AMD
XILINX

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

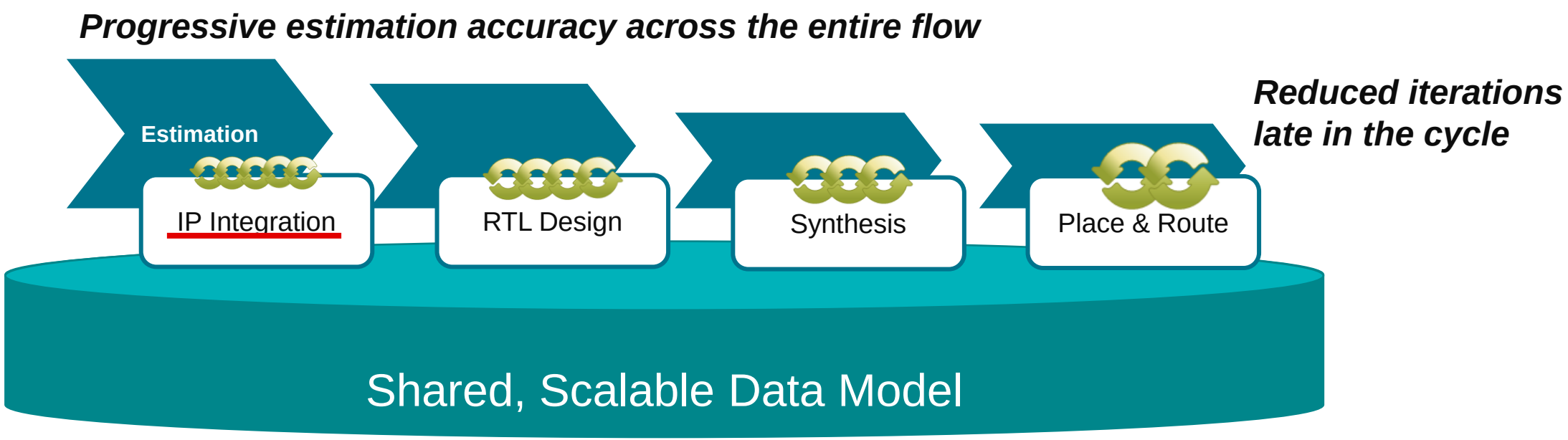
AMD
XILINX

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



AMD XILINX

Vivado System-Level Design Flows

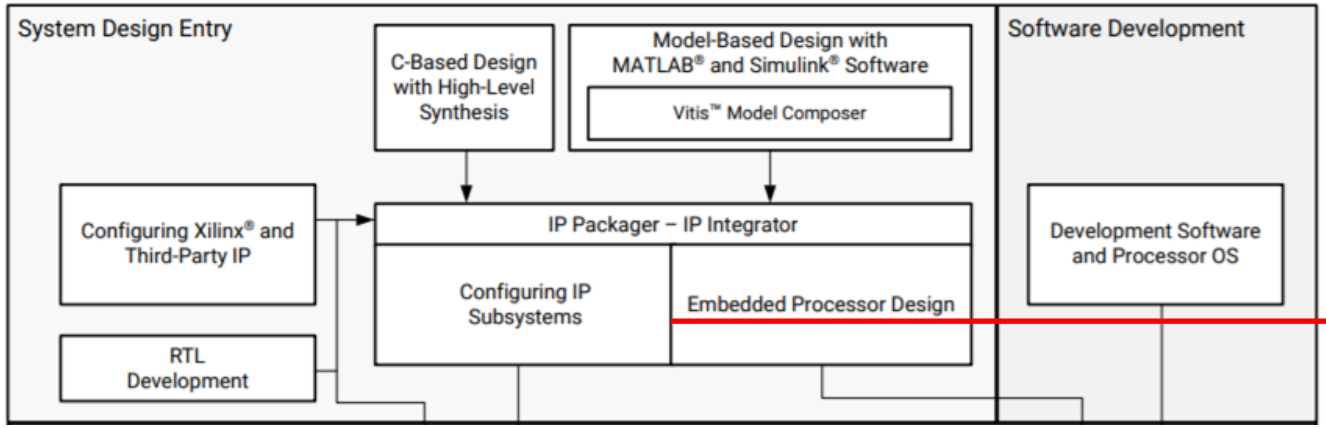
RTL-to-bitstream design flow

- 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

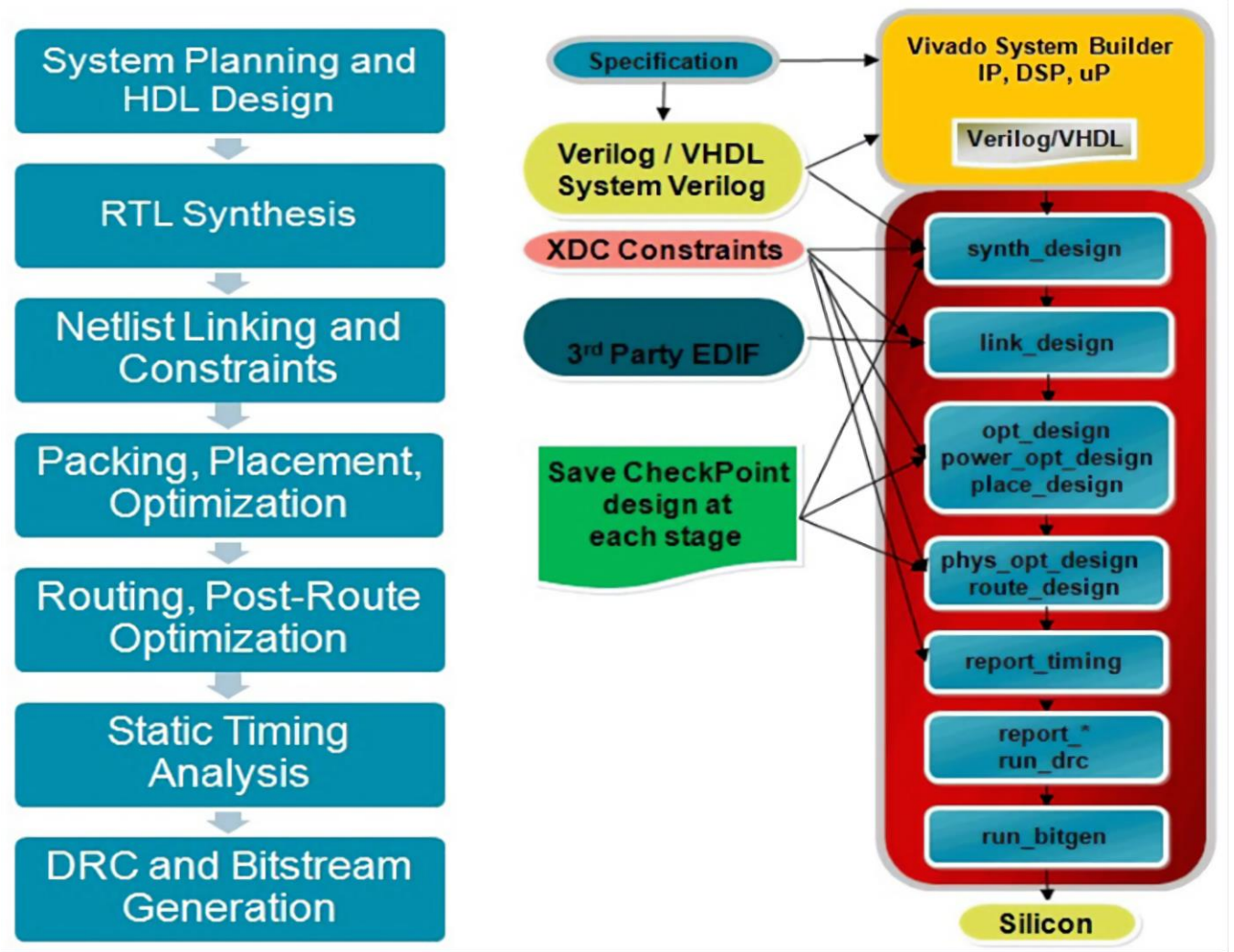


System-level Integration

- 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



AMD XILINX

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

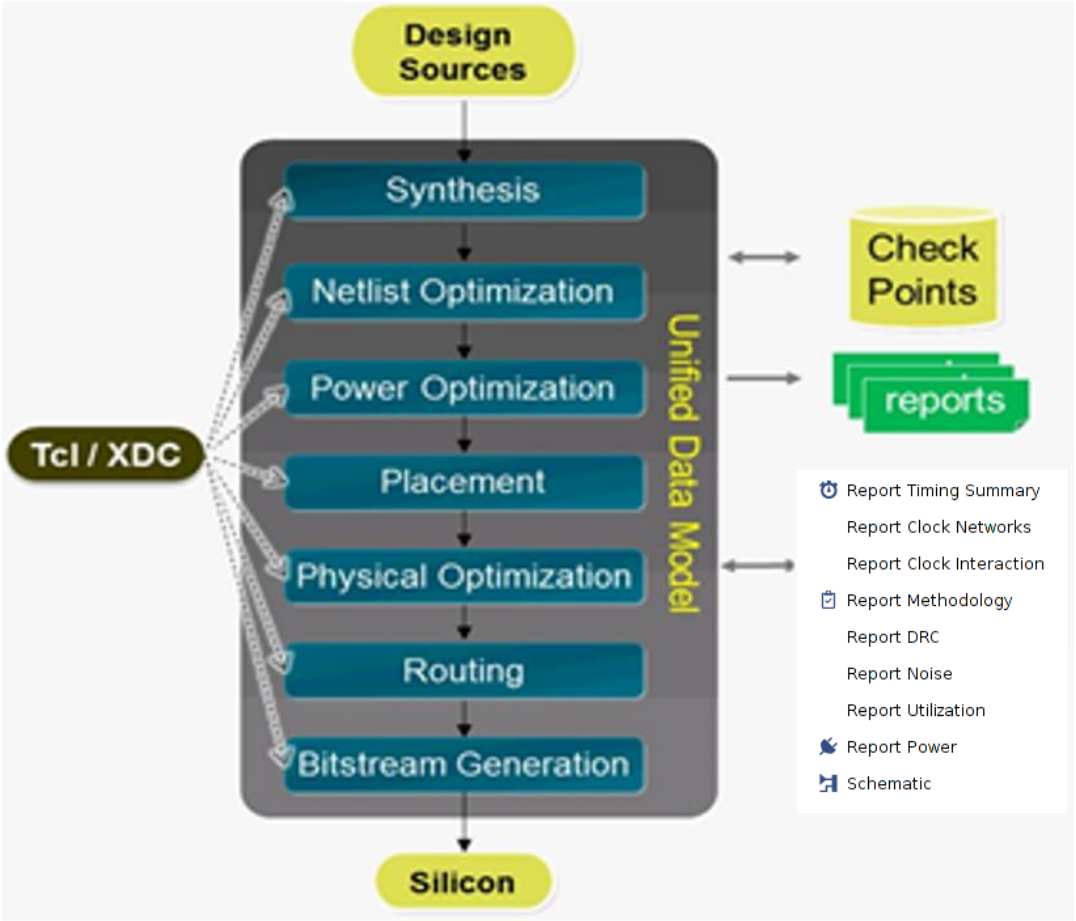
- Checkpoints only written if place/route information exists (after placement)
- Use the `link_design` Tcl command after `read_checkpoint` to initialize the design and load it in memory

Vivado Design Suite Use Models

Figure shows the Vivado system-level design flow

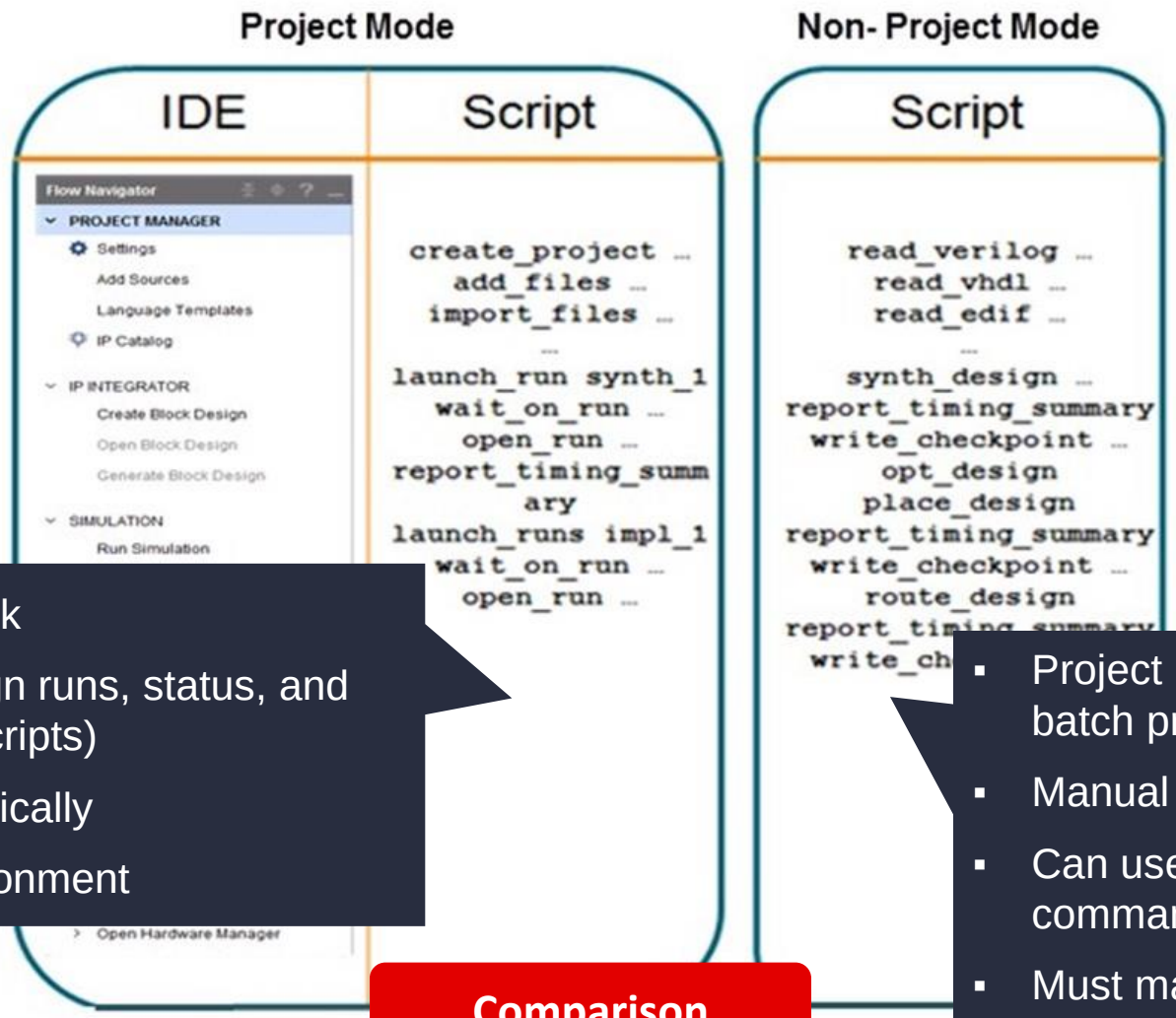
With two primary use models

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



AMD XILINX

Vivado Design Suite Use Models



- Creates project structure on disk
- The Vivado tool manages design runs, status, and data (can use GUI and/or Tcl scripts)
- Reports are generated automatically
- Robust interactive design environment

Comparison

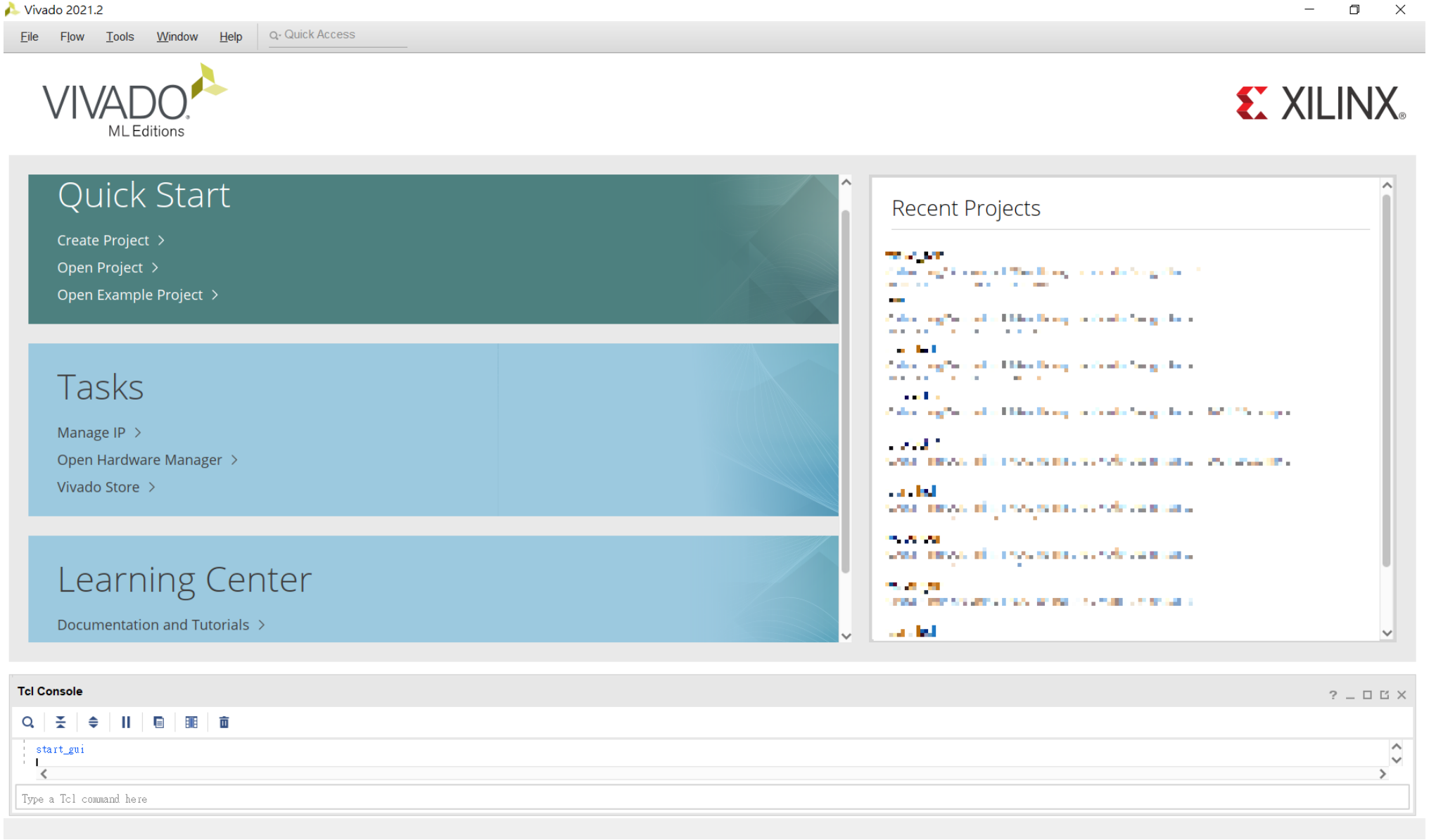
- Project is compiled in memory Tcl-based, batch processing
- Manual design source file management
- Can use GUI for visualization via start_gui command
- Must manually create reports and checkpoints

AMD
XILINX

Vivado Design Suite Project-Based Mode

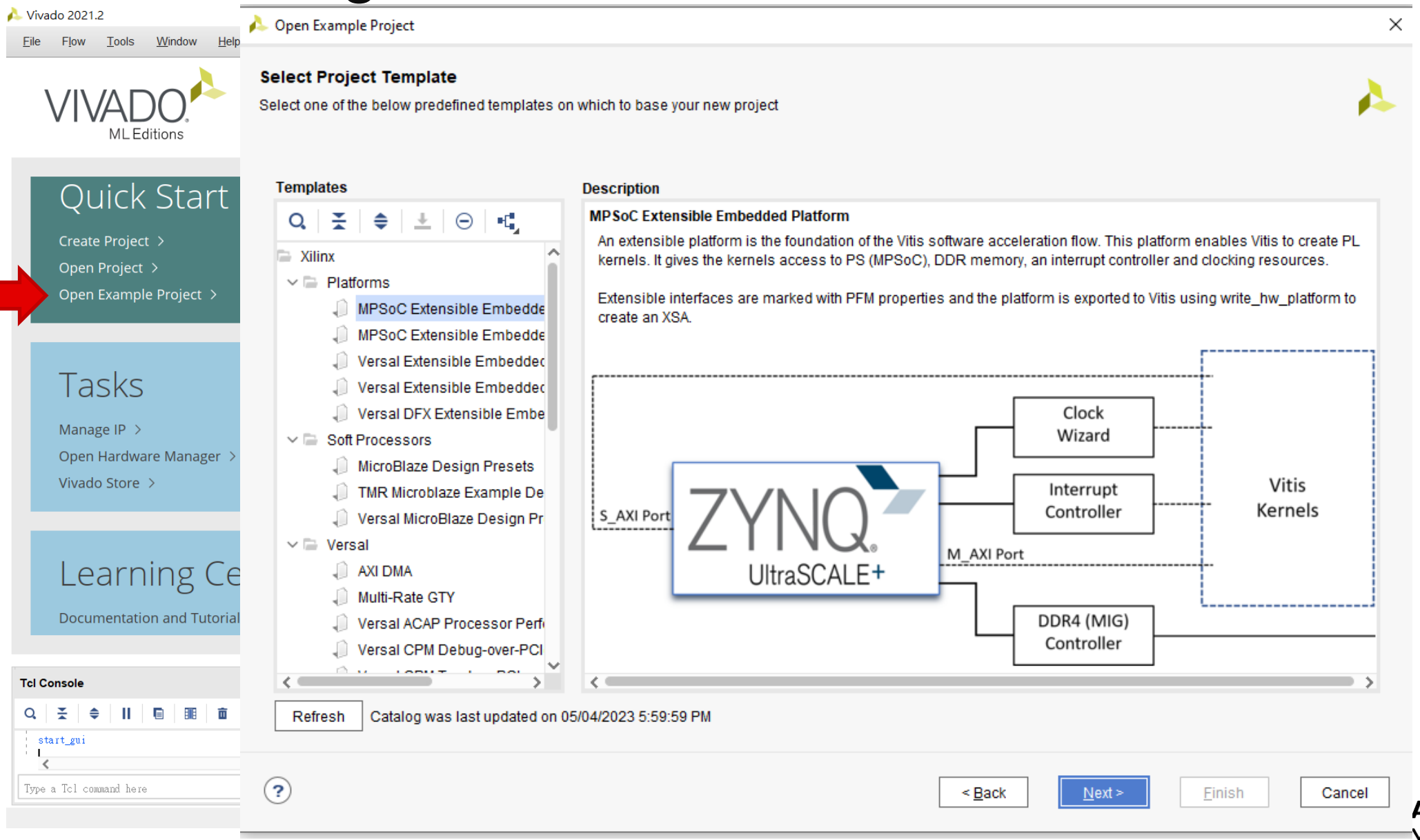
2022.2

Vivado Start Page



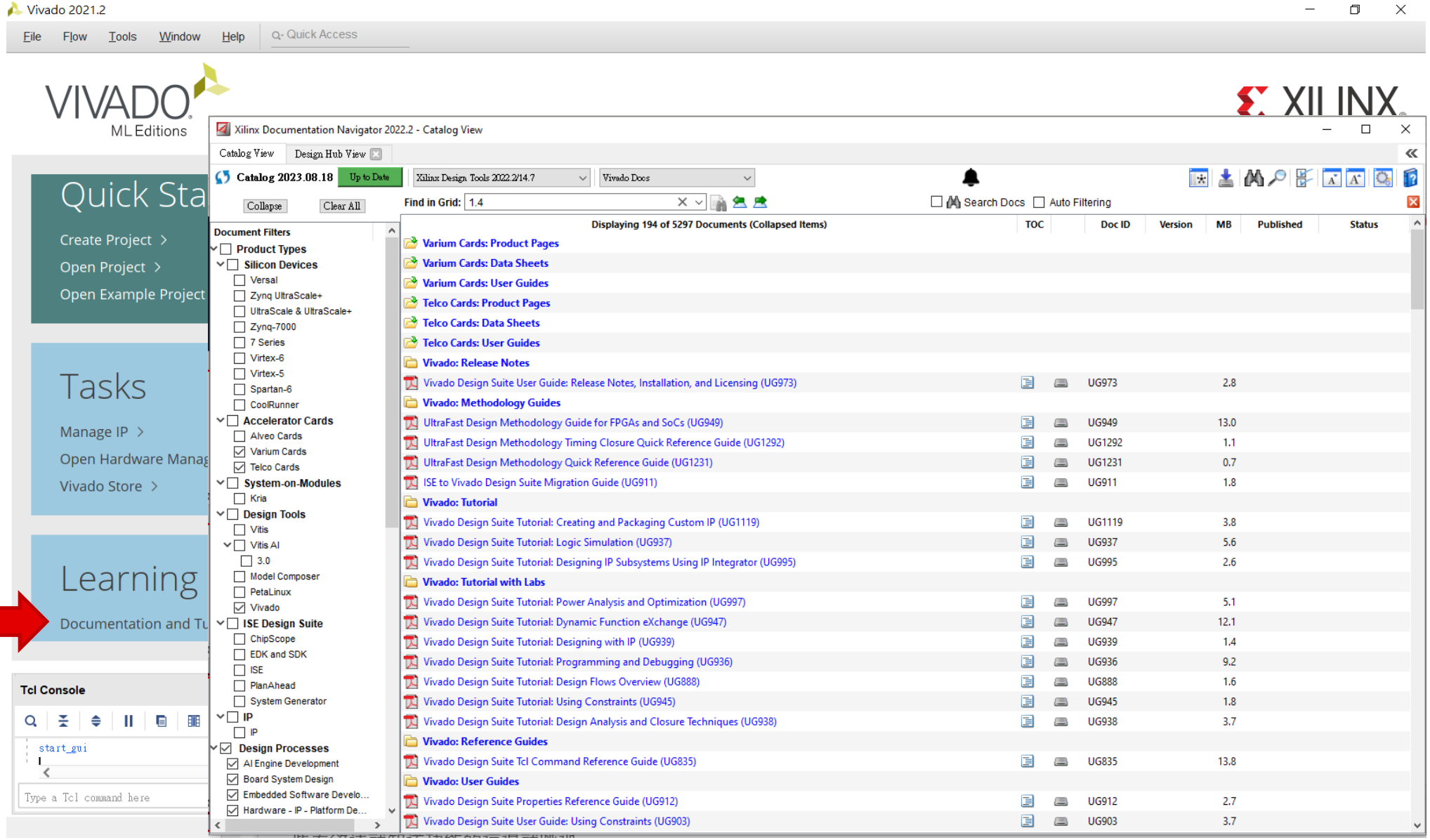
AMD
XILINX

Vivado Start Page



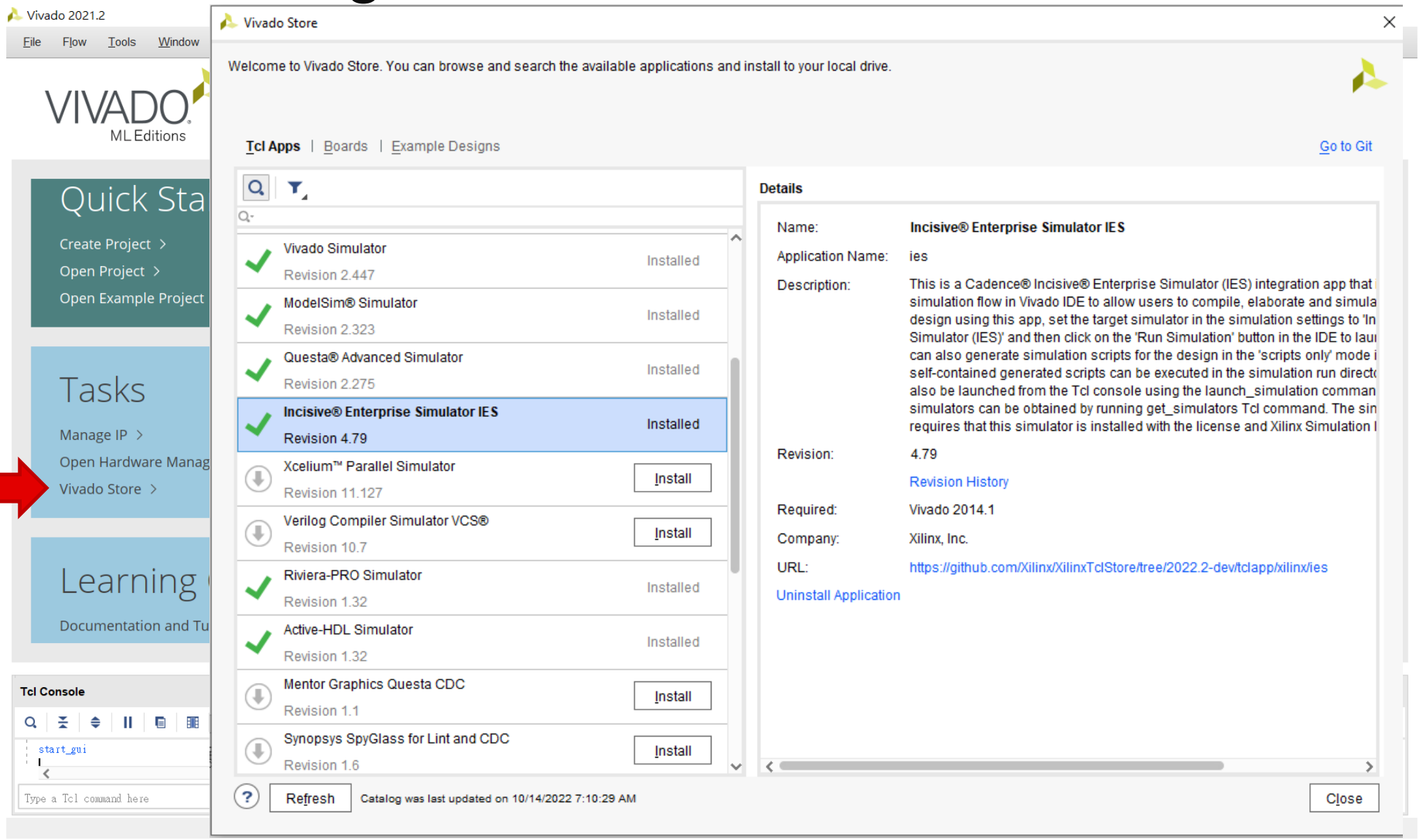
AMD
XILINX

Vivado Start Page



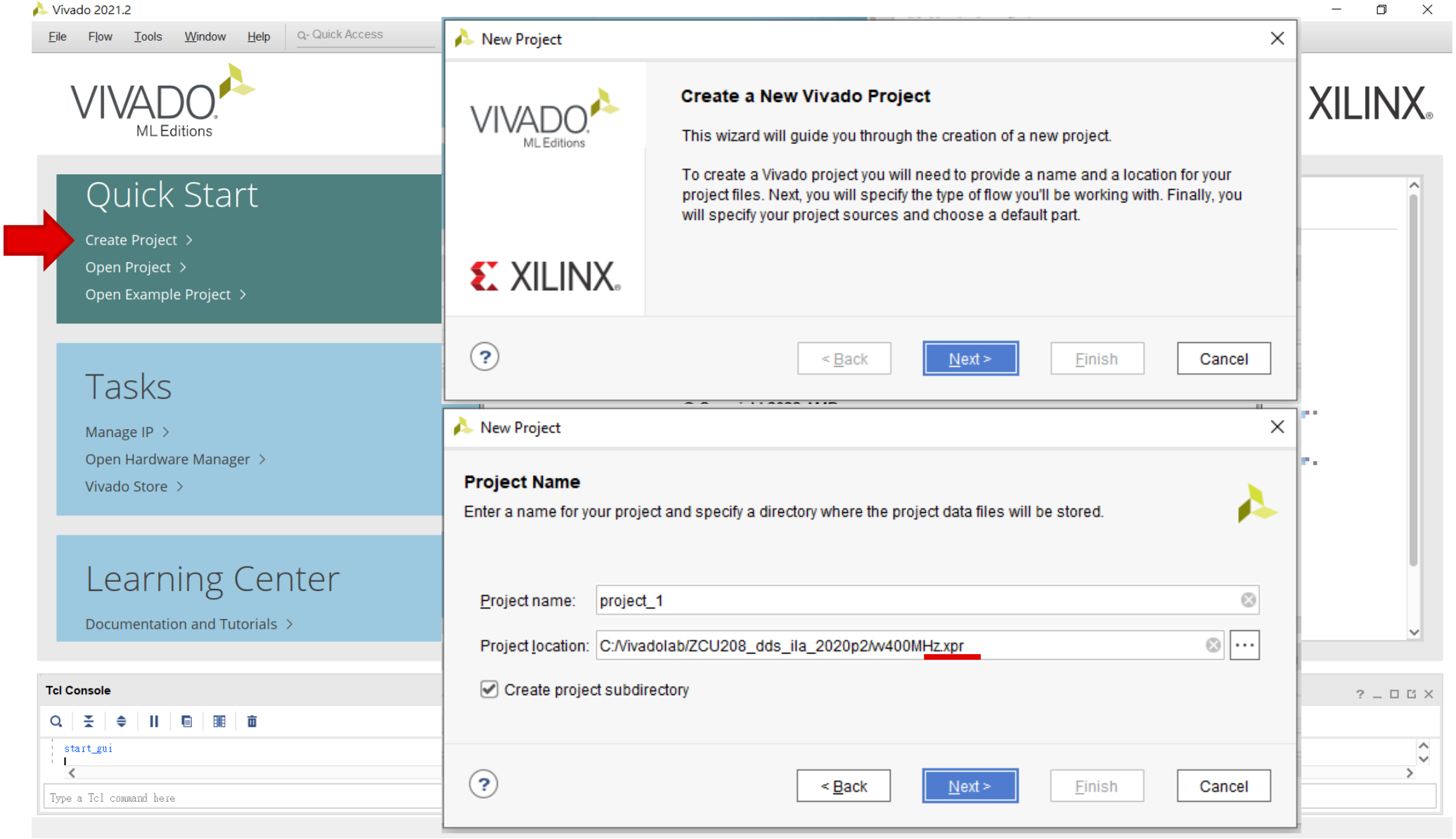
AMD XILINX

Vivado Start Page



AMD XILINX

Vivado Start Page



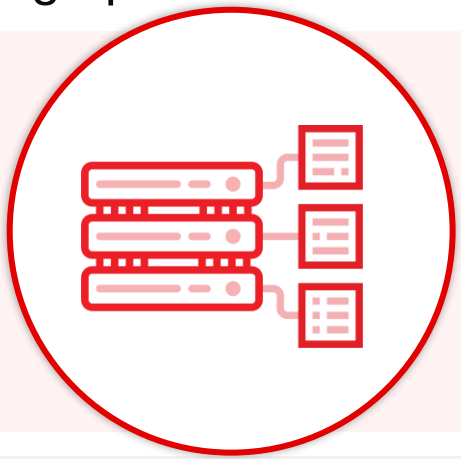
AMD
XILINX

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

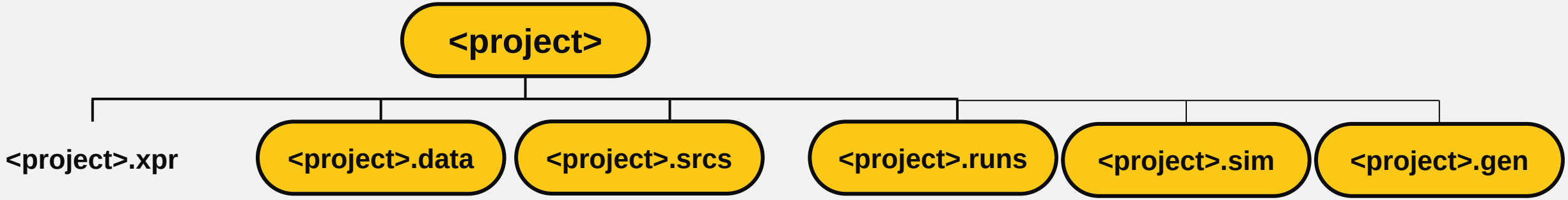


Project data to link all the elements together

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

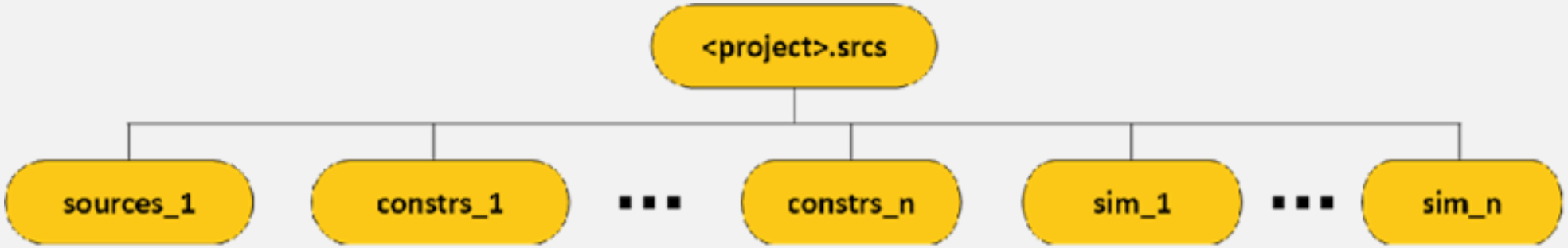


AMD XILINX

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



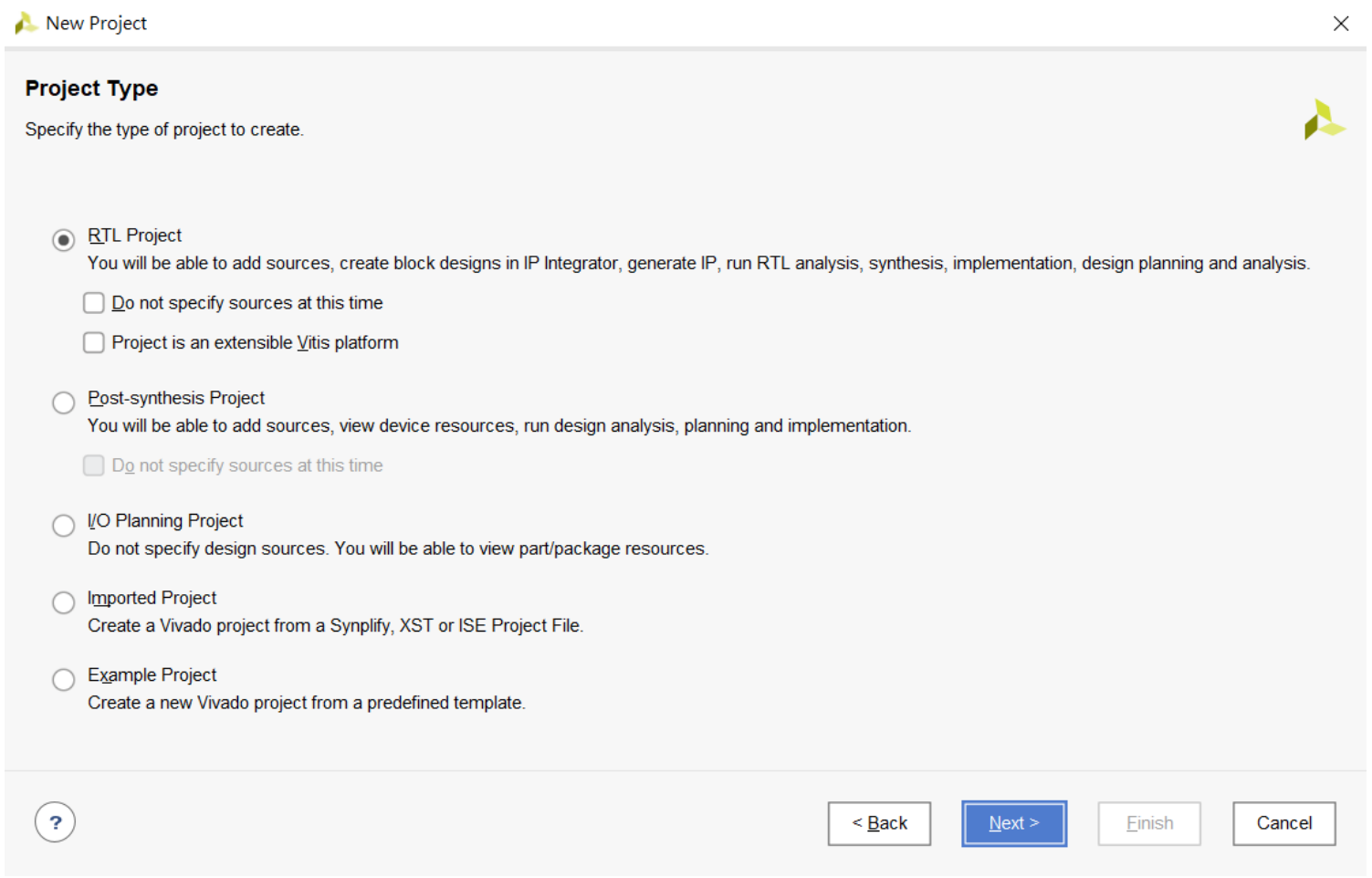
You can have multiple filesets of some types (simulation and constraint)

AMD XILINX

New Project Creation Wizard

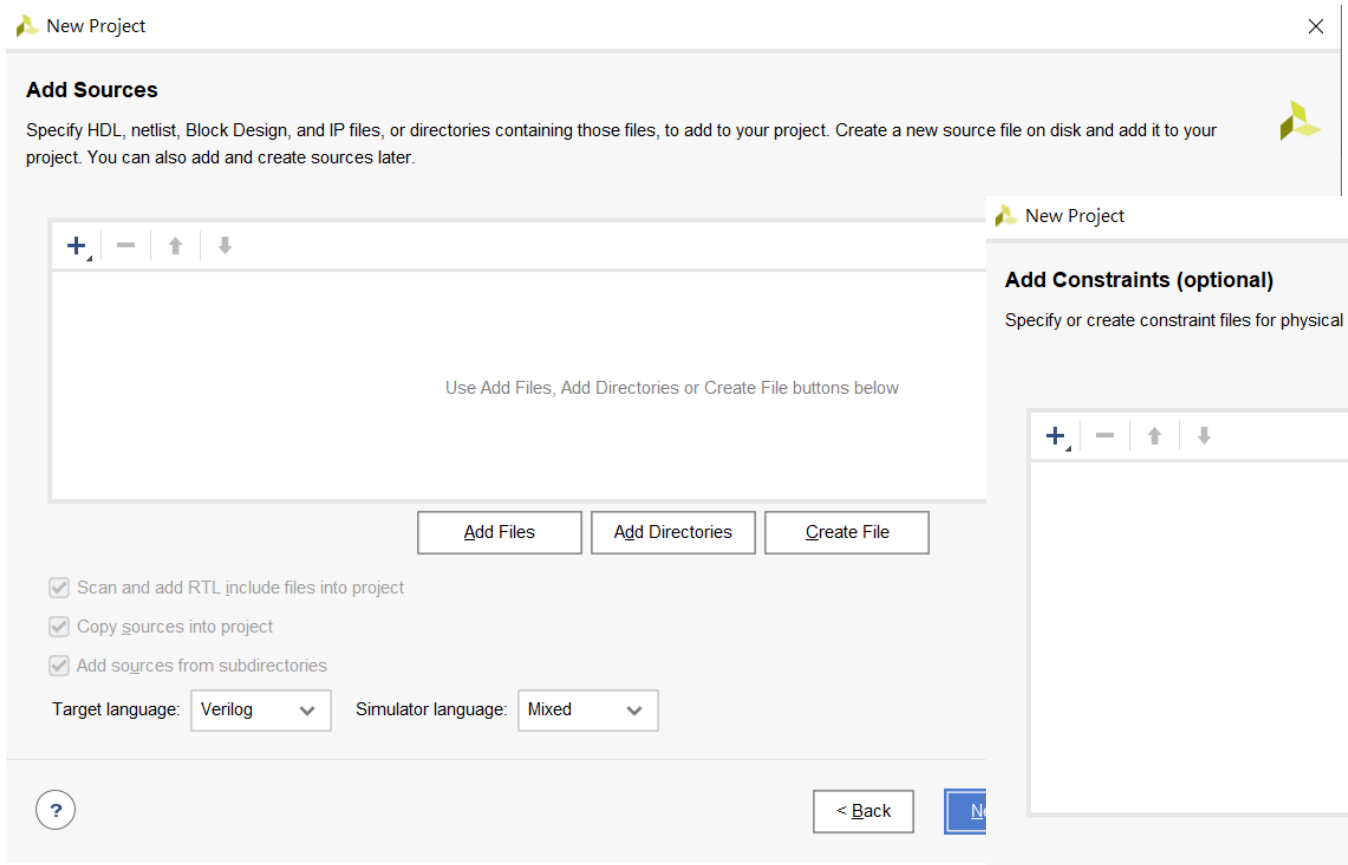
Creating a Project with HDL Sources

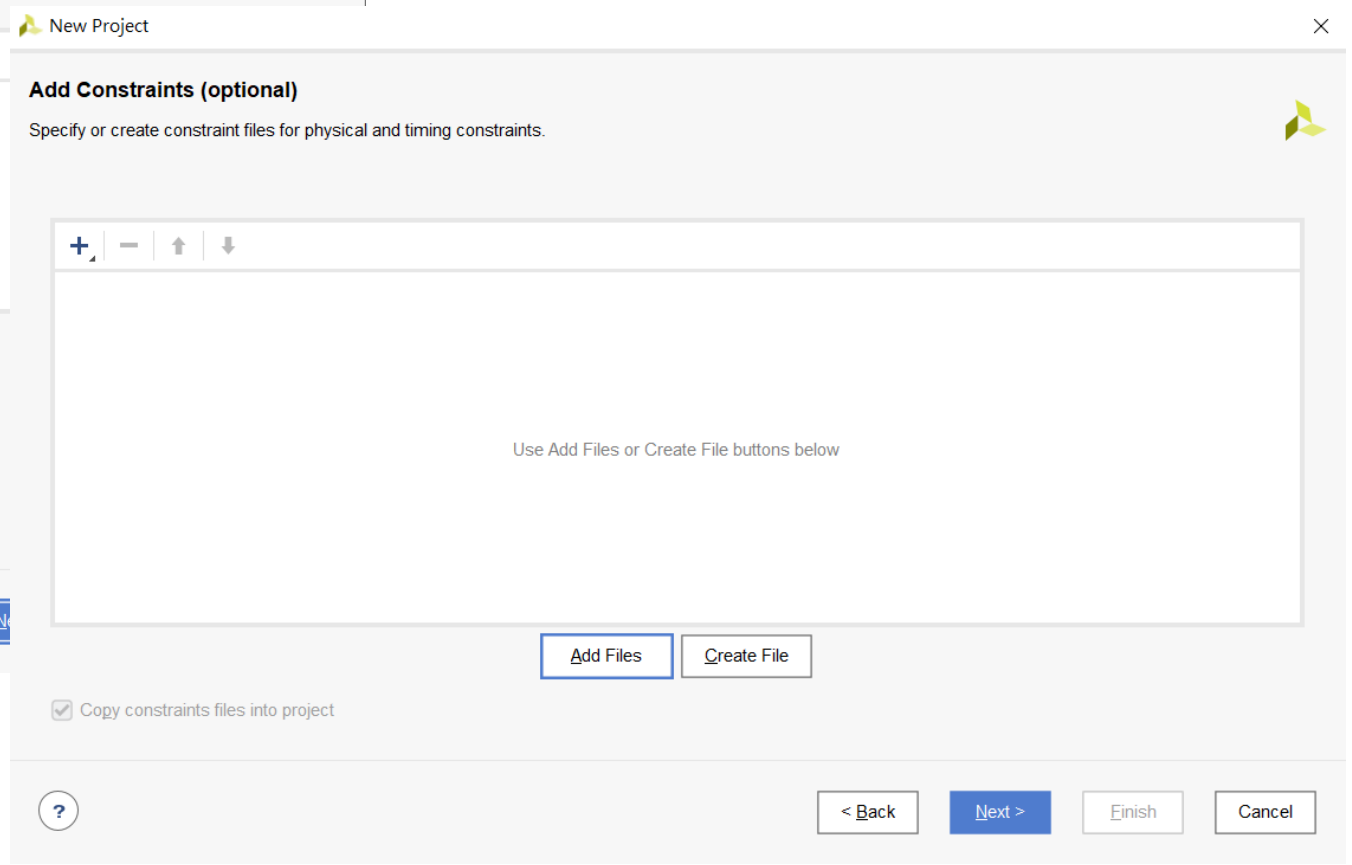
Creating a Project with a Synthesized Netlist



AMD XILINX

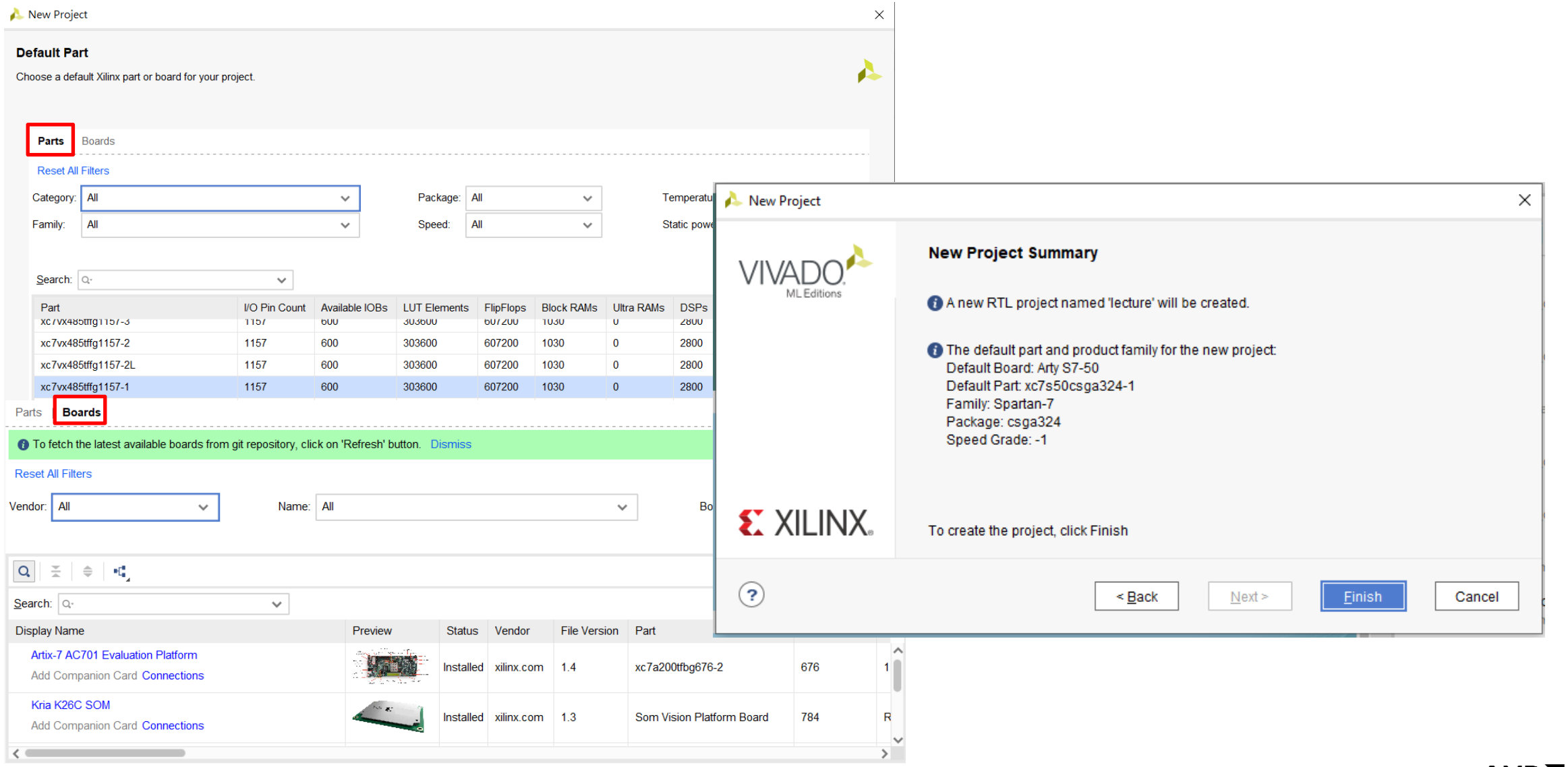
New Project Creation Wizard





AMD XILINX

New Project Creation Wizard



AMD
XILINX

Integrated Design Environment (IDE)

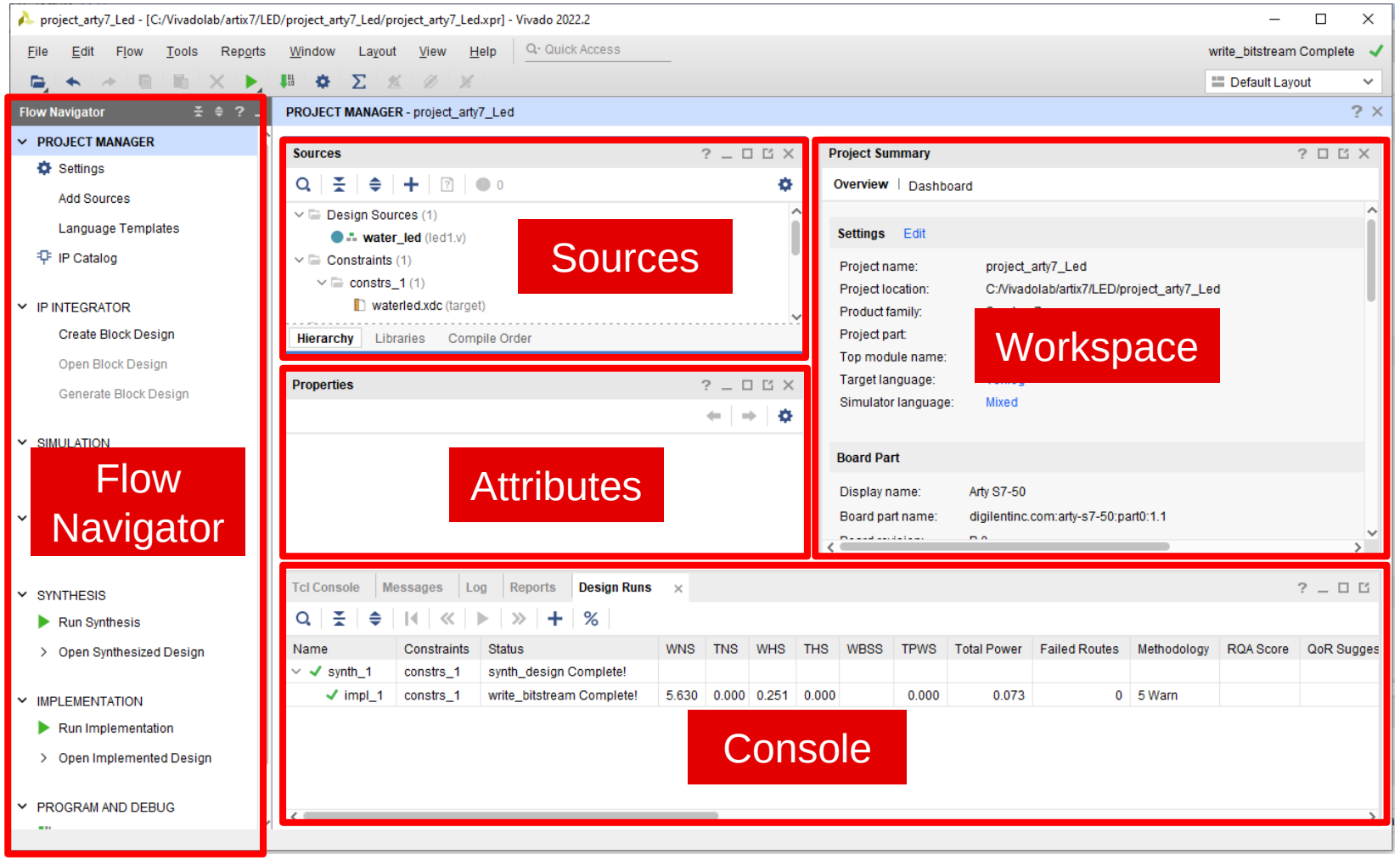
A Flow Navigator

B Sources

C Attributes

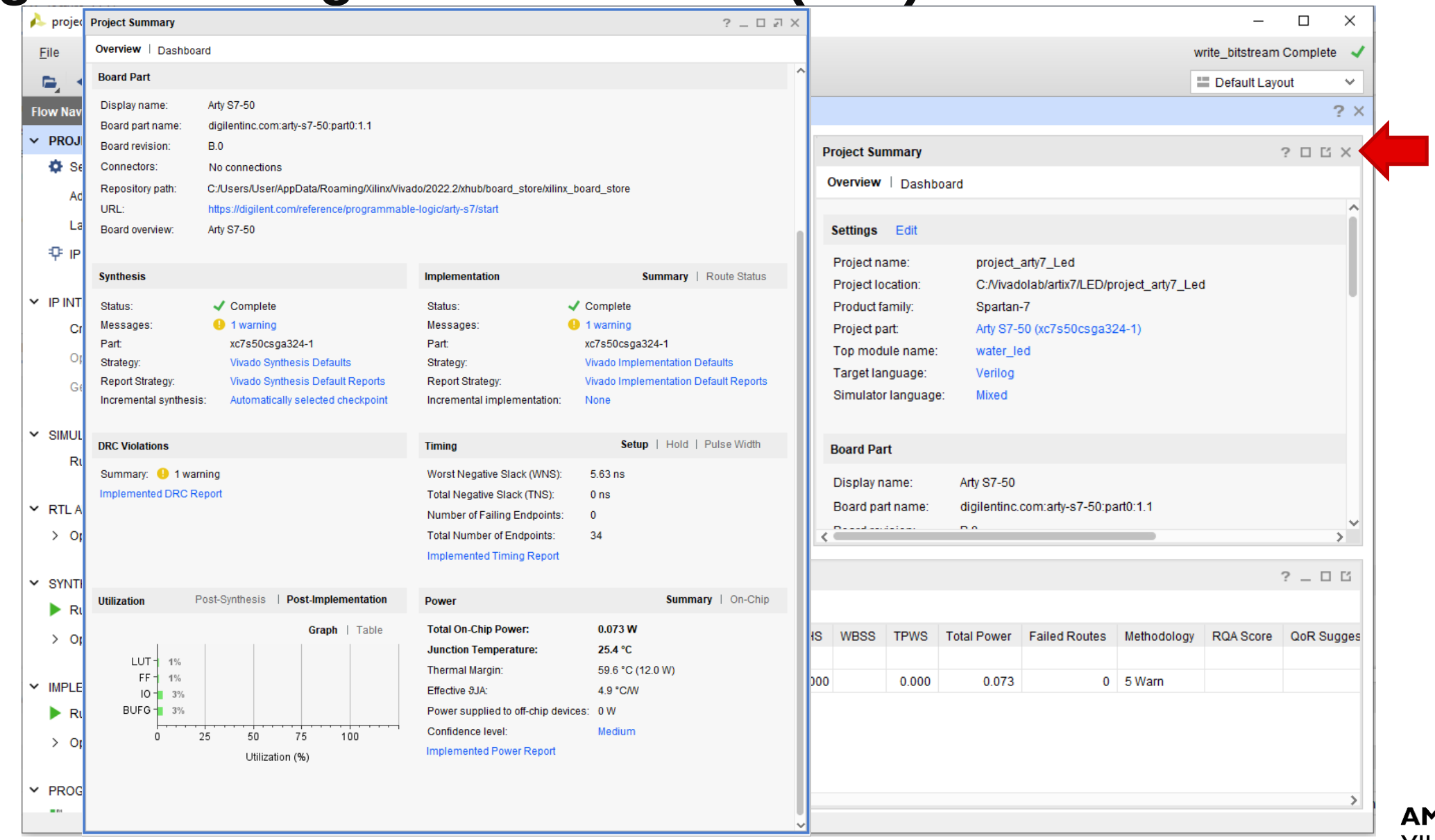
D Workspace

E Console



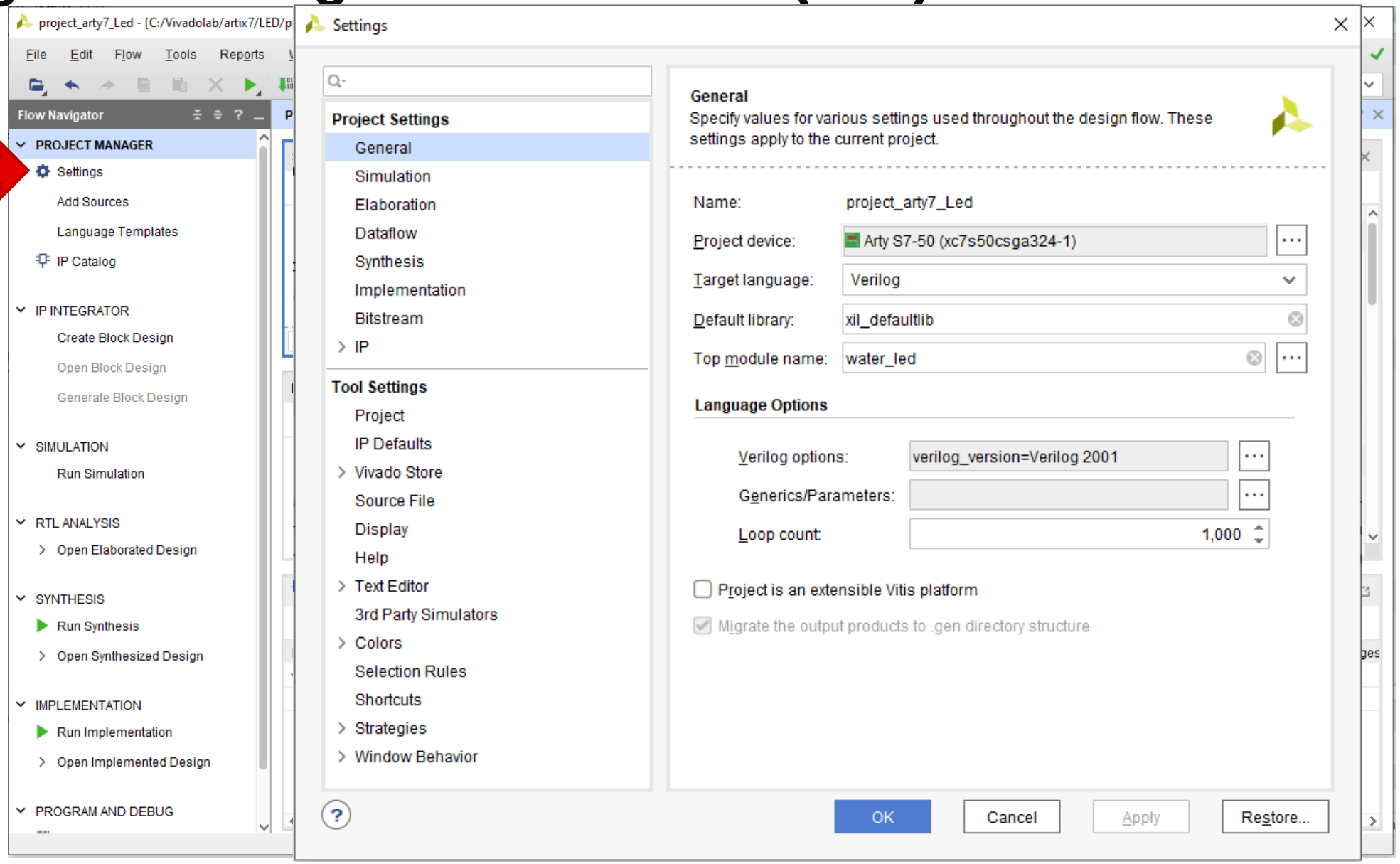
AMD XILINX

Integrated Design Environment (IDE)



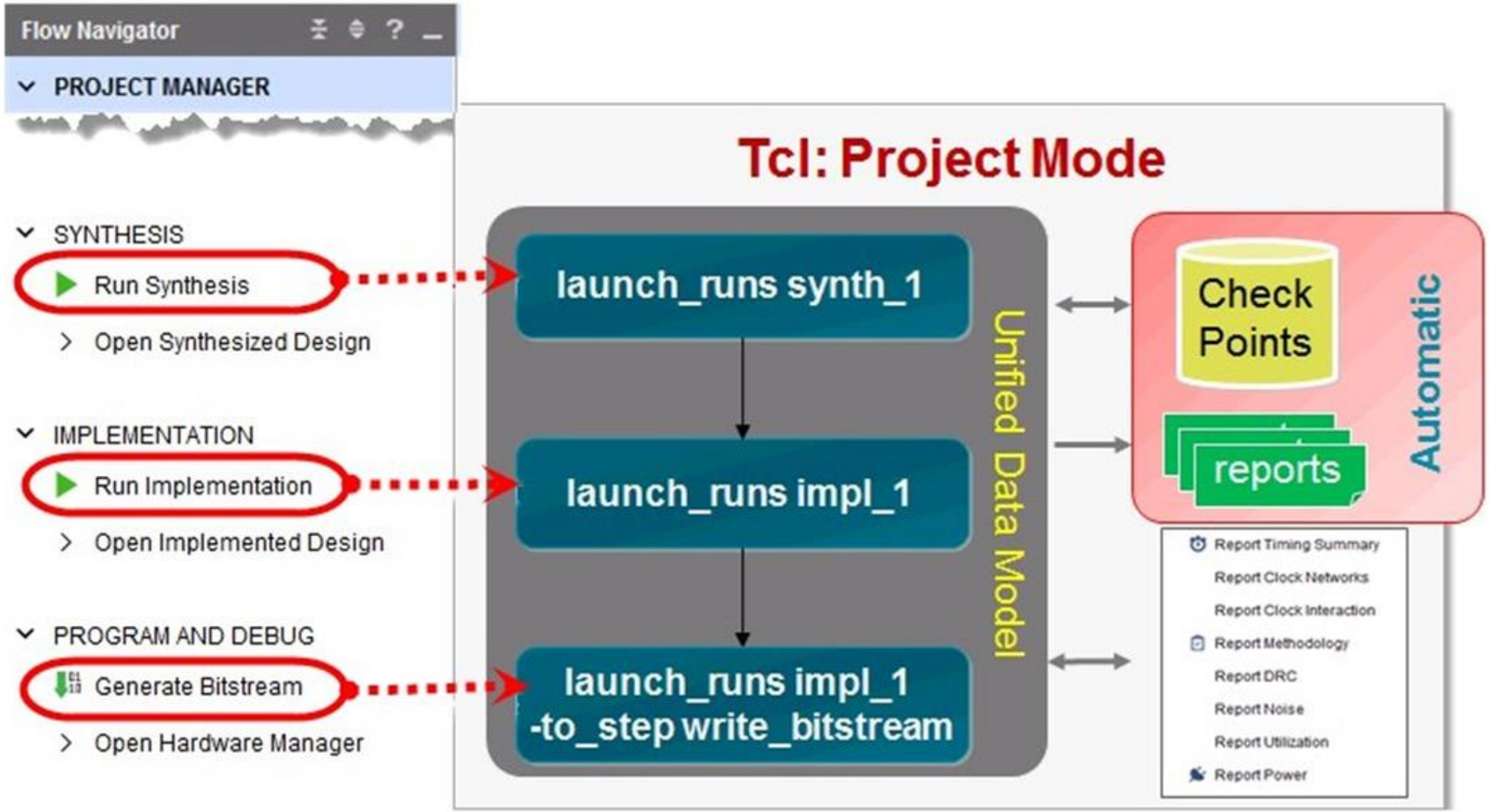
AMD XILINX

Integrated Design Environment (IDE)



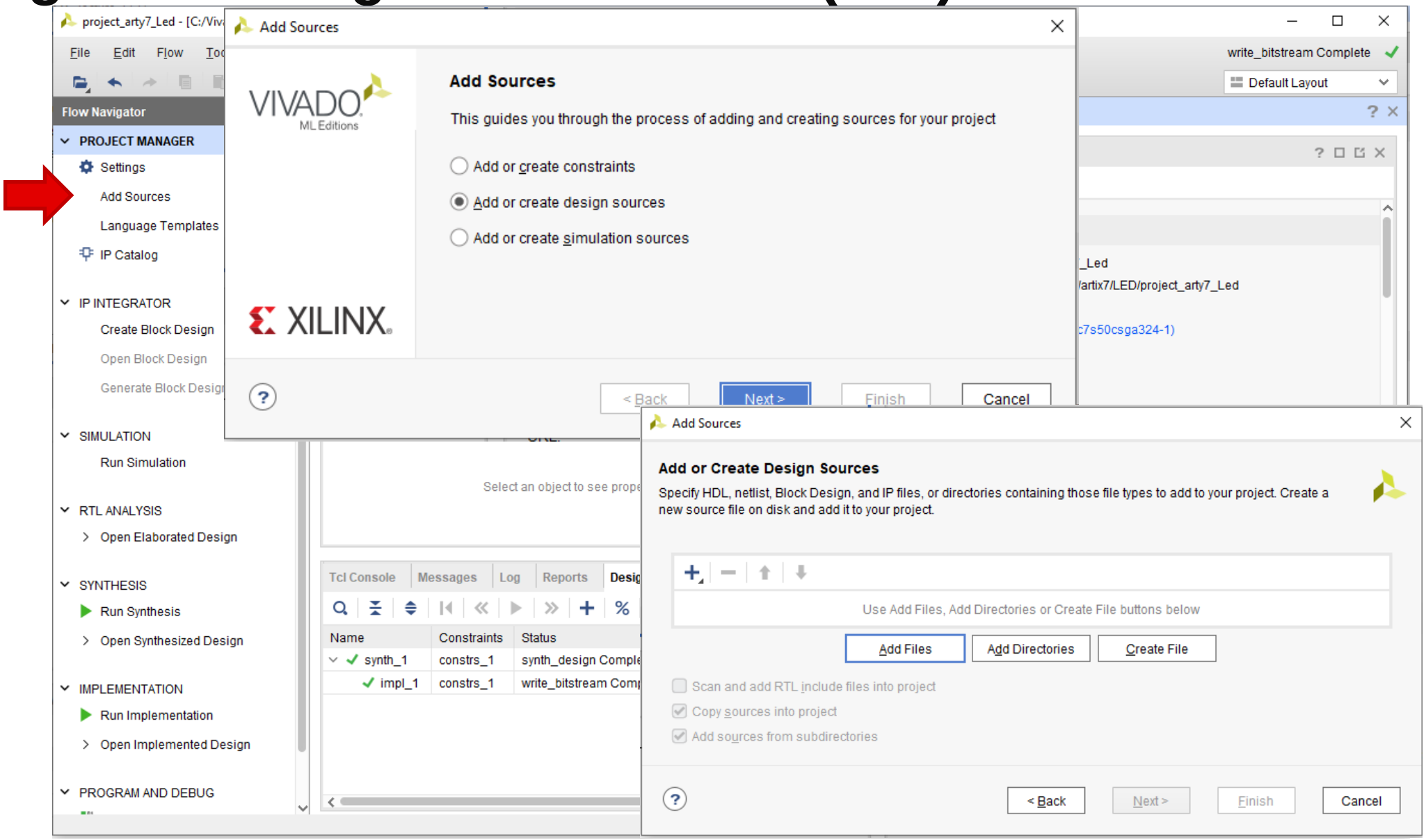
AMD XILINX

Project Mode: GUI → Tcl



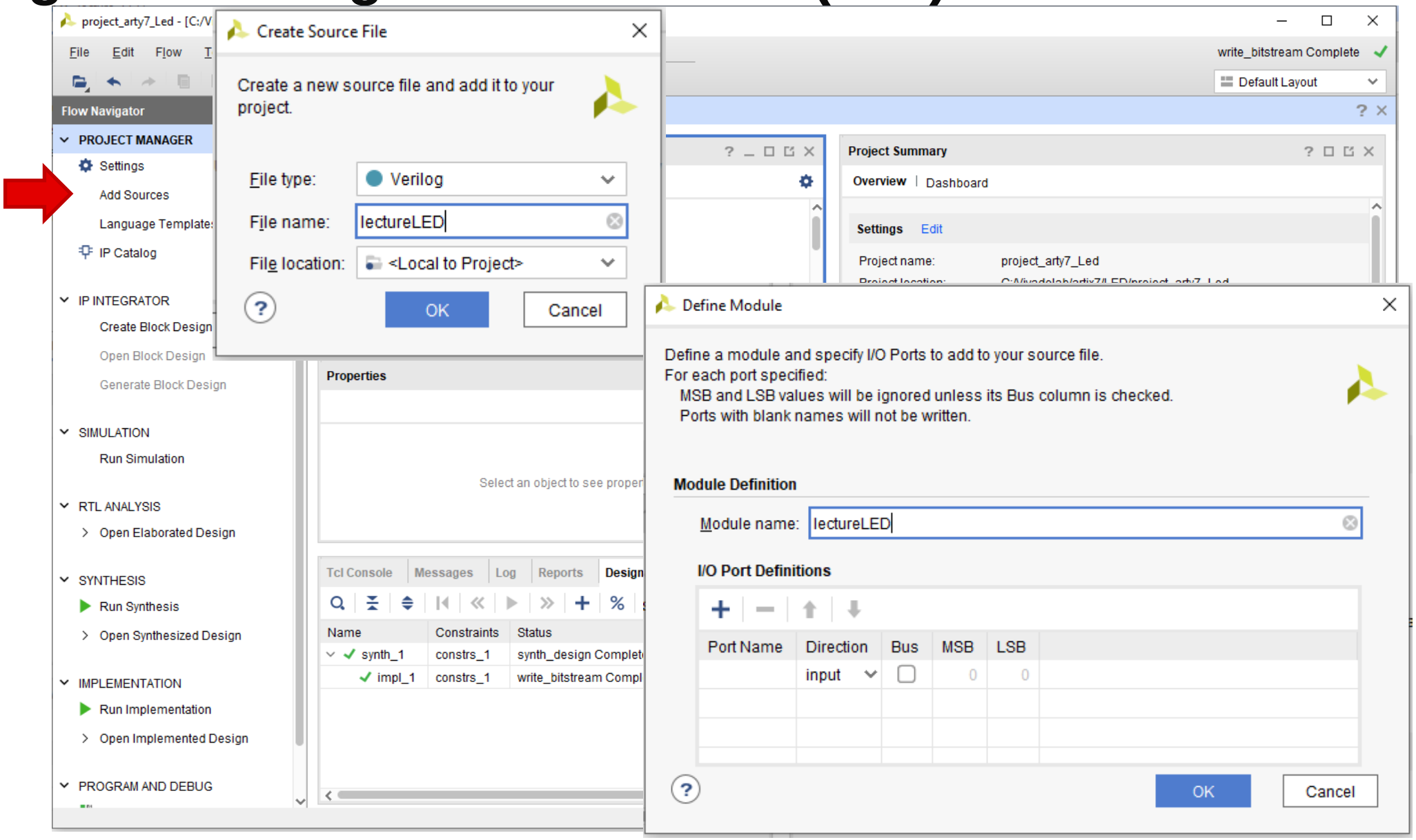
AMD
XILINX

Integrated Design Environment (IDE)



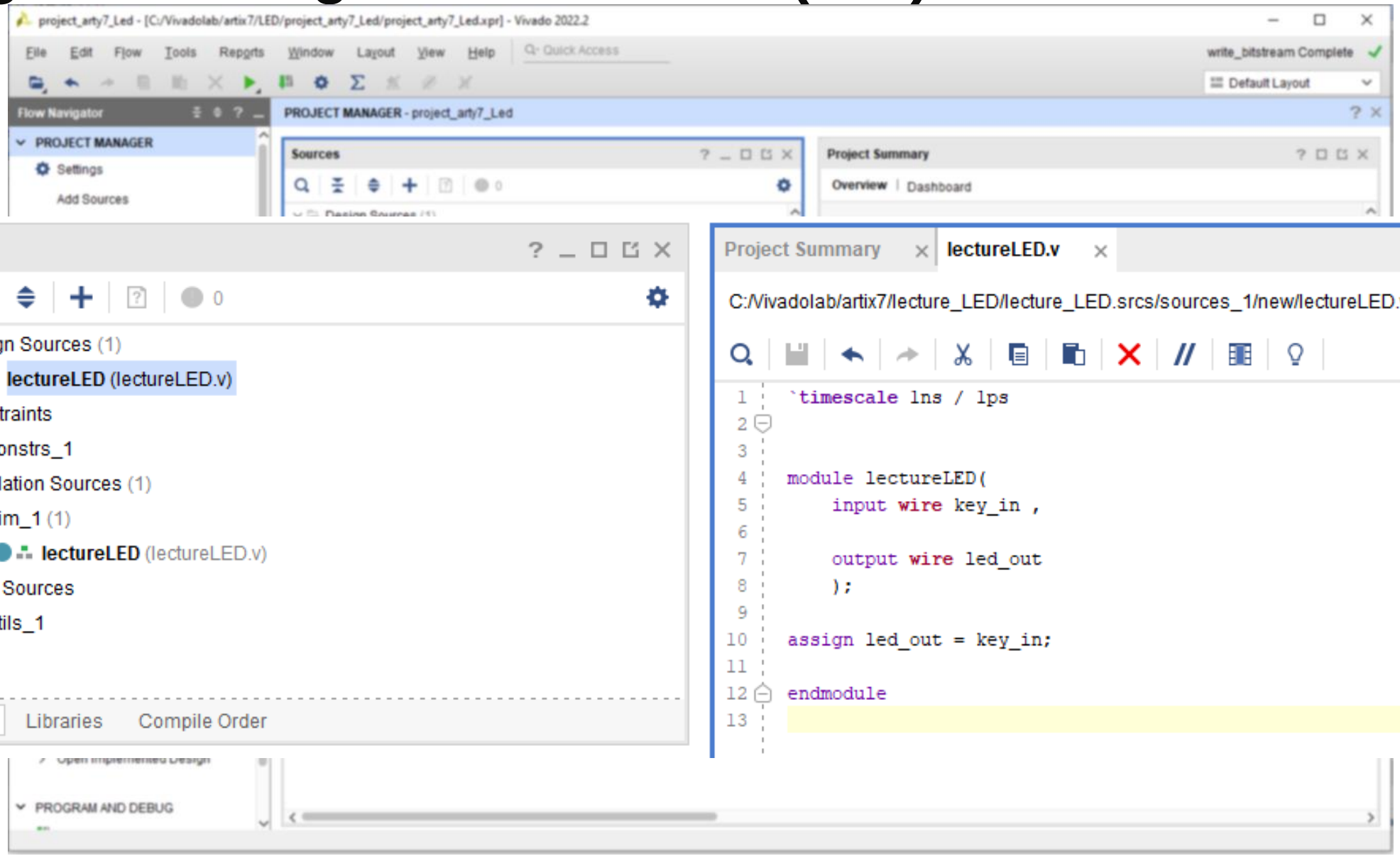
AMD XILINX

Integrated Design Environment (IDE)



AMD XILINX

Integrated Design Environment (IDE)



AMD XILINX

AMD
XILINX

Behavioral Simulation

2022.2

Behavioral Simulation

Design efficiency determined before system implementation

Syntax and functionality of a design verified by injecting stimuli and observing outputs

Performed on a source file or pre-synthesis design

Design modeled with high abstraction

Carried out early in the design cycle

Fastest—no timing information needed

AMD XILINX

What is a Testbench?

Virtual environment for design verification

Components of a Testbench

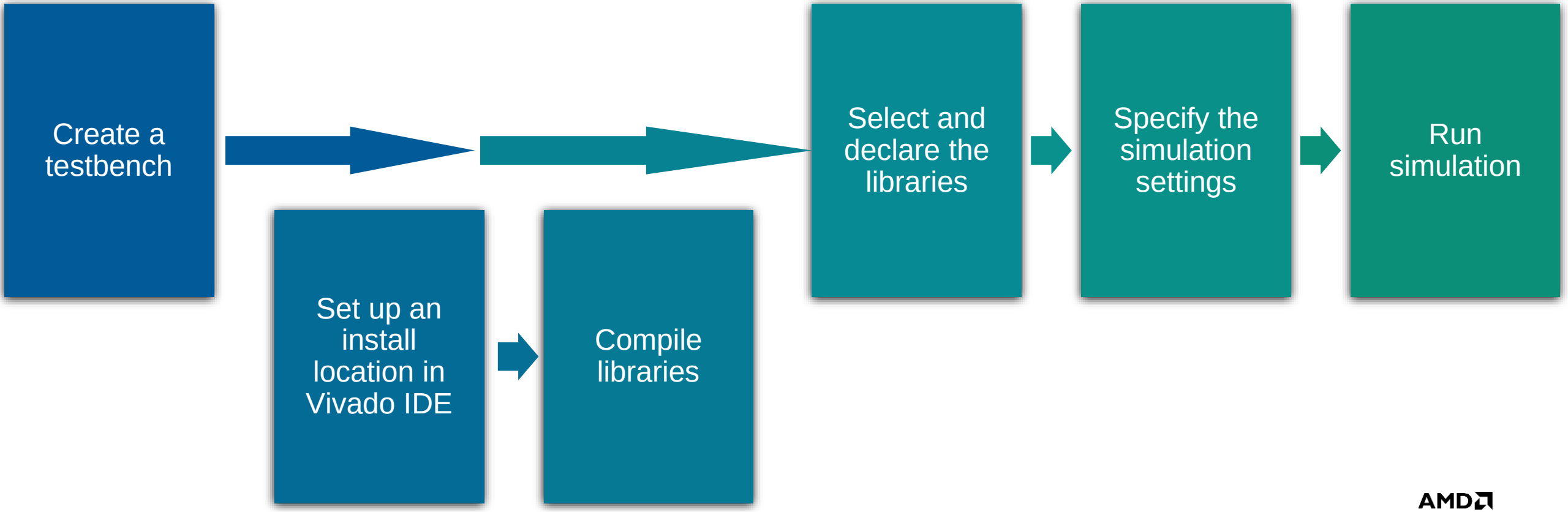
- HDL formatted
- Internal signals
- Unit under test instantiation
- Input stimulus including clock signals
- Output monitoring

Uses of a Testbench

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

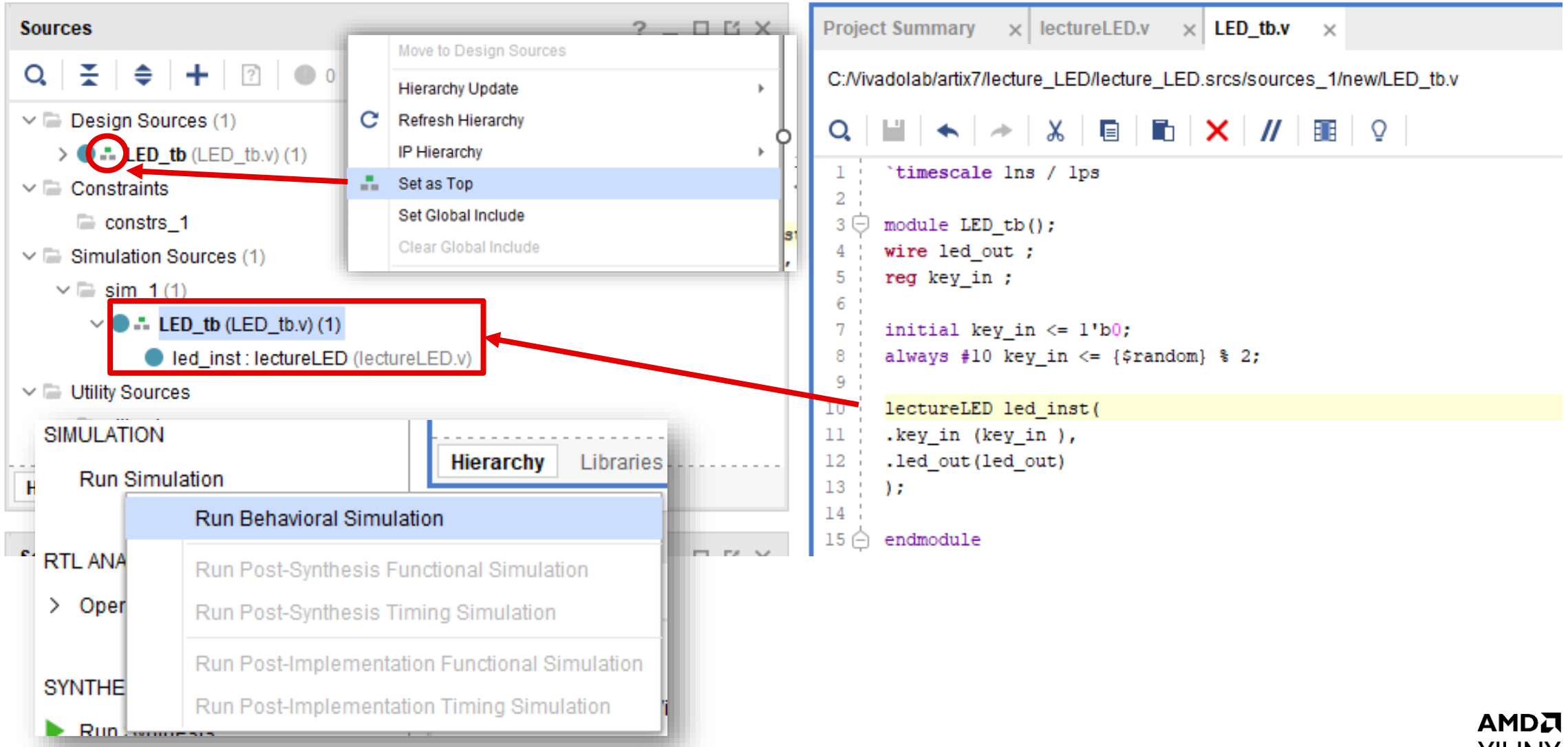
Preparing for Simulation

Set up the following before performing the simulation:



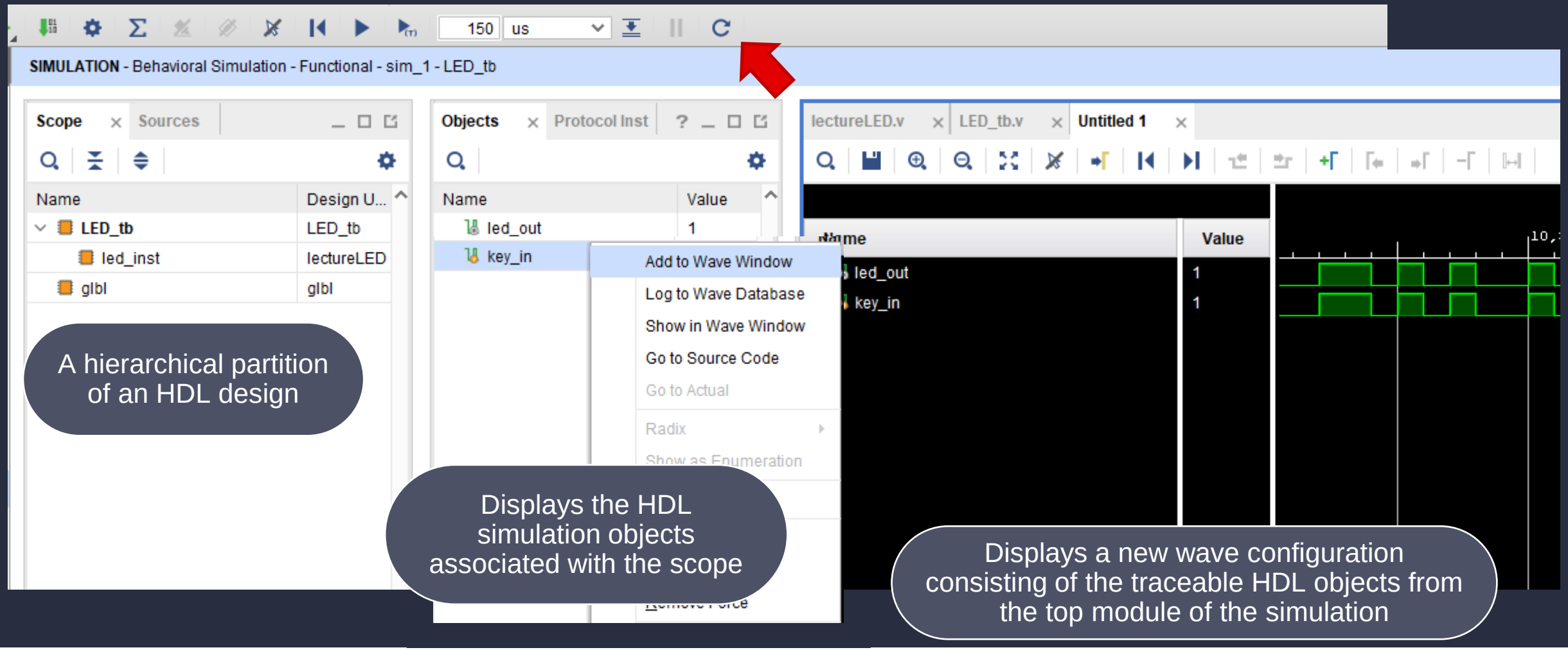
AMD XILINX

Behavioral Simulation



Simulation Waveform

Select **Simulation** > **Behavioral Simulation**



AMD XILINX

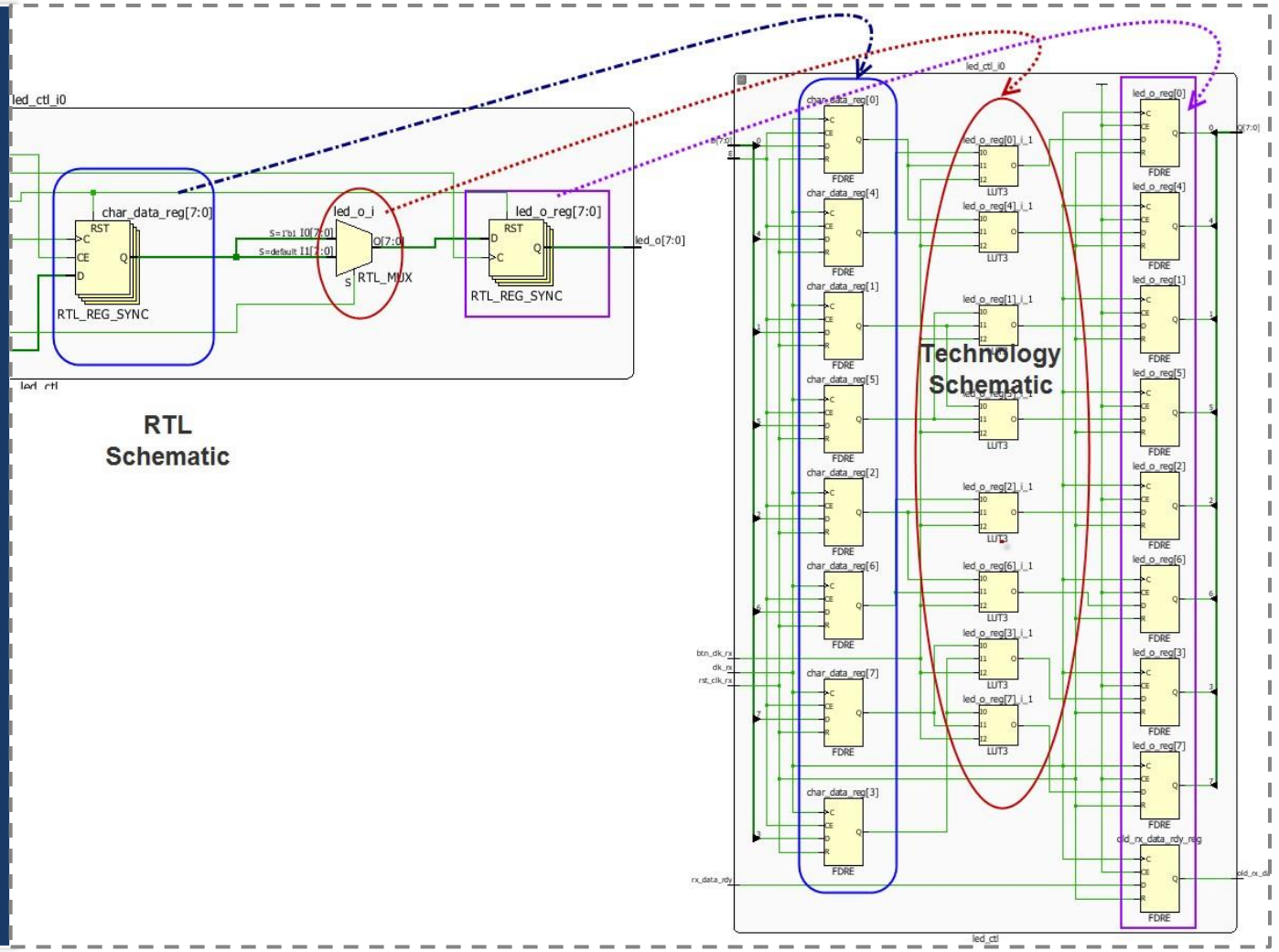
AMD
XILINX

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



AMD
XILINX

Synthesis Design

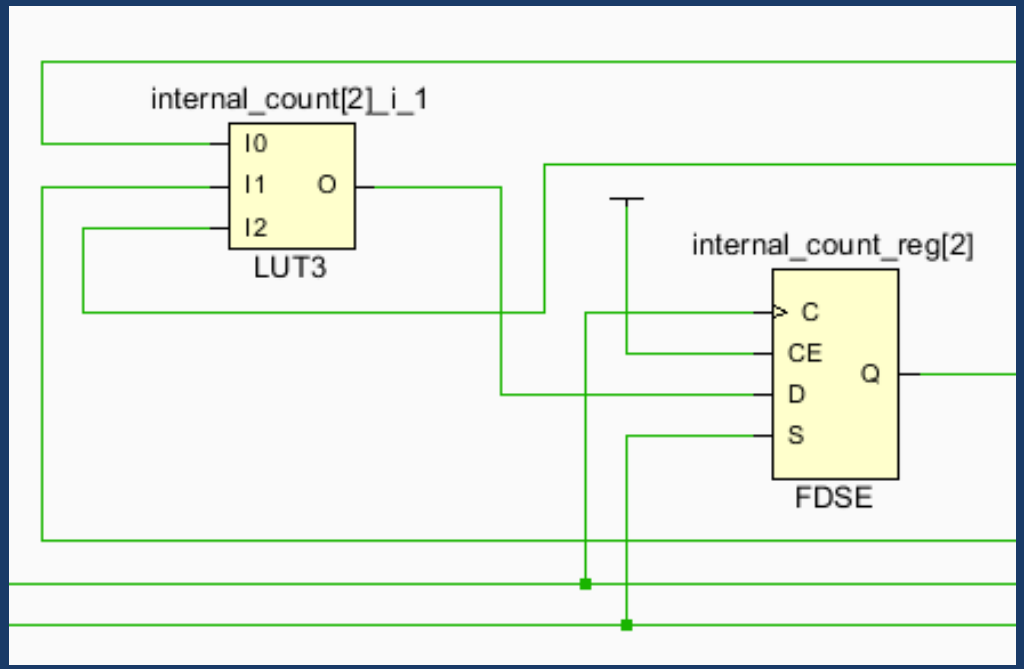
Representation of the design after synthesis

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

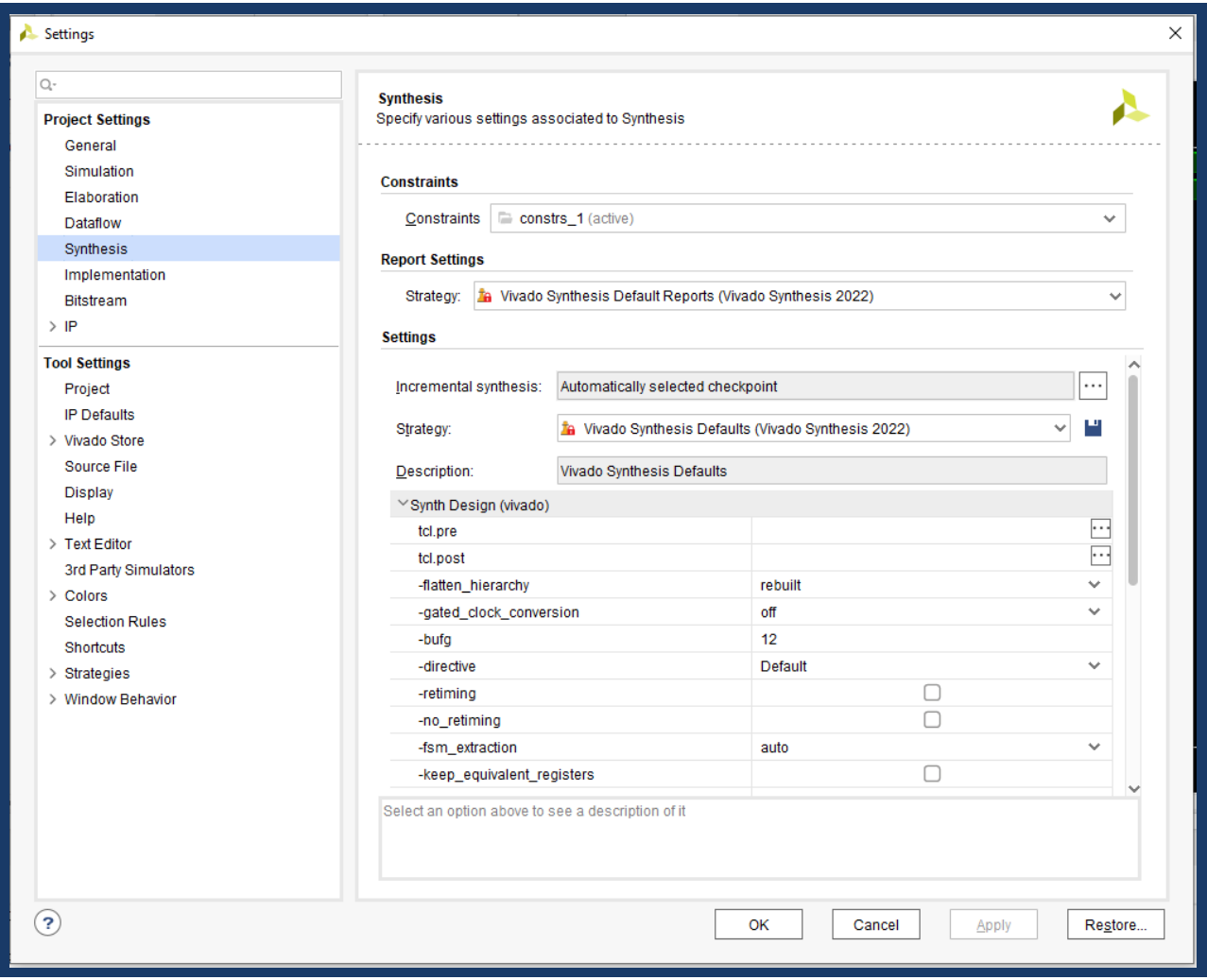
Design timing and power estimates are possible



AMD XILINX

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

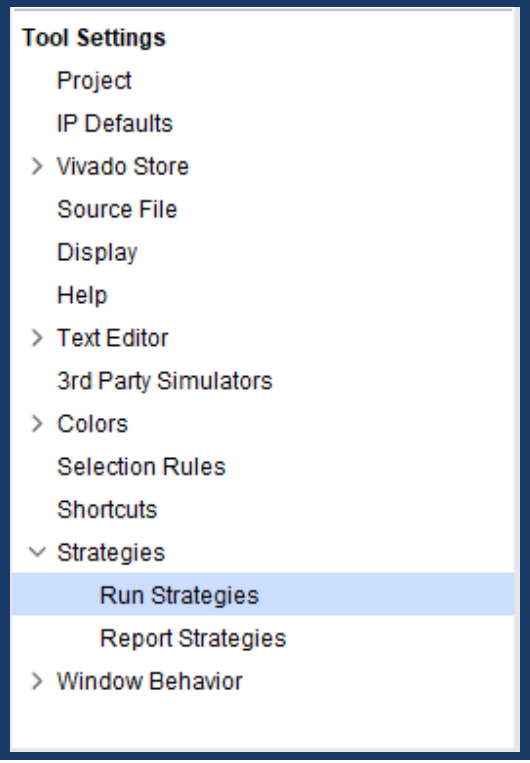
Attribute	Description
translate_off/_on	Tells the tool to ignore blocks of code
full_case	Tells that all possible case values are specified
parallel_case	Case statement should be built as a parallel structure
keep	Tells tool to keep the signal the attribute is placed on
keep_hierarchy	Used to prevent 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

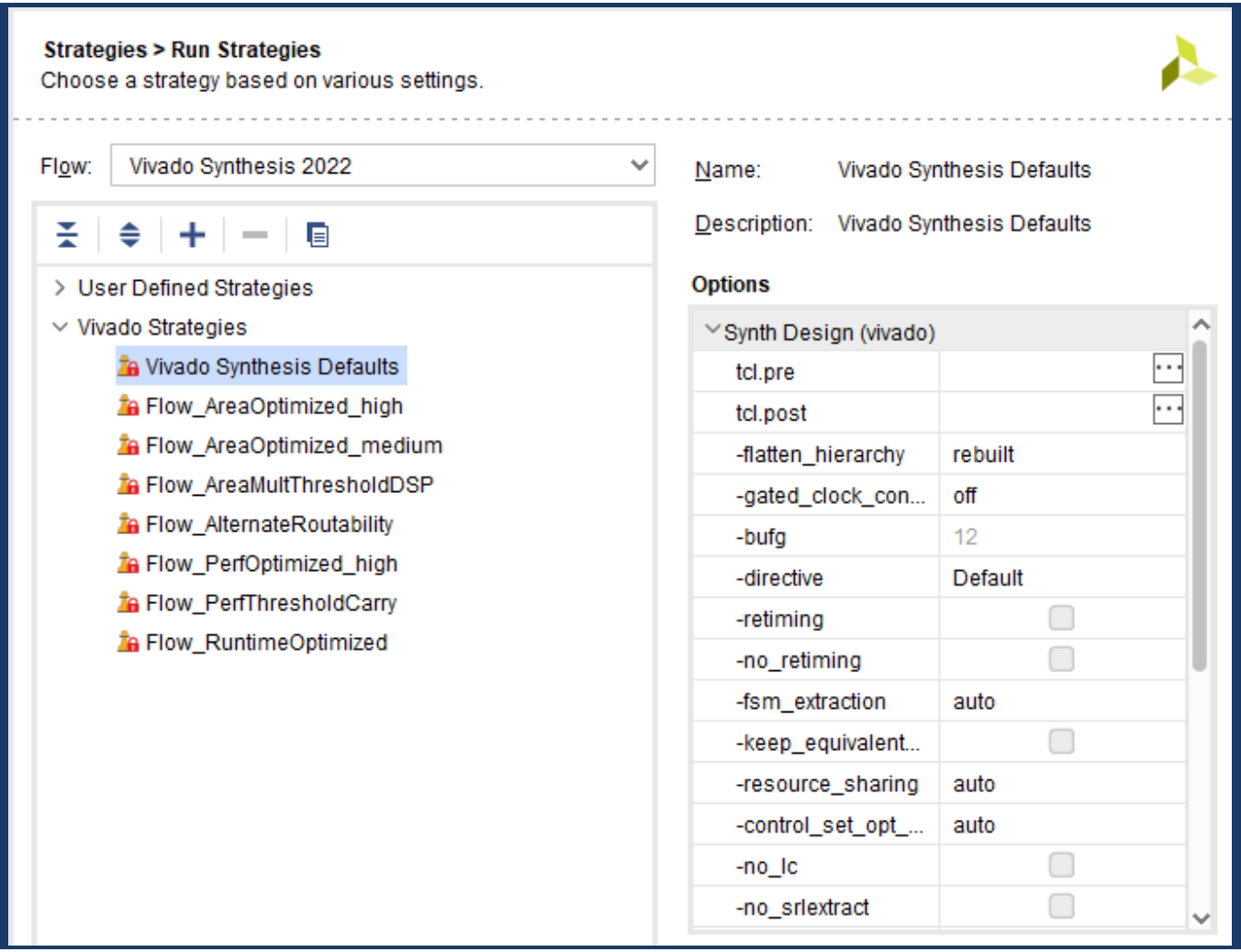
Synthesis RTL Attributes Supported

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





Block-Level Synthesis

Design involves hierarchies, either user created or tool generated

Allows different synthesis strategies for modules or instances

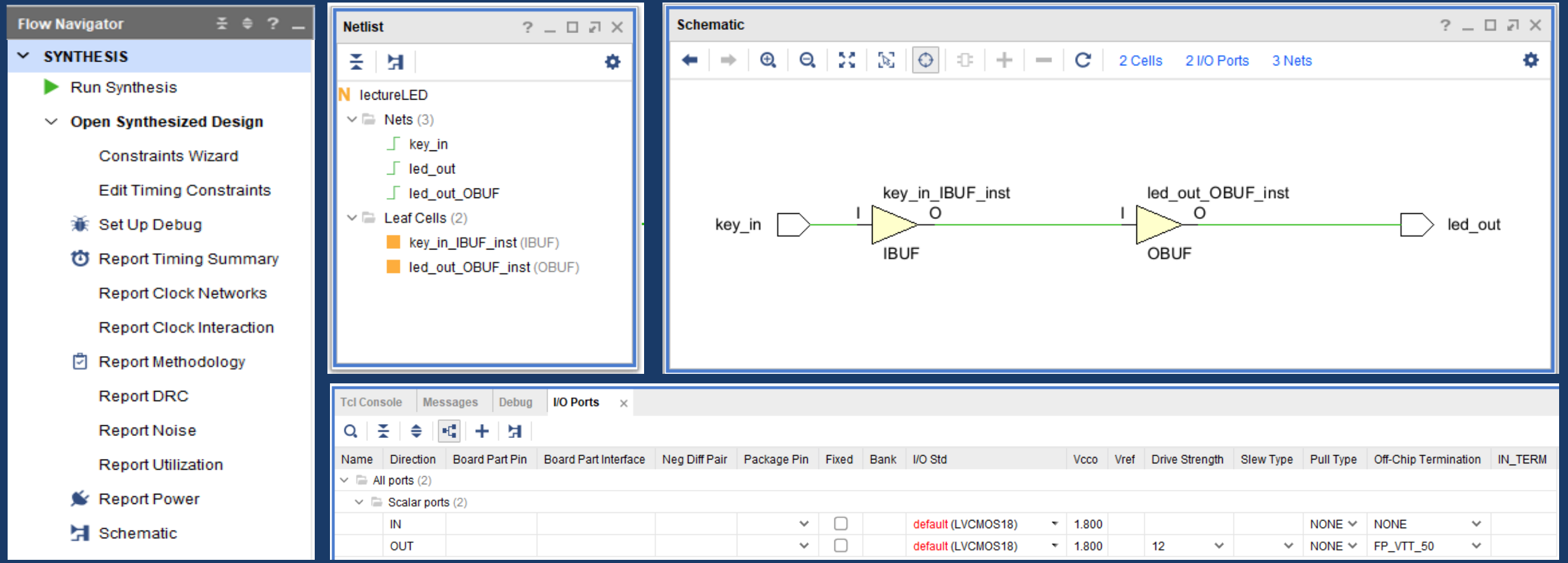
Allows optimization based on area, performance, and routing in addition to logic

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



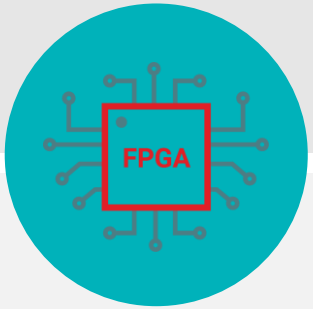
AMD XILINX

AMD
XILINX

I/O Pin Planning

2022.2

I/O Planning

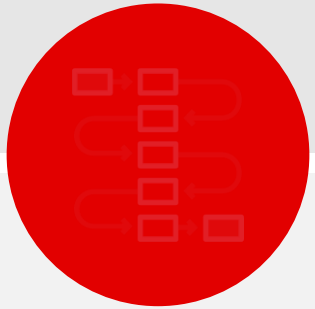


An FPGA design does not exist in a “ready-to-use” environment

- 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

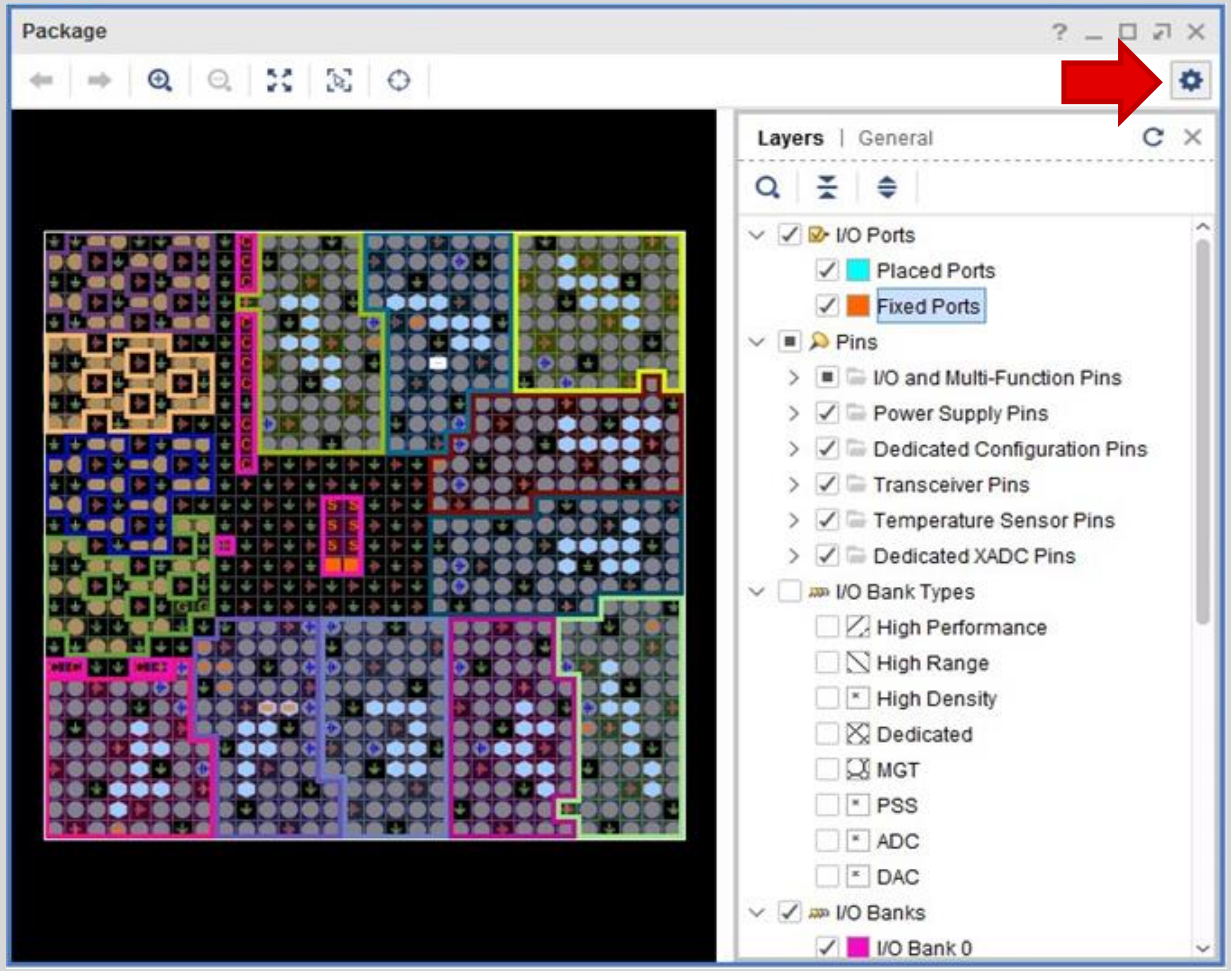
Package View

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

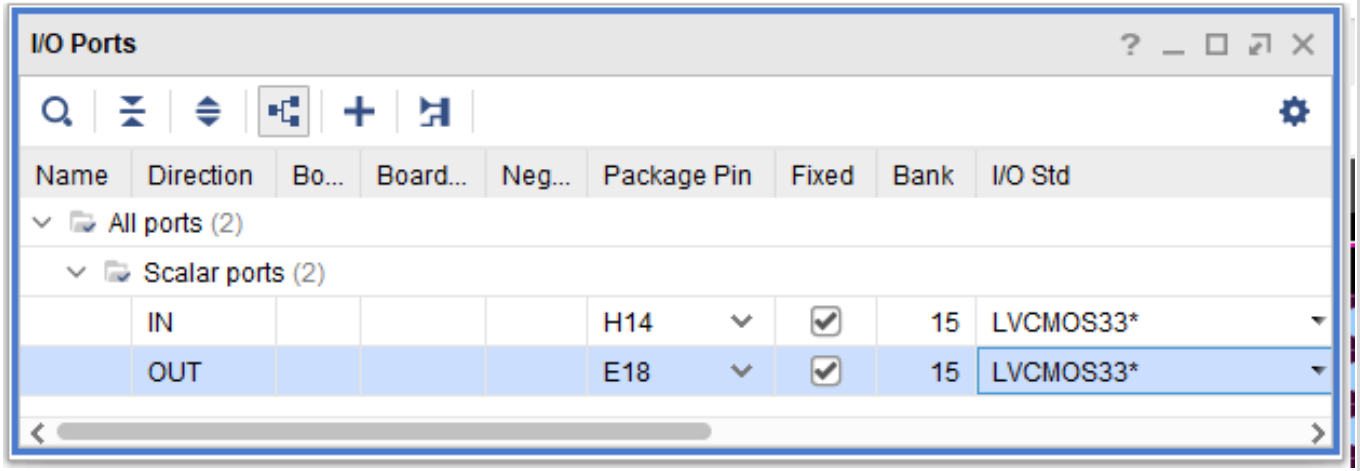
Clock-capable pins ( ), VCC ( ), GND ( ), no connection ( )



AMD XILINX

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



AMD XILINX

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

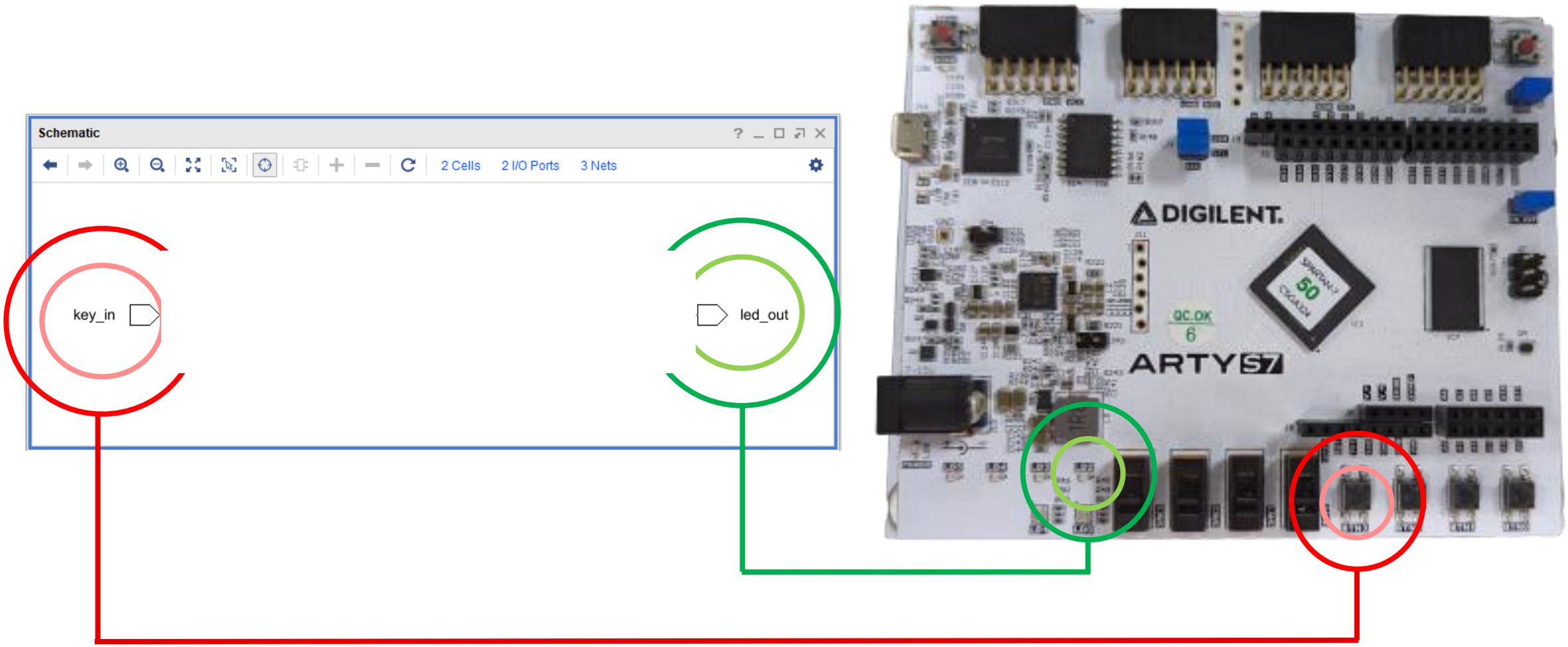
Ensure pin assignment works across multiple parts with a common package

AMD
XILINX

Vivado Implementation

2022.2

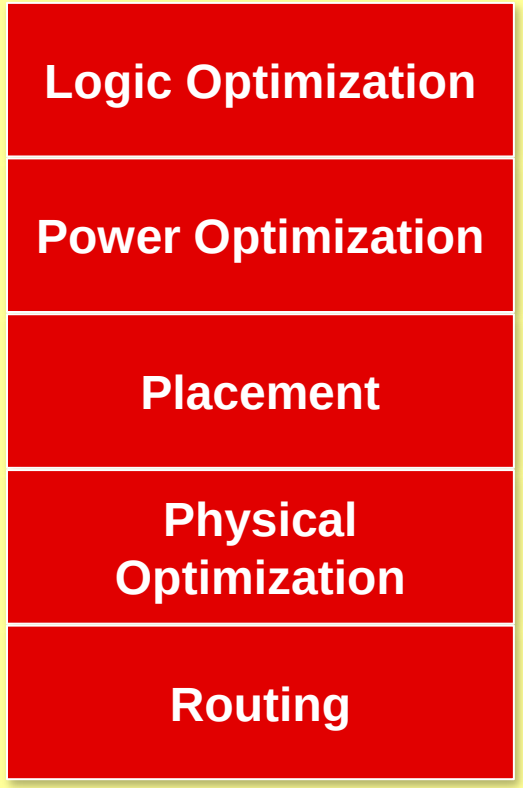
Vivado Implementation



AMD
XILINX

Vivado Implementation

Consists of four mandatory and one optional sub-processes

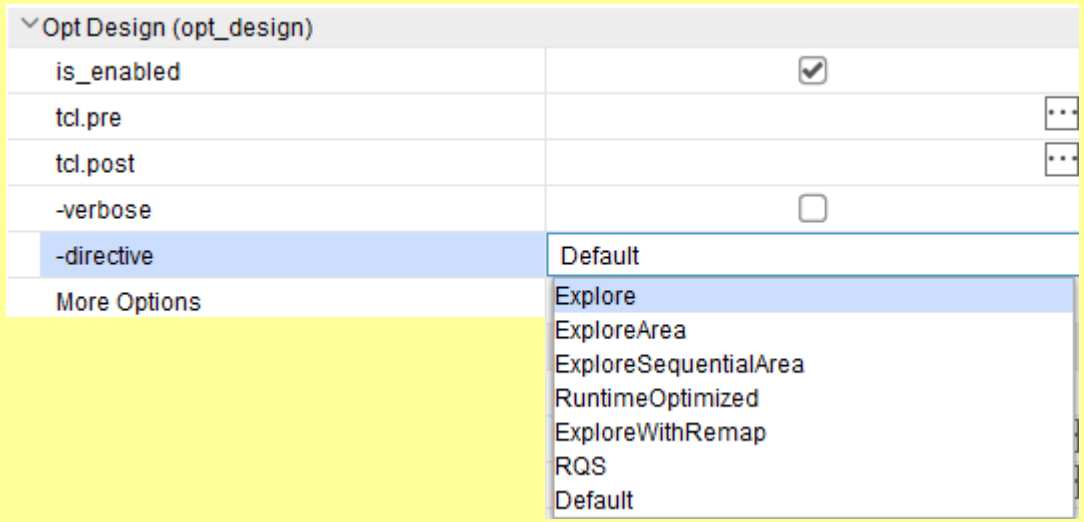


Logic Optimization

- **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)

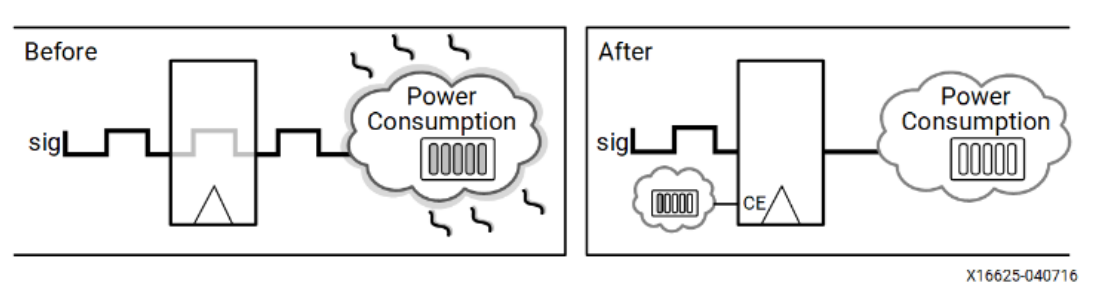


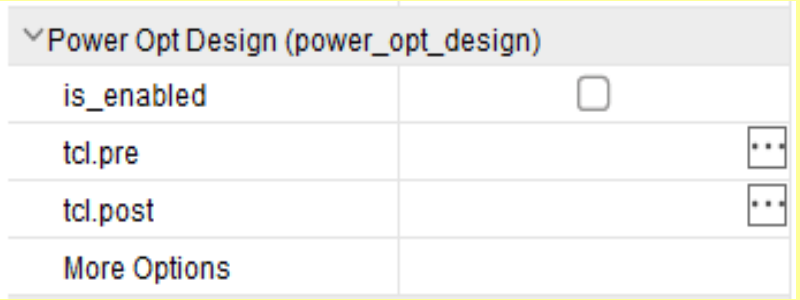
Power Optimization

- 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.



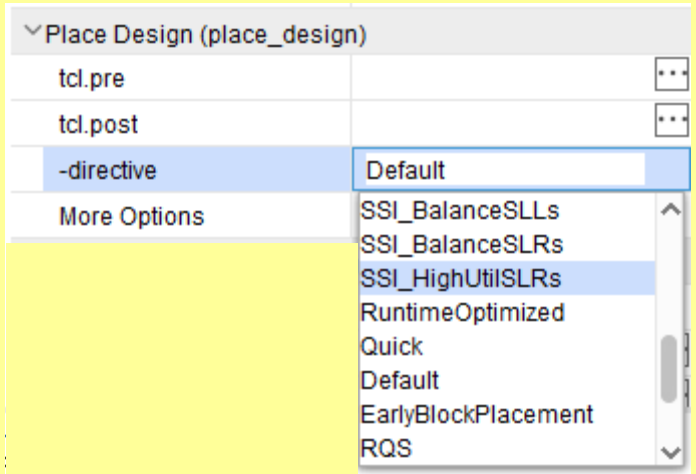


Placement

- 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

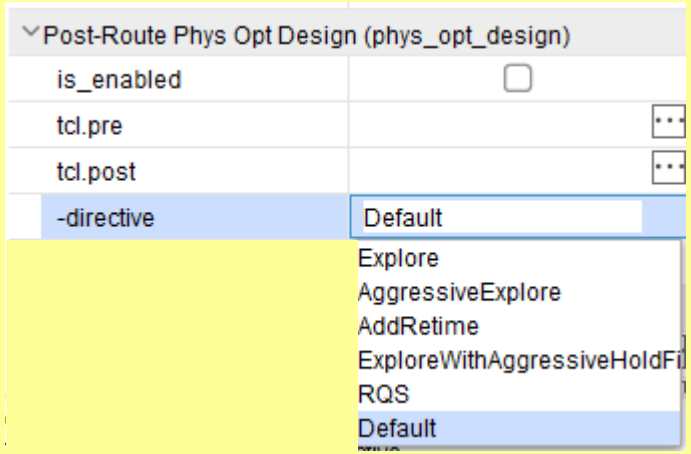


Physical optimization

- Logic Optimization
- Power Optimization
- Placement
- **Physical Optimization**
- Routing

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

Logic Optimization

Power Optimization

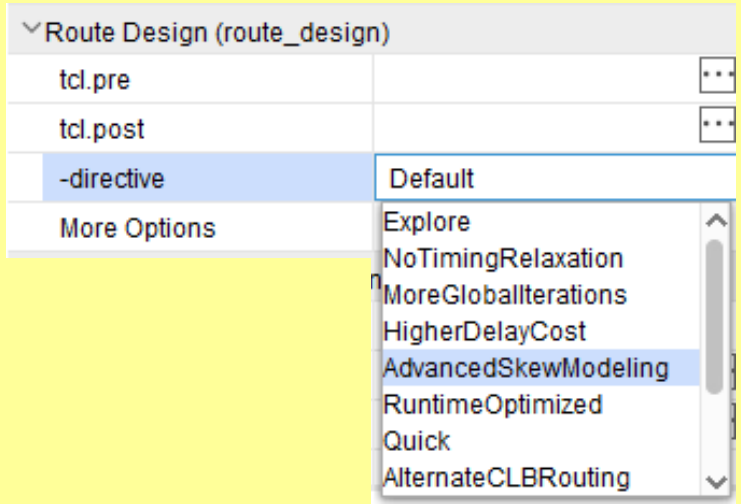
Placement

Physical Optimization

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

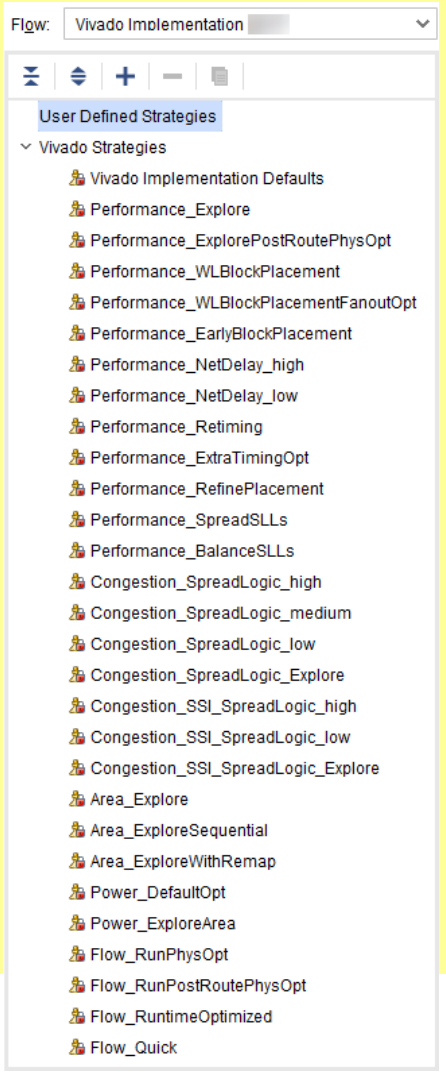


AMD XILINX

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

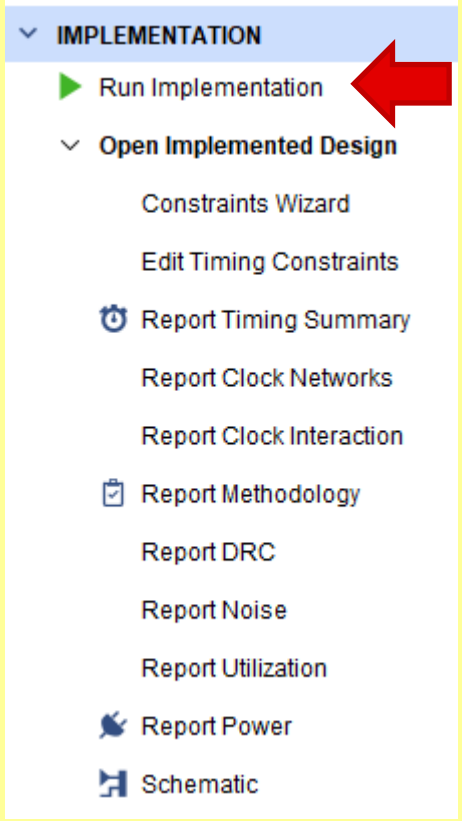


Run Implementation

Accessed through the Flow Navigator by selecting Open Implemented Design

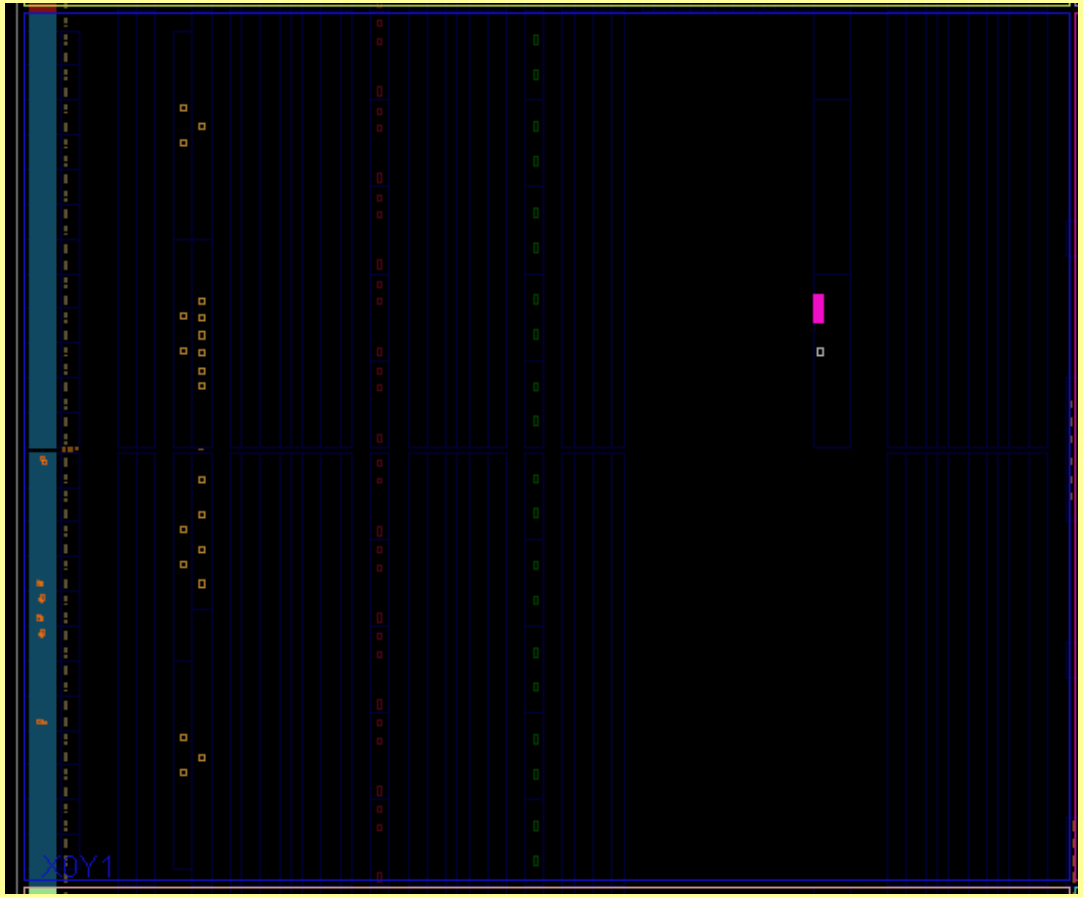
Design representation during and after implementation

- 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

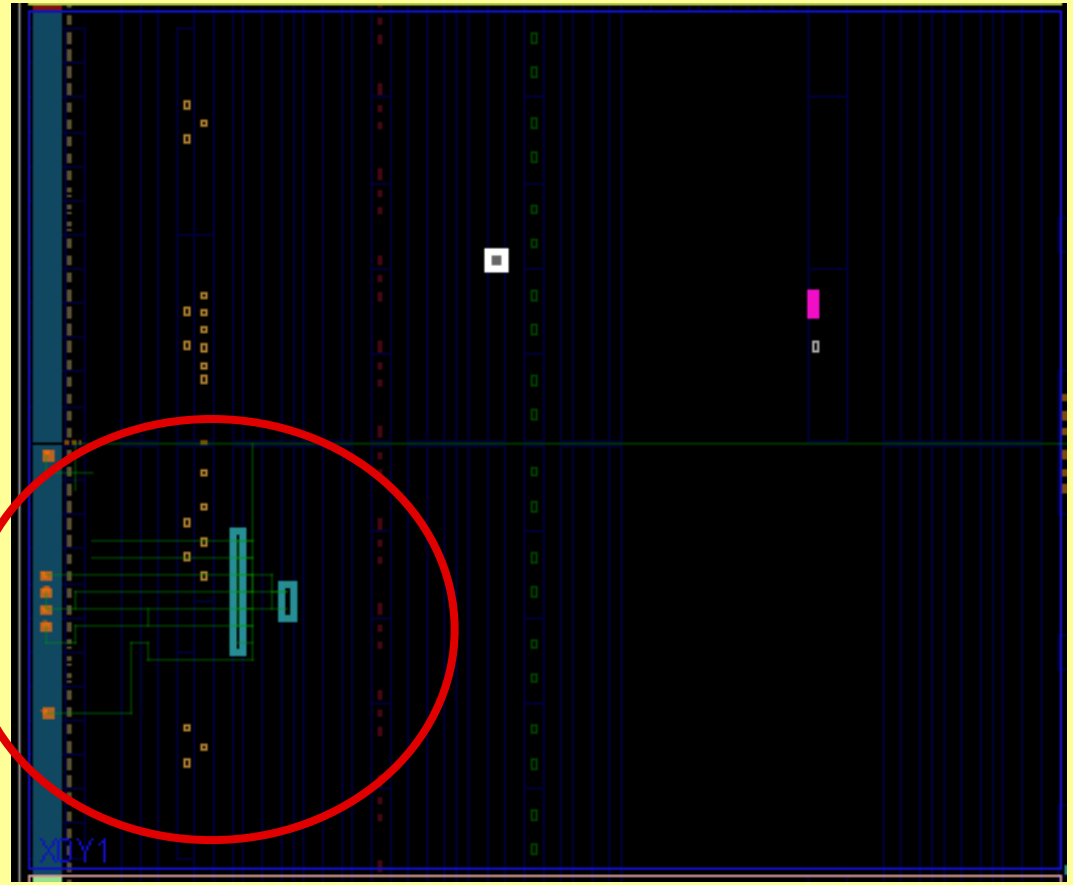


After Implementation

Synthesis stage



Implementation stage



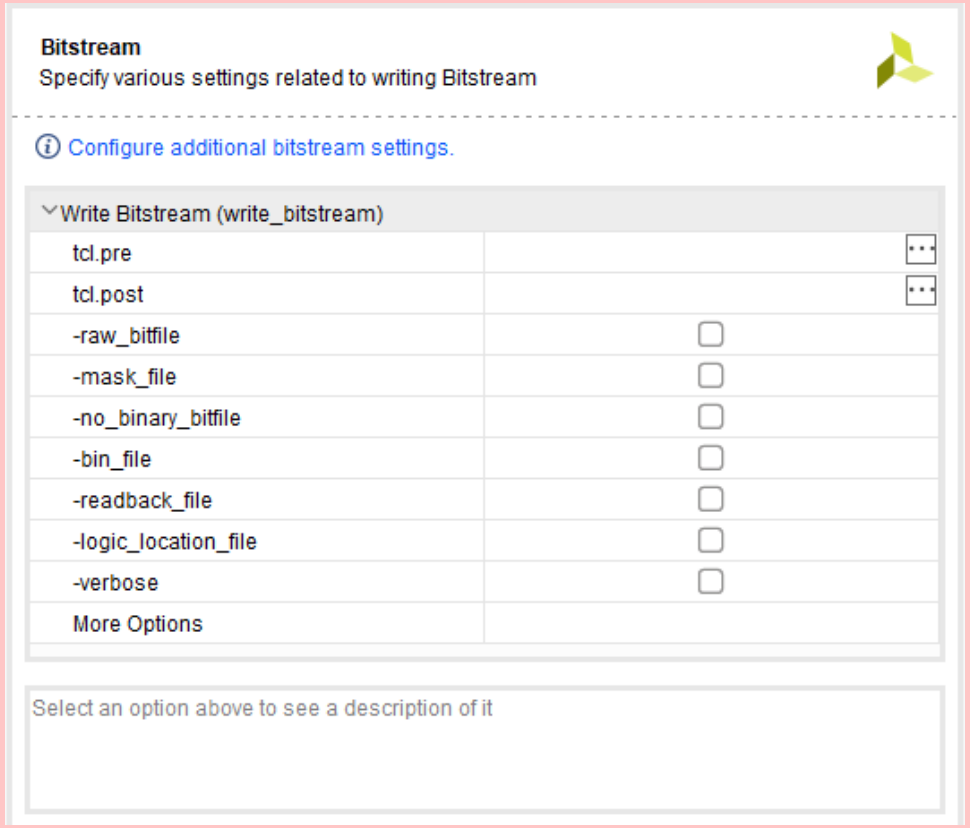
AMD
XILINX

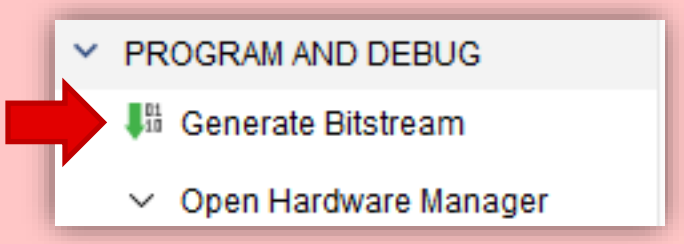
Vivado Bitstream Generation

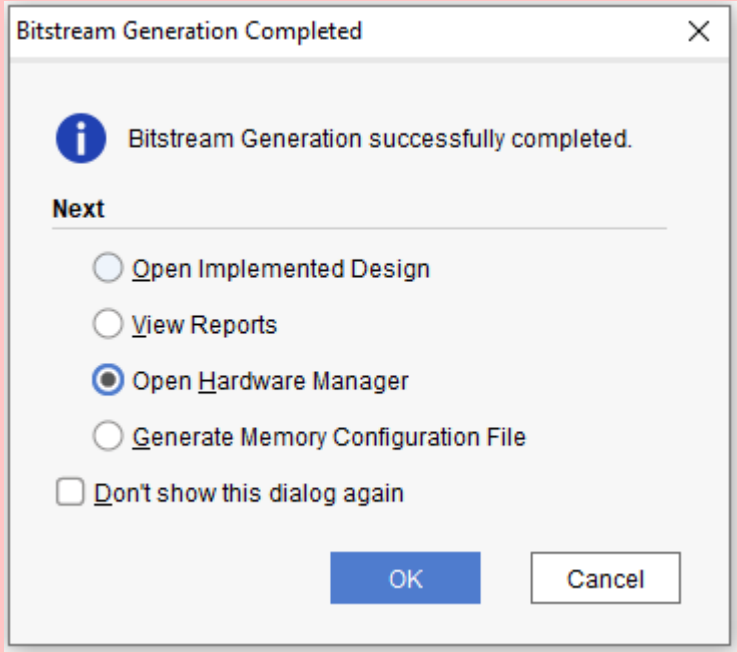
2022.2

Bitstream Generation

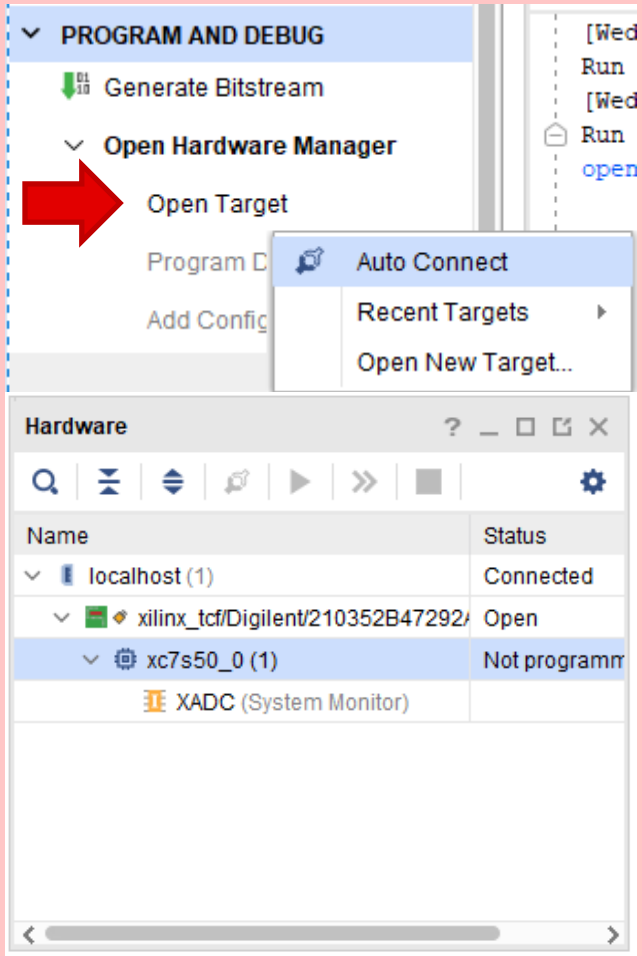
- Ensure the settings are correct

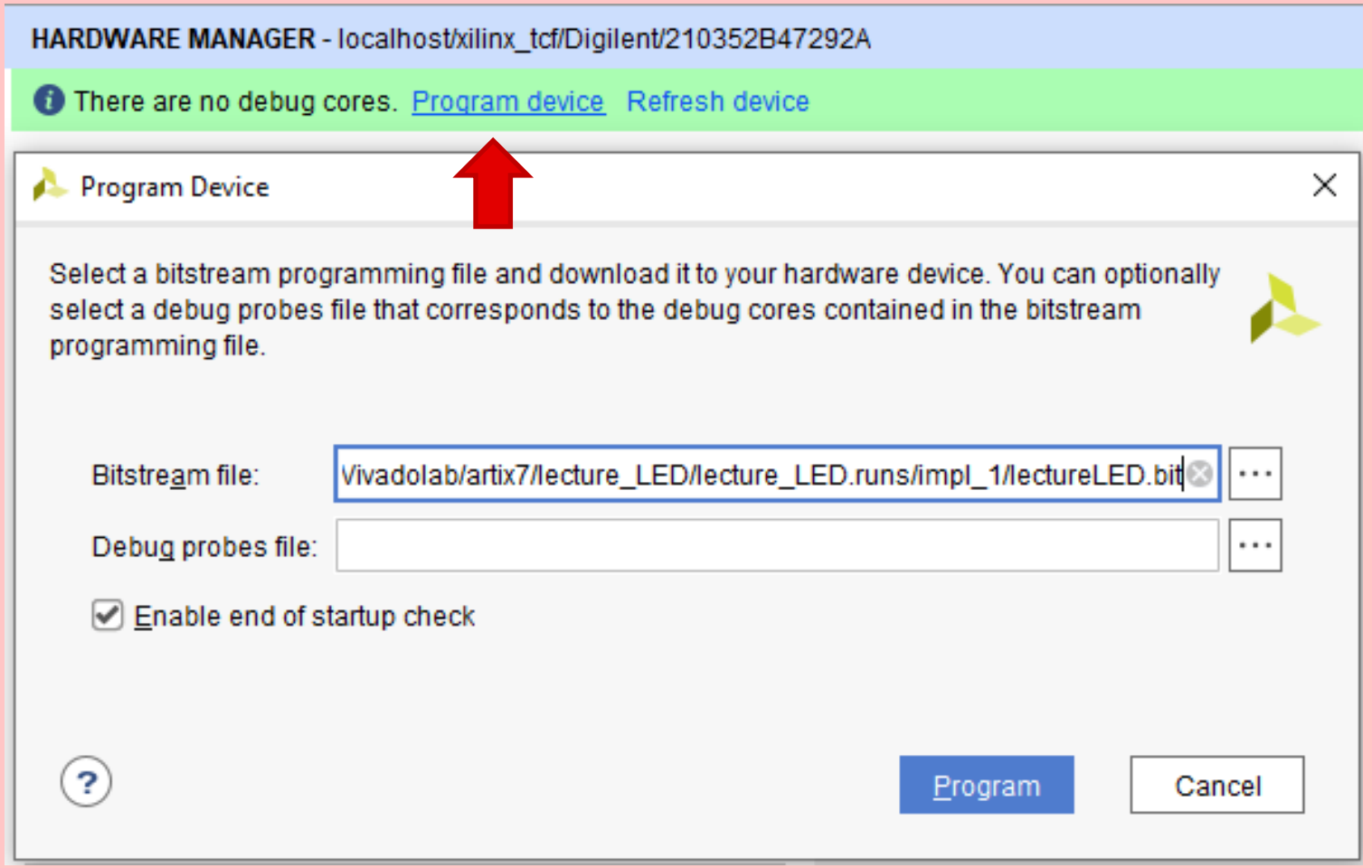






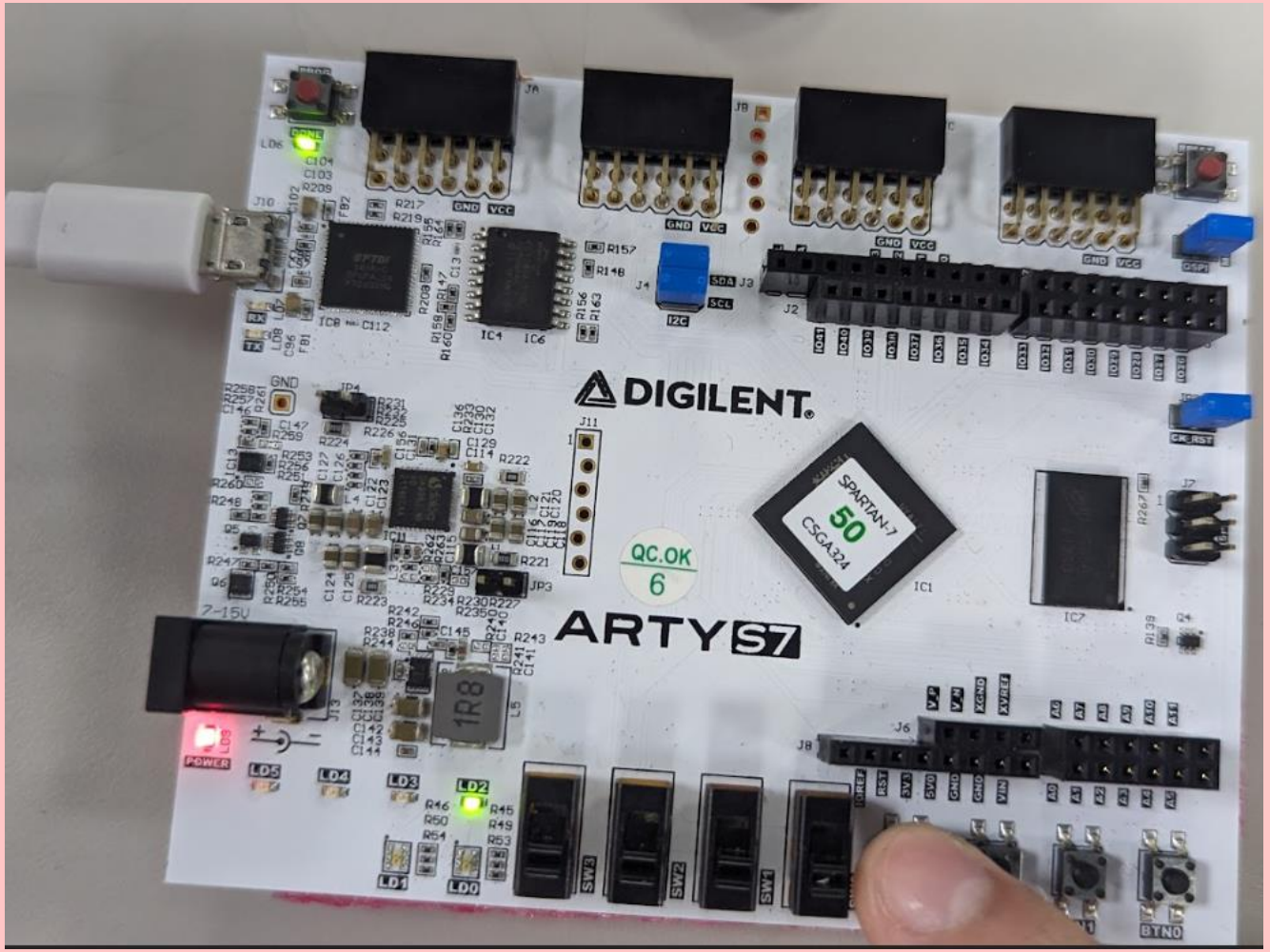
Hardware Manager



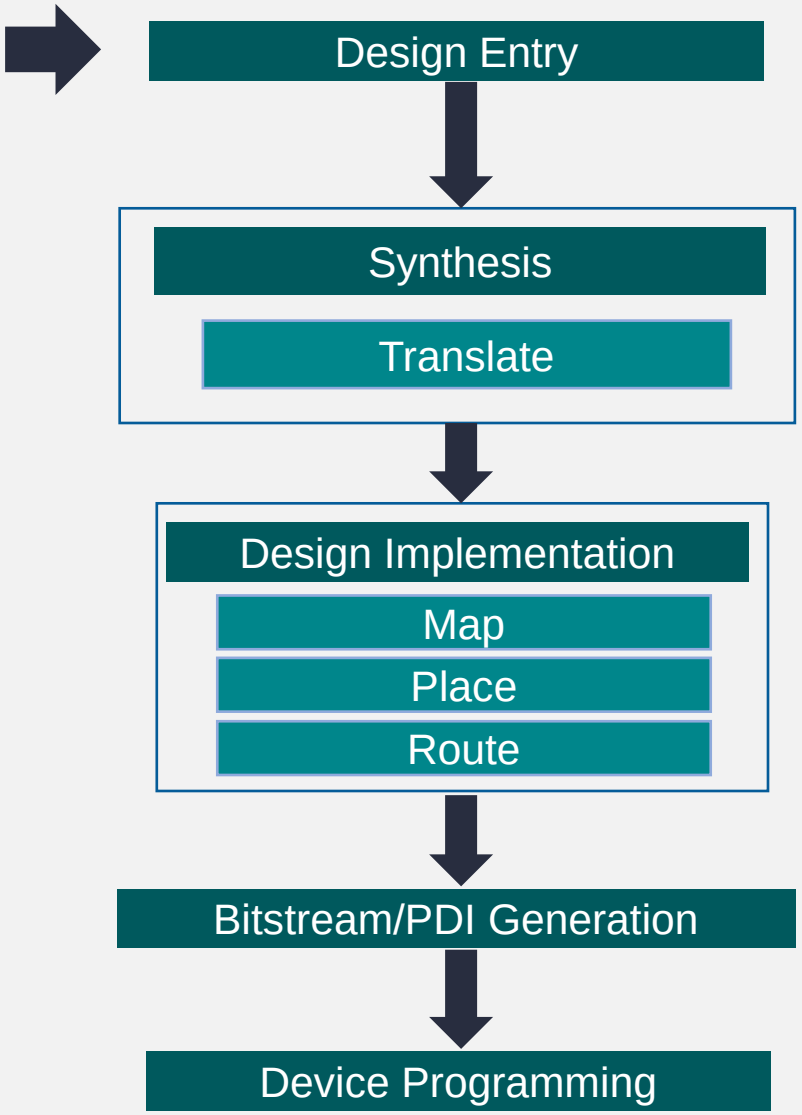


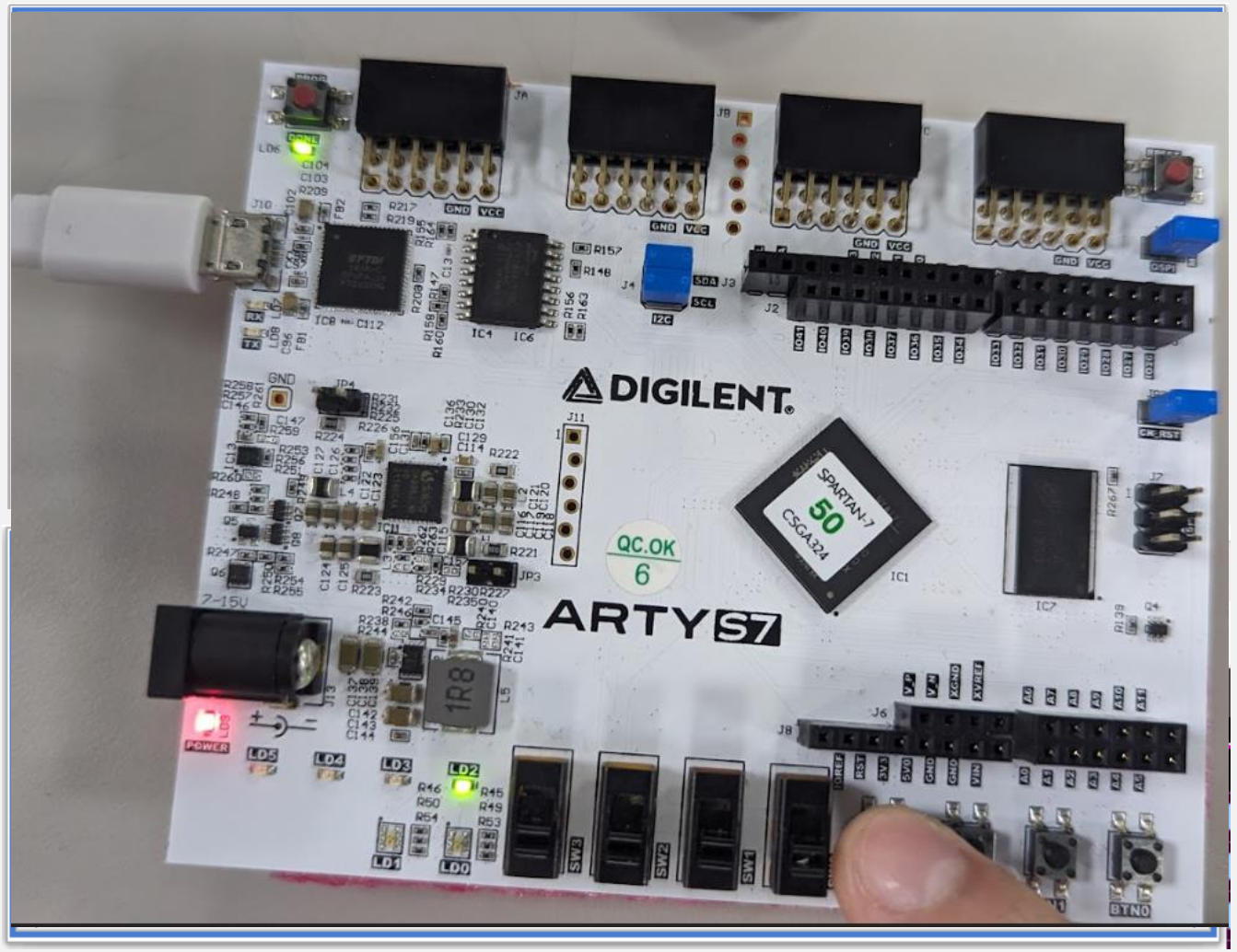
AMD XILINX

Board Test



Traditional RTL Design Flow





AMD
XILINX

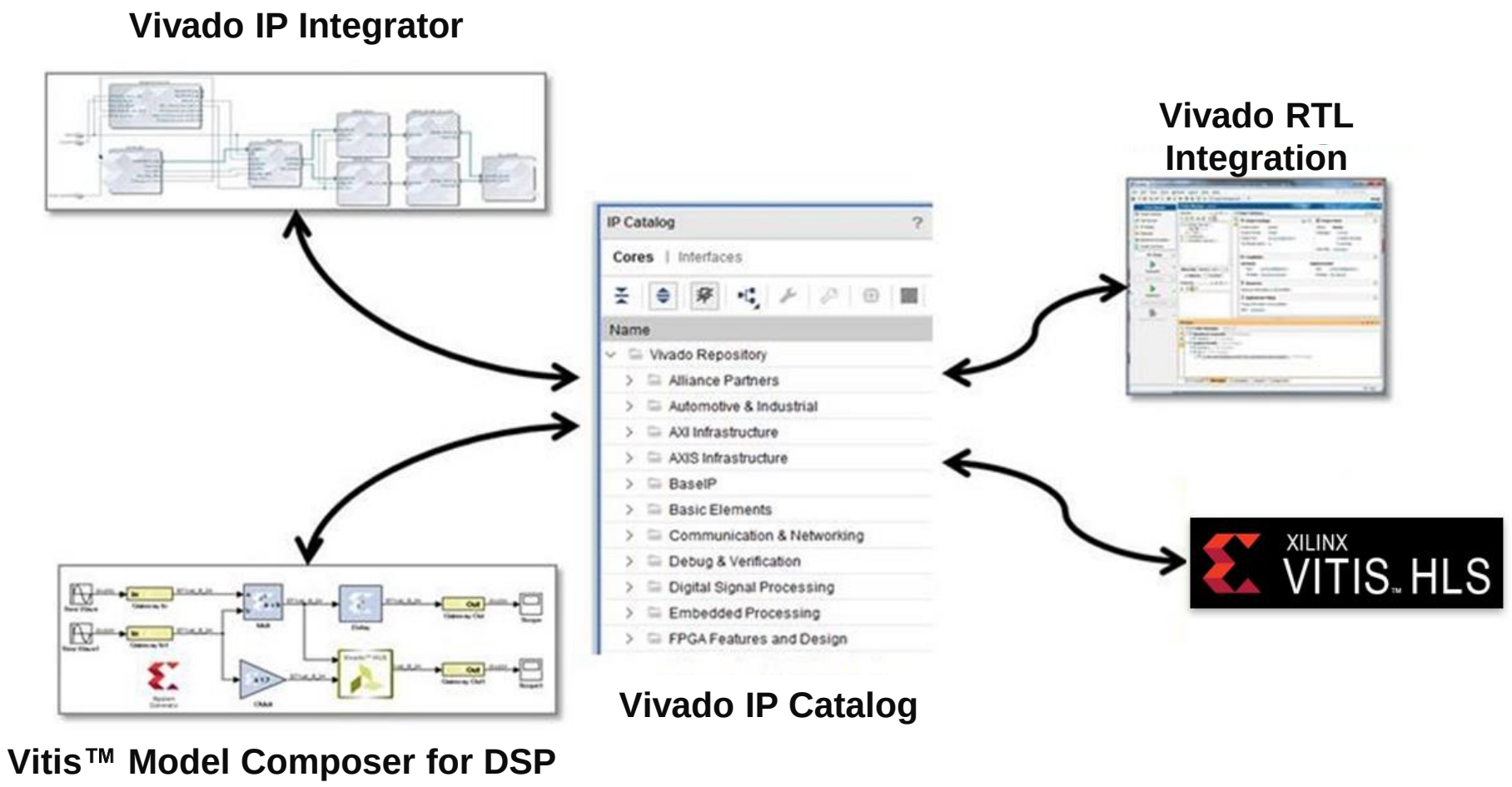
AMD
XILINX

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



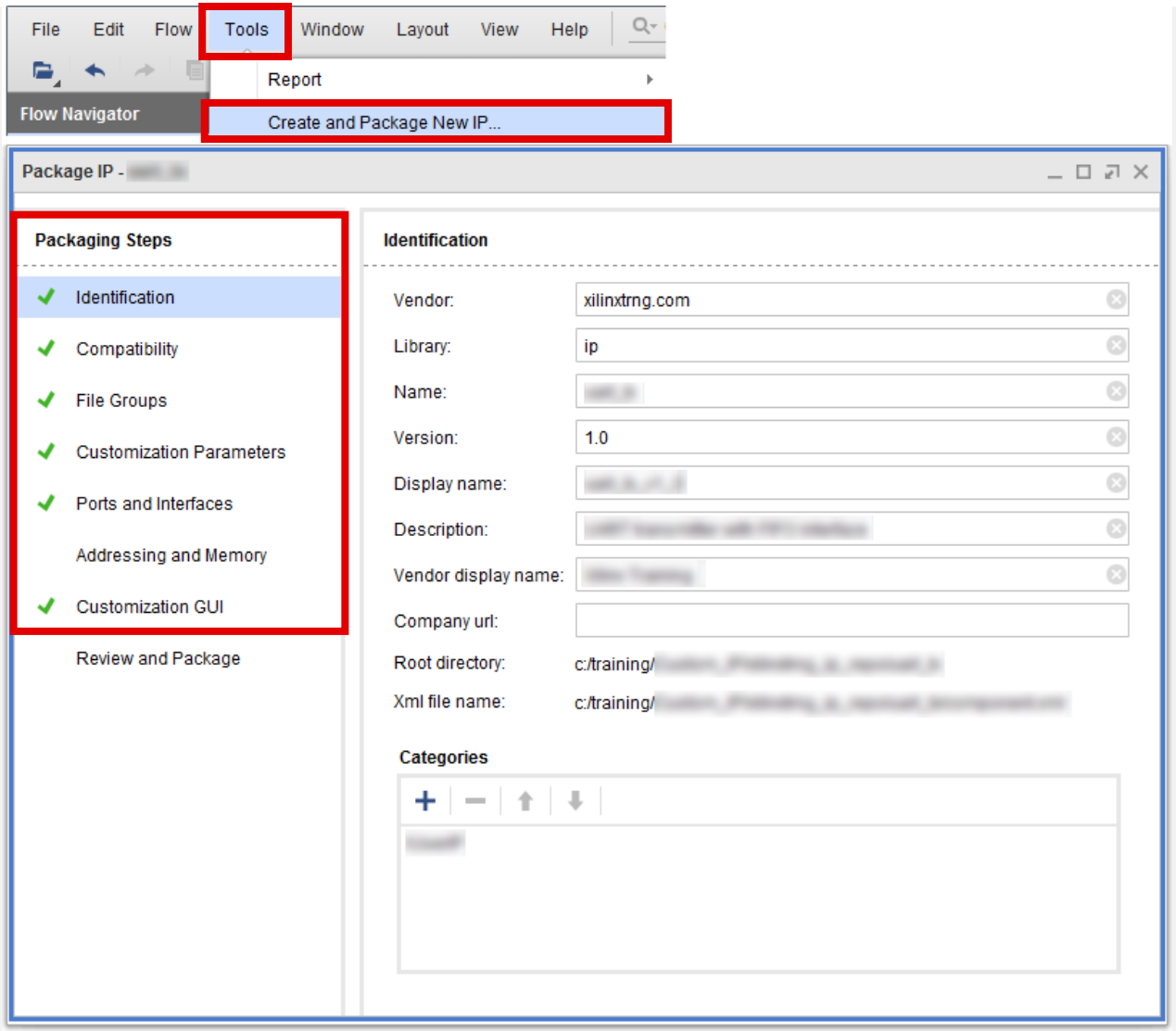
AMD
XILINX

IP Operations

The Vivado IP flow begins by accessing the IP catalog

AMD XILINX

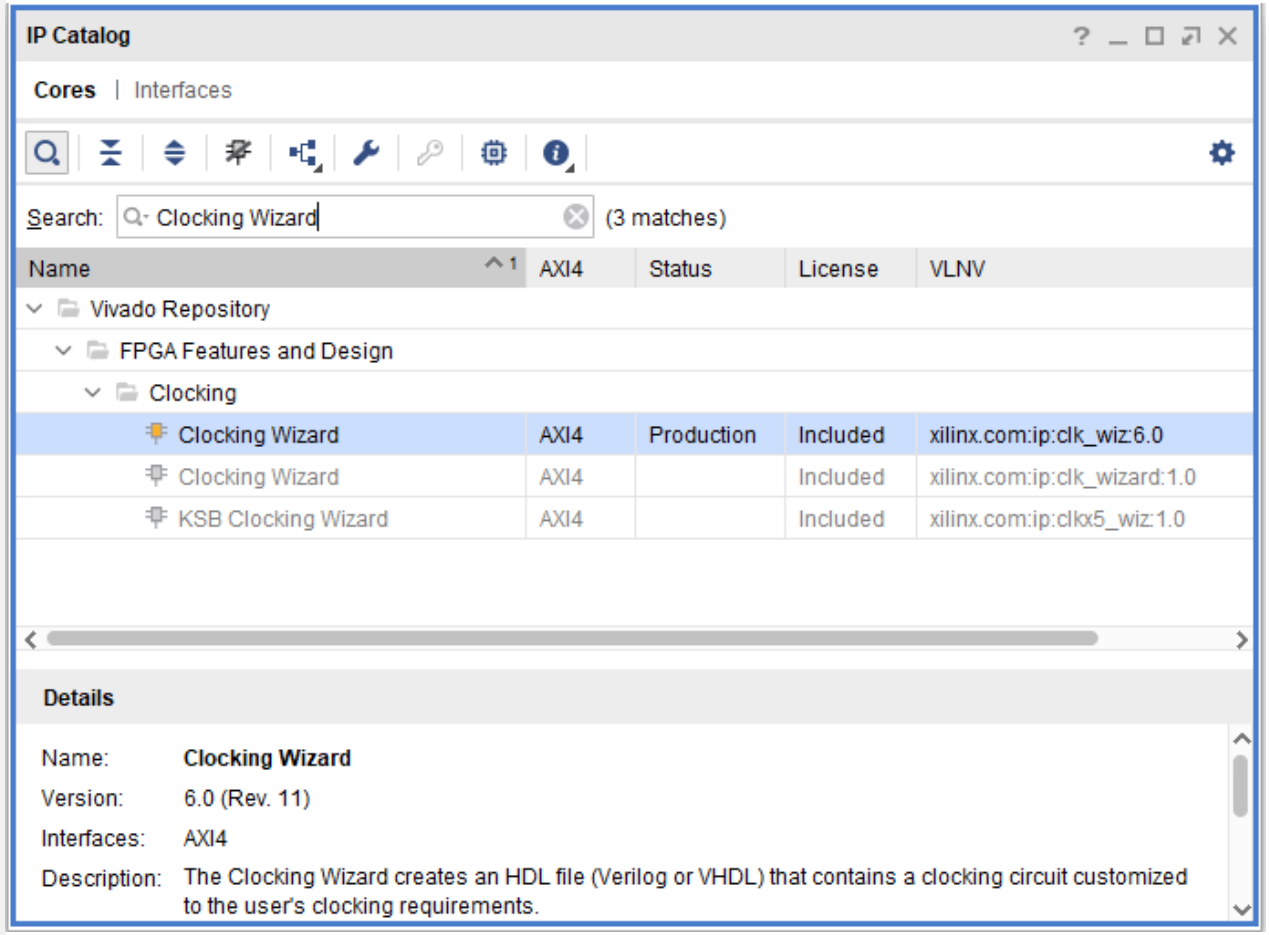
IP Packager



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

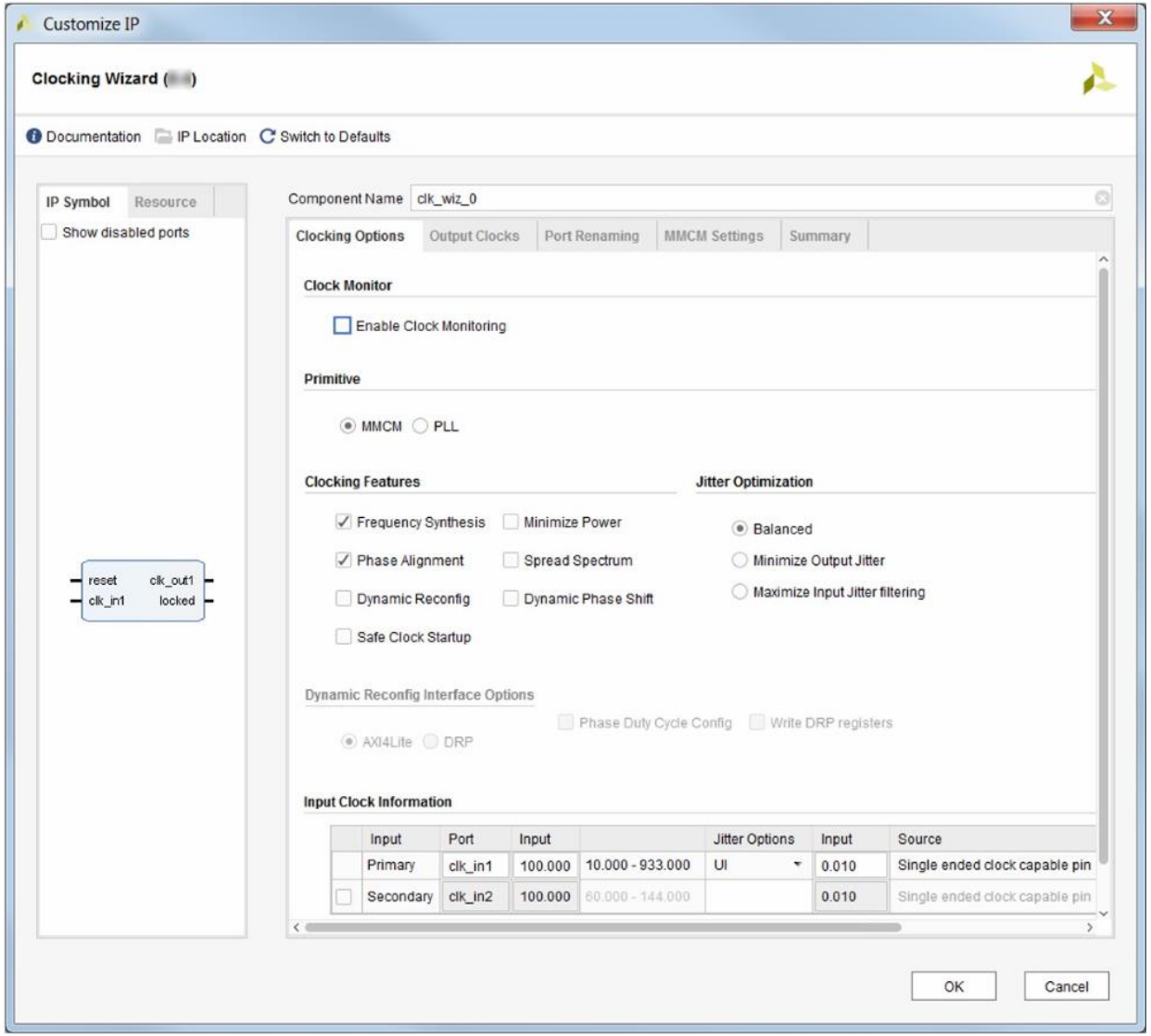


AMD XILINX

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

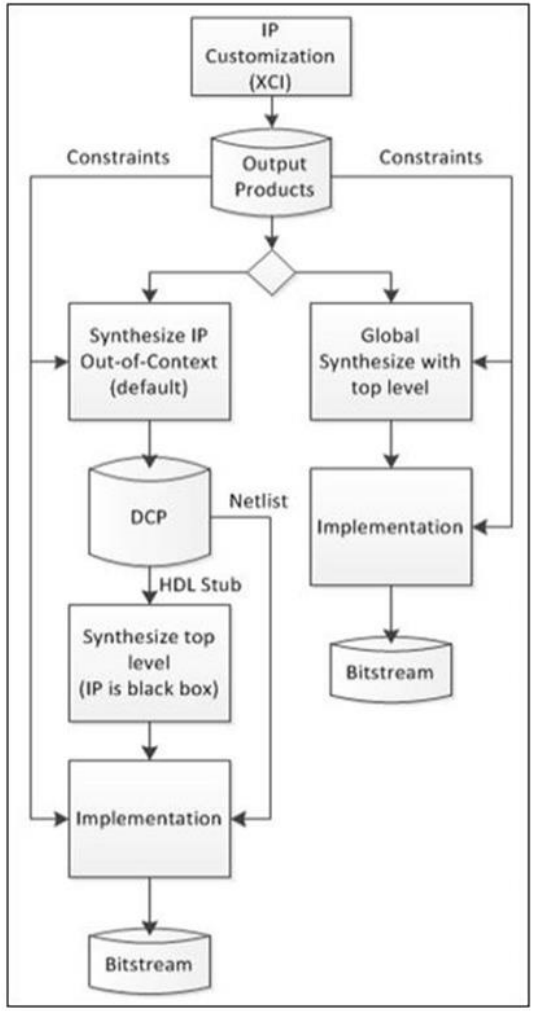


AMD XILINX

IP Synthesis Options

Two ways to reference IP in a design

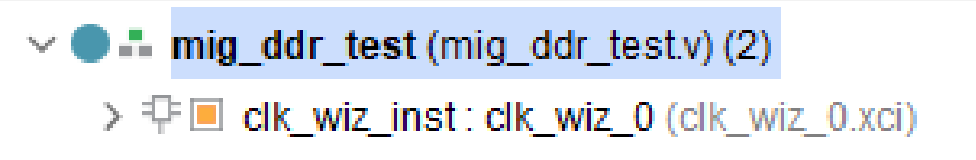
Out of context (OOC)-based (default flow)	Global synthesis
All IP catalog-delivered Vivado IP uses the OOC flow by default ▪ Default settings can be overridden DCP file read for IP as a netlist, avoiding re-synthesis Can validate timing for IP at the synthesis level	Project-synthesized IP RTL Unlike the OOC flow, RTL changes (other than IP) lead to resynthesized IP Synthesis runs for each IP instantiation



AMD XILINX

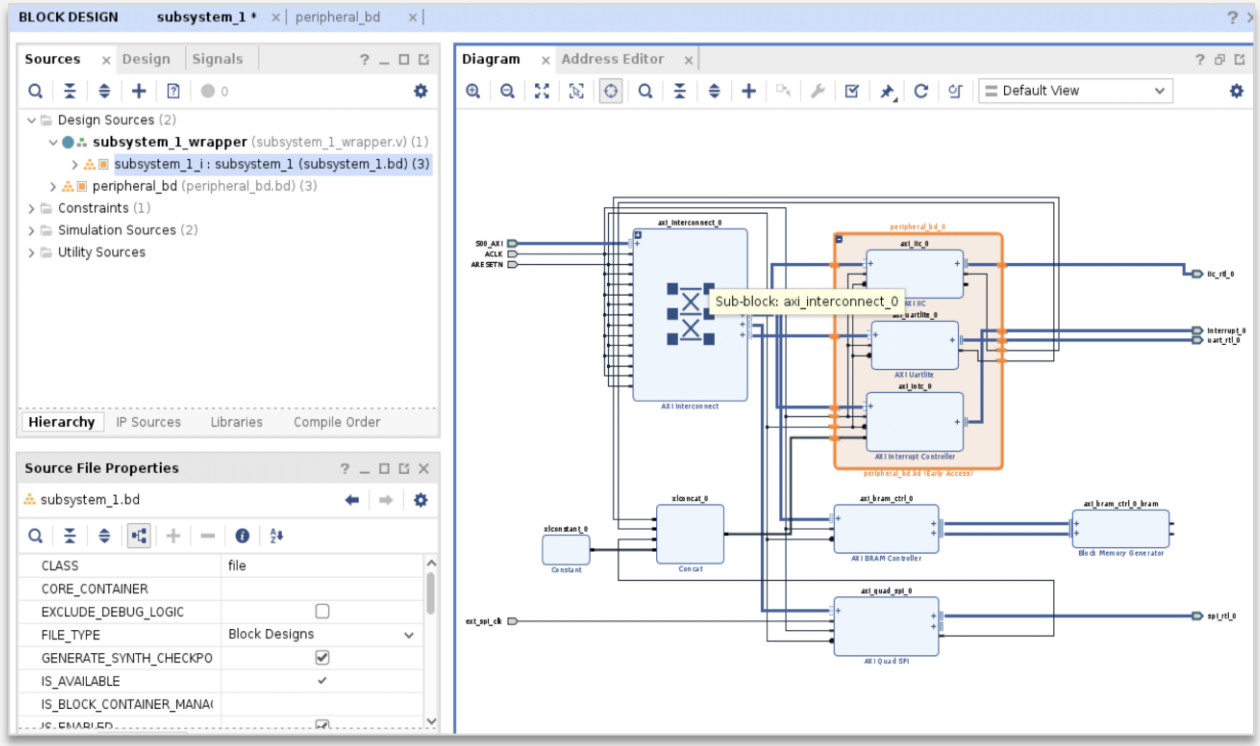
Adding IP to the Design

Instantiating IP




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

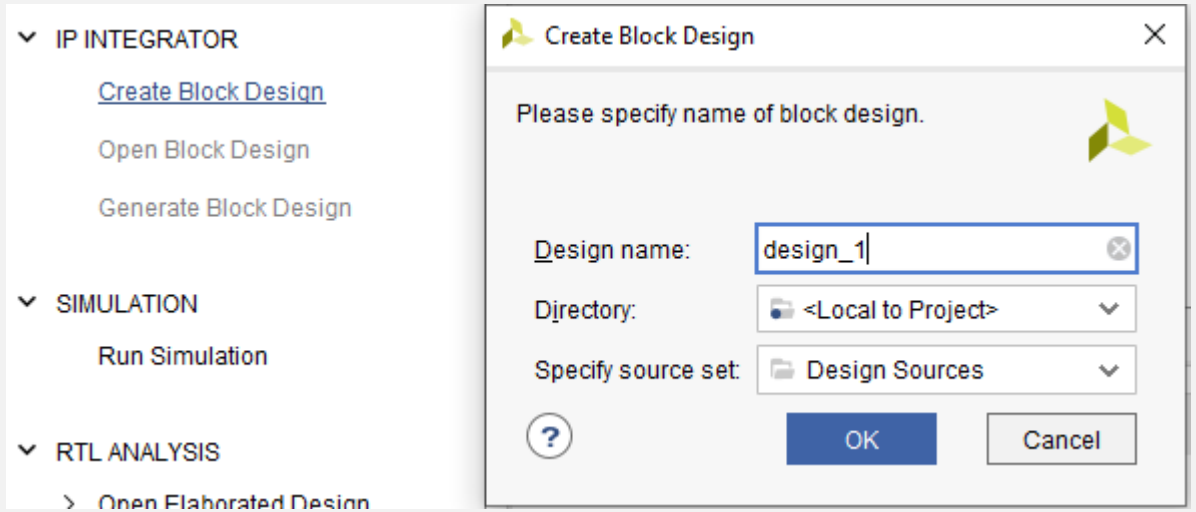
Block Design



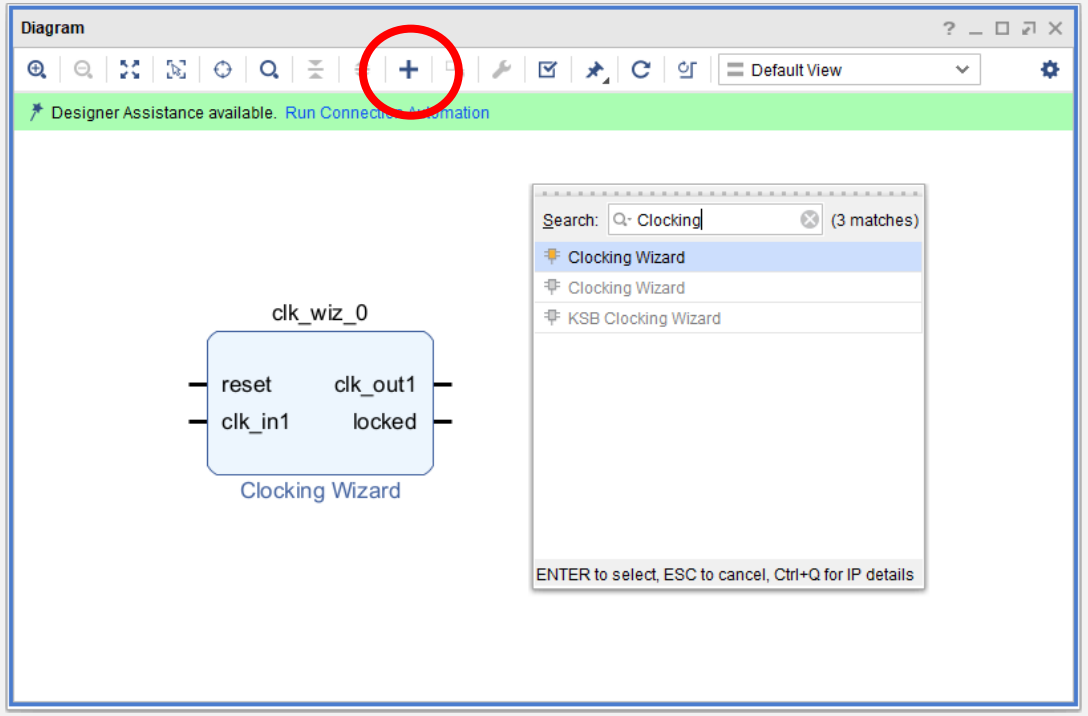
AMD XILINX

Block Design

Create Block Design



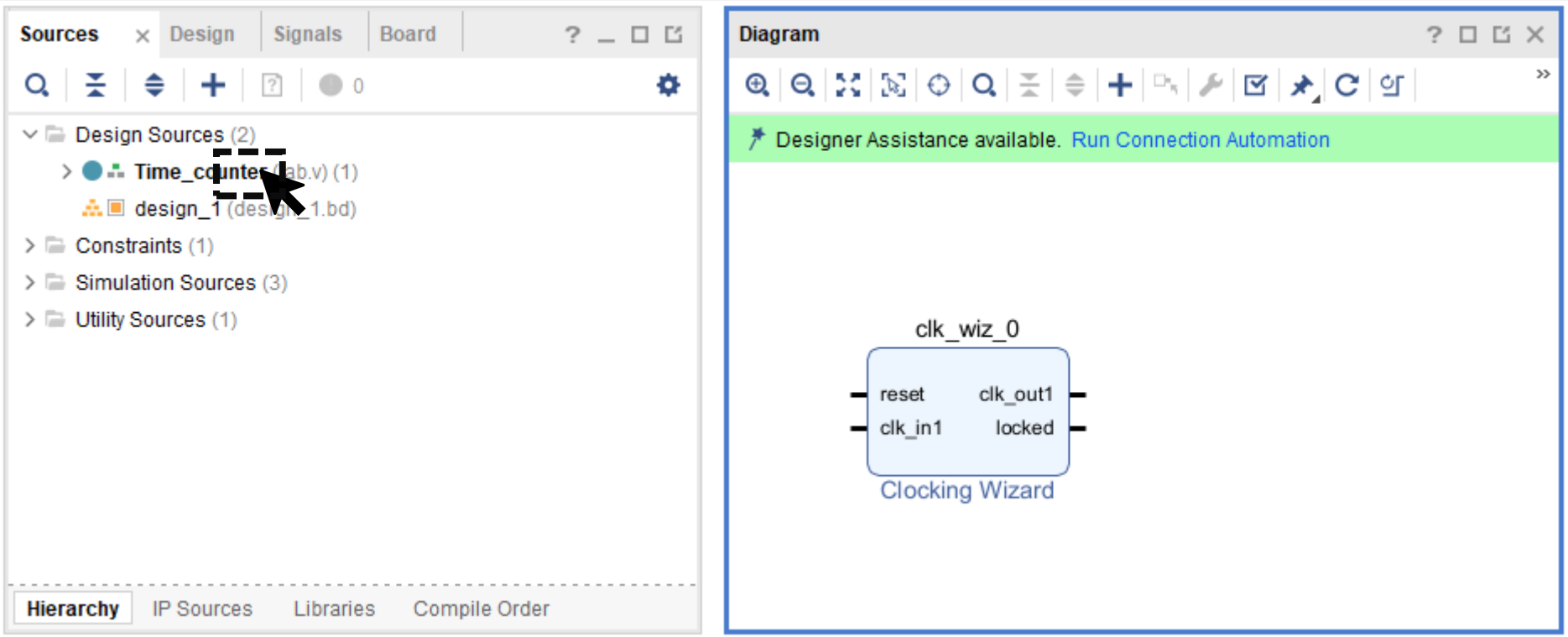
Add IP to the Block Design



AMD XILINX

Block Design

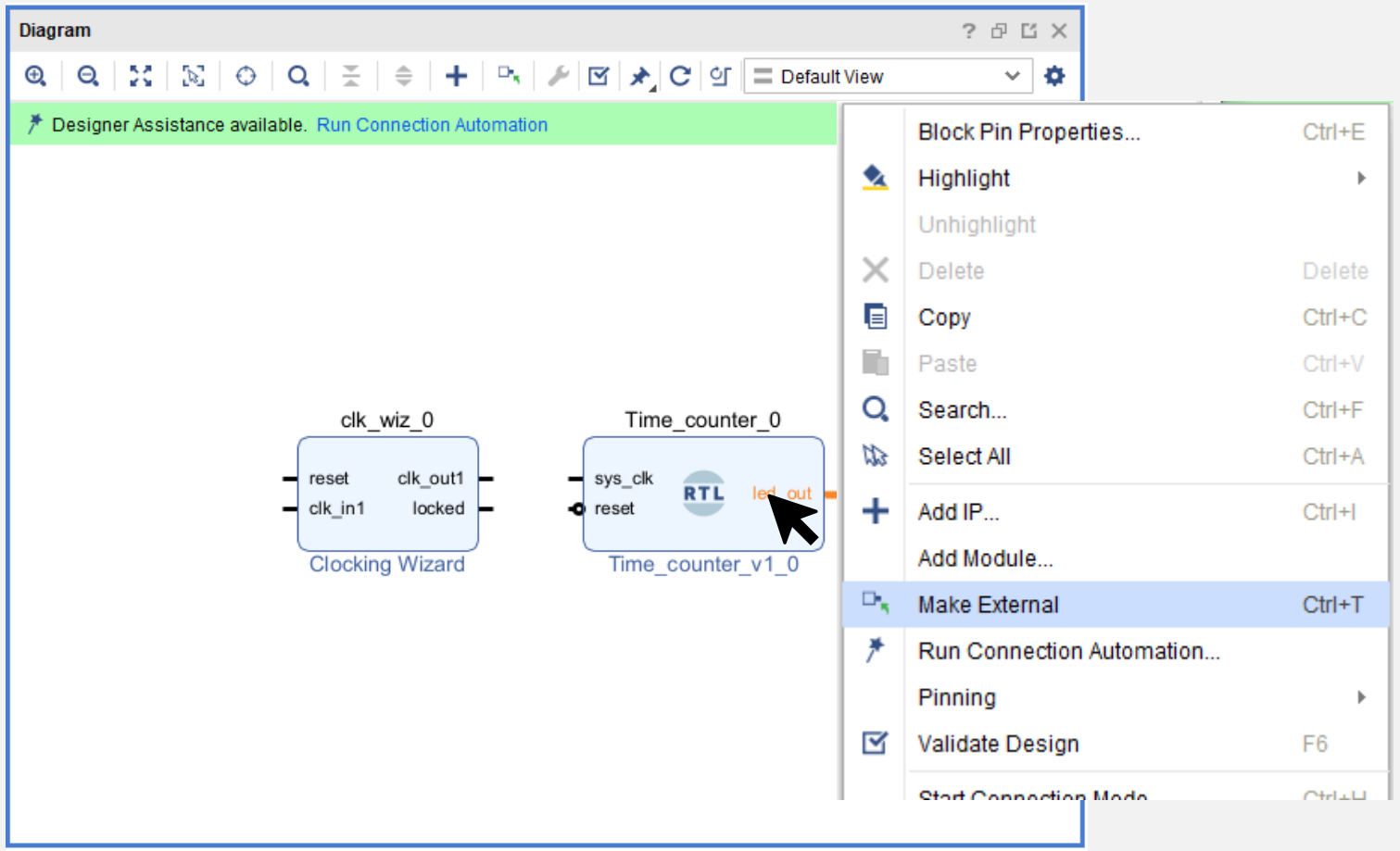
Add RTL Code to the Block Design



AMD
XILINX

Block Design

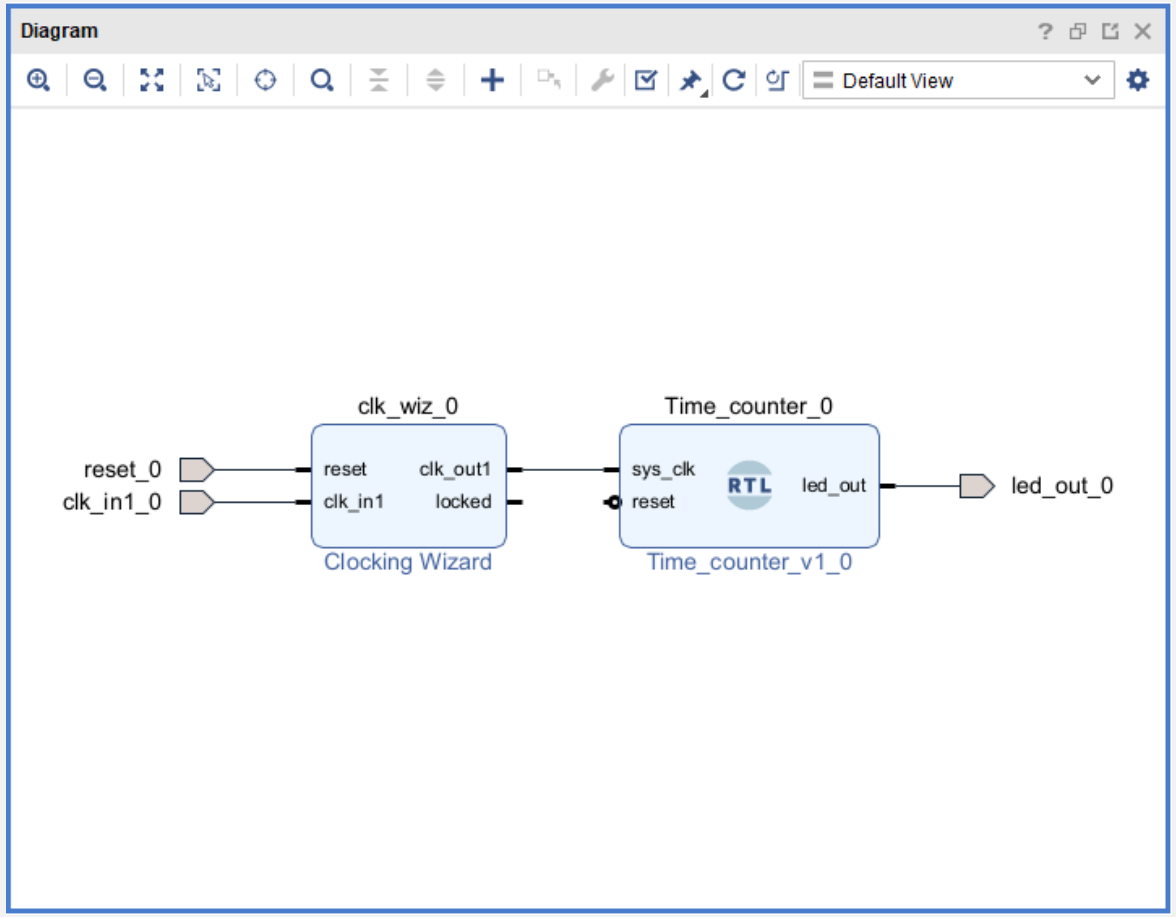
Make External Port



AMD XILINX

Block Design

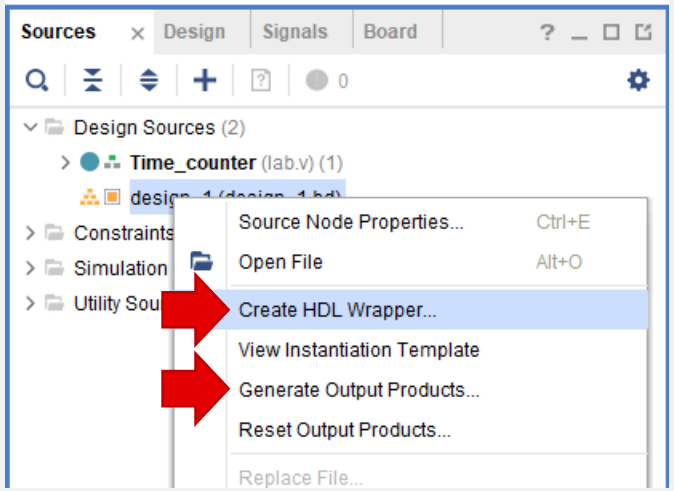
Connection

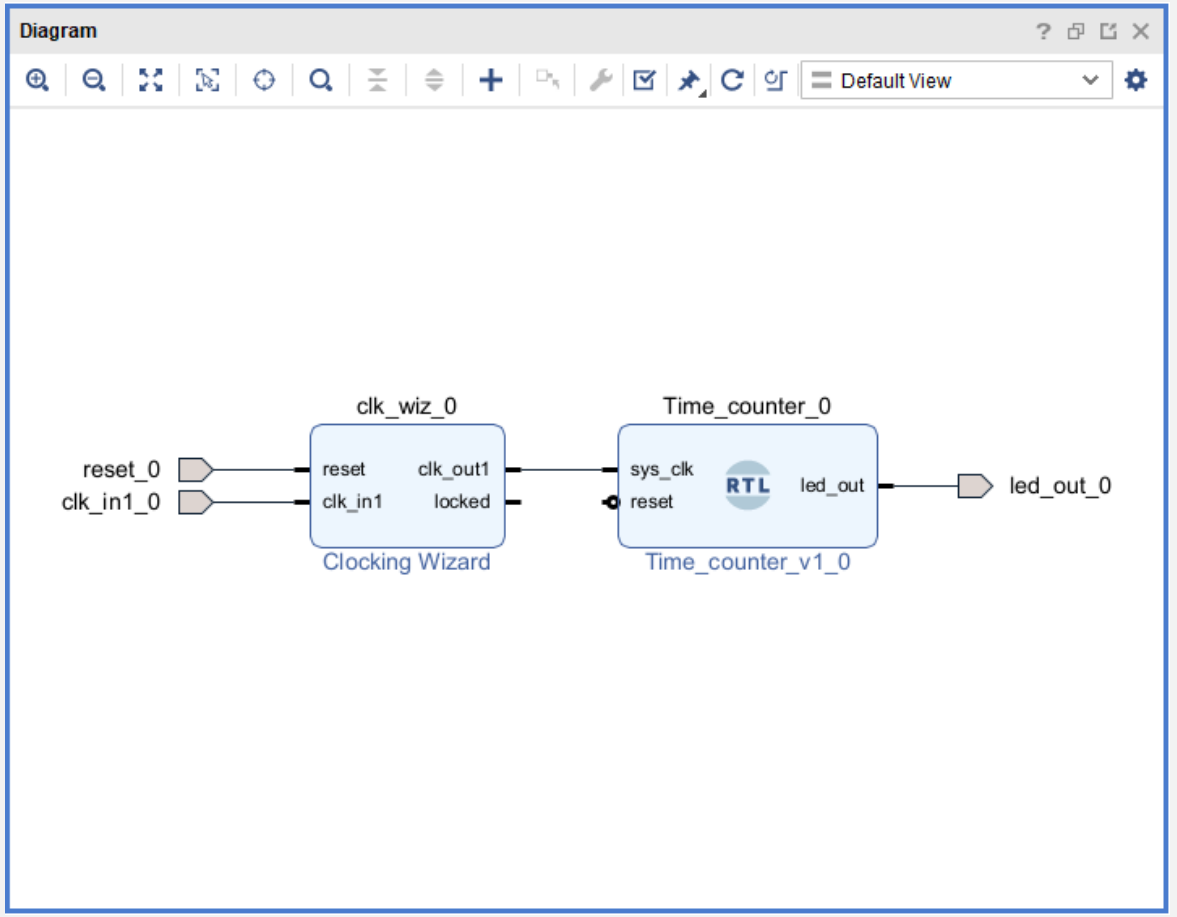


AMD XILINX

Block Design

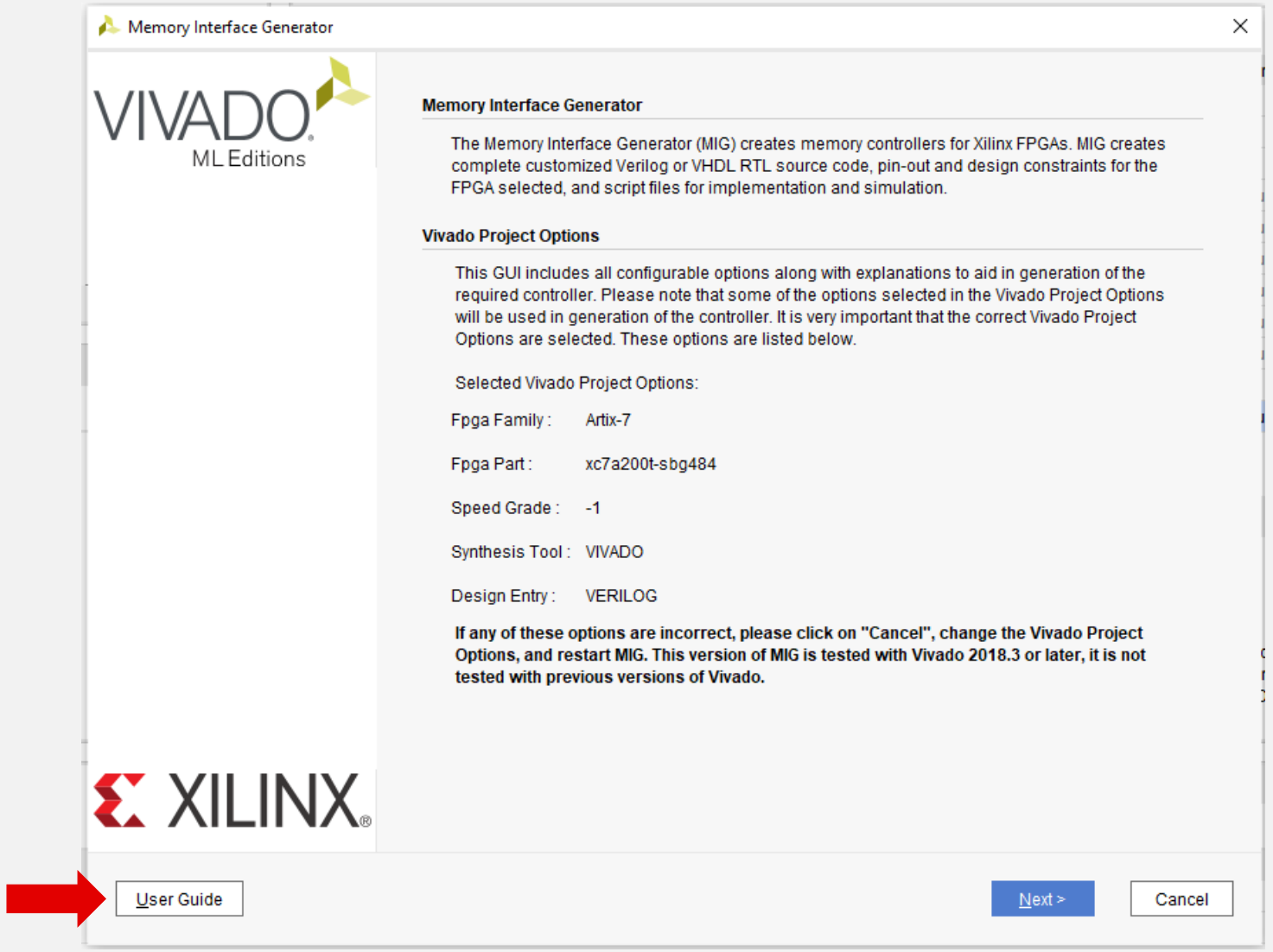
Generation





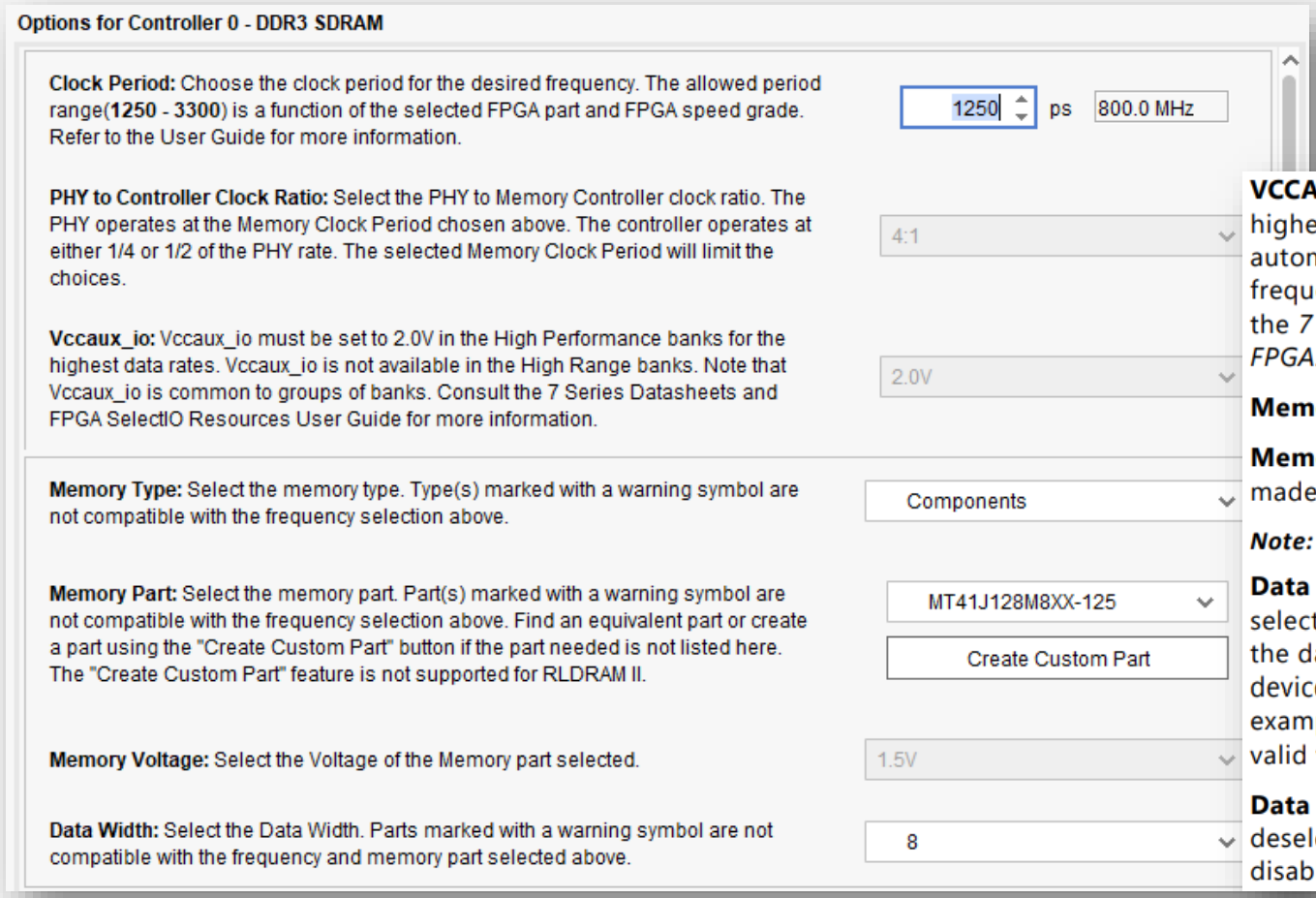
AMD XILINX

IP Use



AMD
XILINX

IP Use



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] 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 curren 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 i the last cycle of output data on app_rd_data[]. This is valid onl 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		

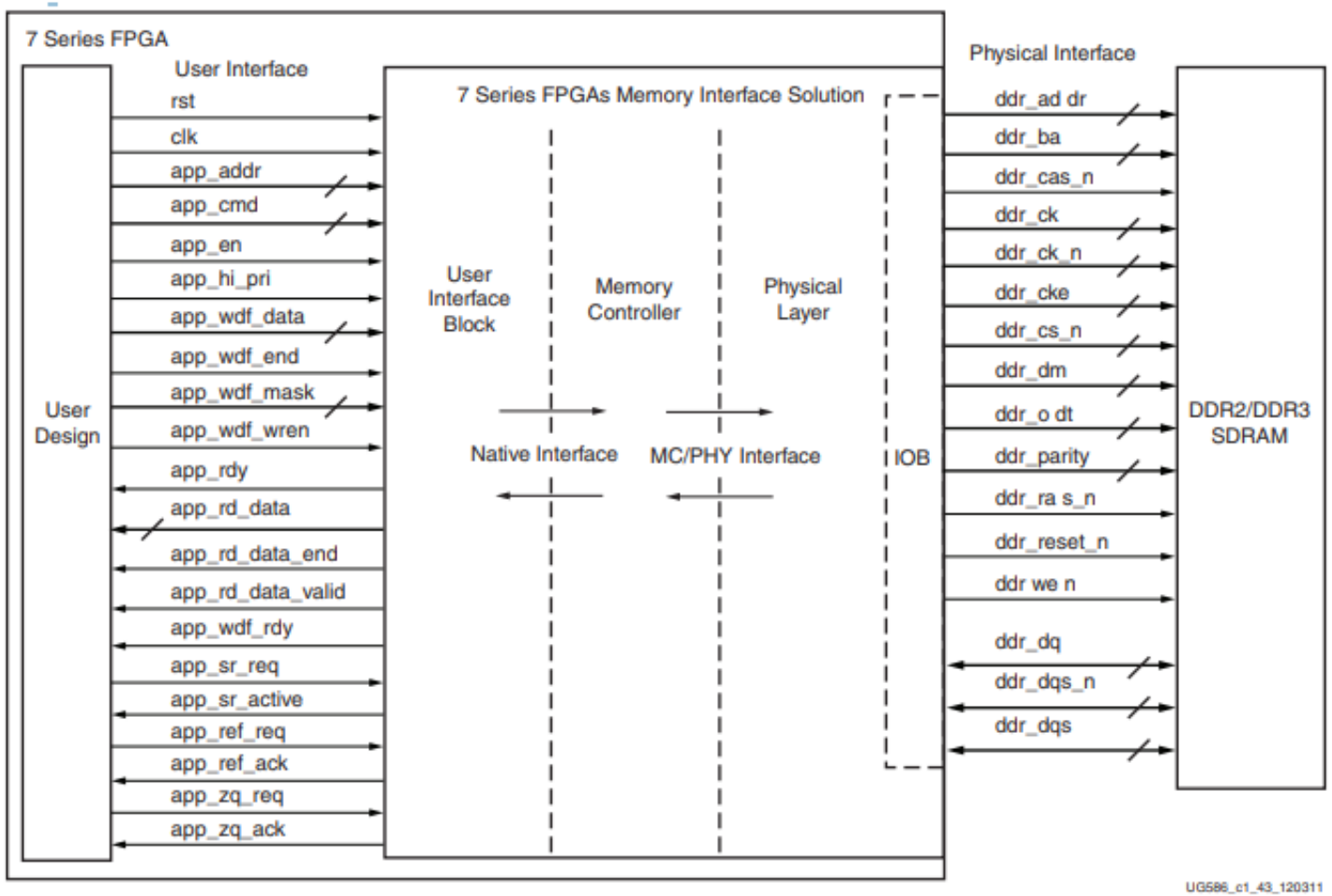


Figure 1-31: **7 Series FPGAs Memory Interface Solution**

AMD XILINX

IP Use

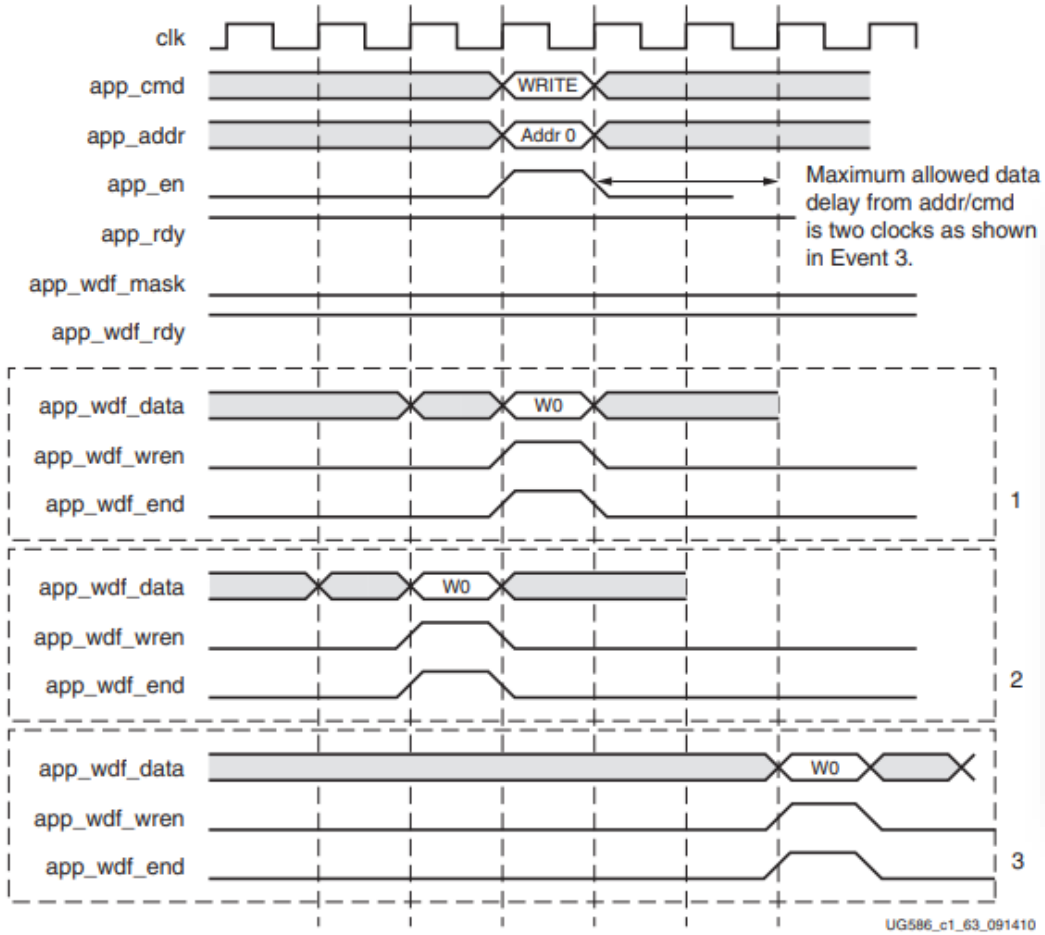


Figure 1-54: **4:1 Mode UI Interface Write Timing Diagram (Memory Burst Type = BL8)**

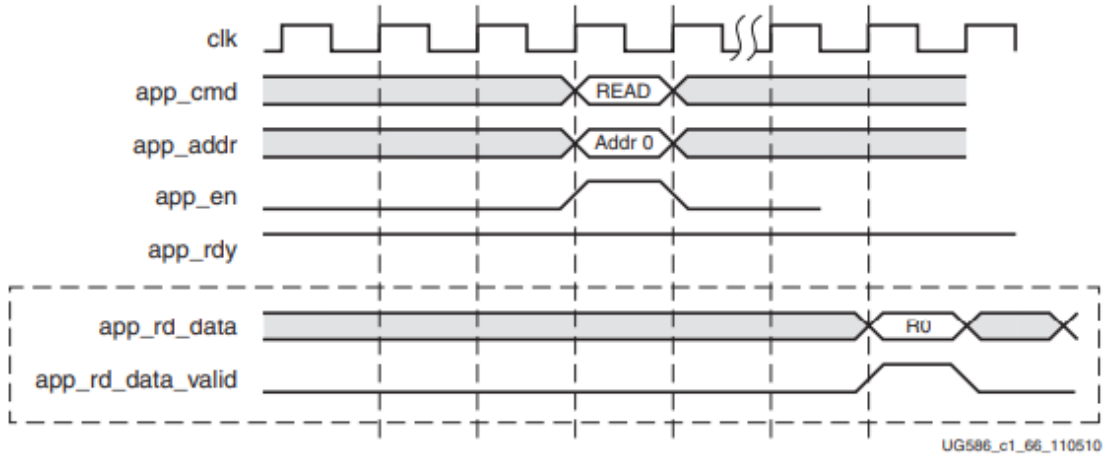
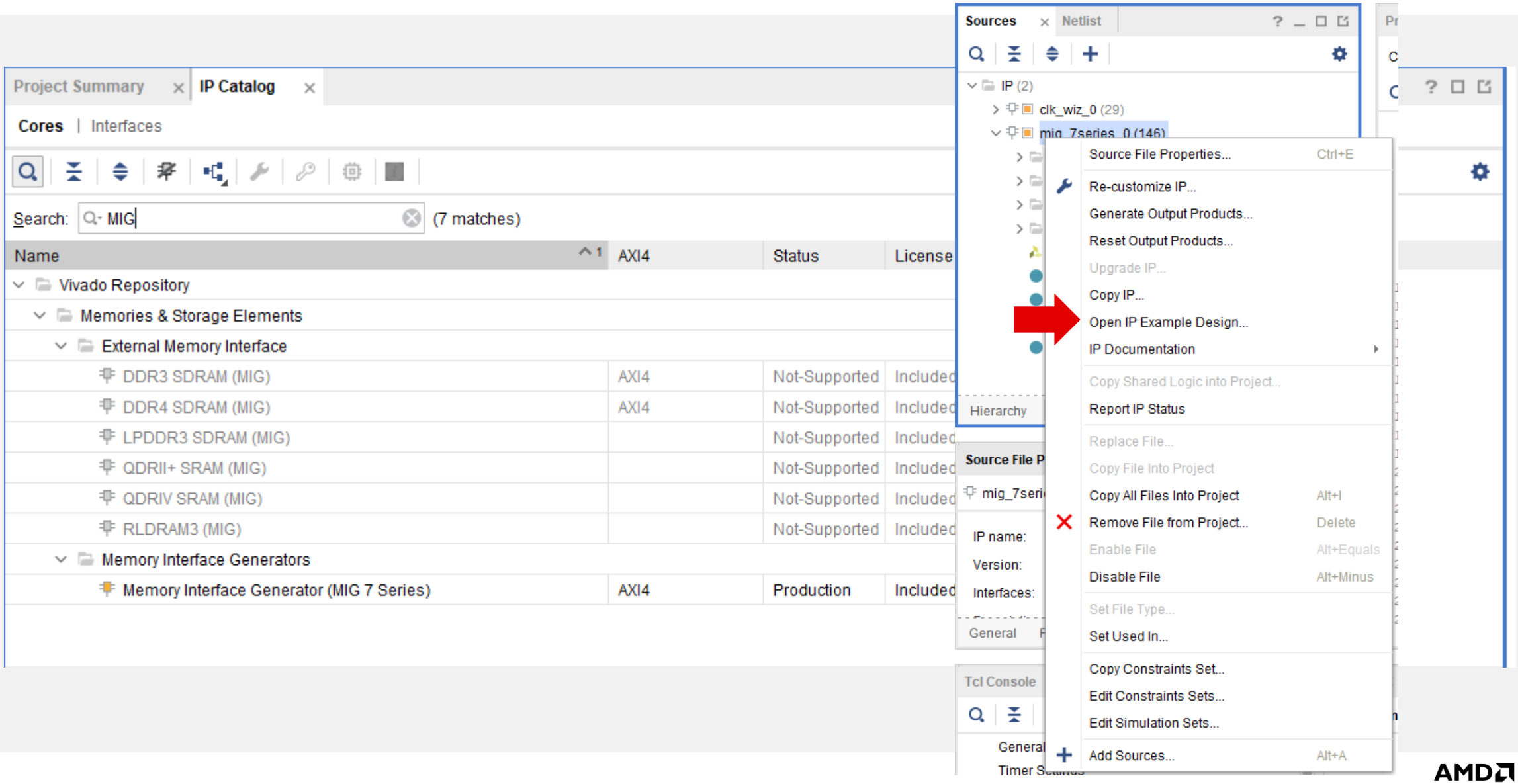


Figure 1-60: **4:1 Mode UI Interface Read Timing Diagram (Memory Burst Type = BL8)**

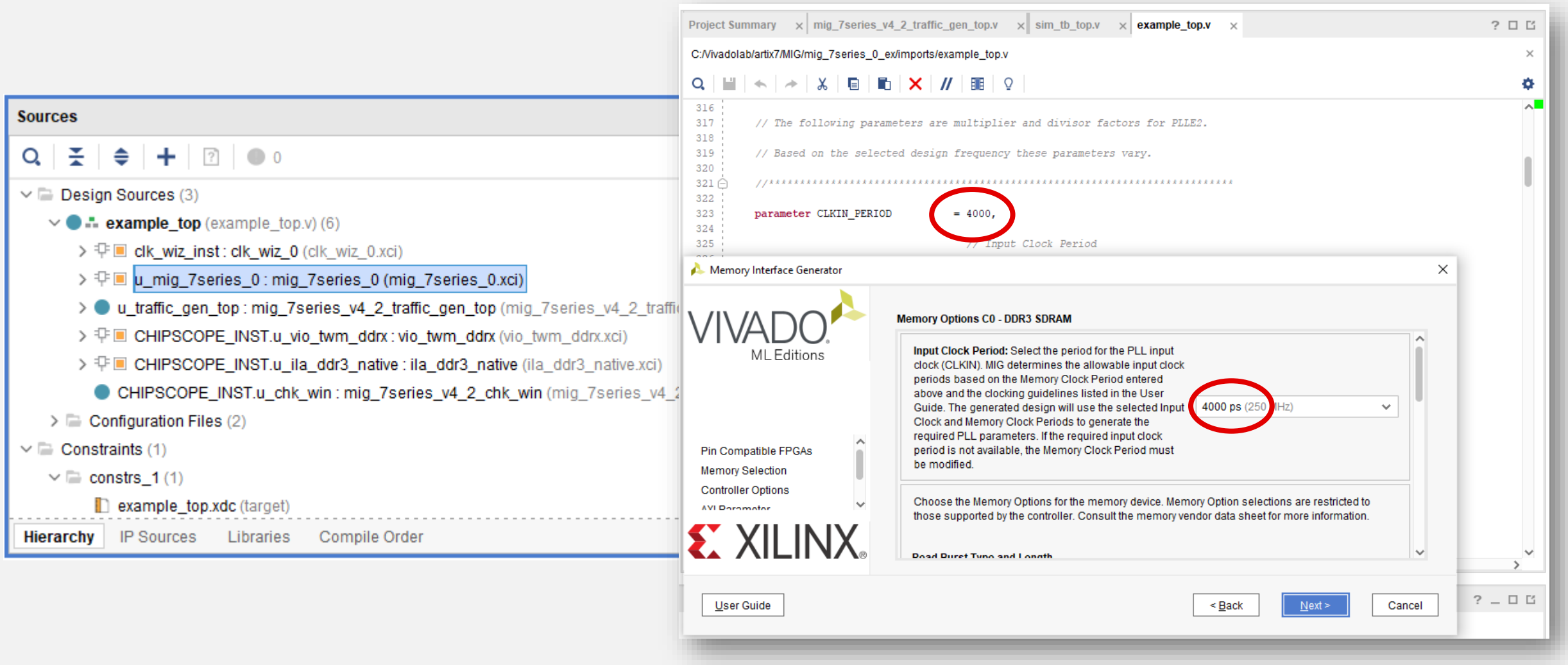
AMD XILINX

IP Use



AMD XILINX

IP Use



AMD XILINX

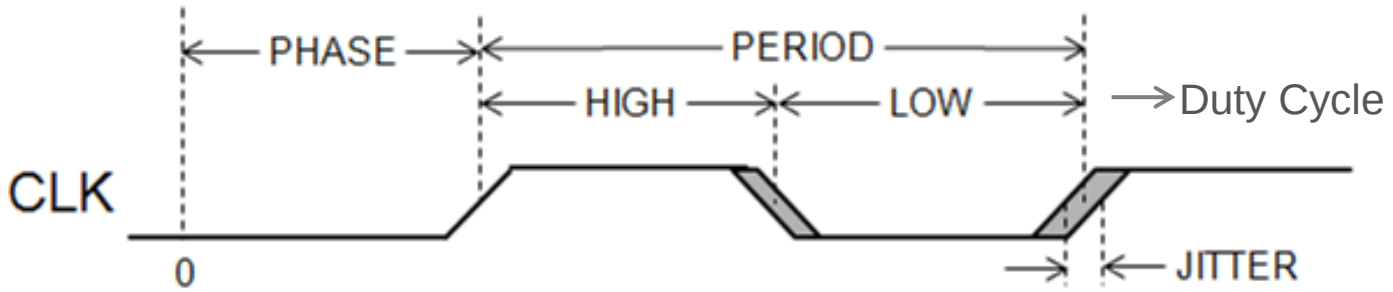
AMD
XILINX

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

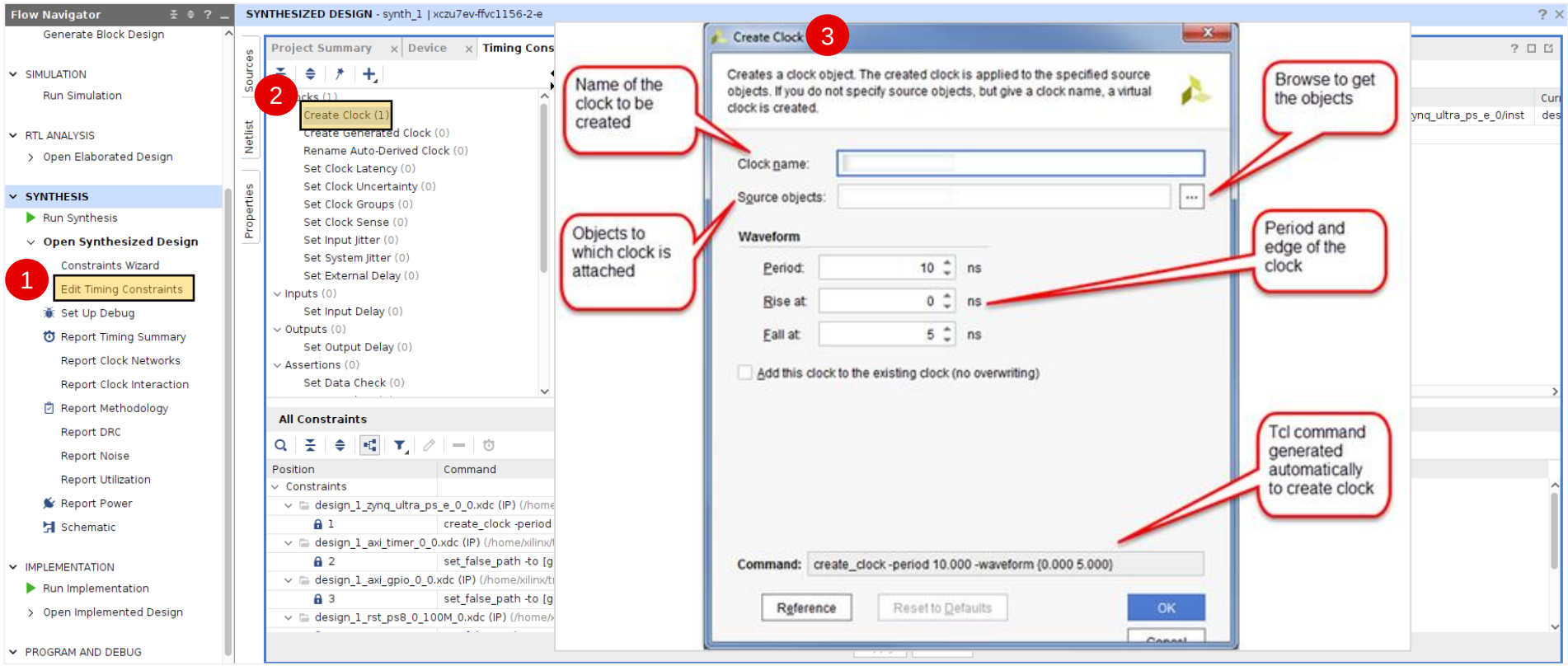
Two methods to create a clock in the XDC file

Using Tcl Commands

Using the GUI

Clocks can be created using the GUI as follows:

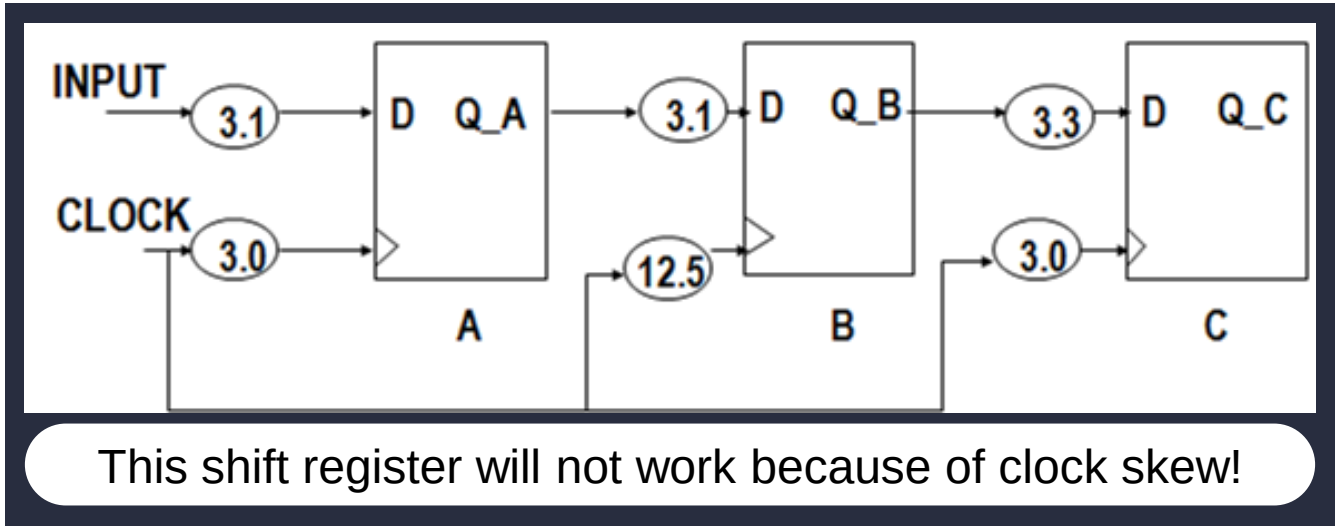
- **Step 1**: Open the Timing Constraints window
- **Step 2**: Double-click **Create Clock** to launch the wizard
- **Step 3**: Define the clock in the Create Clock Wizard

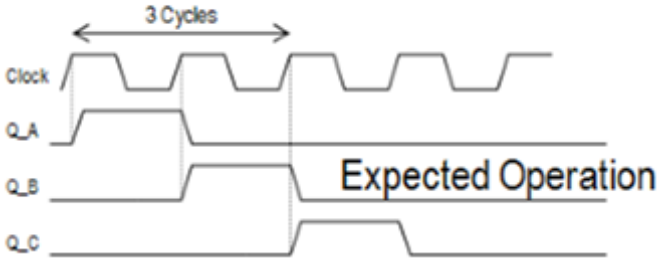


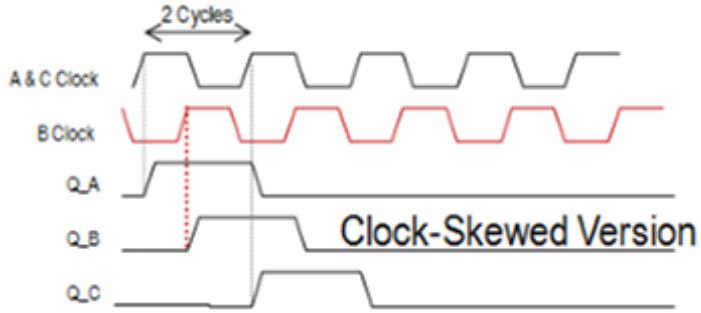
AMD XILINX

Clock Skew

Example







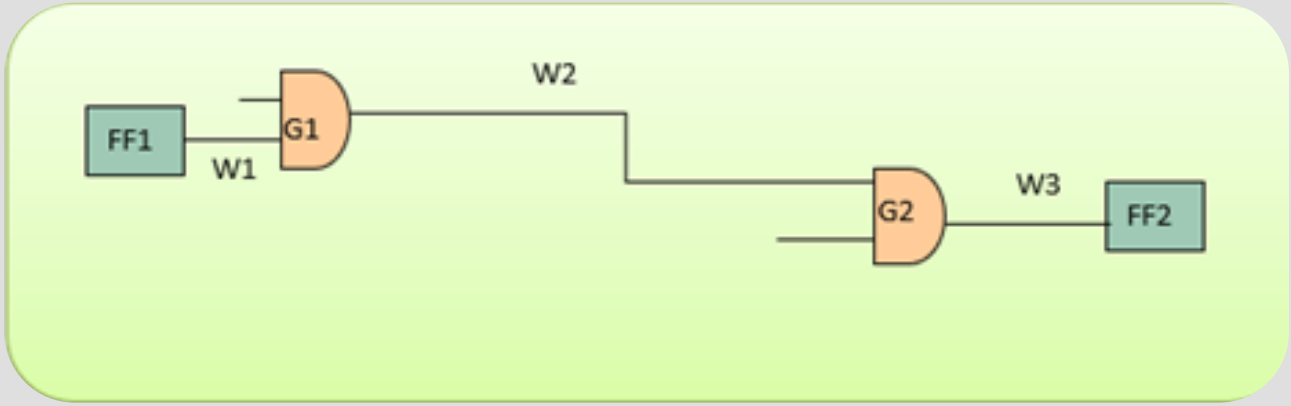
AMD
XILINX

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

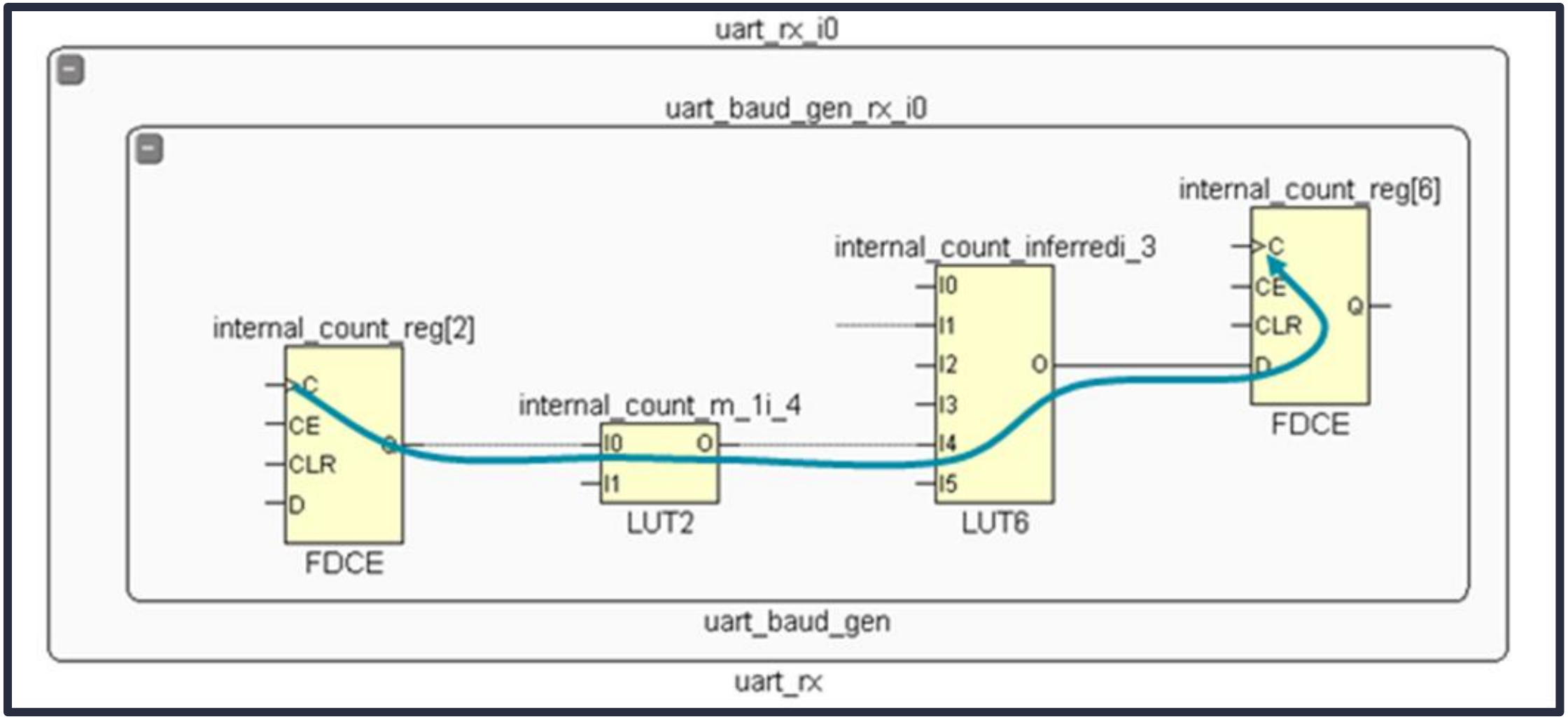
Checks every timing path (gates+wires)



Static Timing Paths

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

Static Timing Paths



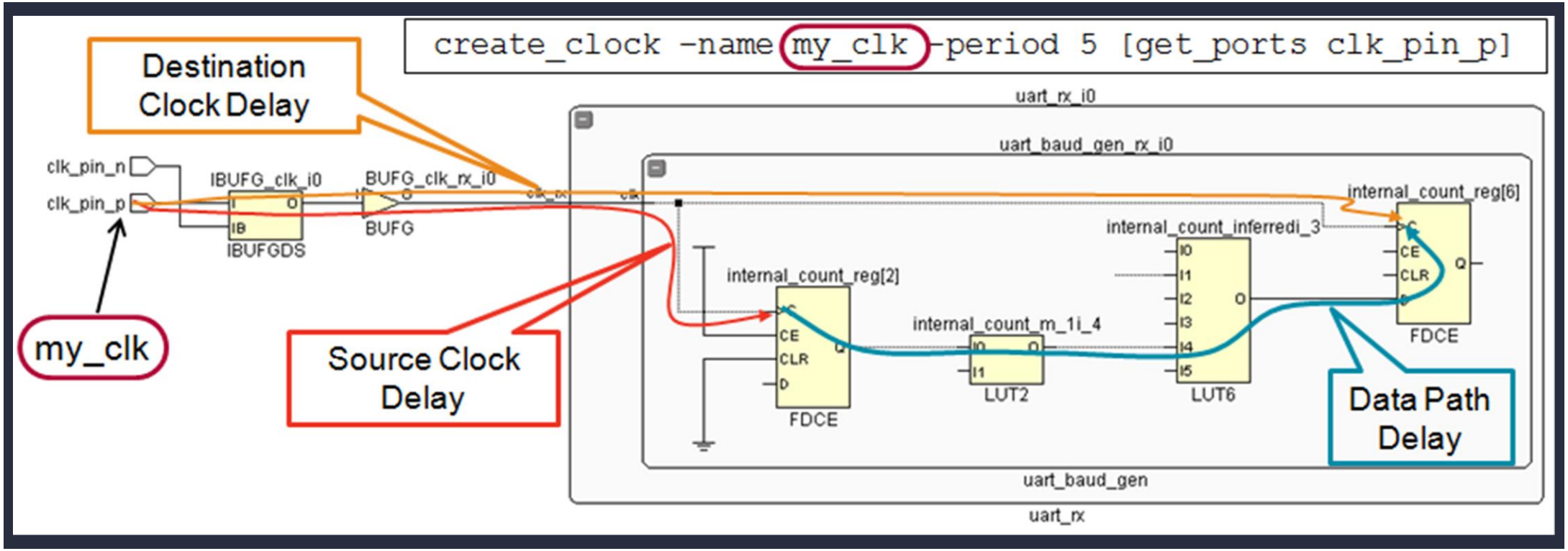
AMD
XILINX

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

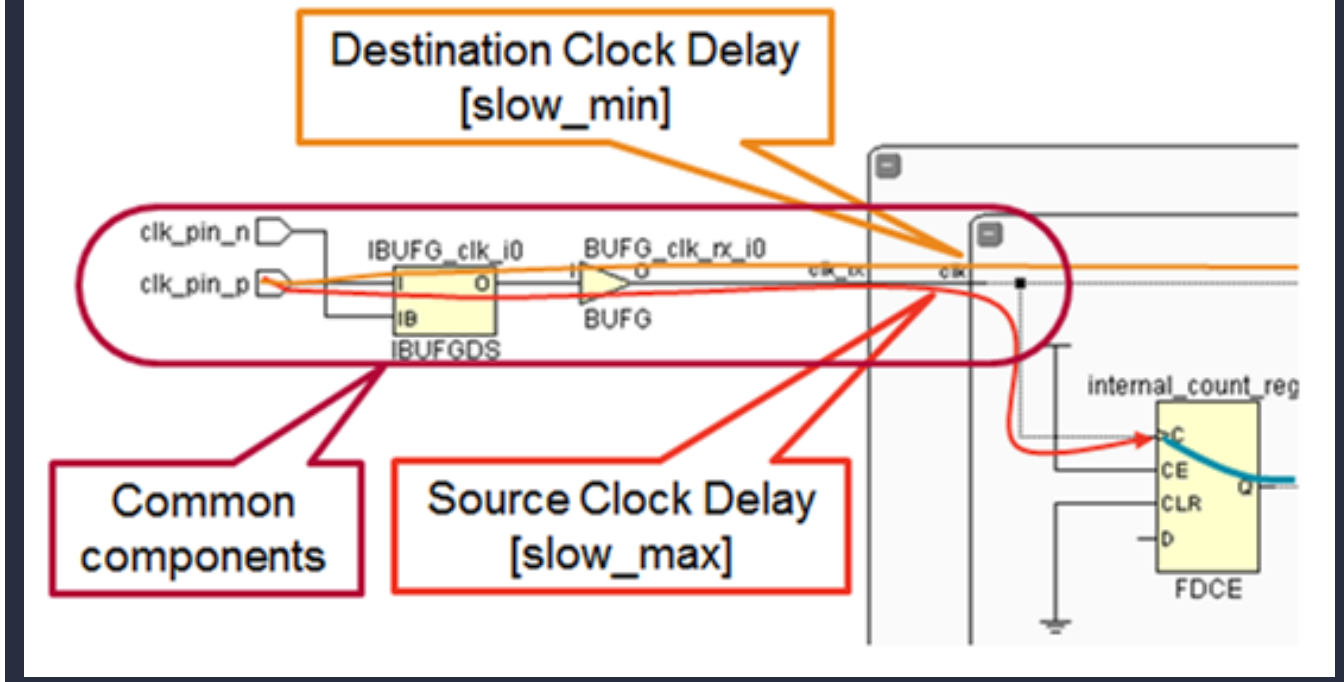
If slack is negative, this indicates a timing failure



AMD XILINX

Clock Pessimism

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



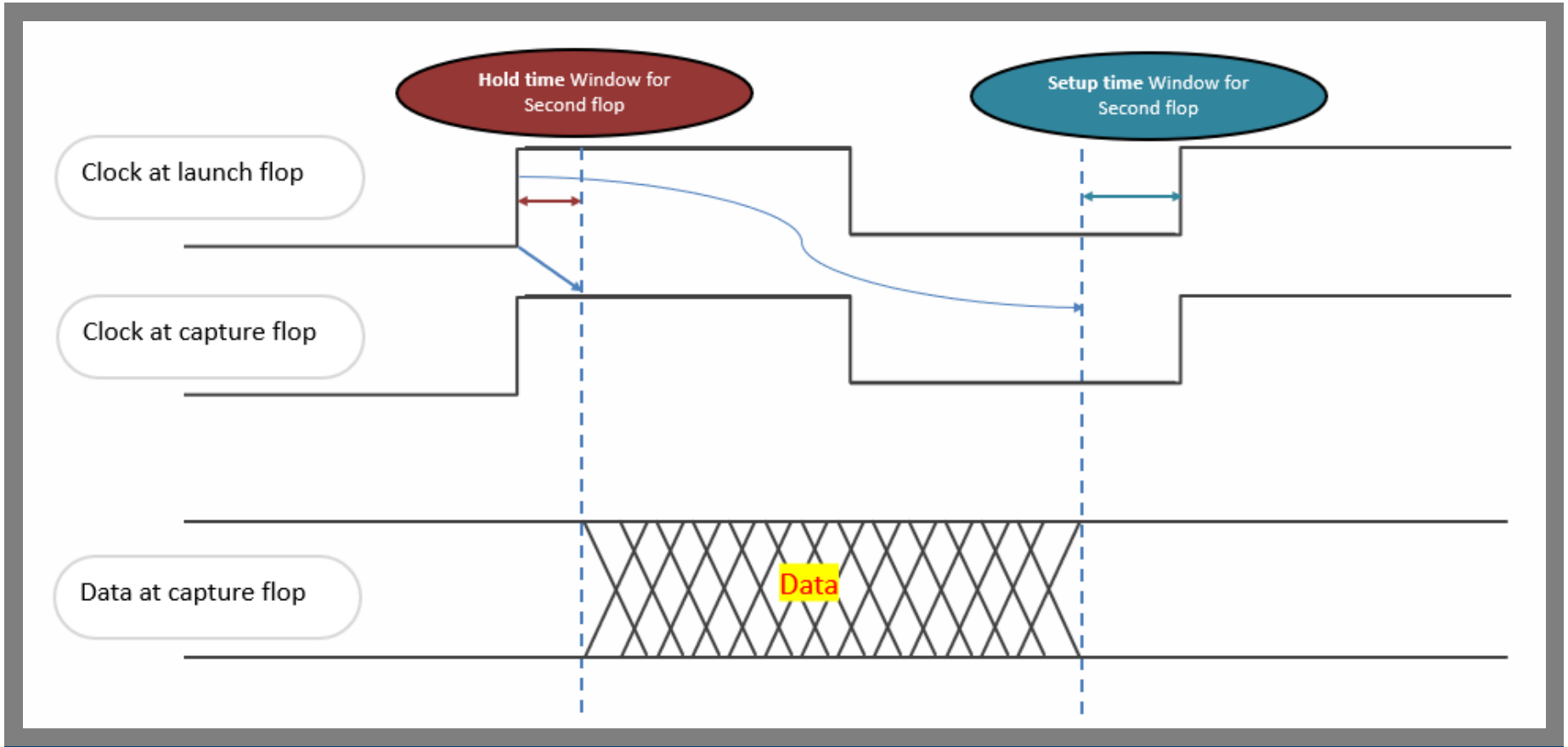
Source clock path is timed at `[slow_max]`

Destination clock path is timed at `[slow_min]`

Clock pessimism = common components (`slow_max - slow_min`)

AMD XILINX

Introduction to Setup and Hold Timing



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

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

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

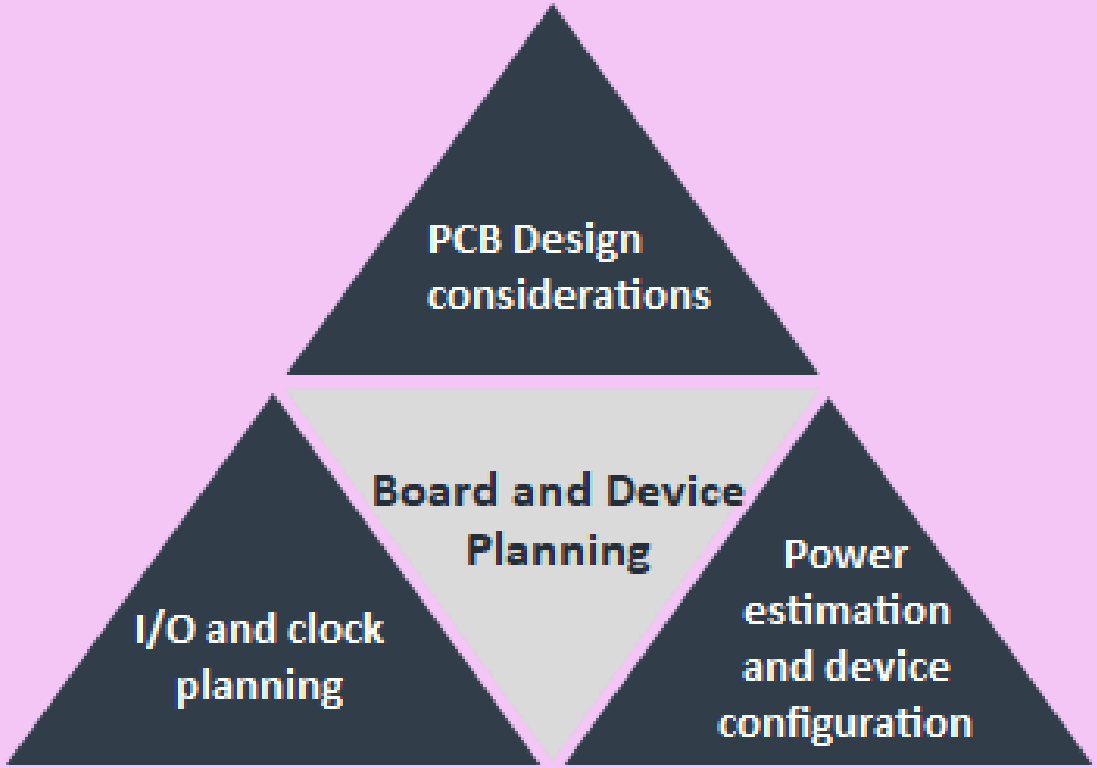
AMD
XILINX

Introduction to Vivado Reports

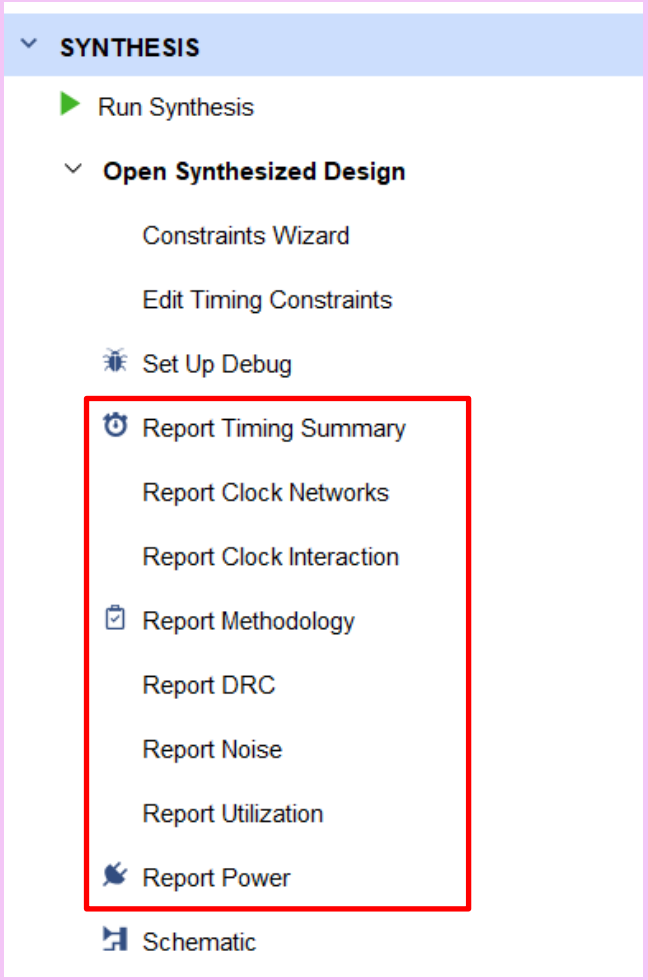
2022.2

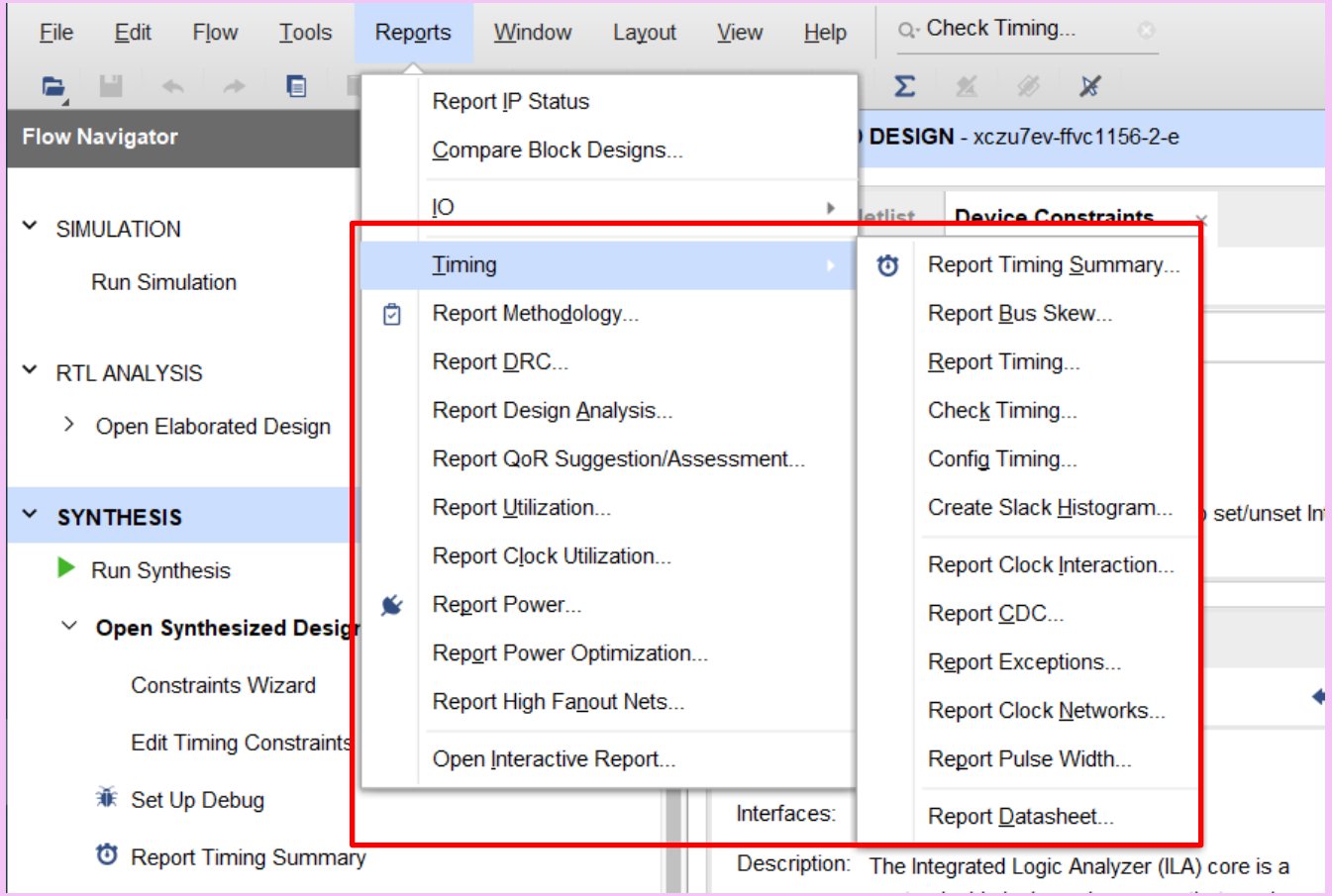
Board and Device Planning

Board and device planning includes:



Vivado Report

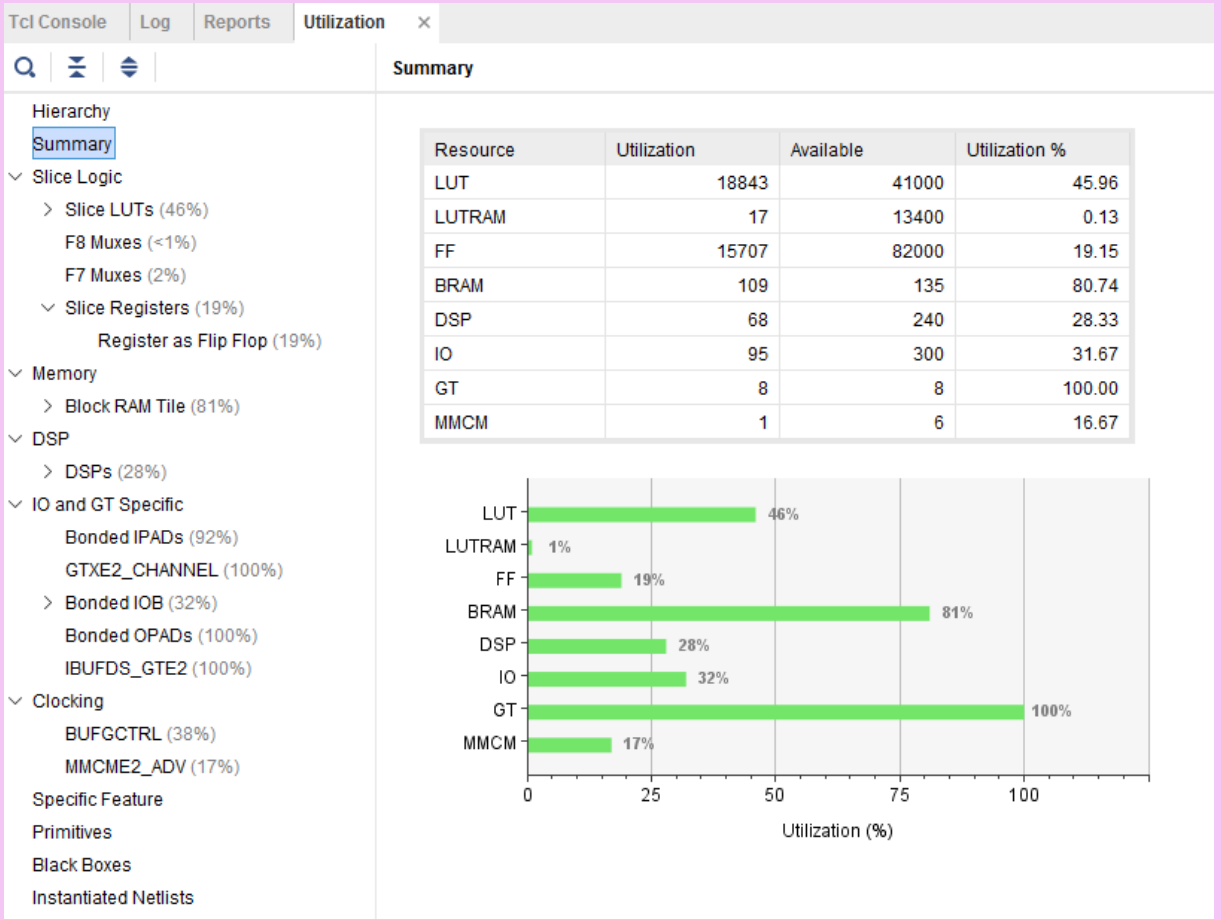




AMD XILINX

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

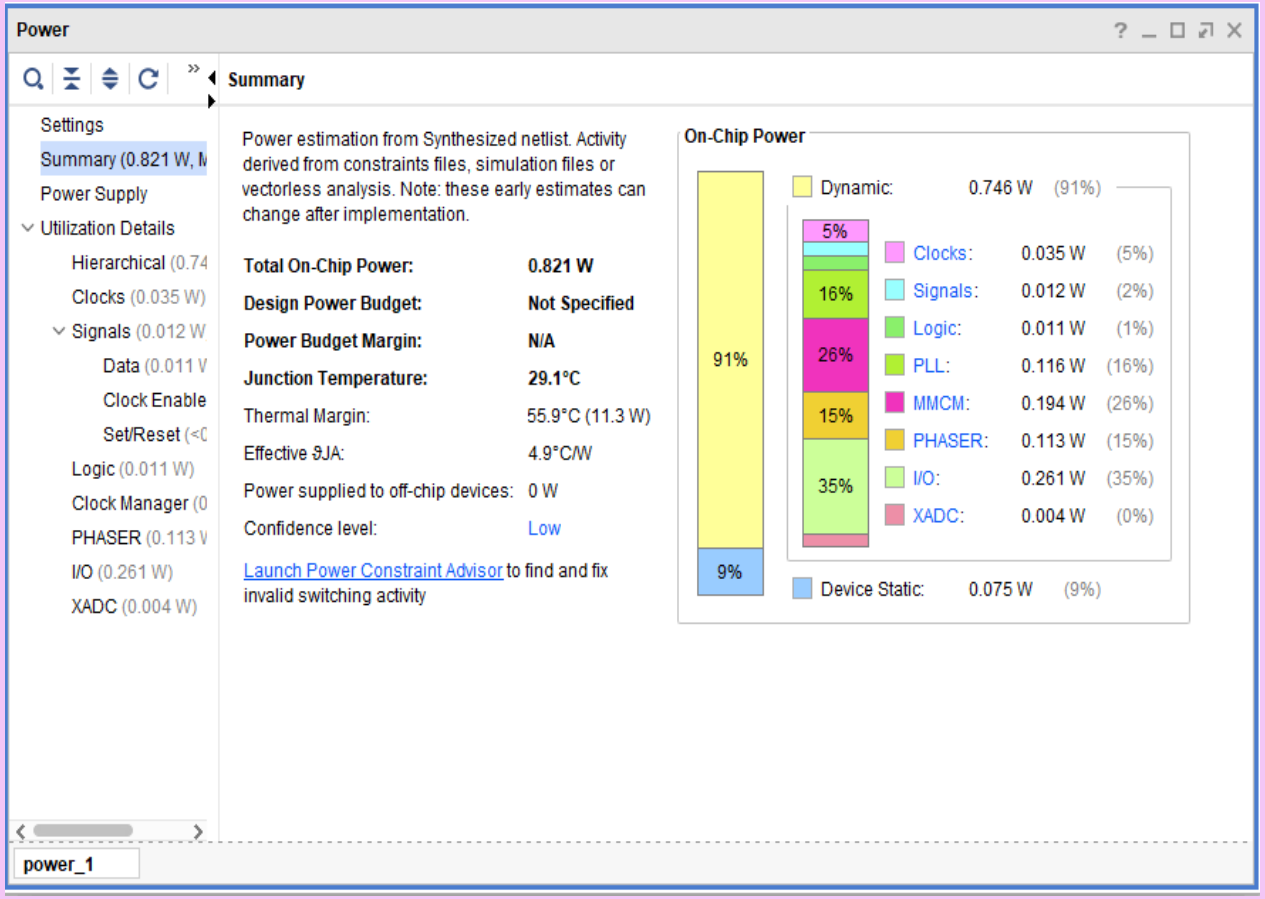


`report_utilization`

AMD XILINX

Report Power

`report_power` command runs power estimation and displays the report



`report_power`

AMD XILINX

Report DRC

`report_drc` command checks the design against the specified set of design rules

Reports any errors or violations that are found

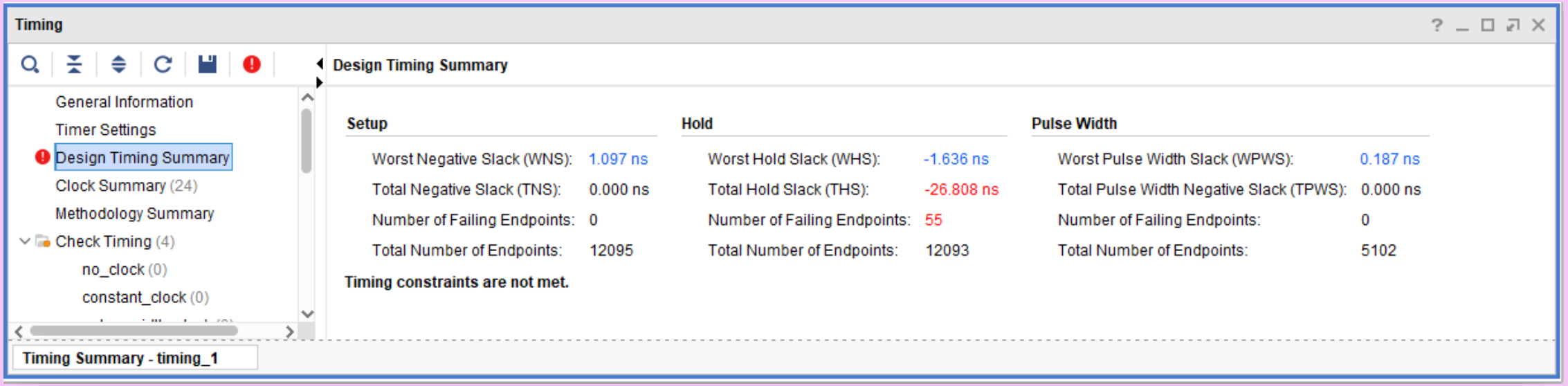
```
Report DRC

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 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>
```

`report_drc`

Report Timing Summary

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

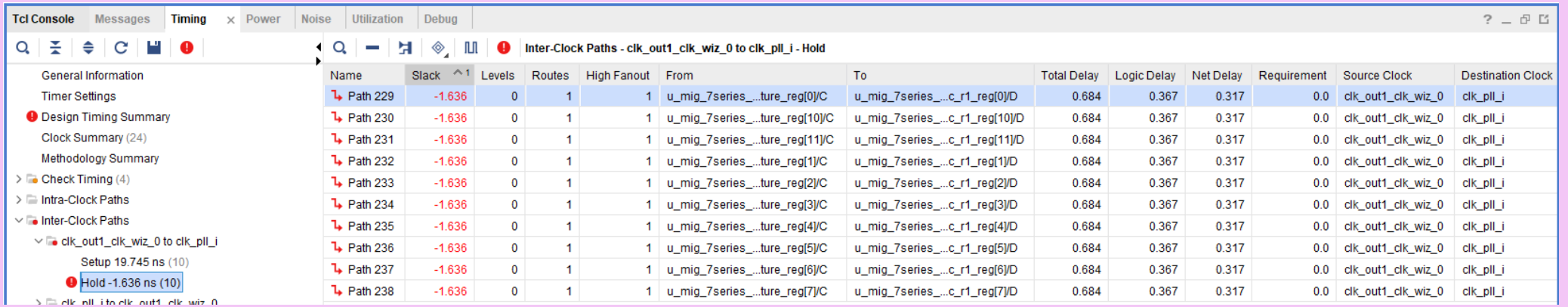
The report gives very detailed information about each static timing path.

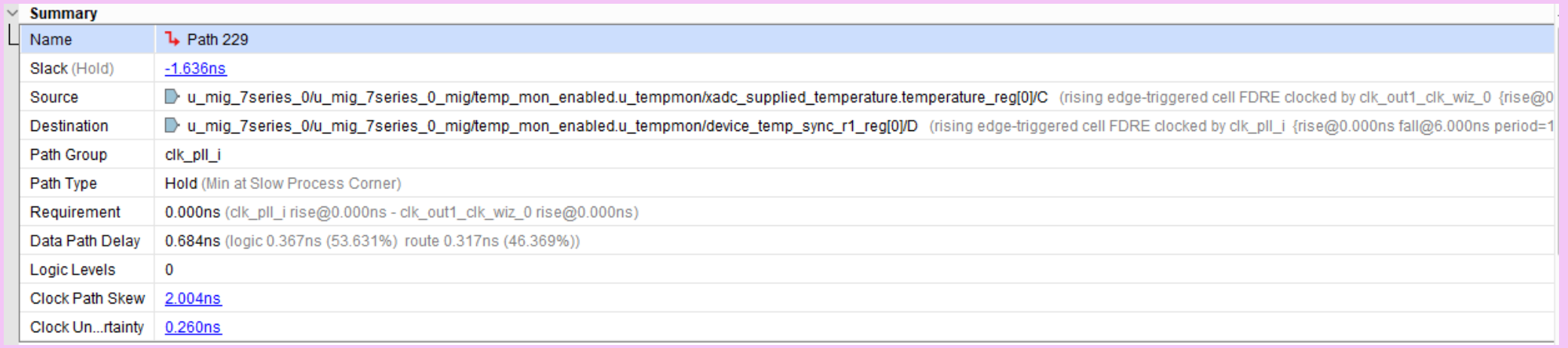


`report_timing_summary`

AMD XILINX

Report Timing Summary

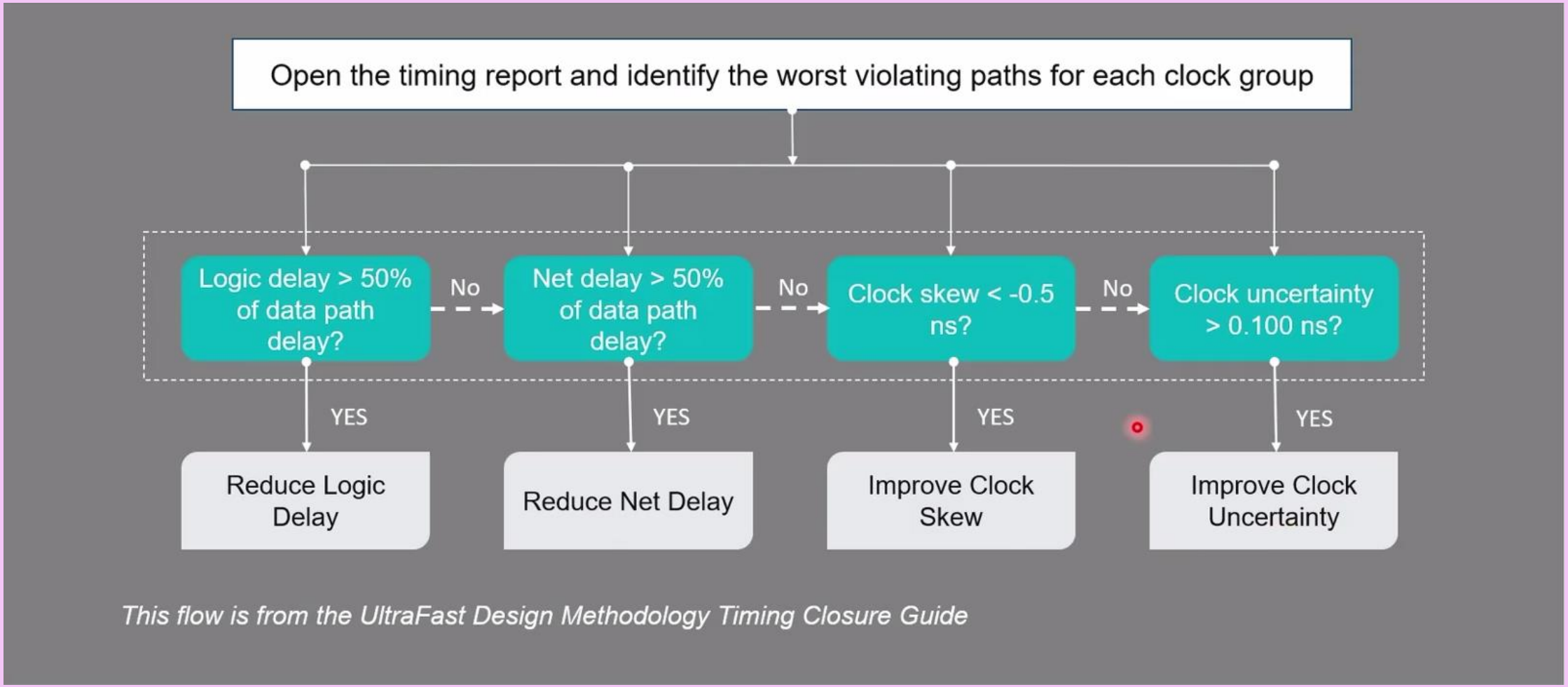





```
report_timing_summary
```


AMD XILINX

Report Timing Summary



Report Timing Summary

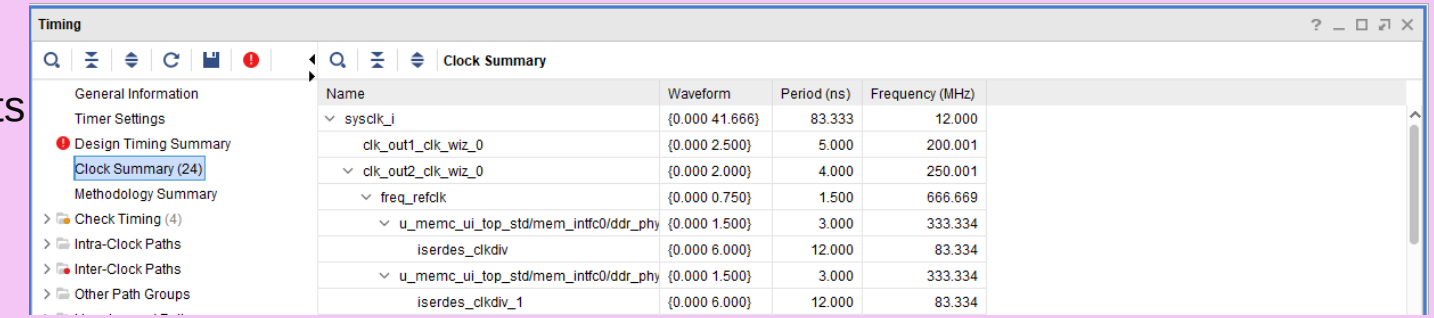
Check Timing

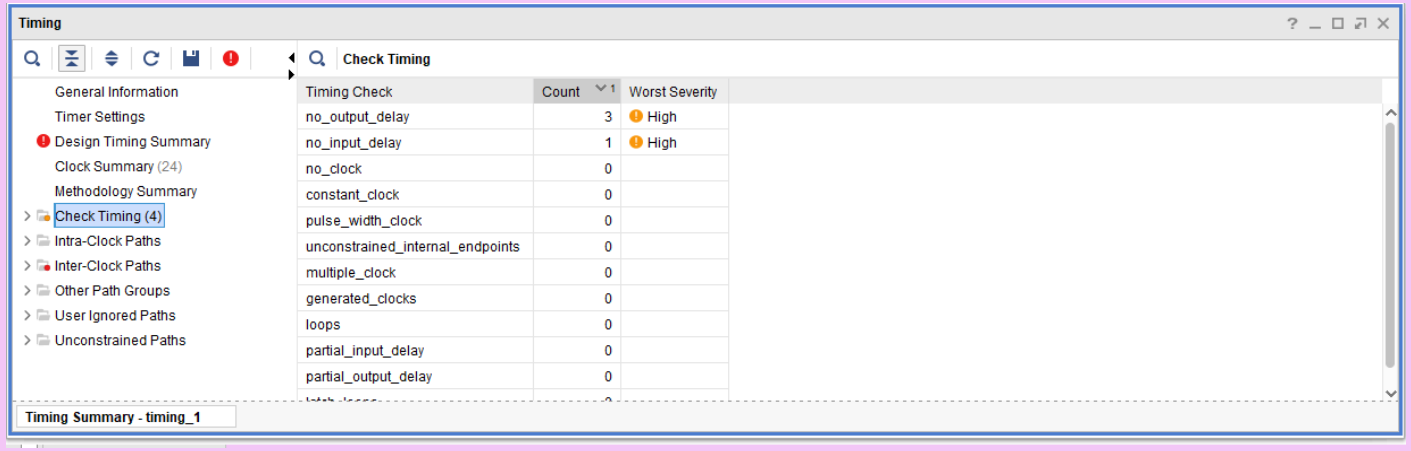
Check ports, pins and paths under timing constraints

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






```
report_timing_summary
```

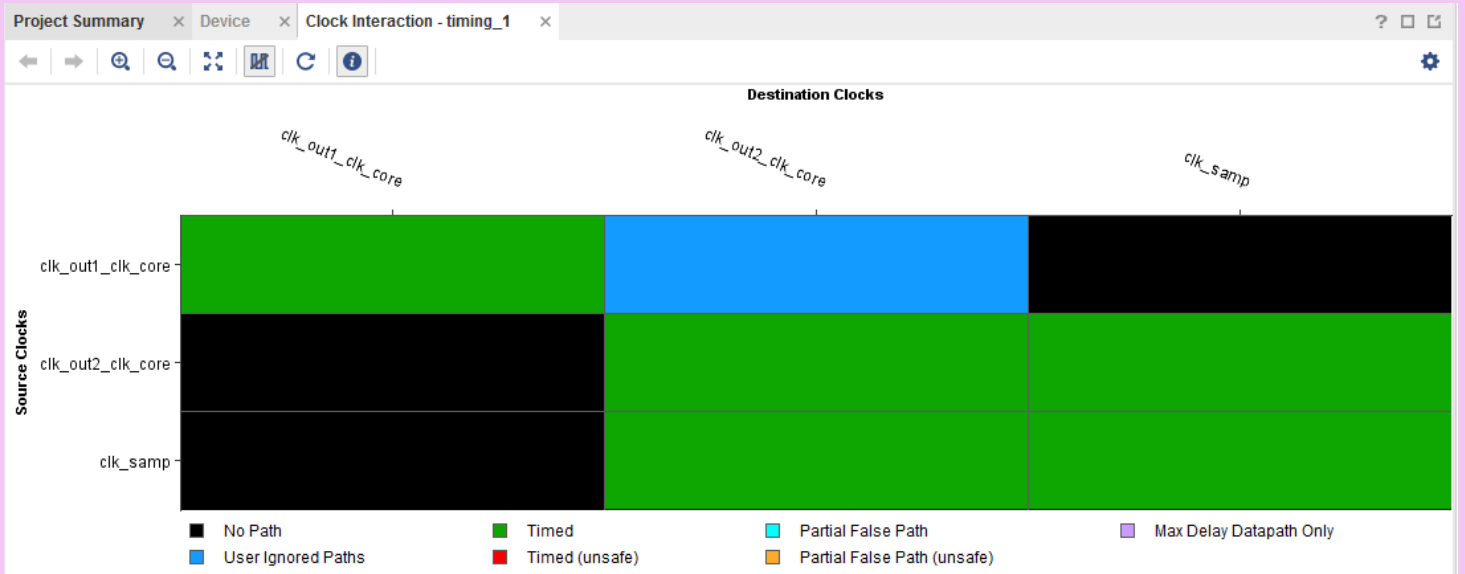

AMD XILINX

Report Clock Interaction

`report_clock_interaction` command

Analyzes timing paths that cross from

one clock domain into another clock domain



1	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)	Clo 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`

AMD XILINX

AMD
XILINX

Introduction to the Vivado Logic Analyzer

2022.2

Vivado Logic Analyzer Tool

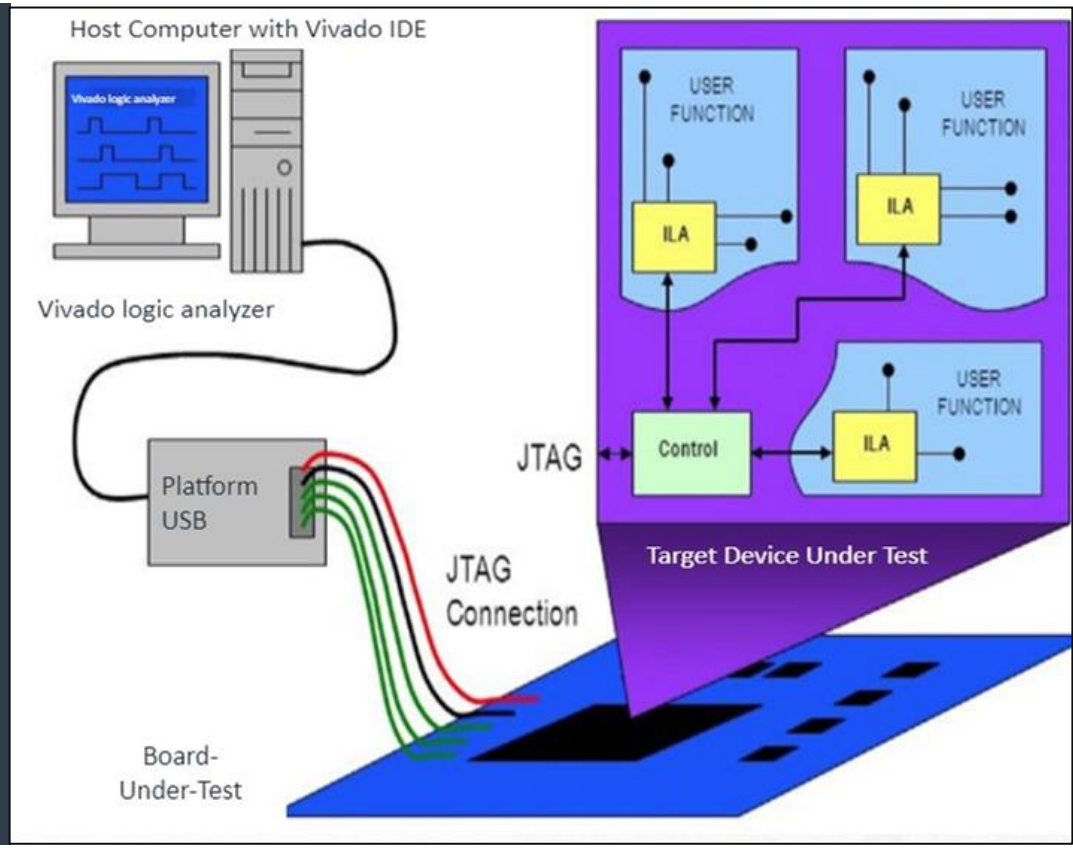
Powerful yet easy-to-use debug tool

Synergistic role with simulation

The Vivado logic analyzer can be used in three contexts:

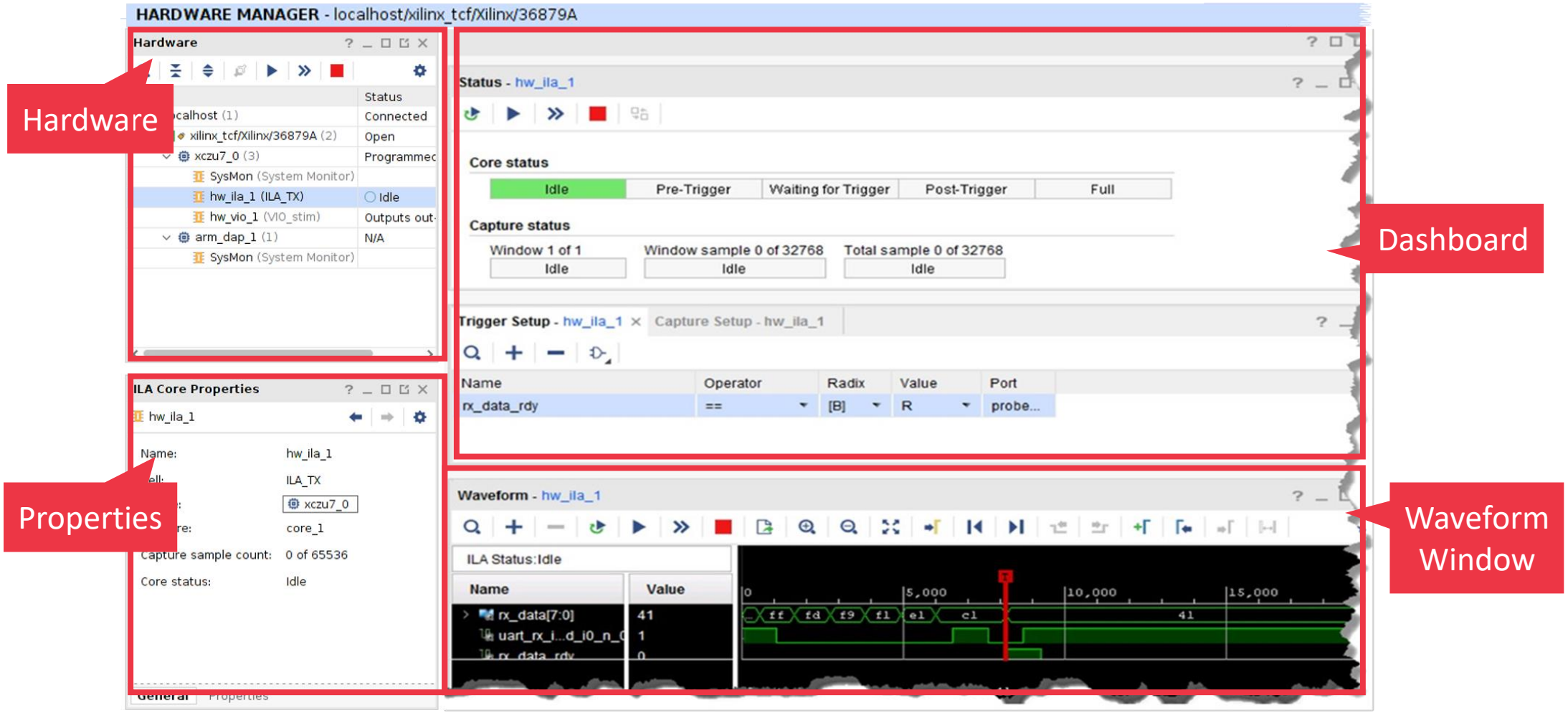
- Verification
- Debugging
- Data capture

Interact with debug IP cores



AMD
XILINX

Vivado Logic Analyzer Tool



Vivado Logic Analyzer Tool

AMD XILINX

VLA Comprised of…

Comprised of two basic items:

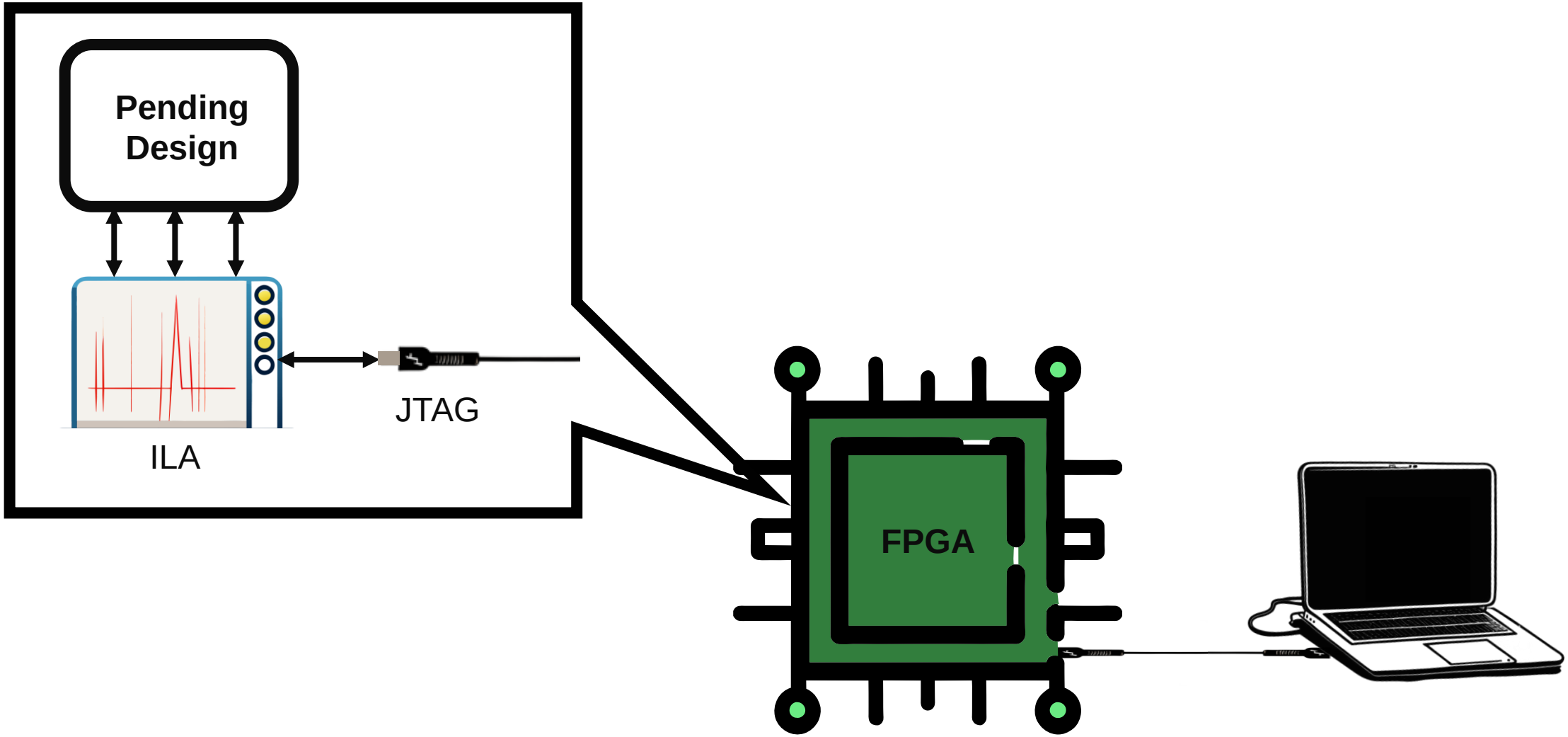
Debug cores:

- 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

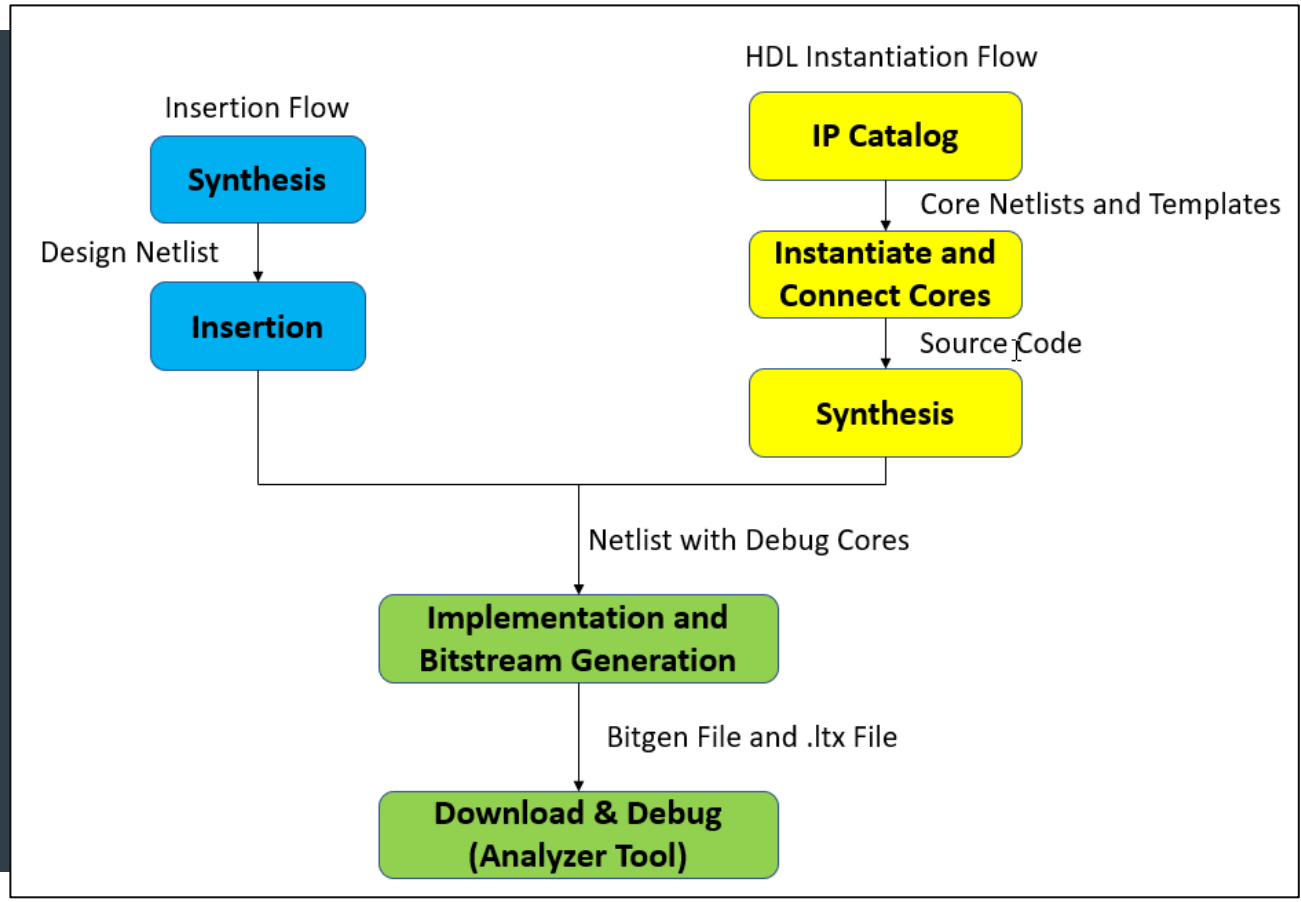


Vivado Logic Analyzer Probing Flows

Two Vivado logic analyzer probing flows:

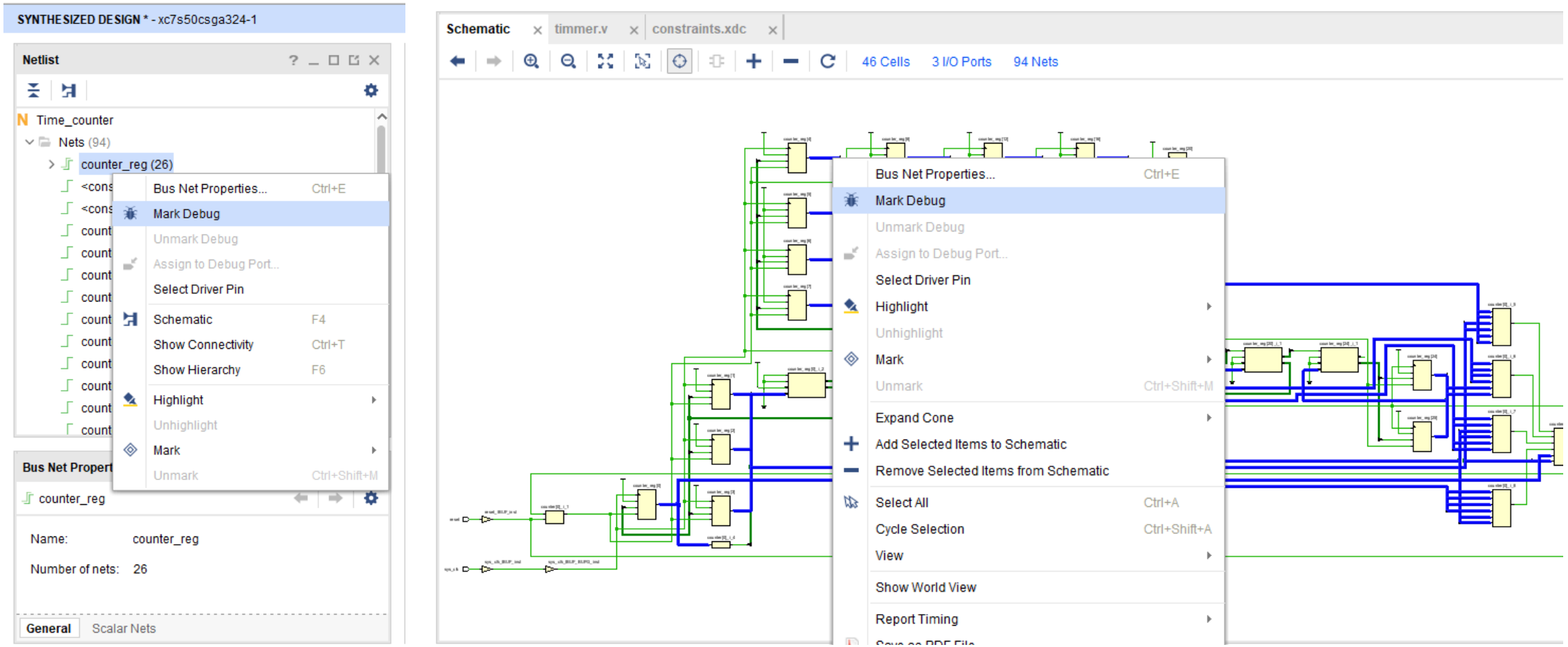
- HDL instantiation flow
- Netlist insertion flow

HDL instantiation flow and netlist insertion flows can be mixed



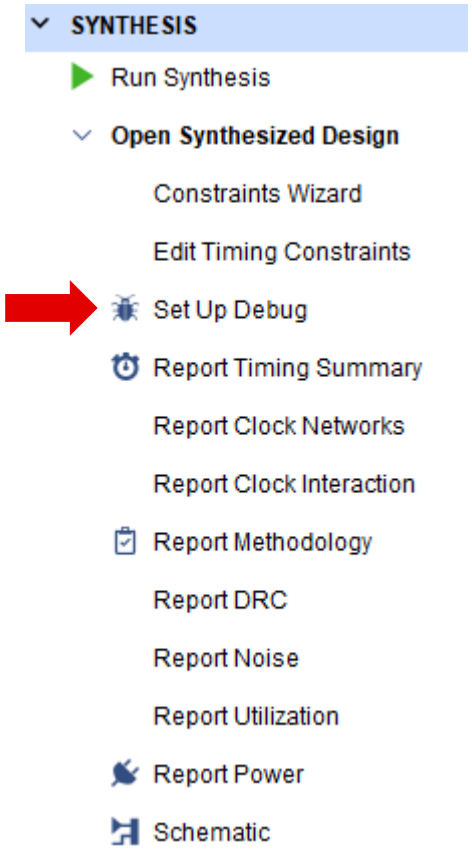
AMD XILINX

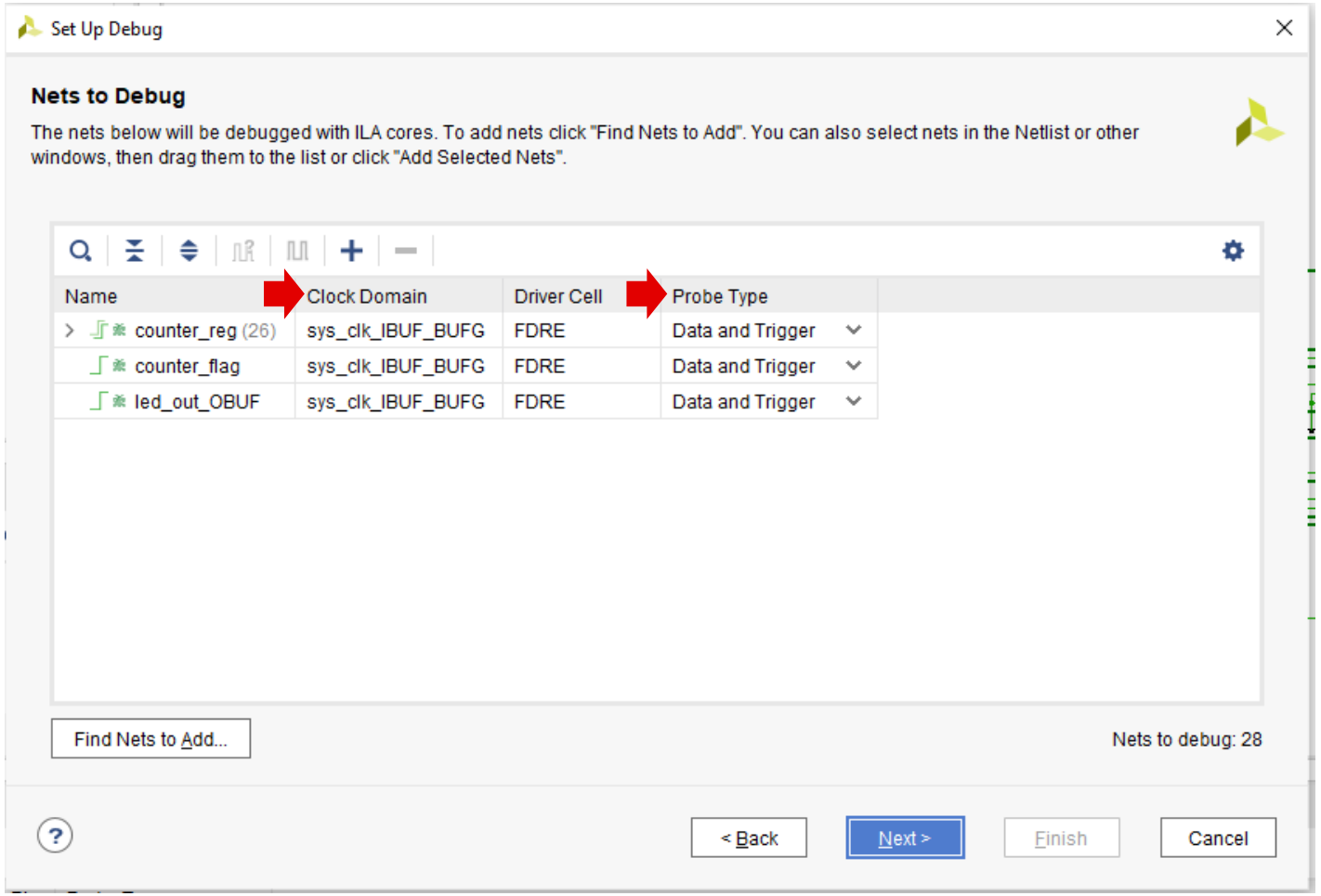
Add debug mark in the netlist



AMD XILINX

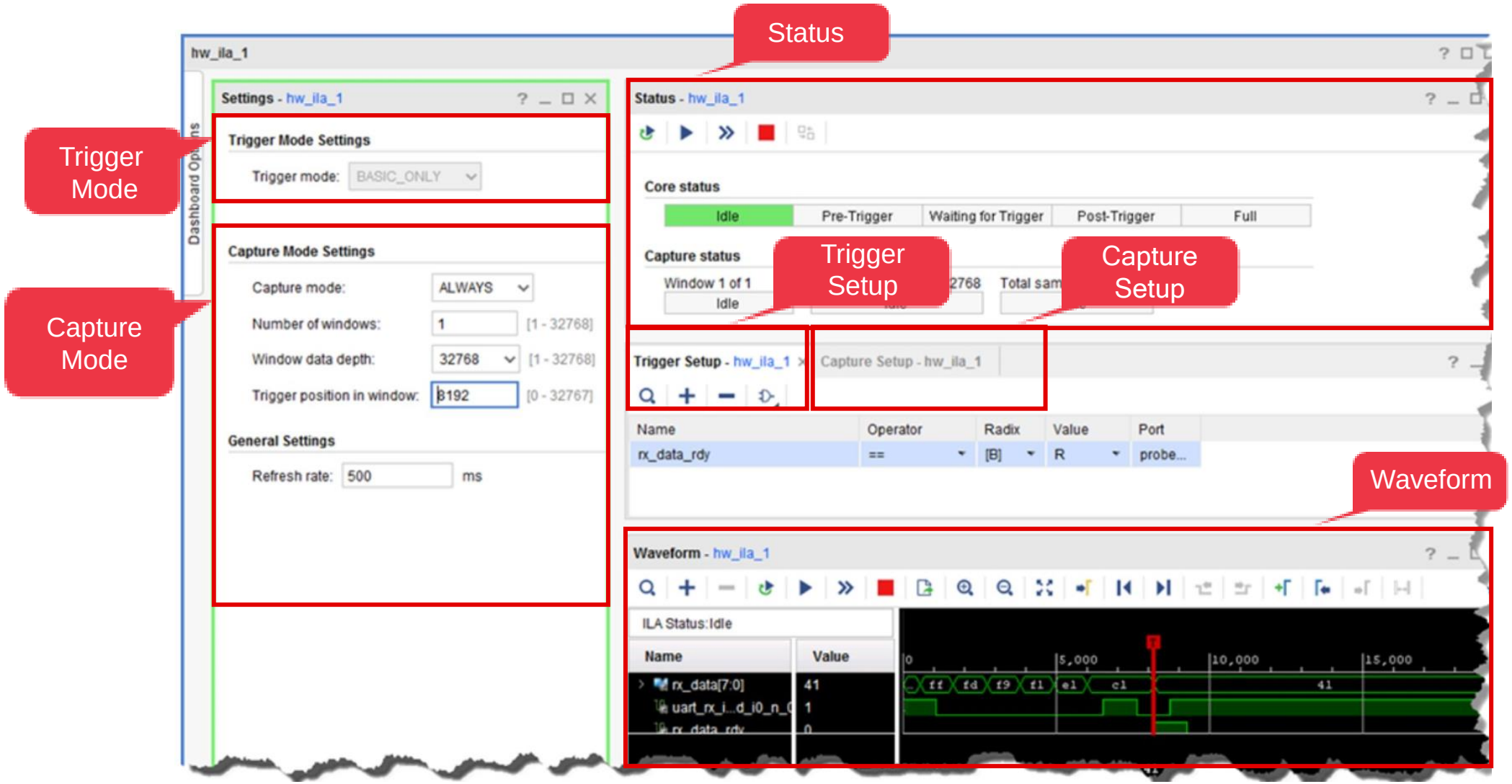
Add debug mark in the netlist





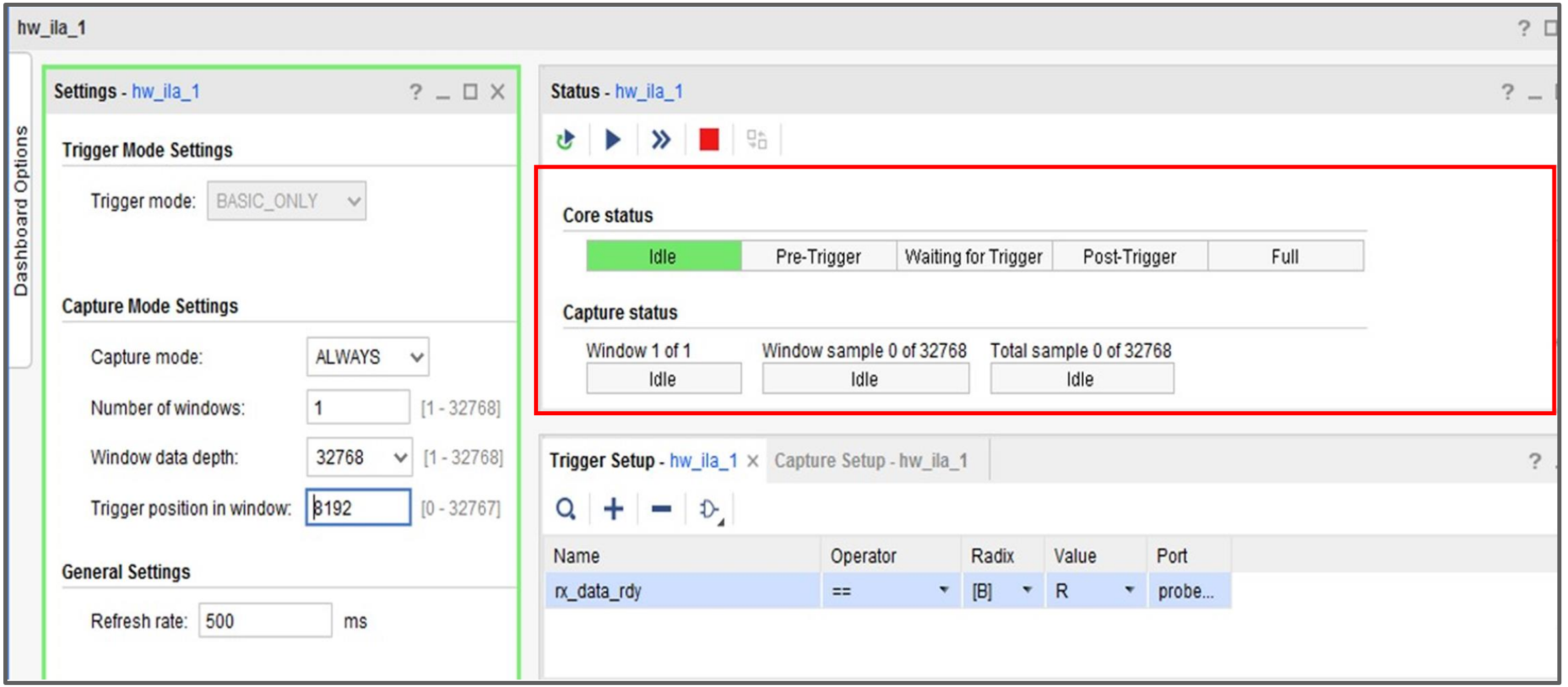
AMD XILINX

ILA Dashboard

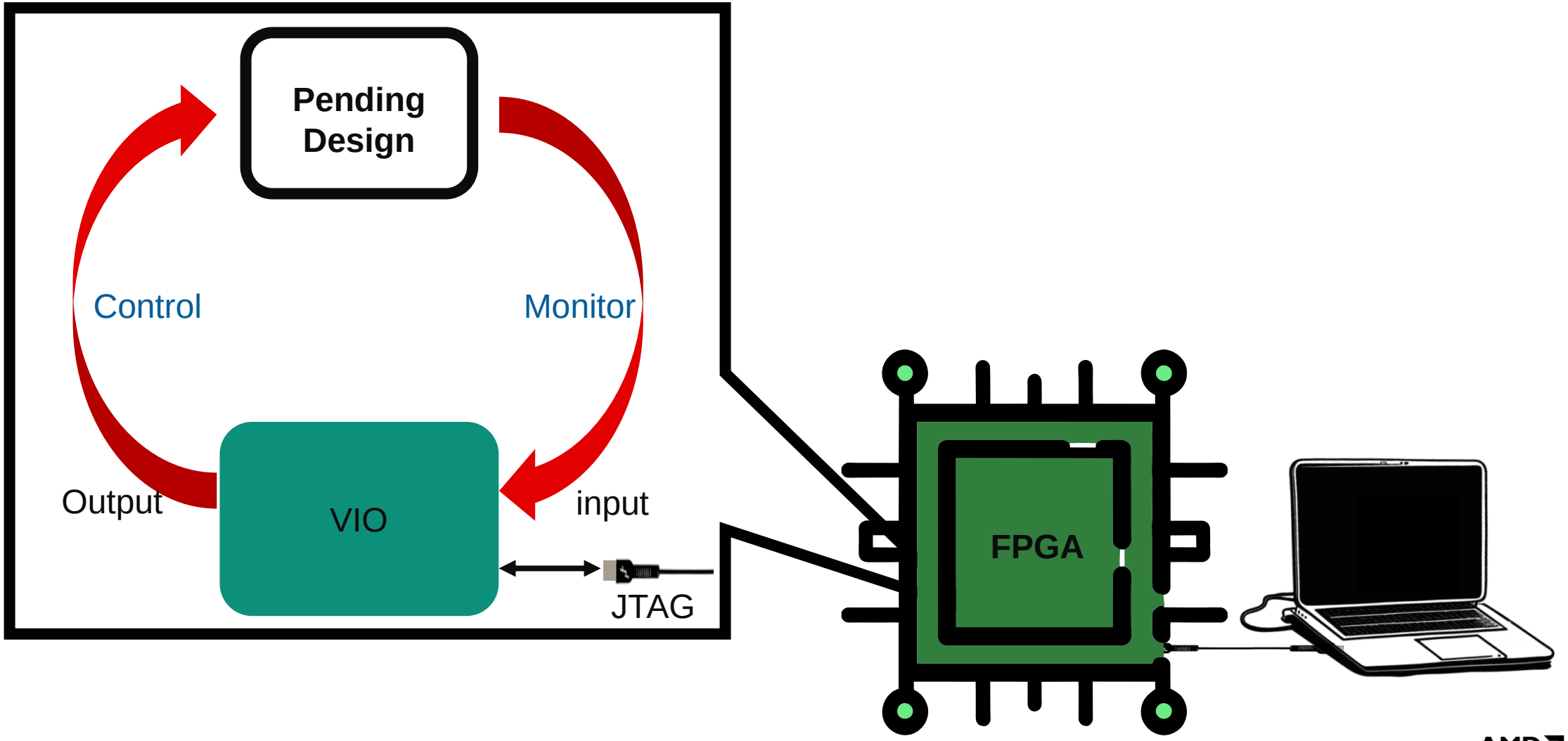


AMD XILINX

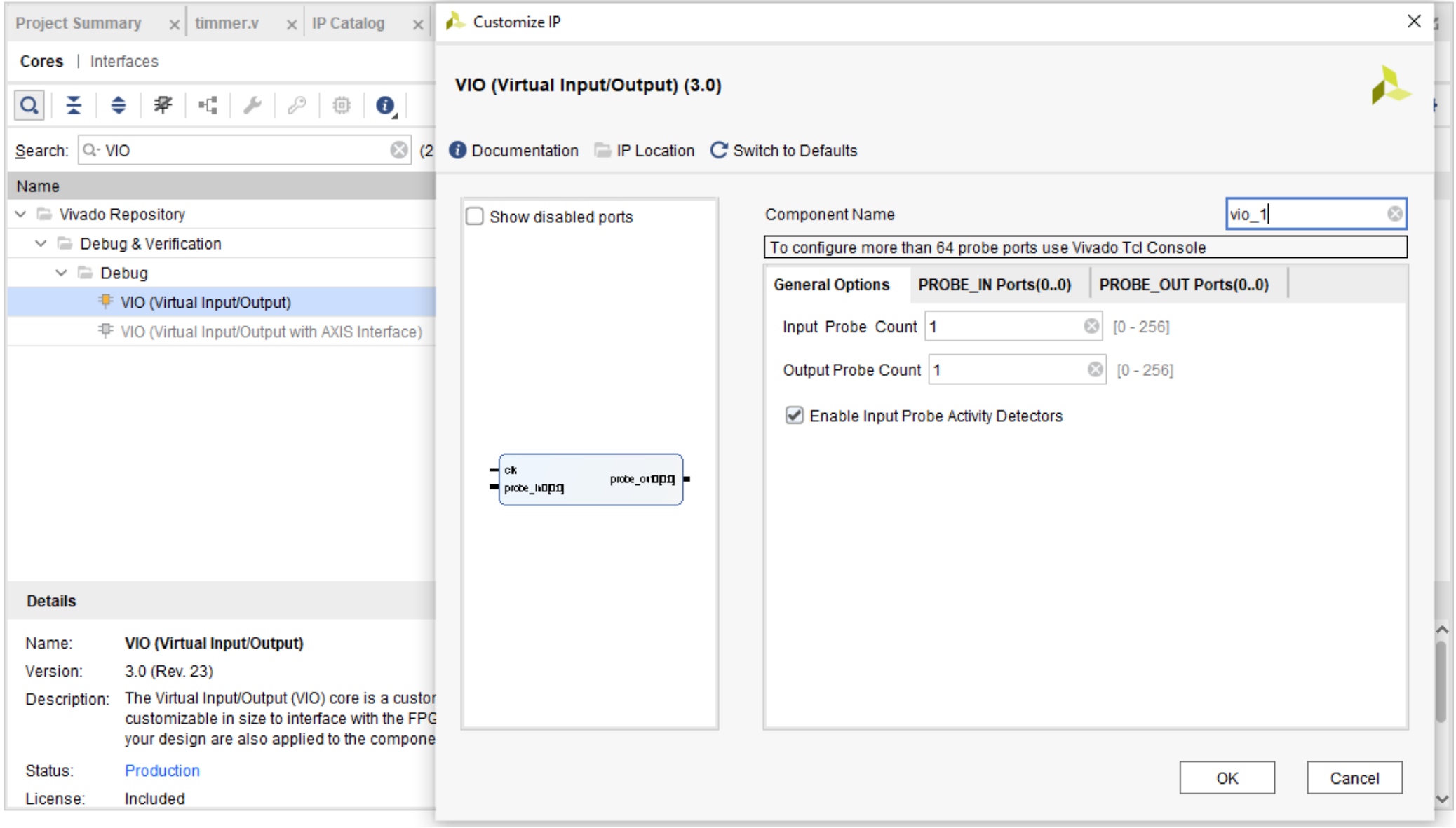
ILA Dashboard – Status



Virtual Input/Output



Add VIO IP Core



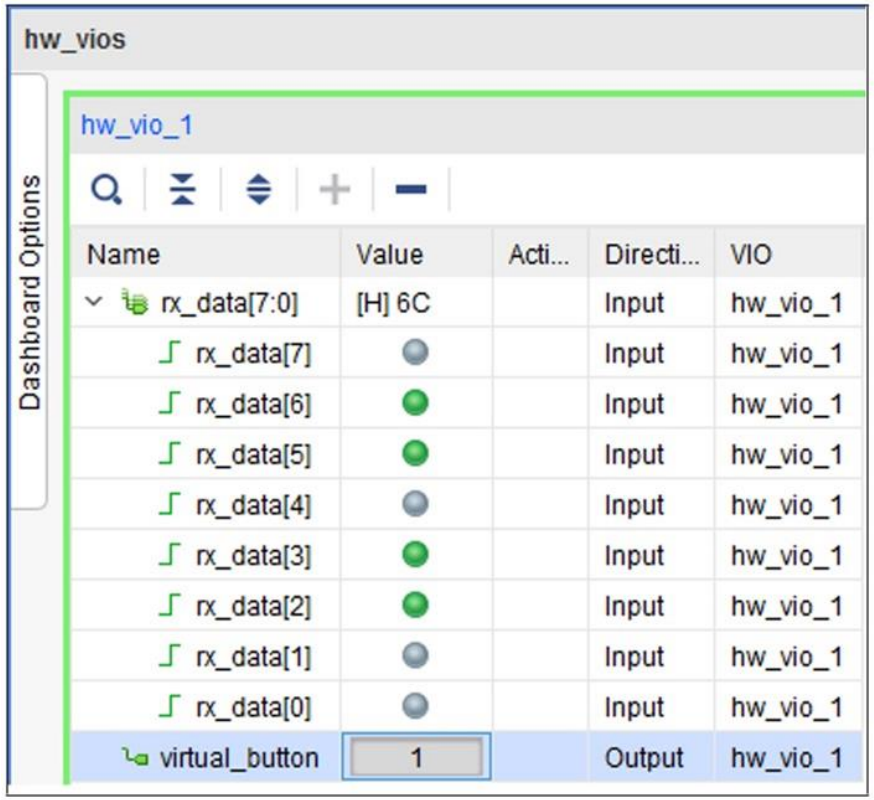
AMD XILINX

Virtual Input/Output (VIO)

VIO Ports and Parameters

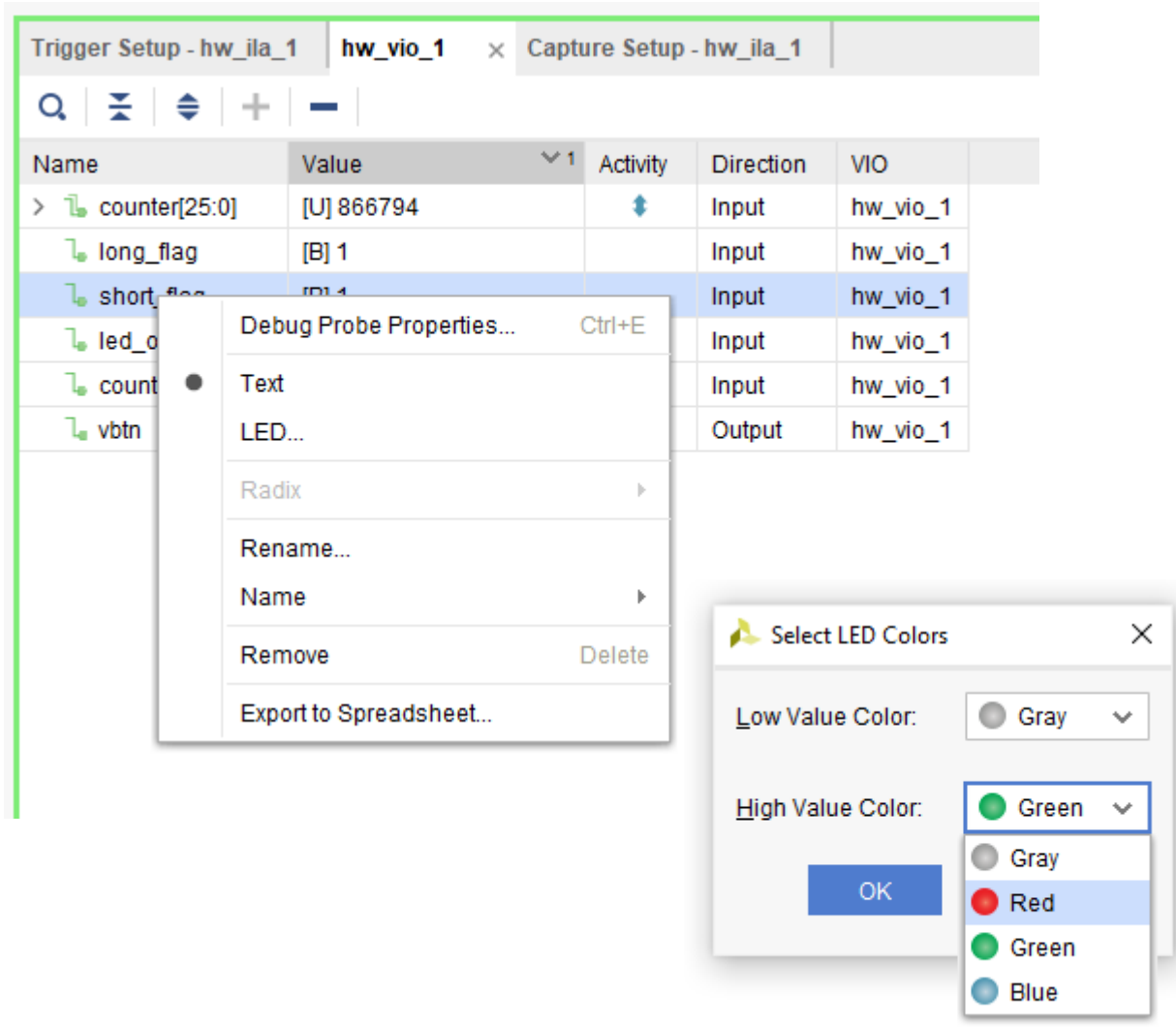
- Shows all status and control information pertaining to a given VIO core
- Drag-and-drop VIO probes from the Debug Probes window
- Monitor “virtual” pins in your design
- Inputs appear as LEDs or text
- Outputs appear as toggle buttons

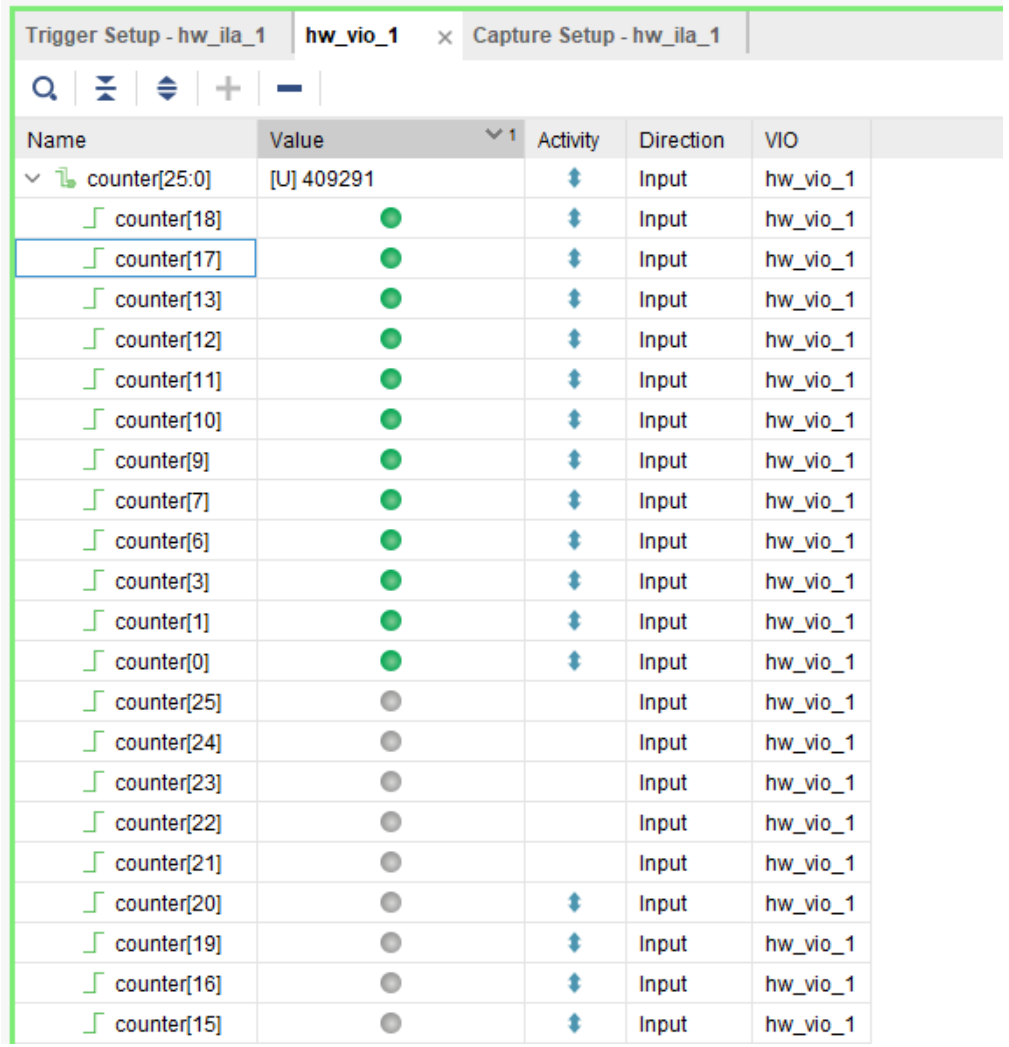
VIO Dashboard



AMD XILINX

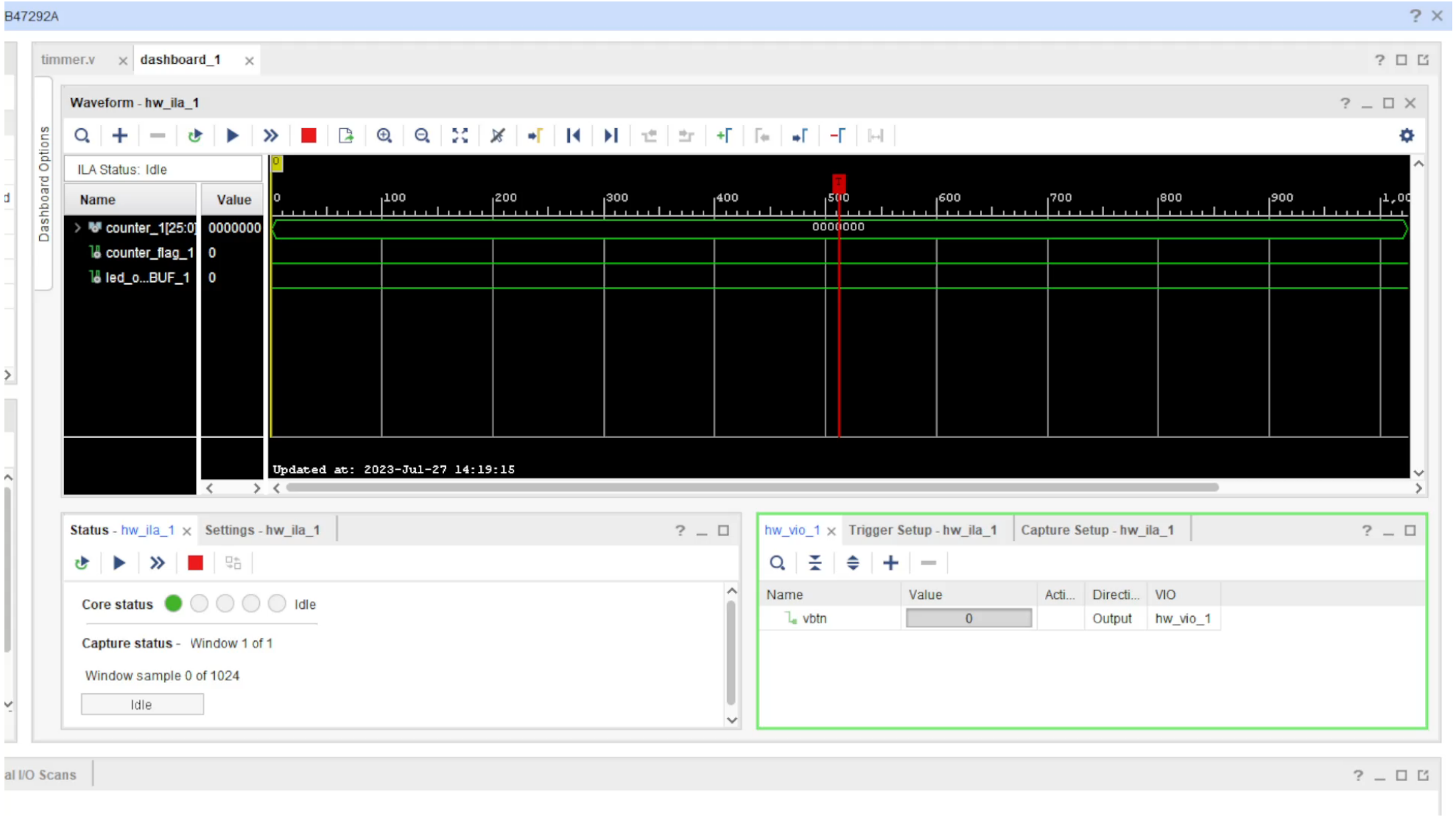
VIO Debugging





AMD XILINX

VIO Debugging



AMD XILINX

AMD
XILINX

Thank you very much for your attention!

Course Agenda
2023