

Advanced Workshop  
on Modern FPGA-  
Based Technology for  
Scientific 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

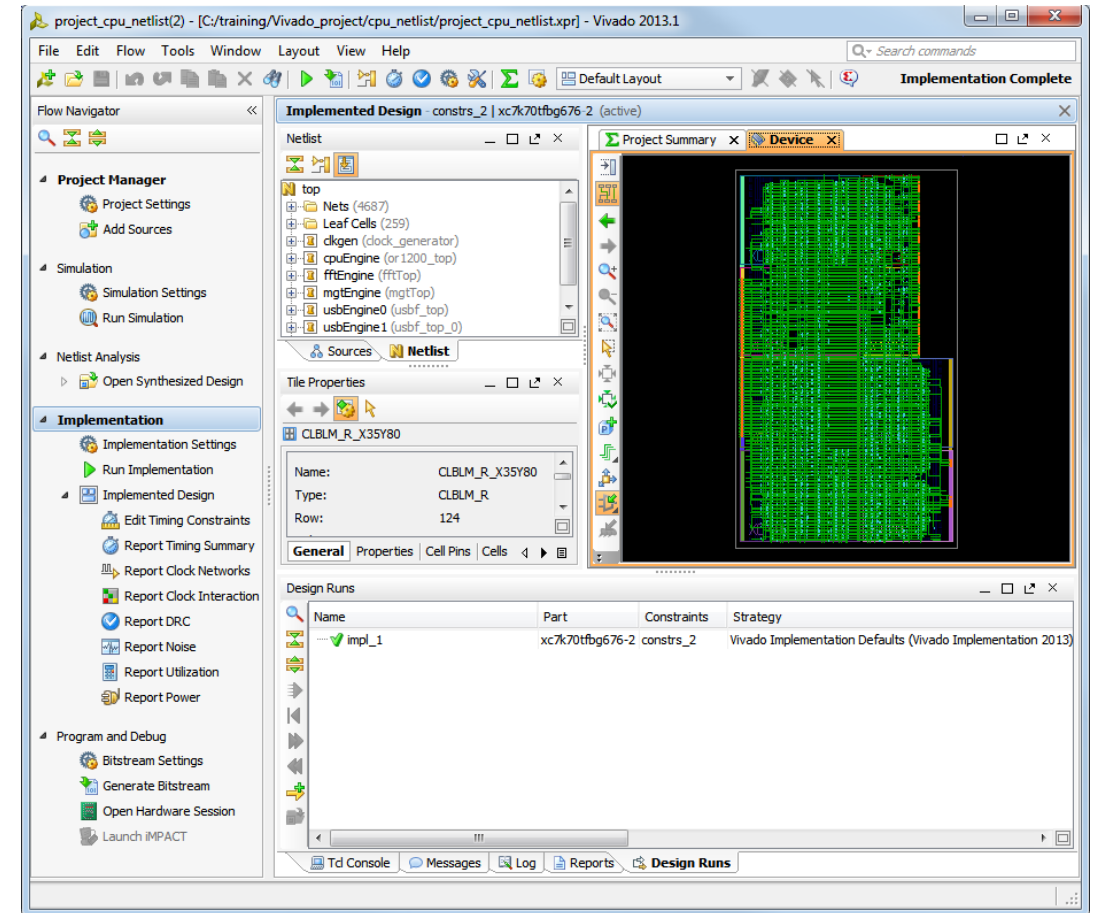
- Elaboration of HDL
- Hierarchical exploration
- Schematic generation

## XSIM simulator integration

- Synthesis, implementation and simulation in one package

## I/O pin planning

- Interactive rule-based I/O assignment

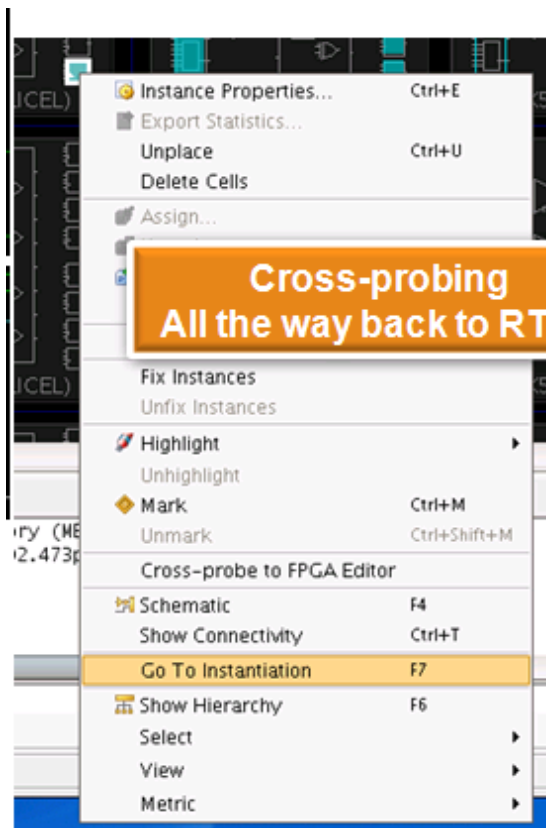


*Hierarchical Design Analysis and Implementation Environment*

# Vivado Visualization Features

Visualize and debug a design at any flow stage

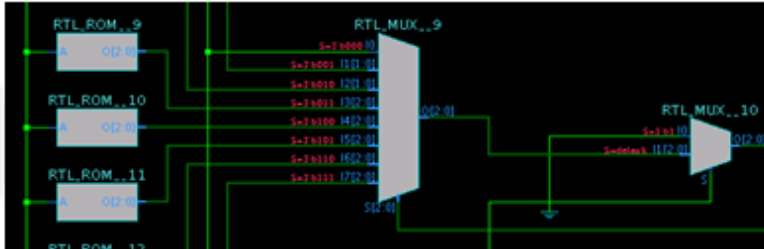
- Cross-probing between netlist/schematic/RTL



The screenshot shows the Vivado Instance Properties menu. The 'Go To Instantiation' option is highlighted in orange, with the keyboard shortcut F7. Other options include Instance Properties... (Ctrl+E), Export Statistics..., Unplace (Ctrl+U), Delete Cells, Assign..., Fix Instances, Unfix Instances, Highlight, Unhighlight, Mark (Ctrl+M), Unmark (Ctrl+Shift+M), Schematic (F4), Show Connectivity (Ctrl+T), Show Hierarchy (F6), Select, View, and Metric.

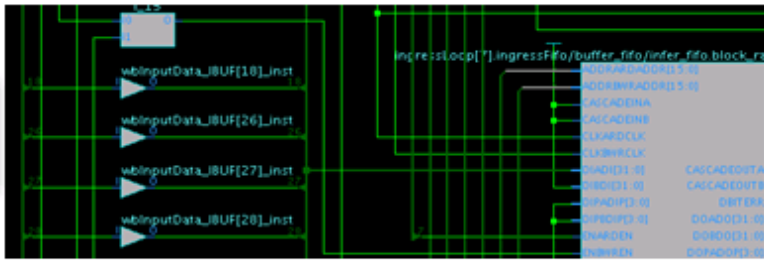
**Cross-probing  
All the way back to RTL!!**

**Elaborated  
RTL**




The Elaborated RTL schematic shows a network of components including RTL\_ROM\_9 through RTL\_ROM\_17 and RTL\_MUX\_9 through RTL\_MUX\_10. Signal lines connect these components, with some labeled with values like 'S=70000 01'.

**Netlist  
Schematic**



The Netlist Schematic shows a hierarchical view of the design. On the left, there are several 'wbinInputData\_IBUF' instances. On the right, there is a large block labeled 'incre:sl0cp' with various internal components like 'ADDR#ADDR0 5 01', 'ADDR#ADDR1 5 01', 'CASCADINA', 'CASCADNB', 'CLEARCLK', 'CLEARCLK', 'DINDEN1 01', 'DINDEN2 01', 'DINDEN3 01', 'DINDEN4 01', 'DINDEN5 01', 'DINDEN6 01', 'DINDEN7 01', 'DINDEN8 01', 'DINDEN9 01', 'DINDEN10 01', 'DINDEN11 01', 'DINDEN12 01', 'DINDEN13 01', 'DINDEN14 01', 'DINDEN15 01', 'DINDEN16 01', 'DINDEN17 01', 'DINDEN18 01', 'DINDEN19 01', 'DINDEN20 01', 'DINDEN21 01', 'DINDEN22 01', 'DINDEN23 01', 'DINDEN24 01', 'DINDEN25 01', 'DINDEN26 01', 'DINDEN27 01', 'DINDEN28 01', 'DINDEN29 01', 'DINDEN30 01', 'DINDEN31 01', 'DINDEN32 01', 'DINDEN33 01', 'DINDEN34 01', 'DINDEN35 01', 'DINDEN36 01', 'DINDEN37 01', 'DINDEN38 01', 'DINDEN39 01', 'DINDEN40 01', 'DINDEN41 01', 'DINDEN42 01', 'DINDEN43 01', 'DINDEN44 01', 'DINDEN45 01', 'DINDEN46 01', 'DINDEN47 01', 'DINDEN48 01', 'DINDEN49 01', 'DINDEN50 01', 'DINDEN51 01', 'DINDEN52 01', 'DINDEN53 01', 'DINDEN54 01', 'DINDEN55 01', 'DINDEN56 01', 'DINDEN57 01', 'DINDEN58 01', 'DINDEN59 01', 'DINDEN60 01', 'DINDEN61 01', 'DINDEN62 01', 'DINDEN63 01', 'DINDEN64 01', 'DINDEN65 01', 'DINDEN66 01', 'DINDEN67 01', 'DINDEN68 01', 'DINDEN69 01', 'DINDEN70 01', 'DINDEN71 01', 'DINDEN72 01', 'DINDEN73 01', 'DINDEN74 01', 'DINDEN75 01', 'DINDEN76 01', 'DINDEN77 01', 'DINDEN78 01', 'DINDEN79 01', 'DINDEN80 01', 'DINDEN81 01', 'DINDEN82 01', 'DINDEN83 01', 'DINDEN84 01', 'DINDEN85 01', 'DINDEN86 01', 'DINDEN87 01', 'DINDEN88 01', 'DINDEN89 01', 'DINDEN90 01', 'DINDEN91 01', 'DINDEN92 01', 'DINDEN93 01', 'DINDEN94 01', 'DINDEN95 01', 'DINDEN96 01', 'DINDEN97 01', 'DINDEN98 01', 'DINDEN99 01', 'DINDEN100 01'.

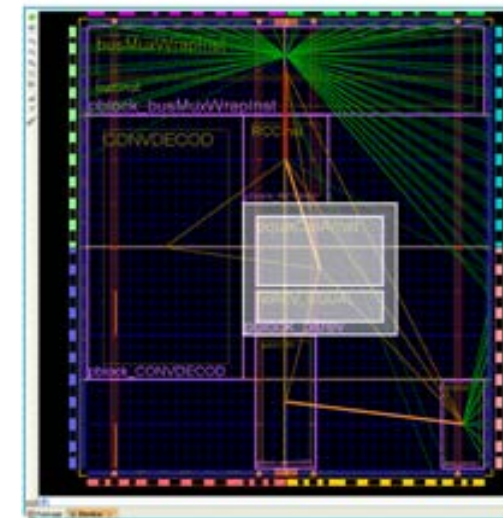
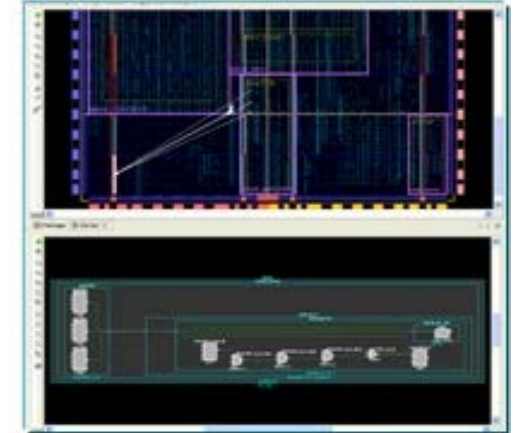
**Implemented  
Design**



The Implemented Design shows the physical layout of the design on the FPGA. Two slices are highlighted: SLICE\_X6Y81 (SLICEM) and SLICE\_X7Y81 (SLICEL). The slices are shown with their internal logic resources and connections.

# Gain Faster Timing Closure

- **Analyze multiple implementation results**
  - Highlight failing timing paths from post-route timing analysis
  - 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

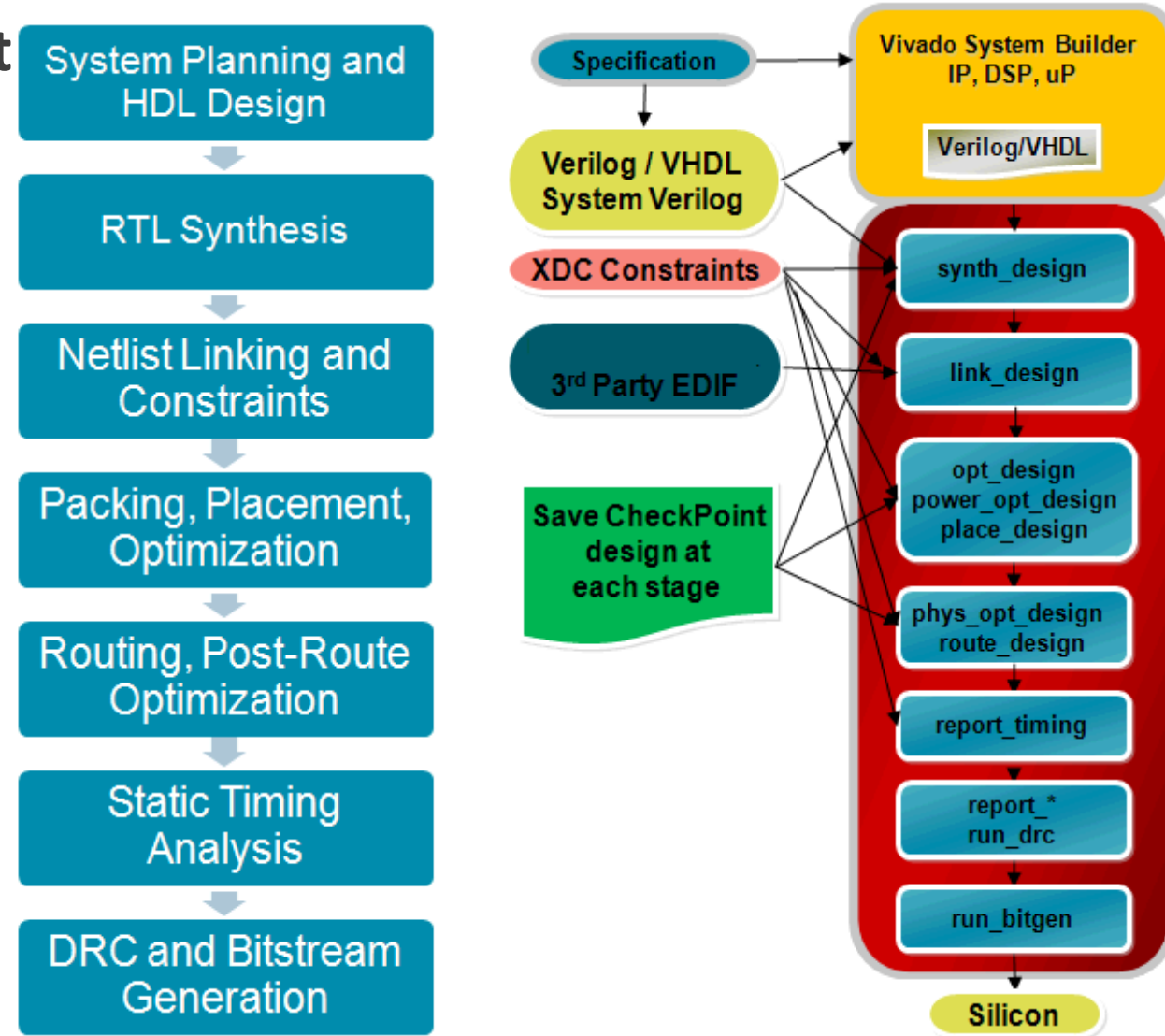
- ✓ **Common data model throughout the flow**

- ✓ “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 (synthesis, implementation)
- ***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

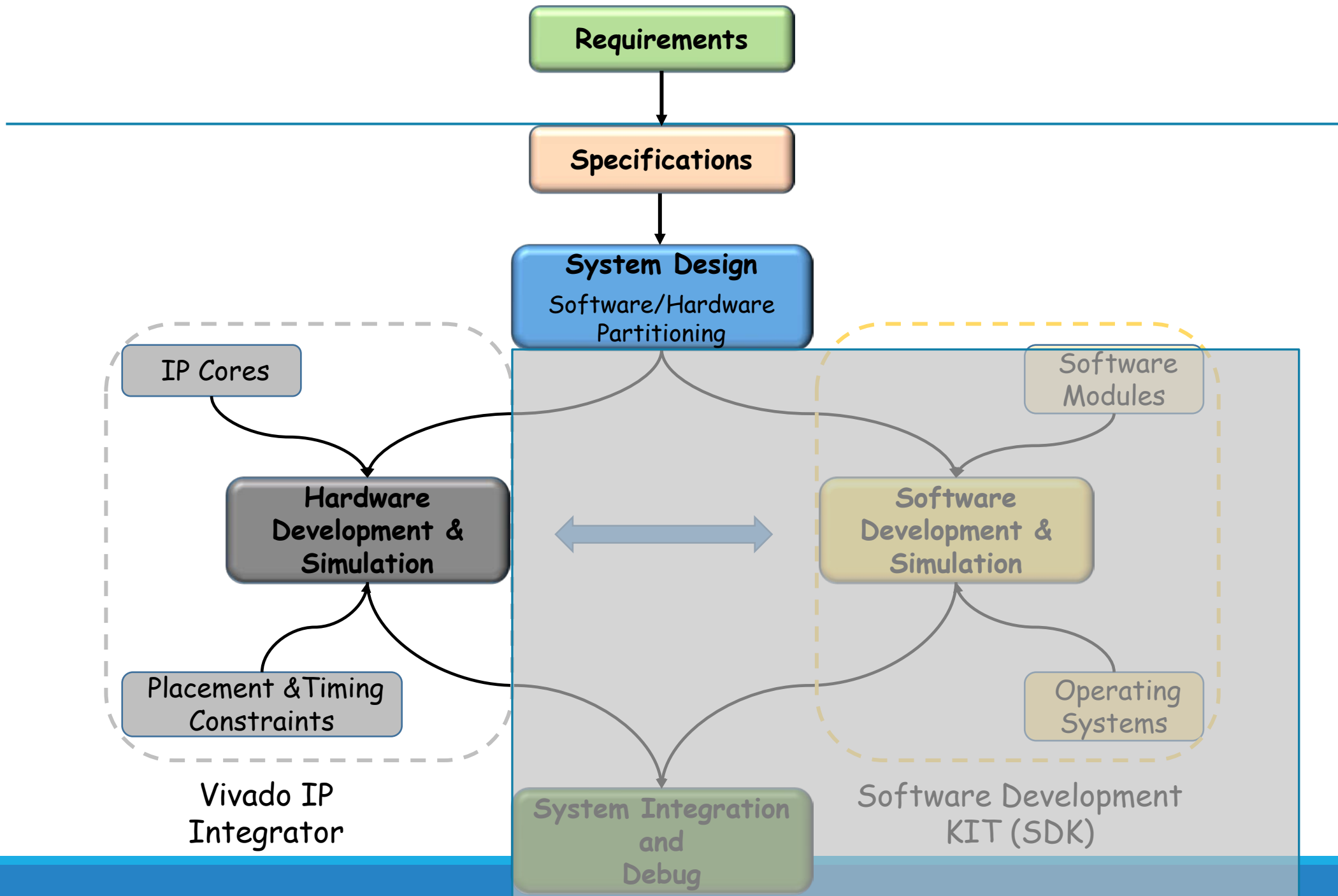
## Log file (*vivado.log*)

- 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`

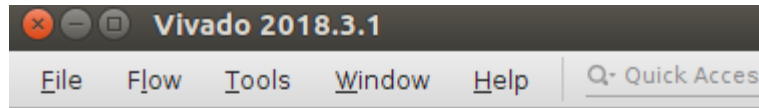




# *Vivado Flow Practical Steps*

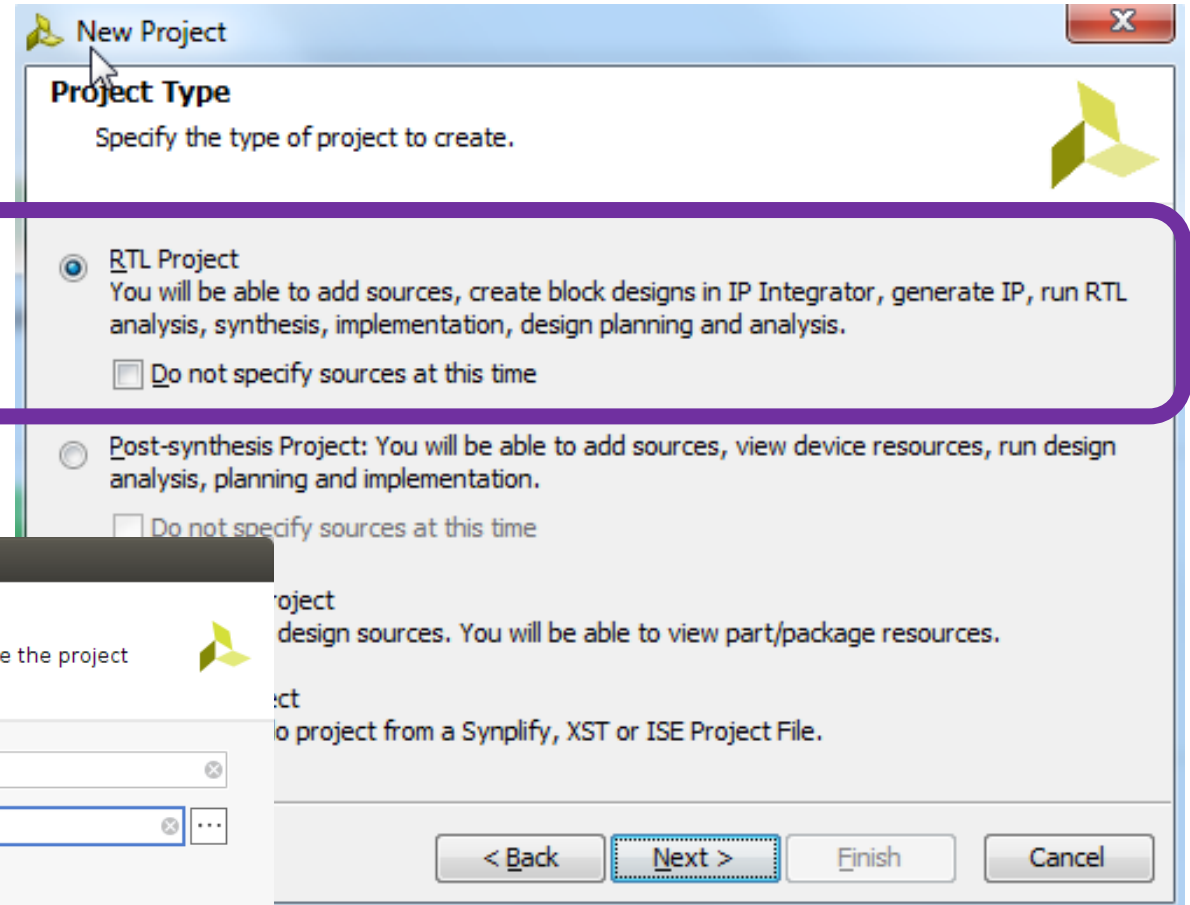
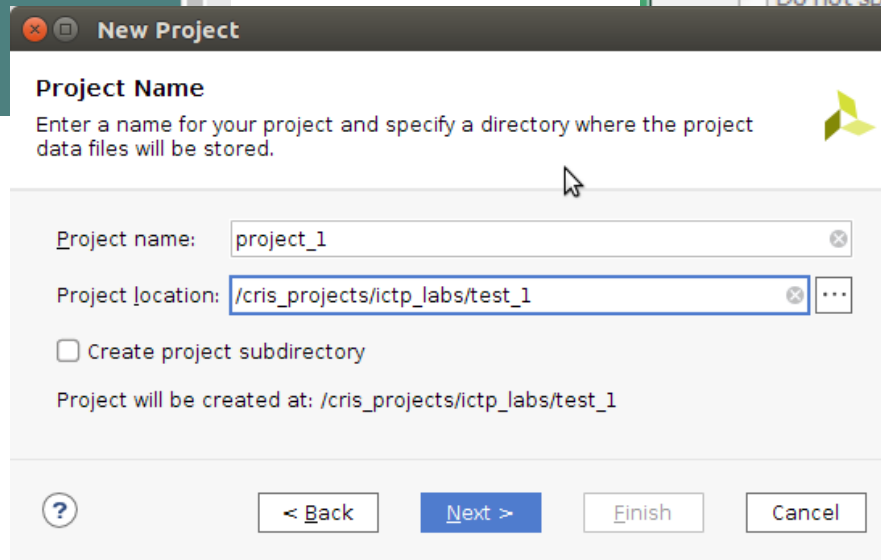
---

# Creating a Project






Quick Start

Create Project



# Creating a Project

The screenshot shows the 'New Project' dialog box in Vivado. The 'Boards' tab is selected and highlighted with a purple box. Below the tab, there are filters for Vendor (All) and Name (All). A search bar is present. A table of boards is displayed, with the 'ZedBoard Zynq Evaluation and Development Kit' row highlighted in blue and circled in red. The table has columns for Display Name, Preview, Vendor, and File Version. At the bottom, there are navigation buttons: Back, Next, Finish, and Cancel.

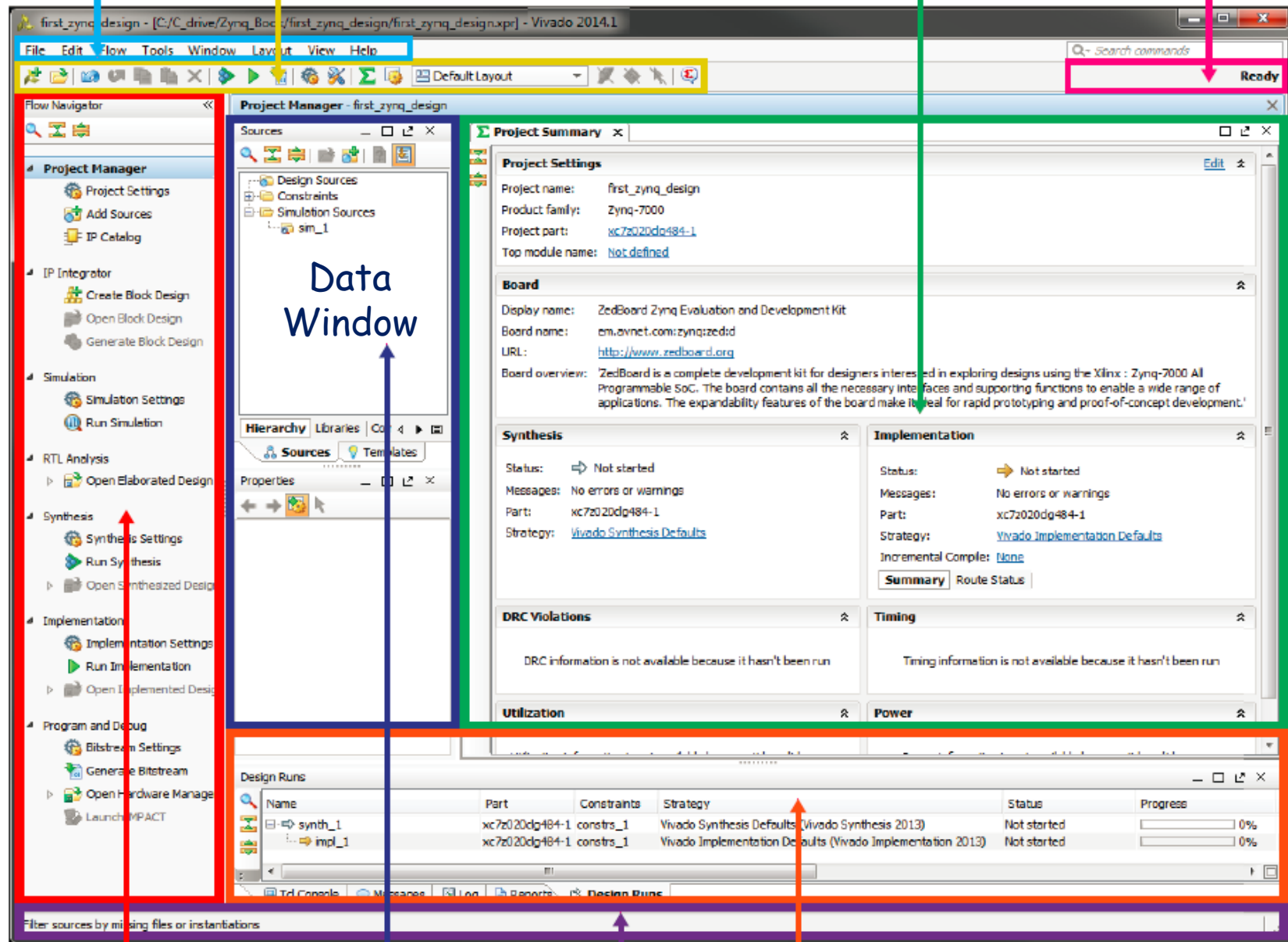
Display Name	Preview	Vendor	File Versic
Alpha-Data ADM-PCIE-7V3		alpha-data.com	1.1
ZedBoard Zynq Evaluation and Development Kit Add Daughter Card Connections		em.avnet.com	1.4
Artix-7 AC701 Evaluation Platform Add Daughter Card Connections		xilinx.com	1.4

Menu Bar 1

Main Toolbar 2

Workspace 3

Status Bar 4



Flow Navigator 5

Status Bar 7

Results Window Area 8

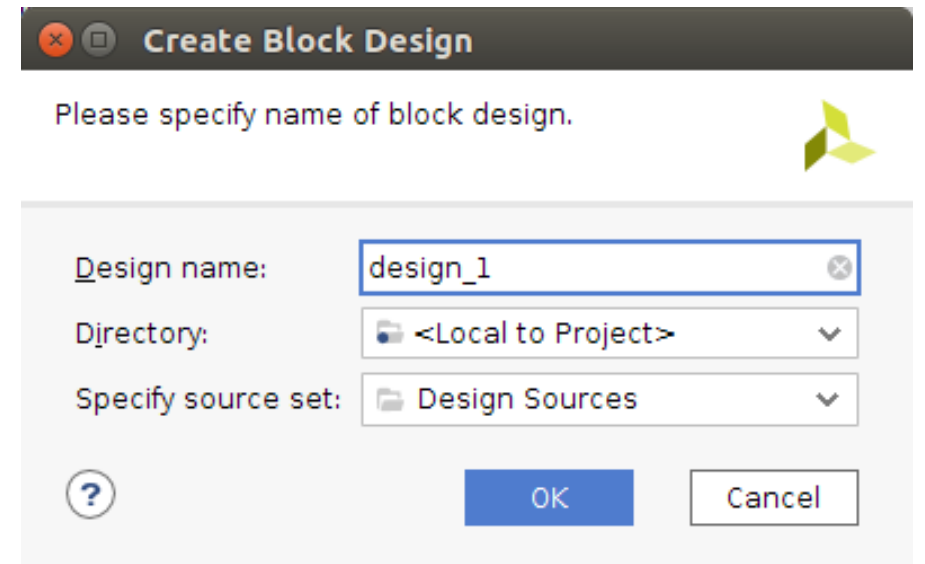
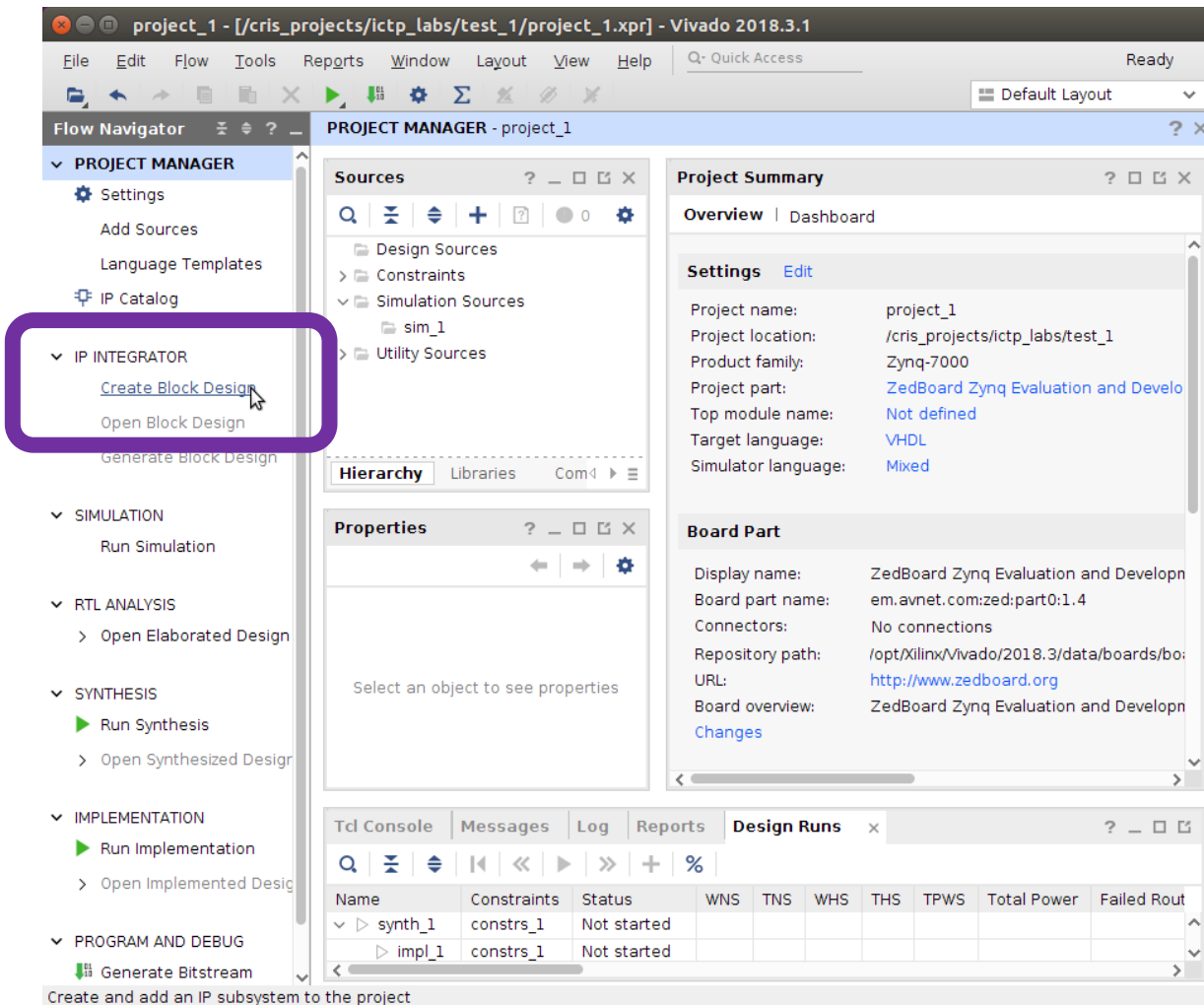


# Main Components of the Project Navigator

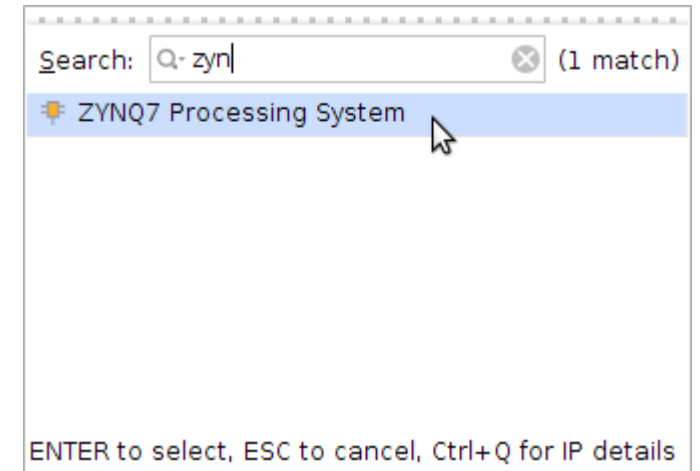
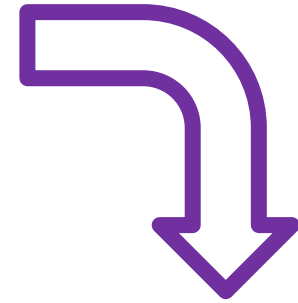
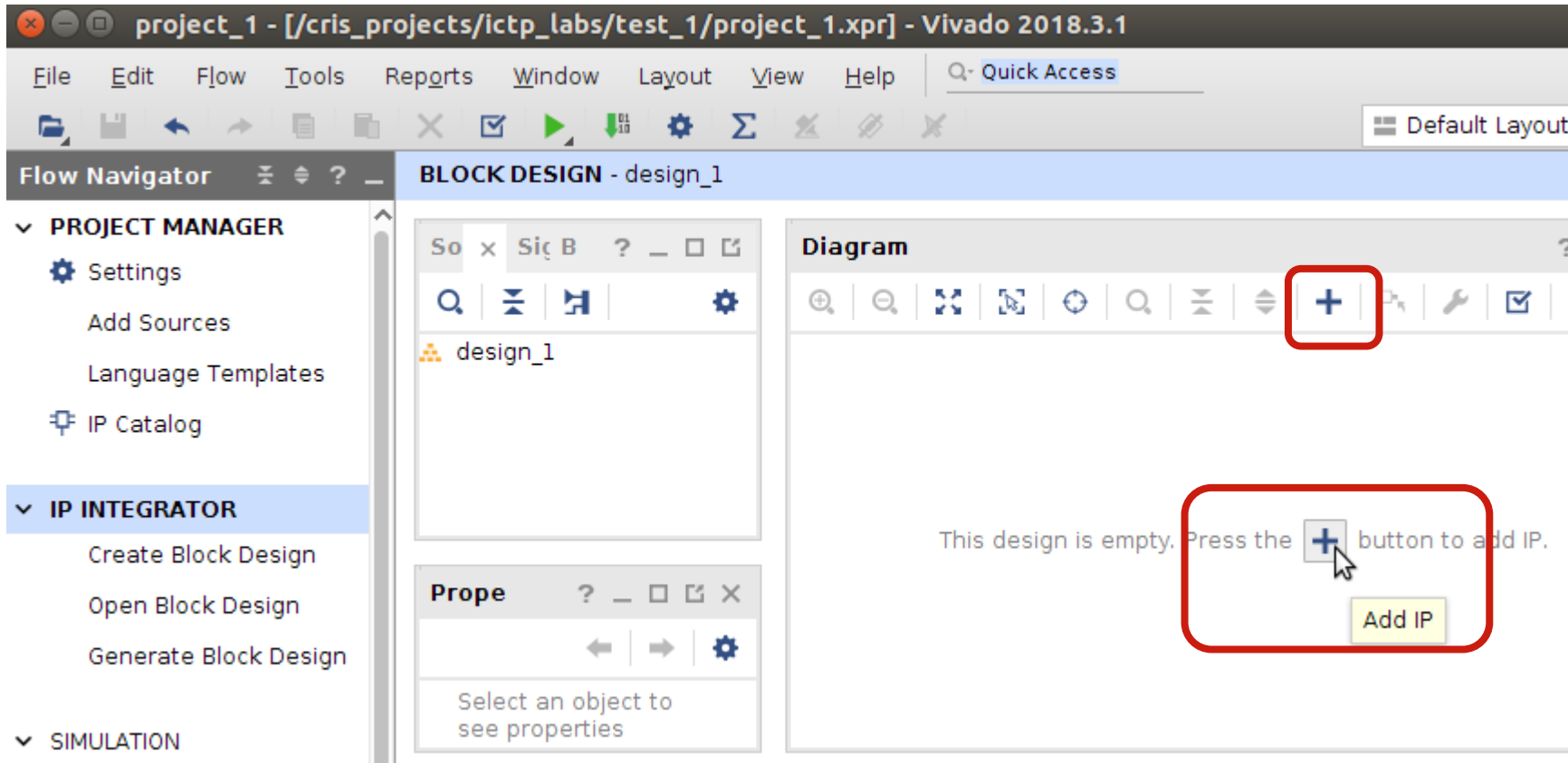
---

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, Design Runs, display the different run for the current project

# Create a Block design



# Adding IP Modules to the Design Canvas

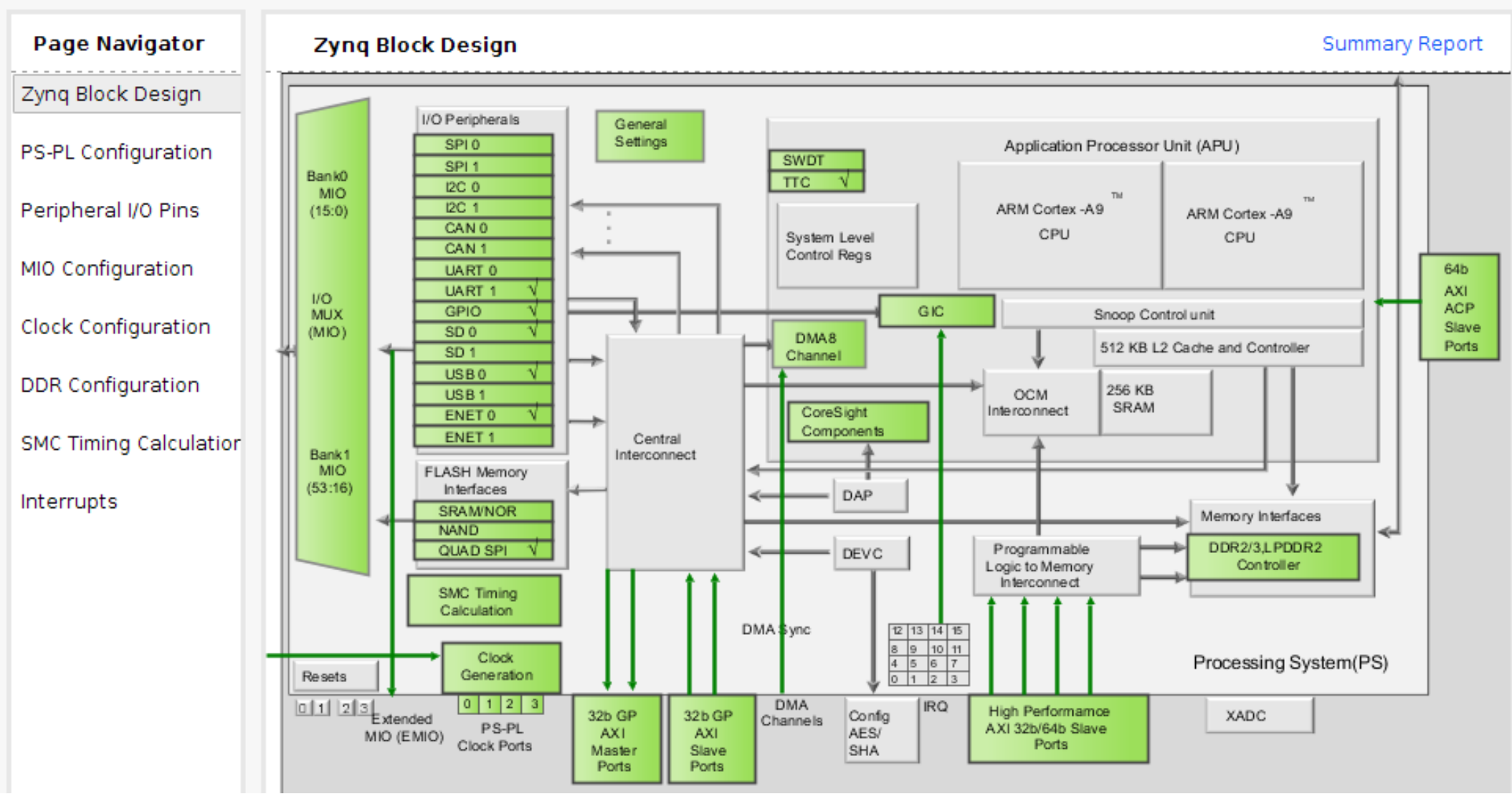


To add multiple IP to the Block Design, you can highlight the additional desired IP (**Ctrl+Click**) and press the **Enter** key.

# Processing System (PS)

The image shows a screenshot of the Vivado IDE interface. The top window is titled "Diagram" and "Address Editor". A toolbar is visible with various icons. A green banner at the top of the diagram window says "Designer Assistance available. Run Block Automation". A red box highlights the "Run Block Automation" link in the banner. Below the banner is a diagram of a "processing\_system7\_0" block, which is a "ZYNQ7 Processing System". The diagram shows the "ZYNQ7" logo and several ports: "M\_AXI\_GP0\_ACLK" on the left, and "DDR", "FIXED\_IO", "M\_AXI\_GP0", "FCLK\_CLK0", and "FCLK\_RESET0\_N" on the right. A "Run Block Automation" dialog box is open in the foreground. The dialog has a title bar "Run Block Automation" and a description: "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." The dialog is divided into two main sections. The left section shows a list of automation options, with "All Automation (1 out of 1 selected)" and "processing\_system7\_0" checked. The right section is titled "Description" and contains the following text: "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". Below the description is the "Options" section, which includes: "Make Interface External: FIXED IO, DDR", "Apply Board Preset: ", "Cross Trigger In: Disable", and "Cross Trigger Out: Disable". A red box highlights the "Apply Board Preset" checkbox. At the bottom right of the dialog are "OK" and "Cancel" buttons. A mouse cursor is pointing at the "OK" button.

# Configuring the PS



# Configuring the PS

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: Q-

Name	Select	Description
General		
> AXI Non Secure Enablement	0	Enable AXI Non Secure
GP Slave AXI Interface		
S AXI GP0 interface	<input checked="" type="checkbox"/>	Enables General purpose
S AXI GP1 interface	<input type="checkbox"/>	Enables General purpose
HP Slave AXI Interface		
> S AXI HP0 interface	<input type="checkbox"/>	Enables AXI high performance
> S AXI HP1 interface	<input type="checkbox"/>	Enables AXI high performance
> S AXI HP2 interface	<input type="checkbox"/>	Enables AXI high performance
> S AXI HP3 interface	<input type="checkbox"/>	Enables AXI high performance
ACP Slave AXI Interface		
> DMA Controller		
> PS-PL Cross Trigger interface	<input type="checkbox"/>	Enables PL cross trigger

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

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

Search: Q-

Peripherals

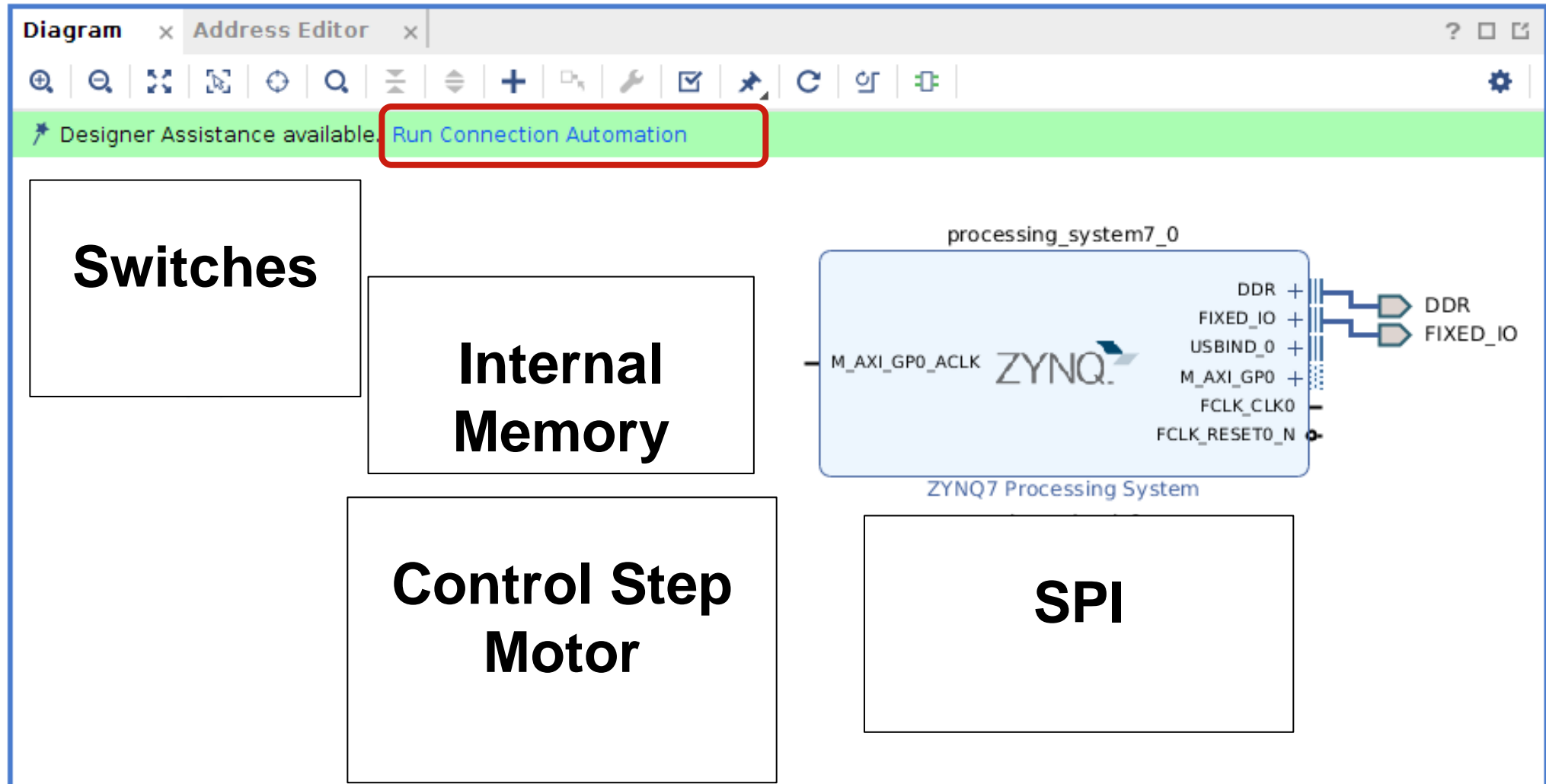
- >  SPI 1
- >  UART 0
- >  UART 1
- I2C 0
- I2C 1
- >  CAN 0
- >  CAN 1
- TTC0

The grid shows the following assignments:

- Pins 31-51 are highlighted in green at the top.
- SPI1 is assigned to pins 34, 35, 46, and 47.
- UART0 is assigned to pins 37, 38, 39, 40, 41, 42, 43, 44, 45, 48, 49, and 50.
- UART1 is assigned to pins 32, 33, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, and 50.
- I2C0 is assigned to pins 34, 35, 46, 47, 48, 49, and 50.
- I2C1 is assigned to pins 36, 37, 48, 49, and 50.
- CAN0 is assigned to pins 34, 35, 46, 47, 48, 49, and 50.
- CAN1 is assigned to pins 36, 37, 48, 49, and 50.

OK Cancel

# Making Up the System

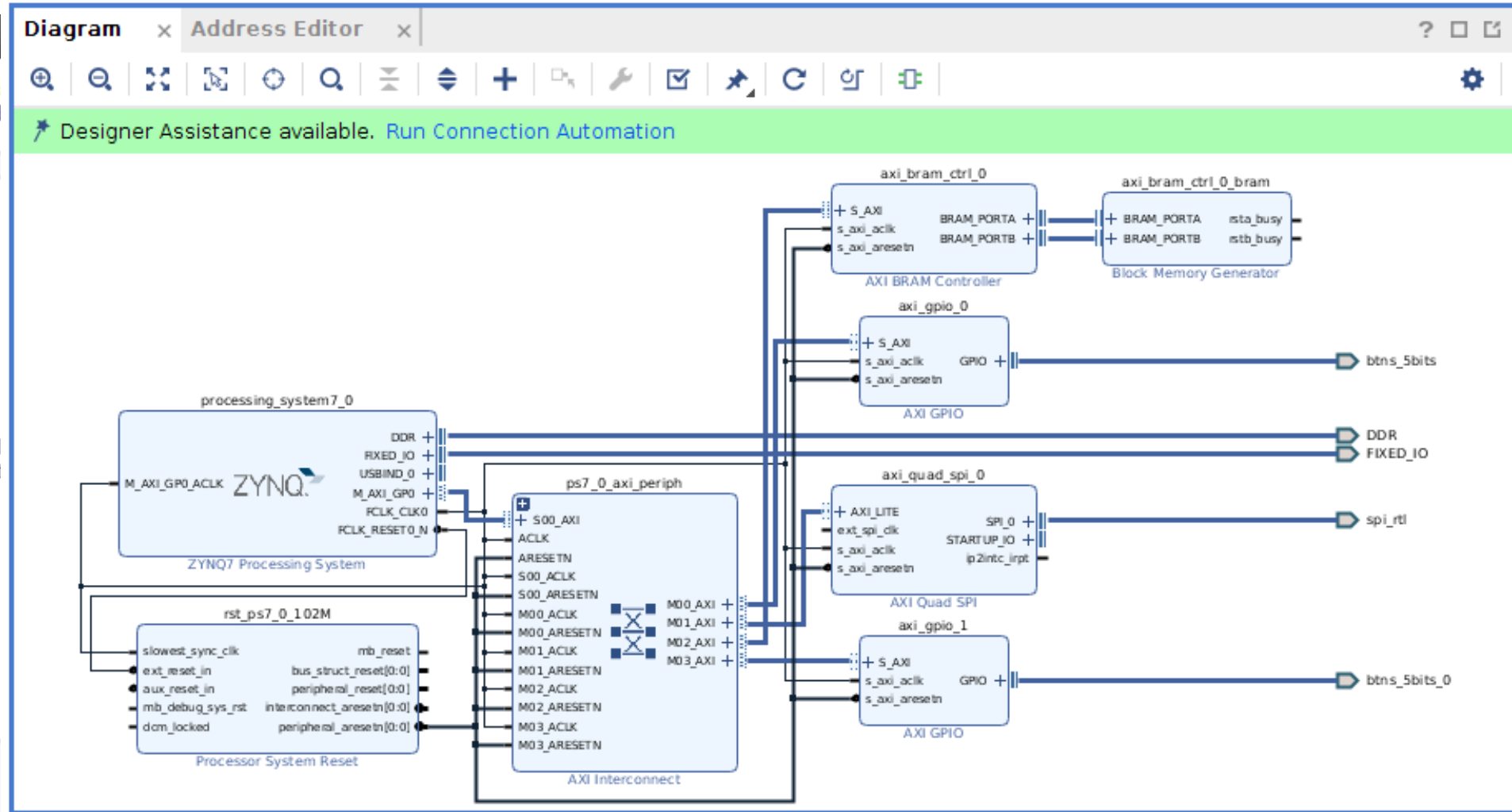


# Running Connection Automation

**Run Connection Automation**

Automatically make connections in your design. Select an interface on the left to do it.

- All Automation (9 out of 9 selected)
- axi\_bram\_ctrl\_0
  - BRAM\_PORTA
  - BRAM\_PORTB
  - S\_AXI
- axi\_gpio\_0
  - GPIO
  - S\_AXI
- axi\_gpio\_1
  - GPIO
  - S\_AXI
- axi\_quad\_spi\_0
  - AXI\_LITE
  - SPI\_0





# Custom GPIO

Re-customize IP

AXI GPIO (2.0)

Documentation IP Location

Component Name

**Board** | **IP Configuration**

Associate IP interface with board interface

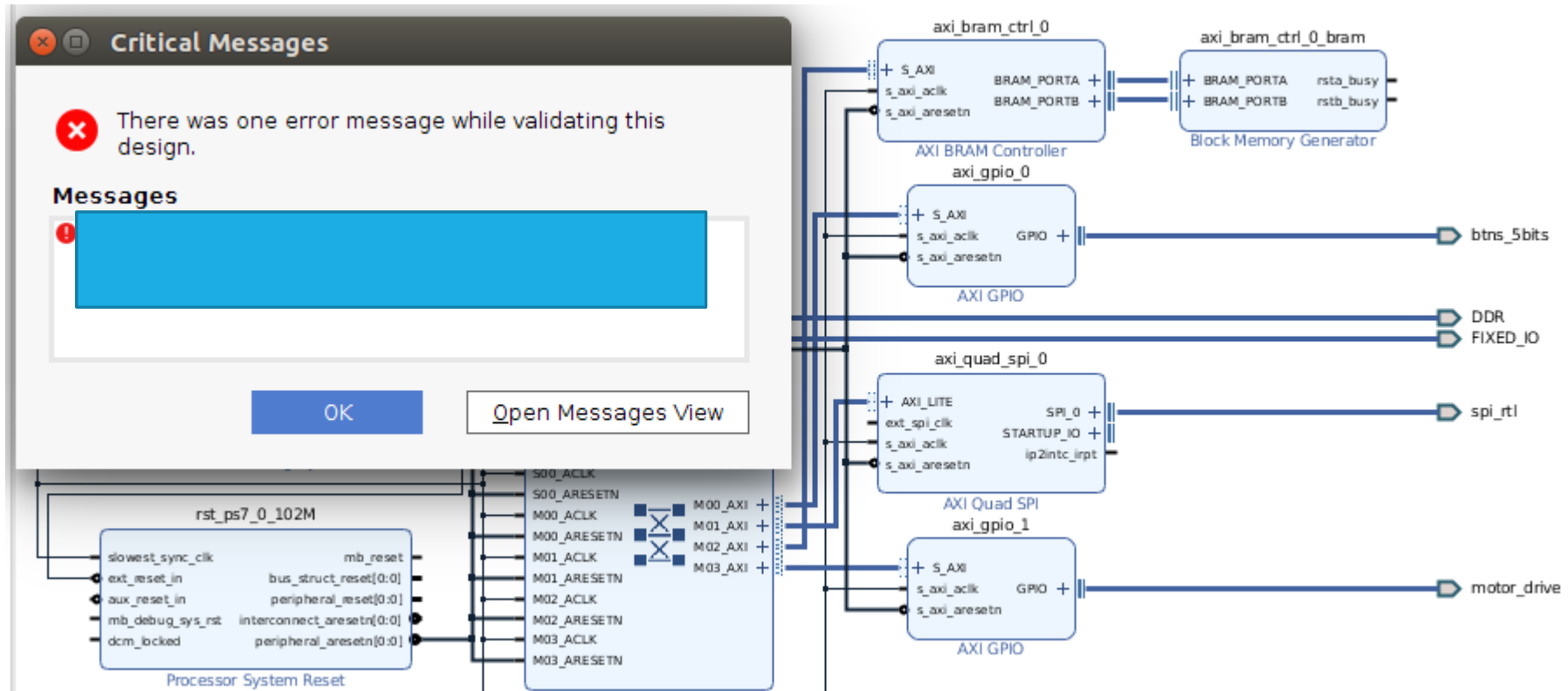
IP Interface	Board Interface
GPIO	btns 5bits
GPIO2	Custom

Clear Board Parameters

Show disabled ports

+ S\_AXI  
- s\_axi\_aclk GPIO +  
s\_axi\_aresetn

# DRC (Design Rule Check) Design Validation



# Memory Map

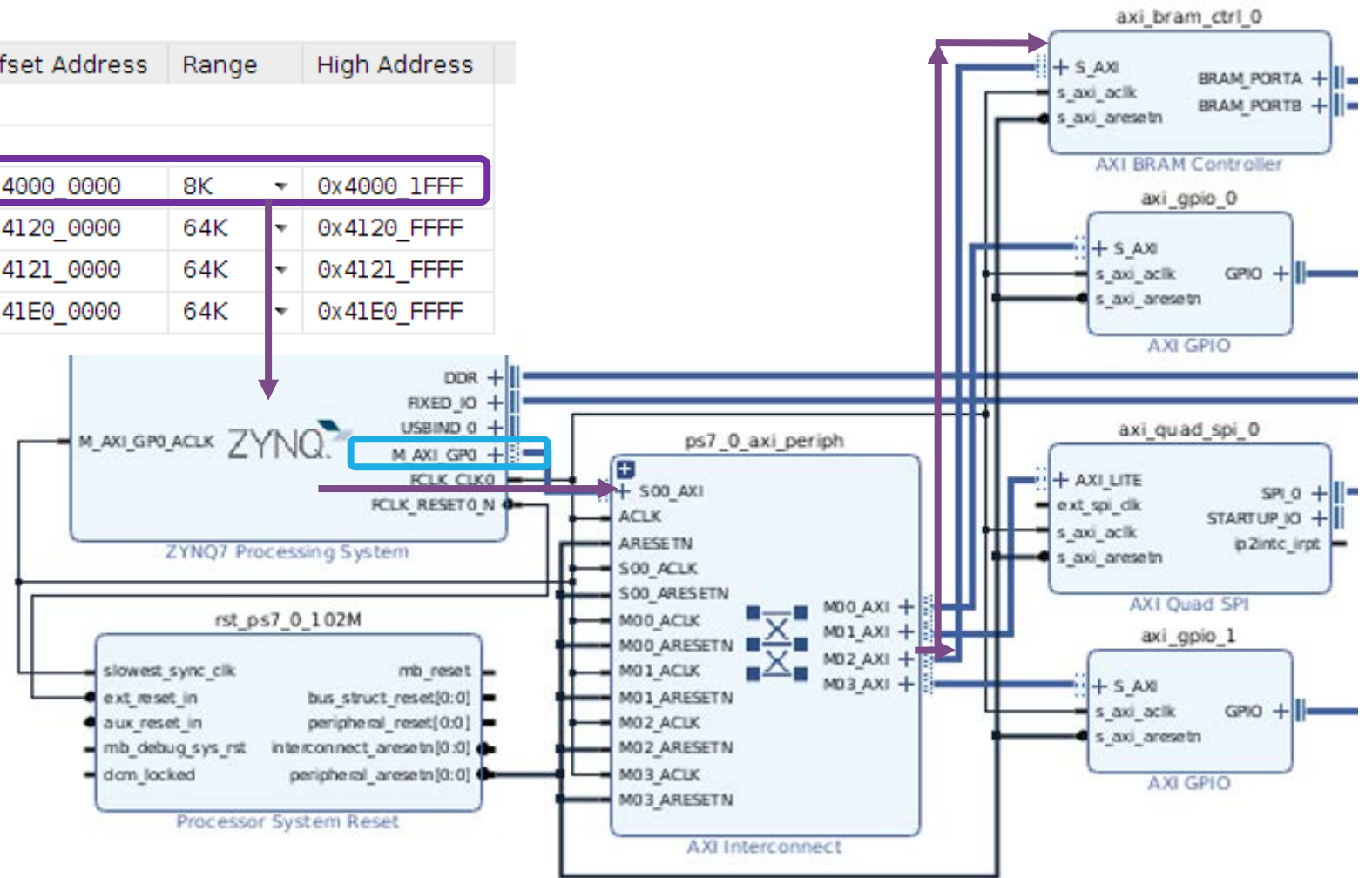
Diagram x Address Editor x ?

Cell Slave Interface Base Name Offset Address Range High Address

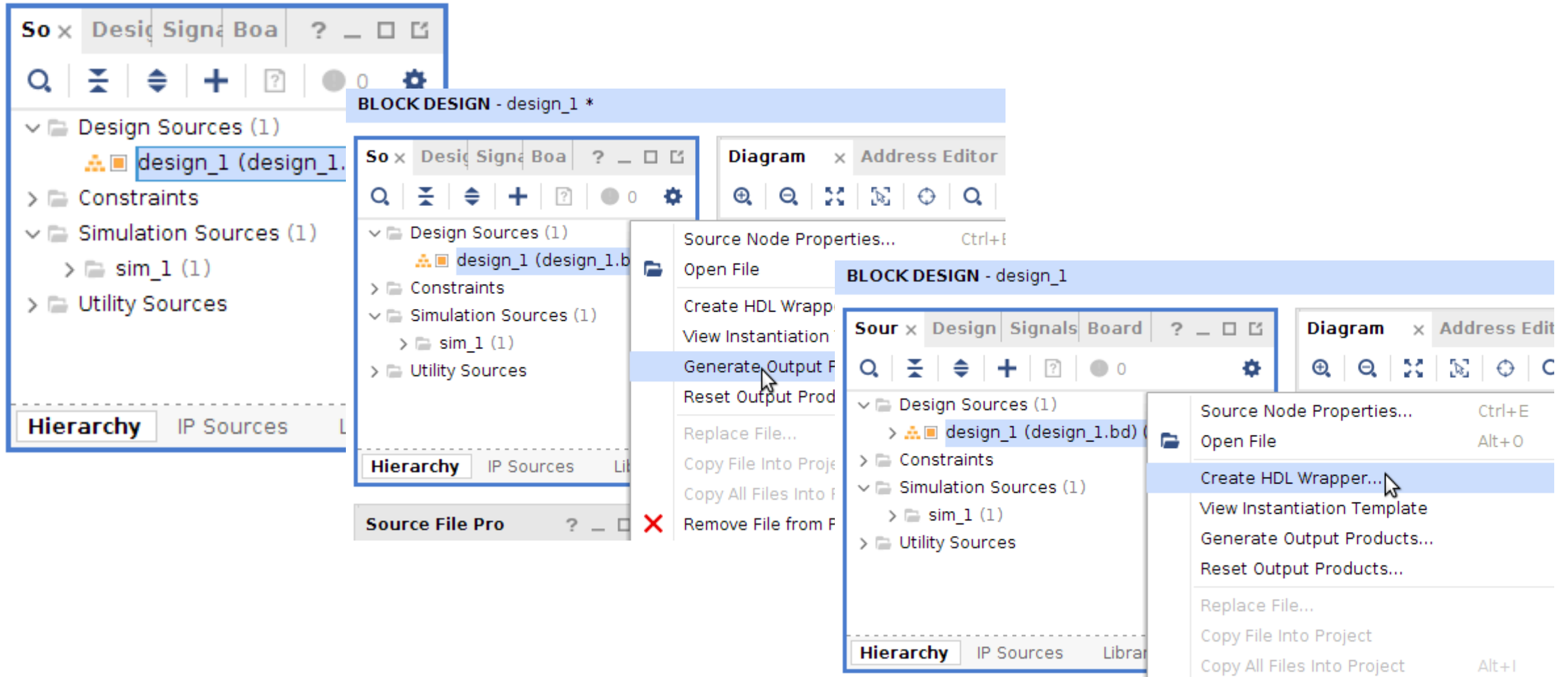
processing\_system7\_0

Data (32 address bits : 0x40000000 [ 1G ])

axi_bram_ctrl_0	S_AXI	Mem0	0x4000_0000	8K	0x4000_1FFF
axi_gpio_0	S_AXI	Reg	0x4120_0000	64K	0x4120_FFFF
axi_gpio_1	S_AXI	Reg	0x4121_0000	64K	0x4121_FFFF
axi_quad_spi_0	AXI_LITE	Reg	0x41E0_0000	64K	0x41E0_FFFF



# Getting the System Ready to be Implemented



# Export Hardware Design to SDK

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

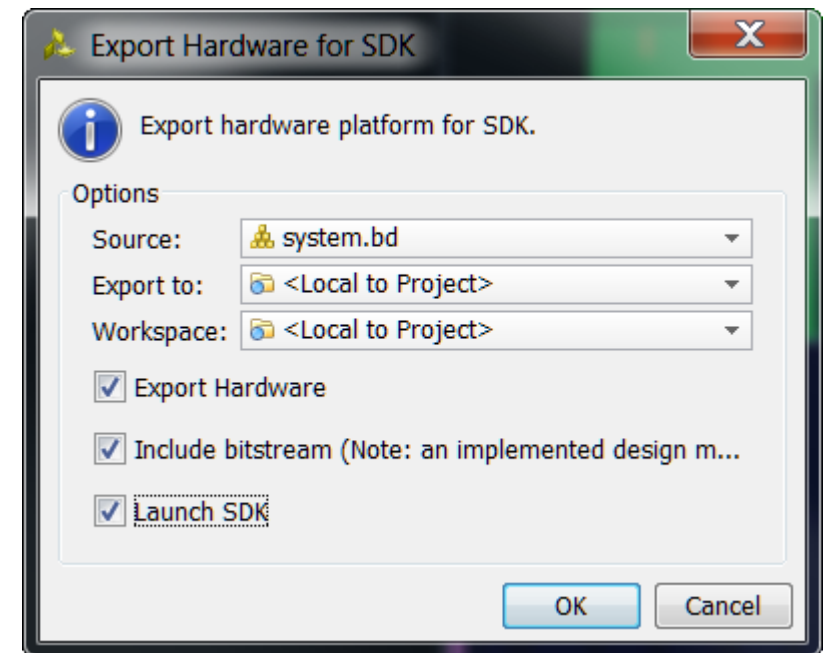
The design must be opened if a bitstream of the design is generated

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

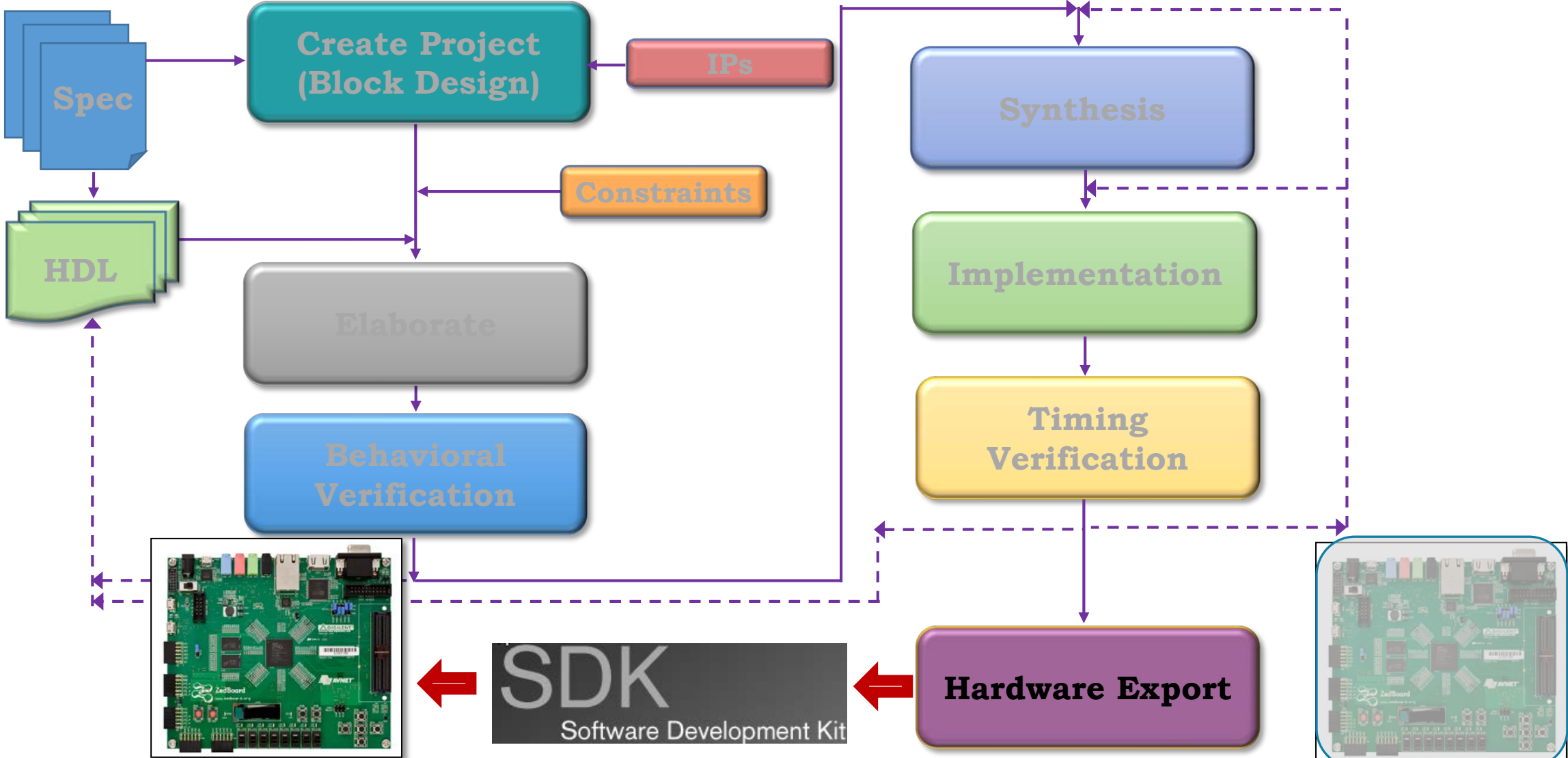
The SDK tool will then associate user software projects to hardware



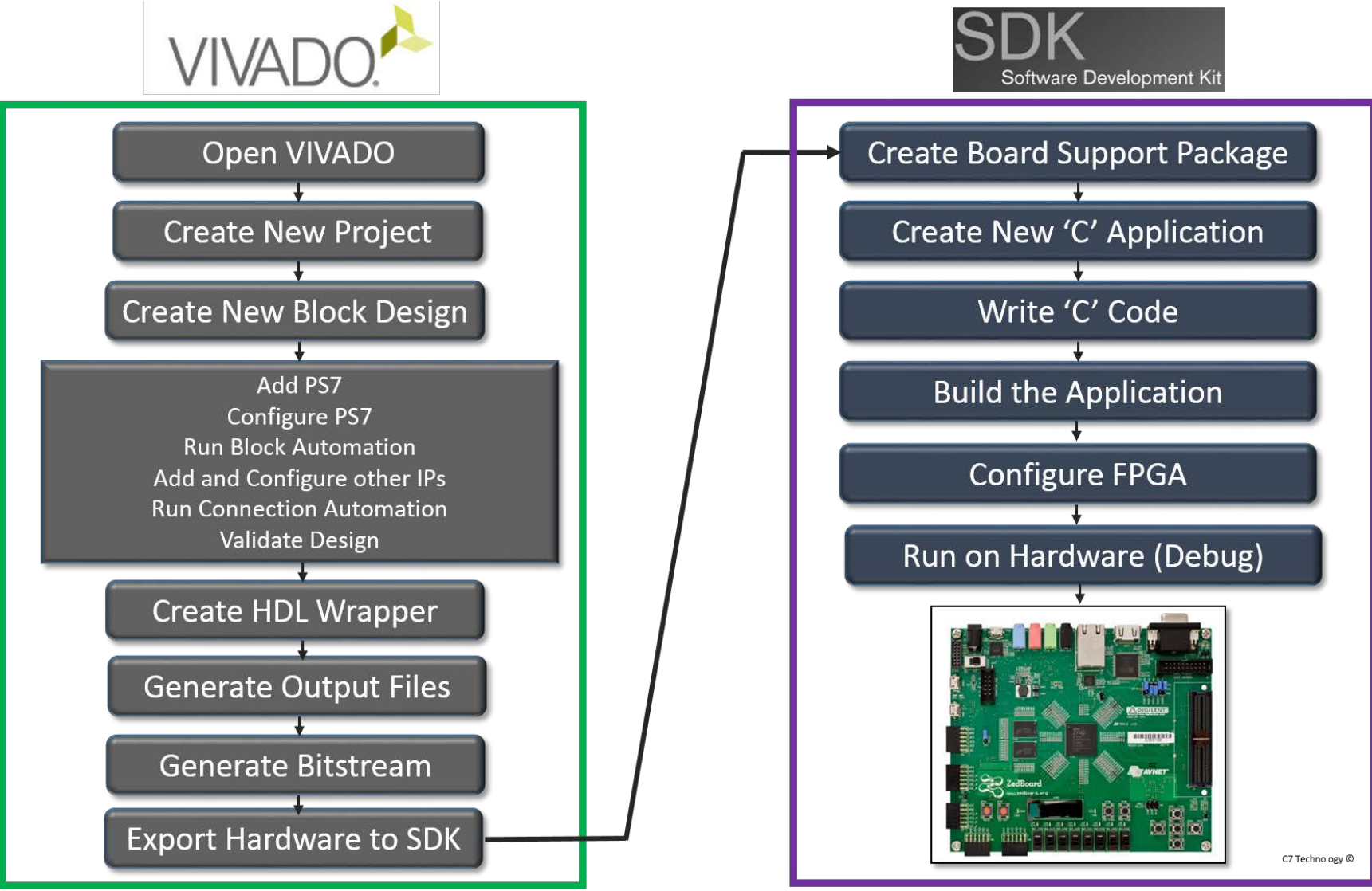
# Software Development Kit (SDK)

---

# Embedded System Design – Vivado-SDK Flow



# Embedded System Design – Vivado-SDK Flow





# Embedded System Tools: Software

---

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

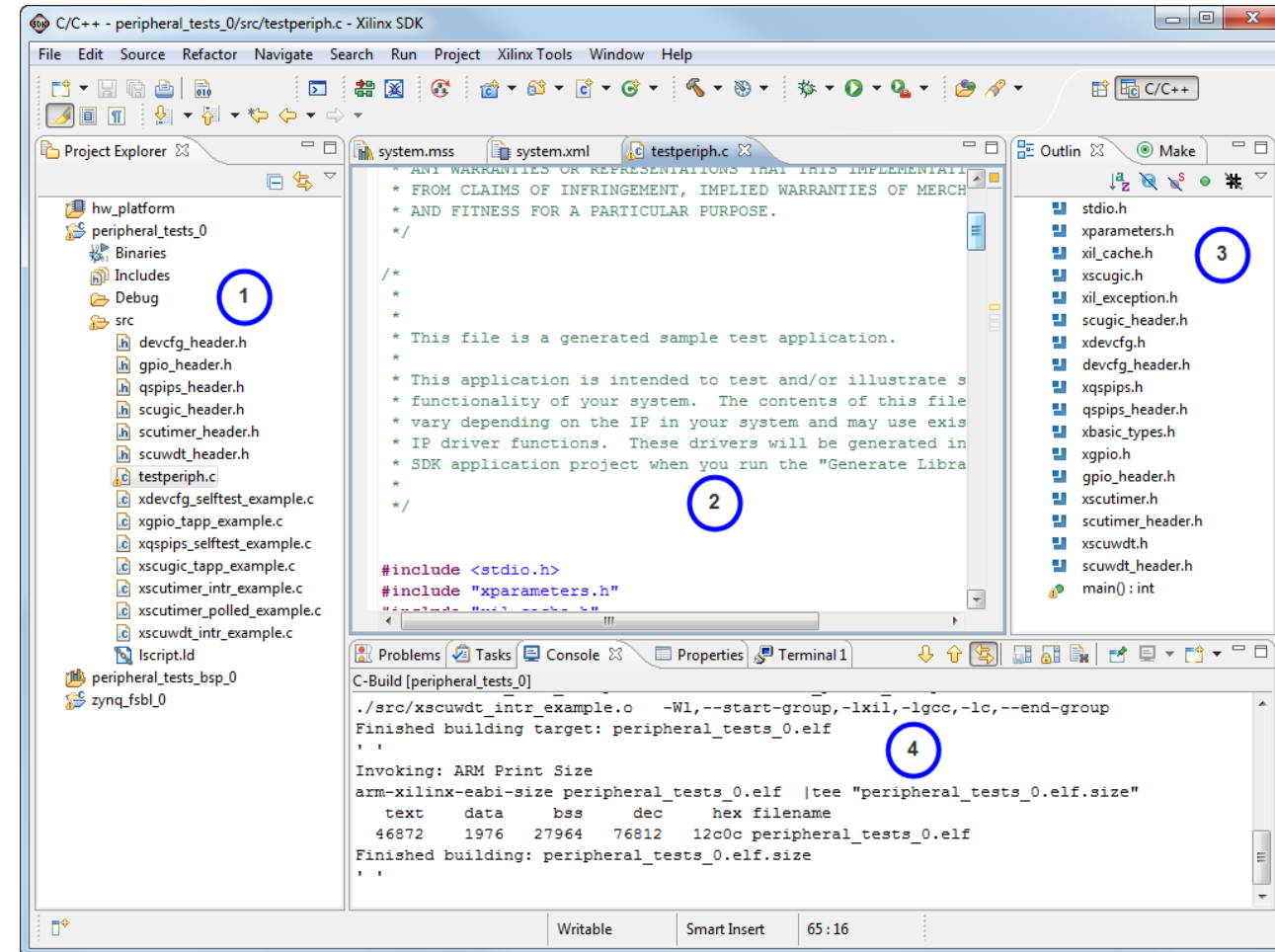
- Board support package creation : LibGen
- GNU software development tools
- C/C++ compiler for the ARM Cortex-A9 processor (gcc)
- Debugger for the ARM Cortex-A9 processor (gdb)

## 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`

```
/* */
/* 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;
}
```

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

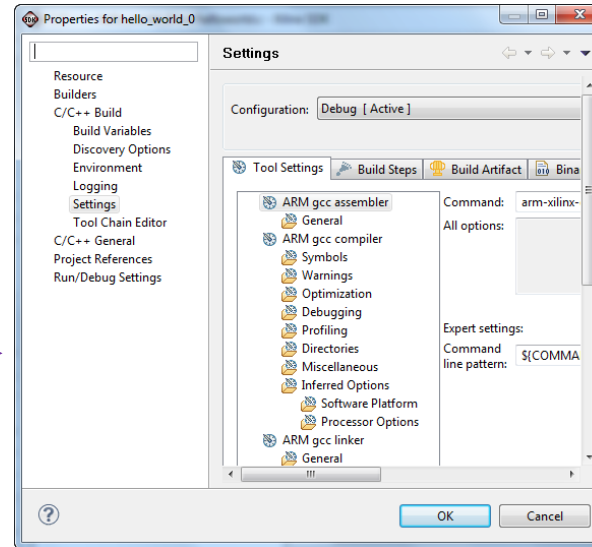
Finished building: hello\_world\_0.elf.size

' '

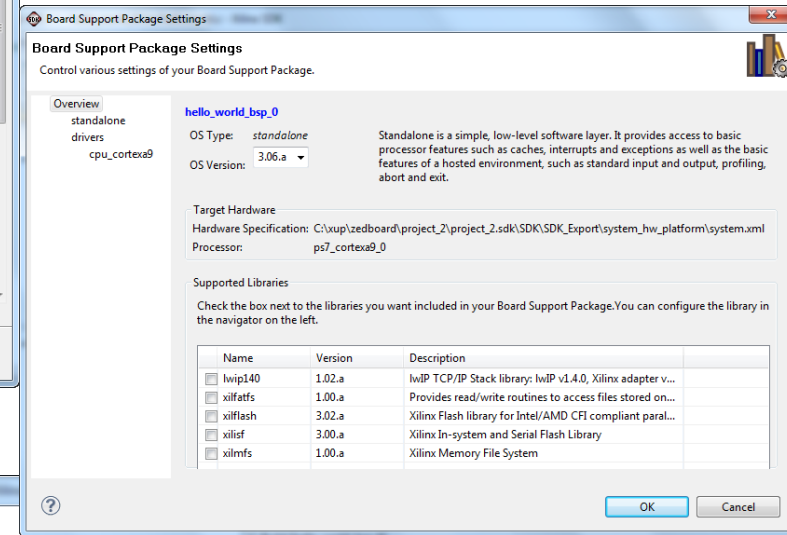
# Software Management Settings

Software is managed in three major areas

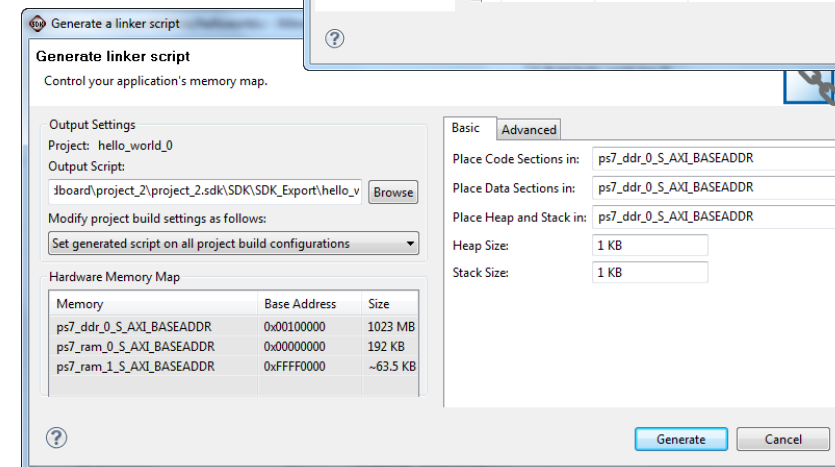
- Compiler/Linker Options
  - Application program



- Software Platform Settings
  - Board support package



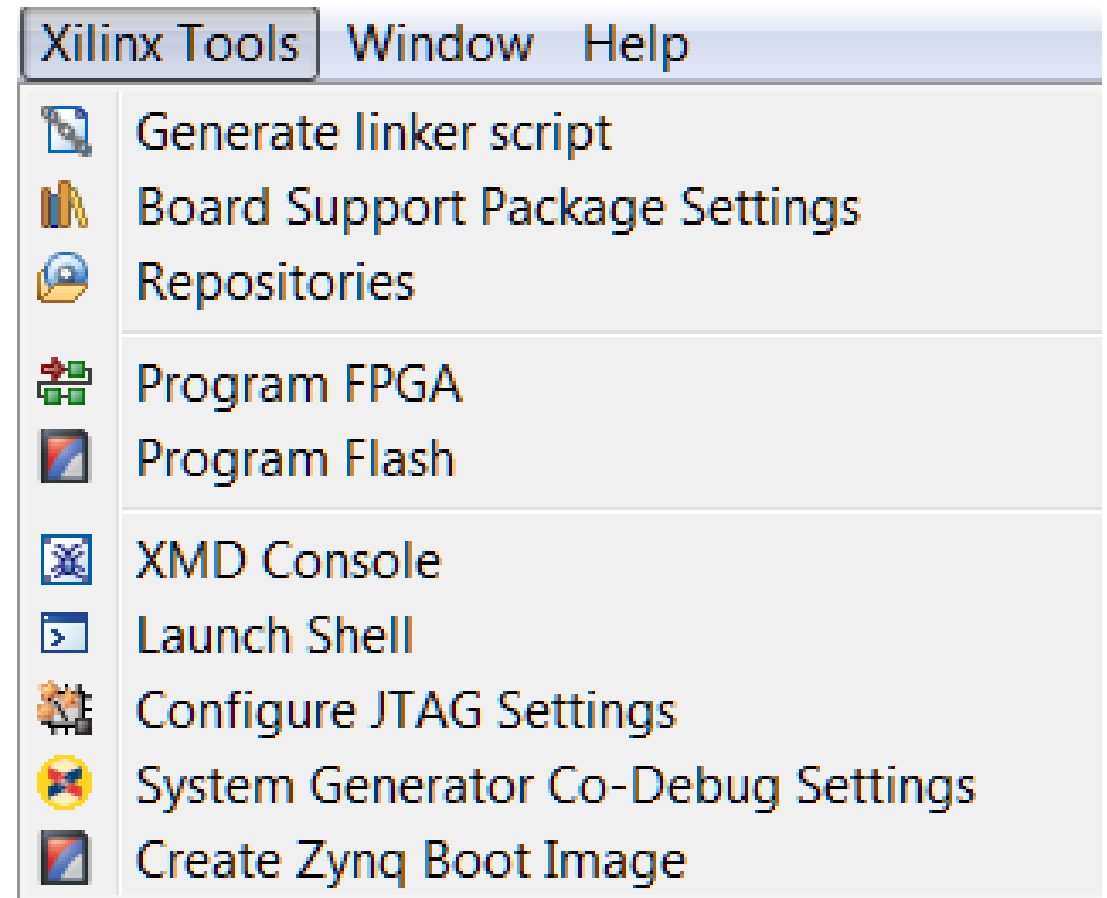
- Linker Script Generation
  - Assigning software to memory resources



# 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



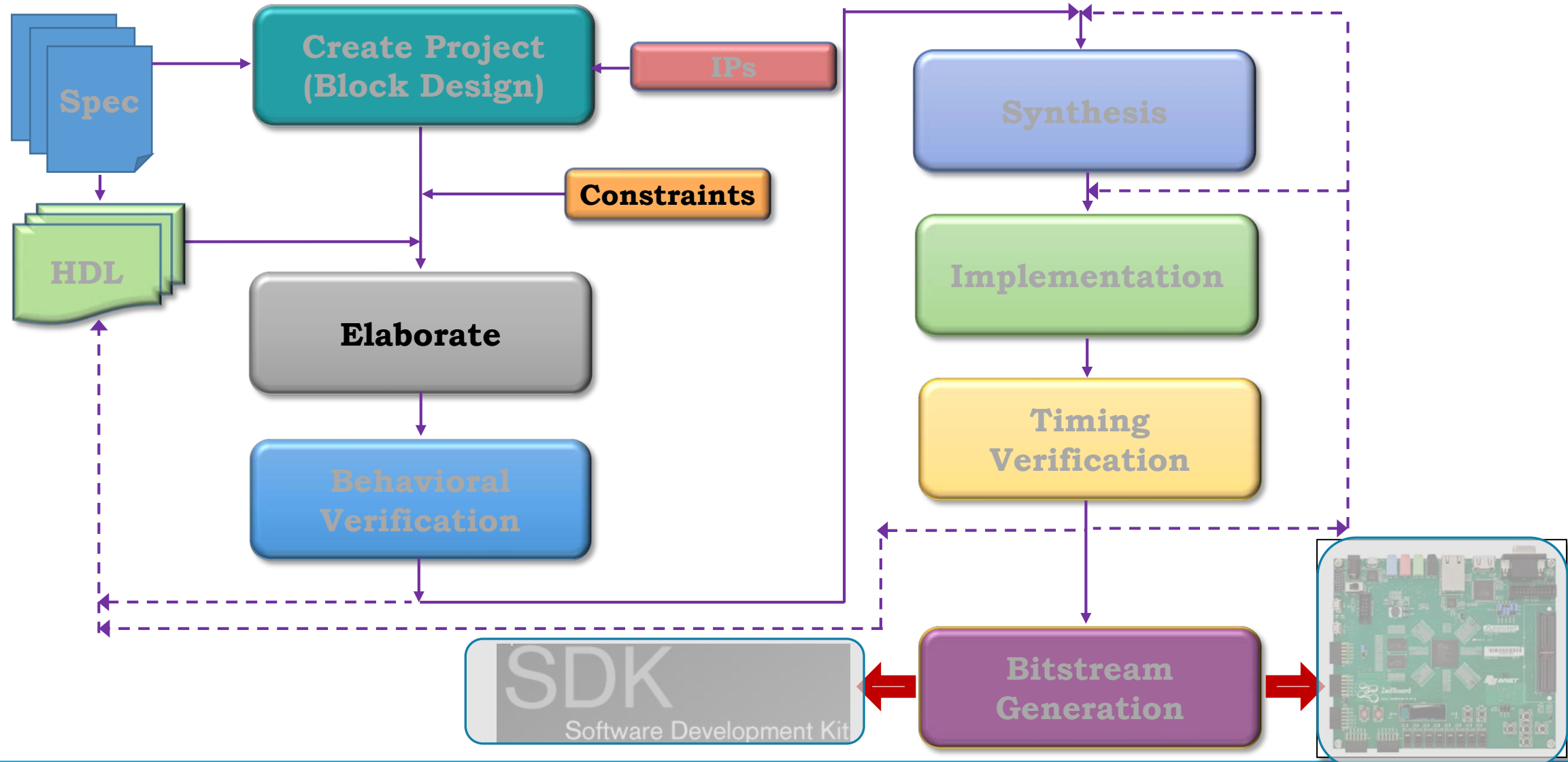
# Apendix

---

# *Vivado Design Suite* *Elaboration Process*

---

# Embedded System Design – Vivado Flow





# Elaboration

---

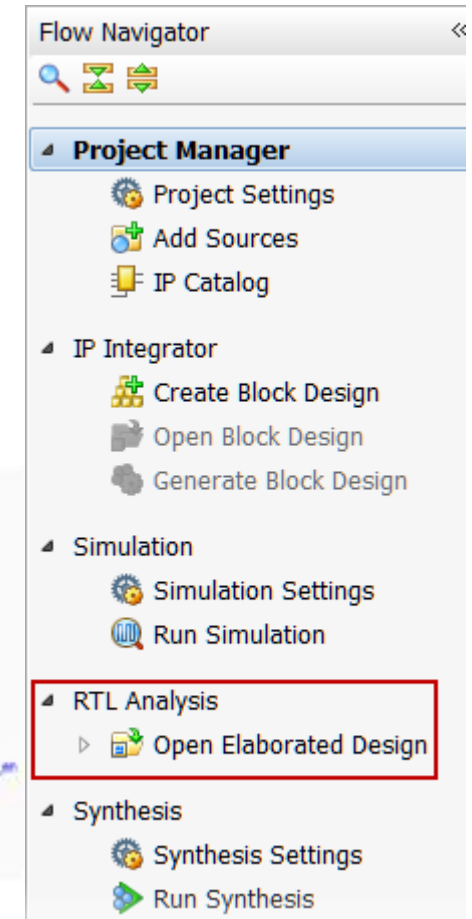
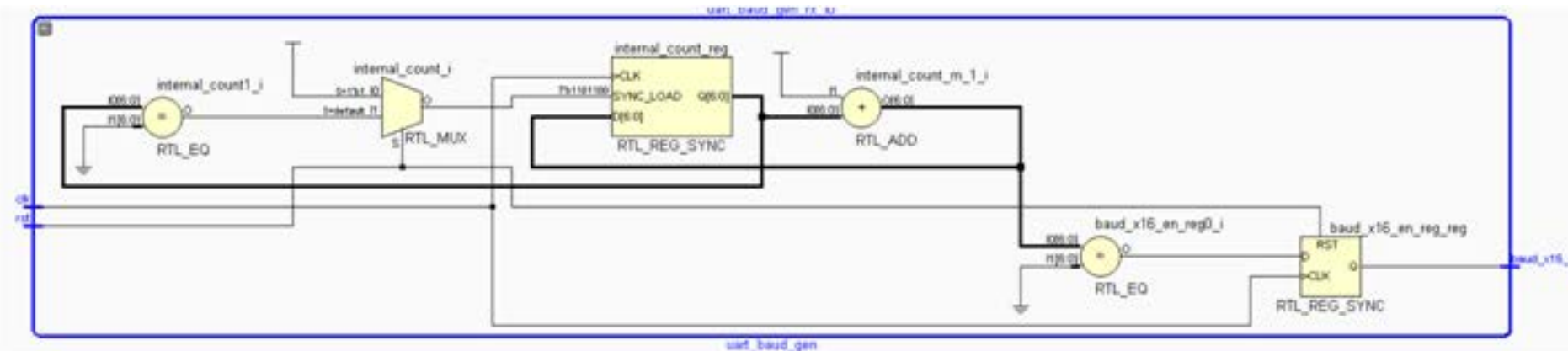
- Elaboration is the RTL optimization to an FPGA technology
- Vivado IDE allows designers to import and manage RTL sources
  - Verilog, System Verilog, VHDL, NGC, or testbenches
- Create and modify sources with the RTL Editor
  - Cross-selection between all the views
- Sources view
  - Hierarchy view: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

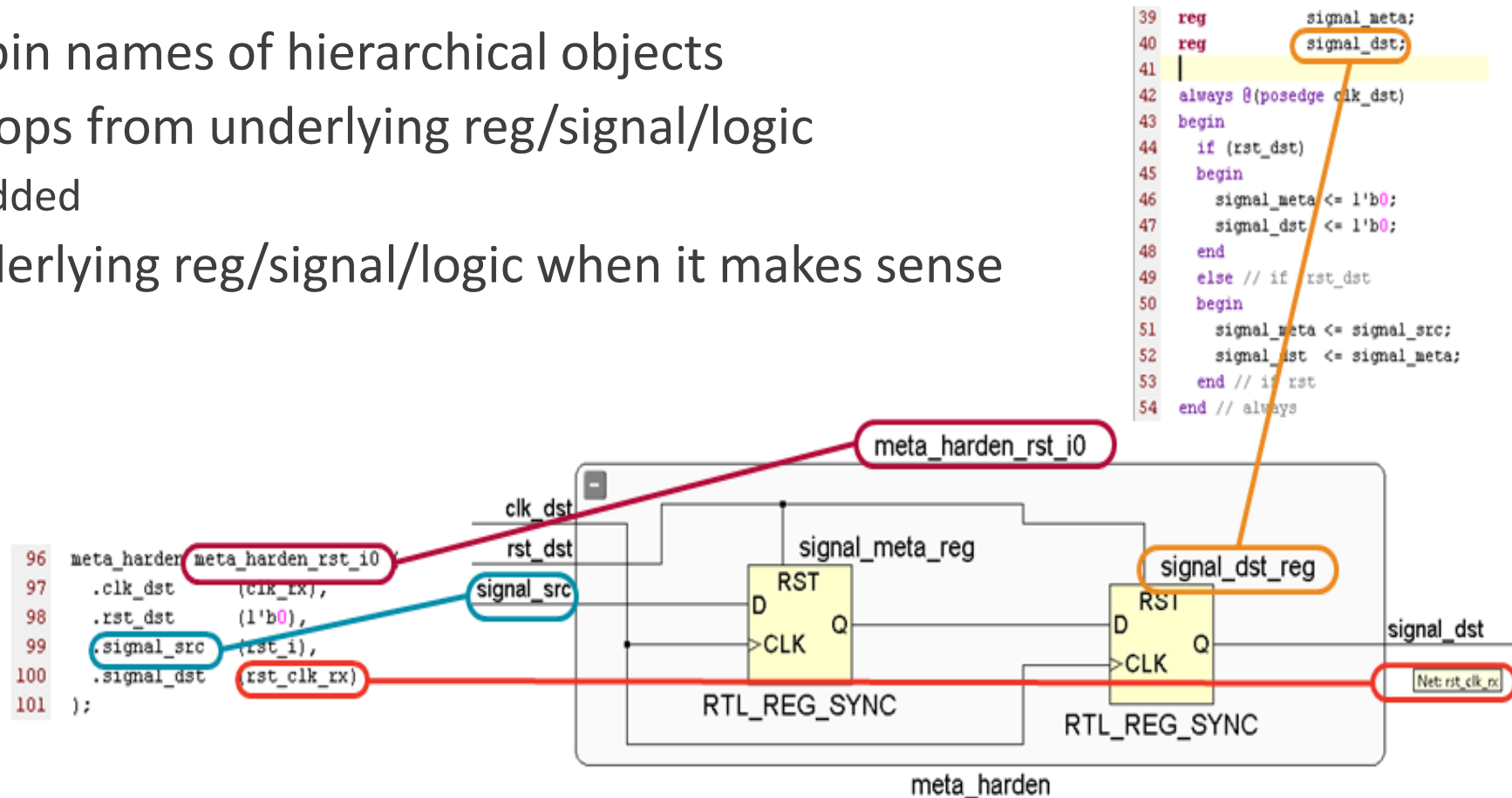
- Interconnected netlist of hierarchical and generic technology cells
- Instances of modules/entities
- Generic technology representations of hardware components
  - AND, OR, buffer, multiplexers, adders, comparators, etc...



# 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

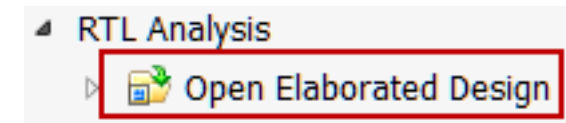
Click on the Open Elaborated Design under RTL Analysis to

- 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:

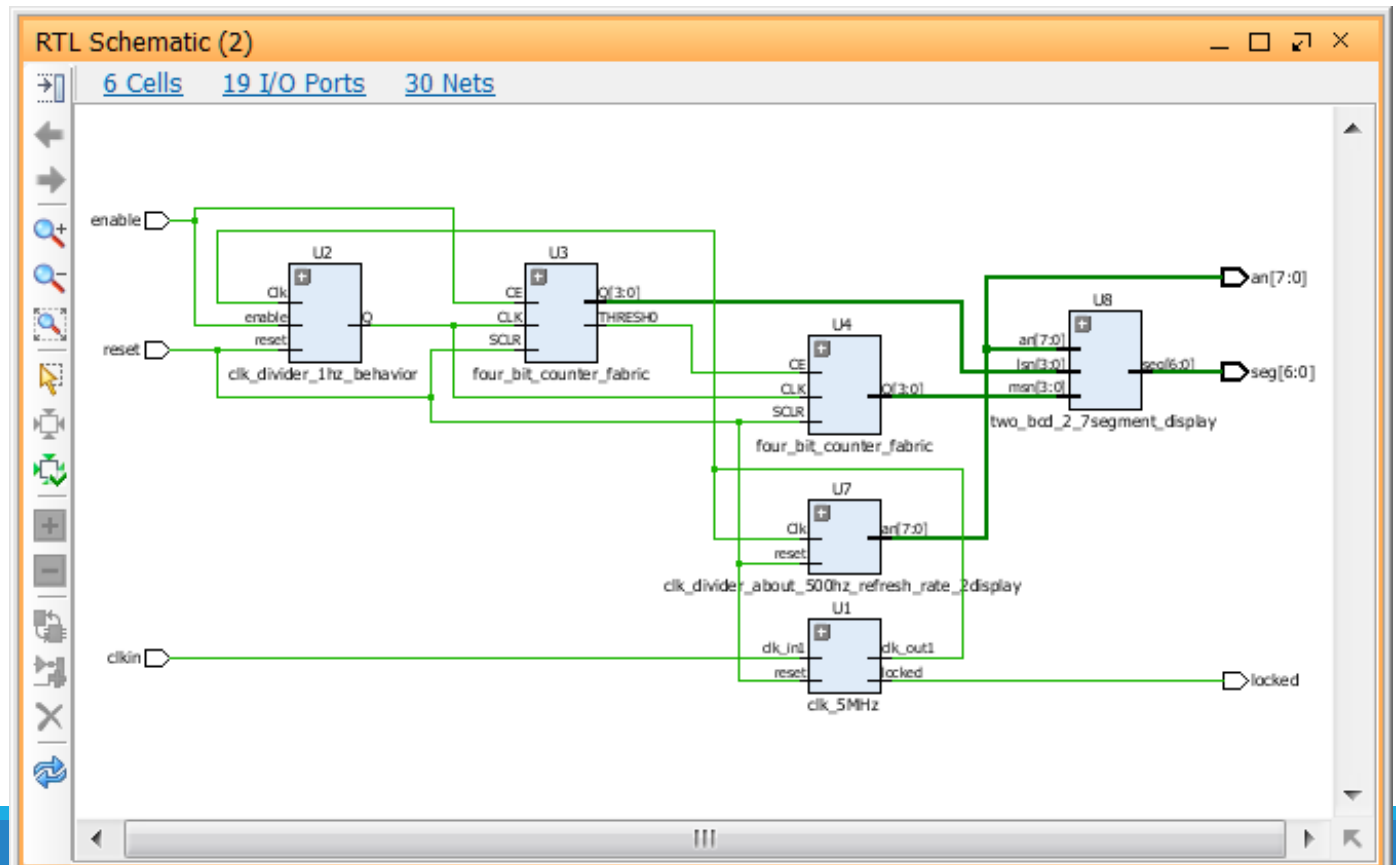
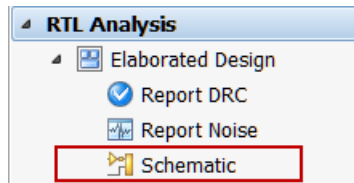
- RTL compilation validation and syntax checking
- Netlist and schematic exploration
- Design rule checks
- Early I/O pin planning using an RTL port list
- Ability to select an object in one view and cross probe to the object in other views, including instantiations and logic definitions within the RTL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



# Schematic View of an Elaborated Design

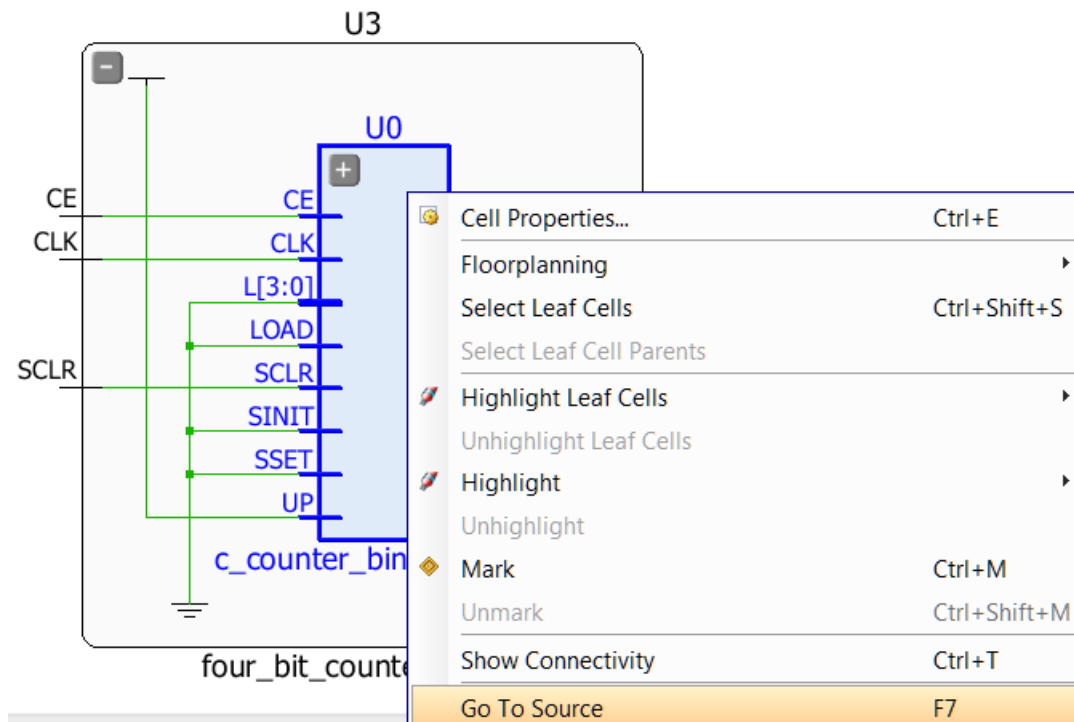
When Schematic is clicked under the Elaborated Design, the schematic is opened showing the hierarchical blocks

- 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

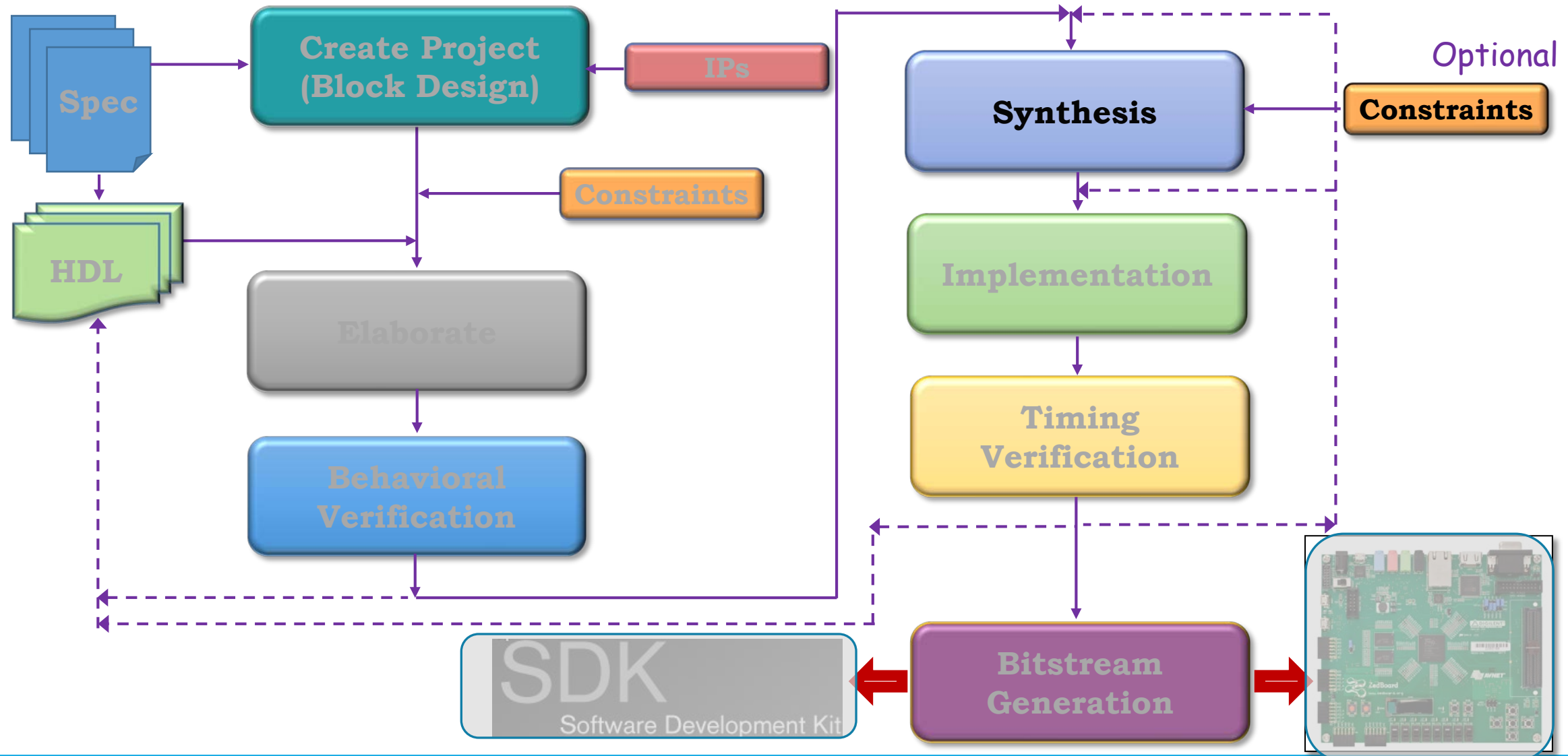


```
114 BEGIN
115   U0 : c_counter_binary_v12_0
116     GENERIC MAP (
117       C_IMPLEMENTATION => 0,
118       C_VERBOSITY => 0,
119       C_XDEVICEFAMILY => "artix7",
120       C_WIDTH => 4,
121       C_HAS_CE => 1,
122       C_HAS_SCLR => 1,
123       C_RESTRICT_COUNT => 1,
124       C_COUNT_TO => "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





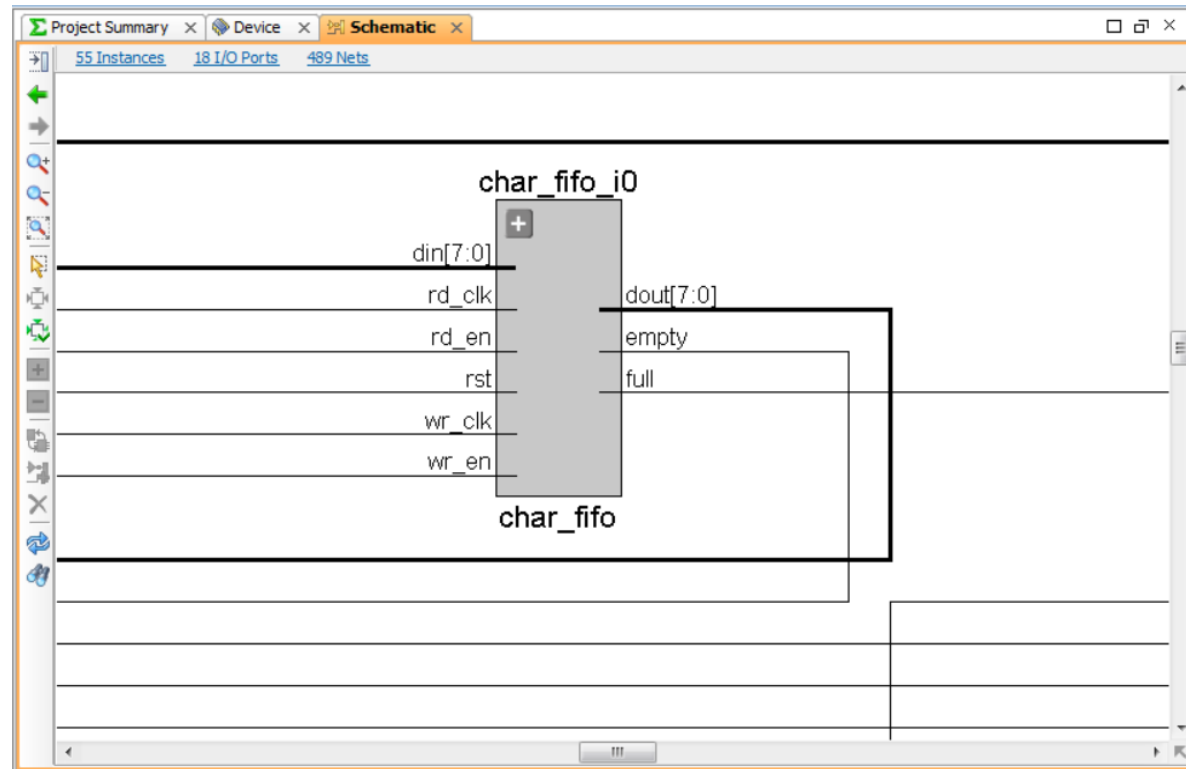
# Vivado IDE Synthesis

---

- **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**

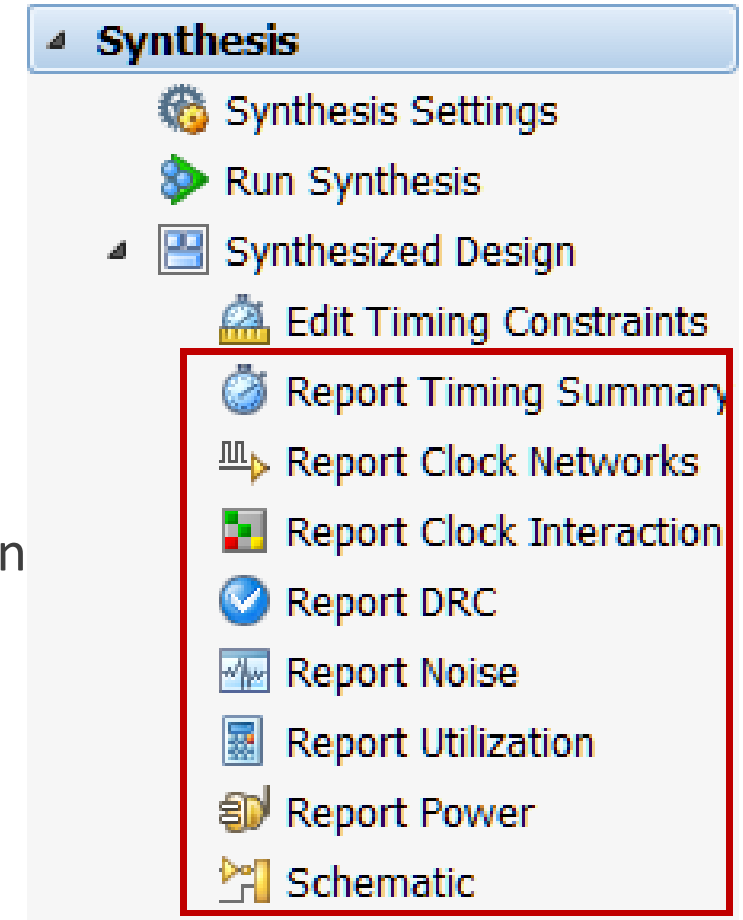
- Interconnected netlist of hierarchical and basic elements (BELs)
  - Instances of modules/entities
  - Basic elements
    - LUTs, flip-flops, carry chain elements, wide MUXes
    - Block RAMs, DSP cells
    - Clocking elements (BUFG, BUFR, MMCM, ...)
    - I/O elements (IBUF, OBUF, I/O flip-flops)

**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

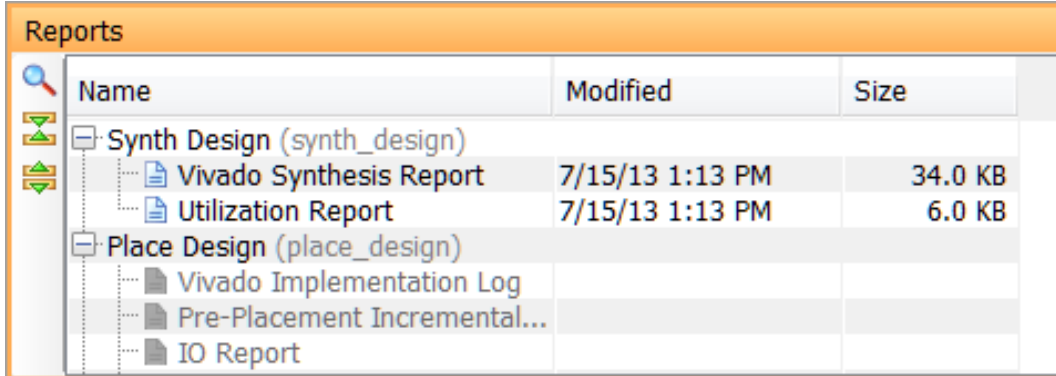
- Report Timing Summary: Generate a default timing report
- Report Clock Networks: Generates a clock tree for the design
- Report Clock Interaction: Verifies constraint coverage on paths between clock domains
- Report DRC: Performs design rule check on the entire design
- Report Noise: Performs an SSO 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



The screenshot shows the 'Reports' window in Vivado. It contains a table with columns for 'Name', 'Modified', and 'Size'. The table lists reports for 'Synth Design (synth\_design)' and 'Place Design (place\_design)'. The 'Synth Design' reports include 'Vivado Synthesis Report' (34.0 KB) and 'Utilization Report' (6.0 KB), both modified on 7/15/13 at 1:13 PM. The 'Place Design' reports include 'Vivado Implementation Log', 'Pre-Placement Incremental...', and 'IO Report'.

Name	Modified	Size
Synth Design (synth_design)		
Vivado Synthesis Report	7/15/13 1:13 PM	34.0 KB
Utilization Report	7/15/13 1:13 PM	6.0 KB
Place Design (place_design)		
Vivado Implementation Log		
Pre-Placement Incremental...		
IO Report		

# 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

```

## 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 -----

```

```

26 1. Slice Logic
27 -----
28
29 +-----+-----+-----+-----+-----+
30 | Site Type | Used | Loded | Available | Util% |
31 +-----+-----+-----+-----+-----+
32 | Slice LUTs* | 73 | 0 | 63400 | 0.11 |
33 | LUT as Logic | 73 | 0 | 63400 | 0.11 |
34 | LUT as Memory | 0 | 0 | 19000 | 0.00 |
35 | Slice Registers | 50 | 0 | 126800 | 0.03 |
36 | Register as Flip Flop | 50 | 0 | 126800 | 0.03 |
37 | Register as Latch | 0 | 0 | 126800 | 0.00 |
38 | F7 Muxes | 4 | 0 | 31700 | 0.01 |
39 | F8 Muxes | 0 | 0 | 15850 | 0.00 |
40 +-----+-----+-----+-----+-----+

```

```

91 5. Clocking
92 -----
93
94 +-----+-----+-----+-----+-----+
95 | Site Type | Used | Loded | Available | Util% |
96 +-----+-----+-----+-----+-----+
97 | BUFGCTRL | 2 | 0 | 32 | 6.25 |
98 | BUFIO | 0 | 0 | 24 | 0.00 |
99 | MMCME2_ADV | 1 | 0 | 6 | 16.66 |
100 | PLLE2_ADV | 0 | 0 | 6 | 0.00 |
101 | BUFMRCE | 0 | 0 | 12 | 0.00 |
102 | BUFHCE | 0 | 0 | 96 | 0.00 |
103 | BUFR | 0 | 0 | 24 | 0.00 |
104 +-----+-----+-----+-----+-----+

```

## 4. IO and GTX Specific

```

67 4. IO and GTX Specific
68 -----
69
70 +-----+-----+-----+-----+-----+
71 | Site Type | Used | Loded | Available | Util% |
72 +-----+-----+-----+-----+-----+
73 | Bonded IOB | 19 | 0 | 210 | 9.04 |
74 | Bonded IPADs | 0 | 0 | 2 | 0.00 |
75 | IBUFGDS | 0 | 0 | 202 | 0.00 |
76 | IDELAYCTRL | 0 | 0 | 6 | 0.00 |
77 | IN_FIFO | 0 | 0 | 24 | 0.00 |
78 | OUT_FIFO | 0 | 0 | 24 | 0.00 |
79 | PHASER_REF | 0 | 0 | 6 | 0.00 |
80 | PHY_CONTROL | 0 | 0 | 6 | 0.00 |
81 | PHASER_OUT/PHASER_OUT_PHY | 0 | 0 | 24 | 0.00 |
82 | PHASER_IN/PHASER_IN_PHY | 0 | 0 | 24 | 0.00 |
83 | IDELAYE2/IDELAYE2_FINEDELAY | 0 | 0 | 300 | 0.00 |
84 | ODELAYE2/ODELAYE2_FINEDELAY | 0 | 0 | 0 | 0.00 |
85 | IBUFDS_GTE2 | 0 | 0 | 4 | 0.00 |
86 | ILOGIC | 0 | 0 | 210 | 0.00 |
87 | OLOGIC | 0 | 0 | 210 | 0.00 |
88 +-----+-----+-----+-----+-----+

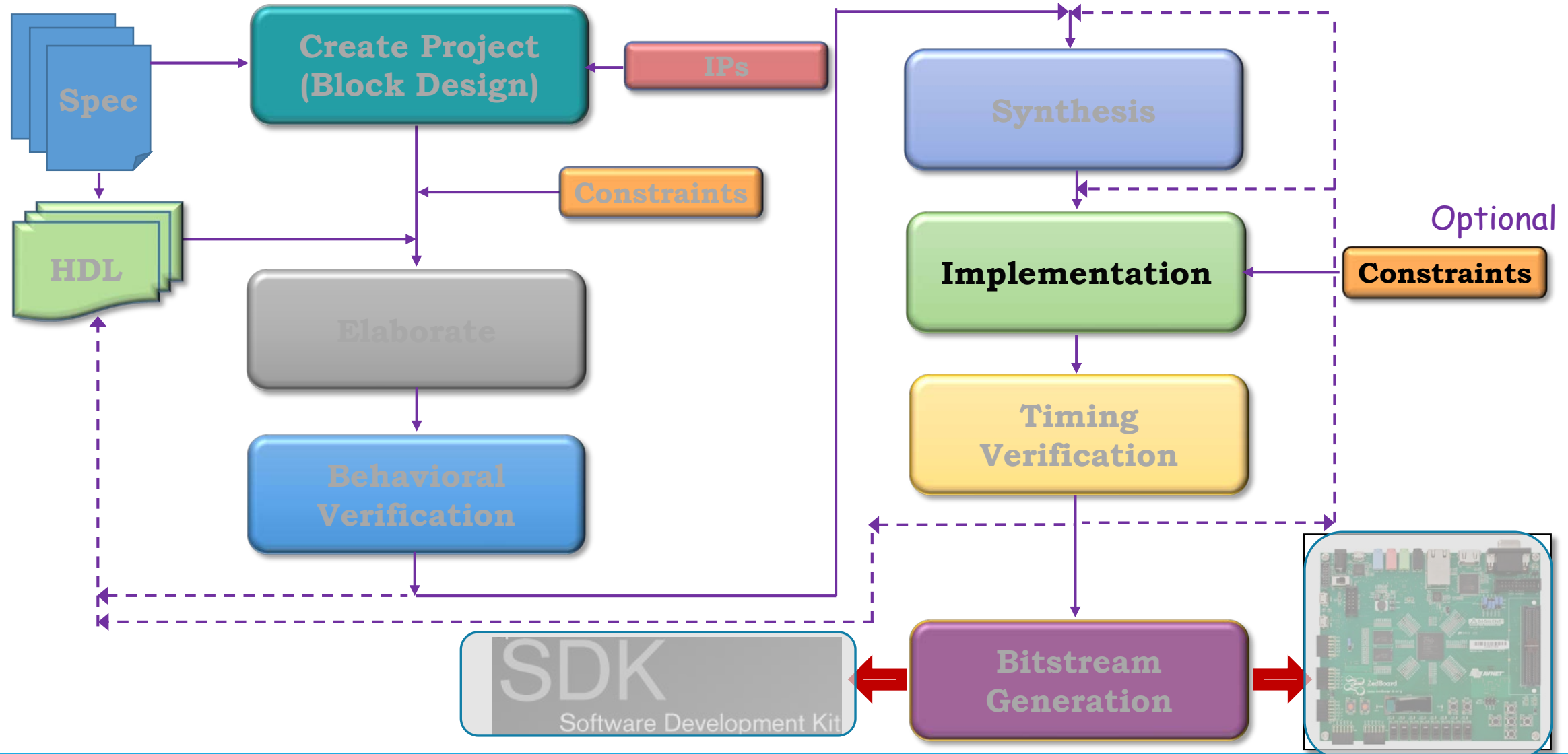
```

# *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

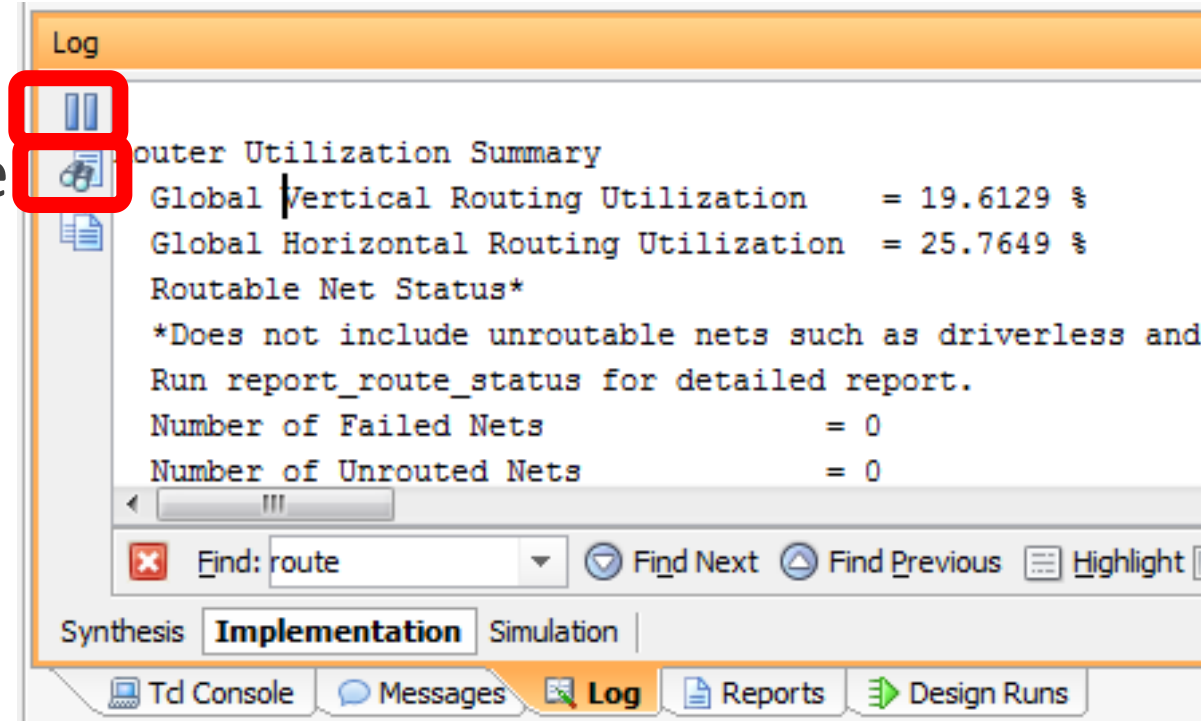
- Package pin placement
- Absolute or relative placement of cells:
  - Block RAM
  - DSP
  - LUTs
  - Flip-Flops
- Floorplanning constraints that assign cells to general regions of an FPGA

**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

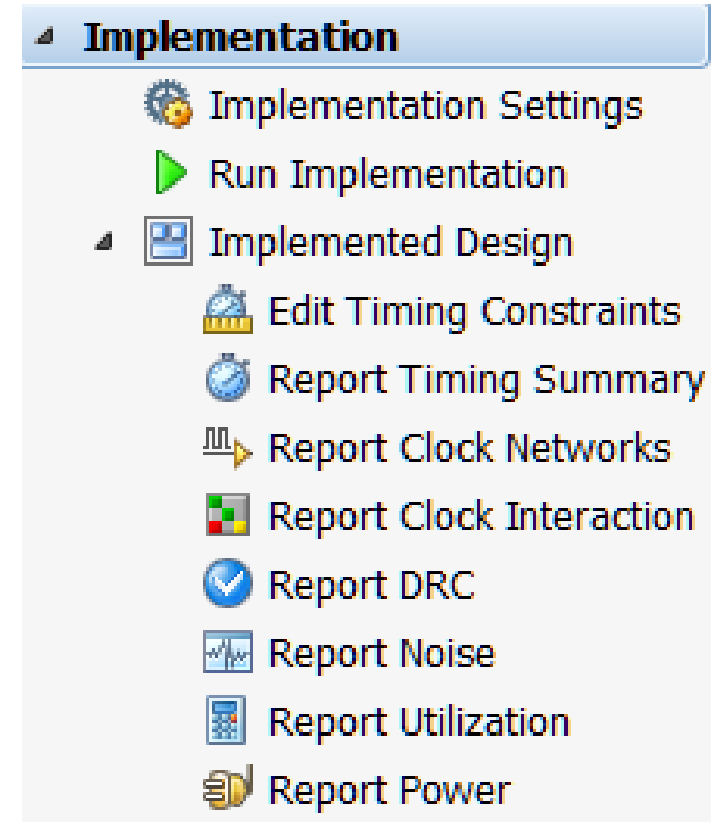
The Log window opens in the Vivado IDE after you launch a run. It shows the standard output messages. It also displays details about the progress of each individual implementation process, such as *place\_design* and *route\_design*.



# After Implementation

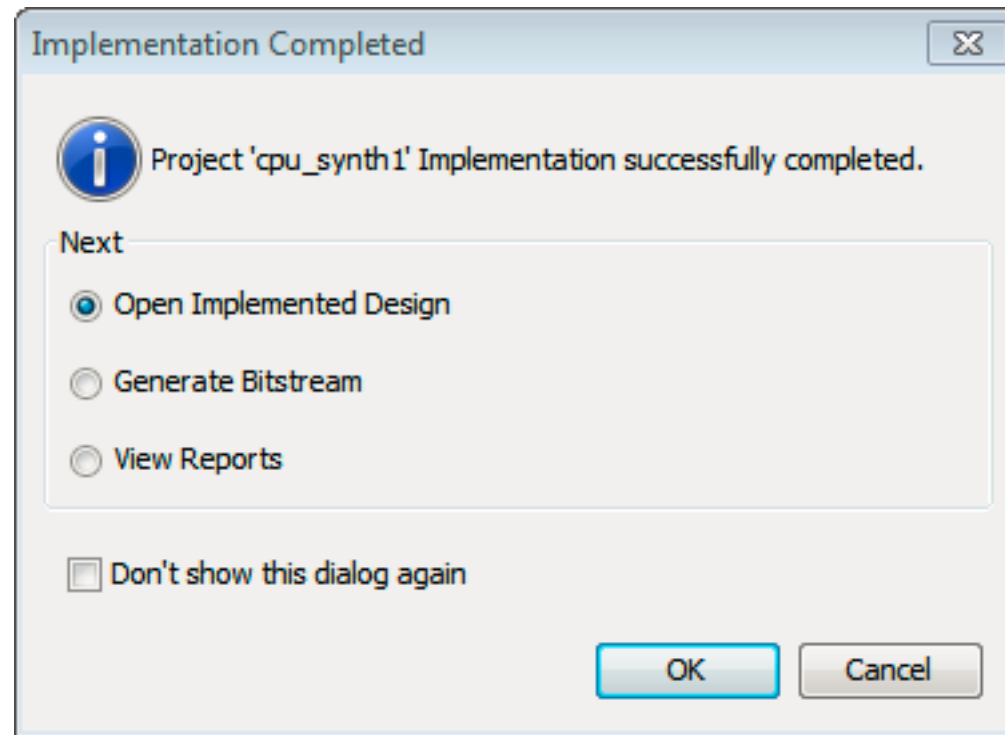
---

- 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



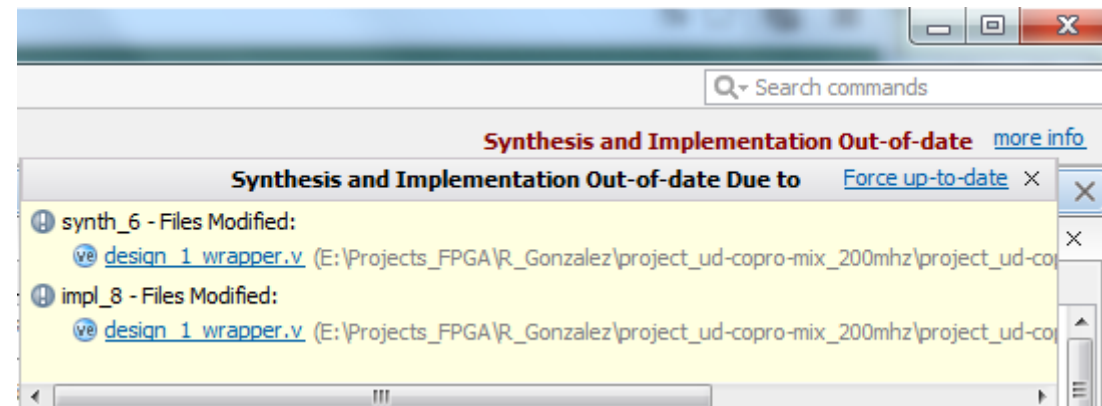
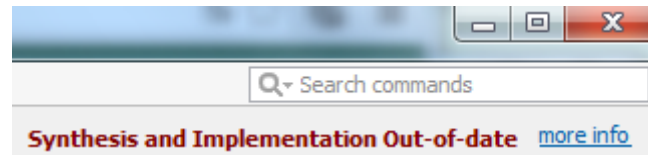
# After Completing Implementation

---

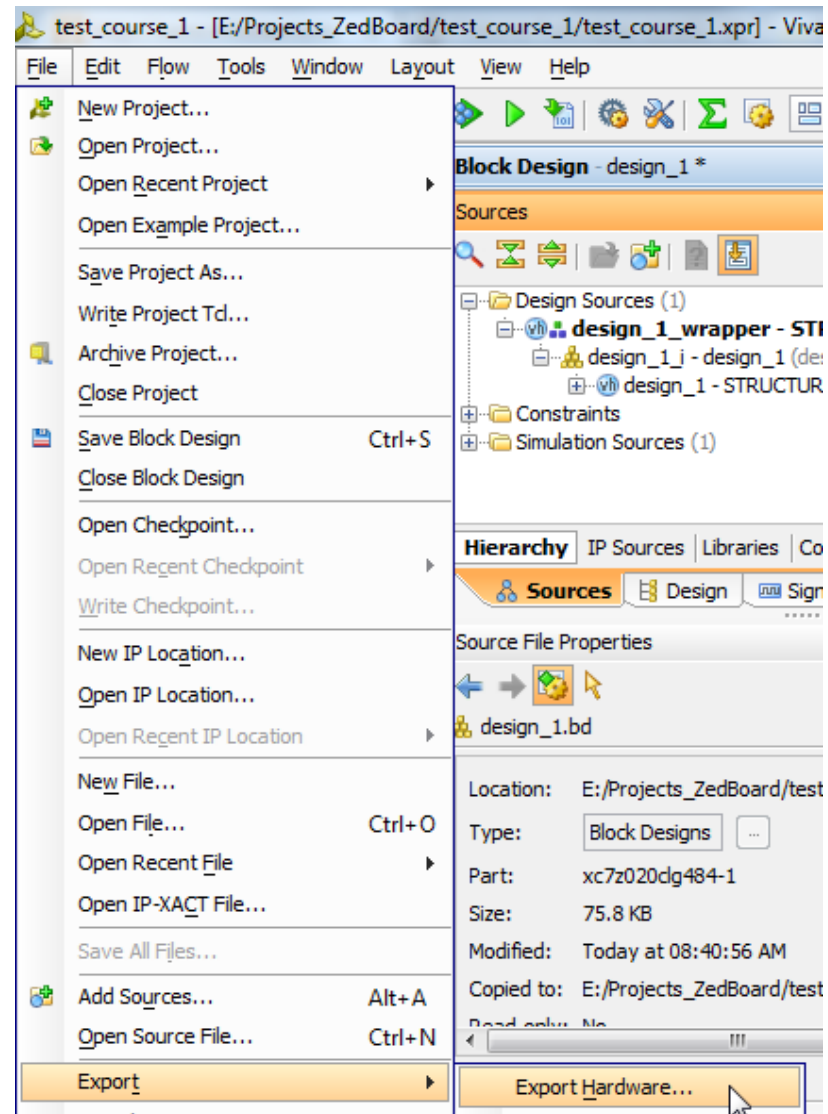


# Implementation Out-of-Date Message

---



# Exporting a Hardware Description



# Export Hardware Design to SDK

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

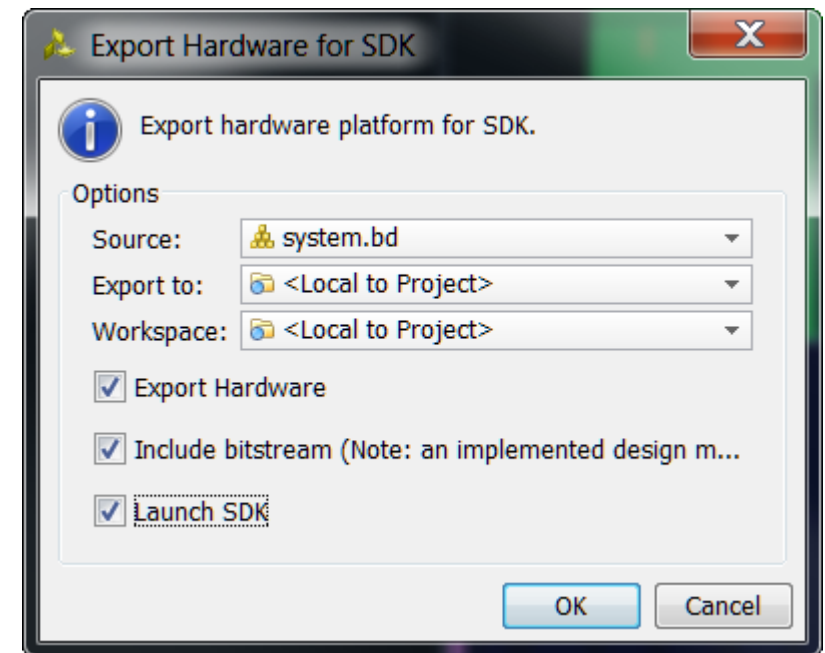
The design must be opened if a bitstream of the design is generated

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 <i>init</i> file
ps7_init.html	This <i>init</i> file describes the initialization data.

# *Vivado Design Suite*

## *Basic Static Timing Constraints*

---

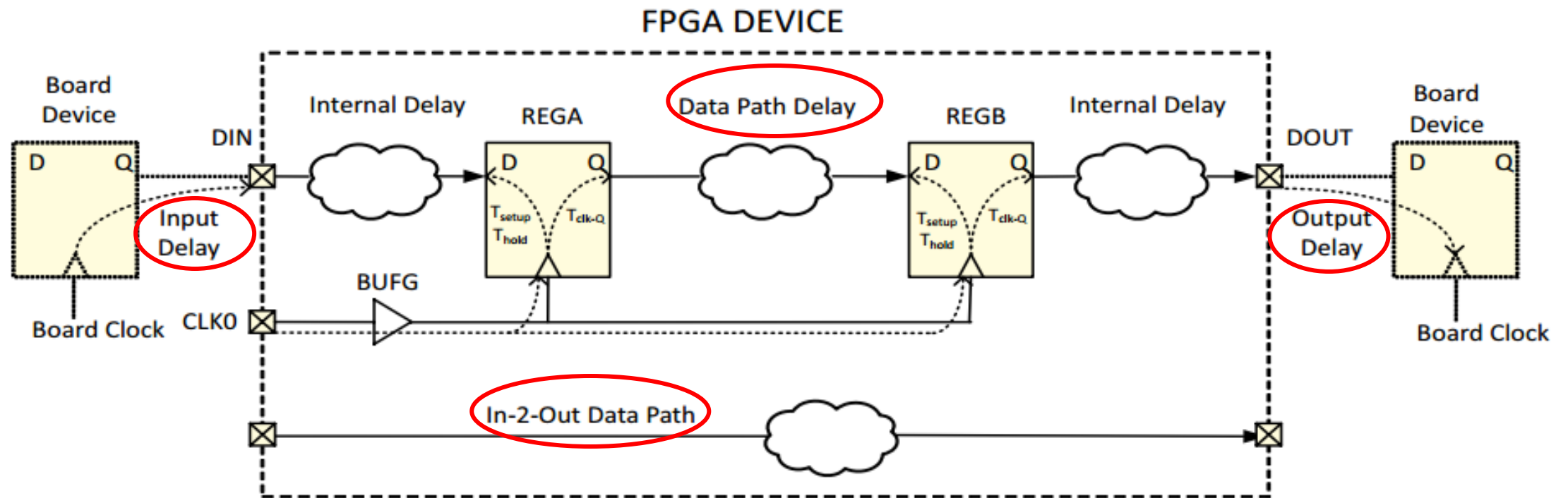
# Basic Timing Constraints

---

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

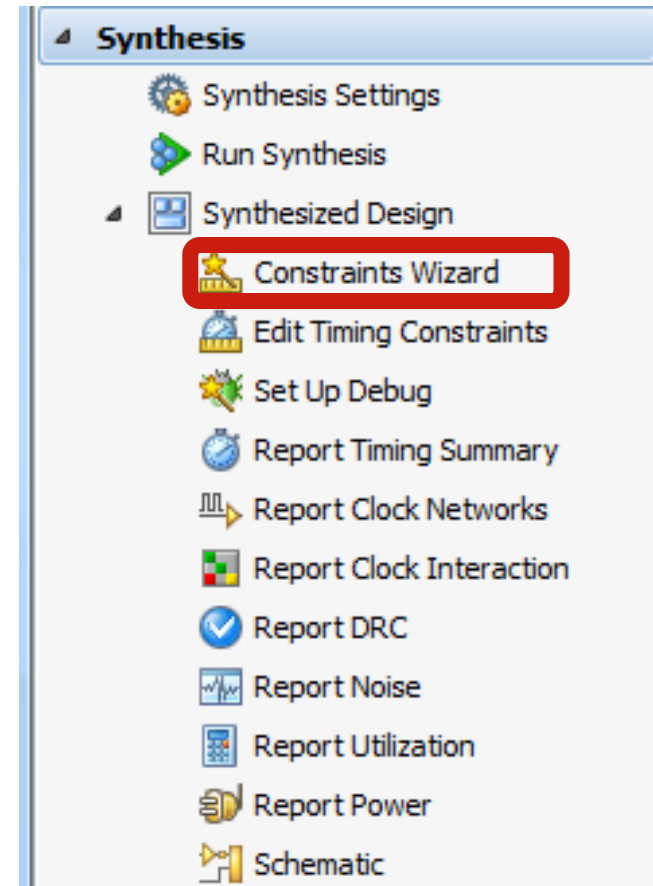
# Timing Paths Example



# Creating Basic Timing Constraints in Vivado IDE

---

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



# Clock Constraint Setting

**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}]
```

**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)

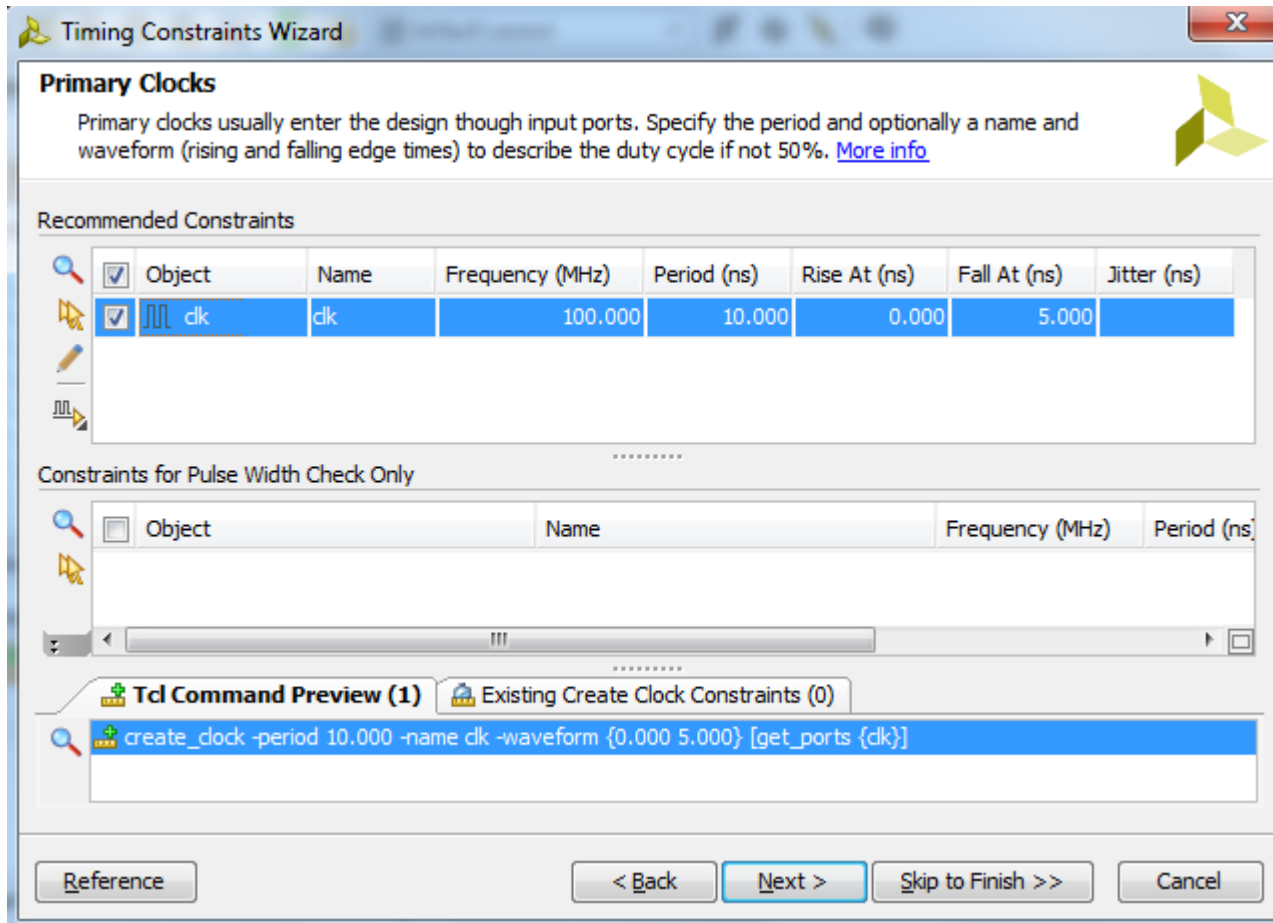
Fall at:  ns (optional)

Jitter:  ns (optional)

OK Cancel

Reference < Back Next > Skip to Finish >> Cancel

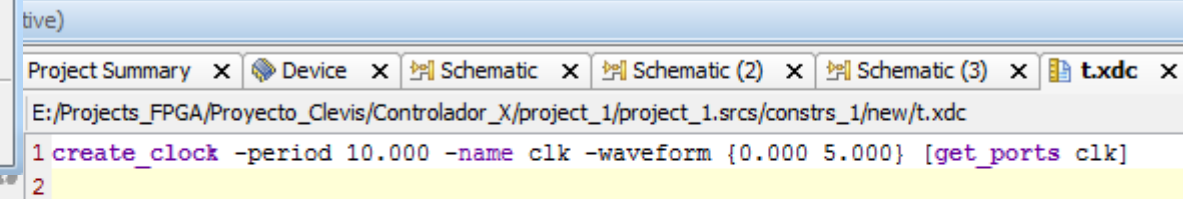
# Clock Constraint Setting



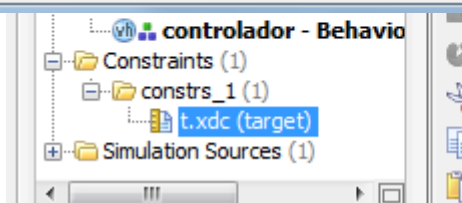
The Timing Constraints Wizard dialog box is shown. It has a title bar with a close button. The main content area is divided into several sections:

- Primary Clocks:** A text box explains that primary clocks enter the design through input ports and provides instructions on specifying period and waveform. A [More info](#) link is present.
- Recommended Constraints:** A table with columns: Object, Name, Frequency (MHz), Period (ns), Rise At (ns), Fall At (ns), and Jitter (ns). One row is selected, showing a clock named 'clk' with a frequency of 100.000 MHz, a period of 10.000 ns, and a 50% duty cycle (0.000 ns rise, 5.000 ns fall).
- Constraints for Pulse Width Check Only:** An empty table with columns: Object, Name, Frequency (MHz), and Period (ns).
- Tcl Command Preview (1):** A text area showing the command: `create_clock -period 10.000 -name clk -waveform {0.000 5.000} [get_ports {clk}]`.

At the bottom, there are navigation buttons: Reference, < Back, Next >, Skip to Finish >>, and Cancel.



The Tcl Command Editor window shows the command from the wizard being entered into a text area. The command is: `1 create_clock -period 10.000 -name clk -waveform {0.000 5.000} [get_ports clk]`. The line number 2 is visible below the command.



The Project Hierarchy view shows the project structure. The 'controlador - Behavior' folder is expanded, showing 'Constraints (1)' which contains 'constrs\_1 (1)'. Inside 'constrs\_1 (1)', the file 't.xdc (target)' is selected and highlighted in blue.





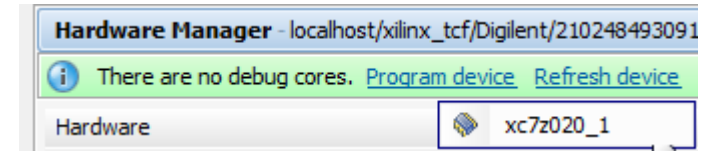
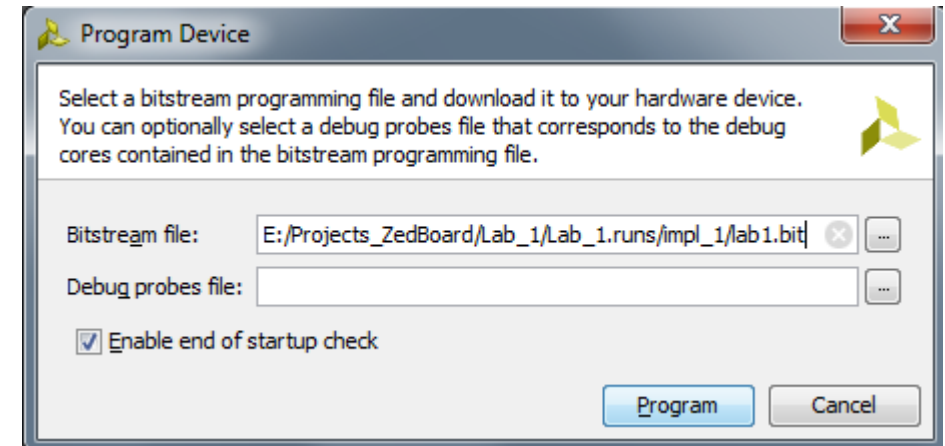
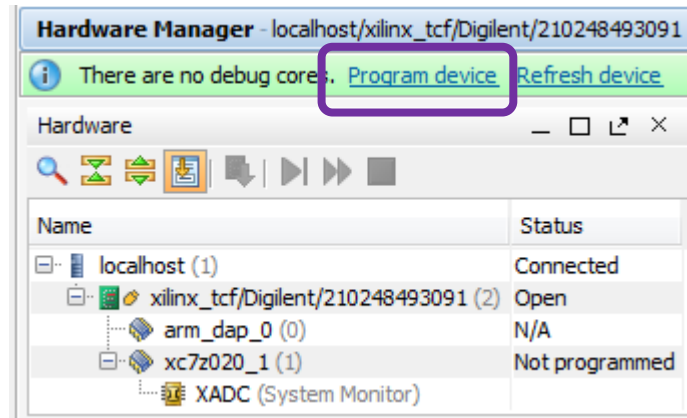
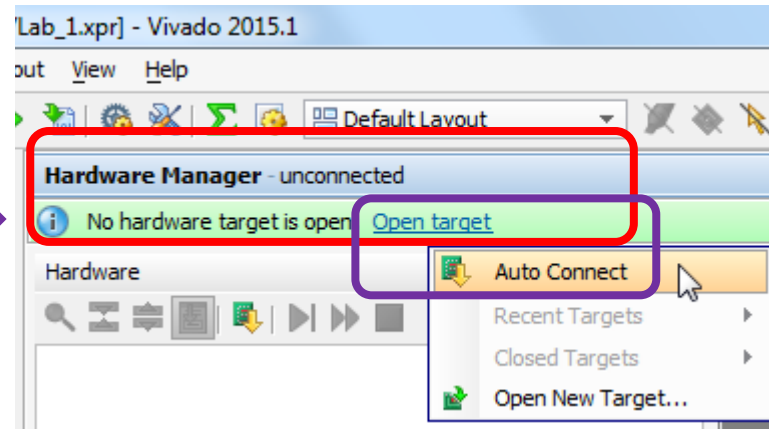
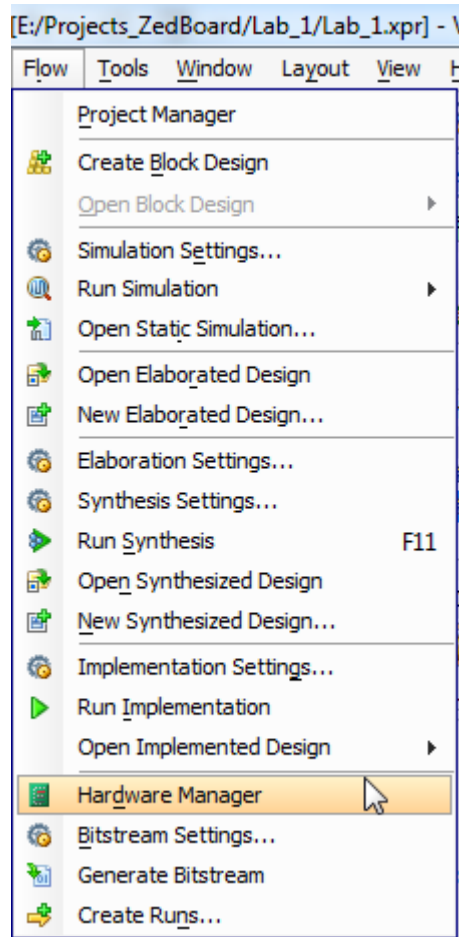
# Clock Network Report and Visualization

The image displays the Vivado IDE interface for generating and visualizing a clock network report. On the left, the 'Implementation' sidebar is visible, with the 'Report Clock Networks' option 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 hierarchical tree of clock networks under 'Clock Networks - network\_1'. The top-level node, 'clk (100.00 MHz) (drives 48 loads)', is highlighted with a red box. Below it, the tree lists various components: 'I (clk\_IBUF\_inst/I)', 'O (clk\_IBUF\_inst/O)', 'I (clk\_IBUF\_BUFG\_inst/I)', and 'O (clk\_IBUF\_BUFG\_inst/O)'. At the bottom, a 'Clock Networks' visualization window shows a detailed network diagram with a central vertical line labeled 'X1Y1' and multiple horizontal lines representing signal paths. A white arrow points from the 'X1Y1' label to the network diagram.

*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
<b>xc7z020_1 (1)</b>	<b>Programmed</b>
XADC (System Monitor)	

Blue  
"Done"  
LED

# Clocking Resources: MMCM and PLL

Up to 24 CMTs per device

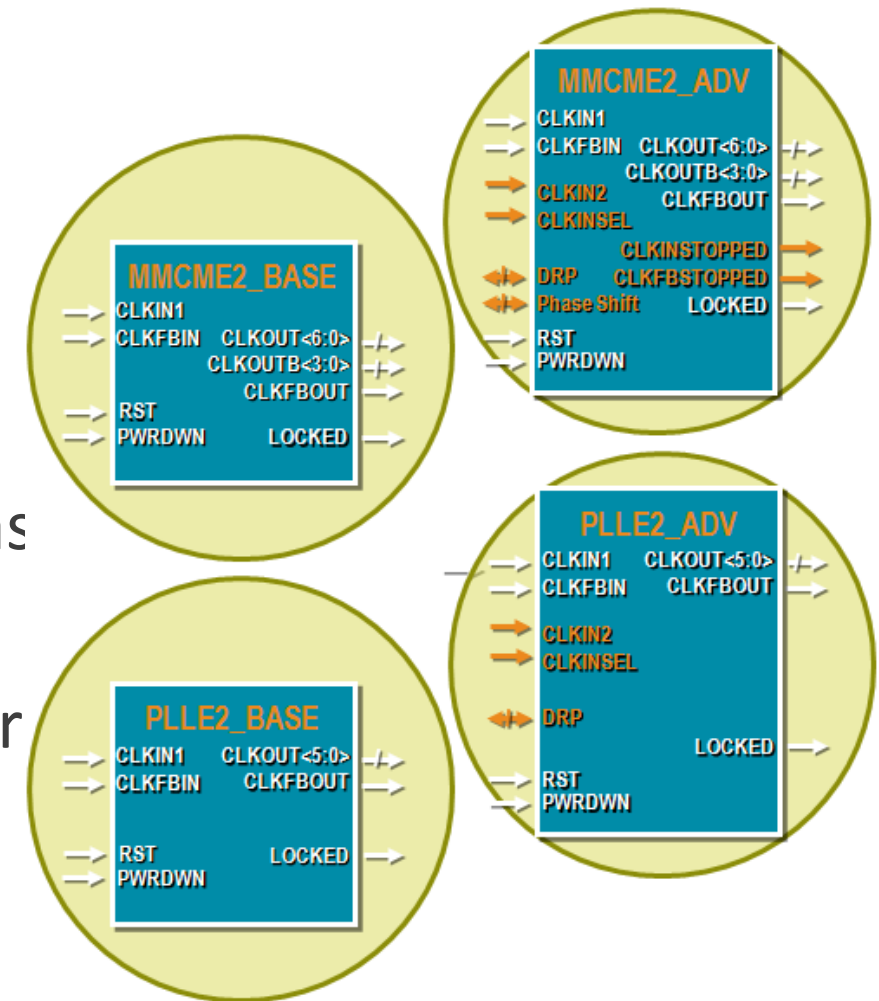
One MMCM and one PLL per CMT

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

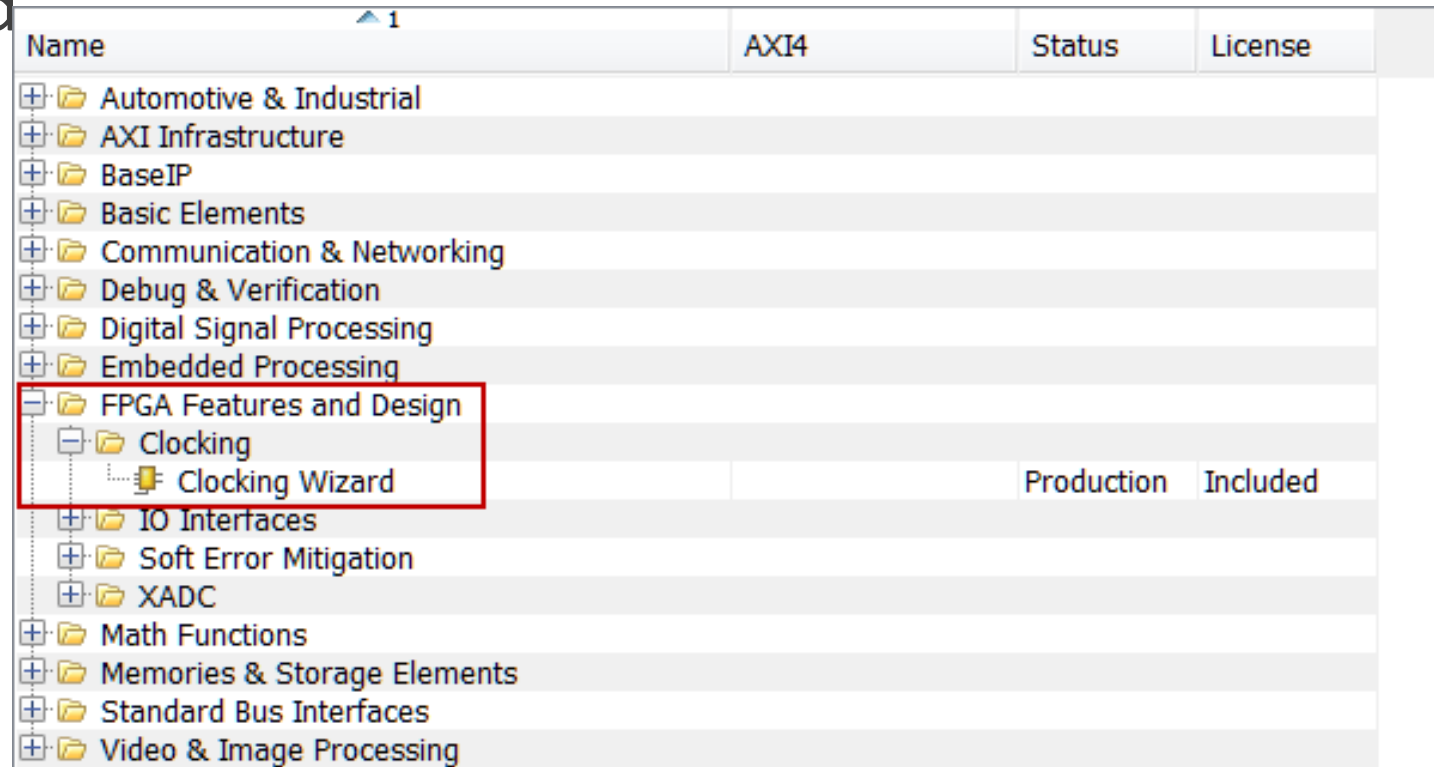
# Invoking Clocking Wizard

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 IP categories. The 'FPGA Features and Design' folder is expanded, and the 'Clocking' sub-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

Customize IP

Clocking Wizard (5.0)

Documentation IP Location Switch to Defaults

Component Name: clk\_core

Clocking Options: Output Clocks MMCM Settings Port Renaming Summary

The phase is calculated relative to the active input clock.

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"/>

USE CLOCK SEQUENCING

Output Clock	Sequence Number
clk_out1	1
clk_out2	1
clk_out3	1
clk_out4	1
clk_out5	1
clk_out6	1
clk_out7	1

Clocking Feedback

Source

- Automatic Control On-Chip
- Automatic Control Off-Chip
- User-Controlled On-Chip
- User-Controlled Off-Chip

Signaling

- Single-ended
- Differential

Enable Optional Inputs / Outputs

- reset
- power\_down
- input\_clk\_stopped
- locked
- clkfbstopped

Reset Type

- Active High
- Active Low

OK Cancel

# 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 features three main sections: 'Input Clock', 'Output Clock', and 'Optional Port Names'. The 'Input Clock' section contains 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 contains 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

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

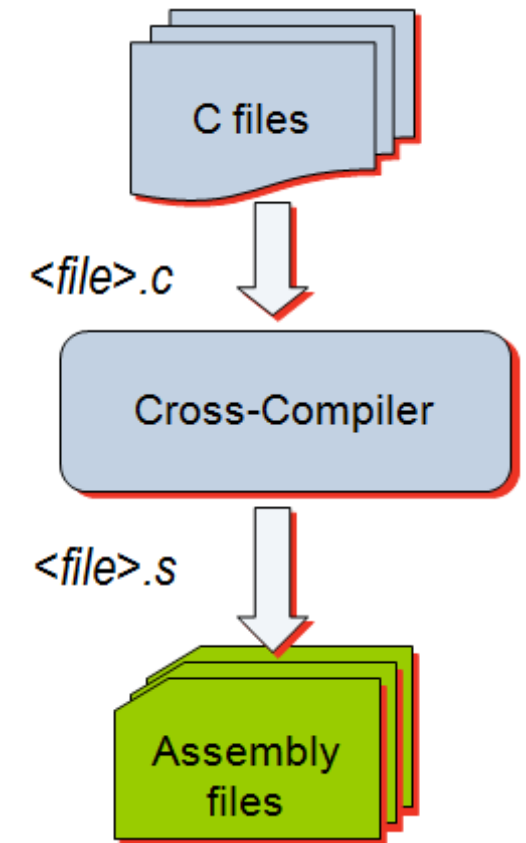
GCC translates C source code into assembly language

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

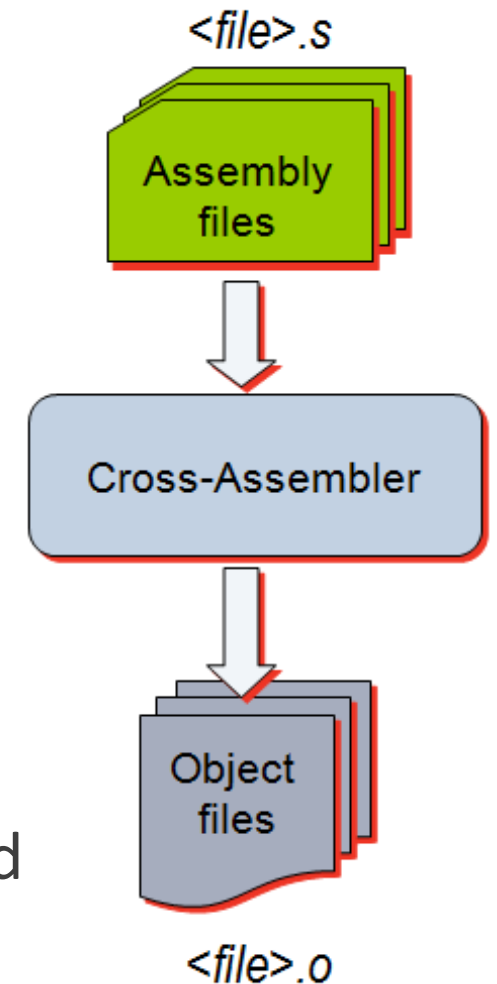
- File extension: .o

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`



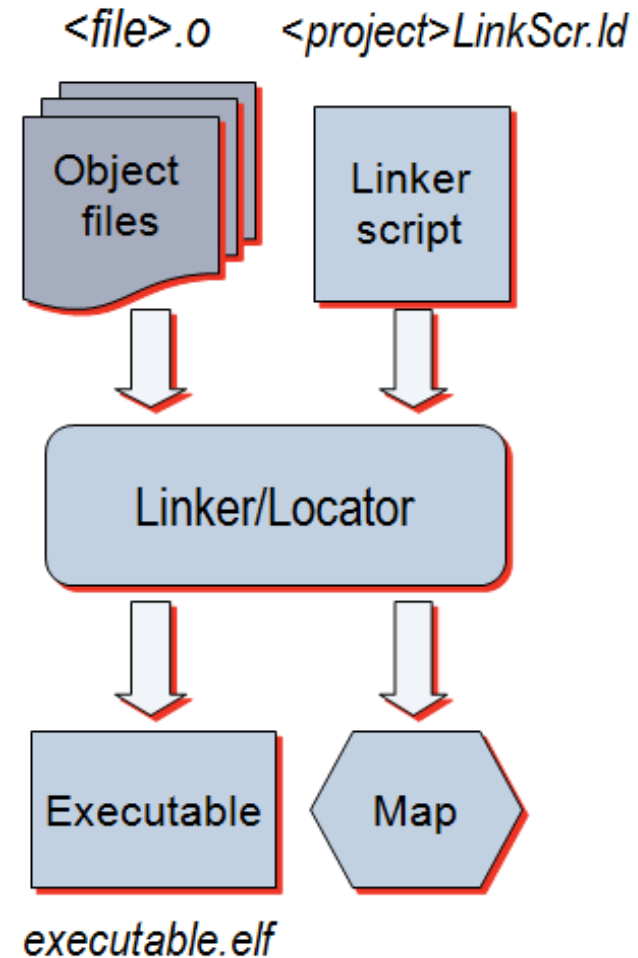
# GNU Tools: Linker (LD)

## Inputs

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

## Outputs

- Executable image (ELF)
- Map file

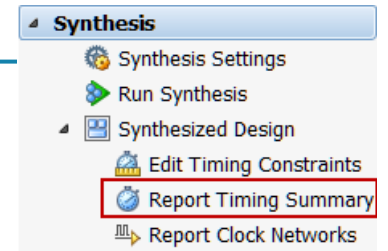


# Timing Reports

# 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

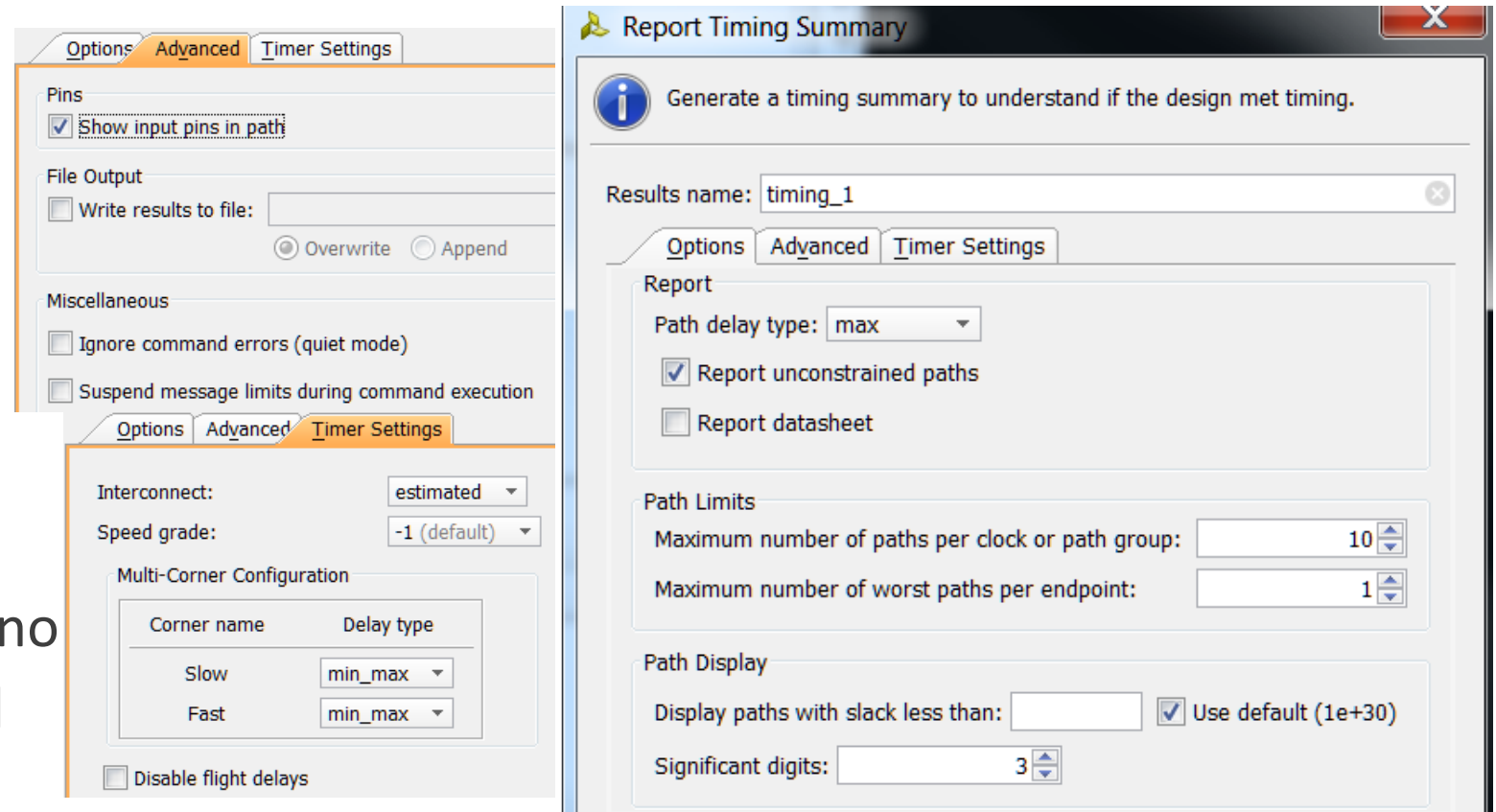
- Maximum number of paths

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

### Design Timing Summary

<b>Setup</b>	<b>Hold</b>	<b>Pulse Width</b>
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)
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

### 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