



# Designing FPGAs Using the Vivado Design Suite

Course Agenda  
2022.2

# Agenda

- ▶ **Introduction to the Vivado Design Suite**
- ▶ **Vivado Design Suite Project-Based Mode**
- ▶ **Behavioral Simulation**
- ▶ **Vivado Synthesis**
- ▶ **I/O Pin Planning**
- ▶ **Vivado Implementation**
- ▶ **Vivado Bitstream Generation**
- ▶ **Vivado IP Flow**
- ▶ **Clock Constraints**
- ▶ **Introduction to Vivado Reports**
- ▶ **Vivado Logic Analyzer**

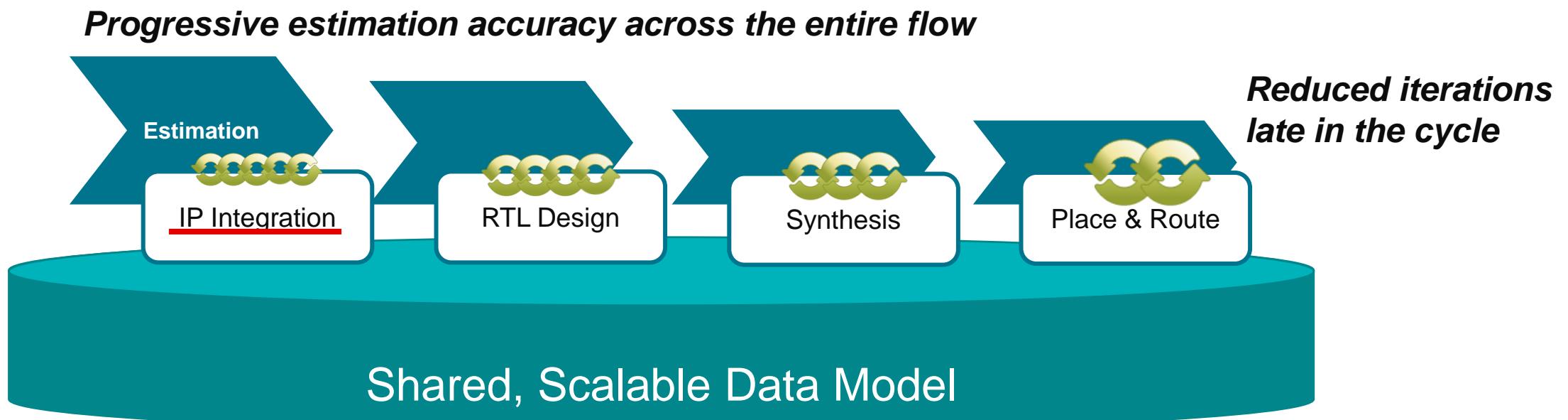


# Introduction to the Vivado Design Suite

2022.2

# What is the Vivado Design Suite?

- ▶ State-of-the-art IDE that encapsulates a complete, fully integrated set of tools for design entry, timing analysis, hardware debugging, and simulation
- ▶ Design analysis offers a shared, scalable data model
- ▶ Early and constant visibility into timing, power, resource utilization, and routing congestion



# Vivado System-Level Design Flows

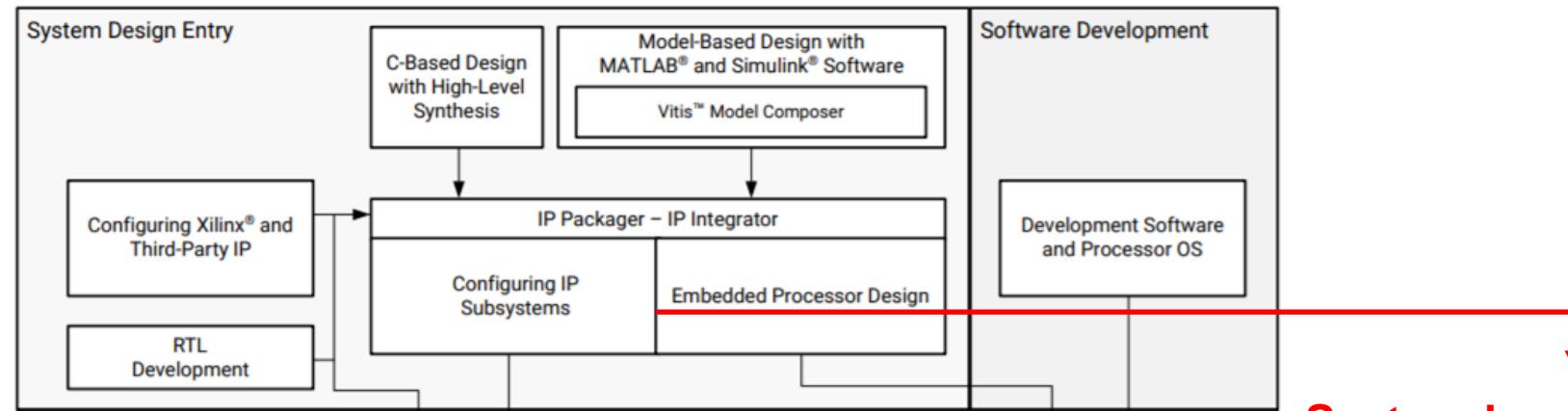
## RTL-to-bitstream design flow

- ▶ Uses traditional register transfer level (RTL)-to-bitstream FPGA design flow for device design, implementation, and verification

## System-level integration flows

- ▶ Focuses on intellectual property (IP)-centric design and C-based design
- ▶ Each of these flows is derived from the RTL-to-bitstream flow

# System-Level Integration Flow

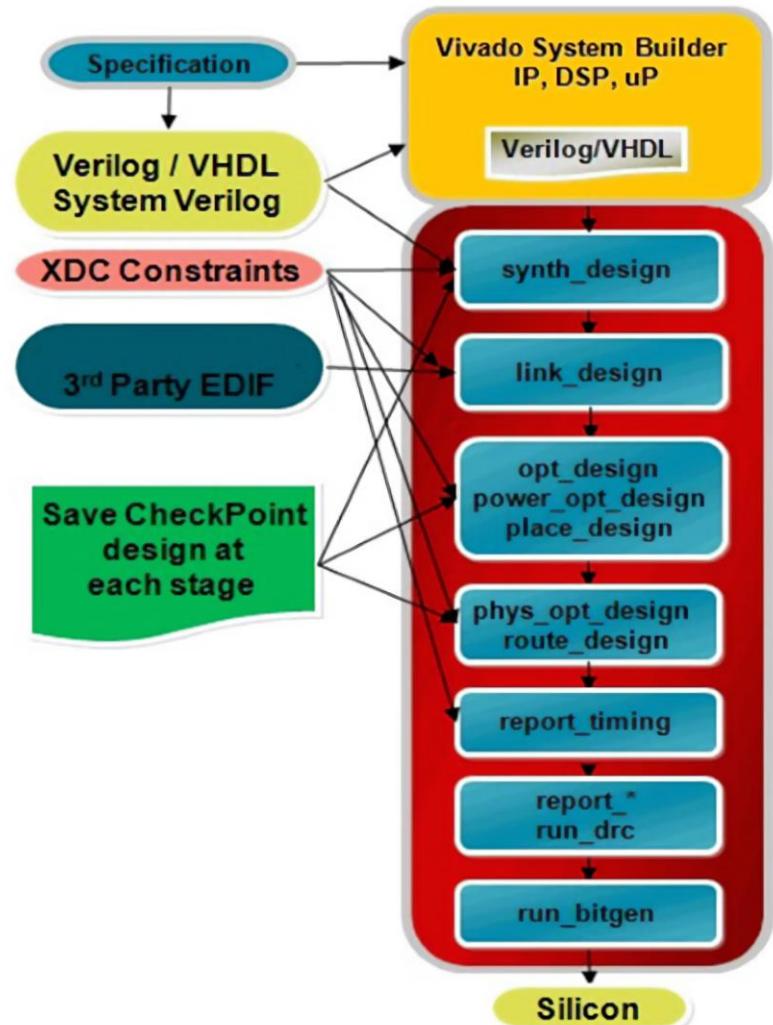


## System-level Integration

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

# Vivado Design Suite Tool Flow

- 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



# Checkpoints

- After the synthesis step, the design can be saved at any stage as a design checkpoint

## What is a Checkpoint?

A checkpoint is a “dump” of the current design as:

- Netlist (in EDIF format)
- Constraints (in XDC format)
- Placement/route information (in XDEF format)

## Accessing a Checkpoint

`open_checkpoint <filename>`

- Creates a new in-memory project and initializes the design with checkpoint contents

`read_checkpoint <filename>`

- Reads the checkpoint file without opening design in memory

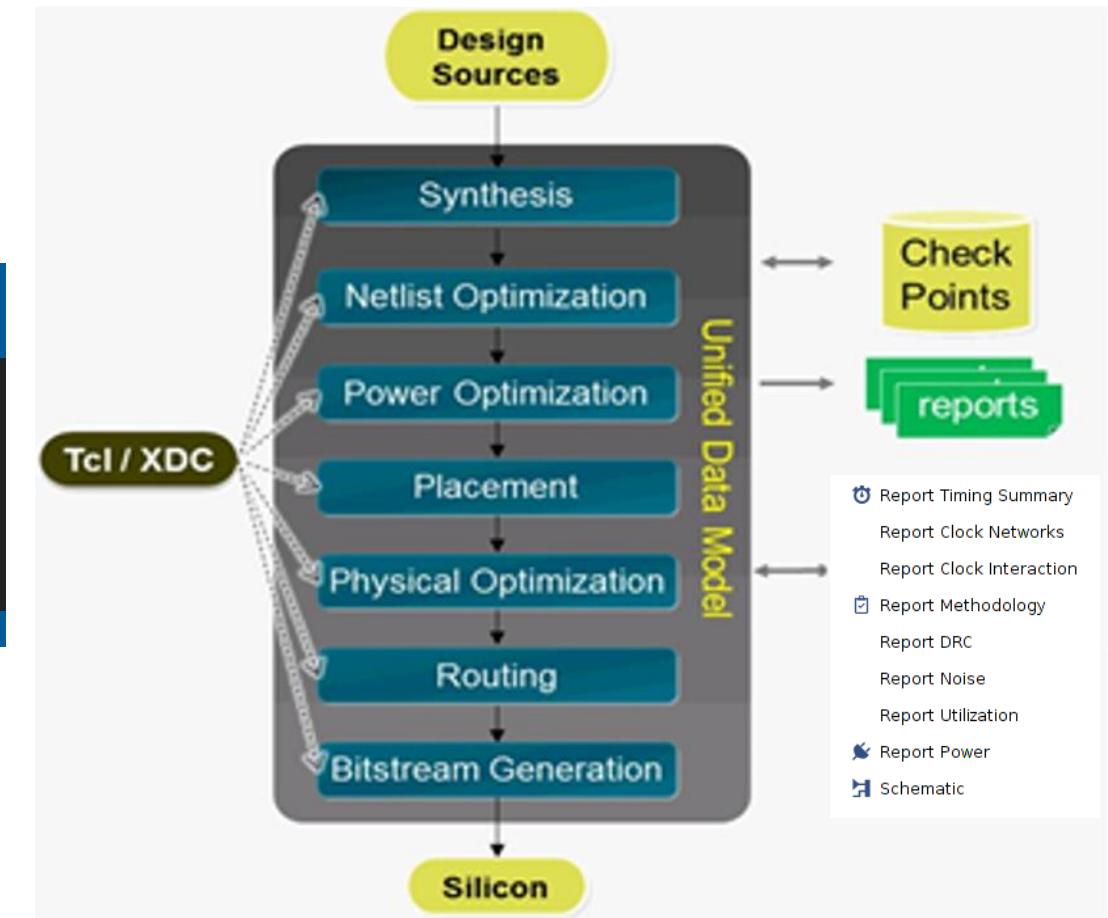
- Checkpoints only written if place/route information exists (after placement)
- Use the `link_design` Tcl command after `read_checkpoint` to initialize the design and load it in memory

# Vivado Design Suite Use Models

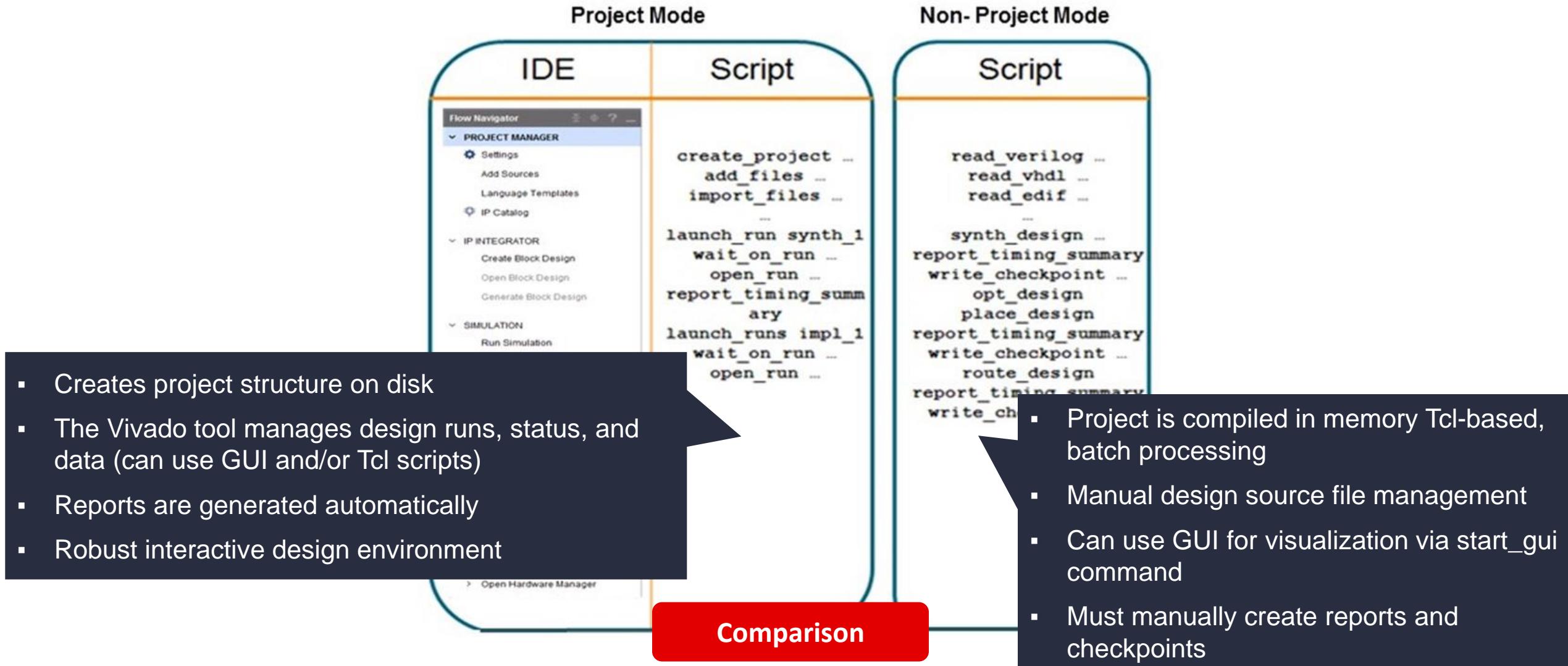
Figure shows the Vivado system-level design flow

With two primary use models

Project Mode	Non-Project Mode
Vivado tools automatically manage your design flow and design data	Fully customizable flow using Tcl enables user control over every aspect of the back-end tools



# Vivado Design Suite Use Models

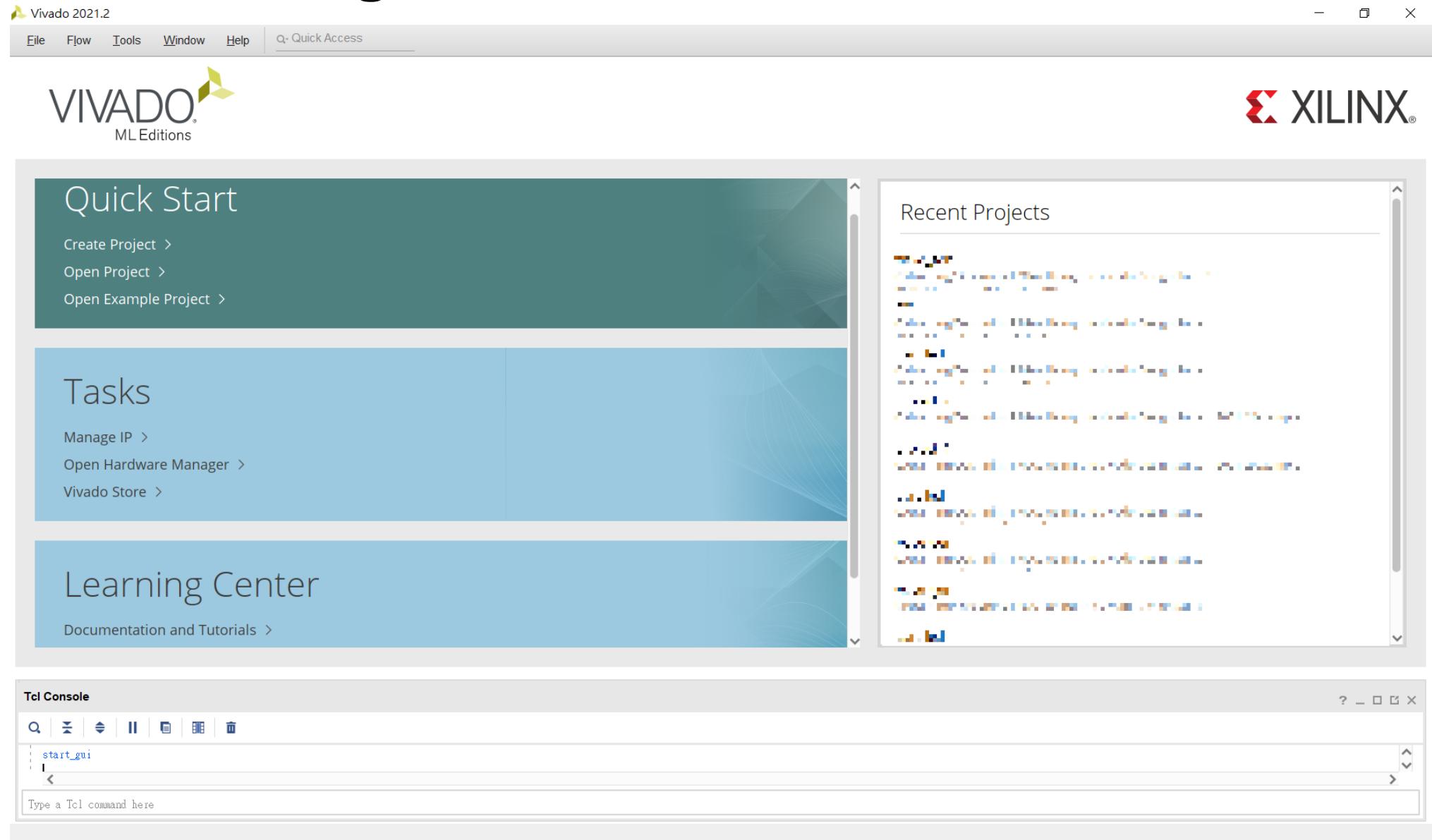




# Vivado Design Suite Project-Based Mode

2022.2

# Vivado Start Page



# Vivado Start Page

Vivado 2021.2

File Flow Tools Window Help

Open Example Project

**VIVADO**  
ML Editions

**Quick Start**

- Create Project >
- Open Project >
- Open Example Project > **Red Arrow**

**Tasks**

- Manage IP >
- Open Hardware Manager >
- Vivado Store >

**Learning Center**

Documentation and Tutorials

Tcl Console

```
start_gui
```

Type a Tcl command here

**Select Project Template**

Select one of the below predefined templates on which to base your new project

**Templates**

- Xilinx
  - Platforms
    - MPSOC Extensible Embedded Platform
    - MPSOC Extensible Embedded
    - Versal Extensible Embedded
    - Versal Extensible Embedded
    - Versal DFX Extensible Embe
  - Soft Processors
    - MicroBlaze Design Presets
    - TMR Microblaze Example De
    - Versal MicroBlaze Design Pr
  - Versal
    - AXI DMA
    - Multi-Rate GTY
    - Versal ACAP Processor Perf
    - Versal CPM Debug-over-PCI

**Description**

**MPSOC Extensible Embedded Platform**

An extensible platform is the foundation of the Vitis software acceleration flow. This platform enables Vitis to create PL kernels. It gives the kernels access to PS (MPSOC), DDR memory, an interrupt controller and clocking resources.

Extensible interfaces are marked with PFM properties and the platform is exported to Vitis using write\_hw\_platform to create an XSA.

S\_AXI Port

ZYNQ® UltraSCALE+

M\_AXI Port

Clock Wizard

Interrupt Controller

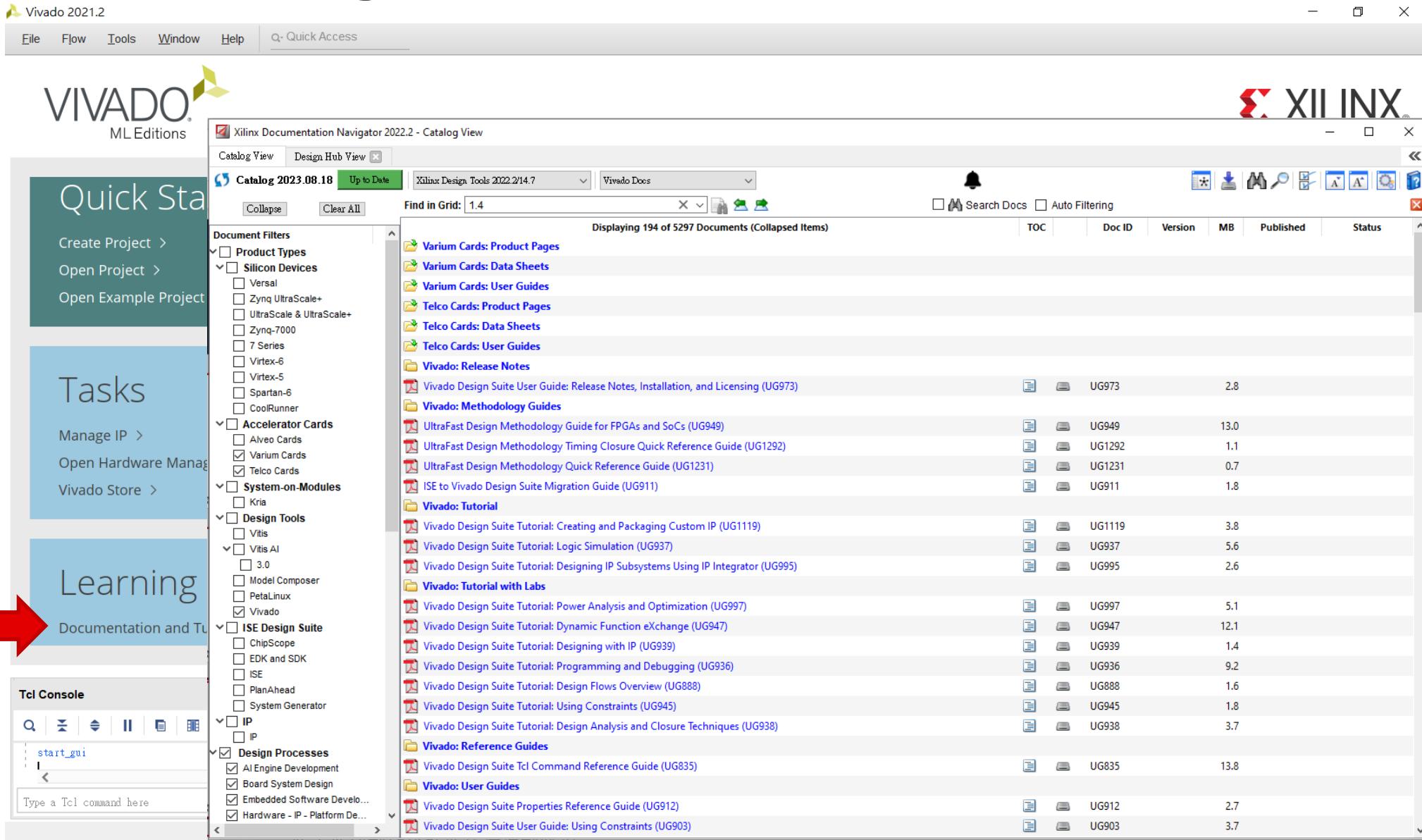
Vitis Kernels

DDR4 (MIG) Controller

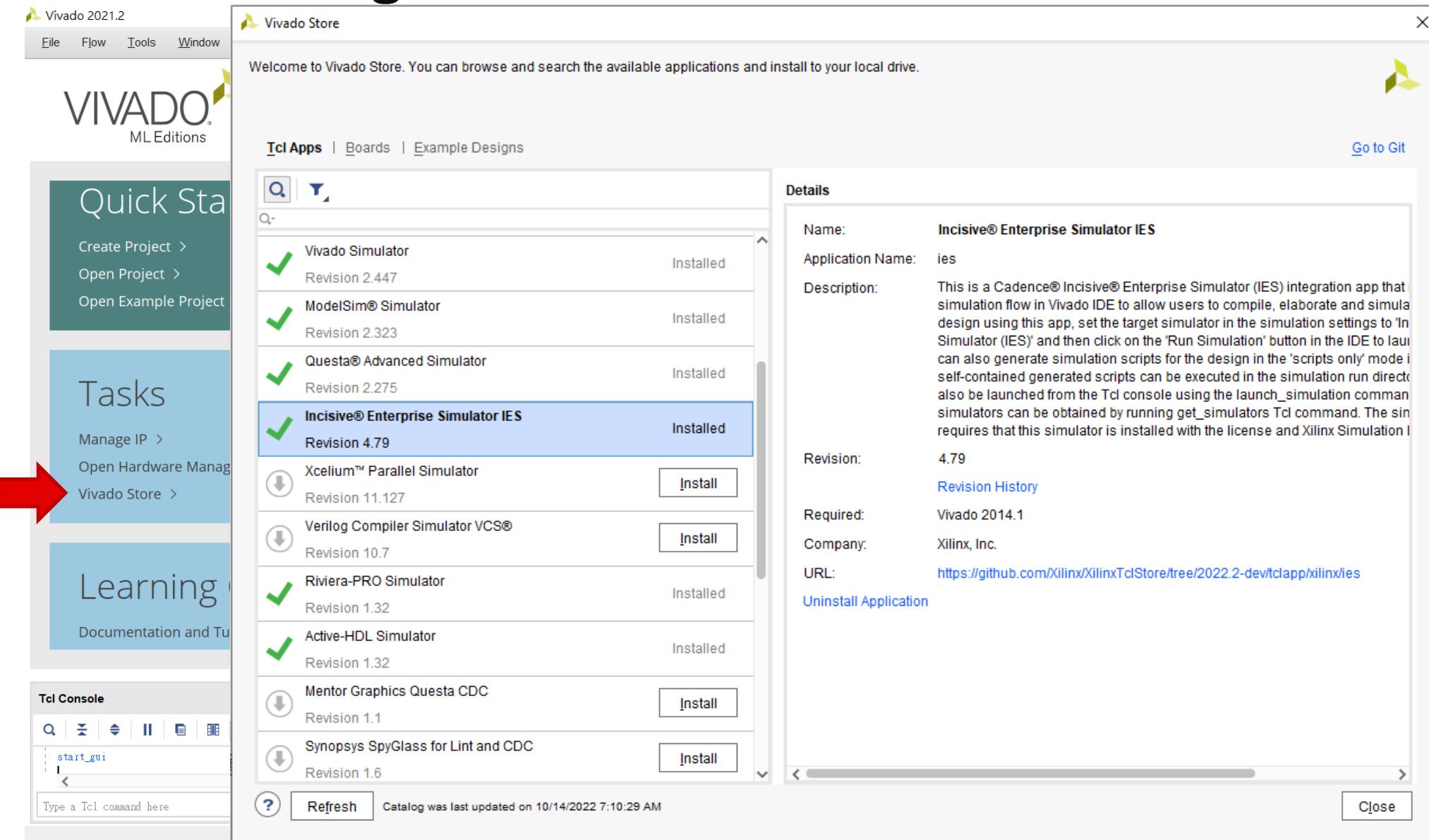
Refresh Catalog was last updated on 05/04/2023 5:59:59 PM

< Back Next > Finish Cancel

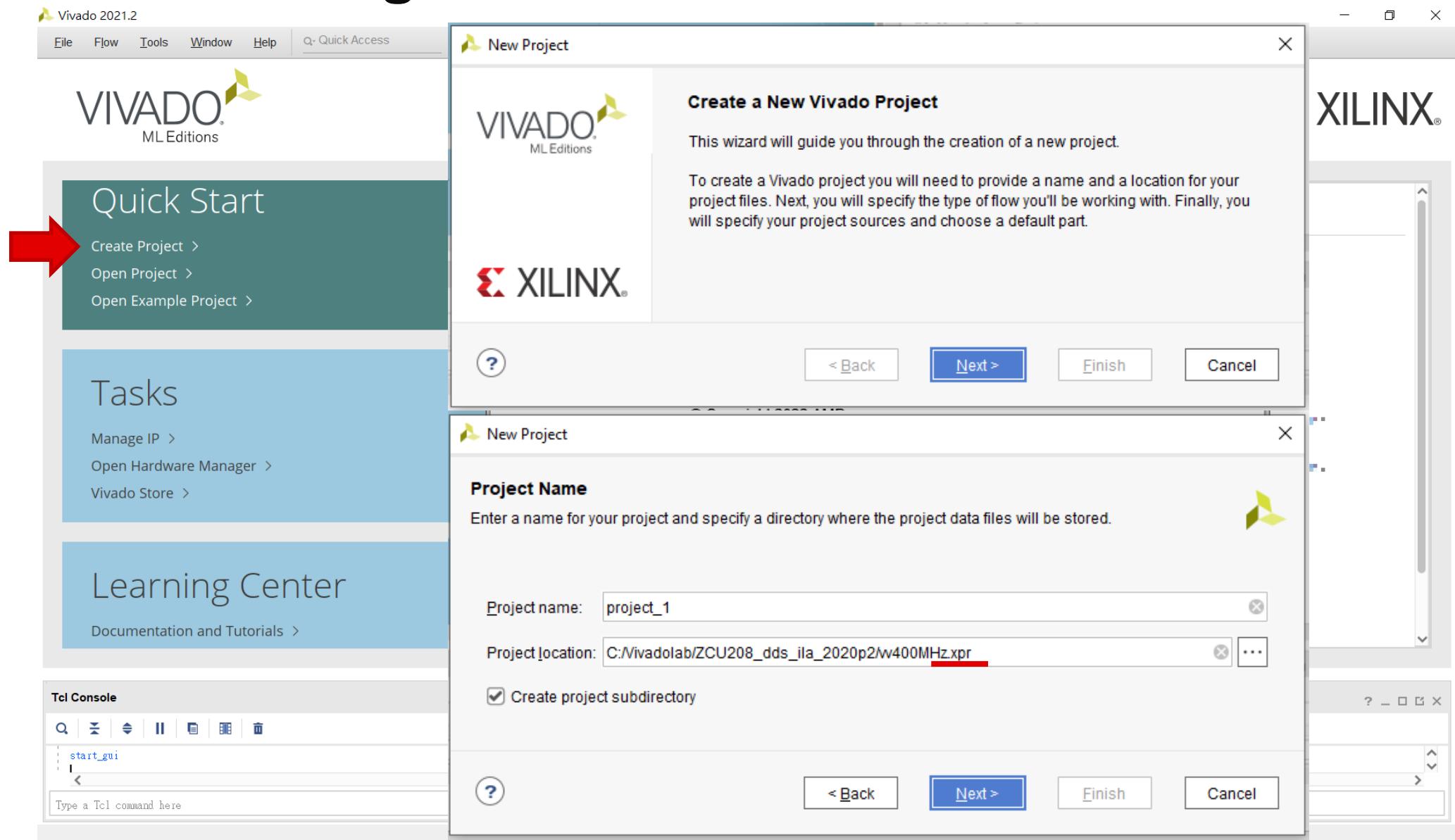
# Vivado Start Page



# Vivado Start Page



# Vivado Start Page

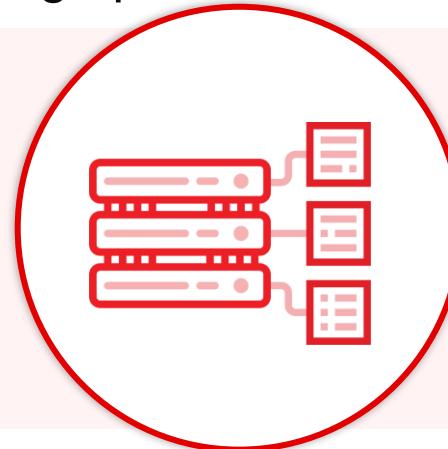


# What is a Project?

A project is an encapsulation of the design process

Repository of all files that are part of the design process

Repository of intermediate and final design netlists

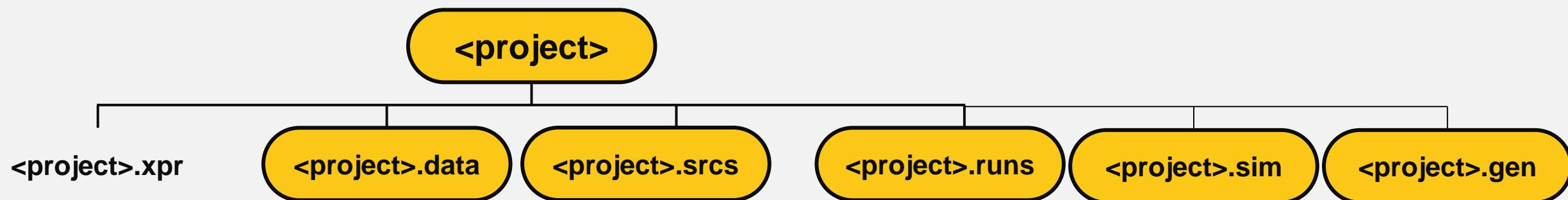


Project data to link all the elements together

State information that stores the current state of the project

## Project Directory

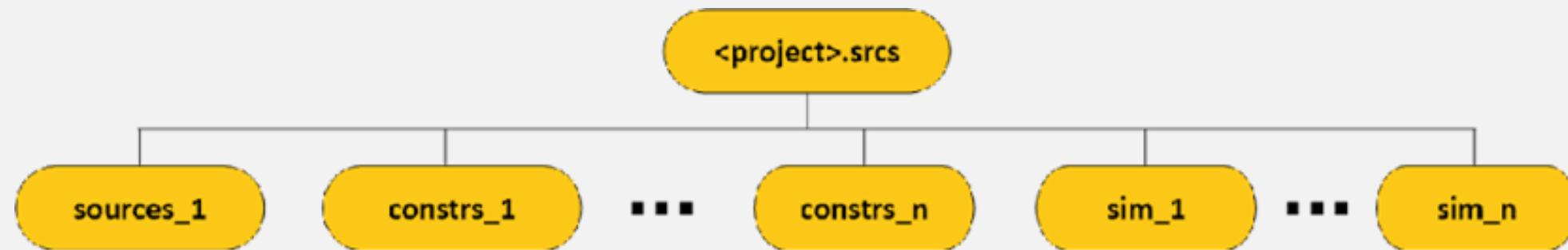
The project is created by using the Create Project option in the IDE or with the `create_project` command



# Project Filesets

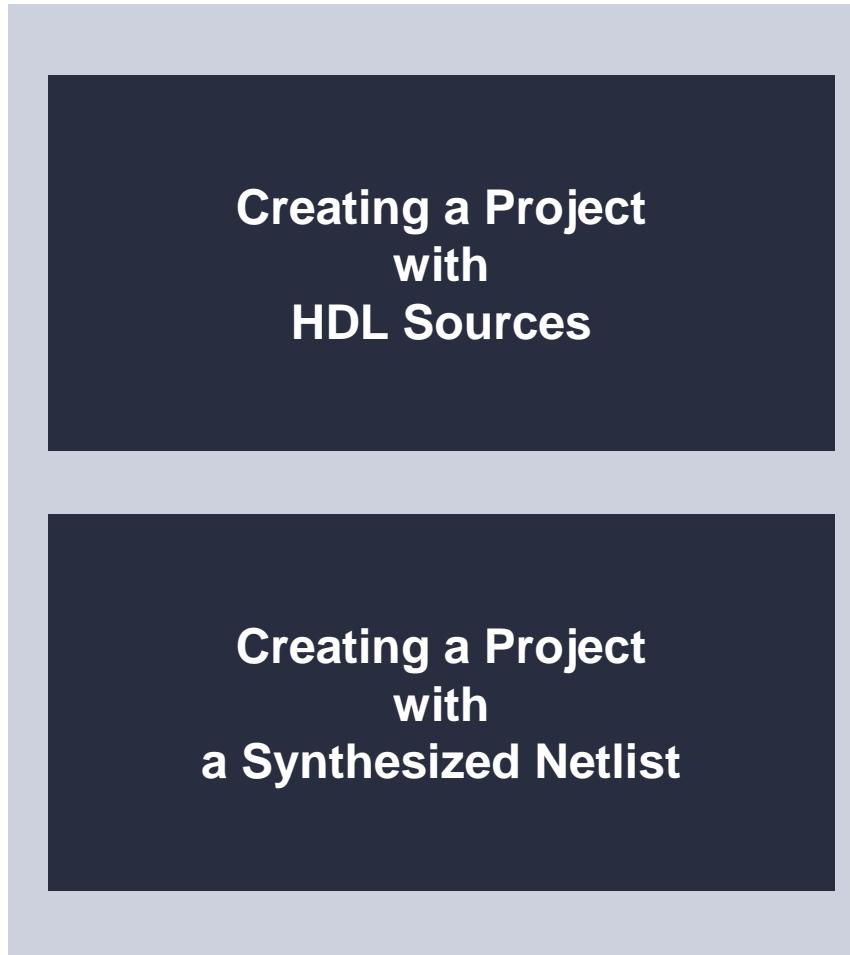
Files are used for specific purposes and are kept in different filesets

Filesets are Vivado Design Suite objects and are associated with information on disk



You can have multiple filesets of some types (simulation and constraint)

# New Project Creation Wizard



New Project

**Project Type**

Specify the type of project to create.

**RTL Project**  
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.

Do not specify sources at this time

Project is an extensible Vitis platform

**Post-synthesis Project**  
You will be able to add sources, view device resources, run design analysis, planning and implementation.

Do not specify sources at this time

**I/O Planning Project**  
Do not specify design sources. You will be able to view part/package resources.

**Imported Project**  
Create a Vivado project from a Synplify, XST or ISE Project File.

**Example Project**  
Create a new Vivado project from a predefined template.

< Back Next > Finish Cancel

# New Project Creation Wizard

The screenshot shows the New Project Creation Wizard with two open windows:

**Add Sources** (Left Window):

- Description:** Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.
- UI Elements:** A large list area with a toolbar (+, -, up, down), a note to "Use Add Files, Add Directories or Create File buttons below", and three buttons: **Add Files**, **Add Directories**, and **Create File**.
- Checkboxes:**  Scan and add RTL include files into project,  Copy sources into project,  Add sources from subdirectories.
- Language Selection:** Target language: Verilog, Simulator language: Mixed.
- Buttons:** < Back, Next >, Finish, Cancel.

**Add Constraints (optional)** (Right Window):

- Description:** Specify or create constraint files for physical and timing constraints.
- UI Elements:** A large list area with a toolbar (+, -, up, down), a note to "Use Add Files or Create File buttons below", and two buttons: **Add Files** (highlighted in blue) and **Create File**.
- Checkboxes:**  Copy constraints files into project.
- Buttons:** < Back, Next >, Finish, Cancel.

# New Project Creation Wizard

The image shows two overlapping windows from the Xilinx Vivado software.

**Left Window: Default Part**

This window allows you to choose a default Xilinx part or board for your project. It has tabs for "Parts" (selected) and "Boards".

**Filters:**

- Category: All
- Family: All
- Search: Q-

**Data Table (Parts):**

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs
xc7vx485tfg1157-3	1157	600	303600	607200	1030	0	2800
xc7vx485tfg1157-2	1157	600	303600	607200	1030	0	2800
xc7vx485tfg1157-2L	1157	600	303600	607200	1030	0	2800
xc7vx485tfg1157-1	1157	600	303600	607200	1030	0	2800

**Right Window: New Project Summary**

This window displays the summary of the new project settings.

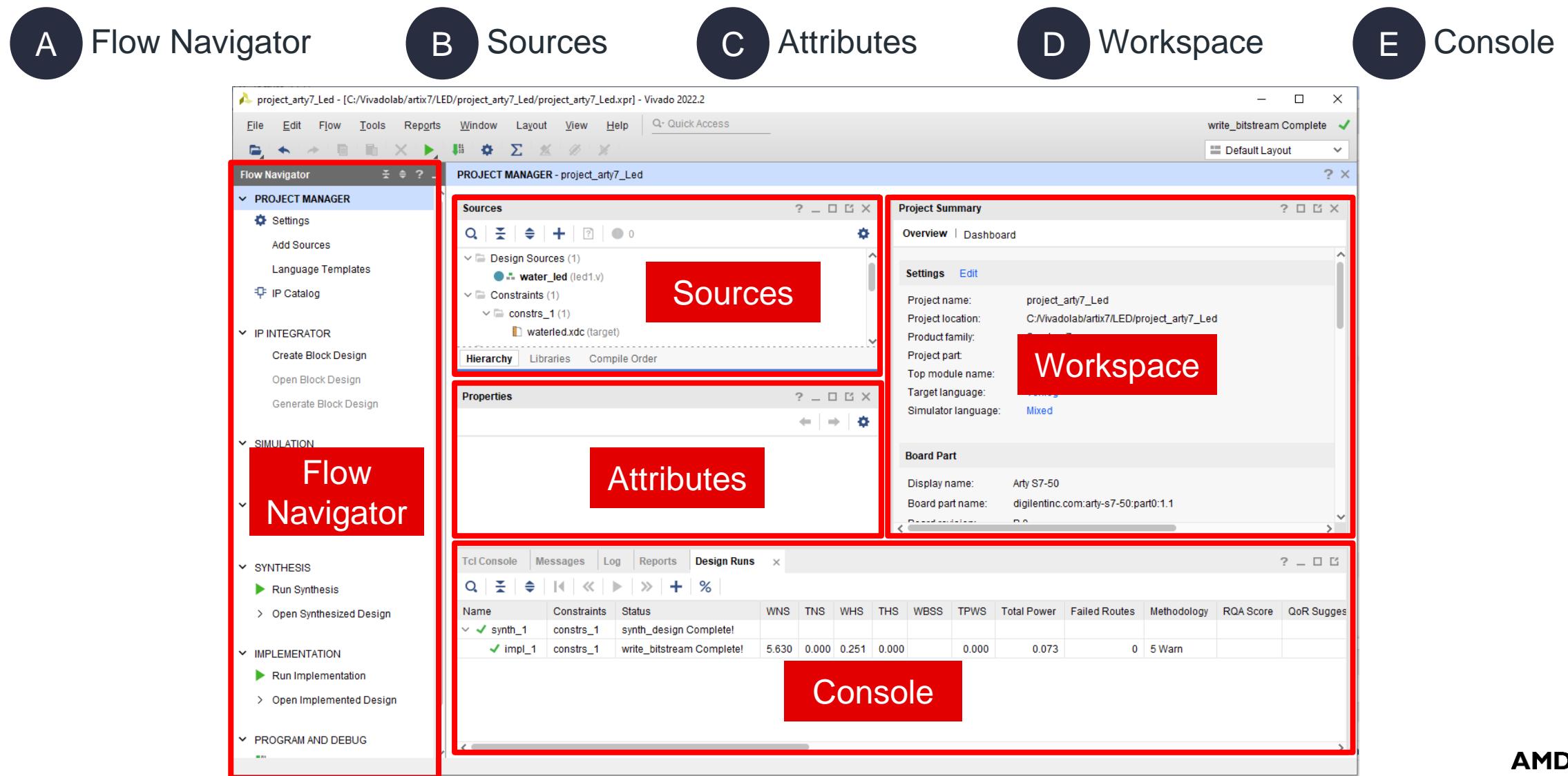
**Project Details:**

- A new RTL project named 'lecture' will be created.
- The default part and product family for the new project:
  - Default Board: Arty S7-50
  - Default Part: xc7s50csga324-1
  - Family: Spartan-7
  - Package: csga324
  - Speed Grade: -1

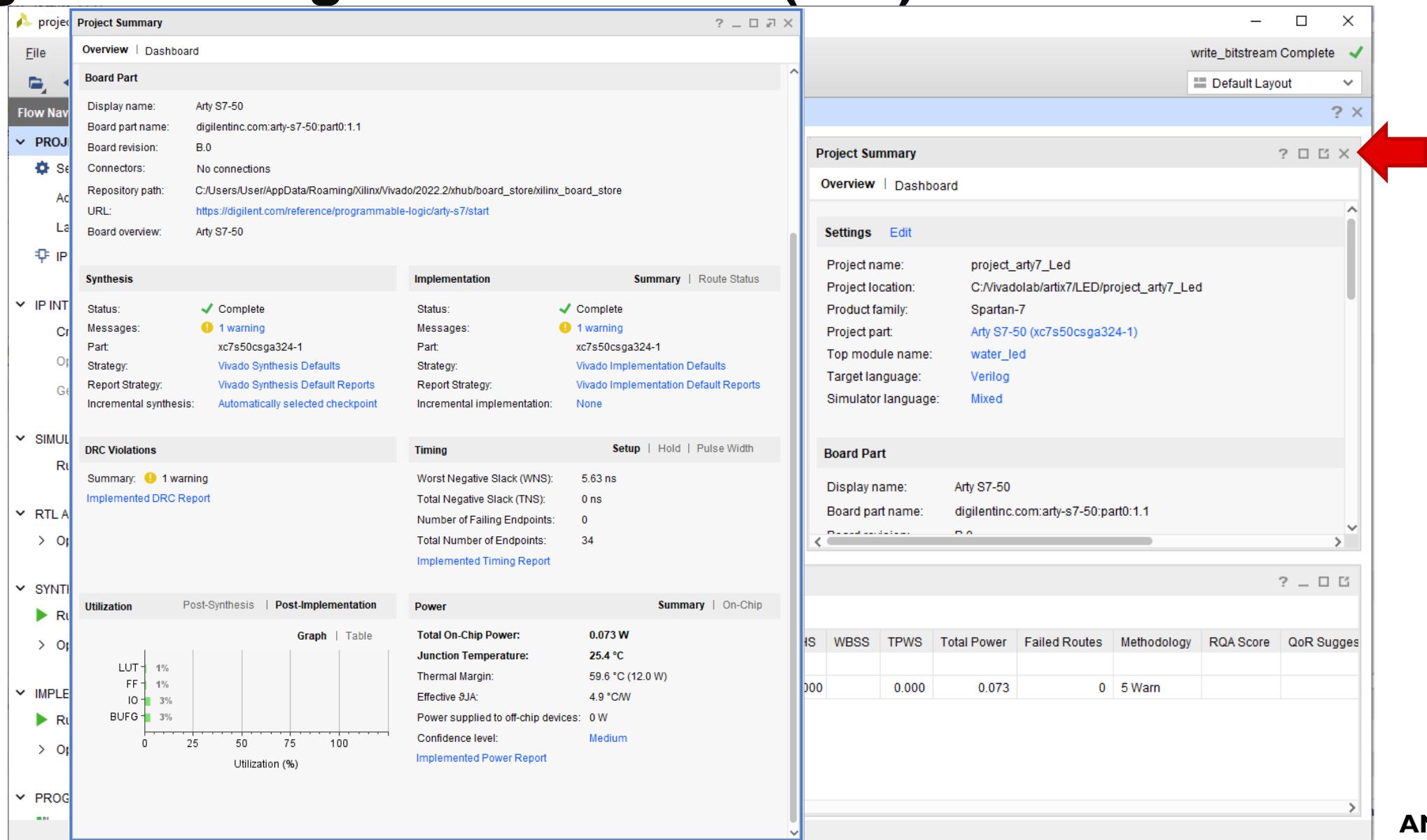
To create the project, click Finish.

Buttons at the bottom: < Back, Next >, Finish, Cancel.

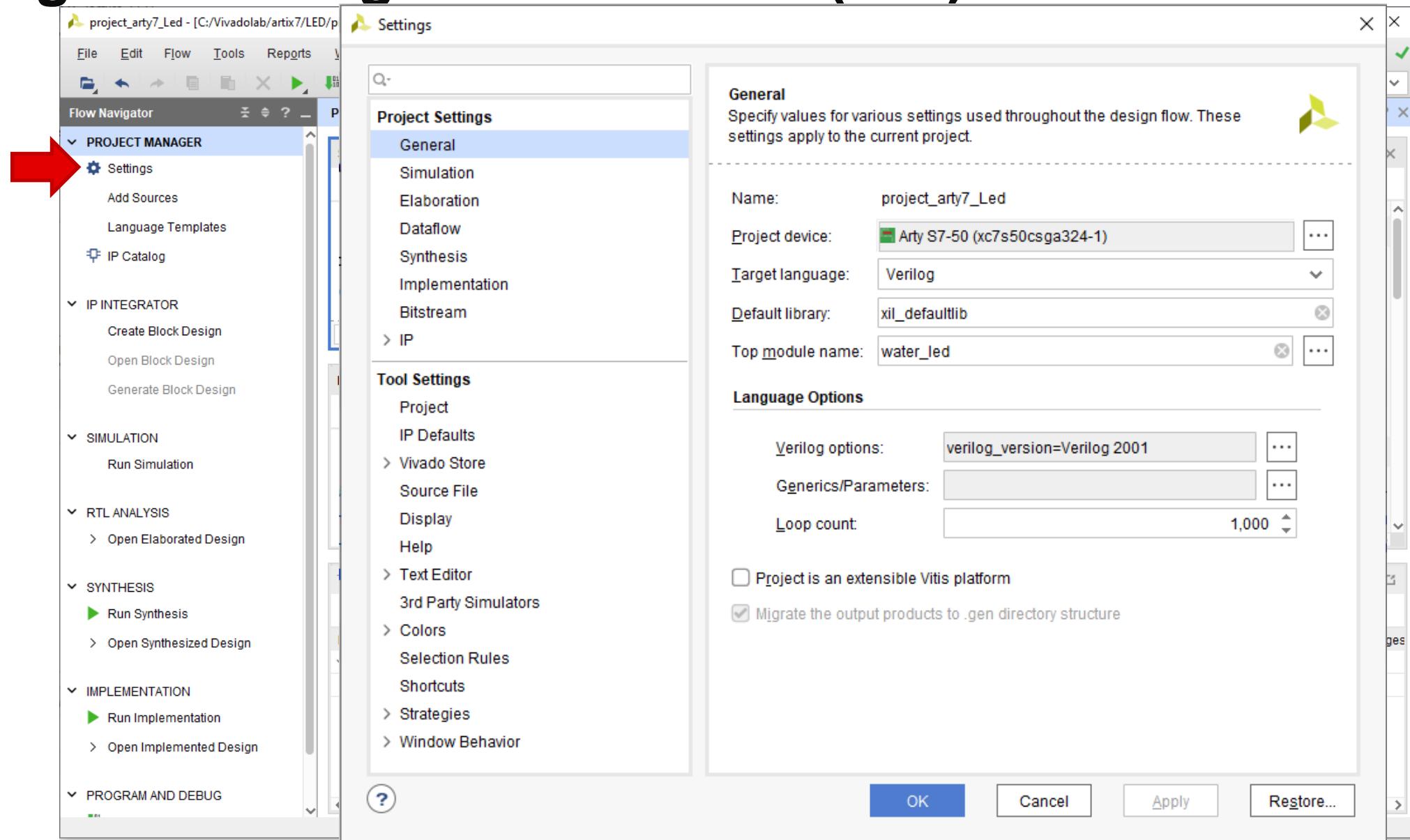
# Integrated Design Environment (IDE)



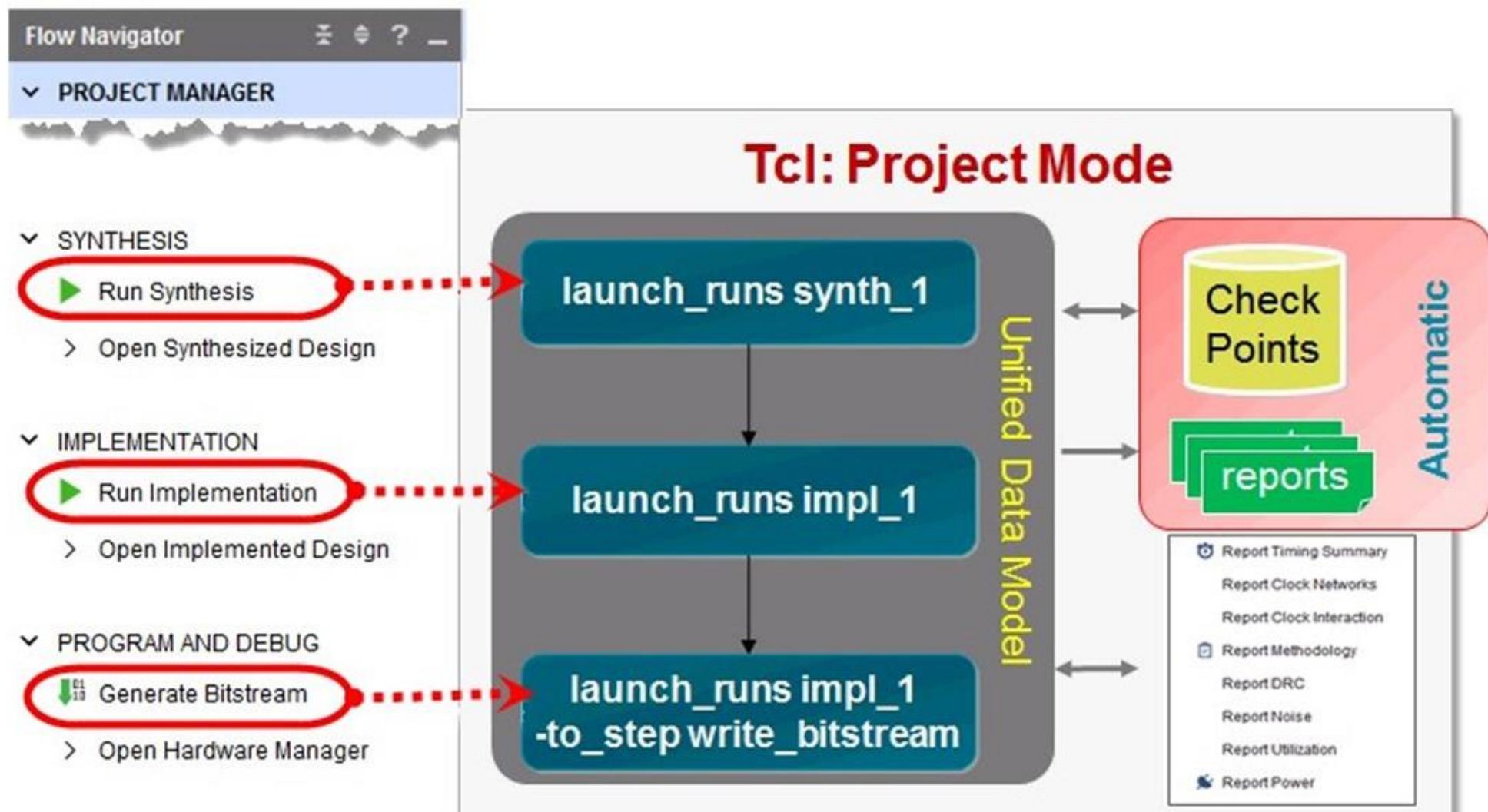
# Integrated Design Environment (IDE)



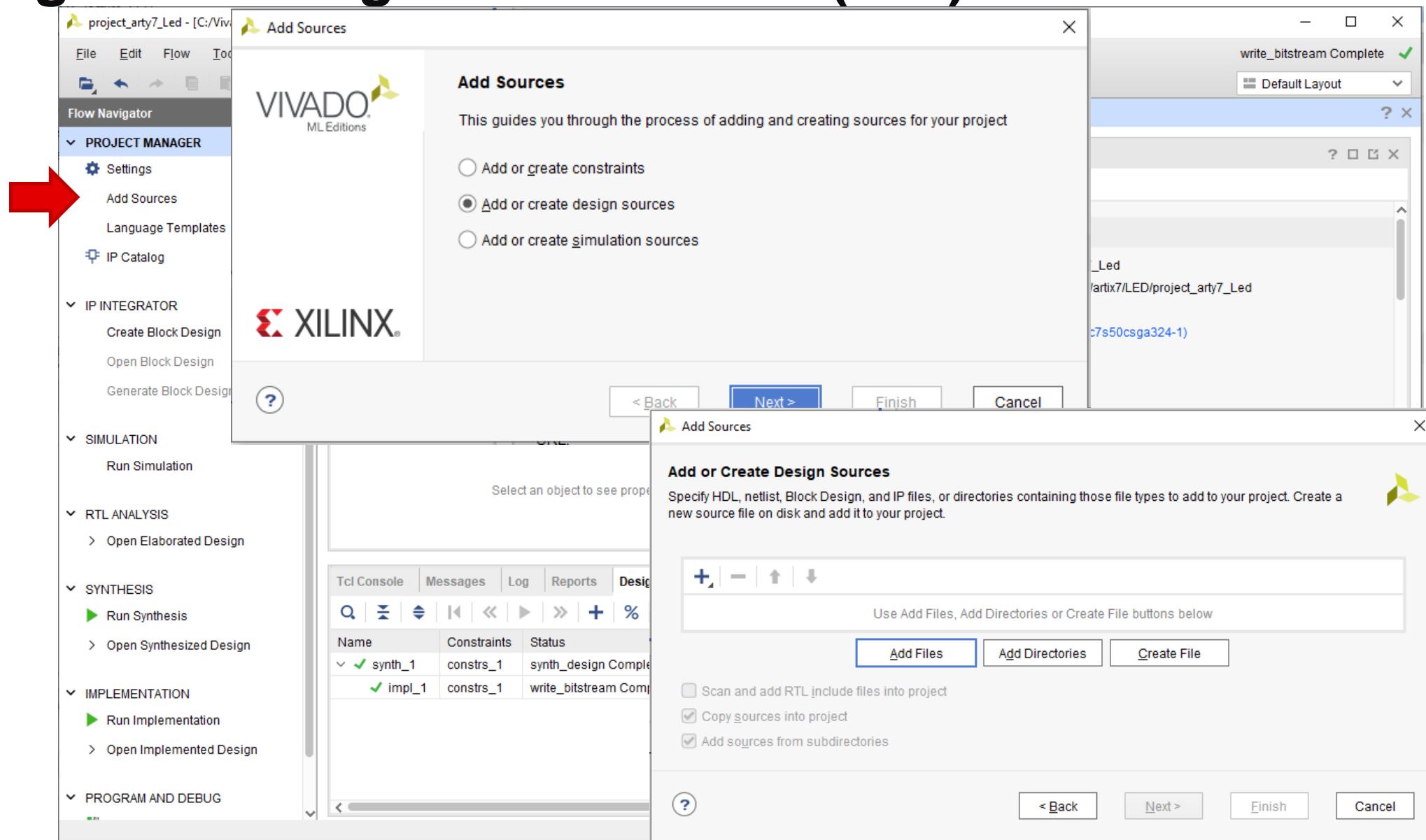
# Integrated Design Environment (IDE)



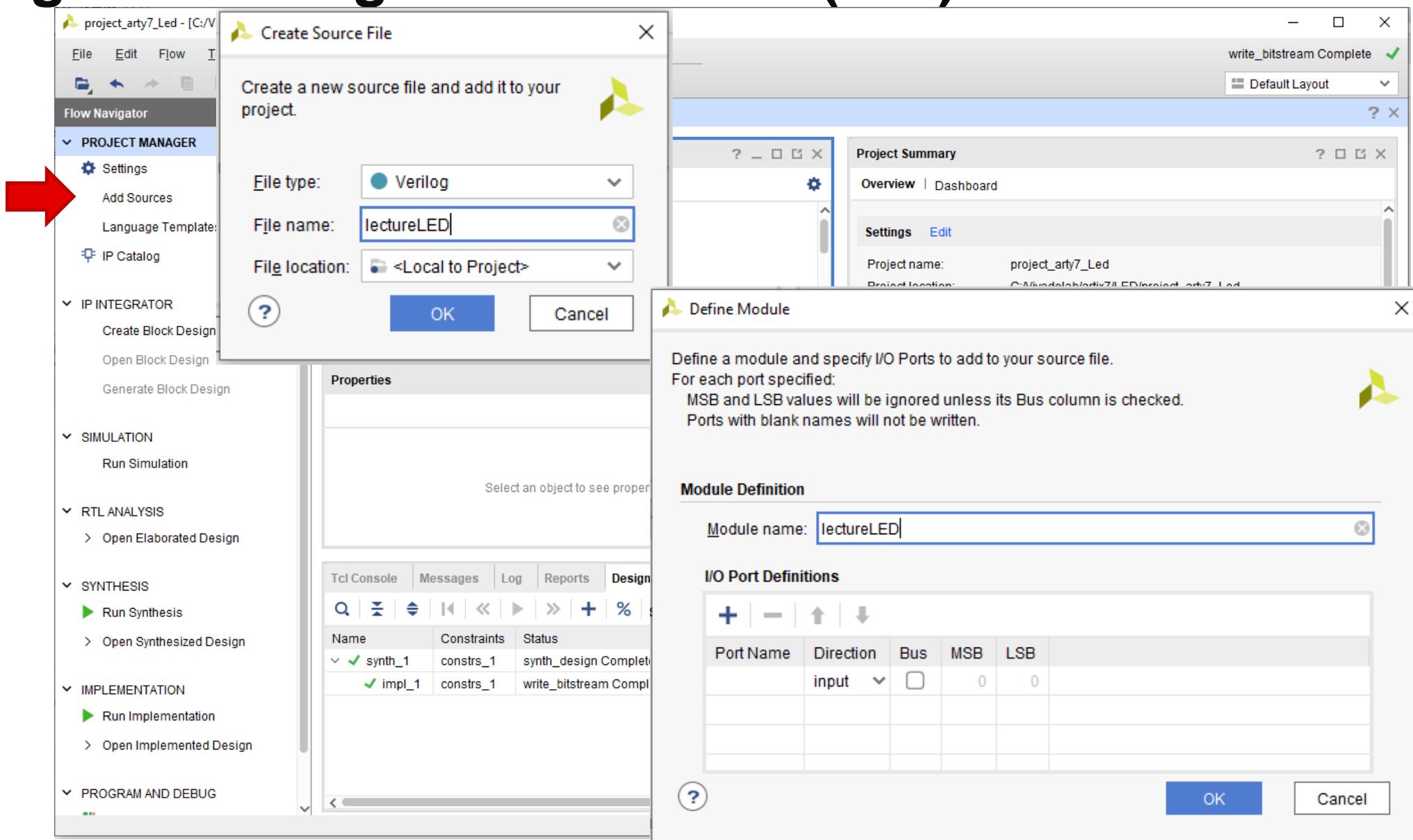
# Project Mode: GUI → Tcl



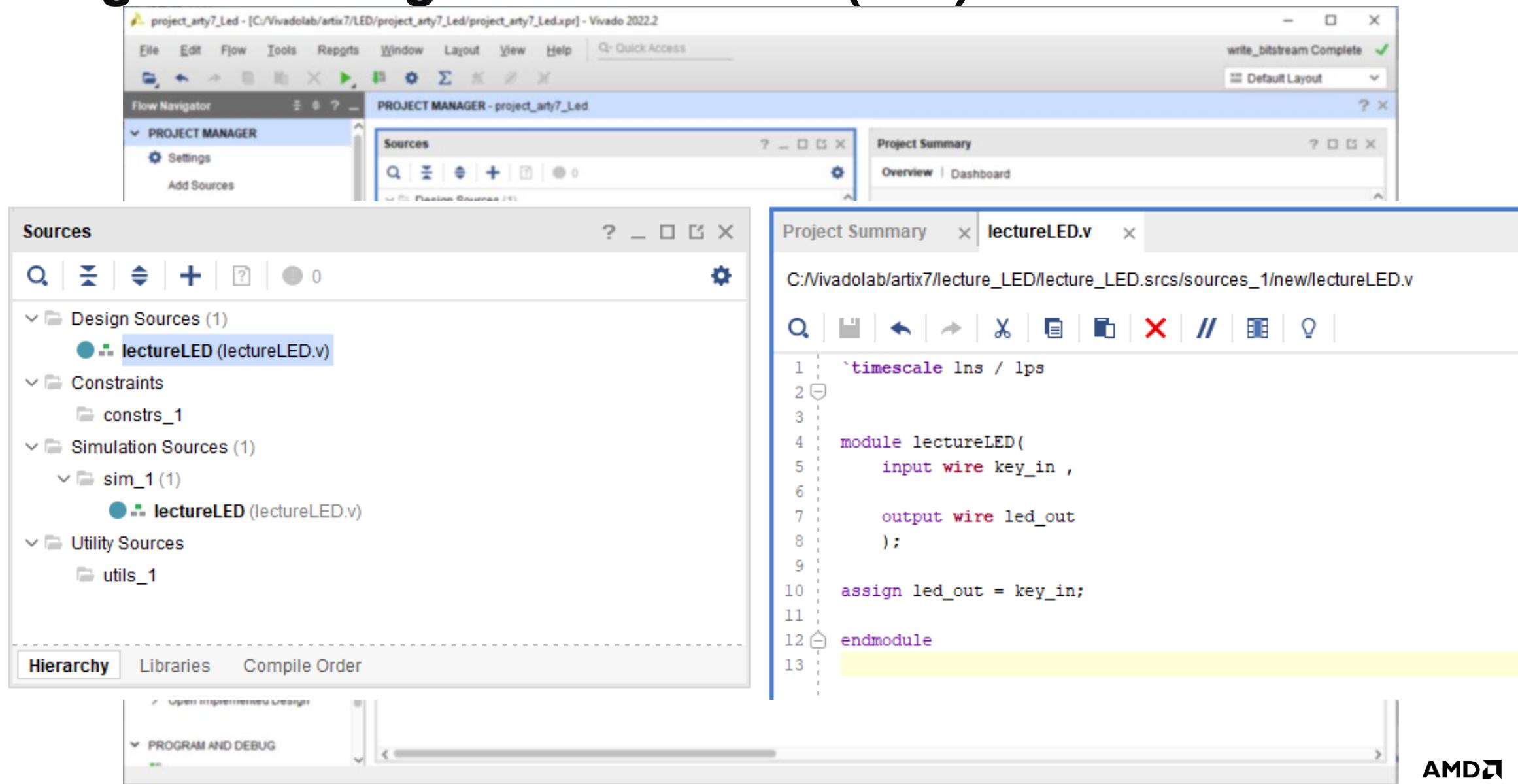
# Integrated Design Environment (IDE)



# Integrated Design Environment (IDE)



# Integrated Design Environment (IDE)

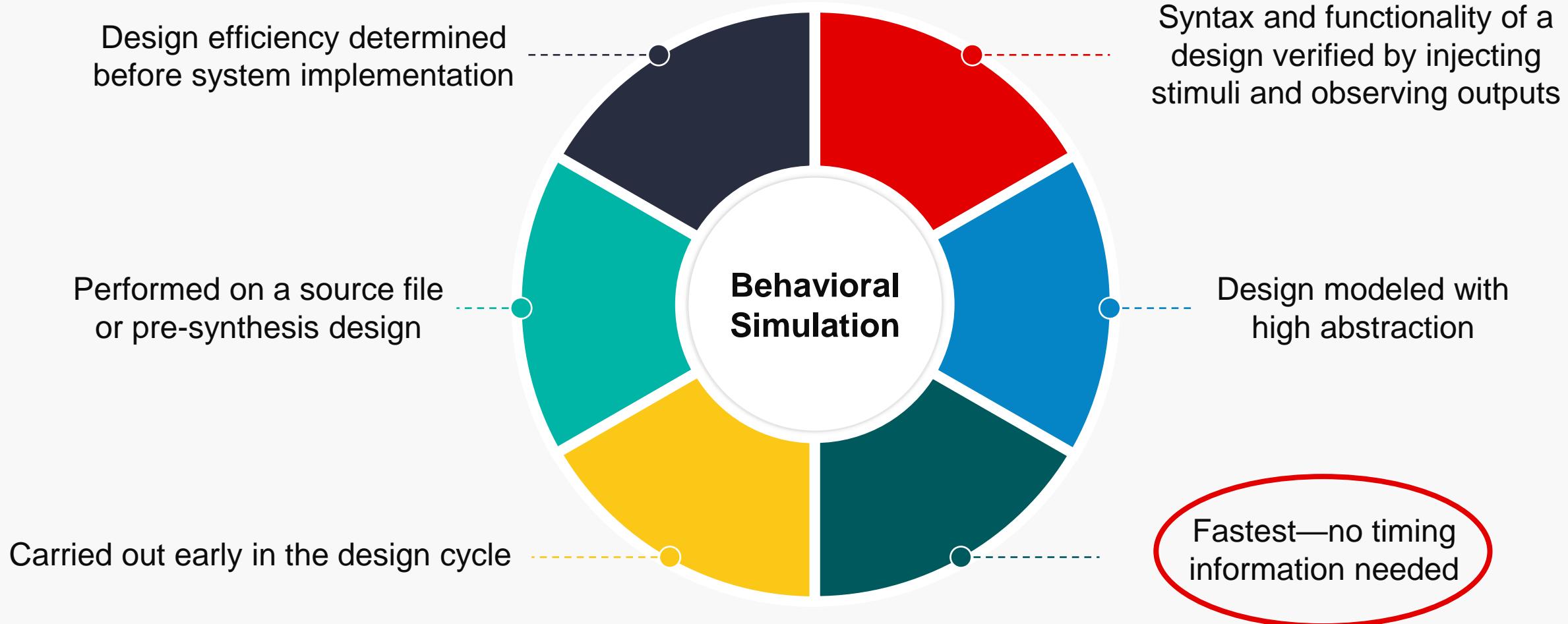




# Behavioral Simulation

2022.2

# Behavioral Simulation



# What is a Testbench?

Virtual environment for design verification



## Components of a Testbench

- HDL formatted
- Internal signals
- Unit under test instantiation
- Input stimulus including clock signals
- Output monitoring

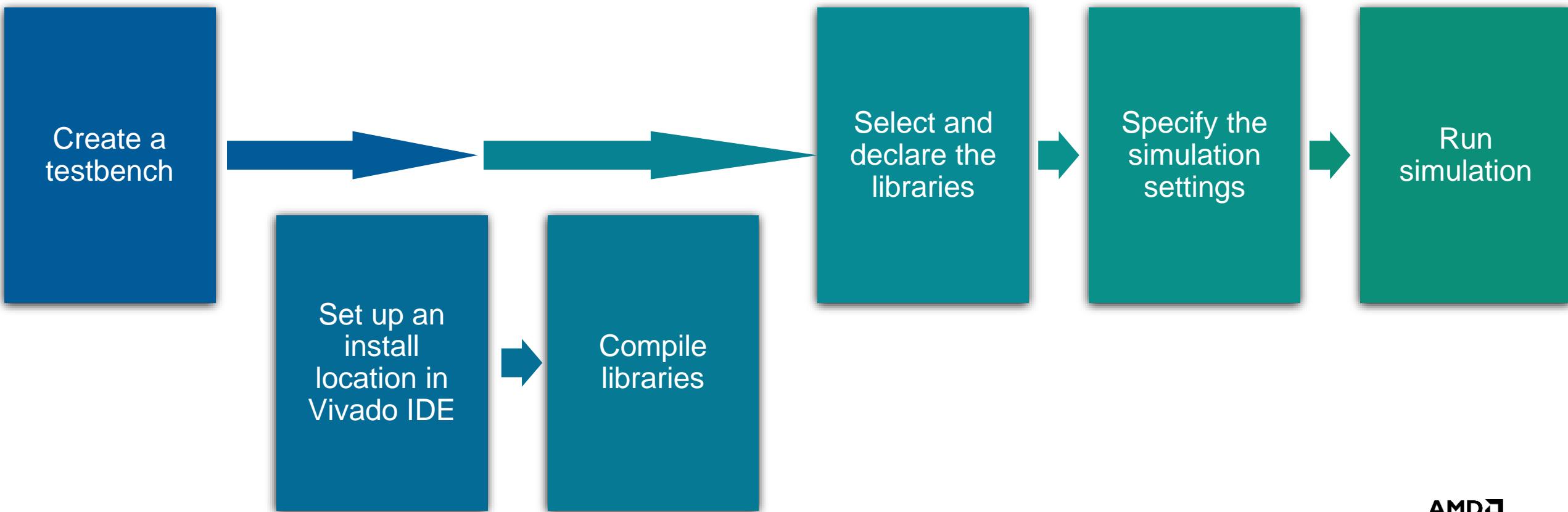


## Uses of a Testbench

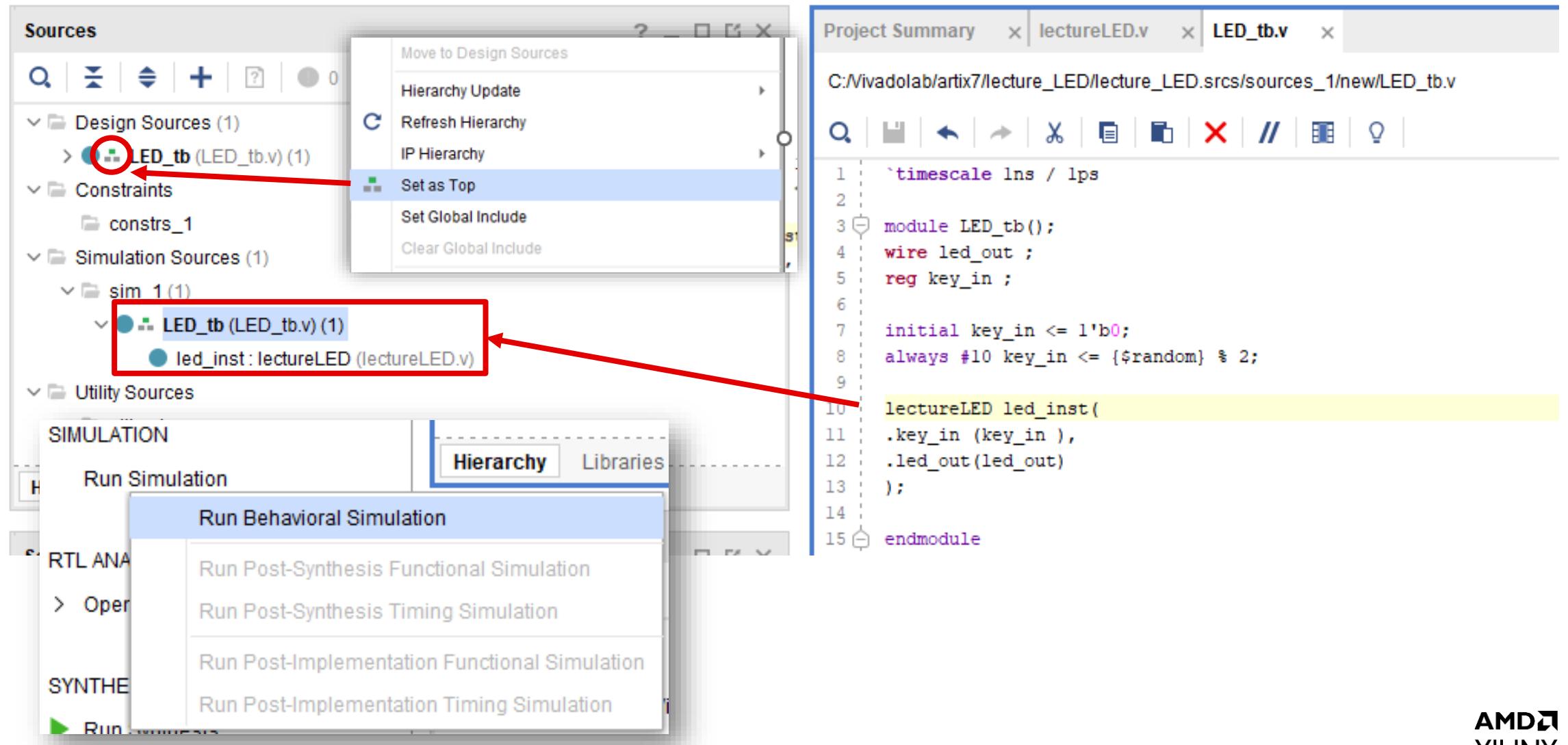
- Initializes the design
- Stimulates UUT (unit under test)
- Examines output and functionality

# Preparing for Simulation

Set up the following before performing the simulation:

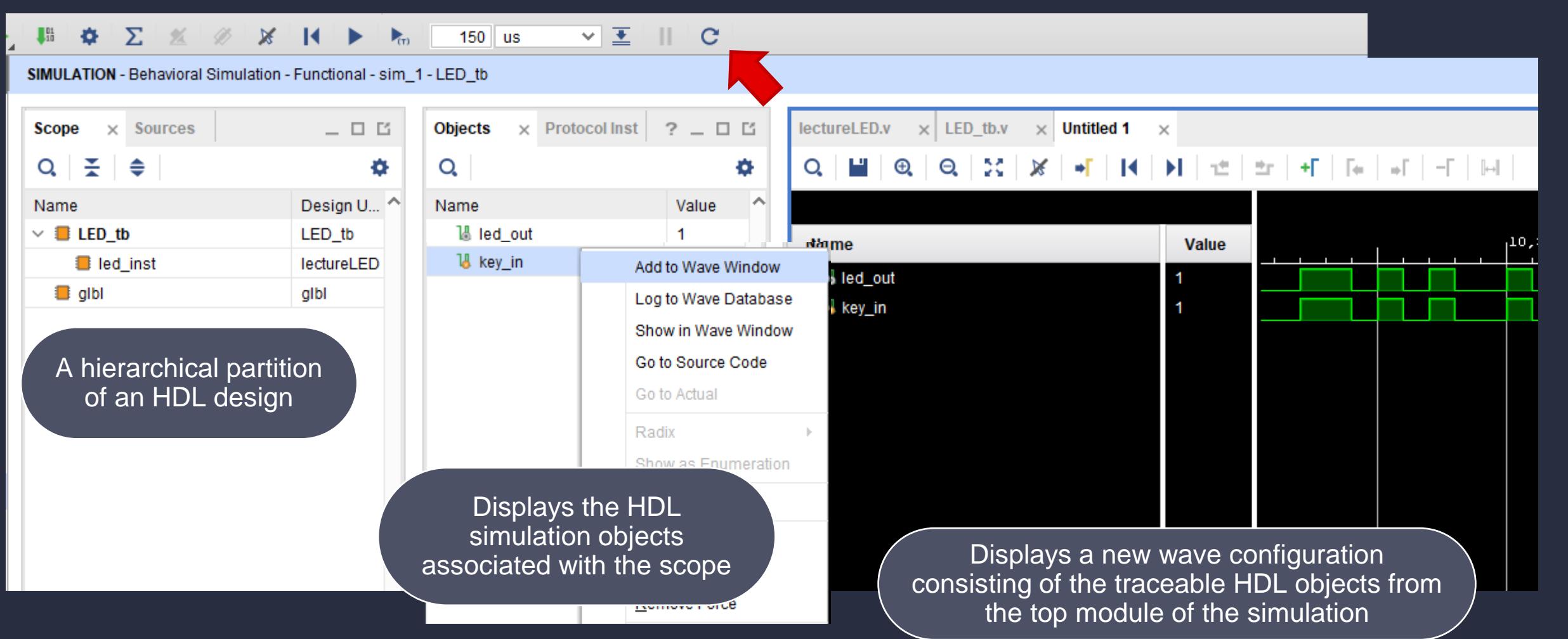


# Behavioral Simulation



# Simulation Waveform

Select **Simulation > Behavioral Simulation**



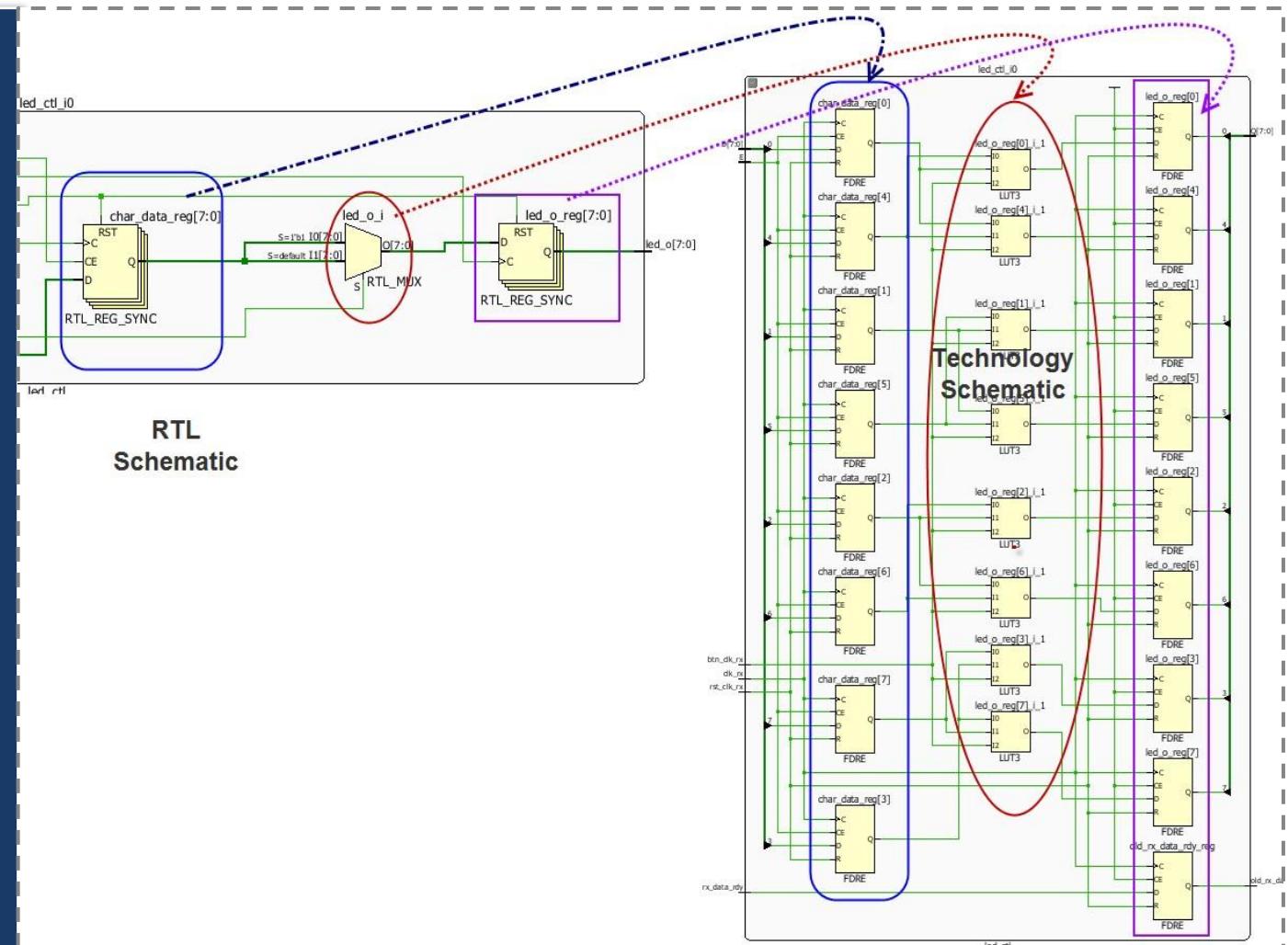


# Vivado Synthesis

2022.2

# Elaborated Design

- RTL-to-gate-level transformation of LUTs, flip-flops, block RAMs, etc.
- Timing driven and optimized for performance
- Verilog, VHDL, mixed HDL, and SystemVerilog support



# Synthesis Design

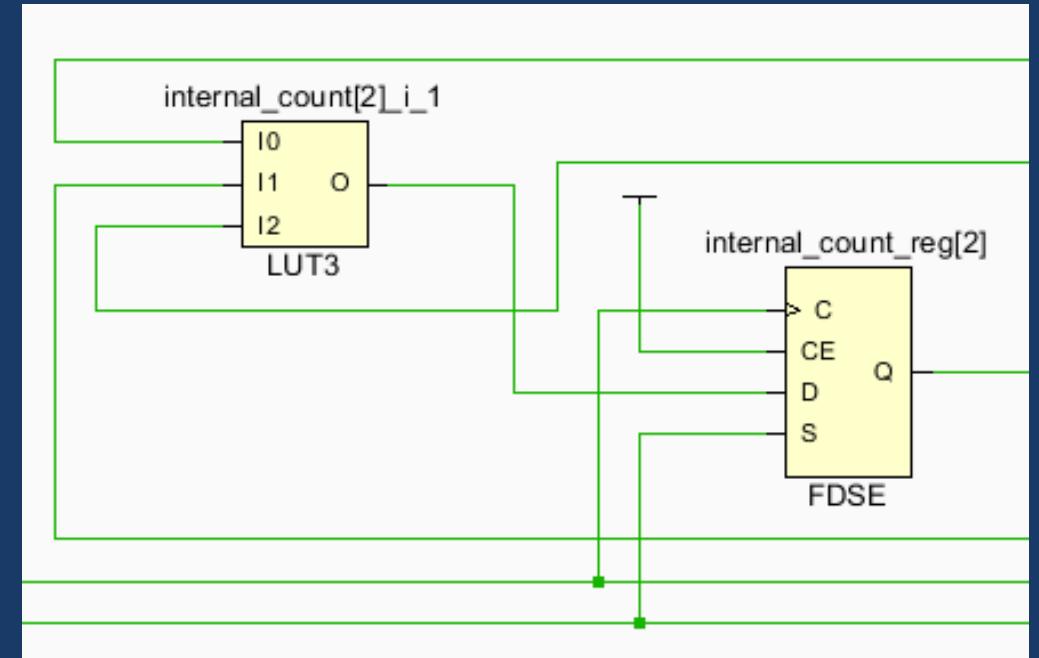
Representation of the design after synthesis

Interconnected netlist of hierarchical and basic elements (BELs)

- Instances of modules/entities
- Basic elements
  - LUTs, flip-flops, carry chain elements, wide MUXs
  - Block RAMs, DSP cells
  - Clocking elements (BUFG, BUFR, MMCM, etc.)
  - I/O elements (IBUF, OBUF, I/O flip-flops)

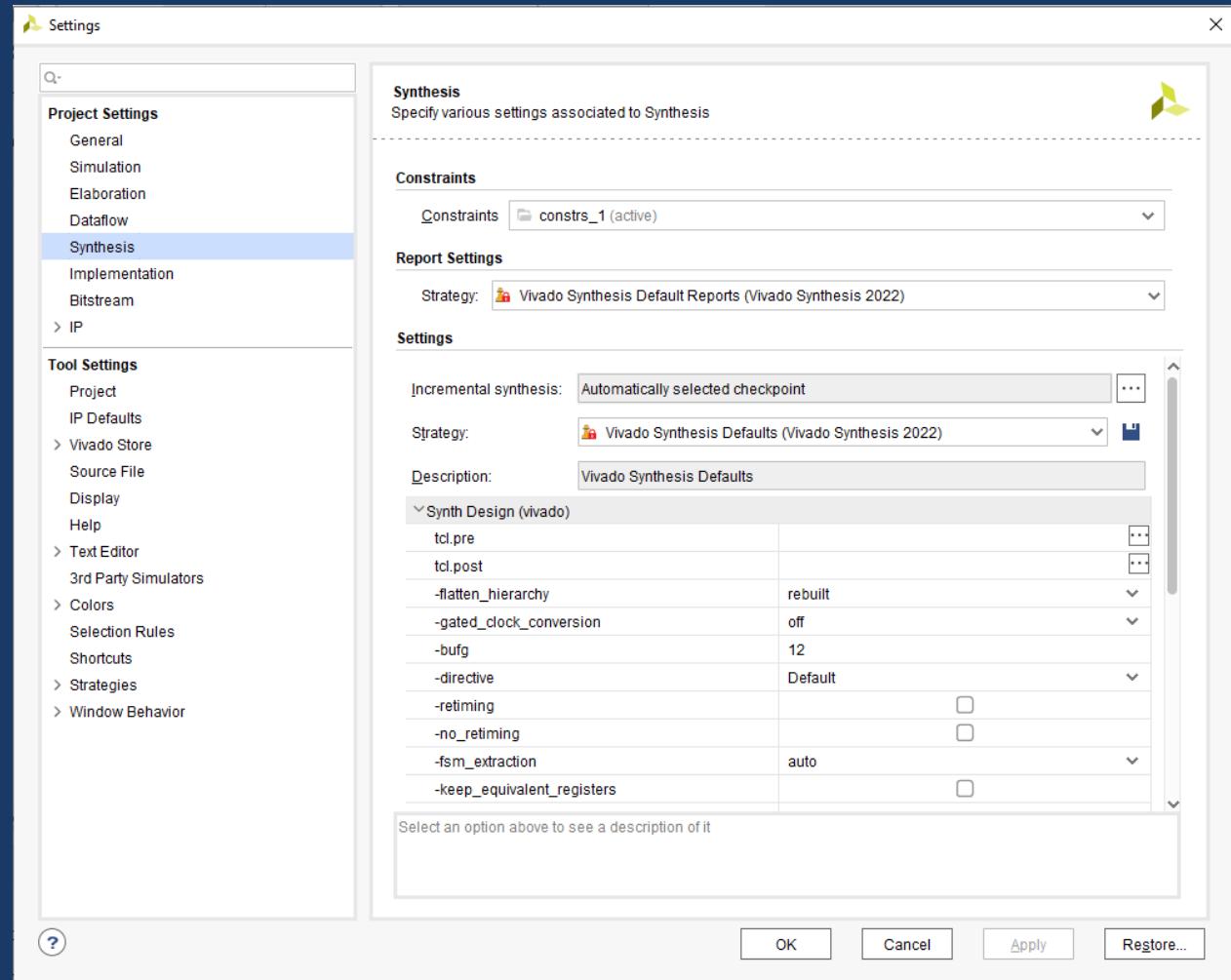
Object names match the elaborated netlist

Design timing and power estimates are possible



# Synthesis Options

- Access the synthesis settings by selecting **Project Manager > Settings > Synthesis** under the Flow Navigator for all synthesis-related settings
- Use of XDC constraints during synthesis improves synthesis results



# Synthesis Options

Many synthesis options that can help you obtain your performance and area objectives

## Flatten Hierarchy

Controls Vivado® synthesis hierarchy

## Fanout Limit

Specifies how many loads a signal must drive before replicating logic

## Directive

Replaces the -effort\_level option

## FSM Extraction

Re-encodes and optimizes the state machine design based on states and inputs

## Retiming

Automatically moves register stages to balance combinatorial delay

## Resource Sharing

Shares arithmetic operator resources with other functions

# Synthesis RTL Attributes Supported

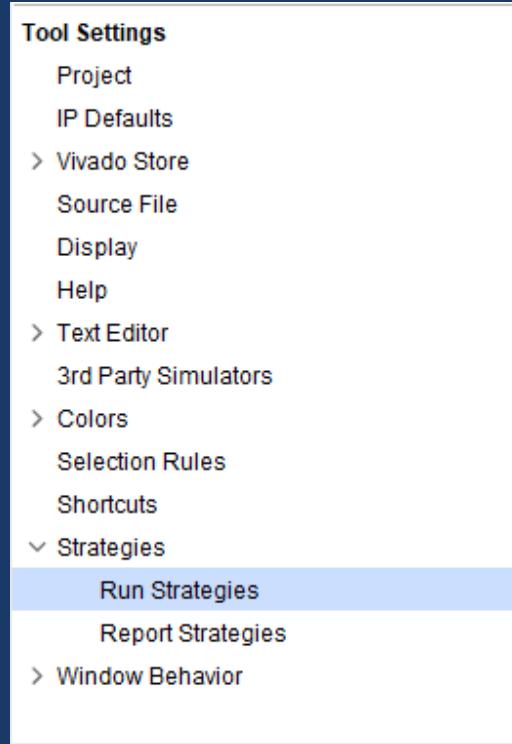
Attribute	Description
translate_off/_on	Tells the tool to ignore blocks of code
full_case	Tells that all possible case values are specified
parallel_case	Case statement should be built as a parallel structure
keep	Tells tool to keep the signal the attribute is placed on
keep_hierarchy	Used to prevent optimizations along the hierarchy boundaries
buffer_type	Tells tool what buffer type to use on an input
max_fanout	Tells the tool the limits for fanout on registers and signals
ram_style	Tells the tool how to infer memory
rom_style	Tells the tool how to infer ROM memory
use_dsp48	Tells the tool how to deal synthesis arithmetic structures
black_box	Turns a whole level of hierarchy off and enables synthesis to create a black box for that module/entity
gated_clock	Allows the conversion of gated clocks; must be enabled
shreg_extract	Tells the tool on whether to infer structures
iob	Not a synthesis attribute but is passed to the implementation tool indicating if a register should be in IOB

# Synthesis RTL Attributes Supported

Attribute	Description
async_reg	Tells the tool that a register can receive asynchronous data at D input
srl_style	Specifies how SRL is inferred in design
clock_buffer_type	Specifies a buffer other than the (default) BUFG for synthesis
dont_touch	Similar to KEEP attribute, use in place of KEEP. Attribute is forward-annotated to place & route
fsm_encoding	Specifies a specific FSM encoding scheme: one_hot, sequential, johnson, gray, auto (default), none
fsm_safe_state	Place on state machine state registers, used to define a safe state in the machine
IOB	Not a synthesis attribute, used by Vivado implementation. Specifies if a register is packed into IOB
io_buffer_type	Instructs the tool to not automatically infer I/O buffers for a specific top-level port.
MARK_DEBUG	Specifies that a net to be marked for debug.

# Synthesis Strategy

- Specifies the predefined strategies or lets you create your own strategy by modifying the options for the selected strategy



The screenshot shows the 'Run Strategies' configuration dialog. The 'Flow' dropdown is set to 'Vivado Synthesis 2022'. The 'User Defined Strategies' section shows a list of available strategies, with 'Vivado Synthesis Defaults' selected. The 'Options' section on the right lists various synthesis parameters with their current values.

Strategies > Run Strategies  
Choose a strategy based on various settings.

Flow: Vivado Synthesis 2022

User Defined Strategies

Vivado Strategies

- Vivado Synthesis Defaults
- Flow\_AreaOptimized\_high
- Flow\_AreaOptimized\_medium
- Flow\_AreaMultThresholdDSP
- Flow\_AlternateRoutability
- Flow\_PerfOptimized\_high
- Flow\_PerfThresholdCarry
- Flow\_RuntimeOptimized

Name: Vivado Synthesis Defaults  
Description: Vivado Synthesis Defaults

Options

Synth Design (vivado)	
tcl.pre	[...]
tcl.post	[...]
-flatten_hierarchy	rebuilt
-gated_clock_con...	off
-bufg	12
-directive	Default
-retiming	<input type="checkbox"/>
-no_retiming	<input type="checkbox"/>
-fsm_extraction	auto
-keep_equivalent...	<input type="checkbox"/>
-resource_sharing	auto
-control_set_opt...	auto
-no_lc	<input type="checkbox"/>
-no_srextract	<input type="checkbox"/>

# Block-Level Synthesis

Design involves hierarchies, either user created or tool generated

Allows different synthesis strategies for modules or instances

Allows optimization based on area, performance, and routing in addition to logic

Allows overriding synthesis settings on a module

## Block-Level Synthesis Example

```
set_property BLOCK_SYNTH.RETIMING 1 [get_cells U1]
set_property BLOCK_SYNTH.STRATEGY {AREA_OPTIMIZED} [get_cells U2]
set_property BLOCK_SYNTH.STRATEGY {AREA_OPTIMIZED} [get_cells U3]
set_property BLOCK_SYNTH.STRATEGY {DEFAULT} [get_cells U3/inst1]
```

# Open Synthesis Design

Flow Navigator

- SYNTHESIS
  - Run Synthesis
  - Open Synthesized Design
    - Constraints Wizard
    - Edit Timing Constraints
    - Set Up Debug
    - Report Timing Summary
    - Report Clock Networks
    - Report Clock Interaction
    - Report Methodology
  - Report DRC
  - Report Noise
  - Report Utilization
  - Report Power
  - Schematic

Netlist

Schematic

Tcl Console | Messages | Debug | 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	Pull Type	Off-Chip Termination	IN_TERM
All ports (2)															
Scalar ports (2)															
IN								default (LVCMS18)	1.800						
OUT								default (LVCMS18)	1.800	12					



# I/O Pin Planning

2022.2

# I/O Planning



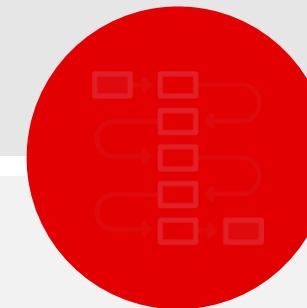
An FPGA design does not exist in a “ready-to-use” environment

- It must physically interface with other system components



The Vivado Design Suite provides simple methods for defining the FPGA-to-system interface

- Use the I/O Planner to assign I/O pins in your design



I/O planning can be done at different stages of the design flow

- Elaborated, synthesized, and implemented design

# I/O Planning Stages

1

I/O Planning  
before RTL

2

I/O Planning  
with RTL

3

I/O Planning on  
Netlist and  
Implemented  
Design

When creating a new project from the Getting Started page, you have the option to create an I/O planning project without RTL

- This allows you to test pin assignments
  - 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

# I/O Planning Stages

1

I/O Planning  
before RTL

2

I/O Planning  
with RTL

3

I/O Planning on  
Netlist and  
Implemented  
Design

This flow is used if you already have RTL

- Open the elaborated design by clicking **Open Elaborated Design**
- Open the I/O Planner by selecting the **I/O Planning** view
- The Vivado Design Suite allows you to check port assignments, I/O standards, and other I/O planning details
  - Running the Design Rule Checks (DRCs) is critical

# I/O Planning Stages

**1**

**I/O Planning  
before RTL**

**2**

**I/O Planning  
with RTL**

**3**

**I/O Planning on  
Netlist and  
Implemented  
Design**

Netlist I/O planning flow is used if you have a synthesized project or netlist

At this stage, more details about the design will be available

- Automatic I/O placement or interactive placement can be done

With the implemented design, the final I/O pin planning needs to be validated

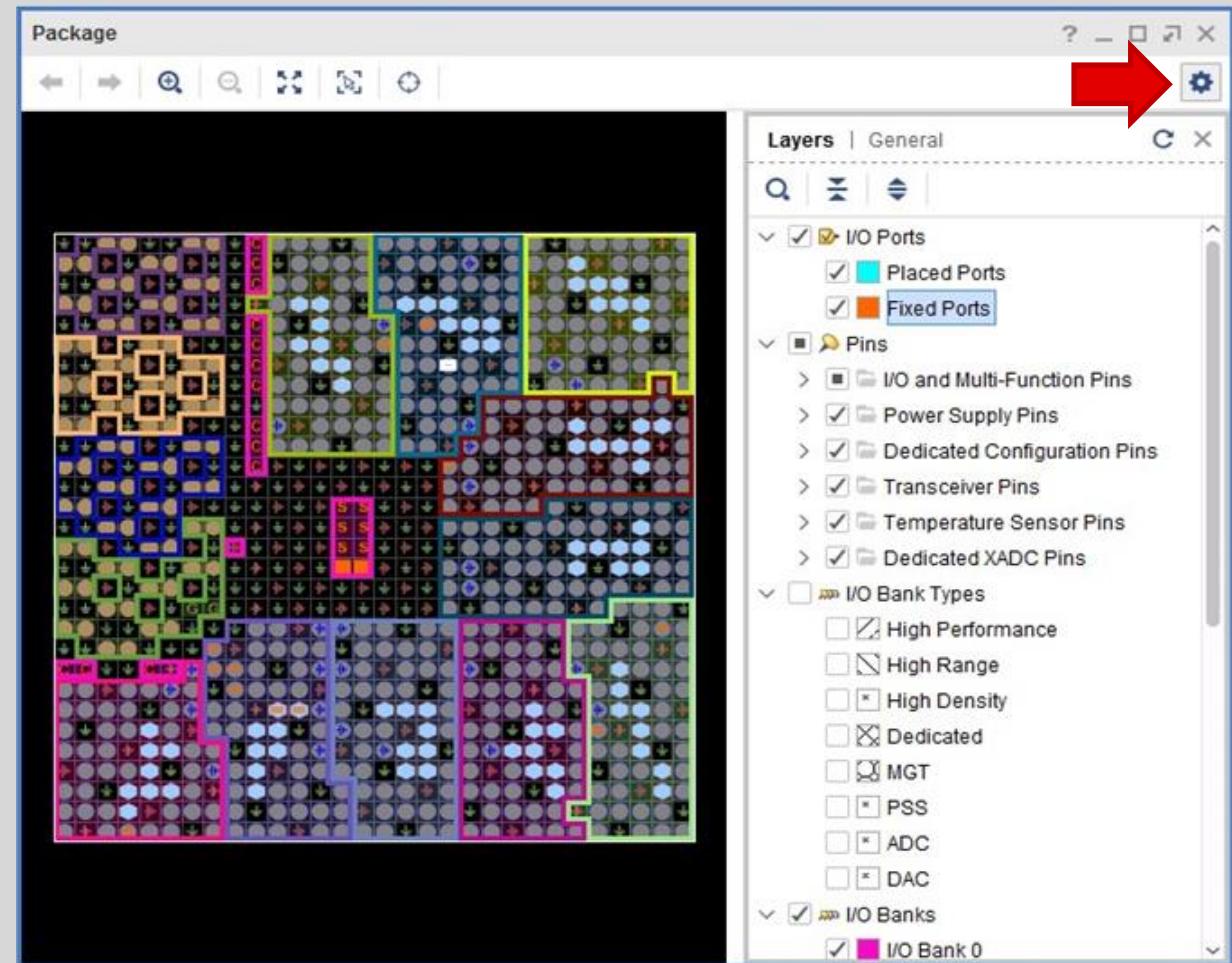
# Package View

Clock-capable pins are displayed as hexagon pins

VCC and GND pins show as red and green square pins

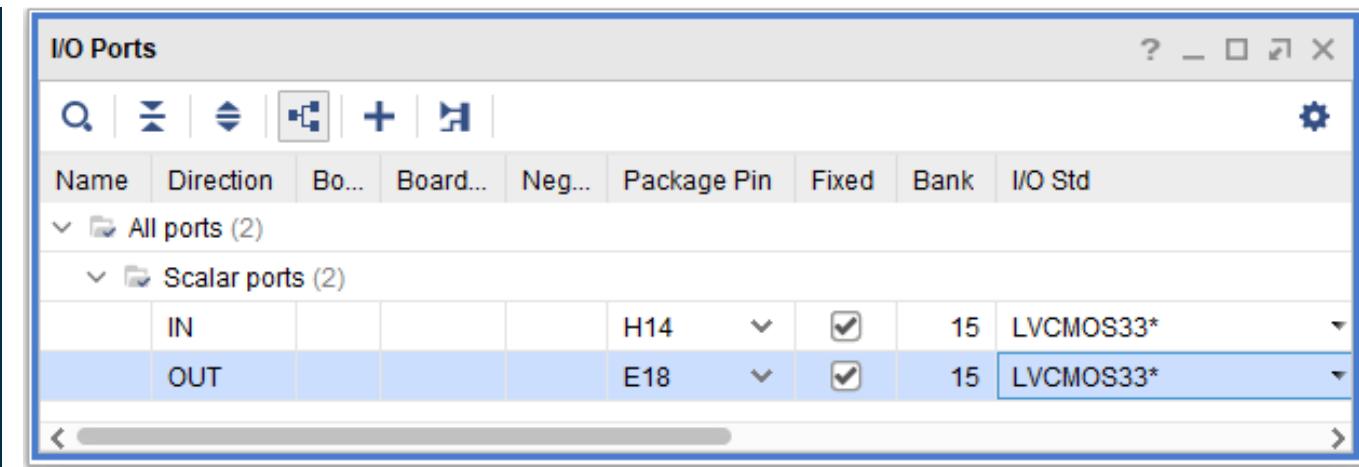
Colored areas between the pins display the I/O banks

Clock-capable pins ( ), VCC ( ), GND ( ), no connection ( )



# I/O Ports View

- Displays all I/O ports defined in the project
- Groups buses into expandable folders
- Can be displayed as groups of buses and interfaces or as a list
- Most I/O port assignment is initiated from this view
- Icons indicate I/O port direction and status



Name	Direction	Bo...	Board...	Neg...	Package Pin	Fixed	Bank	I/O Std
All ports (2)								
Scalar ports (2)								
	IN				H14	▼	<input checked="" type="checkbox"/>	15 LVCMOS33*
	OUT				E18	▼	<input checked="" type="checkbox"/>	15 LVCMOS33*

## Interactively Placing I/O Ports

Place by selecting the I/O ports, buses, or interfaces from the I/O Ports window and right-clicking:

- **Place I/O Ports in an I/O Bank** – Places I/O ports in selected I/O banks
- **Place I/O Ports in Area** – Draws a rectangle around the desire pins or pads
- **Place I/O Ports Sequentially** – Selects a pin or pad for each I/O port individually

The placement mode remains active until all selected I/O ports are placed or until the **ESC** key is pressed

Assignment direction is dictated by the first selected I/O pin or I/O pad

# DRCs and SSN Analysis

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

Define alternate compatible parts

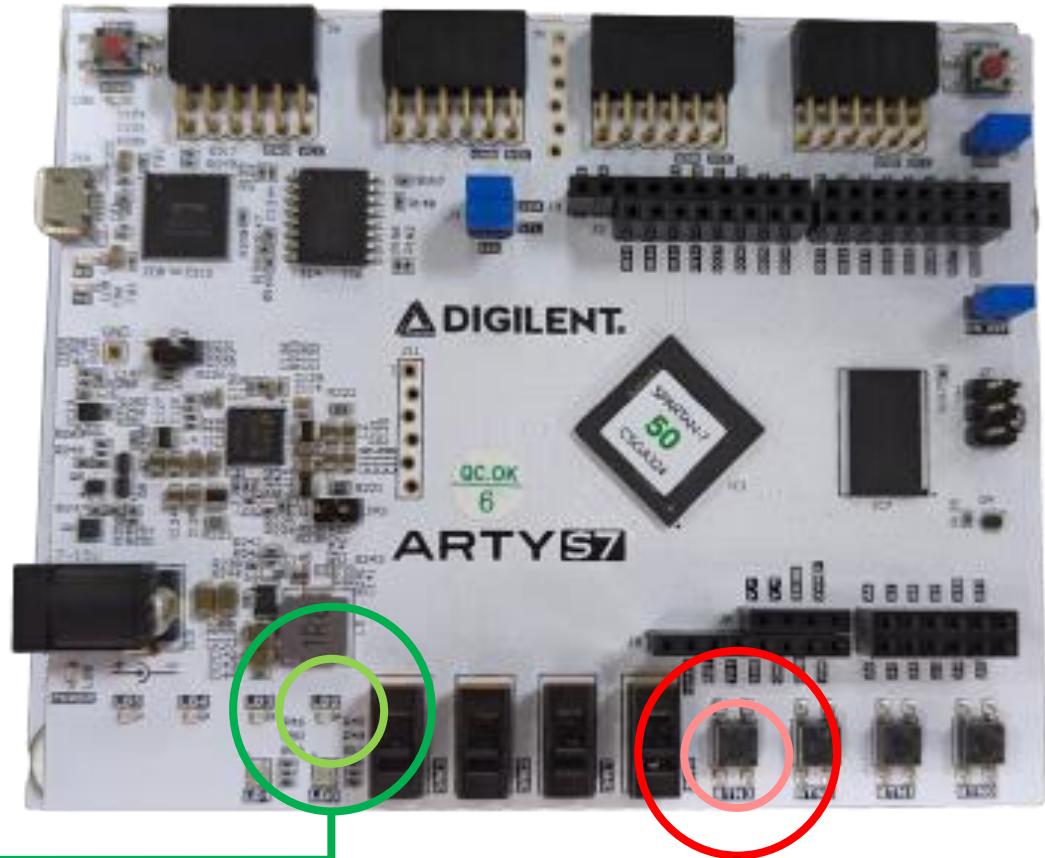
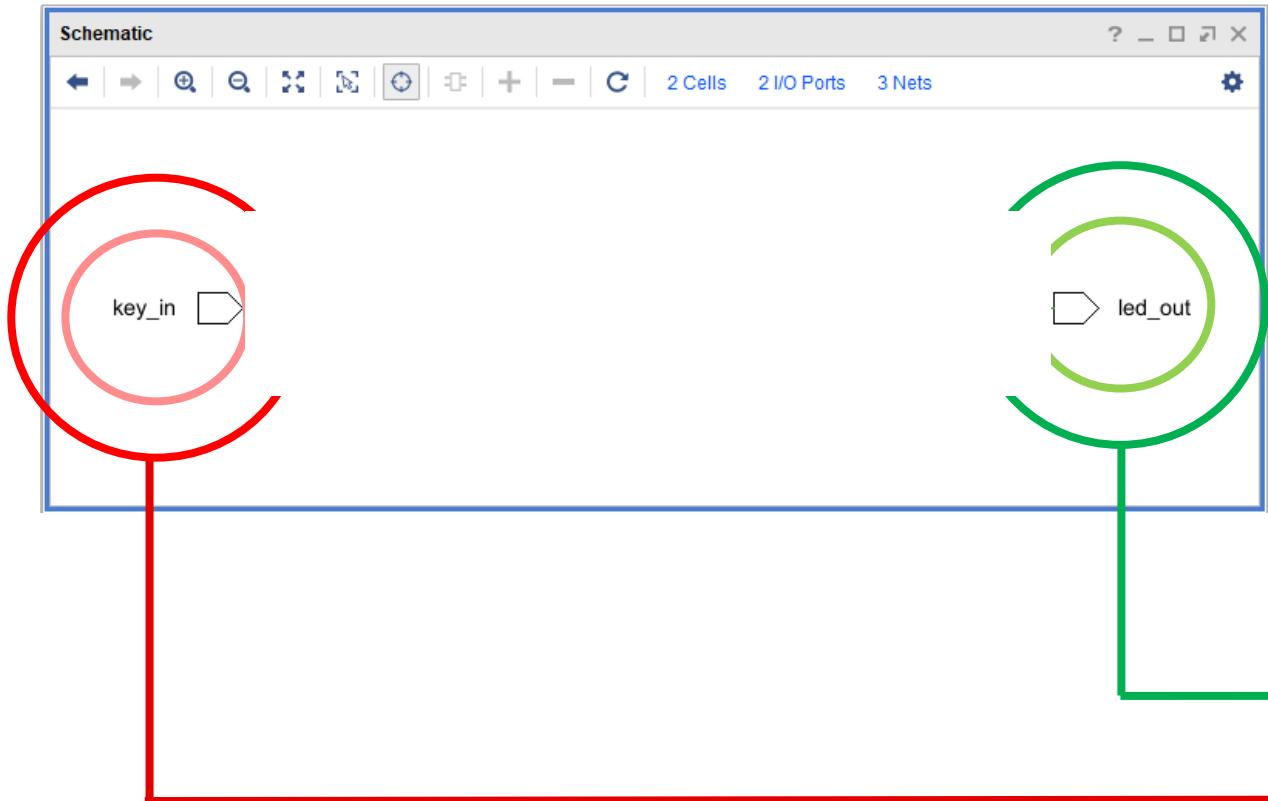
Ensure pin assignment works across multiple parts with a common package



# Vivado Implementation

2022.2

# Vivado Implementation



# Vivado Implementation

Consists of four mandatory and one optional sub-processes

**Logic Optimization**

**Power Optimization**

**Placement**

**Physical Optimization**

**Routing**

# Logic Optimization

## Logic Optimization

## Power Optimization

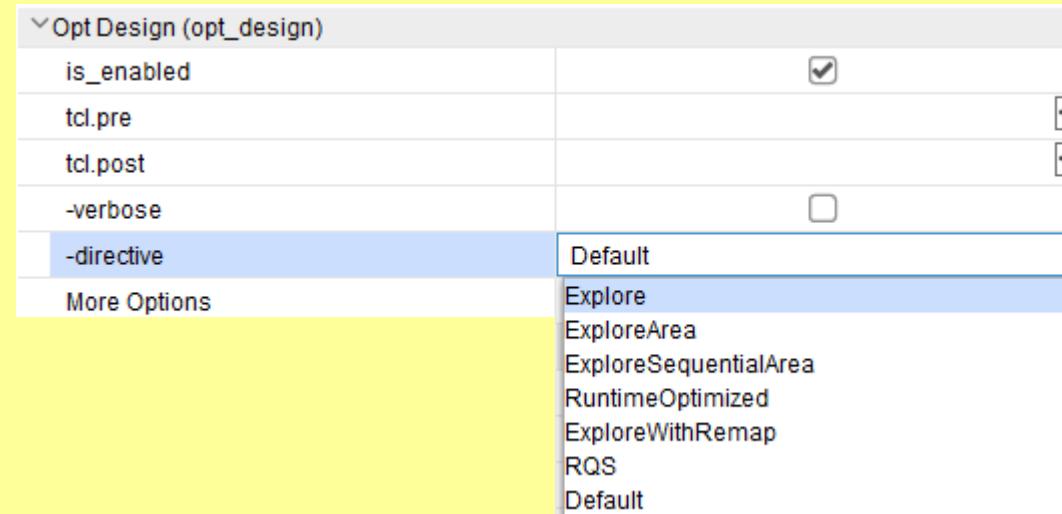
## Placement

## Physical Optimization

## Routing

### Logic optimization: opt\_design

- Ensures efficient design before placement
- Performs constant propagation (unnecessary and redundant logic removal)



# Power Optimization

Logic Optimization

Power Optimization

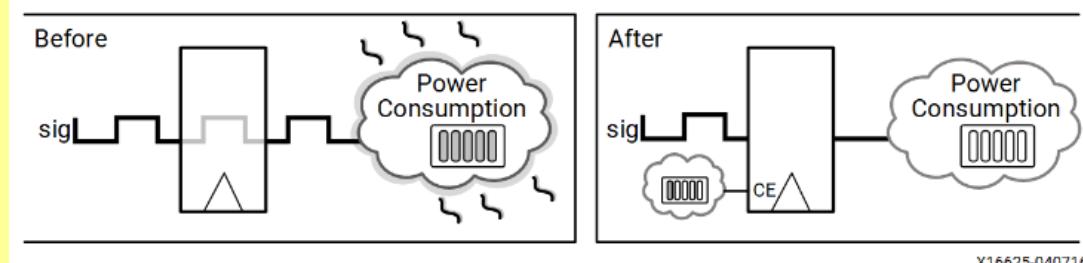
Placement

Physical Optimization

Routing

## Power optimization: power\_opt\_design (optional)

- Optimizes dynamic power using AMD Xilinx-intelligent clock gating solutions.
- Power can reduce dynamic power by up to 30%.
- Algorithm performs analysis on all portions of the design.



▼ Power Opt Design (power_opt_design)	
is_enabled	<input type="checkbox"/>
tcl.pre	<input type="checkbox"/>
tcl.post	<input type="checkbox"/>
More Options	<input type="checkbox"/>

# Placement

Logic Optimization

Power Optimization

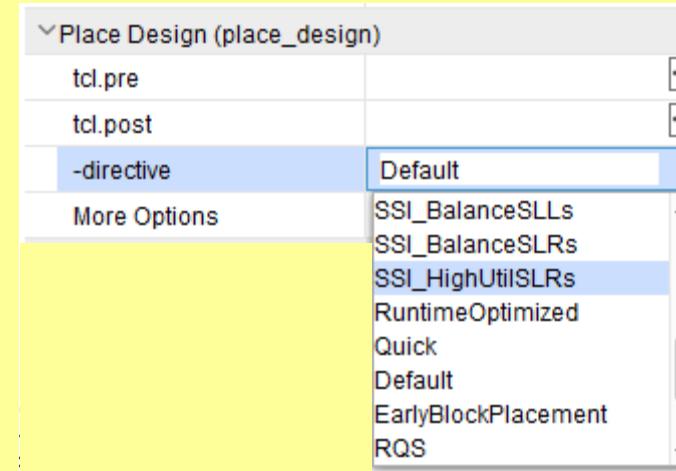
Placement

Physical Optimization

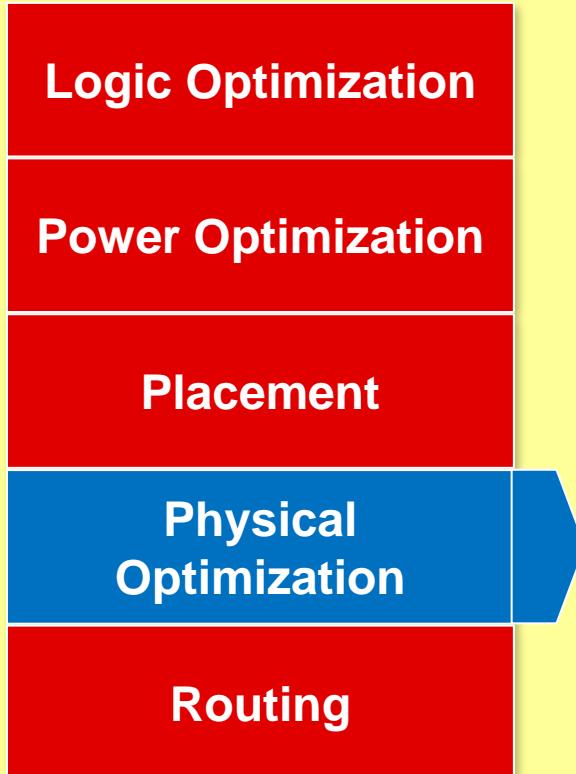
Routing

## Placement: place\_design

- Clock and I/O cell placement done before placing CLBs  
Clock and I/O cells are placed concurrently because they are often related through complex placement rules
- Timing-driven and wire length-driven placement of CLBs

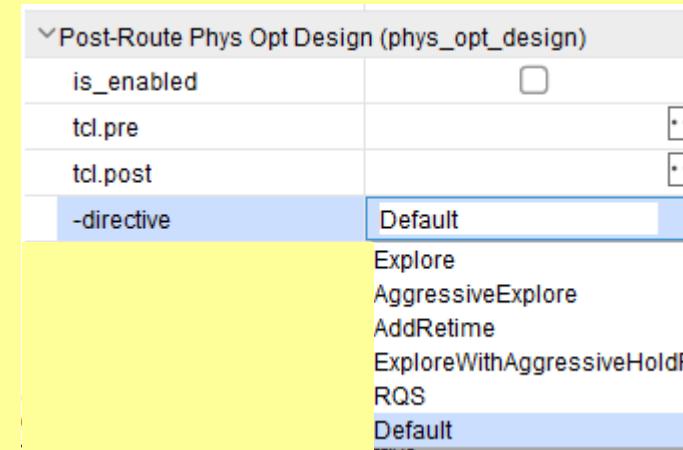


# Physical optimization

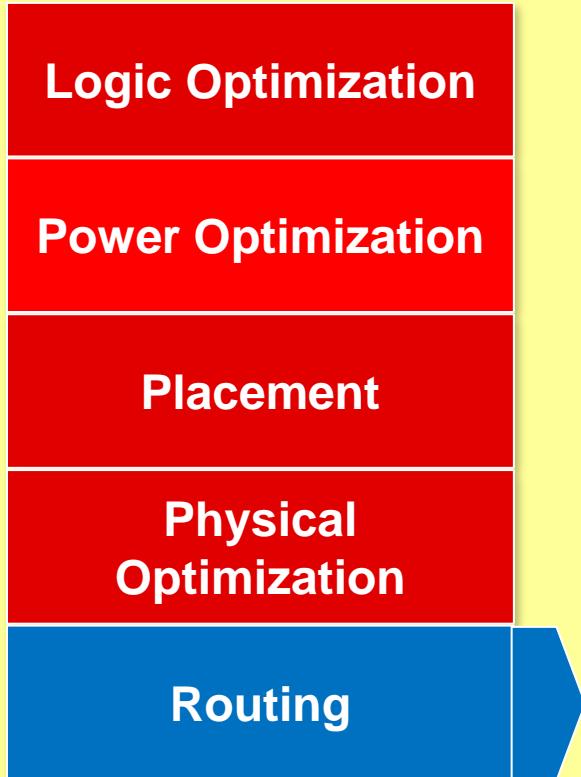


## Physical optimization: phys\_opt\_design

- Available at post-place\_design and post-route\_design
- Performs timing-driven optimization on timing-critical paths
- Net replication replicates logic driving timing-critical nets

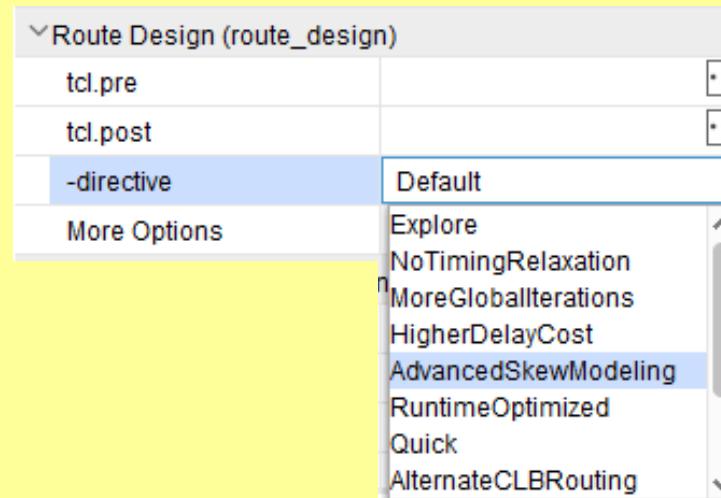


# Routing



## Routing: route\_design

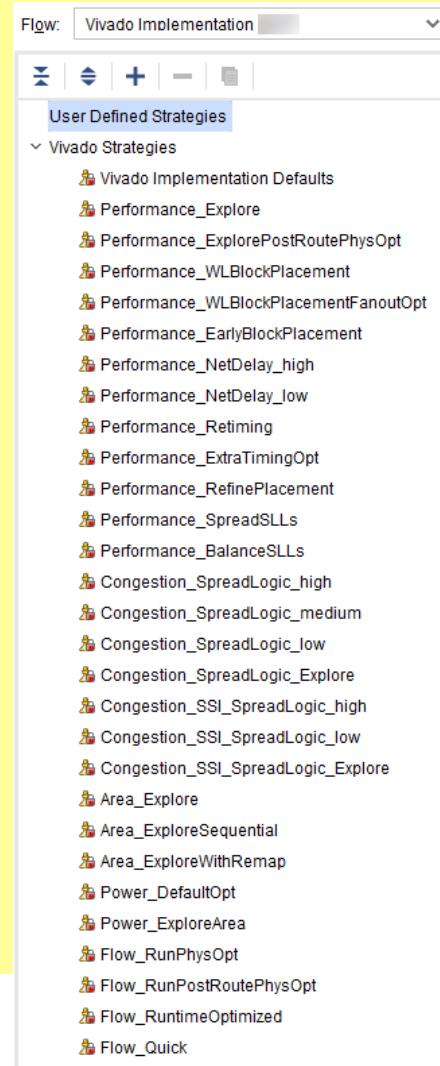
- Runs in timing-driven mode by default to fix hold violations
- Can be run in two modes:  
Normal mode (default) and Re-entrant mode



# Implementation Strategies

- Predefined strategies or create your own by changing the strategy options
- Runtime implementation is the fastest
- Performance Explore is the longest

Category	Purpose
Performance	Improve design performance
Area	Reduce LUT count
Power	Add full power optimization
Flow	Modify flow steps
Congestion	Reduce congestion and related problems

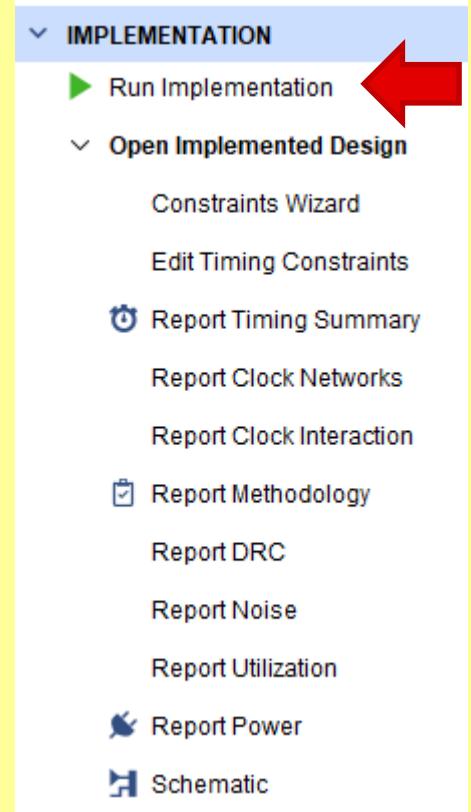


# Run Implementation

Accessed through the Flow Navigator by selecting Open Implemented Design

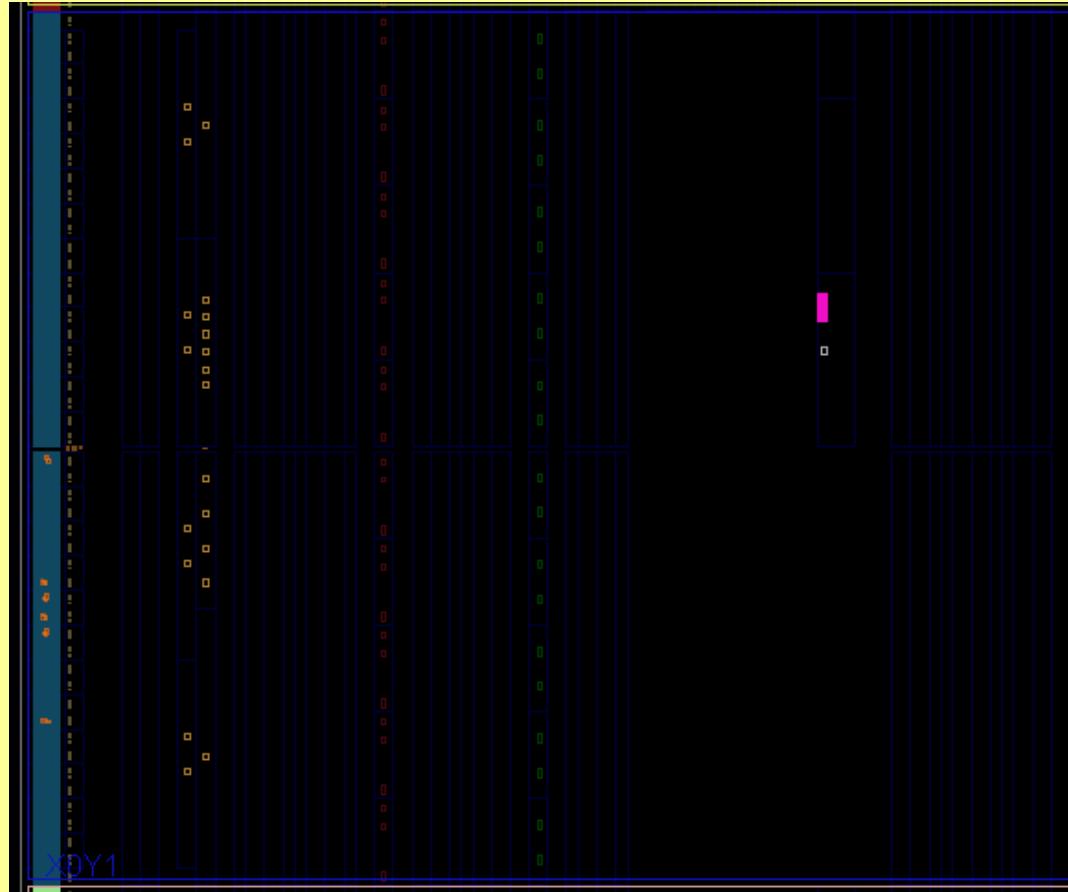
Design representation during and after implementation

- Like synthesized design
- Cells have locations, and nets are mapped to specific routing channels
- Allows applying physical and timing constraints for implementation, placement and routing

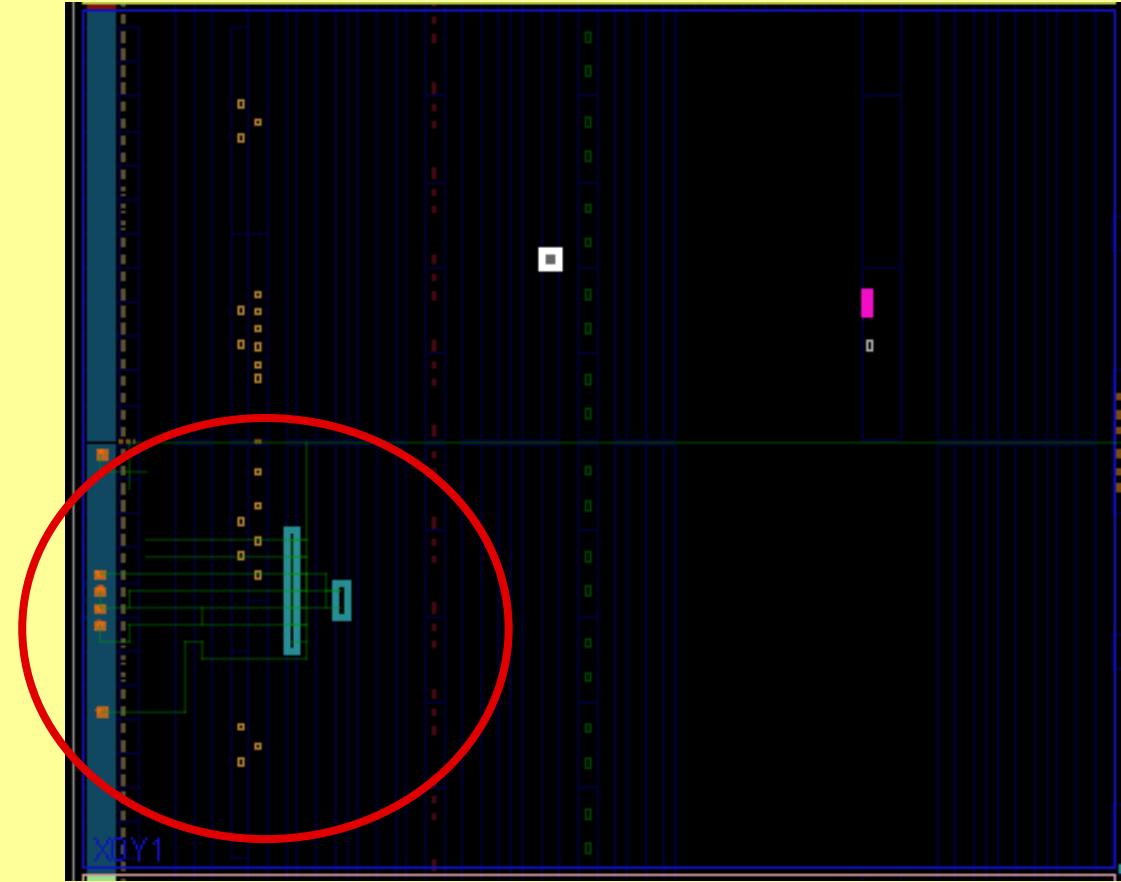


# After Implementation

Synthesis stage



Implementation stage



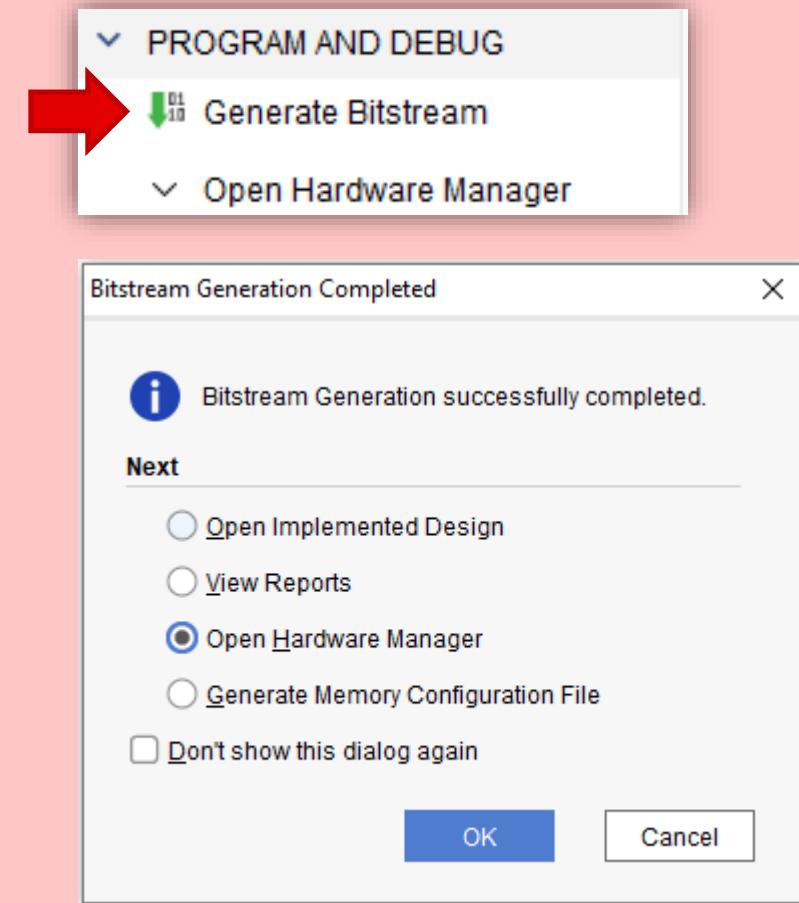
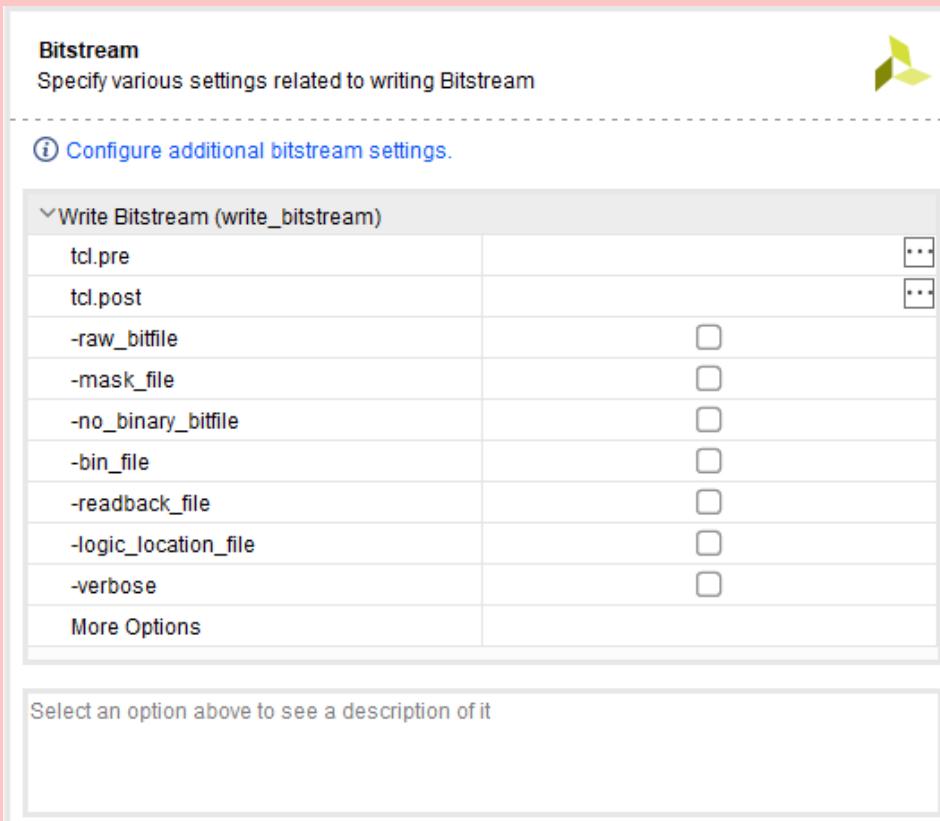


# Vivado Bitstream Generation

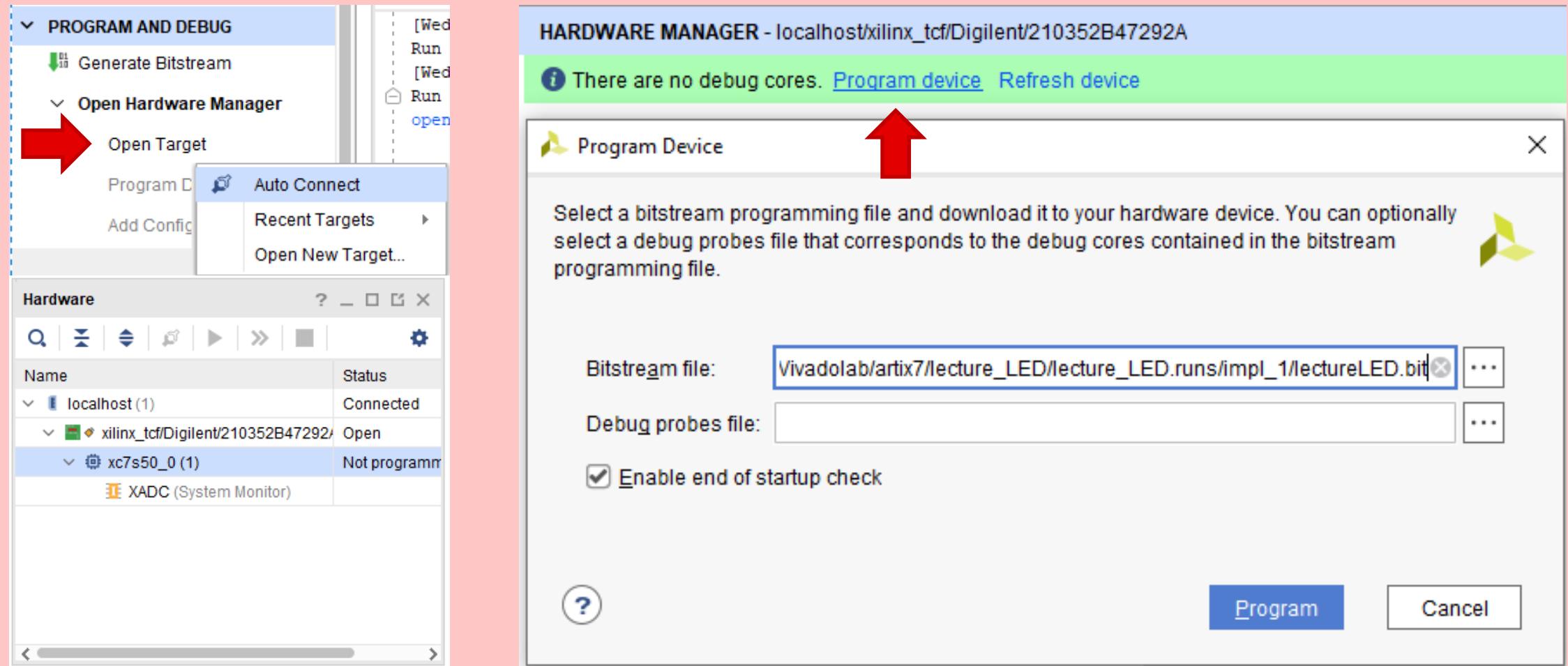
2022.2

# Bitstream Generation

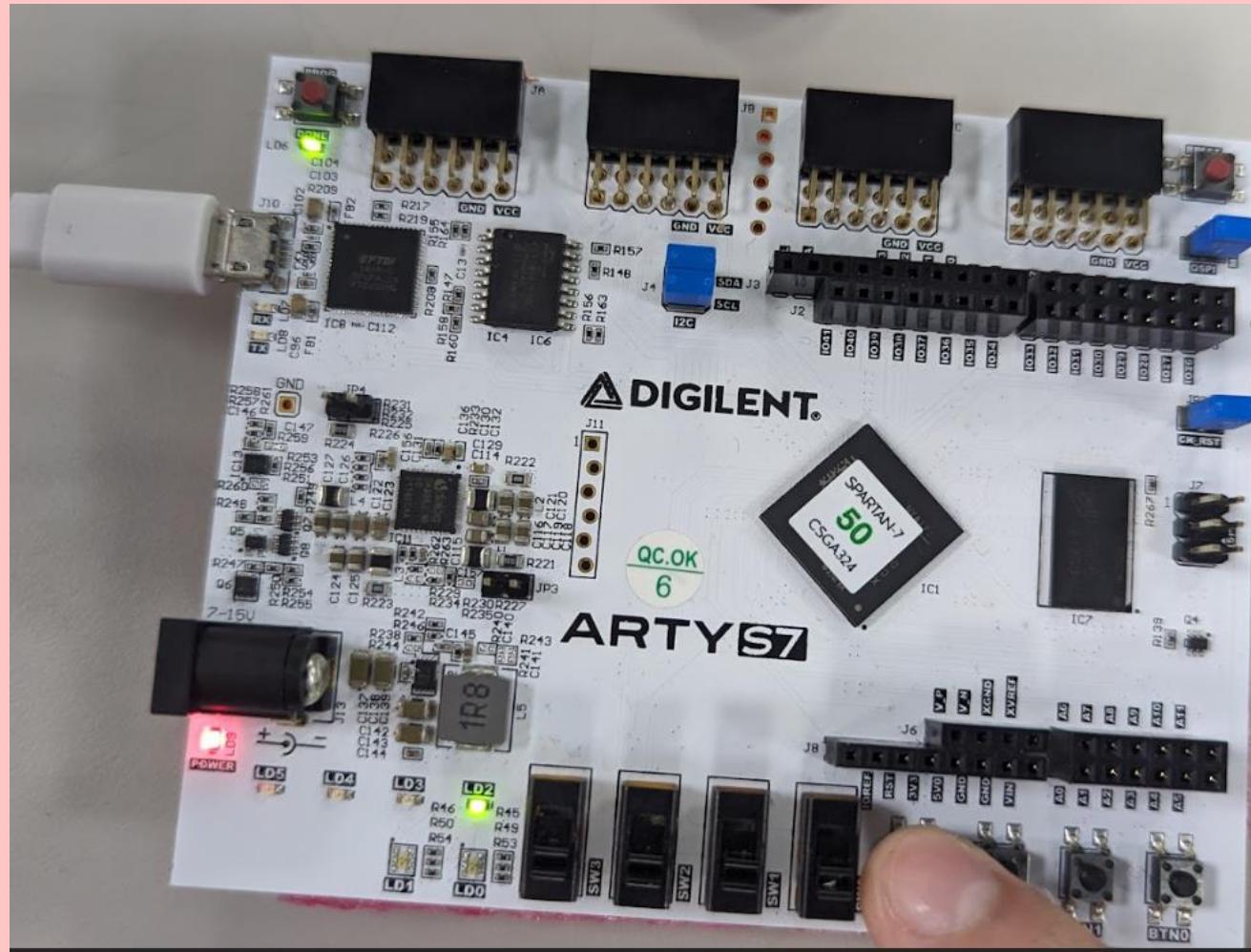
- Ensure the settings are correct



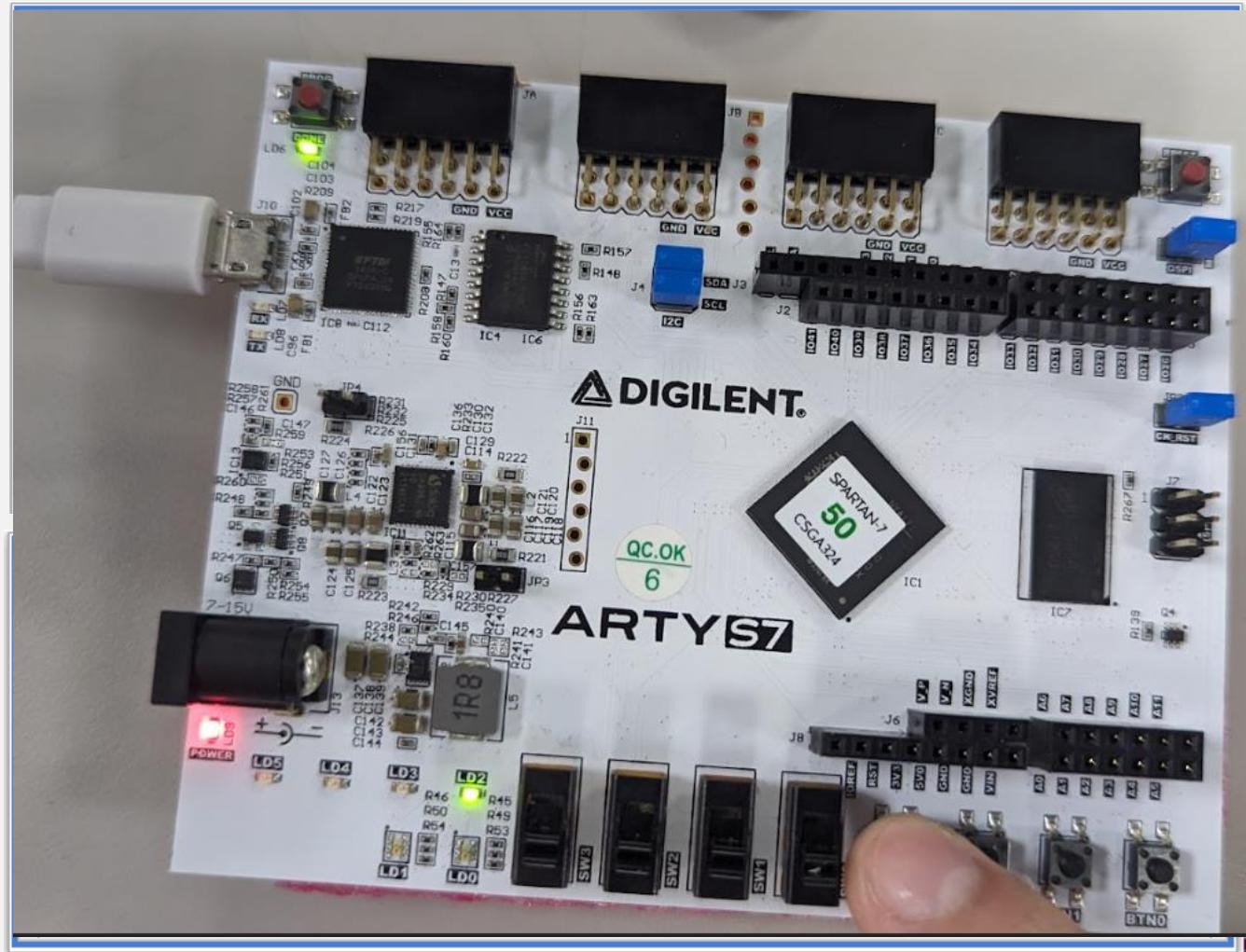
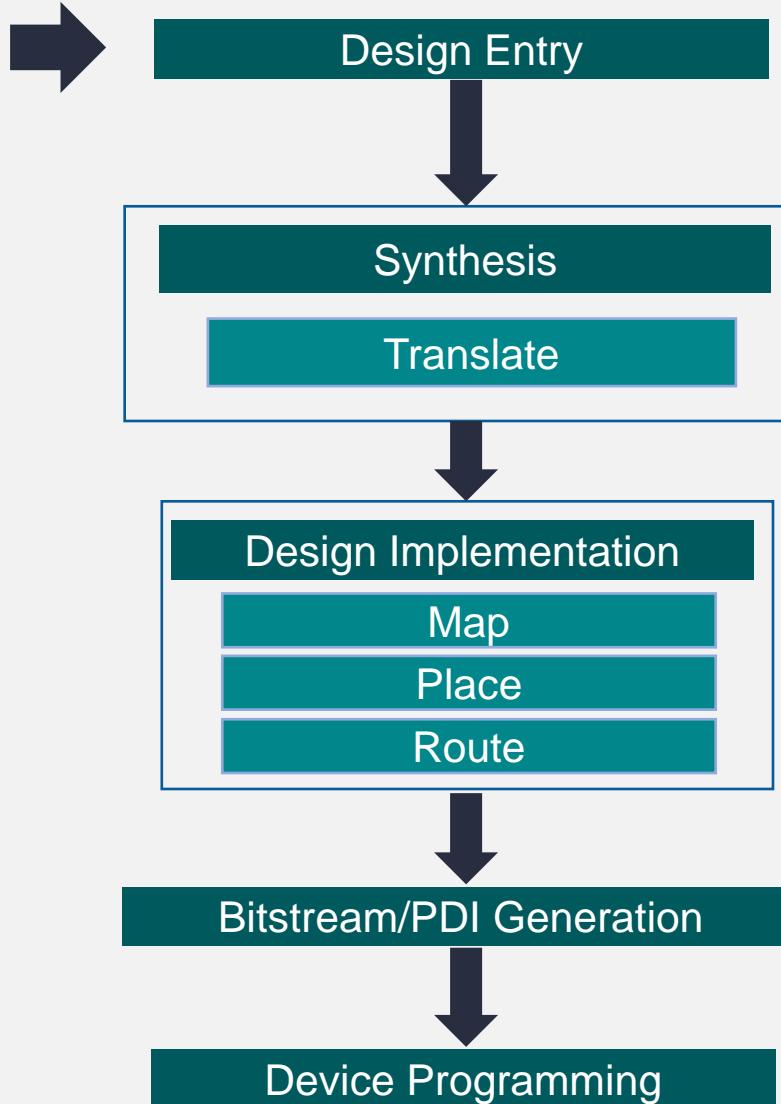
# Hardware Manager



# Board Test



# Traditional RTL Design Flow



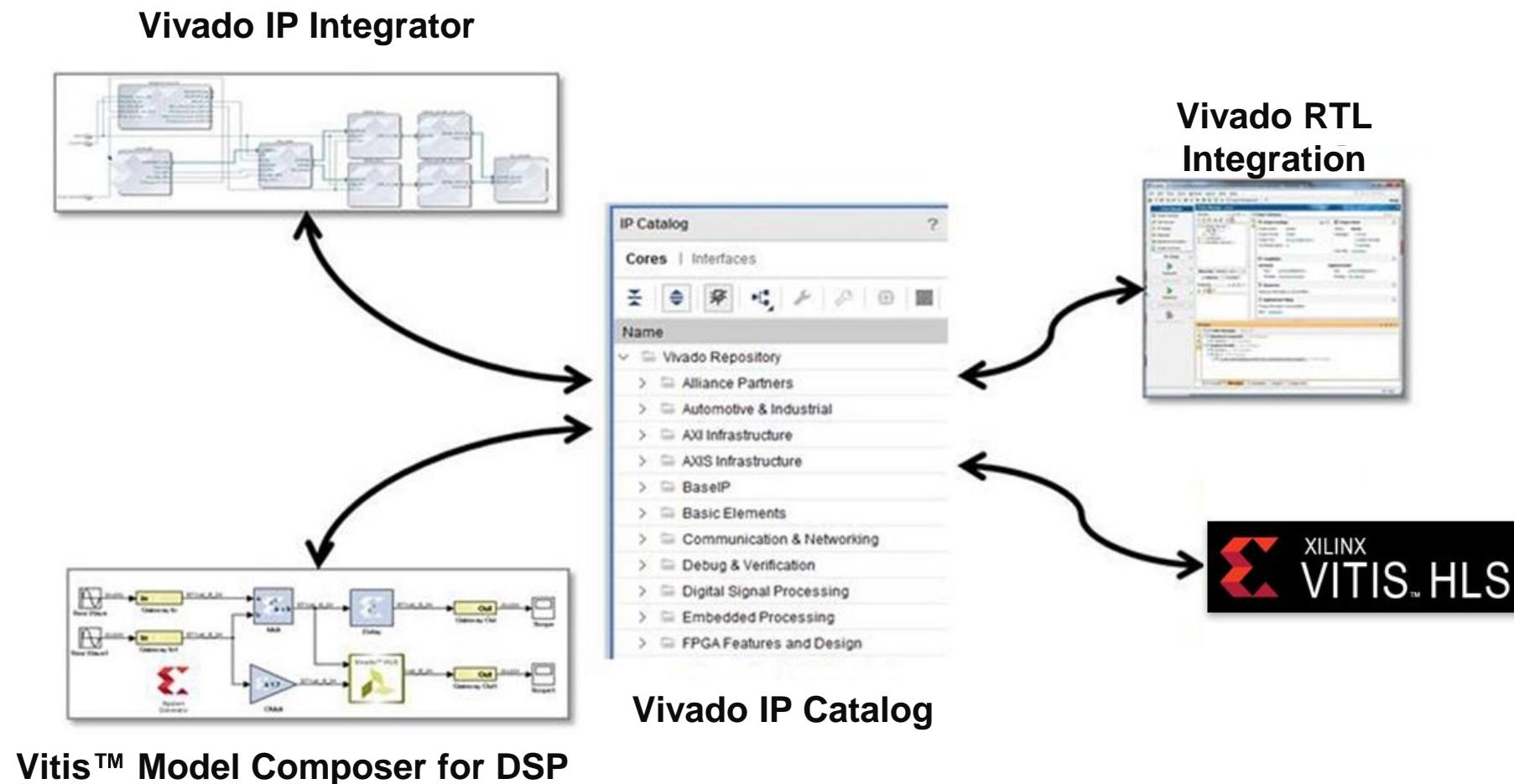


# Vivado IP Flow

2022.2

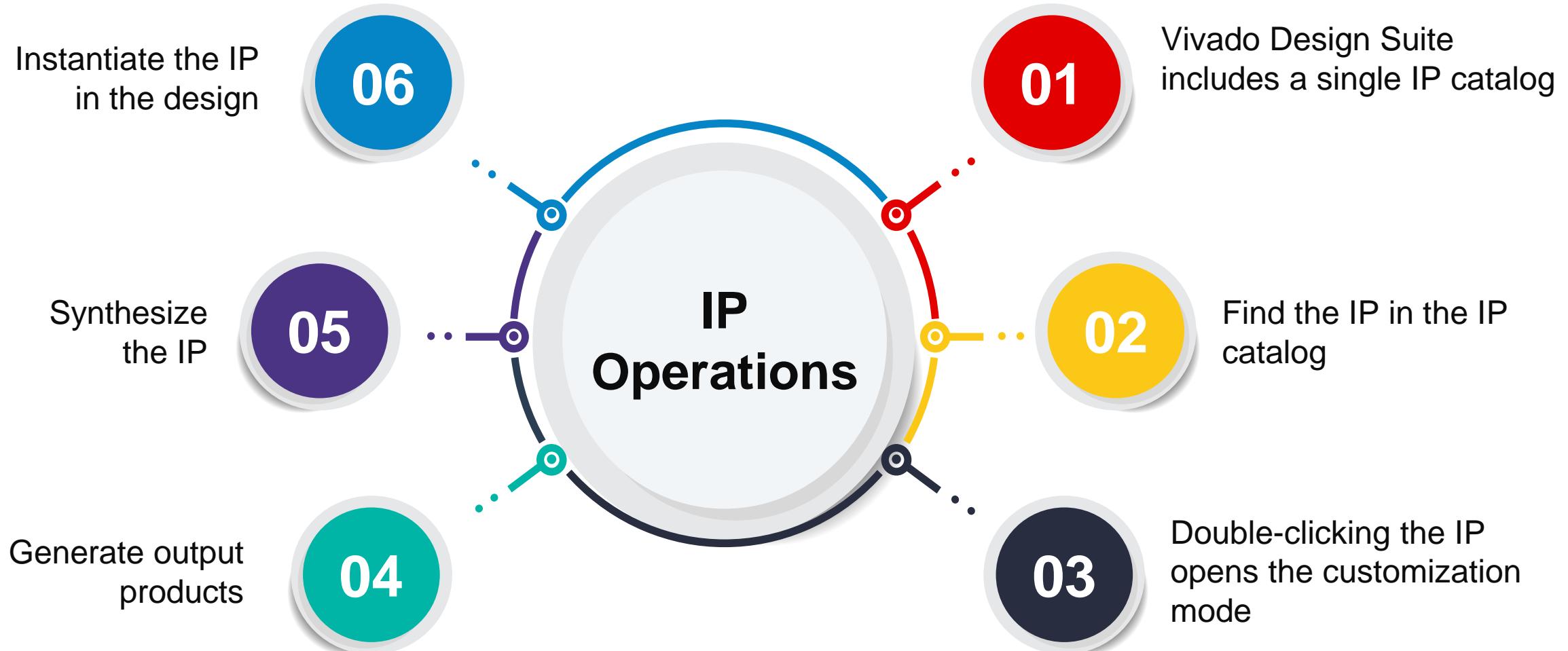
# Vivado Design Suite: IP-Centric Design Tool

The Vivado Design Suite provides direct flows from all four design environments into and out of the Vivado IP catalog

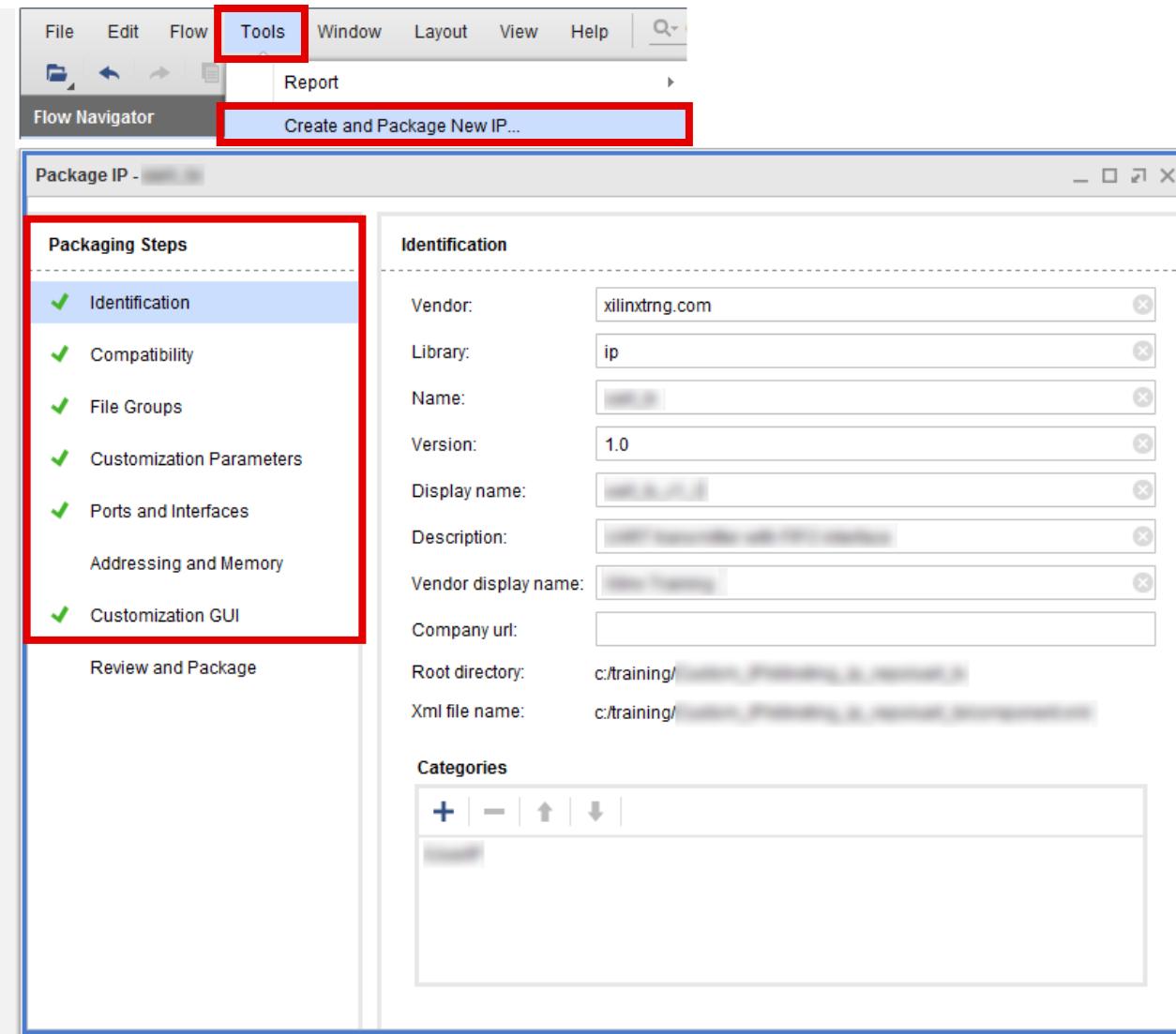


# IP Operations

The Vivado IP flow begins by accessing the IP catalog



# IP Packager



# Adding IP to the Design: Clocking Wizard

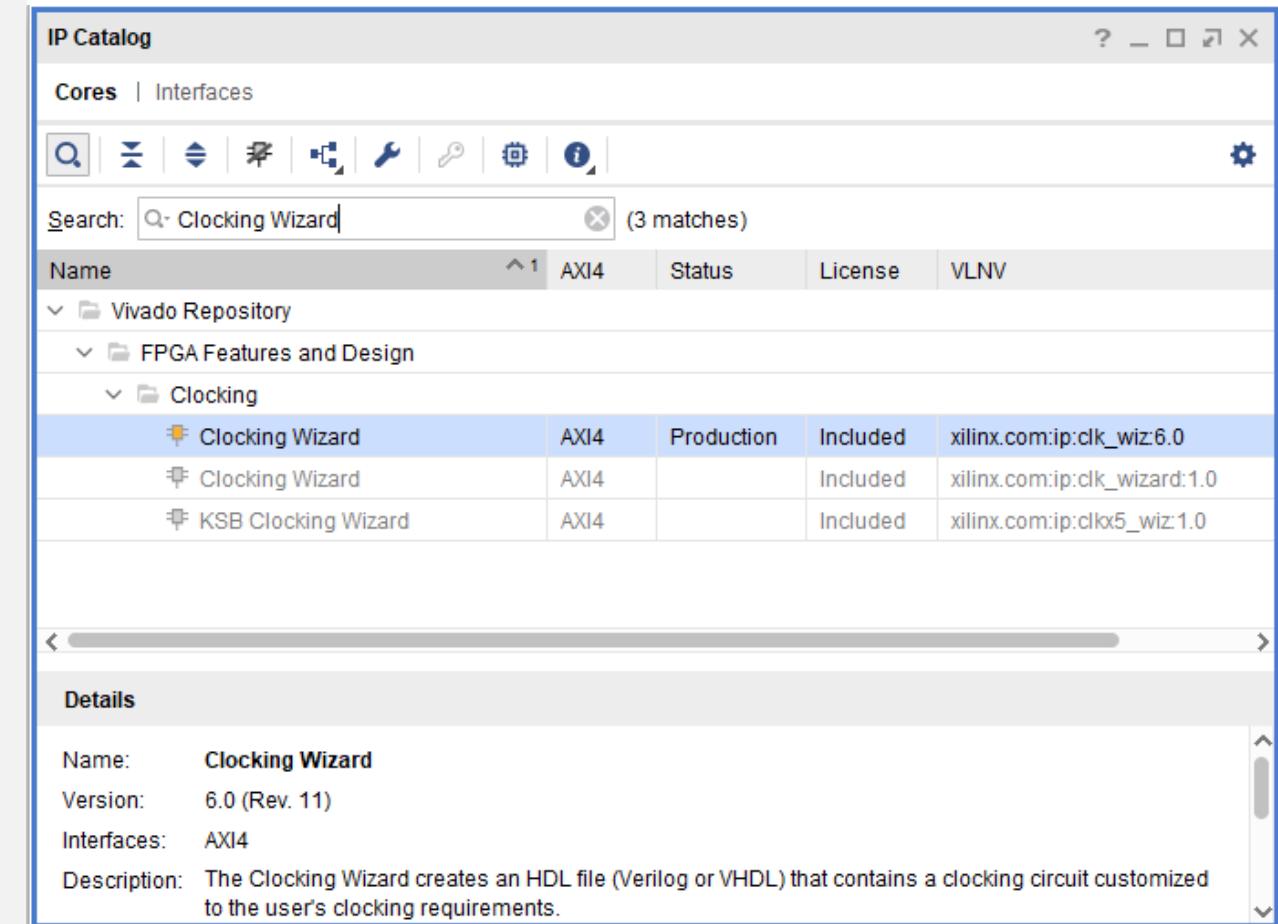
Interactive GUI lets you:

Choose the specific primitive required

Choose your features and capabilities

Modify the default settings for this IP and create the desired IP

Change as desired



# Adding IP to the Design: Clocking Wizard

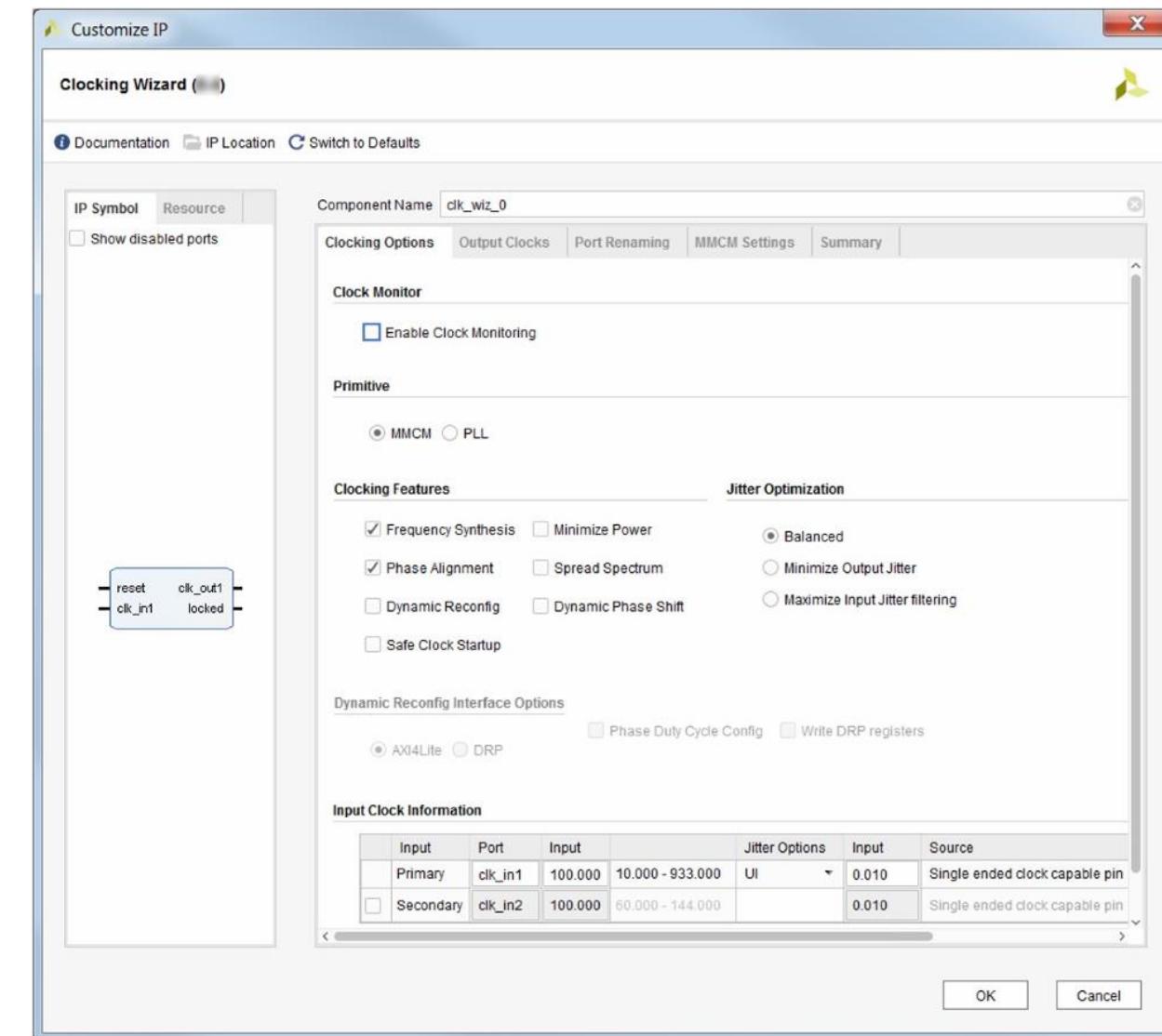
Interactive GUI lets you:

Choose the specific primitive required

Choose your features and capabilities

Modify the default settings for this IP and create the desired IP

Change as desired



# IP Synthesis Options

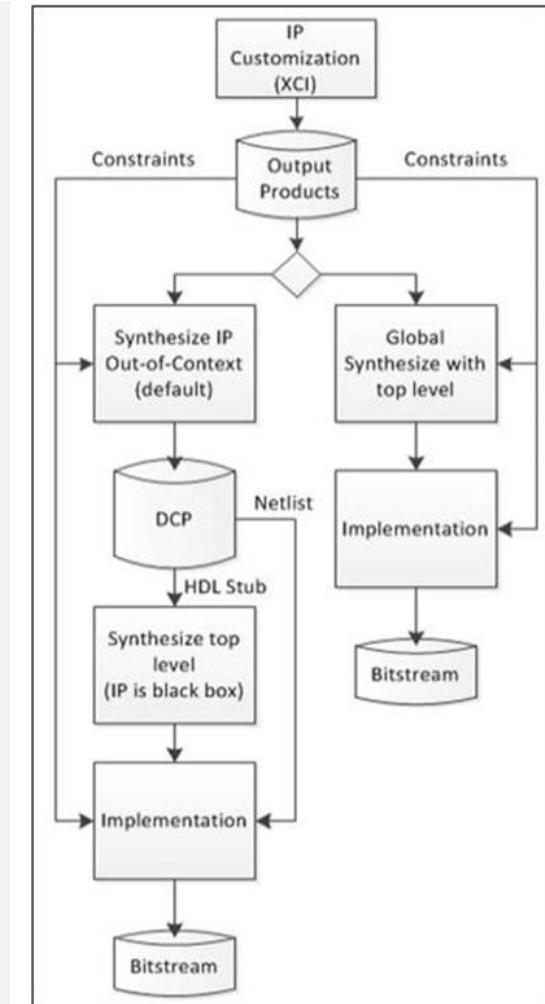
Two ways to reference IP in a design

## Out of context (OOC)-based (default flow)

- ▶ All IP catalog-delivered Vivado IP uses the OOC flow by default
  - Default settings can be overridden
- ▶ DCP file read for IP as a netlist, avoiding re-synthesis
- ▶ Can validate timing for IP at the synthesis level

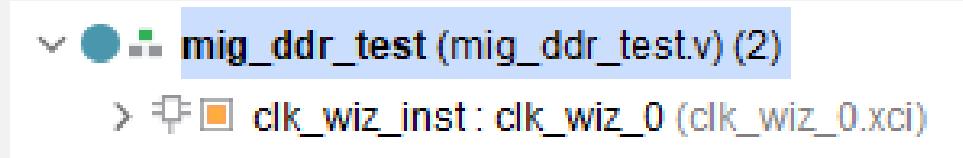
## Global synthesis

- ▶ Project-synthesized IP RTL
- ▶ Unlike the OOC flow, RTL changes (other than IP) lead to resynthesized IP
- ▶ Synthesis runs for each IP instantiation



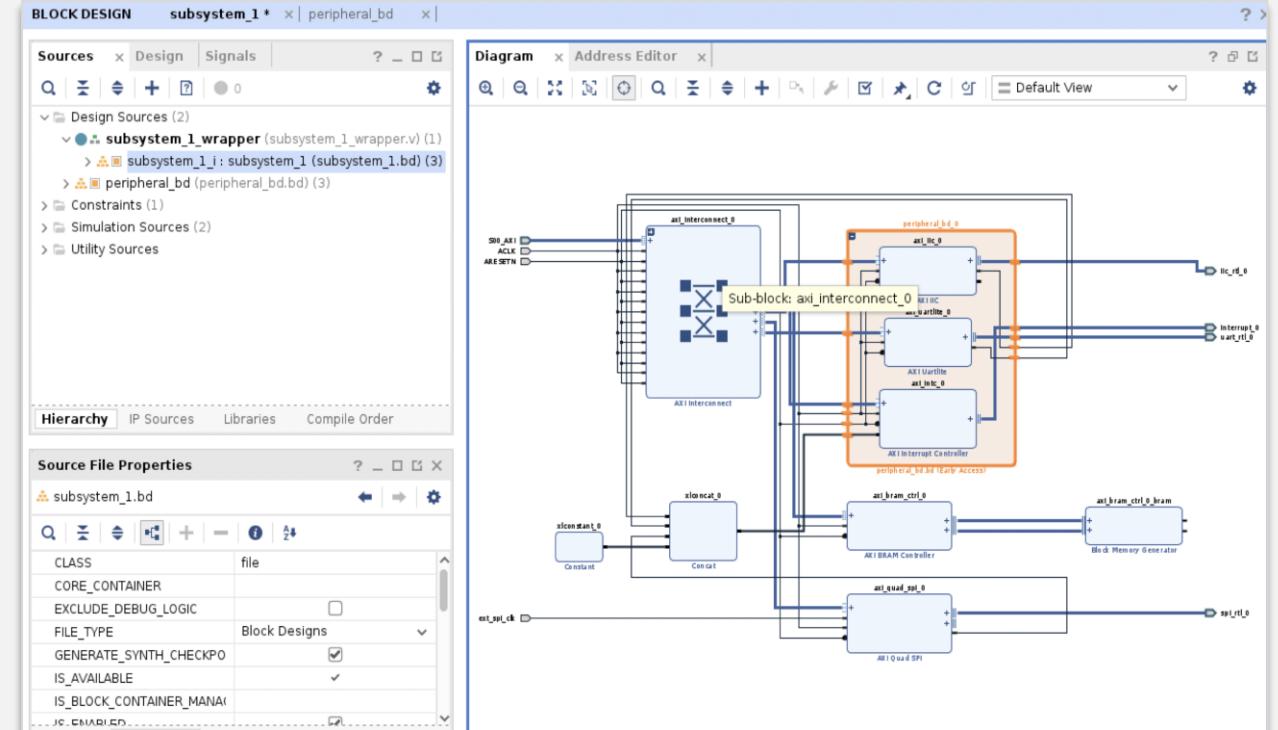
# Adding IP to the Design

## Instantiating IP



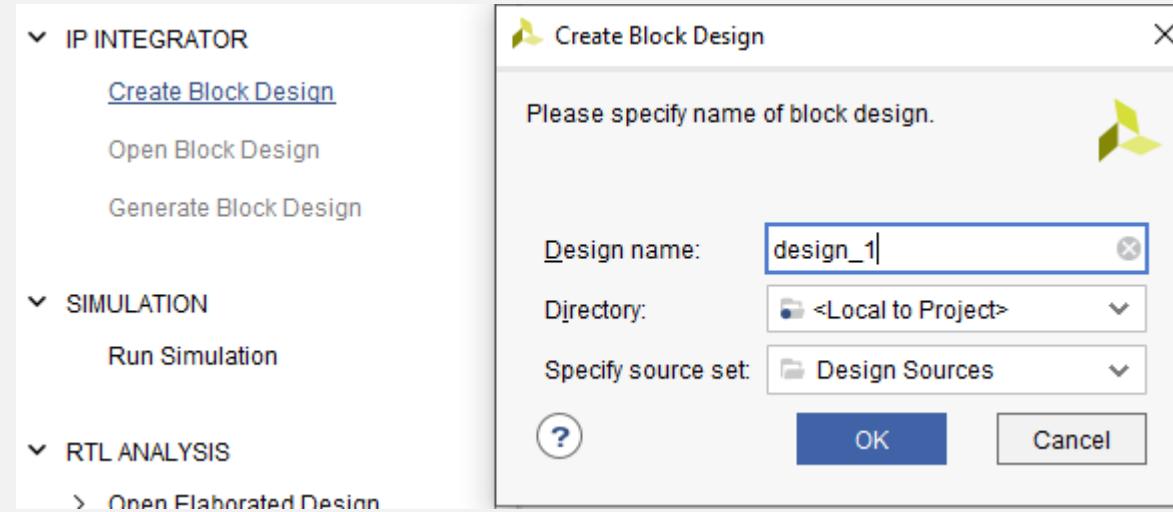
```
clk_wiz_0 clk_wiz_inst(
    .clk_out1(clk_200m),
    .reset(sys_rst),
    .locked(locked),
    .clk_in1(sysclk_i));
```

## Block Design

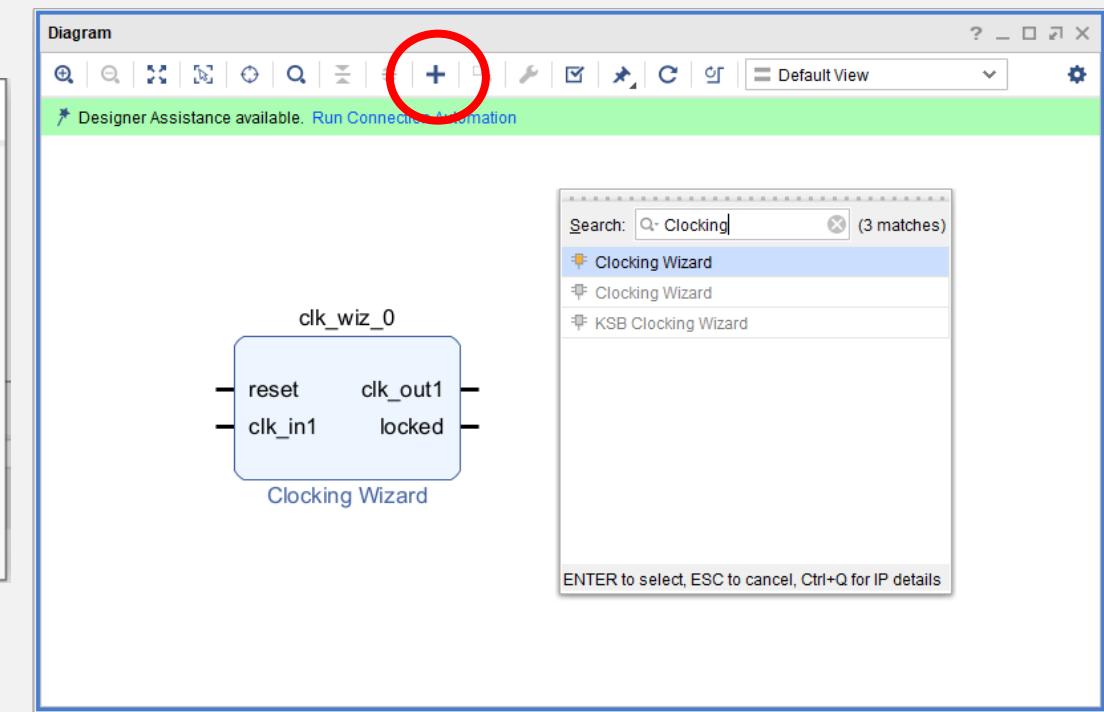


# Block Design

## Create Block Design

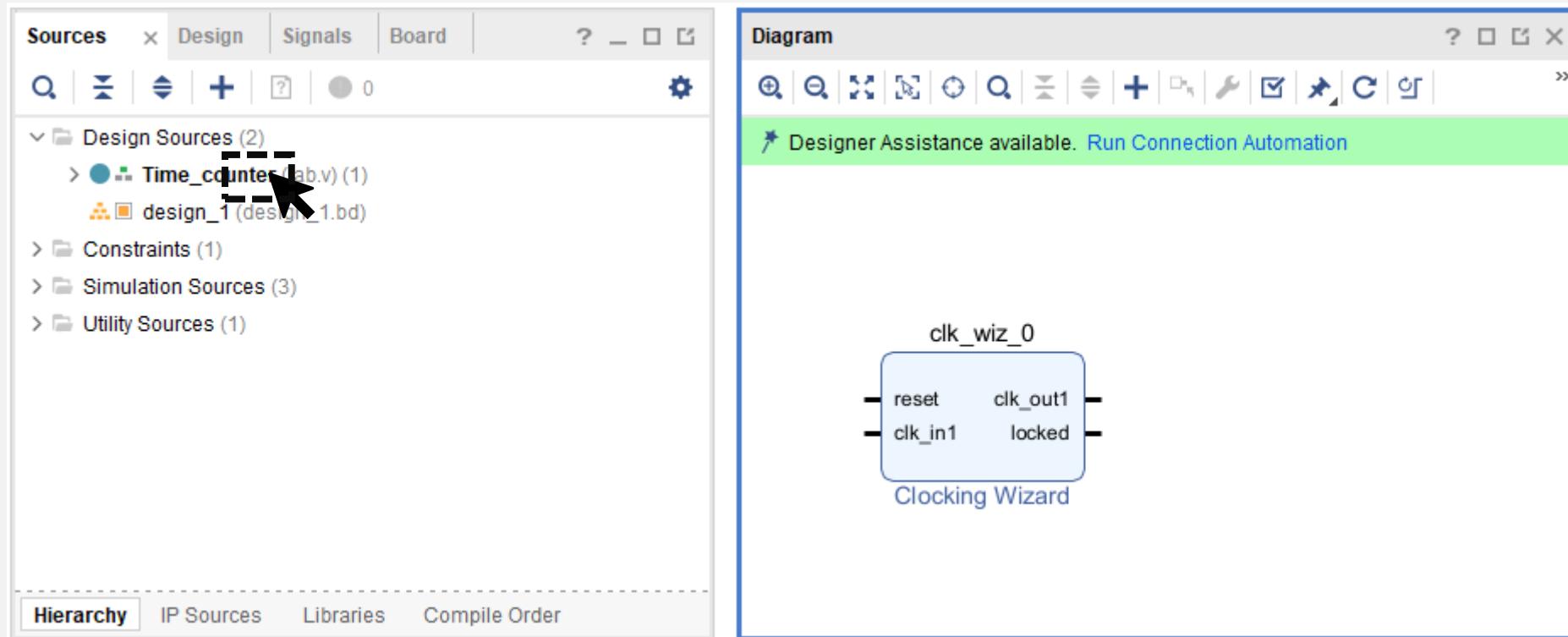


## Add IP to the Block Design



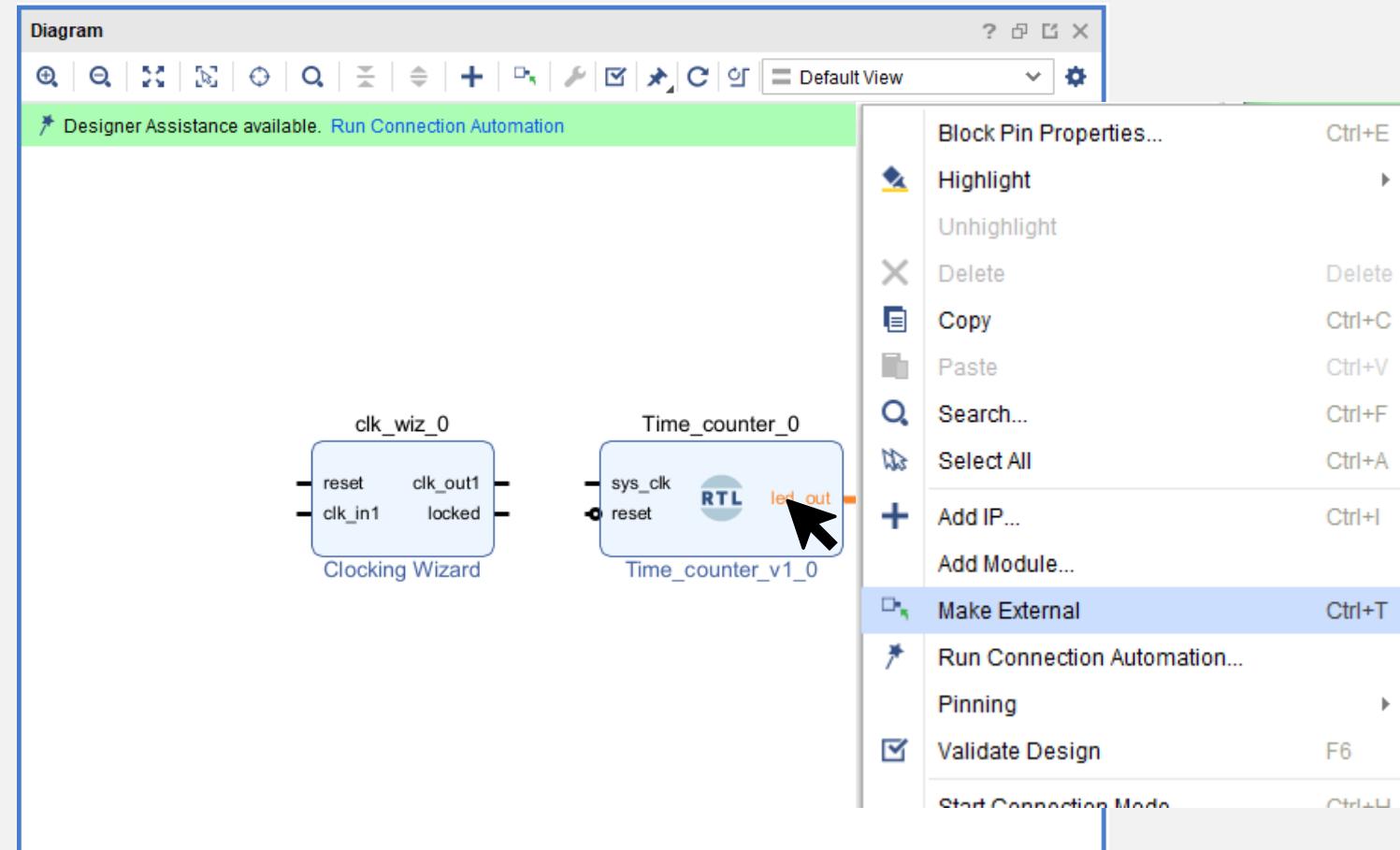
# Block Design

## Add RTL Code to the Block Design



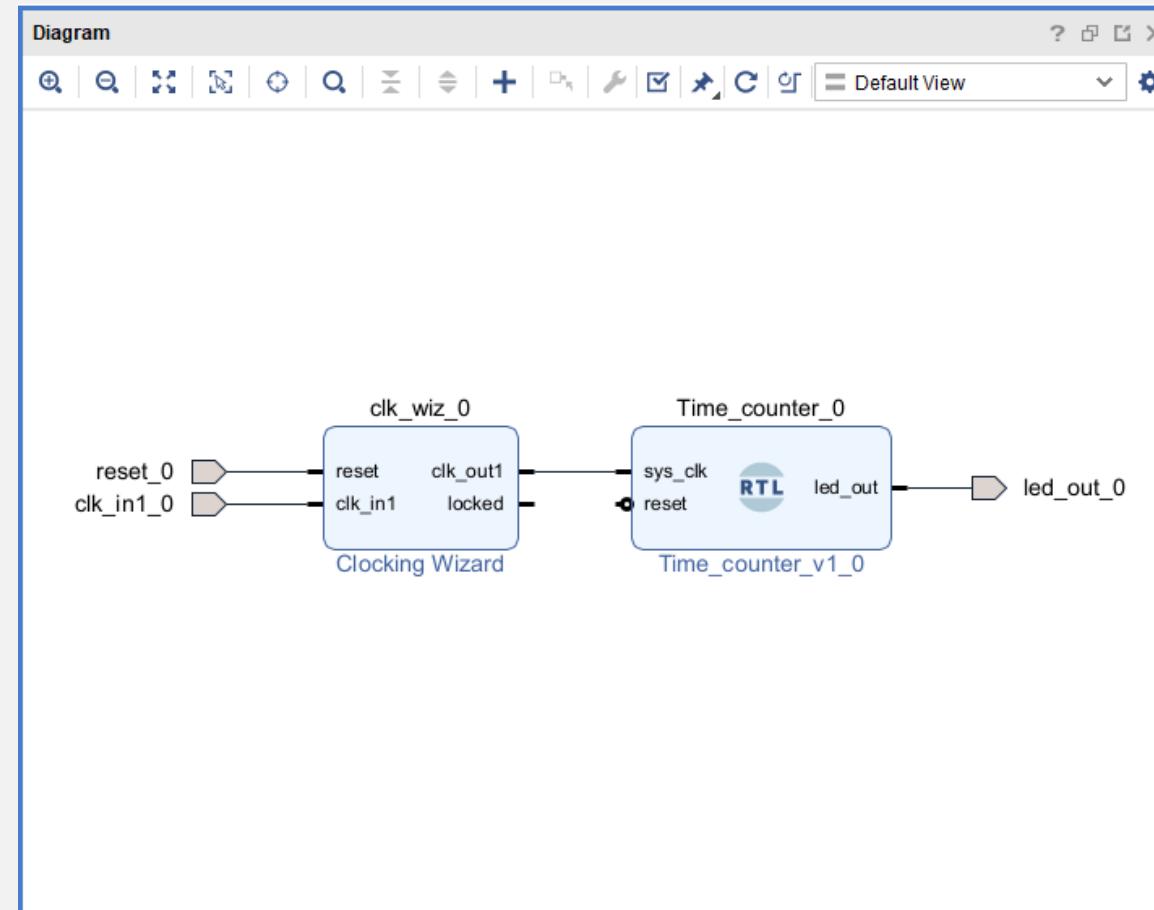
# Block Design

## Make External Port



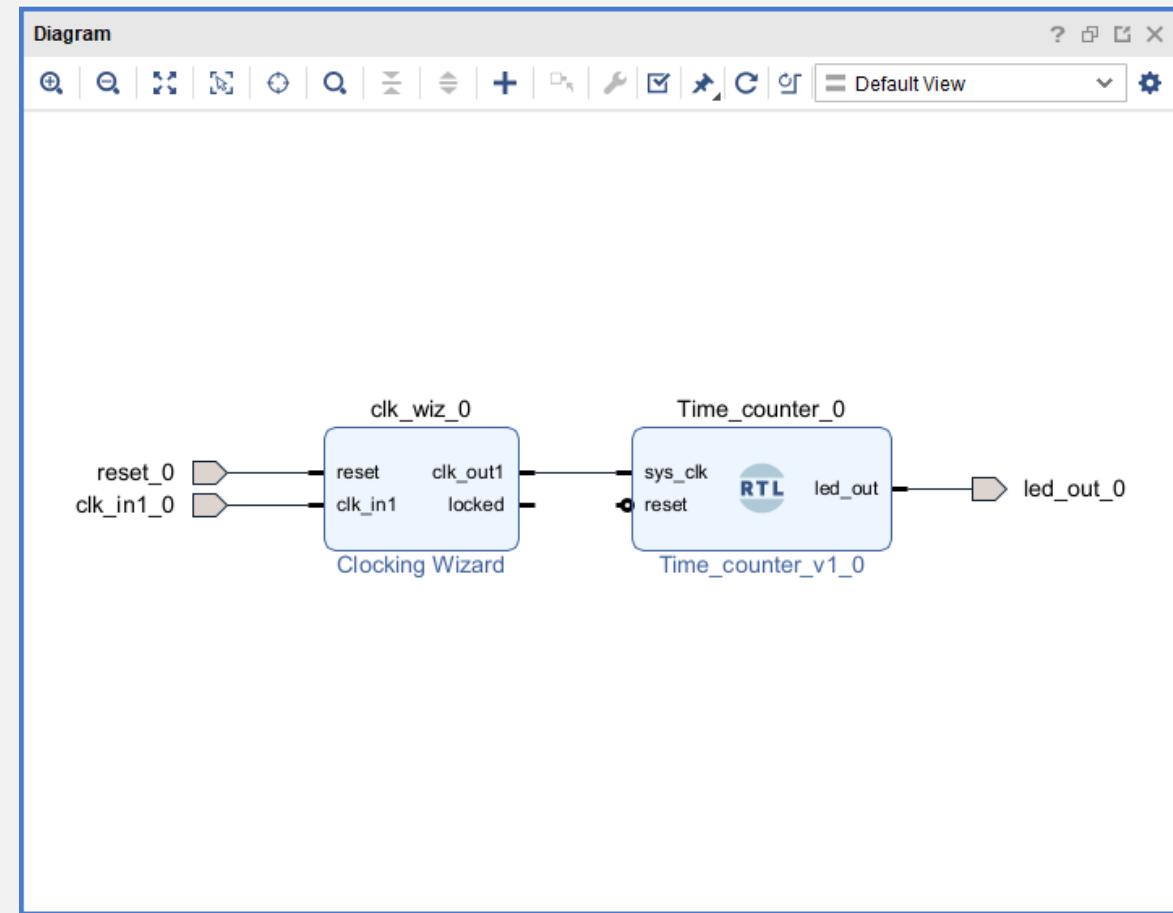
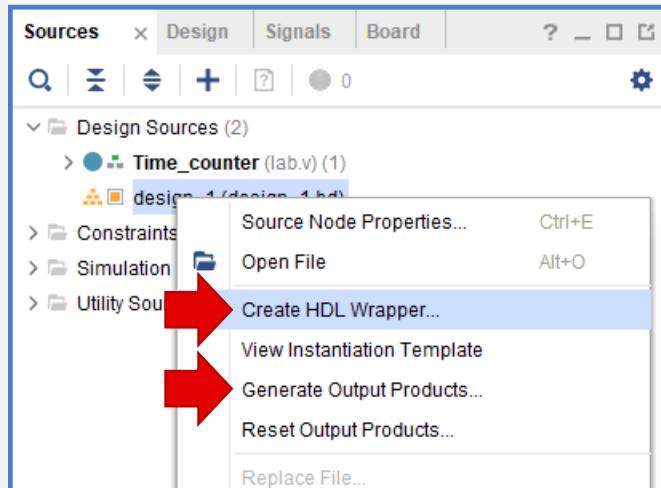
# Block Design

## Connection

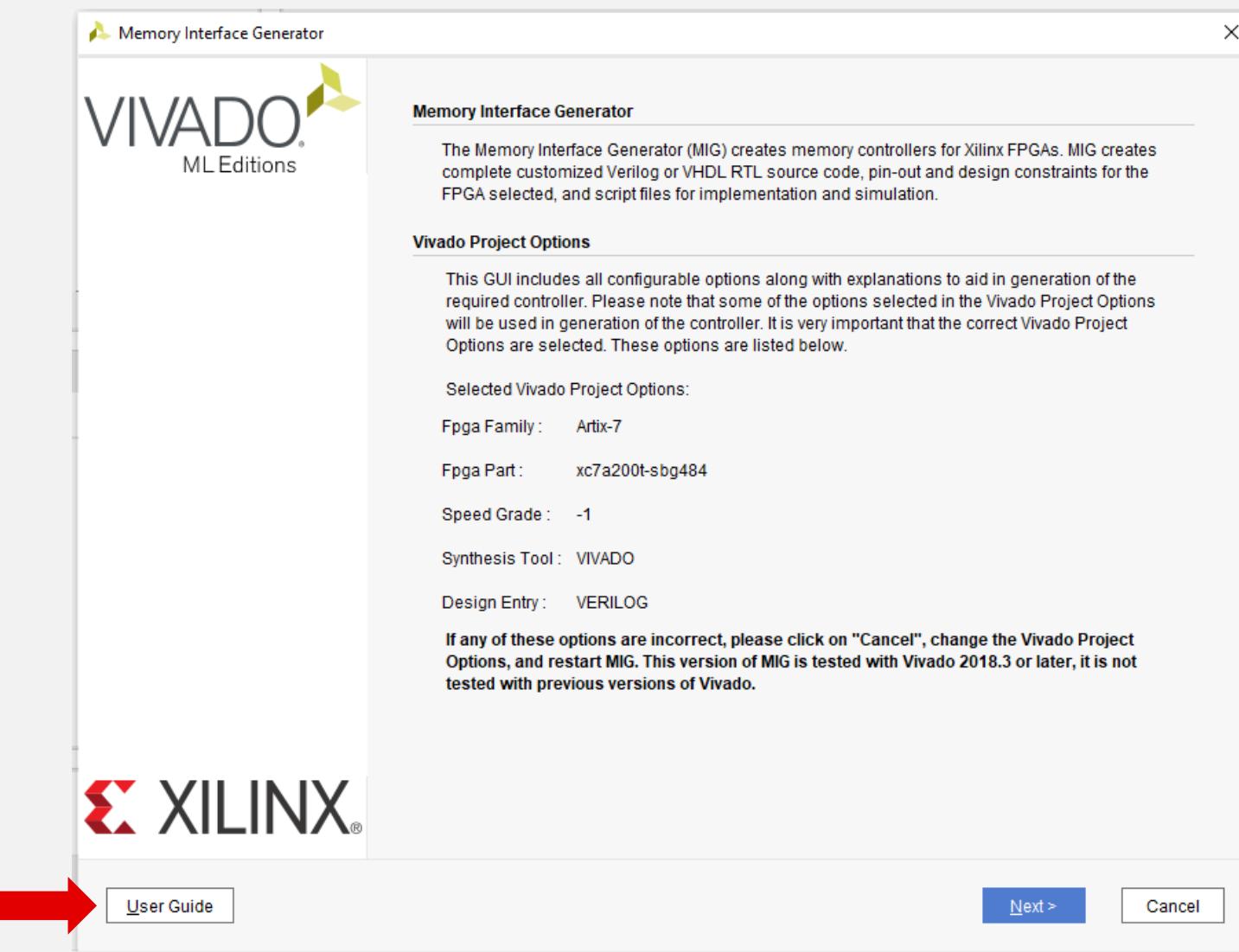


# Block Design

## Generation



# IP Use



# IP Use

## Options for Controller 0 - DDR3 SDRAM

**Clock Period:** Choose the clock period for the desired frequency. The allowed period range(1250 - 3300) is a function of the selected FPGA part and FPGA speed grade. Refer to the User Guide for more information.

1250 ps 800.0 MHz

**PHY to Controller Clock Ratio:** Select the PHY to Memory Controller clock ratio. The PHY operates at the Memory Clock Period chosen above. The controller operates at either 1/4 or 1/2 of the PHY rate. The selected Memory Clock Period will limit the choices.

4:1

**Vccaux\_io:** Vccaux\_io must be set to 2.0V in the High Performance banks for the highest data rates. Vccaux\_io is not available in the High Range banks. Note that Vccaux\_io is common to groups of banks. Consult the 7 Series Datasheets and FPGA SelectIO Resources User Guide for more information.

2.0V

**Memory Type:** Select the memory type. Type(s) marked with a warning symbol are not compatible with the frequency selection above.

Components

MT41J128M8XX-125

Create Custom Part

**Memory Part:** Select the memory part. Part(s) marked with a warning symbol are not compatible with the frequency selection above. Find an equivalent part or create a part using the "Create Custom Part" button if the part needed is not listed here. The "Create Custom Part" feature is not supported for RLDRAM II.

**Memory Voltage:** Select the Voltage of the Memory part selected.

1.5V

**Data Width:** Select the Data Width. Parts marked with a warning symbol are not compatible with the frequency and memory part selected above.

8

**VCCAUX\_IO** – Set based on the period/frequency setting. 2.0V is required at the highest frequency settings in the High Performance column. The MIG tool automatically selects 2.0V when required. Either 1.8 or 2.0V can be used at lower frequencies. Groups of banks share the VCCAUX\_IO supply. For more information, see the *7 Series FPGAs SelectIO™ Resources User Guide (UG471)* [Ref 2] and the *7 Series FPGAs Packaging and Pinout Specification (UG475)* [Ref 3].

**Memory Type** – This feature selects the type of memory parts used in the design.

**Memory Part** – This option selects a memory part for the design. Selections can be made from the list or a new part can be created.

**Note:** For a complete list of memory parts available, see Answer Record: [54025](#).

**Data Width** – The data width value can be selected here based on the memory type selected earlier. The list shows all supported data widths for the selected part. One of the data widths can be selected. These values are generally multiples of the individual device data widths. In some cases, the width might not be an exact multiple. For example, 16 bits is the default data width for x16 components, but eight bits is also a valid value.

**Data Mask** – This option allocates data mask pins when selected. This should be deselected to deallocate data mask pins and increase pin efficiency. Also, this is disabled for memory parts that do not support data mask.

# IP Use

Table 1-17: User Interface

Signal	Direction	Description
app_addr[ADDR_WIDTH - 1:0]	Input	This input indicates the address for the current request.
app_cmd[2:0]	Input	This input selects the command for the current request.
app_en	Input	This is the active-High strobe for the app_addr[], app_cmd[2:0] inputs, app_sz, and app_hi_pri inputs.
app_rdy	Output	This output indicates that the UI is ready to accept commands. If the signal is deasserted when app_en is enabled, the current app_cmd and app_addr must be retried until app_rdy is asserted.
app_hi_pri	Input	This active-High input elevates the priority of the current request.
app_rd_data [APP_DATA_WIDTH - 1:0]	Output	This provides the output data from read commands.
app_rd_data_end	Output	This active-High output indicates that the current clock cycle is the last cycle of output data on app_rd_data[]. This is valid only when app_rd_data_valid is active-High.
app_rd_data_valid	Output	This active-High output indicates that app_rd_data[] is valid.
app_sz	Input	This input is reserved and should be tied to 0.
app_wdf_data [APP_DATA_WIDTH - 1:0]	Input	This provides the data for write commands.
app_wdf_end	Input	This active-High input indicates that the current clock cycle is the last cycle of input data on app_wdf_data[].
app_wdf_mask		

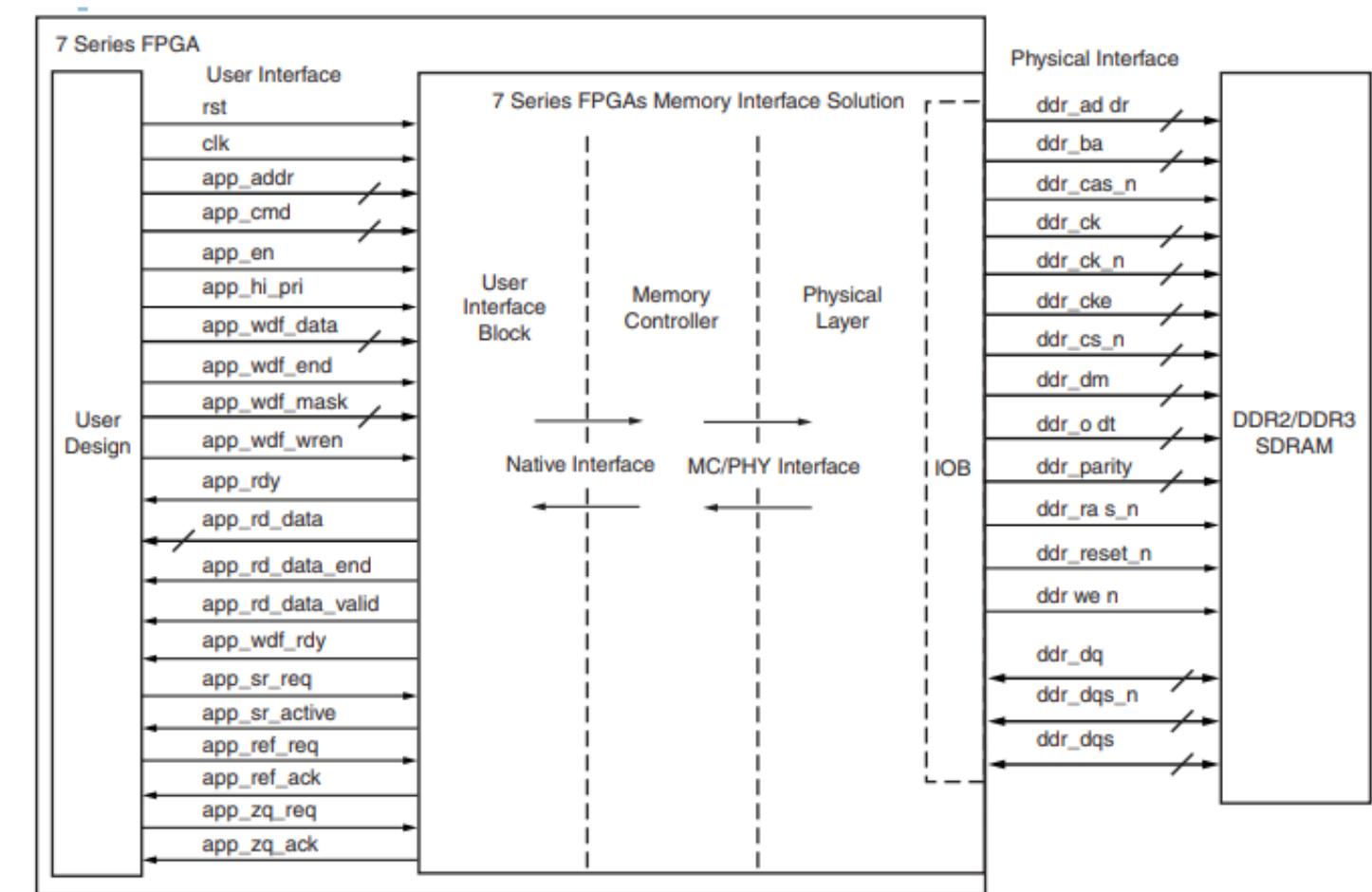


Figure 1-31: 7 Series FPGAs Memory Interface Solution

UG586\_c1\_43\_120311

# IP Use

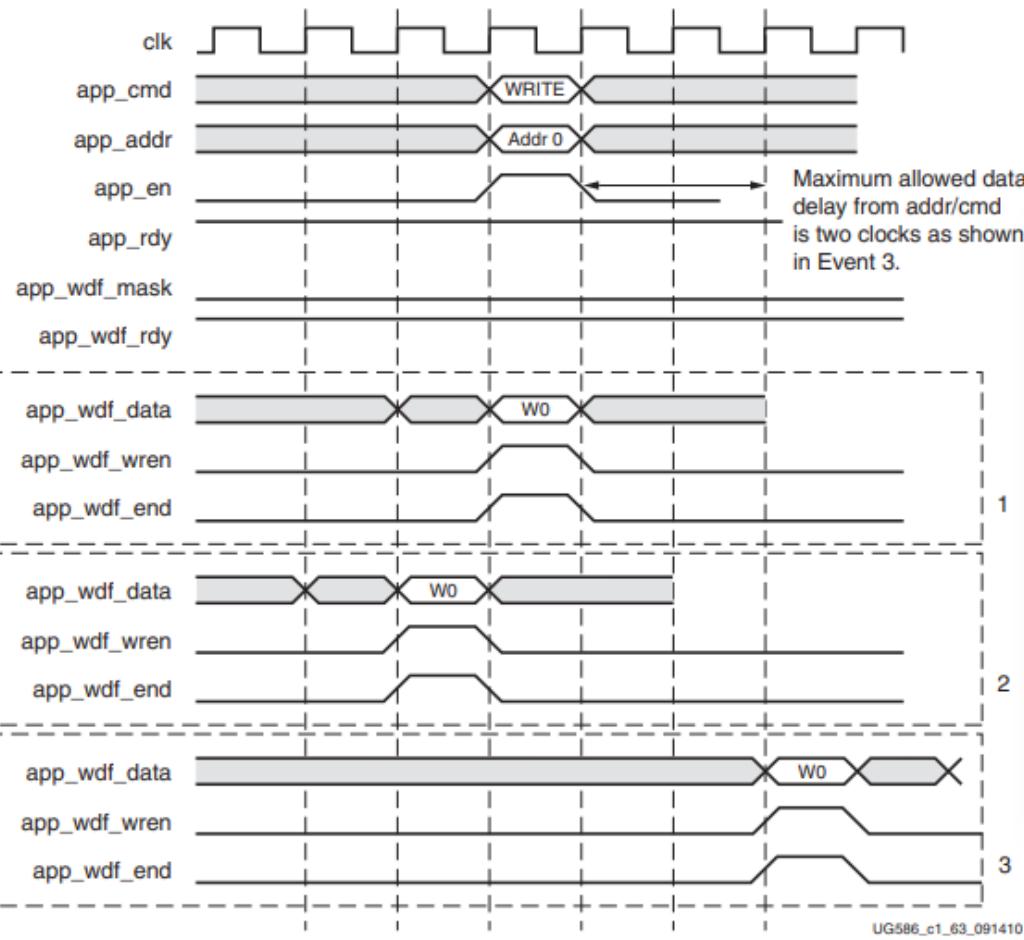


Figure 1-54: 4:1 Mode UI Interface Write Timing Diagram  
(Memory Burst Type = BL8)

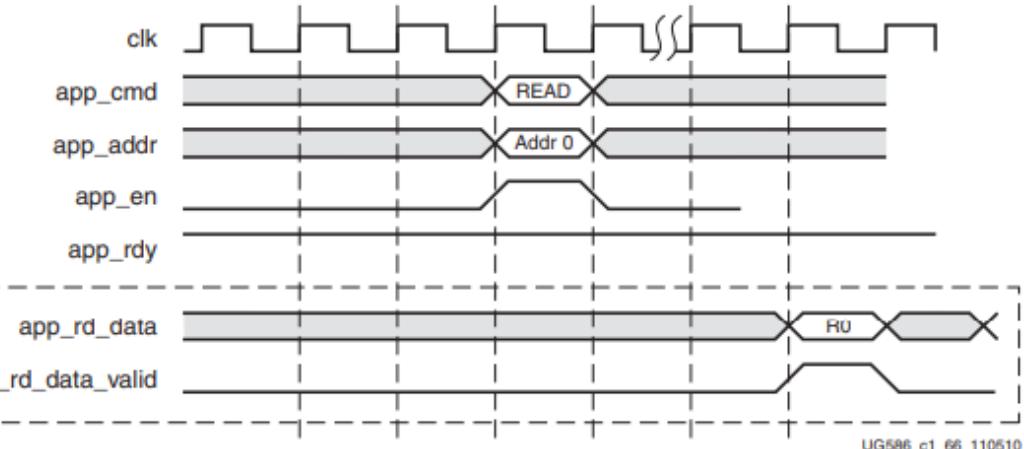
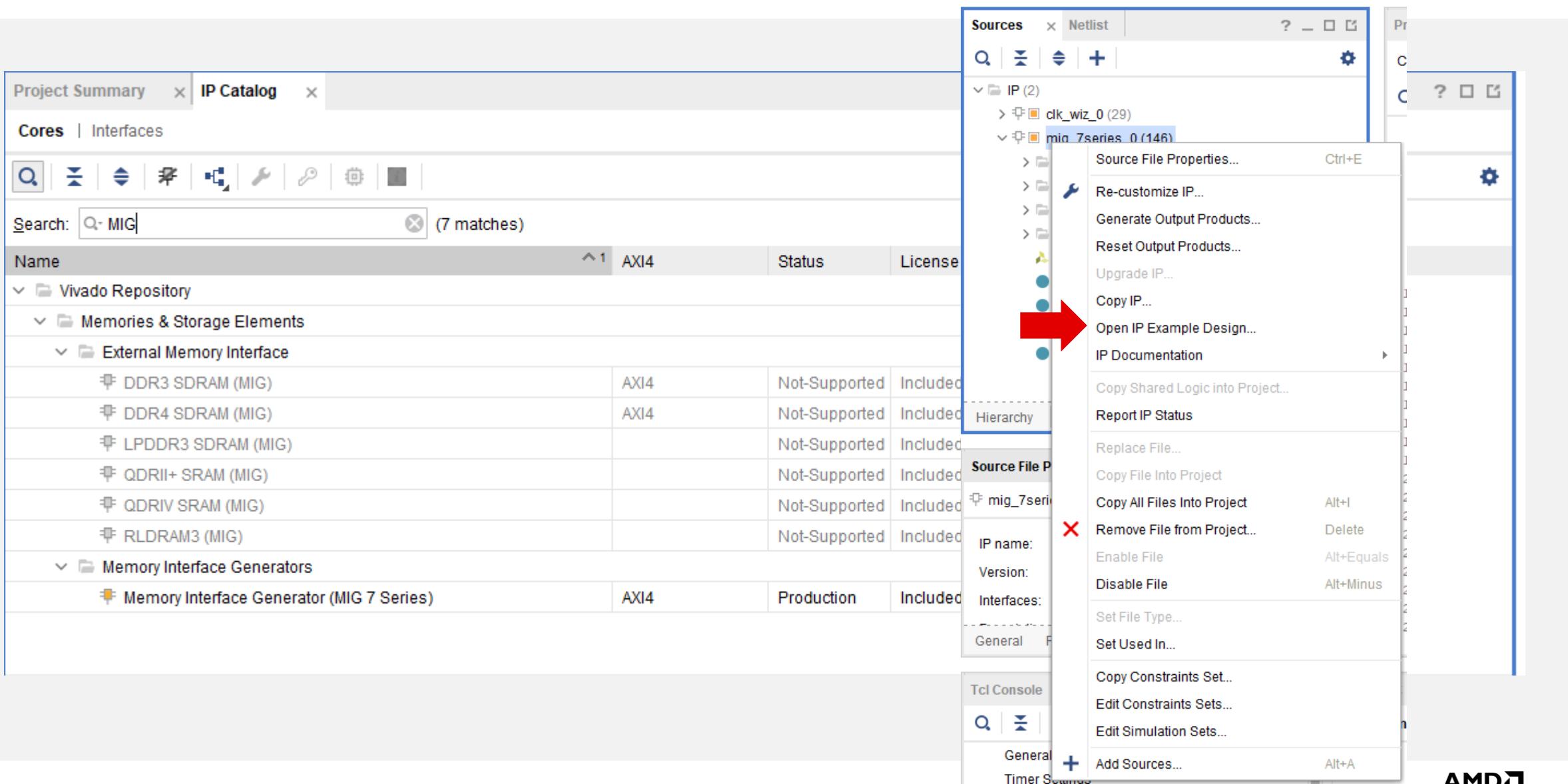
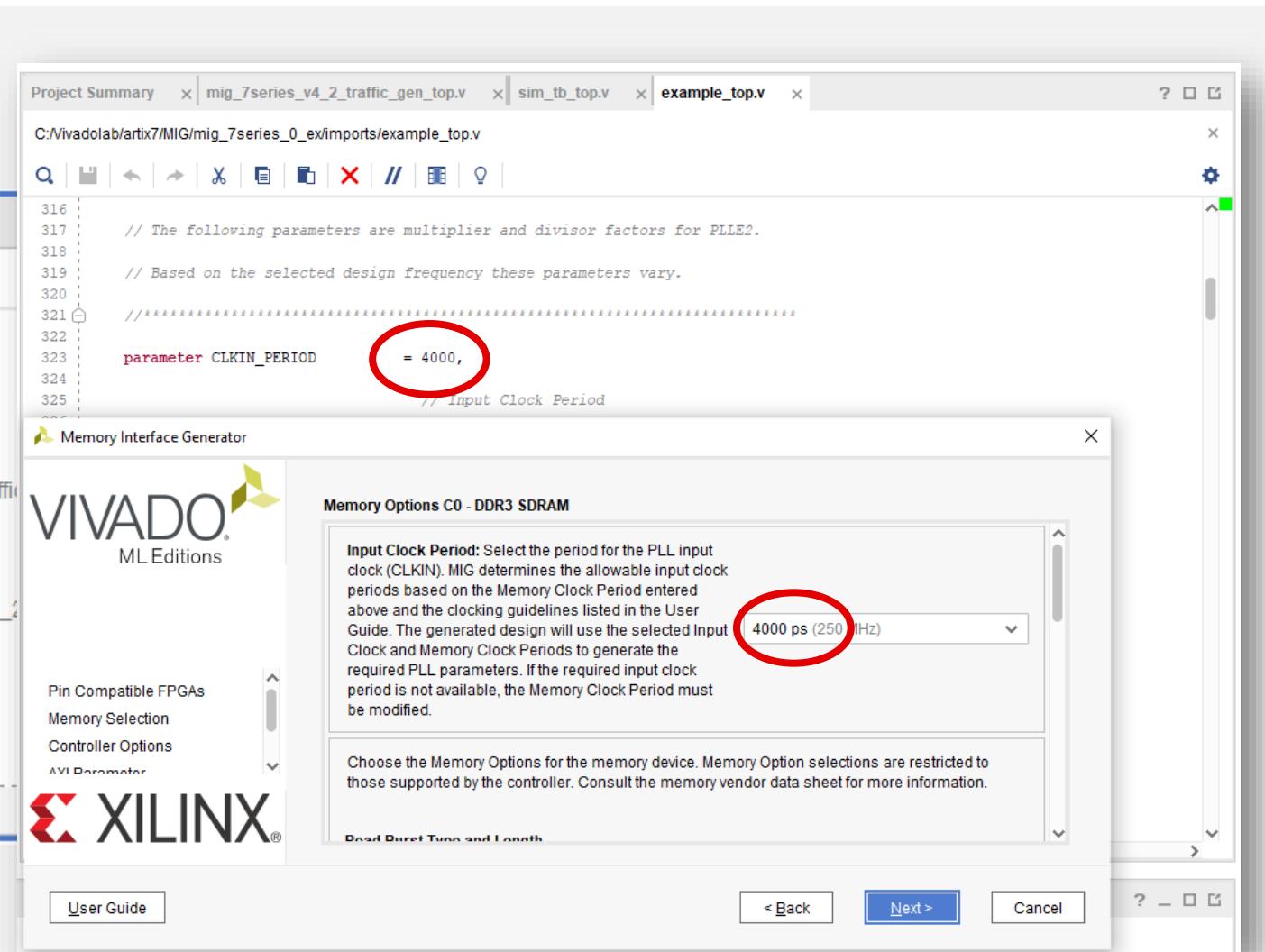
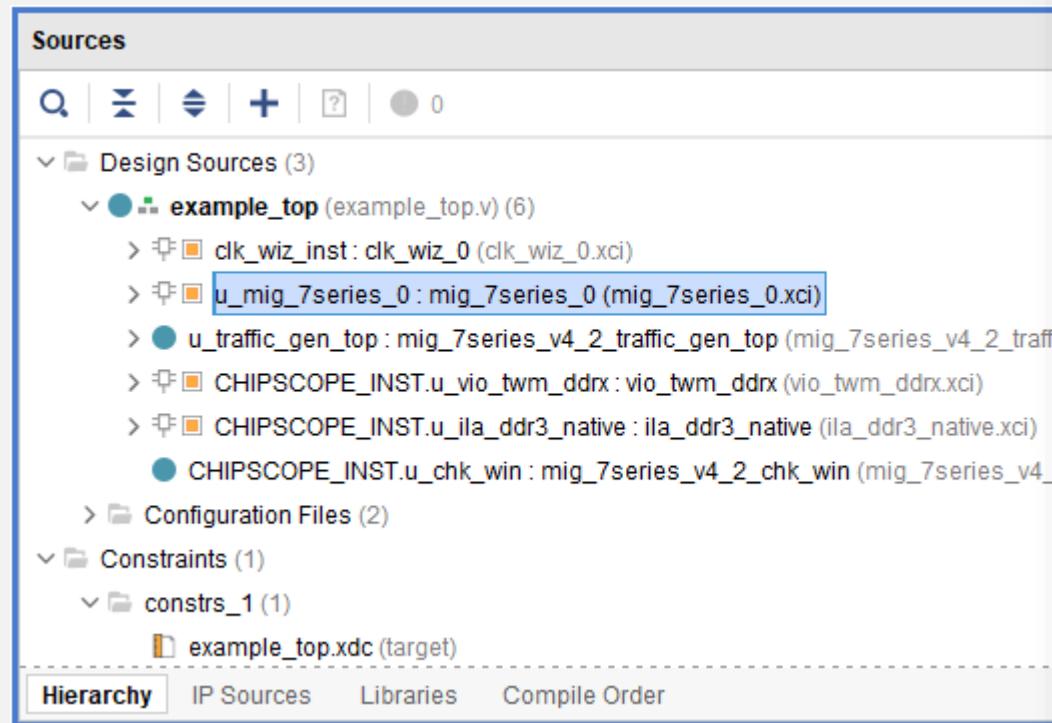


Figure 1-60: 4:1 Mode UI Interface Read Timing Diagram  
(Memory Burst Type = BL8)

# IP Use



# IP Use



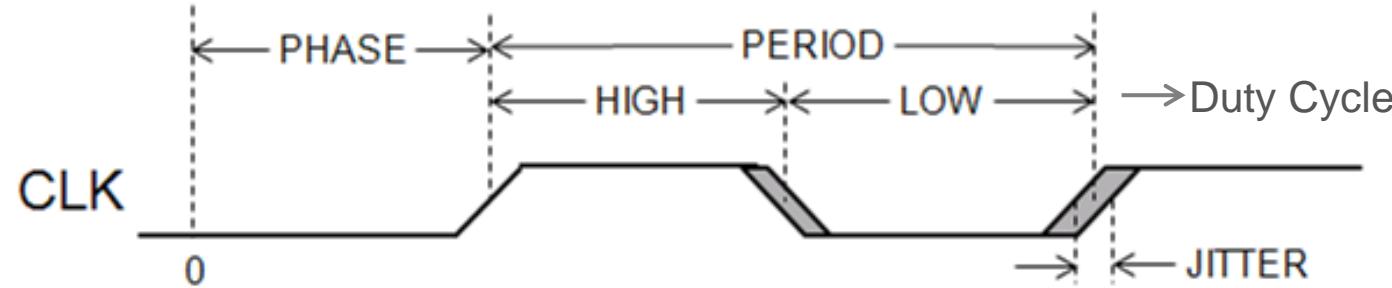


# Introduction to Clock Constraints

2022.2

# Definition of a Clock and its Attributes

A clock oscillates between high and low states and synchronizes circuit signals



## Period

Nominal time from rising edge of the clock to the next rising edge of the clock

## Duty Cycle

Ratio of the high and low time of the clock signal

## Jitter

Deviation from the ideal period of a clock signal

## Phase

Starting point of the rising edge of the clock

# Creating Clocks

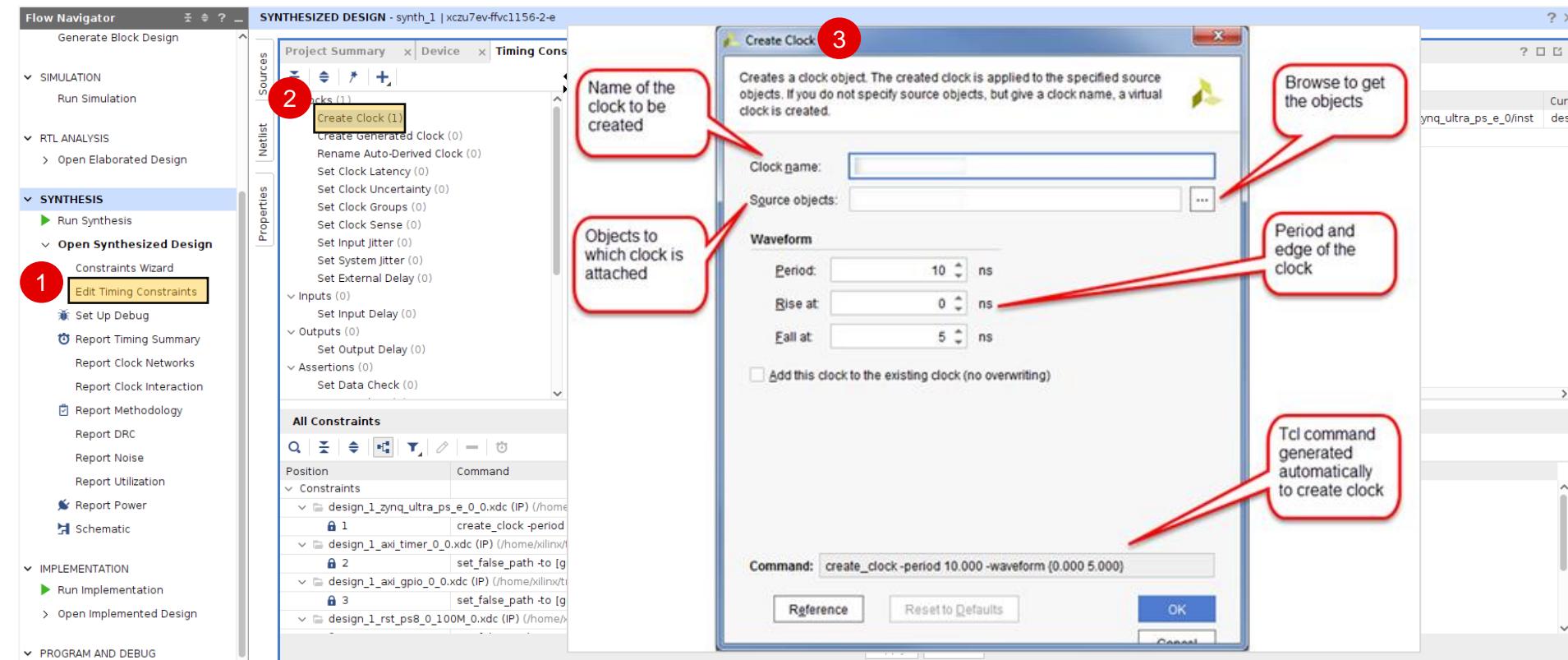
Two methods to create a clock in the XDC file

## Using Tcl Commands

## Using the GUI

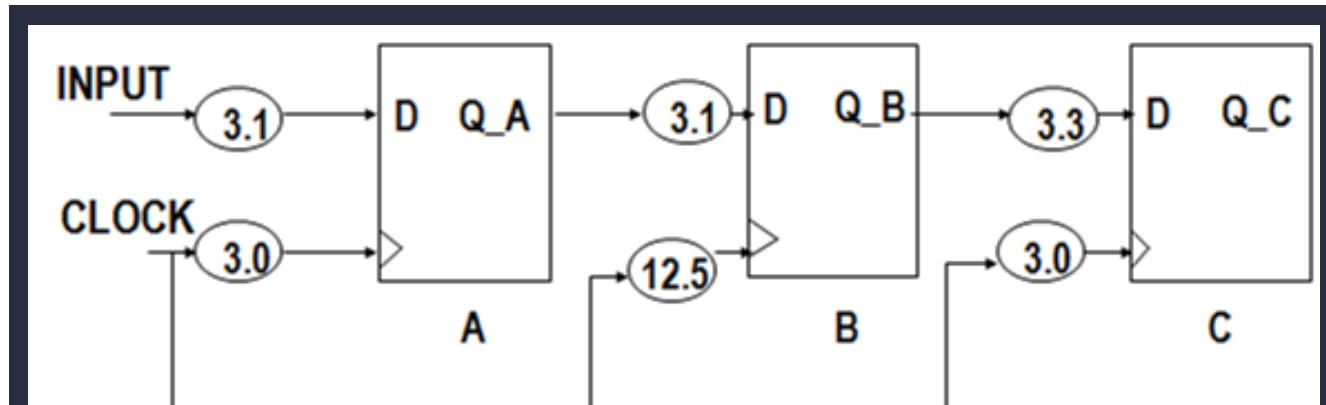
Clocks can be created using the GUI as follows:

- **Step 1:** Open the Timing Constraints window
- **Step 2:** Double-click **Create Clock** to launch the wizard
- **Step 3:** Define the clock in the Create Clock Wizard

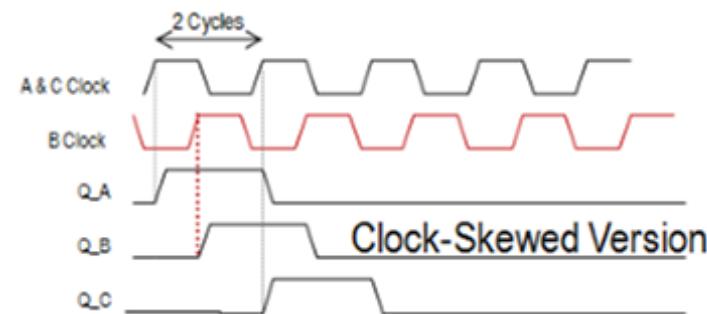
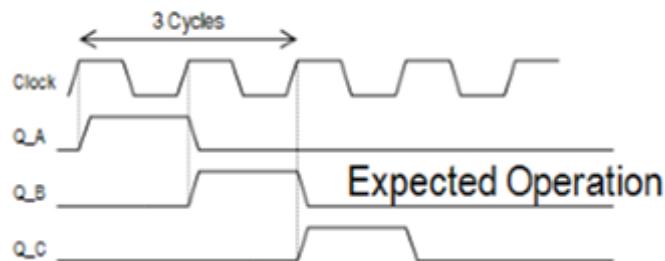


# Clock Skew

## Example



This shift register will not work because of clock skew!

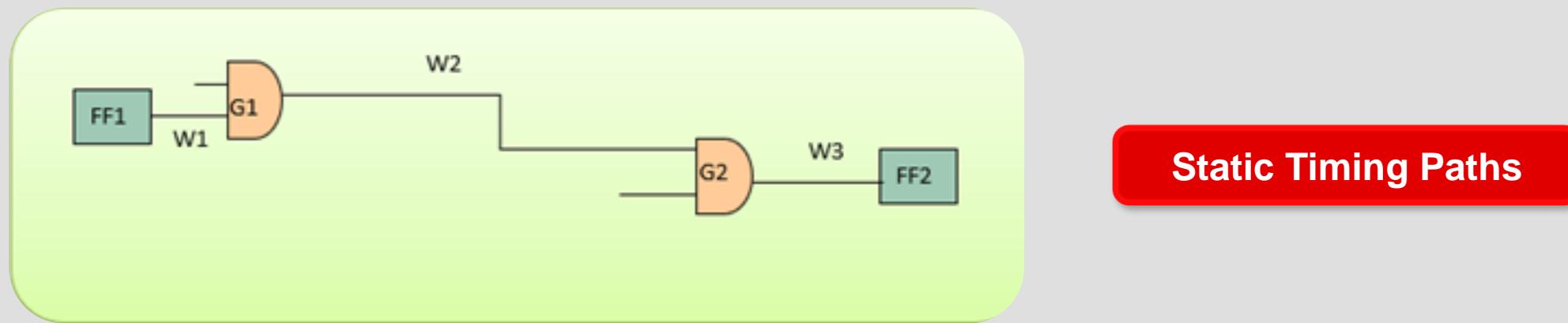


# Static Timing Analysis (STA)

- ▶ A design is an interconnected set of cells and nets
- ▶ RTL sources determine design functionality
- ▶ Device performance is determined by cell delays
- ▶ STA analyzes, debugs, and validates design timing

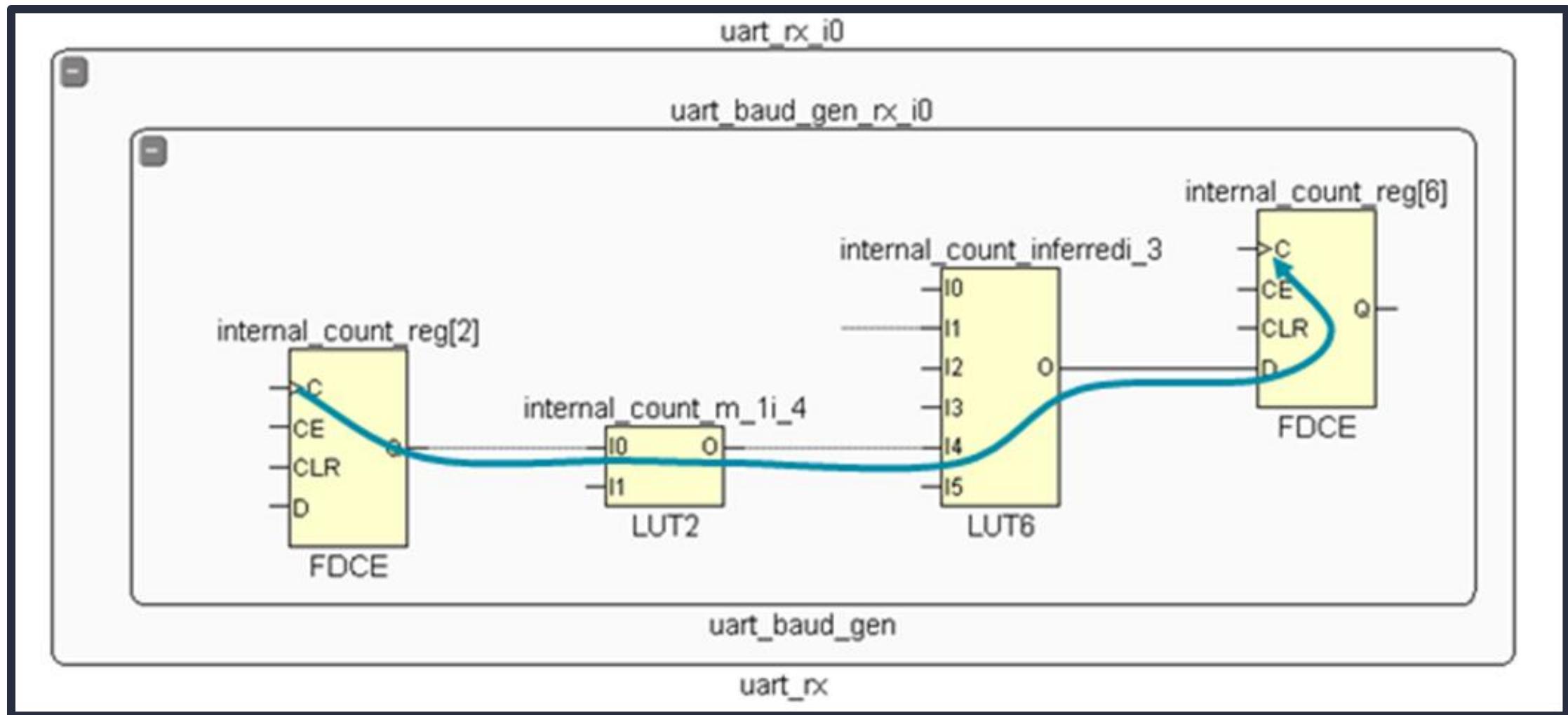
**Goal:** To ensure the design meets timing constraints

Checks every timing path (gates+wires)



Delay in gates (G1, G2) + delay in wires (W1, W2, W3) < expected delay on the path

# Static Timing Paths



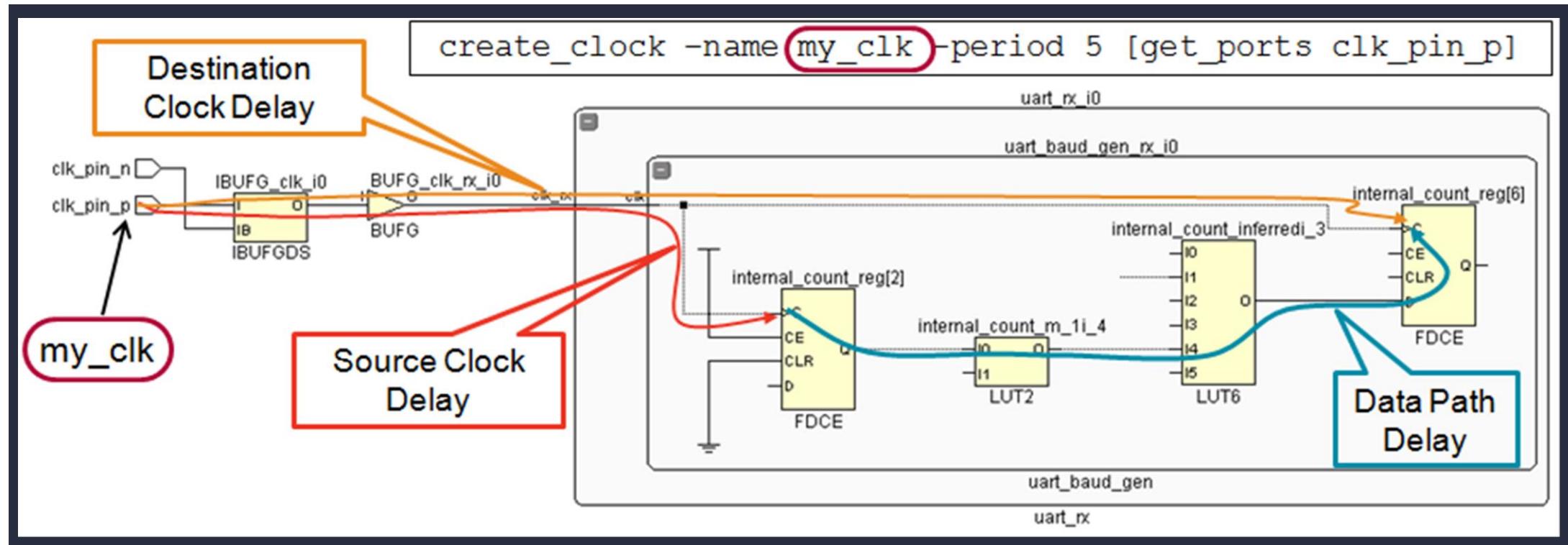
# Setup Check on Clock

The setup check of a static timing path is calculated using the following equation:

Slack = PERIOD + destination clock delay - (source clock delay + data path delay) - clock uncertainty

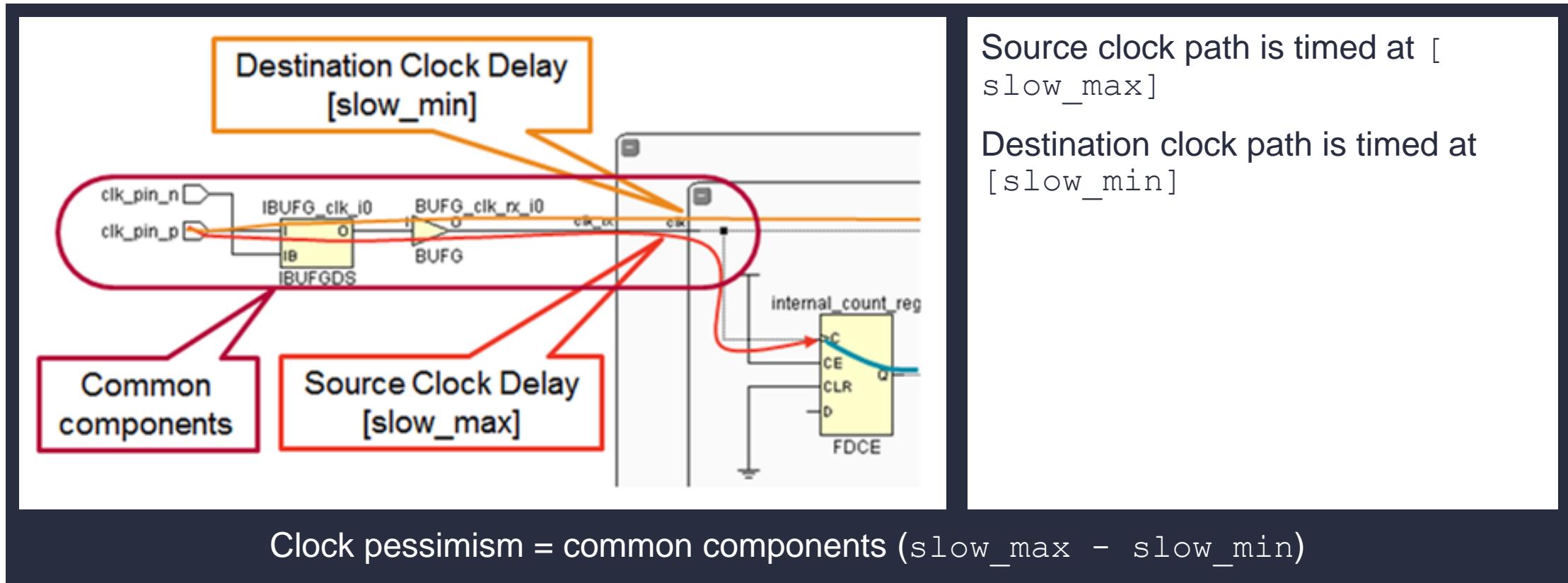
If slack is positive, then the setup check passes

If slack is negative, this indicates a timing failure

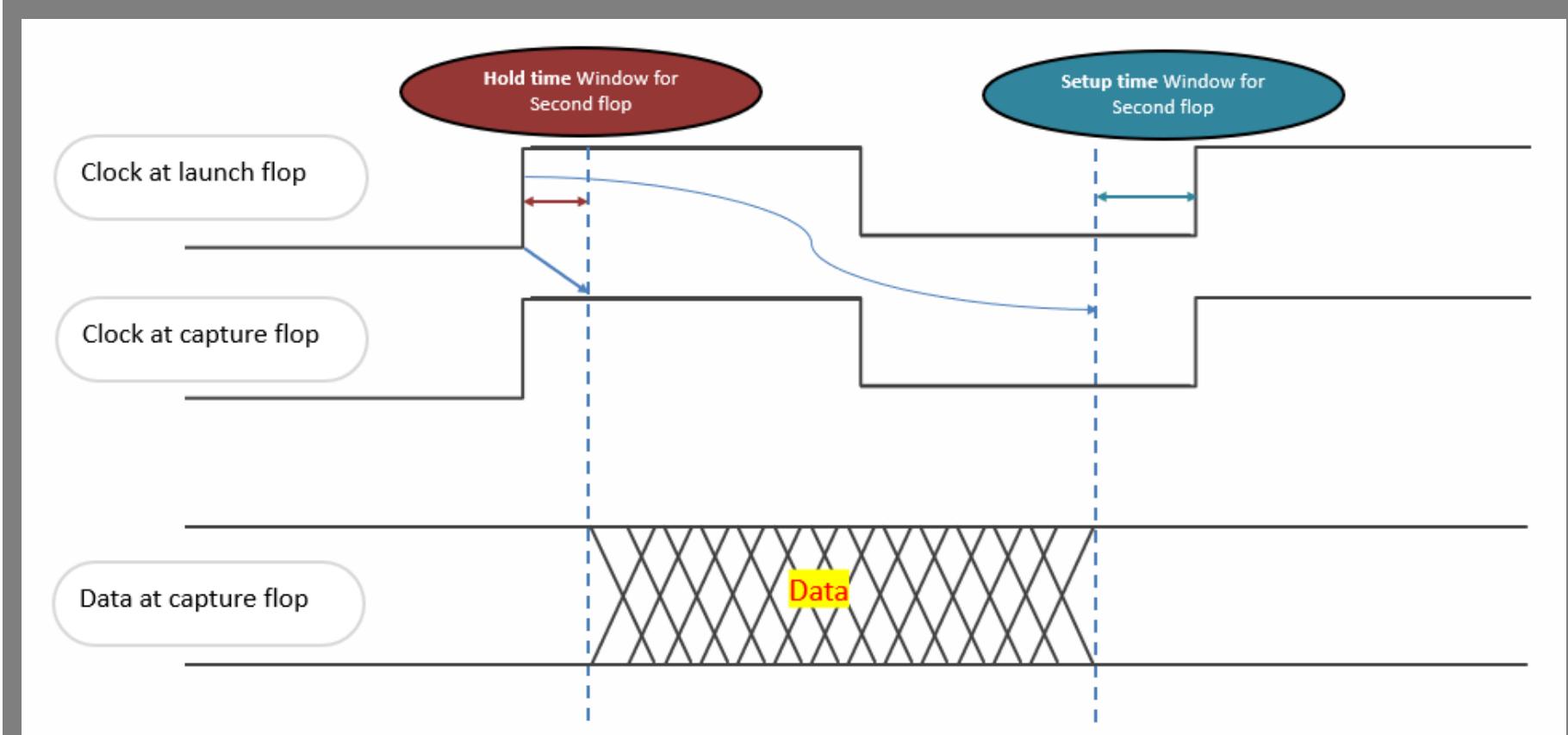


# Clock Pessimism

Occurs from clock path re-convergence when the identical components are on both the source and destination clocks paths



# Introduction to Setup and Hold Timing



Data valid window = Clock period – Setup window – Hold window

$$\text{Start of data valid window} = T_{\text{launch}} + T_{\text{hold}}$$

$$\text{End of data valid window} = T_{\text{launch}} + T_{\text{period}} - T_{\text{setup}}$$

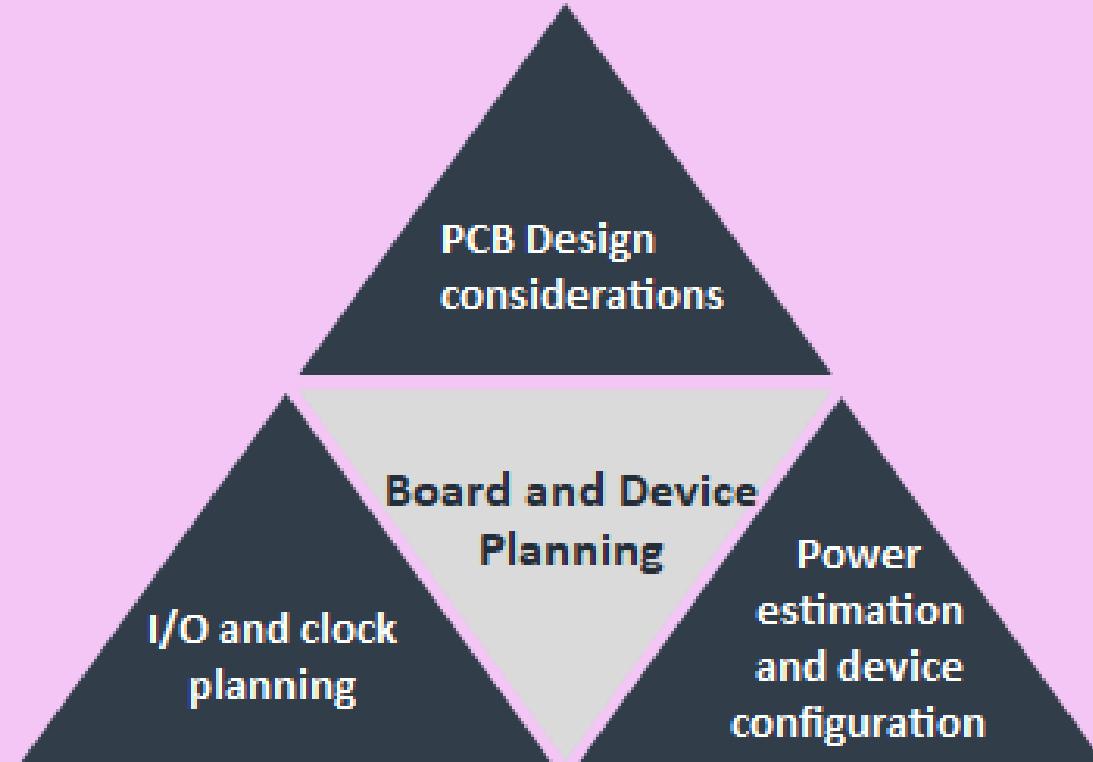


# Introduction to Vivado Reports

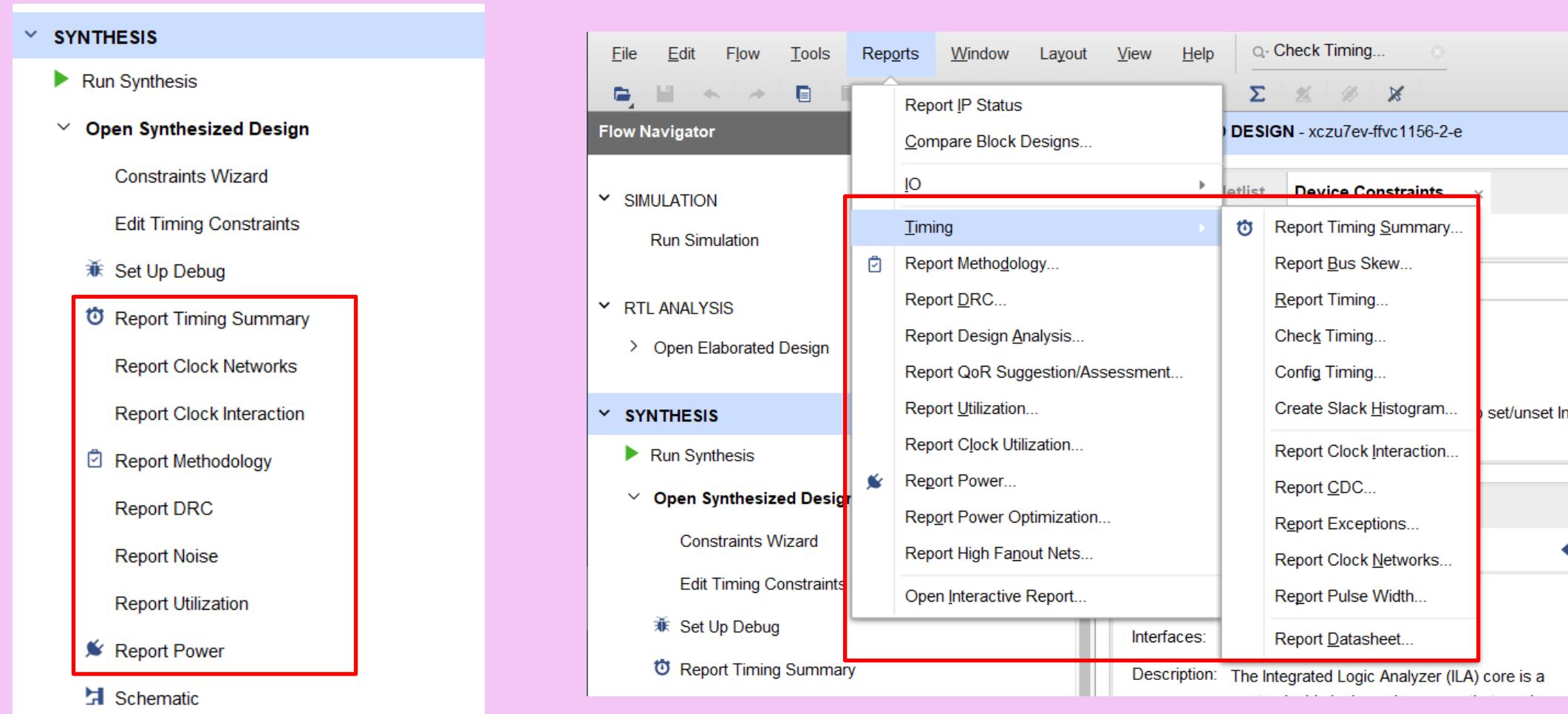
2022.2

# Board and Device Planning

Board and device planning includes:

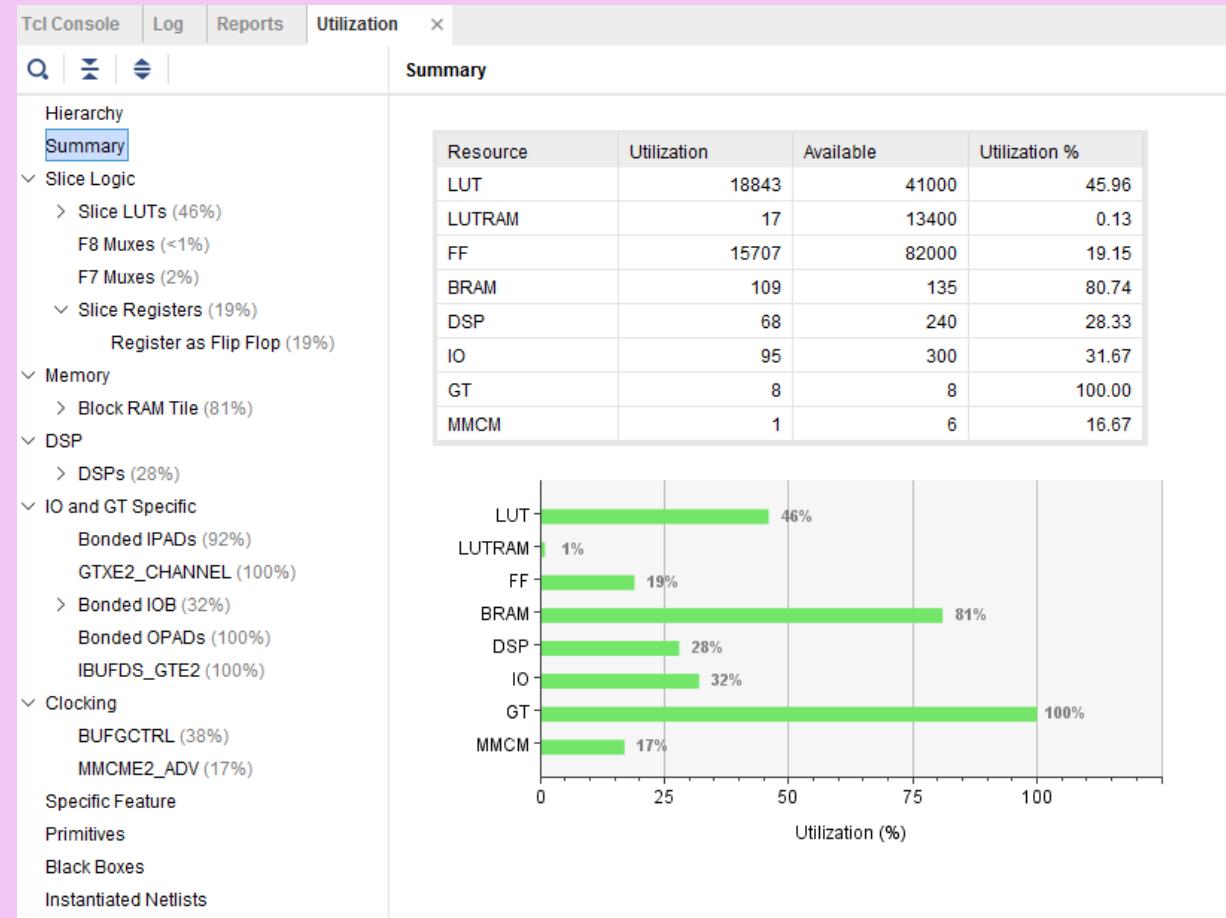


# Vivado Report



# Report Utilization

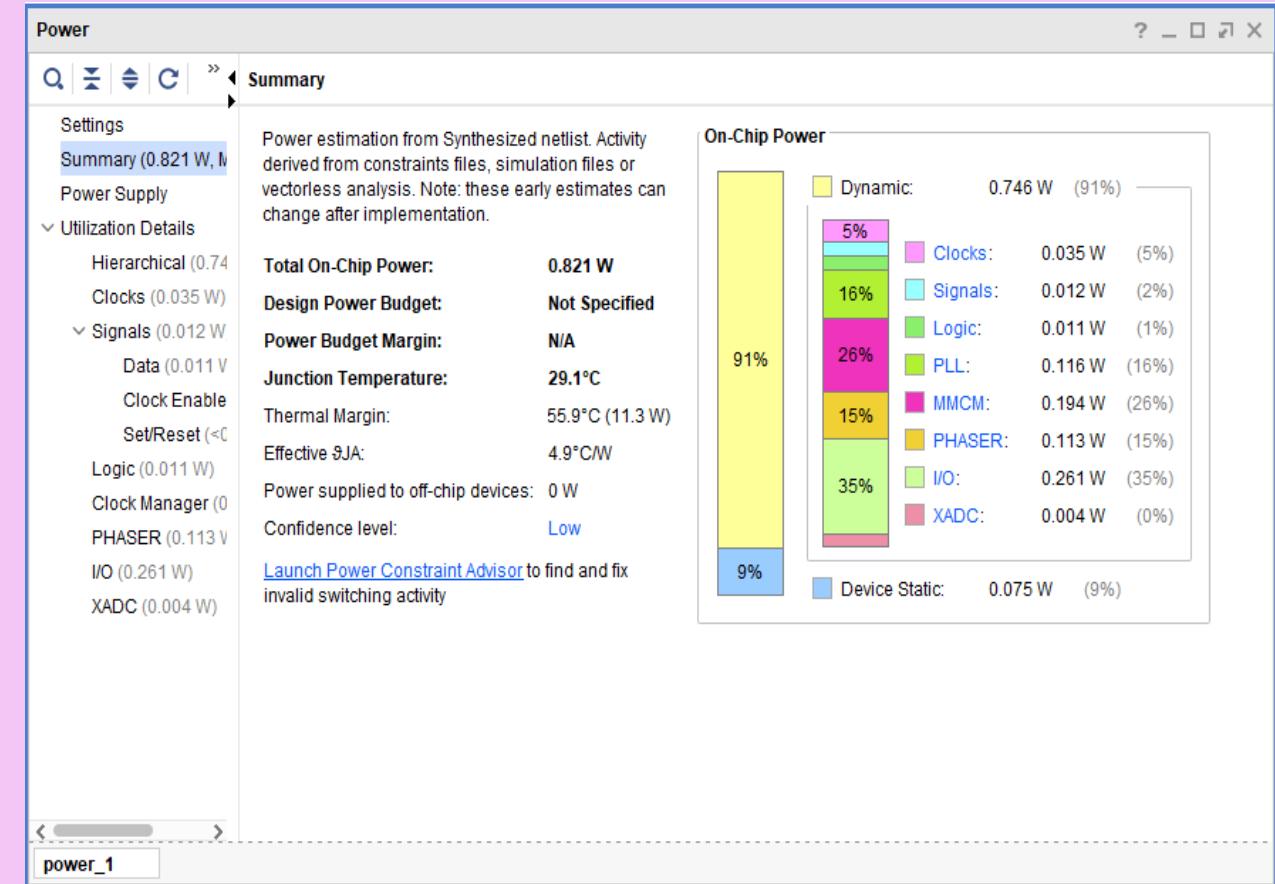
report\_utilization command reports  
LUTs, registers, FIFOs, clocking, and I/O resources  
utilization on the target part by the  
current synthesized or implemented design



report\_utilization

# Report Power

`report_power` command runs power estimation and displays the report



report\_power

# Report DRC

report\_drc command checks the design  
against the specified set of design rules  
Reports any errors or violations that are found

```
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 to
set_property CFGBVS value1 [current_design]
#where value1 is either VCCO or GND

set_property CONFIG_VOLTAGE value2 [current_design]
#where value2 is the voltage provided to configuration bank 0

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

report\_drc

# Report Timing Summary

`report_timing_summary` report is created automatically by the synthesis and implementation runs.

The report gives very detailed information about each static timing path.

The screenshot shows the 'Timing' window in the Xilinx Vivado Design Suite. The left sidebar lists various timing-related sections: General Information, Timer Settings, Design Timing Summary (which is selected and highlighted in blue), Clock Summary (24), Methodology Summary, and Check Timing (4) which includes sub-options like no\_clock (0) and constant\_clock (0). The main pane displays the 'Design Timing Summary' with three columns: Setup, Hold, and Pulse Width. Under Setup, it shows Worst Negative Slack (WNS) as 1.097 ns, Total Negative Slack (TNS) as 0.000 ns, Number of Failing Endpoints as 0, and Total Number of Endpoints as 12095. Under Hold, it shows Worst Hold Slack (WHS) as -1.636 ns, Total Hold Slack (THS) as -26.808 ns, Number of Failing Endpoints as 55, and Total Number of Endpoints as 12093. Under Pulse Width, it shows Worst Pulse Width Slack (WPWS) as 0.187 ns, Total Pulse Width Negative Slack (TPWS) as 0.000 ns, Number of Failing Endpoints as 0, and Total Number of Endpoints as 5102. A message at the bottom states 'Timing constraints are not met.' A status bar at the bottom indicates 'Timing Summary - timing\_1'.

report\_timing\_summary

# Report Timing Summary

Tcl Console   Messages   **Timing**   x   Power   Noise   Utilization   Debug

General Information  
Timer Settings  
Design Timing Summary  
Clock Summary (24)  
Methodology Summary  
Check Timing (4)  
Intra-Clock Paths  
Inter-Clock Paths  
clk\_out1\_clk\_wiz\_0 to clk\_pll\_i  
Setup 19.745 ns (10)  
Hold -1.636 ns (10)

Inter-Clock Paths - clk\_out1\_clk\_wiz\_0 to clk\_pll\_i - Hold

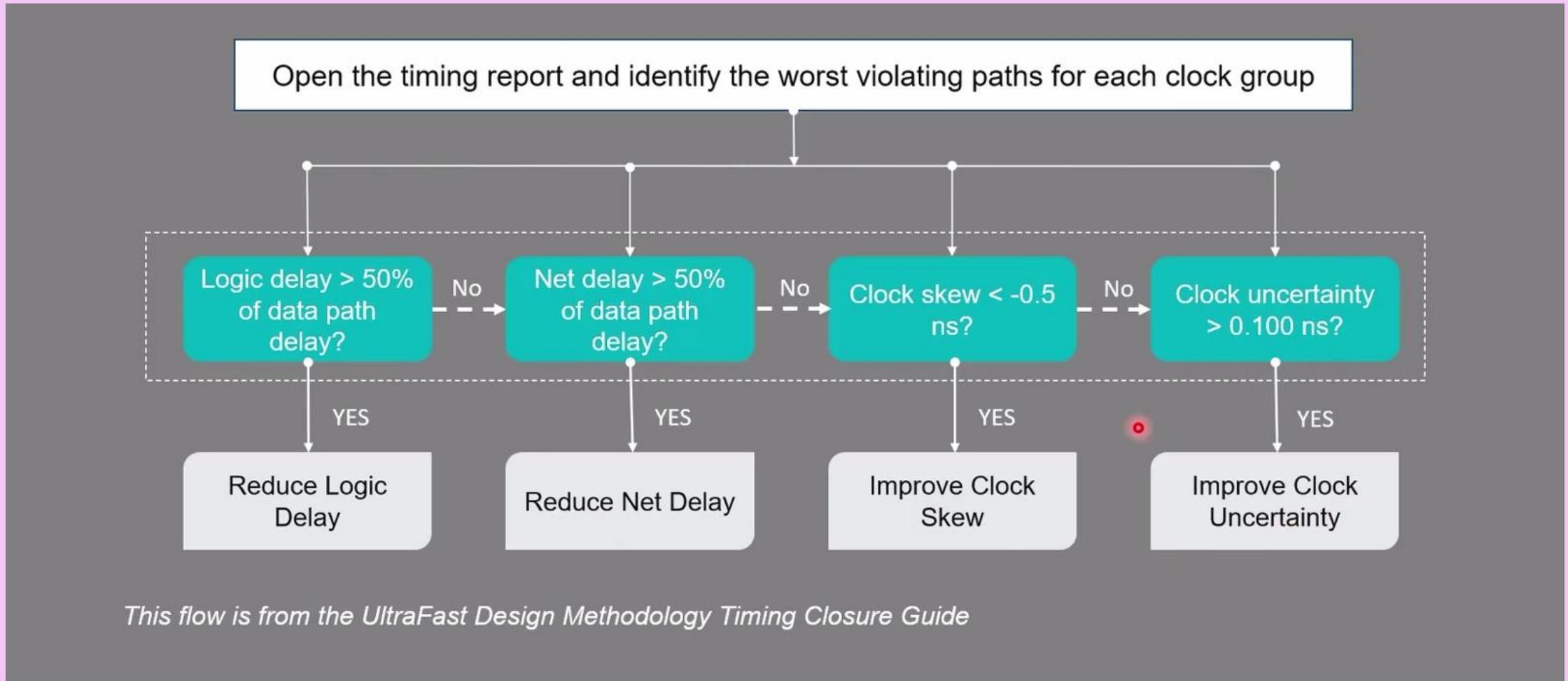
Name	Slack ^1	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock
Path 229	-1.636	0	1	1	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/xadc_supplied_temperature.temperature_reg[0]/C	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/device_temp_sync_r1_reg[0]/D	0.684	0.367	0.317	0.0	clk_out1_clk_wiz_0	clk_pll_i
Path 230	-1.636	0	1	1	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/xadc_supplied_temperature.temperature_reg[10]/C	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/device_temp_sync_r1_reg[10]/D	0.684	0.367	0.317	0.0	clk_out1_clk_wiz_0	clk_pll_i
Path 231	-1.636	0	1	1	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/xadc_supplied_temperature.temperature_reg[11]/C	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/device_temp_sync_r1_reg[11]/D	0.684	0.367	0.317	0.0	clk_out1_clk_wiz_0	clk_pll_i
Path 232	-1.636	0	1	1	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/xadc_supplied_temperature.temperature_reg[1]/C	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/device_temp_sync_r1_reg[1]/D	0.684	0.367	0.317	0.0	clk_out1_clk_wiz_0	clk_pll_i
Path 233	-1.636	0	1	1	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/xadc_supplied_temperature.temperature_reg[2]/C	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/device_temp_sync_r1_reg[2]/D	0.684	0.367	0.317	0.0	clk_out1_clk_wiz_0	clk_pll_i
Path 234	-1.636	0	1	1	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/xadc_supplied_temperature.temperature_reg[3]/C	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/device_temp_sync_r1_reg[3]/D	0.684	0.367	0.317	0.0	clk_out1_clk_wiz_0	clk_pll_i
Path 235	-1.636	0	1	1	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/xadc_supplied_temperature.temperature_reg[4]/C	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/device_temp_sync_r1_reg[4]/D	0.684	0.367	0.317	0.0	clk_out1_clk_wiz_0	clk_pll_i
Path 236	-1.636	0	1	1	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/xadc_supplied_temperature.temperature_reg[5]/C	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/device_temp_sync_r1_reg[5]/D	0.684	0.367	0.317	0.0	clk_out1_clk_wiz_0	clk_pll_i
Path 237	-1.636	0	1	1	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/xadc_supplied_temperature.temperature_reg[6]/C	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/device_temp_sync_r1_reg[6]/D	0.684	0.367	0.317	0.0	clk_out1_clk_wiz_0	clk_pll_i
Path 238	-1.636	0	1	1	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/xadc_supplied_temperature.temperature_reg[7]/C	u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/device_temp_sync_r1_reg[7]/D	0.684	0.367	0.317	0.0	clk_out1_clk_wiz_0	clk_pll_i

Summary

Name	Path 229
Slack (Hold)	-1.636ns
Source	u_mig_7series_0/u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/xadc_supplied_temperature.temperature_reg[0]/C (rising edge-triggered cell FDRE clocked by clk_out1_clk_wiz_0 {rise@0.000ns fall@6.000ns period=1.636ns})
Destination	u_mig_7series_0/u_mig_7series_0/mig/temp_mon_enabled/u_tempmon/device_temp_sync_r1_reg[0]/D (rising edge-triggered cell FDRE clocked by clk_pll_i {rise@0.000ns fall@6.000ns period=1.636ns})
Path Group	clk_pll_i
Path Type	Hold (Min at Slow Process Corner)
Requirement	0.000ns (clk_pll_i rise@0.000ns - clk_out1_clk_wiz_0 rise@0.000ns)
Data Path Delay	0.684ns (logic 0.367ns (53.631%) route 0.317ns (46.369%))
Logic Levels	0
Clock Path Skew	2.004ns
Clock Uncertainty	0.260ns

report\_timing\_summary

# Report Timing Summary



# Report Timing Summary

## Check Timing

Check ports, pins and paths under timing constraints

This will check the probabilistic problem  
(design data, timing constraints)  
before running report timing

This will find any delays of clocks,  
inputs/outputs and partial inputs/outputs

This also will find out generated clocks,  
loops or multiple clocks

Clock Summary				
Name	Waveform	Period (ns)	Frequency (MHz)	
sysclk_i	{0.000 41.666}	83.333	12.000	
clk_out1_clk_wiz_0	{0.000 2.500}	5.000	200.001	
clk_out2_clk_wiz_0	{0.000 2.000}	4.000	250.001	
freq_refclk	{0.000 0.750}	1.500	666.669	
u_memc_ui_top_std/mem_intf0/ddr_phy	{0.000 1.500}	3.000	333.334	
iserdes_clkdiv	{0.000 6.000}	12.000	83.334	
u_memc_ui_top_std/mem_intf0/ddr_phy	{0.000 1.500}	3.000	333.334	
iserdes_clkdiv_1	{0.000 6.000}	12.000	83.334	

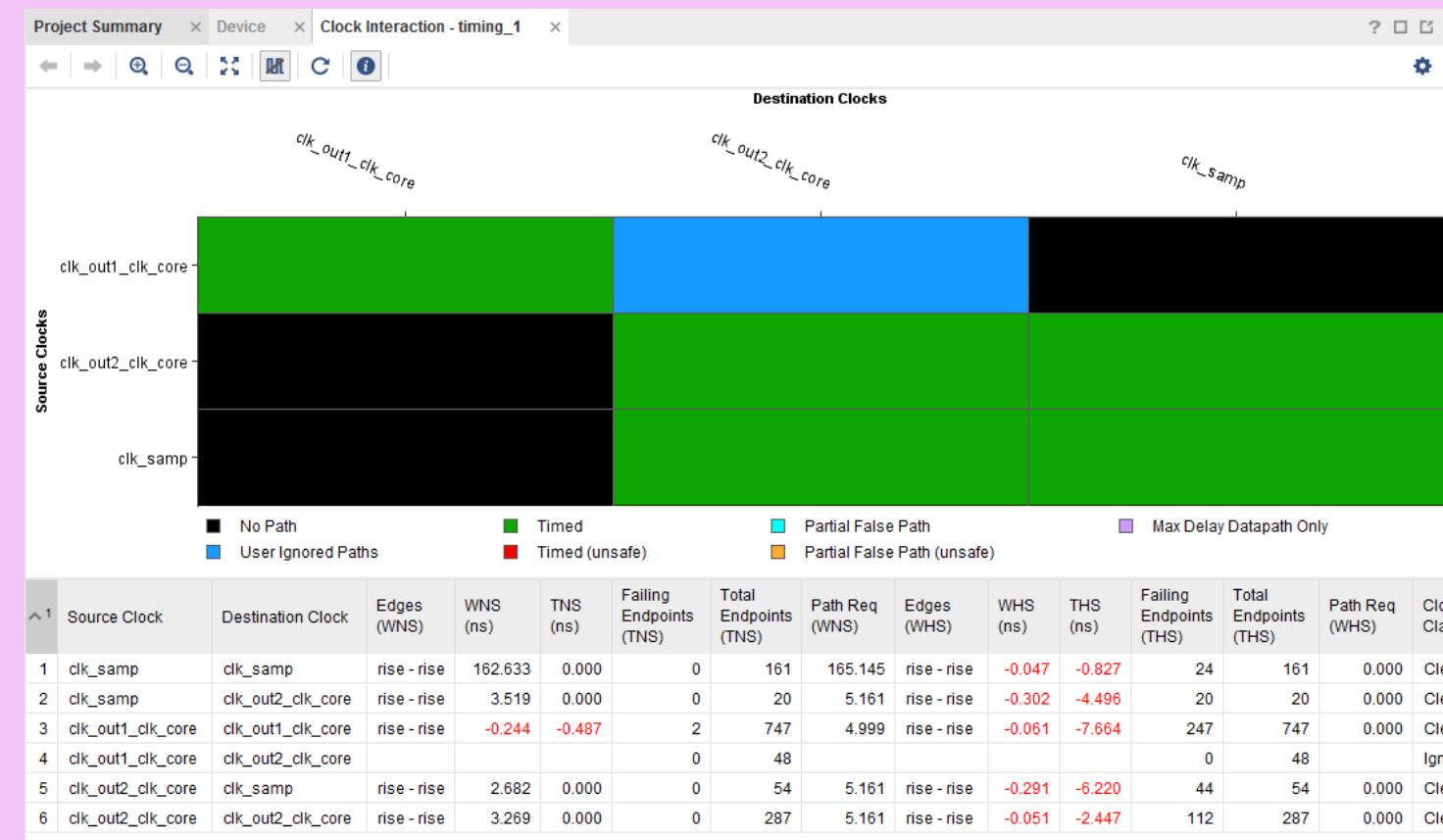
Check Timing			
Timing Check	Count	Worst Severity	
no_output_delay	3	High	
no_input_delay	1	High	
no_clock	0		
constant_clock	0		
pulse_width_clock	0		
unconstrained_internal_endpoints	0		
multiple_clock	0		
generated_clocks	0		
loops	0		
partial_input_delay	0		
partial_output_delay	0		

report\_timing\_summary

# Report Clock Interaction

`report_clock_interaction` command

Analyzes timing paths that cross from  
one clock domain into another clock domain



report\_clock\_interaction



# Introduction to the Vivado Logic Analyzer

2022.2

# Vivado Logic Analyzer Tool

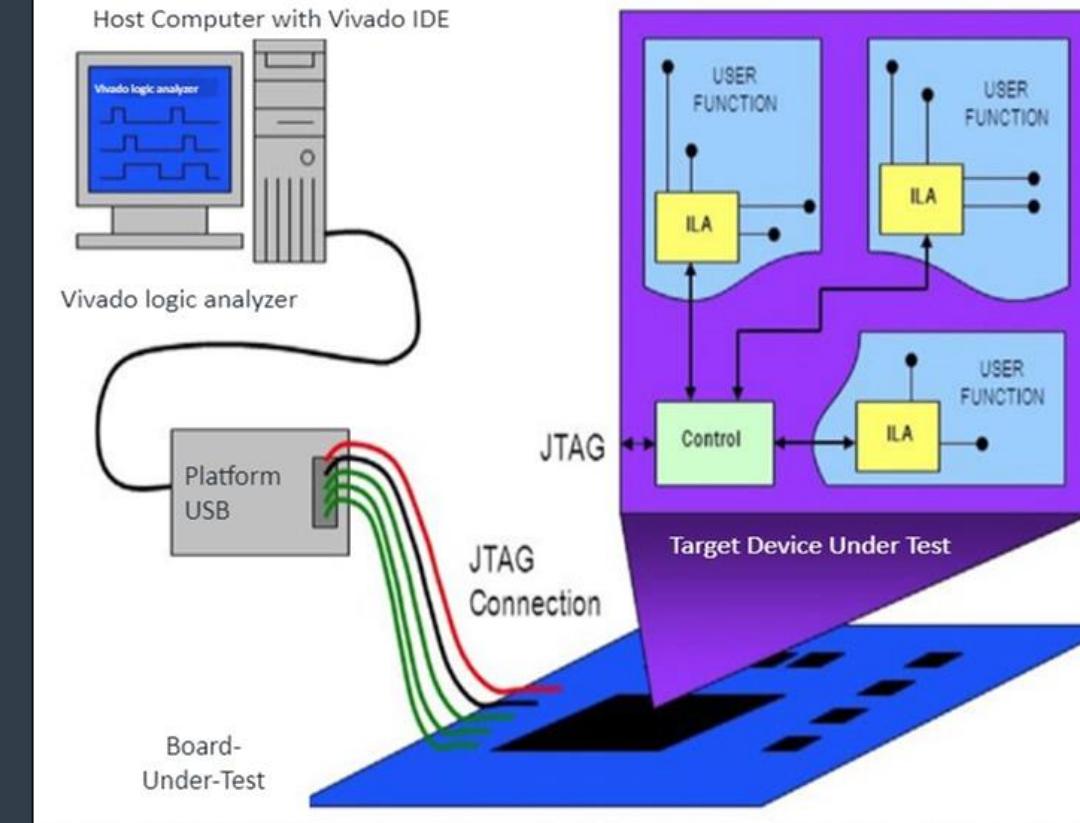
Powerful yet easy-to-use debug tool

Synergistic role with simulation

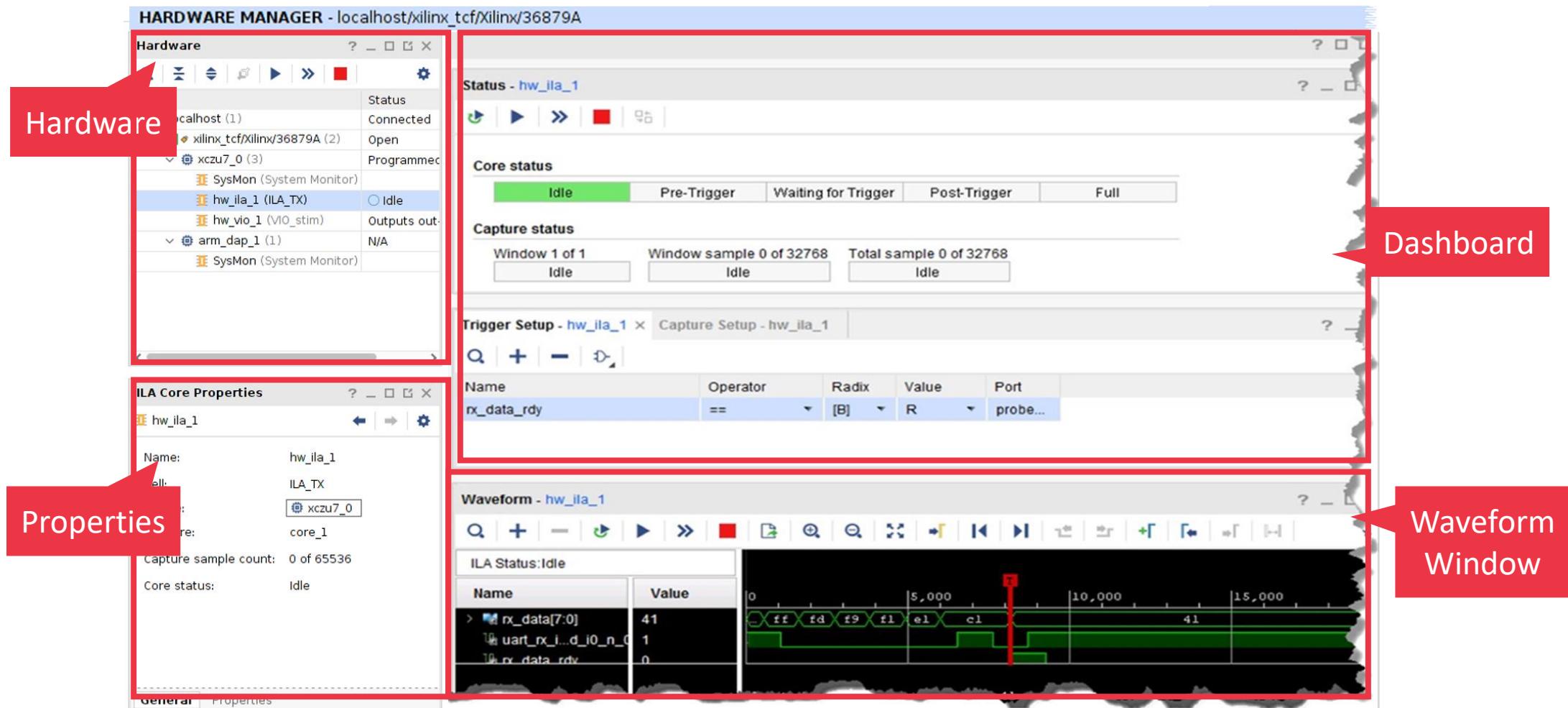
The Vivado logic analyzer can be used in three contexts:

- Verification
- Debugging
- Data capture

Interact with debug IP cores



# Vivado Logic Analyzer Tool



Vivado Logic Analyzer Tool

# VLA Comprised of...

Comprised of two basic items:

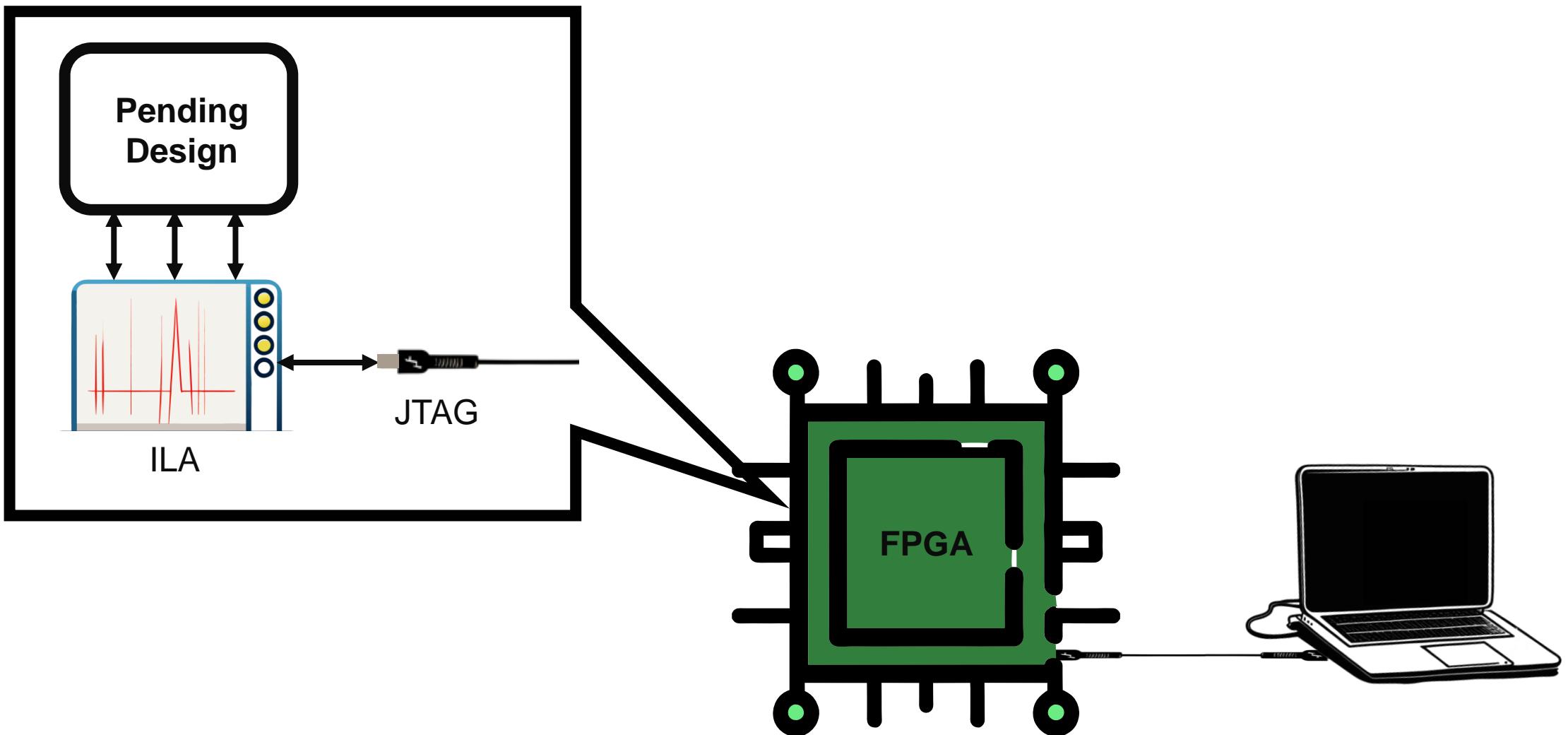
Debug cores:

- Includes ILA, System ILA, VIO, JTAG to AXI Master, and IBERT, etc.

Debug flows:

- Includes the HDL instantiation and netlist insertion flows

# Integrated Logic Analyzer

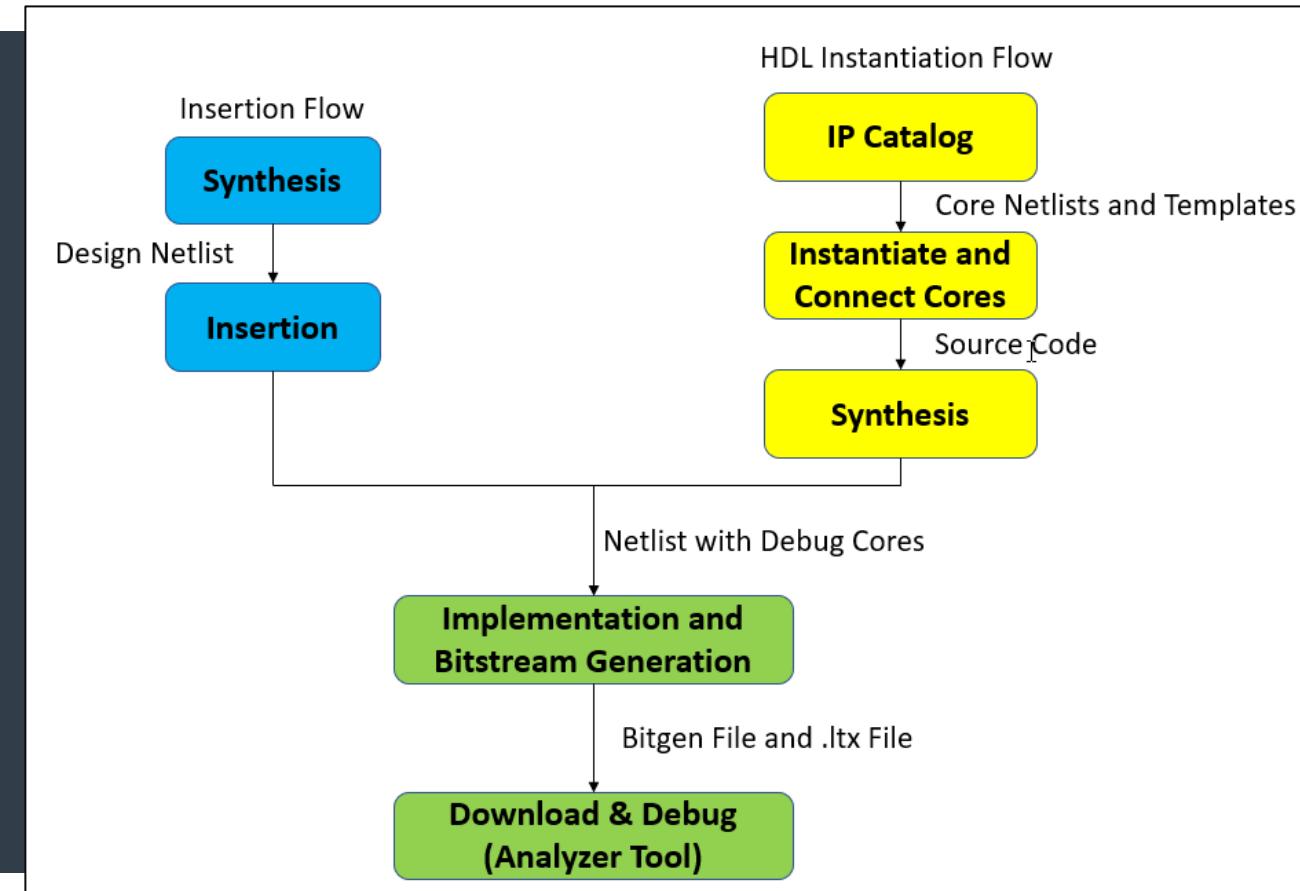


# Vivado Logic Analyzer Probing Flows

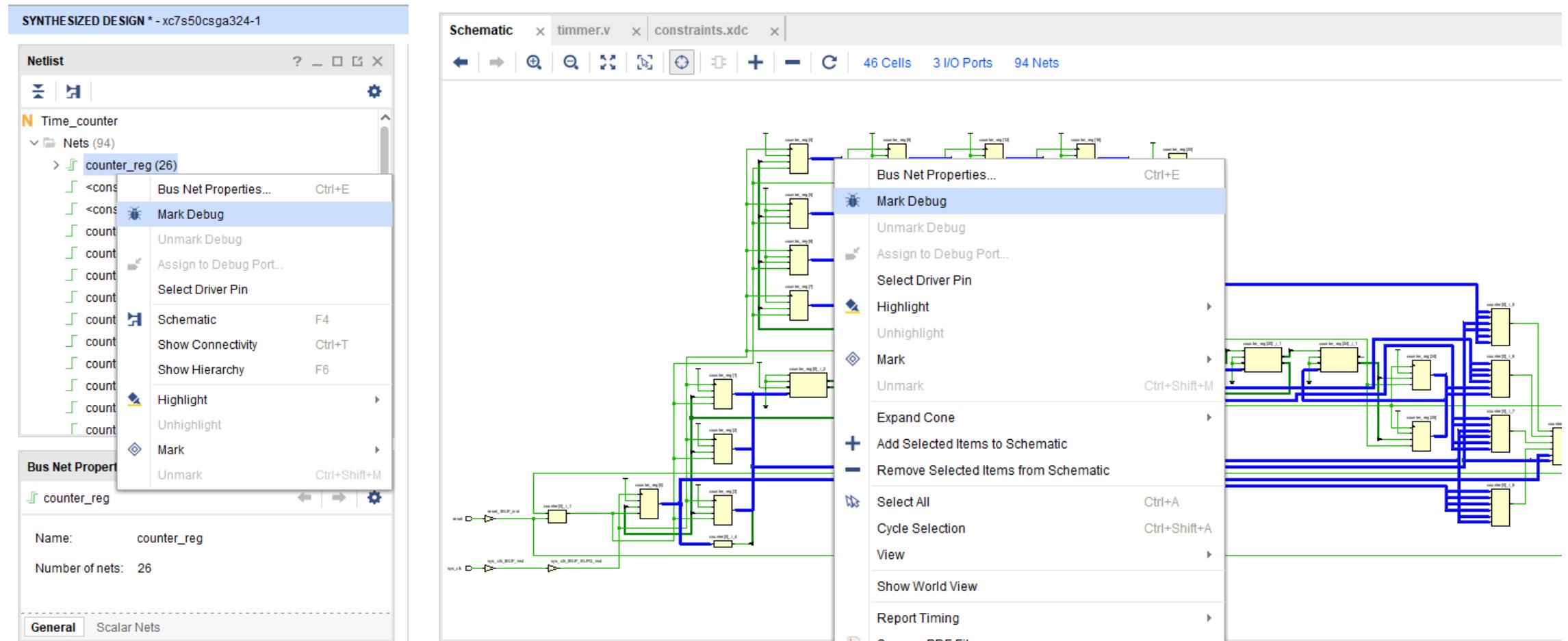
Two Vivado logic analyzer probing flows:

- HDL instantiation flow
- Netlist insertion flow

HDL instantiation flow and netlist insertion flows can be mixed



# Add debug mark in the netlist

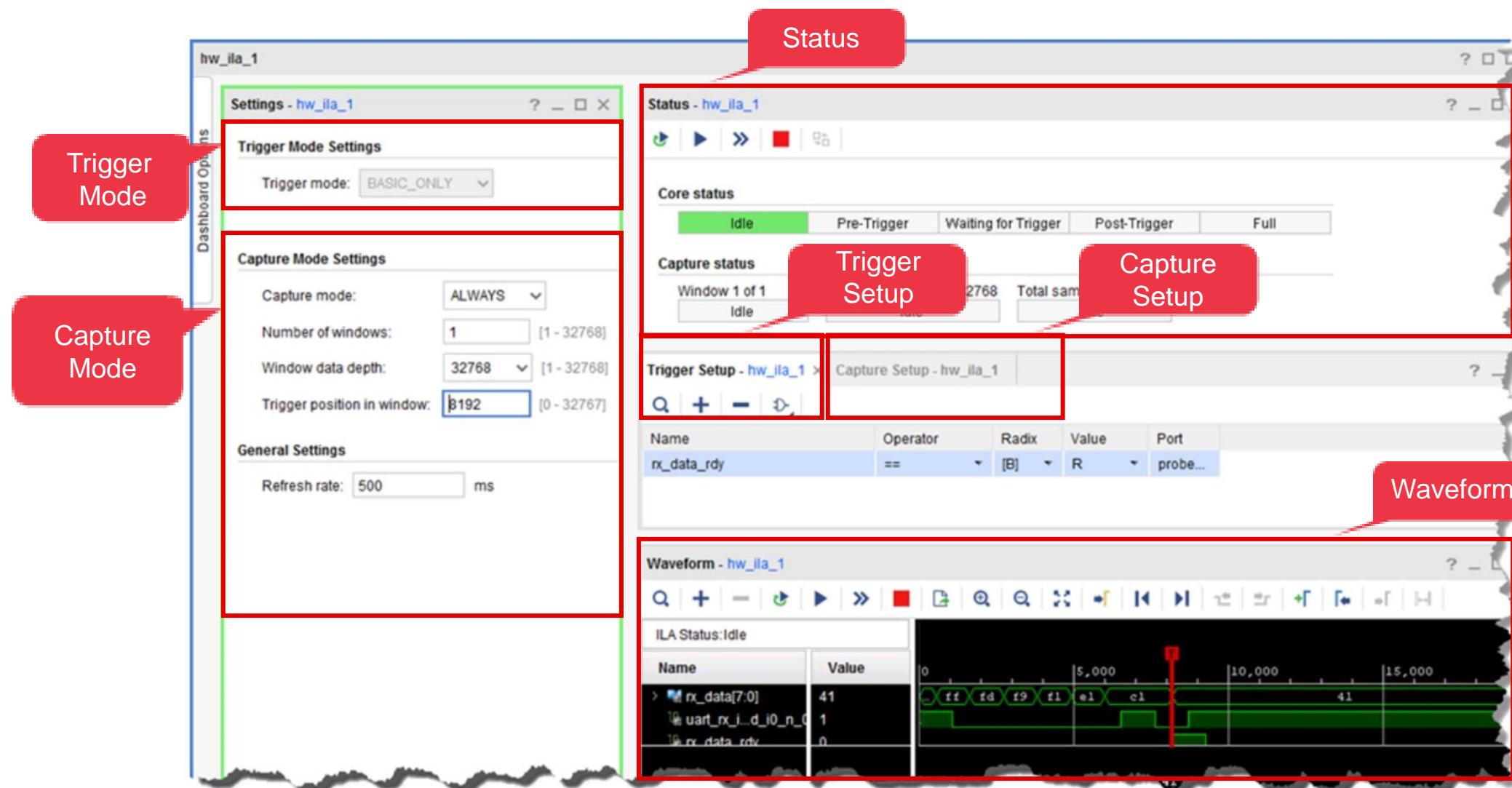


# Add debug mark in the netlist

The screenshot shows the Xilinx Vivado IDE interface. On the left, the 'SYNTHESIS' section of the navigation menu is highlighted. A red arrow points from the 'SYNTHESIS' menu to the 'Set Up Debug' window. The 'Set Up Debug' window is titled 'Set Up Debug' and contains a table titled 'Nets to Debug'. The table lists three nets: 'counter\_reg (26)', 'counter\_flag', and 'led\_out\_OBUF'. Each row includes columns for 'Name', 'Clock Domain', 'Driver Cell', and 'Probe Type'. Red arrows point to the 'Clock Domain' and 'Probe Type' columns. Below the table are buttons for 'Find Nets to Add...', 'Nets to debug: 28', '?', '< Back', 'Next >', 'Finish', and 'Cancel'.

Name	Clock Domain	Driver Cell	Probe Type
> <code>counter_reg (26)</code>	sys_clk_IBUF_BUFG	FDRE	Data and Trigger
<code>counter_flag</code>	sys_clk_IBUF_BUFG	FDRE	Data and Trigger
<code>led_out_OBUF</code>	sys_clk_IBUF_BUFG	FDRE	Data and Trigger

# ILA Dashboard



# ILA Dashboard – Status

The screenshot shows the ILA Dashboard interface for hw\_il\_1. The left side features the 'Settings - hw\_il\_1' tab, which includes sections for Trigger Mode Settings, Capture Mode Settings, and General Settings. The right side features the 'Status - hw\_il\_1' tab, which displays Core status, Capture status, and Trigger Setup.

**Settings - hw\_il\_1**

- Trigger Mode Settings:** Trigger mode: BASIC\_ONLY
- Capture Mode Settings:** Capture mode: ALWAYS, Number of windows: 1 [1-32768], Window data depth: 32768 [1-32768], Trigger position in window: 8192 [0-32767]
- General Settings:** Refresh rate: 500 ms

**Status - hw\_il\_1**

**Core status:** Idle

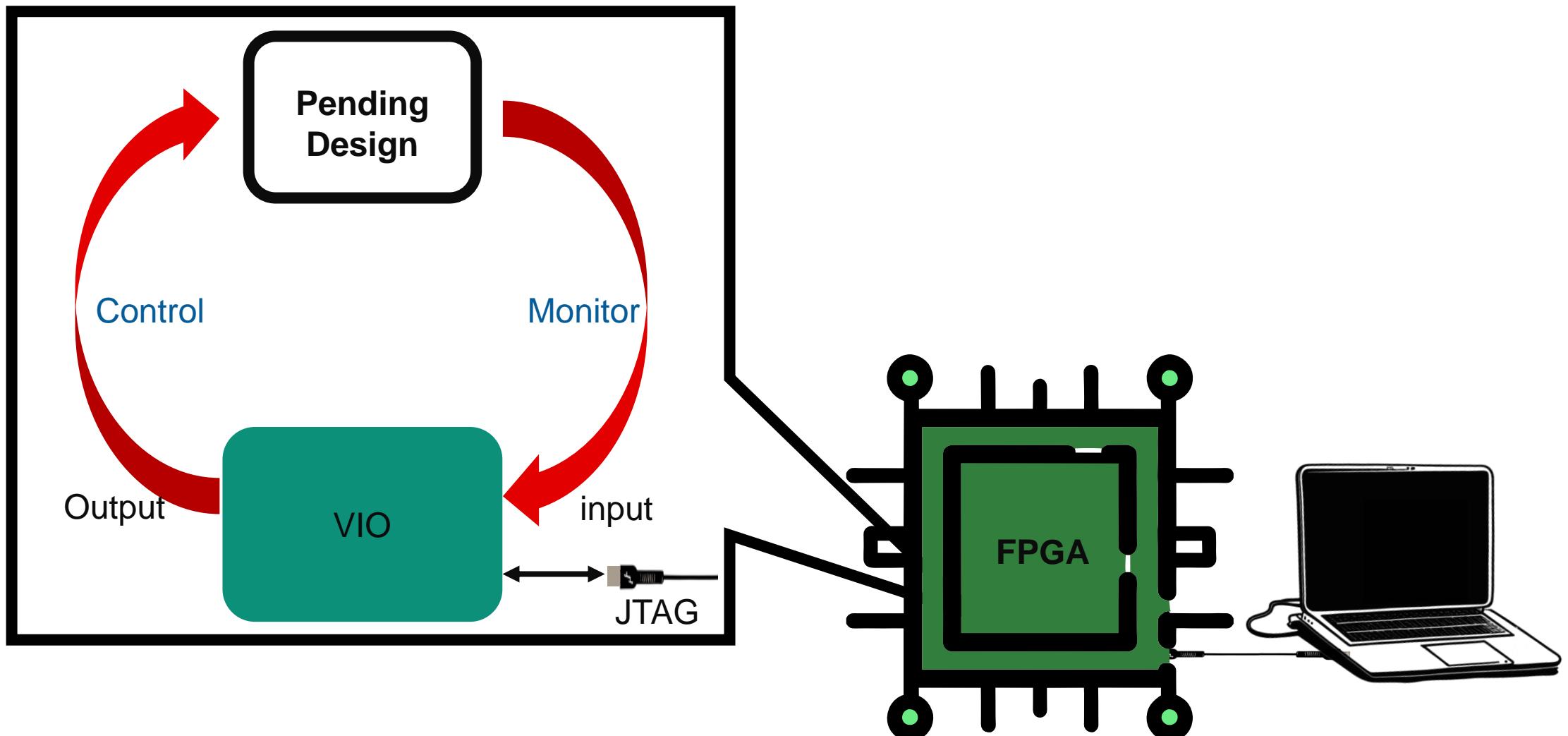
**Capture status:**

Window 1 of 1	Window sample 0 of 32768	Total sample 0 of 32768
Idle	Idle	Idle

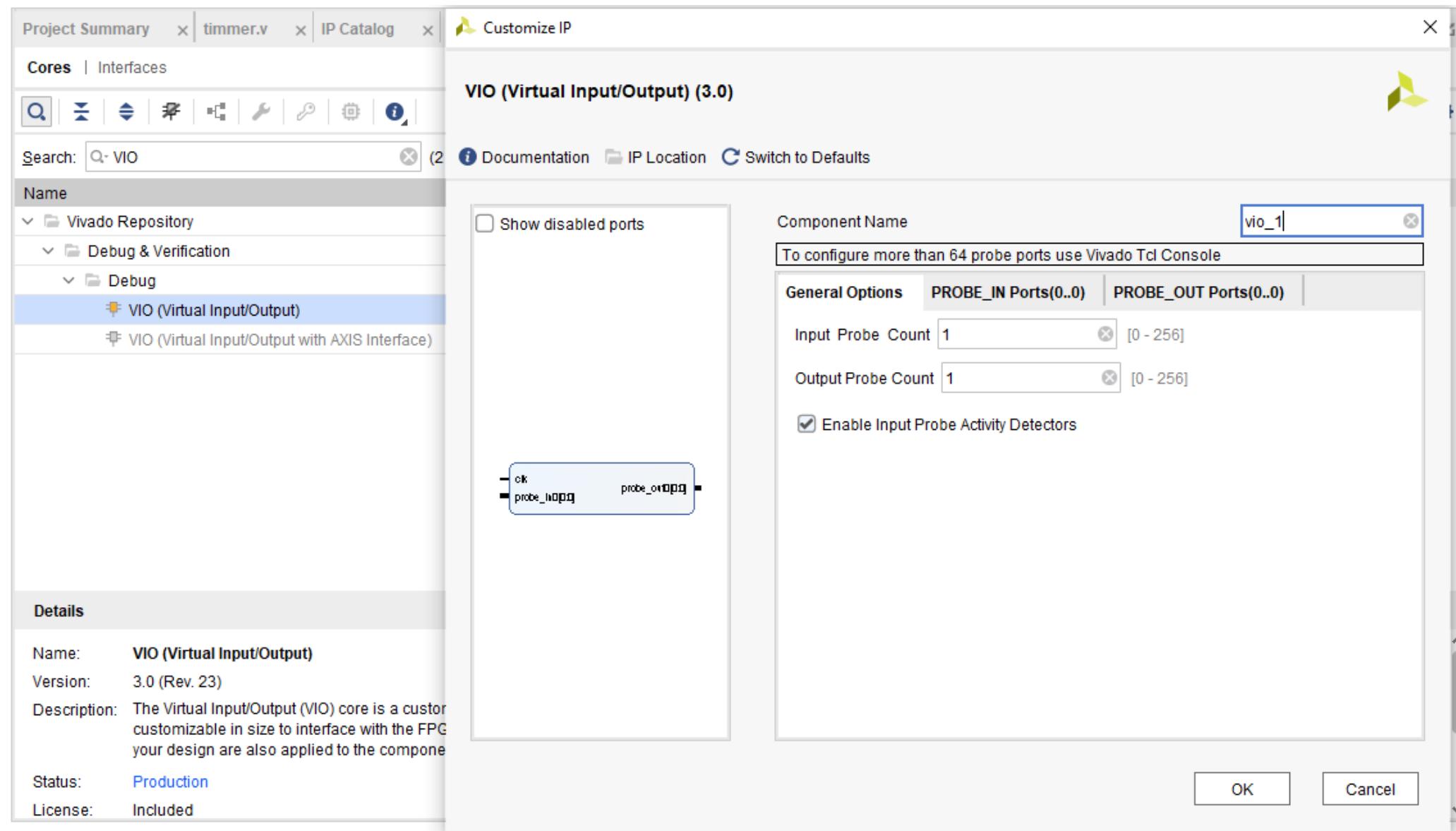
**Trigger Setup - hw\_il\_1**

Name	Operator	Radix	Value	Port
rx_data_rdy	==	[B]	R	probe...

# Virtual Input/Output



# Add VIO IP Core

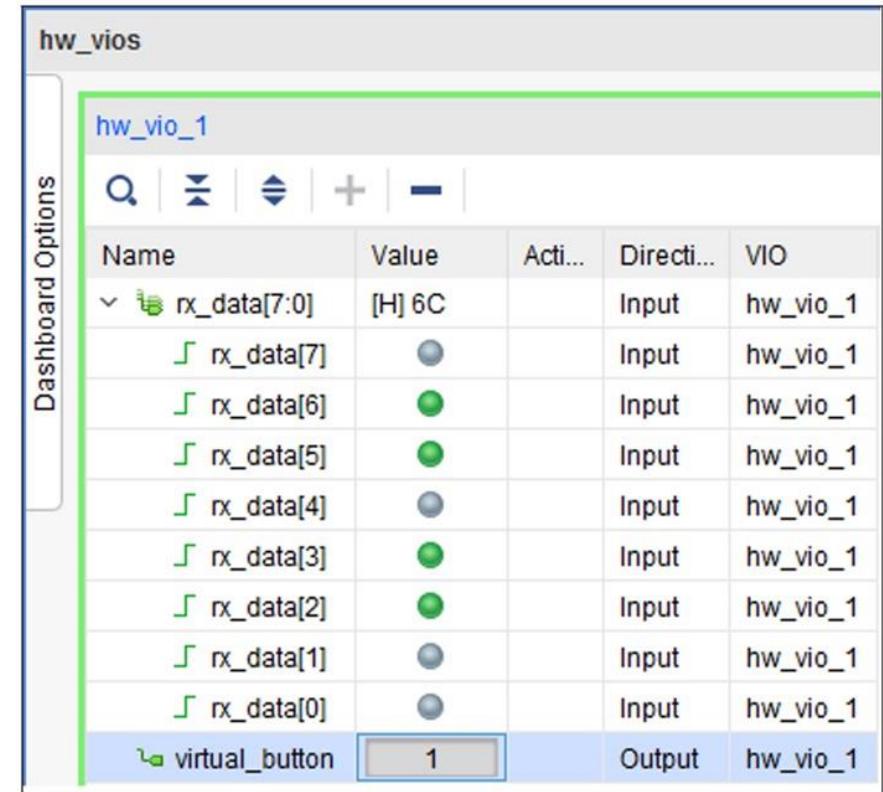


# Virtual Input/Output (VIO)

## VIO Ports and Parameters

- Shows all status and control information pertaining to a given VIO core
- Drag-and-drop VIO probes from the Debug Probes window
- Monitor “virtual” pins in your design
- Inputs appear as LEDs or text
- Outputs appear as toggle buttons

## VIO Dashboard



hw_vios				
hw_vio_1				
Name	Value	Action	Direction	VIO
rx_data[7:0]	[H] 6C		Input	hw_vio_1
rx_data[7]	■		Input	hw_vio_1
rx_data[6]	●		Input	hw_vio_1
rx_data[5]	●		Input	hw_vio_1
rx_data[4]	■		Input	hw_vio_1
rx_data[3]	●		Input	hw_vio_1
rx_data[2]	●		Input	hw_vio_1
rx_data[1]	■		Input	hw_vio_1
rx_data[0]	■		Input	hw_vio_1
virtual_button	1		Output	hw_vio_1

# VIO Debugging

Trigger Setup - hw\_il\_1 hw\_vio\_1 x Capture Setup - hw\_il\_1

Name	Value	Activity	Direction	VIO
> counter[25:0]	[U] 866794	↑↓	Input	hw_vio_1
long_flag	[B] 1		Input	hw_vio_1
short_flag	[B] 1		Input	hw_vio_1
led_o	Debug Probe Properties... Ctrl+E		Input	hw_vio_1
count	Text		Input	hw_vio_1
vbtn	LED...		Output	hw_vio_1

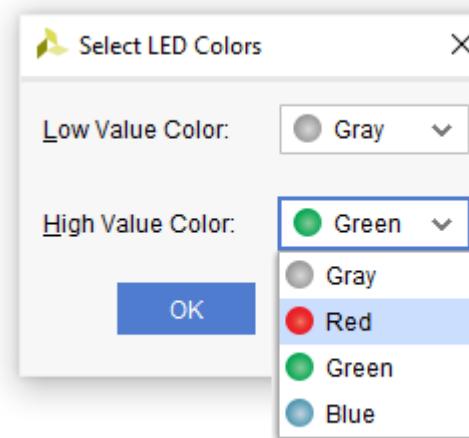
Radix

Rename...

Name

Remove Delete

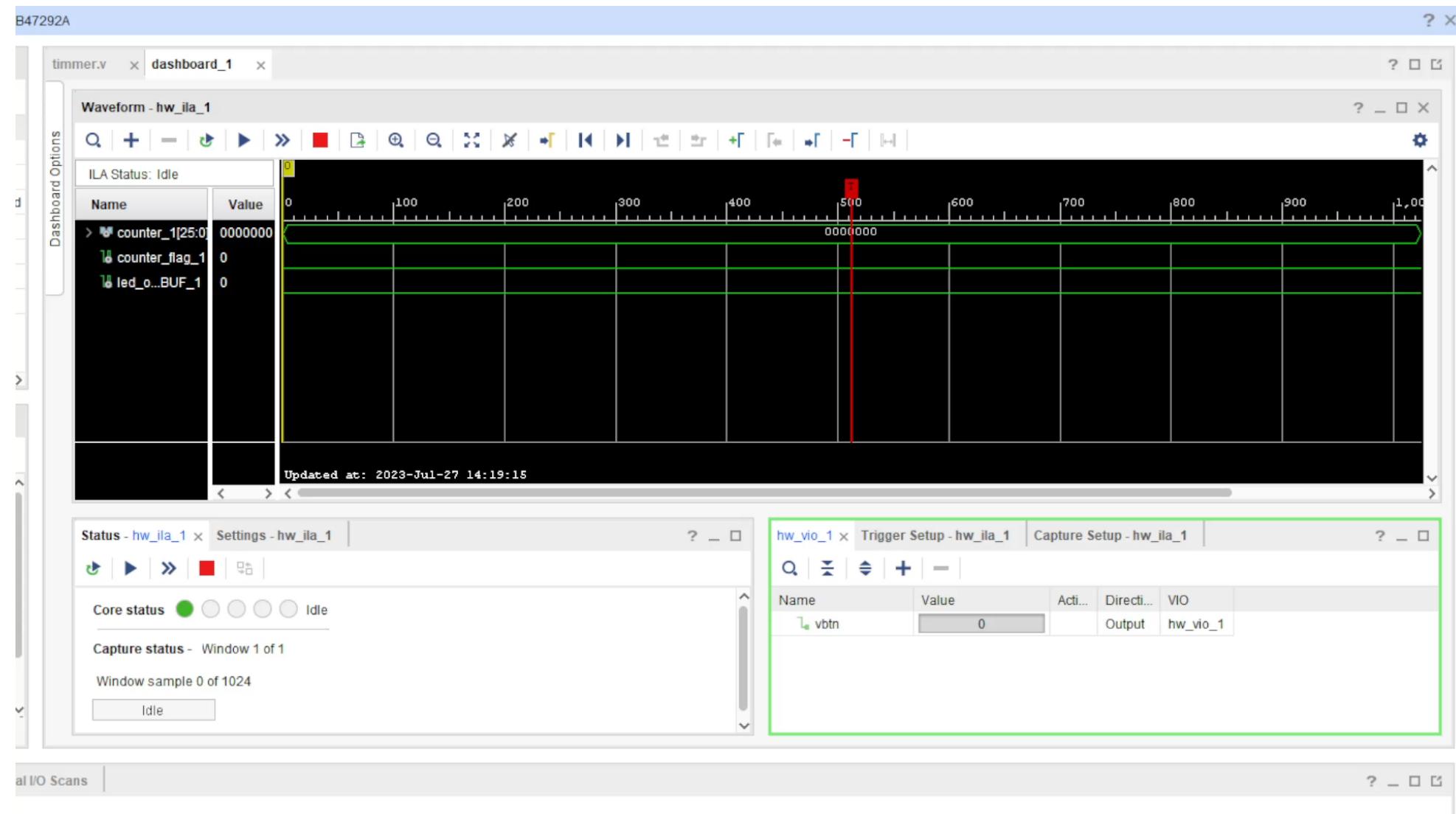
Export to Spreadsheet...



Trigger Setup - hw\_il\_1 hw\_vio\_1 x Capture Setup - hw\_il\_1

Name	Value	Activity	Direction	VIO
> counter[25:0]	[U] 409291	↑↓	Input	hw_vio_1
counter[18]	●	↑↓	Input	hw_vio_1
counter[17]	●	↑↓	Input	hw_vio_1
counter[13]	●	↑↓	Input	hw_vio_1
counter[12]	●	↑↓	Input	hw_vio_1
counter[11]	●	↑↓	Input	hw_vio_1
counter[10]	●	↑↓	Input	hw_vio_1
counter[9]	●	↑↓	Input	hw_vio_1
counter[7]	●	↑↓	Input	hw_vio_1
counter[6]	●	↑↓	Input	hw_vio_1
counter[3]	●	↑↓	Input	hw_vio_1
counter[1]	●	↑↓	Input	hw_vio_1
counter[0]	●	↑↓	Input	hw_vio_1
counter[25]	●		Input	hw_vio_1
counter[24]	●		Input	hw_vio_1
counter[23]	●		Input	hw_vio_1
counter[22]	●		Input	hw_vio_1
counter[21]	●		Input	hw_vio_1
counter[20]	●	↑↓	Input	hw_vio_1
counter[19]	●	↑↓	Input	hw_vio_1
counter[16]	●	↑↓	Input	hw_vio_1
counter[15]	●	↑↓	Input	hw_vio_1

# VIO Debugging





# Thank you very much for your attention!

Course Agenda  
2023