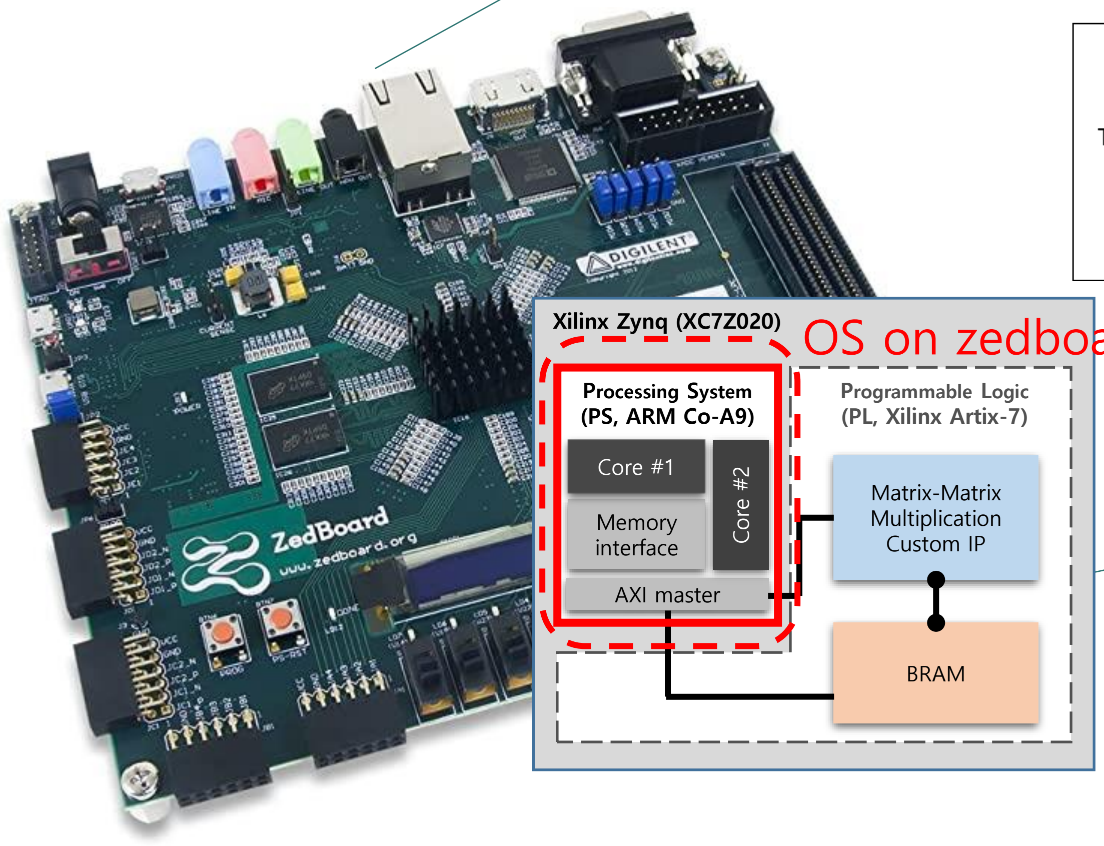
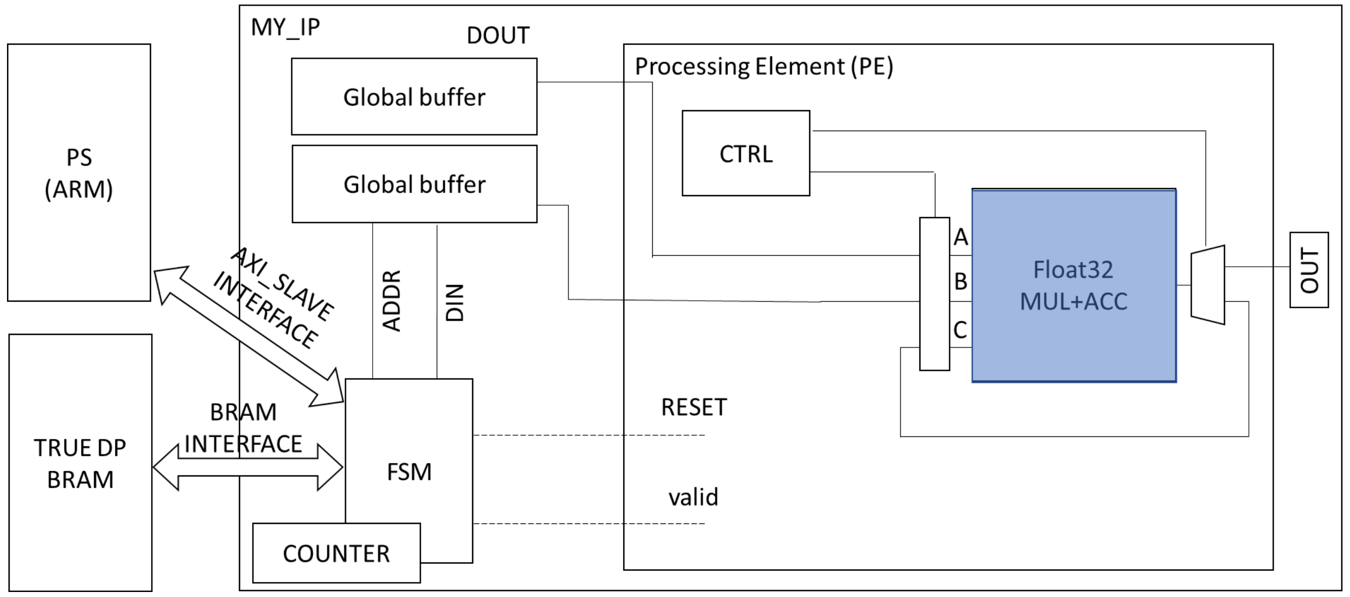
Practice 8

- OS + FPGA System

Computing Memory Architecture Lab.

Final Project Overview: Matrix Multiplication IP


Xilinx Zynq (XC7Z020)
OS on zedboard system
Processing System (PS, ARM Co-A9)
Core #1
Core #2
Memory interface
AXI master
Programmable Logic (PL, Xilinx Artix-7)
Matrix-Matrix Multiplication Custom IP
BRAM
ZedBoard
DIGILENT



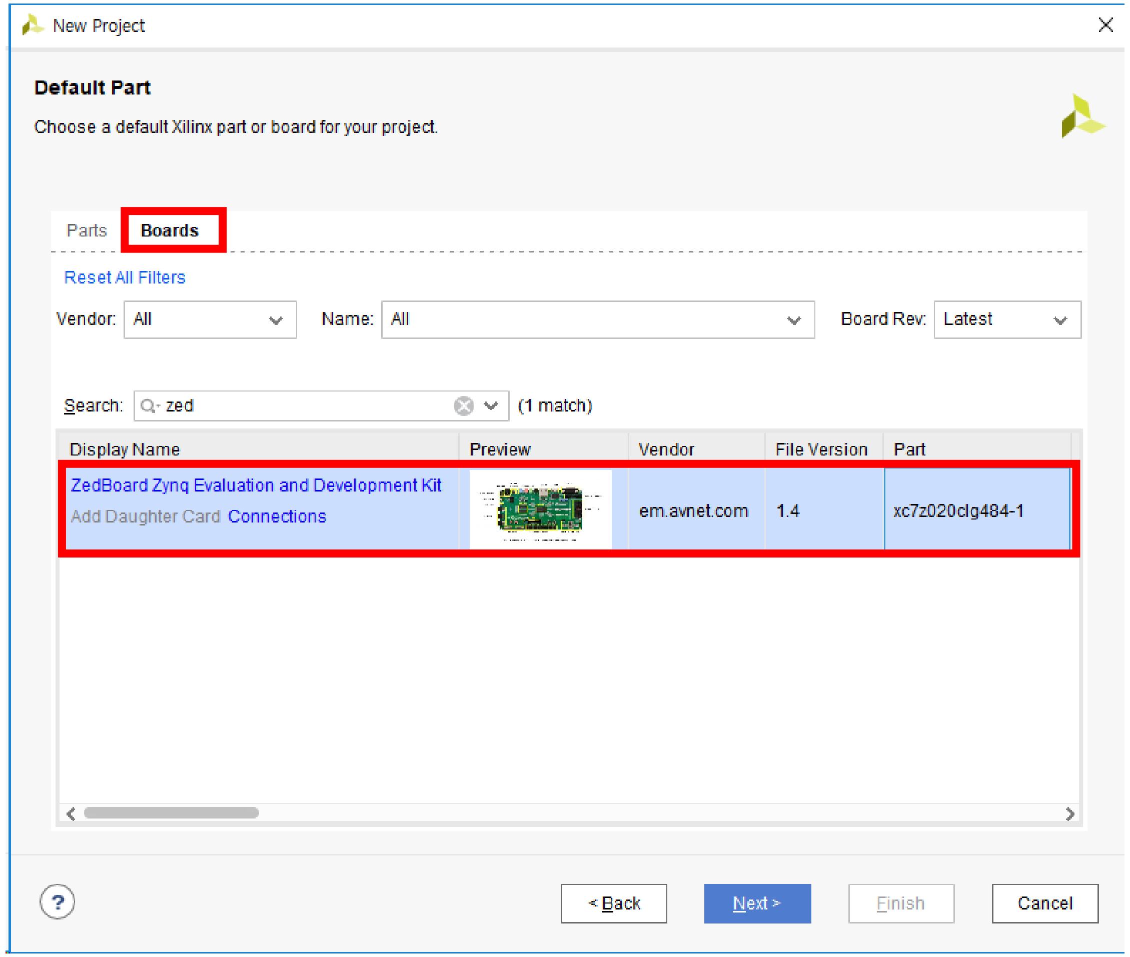
Overview

- **Vivado Block Design Tutorial**
  - First block design = Processing System + BRAM + Connectivity
    - Note) Term project = PS + BRAM + Connectivity + Custom IP
- **FPGA + Linux Tutorial**
  - Debian Linux on zedboard
  - Access BRAM via C program
- **Practice**
  - Running a sample project

Vivado Block Design Tutorial

Vivado project creation

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



Default Part

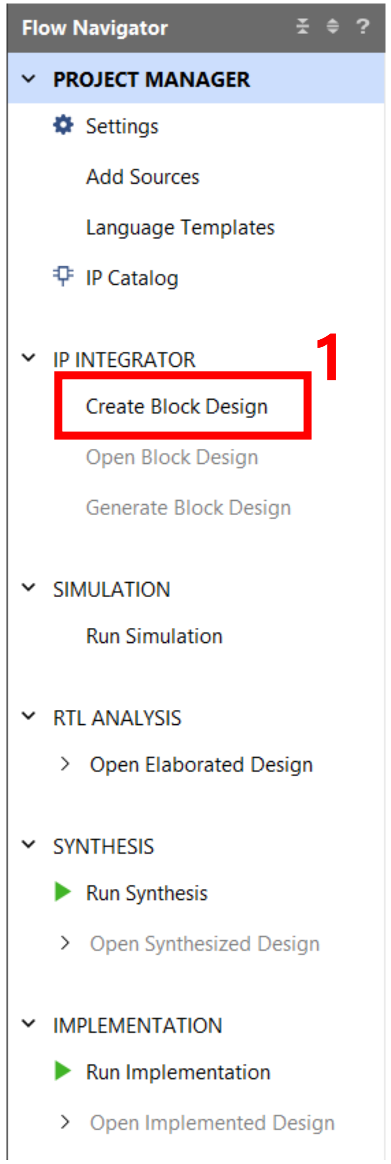
Choose a default Xilinx part or board for your project.

Vendor: All   Name: All   Board Rev: Latest

Search: zed (1 match)

Display Name	Preview	Vendor	File Version	Part
ZedBoard Zynq Evaluation and Development Kit Add Daughter Card Connections		em.avnet.com	1.4	xc7z020clg484-1

Block Design - PS



Flow Navigator

PROJECT MANAGER
- Settings
- Add Sources
- Language Templates
- IP Catalog

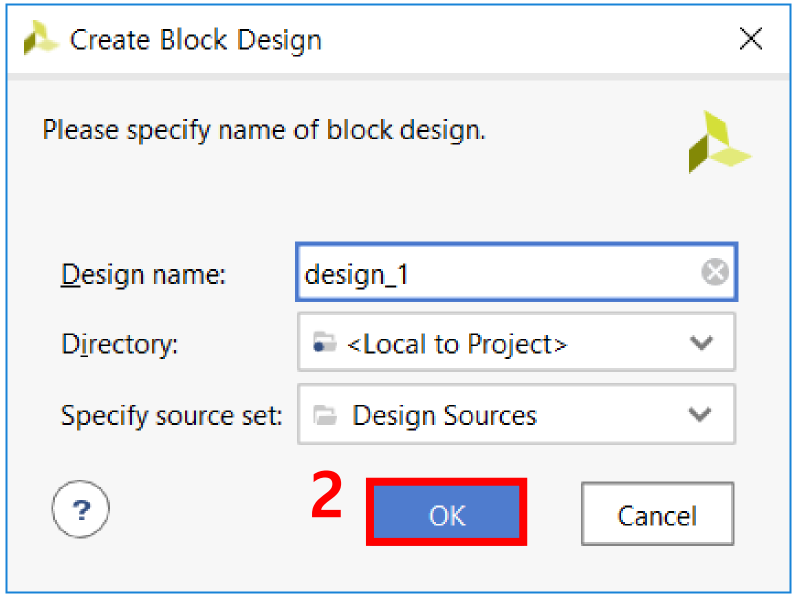
IP INTEGRATOR
- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION
- Run Simulation

RTL ANALYSIS
- Open Elaborated Design

SYNTHESIS
- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION
- Run Implementation
- Open Implemented Design



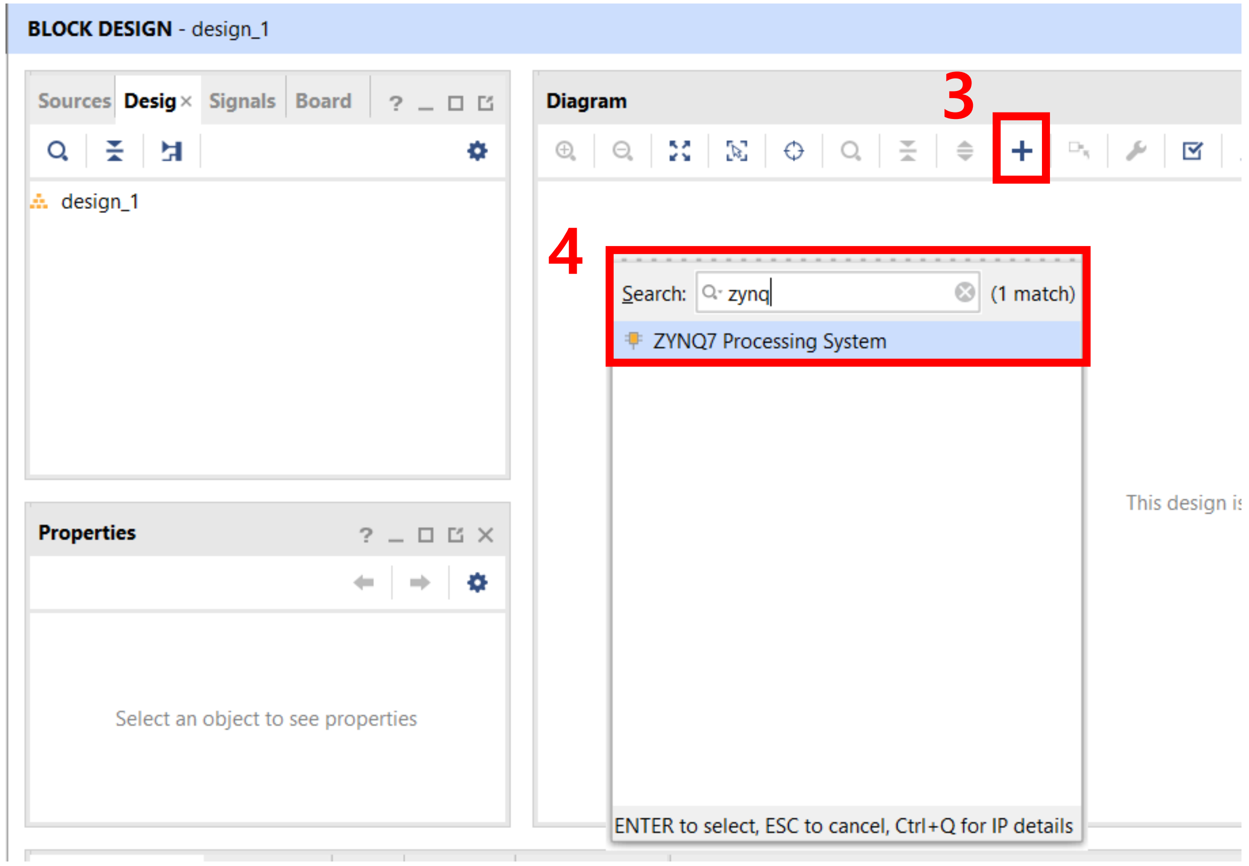
Create Block Design

Please specify name of block design.

Design name: design_1

Directory: <Local to Project>

Specify source set: Design Sources



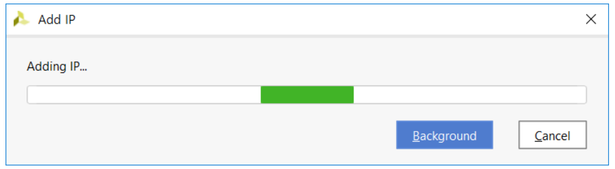
BLOCK DESIGN - design_1

Search: zynq (1 match)

ZYNQ7 Processing System

ENTER to select, ESC to cancel, Ctrl+Q for IP details

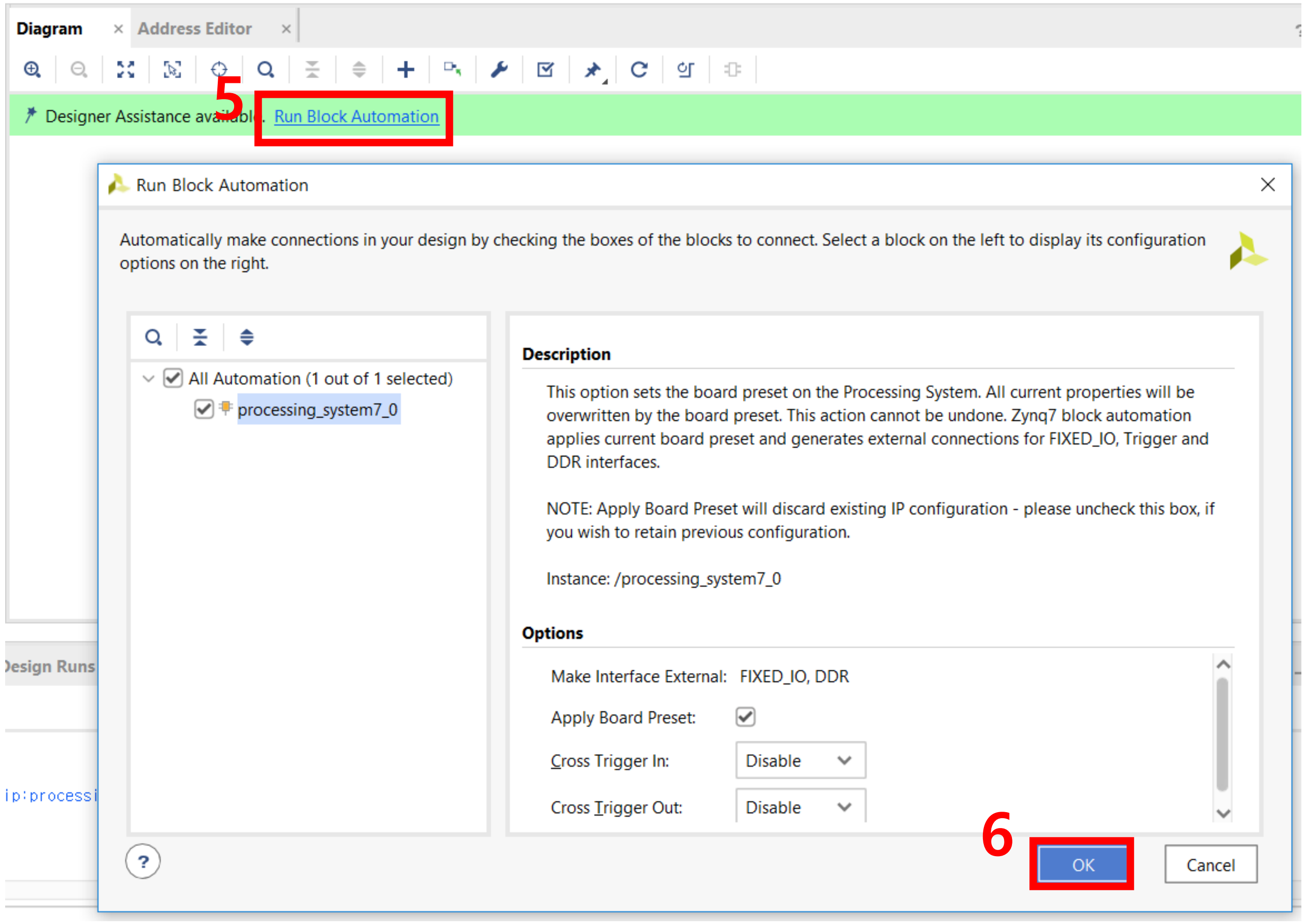
This design is



Add IP

Adding IP...

Block Design - PS



Run Block Automation

Automatically make connections in your design by checking the boxes of the blocks to connect. Select a block on the left to display its configuration options on the right.

- ☑ All Automation (1 out of 1 selected)
  - ☑ processing_system7_0

Description

This option sets the board preset on the Processing System. All current properties will be overwritten by the board preset. This action cannot be undone. Zynq7 block automation applies current board preset and generates external connections for FIXED_IO, Trigger and DDR interfaces.

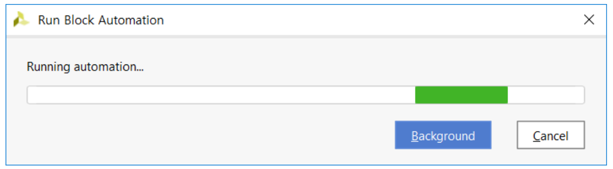
NOTE: Apply Board Preset will discard existing IP configuration - please uncheck this box, if you wish to retain previous configuration.

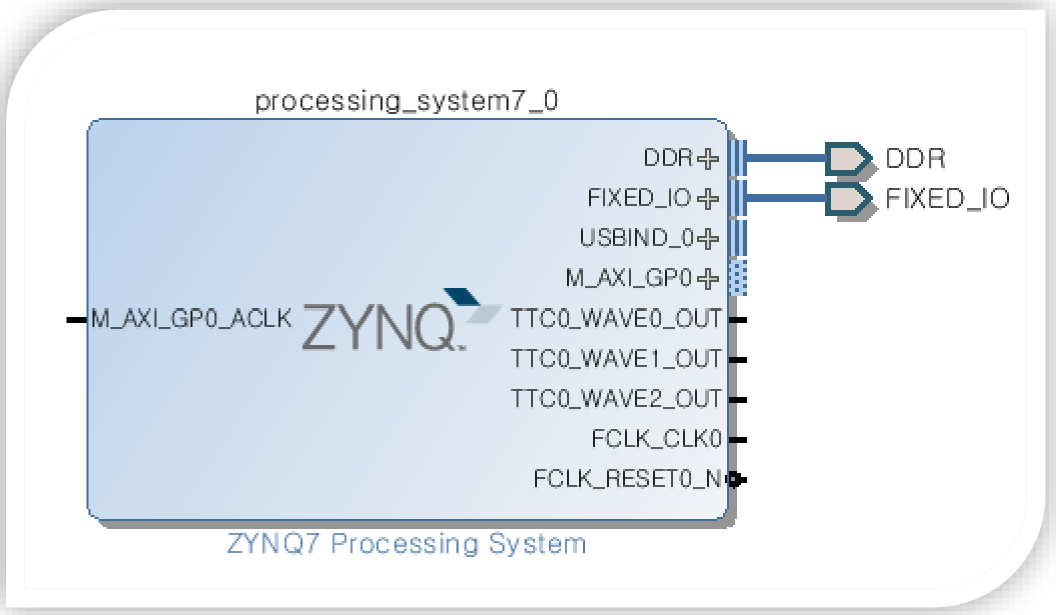
Instance: /processing_system7_0

Options

Make Interface External:	FIXED_IO, DDR
Apply Board Preset:	☑
Cross Trigger In:	Disable
Cross Trigger Out:	Disable

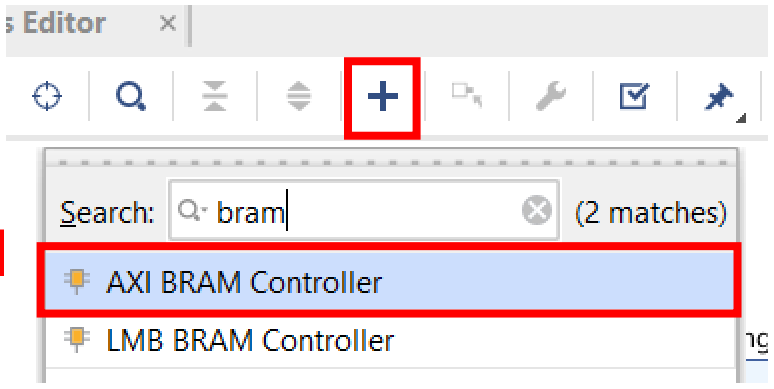
OK Cancel



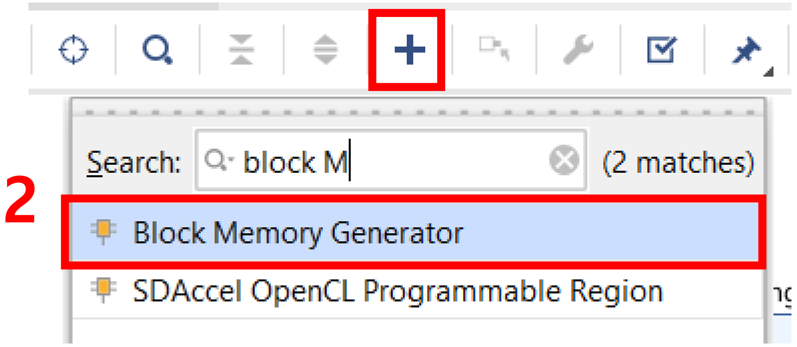


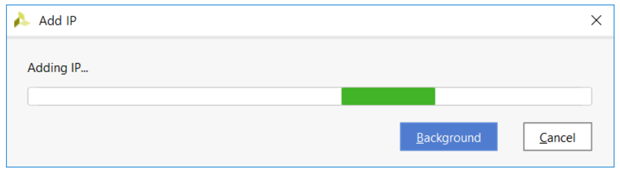
Block Design - BRAM

1

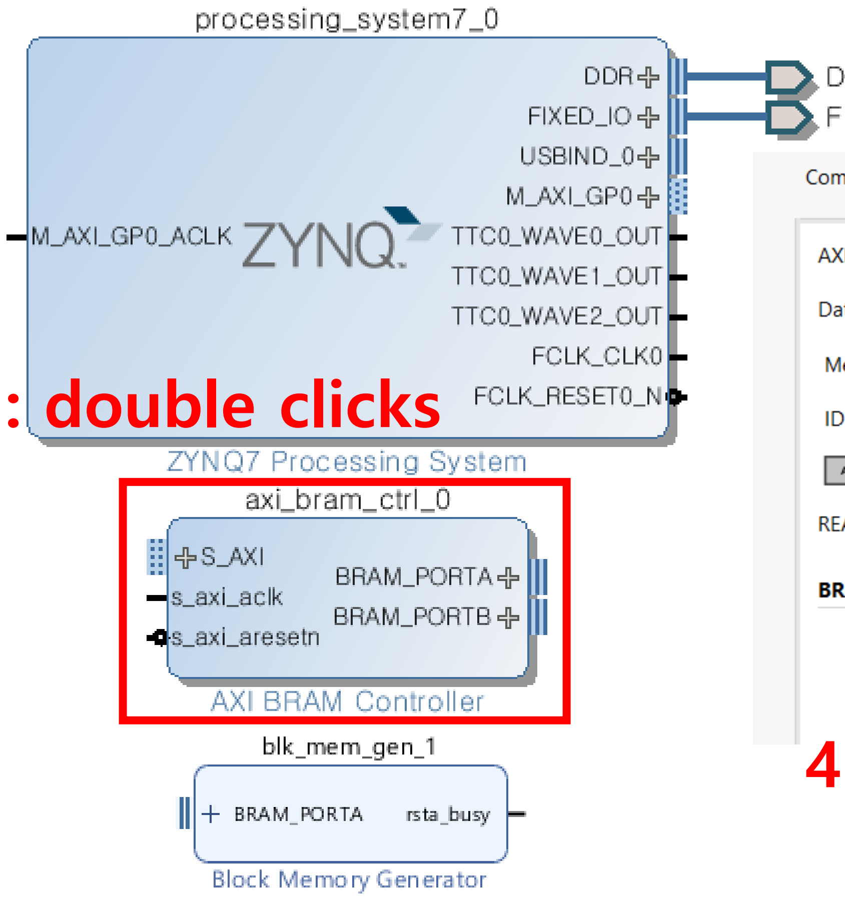


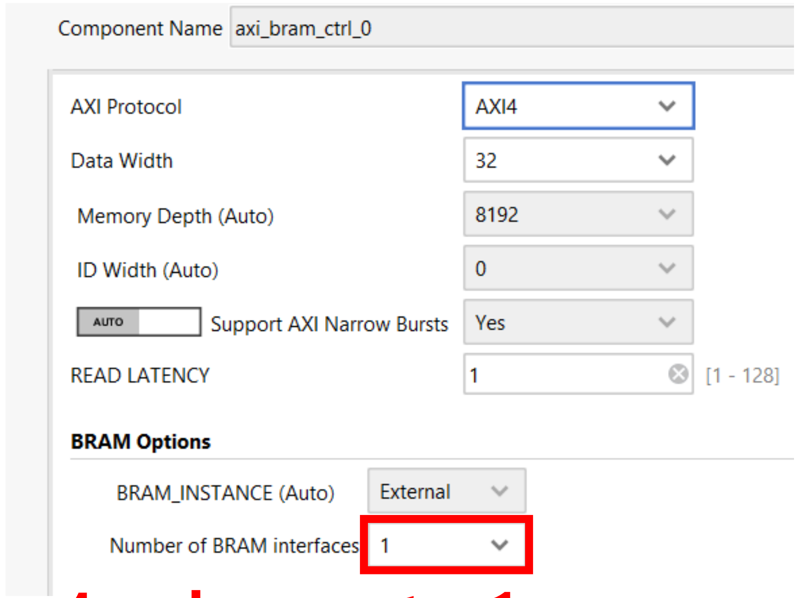
2





3 : double clicks





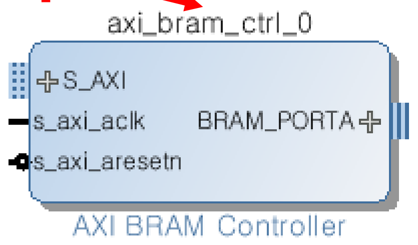
Component Name axi_bram_ctrl_0

AXI Protocol	AXI4
Data Width	32
Memory Depth (Auto)	8192
ID Width (Auto)	0
AUTO Support AXI Narrow Bursts	Yes
READ LATENCY	1 [1 - 128]

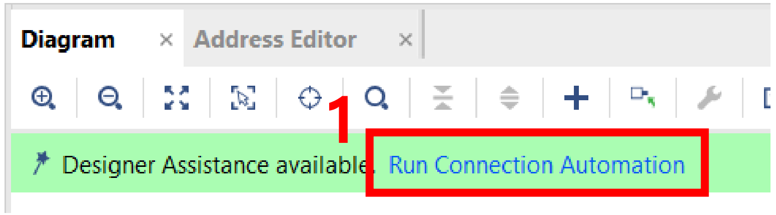
BRAM Options

BRAM_INSTANCE (Auto)	External
Number of BRAM interfaces	1

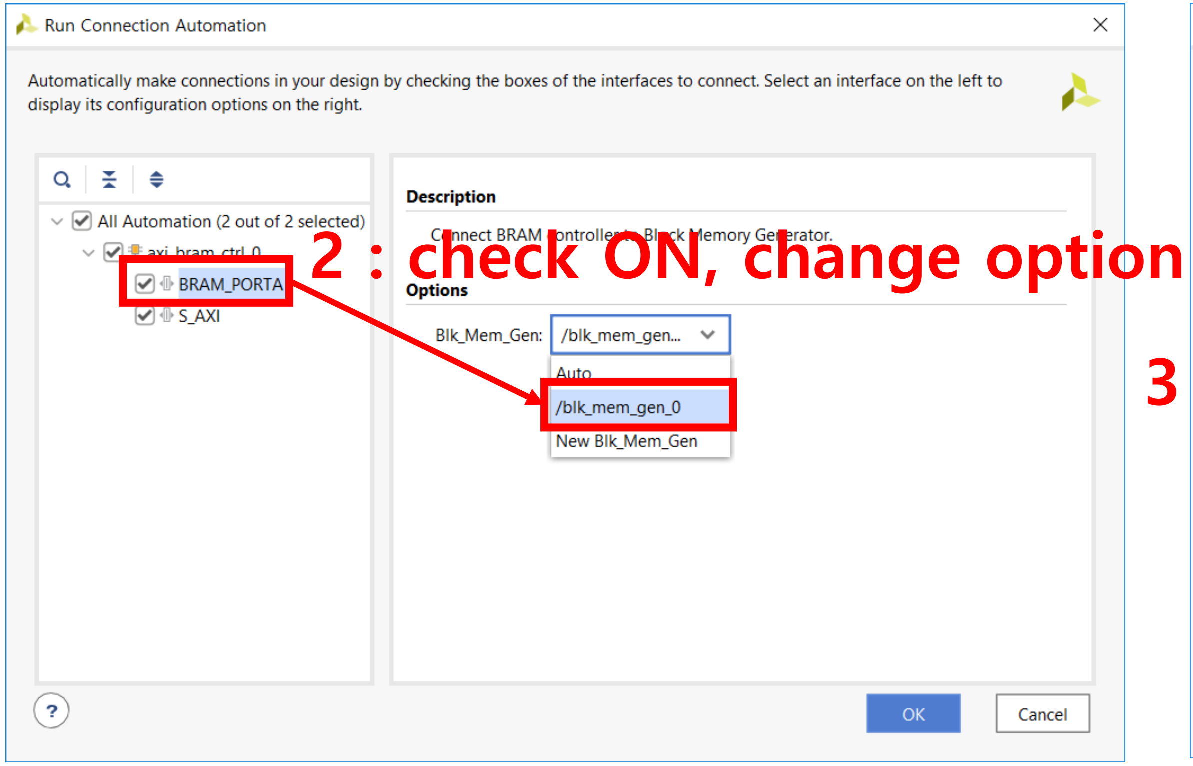
4 : change to 1



Block Design - Connectivity



Designer Assistance available. Run Connection Automation



Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

- ☑ All Automation (2 out of 2 selected)
  - ☑ axi_bram_ctrl_0
    - ☑ BRAM_PORTA
    - ☑ S_AXI

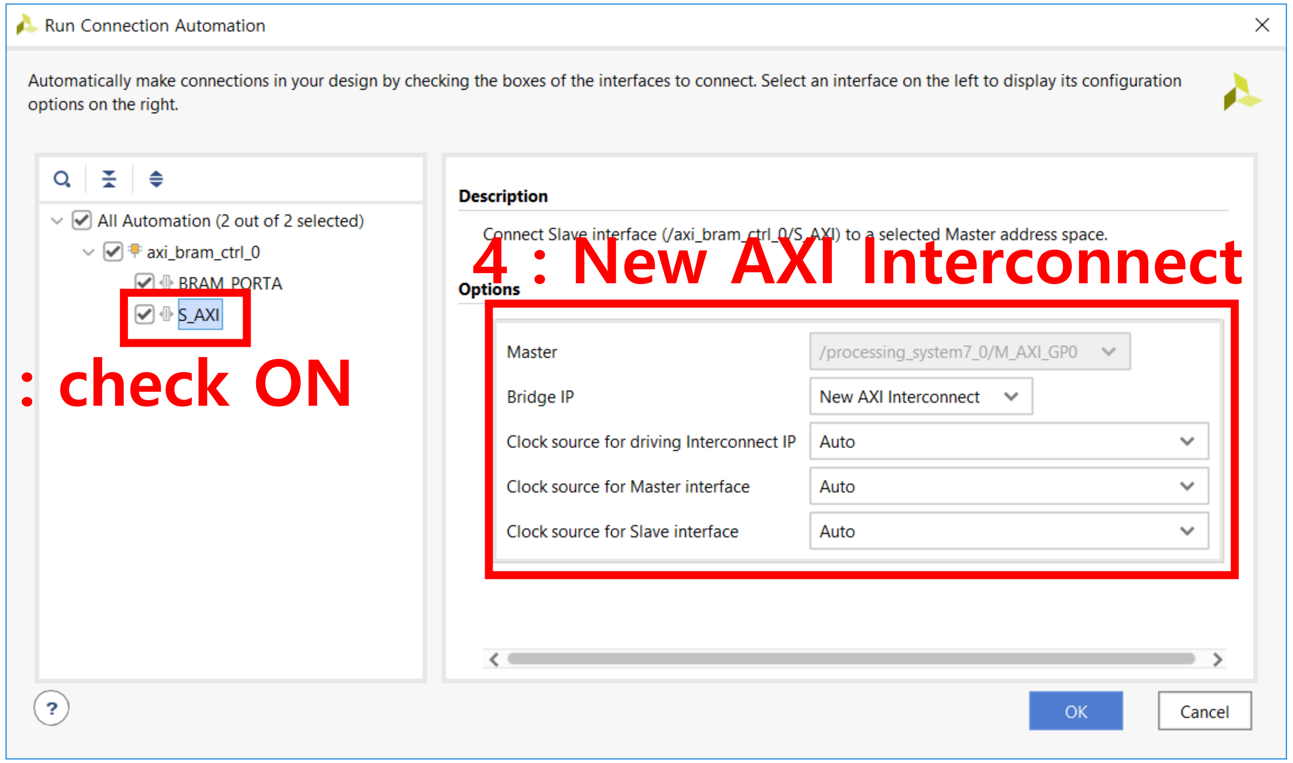
Description

Connect BRAM Controller to Block Memory Generator.

Options

Blk_Mem_Gen: /blk_mem_gen...

- Auto
- /blk_mem_gen_0
- New Blk_Mem_Gen



Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

- ☑ All Automation (2 out of 2 selected)
  - ☑ axi_bram_ctrl_0
    - ☑ BRAM_PORTA
    - ☑ S_AXI

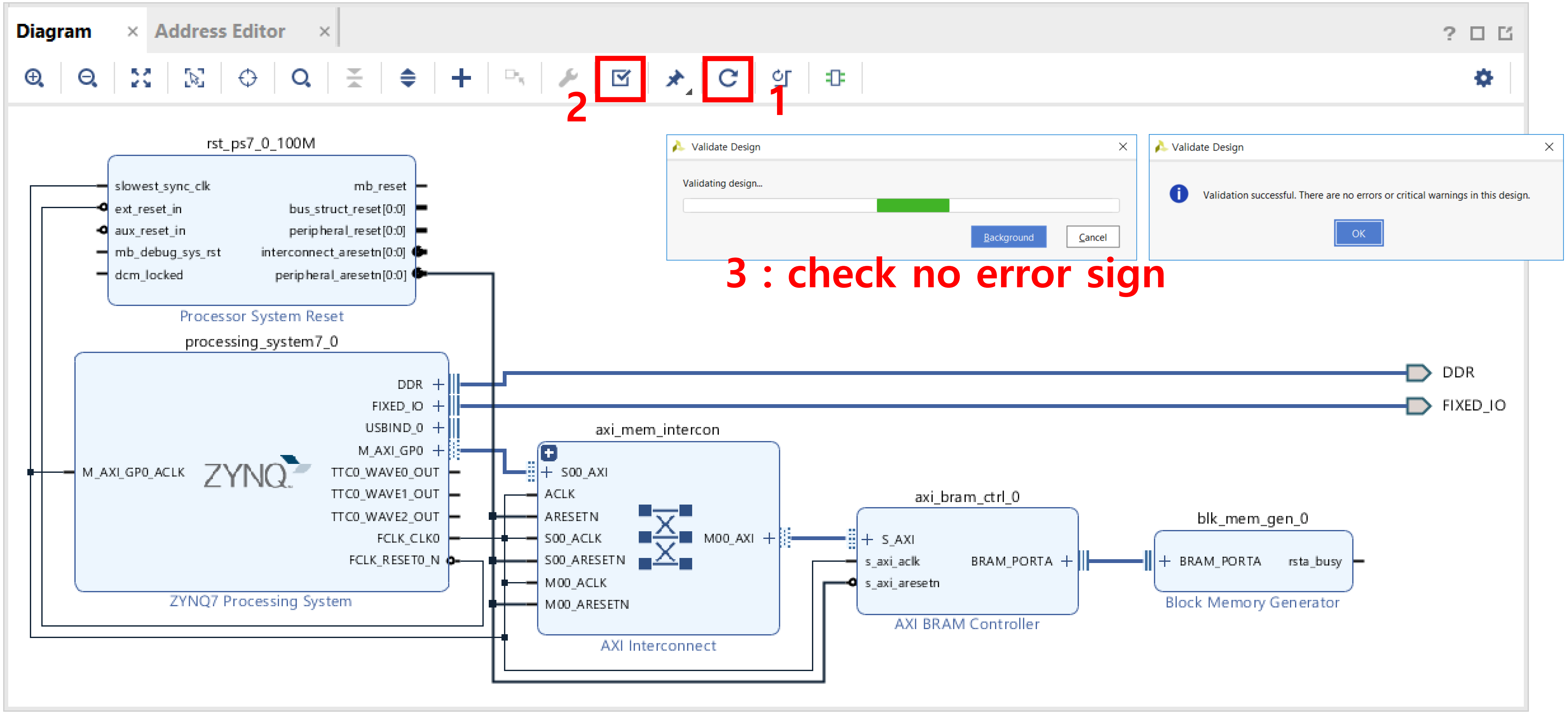
Description

Connect Slave interface (/axi_bram_ctrl_0/S_AXI) to a selected Master address space.

Options

Master	/processing_system7_0/M_AXI_GP0
Bridge IP	New AXI Interconnect
Clock source for driving Interconnect IP	Auto
Clock source for Master interface	Auto
Clock source for Slave interface	Auto

Block Design - Connectivity



Validating design...

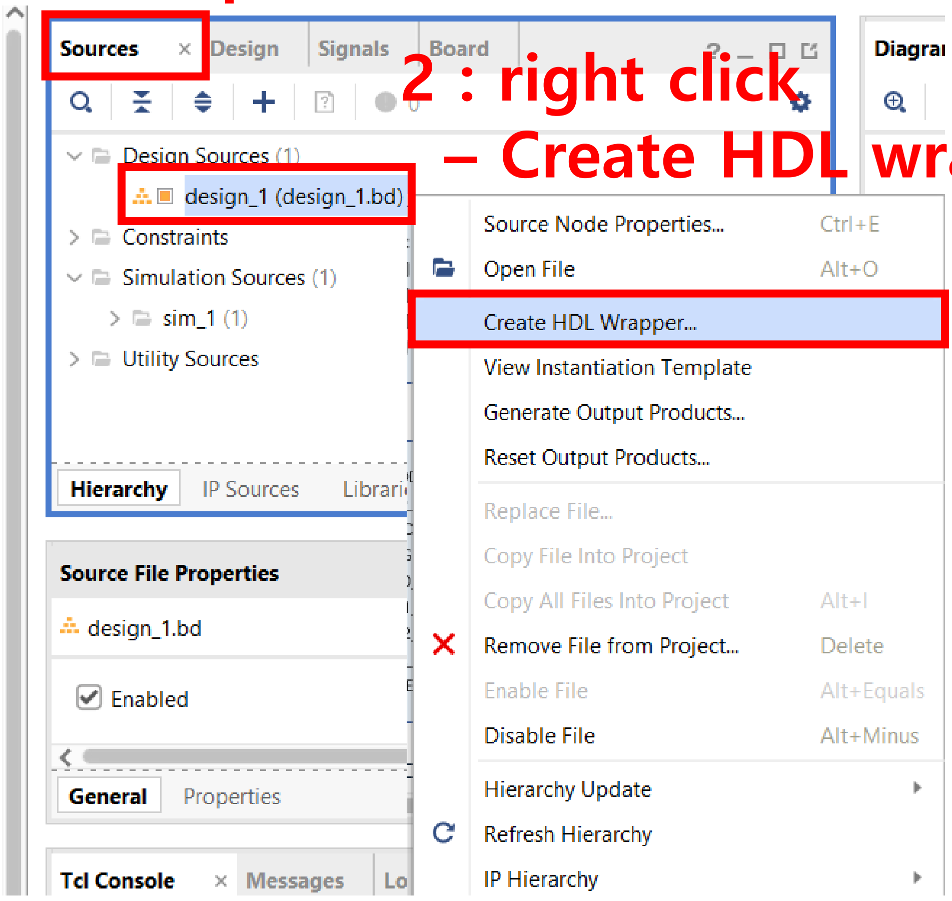
Validation successful. There are no errors or critical warnings in this design.

3 : check no error sign

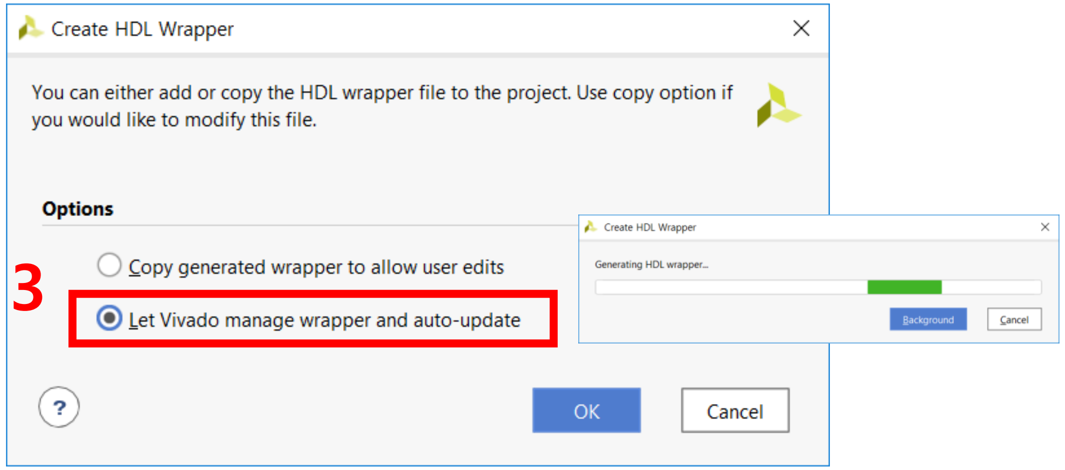
Block Design - Bitstream

1

2 : right click – Create HDL wrapper



3


Create HDL Wrapper

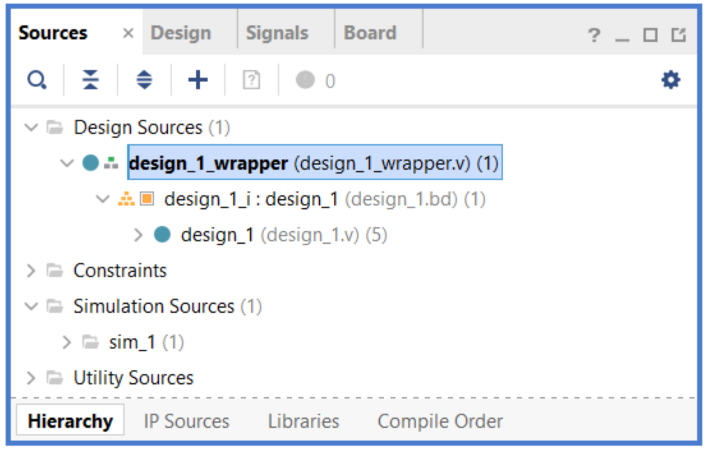
You can either add or copy the HDL wrapper file to the project. Use copy option if you would like to modify this file.

Options

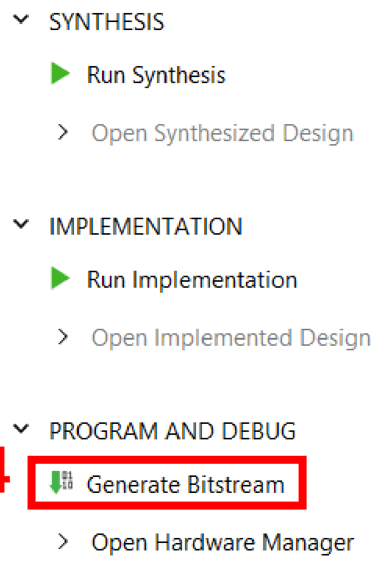
( ) Copy generated wrapper to allow user edits
(●) Let Vivado manage wrapper and auto-update

Create HDL Wrapper

Generating HDL wrapper...

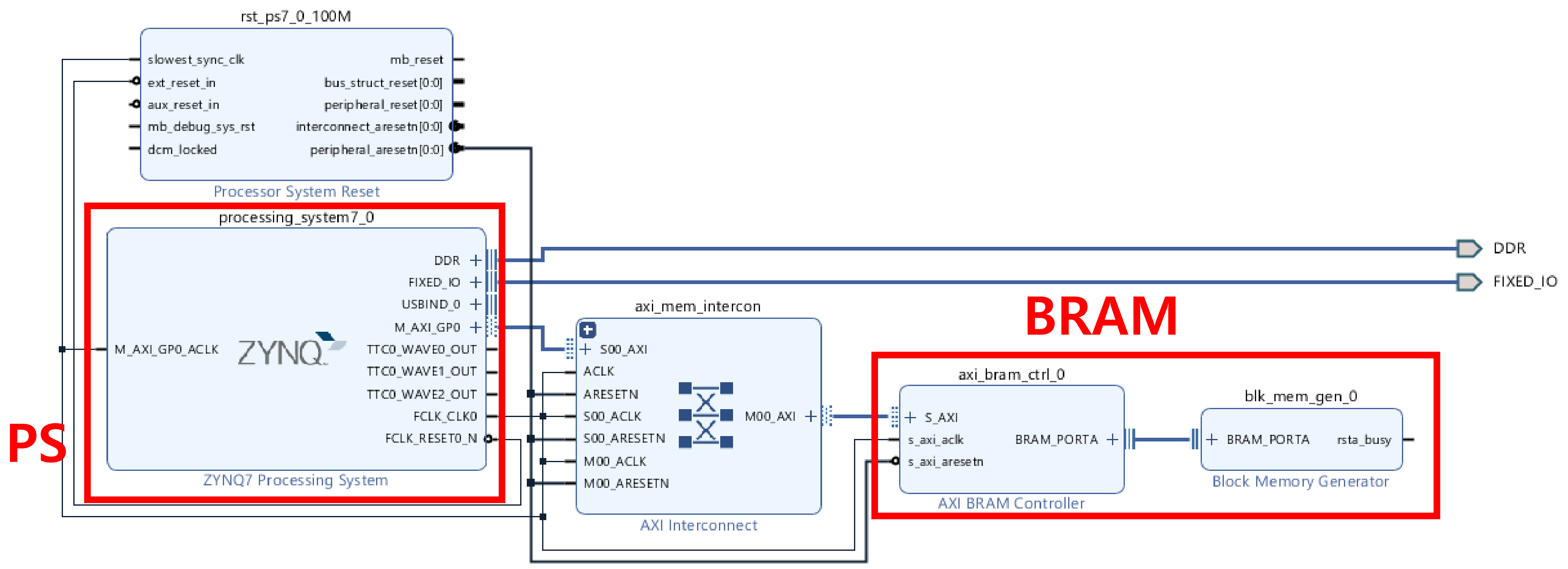


4



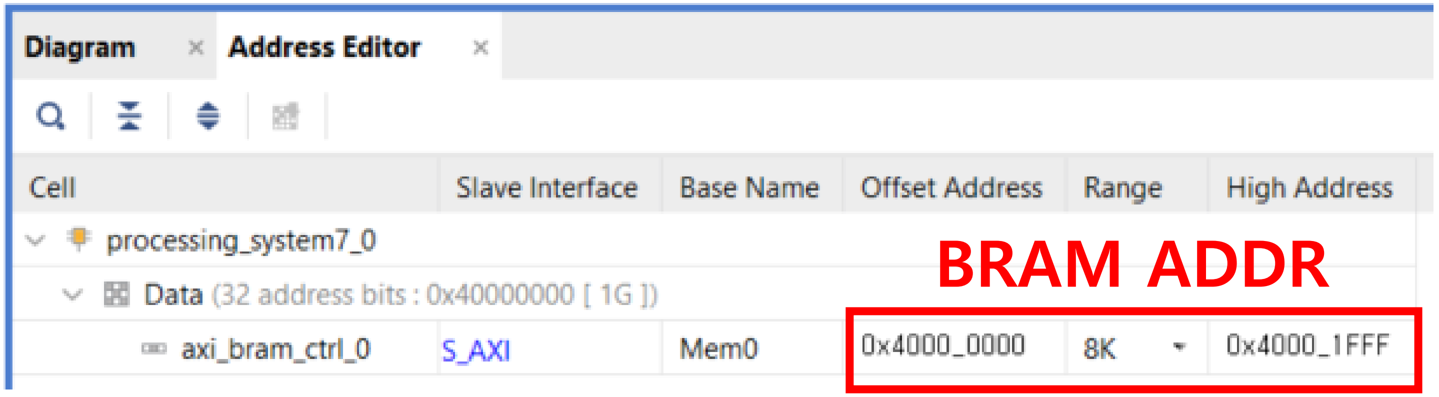
Block Design - Summary

- Having built **a first block design** equipped with
  - PS: Processing System (part of Zynq-PS, ARM Cortex A9)
  - BRAM: Block Random Access Memory (part of Zynq-FPGA)
  - Connectivity (AXI interconnect, part of Zynq-FPGA)



Programmer's Perspective

- BRAM is @ address 0x4000_0000 ~ 0x4000_1FFF
  - Note) DRAM (BD.IC25/26) is @ address 0x0000_0000 ~ 0x3FFF_FFFF
- System call **mmap** can be used to access BRAM *(TBD)*
  - int foo = open("/dev/mem", O_RDWR);
  - float *ptr = mmap(NULL, size, PROT_READ|PROT_WRITE, MAP_SHARED, foo, 0x40000000);

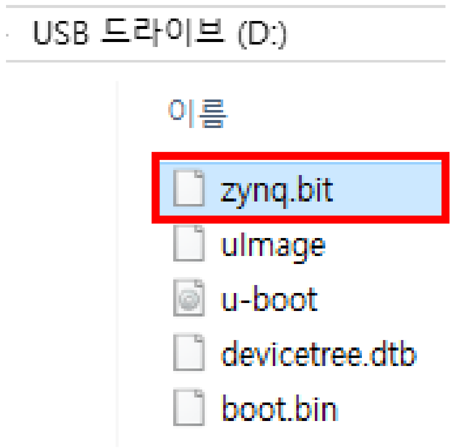


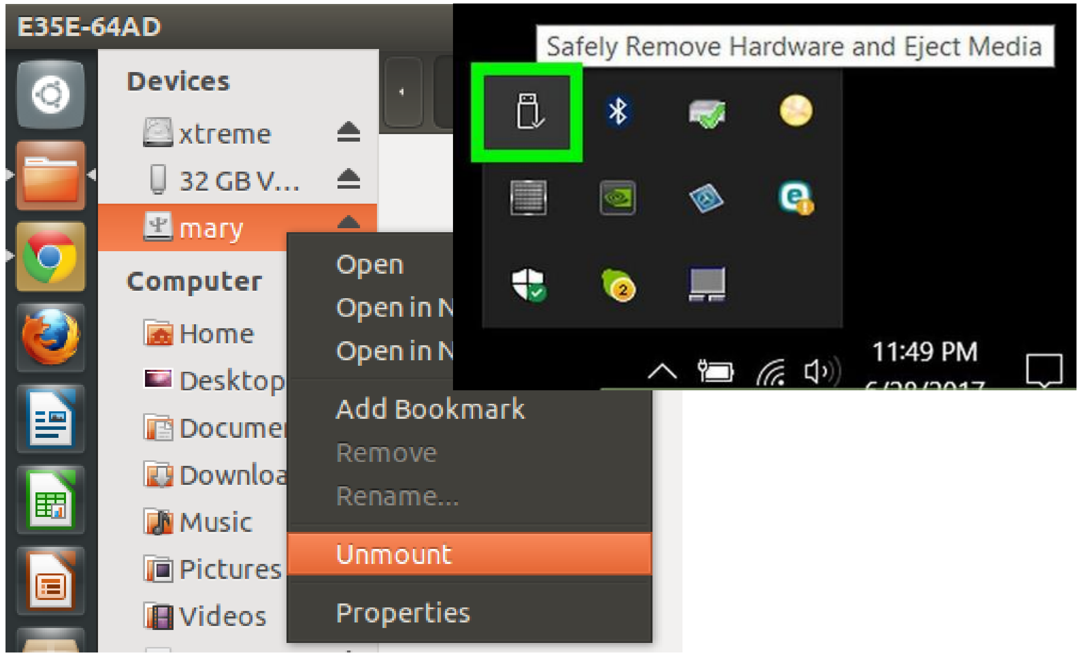
Cell	Slave Interface	Base Name	Offset Address	Range	High Address
processing_system7_0					
Data (32 address bits : 0x40000000 [1G])					
axi_bram_ctrl_0	S_AXI	Mem0	0x4000_0000	8K	0x4000_1FFF

FPGA + Linux Tutorial

Preparing the Bitstream

- Find your bitstream(.bit) file in ($project_name).runs/impl_1/design_1_wrapper.bit
  - Move your bitstream file to partition1 and change file name to zynq.bit
  - (before detach the sdcard from computer) unmount or eject your sdcard from your computer. Or not your sdcard will be broken. (penalty ☺)





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

[ZEDBoard Hardware User's Guide, 2014]

Preparing the board

- Insert SD card (1)
- Set SD boot mode (2)
  - BD.JP7 ~ BD.JP11
    - 5'b00000 -> **5'b01100**
- Insert LAN Cable (3)
- Insert power cable (4)
  - Yet stay power OFF
- Connect USB cable (5)
  - BD.J14
  - HOST$ dmesg

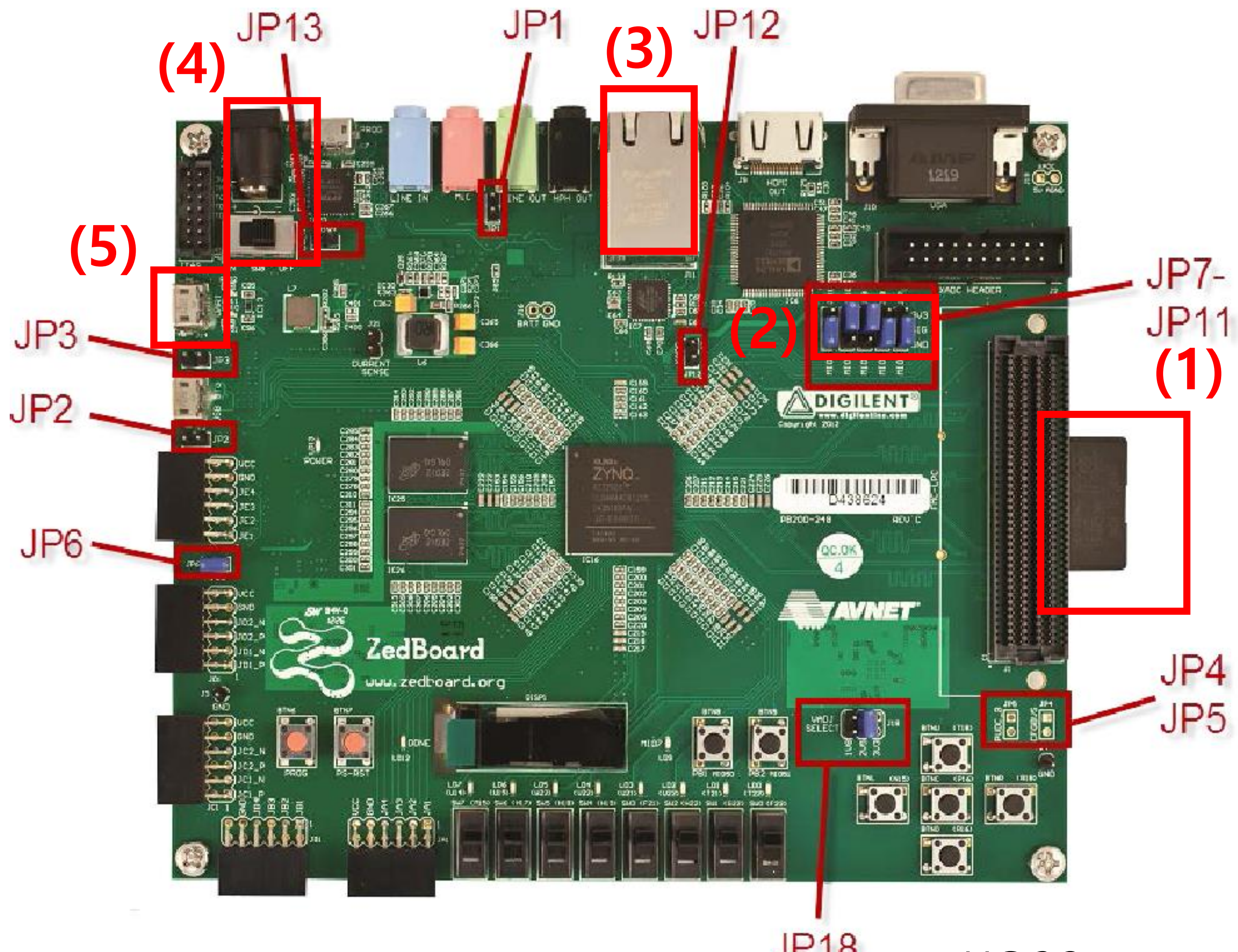
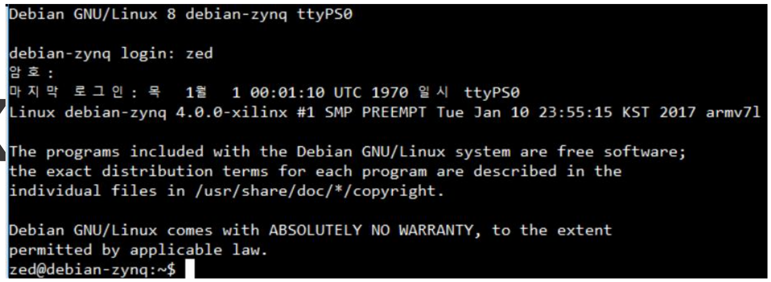


Figure 20 - ZedBoard Jumper Map

UG.36

First Contact to Embedded Linux



```
Debian GNU/Linux 8 debian-zynq ttyPS0

debian-zynq login: zed
암 호 :
마 지 막  로 그 인 : 목   1월   1 00:01:10 UTC 1970 일 시  ttyPS0
Linux debian-zynq 4.0.0-xilinx #1 SMP PREEMPT Tue Jan 10 23:55:15 KST 2017 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
zed@debian-zynq:~$
```

- Board Power ON
- Open terminal @ HOST
  - HOST$ minicom -D /dev/ttyACM0
  - Login ID/PW: zed/zedzed
- Run example program
  - BOARD$ git clone https://github.com/tahsd/hsd21_lab09
  - BOARD$ cd hsd21_lab09
  - BOARD$ make
  - (before un-plugging the board) BOARD$ sudo poweroff
    - If you do not execute poweroff on terminal and eject the sdcard, your sdcard will not work permanantly. (penalty ☺)

```
[  OK  ] Reached target Multi-User System.
[  OK  ] Reached target Graphical Interface.
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Started Update UTMP about System Runlevel Changes.

Debian GNU/Linux 8 debian-zynq ttyPS0

debian-zynq login:
```

```
addr      FPGA(hex)
0         0
1         2
2         4
3         6
...
```

Source Code – main.c

```
  int foo = open("/dev/mem", O_RDWR);
  // Given a pathname for a file, open() returns a file descriptor
  // 'dev/mem' refers to the system's physical memory
  // O_RDWR means both readable and writable access mode
  int *fpga_bram = mmap(NULL, SIZE * sizeof(int), PROT_READ|PROT_WRITE, MAP_SHARED, foo, 0x40000000);
  // mmap() creates a new mapping in the virtual address space of the calling process
  // NULL means that the kernel chooses the address for mapping
  // SIZE specifies the length of the mapping
  // PROT_ arguments describe the memory protection (RD/WR)
  // MAP_SHARED makes updates visible to other processes
  // foo indicates the file descriptor to be mapped
  // 0x4000_0000 refers to offset of the file descriptor
for (i = 0; i < SIZE; i++)
    *(fpga_bram + i) = (i * 2);
  // write arbitrary data on the BRAM area
 printf("%-10s%-10s₩n", "addr", "FPGA(hex)");
 for (i = 0; i < SIZE; i++)
    printf("%-10d%-10X₩n", i, *(fpga_bram + i));
  // read and show the data to check if BRAM's working correctly
```

Source Code – Makefile

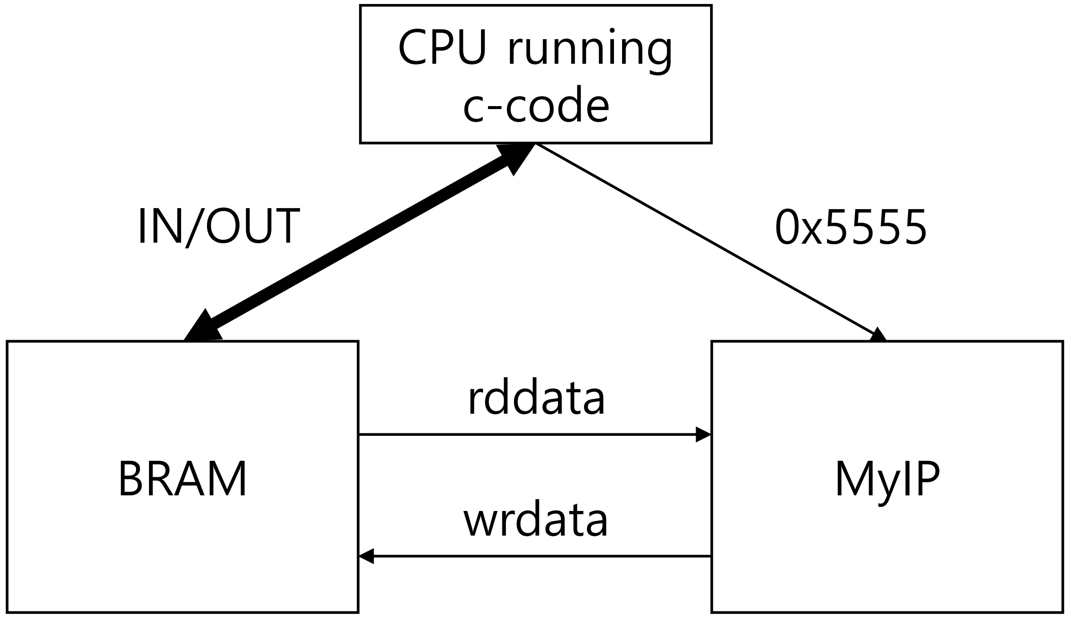
```
all: main.c
	gcc main.c && sudo ./a.out

// target : prerequisites
// [TAB]recipe
//
// ex)
// make $target : run recipes of $target using prerequisites
// make : (missing $target arguments) run the first target
```

Main Practice

Practice

- **Running a sample project**
  - Run a given sample project in your FPGA linux.
    - Run and explain a brief functionality of sample project on report.
  - Sample Project
    - https://github.com/tahsd/hsd21_lab09_practice
      - Sample HW: zynq.bit (made by MyIP)
      - Sample SW: main.c Makefile
    - You don't need anything to edit or implement on HW.
    - For investigation purposes, you can modify the c-code at your disposal.



Comments from TA

- Q. How to export my own *.bit into SD card?
  - Copy *.bit **A** to **B**
  - **A**: {project}.runs/impl_1/design_1_wrapper.bit
  - **B**: /SDCARD_1.1G_PARTITION/zynq.bit
- Q. How to quit my minicom instance?
  - Press **ctrl A q** -> yes
- Q. How to turn the board off and on without deterring minicom instance?
  - Press BD.BTN7
- Click [eject] before removing SD card from the card reader

Homework

- Requirements
  - Result
    - Attach a screenshot that can show your code works in ZedBoard
    - Just for main practice
  - Report
    - Explain functionality of MyIP in Main Practice
    - In your own words
    - Either in Korean or in English
    - # of pages does not matter
    - **PDF only!!**
  - **Result + Report to a .zip**
- Upload (.zip) file on ETL
  - Submit one (.zip) file
    - zip file name : [Lab08]name.zip (ex : [Lab08]홍길동.zip)
  - Due: 5/12(WED) 23:59
    - **No Late Submission**