

# Xilinx Vivado 2021.2 Training Course



# Agenda

- Installation of Viavado
- Vivado Design Suite Flow
- Introduction of Vivado Basic Interface
- Behavioral Simulation
- Vivado Synthesis
- Vivado Design Suite I/O Pin Planning
- Vivado Clock Constraints
- Vivado Implementation
- Vivado Design Checkpoint
- Programming Devices
- Vivado IP flow
- Vivado Logic Analyzer
- Vivado Reports
- Vivado Configuration
- Vivado TCL Environment

# Installation of Vivado



# Vivado ML Edition Features

Vivado ML Edition Features	Vivado ML Standard Edition	Vivado ML Enterprise Edition	Vivado Lab Edition
Licensing Option	Free	30-day Evaluation - Free On Demand on AWS Marketplace NL: \$2995 FL: \$3595	
Device Support	Limited Xilinx Devices	All Xilinx Devices	
Vivado IP Integrator	•	•	
Dynamic Function eXchange	•	•	
Vitis High-Level Synthesis	•	•	
Vivado Simulator	•	•	
Vivado Device Programmer	•	•	•
Vivado Logic Analyzer	•	•	•
Vivado Serial I/O Analyzer	•	•	•
Debug IP (ILA/VIO/IBERT)	•	•	
Synthesis and Place and Route	•	•	
Vitis Model Composer	Buy NL - \$500 FL - \$700	Buy NL - \$500 FL - \$700	

# Vivado ML Edition Features

- Xilinx Devices Support

Device	Vivado ML Standard Edition	Vivado ML Enterprise Edition
Zynq®	Zynq-7000 SoC Device: • XC7Z010, XC7Z015, XC7Z020, XC7Z030, XC7Z007S, XC7Z012S, and XC7Z014S	Zynq-7000 SoC Device: • All
Zynq® UltraScale+™ MPSoC	UltraScale+ MPSoC: • XCZU2EG, XCZU2CG, XCZU3EG, XCZU3CG XCZU4EG, XCZU4CG, XCZU4EV, XCZU5EG, XCZU5CG, XCZU5EV, XCZU7EV, XCZU7EG, and XCZU7CG	UltraScale+ MPSoC: • All
Zynq UltraScale+ RFSoC	UltraScale+ RFSoC: • None	UltraScale+ RFSoC: • All
Alveo	Alveo: • All	Alveo: • All
Kria	Kria • All	Kria: • All
Versal	N/A 	AI Core Series: • VC1902 • VC1802 Prime Series • VM1802
Virtex FPGA	Virtex-7 FPGA: • None  Virtex UltraScale FPGA: • None	Virtex-7 FPGA: • All  Virtex UltraScale FPGA: • All  Virtex UltraScale+ FPGA: • All  Virtex UltraScale+ HBM: • All  Virtex UltraScale+ 58G: • All

Device	Vivado ML Standard Edition	Vivado ML Enterprise Edition
Kintex FPGA	Kintex®-7 FPGA: • XC7K70T, XC7K160T	Kintex®-7 FPGA: • All
	Kintex UltraScale FPGA: • XCKU025, XCKU035	Kintex UltraScale FPGA: • All
	Kintex UltraScale+ FPGA: • XCKU3P, XCKU5P	Kintex UltraScale+: • All
Artix FPGA	Artix-7 FPGA: • XC7A12T, XC7A15T, XC7A25T, XC7A35T, XC7A50T, XC7A75T, XC7A100T, XC7A200T	Artix-7 FPGA: • All
Artix UltraScale+	Artix UltraScale+ • XCAU25P • XCU20P	Artix UltraScale+: • All
Spartan-7	Spartan-7: • XC7S6, XC7S15 • XC7S25, XC7S50• XC7S75, XC7S100	Spartan-7: • All

# Vivado ML Edition Features

- Operating System Support

## Microsoft Windows Support

- Windows update: 10.0 1809 Update; 10.0 1903 Update; 10.0 1909 Update; 10.0 2004 Update

## Linux Support

- RHEL 7 / Cent OS 7: 7.4, 7.5, 7.6, 7.7, 7.8, 7.9
- RHEL8/Cent OS: 8.1, 8.2, 8.3
- SUSE EL: 12.4, 15.2
- Ubuntu: 16.04.5 LTS; 16.04.6 LTS; 18.04.1 LTS; 18.04.2 LTS; 18.04.3 LTS; 18.04.4 LTS; 20.04 LTS; 20.04.1 LTS

Note: Please refer to [PetaLinux Tools Documentation: Reference Guide \(UG1144\)](#) for more information on Installation Requirements for supported Operating Systems with PetaLinux.

# Vivado Installation

Solutions   Products   Company

AMD XILINX

Xilinx is now part of AMD | [Updated Privacy Policy](#)

Home / Support / Downloads

## Downloads

Log4j Vulnerability Patch   Licensing Help   NIC Software & Drivers

Vivado (HW Developer)   Vitis (SW Developer)   Vitis Embedded Platforms   Alveo Packages   PetaLinux   Device Models   Documentation Navigator

**Version**

2022.1   2021.2   2021.1   2020.3

Vivado Archive   ISE Archive   CAE Vendor Libraries Archive

**Vivado ML Edition - 2022.1 Full Product Installation**

**Important Information**

Vivado ML 2022.1 is now available for download:

- 5-8% Versal QoR improvement
- Introducing ML-based resource estimation
- ML Strategy Runs now available for Versal devices
- Devices enabled in the Enterprise & Standard Editions of Vivado ML
  - Artix® UltraScale+™ devices: XCAU15P and XCAU10P
  - Additional Versal® Prime, Premium, AI Core, and AI Edge series devices

We strongly recommend to use the web installers as it reduces

Download Includes	Vivado ML Edition
Download Type	Full Product Installation
Last Updated	Apr 26, 2022
Answers	<a href="#">2022.x - Vivado Known Issues</a>
Documentation	<a href="#">Release Notes</a> <a href="#">OS Support Update</a> <a href="#">What's New in Vivado</a>
Support Forums	<a href="#">Installation and Licensing</a>

Feedback

# Vivado Installation

Solutions Products Company

AMD XILINX

[Xilinx Unified Installer 2022.1: Windows Self Extracting Web Installer \(EXE - 205.92 MB\)](#)  
MD5 SUM Value : 76a6221ea8eed8c635c654cc100a1cae

Download Verification [i](#)

Digests Signature Public Key

[Xilinx Unified Installer 2022.1: Linux Self Extracting Web Installer \(BIN - 266.73 MB\)](#)  
MD5 SUM Value : db5056feaaf271fe90ba54bae4768ed2

Download Verification [i](#)

Digests Signature Public Key

[Xilinx Unified Installer 2022.1 SFD \(TAR/GZIP - 73.81 GB\)](#)    
MD5 SUM Value : 0bf810cf5eaa28a849ab52b9bfdd20a5

Download Verification [i](#)

Digests Signature Public Key

Feedback

Vivado Lab Solutions - 2022.1

# Vivado Installation

Xilinx Unified 2021.2 Installer - Select Install Type

## Select Install Type

Please select install type and provide your Xilinx.com E-mail Address and password for authentication.

**User Authentication**

Please provide your Xilinx user account credentials to download the required files.  
If you don't have an account, [please create one](#). If you forgot your password, you can [reset it here](#).

E-mail Address

Password

Download and Install Now

Select your desired device and tool installation options and the installer will download and install just what is required.

Download Image (Install Separately)

The installer will download an image containing all devices and tool options for later installation. Use this option if you wish to install a full image on a network drive or allow different users maximum flexibility when installing.

Copyright © 1986-2022 Xilinx, Inc. All rights reserved.

< Back Next > Cancel

**XILINX**

# Vivado Installation

Xilinx Unified 2021.2 Installer - Select Product to Install

## Select Product to Install

Select a product to continue installation. You will be able to customize the content in the next page.

Vitis

Installs Vitis Core Development Kit for embedded software and application acceleration development on Xilinx platforms. Vitis installation includes Vivado Design Suite. Users can also install Vitis Model Composer to design for AI Engines and Programmable Logic in MATLAB and Simulink.

Vivado

Includes the full complement of Vivado Design Suite tools for design, including C-based design with Vitis High-Level Synthesis, implementation, verification and device programming. Complete device support, cable driver, and Document Navigator included. Users can also install Vitis Model Composer to design for AI Engines and Programmable Logic in MATLAB and Simulink.

BootGen

Installs Bootgen for creating bootable images targeting Xilinx SoCs and FPGAs.

Lab Edition

Installs only the Xilinx Vivado Lab Edition. This standalone product includes the Vivado Device Programmer and Vivado Logic Analyzer tools.

Hardware Server

Installs hardware server and JTAG cable drivers for remote debugging.

Documentation Navigator (Standalone)

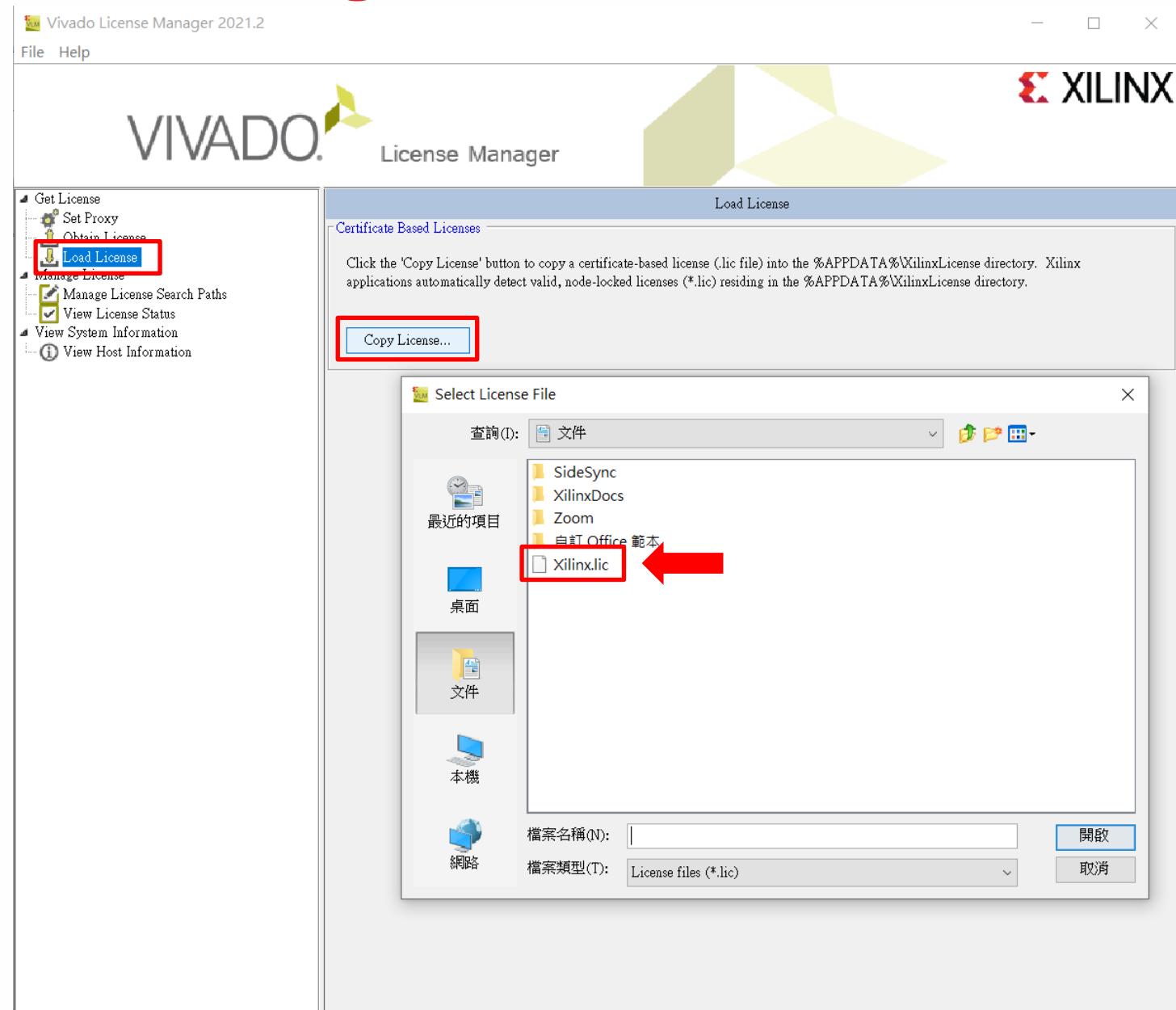
Xilinx Documentation Navigator (DocNav) provides access to Xilinx technical documentation both on the Web and on the Desktop. This is a standalone installation without Vivado Design Suite.

Copyright © 1986-2022 Xilinx, Inc. All rights reserved.

< Back    Next >    Cancel

XILINX

# Vivado License Management



# Vivado License Management

Vivado License Manager 2021.2

File Help

VIVADO License Manager XILINX

Get License

- Set Proxy
- Obtain License
- Load License

Manage License

- Manage License Search Paths
- View License Status** (highlighted with a red box)

View System Information

- View Host Information

View License Status

Certificate Based Licenses:

Filter:  Hide Free Built-in Licenses

License Name	Tools/IP	Expiration Date	Version Limit	License Type	Location	# of S
Analyzer	Tools	Permanent	2021-07-01	Nodelocked	C:\Users\dlab\Downloads	Uncoupled
ChipScopePro_SIOTK	Tools	Permanent	2021-07-01	Nodelocked	C:\Users\dlab\Downloads	Uncoupled
ChipScopePro	Tools	Permanent	2021-07-01	Nodelocked	C:\Users\dlab\Downloads	Uncoupled
HLS	Tools	Permanent	2021-07-01	Nodelocked	C:\Users\dlab\Downloads	Uncoupled
ISE	Tools	Permanent	2021-07-01	Nodelocked	C:\Users\dlab\Downloads	Uncoupled
ISE_EMBEDDED_Edition	Tools	Permanent	2021-07-01	Nodelocked	C:\Users\dlab\Downloads	Uncoupled
ISIM	Tools	Permanent	2021-07-01	Nodelocked	C:\Users\dlab\Downloads	Uncoupled
Implementation	Tools	Permanent	2021-07-01	Nodelocked	C:\Users\dlab\Downloads	Uncoupled
PartialReconfiguration	Tools	Permanent	2021-07-01	Nodelocked	C:\Users\dlab\Downloads	Uncoupled
PlanAhead	Tools	Permanent	2021-07-01	Nodelocked	C:\Users\dlab\Downloads	Uncoupled
SDK	Tools	Permanent	2021-07-01	Nodelocked	C:\Users\dlab\Downloads	Uncoupled
SDK	Tools	Permanent	2021-07-01	Nodelocked	C:\Users\dlab\Downloads	Uncoupled
Simulation	Tools	Permanent	2021-07-01	Nodelocked	C:\Users\dlab\Downloads	Uncoupled
Synthesis	Tools	Permanent	2021-07-01	Nodelocked	C:\Users\dlab\Downloads	Uncoupled
Vivado_ML_Enterprise...	Tools	Permanent	2021-07-01	Nodelocked	C:\Users\dlab\Downloads	Uncoupled
XPS	Tools	Permanent	2021-07-01	Nodelocked	C:\Users\dlab\Downloads	Uncoupled

< >

Clear Cache Refresh

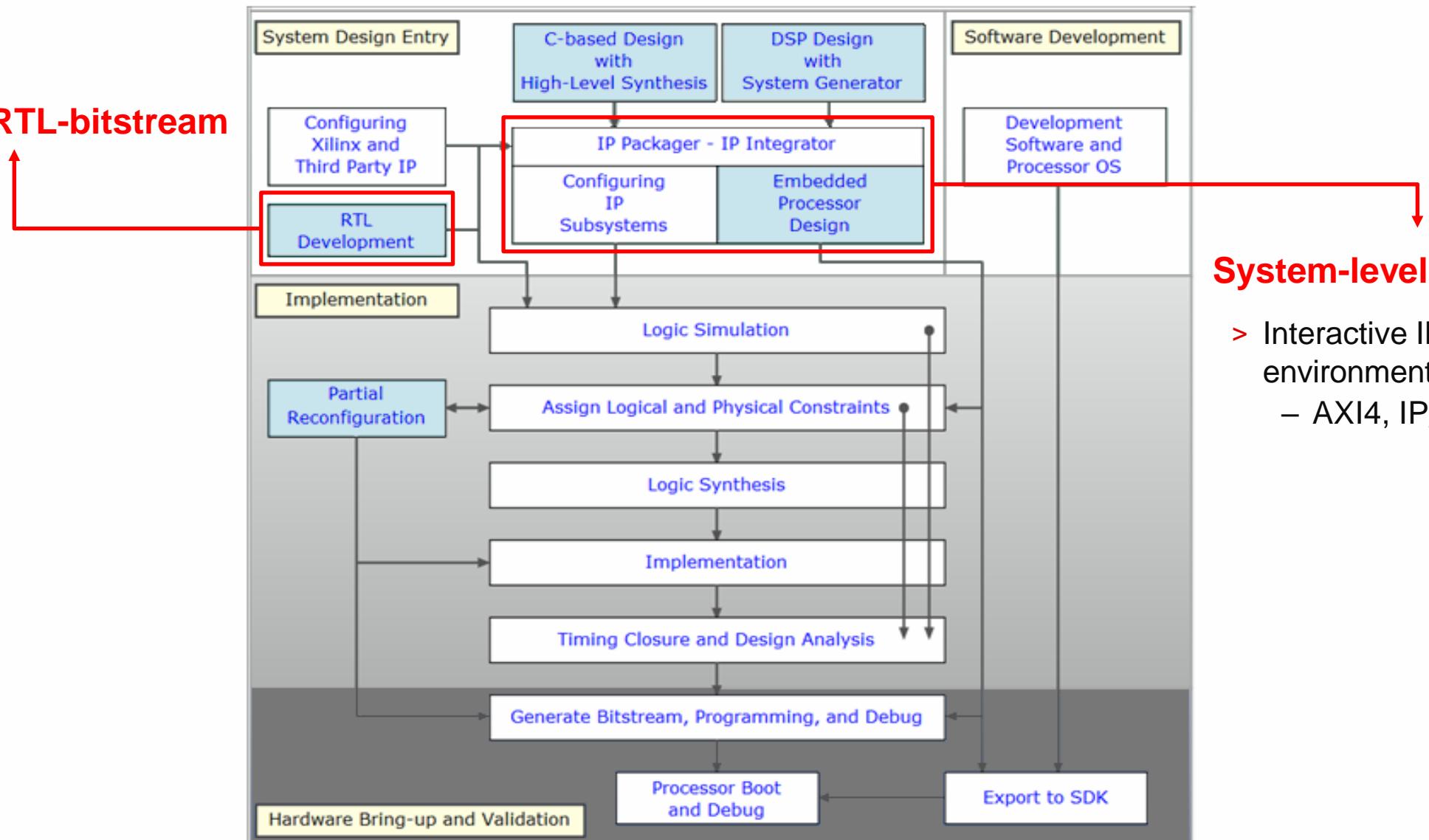
XILINX

# Vivado Design Suite Flow



# Vivado Design Flow

Traditional RTL-bitstream

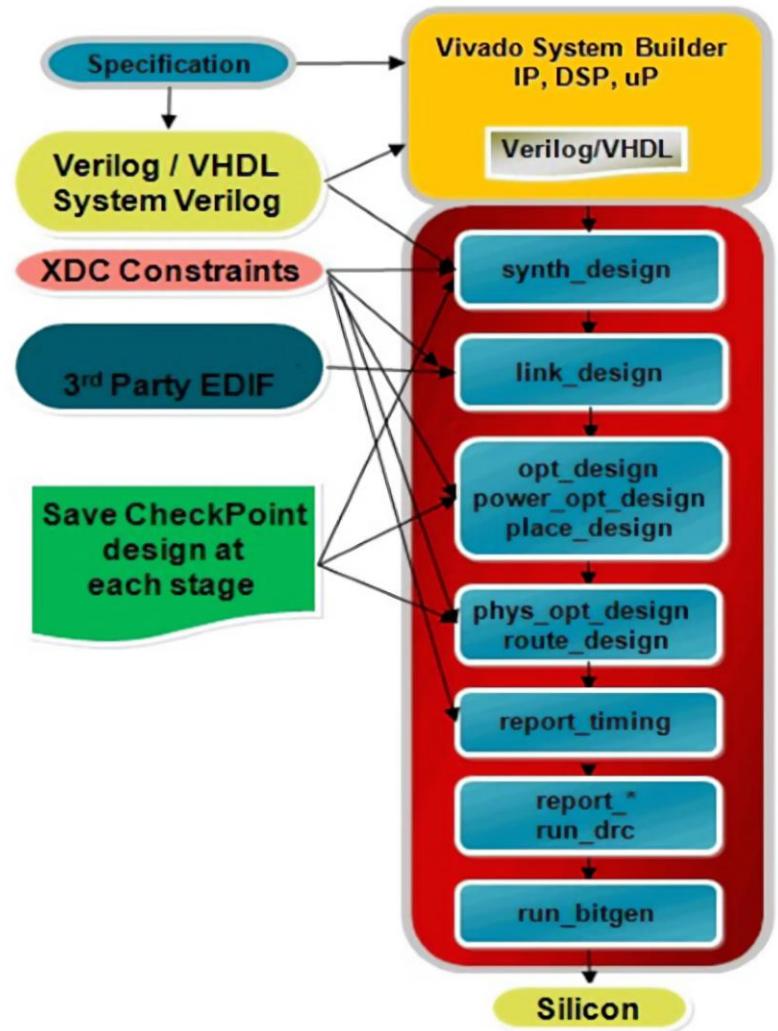


System-level Integration

- > Interactive IP plug-n-play environment
  - AXI4, IP\_XACT

# Vivado Design Flow

- Traditional RTL-bitstream
  - > Common constraint file (XDC) throughout flow
    - Apply constraints at any stage
    - Real-time debug
  - > Design analysis and reporting
    - Cross-probing
    - Report generation at any stage
    - Robust TCL API
  - > Common data model throughout the flow
    - Cross-probing support
  - > Save checkpoint designs at any stage
    - Netlist, constraints, place and route results



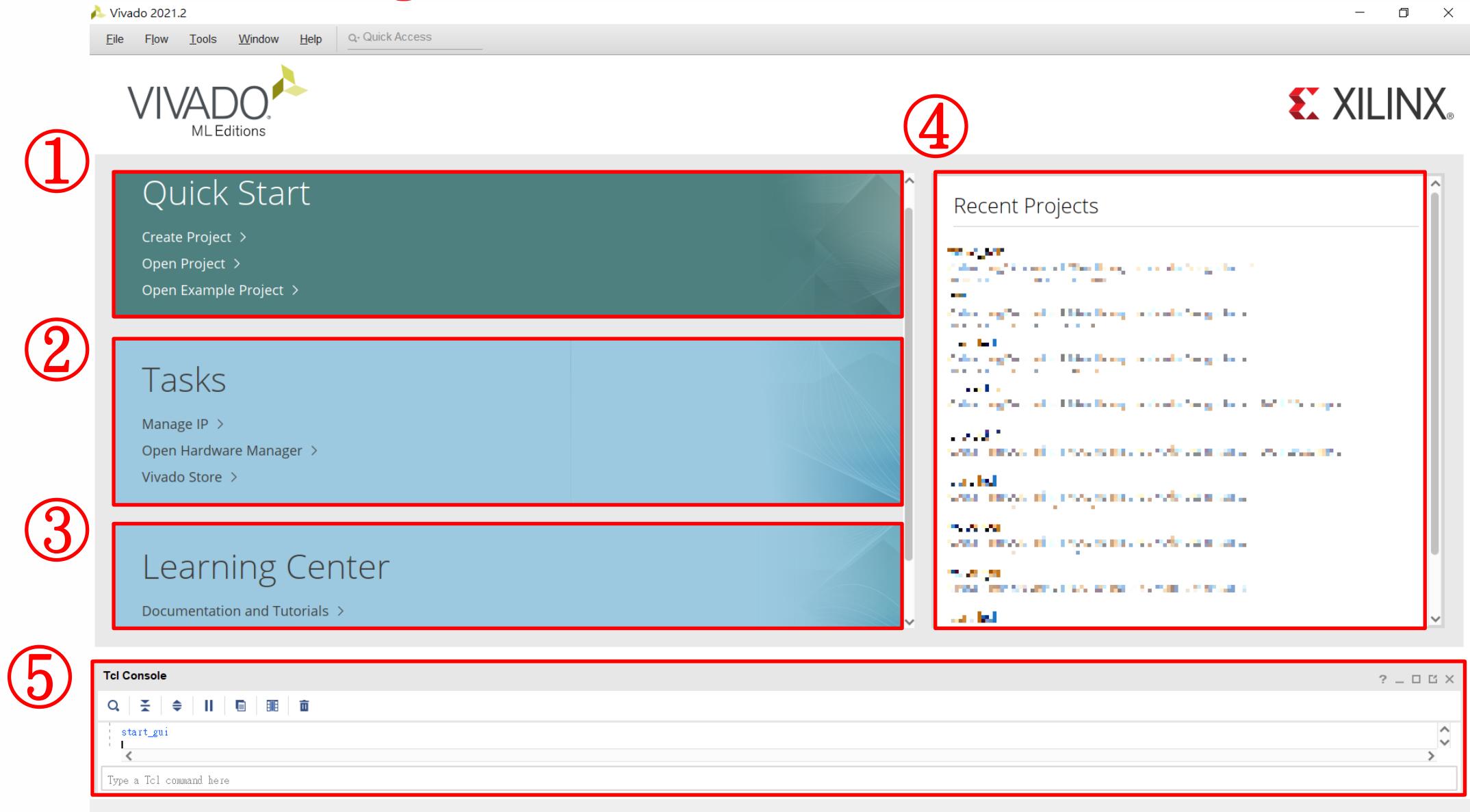
# Vivado Design Flow

- Project vs. Non-Project

		Project	Non-Project
IDE	Script	<p>High degree of automation</p> <p>: a single Tcl command – multiple steps</p> <p>ce and result management</p> <p>ole strategies: easy run</p> <pre>create_project ...     add_files ...     import_files ...     ...     launch_run synth_1     wait_on_run ...     open_run ...     report_timing_summary     ...     launch_runs impl_1     wait_on_run ...     open_run ...</pre>	<p>Fine-granularity commands</p> <ul style="list-style-type: none"><li>• More efficient scripts.</li><li>Enable focusing on specific challenges</li><li>• Easy access to all features</li><li>(ex: re-entrant routing)</li></ul> <p>File structure: fully customizable</p> <ul style="list-style-type: none"><li>• Sources and result files</li><li>• Easy adoption to revision control systems</li></ul> <p>Runtime: potentially faster</p>
	FILE STRUCTURE: fixed file organization <ul style="list-style-type: none"><li>• Revision control – many project files</li></ul>	<p>duced flexibility to work with immediate implementation results</p> <p>ll features are easily accessible (re-entrant routing)</p>	<p>Requires more in-depth knowledge</p> <p>You are responsible for</p> <ul style="list-style-type: none"><li>• Dependency file management</li><li>• Saving key result files (ex: checkpoints)</li></ul>

# Introduction of Vivado Basic Interface

# Vivado Start Page



# Vivado Start Page

## Quick Start

Create Project >

Open Project >

Open Example Project >



Create new or open existed project

## Tasks

Manage IP >

Open Hardware Manager >

Vivado Store >



Create new or open existed IP or get some successful project in Vivado

## Learning Center

Documentation and Tutorials >

Quick Take Videos >

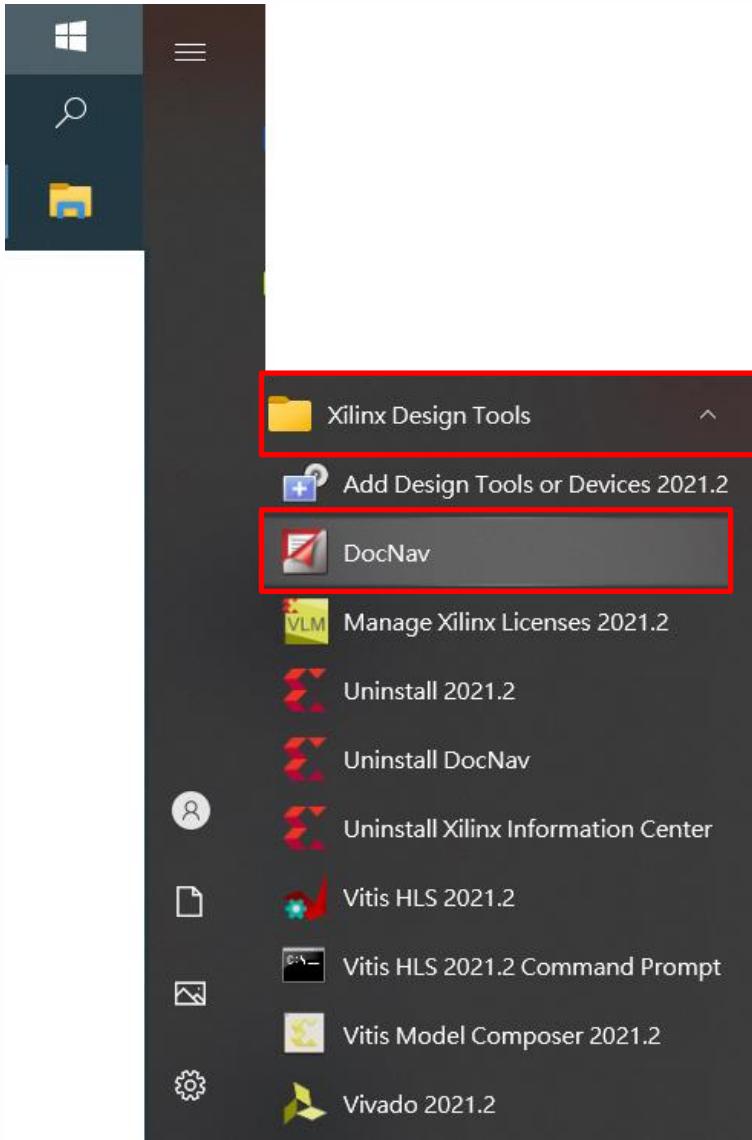
What's New in 2021.2 >



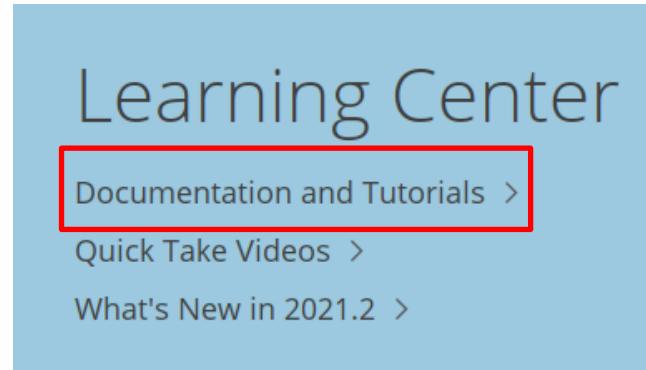
Tutorials in image, text and video form, and show the release notes which contains What's new, Known Issues, and Important Information

# Xilinx Documentation Navigator

①



②



# Xilinx Documentation Navigator

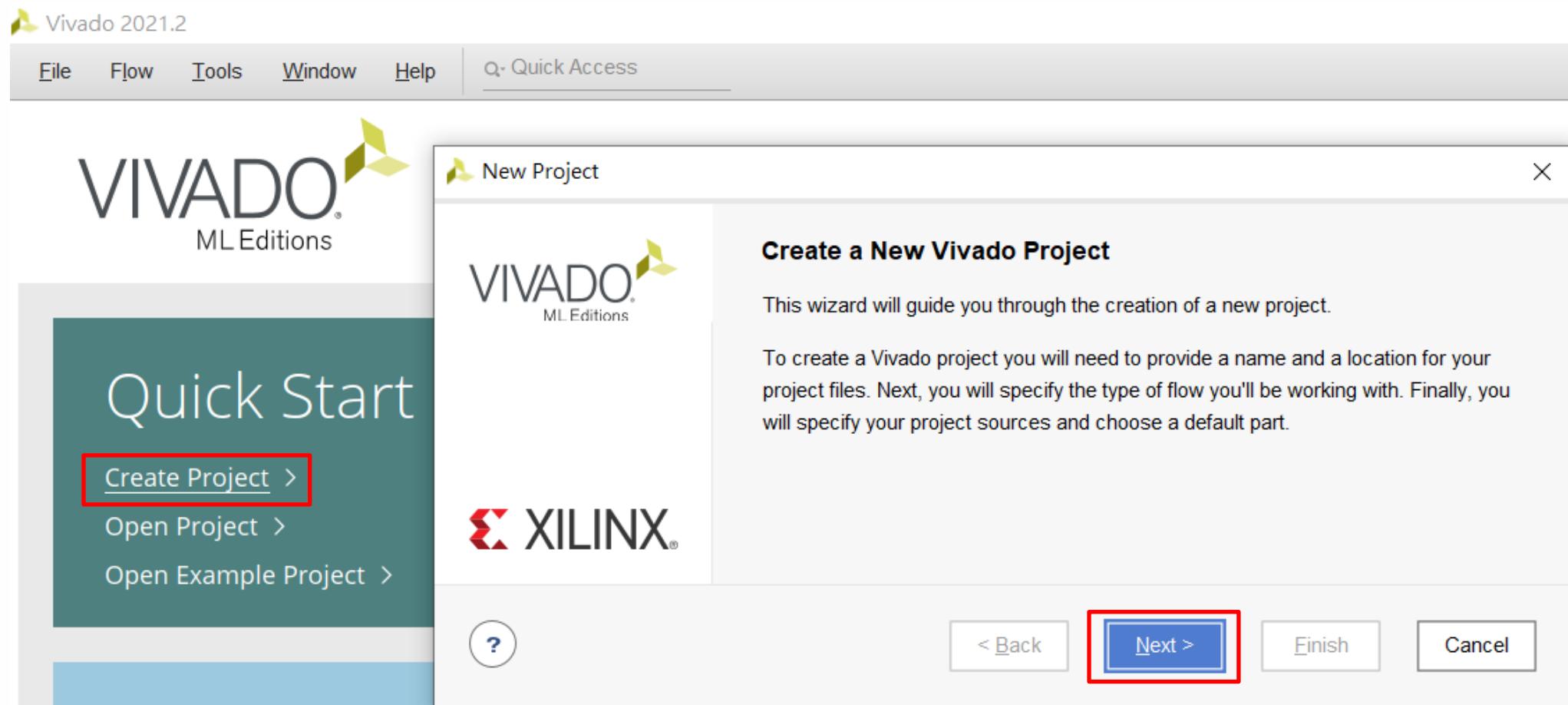
- > Xilinx Documentation Navigator
  - Contains user guides, data sheets, video tutorials, etc.

The screenshot shows the Xilinx Documentation Navigator interface. The left side features a navigation tree with categories like Product Types, Silicon Devices, Accelerator Cards, System-on-Modules, Design Tools, ISE Design Suite, IP, Design Processes, and Document Types. The main area lists 256 documents out of 4830, with columns for TOC, Doc ID, Version, MB, Published, and Status. Several documents are marked as 'New!'.

Category	Document Type	Doc ID	Version	MB	Published	Status
Varium Cards	Product Pages	DS1003	0.1	0.1	New!	
	Data Sheets	DS1003	0.3	0.3	New!	
T1 Telco Cards	Product Pages	UG1525	0.6	0.6	New!	
	Data Sheets	UG1525	0.7	0.7	New!	
Vitis AI	User Guides	UG999	0.7	0.7	New!	
	Design Guides	XTP710	0.2	0.2	New!	
ISE Design Suite	User Guides	UG1518	10.7	10.7	New!	
	Design Guides	UG1495	9.9	9.9	New!	
IP	User Guides	UG1431	1.1	1.1	New!	
	Design Guides	UG1431	4.9	4.9	New!	
Design Processes	User Guides	UG1354	8.3	8.3	New!	
	Design Guides	UG1563	0.4	0.4	New!	
Document Types	User Guides	PG338	1.8	1.8	New!	
	Design Guides	PG367	0.5	0.5	New!	
Application Notes	User Guides	PG389	0.5	0.5	New!	
	Design Guides	UG1414	4.6	4.6	New!	
Architecture Manuals	User Guides	UG1563	0.5	0.5	New!	
	Design Guides	UG1563	0.5	0.5	New!	
Characterization Reports	User Guides	PG338	1.7	1.7	New!	
	Design Guides	PG403	0.5	0.5	New!	
Customer Notices	User Guides	PG367	0.5	0.5	New!	
	Design Guides	PG368	0.4	0.4	New!	
Data Sheets	User Guides	PG389	0.5	0.5	New!	
	Design Guides	PG389	0.5	0.5	New!	

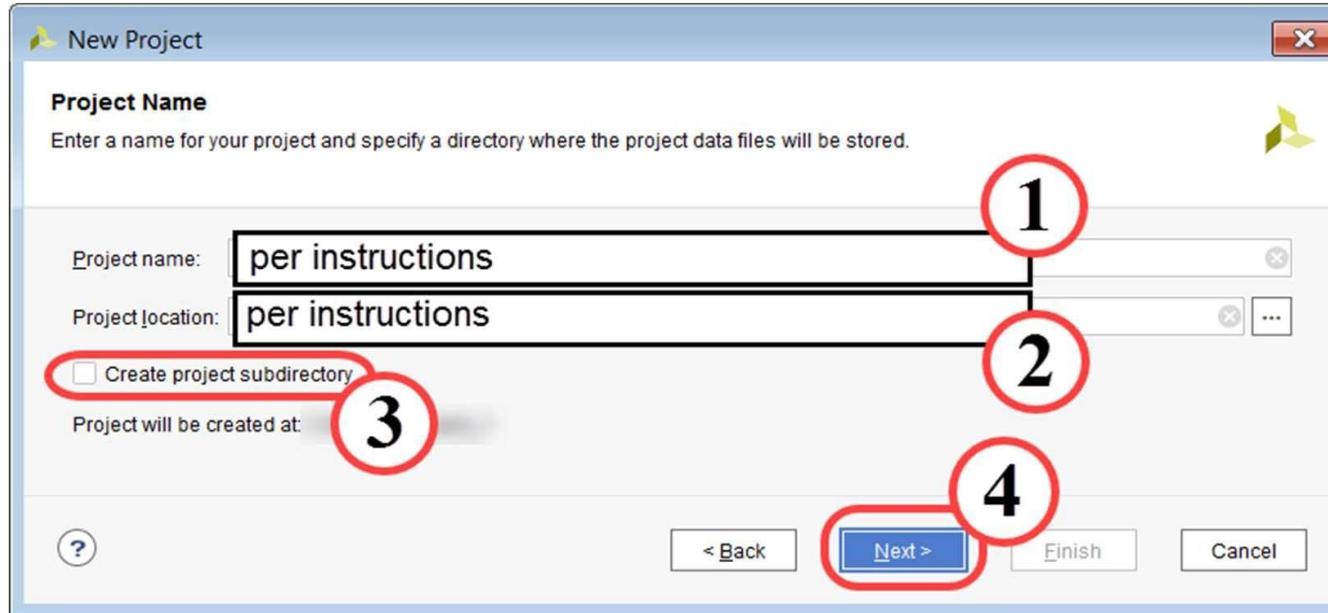
# New Project Creation Wizard

- Create Project



# New Project Creation Wizard

- Project Name

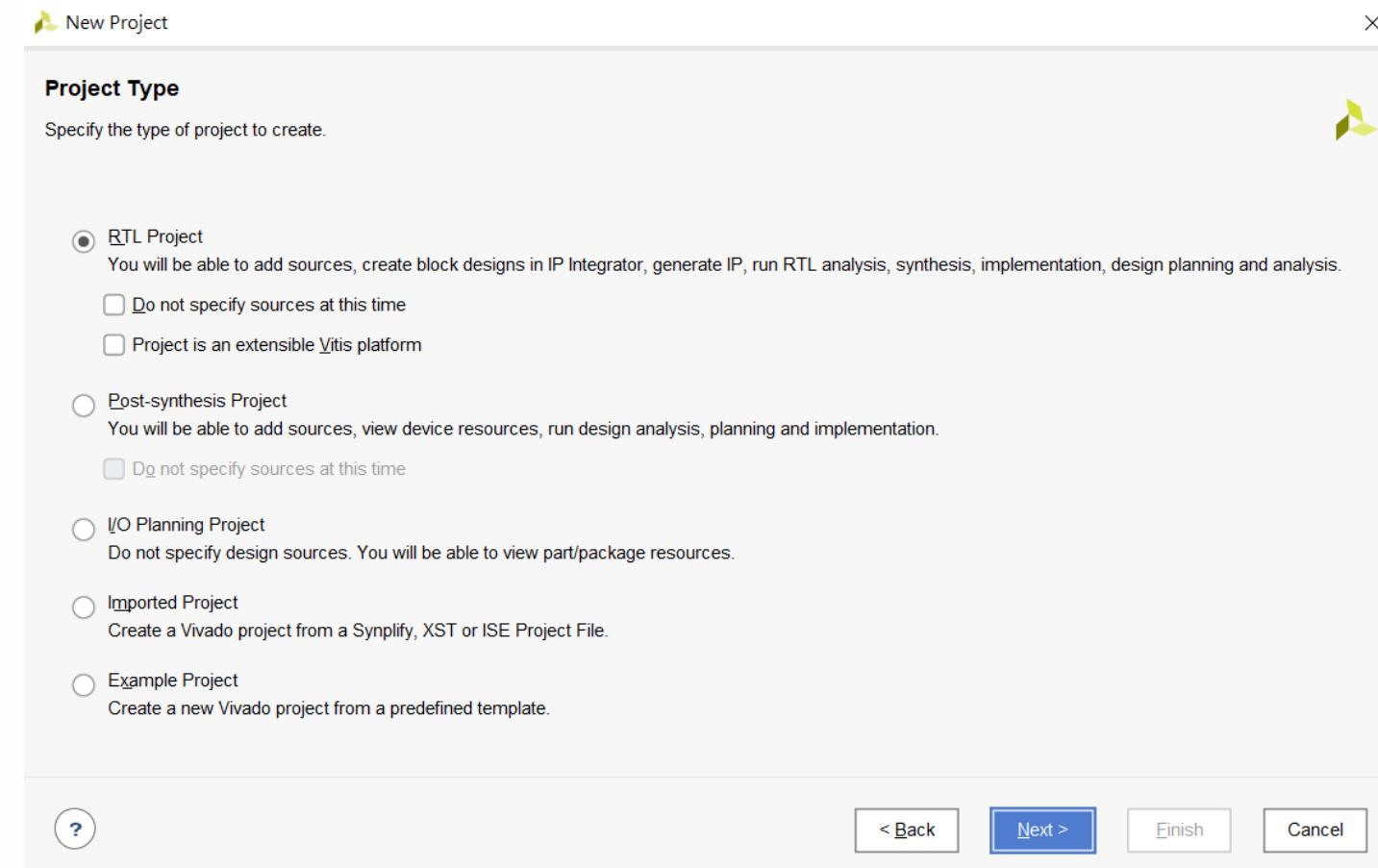


- ① Enter Project name
- ② Enter location in the Project location field
- ③ Select this option will create the folder with Project name, if you do not want to create a new folder, deselect it
- ④ Click "Next" to continue the project creation

# New Project Creation Wizard

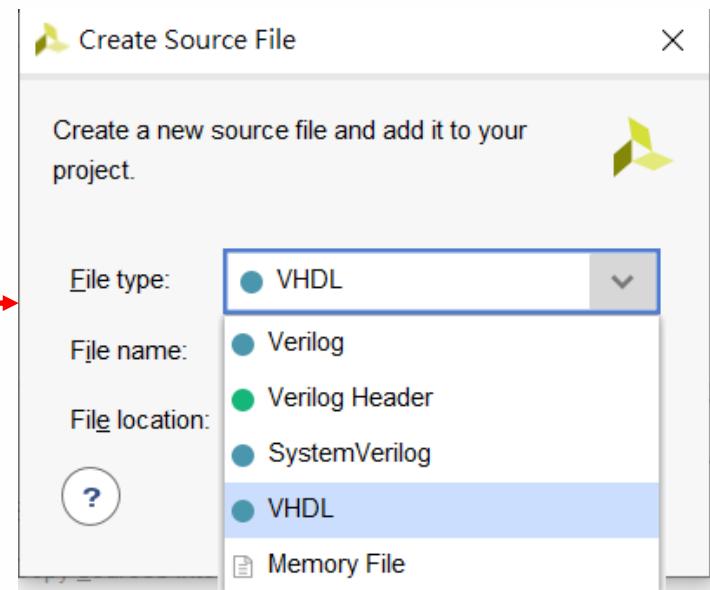
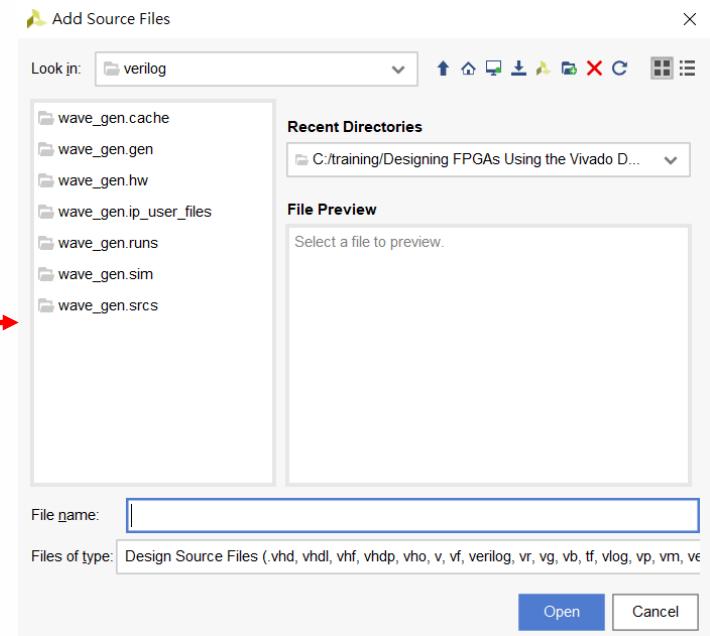
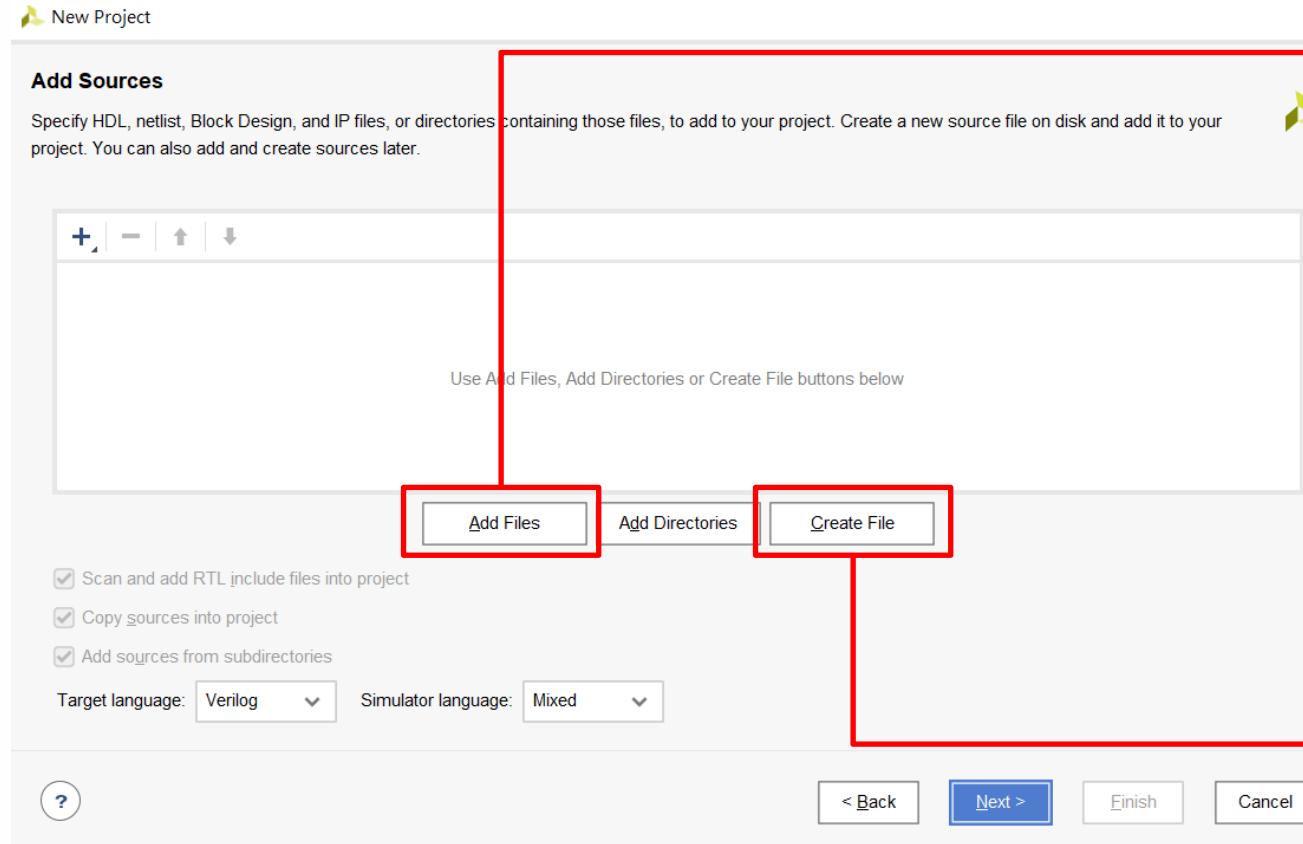
- Project Type

- > Five different types of project can be created
  - **RTL**
    - Using HDL source files
  - **Post-synthesis**
    - EDIF or NGC
    - Design checkpoint
  - **I/O planning**
    - For early pin testing
    - No design sources
  - **Imported project** from Project Navigator, XST, and Synplify projects
  - **Example project**
    - Using available design templates



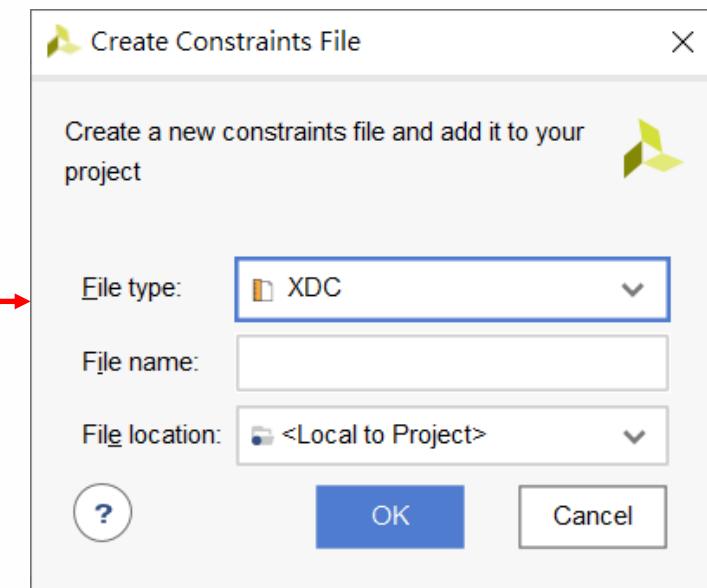
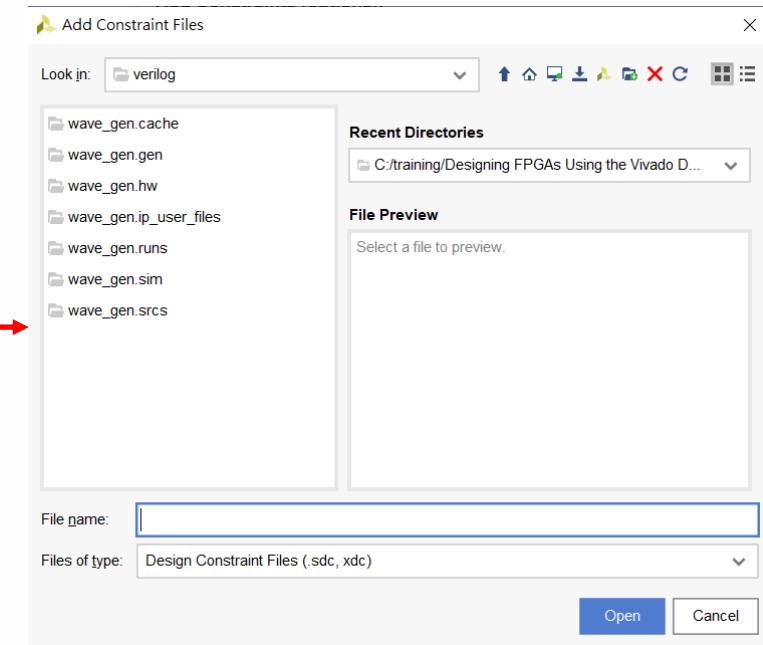
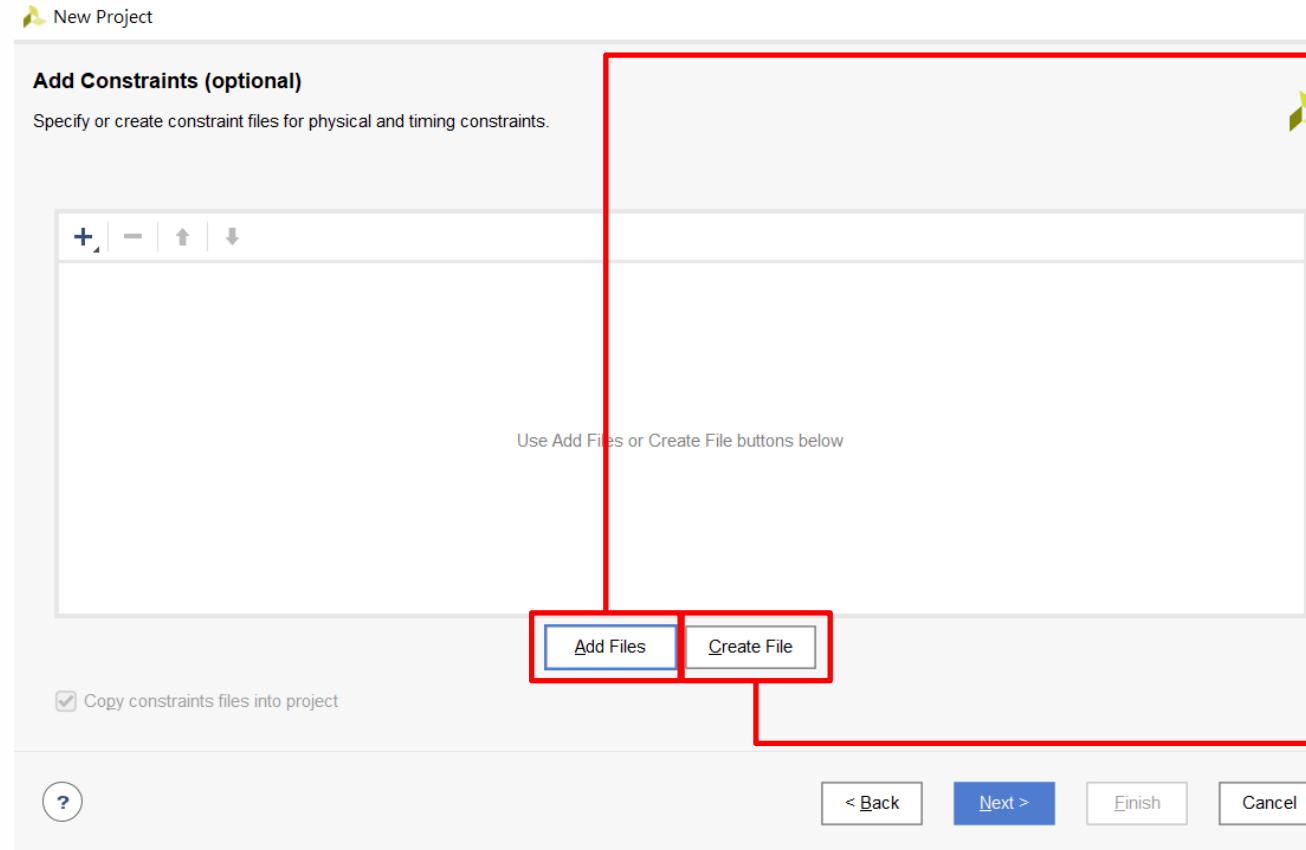
# New Project Creation Wizard

- Add Sources



# New Project Creation Wizard

- Add Constraints



# New Project Creation Wizard

- Choose a Xilinx Part or a Board

New Project

### Default Part

Choose a default Xilinx part or board for your project.

**Parts**  **Boards**

Reset All Filters

Category:	All	Package:	All	Temperature:	All				
Family:	All	Speed:	All	Static power:	All				
Search:	Q:								
Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Transceivers	GTPE2
xc7vx485tffg1157-3	1157	600	303600	607200	1030	0	2800	20	0
xc7vx485tffg1157-2L	1157	600	303600	607200	1030	0	2800	20	0
xc7vx485tffg1157-1	1157	600	303600	607200	1030	0	2800	20	0
xc7vx485tffg1158-3	1158	350	303600	607200	1030	0	2800	48	0
xc7vx485tffg1158-2	1158	350	303600	607200	1030	0	2800	48	0
xc7vx485tffg1158-2L	1158	350	303600	607200	1030	0	2800	48	0
xc7vx485tffg1158-1	1158	350	303600	607200	1030	0	2800	48	0

< Back

New Project

**Boards**  **Parts**

To fetch the latest available boards from git repository, click on 'Refresh' button. Dismiss

Reset All Filters

Vendor:	Name:	Board Rev:
All	All	Latest

Search: Q:

Display Name	Preview	Status	Vendor	File Version	Part	I/O Pin Count	B
Artix-7 AC701 Evaluation Platform Add Companion Card <a href="#">Connections</a>		Installed	xilinx.com	1.4	xc7a200tfgb676-2	676	1
Kria K26C SOM Add Companion Card <a href="#">Connections</a>		Installed	xilinx.com	1.3	Som Vision Platform Board	784	R

# New Project Creation Wizard

- Finish the project creation

The screenshot shows the 'New Project Summary' step of the Vivado New Project Creation Wizard. The window title is 'New Project'. The main content area displays the 'VIVADO ML Editions' logo and the 'New Project Summary' heading. It contains several informational messages and settings:

- A new RTL project named 'project\_1' will be created.
- No source files or directories will be added. Use Add Sources to add them later.
- No constraints files will be added. Use Add Sources to add them later.
- The default part and product family for the new project:
  - Default Part: xc7vx485tffg1157-1
  - Family: Virtex-7
  - Package: ffg1157
  - Speed Grade: -1

To create the project, click Finish

At the bottom, there are navigation buttons: '?', '< Back', 'Next >', 'Finish' (highlighted in blue), and 'Cancel'.

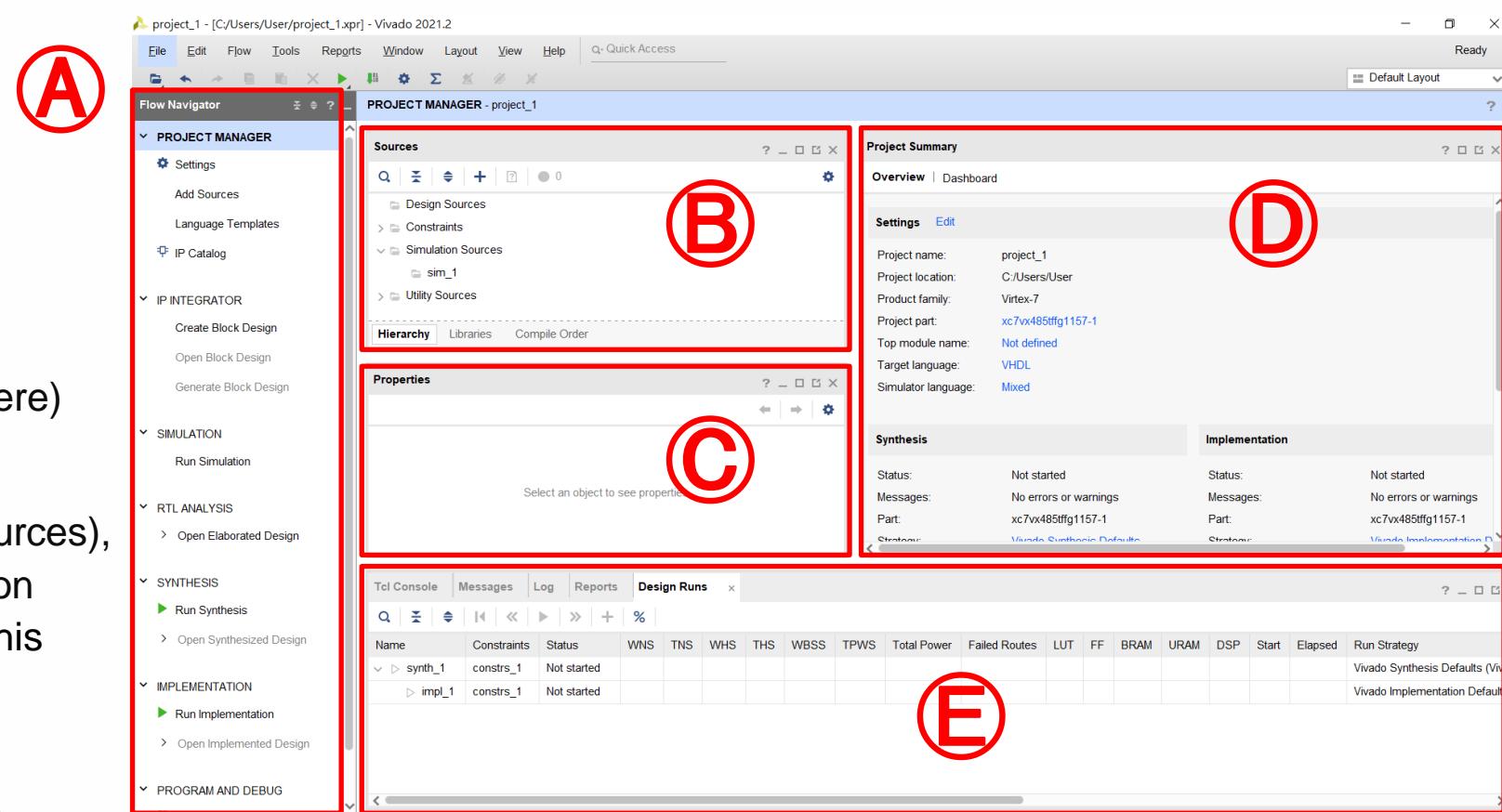
# Integrated Design Environment (IDE)

- > A: Flow Navigator
- > B: Sources view
  - Hierarchical display of sources
  - IP Sources and Libraries view
  - Gives access to constraints file
- > C: Block Properties / Attributes
- > D: Workspace (Project Summary shown here)

## Project Summary

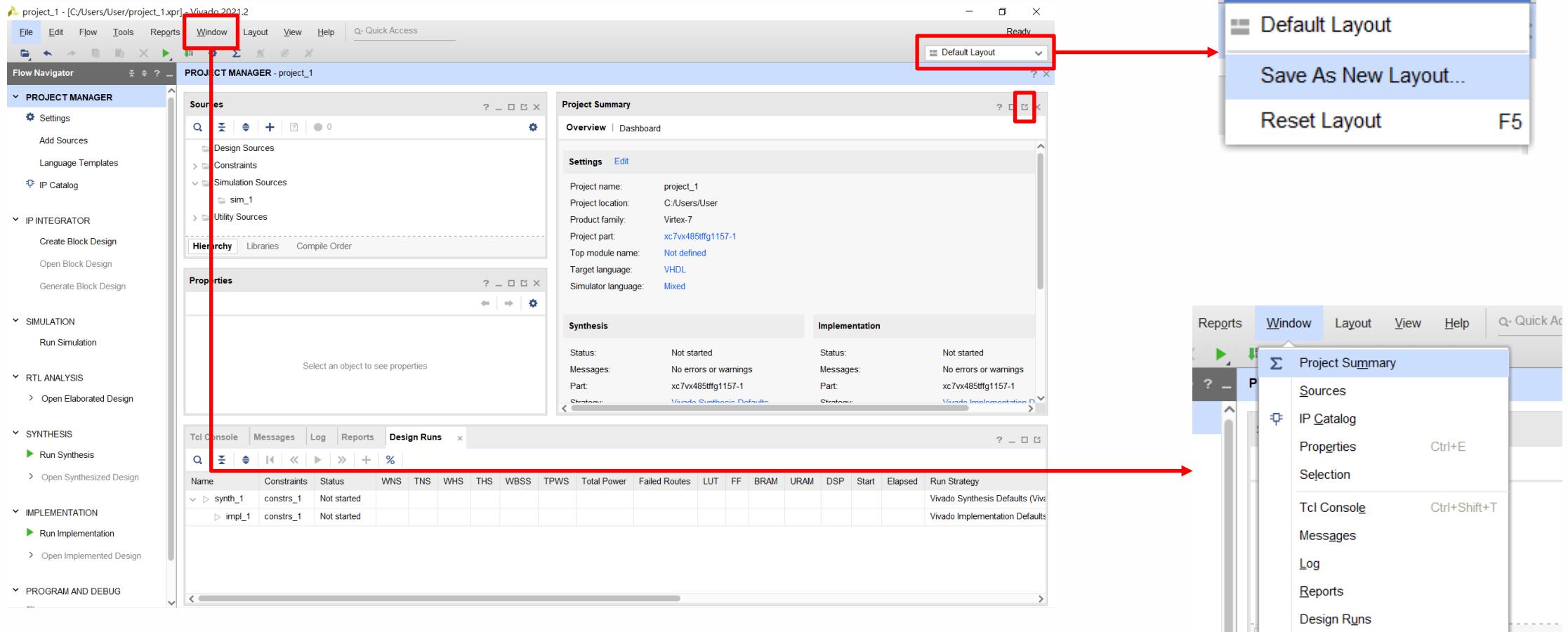
- Gives access to device utilization (resources), timing summary, and strategy information
- Enable you to view and analyze all of this information in one place

- > E: Console
  - Tcl console, Messages, Log and Reports.



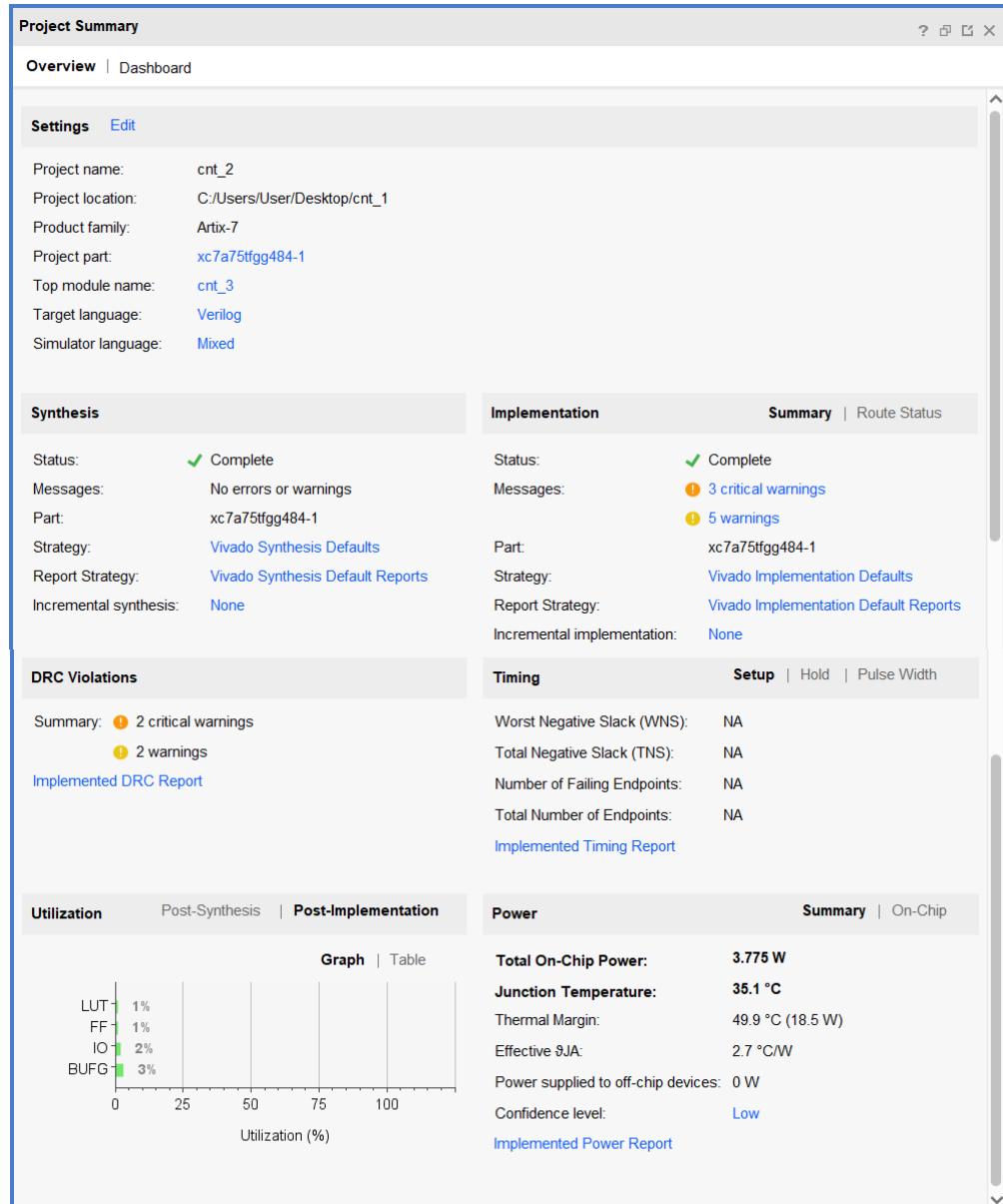
# Integrated Design Environment (IDE)

- Reset layout or Restore window



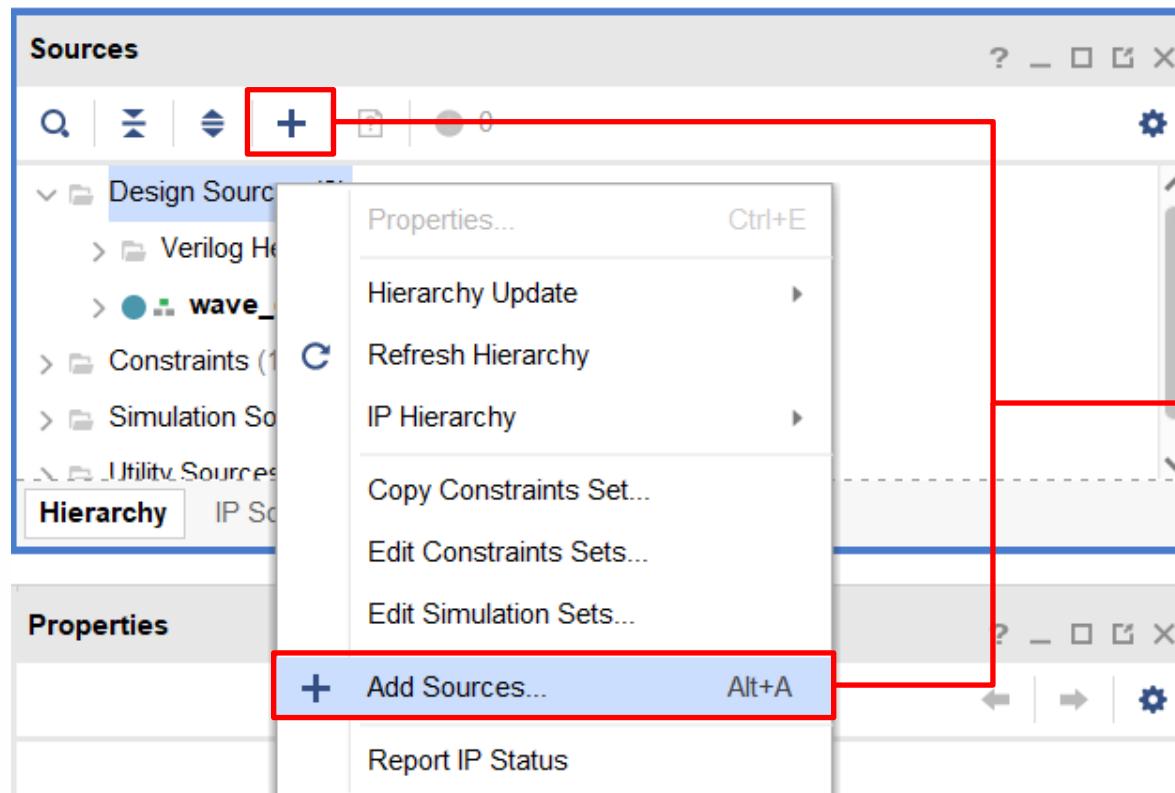
# Integrated Design Environment (IDE)

- Project Summary
  - Shows the status of the current project design
  - Includes status of Synthesis, status of Implementation, DRC Violations, Timing, Utilization, Power consumption and Junction Temperature



# Integrated Design Environment (IDE)

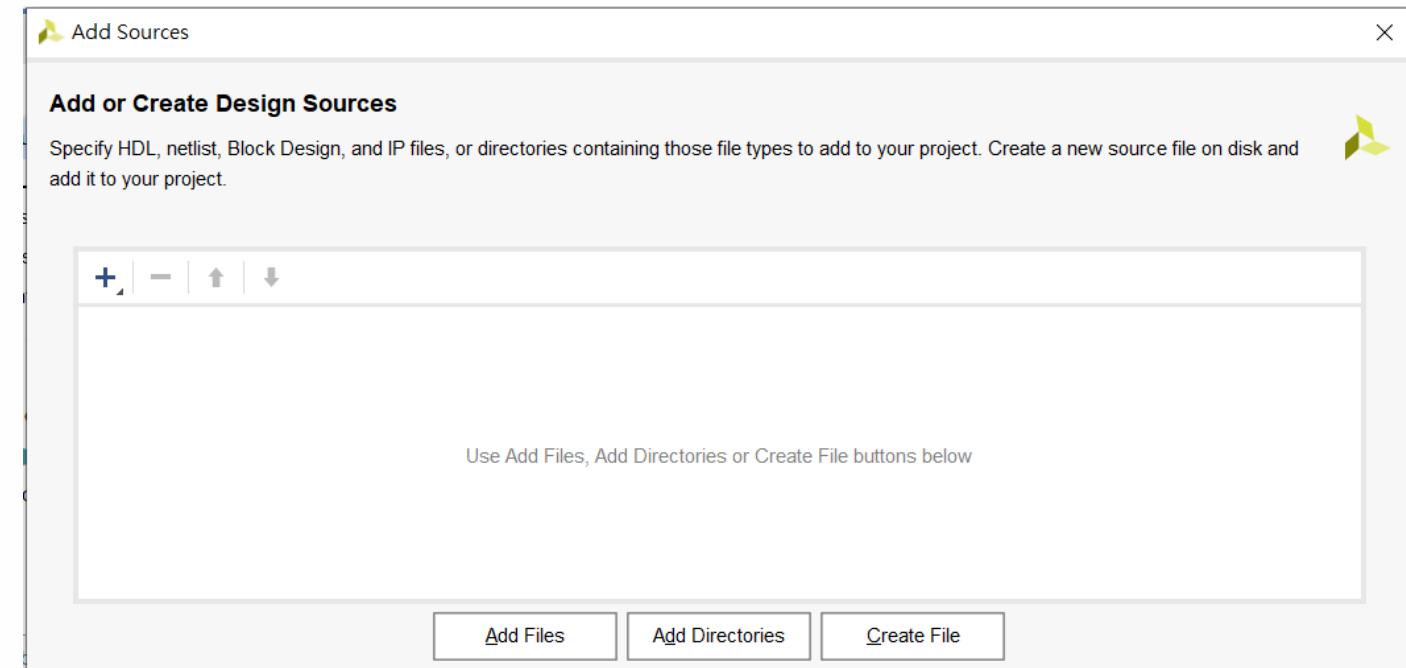
- Create Design Source file



# Integrated Design Environment (IDE)

- Create Design Source file

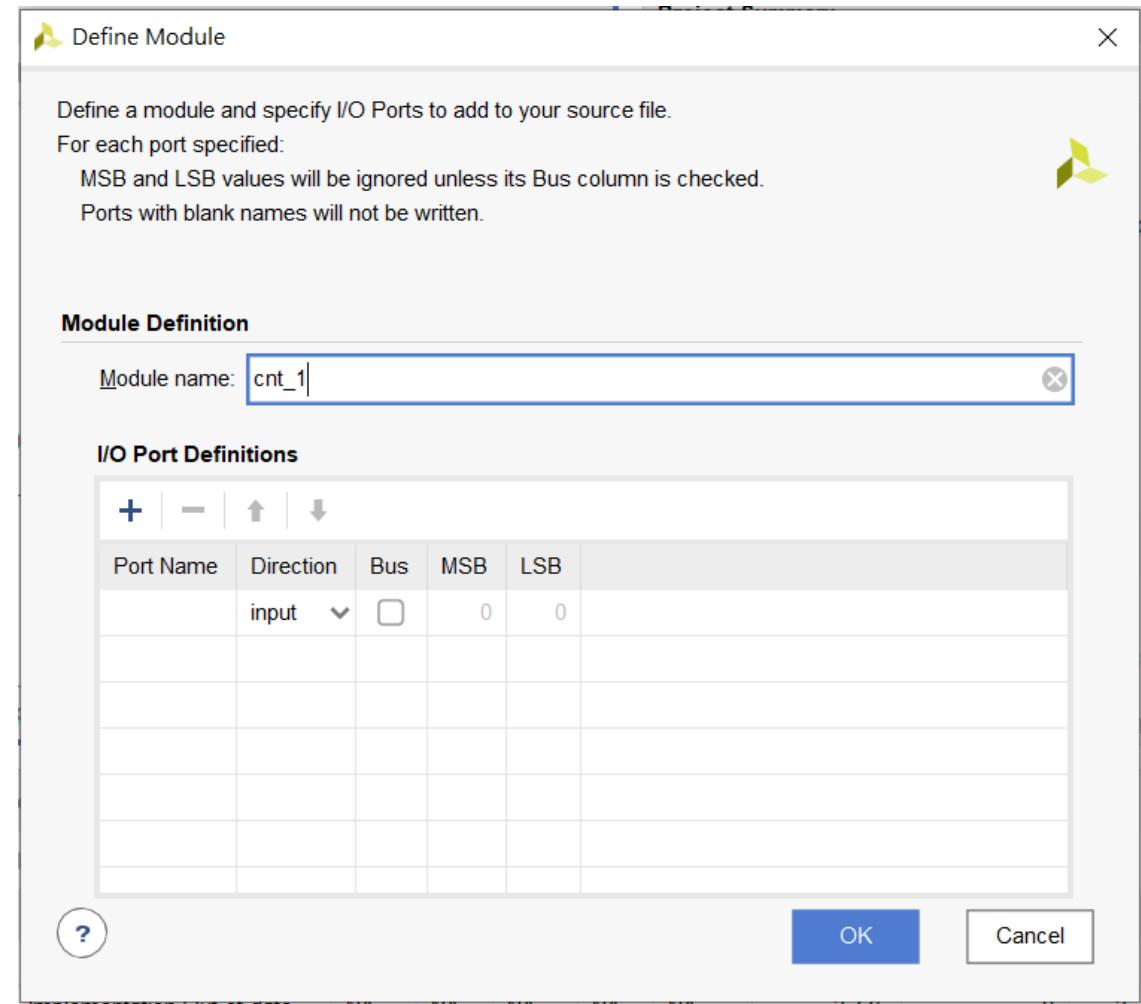
After selecting “Add or create design sources”, you can choose add existing design source files or create a new one.



# Integrated Design Environment (IDE)

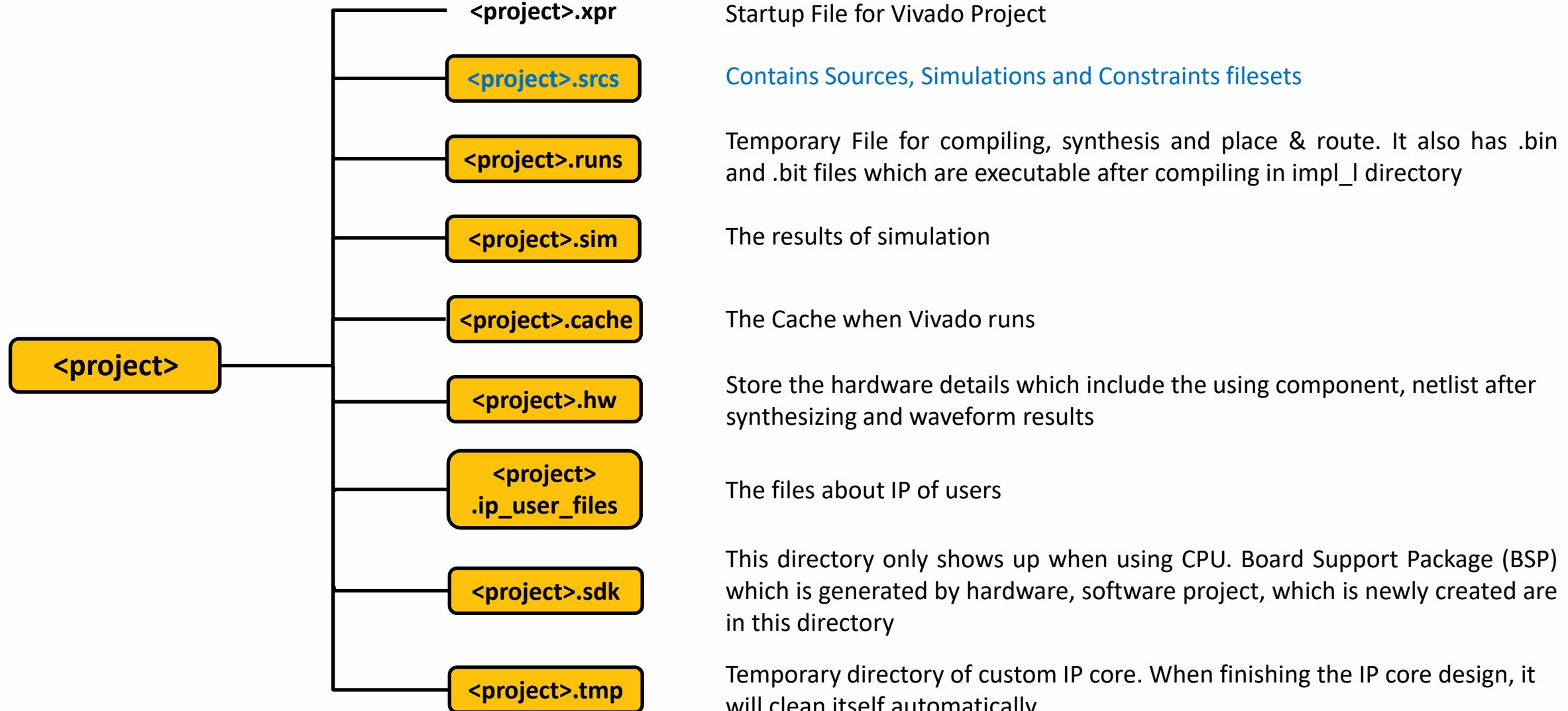
- Create Design Source file

If you create a new design source from scratch, then “Define Module” window pops out. In this window, you will be able to pre-define module.



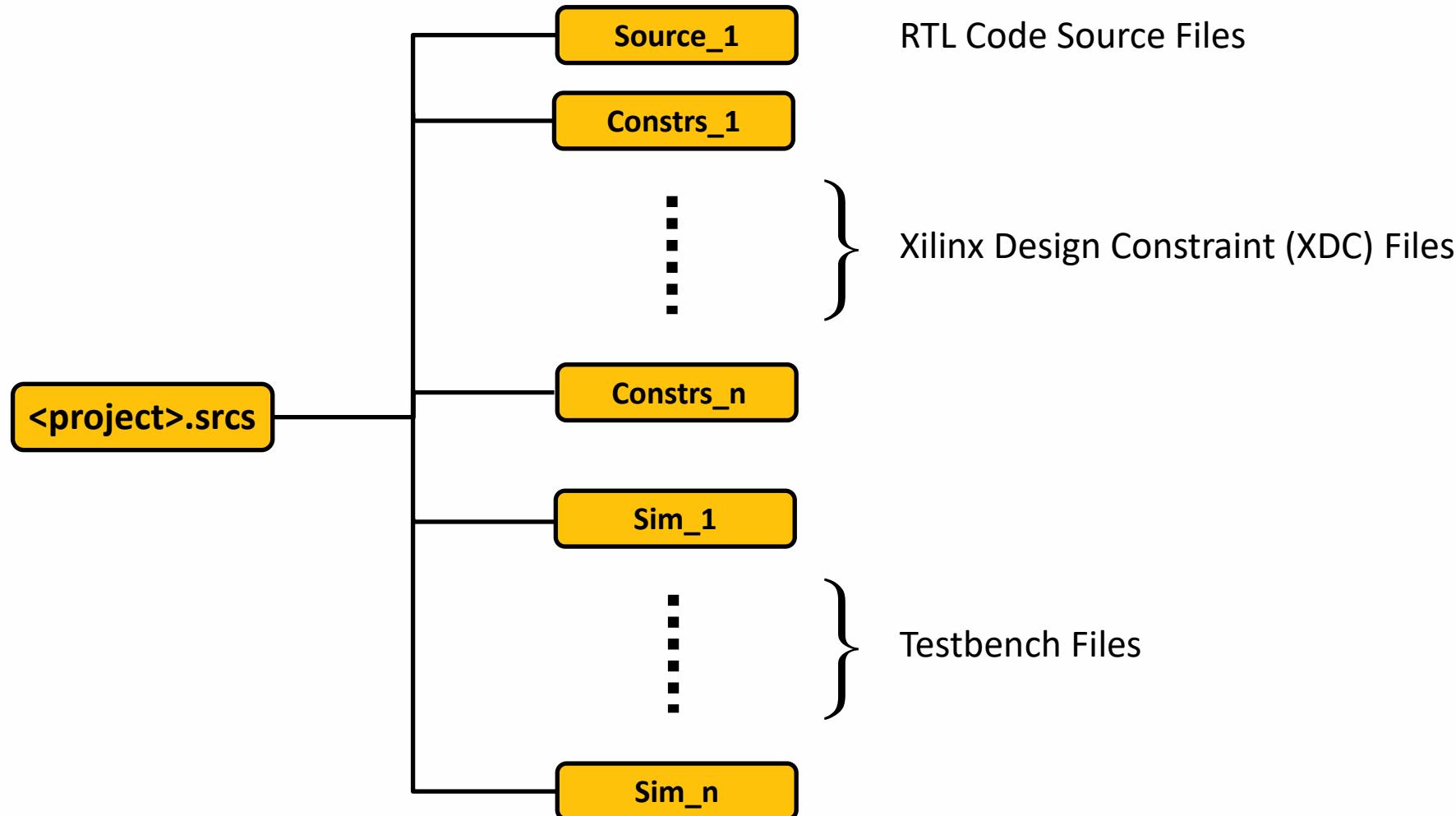
# Integrated Design Environment (IDE)

- Project Filesets



# Integrated Design Environment (IDE)

- Project Filesets



# Behavioral Simulation

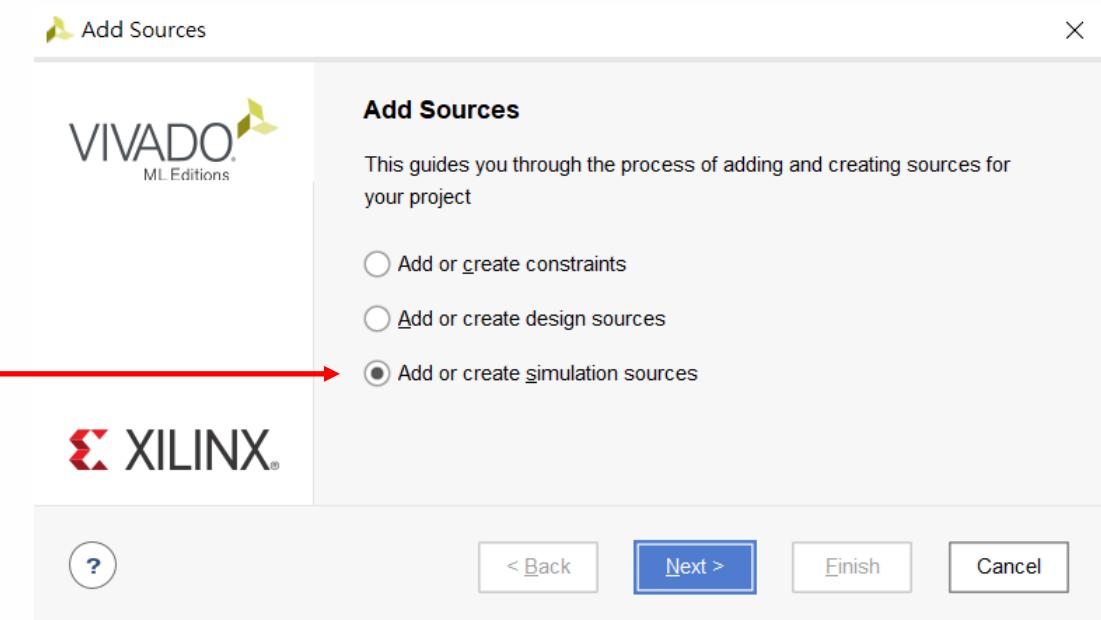
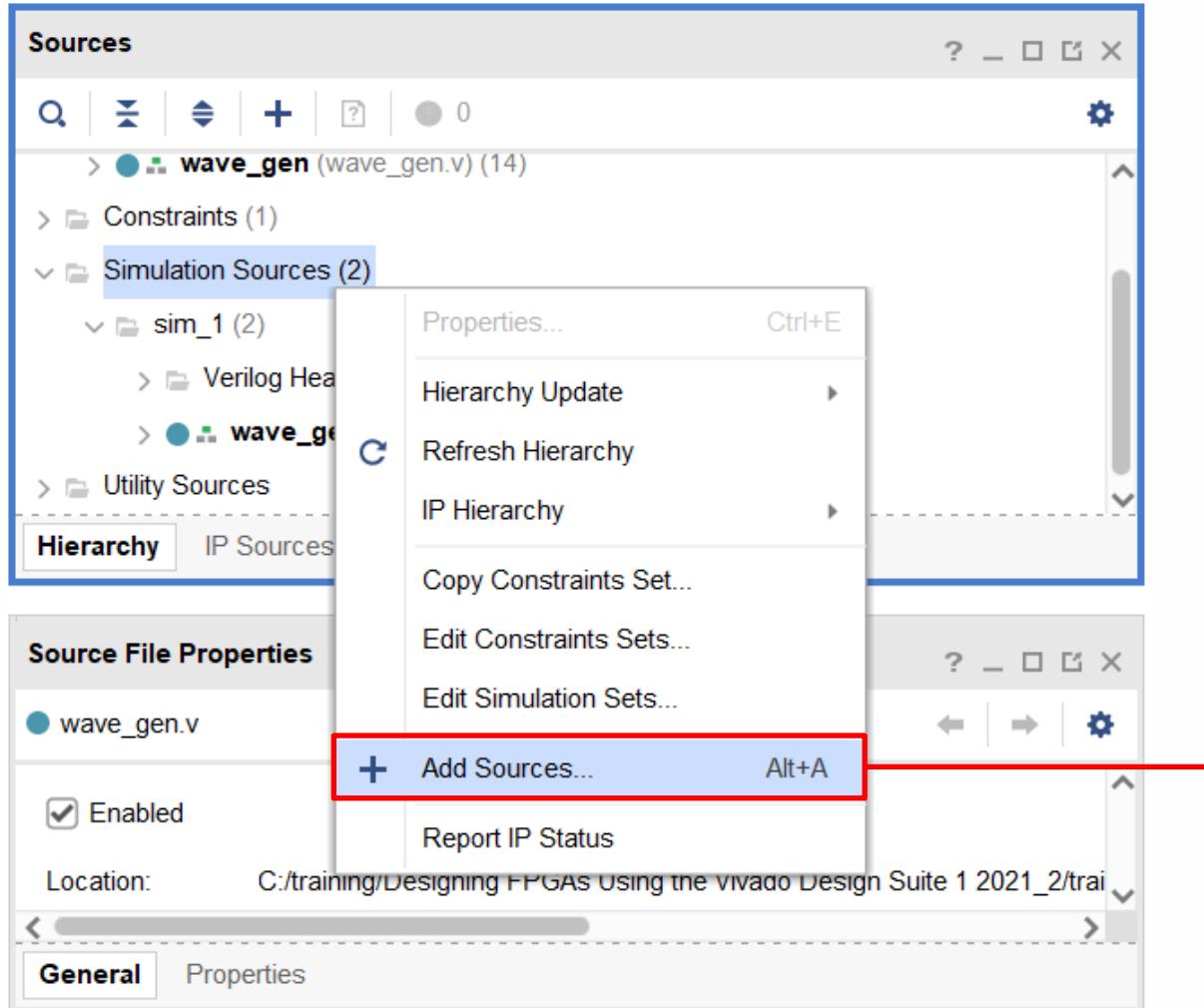


# Behavioral Simulation

- Testbench
  - 1. Internal Signals
  - 2. Unit under Test Instantiation
  - 3. Input Stimulus including Clock Signals
  - 4. Output Monitoring
- > Verifies syntax and functionality of the design without timing information

# Behavioral Simulation

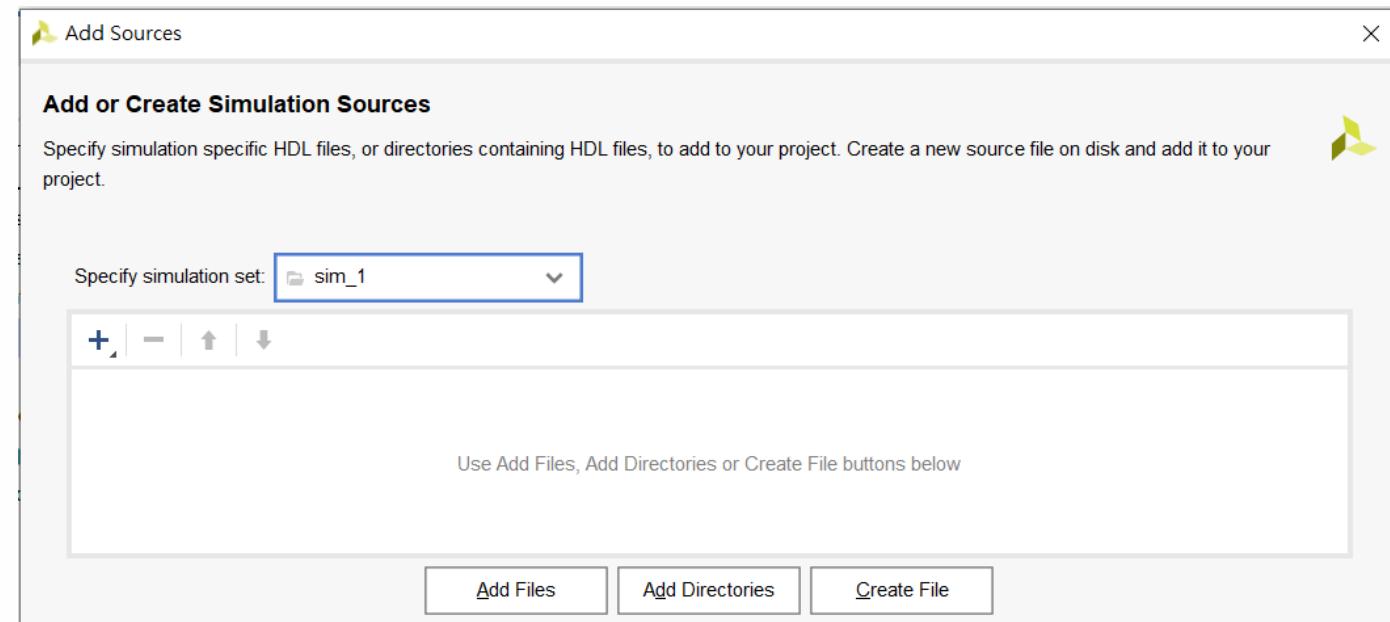
- Add Simulation Sources



# Behavioral Simulation

- Add Simulation Sources

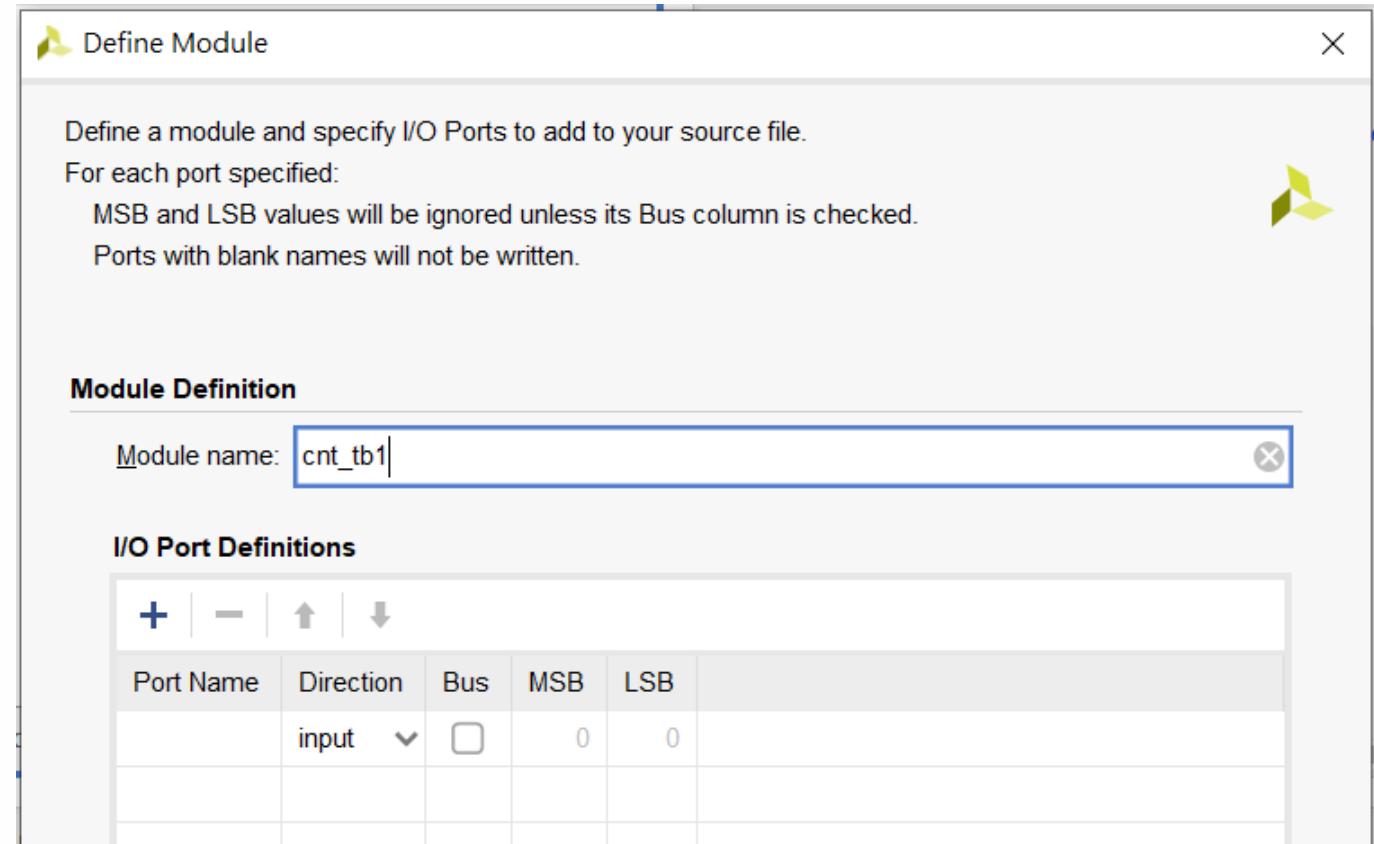
After selecting “Add or create simulation sources”, you can choose add an existing simulation source files or create a new one



# Behavioral Simulation

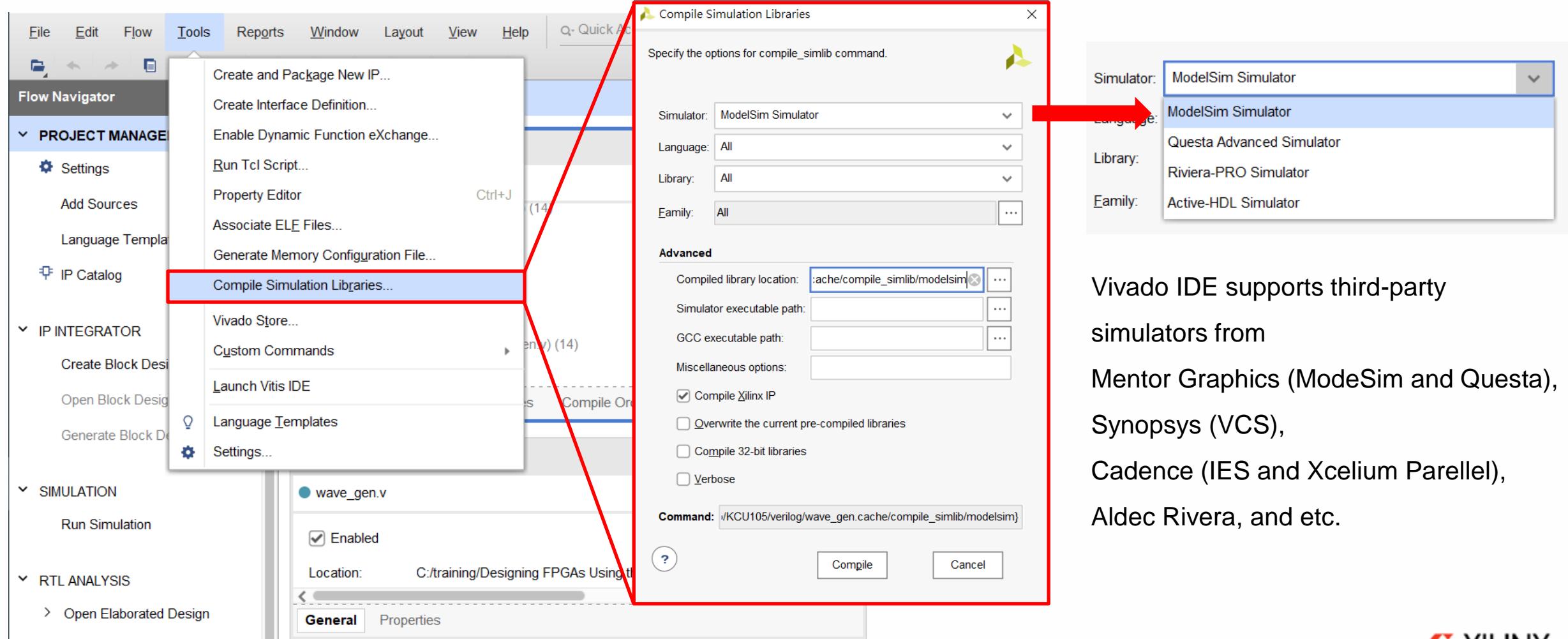
- Add Simulation Sources

If you create a new simulation source from scratch, then “Define Module” window pops out. In this window, you will be able to pre-define testbench module.



# Behavioral Simulation

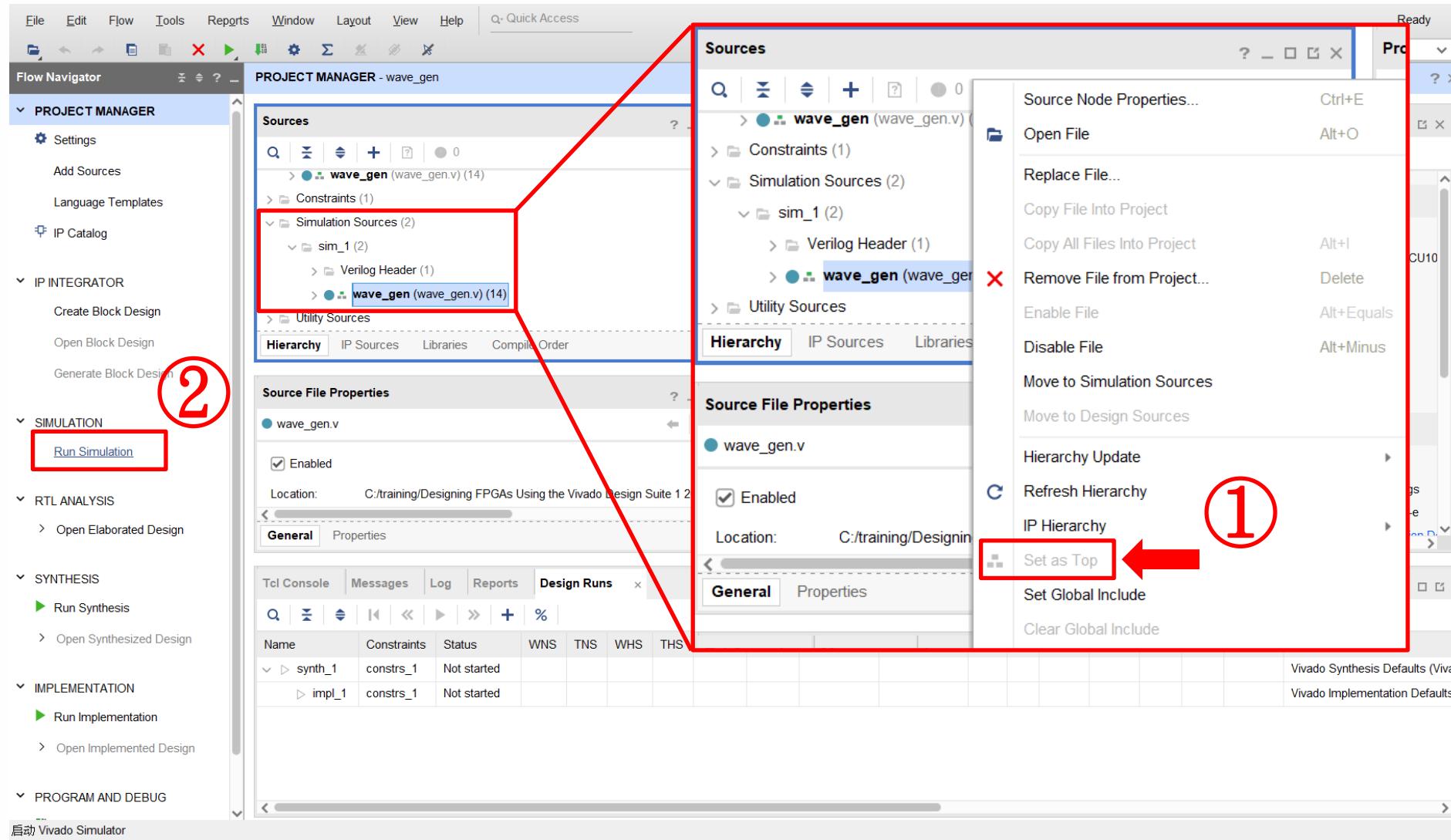
- 3-rd Party Simulation Libraries
  - > Some IPs and Devices have Simulation Libraries that have already been built



Vivado IDE supports third-party simulators from Mentor Graphics (ModeSim and Questa), Synopsys (VCS), Cadence (IES and Xcelium Parallel), Aldec Rivera, and etc.

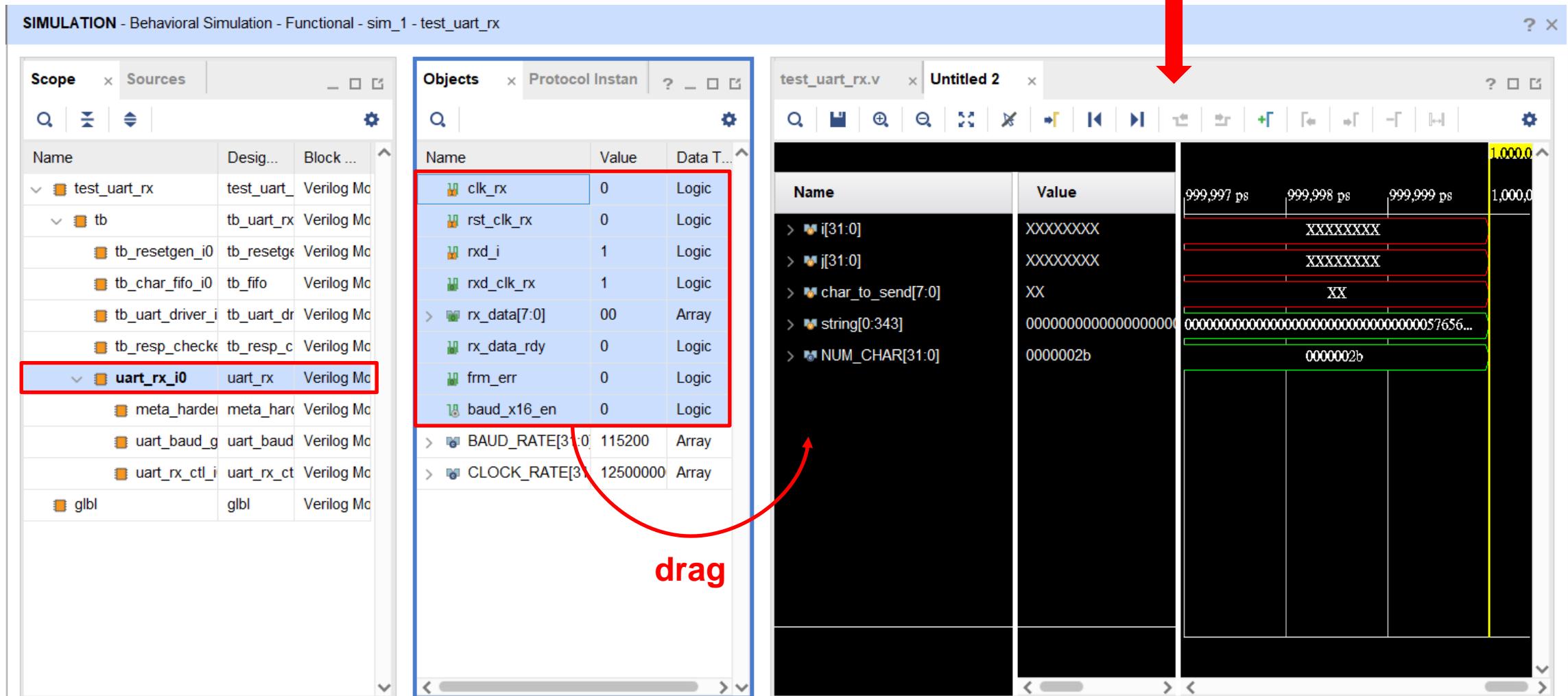
# Behavioral Simulation

- Run Simulation



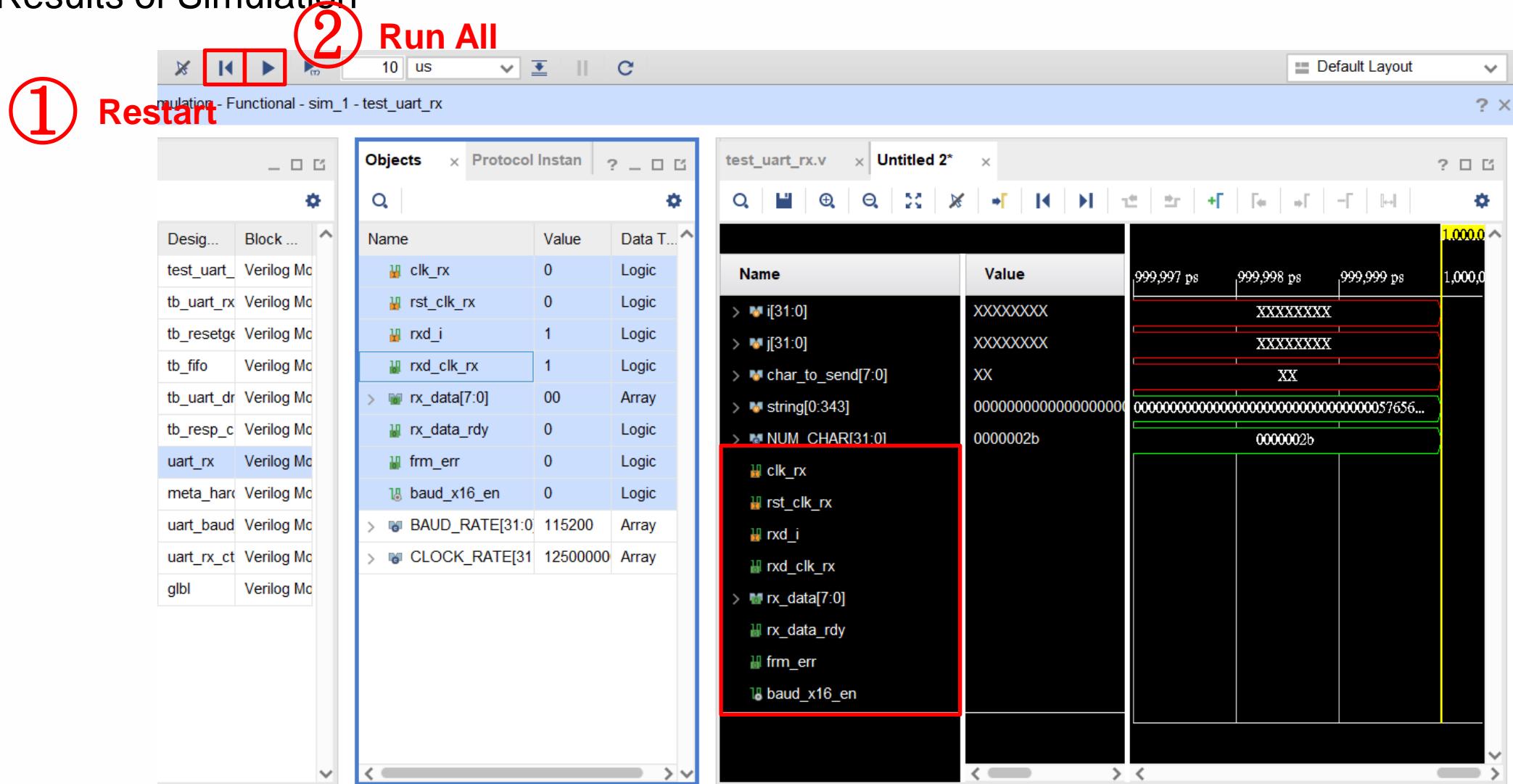
# Behavioral Simulation

- Results of Simulation



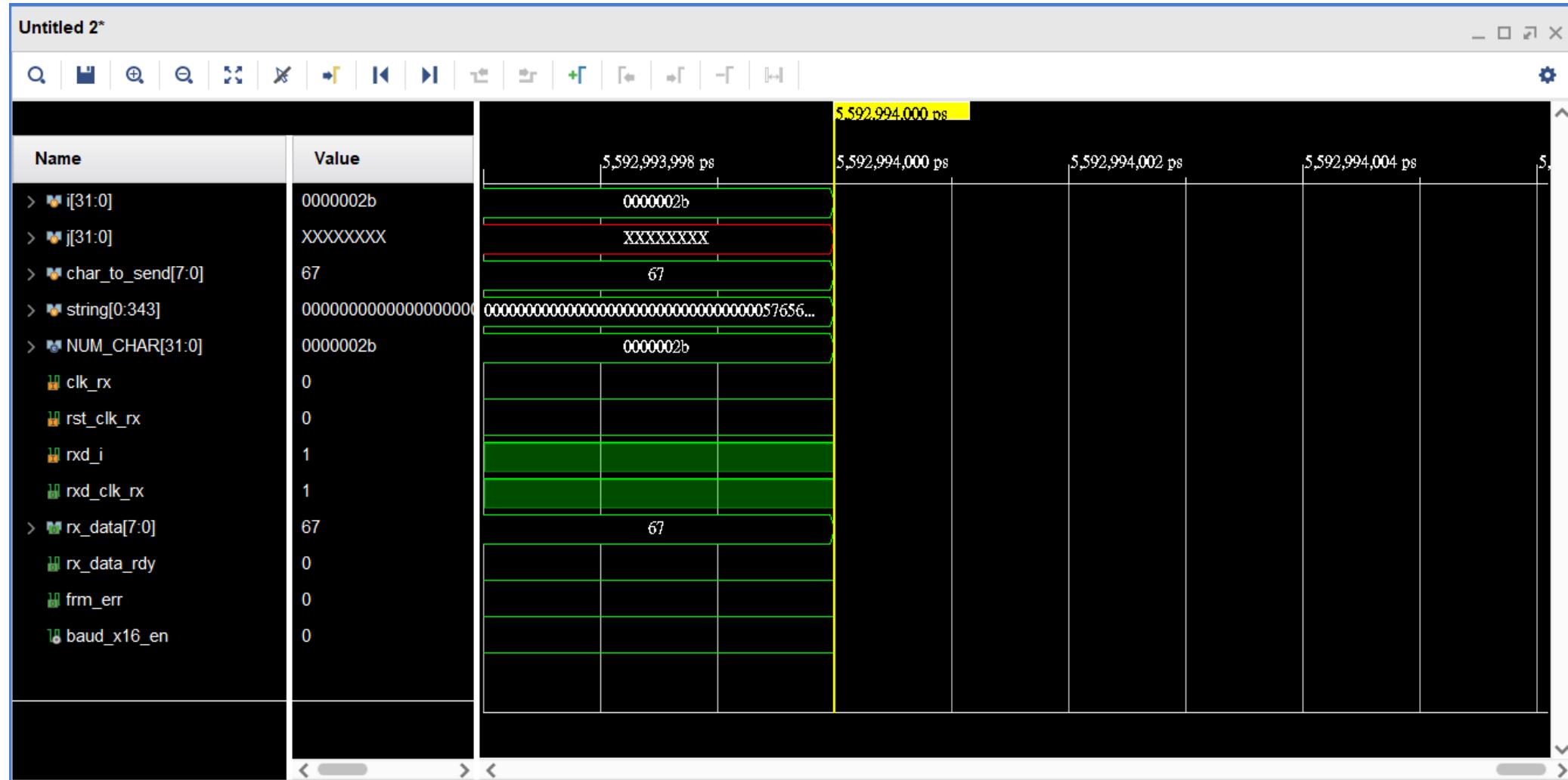
# Behavioral Simulation

- Results of Simulation



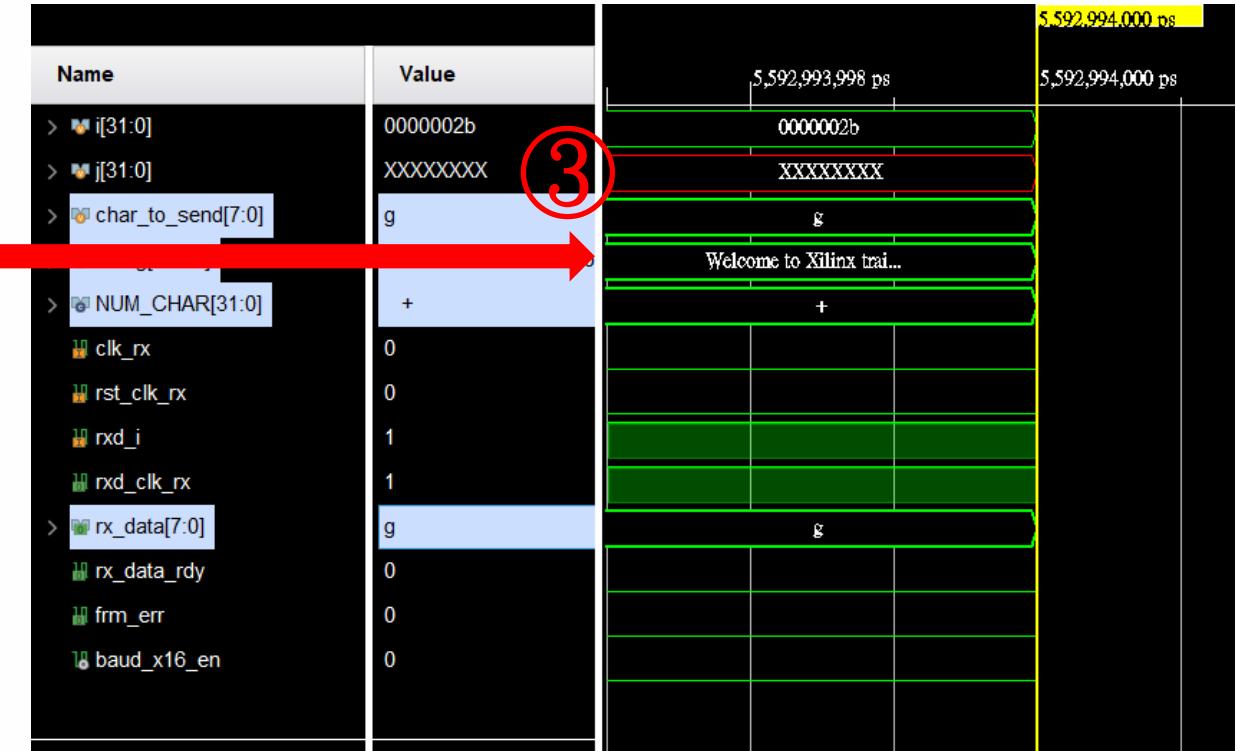
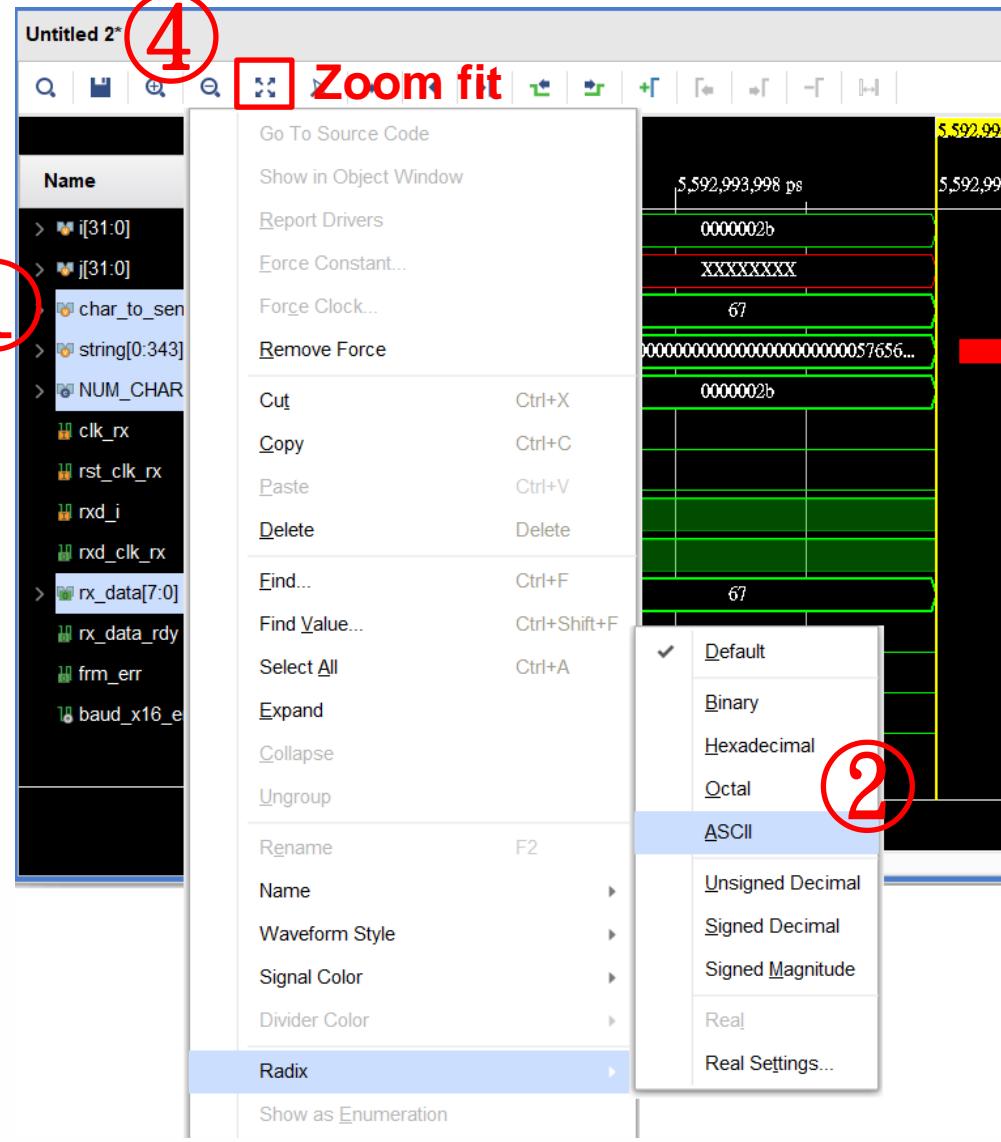
# Behavioral Simulation

- Results of Simulation



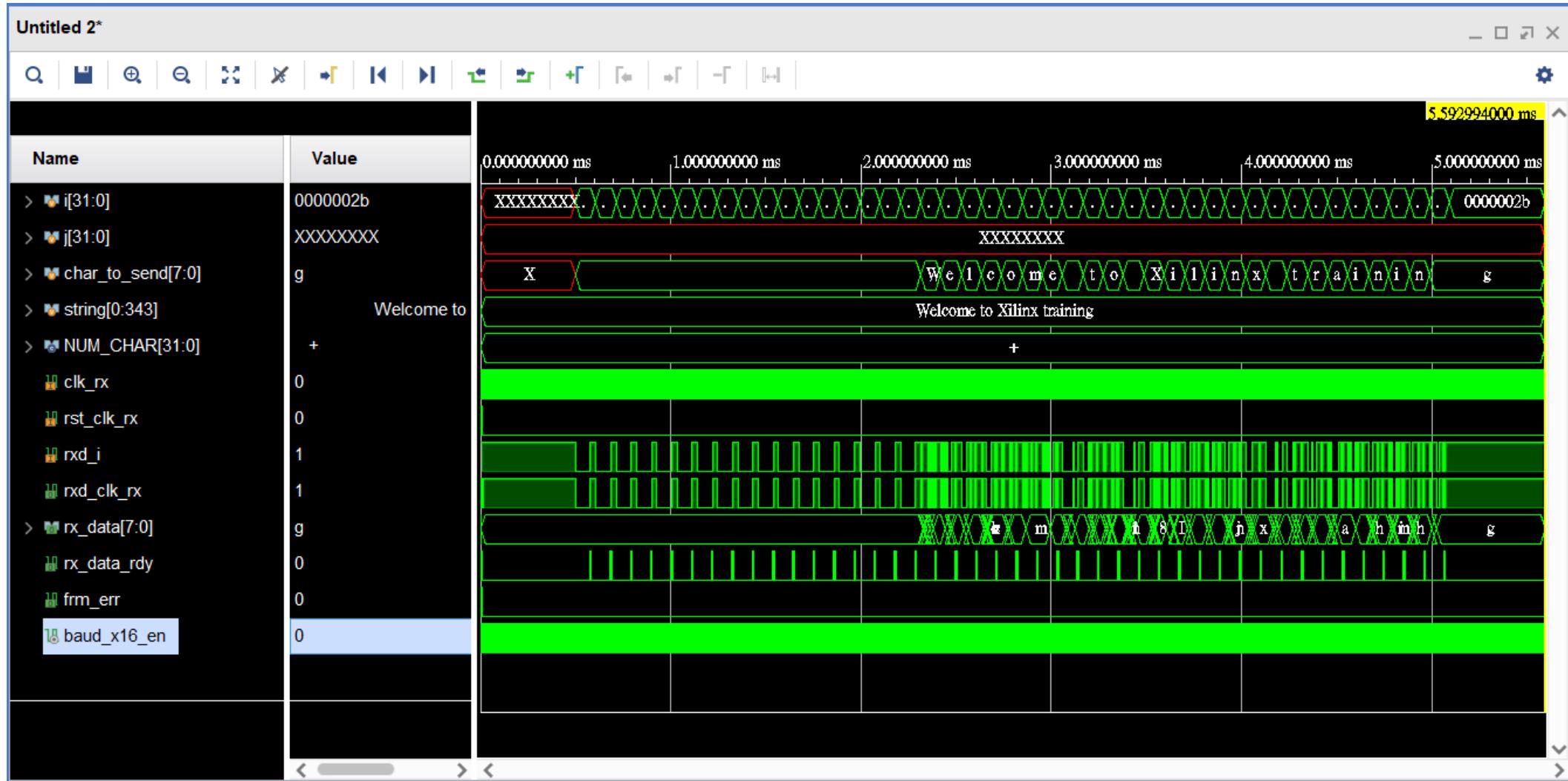
# Behavioral Simulation

- Results of Simulation



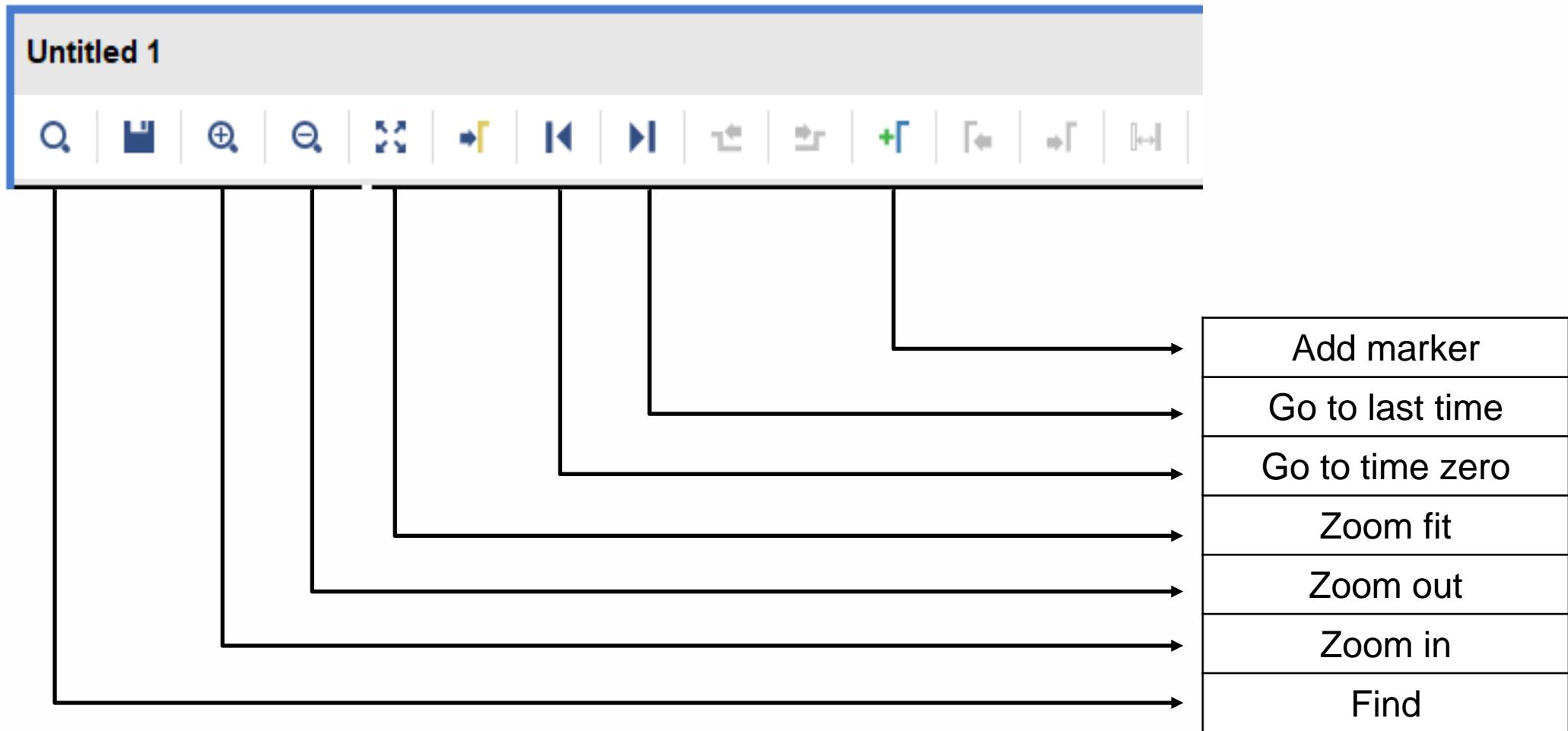
# Behavioral Simulation

- Results of Simulation



# Behavioral Simulation

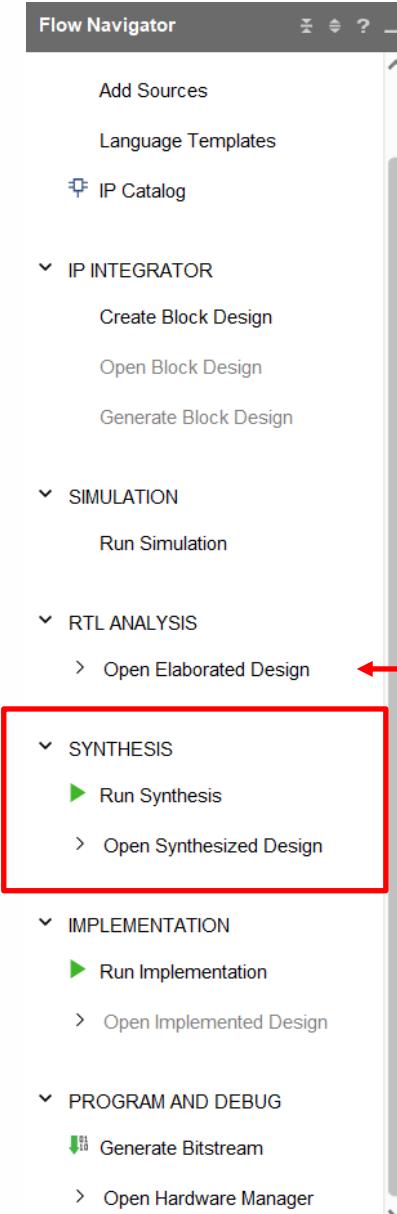
- Waveform toolbar



# Vivado Synthesis



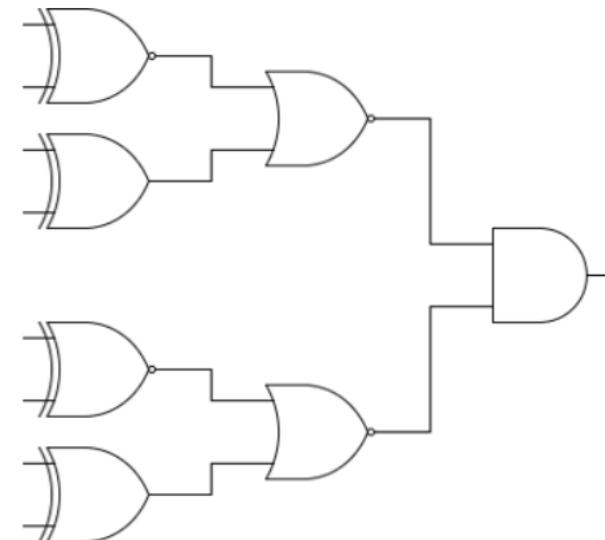
# Vivado Synthesis



- **Synthesis**

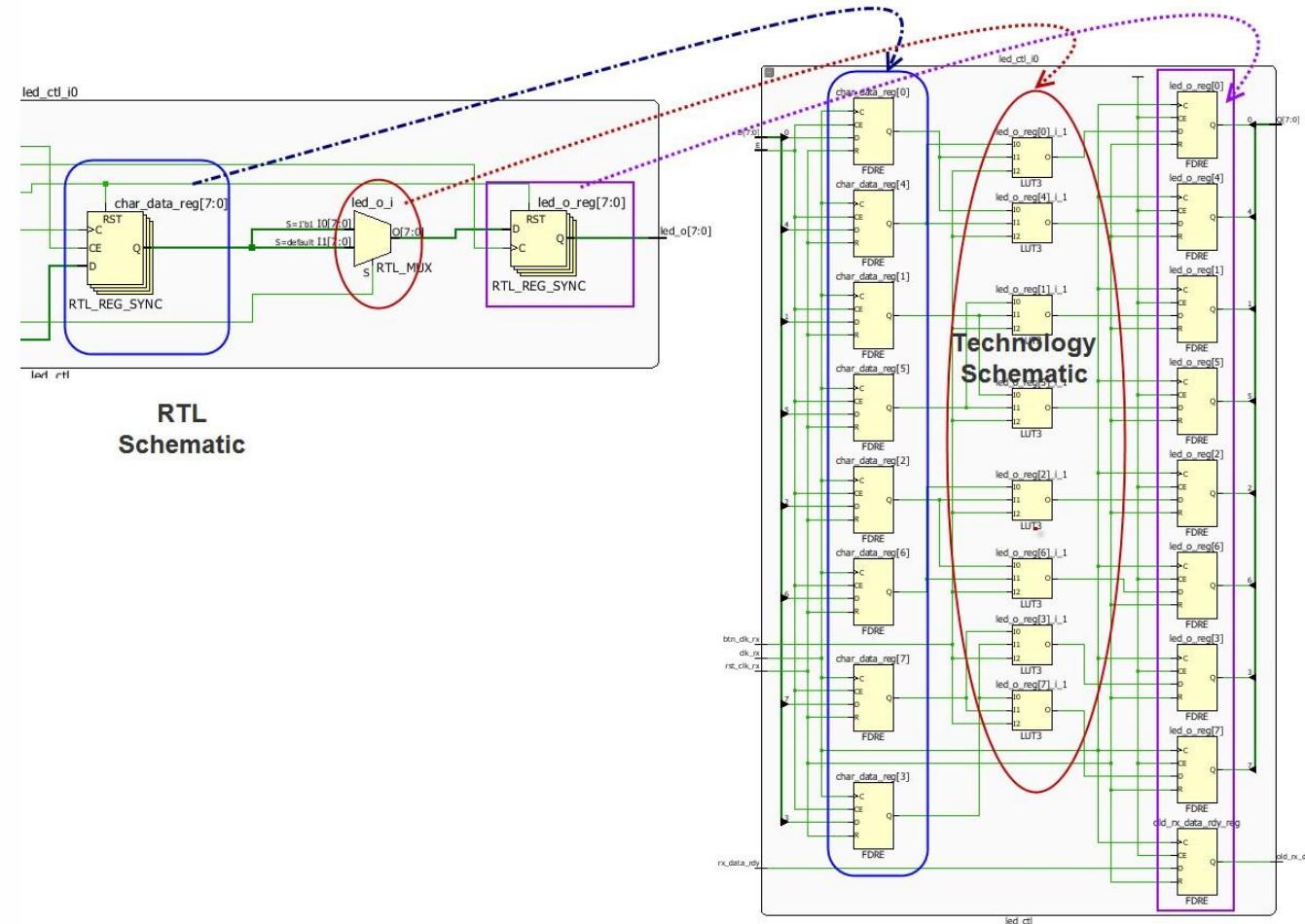
1. Elaborate the Design → Convert RTL code to Netlist
2. Apply Constraints to the Design
3. Do high level optimizations of the Design
4. Technology map the Design
5. Do low level optimizations of the Design

Netlist



# Vivado Synthesis

- > Transformation of an RTL design into a gate-level representation of LUTs, flip-flops, block RAMs, etc.
- > Timing driven and optimized for performance
- > Verilog, VHDL, mixed HDL, and SystemVerilog support



# Vivado Synthesis

- Synthesis Settings

- SYNTHESIS

- Run Synthesis

- Open Synthesiz

- Synthesis Settings...

- Launch Synthesis Run...

- Reset Synthesis Run

- Launch Next Step

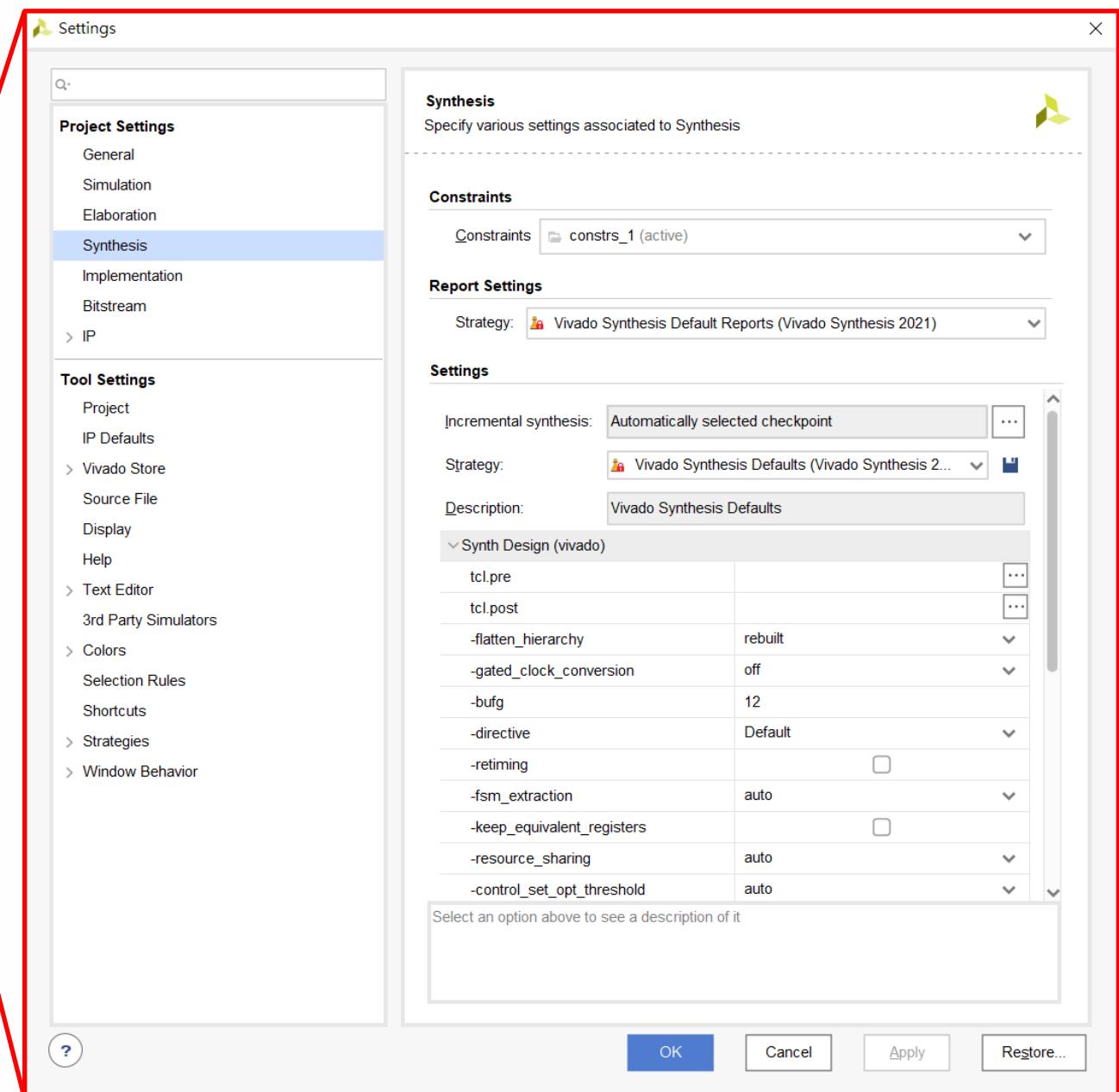
- Reset to Previous Step: synth\_design

- Create Synthesis Runs...

- IMPLEMENTATION

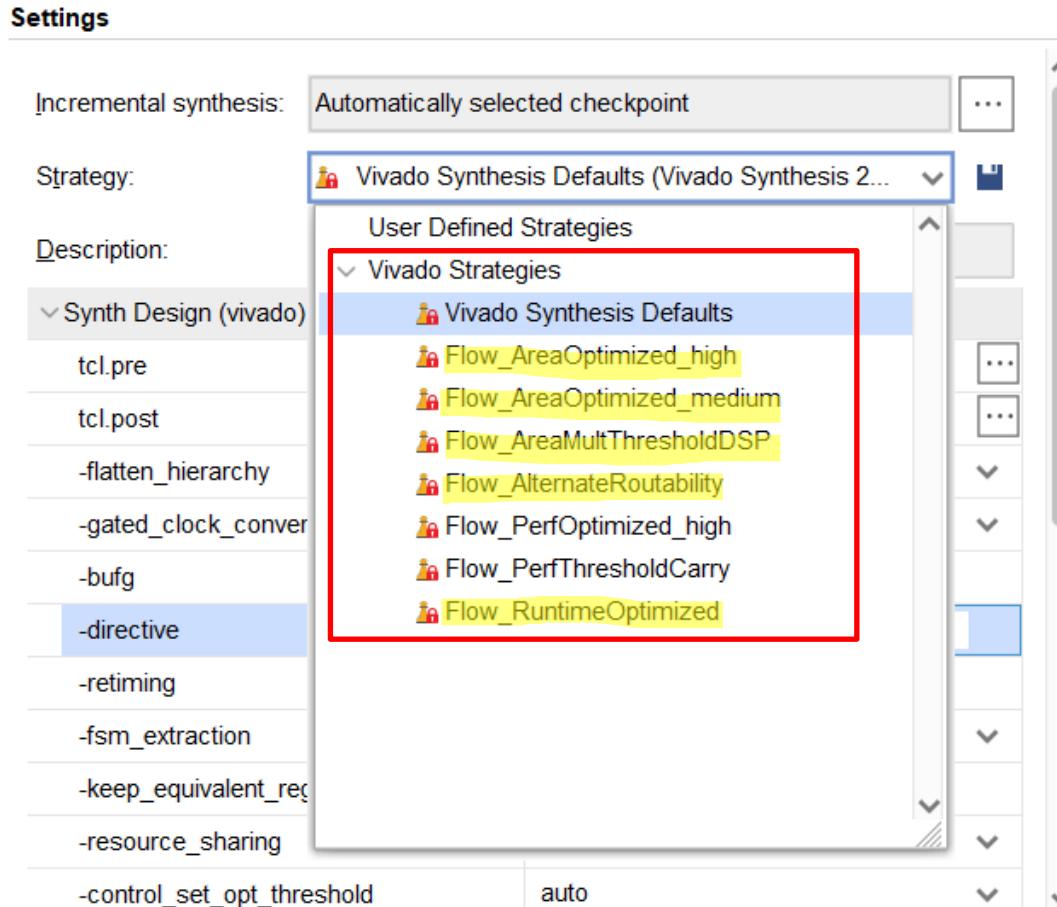
- Run Implementa

- Open Implemen



# Vivado Synthesis

- Synthesis Strategy



**Flow\_AreaOptimized\_high  
Flow\_AreaOptimized\_medium**

Global Area Optimized

**Flow\_AreaMultThresholdDSP**

Using more DSP resources

**Flow\_AlternateRoutability**

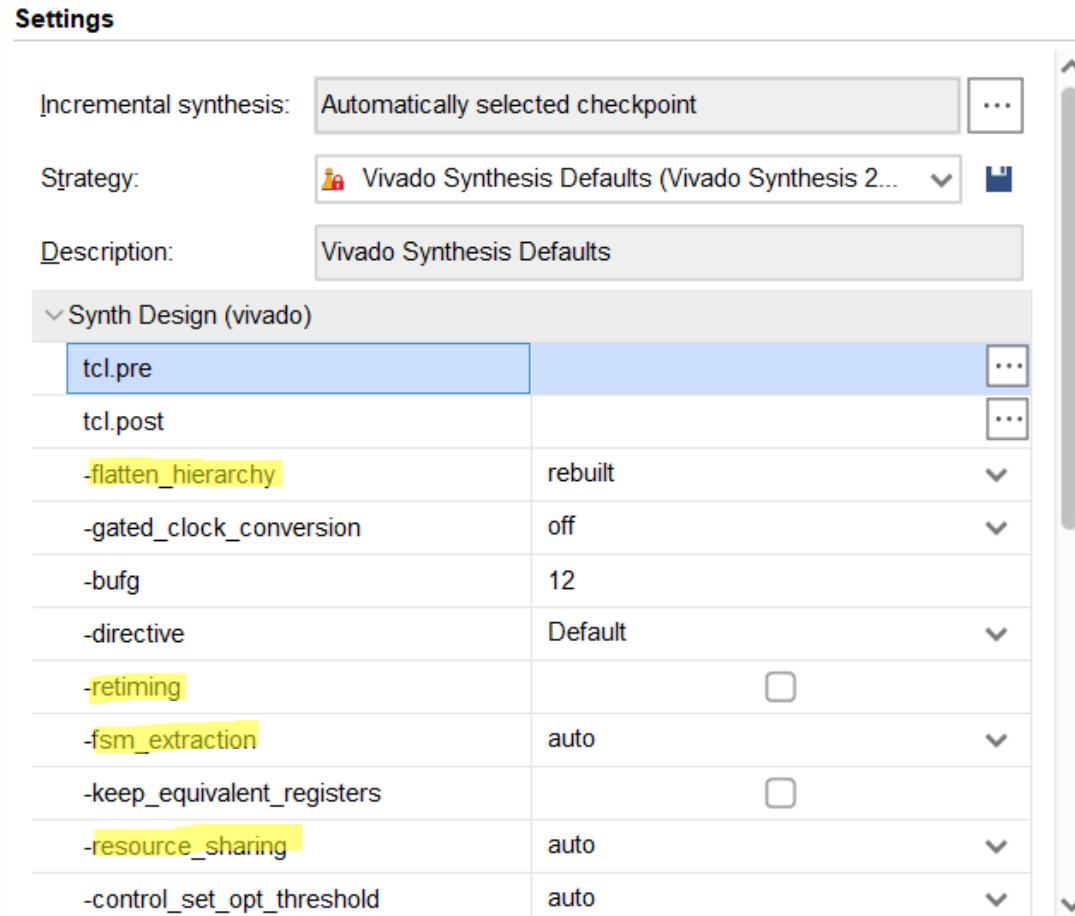
Increase Route ability and decrease the using of MUXF and CARRY

**Flow\_RuntimeOptimized**

This will ignore some RTL to decrease the Synthesis time

# Vivado Synthesis

- Synthesis Options



**Flatten\_hierarchy**

**None**

Low Optimization, High Hierarchy

**Rebuilt (default)**

**Full**

High Optimization, Low Hierarchy

Timing Optimization, Compressing and Integrate Register to form a new path

**Retiming**

**Fsm\_extraction**

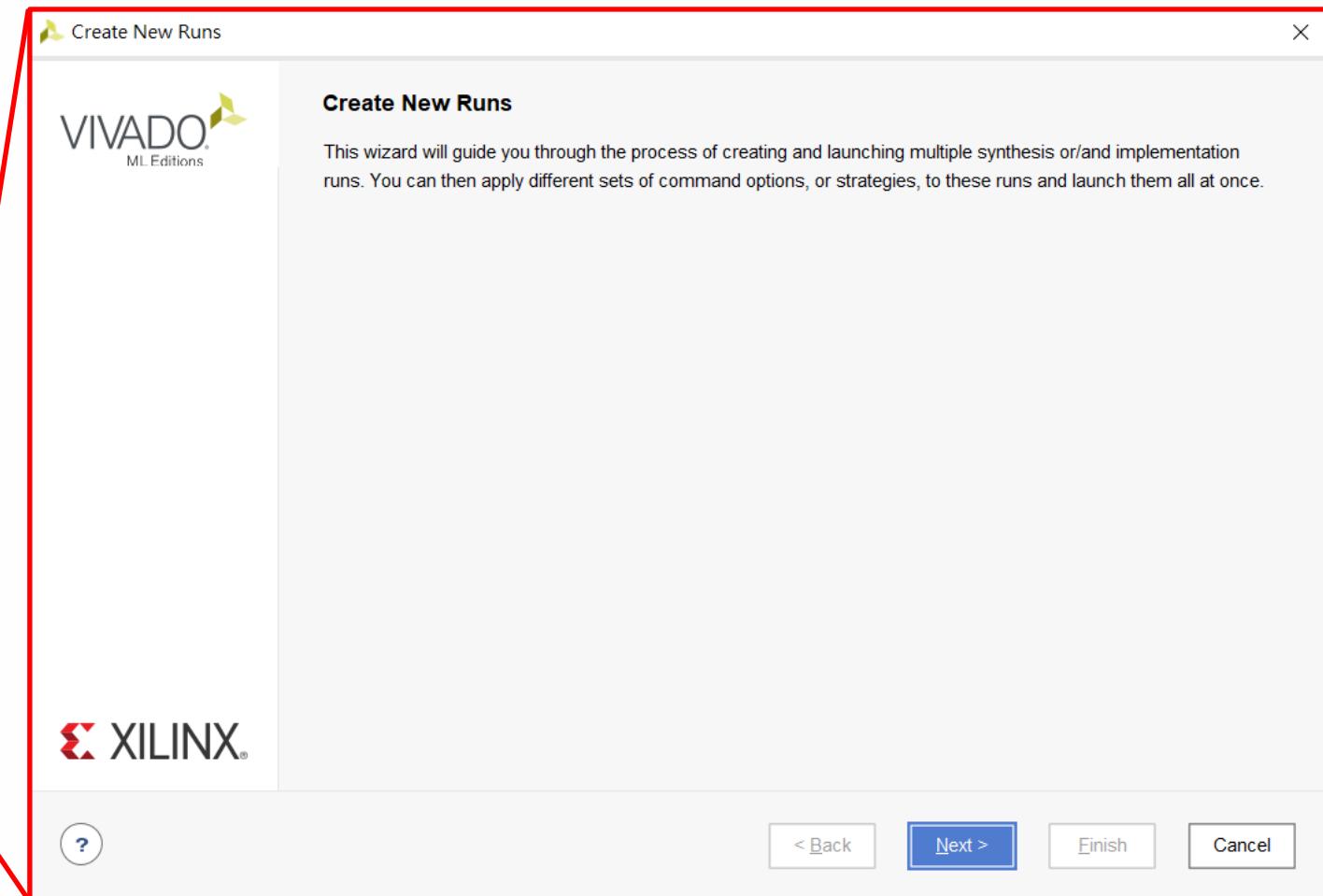
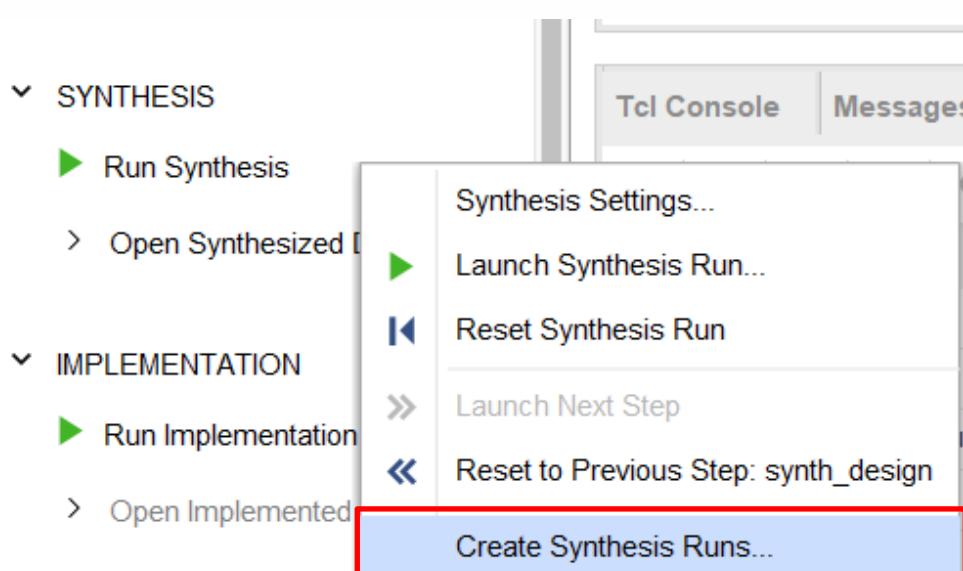
Vivado will infer the encoding way of FSM automatically

**Resource\_sharing**

Through using the shared resources to optimize the arithmetic operations, but will cause the timing to be longer

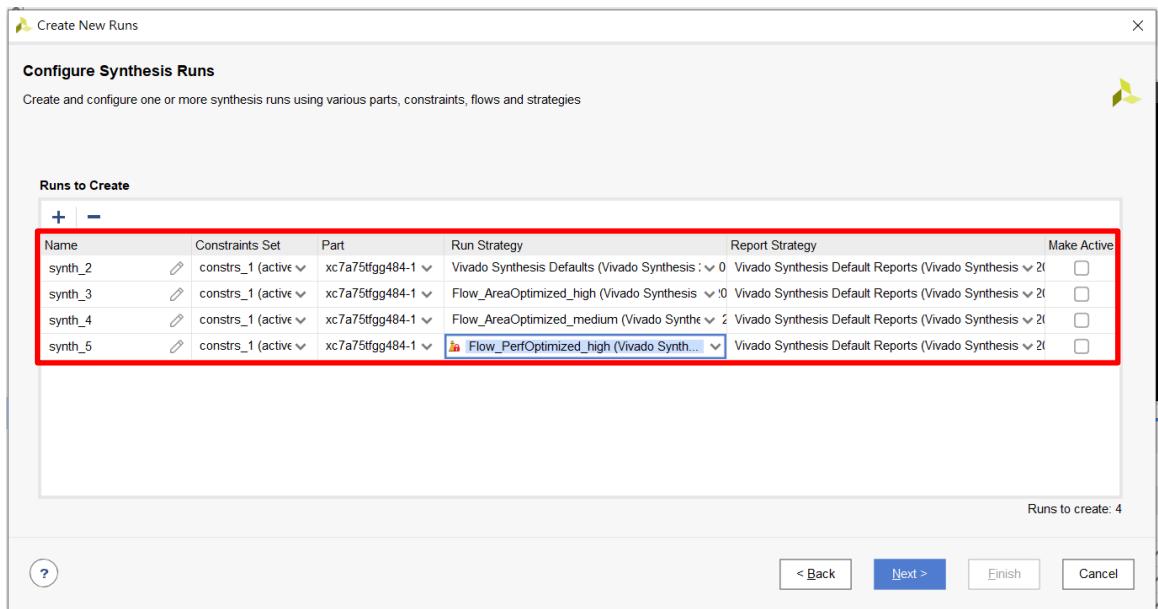
# Vivado Synthesis

- Create Synthesis Runs

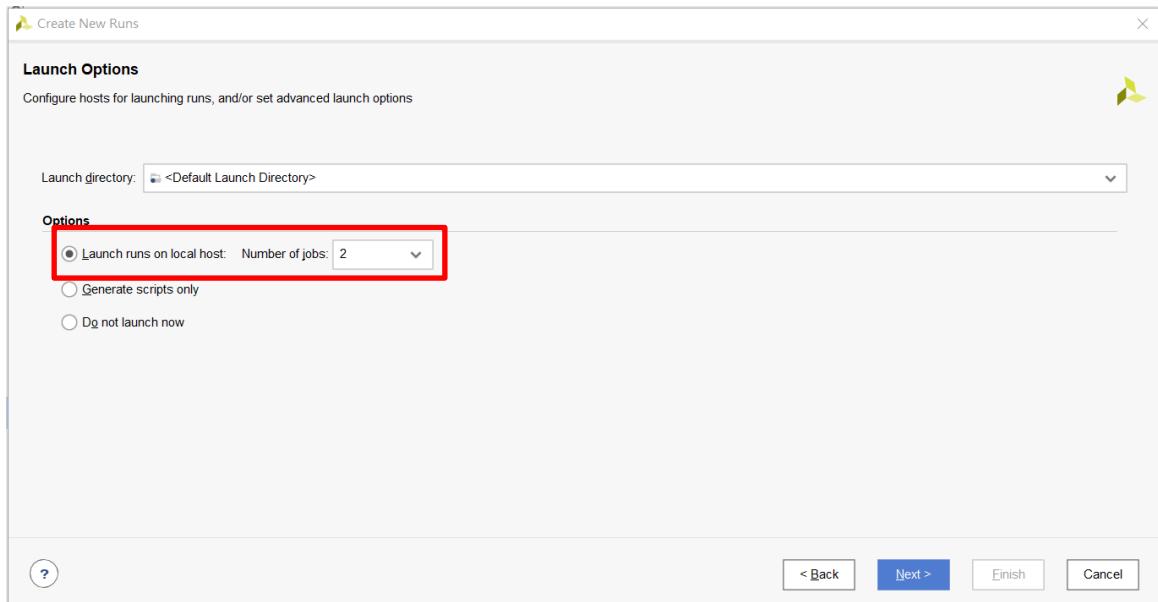


# Vivado Synthesis

- > Click “next” to add runs. This window offers creating multiple runs with different strategies, different constraint, even different report strategies.

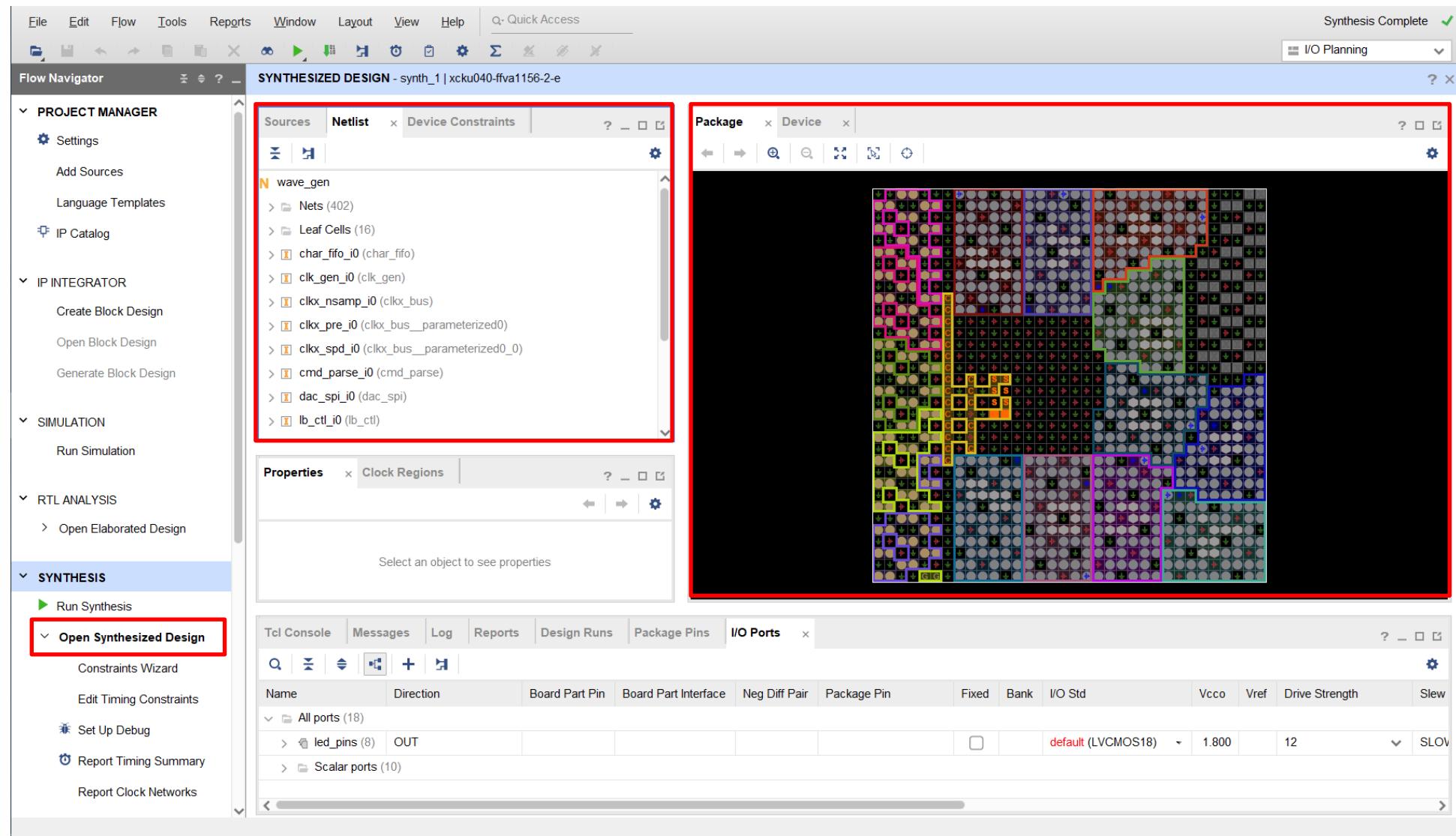


- > Next step offers changing launch directory and changing number of jobs



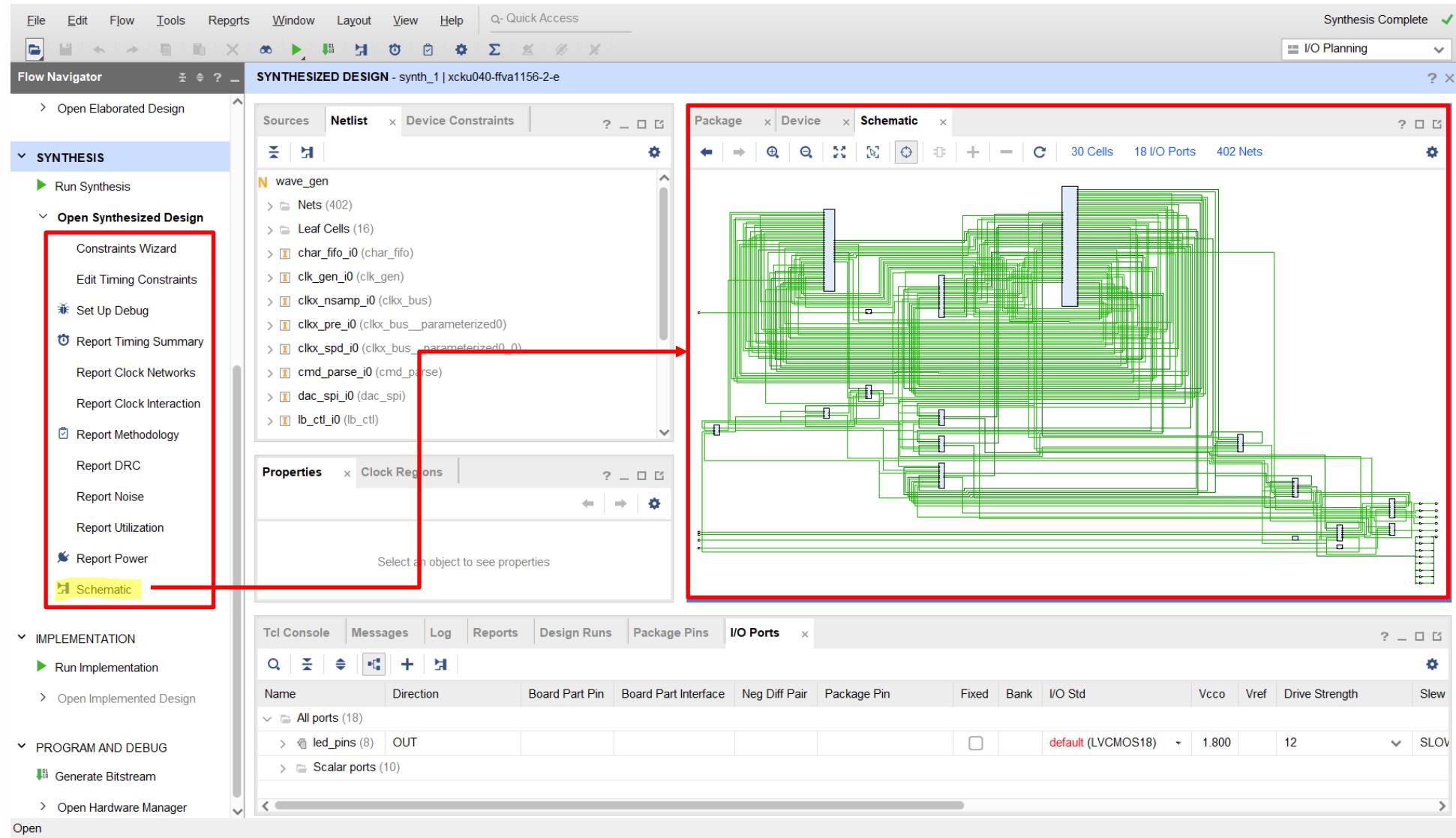
# Vivado Synthesis

- After Synthesis



# Vivado Synthesis

- After Synthesis

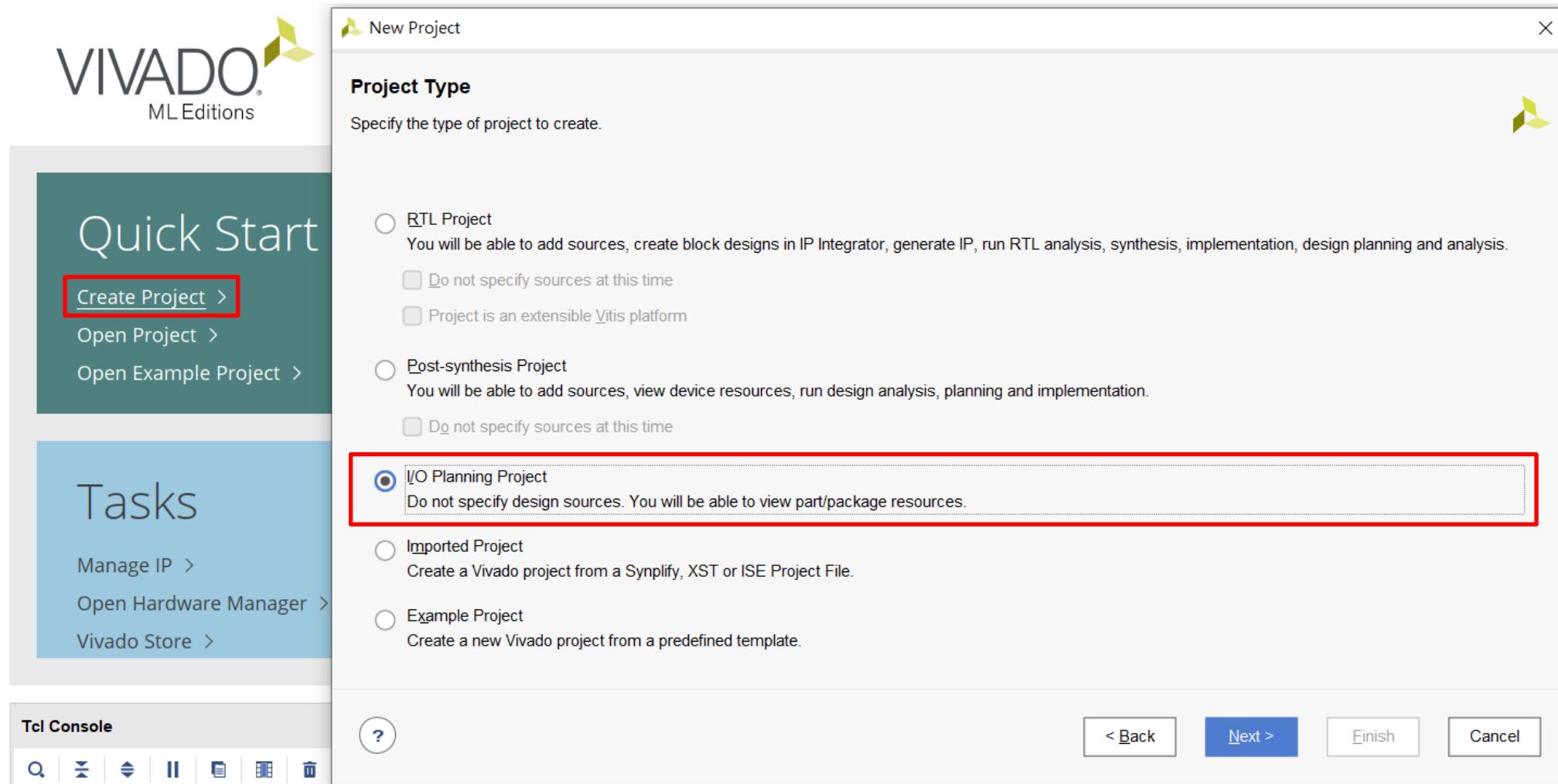


# Vivado Design Suite I/O Pin Planning

# Vivado I/O Pin Planning

- I/O Pin Planning can be executed at three different steps

## 1. Before RTL



# Vivado I/O Pin Planning

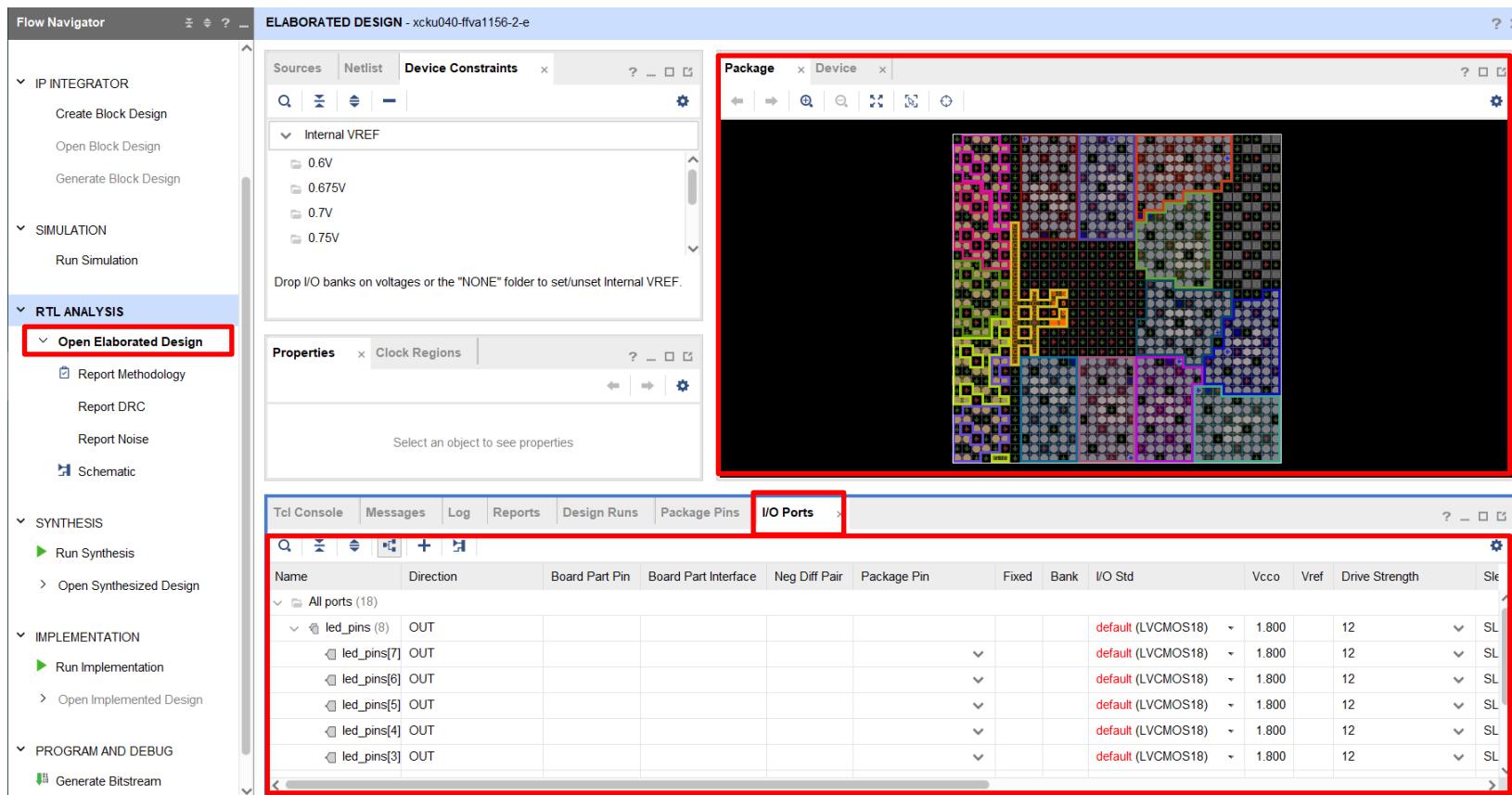
- I/O Pin Planning can be executed at three different steps
  1. Before RTL
    - This enables an early pinout definition, which can eliminate iterations related to device pinout changes later in the design cycle
    - Design Rule Check (DRC) will only check the basic I/O Bank spec, include
      - I/O banking rules
      - Avoid ground bounce
      - I/O Planner performs error checking for your pinout
- ◆ It is recommended that you have RTL associated Error checking is better.

# Vivado I/O Pin Planning

- I/O Pin Planning can be executed at three different steps

## 2. With RTL

- Can be executed at **Elaborated Design**
- Not add Xilinx Design Constraint (XDC) files in this step

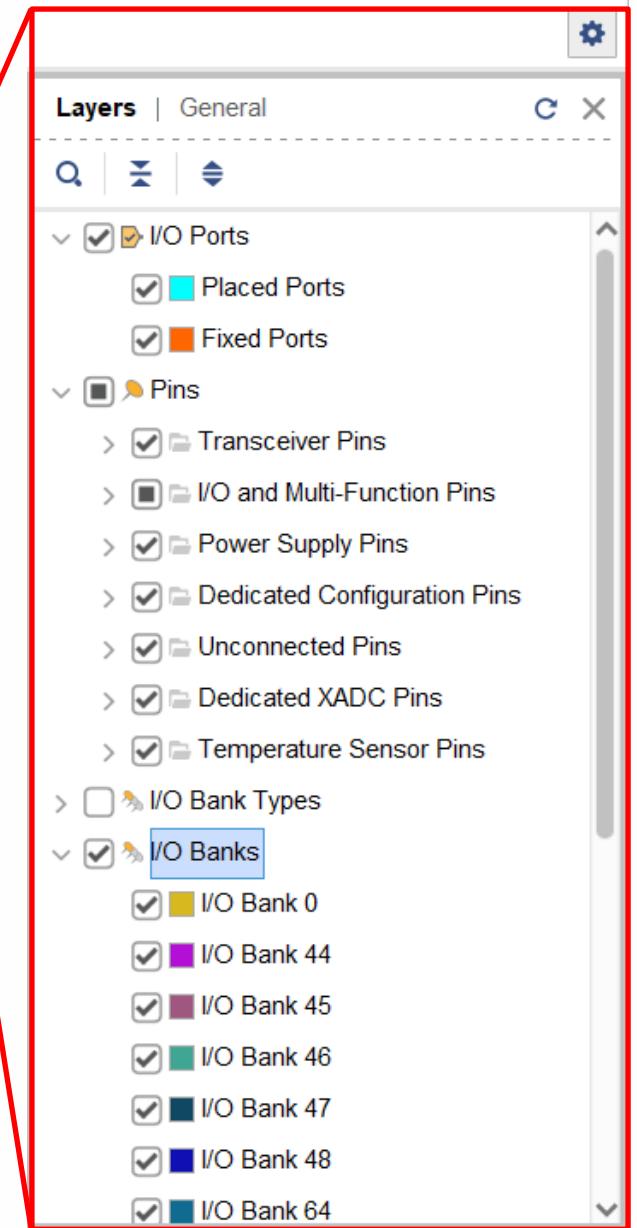
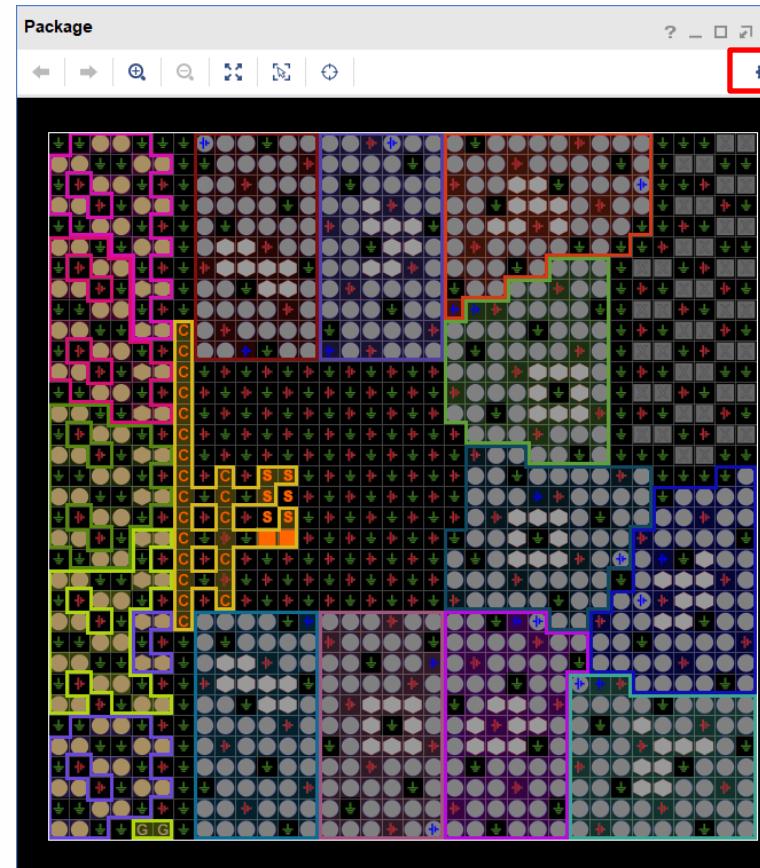
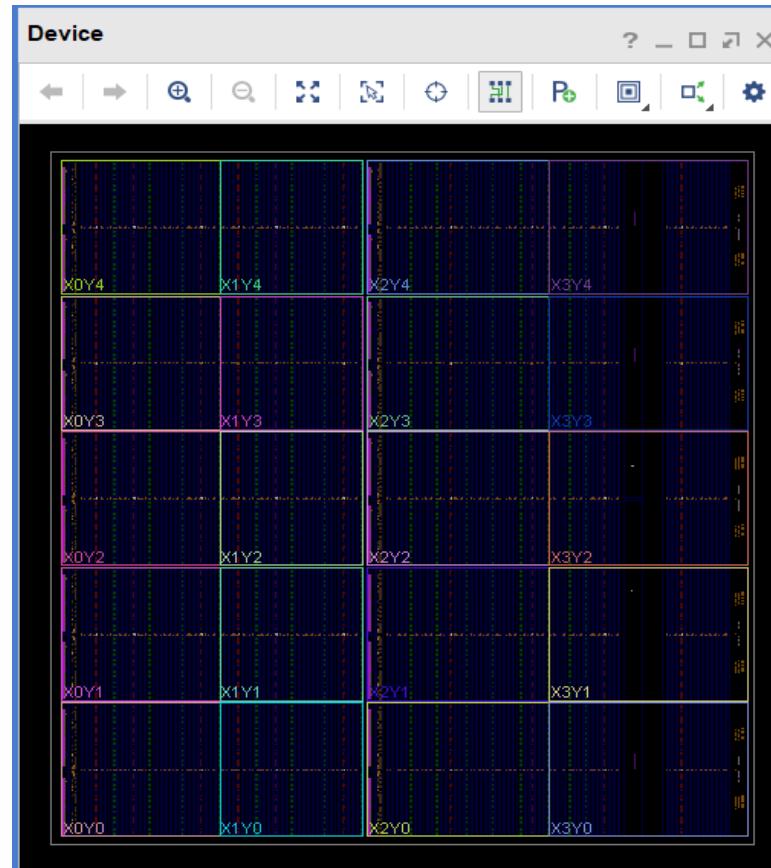


# Vivado I/O Pin Planning

- I/O Pin Planning can be executed at three different steps
  - 3. On Synthesized & Implemented Design
    - Already has Constraint files or the result of Place & Route
    - Can execute auto layout of I/O planning
    - I/O planning need to be validated at Implementation
- ◆ It is recommended that before doing the auto layout of I/O planning, you can layout the timing-critical path manually first.

# Vivado I/O Pin Planning Layout View

- Device & Package View



# Vivado I/O Pin Planning Layout View

- Device & Package View

The screenshot shows two windows from the Vivado interface. On the left is the 'Package' view, which displays a grid of circular pads on a device. A red box highlights a specific group of pads. On the right is the 'I/O Ports' table, which lists the assigned pins for various port names. The row for 'led\_pins[1]' is highlighted with a red box. A red arrow points from the highlighted area in the package view to the text 'Different color blocks mean different I/O Banks'.

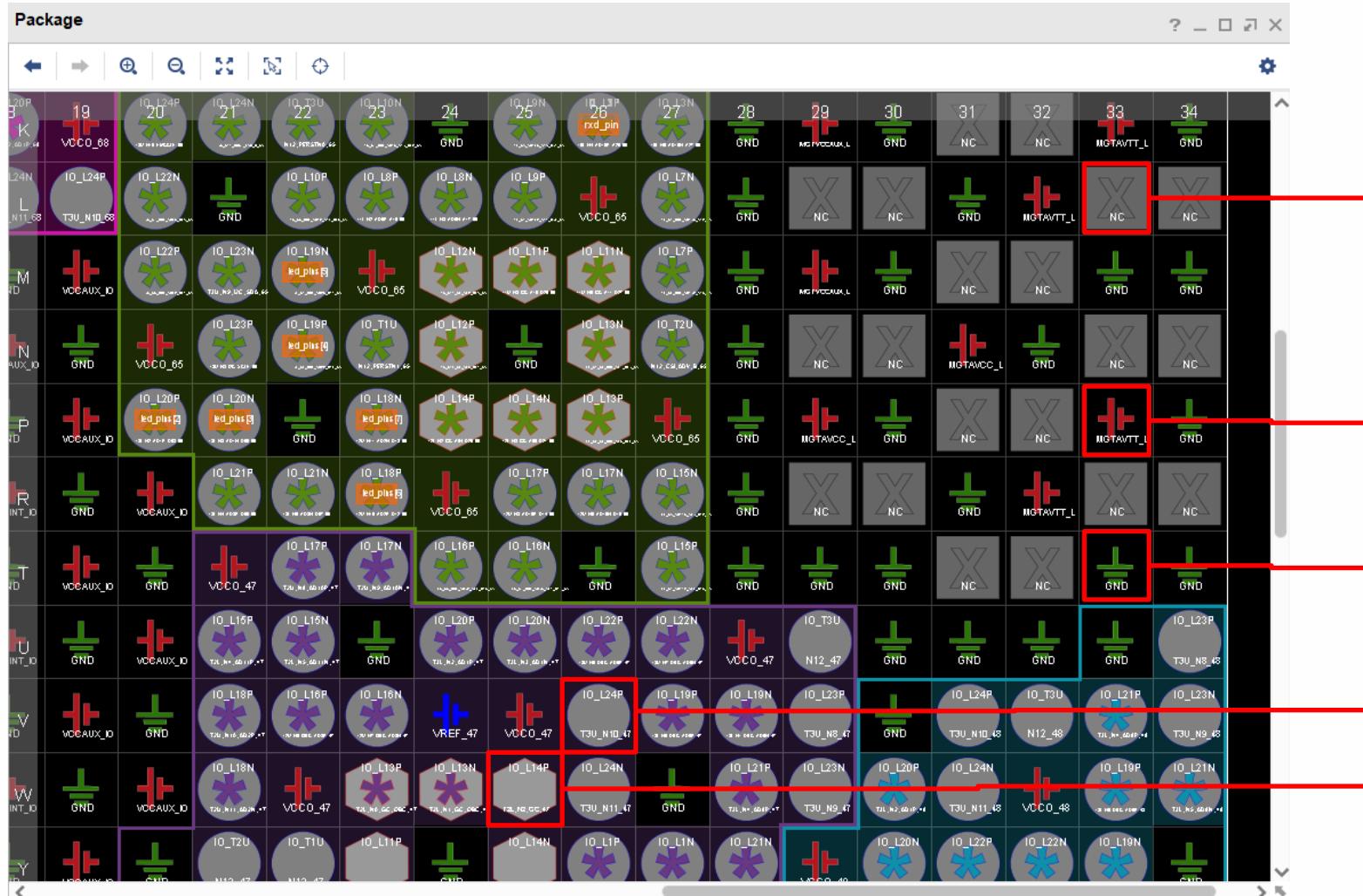
Different color blocks mean different I/O Banks

Name	Direction	Board Part Pin	Board Part Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type
led_pins (4)	OUT				88	LVC MOS33*	3.300	12			SLOW	
led_pins[3]	OUT	B5			88	LVC MOS33*	3.300	12			SLOW	
led_pins[2]	OUT	A5			88	LVC MOS33*	3.300	12			SLOW	
led_pins[1]	OUT	D6			88	LVC MOS33*	3.300	12			SLOW	
led_pins[0]	OUT	D5			88	LVC MOS33*	3.300	12			SLOW	
Scalar ports (5)												

Cross-probing Check

# Vivado I/O Pin Planning Layout View

- Device & Package View



No Connection

VCC

GND

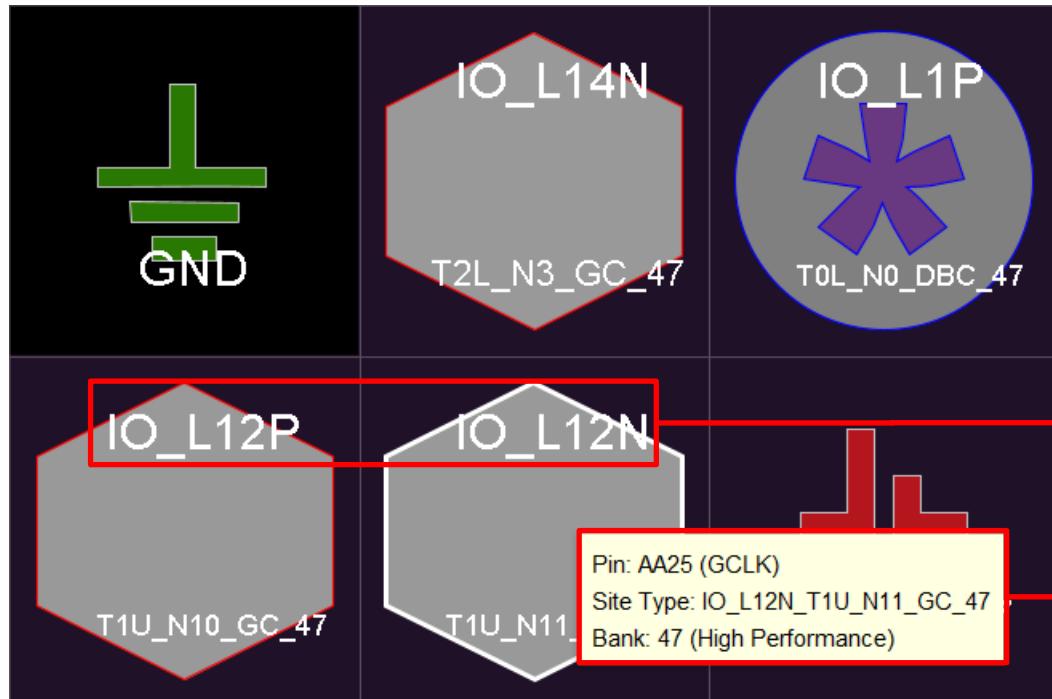
Normal I/O pins

Clock-capable pins

➤ For the input/output clock of PCB

# Vivado I/O Pin Planning Layout View

- Device & Package View



Differential Pair

Site Type : IO\_L12N  
Bank Number : 47  
Package Pin : L12  
High Performance Bank

# Vivado I/O Pin Planning Layout View

- I/O Ports View
  - List the current all I/O ports and show the detail information

Gray color means the pin has no assignment

Yellow color means the pin has already been assigned

Name	Direction	Board Part Pin	Board Part Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew T
clk_pin_n	IN					▼	<input type="checkbox"/>		default (LVC)	1.800		
clk_pin_p	IN				AK17	▼	<input checked="" type="checkbox"/>	45	DIFF_SSTL1	1.200		
dac_clr_n_	OUT				AB34	▼	<input checked="" type="checkbox"/>	48	LVC MOS18	1.800	12	SLOW
dac_cs_n_	OUT				AA34	▼	<input checked="" type="checkbox"/>	48	LVC MOS18	1.800	12	SLOW
lb_sel_pin	IN				AN16	▼	<input checked="" type="checkbox"/>	45	LVC MOS12*	1.200		
rst_pin	IN				AN8	▼	<input checked="" type="checkbox"/>	64	LVC MOS18	1.800		
rxd_pin	IN				K26	▼	<input checked="" type="checkbox"/>	65	LVC MOS18	1.800		

# Vivado I/O Pin Planning Layout View

- Place I/O Ports
  - If you determine the relationship between the I/O ports and the Package Pins, you can use three ways of placing

## 1. I/O Bank

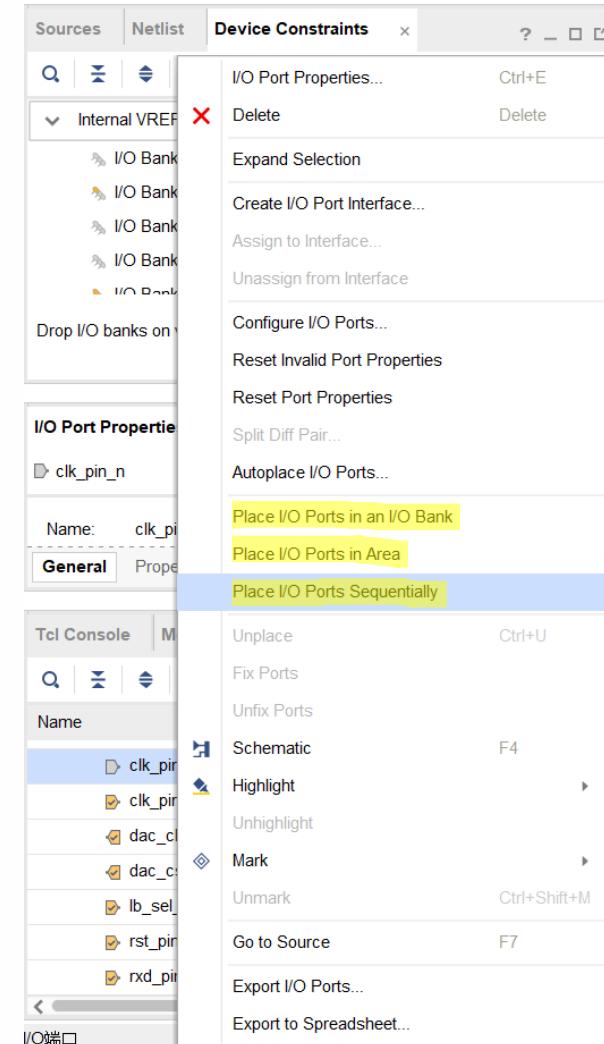
Select multiple I/O and place them in one bank in Device View, Vivado will keep them all in the same bank as much as possible

## 2. Area

Select multiple I/O and the pointer will become “cross” type, then frame an area where you want to place. Vivado will keep them all in the same area as much as possible

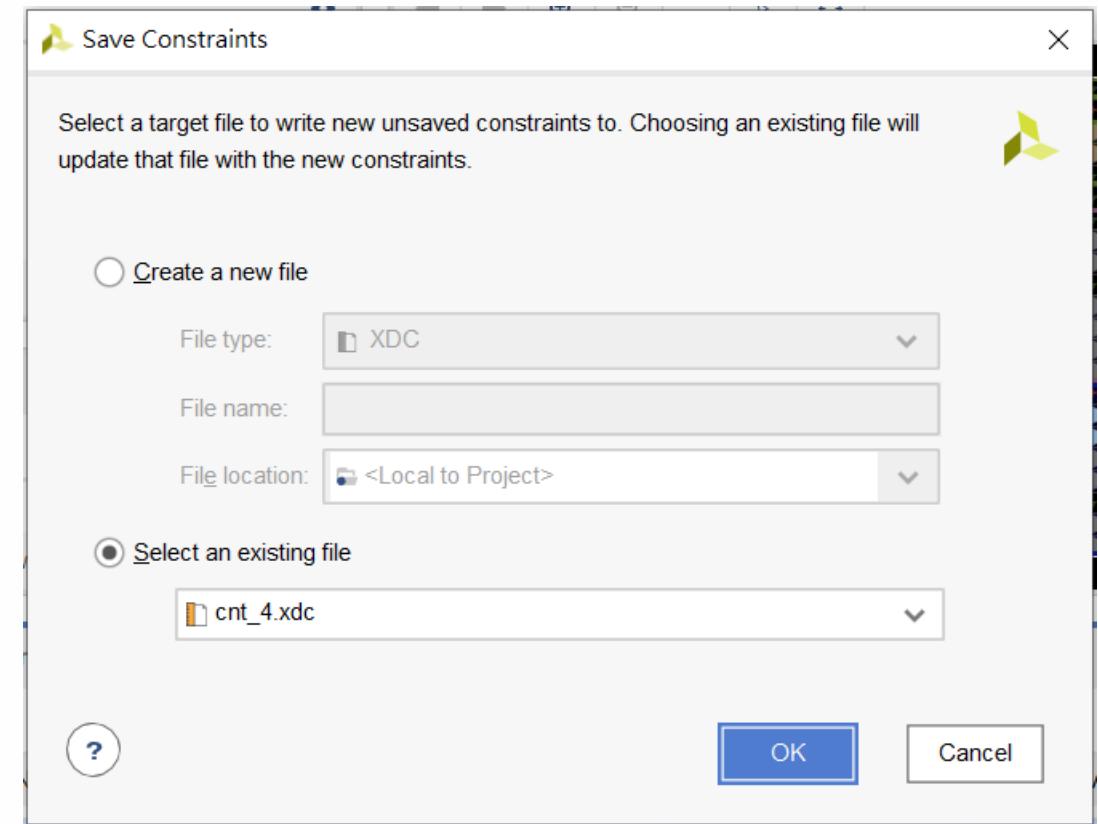
## 3. Sequentially

Select multiple I/O and click the location you want to place in Device View



# Vivado I/O Pin Planning

- > Since I/O port setting is a kind of physical constraint, it saves to the XDC file
- > After setting the I/O ports, press “Ctrl + S” on keyboard to save.
- > “Save Constraint” pops out. In this step, you can choose to create a new constraint file or save to an existing constraint file



# Vivado I/O Pin Planning Flow

Define I/Os early in the design process

- Import RTL, CSV, XDC

Interactively analyze and assign I/Os

- DRC legal group drag and drop; assign clock logic

Robust DRCs

- Find problems prior to implementation

SSN analysis

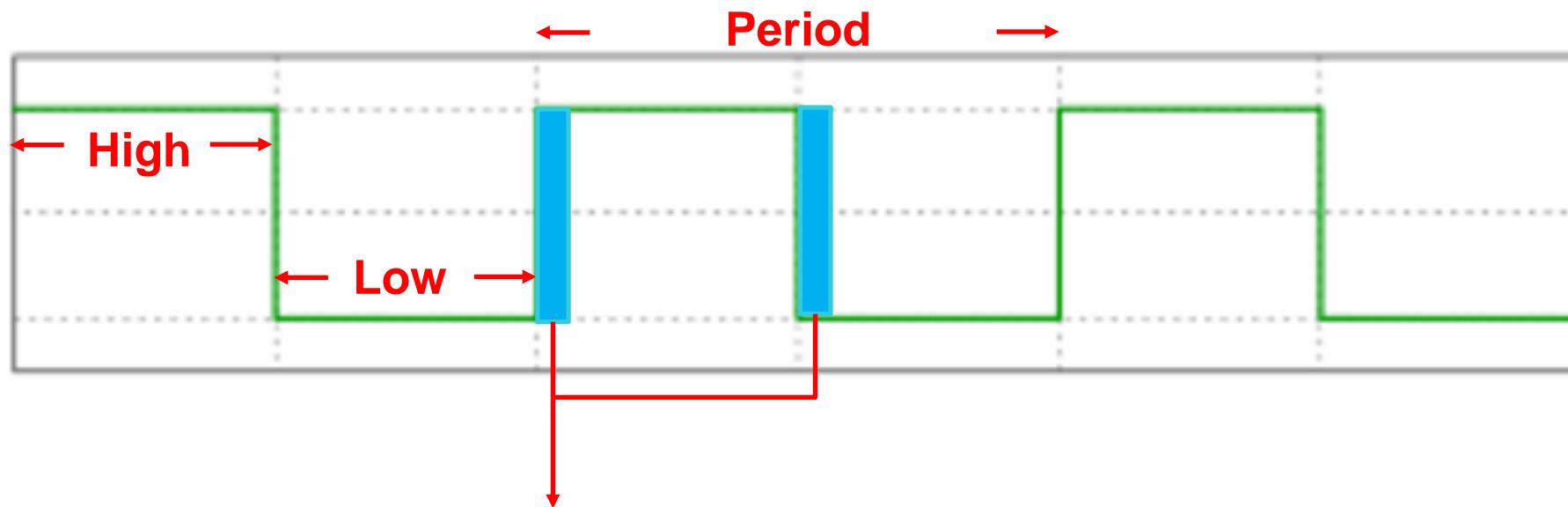
- Estimates switching noise levels caused by switching output ports

## ◆ Simultaneous Switch Noise (SSN) Analysis

- Avoid Ground Bounce happening
- High-speed circuits like Transceivers need SSN analysis very much

# Vivado Clock Constraints

# Clocks



$$\text{Duty Cycle} = \frac{\text{High}}{\text{Period}}$$

# Clocks

- Using TCL to control clocks

## 1. Create

```
create_clock -name <clock name> -period <period value> -waveform {value1 value2}
```

ns

Start

End

e.g., `create_clock -name clk2 -period 10.0 -waveform {0.1 2.1}`



$$\text{Duty Cycle} = \frac{2.1 - 0.1}{10} = 20\%$$

# Clocks

- Using TCL to control clocks

## 2. Get

- `get_clocks <clock name>`: Often used with other commands
- `get_property [ get_clocks <clock name> ]`
- `report_property[ get_clocks <clock name> ]`

```
report_property [get_clocks clk_samp]
Property          Type   Read-only  Value
CLASS            string  true      clock
DIVIDE_BY        int    true      32
FILE_NAME        string  true      C:/training/Designing FPGAs Using the Vivado Design Sui-
INPUT_JITTER     double  true      0.000
IS_GENERATED     bool   true      1
IS_INVERTED      bool   true      0
IS_PROPAGATED    bool   true      1
IS_RENAMED       bool   true      0
IS_USER_GENERATED bool  true      1
IS_VIRTUAL       bool   true      0
LINE_NUMBER      int    true      29
NAME             string  true      clk_samp
PERIOD           double  true      0.000
SOURCE           pin    true      clk_gen i0/BUFGCE clk samp i0/I
```

# Clocks

- Using TCL to control clocks

## 3. Set

### ➤ Setting Jitter

✓ `set_system_jitter <value>` **set jitter of all clock in system**

✓ `set_input_jitter <clock name> <value>` **set jitter for a certain clock**

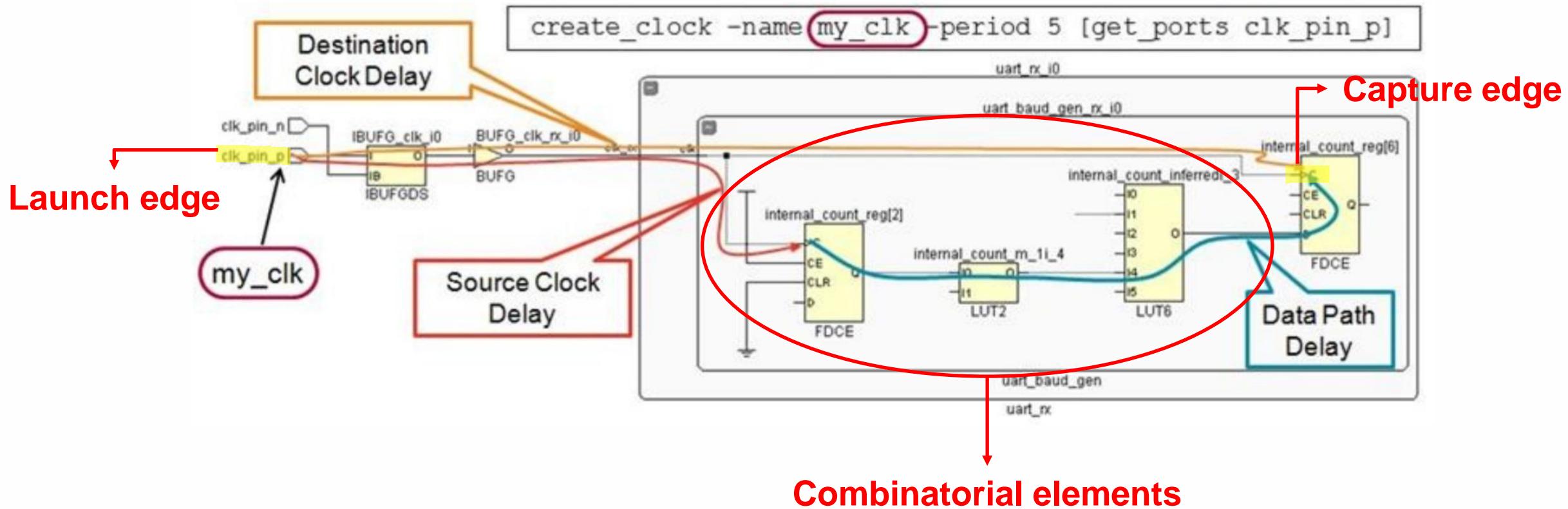
### ➤ Setting Latency **additional latency**

✓ `set_clock_latency -source <latency value> <objects>`

e.g., `set_clock_latency -source {  
-early  
-late}` 0.2 [get\_clocks <clock name>]

# Static Timing Analysis (STA)

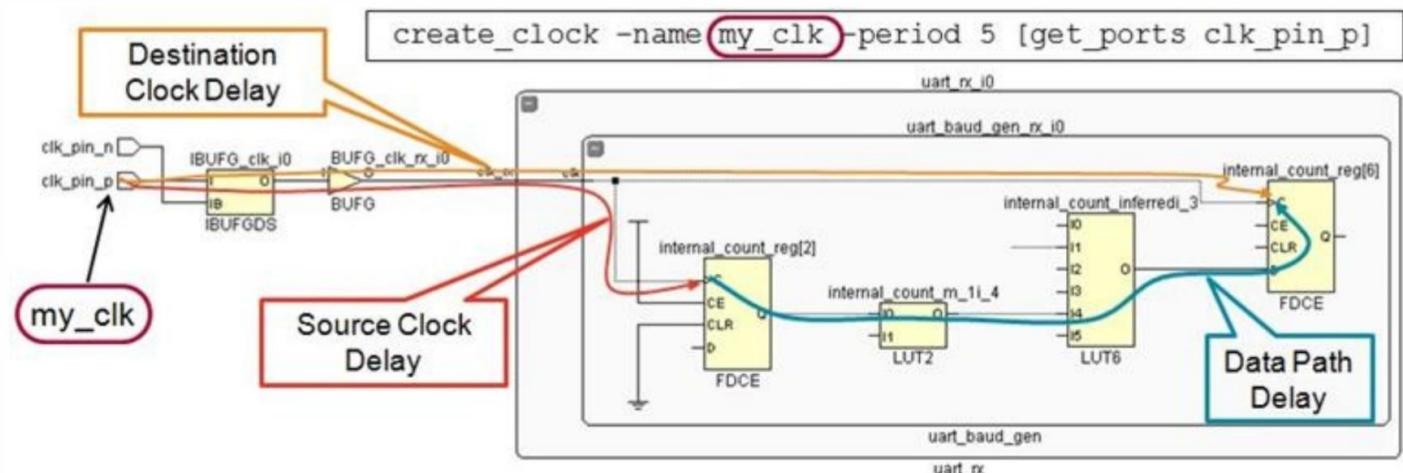
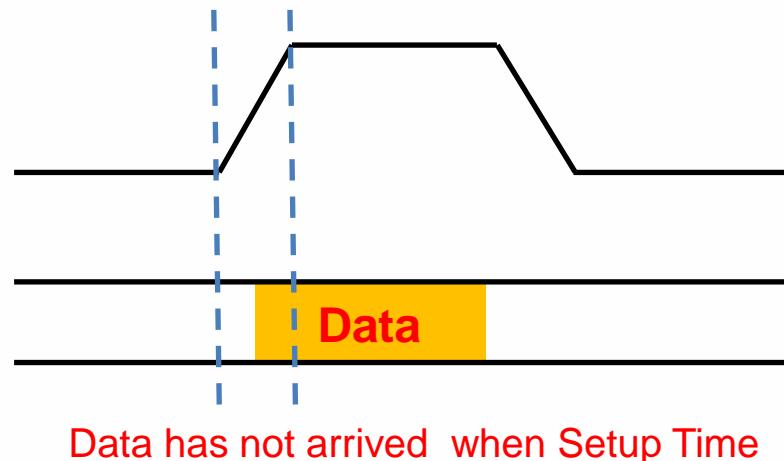
- Analyze the timing performance but not function in circuit



# Static Timing Analysis (STA)

## 1. Setup Check

- It is required that the data transmission should not be too slow, otherwise the data cannot be read correctly because it has not arrived at the Capture edge when Setup Time.

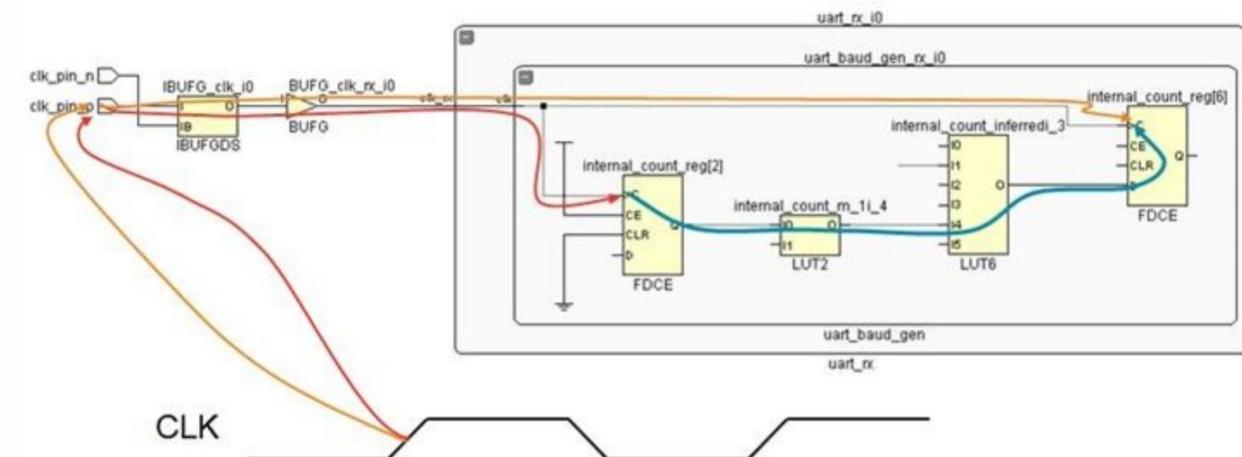
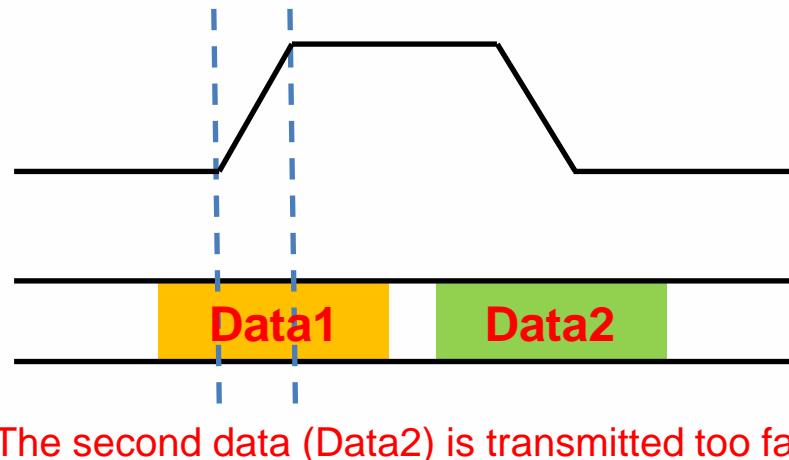


$$\text{Setup Slack} = T_{Period} + T_{Destination} - T_{source} - T_{Data} - T_{Uncertainty} > 0$$

# Static Timing Analysis (STA)

## 2. Hold Check

- If the data is transmitted too fast, the Capture edge is still processing the previous data, and the second data comes first, which will cause the read errors.



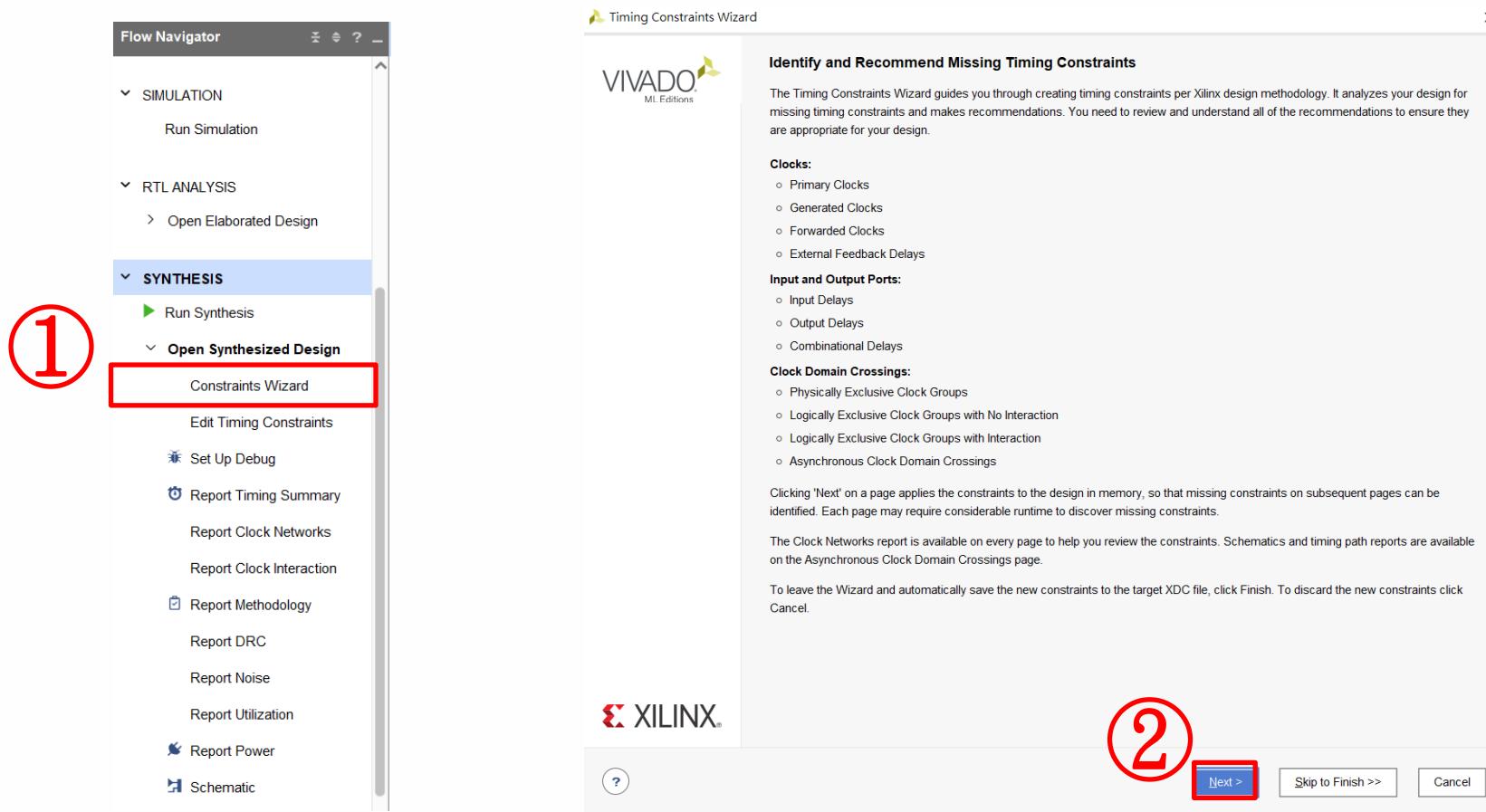
$$\text{Hold Slack} = T_{source} + T_{Data} + T_{Uncertainty} - T_{Destination} > 0$$

# Timing Constraints Wizard

- Can be run after Synthesis or Implementation
- Consists of [Constraints Wizard](#) and [Edit Timing Constraints](#)

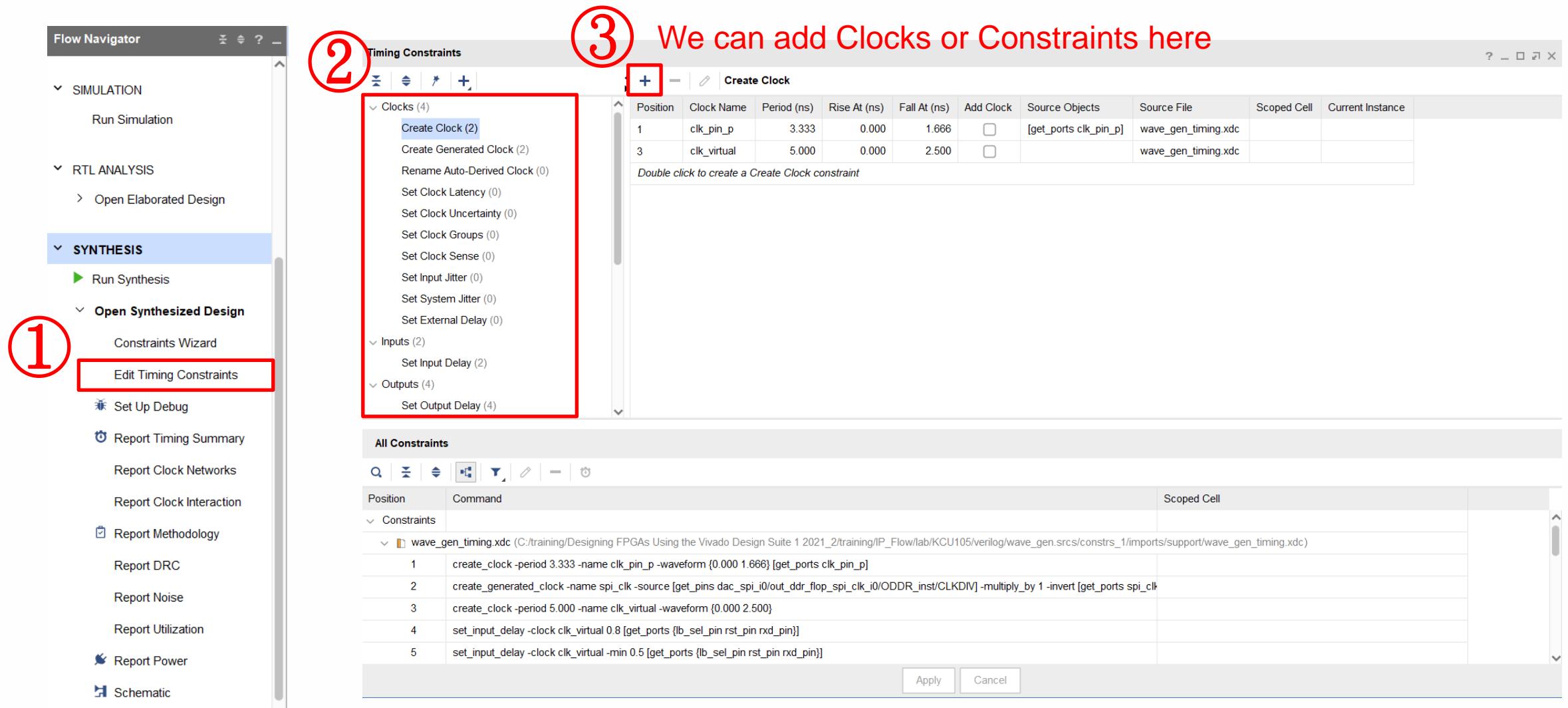
# Timing Constraints Wizard

- Constraints Wizard
  - Vivado will auto recognize the clocks with no constraints, and create constraints in order of Primary CLK, Generated CLK, I/O delay Constraints,... etc.



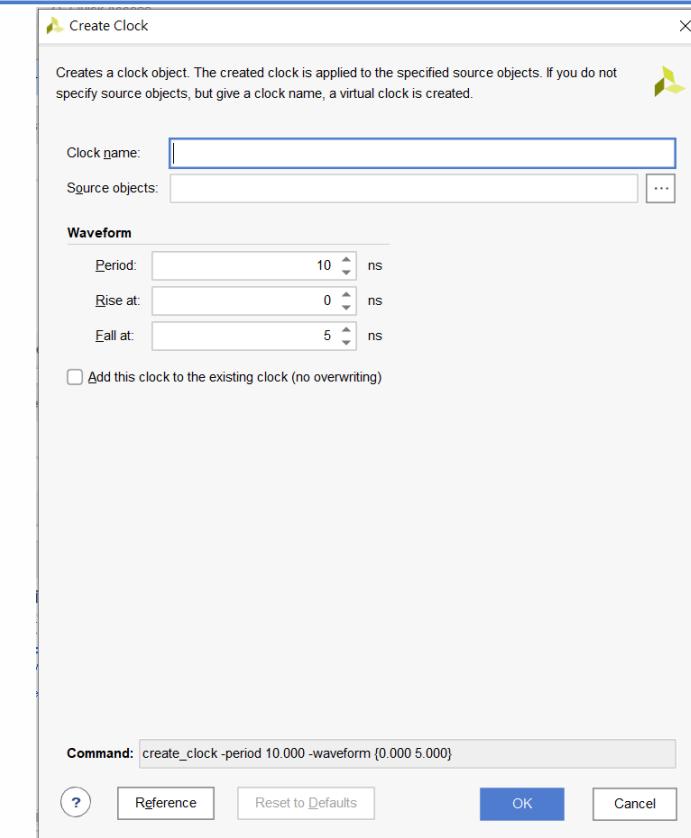
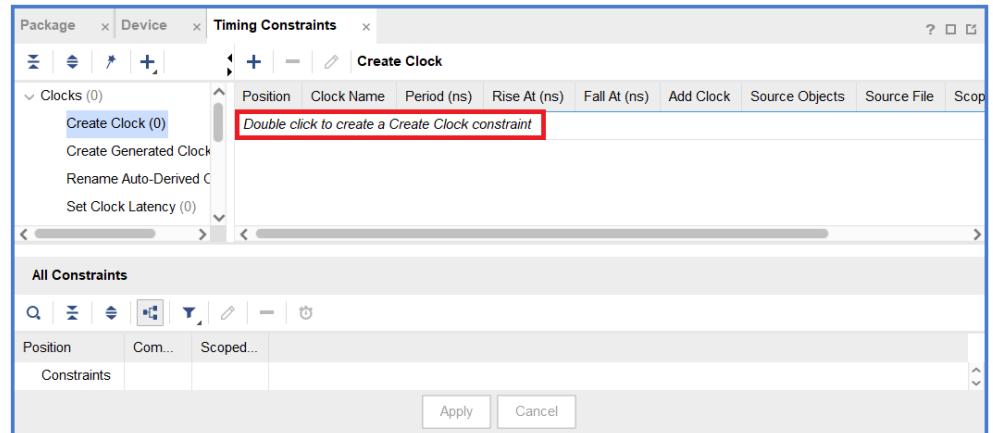
# Timing Constraints Wizard

- Edit Timing Constraints



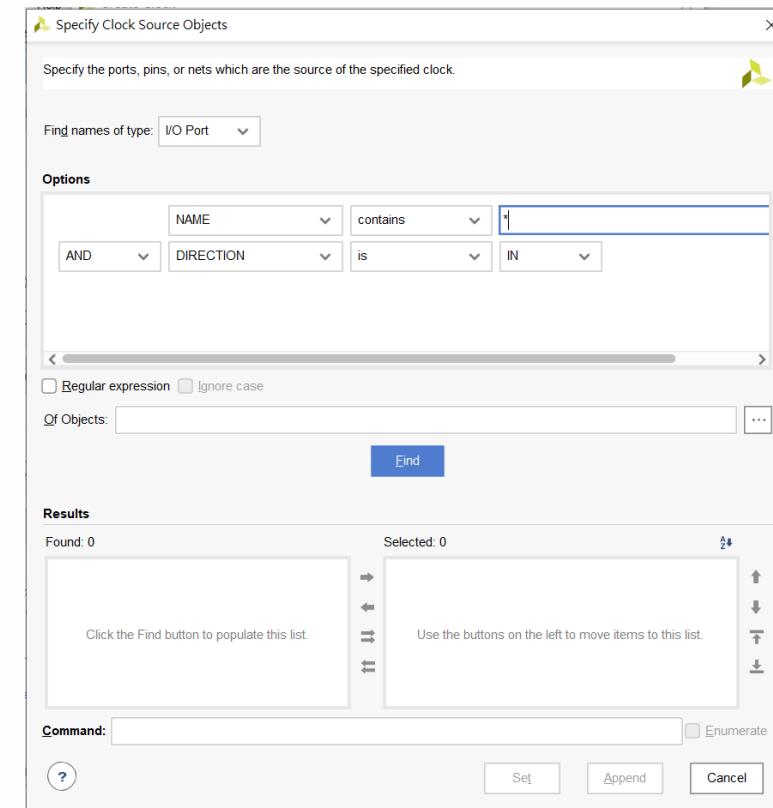
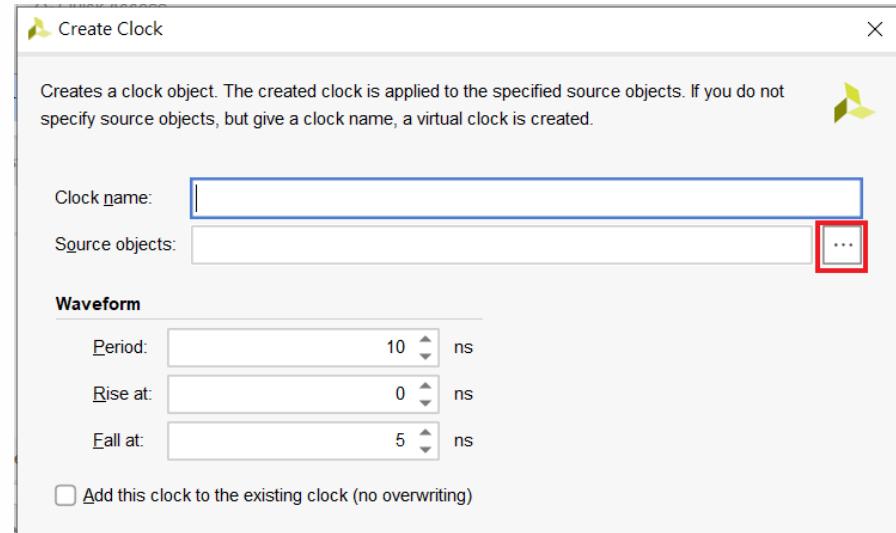
# Timing Constraints Wizard

- Edit Timing Constraints
  - > Move cursor to “Double click to create a Create Clock constraint”, then double click.
  - > “Create Clock” window pops out



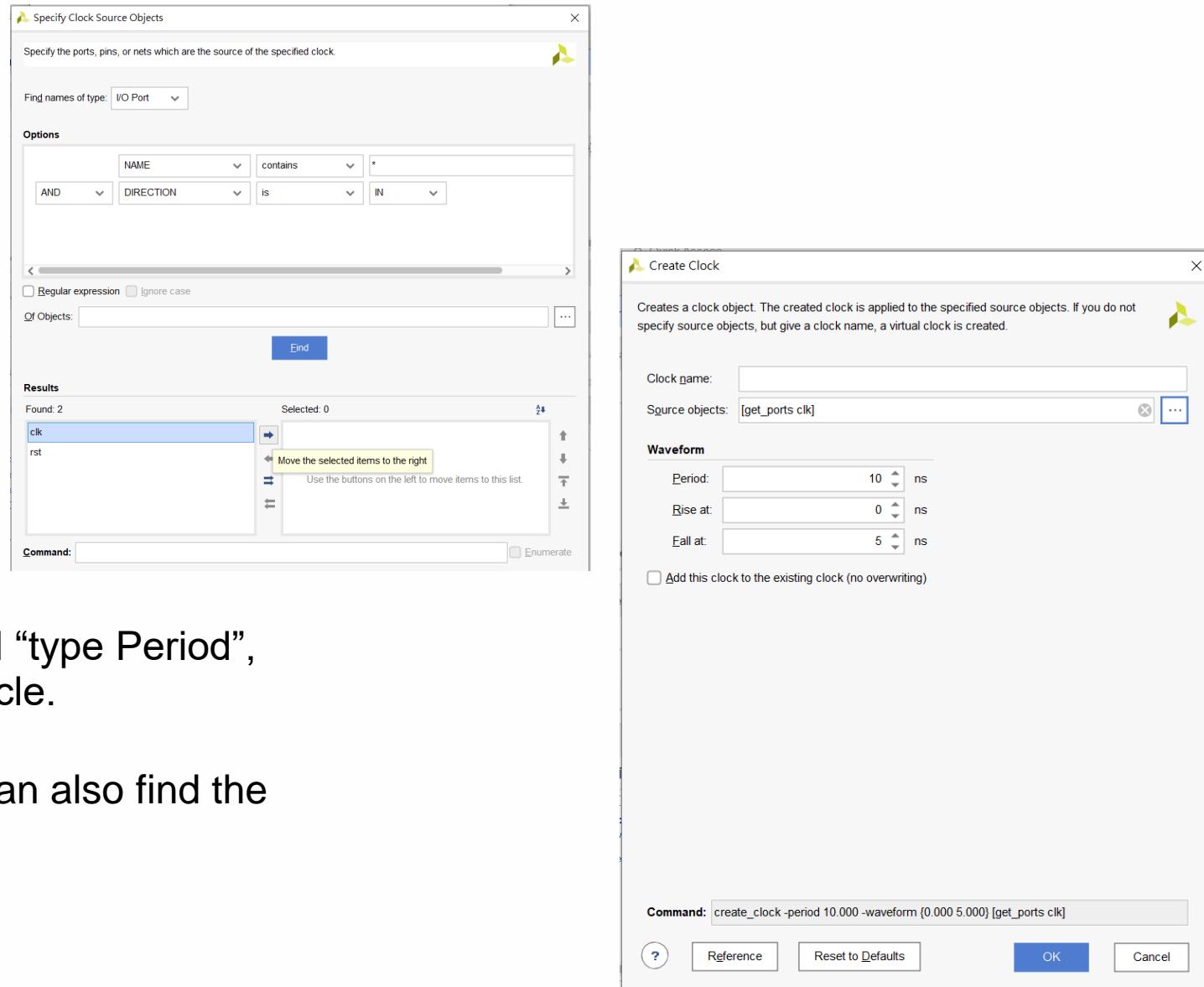
# Timing Constraints Wizard

- Edit Timing Constraints
- > Move the cursor to “...”, then press it to find source objects
- > Then the “Specify Clock Source Objects” window pops out, it offers multiple ways to find the object that can be put timing constraint rules on it.



# Timing Constraints Wizard

- Edit Timing Constraints
- > In this tutorial, randomly choose “clk” as an object to add a simple timing constraint on it. After choosing an object, move the selected object to the right.
- > In this step, you can also choose multiple objects to the right at once
- > Since the object has been chosen, you could “type Period”, “Rise at”, “Fall at” in ns to define the duty cycle.
- > At the bottom of Create Clock window, you can also find the corresponding Tcl command.



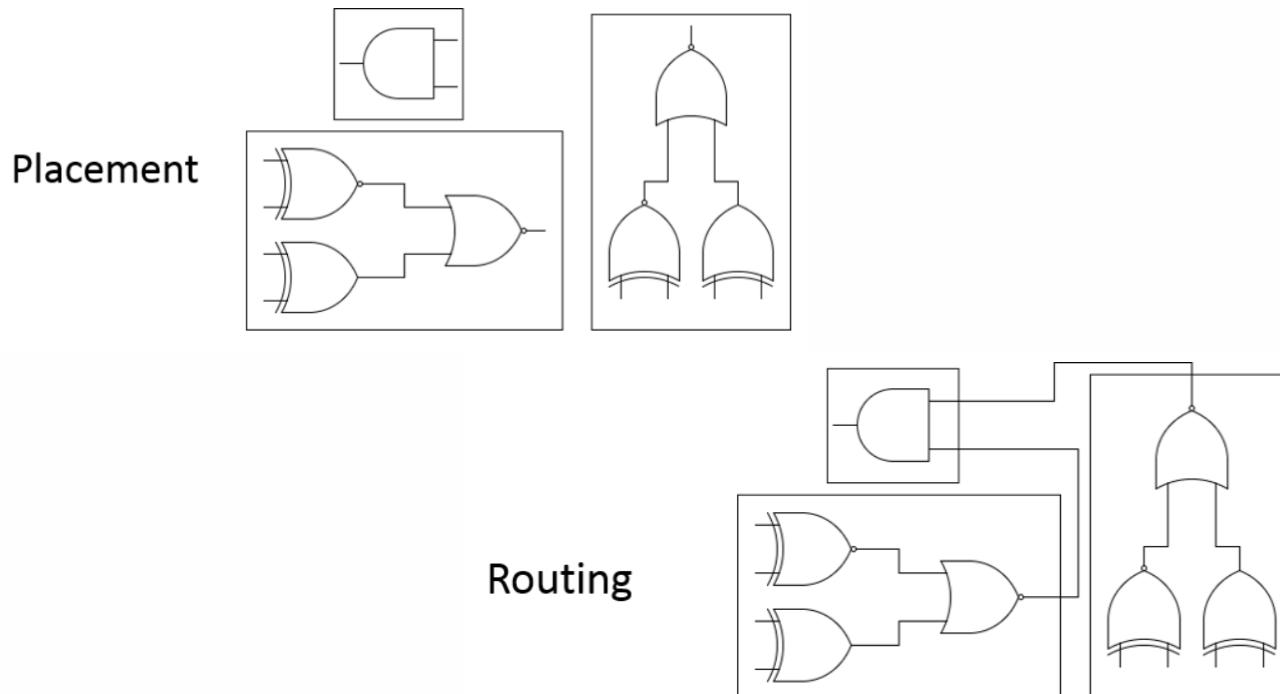
# Vivado Implementation



# Vivado Implementation

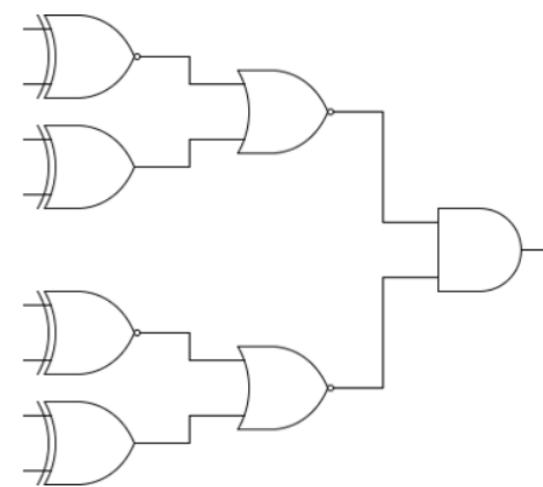
- Implementation vs. Synthesis
  - Implementation is mainly in the process of **Place & Route**, and focus on the **Timing** in the actual circuit to match the rule

## Implementation



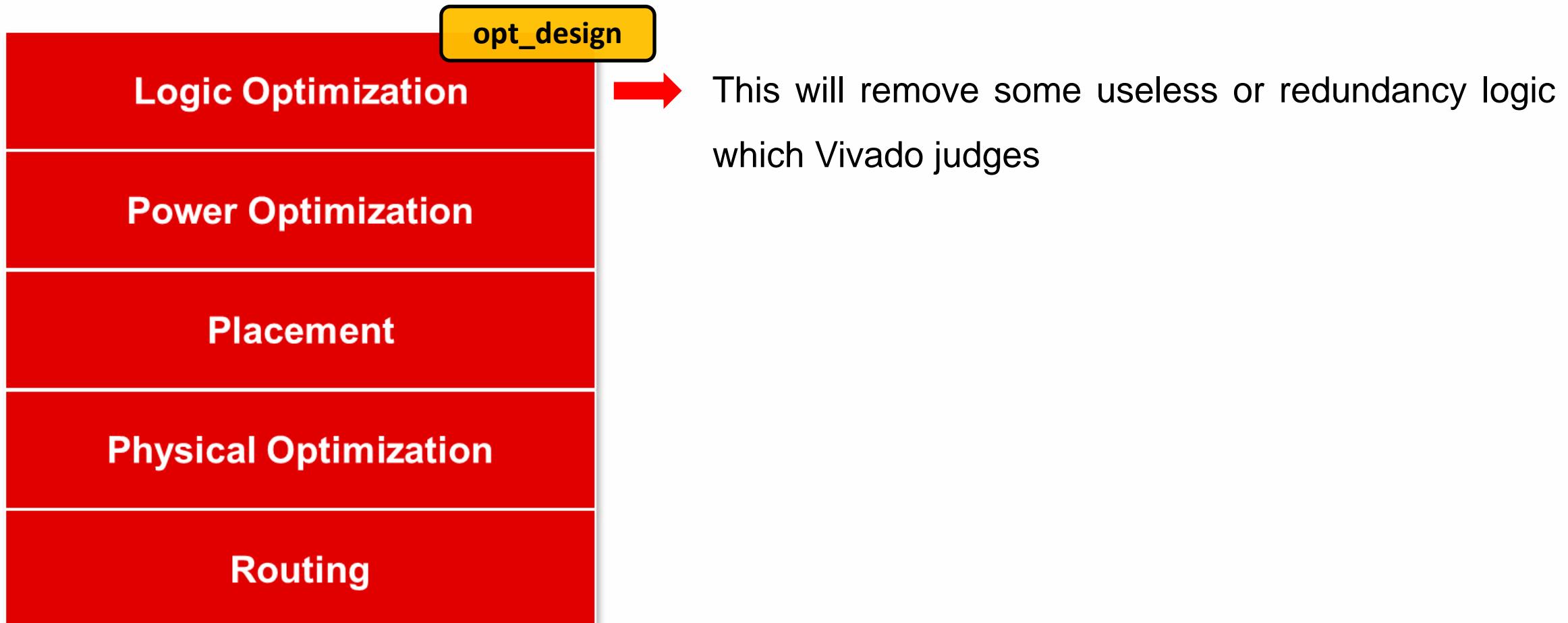
## Synthesis

### Netlist



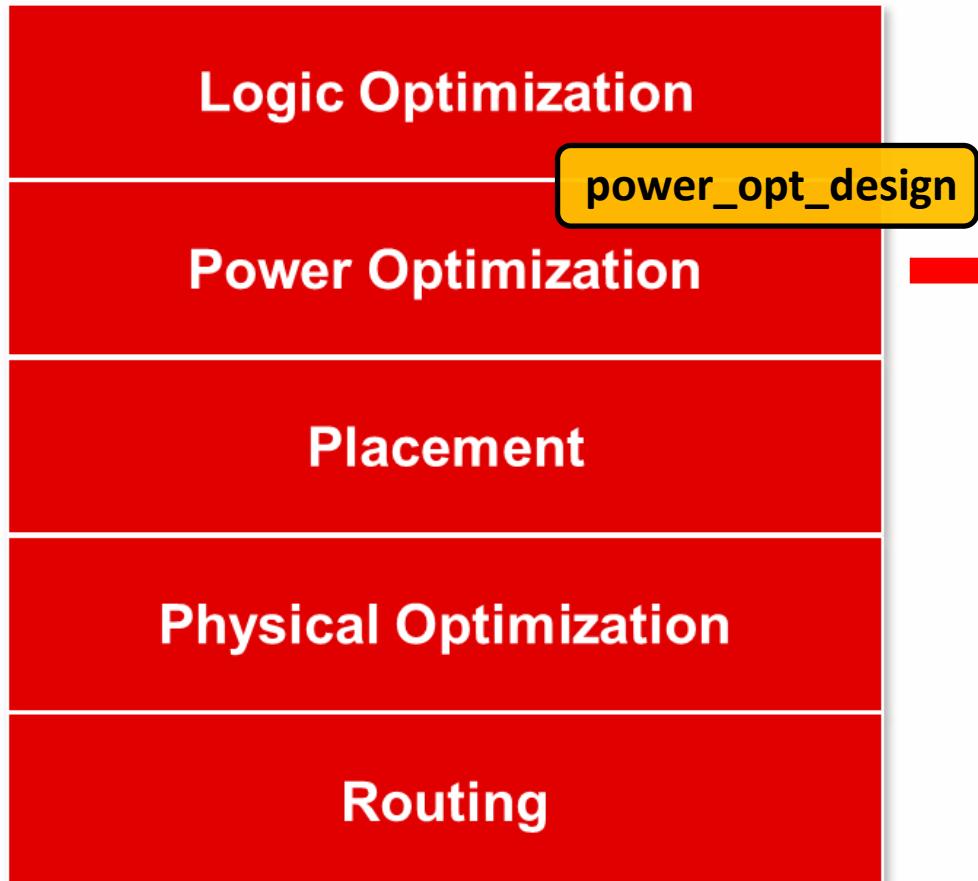
# Vivado Implementation Processes

- 5 Processes but one is optional, and all will generate .dcp file



# Vivado Implementation Processes

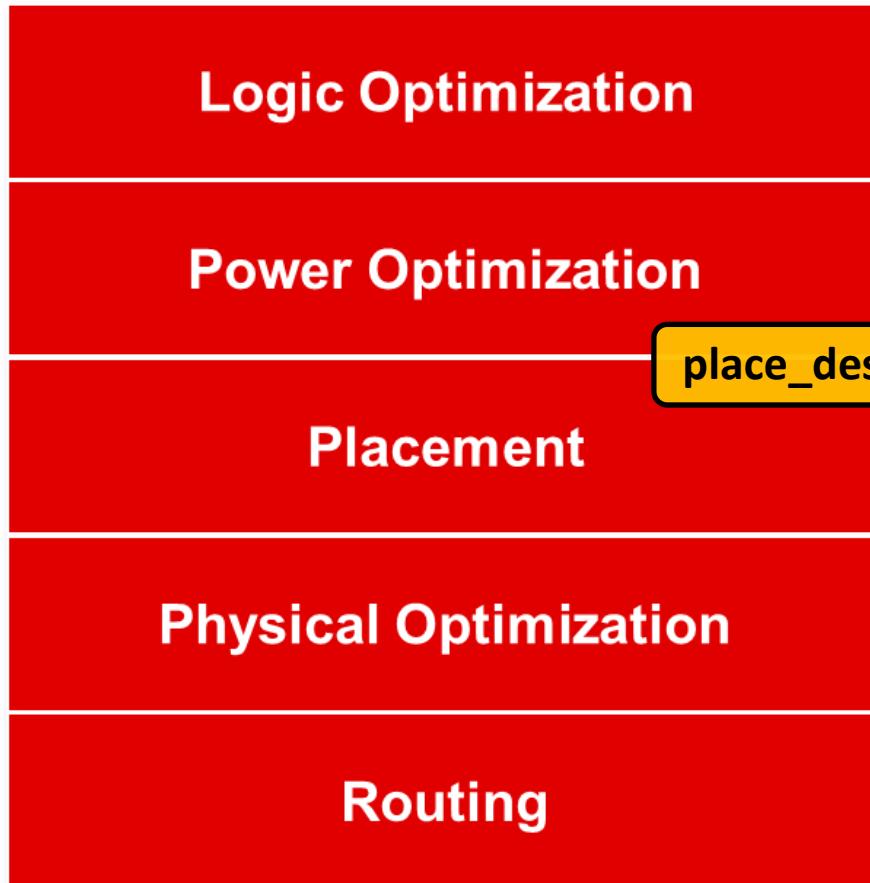
- 5 Processes but one is optional, and all will generate .dcp file



Optional, and decreasing the dynamic power consumption with not modifying the design of the clock and logic in FPGA. This can execute after `opt_design` or `place_design`.

# Vivado Implementation Processes

- 5 Processes but one is optional, and all will generate .dcp file

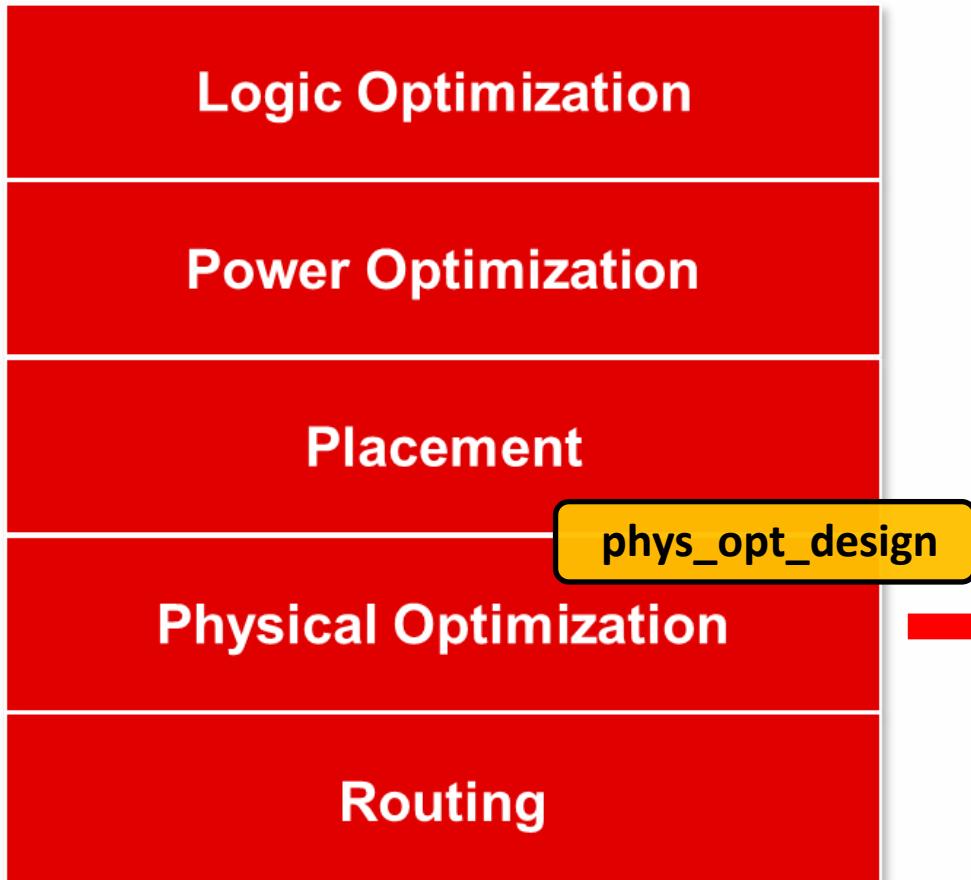


Vivado will run DRC before placement, and considering three points including Timing Slack, Wirelength and Congestion.

1. Clock and I/O cell will be placed before CLBs
2. Timing-Critical

# Vivado Implementation Processes

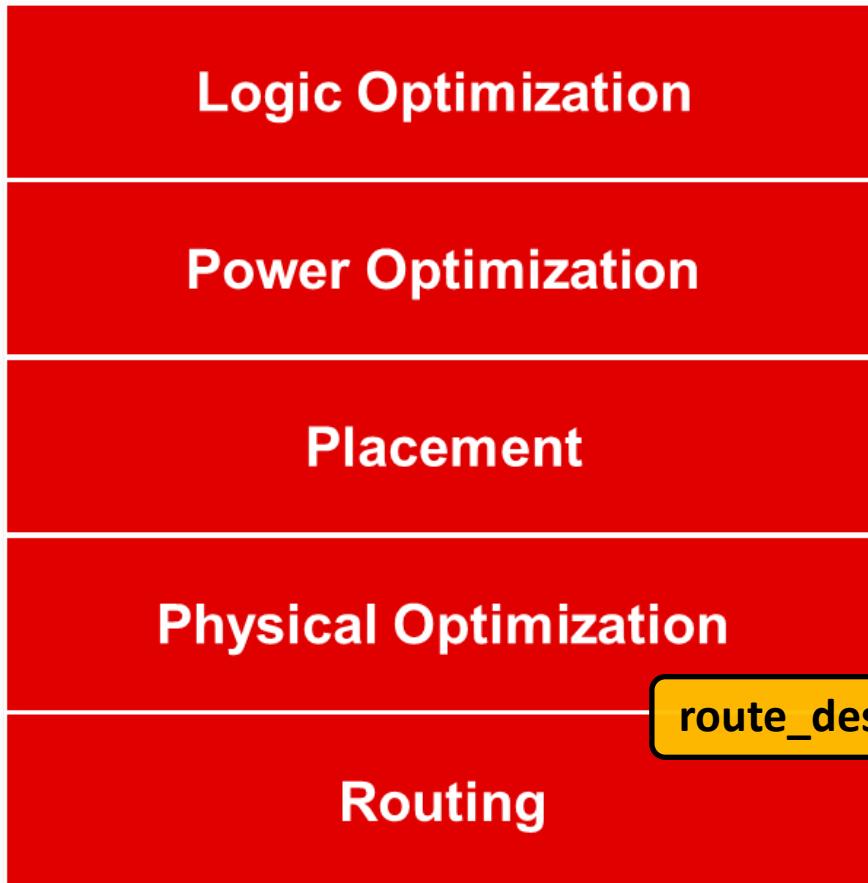
- 5 Processes but one is optional, and all will generate .dcp file



Available after doing `place_design` and `route_design`.  
For high-fanout networks it will consider copying the Register to reduce the high fanout.

# Vivado Implementation Processes

- 5 Processes but one is optional, and all will generate .dcp file



- **Normal**

Default mode. Compile only the modified parts without starting from scratch.

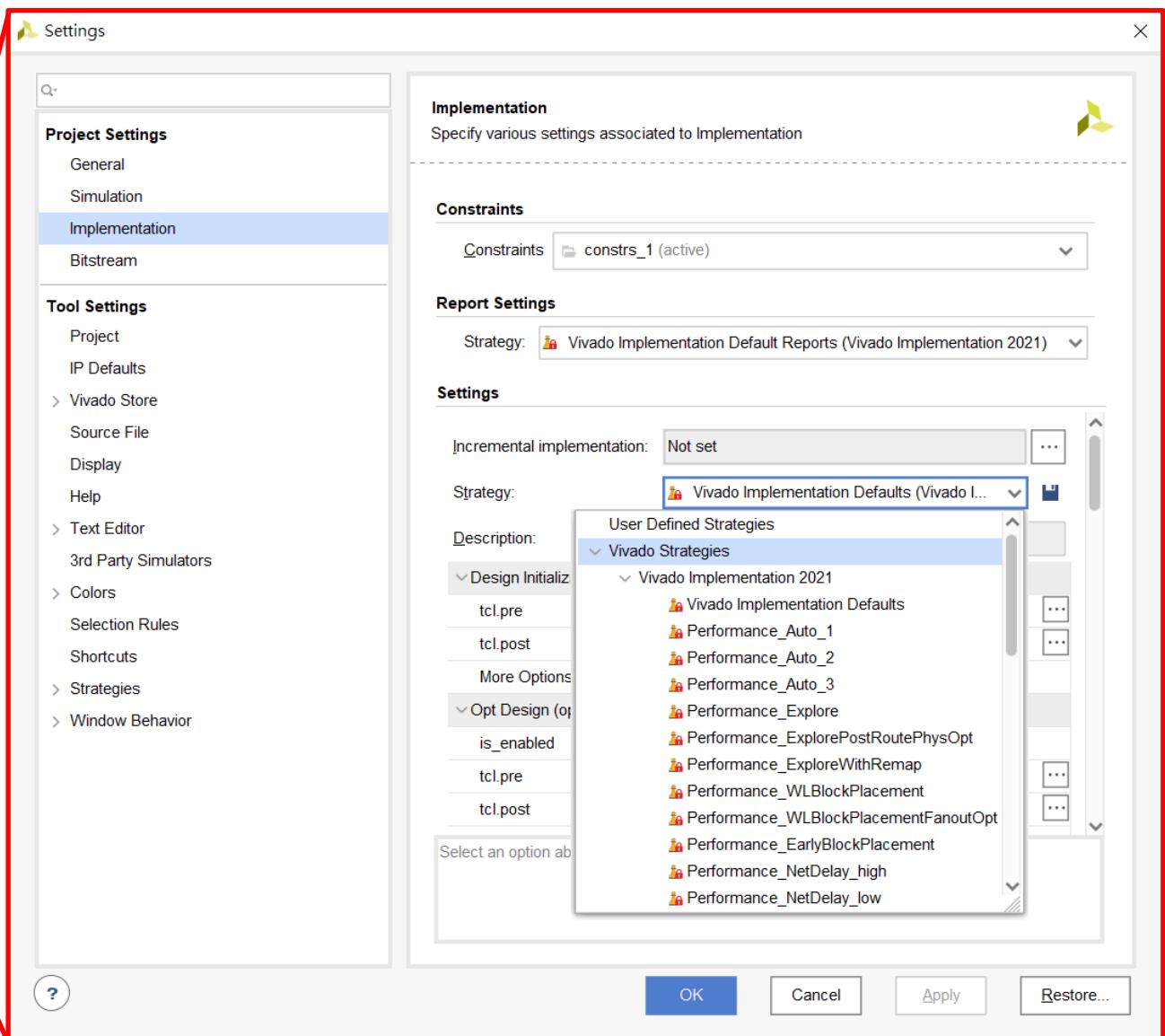
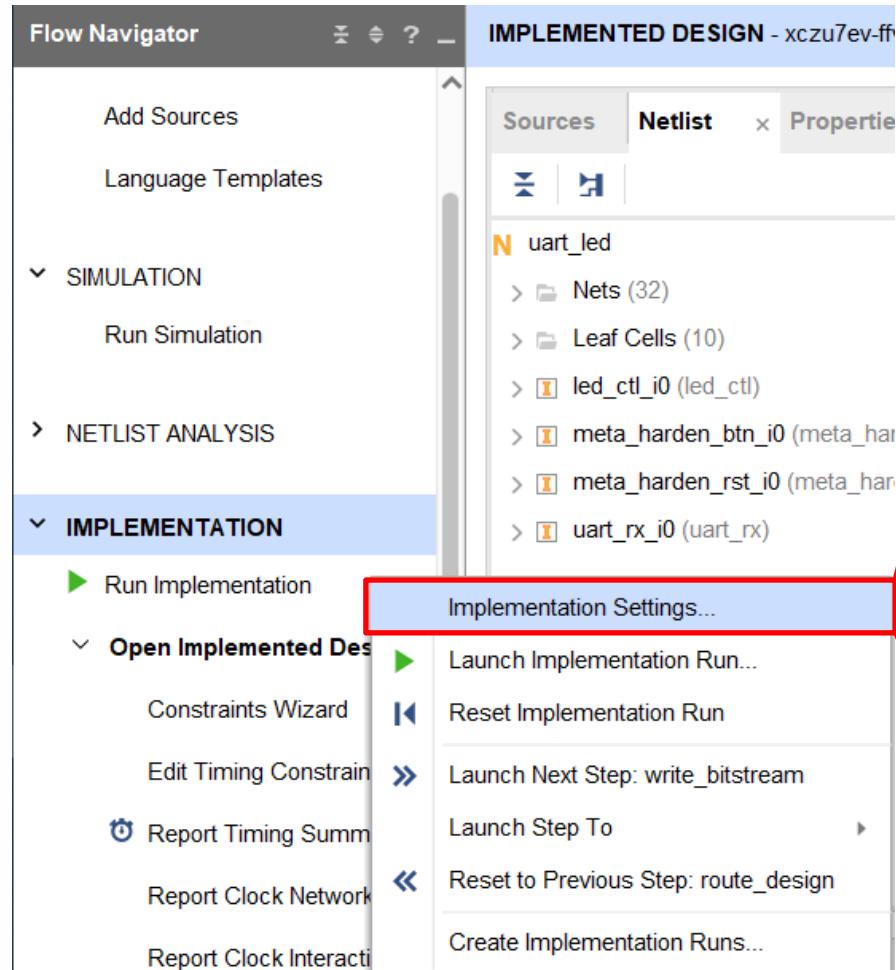
- **Re-Entrant**

Using this mode when running multiple Routing and there is an association between them.

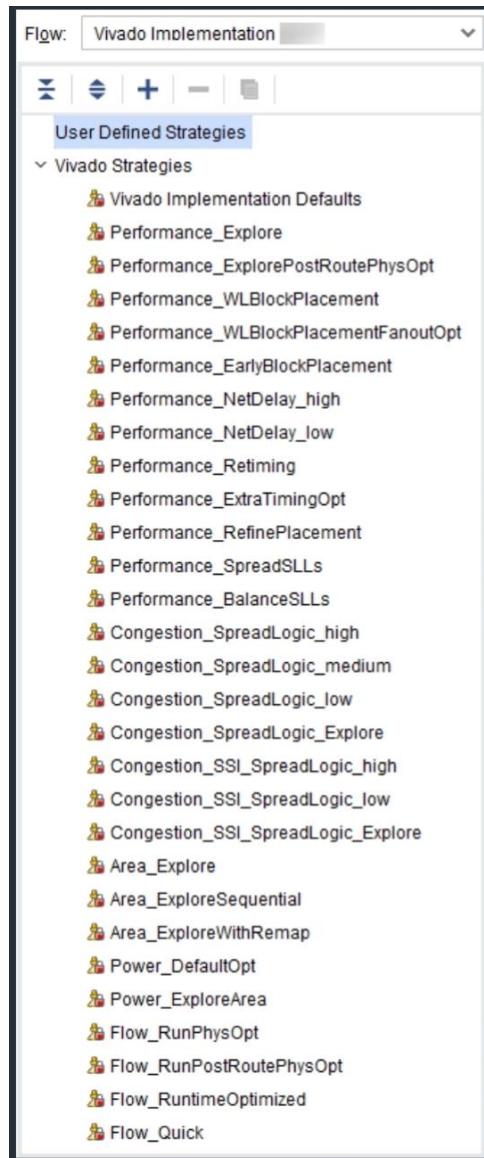


Normal and Re-Entrant mode

# Vivado Implementation Strategy



# Vivado Implementation Strategy

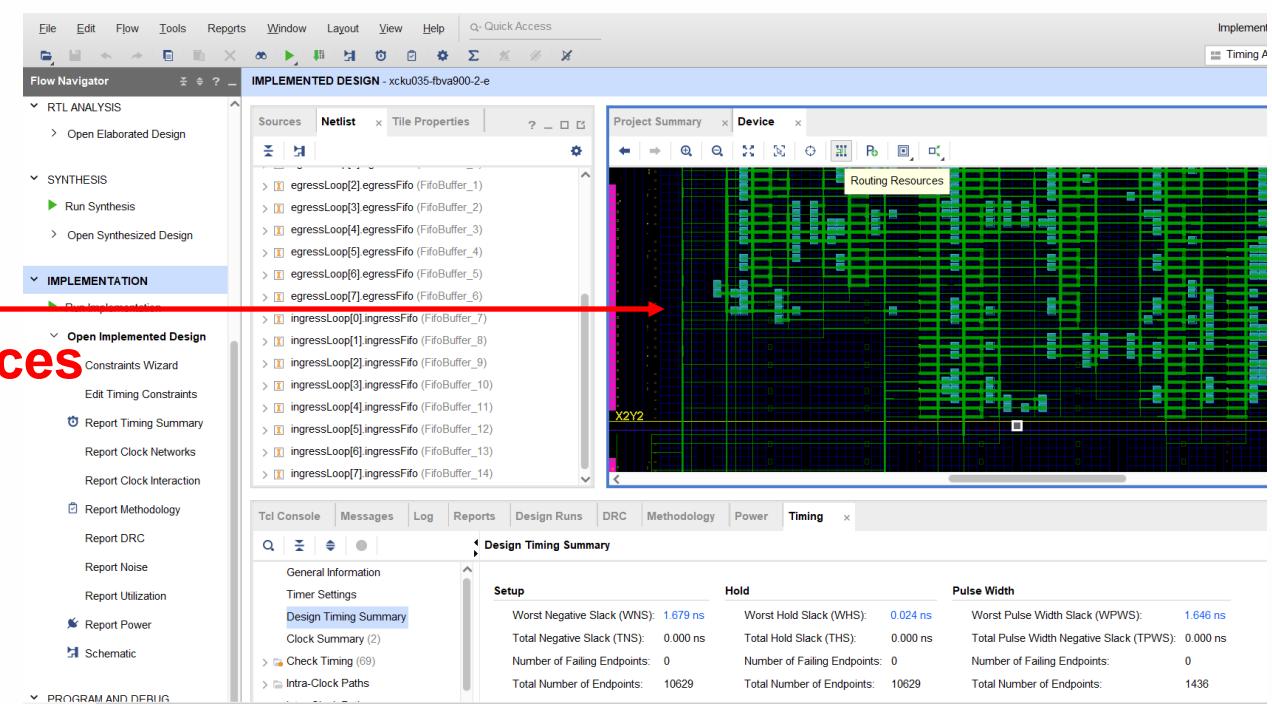
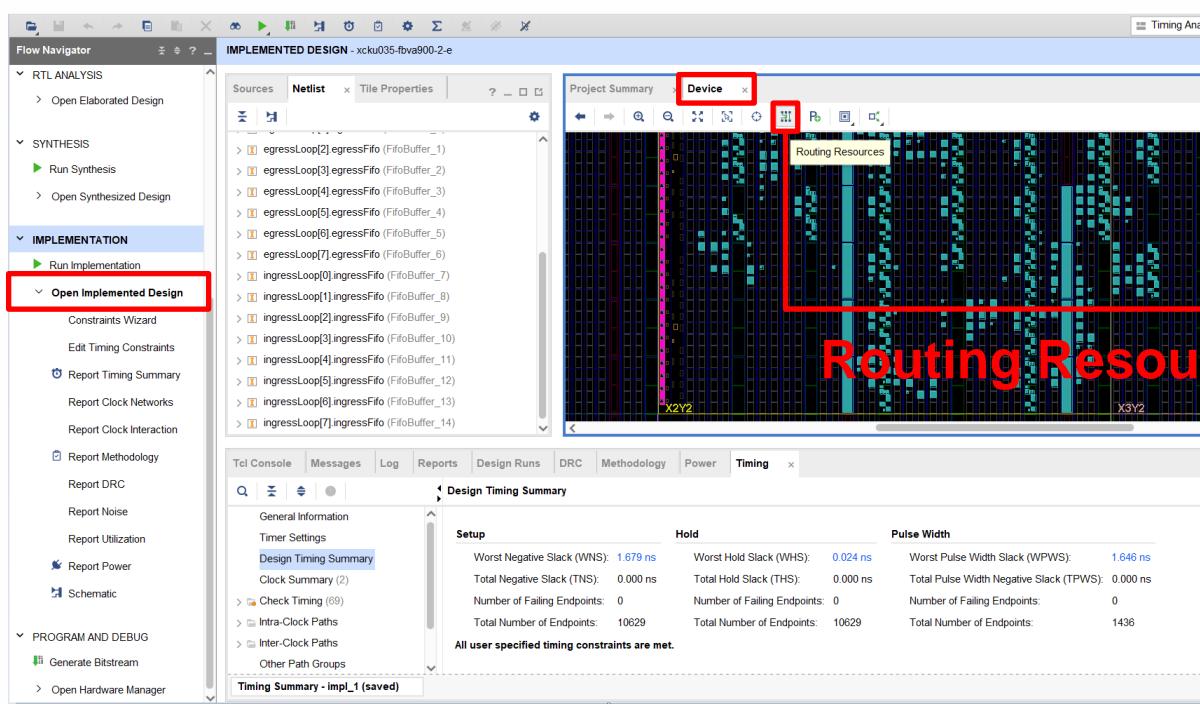


category	purpose
Performance	Increasing performance
Area	Optimize design to use minimal LUTs
Power	Optimize the power consumption
Flow	Optimize overall flows
Congestion	Decrease congestion

Global	Combined logic or Controller Unit too much
Long	BRAM, URAM or DSP too much
Short	MUXF or Carry Chain too much

# Vivado Implementation

- After Implementation
  - Sources and Netlist window will not change, but all resources have their own actual layout and routing.

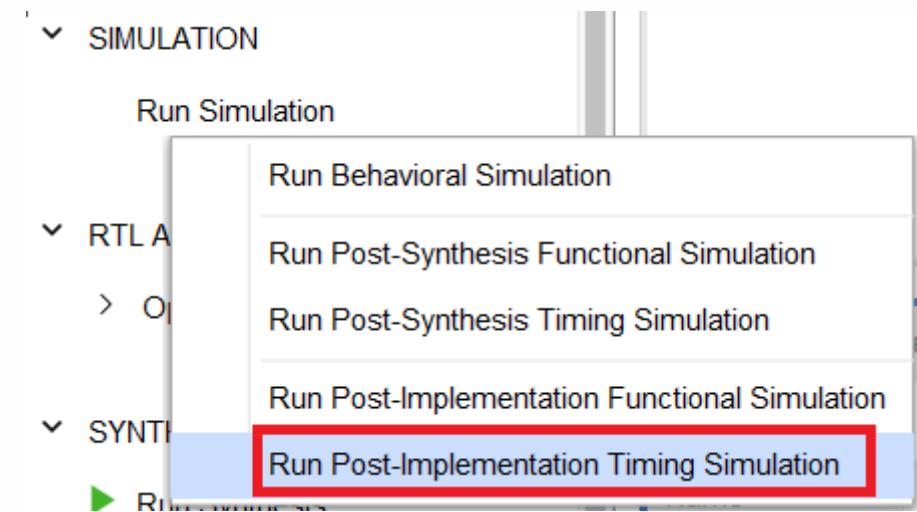


# Vivado Post-Implementation Timing Simulation

- > The timing simulation in previous steps is tent to be ideal. In the reality, signals transmit with delays.
- > Due to the wondering of the actual timing information, the running of post-implementation timing simulation is essential.

# Vivado Post-Implementation Timing Simulation

- > On the Flow Navigator, move to "Simulation" > "Run Simulation", select "Run Post-Implementation Timing Simulation"
- > After done running the simulation, the actual timing waveform presents.

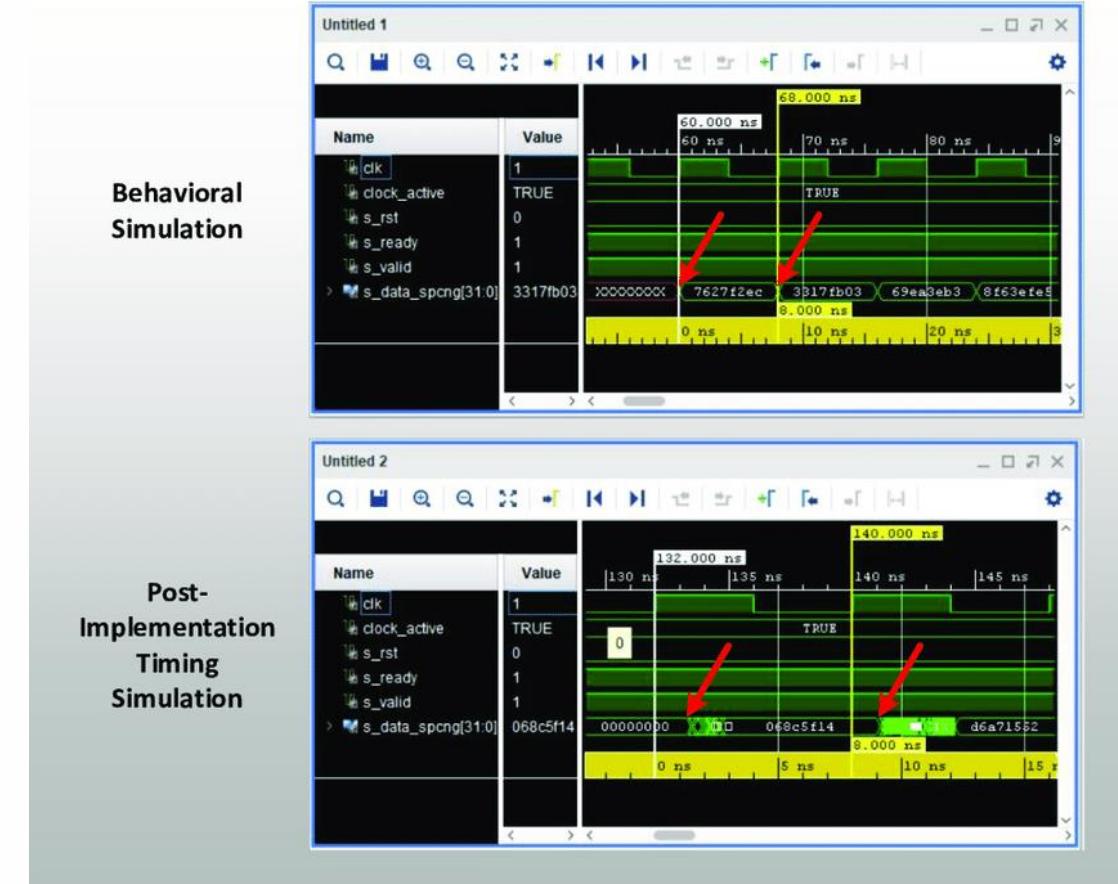


# Vivado Post-Implementation Timing Simulation

- Here is the comparison between behavioral simulation timing and post-implementation timing simulation.
- Between two waveforms, there is a visible tiny delay in the post-implementation timing comparing to the ideal timing simulation.

Behavioral  
Simulation

Post-  
Implementation  
Timing  
Simulation



# Vivado Design Checkpoint

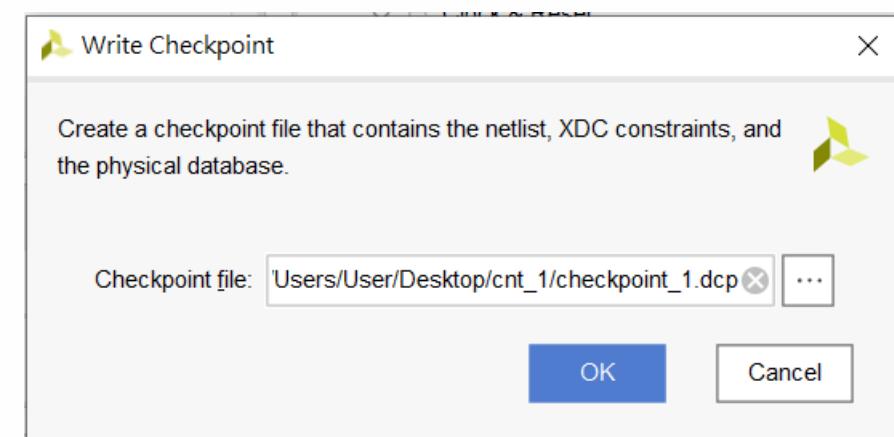
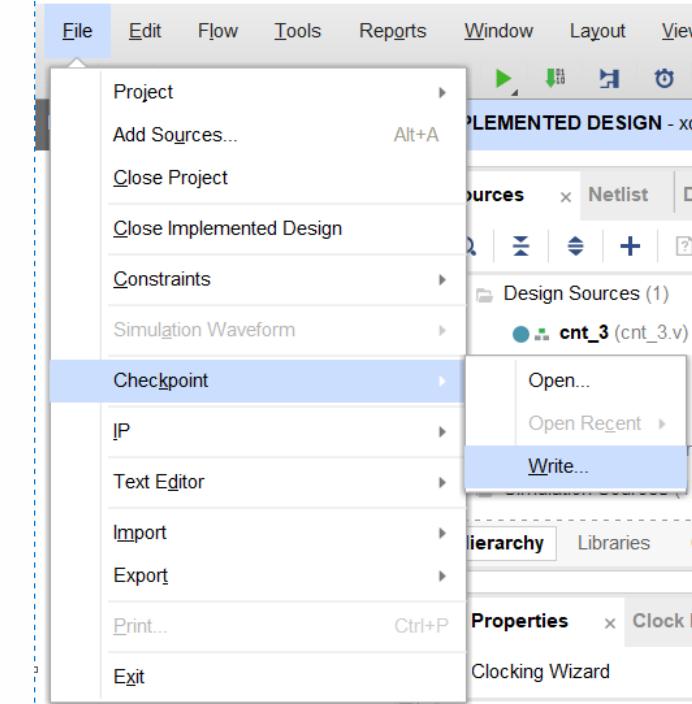


# Design Checkpoint

- > The Vivado Design Suite uses a physical design database to store placement and routing information.
- > Design checkpoint files (.dcp) allow you to save and restore this physical database at key points in the design flow.
- > Design ckeckpoint file includes:
  - Current netlist, including any optimizations made during implementation
  - Design constraints
  - Implementation results

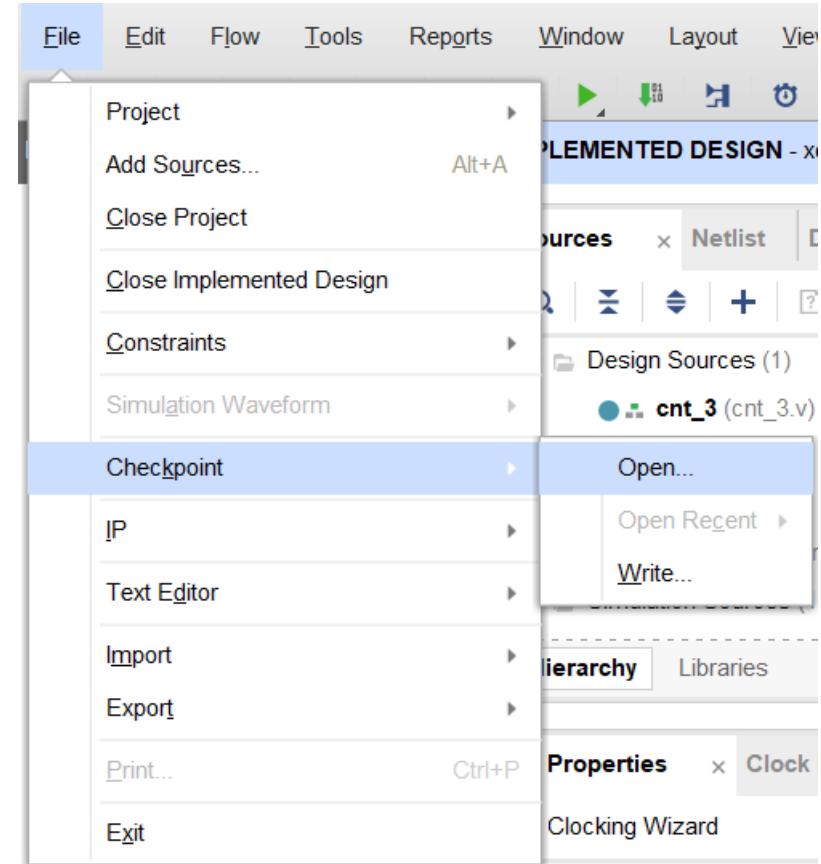
# Writing Checkpoint Files

- > Run “File” > “Checkpoint” > “Write” to capture a snapshot of the design database at any point in the flow.
- > Then the “Write Checkpoint” window pops. It shows where the checkpoint saves, and the directory is changeable.
- > This creates a file with a dcp extension.



# Reading Checkpoint Files

- > Run “File” > “Checkpoint” > “Open” to open the checkpoint in the Vivado Design Suite.
- > The design checkpoint is opened as a separate project.

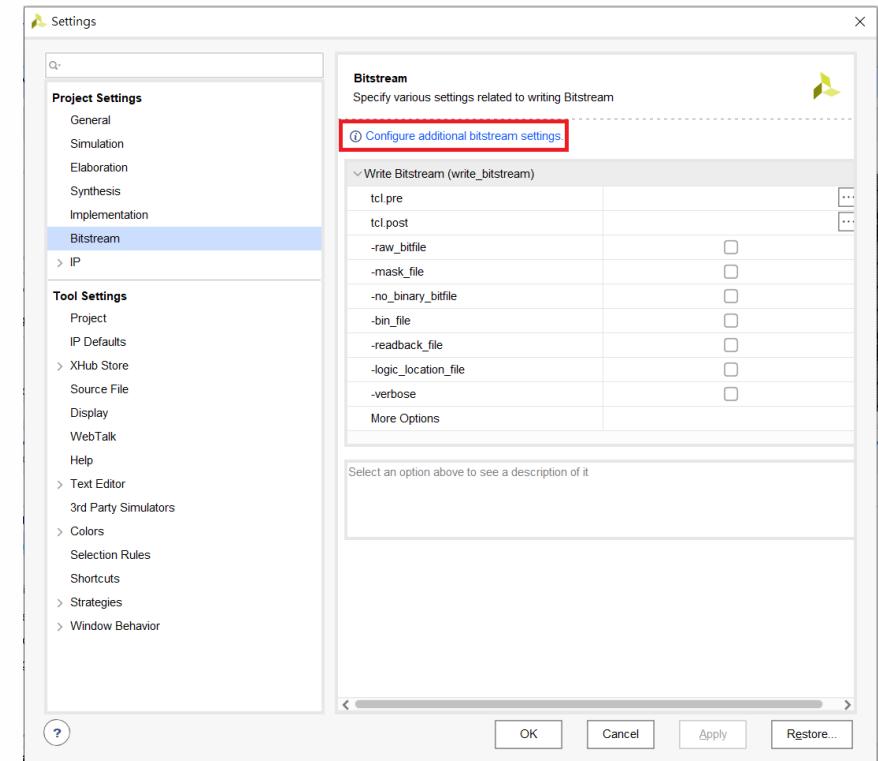
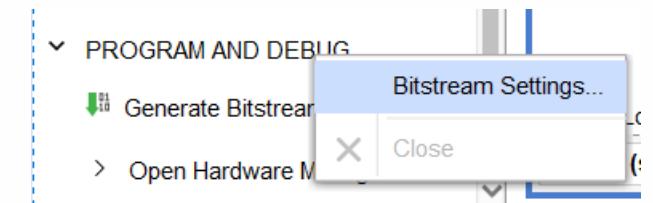


# Programming Devices



# Bitstream Settings

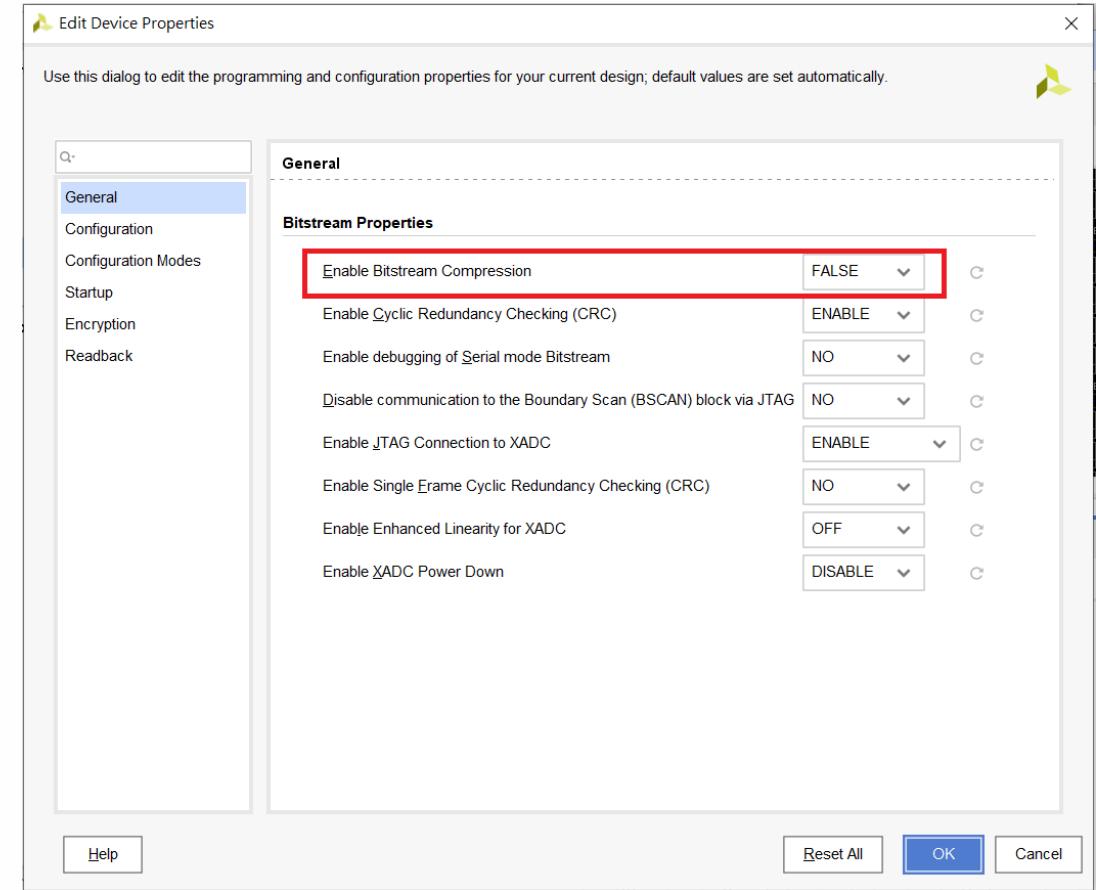
- > There are many options to be set before generating bitstream. Right click “Program and Debug”, select “Bitstream Settings”.



- > When “Settings” pops out, click “Configure additional bitstream settings”.

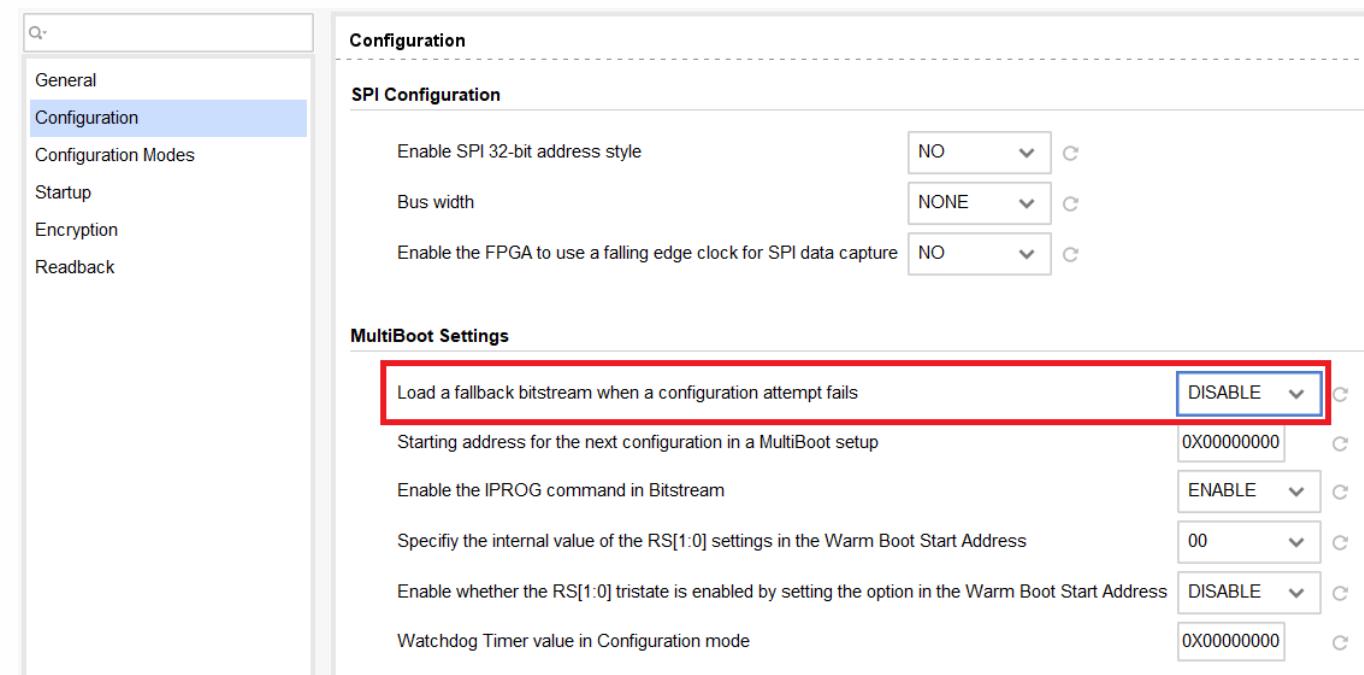
# Bitstream Settings

- > By clicking “Configure additional bitstream settings”, the “Edit Device Properties” window shows.
- > Some options are important and useful.  
In “General” > “Enable Bitstream Compress” compresses the bitstream files, make the whole download process faster.



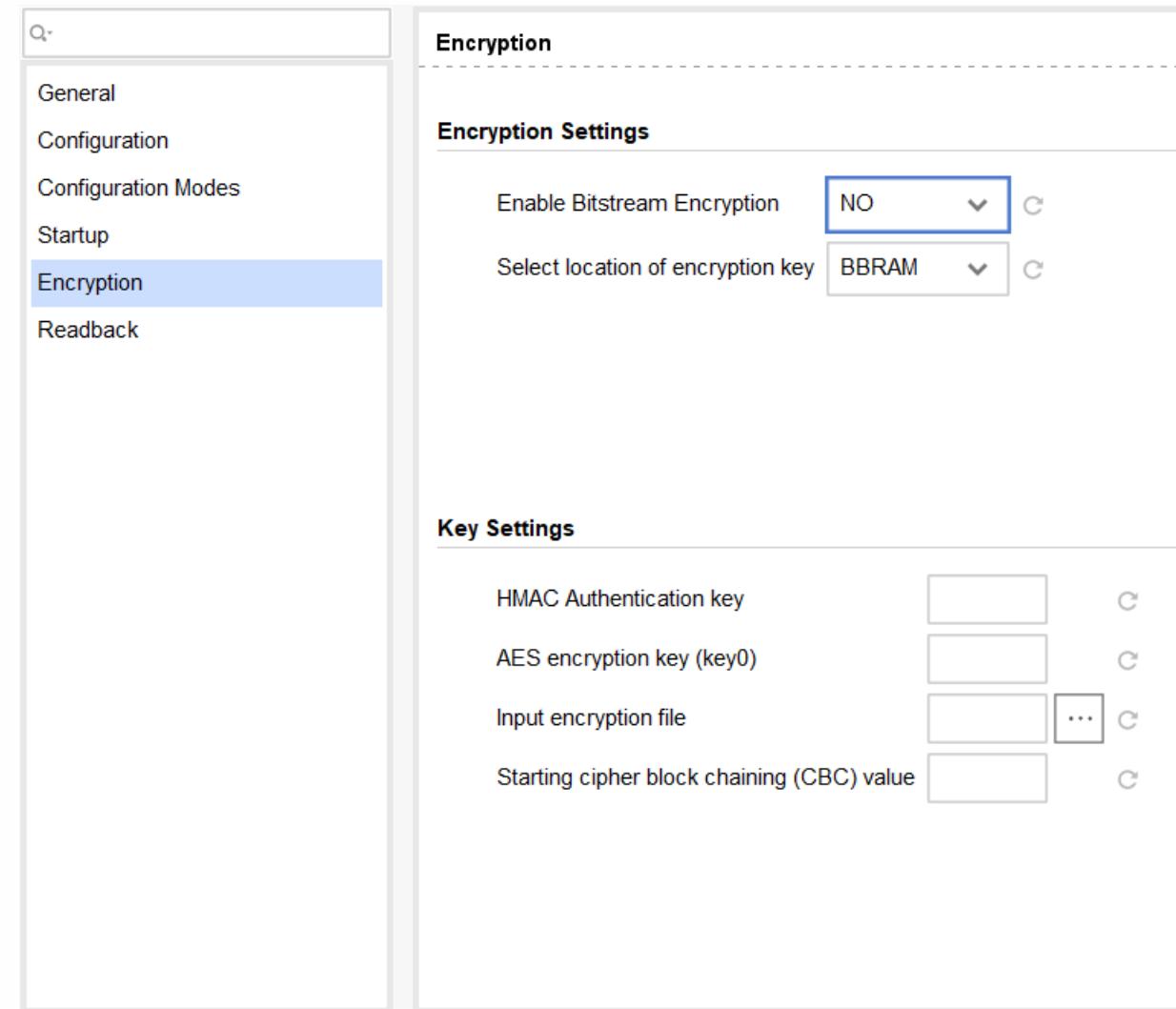
# Bitstream Settings

- > In “Configuration” > “MultiBoot Settings”, it can be set a fallback bitstream when a configuration attempt fails.



# Bitstream Settings

- > In “Encryption” > “Encryption Settings”, it can be set to enable bitstream encryption.
  
- > It can also be set to store the encryption key on the erasable battery-backed RAM(BBRAM) or one-time EFUSE.

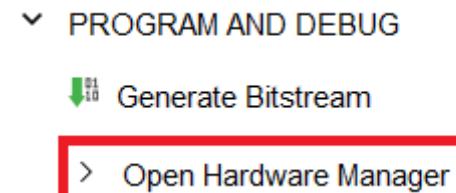


# Generate Bitstream/ Open Hardware Manager

> Click "Generate Bitstream" to create bitstream file

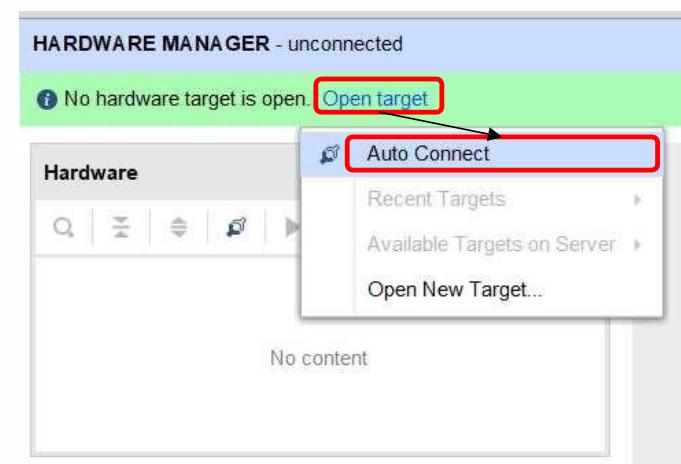


> Once the bitstream generation is done, click "Open Hardware Manager"



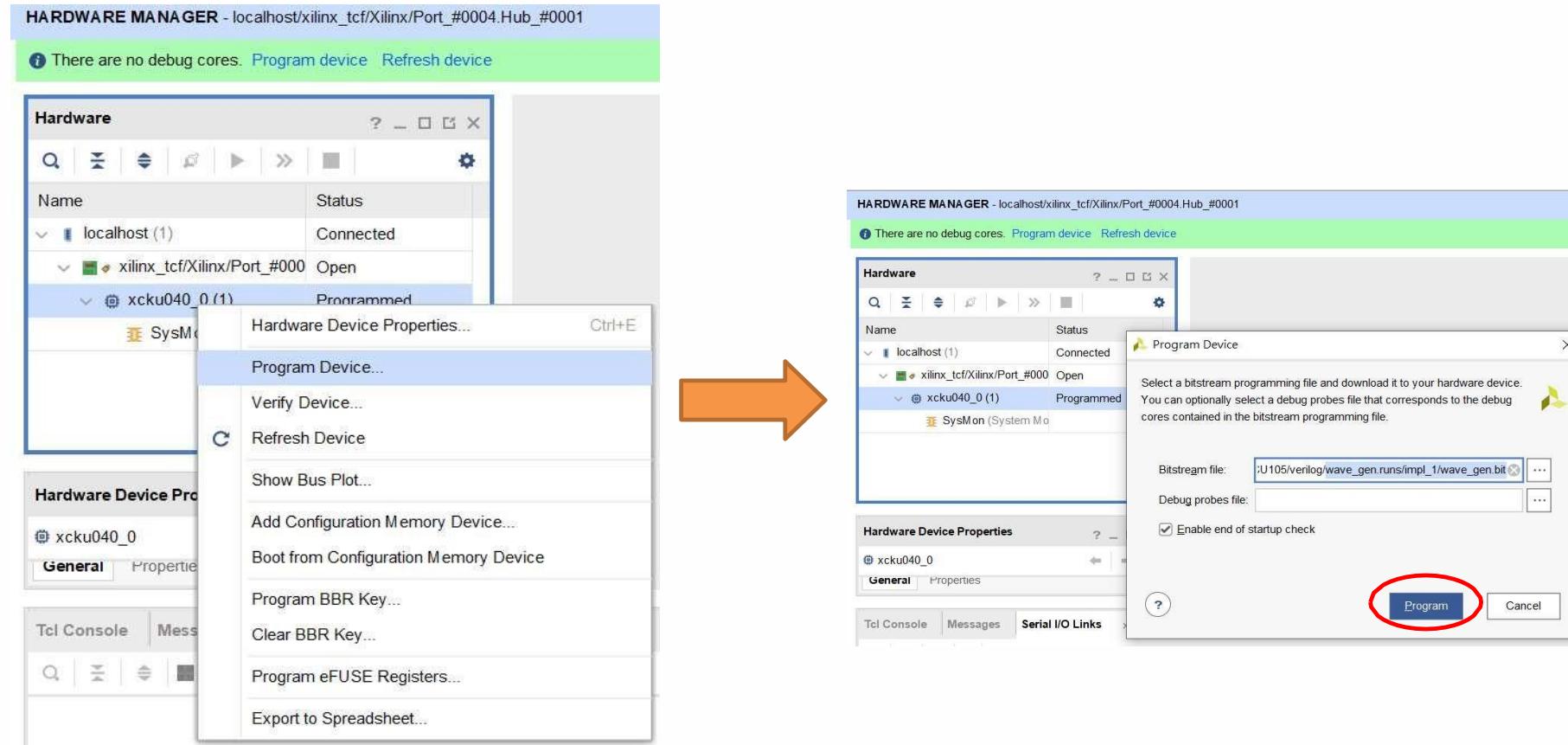
> Select "Open target", then click "Auto Connect"

> Make sure the device is connected to PC that runs Vivado, and then the information bar suggest that open a new target or an existing one.



# Programming Device

- > Right-click “the connected device” and select "Program Device".

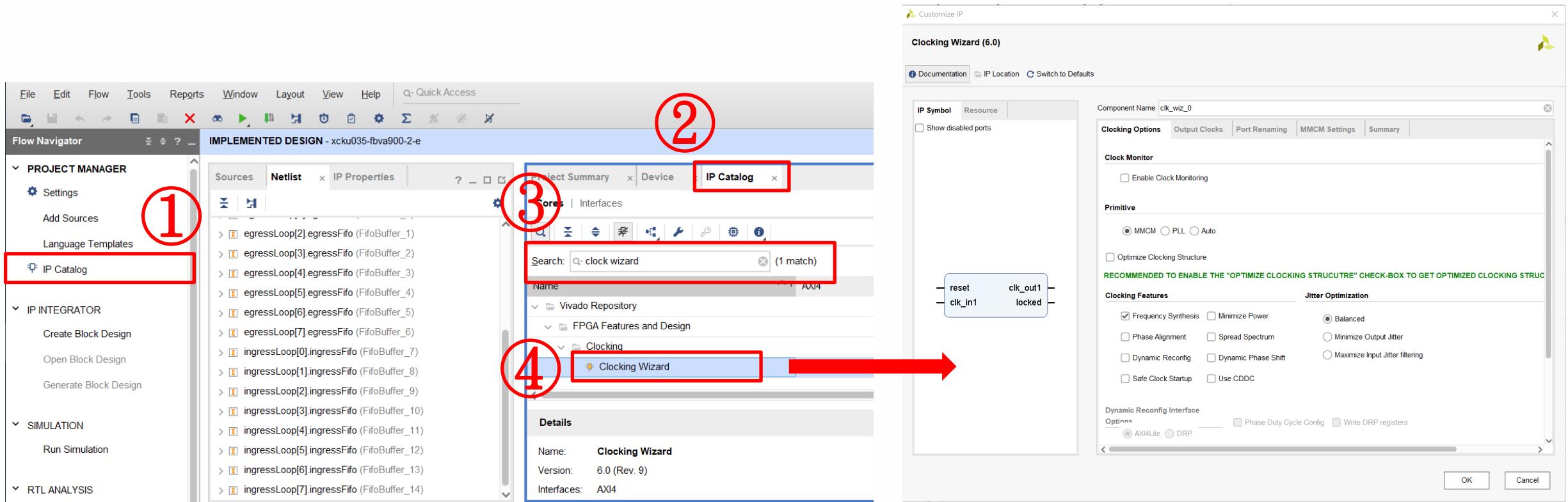


# Vivado IP flow



# Vivado IP flow

- Intellectual Property (IP) can be seen as a library in program language (C, C++, ... etc.) or a module that has been designed and verified.
- Add IP
  - IP catalog → Search → Double Click it → the window of Custom IP pops out



# Vivado IP flow

- Once IP has been created, some associated IP files will appear.

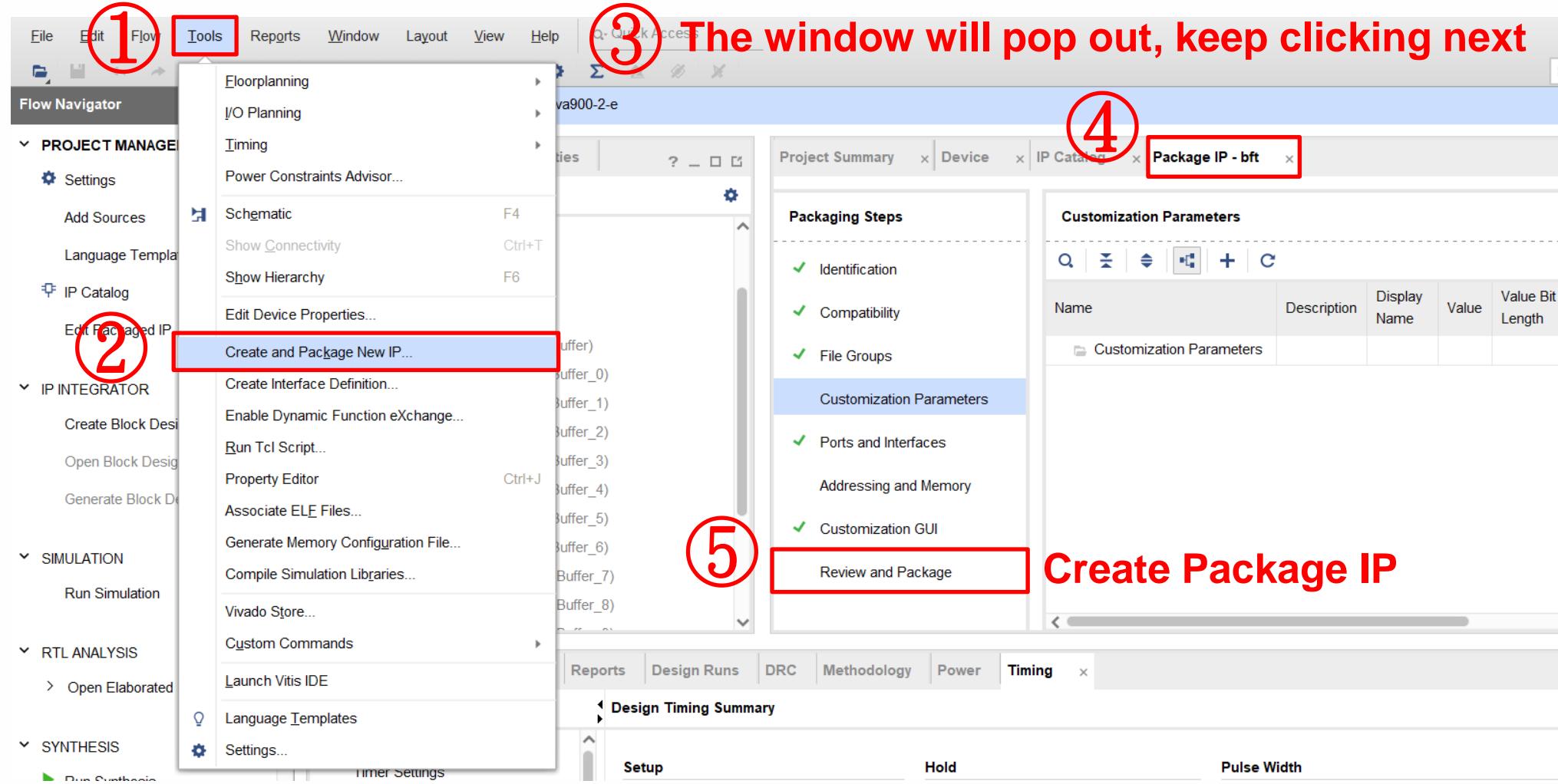
A design checkpoint (.dcp)	An instantiation template	Synthesis files	Simulation files
An IP can deliver a number of XDC files (prepended with IP name)			
File	Description		
<ip_name>.xml file	IP configuration data		
<ip_name>.xci	IP customization information		
<ip_element 1> .. <ip_element n>	Sub-directories corresponding to the sub-blocks that make the IP		
<ip_name>.ex.tcl	Tcl script for the given IP		
<ip_name>.vho, .veo	Instantiation templates		

# Vivado IP Synthesis

- Out of Context (OOC)
    1. The default way of IP synthesis, this will create some files including .dcp, .xci, changelog,... etc.
    2. If a certain IP core not be modified after synthesizing, only synthesize other user logic which is modified. This way will reduce synthesis time massively.
  - Global
    - IP will be synthesized with all user logic.
- ◆ The difference between the two ways only on the speed of synthesis, the optimization is a minor difference.

# Vivado IP Packager

- Custom IP can be exported that can be reused by others according to IEEE-1685.
- Tools → Create and Package New IP → Click Next to the window of Package IP -bft pops out



# Vivado Logic Analyzer



# Vivado Logic Analyzer

- Main purpose
  1. Verification
  2. Debugging
  3. Data Capture

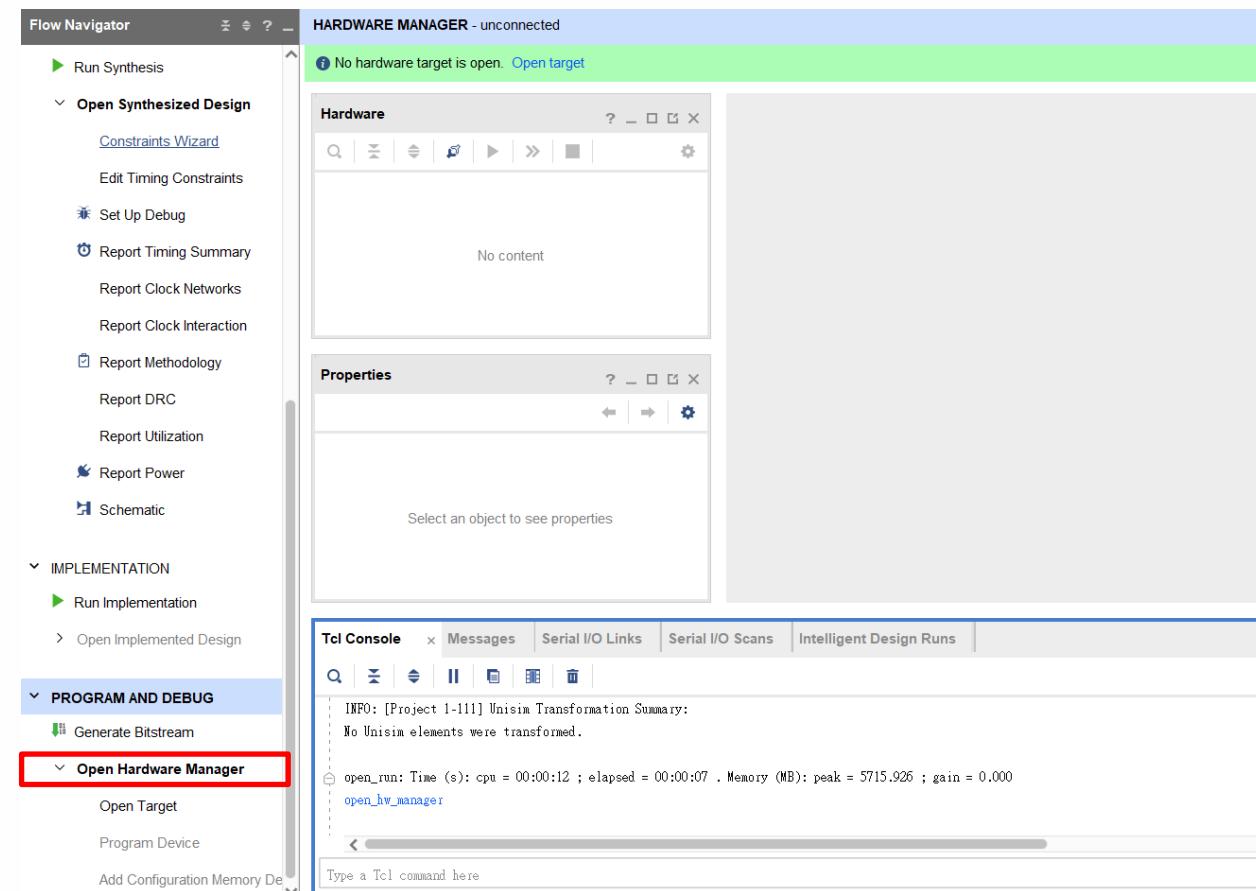
- Contains two main parts

## 1. Debug Cores

Integrated Logic Analysis (ILA), System ILA, VIO ... etc.

## 2. Debug Flow

HDL instantiation and netlist insertion



# Vivado Logic Analyzer

- Debug Flow

## HDL Instantiation Flow

The HDL instantiation flow gives the user the **flexibility** to connect **to any signal** in the design

The user has to **modify the HDL code in order to instantiate the cores** in the design

VS.

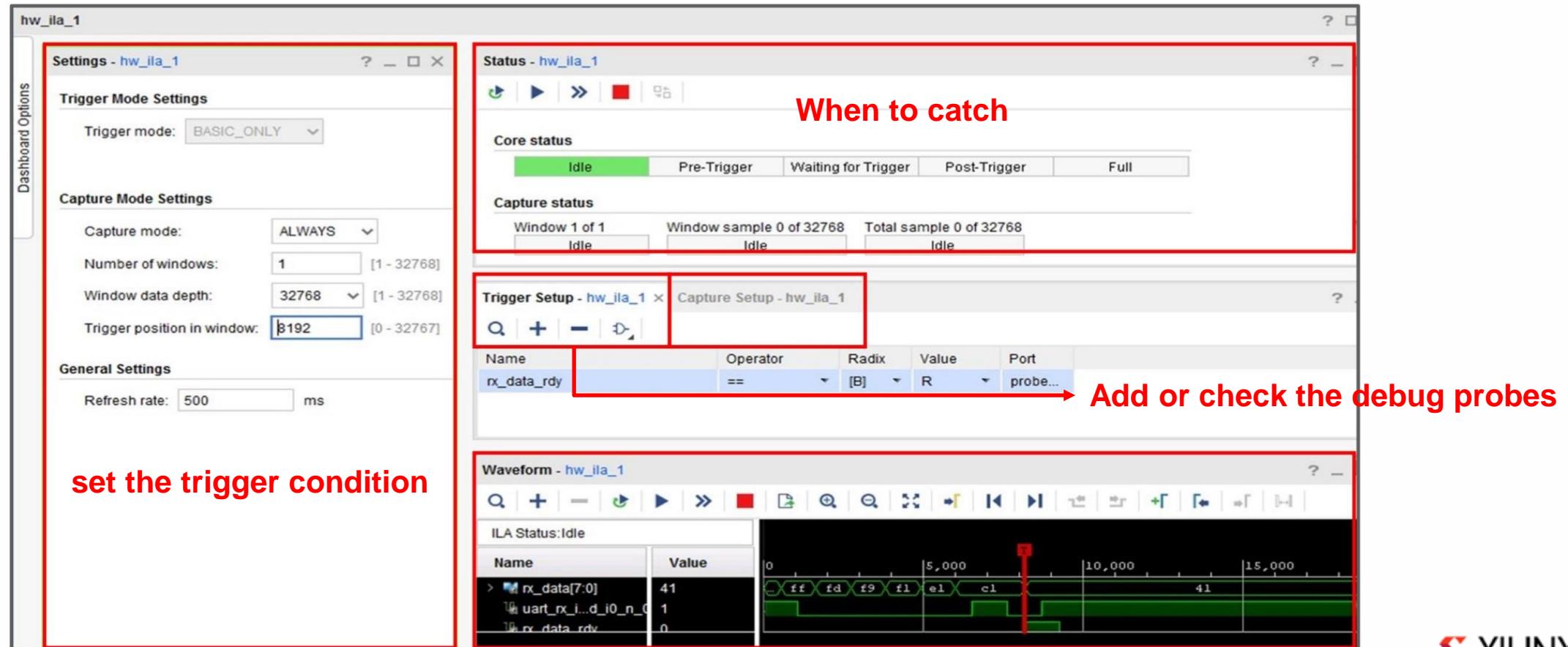
## Netlist Insertion Flow

The netlist insertion flow is **easier to add and remove the debug core** and is the more recommended flow

The Vivado tool automatically inserts the debug core into the post-synthesis netlist

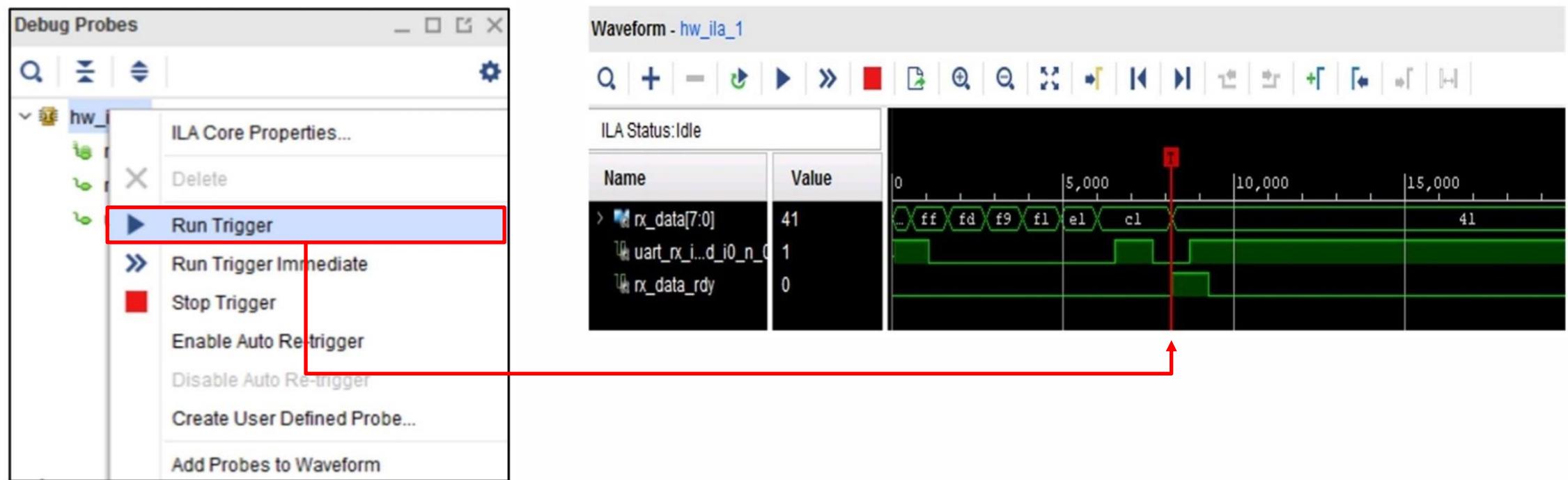
# Vivado Logic Analyzer

- Triggering
  - The way of debug. We can check the situation of a certain IP by setting the trigger condition.
  - IP catalog offers ILA cores that can be set the trigger condition.



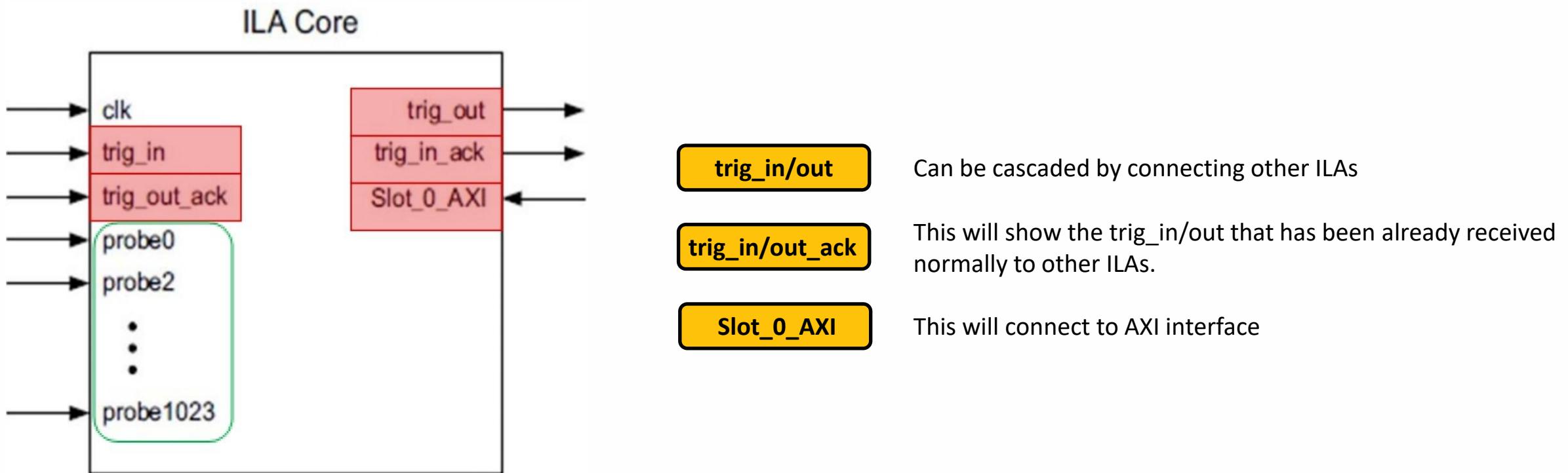
# Vivado Logic Analyzer

- Triggering
  - The way of debug. We can check the situation of a certain IP by setting the trigger condition.
  - IP catalog offers ILA cores that can be set the trigger condition.



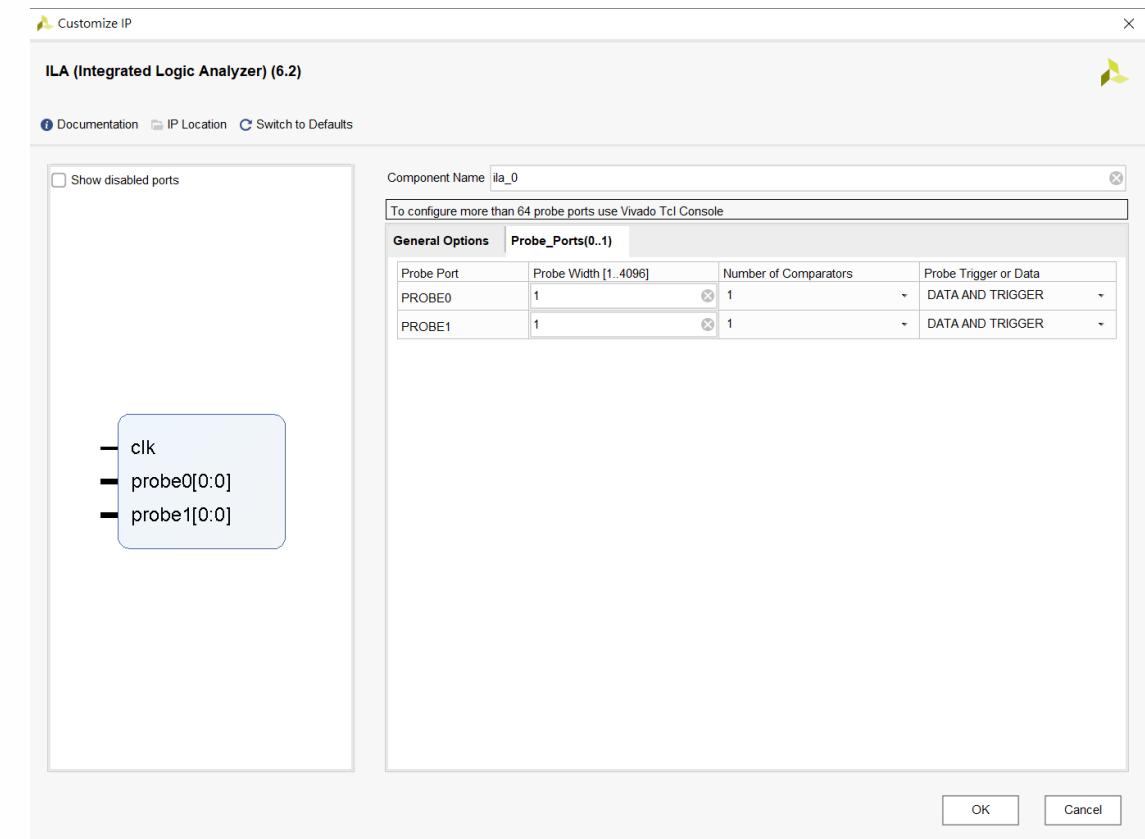
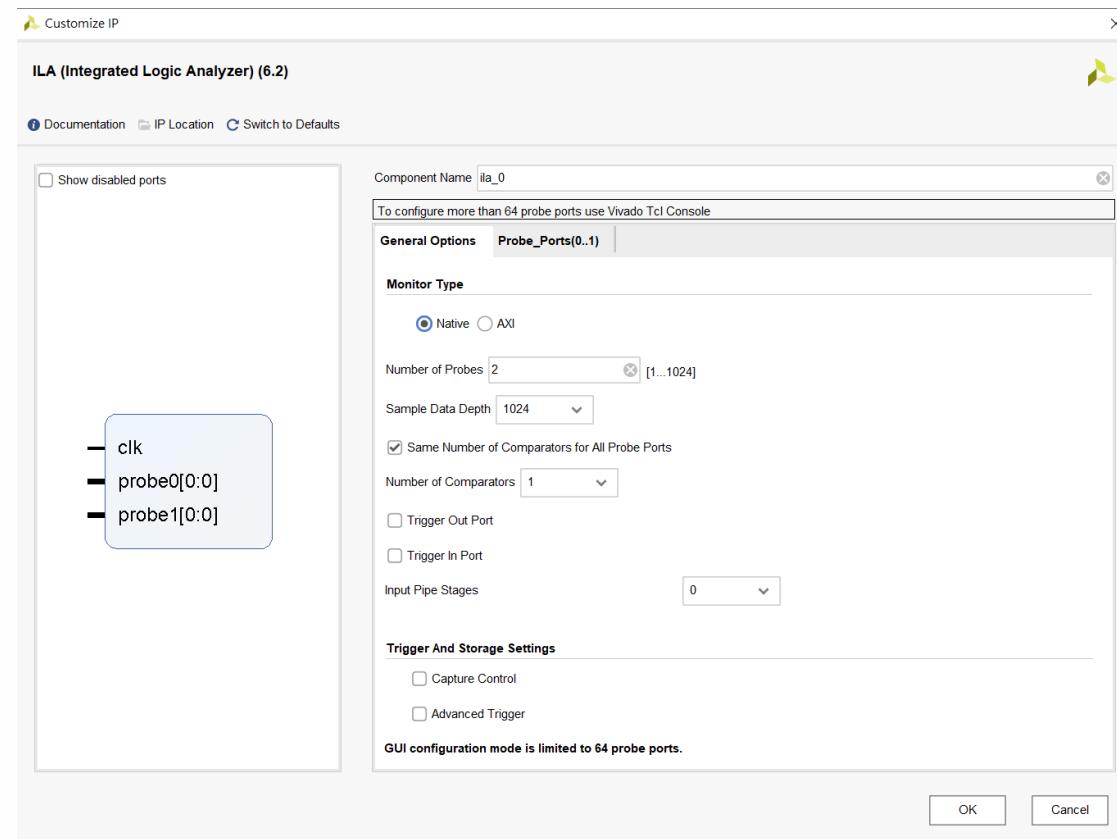
# Vivado Logic Analyzer

- Integrated Logic Analyzer (ILA)
  1. Monitor the waveform of signals which you want to debug
  2. Monitor type can be set two interfaces: **Native** and **AXI**, **Communication is JTAG**



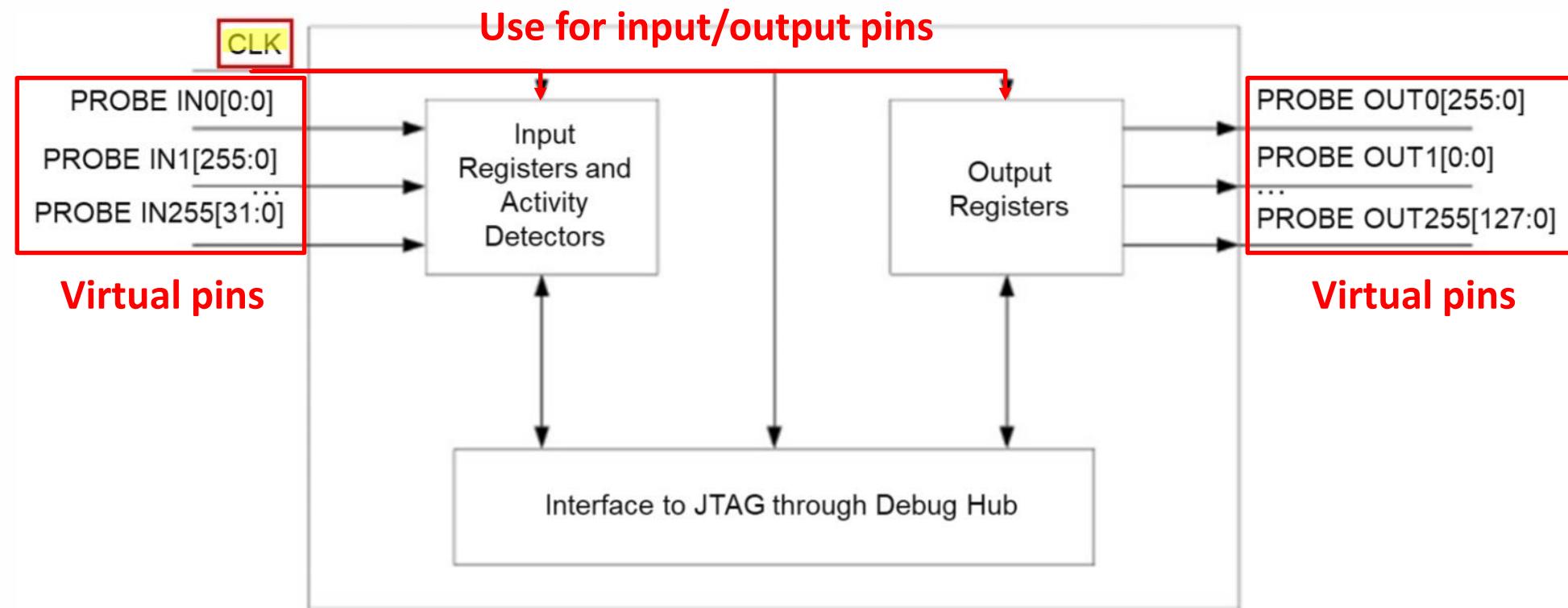
# Vivado Logic Analyzer

- Integrated Logic Analyzer (ILA)
  1. Monitor the waveform of signals which you want to debug
  2. Monitor type can be set two interfaces: **Native** and **AXI**, Communication is JTAG



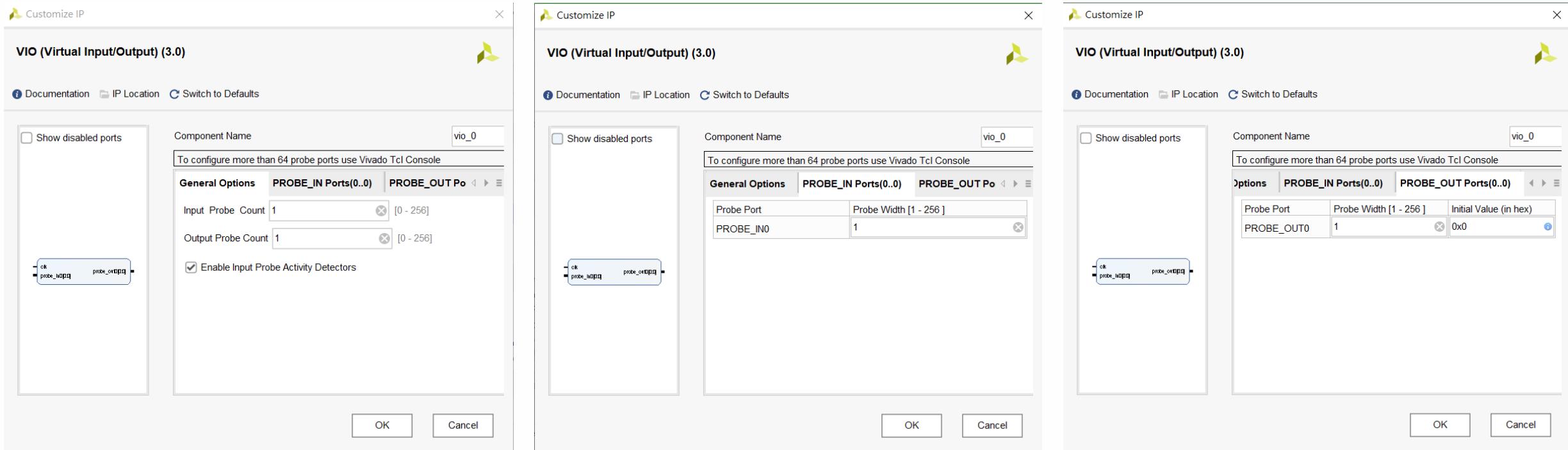
# Vivado Logic Analyzer

- Virtual Input/Output (VIO)
  1. Can monitor and drive the internal signals of FPGA in real time
  2. Not as ILA, VIO doesn't need BRAM



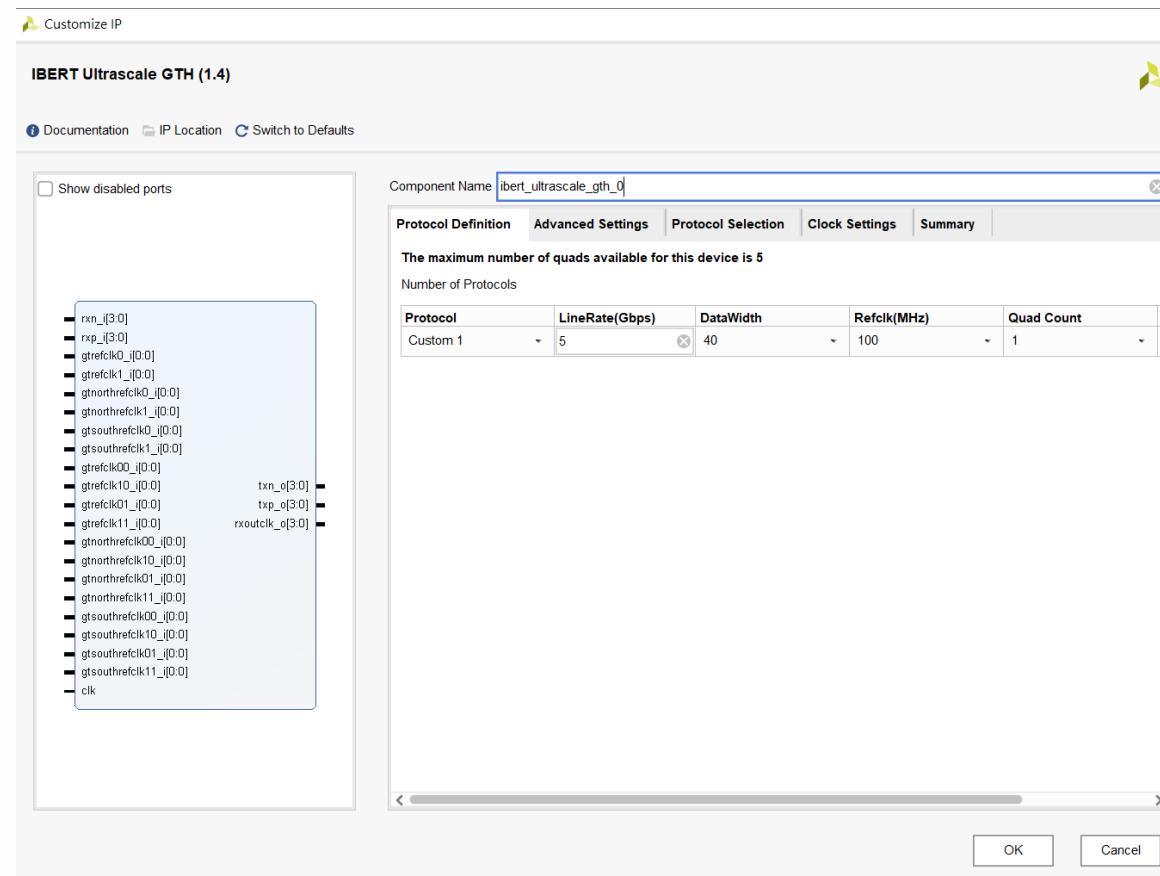
# Vivado Logic Analyzer

- Virtual Input/Output (VIO)
  1. Can monitor and drive the internal signals of FPGA in real time
  2. Not as ILA, VIO doesn't need BRAM



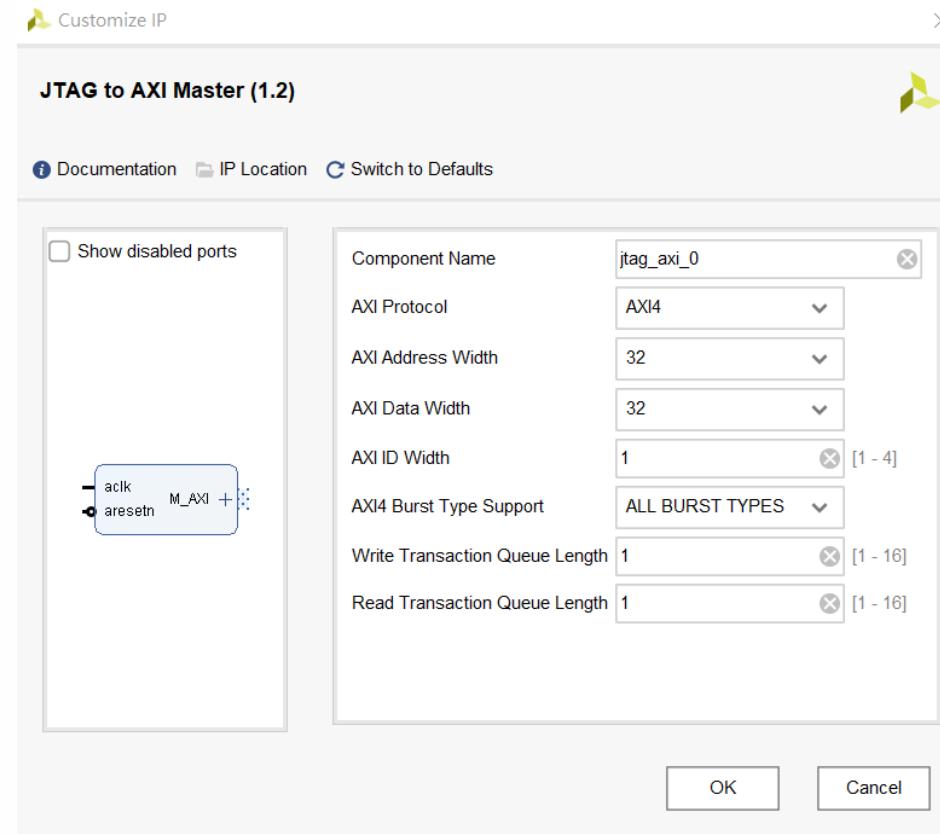
# Vivado Logic Analyzer

- Integrated Bit Error Ratio Tester (IBERT)
  1. Can test signals on transceiver
  2. Can analyze the bit-error ratio on MGT/GTP/GTX/GTH channels



# Vivado Logic Analyzer

- JTAG to AXI Master
  - 1. Describe the way of data transmission between Master and Slave device
  - 2. This IP can be AXI Master to drive AXI Slave
  - 3. Support AXI and AXI-Lite

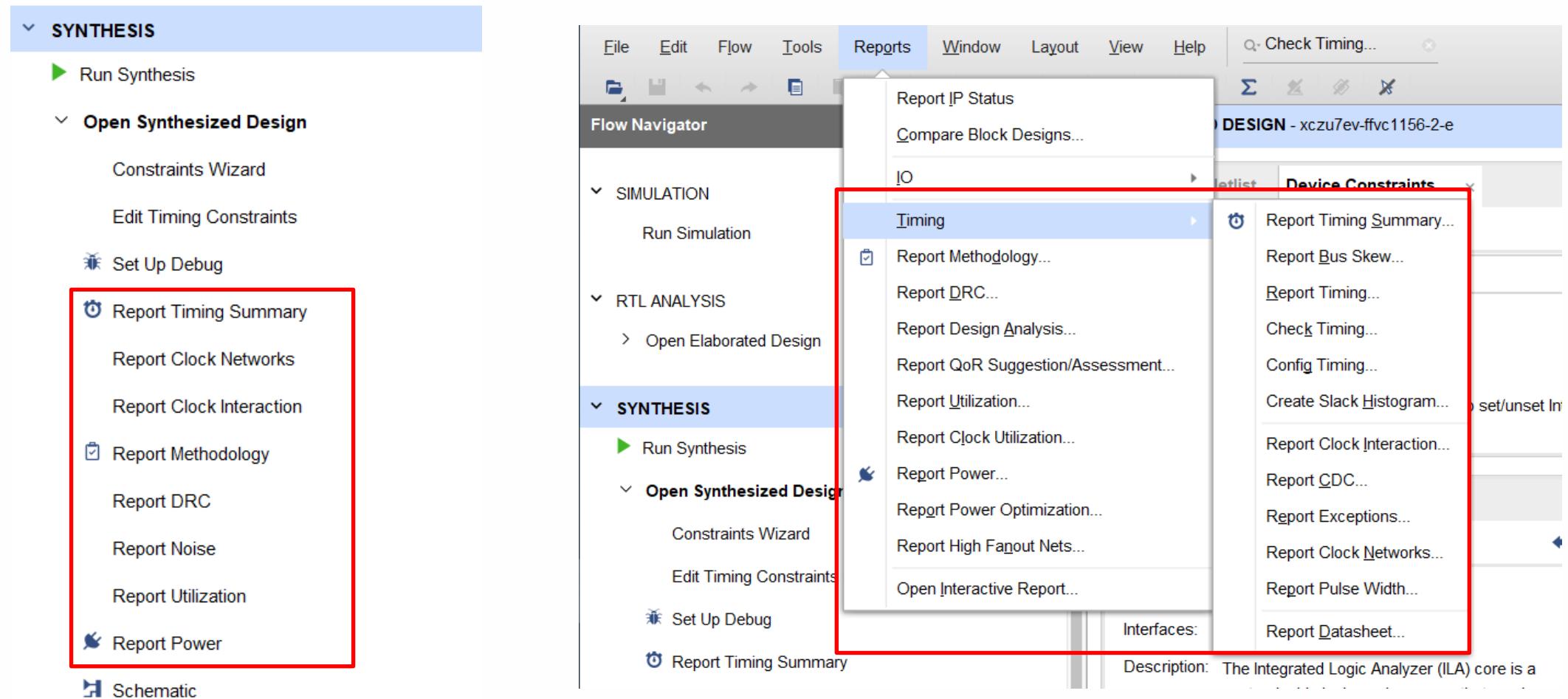


# Vivado Reports



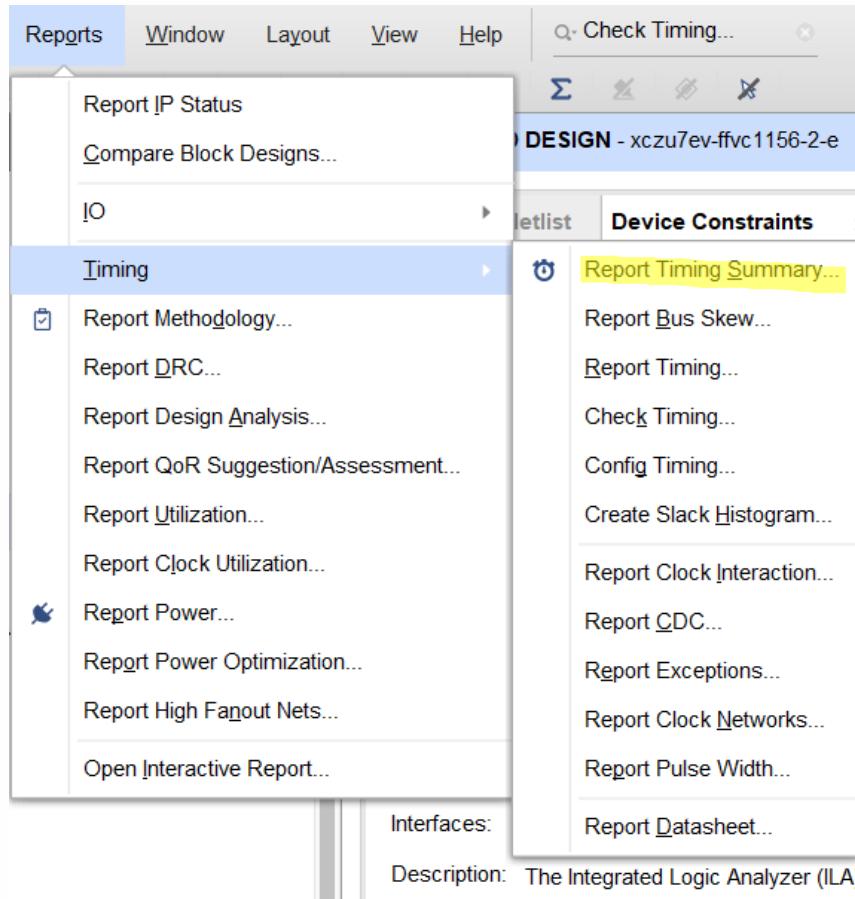
# Vivado Reports

- Vivado can generate various reports at different stages (Synthesis, Implement,... etc.)



# Vivado Reports

- Vivado can generate various reports at different stages (Synthesis, Implement,... etc.)



- Report Timing Summary

- This will be generated automatically after synthesis or implementation.
- Synthesis
  - ◆ Calculate net delays, according to connectivity and fanout
- Implementation
  - ◆ According to the actual routing information

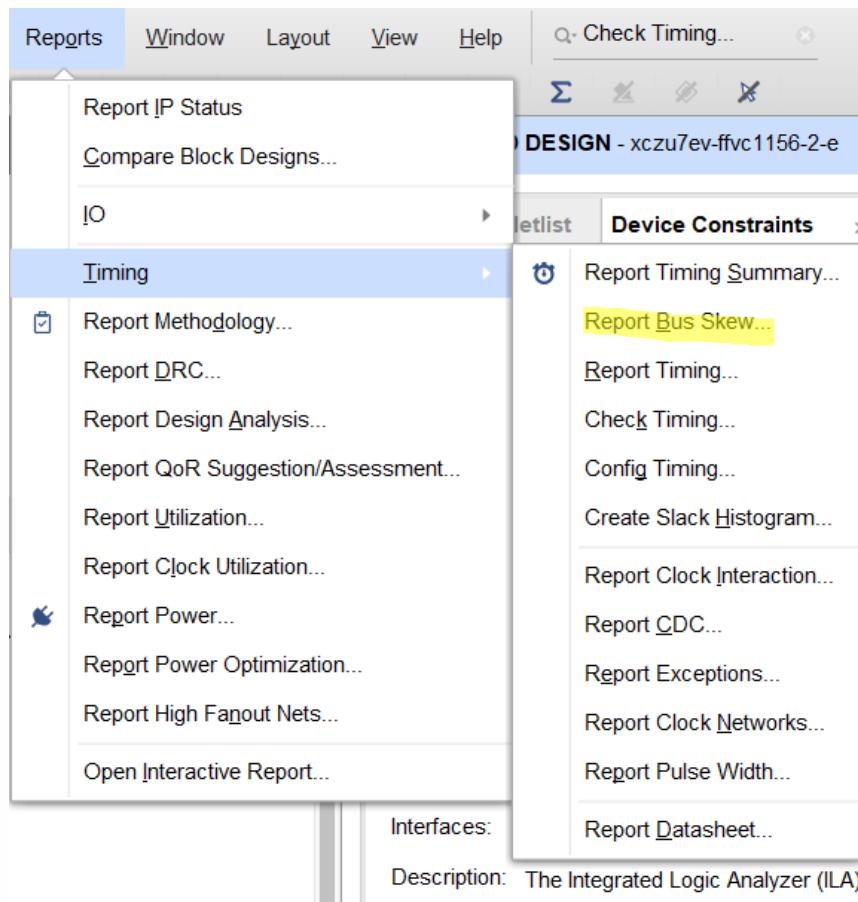
The screenshot shows the "Design Timing Summary" report. The top navigation bar has tabs for Not Sync'd, Messages, Log, Reports, Design Runs, Timing (which is selected and highlighted in red), Power, Methodology, DRC, Package Pins, I/O Ports, and Help. The main content area is titled "Design Timing Summary". It contains sections for General Information, Timer Settings, Design Timing Summary, Clock Summary (1), Check Timing (5), Intra-Clock Paths, Inter-Clock Paths, Other Path Groups, User Ignored Paths, and Unconstrained Paths. Below these sections is a table with three columns: Setup, Hold, and Pulse Width. The table data is as follows:

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 8.213 ns	Worst Hold Slack (WHS): 0.251 ns	Worst Pulse Width Slack (WPWS): 4.500 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWNS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 4	Total Number of Endpoints: 4	Total Number of Endpoints: 5

All user specified timing constraints are met.

# Vivado Reports

- Vivado can generate various reports at different stages (Synthesis, Implement,... etc.)



- Report Bus Skew

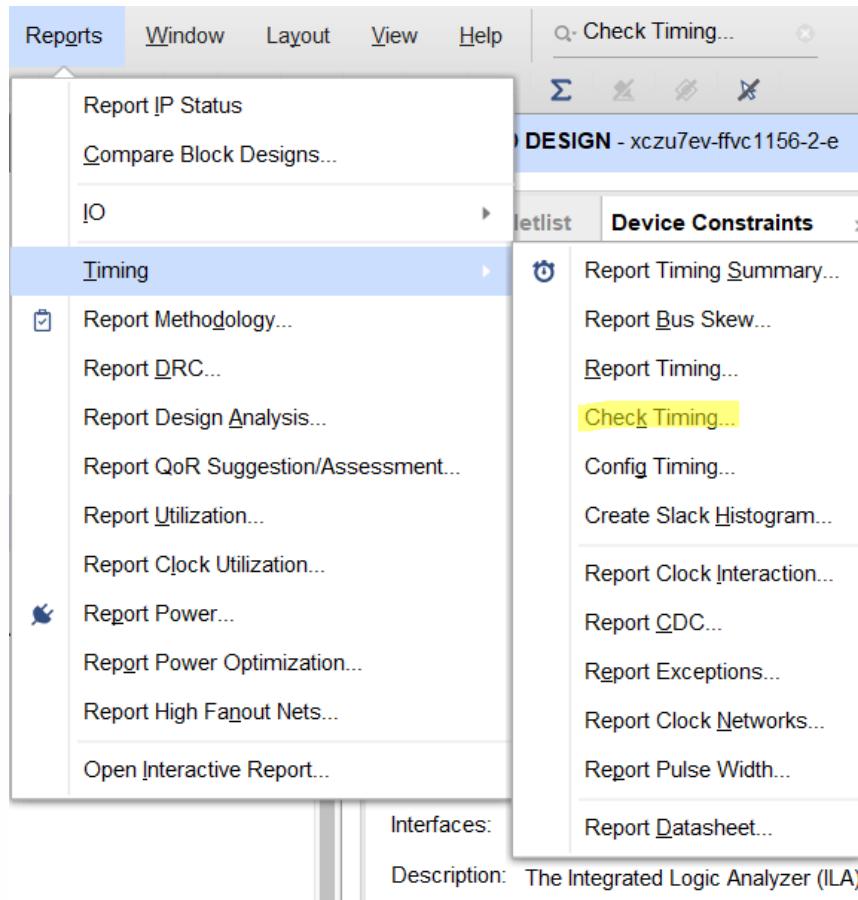
- Skew means signals transport to every component will cause timing differences because the length of each bus is not the same in circuit.

The screenshot shows the 'Timing' report window. The 'Summary' section displays a table of bus skew constraints. The columns are: Constraint, From, To, Corner, Requirement (ns), Actual (ns), and Slack (ns). The table data is as follows:

Constraint	From	To	Corner	Requirement (ns)	Actual (ns)	Slack (ns)
956	cells	cells	Slow	4.000	0.875	3.125
958	cells	cells	Slow	4.000	1.017	2.983
961	cells	cells	Slow	4.000	0.826	3.174
963	cells	cells	Slow	4.000	0.638	3.362
1004	cells	cells	Slow	4.000	2.559	1.441
1044	cells	cells	Slow	4.000	5.561	-1.561
1084	cells	cells	Slow	16.000	0.669	15.331

# Vivado Reports

- Vivado can generate various reports at different stages (Synthesis, Implement,... etc.)

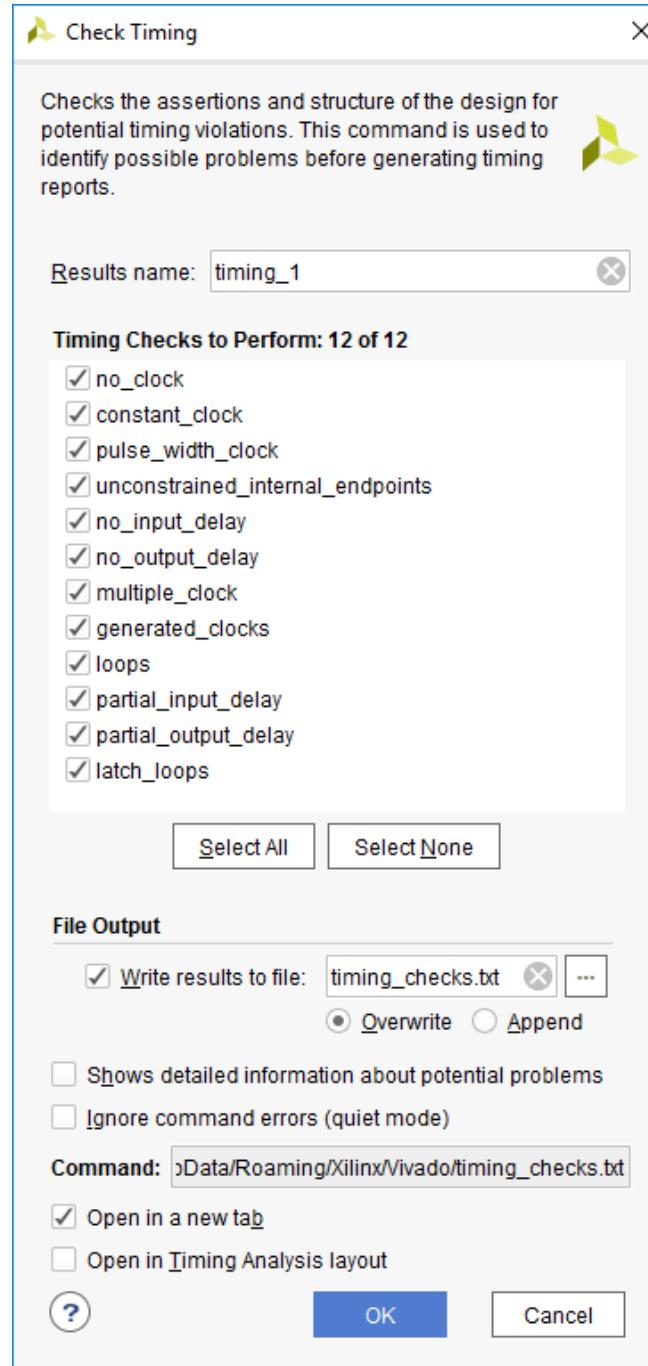


- Check Timing

1. Check ports, pins and paths under timing constraints
2. This will check the probabilistic problem (design data, timing constraints) before running report timing
3. This will find any delays of clocks, inputs/outputs and partial inputs/outputs
4. This also will find out generated clocks, loops or multiple clocks

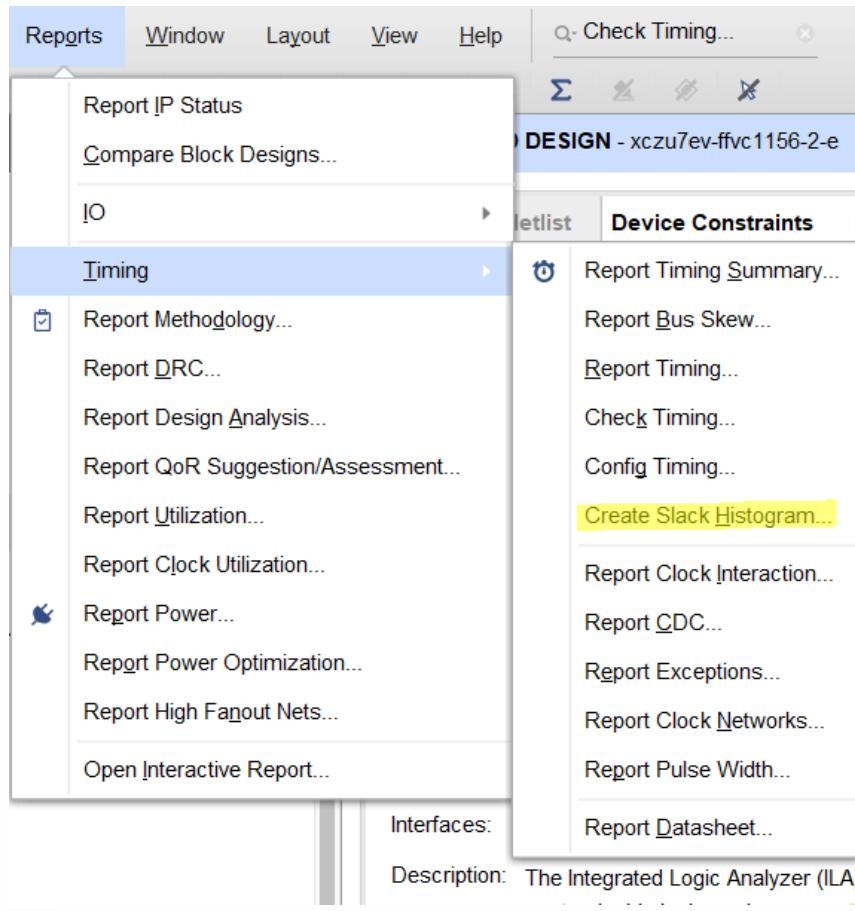
# Vivado Reports

- Check Timing



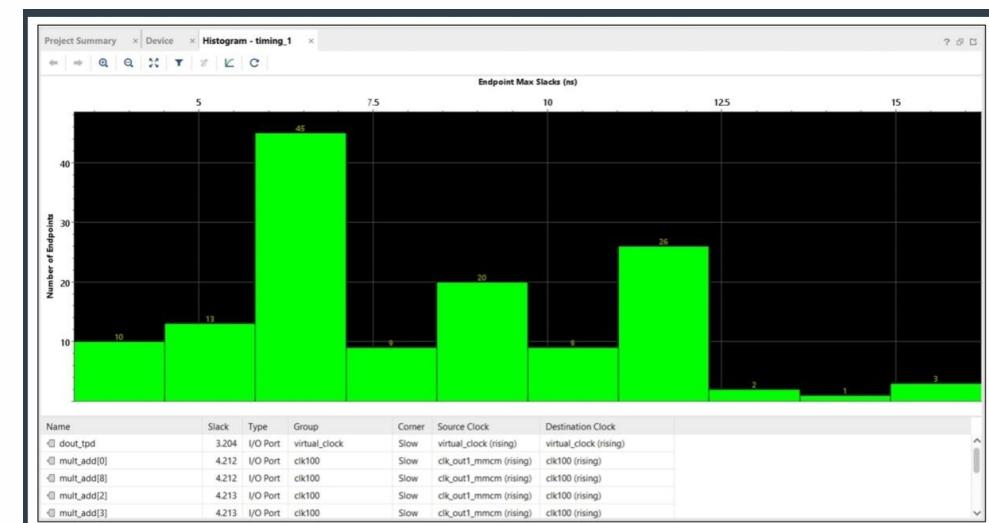
# Vivado Reports

- Vivado can generate various reports at different stages (Synthesis, Implement,... etc.)



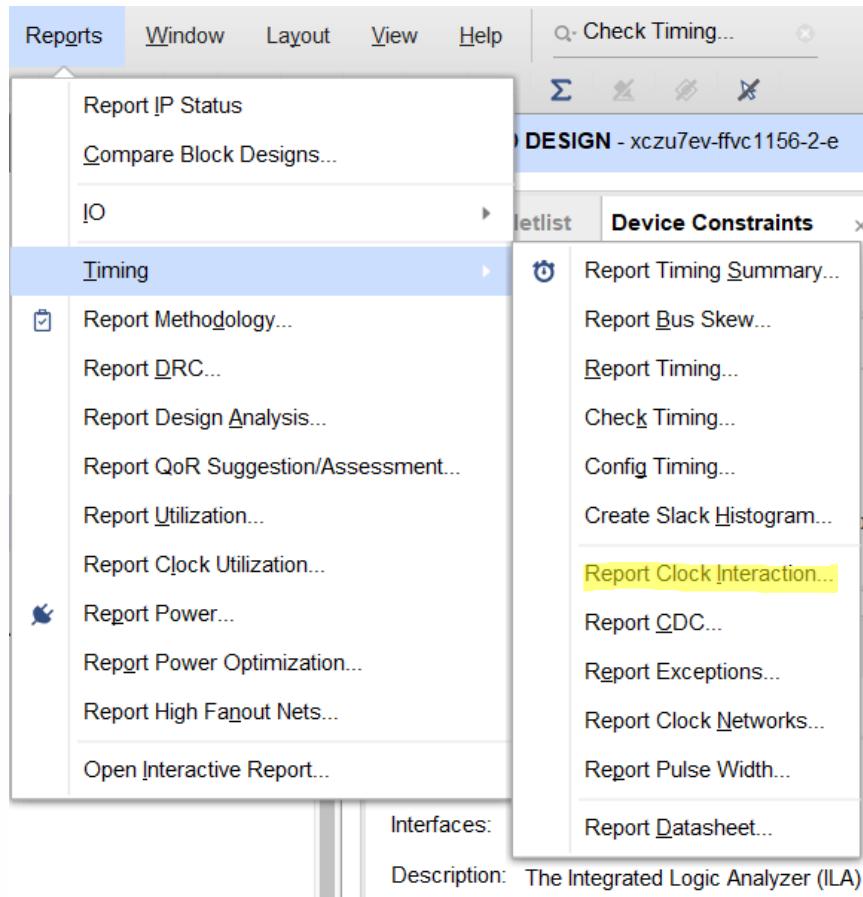
- Slack Histogram

1. Visualize the performance of the whole designed circuit timing paths through the histogram in order to check if having any timing problem at each part of the circuit
2. Positive, means match the timing constraint  
Negative, means not match



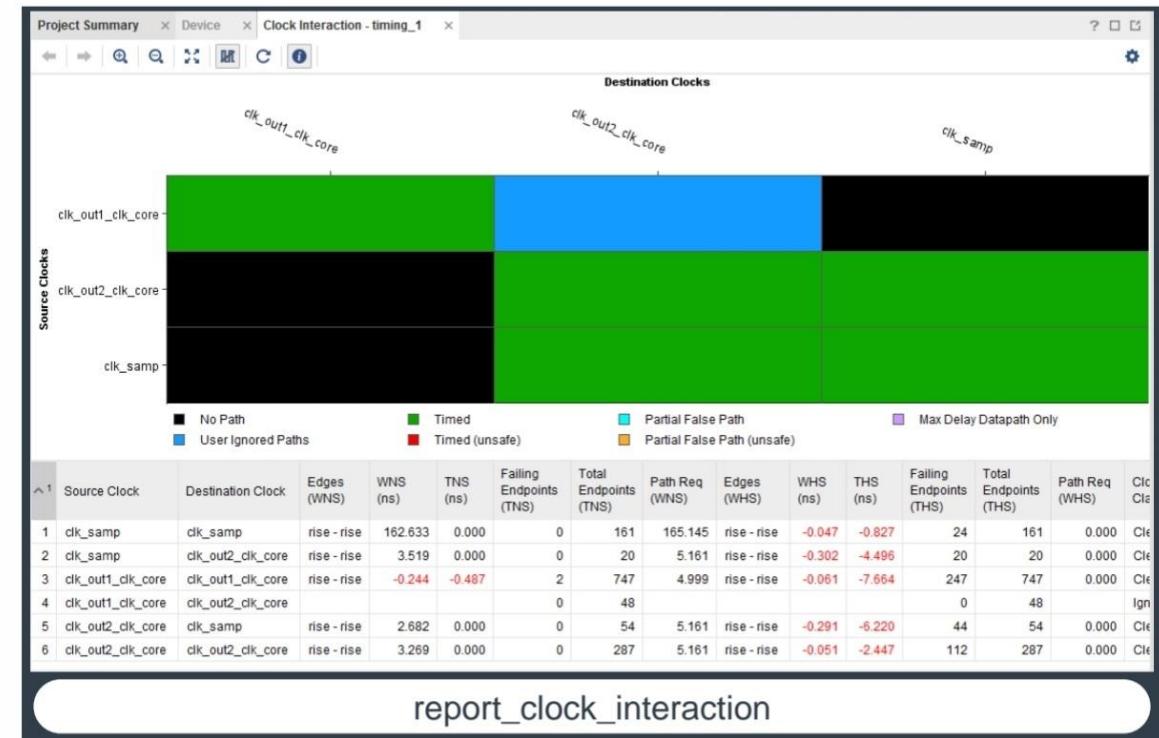
# Vivado Reports

- Vivado can generate various reports at different stages (Synthesis, Implement,... etc.)



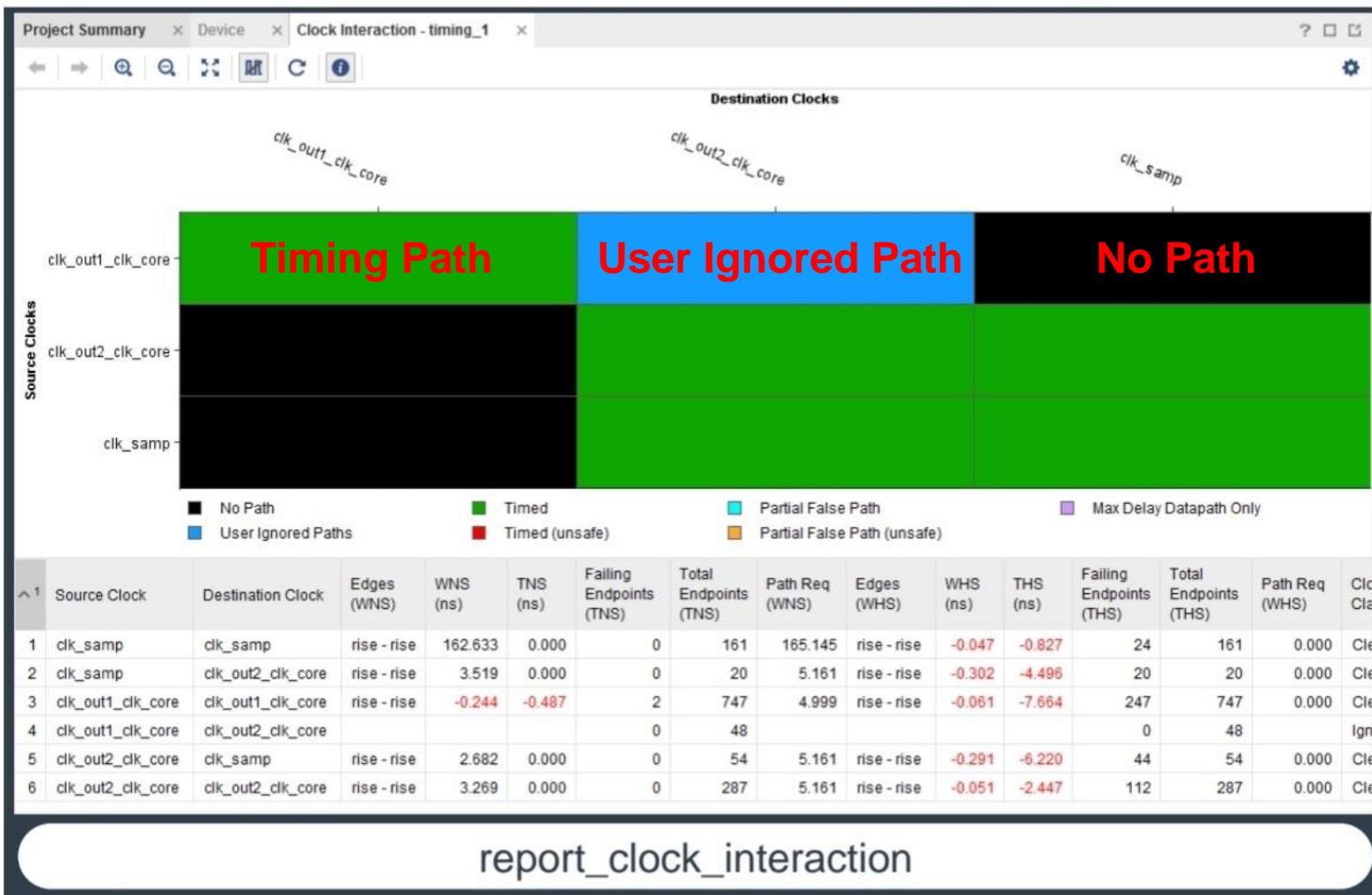
- Clock Interaction

- Analyze the situation of data loss or Metastability from a clock domain to next or source to destination



# Vivado Reports

- Clock Interaction



Partial False Path

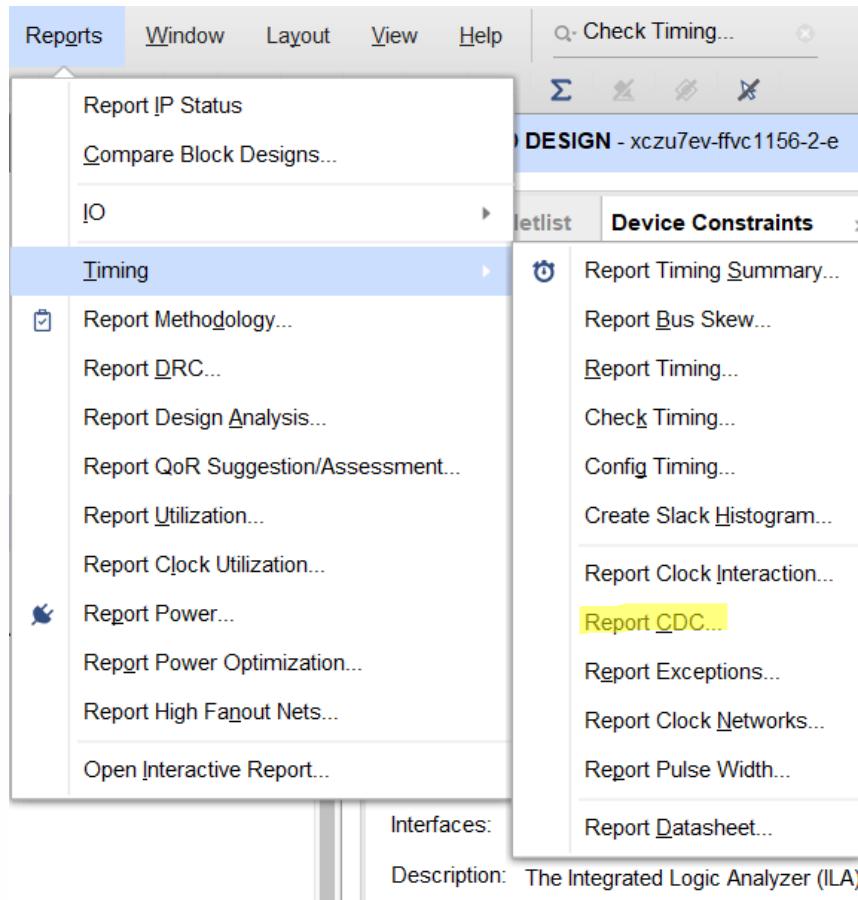
Unsafe Time

Partial False Path (Unsafe)

Max Delay Datapath Only

# Vivado Reports

- Vivado can generate various reports at different stages (Synthesis, Implement,... etc.)

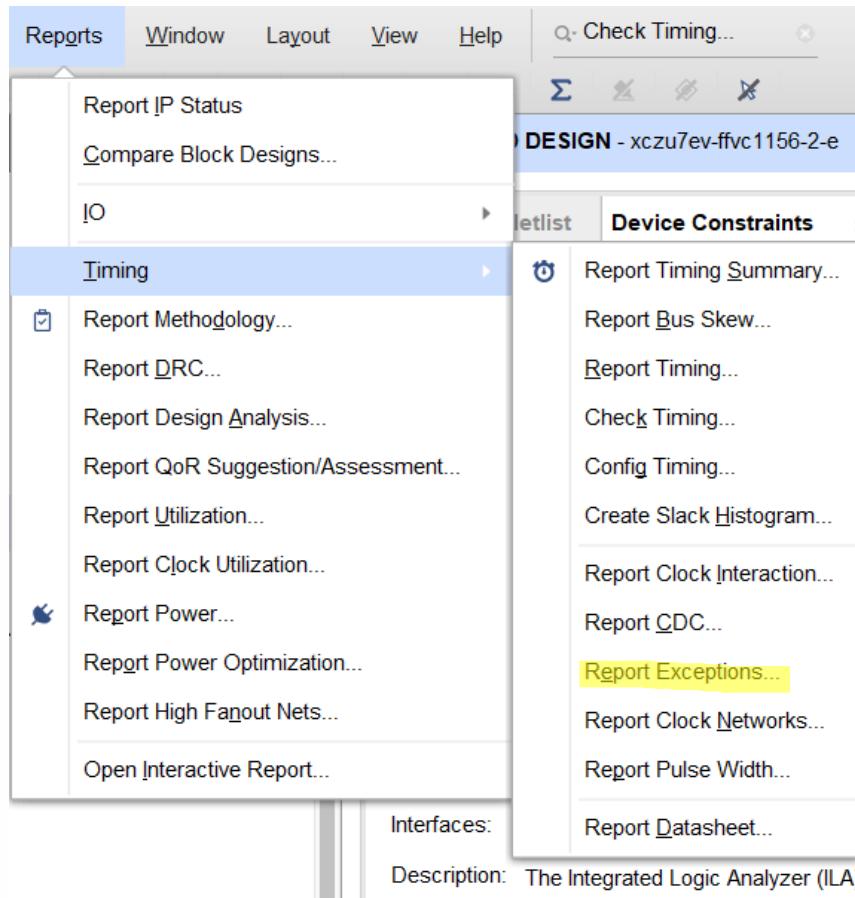


- Clock Domain Clocking (CDC)

- CDC means start and end path are driven from different clock so cause the metastable situation easily.
- Analyze the unsafe path of CDC, which causes the metastable situation.

# Vivado Reports

- Vivado can generate various reports at different stages (Synthesis, Implement,... etc.)

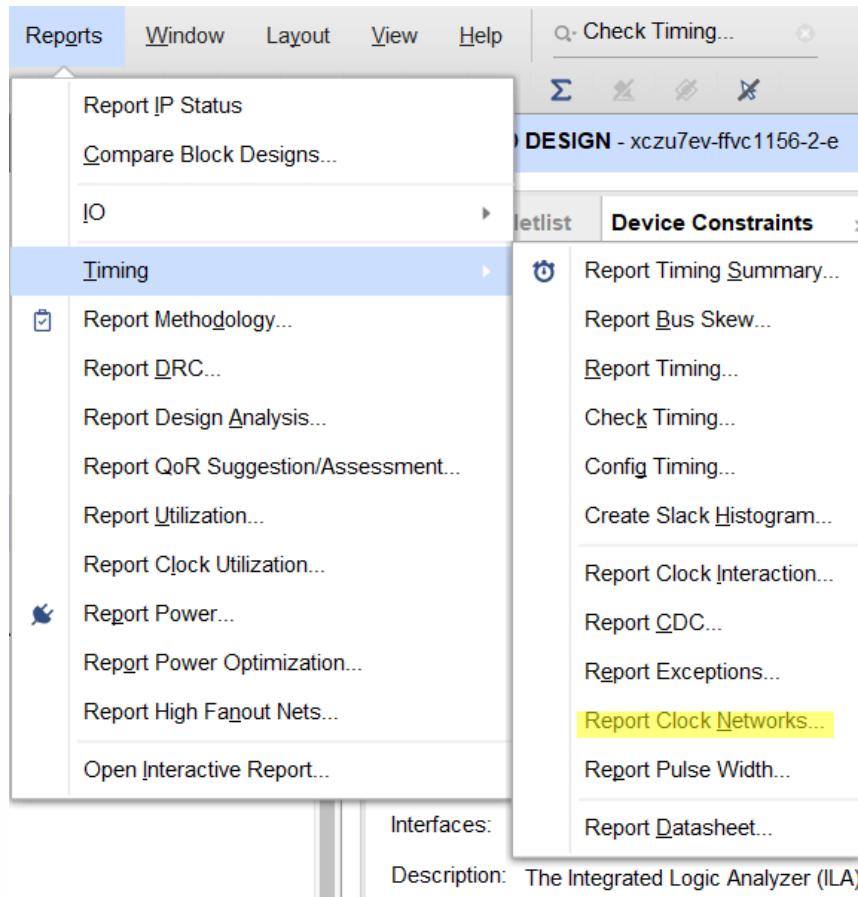


- Report Exceptions

- Report the parts of Timing Exceptions
  - 1. set\_multicycle\_path
  - 2. set\_false\_path
  - 3. set\_case\_analysis

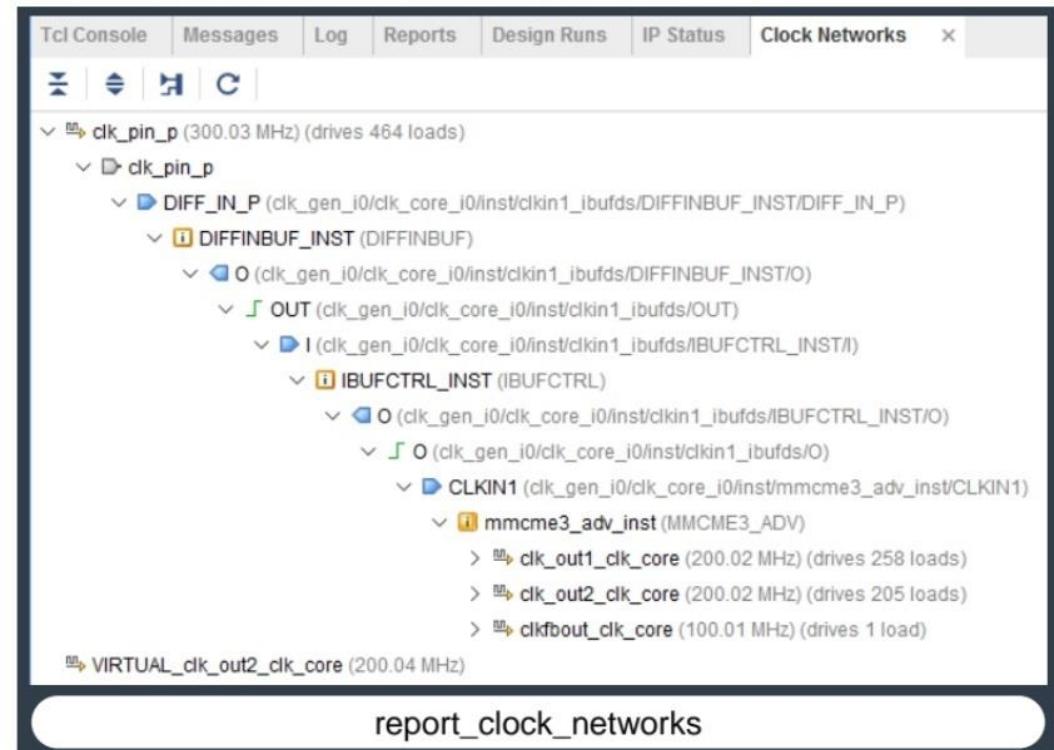
# Vivado Reports

- Vivado can generate various reports at different stages (Synthesis, Implement,... etc.)



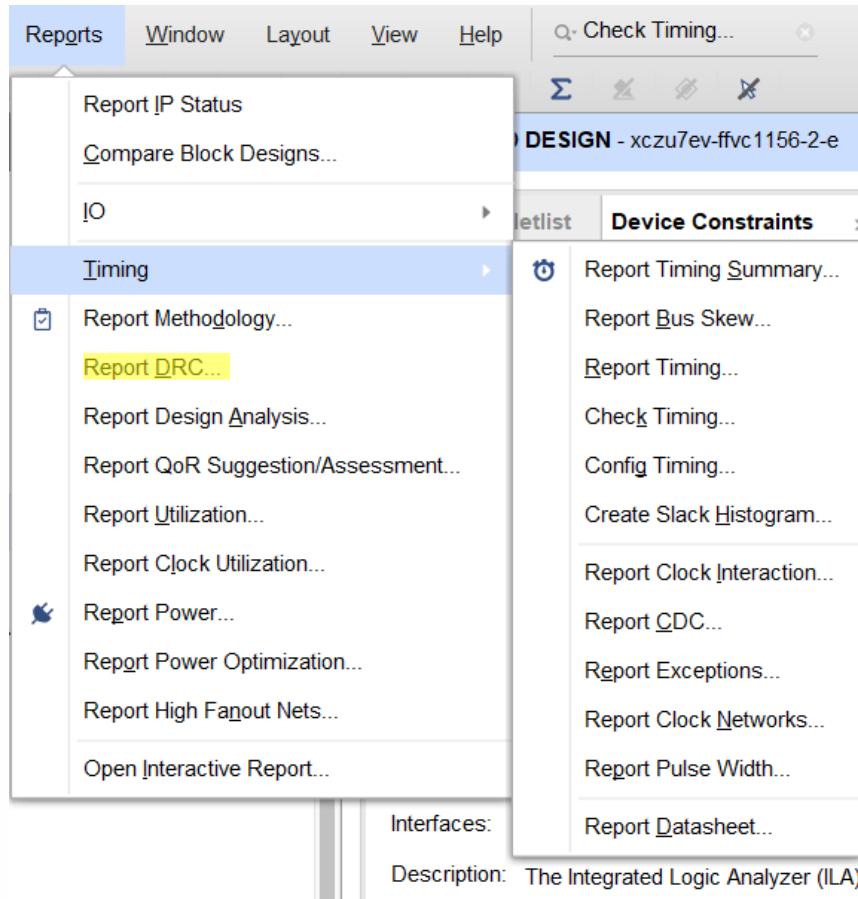
- Report Clock Networks

- Offer a structure to describe the clock distribution from source to destination.



# Vivado Reports

- Vivado can generate various reports at different stages (Synthesis, Implement,... etc.)



- Report Design Rule Check (DRC)

- Offer any parts which violate the design rule
- This will run automatically after Synthesis or Implementation

The screenshot shows the 'Report DRC' window. It has a 'Table of Contents' with sections for 'REPORT SUMMARY' and 'REPORT DETAILS'. In 'REPORT SUMMARY', it lists the netlist, floorplan, design limits, ruledeck, max violations, and violations found (2). A table follows:

Rule	Severity	Description	Violations
UCIO-1	Critical Warning	Unconstrained Logical Port	1
CFGBVS-1	Warning	Missing CFGBVS and CONFIG_VOLTAGE Design Properties	1

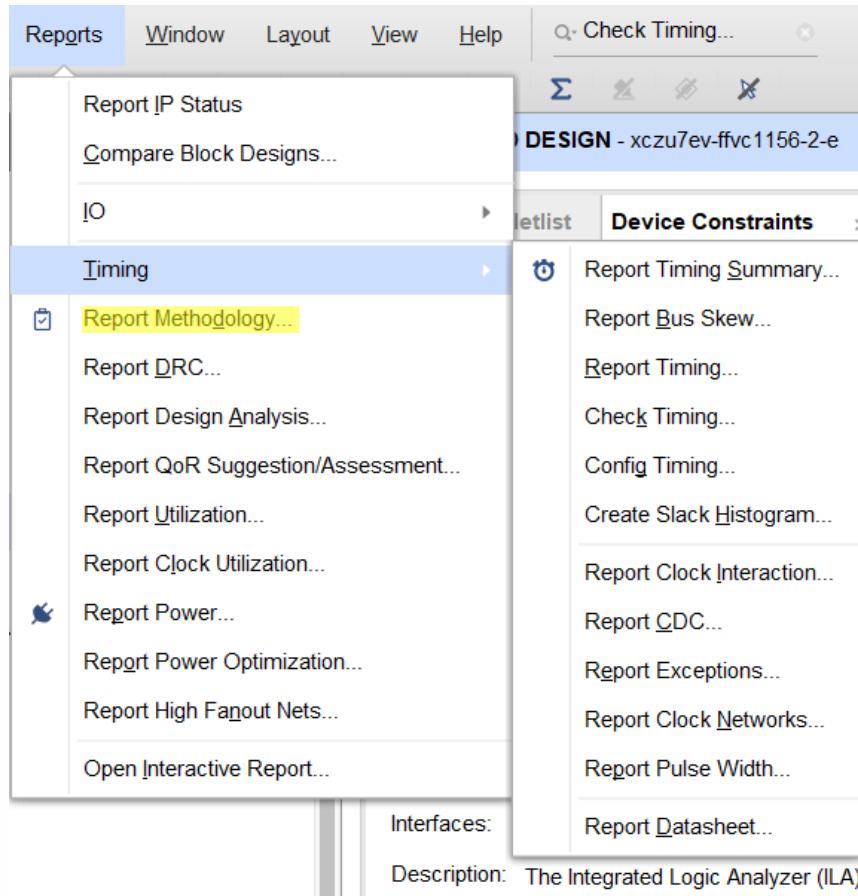
In 'REPORT DETAILS', it details the violations:

- UCIO-1#1 Critical Warning: Unconstrained Logical Port. 18 out of 18 logical ports have no user assigned specific location constraint (LOC). This may cause I/O contention or incompatibility with the Related violations: <none>
- CFGBVS-1#1 Warning: Missing CFGBVS and CONFIG\_VOLTAGE Design Properties. Neither the CFGBVS nor CONFIG\_VOLTAGE voltage property is set in the current\_design. Configuration bank voltage select (CFGBVS) must be set to set\_property CFGBVS value1 [current\_design] #where value1 is either VCCO or GND
- set\_property CONFIG\_VOLTAGE value2 [current\_design] #where value2 is the voltage provided to configuration bank 0

Refer to the device configuration user guide for more information.  
Related violations: <none>

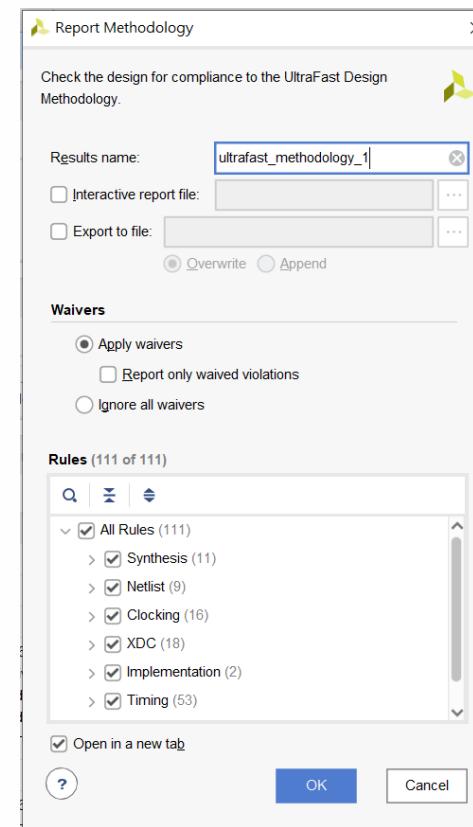
# Vivado Reports

- Vivado can generate various reports at different stages (Synthesis, Implementation, ... etc.)



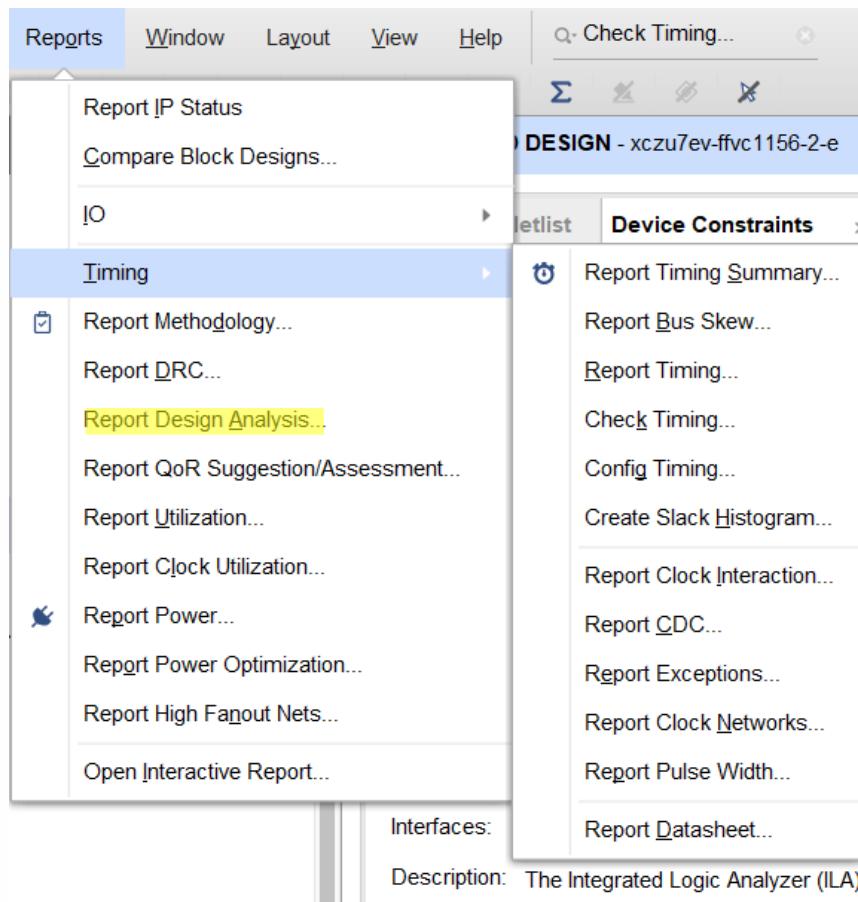
- Report Methodology

- Major important DRC sections
- This will run automatically after Synthesis or Implementation



# Vivado Reports

- Vivado can generate various reports at different stages (Synthesis, Implement,... etc.)



- Report Design Analysis

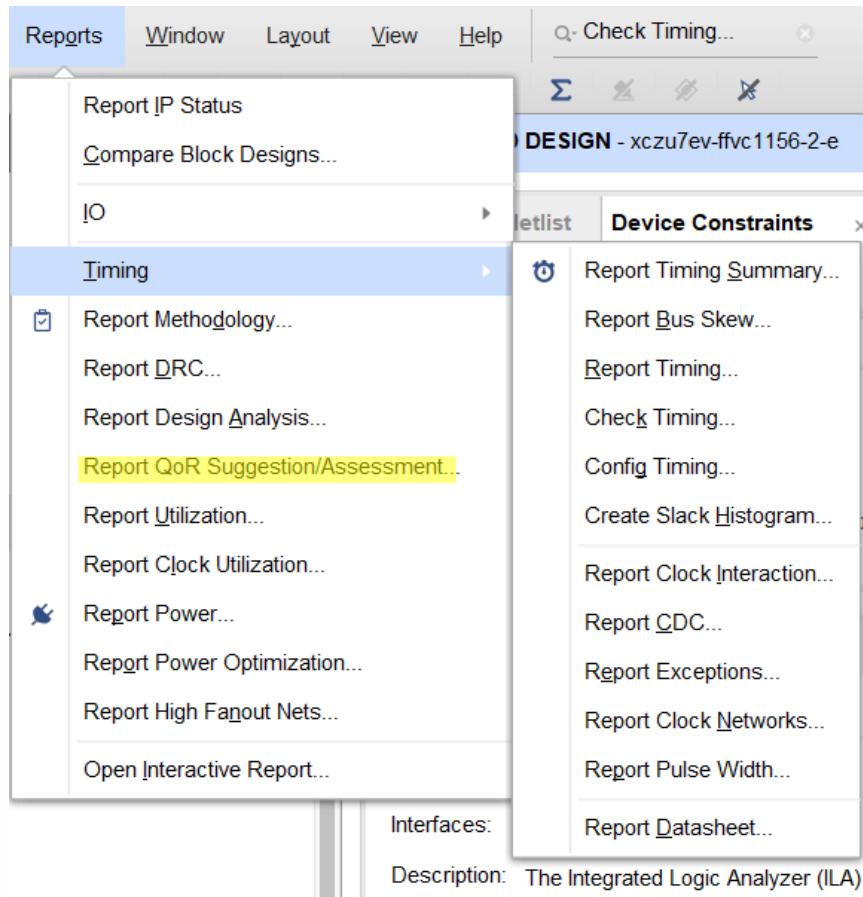
- Report Design Analysis
  - Offer the information of Timing Path Characteristics, Design Interconnect Complexity, Congestion,... etc.

The screenshot shows the 'Design Analysis' tool window in Vivado. The 'Setup Path Characteristics' tab is selected. A table displays six setup paths with their respective parameters:

Name	Requirement	Path Delay	Logic Delay	Net Delay	Clock Skew	Slack	Clock Relationship	Logic Levels	Logical Path	Start Clock
Path 1	0	3.079	3.079	0	0	$\infty$	Safely Timed	2	FDRE OBUF	
Path 2	0	3.06	3.06	0	0	$\infty$	Safely Timed	2	FDRE OBUF	
Path 3	0	3.03	3.03	0	0	$\infty$	Safely Timed	2	FDRE OBUF	
Path 4	0	2.994	2.994	0	0	$\infty$	Safely Timed	2	FDRE OBUF	
Path 5	0	2.435	2.435	0	0	$\infty$	Safely Timed	2	FDRE OBUF	
Path 6	0	2.431	2.431	0	0	$\infty$	Safely Timed	2	FDRE OBUF	

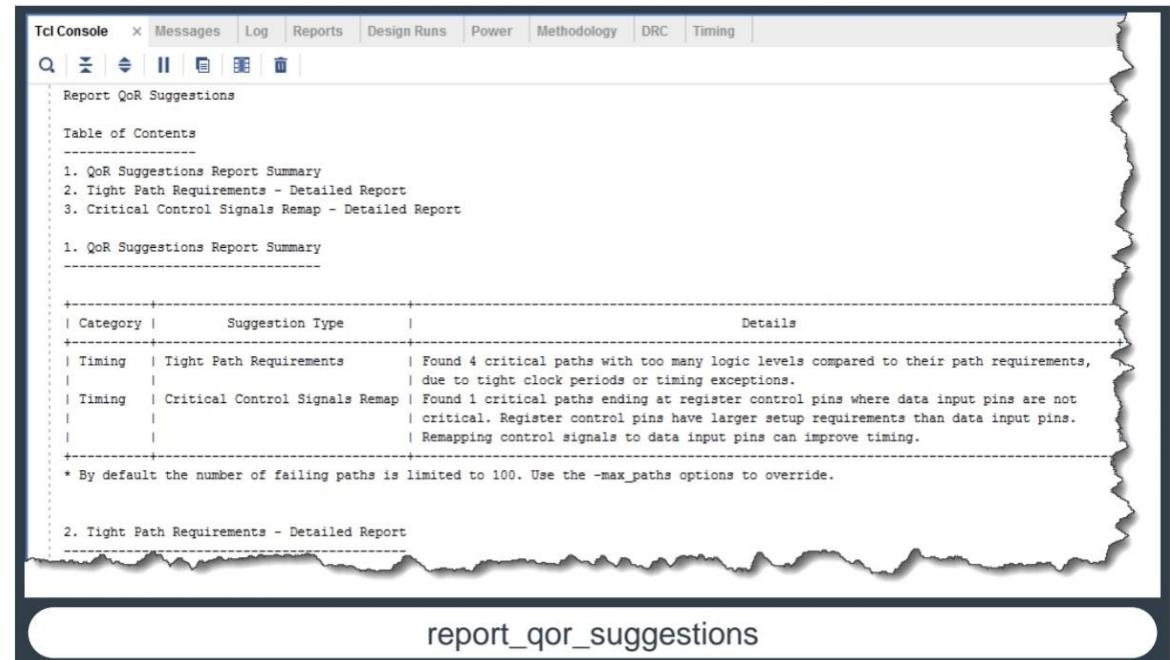
# Vivado Reports

- Vivado can generate various reports at different stages (Synthesis, Implement,... etc.)



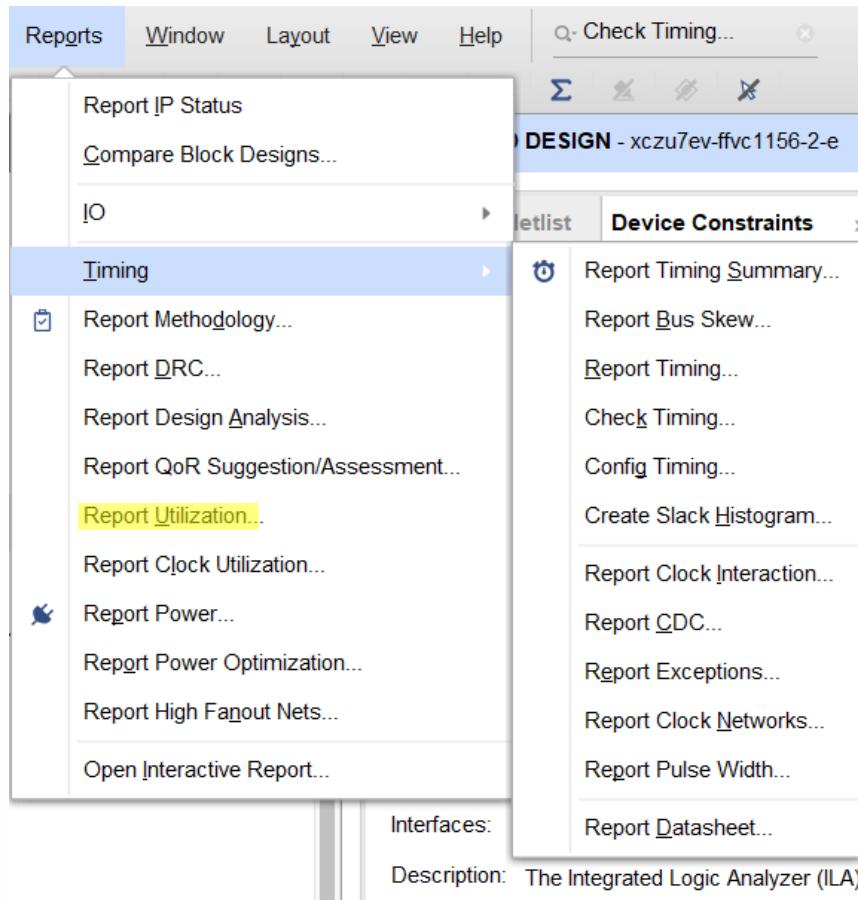
- Report QoR Suggestions (RQS)

- This will offer the suggestion of optimization through analyzing netlist characteristics, timing violations and congestion information.
- This needs to be run after Synthesis or Implementation



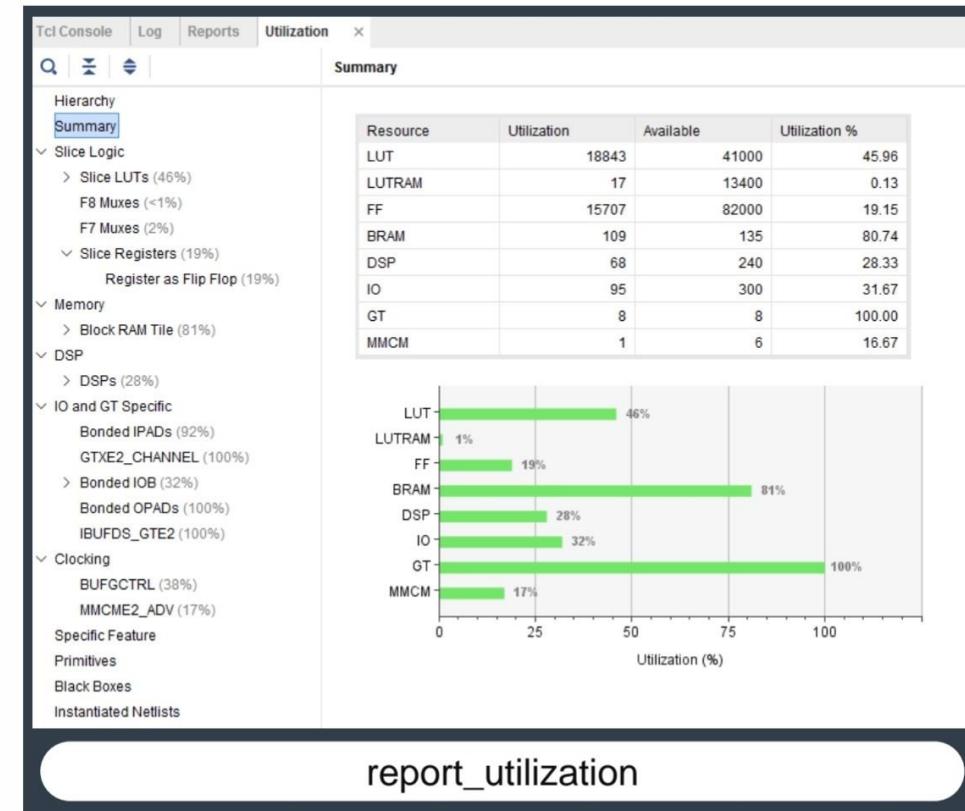
# Vivado Reports

- Vivado can generate various reports at different stages (Synthesis, Implement,... etc.)



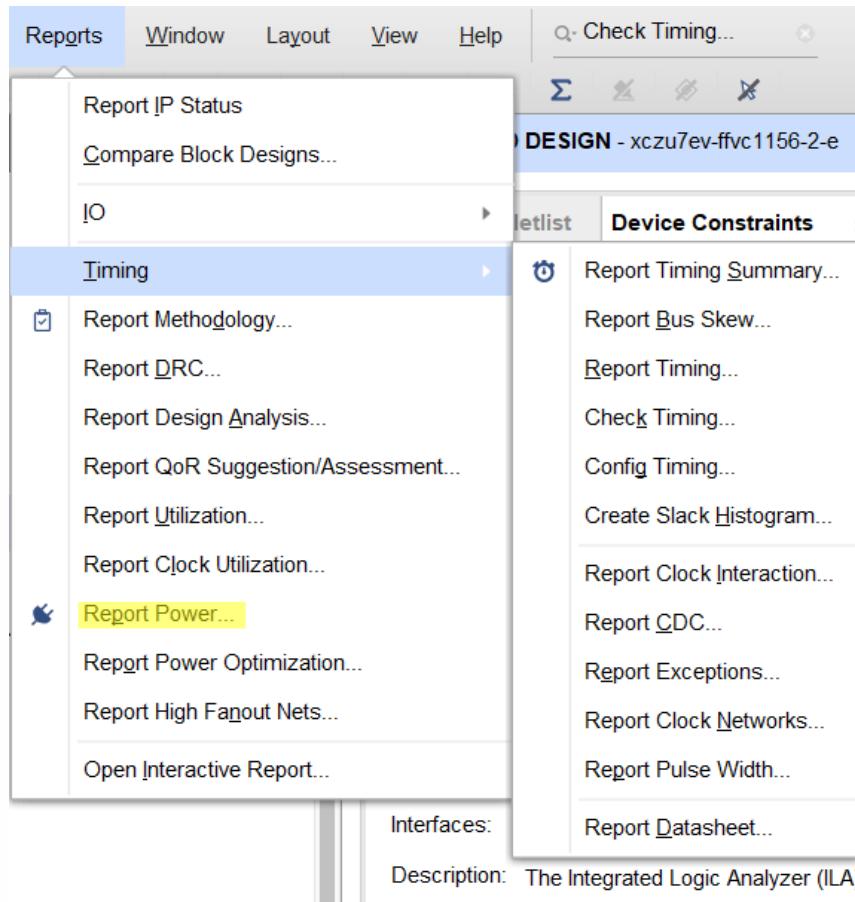
- Report Utilization

- Offer the ratio of resource distribution which including LUTs, Registers, FIFO, I/O,... etc., after Synthesis or Implementation.



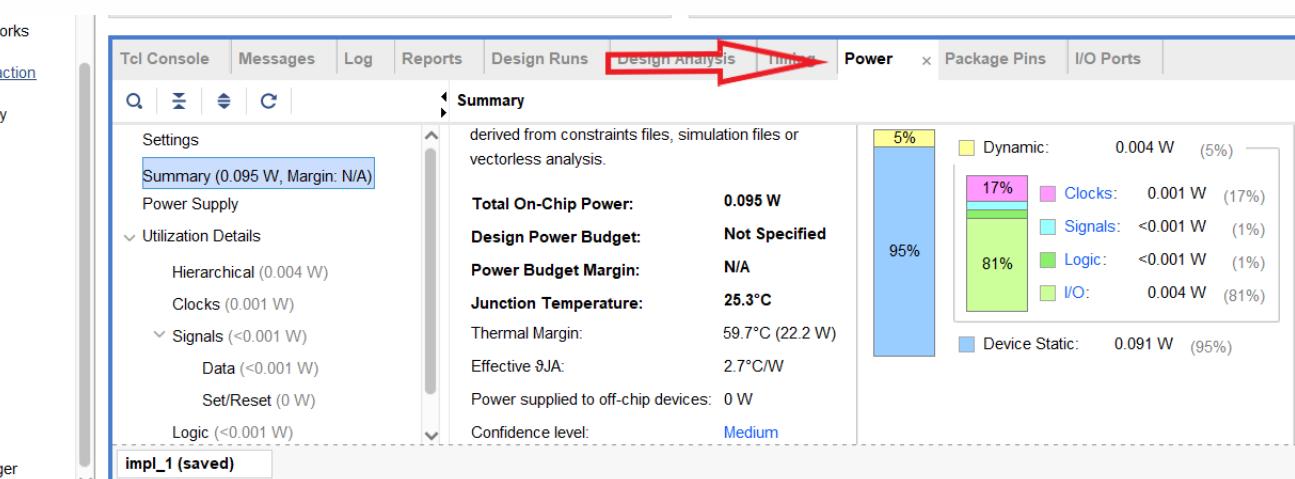
# Vivado Reports

- Vivado can generate various reports at different stages (Synthesis, Implement,... etc.)



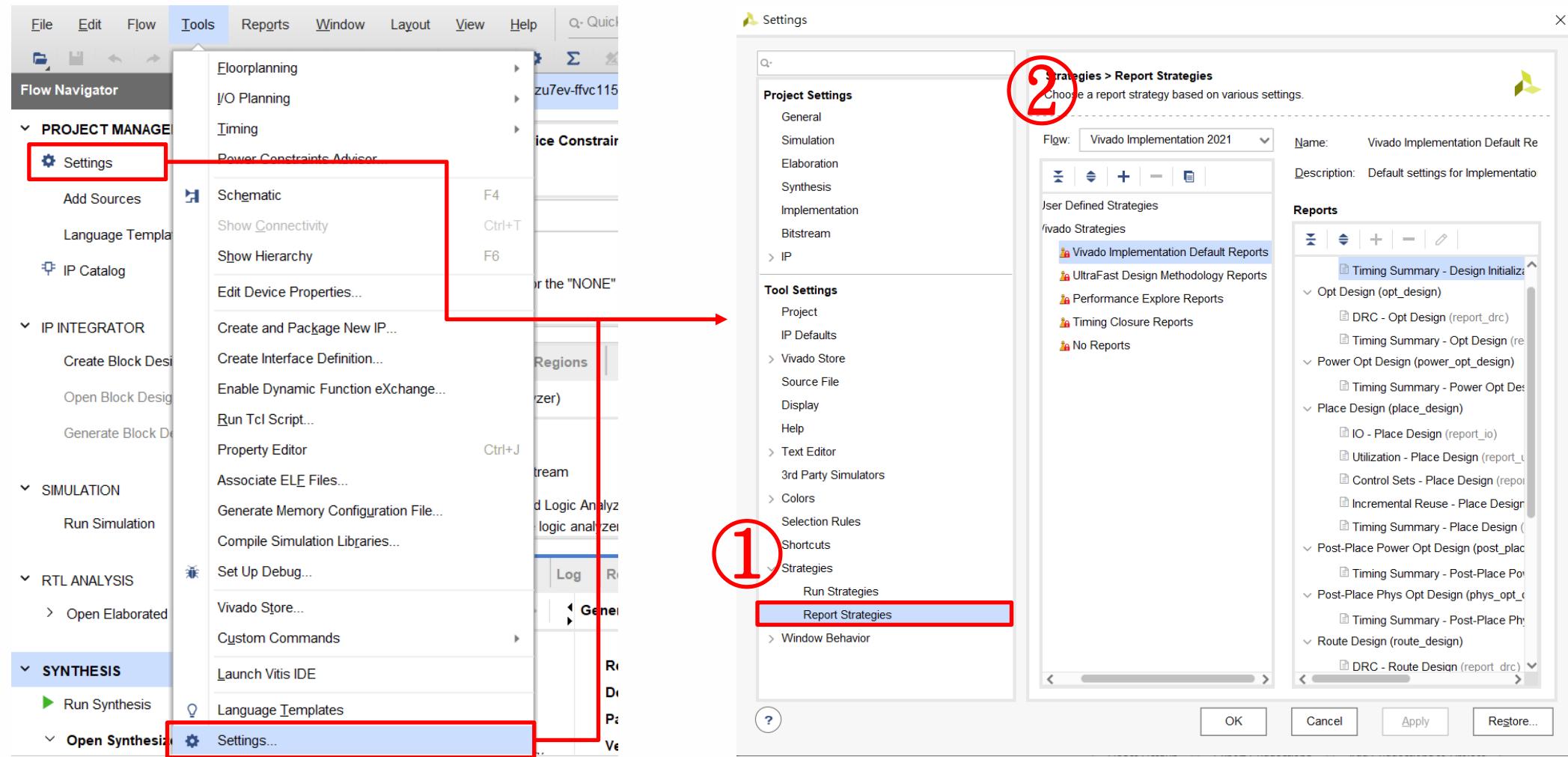
- Report Power

- Power consumption is calculated based on the junction and the ambient temperature.
- Generally used after routing stage
- Allows detailed power distribution analysis to guide power saving strategies and to reduce dynamic, thermal, or off-chip power.



# Vivado Report Strategies

- We can choose or customize the various reports which run automatically after finishing the Synthesis or Implementation.



# Vivado Configuration

# Vivado Configuration

- FPGA needs some special pins when loading the bitstream file

DONE

Bi-directional signal. High potential means the configuration has been finished

DIN

Input pin of data

DOUT

Output to next device in Daisy Chain

CFGVB5

Can be connected to Vcc or GND, the voltage value of Vcc can be 1.5, 1.8, 2.5 or 3.3

POR\_OVERRIDE

Disable Power-on-reset Delay

# Vivado Configuration

- FPGA needs some special pins when loading the bitstream file

INIT\_B

Bi-directional signal. Low potential means the configuration storage is empty.

M2,M1,M0 pins

Means the configuration mode of selection currently

PROGRAM\_B

Reset the configuration of FPGA when lowering the signals, on the contrary it starts to load new configuration

CCLK

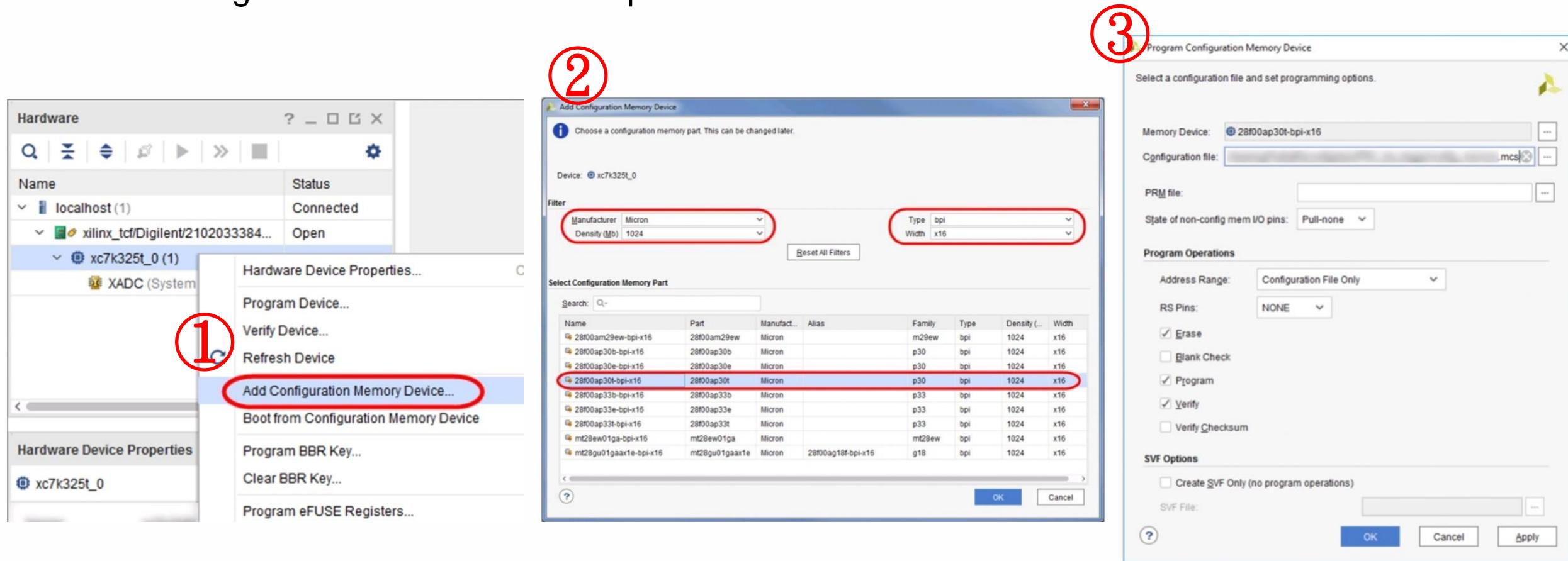
Configurate clock up to 100MHz. This is input/output in Slave/Master mode

PUDC\_B

Select the enable or disable of pull-ups during configuration on user I/O pins

# Vivado Configuration

- Configuration Memory File
  - Program the bitstream file to the flash which with spi/bpi interface in order to store the configuration information when power off.



# Vivado TCL Environment



# Vivado TCL Environment

- Tool Command Language (TCL)
  - In Vivado, only support some TCL commands, but add some specific TCL for synthesis, implementation,... etc.

<u>Safe Base</u>	<u>encoding</u>	<u>if</u>	<u>pid</u>	<u>switch</u>
<u>Tcl</u>	<u>eof</u>	<u>incr</u>	<u>pkg::create</u>	<u>tcl_endOfWord</u>
<u>after</u>	<u>error</u>	<u>info</u>	<u>pkg_mkIndex</u>	<u>tcl_findLibrary</u>
<u>append</u>	<u>eval</u>	<u>interp</u>	<u>proc</u>	<u>tcl_startOfNextWord</u>
<u>array</u>	<u>exec</u>	<u>join</u>	<u>puts</u>	<u>tcl_startOfPreviousWord</u>
<u>auto_execok</u>	<u>exit</u>	<u>lappend</u>	<u>pwd</u>	<u>tcl_wordBreakAfter</u>
<u>auto_import</u>	<u>expr</u>	<u>lindex</u>	<u>re_syntax</u>	<u>tcl_wordBreakBefore</u>
<u>auto_load</u>	<u>fblocked</u>	<u>linsert</u>	<u>read</u>	<u>tcetest</u>
<u>auto_mkindex</u>	<u>fconfigure</u>	<u>list</u>	<u>regexp</u>	<u>tclvars</u>
<u>auto_mkindex_old</u>	<u>fcopy</u>	<u>llength</u>	<u>registry</u>	<u>tell</u>
<u>auto_qualify</u>	<u>file</u>	<u>load</u>	<u>time</u>	
<u>auto_reset</u>	<u>fileevent</u>	<u>lrange</u>	<u>trace</u>	
<u>bgerror</u>	<u>filename</u>	<u>lreplace</u>	<u>resource</u>	
<u>binary</u>	<u>flush</u>	<u>lsearch</u>	<u>return</u>	
<u>break</u>	<u>for</u>	<u>lset</u>	<u>scan</u>	
<u>catch</u>	<u>foreach</u>	<u>lsort</u>	<u>seek</u>	
<u>cd</u>	<u>format</u>	<u>memory</u>	<u>set</u>	
<u>clock</u>	<u>gets</u>	<u>msgcat</u>	<u>socket</u>	
<u>close</u>	<u>glob</u>	<u>namespace</u>	<u>source</u>	
<u>concat</u>	<u>global</u>	<u>open</u>	<u>split</u>	
<u>continue</u>	<u>history</u>	<u>package</u>	<u>string</u>	
<u>dde</u>	<u>http</u>	<u>parray</u>	<u>subst</u>	

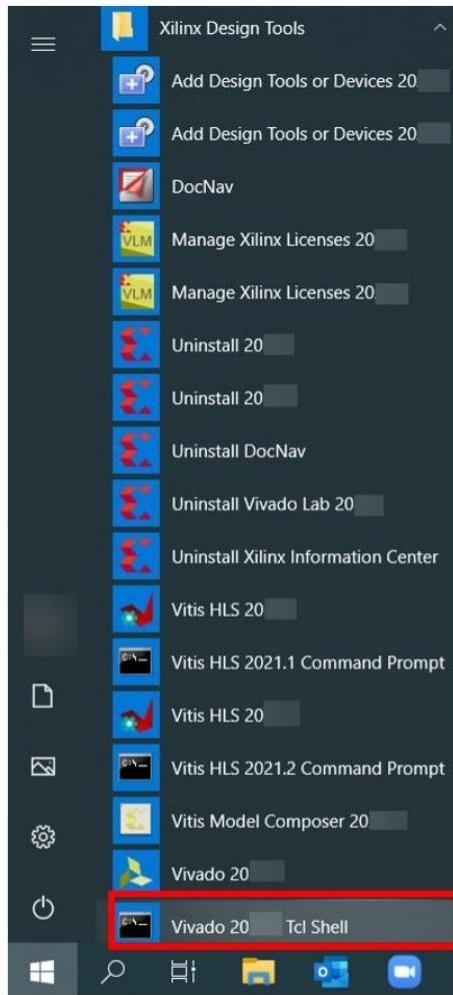
<u>Safe Base</u>	<u>encoding</u>	<u>if</u>	<u>pid</u>	<u>switch</u>
<u>Tcl</u>	<u>eof</u>	<u>incr</u>	<u>pkg::create</u>	<u>tcl_endOfWord</u>
<u>after</u>	<u>error</u>	<u>info</u>	<u>pkg_mkIndex</u>	<u>tcl_findLibrary</u>
<u>append</u>	<u>eval</u>	<u>interp</u>	<u>proc</u>	<u>tcl_startOfNextWord</u>
<u>array</u>	<u>exec</u>	<u>join</u>	<u>puts</u>	<u>tcl_startOfPreviousWord</u>
<u>auto_execok</u>	<u>exit</u>	<u>lappend</u>	<u>pwd</u>	<u>tcl_wordBreakAfter</u>
<u>auto_import</u>	<u>expr</u>	<u>lindex</u>	<u>re_syntax</u>	<u>tcl_wordBreakBefore</u>
<u>auto_load</u>	<u>fblocked</u>	<u>linsert</u>	<u>read</u>	<u>tcetest</u>
<u>auto_mkindex</u>	<u>fconfigure</u>	<u>list</u>	<u>regexp</u>	<u>tclvars</u>
<u>auto_mkindex_old</u>	<u>fcopy</u>	<u>llength</u>	<u>registry</u>	<u>tell</u>
<u>auto_qualify</u>	<u>file</u>	<u>load</u>	<u>time</u>	
<u>auto_reset</u>	<u>fileevent</u>	<u>lrange</u>	<u>trace</u>	
<u>bgerror</u>	<u>filename</u>	<u>lreplace</u>	<u>resource</u>	
<u>binary</u>	<u>flush</u>	<u>lsearch</u>	<u>return</u>	
<u>break</u>	<u>for</u>	<u>lset</u>	<u>scan</u>	
<u>catch</u>	<u>foreach</u>	<u>lsort</u>	<u>seek</u>	
<u>cd</u>	<u>format</u>	<u>memory</u>	<u>set</u>	
<u>clock</u>	<u>gets</u>	<u>msgcat</u>	<u>socket</u>	
<u>close</u>	<u>glob</u>	<u>namespace</u>	<u>source</u>	
<u>concat</u>	<u>global</u>	<u>open</u>	<u>split</u>	
<u>continue</u>	<u>history</u>	<u>package</u>	<u>string</u>	
<u>dde</u>	<u>http</u>	<u>parray</u>	<u>subst</u>	

The highlighted Tcl commands are supported in the Vivado IDE

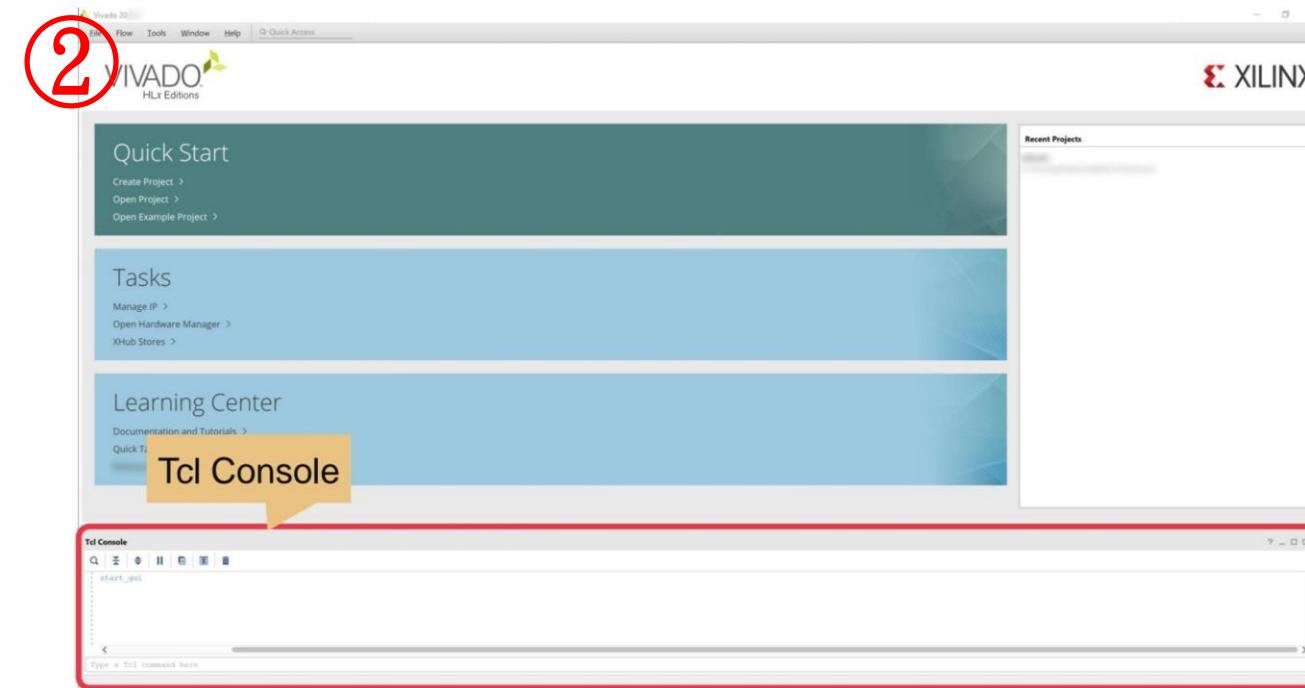
# Vivado TCL Environment

- Tool Command Language (TCL)
  - Can be run at three situations

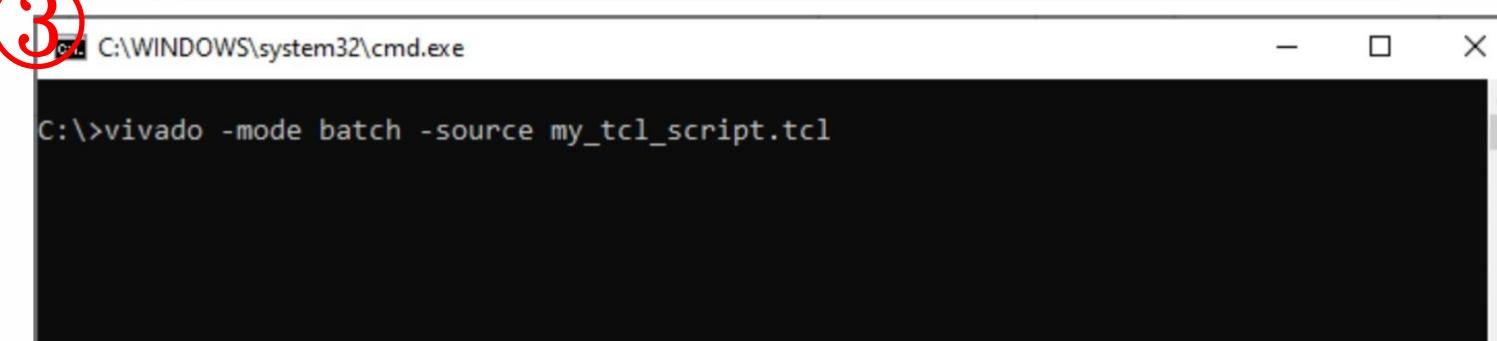
①



②



③



# Vivado TCL Environment

- Tool Command Language (TCL) Basic Syntax

```
% expr 3+4  
7  
% set age 35  
35  
% puts "I am $age years old"  
I am 35 years old  
% Set age 42  
Invalid command name "Set"
```

**expr** is arithmetic operation

**result**

**set** variable name and value

**result**

**puts** is as to “print” on the console

“\$” will get the value of variable

Upper and lower case have syntax difference

# Vivado TCL Environment

- Tool Command Language (TCL) Basic Syntax
  - Command Substitution

```
% set Pi 3.142; set radius 10  
10  
% set area [expr $Pi * $radius * $radius]  
314.2
```

Semicolon is command separator

[ ] presents we can calculate multiple variables inside and get one value

Everything as far as the next ] is treated as a command

```
set a [set b ""]
```

Matching [ ]

Single ] in quotes doesn't match [

Write multiple TCL in one line and separate by semicolon

```
% set ten 10 ; set twenty 20 ; set thirty 30  
30  
%
```

Output on console just only show the value behind the last semicolon

# Vivado TCL Environment

- Tool Command Language (TCL) Basic Syntax
  - Quoting



# Vivado TCL Environment

- Tool Command Language (TCL) Basic Syntax
  - Quoting with Braces

{ } presents that all things inside is the string

```
% set netname {data$bus[31]}  
data$bus[31]
```

Word

```
% set Tcl/Tk 8.4  
8.4  
% set version ${Tcl/Tk}f  
8.4f
```

Not recommended

# Vivado TCL Environment

- Tool Command Language (TCL) Basic Syntax
  - Backslash

```
% puts "Special characters: \"\t\n\\\"."
Special characters: "[{\\".
% puts {Not special: \"\[\\"}.
Not special: \"\[\\".
```

print special string

No substitution inside braces

```
% puts "\tTab and \nnewline"
      Tab and
      newline
```

\t means tab, \n means newline

Similar to C backslash-escapes

```
% puts {All on \
one line}
All on one line
```

Must be last character on line

Even inside braces and comments!

# Vivado TCL Environment

- Tool Command Language (TCL) Basic Syntax
  - Comments

```
# this is a comment; and so is this  
set multiline "Line 1  
# But this is just Line 2 of a string!      This will become string in quoting  
Line 3"  
  
set a 5      # This is a syntax error  
set b 6 ;   # but this is a valid comment
```

USE COMMENTS to document your scripts!

# Vivado TCL Environment

- Tool Command Language (TCL)

Script file grandpa.tcl

```
set age 40  
set name Jonathan  
puts "Hello $name"  
puts "Life begins at $age"
```

Variables can store text

→ Write multiple TCL commands in .tcl file

Results

```
C:\training> tclsh grandpa.tcl  
Hello Jonathan  
Life begins at 40
```

→ Run .tcl file through **tclsh** or  
**source xxx.tcl**

# Vivado TCL Environment

- Run Tool Command Language (TCL) in different OS
  - 1. Linux

Script file grandpa

```
#! /usr/local/bin/tclsh  
set age 40  
set name Jonathan  
puts "Hello $name"  
puts "Life begins at $age"
```

Specify which interpreter to use

Running the script

```
training@hannah $ chmod 555 grandpa  
training@hannah $ grandpa
```

Anyone can read and execute

# Vivado TCL Environment

- Run Tool Command Language (TCL) in different OS

## 2. Windows

- 
- The list consists of six horizontal bars, each containing a step number and a descriptive text. The bars are color-coded: the first three are red, the next two are blue, and the last one is green.
- 01 Give the script file a .tcl extension so Windows knows its type
  - 01 Associate .tcl with the Tcl interpreter (tclsh)
  - 01 Double-click the script file to execute it
  - 02 Run tclsh from a command window
  - 02 Supply the script file name on the command line
  - 03 Run Tcl scripts for the Vivado IDE from within the Vivado tool

# Vivado TCL Environment

- TCL Capabilities in Vivado

## **Find all BUFG primitives:**

```
set my_bufgs [get_cells -hier -filter {ref_name == BUFG}]
```

## **Find all drivers(s) for every BUFG:**

```
foreach x $my_bufgs { puts "$x: [all_fanin -flat -startpoints_only  
[get_pins -filter {direction == IN} -of [get_cells $x]]]"}
```

## **Report the ten longest paths from resp\_data to bcd\_out:**

```
report_timing -from [get_cells cmd_parse_i0/send_resp_data*]  
-to [get_cells resp_gen_i0/to_bcd_i0/bcd_out*] -max_paths 10
```

## **Create a period constraint:**

```
create_clock -name clk_pin_p -period 10.000 [get_ports clk_pin_p]
```

## **Execute programs:**

```
place_design, synth_design
```

# Vivado TCL Environment

- **TCL Capabilities in Vivado**

Tcl allows users to query a design database

Numerous reasons for searching objects in the design database

A common mechanism is used to search for all Vivado Design Suite objects

- Constraining the design
- Generating custom timing reports
- Finding objects in a schematic
- Creating custom schematics
- Setting properties on objects (primitive attributes, I/O standards, etc.)
- Manually placing cells (LOC)
- Floorplanning (Pblocks)
- Controlling the incremental router
- Performing minor changes (ECOs)

# Vivado TCL Environment

- TCL Initialization Scripts
  - Vivado will search Scripts from three locations when executing
    1. Installation of Vivado
    2. Local user directory for specific version of Vivado
    3. Local user directory for general version of Vivado
  - It doesn't offer TCL Initialization Scripts when we install Vivado, so we need to create **Vivado\_init.tcl** manually to replace or add the parts of original Vivado TCL commands.

**Thank you very much for your attention!**