

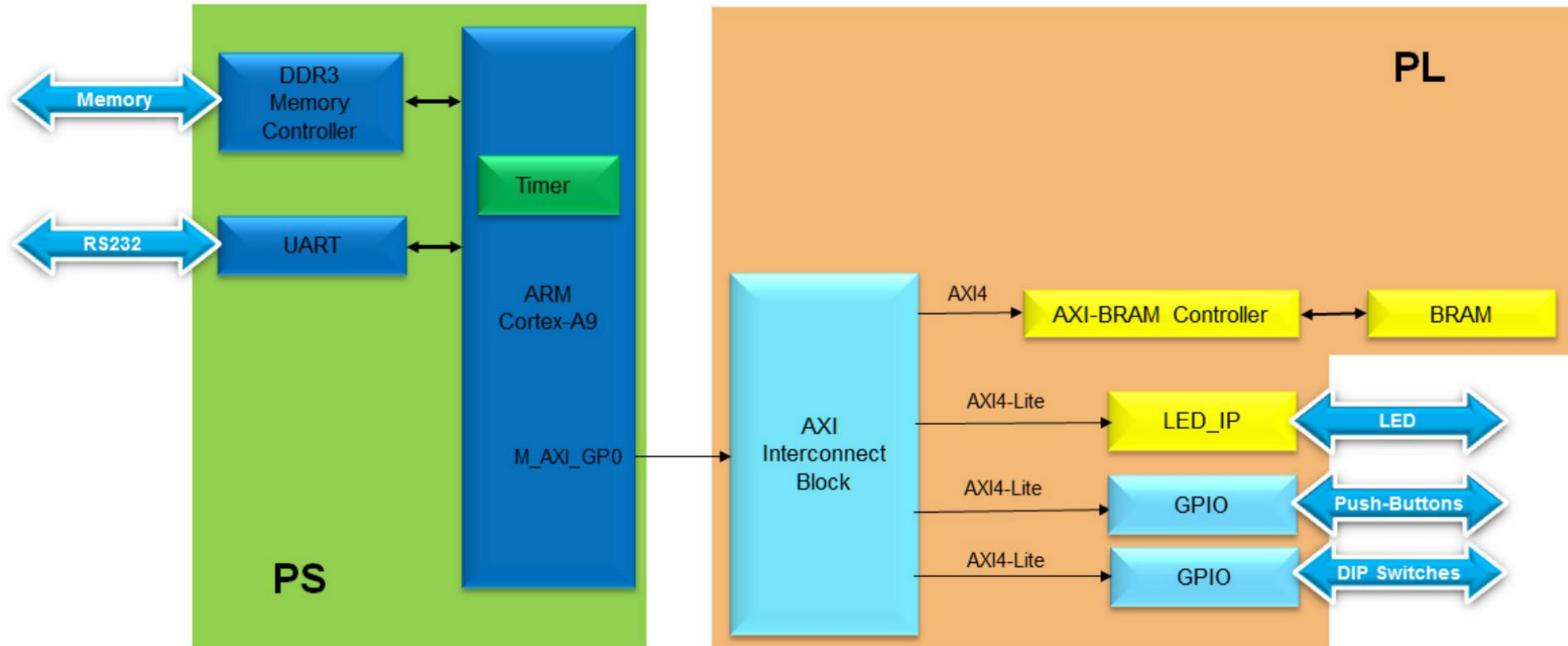
# FPGA embedded system design using AXI

With Vivado

10<sup>th</sup> February 2023

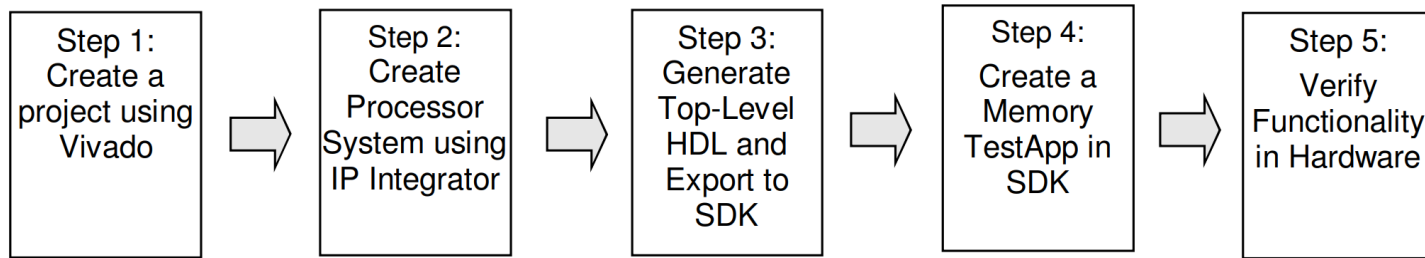
Judong Park

# Completed design

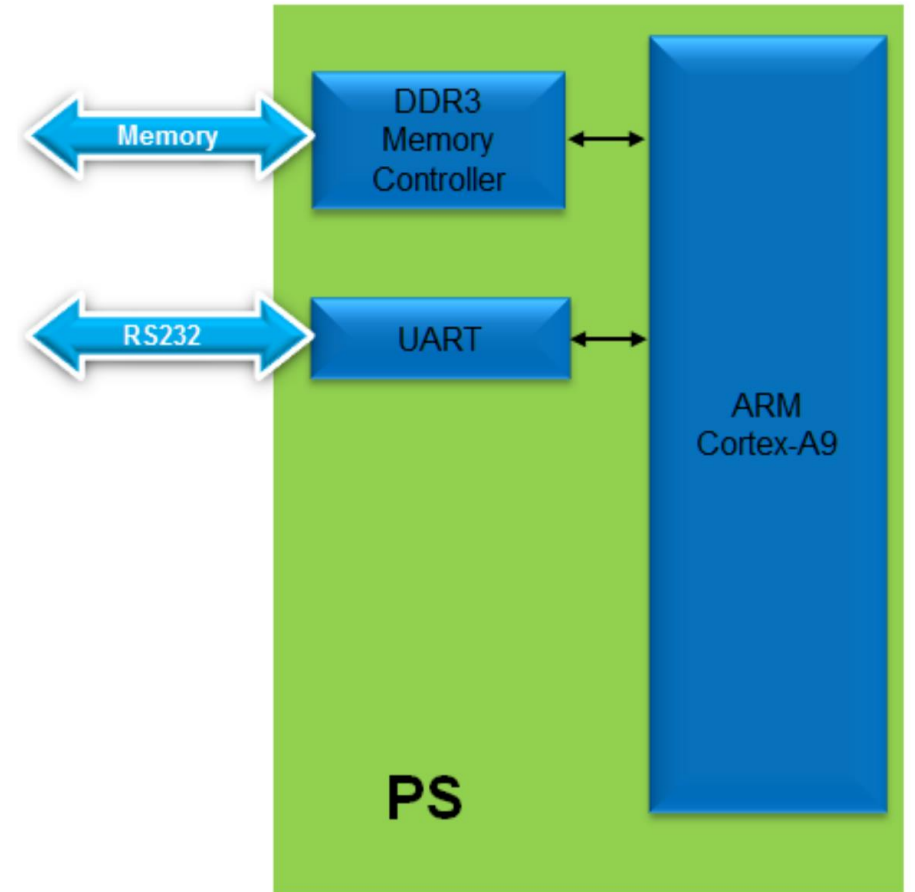


- 세미나를 통해 완성할 embedded system design
- Labs overview:
  - PS, AXI, GPIO(general Purpose Input/Output), BRAM Controller, Timer

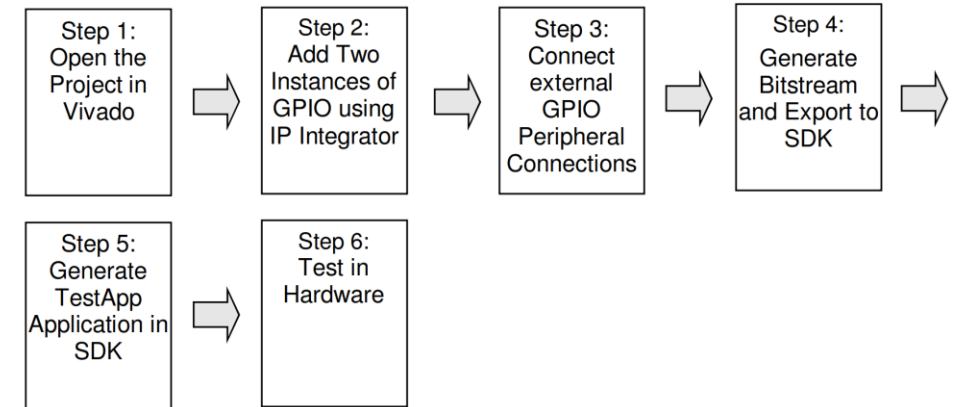
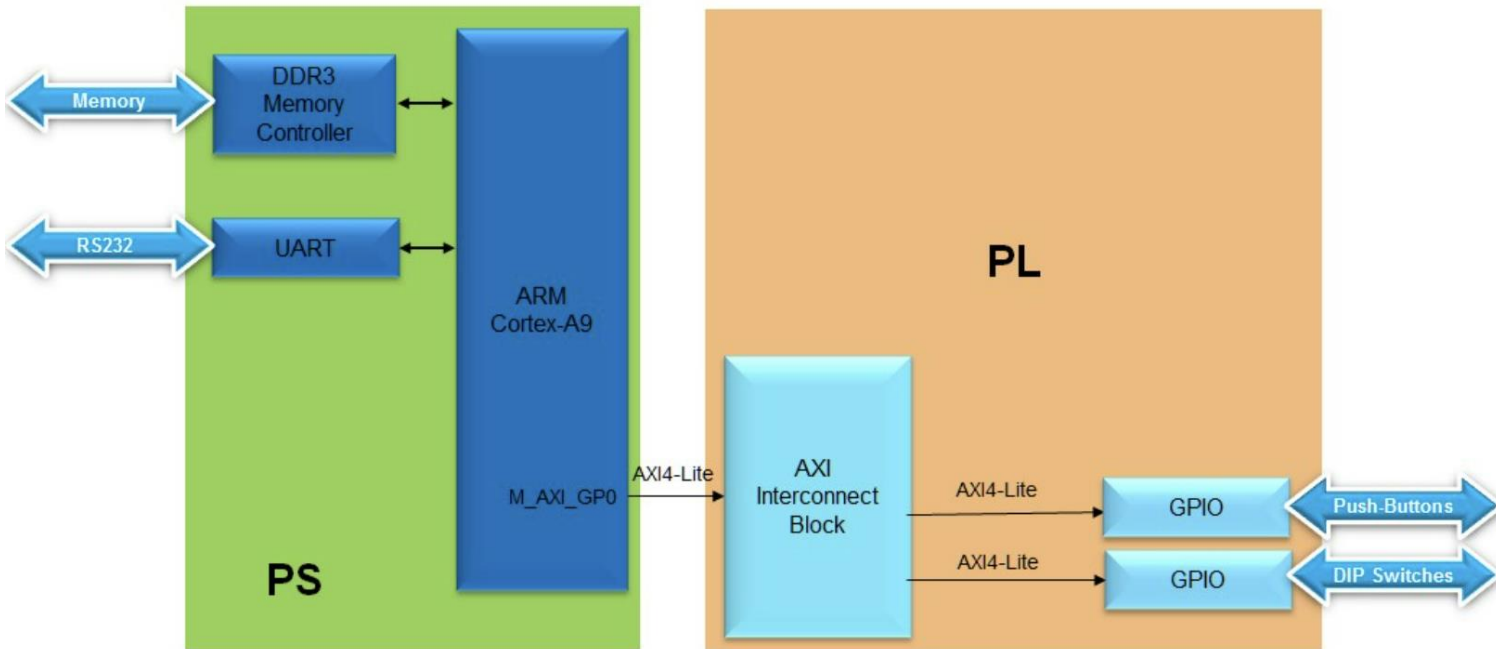
# Building an Embedded System flow



- PS 하드웨어를 sdk를 통해 사용
- 학습목표
  - Zynq system 만들기
  - IP를 이용하여 하드웨어 design
  - Sdk를 사용하여 memory test project 생성
  - Sdk를 사용하여 보드 동작



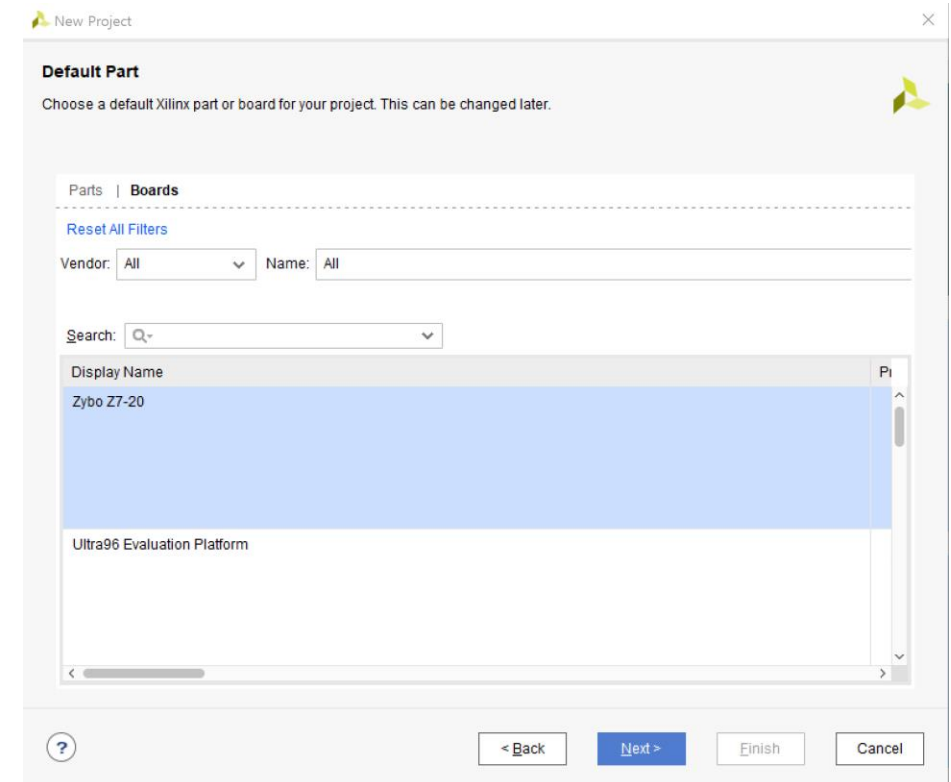
# Adding IP cores in PL flow



- GPIO IP를 추가하고 AXI를 통해 PS 영역에서 GPIO 값을 read하는 구조 설계 flow
- 학습목표:
  - PS의 GP Master port를 이용하여 PL의 IP와 연결
  - IP를 추가하여 하드웨어 설계

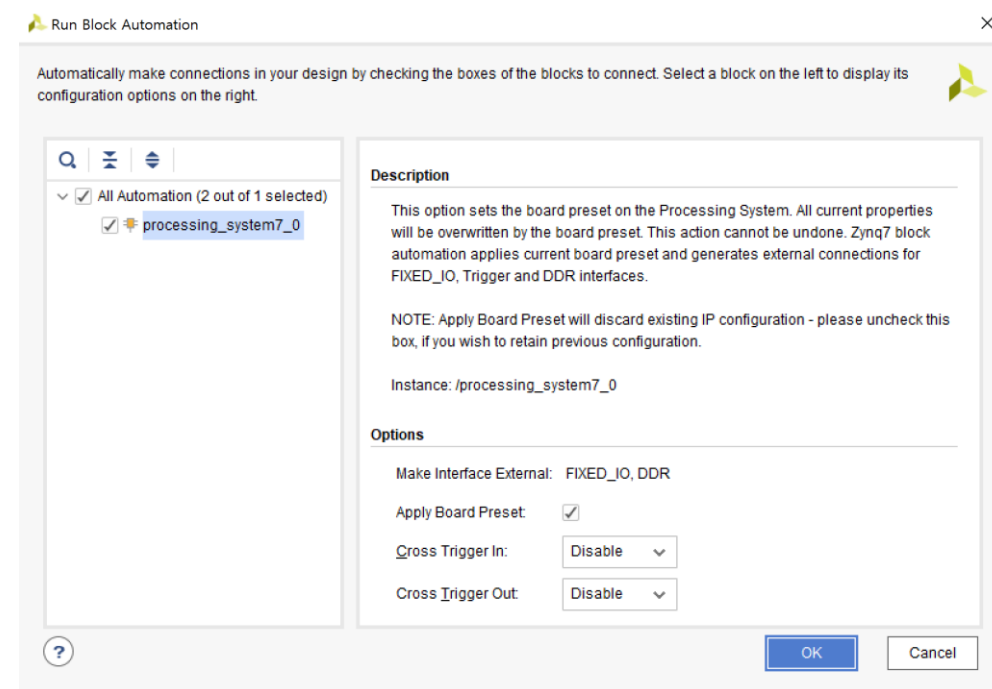
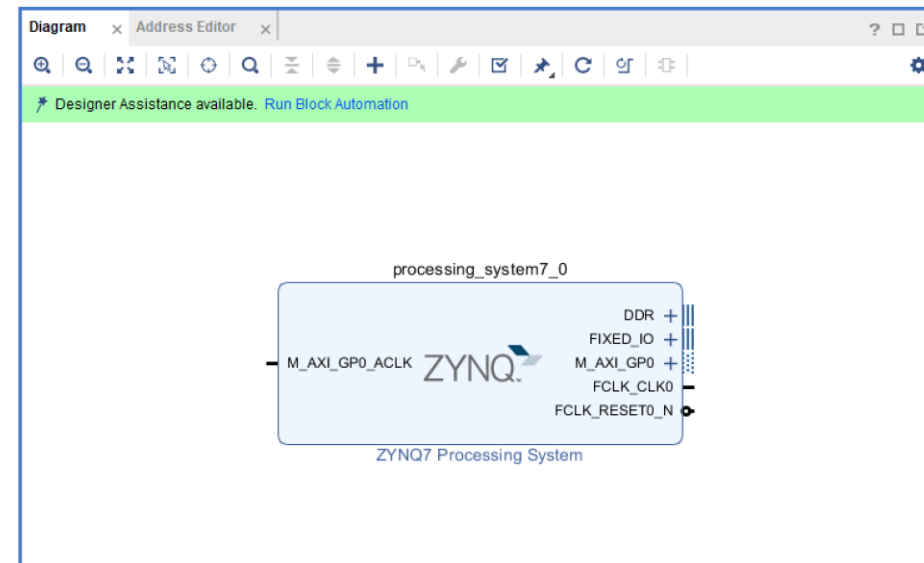
# Create a vivado project

- 프로젝트 생성 및 파일 추가
  - 보드 파일 저장: <자일링스>\Vivado\<버전>\data\boards\board\_parts
- 비바도 실행 및 New project 생성
  - Project: **fpga\_embedded\_system\_230210**
- Board 선택
  - Default Part – **Boards** 선택
  - Zybo z7-20**



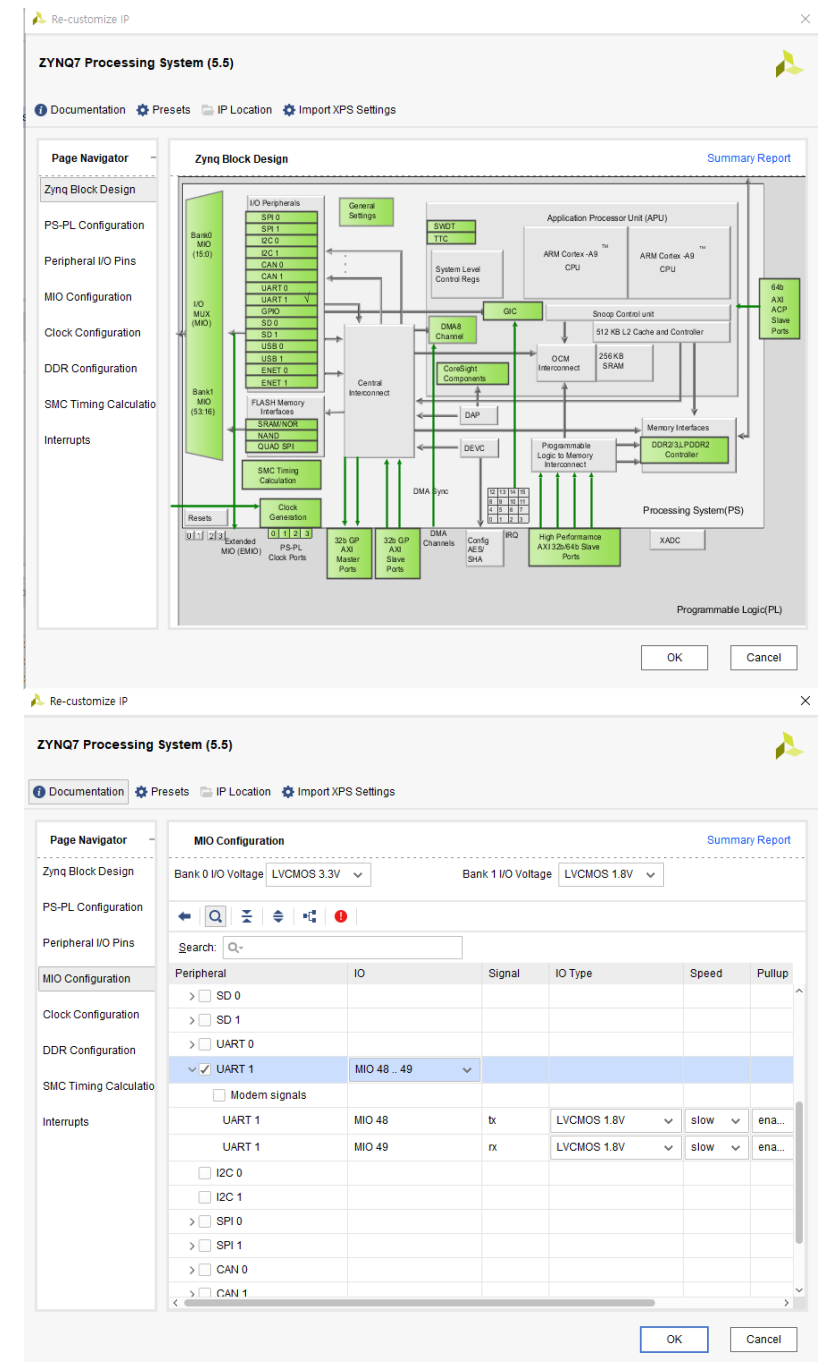
# Creating a Hardware System 1

- Block design 생성
  - Project manager – **create block design** 클릭
  - Design name: **design\_1**
- IP 추가
  - Add IP 클릭
  - Zynq processing system IP 추가
    - IP: **Zynq7 processing system**
- 보드 preset 설정
  - Run block automation 실행



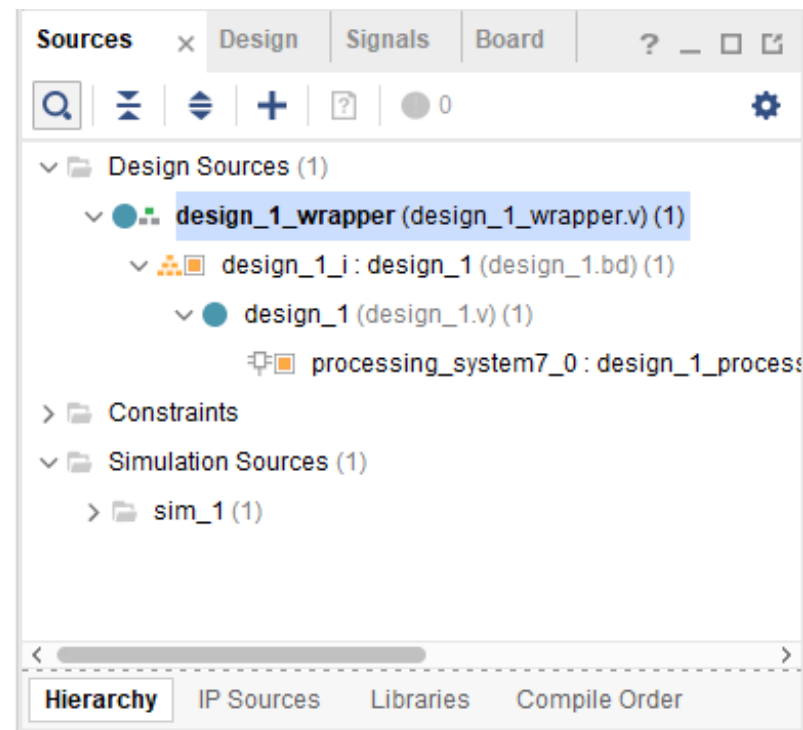
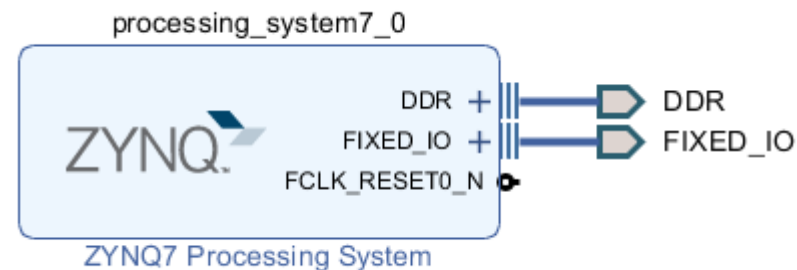
# Creating a Hardware System 2

- PS 커스터마이징
  - 사용할 PS 부분을 설정
- MIO configuration:
  - UART1 체크
  - UART1: Bank 1 I/O voltage: 1.8V 전압 설정
  - 나머지 인터페이스 모두 disabled
    - ENET, USB, SD, GPIO, QSPI
- Application processor unit:
  - timer 체크 해제
- PS-PL configuration - AXI non secure enablement – GP master AXI interface:
  - M AXI GP0 체크 해제
- Clock configuration – PL fabric clocks:
  - FCLK\_CLK0 체크 해제



# Creating a Hardware System 3

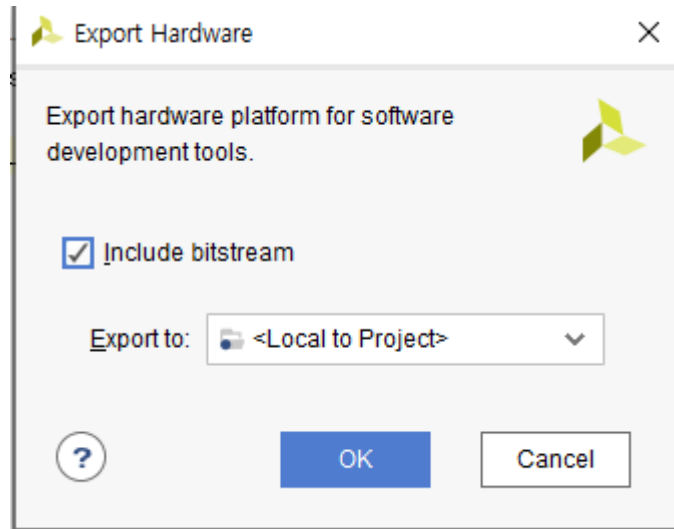
- IP 외부 I/O 핀 연결
  - Run Block automation 클릭 – OK
- IP 오류 유무 검증
  - Validate design 클릭
- Block design을 Verilog 코드로 변환
  - Sources – design source – design\_1 오른쪽 클릭
  - Create HDL wrapper
    - Let vivado manage wrapper and auto-update 클릭





# Generating Top-level and export to SDK

- 비트스트림 생성
  - Generate bitstream 클릭
- Sdk 실행
  - File – export hardware
    - Include bitstream 체크
  - File – launch sdk



**SDK**  
Software Development Kit

2018.2

 **XILINX.**

Copyright 1986-2018 Xilinx, Inc.  
All Rights Reserved.

# Generating Memory TestApp in SDK

- 프로젝트 생성
  - File – new – application project
  - Project name: **fpga\_embedded\_system\_230210**
  - 템플릿: **memory Tests**
  - Finish 클릭

SDK New Project

Application Project

Create a managed make application project.

Project name: fpga\_embedded\_system\_230206

☒ Use default location

Location: C:\WPFGA\_2023\_seminar\wpfga\_230206\WPFGA\_embedded\_syst Browse...

Choose file system: default

OS Platform: standalone

Target Hardware

Hardware Platform: design\_1\_wrapper\_hw\_platform\_0 New...

Processor: ps7\_cortexa9\_0

Target Software

Language: ☒ C ☐ C++

Compiler: 32-bit

Hypervisor Guest: N/A

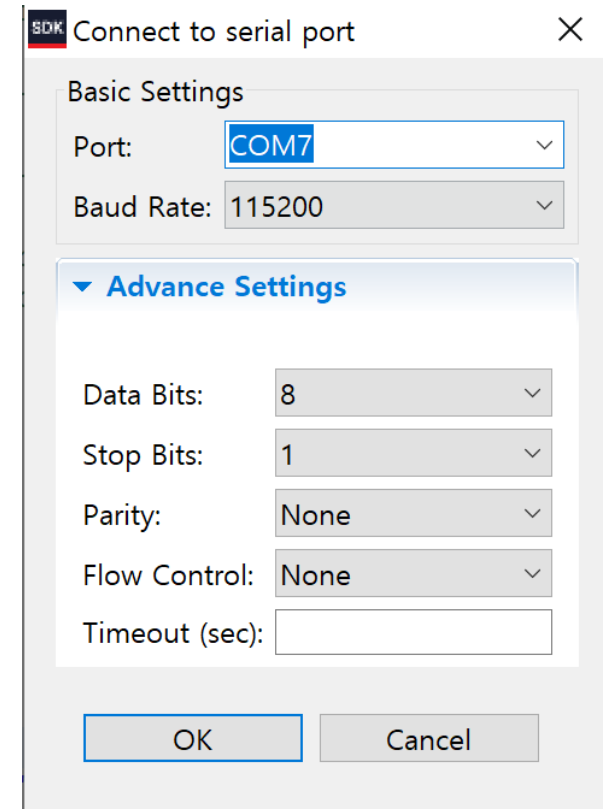
Board Support Package: ☒ Create New fpga\_embedded\_system\_230206\_bsp

☐ Use existing

? < Back Next > Finish Cancel

# Testing in Hardware

- FPGA에 코드 프로그래밍
  - **FPGA Program** 클릭: FPGA에 bitstream 업로드
  - 생성한 fpga\_ps\_230203 프로젝트 우클릭
  - **Build project**
  - **Sdk 터미널 연결**
    - Port: 장치관리자에서 FPGA와 연결된 포트 확인
  - 생성한 fpga\_ps\_230203 프로젝트 우클릭
  - Run as – **launch on hardware**
- 동작 확인
  - 터미널 콘솔창에서 결과 확인 가능



NOTE: This application runs with D-Cache disabled.As a result, cacheline requests will not be

Testing memory region: ps7\_dds\_0

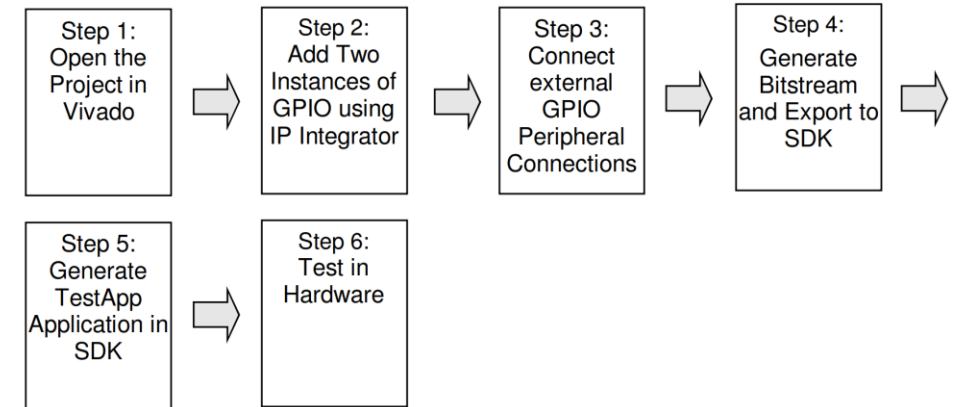
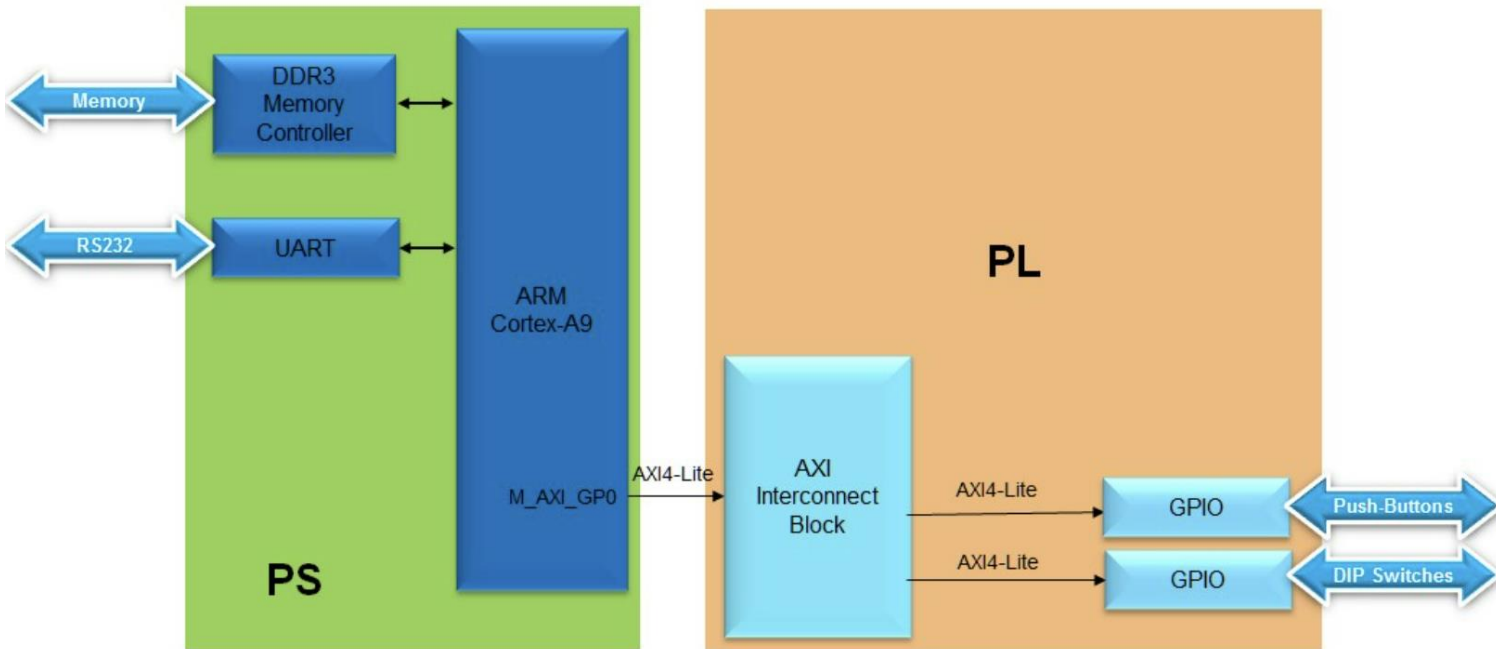
Memory Controller: ps7\_dds\_0

Base Address: 0x100000

Size: 0x3FF00000 bytes

32-bit test: PASSED!

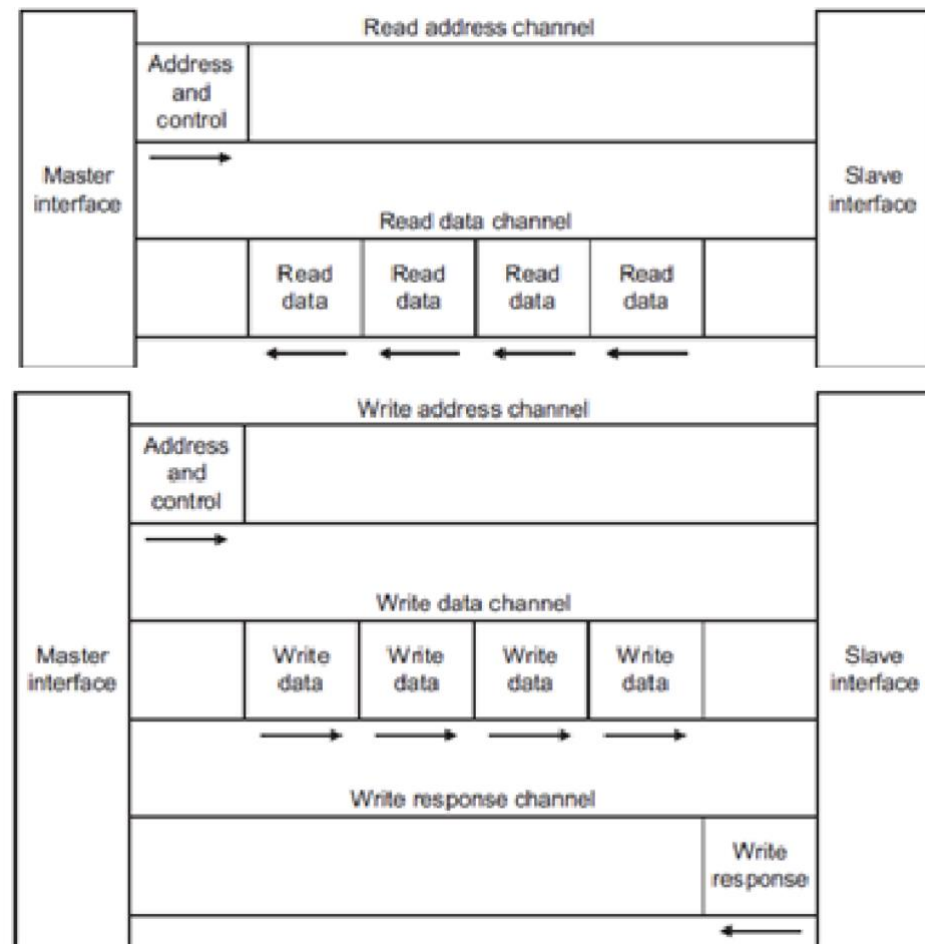
# Adding IP cores in PL flow



- GPIO IP를 추가하고 AXI를 통해 PS 영역에서 GPIO 값을 read하는 구조 설계 flow
- 학습목표:
  - PS의 GP Master port를 이용하여 PL의 IP와 연결
  - IP를 추가하여 하드웨어 설계

# AXI Interface

- Read
  - Read address channel
  - Read data channel
- Write
  - Write address channel
  - Write data channel
  - Write response channel
- Axi protocol
  - Valid/ready handshake
    - 데이터 보내기 전:
      - Source (출발지): 유효한 data를 보낼 수 있을 때, valid 신호 on
      - Destination (도착지): data를 받을 수 있다면 ready 신호 on
  - Valid, ready 신호 모두 1인 경우 ➔ 데이터 전송
    - Source (출발지): 더 이상 전송할 데이터가 없다면 valid 신호 off
    - Destination (도착지): data를 더 이상 받을 수 없다면, ready 신호 off

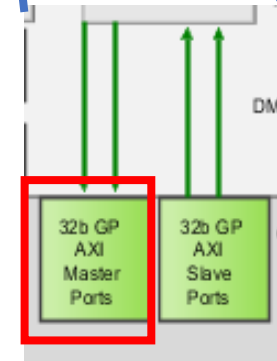
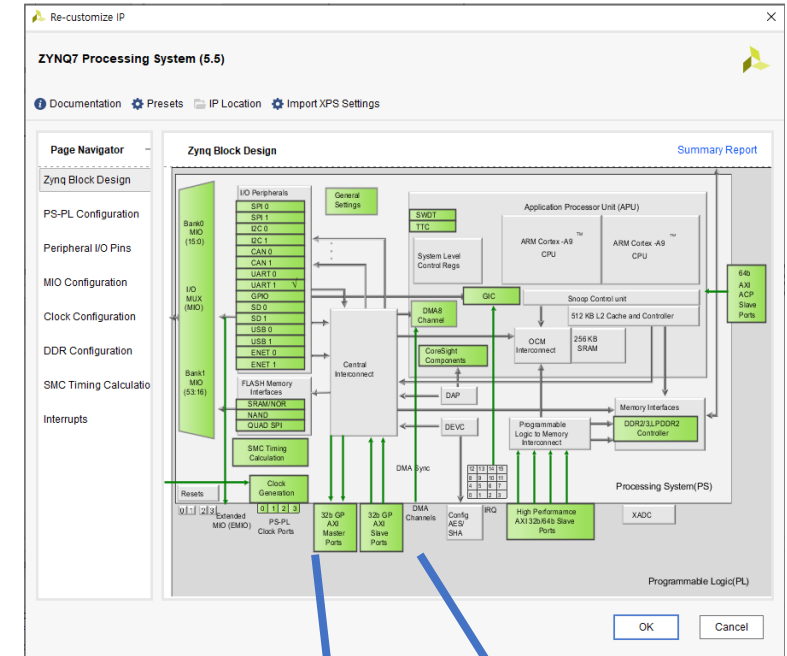


## AXI 전달 채널

Global	Write address channel	Write data channel	Write response channel	Read address channel	Read data channel
ACLK	AWVALID	WVALID	BVALID	ARVALID	RVALID
ARESETn	AWREADY	WREADY	BREADY	ARREADY	RREADY
-	AWADDR	WDATA	BRESP	ARADDR	RDATA
-	AWPROT	WSTRB	-	ARPROT	RRESP

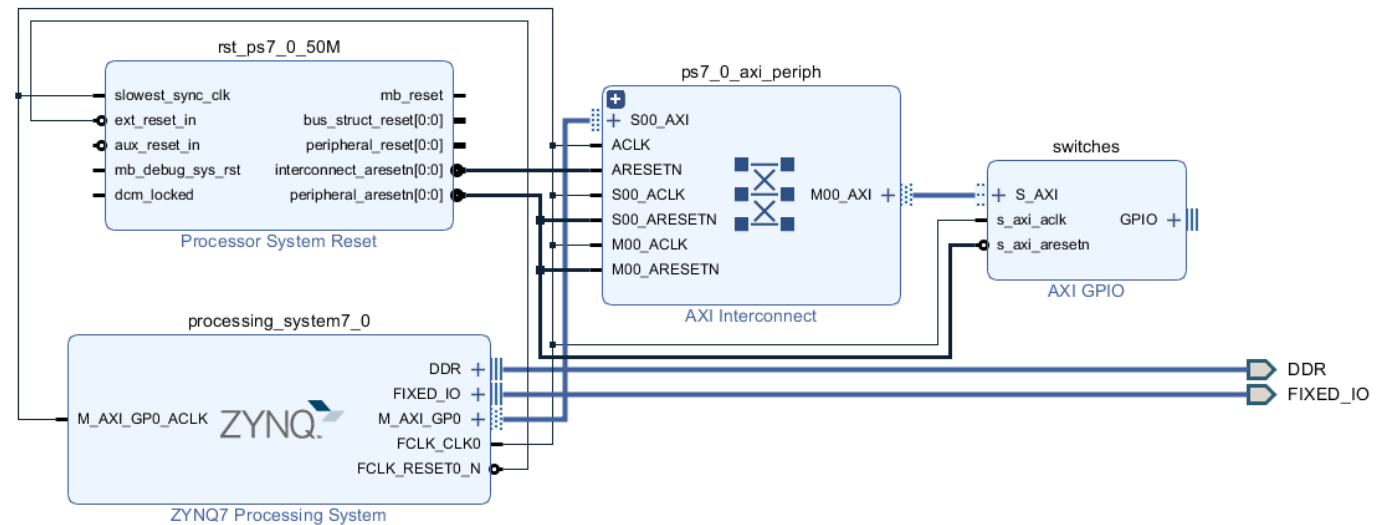
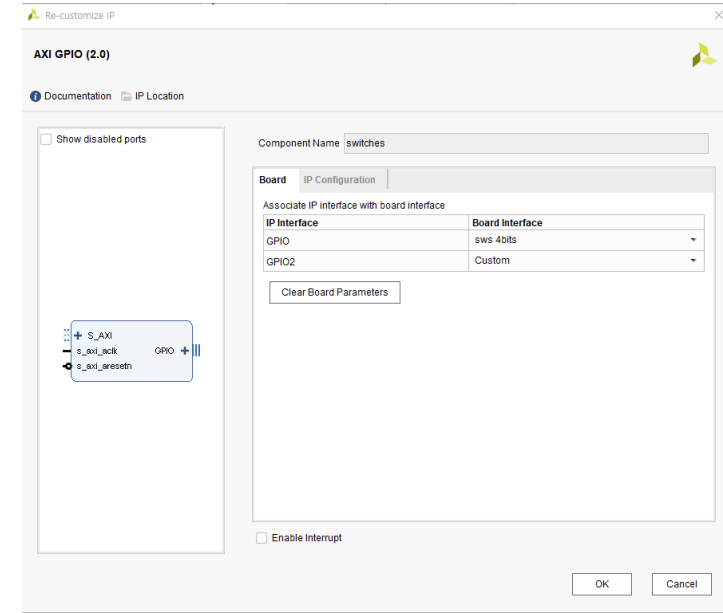
# Adding Two Instances of GPIO 1

- 이전 실습 프로젝트에 이어서...
- PS IP 커스터마이징
  - AXI\_Master, Clock signal enable하도록 설정
  - PS-PL Configuration – 32b GP AXI Master Port:
    - AXI\_M\_GP0 interface 체크
  - General – enable clock resets:
    - FCLK\_RESET0\_N 체크
  - Clock configuration – PL Fabric clocks:
    - FCLK\_CLK0 체크
- 설정 후, PS7 IP 포트 변화



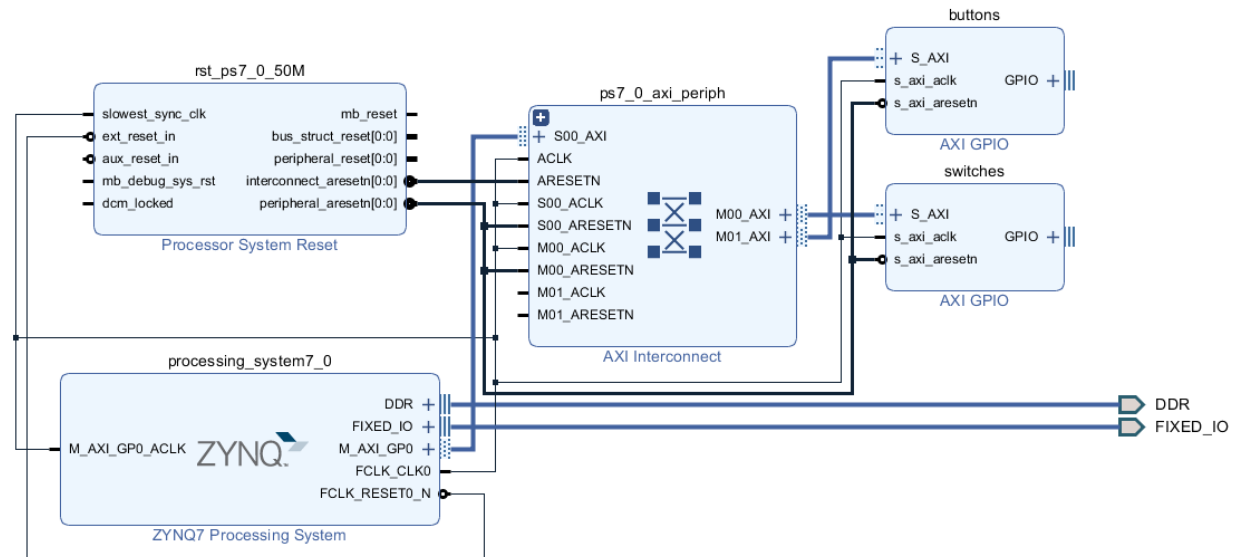
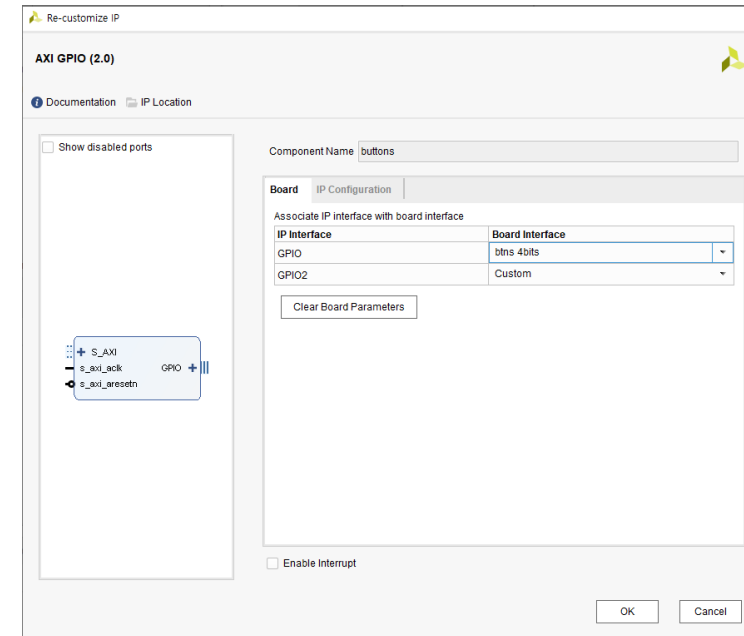
# Adding Two Instances of GPIO 2

- 블록 디자인 완성
  - AXI GPIO IP 추가:
    - Add IP: AXI GPIO
    - IP 이름: switches
  - AXI GPIO 커스터마이징
    - GPIO 포트: sws 4bits
  - Switches, PS 간 AXI connection 클릭
    - Run connection automation 클릭
    - S\_AXI만 선택



# Adding Two Instances of GPIO 3

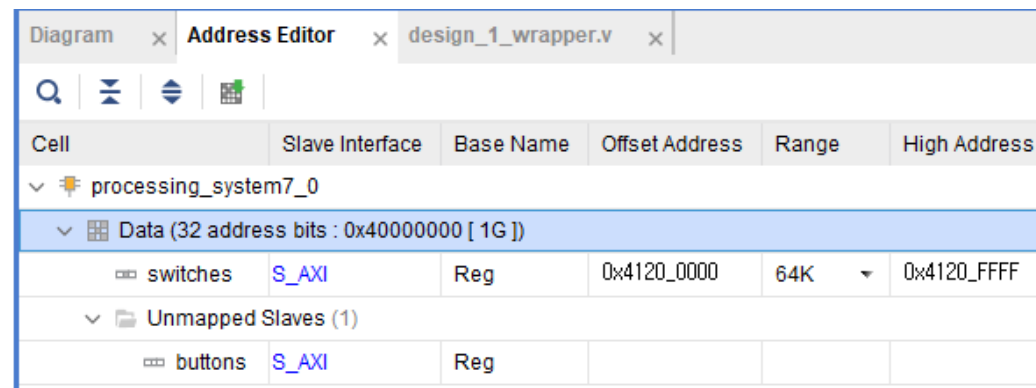
- 블록 디자인 완성
  - AXI GPIO IP 추가:
    - Add IP: AXI GPIO
    - IP 이름: buttons
  - AXI GPIO 커스터마이징
    - GPIO 포트: btns 4bits
  - buttons, PS 간 AXI connection
    - Run connection automation 클릭
    - S\_AXI만 선택
  - 혹은 수동으로 완성가능



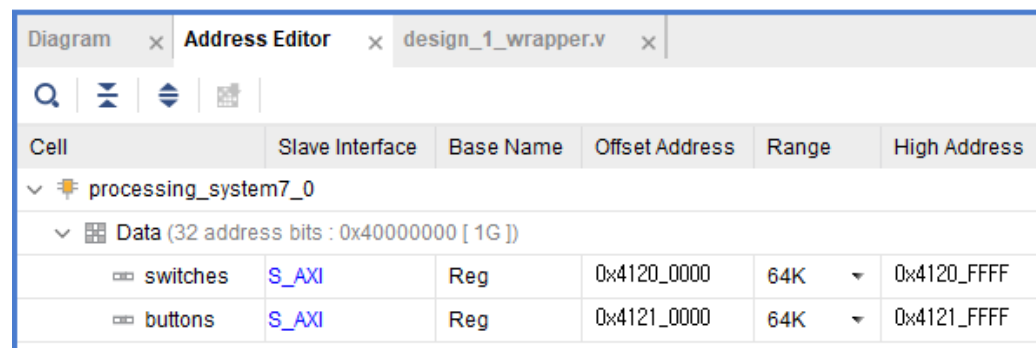


# Adding Two Instances of GPIO 4

- Address mapping
  - Switches: 자동으로 address 할당
  - Buttons: unmapped 상태로, address 할당 필요
    - ➔ 손으로 포트를 잇는 경우
- Buttons 오른쪽 클릭 – assign address
- 참고: GPIO0 주소 범위:
  - 0x40000000 ~ 0x7FFFFFFF



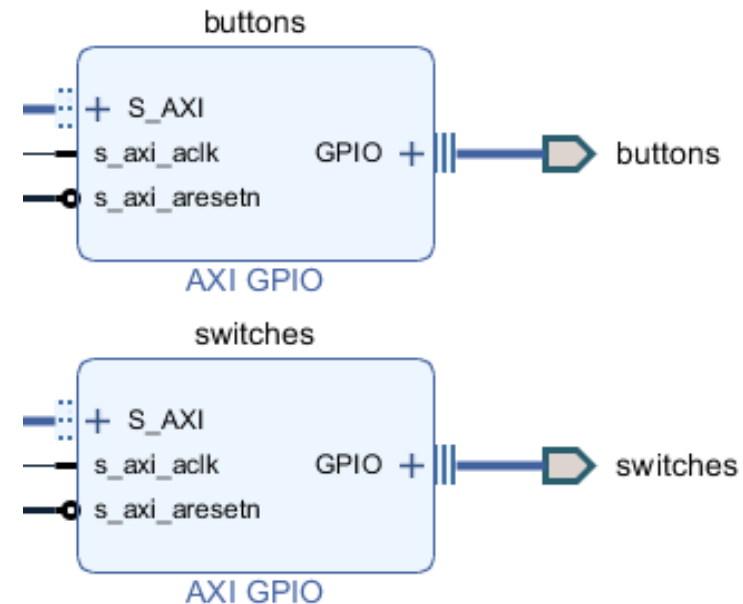
Cell	Slave Interface	Base Name	Offset Address	Range	High Address
processing_system7_0					
Data (32 address bits : 0x40000000 [ 1G ])					
switches	S_AXI	Reg	0x4120_0000	64K	0x4120_FFFF
Unmapped Slaves (1)					
buttons	S_AXI	Reg			



Cell	Slave Interface	Base Name	Offset Address	Range	High Address
processing_system7_0					
Data (32 address bits : 0x40000000 [ 1G ])					
switches	S_AXI	Reg	0x4120_0000	64K	0x4120_FFFF
buttons	S_AXI	Reg	0x4121_0000	64K	0x4121_FFFF

# Making GPIO Peripheral Connections External 1

- Buttons, Switches IP에 대응하는 핀들 연결
  - External 포트 만들기
    - 포트 오른쪽 클릭 – make external
    - 각 포트 이름: buttons, switches
- Design validation 검증
  - Validate design



# Making GPIO Peripheral Connections External 2

- Synthesize
  - 합성 진행 후, I/O planning tool을 통해 constraints 확인

- Run synthesis

- Open synthesized design
  - Layout - I/O planning 클릭

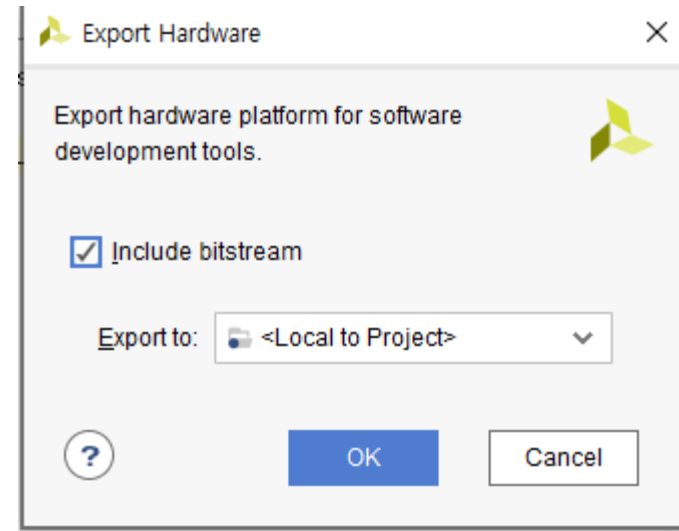
- Switches, buttons 포트들이 pin들에 자동으로 할당

➔ GPIO 인터페이스를 지정했기 때문에 pin 자동 할당

Tcl Console Messages Log Reports Design Runs Package Pins I/O Ports									
Name	Direction	Board Part Pin	Board Part Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	
All ports (138)									
> DDR_54576 (71)	INOUT					✓	502	(Multiple)*	
> FIXED_IO_54576 (59)	INOUT					✓	(Multiple)	(Multiple)*	
> GPIO_16763 (4)	IN					✓	(Multiple)	LVCMOS33*	
> buttons_tri_i (4)	IN					✓	(Multiple)	LVCMOS33*	
> buttons_tri_i[3]	IN	btns_4bits_tri_i_3			Y16	✓	34	LVCMOS33*	
> buttons_tri_i[2]	IN	btns_4bits_tri_i_2			K19	✓	35	LVCMOS33*	
> buttons_tri_i[1]	IN	btns_4bits_tri_i_1			P16	✓	34	LVCMOS33*	
> buttons_tri_i[0]	IN	btns_4bits_tri_i_0			K18	✓	35	LVCMOS33*	
Scalar ports (0)									
> GPIO_31309 (4)	IN					✓	(Multiple)	LVCMOS33*	
> switches_tri_i (4)	IN					✓	(Multiple)	LVCMOS33*	
> switches_tri_i[3]	IN	sws_4bits_tri_i_3			T16	✓	34	LVCMOS33*	
> switches_tri_i[2]	IN	sws_4bits_tri_i_2			W13	✓	34	LVCMOS33*	
> switches_tri_i[1]	IN	sws_4bits_tri_i_1			P15	✓	34	LVCMOS33*	
> switches_tri_i[0]	IN	sws_4bits_tri_i_0			G15	✓	35	LVCMOS33*	
Scalar ports (0)									

# Generating Bitstream and export to SDK

- 비트스트림 생성
  - Generate bitstream 클릭
- Sdk 실행
  - File – export hardware
    - Include bitstream 체크
  - File – launch sdk



**SDK**  
Software Development Kit

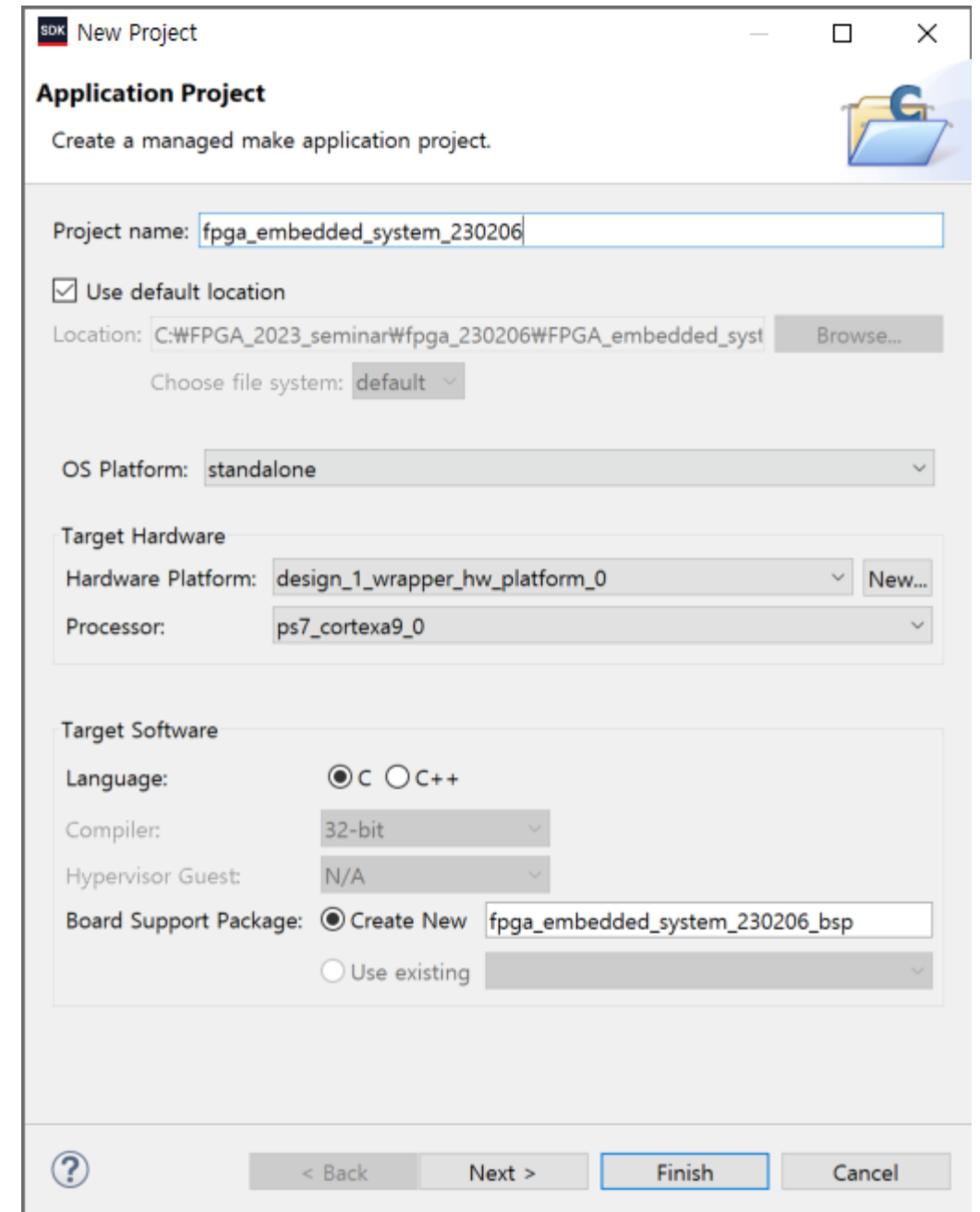
2018.2



Copyright 1986-2018 Xilinx, Inc.  
All Rights Reserved.

# Generating Application project in SDK

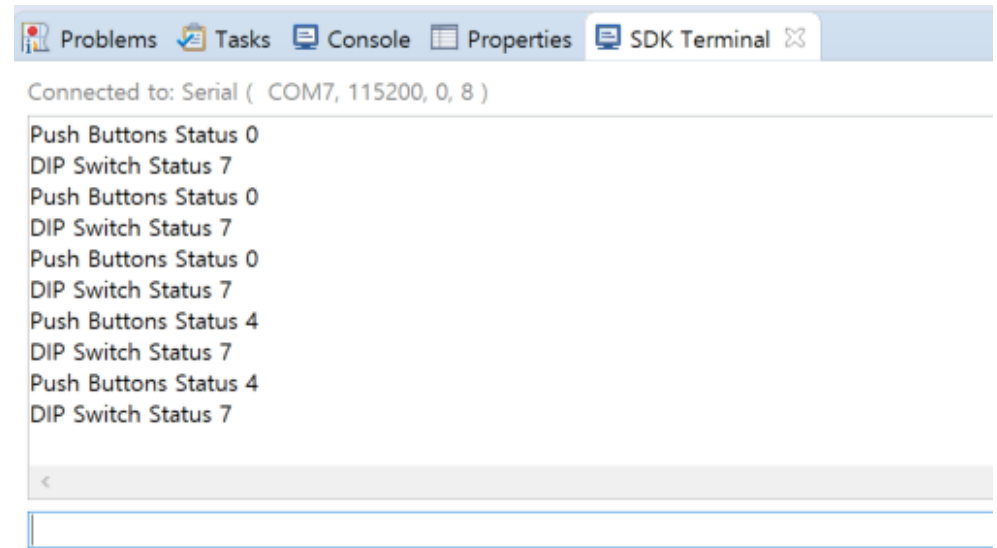
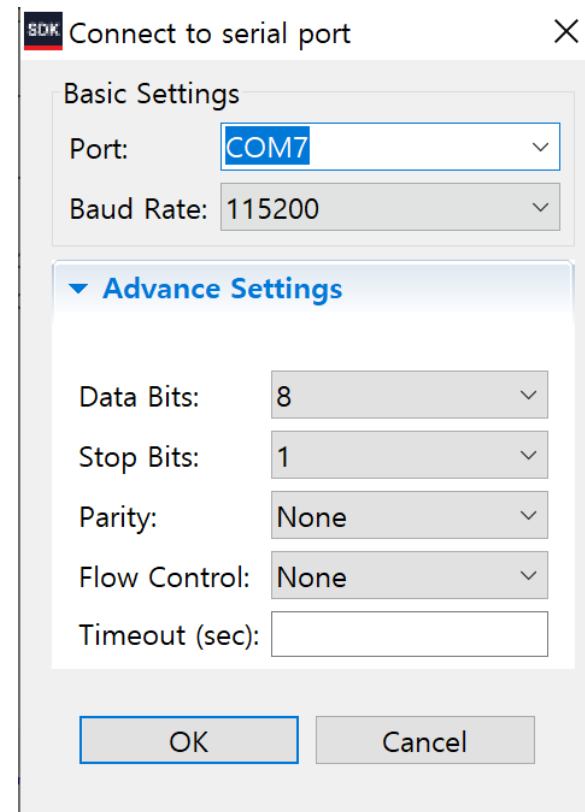
- 프로젝트 생성
  - File – new – application project
  - Project name: **fpga\_embedded\_system\_230210**
  - 템플릿: **empty application**
  - Finish 클릭
- 소스파일 추가:
  - Main.c



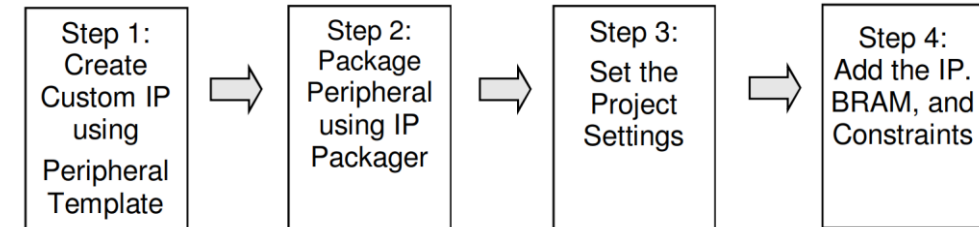
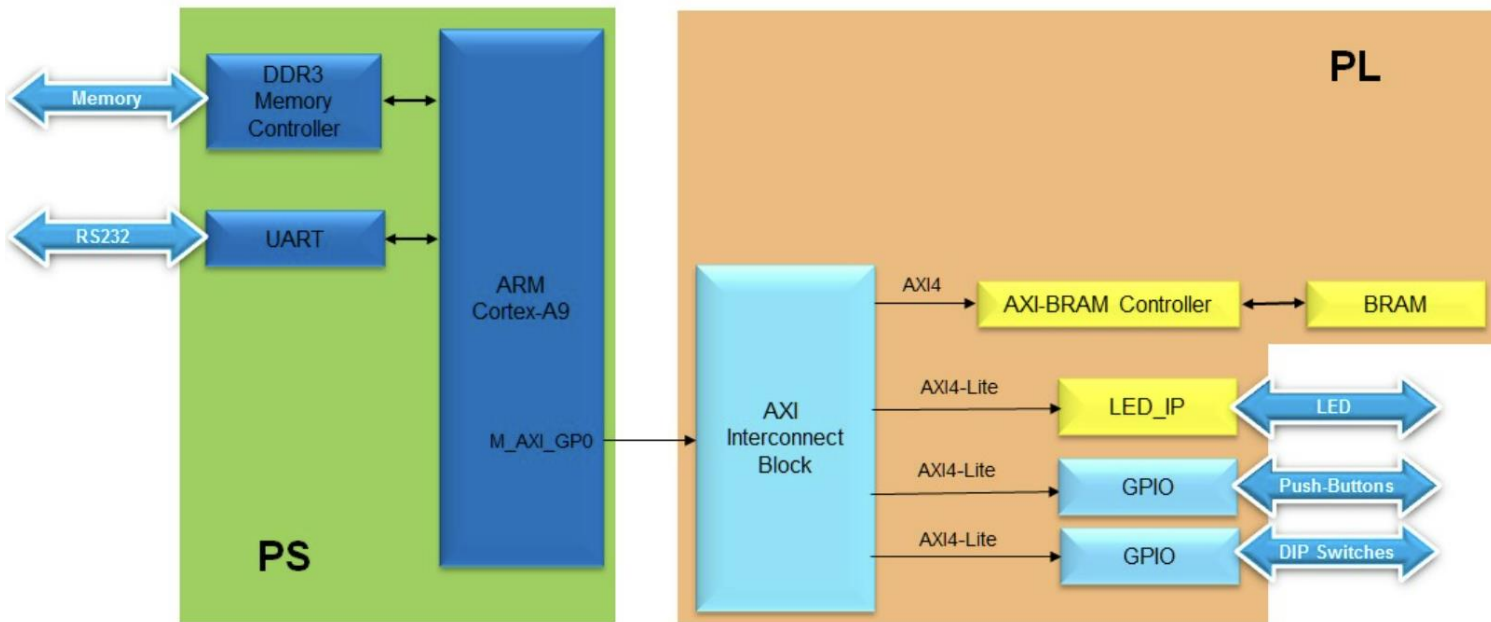
The screenshot shows the 'New Project' dialog box in the SDK. The title bar says 'New Project'. The main heading is 'Application Project' with a subtext 'Create a managed make application project.' and a folder icon. The 'Project name' field is filled with 'fpga\_embedded\_system\_230206'. The 'Use default location' checkbox is checked. The 'Location' field shows 'C:\WFPGA\_2023\_seminar\wfpga\_230206\WFPGA\_embedded\_syst' with a 'Browse...' button. The 'Choose file system' dropdown is set to 'default'. The 'OS Platform' dropdown is set to 'standalone'. Under 'Target Hardware', the 'Hardware Platform' dropdown is set to 'design\_1\_wrapper\_hw\_platform\_0' with a 'New...' button, and the 'Processor' dropdown is set to 'ps7\_cortexa9\_0'. Under 'Target Software', the 'Language' has radio buttons for 'C' (selected) and 'C++'. The 'Compiler' dropdown is set to '32-bit'. The 'Hypervisor Guest' dropdown is set to 'N/A'. The 'Board Support Package' has radio buttons for 'Create New' (selected) and 'Use existing'. The 'Create New' option has a text field containing 'fpga\_embedded\_system\_230206\_bsp'. At the bottom, there are buttons for '?', '< Back', 'Next >', 'Finish' (highlighted with a blue border), and 'Cancel'.

# Testing in Hardware

- FPGA에 코드 프로그래밍
  - **FPGA Program** 클릭: FPGA에 bitstream 업로드
  - 생성한 fpga\_embedded\_system\_230210 프로젝트 우 클릭
  - **Build project**
- **Sdk 터미널 연결**
  - Port: 장치관리자에서 FPGA와 연결된 포트 확인
- 생성한 fpga\_embedded\_system\_230210 프로젝트 우 클릭
- Run as – **launch on hardware**
- 동작 확인
  - 터미널 콘솔창에서 결과 확인 가능



# Adding custom IP to the system



- Custom IP 및 bram controller를 이용한 하드웨어 설계 flow
- 학습목표:
  - IP packager를 이용하여 custom IP 생성 및 기능 검증
  - Custom IP 활용
  - Pin 할당 및 block memory 추가

# 감사합니다!

- Q&A