

Advanced Workshop on  
FPGA-based Systems-On-Chip  
for Scientific Instrumentation and  
Reconfigurable Computing



# *Vivado Design Flow for SoC*

---

*Cristian Sisterna*  
*Universidad Nacional de San Juan*  
*Argentina*

# Why Vivado Design Suite?

---

Larger FPGAs lead to more difficult design issues

- Users integrating more functionality into the FPGA
  - Use of multiple hard logic objects (block RAMs, GTs, DSP slices, and microprocessors, for example)
- I/O and clock planning critical to FPGA performance
- Higher routing and utilization density
- Complex timing constraints with designs that have multiple clock domains

FPGA designs are now looking like ASIC platform designs

- Assembled from IP cores—commercial or developed in-house
  - Maintaining place and route solutions is very important (this is resolved with the use of partitions)
  - Bottom-up design methodology
  - Team design flows becoming a necessity

***Vivado Design Suite provides solution to all of the above***

# Vivado IDE Solution

## Interactive design and analysis

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

## RTL development and analysis

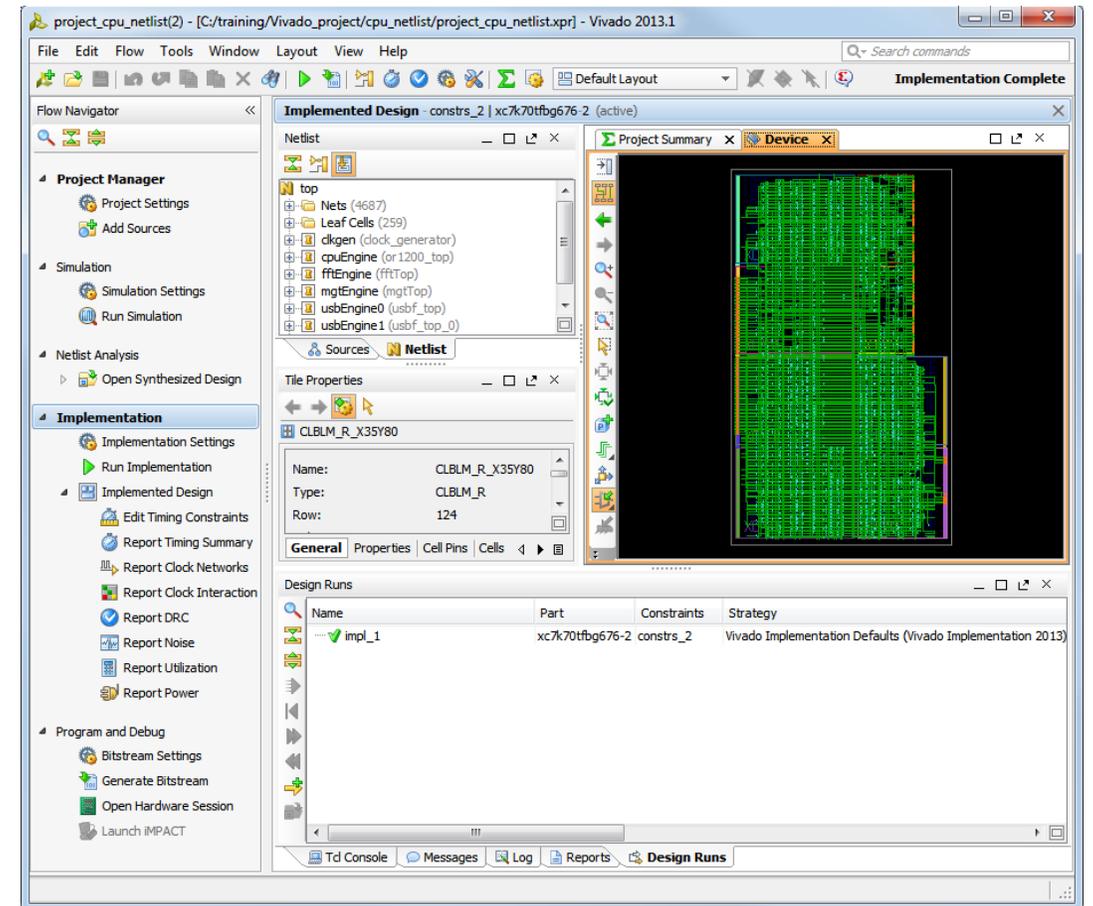
- Elaboration of HDL
- Hierarchical exploration
- Schematic generation

## XSIM simulator integration

- Synthesis, implementation and simulation in one package

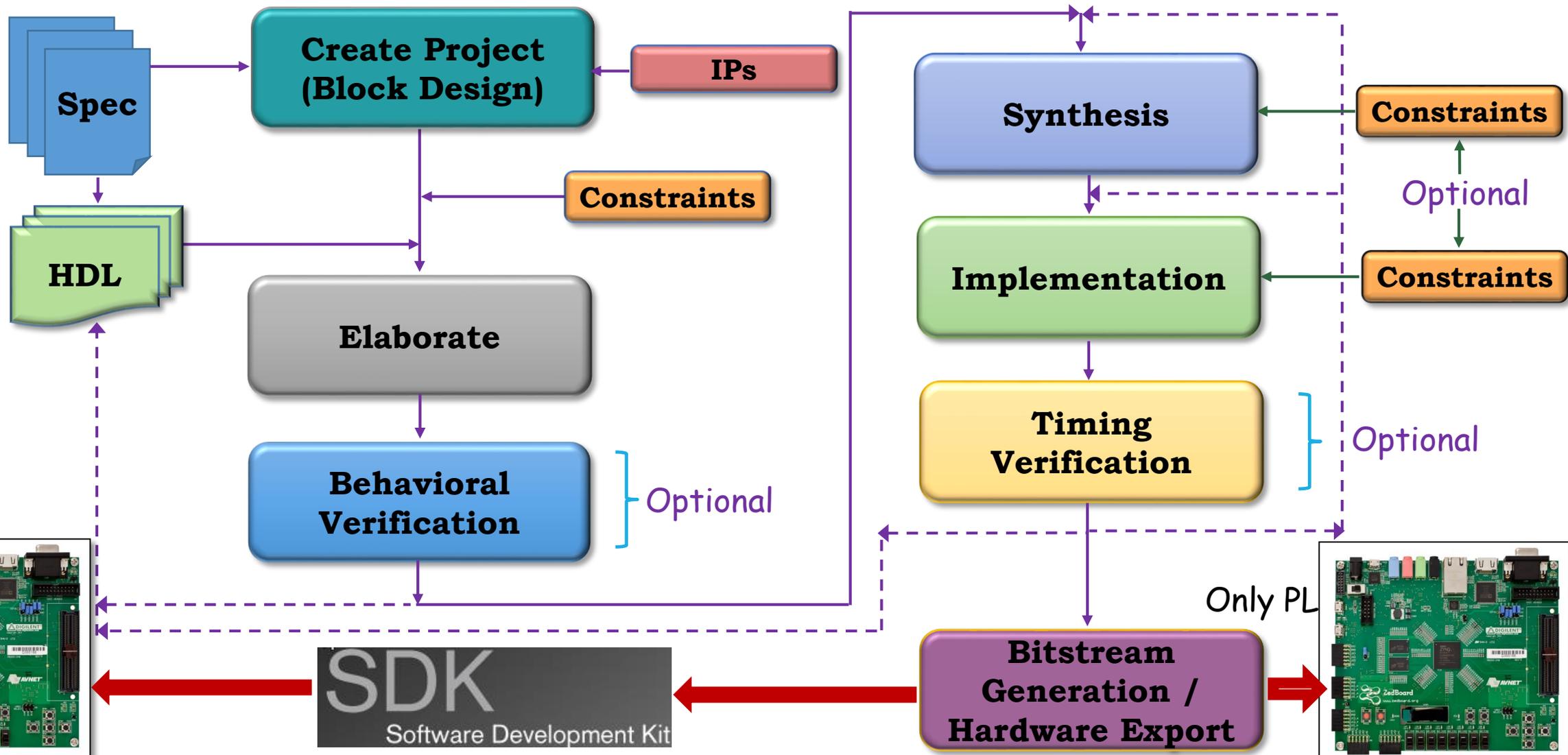
## I/O pin planning

- Interactive rule-based I/O assignment



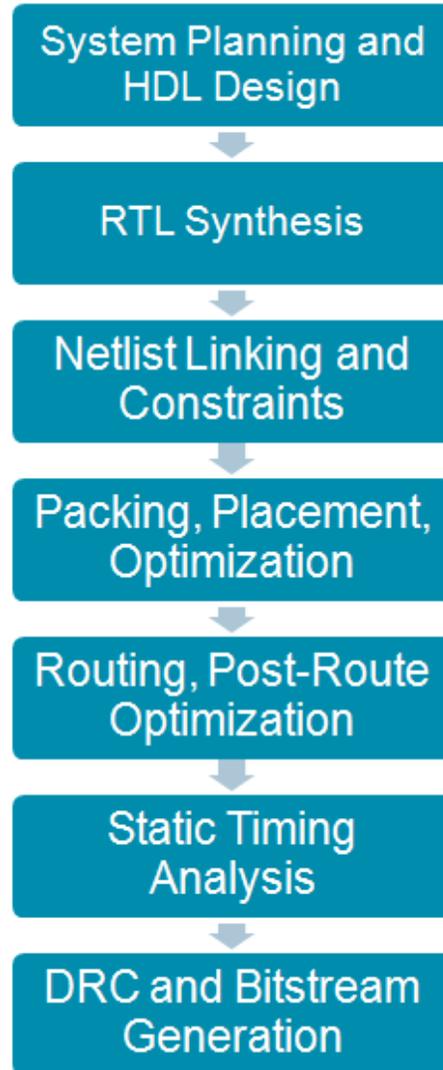
*Hierarchical Design Analysis and Implementation Environment*

# Embedded System Design – Vivado Flow



# Typical vs Vivado Design Flow

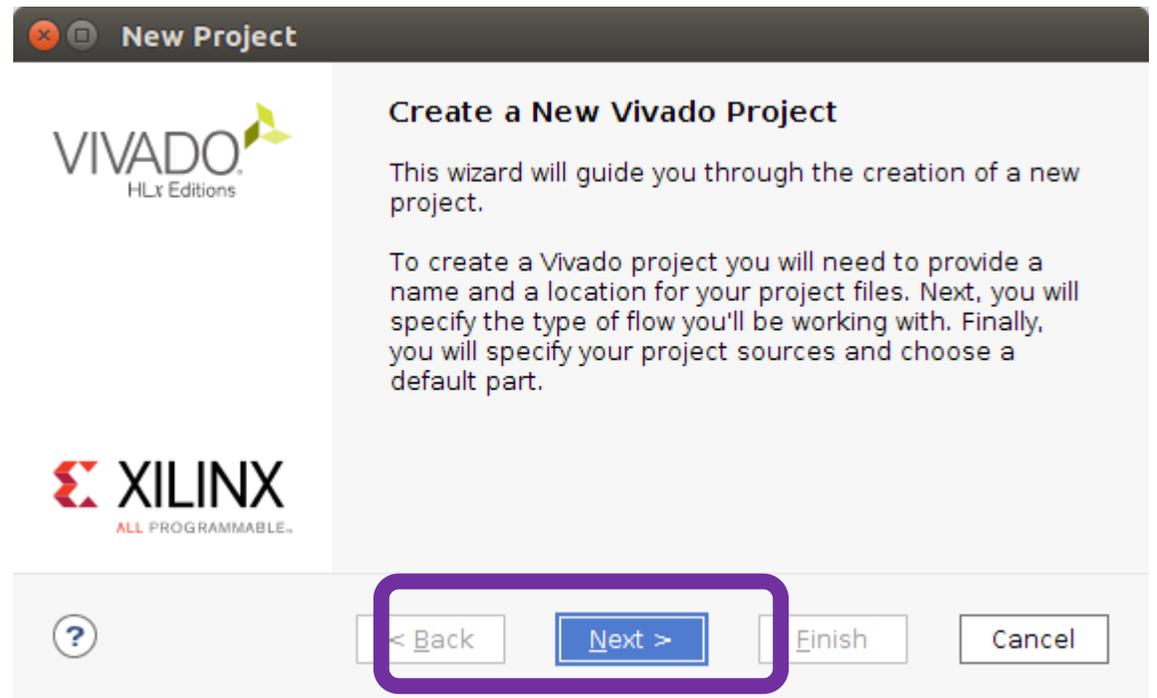
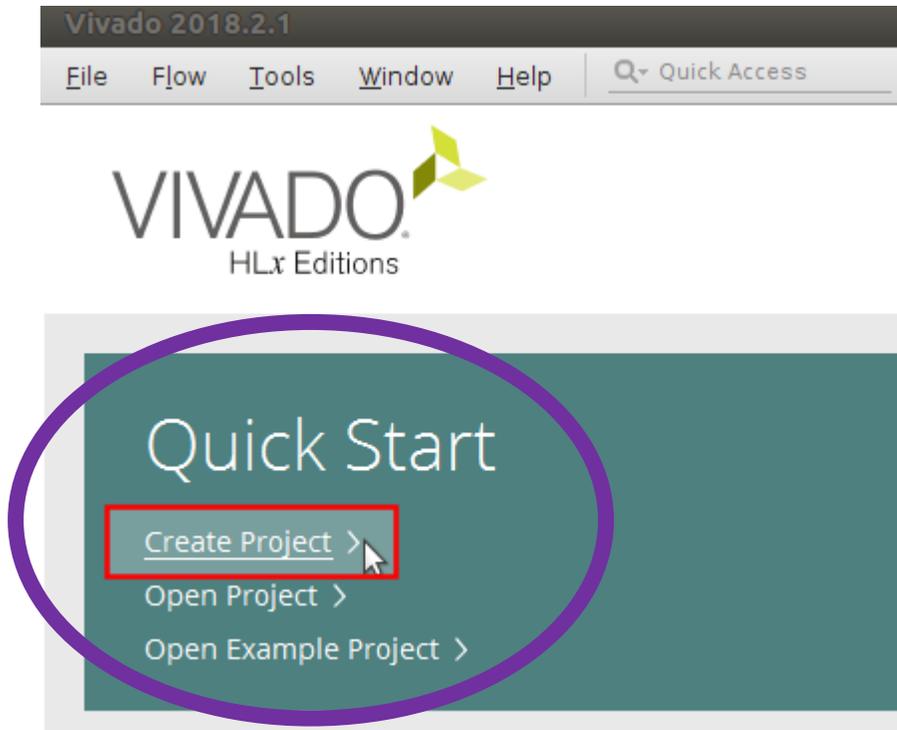
---



# *Vivado Flow Practical Steps*

---

# Creating a Project in Vivado



# Creating a Project in Vivado

**New Project**

**Project Name**  
Enter a name for your project and specify a directory where the project data files will be stored.

Project name:

Project location:

Create project subdirectory

Project will be created at: /projects/lab\_hello\_world



**New Project**

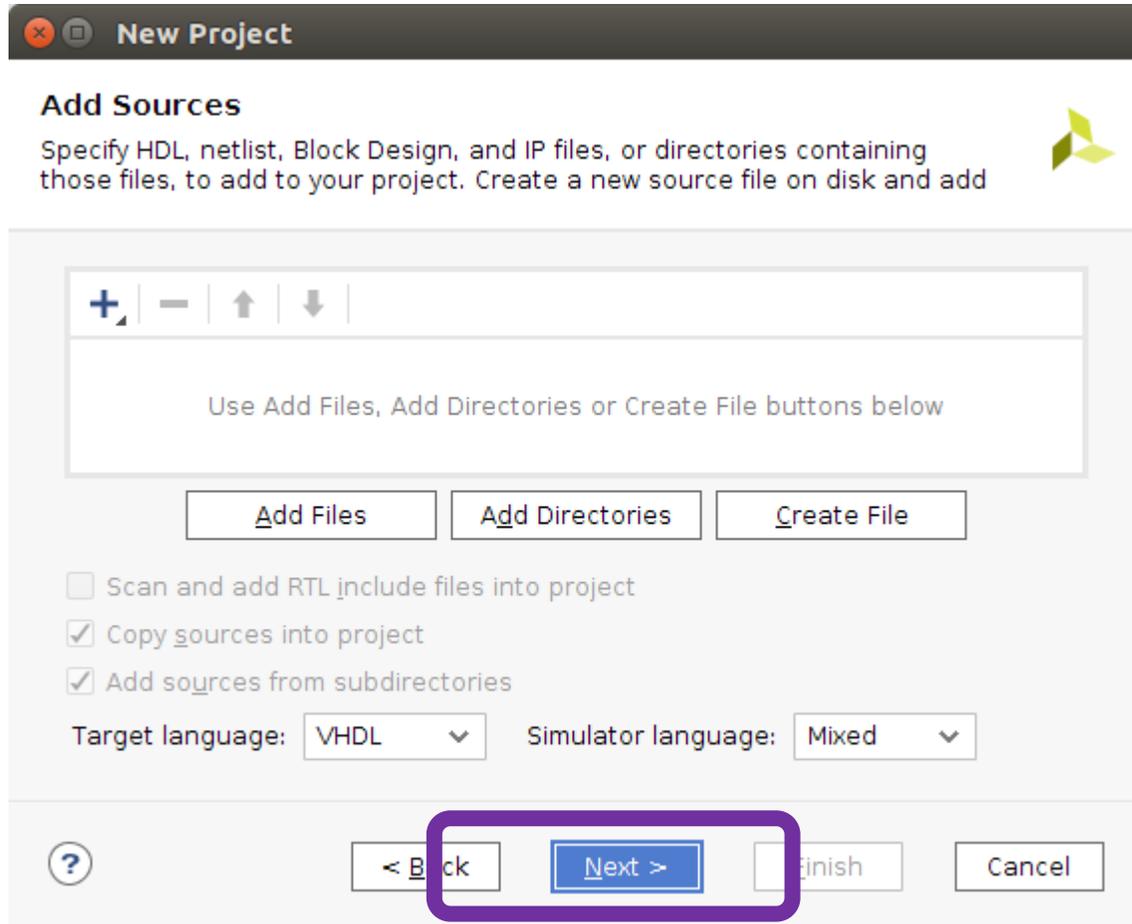
**Project Type**  
Specify the type of project to create.

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

Do not specify sources at this time

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

# Creating a Project in Vivado



**New Project**

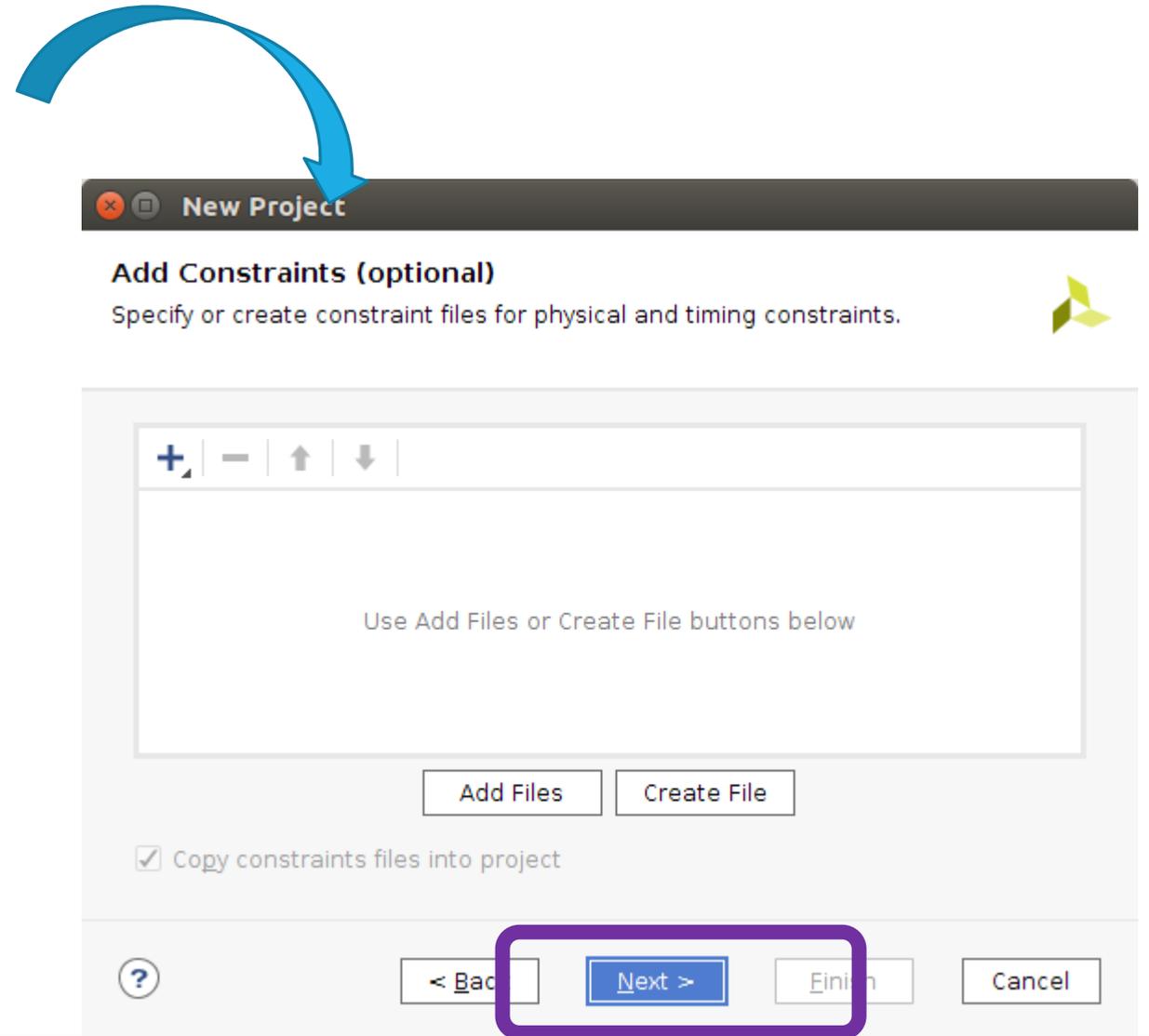
### Add Sources

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

Use Add Files, Add Directories or Create File buttons below

Scan and add RTL include files into project  
 Copy sources into project  
 Add sources from subdirectories

Target language:  Simulator language:



**New Project**

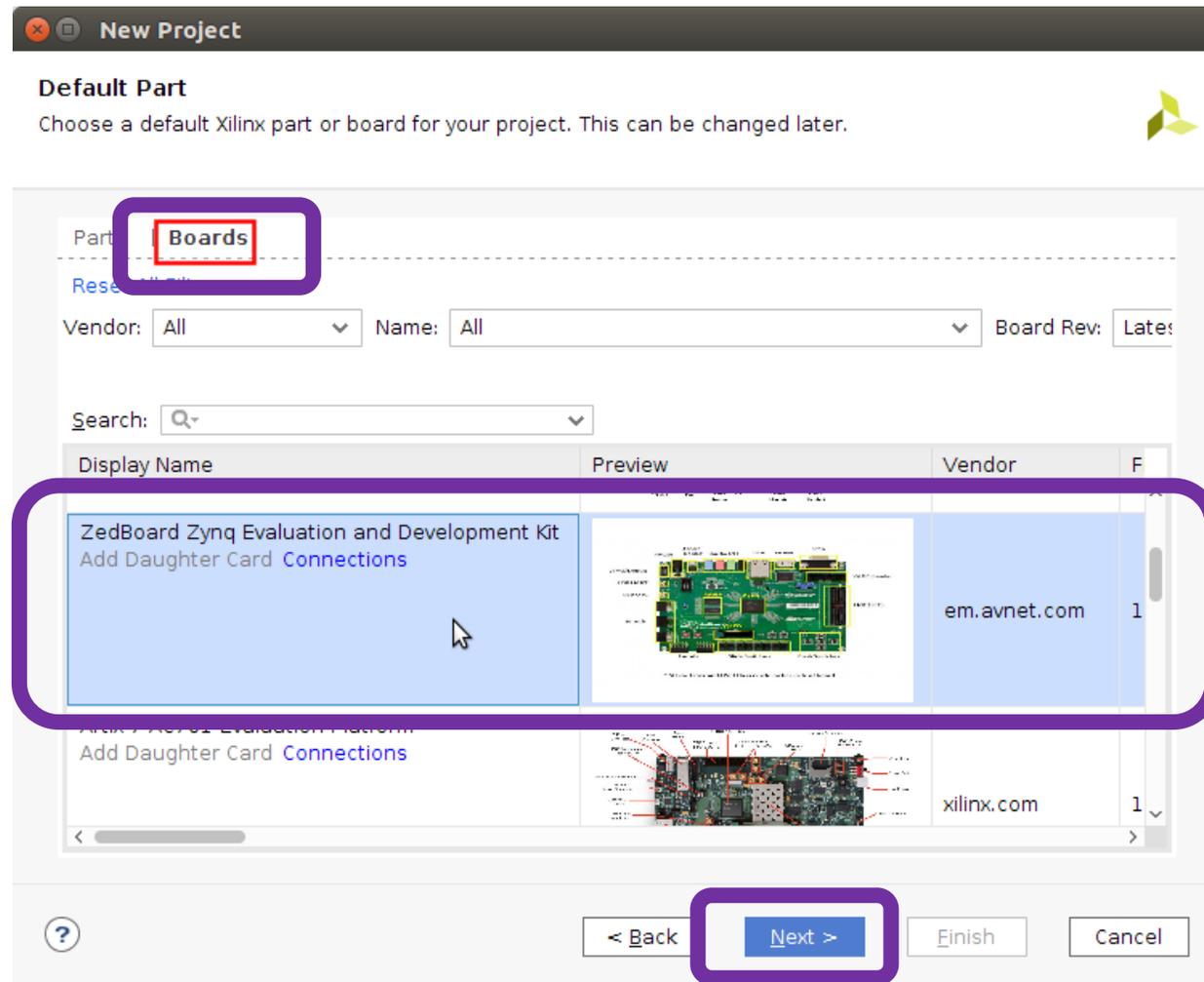
### Add Constraints (optional)

Specify or create constraint files for physical and timing constraints.

Use Add Files or Create File buttons below

Copy constraints files into project

# Creating a Project in Vivado



File Edit Flow Tools Reports Window Layout View Help | Q- Quick Access

**Menu Bar 1** **Project Status Bar 1** Ready Default Layout

**Main Toolbar 2**

**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 Mana **1**

**Flow Navigator**

PROJECT MANAGER - lab\_hello\_world

**Sources**

- Design Sources
- Constraints
- Simulation Sources
  - sim\_1

Hierarchy Libraries Compile Order

**Properties**

Select an object to see properties

**Project Manager 6**

**Project Summary**

Settings Edit

Project name: lab\_hello\_world  
 Project location: /projects/lab\_hello\_world  
 Product family: Zynq-7000  
 Project part: ZedBoard Zynq Evaluation and Development Kit (xc7z020clg484-1)  
 Top module name: Not defined  
 Target language: VHDL  
 Simulator language: Mixed

**Board Part**

Display name: ZedBoard Zynq Evaluation and Development Kit  
 Board part name: em.avnet.com:zed:part0:1.4  
 Connectors: No connections  
 Repository path: /opt/Xilinx/Vivado/2018.2/data/boards/board\_files  
 URL: <http://www.zedboard.org>  
 Board overview: ZedBoard Zynq Evaluation and Development Kit  
[Changes](#)

**Workspace 3**



Tcl Console Messages Log Reports Design Runs x

Name Constraints Status WNS TNS WHS THS TPWS Total Power Failed Routes LUT FF BRAMs URAM DSP Start Elapsed Run Strategy

synth_1	constrs_1	Not started																Vivado Synthesis Defaults (Vivado
impl_1	constrs_1	Not started																Vivado Implementation Defaults (\

**Message & Results 7**

# Project Navigator Main Components

---

1. **Menu Bar:** Vivado IDE commands
2. **Main Toolbar:** Access to the most commonly used Vivado IDE commands
3. **Workspace:** area for schematic panel, device panel, package panel, text editor panel.
4. **Project Status Bar:** displays the status of the currently active design
5. **Flow Navigator:** provide easy access to the tools and commands necessary to guide the design from start to finish.
6. **Project Manager Pane:** by default displays information related to design data and sources, such as Property Window, Netlist Window, and Source Window
7. **Project Status Bar:** displays information about menu bar and toolbar commands; task progresses
8. **Results Window Area:** there are a set of windows, such as Messages, showing message for each process, Tcl Console, Tcl commands of each activity, Reports, reports generated throughout the design flow, Desing Runs, display the different run for the current project

# Create a Block design

The screenshot displays the Vivado 2018.2.1 interface for a project named 'lab\_hello\_world'. The 'IP INTEGRATOR' section in the left sidebar is highlighted with a purple box, and a purple arrow points from the 'Create Block Design' button to the 'Create Block Design' dialog box. The dialog box prompts for a design name, directory, and source set.

**Project Manager - lab\_hello\_world**

**Sources**

- Design Sources
- Constraints
- Simulation Sources
  - sim\_1

**Project Summary**

**Settings** Edit

Project name:	lab_hello_world
Project location:	/projects/lab_hello_world
Product family:	Zynq-7000
Project part:	ZedBoard Zynq
Top module name:	Not defined
Target language:	VHDL
Simulator language:	Mixed

**Create Block Design**

Please specify name of block design.

Design name:

Directory:

Specify source set:

OK Cancel

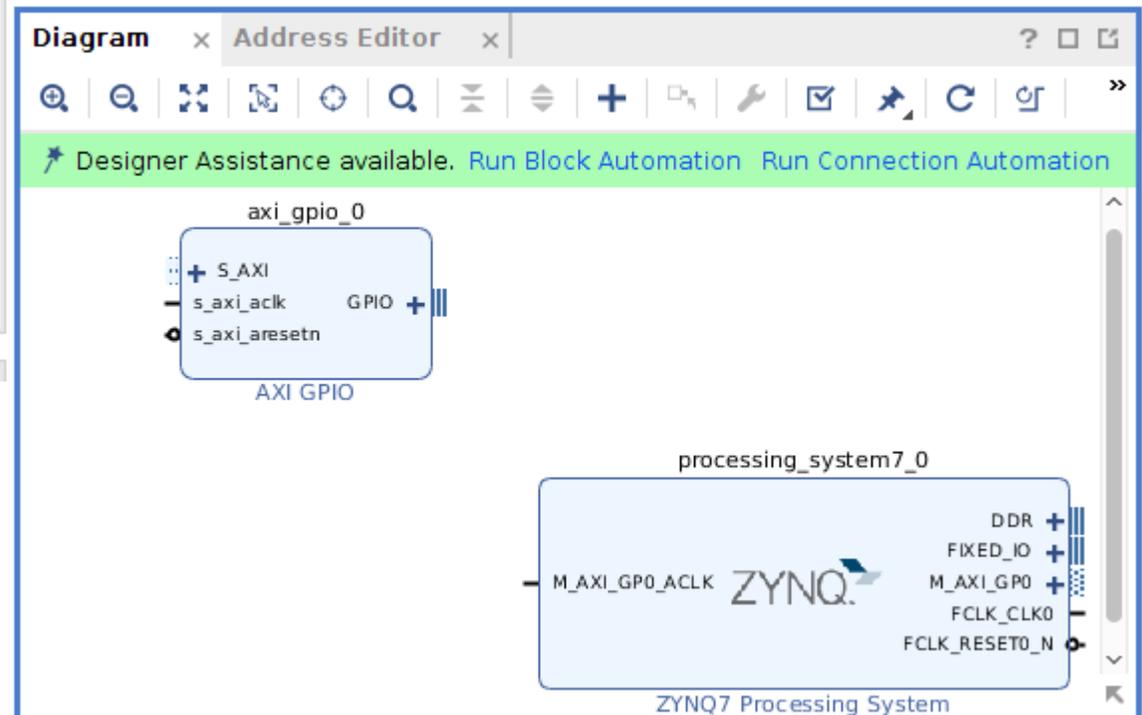
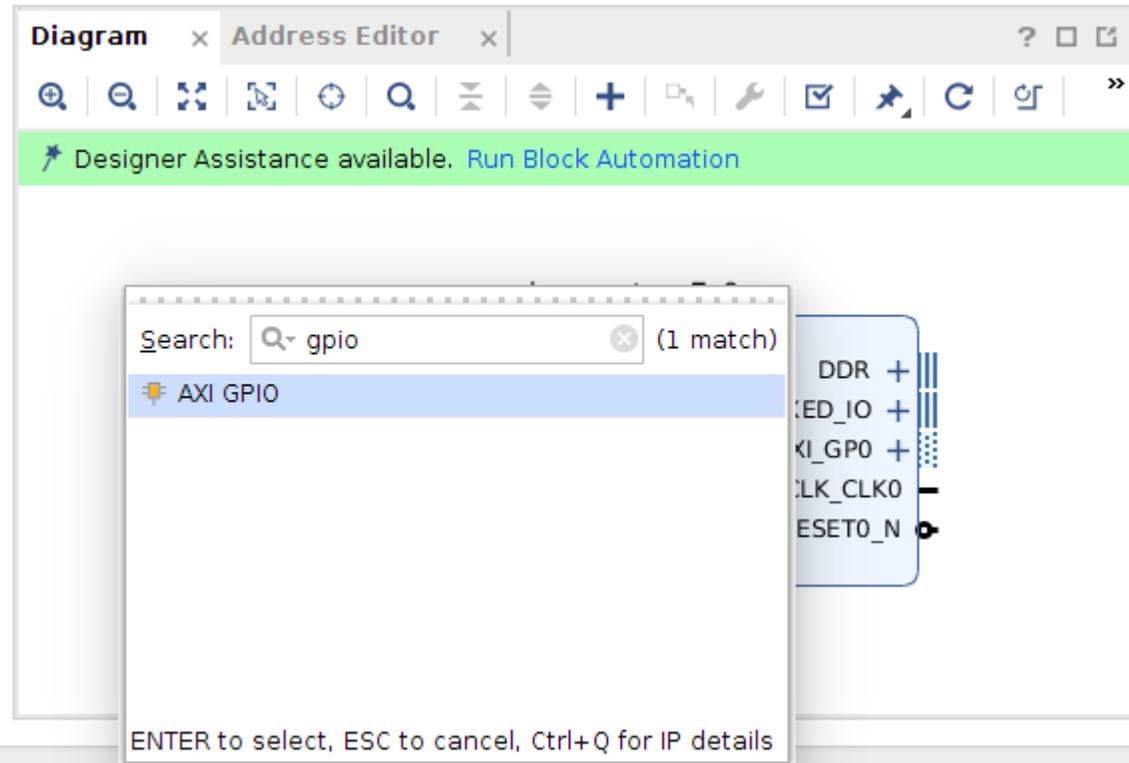
# Adding IP Modules to the Design Canvas

The image illustrates the process of adding an IP module to a design canvas in Vivado. It is divided into two main parts:

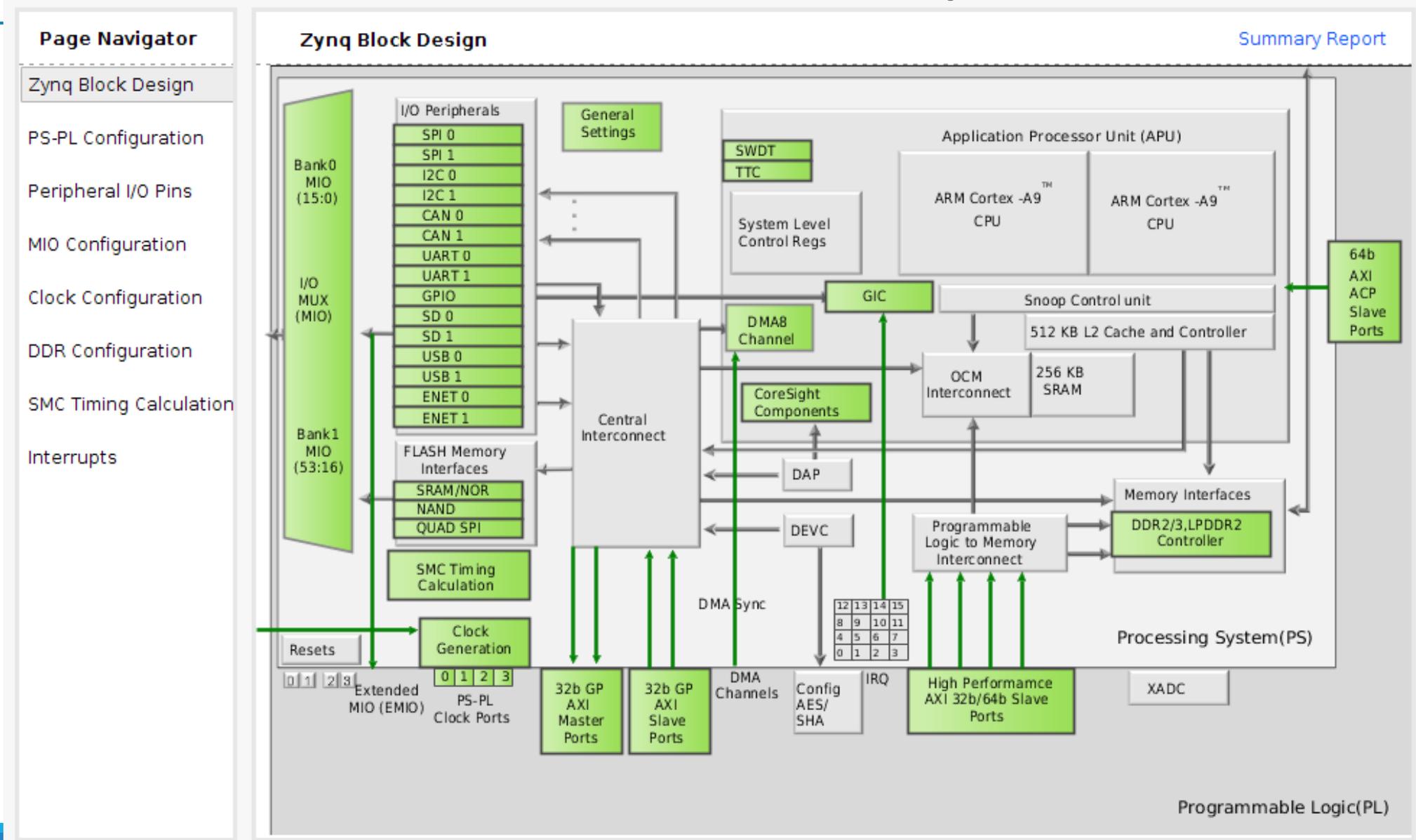
- Left Panel (Initial State):** Shows the Vivado interface for a design named "lab\_hw". The "Diagram" window is empty and contains the text: "This design is empty. Press the + button to add IP." A purple box highlights the "+" button in the Diagram toolbar, with a tooltip that reads "Add IP (Ctrl+I)".
- Right Panel (Search Results):** Shows the same "Diagram" window after the "+" button is clicked. A search dialog box is open, displaying the search results for "zy". The search results list "ZYNQ7 Processing System" as the only match. A blue arrow points from the "+" button in the left panel to the search dialog in the right panel.

At the bottom of the search dialog, instructions are provided: "ENTER to select, ESC to cancel, Ctrl+Q for IP details".

# Adding More IPs



# PS Customization Options



# PS-PL Configuration Options

Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation Presets IP Location Import XPS Settings

Page Navigator

- Zynq Block Design
- PS-PL Configuration**
- Peripheral I/O Pins
- MIO Configuration
- Clock Configuration
- DDR Configuration
- SMC Timing Calculati
- Interrupts

PS-PL Configuration [Summary Report](#)

Search:

Name	Select	Description
> General		
AXI Non Secure Enablement	0	Enable AXI Non Secure Transaction
> GP Master AXI Interface		
> M AXI GP0 interface	<input checked="" type="checkbox"/>	Enables General purpose AXI master interface 0
> M AXI GP1 interface	<input type="checkbox"/>	Enables General purpose AXI master interface 1
> GP Slave AXI Interface		
> HP Slave AXI Interface		
> ACP Slave AXI Interface		
> DMA Controller		
> PS-PL Cross Trigger interface	<input type="checkbox"/>	Enables PL cross trigger signals to PS and vice-versa

OK Cancel

# MIO and EMIO Configuration

Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation Presets IP Location Import XPS Settings

Page Navigator

- Zynq Block Design
- PS-PL Configuration
- Peripheral I/O Pins
- MIO Configuration
- Clock Configuration
- DDR Configuration
- SMC Timing Calculati
- Interrupts

Peripheral I/O Pins [Summary Report](#)

Search:

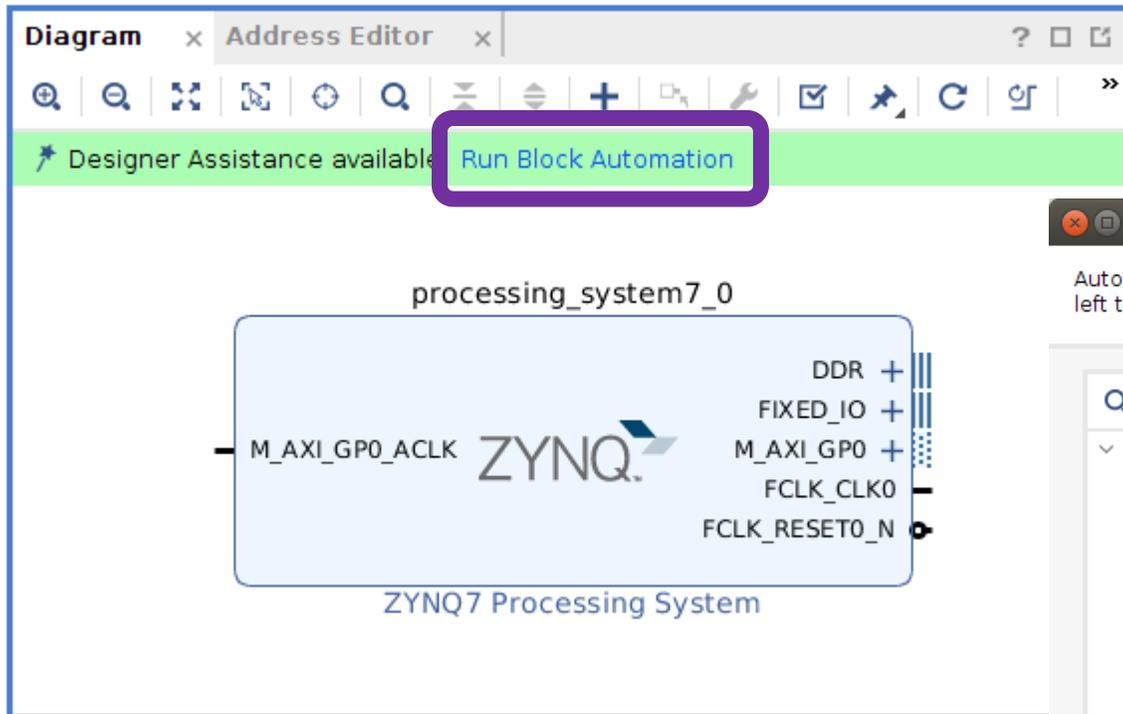
Peripherals

- UART 0
- UART 1
- I2C 0
- I2C 1
- CAN 0
- CAN 1
- TTC0
- TTC1

34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	EMIO
UART0				UART0				UART0				UART0				UART0				EMIO
			UART1			UART1		EMIO												
I2C0			I2C0			I2C0				I2C0				I2C0						EMIO
			I2C1			I2C1		EMIO												
CAN0			CAN0			CAN0				CAN0				CAN0						EMIO
			CAN1			CAN1		EMIO												
								TTC0												EMIO

OK Cancel

# Running Block Automation



### Run Block Automation

Automatically make connections in your design by checking the boxes of the blocks to connect. Select a block on the left to display its configuration options on the right.

**Description**

This option sets the board preset on the Processing System. All current properties will be overwritten by the board preset. This action cannot be undone. Zynq7 block automation applies current board preset and generates external connections for FIXED\_IO, Trigger and DDR interfaces.

NOTE: Apply Board Preset will discard existing IP configuration - please uncheck this box, if you wish to retain previous configuration.

Instance: /processing\_system7\_0

**Options**

Make Interface External: FIXED\_IO, DDR

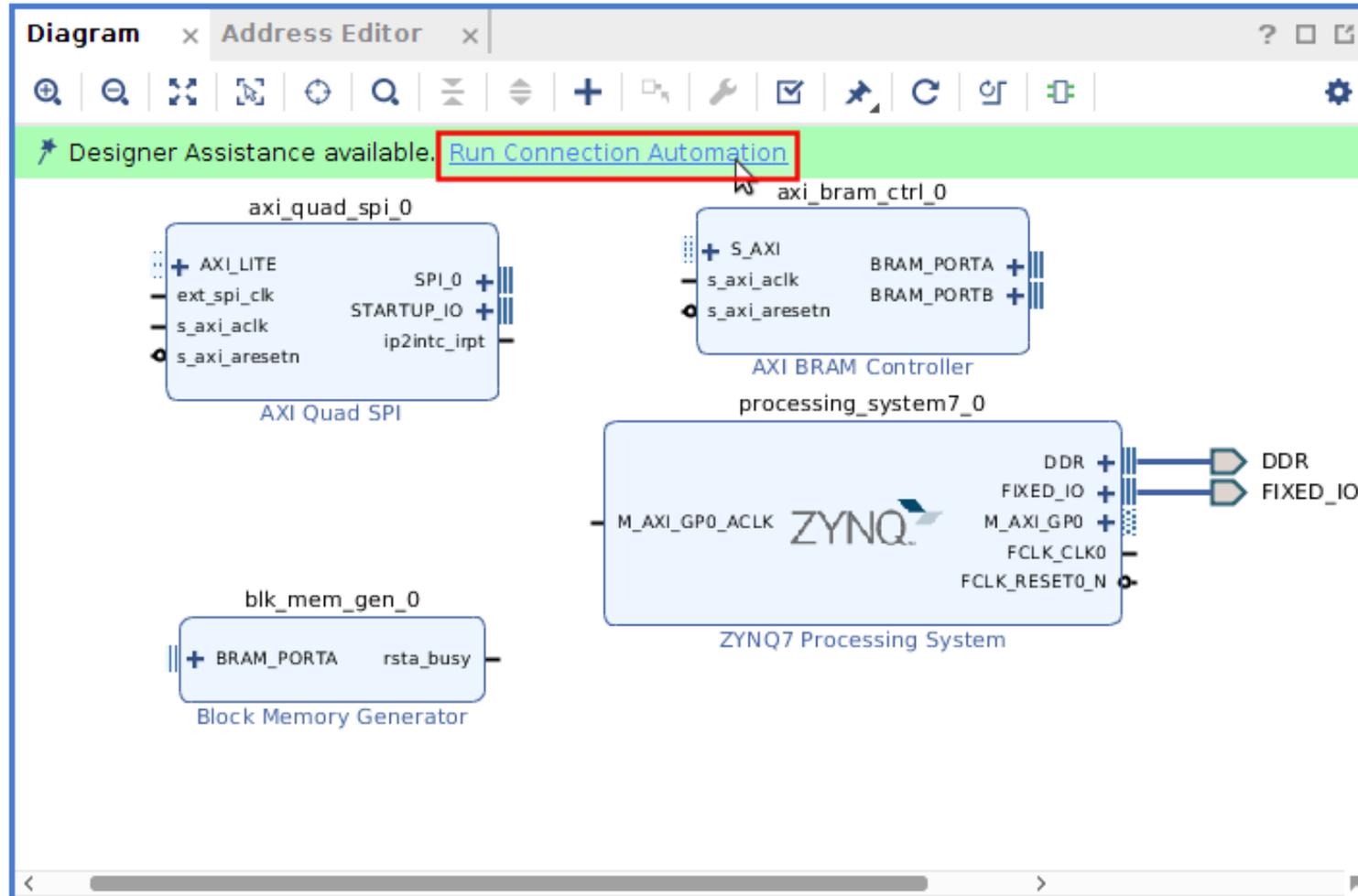
Apply Board Preset:

Cross Trigger In: Disable

Cross Trigger Out: Disable

OK Cancel

# Run Connection Automation



# Connecting IPs – Making Connections With the Tool

**Run Connection Automation**

Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

**Description**

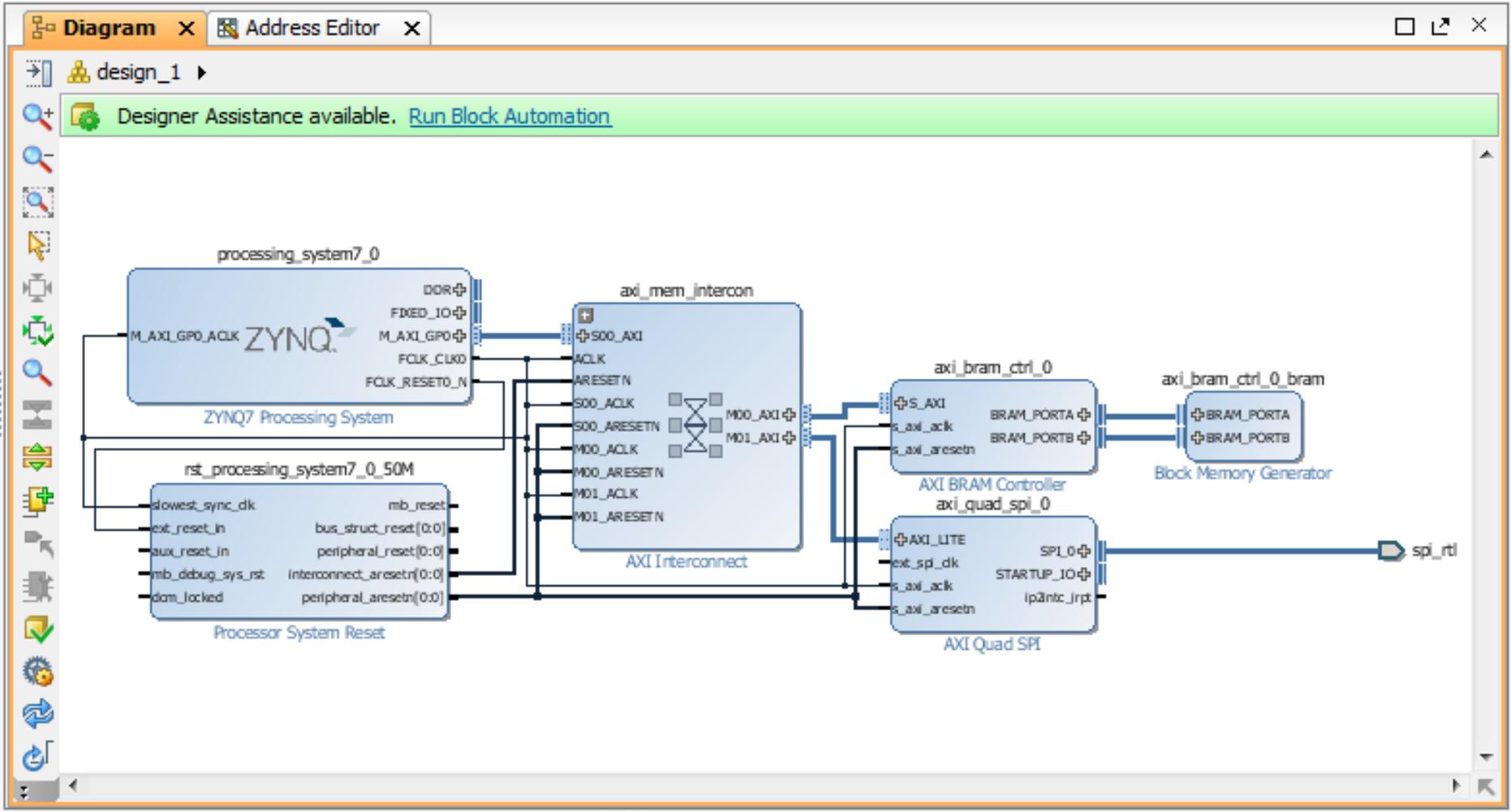
Connect Slave interface (/axi\_quad\_spi\_0/AXI\_LITE) to a selected Master address space.

**Options**

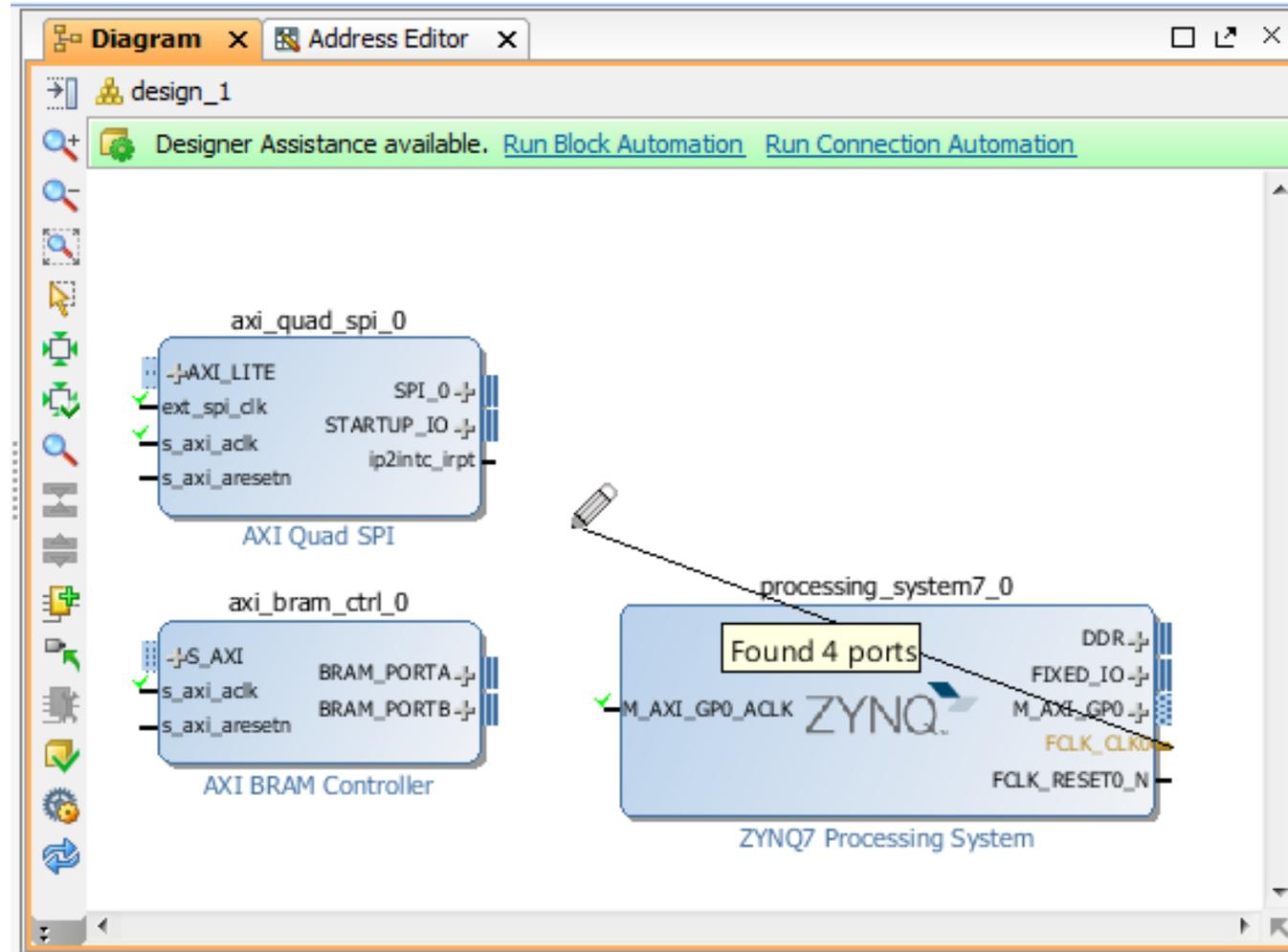
Master	/processing_system7_0/
Bridge IP	New AXI Interconnect
Clock source for driving Interconnect IP	Auto
Clock source for Master interface	Auto
Clock source for Slave interface	Auto

OK Cancel

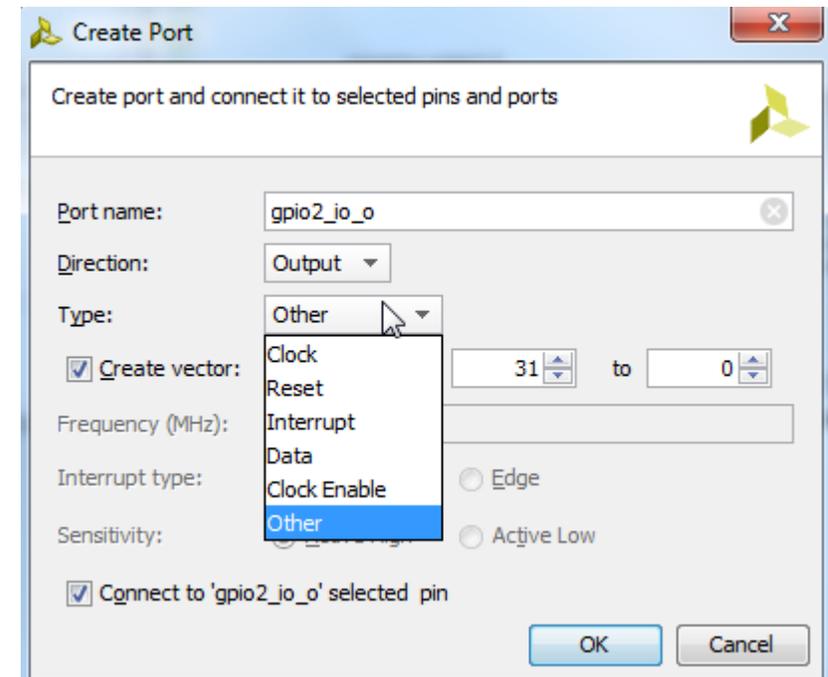
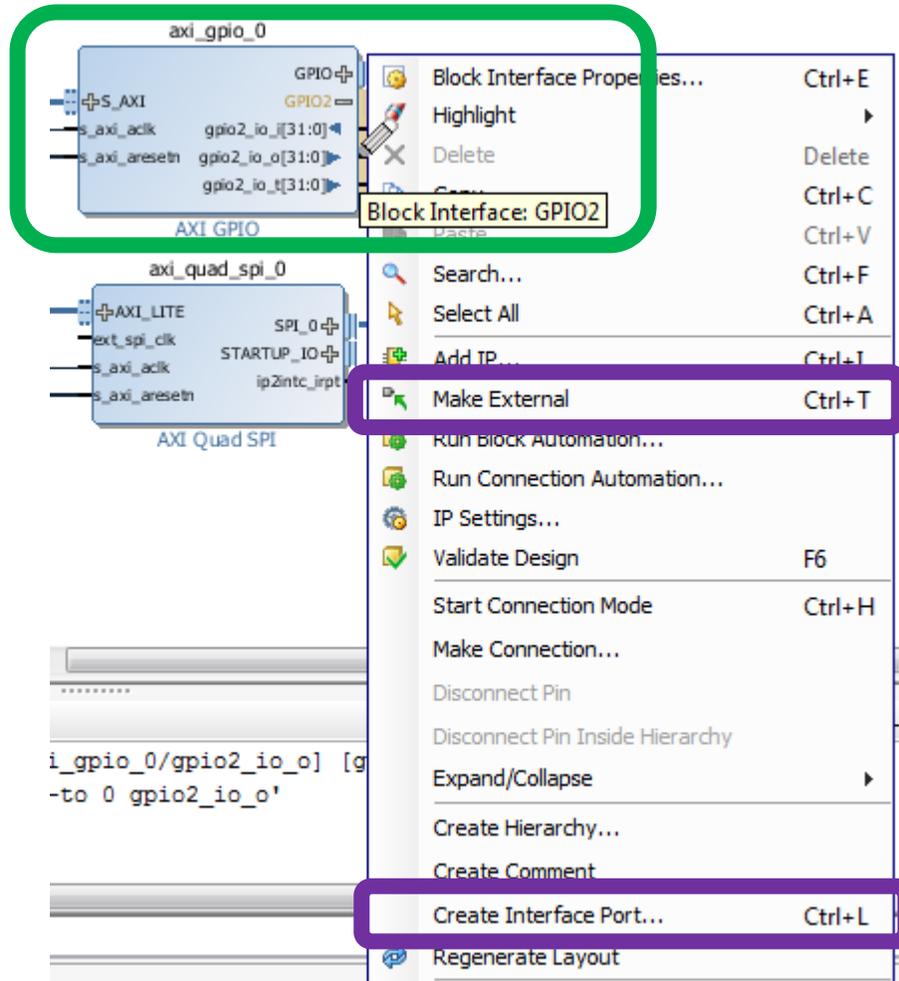
# Connecting IPs – Making Connections With the Tool



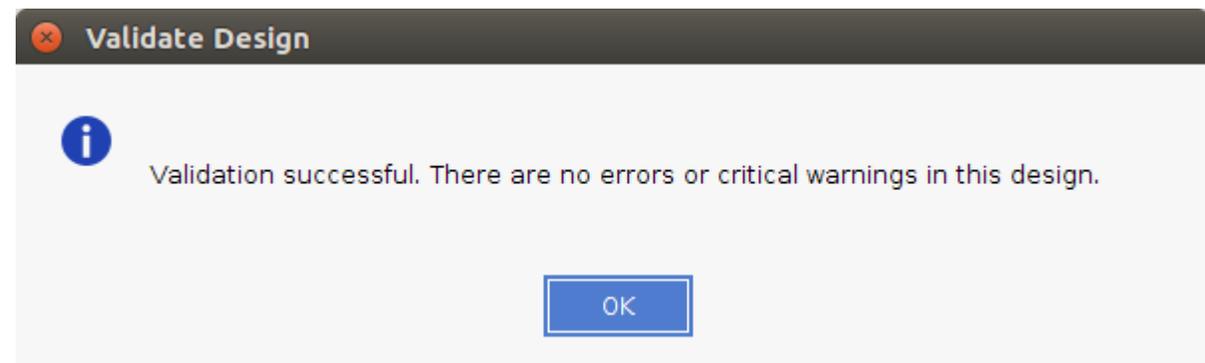
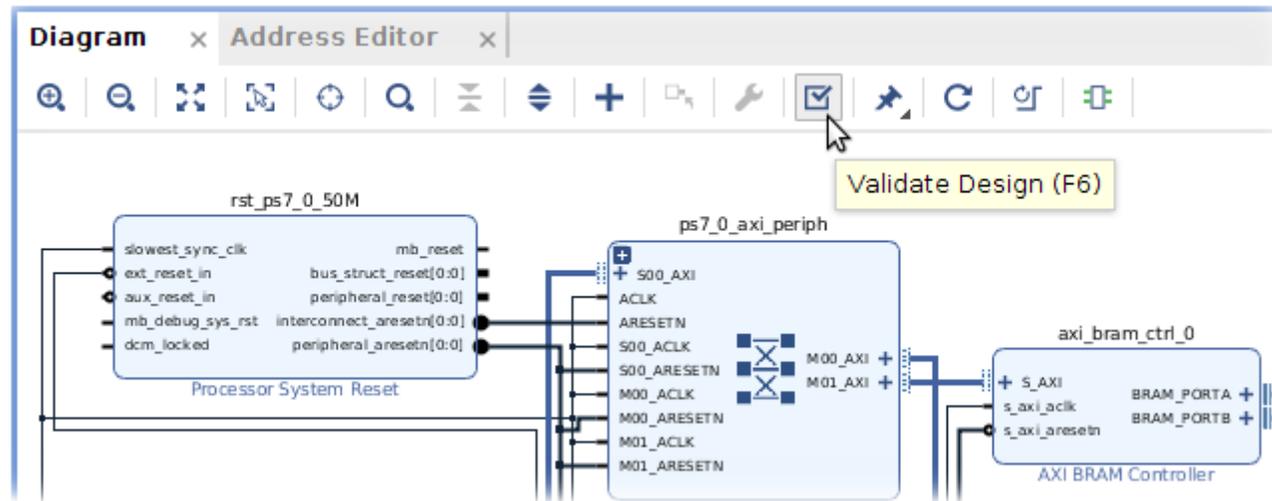
# Connecting IPs – Making Connections Manually



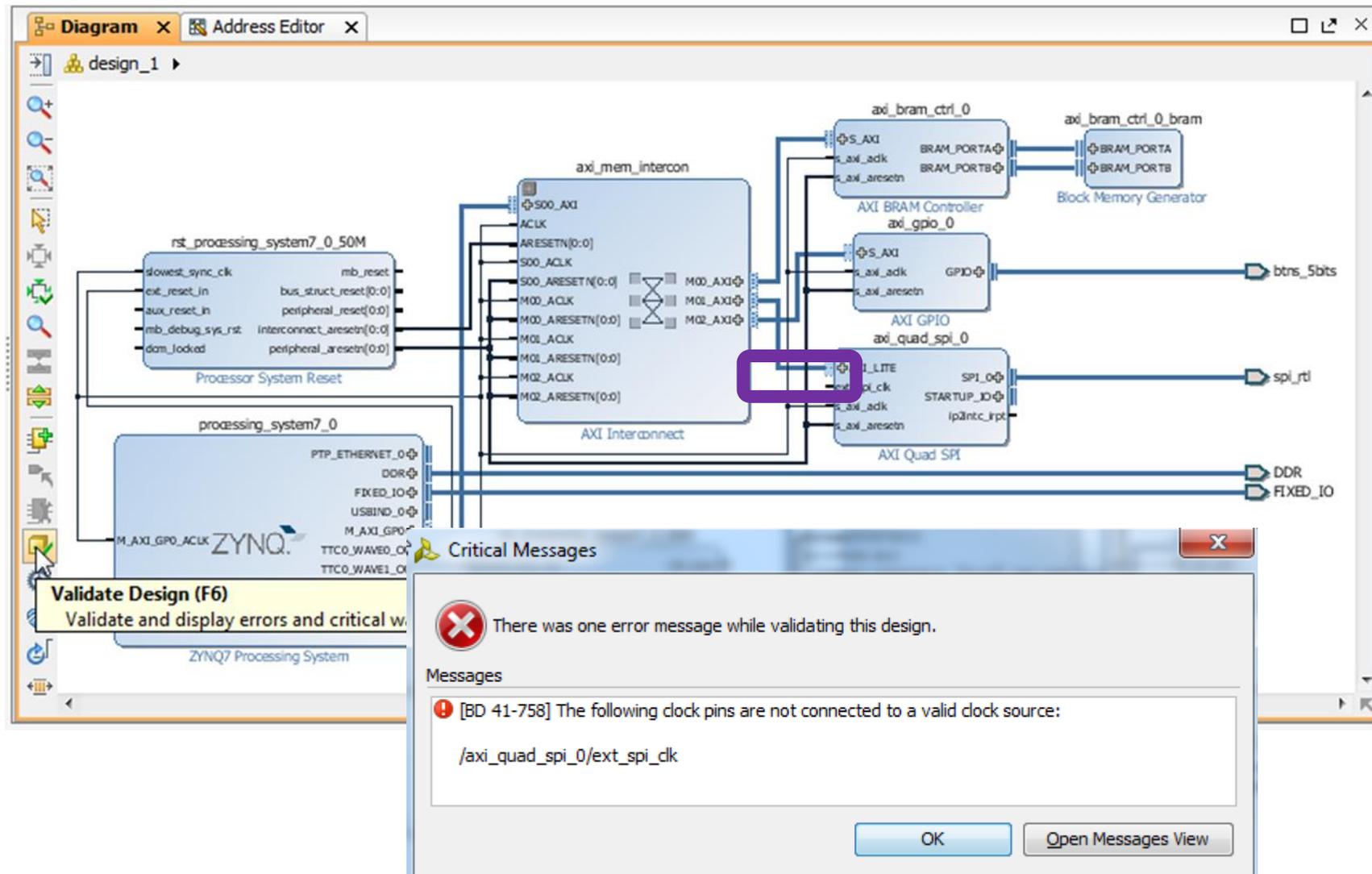
# Options for External Connections



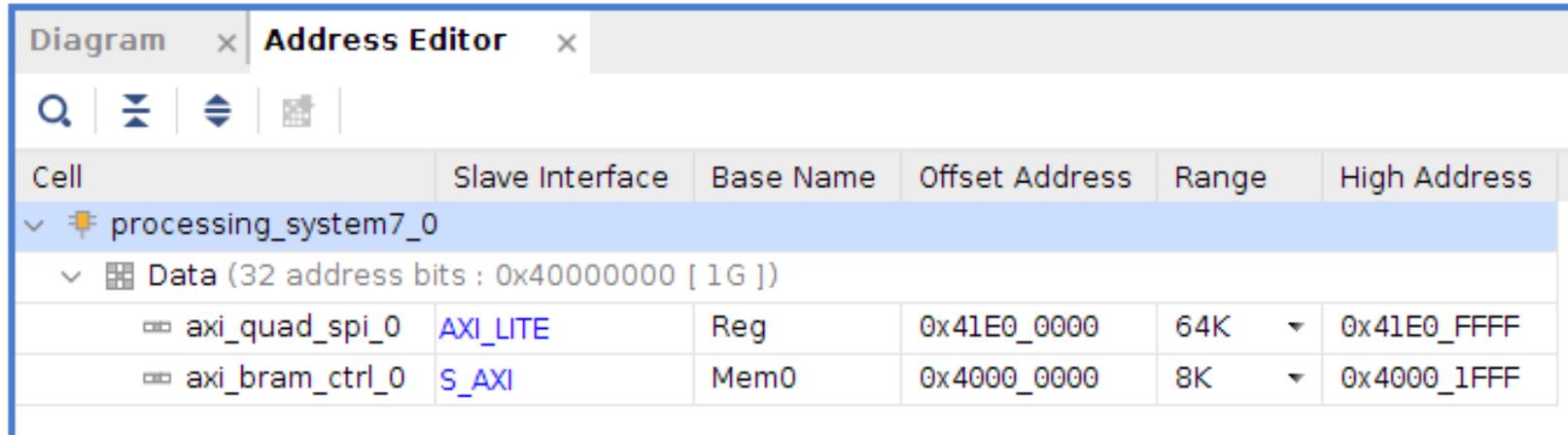
# DRC (Design Rule Check) Design Validation



# DRC – Design Validation



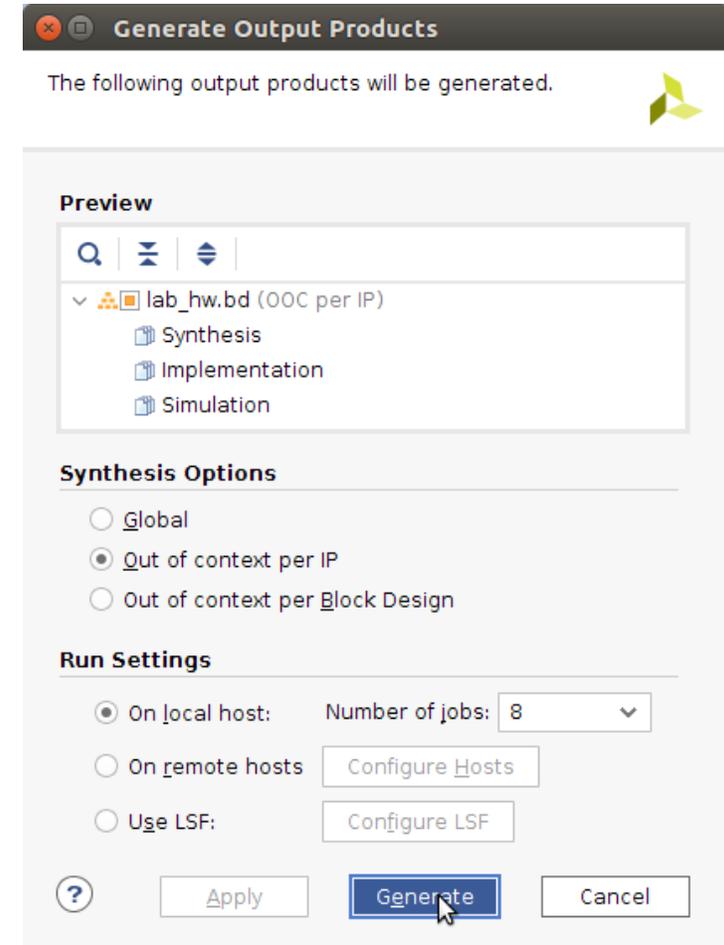
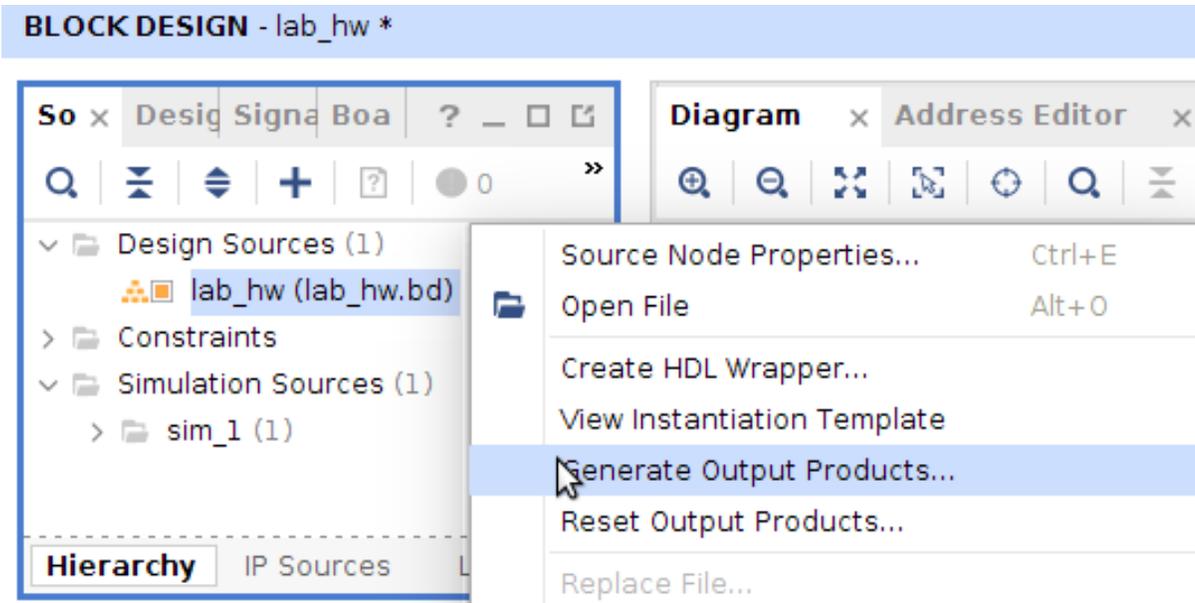
# Address Map



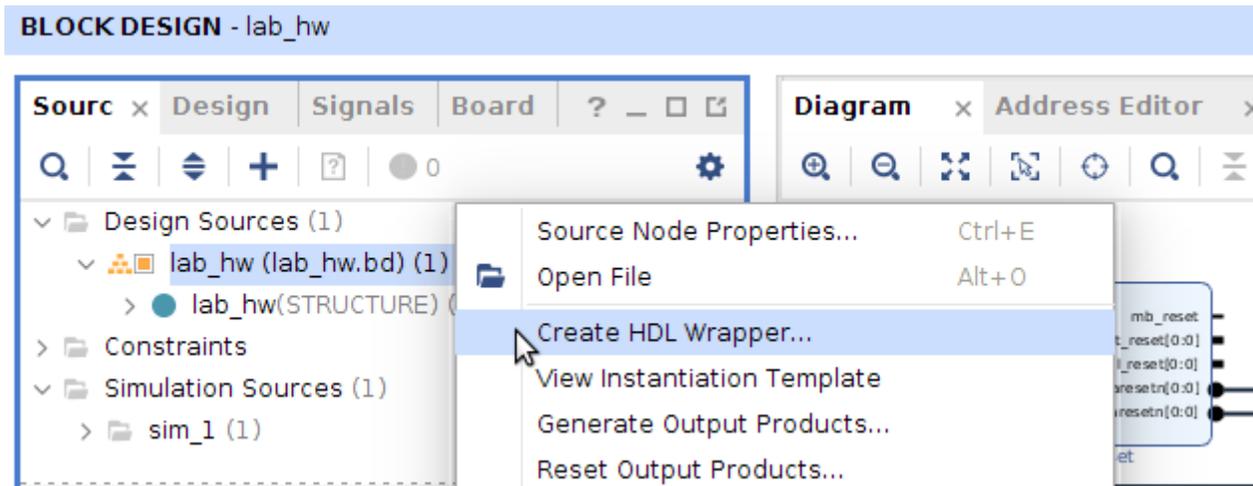
The screenshot shows the Vivado Address Editor window. The title bar includes 'Diagram' and 'Address Editor'. Below the title bar are icons for search, zoom, and other functions. The main content is a table with the following columns: Cell, Slave Interface, Base Name, Offset Address, Range, and High Address. The table is expanded to show the address map for 'processing\_system7\_0', which is further expanded to show 'Data (32 address bits : 0x40000000 [ 1G ])'.

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
processing_system7_0					
axi_quad_spi_0	AXI_LITE	Reg	0x41E0_0000	64K	0x41E0_FFFF
axi_bram_ctrl_0	S_AXI	Mem0	0x4000_0000	8K	0x4000_1FFF

# Generating Output Products



# Creating an HDL Wrapper



# Project Data and Directories

---

All project data is stored in a *project\_name* directory containing the following directories

- ***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.

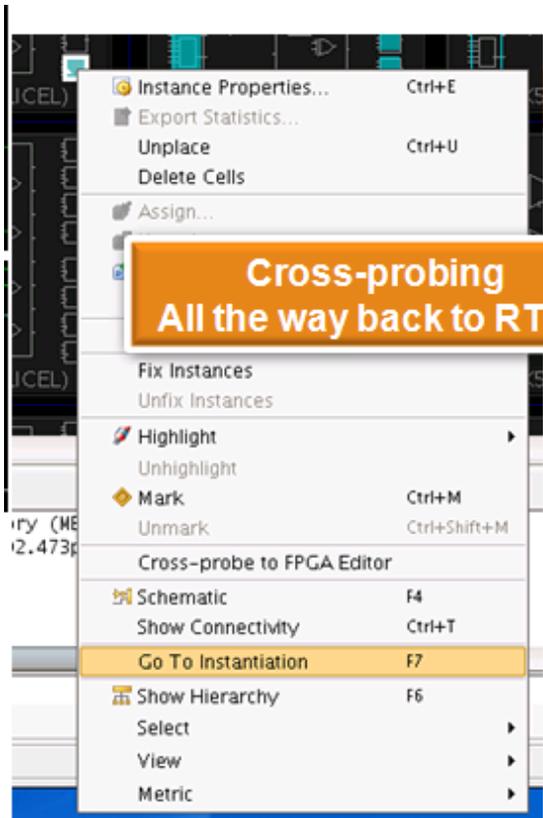
## Location

- 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

# Vivado Visualization Features

Visualize and debug a design at any flow stage

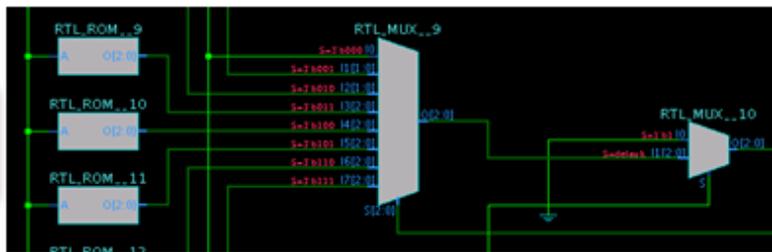
- Cross-probing between netlist/schematic/RTL



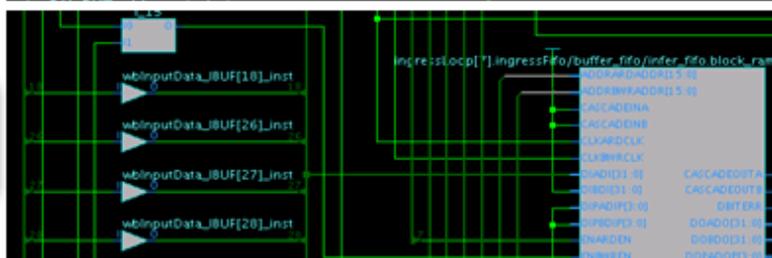
The screenshot shows a context menu in Vivado with the following items:

- Instance Properties... (Ctrl+E)
- Export Statistics...
- Unplace (Ctrl+U)
- Delete Cells
- Assign...
- Cross-probing All the way back to RTL!!**
- Fix Instances
- Unfix Instances
- Highlight
- Unhighlight
- Mark (Ctrl+M)
- Unmark (Ctrl+Shift+M)
- Cross-probe to FPGA Editor
- Schematic (F4)
- Show Connectivity (Ctrl+T)
- Go To Instantiation (F7)**
- Show Hierarchy (F6)
- Select
- View
- Metric

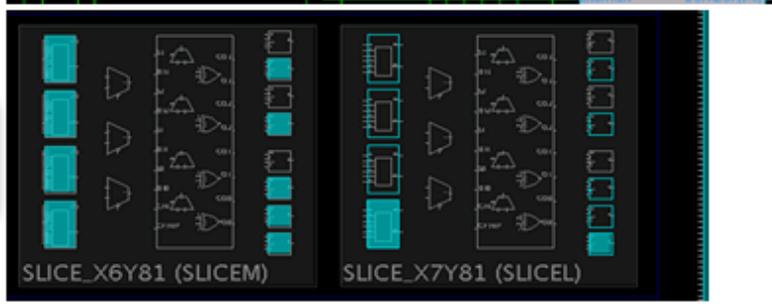
**Elaborated RTL**



**Netlist Schematic**

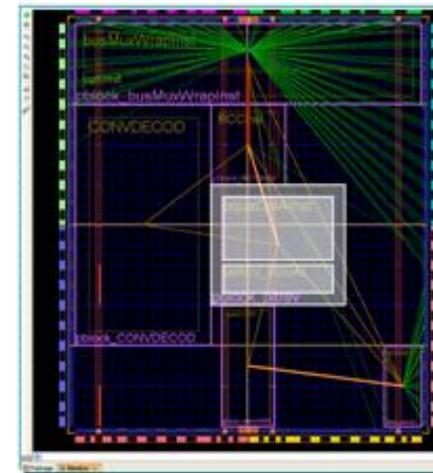
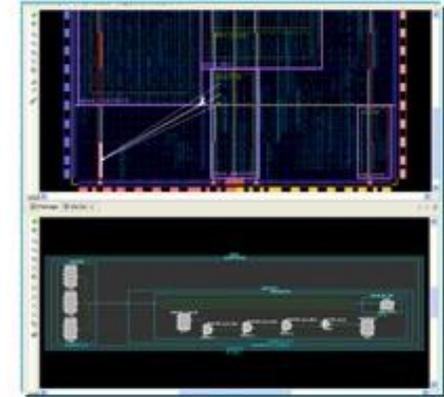


**Implemented Design**



# Gain Faster Timing Closure

- **Analyze multiple implementation results**
  - Highlight failing timing paths from post-route timing
  - Quickly identify and constrain critical logic path
- **Connectivity display**
  - I/Os, net bundles, clock domains
- **Hierarchical floorplanning**
  - Guide *place & route* toward better results
- **Utilization estimates**
  - All resource types shown for each Pblock
  - Clocks or carry chains



# Tool Command Line (.tcl) Features

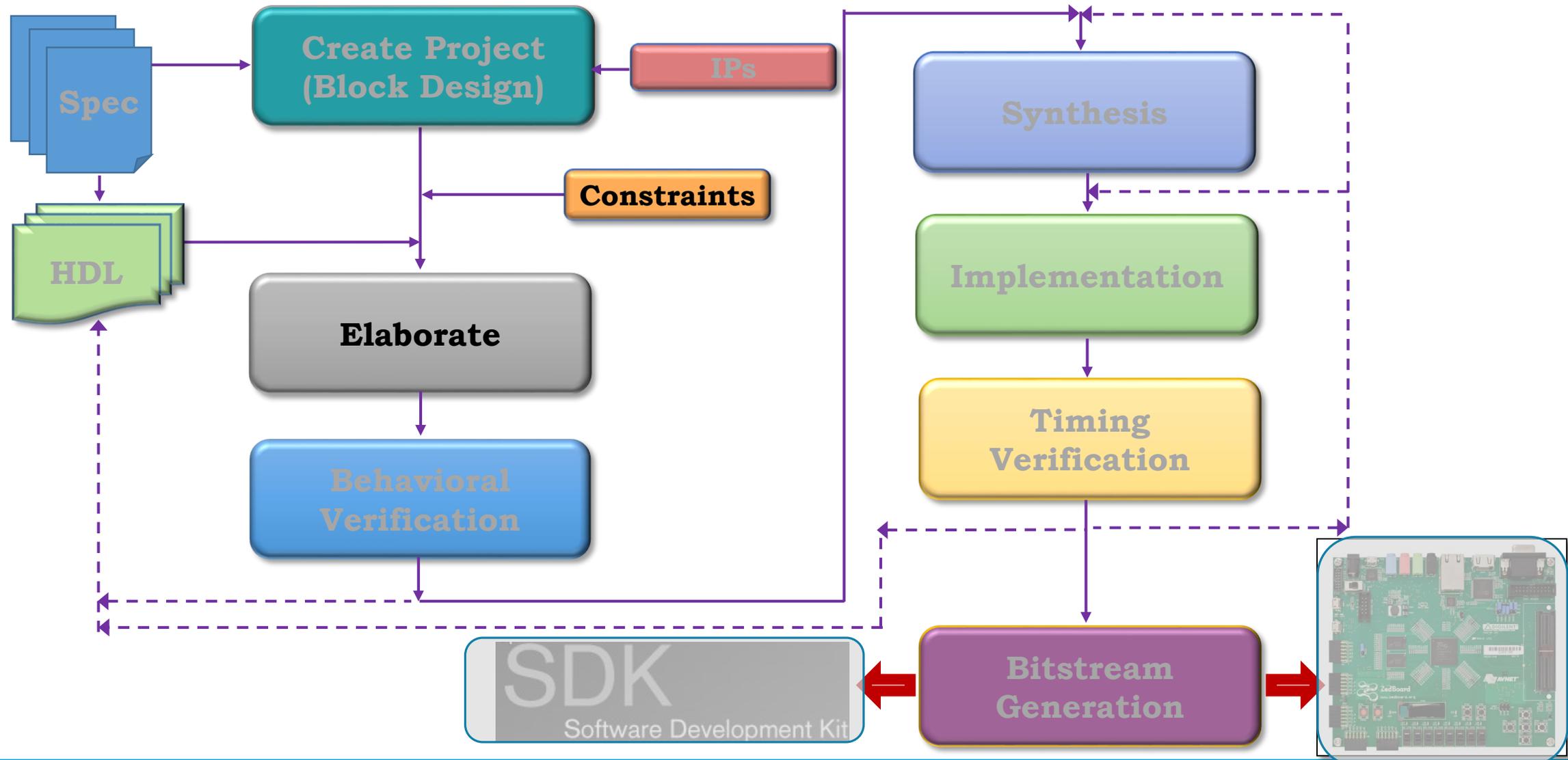
---

- ❖ 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

# *Vivado Design Suite* *Elaboration Process*

---

# Embedded System Design – Vivado Flow



# Elaboration

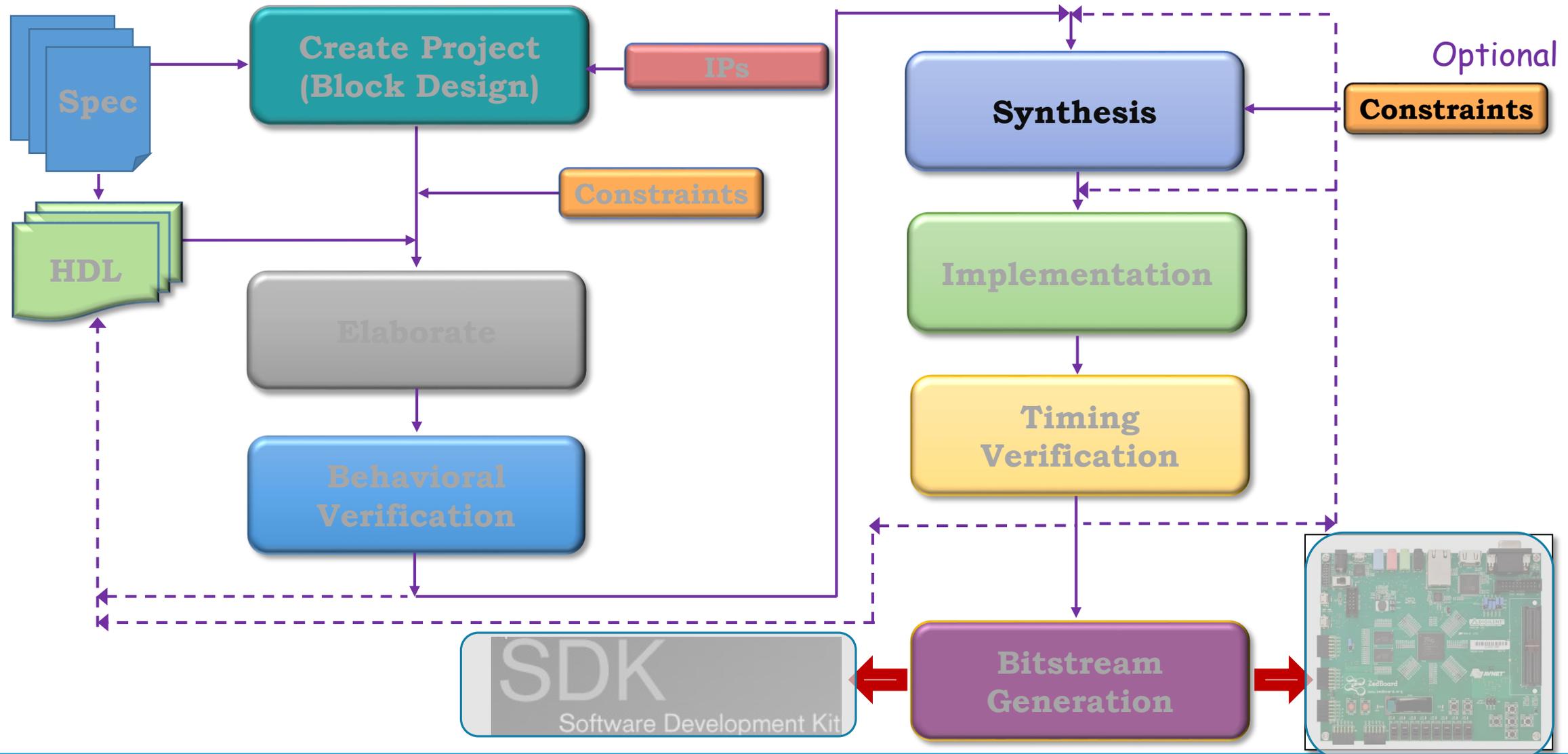
---

- Elaboration is the RTL optimization to an FPGA technology
- Vivado IDE allows designers to import and manage RTL sources
  - Verilog, System Verilog, VHDL, NGC, or testbenches
- Create and modify sources with the RTL Editor
  - Cross-selection between all the views
- Sources view
  - Hierarchy view: Display the modules in the design by hierarchy
  - Libraries view: Display sources by category

# *Vivado Design Suite* *Synthesis Process*

---

# Embedded System Design – Vivado Flow



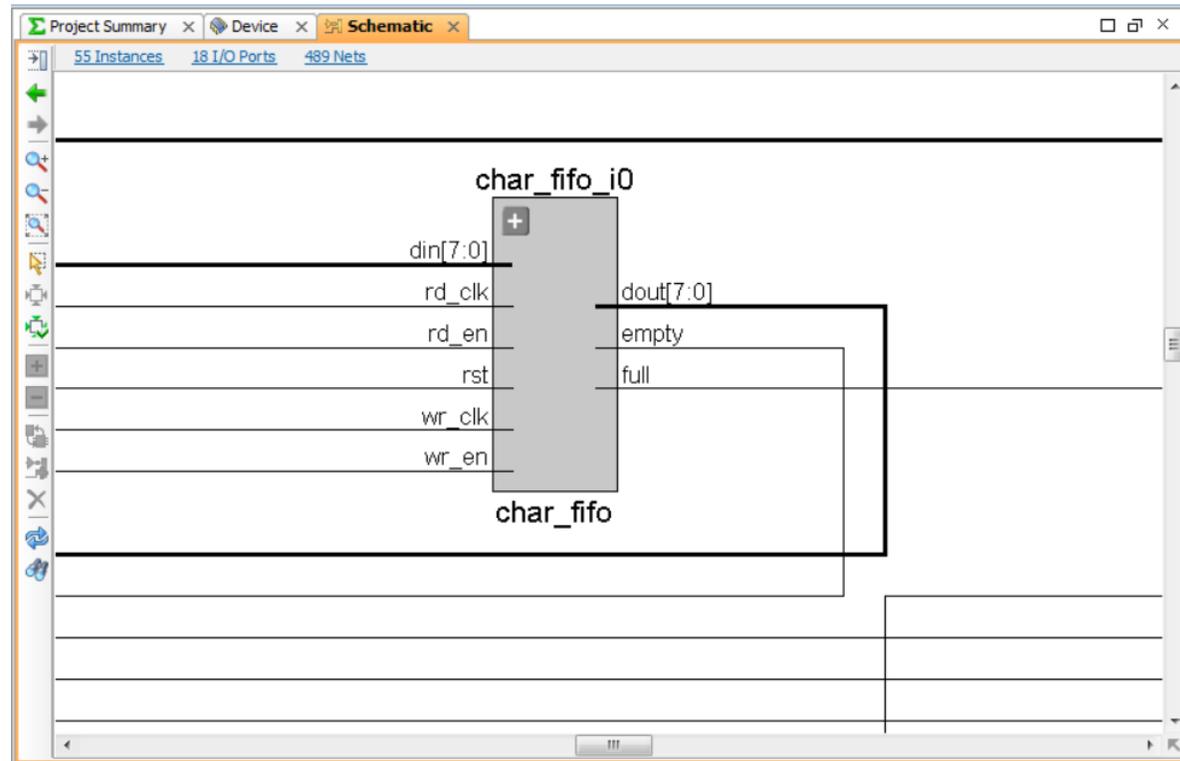
# Vivado IDE Synthesis

---

- **Applicable only for RTL (HDL) design flows**
  - EDIF is black boxed and linked after synthesis
- **Synthesis tool uses XDC constraints to drive synthesis optimization**
  - Design must first be synthesized without timing constraints for constraints editor usage
  - XDC file must exist
- **Synthesis settings provide access to additional options**

# Logic Optimization and Mapping to Device Primitives

Synthesis of an RTL design not only optimizes the gate-level design but also maps the netlist to Xilinx primitives (sometimes called technology mapping)



# Synthesized Design

---

**Accessed through the Flow Navigator by selecting Open Synthesized Design**

## **Representation of the design after synthesis**

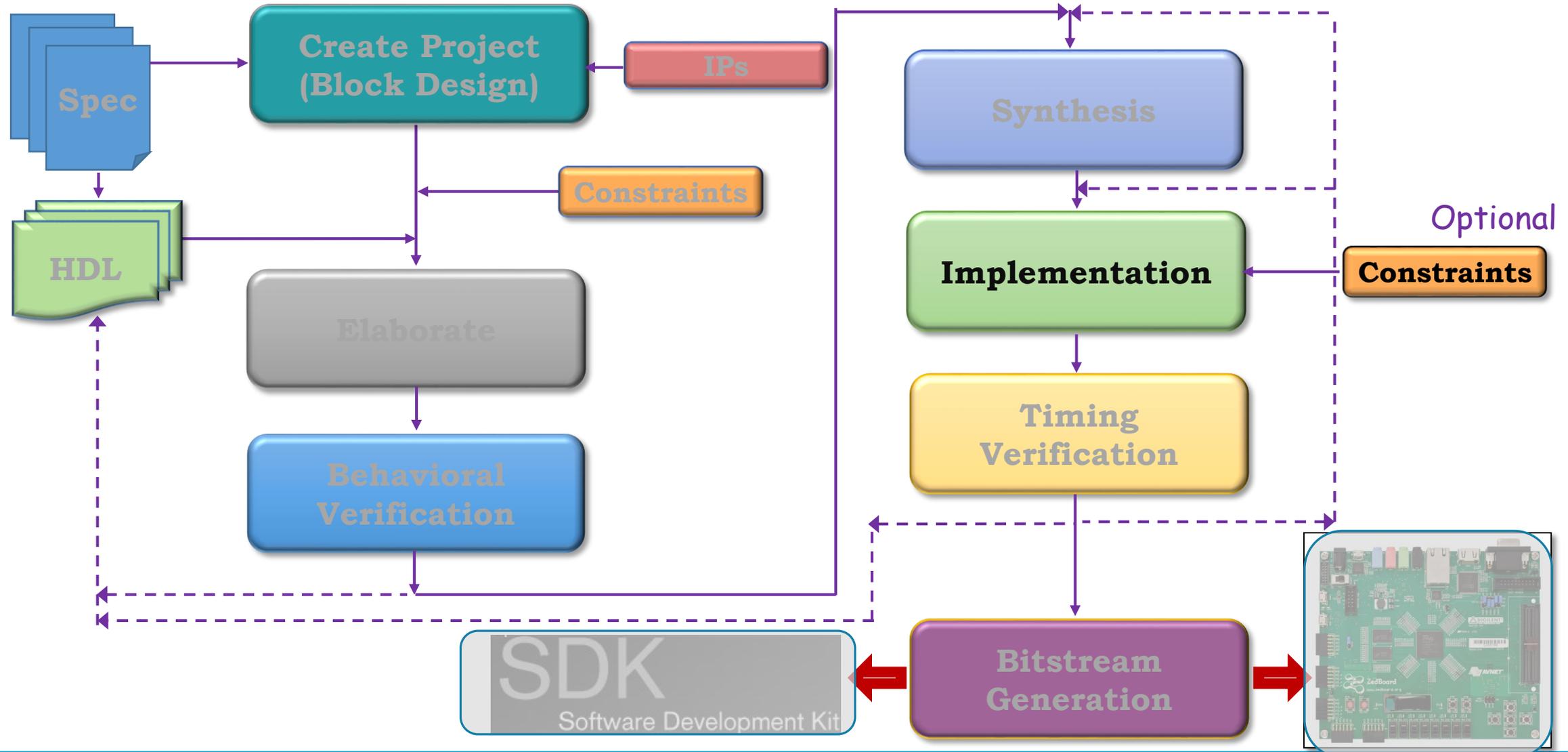
- Interconnected netlist of hierarchical and basic elements (BELs)
  - Instances of modules/entities
  - Basic elements
    - LUTs, flip-flops, carry chain elements, wide MUXes
    - Block RAMs, DSP cells
    - Clocking elements (BUFG, BUFR, MMCM, ...)
    - I/O elements (IBUF, OBUF, I/O flip-flops)

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

# *Vivado Design Suite* *Implementacion Process*

---

# Embedded System Design – Vivado Flow



# Vivado Implementation Sub-Processes

---

Vivado Design Suite Implementation process transform a logical netlist (generated by the synthesis tool) into a placed and routed design ready for bitstream generation

- **Opt design**
  - Optimizes the logical design to make it easier to fit onto the target FPGA
- **Place design**
  - Places the design onto the FPGA's logic cells
- **Route design**
  - Routing of connections between the FPGA's cells

# Using Design Constraints for Guiding Implementation

---

There are two types of design constraints, physical constraints and timing constraints.

**Physical Constraints:** define a relationship between logic design objects and device resources

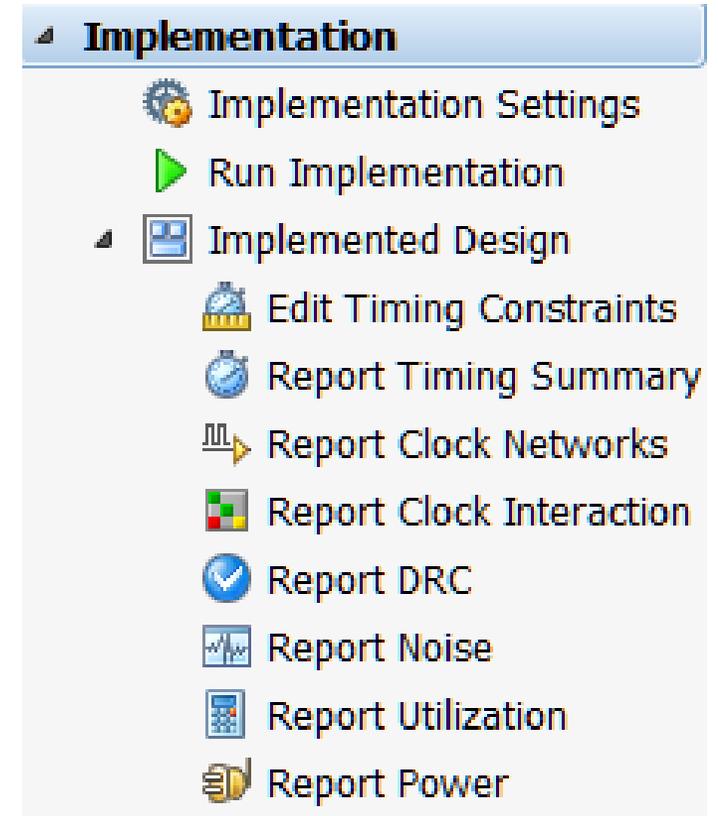
- Package pin placement
- Absolute or relative placement of cells:
  - Block RAM
  - DSP
  - LUTs
  - Flip-Flops
- Floorplanning constraints that assign cells to general regions of an FPGA

**Timing Constraints:** define the frequency requirements for the design. Without timing constraints, Vivado Design Suite optimizes the design solely for wire length and routing congestion and makes no effort to assess or improve design performance

# After Implementation

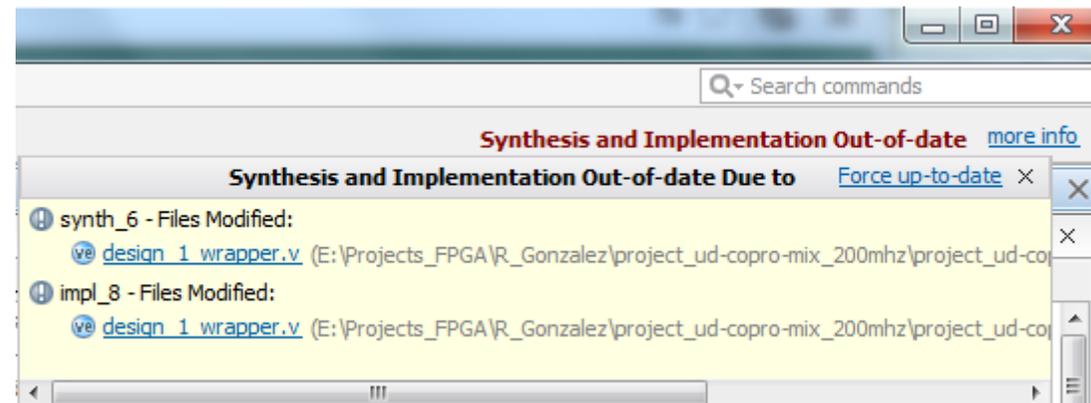
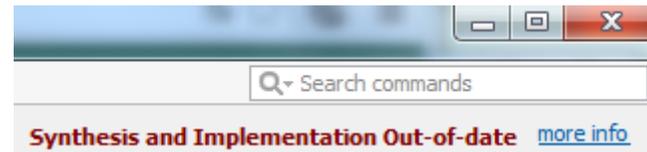
---

- Sources and Netlist tabs do not change
- Now as each resources is selected, it will show the exact placement of the resource on the die
- Timing results have to be generated with the Report Timing Summary
- 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



# Implementation Out-of-Date Message

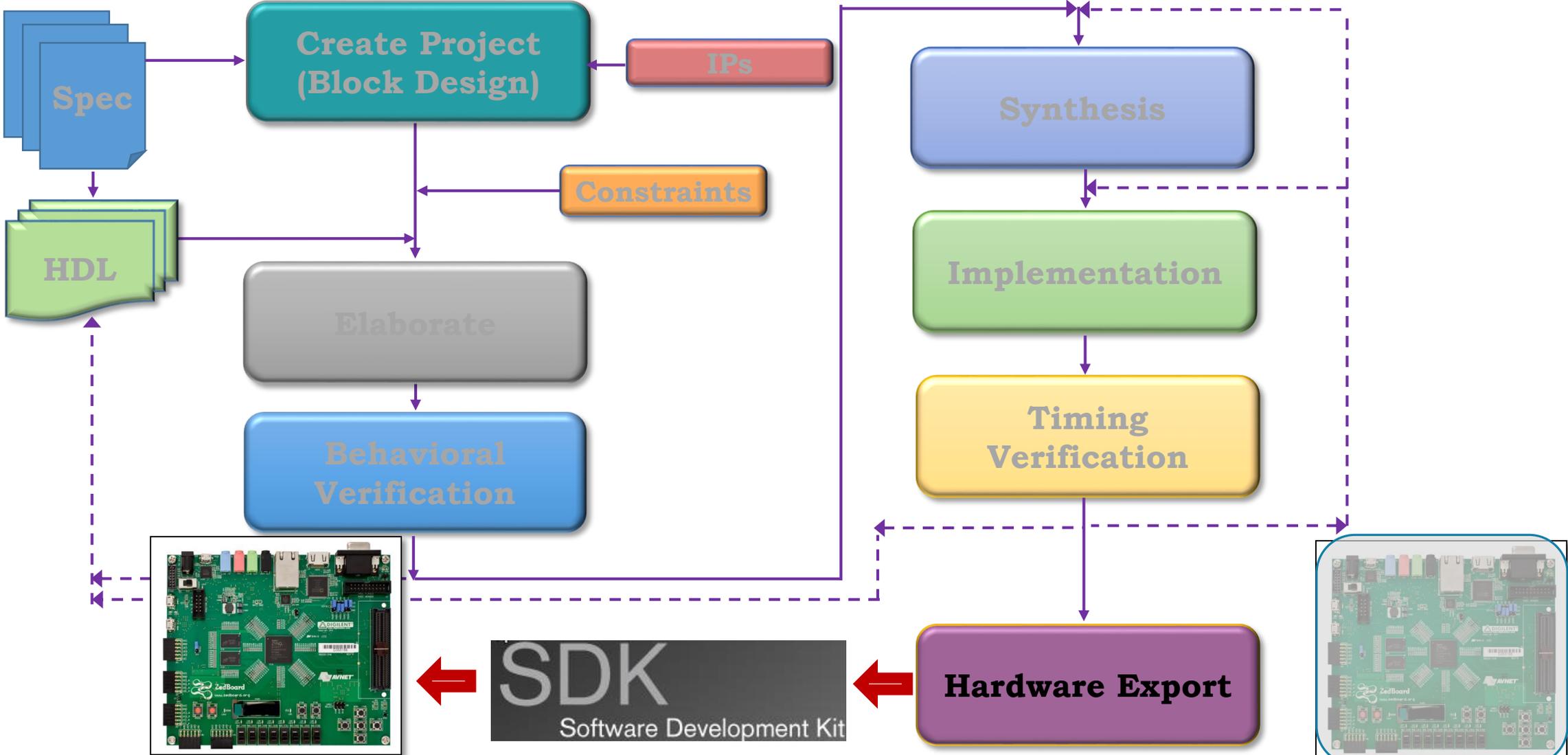
---



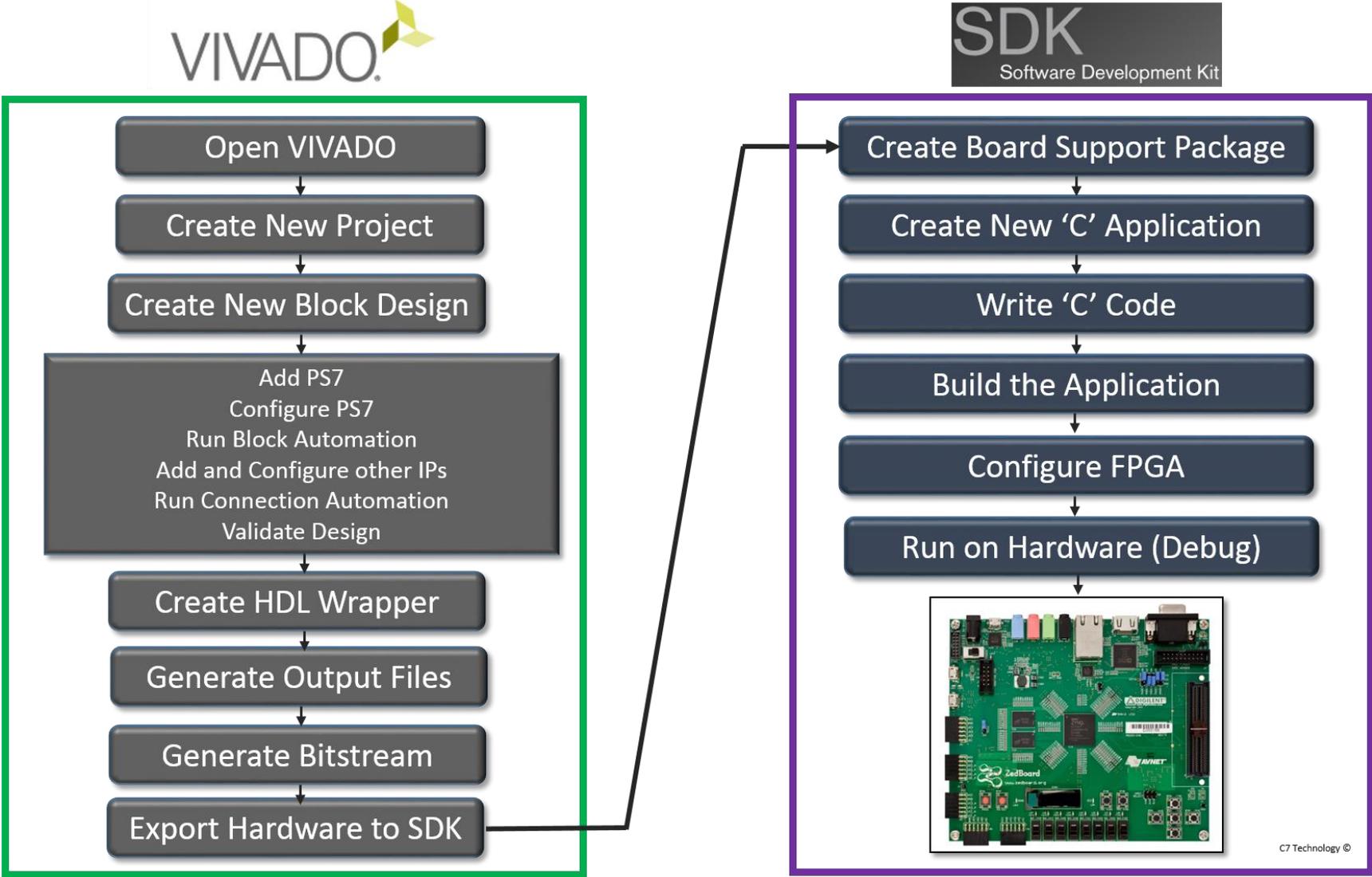
# Software Development Kit (SDK)

---

# Embedded System Design – Vivado-SDK Flow



# Embedded System Design – Vivado-SDK Flow



# Embedded System Tools: Software

---

## Eclipse IDE-based Software Development Kit (SDK)

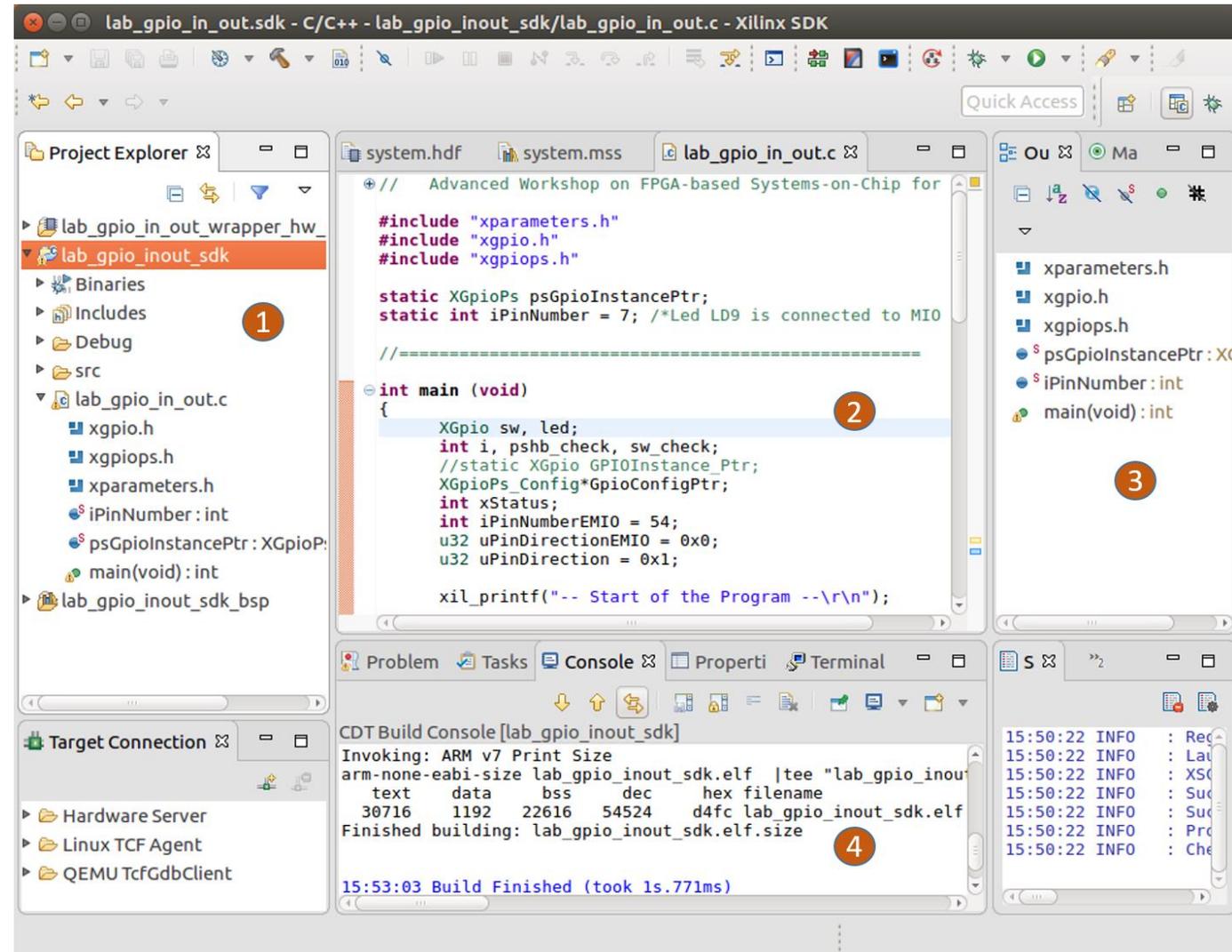
- Board support package creation : LibGen
- GNU software development tools
- C/C++ compiler for the ARM Cortex-A9 processor (gcc)
- Debugger for the ARM Cortex-A9 processor (gdb)

## Board support packages (BSPs)

- Stand-alone BSP
  - Free basic device drivers and utilities from Xilinx
  - NOT an RTOS

# SDK Workbench Views

- ① C/C++ project outline displays the elements of a project with file decorators (icons) for easy identification
- ② C/C++ editor for integrated software creation
- ③ Code outline displays elements of the software file under development with file decorators (icons) for easy identification
- ④ Problems, Console, Properties views list output information associated with the software development flow



# Software Management Settings

Software is managed in three major areas

- Compiler/Linker Options
- Application program



The screenshot shows the 'Properties for lab\_gpio\_inout\_sdk' dialog box. The 'Settings' tab is active, displaying a tree view of settings categories on the left and configuration options on the right. The 'Settings' category is selected in the tree view. The configuration options include:

- Configuration: Debug [ Active ] (with a 'Manage Configurations...' button)
- Tool Settings: ARM v7 gcc assembler (selected), ARM v7 gcc compiler, ARM v7 gcc linker
- Command: arm-none-eabi-gcc
- All options: (empty text area)
- Expert settings: Command line pattern: \${COMMAND} -c \${FLAGS} \${OUTPUT\_}

Buttons for 'Cancel' and 'OK' are visible at the bottom right of the dialog box.

# Software Management Settings

Software is managed in three major areas

- Software Platform Settings
- Board support package



**Board Support Package Settings**  
Control various settings of your Board Support Package.

**Overview**

- standalone
- drivers
  - ps7\_cortexa9\_0

**lab\_gpio\_inout\_sdk\_bsp**

OS Type: *standalone*

OS Version: 6.7

Standalone is a simple, low-level software layer. It provides access to basic processor features such as caches, interrupts and exceptions as well as the basic features of a hosted environment, such as standard input and output, profiling, abort and exit.

**Target Hardware**

Hardware Specification: /cris\_projects/ictp\_labs/lab3/lab\_gpio\_in\_out/lab\_gpio\_in\_out.sdk/lab

Processor: ps7\_cortexa9\_0

**Supported Libraries**

Check the box next to the libraries you want included in your Board Support Package. You can configure the library in the navigator on the left.

Name	Version	Description
<input type="checkbox"/> libmetal	1.4	Libmetal Library
<input type="checkbox"/> lwip202	1.1	Lwip202 library: lwIP (light weight IP) is an open source TCP/IP
<input type="checkbox"/> openamp	1.5	OpenAmp Library
<input type="checkbox"/> xilffs	3.9	Generic Fat File System Library

Cancel OK

# Software Management Settings

Software is managed in three major areas

- Software Platform Settings
  - Board support package



**Generate a linker script**

Generate linker script  
Control your application's memory map.

**Output Settings**

Project: lab\_gpio\_inout\_sdk  
Output Script:  
lab\_gpio\_inout\_sdk/src/lscript.ld   
Modify project build settings as follows:  
Set generated script on all project build configurations

**Hardware Memory Map**

Memory	Base Address	Size
ps7_ddr_0	0x00100000	511 MB
ps7_ram_0	0x00000000	192 KB
ps7_ram_1	0xFFFF0000	~63.5 KB

Fixed Section Assignments

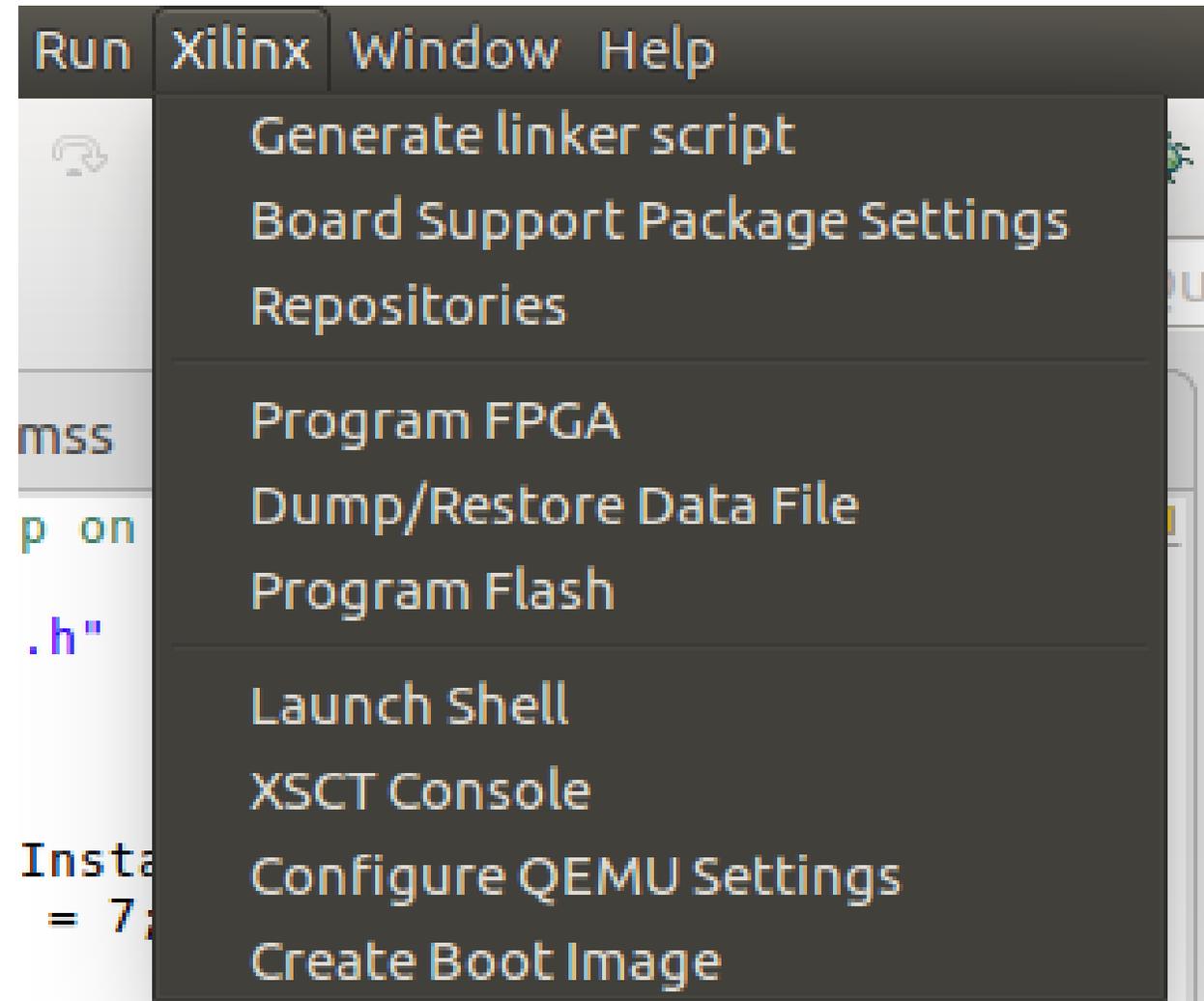
Basic Advanced

Place Code Sections in: ps7\_ddr\_0  
Place Data Sections in: ps7\_ddr\_0  
Place Heap and Stack in: ps7\_ddr\_0  
Heap Size: 1 KB  
Stack Size: 1 KB

# Integrated Xilinx Tools in the SDK

## Xilinx additions to the Eclipse IDE

- BSP Settings
- Software Repositories
- Generate Linker Script
- Program the programmable logic
  - Bitstream must be available
- Create Zynq Boot Image
- Program Flash Memory
- Launch XMD Console
- Launch Shell
- Configure JTAG Settings
- SysGen Co-Debug Settings





# Apendix

---

# *Vivado Design Suite*

## *Basic Static Timing Constraints*

---

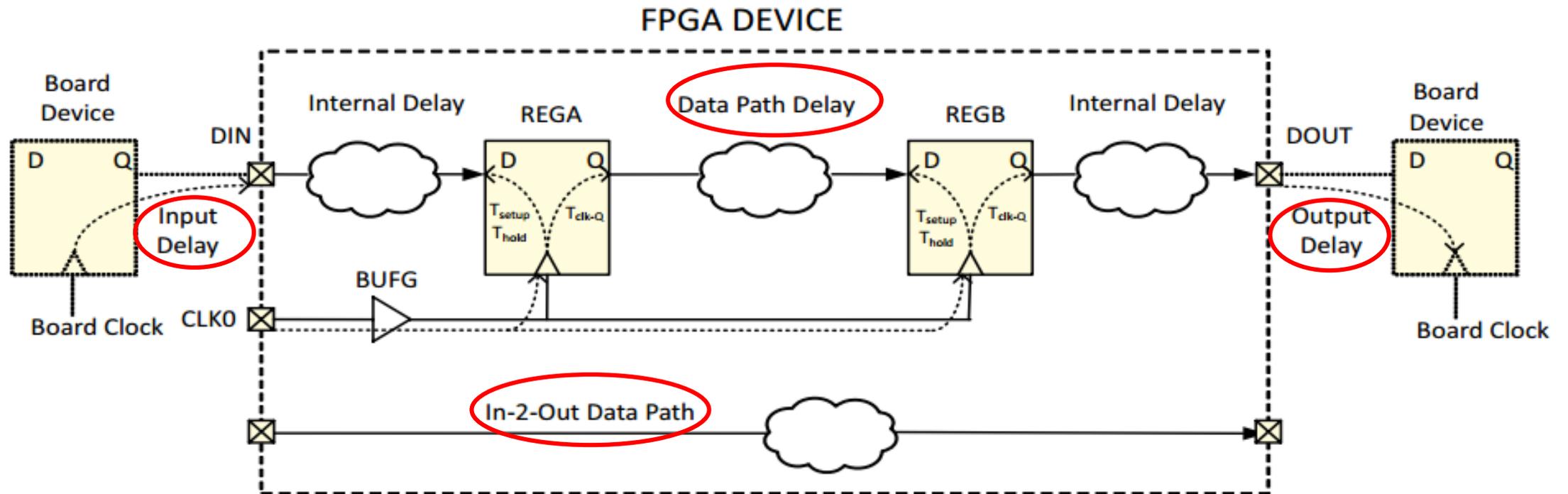
# Basic Timing Constraints

---

There are three basic timing constraints applicable to a sequential machine

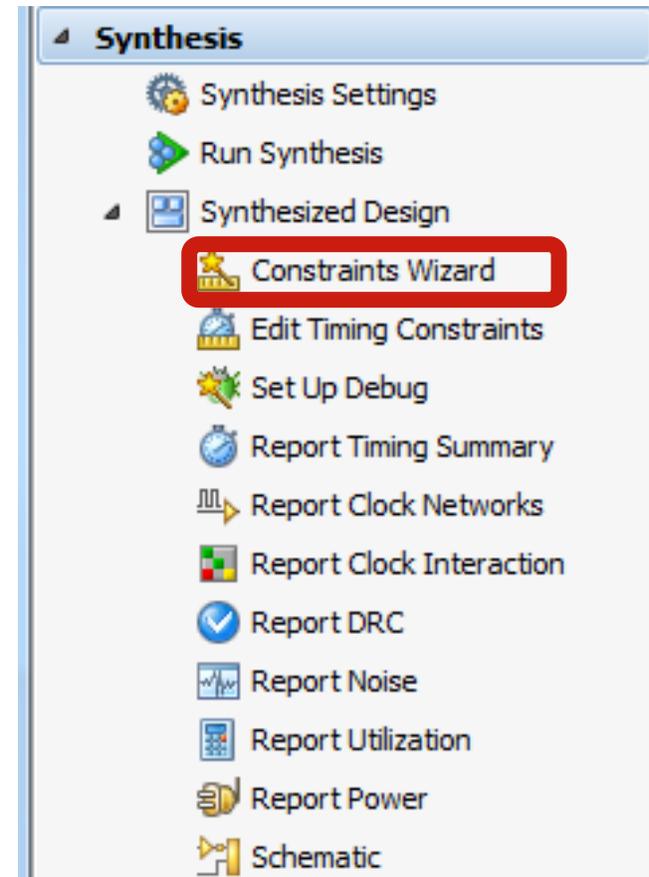
- Period
  - Paths between synchronous elements clocked by the reference clock net
    - Synchronous elements include flip-flops, latches, synchronous RAM, and DSP slices
  - Use `create_clock` to create the constraint
- Input Delay
  - Paths between input pin and synchronous elements
  - Use `set_input_delay` to create the constraint
- Output delay
  - Paths between synchronous elements and output pin
  - Use `set_output_delay` to create the constraint

# Timing Paths Example



# Creating Basic Timing Constraints in Vivado IDE

1. Run Synthesis
2. Open the synthesized design
3. Invoke constraints editor



# Clock Constraint Setting

**Timing Constraints Wizard**

**Primary Clocks**

Primary clocks usually enter the design through input ports. Specify the period and optionally a name and waveform (rising and falling edge times) to describe the duty cycle if not 50%. [More info](#)

Recommended Constraints

<input checked="" type="checkbox"/>	Object	Name	Frequency (MHz)	Period (ns)	Rise At (ns)	Fall At (ns)	Jitter (ns)
<input checked="" type="checkbox"/>	clk	clk	undefined	undefined			

Constraints for Pulse Width Check Only

<input type="checkbox"/>	Object	Name	Frequency (MHz)	Per
--------------------------	--------	------	-----------------	-----

Tcl Command Preview (1) Existing Create Clock Constraints (0)

```
create_clock -name clk [get_ports {clk}]
```

Reference < Back Next > Skip to Finish >> Cancel

**Timing Constraints Wizard**

**Primary Clocks**

Primary clocks usually enter the design through input ports. Specify the period and optionally a name and waveform (rising and falling edge times) to describe the duty cycle if not 50%. [More info](#)

Recommended Constraints

<input checked="" type="checkbox"/>	Object	Name	Frequency (MHz)	Period (ns)	Rise At (ns)	Fall At (ns)	Jitter (ns)
<input checked="" type="checkbox"/>	clk	clk					

Constraints for Pulse Width Check Only

<input type="checkbox"/>	Object	Name	Frequency (MHz)	Per
--------------------------	--------	------	-----------------	-----

Tcl Command Preview (1) Existing Create Clock Constraints (0)

```
create_clock -name clk [get_ports {clk}]
```

Reference < Back Next > Skip to Finish >> Cancel

**Primary Clock Constraints**

Specify the 'period' or 'frequency', and optionally the 'rise at', 'fall at' and 'jitter' values.

Frequency:  MHz (required)

Period:  ns (required)

Rise at:  ns (optional)

Fall at:  ns (optional)

Jitter:  ns (optional)

OK Cancel

# Clock Constraint Setting

**Timing Constraints Wizard**

**Primary Clocks**

Primary clocks usually enter the design through input ports. Specify the period and optionally a name and waveform (rising and falling edge times) to describe the duty cycle if not 50%. [More info](#)

Recommended Constraints

<input checked="" type="checkbox"/>	Object	Name	Frequency (MHz)	Period (ns)	Rise At (ns)	Fall At (ns)	Jitter (ns)
<input checked="" type="checkbox"/>	clk	clk	100.000	10.000	0.000	5.000	

Constraints for Pulse Width Check Only

<input type="checkbox"/>	Object	Name	Frequency (MHz)	Period (ns)
--------------------------	--------	------	-----------------	-------------

Tcl Command Preview (1) Existing Create Clock Constraints (0)

```
create_clock -period 10.000 -name clk -waveform {0.000 5.000} [get_ports {clk}]
```

Reference < Back Next > Skip to Finish >> Cancel

```
Project Summary x Device x Schematic x Schematic (2) x Schematic (3) x t.xdc x  
E:/Projects_FPGA/Proyecto_Clevis/Controlador_X/project_1/project_1.srcs/constrs_1/new/t.xdc  
1 create_clock -period 10.000 -name clk -waveform {0.000 5.000} [get_ports clk]  
2
```

# Clock Network Report

The screenshot displays the Vivado Clock Networks report for a specific network. On the left, the 'Synthesis' menu is open, showing various options like 'Synthesis Settings', 'Run Synthesis', and 'Report Clock Networks'. The main window shows a hierarchical tree for 'Clock Networks - network\_3'. The root node is 'clk (100.00 MHz) (drives 48 loads)'. Below it, the hierarchy includes: 'clk' (input), 'I (clk\_IBUF\_inst/I)', 'clk\_IBUF\_inst (IBUF)', 'O (clk\_IBUF\_inst/O)', 'clk\_IBUF (clk\_IBUF)', 'I (clk\_IBUF\_BUFG\_inst/I)', 'clk\_IBUF\_BUFG\_inst (BUFG)', 'O (clk\_IBUF\_BUFG\_inst/O)', 'clk\_IBUF\_BUFG (clk\_IBUF\_BUFG)', and 'FDCE (48 loads)'. The bottom of the window shows a tabbed interface with 'Clock Networks' selected.

# Clock Network Report and Visualization

The image displays the Vivado software interface. On the left, the 'Implementation' tree is visible, with 'Report Clock Networks' highlighted in a red box. A tooltip for this option reads: 'Report Clock Networks Specify analysis options and create a clock networks report.' The main workspace shows a 'Clock Networks - network\_1' hierarchy. A red box highlights the top-level entry: 'clk (100.00 MHz) (drives 48 loads)'. Below it, the hierarchy includes 'I (clk\_IBUF\_inst/I)', 'O (clk\_IBUF\_inst/O)', 'I (clk\_IBUF\_BUFG\_inst/I)', and 'O (clk\_IBUF\_BUFG\_inst/O)', all leading to 'FDCE (48 loads)'. The bottom part of the image shows a 'Clock Networks' visualization window with a dark background and a grid. A white cone-shaped visualization represents the clock network, with a label 'X1Y1' in the top right corner.

# Clocking Resources: MMCM and PLL

Up to 24 CMTs per device

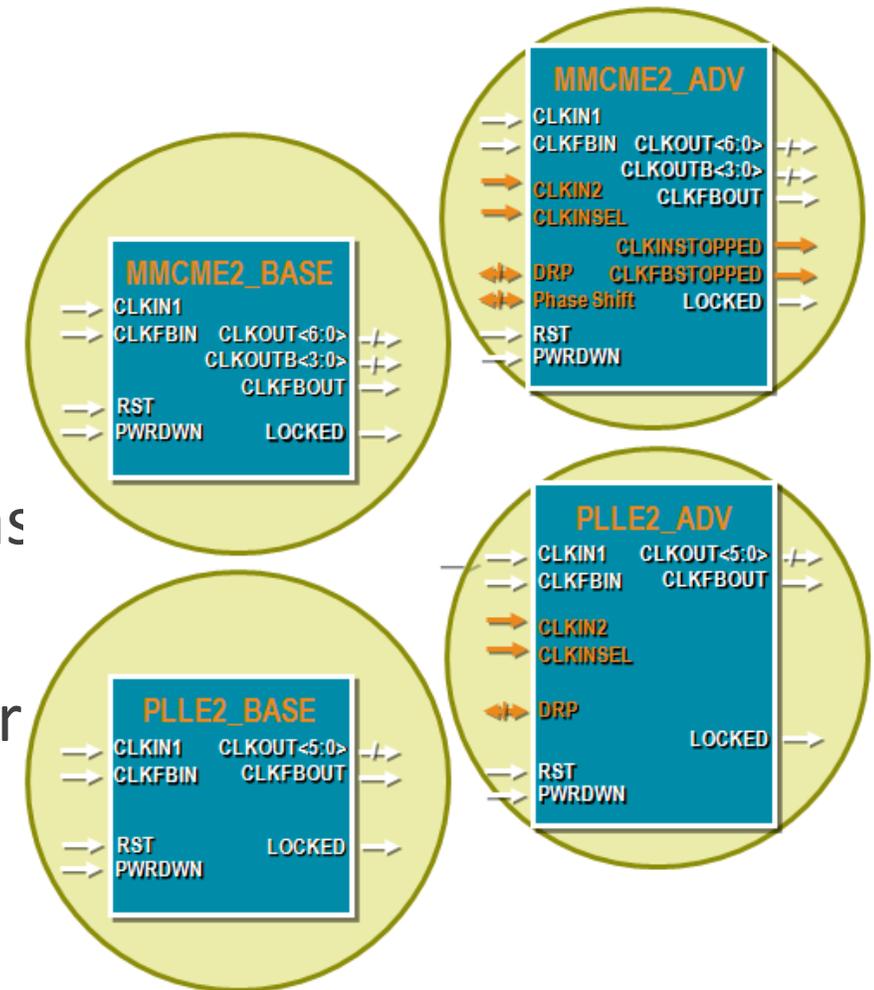
One MMCM and one PLL per CMT

Two software primitives (instantiation)

- \*\_BASE has only the basic ports
- \*\_ADV provides access to all ports

PLL is primarily intended for use with the I/O phases for high-speed memory controllers

The MMCM is the primary clock resource for user



# Inference

---

Clock networks are represented by nets in your RTL design

- The mapping of an RTL net to a clock network is managed by using the appropriate clock buffer to generate that net

Certain resources can be inferred

- A primary input net (with or without an IBUF instantiated) will be mapped to a global clock if it drives the clock inputs of clocked resources
  - The BUFG will be inferred
- BUFH drivers will be inferred whenever a global clock (driven by a BUFG) is required in a clock region
  - BUFHs for each region required will be inferred

BUFIO, BUFR, and BUFMR cannot be inferred

- Instantiating these buffers tells the tools that you want to use the corresponding clock networks

PLLs and MMCMs cannot be inferred

# Instantiation

---

All clocking resources can be directly instantiated in your RTL code

- Simulation models exist for all resources
- Refer to the Library Guide for HDL Designs
- Use the Language Templates (  ) tab

PLLs and MMCMs have many inputs and outputs, as well as many attributes

- Optimal dividers for obtaining the desired characteristics may be hard to derive
- The Clocking Wizard via the IP Catalog
  - Only \*\_ADV available

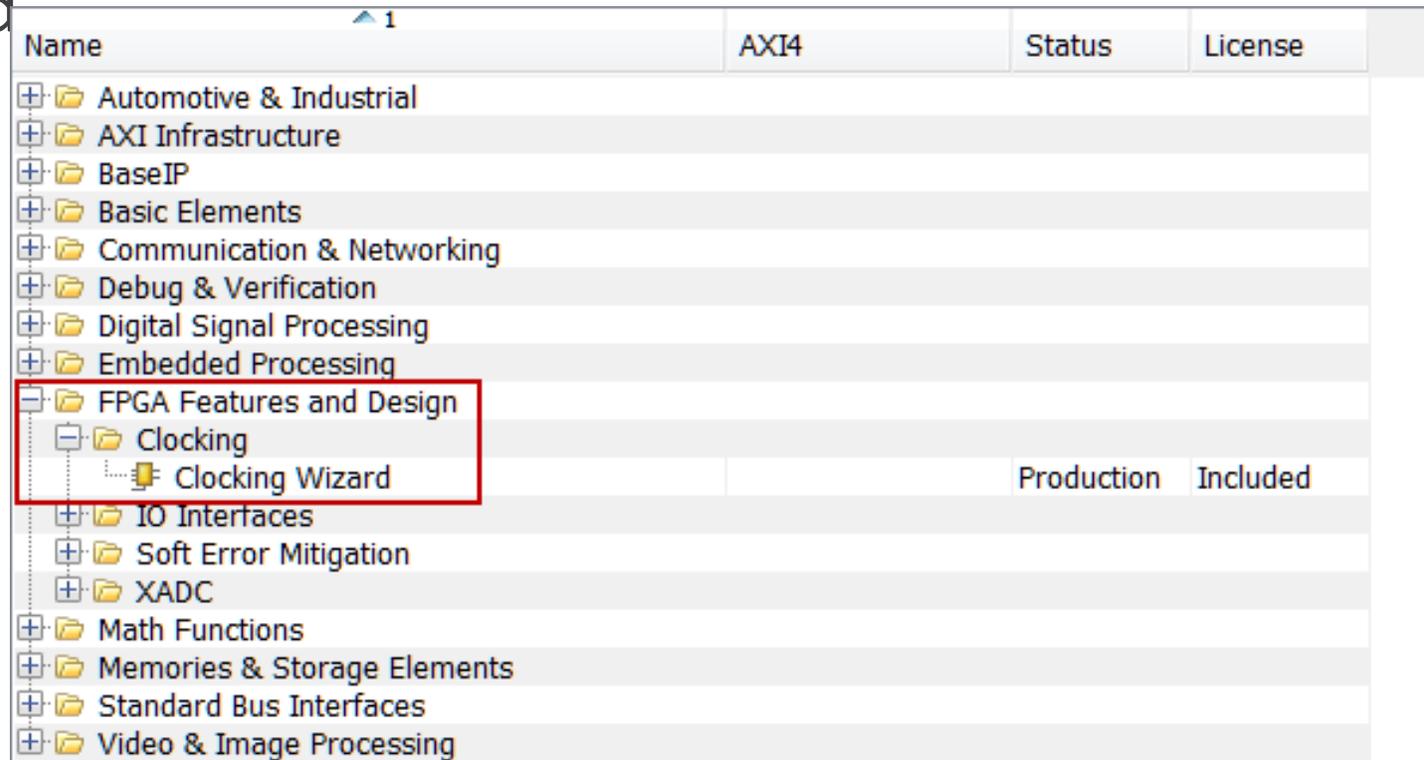
# Invoking Clocking Wizard

Click on the IP Catalog

Expand FPGA Features and Design > Clocking

Double-click on Clocking Wizard

The Clocking Wizard walks you through the generation of complete clocking subsystems



The screenshot shows a tree view of the IP Catalog. The 'FPGA Features and Design' folder is expanded, and the 'Clocking' folder is also expanded. The 'Clocking Wizard' is highlighted with a red box. The table below shows the details for the 'Clocking Wizard'.

Name	AXI4	Status	License
Automotive & Industrial			
AXI Infrastructure			
BaseIP			
Basic Elements			
Communication & Networking			
Debug & Verification			
Digital Signal Processing			
Embedded Processing			
FPGA Features and Design			
Clocking			
Clocking Wizard		Production	Included
IO Interfaces			
Soft Error Mitigation			
XADC			
Math Functions			
Memories & Storage Elements			
Standard Bus Interfaces			
Video & Image Processing			

# The Clocking Wizard: Clocking Options

Select Primitives to be used

- MMCME2\_ADV
- PLLE2\_ADV

Specify the primary input frequency and source type

- Optionally, select and specify secondary input

Select clocking features

- Frequency synthesis
- Phase alignment
- Dynamic phase shift
- ...

Switch to Defaults

Component Name: clk\_wiz\_0

**Clocking Options** | Output Clocks | MMCM Settings | Port Renaming | Summary

Primitive

MMCME2 ADV  PLLE2 ADV

Clocking Features

Frequency Synthesis  Spread Spectrum

Phase Alignment  Minimize Power

Dynamic Phase Shift  Dynamic Reconfiguration

Safe Clock Startup

Jitter Optimization

Balanced

Minimize Output Jitter

Maximize Input Jitter filtering

Input Clock Information

	Input Clock	Input Frequency(MHz)		Jitter Options	Input Jitter	Source
<input type="checkbox"/>	Primary	100.000	10.000 - 800.000	UI	0.010	Single ended clock capable pin
<input type="checkbox"/>	Secondary	100.000	50.000 - 200.000		0.010	Single ended clock capable pin

# The Clocking Wizard: Output Clocks

- Select the desired number of output clocks
- Set the desired output frequencies
- Select optional ports

The screenshot shows the 'Clocking Wizard (5.0)' window with the 'Output Clocks' tab selected. The component name is 'clk\_core'. The main table lists output clocks from 'clk\_out1' to 'clk\_out7'. 'clk\_out2' is selected with a checkmark. The 'Output Freq (MHz)' column shows 'Requested' and 'Actual' values, both set to 100.000 for the selected clock. The 'Phase (degrees)' column shows 'Requested' and 'Actual' values, both set to 0.000. The 'Duty Cycle (%)' column shows 'Requested' and 'Actual' values, both set to 50.000. The 'Drives' column is set to 'BUFG' and 'Use Fine PS' is unchecked. Below the table, there are sections for 'USE CLOCK SEQUENCING', 'Clocking Feedback' (with 'Automatic Control On-Chip' selected), and 'Reset Type' (with 'Active High' selected). The 'Enable Optional Inputs / Outputs' section has 'reset' and 'locked' checked.

Output Clock	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)		Drives	Use Fine PS
	Requested	Actual	Requested	Actual	Requested	Actual		
<input type="checkbox"/> clk_out1	100.000	100.000	0.000	0.000	50.000	50.0	BUFG	<input type="checkbox"/>
<input checked="" type="checkbox"/> clk_out2	100.000	100.000	0.000	0.000	50.000	50.0	BUFG	<input type="checkbox"/>
<input type="checkbox"/> clk_out3	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>
<input type="checkbox"/> clk_out4	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>
<input type="checkbox"/> clk_out5	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>
<input type="checkbox"/> clk_out6	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>
<input type="checkbox"/> clk_out7	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>

# The Clocking Wizard: Port Renaming

Change input/output port names

Change optional port names

The screenshot shows the 'Port Renaming' tab of the Clocking Wizard. It contains three sections: 'Input Clock', 'Output Clock', and 'Optional Port Names'. The 'Input Clock' section has a table with one row. The 'Output Clock' section shows 'VCO Freq = 1000.000 MHz' and a table with two rows. The 'Optional Port Names' section has a table with two rows.

Input Clock	Port Name	Freq (MHz)	Input Jitter (UI)
Primary	clk_in1	100.000	0.010

Output Clock  
VCO Freq = 1000.000 MHz

Output Clock	Port Name	Output Freq (MHz)	Phase (degrees)	Duty Cycle (%)
clk_out1	clk_out1	100.000	0.000	50.0
clk_out2	clk_out2	100.000	0.000	50.0

Optional Port Names

Other Pins	Port Name
reset	reset
locked	locked

# The Clocking Wizard: Summary

Shows the input, output frequencies

Other attributes depending on the selections made

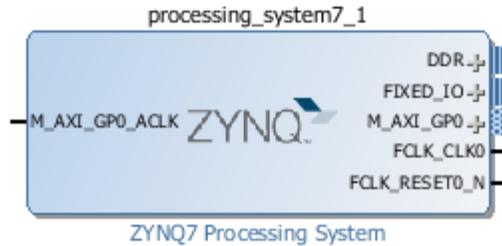
Attribute	Value
Input Clock (MHz)	100.000
Phase Shift	None
Divide Counter	1
Mult Counter	10.000
CLKOUT0 Divider	10.000
CLKOUT1 Divider	10
CLKOUT2 Divider	OFF
CLKOUT3 Divider	OFF
CLKOUT4 Divider	OFF
CLKOUT5 Divider	OFF
CLKOUT6 Divider	OFF

The Resource tab on the left provides summary of type and number of resources used

IP Symbol	Resource
1	MMCME2
1	IBUFG
3	BUFG

# Reset and Clock Topology

# Enabling Clock for PL



Re-customize IP

## ZYNQ7 Processing System (5.5)

Documentation Presets IP Location Import XPS Settings

Page Navigator << Zynq Block Design PS-PL Configuration Peripheral I/O Pins MIO Configuration **Clock Configuration** DDR Configuration SMC Timing Calculation Interrupts

### Clock Configuration [Summary Report](#)

Basic Clocking Advanced Clocking

Input Frequency (MHz) 33.333333 CPU Clock Ratio 6:2:1

Search: Q

Component	Clock Source	Requested Frequen...	Actual Frequency(M...	Range(MHz)
+ Processor/Memory Clocks				
+ IO Peripheral Clocks				
- PL Fabric Clocks				
<input checked="" type="checkbox"/> FCLK_CLK0	IO PLL	50	50.000000	0.100000 : 250.000000
<input type="checkbox"/> FCLK_CLK1	IO PLL	50	50.000000	0.100000 : 250.000000
<input type="checkbox"/> FCLK_CLK2	IO PLL	50	50.000000	0.100000 : 250.000000
<input type="checkbox"/> FCLK_CLK3	IO PLL	50	50.000000	0.100000 : 250.000000
+ System Debug Clocks				
+ Timers				

Re-customize IP

## ZYNQ7 Processing System (5.5)

Documentation Presets IP Location Import XPS Settings

Page Navigator << Zynq Block Design PS-PL Configuration [Summary Report](#)

PS-PL Configuration

Peripheral I/O Pins MIO Configuration Clock Configuration DDR Configuration SMC Timing Calculation Interrupts

Search: Q

Name	Select	Description
General		
UART0 Baud Rate	115200	Baud rate is generated with internally fixed UART Ref Clod
UART1 Baud Rate	115200	Baud rate is generated with internally fixed UART Ref Clod
PL AXI idle Port	<input type="checkbox"/>	Enables idle AXI signal to the PS used to indicate that there
DDR ARB bypass Port	<input type="checkbox"/>	Enables DDR urgent/arb signal used to signal a critical mem
PS-PL Debug interface	<input type="checkbox"/>	Enables PL debug signals to PS and vice-versa
FTM Trace data interface	<input type="checkbox"/>	Enables FTM Trace AXI stream interface used to capture d
FTM Trace buffer	0	Generates a FIFO to hold trace data
FTM Data edge detector	0	Stores trace data in the FIFO when the data changes as m
FTM Trace buffer FIFO size	128	FTM Trace buffer FIFO size
FTM Trace buffer clock delay	12	Number of clock cycles interval for a trace data output fro
Include ACP transaction checker	<input type="checkbox"/>	Enables ACP transaction checker.
Trace data/control signal pipeline width	8	Enables configurable number of pipeline stages on the TRA
Power-on reset(POR) 4k timer	<input type="checkbox"/>	Enables power-on reset(POR) 4k timer. By default, 64k tim
Processor event interface	<input type="checkbox"/>	Enables event bus which provides a low-latency and direct
+ Address Editor		
+ Enable Clock Triggers		
+ Enable Clock Resets		
FCLK_RESET0_N	<input checked="" type="checkbox"/>	Enables general purpose reset signal 0 for PL logic
FCLK_RESET1_N	<input type="checkbox"/>	Enables general purpose reset signal 1 for PL logic
FCLK_RESET2_N	<input type="checkbox"/>	Enables general purpose reset signal 2 for PL logic
FCLK_RESET3_N	<input type="checkbox"/>	Enables general purpose reset signal 3 for PL logic

OK Cancel

# SDK Compilers

# GNU Tools: GCC

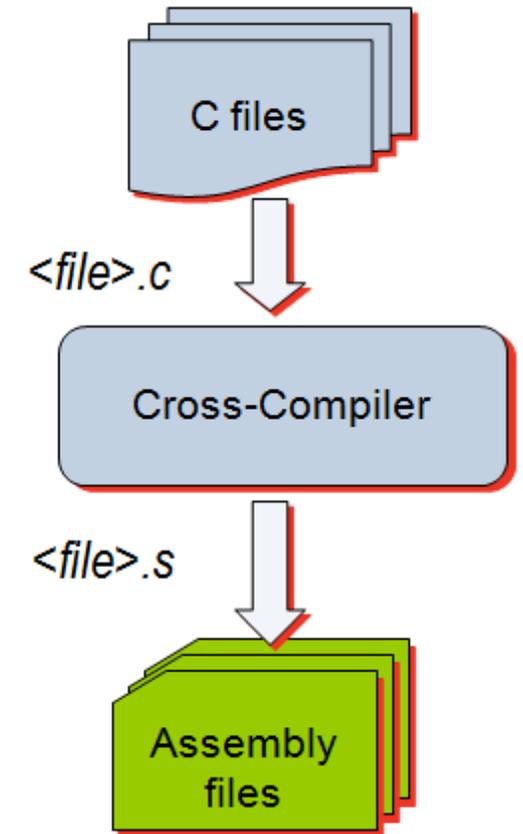
GCC translates C source code into assembly language

GCC also functions as the user interface, passing options to GNU assembler and to the GNU linker, calling the assembler and the linker with the appropriate parameters

Supported cross-compilers

ARM processor compiler

- GNU GCC (arm-xilinx-eabi-gcc)
- GNU Linux GCC (arm-xilinx-linux-eabi-gcc)



# GNU Tools: AS

Input: assembly language files

- File extension: .s

Output: object code

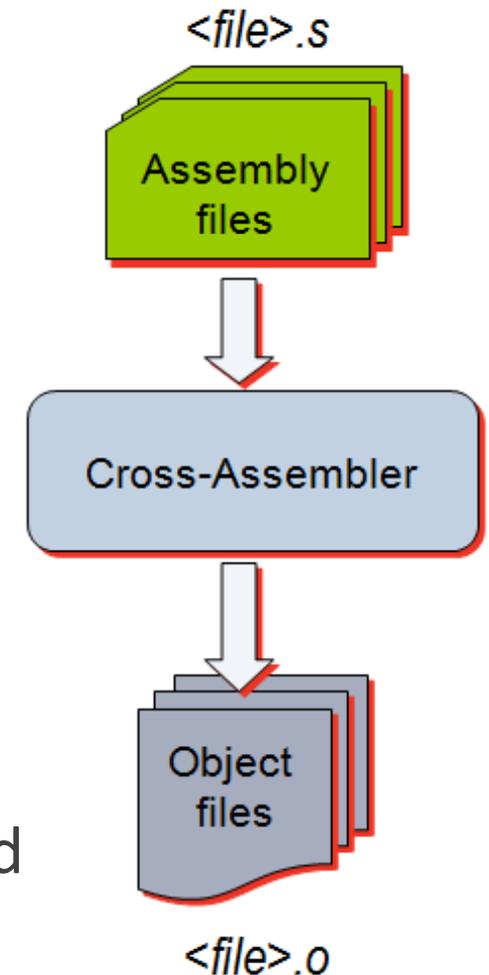
- File extension: .o

Contains

- Assembled piece of code
- Constant data
- External references
- Debugging information

Typically, the compiler automatically calls the assembler

Use the `-Wa` switch if the source files are assembly only and use `gcc`



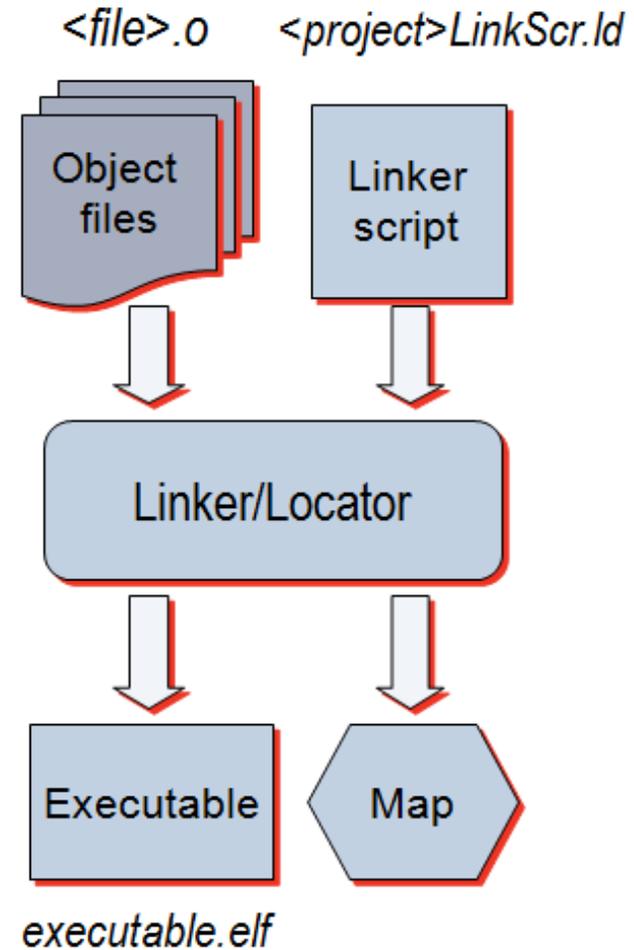
# GNU Tools: Linker (LD)

## Inputs

- Several object files
- Archived object files (library)
- Linker script (*\*.ld*)

## Outputs

- Executable image (ELF)
- Map file

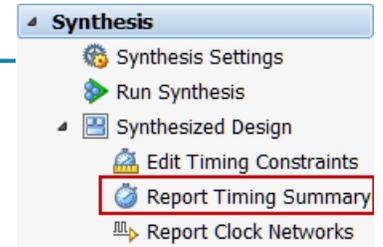


# Timing Reports

# Report Timing Summary

Tcl command: `report_timing_summary`

`report_timing_summary -delay_type max -report_unconstrained -check_timing_verbose -max_paths 10 -input_pins -name timing_1`



Vivado IDE

Options tab

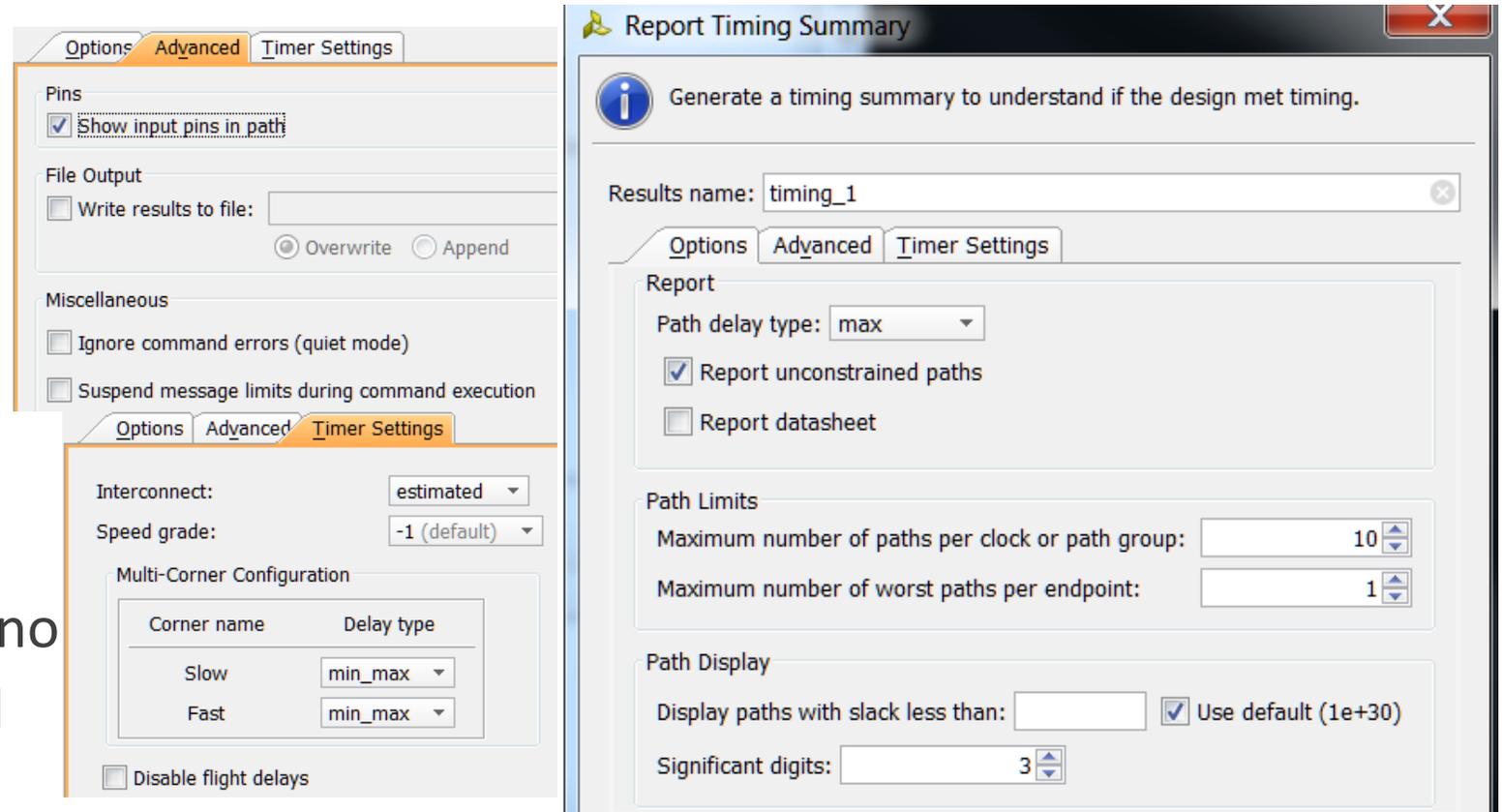
- Maximum number of paths

Advanced tab

- Write to a file

Timer Settings

- Interconnect delay can be ignored
- Flight delays can be disabled



# Report Timing Summary

## Design Timing Summary

- WNS, TNS, total number of endpoints are of interest

## Clock Summary

- Primary and derived clocks

## Check Timing

- Number of unconstrained internal endpoints

Timer Settings
<b>Design Timing Summary</b>
Clock Summary (3)
Check Timing (45)
Intra-Clock Paths
Inter-Clock Paths
Path Groups
User Ignored Paths
Timer Settings
Design Timing Summary
<b>Clock Summary (3)</b>
Check Timing (45)
Intra-Clock Paths
Inter-Clock Paths
Path Groups
User Ignored Paths
Timer Settings
Design Timing Summary
Clock Summary (3)
<b>Check Timing (45)</b>
Intra-Clock Paths
Inter-Clock Paths
Path Groups
User Ignored Paths

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): <a href="#">1.826 ns</a>	Worst Hold Slack (WHS): NA	Worst Pulse Width Slack (WPWS): <a href="#">3.000 ns</a>
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): NA	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: NA	Number of Failing Endpoints: 0
Total Number of Endpoints: 102	Total Number of Endpoints: NA	Total Number of Endpoints: 45

Name	Waveform	Period (ns)	Frequency (MHz)
clk_in	{0.000 5.000}	10.000	100.000
clk_out1_clk_5MHz	{0.000 100.000}	200.000	5.000
clkfbout_clk_5MHz	{0.000 25.000}	50.000	20.000

Timing Check	Count
unconstrained_internal_endpoints	26
no_clock	10
no_output_delay	9
no_input_delay	0
multiple_clock	0
generated_clocks	0
loops	0
partial_input_delay	0
partial_output_delay	0