Xilinx Vivado 2021.2 Training Course

XILINX.

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 Viavado

XILINX.

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 UltraScale FPGA: • XCKU025, XCKU035 Kintex UltraScale+ FPGA: • XCKU3P, XCKU5P	Kintex®-7 FPGA: • All Kintex UltraScale FPGA: • All 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

XILINX.

Vivado ML Edition Features

- Operating System Support

Microsoft Windows Support –

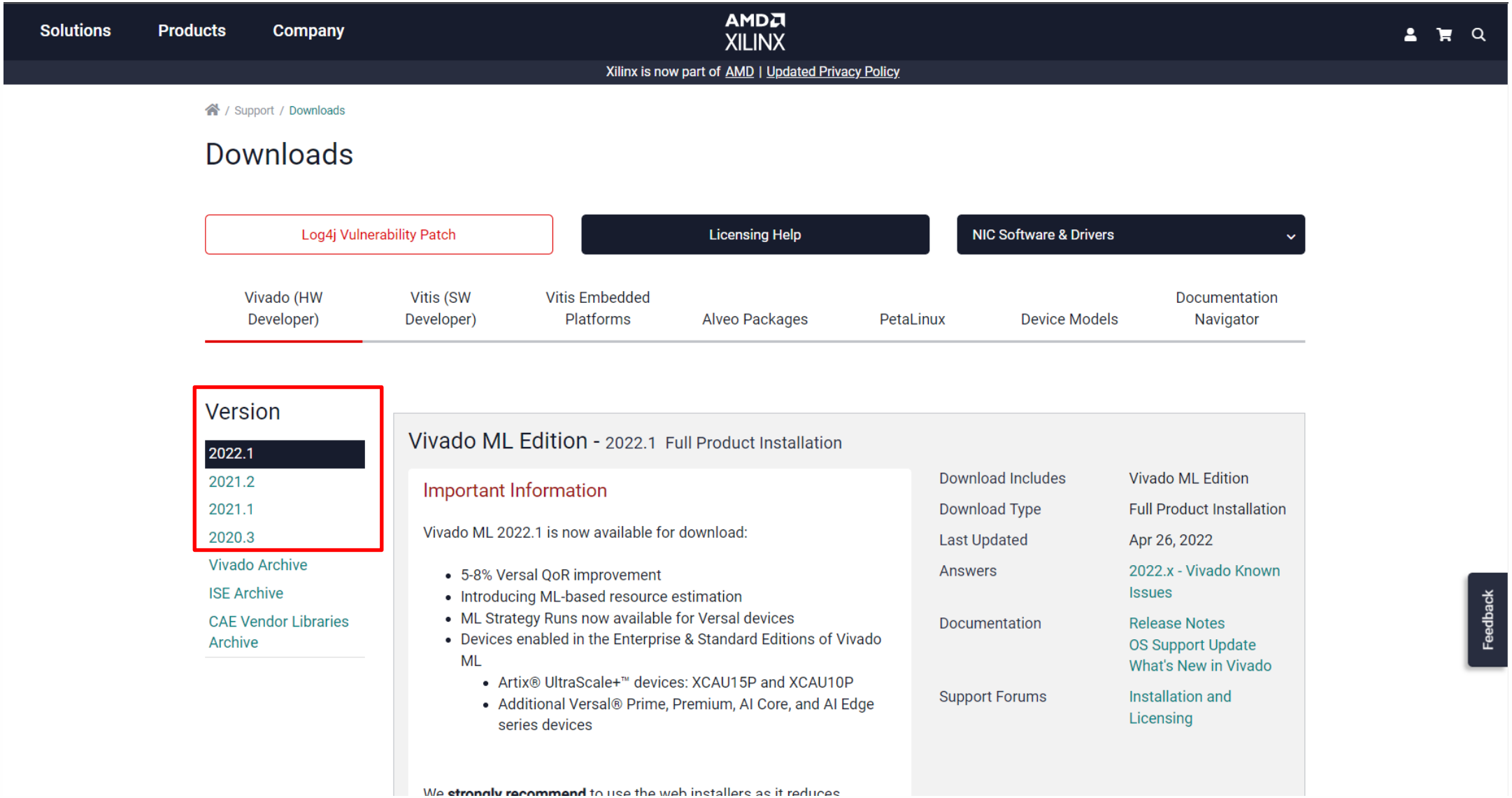
- Windows update: 10.0 1809 Update; 10.0 1903 Update; 10.0 1909 Update; 10.0 2004 Update

Linux Support –

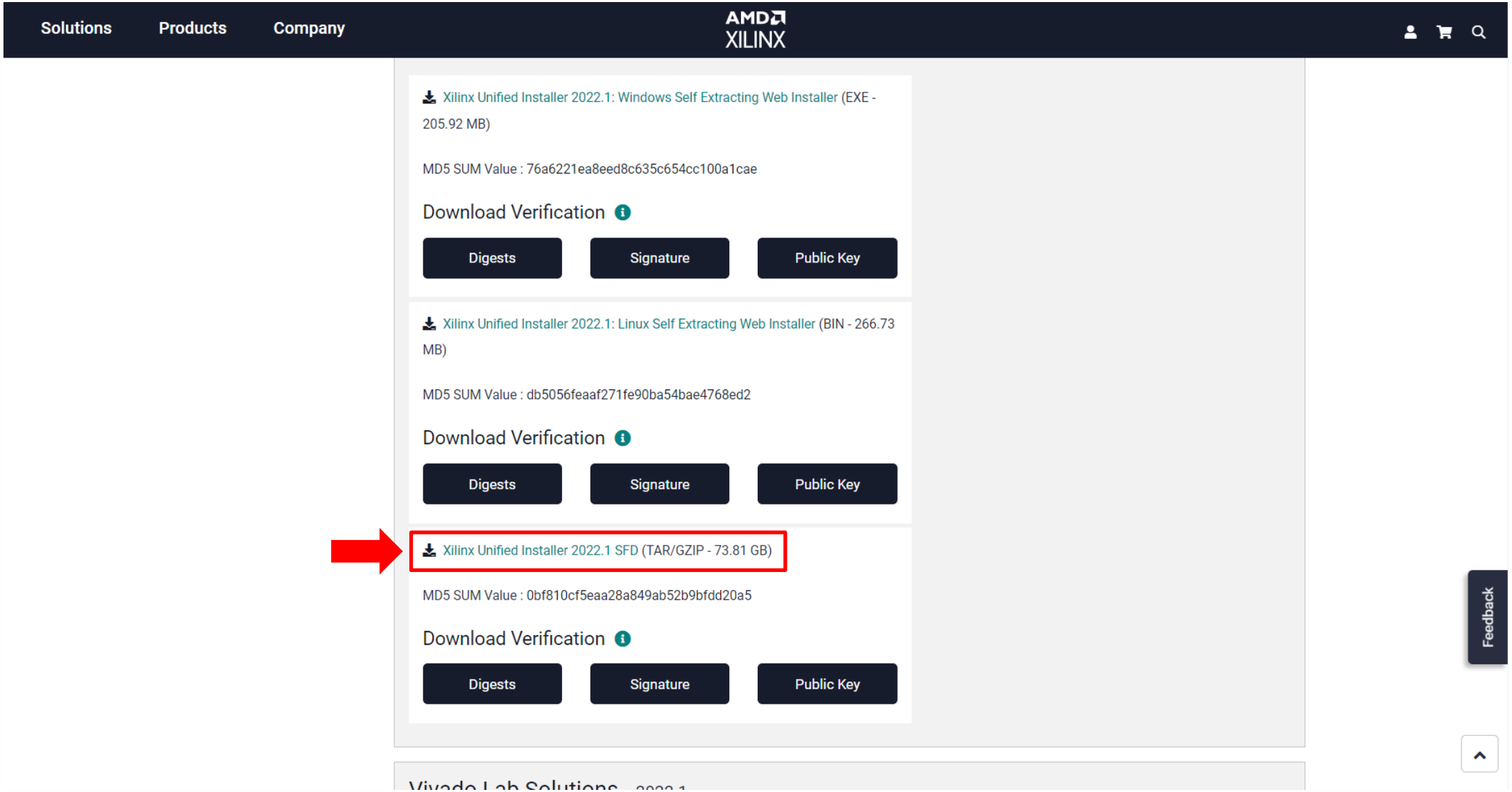
- RHEL 7 / Cent OS 7: 7.4, 7.5, 7.6, 7.7, 7.8, 7.9
- RHEL8/Cent OS: 8.1, 8.2, 8.3
- SUSE EL: 12.4, 15.2
- Ubuntu: 16.04.5 LTS;16.04.6 LTS; 18.04.1 LTS; 18.04.2 LTS, 18.04.3 LTS; 18.04.4 LTS; 20.04 LTS; 20.04.1 LTS

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

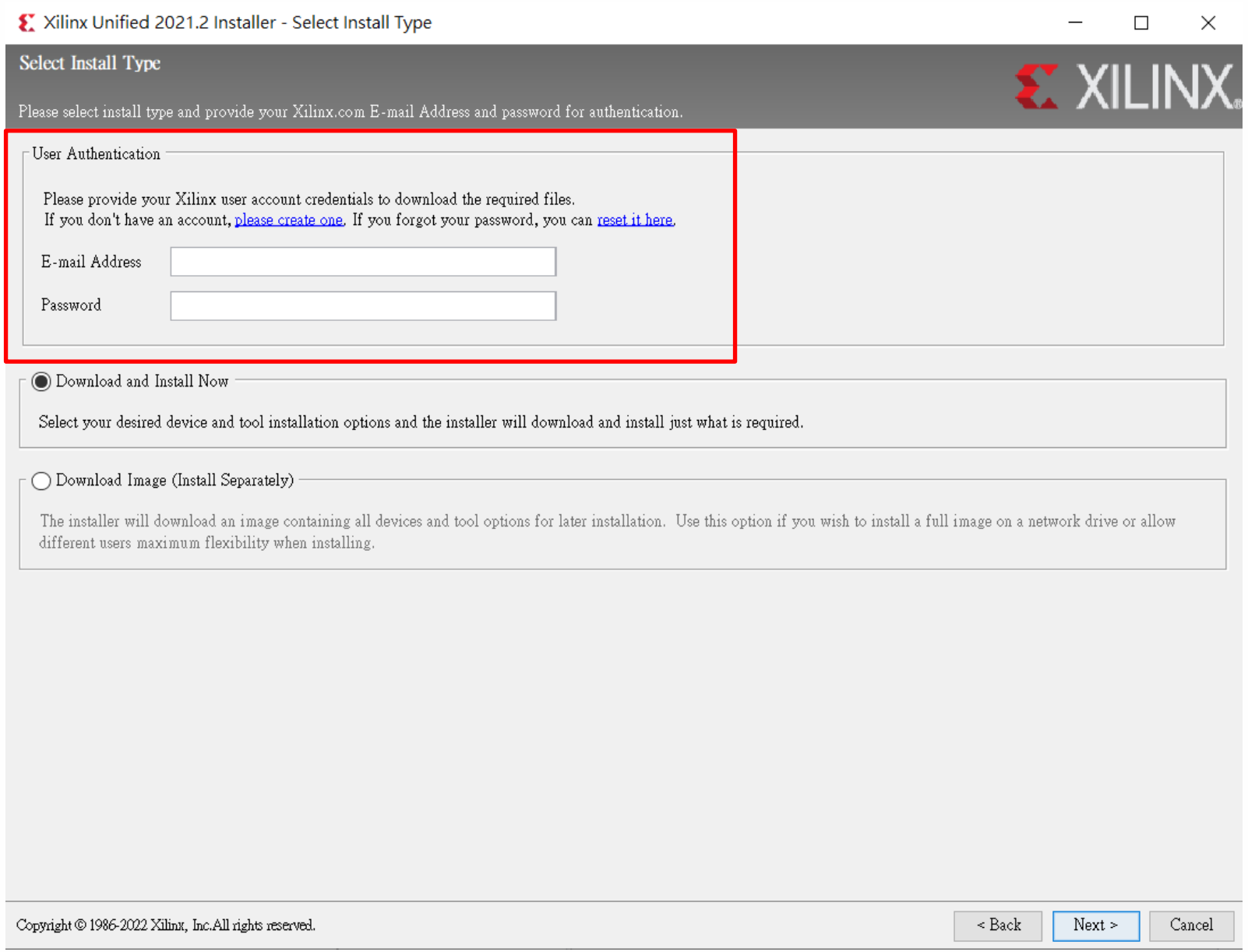
Vivado Installation



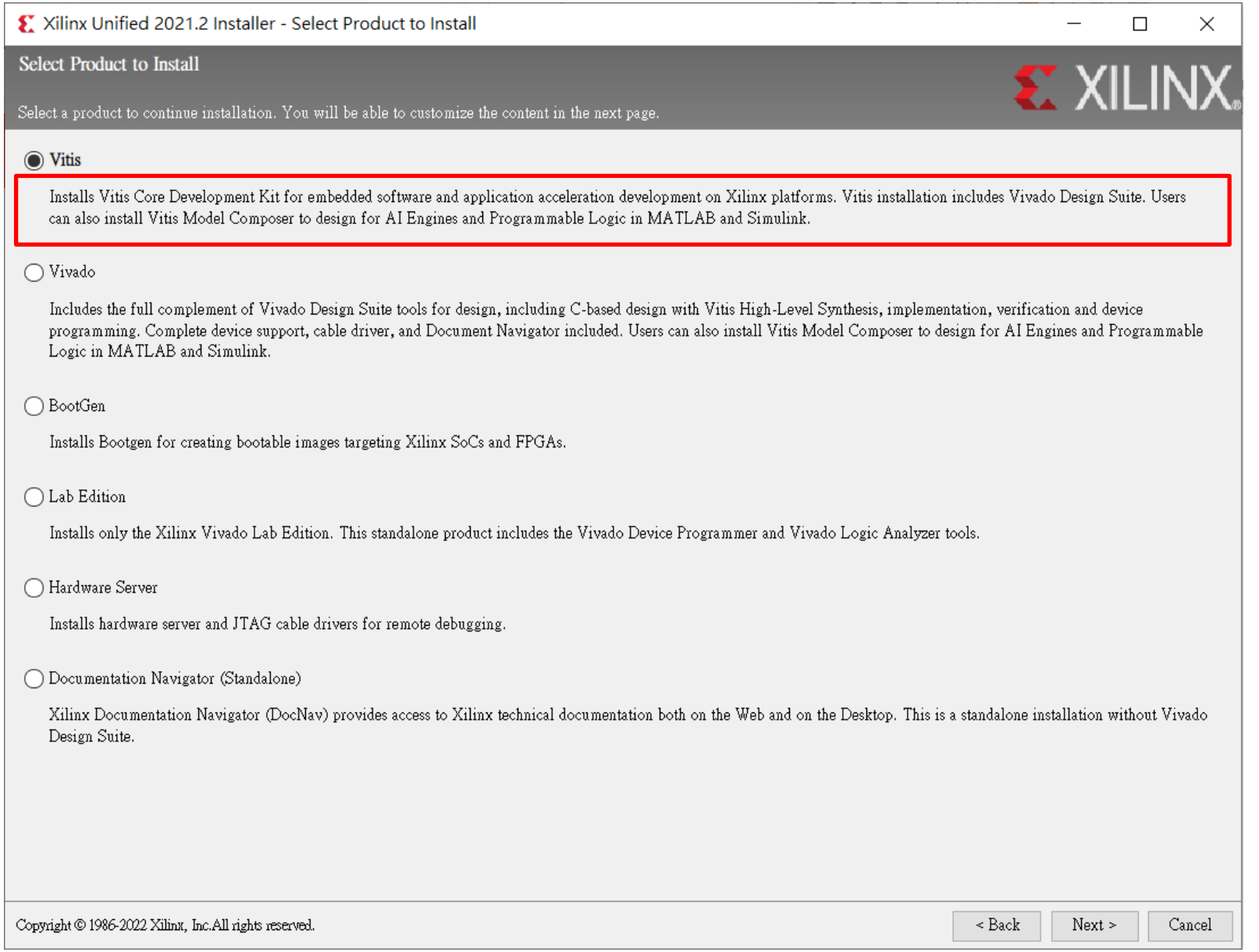
Vivado Installation



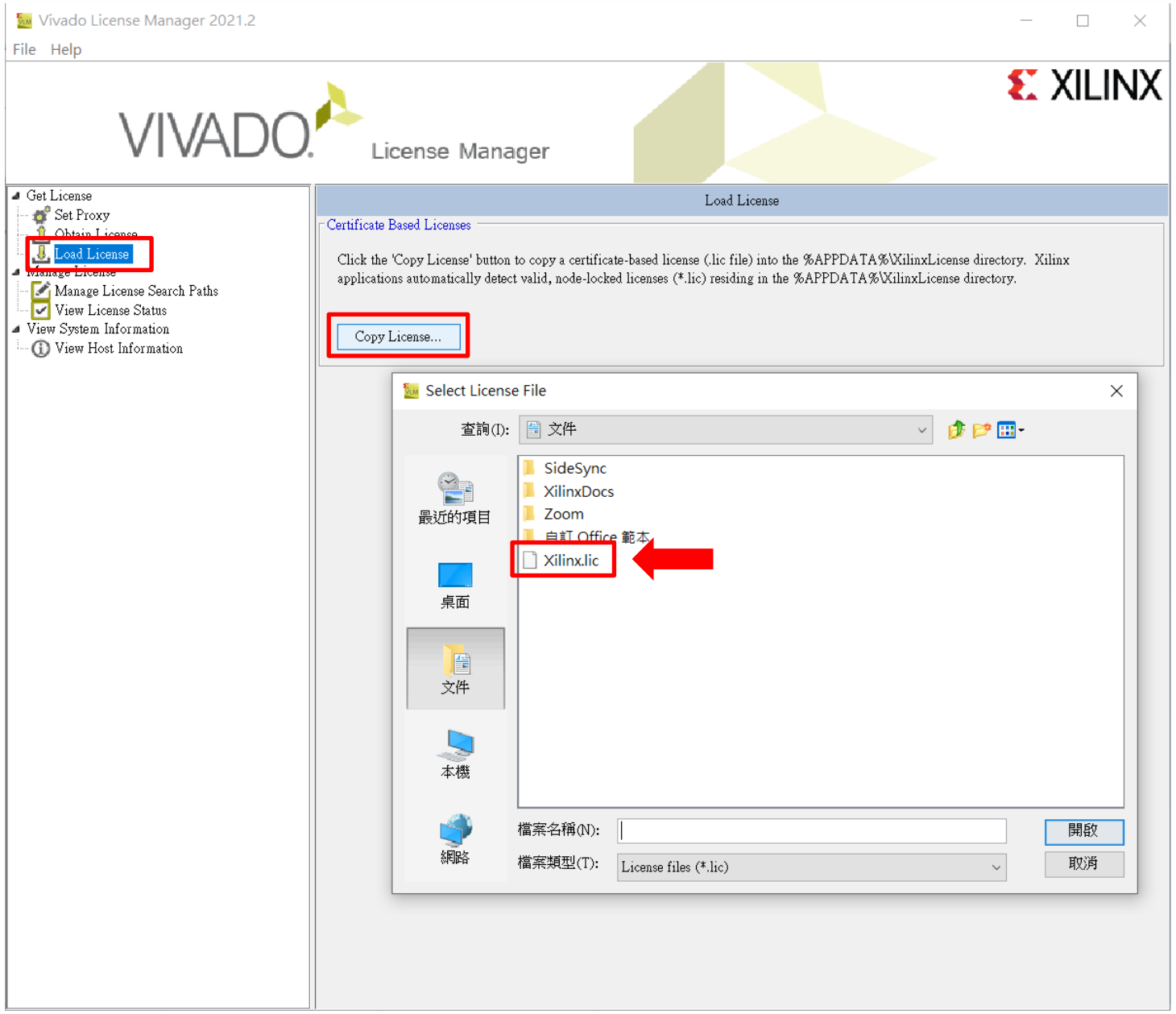
Vivado Installation



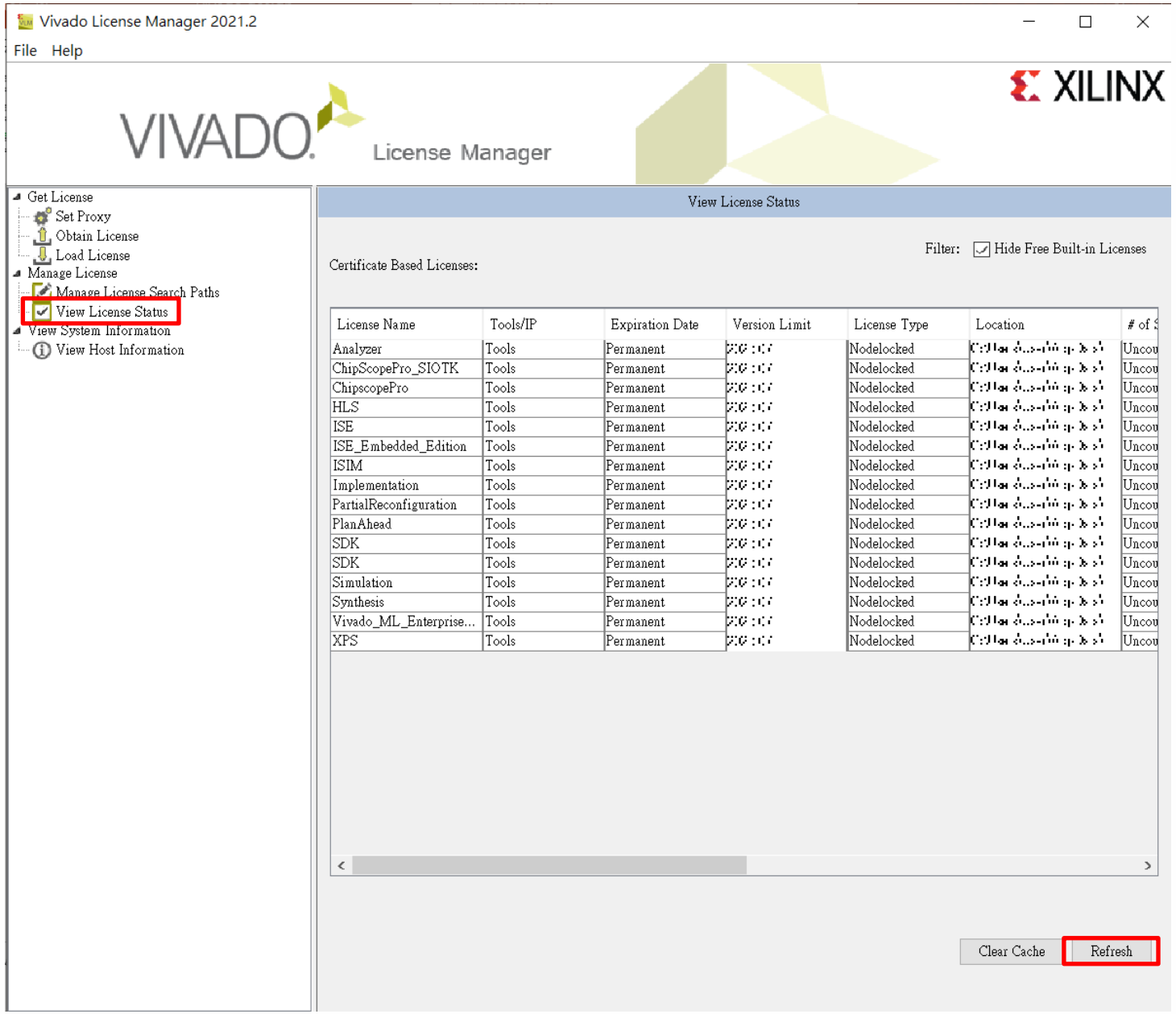
Vivado Installation



Vivado License Management



Vivado License Management

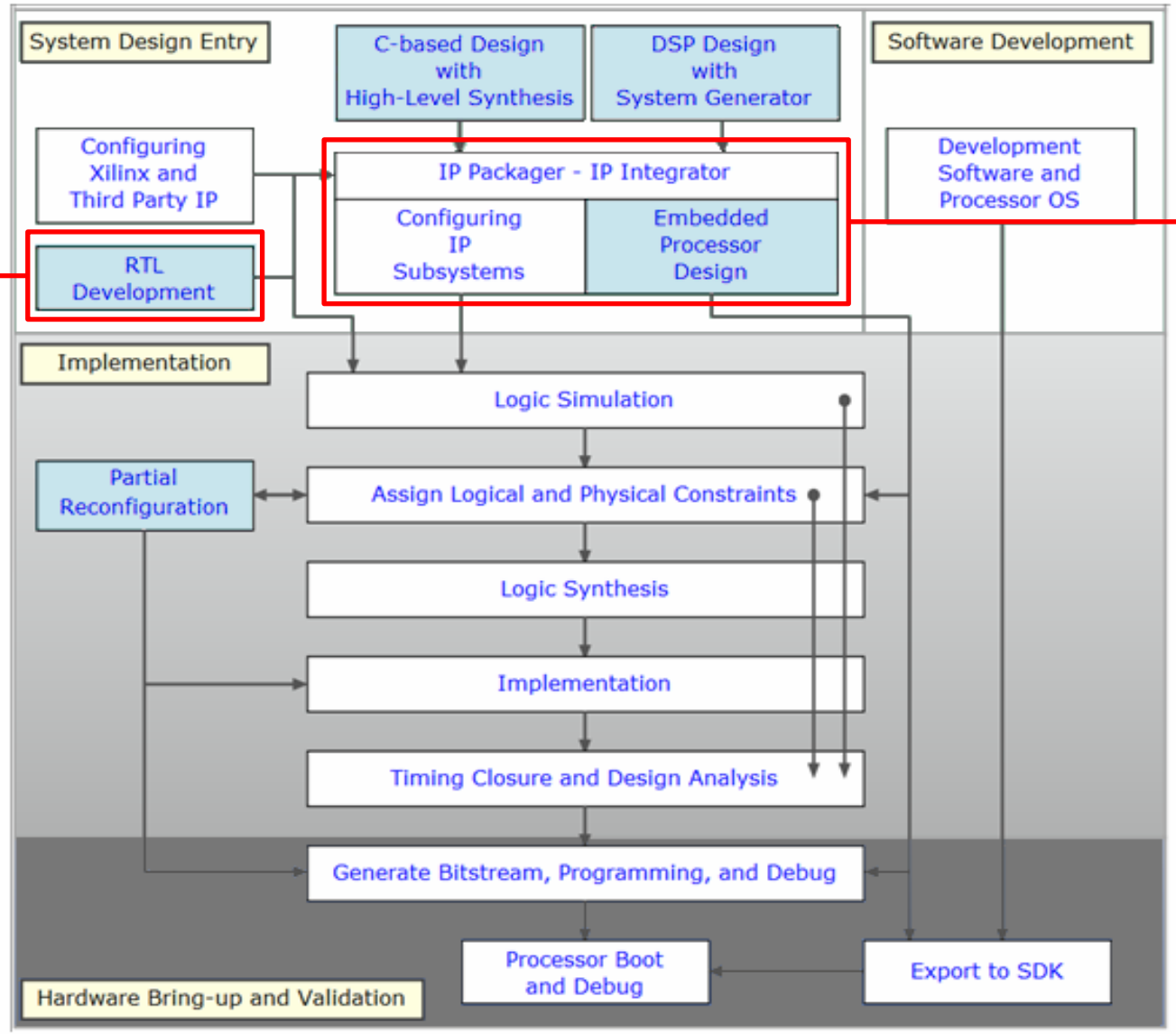


Vivado Design Suite Flow

XILINX.

Vivado Design Flow

Traditional RTL-bitstream



System-level Integration

- Interactive IP plug-n-play environment
  - AXI4, IP_XACT

Vivado Design Flow

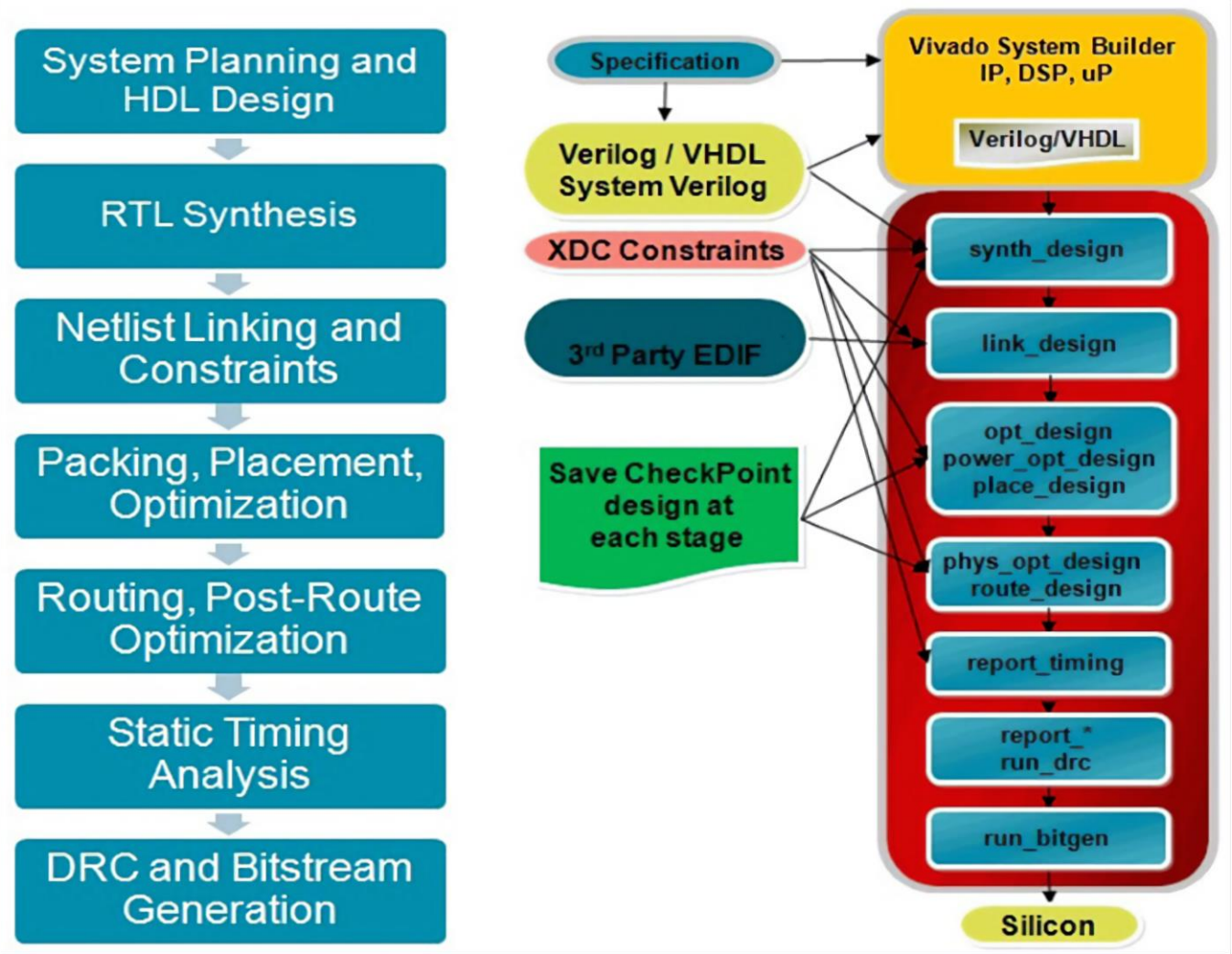
- Traditional RTL-bitstream

> 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

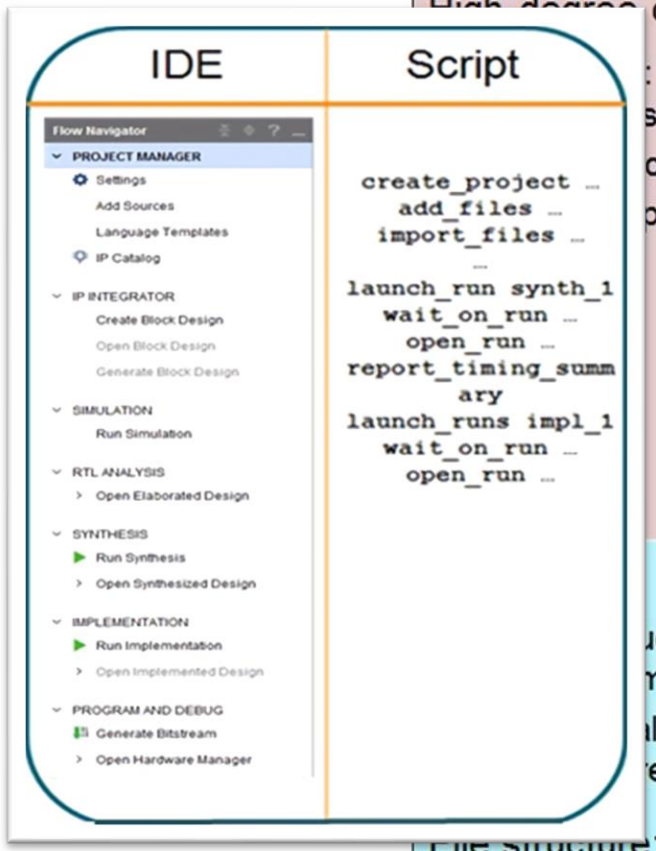
> Save checkpoint designs at any stage
  – Netlist, constraints, place and route results

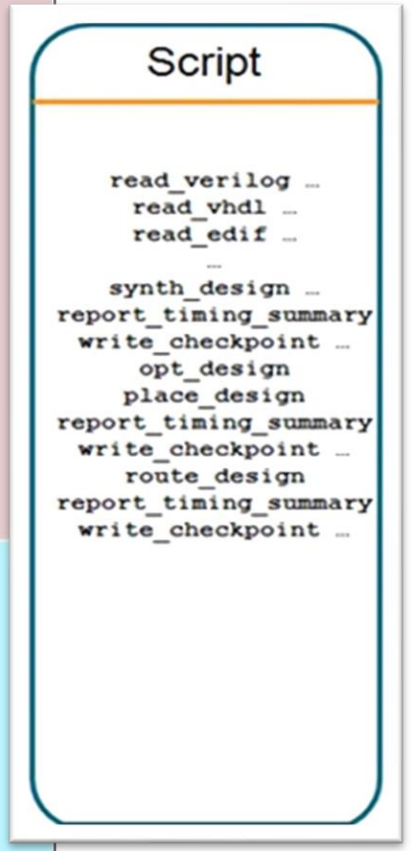


Vivado Design Flow

- Project vs. Non-Project

Project	Non-Project
High degree of automation : a single Tcl command – multiple steps ce and result management ple strategies: easy run	Fine-granularity commands • More efficient scripts. Enable focusing on specific challenges • Easy access to all features (ex: re-entrant routing) File structure: fully customizable • Sources and result files • Easy adoption to revision control systems Runtime: potentially faster
uced flexibility to work with mediate implementation results ll features are easily accessible e-entrant routing) File structure: fixed file organization • Revision control – many project files	Requires more in-depth knowledge You are responsible for • Dependency file management • Saving key result files (ex: checkpoints)

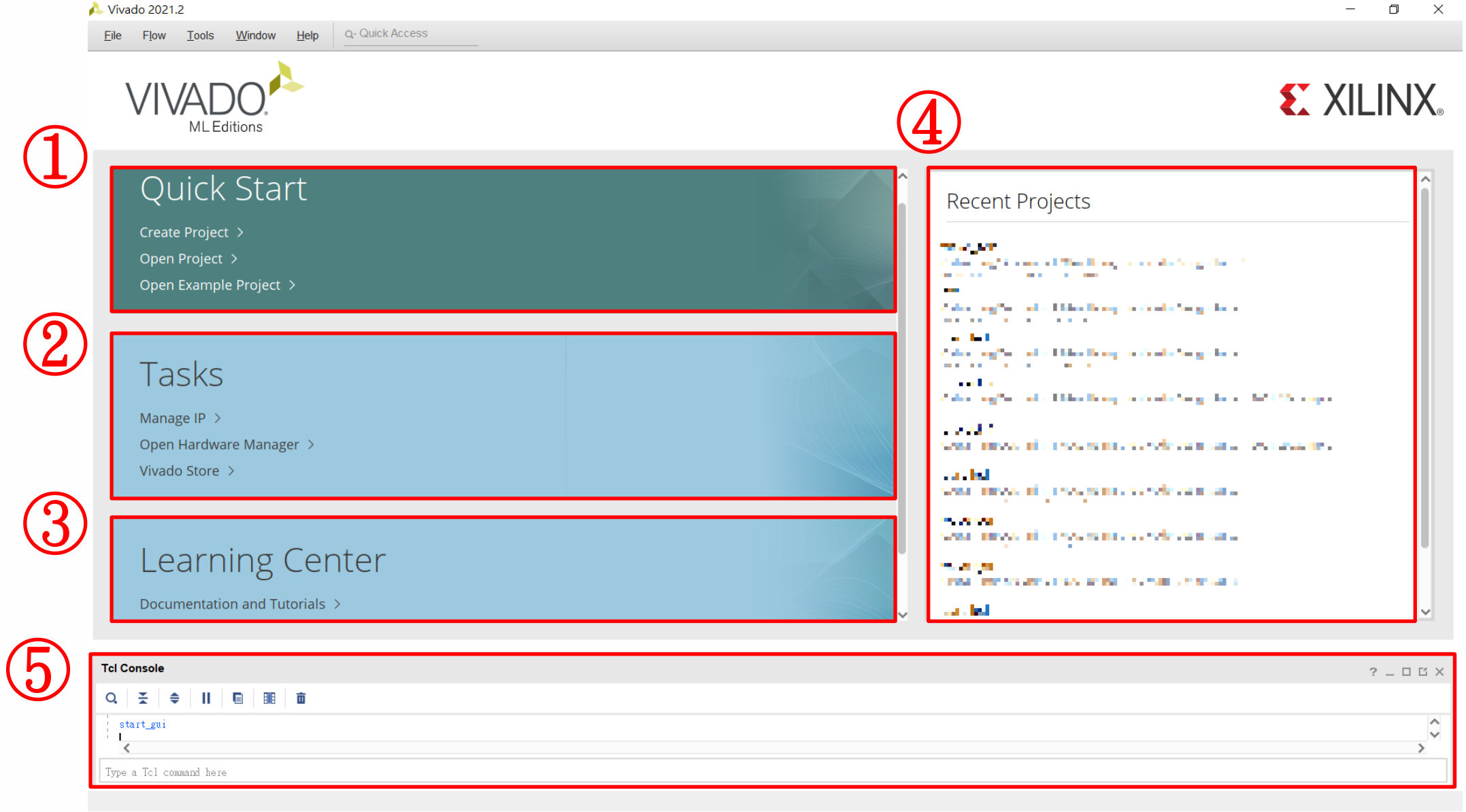




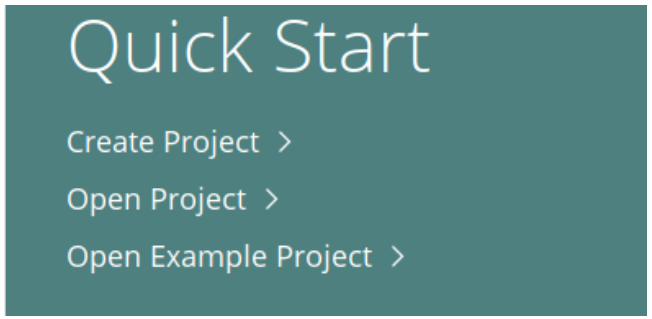
Introduction of Vivado Basic Interface

XILINX.

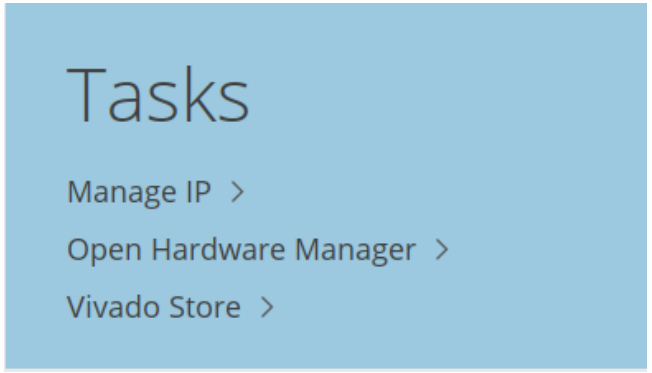
Vivado Start Page



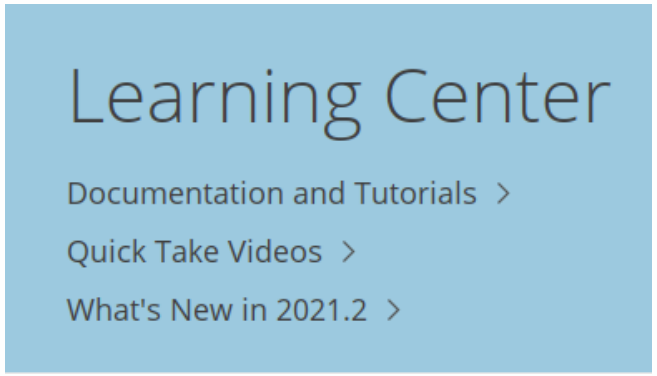
Vivado Start Page



Create new or open existed project



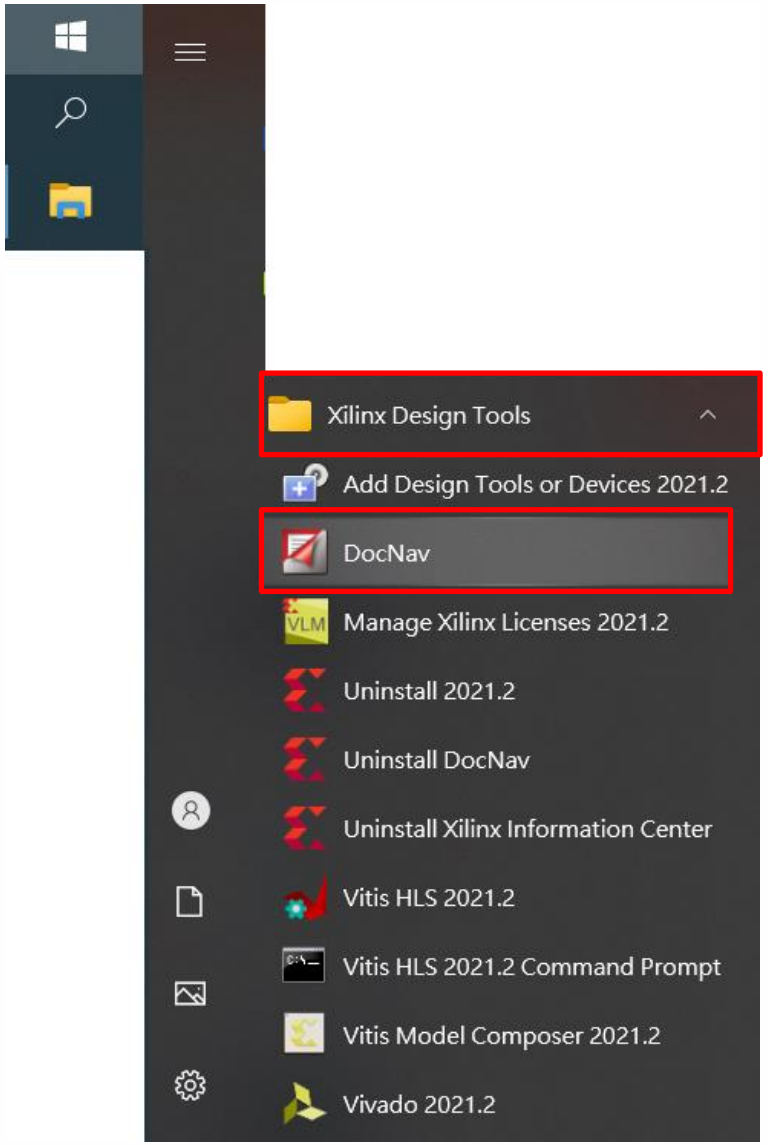
Create new or open existed IP or get some successful project in Vivado



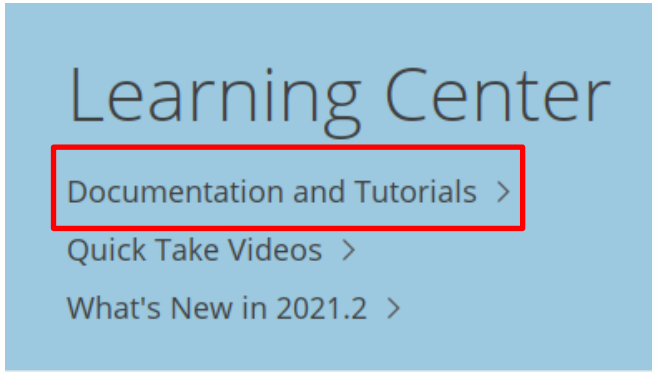
Tutorials in image, text and video form, and show the release notes which contains What's new, Known Issues, and Important Information

Xilinx Documentation Navigator

①

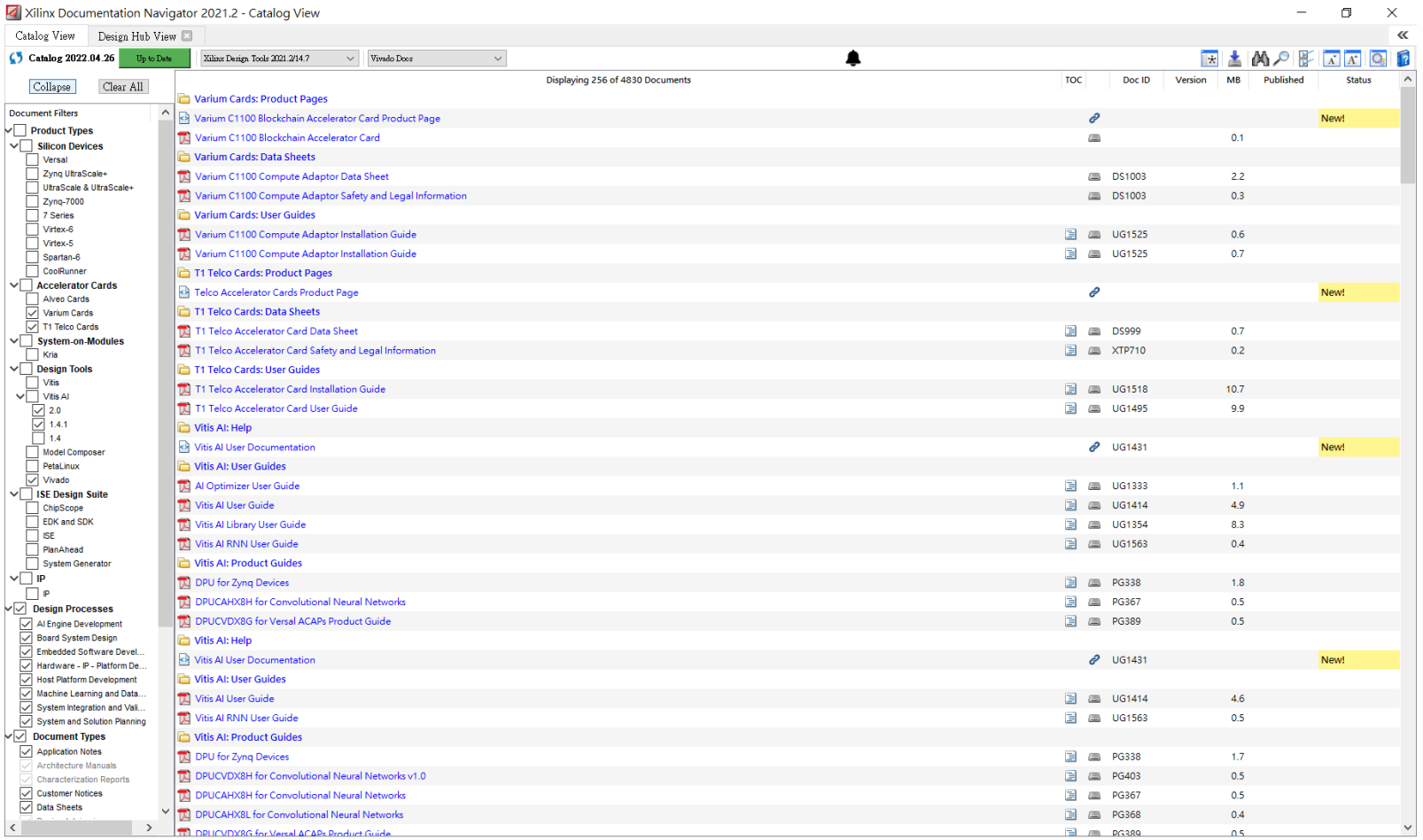


②



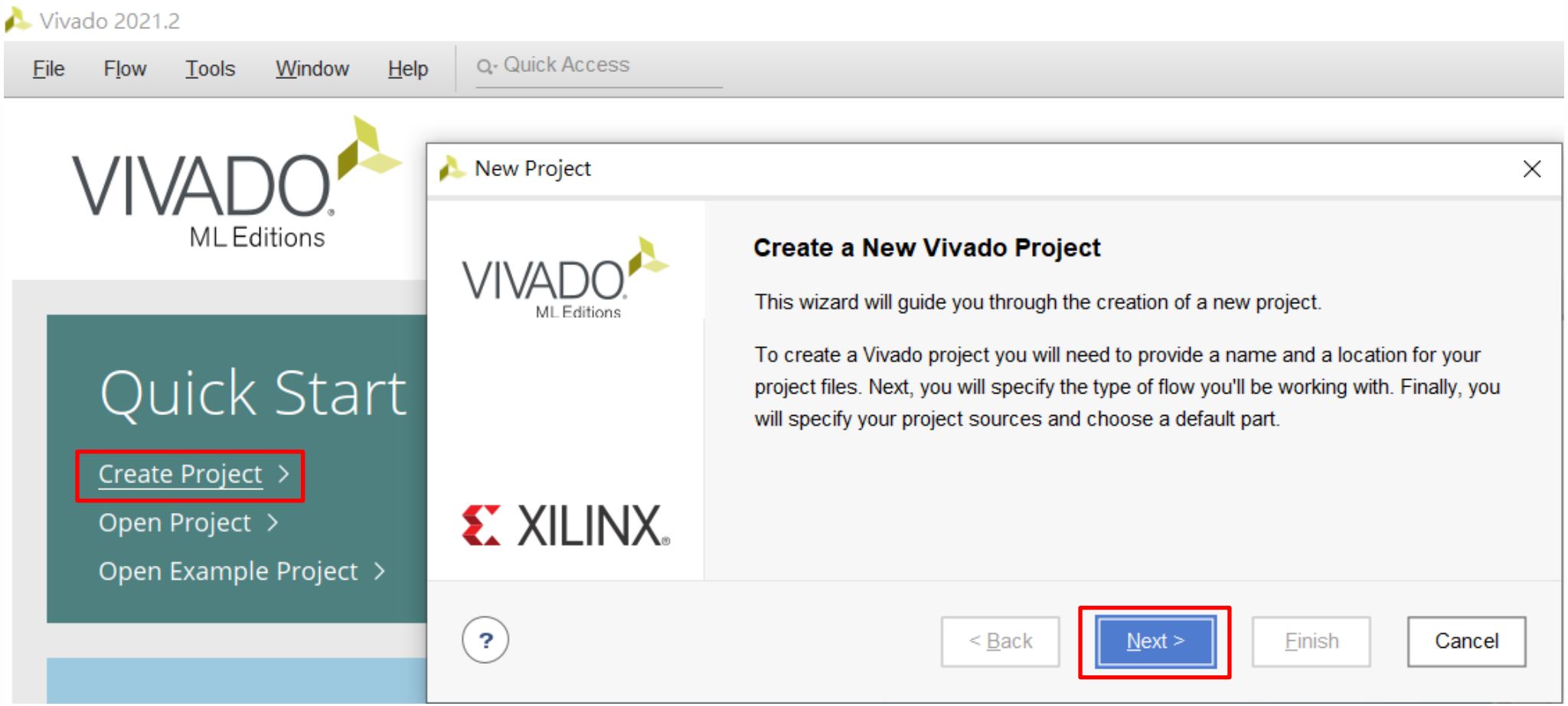
Xilinx Documentation Navigator

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



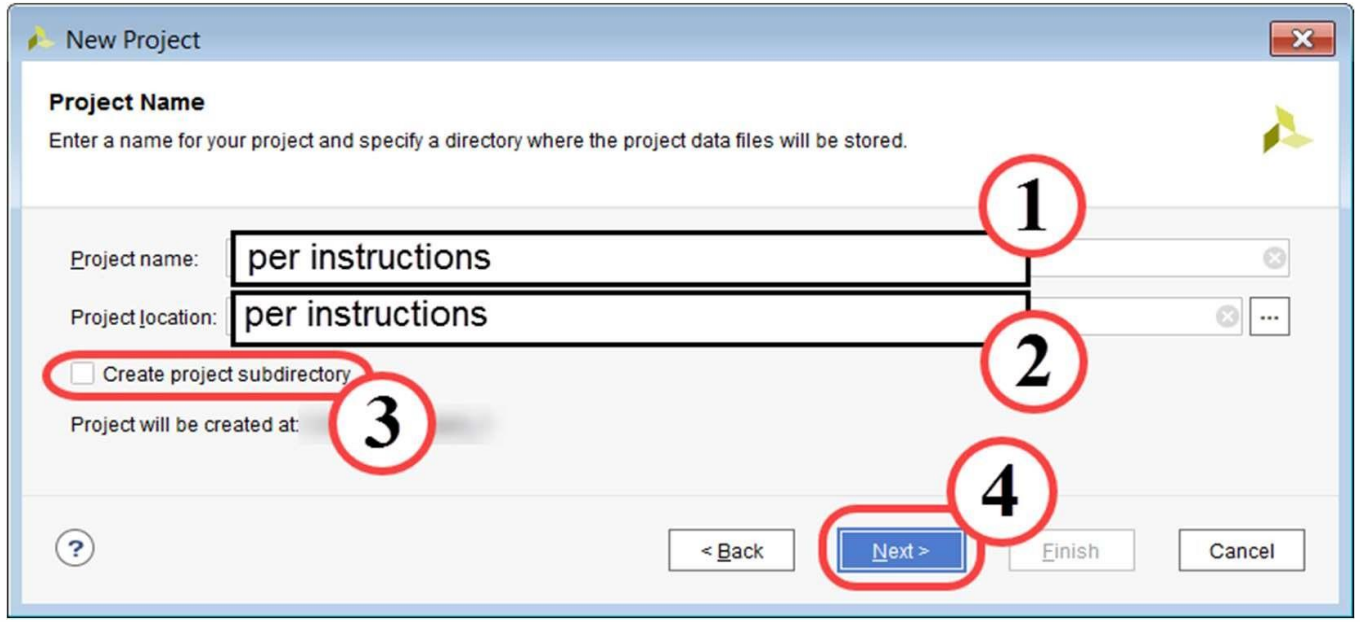
New Project Creation Wizard

- Create Project



New Project Creation Wizard

- Project Name



① Enter Project name

② Enter location in the Project location field

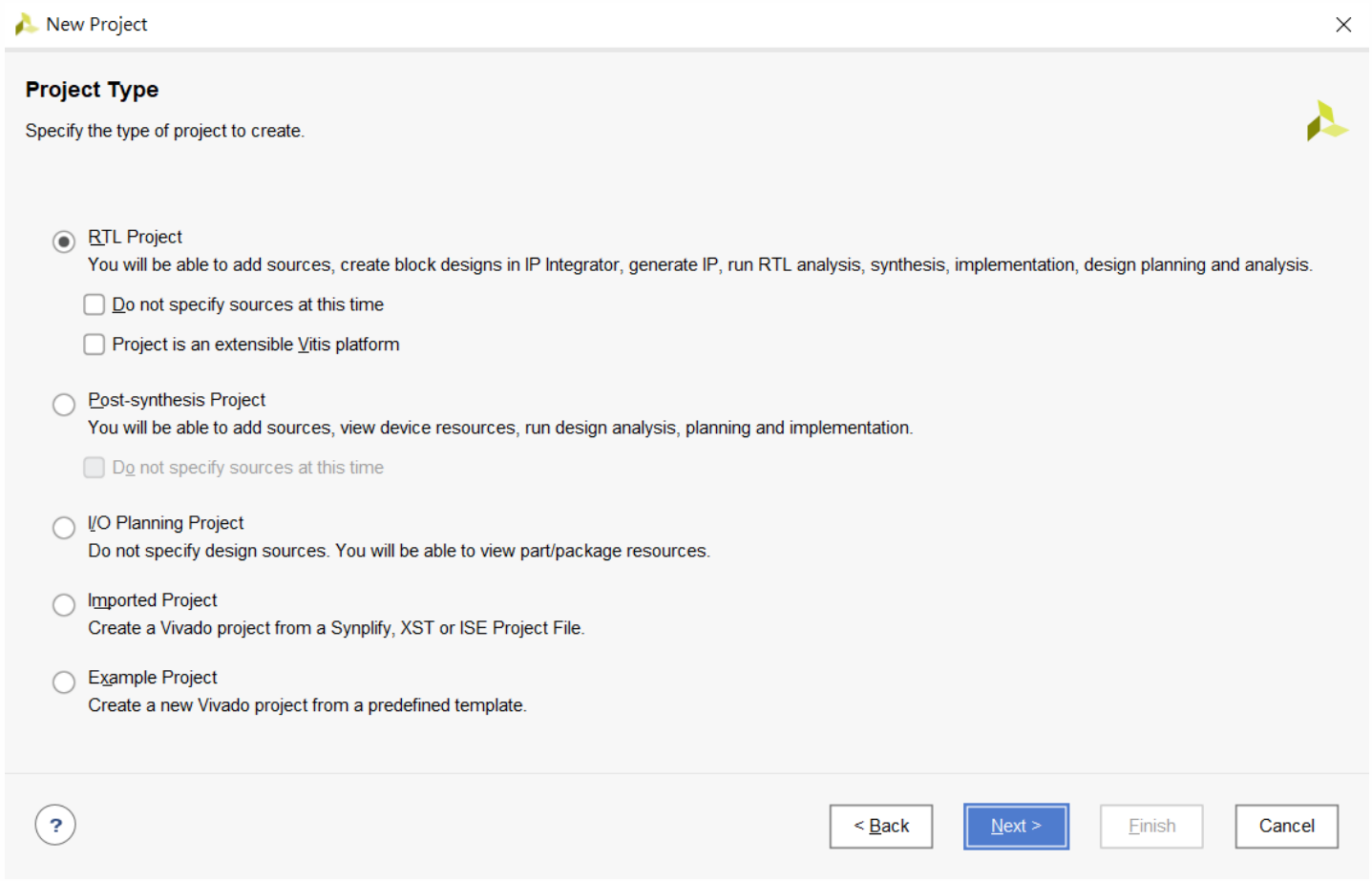
③ Select this option will create the folder with Project name, if you do not want to create a new folder, deselect it

④ Click “Next” to continue the project creation

New Project Creation Wizard

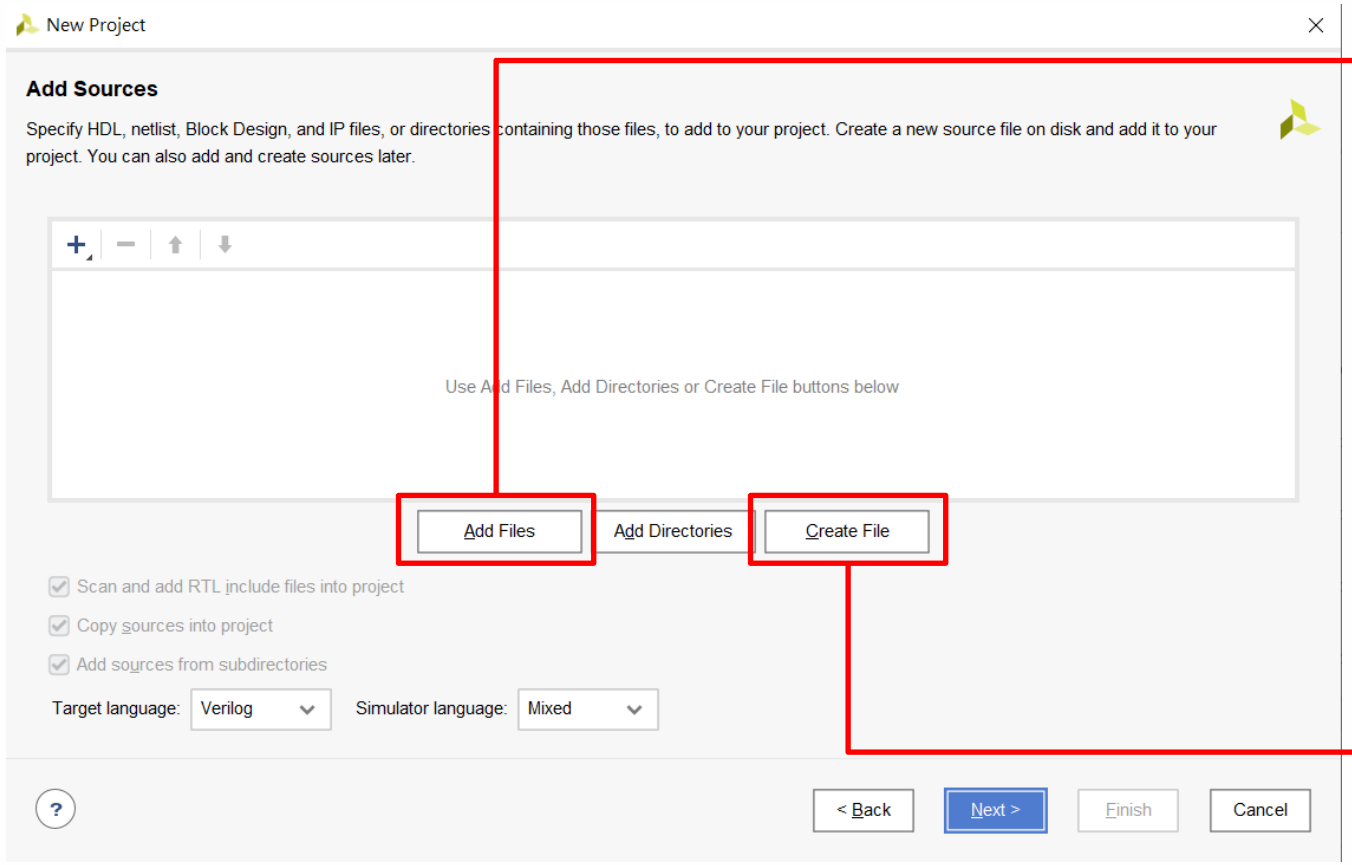
- Project Type

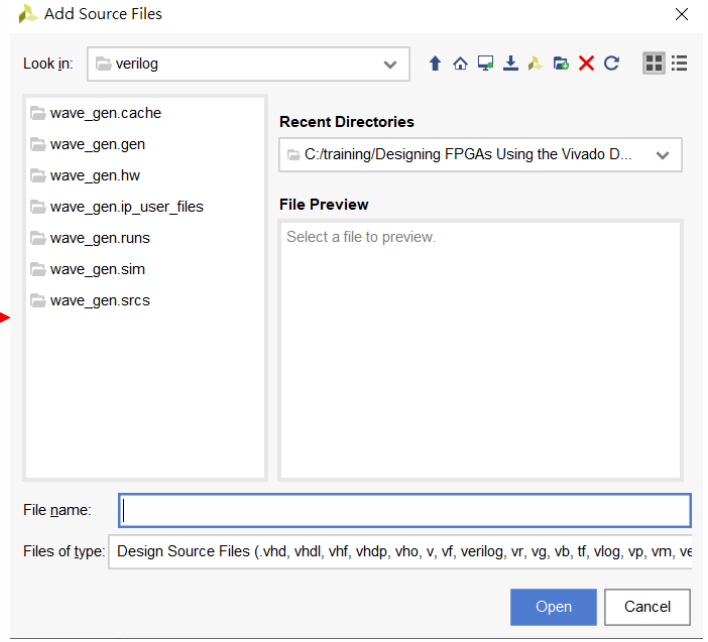
> Five different types of project can be created
  - **RTL**
    - Using HDL source files
  - **Post-synthesis**
    - EDIF or NGC
    - Design checkpoint
  - **I/O planning**
    - For early pin testing
    - No design sources
  - **Imported project** from Project Navigator, XST, and Synplify projects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

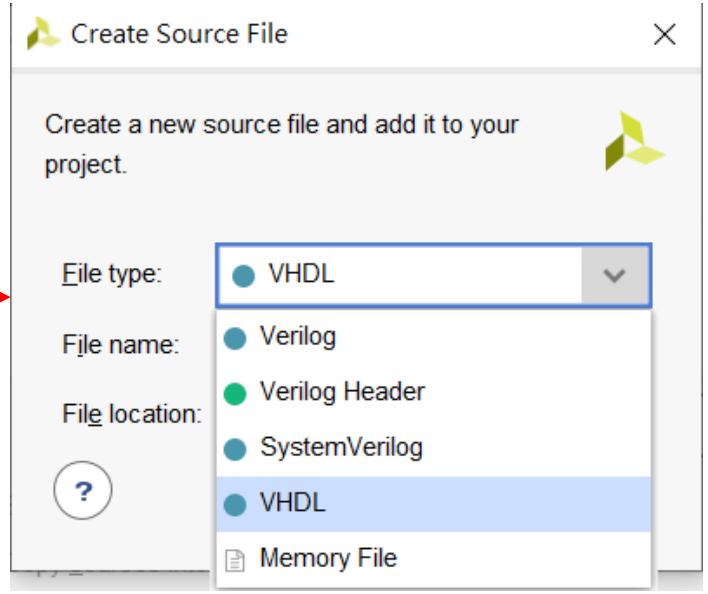


New Project Creation Wizard

- Add Sources



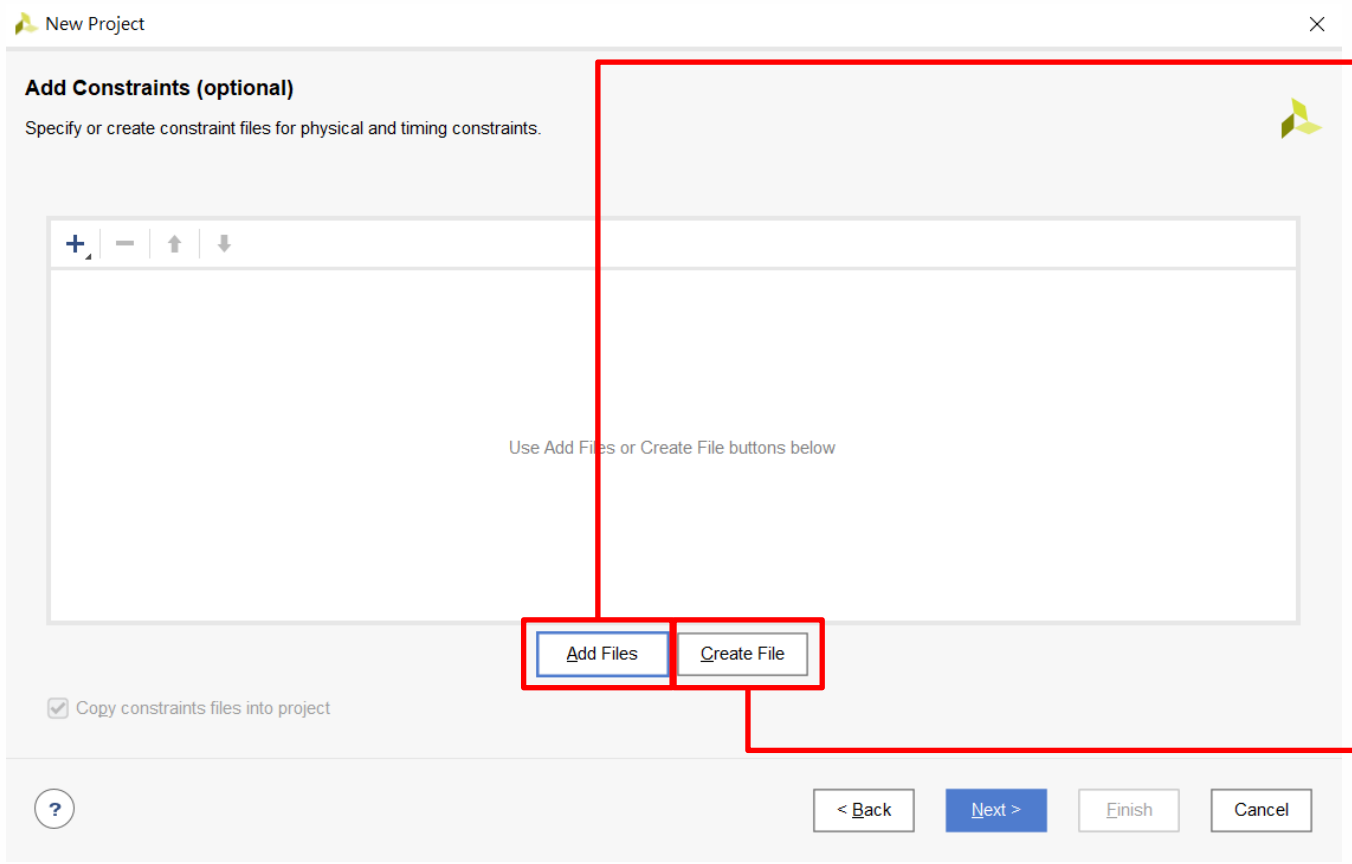


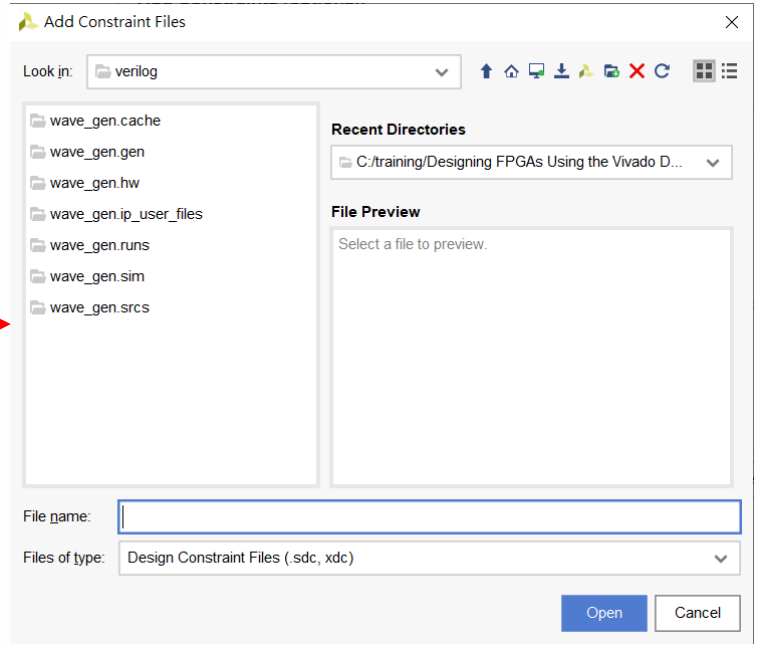


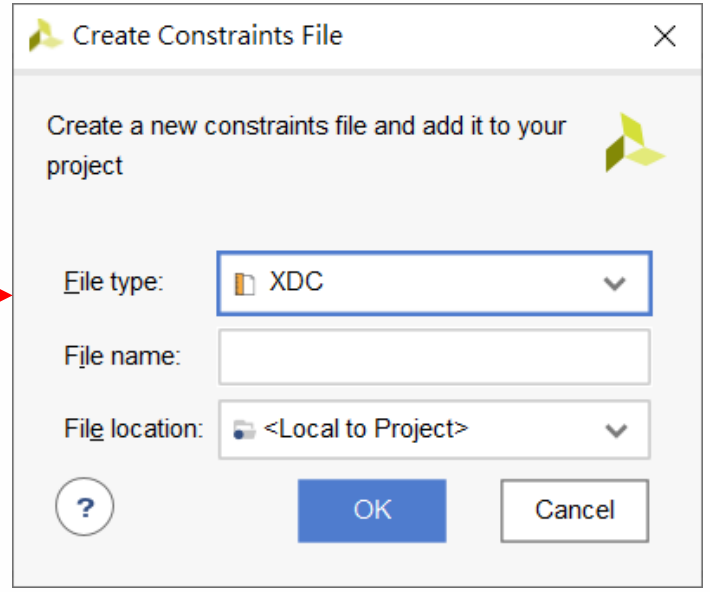
XILINX.

New Project Creation Wizard

- Add Constraints



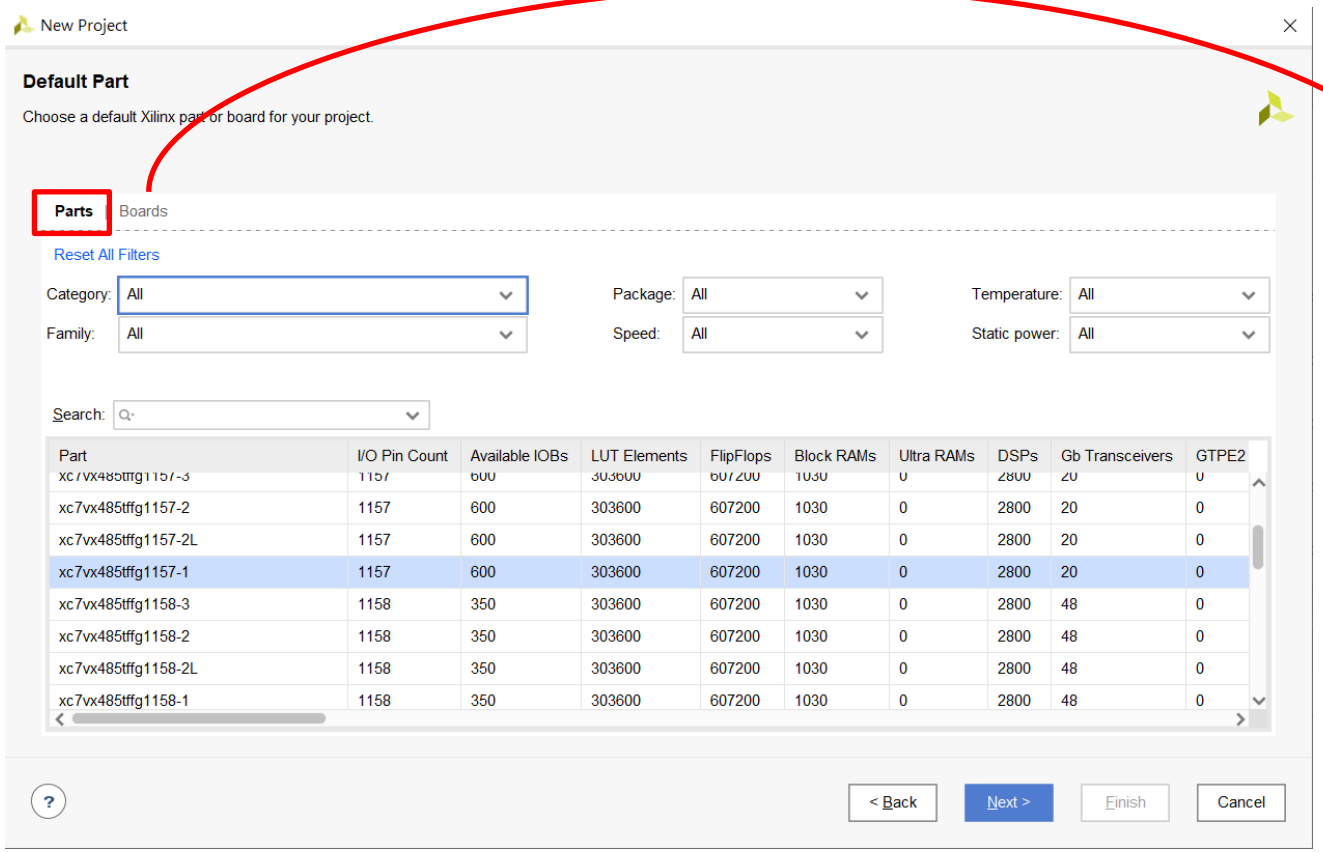




XILINX.

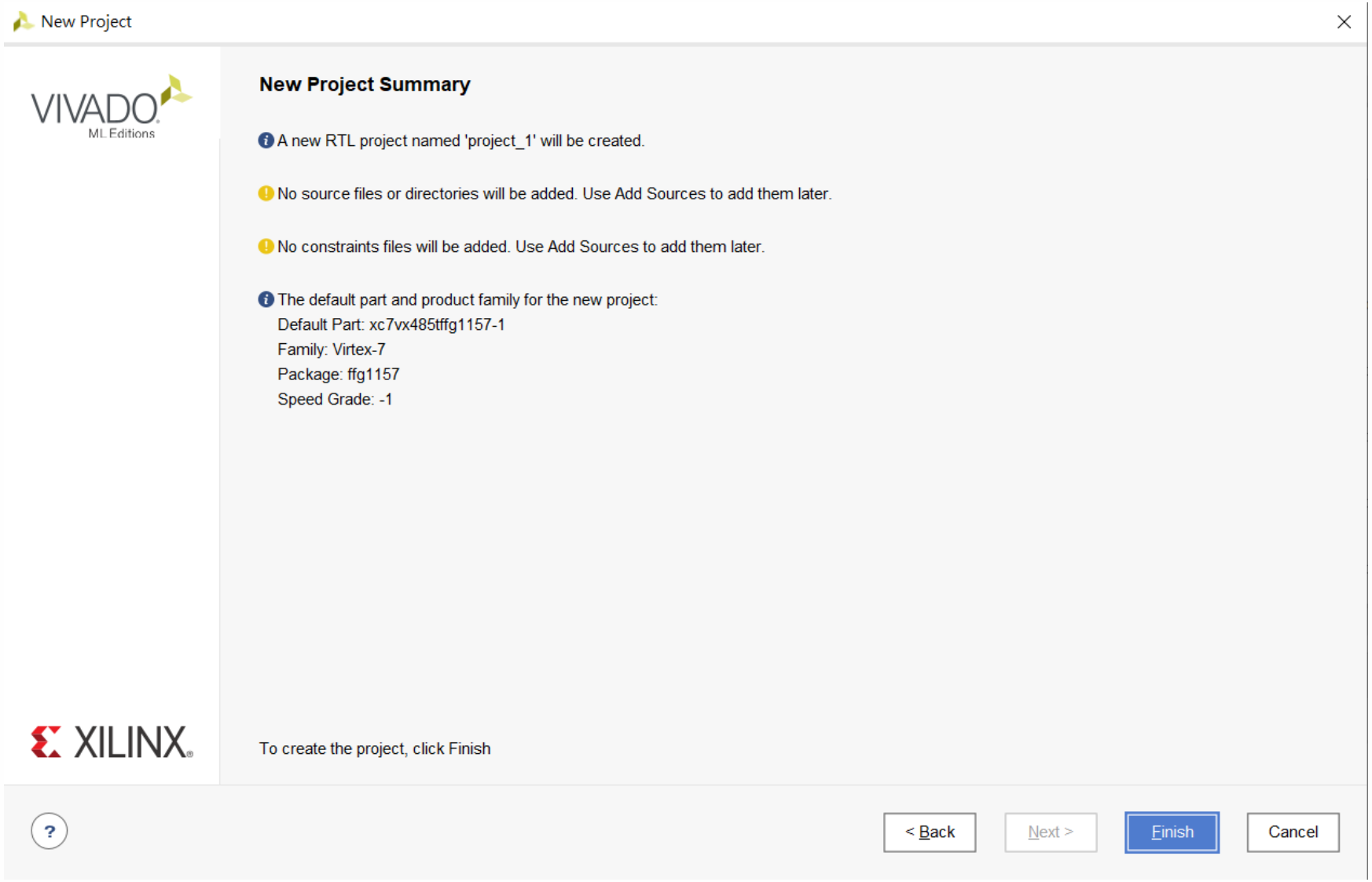
New Project Creation Wizard

- Choose a Xilinx Part or a Board



New Project Creation Wizard

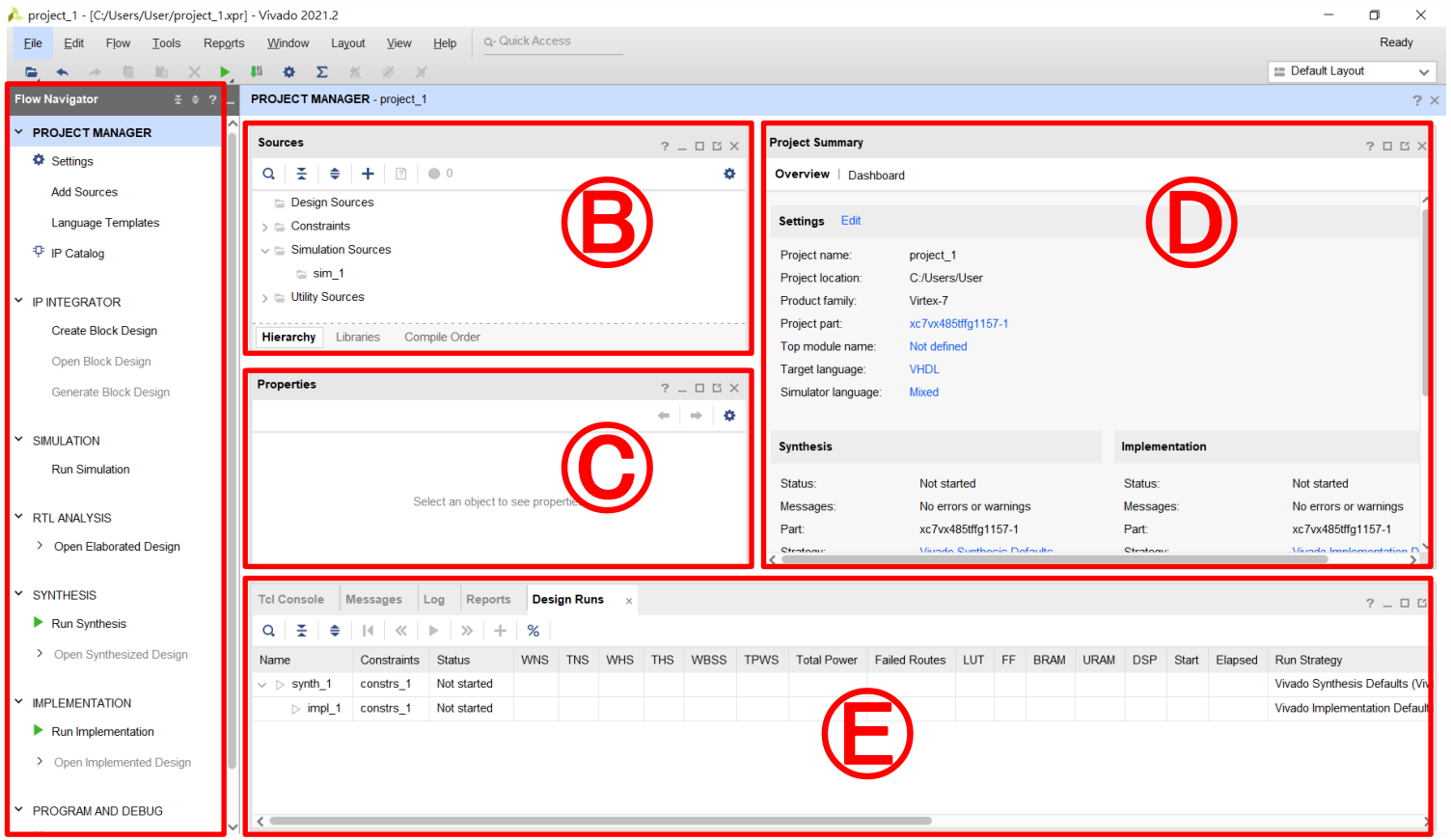
- Finish the project creation



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)

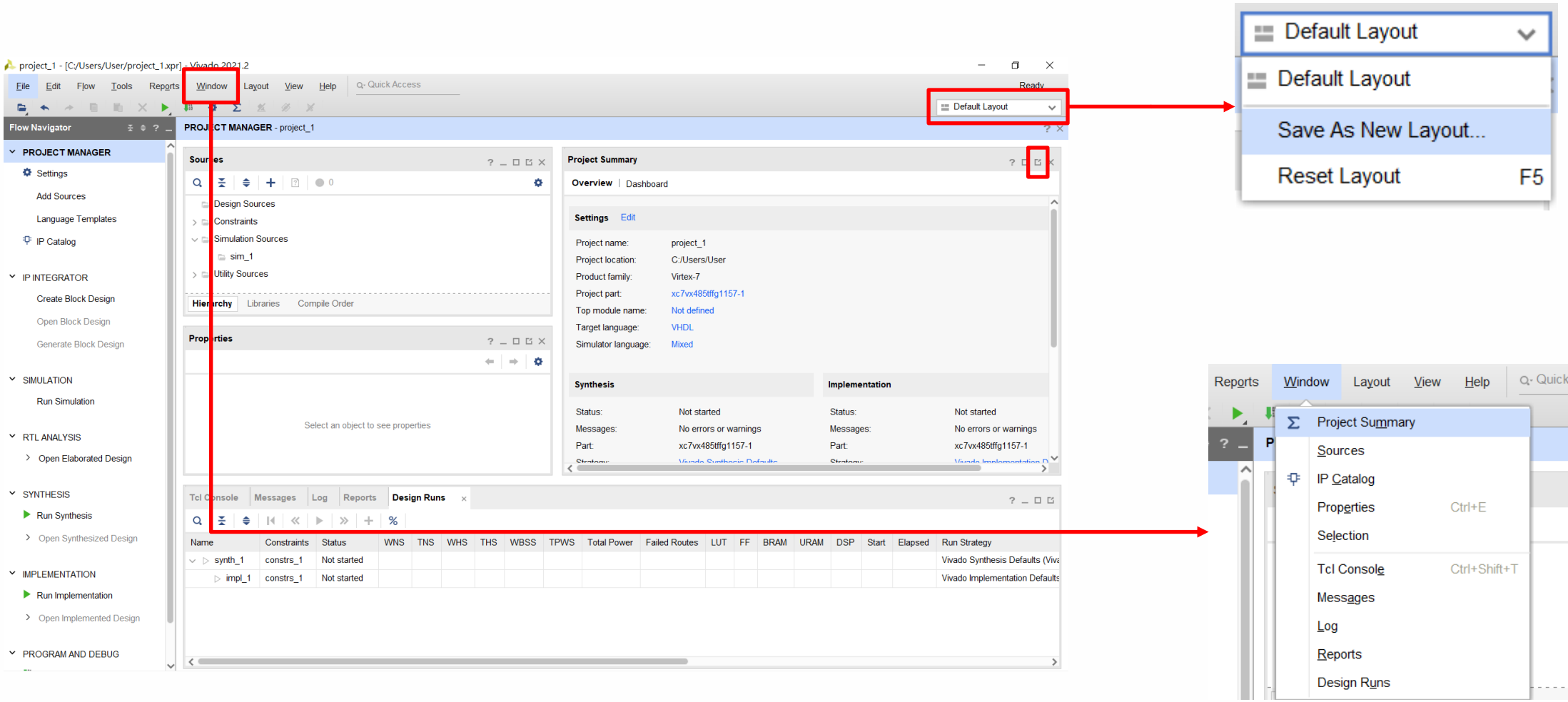
  Project Summary
  - Gives access to device utilization (resources), timing summary, and strategy information
  - Enable you to view and analyze all of this information in one place
- E: Console
  - Tcl console, Messages, Log and Reports.



XILINX.

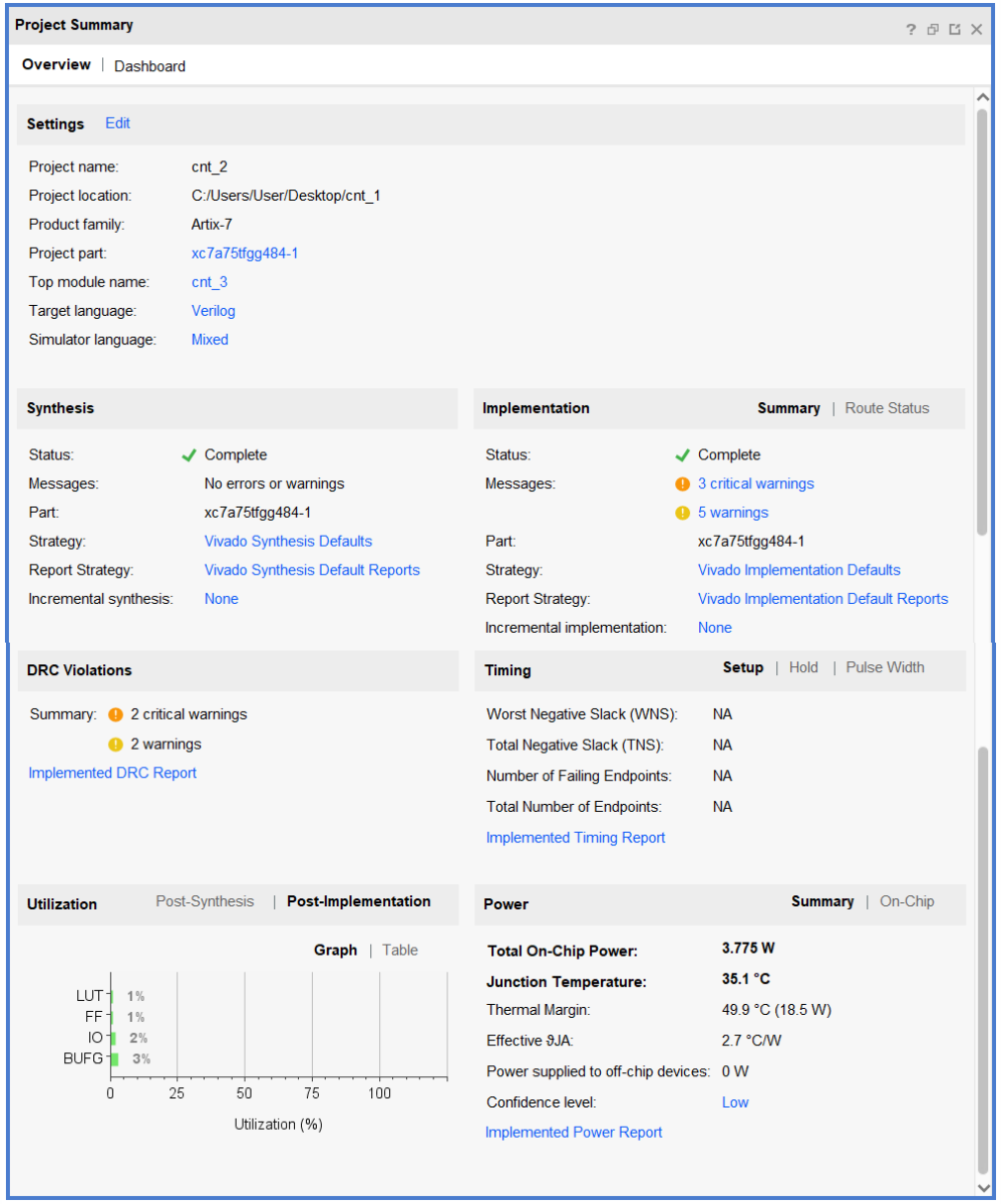
Integrated Design Environment (IDE)

- Reset layout or Restore window



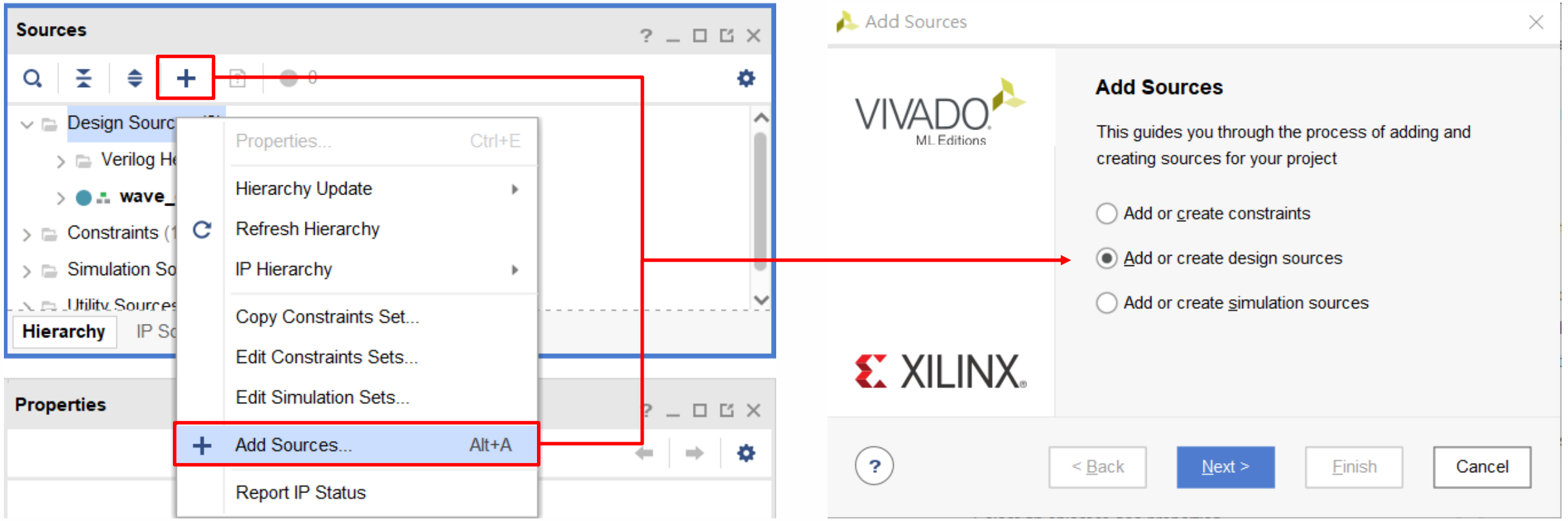
Integrated Design Environment (IDE)

- 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)

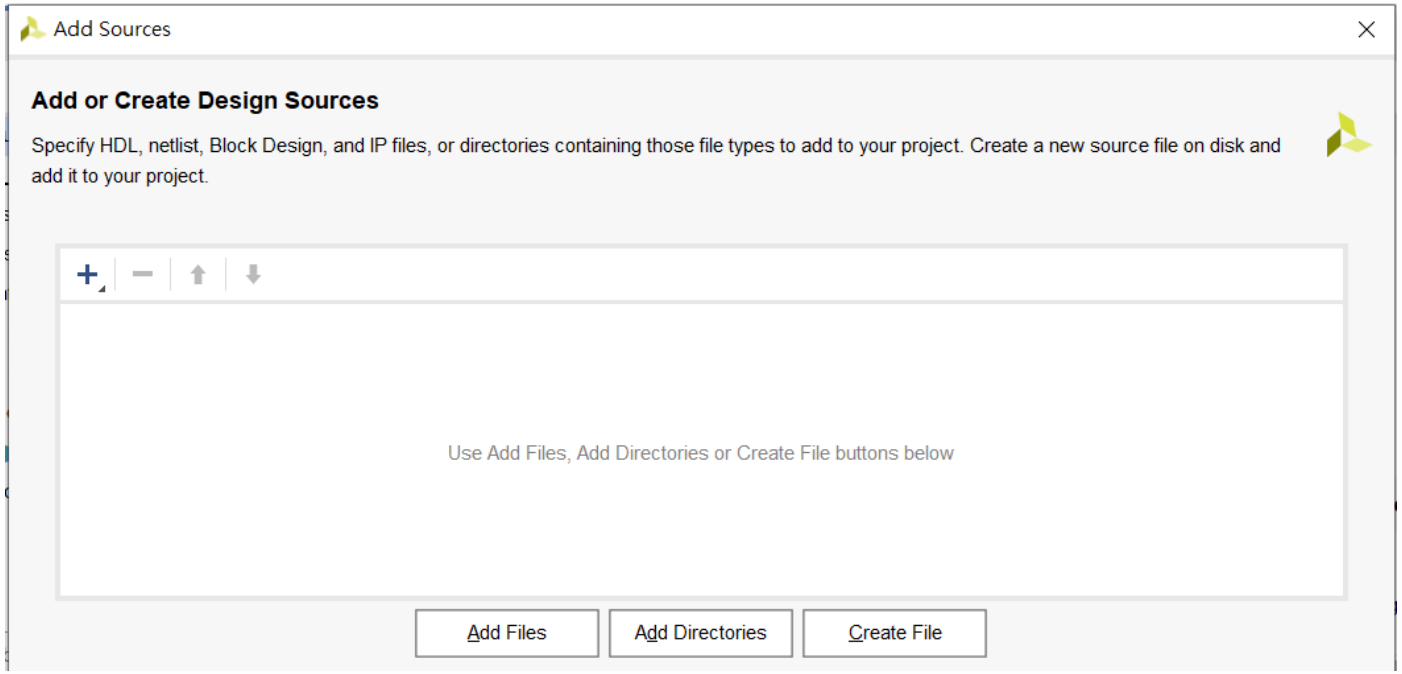
- Create Design Source file



Integrated Design Environment (IDE)

- Create Design Source file

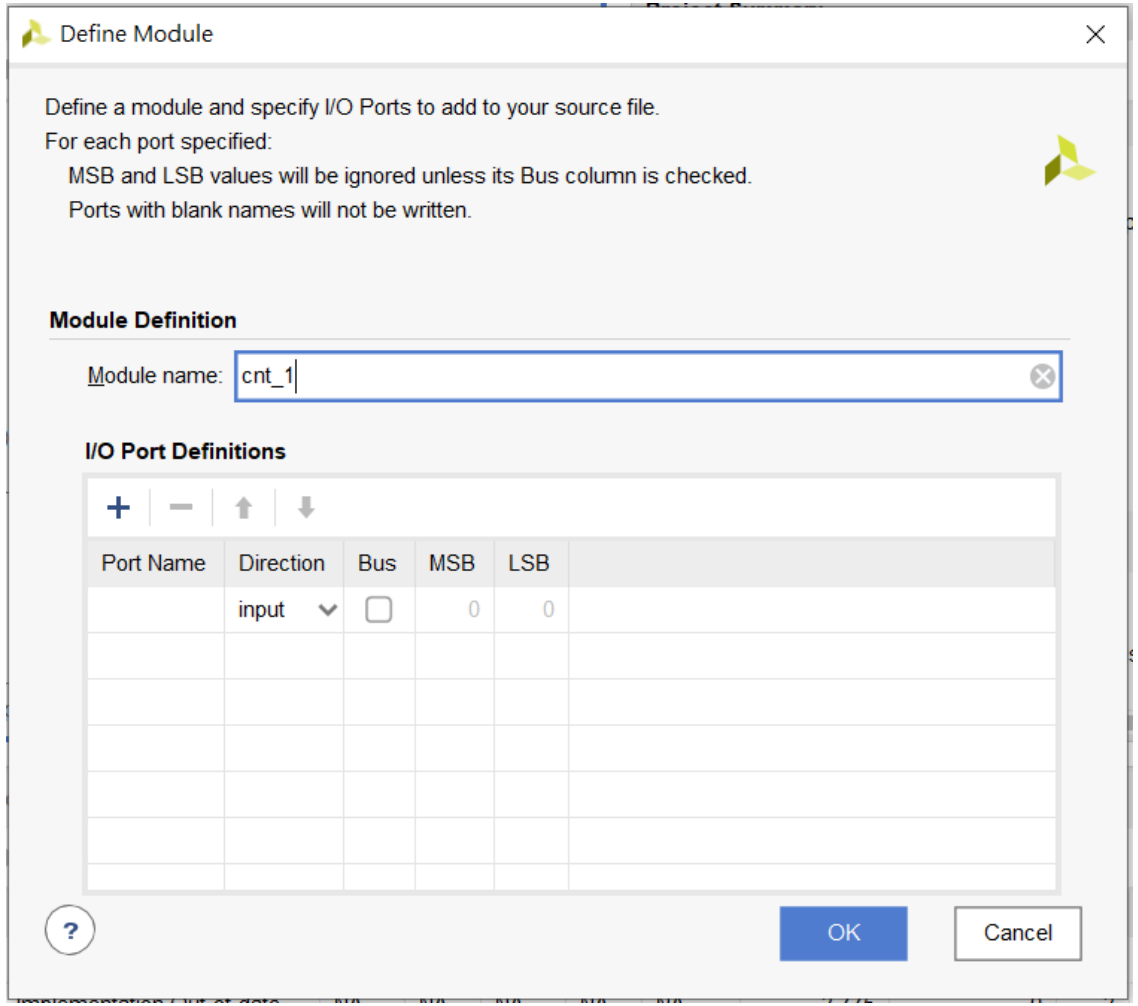
After selecting “Add or create design sources”, you can choose add existing design source files or create a new one.



Integrated Design Environment (IDE)

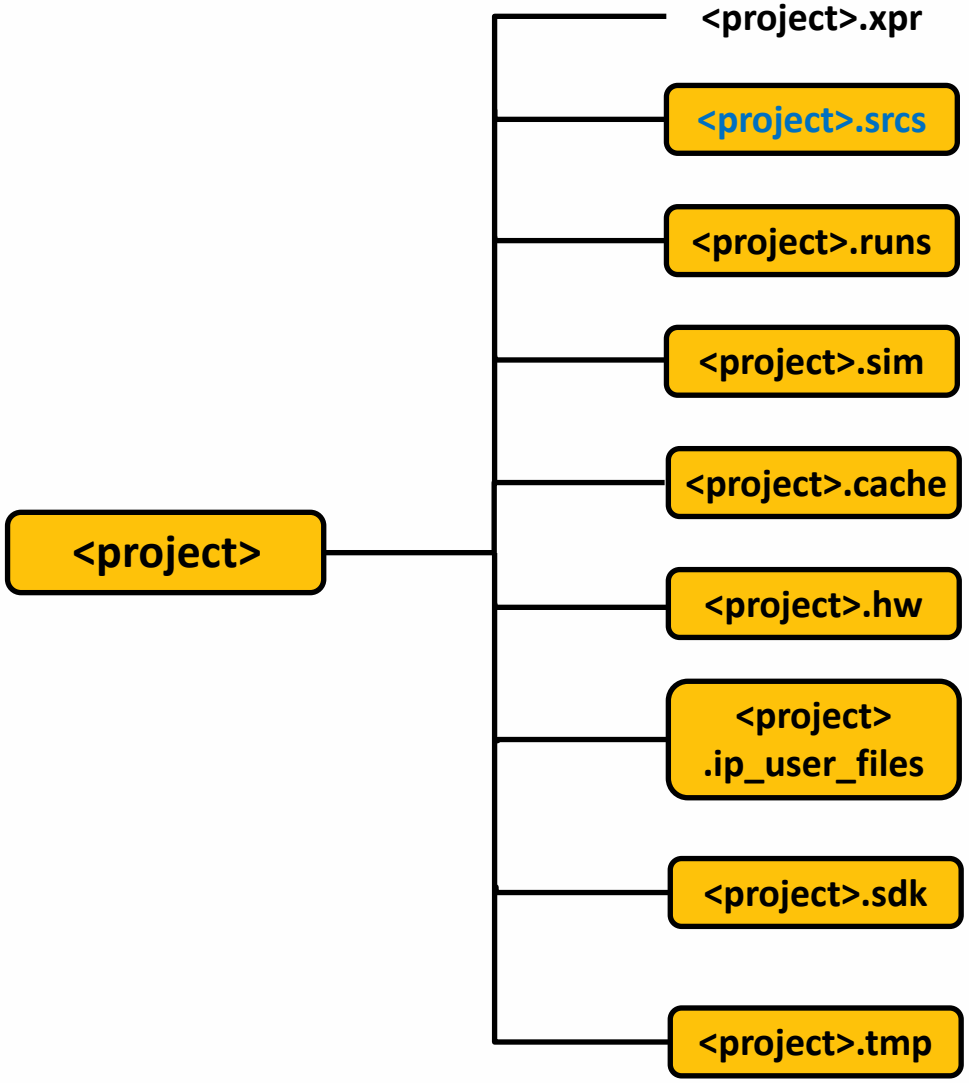
- Create Design Source file

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



Startup File for Vivado Project

Contains Sources, Simulations and Constraints filesets

Temporary File for compiling, synthesis and place & route. It also has .bin and .bit files which are executable after compiling in impl_l directory

The results of simulation

The Cache when Vivado runs

Store the hardware details which include the using component, netlist after synthesizing and waveform results

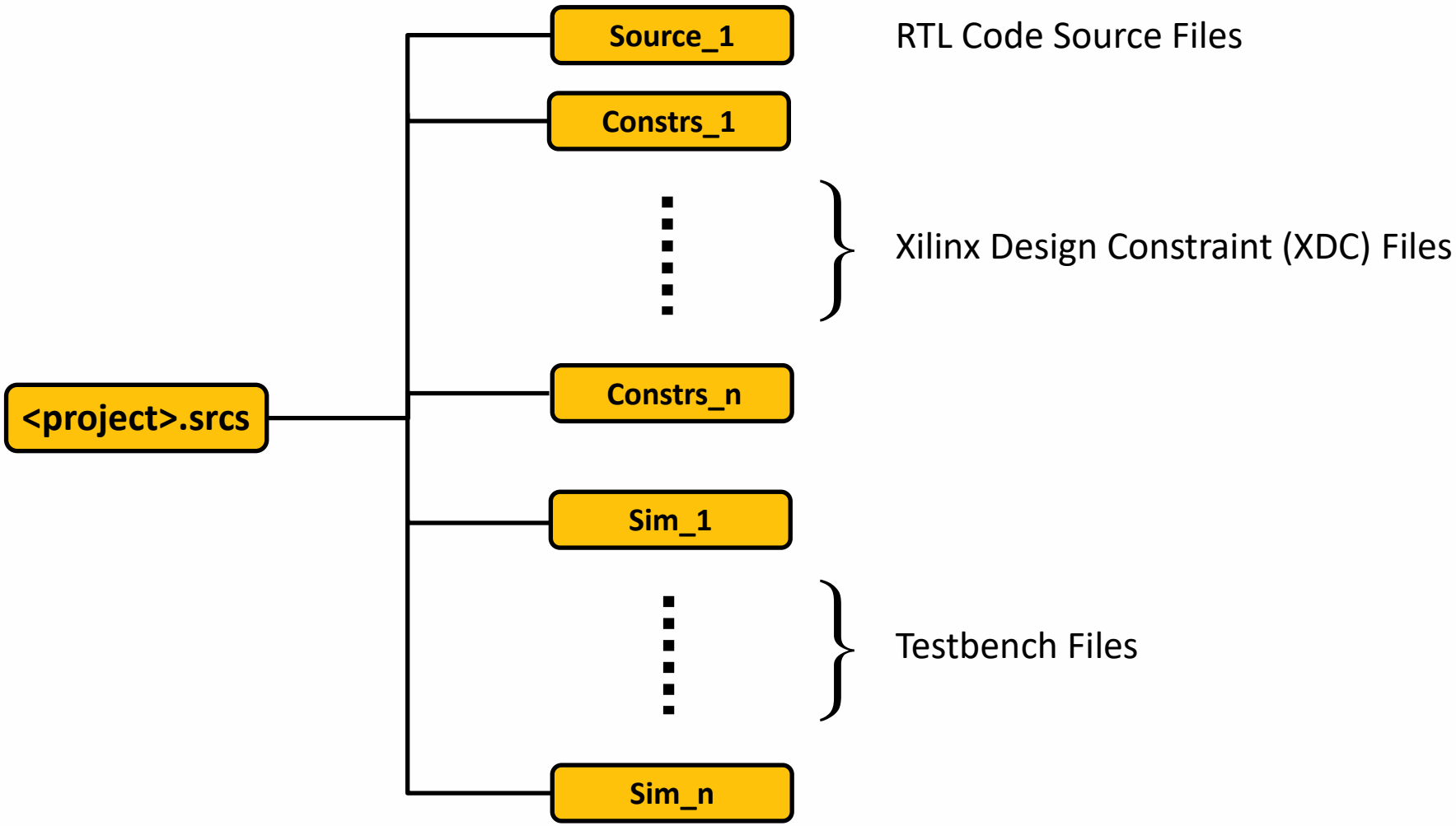
The files about IP of users

This directory only shows up when using CPU. Board Support Package (BSP) which is generated by hardware, software project, which is newly created are in this directory

Temporary directory of custom IP core. When finishing the IP core design, it will clean itself automatically

Integrated Design Environment (IDE)

- Project Filesets



Behavioral Simulation

XILINX.

Behavioral Simulation

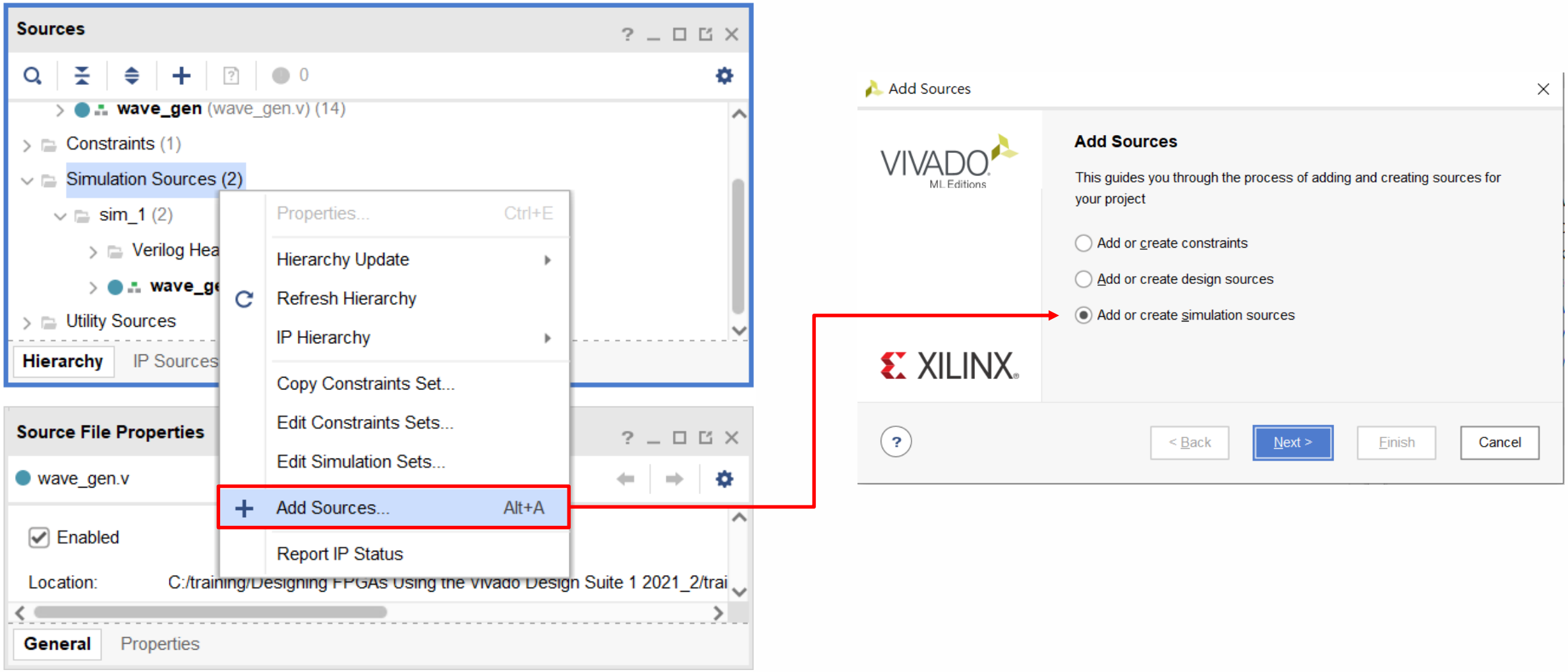
- Testbench

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

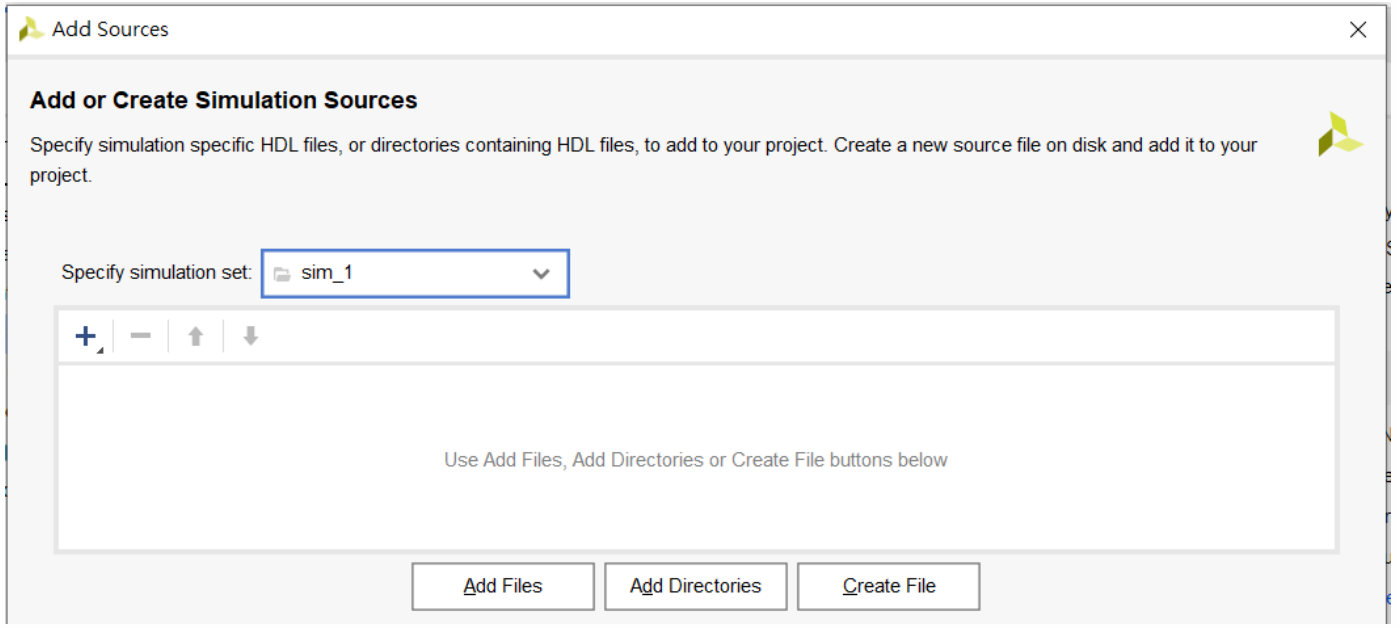
- Add Simulation Sources



Behavioral Simulation

- Add Simulation Sources

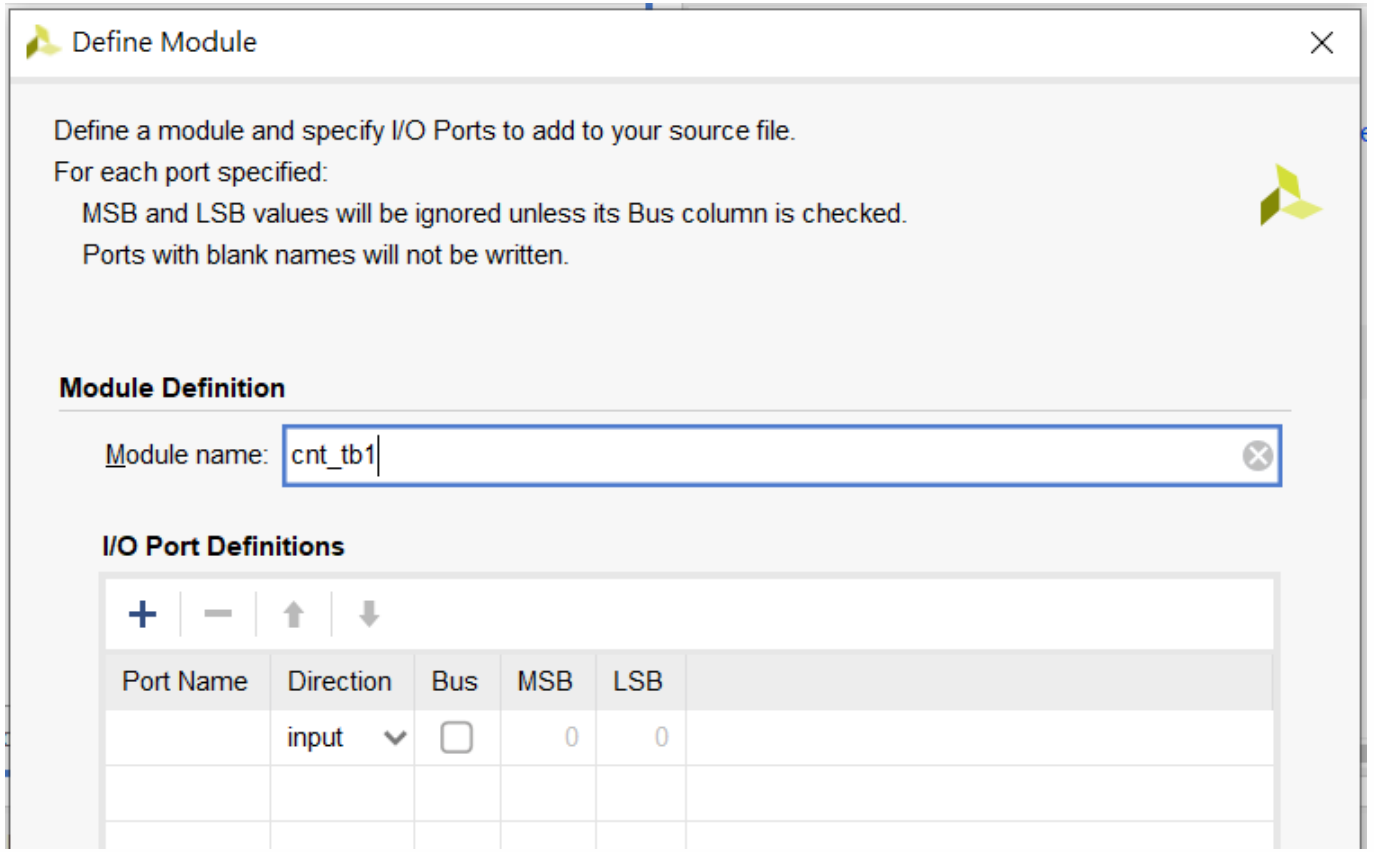
After selecting “Add or create simulation sources”, you can choose add an existing simulation source files or create a new one



Behavioral Simulation

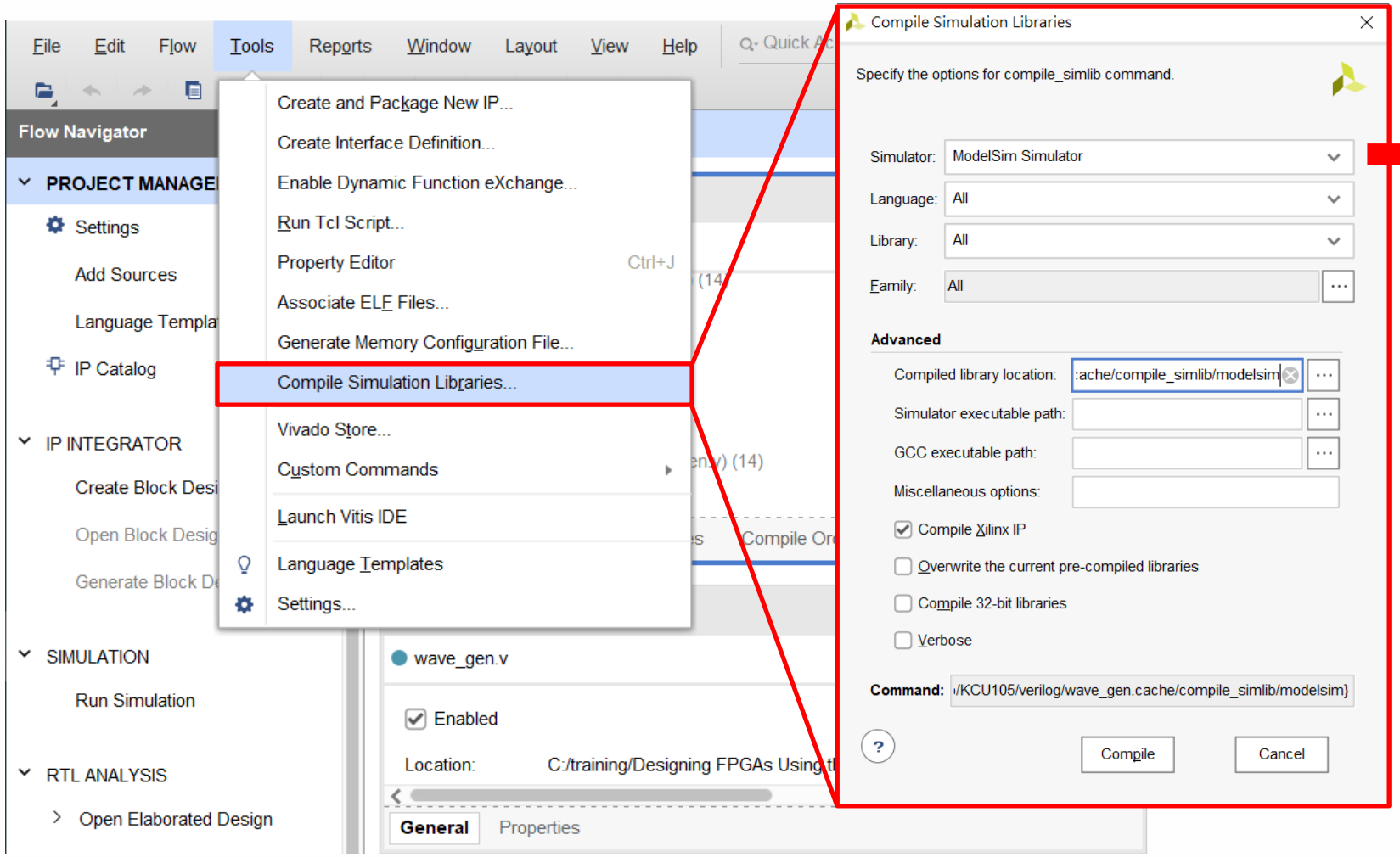
- Add Simulation Sources

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

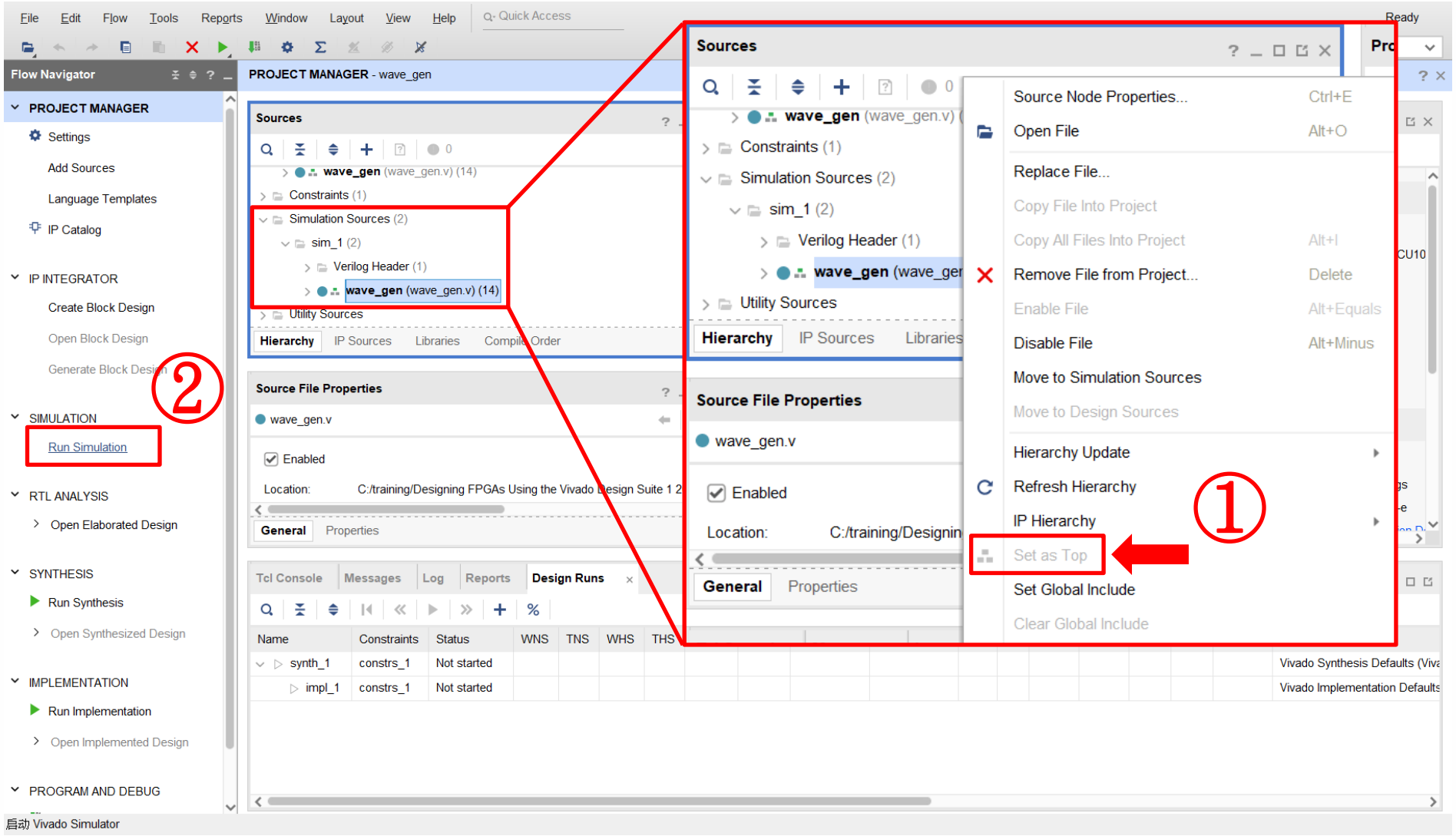
- 3-rd Party Simulation Libraries
  - > Some IPs and Devices have Simulation Libraries that have already been built



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

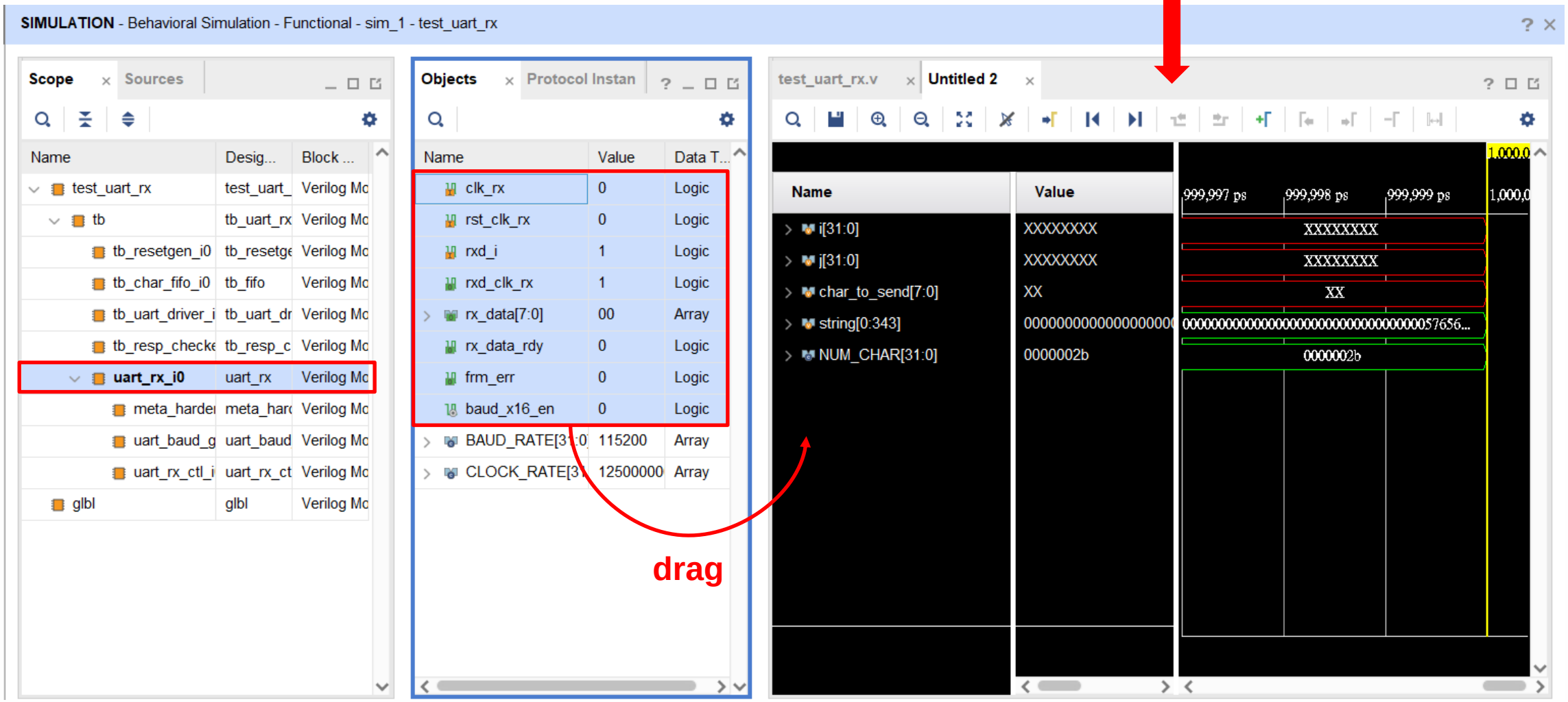
Behavioral Simulation

- Run Simulation



Behavioral Simulation

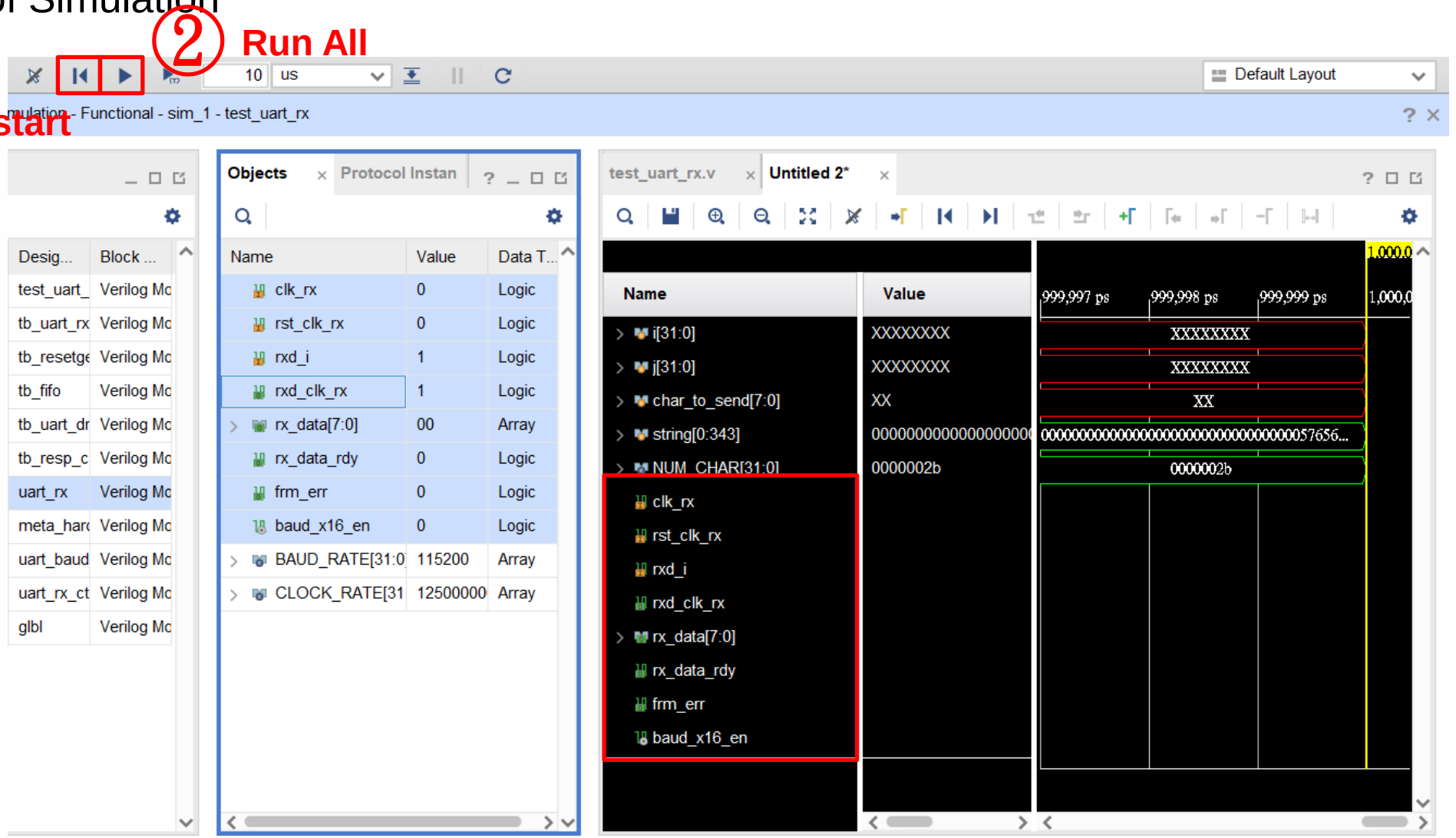
- Results of Simulation



XILINX.

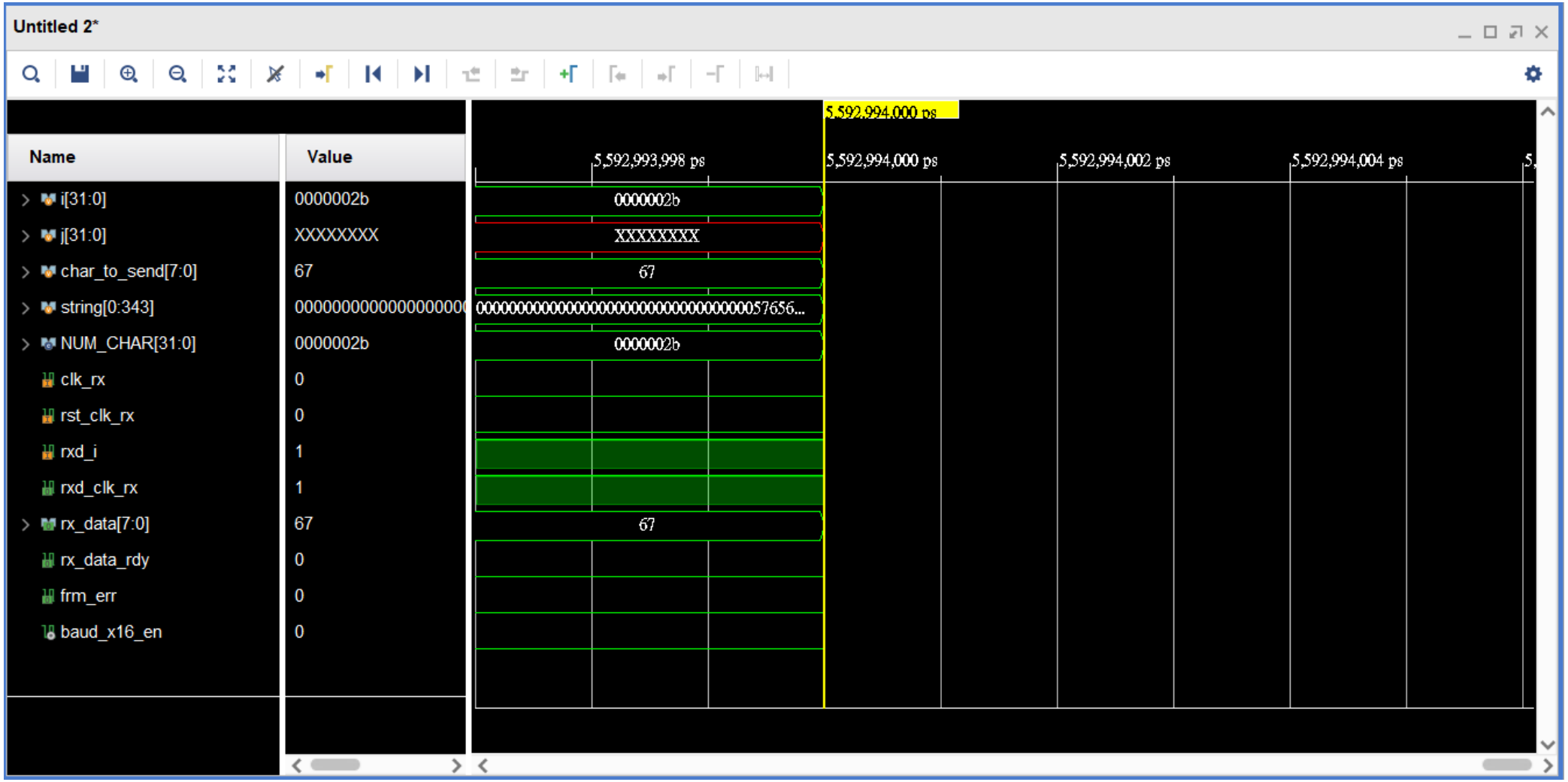
Behavioral Simulation

- Results of Simulation



Behavioral Simulation

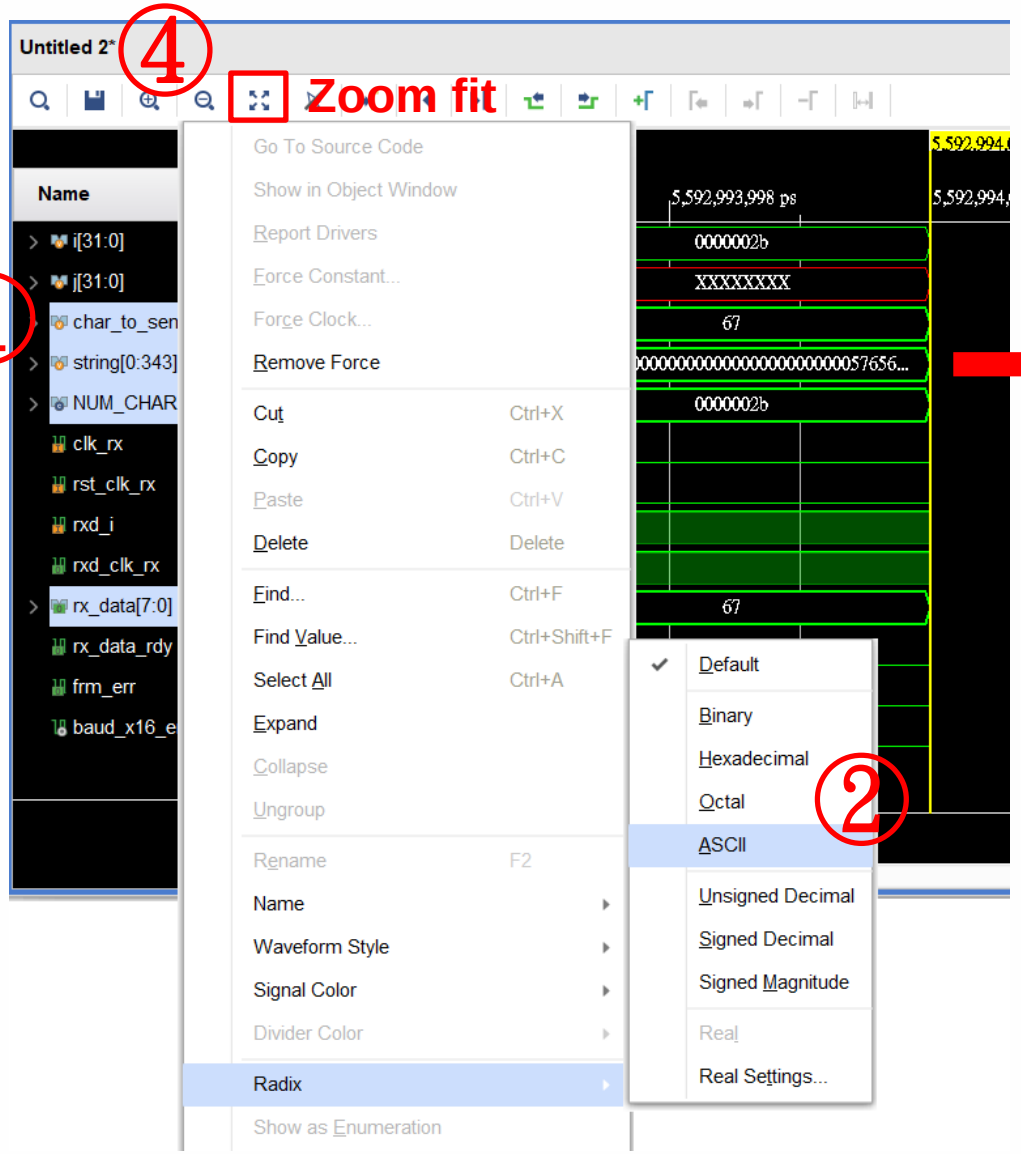
- Results of Simulation

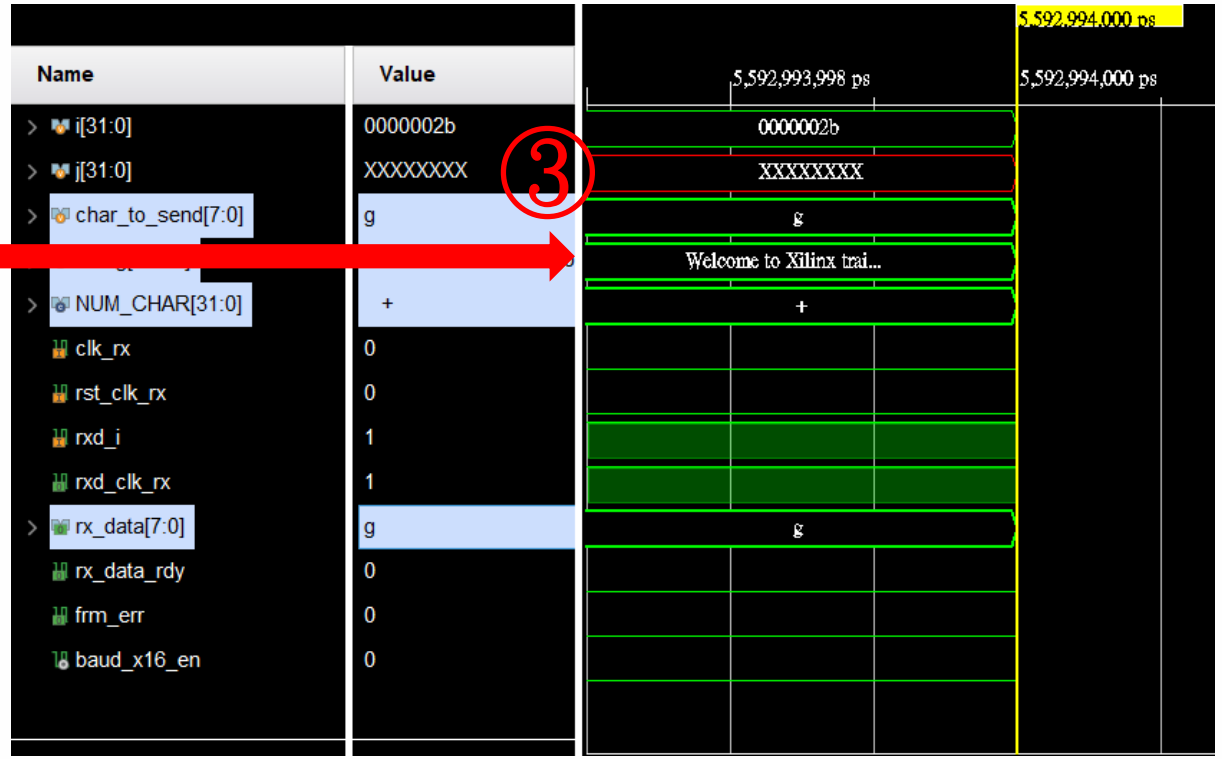


XILINX.

Behavioral Simulation

- Results of Simulation

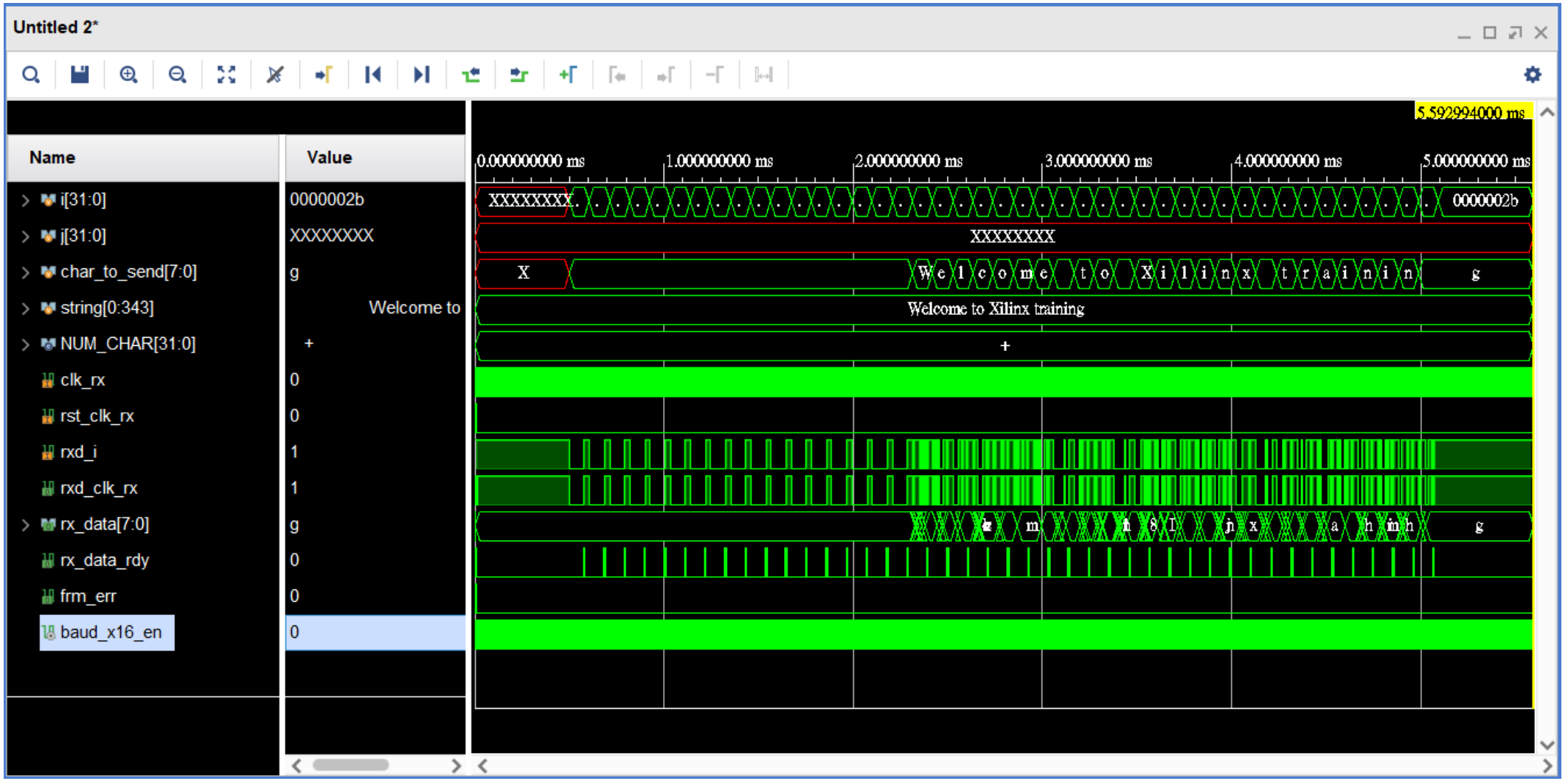




XILINX.

Behavioral Simulation

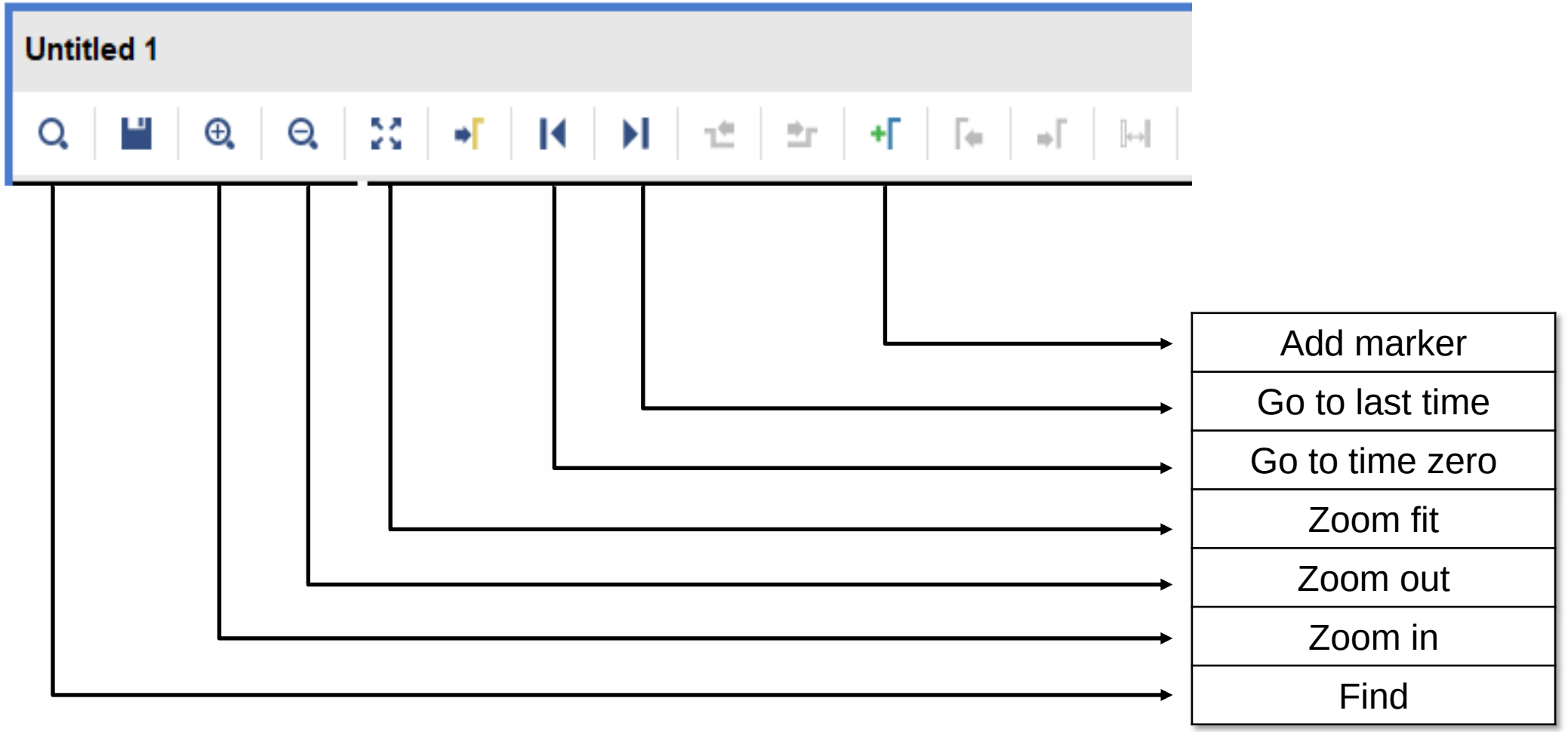
- Results of Simulation



XILINX.

Behavioral Simulation

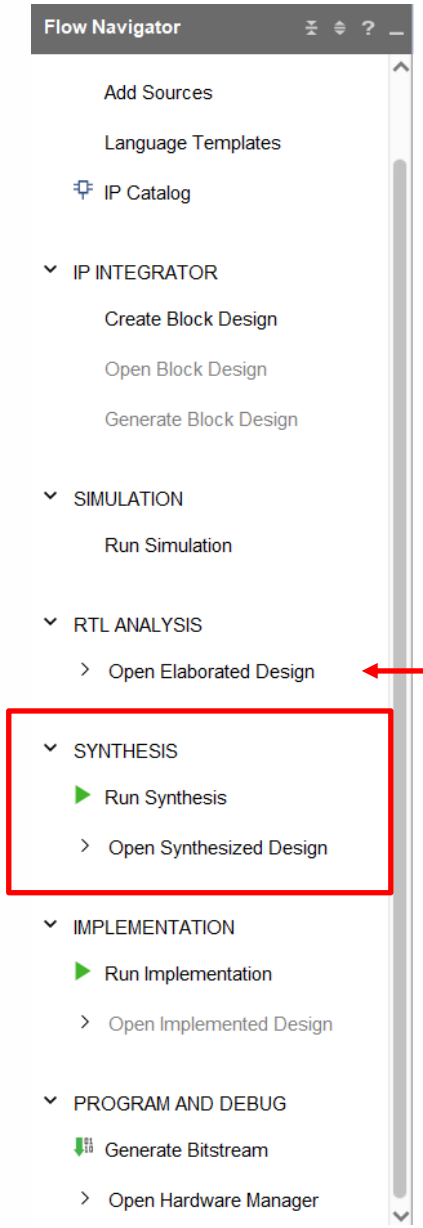
- Waveform toolbar



Vivado Synthesis

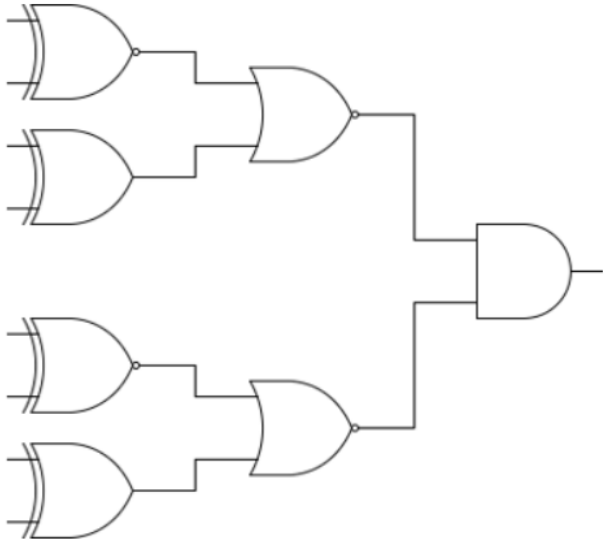
XILINX.

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



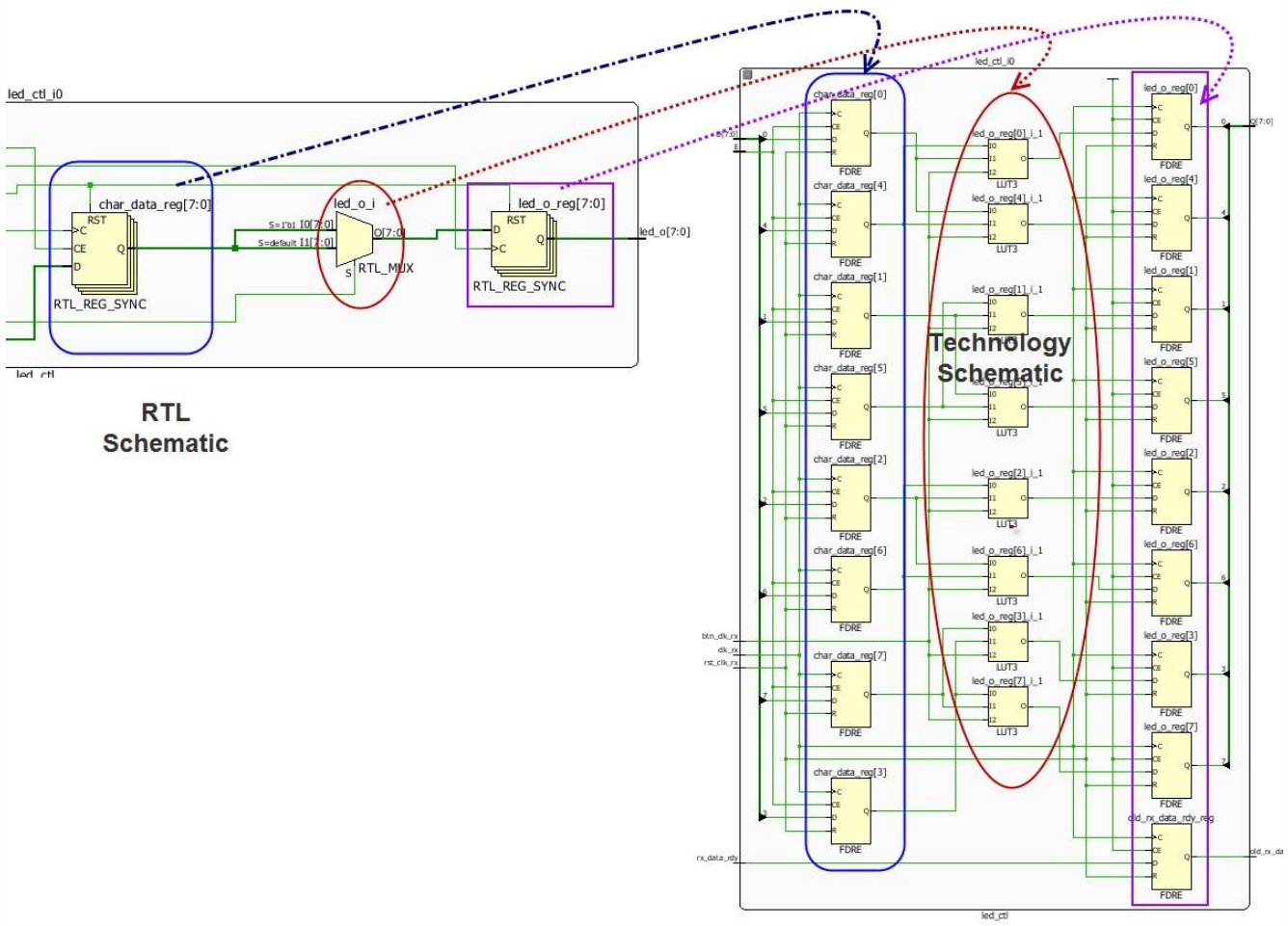
XILINX.

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

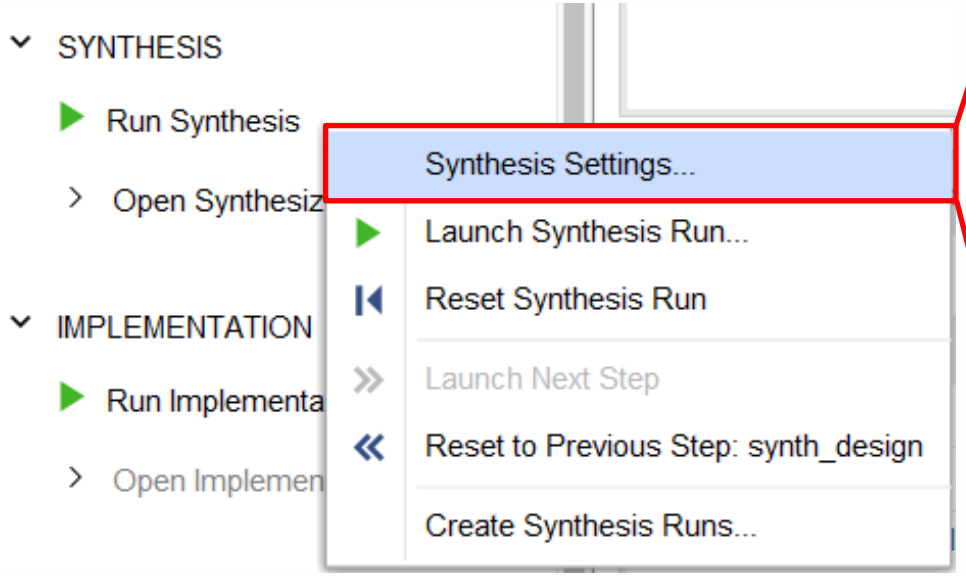
> Verilog, VHDL, mixed HDL, and SystemVerilog support

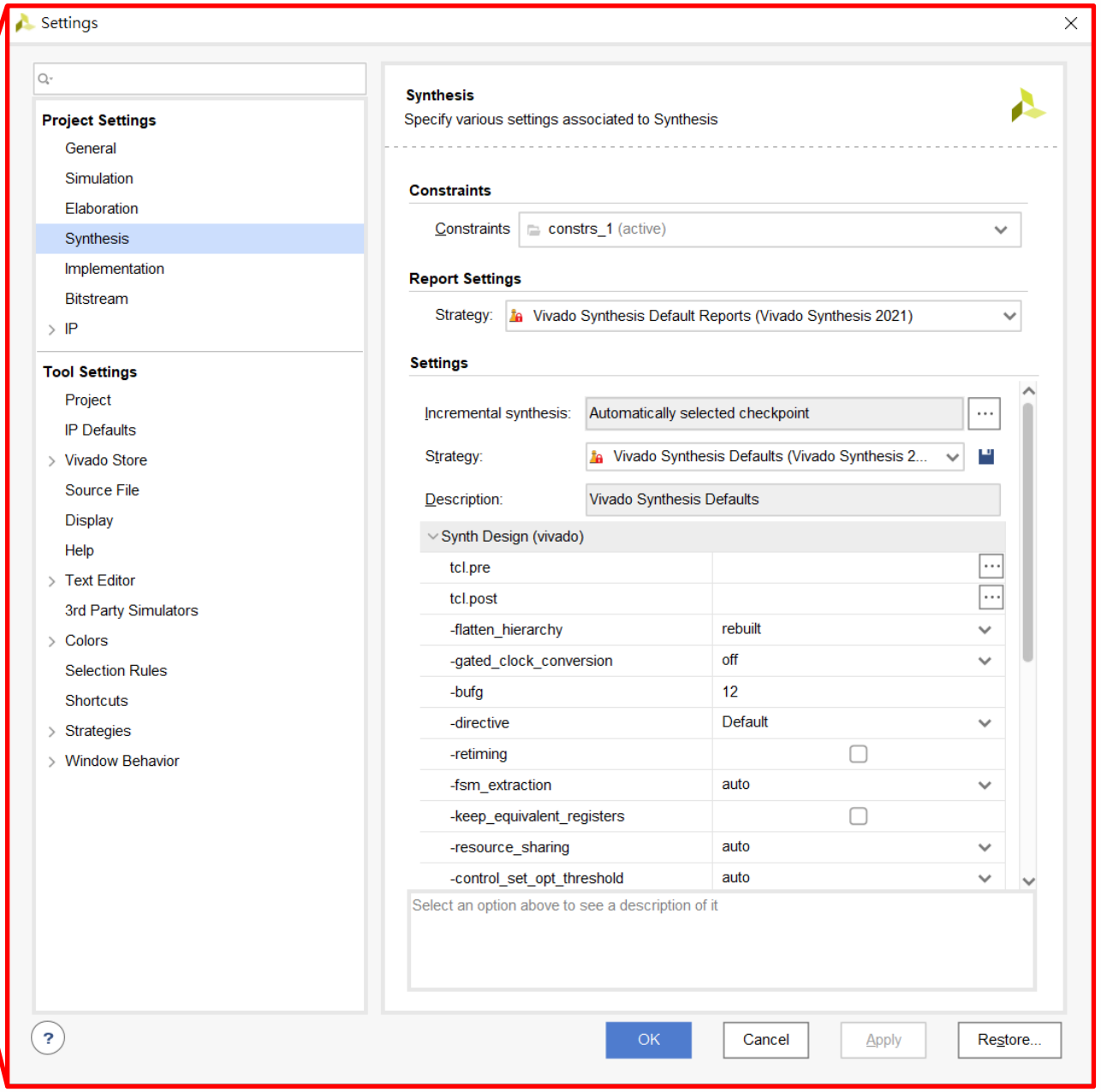


XILINX.

Vivado Synthesis

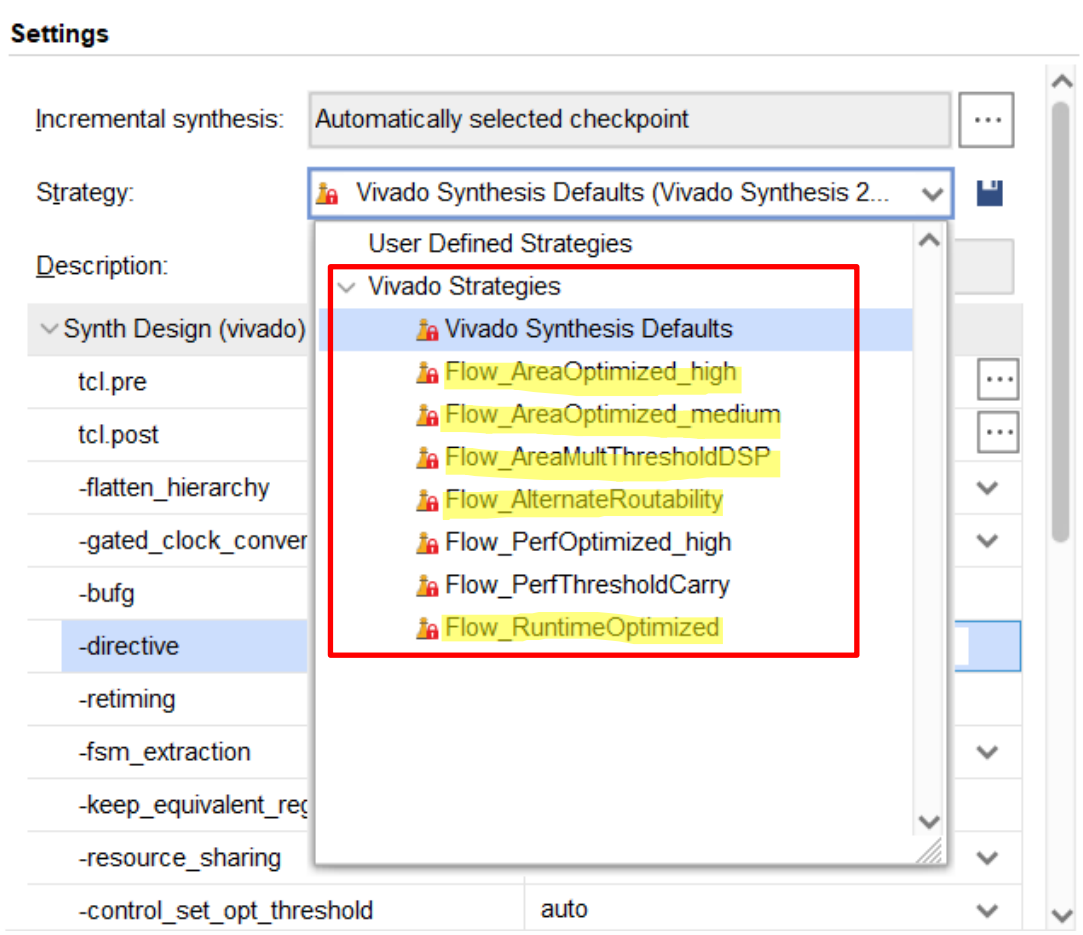
- Synthesis Settings





Vivado Synthesis

- Synthesis Strategy



Flow_AreaOptimized_high
Flow_AreaOptimized_medium

Global Area Optimized

Flow_AreaMultThresholdDSP

Using more DSP resouces

Flow_AlternateRoutability

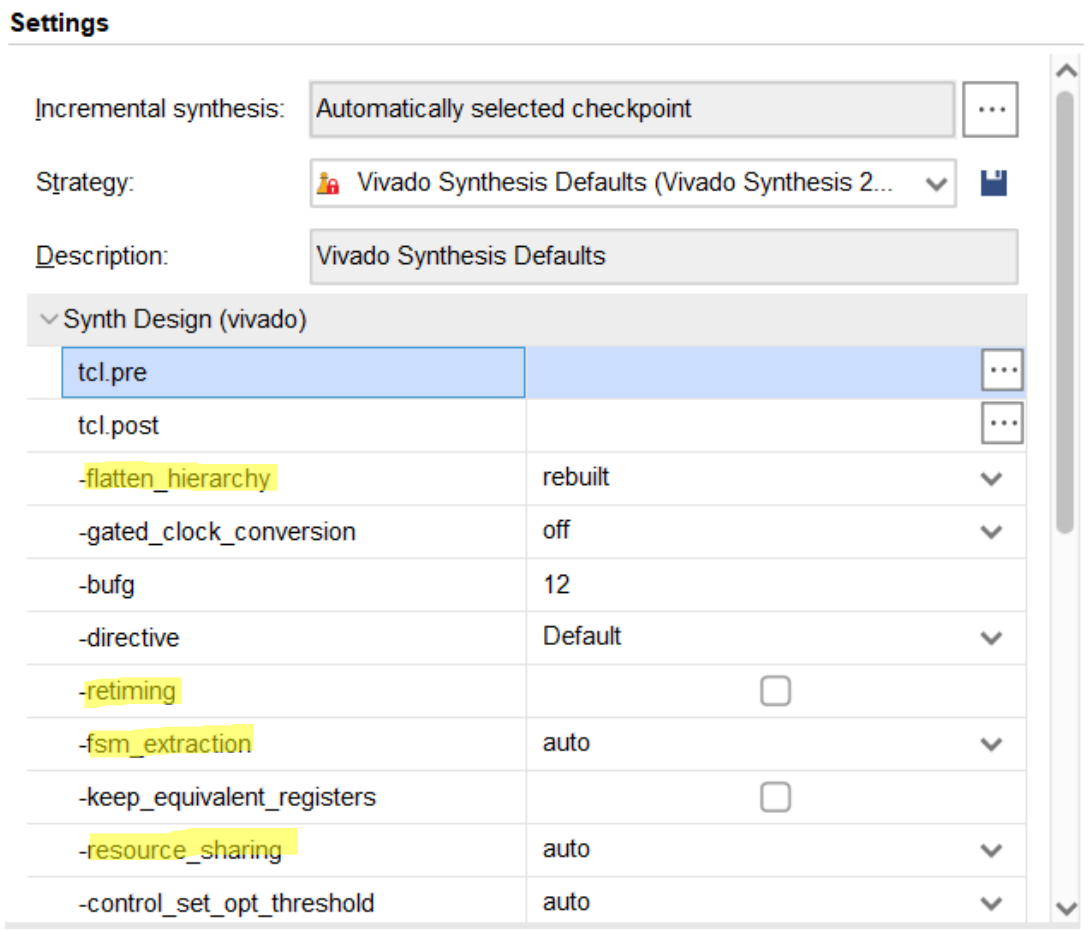
Increase Route ability and decrease the using of MUXF and CARRY

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

Retiming

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

Fsm_extraction

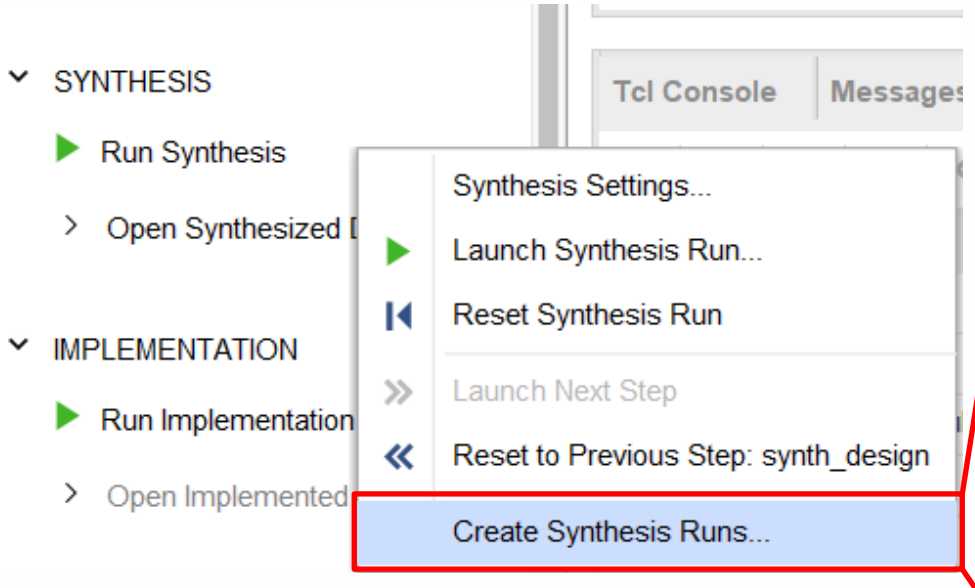
Vivado will infer the encoding way of FSM automatically

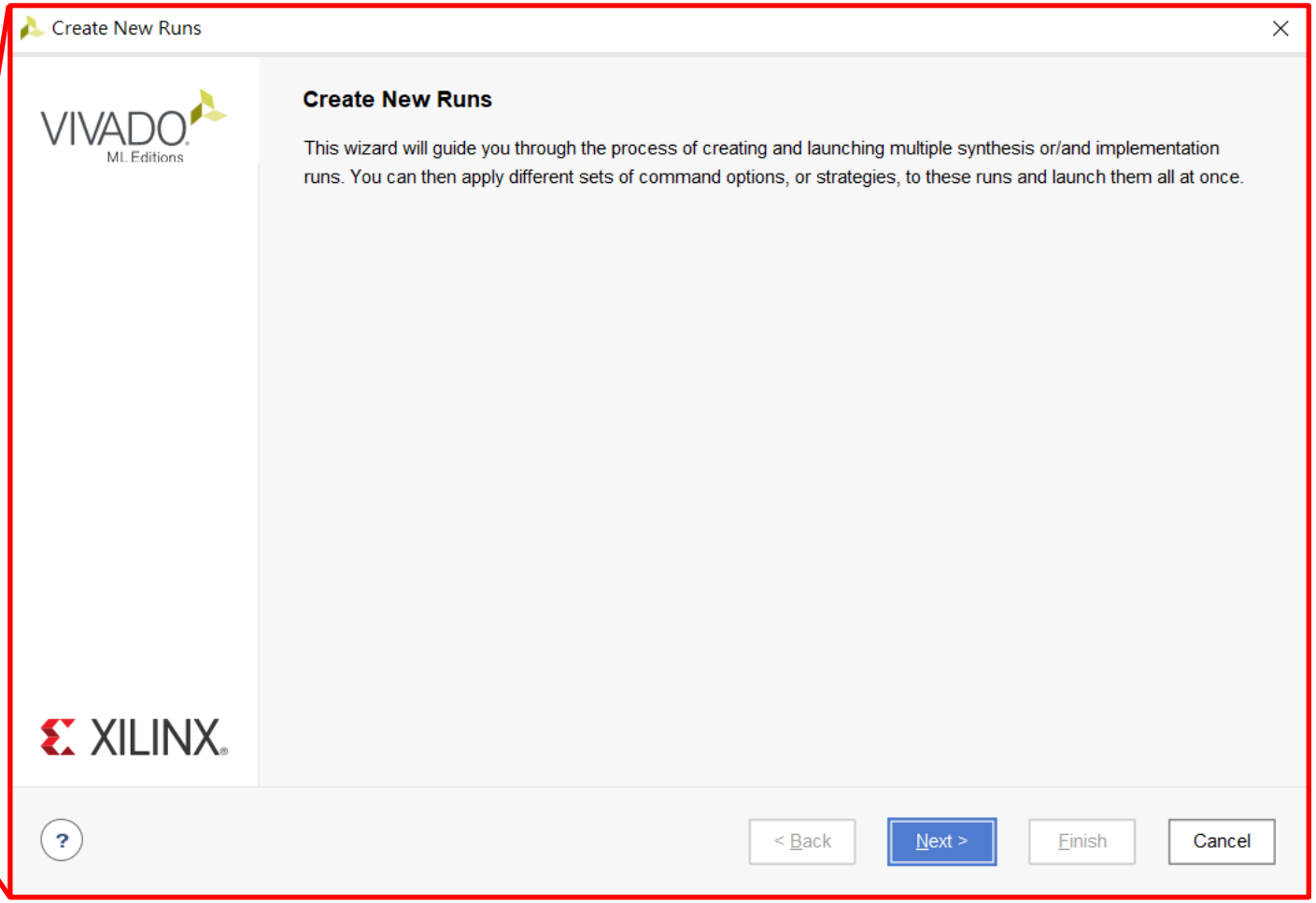
Resource_sharing

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

Vivado Synthesis

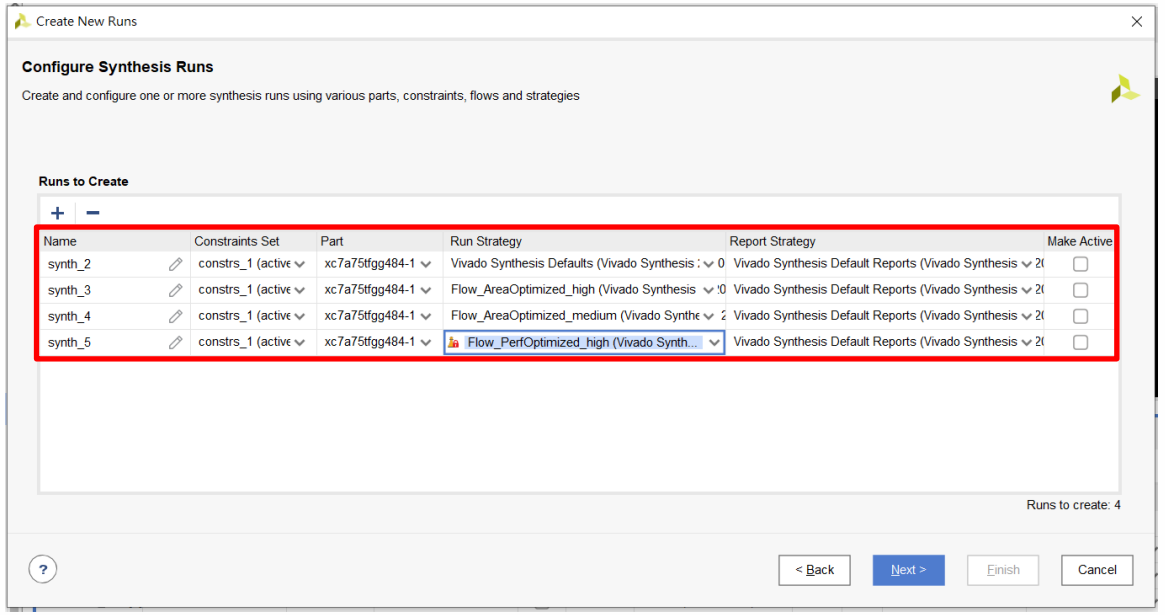
- Create Synthesis Runs



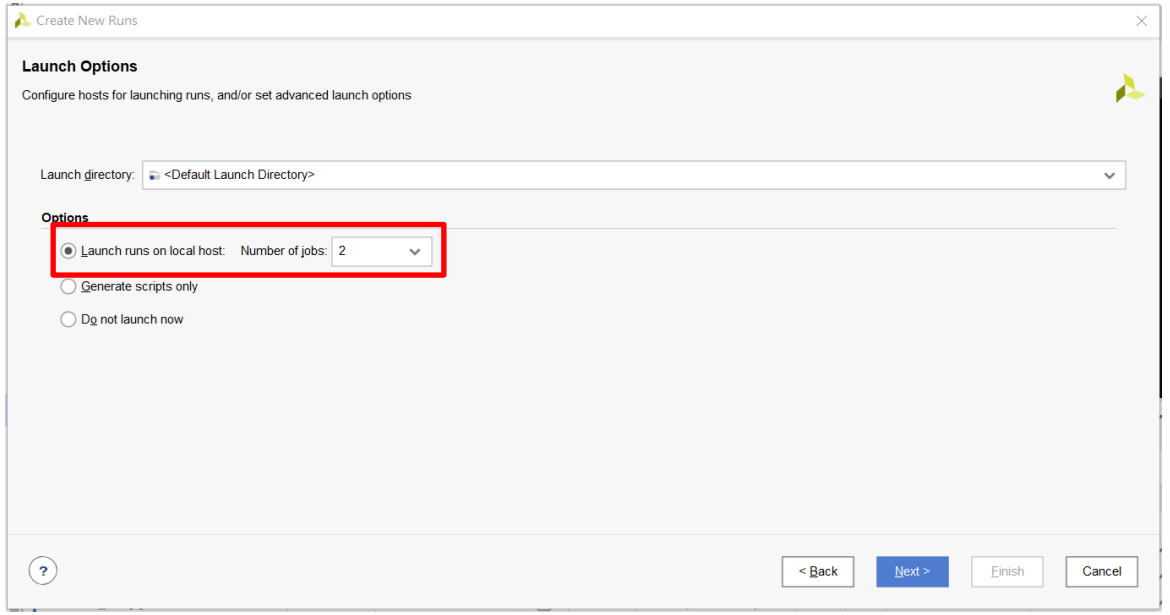


Vivado Synthesis

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

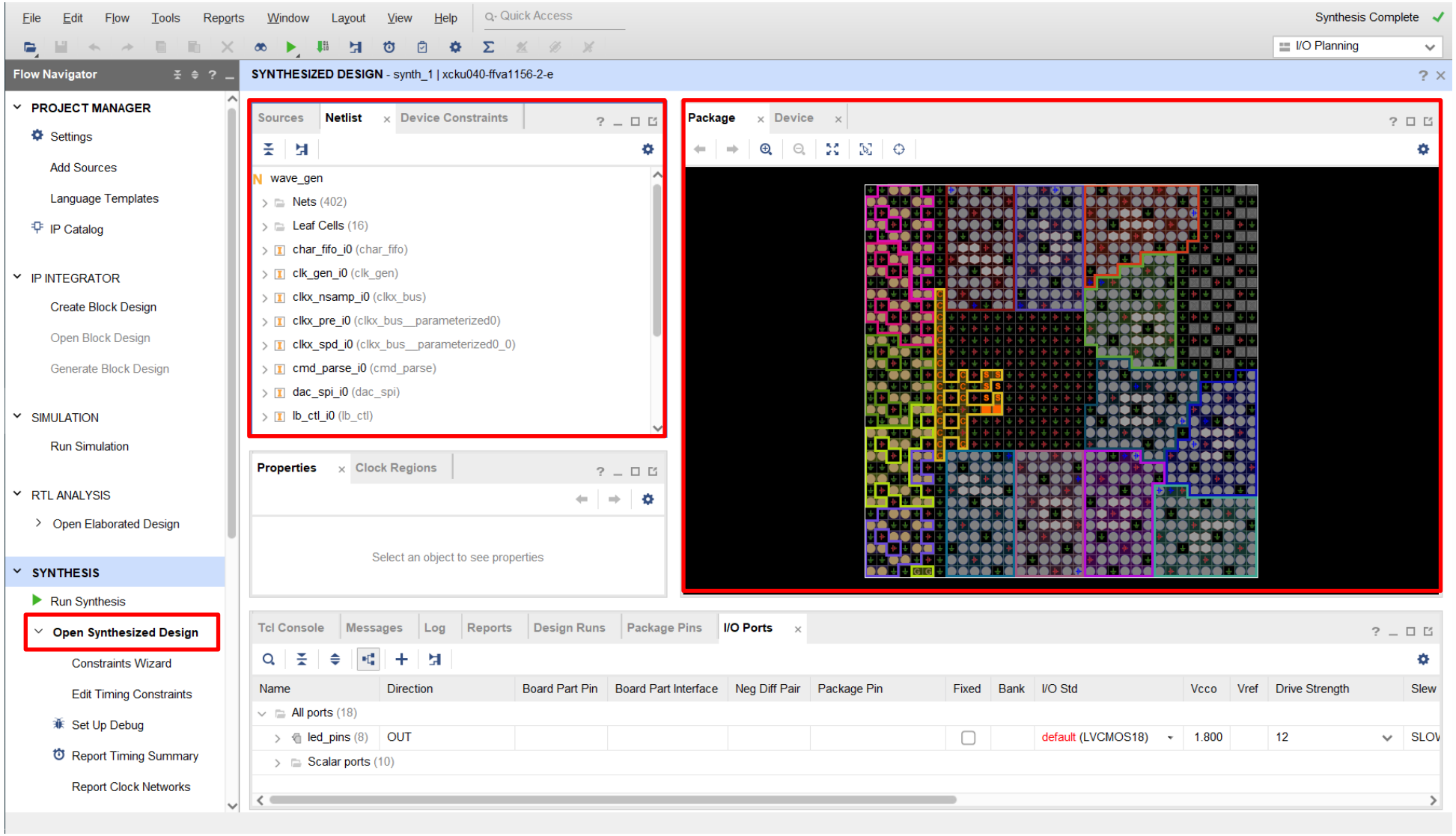


- Next step offers changing launch directory and changing number of jobs



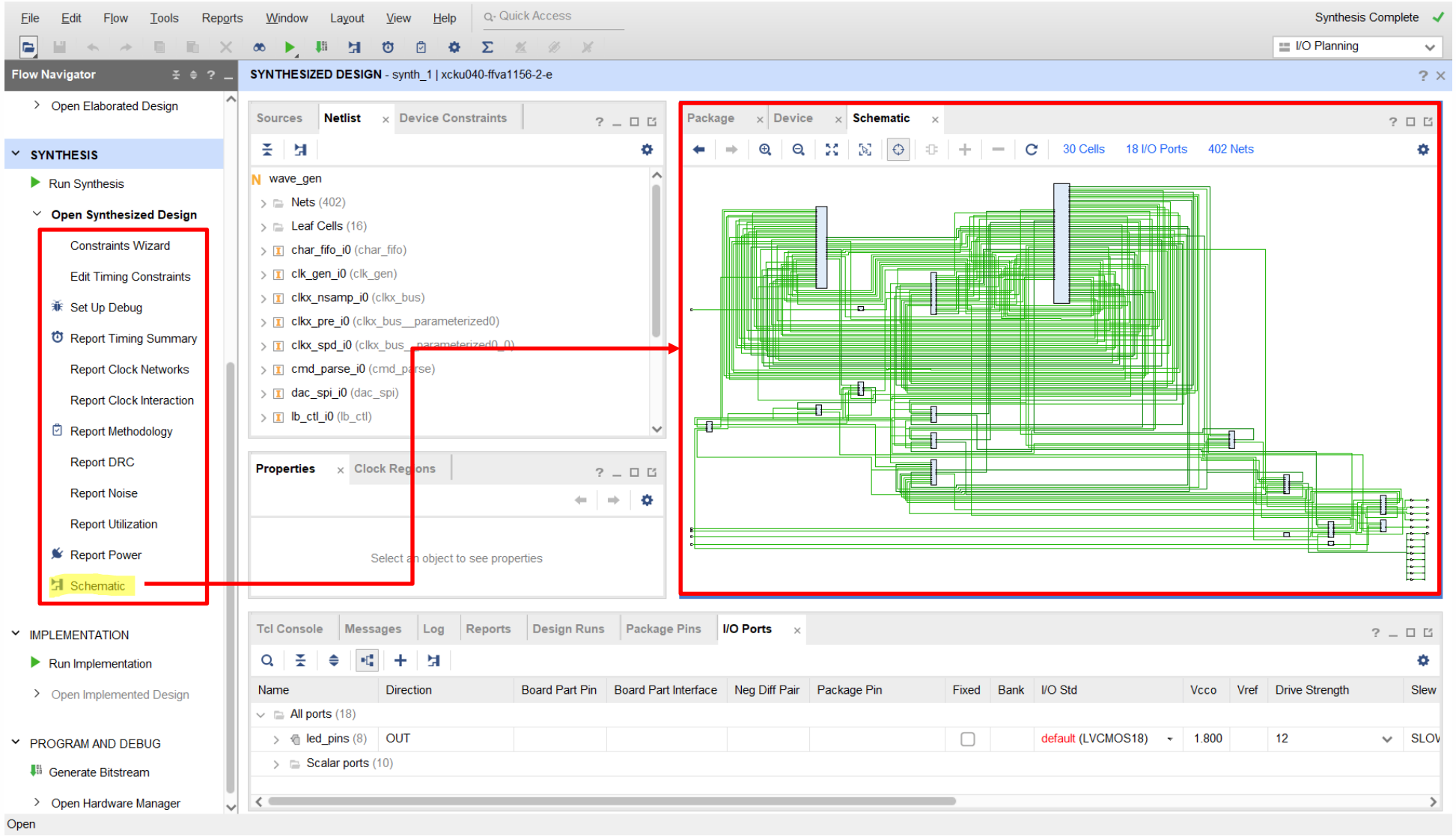
Vivado Synthesis

- After Synthesis



Vivado Synthesis

- After Synthesis

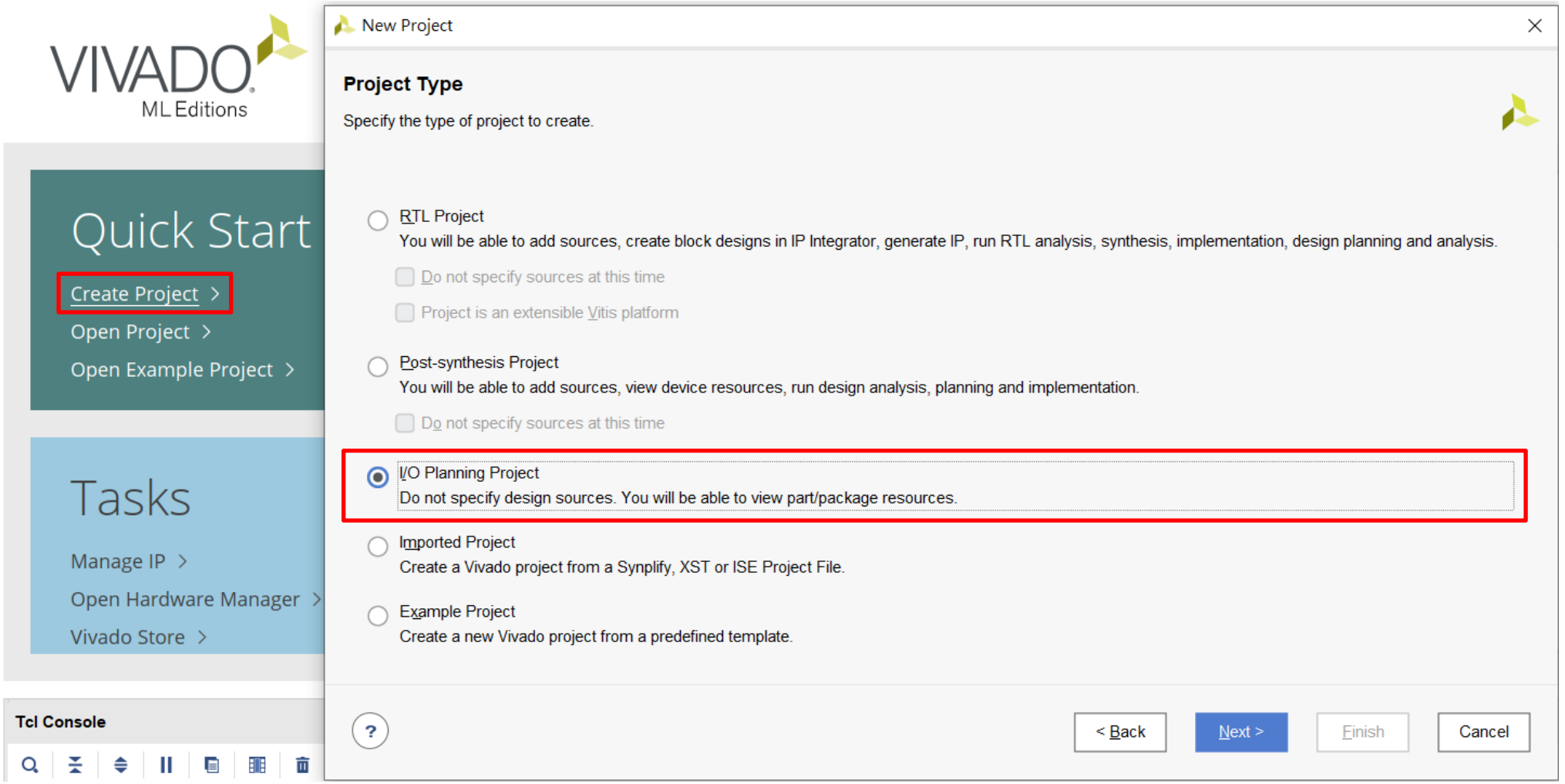


Vivado Design Suite I/O Pin Planning

XILINX.

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 on 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

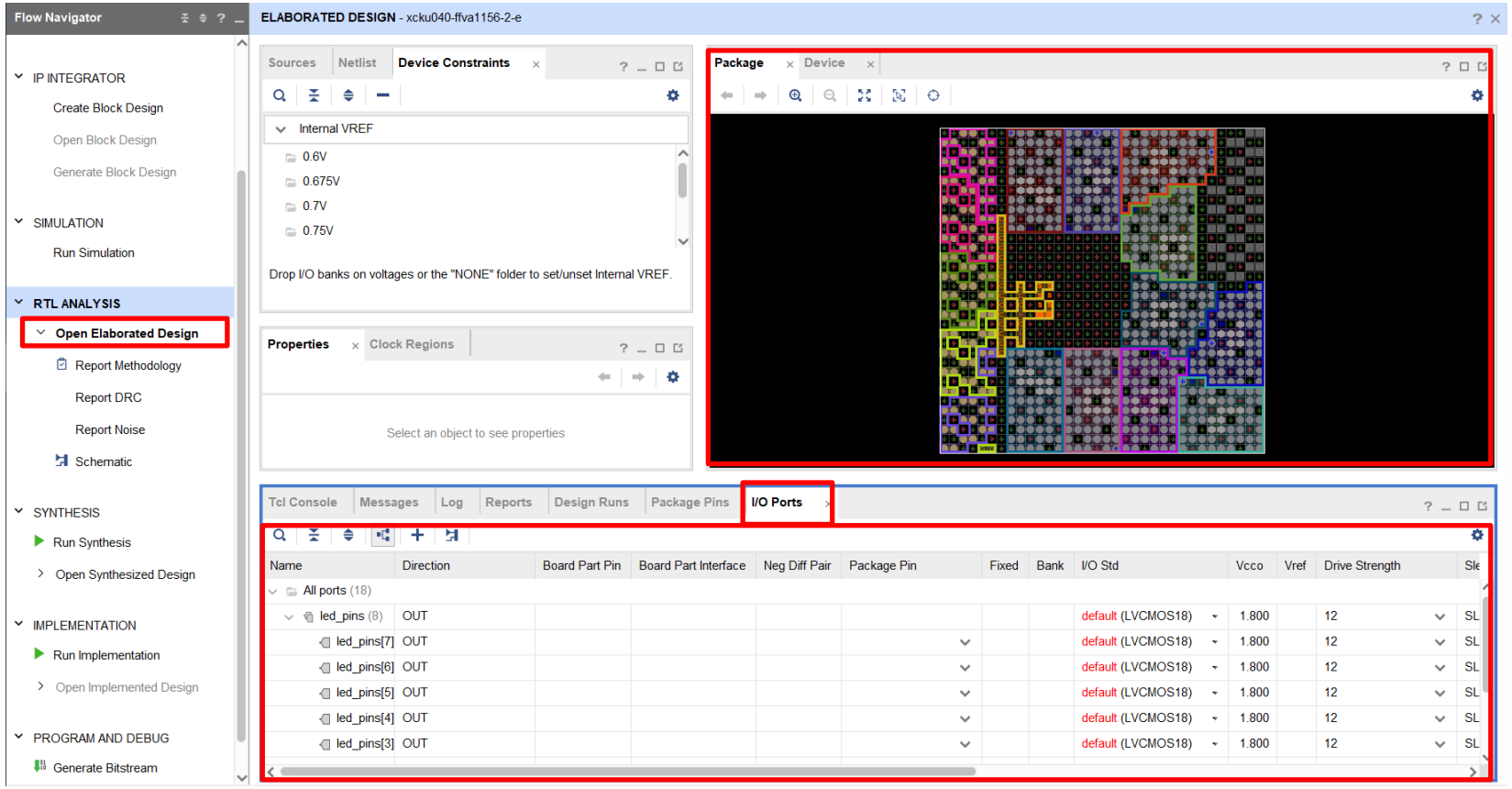
◆ It is recommended that you have RTL associated Error checking is better.

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



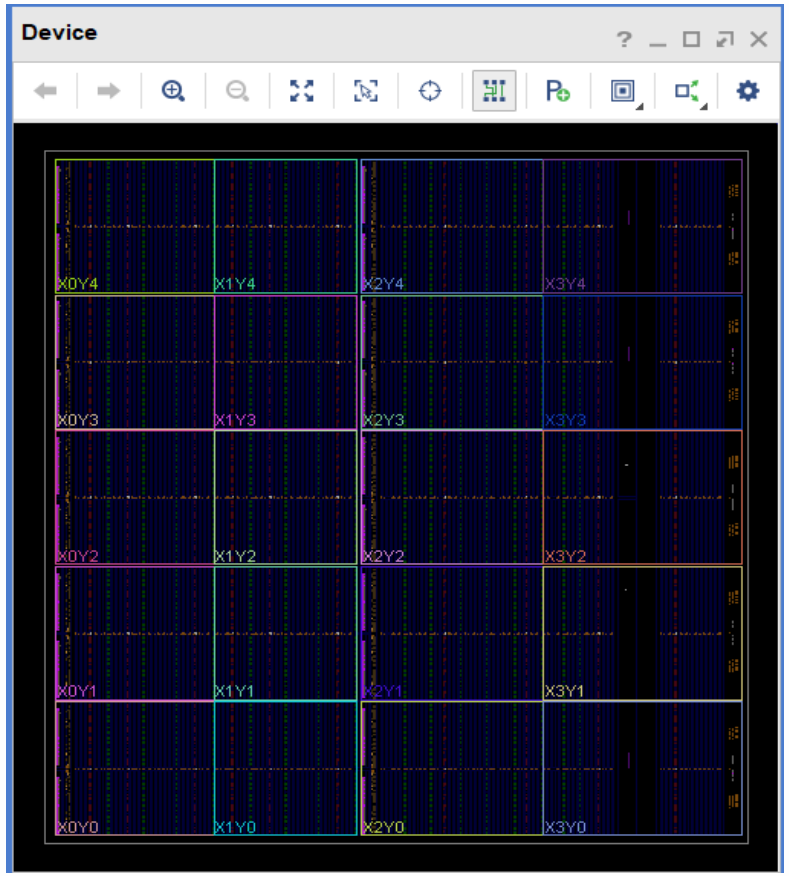
Vivado I/O Pin Planning

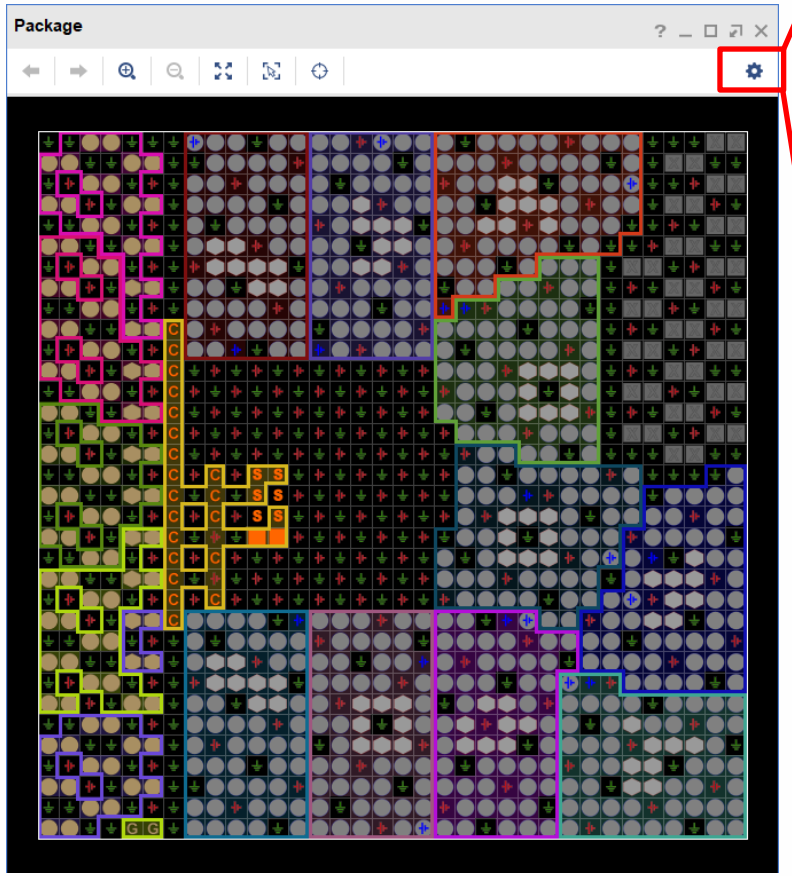
- I/O Pin Planning can be executed at three different steps
  3. On Synthesized & Implemented Design
     - Already has Constraint files or the result of Place & Route
     - Can execute auto layout of I/O planning
     - I/O planning need to be validated at Implementation

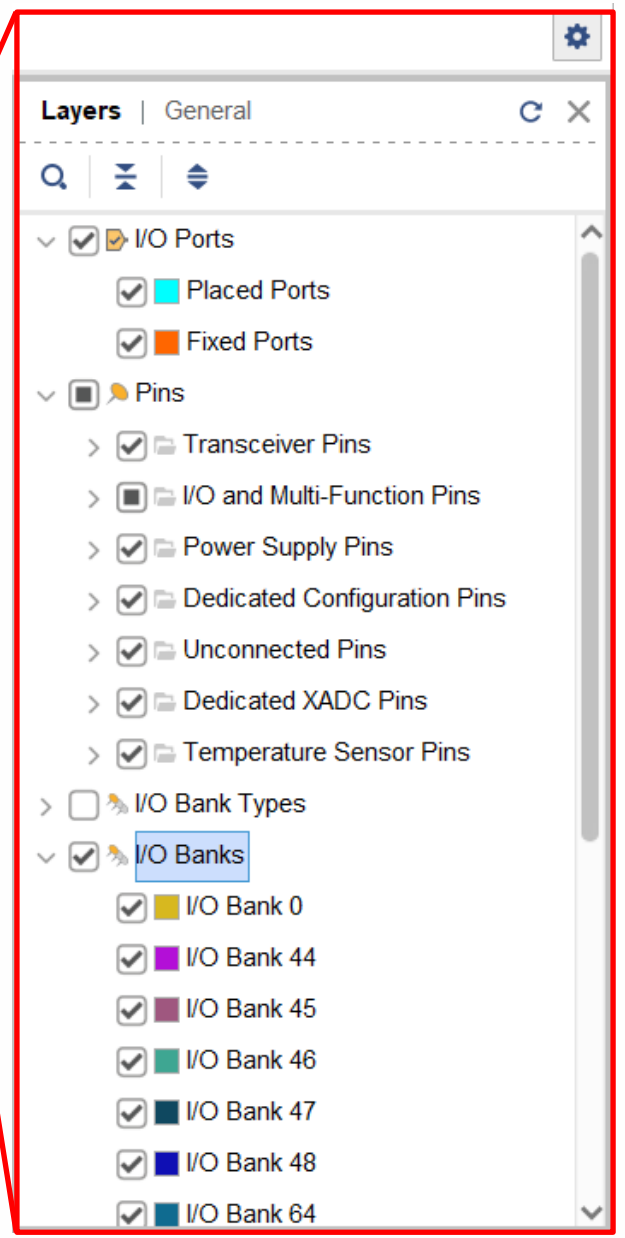
- It is recommended that before doing the auto layout of I/O planning, you can layout the timing-critical path manually first.

Vivado I/O Pin Planning Layout View

- Device & Package View

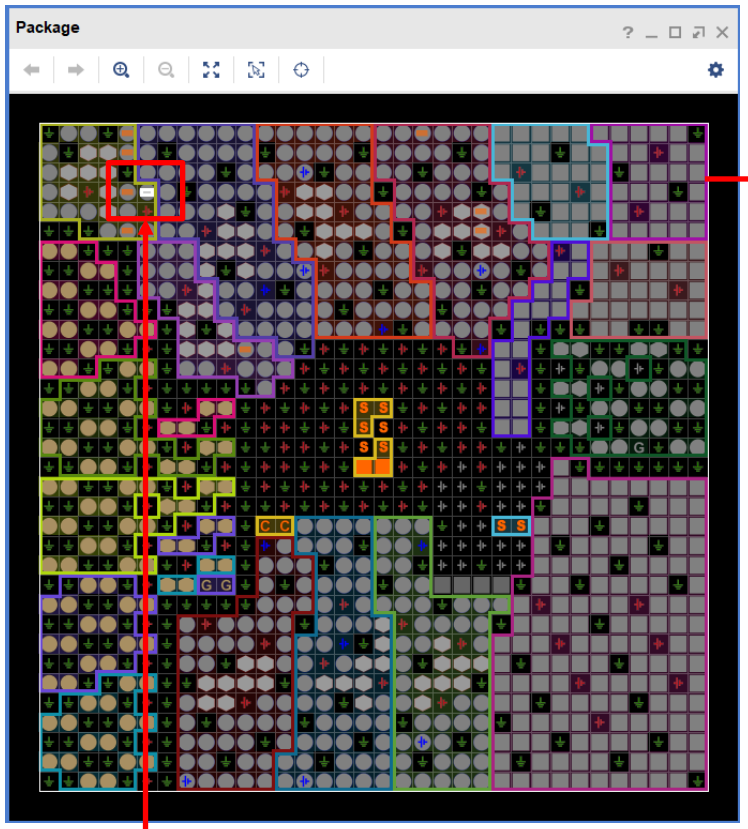






Vivado I/O Pin Planning Layout View

- Device & Package View



Different color blocks mean different I/O Banks

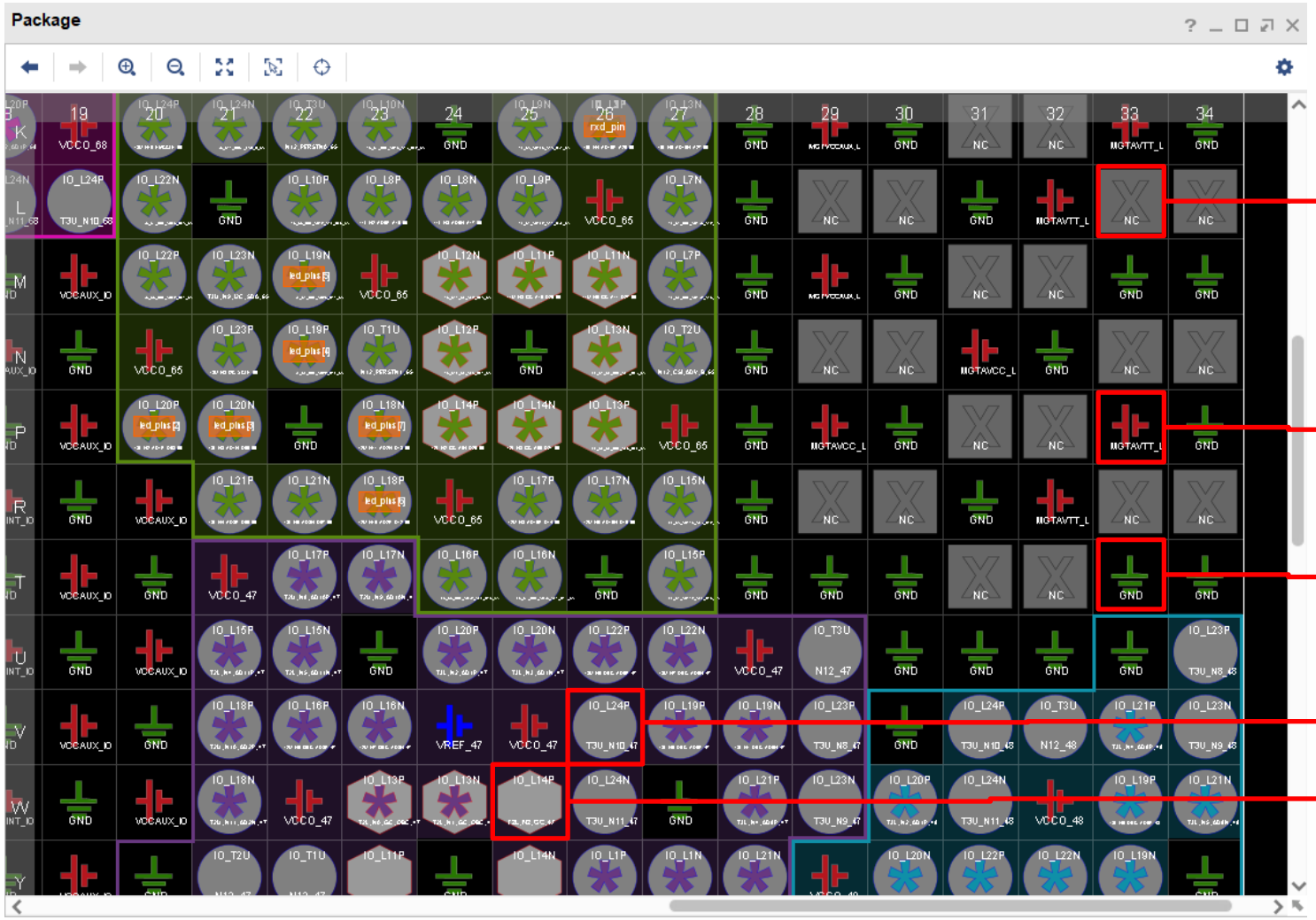
Tcl Console | Messages | Log | Reports | Design Runs | Package Pins | I/O Ports

Name	Direction	Board Part Pin	Board Part Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type
All ports (9)												
led_pins (4)	OUT					☑	88	LVCMOS33*	3.300		12	SLOW
led_pins[3]	OUT				B5	☑	88	LVCMOS33*	3.300		12	SLOW
led_pins[2]	OUT				A5	☑	88	LVCMOS33*	3.300		12	SLOW
led_pins[1]	OUT				D6	☑	88	LVCMOS33*	3.300		12	SLOW
led_pins[0]	OUT				D5	☑	88	LVCMOS33*	3.300		12	SLOW
Scalar ports (5)												

Cross-probing Check

Vivado I/O Pin Planning Layout View

- Device & Package View



No Connection

VCC

GND

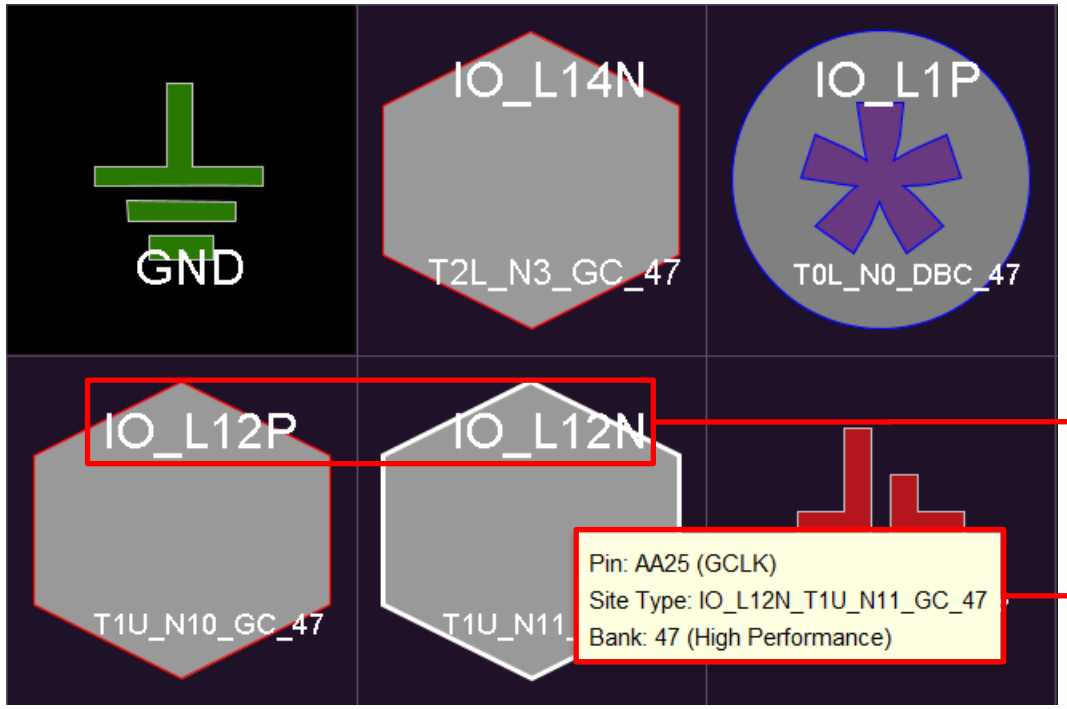
Normal I/O pins

Clock-capable pins

- For the input/output clock of PCB

Vivado I/O Pin Planning Layout View

- Device & Package View



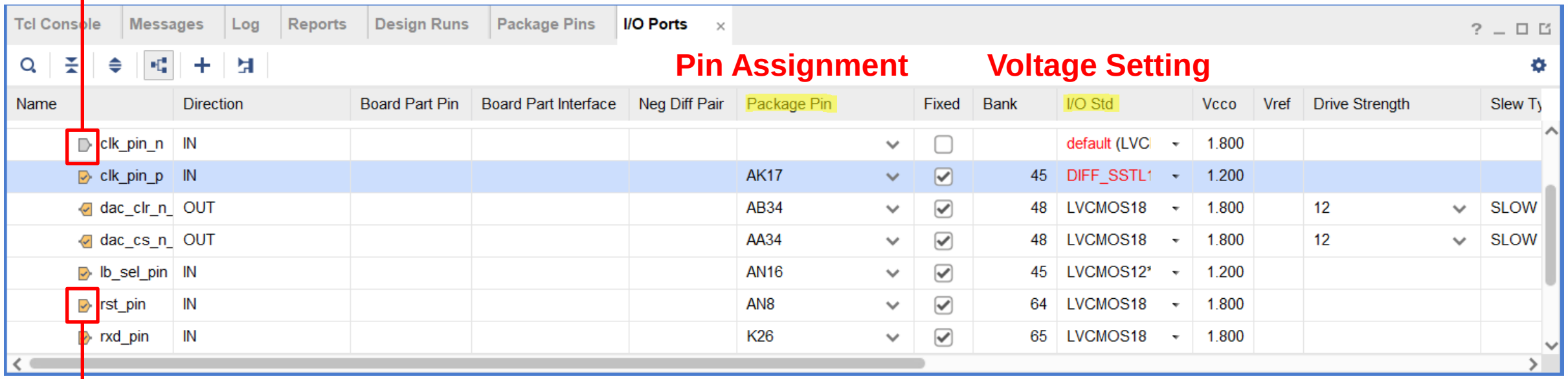
Differential Pair

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

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

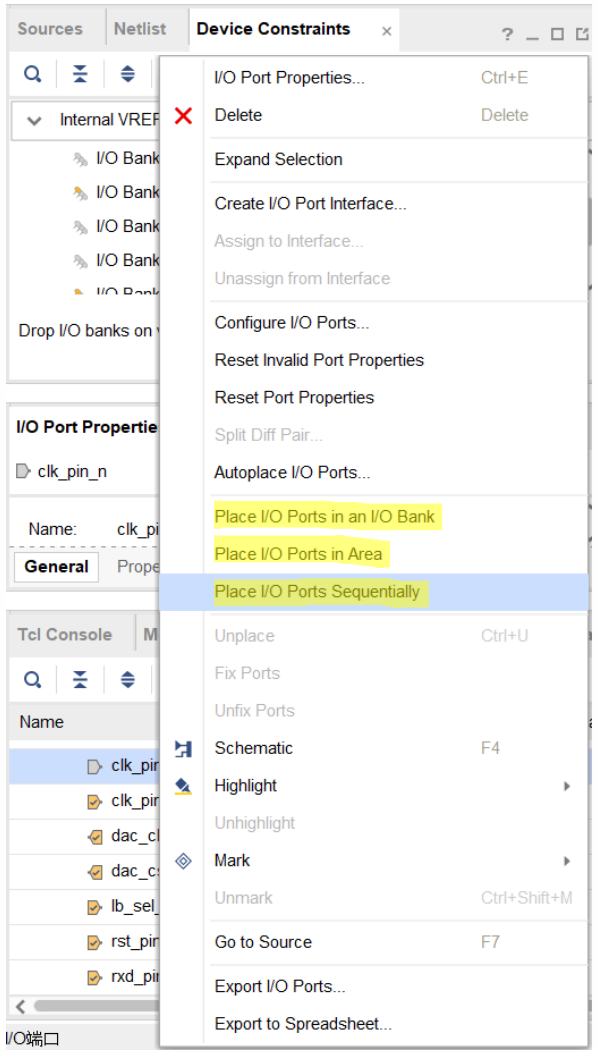
   Select multiple I/O and place them in one bank in Device View, Vivado will keep them all in the same bank as much as possible

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

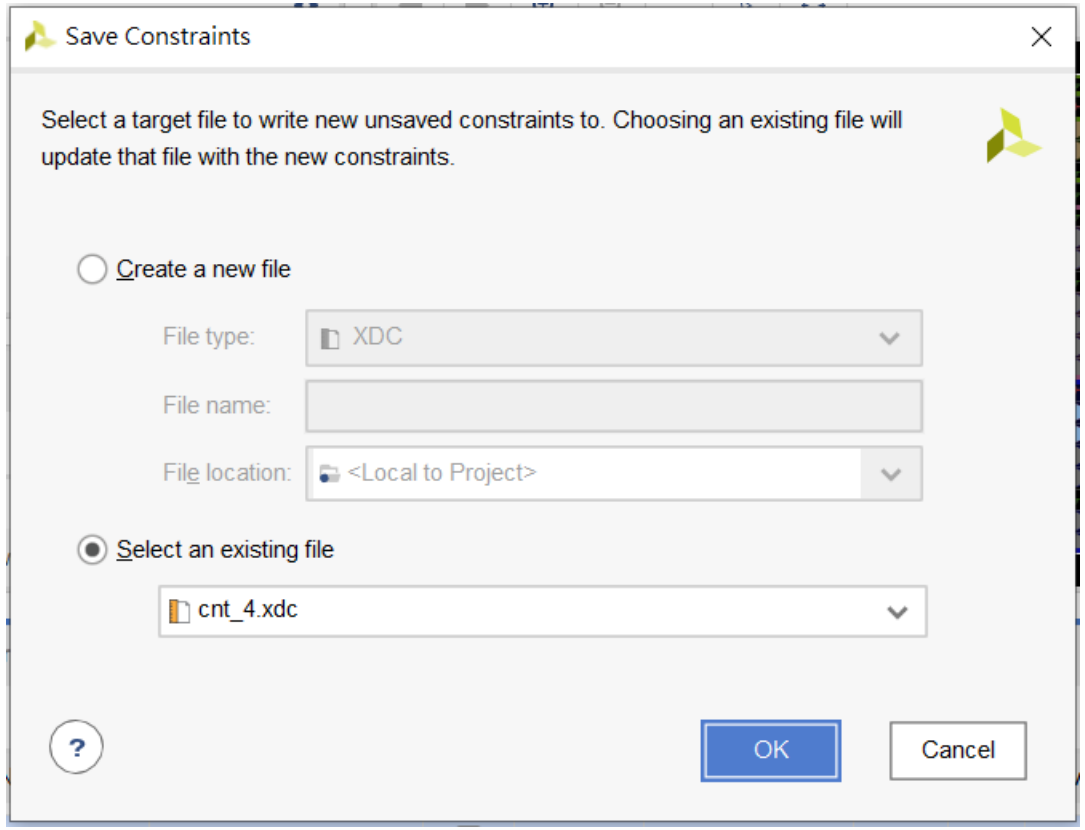
3. Sequentially

   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

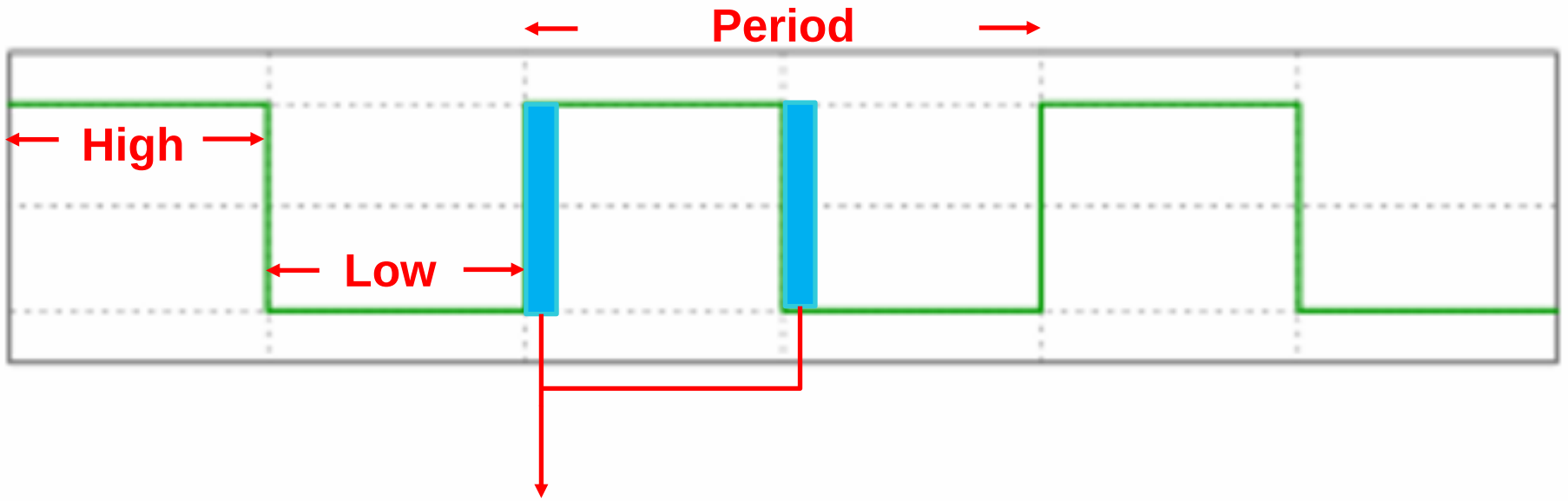
Define I/Os early in the design process	Interactively analyze and assign I/Os	Robust DRCs	SSN analysis
• Import RTL, CSV, XDC	• DRC legal group drag and drop; assign clock logic	• Find problems prior to implementation	• Estimates switching noise levels caused by switching output ports

- Simultaneous Switch Noise (SSN) Analysis
  - Avoid Ground Bounce happening
  - High-speed circuits like Transceivers need SSN analysis very much

Vivado Clock Constraints

XILINX.

Clocks



Jitter the timing difference between ideal and actual waveform

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

Clocks

- Using TCL to control clocks
  1. Create

Start

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

ns

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
```

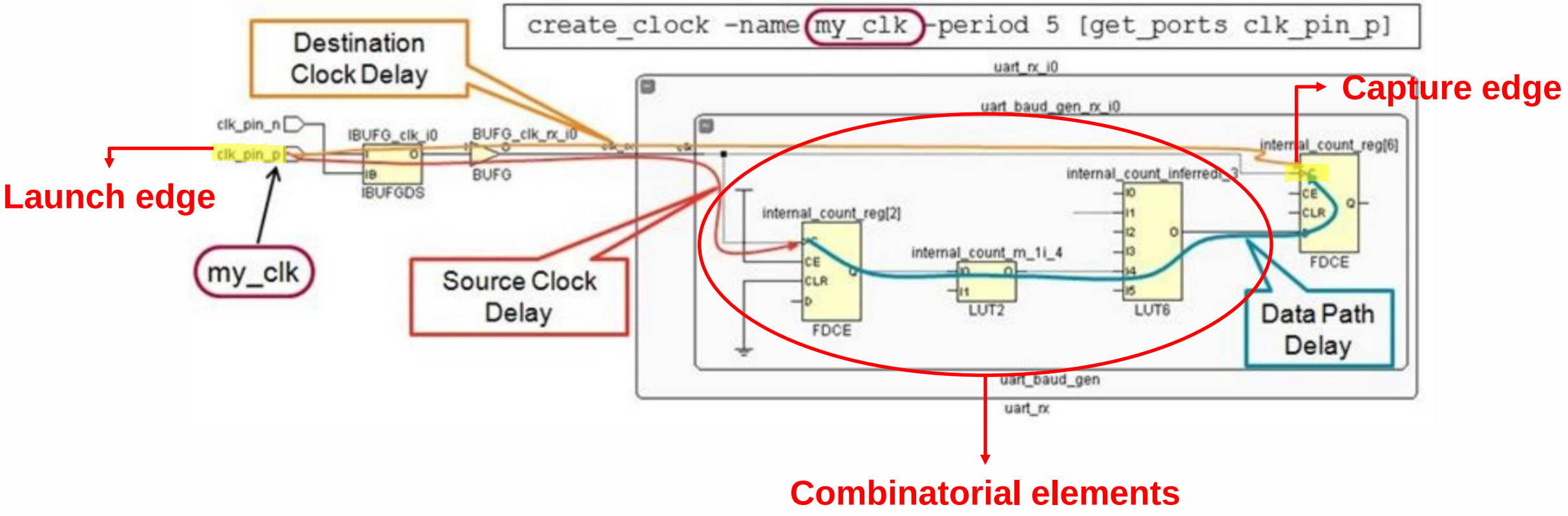
Clocks

- 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)

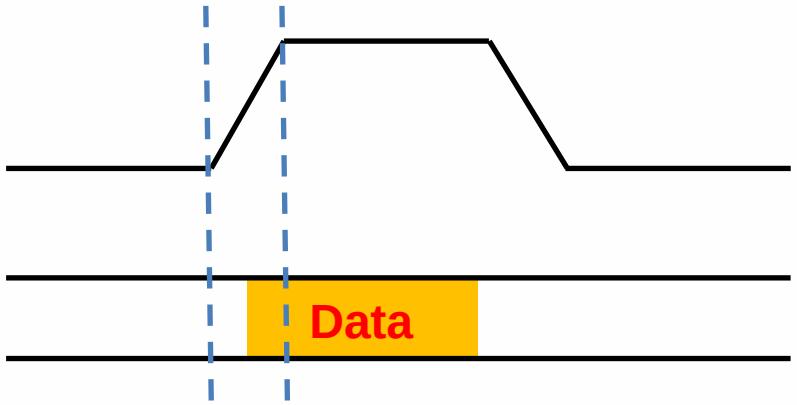
- Analyze the timing performance but not function in circuit



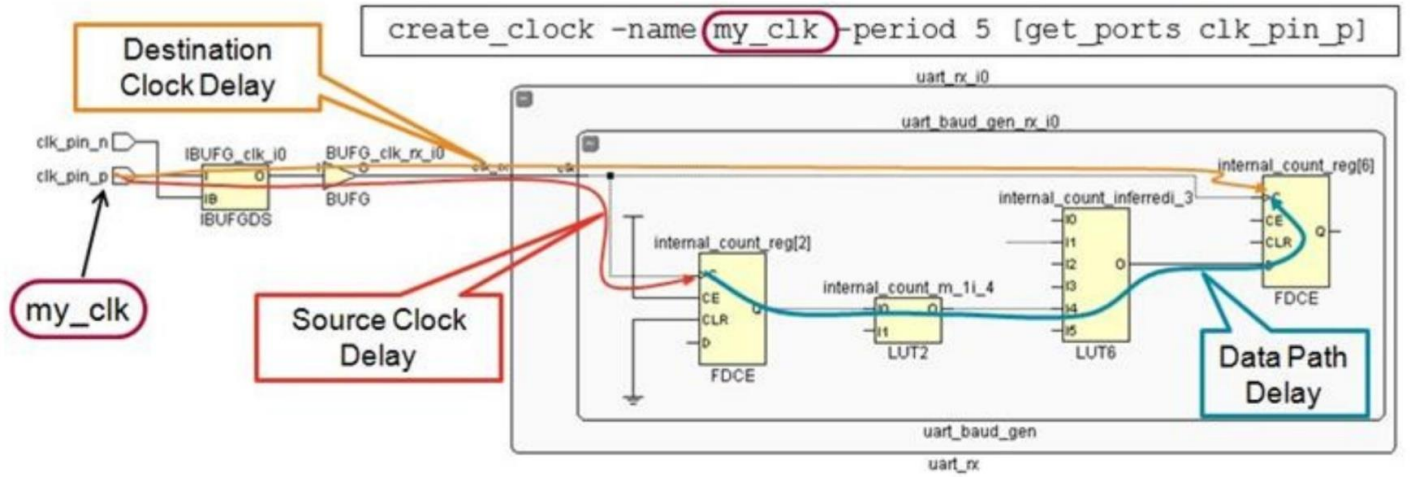
Static Timing Analysis (STA)

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.



Data has not arrived when Setup Time



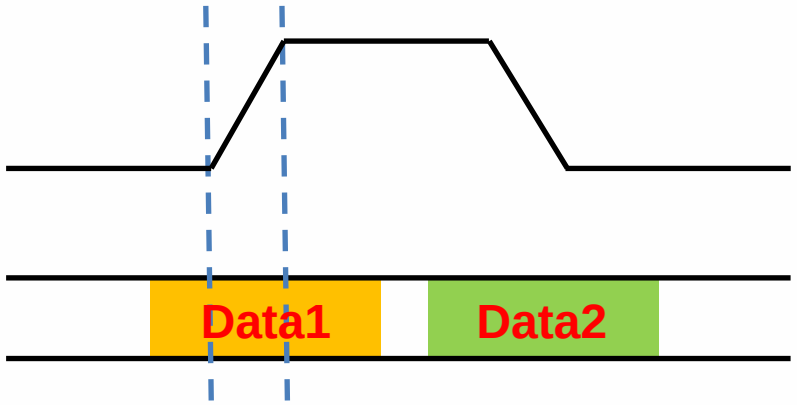
Setup Slack = $T_{Period} + T_{Destination} - T_{source} - T_{Data} - T_{Uncertainty} > 0$

XILINX.

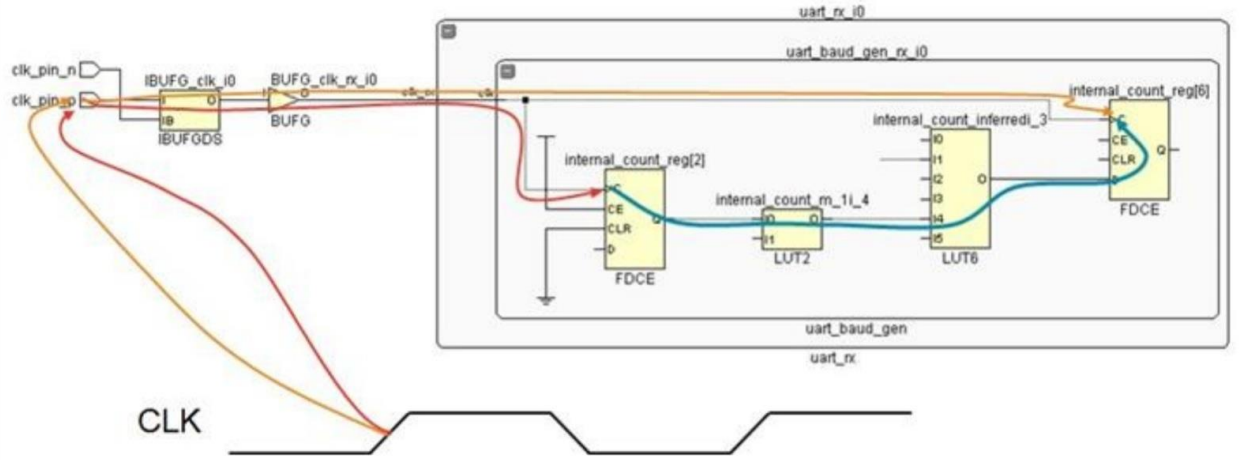
Static Timing Analysis (STA)

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.



The second data (Data2) is transmitted too fast



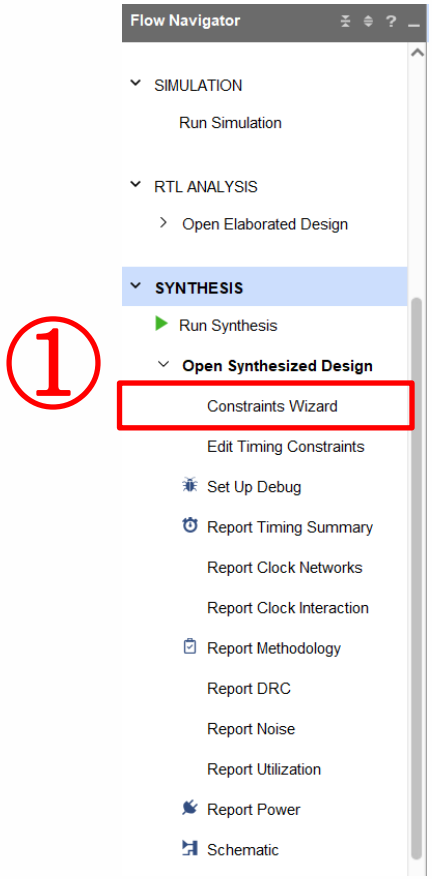
Hold Slack = $T_{source} + T_{Data} + T_{Uncertainty} - T_{Destination} > 0$

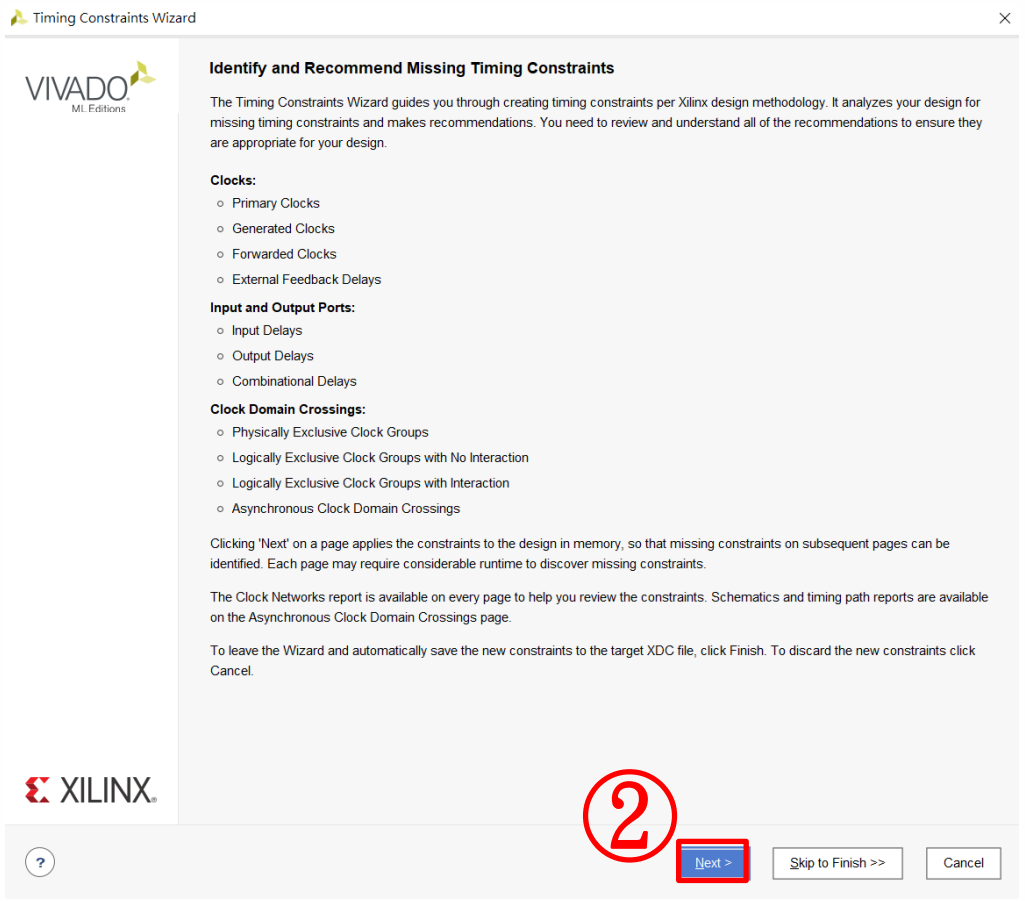
Timing Constraints Wizard

- Can be run after Synthesis or Implementation
- Consists of Constraints Wizard and Edit Timing Constraints

Timing Constraints Wizard

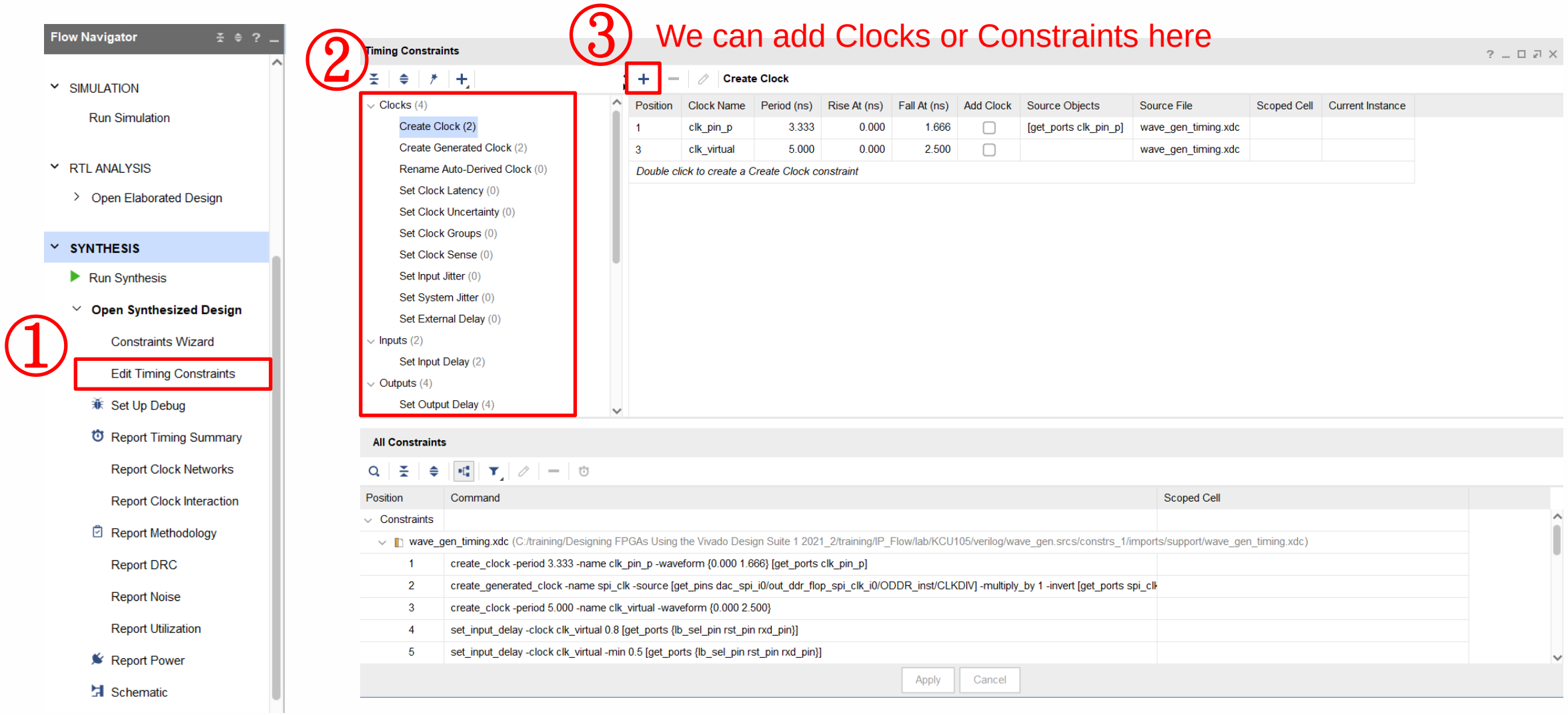
- Constraints Wizard
  - Vivado will auto recognize the clocks with no constraints, and create constraints in order of Primary CLK, Generated CLK, I/O delay Constraints,… etc.





Timing Constraints Wizard

- Edit Timing Constraints

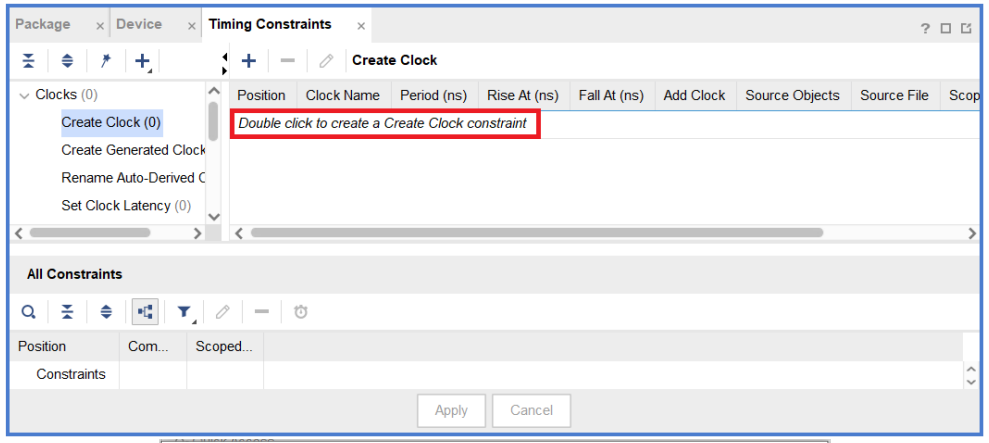


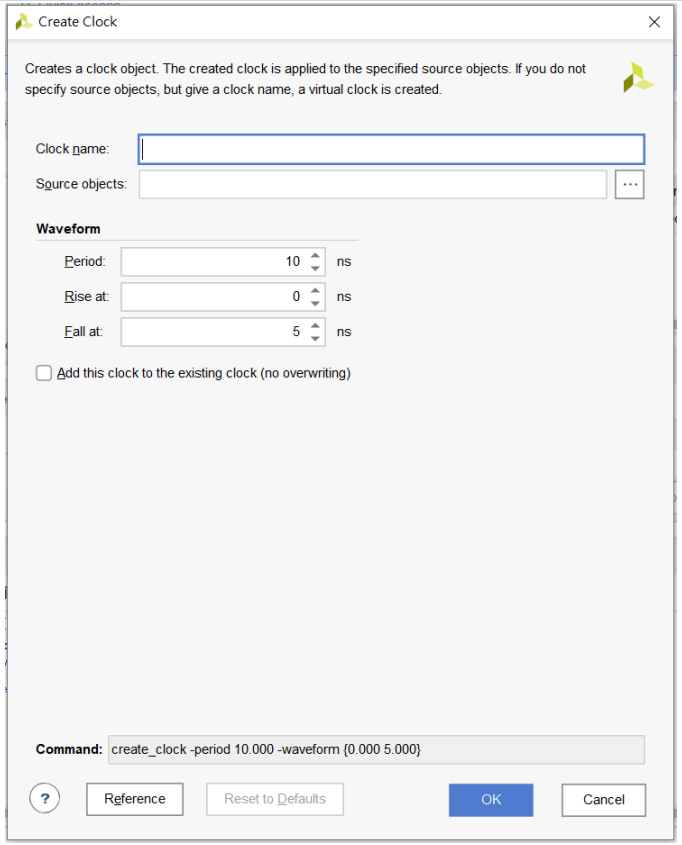
Timing Constraints Wizard

- Edit Timing Constraints

> Move cursor to “Double click to create a Create Clock constraint”, then double click.

> “Create Clock” window pops out



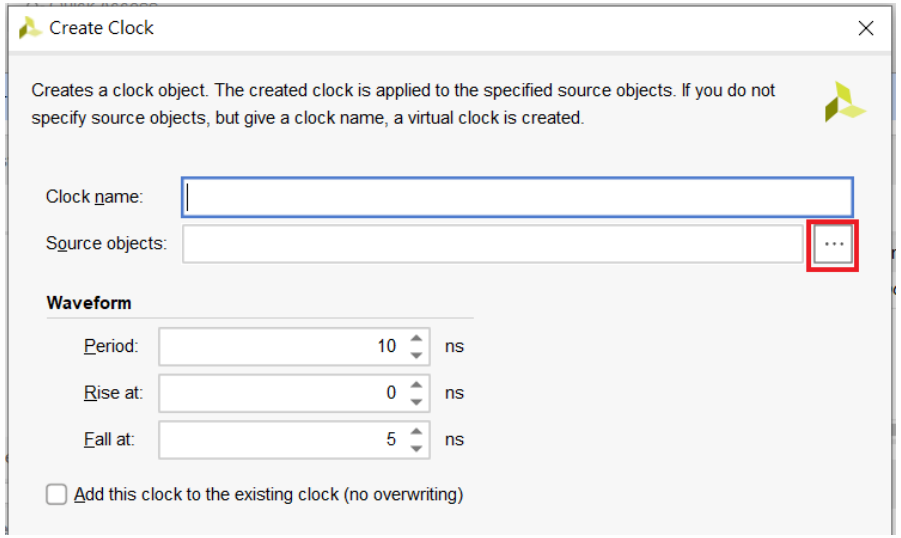


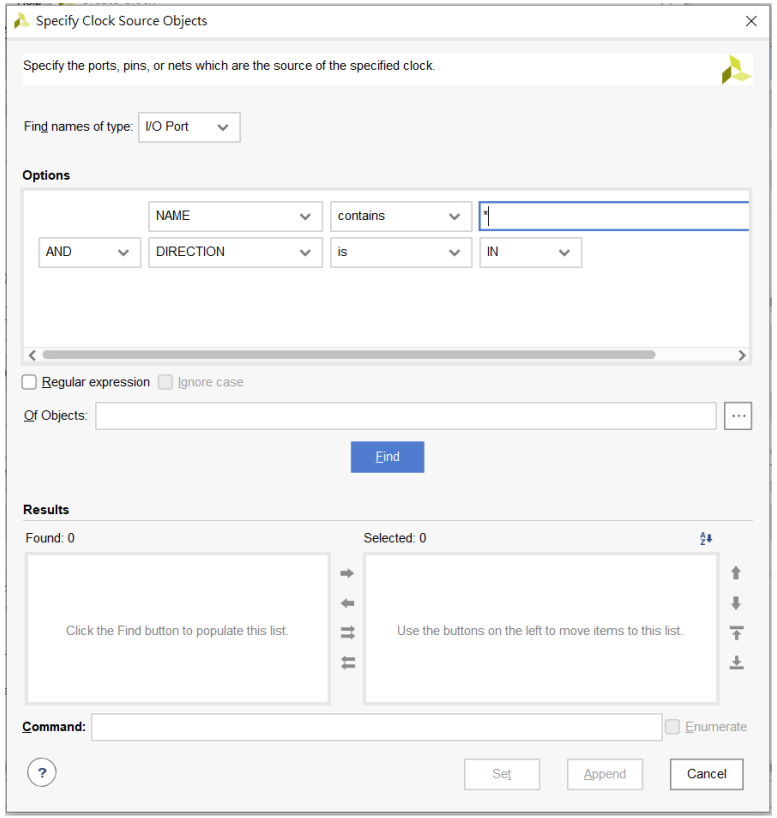
Timing Constraints Wizard

- 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.



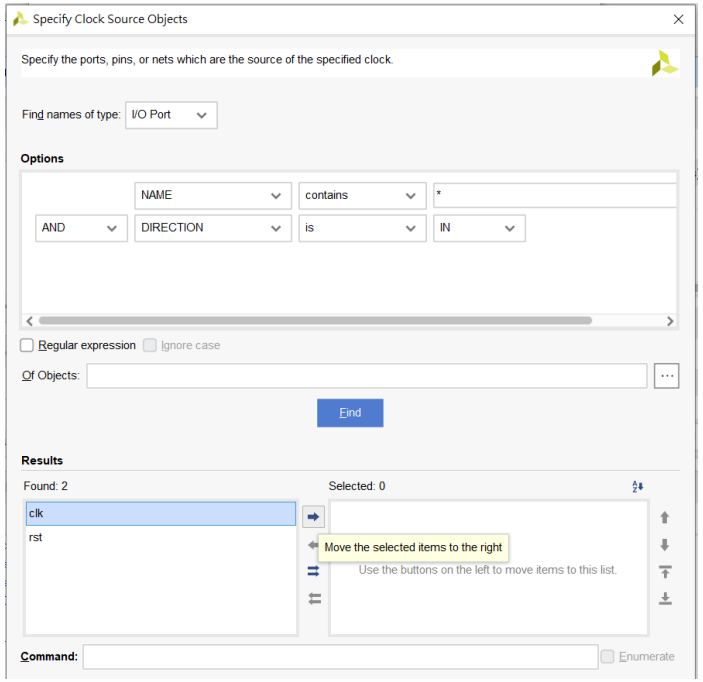


Timing Constraints Wizard

- Edit Timing Constraints

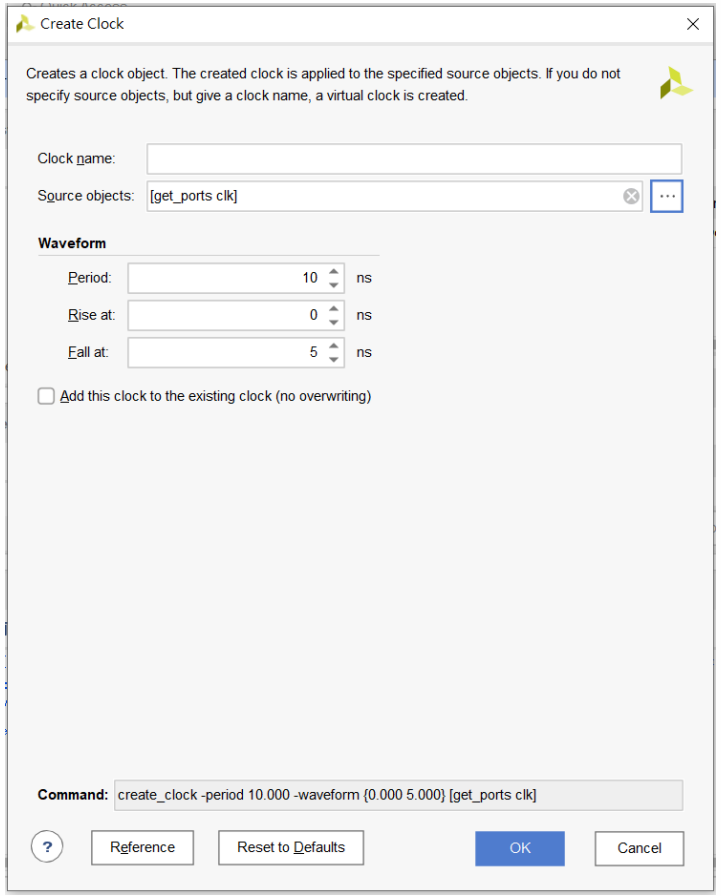
> In this tutorial, randomly choose “clk” as an object to add a simple timing constraint on it. After choosing an object, move the select object to the right.

> 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.



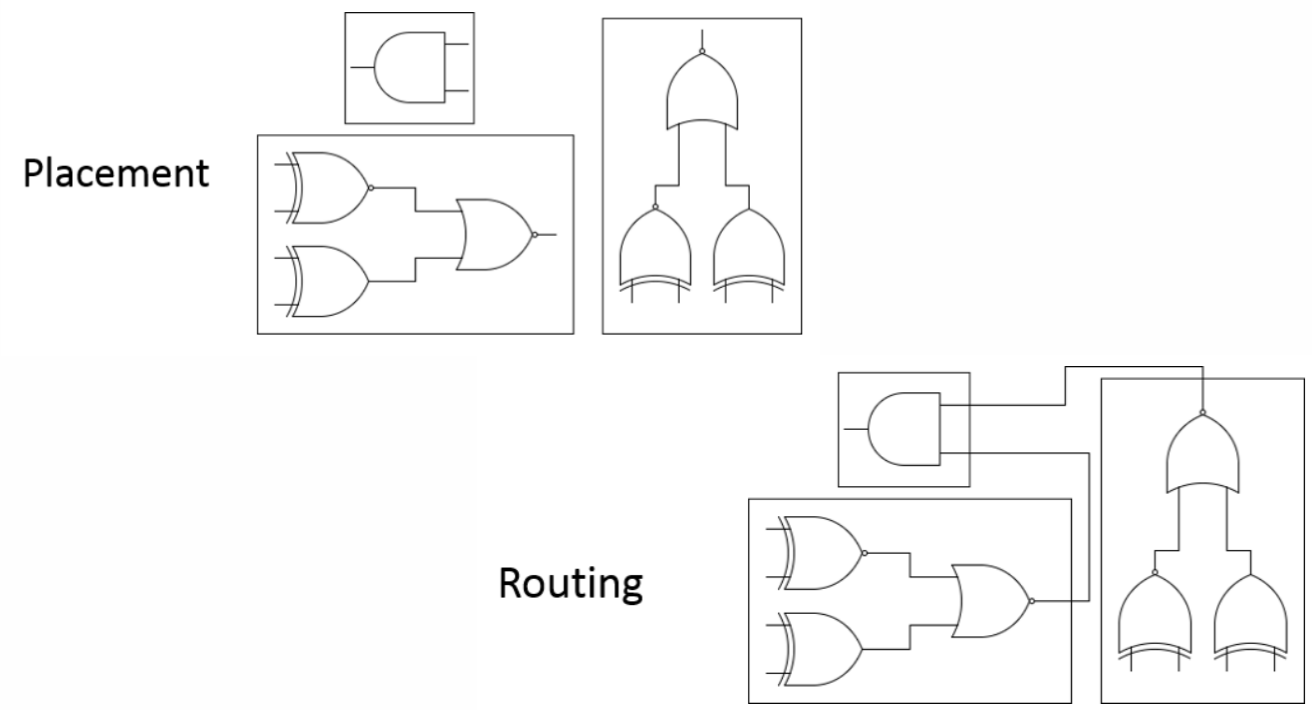
Vivado Implementation

XILINX.

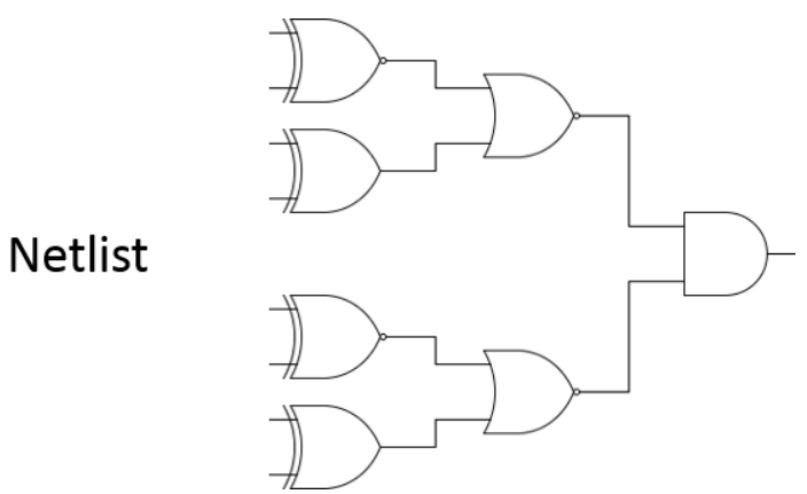
Vivado Implementation

- 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

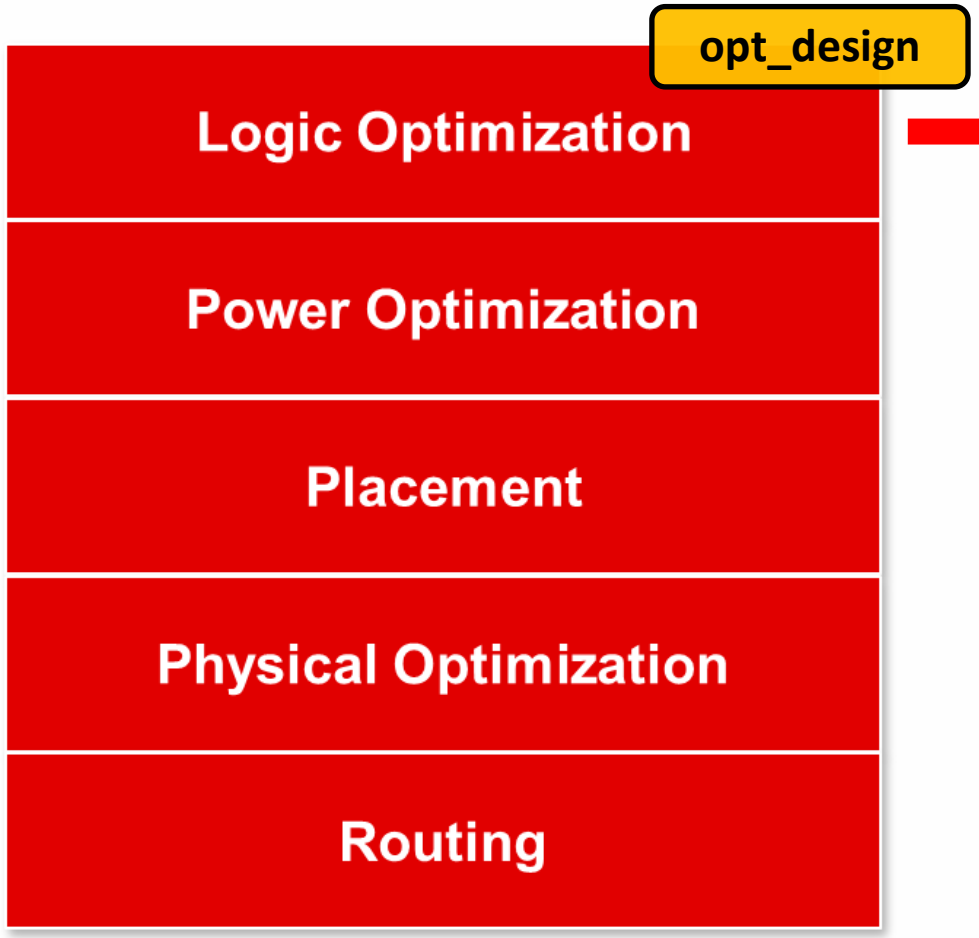


Synthesis



Vivado Implementation Processes

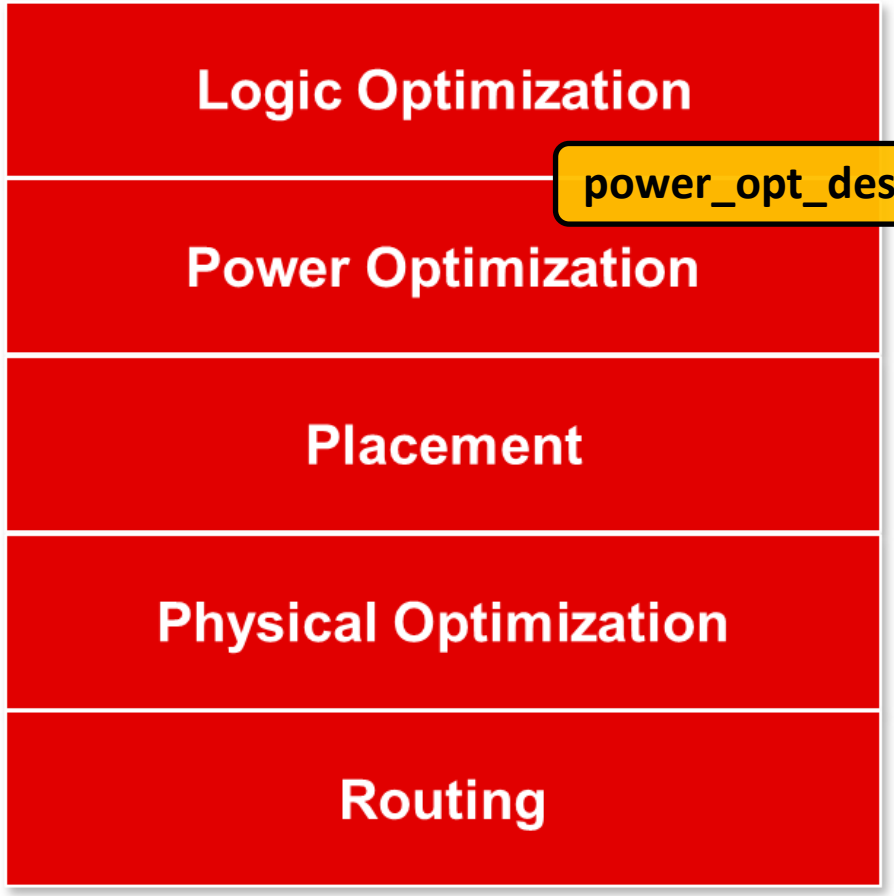
- 5 Processes but one is optional, and all will generate .dcp file



This will remove some useless or redundancy logic which Vivado judges

Vivado Implementation Processes

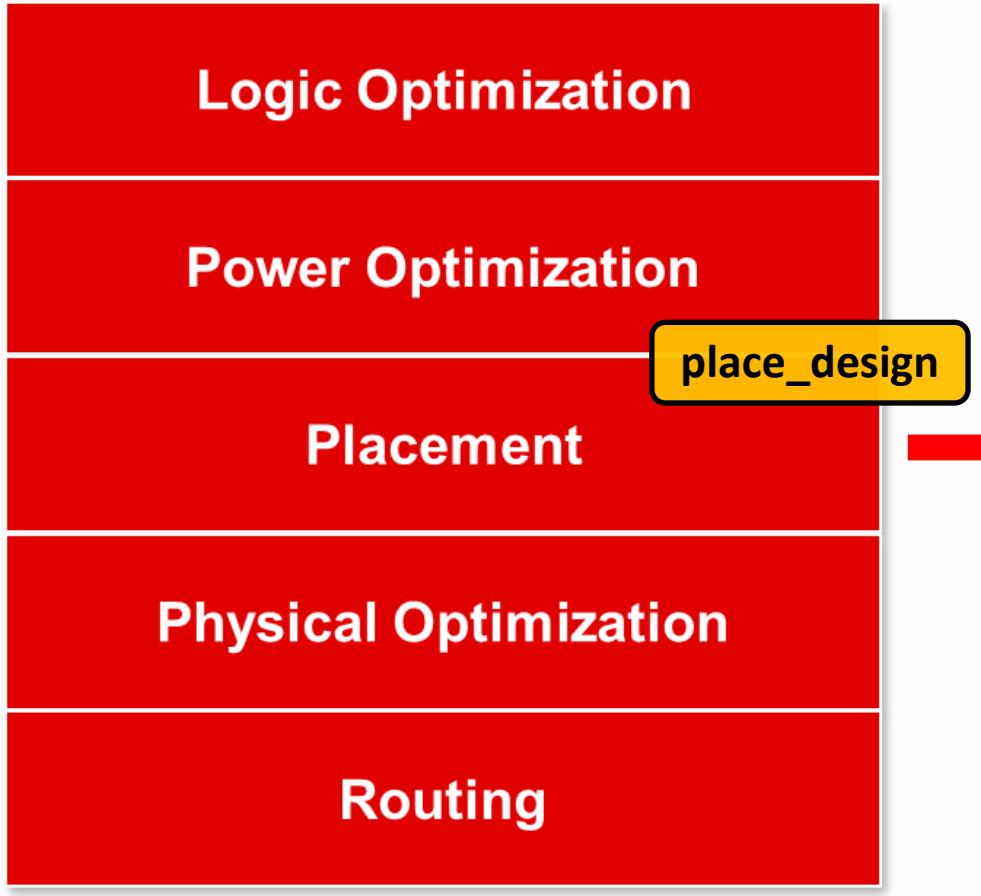
- 5 Processes but one is optional, and all will generate .dcp file



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

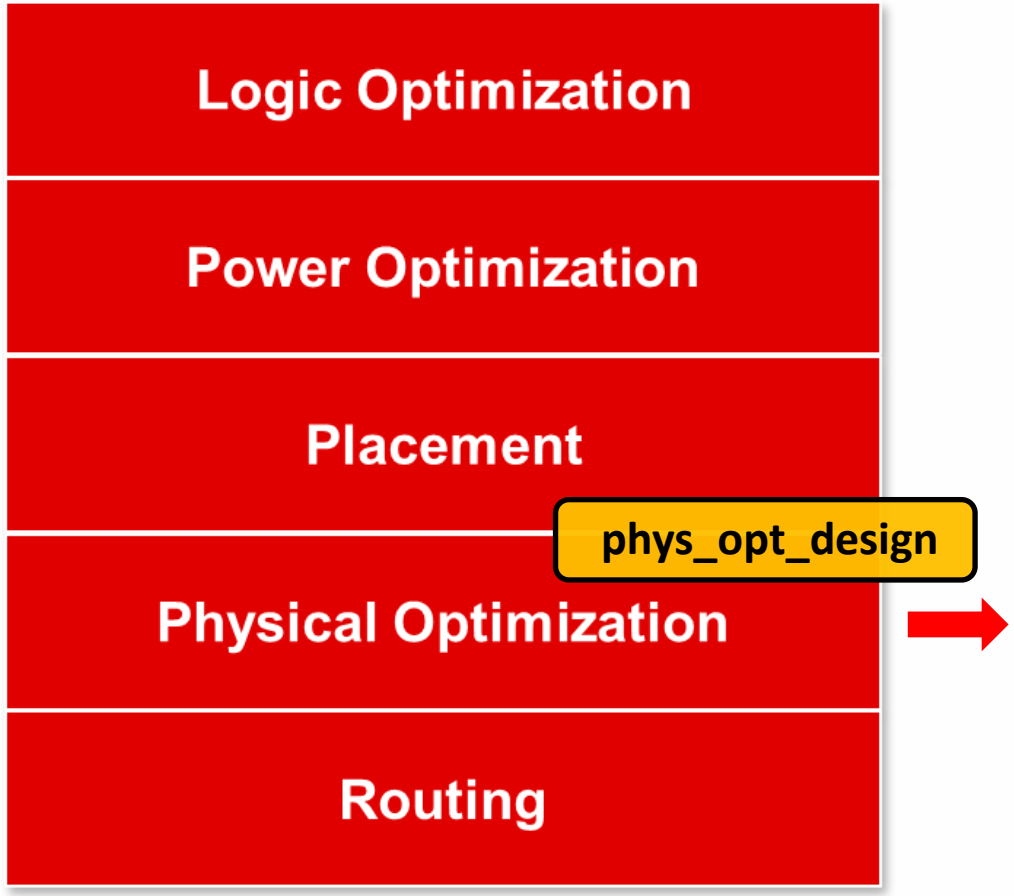


Vivado will run DRC before placement, and considering three points including Timing Slack, Wirelength and Congestion.

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

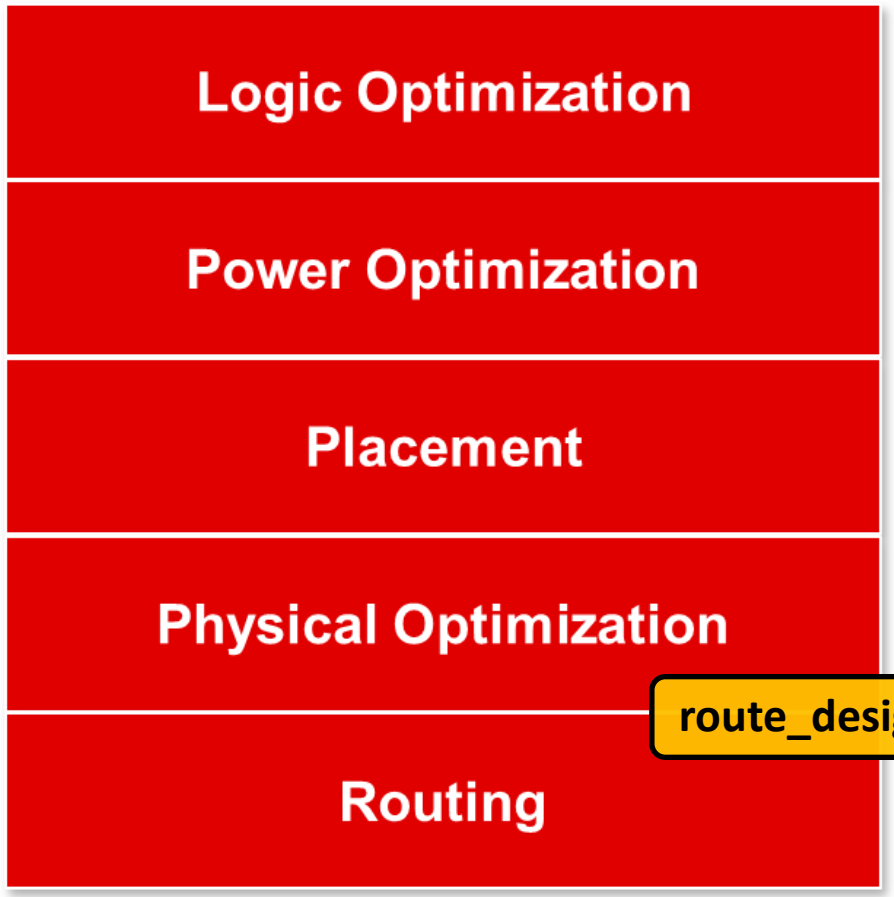


Available after doing place_design and route_design.

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**

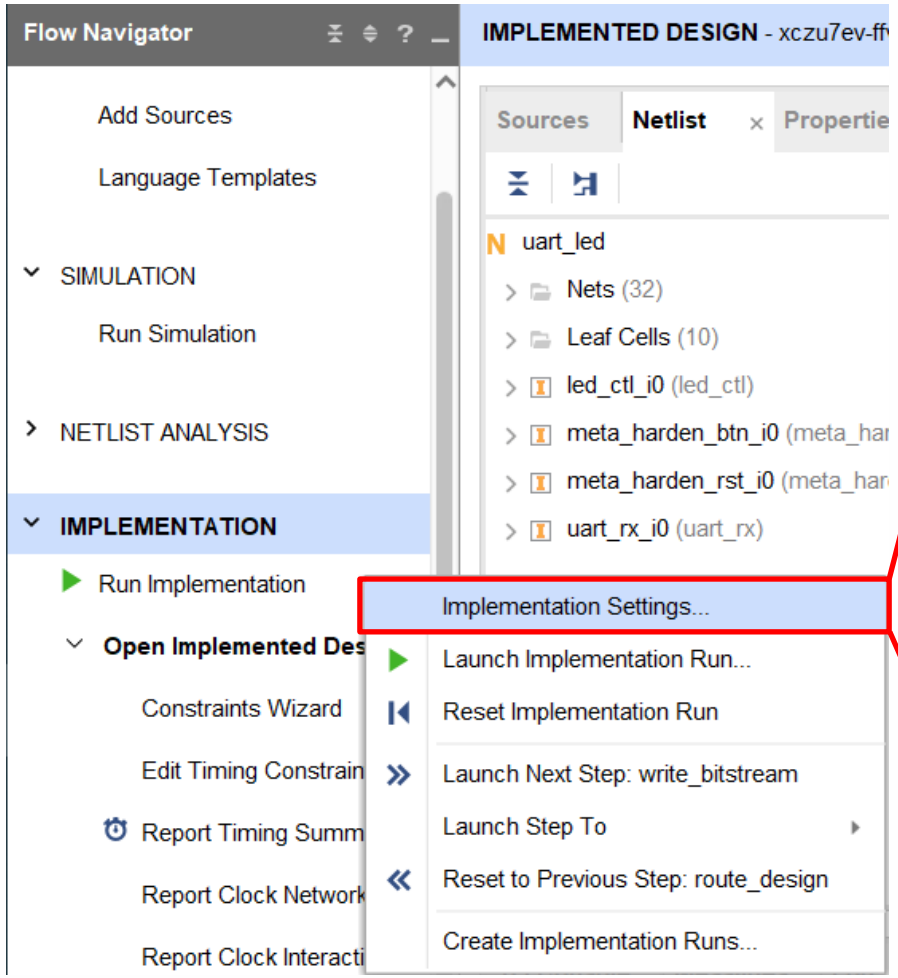
  Default mode. Compile only the modified parts without starting from scratch.

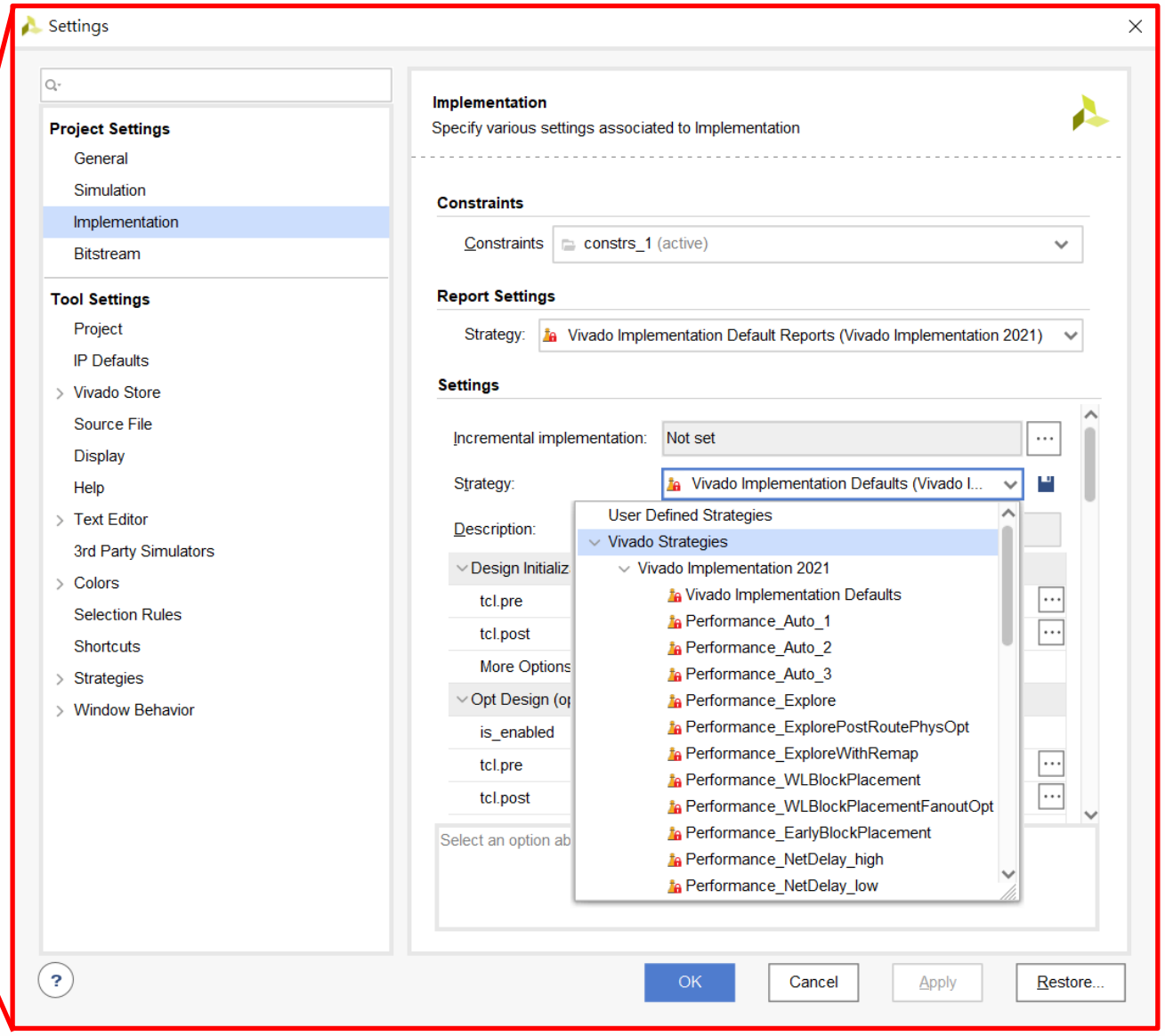
- **Re-Entrant**

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

→ Normal and Re-Entrant mode

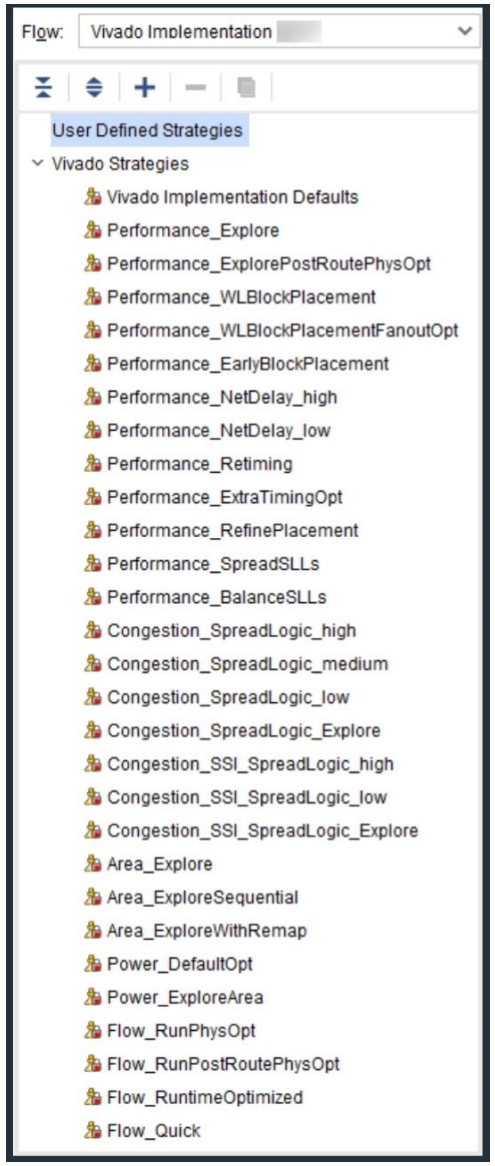
Vivado Implementation Strategy





XILINX.

Vivado Implementation Strategy

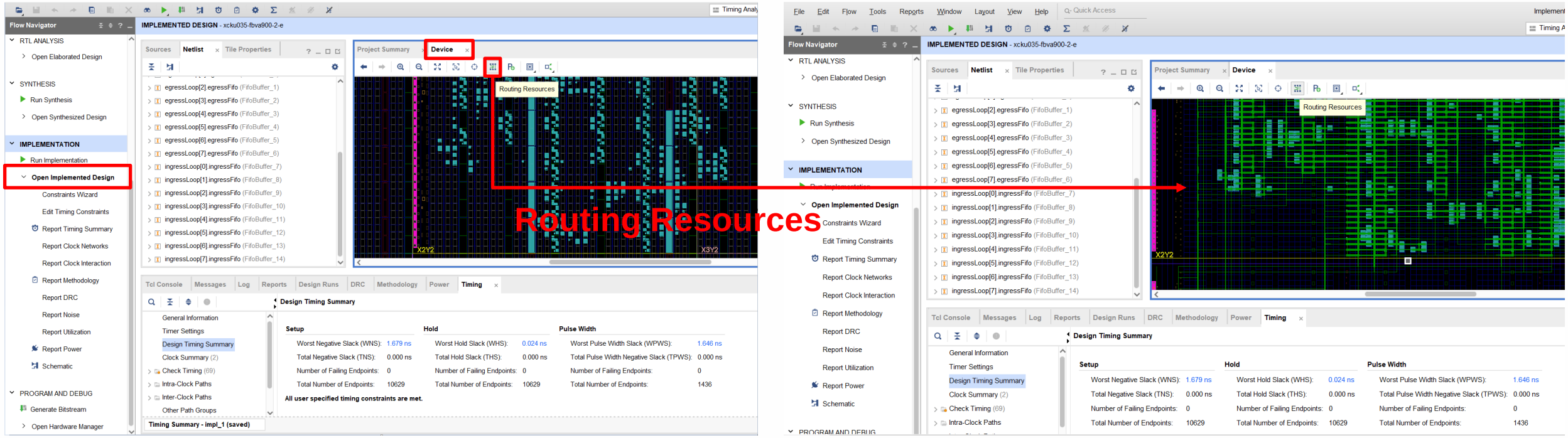


category	purpose
Performance	Increasing performance
Area	Optimize design to use minimal LUTs
Power	Optimize the power consumption
Flow	Optimize overall flows
Congestion	Decrease congestion

- **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.



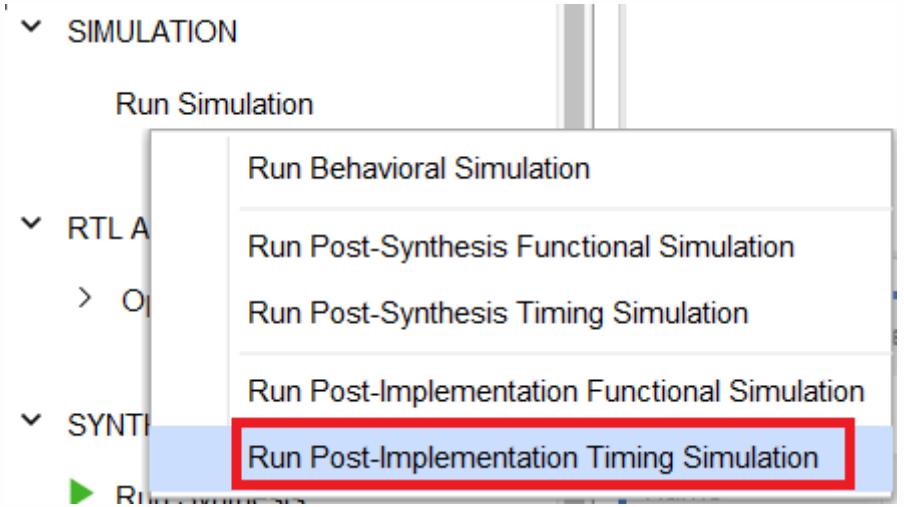
Vivado Post-Implementation Timing Simulation

- 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.

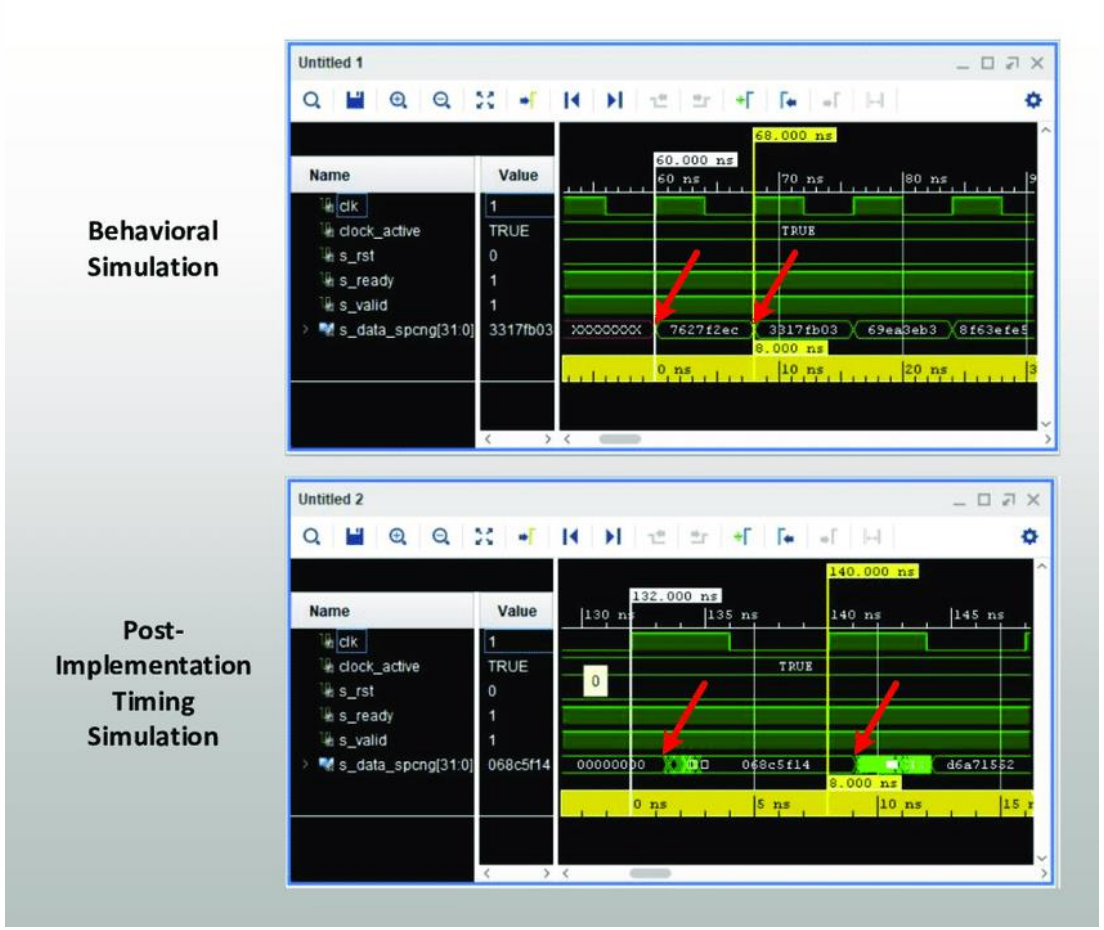
Vivado Post-Implementation Timing Simulation

- 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.



XILINX.

Vivado Design Checkpoint

XILINX.

Design Checkpoint

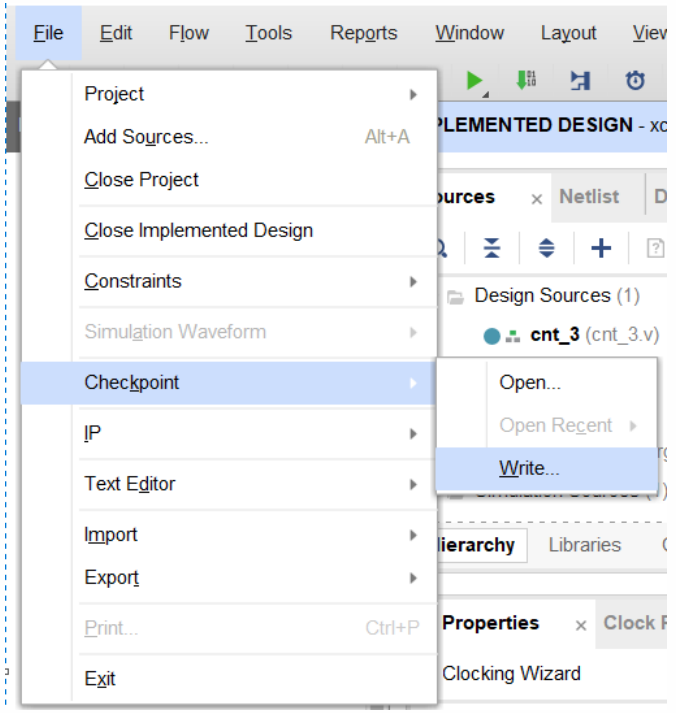
- The Vivado Design Suite uses a physical design database to store placement and routing information.

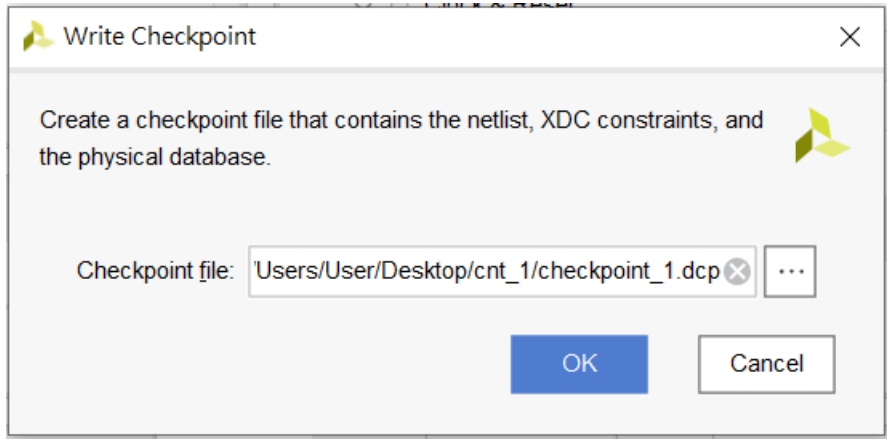
- Design checkpoint files (.dcp) allow you to save and restore this physical database at key points in the design flow.

- Design ckeckpoint file includes:
  - Current netlist, including any optimizations made during implementation
  - Design constraints
  - Implementation results

Writing Checkpoint Files

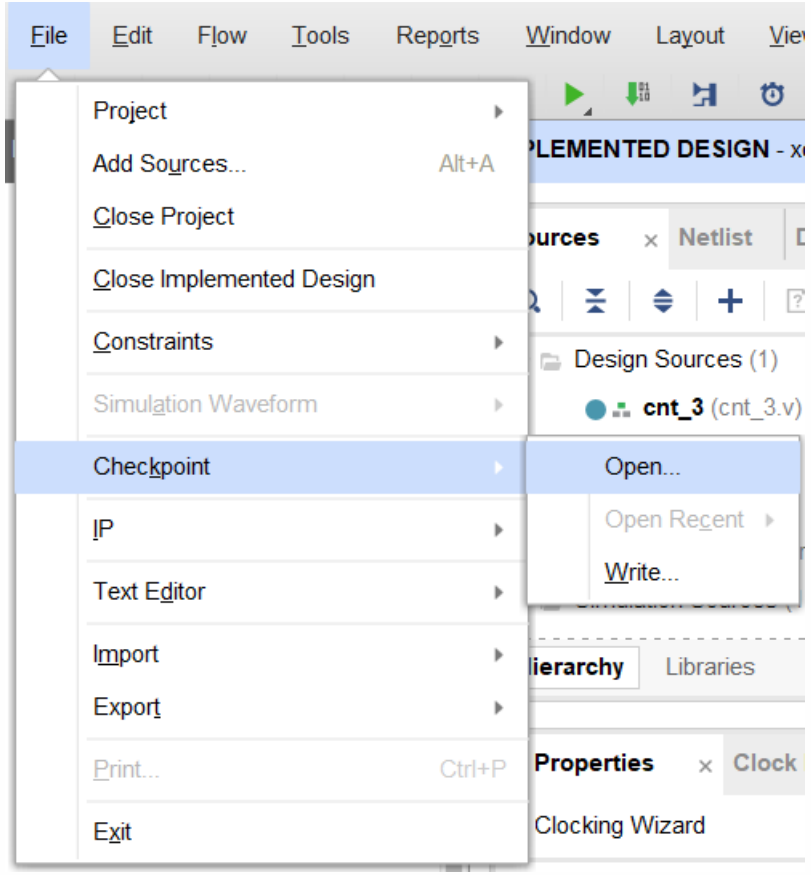
- Run “File” > “Checkpoint” > ”Write” to capture a snapshot of the design database at any point in the flow.
- Then the ”Write Checkpoint” window pops. It shows where the checkpoint saves, and the directory is changeable.
- This creates a file with a dcp extension.





Reading Checkpoint Files

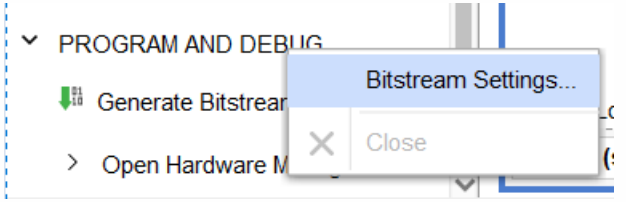
- Run “File” > “Checkpoint” > “Open” to open the checkpoint in the Vivado Design Suite.
- The design checkpoint is opened as a separate project.



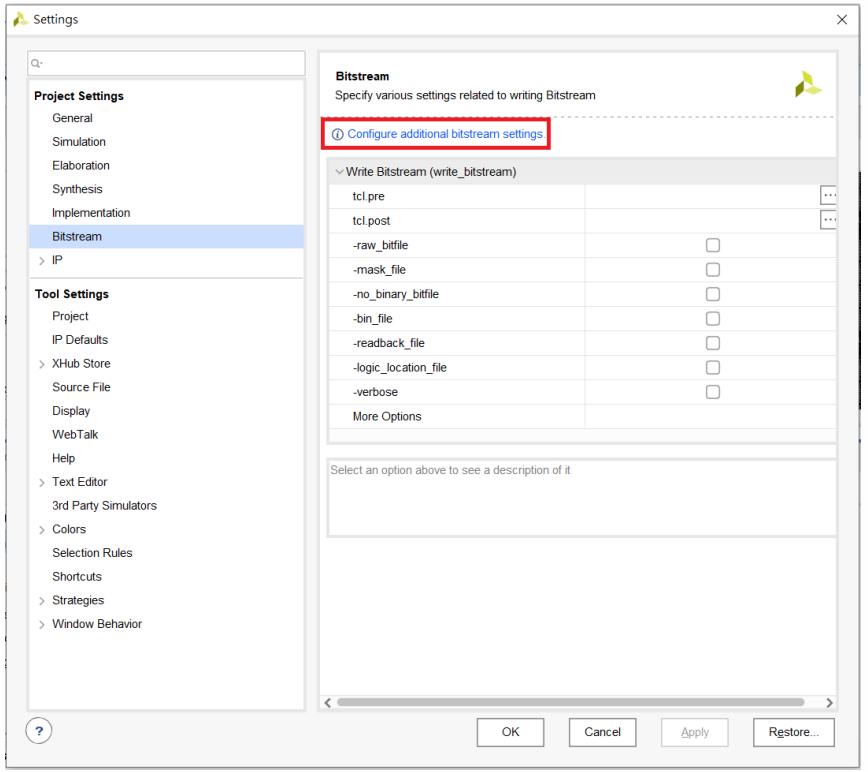
Programming Devices

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”.

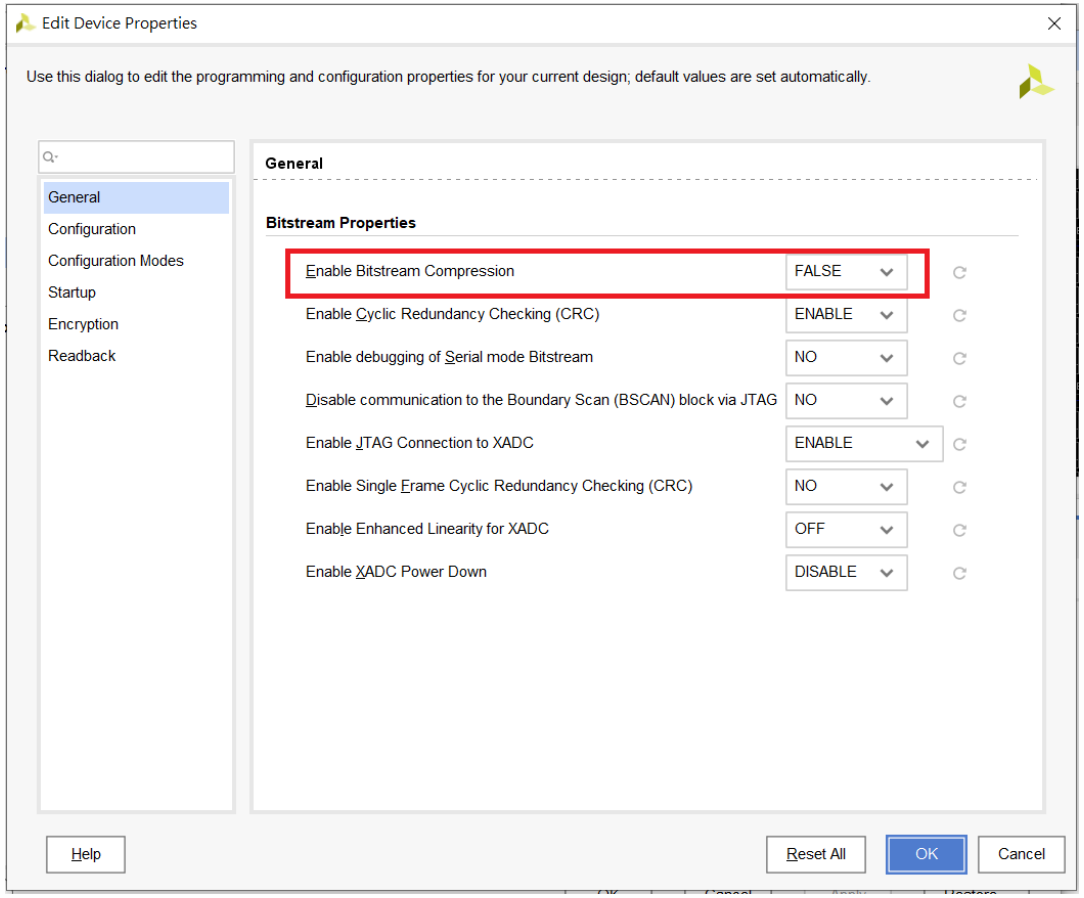


Bitstream Settings

- By clicking “Configure additional bitstream settings”, the “Edit Device Properties” window shows.

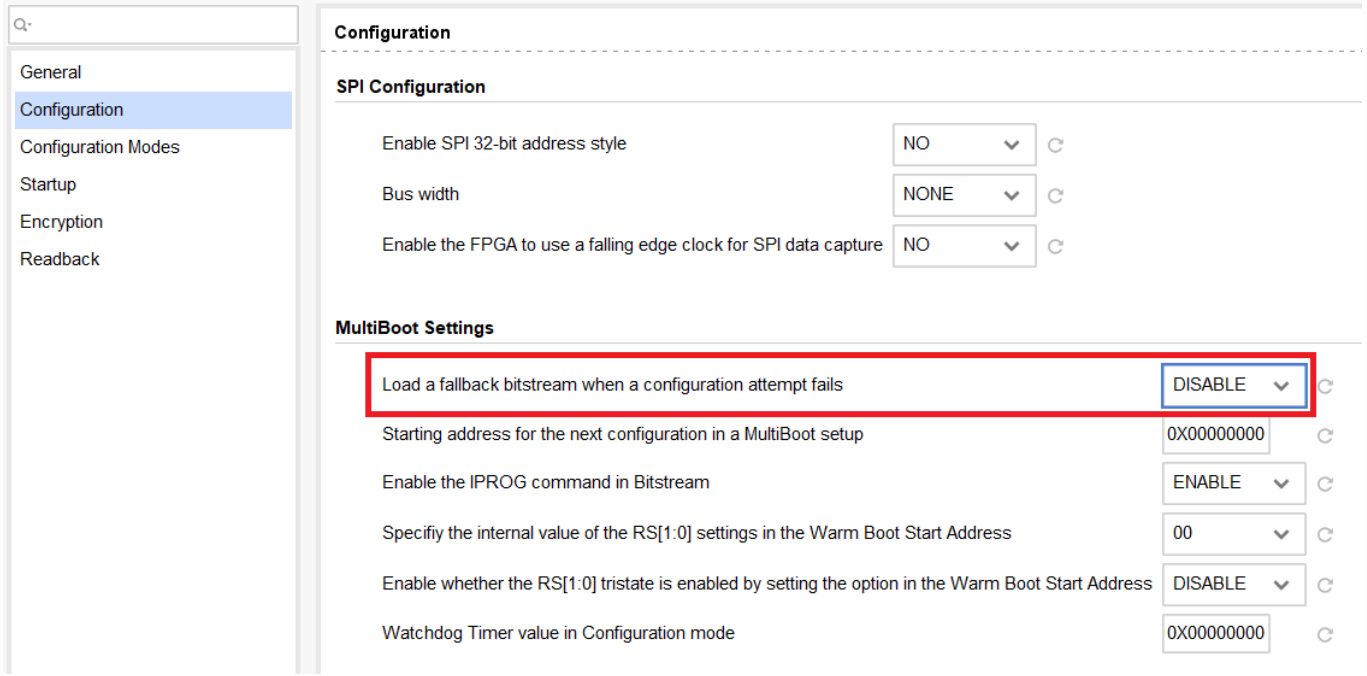
- Some options are important and useful.

  In ”General” > ”Enable Bitstream Compress” compresses the bitstream files, make the whole download process faster.



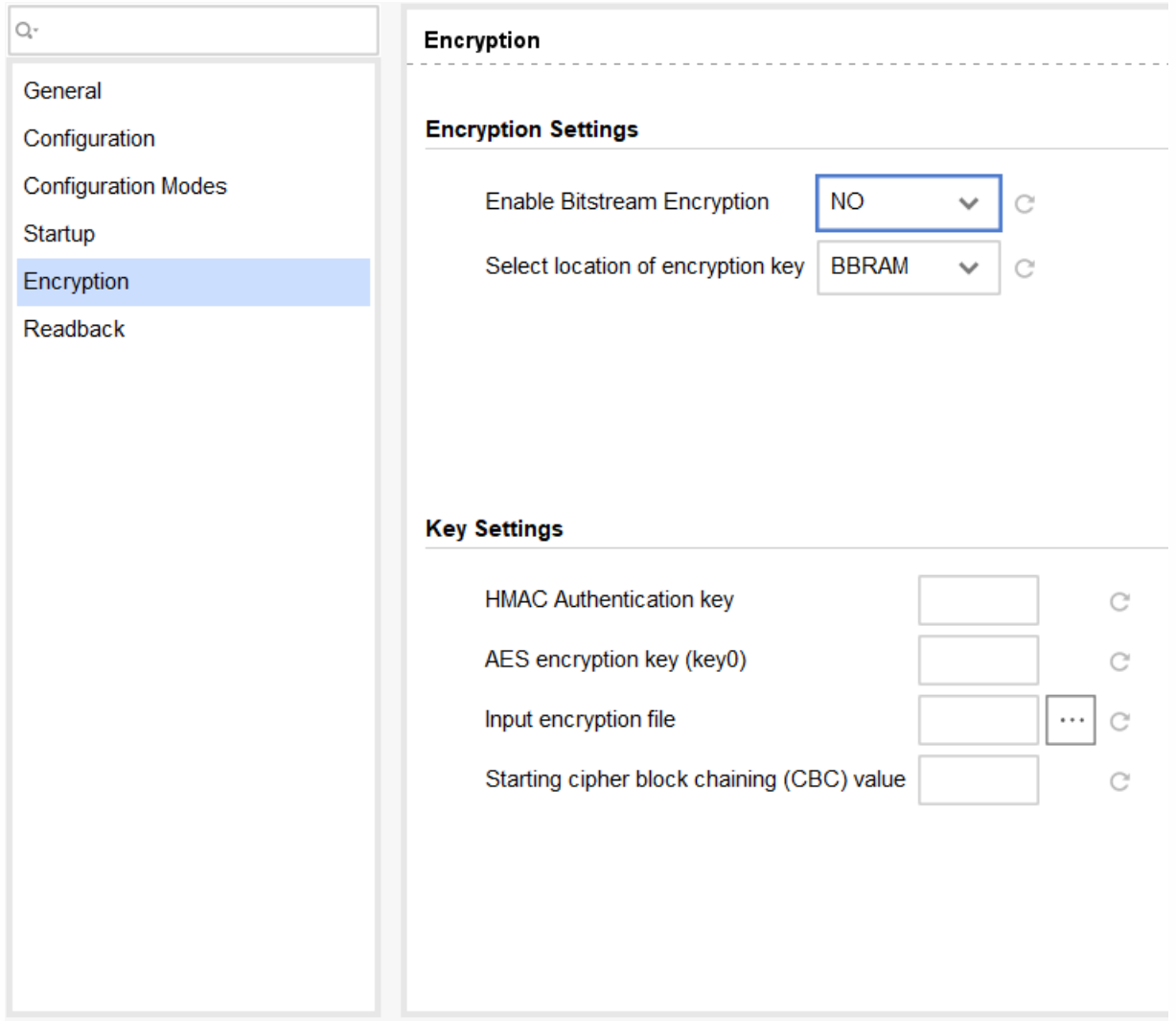
Bitstream Settings

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



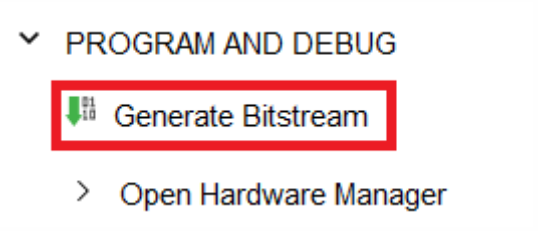
Bitstream Settings

- 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.

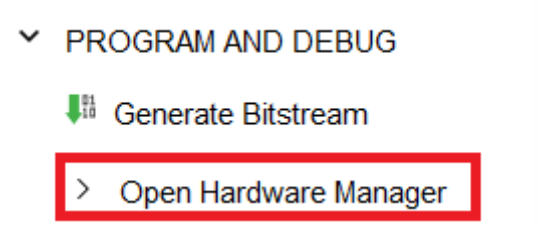


Generate Bitstream/ Open Hardware Manager

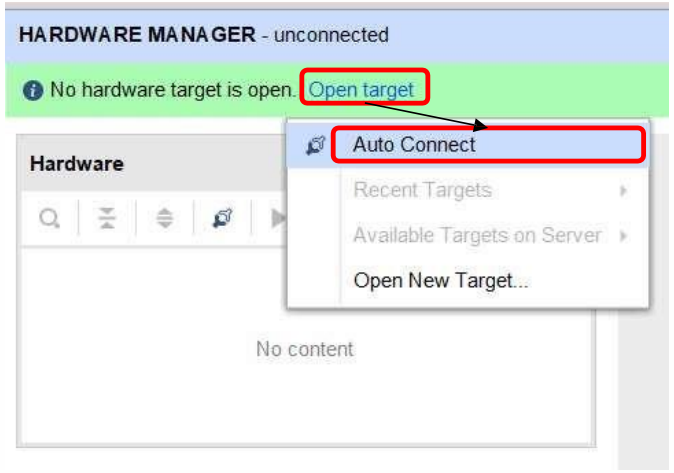
- Click ”Generate Bitstream” to create bitstream file



- Once the bitstream generation is done, click ”Open Hardware Manager”

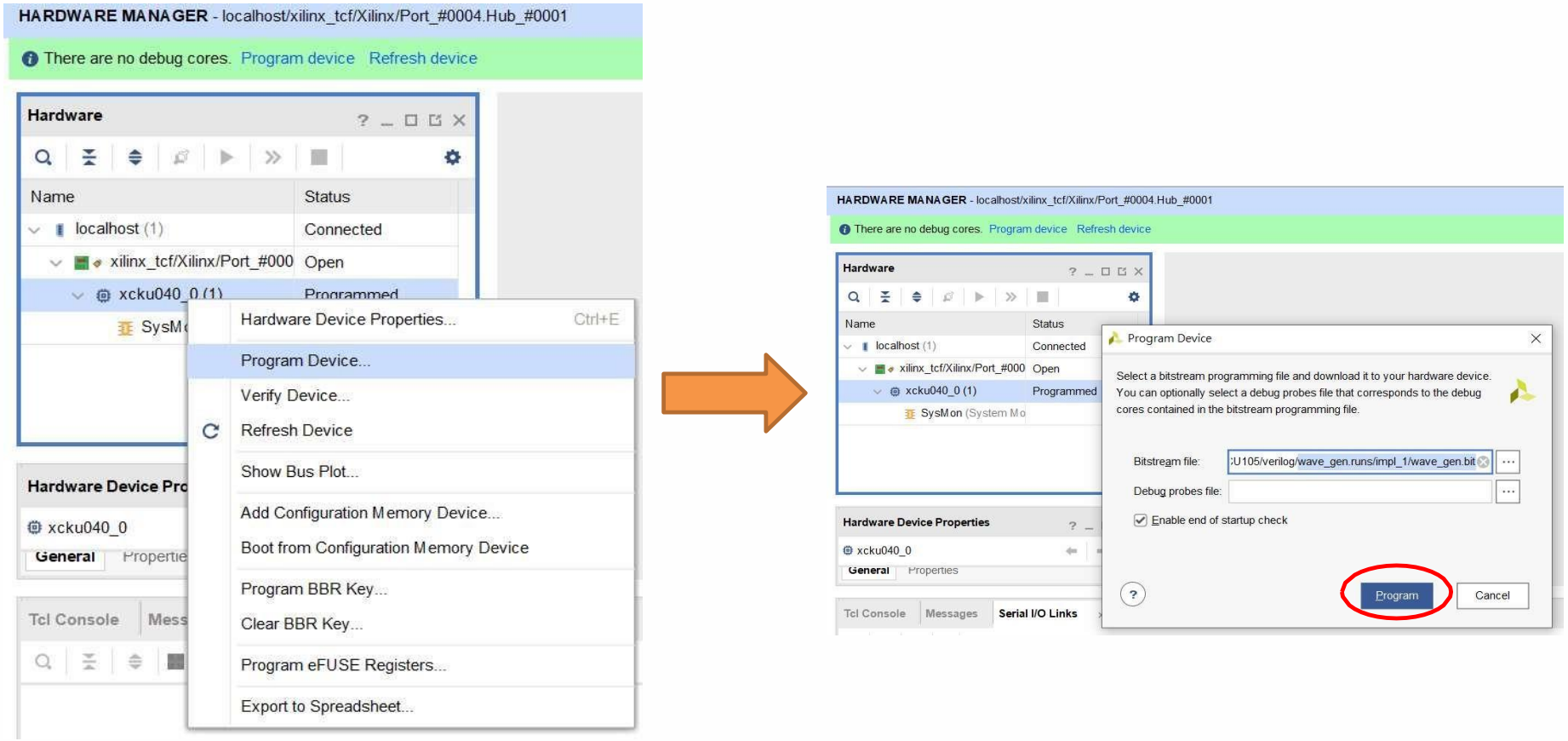


- Select “Open target”, then click ”Auto Connect”
- Make sure the device is connected to PC that runs Vivado, and then the information bar suggest that open a new target or an existing one.



Programming Device

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

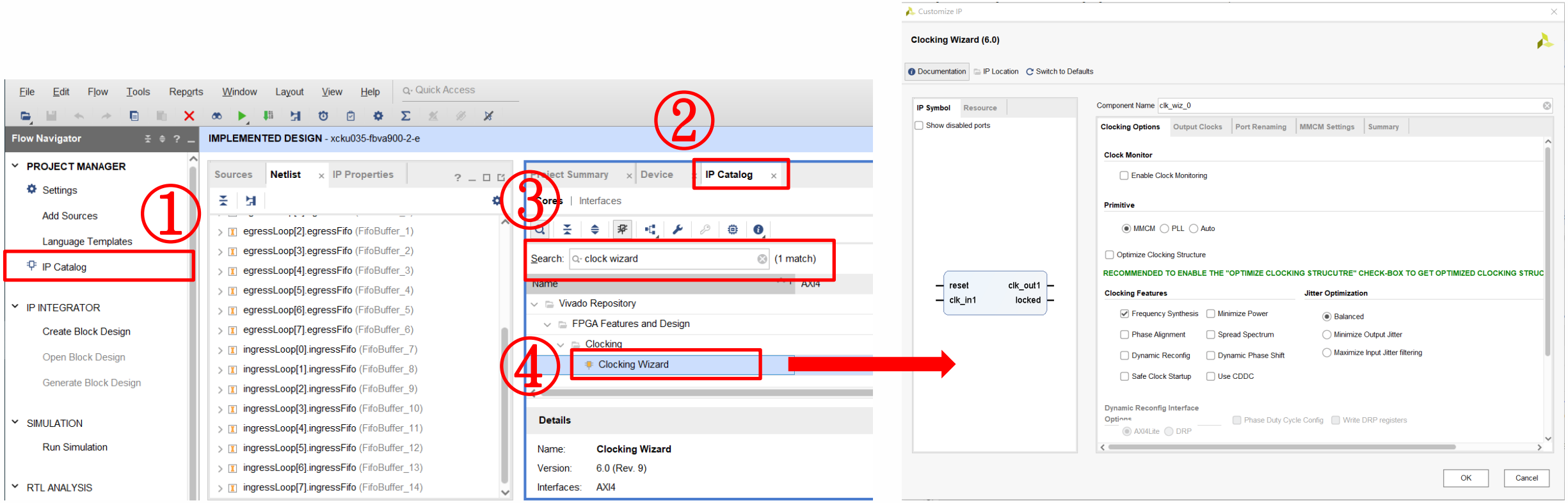


Vivado IP flow

XILINX.

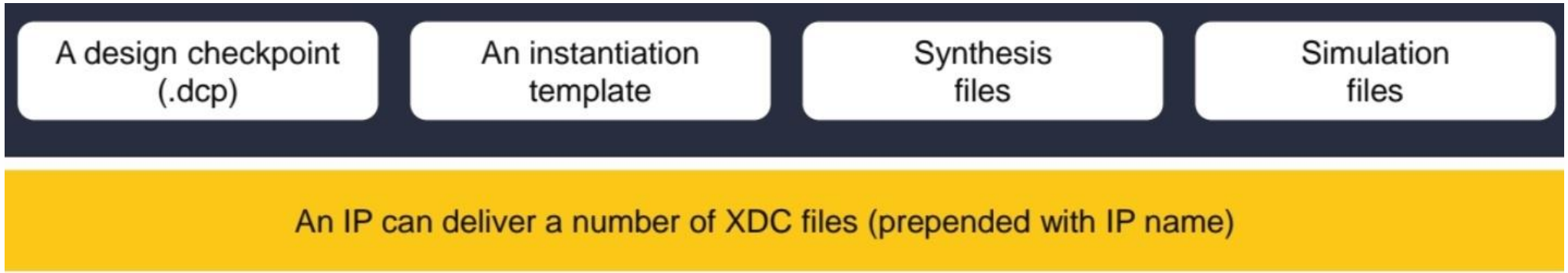
Vivado IP flow

- 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.



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

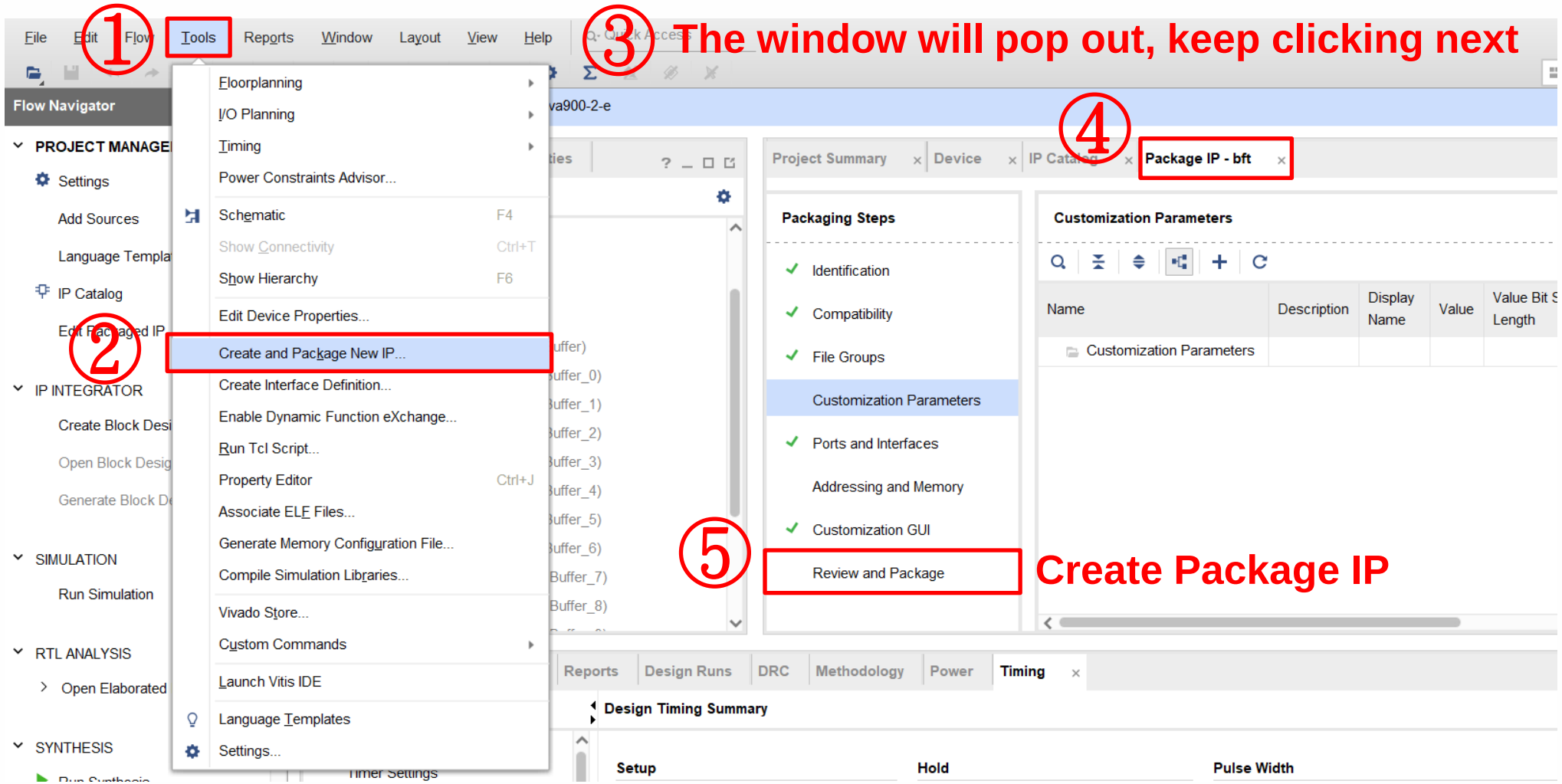
- Out of Context (OOC)
  1. The default way of IP synthesis, this will create some files including .dcp, .xci, changelog,… etc.
  2. If a certain IP core not be modified after synthesizing, only synthesize other user logic which is modified. This way will reduce synthesis time massively.

- 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

XILINX.

Vivado Logic Analyzer

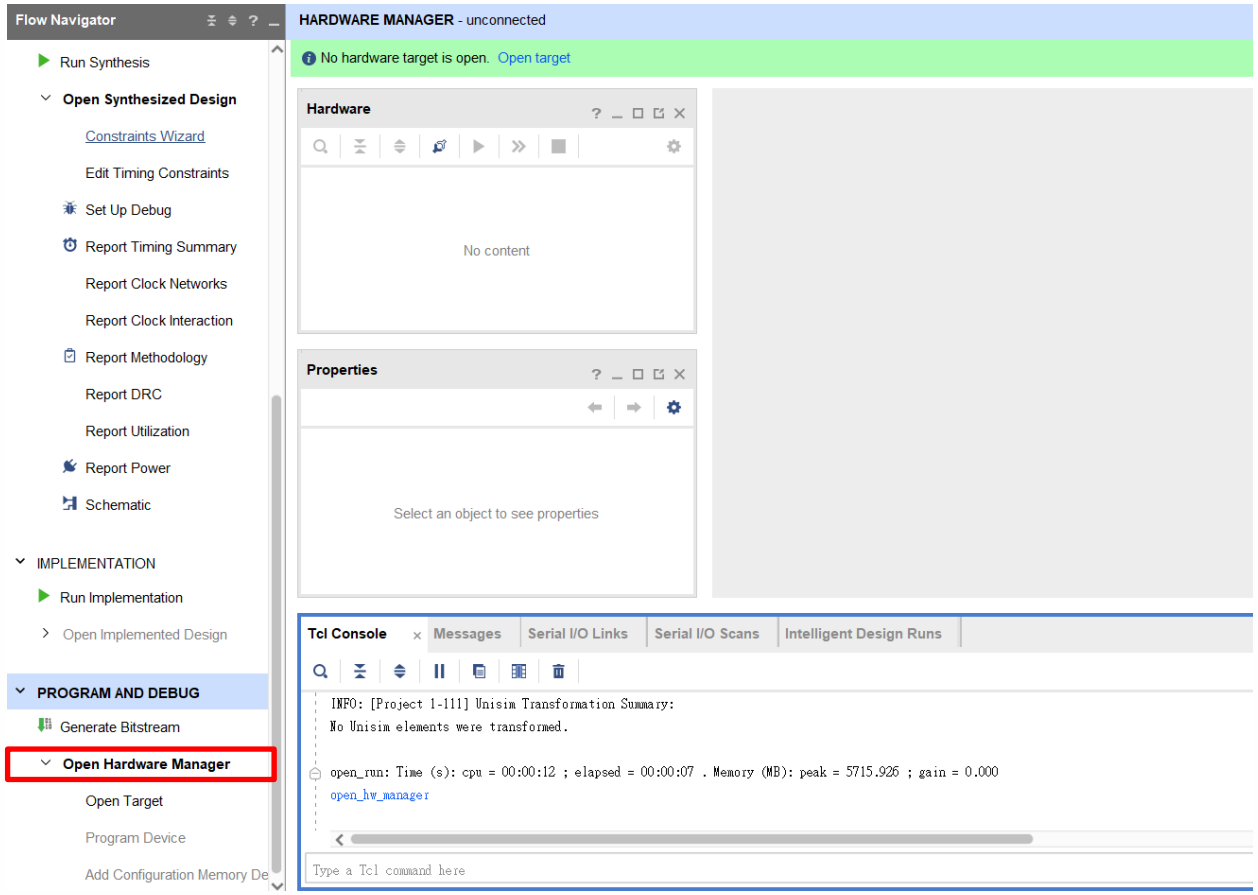
- Main purpose
  1. Verification
  2. Debugging
  3. Data Capture

- Contains two main parts
  1. Debug Cores

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

  2. Debug Flow

     HDL instantiation and netlist insertion



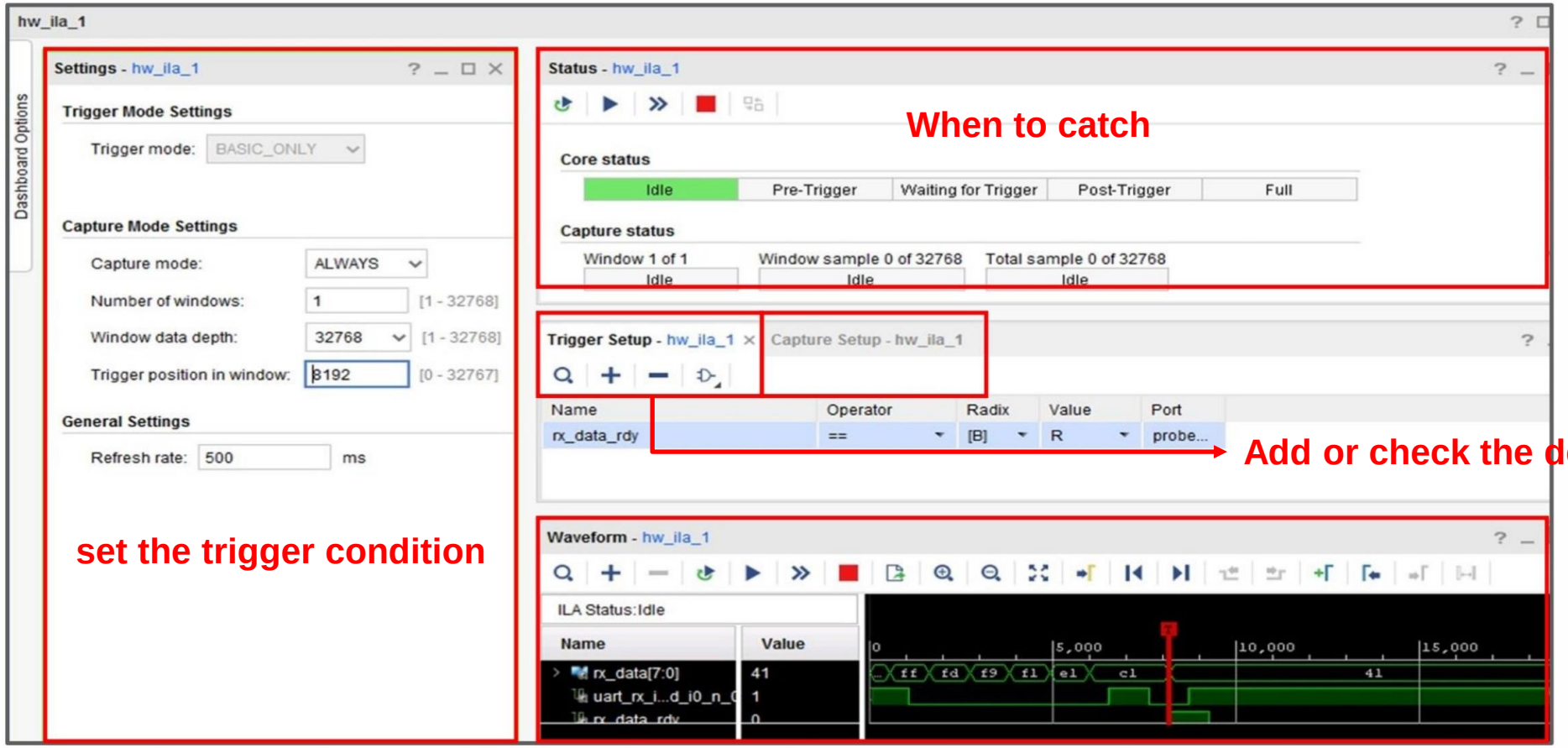
Vivado Logic Analyzer

- Debug Flow

HDL Instantiation Flow		Netlist Insertion 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.	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

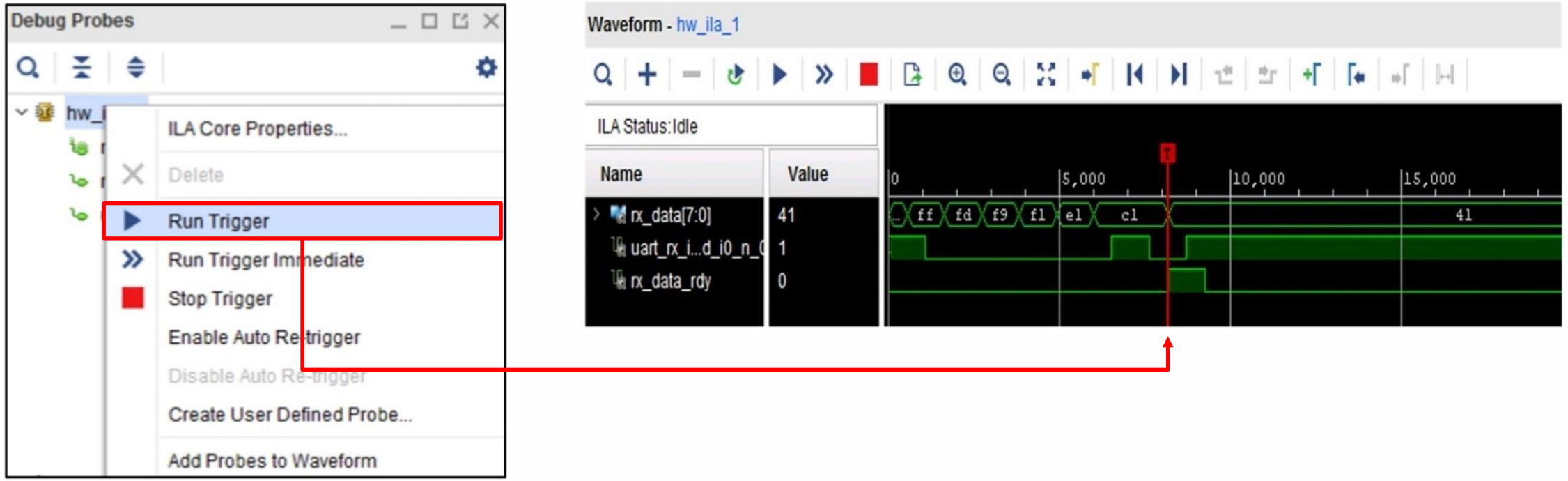
- Triggering
  - The way of debug. We can check the situation of a certain IP by setting the trigger condition.
  - IP catalog offers ILA cores that can be set the trigger condition.



XILINX.

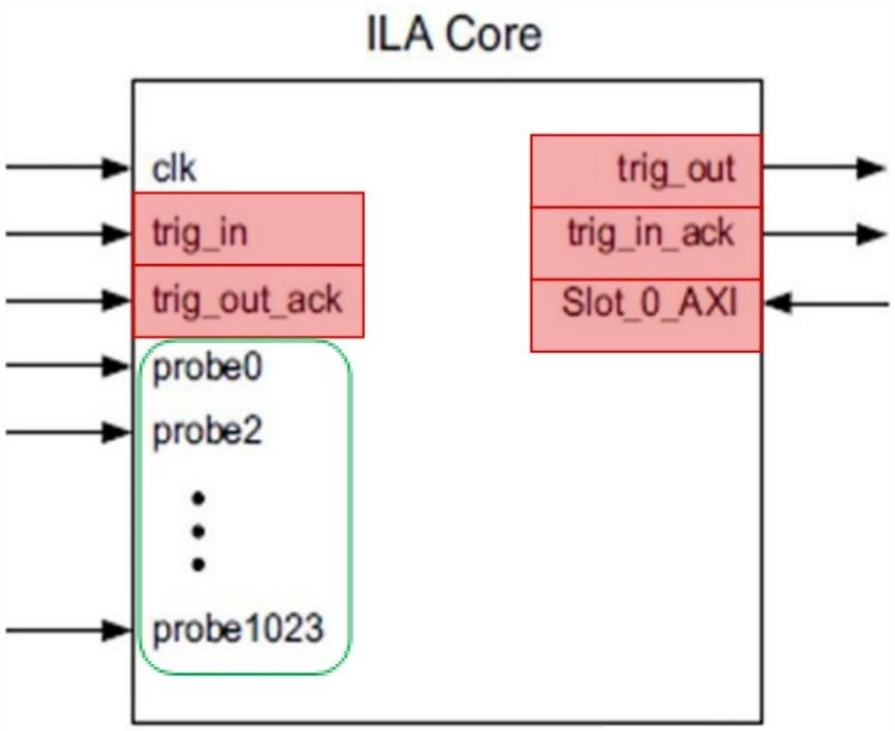
Vivado Logic Analyzer

- 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

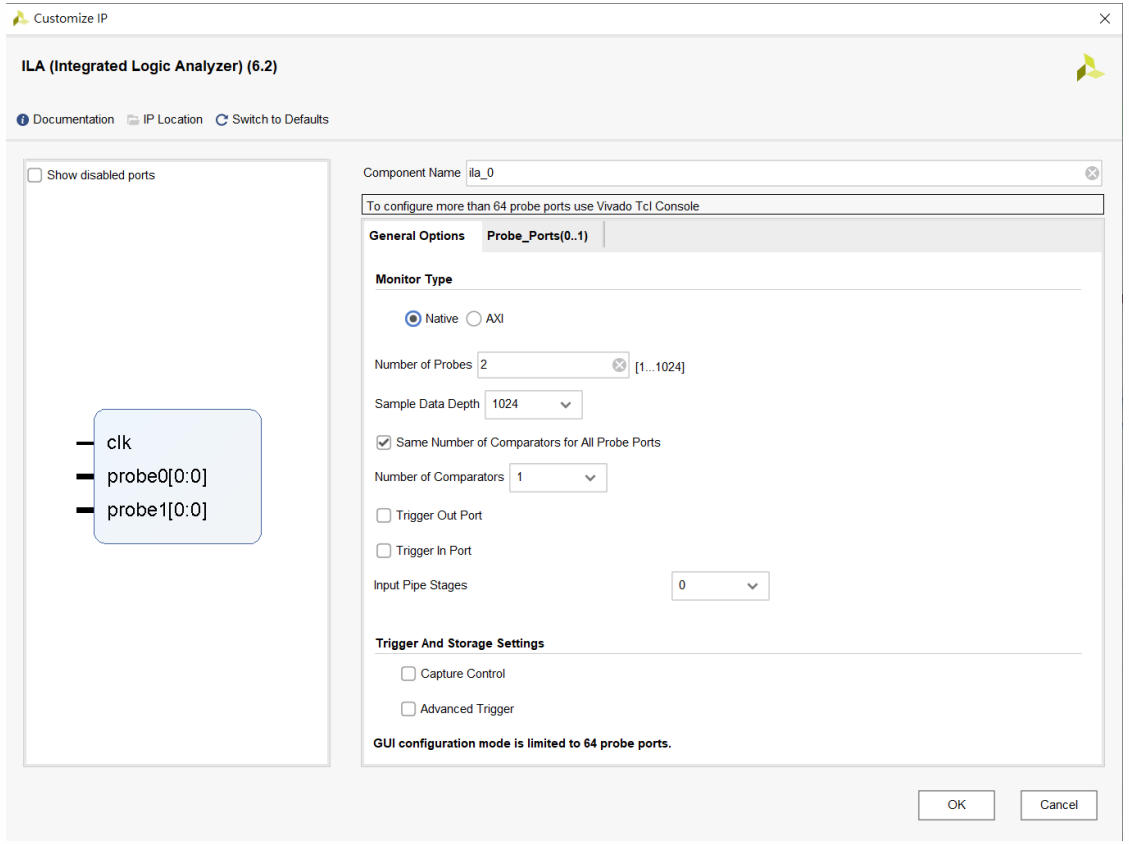
- Integrated Logic Analyzer (ILA)
  1. Monitor the waveform of signals which you want to debug
  2. Monitor type can be set two interfaces: Native and AXI, Communication is JTAG

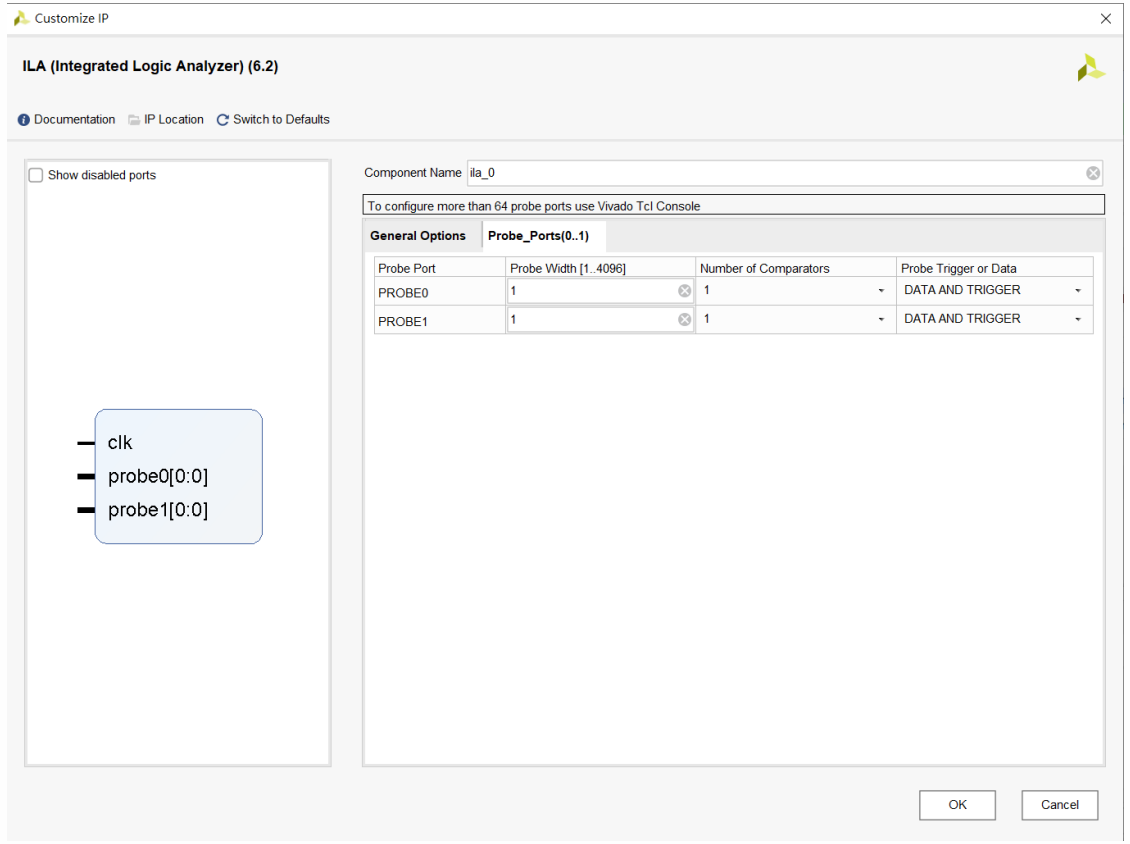


trig_in/out	Can be cascaded by connecting other ILAs
trig_in/out_ack	This will show the trig_in/out that has been already received normally to other ILAs.
Slot_0_AXI	This will connect to AXI interface

Vivado Logic Analyzer

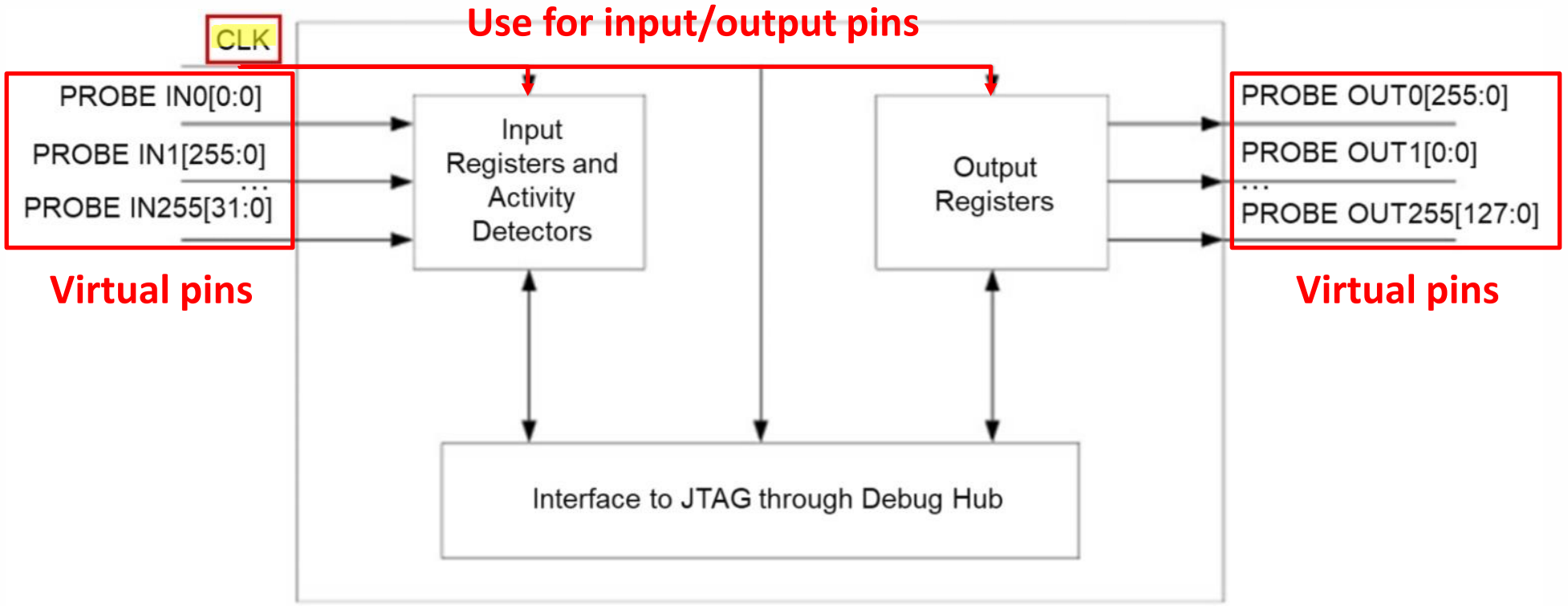
- Integrated Logic Analyzer (ILA)
  1. Monitor the waveform of signals which you want to debug
  2. Monitor type can be set two interfaces: Native and AXI, Communication is JTAG





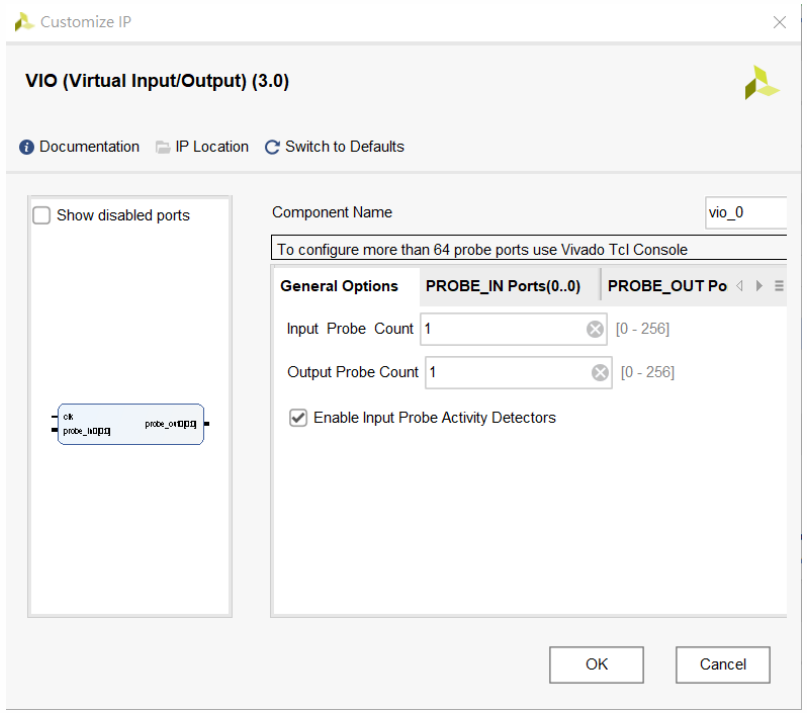
Vivado Logic Analyzer

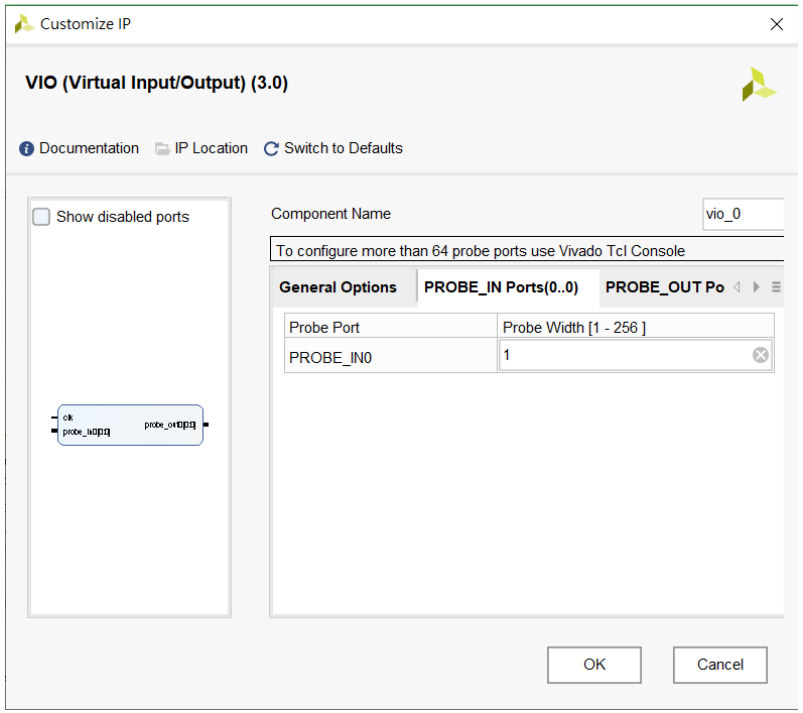
- Virtual Input/Output (VIO)
  1. Can monitor and drive the internal signals of FPGA in real time
  2. Not as ILA, VIO doesn't need BRAM

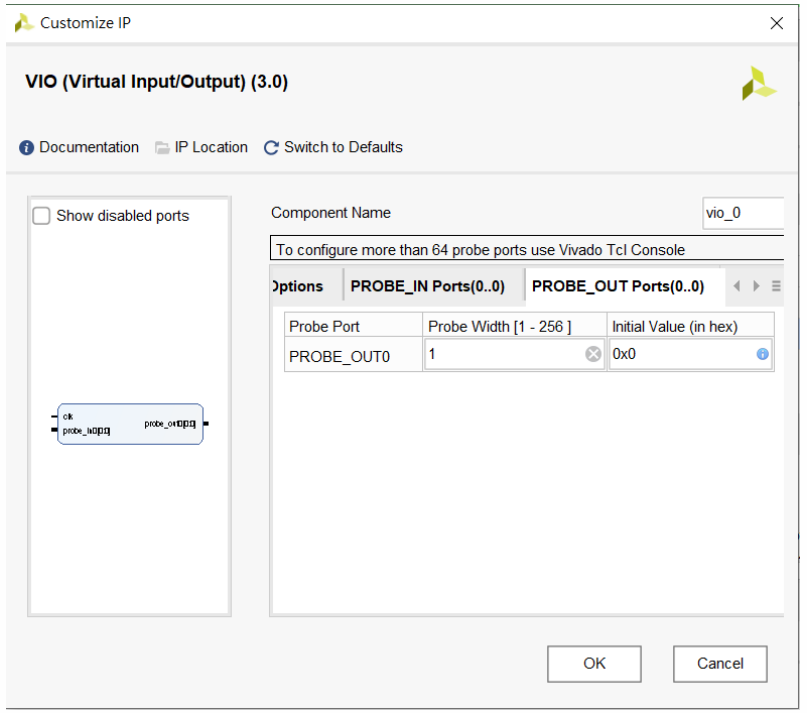


Vivado Logic Analyzer

- 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

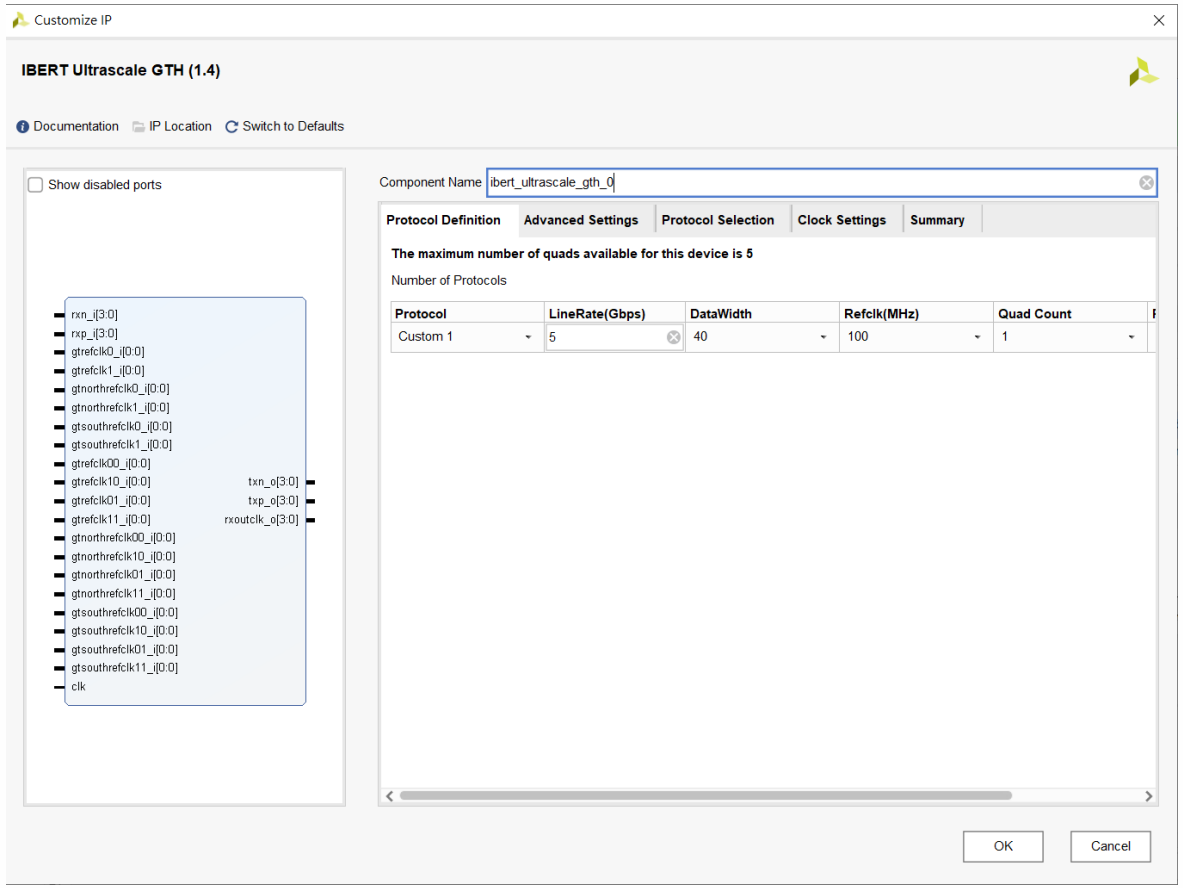






Vivado Logic Analyzer

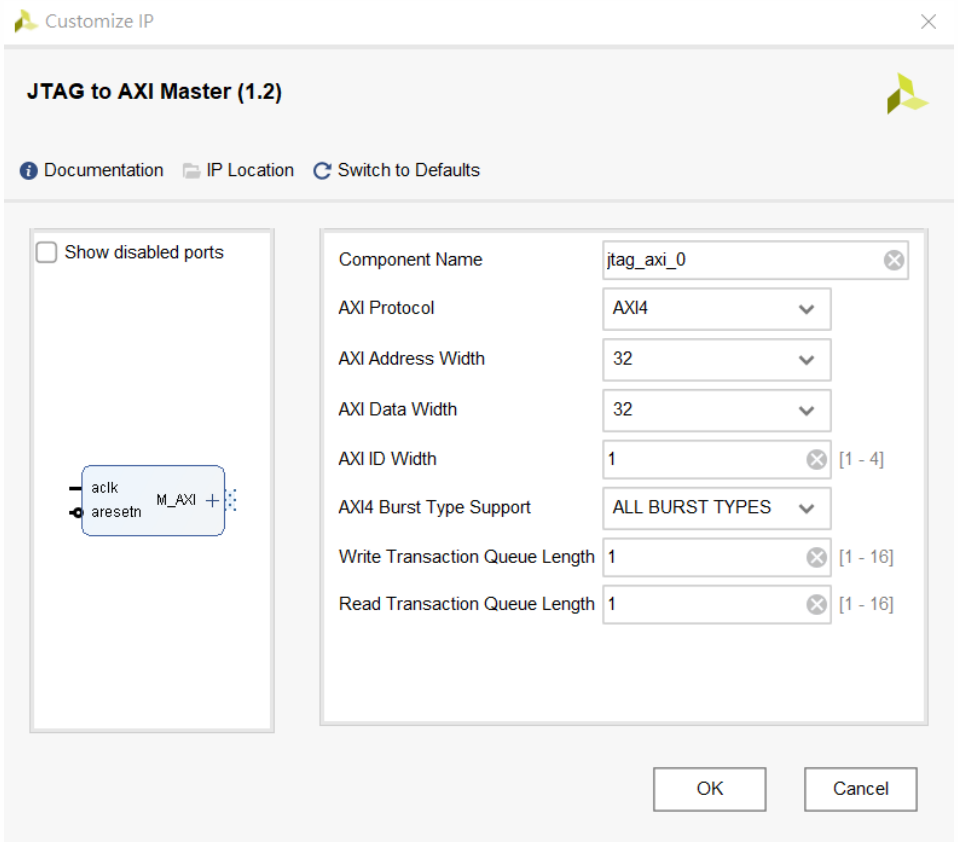
- Integrated Bit Error Ratio Tester (IBERT)
  1. Can test signals on transceiver
  2. Can analyze the bit-error ratio on MGT/GTP/GTX/GTH channels



XILINX.

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

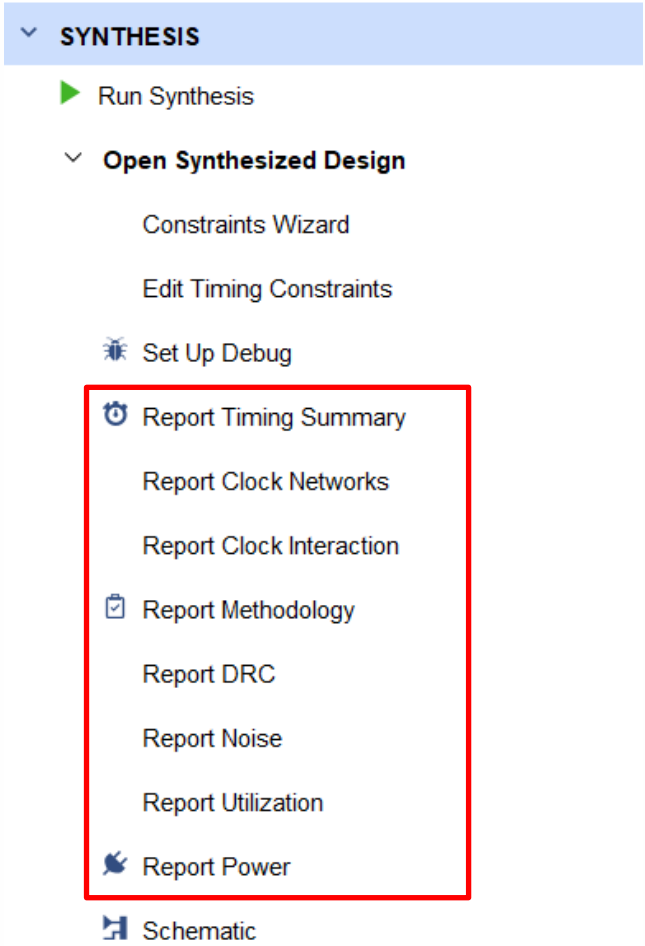


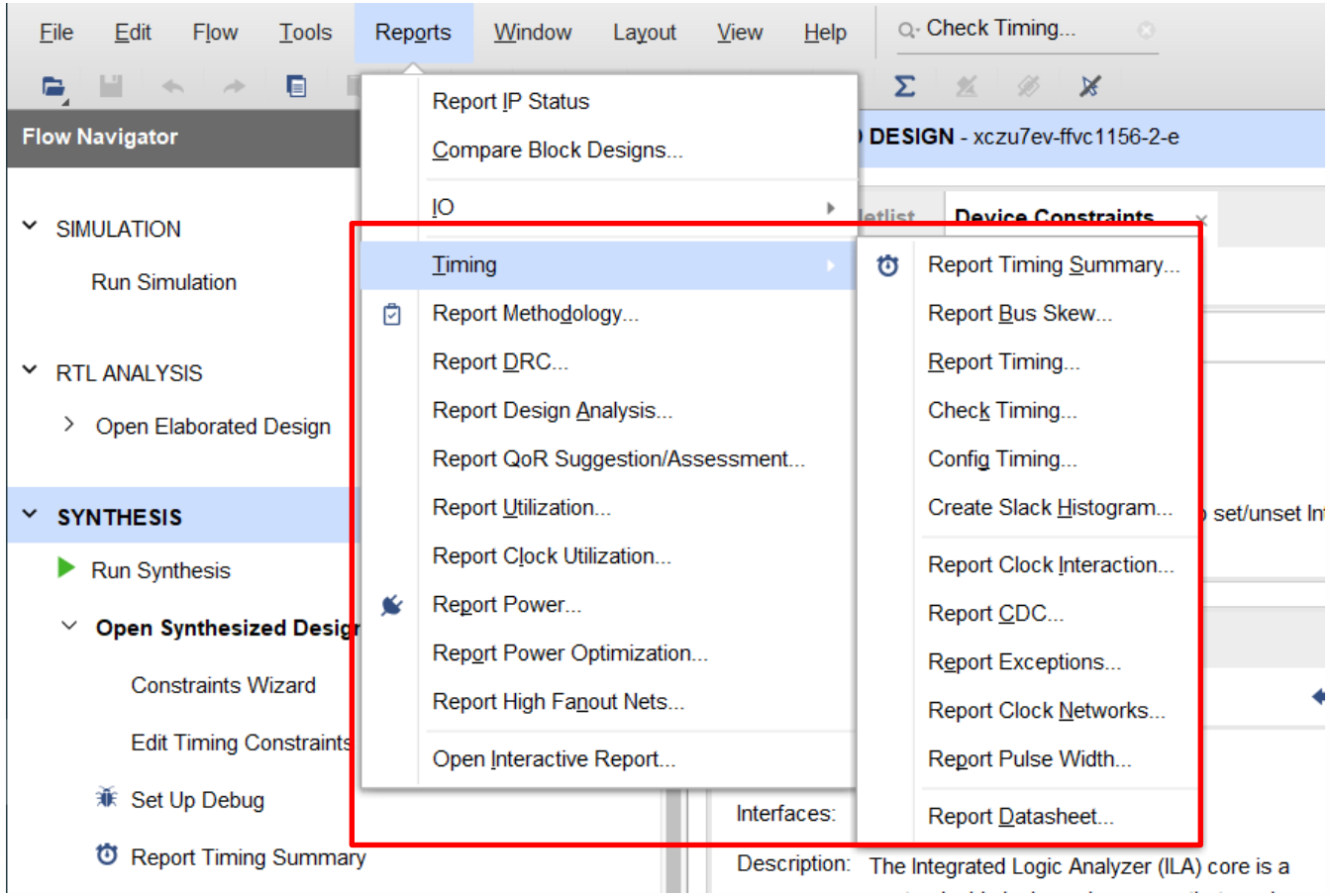
Vivado Reports

XILINX.

Vivado Reports

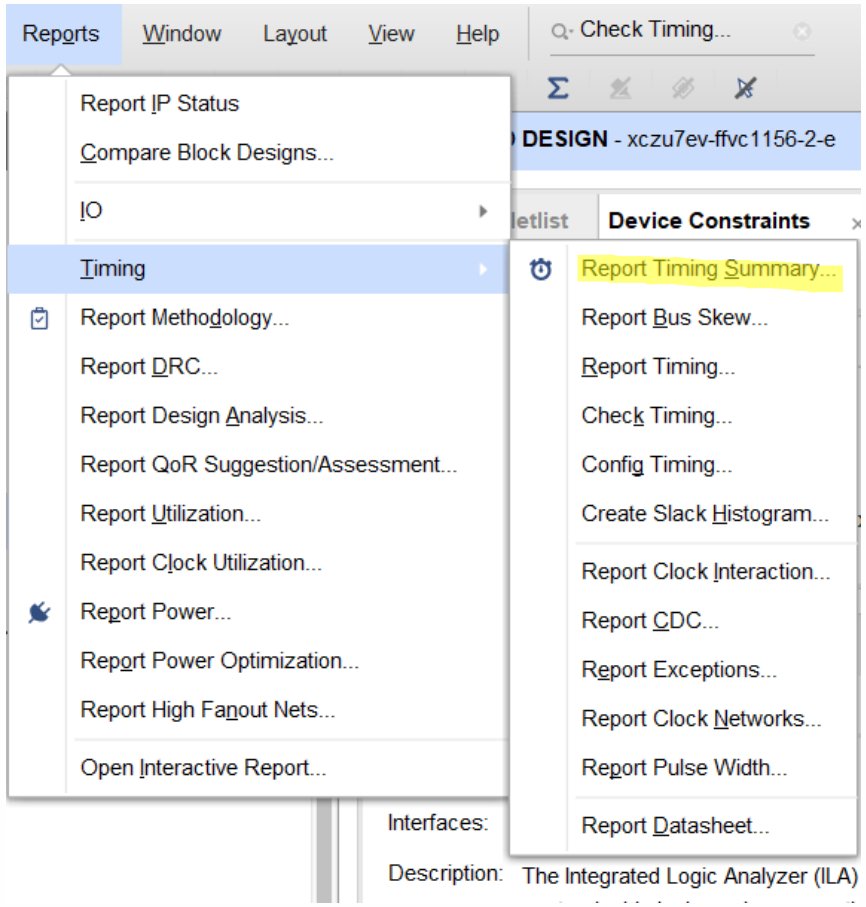
- Vivado can generate various reports at different stages (Synthesis, Implement,… etc.)



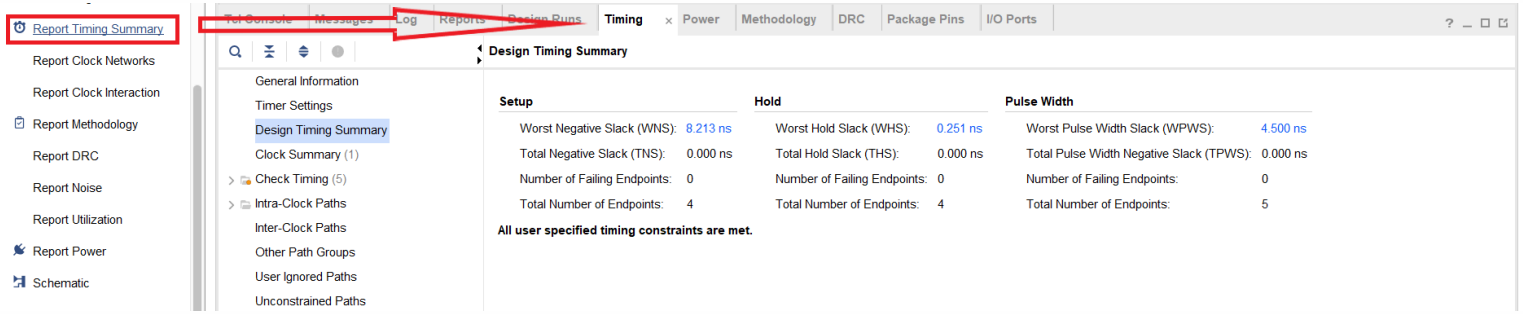


Vivado Reports

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

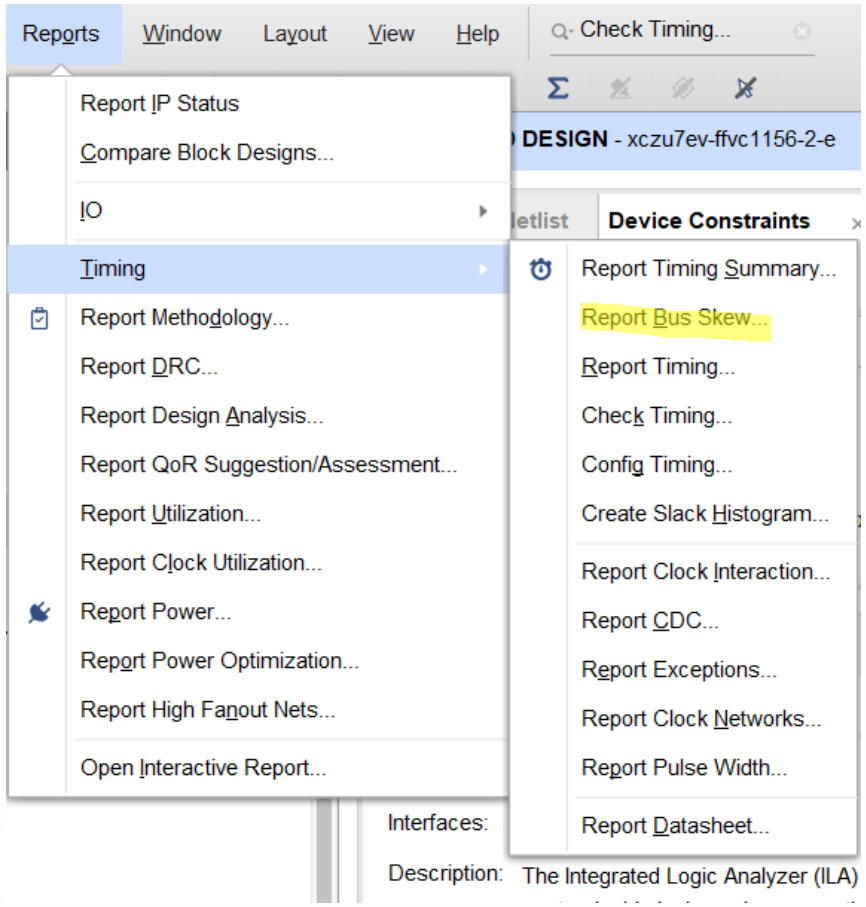


- Report Timing Summary
  - This will be generated automatically after synthesis or implementation.
  - Synthesis
    - Calculate net delays, according to connectivity and fanout
  - Implementation
    - According to the actual routing information

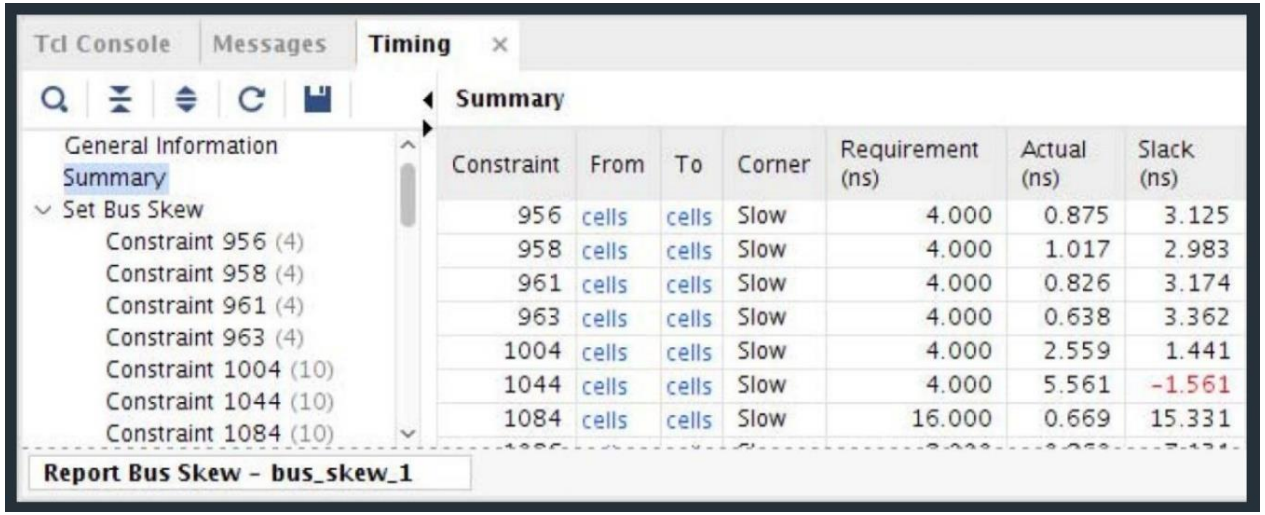


Vivado Reports

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

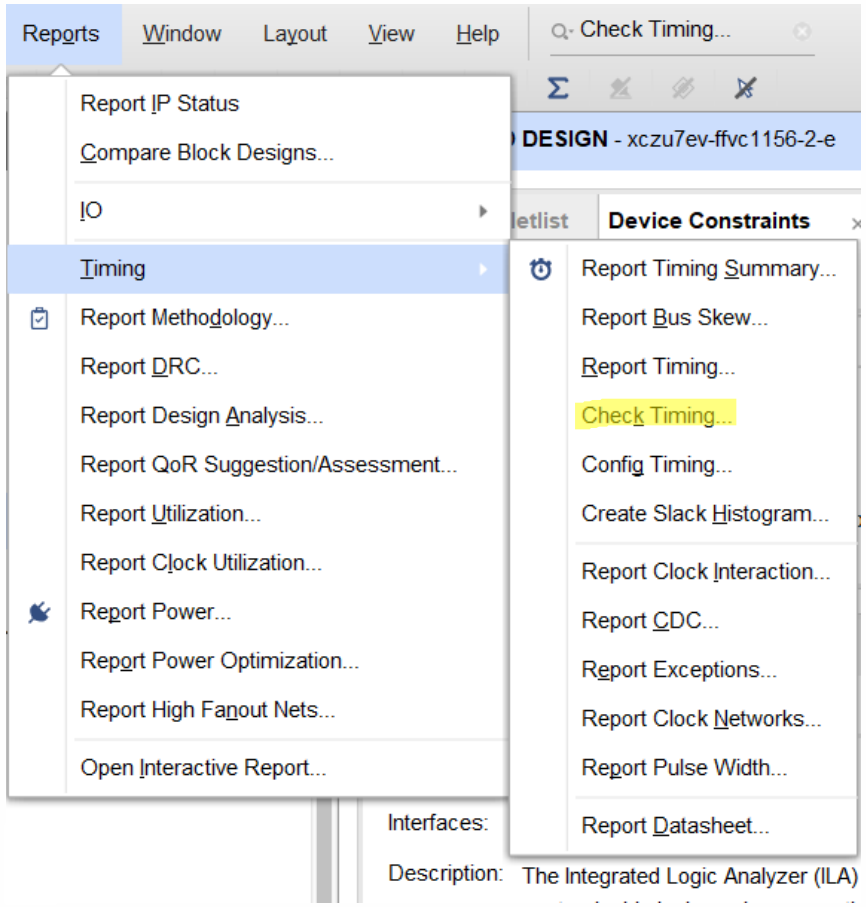


- Report Bus Skew
  - Skew means signals transport to every component will cause timing differences because the length of each bus is not the same in circuit.



Vivado Reports

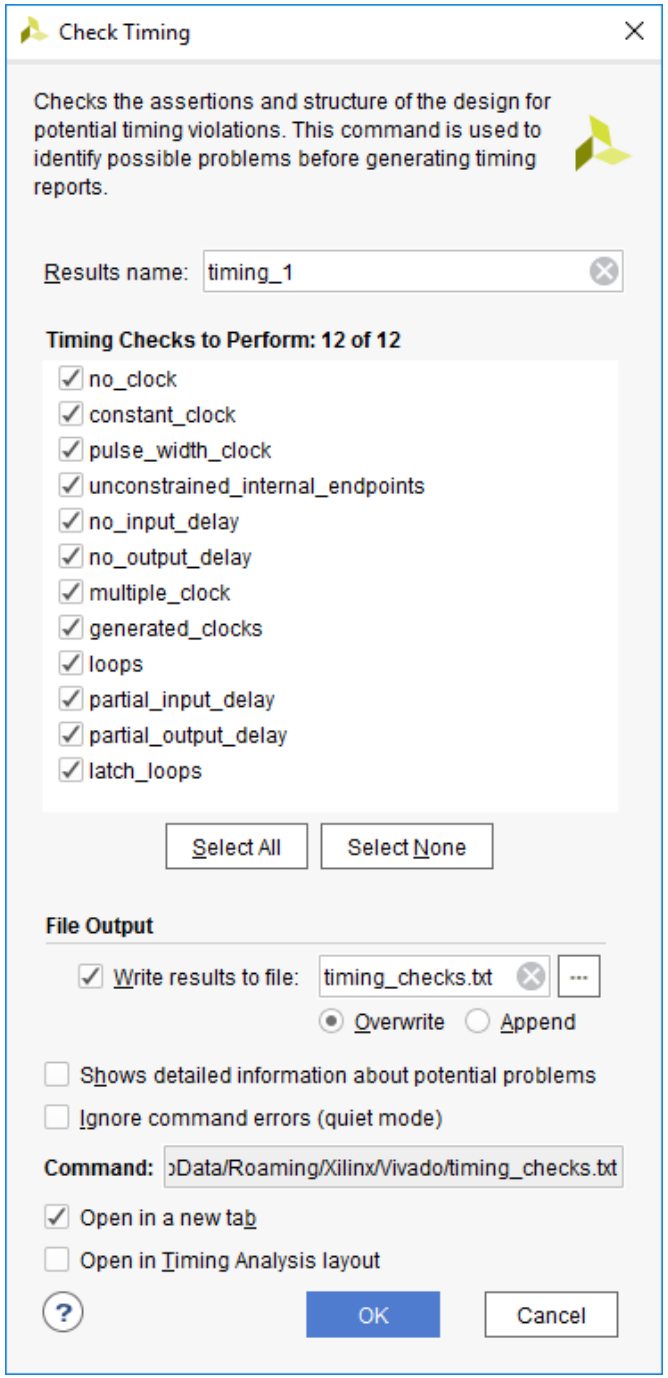
- Vivado can generate various reports at different stages (Synthesis, Implement,… etc.)



- 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

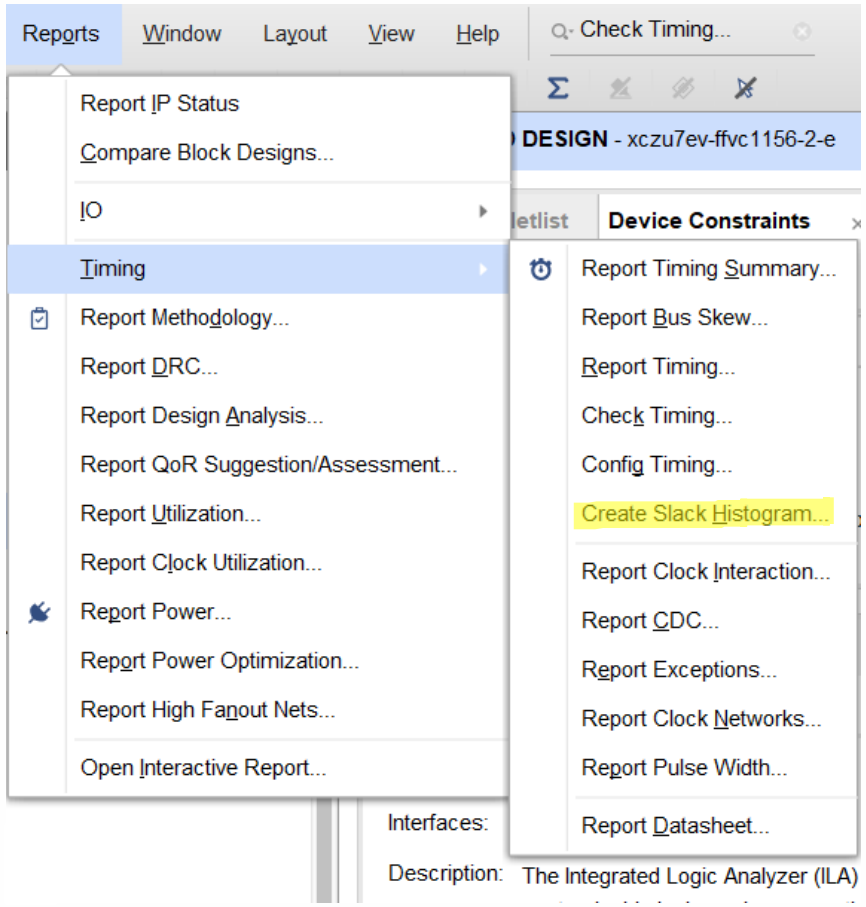
Vivado Reports

- Check Timing



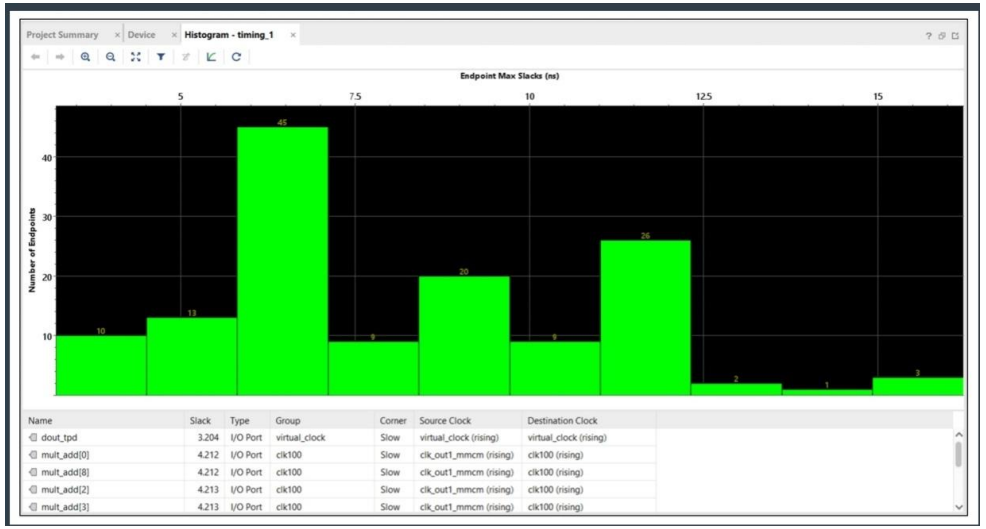
Vivado Reports

- 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

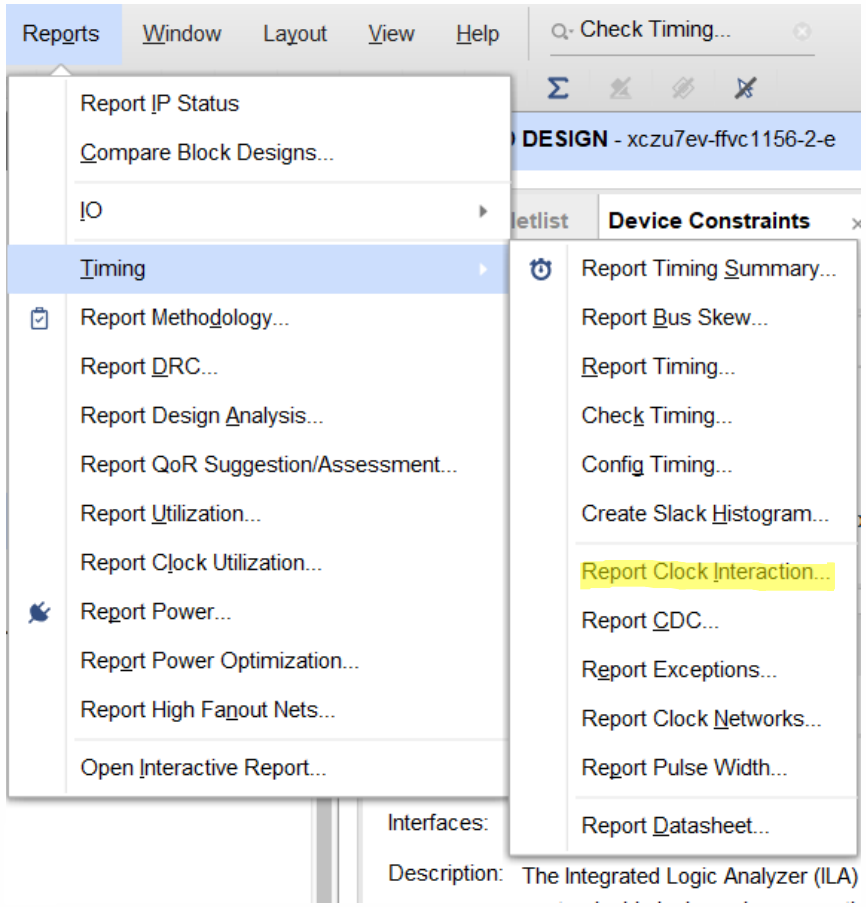
     Negative, means not match



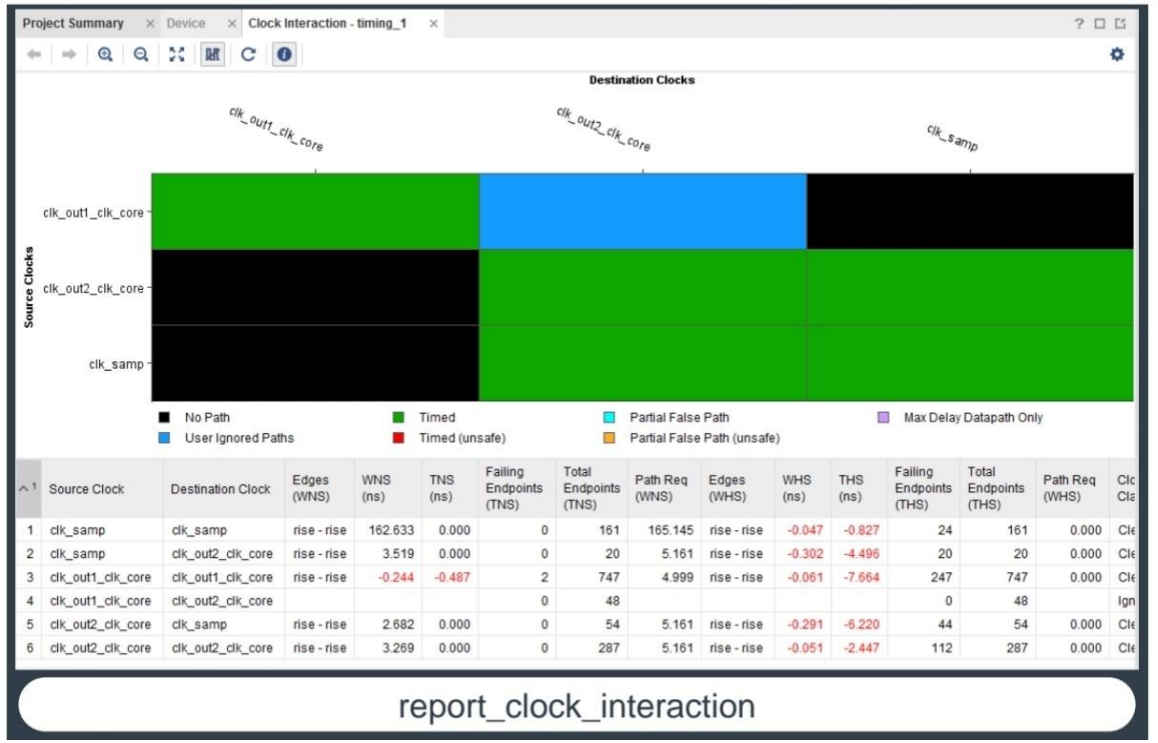
XILINX.

Vivado Reports

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

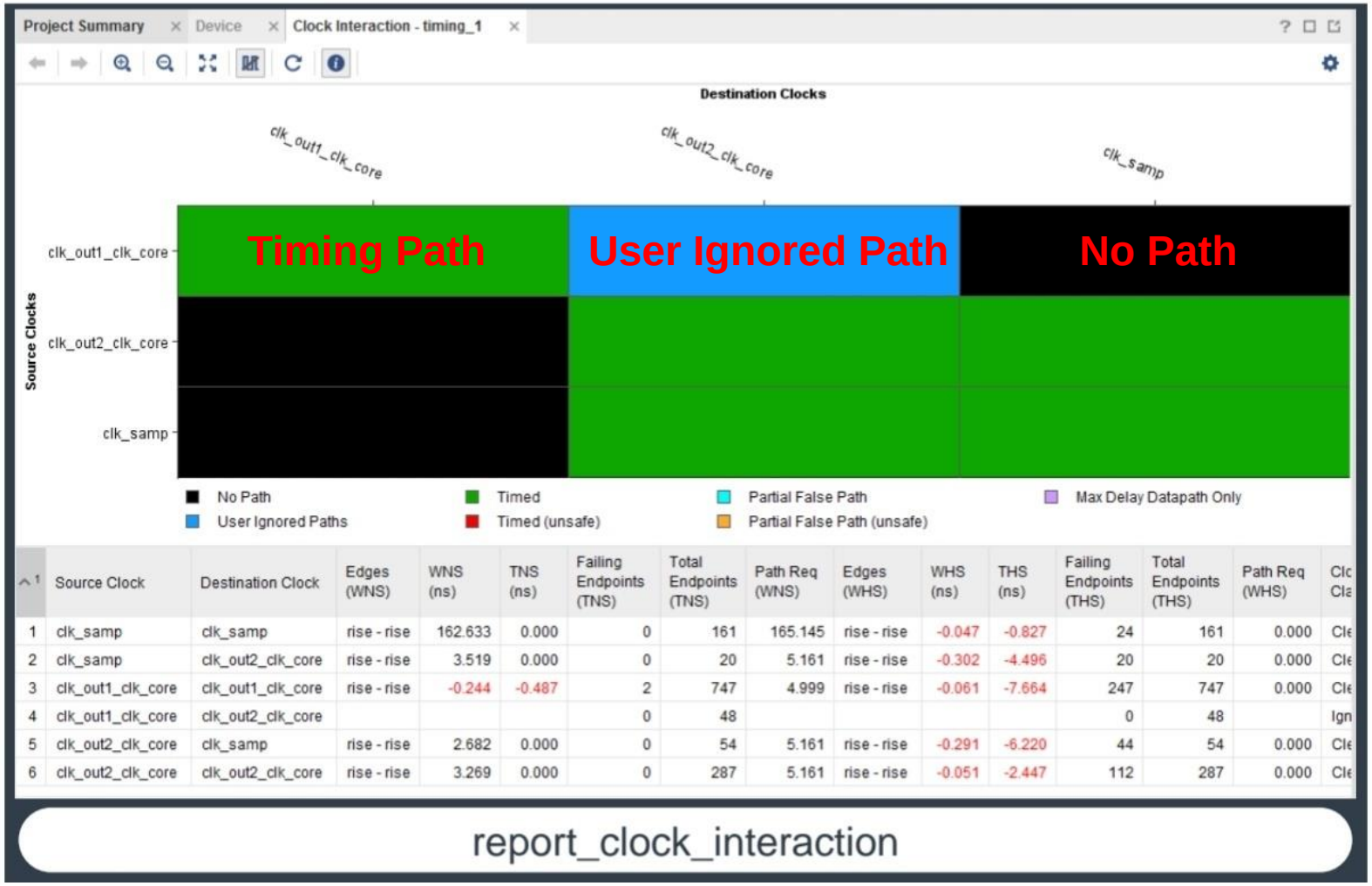


- Clock Interaction
  - Analyze the situation of data loss or Metastability from a clock domain to next or source to destination



Vivado Reports

- Clock Interaction



	Source Clock	Destination Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoints (TNS)	Total Endpoints (TNS)	Path Req (WNS)	Edges (WHS)	WHS (ns)	THS (ns)	Failing Endpoints (THS)	Total Endpoints (THS)	Path Req (WHS)	Clc Cla
1	clk_samp	clk_samp	rise - rise	162.633	0.000	0	161	165.145	rise - rise	-0.047	-0.827	24	161	0.000	Cle
2	clk_samp	clk_out2_clk_core	rise - rise	3.519	0.000	0	20	5.161	rise - rise	-0.302	-4.496	20	20	0.000	Cle
3	clk_out1_clk_core	clk_out1_clk_core	rise - rise	-0.244	-0.487	2	747	4.999	rise - rise	-0.061	-7.664	247	747	0.000	Cle
4	clk_out1_clk_core	clk_out2_clk_core				0	48					0	48		Ign
5	clk_out2_clk_core	clk_samp	rise - rise	2.682	0.000	0	54	5.161	rise - rise	-0.291	-6.220	44	54	0.000	Cle
6	clk_out2_clk_core	clk_out2_clk_core	rise - rise	3.269	0.000	0	287	5.161	rise - rise	-0.051	-2.447	112	287	0.000	Cle

report_clock_interaction

Partial False Path

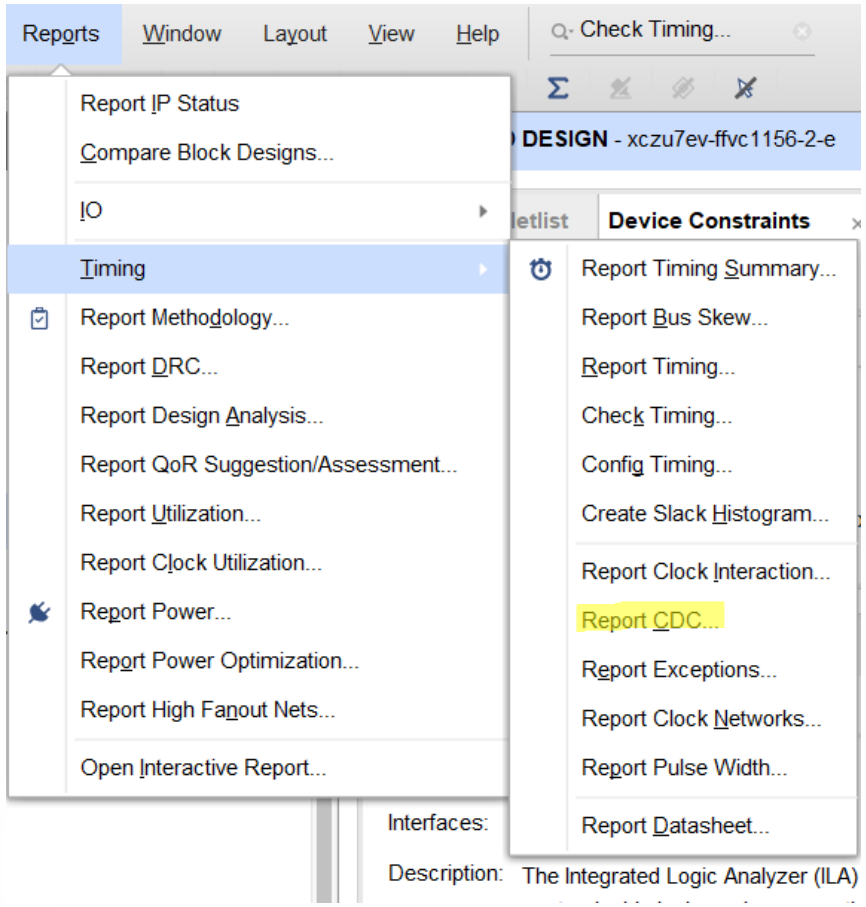
Unsafe Time

Partial False Path (Unsafe)

Max Delay Datapath Only

Vivado Reports

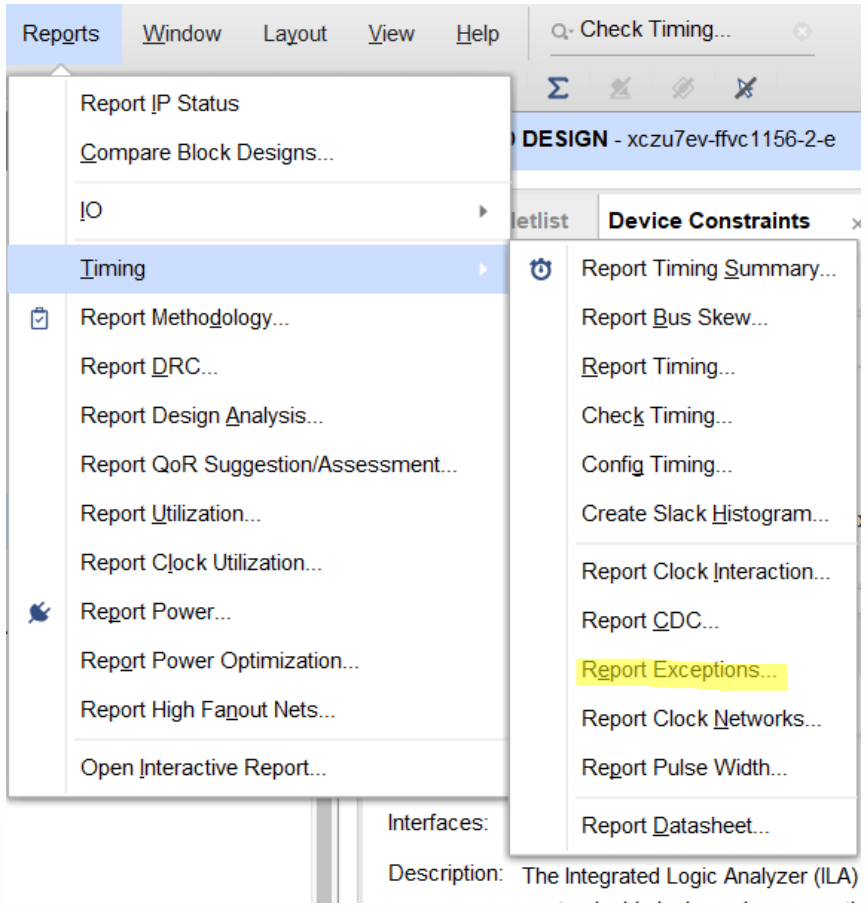
- Vivado can generate various reports at different stages (Synthesis, Implement,… etc.)



- 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

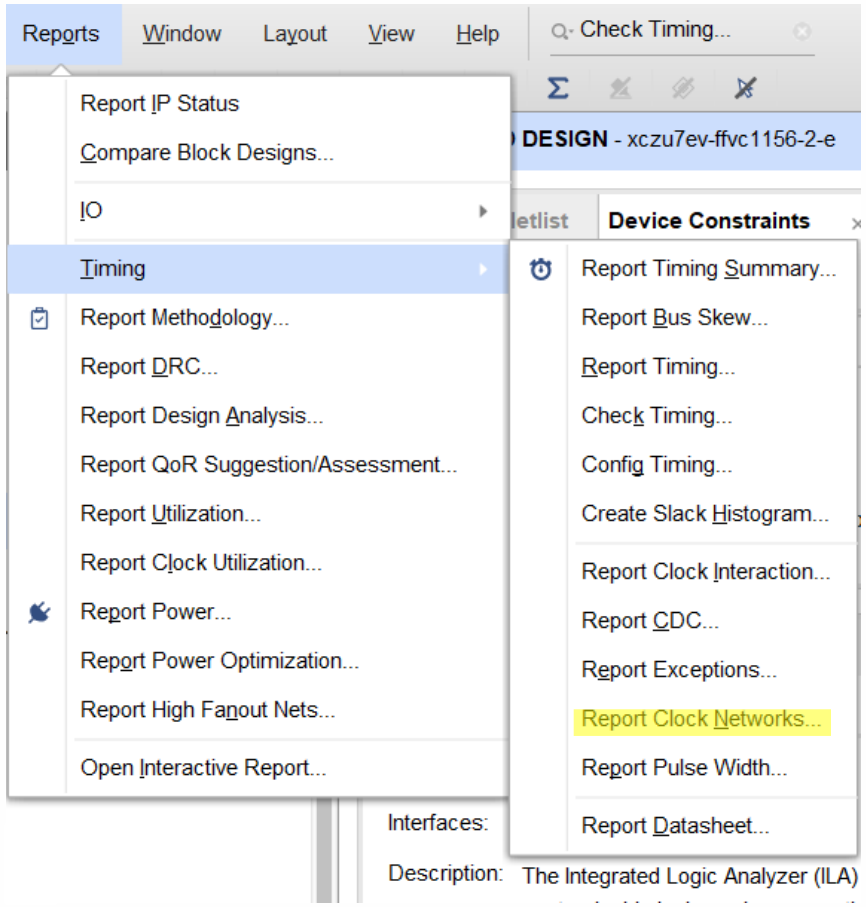
- Vivado can generate various reports at different stages (Synthesis, Implement,… etc.)



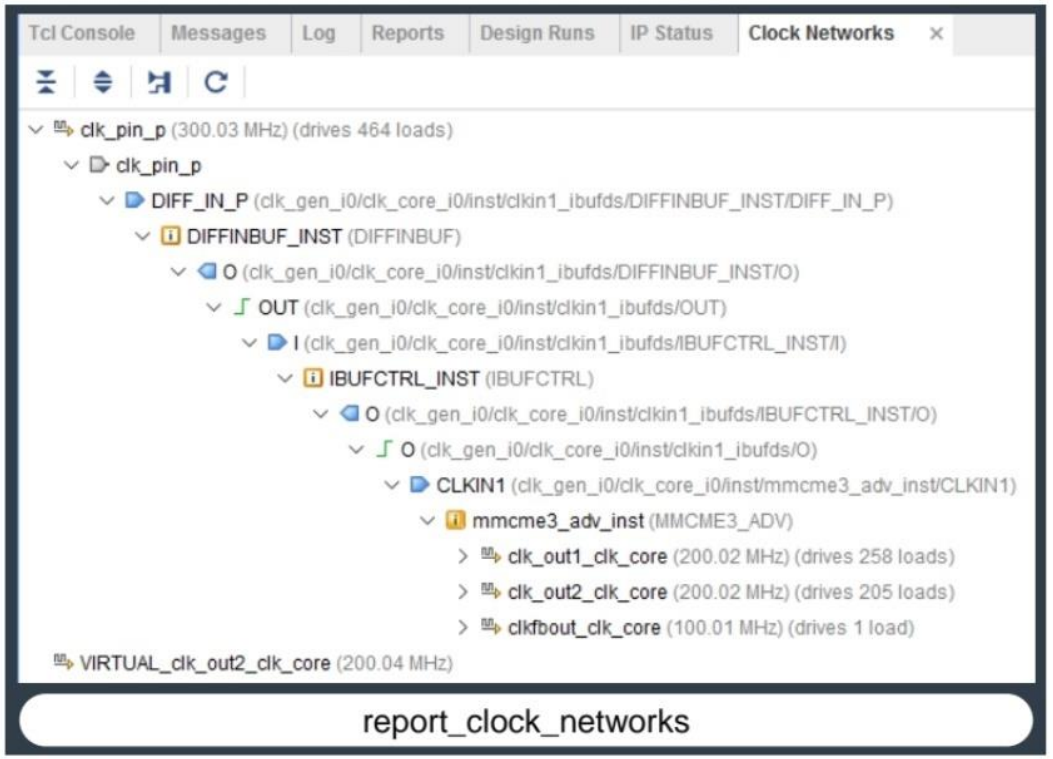
- Report Exceptions
  - Report the parts of Timing Exceptions
    1. set_multicycle_path
    2. set_false_path
    3. set_case_analysis

Vivado Reports

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

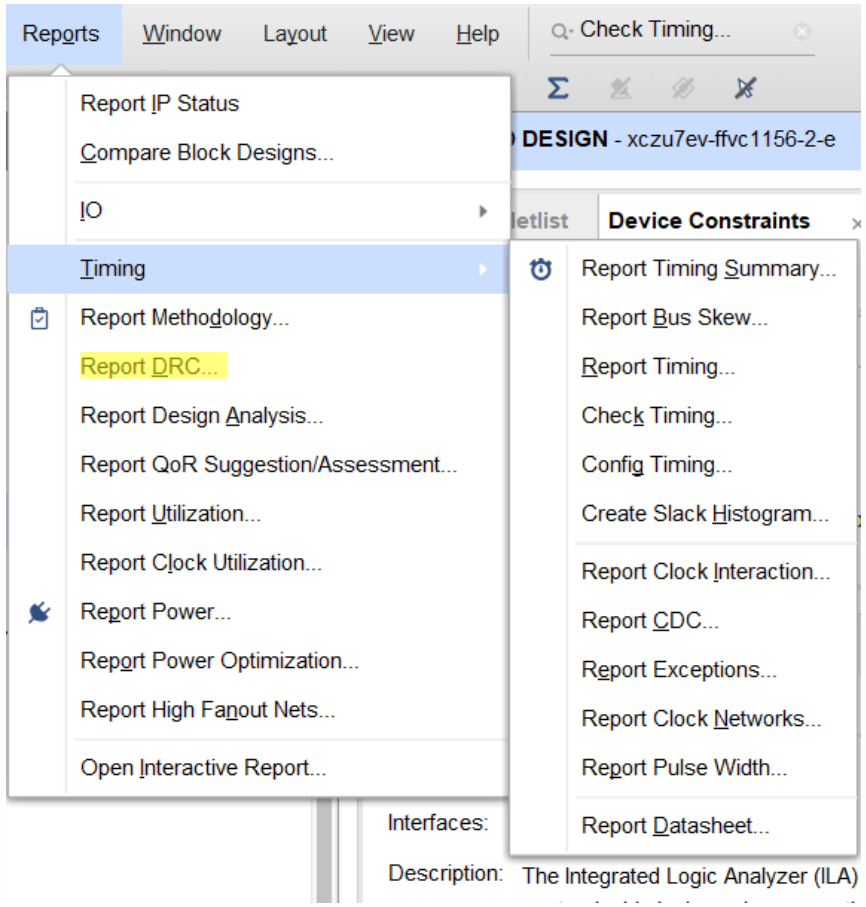


- Report Clock Networks
  - Offer a structure to describe the clock distribution from source to destination.

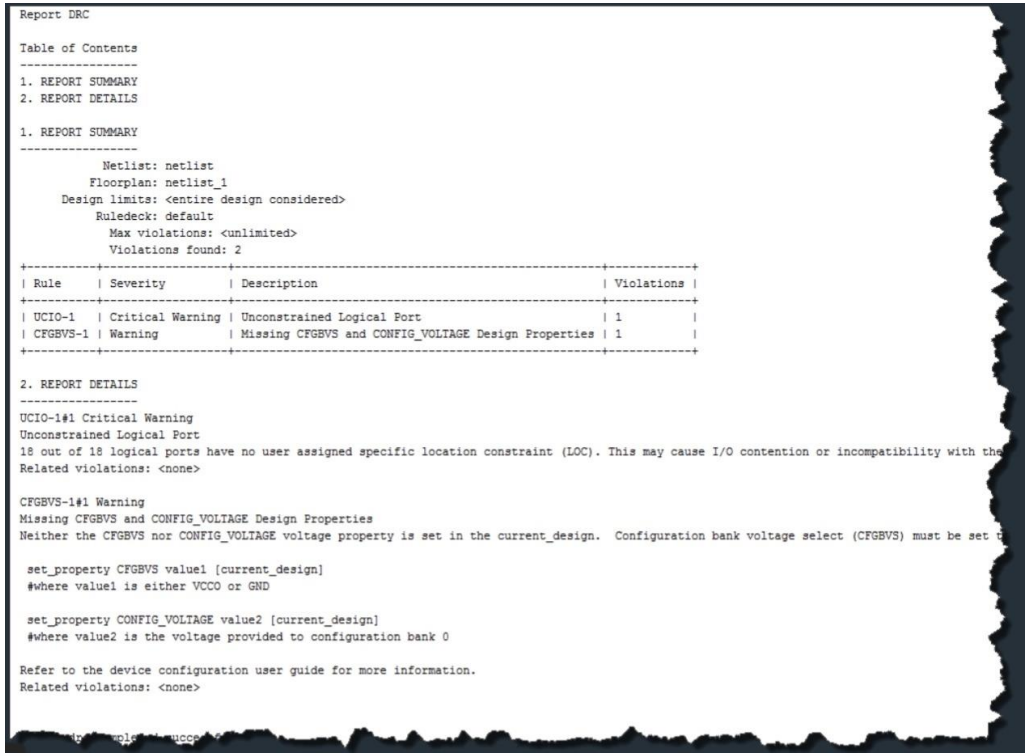


Vivado Reports

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



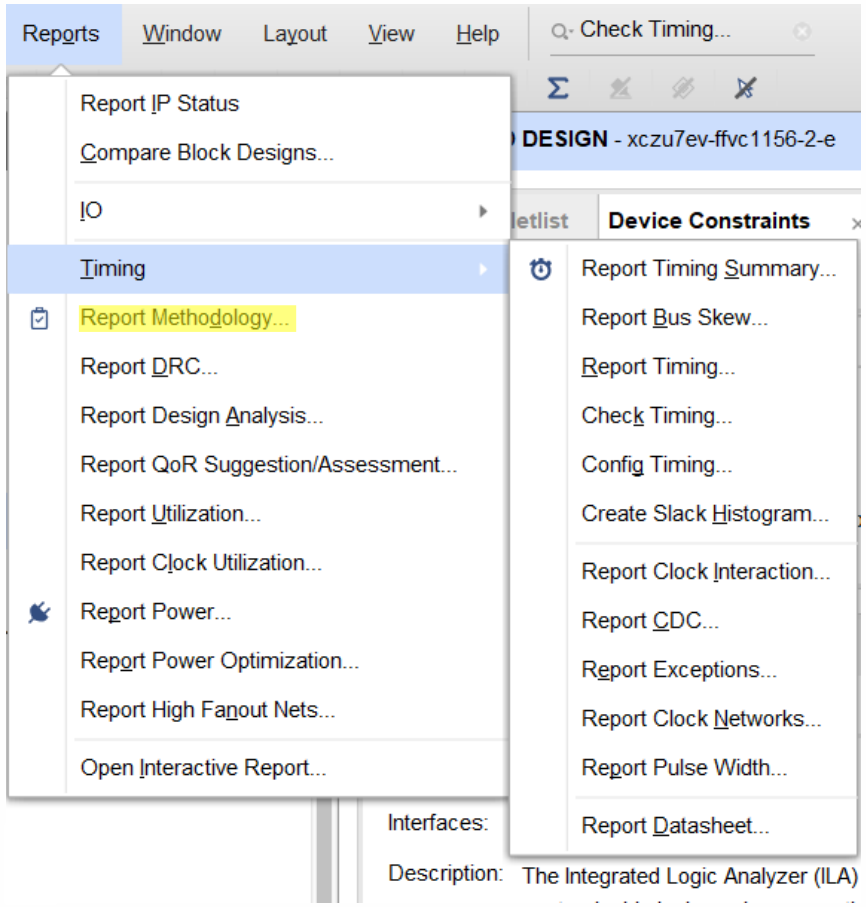
- Report Design Rule Check (DRC)
  - Offer any parts which violate the design rule
  - This will run automatically after Synthesis or Implementation



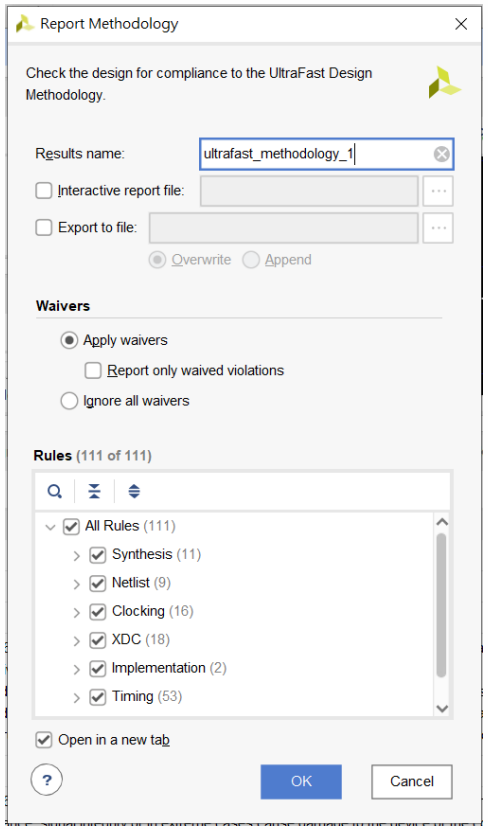
```
Report DRC

Table of Contents
-----------------
1. REPORT SUMMARY
2. REPORT DETAILS

1. REPORT SUMMARY
-----------------
           Netlist: netlist
         Floorplan: netlist_1
     Design limits: <entire design considered>
          Ruledeck: default
            Max violations: <unlimited>
            Violations found: 2
+----------+-----------------+-------------------------------------------------------+------------+
| Rule     | Severity        | Description                                           | Violations |
+----------+-----------------+-------------------------------------------------------+------------+
| UCIO-1   | Critical Warning | Unconstrained Logical Port                           | 1          |
| CFGBVS-1 | Warning          | Missing CFGBVS and CONFIG_VOLTAGE Design Properties  | 1          |
+----------+-----------------+-------------------------------------------------------+------------+

2. REPORT DETAILS
-----------------
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 t

 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, Implement,… etc.)

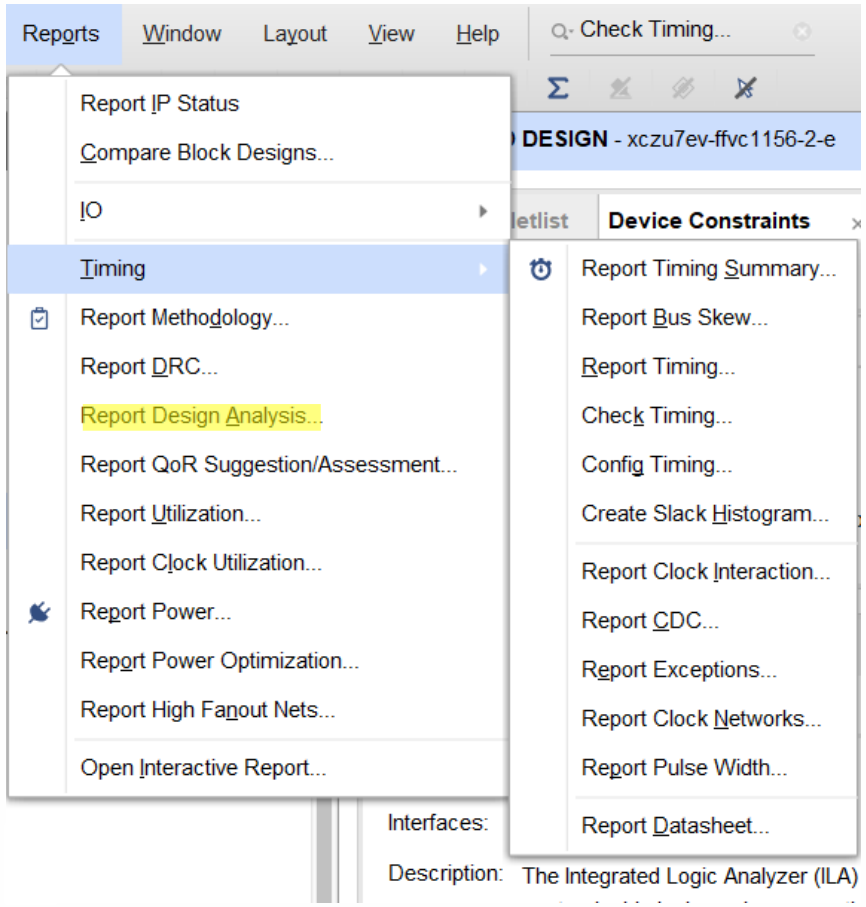


- Report Methodology
  - Major important DRC sections
  - This will run automatically after Synthesis or Implementation

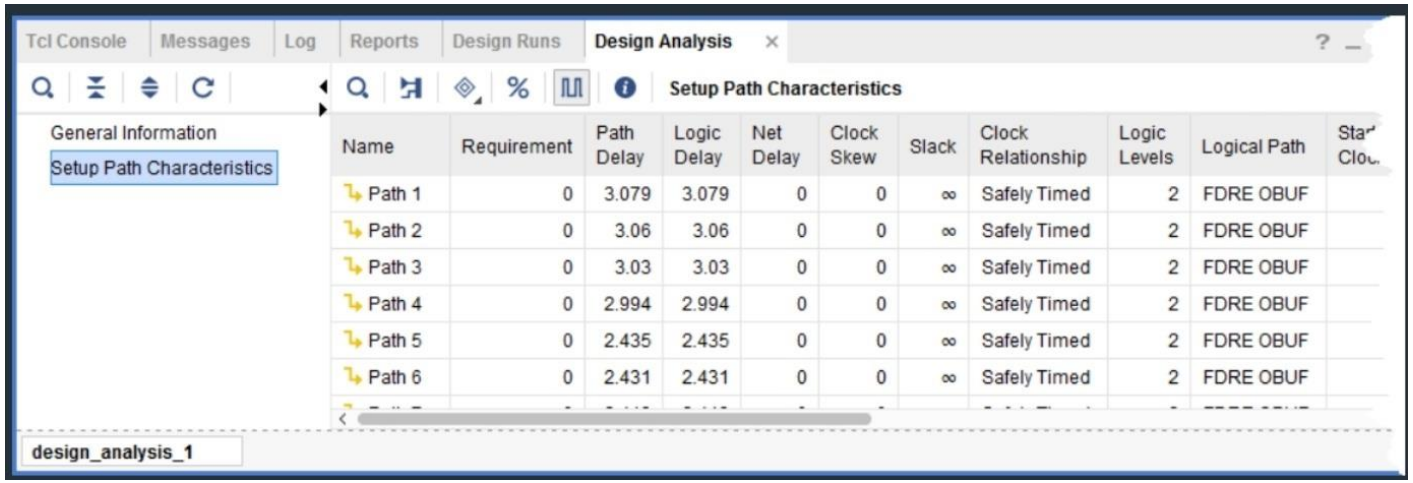


Vivado Reports

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



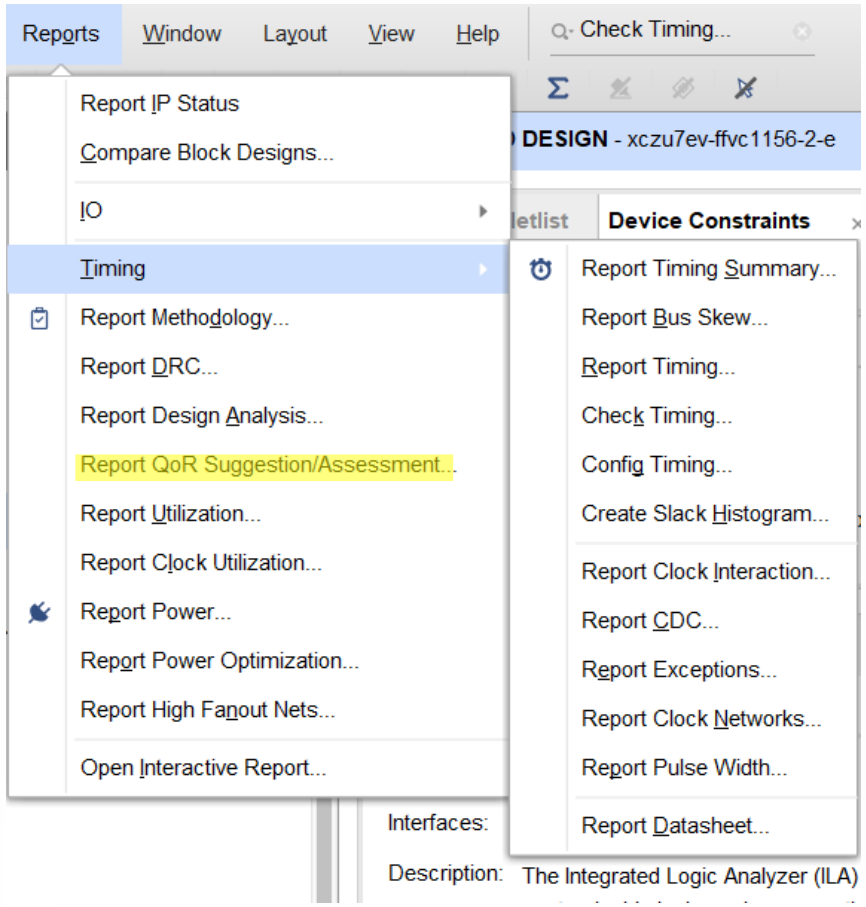
- Report Design Analysis
  - Offer the information of Timing Path Characteristics, Design Interconnect Complexity, Congestion,… etc.



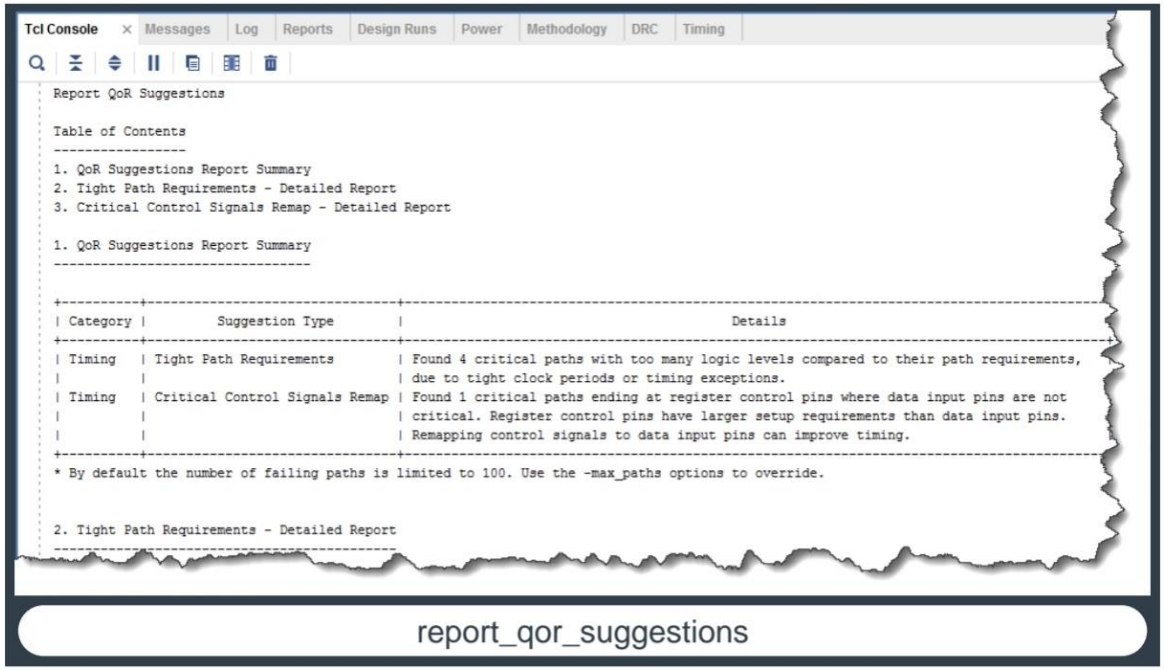
Name	Requirement	Path Delay	Logic Delay	Net Delay	Clock Skew	Slack	Clock Relationship	Logic Levels	Logical Path
Path 1	0	3.079	3.079	0	0	∞	Safely Timed	2	FDRE OBUF
Path 2	0	3.06	3.06	0	0	∞	Safely Timed	2	FDRE OBUF
Path 3	0	3.03	3.03	0	0	∞	Safely Timed	2	FDRE OBUF
Path 4	0	2.994	2.994	0	0	∞	Safely Timed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

Vivado Reports

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

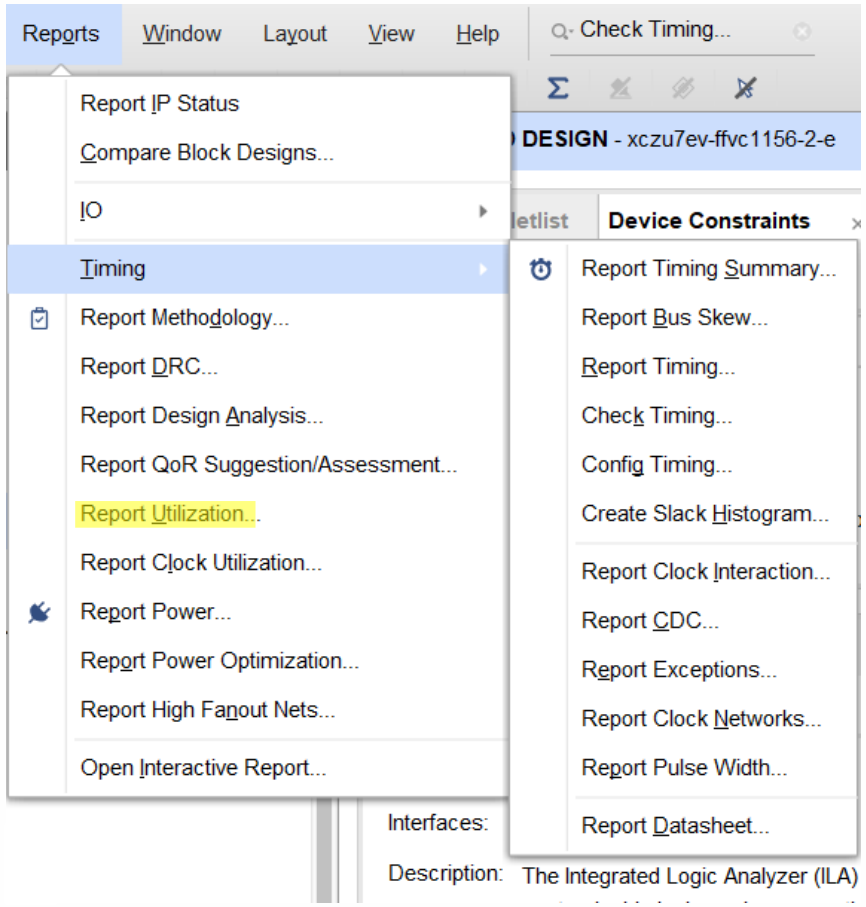


- Report QoR Suggestions (RQS)
  - This will offer the suggestion of optimization through analyzing netlist characteristics, timing violations and congestion information.
  - This needs to be run after Synthesis or Implementation

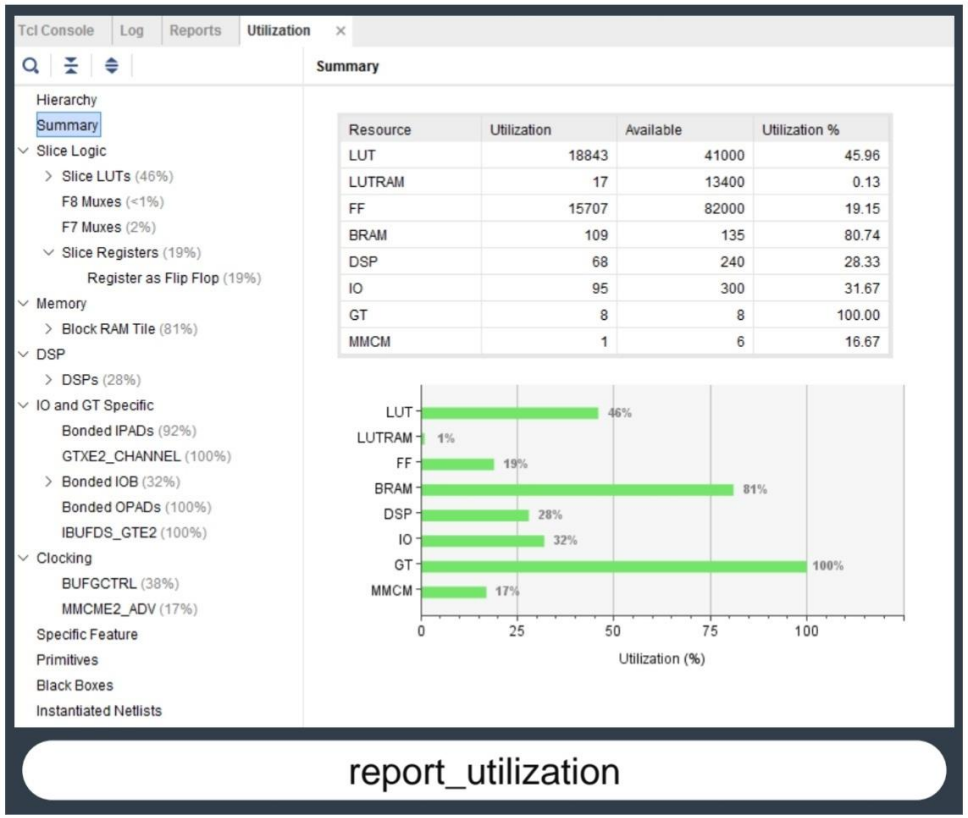


Vivado Reports

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

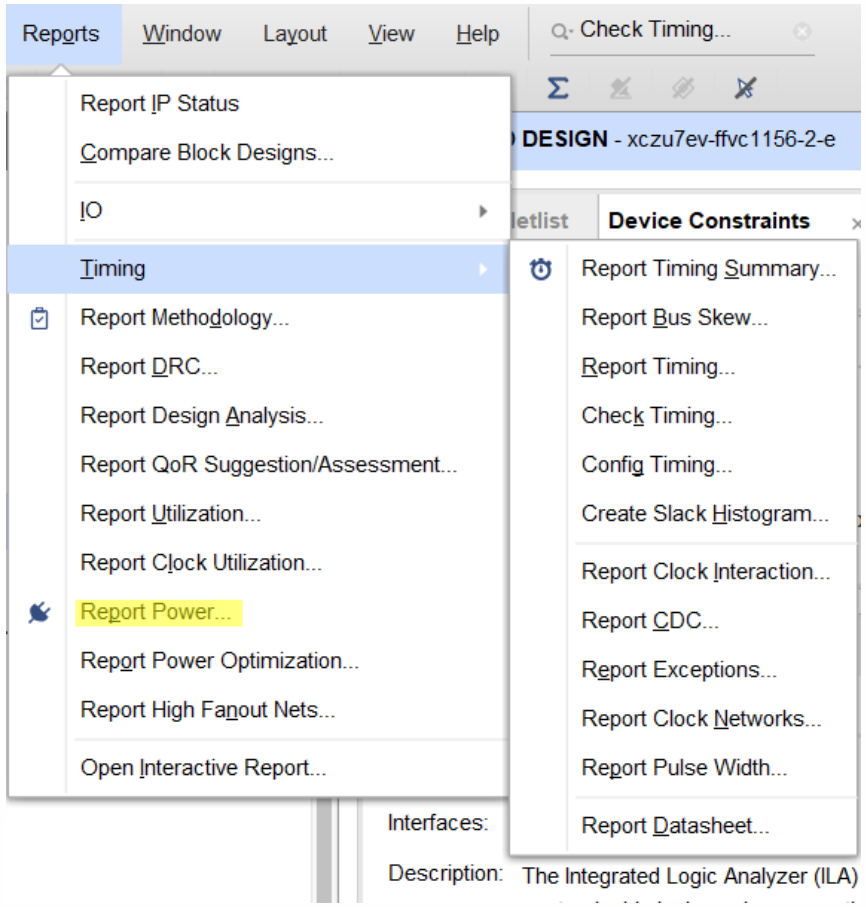


- Report Utilization
  - Offer the ratio of resource distribution which including LUTs, Registers, FIFO, I/O,… etc., after Synthesis or Implementation.

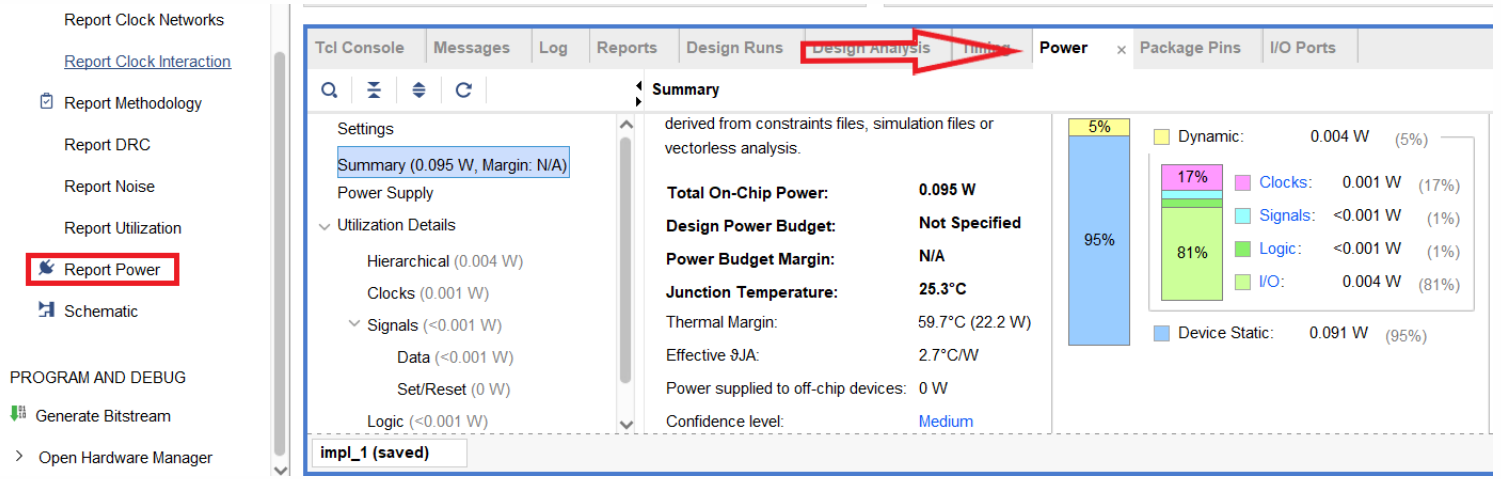


Vivado Reports

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

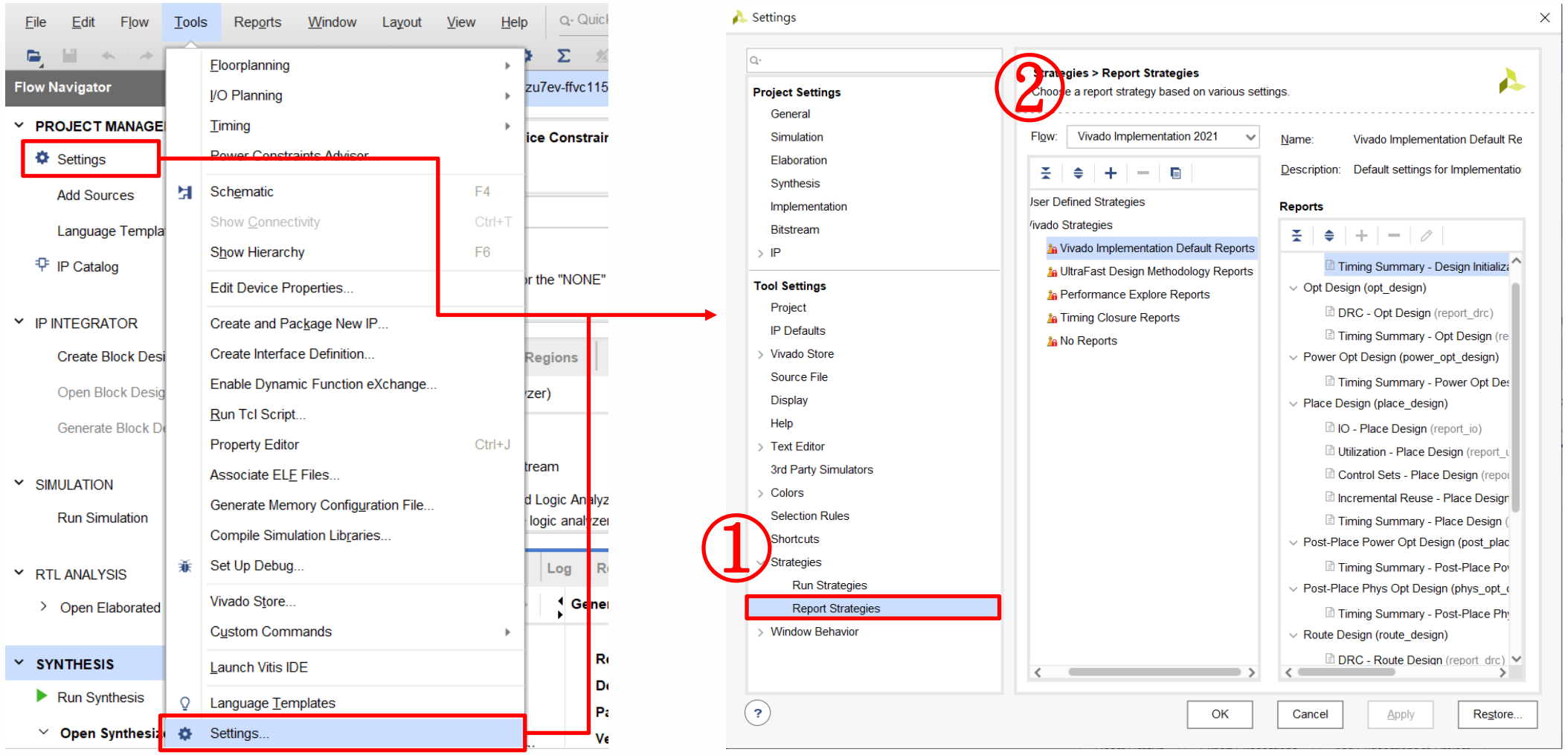


- Report Power
  - Power consumption is calculated based on the junction and the ambient temperature.
  - Generally used after routing stage
  - Allows detailed power distribution analysis to guide power saving strategies and to reduce dynamic, thermal, or off-chip power.



Vivado Report Strategies

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



Vivado Configuration

XILINX.

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
CFGBVS	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

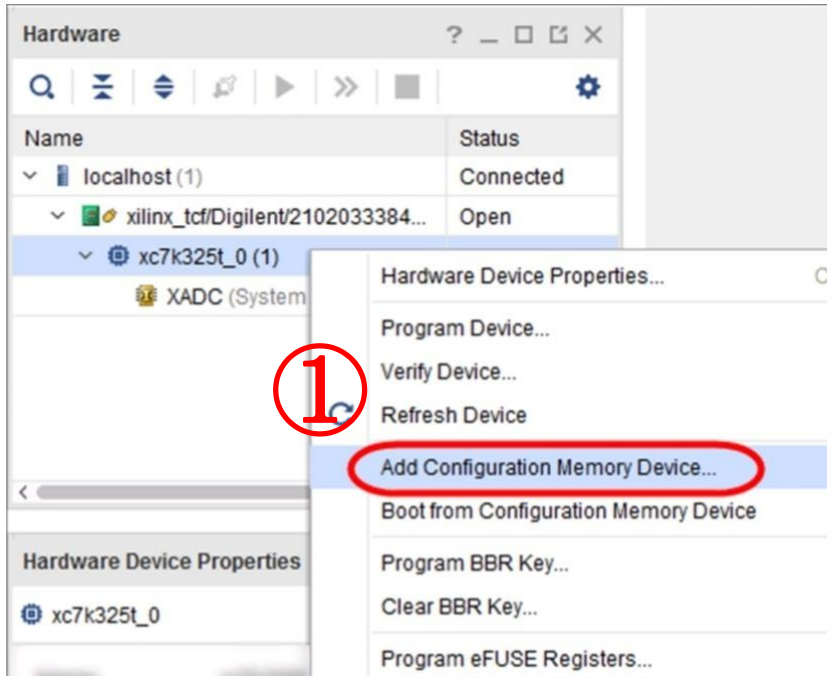
Vivado Configuration

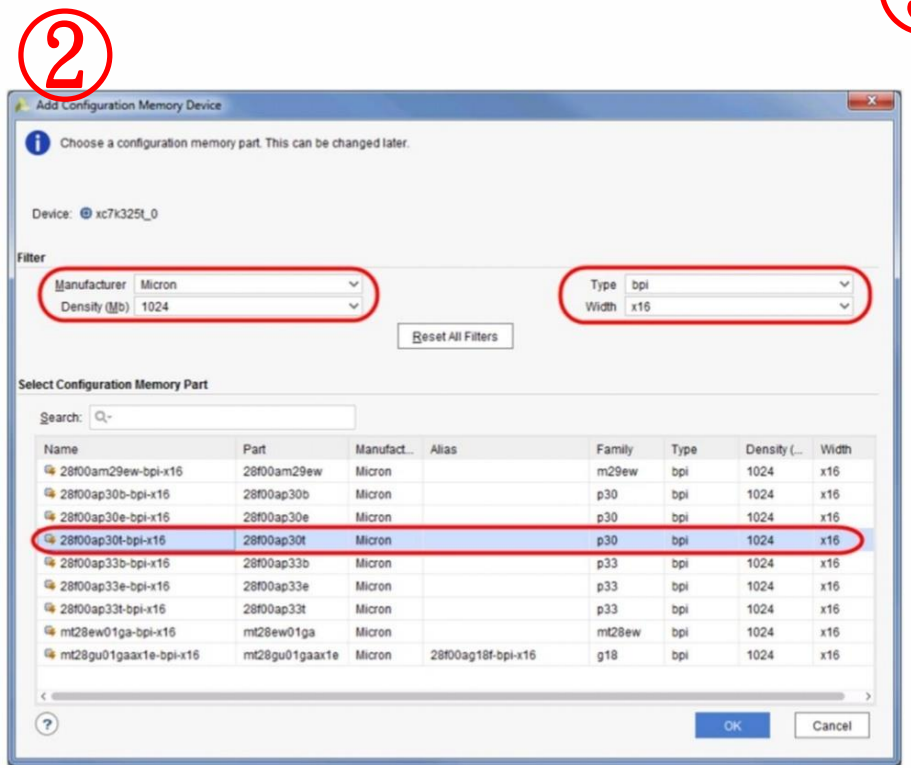
- FPGA needs some special pins when loading the bitstream file

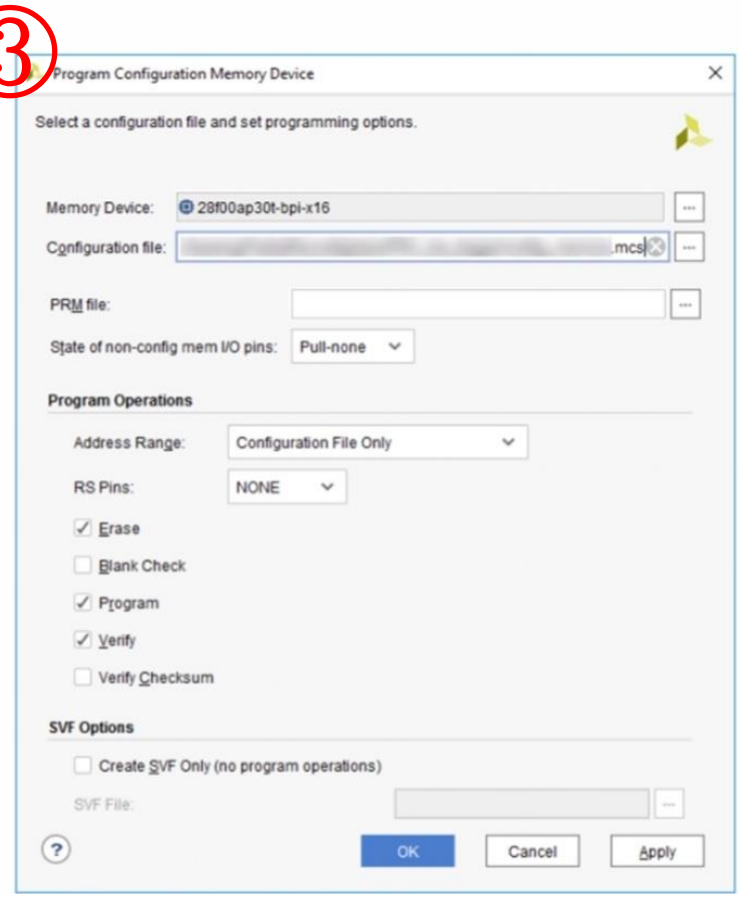
INIT_B	Bi-directional signal. Low potential means the configuration storage is empty.
M2,M1,M0 pins	Means the configuration mode of selection currently
PROGRAM_B	Reset the configuration of FPGA when lowering the signals, on the contrary it starts to load new configuration
CCLK	Configurate clock up to 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.



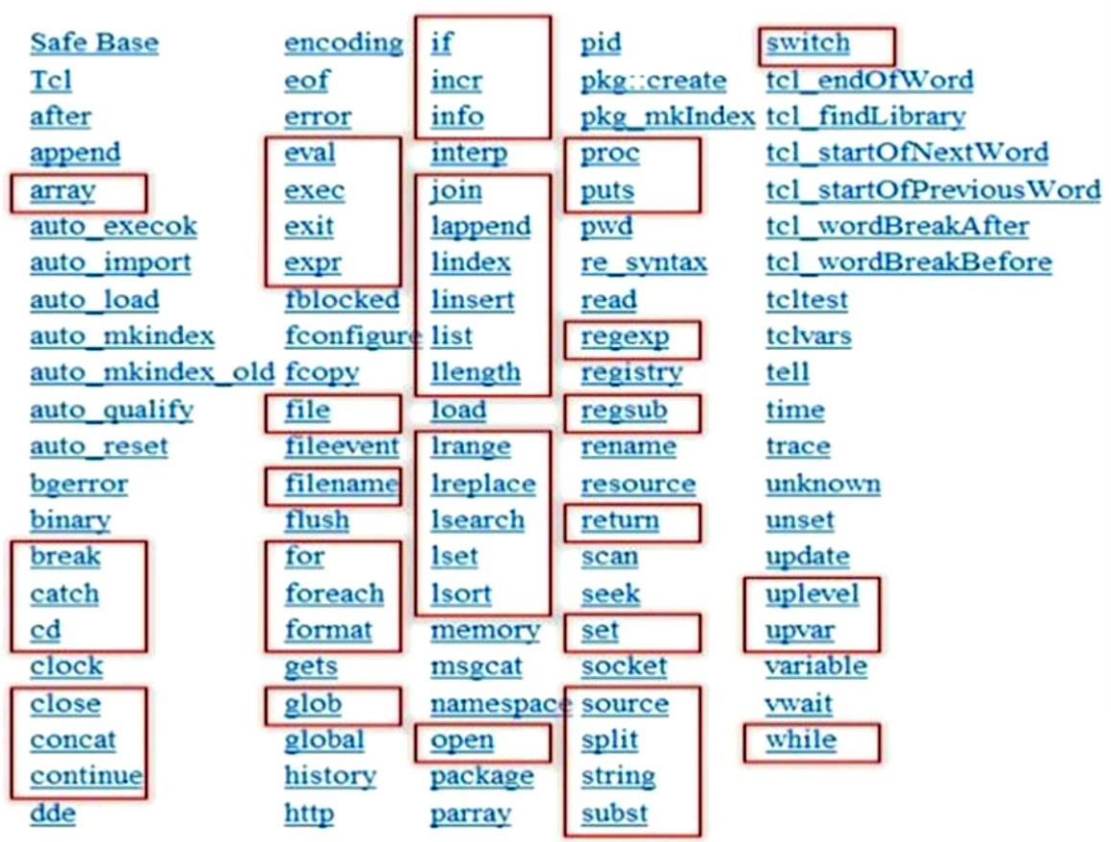




Vivado TCL Environment

Vivado TCL Environment

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

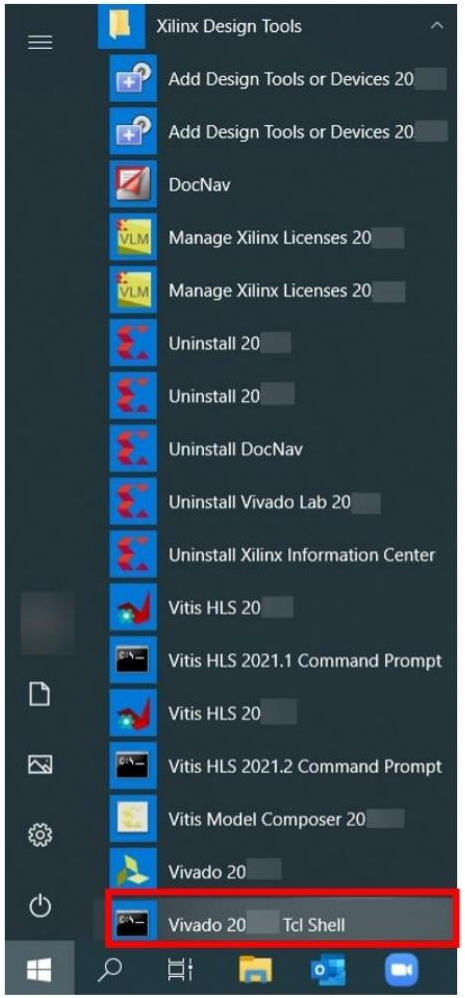


The highlighted Tcl commands are supported in the Vivado IDE

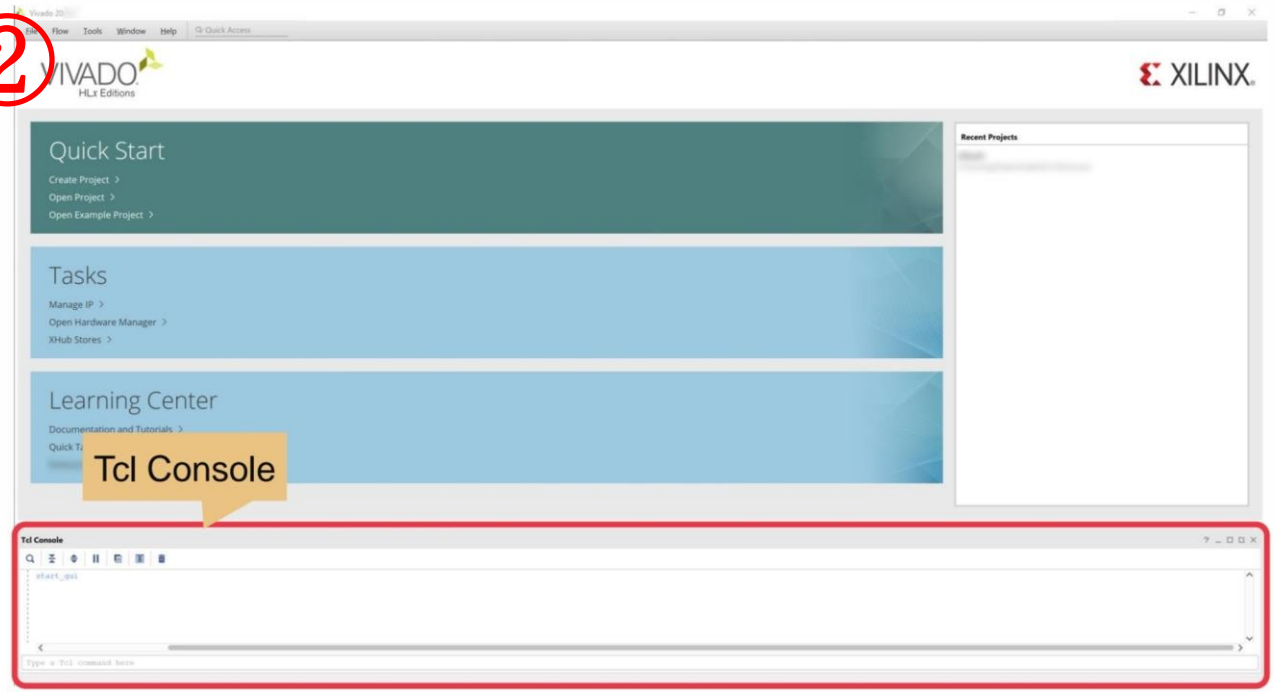
Vivado TCL Environment

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

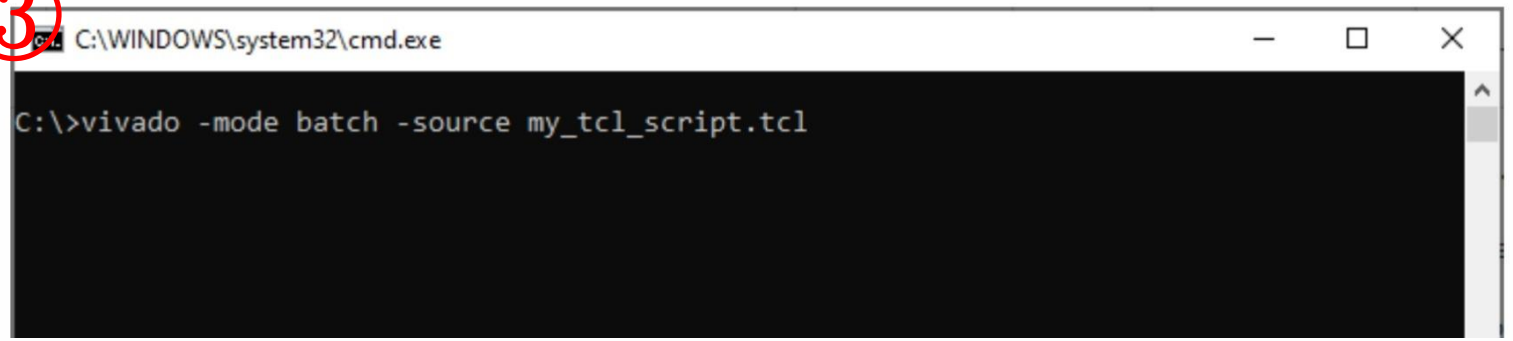
①



②



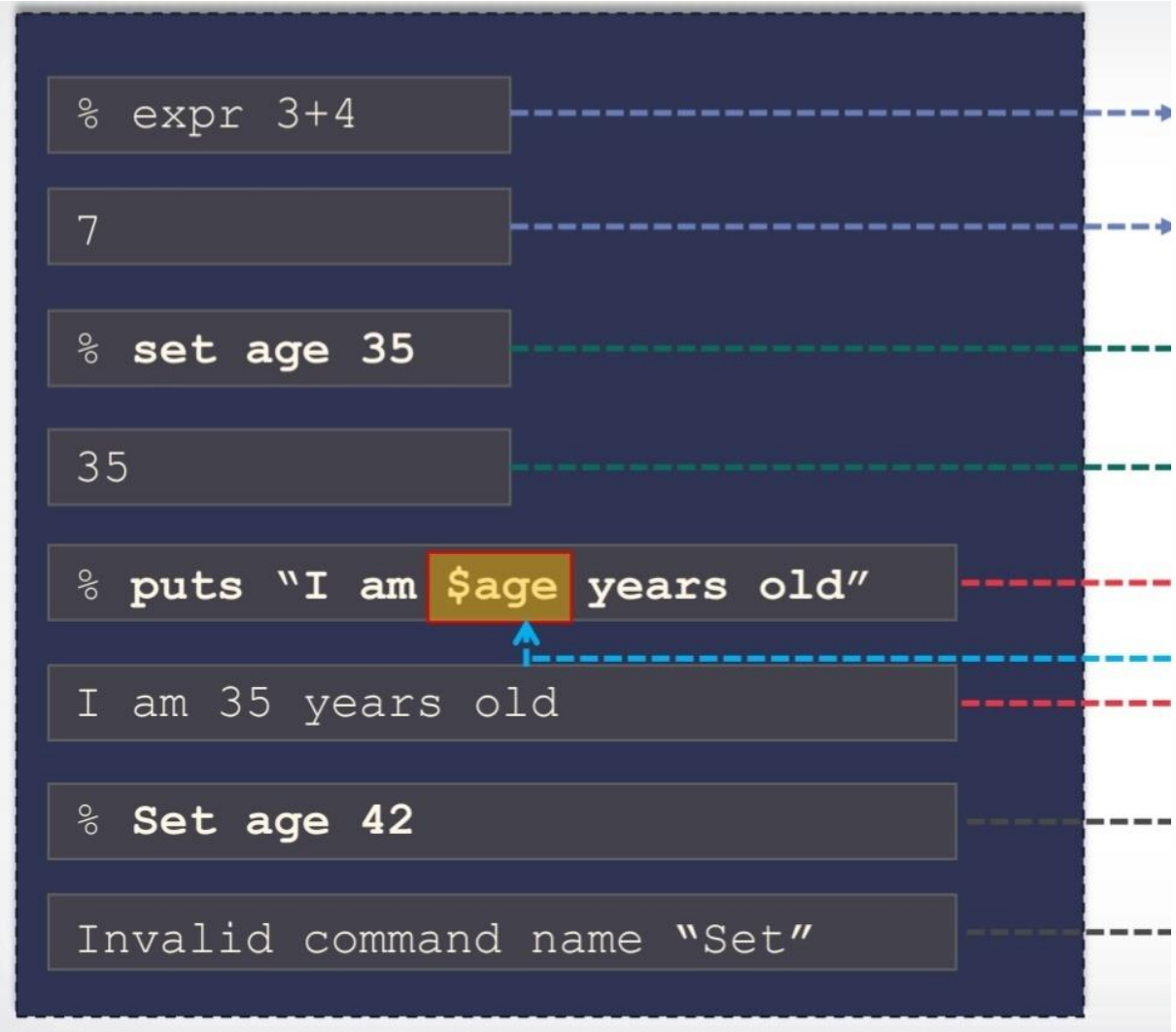
③



XILINX

Vivado TCL Environment

- Tool Command Language (TCL) Basic Syntax



expr is arithmetic operation

result

set variable name and value

result

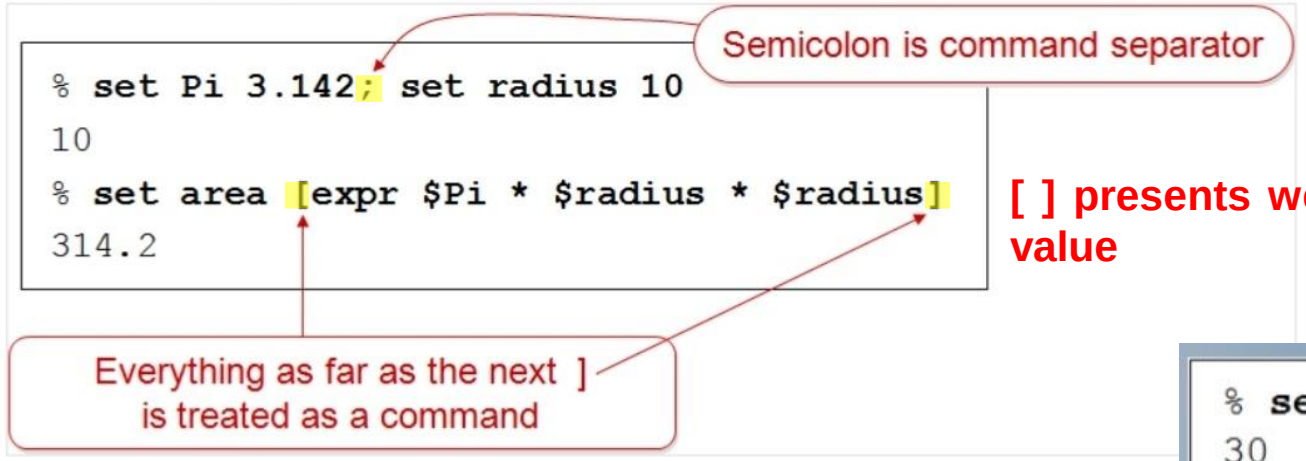
puts is as to “print” on the console

“$” will get the value of variable

Upper and lower case have syntax difference

Vivado TCL Environment

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

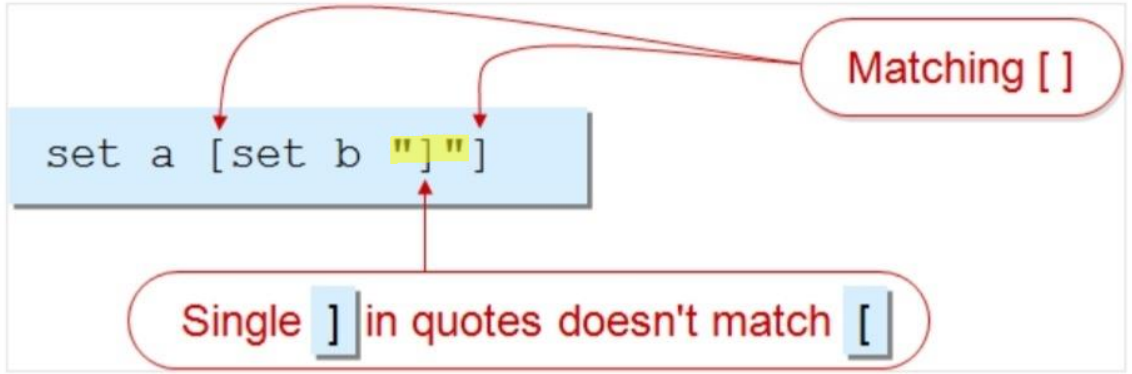


Write multiple TCL in one line and separate by semicolon

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

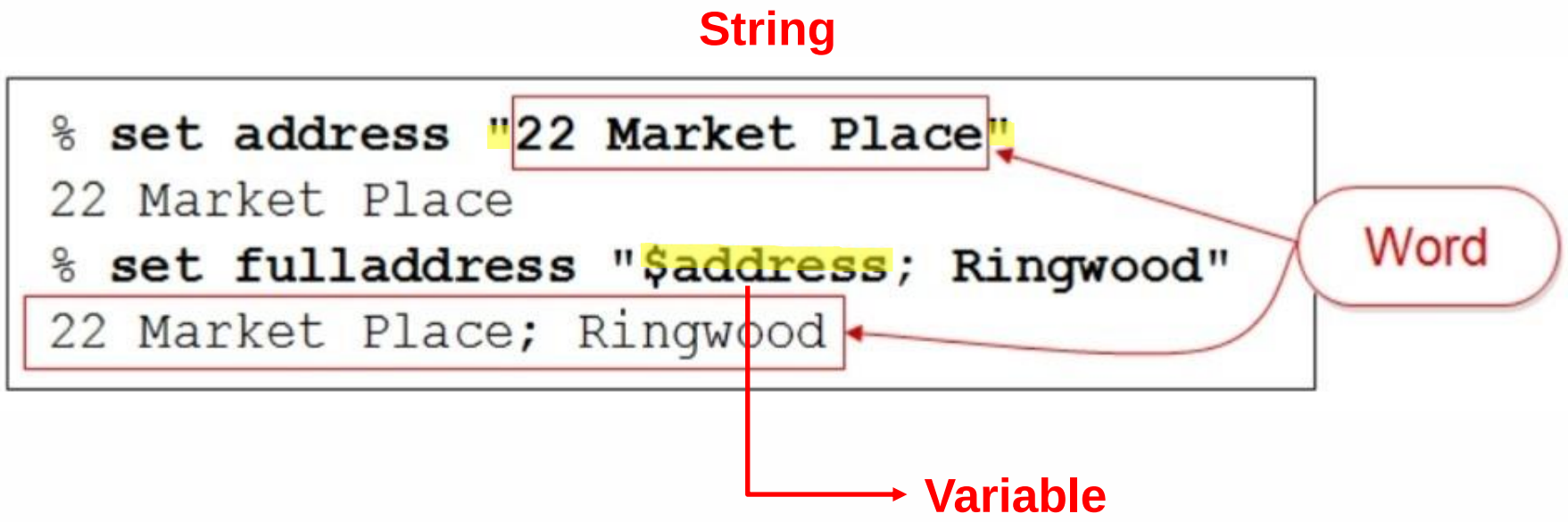
```
% set ten 10 ; set twenty 20 ; set thirty 30
30
%
```

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]
```

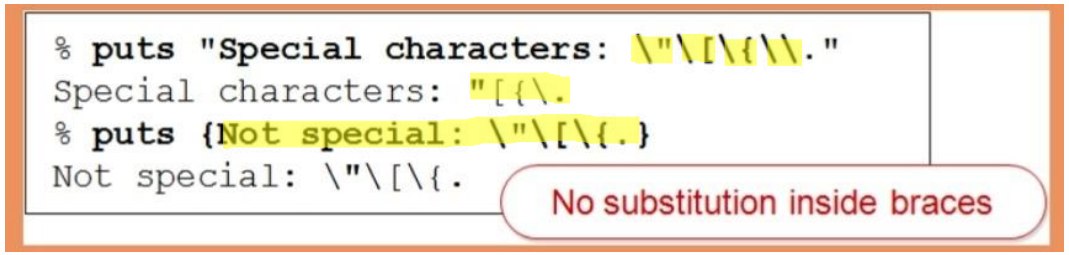
Word

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

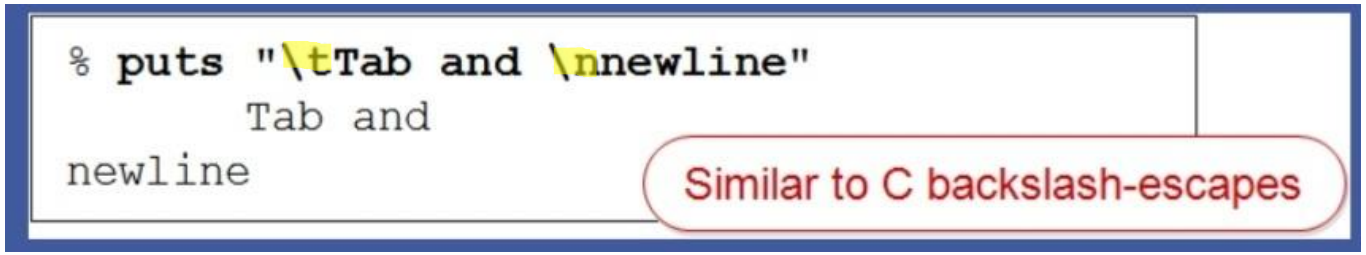
Not recommended

Vivado TCL Environment

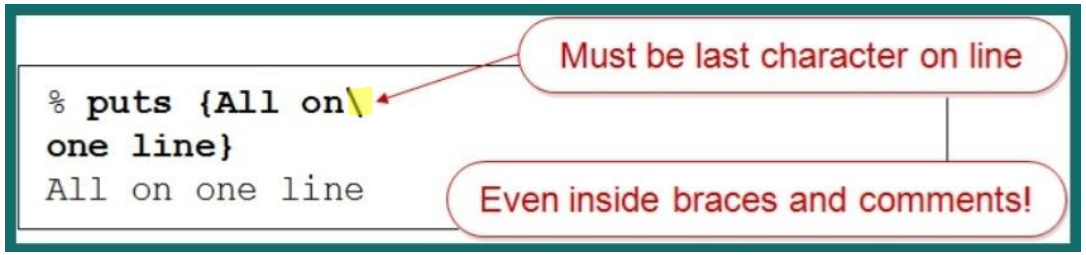
- Tool Command Language (TCL) Basic Syntax
  - Backslash



print special string



\t means tab, \n means newline



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!
Line 3"
set a 5     # This is a syntax error
set b 6 ;   # but this is a valid comment
```

This will become string in quoting

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

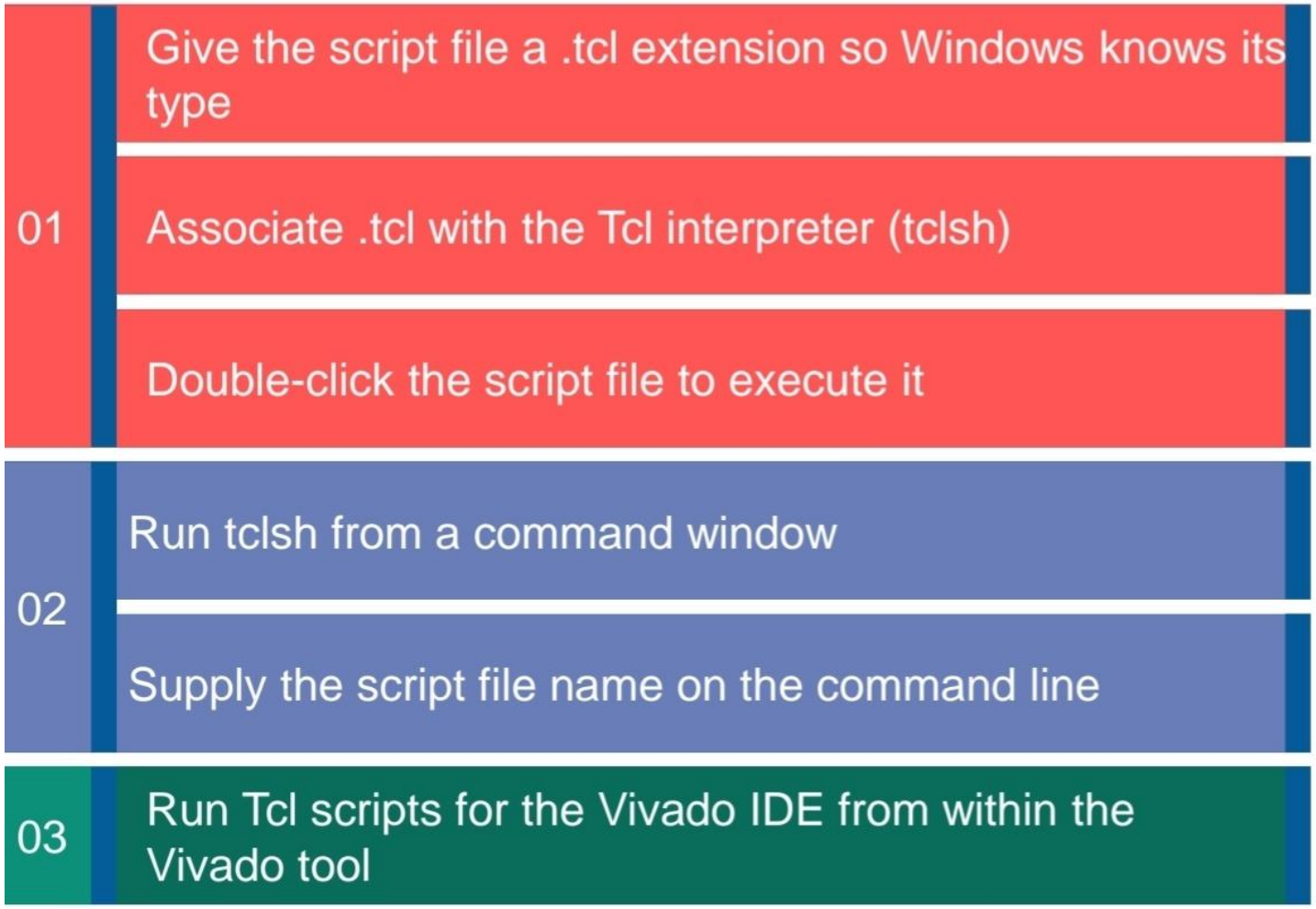
Running the script

```
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



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:

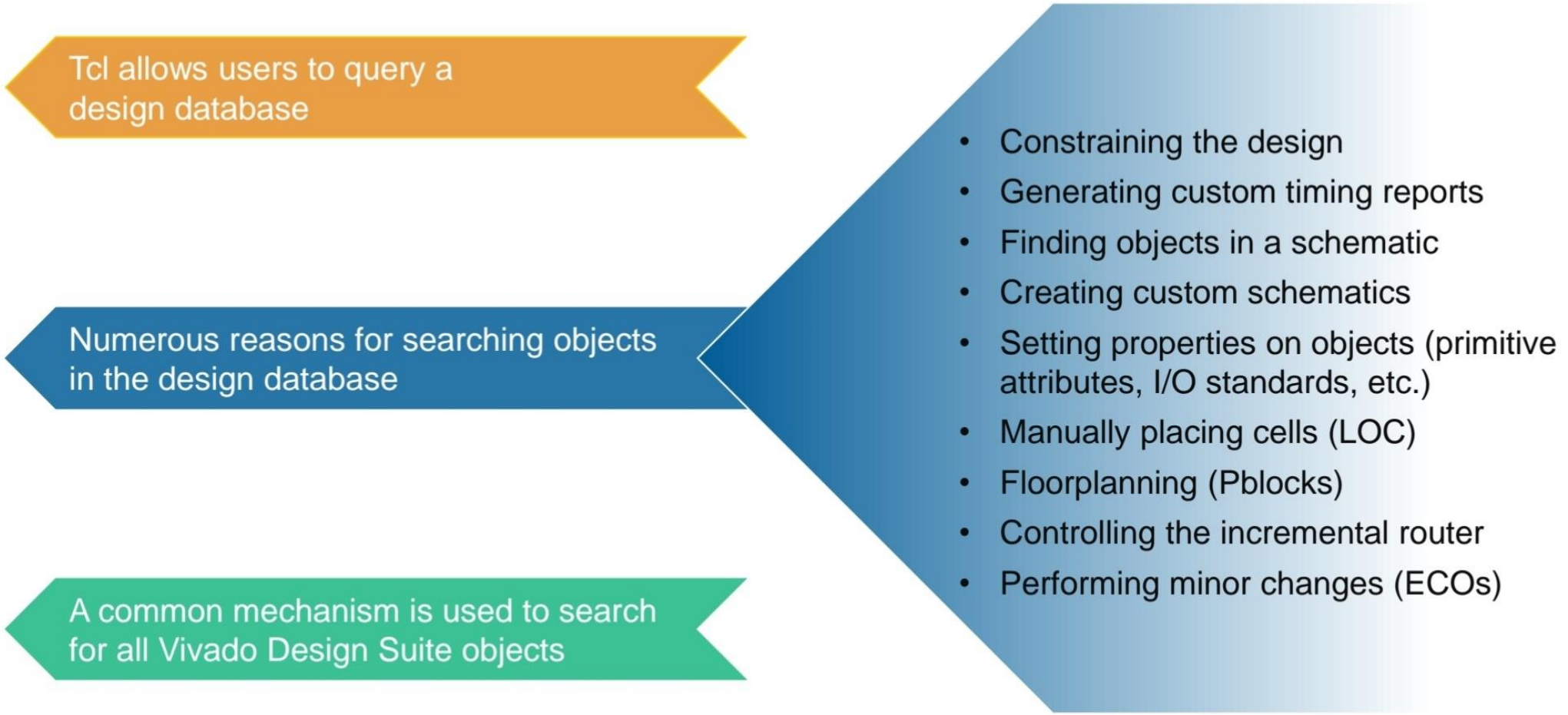
```
create_clock -name clk_pin_p -period 10.000 [get_ports clk_pin_p]
```

Execute programs:

```
place_design, synth_design
```

Vivado TCL Environment

- TCL Capabilities in Vivado



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!

XILINX.