Vivado Design Flow

Vivado 2013.4 Version

Objectives

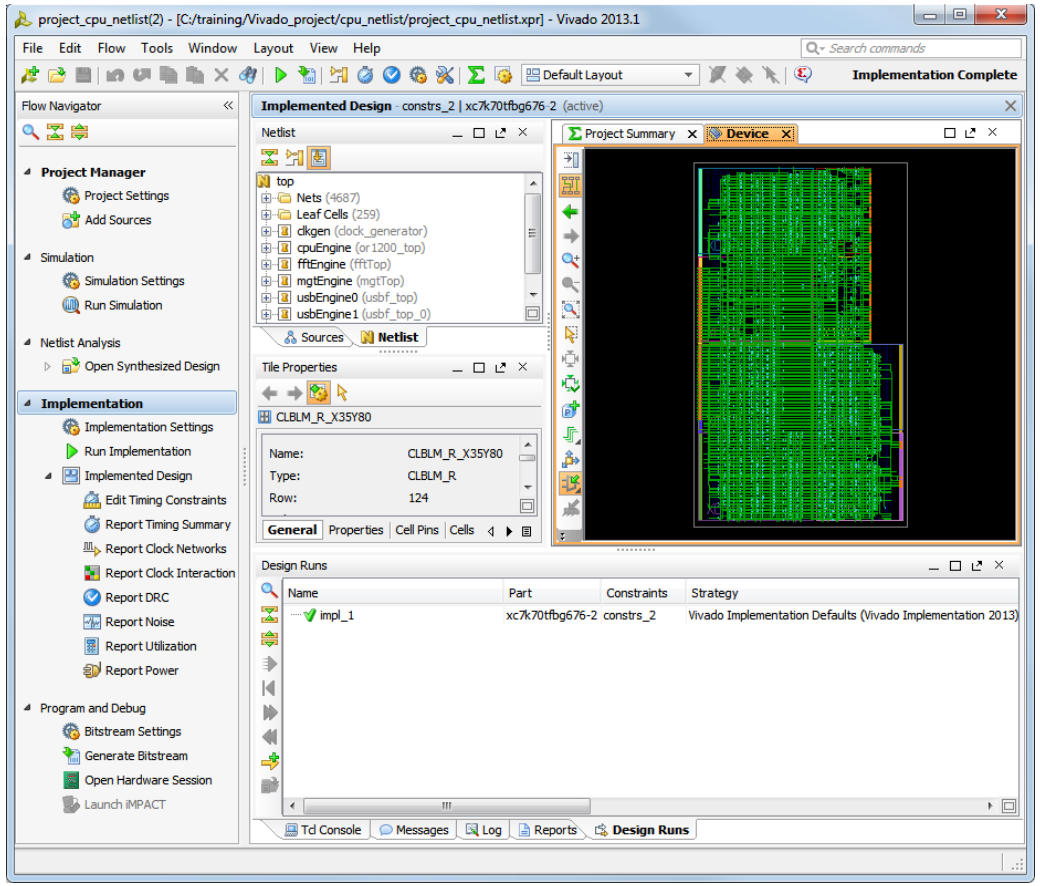
- **After completing this module, you will be able to:**
  - Explain how the design analysis features of the Vivado IDE can help in FPGA design development
  - List the main features of the Vivado IDE
  - Describe the Vivado IDE Design flow
  - Introduce the scripted Vivado IDE design flows

Outline

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

Vivado IDE Solution

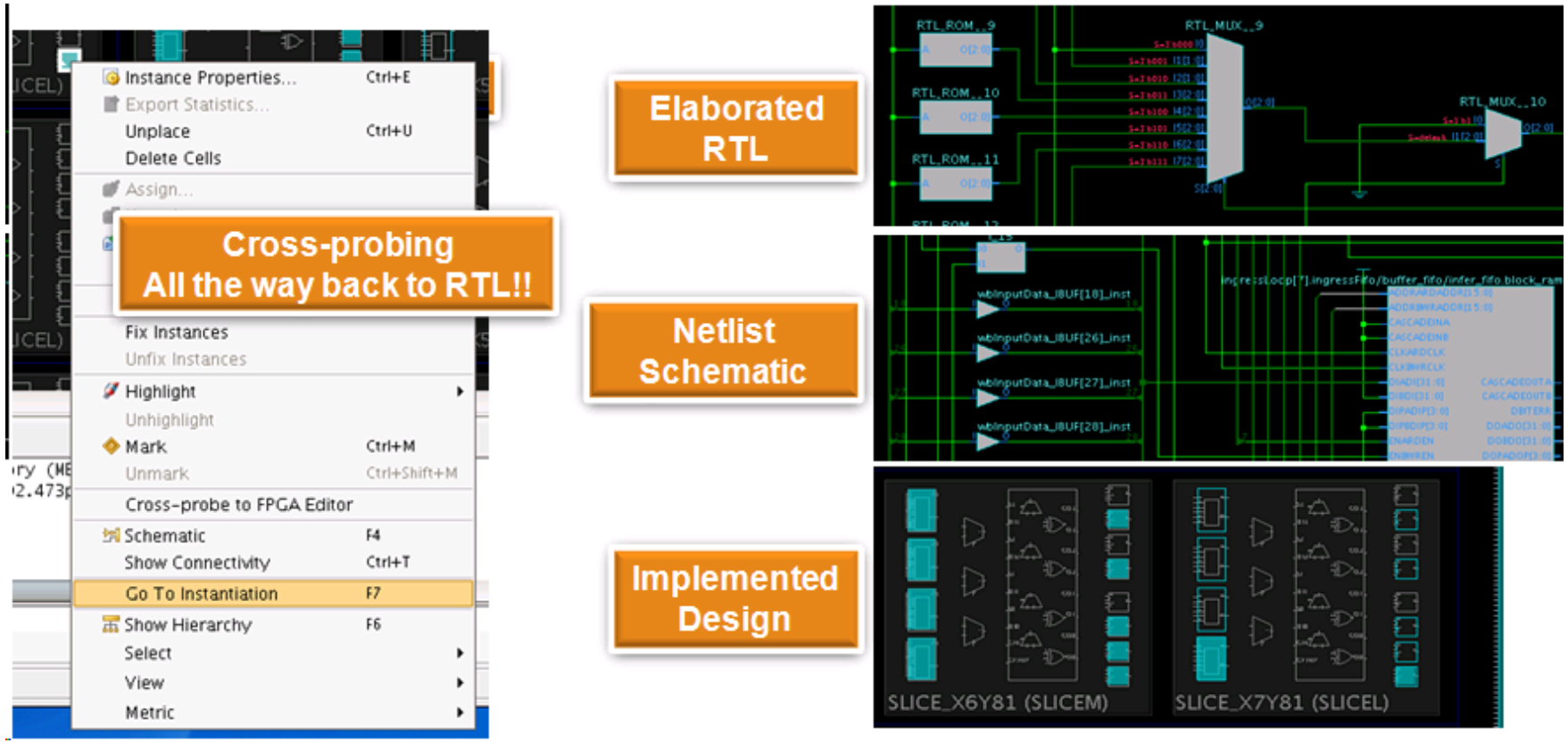
- **Interactive design and analysis**
  - Timing analysis, connectivity, resource utilization, timing constraint analysis, and entry
- **RTL development and analysis**
  - Elaboration of HDL
  - Hierarchical exploration
  - Schematic generation
- **XSIM simulator integration**
- **Synthesis and implementation in one package**
- **I/O pin planning**
  - Interactive rule-based I/O assignment



Hierarchical Design Analysis and Implementation Environment

Vivado's Visualization Feature

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



Tcl Features

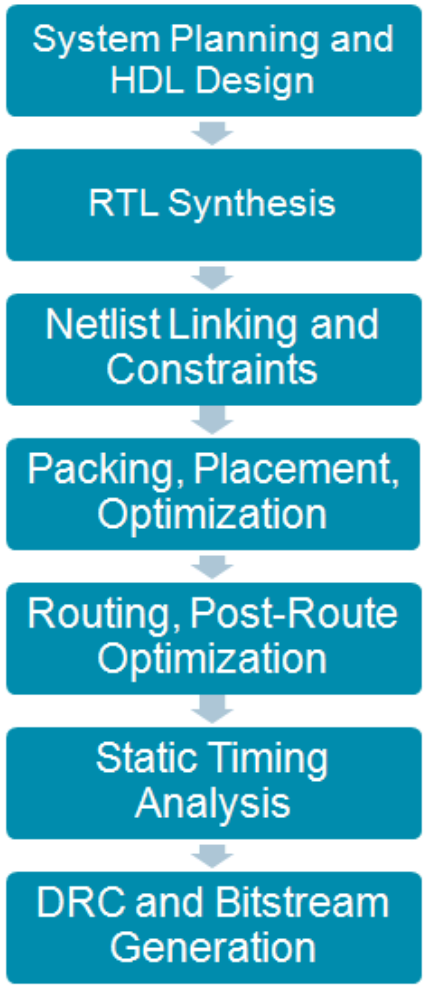
- **Tcl Console enables the designer to actively query the design netlist**
- **Full Tcl scripting support in two design flows**
  - Project-based design flow provides easy project management by the Vivado IDE
  - Non-project batch design flow enables entire flow to be executed in memory
- **Journal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

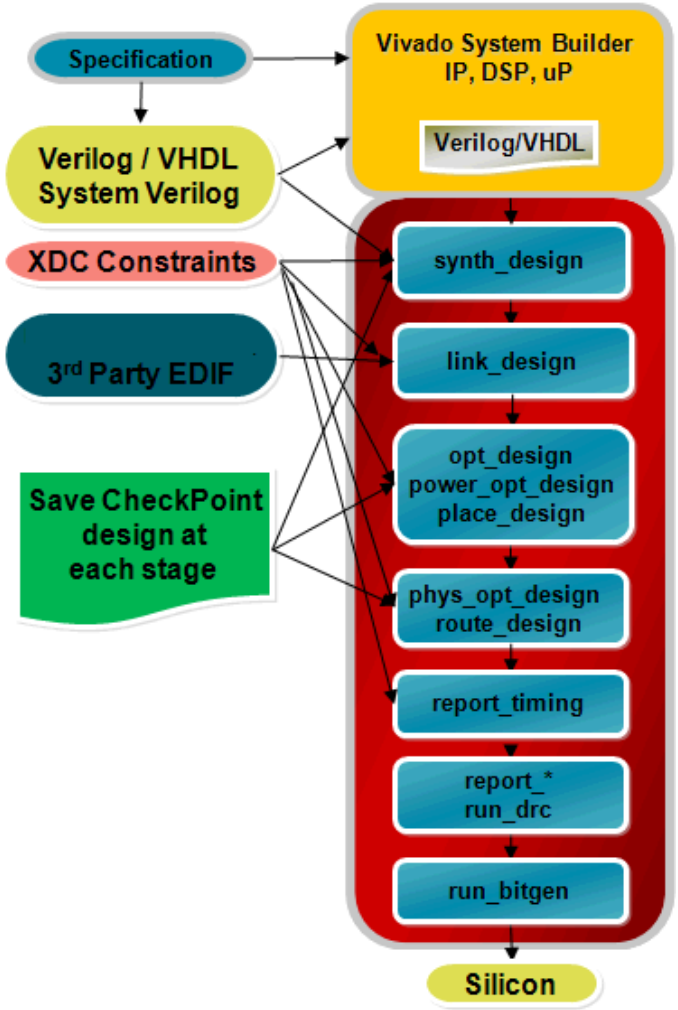
Outline

- 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

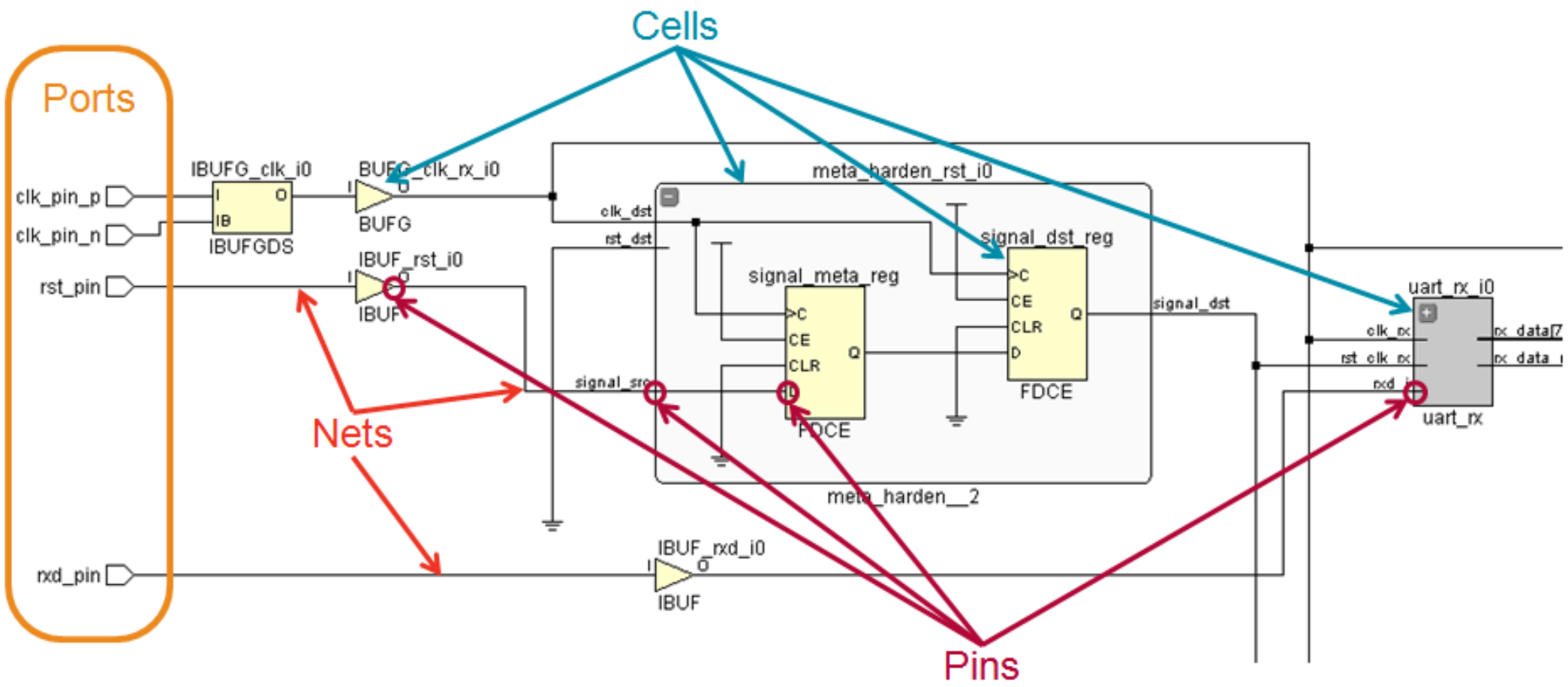




What is a Netlist?

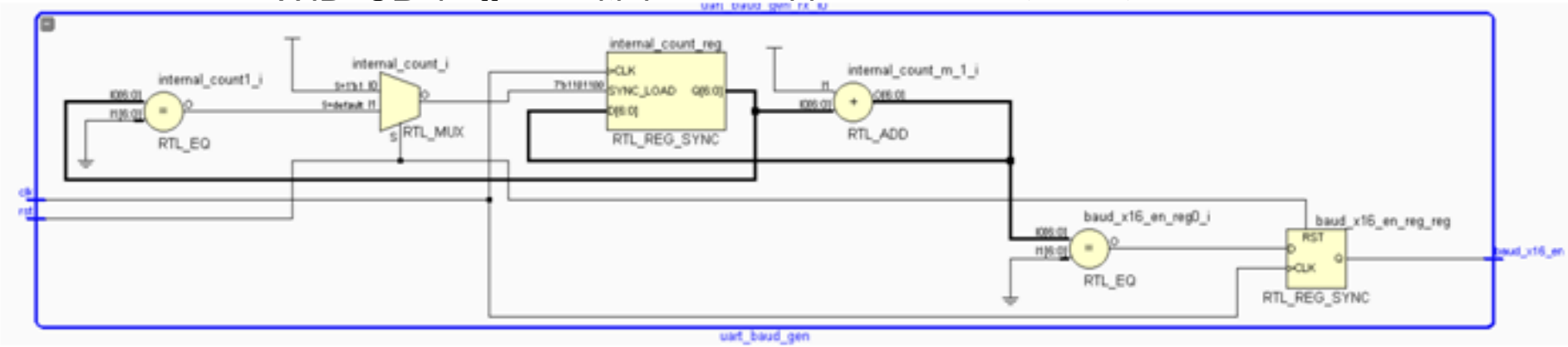
- **A netlist is a description of your design**
- **Consists of cells, pins, port and nets**
  - Cells are design objects
    - Instances of user modules/entities
    - Instances of library elements (BELs)
      - LUTs, FF, RAMs, DSP cells, etc…
    - Generic technology representations of hardware functions
    - 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

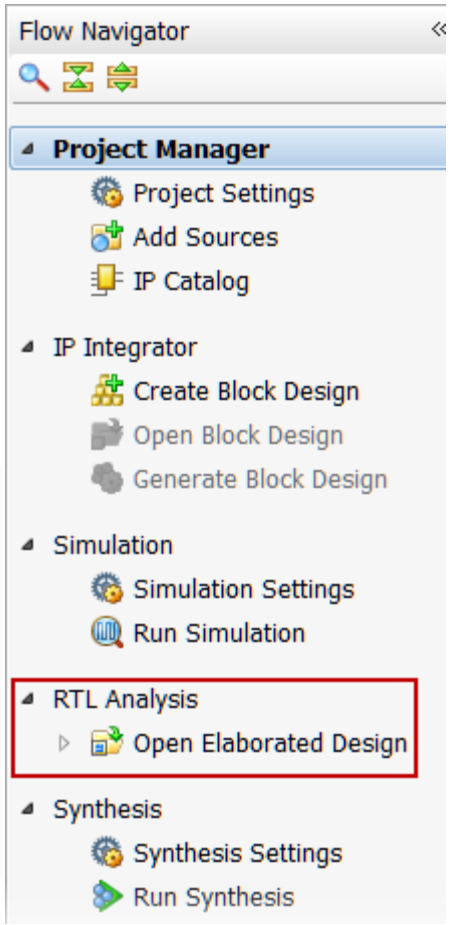
Netlist Objects



Elaborated Design

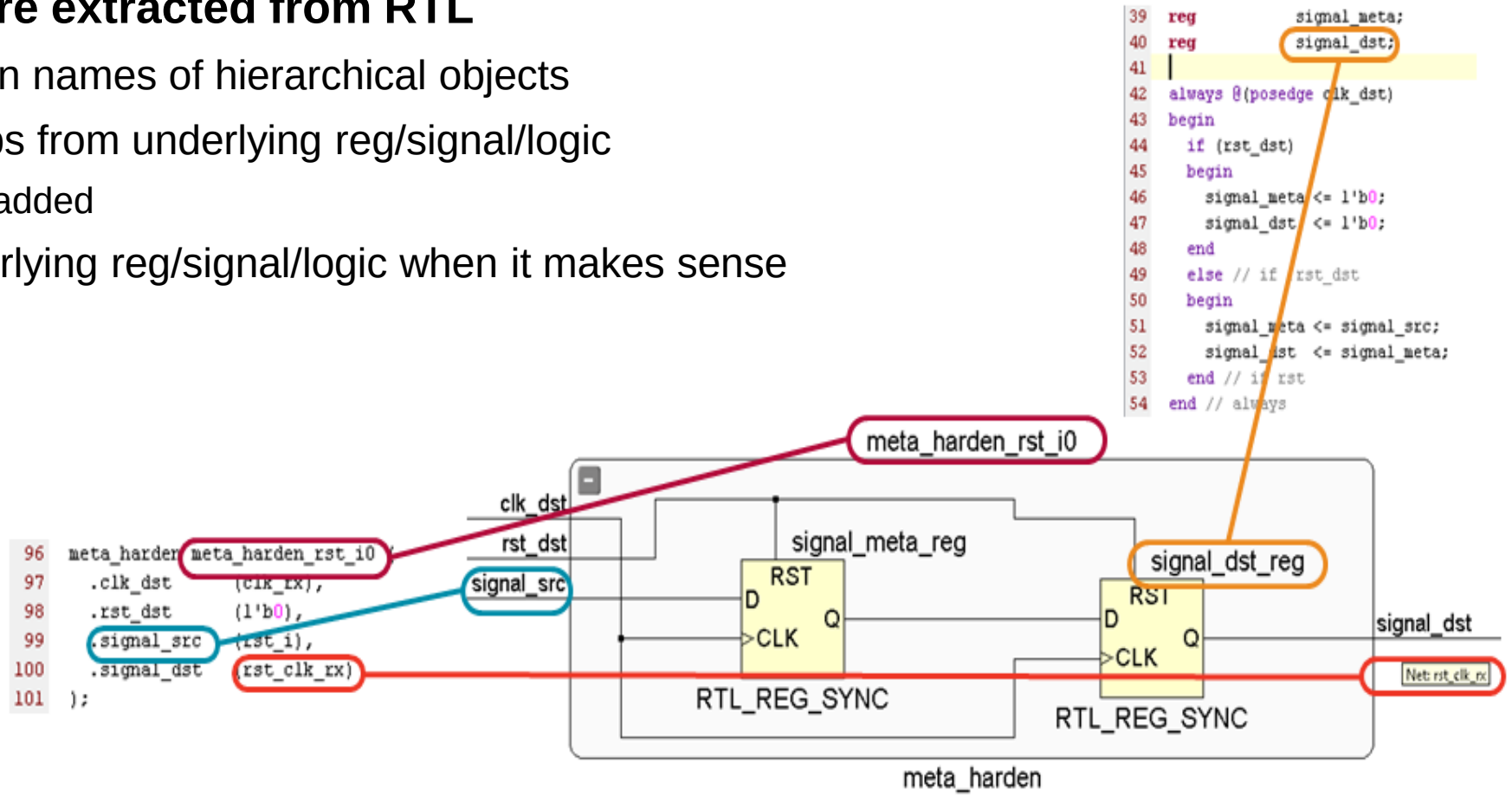
- **Accessed through the Flow Navigator by selecting Open Elaborated Design**
- **Representation of the design before synthesis**
  - Interconnected netlist of hierarchical and generic technology cells
    - Instances of modules/entities
    - Generic technology representations of hardware components





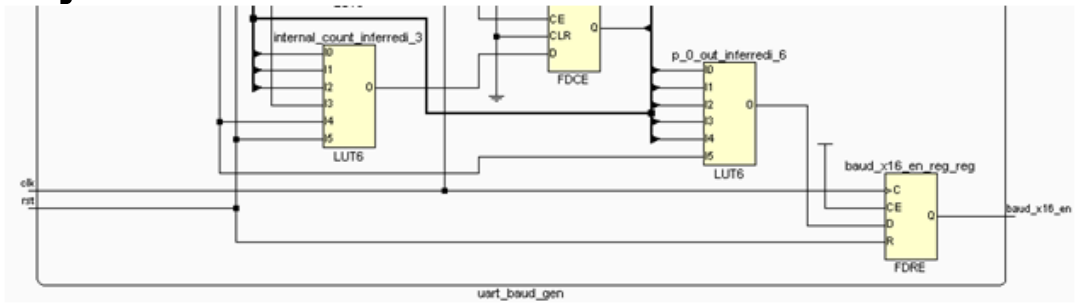
Object Names in Elaborated Design

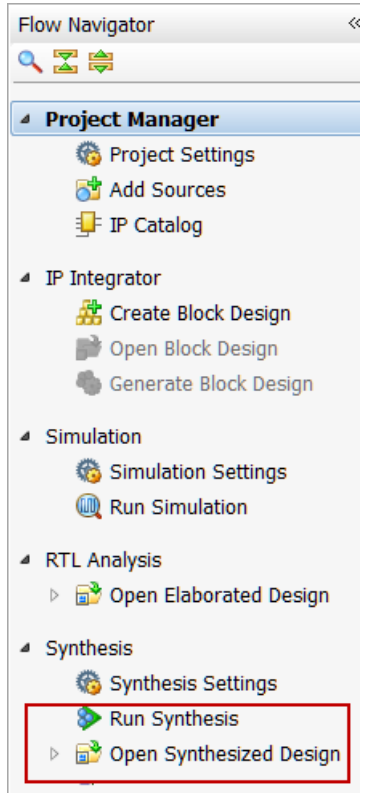
- **Object names are extracted from RTL**
  - Instance and pin names of hierarchical objects
  - Inferred flip-flops from underlying reg/signal/logic
    - Suffix _reg is added
  - Nets from underlying reg/signal/logic when it makes sense



Synthesized Design

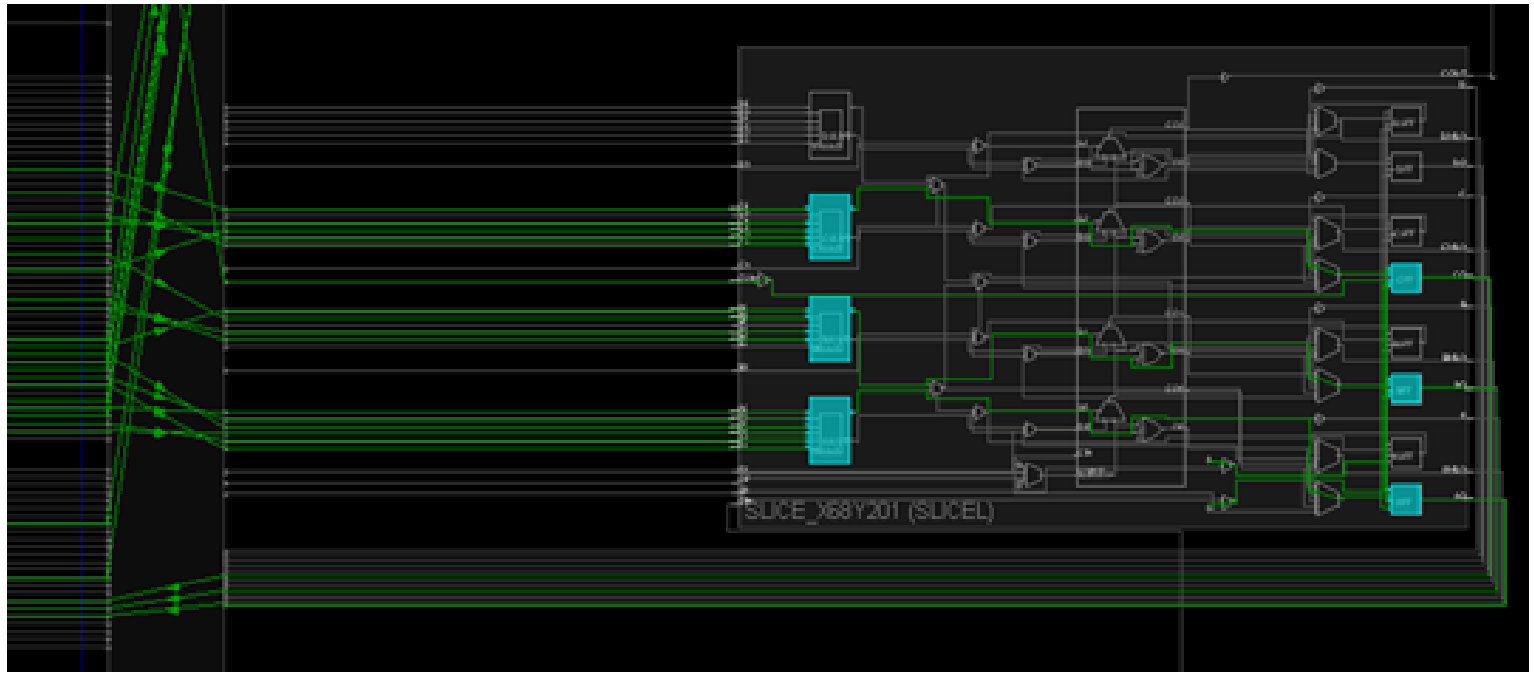
- **Accessed through the Flow Navigator by selecting Open Synthesized Design**
- **Representation of the design after synthesis**
  - Interconnected netlist of hierarchical and basic elements (BELs)
    - Instances of modules/entities
    - Basic elements
      - LUTs, flip-flops, carry chain elements, wide 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**

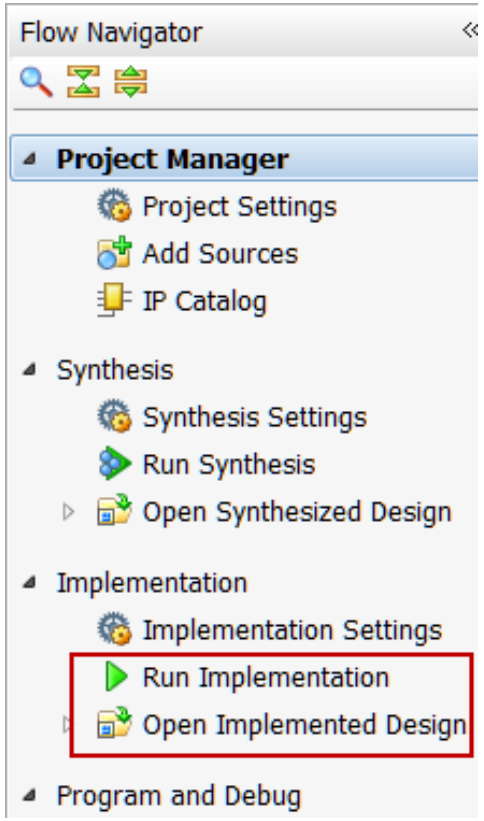




Implemented Design

- **Accessed through the Flow Navigator by selecting Open 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**
  - *project_name.xpr* file: Object that is selected to open a project (Vivado IDE project file)
  - *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

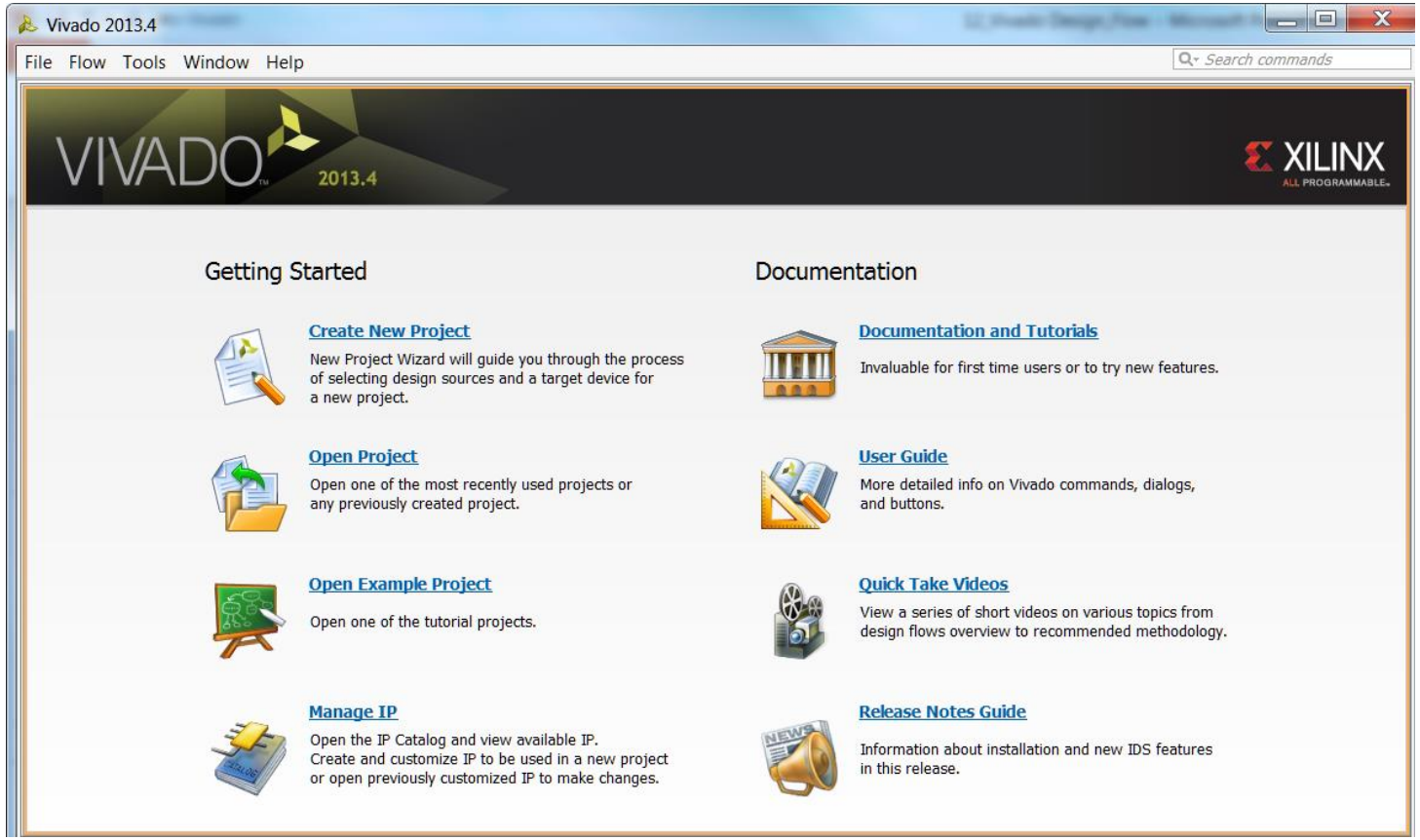
- **Journal file (*vivado.jou*)**
  - Contains just the Tcl commands executed by the Vivado IDE
- **Log file (*vivado.log*)**
  - Contains all messages produced by the Vivado IDE, including Tcl commands and results, info/warning/error messages, etc.
- **Location**
  - Linux: directory where the Vivado 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

Outline

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

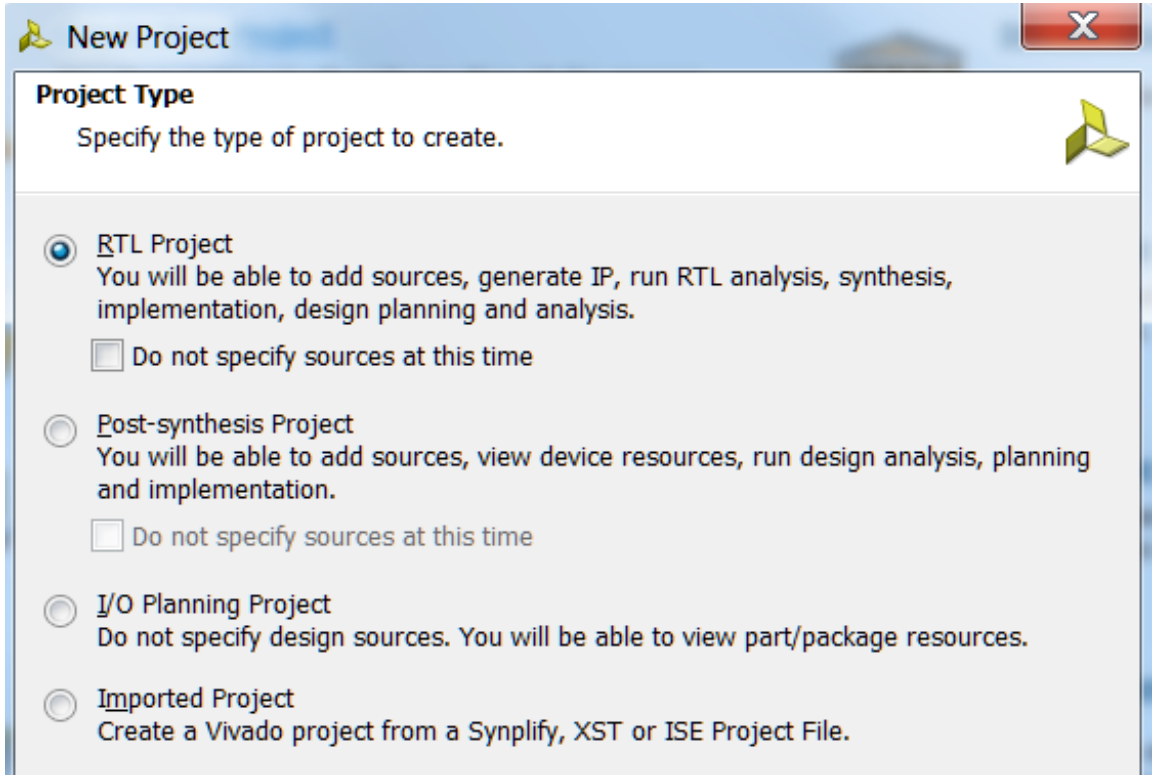
Getting Started Jump Page

- **Links to create new or open existing projects**
- **Open example projects**
  - Used for tutorials
- **Documentation links**
  - Invokes PDF viewer for
    - Release notes
    - User guide
    - Methodology guide
    - Tutorials



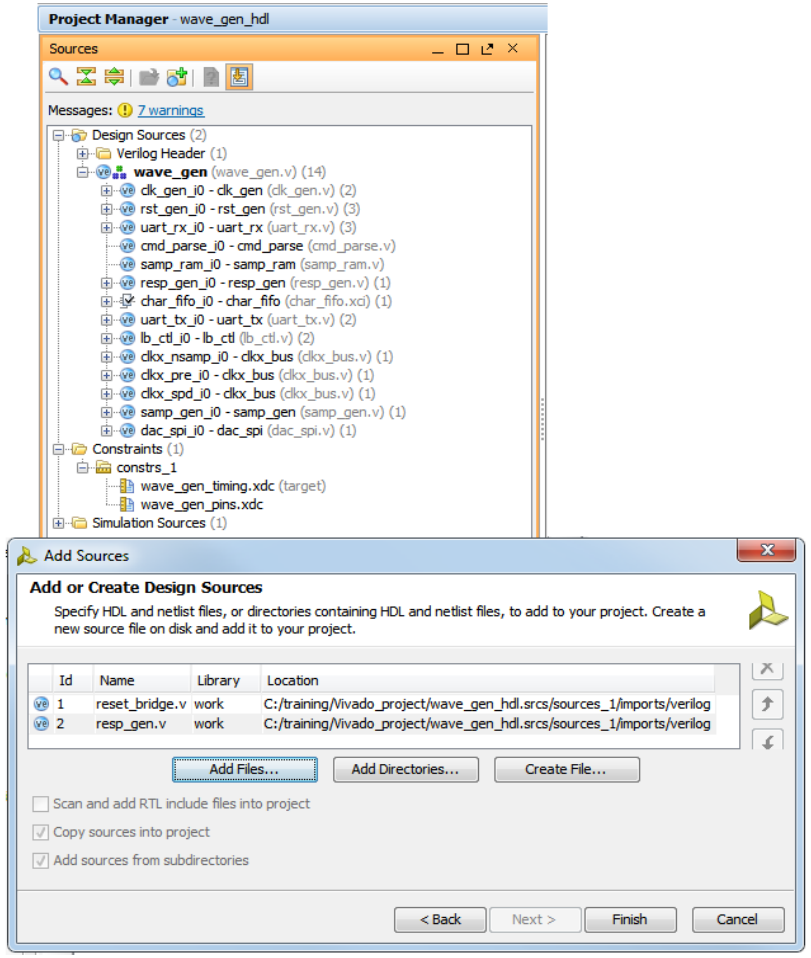
New Project Creation Wizard

- **Four different types of project can be created**
  - RTL
    - Front-to-back
  - Post-synthesis
    - EDIF or NGC
  - I/O planning
    - For early pin testing
    - No design sources
  - Import Project existing project from Project Navigator, XST, and Synplify Projects
- **Next, add source files, constraints files, and select an FPGA**



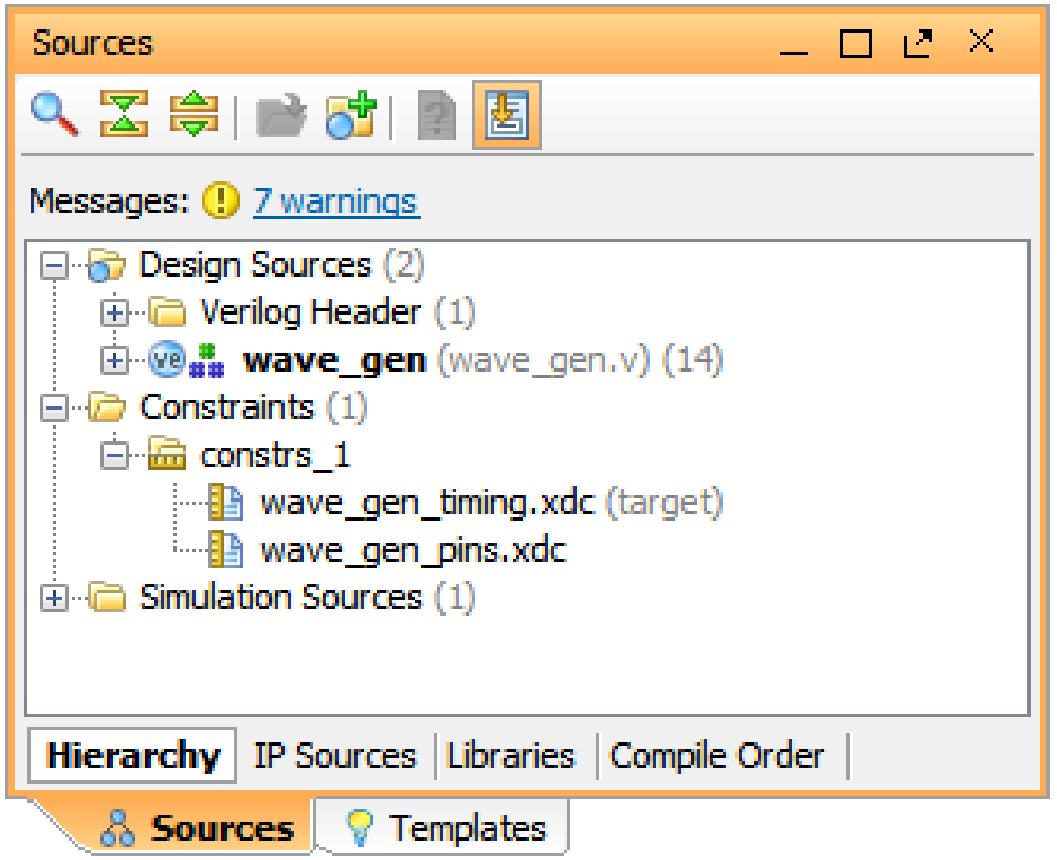
Creating a Project with HDL Sources and/or Synthesized Design

- **Define 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 the family and target device, or pre-defined board**
- **Reference original existing files or import and copy them into the project**



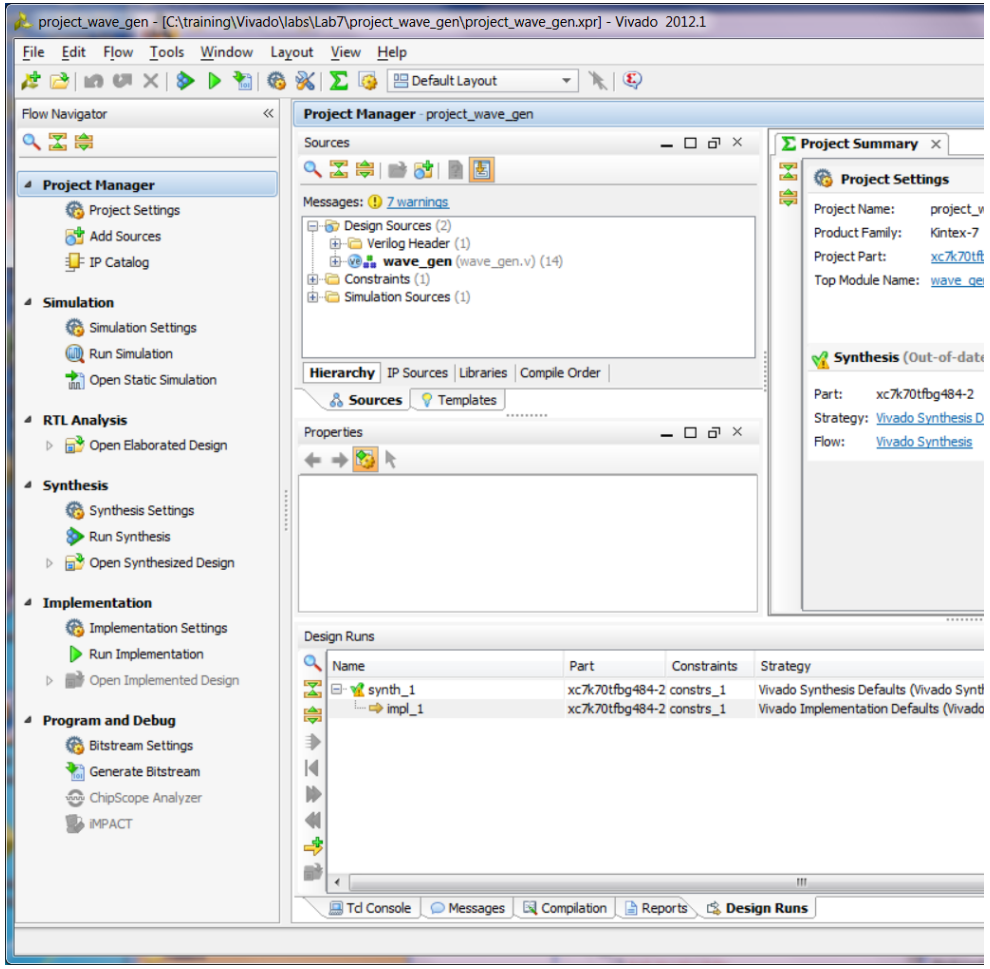
Constraints File Management

- **Constraint sets are a collection of XDC files**
  - A project can contain multiple constraint sets but all must be of the same type
  - 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
  - Modified existing constraints are written back to the original constraint file



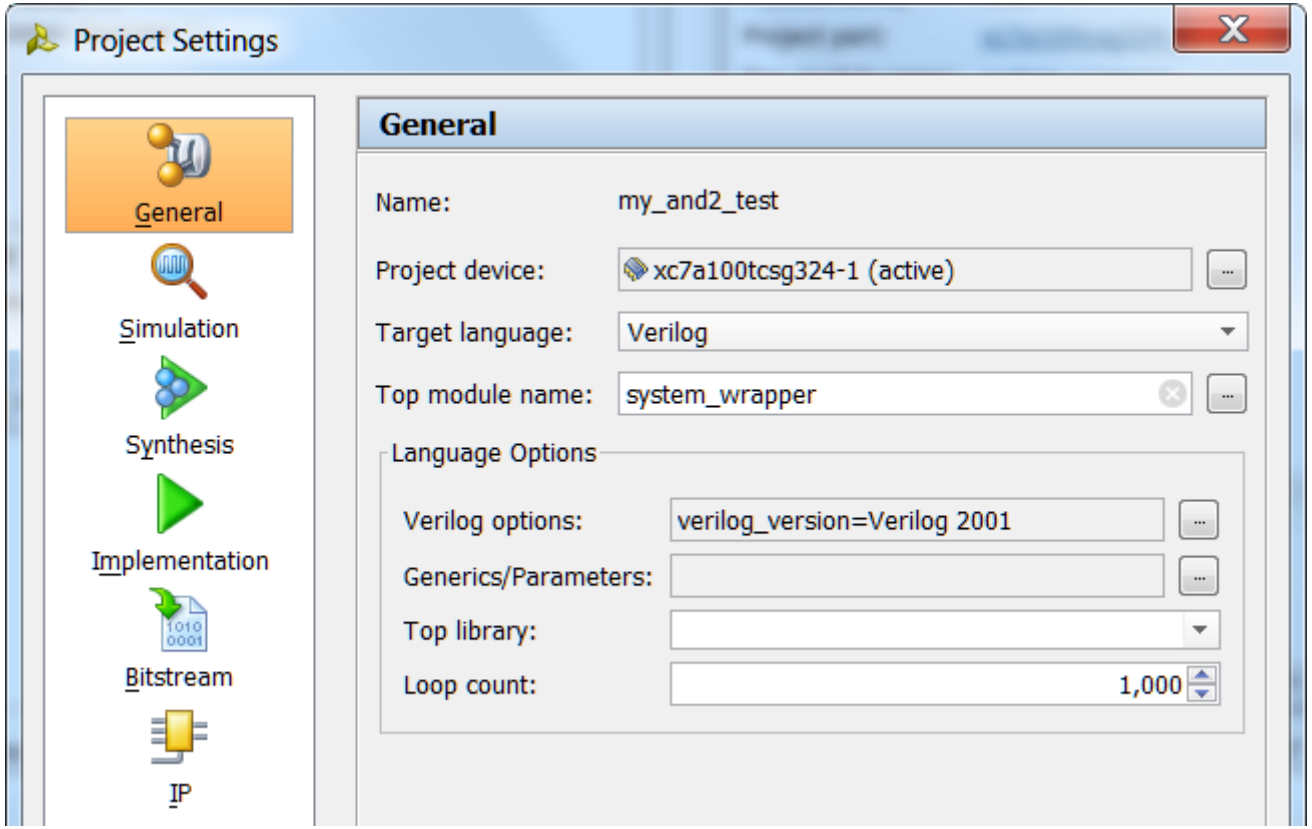
Project Manager

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



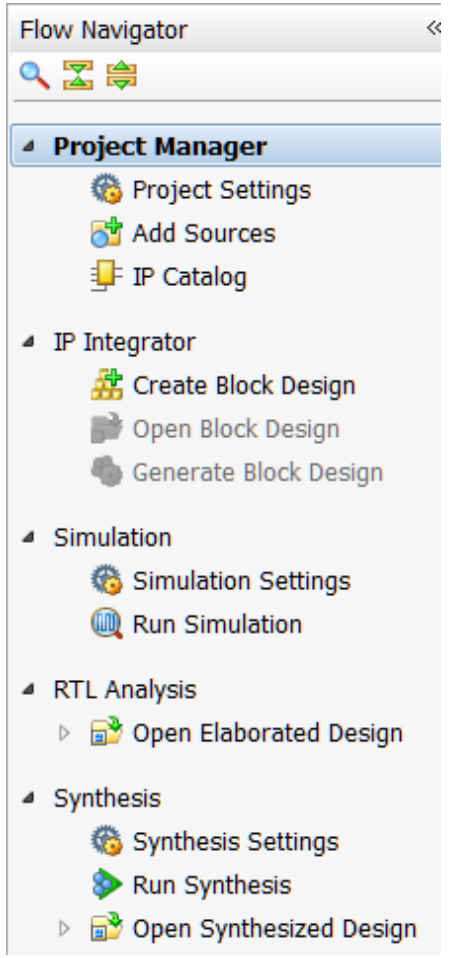
Project Settings

- **General settings**
  - Select device
  - Target HDL language
  - Top module name
  - Language options
- **Other settings are covered in their respective modules**



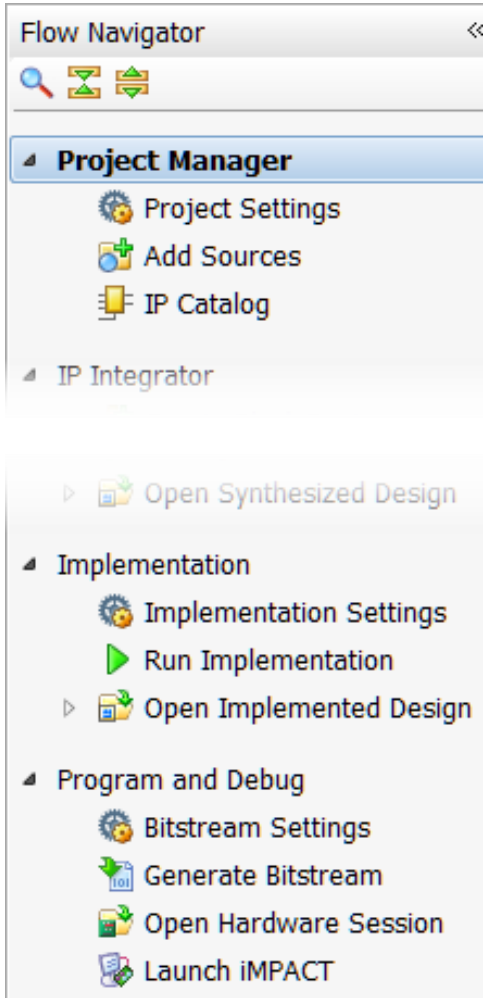
Flow Navigator – RTL Project

- **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
- **Run Synthesis**
  - Timing driven
  - Open Synthesized Design button: Loads synthesized netlist



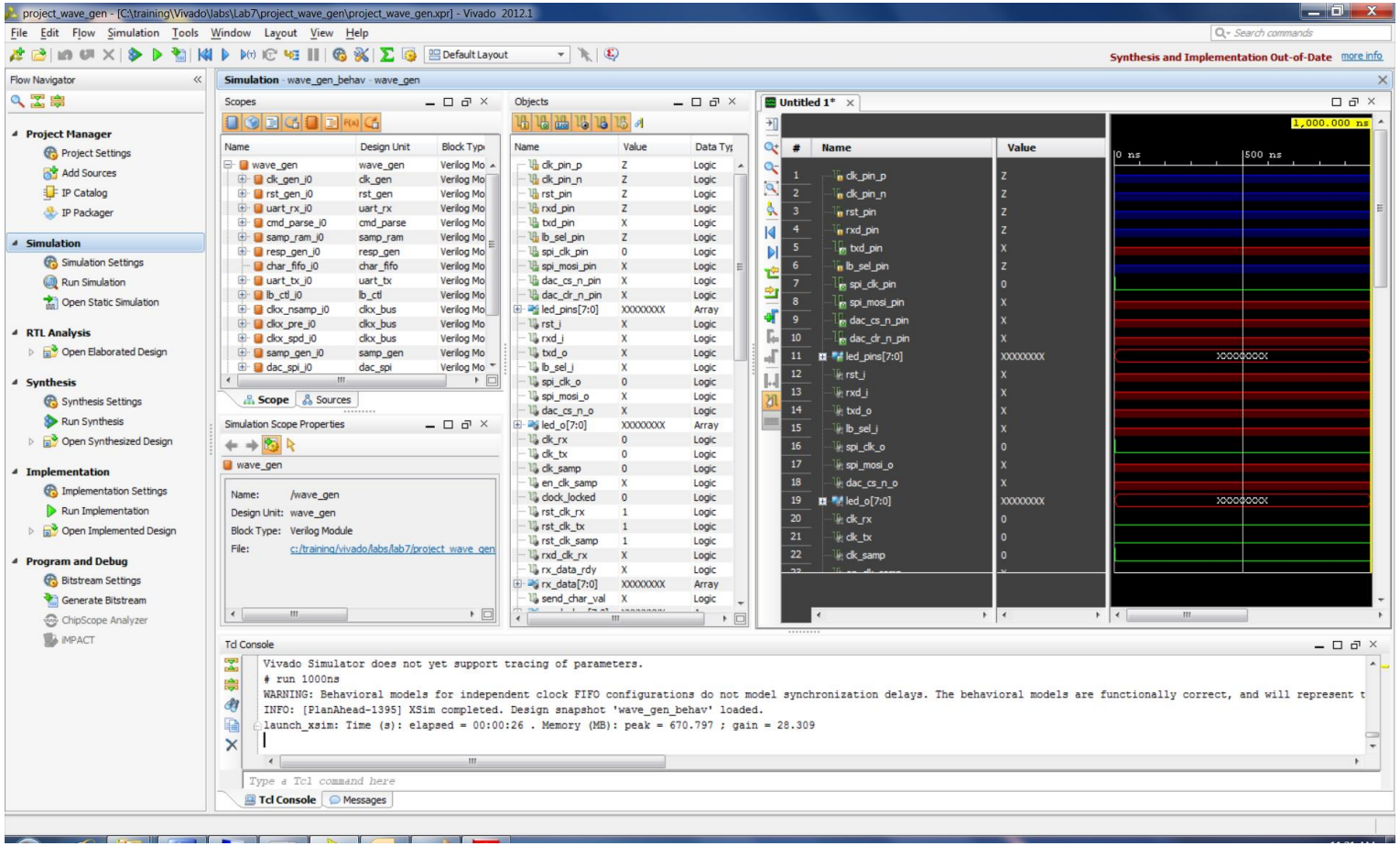
Flow Navigator – RTL Project

- **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 session to program the FPGA
  - Also can use iMPACT program if installed to program the FPGA



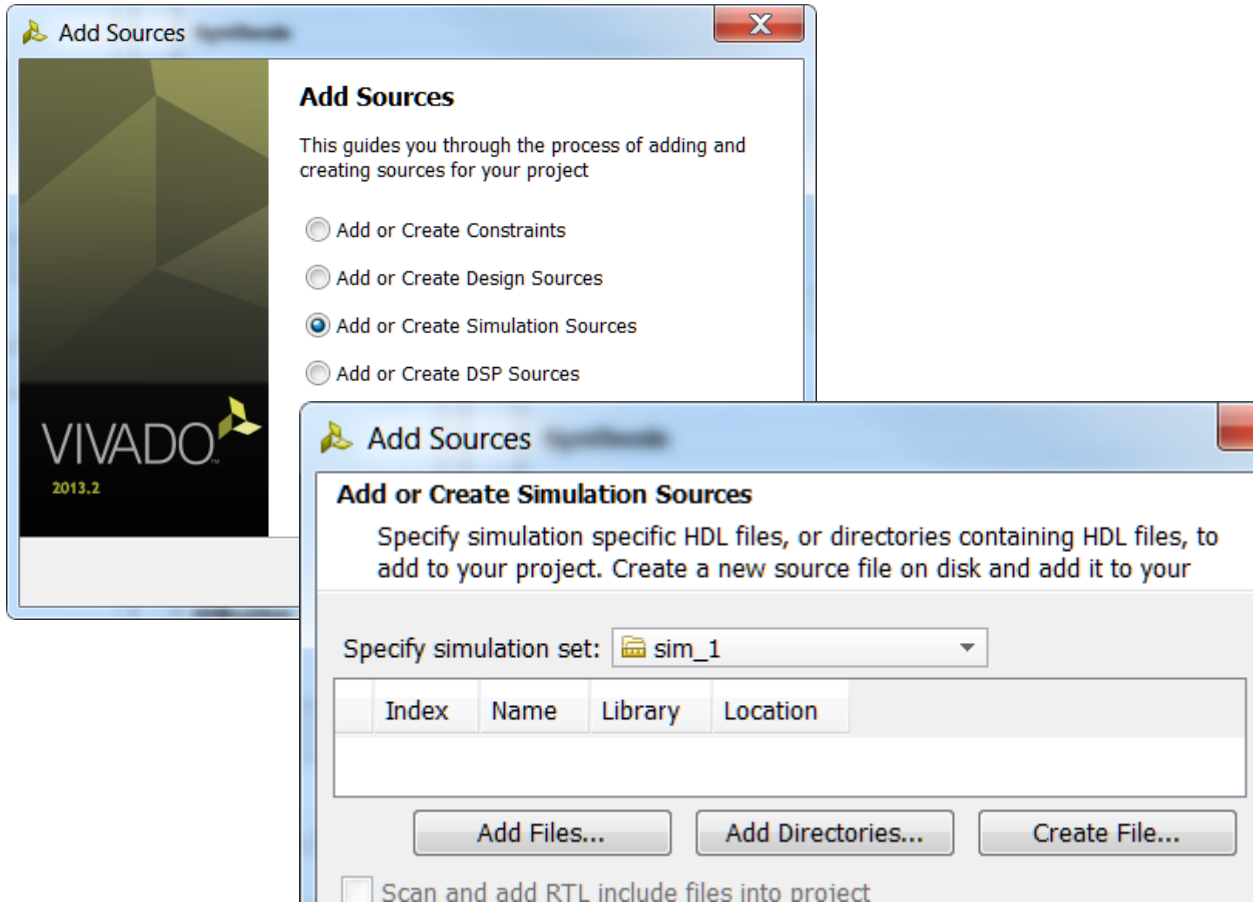
Simulation

- The Vivado simulator, XSIM, supports RTL, netlist, and timing simulation
- 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
- By default, the top-level signals are displayed



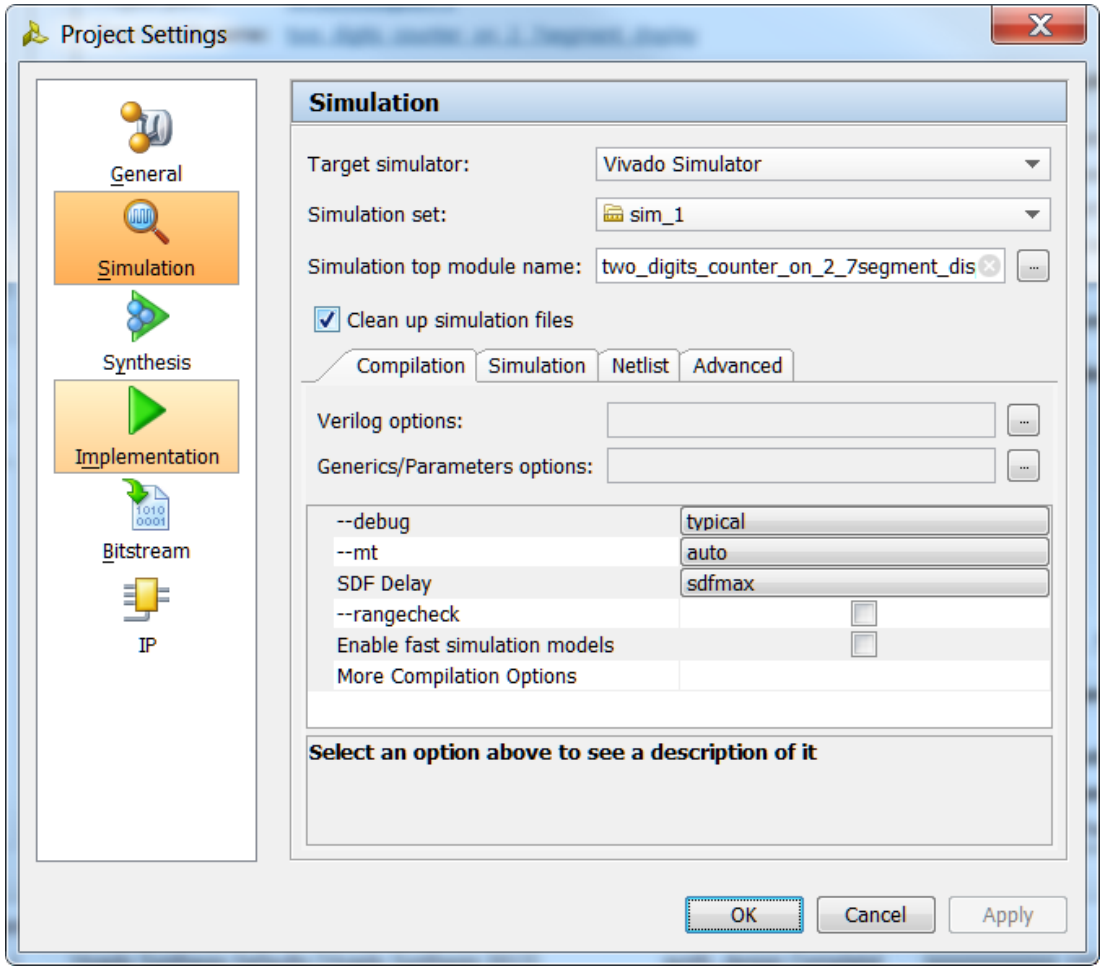
Adding Testbench Files

- In the Flow Navigator, click Add Sources
- Select Add or Create Simulation Sources option and click Next
- Click on Add Files…, Add directories… buttons 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

- From the Flow Navigator, click Simulation Settings
- Allows selection of compilation and simulation properties
- Additional options can be entered via More Compilation Options field in Compilation tab and More Simulation Options field in Simulation tab
- Refer to the *Vivado Design Suite Simulation Guide* (UG900) for more information



Outline

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

Summary

- **Features and benefits of the Vivado IDE include**
  - 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**