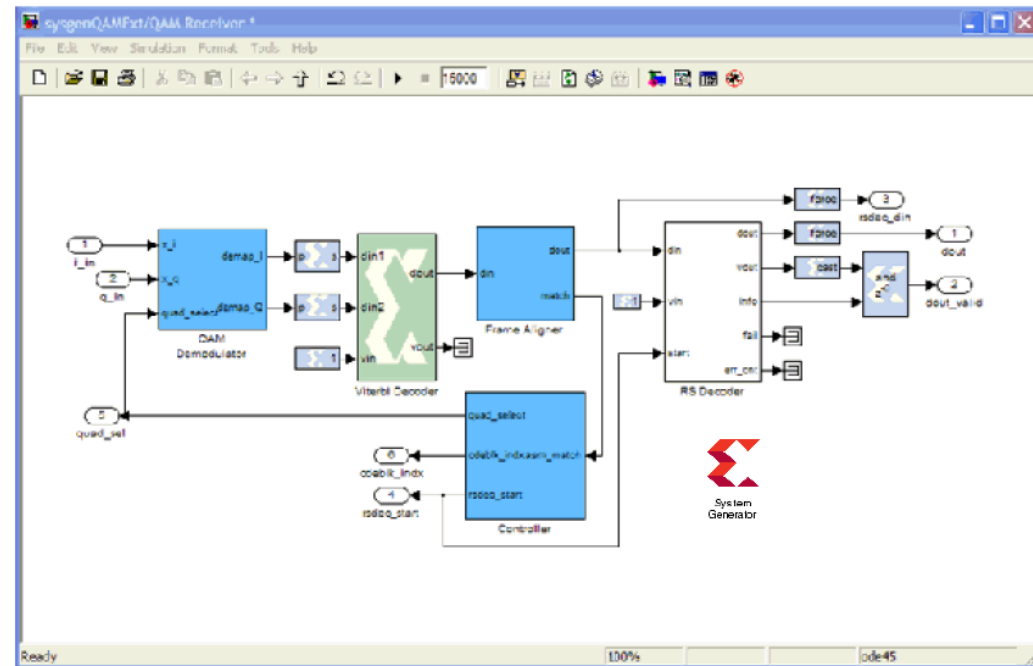


Lab 0: Introduction Model Based DSP Designs using High Level Synthesize and Vivado

ELEC 4601

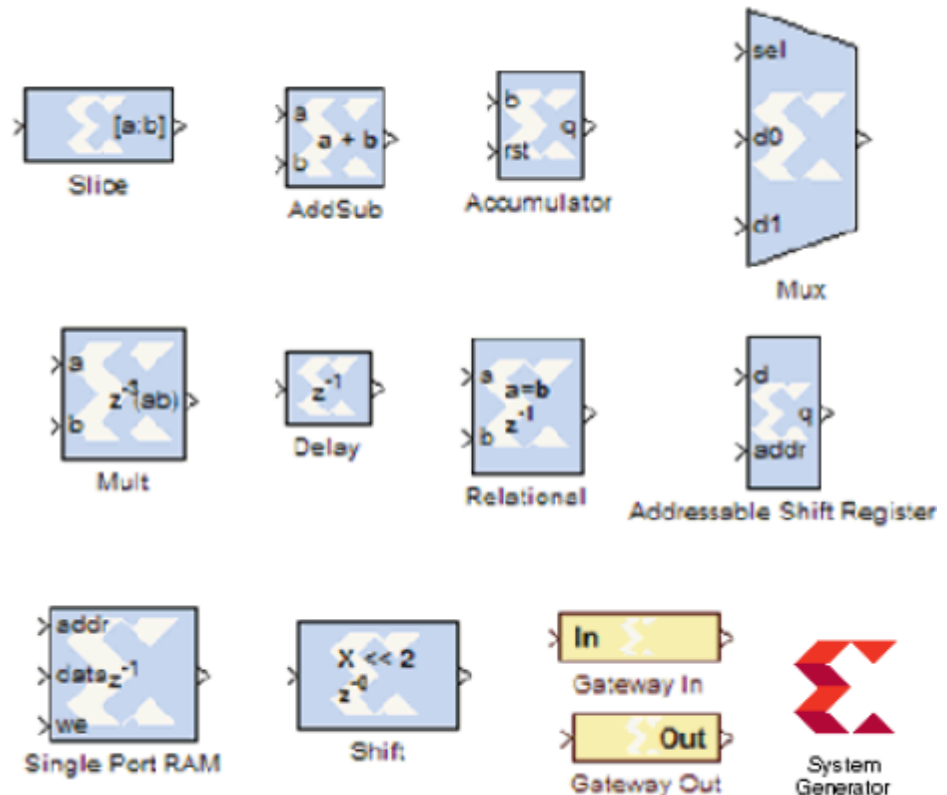
System Generator

- System Generator is a DSP design tool from Xilinx that enables the use of the MathWorks model-based Simulink® design environment for FPGA design.
- Previous experience with Xilinx FPGAs or RTL design methodologies are not required when using System Generator.
- Designs are captured in the DSP friendly Simulink modeling environment using a Xilinx specific blockset.



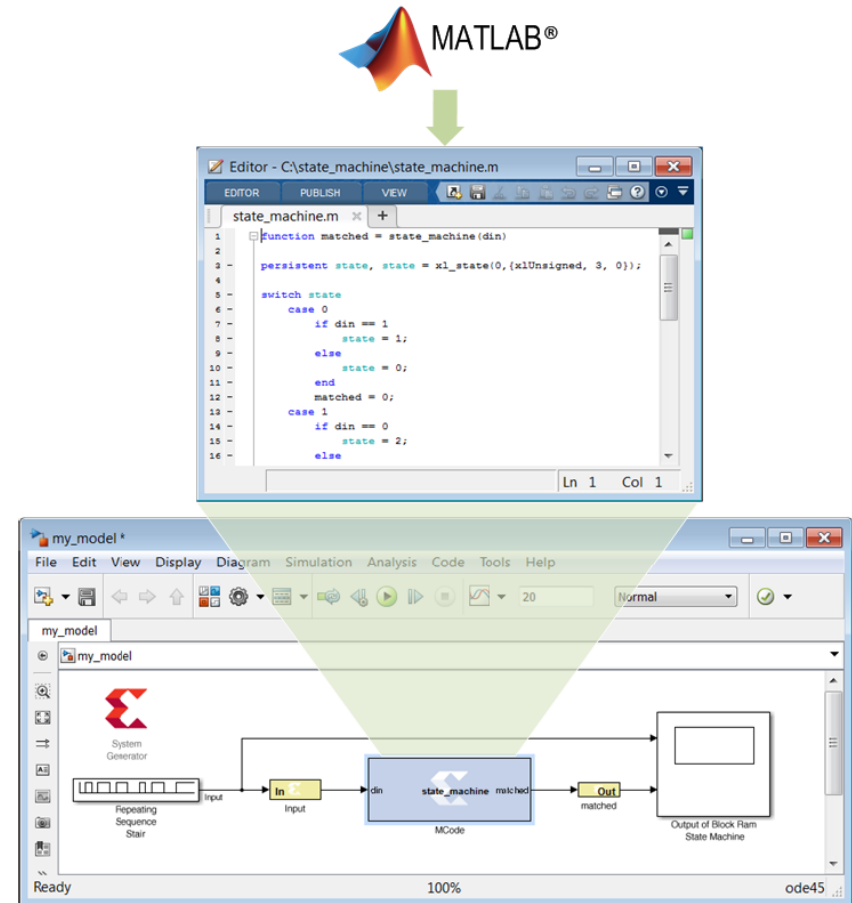
Xilinx DSP Block Set

- Over 90 DSP building blocks are provided in the Xilinx DSP blockset for Simulink. These blocks include the common DSP building blocks such as adders, multipliers and registers.
- Also included are a set of complex DSP building blocks such as forward error correction blocks, FFTs, filters and memories.
- These blocks leverage the Xilinx IP core generators to deliver optimized results for the selected device.



System Generator Supports

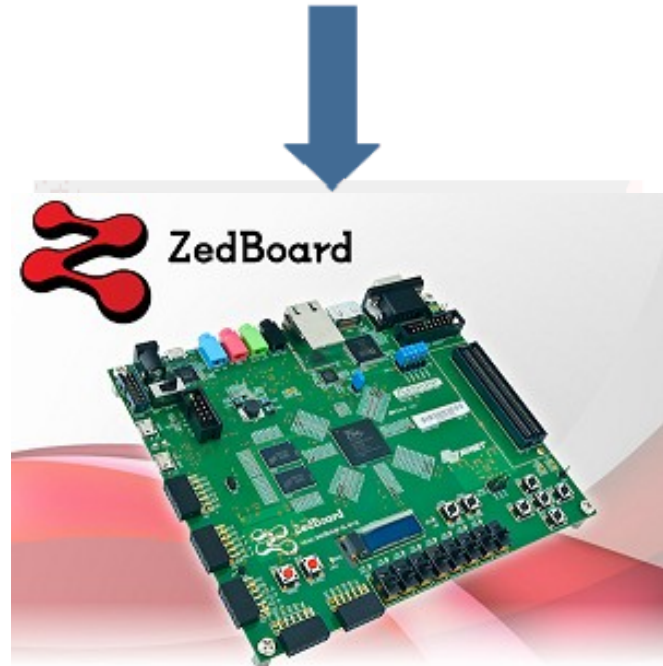
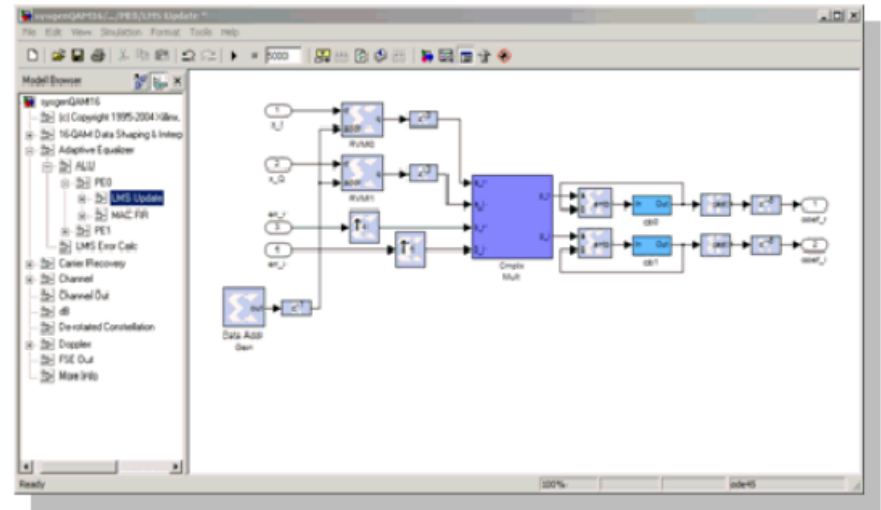
- Pure HDL
- VHDL (Ex: blackbox)
- Model based design
- M code based functional design
- C/C++ based functional design



Included in System Generator is an MCode block that allows the use of non-algorithmic MATLAB for the modeling and implementation of simple control operations.

Hardware Co-Simulation

- System Generator provides accelerated simulation through hardware co-simulation.
- System Generator will automatically create a hardware simulation token for a design captured in the Xilinx DSP blockset that will run on supported hardware platforms.
- **ZedBoard Zynq-7000 ARM/FPGA SoC** development board will be used for this course



Design Flows Using System Generator

1. Algorithm Exploration
2. Implementing Part of a Larger Design
3. Implementing a Complete Design
4. Compilation
5. H/W Co Simulation
6. Results Analysis (Resources, Timing ,Critical Paths etc)

Basics of System Level Modelling

- System Generator Blocksets
- Xilinx Commands that Facilitate Rapid Model Creation and Analysis
- Bit-True and Cycle-True Modeling
- Timing and Clocking
- Synchronization Mechanisms
- Block Masks and Parameter Passing

System Generator Blocksets

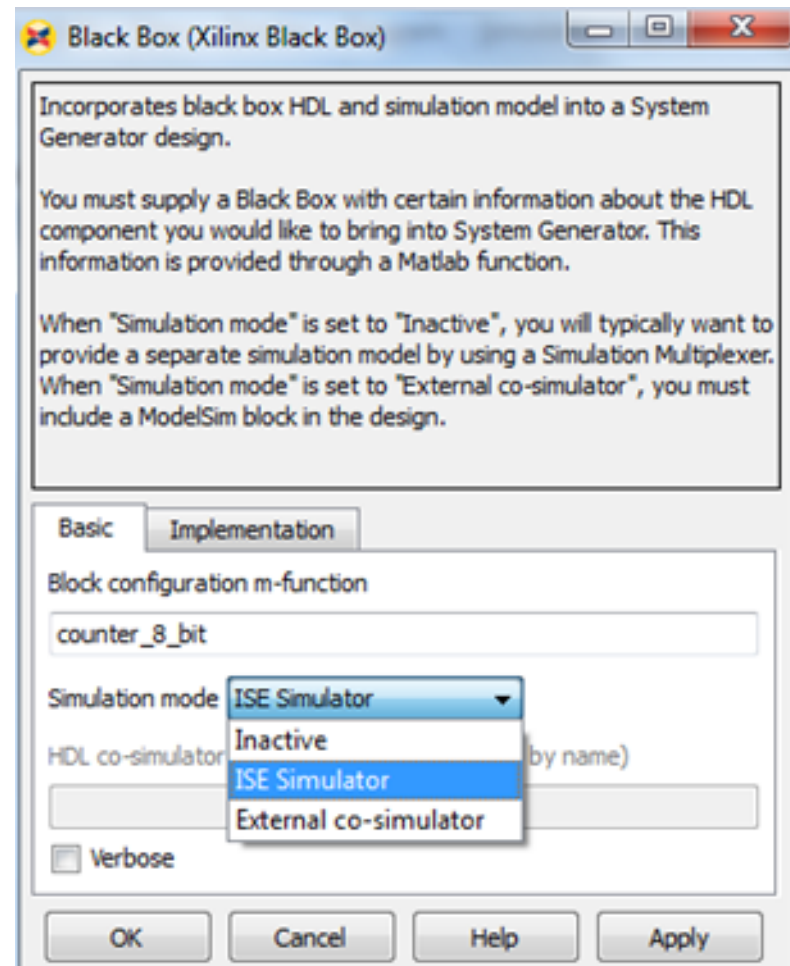
- System Generator blocksets are used like other Simulink blocksets.
- The blocks provide abstractions of mathematical, logic, memory, and DSP functions that can be used to build sophisticated signal processing (and other) systems.
- There are also blocks that provide interfaces to other software tools (e.g., FDATool, ModelSim) as well as the System Generator code generation software.
- System Generator blocks are *bit-accurate* and *cycle-accurate*.
 - Bit-accurate blocks produce values in Simulink that match corresponding values produced in hardware; cycle-accurate blocks produce corresponding values at corresponding times

Xilinx Blocksets

Library	Description
AXI4	Blocks with interfaces that conform to the AXI™4 specification
Basic Elements	Standard building blocks for digital logic
Communication	Forward error correction and modulator blocks, commonly used in digital communications systems
Control Logic	Blocks for control circuitry and state machines
Data Types	Blocks that convert data types (includes gateways)
DSP	Digital signal processing (DSP) blocks
Floating-Point	Blocks that support the Floating-Point data type
Index	Every block in the Xilinx Blockset
Math	Blocks that implement mathematical functions
Memory	Blocks that implement and access memories
Tools	"Utility" blocks, e.g., code generation (System Generator token), resource estimation, HDL co-simulation, etc.

Example: Black Box

- Each reference block has a description of its implementation and hardware resource requirements.

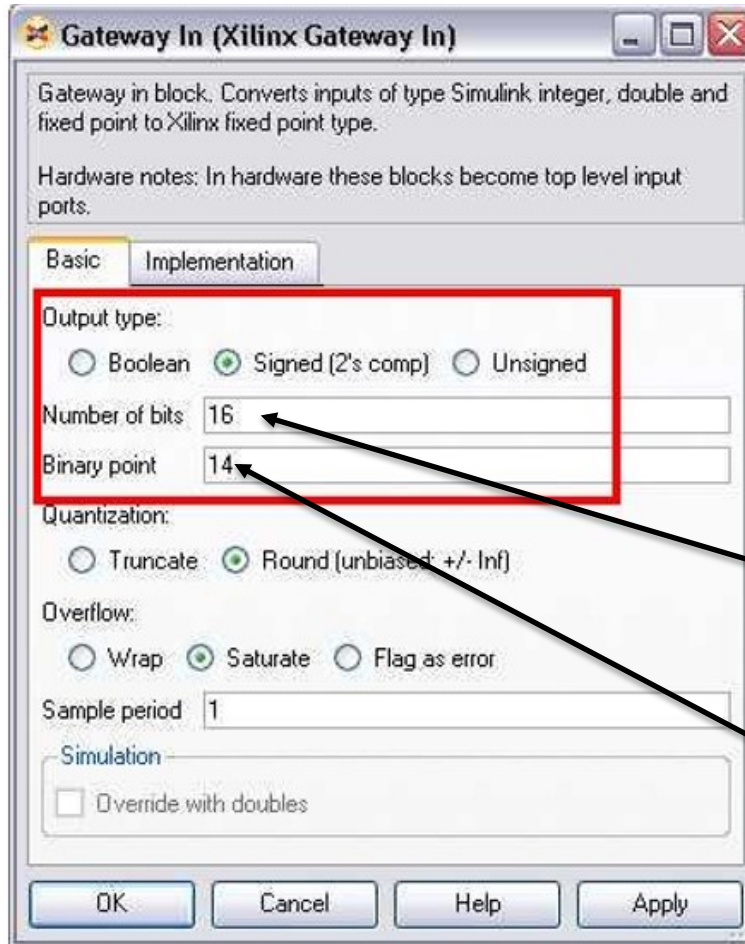


Signal Types

- In order to provide bit-accurate simulation of hardware, System Generator blocks operate on Boolean, floating-point, and arbitrary precision fixed-point values.
- By contrast, the fundamental scalar signal type in Simulink is double precision floating point.
- The connection between Xilinx blocks and non-Xilinx blocks is provided by *gateway blocks*.
- The **Gateway In** converts a double precision signal into a Xilinx signal, and the **Gateway Out** converts a Xilinx signal into double precision.
- Simulink continuous time signals must be sampled by the Gateway In block.

Note :Please use Fixed Point for the H/W Operations through out this course. Don't use recourse costly floating point operations.

Fixed Point Data



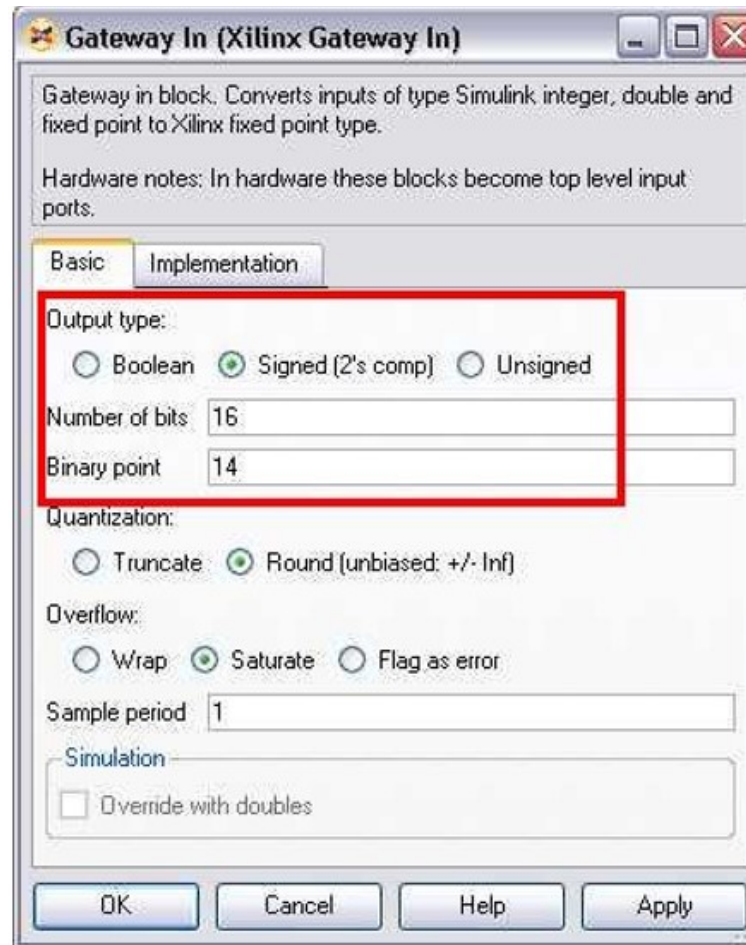
**Number of bits (W)
& Binary Points(D) define the
Fixed Point data**

**Pre define the W , D
values in the Matlab
workspace as variables
to use it here**

W

D

Fixed Point Data



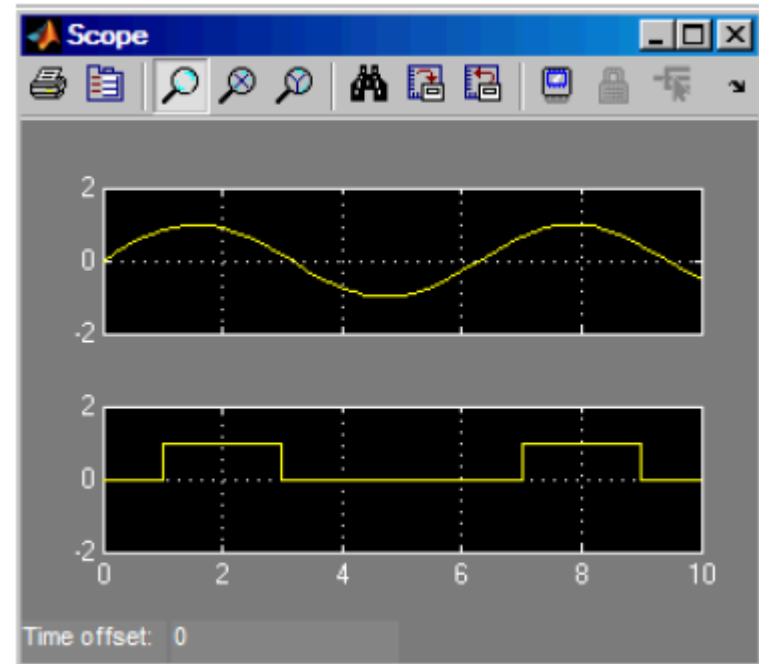
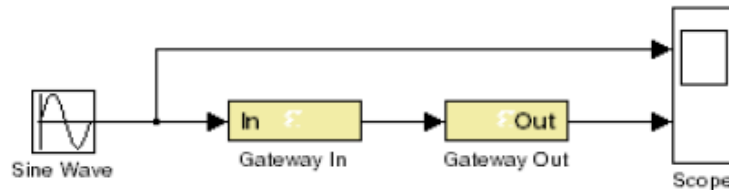
**Number of bits &
Binary Points
define the Fixed
Point data**

Timing and Clocking

- *Discrete Time Systems*
 - Designs in System Generator are discrete time systems. In other words, the signals and the blocks that produce them have associated sample rates.
 - A block's sample rate determines how often the block is awoken (allowing its state to be updated).
 - System Generator sets most sample rates automatically. A few blocks, however, set sample rates explicitly or implicitly.

Timing and Clocking

- A simple System Generator model illustrates the behavior of discrete time systems. Consider the model shown below. It contains a gateway that is driven by a Simulink source (Sine Wave), and a second gateway that drives a Simulink sink (Scope).



Automatic Code Generation

System Generator automatically compiles designs into low-level representations

Compiling and Simulating
Using the System Generator
Token

Describes how to use the System Generator token to compile designs into equivalent low-level HDL.

Compilation Results

Describes the low-level files System Generator produces when **HDL Netlist** is selected on the System Generator token and **Generate** is pushed.

Vivado Project

Describes the example project System Generator produces when **HDL Netlist** or **IP Catalog** is selected on the System Generator token and **Generate** is pushed.

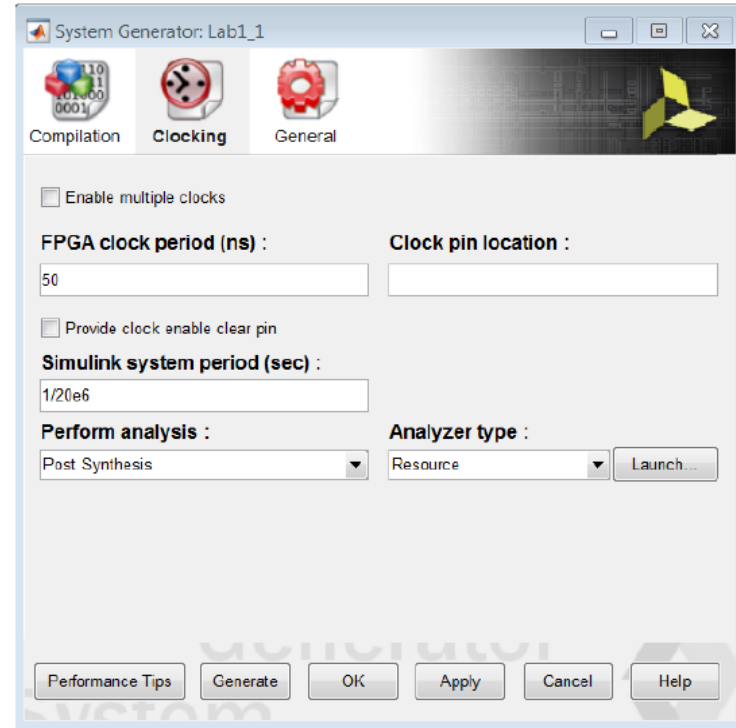
HDL Testbench

Describes the VHDL testbench that System Generator can produce.

Set Clock rate for FPGA

➤ If you want to set the clock rate of the FPGA and the Simulink sample period to 20 MHz.

1. Double-click the **System Generator** token to open the Properties Editor.
2. Select the **Clocking** tab.
 - a. Specify an **FPGA clock Period** of 50 ns (1/20 MHz).
 - b. Specify a **Simulink system period** of 1/20e6 seconds.

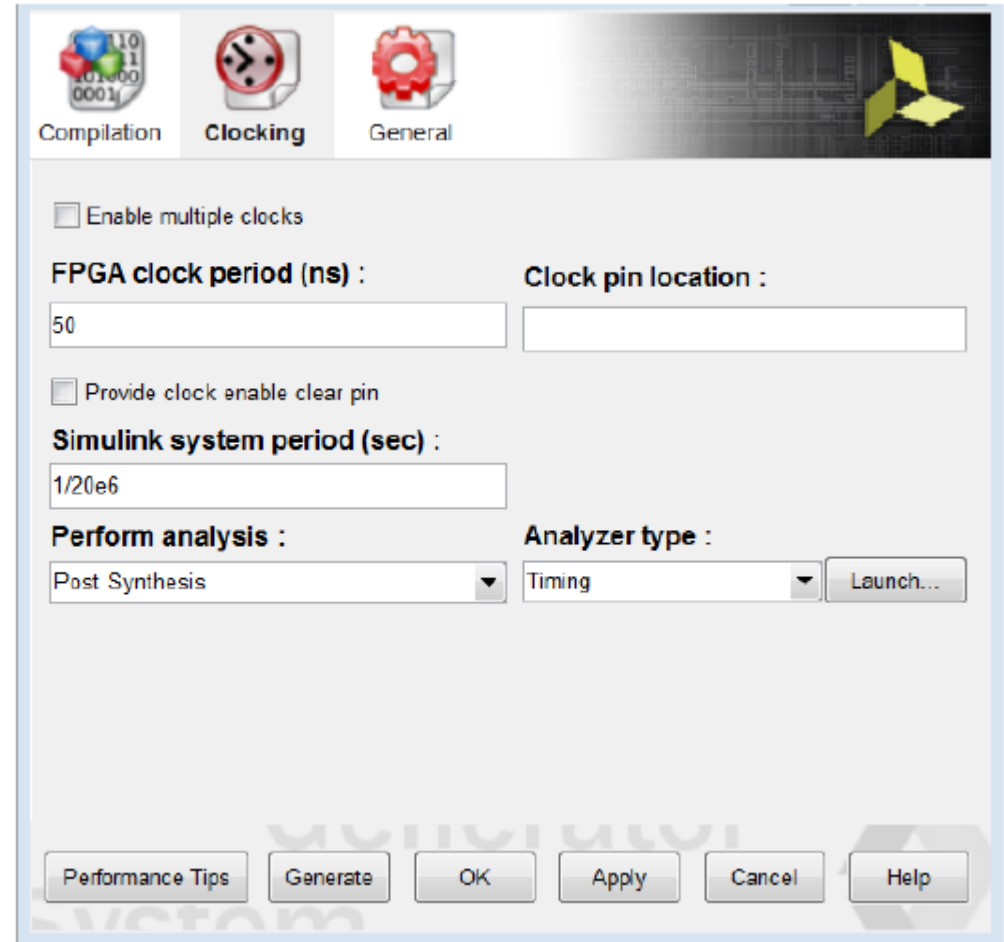


Note: **Simulink system period** and **FPGA clock period** define the scaling factor between time in a Simulink simulation, and time in the actual hardware implementation.

The Simulink system period must be the greatest common divisor (gcd) of the sample periods that appear in the model, and the FPGA clock period is the period, in nanoseconds, of the system clock.

Timing Analysis in System Generator

1. From your Simulink project worksheet, select **Simulation > Run** or click the **Run** simulation button to simulate the design.
2. Double-click the **System Generator** token to open the Properties Editor.
3. Select the **Clocking** tab.
4. From the **Perform analysis** menu, select **Post Synthesis** and from **Analyzer type** menu select **Timing** as shown.



Timing Analysis in System Generator

→ In the System Generator token dialog box, click **Generate**.

When you generate, the following occurs:

- System Generator generates the required files for the selected compilation target. For timing analysis System Generator invokes Vivado in the background for the design project, and ***passes design timing constraints to Vivado***.
- Depending on your selection for **Perform Analysis (Post Synthesis or Post Implementation)**, the design runs in Vivado through synthesis or through implementation.
- After the Vivado tools run is completed, timing paths information is collected and saved in a specific file format from the Vivado timing database. At the end of the timing paths data collection the Vivado project is closes and control is passed to the MATLAB/System Generator process.

- System Generator processes the timing information and displays a Timing Analyzer table with timing paths information as shown below.

Timing Analyzer: Lab3

Post Synthesis Timing Paths: Clicking on a timing path highlights corresponding blocks in the model.

Violation type: Status: **FAILED**

	Slack (ns)	Delay (ns)	gic Delay (ns)	ing Delay (ns)	Levels of Logic	Source	Destination	Source Clock	Destination Clock
1	-0.818	2.806	2.365	0.441	13	Lab3/sub...	Lab3/sub...	clk	clk
2	0.687	1.3	0.963	0.337	9	Lab3/add...	Lab3/add...	clk	clk
3	0.692	0.973	0.539	0.434	0	Lab3/sub...	Lab3/sub...	clk	clk
4	0.812	1.175	0.684	0.491	3	Lab3/add...	Lab3/add...	clk	clk
5	0.827	1.158	0.752	0.406	3	Lab3/add...	Lab3/add...	clk	clk
6	0.837	0.65	0.216	0.434	0	Lab3/De1...	Lab3/sub...	clk	clk
7	0.845	0.902	0.335	0.567	1	Lab3/De1...	Lab3/Reg...	clk	clk
8	1.058	0.962	0.962	0	0	Lab3/De1...	Lab3/De1...	clk	clk
9	1.36	0.484	0.232	0.252	0	Lab3/De1...	Lab3/De1...	clk	clk

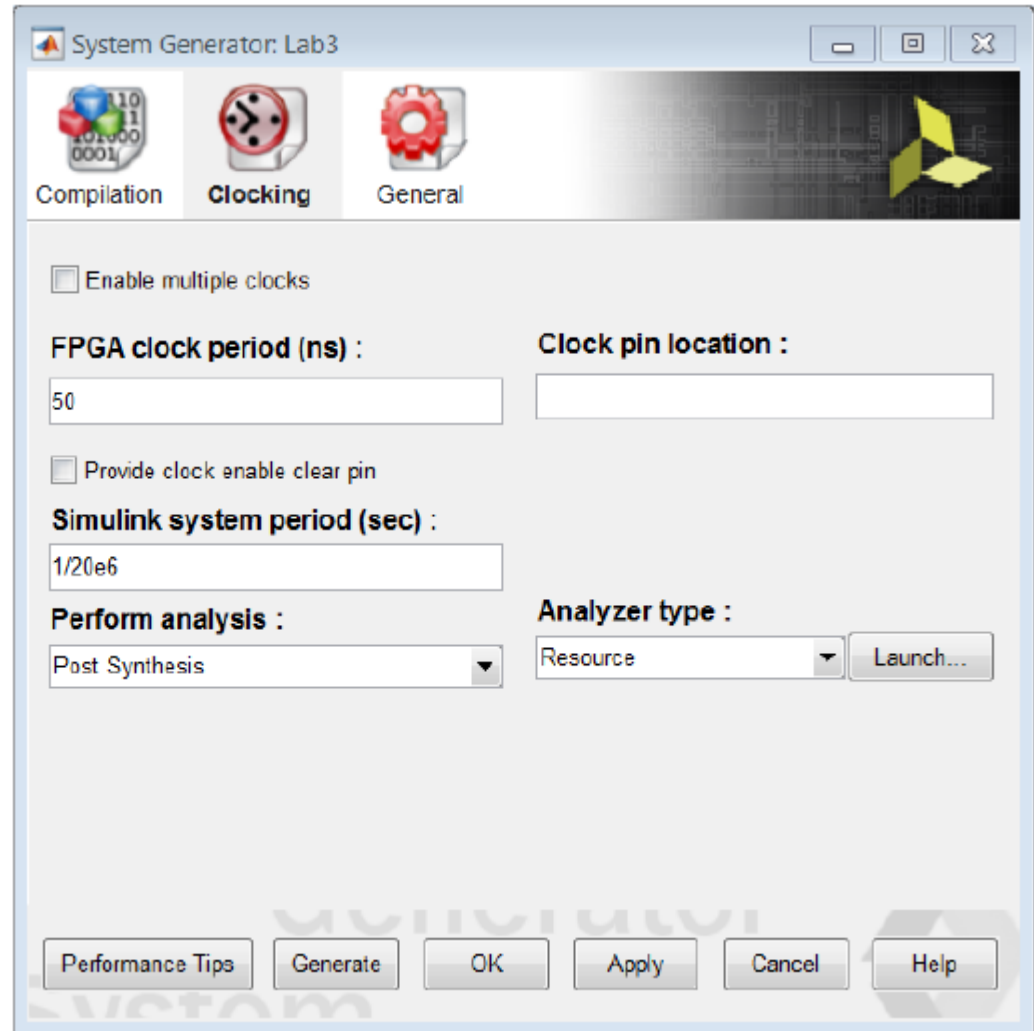
OK Help

- In the timing analyzer table:
 - Paths with lowest slack values display, with the worst Slack at the top and increasing slack below.
 - Paths with timing violations have a negative slack and display in red.

- Cross probe from the Timing Analyzer table to the Simulink model by clicking any path in the Timing Analyzer table, which highlights the corresponding System Generator blocks in the model.
- This allows you to troubleshoot timing violations by analyzing the path on which they occur.
- By inserting some registers in the combinational path may give better timing results and may help **overcome timing violations** if any. This can be done by changing latency of the combinational blocks as well.
- After modifying the design to have no timing violations in the design generate again to launch timing Analyzer table. If the status is pass at the top-right corner of the timing Analyzer table, it indicates there are no timing violated paths in the design.
- If you close the Timing Analyzer and sometime later you may want to relaunch the Timing Analyzer table using the existing timing analyzer results for the model. A **Launch** button is provided under the **Clocking** tab of the System Generator token dialog box.
 - This will only work if you already ran timing analysis on the Simulink model.

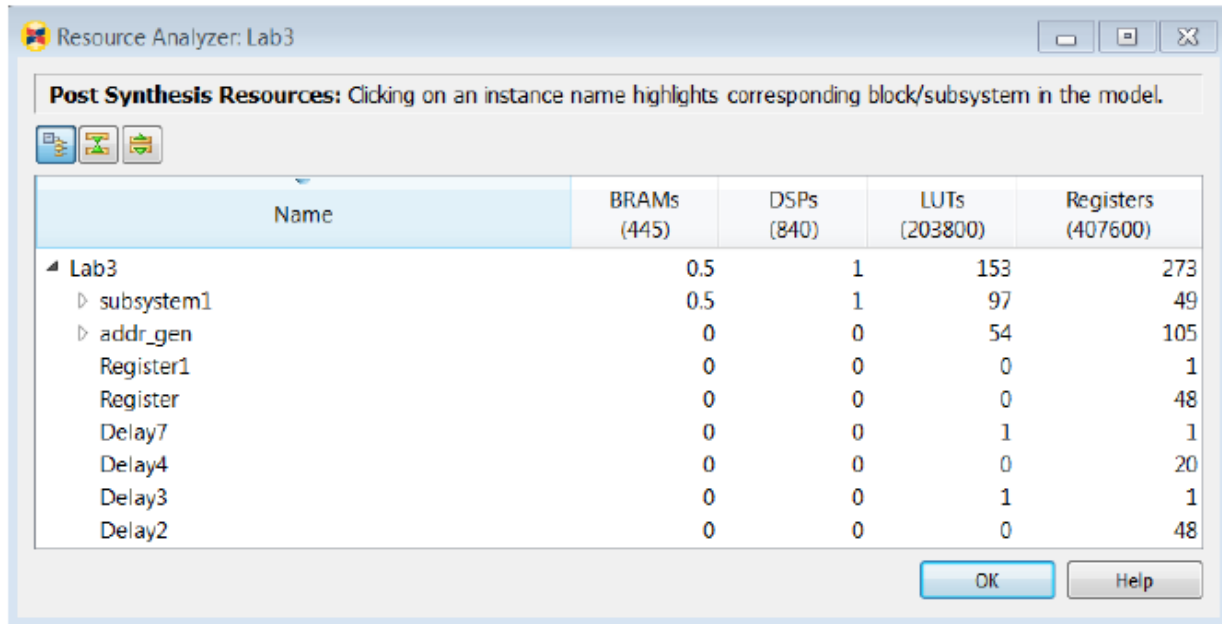
Resource Analysis in System Generator

- Double-click the **System Generator** token in the Simulink model. Make sure that the **part** is specified and **Compilation** is set to **Hardware Co-Simulation** compilation target.
- In the Clocking tab, set the Perform Analysis field to Post Synthesis and Analyzer type field to Resource.
- In the System Generator token dialog box, click **Generate**.



Resource Analysis in System Generator

- System Generator processes the resource utilization data and displays a **Resource Analyzer table** with resource utilization information.



The screenshot shows a window titled "Resource Analyzer: Lab3". Below the title bar is a text box that says "Post Synthesis Resources: Clicking on an instance name highlights corresponding block/subsystem in the model." Below this are three icons: a folder, a circuit diagram, and a list. The main part of the window is a table with the following data:

Name	BRAMs (445)	DSPs (840)	LUTs (203800)	Registers (407600)
Lab3	0.5	1	153	273
subsystem1	0.5	1	97	49
addr_gen	0	0	54	105
Register1	0	0	0	1
Register	0	0	0	48
Delay7	0	0	1	1
Delay4	0	0	0	20
Delay3	0	0	1	1
Delay2	0	0	0	48

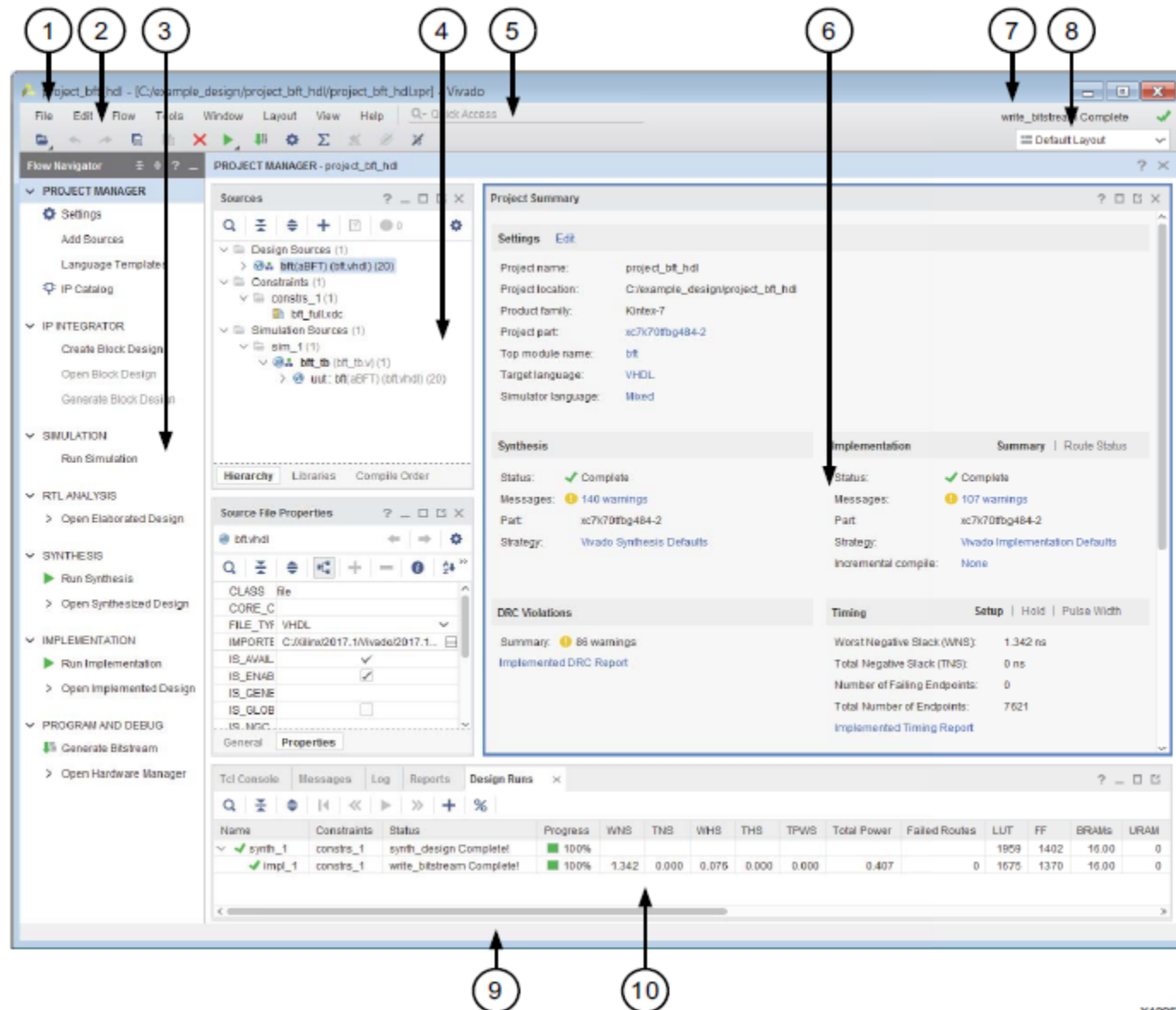
At the bottom right of the table are "OK" and "Help" buttons.

- Each column heading (for example, **BRAMs**, **DSPs**, or **LUTs**) in the table shows the total number of each type of resources available in the Xilinx device for which you are targeting your design.
- The rest of the table displays a hierarchical listing of each subsystem and block in the design, with the count of these resource types.

Vivado Integrated Design Environment

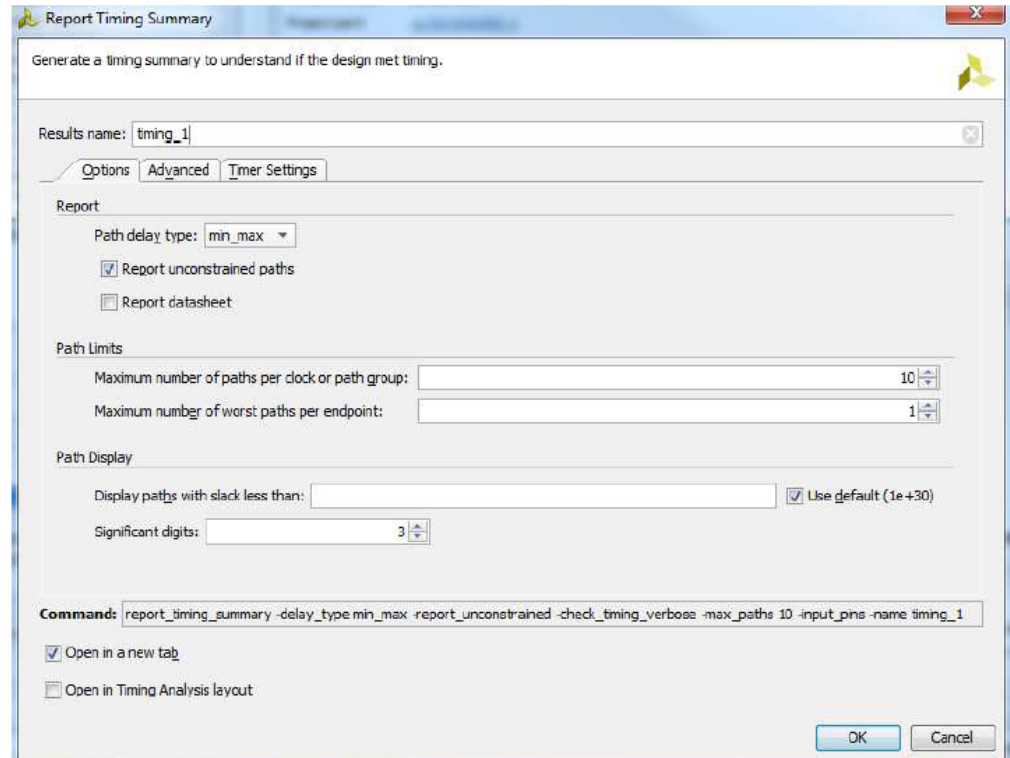
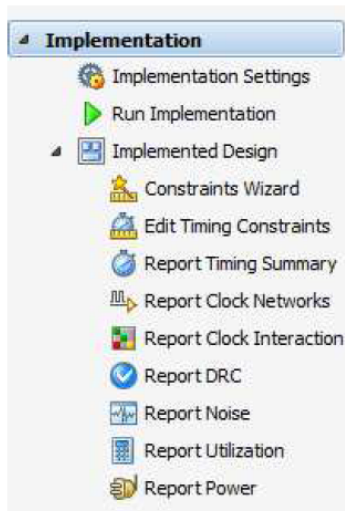
- The **Vivado Integrated Design Environment (IDE)** provides an intuitive graphical user interface (GUI) with powerful features.
- Analysis and constraint assignment is enabled throughout the entire design process. For example,
 - can run timing or power estimations after synthesis, placement, or routing.
- Algorithms delivered by the Vivado IDE includes:
 - Register transfer level (RTL) design in VHDL, Verilog, and System Verilog
 - Intellectual property (IP) integration for cores
 - Behavioral, functional, and timing simulation with Vivado simulator
 - Vivado synthesis
 - Vivado implementation for place and route
 - Vivado power analysis
 - Static timing analysis
 - High-level floorplanning
 - Detailed placement and routing modification
 - Bitstream generation

Vivado IDE Viewing Environment



Timing Analysis in Vivado

- After Synthesizing and implementing the design in **Vivado IDE**, to start a new timing summary report select **Implementation > Report Timing Summary**
- A new window will pop up that shows all the option for timing summary. Do not change anything and click OK to select the default options.



Timing Analysis in Vivado

- A new timing summary will appear that will show you timing information. Click on the “**Worst Negative Slack (WNS)**”

Timing - Timing Summary - timing_1

Design Timing Summary

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): <u>13.500 ns</u>	Worst Hold Slack (WHS): <u>0.075 ns</u>	Worst Pulse Width Slack (WPWS): <u>9.020 ns</u>
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 1446	Total Number of Endpoints: 1446	Total Number of Endpoints: 727

All user specified timing constraints are met.

Timing Summary - timing_1 x

Td Console Messages Log Reports Design Runs Timing

- Clicking on the value of WNS will open the information for the 10 paths (Path 1 to 10) that have maximum delay in the design. Path 1 has the largest delay in the design, and thus is called the **critical path**.

Timing - Timing Summary - timing_1

Intra-Clock Paths - clk_fpga_0 - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Source Clock	Destination Clock	Except
Path 1	13.500	2		133 led_controller_i/processing_system7_0/inst/PS7_I/MAX1GP0ACLK led_contr...reg[7]/D		6.081	1.582	4.499	clk_fpga_0	clk_fpga_0	
Path 2	13.500	2		133 led_controller_i/processing_system7_0/inst/PS7_I/MAX1GP0ACLK led_contr...reg[6]/D		6.080	1.582	4.498	clk_fpga_0	clk_fpga_0	
Path 3	13.672	2		133 led_controller_i/processing_system7_0/inst/PS7_I/MAX1GP0ACLK led_contr...reg[7]/D		5.908	1.582	4.326	clk_fpga_0	clk_fpga_0	
Path 4	13.847	2		133 led_controller_i/processing_system7_0/inst/PS7_I/MAX1GP0ACLK led_contr...reg[10]/D		5.787	1.582	4.205	clk_fpga_0	clk_fpga_0	
Path 5	13.890	2		133 led_controller_i/processing_system7_0/inst/PS7_I/MAX1GP0ACLK led_contr...reg[7]/D		5.690	1.582	4.108	clk_fpga_0	clk_fpga_0	
Path 6	13.891	2		133 led_controller_i/processing_system7_0/inst/PS7_I/MAX1GP0ACLK led_contr...reg[11]/D		5.692	1.582	4.110	clk_fpga_0	clk_fpga_0	
Path 7	14.119	2		133 led_controller_i/processing_system7_0/inst/PS7_I/MAX1GP0ACLK led_contr...reg[11]/D		5.466	1.582	3.884	clk_fpga_0	clk_fpga_0	

Timing Summary - timing_1 x

Td Console Messages Log Reports Design Runs Timing

Timing Analysis in Vivado

- Double click on Path 1 to show you all the information for Path 1. This information includes

1. Summary
2. Source Clock Path
3. Data Path
4. Destination Clock Path

Summary

Name	Path 1			
Slack	13.500ns			
Source	led_controller_i/processing_system7_0/inst/PS7_i/MAXIGP0ACLK (rising edge-triggered cell PS7 clocked by clk_fpga_0 {rise@0.000ns fall@10.000ns period=20.000ns})			
Destination	led_controller_i/led_controller_0/U0/led_controller_v1_0_S00_AXI_inst/slv_reg3_reg[7]/D (rising edge-triggered cell FDRE clocked by clk_fpga_0 {rise@0.000ns fall@10.000ns})			
Path Group	clk_fpga_0			
Path Type	Setup (Max at Slow Process Corner)			
Requirement	20.000ns (clk_fpga_0 rise@20.000ns - clk_fpga_0 rise@0.000ns)			
Data Path Delay	6.081ns (logic 1.582ns (26.015%) route 4.499ns (73.985%))			
Logic Levels	2 (LUT2=1 LUT6=1)			
Clock Path Skew	-0.148ns			
Clock Uncertainty	0.302ns			

Source Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(dock clk_fpga_0 rise edge)	(r) 0.000	0.000		
PS7	(r) 0.000	0.000	Site: PS7_X0Y0	led_controller_i/processing_system7_0/inst/PS7_i/FCLKCLK[0]
net (fo=1, routed)		1.207		led_controller_i/processing_system7_0/inst/FCLK_CLK_unbuffered[0]
			Site: BUFGCTRL_X0Y0	led_controller_i/processing_system7_0/inst/buffer_fclk_clk_0/FCLK_CLK_0_BUFG/I
BUFG (Prop_bufg_1_O)	(r) 0.101	1.308	Site: BUFGCTRL_X0Y0	led_controller_i/processing_system7_0/inst/buffer_fclk_clk_0/FCLK_CLK_0_BUFG/O
net (fo=727, routed)		1.765		led_controller_i/processing_system7_0/inst/M_AXI_GP0_ACLK
			Site: PS7_X0Y0	led_controller_i/processing_system7_0/inst/PS7_i/MAXIGP0ACLK

Data Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
PS7 (Prop_ps7_MAXIGP0ACLK_MAXIGP0WVALID)	(r) 1.334	4.407	Site: PS7_X0Y0	led_controller_i/processing_system7_0/inst/PS7_i/MAXIGP0WVALID
net (fo=1, routed)		1.285		led_controller_i/led_controller_0/U0/led_controller_v1_0_S00_AXI_inst/s00_axi_wvalid
			Site: SLICE_X3Y48	led_controller_i/led_controller_0/U0/led_controller_v1_0_S00_AXI_inst/h_0_slv_reg0[3]_j_5/I1
LUT2 (Prop_lut2_j1_O)	(r) 0.124	5.815	Site: SLICE_X3Y48	led_controller_i/led_controller_0/U0/led_controller_v1_0_S00_AXI_inst/slv_reg0[3]_j_5/O
net (fo=133, routed)		3.215		led_controller_i/led_controller_0/U0/led_controller_v1_0_S00_AXI_inst/h_0_slv_reg0[3]_i_5
			Site: SLICE_X14Y43	led_controller_i/led_controller_0/U0/led_controller_v1_0_S00_AXI_inst/slv_reg3[7]_j_1/I2
LUT6 (Prop_lut6_j2_O)	(r) 0.124	9.154	Site: SLICE_X14Y43	led_controller_i/led_controller_0/U0/led_controller_v1_0_S00_AXI_inst/slv_reg3[7]_j_1/O
net (fo=1, routed)		0.000		led_controller_i/led_controller_0/U0/led_controller_v1_0_S00_AXI_inst/h_0_slv_reg3[7]_i_1
FDRE			Site: SLICE_X14Y43	led_controller_i/led_controller_0/U0/led_controller_v1_0_S00_AXI_inst/slv_reg3_reg[7]/D
Arrival Time			9.154	

Destination Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(dock clk_fpga_0 rise edge)	(r) 20.000	20.000		
PS7	(r) 0.000	20.000	Site: PS7_X0Y0	led_controller_i/processing_system7_0/inst/PS7_i/FCLKCLK[0]
net (fo=1, routed)		1.101		led_controller_i/processing_system7_0/inst/FCLK_CLK_unbuffered[0]
			Site: BUFGCTRL_X0Y0	led_controller_i/processing_system7_0/inst/buffer_fclk_clk_0/FCLK_CLK_0_BUFG/I
BUFG (Prop_bufg_1_O)	(r) 0.091	21.192	Site: BUFGCTRL_X0Y0	led_controller_i/processing_system7_0/inst/buffer_fclk_clk_0/FCLK_CLK_0_BUFG/O
net (fo=727, routed)		1.501		led_controller_i/led_controller_0/U0/led_controller_v1_0_S00_AXI_inst/s00_axi_ack
			Site: SLICE_X14Y43	led_controller_i/led_controller_0/U0/led_controller_v1_0_S00_AXI_inst/slv_reg3_reg[7]/C
clock pessimism		0.231	22.924	
clock uncertainty		-0.302	22.622	
FDRE (Setup_fdre_C_D)		0.032	22.654	led_controller_i/led_controller_0/U0/led_controller_v1_0_S00_AXI_inst/slv_reg3_reg[7]
Required Time			22.654	

Example

Invoke System Generator.

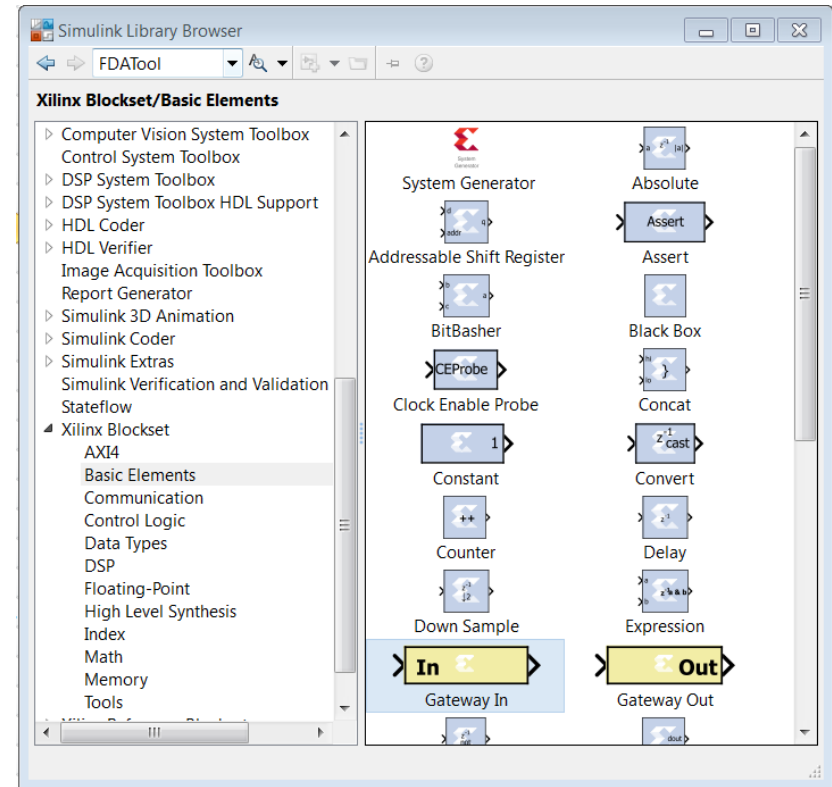
Step 1

select **Start > All Programs > Xilinx Design Tools > Vivado 2017.4 > System Generator > System Generator 2017.4.**

Step 2

Click the **Library Browser** button in the Simulink toolbar to open the Simulink Library Browser.

Expand the **Xilinx Blockset** menu, select the blocks needed for design



Example

Step 3

Configure the system generator blocks

Select Output type as **Fixed Point**

Configure the Fixed Point Precision
W, D
(Refer Slide 12)

Gateway In (Xilinx Gateway In)

Gateway in block. Converts inputs of type Simulink integer, single, double and fixed-point to Xilinx fixed-point or floating-point data type.

Hardware notes: In hardware these blocks become top level input ports.

Basic Implementation

Output Type

☐ Boolean ☒ Fixed-point ☐ Floating-point

Arithmetic type Signed (2's comp)

Fixed-point Precision

Number of bits 16 Binary point 14

Floating-point Precision

☐ Single ☐ Double ☐ Custom

Exponent width 8 Fraction width 24

Quantization:

☐ Truncate ☒ Round (unbiased: +/- Inf)

Overflow:

☐ Wrap ☒ Saturate ☐ Flag as error

Sample period 1/20e6

OK Cancel Help Apply

Example

Step 4

Configure Quantization

Configure Sampling rate:

Here the input port is sampled to a frequency (20 MHz) in order to adequately represent data.

The screenshot shows the 'Gateway In (Xilinx Gateway In)' configuration window. The 'Implementation' tab is selected. The 'Output Type' section has 'Fixed-point' selected. The 'Arithmetic type' is set to 'Signed (2's comp)'. Under 'Fixed-point Precision', 'Number of bits' is 16 and 'Binary point' is 14. Under 'Floating-point Precision', 'Single' is selected, 'Exponent width' is 8, and 'Fraction width' is 24. The 'Quantization' section has 'Round (unbiased: +/- Inf)' selected. The 'Overflow' section has 'Saturate' selected. The 'Sample period' is set to 1/20e6. The window includes 'OK', 'Cancel', 'Help', and 'Apply' buttons at the bottom.

Gateway In (Xilinx Gateway In)

Gateway in block. Converts inputs of type Simulink integer, single, double and fixed-point to Xilinx fixed-point or floating-point data type.

Hardware notes: In hardware these blocks become top level input ports.

Basic Implementation

Output Type

☐ Boolean ☒ Fixed-point ☐ Floating-point

Arithmetic type Signed (2's comp)

Fixed-point Precision

Number of bits 16 Binary point 14

Floating-point Precision

☒ Single ☐ Double ☐ Custom

Exponent width 8 Fraction width 24

Quantization:

☐ Truncate ☒ Round (unbiased: +/- Inf)

Overflow:

☐ Wrap ☒ Saturate ☐ Flag as error

Sample period 1/20e6

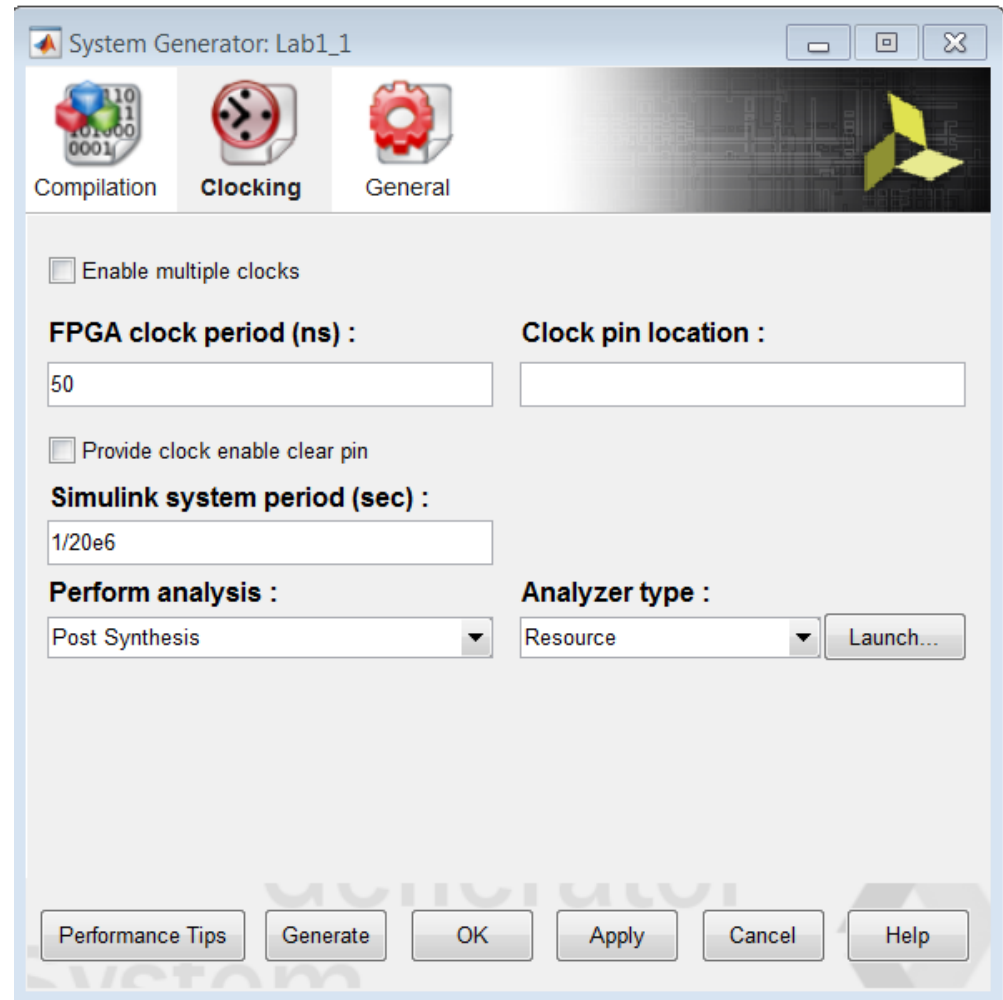
OK Cancel Help Apply

Example

Step 5

Select the **Clocking** tab.
(Refer slide 17)

- Specify an **FPGA clock Period**
- Specify a **Simulink system period**
- From the **Perform analysis** menu, select **Post Synthesis** and from **Analyzer type** menu select **Resource** as shown below. This option gives the resource utilization details after completion.



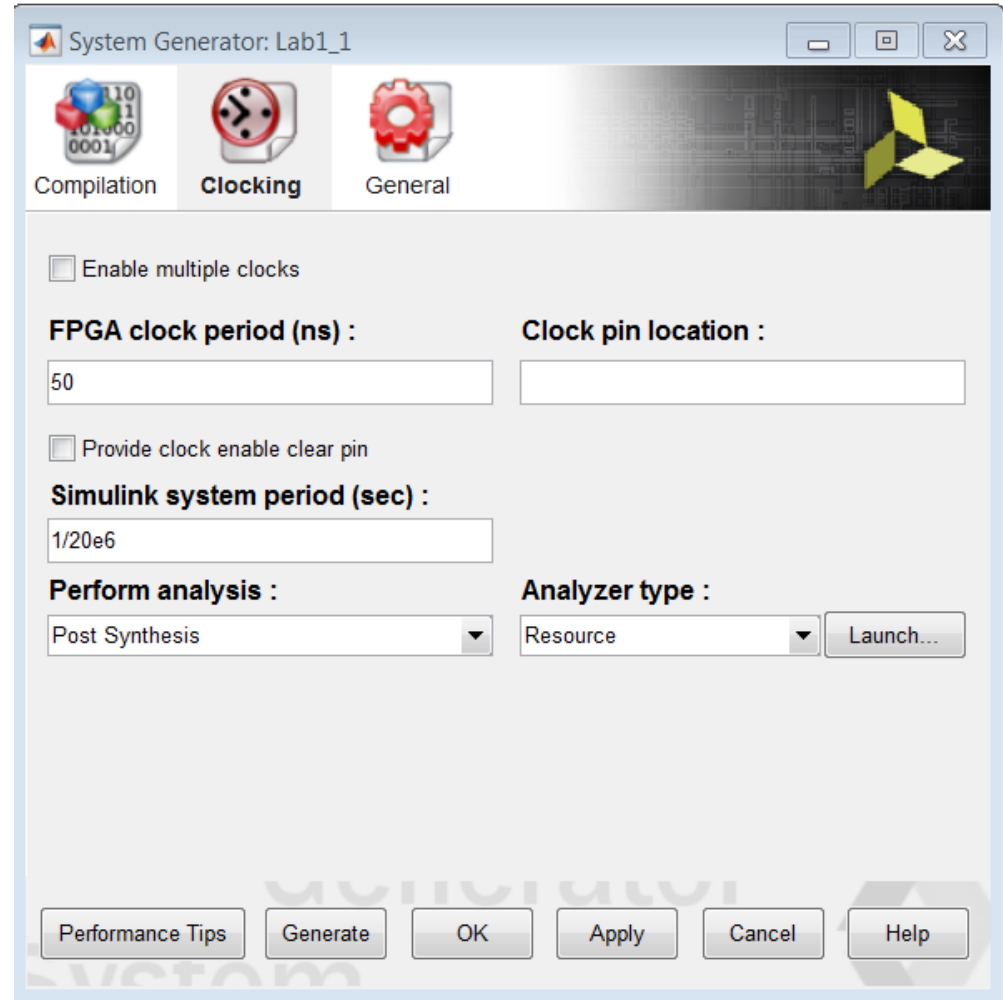
Example

Step 6

1. Click **OK** to exit the **System Generator** token.
2. Click the **Run** simulation button to simulate the design and view the results
3. Verify the simulation results

If the simulation results are fine,
Move to next step

Else debug the design and redo 2,3



Example

Step 7

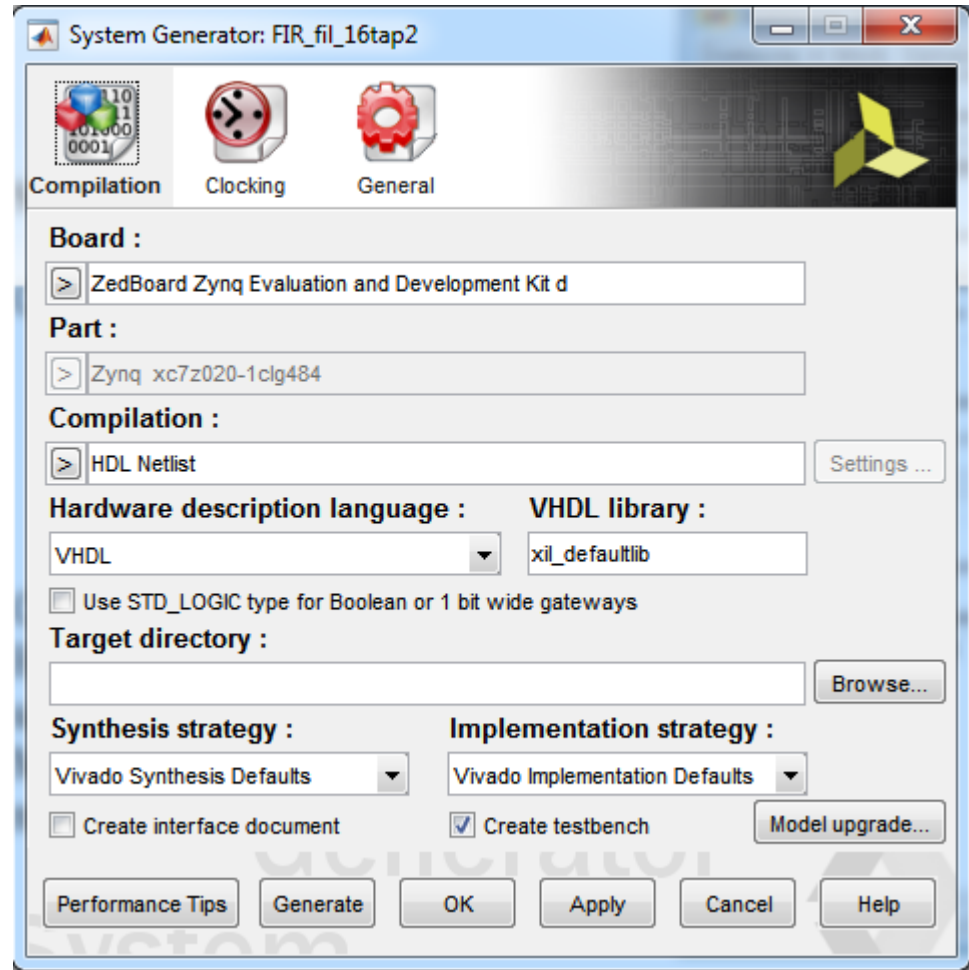
Double-click the **System Generator** token to open the Properties Editor.

Select the **Compilation** tab to specify details on the device and design flow.

From the **Compilation** menu, select the **H/W Co Simulation** or **HDL Netlist** as compilation target.

- **H/W Co Simulation:** To do hardware co simulation with ZedBoard.
- **HDL Netlist:** Preferred for timing analysis

The **Part** menu selects the FPGA device. For now, use the default device. Also, use the default **Hardware description language**, VHDL.

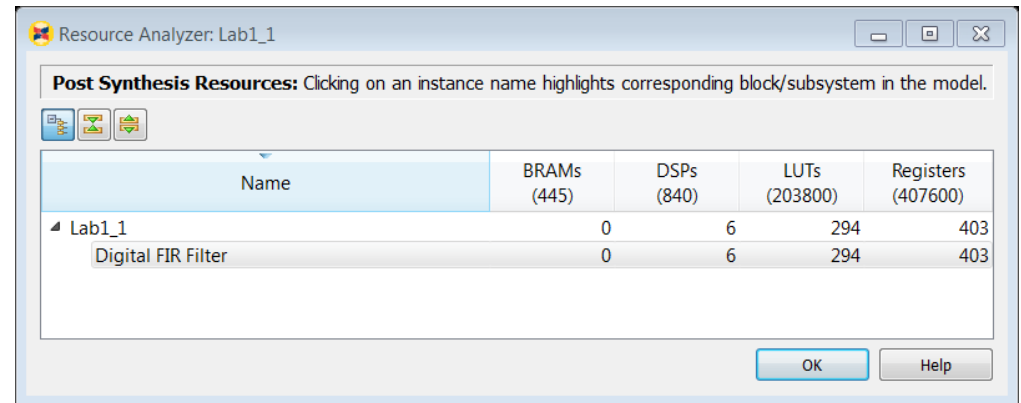


Example

Step 8

Click **Generate** to compile the design into hardware.

The compilation process transforms the design captured in Simulink blocks into an industry standard RTL (Register Transfer Level) design description. The RTL design can be synthesized into a hardware design. A Resource Analyzer window appears when the hardware design description has been generated.



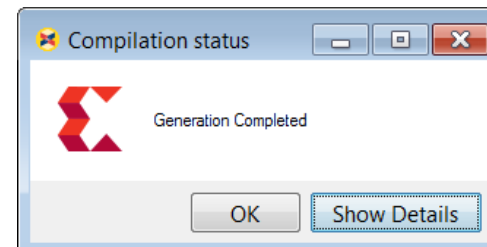
Resource Analyzer: Lab1_1

Post Synthesis Resources: Clicking on an instance name highlights corresponding block/subsystem in the model.

Name	BRAMs (445)	DSPs (840)	LUTs (203800)	Registers (407600)
Lab1_1	0	6	294	403
Digital FIR Filter	0	6	294	403

OK Help

The Compilation status dialog box also appears.

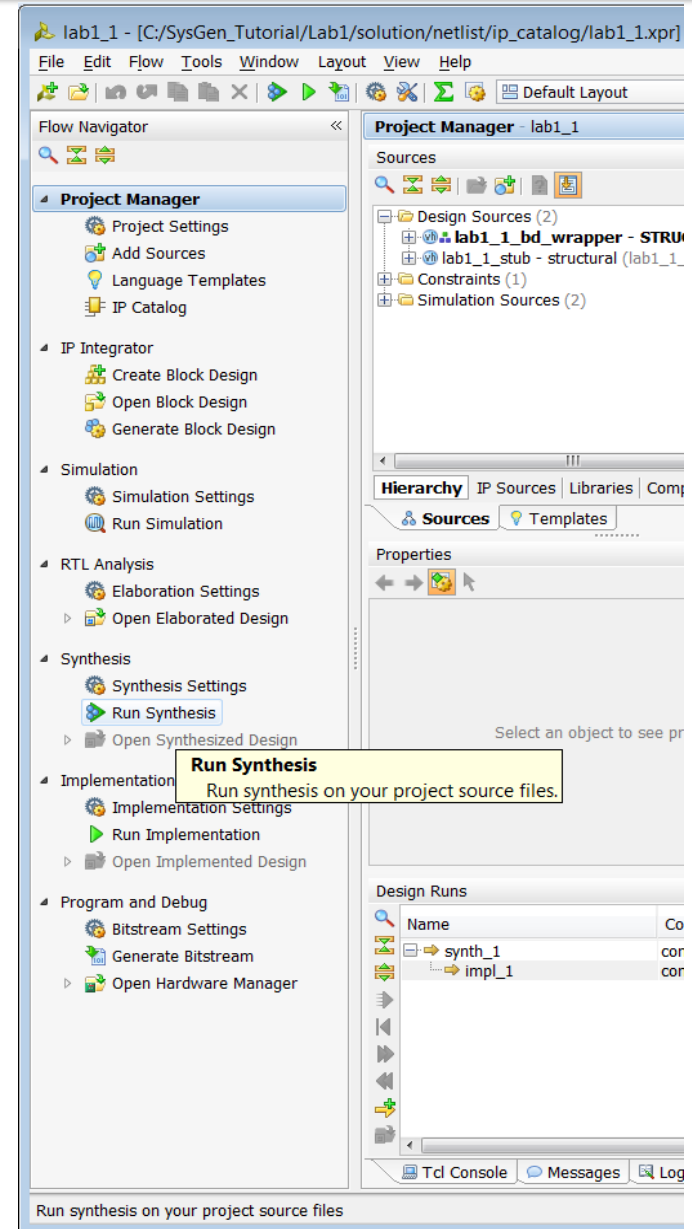


Example

Step 9

Invoke the Vivado Design Suite: **Start > All Programs > Xilinx Design Tools > Vivado 2017.4 > Vivado 2017.4**

Click **Open Project** and then navigate to the target directory to find the folder **hwcosim** or **hdl_netlist** and select the **Vivaldo Project File**. This invokes the generated project file in Vivado IDE.



Example

Step 10

In the Utilization section of the Project Summary, click the **Table** tab to view a summary of the resources used after the design is synthesized.

Utilization - Post-Synthesis

Resource	Estimation	Available	Utilization %
LUT	282	203800	0.14
LUTRAM	161	64000	0.25
FF	403	407600	0.10
DSP	6	840	0.71
IO	53	400	13.25
BUFG	1	32	3.13

Graph **Table**

Post-Synthesis Post-Implementation

Step 11

Now continue the steps given in Slide 26 for timing analysis.

Online Resources for Vivado and System Generator

- **Vivado Design Hub - System Generator for DSP:**
<https://www.xilinx.com/support/documentation-navigation/design-hubs/dh0014-vivado-system-generator-hub.html>
- **Vivado quick take videos:**
<https://www.xilinx.com/products/design-tools/vivado/vivado-video.html>