

Practice 10

- Custom IP

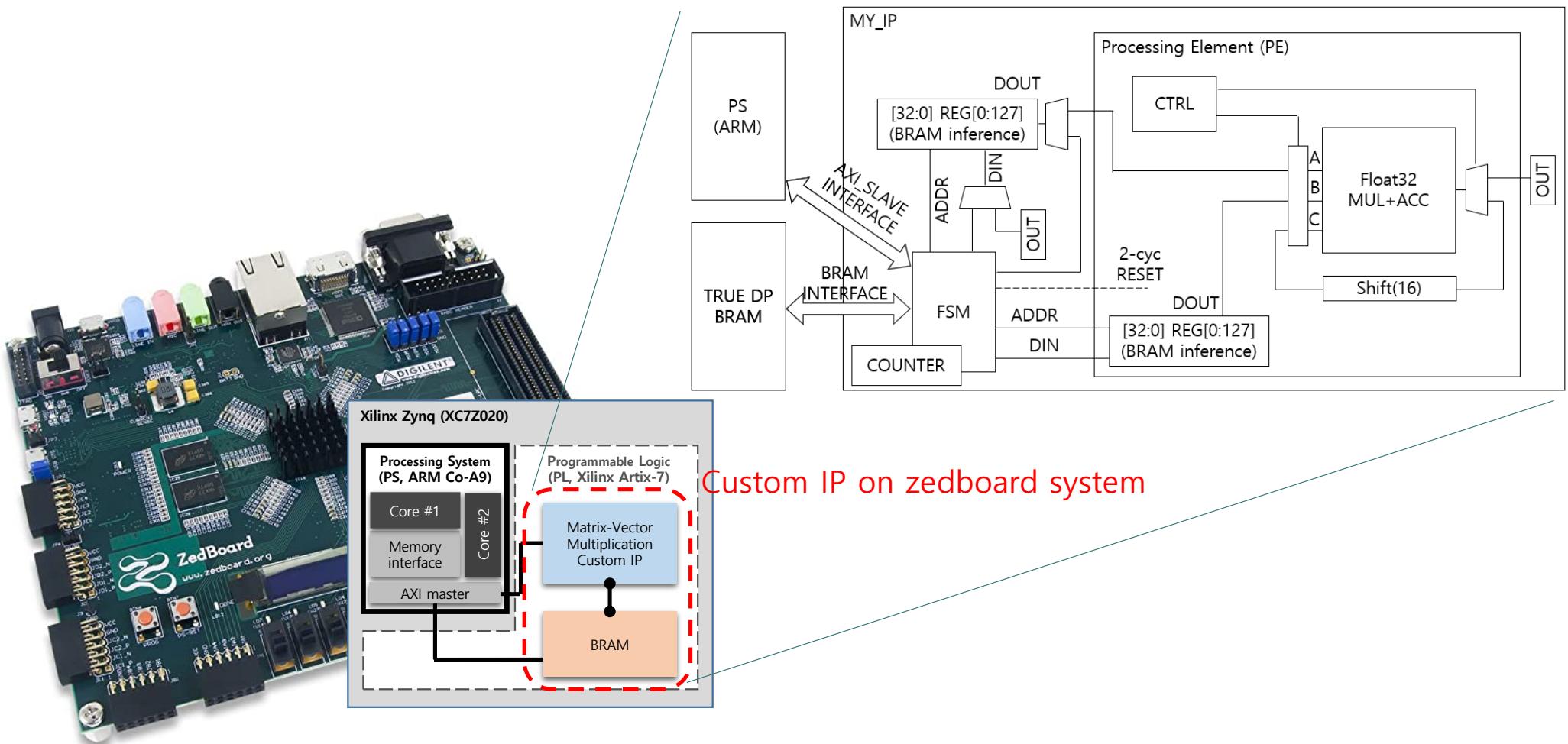
Computing Memory Architecture Lab.

Overview

■ Custom IP Tutorial

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

Final Project Overview: Matrix Multiplication IP



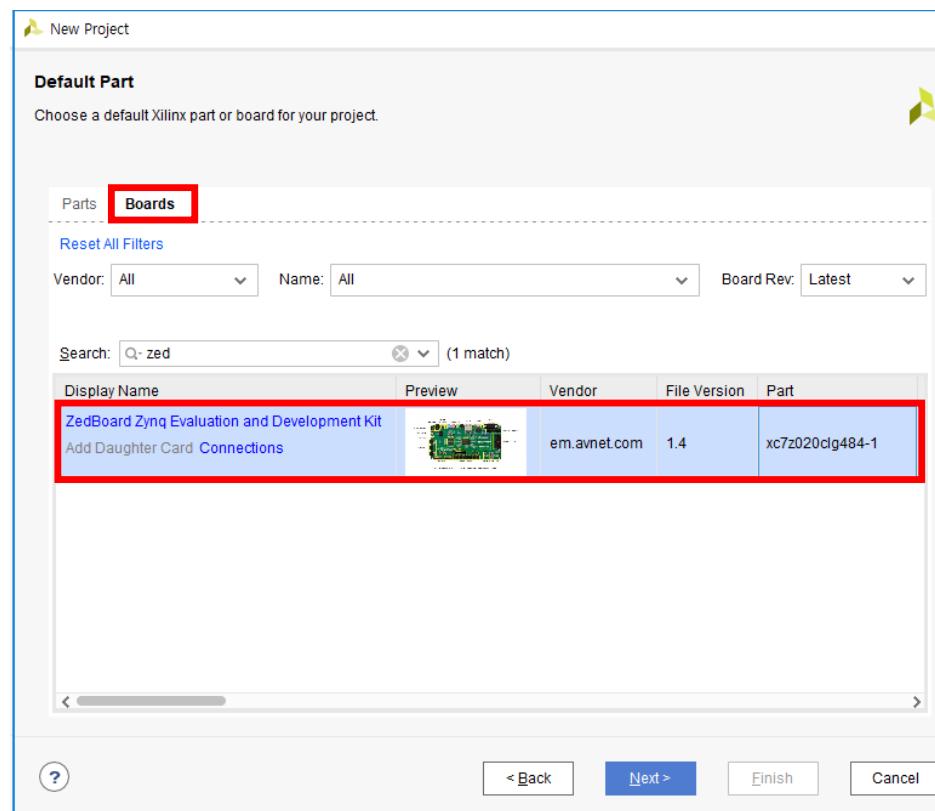
Custom IP Tutorial

Notations

- **HOST\$ XXX**
 - Type XXX @ the terminal of your Ubuntu-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)

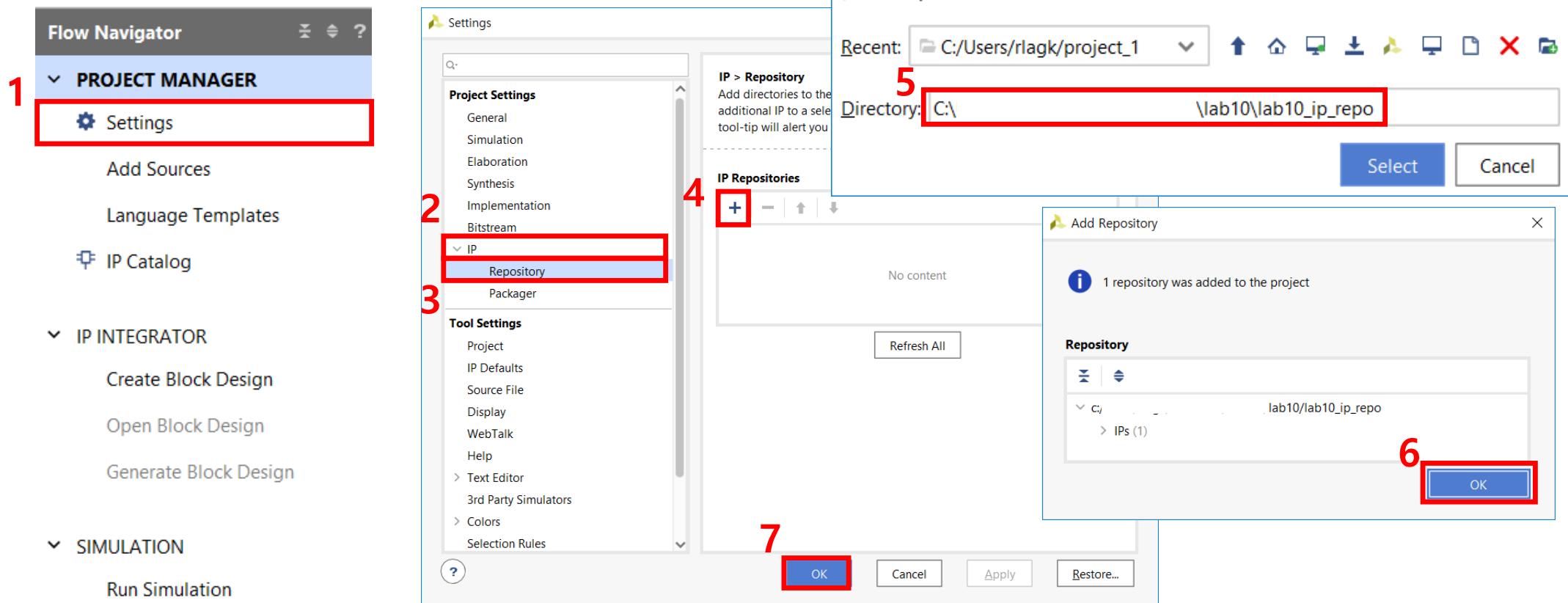
Vivado project creation

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



Example IP - Shifter

- Download an example IP repository
 - HOST\$ tar -zxvf lab10_ip_repo.tar.gz



Example IP - Shifter

- Execute TCL scripts of the example design
 - TCL_Console\$ cd \${DOWNLOAD_LAB10_DIR}
 - TCL_Console\$ source block_design.tcl

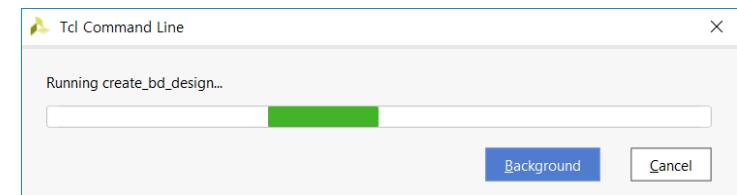
1

The screenshot shows the 'Tcl Console' tab selected in a software interface. Below the tabs are several icons. The main area displays a list of files with their creation dates and names. At the bottom, there is a command input field containing the command 'source block_design.tcl'. A red box highlights the 'Tcl Console' tab, and a red number '1' is positioned to its left.

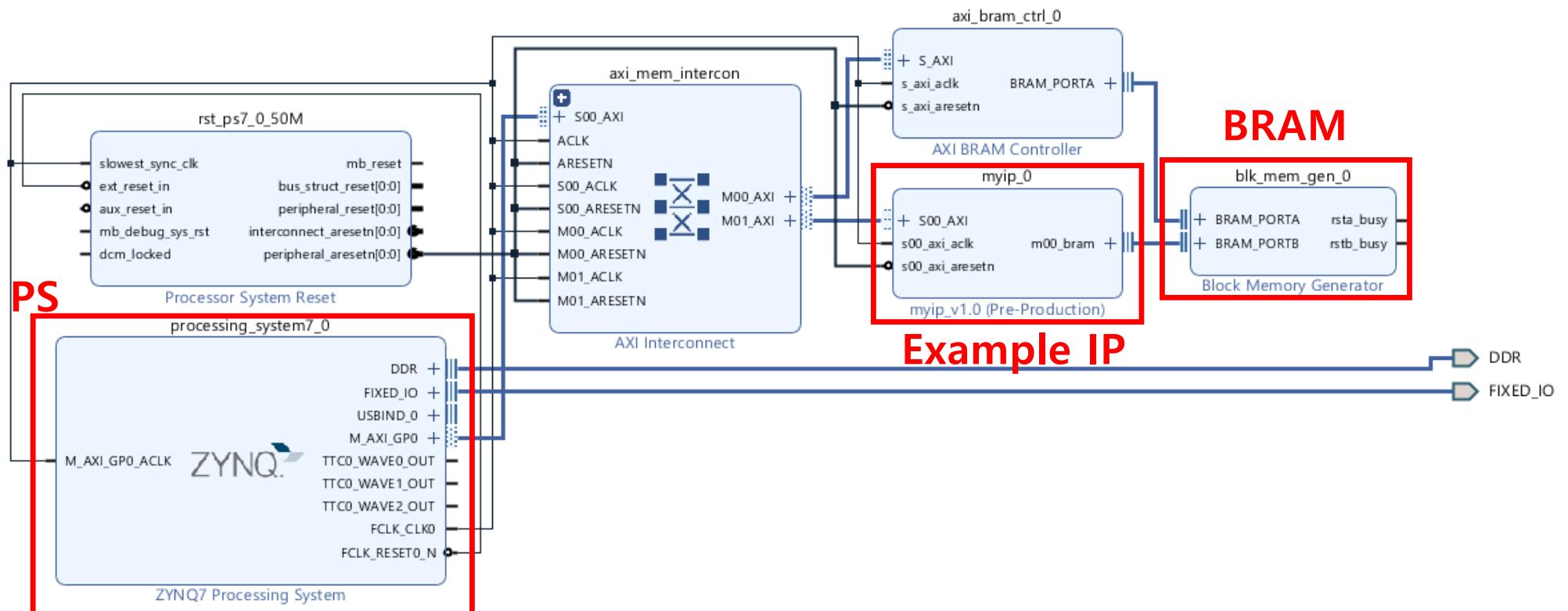
Date	Time	Type	Name
2019-03-27	오전 11:15	<DIR>	.AndroidStudio3.3
2019-03-27	오후 12:19	<DIR>	.gradle
2019-03-03	오후 05:33	<DIR>	.oracle_jre_usage
2019-02-19	오후 05:08	<DIR>	.vscode
2019-03-03	오후 05:00	<DIR>	.Xilinx
2018-12-13	오전 09:46	<DIR>	3D Objects
2017-03-13	오후 07:18		2,215 block_design.tcl
2018-12-13	오전 09:46	<DIR>	Contacts

2

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 x Address Editor x

Cell Slave Interface Base Name Offset Address Range High Address

processing_system7_0

Data (32 address bits : 0x40000000 [1G])

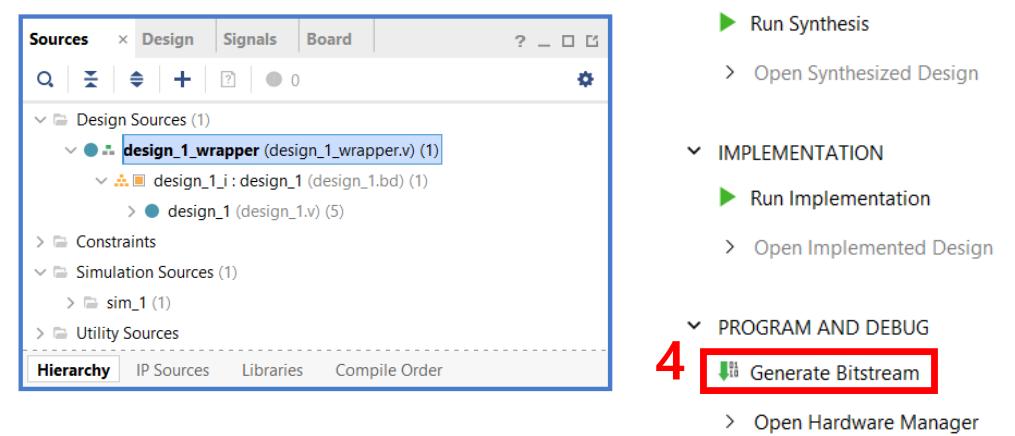
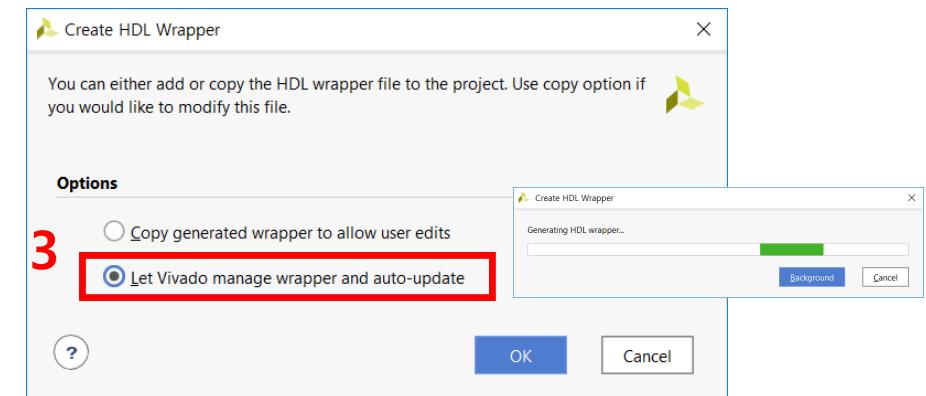
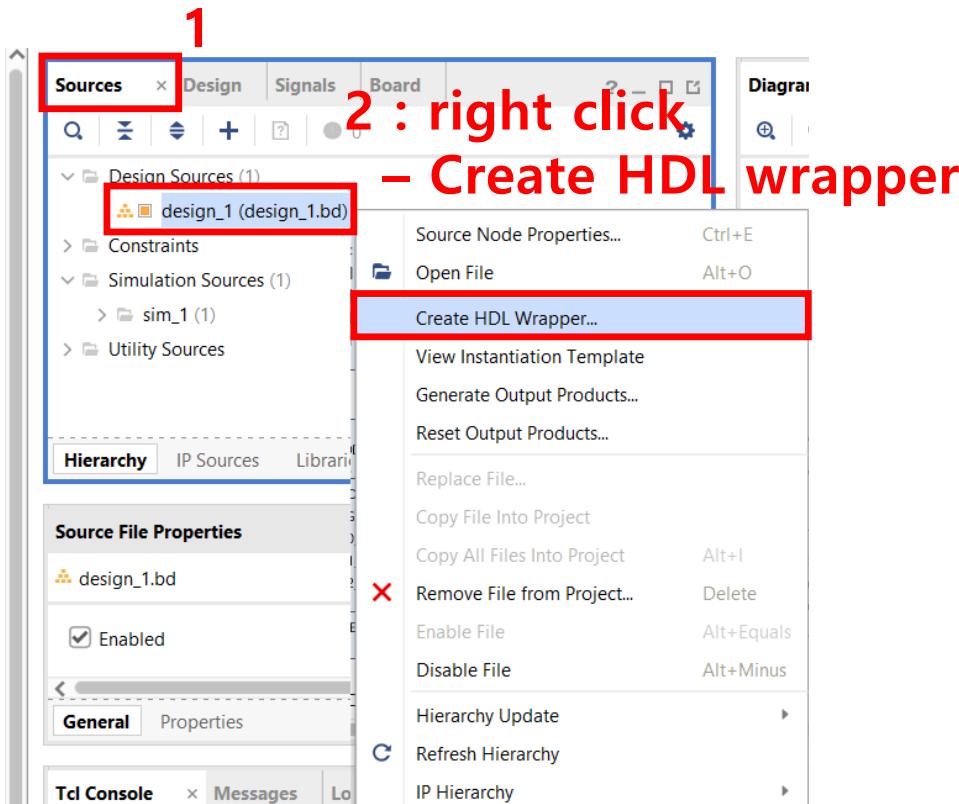
S_AXI	Mem0	0x4000_0000	8K	▼	0x4000_1FFF
S00_AXI	S00_AXI_reg	0x43C0_0000	64K	▼	0x43C0_FFFF

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



Example IP - Shifter

- Preparing SD card (1)
 - Change **zynq.bit** on host
 - Insert sdcard to board
- Set SD boot mode (2)
 - BD.JP7 ~ BD.JP11
 - 5'b01100
- Insert LAN Cable (3)
- Insert power cable (4)
 - Yet stay power OFF
- Connect USB cable (5)
 - BD.J14
 - HOST\$ dmesg
 - Micro 5pin cable
 - If 뻣뻣(stiff) -> 칼팁(sharp tip) -> 후크(hook)
 - No further labs for broken board

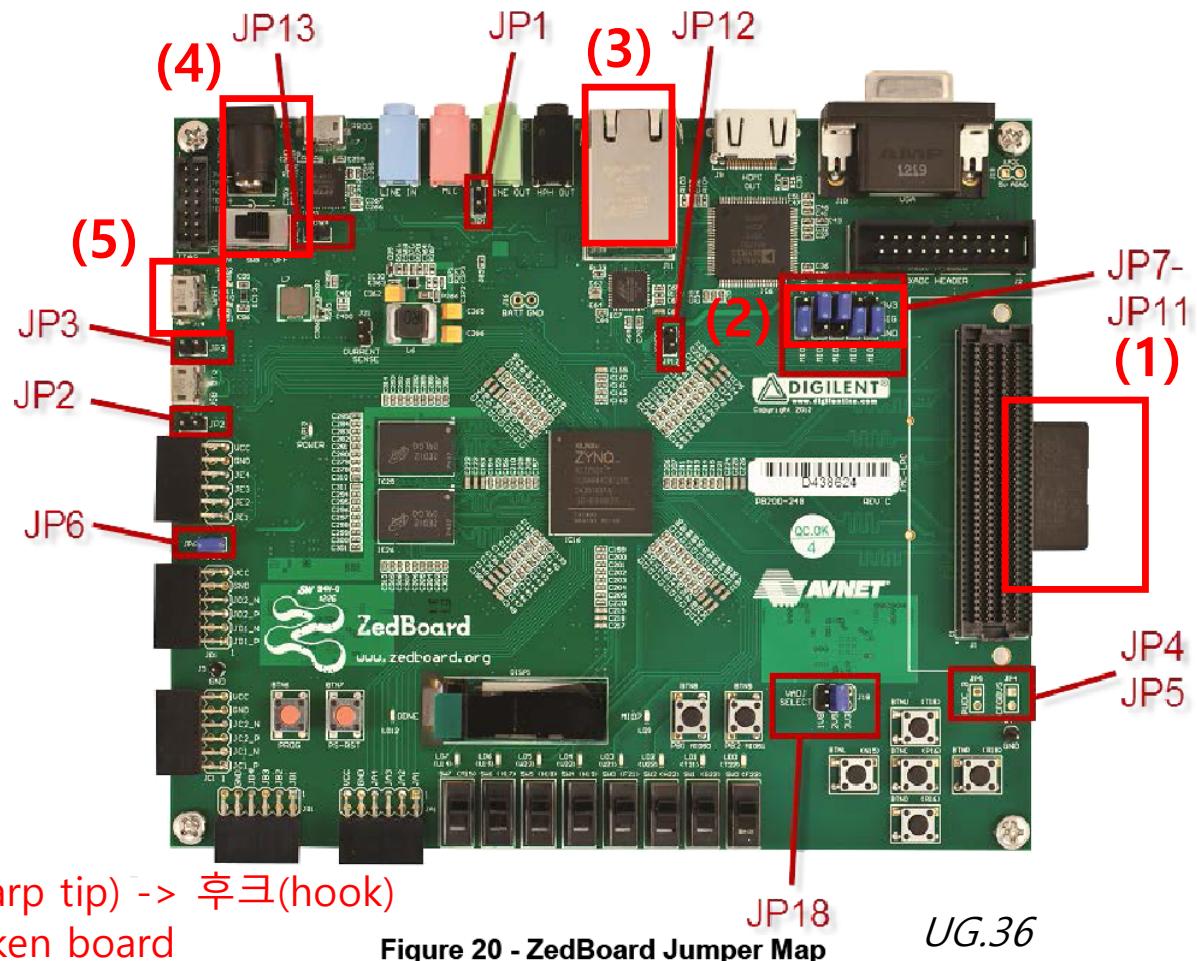


Figure 20 - ZedBoard Jumper Map

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

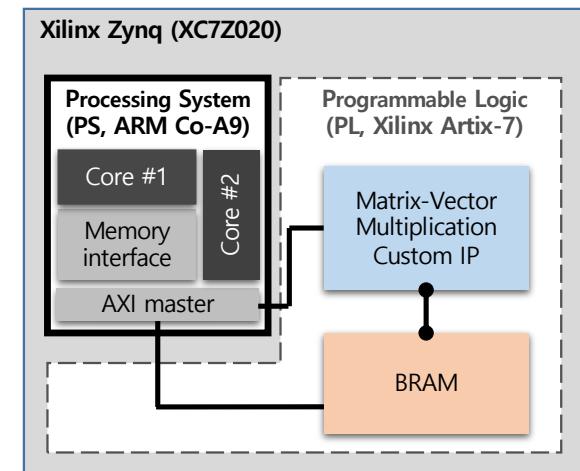
addr	FPGA(hex)
0	0
1	2
2	4
3	6
4	0
5	*(0)<<1
6	*(1)<<1
7	*(2)<<1

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*V 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^*V by making PE Array

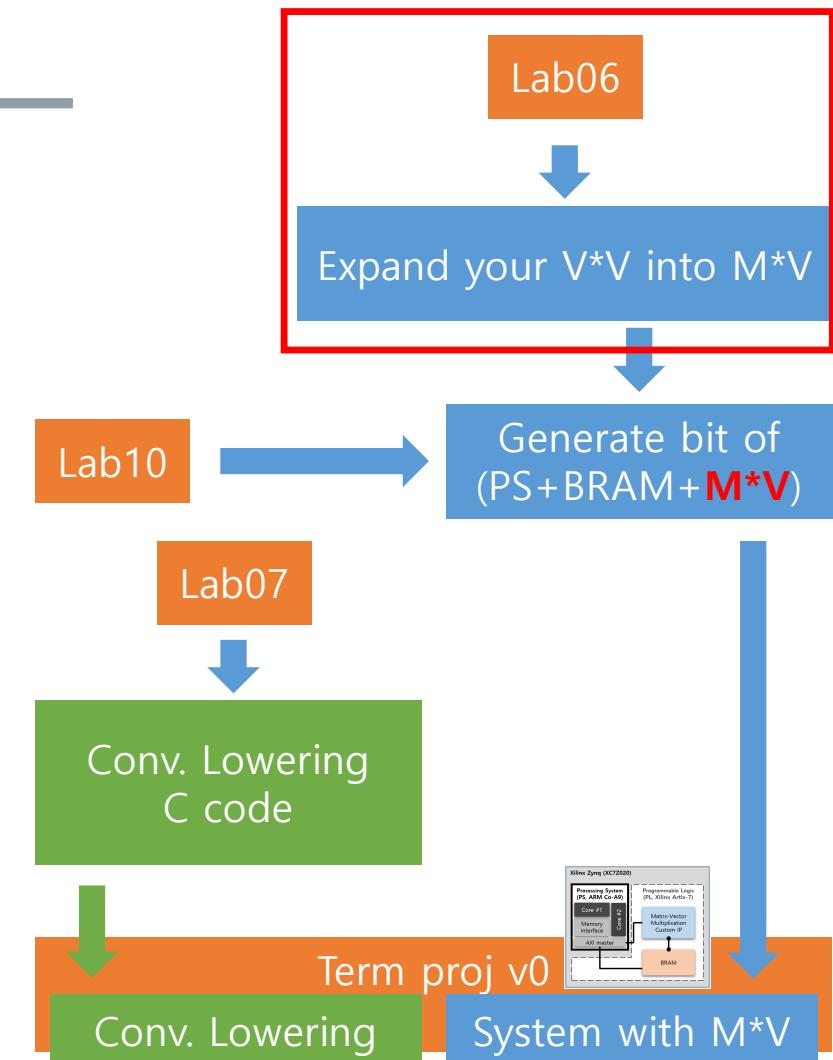
- M^*V by making PE Array

- No required specs

- 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*V, test bench)
- Attach your M*V waveform(simulation result) with [student_number, name]
 - The correct waveform should be shown to confirm the operation of your code.
 - Refer to Practice3 about screenshot.

- Report

- Explain operation of M*V 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(TUE) 23:59

- **No Late Submission**

Term Project v0 최종제출

Lab06

■ Project Requirement

- Convolution lowering + Custom IP system with M*V

- Convolution lowering
 - Refer Lab07
- Custom IP system with M*V
 - Refer Lab10 tutorial(Custom IP) & appendix(editing the custom IP)

- Demo

- Zedboard

Expand your V*V into M*V

Lab10

Generate bit of
(PS+BRAM+**M*V**)

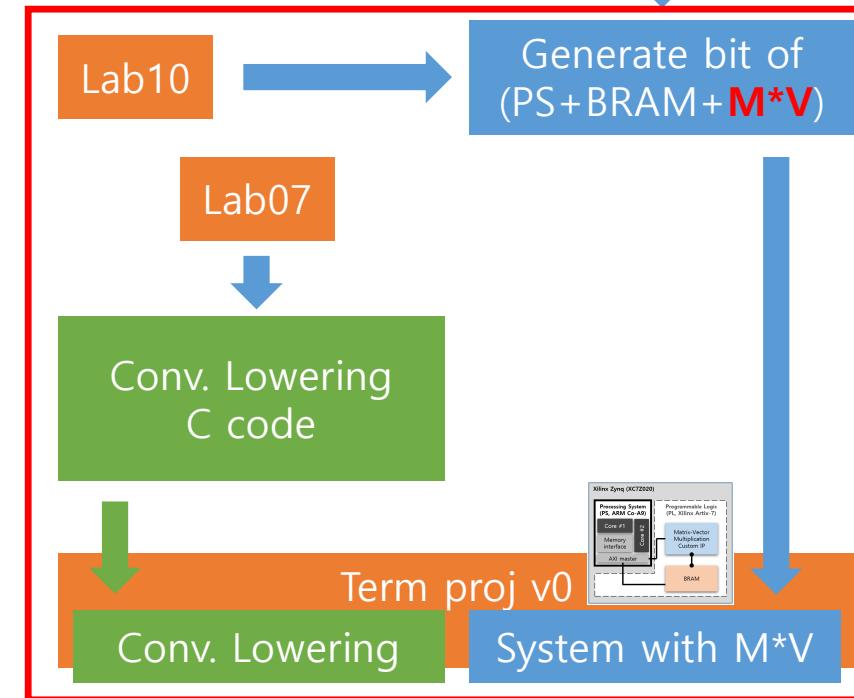
Lab07

Conv. Lowering
C code

Term proj v0

Conv. Lowering

System with M*V



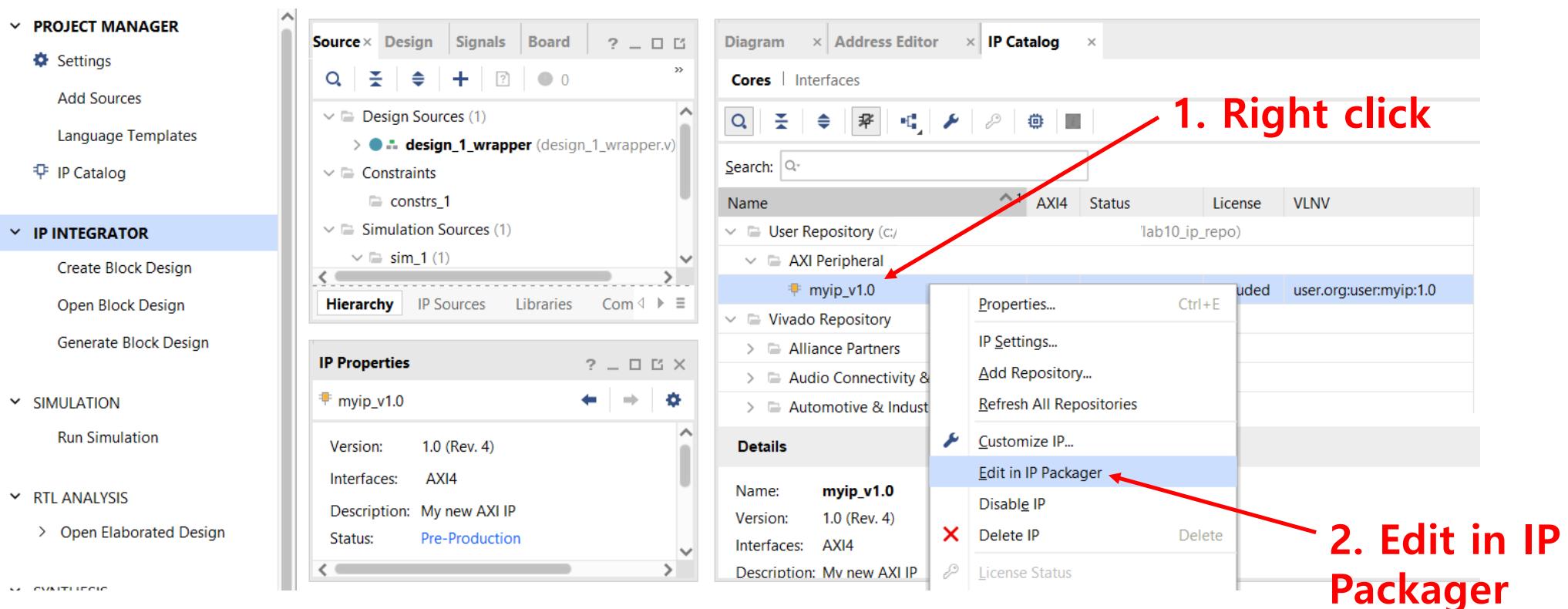
Term Project v0 최종제출

- Due: 6/16(TUE) 23:59
 - **Details will be noticed next week**

Appendix – Editing the Custom IP

Editing Custom IP

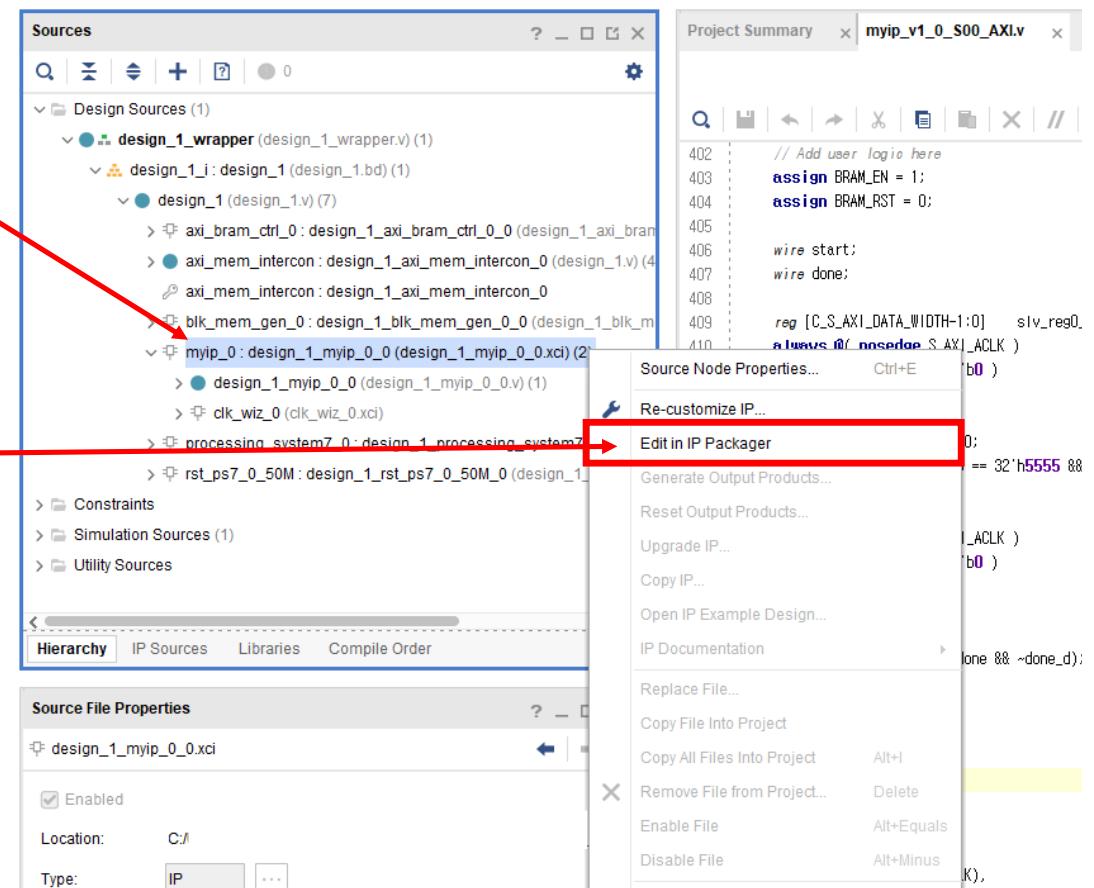
- (1) IP Catalog -> Edit in IP Packager -> IP project



Editing Custom IP

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

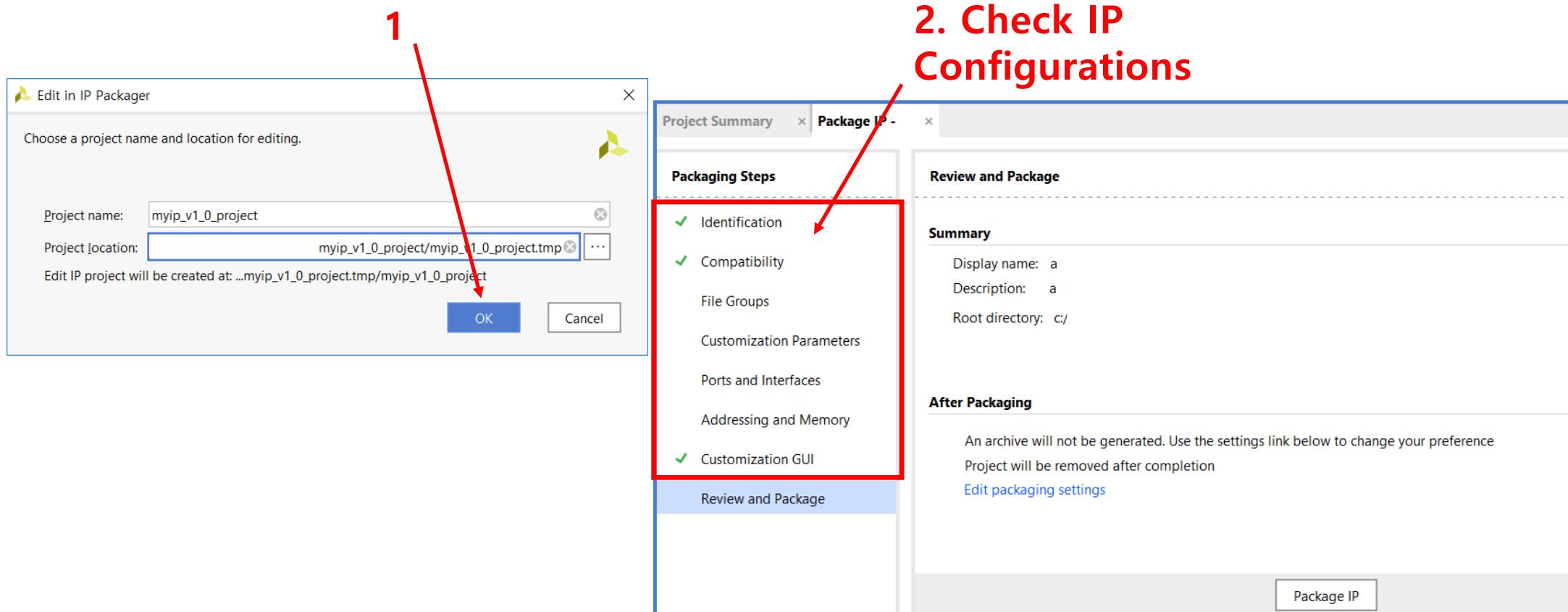
1. Right click on custom IP



2. Edit in IP Packager

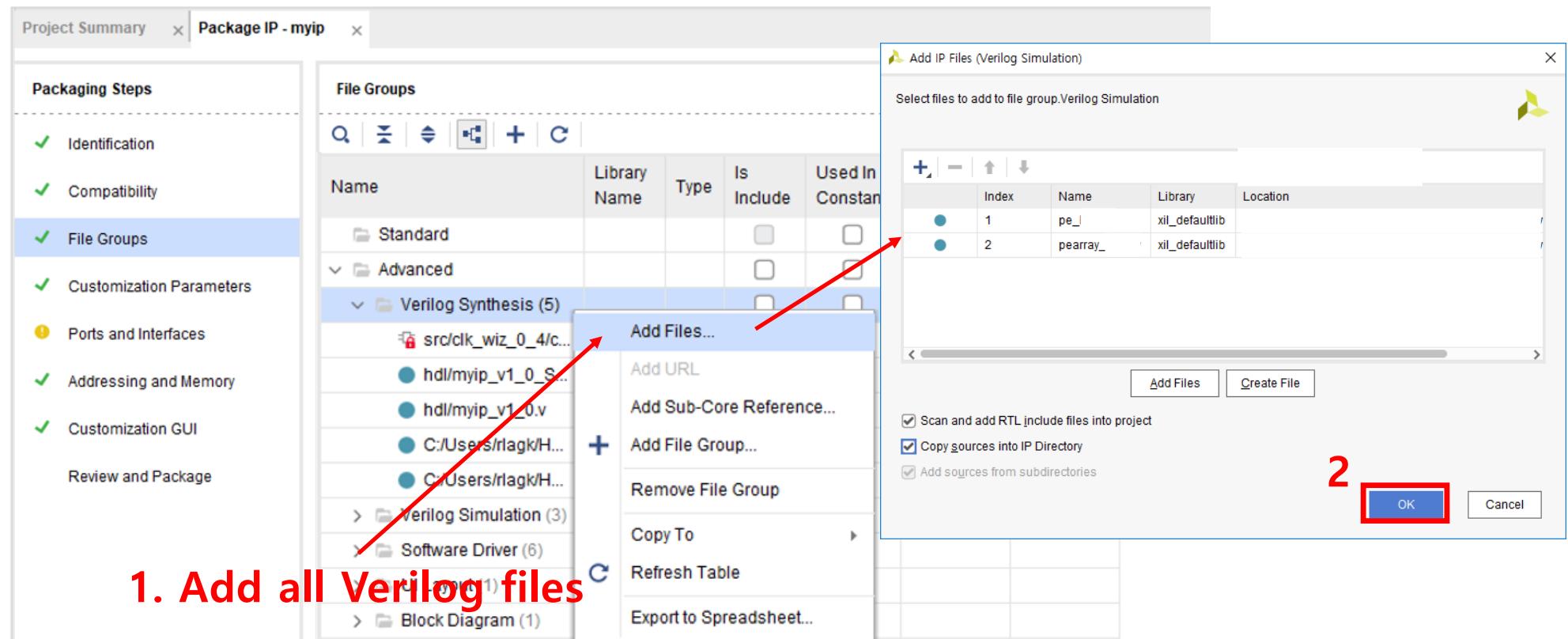
Packaging Custom IP

- Package IP -> Configurations



Checking IP Configurations

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



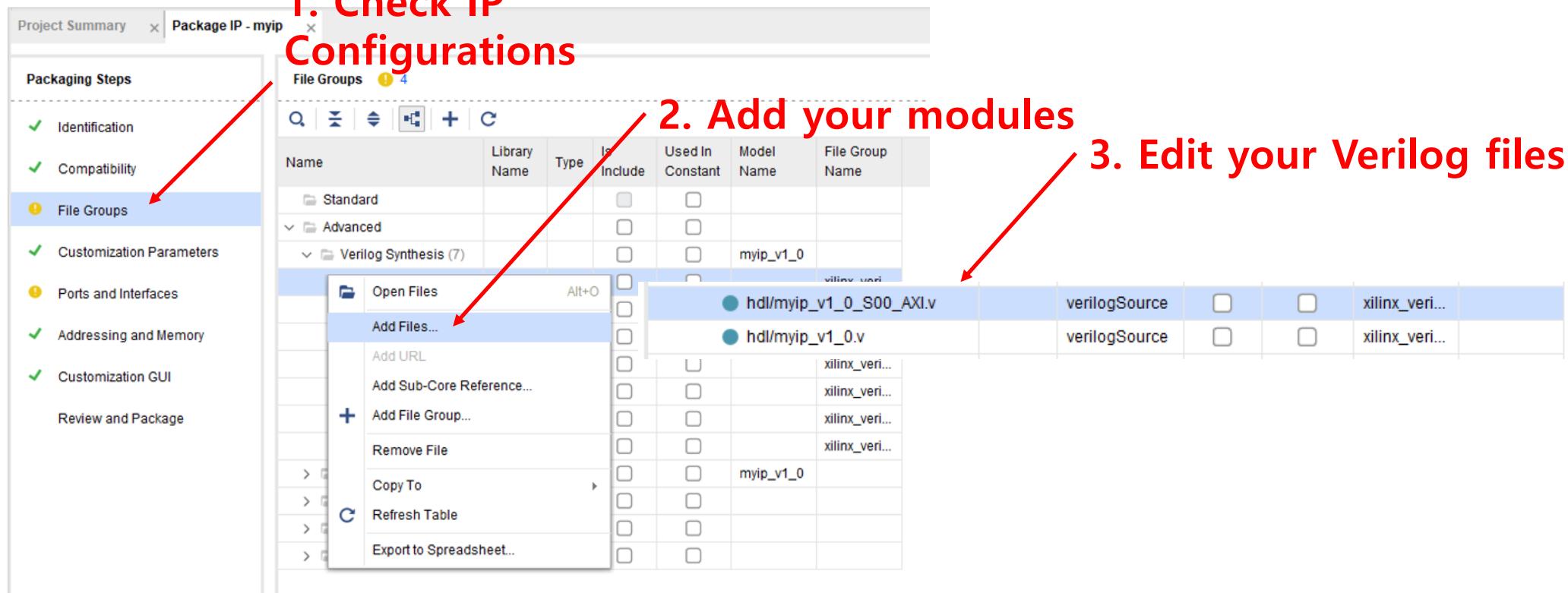
Checking IP Configurations

- 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

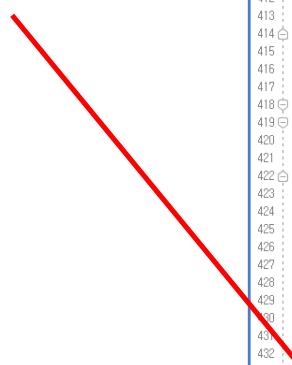


Checking IP Configurations

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

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

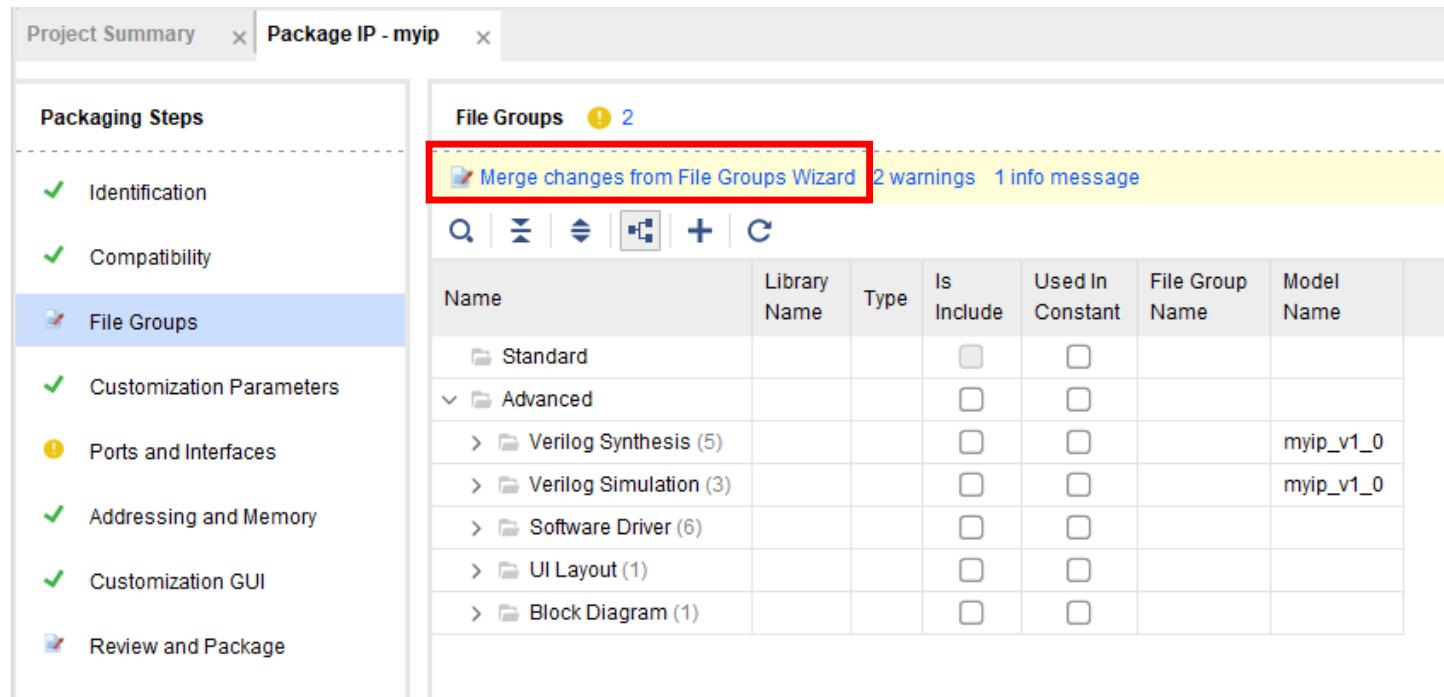
2. Revise myip_v1_0_S00_AXI.v &
myip_v1_0.v codes
- especially ports, parameters, modules
you needed



```
Project Summary x Package IP - myip x myip_v1_0_S00_AXI.v x peararray_my.v x
Q | I | ← | → | X | D | B | X | // | E | ? |
403 assign BRAM_EN = 1;
404 assign BRAM_RST = 0;
405
406 wire start;
407 wire done;
408
409 req [C_S_AXI_DATA_WIDTH-1:0] siv_req0_d;
410 always @(posedge S_AXI_ACLK)
411 if (S_AXI_ARESETN == 1'b0)
412 siv_req0_d <= 32'd0;
413 else
414 siv_req0_d <= siv_req0;
415 assign start = (siv_req0 == 32'h5555 && siv_req0_d == 32'd0);
416
417 reg done_d;
418 always @(posedge S_AXI_ACLK)
419 if (S_AXI_ARESETN == 1'b0)
420 done_d <= 1'b0;
421 else
422 done_d <= done;
423 assign run_complete = (done && ~done_d);
424
425 peararray_my #((
426 .HSIZE(H_SIZE)
427 )
428 (
429 //from axi
430 .start(start),
431 .done(done),
432 .S_AXI_ACLK(S_AXI_ACLK),
433 .S_AXI_ARESETN(S_AXI_ARESETN),
434
435 //to BRAM
436 .BRAM_ADDR(BRAM_ADDR),
437 .BRAM_WDATA(BRAM_WDATA),
438 .BRAM_WEN(BRAM_WEN),
439 .BRAM_CLK(BRAM_CLK), // 180 degree shifted version of S_AXI_ACLK
440 .BRAM_RDATA(BRAM_RDATA)
441 );
442
443 // User logic ends
444
445 endmodule
446
```

Checking IP Configurations

- Changing IP files
- File Groups -> Merge changes

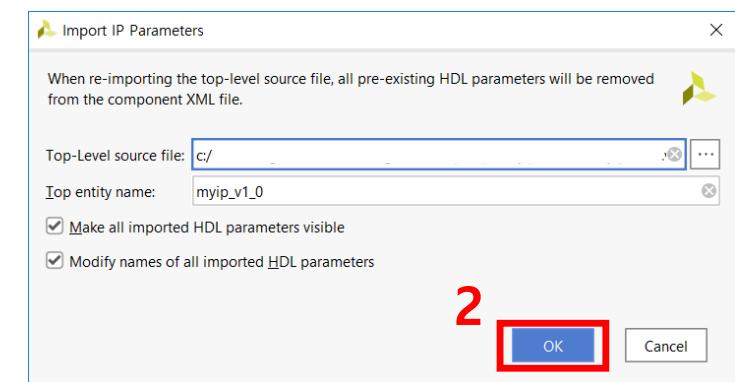


Checking IP Configurations

- Parameter configuration
- Customization Parameters -> import IP parameters

The screenshot shows the Quartus Project Manager interface. On the left, a sidebar lists 'Packaging Steps': Identification, Compatibility, File Groups, **Customization Parameters**, Ports and Interfaces, Addressing and Memory, Customization GUI, and Review and Package. The 'Customization Parameters' step is selected. The main area displays a table titled 'Customization Parameters' with columns: Name, Description, and Display Name. The table contains four rows under 'Customization Parameters' and one row under 'Hidden Parameters'. A context menu is open over the table, with the 'Import IP Parameters...' option highlighted.

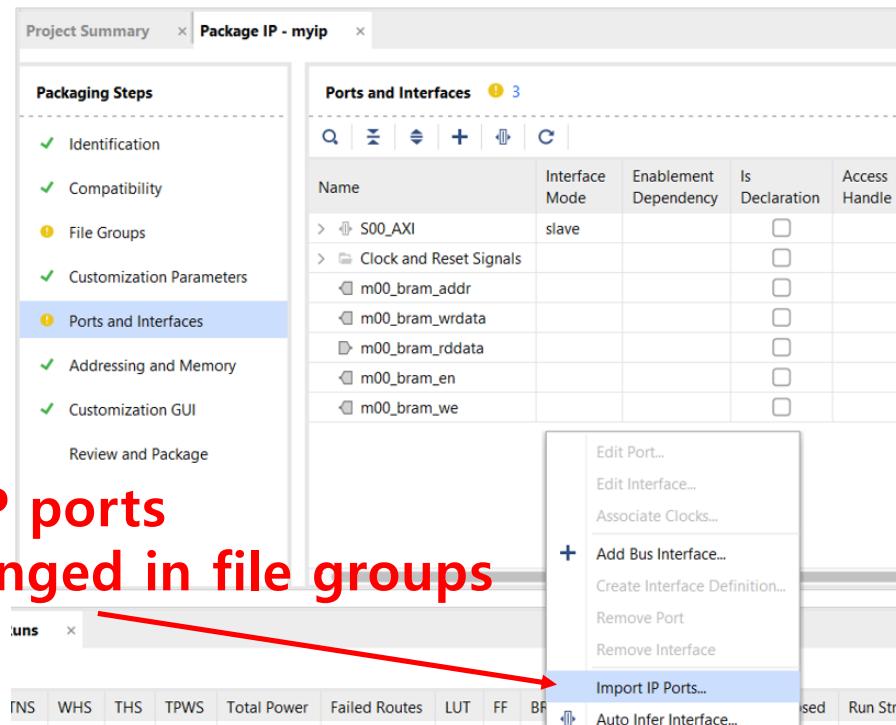
Name	Description	Display Name
C_S00_AXI_DATA_WIDTH	Width of S_AXI data bus	C S00 AXI DATA WII
C_S00_AXI_ADDR_WIDTH	Width of S_AXI address bus	C S00 AXI ADDR WI
C_S00_AXI_BASEADDR		C S00 AXI BASEADC
C_S00_AXI_HIGHADDR		C S00 AXI HIGHADC



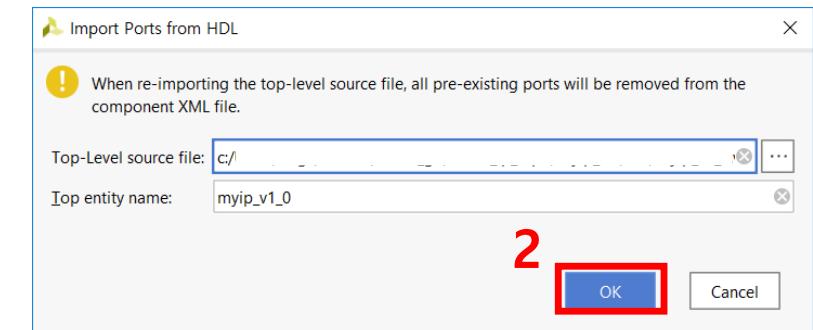
1. Import IP parameters you've changed in file groups

Checking IP Configurations

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

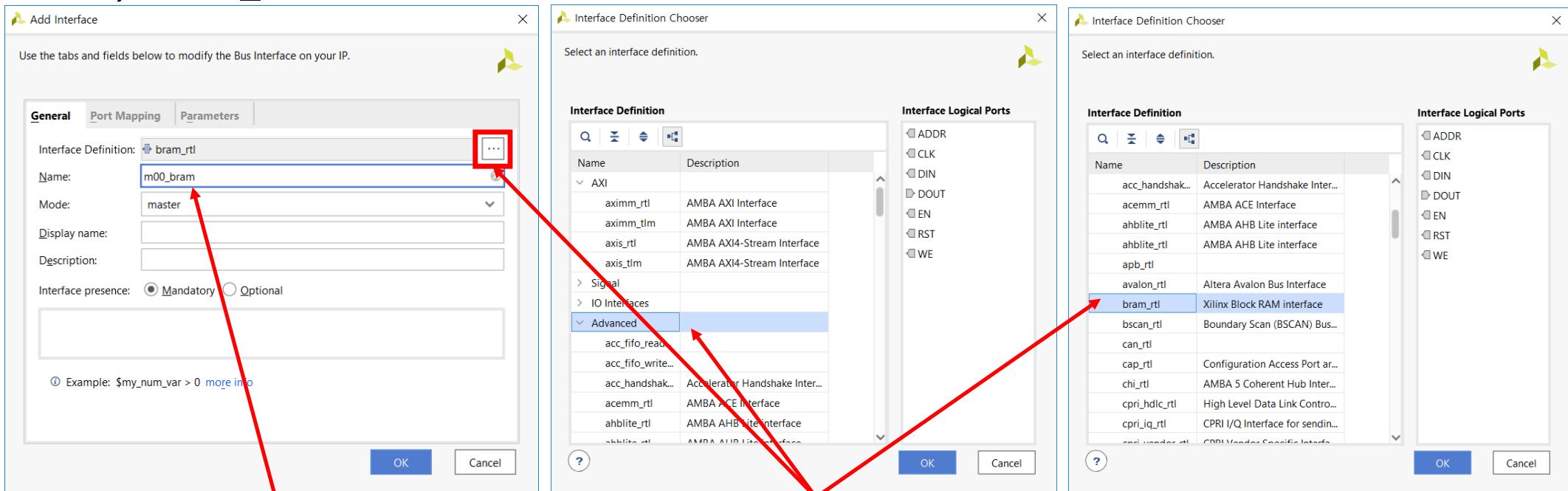


1. Import IP ports
you've changed in file groups



Checking IP Configurations

- Port configuration
- Grouping ports - case1 : when the group doesn't exist
- ex) m00_bram

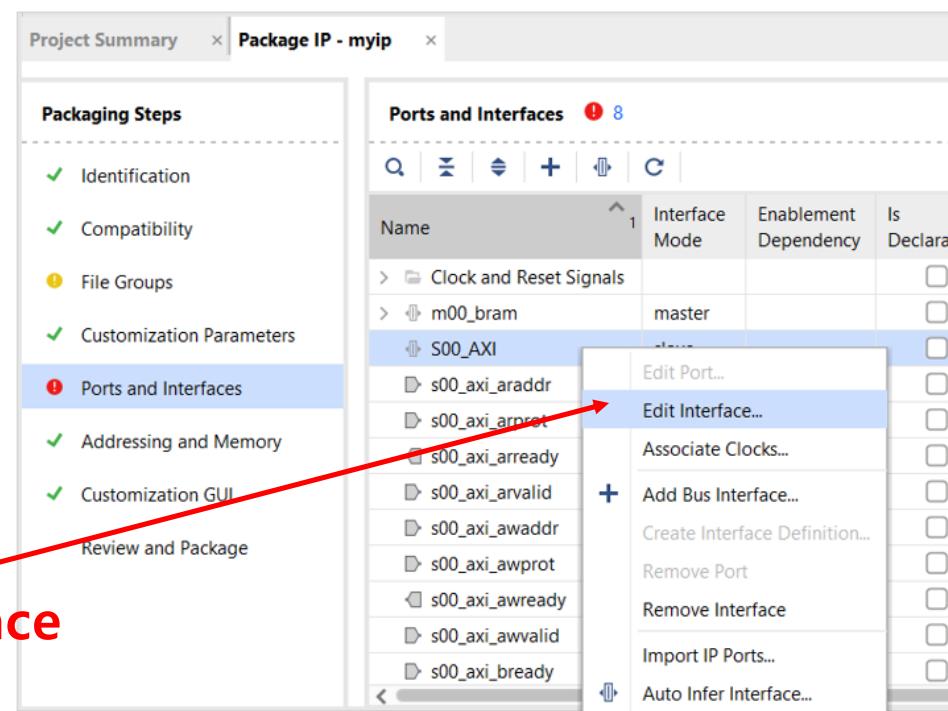


1. Set your port name

2. Advanced – bram rtl

Checking IP Configurations

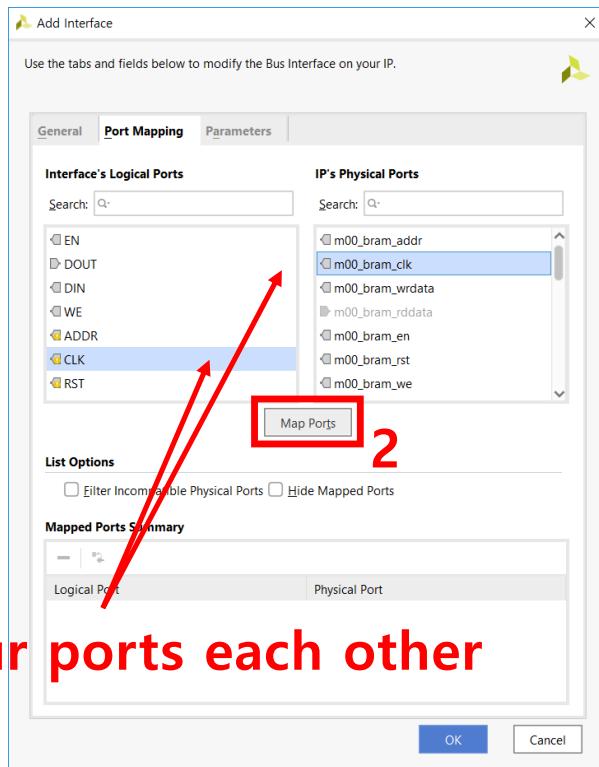
- Port configuration
- Grouping ports – case2 : when the group exists
- ex) S00_AXI



1. Edit Interface

Checking IP Configurations

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



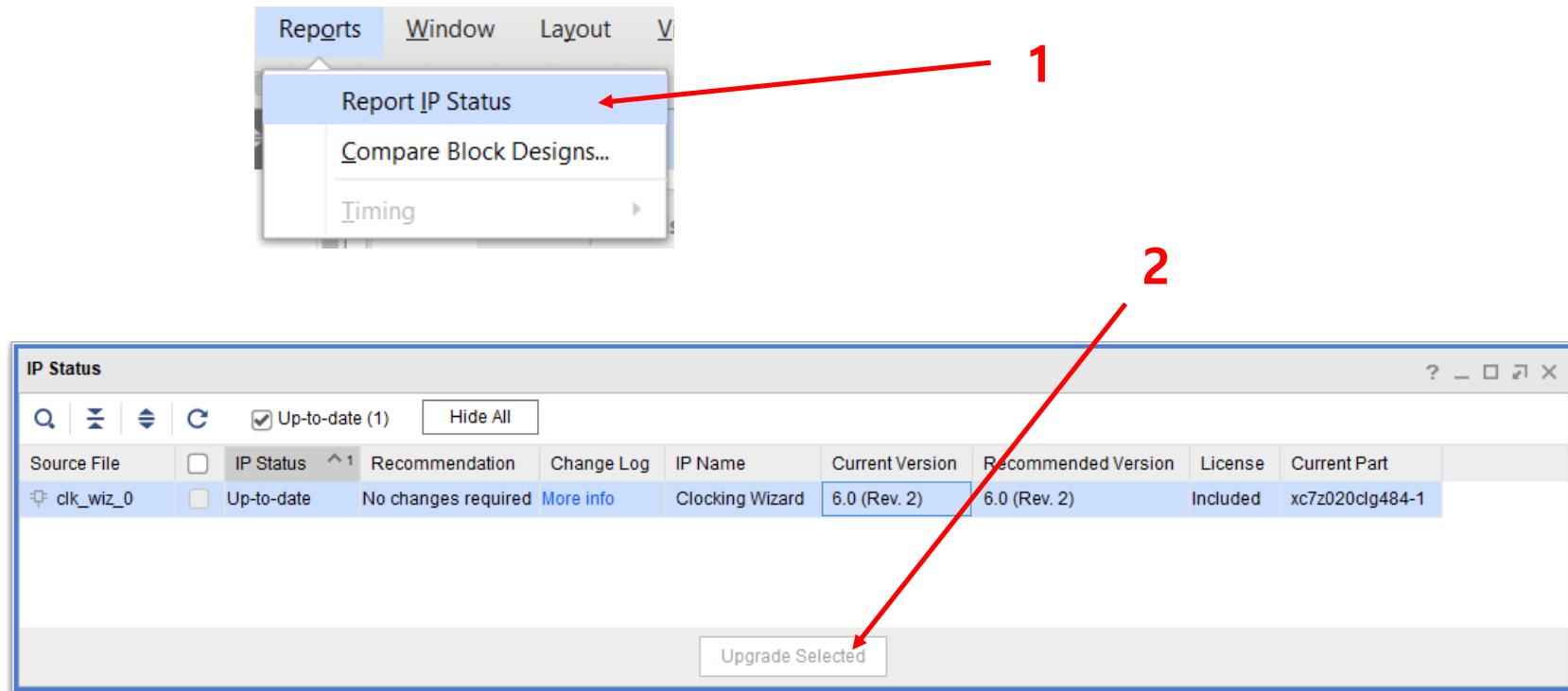
1. Match your ports each other

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

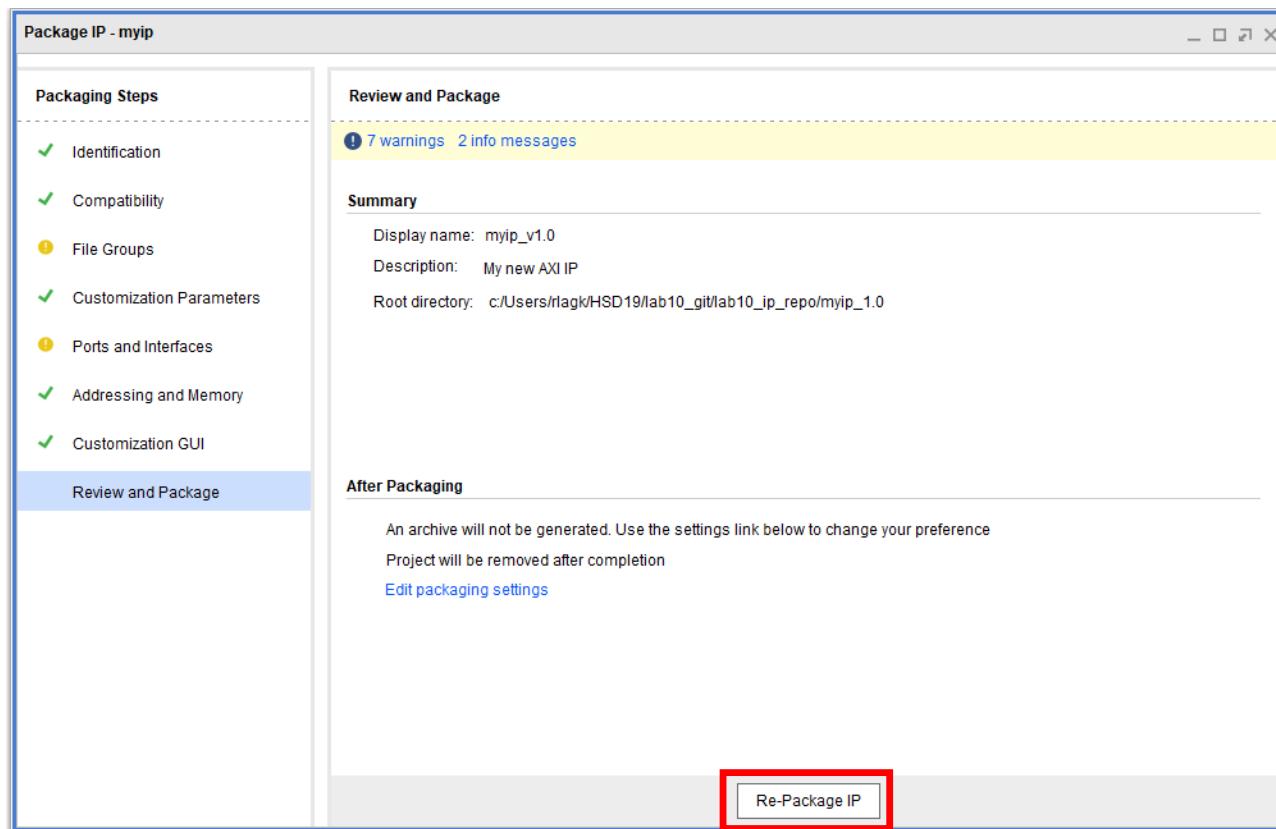
Upgrading Custom IP

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



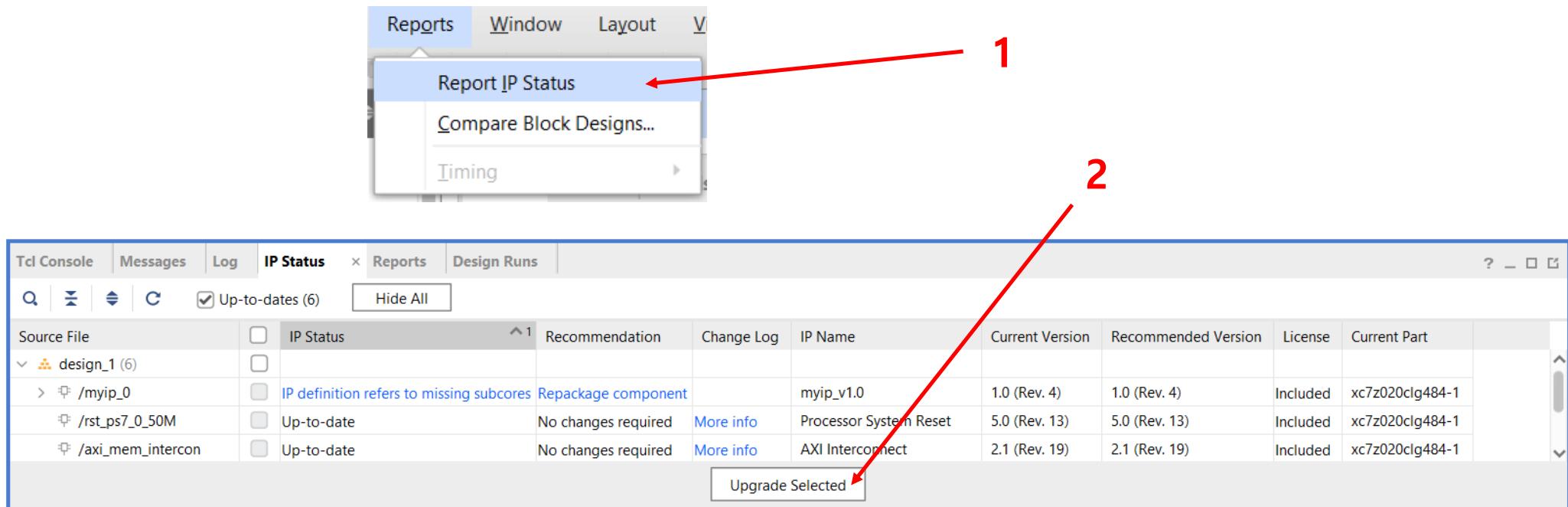
Checking IP Configurations

■ Re-Package IP



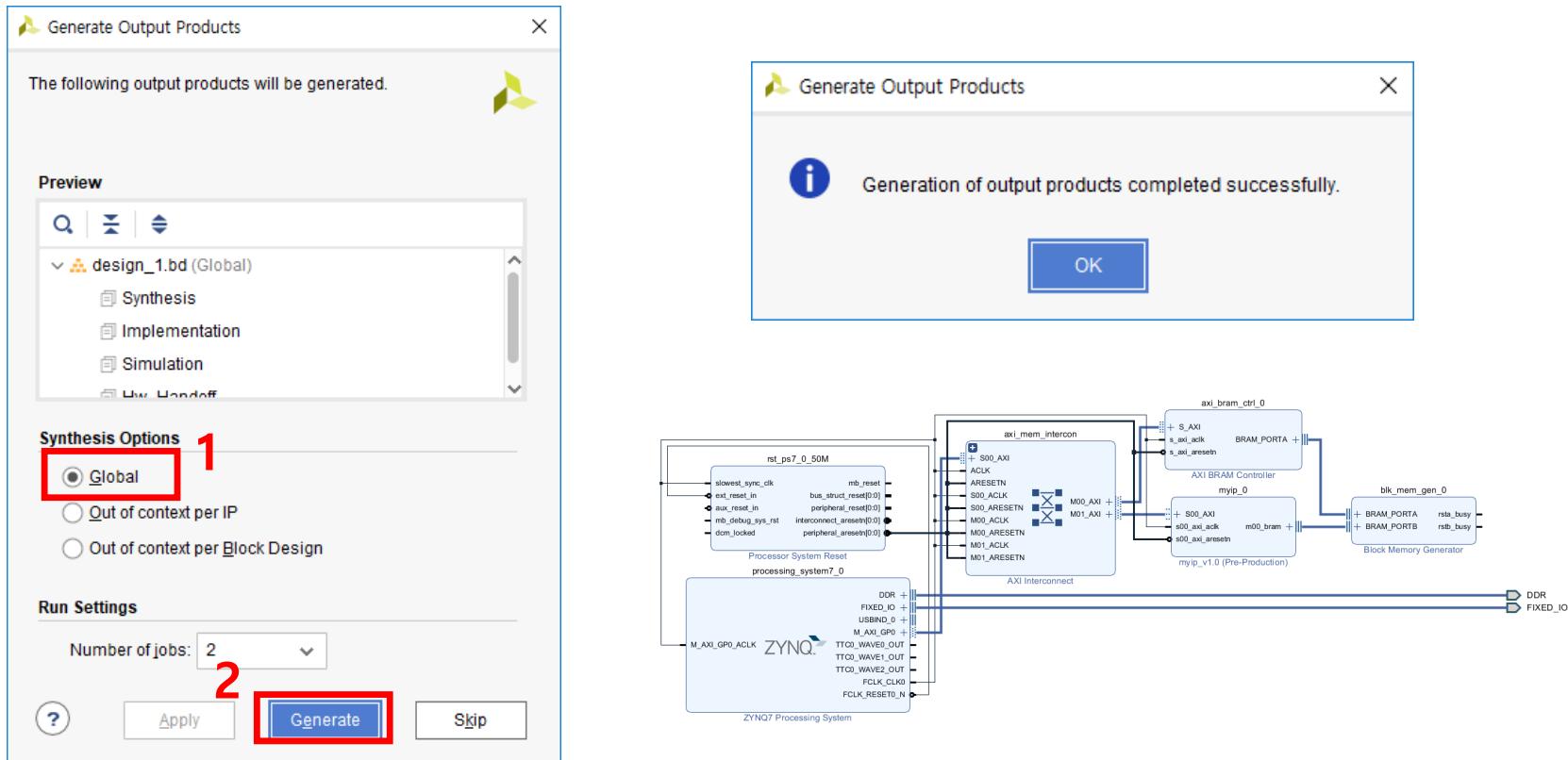
Upgrading Custom IP

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



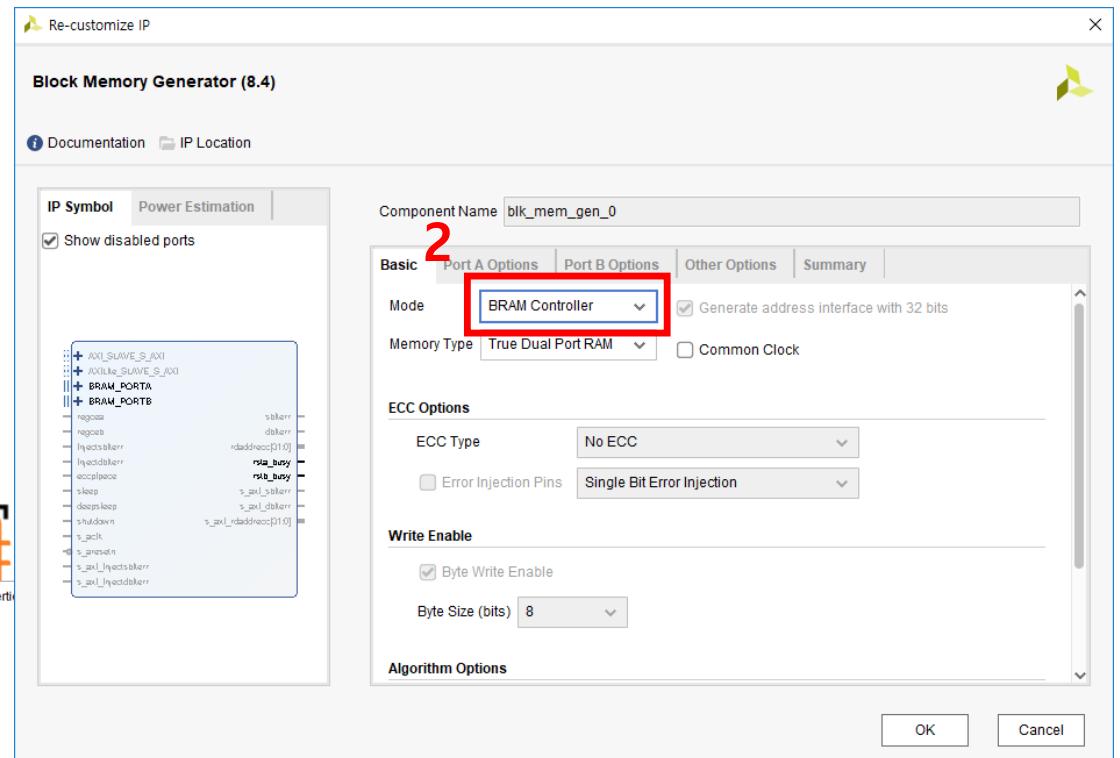
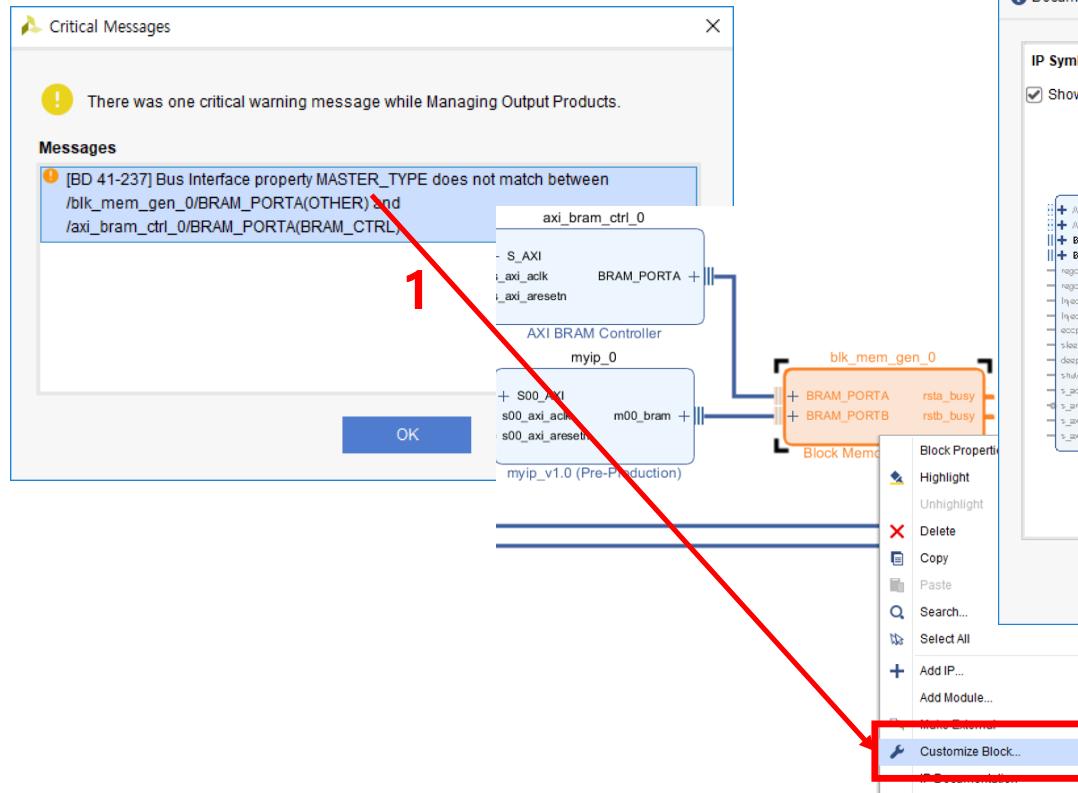
Upgrading Custom IP

- 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.

