Vivado Design Flow

XILINX.

Objectives

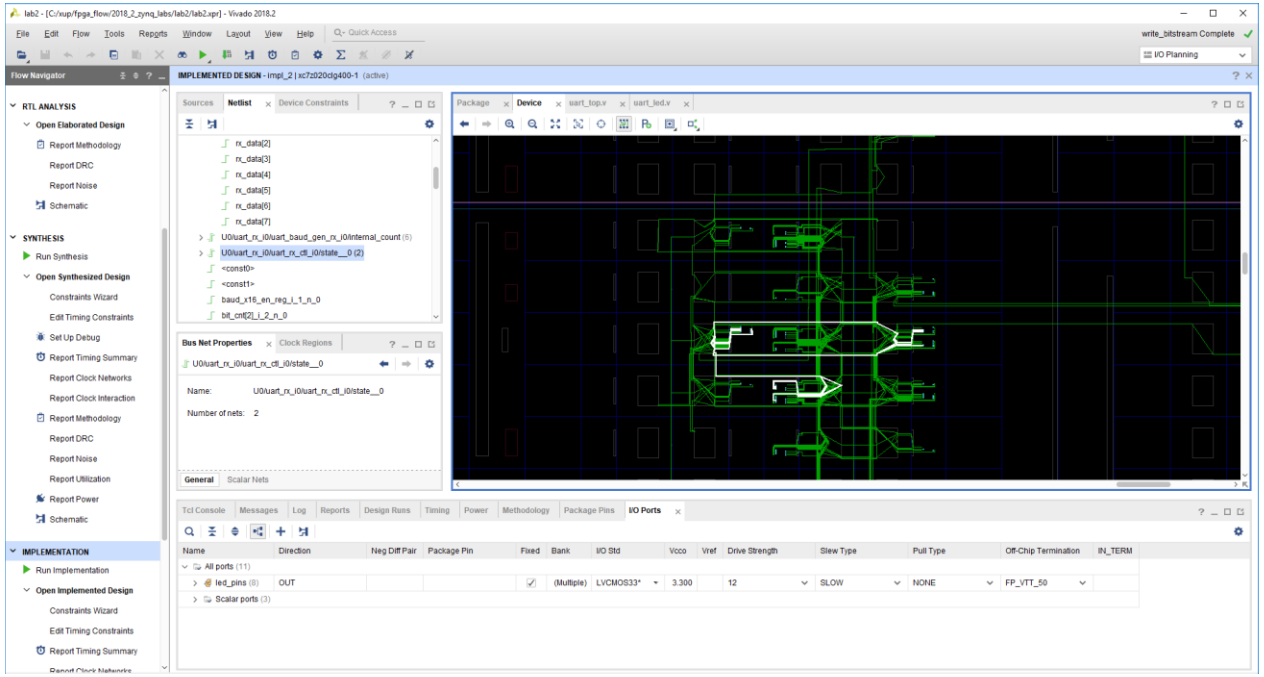
- **After completing this module, you will be able to:**
  - Explain how the design analysis features of the Vivado IDE can help in FPGA design development
  - List the main features of the Vivado IDE
  - Describe the Vivado IDE Design flow
  - Introduce the scripted Vivado IDE design flows

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



lab2 - [C:/xup/fpga_flow/2018_2_zynq_labs/lab2/lab2.xpr] - Vivado 2018.2

write_bitstream Complete ✓

IMPLEMENTED DESIGN - impl_2 | xc7z020clg400-1 (active)

rx_data[2]
rx_data[3]
rx_data[4]
rx_data[5]
rx_data[6]
rx_data[7]
U0/uart_rx_i0/uart_baud_gen_rx_i0/internal_count (6)
U0/uart_rx_i0/uart_rx_ctl_i0/state__0 (2)
<const0>
<const1>
baud_x16_en_reg_i_1_n_0
bit_cnt[2]_i_2_n_0

U0/uart_rx_i0/uart_rx_ctl_i0/state__0

Name: U0/uart_rx_i0/uart_rx_ctl_i0/state__0

Number of nets: 2

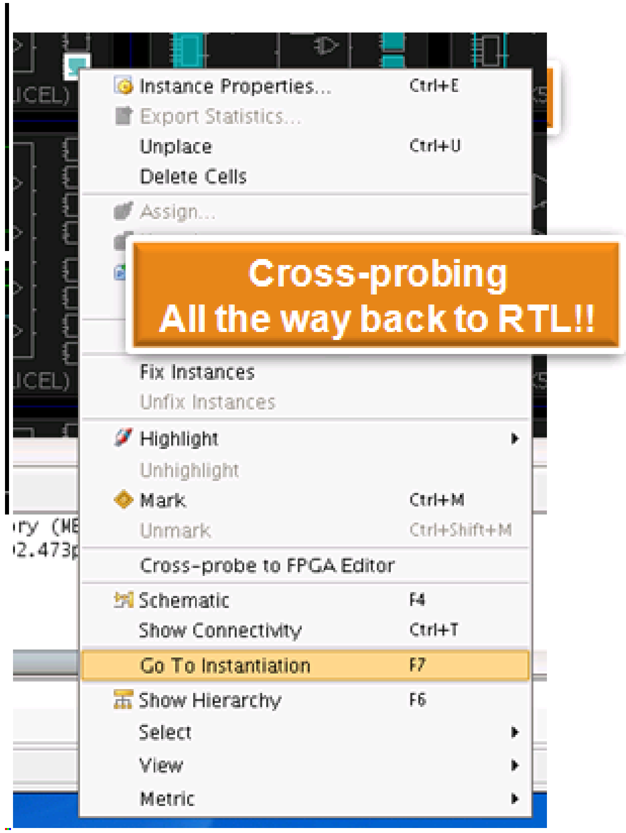
Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination	IN_TERM
All ports (11)													
led_pins (8)	OUT			✓	(Multiple)	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
Scalar ports (3)													

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**

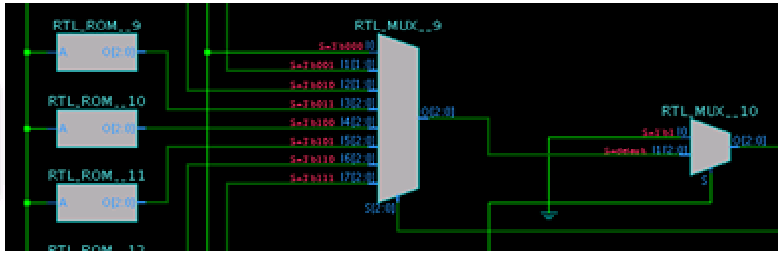
Vivado’s Visualization Feature

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

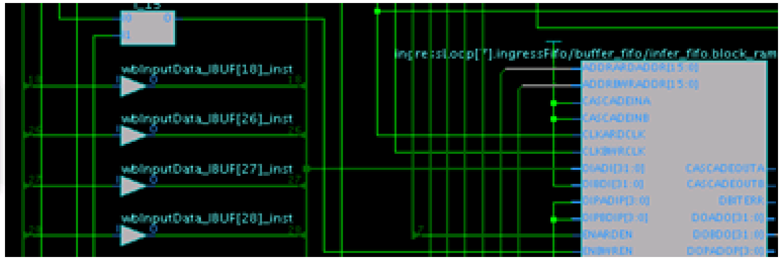


Cross-probing
All the way back to RTL!!

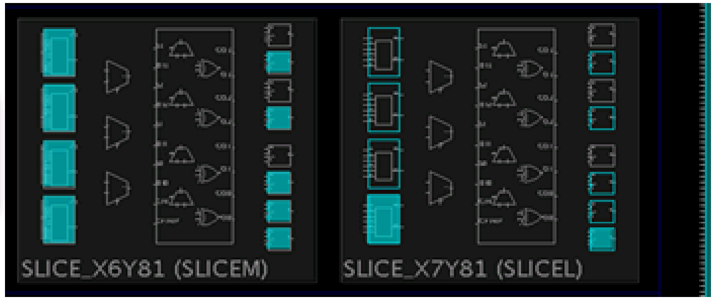
Elaborated RTL



Netlist Schematic

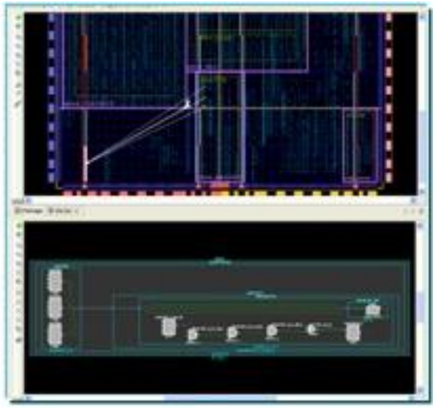


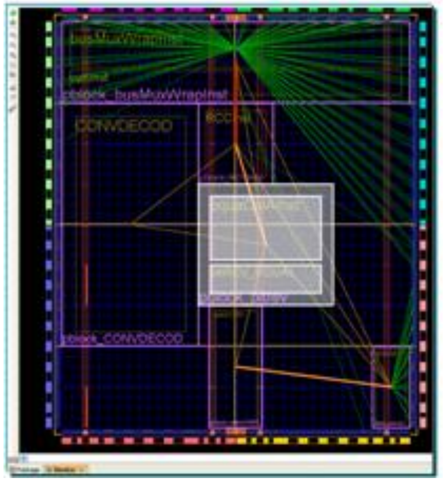
Implemented Design



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



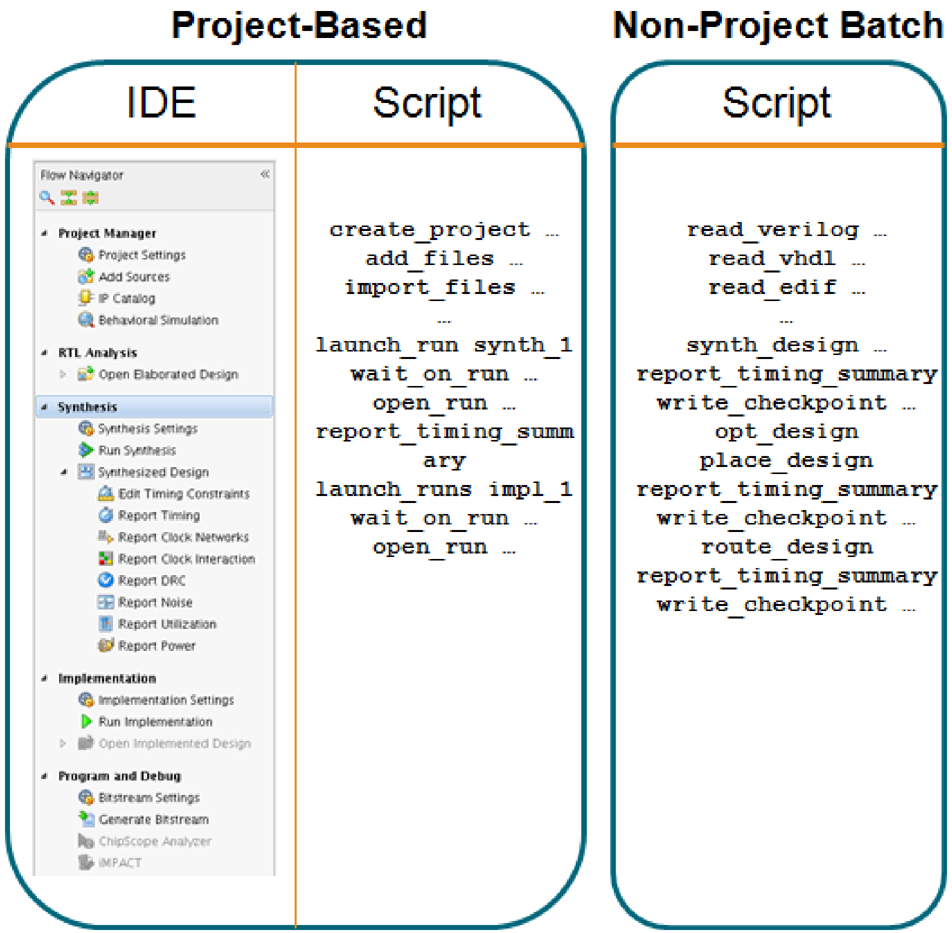


Tcl Capability

- **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



Project-Based Script:

```
create_project …
add_files …
import_files …
…
launch_run synth_1
wait_on_run …
open_run …
report_timing_summ
ary
launch_runs impl_1
wait_on_run …
open_run …
```

Non-Project Batch Script:

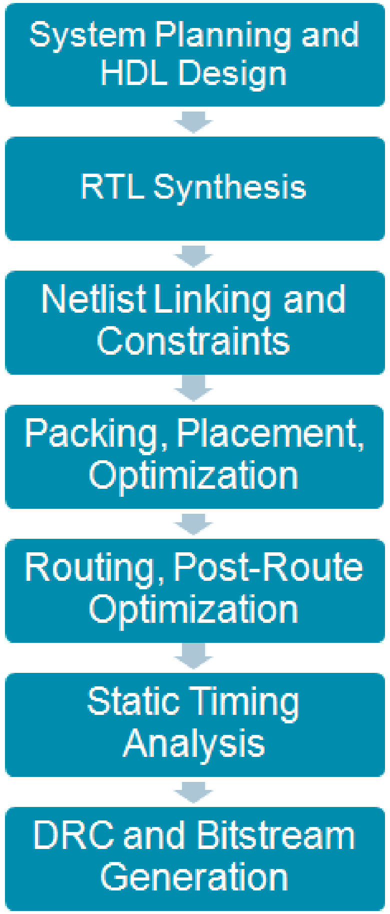
```
read_verilog …
read_vhdl …
read_edif …
…
synth_design …
report_timing_summary
write_checkpoint …
opt_design
place_design
report_timing_summary
write_checkpoint …
route_design
report_timing_summary
write_checkpoint …
```

Vivado Design Suite Introduction

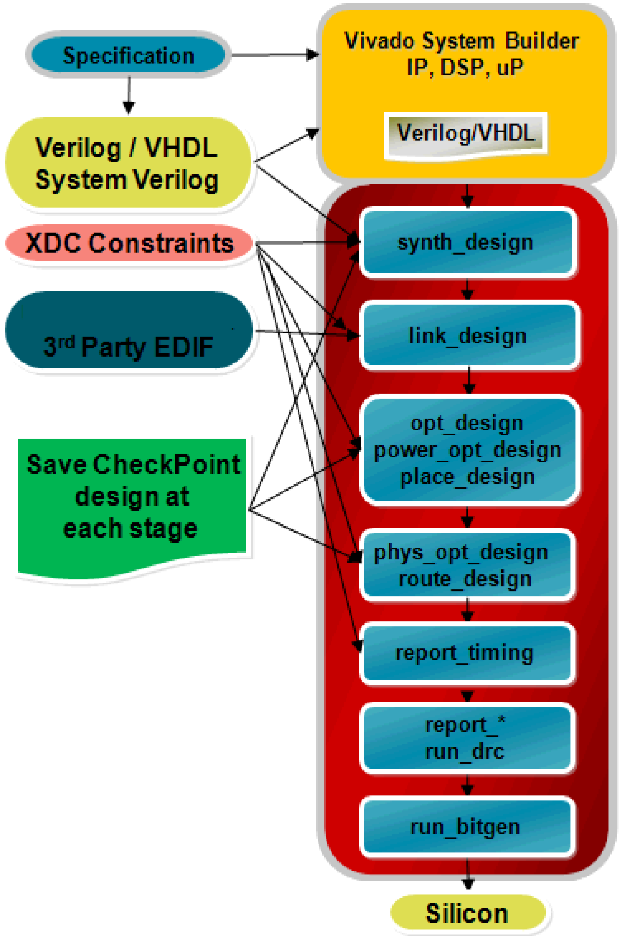
XILINX.

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



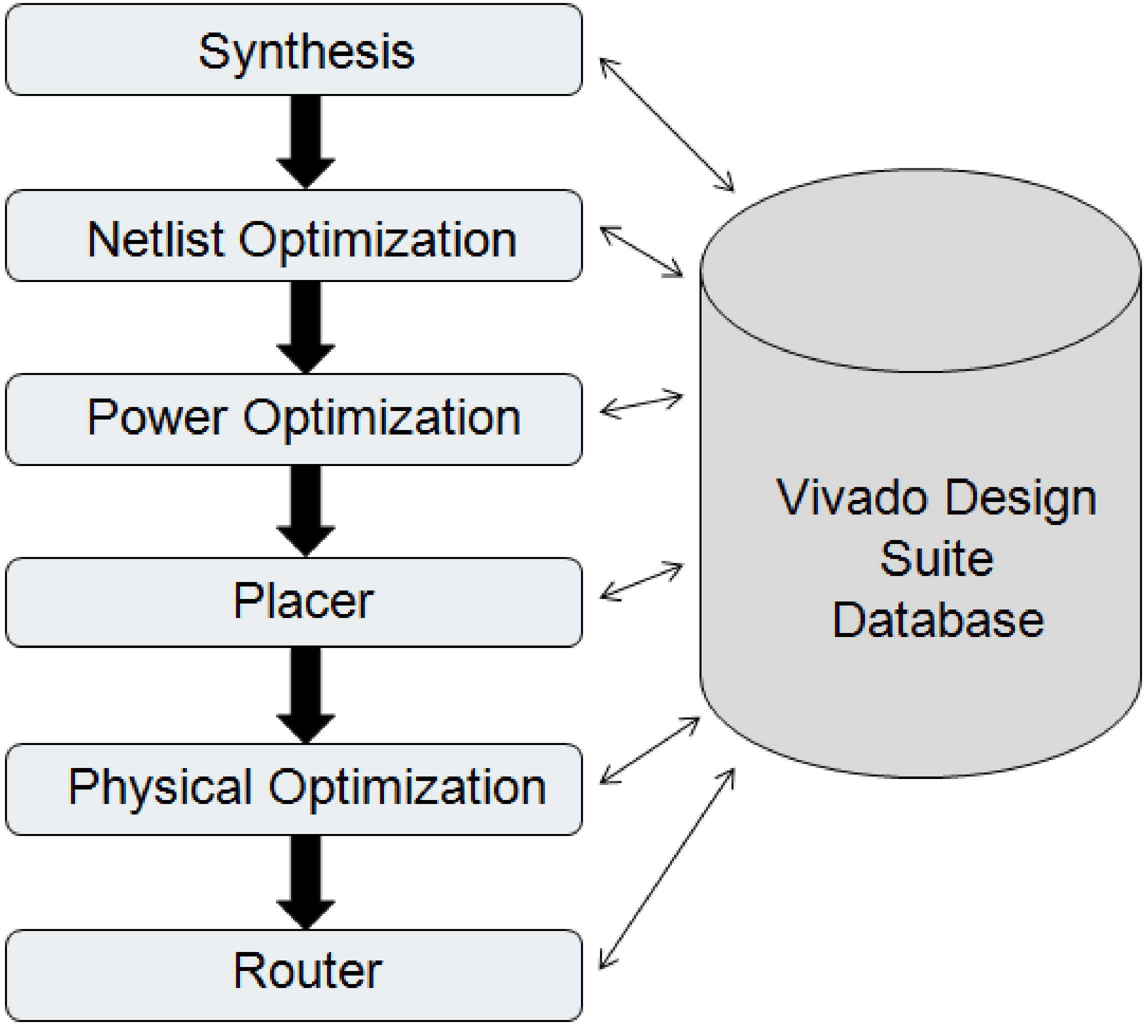
System Planning and HDL Design
RTL Synthesis
Netlist Linking and Constraints
Packing, Placement, Optimization
Routing, Post-Route Optimization
Static Timing Analysis
DRC and Bitstream Generation



XILINX.

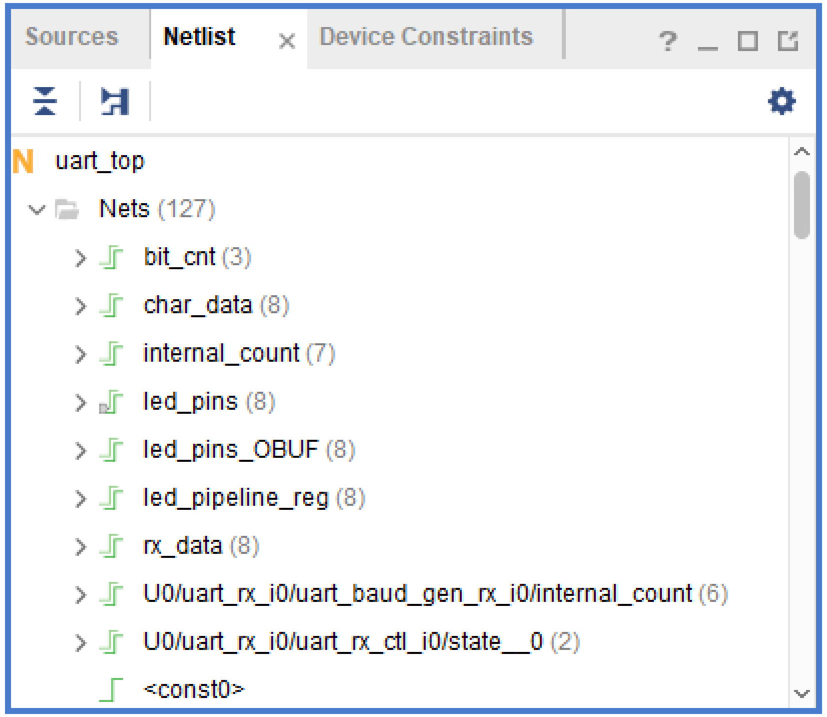
Design Database

- **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



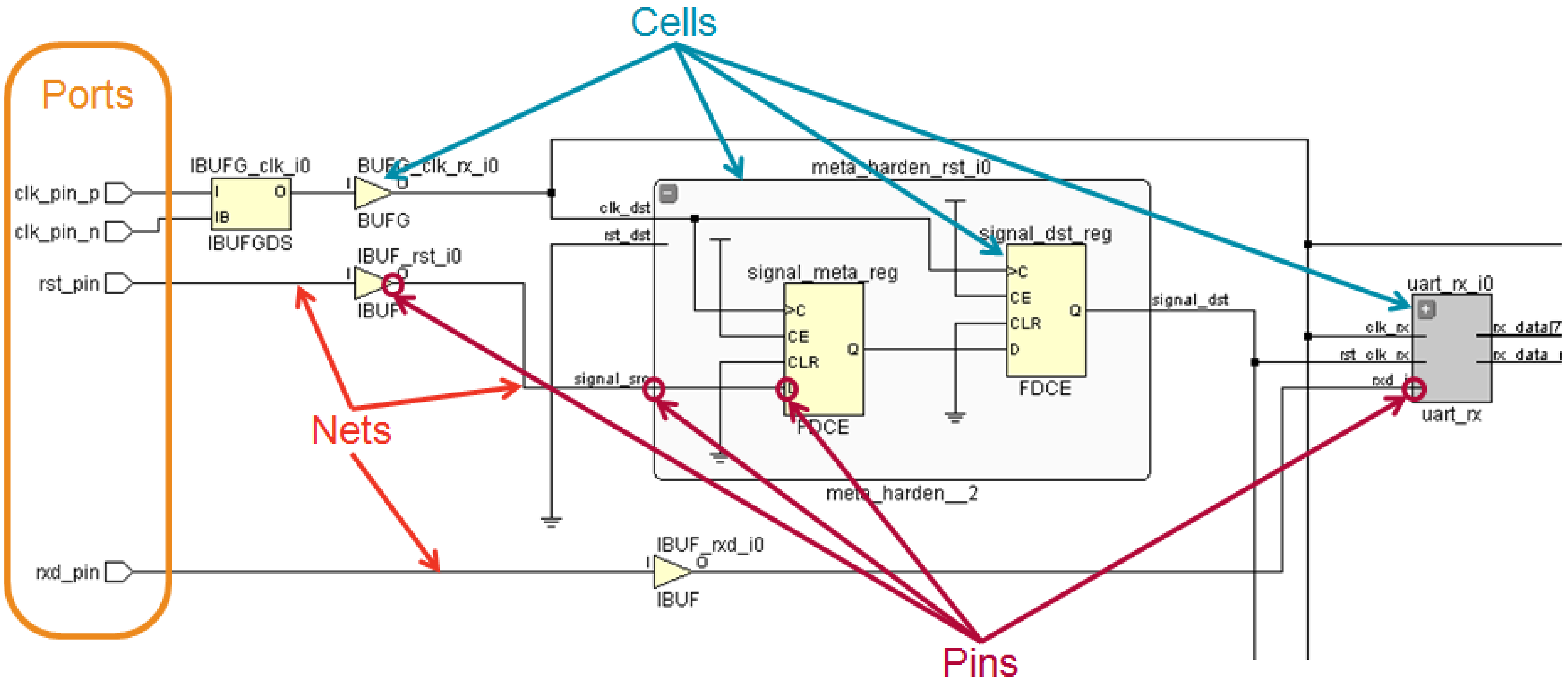
What is a Netlist?

- **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



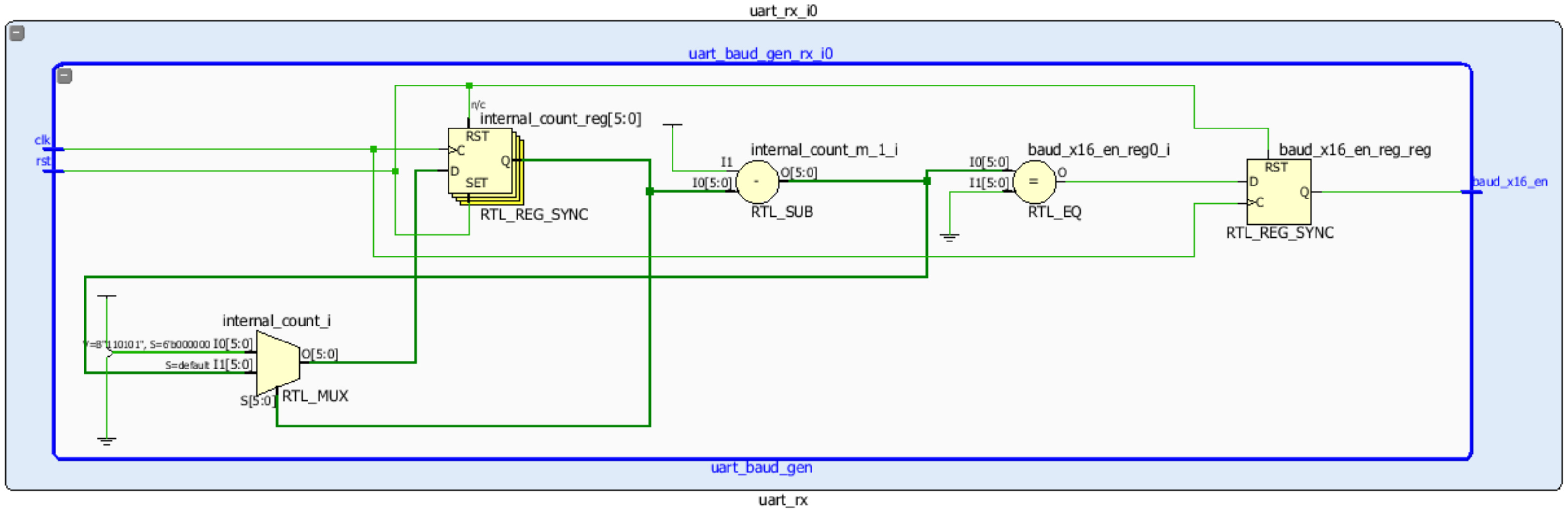
uart_top
- Nets (127)
  - bit_cnt (3)
  - char_data (8)
  - internal_count (7)
  - led_pins (8)
  - led_pins_OBUF (8)
  - led_pipeline_reg (8)
  - rx_data (8)
  - U0/uart_rx_i0/uart_baud_gen_rx_i0/internal_count (6)
  - U0/uart_rx_i0/uart_rx_ctl_i0/state__0 (2)
  - <const0>

Netlist Objects



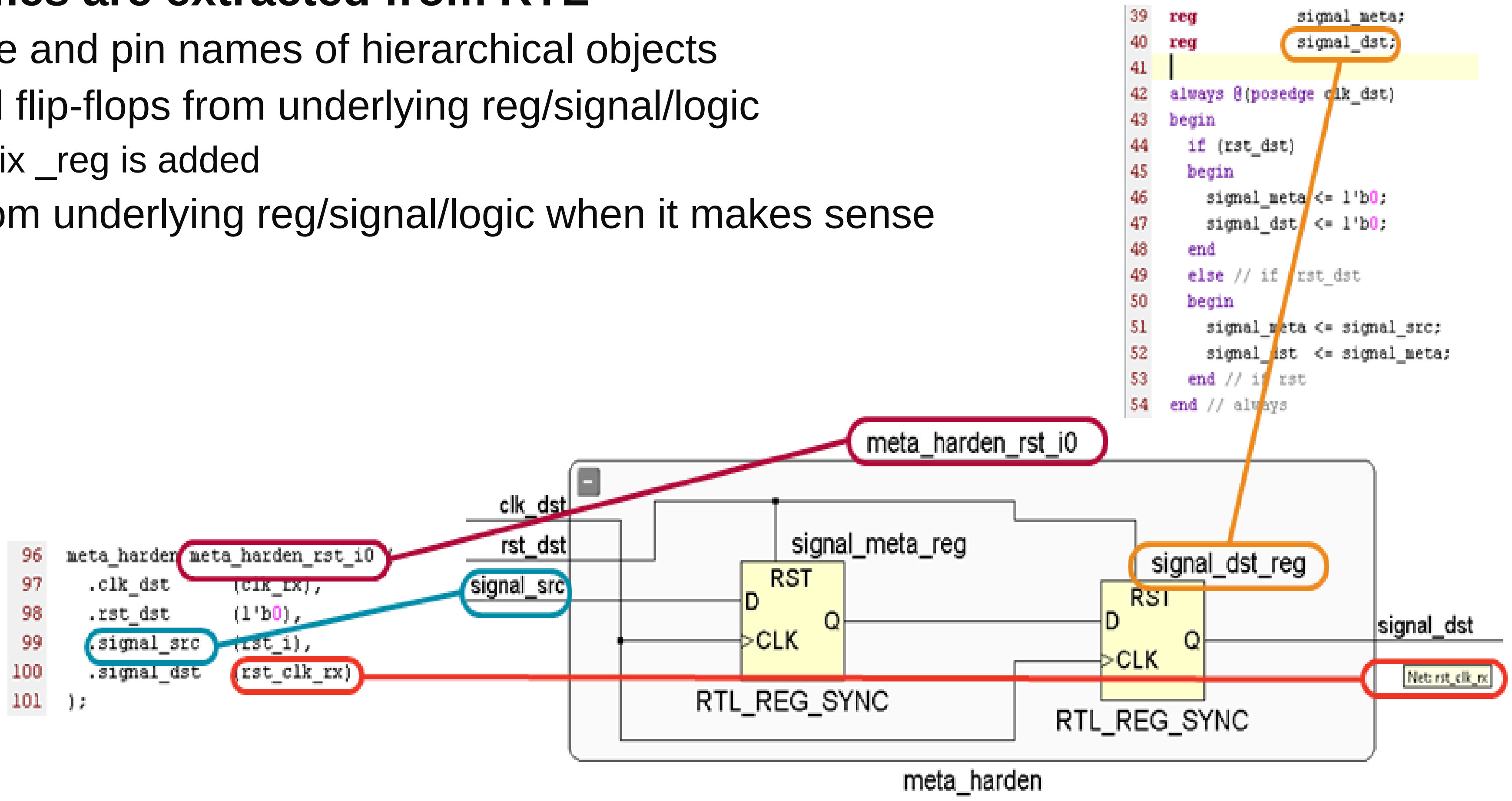
Elaborated Design

- **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



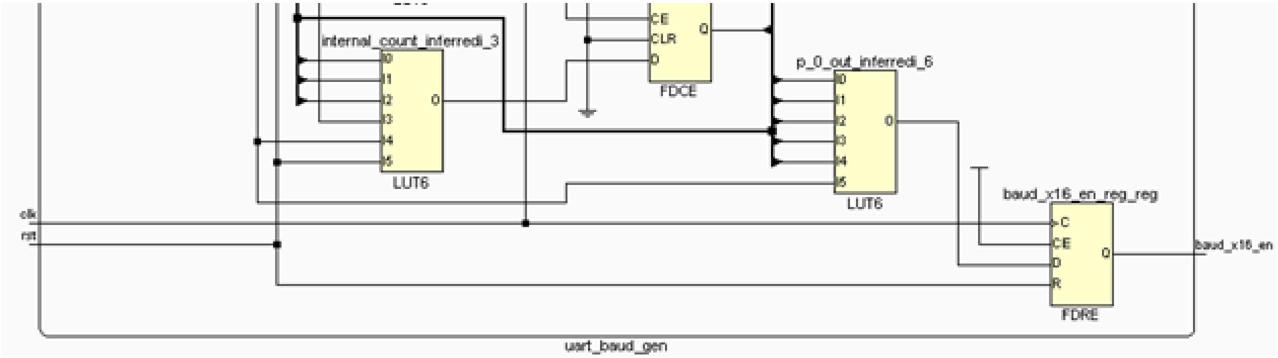
```
reg             signal_meta;
reg             signal_dst;

always @(posedge clk_dst)
begin
  if (rst_dst)
  begin
    signal_meta <= 1'b0;
    signal_dst  <= 1'b0;
  end
  else // if rst_dst
  begin
    signal_meta <= signal_src;
    signal_dst  <= signal_meta;
  end // if rst
end // always
```

```
meta_harden meta_harden_rst_i0 (
  .clk_dst      (clk_rx),
  .rst_dst      (1'b0),
  .signal_src   (rst_i),
  .signal_dst   (rst_clk_rx)
);
```

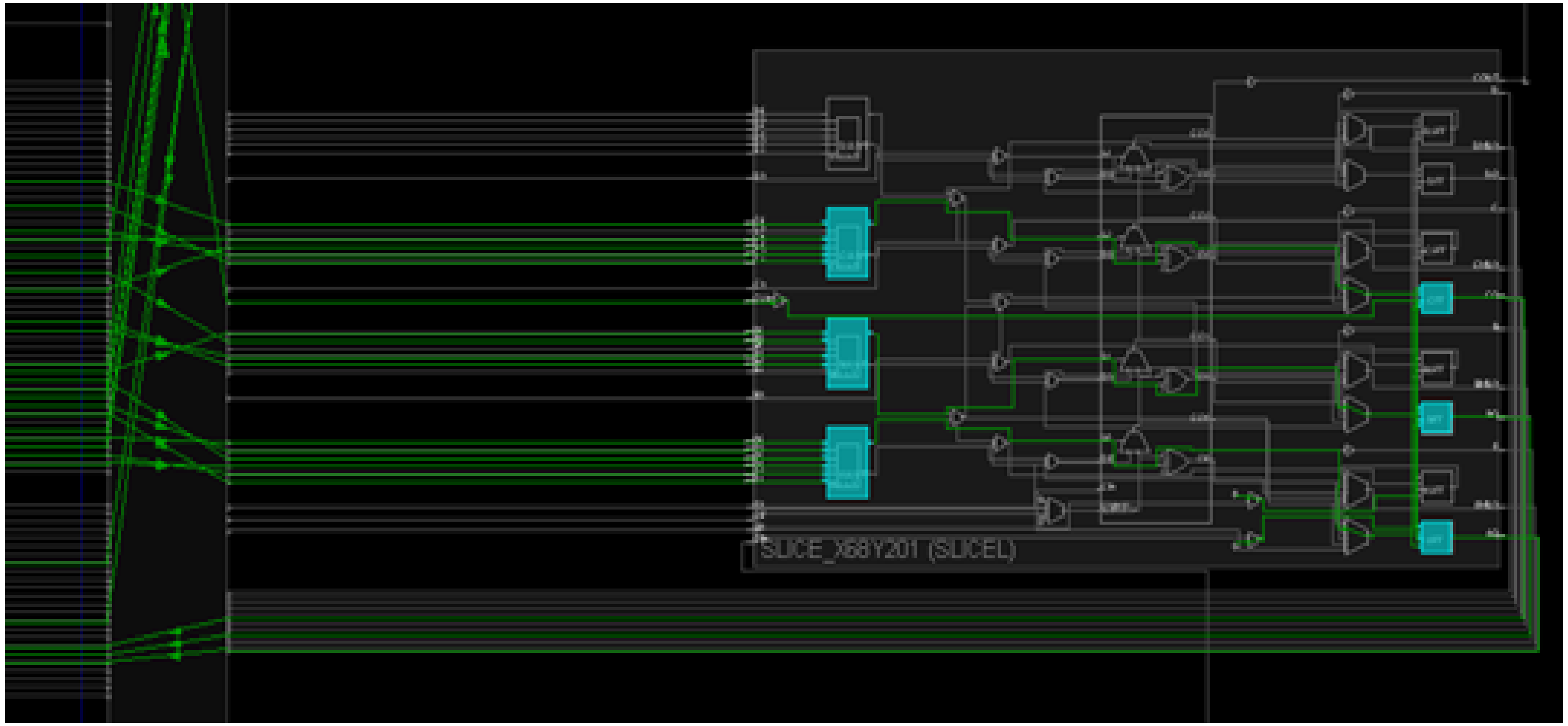
Synthesized Design

- **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>

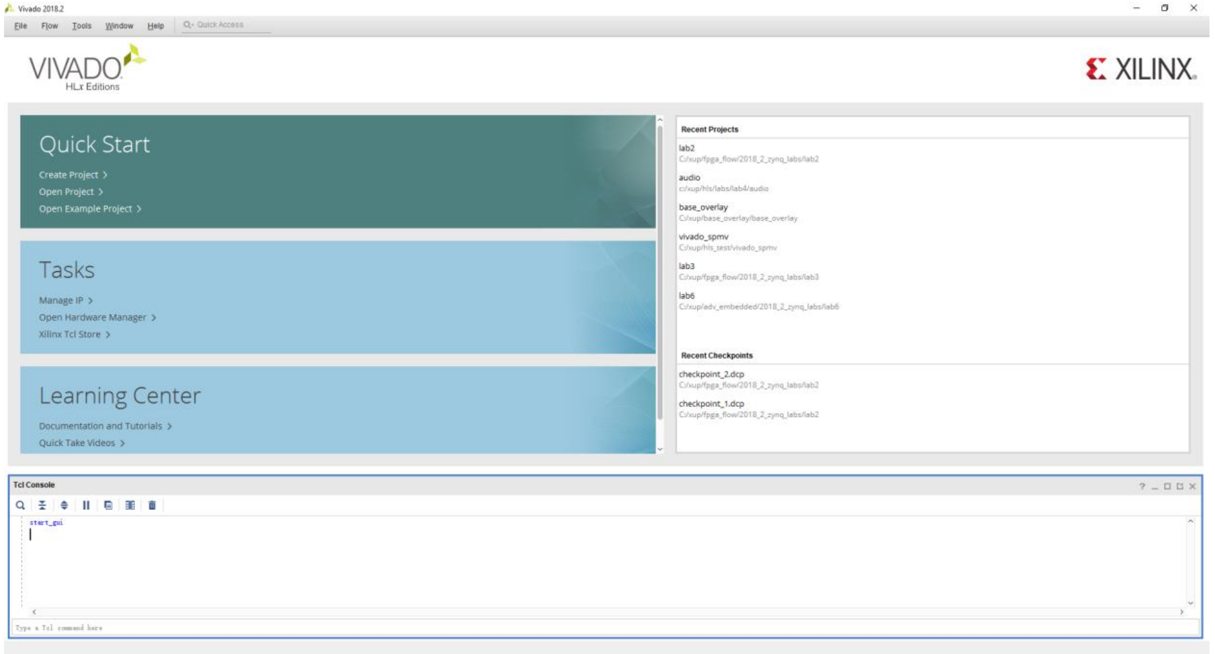
> **Can read and write in the IDE as well**
>> When read in gives a project-less setup

> **Can open and modify checkpoints in IDE**

Vivado Design Flow

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**



Recent Projects

lab2
C:/xup/fpga_flow/2018_2_zynq_labs/lab2

audio
c:/xup/hls/labs/lab4/audio

base_overlay
C:/xup/base_overlay/base_overlay

vivado_spmv
C:/xup/hls_test/vivado_spmv

lab3
C:/xup/fpga_flow/2018_2_zynq_labs/lab3

lab6
C:/xup/adv_embedded/2018_2_zynq_labs/lab6

Recent Checkpoints

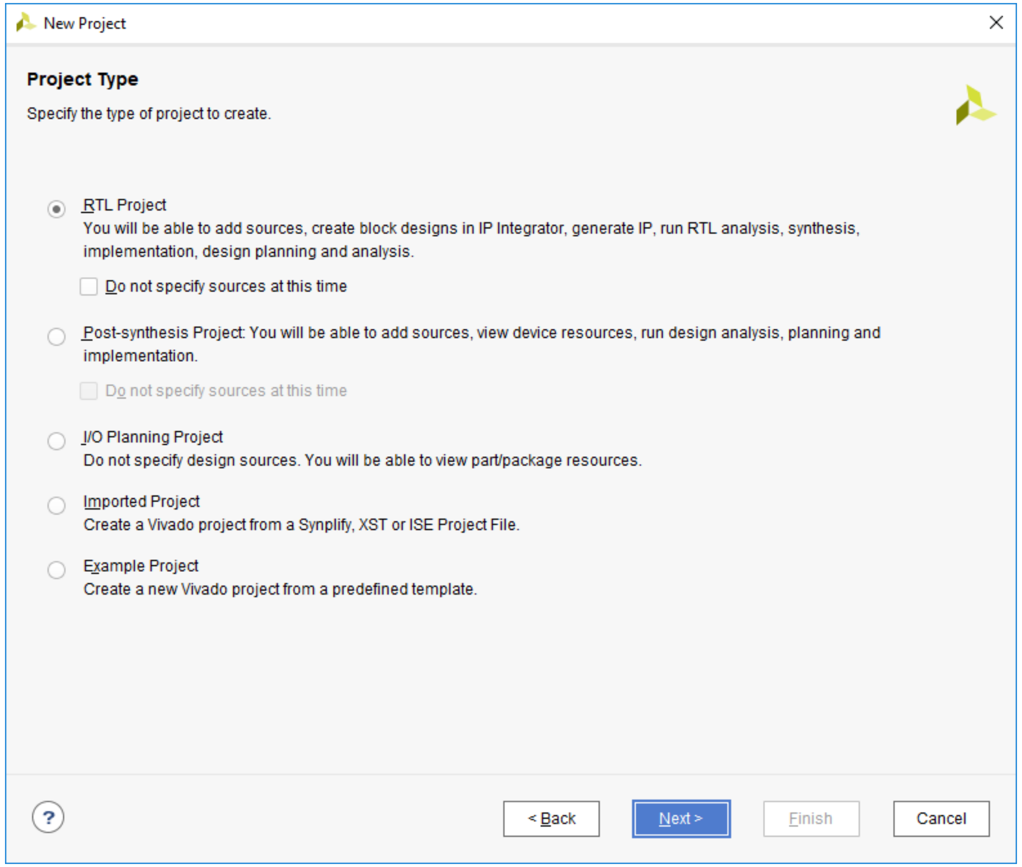
checkpoint_2.dcp
C:/xup/fpga_flow/2018_2_zynq_labs/lab2

checkpoint_1.dcp
C:/xup/fpga_flow/2018_2_zynq_labs/lab2

```
start_gui
```

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 Type

Specify the type of project to create.

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

□ Do not specify sources at this time

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

□ Do not specify sources at this time

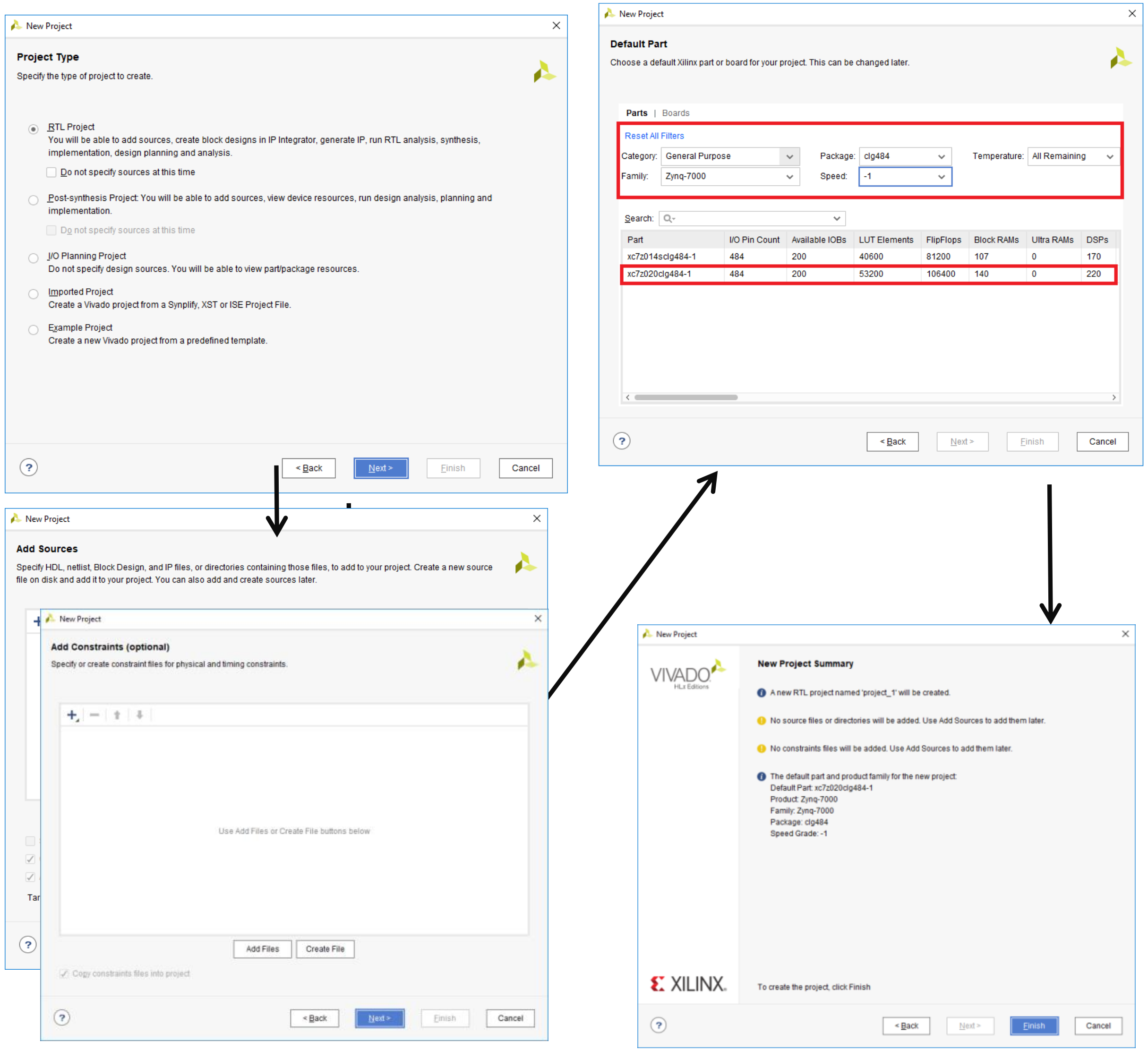
○ I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

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

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

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**



Project Type

Specify the type of project to create.

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

□ Do not specify sources at this time

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

□ Do not specify sources at this time

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

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

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

Add Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

Add Constraints (optional)

Specify or create constraint files for physical and timing constraints.

Use Add Files or Create File buttons below

☑ Copy constraints files into project

Default Part

Choose a default Xilinx part or board for your project. This can be changed later.

Category: General Purpose
Family: Zynq-7000
Package: clg484
Speed: -1
Temperature: All Remaining

Search:

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs
xc7z014sclg484-1	484	200	40600	81200	107	0	170
xc7z020clg484-1	484	200	53200	106400	140	0	220

New Project Summary

A new RTL project named 'project_1' will be created.

No source files or directories will be added. Use Add Sources to add them later.

No constraints files will be added. Use Add Sources to add them later.

The default part and product family for the new project:
Default Part: xc7z020clg484-1
Product: Zynq-7000
Family: Zynq-7000
Package: clg484
Speed Grade: -1

To create the project, click Finish

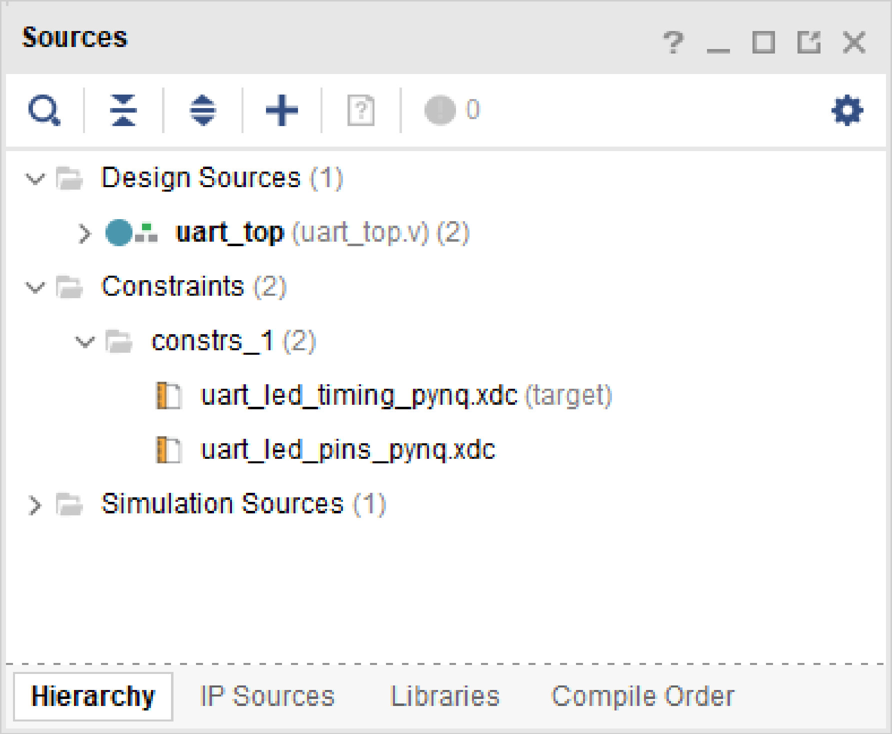
Constraints File Management

> **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

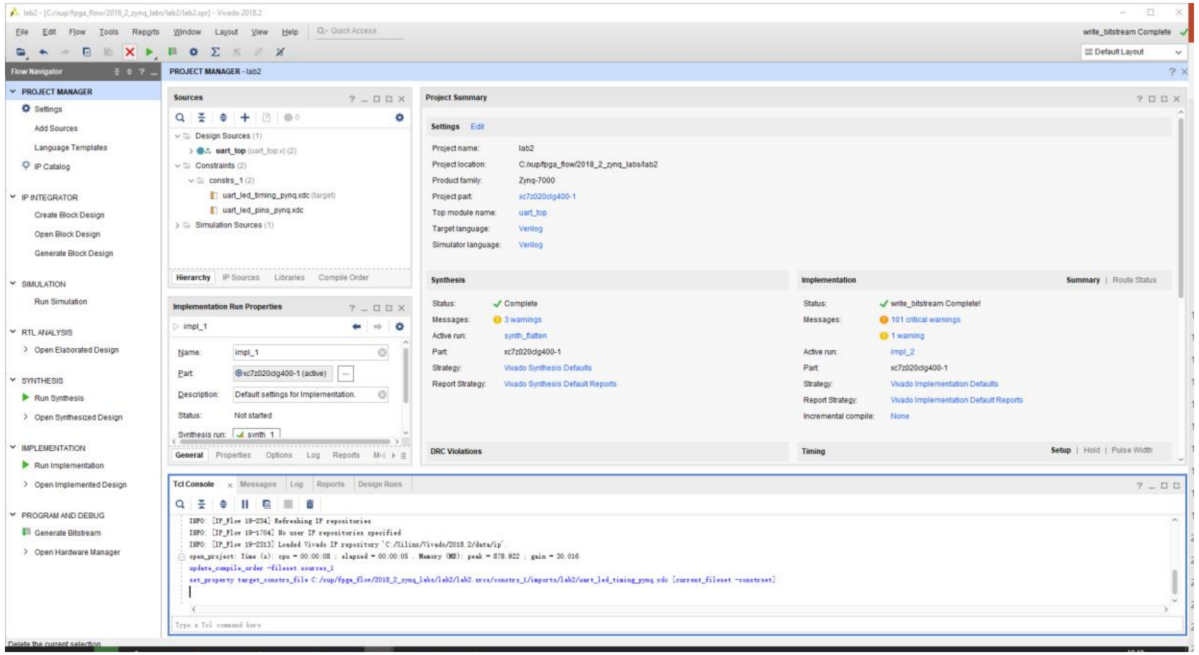


Sources

Design Sources (1)
uart_top (uart_top.v) (2)
Constraints (2)
constrs_1 (2)
uart_led_timing_pynq.xdc (target)
uart_led_pins_pynq.xdc
Simulation Sources (1)

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**



Name: impl_1
Part: xc7z020clg400-1 (active)
Description: Default settings for Implementation.
Status: Not started
Synthesis run: synth_1

Project name: lab2
Project location: C:/xup/fpga_flow/2018_2_zynq_labs/lab2
Product family: Zynq-7000
Project part: xc7z020clg400-1
Top module name: uart_top
Target language: Verilog
Simulator language: Verilog

Synthesis

Status: ✓ Complete
Messages: 3 warnings
Active run: synth_flatten
Part: xc7z020clg400-1
Strategy: Vivado Synthesis Defaults
Report Strategy: Vivado Synthesis Default Reports

Implementation

Status: ✓ write_bitstream Complete!
Messages: 101 critical warnings
1 warning
Active run: impl_2
Part: xc7z020clg400-1
Strategy: Vivado Implementation Defaults
Report Strategy: Vivado Implementation Default Reports
Incremental compile: None

```
INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1704] No user IP repositories specified
INFO: [IP_Flow 19-2313] Loaded Vivado IP repository 'C:/Xilinx/Vivado/2018.2/data/ip'.
open_project: Time (s): cpu = 00:00:08 ; elapsed = 00:00:05 . Memory (MB): peak = 878.922 ; gain = 30.016
update_compile_order -fileset sources_1
set_property target_constrs_file C:/xup/fpga_flow/2018_2_zynq_labs/lab2/lab2.srcs/constrs_1/imports/lab2/uart_led_timing_pynq.xdc [current_fileset -constrset]
```

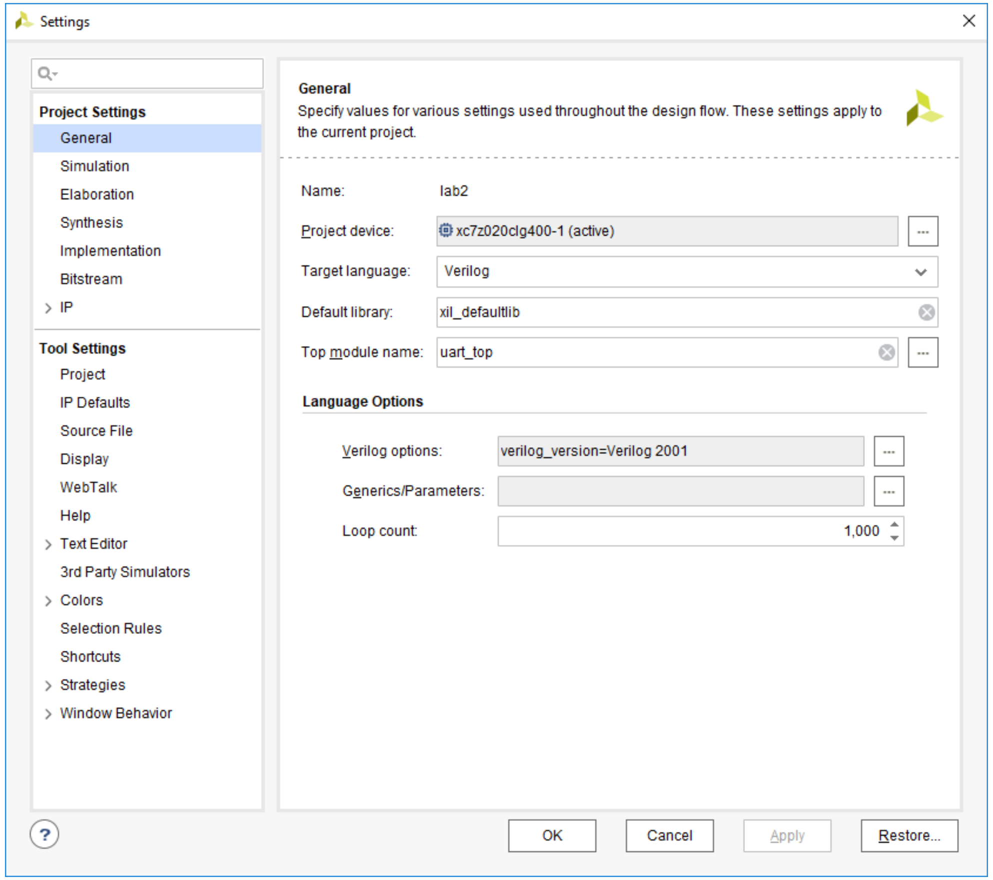
Type a Tcl command here

XILINX.

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**



Settings

General

Specify values for various settings used throughout the design flow. These settings apply to the current project.

Name: lab2

Project device: xc7z020clg400-1 (active)

Target language: Verilog

Default library: xil_defaultlib

Top module name: uart_top

Language Options

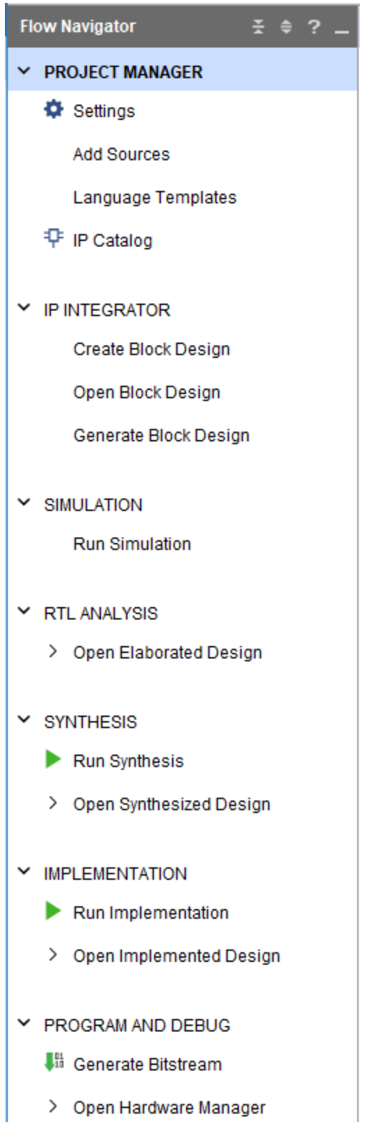
Verilog options: verilog_version=Verilog 2001

Generics/Parameters:

Loop count: 1,000

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

PROJECT MANAGER
- Settings
- Add Sources
- Language Templates
- IP Catalog

IP INTEGRATOR
- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION
- Run Simulation

RTL ANALYSIS
- Open Elaborated Design

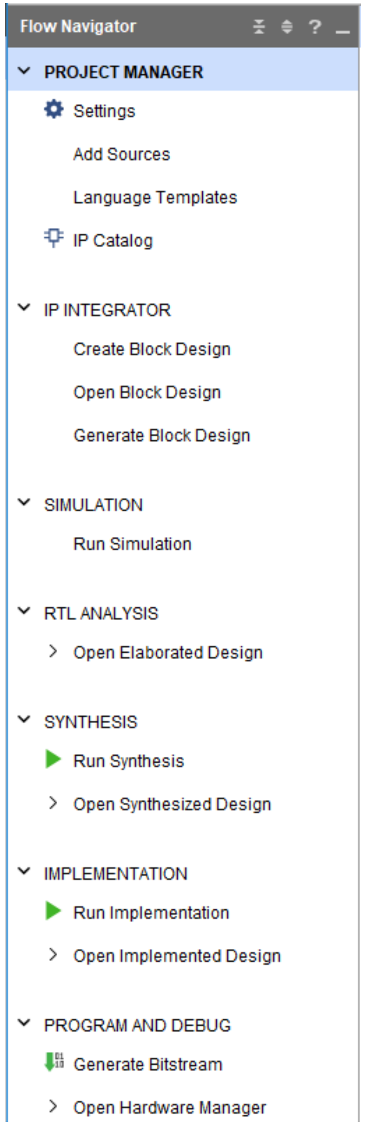
SYNTHESIS
- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION
- Run Implementation
- Open Implemented Design

PROGRAM AND DEBUG
- Generate Bitstream
- Open Hardware Manager

Flow Navigator – RTL Project, cont'd…

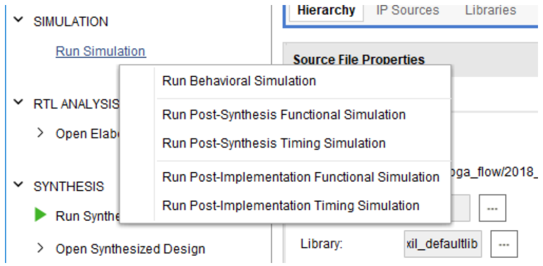
- **Run Synthesis**
  - Timing driven
  - Open Synthesized Design button: Loads synthesized netlist
- **Run Implement button: Runs implementation tools**
  - link, opt, power_opt, place, phys_opt, and route
  - Open Implemented Design button: Loads implemented design
- **Program and Debug: Launches programming and debugging tools**
  - Open Hardware Manager to program the FPGA
  - Hardware Manager also contains debug capabilities

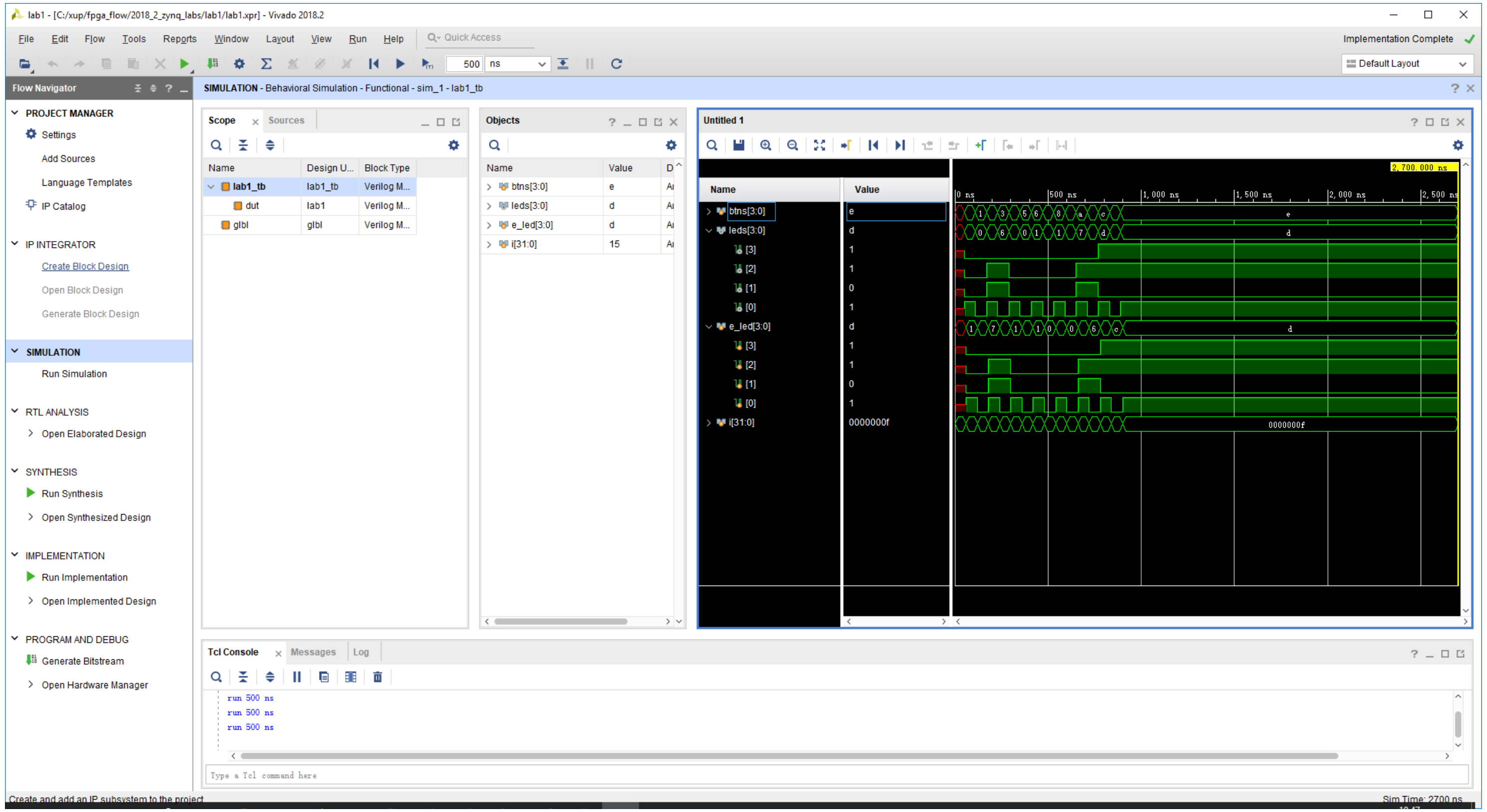


Simulation

> The Vivado simulator, XSIM, supports RTL, netlist, and timing simulation

> Part of the Vivado installation


SIMULATION
Run Simulation
RTL ANALYSIS
Open Elab
SYNTHESIS
Run Synthe
Open Synthesized Design
Hierarchy
IP Sources
Libraries
Source File Properties
Run Behavioral Simulation
Run Post-Synthesis Functional Simulation
Run Post-Synthesis Timing Simulation
Run Post-Implementation Functional Simulation
Run Post-Implementation Timing Simulation
ga_flow/2018_
Library:
xil_defaultlib



Name	Design U...	Block Type
lab1_tb	lab1_tb	Verilog M...
dut	lab1	Verilog M...
glbl	glbl	Verilog M...

Name	Value
btns[3:0]	e
leds[3:0]	d
e_led[3:0]	d
i[31:0]	15

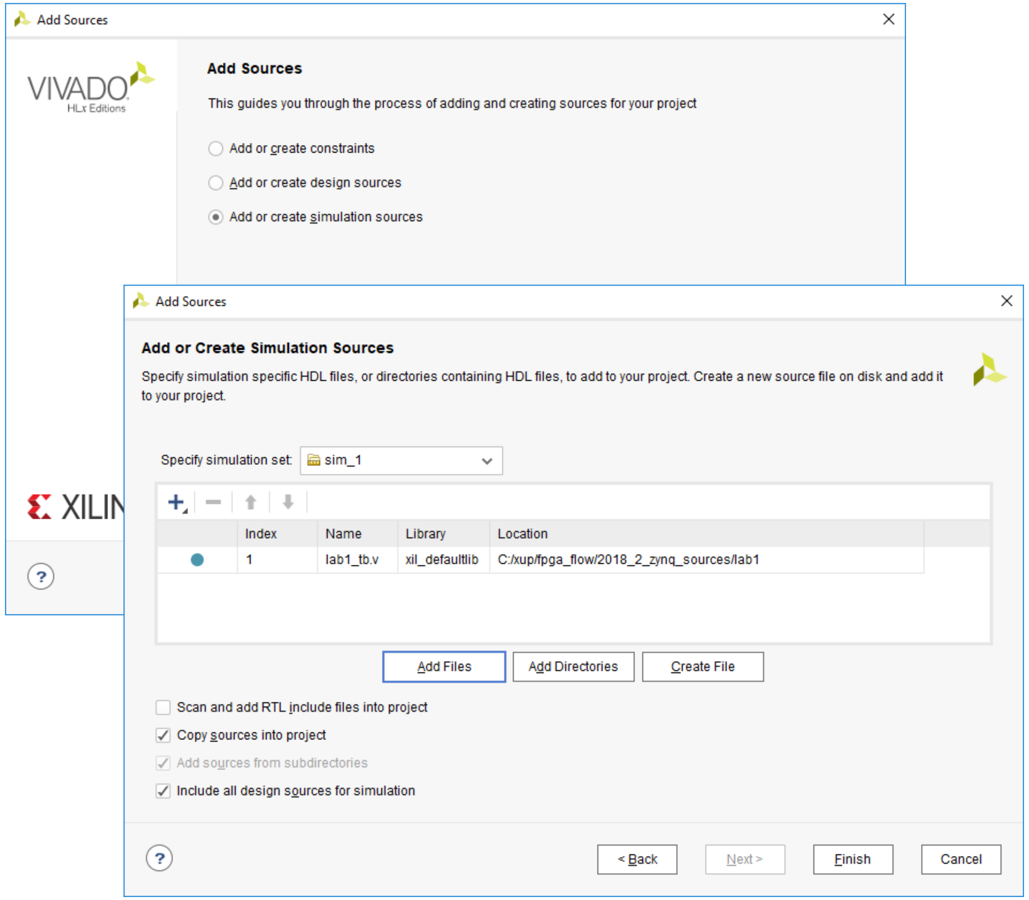
Name	Value
btns[3:0]	e
leds[3:0]	d
[3]	1
[2]	1
[1]	0
[0]	1
e_led[3:0]	d
[3]	1
[2]	1
[1]	0
[0]	1
i[31:0]	0000000f

```
run 500 ns
run 500 ns
run 500 ns
```

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



Add Sources

This guides you through the process of adding and creating sources for your project

○ Add or create constraints
○ Add or create design sources
● Add or create simulation sources

Add or Create Simulation Sources

Specify simulation specific HDL files, or directories containing HDL files, to add to your project. Create a new source file on disk and add it to your project.

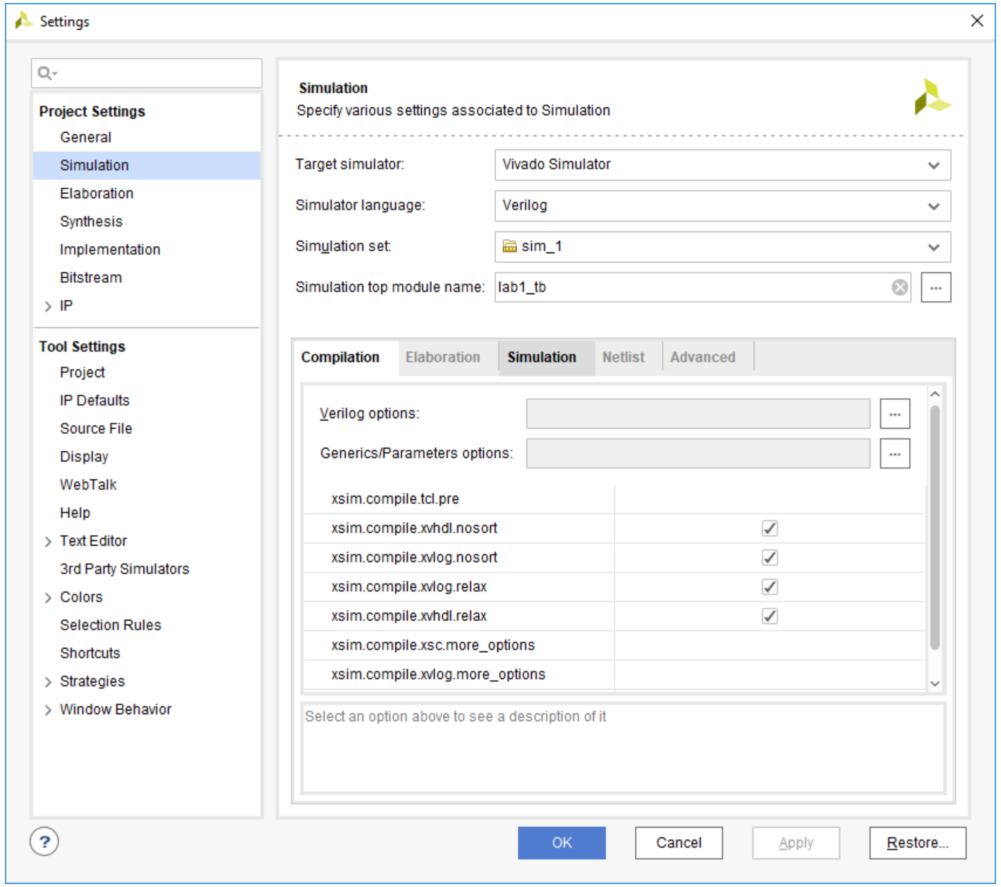
Specify simulation set: sim_1

	Index	Name	Library	Location
●	1	lab1_tb.v	xil_defaultlib	C:/xup/fpga_flow/2018_2_zynq_sources/lab1

☐ Scan and add RTL include files into project
☑ Copy sources into project
☑ Add sources from subdirectories
☑ Include all design sources for simulation

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

Specify various settings associated to Simulation

Target simulator: Vivado Simulator

Simulator language: Verilog

Simulation set: sim_1

Simulation top module name: lab1_tb

Verilog options:

Generics/Parameters options:

xsim.compile.tcl.pre	
xsim.compile.xvhdl.nosort	☑
xsim.compile.xvlog.nosort	☑
xsim.compile.xvlog.relax	☑
xsim.compile.xvhdl.relax	☑
xsim.compile.xsc.more_options	
xsim.compile.xvlog.more_options	

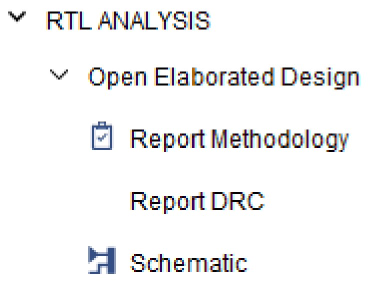
Select an option above to see a description of it

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

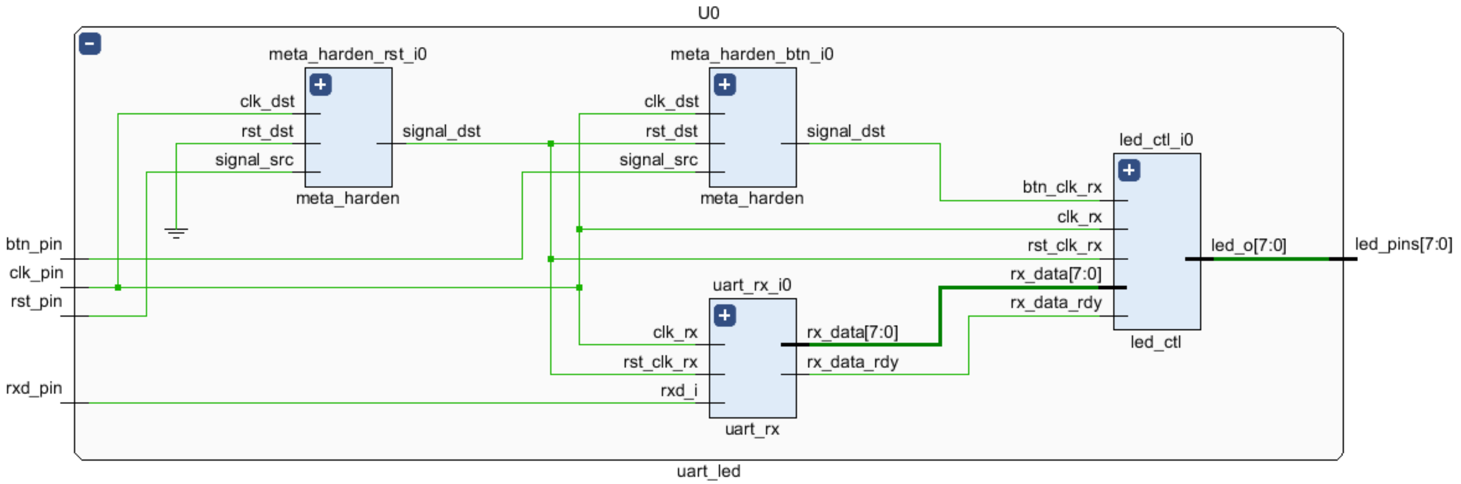
After Elaboration

- 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



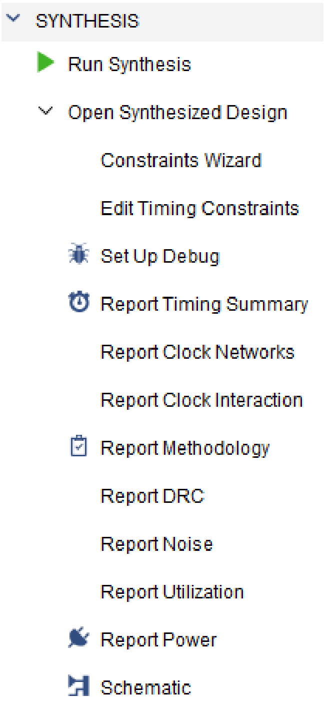
RTL ANALYSIS

- Open Elaborated Design
  - Report Methodology
  - Report DRC
  - Schematic

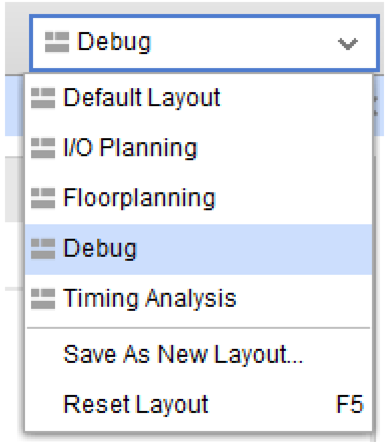


After Synthesis

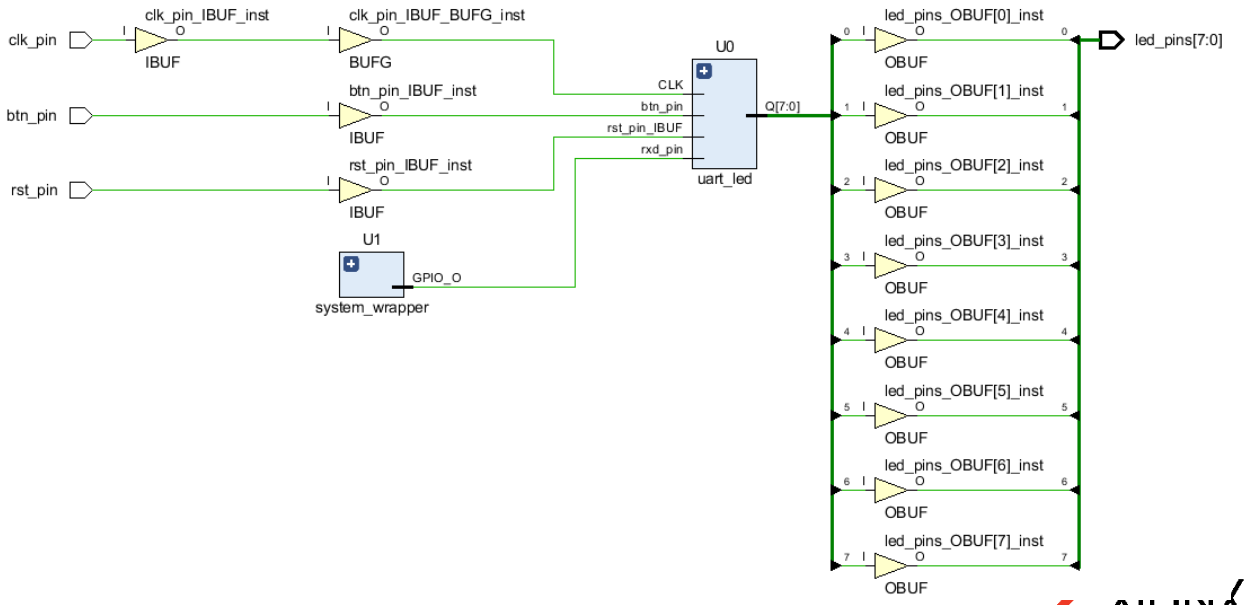
- **The Flow Navigator includes access to:**
  - Constraints Wizard,
  - Edit Timing Constraints,
  - Set Up Debug,
  - Report Timing Summary,
  - Report Clock Networks,
  - Report Clock Interaction,
  - Report DRC,
  - Report Noise,
  - Report Utilization,
  - Report Power,
  - and Schematic utilities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



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

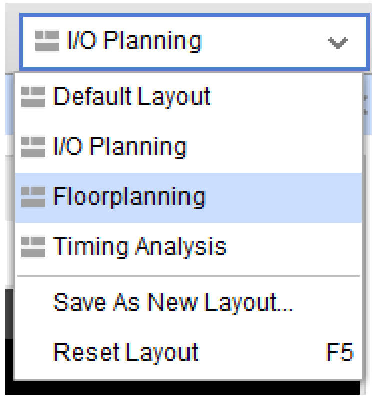


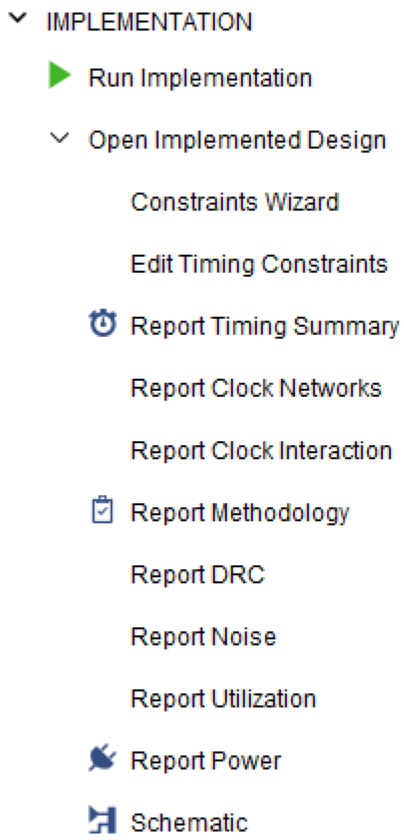
Debug
- Default Layout
- I/O Planning
- Floorplanning
- Debug
- Timing Analysis
- Save As New Layout...
- Reset Layout F5



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**





Summary

XILINX

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