

Diseño avanzado de sistemas embebidos  
en lógica programable:  
**Zynq APSoC, Vivado-HLS y SDSoC.**

Curso de Posgrado

19 al 23 de Marzo de 2018  
San Luis - Argentina



# *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

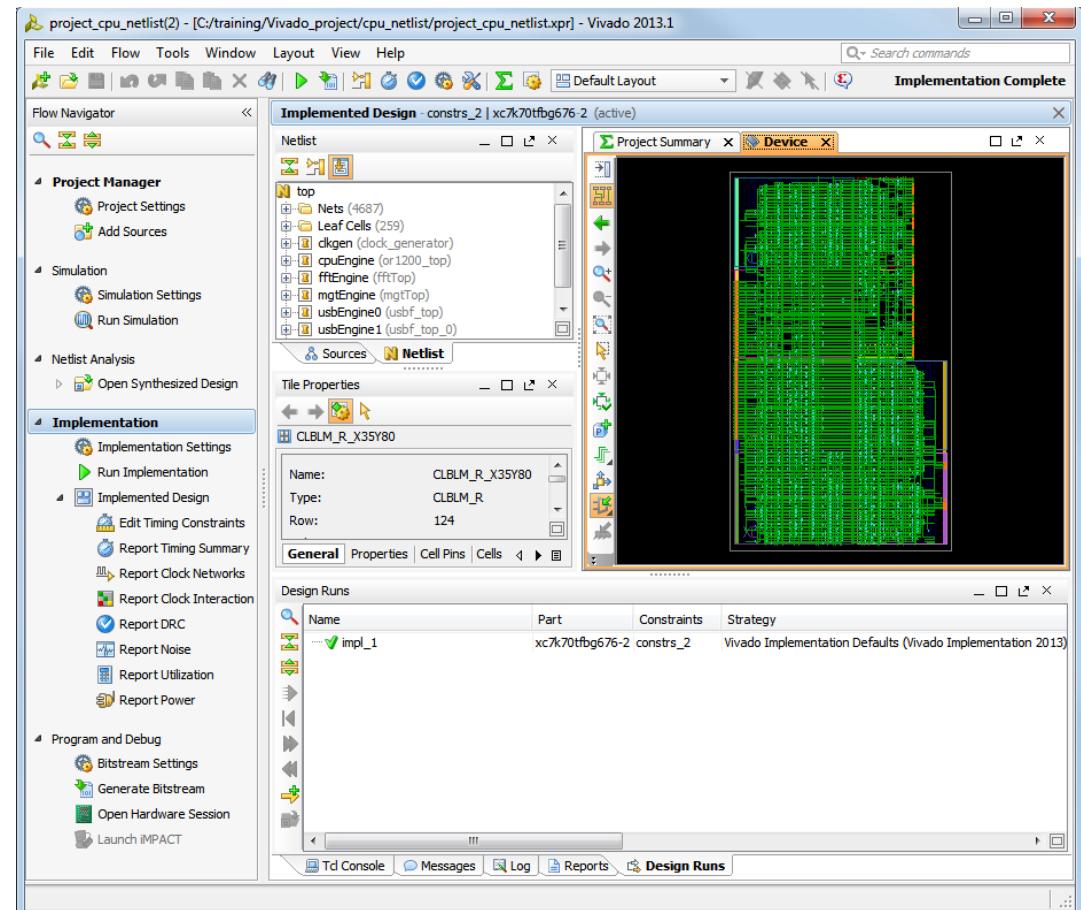
- Elaboration of HDL
- Hierarchical exploration
- Schematic generation

## XSIM simulator integration

- Synthesis, implementation and simulation in one package

## I/O pin planning

- Interactive rule-based I/O assignment

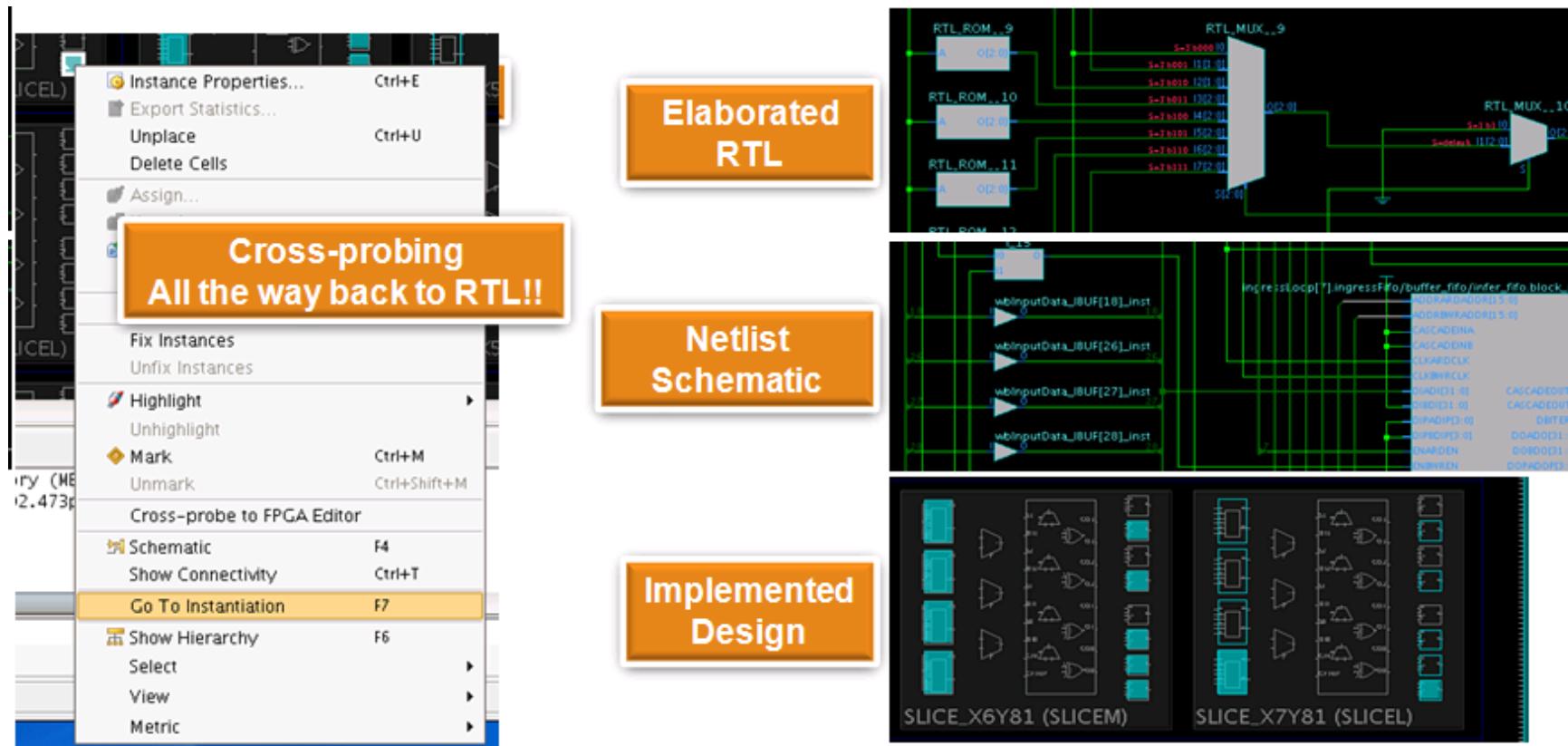


*Hierarchical Design Analysis and  
Implementation Environment*

# Vivado Visualization Features

Visualize and debug a design at any flow stage

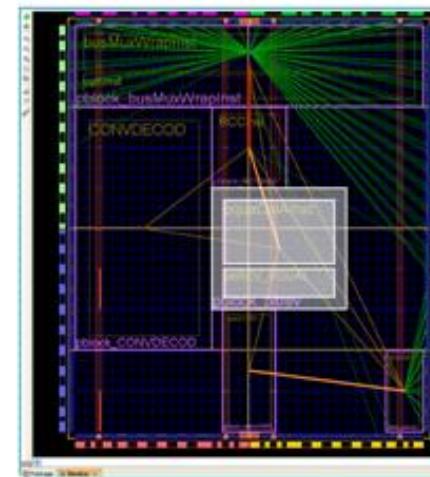
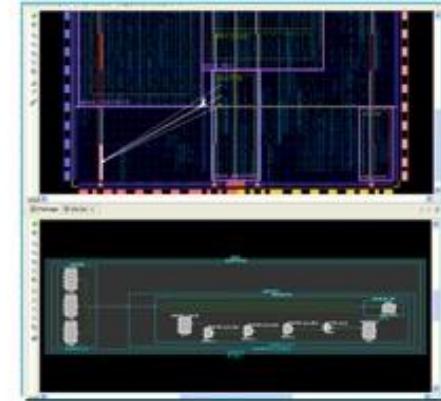
- Cross-probing between netlist/schematic/RTL



# 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

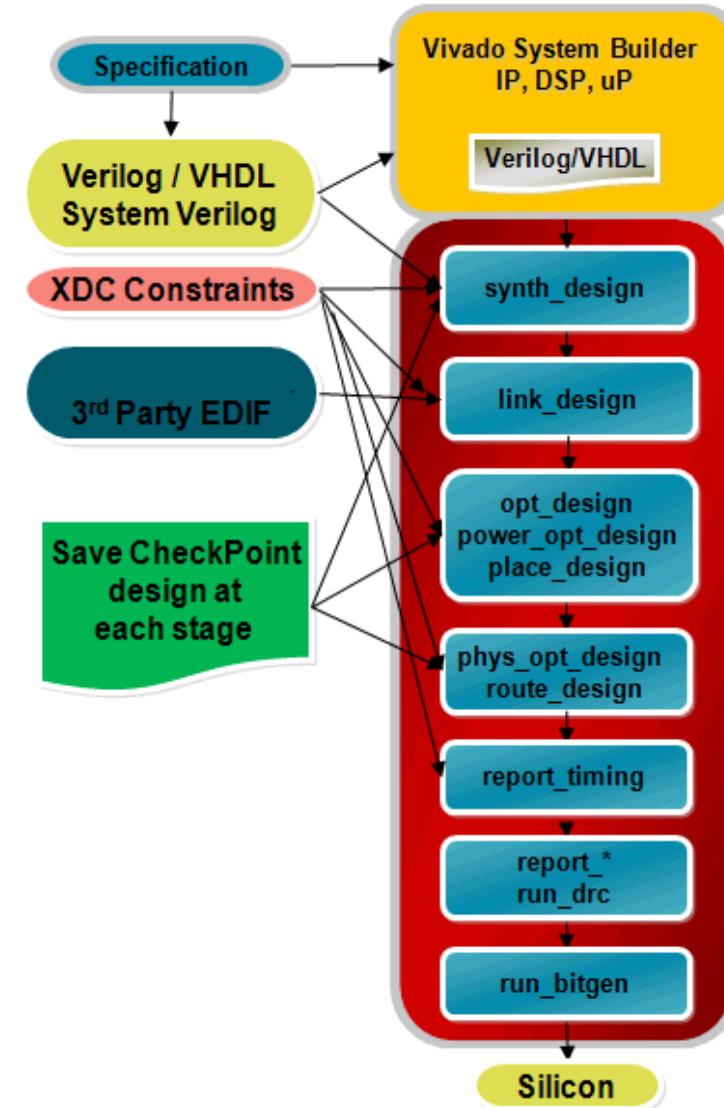
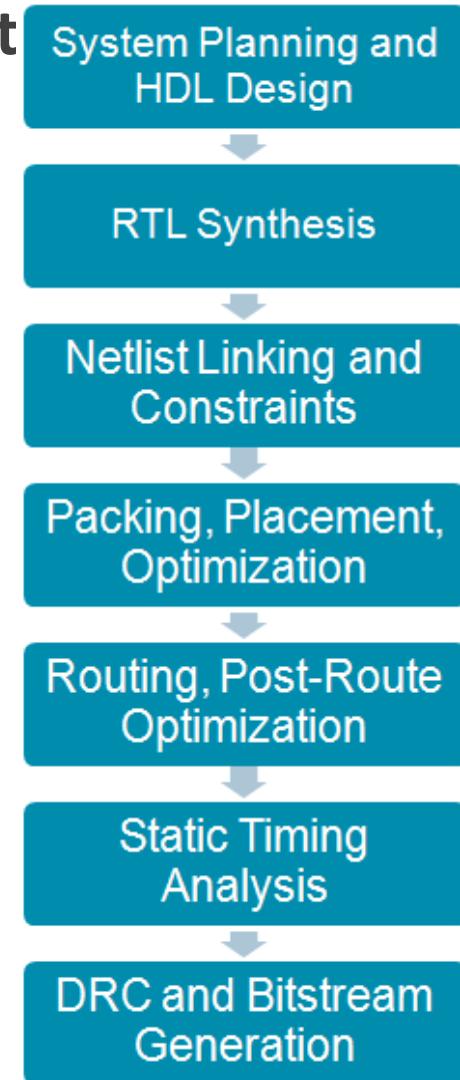
# *Vivado Design Suite*

## *Introduction*

---

# 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



# 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

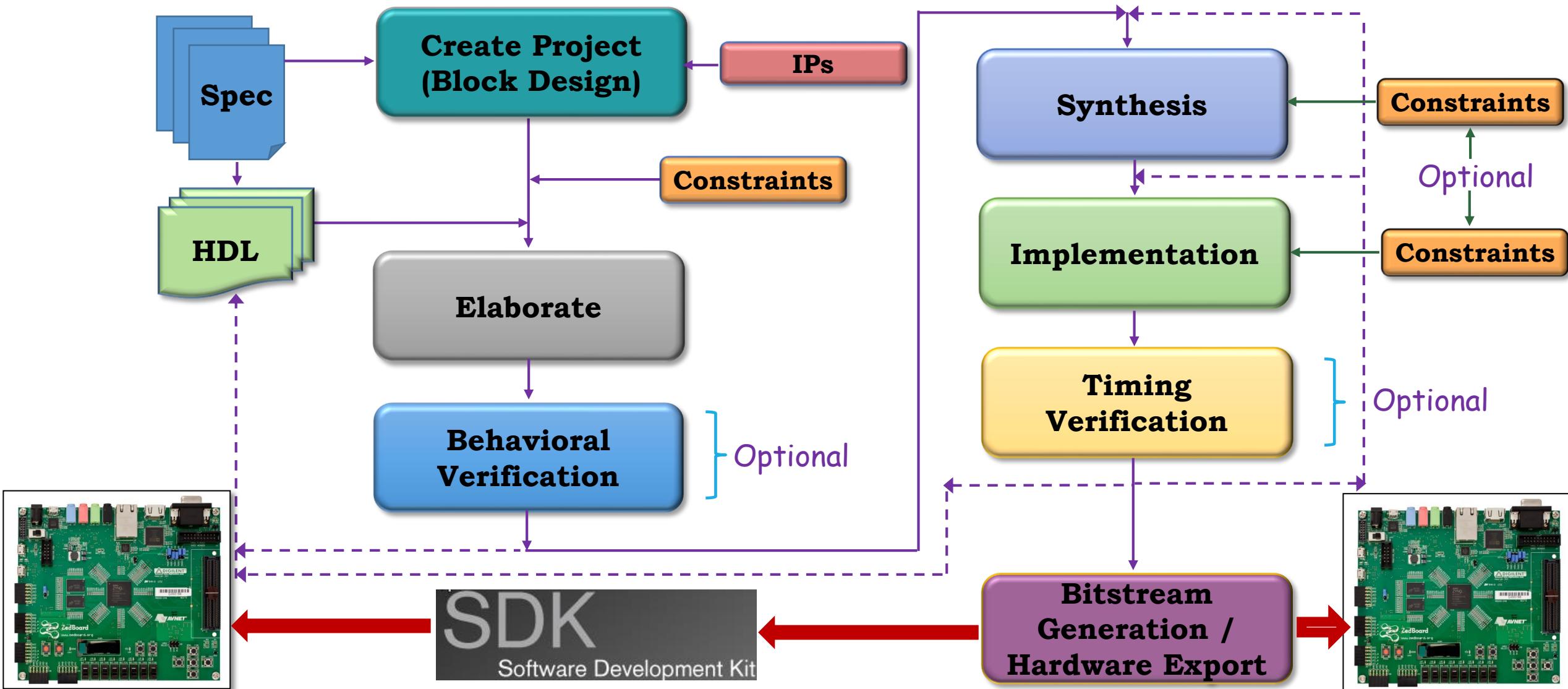
## Log file (*vivado.log*)

- 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

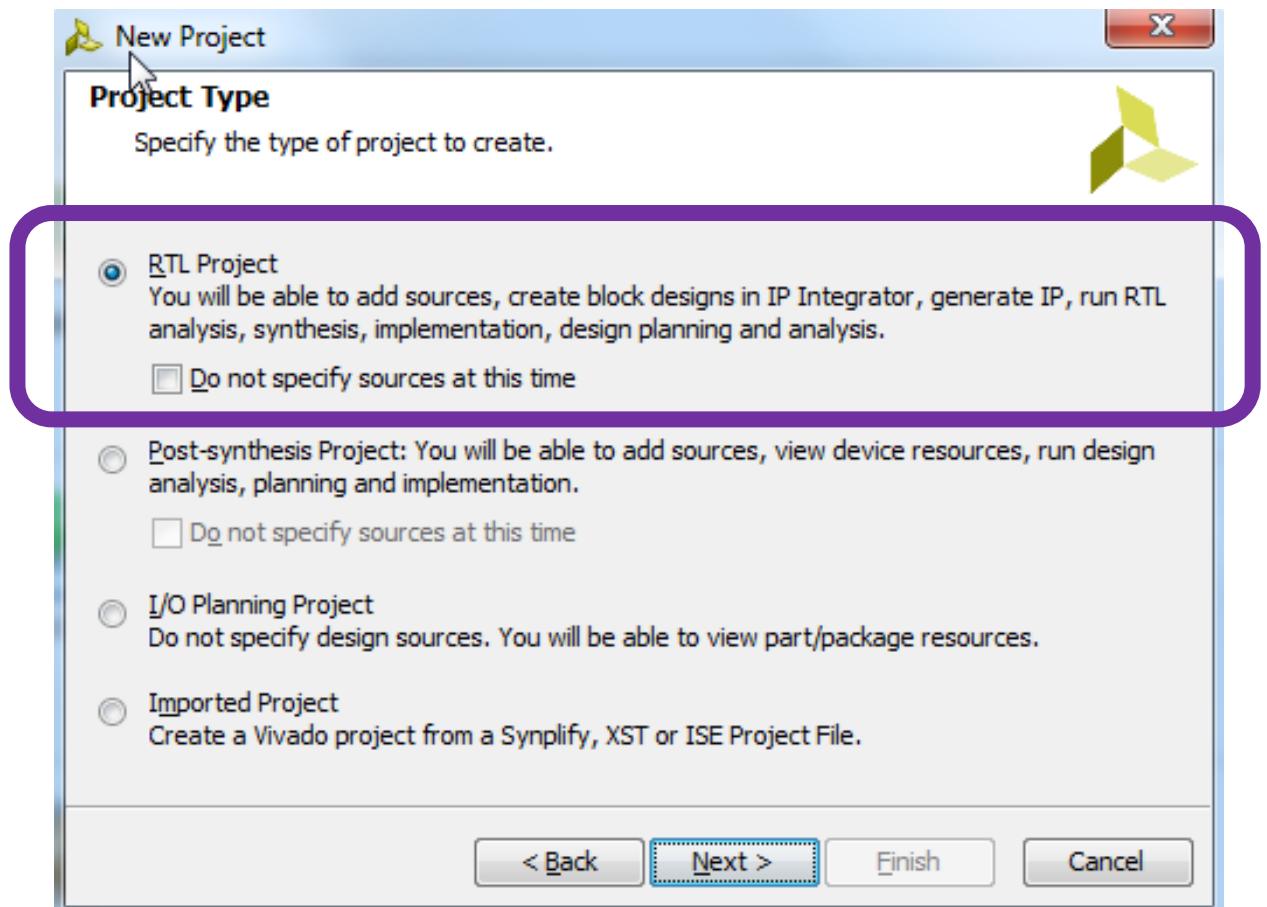
# Embedded System Design – Vivado Flow



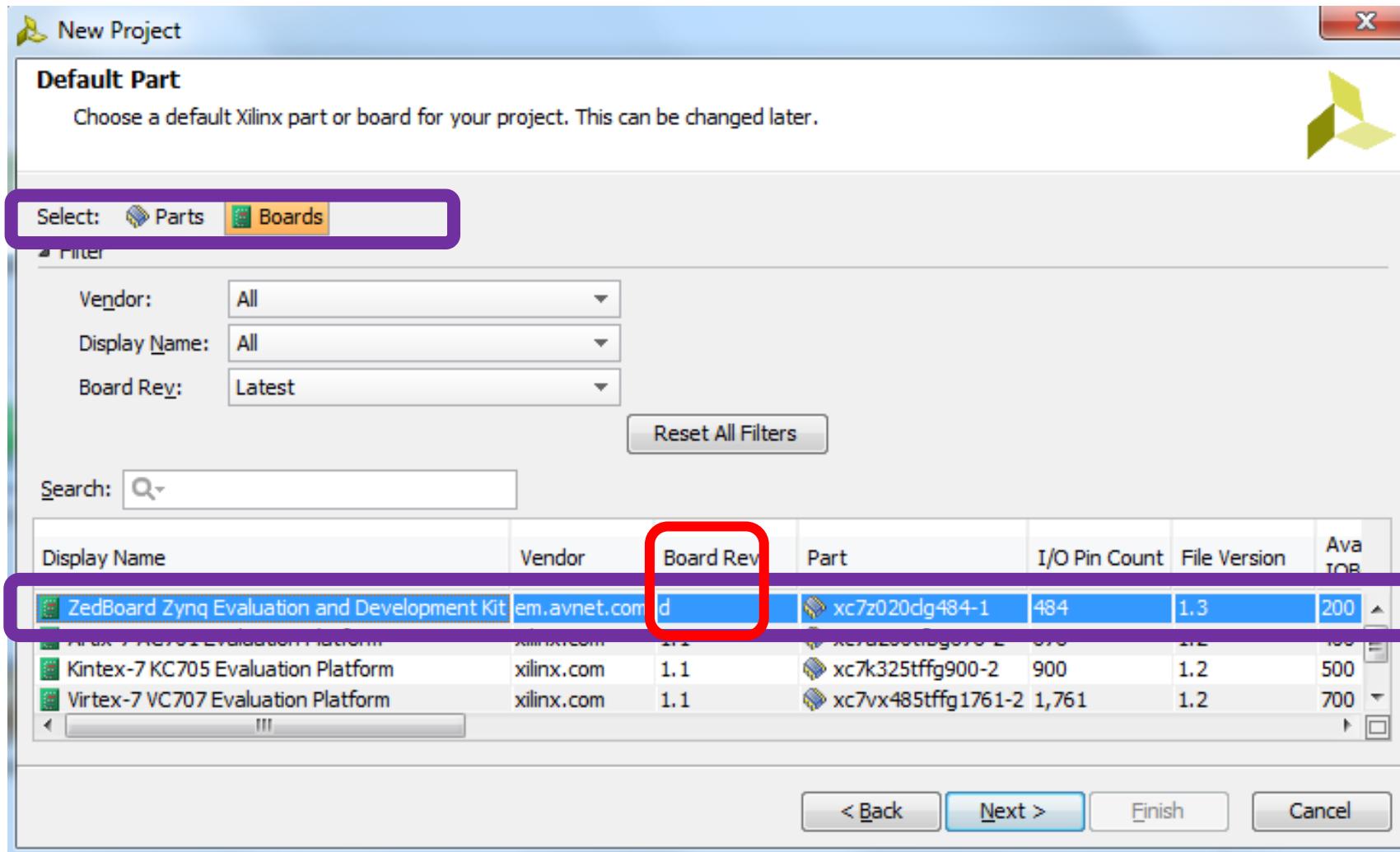
# *Vivado Flow Practical Steps*

---

# Creating a Project

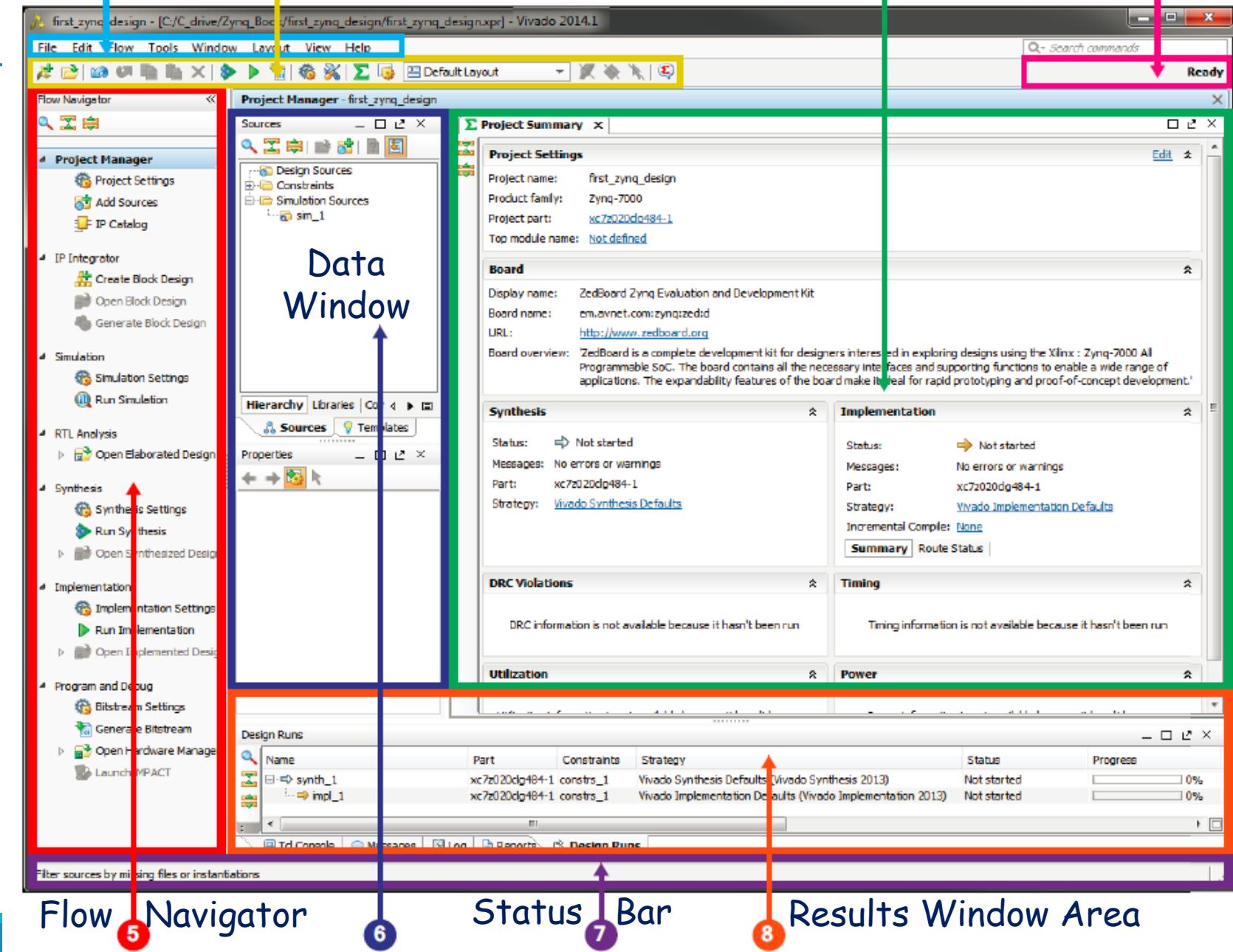


# Creating a Project



## Menu Bar 1 Main Toolbar 2

## Workspace 3 Status Bar 4

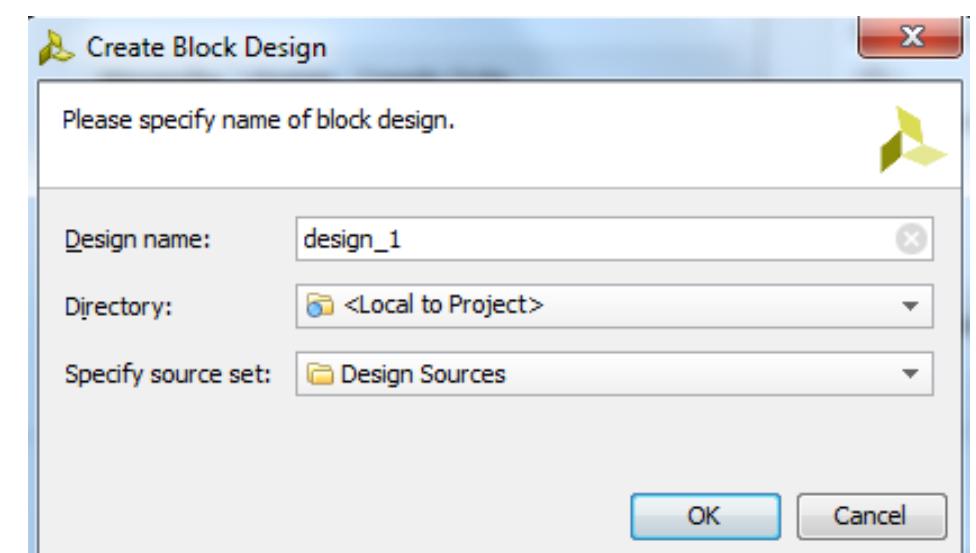
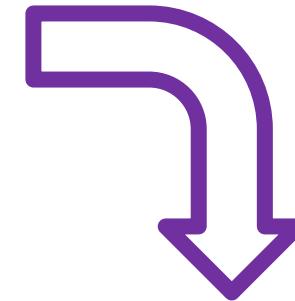
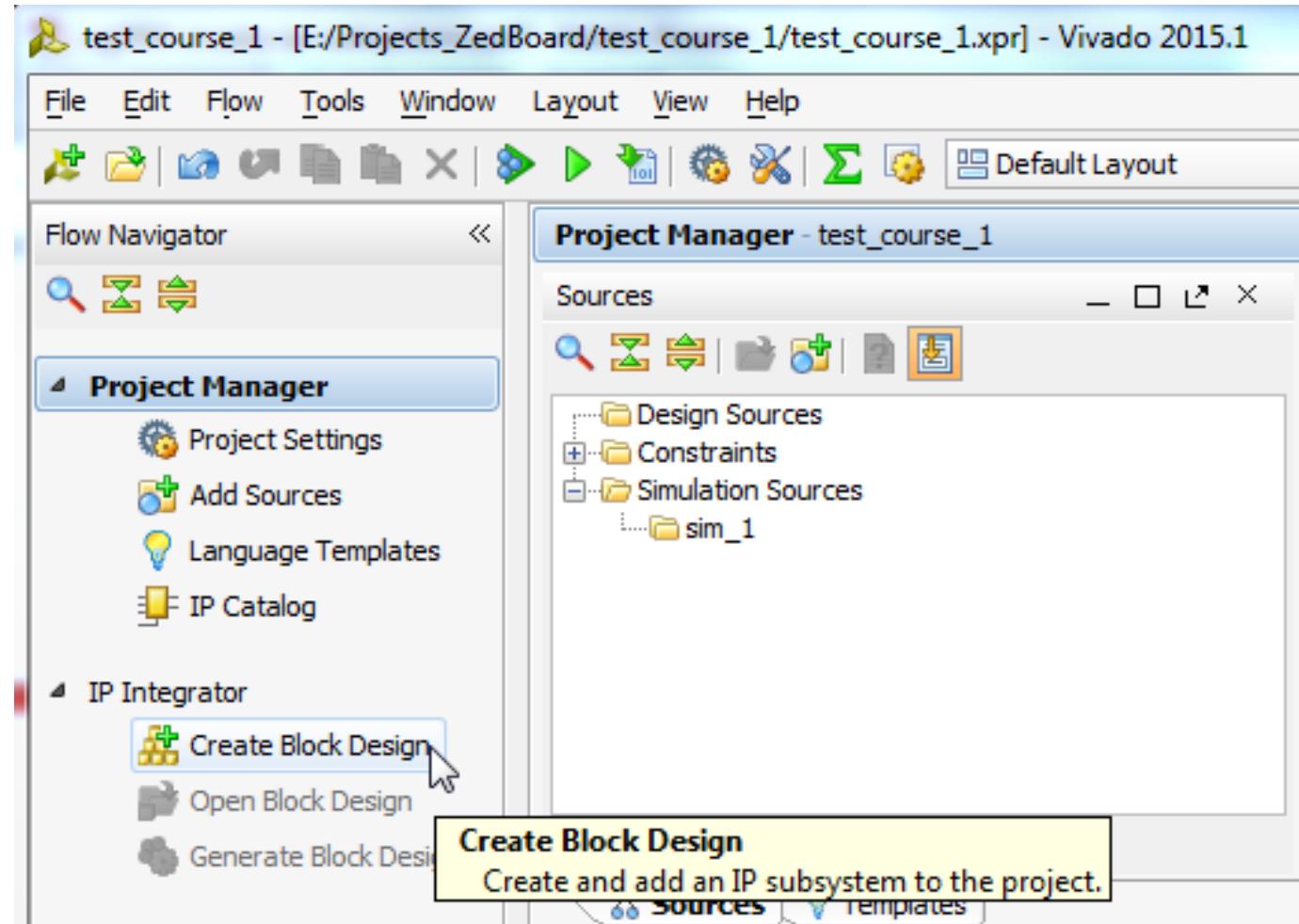


# Main Components of the Project Navigator

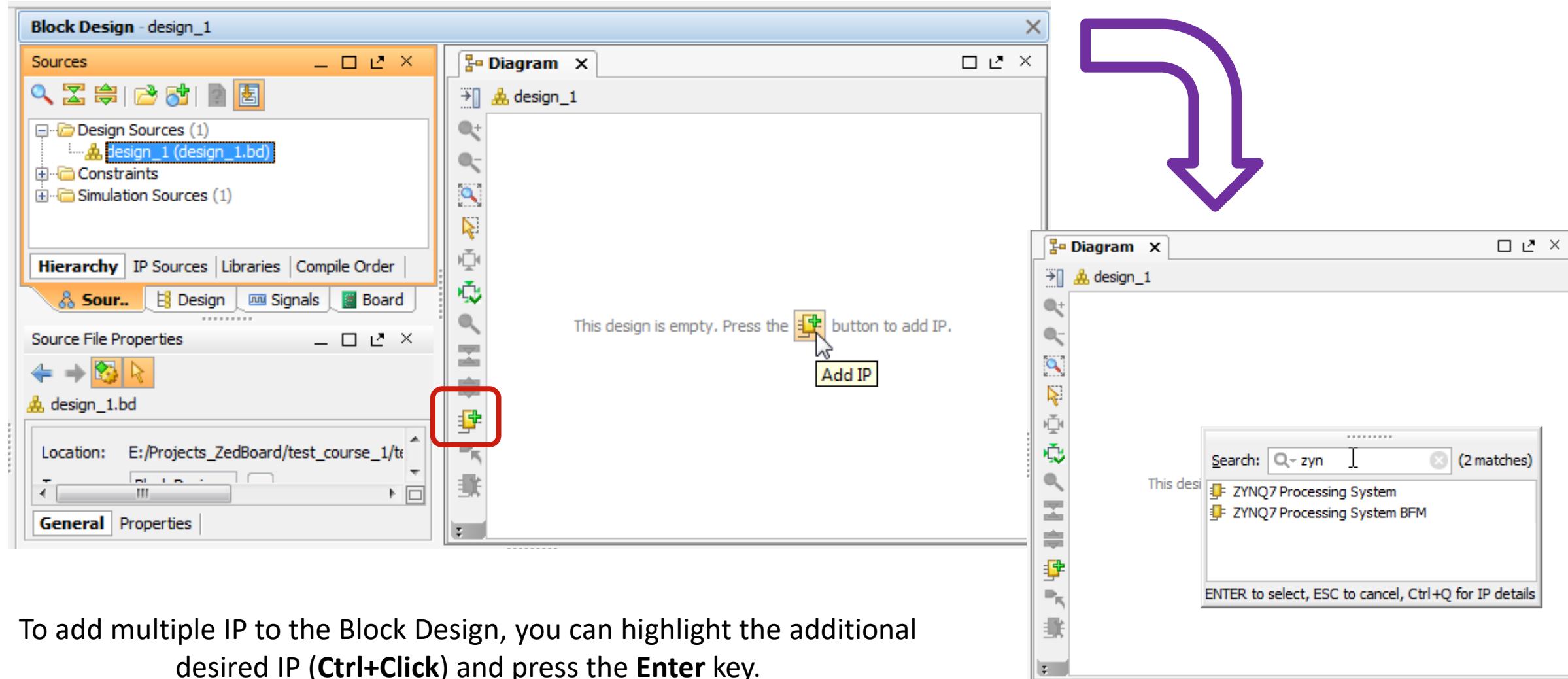
---

- 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. Data Window Pane:** by default displays information that relates to design data and sources, such as Property Window, Netlist Window, and Source Window
- 7. 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

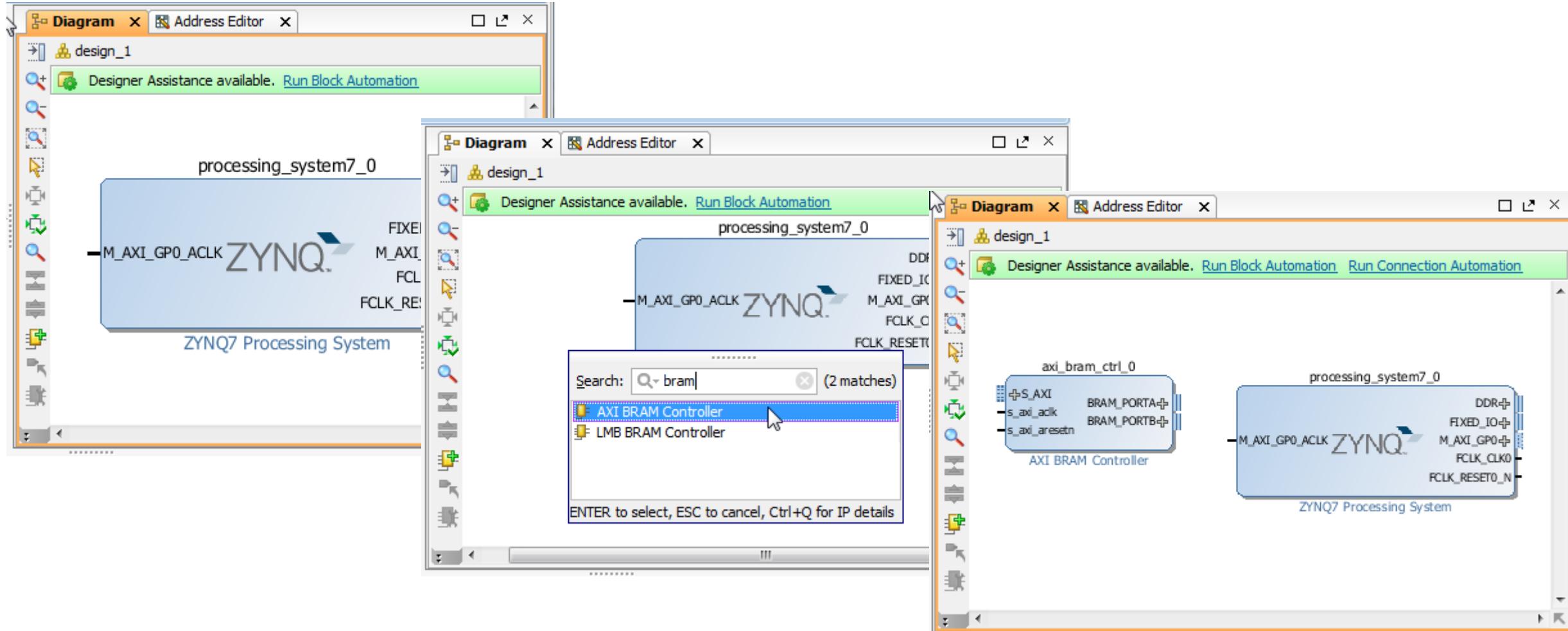


# Adding IP Modules to the Design Canvas

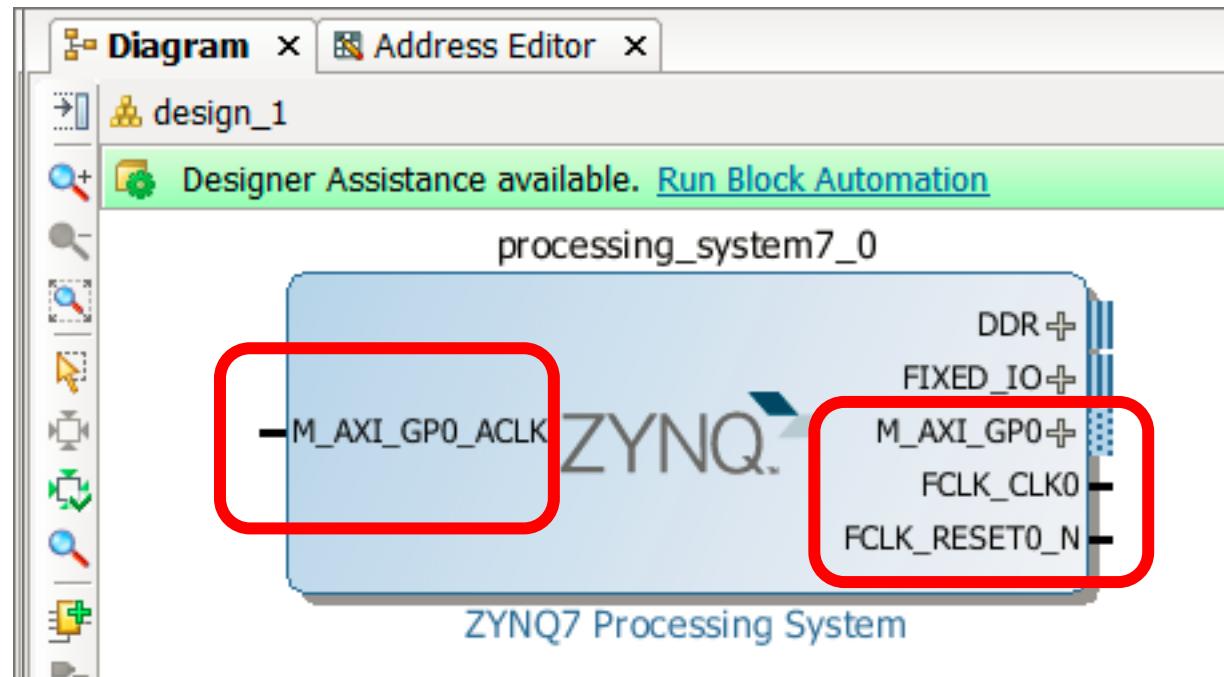


To add multiple IP to the Block Design, you can highlight the additional desired IP (**Ctrl+Click**) and press the **Enter** key.

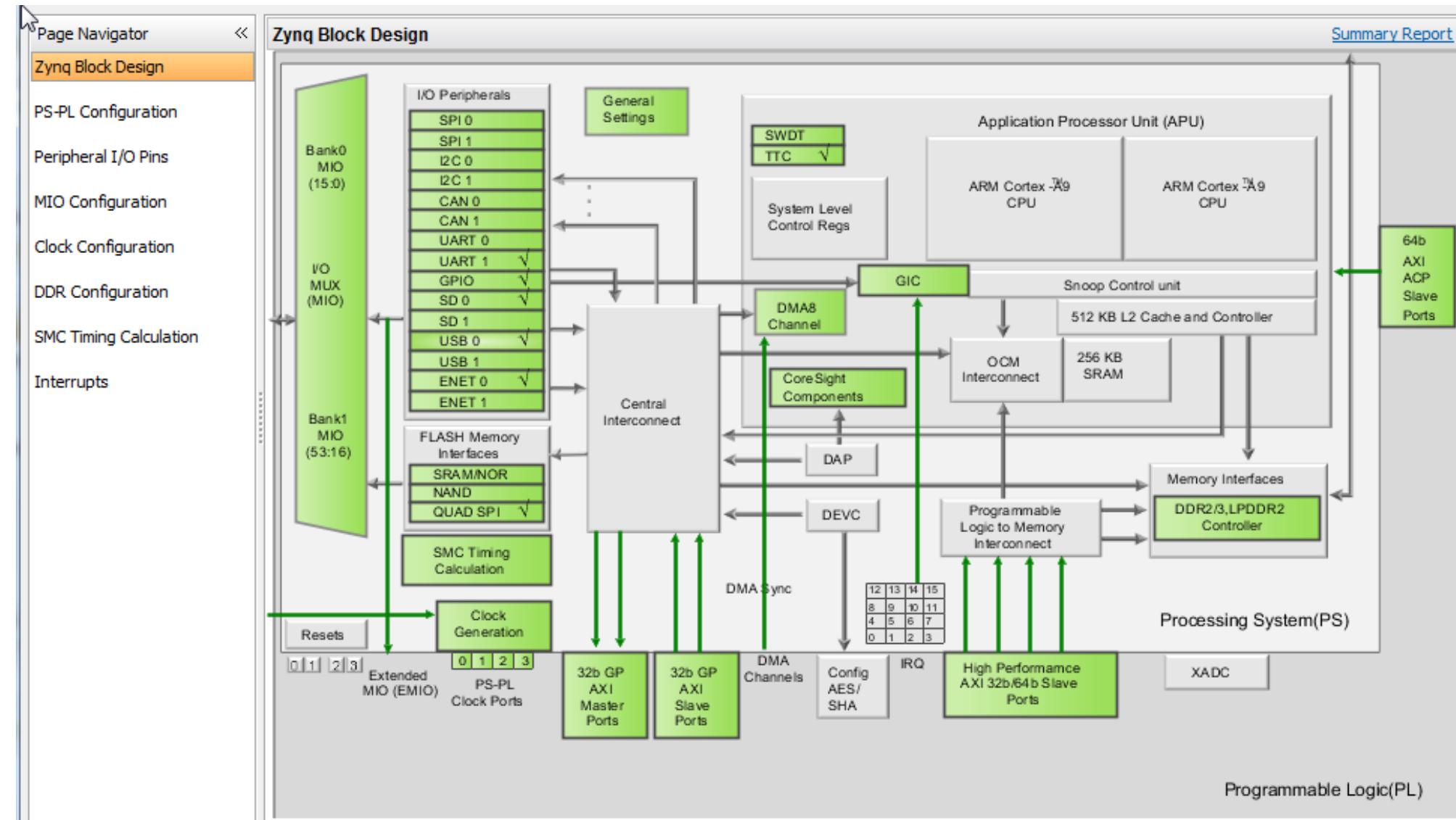
# Adding More IPs



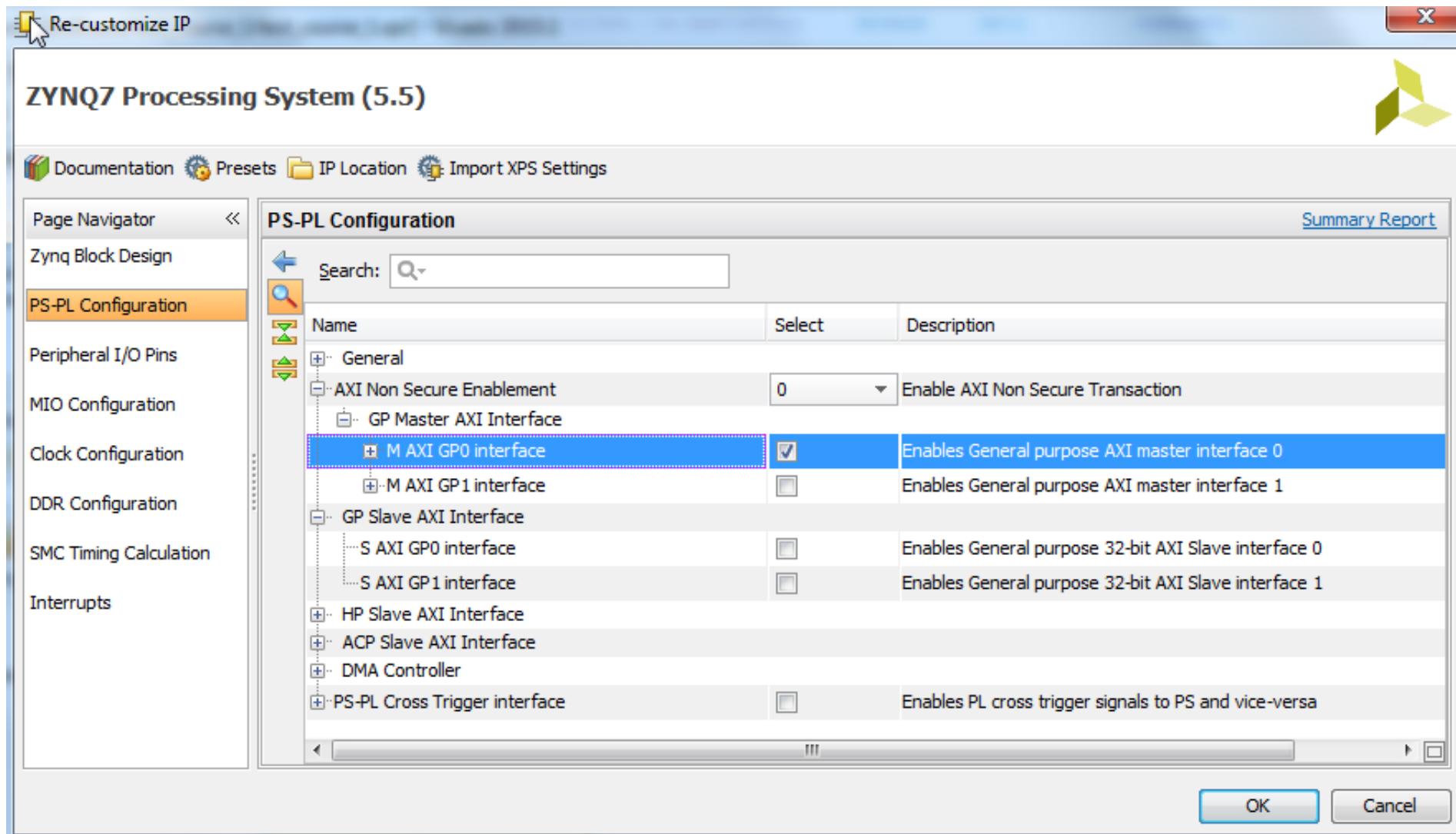
# Customizing the PS



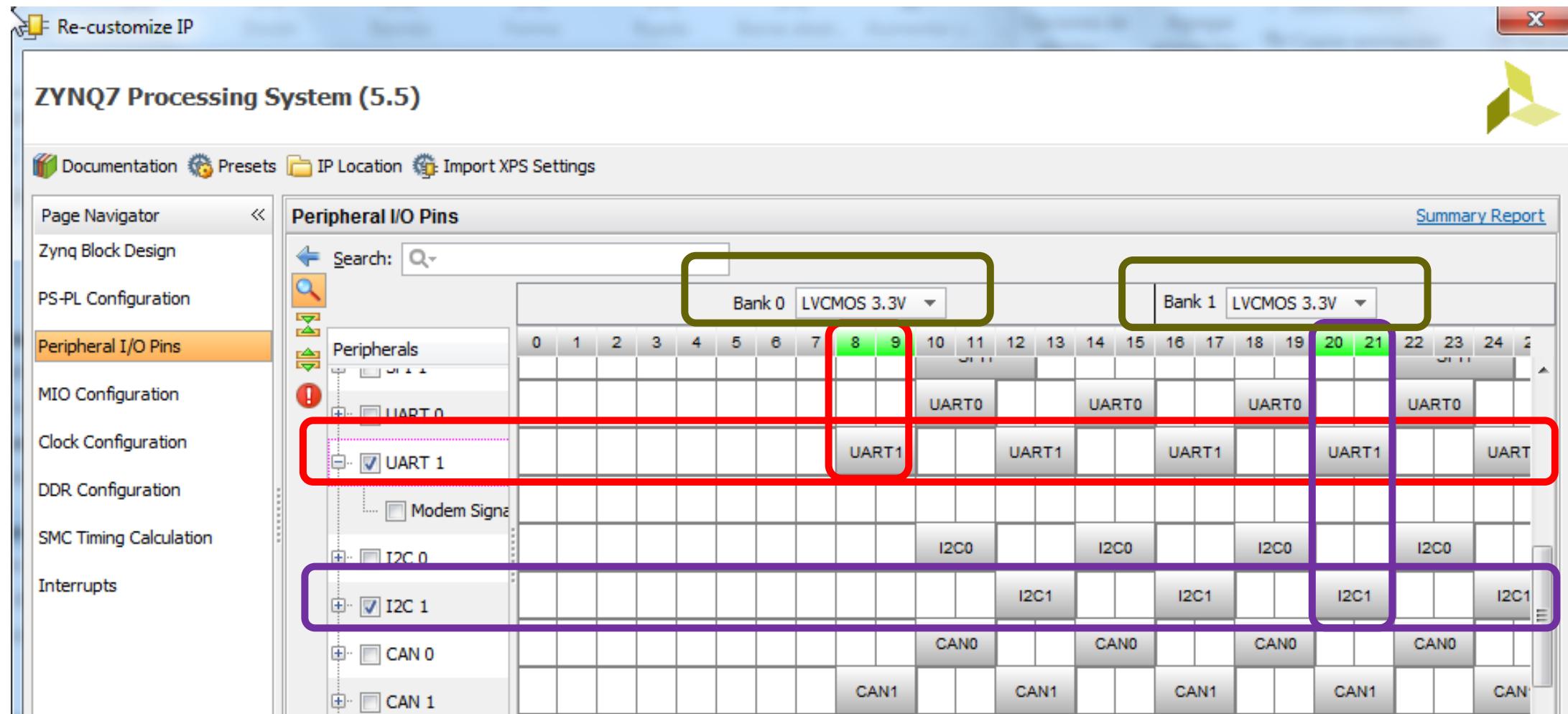
# PS Customization Options



# PS-PL Configuration Options



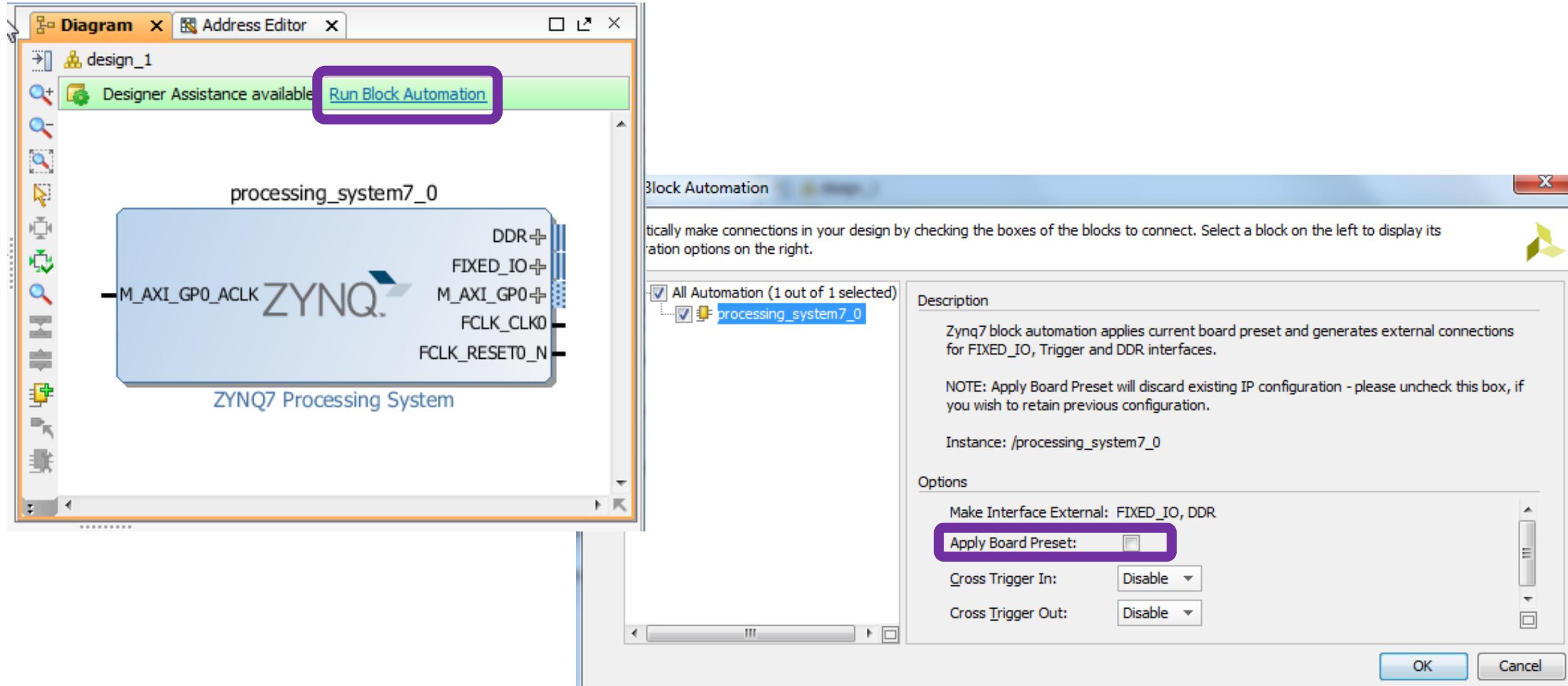
# MIO and EMIO Configuration



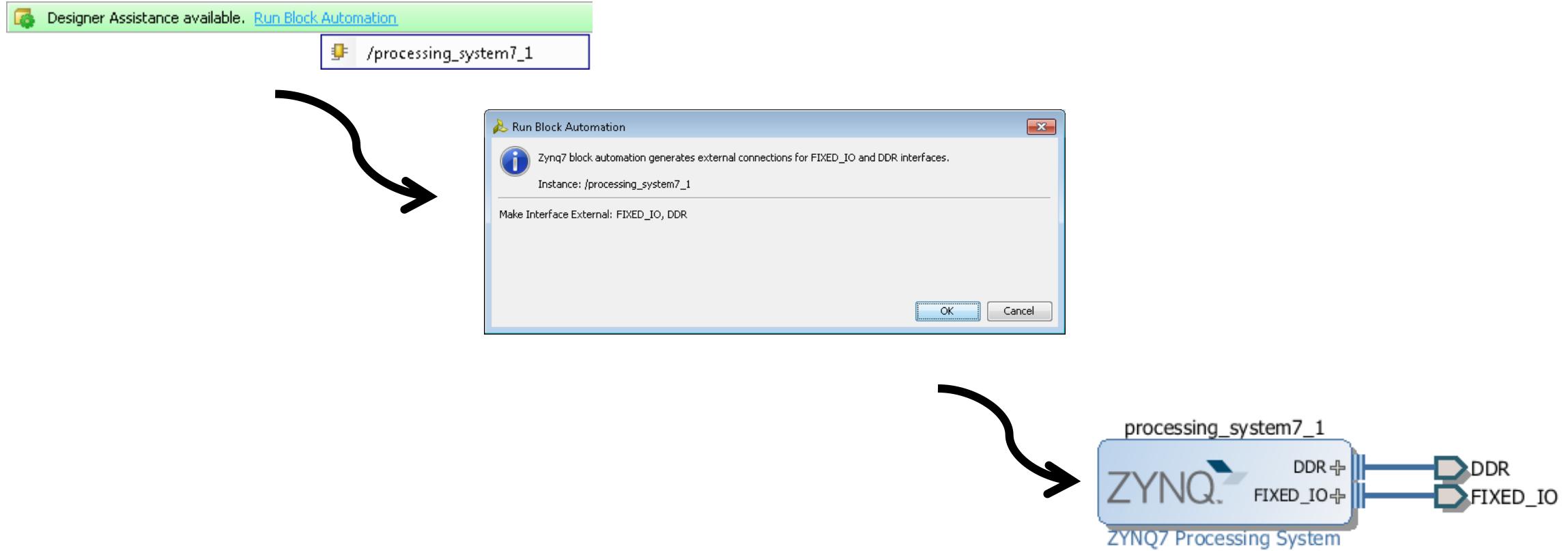
# MIO I/O Pins Configuration

The screenshot shows the Vivado MIO Configuration interface. On the left, a page navigator lists various configuration options: Zynq Block Design, PS-PL Configuration, Peripheral I/O Pins, MIO Configuration (which is selected and highlighted in orange), Clock Configuration, DDR Configuration, SMC Timing Calculation, and Interrupts. The main area is titled "MIO Configuration" and displays two dropdown menus for Bank 0 I/O Voltage (LVCMS 3.3V) and Bank 1 I/O Voltage (LVCMS 1.8V). A search bar is also present. The central part of the interface is a table with columns: Peripheral, IO, Signal, IO Type, Speed, Pullup, Direction, and Polarity. The "Peripheral" column lists various peripheral components like USB 1, SD 0, SD 1, UART 0, and so on. The "IO" column shows the specific MIO pin ranges assigned to each peripheral. In the screenshot, the row for "UART 1" is selected, and a dropdown menu is open over the "IO" column, listing "MIO 48 .. 49", "MIO 16 .. 17", "MIO 20 .. 21", "MIO 24 .. 25", "MIO 28 .. 29", "MIO 32 .. 33", "MIO 36 .. 37", "MIO 40 .. 41", "MIO 44 .. 45", and "MIO 52 .. 53".

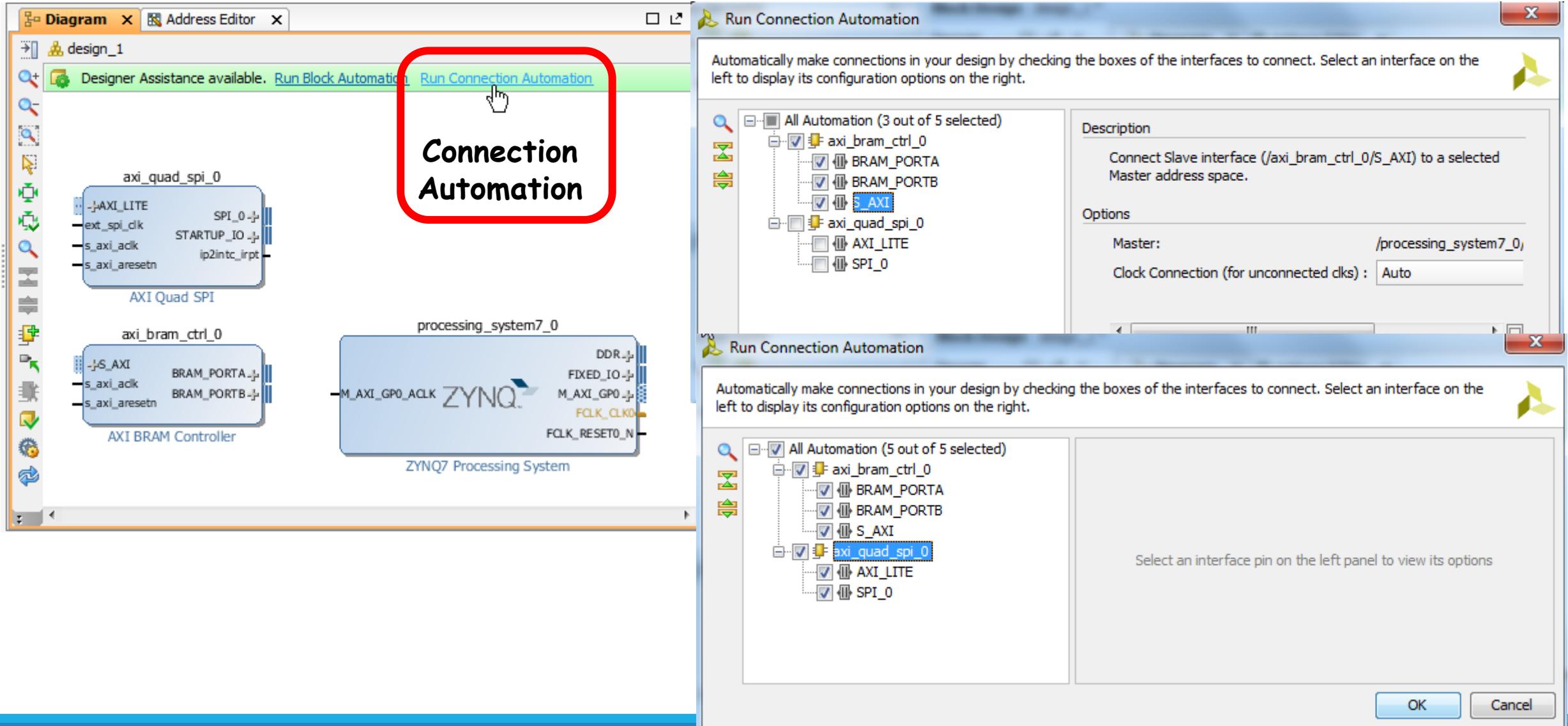
# Running Block Automation



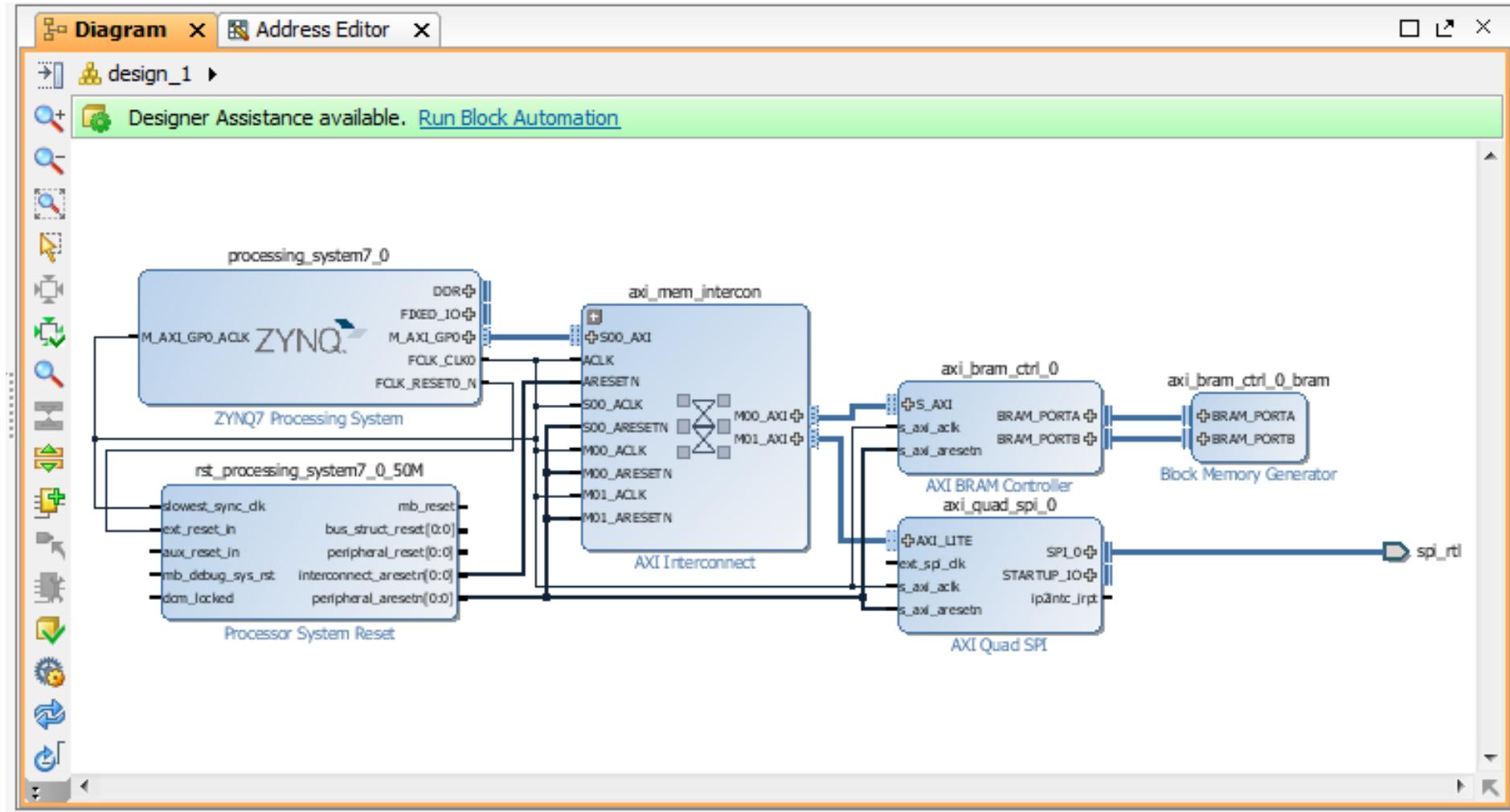
# Run Block Automation – Customized PS



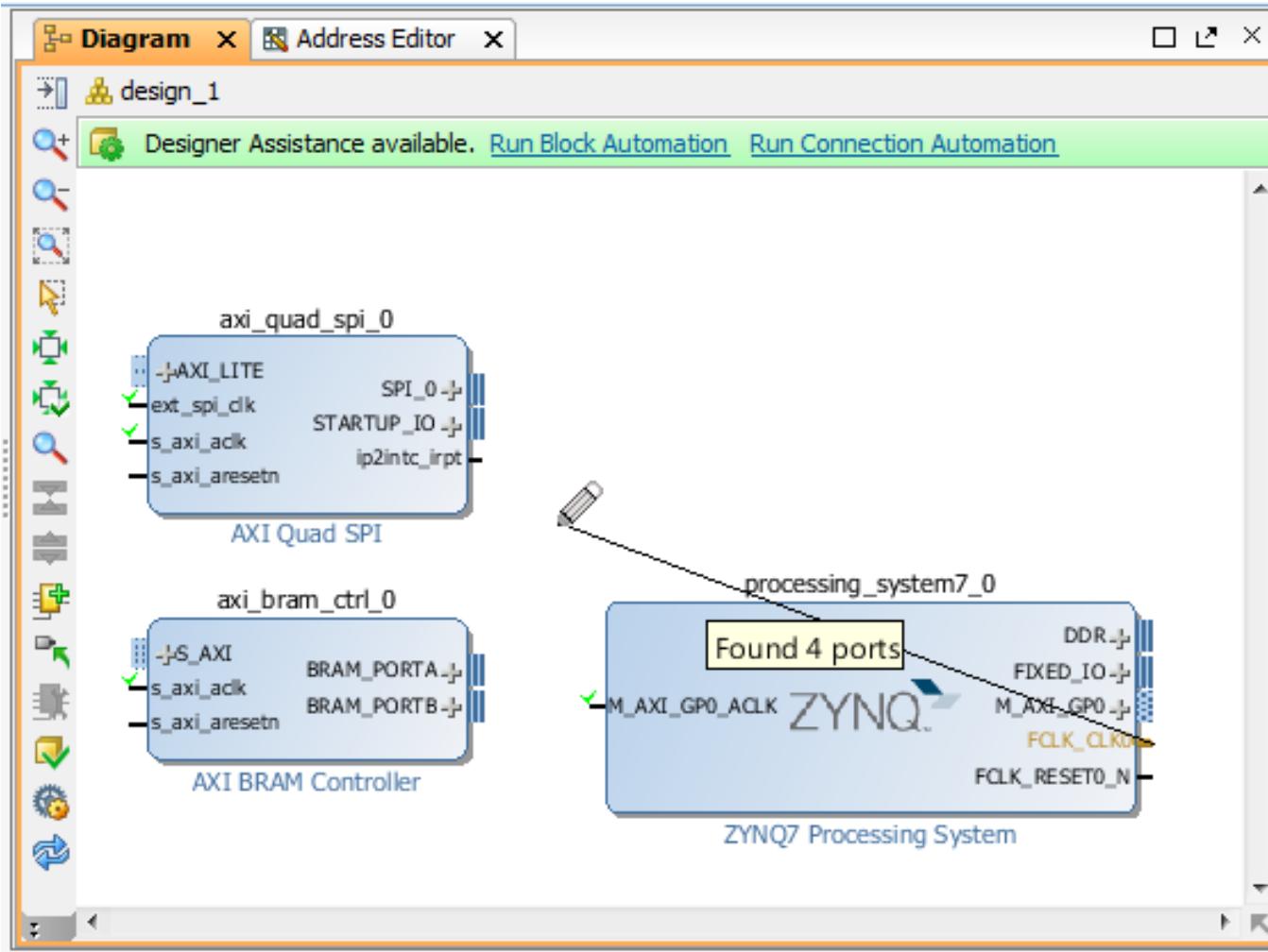
# Connecting IPs – Making Connections With the Tool



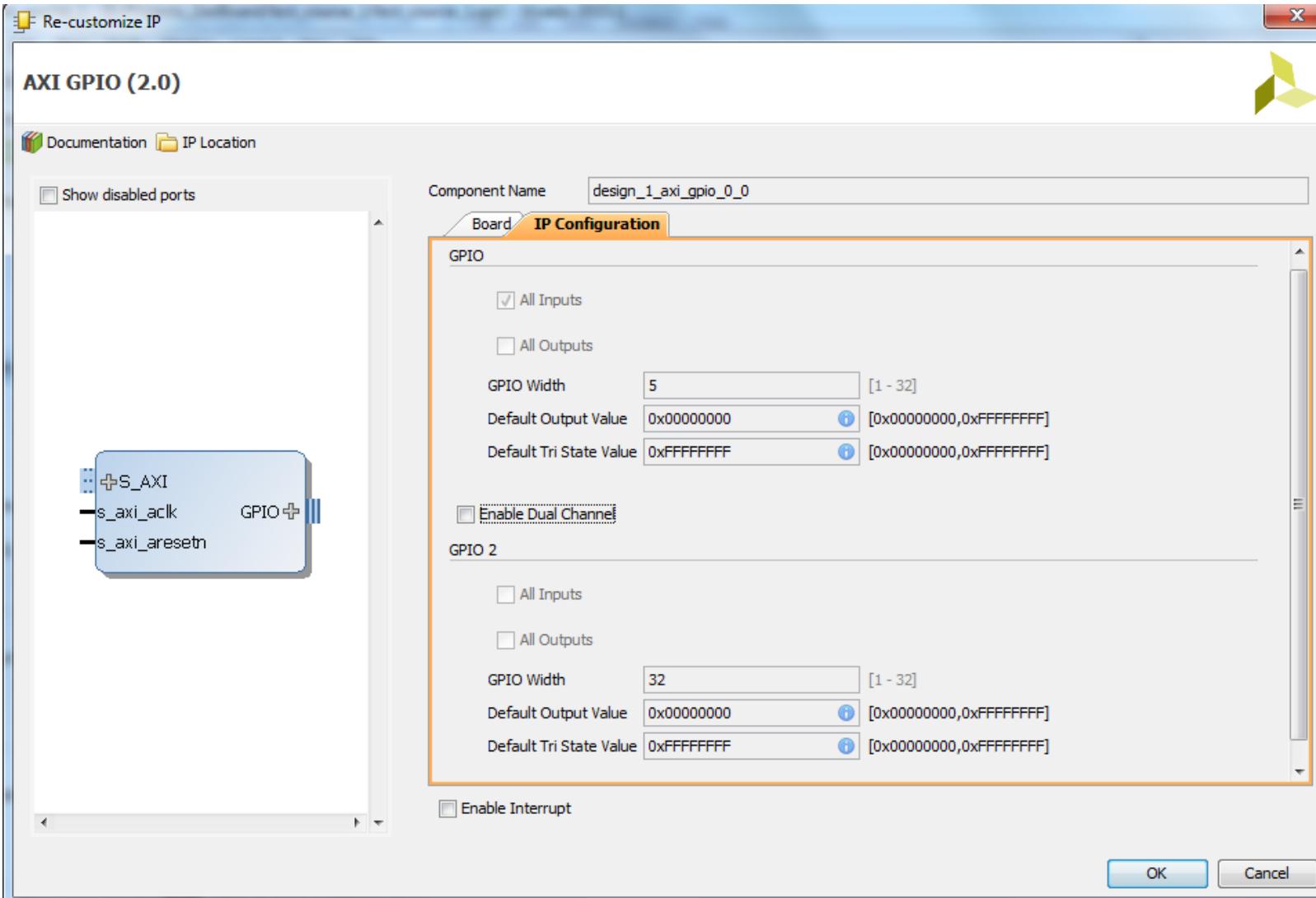
# Connecting IPs – Making Connections With the Tool



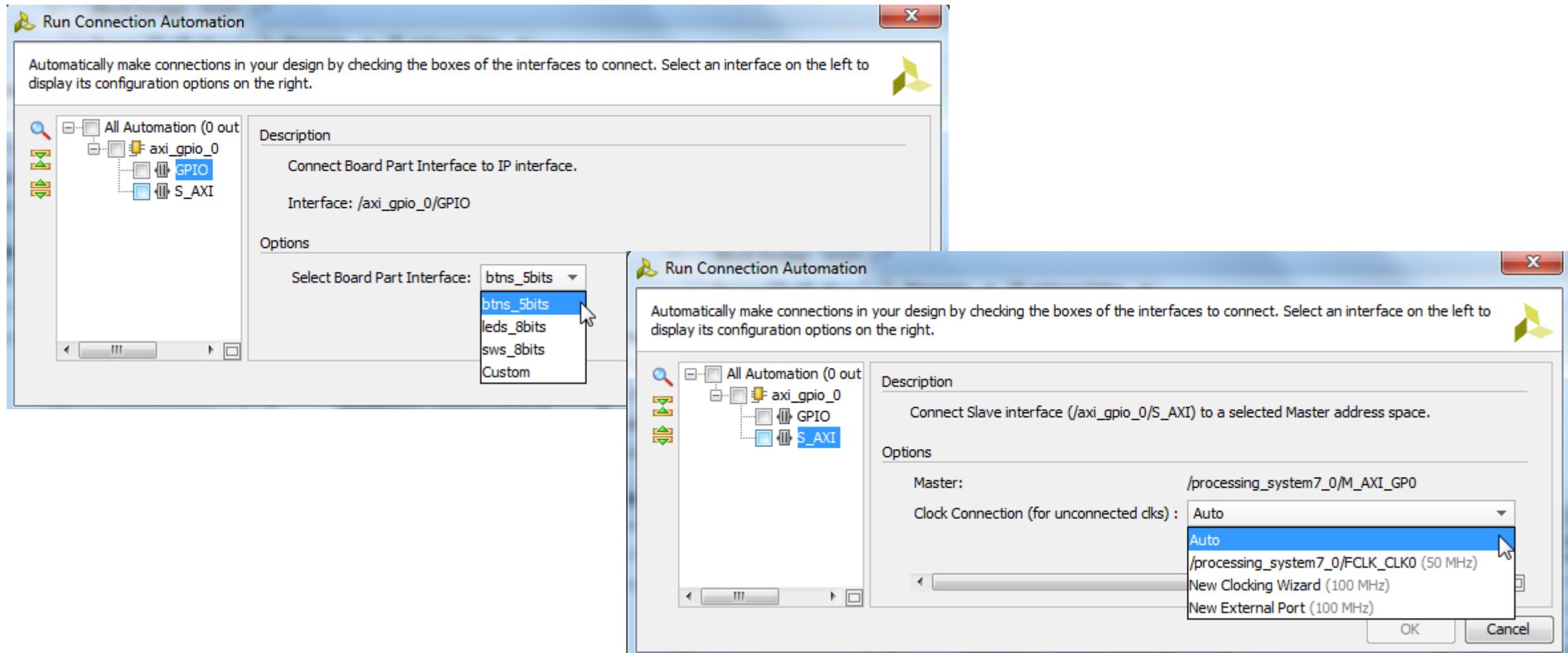
# Connecting IPs – Making Connections Manually



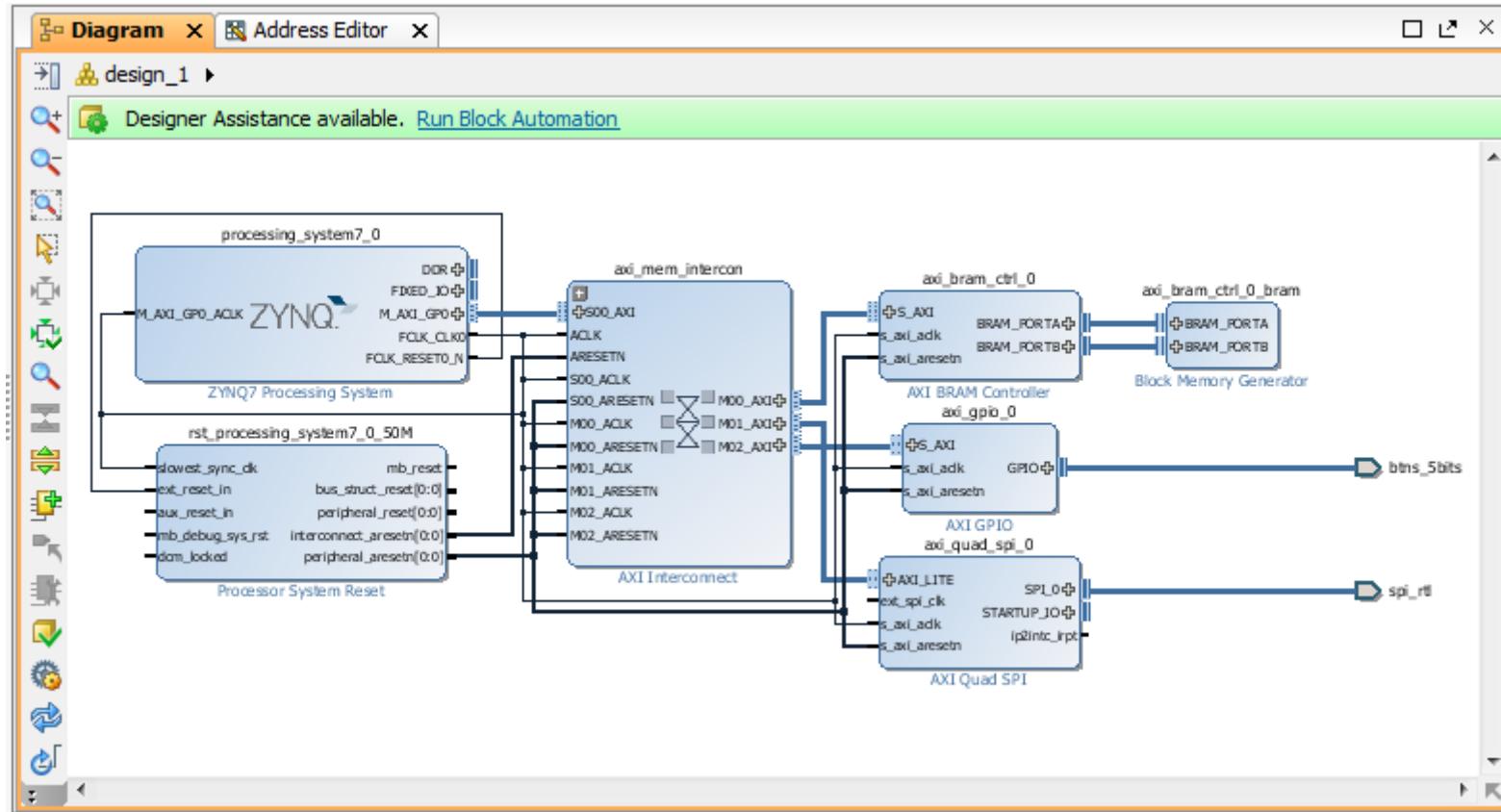
# Adding GPIO IP Block



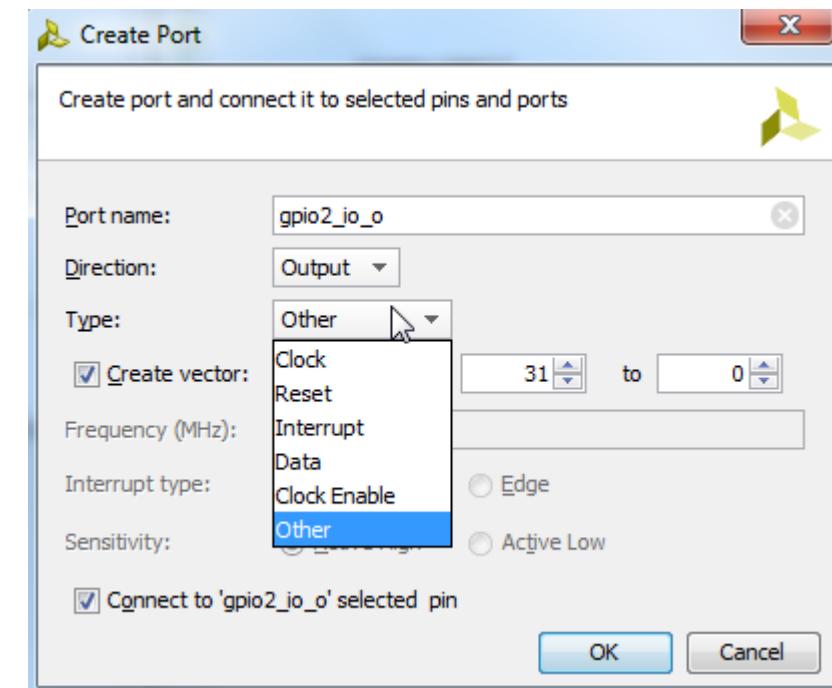
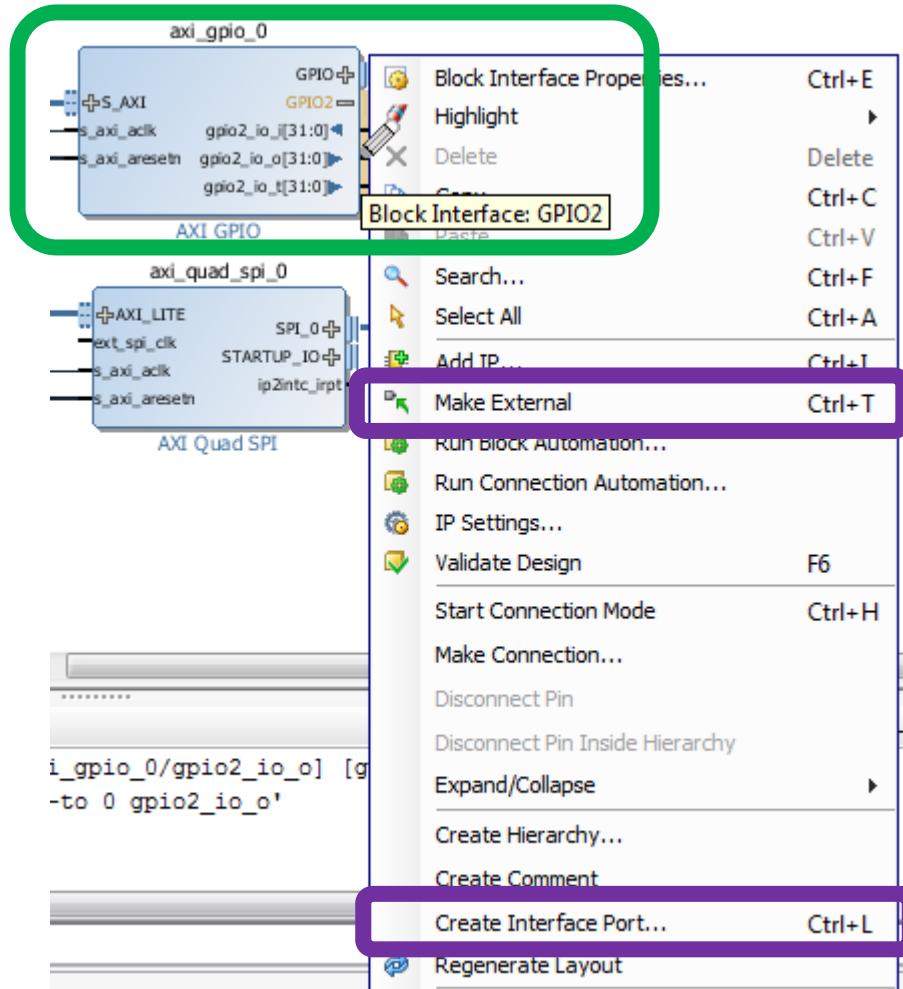
# Run Connection Automation for GPIO IP



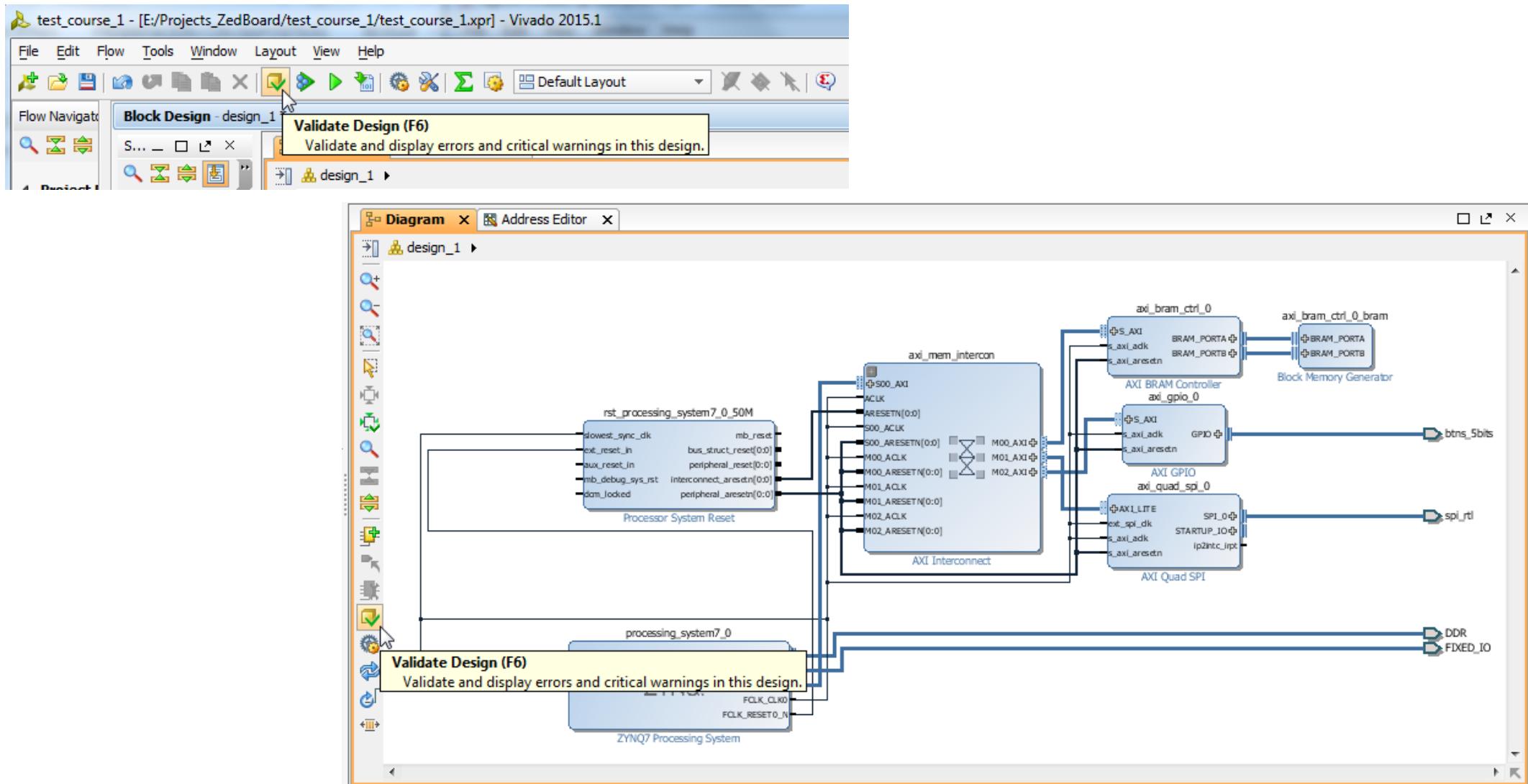
# GPIO IP Connected



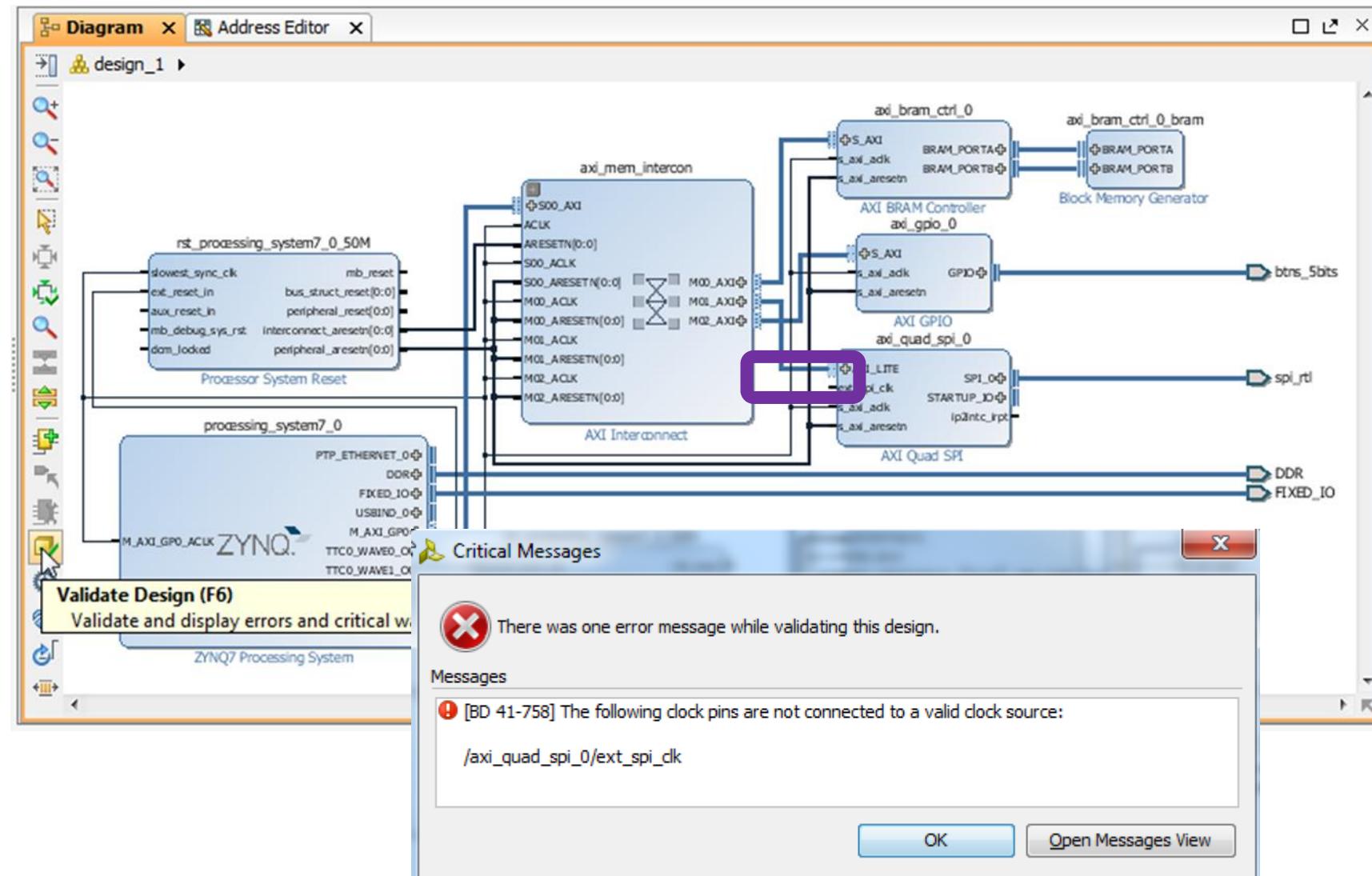
# Options for External Connections



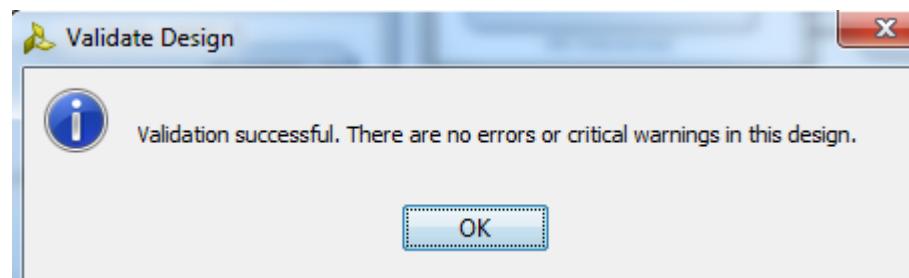
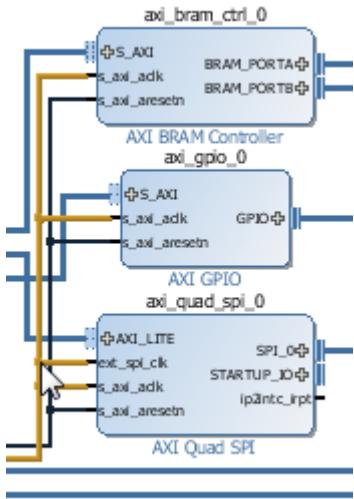
# DRC (Design Rule Check) Design Validation



# DRC – Design Validation

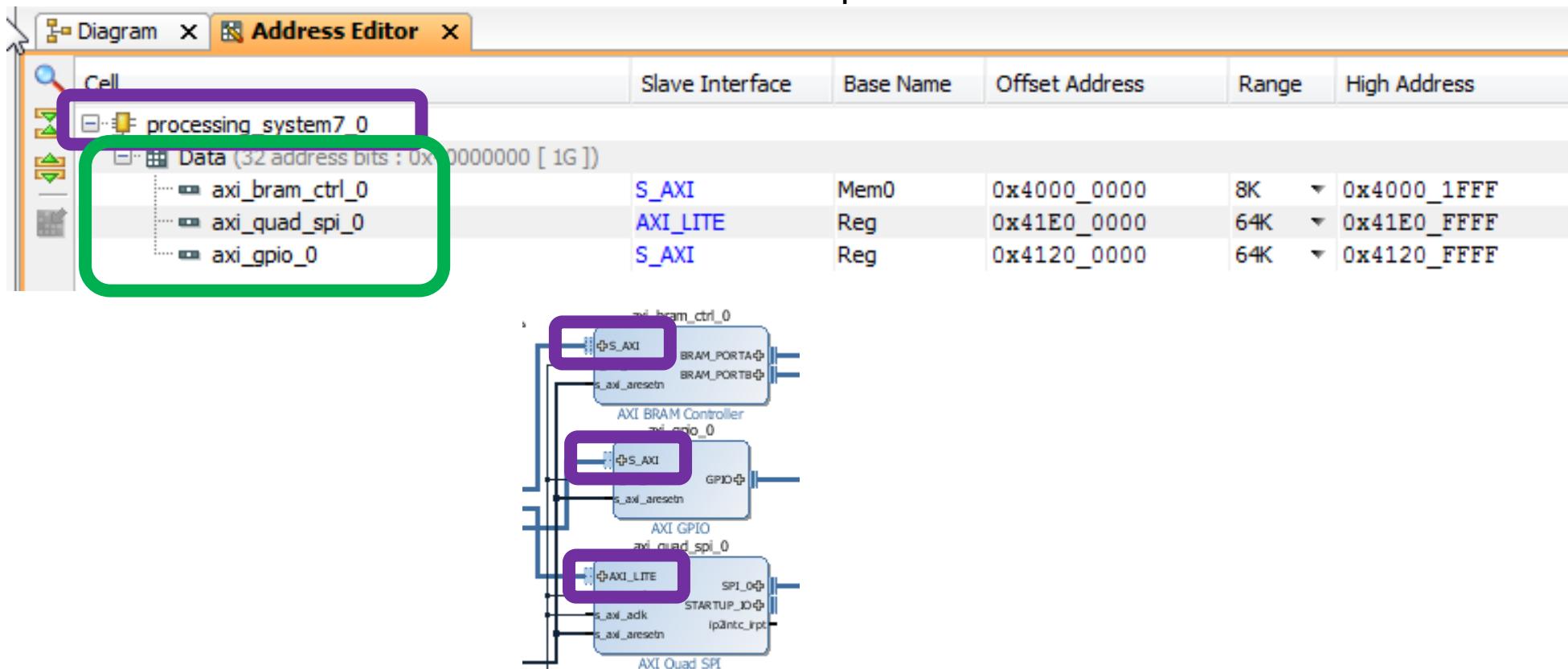


# DRC

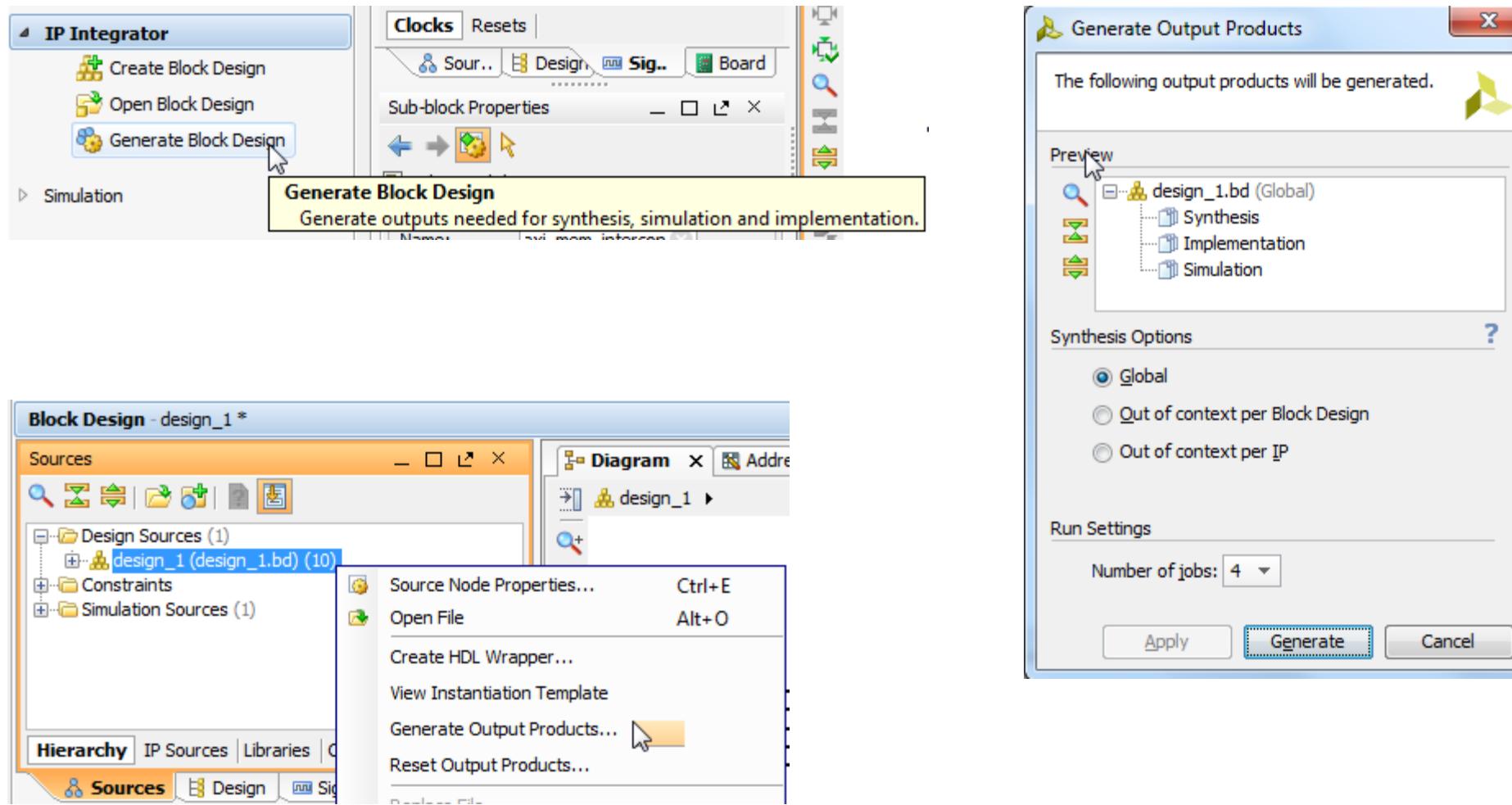


# Address Map

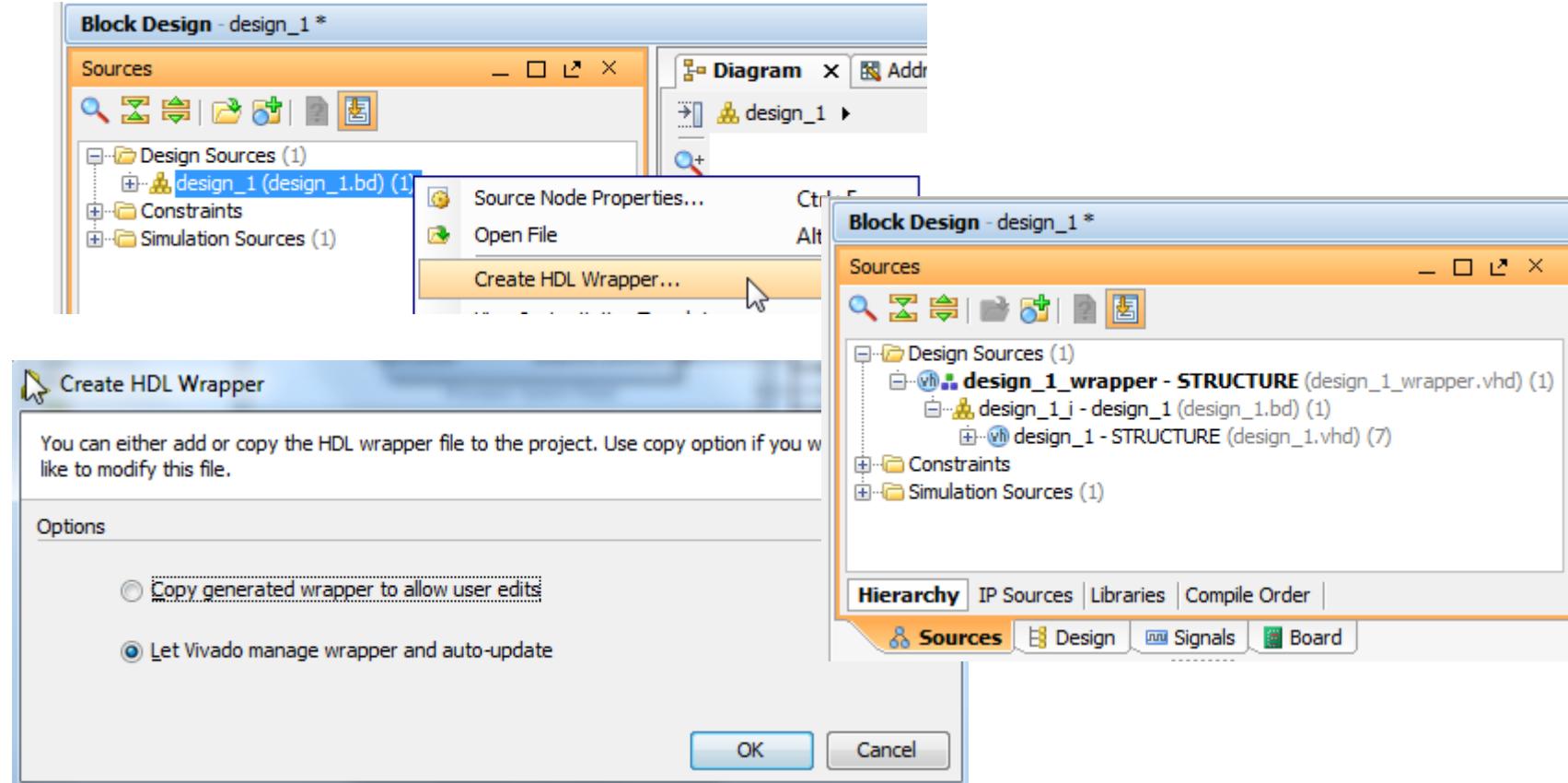
Memory-Map  
Address Space



# Generating Output Products

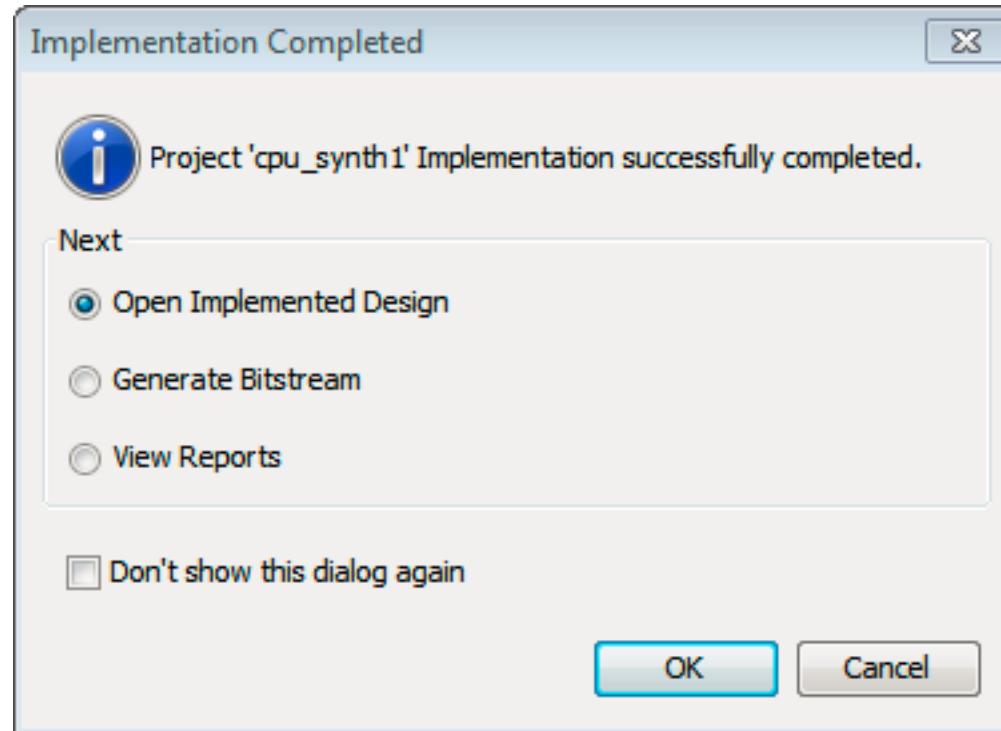


# Creating an HDL Wrapper



# After Completing Implementation

---



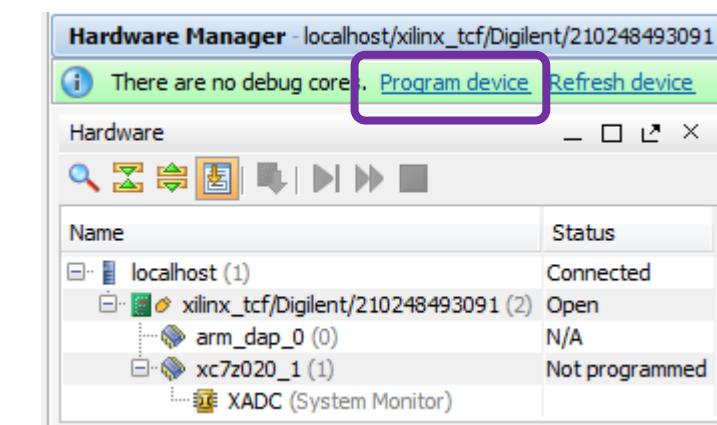
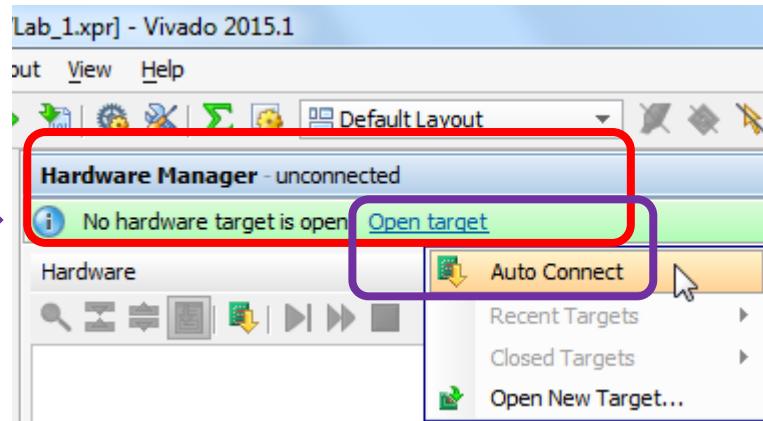
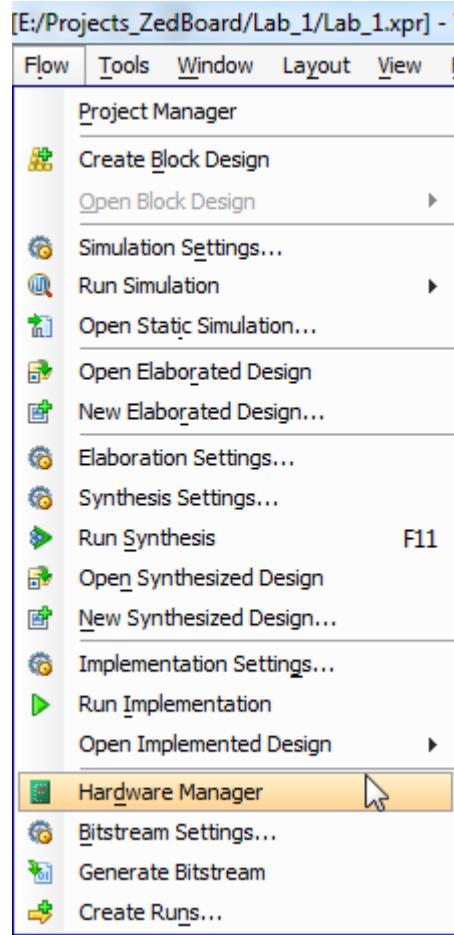
# *Vivado Design Suite*

## *Generate Bit Stream Process*

## *Configuring FPGA Process*

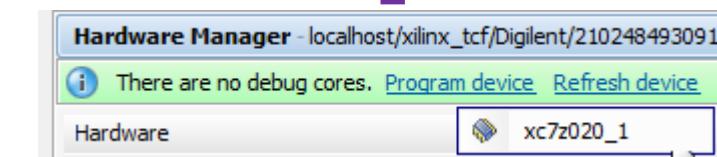
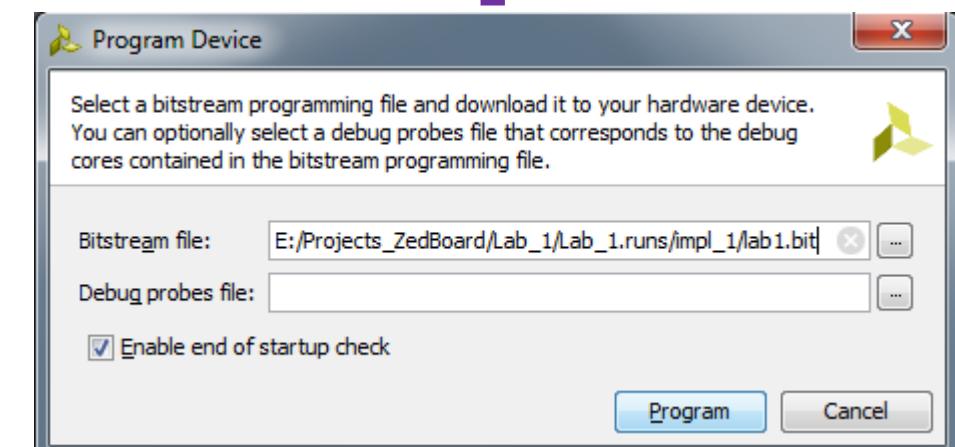
---

# Steps to Configure *only* the PL



Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210248493091 (2)	Open
arm_dap_0 (0)	N/A
xc7z020_1 (1)	Programmed
XADC (System Monitor)	

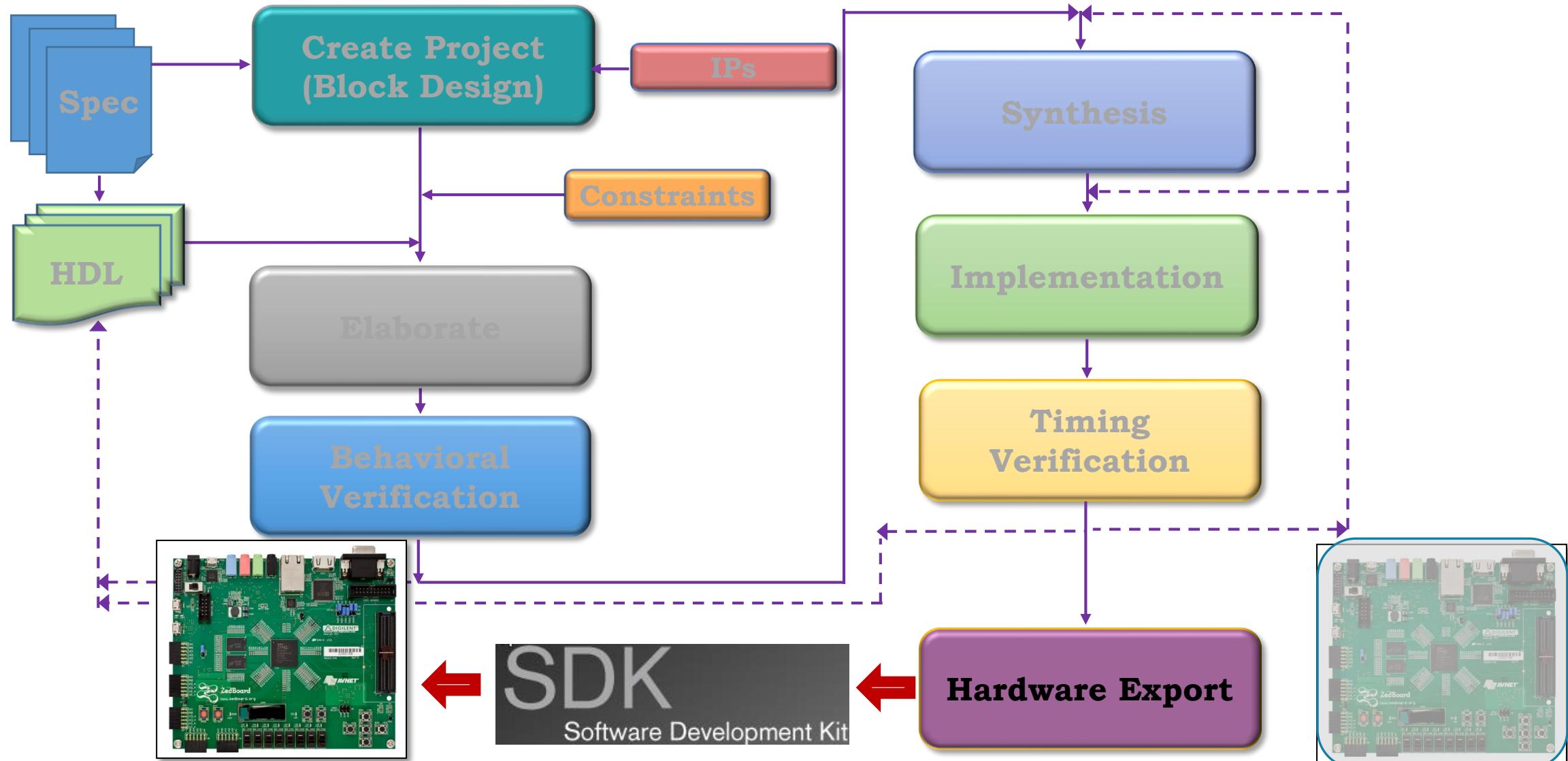
Blue  
"Done"  
LED



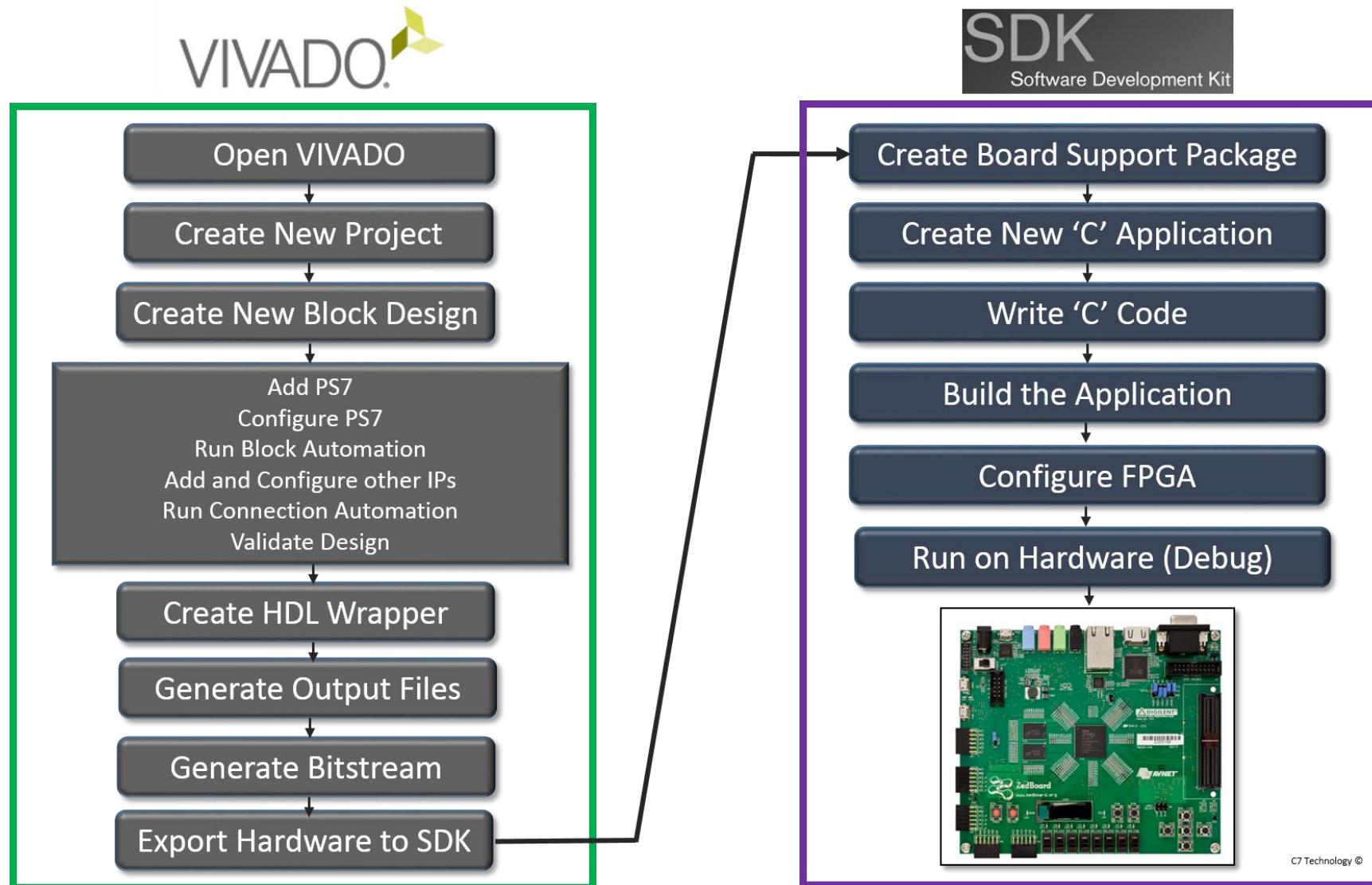
# Software Development Kit (SDK)

---

# Embedded System Design – Vivado-SDK Flow



# Embedded System Design – Vivado-SDK Flow



# Embedded System Tools: Software

---

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

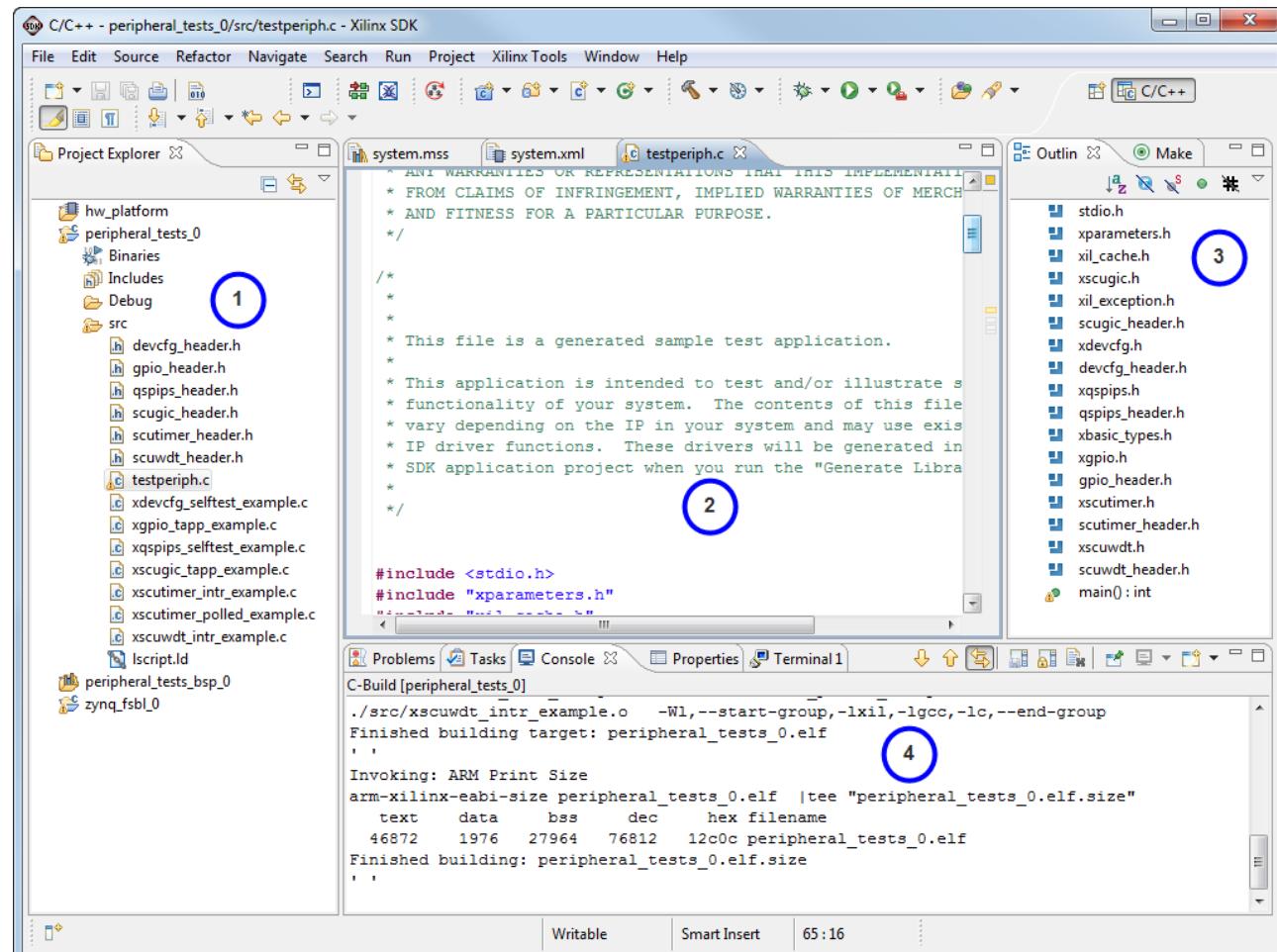
- Board support package creation : LibGen
- GNU software development tools
- C/C++ compiler for the ARM Cortex-A9 processor (gcc)
- Debugger for the ARM Cortex-A9 processor (gdb)

## Board support packages (BSPs)

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

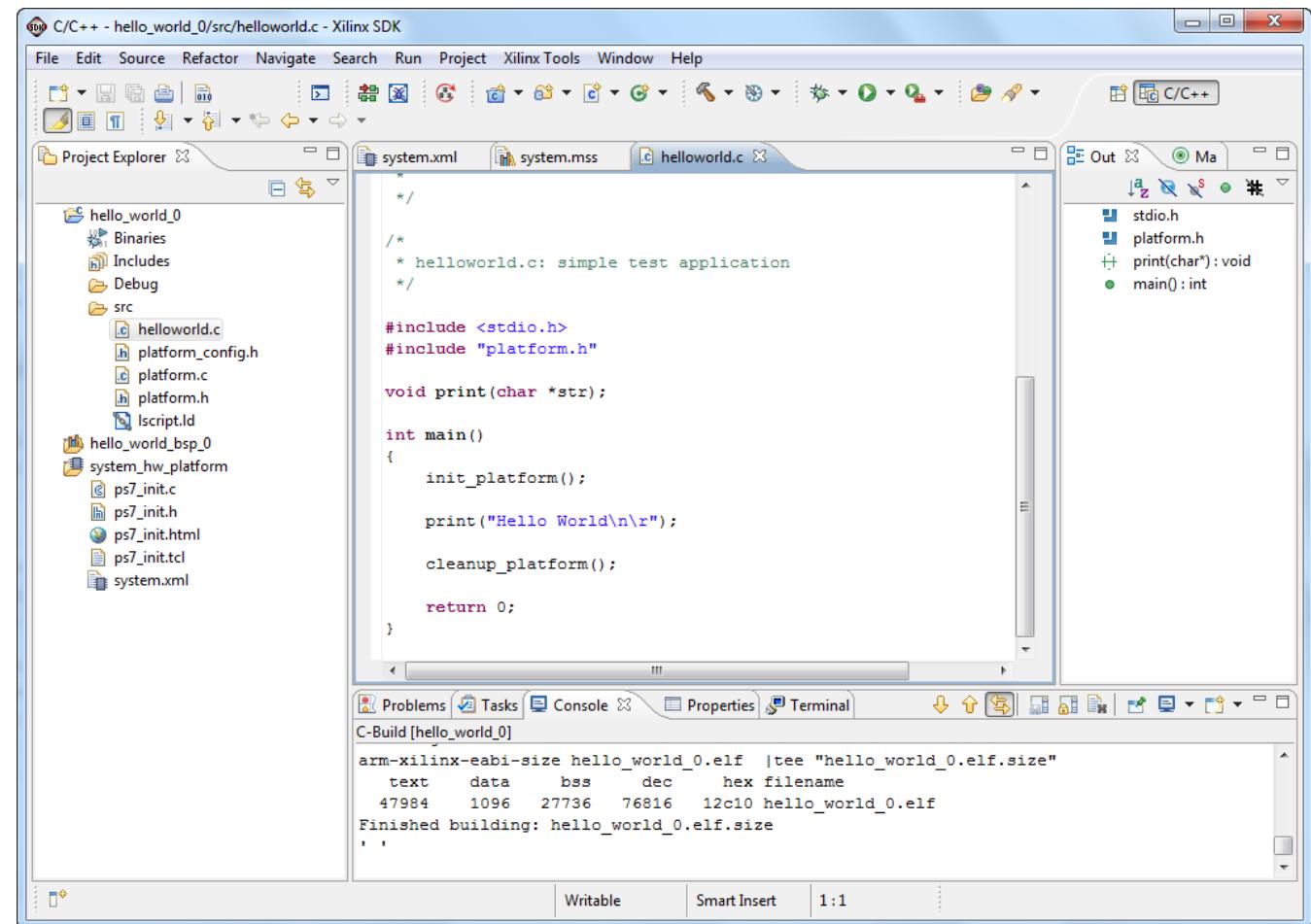
# SDK Workbench Views

- 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



# Build Software Application in SDK

- ❖ Create software platform
  - ❖ System software, board support package
  - ❖ LibGen program
- ❖ Create software application
- ❖ Optionally, create linker script
- ❖ Build project
  - ❖ Compile, assemble, link output file  
*<app\_project>.elf*



The screenshot shows the Xilinx SDK IDE interface. The title bar reads "C/C++ - hello\_world\_0/src/helloworld.c - Xilinx SDK". The left pane is the "Project Explorer" showing the project structure:

- hello\_world\_0
- Binaries
- Includes
- Debug
- src
  - helloworld.c
  - platform\_config.h
  - platform.c
  - platform.h
  - lscript.ld
- hello\_world\_bsp\_0
- system\_hw\_platform
  - ps7\_init.c
  - ps7\_init.h
  - ps7\_init.html
  - ps7\_init.tcl
- system.xml

The main editor window displays the content of helloworld.c:

```
/*
 * helloworld.c: simple test application
 */

#include <stdio.h>
#include "platform.h"

void print(char *str);

int main()
{
    init_platform();

    print("Hello World\n\r");

    cleanup_platform();

    return 0;
}
```

The bottom pane shows the terminal output of the build process:

```
C-Build [hello_world_0]
arm-xilinx-eabi-size hello_world_0.elf | tee "hello_world_0.elf.size"
text      data      bss      dec      hex filename
47984     1096    27736    76816   12c10 hello_world_0.elf
Finished building: hello_world_0.elf.size
'
```

# Apendix

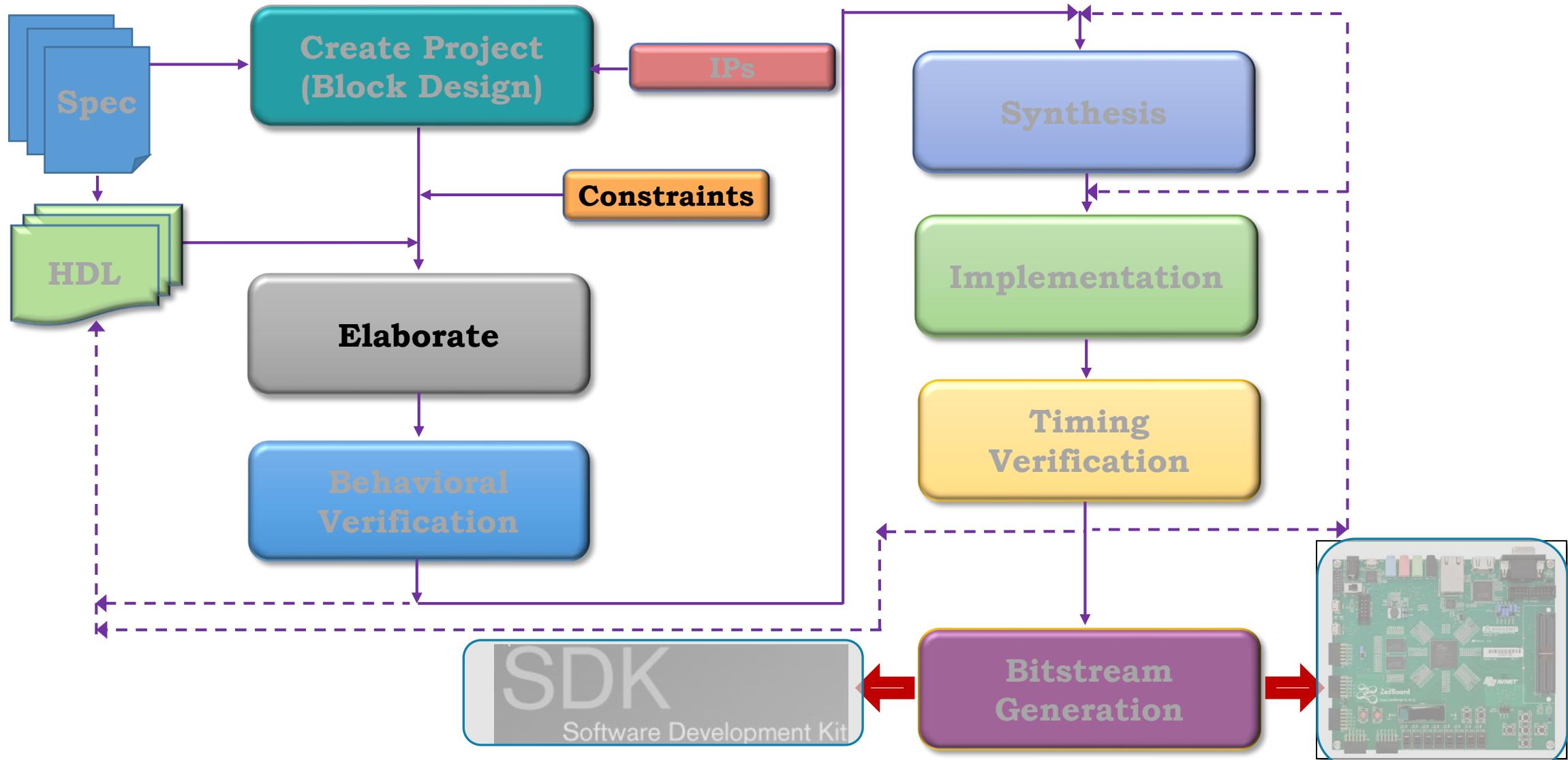
---

# *Vivado Design Suite*

## *Elaboration Process*

---

# Embedded System Design – Vivado Flow



# Elaboration

---

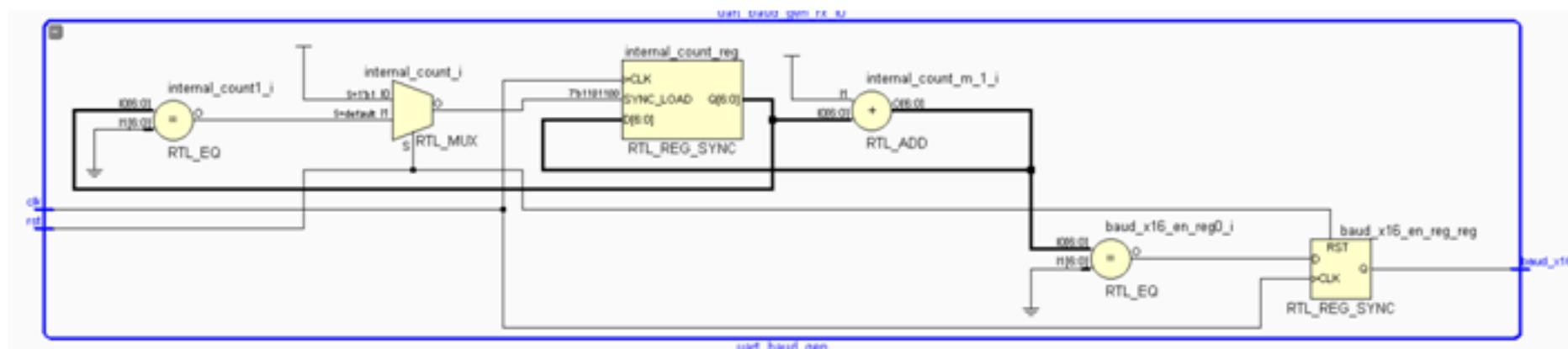
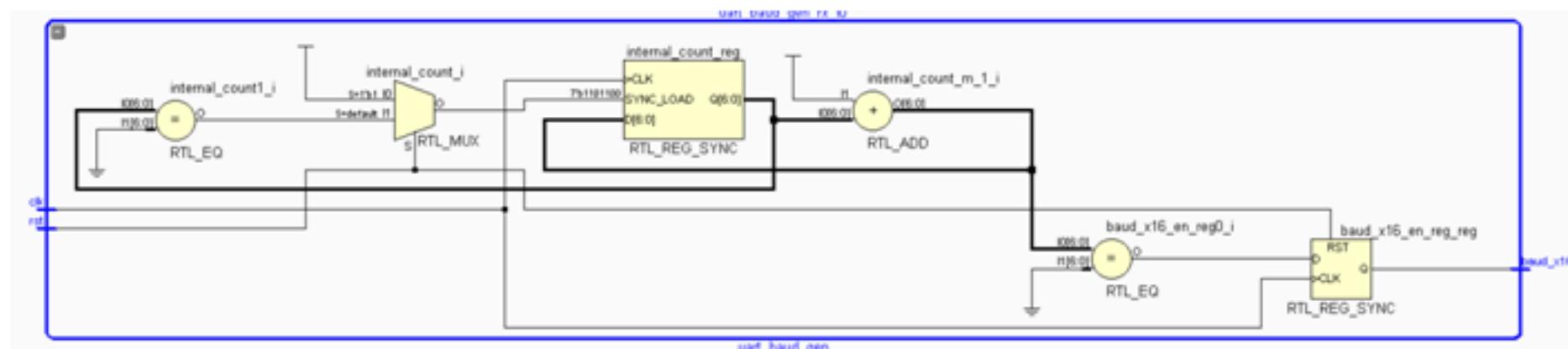
- Elaboration is the RTL optimization to an FPGA technology
- Vivado IDE allows designers to import and manage RTL sources
  - Verilog, System Verilog, VHDL, NGC, or testbenches
- Create and modify sources with the RTL Editor
  - Cross-selection between all the views
- Sources view
  - Hierarchy view: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

# Elaborated Design

Accessed through the Flow Navigator by selecting Open Elaborated Design

Representation of the design before synthesis

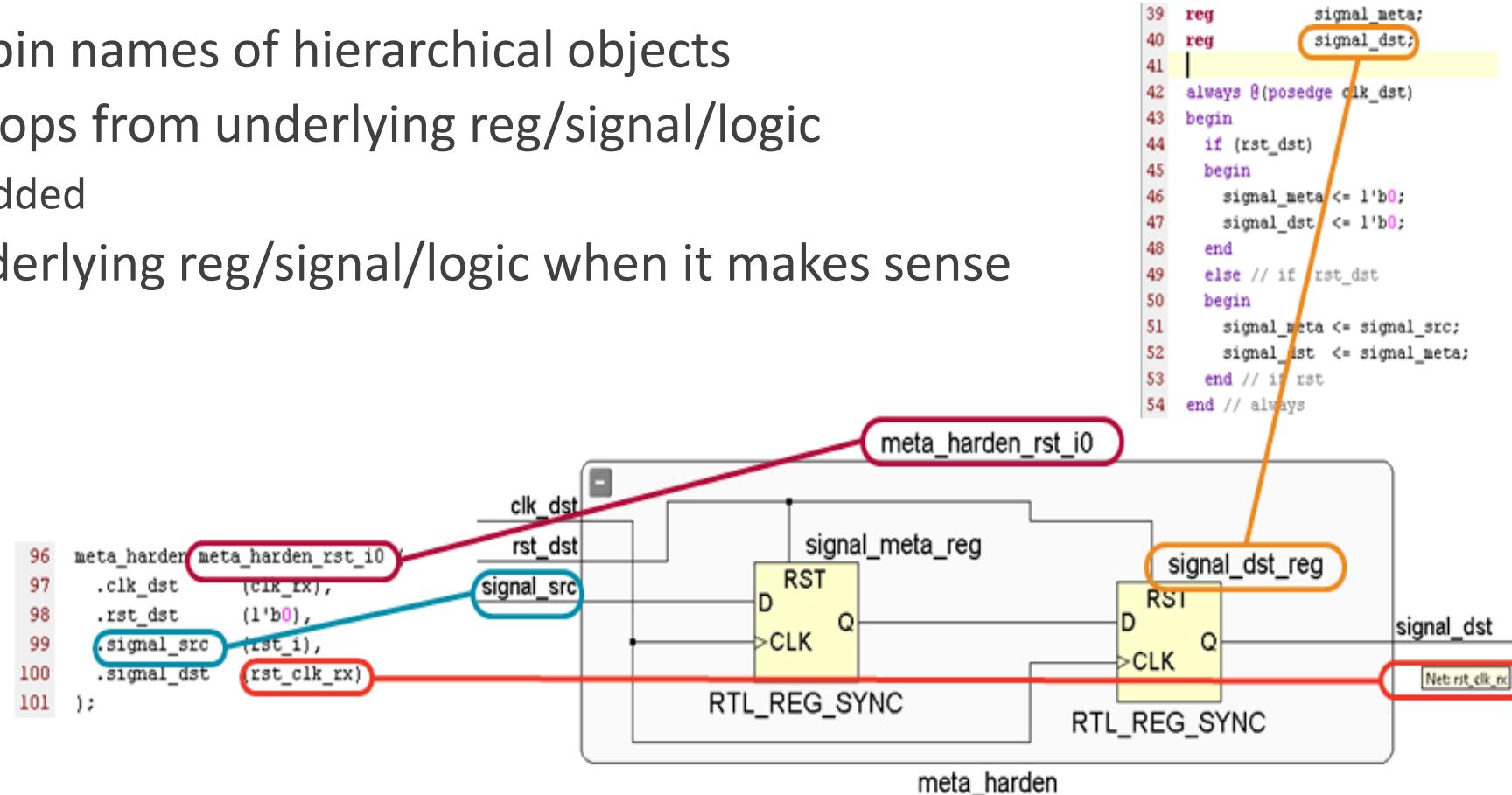
- Interconnected netlist of hierarchical and generic technology cells
  - Instances of modules/entities
  - Generic technology representations of hardware components
    - AND, OR, buffer, multiplexers, adders, comparators, etc...



# 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



# Elaboration and Analysis

---

In a RTL based design, elaboration is the first step

Click on the Open Elaborated Design under RTL Analysis to

- Compile the RTL source files and load the RTL netlist for interactive analysis

You can check RTL structure, syntax, and logic definitions

Analysis and reporting capabilities include:

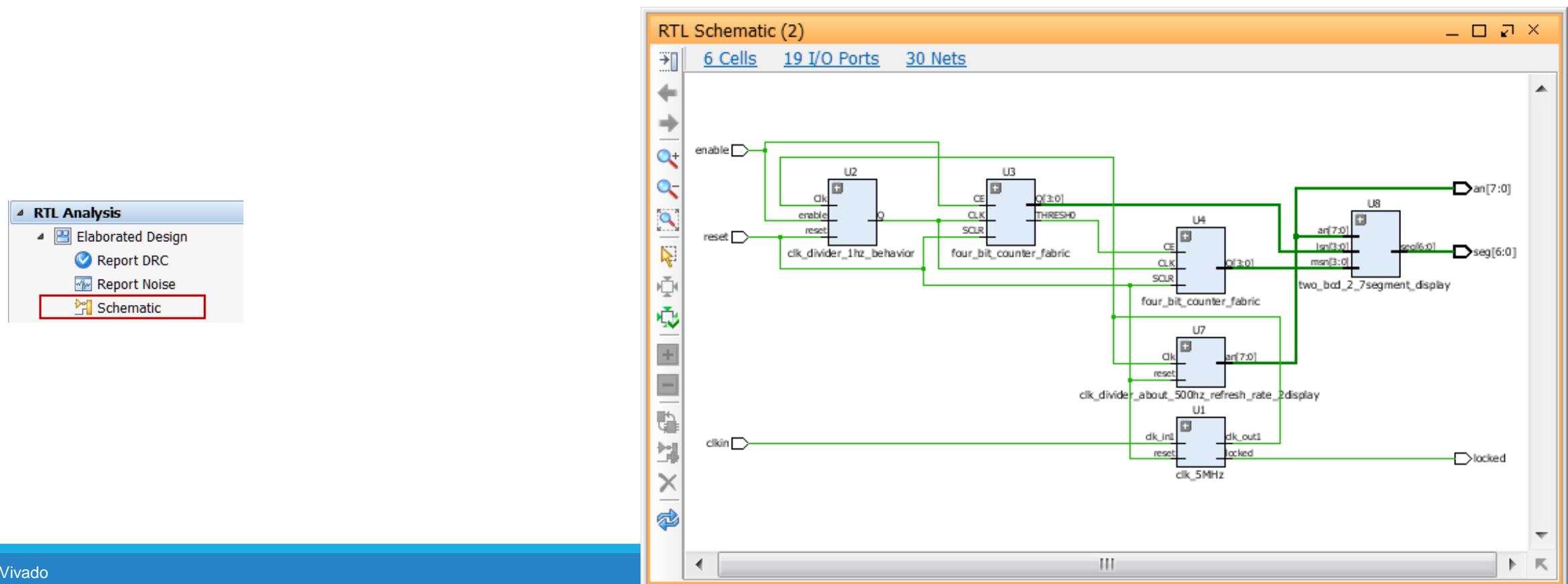
- RTL compilation validation and syntax checking
- Netlist and schematic exploration
- Design rule checks
- Early I/O pin planning using an RTL port list
- Ability to select an object in one view and cross probe to the object in other views, including instantiations and logic definitions within the RTL source files



# Schematic View of an Elaborated Design

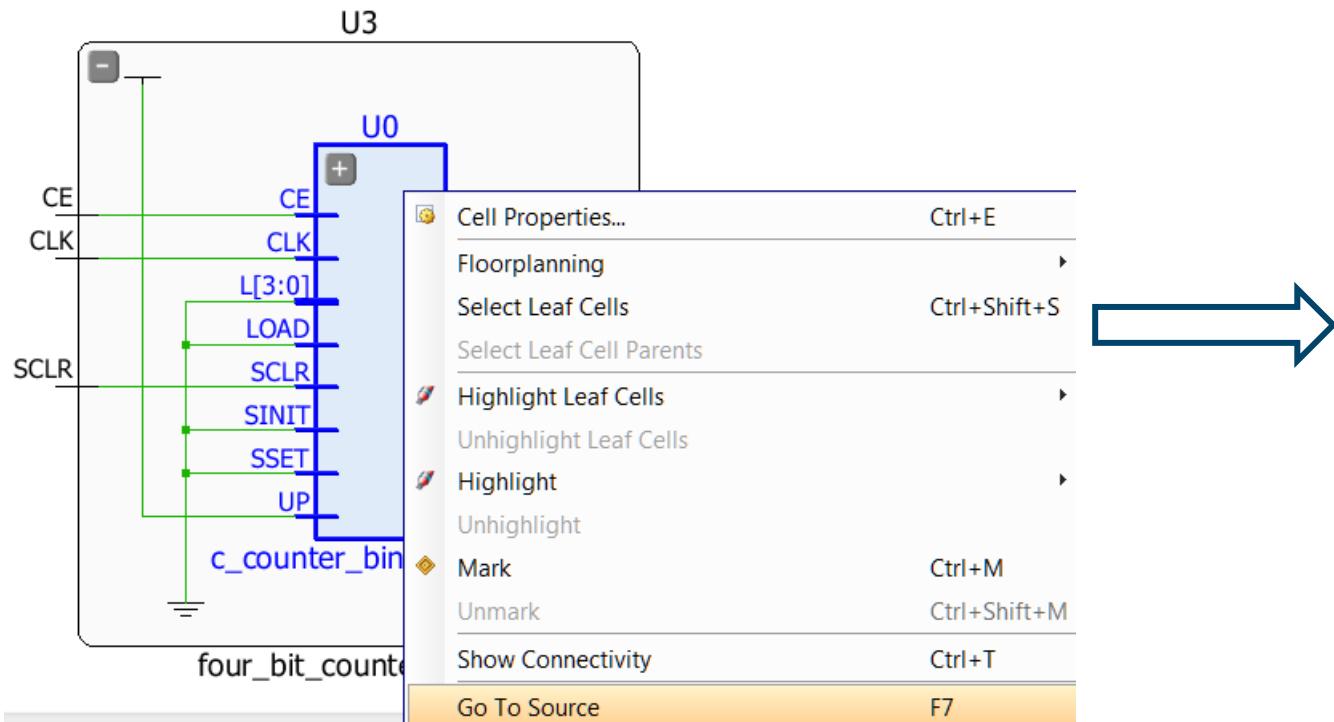
When Schematic is clicked under the Elaborated Design, the schematic is opened showing the hierarchical blocks

- Note that no IO buffers are inferred at this stage



# Cross Probing

Select an object in the schematic, right-click, and select Go To Source to view where the object is defined in the source file



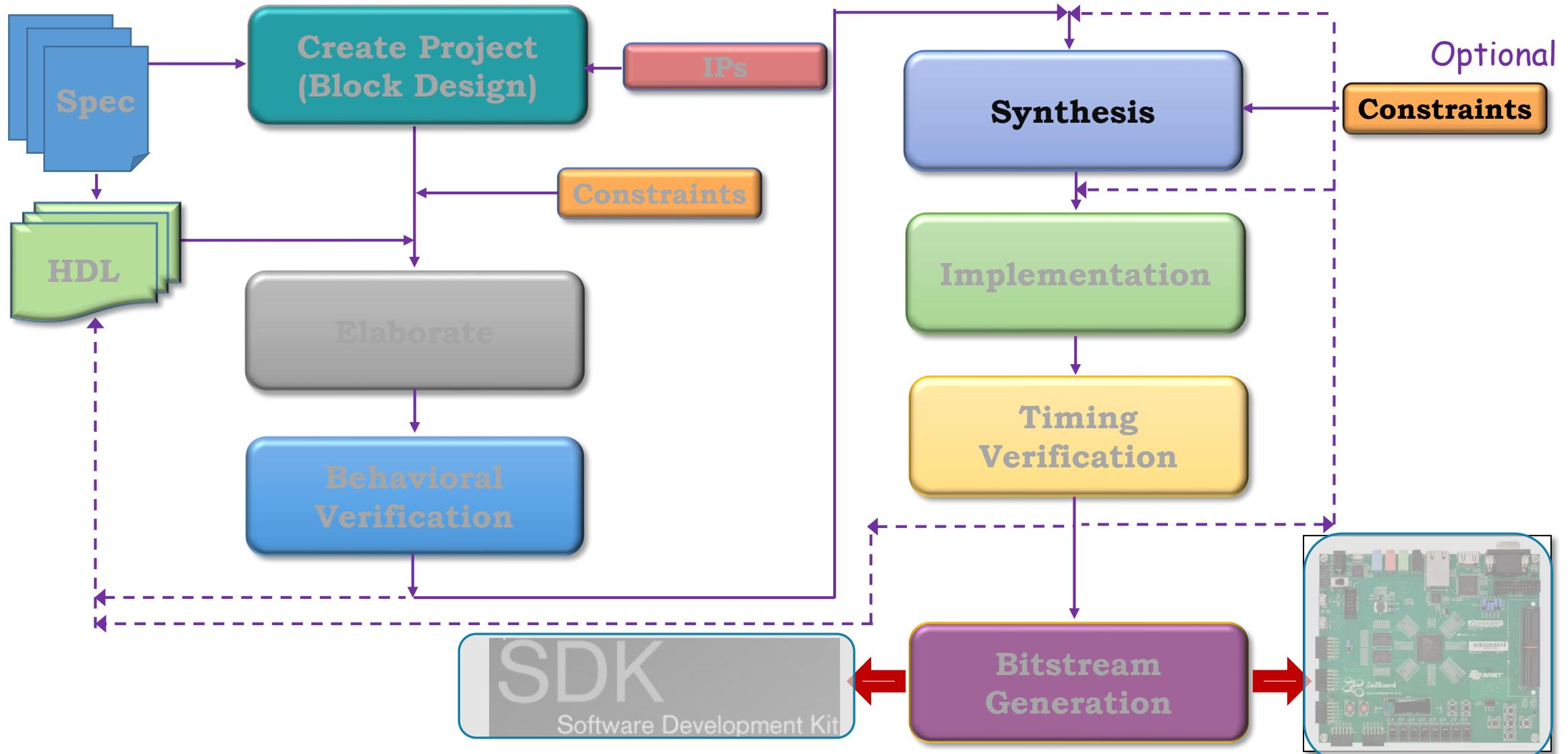
```
114 BEGIN
115   U0 : c_counter_binary_v12_0
116   GENERIC MAP (
117     C_IMPLEMENTATION => 0,
118     C_VERBOSITY => 0,
119     C_XDEVICEFAMILY => "artix7",
120     C_WIDTH => 4,
121     C_HAS_CE => 1,
122     C_HAS_SCLR => 1,
123     C_RESTRICT_COUNT => 1,
124     C_COUNT_TO => "1001",
125     C_COUNT_BY => "1",
126     C_COUNT_MODE => 0,
127     C_THRESH0_VALUE => "1001",
128     C_CE_OVERRIDES_SYNC => 0,
129     C_HAS_THRESH0 => 1,
130     C_HAS_LOAD => 0,
131     C_LOAD_LOW => 0,
132     C_LATENCY => 1,
133     C_FB_LATENCY => 0,
134     C_AINIT_VAL => "0",
135     C_SINIT_VAL => "0",
136     C_SCLR_OVERRIDES_SSET => 1,
137     C_HAS_SSET => 0,
138     C_HAS_SINIT => 0
139   )
```

# *Vivado Design Suite*

## *Synthesis Process*

---

# Embedded System Design – Vivado Flow



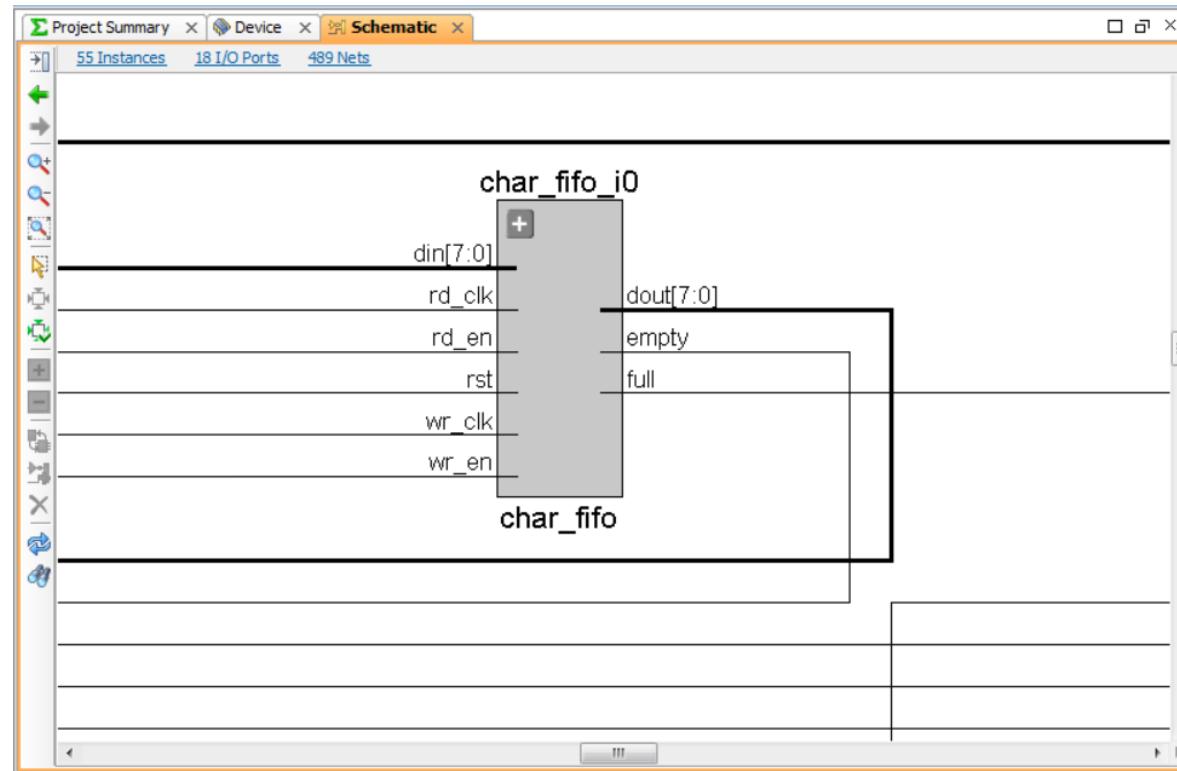
# Vivado IDE Synthesis

---

- **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**

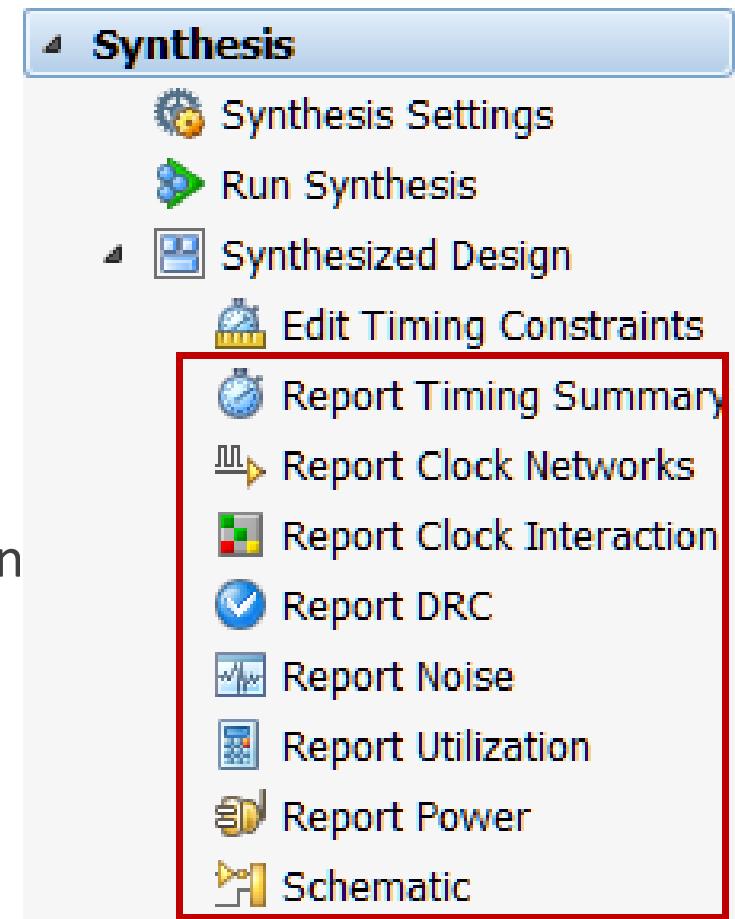
- Interconnected netlist of hierarchical and basic elements (BELs)
  - Instances of modules/entities
  - Basic elements
    - LUTs, flip-flops, carry chain elements, wide MUXes
    - Block RAMs, DSP cells
    - Clocking elements (BUFG, BUFR, MMCM, ...)
    - I/O elements (IBUF, OBUF, I/O flip-flops)

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

# Commands Available After Synthesis

Flow Navigator is optimized to provide quick access to the options most frequently used after synthesis

- Report Timing Summary: Generate a default timing report
- Report Clock Networks: Generates a clock tree for the design
- Report Clock Interaction: Verifies constraint coverage on paths between clock domains
- Report DRC: Performs design rule check on the entire design
- Report Noise: Performs an SSO analysis of output and bidirectional pins in the design
- Report Utilization: Generates a graphical version of the Utilization Report
- Report Power: Detailed power analysis reports that can be customized for the power supply and application environment
- Schematic: Opens the Schematic viewer

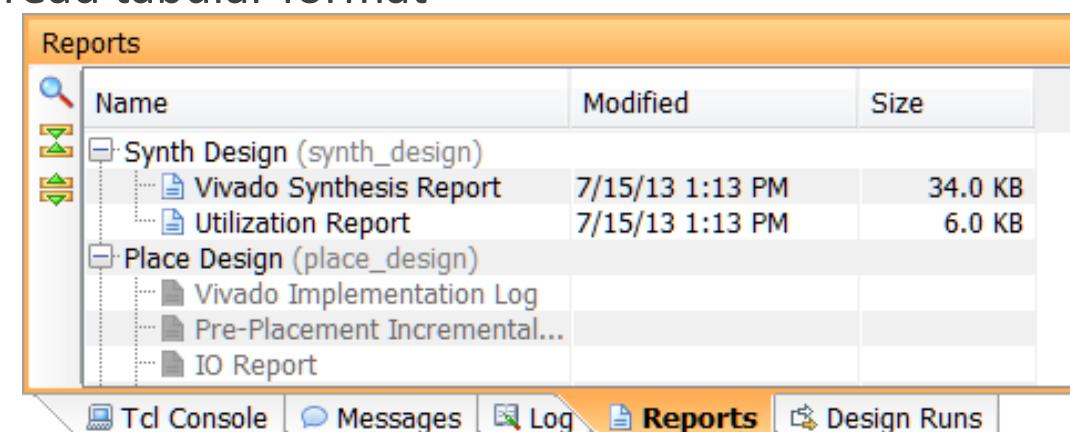


# Synthesis Reports

---

While the Flow Navigator points to the most important reports, the Reports tab contains several other useful reports

- Vivado Synthesis Report shows
  - HDL files synthesized, synthesis progress, timing constraints read, and RTL primitives from the RTL design
  - Timing optimization goals, technology mapping, removed pins/ports, and final cell usage (technology-mapped cell usage)
- Utilization Report shows
  - Technology-mapped cell usage in an easy-to-read tabular format



# Synthesis Utilization Report

Reports slice logic, memory, DSP slice, IO, clocking, and other resources used by the design

```
1 Copyright 1986-1999, 2001-2013 Xilinx, Inc. All Rights Reserved.  
2-----  
3 | Tool Version : Vivado v.2013.2 (win64) Build 272601 Sat Jun 15  
4 | Date       : Mon Jul 15 13:13:54 2013  
5 | Host        : running 64-bit Service Pack 1 (build 7601)  
6 | Command     : report_utilization -file two_digits_counter_on_  
7 | Design      : two_digits_counter_on_2_7segment_display  
8 | Device      : xc7a100t  
9 | Design State: Synthesized  
10-----  
11-----  
12 Utilization Design Information  
13-----  
14 Table of Contents  
15-----  
16 1. Slice Logic  
17 2. Memory  
18 3. DSP  
19 4. IO and GTX Specific  
20 5. Clocking  
21 6. Specific Feature  
22 7. Primitives  
23 8. Black Boxes  
24 9. Instantiated Netlists  
25-----  
26 1. Slice Logic  
27-----
```

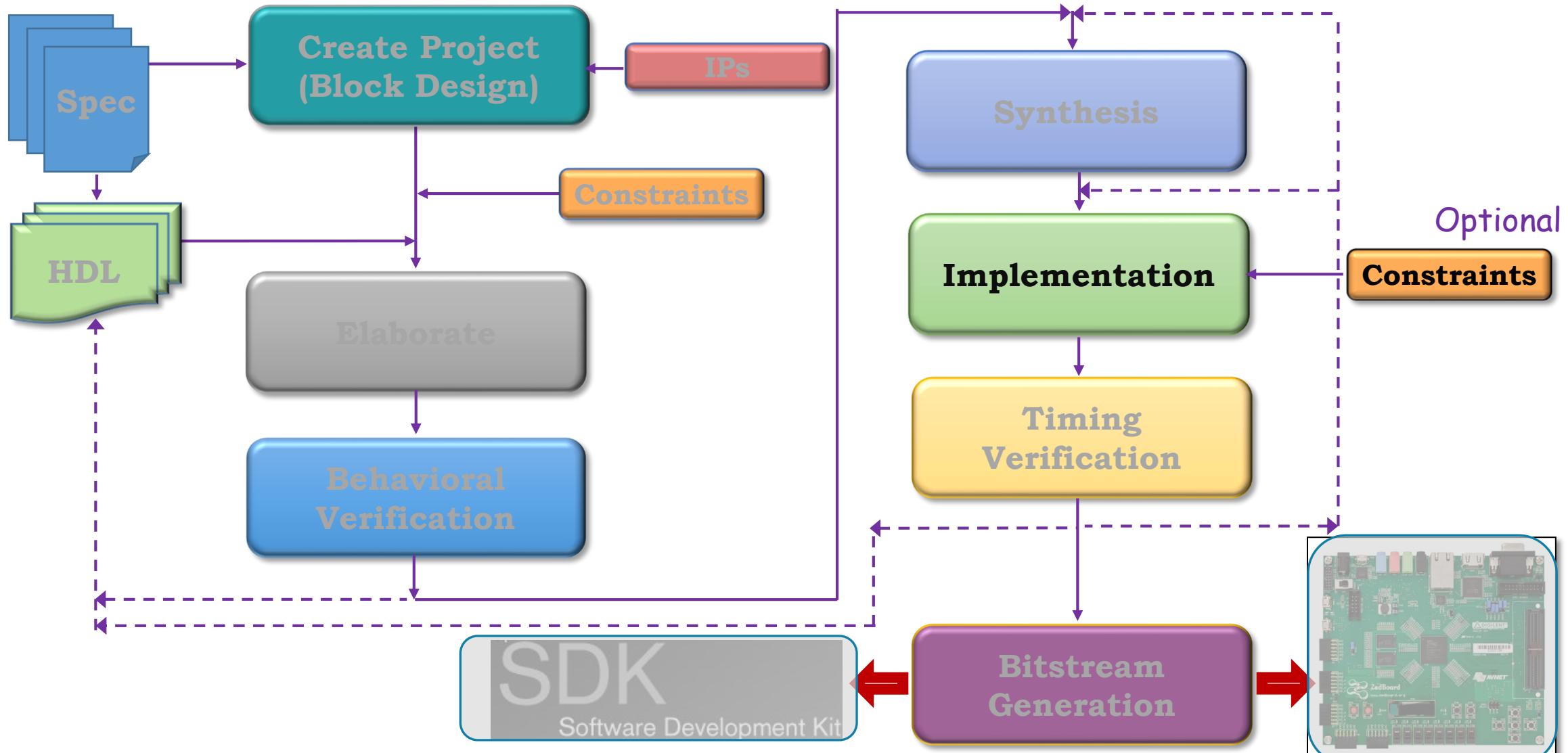
5. Clocking					
Site Type	Used	Loced	Available	Util%	
BUFGCTRL	2	0	32	6.25	
BUFIO	0	0	24	0.00	
MMCME2_ADV	1	0	6	16.66	67 4. IO and GTX Specific
PILLE2_ADV	0	0	6	0.00	68-----
BUFMRC	0	0	12	0.00	69
BUFHCE	0	0	96	0.00	70-----
BUFR	0	0	24	0.00	71-----
Site Type	Used	Loced	Available	Util%	
Bonded IOB	19	0	210	9.04	
Bonded IPADs	0	0	2	0.00	
IBUFGDS	0	0	202	0.00	
IDELAYCTRL	0	0	6	0.00	
IN_FIFO	0	0	24	0.00	
OUT_FIFO	0	0	24	0.00	
PHASER_REF	0	0	6	0.00	
PHY_CONTROL	0	0	6	0.00	
PHASER_OUT/PHASER_OUT_PHY	0	0	24	0.00	
PHASER_IN/PHASER_IN_PHY	0	0	24	0.00	
IDELAYE2/IDELAYE2_FINEDELAY	0	0	300	0.00	
ODELAYE2/ODELAYE2_FINEDELAY	0	0	0	0.00	
IBUFDS_GTE2	0	0	4	0.00	
ILOGIC	0	0	210	0.00	
OLOGIC	0	0	210	0.00	

# *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

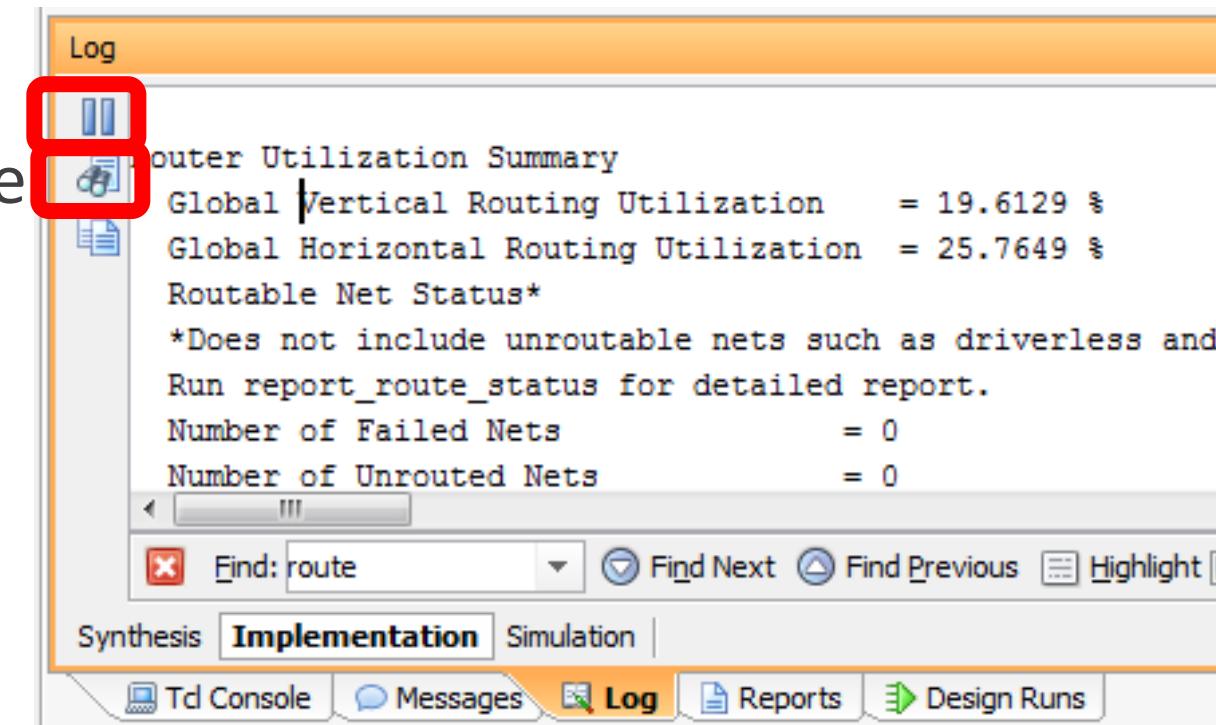
- Package pin placement
- Absolute or relative placement of cells:
  - Block RAM
  - DSP
  - LUTs
  - Flip-Flops
- Floorplanning constraints that assign cells to general regions of an FPGA

**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

# Implementation Log Messages

## Viewing the Log in the Log Window

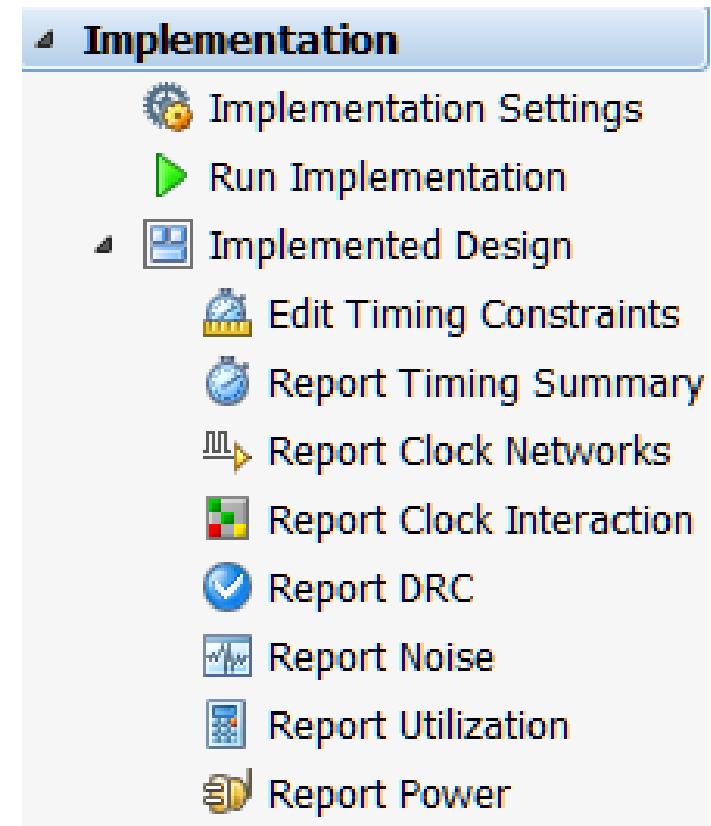
The Log window opens in the Vivado IDE after you launch a run. It shows the standard output messages. It also displays details about the progress of each individual implementation process, such as *place\_design* and *route\_design*.



# After Implementation

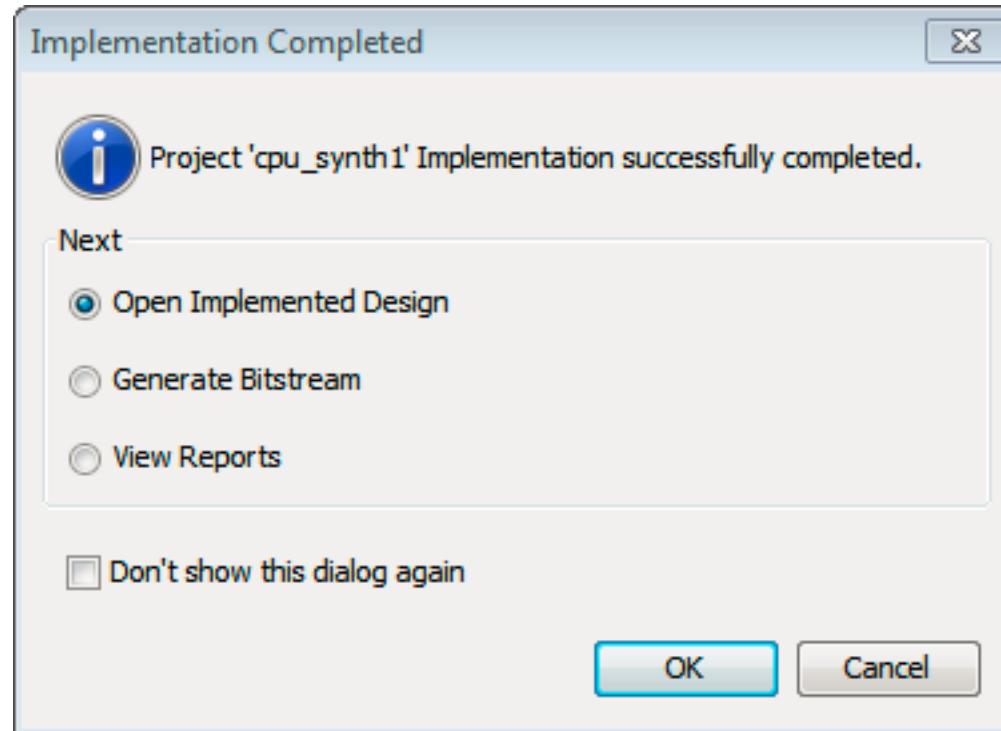
---

- 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



# After Completing Implementation

---

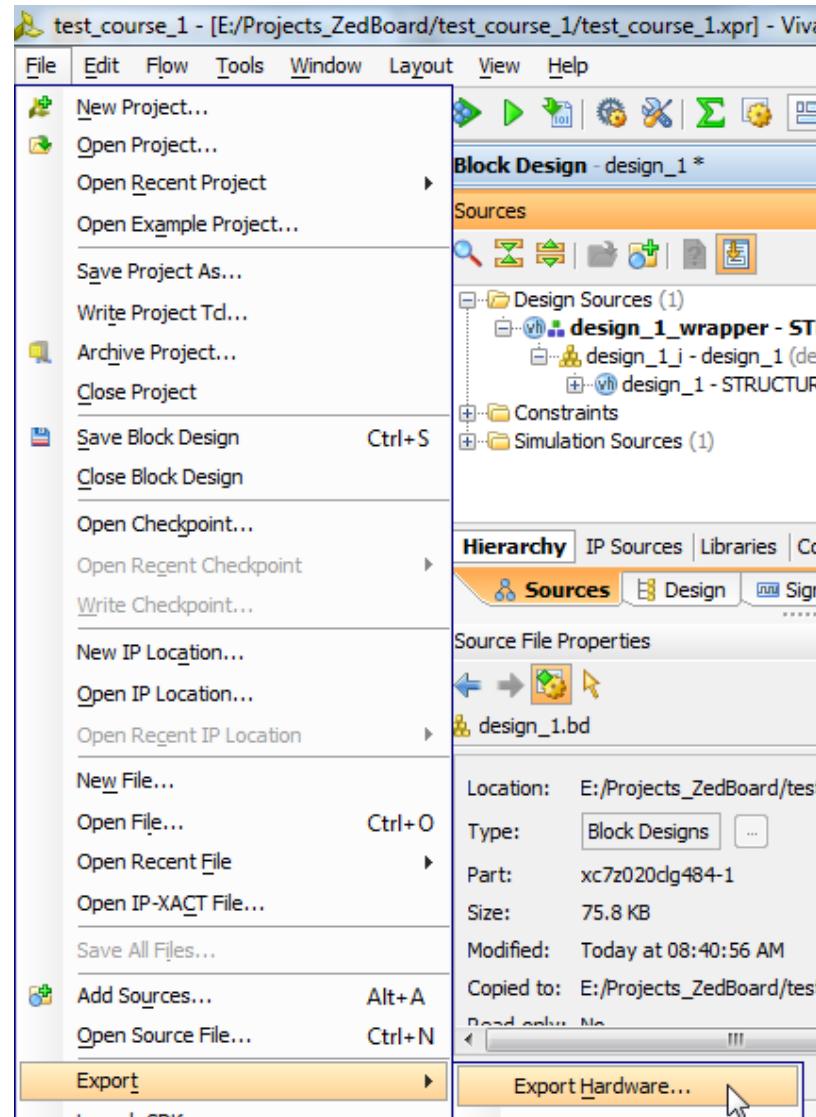


# Implementation Out-of-Date Message

---



# Exporting a Hardware Description



# Export Hardware Design to SDK

Software development is performed with the Xilinx Software Development Kit tool (SDK)

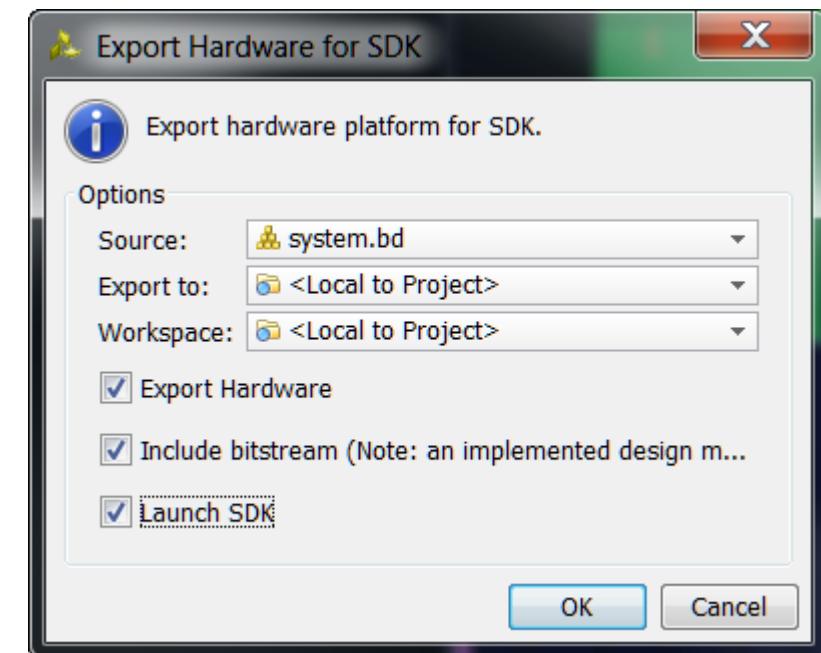
The design must be opened if a bitstream of the design is generated

The Block design must be open before the design can be exported

An XML description of the hardware is imported in the SDK tool

- The hardware platform is built on this description
- Only one hardware platform for an SDK project

The SDK tool will then associate user software projects to hardware



# Exporting IP Integrator Design to SDK – Main Files

---

File	Description
system.xml	This file opens by default when you launch SDK and displays the address map of your system
ps7_init.c s7_init.h	The ps7_init.c and ps7_init.h files contain the initialization code for the Zynq Processing System and initialization settings for DDR, clocks, PLLs and MIOs. SDK uses these settings when initializing the PS so applications can run on top of the PS.
ps7_init.tcl	This is the Tcl version of the <b><i>init</i></b> file
ps7_init.html	This <b><i>init</i></b> file describes the initialization data.

# *Vivado Design Suite*

## *Basic Static Timing Constraints*

---

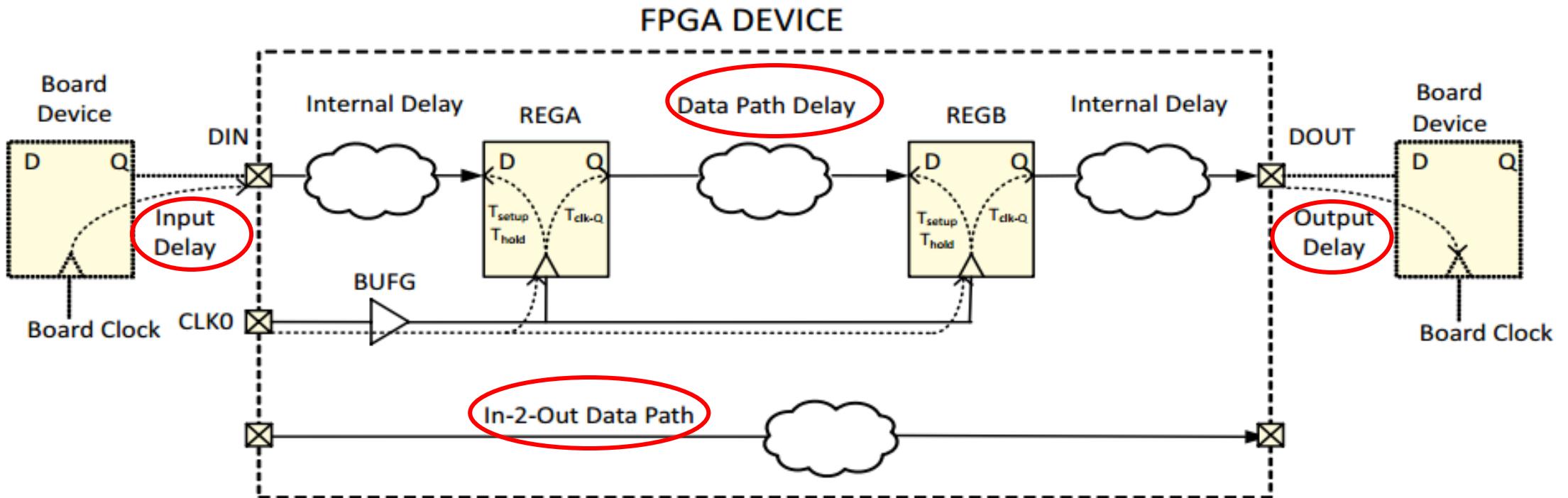
# Basic Timing Constraints

---

There are three basic timing constraints applicable to a sequential machine

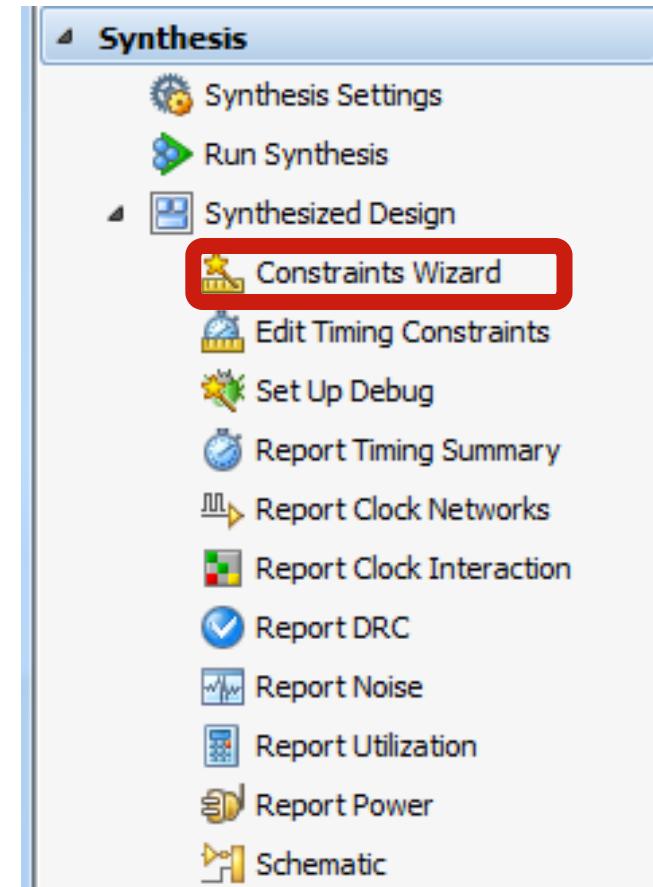
- Period
  - Paths between synchronous elements clocked by the reference clock net
    - Synchronous elements include flip-flops, latches, synchronous RAM, and DSP slices
    - Use `create_clock` to create the constraint
- Input Delay
  - Paths between input pin and synchronous elements
  - Use `set_input_delay`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

The screenshot shows the Vivado Timing Constraints Wizard interface. On the left, the main window displays the 'Primary Clocks' section with a table for recommended constraints. A red box highlights the pencil icon in the toolbar above the table. Below the table is a section for 'Constraints for Pulse Width Check Only' with its own table. At the bottom, there's an 'Tcl Command Preview' pane showing the command `create_clock -name clk [get_ports {clk}]`. Navigation buttons at the bottom include 'Reference', '< Back', 'Next >', 'Skip to Finish >>', and 'Cancel'. On the right, a modal dialog titled 'Primary Clock Constraints' is open, prompting the user to specify 'Frequency' (MHz) and 'Period' (ns), and optionally 'Rise at', 'Fall at', and 'Jitter' values. Both windows have a yellow logo in the top right corner.

**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**

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**

Object	Name	Frequency (MHz)	Period (ns)
clk	clk		

**Tcl Command Preview (1)**

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

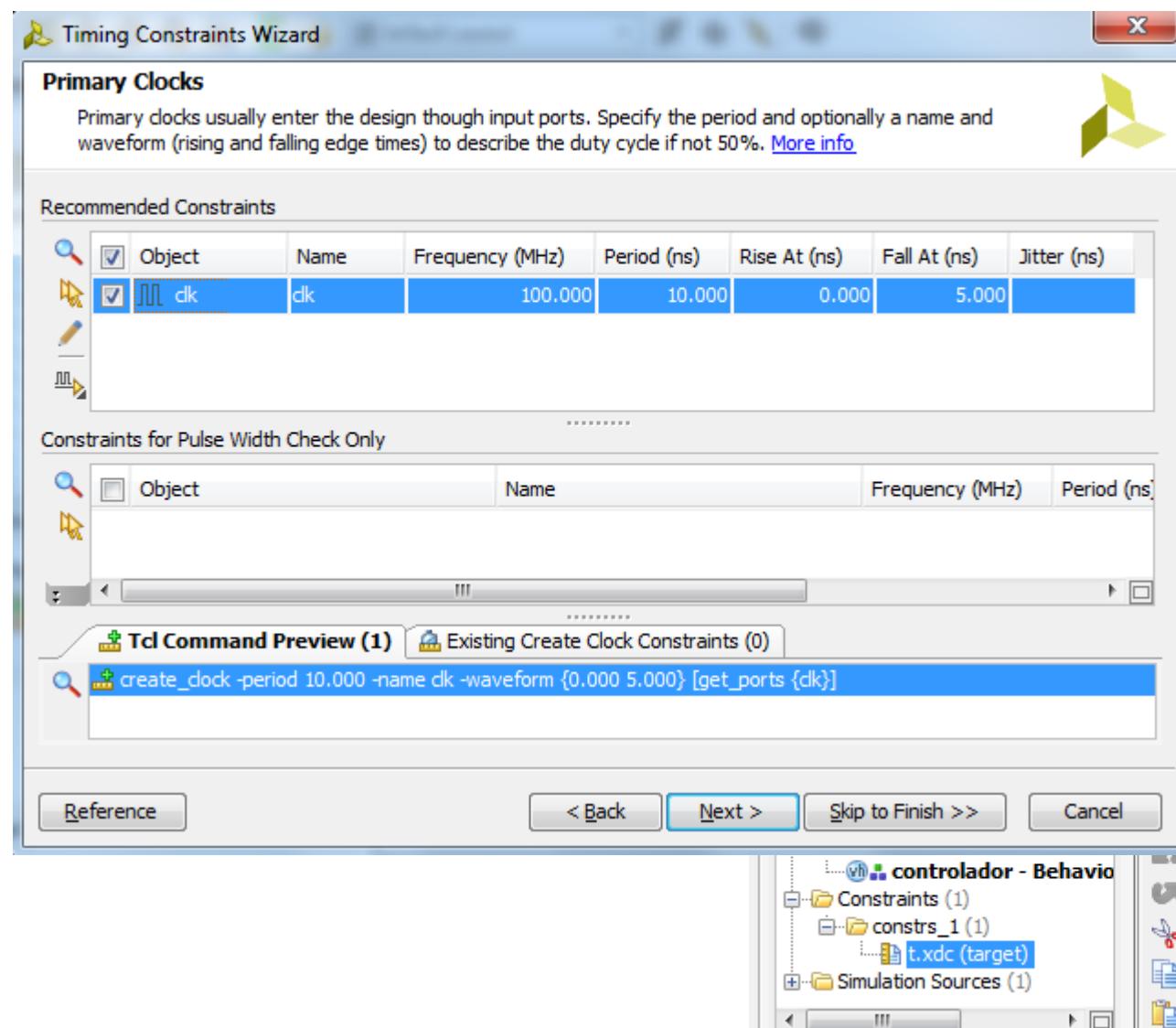
**Primary Clock Constraints**

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

Frequency:	<input type="text"/>	MHz (required)
Period:	<input type="text"/>	ns (required)
Rise at:	<input type="text"/>	ns (optional)
Fall at:	<input type="text"/>	ns (optional)
Jitter:	<input type="text"/>	ns (optional)

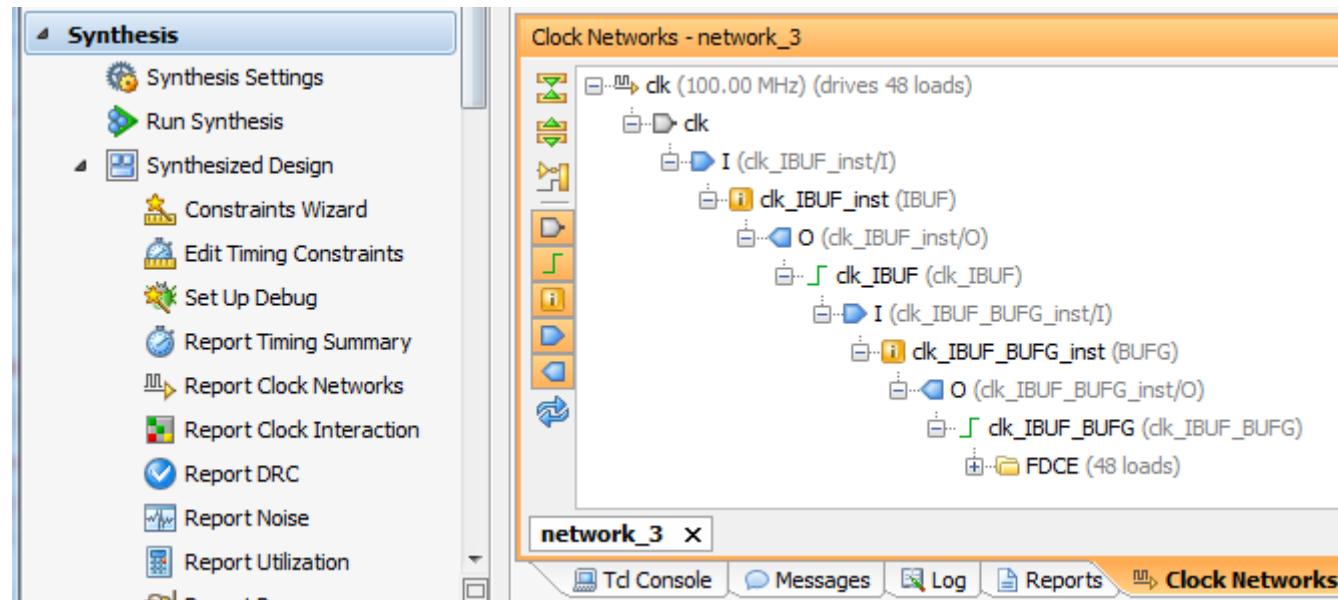
**Buttons:** Reference, < Back, Next >, Skip to Finish >>, Cancel

# Clock Constraint Setting

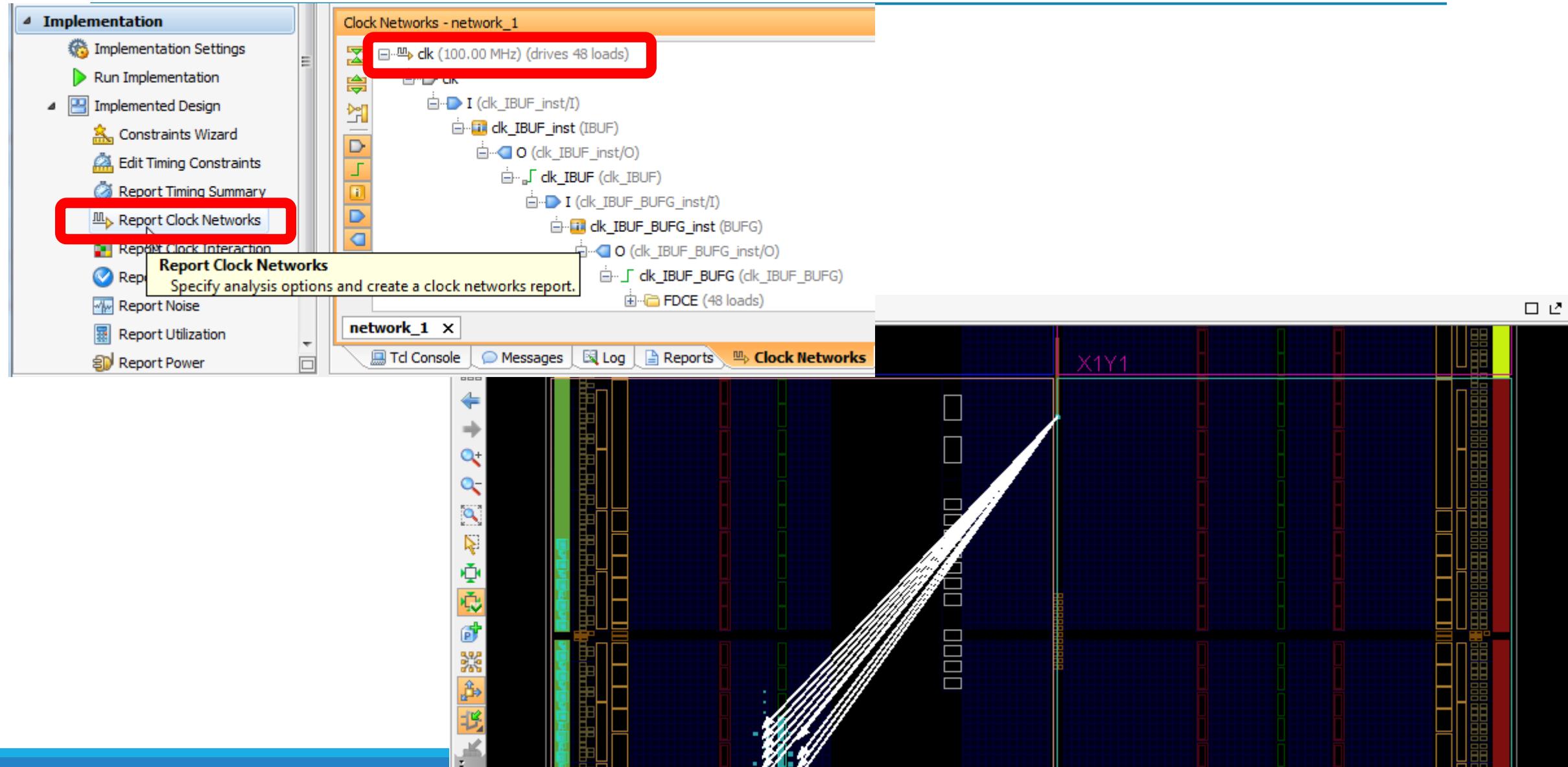


```
Project Summary X Device X Schematic X Schematic (2) X Schematic (3) X t.xdc X  
E:/Projects_FPGA/Proyecto_Clevis/Controlador_X/project_1/project_1.srcs/constrs_1/new/t.xdc  
1 create_clock -period 10.000 -name clk -waveform {0.000 5.000} [get_ports clk]  
2
```

# Clock Network Report



# Clock Network Report and Visualization



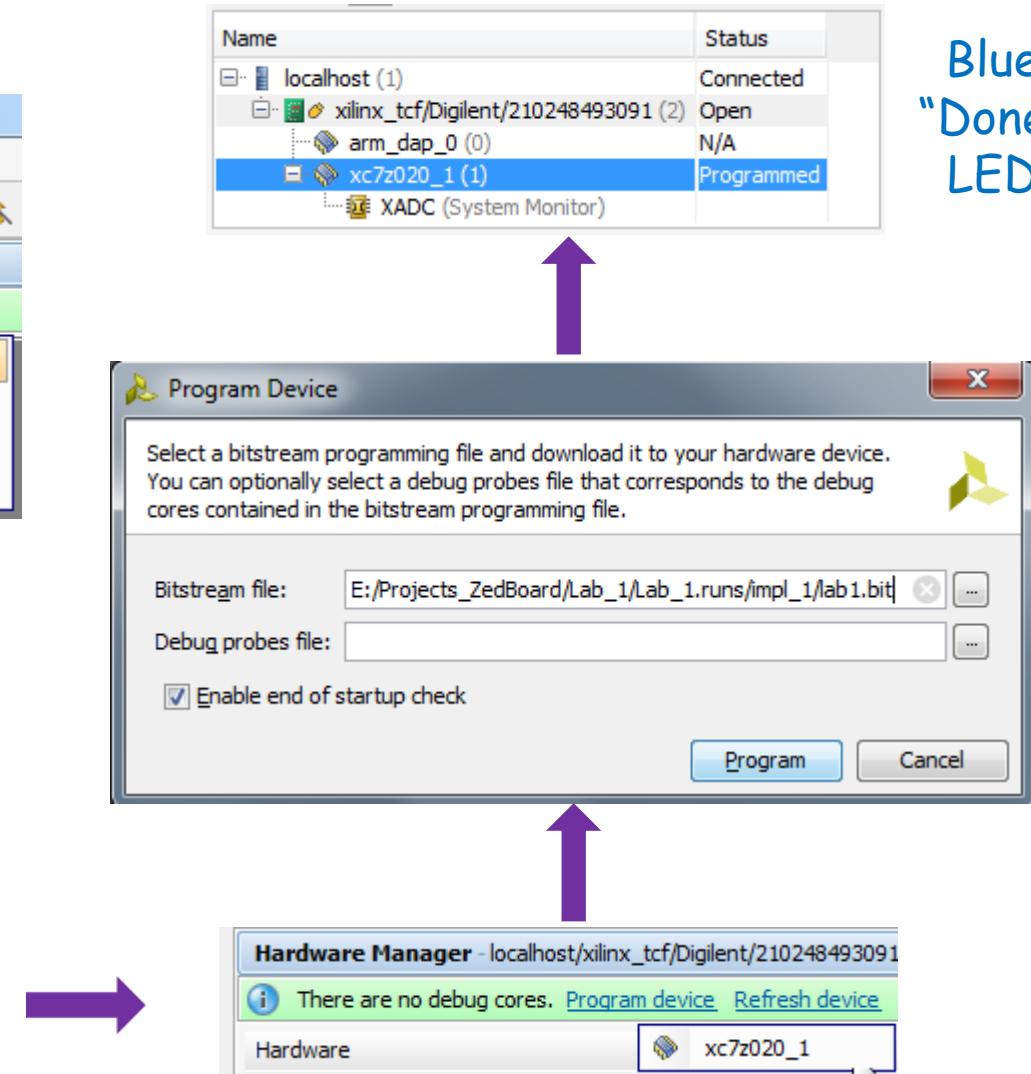
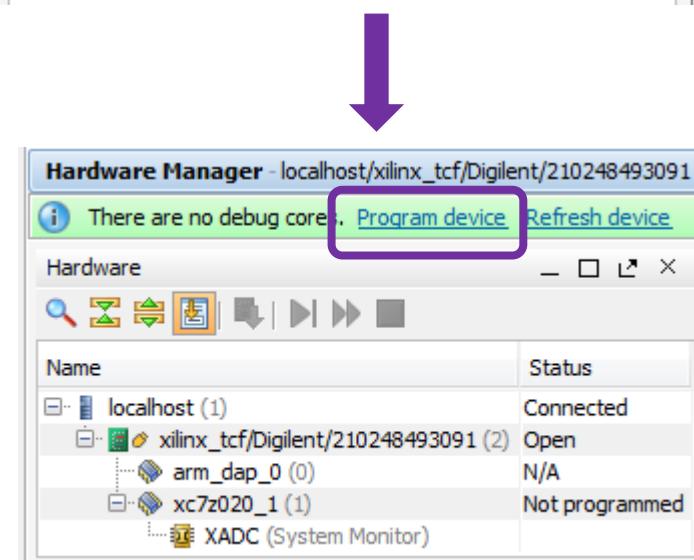
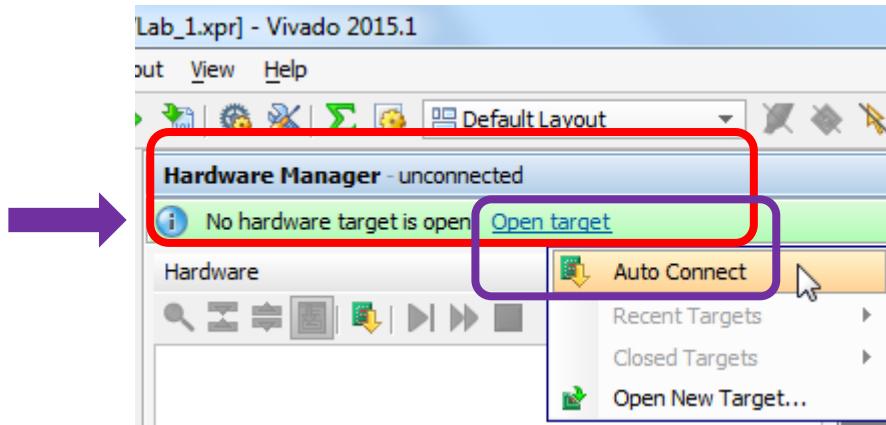
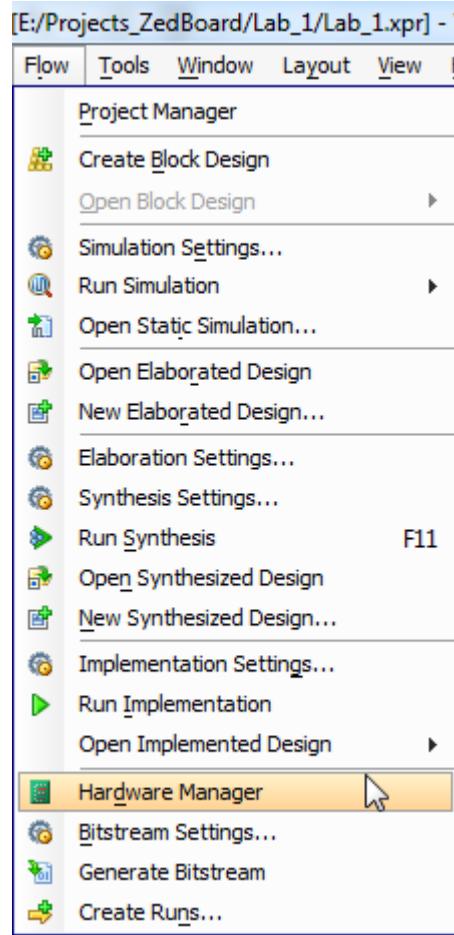
# *Vivado Design Suite*

## *Generate Bit Stream Process*

## *Configuring FPGA Process*

---

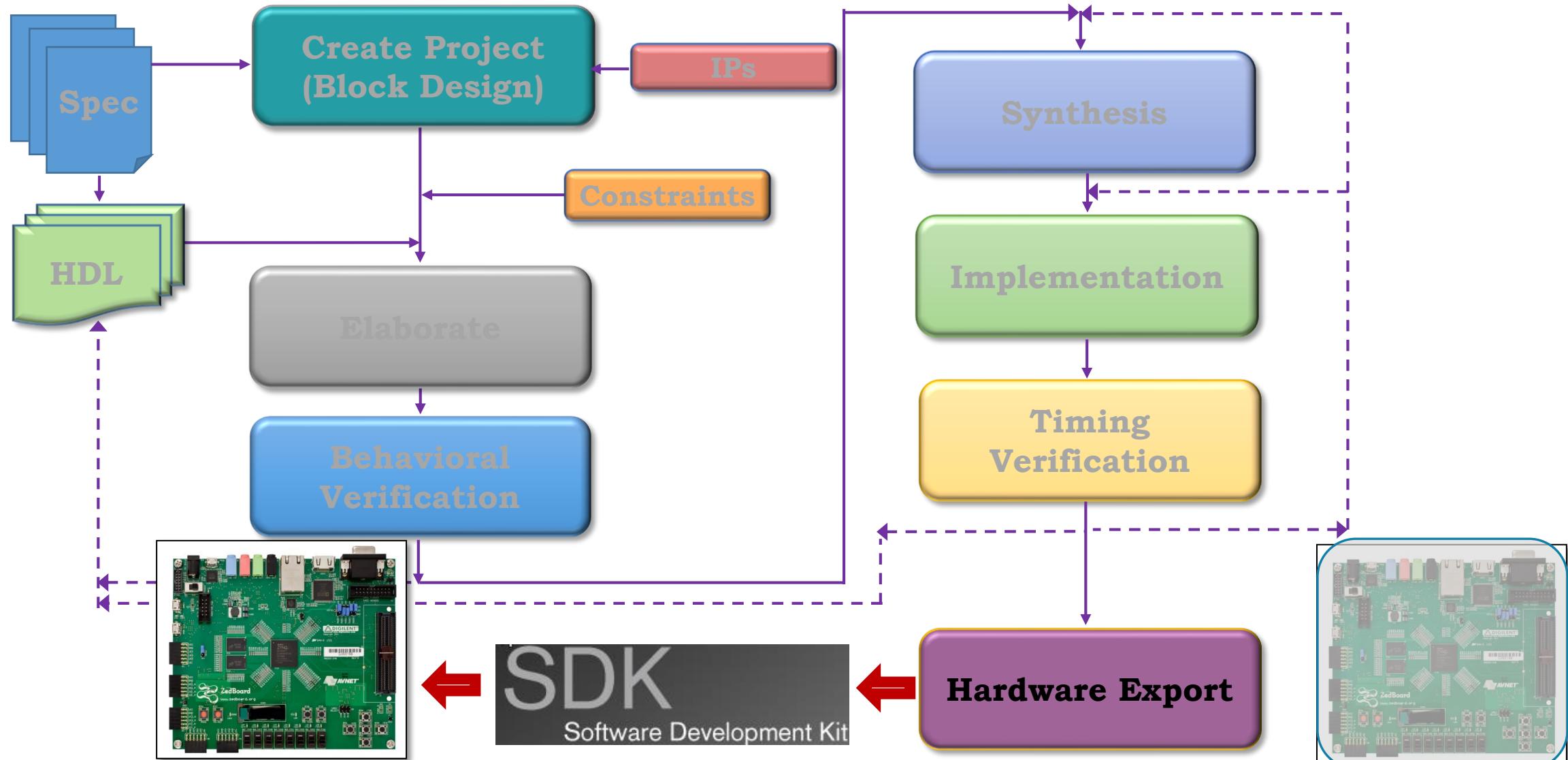
# Steps to Configure *only* the PL



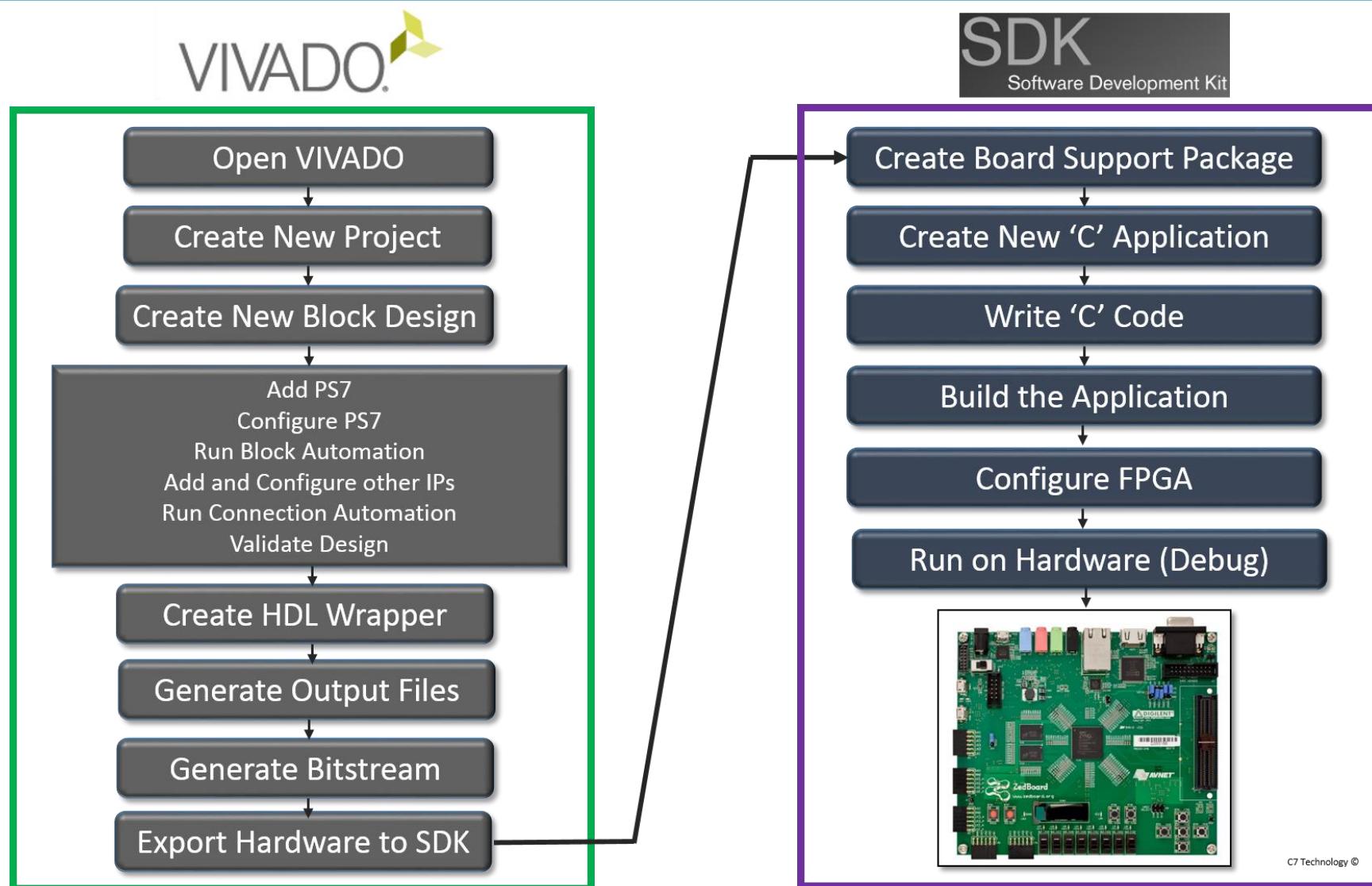
# Software Development Kit (SDK)

---

# Embedded System Design – Vivado-SDK Flow



# Embedded System Design – Vivado-SDK Flow



# Embedded System Tools: Software

---

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

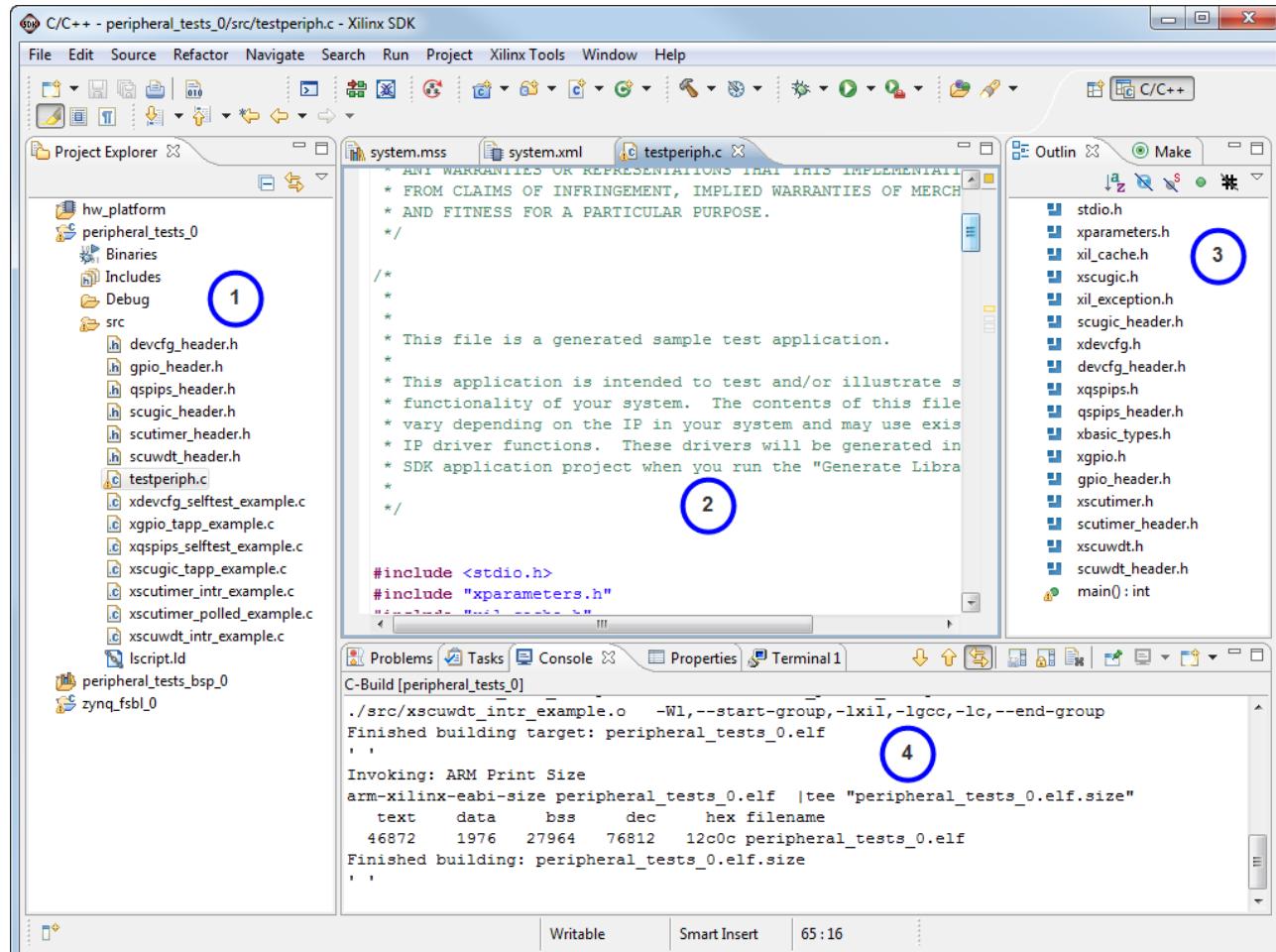
- Board support package creation : LibGen
- GNU software development tools
- C/C++ compiler for the ARM Cortex-A9 processor (gcc)
- Debugger for the ARM Cortex-A9 processor (gdb)

## Board support packages (BSPs)

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

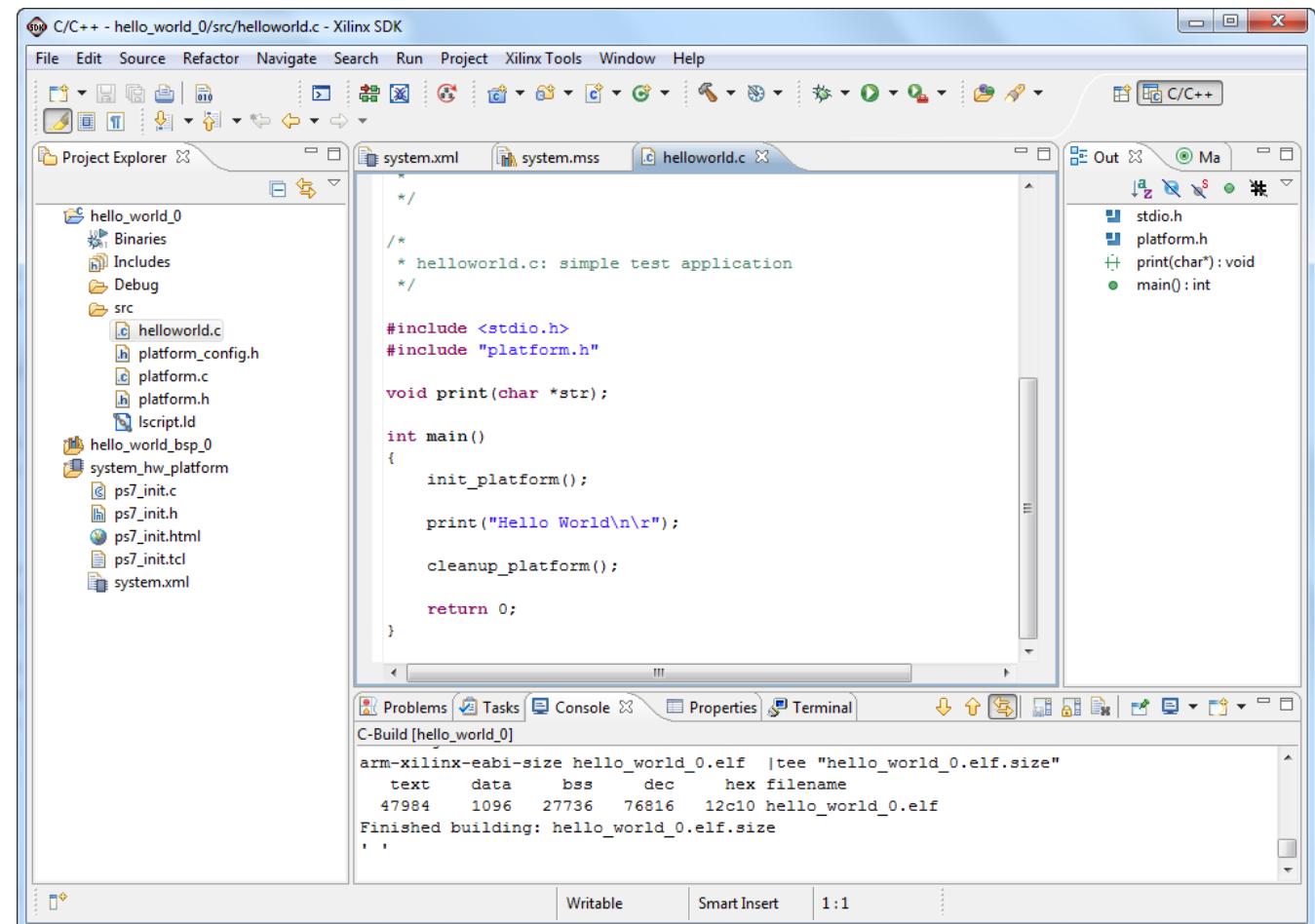
# SDK Workbench Views

- 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



# Build Software Application in SDK

- ❖ Create software platform
  - ❖ System software, board support package
  - ❖ LibGen program
- ❖ Create software application
- ❖ Optionally, create linker script
- ❖ Build project
  - ❖ Compile, assemble, link output file  
*<app\_project>.elf*



The screenshot shows the Xilinx SDK IDE interface. The title bar reads "C/C++ - hello\_world\_0/src/helloworld.c - Xilinx SDK". The left pane is the "Project Explorer" showing the project structure:

- hello\_world\_0
- Binaries
- Includes
- Debug
- src
  - helloworld.c
  - platform\_config.h
  - platform.c
  - platform.h
  - lscript.ld
- hello\_world\_bsp\_0
- system\_hw\_platform
  - ps7\_init.c
  - ps7\_init.h
  - ps7\_init.html
  - ps7\_init.tcl
- system.xml

The main editor window displays the content of helloworld.c:

```
/*
 * helloworld.c: simple test application
 */

#include <stdio.h>
#include "platform.h"

void print(char *str);

int main()
{
    init_platform();

    print("Hello World\n\r");

    cleanup_platform();

    return 0;
}
```

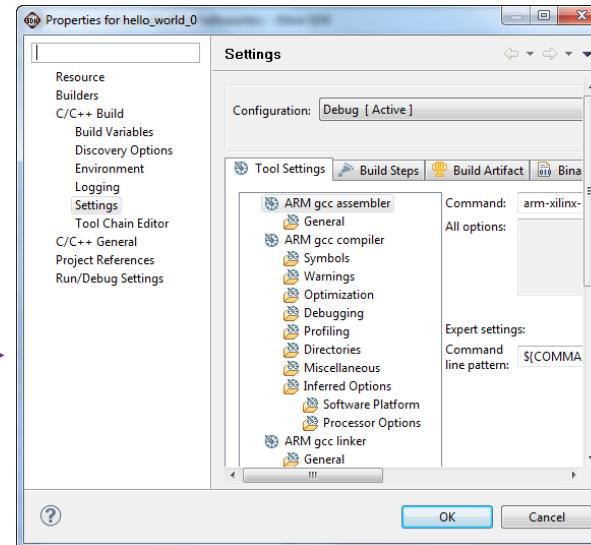
The bottom pane shows the terminal output of the build process:

```
C-Build [hello_world_0]
arm-xilinx-eabi-size hello_world_0.elf | tee "hello_world_0.elf.size"
text      data      bss      dec      hex filename
47984     1096    27736    76816   12c10 hello_world_0.elf
Finished building: hello_world_0.elf.size
'
```

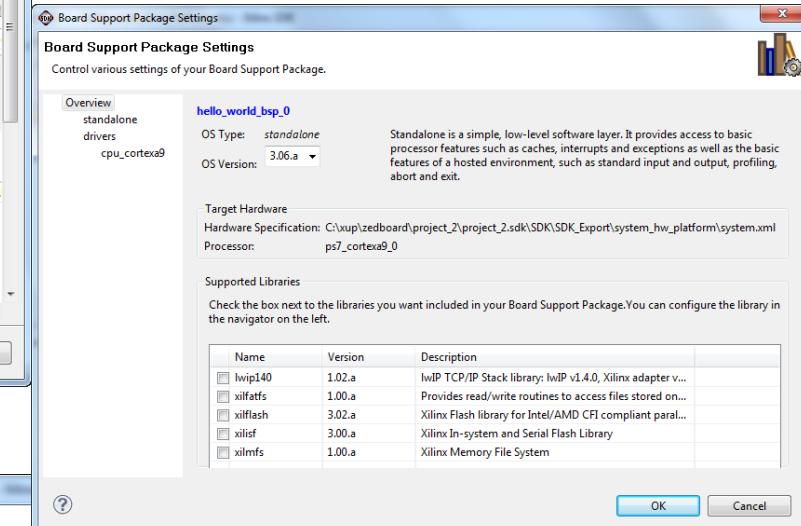
# Software Management Settings

Software is managed in three major areas

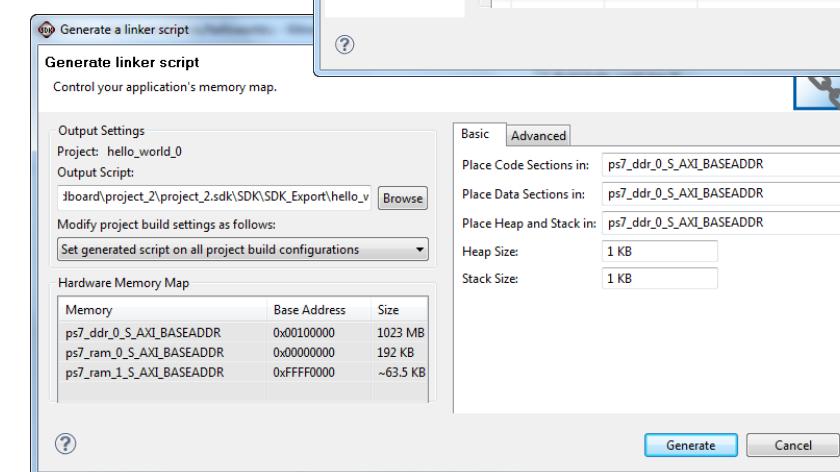
- Compiler/Linker Options
- Application program



- Software Platform Settings
- Board support package



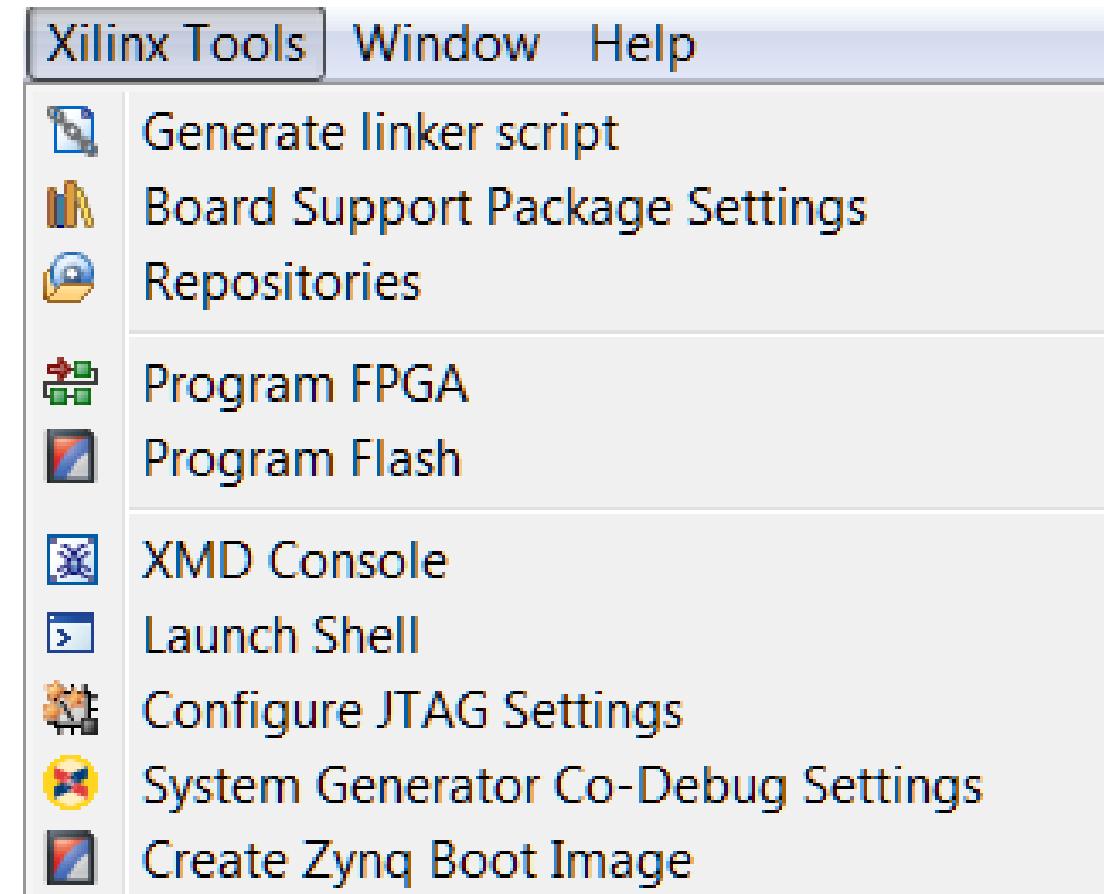
- Linker Script Generation
- Assigning software to memory resources



# 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



# Clocking Resources: MMCM and PLL

Up to 24 CMTs per device

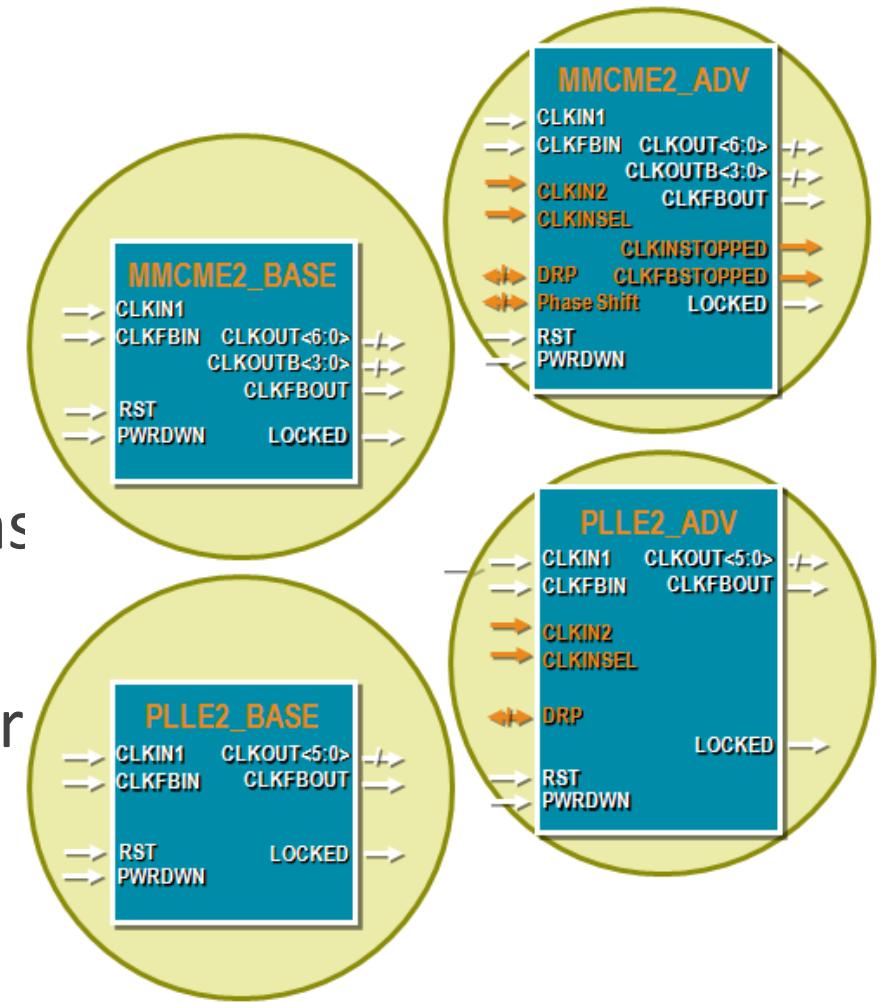
One MMCM and one PLL per CMT

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

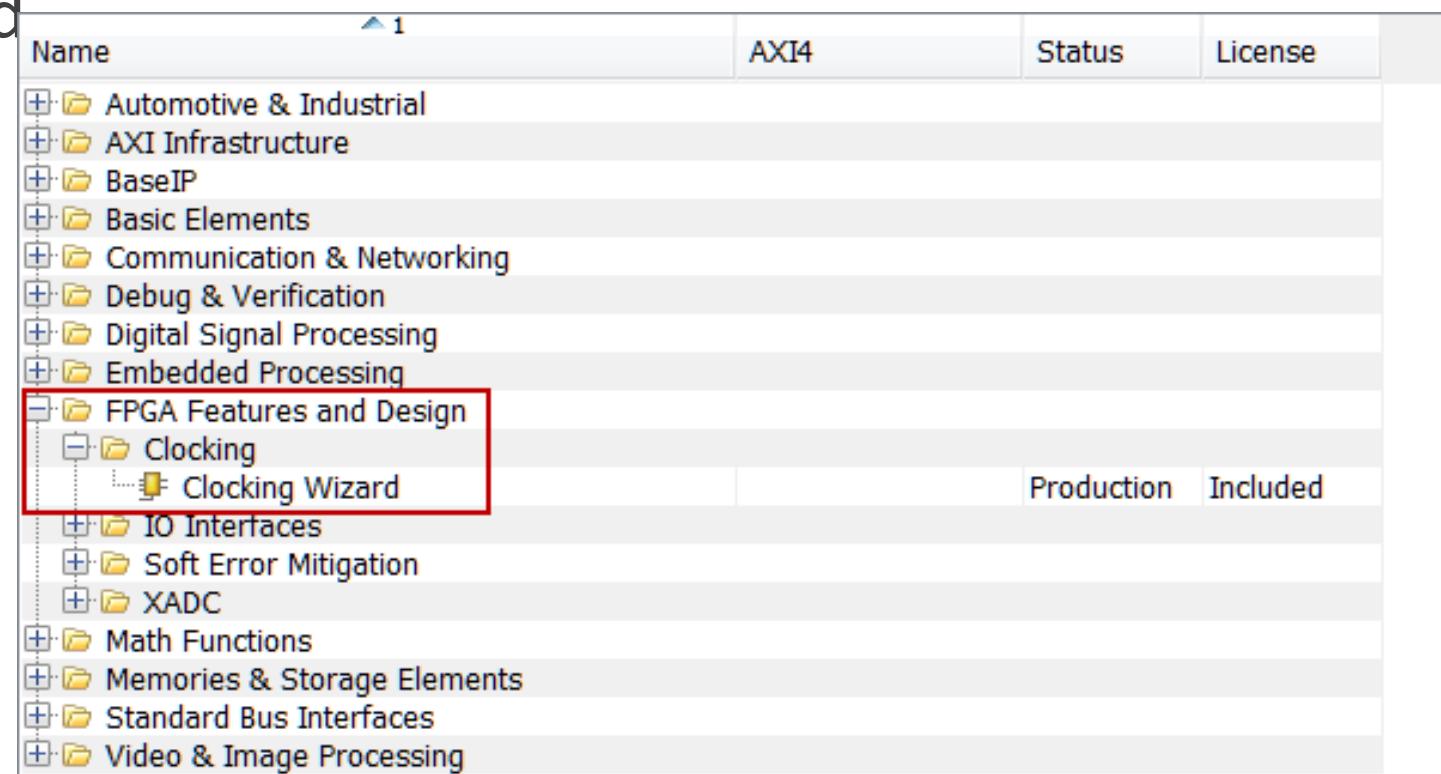
---

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 Clocking Wizard: Clocking Options

Select Primitives to be used

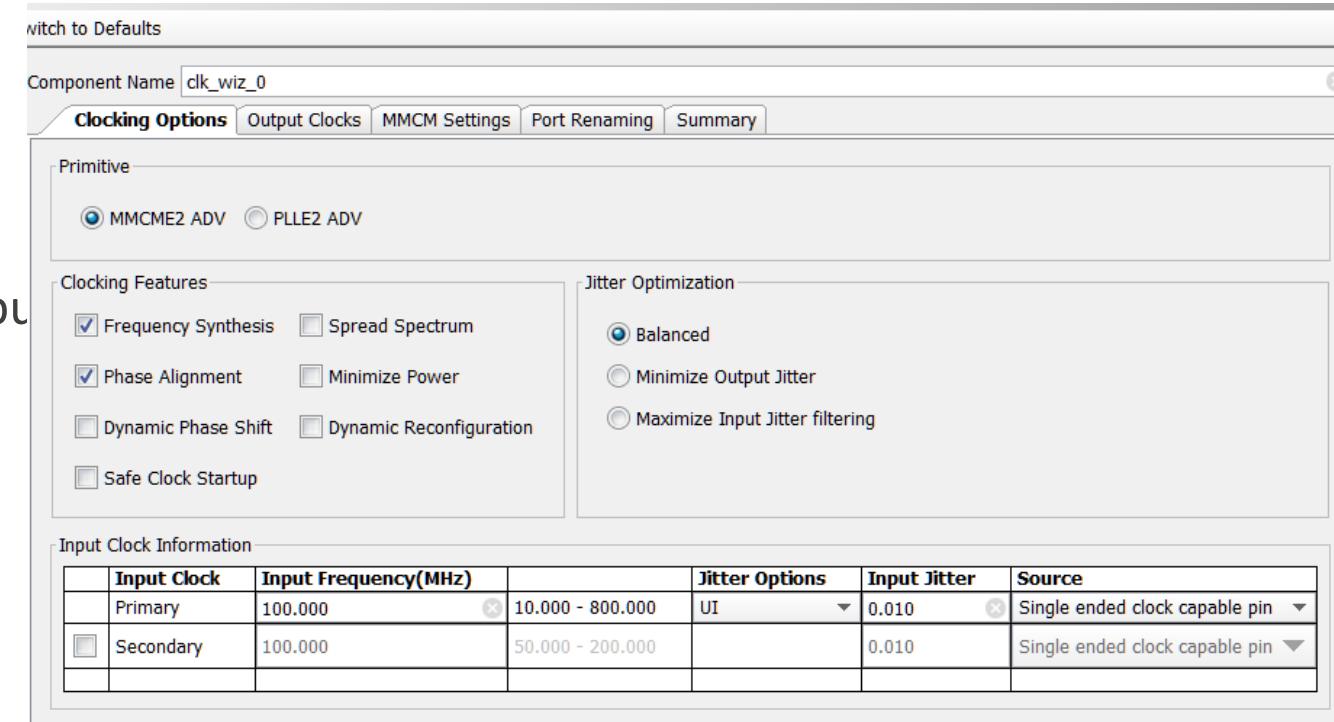
- MMCME2\_ADV
- PLLE2\_ADV

Specify the primary input frequency and source type

- Optionally, select and specify secondary input

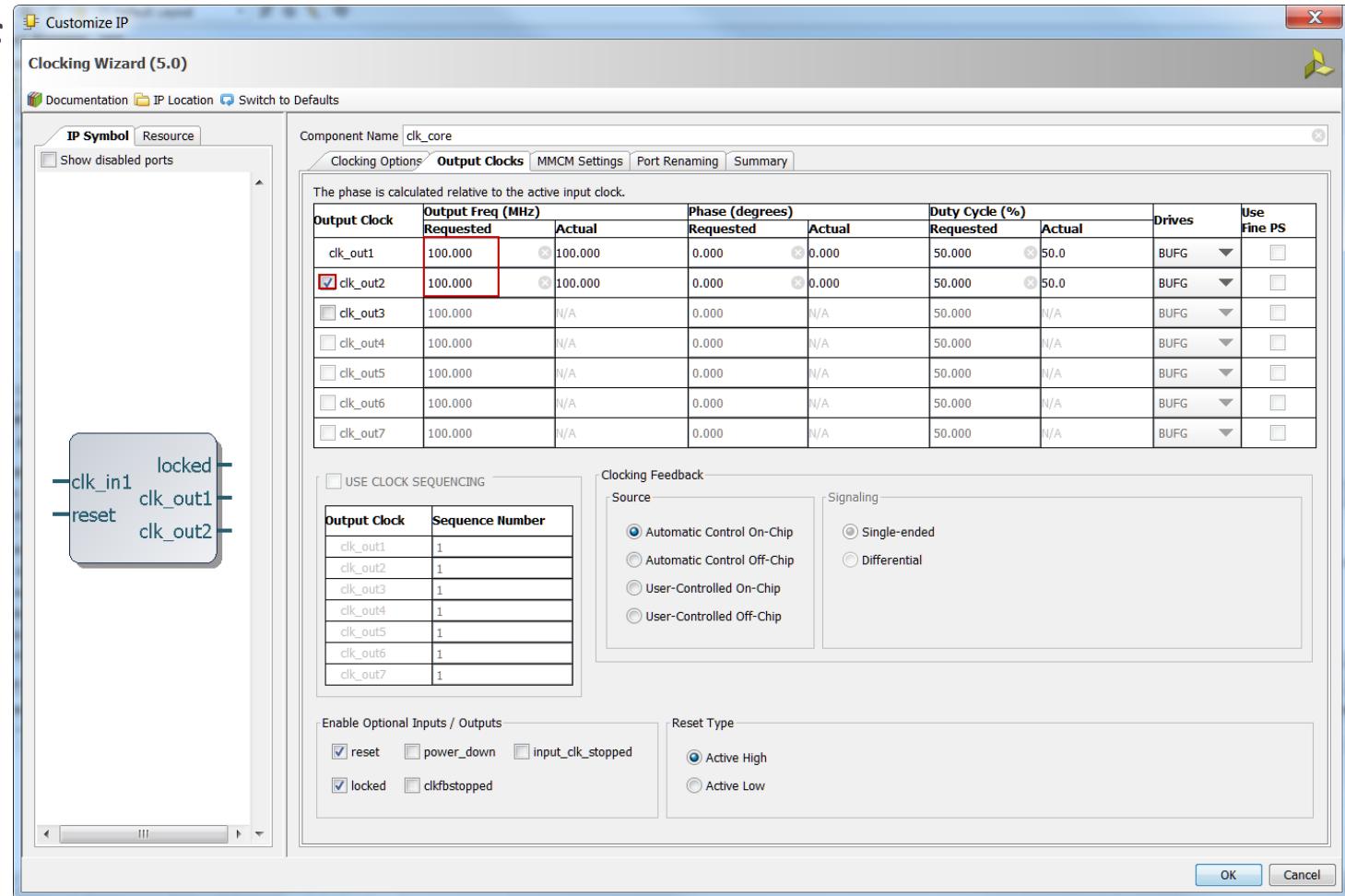
Select clocking features

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



# The Clocking Wizard: Output Clocks

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



# The Clocking Wizard: Port Renaming

Change input/output port names

Change optional port names

The screenshot shows the 'Port Renaming' tab of the Vivado Clocking Wizard. The interface is divided into three main sections: Input Clock, Output Clock, and Optional Port Names, each containing a table for renaming.

**Input Clock:**

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

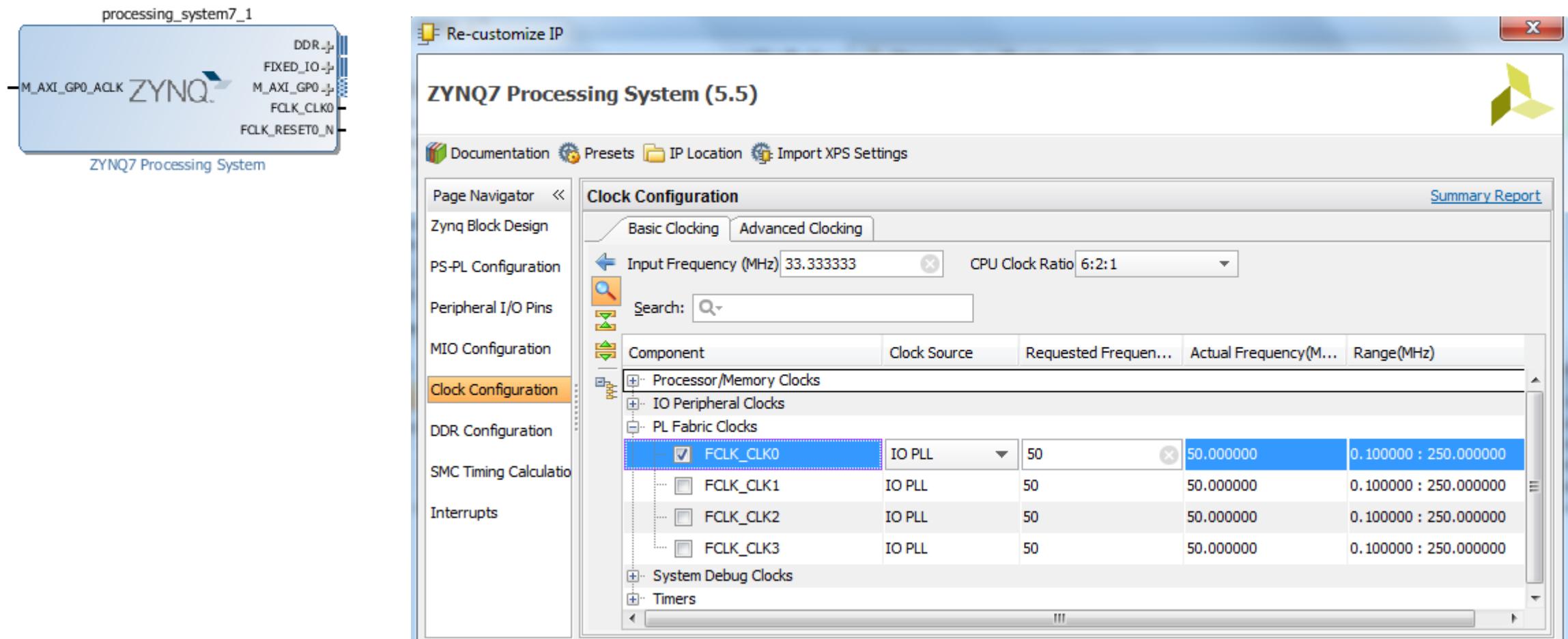
Attribute	Value
Input Clock (MHz)	100.000
Phase Shift	None
Divide Counter	1
Mult Counter	10.000
CLKOUT0 Divider	10.000
CLKOUT1 Divider	10
CLKOUT2 Divider	OFF
CLKOUT3 Divider	OFF
CLKOUT4 Divider	OFF
CLKOUT5 Divider	OFF
CLKOUT6 Divider	OFF

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

PS-PL Configuration

Search:

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/arbit 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 di
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

---

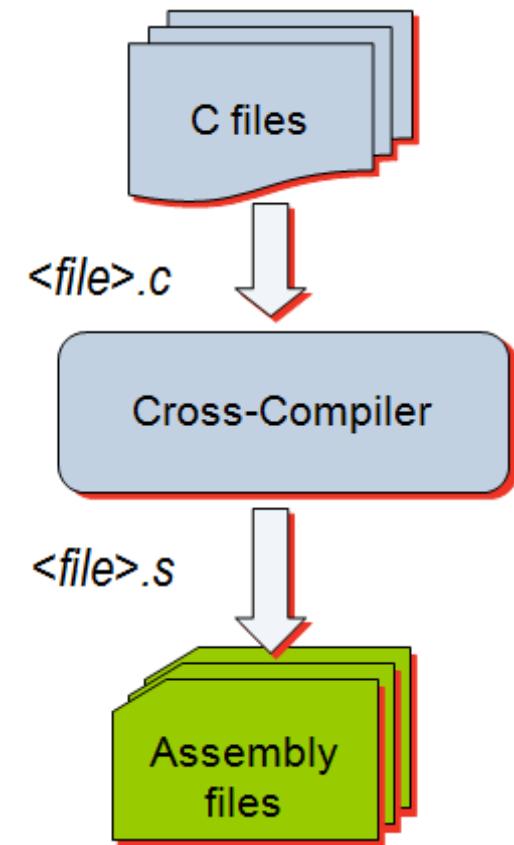
GCC translates C source code into assembly language

GCC also functions as the user interface, passing options to the 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

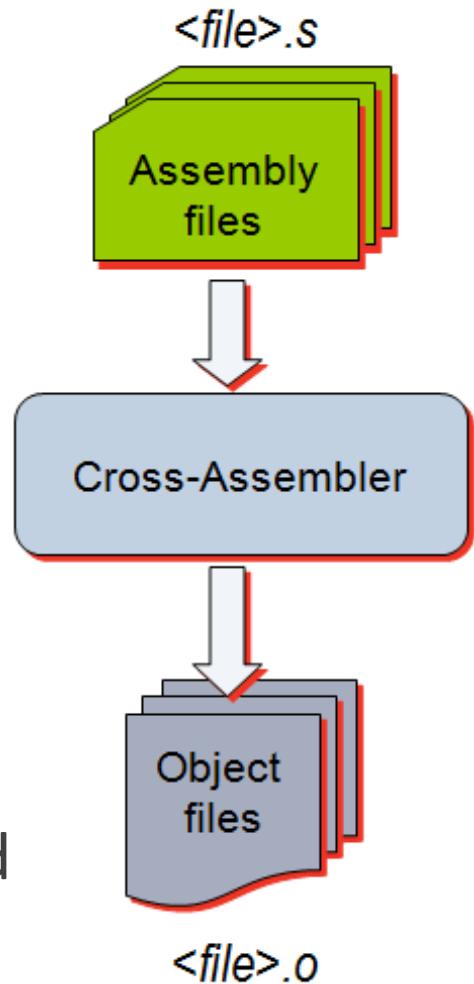
- File extension: .o

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



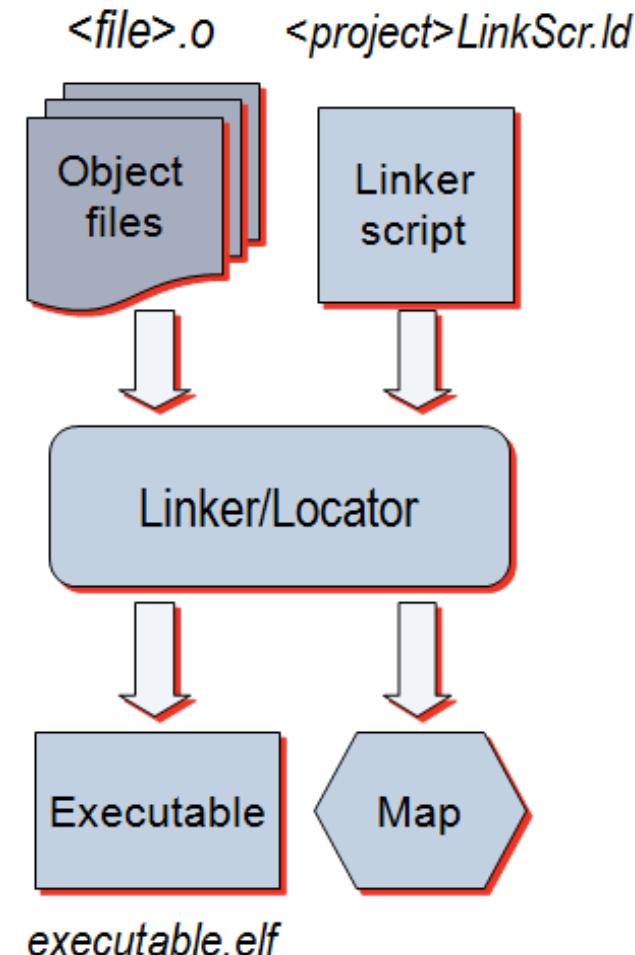
# GNU Tools: Linker (LD)

## 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

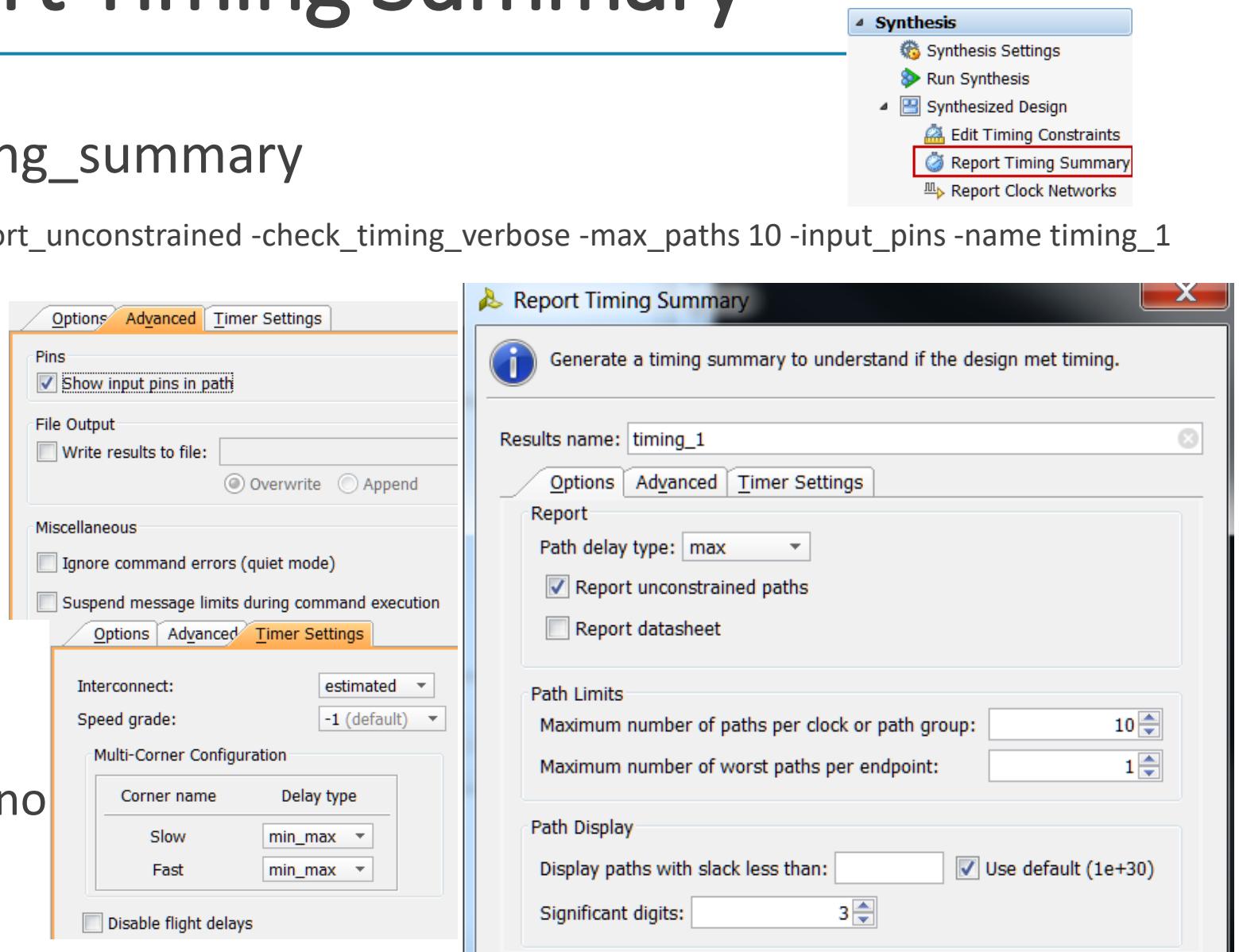
- Maximum number of paths

Advanced tab

- Write to a file

Timer Settings

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



# Report Timing Summary

## Design Timing Summary

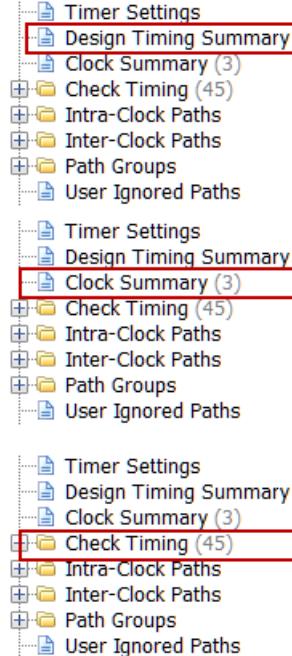
- WNS, TNS, total number of endpoints are of interest

## Clock Summary

- Primary and derived clocks

## Check Timing

- Number of unconstrained internal endpoints



The screenshot displays three summary tables from the Vivado Timing Analyzer:

- Design Timing Summary:**

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
- Clock Summary:**

Name	Waveform	Period (ns)	Frequency (MHz)
clkin	{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
- Check Timing:**

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