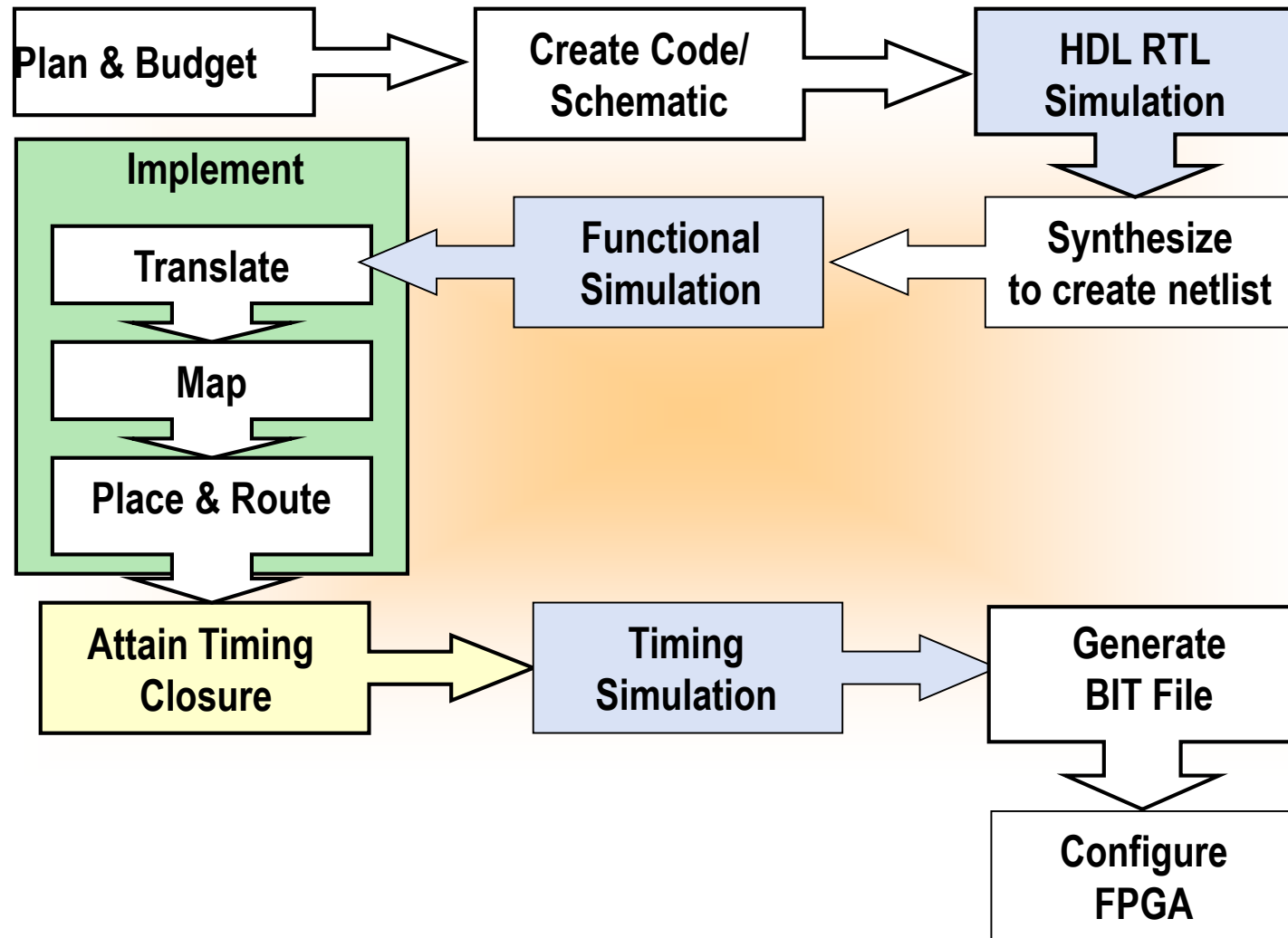




Vivado Design Flow

Artix-7
Vivado 2015.2 Version

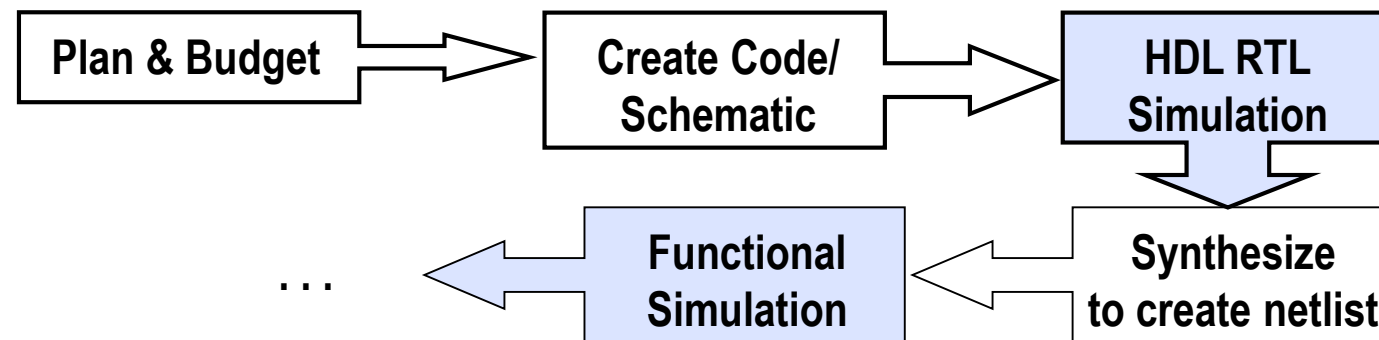
Xilinx Design Flow



Design Entry

Create designs in HDL or Schematic

- **Plan and budget**
- **Whichever method you use, you will need a tool to generate an EDIF or NGC netlist to bring into the Xilinx implementation tools**
 - Popular synthesis tools include: Synplify, Precision, FPGA Compiler II, and XST
- **Tools available to assist in design entry**
 - Architecture Wizard, CORE Generator™ system, and StateCAD tools
- **Simulate the design to ensure that it works as expected!**



Synthesis

Generate a netlist file

- **After coding up your HDL code, you will need a tool to generate a netlist (NGC or EDIF)**
 - Xilinx Synthesis Tool (XST) included
 - Support for Popular Third Party Synthesis tools: Synplify and Synplify Pro from Synplicity, and Precision from Mentor Graphics

Implementation

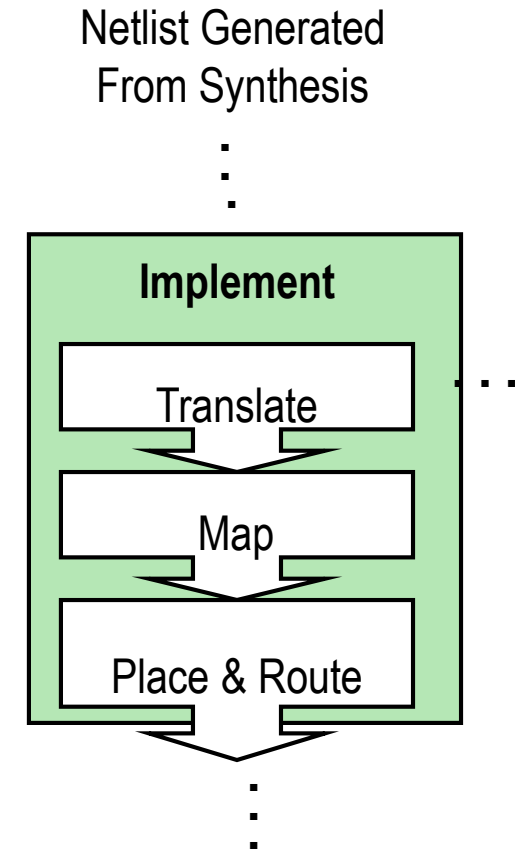
Process a netlist file

➤ Consists of three phases

- **Translate:** Merge multiple design files into a single netlist
- **Map:** Group logical symbols from the netlist (gates) into physical components (slices and IOBs)
- **Place & Route:** Place components onto the chip, connect the components, and extract timing data into reports

➤ Access Xilinx reports and tools at each phase

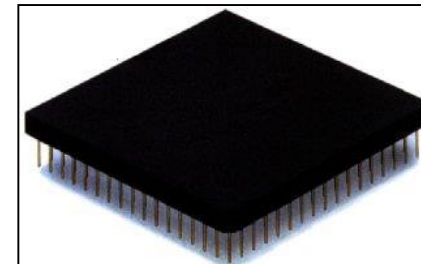
- Timing Analyzer, Floorplanner, FPGA Editor, XPower



Configuration

Testing and Verification

- **Once a design is implemented, you must create a file that the FPGA can understand**
 - This file is called a bitstream: a BIT file (.bit extension)
- **The BIT file can be downloaded**
 - Directly into the FPGA
 - Use a download cable such as Platform USB
 - To external memory device such as a Xilinx Platform Flash PROM
 - Must first be converted into a PROM file



Outline

- ***Vivado IDE Features and Benefits***
- **Vivado Design Suite Introduction**
- **Vivado Design Flow**
- **Summary**

Vivado IDE Solution

➤ Interactive design and analysis

- Timing analysis, connectivity, resource utilization, timing constraint analysis, and entry

➤ RTL development and analysis

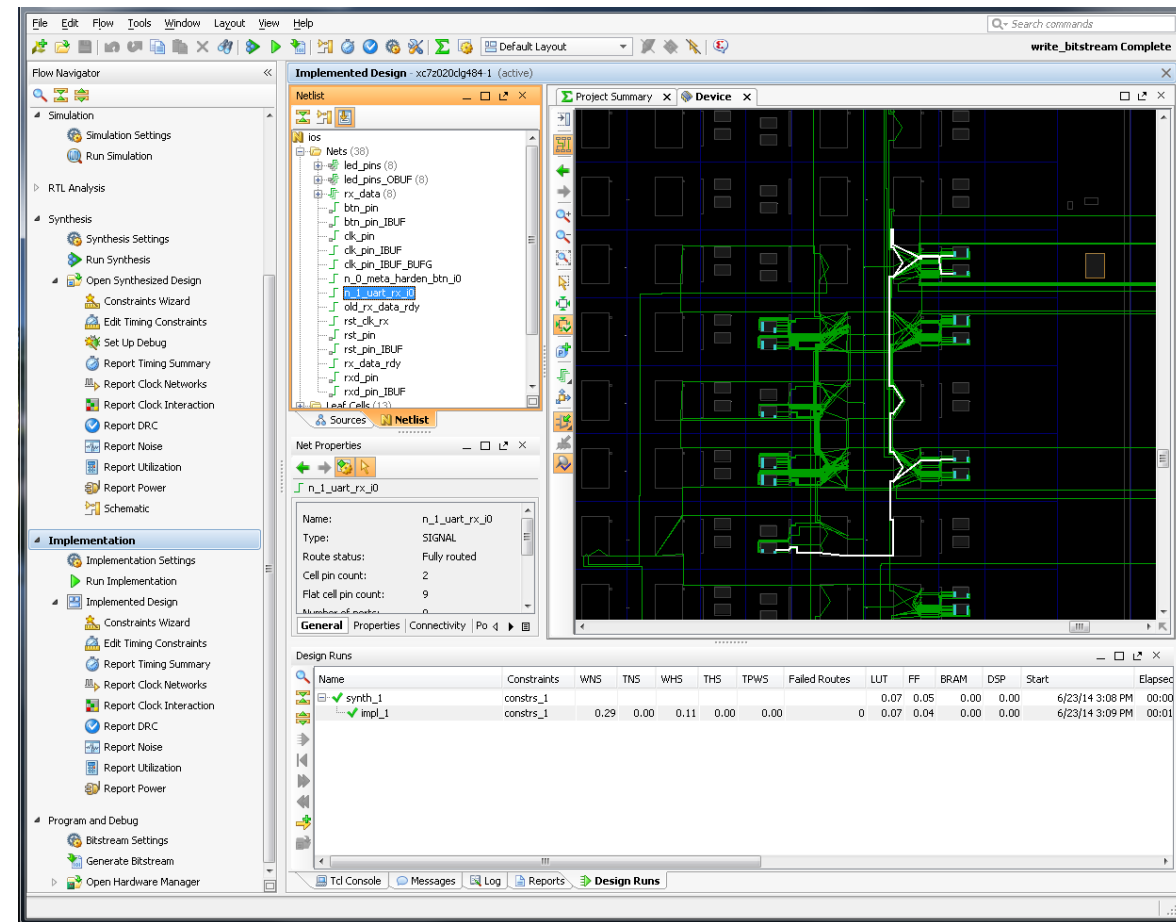
- Elaboration of HDL
- Hierarchical exploration
- Schematic generation

➤ XSIM simulator integration

➤ Synthesis and implementation in one package

➤ I/O pin planning

- Interactive rule-based I/O assignment



Who Should Use Vivado?

➤ Designers needing an interactive design approach

- Analysis and area constraints to drive place & route

➤ Challenging designs

- Large devices, complex constraints, and high device utilization
- Advantages are also seen with small devices

➤ Designs experiencing implementation issues

- Performance, capacity, run time, and repeatability

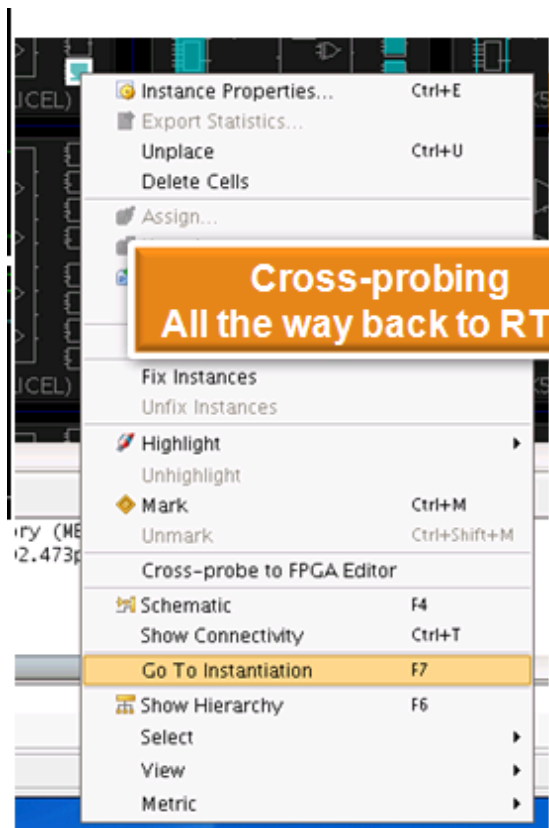
➤ Designs requiring implementation control

- Users looking for options other than just a pushbutton flow
- Visualize design issues from many aspects
- Block-based design constraints

➤ Designs targeting the 7-Series (or newer) devices

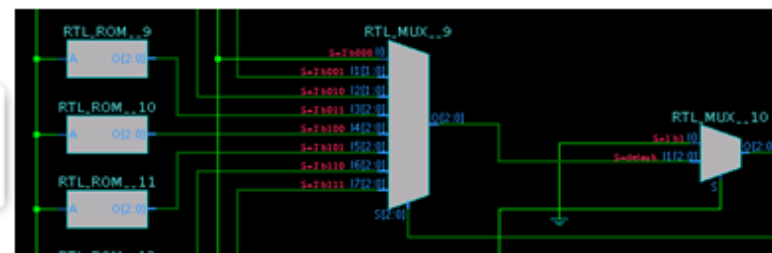
Vivado's Visualization Feature

- Visualize and debug your design at any flow stage
 - Cross-probing between netlist/schematic/RTL

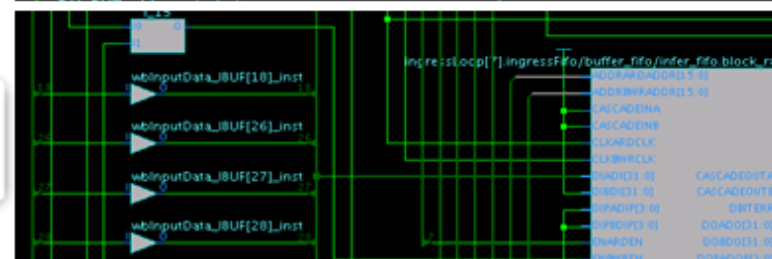


**Cross-probing
All the way back to RTL!!**

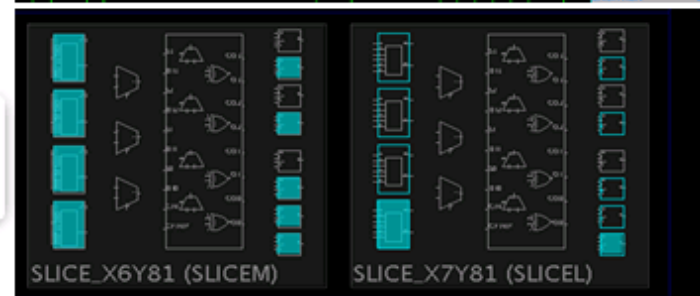
**Elaborated
RTL**



**Netlist
Schematic**



**Implemented
Design**



Gain Faster Timing Closure

➤ Analyze multiple implementation results

- Highlight failing timing paths from post-route timing analysis
- Quickly identify and constrain critical path logic

➤ Hierarchical floorplanning

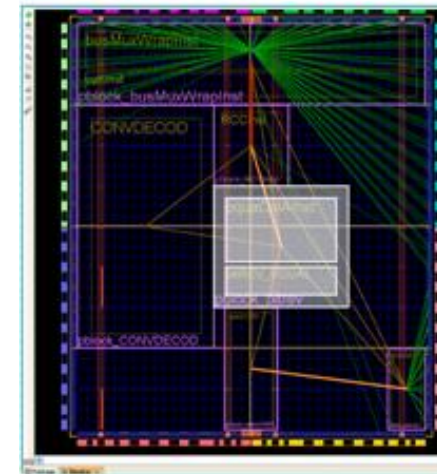
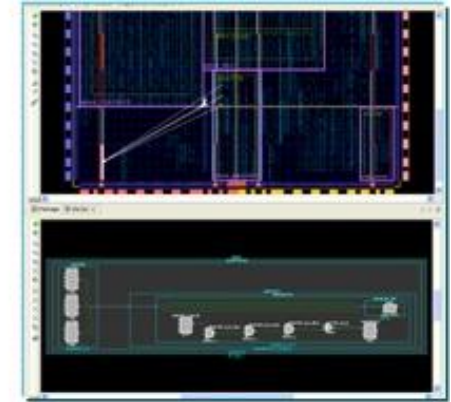
- Guide place & route toward better results

➤ Utilization estimates

- All resource types shown for each Pblock
- Clocks or carry chains

➤ Connectivity display

- I/Os, net bundles, clock domains



Tcl Capability

- **Tcl Console enables the designer to actively query the design netlist**
- **Full Tcl scripting support in two design flows**
 - Project-based design flow provides easy project management by the Vivado IDE
 - Non-project batch design flow enables entire flow to be executed in memory
- **Journal and log files can be used for script construction**

Project Based vs Non-Project Batch Flows

➤ Vivado tools support two flows

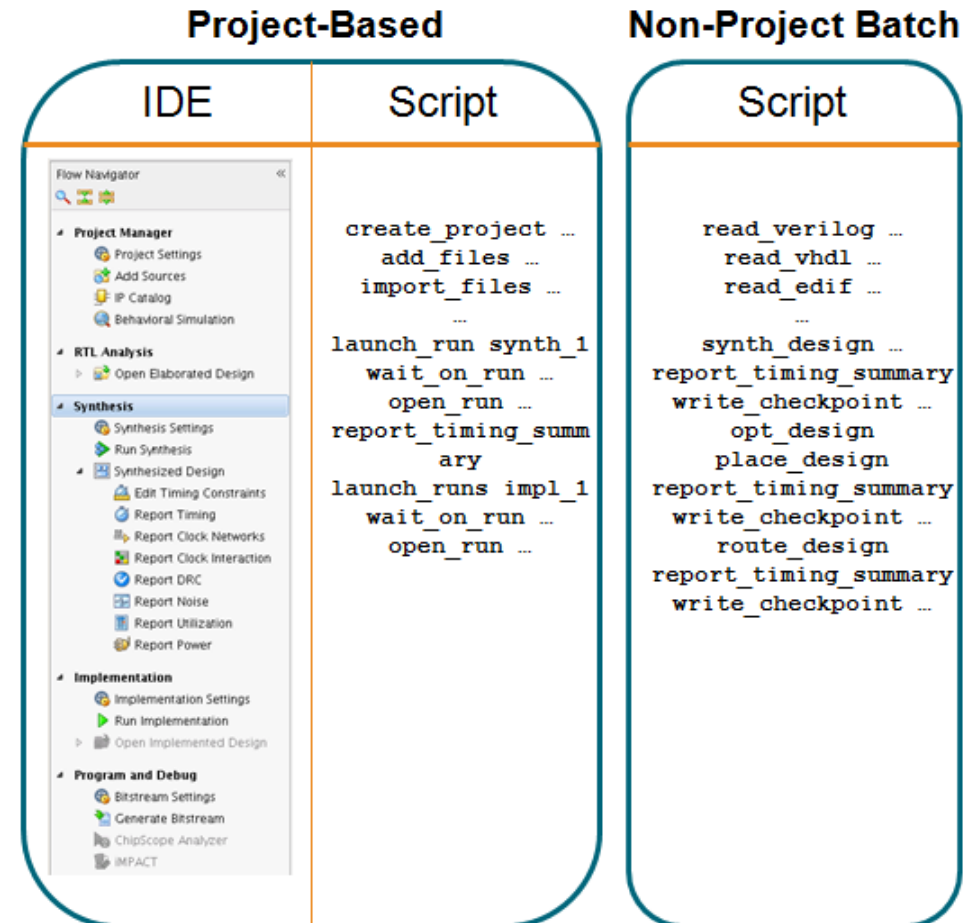
- Project based
- Non-project batch

➤ Non-project batch flow

- No project infrastructure
- Tcl based
- Can use GUI for visualization via the start_gui command
- Must manually create reports and checkpoints via commands

➤ Project-based flow

- Project infrastructure is saved in *.XPR file
- Reports/state/runs/cross-probing is available
- IDE GUI or Tcl script both available

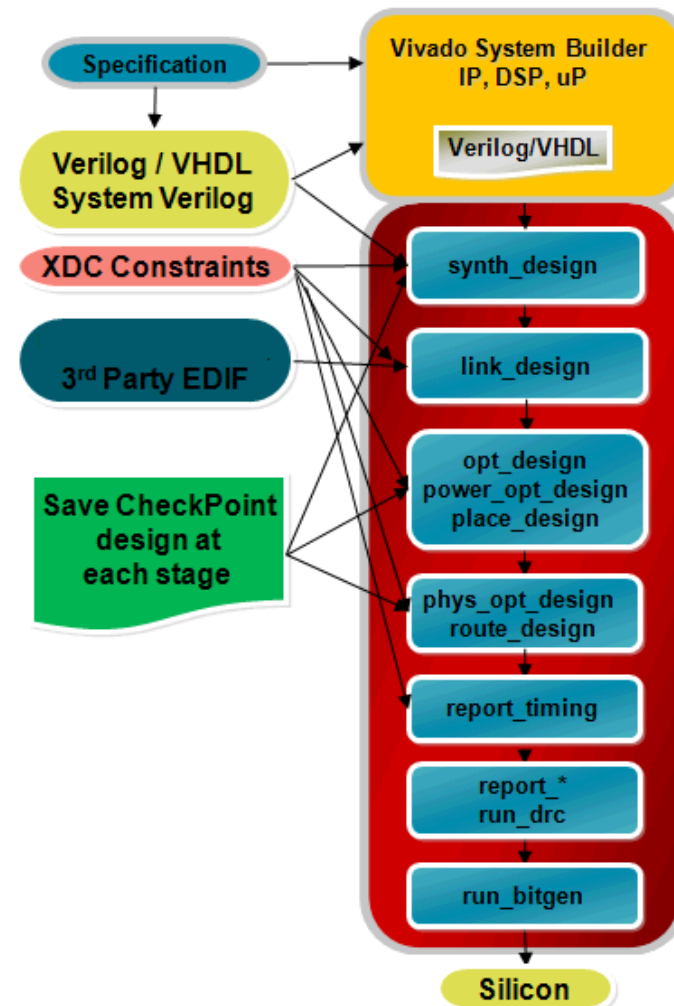


Outline

- **Vivado IDE Features and Benefits**
- ***Vivado Design Suite Introduction***
- **Vivado Design Flow**
- **Summary**

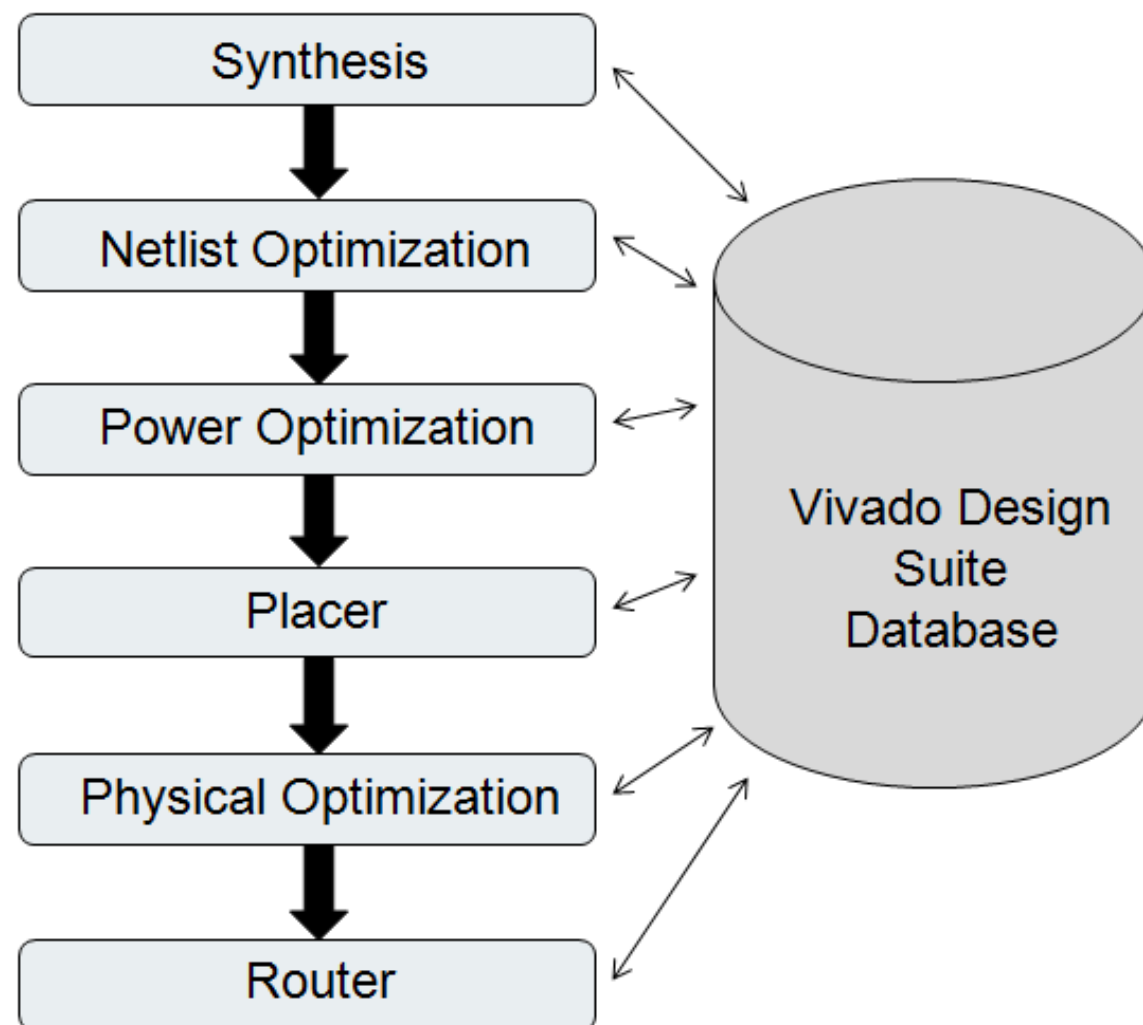
Typical vs Vivado Design Flow

- **Interactive IP plug-n-play environment**
 - AXI4, IP_XACT
- **Common constraint language (XDC) throughout flow**
 - Apply constraints at any stage
- **Reporting at any stage**
 - Robust Tcl API
- **Common data model throughout the flow**
 - “In memory” model improves speed
 - Generate reports at all stages
- **Save checkpoint designs at any stage**
 - Netlist, constraints, place and route results



Design Database

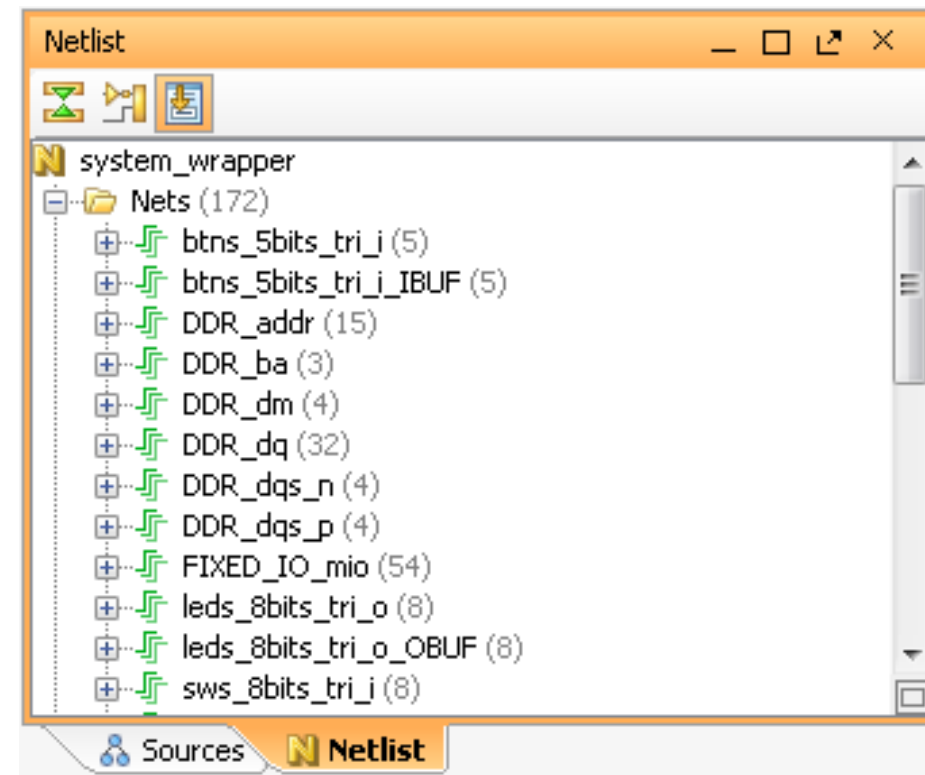
- **Processes access the underlying database of your design**
 - Each process operates on a netlist and will modify the netlist or create a new netlist
- **Different netlists are used throughout the design process**
 - Elaborated
 - Synthesized
 - Implemented



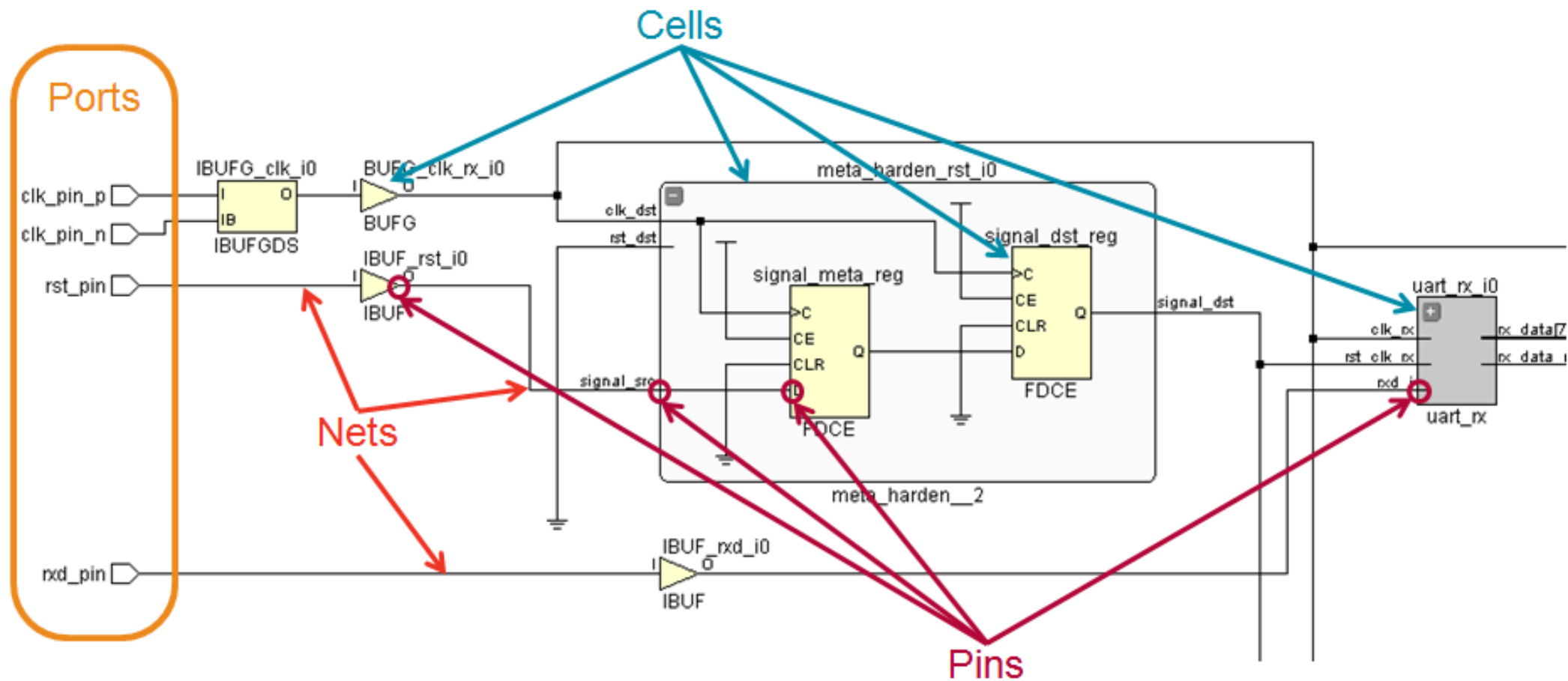
What is a Netlist?

➤ A Netlist is a description of your design

- Consists of cells, pins, port and nets
- Cells are design objects
 - Instances of user modules/entities
 - Instances of library Basic Elements (BELs)
 - LUTs, FF, RAMs, DSP cells, etc...
 - Generic technology representations of hardware functions
 - Black boxes
- Pins are connection points on cells
- Ports are the top level ports of your design
- Nets make connections between pins and from pins to ports



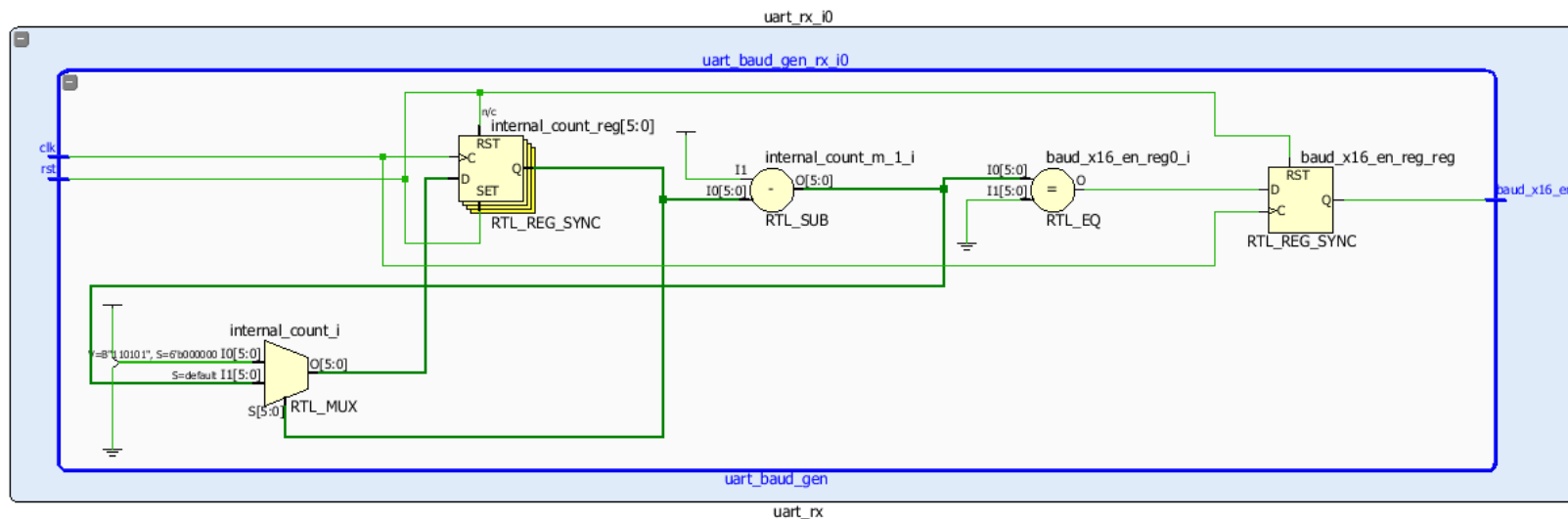
Netlist Objects



Elaborated Design

➤ Representation of the design before synthesis

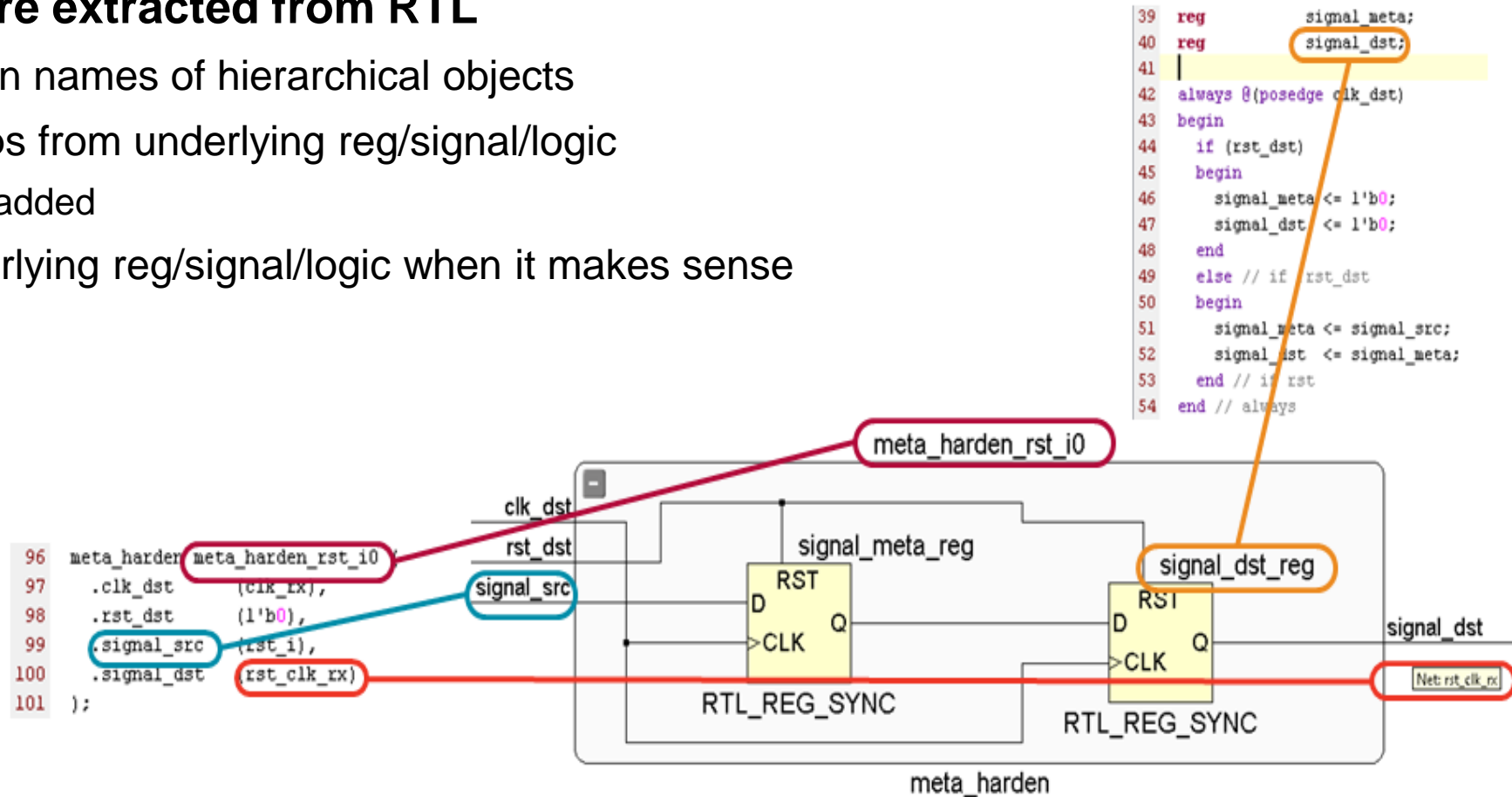
- Interconnected netlist of hierarchical and generic technology cells
 - Instances of modules/entities
 - Generic technology representations of hardware components
 - AND, OR, buffer, multiplexers, adders, comparators, etc...



Object Names in Elaborated Design

➤ Object names are extracted from RTL

- Instance and pin names of hierarchical objects
- Inferred flip-flops from underlying reg/signal/logic
 - Suffix _reg is added
- Nets from underlying reg/signal/logic when it makes sense



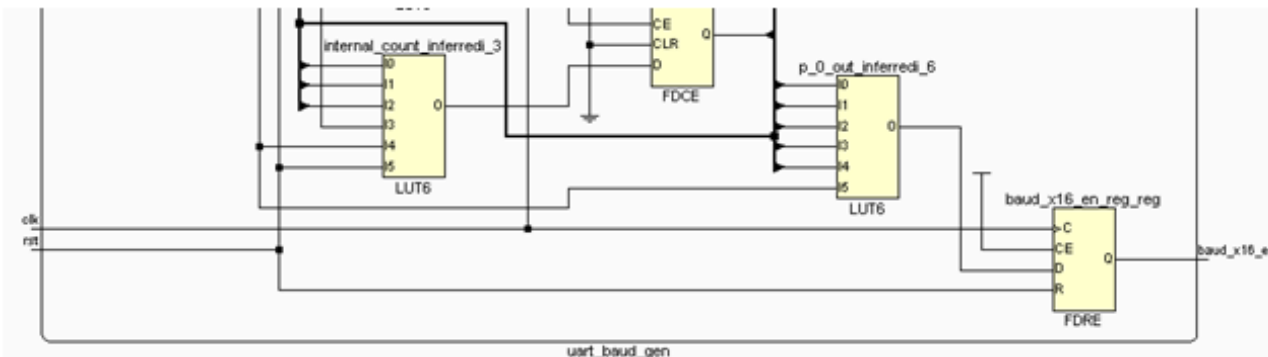
Synthesized Design

➤ Representation of the design after synthesis

– Interconnected netlist of hierarchical and Basic Elements (BELs)

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

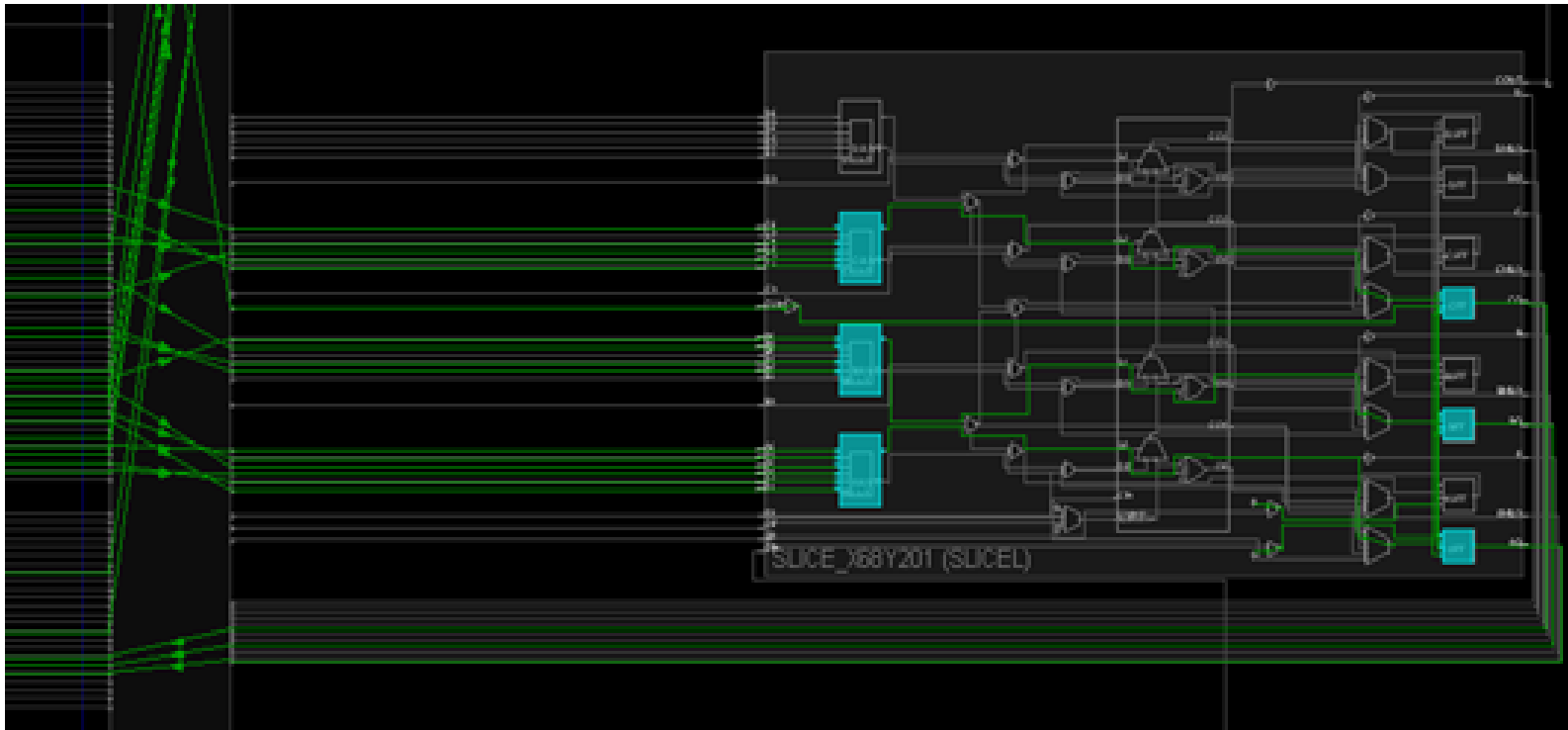
➤ Object names are the same as names in the elaborated netlist when possible



Implemented Design

➤ Representation of the design during and after the implementation process

- Structurally similar to the Synthesized Design
- Cells have locations, and nets are mapped to specific routing channels



Project Data

➤ All project data is stored in a *project_name* directory containing the following basic set of directories:

- *project_name.xpr* file: Vivado project file, contains project settings
- *project_name.runs* directory: Contains all run data
- *project_name.srcs* directory: Contains all imported local HDL source files, netlists, and XDC files
- *project_name.data* directory: Stores floorplan and netlist data

Journal and Log Files

➤ Journal file (vivado.jou)

- Contains just the Tcl commands executed by the Vivado IDE

➤ Log file (vivado.log)

- Contains all messages produced by the Vivado IDE, including Tcl commands and results, info/warning/error messages, etc.

➤ Location

- Linux: directory where the Vivado IDE is invoked
- Windows via icon: %APPDATA%\Xilinx\Vivado or
C:\Users\<user_name>\AppData\Roaming\Xilinx\Vivado
- Windows via command line: directory where the Vivado IDE is invoked
- From the GUI
 - Select File > Open Log File
 - Select File > Open Journal File

Checkpoints

➤ Saves the database from memory to disk

- Netlist – edif (eventually a Verilog netlist)
- Constraints – xdc
- Placement/routing/config db – xdef

➤ Tcl commands

- write_checkpoint <filename>
 - Results in a filename.dcp file, which is a compressed file
 - Can unzip this file if needed: filename.edf, filename.xdc, filename.xdef, filename.wdf, and dcp.xml
- read_checkpoint <filename>

➤ Can read and write in the IDE as well

- When read in gives a project-less setup

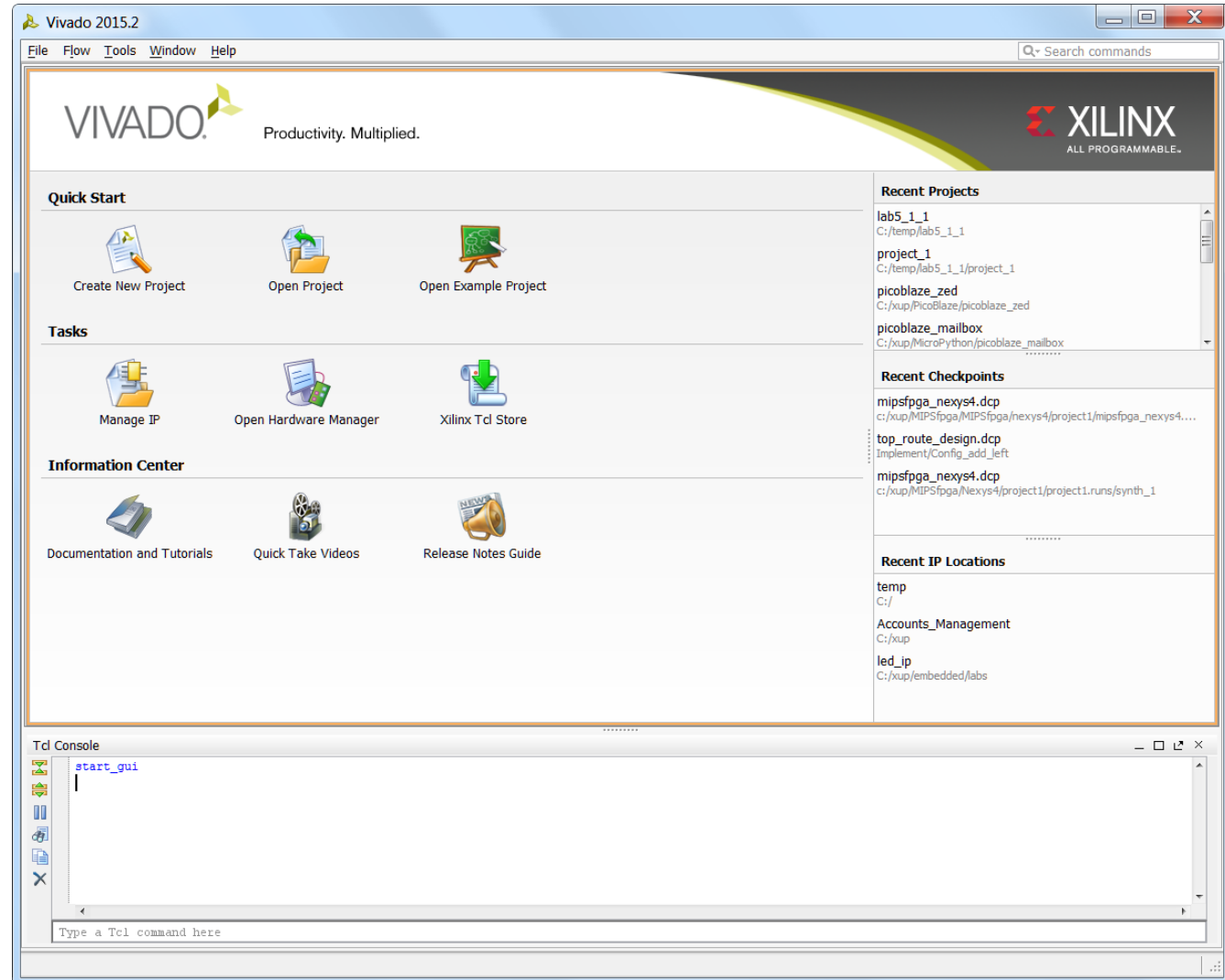
➤ Can open and modify checkpoints in IDE

Outline

- Vivado IDE Features and Benefits
- Vivado Design Suite Introduction
- *Vivado Design Flow*
- Summary

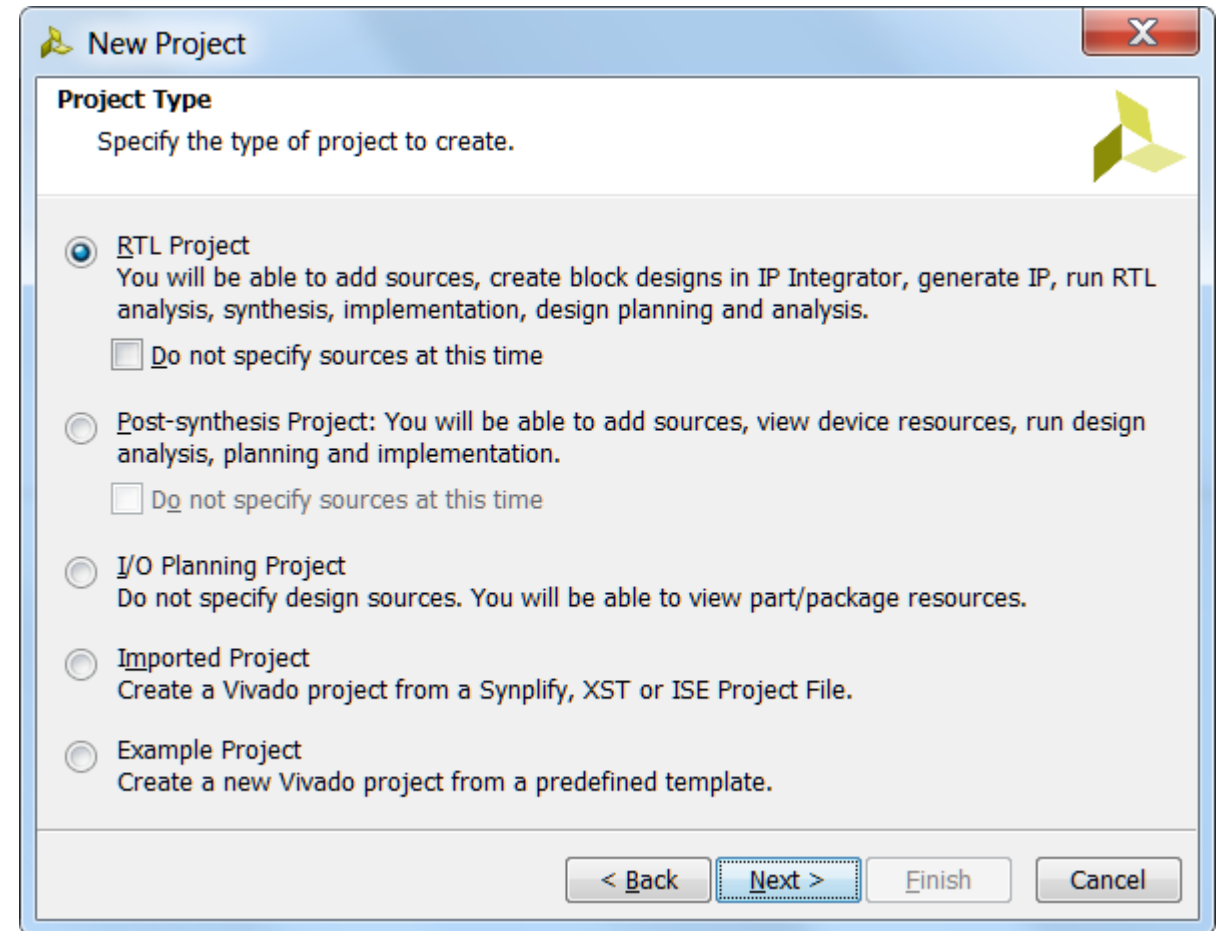
Getting Started Jump Page

- **All buttons grouped by functionality**
- **Quick Start**
 - Previous projects can quickly opened
 - Links to create new or example projects
- **Tasks**
 - IP management, hardware manager, Tcl Store
- **Helpful links**
 - Documentation and Tutorials
 - Invokes PDF viewer for documentation
 - Quick Take Videos
 - Release Notes Guide
- **Tcl Console for command line access**



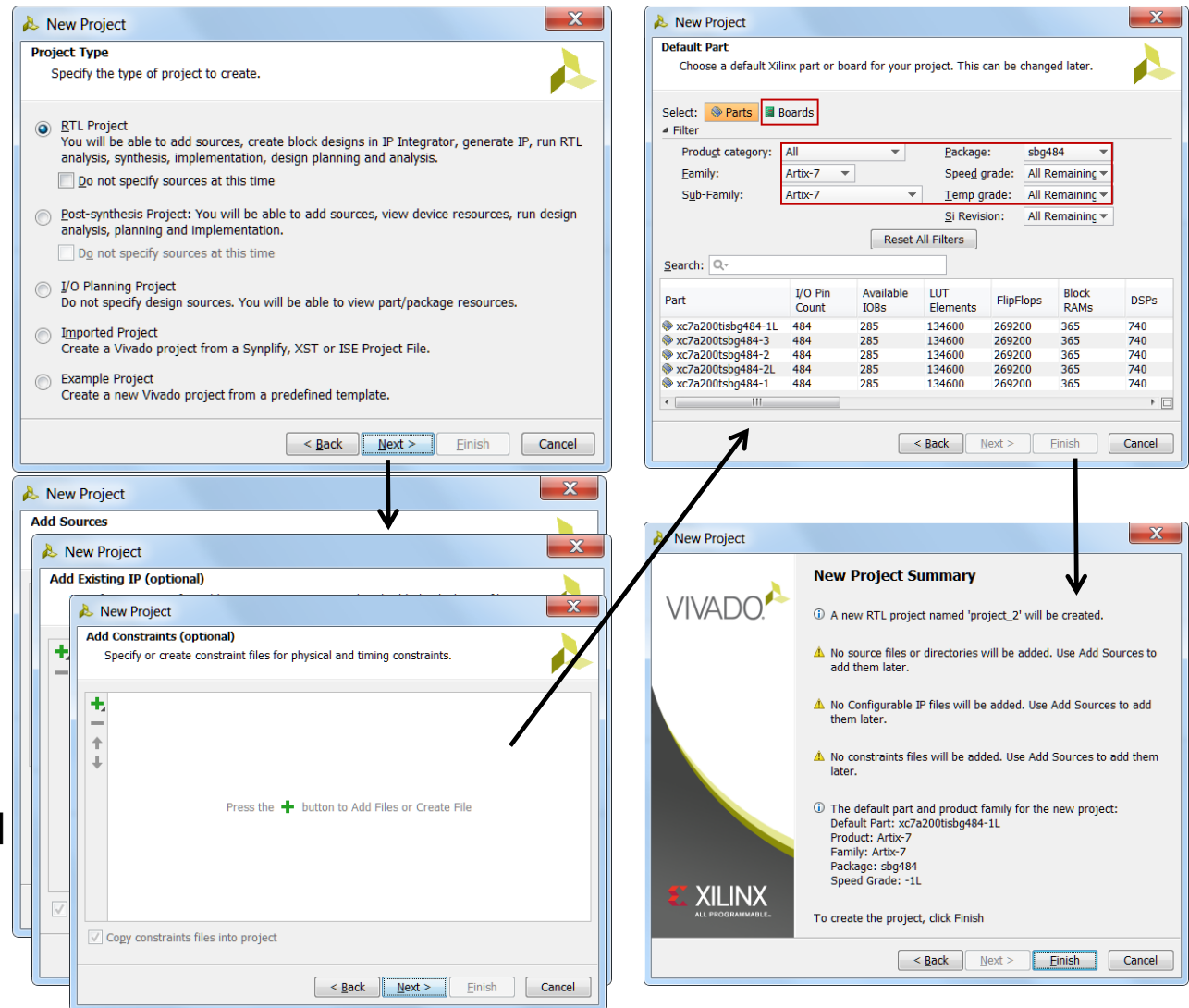
New Project Creation Wizard

- Very first step is project creation with five choices
- Five different types of projects:
 - RTL
 - Front-to-back (even for Block Designs)
 - Post-synthesis
 - EDIF or NGC
 - I/O planning
 - For early pin testing
 - No design sources
 - Import Project
 - Imports existing project from Synplify, XST, or ISE
 - Example Project



Project creation flow (RTL/Synthesized Design)

- Defines the project name and location
- Select source files in RTL project creation
 - All recognized source files, Verilog, VHDL, in the directory and subdirectories, can be added
- Select post-synthesized netlist in Post-synthesized project creation
 - All synthesized files in the directory and subdirectories, are added
- Select constraint files
 - One or more constraints files including IP specific and top-level can be added
- Select target device or pre-defined board
- Reference original existing files or import and copy them into the project



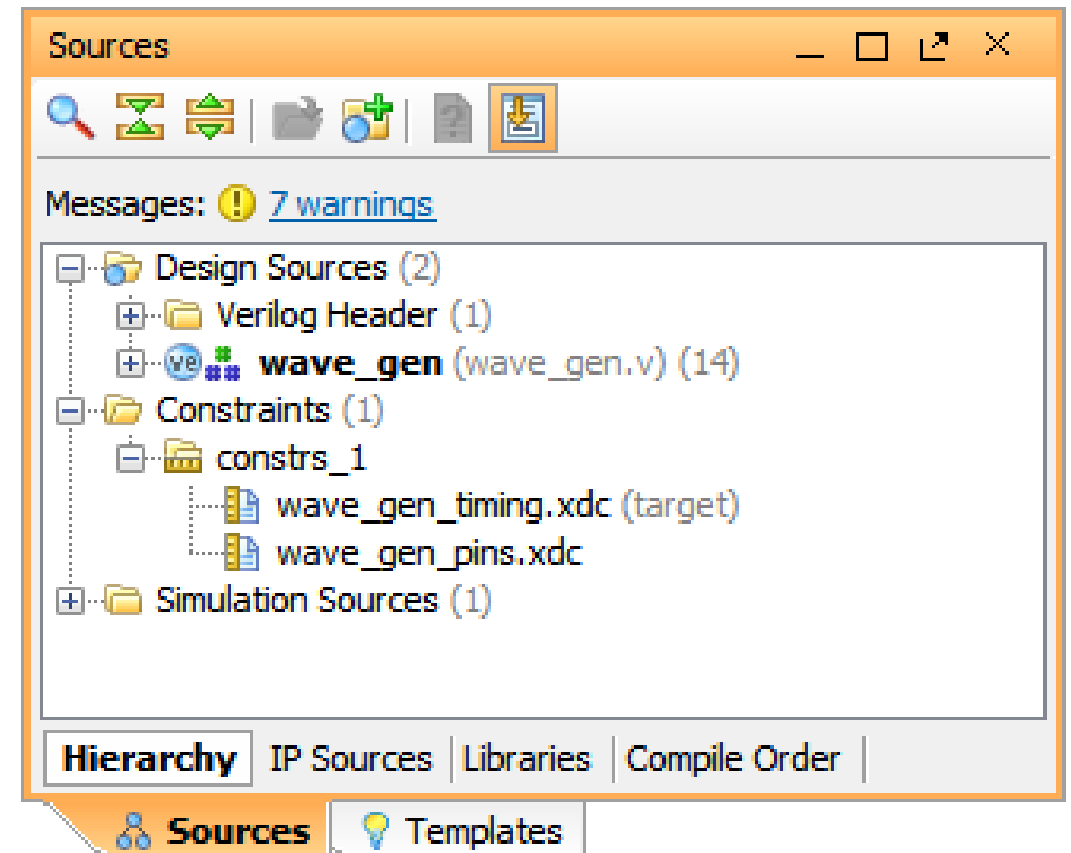
Constraints File Management

➤ Constraint sets are a collection of XDC files

- A project can contain multiple constraint sets
- For a constraints set to be applied, it must be set to "active"
 - Any constraint set can be made active by right-clicking and selecting Make Active

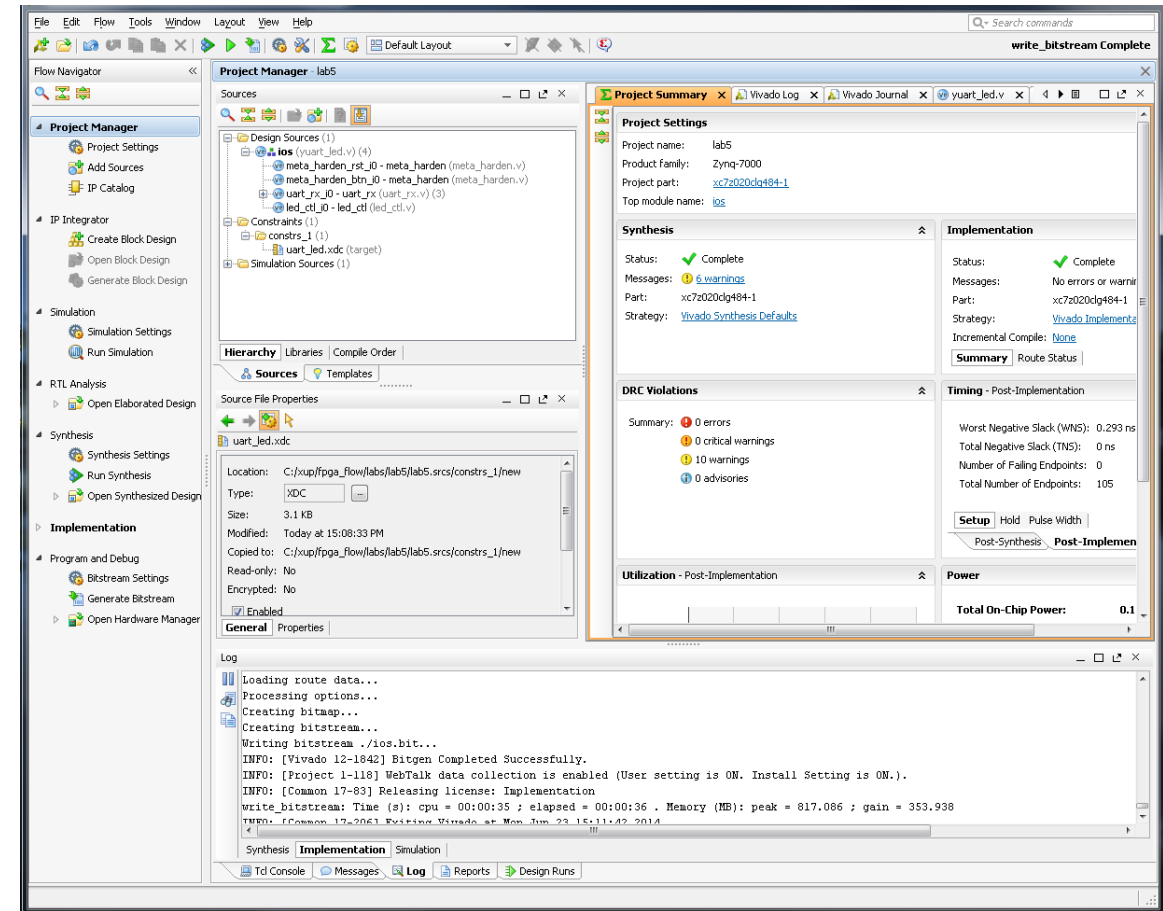
➤ Target XDC

- The XDC file in a constraint set to which new constraints are written
 - From the Constraints Wizard for example
- Target XDC can be specified by right-clicking and selecting Set As Target Constraint File



Project Navigator

- Used to manage sources, customize IP, and view project details in the Project Summary
- Flow Navigator
- Sources view
 - Hierarchical display of sources, including constraints files
 - IP Sources and Libraries view
 - HDL and netlists including references to library and location
- Project Summary
 - Gives access to device utilization (resources), timing summary, and strategy information
- Tcl Console, Messages, Compilation, Reports, and Design Runs

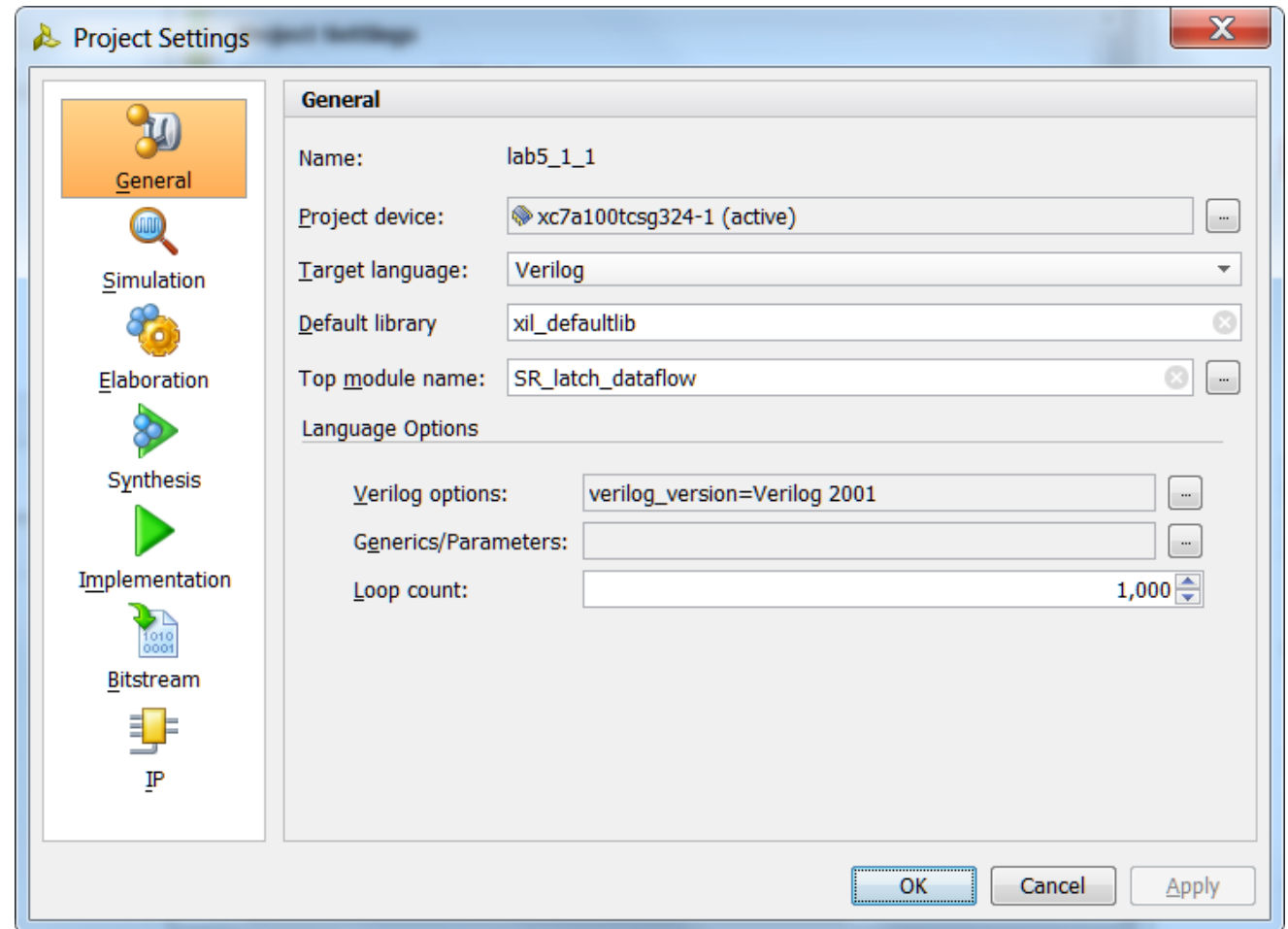


Project Settings

➤ General settings

- Select device
- Target HDL language
- Simulation tool (Vivado simulator included)
- Top module name
- Language options

➤ Other settings are covered in their respective modules



Flow Navigator – RTL Project

➤ Configure project sources

- Add HDL source files, constraints files, simulation files, block designs

➤ IP Integrator

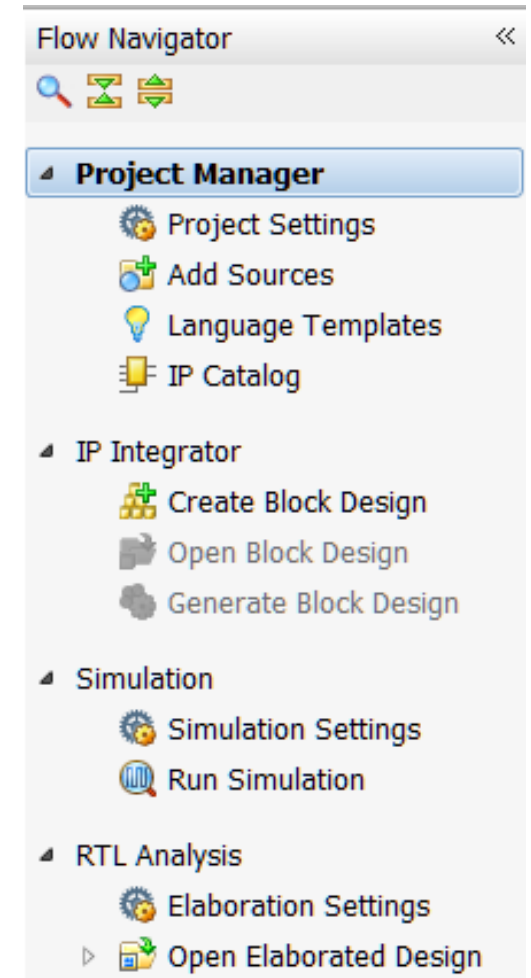
- Create, open, generate a block design

➤ Run Simulation

- XSIM simulator included
- Behavioral, post-synthesis, post-implementation

➤ RTL Analysis

- Open Elaborated Design button: Loads the elaborated RTL design



Flow Navigator – RTL Project, cont'd...

➤ Run Synthesis

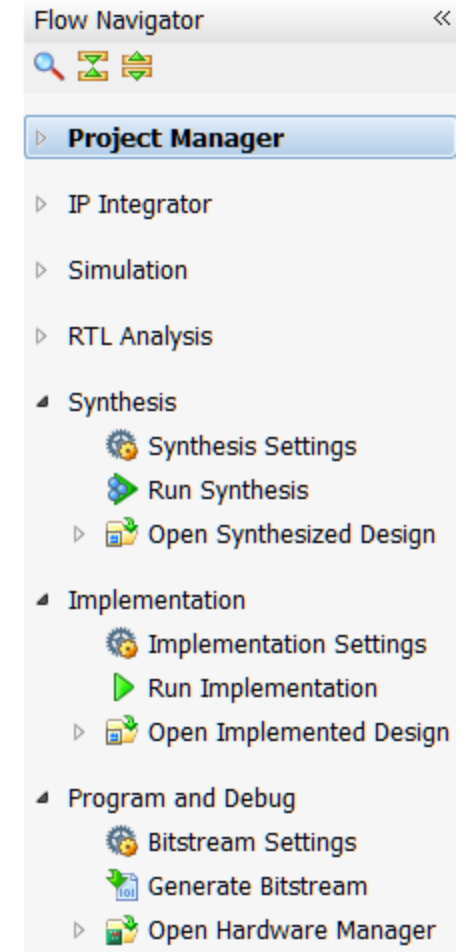
- Timing driven
- Open Synthesized Design button: Loads synthesized netlist

➤ Run Implement button: Runs implementation tools

- link, opt, power_opt, place, phys_opt, and route
- Open Implemented Design button: Loads implemented design

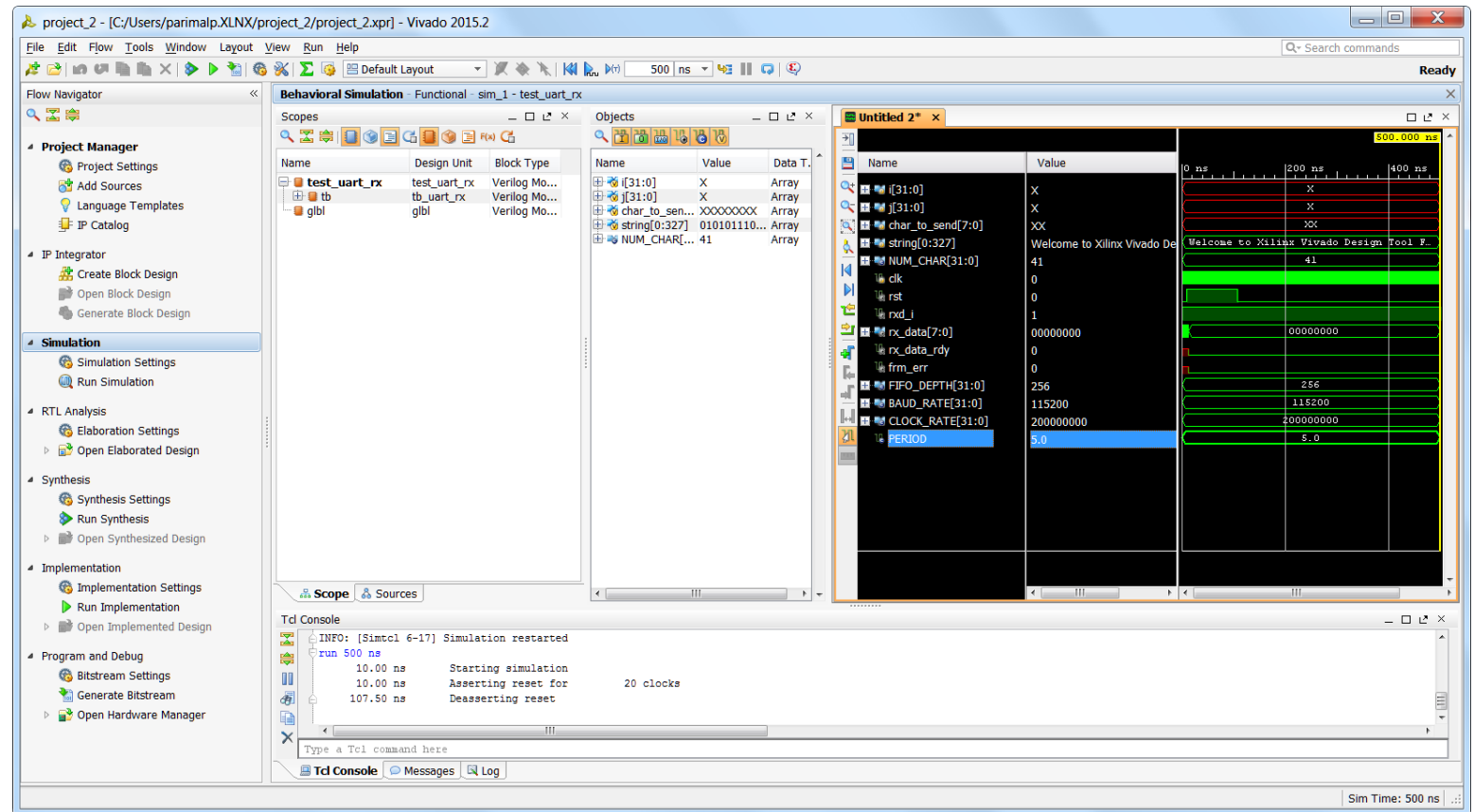
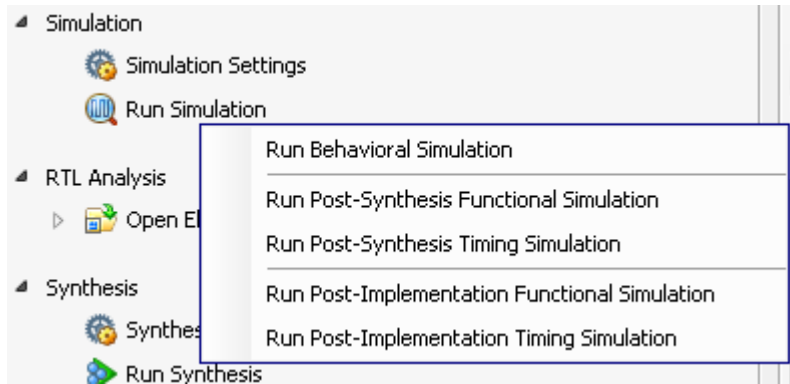
➤ Program and Debug: Launches programming and debugging tools

- Open Hardware Manager to program the FPGA
- Hardware Manager also contains debug capabilities



Simulation

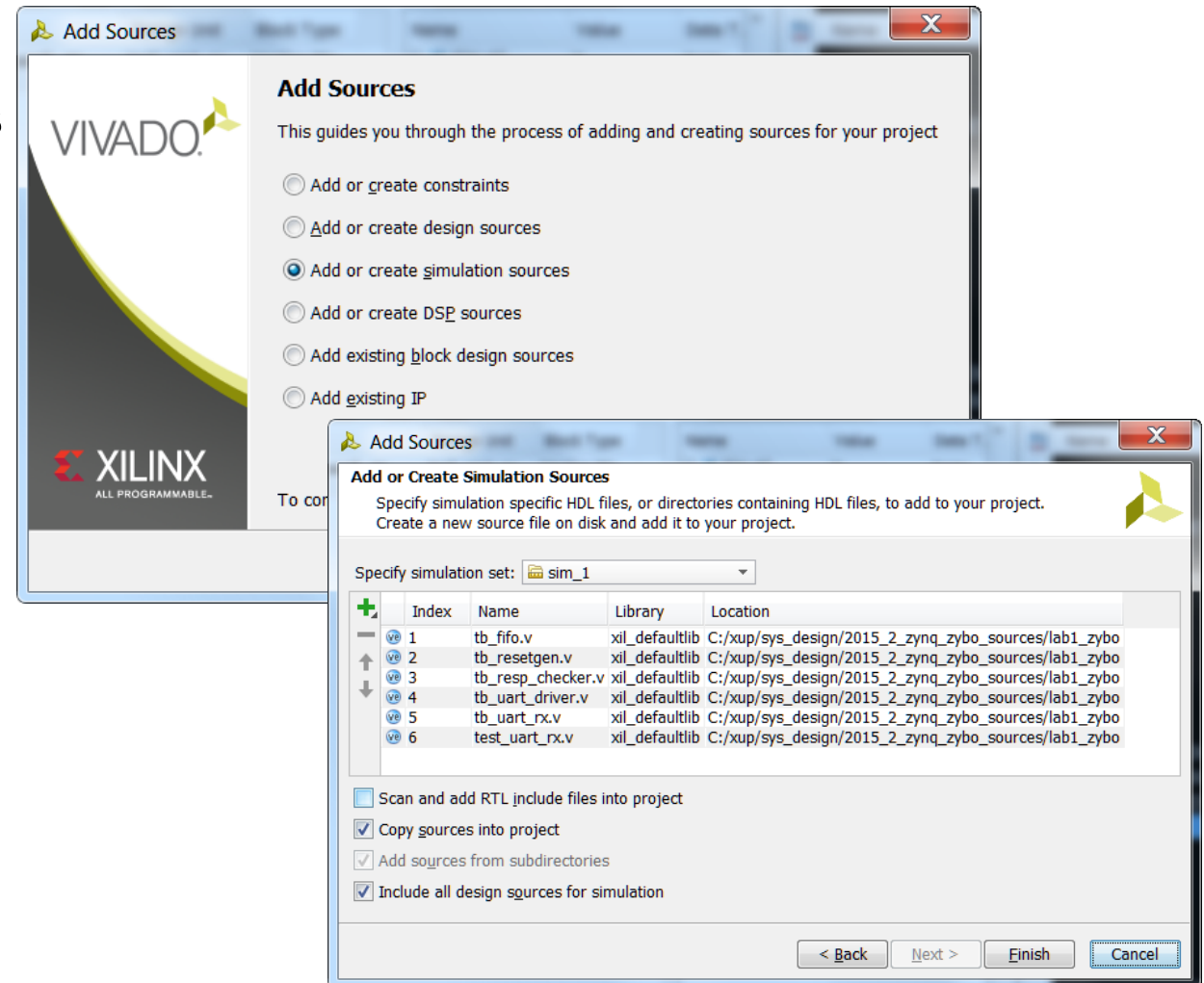
- The Vivado simulator, XSIM, supports RTL, netlist, and timing simulation
- Part of the Vivado installation



Adding Testbench Files

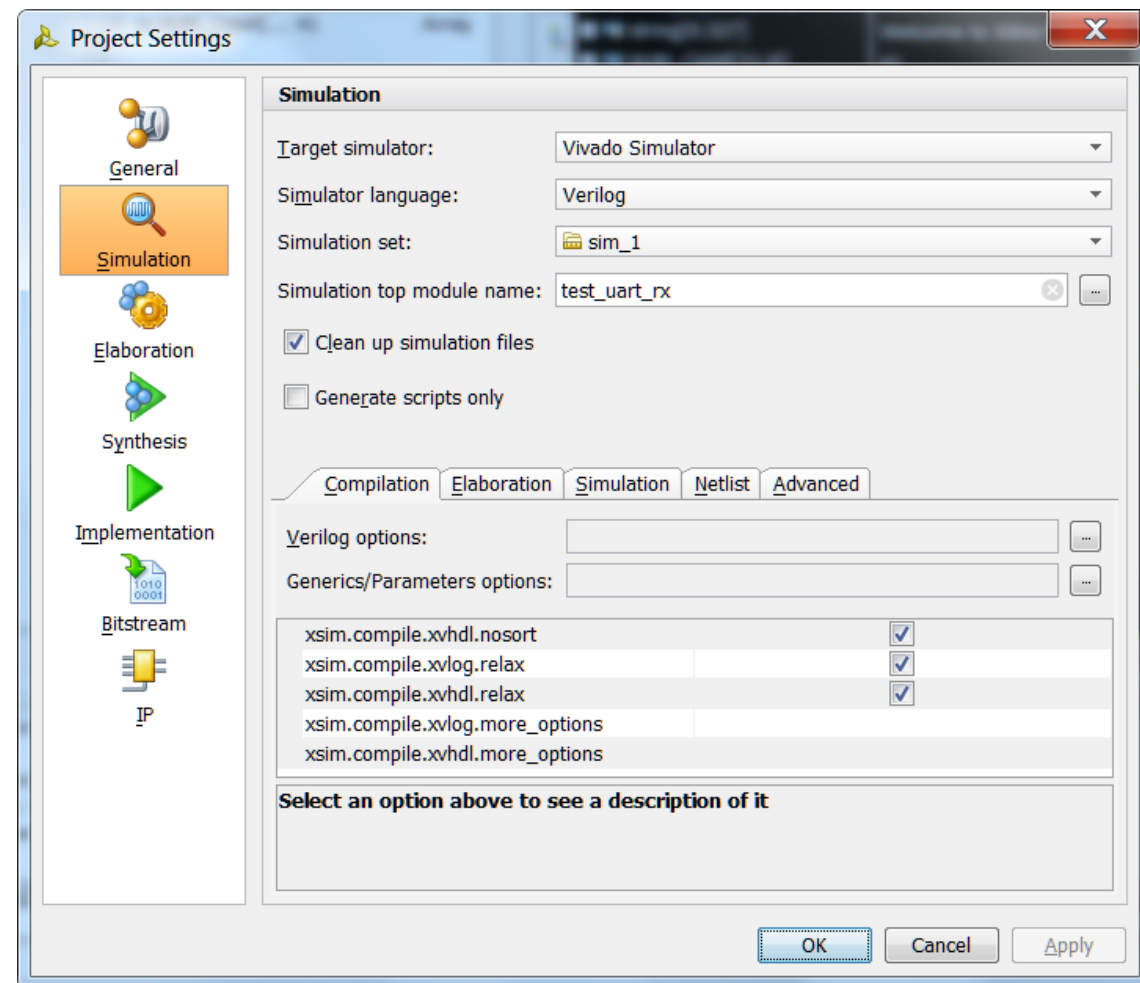
➤ Testbench sources (*.V or *.VHD) must be added separately from design sources

- In the Flow Navigator, click Add Sources
- Select Add or Create Simulation Sources option and click Next
- Click on the Green + button, add files..., add directories... if the testbench file is already available or click on Create File... button to create a new testbench file
- Select a file type- Verilog, Verilog Header, SystemVerilog, or VHDL
- Browse to an existing testbench or enter a filename to create



Simulation Settings

- **Allows selection of compilation and simulation properties**
 - From the Flow Navigator, click Simulation Settings
 - The simulation top module is the testbench you have written and specify
 - Additional options can be entered
 - More Compilation Options field under Compilation tab
 - More Simulation Options field under Simulation tab
- **Refer to the Vivado Design Suite Simulation Guide (UG900) for more information**



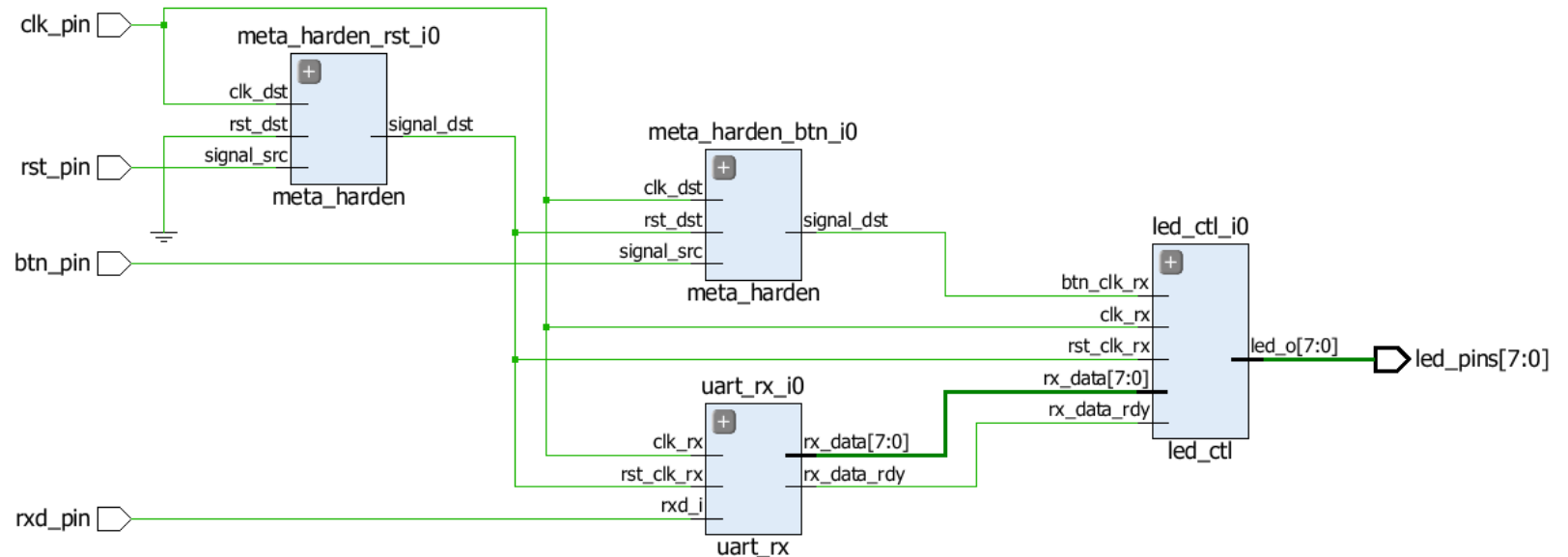
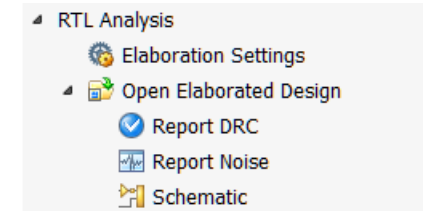
Simulation Results

- Graphical waveform display
- Toolbar buttons for adding markers, measuring delays, and zooming
- Buses can be expanded to view individual signals
- Dividers can be inserted to visually isolate groups of related signals
- Console displays any messages output from testbench
- By default, the top-level signals are displayed

After Elaboration

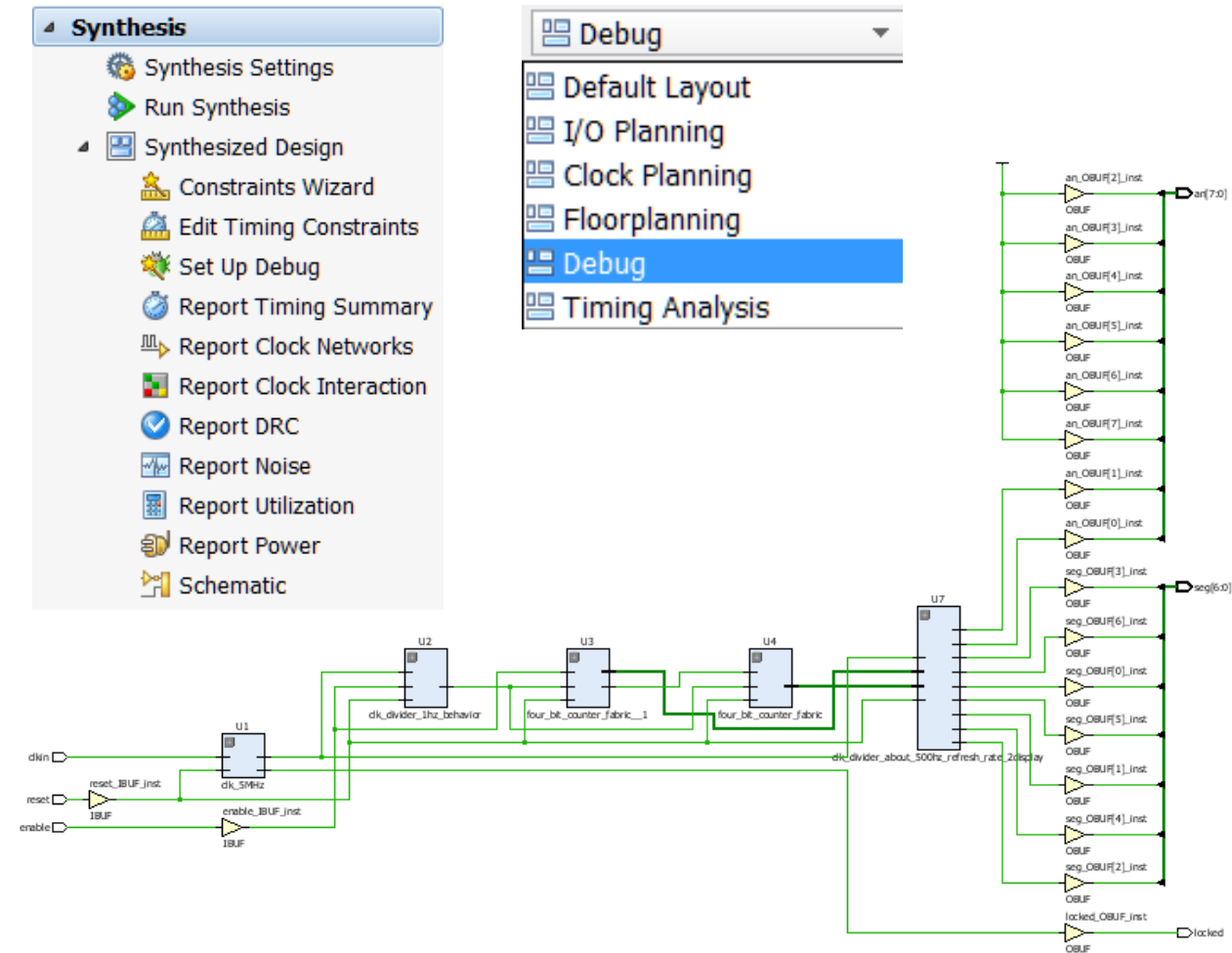
- Sources, RTL Netlist, and Device Constraints tabs available
- Properties of selected items
- Generate DRC and Noise reports
- RTL Schematic View
- Views *can be selected by purpose*

- Clock Planning
- I/O Planning
- Floorplanning



After Synthesis

- The Flow Navigator includes access to Constraints Wizard, Edit Timing Constraints, Set Up Debug, Report Timing Summary, Report Clock Networks, Report Clock Interaction, Report DRC, Report Noise, Report Utilization, Report Power, and Schematic utilities
- The schematic view will be opened including input/output buffers
- Views can be selected by purpose
 - I/O Planning, Clock Planning, Floorplanning, Debug, Timing Analysis
 - All timing information is only an estimate (until implementation has completed)



After Implementation

➤ Sources and Netlist tabs do not change

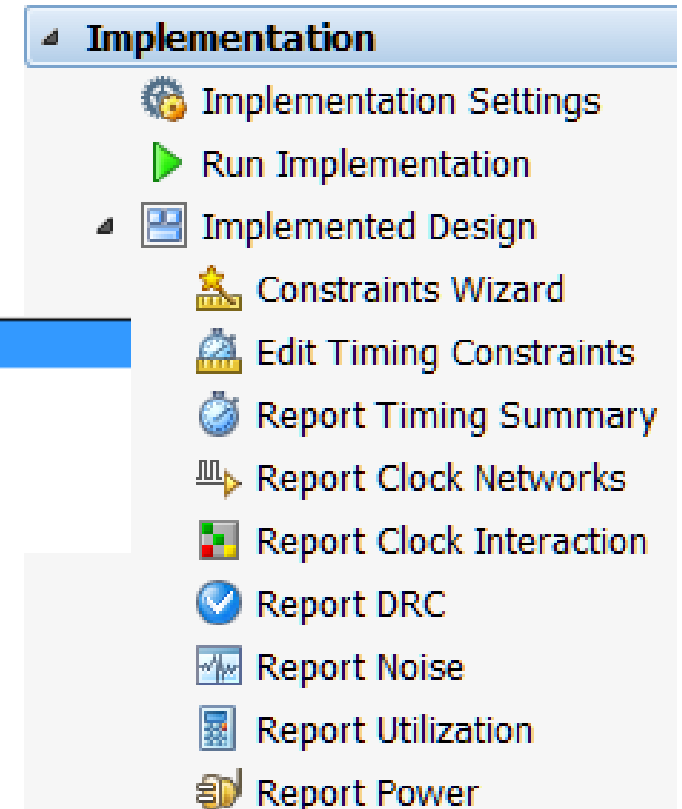
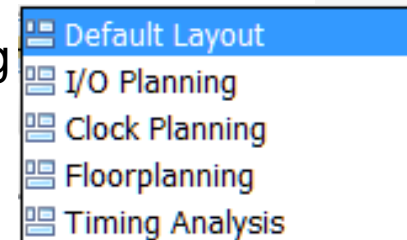
- As each resources in the Netlist tab is selected, it will show the exact placement of the resource on the die in the Device view
- As each path is selected, the placement of the logic and its connections is shown in the Device view
- This is the cross-probing feature that helps with static timing analysis

➤ Views can selected by purpose

- I/O Planning, Clock Planning, Floorplanning, Timing Analysis

➤ Access to Constraints Wizard

➤ Timing results have to be generated with the Report Timing Summary



Outline

- Vivado IDE Features and Benefits
- Vivado Design Suite Introduction
- Vivado Design Flow
- ***Summary***

Summary

➤ Features and benefits of the Vivado IDE include

- Improved performance and device utilization with the use of Pblocks and area constraints
- Performance predictability
- Design analysis features that speed a designer's ability to gain timing closure Tcl features (commands) that make scripting easier and powerful

➤ Vivado tools use a common data model throughout the FPGA design process

- This yields runtime and memory resource benefits to the user

➤ Vivado tools support scripting in non-project batch and project-based design flows

- Vivado tools support the use of Tcl for all commands

➤ Vivado tools use a common constraint language (XDC) throughout the design process

- This enables synthesis optimization significantly better than the ISE software

➤ Pushbutton flows for most designs

➤ Advanced tools for challenging designs