Practice 10

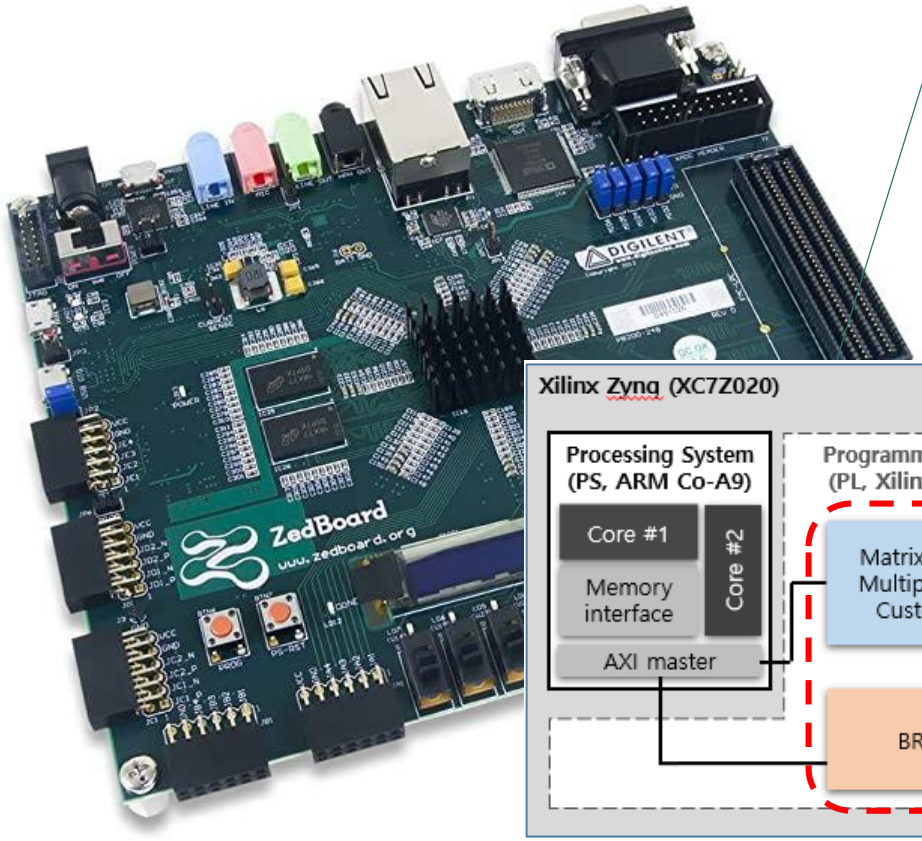
- Custom IP

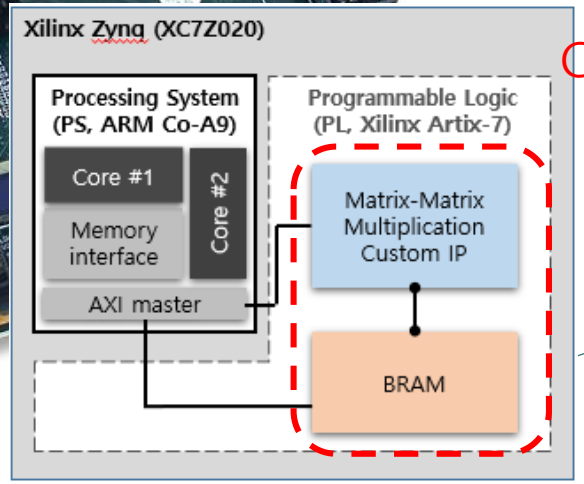
Computing Memory Architecture Lab.

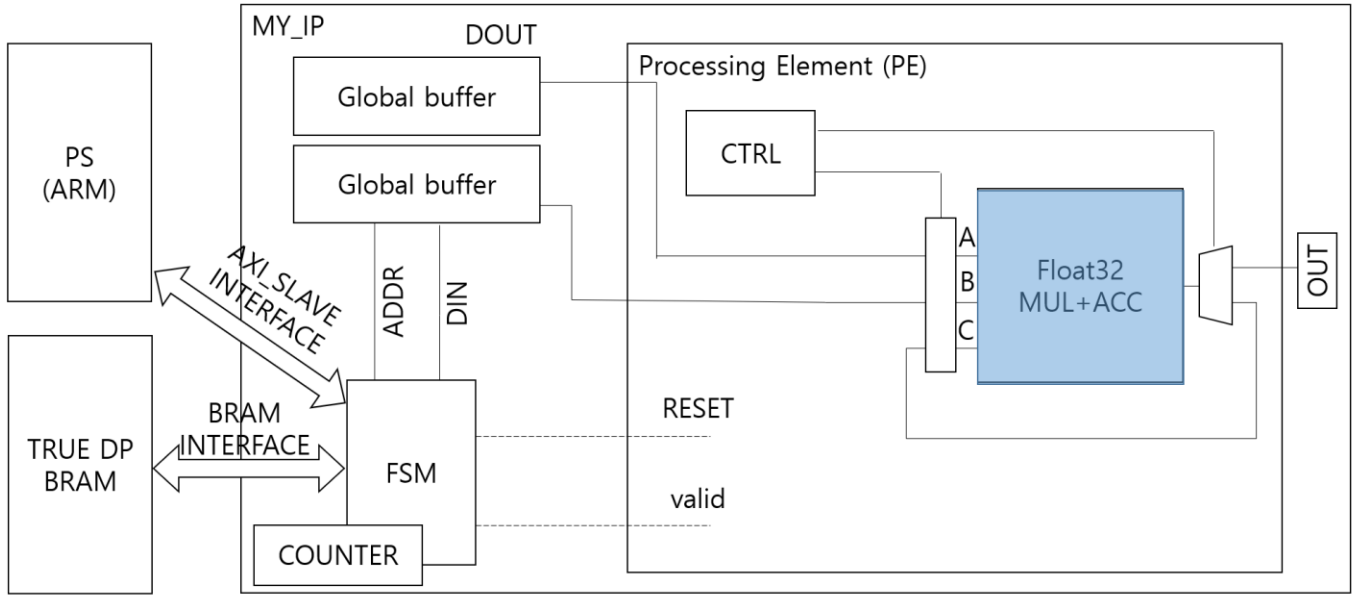
Overview

- **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

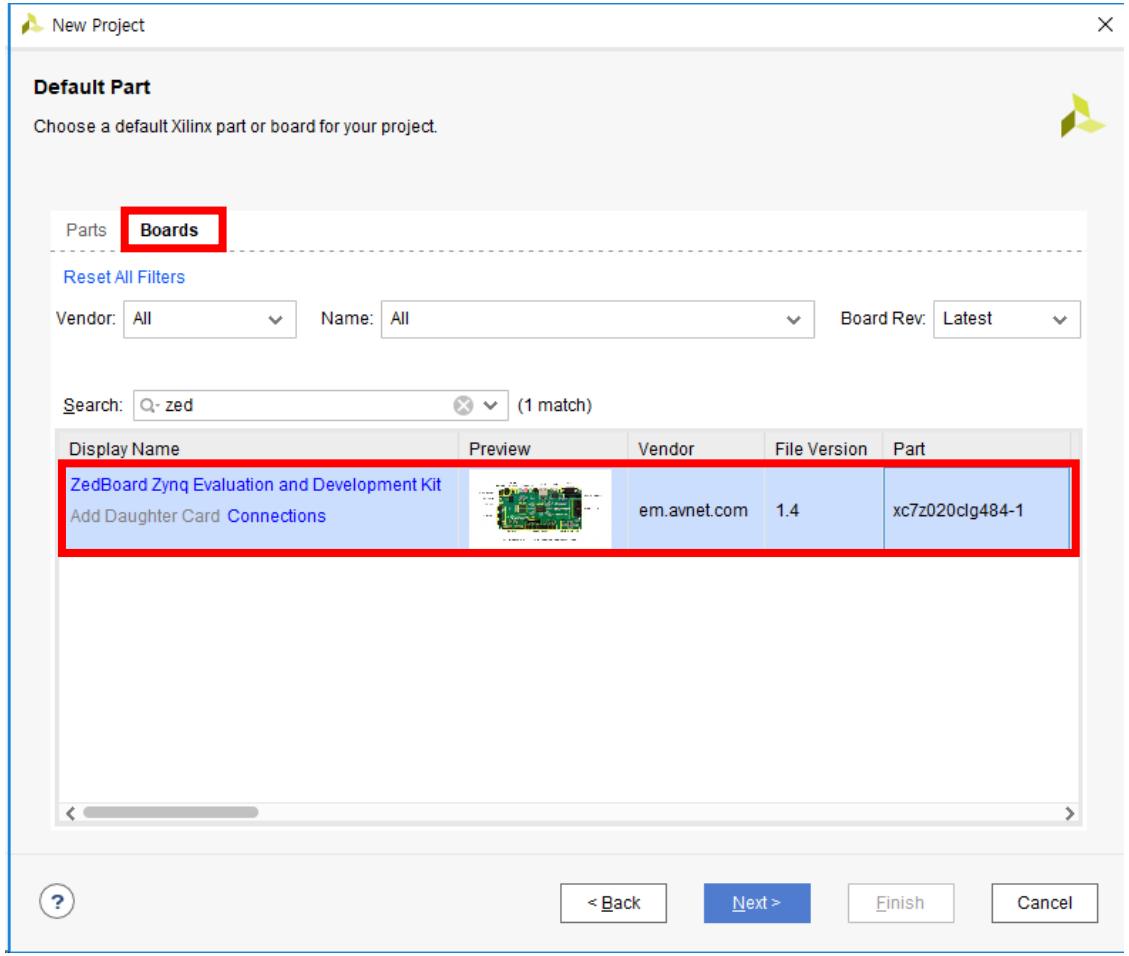
Custom IP Tutorial

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)

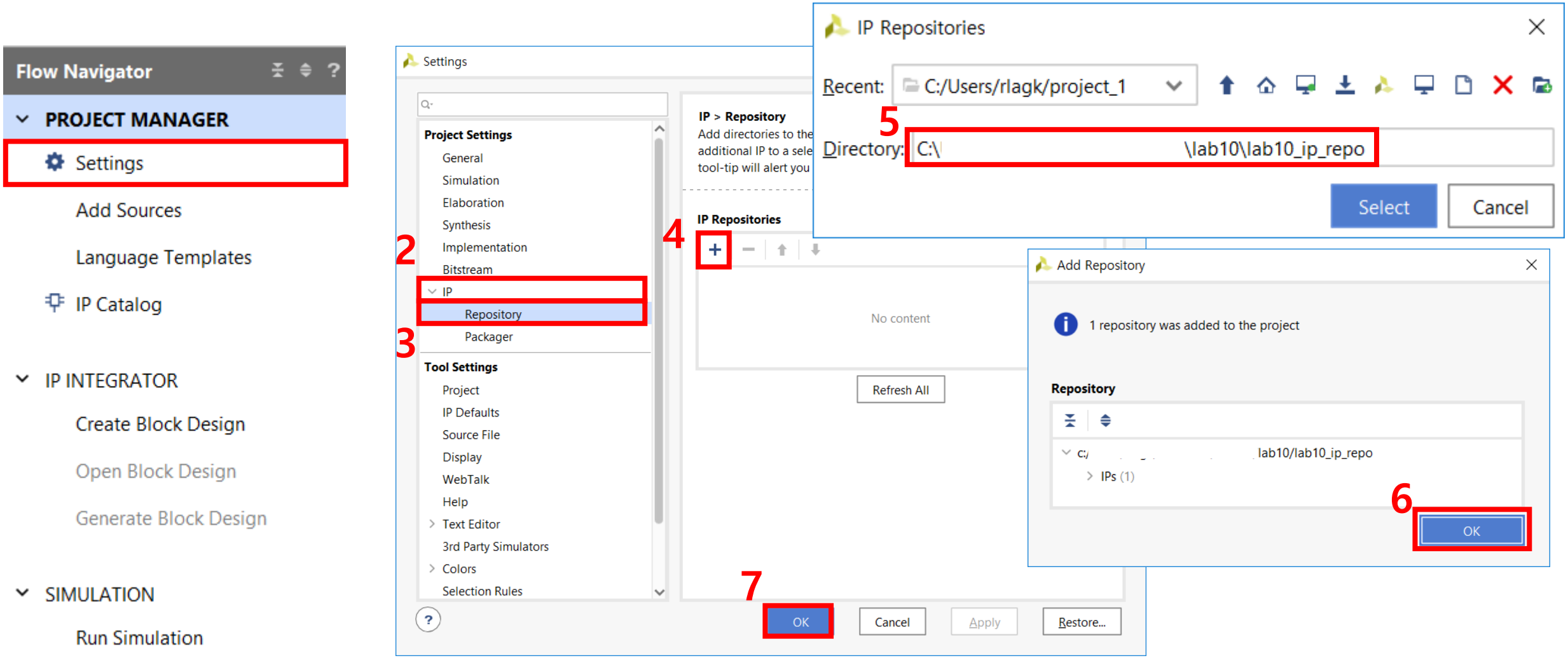
Vivado project creation

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



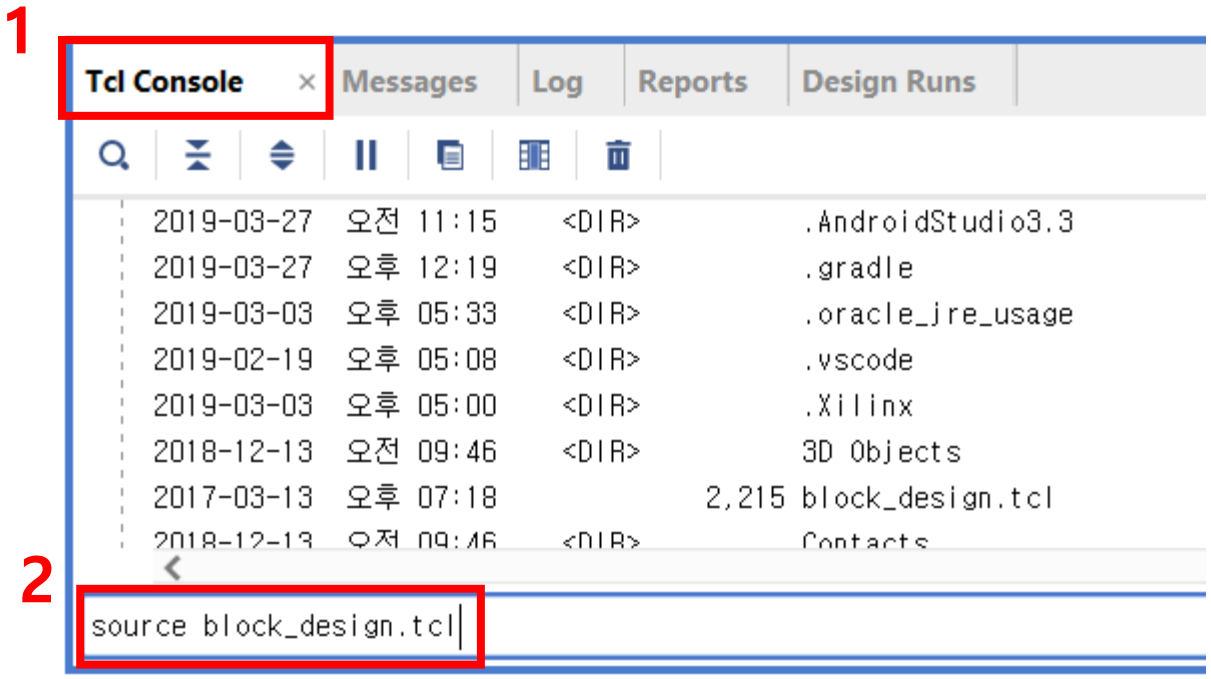
Example IP - Shifter

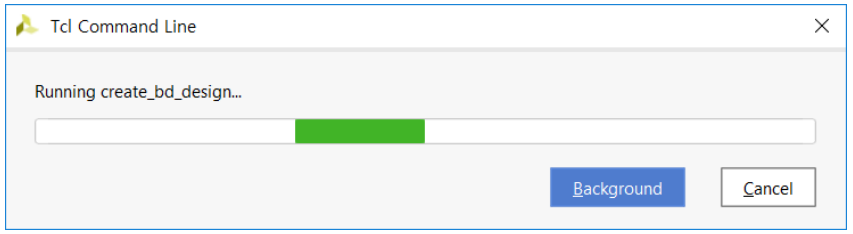
- Download an example IP repository(lab10_ip_repo.tar)



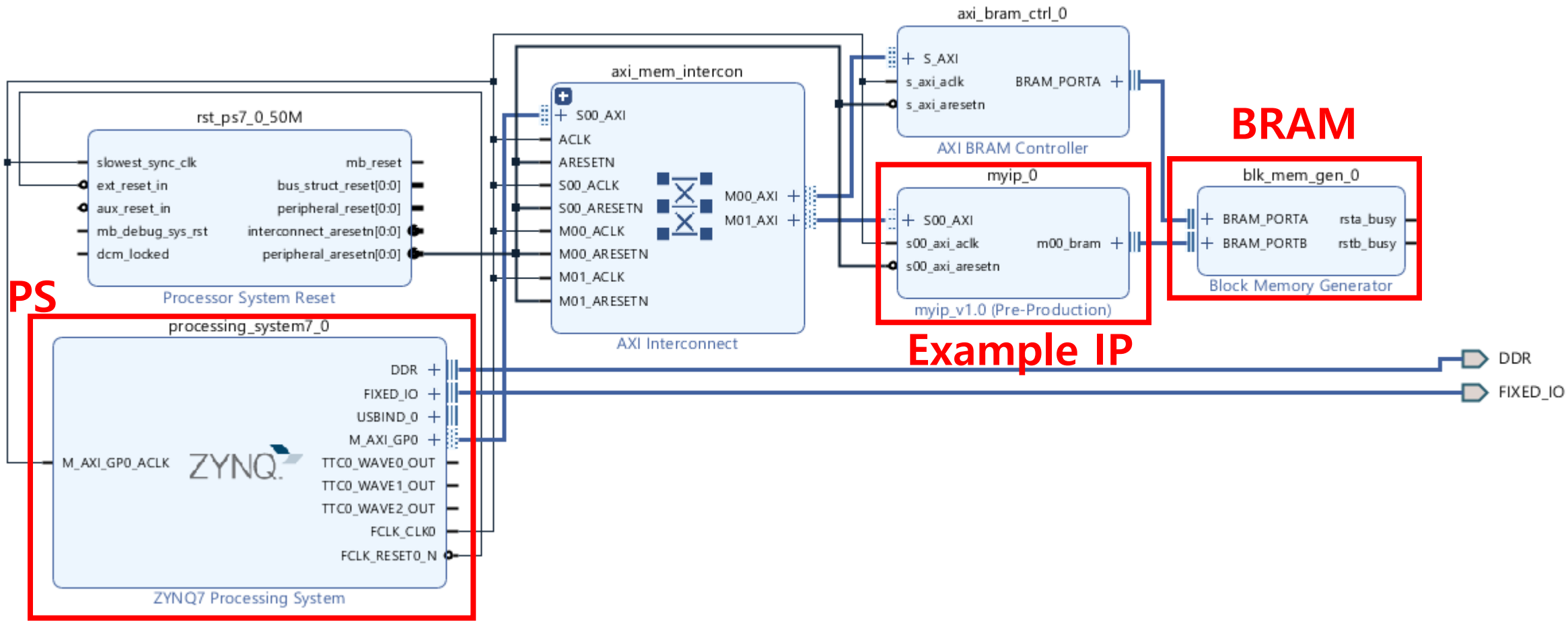
Example IP - Shifter

- 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

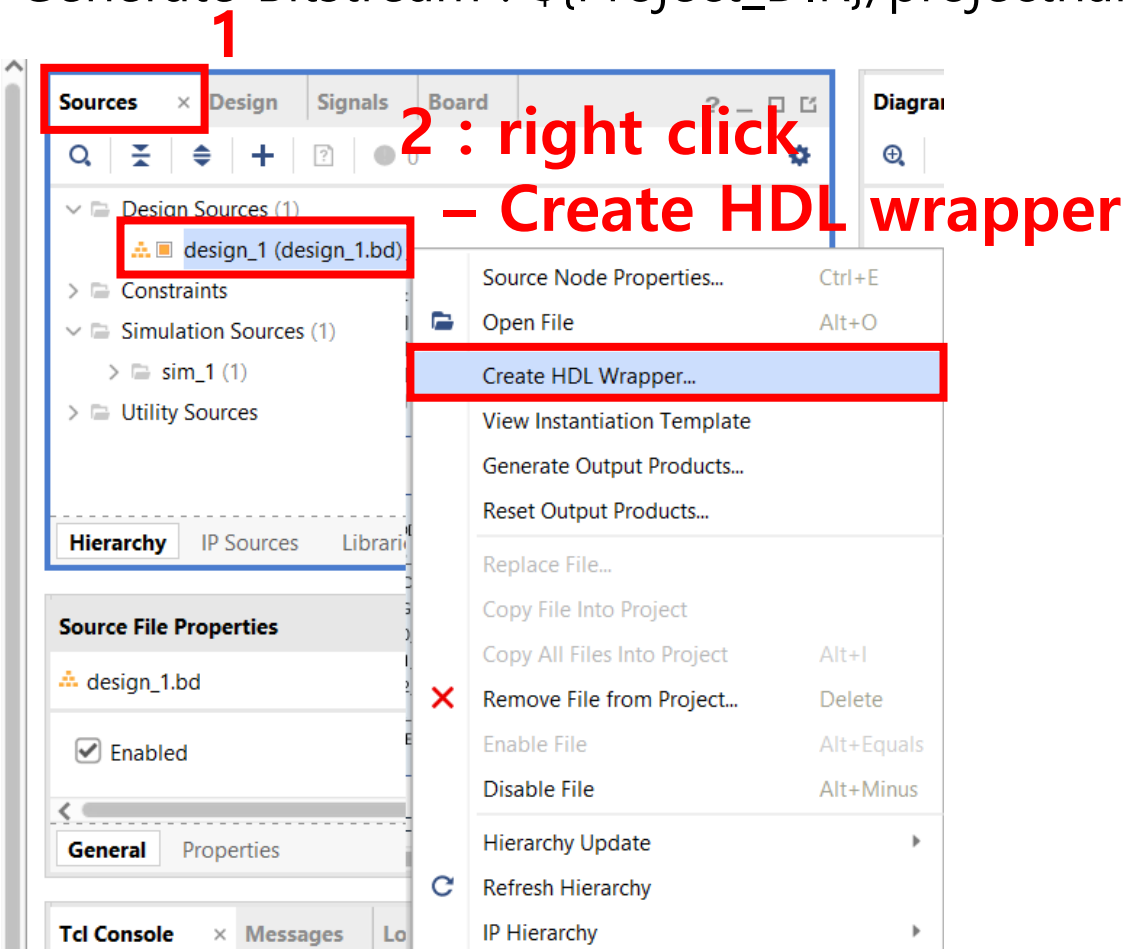
BRAM ADDR

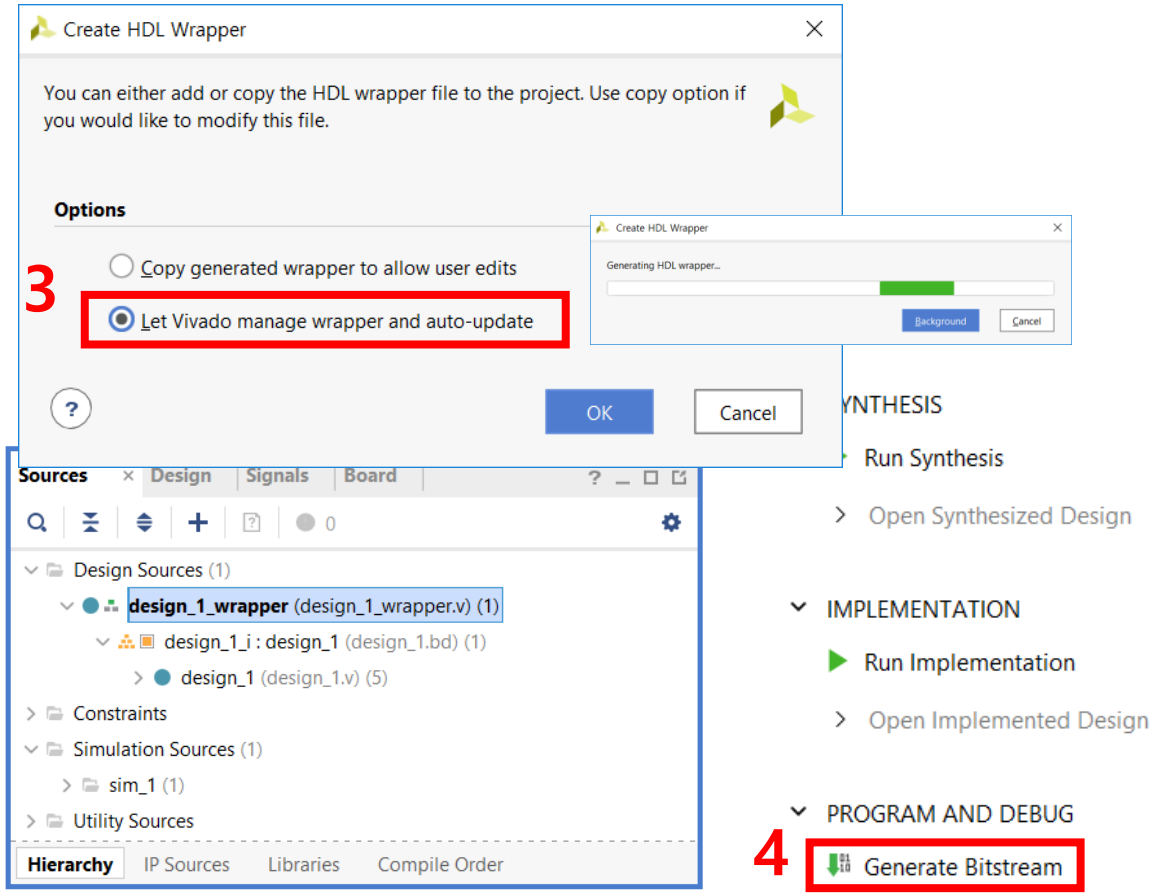
Shifter IP

```
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





[ZEDBoard Hardware User's Guide, 2014]

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

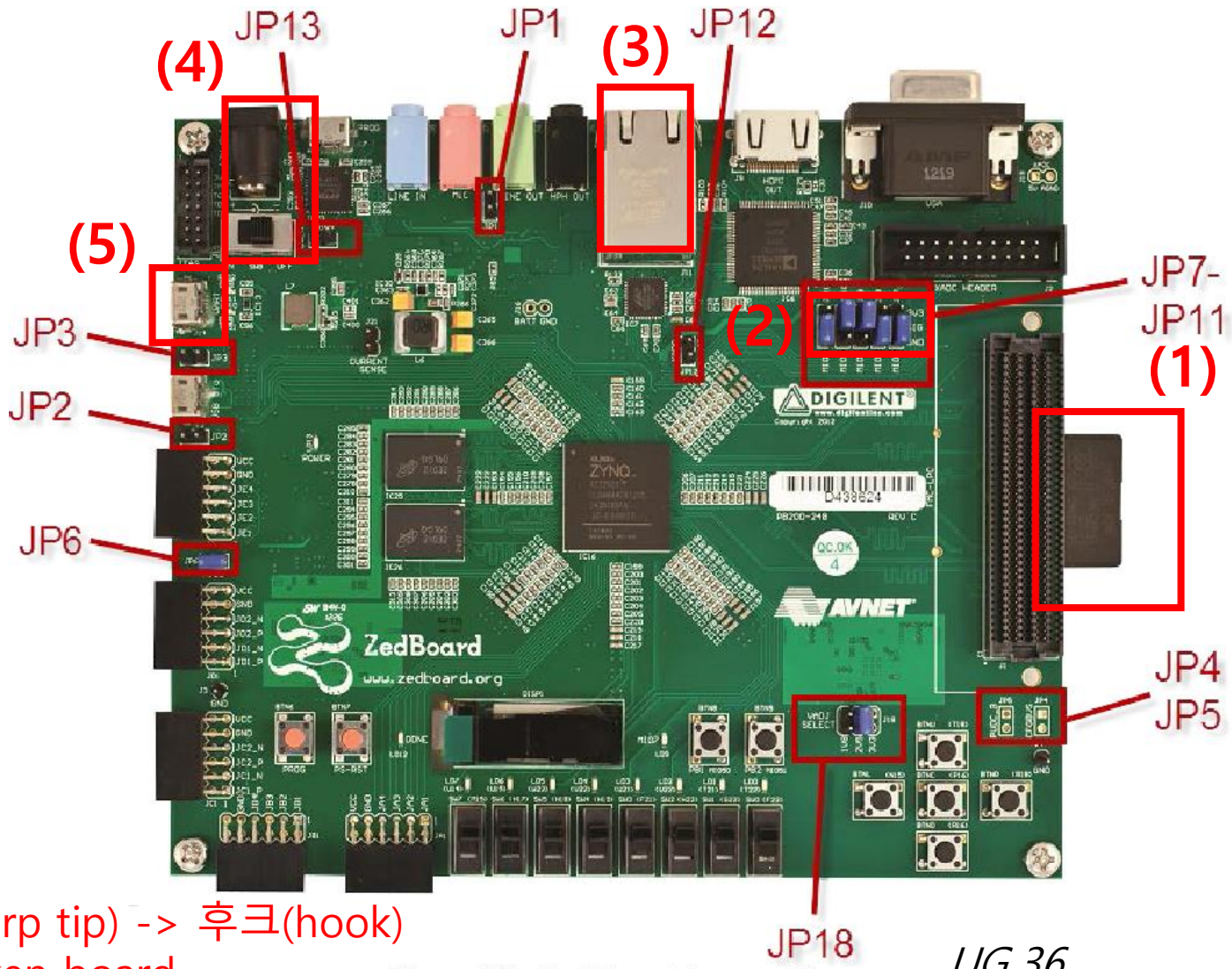


Figure 20 - ZedBoard Jumper Map

UG.36

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

Example IP - Shifter

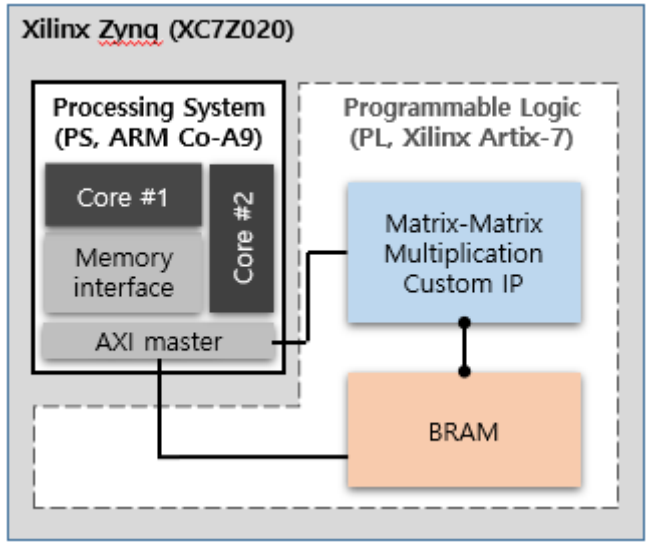
- Board Power ON
- Open terminal @ HOST
  - Login ID/PW: zed/zedzed
  - HOST$ minicom -D /dev/ttyACM0
- Run example program
  - BOARD$ cd ${LAB10_DIR}
  - BOARD$ make
  - (before un-plugging power) BOARD$ sudo poweroff
    - If you do not execute poweroff on terminal and eject the sdcard, your sdcard will not work permanantly. (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)



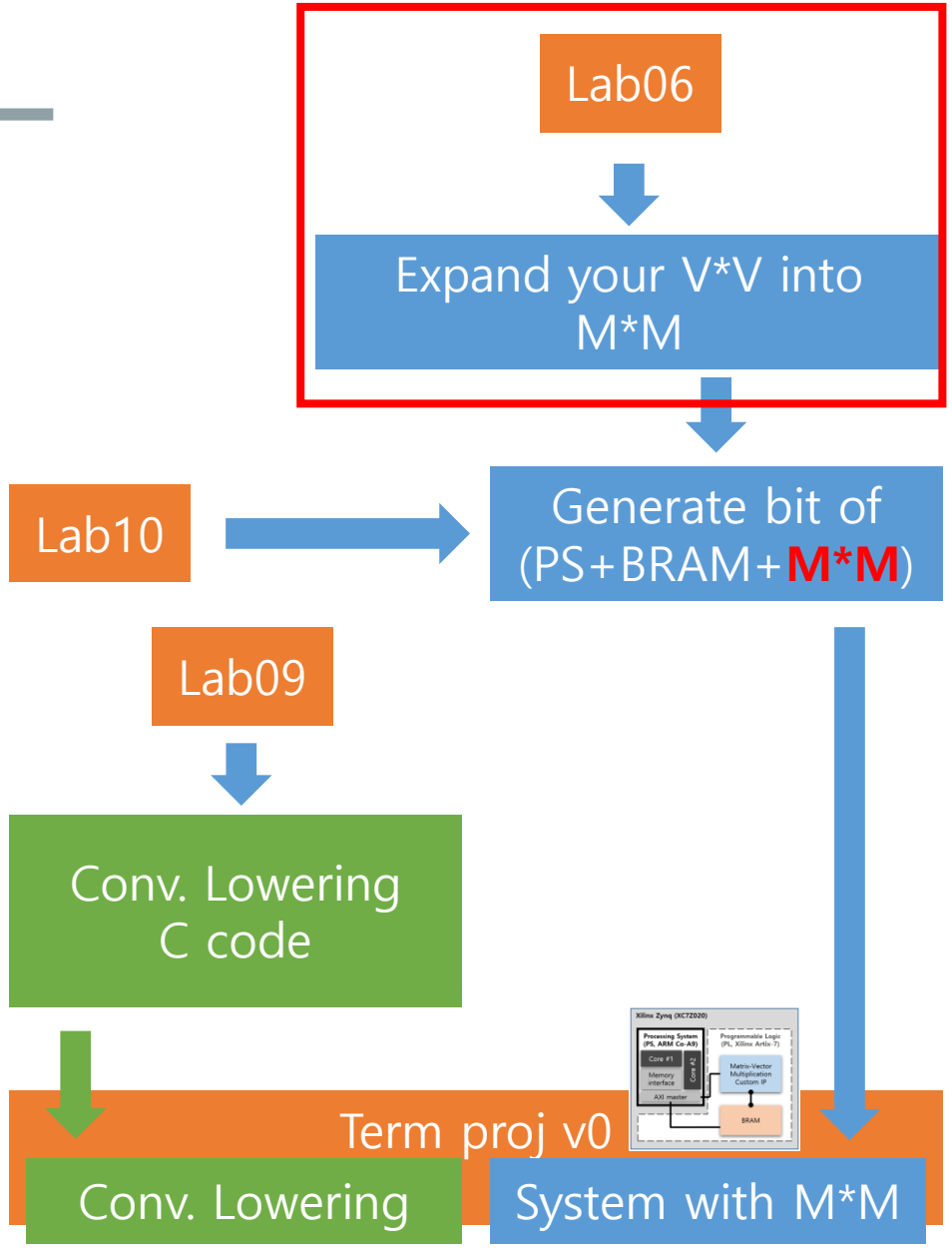
Homework

- 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

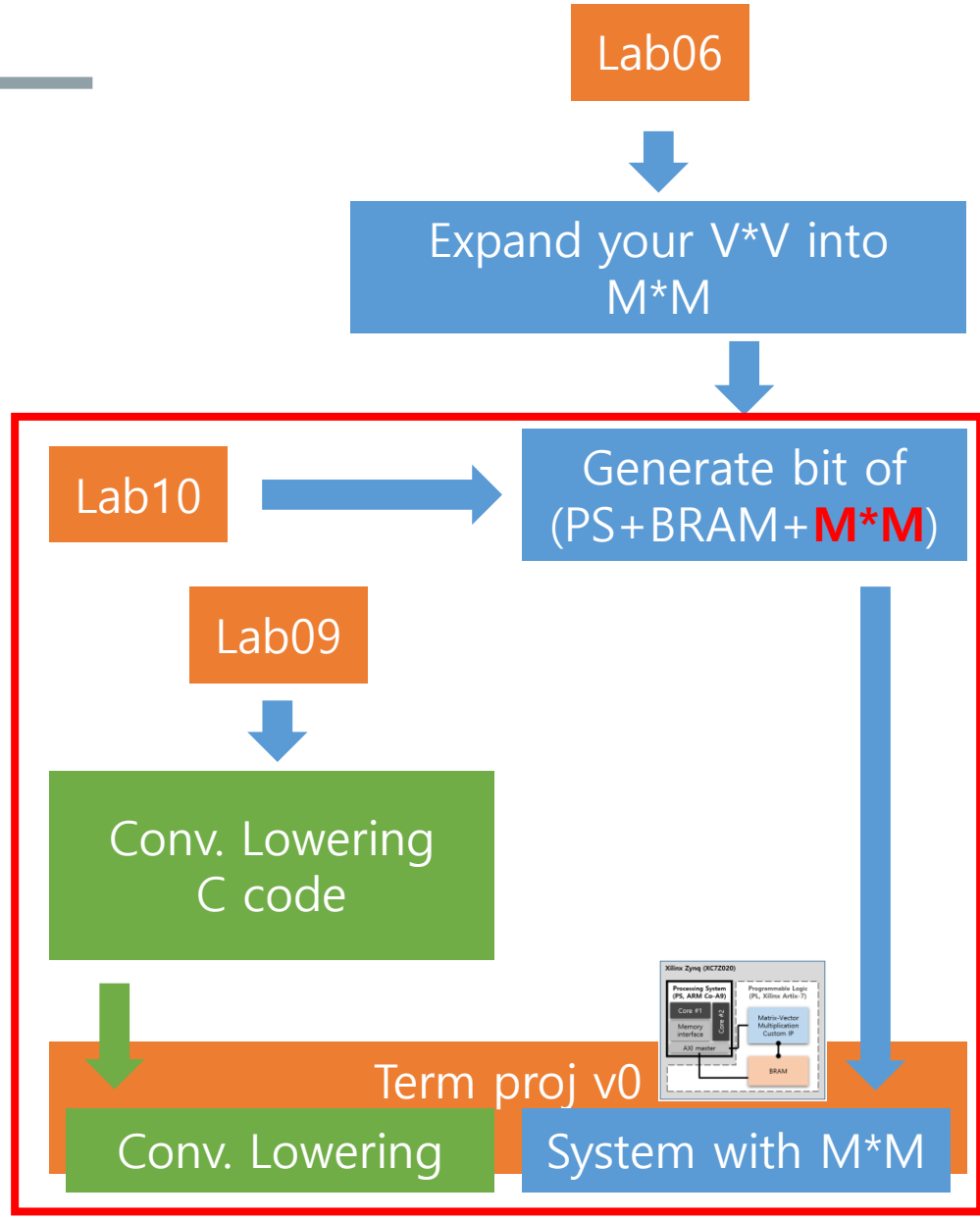


Term Project v0 중간제출

- 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

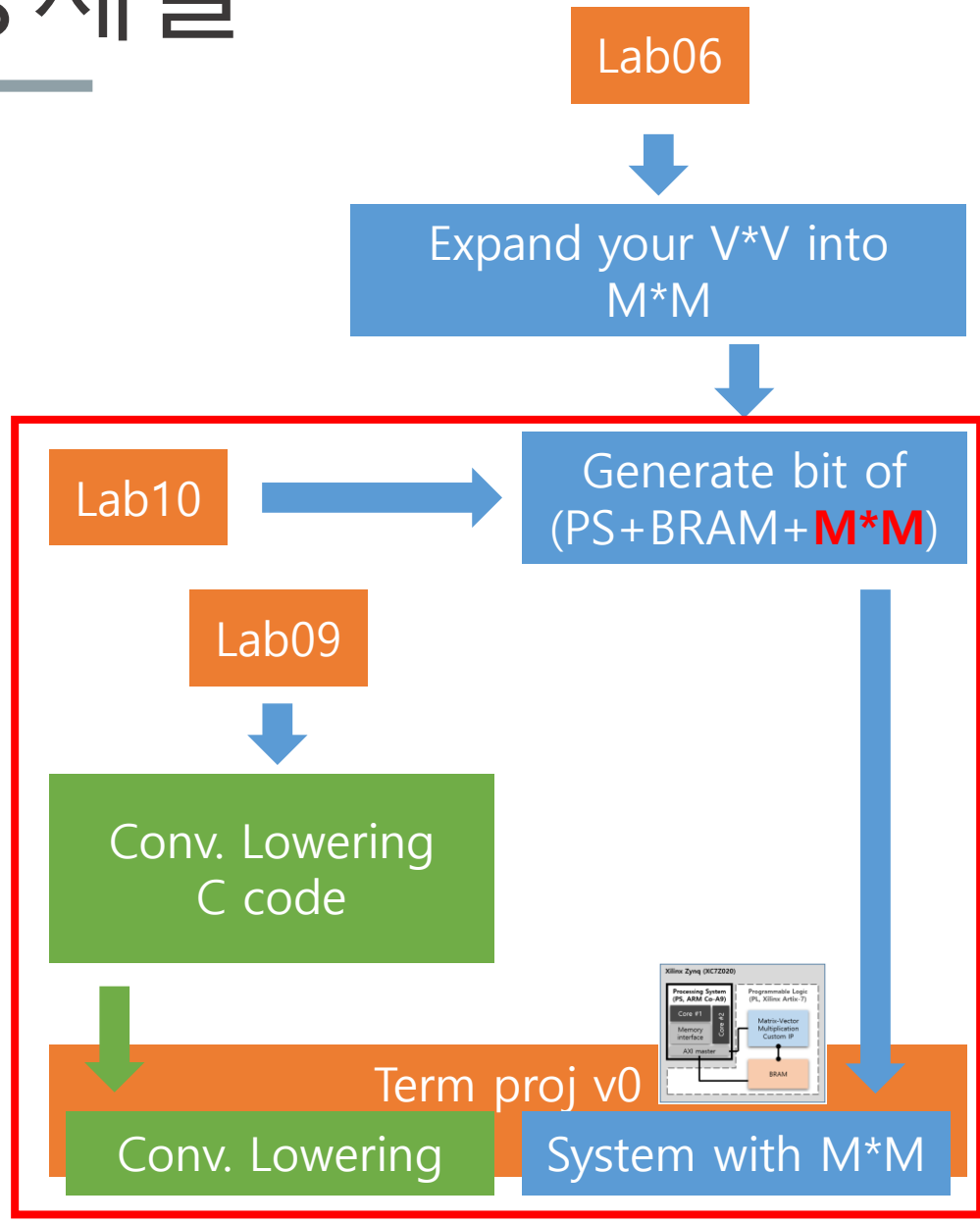
- Project Requirement
  - Convolution lowering + Custom IP system with M*M
    - Convolution lowering
      - Refer Lab09
    - Custom IP system with M*M
      - Refer Lab10 tutorial(Custom IP) & appendix(editing the custom IP)
  - Demo
    - Zedboard



Term Project Optimization 최종제출

- Optional Project
- Convolution lowering + Custom IP system with M*M + Optimization
  - DMA
  - Zero-Skipping
  - Quantization
  - …etc

  Demo
  - Zedboard



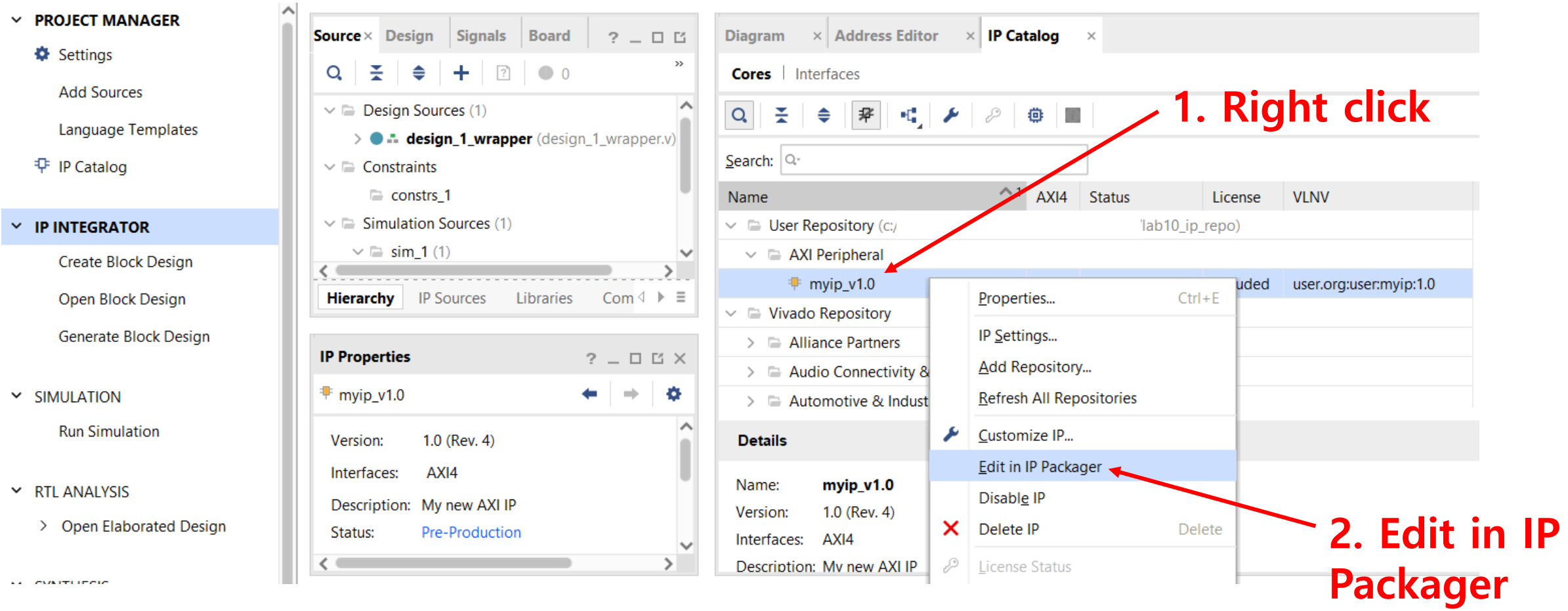
Term Project v0 + Optimization 최종제출

- 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

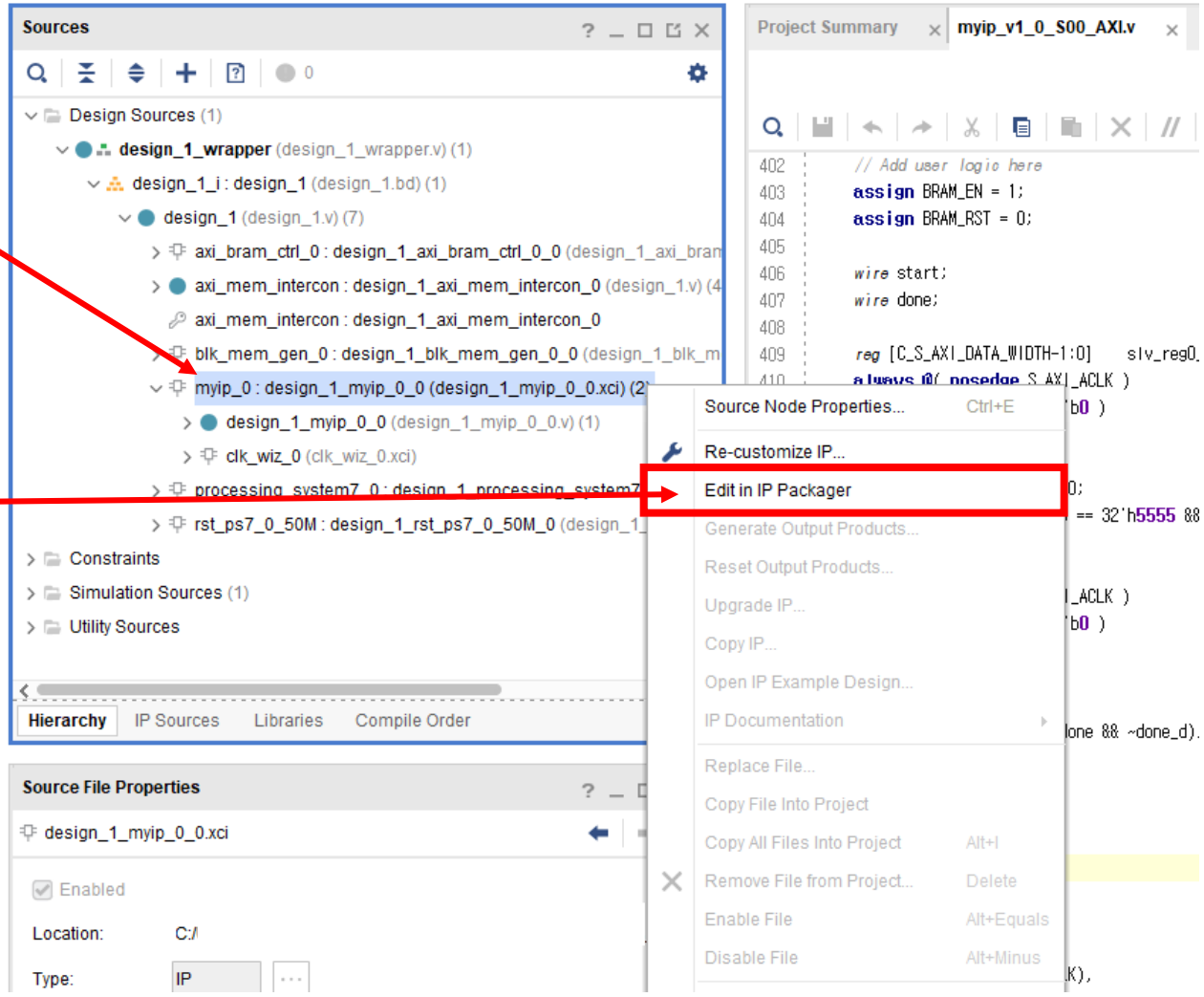


Editing Custom IP

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

1. Right click on custom IP

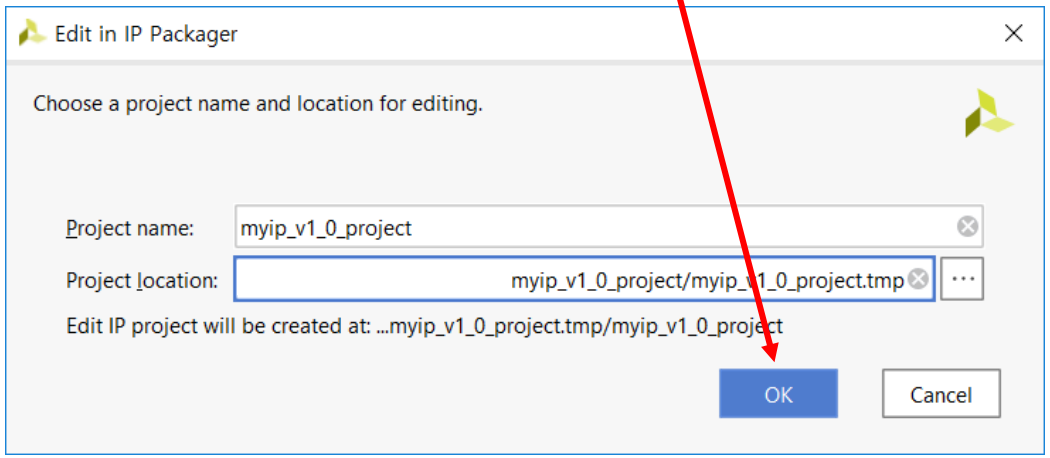
2. Edit in IP Packager



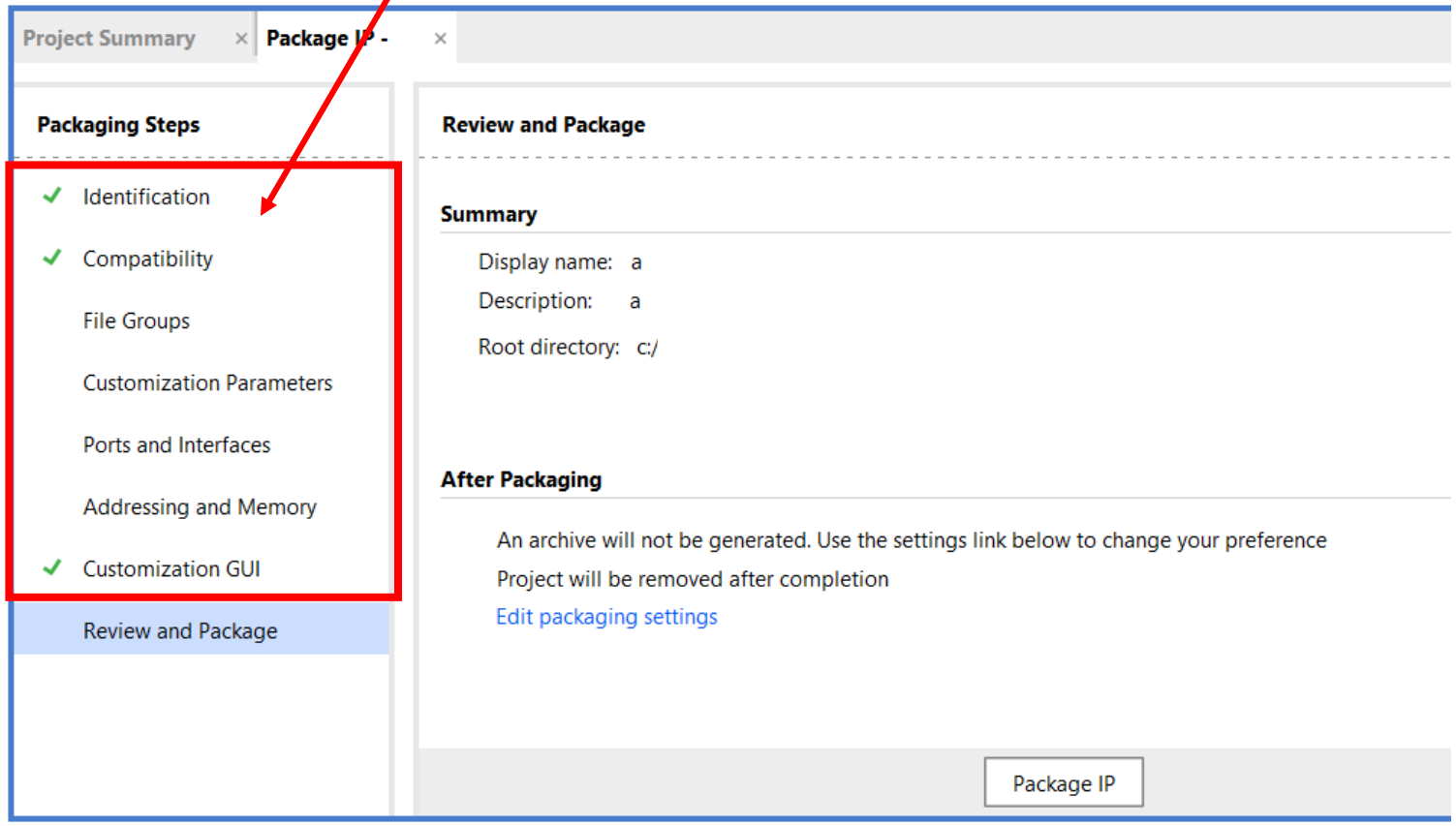
Packaging Custom IP

- Package IP -> Configurations

1

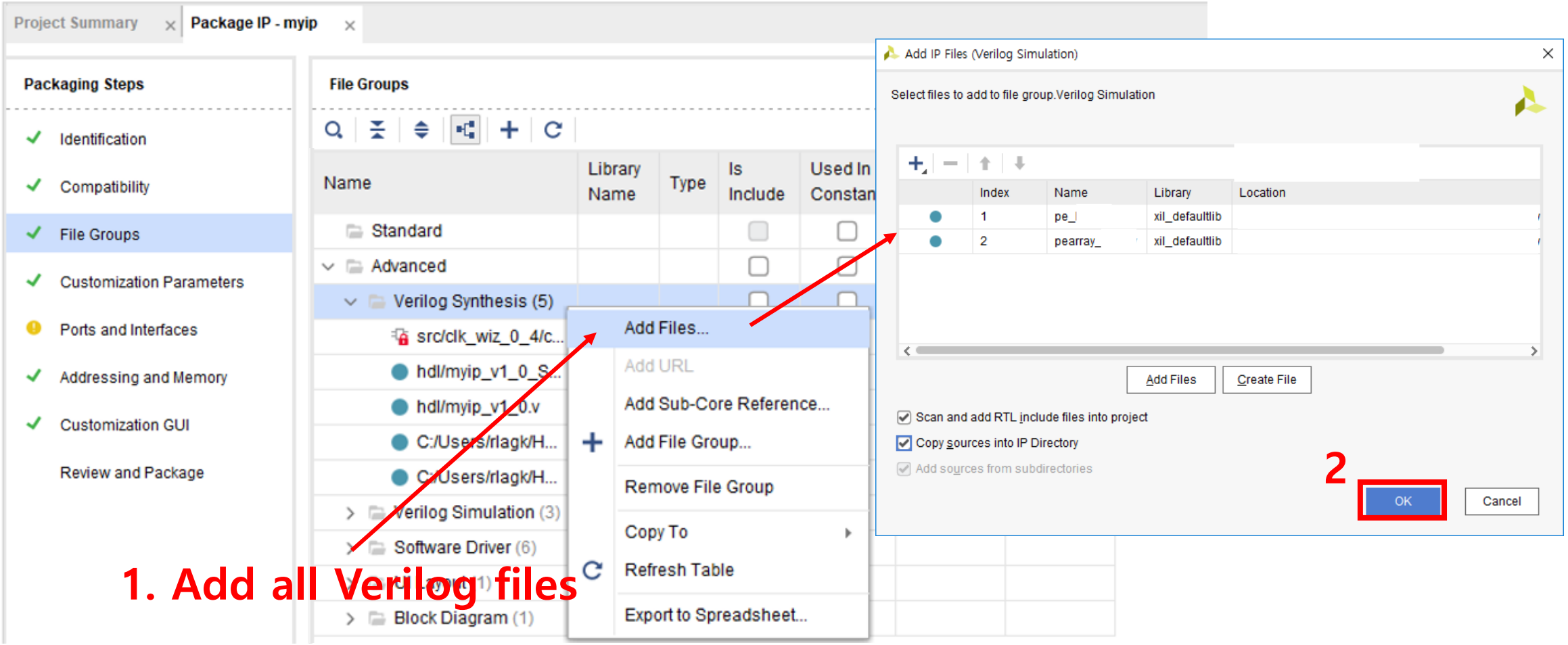


2. Check IP Configurations



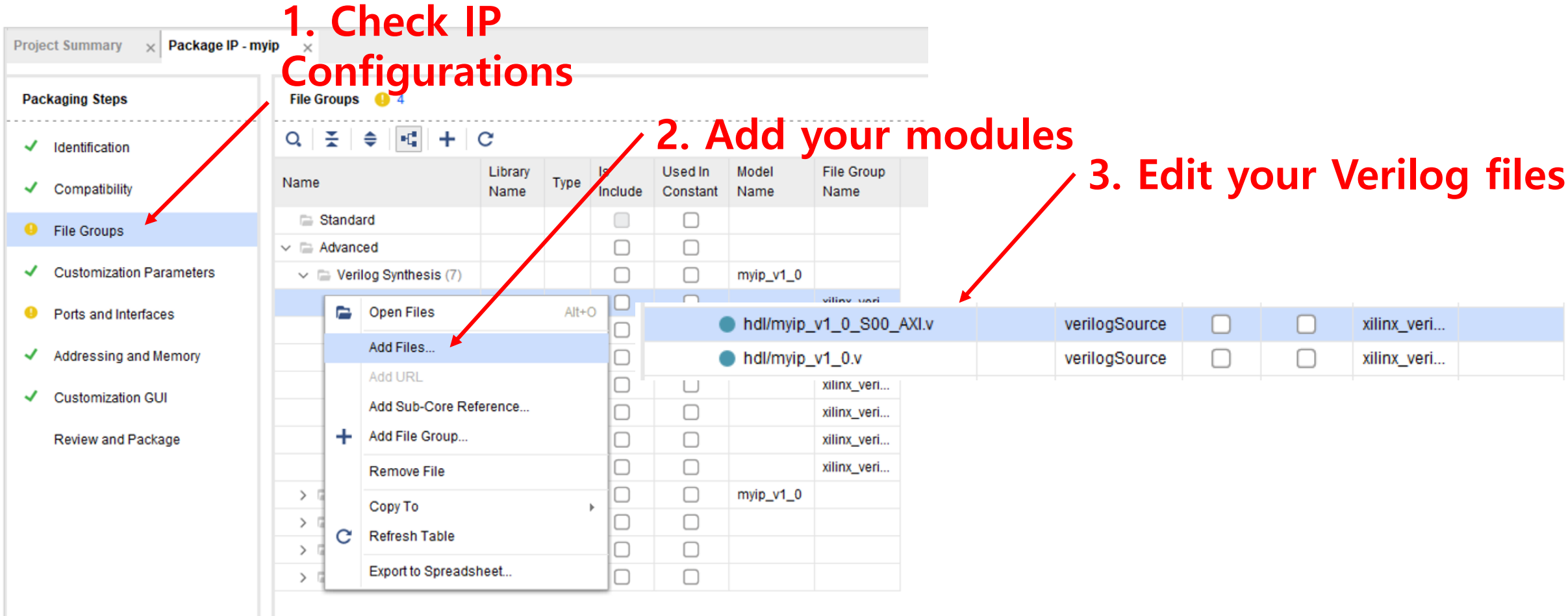
Checking IP Configurations

- 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



Checking IP Configurations

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

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

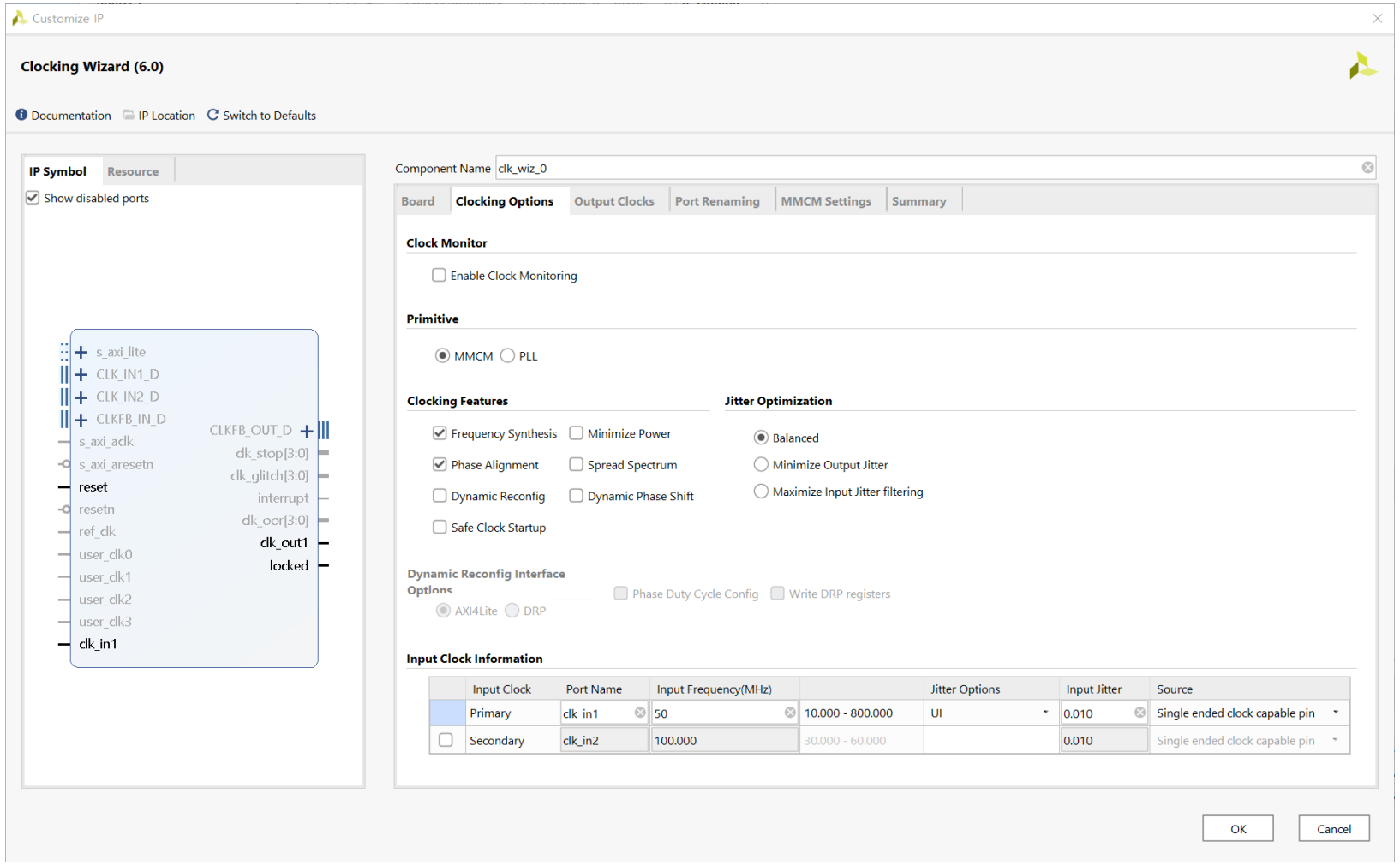
2. Revise myip_v1_0_S00_AXI.v & myip_v1_0.v codes
- especially ports, parameters, modules you needed

myip_v1_0_S00_AXI.v *

```
398
399     // Add user logic here
400     assign BRAM_EN = 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_pearray #(
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_WRDATA(BRAM_WRDATA),
435         .BRAM_WE(BRAM_WE),
436         .BRAM_CLK(BRAM_CLK),
437         .BRAM_RDDATA(BRAM_RDDATA)
438     );
439
440     // User logic ends
441
442 endmodule
443
```

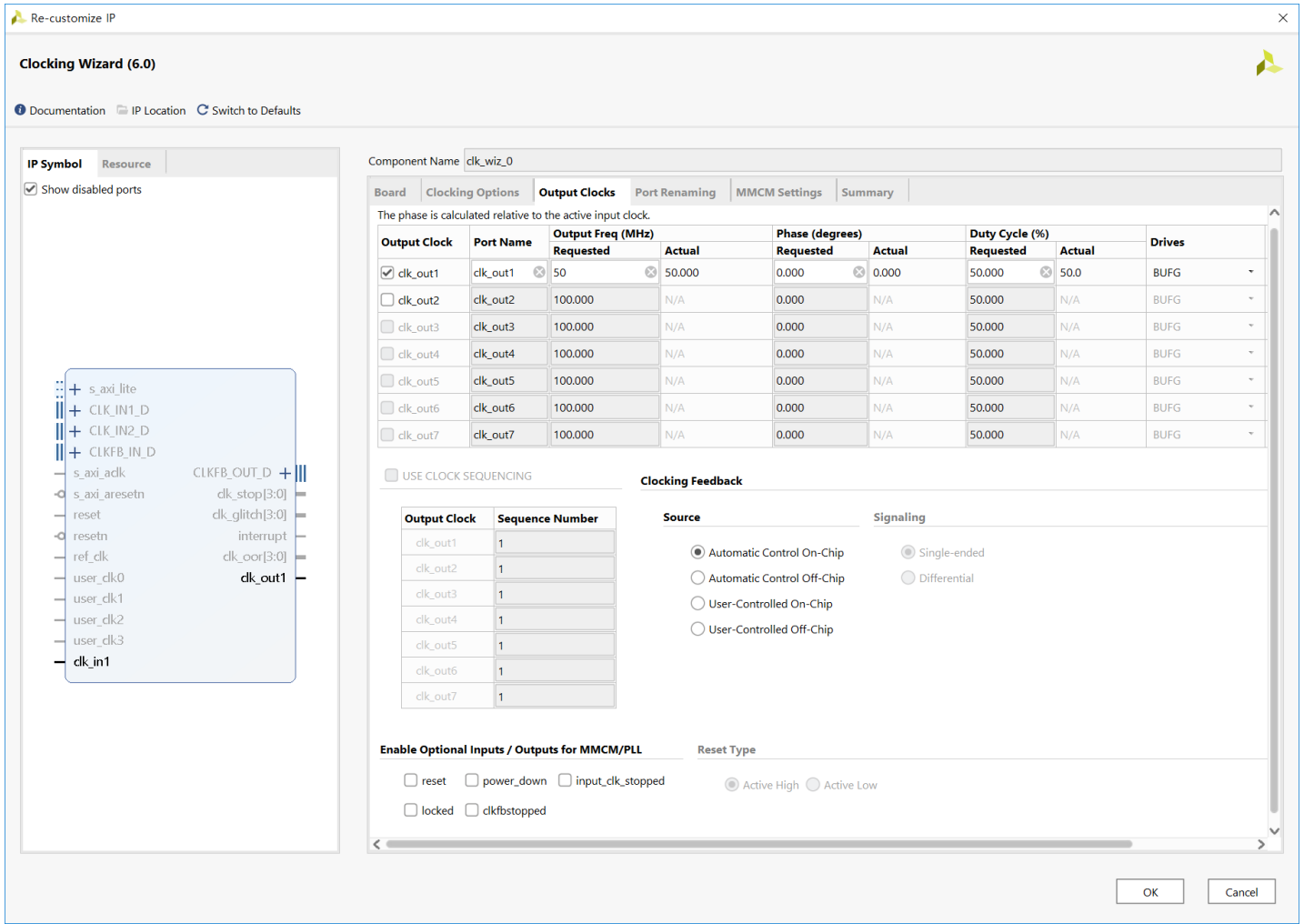
Add clocking wizard IP

- Step2: File Groups (Add clocking wizard)



Add clocking wizard IP

- Step2: File Groups (Add clocking wizard)



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

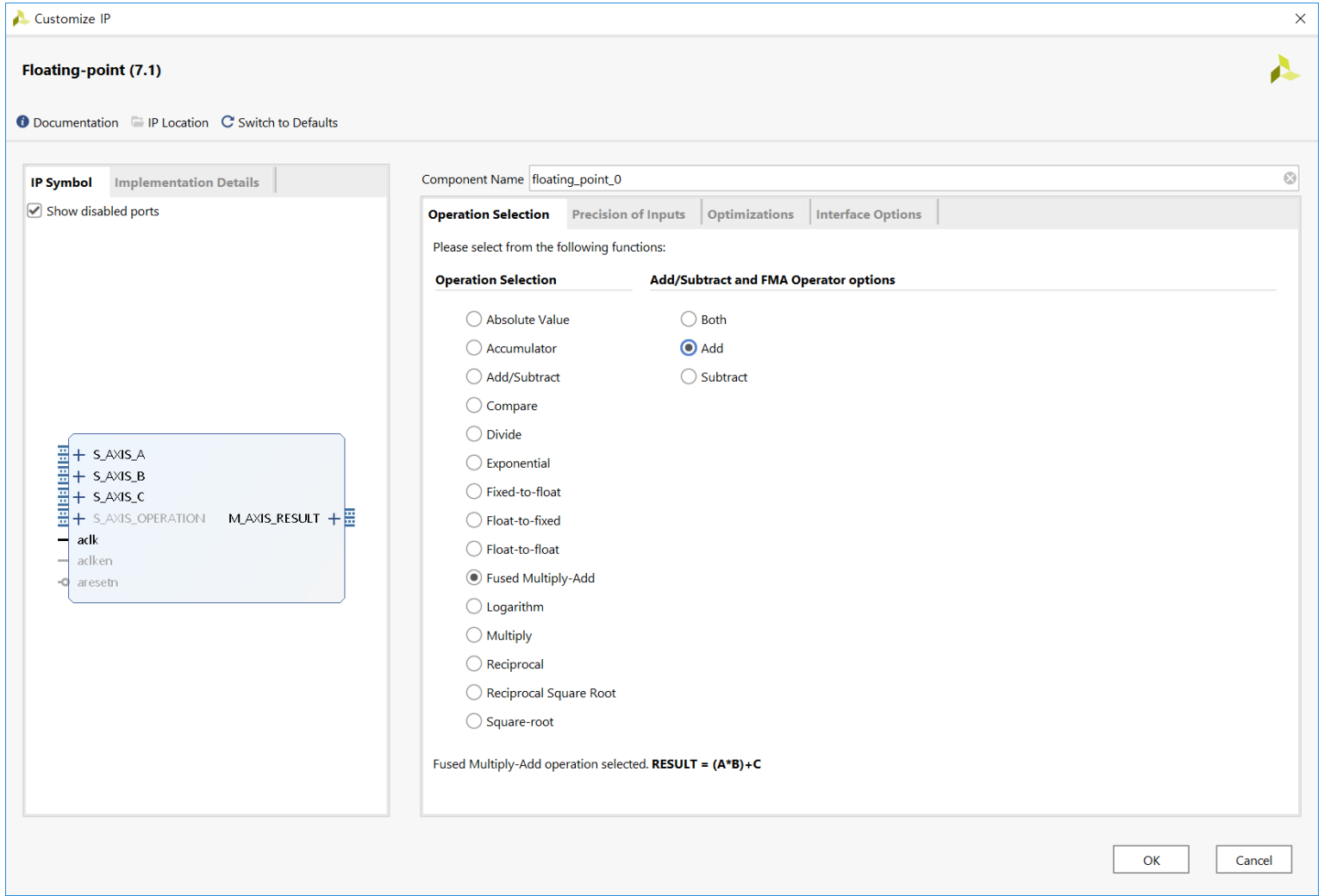
```
// Users to add ports here
output wire [31:0] BRAM_ADDR,
output wire BRAM_CLK,
output wire [31:0] BRAM_WRDATA,
input wire [31:0] BRAM_RDDATA,
output wire [3:0] BRAM_WE,
output wire BRAM_EN,
output wire BRAM_RST,
```

```
clk_wiz_0 u_clk (.clk_out1(BRAM_CLK), .clk_in1(aclk));
```

```
my_pe #(.L_RAM_SIZE(L_RAM_SIZE)
) u_pe (
    .aclk(aclk),
    .aresetn(aresetn),
    .ain(ain),
    .din(din),
    .addr(addr),
    .we(we[i]),
    .valid(valid),
    .dvalid(dvalid[i]),
    .dout(dout[i])
);
```

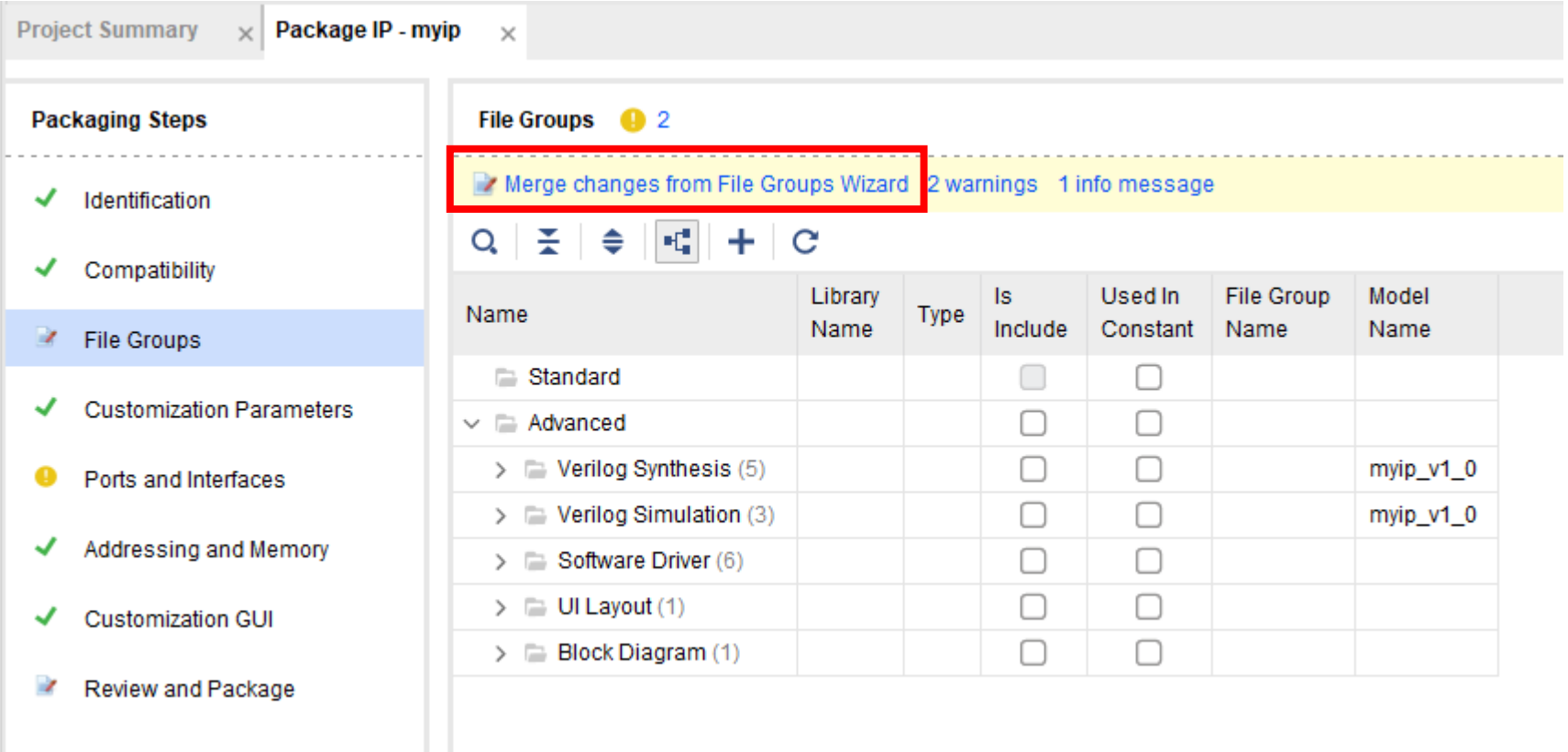
Add floating point IP

- Step2: File Groups (Add floating point IP)



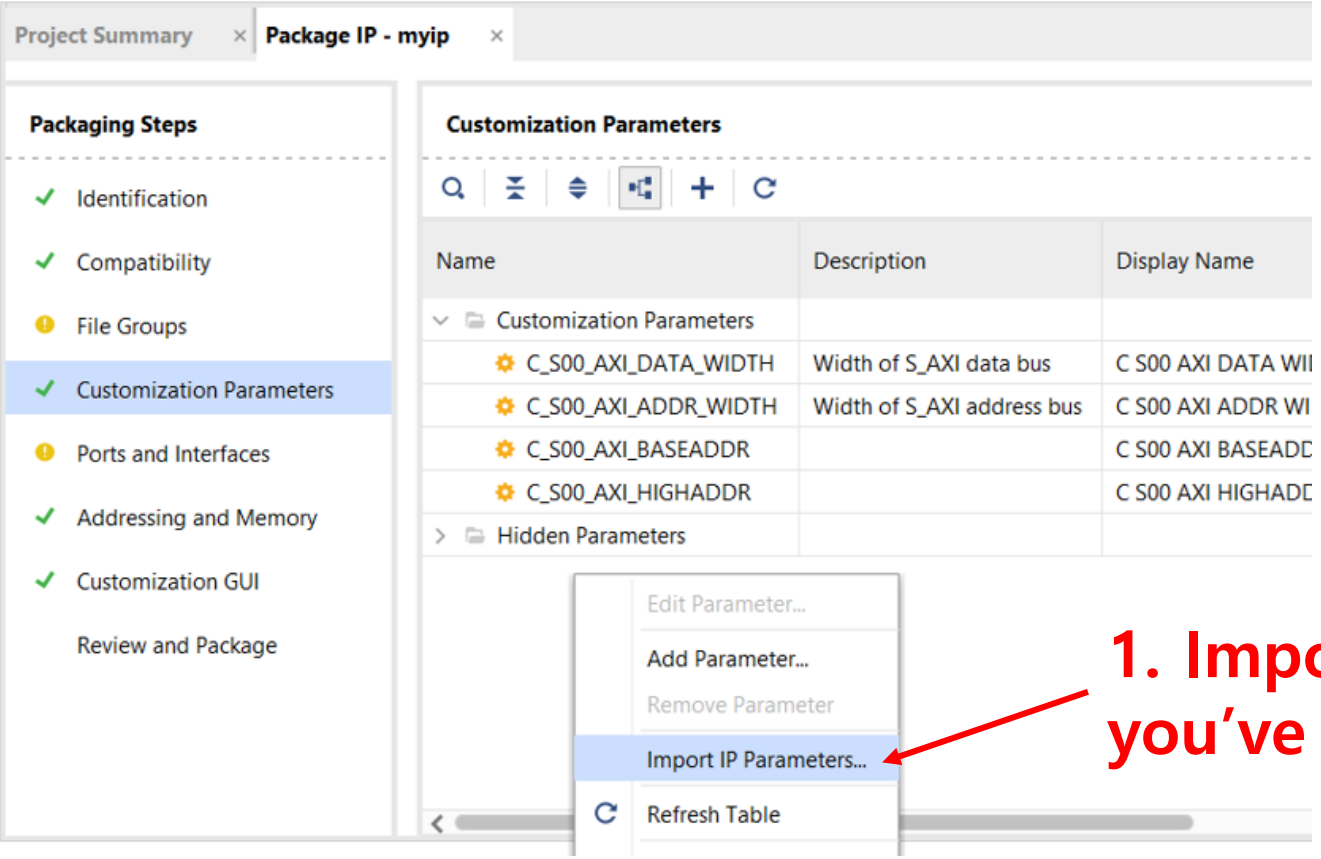
Checking IP Configurations

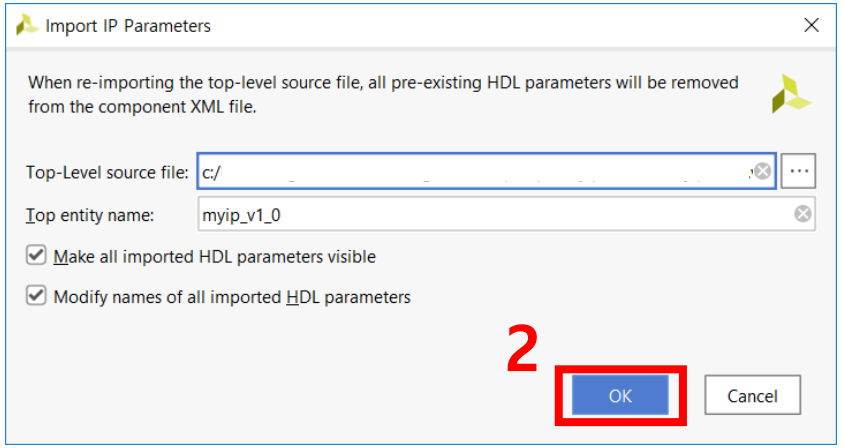
- Changing IP files
- File Groups -> Merge changes



Checking IP Configurations

- Parameter configuration
- Customization Parameters -> import IP parameters

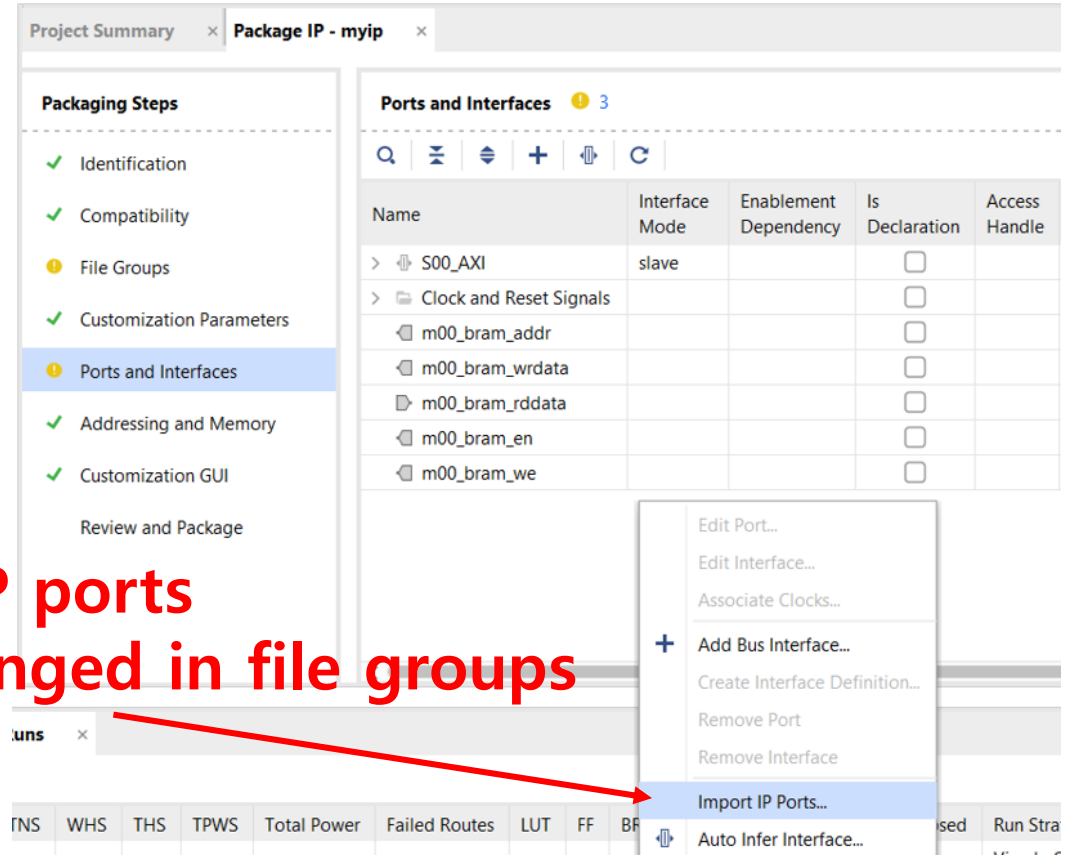




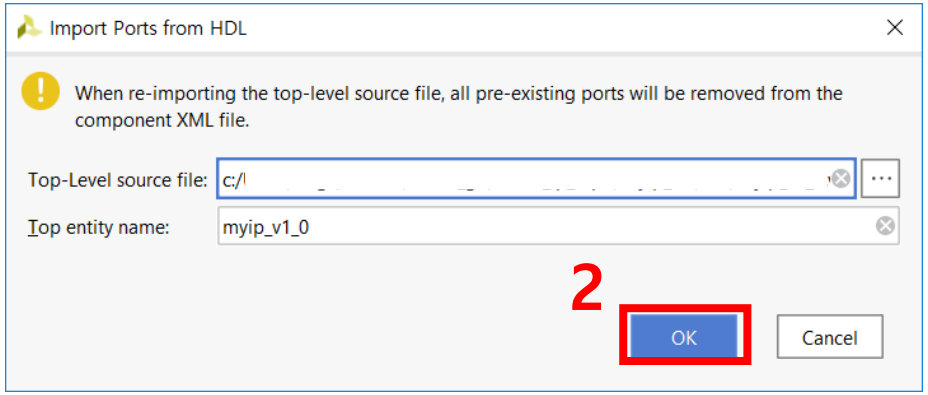
1. Import IP parameters you've changed in file groups

Checking IP Configurations

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

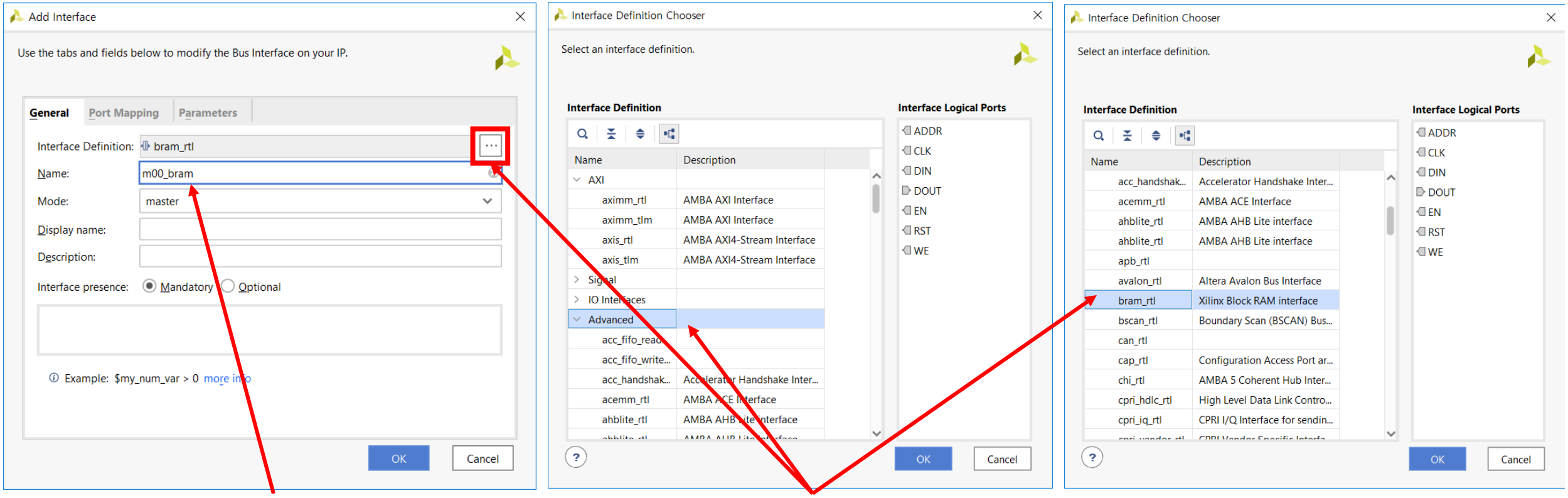


1. Import IP ports you've changed in file groups



Checking IP Configurations

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

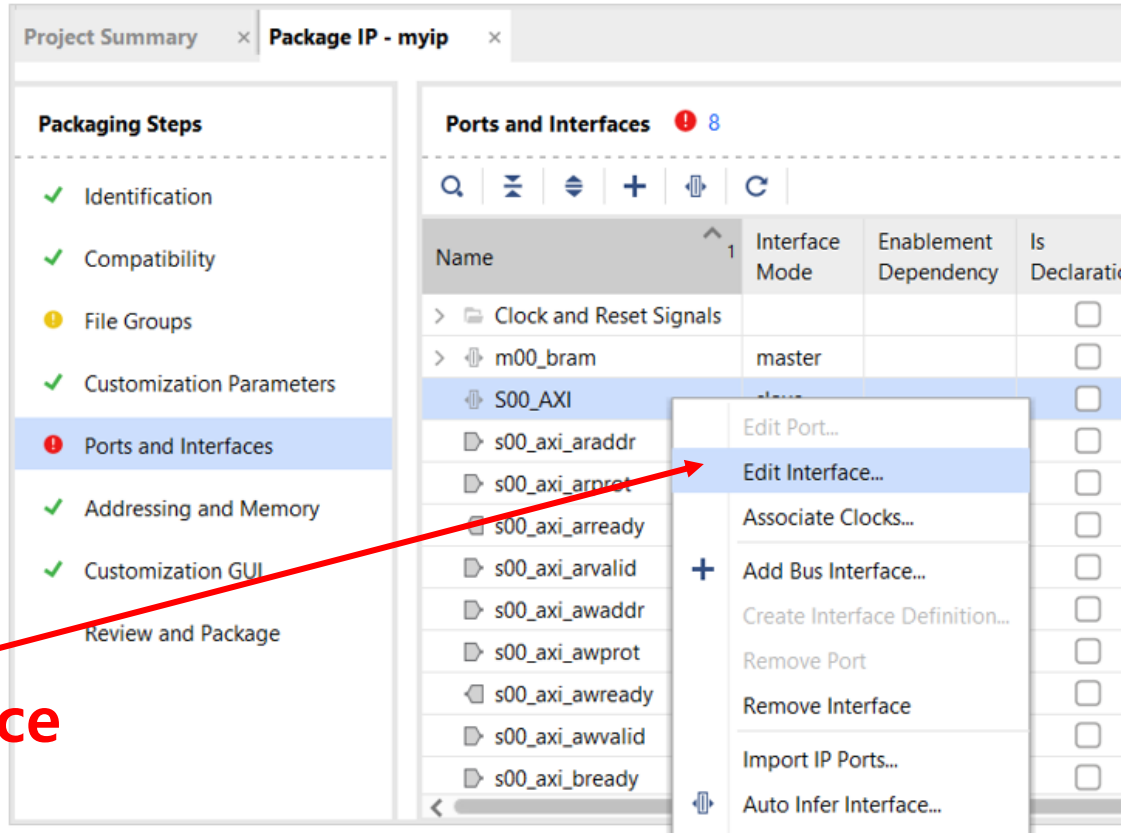


1. Set your port name

2. Advanced – bram_rtl

Checking IP Configurations

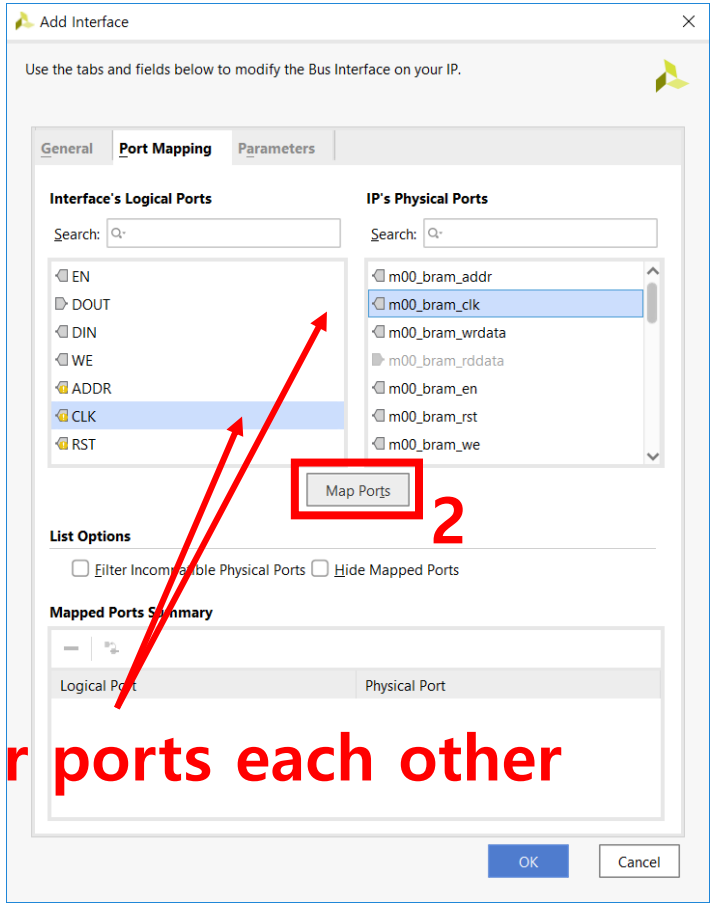
- Port configuration
- Grouping ports – case2 : when the group exists
- ex) S00_AXI



1. Edit Interface

Checking IP Configurations

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



1. Match your ports each other

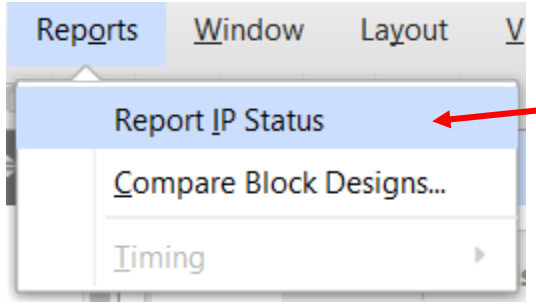
Mapped Ports Summary

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

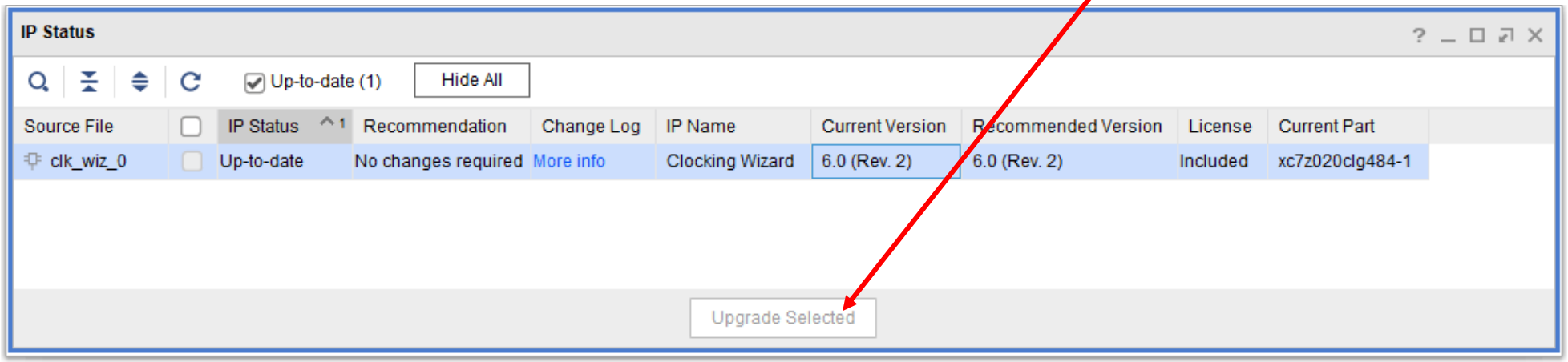
Upgrading Custom IP

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



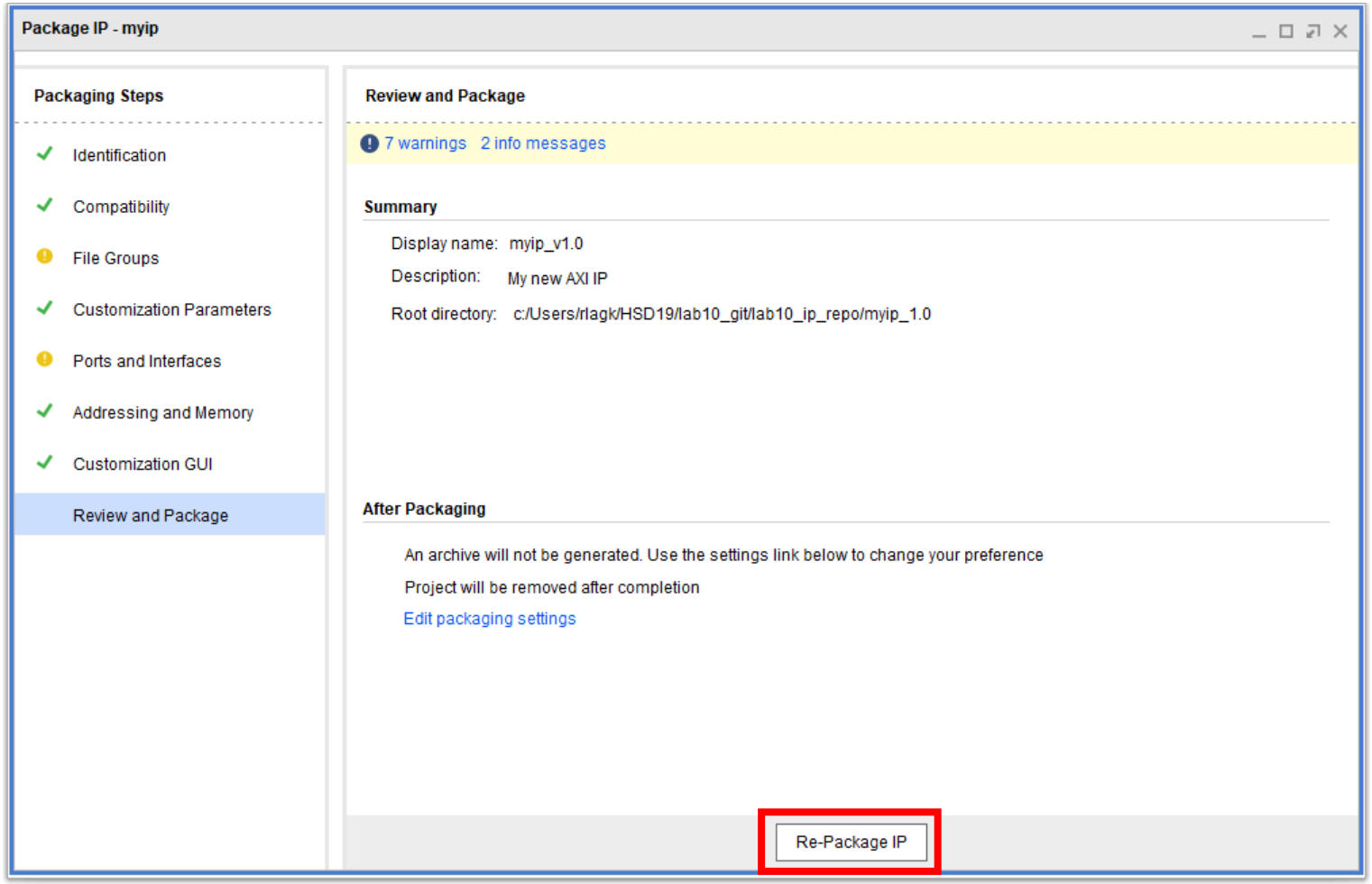
1

2



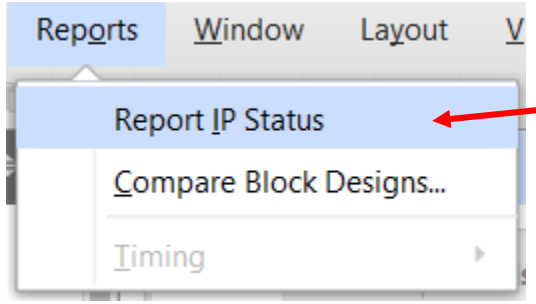
Checking IP Configurations

- Re-Package IP



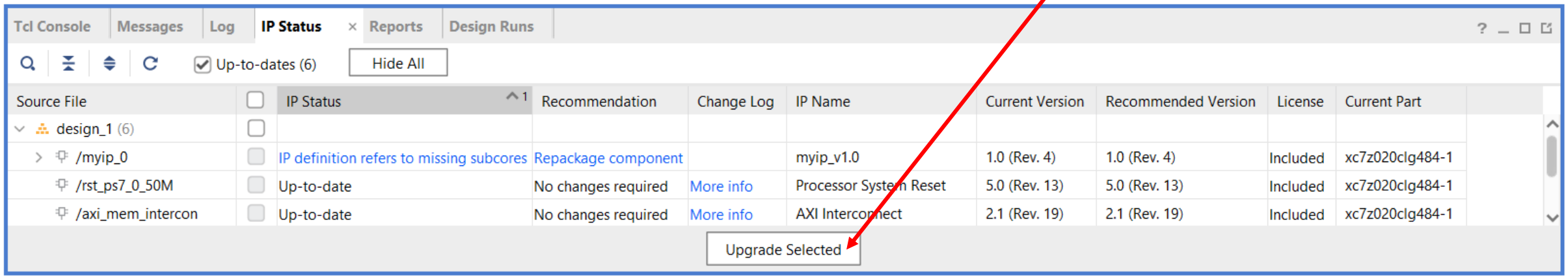
Upgrading Custom IP

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



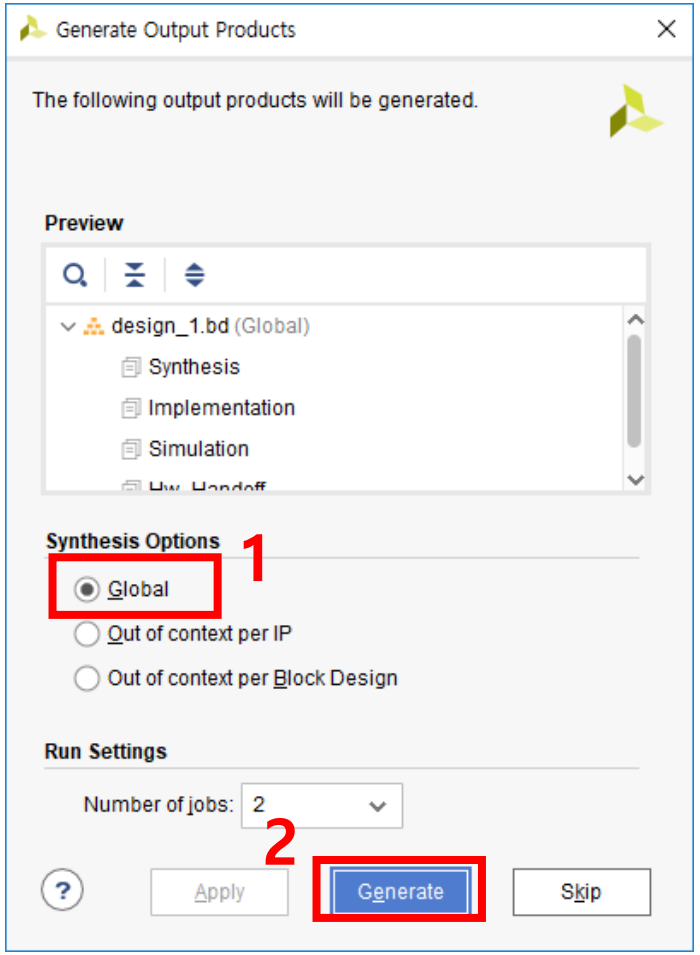
1

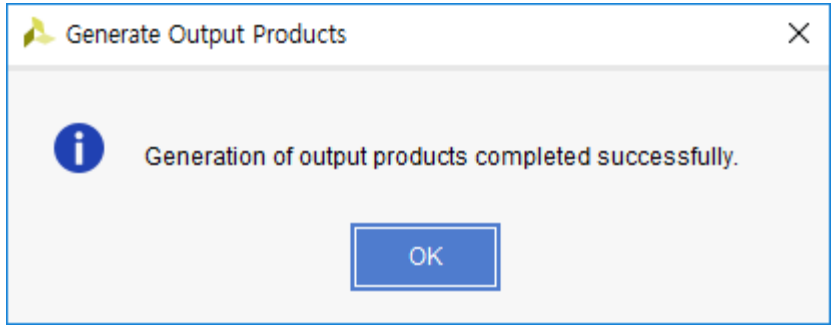
2

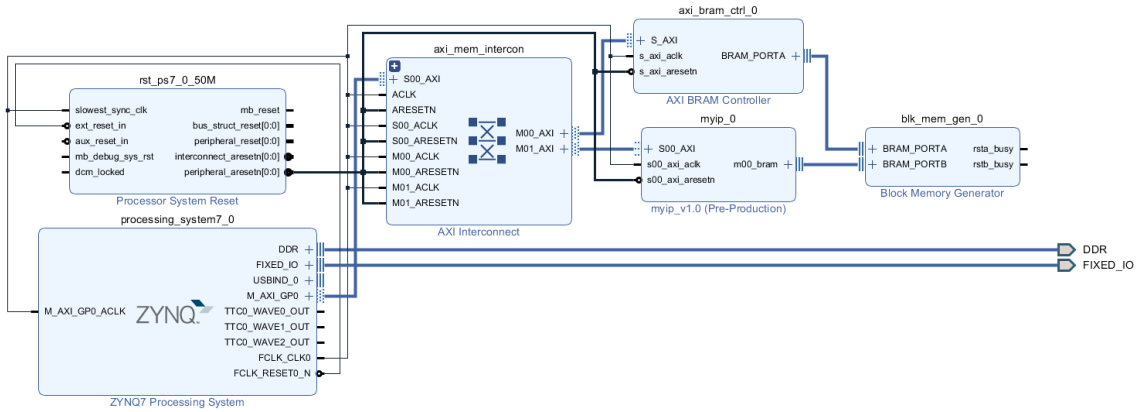


Upgrading Custom IP

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

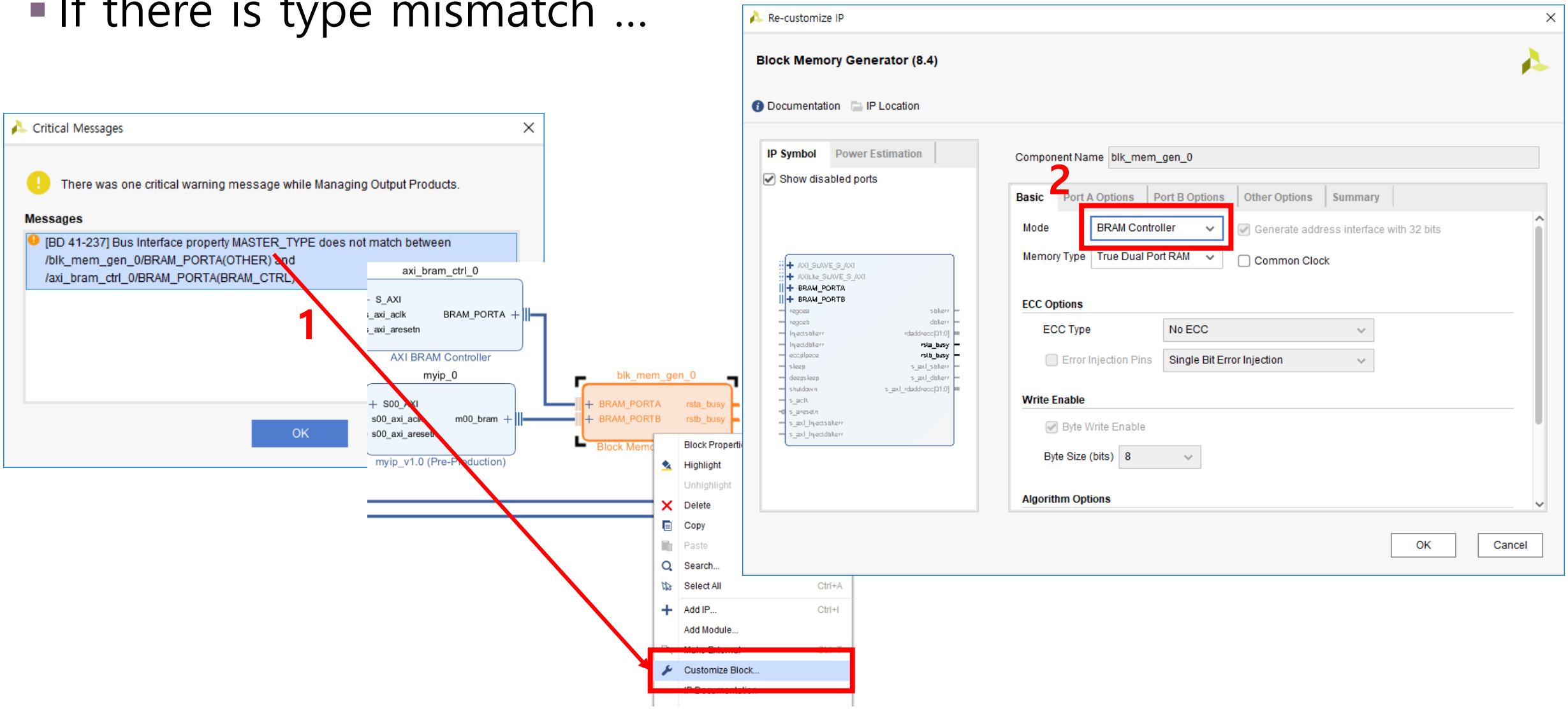






Type mismatch(General IP)

- If there is type mismatch …



Type mismatch(Custom IP)

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