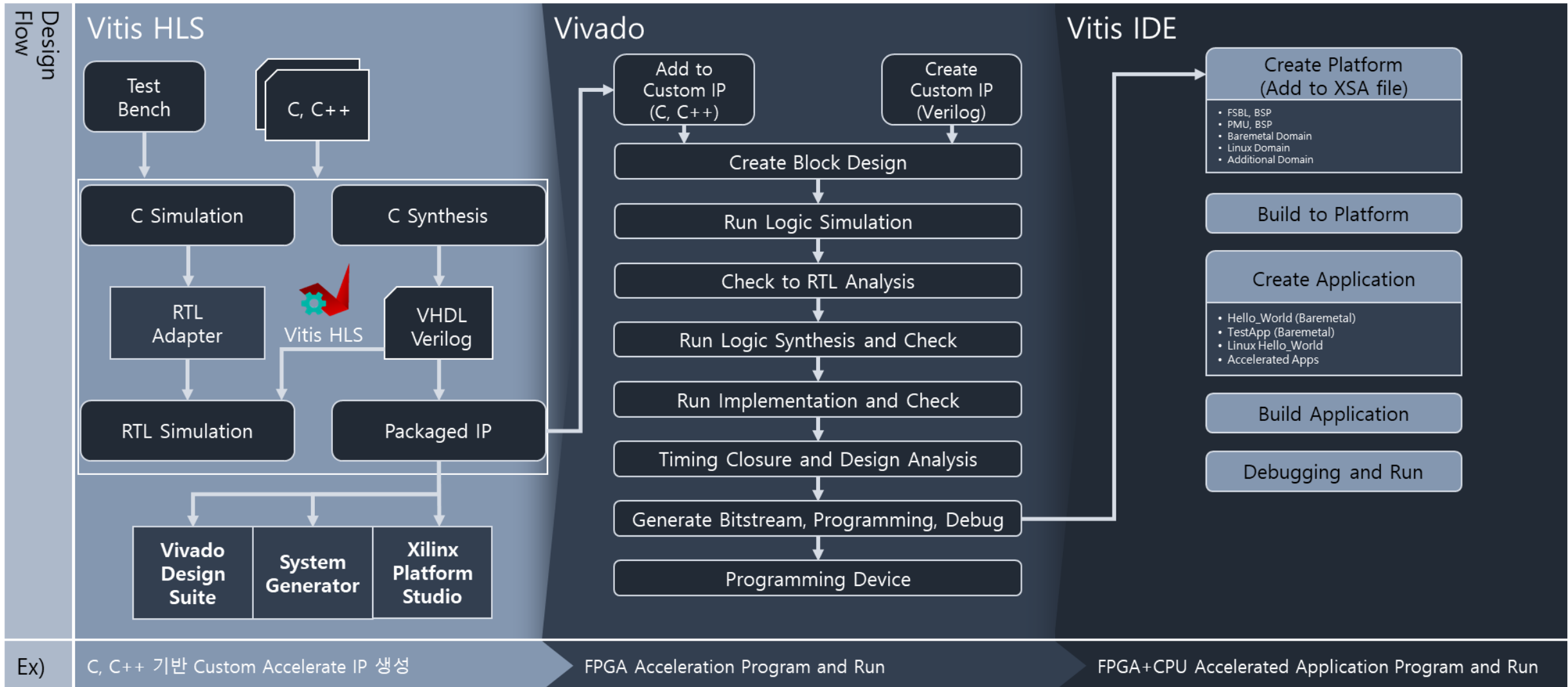


# Vitis Unified Software Development Environments

Vitis HLS / Vivado / Vitis IDE Tutorial

# Vitis Unified Platform - Design Flow



# Vitis HLS

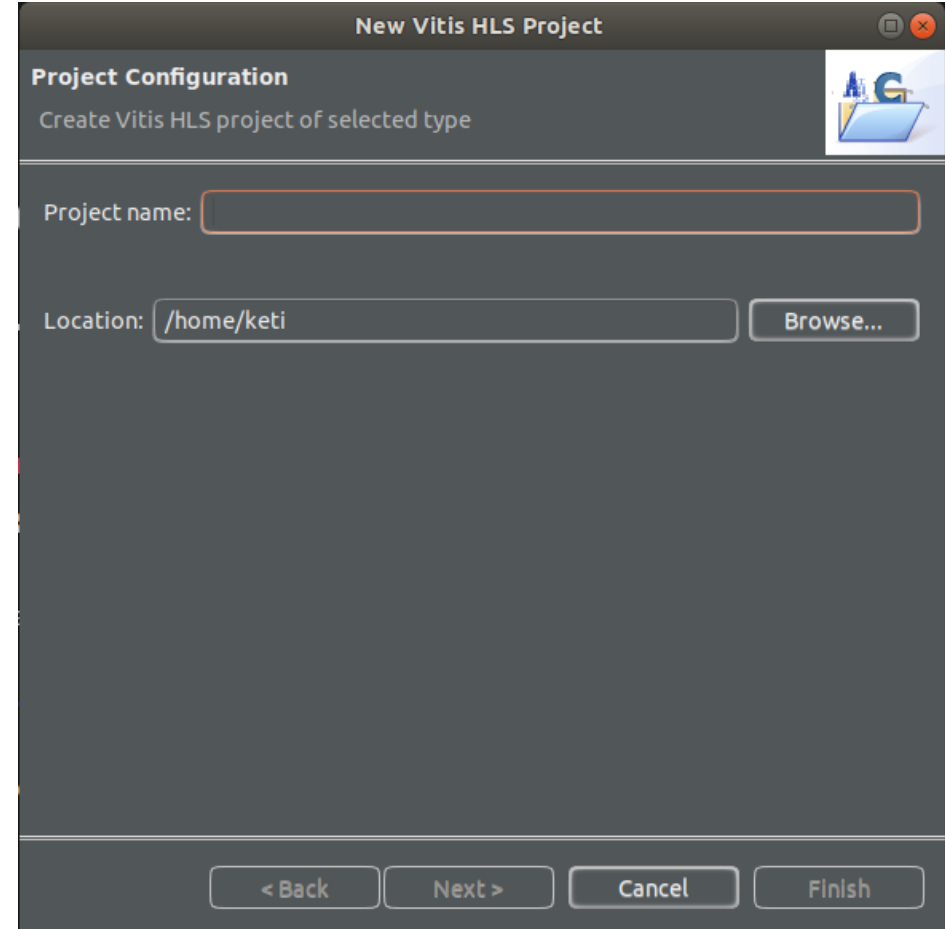
- Vitis HLS(High-Level Synthesis) project
  - Create
  - Synthesis
  - Export RTL
  - [Vitis HLS \(xilinx.com\)](https://www.xilinx.com/products/development-tools/vitis.html)
- Vitis HLS Libraries
  - 라이브러리 설명서
  - [https://xilinx.github.io/Vitis\\_Libraries/](https://xilinx.github.io/Vitis_Libraries/)

# Vitis HLS - Create

- `sudo /tools/Xilinx/Vitis_HLS/2020.2/bin/vitis_hls`
- 위의 명령어로 Vitis HLS를 실행함

# Vitis HLS - Create

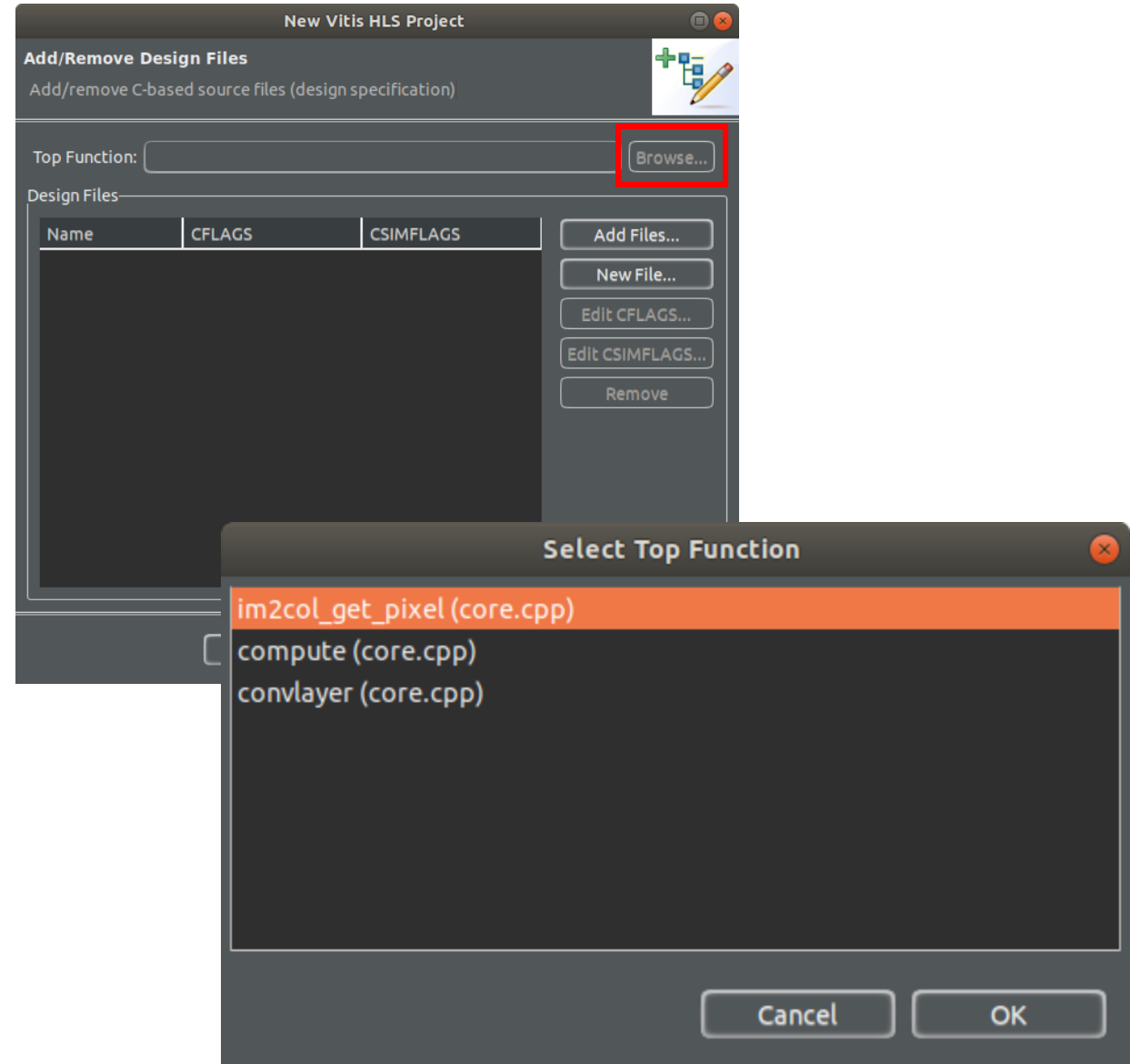
- Create new project
- Project name 및 location 설정
  - 한글이 포함된 폴더 있으면 시뮬레이션 오류 발생가능
- Next



The screenshot shows a window titled "New Vitis HLS Project" with a close button in the top right corner. The main area is labeled "Project Configuration" and contains the instruction "Create Vitis HLS project of selected type". There is a small icon of a folder with a blue 'G' on the right. Below the instruction, there are two input fields: "Project name:" followed by an empty text box, and "Location:" followed by a text box containing "/home/keti" and a "Browse..." button. At the bottom of the window, there are four buttons: "< Back", "Next >", "Cancel", and "Finish".

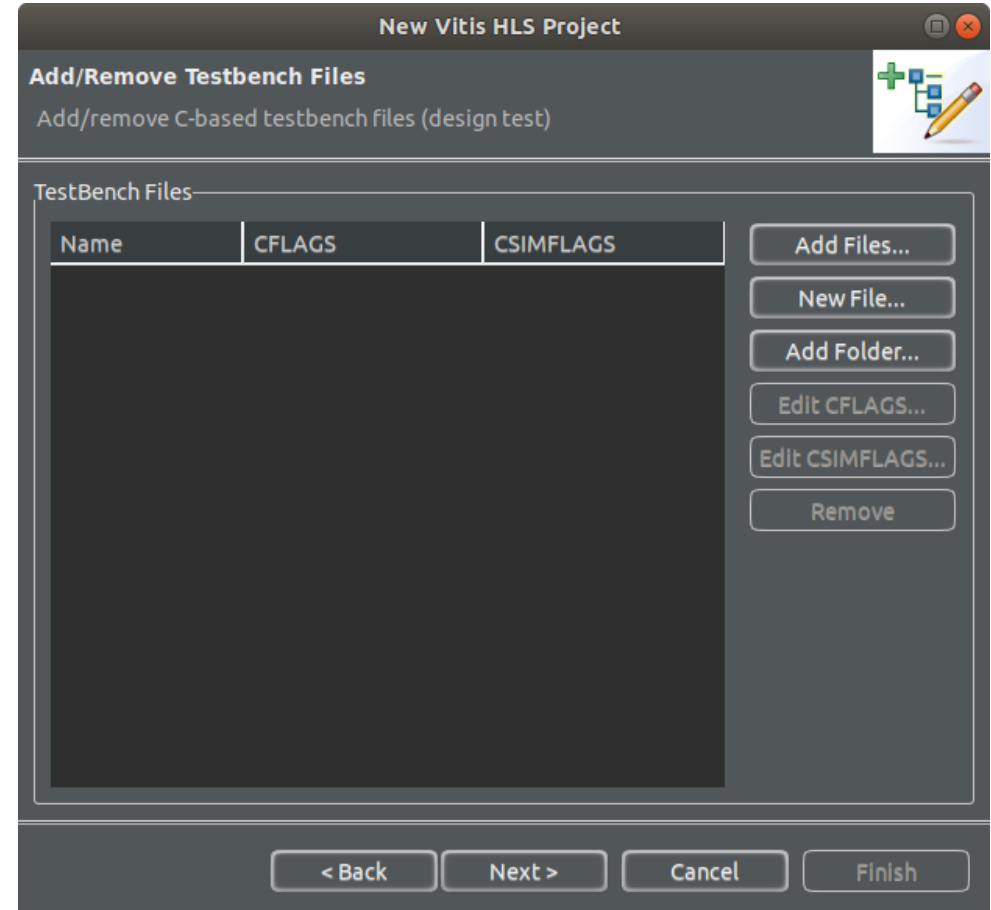
# Vitis HLS - Create

- 미리 만들어 놓은 source file 이 있으면 "Add files.."로 가져옴
  - 만들어 놓은 소스파일이 존재하면 추가 (.c / .cpp)
  - Top Function > Browse 클릭 > 최상위 함수 선택
- next



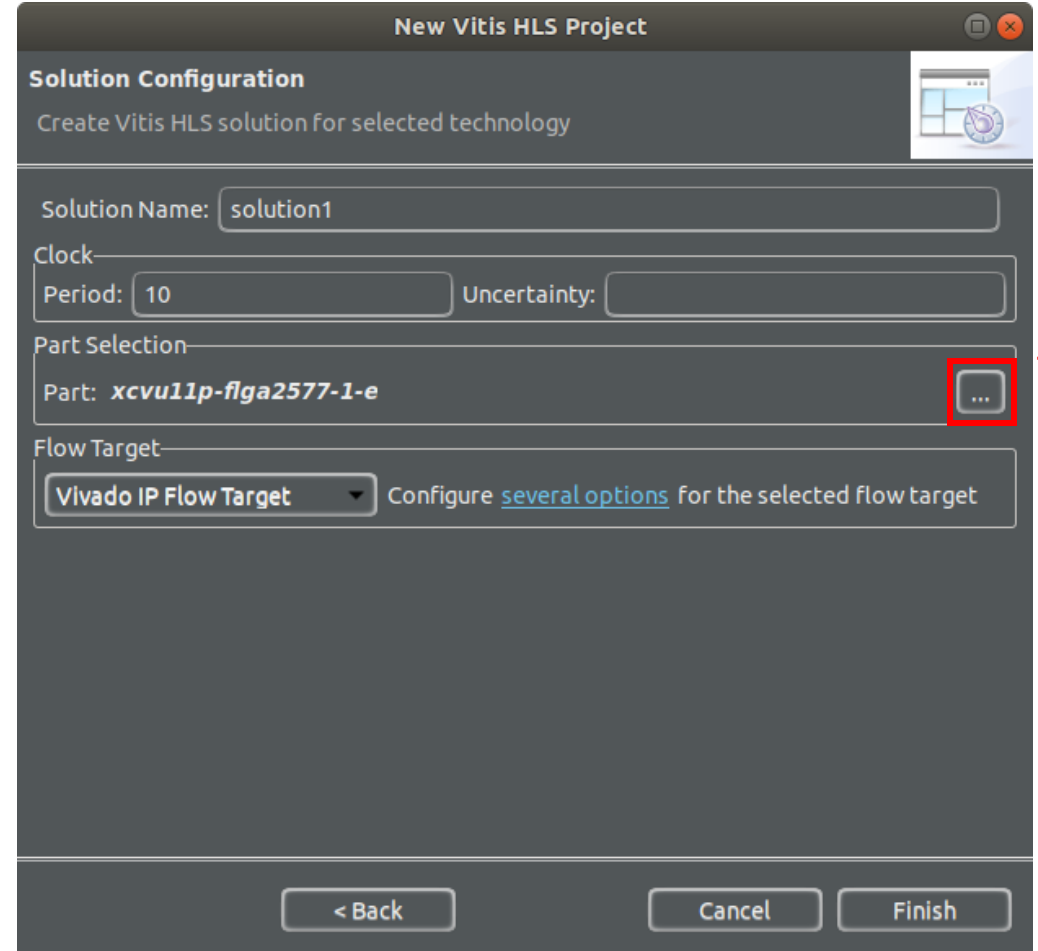
# Vitis HLS - Create

- 미리 만들어 놓은 test bench file이 있으면 "Add files.."로 가져옴
- 만들어 놓은 Testbench 파일이 존재하면 추가 (.c / .cpp)
- next



# Vitis HLS - Create

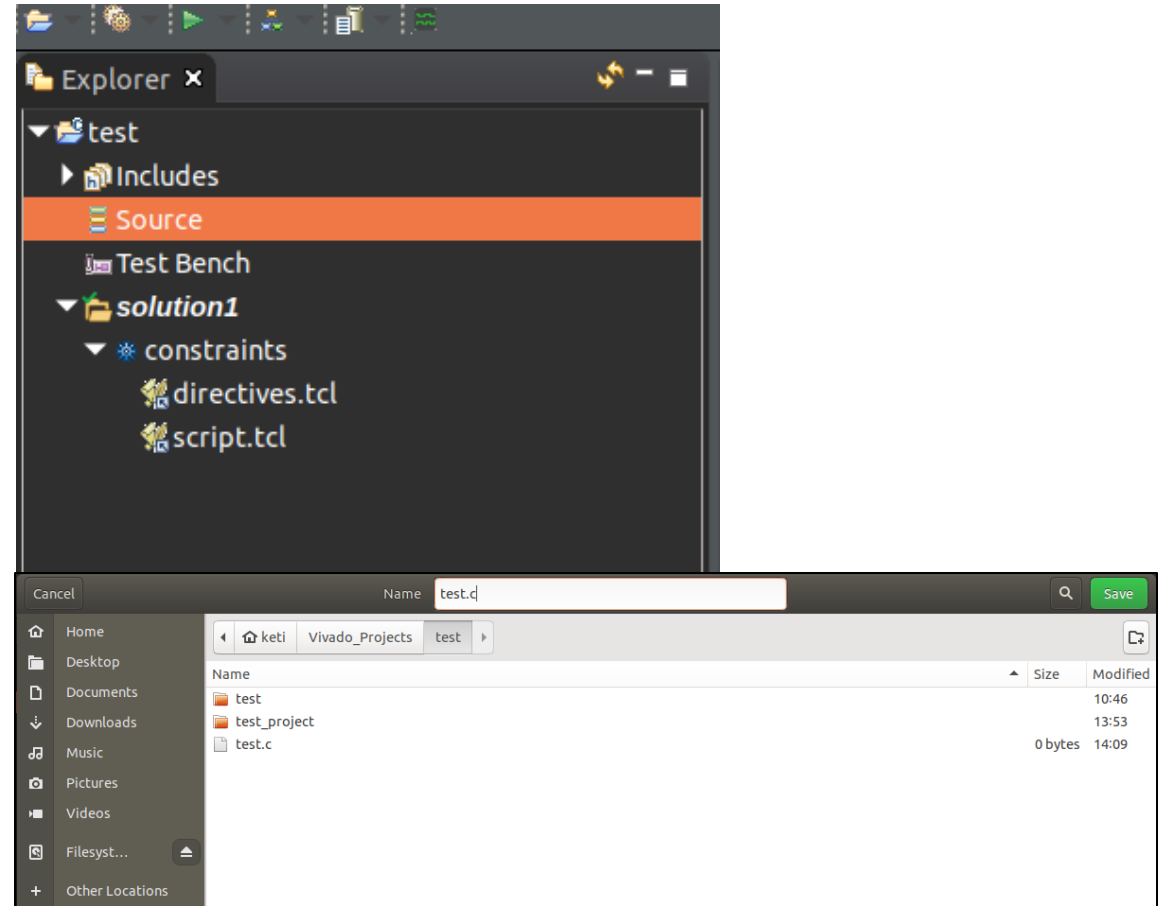
- 사용할 보드의 parts number 찾기
  - Part Selection 클릭 > Board 클릭 > Ultra96의 parts number는 xczu3eg-sbva484-1-e
  - 각각의 고유 parts number에 맞게끔 선택한 후 없을 시 보드정보가 들어있는 파일을 다운로드 받아 아래의 경로에 추가
  - Ubuntu : tools/Xilinx/Vitis\_HLS and Vivado and Vitis/2020.2/data/boards/board\_files
  - 보드 파일 다운로드 : `sudo git clone https://github.com/Avnet/bdf.git`
- Ok
- Finish






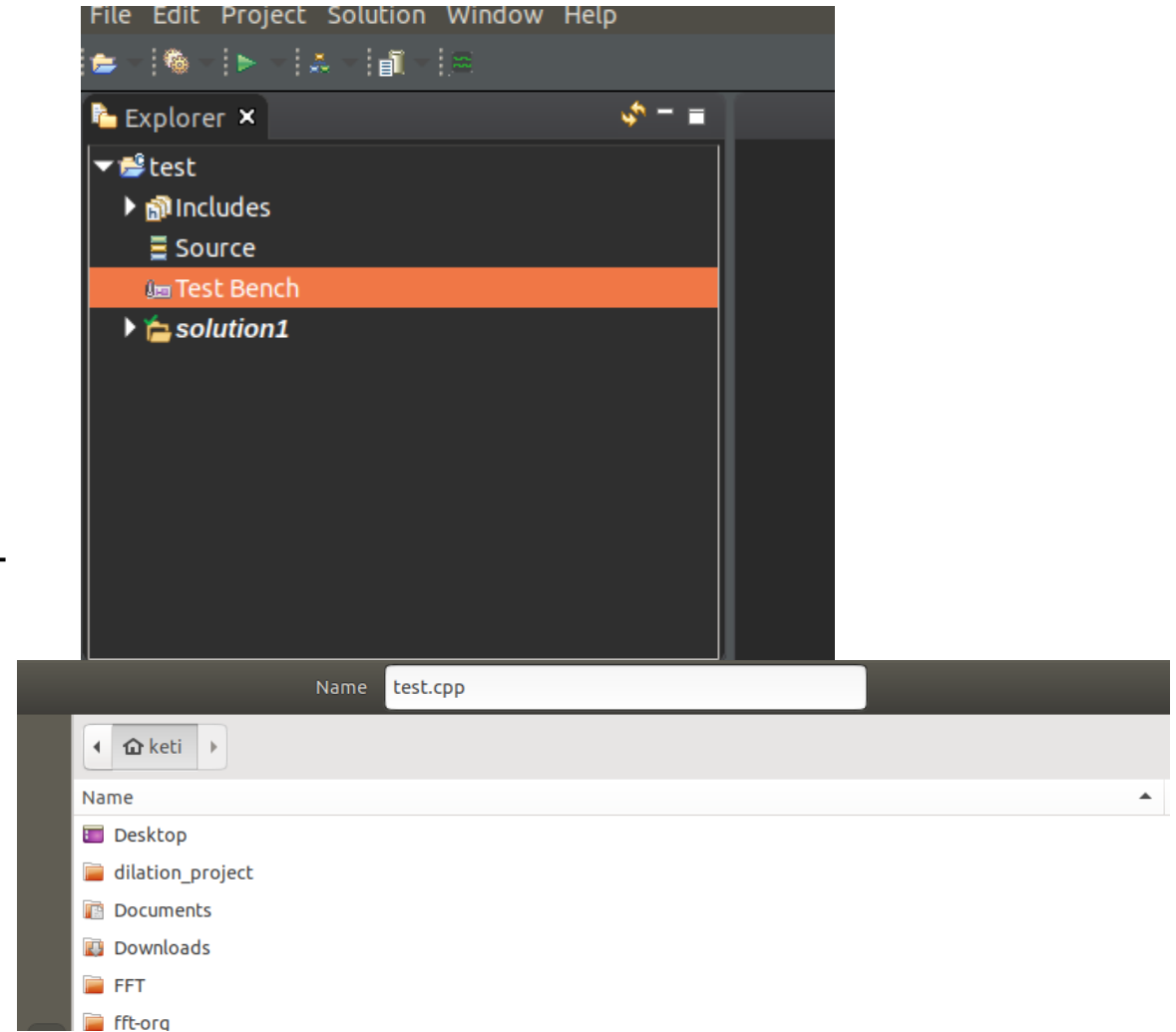
# Vitis HLS - Create

- 위 과정 중 source file이 없을 시 그림과 같이 진행
  1. Explorer window 아래의 Source 오른쪽 클릭
  2. new file 클릭
  3. ~.c /.cpp 이름으로 파일 생성 후 소스 코드 작성



# Vitis HLS - Create

- test bench(.c / .cpp) 설계 or 없을 시 Explorer window 아래의 Test Bench 오른쪽 클릭
- new file 클릭
- tb\_~ .c / .cpp이름으로 생성 후 코드 작성 진행
- Run C simulation 
- 설계한 대로 잘 동작하는지 확인



# Vitis HLS - Synthesis

- Run C Synthesis
- 합성 결과 확인

The screenshot displays the Vitis Synthesis Summary Report for a project named 'FFT'. The report is organized into several sections: General Information, Timing Estimate, Performance & Resource Estimates, HW Interfaces, and SW I/O Information.

**General Information**

Date: Thu Aug 19 13:11:46 2021  
Version: 2020.2 (Build 3064766 on Wed Nov 18 09:12:47 MST 2020)  
Project: fft\_org

**Timing Estimate**

Target	Estimated	Uncertainty
10.00 ns	2.637 ns	2.70 ns

**Performance & Resource Estimates**

Modules & Loops

Module/Loop	Issue Type	Slack	Latency	Iteration	Latency
FFT					
FFT0_1					
VITIS_LOOP_58_1					
bitreversal_label1					
FFT_label1					
VITIS_LOOP_68_2					

**HW Interfaces**

AXIS

Interface	Register Mode	TDATA	TREADY	TVALID
data_IN	both	32	1	1
data_OUT	both	32	1	1

**TOP LEVEL CONTROL**

Interface	Type	Ports
ap_clk	clock	ap_clk
ap_rst_n	reset	ap_rst_n
ap_ctrl	ap_ctrl_hs	ap_done ap_idle ap_ready ap_start


**SW I/O Information**

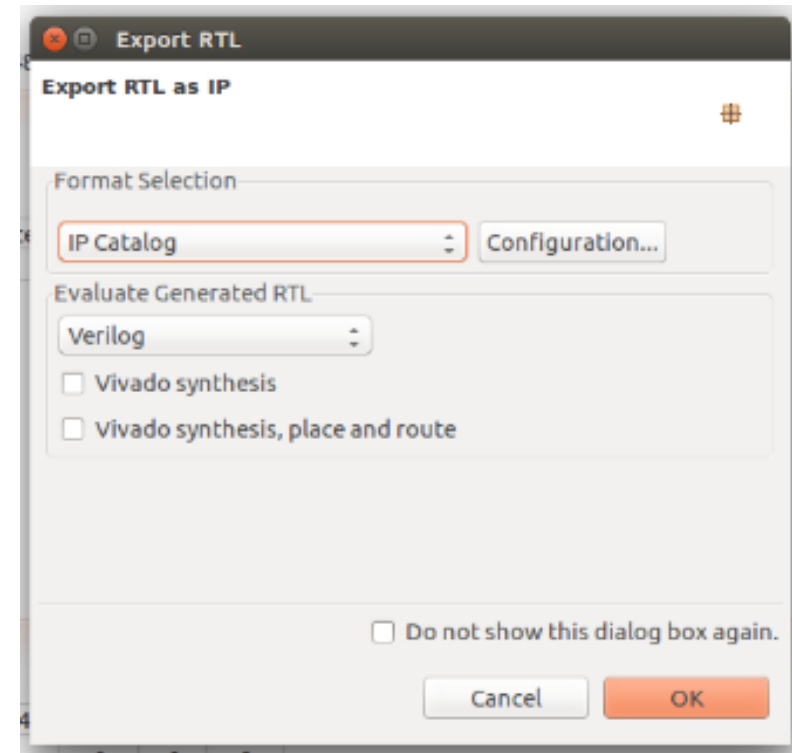
Top Function Arguments

Argument	Direction	Datatype
data_IN	in	complex<ap_fixed<16 8 AP_TRN AP_WRAP 0>>*
data_OUT	out	complex<ap_fixed<16 8 AP_TRN AP_WRAP 0>>*

**SW-to-HW Mapping**

# Vitis HLS - Export RTL

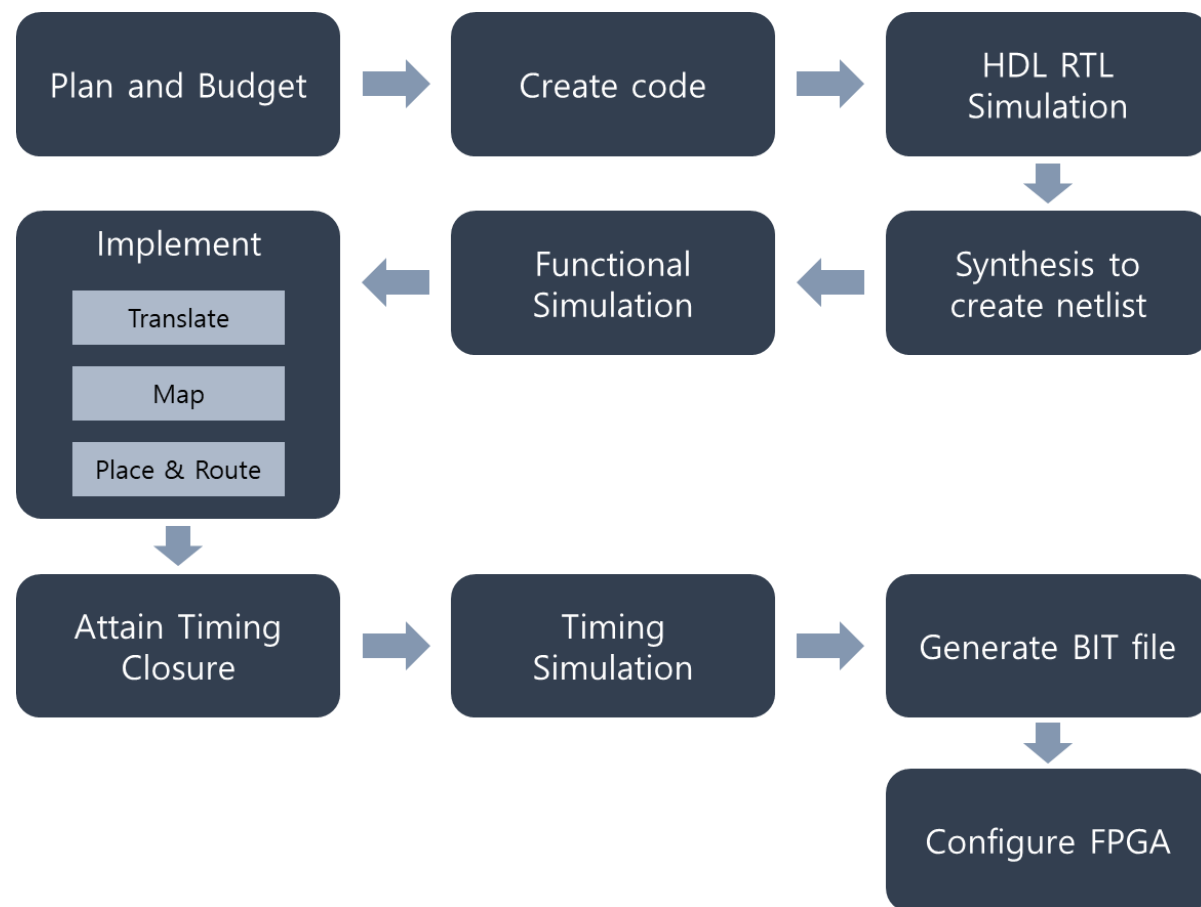
- 상단의 메뉴탭에서 Solution 클릭
-  (Export RTL) → OK



# Vivado Project

- Vivado Layout Flow

- Plan and Budget
  - 만들 제품에 대한 규격 및 예상 사용 자원 정리
- Create code
  - 소스코드 작성 (Verilog/VHDL or HLS 과정에서 C 언어로 작성)
- HDL RTL Simulation
  - RTL 시뮬레이션 진행 (단순히 코드가 잘 작동하는지 테스트)
- Synthesis to create netlist
  - HDL 코드 합성 후 Netlist 생성
- Functional Simulation
  - 기능적인 부분 시뮬레이션 진행 (실제 합성이 제대로 되었는지 체크)
- Implement
  - Target FPGA에 디지털 회로를 배치하고 배선진행 (Place & Route)
- Attain Timing Closure
- Timing Simulation
  - 원하는 클럭 스피드에서 구동되는지에 대한 검증작업
- Generate BIT File
- Configure FPGA
  - FPGA 보드에 실제로 구동



# Vivado Project

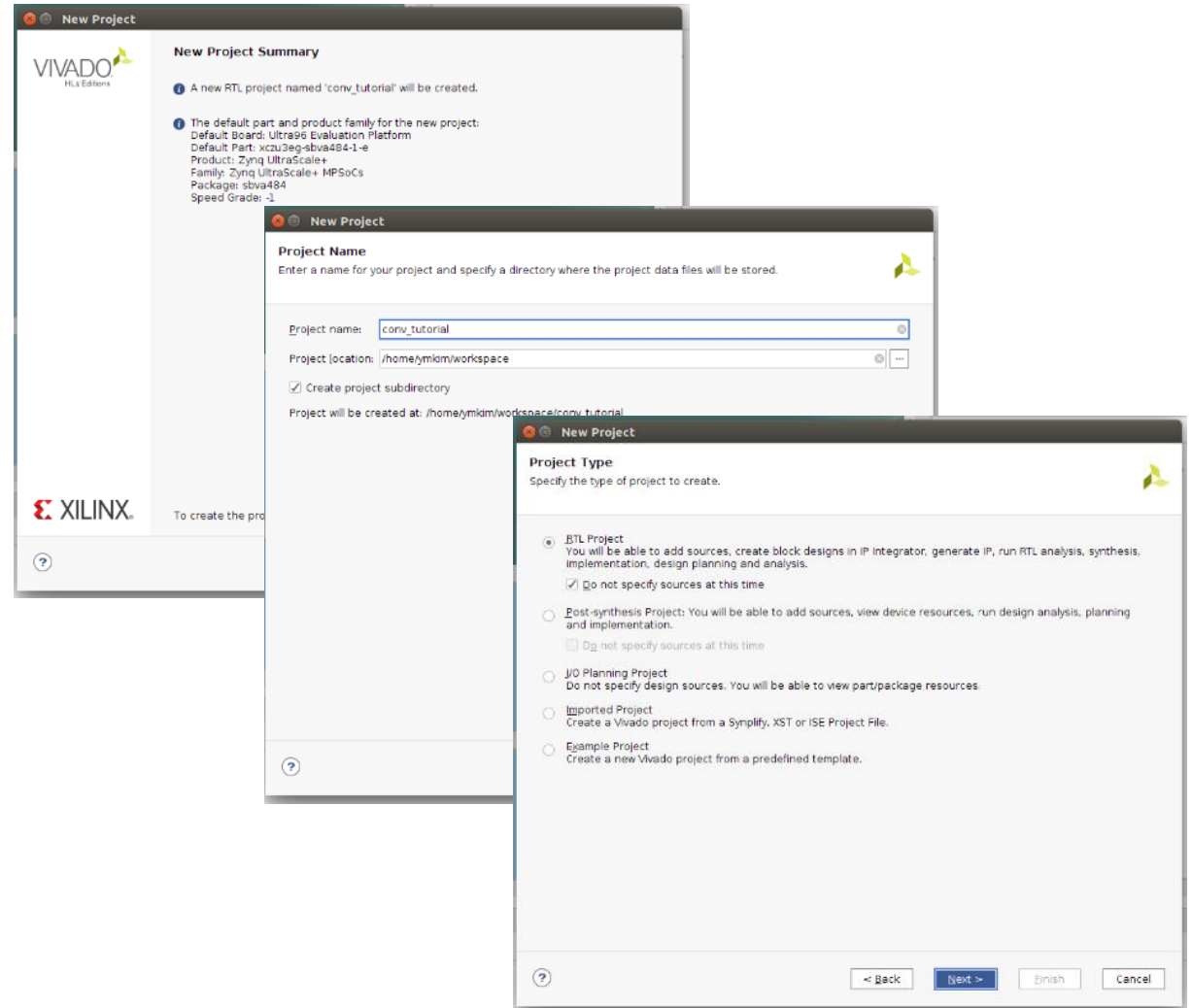
- Vivado Project
  - Create
  - Block Design
    - FPGA or FPGA+CPU 블록 디자인 진행
  - HDL Wrapper
    - 블록디자인 소스를 직접 합성 할 수 없기에 최상위 HDL Wrapper로 변환
  - Simulation
    - TestBench에서 작성한 데이터 입력 후 출력 값 확인 (정상작동 체크)
  - Synthesis
    - HDL Wrapper 합성 > Netlist 생성 (실제 디지털 회로로 구현)
  - Implement
    - Device 리소스에 넷리스트를 배치 및 라우팅 (블록디자인의 논리적, 물리적 및 타이밍 제약조건 충족 확인)
  - Generate Bitstream
  - Programming Device
    - FPGA 보드에 바로 구동 시킬 시 진행
  - Export BIT
    - FPGA+CPU에 구동 시킬 시 진행

# Vivado Project - Create

- `sudo /tools/Xilinx/Vivado/2020.2/bin/vivado`
- 위의 명령어로 vivado를 실행함

# Vivado Project - Create

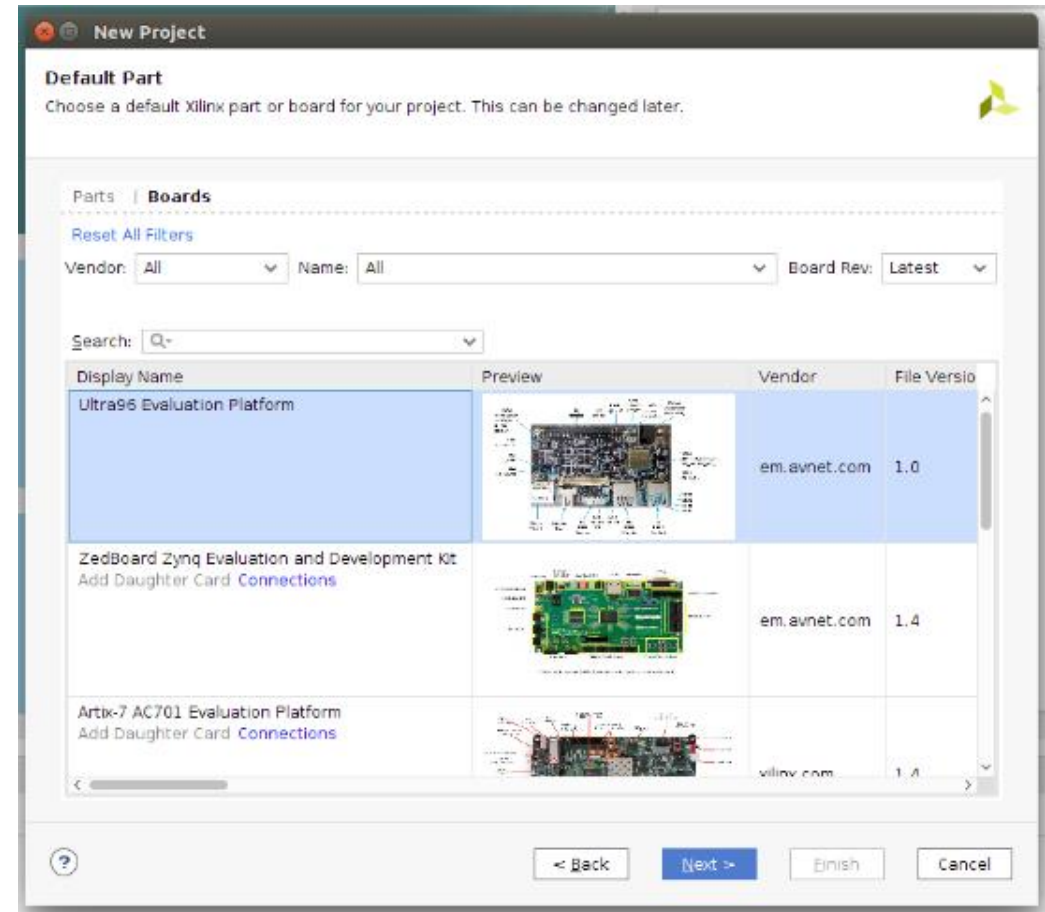
- Create Project → Next
- Project name, location 설정 → Next
- RTL Project → Next





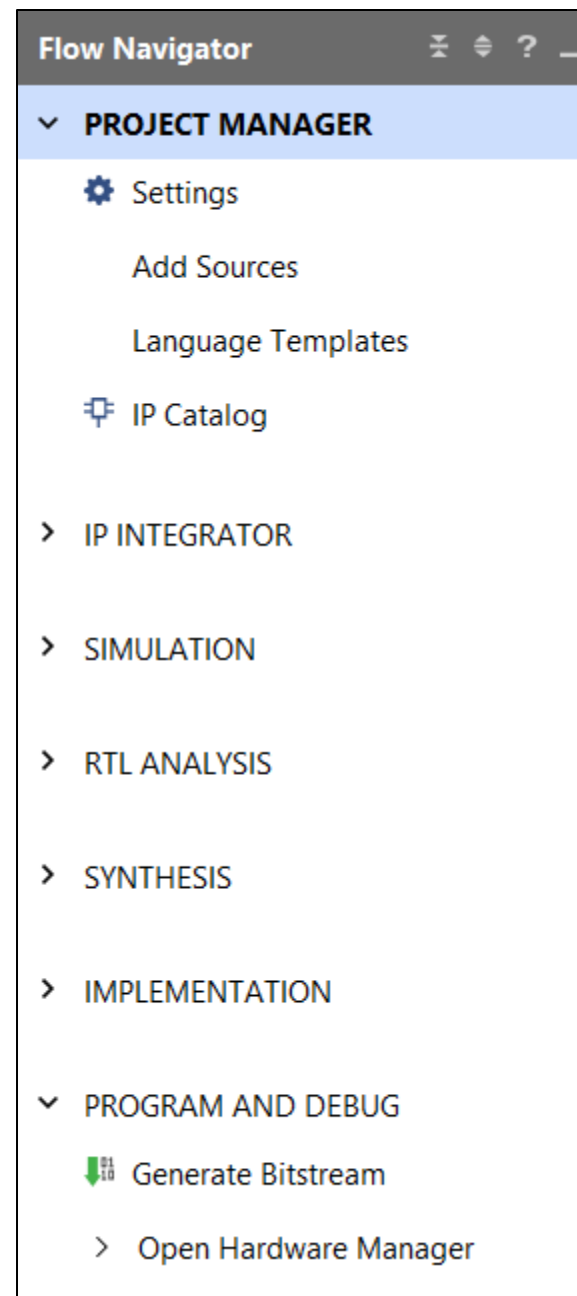
# Vivado Project - Create

- Ultra96v2 file
  - 없다면 ultra96 BSP받아서 "vivado 설치 위치/data/boards/board\_files" 에 넣음
- Next → Finish



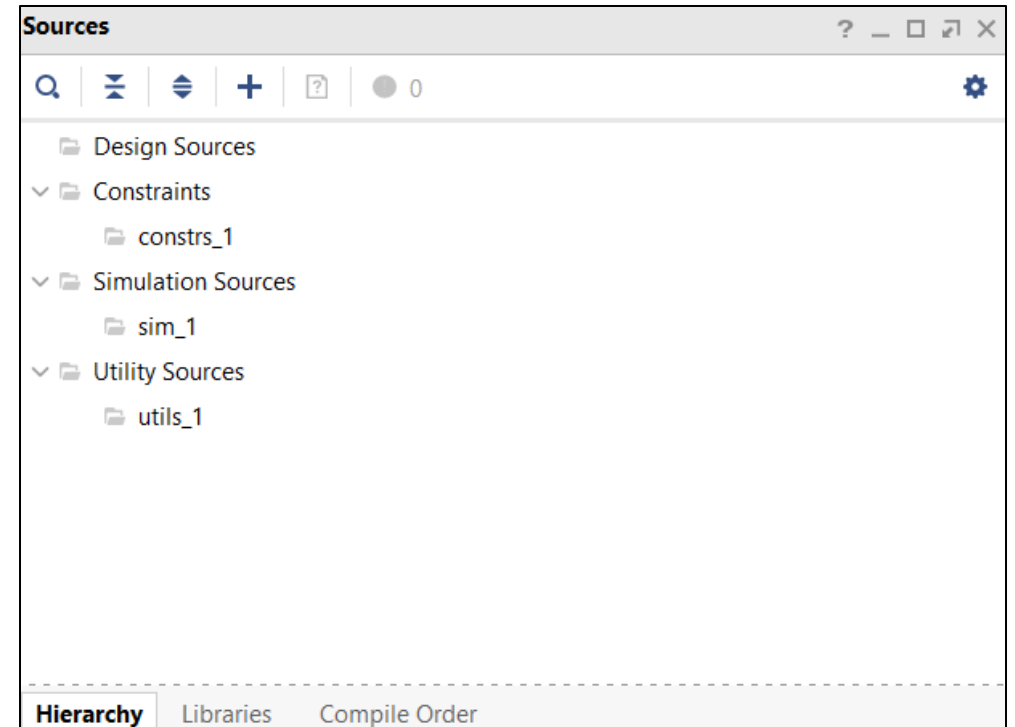
# Vivado Project - Create

- Flow Navigator
  - Project Manager
    - 소스, 언어 템플릿 및 IP 카탈로그를 추가하여 프로젝트 설정에 빠르게 액세스 가능
  - IP Integrator
    - Zynq 처리 시스템 또는 MicroBlaze 프로세서와 관련된 설계에 가장 많이 사용되는 블록 설계 작성 도구
    - 시뮬레이션 장치를 프로그래밍하기 전에 설계 출력을 확인 할 수 있음
  - RTL Analysis
    - RTL 분석 결과를 보여주며, 결과로는 DRC 및 RTL 회로도, 분석방법을 확인 가능
  - Synthesis
    - 합성 설정, 합성된 디자인 보기 및 합성 후 보고서에 액세스 가능
  - Implementation
    - 구현 설정, 구현된 디자인 보기 및 구현 후 보고서에 액세스 가능
  - Program and Debug
    - 비트 스트림을 생성하고 하드웨어 관리자를 사용하여 보드를 프로그래밍 진행



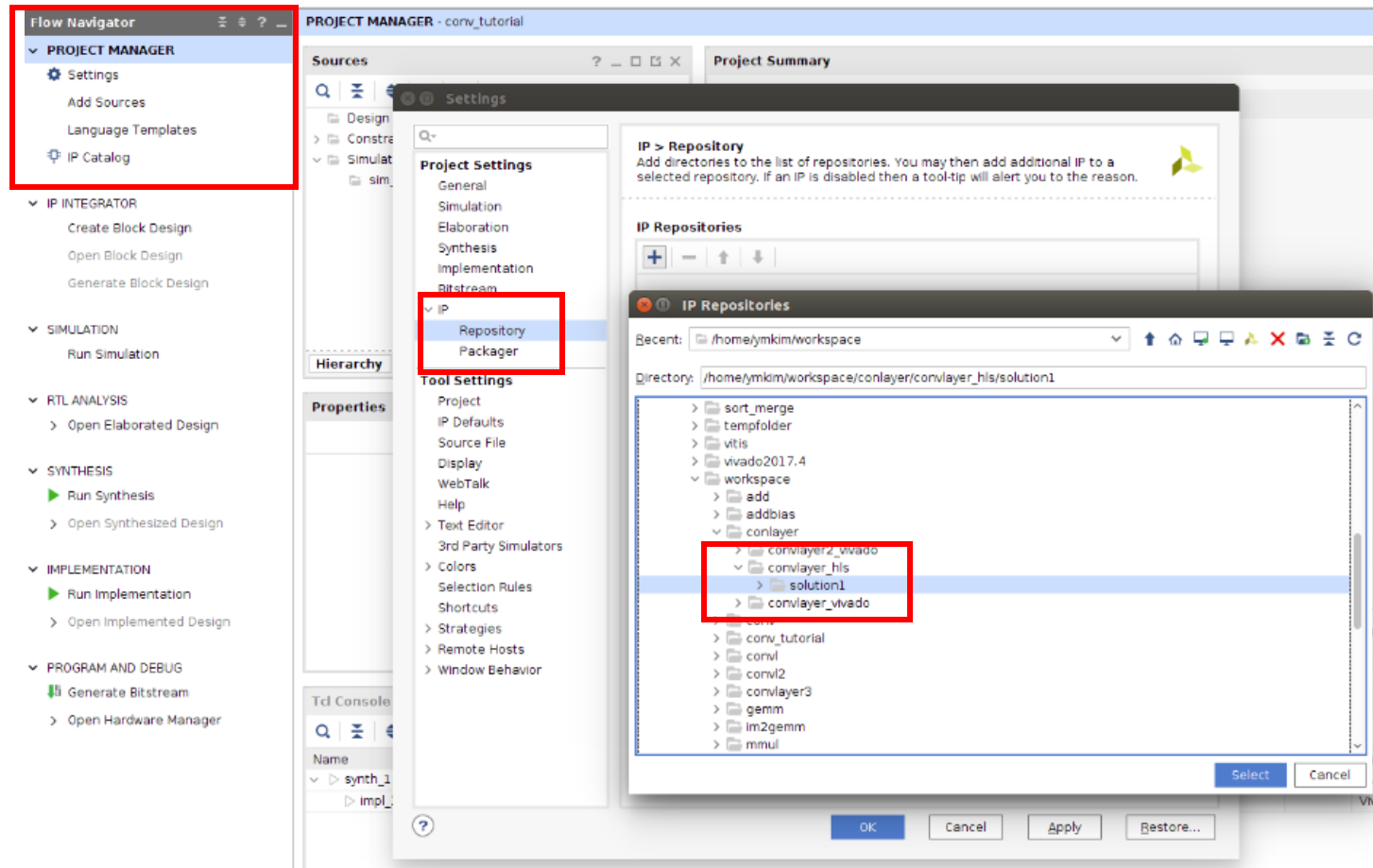
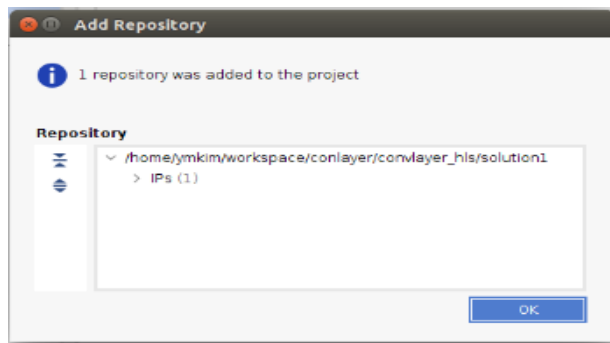
# Vivado Project - Create

- Sources
  - 모든 프로젝트 계층이 포함되며, 파일을 여는데 사용
  - Design Sources
    - HDL 파일이 Design Sources 폴더에 유지
  - Constraints
    - HDL 파일에 대한 구속조건
  - Simulation Sources
    - HDL 파일에 대한 시뮬레이션(HLS 시뮬레이션 결과)



# Vivado Project – Block Design

- Export한 RTL 파일을 불러와야 함
  - Flow Navigator의 Settings
  - IP → Repository (추출한 RTL 파일 추가)
  - HLS project 만든 폴더 내의 solution1
  - Select → OK

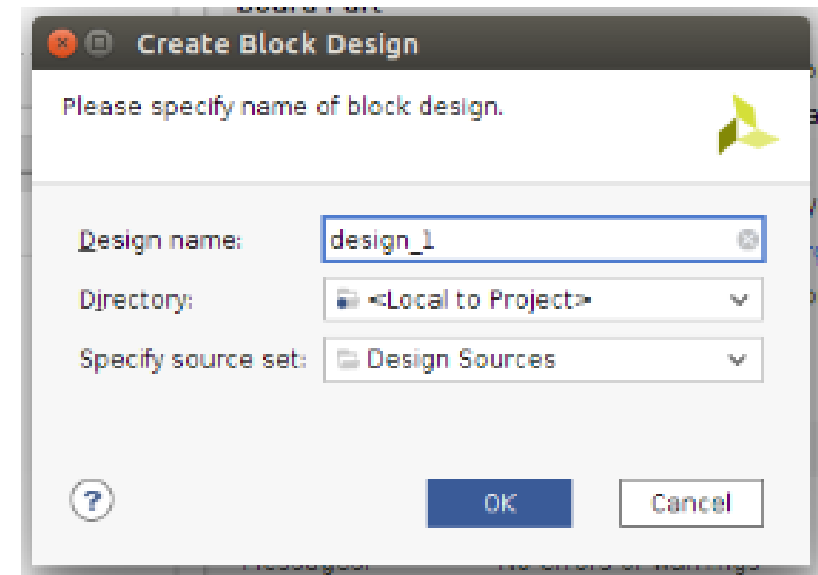


# Vivado Project – Block Design

- Export한 RTL 파일을 불러와야 함
  - Flow Navigator의 Settings
  - IP → Repository (추출한 RTL 파일 추가)
  - HLS project 만든 폴더 내의 solution1
  - Select → OK

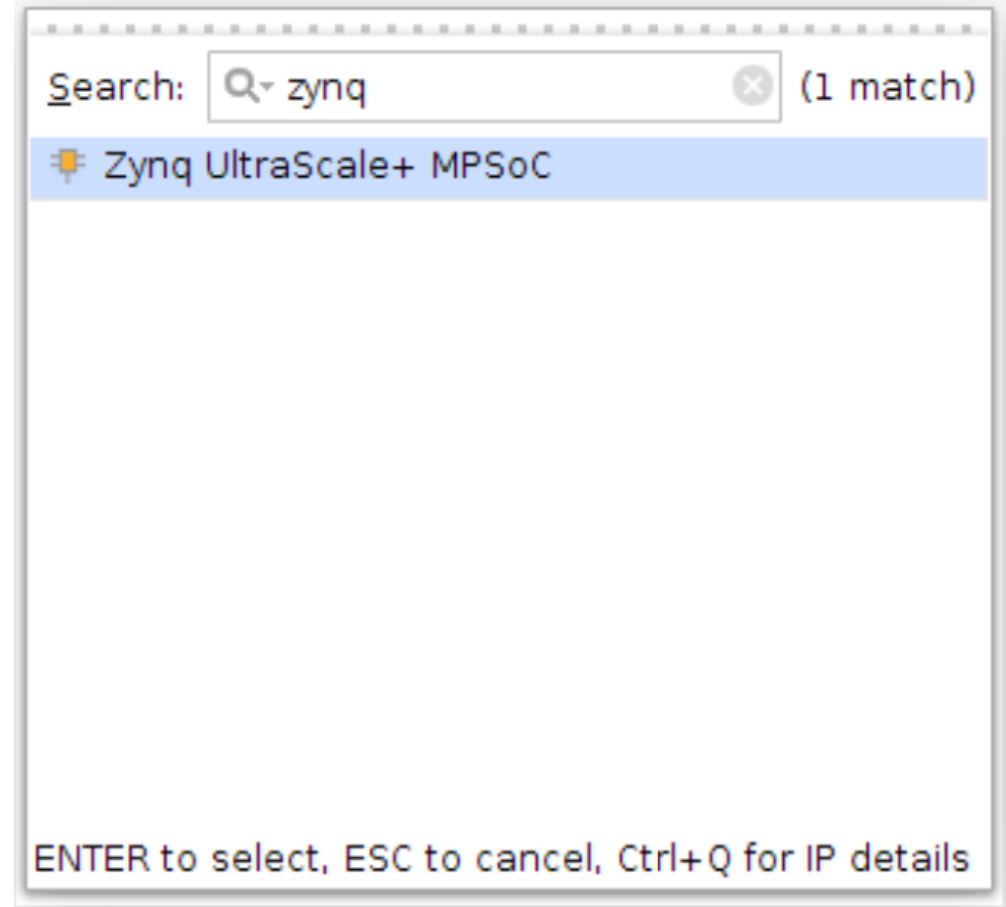
# Vivado Project – Block Design

- Flow Navigator
  - IP generator
    - Create Block Design → OK



# Vivado Project – Block Design

- + 버튼 눌러 block 추가
  - Zynq UltraScale MPSoC



# Vivado Project – Block Design

- Zynq UltraScale MPSoC 세부 기능 변경

- I/O Configuration
- Clock Configuration
- DDR Configuration
- PS-PL Configuration
- 위의 기능들의 설정값(레퍼런스 값) 및 기능 사용 여부만 변경이 가능
- 자세한 내용은 [Ug1085 Zynq Ultrascale TRM | PDF | Central Processing Unit | Booting \(scribd.com\)](#) 참고

I/O Configuration

Clock Configuration

DDR Configuration

PS-PL Configuration



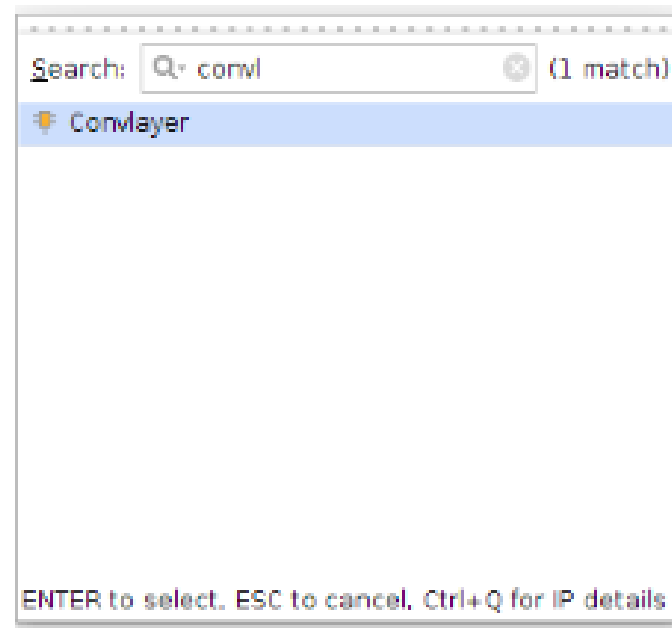
# Vivado Project – Block Design

- Run Block Automation 클릭
  - Ultra96 BSP에 존재하는 정보로  
세부 parameter를 조정해줌
  - OK

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

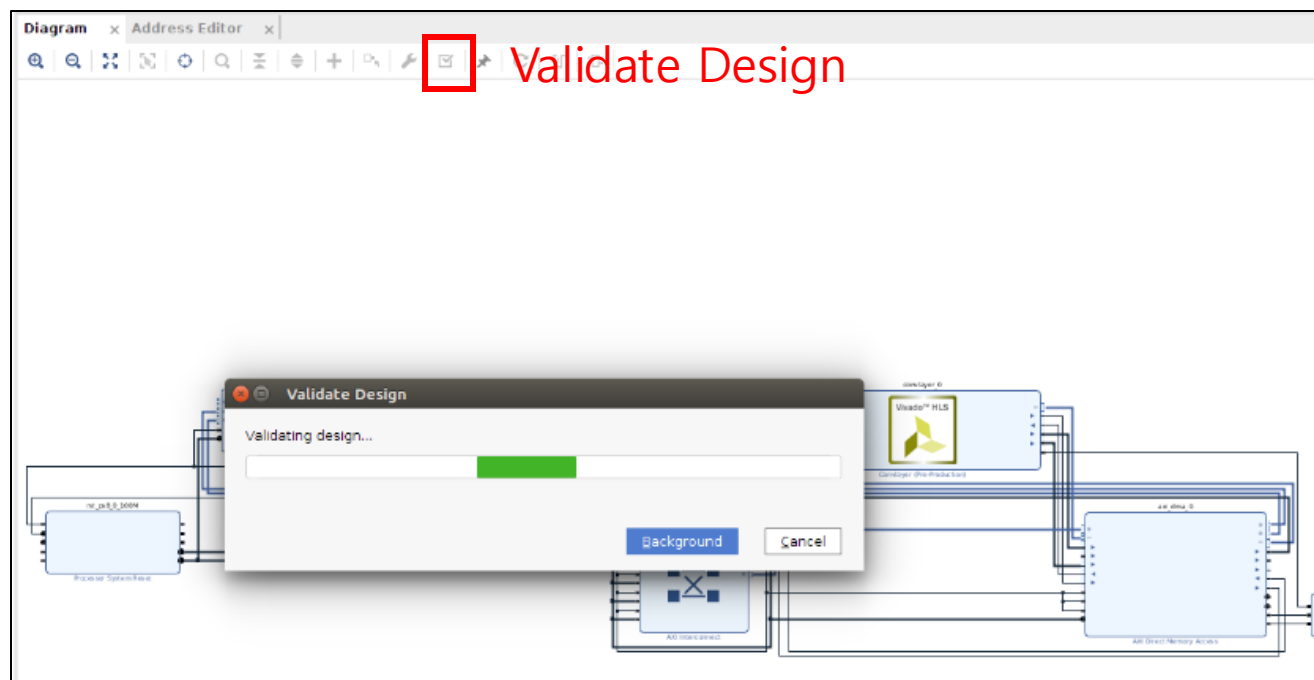
# Vivado Project – Block Design

- + 버튼 눌러 block 추가
  - 앞서 IP Repository에 불러온 HW(HLS 가져온 IP이름) 추가



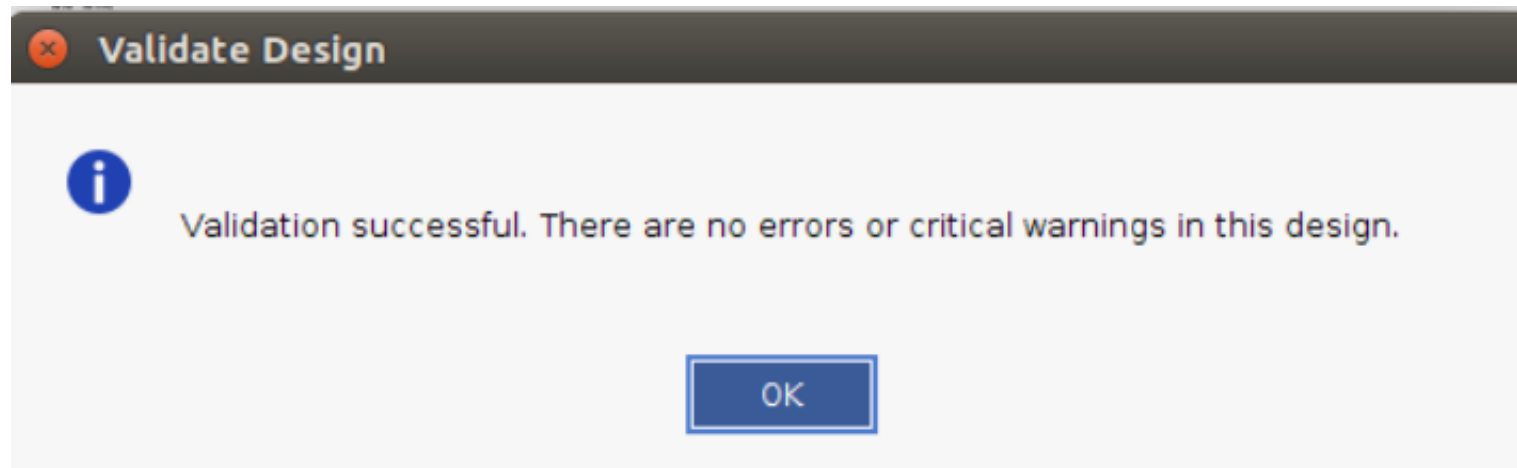
# Vivado Project – Block Design

- 블록 간 연결을 마쳤으면 이 후 단계 진행
- Validate Design 클릭



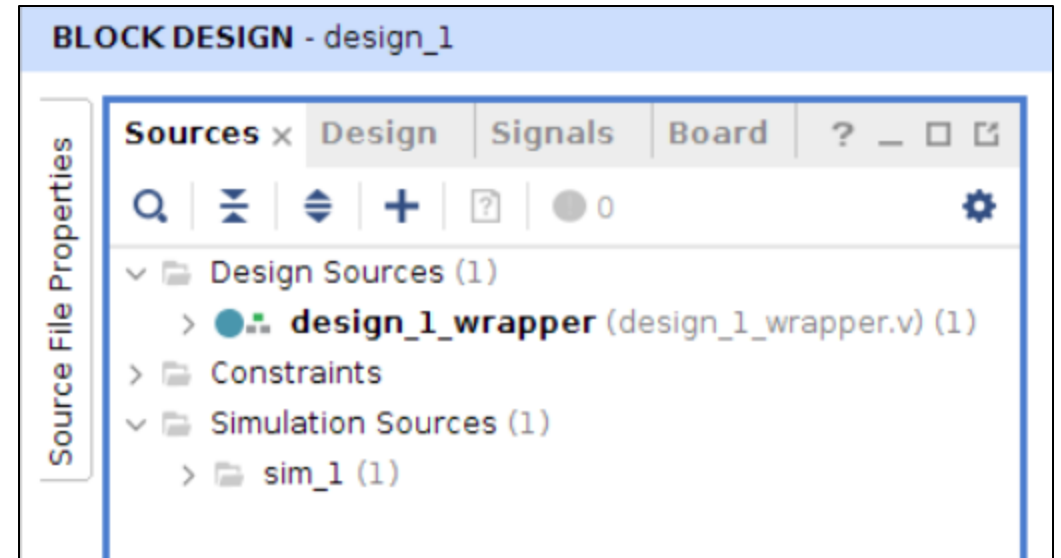
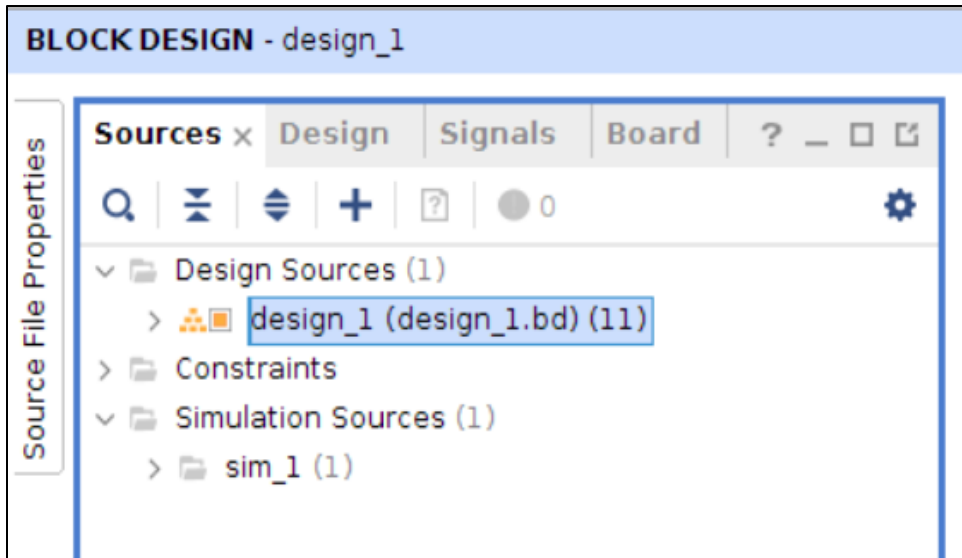
# Vivado Project – Block Design

- 다시 Validate Design하면 successful



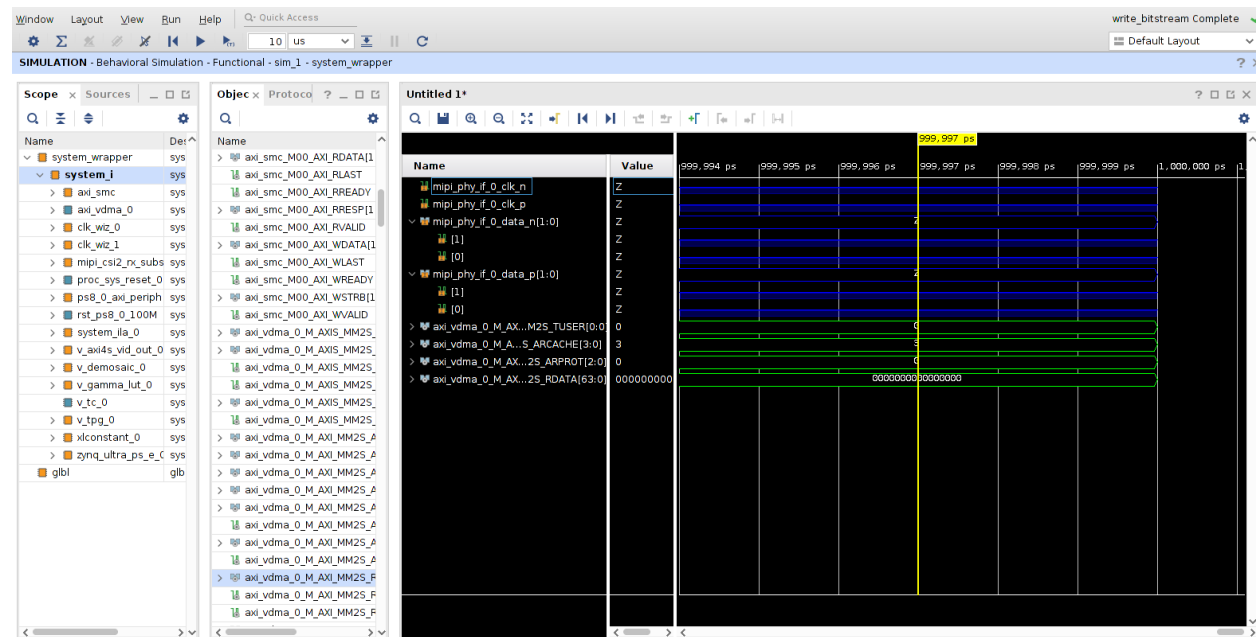
# Vivado Project – HDL Wrapper 생성

- Sources
  - Design Sources
    - design\_1.bd를 오른쪽 클릭 → Create HDL Wrapper
    - Let vivado manage wrapper and auto-update 선택 후 변환 진행

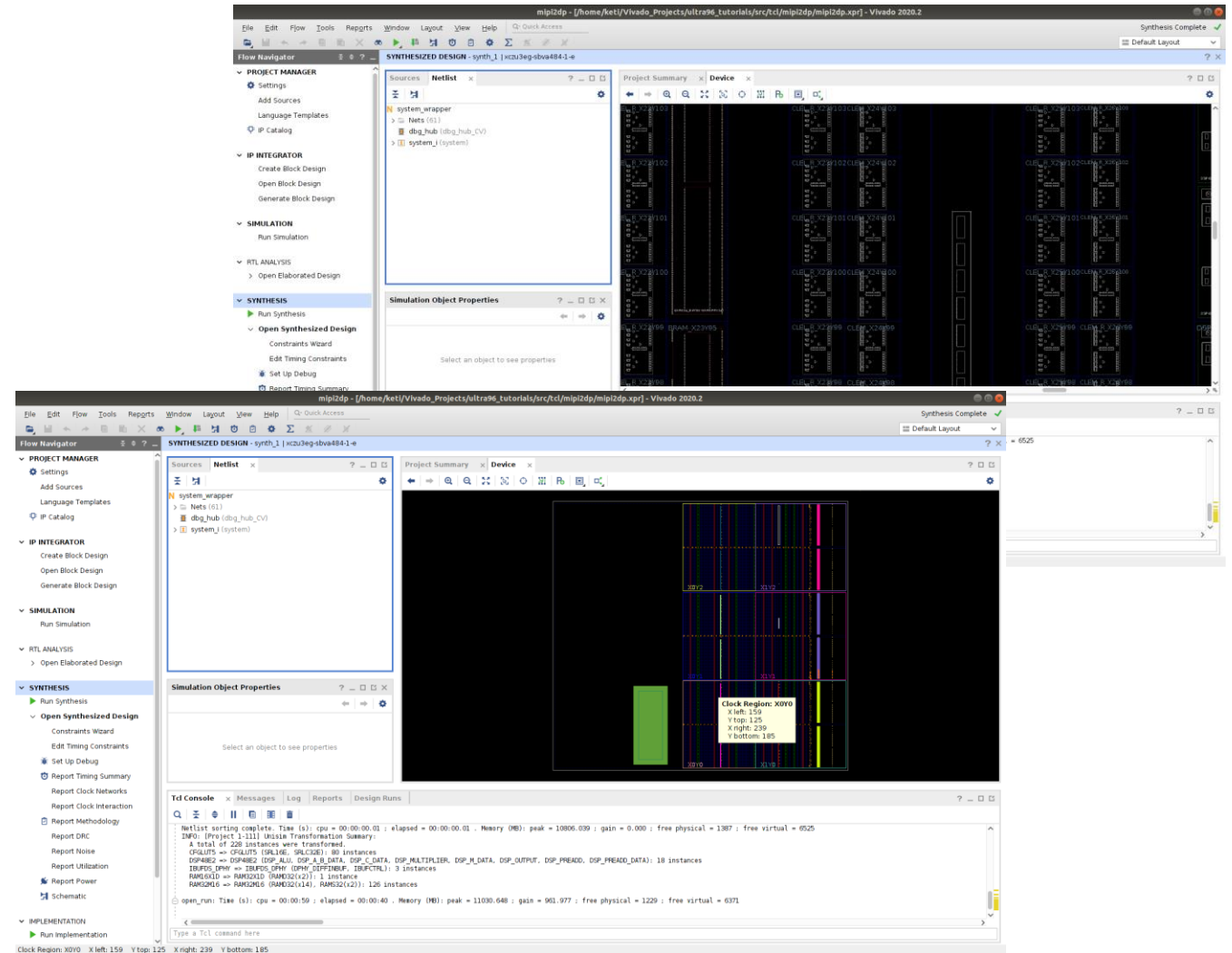


# Vivado Project – Simulation

- Run Simulation
  - TestBench에서 작성한 데이터를 기반으로 시뮬레이션 진행
  - Open Elaborated Design
    - DRC 체크
  - 우측에 waveview로 데이터 확인

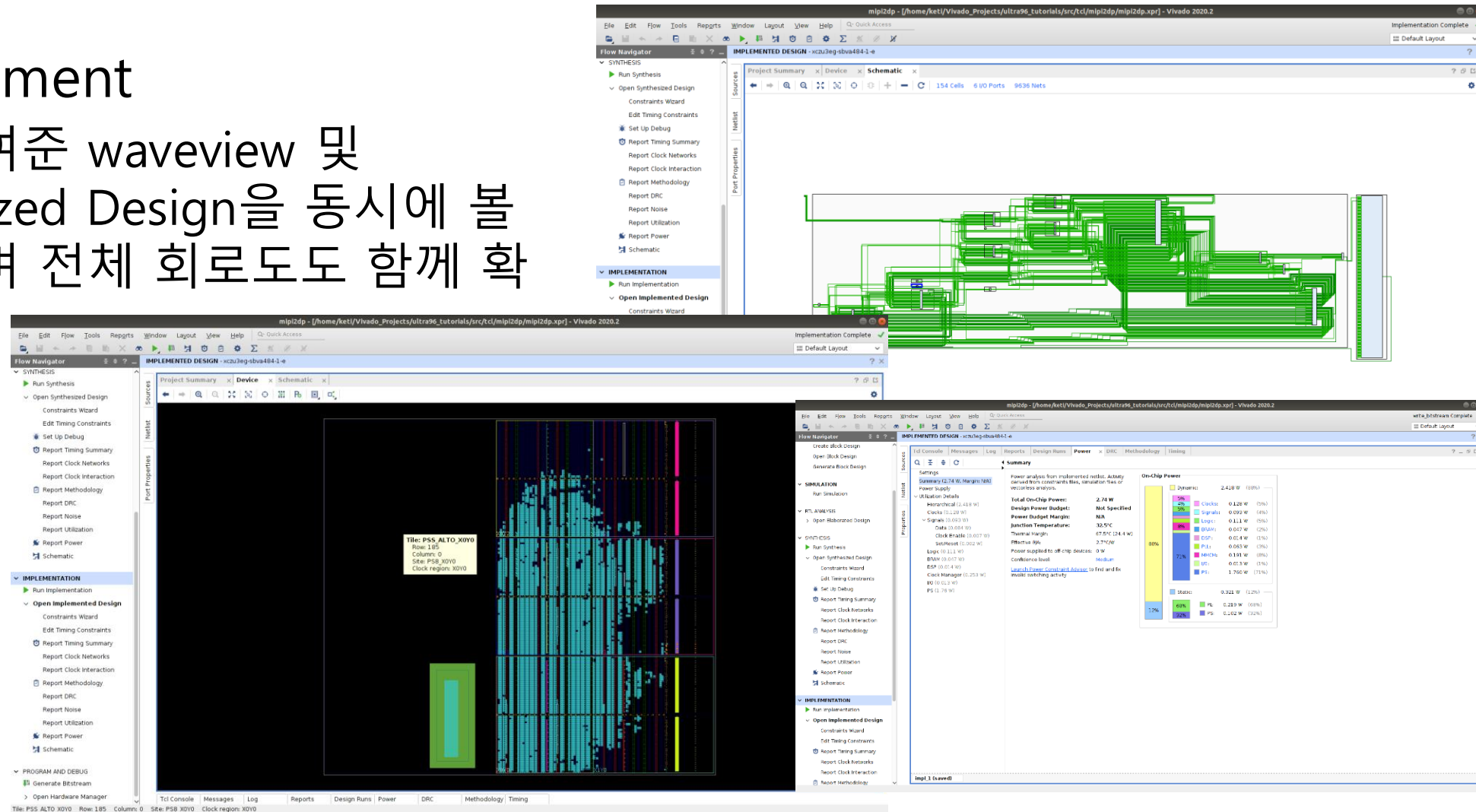


- Run Synthesis
  - Open Synthesized Design
    - Netlist
    - Schematic
    - DRC 체크



# Vivado Project – Implement

- Run Implement
  - 앞서 보여준 waveview 및 Synthesized Design을 동시에 볼 수 있으며 전체 회로도도 함께 확인 가능



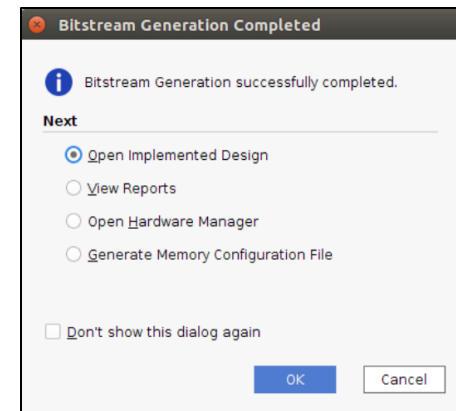
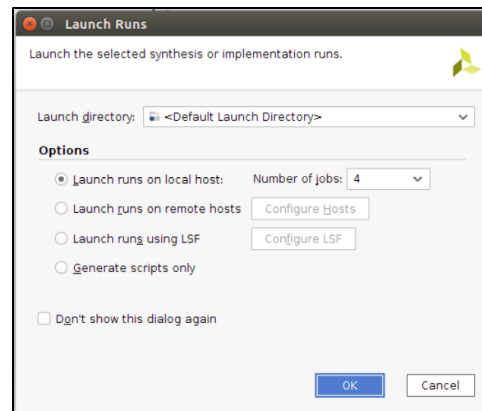
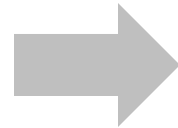
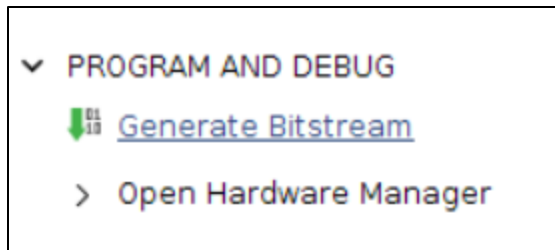


# Vivado Project – Generate Bitstream

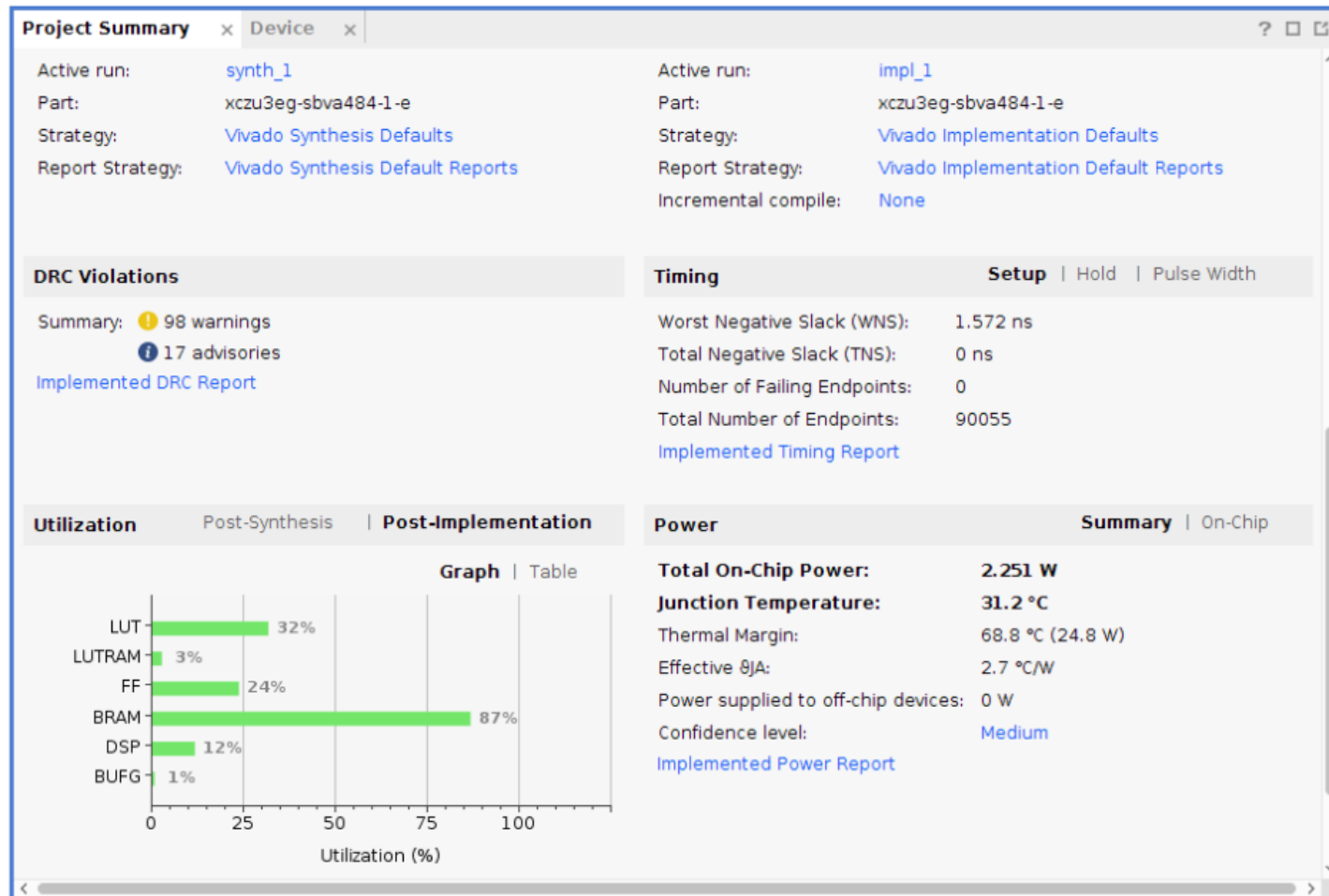
- Flow Navigator
  - Program and DEBUG
    - Generate Bitstream : 아래 과정 수행
    - Bitstream 완료 시

- Elaboration(RTL 코드 분석)
- Synthesis(RTL 코드 합성하여 netlist 생성)
- Implementation(device에 맞게 Place&Route)
- generate bitstream(FPGA에 올릴 수 있도록 .bit 파일 생성)

4개 항목 중 선택하여 확인 가능

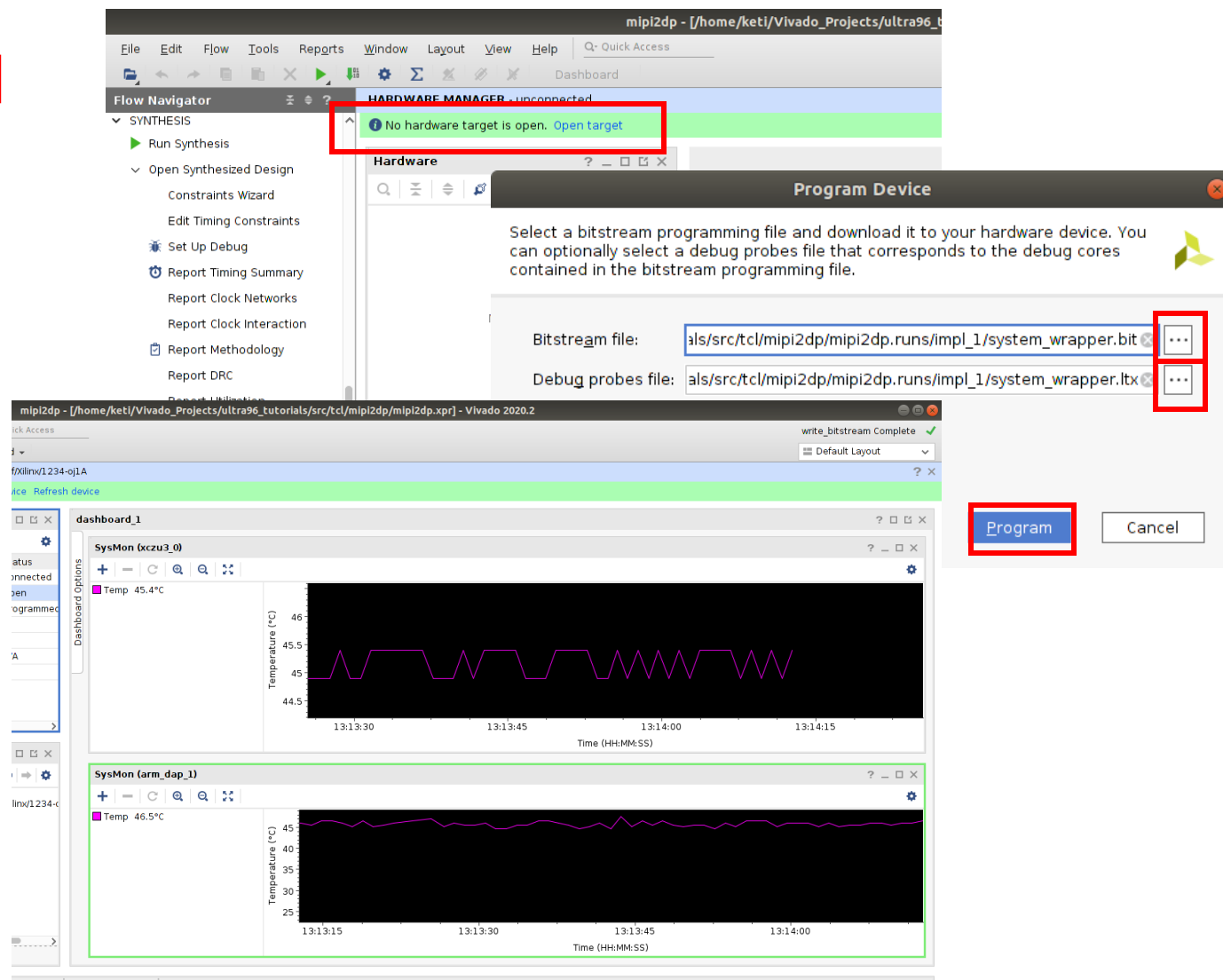


# Vivado Project – Generate Bitstream



# Vivado Project – Programming Device

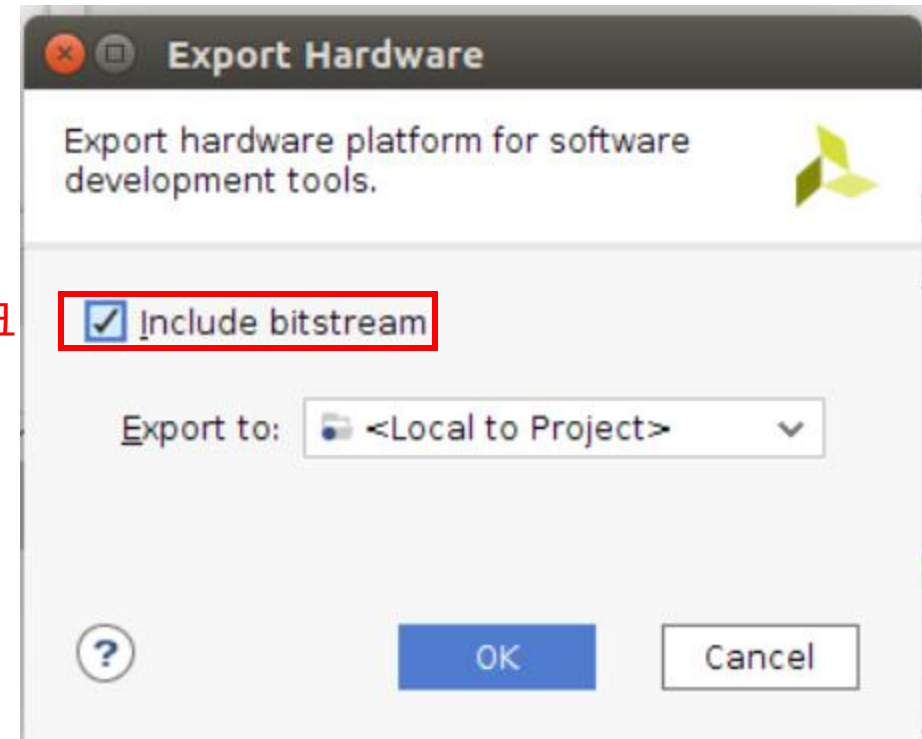
- PS (ARM Core) 영역도 같이 사용해야 하면 Bitstream 추출
- Generate Bitstream
  - Open Hardware Manager
    - Open Target 클릭
    - Program Device 클릭
    - Bitstream File 경로 선택
    - Ltx File 경로 선택
    - Program 클릭
    - Add Configuration Memory Device 클릭 후 Memory name 선택 후 진행



# Vivado Project – Export BIT

- 메뉴의 File
  - Export
    - Export Hardware (회로도 및 블록 디자인 등을 추출)
- 메뉴의 Tools
  - Launch Vitis IDE

반드시 체크



# Vitis IDE

- Vitis IDE(Software Development Kit)
  - Create
  - Program
    - Build, Compiling, Debugging 등 구동 전 체크
    - FPGA+CPU 구동 프로그램 작성 및 동작 확인


# Vitis IDE - Create

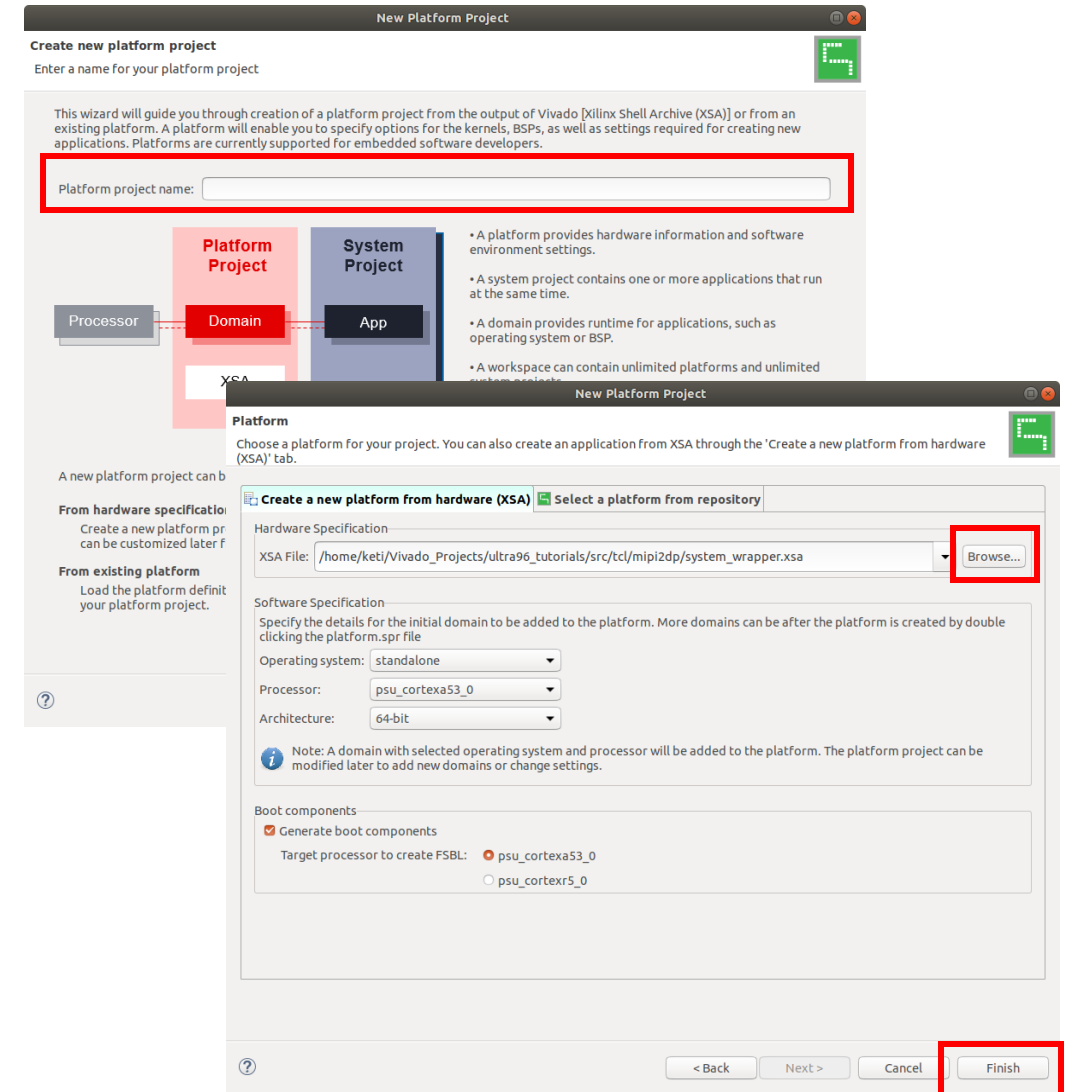
- Vivado Projects 상단 바
  - Tools > Launch Vitis IDE 실행



클릭 후 Launch Vitis IDE 실행

# Vitis IDE - Create

- 좌측 상단의 
- Create Platform Project
- Project name 정하고 next
- Bitstream 추출한 폴더로 경로 지정 후 wrapper.xsa 파일 가져 오기



# Vitis IDE - Create

- 좌측 상단의
  - Platform 생성 후 상단의 그림과 같이 zynqmp\_fsbl, standalone on psu\_cortexa53\_0의 Board Support Package 클릭
- Modify BSP Settings 클릭
  - 클릭 후 아래 그림과 같은 화면이 나오면 standalone 클릭 후 stdin, stdout의 Value 값을 psu\_uart\_1로 변경

The screenshot shows the Vitis IDE interface. The top panel displays the project tree with 'test\_platform' selected. Under 'zynqmp\_fsbl', 'standalone on psu\_cortexa53\_0' is highlighted. The right panel shows the 'Board Support Package' settings for 'standalone'. The 'Modify BSP Settings...' button is highlighted. Below this, the 'Board Support Package Settings' dialog is open, showing the 'Configuration for OS: standalone' tab. The 'stdin' and 'stdout' settings are highlighted, both with a value of 'psu\_uart\_1'.

Board Support Package Settings

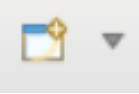
Control various settings of your Board Support Package.

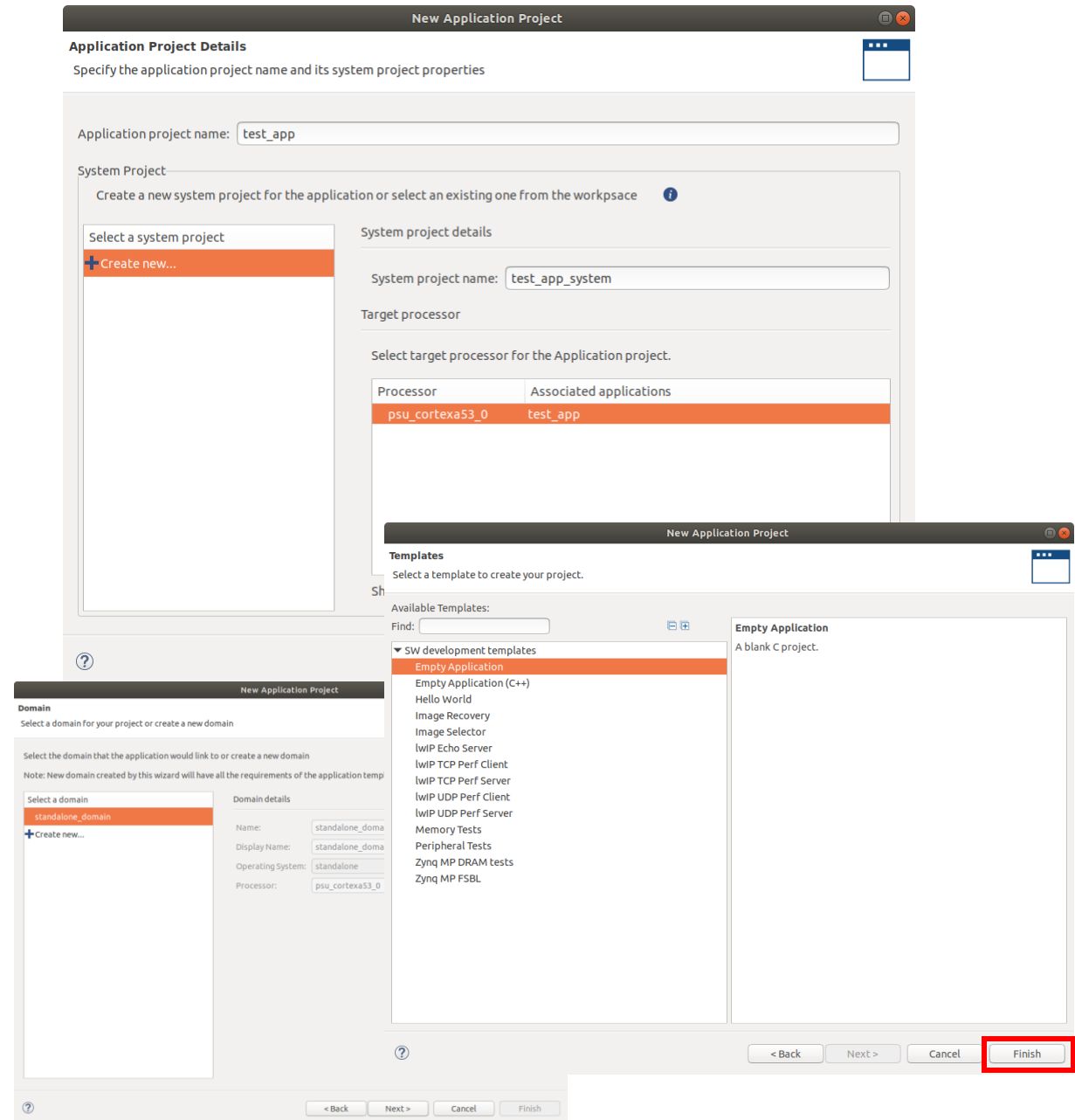
Configuration for OS: standalone

Name	Value	Default	Type	Description
clocking	false	false	boolean	Enable clocking support
hypervisor_guest	false	false	boolean	Enable hypervisor guest support
lockstep_mode_debug	false	false	boolean	Enable debug logic in non-lockstep mode
sleep_timer	none	none	peripheral	This parameter is used to select the sleep timer peripheral
stdin	psu_uart_1	none	peripheral	stdin peripheral
stdout	psu_uart_1	none	peripheral	stdout peripheral
ttc_select_counter	2	2	enum	Selects the counter to be used for timing
zynqmp_fsbl_bsp	false	false	boolean	Disable or Enable Optimized BSP
microblaze_exceptions	false	false	boolean	Enable MicroBlaze Exception Handling
enable_sw_intrusive_profiling	false	false	boolean	Enable S/W Intrusive Profiling



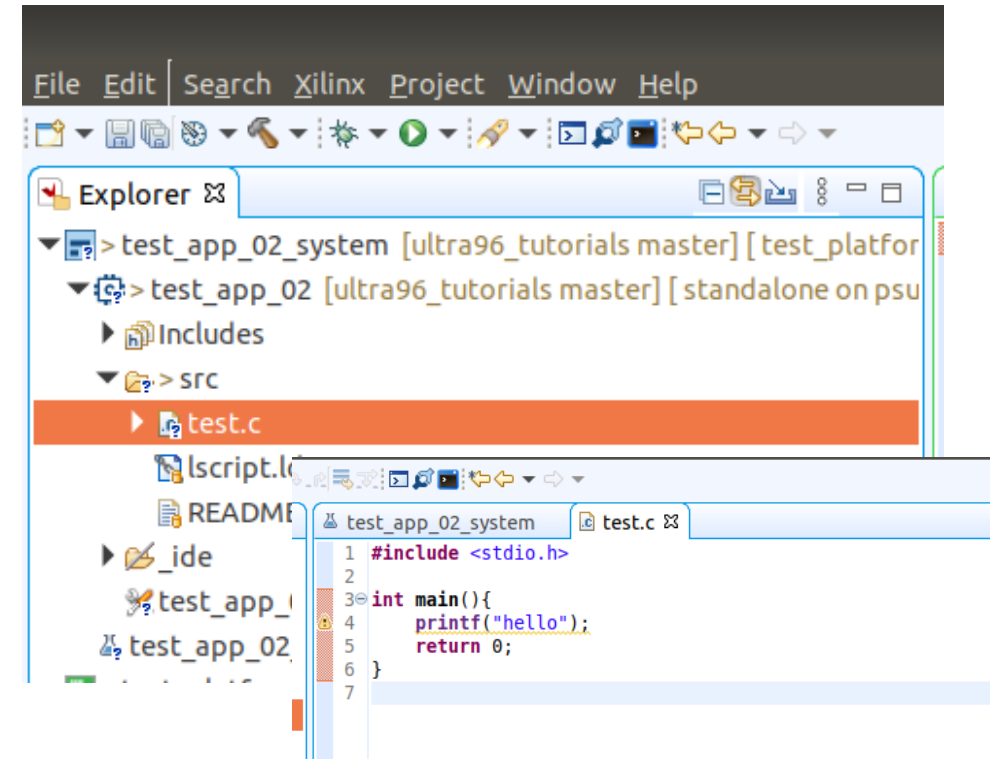
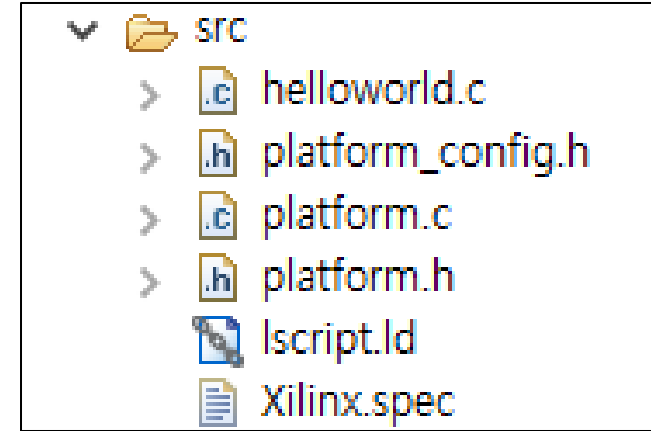
# Vitis IDE - Create

- 좌측 상단의 
- Application Project
- Project name 정하고 next
- Hello World → Finish
- Or Empty Application → Finish



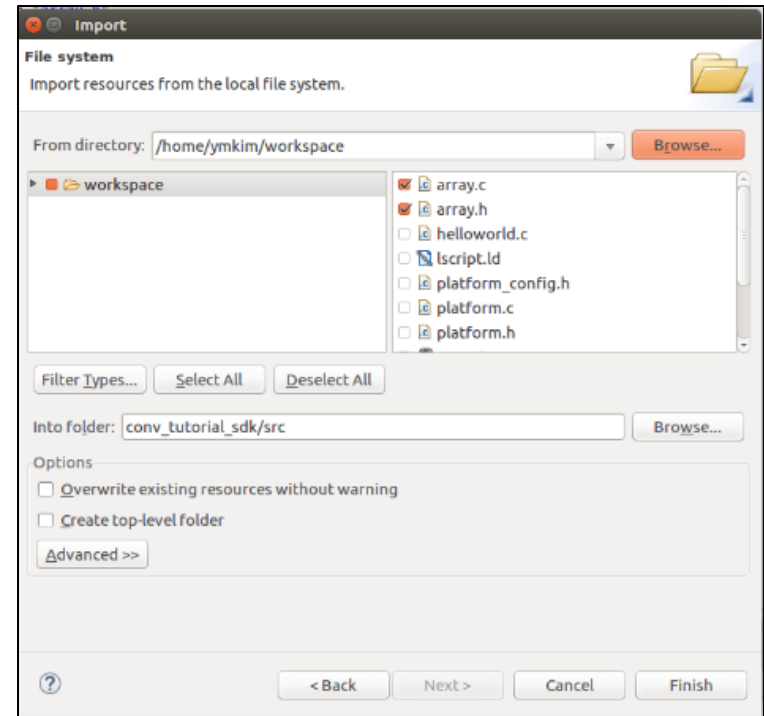
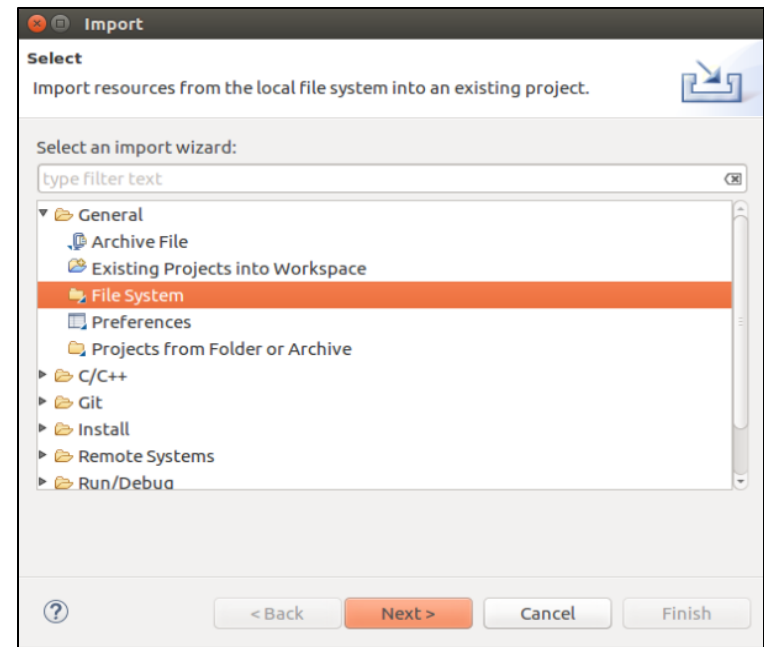
# Vitis IDE - Program

- Hello\_world로 시작 시 기본적으로 src 폴더 내에 다음과 같은 파일이 생성 되어있음
  - helloworld.c
  - platform.c
  - platform.h
  - platform\_config.h
  - 이 후 helloworld.c 내용을 수정하여 원하는 코드 작성
- Empty Application 으로 시작 시
  - 아무 파일도 없으며, src 오른쪽 클릭 후 new file > .c/.cpp 파일로 새로 생성 후 내용 작성



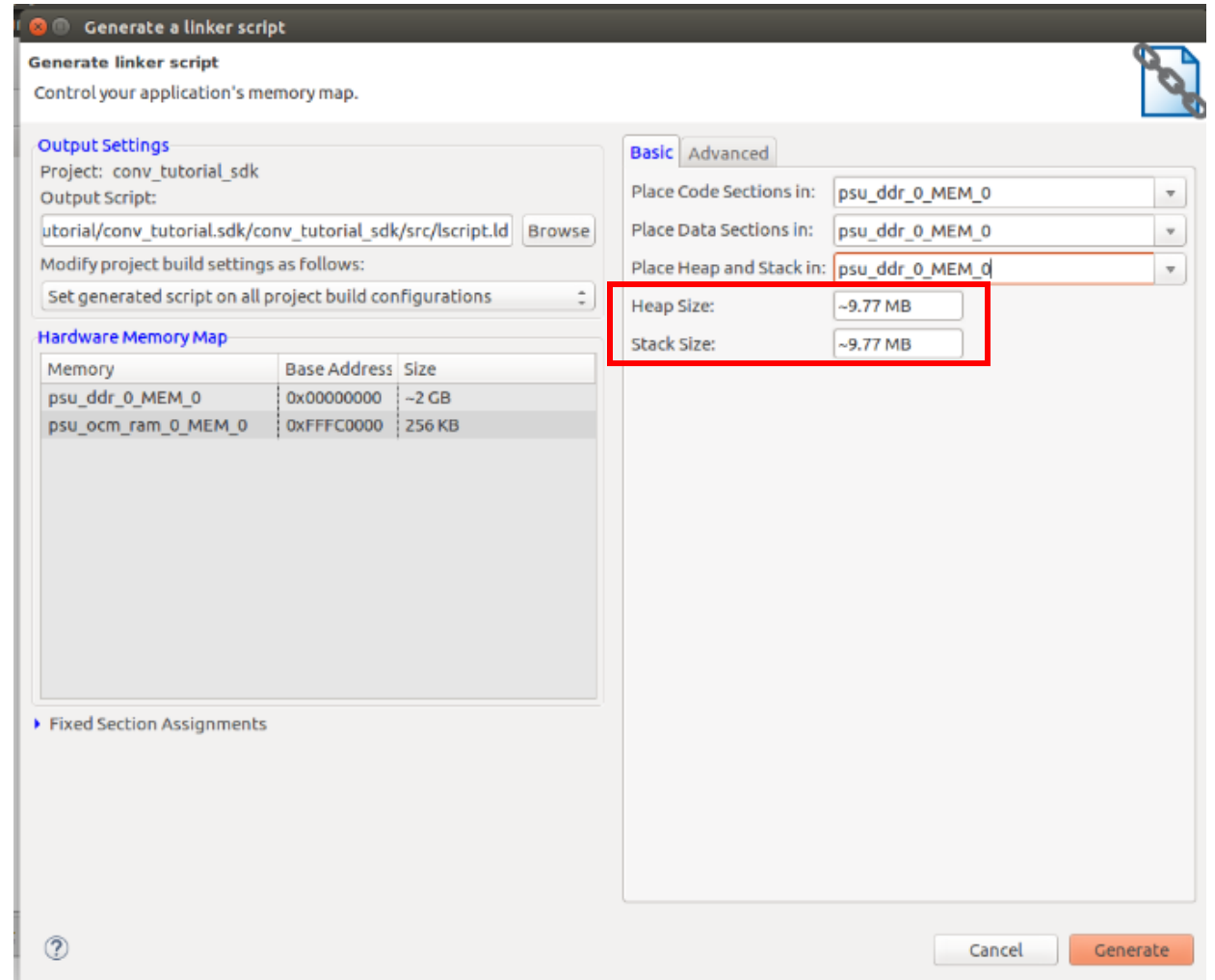
# Vitis IDE - Program

- Program에 사용할 헤더파일을 include 하기 위해서
- src 폴더 우클릭
  - Import
    - General
      - File System → Next
      - Browse로 헤더파일이 있는 폴더로 가서 그 파일들 체크 후 finish
- cp 명령어로 가져와도 됨



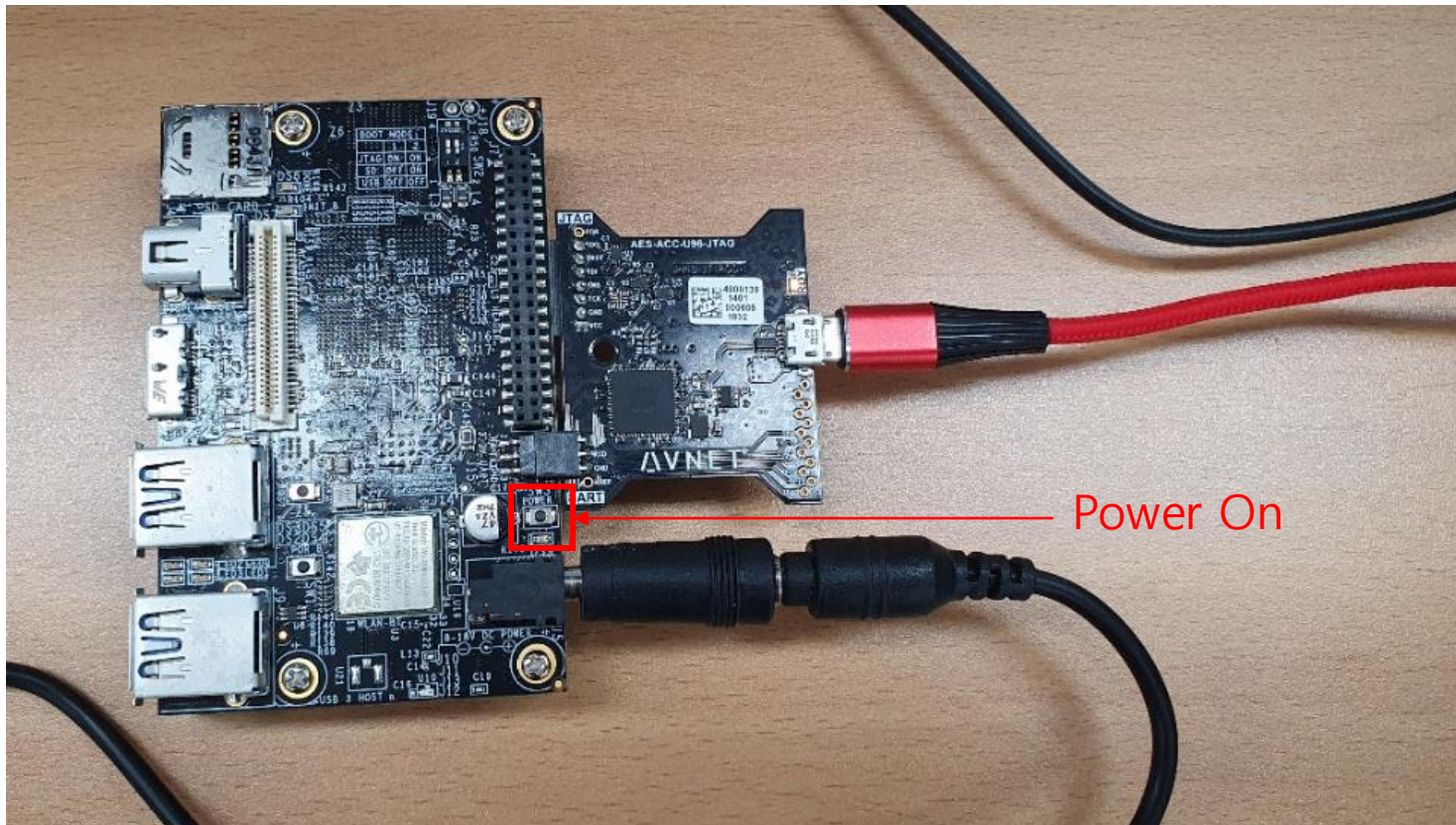
# Vitis IDE - Program

- Application 클릭 후 메뉴의 Xilinx
  - Generate linker script
  - Heap, Stack size 최대



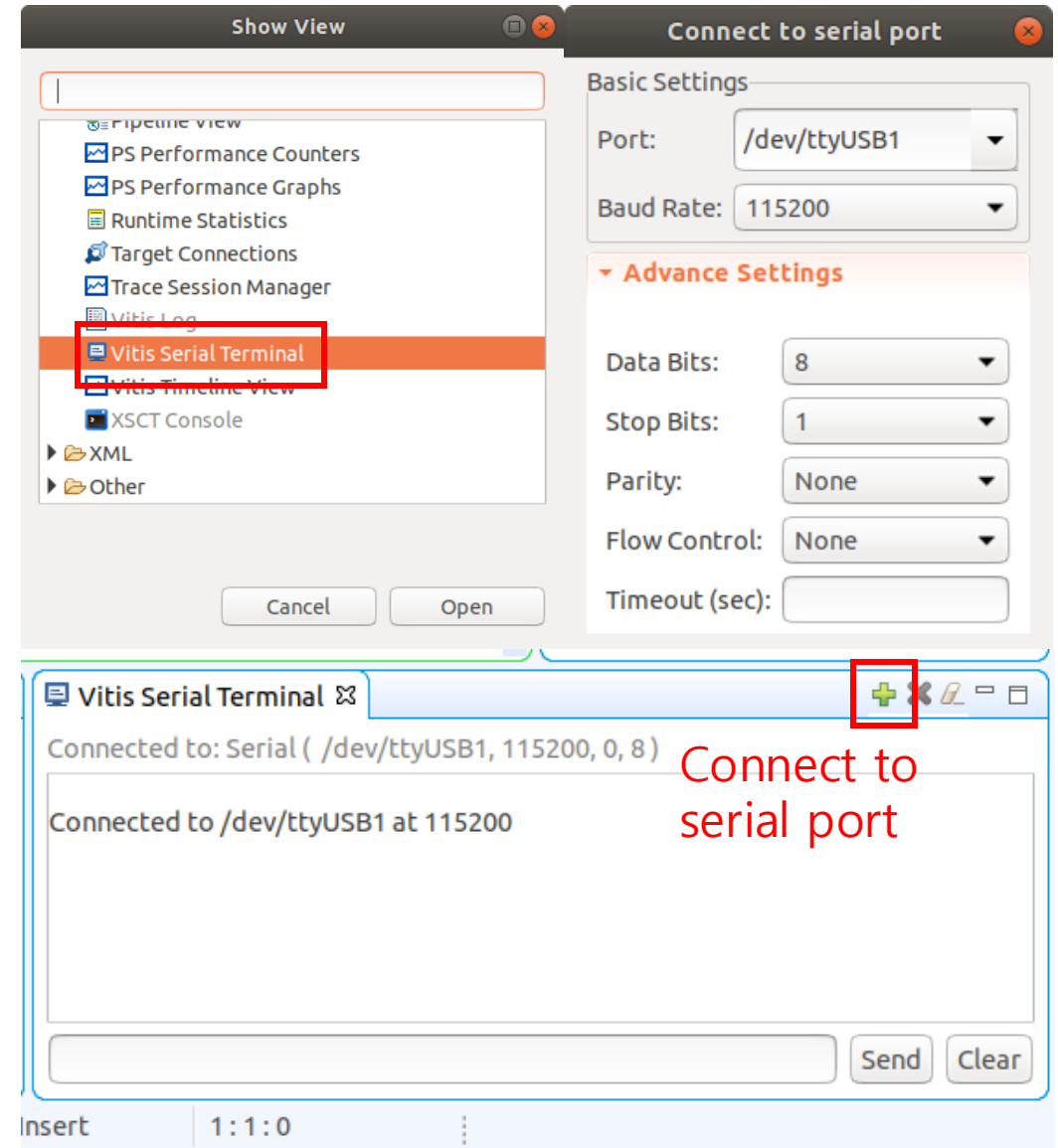
# Vitis IDE - Program

- Ultra96 board 연결



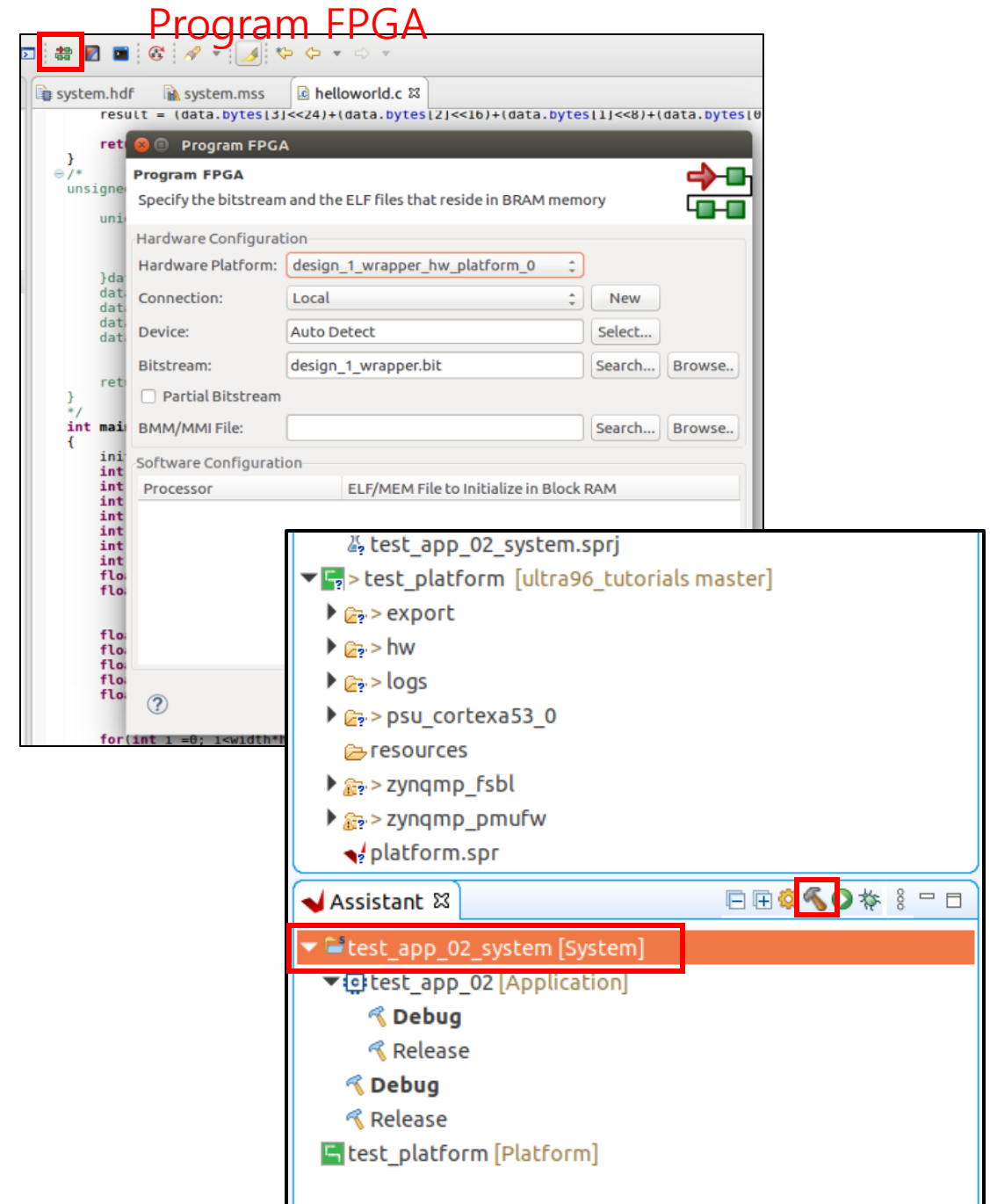
# Vitis IDE - Program

- 출력을 확인하기 위해 IDE terminal에 serial port 연결
  - 상단의 Window 메뉴바에서 show view 클릭 > Xilinx 폴더의 Vitis Serial Terminal 클릭
  - 이 후 우측 그림과 같이 +버튼 클릭
  - 포트 설정 후 OK 클릭
  - Linux는 /dev 에 포트(ttyUSB) 존재
  - Windows는 제어판→장치관리자 →포트(COM&LPT)→COM 번호 확인



# Vitis IDE - Program

- HW에 bitstream을 올리고  
작성한 프로그램을 실행
  - Program FPGA → program
- SW와 같이 구동 하고 싶을 시
  - app system 하단의 test app 02를  
빌드





# Vitis IDE - Program

- Application 우클릭
  - Run As..
    - Launch on Hardware(Single Application Debug) 클릭
- Program 실행 결과
  - ex) Hello 출력
  - ex) CNN 가속 (FPGA 사용 / 미사용 비교)

