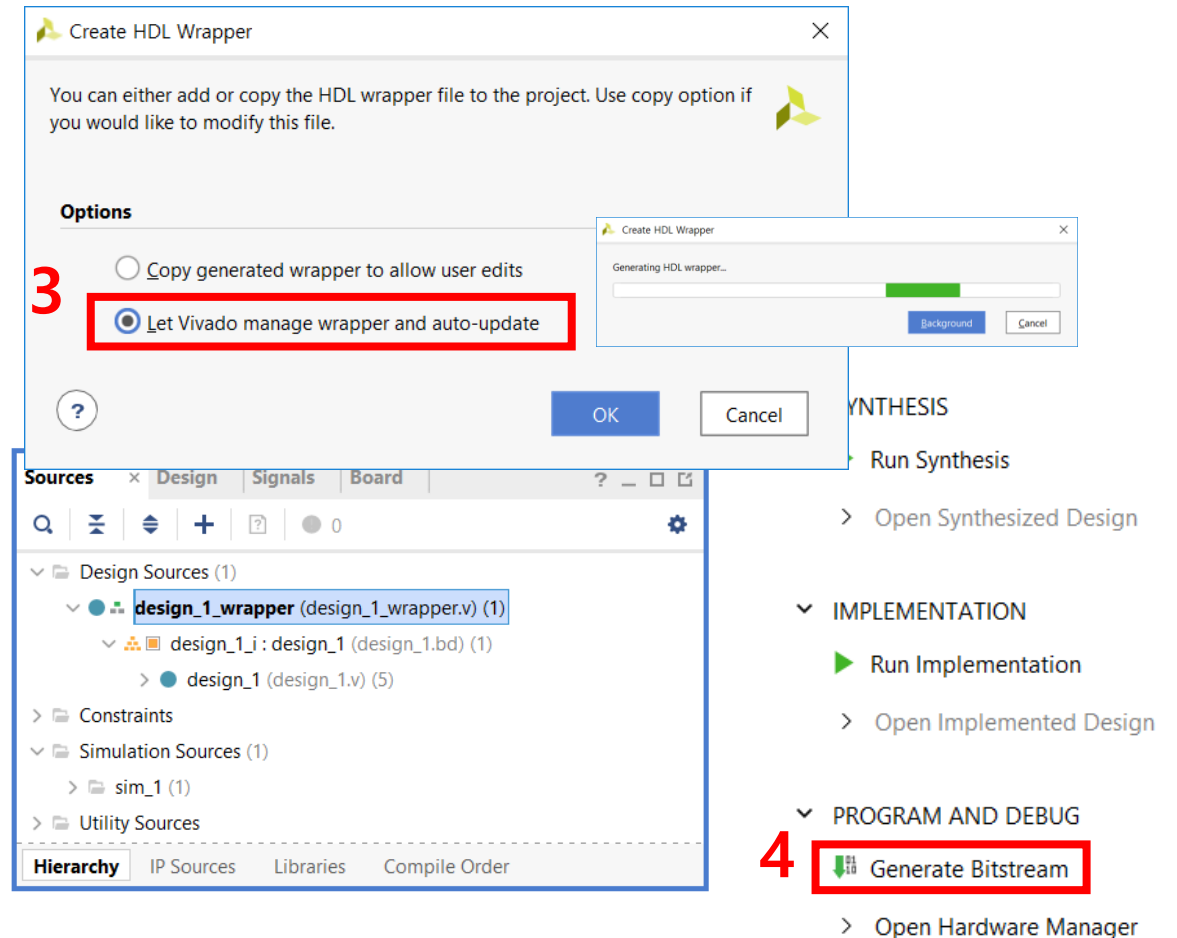
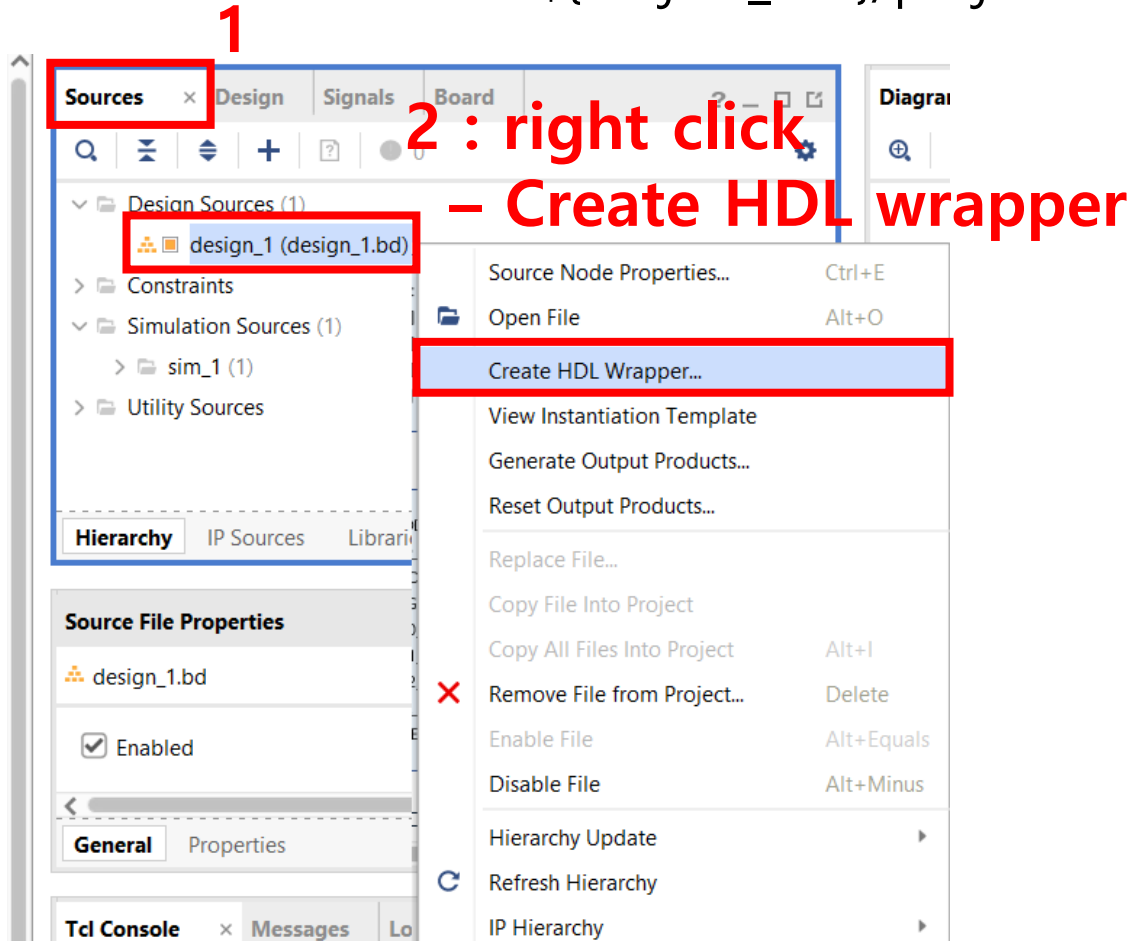
**BRAM ADDR**

**Shifter IP**

```
422 wire magic_code = (slv_reg0 == 32'h5555);
423
424 always @( posedge S_AXI_ACLK )
425 begin
426     if ( S_AXI_ARESETN == 1'b0 )
427         bram_state <= BRAM_IDLE;
428     else
429         case (bram_state)
430             BRAM_IDLE: bram_state <= (magic_code)? BRAM_READ : BRAM_IDLE;
431             BRAM_READ: bram_state <= BRAM_WAIT;
432             BRAM_WAIT: bram_state <= BRAM_WRITE;
433             BRAM_WRITE: bram_state <= (run_complete)? BRAM_IDLE: BRAM_READ;
434             default : bram_state <= BRAM_IDLE;
435         endcase
436     end
```

# Example IP - Shifter

- Generate Bitstream : \${Project\_DIR}/projectname.runs/impl\_1/design\_1\_wrapper.bit



- Micro 5pin cable
- If 뻣뻣(stiff) -> 칼팁(sharp tip) -> 후크(hook)
- No further labs for broken board



# Example IP - Shifter

- Board Power ON
- Open terminal @ HOST
  - Login ID/PW: zed/zedzed
  - HOST\$ minicom -D /dev/ttyACM0
- Run example program
  - BOARD\$ cd \${LAB10\_DIR}
  - BOARD\$ make
  - (before un-plugging power) BOARD\$ sudo poweroff
    - If you do not execute poweroff on terminal and eject the sdcard, your sdcard will not work permanently. (penalty ☺)

addr	FPGA(hex)
0	0
1	2
2	4
3	6
4	0
5	0
6	0
7	0

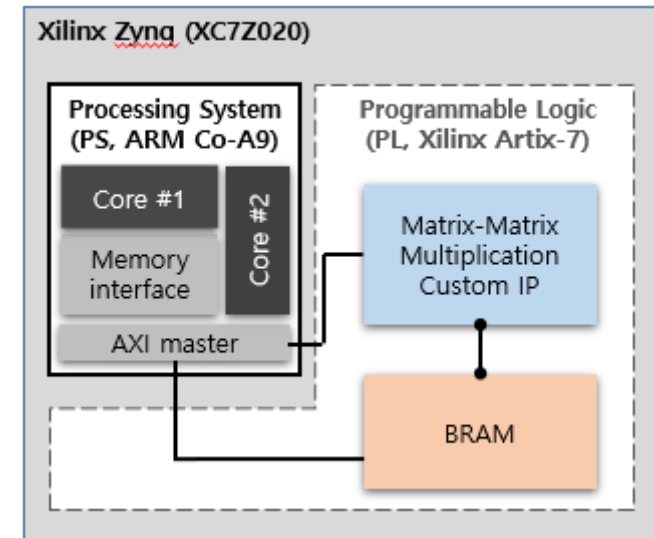
  

addr	FPGA(hex)	
0	0	
1	2	
2	4	
3	6	
4	0	*(0)<<1
5	4	*(1)<<1
6	8	*(2)<<1
7	C	*(3)<<1

# Main Practice

# Practice

- **Follow tutorial & understand the functionality**
  - Run a given example IP in your FPGA linux.
    - Run and understand ip customizing of sample project on report.
  - Our practice
    - Processing System + BRAM + Connectivity + Custom IP (shifter)
  - Our project
    - Note) Term project = PS + BRAM + Connectivity + Custom IP (M\*M multiplier)



# Homework

---

- Follow tutorial & Understand functionality of MyIP
- No Report
- Nothing to Submit for Lab 10
  - There will be no submit page either.

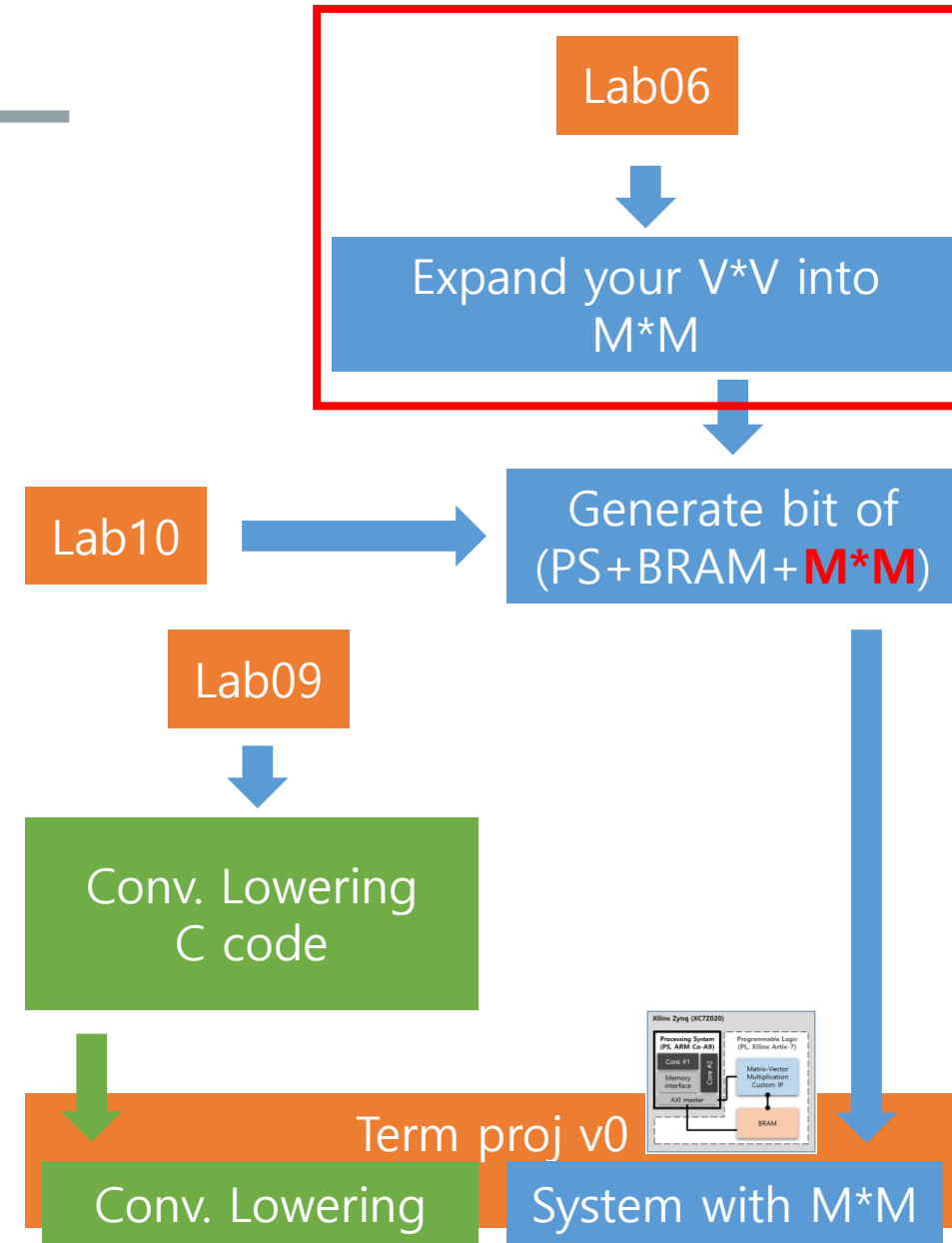


Term Project v0

# Term Project v0 중간제출

## ■ Project Requirement

- Expanding  $V*V$  into  $M*M$  by making PE Array
  - $M*M$  by making PE Array
    - Edit your  $V*V$  code(lab06) on your own
- Demo
  - Waveform



# Term Project v0 중간제출

---

## ■ Requirements

### - Result

- Attach your project folder with all your verilog codes (e.g., M\*M, test bench)
- Attach your M\*M waveform(simulation result) with [student\_number, name]
  - The correct waveform should be shown to confirm the operation of your code.

### - Report

- Explain operation of M\*M with waveform that you implemented
- In your own words
- Either in Korean or in English
- # of pages does not matter
- **PDF only!!**

### - **Result + Report to one .zip file**

## ■ Upload (.zip) file on ETL

### - Submit one (.zip) file

- zip file name : [Term0-mid]name.zip (ex : [Term0-mid]홍길동.zip)

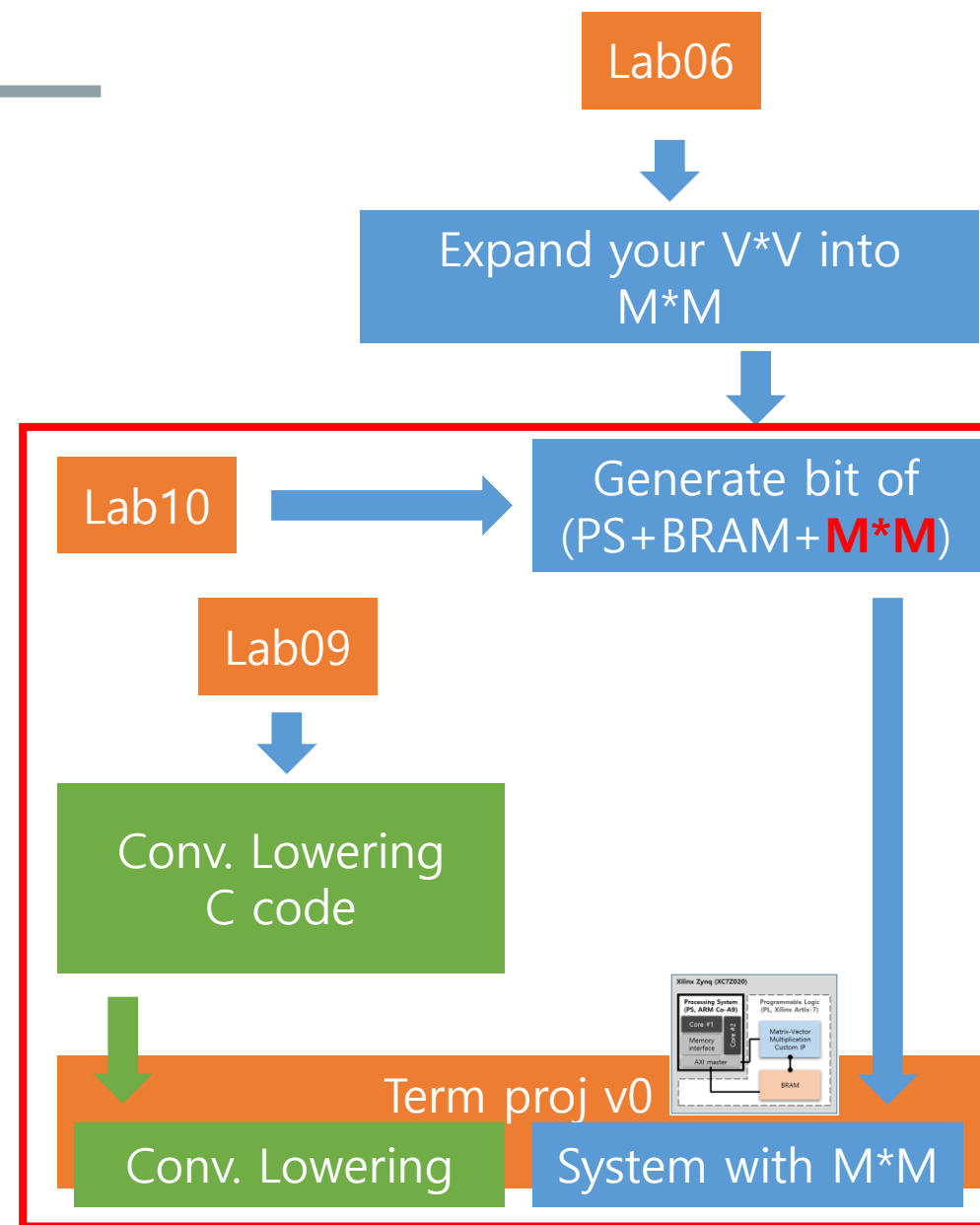
### - Due: 6/2(WED) 23:59

- **No Late Submission**

# Term Project v0 최종제출

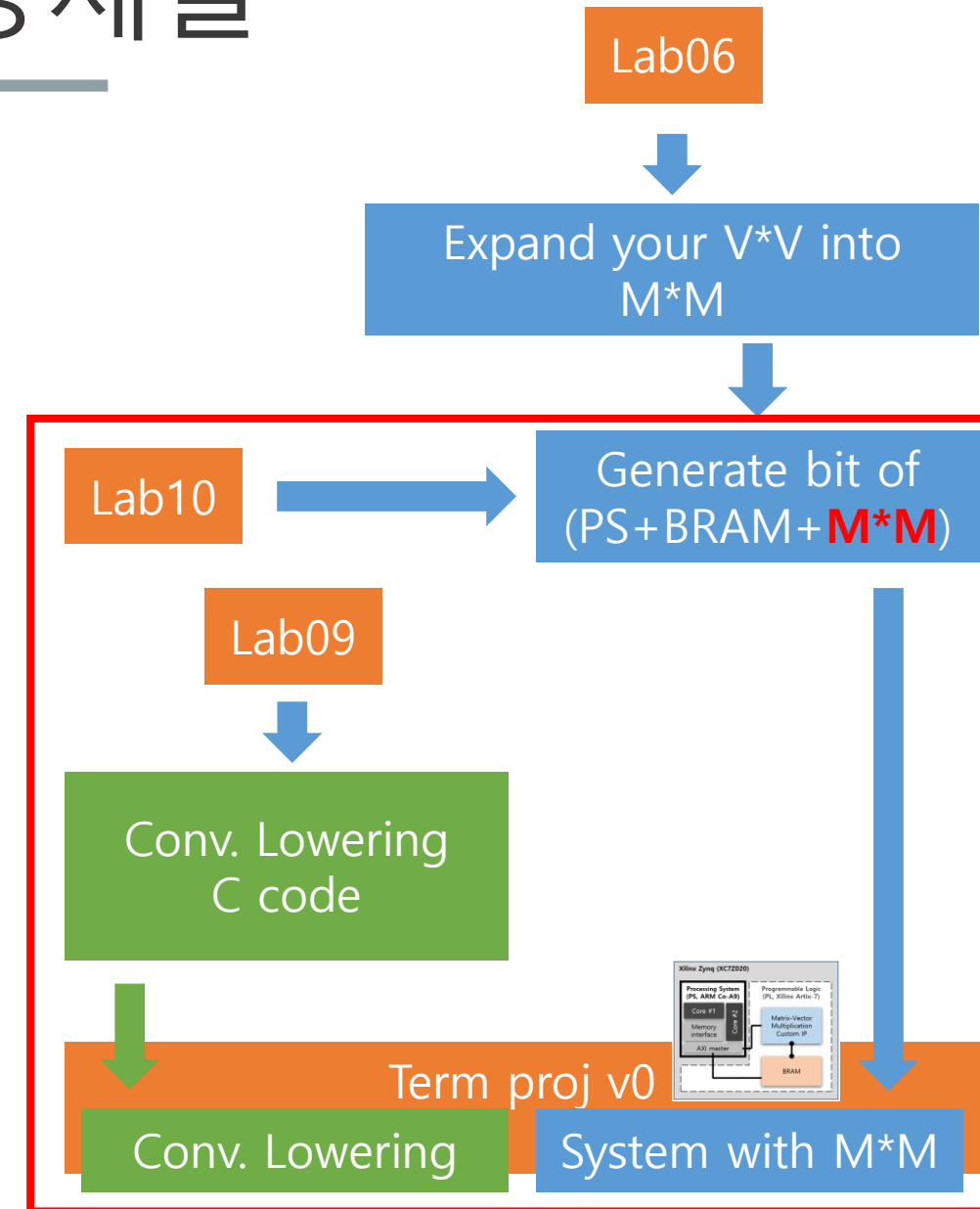
## ■ Project Requirement

- Convolution lowering + Custom IP system with  $M \times M$ 
  - Convolution lowering
    - Refer Lab09
  - Custom IP system with  $M \times M$ 
    - Refer Lab10 tutorial(Custom IP) & appendix(editing the custom IP)
- Demo
  - Zedboard



# Term Project Optimization 최종제출

- Optional Project
  - Convolution lowering + Custom IP system with  $M*M$  + Optimization
    - DMA
    - Zero-Skipping
    - Quantization
    - ...etc
- Demo
- Zedboard



# Term Project v0 + Optimization 최종제출

---

- Due: 6/14(MON) 23:59
  - **Details will be noticed next week**

# Appendix – Editing the Custom IP

# Editing Custom IP

- (1) IP Catalog -> Edit in IP Packager -> IP project

The screenshot displays the Vivado IDE interface with the following components:

- PROJECT MANAGER**
  - Settings
  - Add Sources
  - Language Templates
  - IP Catalog
- IP INTEGRATOR**
  - Create Block Design
  - Open Block Design
  - Generate Block Design
- SIMULATION**
  - Run Simulation
- RTL ANALYSIS**
  - Open Elaborated Design

The **IP Properties** window for **myip\_v1.0** shows:

- Version: 1.0 (Rev. 4)
- Interfaces: AXI4
- Description: My new AXI IP
- Status: Pre-Production

The **IP Catalog** window shows a table of IP cores:

Name	AXI4	Status	License	VLNV
User Repository (c:/.../lab10_ip_repo)				
AXI Peripheral				
myip_v1.0				user.org:user:myip:1.0
Vivado Repository				
Alliance Partners				
Audio Connectivity &				
Automotive & Indust				

A context menu is open over the **myip\_v1.0** entry, with the following options:

- Properties... (Ctrl+E)
- IP Settings...
- Add Repository...
- Refresh All Repositories
- Customize IP...
- Edit in IP Packager** (highlighted by a red arrow)
- Disable IP
- Delete IP (Delete)
- License Status

Red text annotations with arrows indicate the steps:

1. Right click
2. Edit in IP Packager

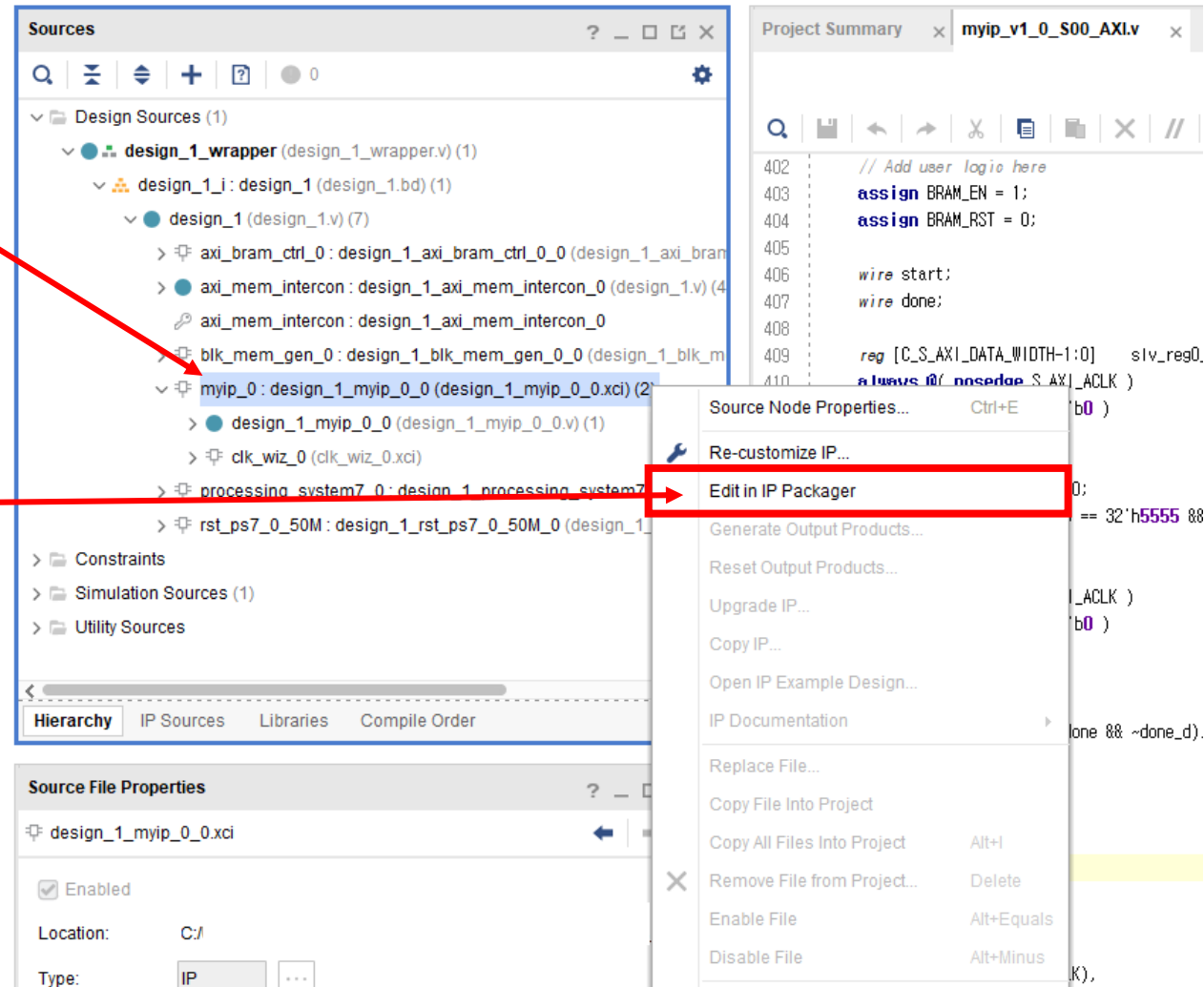


# Editing Custom IP

- (2) On your source list -> Edit in IP Packager -> Packaging Custom IP on IP Project

1. Right click on custom IP

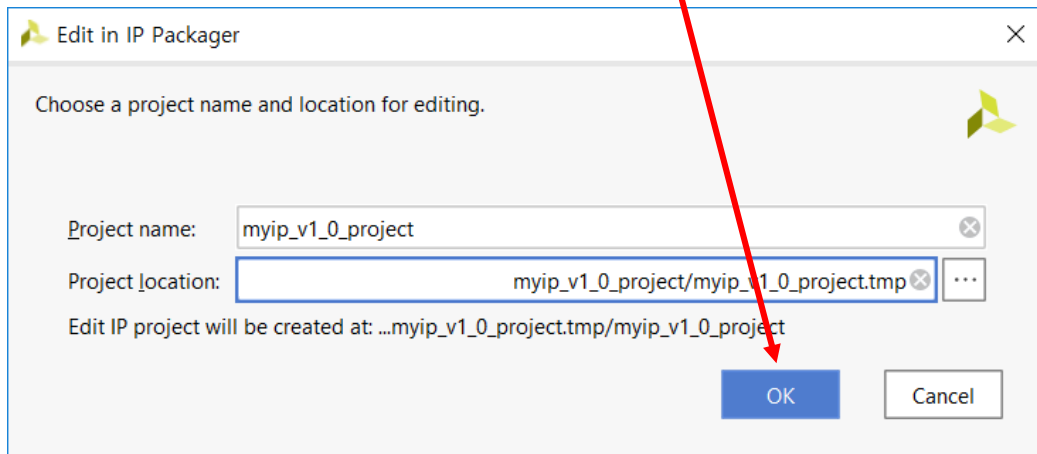
2. Edit in IP Packager



# Packaging Custom IP

## ■ Package IP -> Configurations

1



Edit in IP Packager

Choose a project name and location for editing.

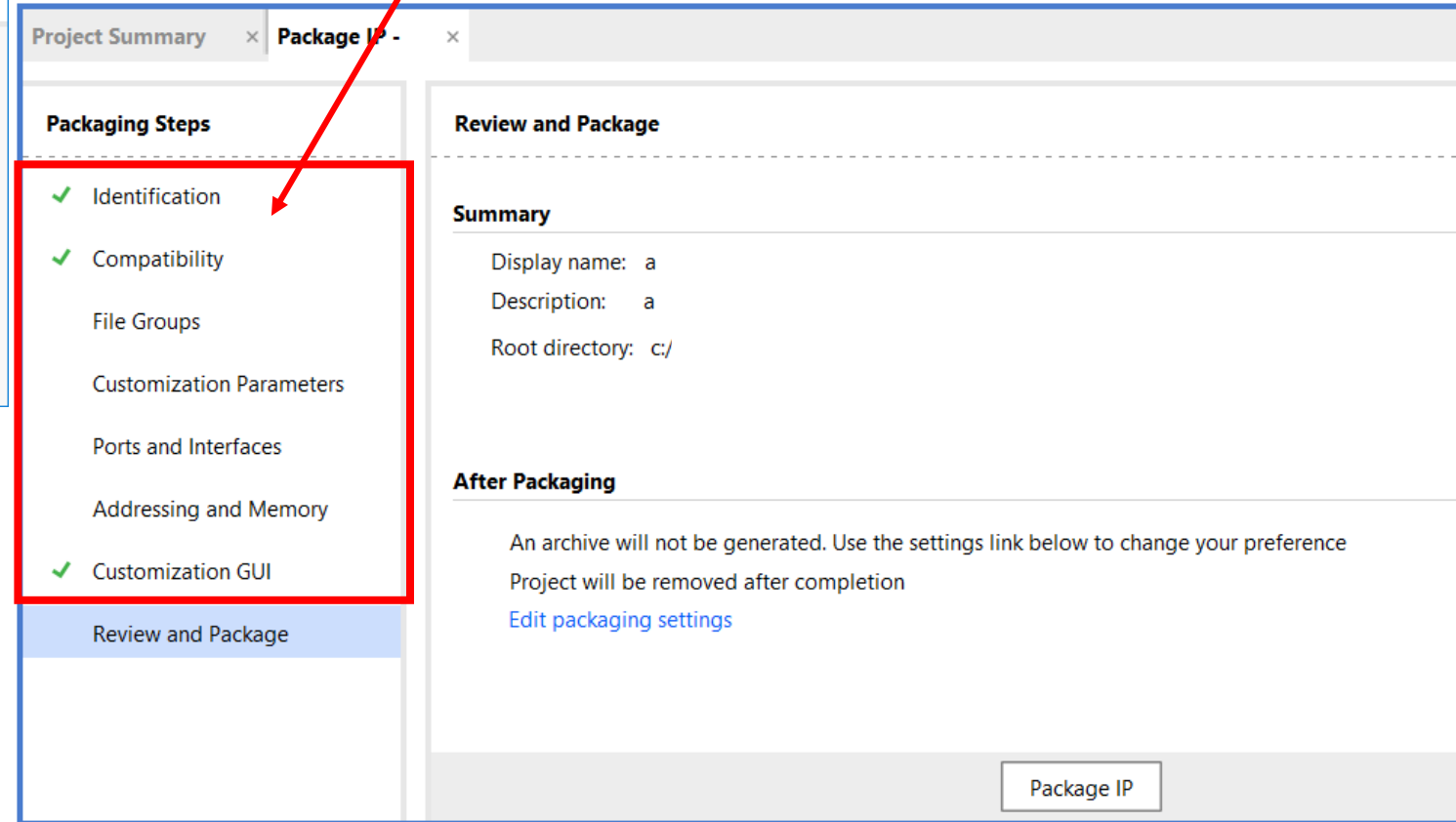
Project name: myip\_v1\_0\_project

Project location: myip\_v1\_0\_project/myip\_v1\_0\_project.tmp

Edit IP project will be created at: ...myip\_v1\_0\_project.tmp/myip\_v1\_0\_project

OK Cancel

## 2. Check IP Configurations



Project Summary x Package IP - x

**Packaging Steps**

- ✓ Identification
- ✓ Compatibility
- File Groups
- Customization Parameters
- Ports and Interfaces
- Addressing and Memory
- ✓ Customization GUI

**Review and Package**

**Summary**

Display name: a

Description: a

Root directory: c:/

**After Packaging**

An archive will not be generated. Use the settings link below to change your preference

Project will be removed after completion

[Edit packaging settings](#)

Package IP

# Checking IP Configurations

- Changing IP files
- File Groups -> Add all Verilog files needed for IP

The screenshot shows the Xilinx Vivado GUI with the 'Package IP' window open. The 'Packaging Steps' list on the left includes Identification, Compatibility, File Groups (highlighted), Customization Parameters, Ports and Interfaces, Addressing and Memory, and Customization GUI. The 'File Groups' table shows a hierarchy: Standard, Advanced, and Verilog Synthesis (5). A red arrow points to the 'Add Files...' option in the context menu for the 'Verilog Synthesis' group. Another red arrow points to the 'Add IP Files (Verilog Simulation)' dialog box, which is open. The dialog box has a table with columns: Index, Name, Library, and Location. It contains two rows: Index 1, Name 'pe\_', Library 'xil\_defaultlib'; and Index 2, Name 'pearray\_', Library 'xil\_defaultlib'. Below the table are checkboxes for 'Scan and add RTL include files into project', 'Copy sources into IP Directory', and 'Add sources from subdirectories'. The 'OK' button is highlighted with a red box and the number '2'.

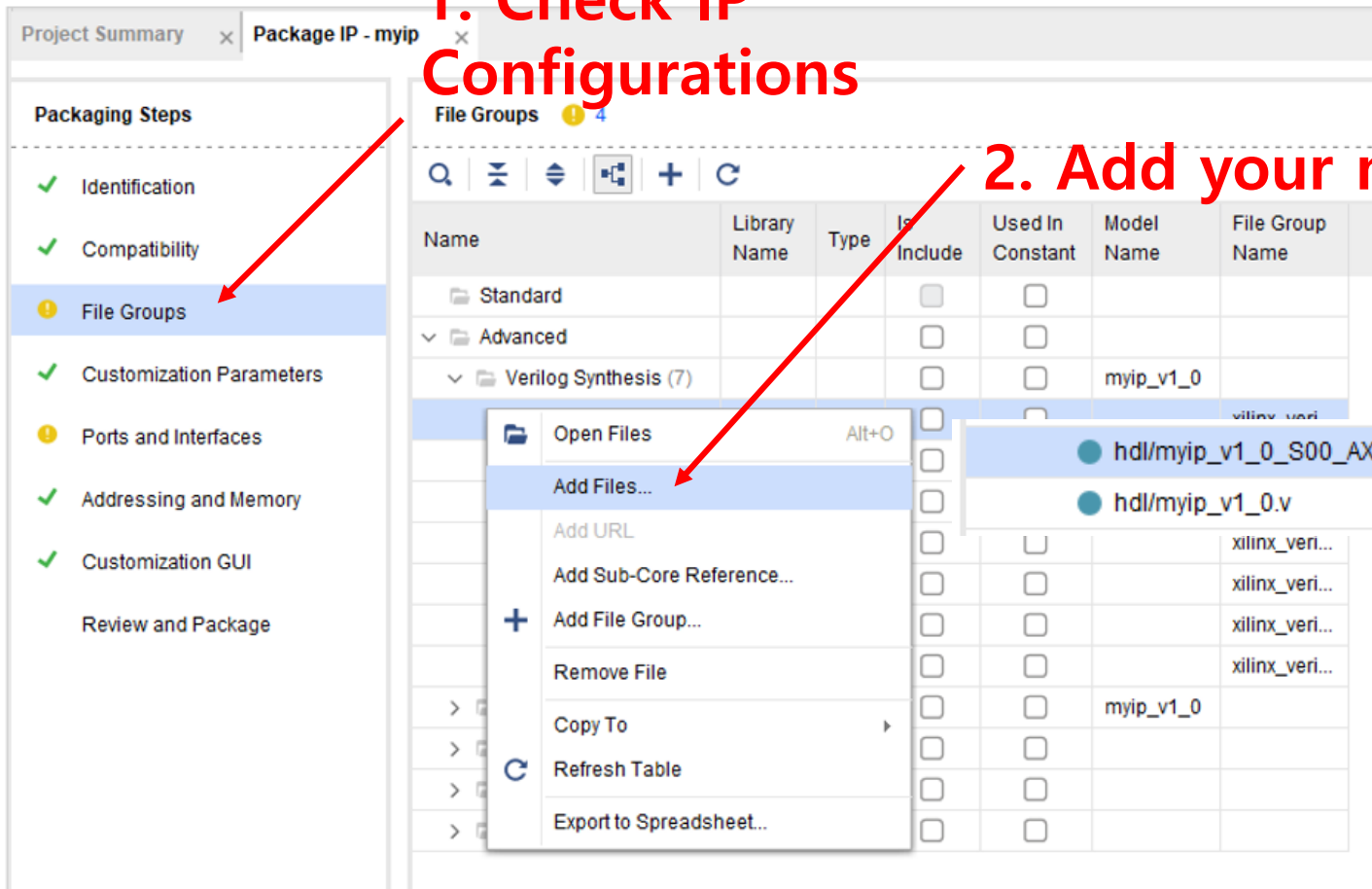
1. Add all Verilog files

2

# Checking IP Configurations

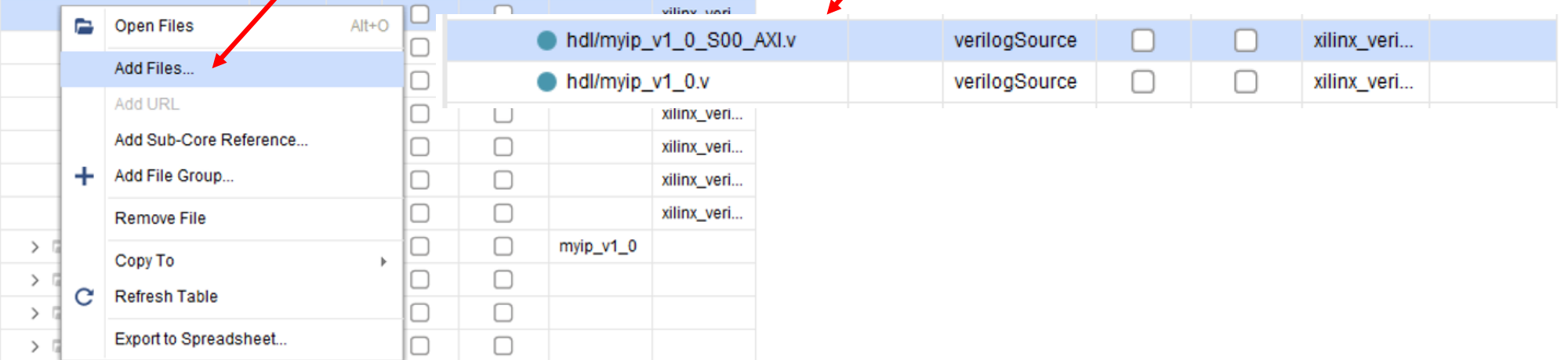
- Changing IP files
- File Groups -> Add your modules -> Edit Verilog files for IP

**1. Check IP Configurations**



**2. Add your modules**

**3. Edit your Verilog files**

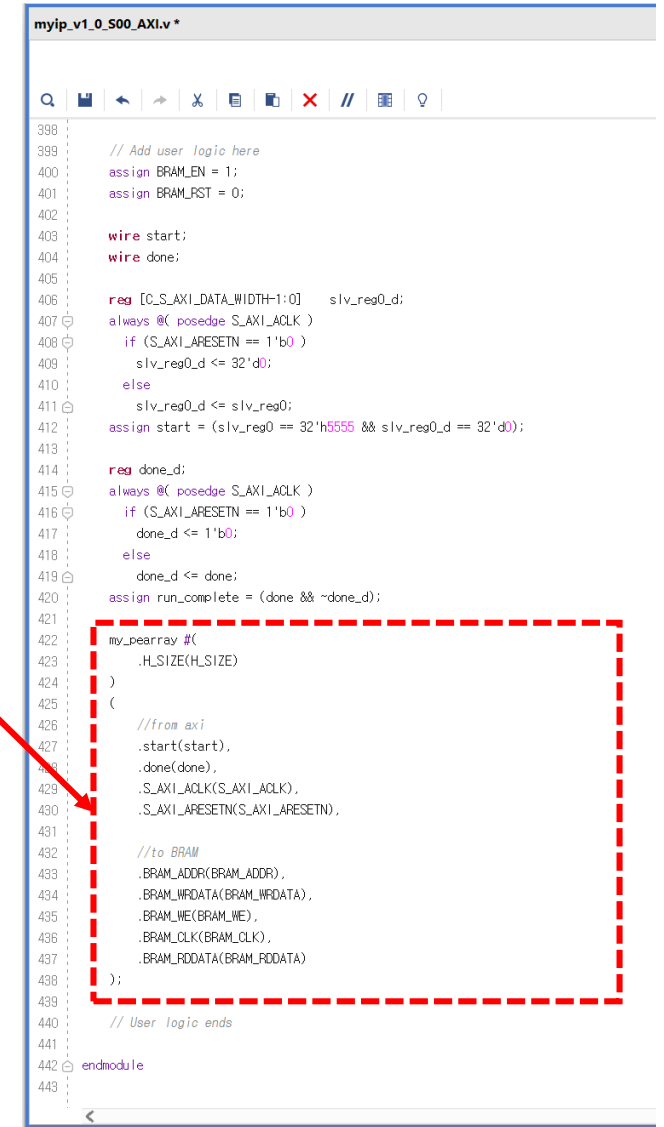


# Checking IP Configurations

- Changing IP files
- File Groups -> Add & Edit Verilog files for IP

**1. Add your modules**  
**My ip, AXI parameter, port modules**

**2. Revise myip\_v1\_0\_S00\_AXI.v & myip\_v1\_0.v codes**  
**- especially ports, parameters, modules you needed**



```
myip_v1_0_S00_AXI.v
398
399 // Add user logic here
400 assign BRAM_LEN = 1;
401 assign BRAM_RST = 0;
402
403 wire start;
404 wire done;
405
406 reg [C_S_AXI_DATA_WIDTH-1:0] slv_reg0_d;
407 always @(posedge S_AXI_ACLK)
408 if (S_AXI_ARESETN == 1'b0)
409   slv_reg0_d <= 32'd0;
410 else
411   slv_reg0_d <= slv_reg0;
412 assign start = (slv_reg0 == 32'h5555 && slv_reg0_d == 32'd0);
413
414 reg done_d;
415 always @(posedge S_AXI_ACLK)
416 if (S_AXI_ARESETN == 1'b0)
417   done_d <= 1'b0;
418 else
419   done_d <= done;
420 assign run_complete = (done && ~done_d);
421
422 my_parray #(
423   .H_SIZE(H_SIZE)
424 )
425 (
426   //from axi
427   .start(start),
428   .done(done),
429   .S_AXI_ACLK(S_AXI_ACLK),
430   .S_AXI_ARESETN(S_AXI_ARESETN),
431
432   //to BRAM
433   .BRAM_ADDR(BRAM_ADDR),
434   .BRAM_WFDATA(BRAM_WFDATA),
435   .BRAM_WIE(BRAM_WIE),
436   .BRAM_CLK(BRAM_CLK),
437   .BRAM_RDDATA(BRAM_RDDATA)
438 );
439
440 // User logic ends
441
442 endmodule
443
```

# Add clocking wizard IP

- Step2: File Groups (Add clocking wizard)

The screenshot shows the 'Customize IP' window for the 'Clocking Wizard (6.0)'. The window is divided into two main sections: 'IP Symbol' on the left and configuration options on the right.

**IP Symbol:** The 'Resource' tab is selected. It shows a list of ports and signals. On the left, there are inputs: `s_axi_lite`, `CLK_IN1_D`, `CLK_IN2_D`, `CLKFB_IN_D`, `s_axi_adk`, `s_axi_aresetn`, `reset`, `resetn`, `ref_clk`, `user_clk0`, `user_clk1`, `user_clk2`, `user_clk3`, and `clk_in1`. On the right, there are outputs: `CLKFB_OUT_D`, `clk_stop[3:0]`, `clk_glitch[3:0]`, `interrupt`, `clk_oor[3:0]`, `clk_out1`, and `locked`.

**Configuration Options:** The 'Clocking Options' tab is selected. The 'Component Name' is `clk_wiz_0`.

- Clock Monitor:** ☐ Enable Clock Monitoring
- Primitive:** ☒ MMCM ☐ PLL
- Clocking Features:**
  - ☒ Frequency Synthesis
  - ☐ Minimize Power
  - ☒ Phase Alignment
  - ☐ Spread Spectrum
  - ☐ Dynamic Reconfig
  - ☐ Dynamic Phase Shift
  - ☐ Safe Clock Startup
- Jitter Optimization:**
  - ☒ Balanced
  - ☐ Minimize Output Jitter
  - ☐ Maximize Input Jitter filtering
- Dynamic Reconfig Interface:** ☒ AXI4Lite ☐ DRP. ☐ Phase Duty Cycle Config. ☐ Write DRP registers.
- Input Clock Information:**

	Input Clock	Port Name	Input Frequency(MHz)		Jitter Options	Input Jitter	Source
<input checked="" type="checkbox"/>	Primary	clk_in1	50	10.000 - 800.000	UI	0.010	Single ended clock capable pin
<input type="checkbox"/>	Secondary	clk_in2	100.000	30.000 - 60.000		0.010	Single ended clock capable pin

Buttons: OK, Cancel

# Add clocking wizard IP

## ■ Step2: File Groups (Add clocking wizard)

Re-customize IP

**Clocking Wizard (6.0)**

Documentation IP Location Switch to Defaults

Component Name: clk\_wiz\_0

Board Cloning Options **Output Clocks** Port Renaming MMCM Settings Summary

The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)		Drives
		Requested	Actual	Requested	Actual	Requested	Actual	
<input checked="" type="checkbox"/> clk_out1	clk_out1	50	50.000	0.000	0.000	50.000	50.0	BUFG
<input type="checkbox"/> clk_out2	clk_out2	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out3	clk_out3	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out4	clk_out4	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out5	clk_out5	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out6	clk_out6	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out7	clk_out7	100.000	N/A	0.000	N/A	50.000	N/A	BUFG

☐ USE CLOCK SEQUENCING

**Clocking Feedback**

Output Clock	Sequence Number	Source	Signaling
clk_out1	1	<input checked="" type="radio"/> Automatic Control On-Chip	<input checked="" type="radio"/> Single-ended
clk_out2	1	<input type="radio"/> Automatic Control Off-Chip	<input type="radio"/> Differential
clk_out3	1	<input type="radio"/> User-Controlled On-Chip	
clk_out4	1	<input type="radio"/> User-Controlled Off-Chip	
clk_out5	1		
clk_out6	1		
clk_out7	1		

**Enable Optional Inputs / Outputs for MMCM/PLL**

☐ reset ☐ power\_down ☐ input\_clk\_stopped ☒ Active High ☐ Active Low

☐ locked ☐ clkfbstopped

OK Cancel

# Add clocking wizard IP

- Step2: File Groups (Add clocking wizard)
  - Add clk\_wiz between your PE clk on your PE or PE array & BRAM clk on myip\_v1\_0\_S00\_AXI.v

*// Users to add ports here*

output wire [31:0] BRAM\_ADDR,

output wire BRAM\_CLK,

output wire [31:0] BRAM\_WRDATA,

input wire [31:0] BRAM\_RDDATA,

output wire [3:0] BRAM\_WE,

output wire BRAM\_EN,

output wire BRAM\_RST,

clk\_wiz\_0 u\_clk (.clk\_out1(BRAM\_CLK), .clk\_in1(ac1k));

my\_pe #(L\_RAM\_SIZE(L\_RAM\_SIZE)

) u\_pe (

.ac1k(ac1k),

.aresetn(aresetn),

.ain(ain),

.din(din),

.addr(addr),

.we(we[i]),

.valid(valid),

.dvalid(dvalid[i]),

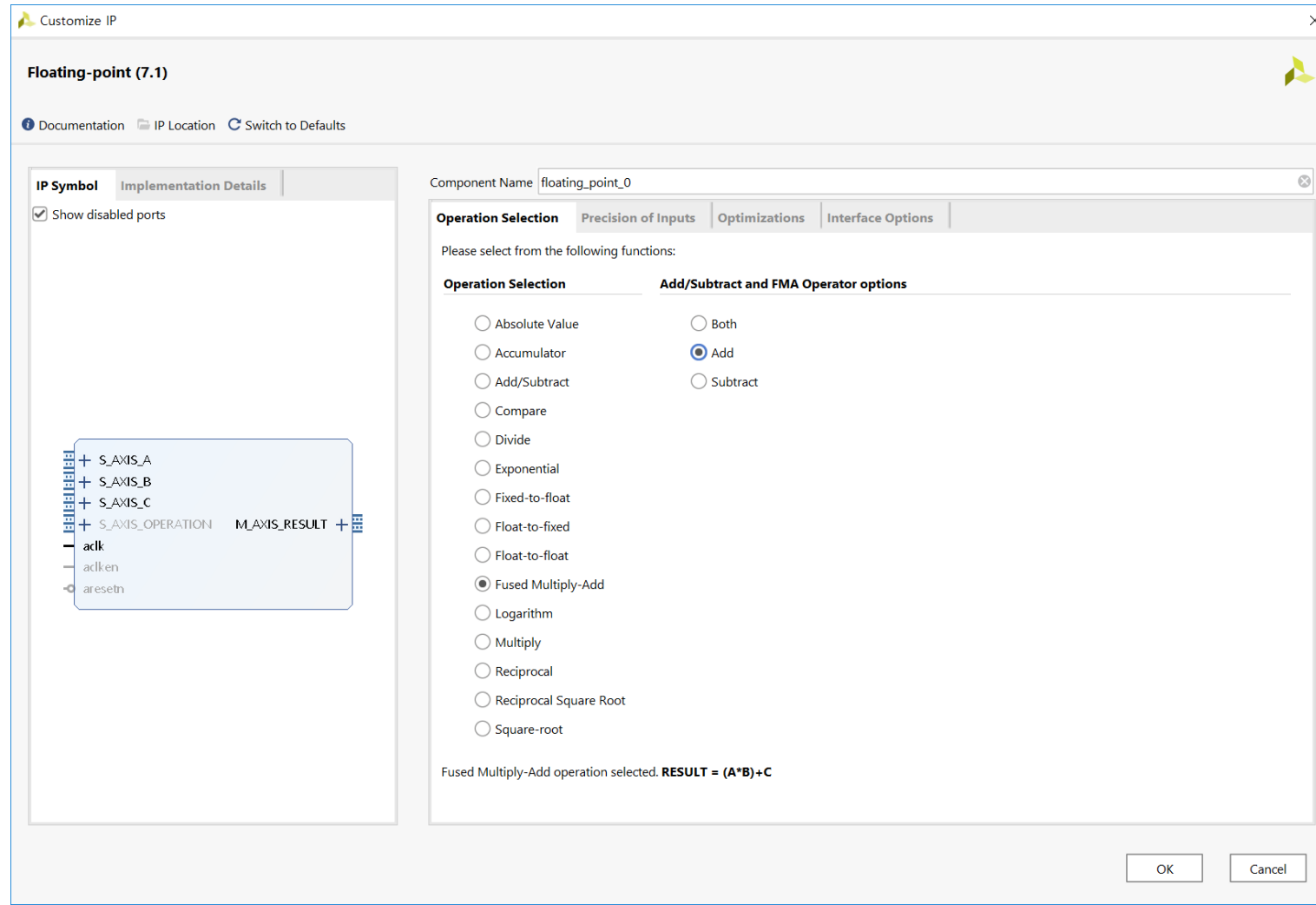
.dout(dout[i])

);



# Add floating point IP

## ■ Step2: File Groups (Add floating point IP)



# Checking IP Configurations

- Changing IP files
- File Groups -> Merge changes

Project Summary

Package IP - myip

Packaging Steps

Identification

Compatibility

File Groups

Customization Parameters

Ports and Interfaces

Addressing and Memory

Customization GUI

Review and Package

File Groups 2

Merge changes from File Groups Wizard

2 warnings 1 info message

Search

Filter

Sort

Layout

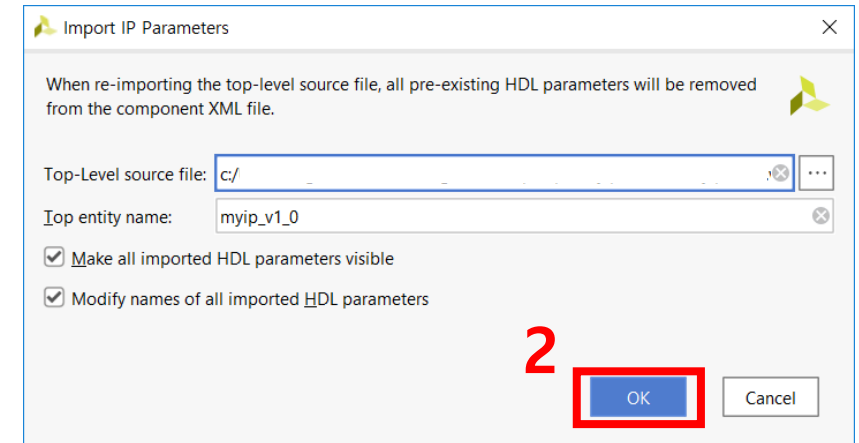
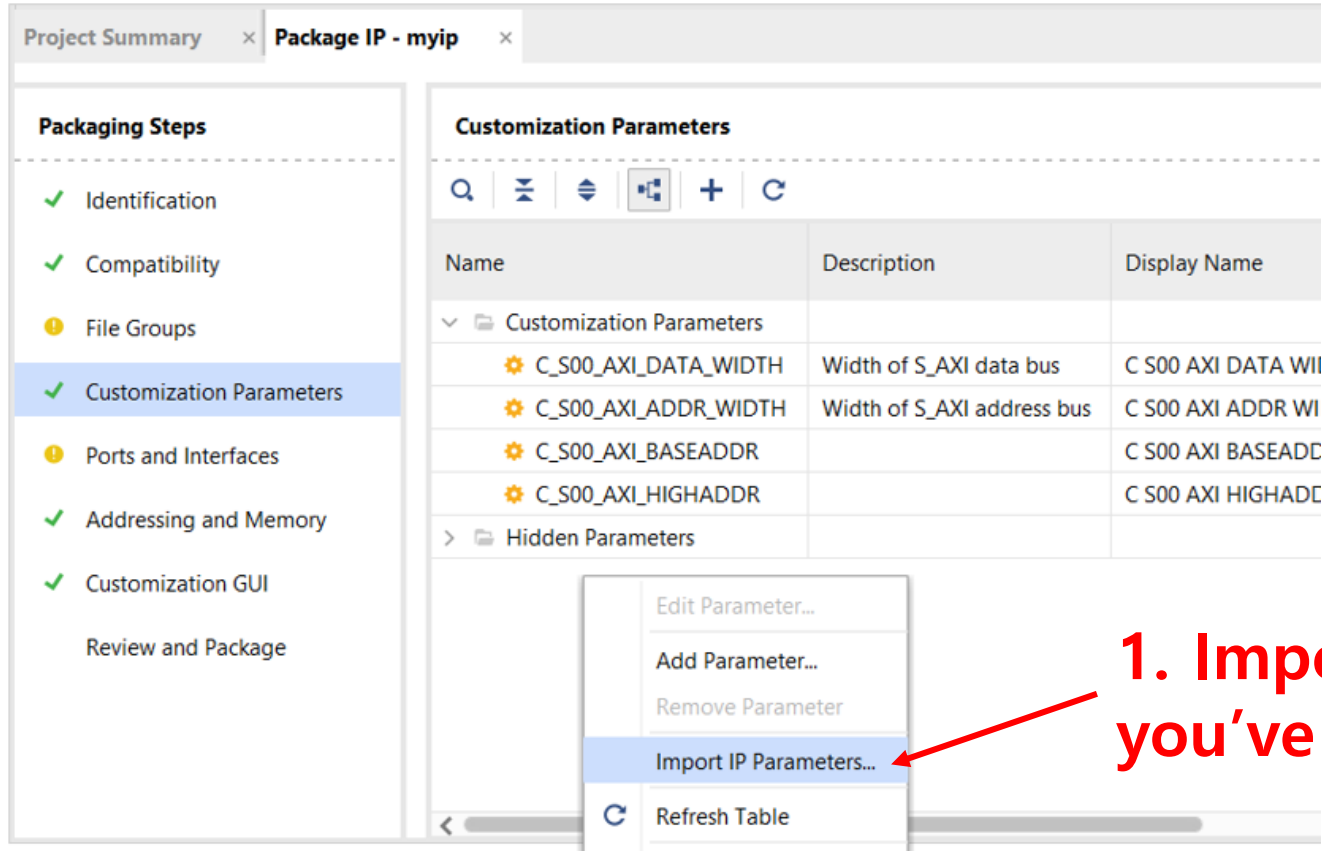
+

Refresh

Name	Library Name	Type	Is Include	Used In Constant	File Group Name	Model Name
Standard			<input type="checkbox"/>	<input type="checkbox"/>		
Advanced			<input type="checkbox"/>	<input type="checkbox"/>		
Verilog Synthesis (5)			<input type="checkbox"/>	<input type="checkbox"/>		myip_v1_0
Verilog Simulation (3)			<input type="checkbox"/>	<input type="checkbox"/>		myip_v1_0
Software Driver (6)			<input type="checkbox"/>	<input type="checkbox"/>		
UI Layout (1)			<input type="checkbox"/>	<input type="checkbox"/>		
Block Diagram (1)			<input type="checkbox"/>	<input type="checkbox"/>		

# Checking IP Configurations

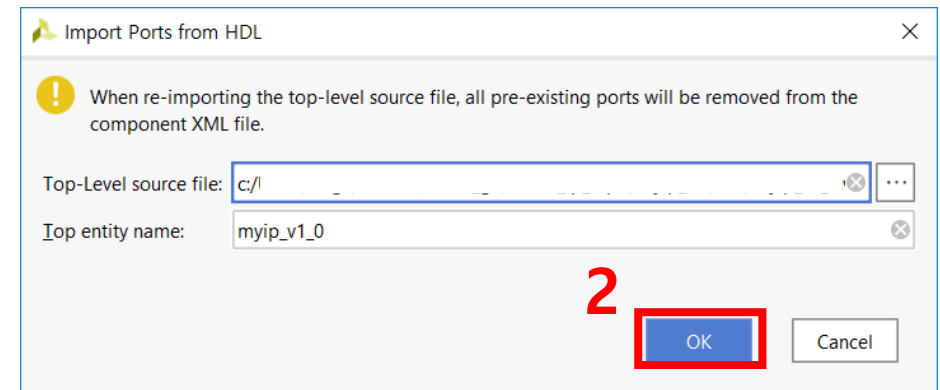
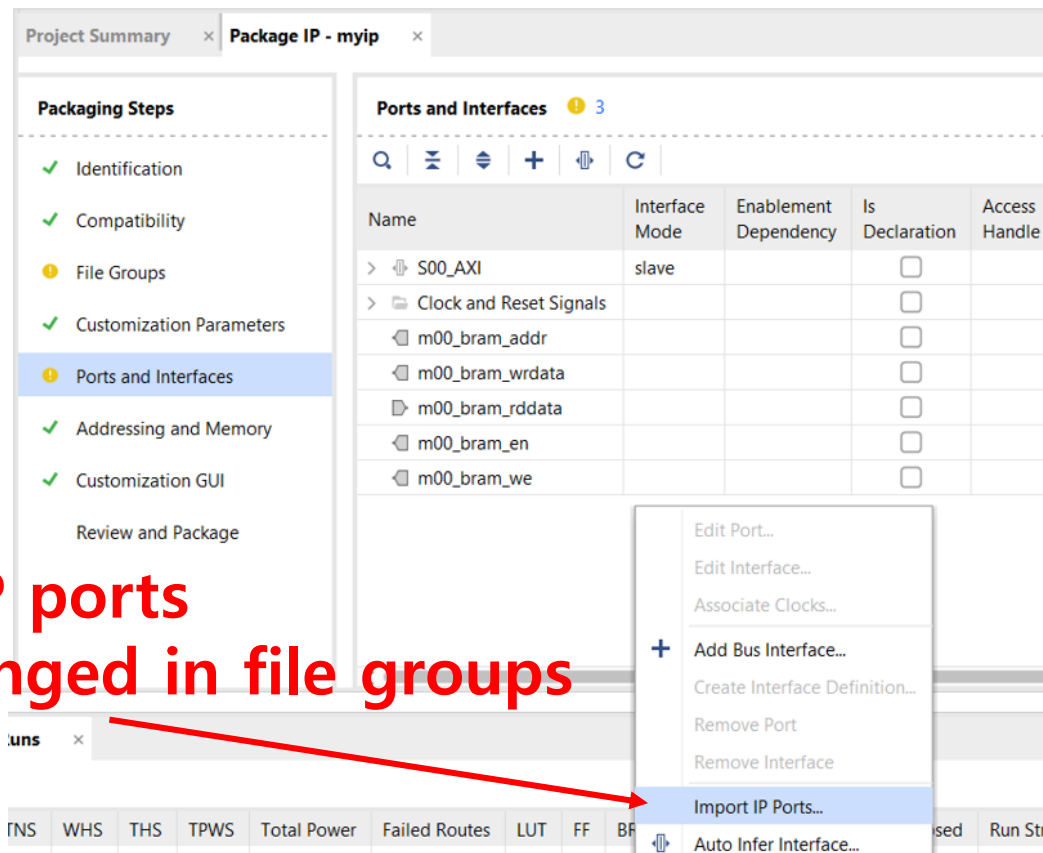
- Parameter configuration
- Customization Parameters -> import IP parameters



**1. Import IP parameters  
you've changed in file groups**

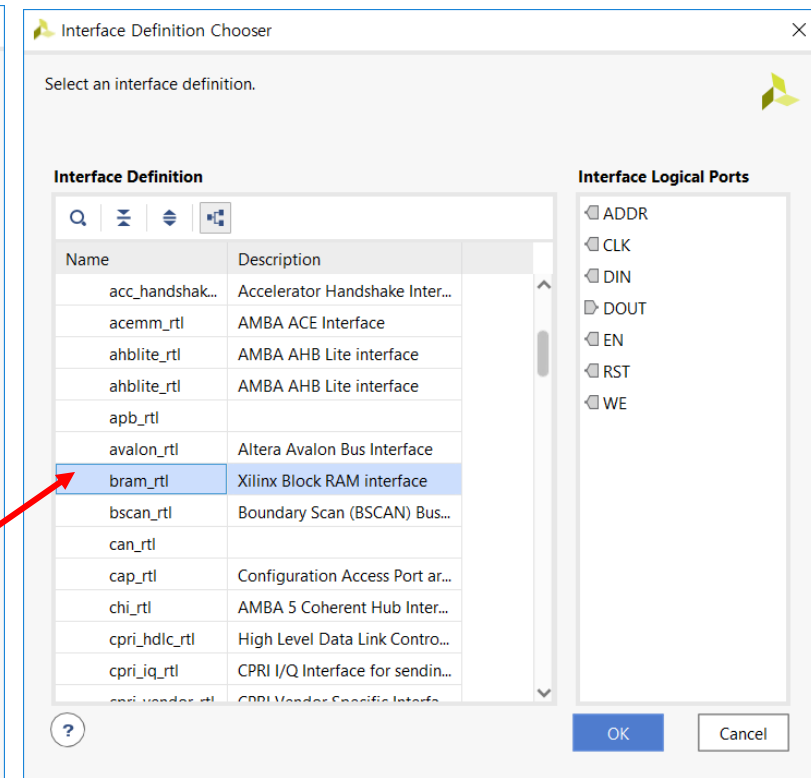
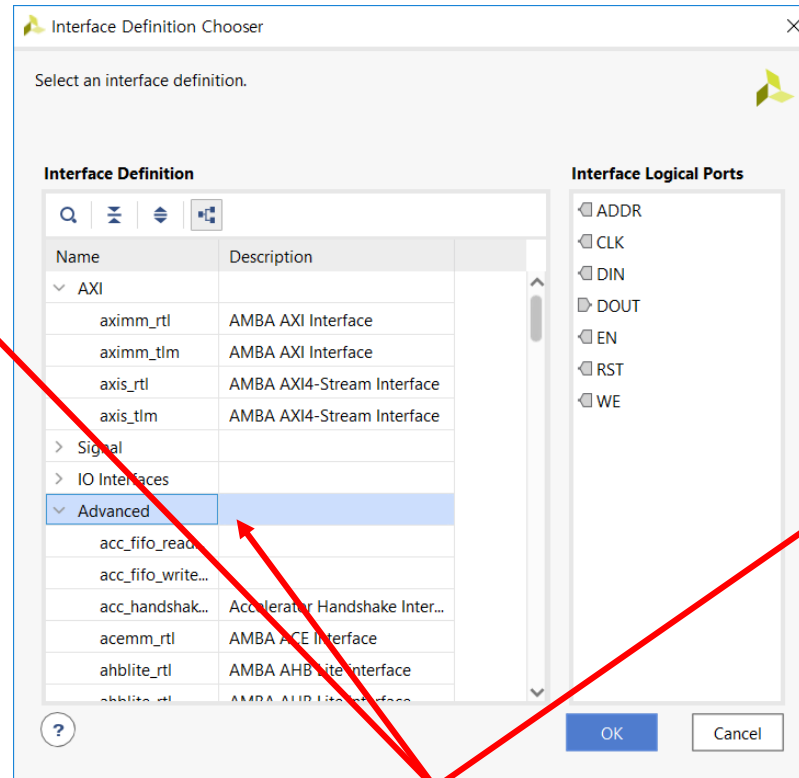
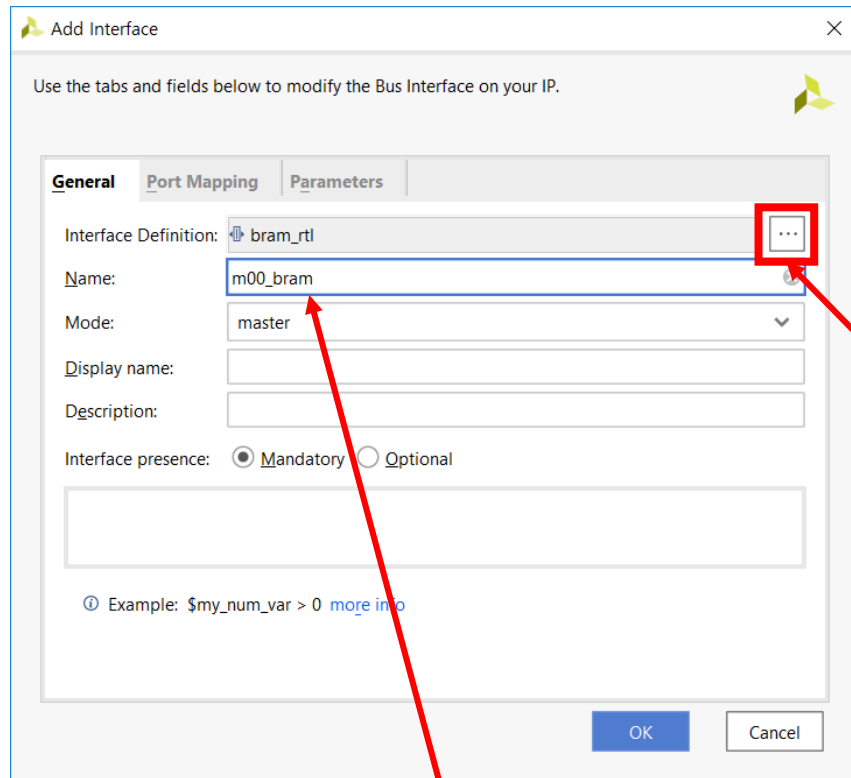
# Checking IP Configurations

- Port configuration
- Ports and Interfaces -> Import IP ports



# Checking IP Configurations

- Port configuration
- Grouping ports - case1 : when the group doesn't exist
- ex) m00\_bram



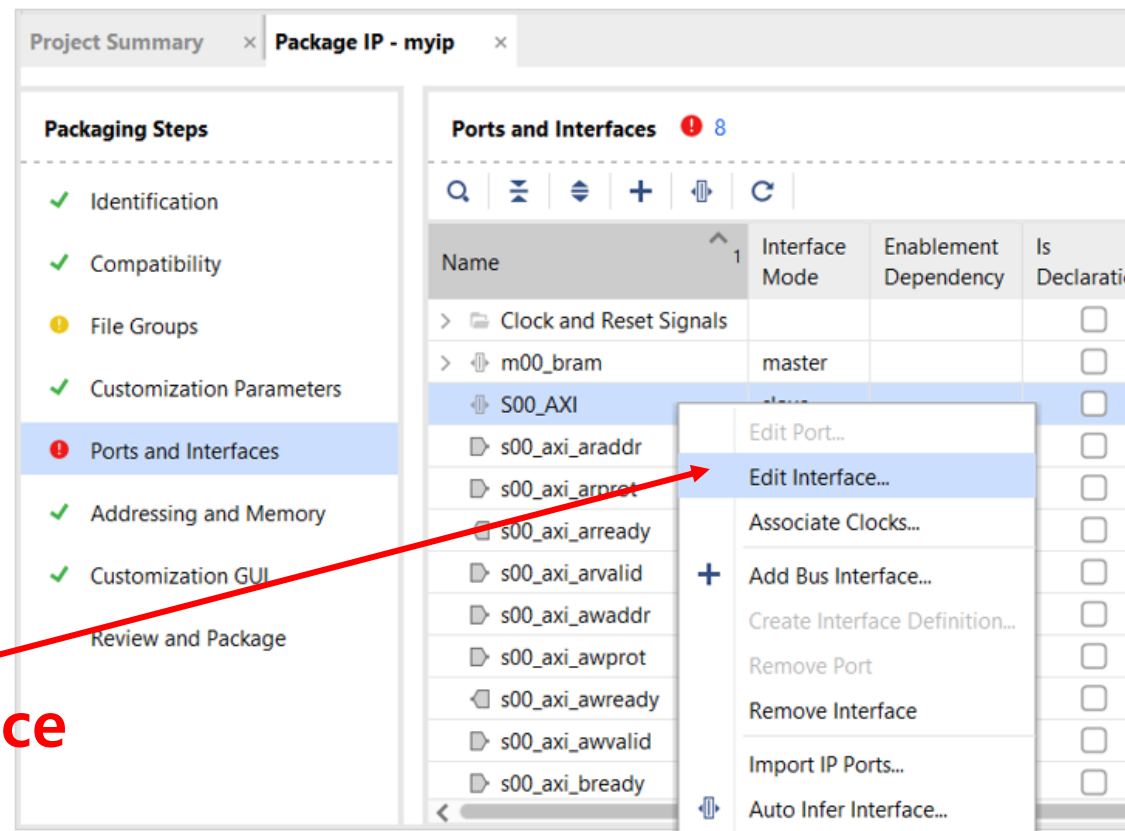
1. Set your port name

2. Advanced – bram\_rtl

# Checking IP Configurations

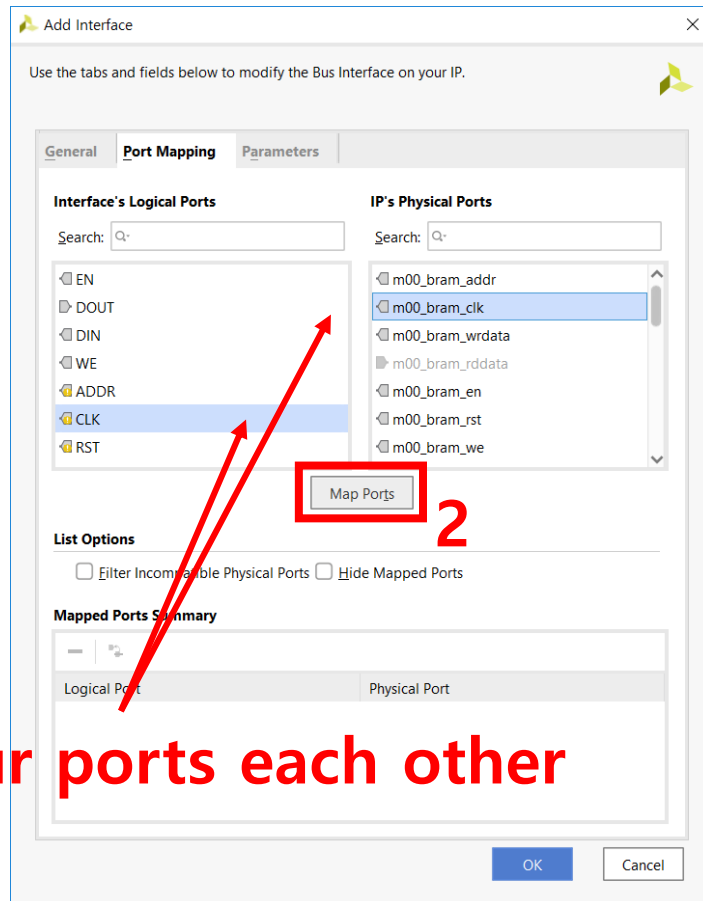
- Port configuration
- Grouping ports – case2 : when the group exists
- ex) S00\_AXI

1. Edit Interface



# Checking IP Configurations

- Port configuration
- Grouping ports – case1 & case2 both common



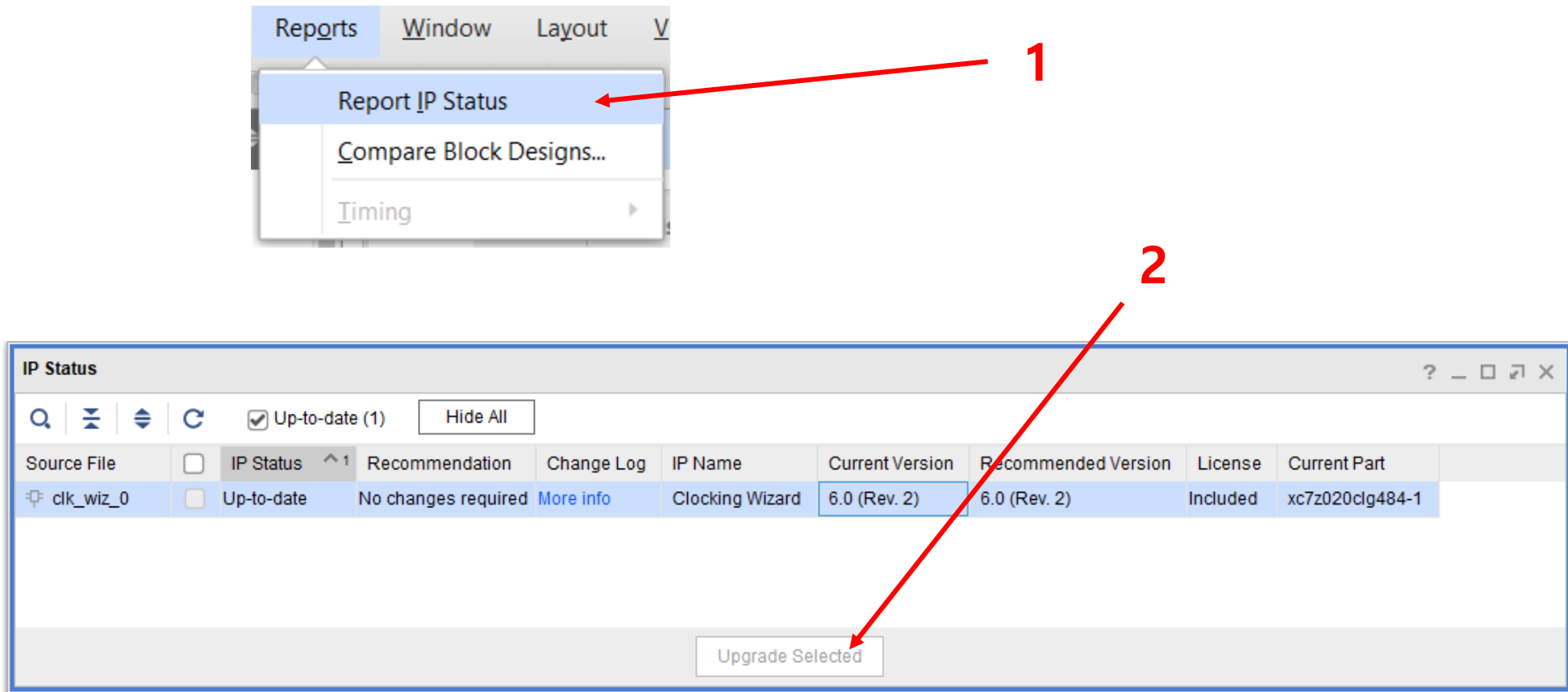
## Mapped Ports Summary

Logical Port	Physical Port
CLK	m00_bram_clk
RST	m00_bram_rst
ADDR	m00_bram_addr
WE	m00_bram_we
EN	m00_bram_en
DOUT	m00_bram_rddata
DIN	m00_bram_wrdata

3. Check your port mapping

# Upgrading Custom IP

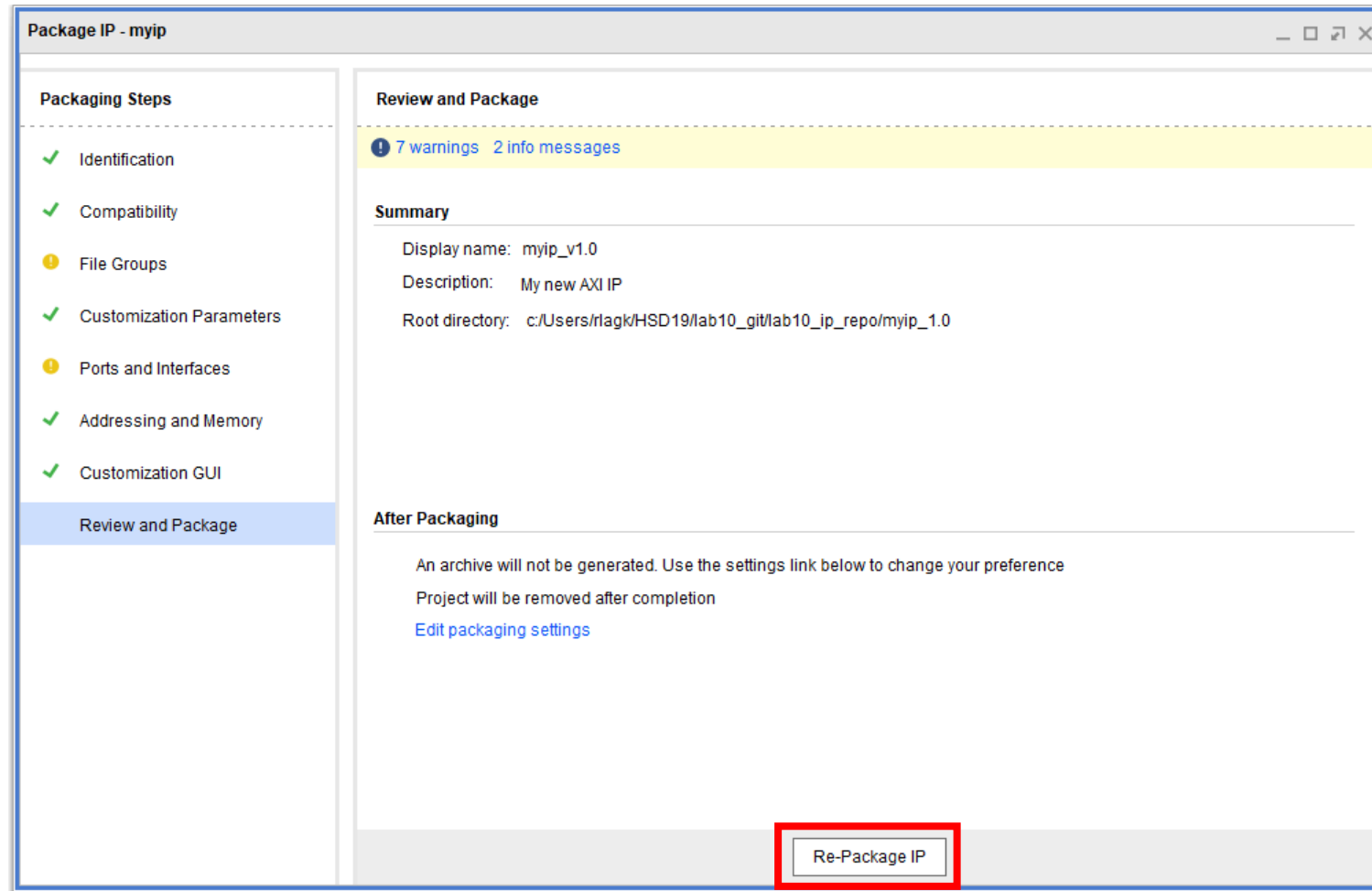
- Execute update for existing Ips, if your project needed.





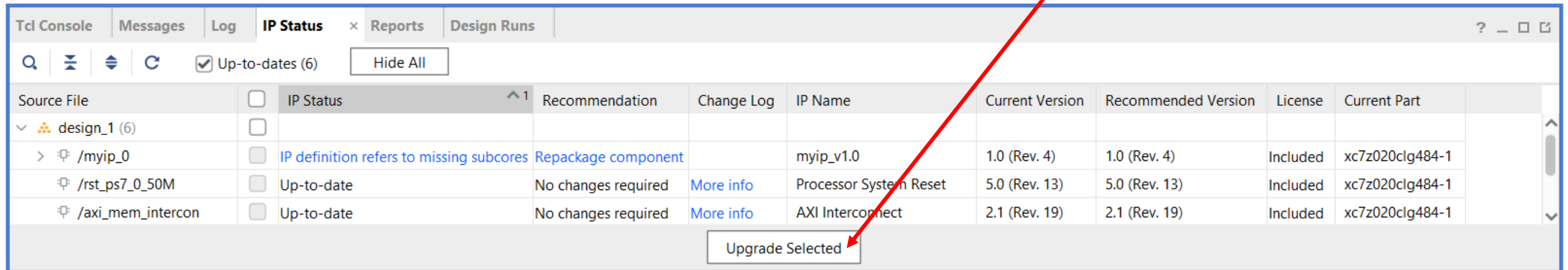
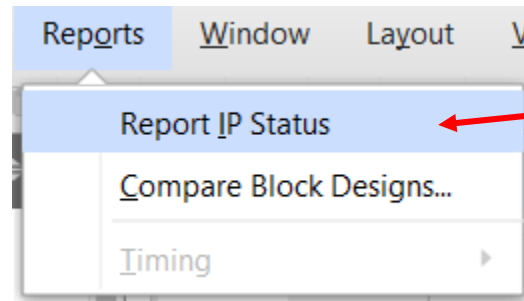
# Checking IP Configurations

## ■ Re-Package IP



# Upgrading Custom IP

- Report IP Status -> Upgrade Selected
- You always have to update IP Status when you revise your IPs.

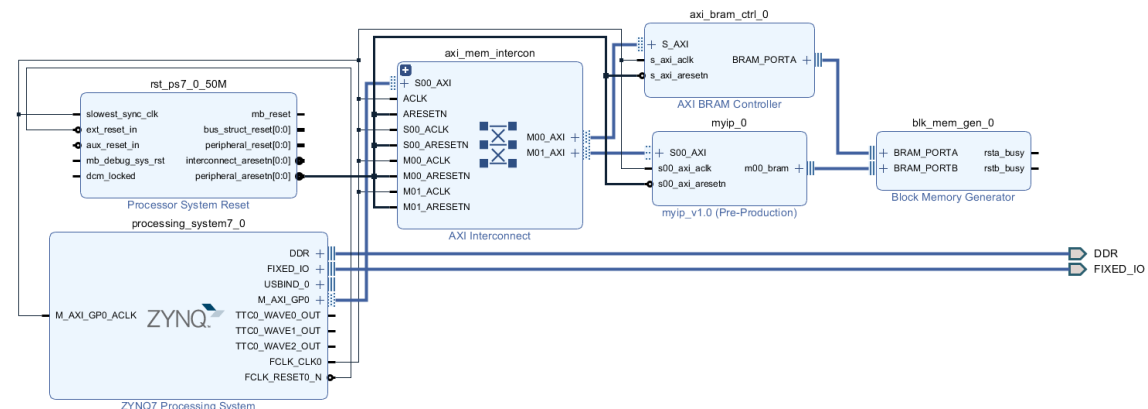
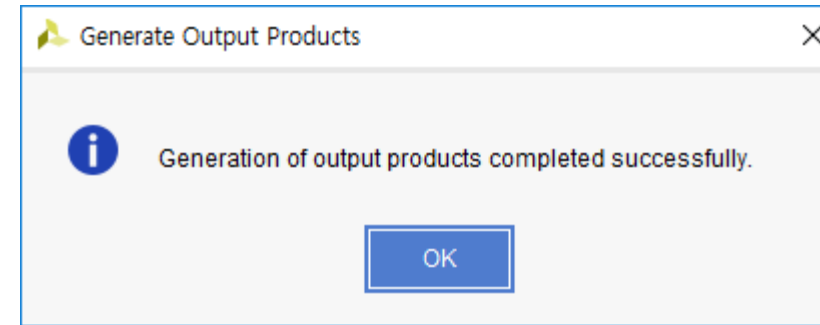
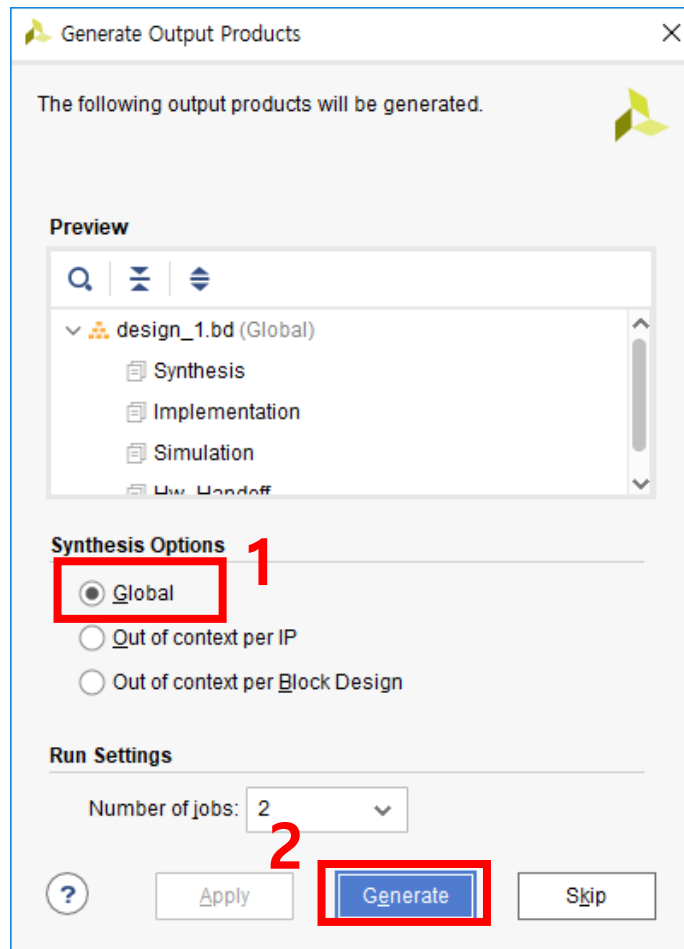


A screenshot of the 'IP Status' report window. The window has tabs for 'Tcl Console', 'Messages', 'Log', 'IP Status', 'Reports', and 'Design Runs'. The 'IP Status' tab is active. Below the tabs is a search bar and a 'Hide All' button. The main area contains a table with columns: Source File, IP Status, Recommendation, Change Log, IP Name, Current Version, Recommended Version, License, and Current Part. The table lists three IP components: /myip\_0, /rst\_ps7\_0\_50M, and /axi\_mem\_intercon. The /myip\_0 component has a status of 'IP definition refers to missing subcores' and a recommendation of 'Repackage component'. A red arrow labeled '2' points to the 'Upgrade Selected' button at the bottom right of the window.

Source File	IP Status	Recommendation	Change Log	IP Name	Current Version	Recommended Version	License	Current Part
design_1 (6)								
> /myip_0	IP definition refers to missing subcores	Repackage component		myip_v1.0	1.0 (Rev. 4)	1.0 (Rev. 4)	Included	xc7z020clg484-1
> /rst_ps7_0_50M	Up-to-date	No changes required	More info	Processor System Reset	5.0 (Rev. 13)	5.0 (Rev. 13)	Included	xc7z020clg484-1
> /axi_mem_intercon	Up-to-date	No changes required	More info	AXI Interconnect	2.1 (Rev. 19)	2.1 (Rev. 19)	Included	xc7z020clg484-1

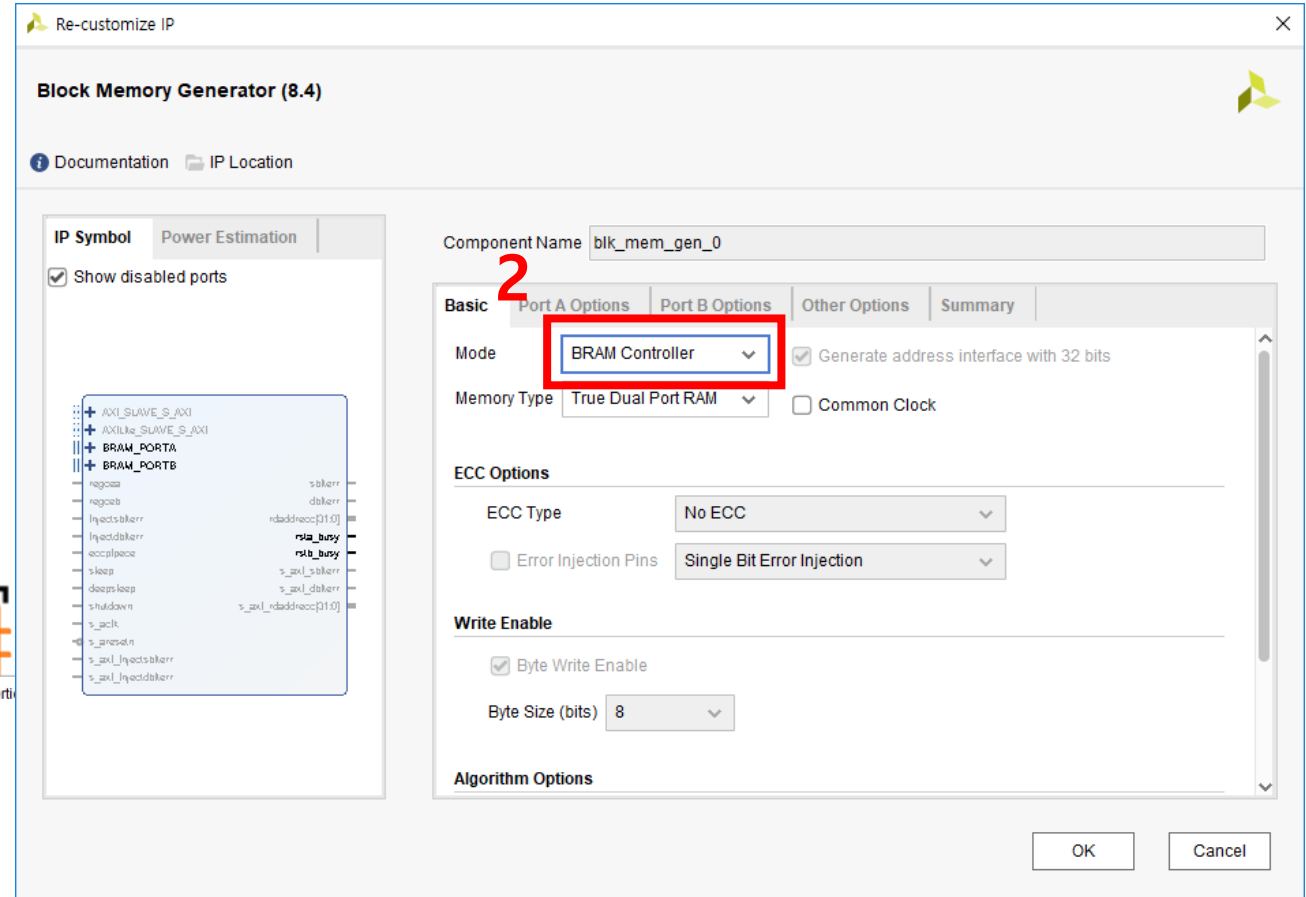
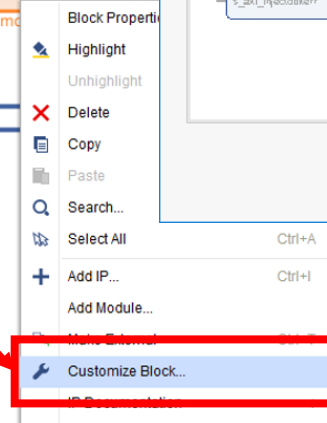
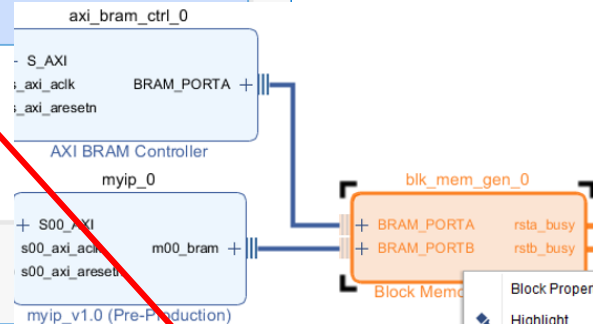
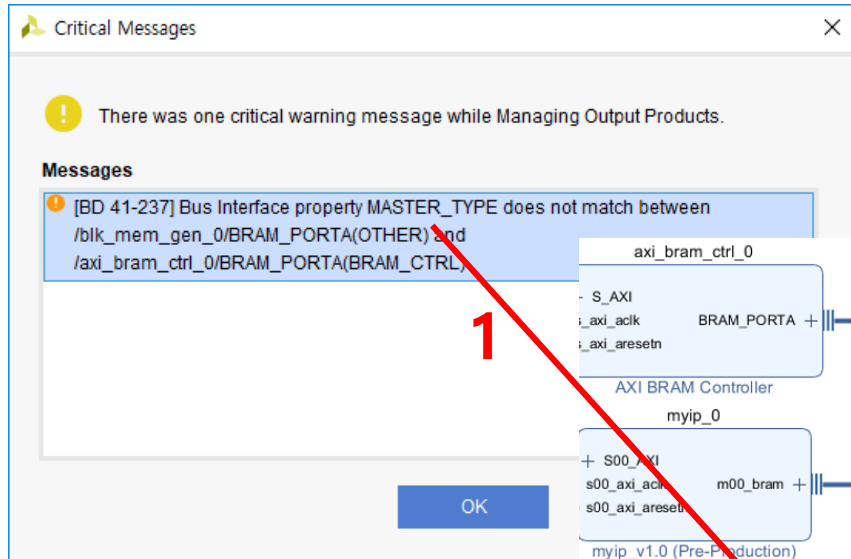
# Upgrading Custom IP

- Don't forget to generate your output products with global synthesis option.



# Type mismatch(General IP)

- If there is type mismatch ...



# Type mismatch(Custom IP)

- In custom IP case, you can solve every problems in IP Packager.

The image shows a sequence of steps to resolve a type mismatch error in the Xilinx IP Packager:

- 1** A critical warning message is displayed: "[BD 41-237] Bus Interface property MASTER\_TYPE does not match between /blk\_mem\_gen\_0/BRAM\_PORTB(BRAM\_CTRL) and /myip\_0/m00\_bram(OTHER)".
- 2** The user right-clicks on the IP block in the block design and selects "Edit in IP Packager" from the context menu.
- 3** The "Edit Interface" dialog is open, showing the "Parameters" tab. The "Choose Parameters to Override" list includes MASTER\_TYPE.
- 4** The "Parameters" tab is selected, and the "MASTER\_TYPE" parameter is set to "BRAM\_CTRL" in the "User Set" section.

The "Edit Interface" dialog also shows a table of parameters to override:

Name	Description	Value
Overridden		
User Set		
Optional		

Note: Parameters presented in the table above will override the values normally calculated by the software. It is recommended that you only set appropriate values for the parameters in the Required User Setting section.