

# **Extending Memory Space with BRAM**

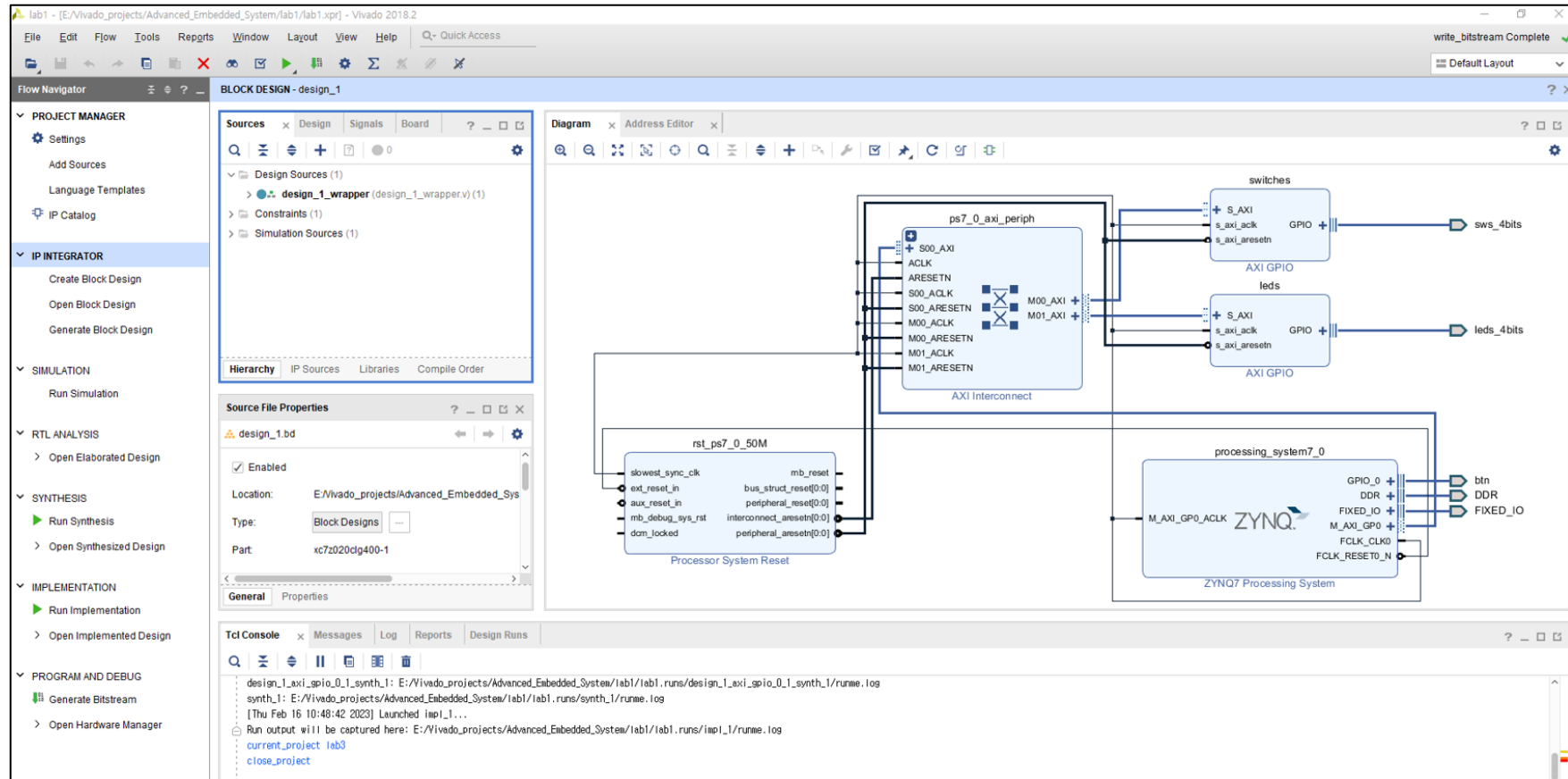
(lab3)

17<sup>th</sup> February 2023

# 1. lab3 프로젝트 생성하기

> lab1 프로젝트를 lab3 프로젝트로 복사하기

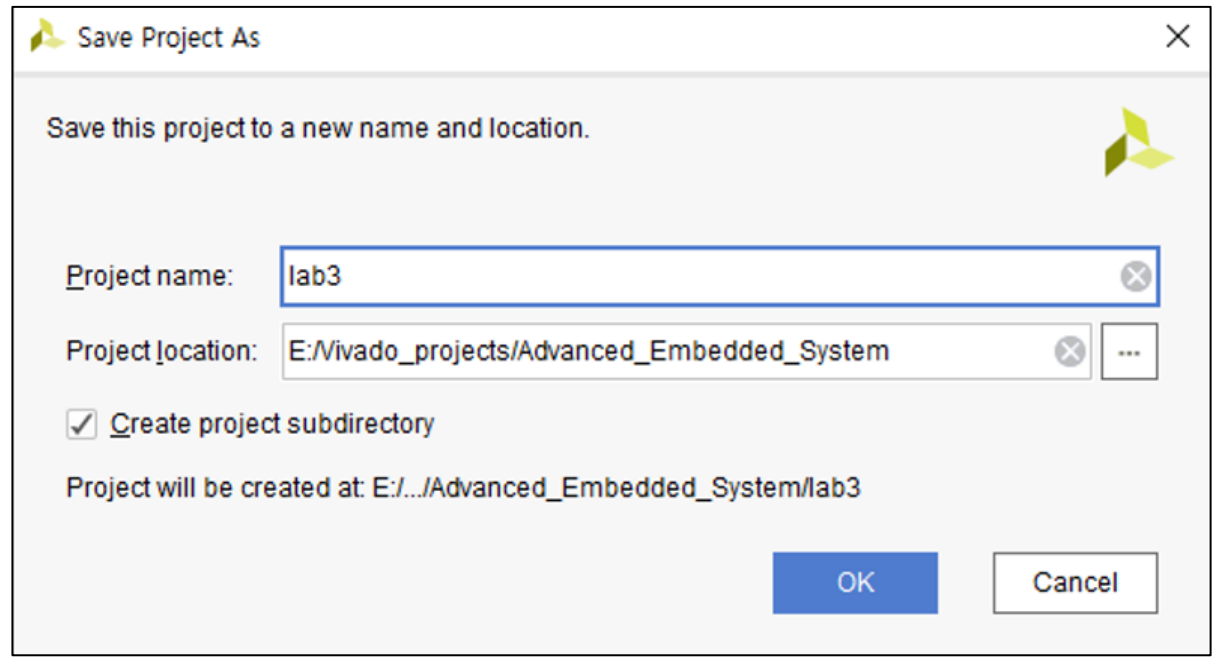
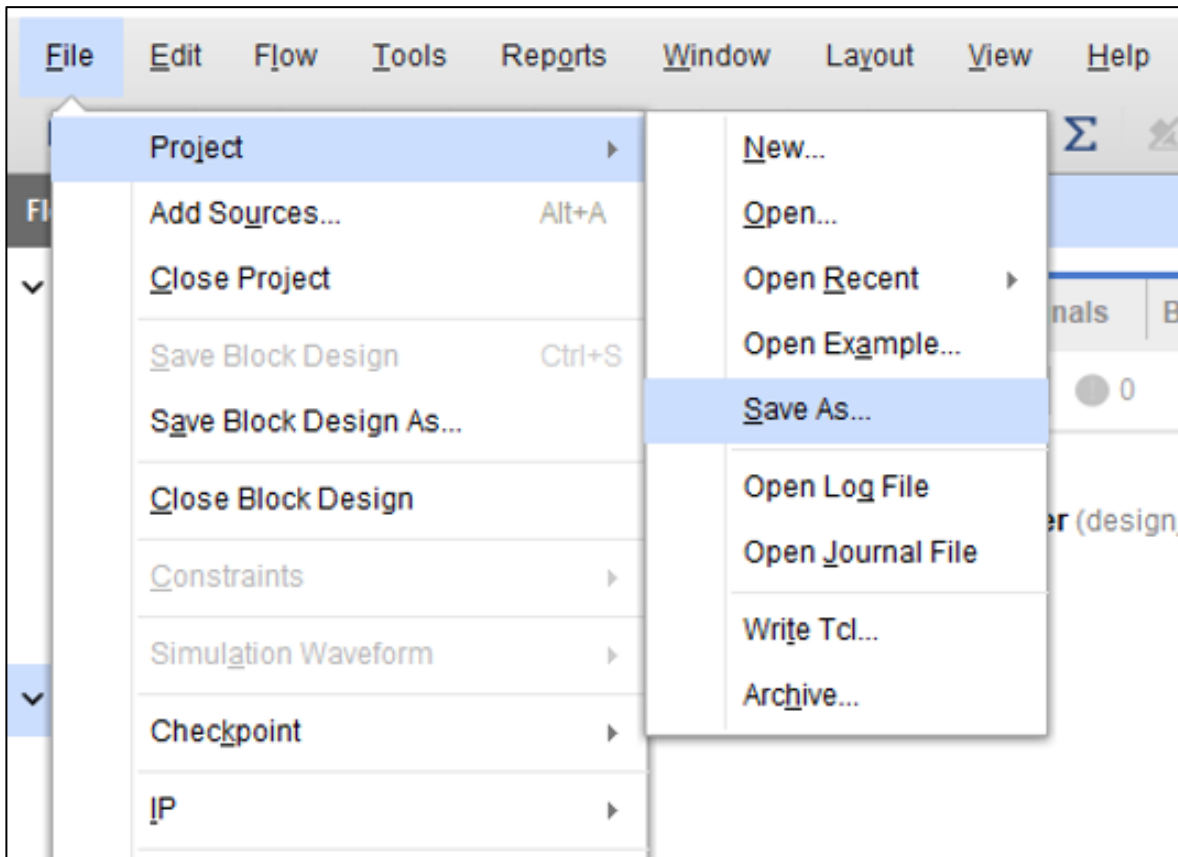
• 기존 lab1 프로젝트를 열기



# 1. lab3 프로젝트 생성하기

## > lab1 프로젝트를 lab3 프로젝트로 복사하기

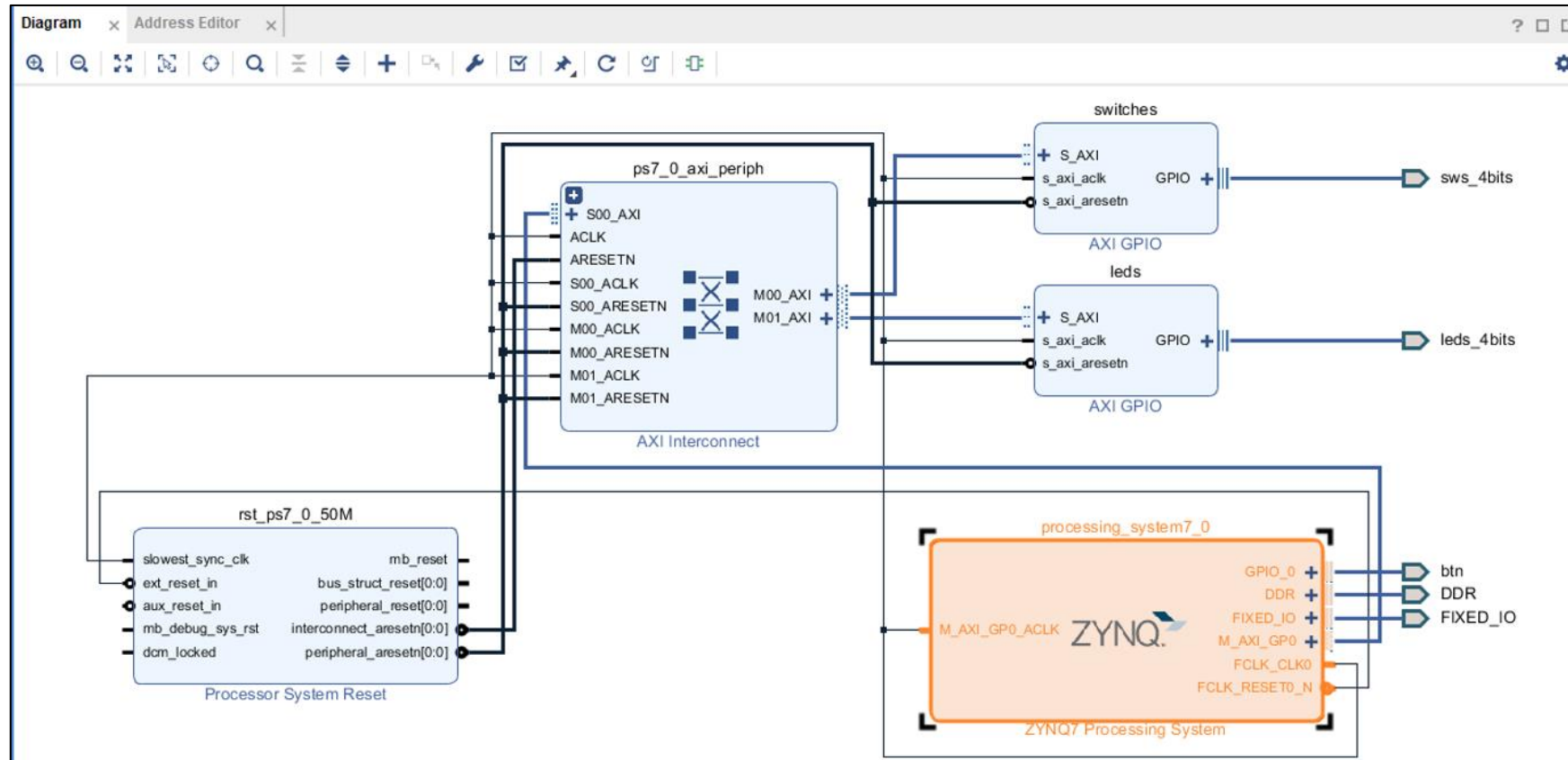
- File > Project > Save As 클릭 -> 새 project name을 lab3로 저장



## 2. Enabling M\_AXI\_GP1 in PS

### > ZYNQ 프로세서 재설정하기

- ZYNQ PS 블록을 더블 클릭하여 커스터마이징 진행



## 2. Enabling M\_AXI\_GP1 in PS

### > ZYNQ 프로세서 재설정하기

- PS-PL Configuration에서 다음과 같이 **M AXI GP1 interface**를 체크하여 enabling

The screenshot shows the 'PS-PL Configuration' window in Vivado. The left sidebar contains a 'Page Navigator' with the following items: Zynq Block Design, PS-PL Configuration (selected), Peripheral I/O Pins, MIO Configuration, Clock Configuration, DDR Configuration, SMC Timing Calculation, and Interrupts. The main area displays the 'PS-PL Configuration' settings. At the top, there is a search bar and navigation icons. Below this is a table with columns 'Name', 'Select', and 'Description'. The table is organized into sections: 'General', 'AXI Non Secure Enablement', 'GP Master AXI Interface', and 'GP Slave AXI Interface'. Under 'GP Master AXI Interface', the 'M AXI GP1 interface' is selected (checked). A yellow callout box with the text 'PCW USE M AXI GP1' points to the 'M AXI GP1 interface' row. The 'PS-PL Cross Trigger interface' is also visible at the bottom of the table.

Name	Select	Description
> General		
> AXI Non Secure Enablement	0	Enable AXI Non Secure Transaction
> GP Master AXI Interface		
> M AXI GP0 interface	<input checked="" type="checkbox"/>	Enables General purpose AXI master interface 0
> M AXI GP1 interface	<input checked="" type="checkbox"/>	Enables General purpose AXI master interface 1
> GP Slave AXI Interface		
> HP Slave AXI Interface		
> ACP Slave AXI Interface		
> DMA Controller		
> PS-PL Cross Trigger interface	<input type="checkbox"/>	Enables PL cross trigger signals to PS and vice-versa

## 2. Enabling M\_AXI\_GP1 in PS

### > ZYNQ 프로세서 재설정하기

- Clock Configuration에서 다음과 같이 **FCLK\_CLK1**을 체크 + 주파수를 **140(MHz)**로 변경

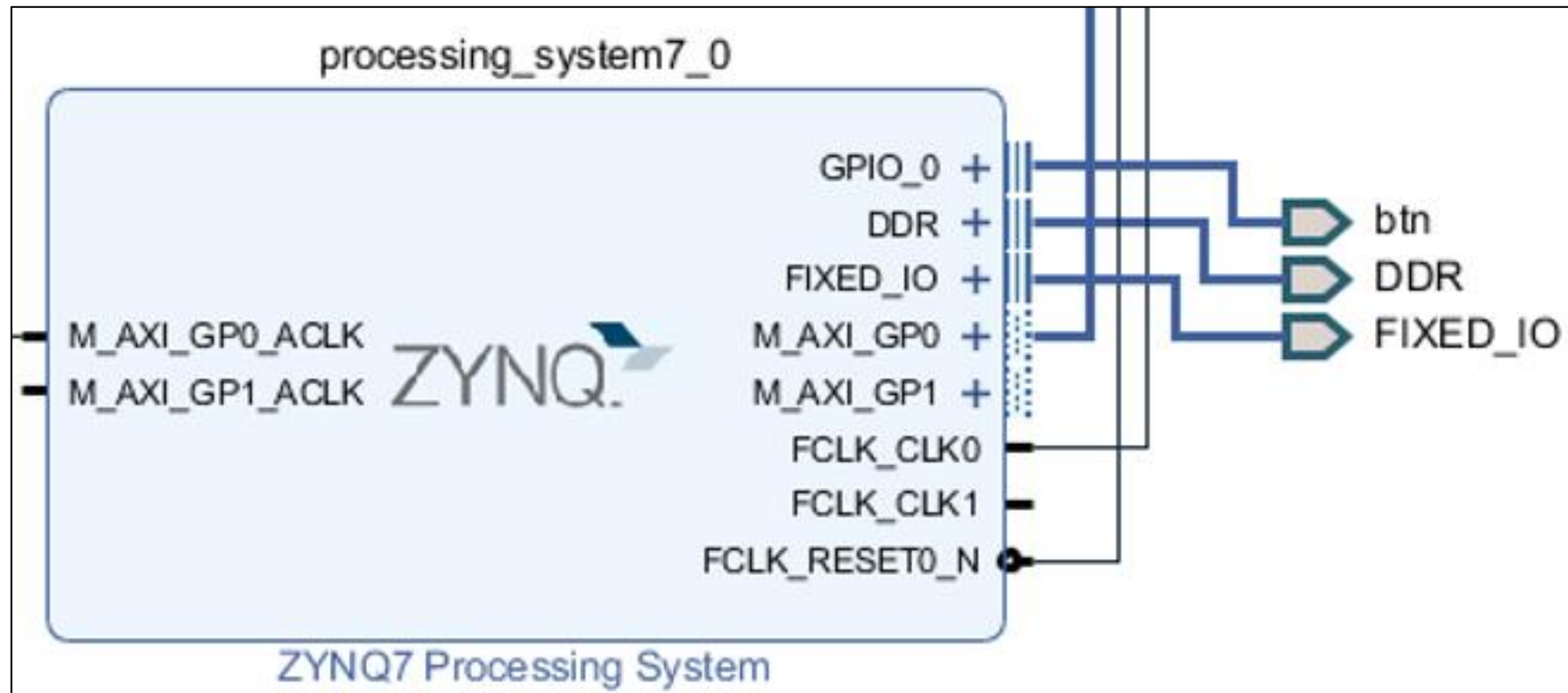
The screenshot shows the 'Clock Configuration' window in Xilinx Vivado. The 'Page Navigator' on the left lists various configuration options, with 'Clock Configuration' selected. The main area is divided into 'Basic Clocking' and 'Advanced Clocking' tabs. Under 'Basic Clocking', the 'Input Frequency (MHz)' is 33.333333 and the 'CPU Clock Ratio' is 6:2:1. Below this is a table of clock components.

Component	Clock Source	Requested Frequ...	Actual Frequency(...)	Range(MHz)
> Processor/Memory Clocks				
> IO Peripheral Clocks				
▼ PL Fabric Clocks				
<input checked="" type="checkbox"/> FCLK_CLK0	IO PLL	50	50.000000	0.100000 : 250.000000
<input checked="" type="checkbox"/> FCLK_CLK1	IO PLL	140	140.000000	0.100000 : 250.000000
<input type="checkbox"/> FCLK_CLK2	IO PLL	50	10.000000	0.100000 : 250.000000
<input type="checkbox"/> FCLK_CLK3	IO PLL	50	10.000000	0.100000 : 250.000000
> System Debug Clocks				
> Timers				

## 2. Enabling M\_AXI\_GP1 in PS

> ZYNQ 프로세서 재설정하기

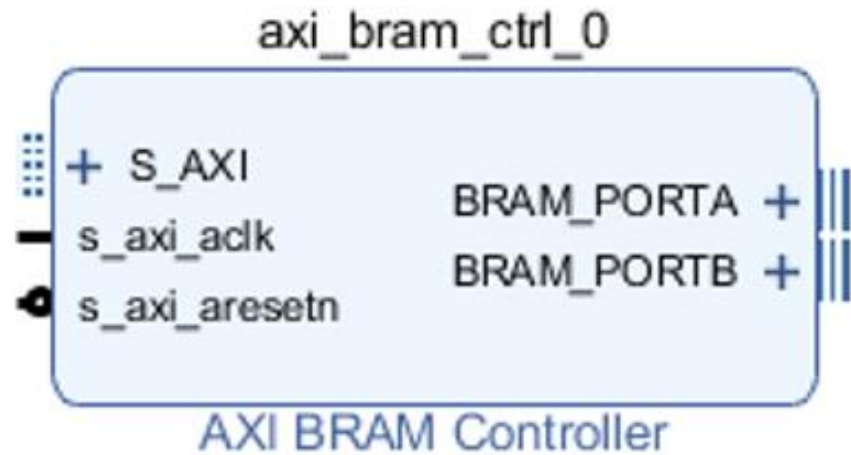
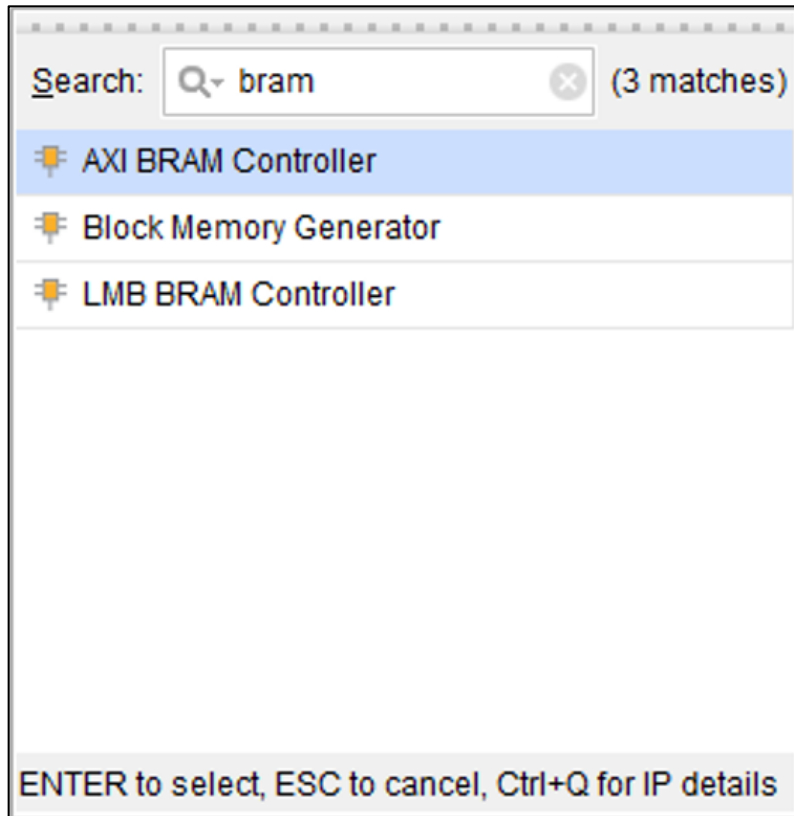
- 재설정이 완료된 PS 모습



### 3. Block RAM(BRAM) 추가하기

#### > BRAM 컨트롤러 추가하기

- Add IP에서 BRAM을 검색하여 **AXI BRAM Controller** 블록 추가



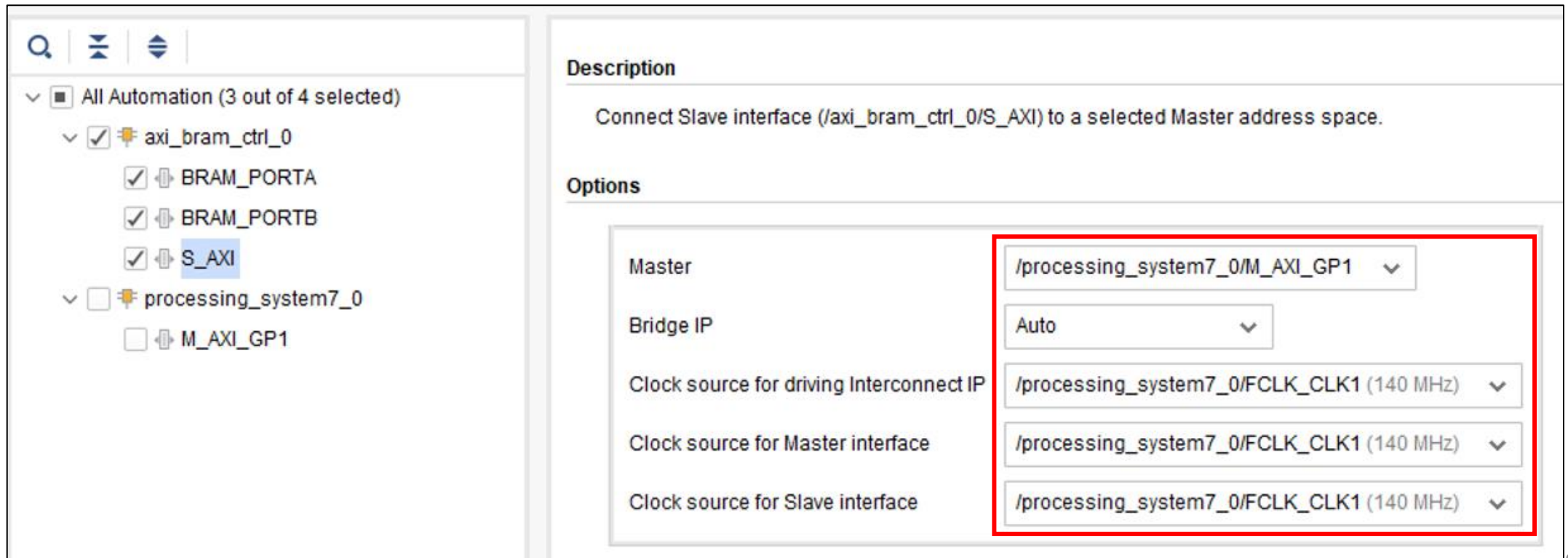


### 3. Block RAM(BRAM) 추가하기

#### > BRAM 컨트롤러 추가하기

★ Designer Assistance available. [Run Connection Automation](#)

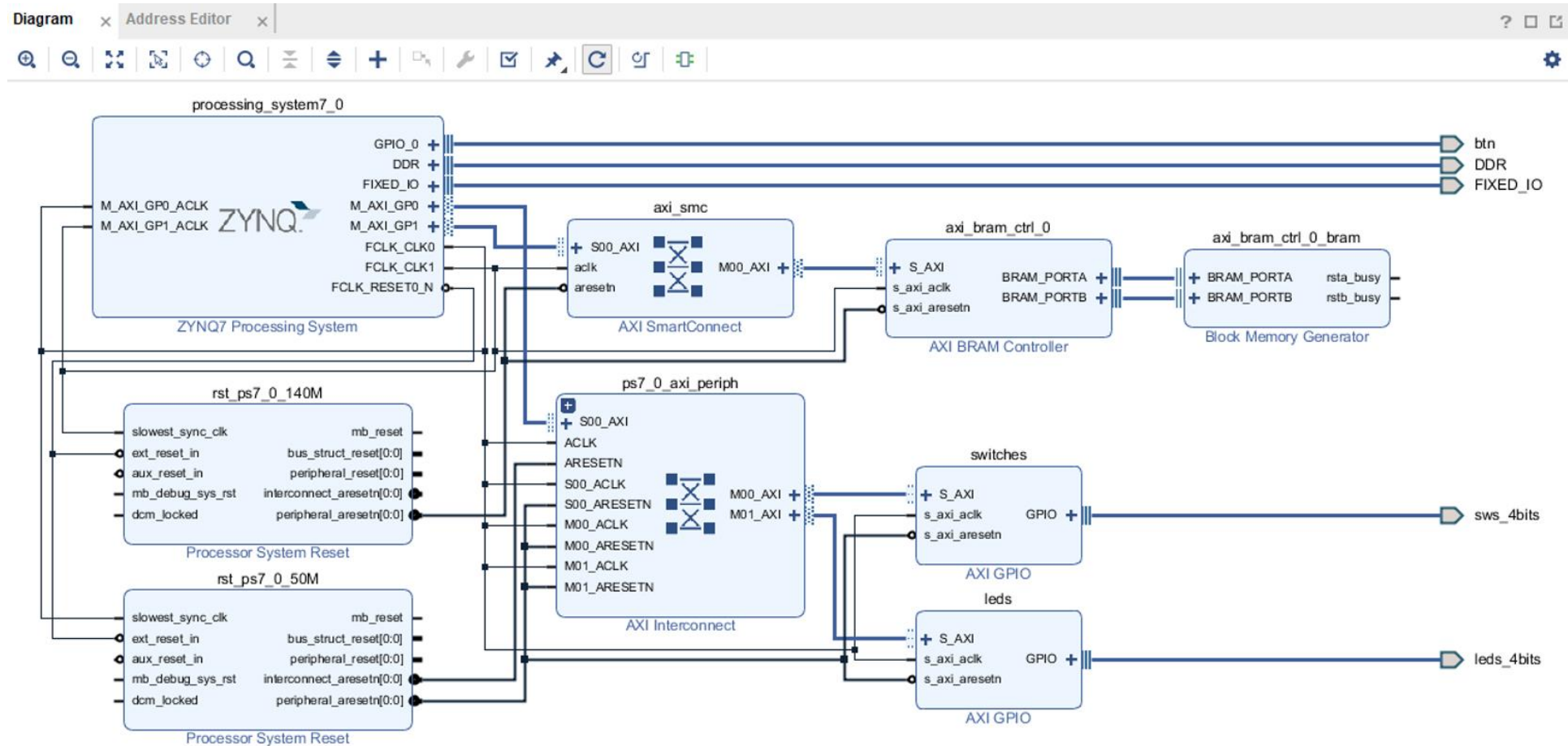
- Run Connection Automation에서 **axi\_bram\_ctrl\_0**만 체크
- 이때 **S\_AXI**의 옵션을 다음과 같이 변경 -> OK 클릭



### 3. Block RAM(BRAM) 추가하기

#### > BRAM 컨트롤러 추가하기

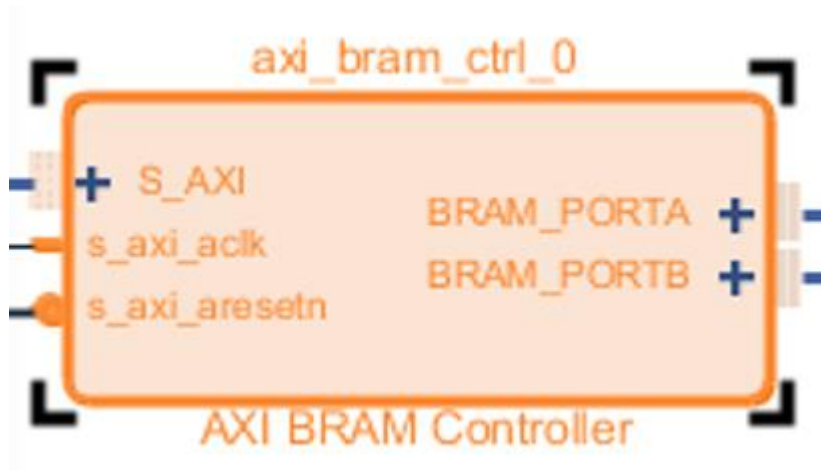
- Automation0 | 완료된 모습



### 3. Block RAM(BRAM) 추가하기

#### > BRAM 컨트롤러 추가하기

- axi\_bram\_ctrl\_0을 더블 클릭한 후 **Data Width**를 64로 변경 -> OK 클릭

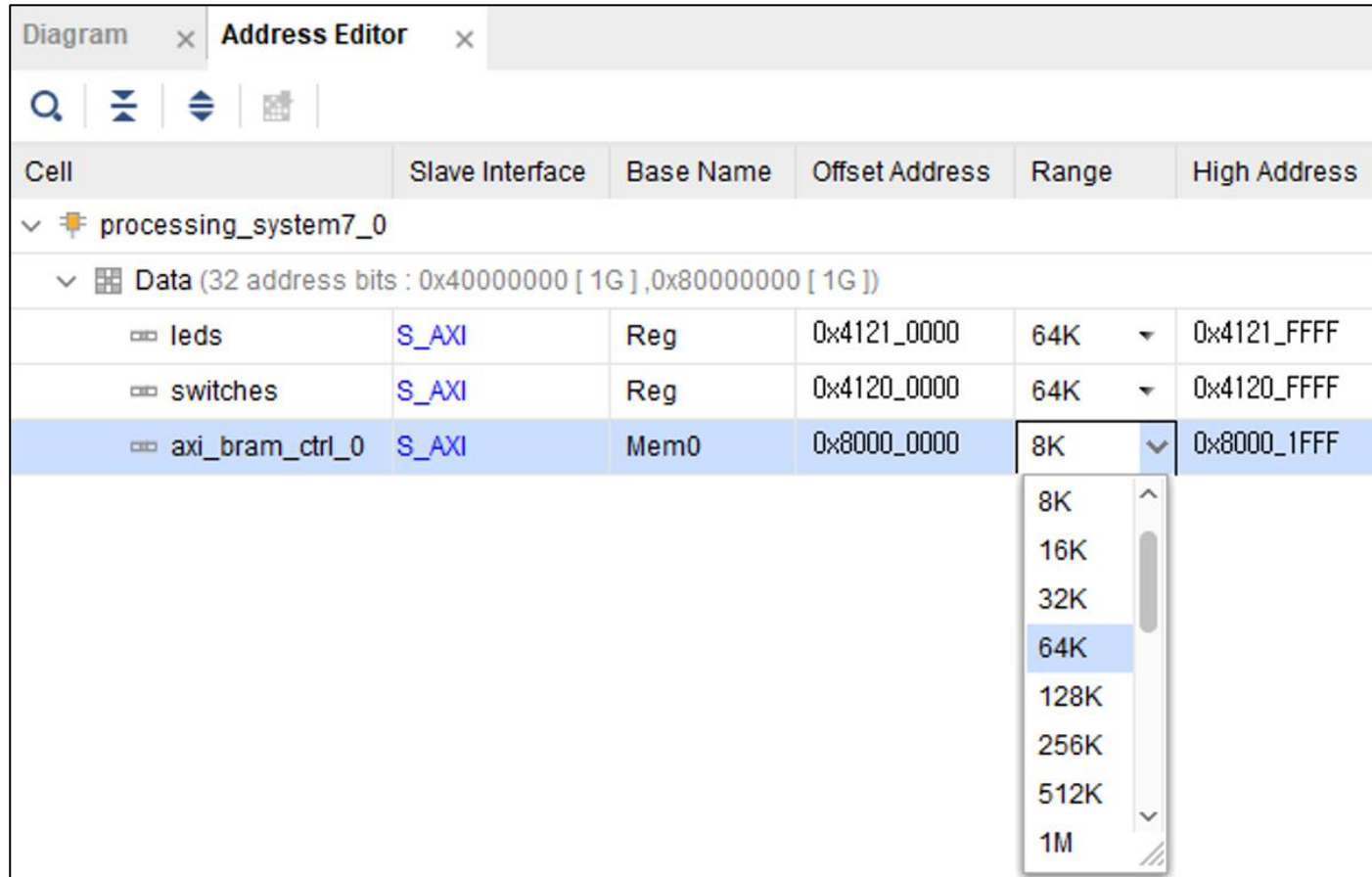


AXI Protocol	AXI4
Data Width	64
Memory Depth (Auto)	32
ID Width (Auto)	64
<input type="checkbox"/> Auto Support AXI Narrow Bursts	128
	256
	512
	1024
BRAM Options	
BRAM_INSTANCE (Auto)	External
Number of BRAM interfaces	2

### 3. Block RAM(BRAM) 추가하기

#### > BRAM 컨트롤러 추가하기

- Address Editor 탭에서 axi\_bram\_ctrl\_0의 address **range**를 **64K**로 변경



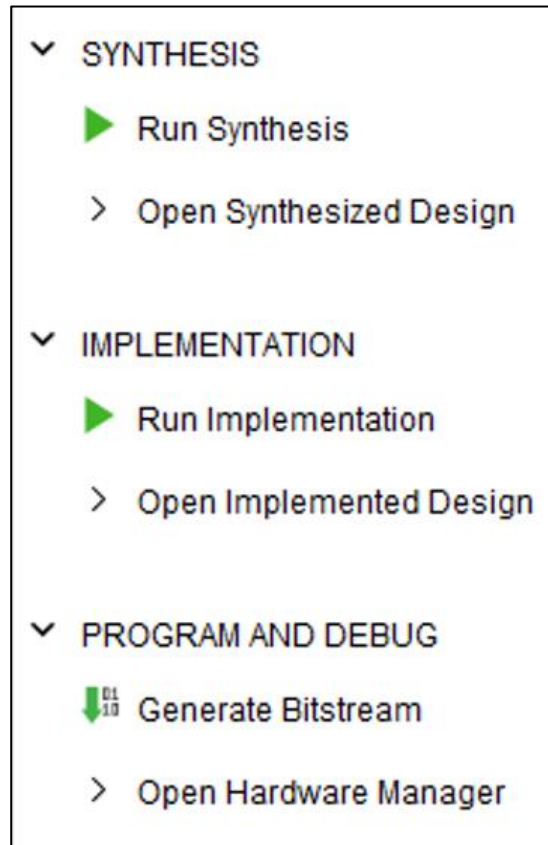
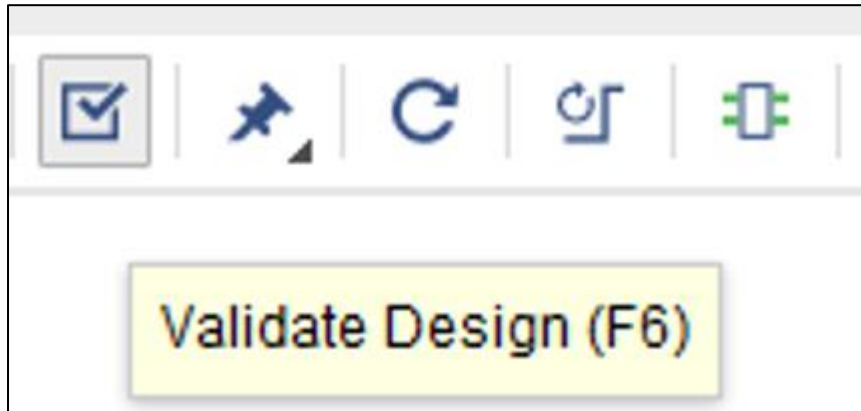
The screenshot shows the 'Address Editor' window with a table of memory components. The component 'axi\_bram\_ctrl\_0' is selected, and its 'Range' is currently '8K'. A dropdown menu is open, showing a list of range options: 8K, 16K, 32K, 64K, 128K, 256K, 512K, and 1M. The '64K' option is highlighted in blue.

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
processing_system7_0					
Data (32 address bits : 0x40000000 [ 1G ], 0x80000000 [ 1G ])					
leds	S_AXI	Reg	0x4121_0000	64K	0x4121_FFFF
switches	S_AXI	Reg	0x4120_0000	64K	0x4120_FFFF
axi_bram_ctrl_0	S_AXI	Mem0	0x8000_0000	8K	0x8000_1FFF

## 4. Bitstream 생성하기

### > Bitstream 생성하기

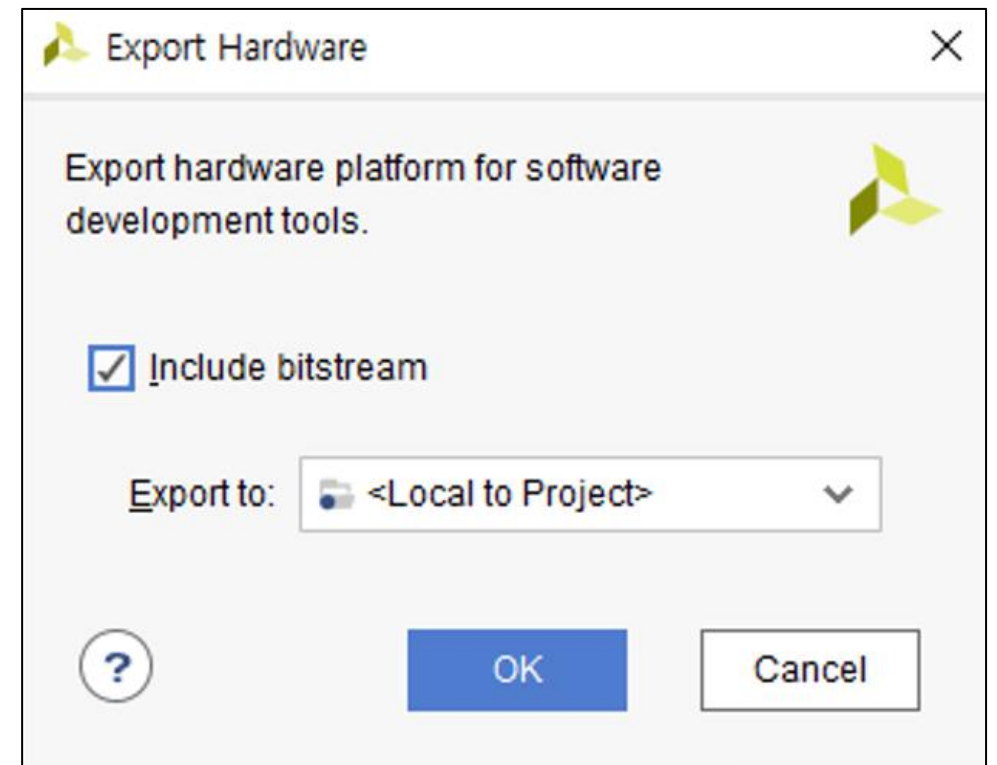
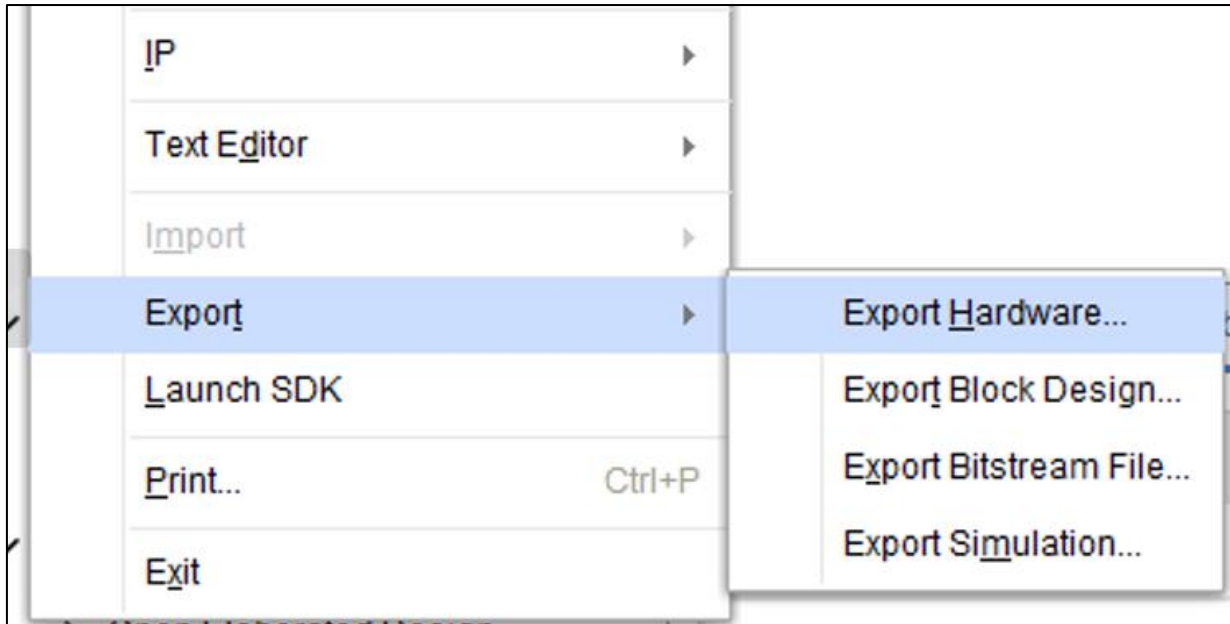
- **Validate Design**에서 error가 없다면, design을 저장한 후 **Generate Bitstream** 진행



## 5. SDK application 만들기

### > Export Hardware

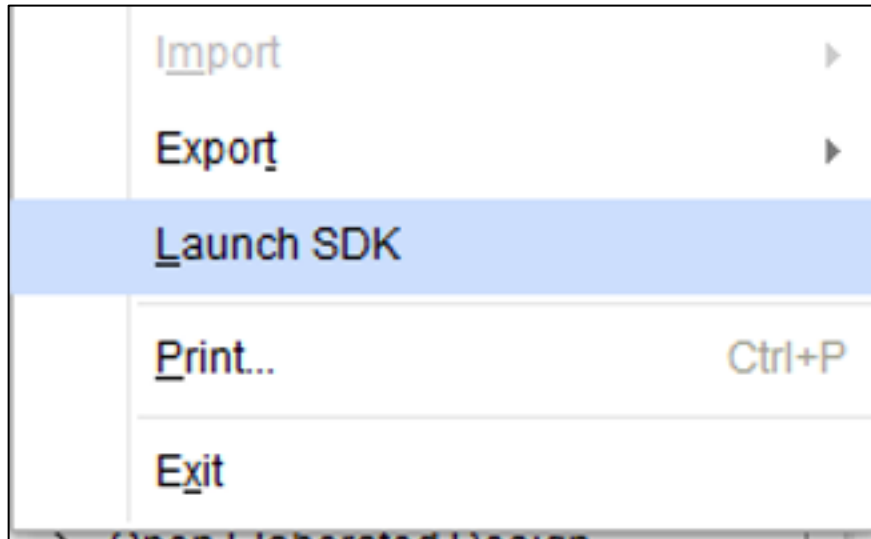
- File>Export>**Export Hardware** 클릭
- **Include bitstream** 반드시 체크!



## 5. SDK application 만들기

### > SDK 실행

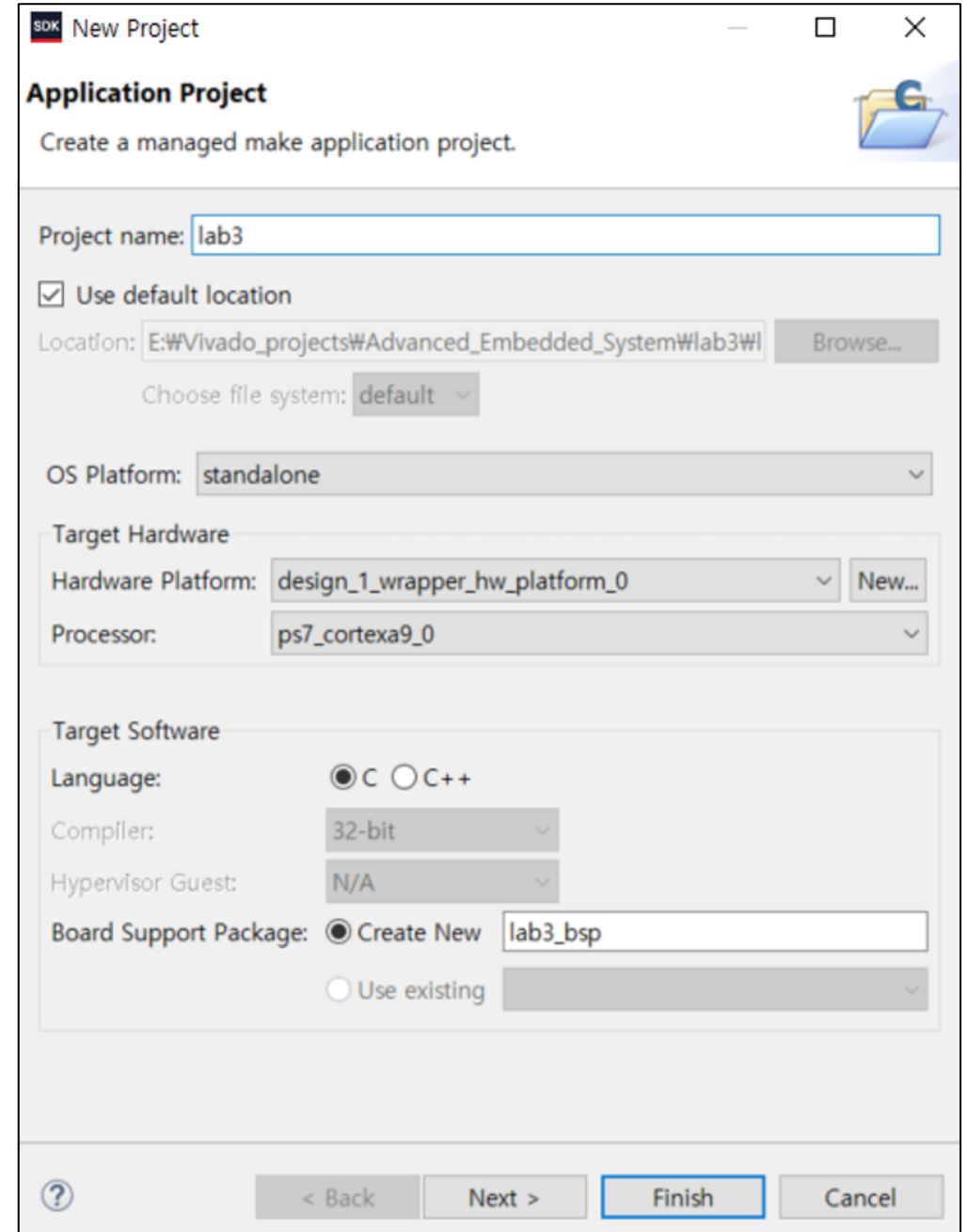
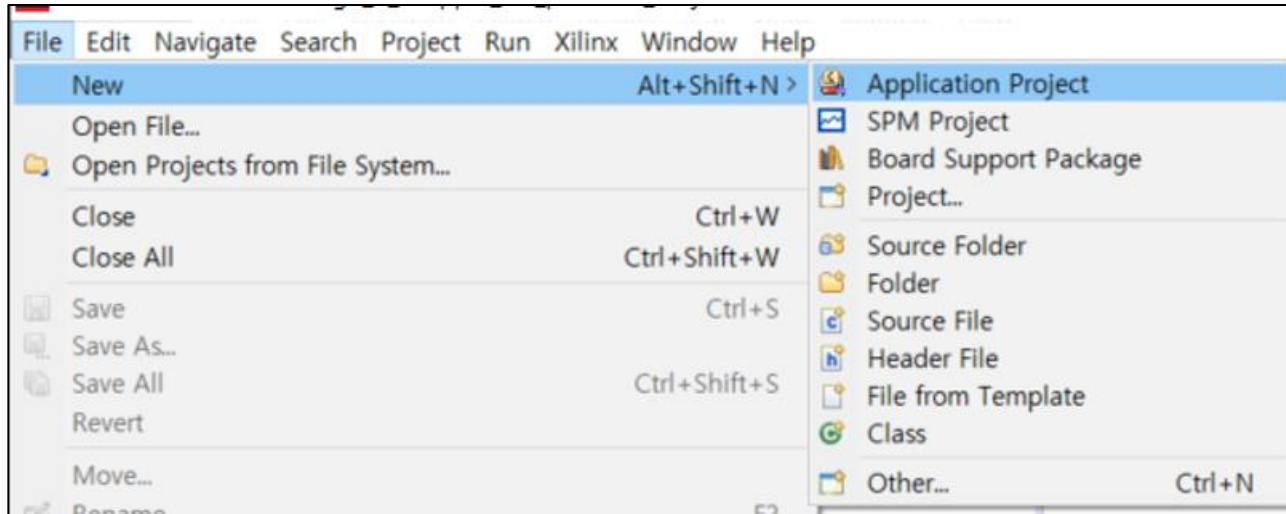
- File>Launch SDK 클릭 -> SDK IDE 창이 열림



# 5. SDK application 만들기

## > lab3 프로젝트 생성

- File>New>**Application Project** 클릭
- "lab3"로 이름 입력 후 **Next** 클릭

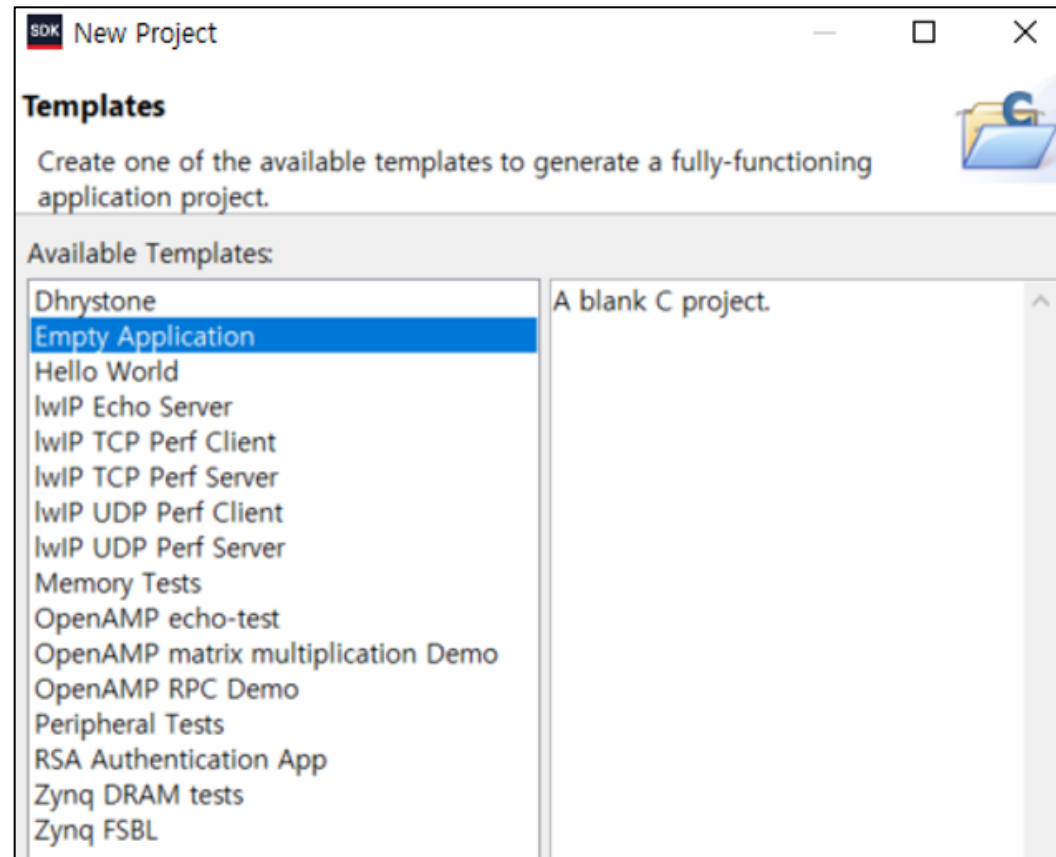




## 5. SDK application 만들기

> lab3 프로젝트 생성

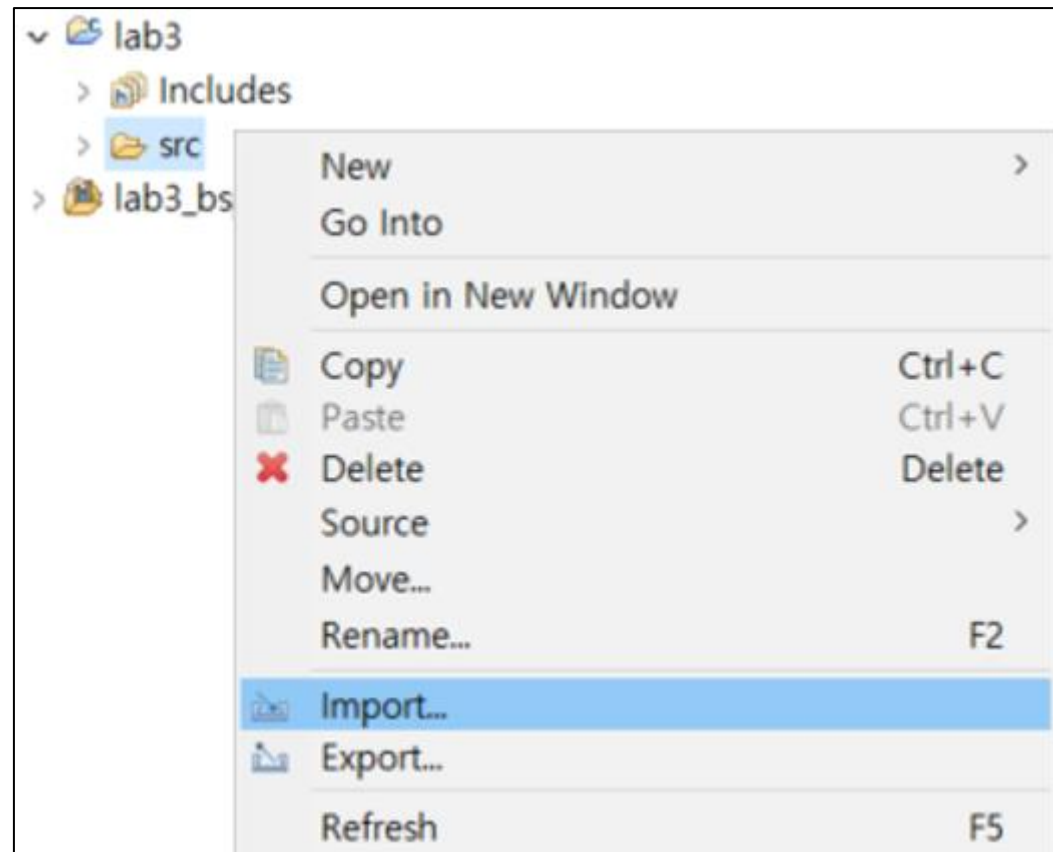
- **Empty Application** 선택 -> Finish 클릭



## 5. SDK application 만들기

> lab3.c를 import하기

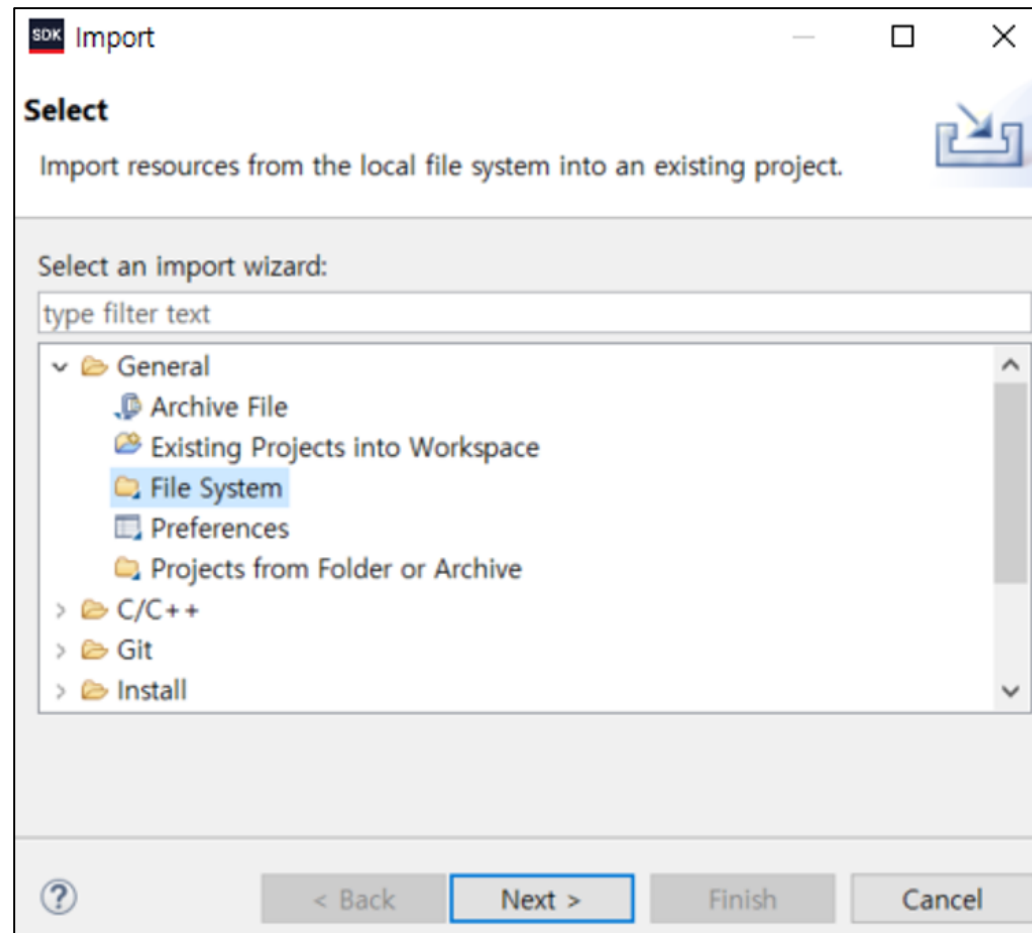
- lab3 / src > Import 클릭



## 5. SDK application 만들기

> lab3.c를 import하기

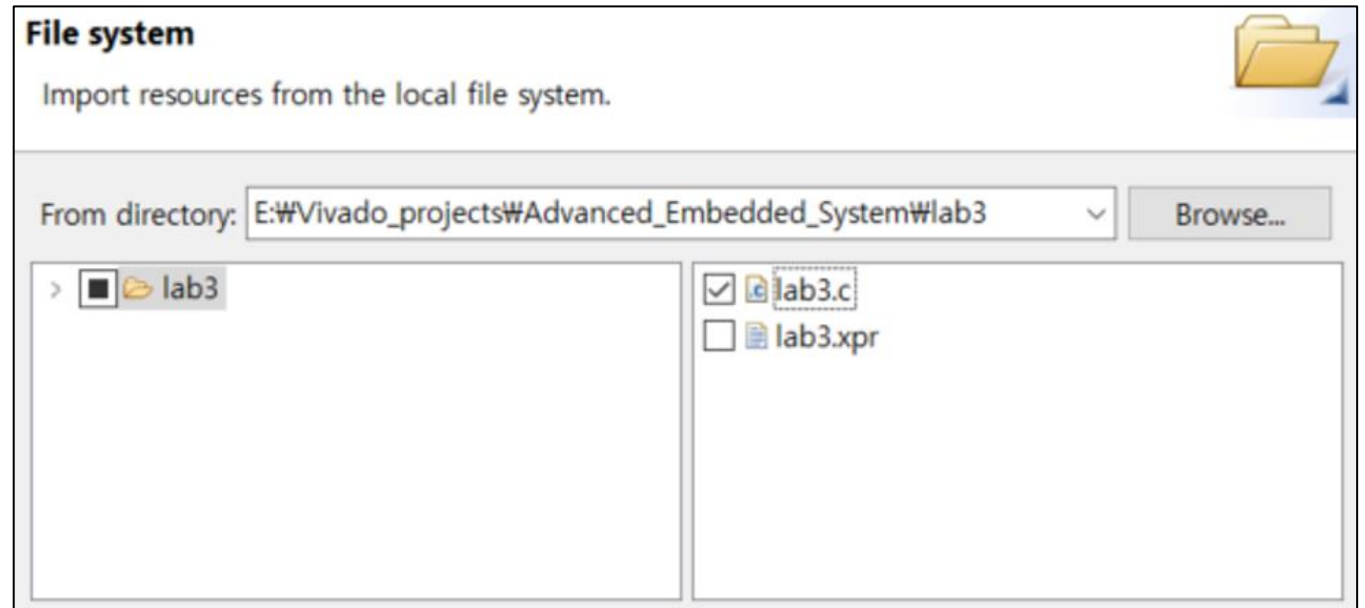
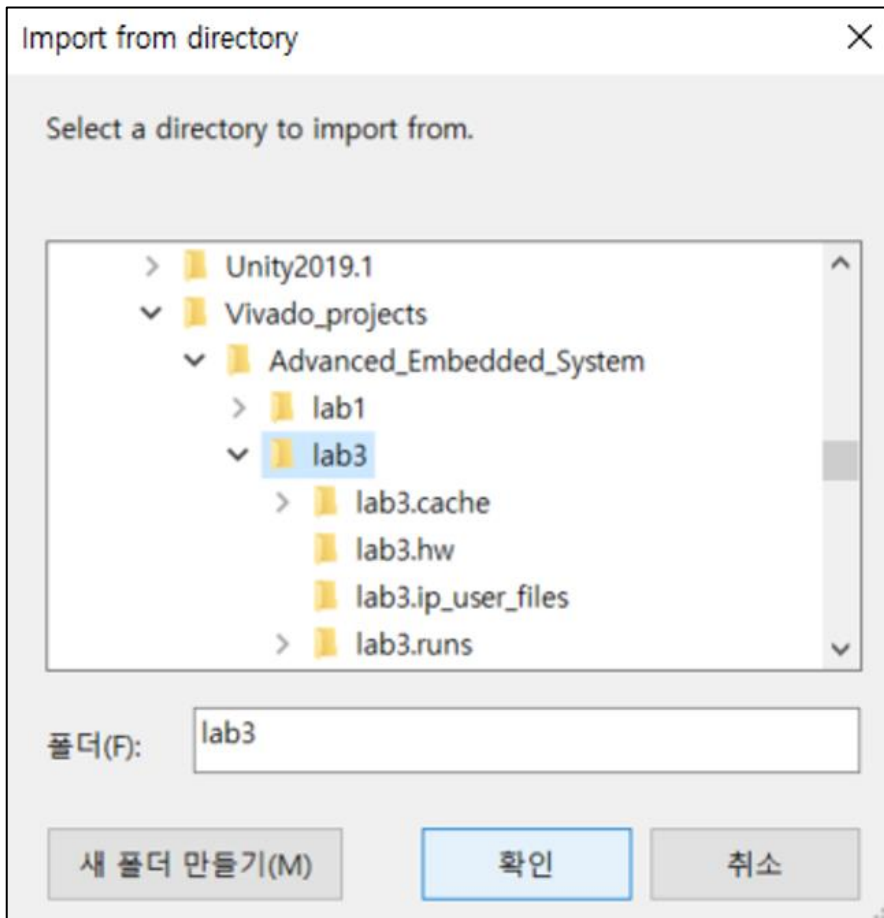
- General > File System 선택 후 Next 클릭



## 5. SDK application 만들기

### > lab3.c를 import하기

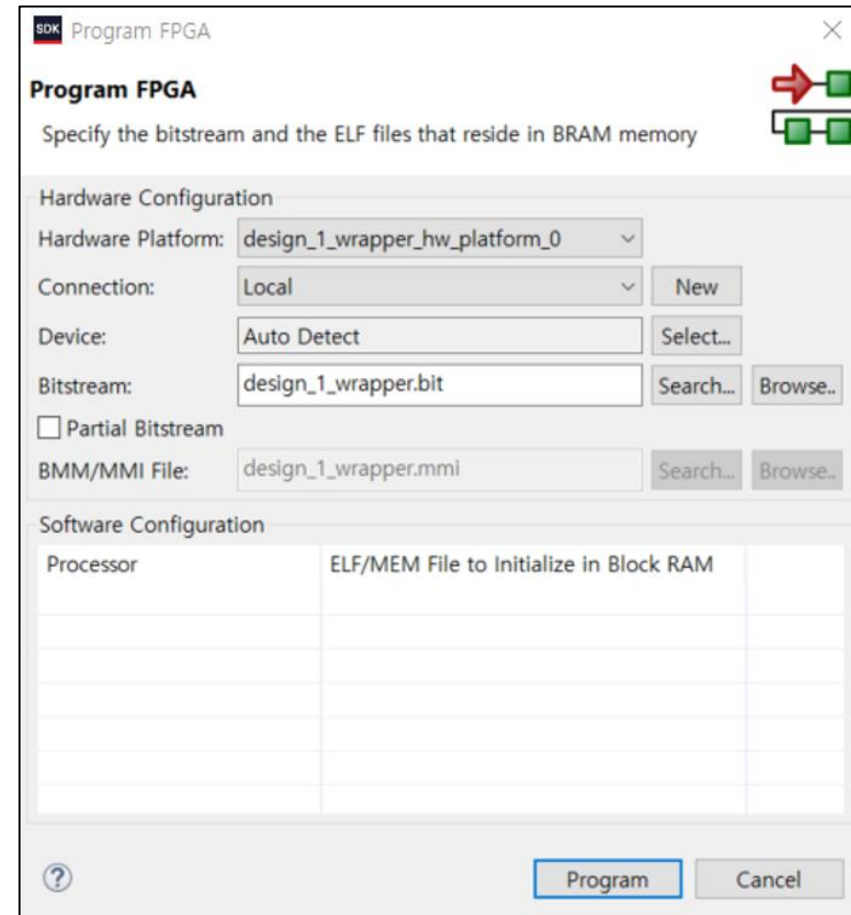
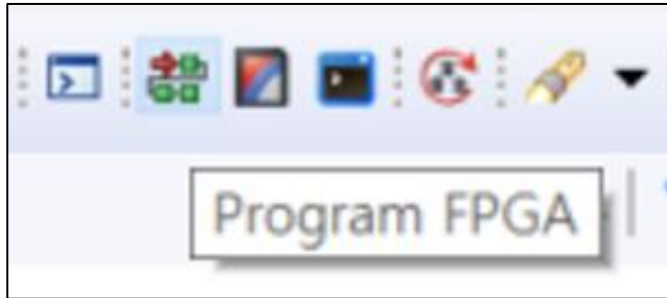
- lab3.c가 들어있는 폴더를 선택 후 lab3.c 체크 -> Finish 클릭



## 6. lab3.c 테스트하기

### > FPGA Program하기

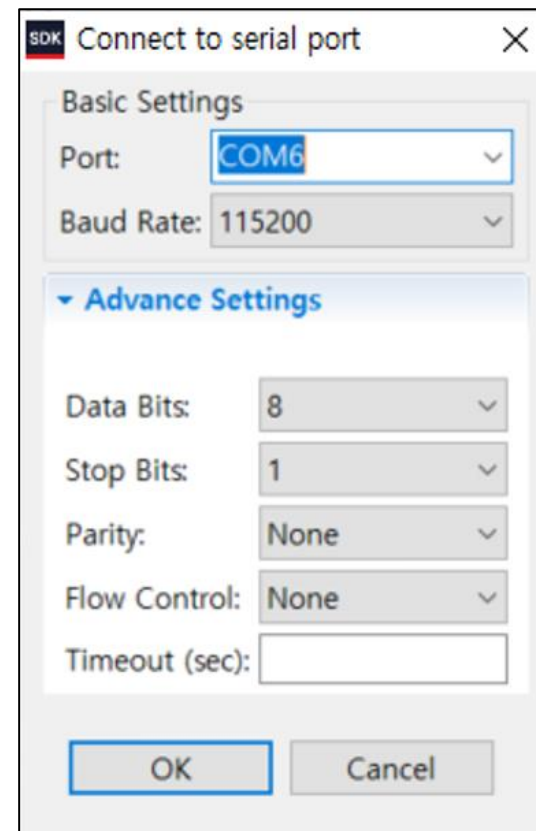
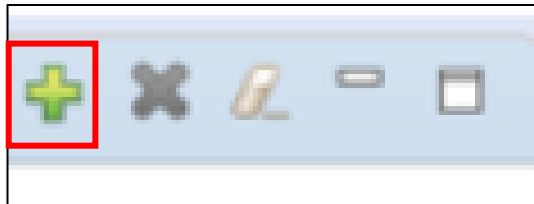
- Zybo 보드를 연결한 후, FPGA Program 버튼 클릭



## 6. lab3.c 테스트하기

### > Comport 연결하기

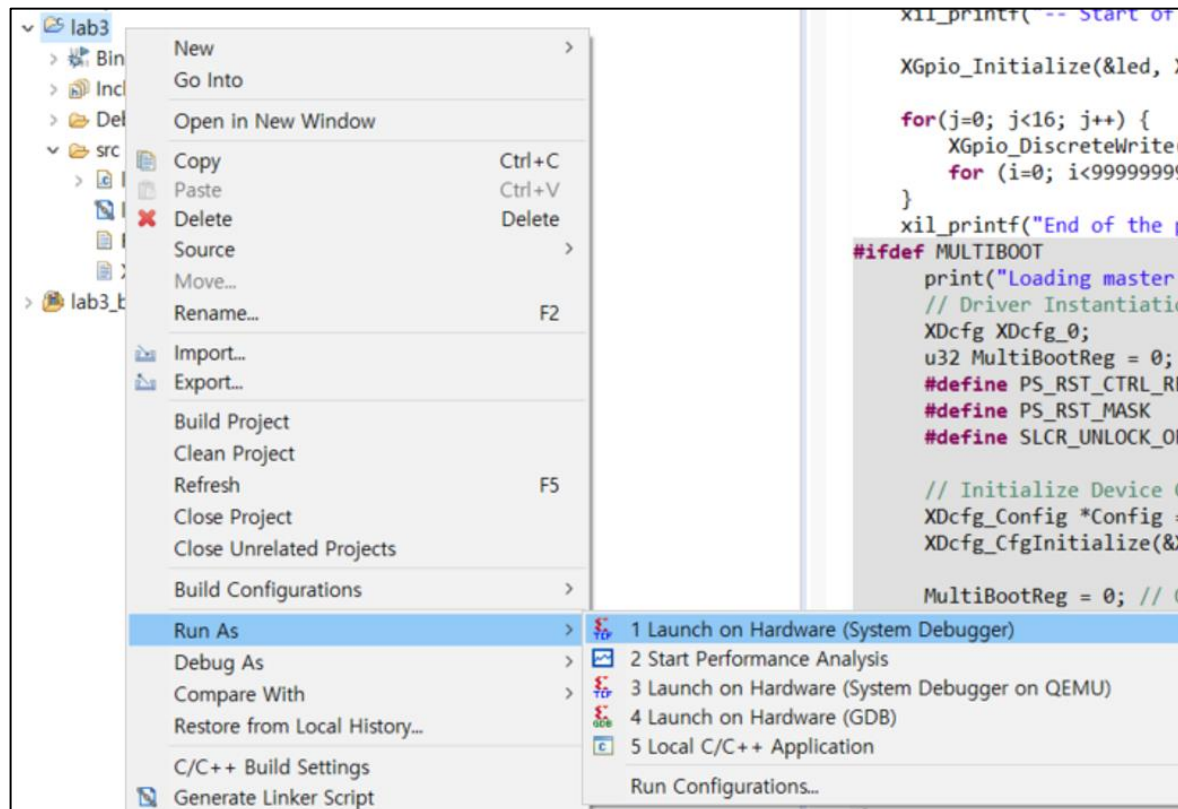
- Zybo 보드를 연결한 후, FPGA Program 버튼 클릭
- 보드가 연결된 comport 등록, baud rate : 115200



## 6. lab3.c 테스트하기

### > 보드 테스트 실행하기

- Lab3 프로젝트 폴더를 오른쪽 클릭 > **Run As** > **Launch on Hardware (System Debugger)**
- Zybo 보드의 LED가 0000 ~ 1111까지 순서대로 켜지는 것을 확인하기



**감사합니다**