

Vivado Design Suite Tutorial

Power Analysis and Optimization

UG997 (v2018.3) December 5, 2018



Revision History

12/05/2018: Released with Vivado® Design Suite 2018.3 without changes from 2018.2.

Section	Revision Summary
06/06/2018 Version 2018.2	
General updates	Editorial updates only. No technical content updates.
04/04/2018 Version 2018.1	
General updates	Updated menu commands
Generating a SAIF File using Questa Advanced Simulator	Removed an important note regarding the support of Questa Advanced Simulator support.

Table of Contents

Revision History	2
Power Analysis and Optimization Tutorial	5
Software Requirements	5
Hardware Requirements.....	6
Locating Tutorial Design Files	7
Lab 1: Running Power Analysis in the Vivado Tools.....	9
Introduction	9
Step 1: Creating a New Project.....	9
Step 2: Synthesizing the Design	15
Step 3: Setting Up the Report Power	16
Step 4: Running Report Power.....	22
Step 5: Viewing the Power Properties.....	25
Step 6: Editing Power Properties and Refining the Power Analysis.....	26
Step 7: Running Functional Simulation with SAIF Output.....	29
Step 8: Incorporating SAIF Data into Power Analysis.....	32
Step 9: Implementing the Design	36
Conclusion	37
Lab 2: Running Timing Simulation and Estimating Power	38
Introduction	38
Step 1: Configuring and Running the Timing Simulation using Vivado Simulator.....	38
Step 2: Running Report Power in Vectorless Mode.....	41
Step 3: Running Report Power with Vivado Simulator SAIF Data	42
Generating a SAIF File using Questa Advanced Simulator.....	45
Step 1: Configuring and Running Timing Simulation in Questa Advanced Simulator	47
Step 2: Running Report Power in Vectorless Mode.....	49
Step 3: Running Report Power with Questa Advanced Simulator SAIF Data.....	51
Conclusion	52
Lab 3: Measuring Hardware Power Using the KC705 Evaluation Board.....	53
Introduction	53



Step 1: Generating a Bit File from the Implemented Design (Non-Power Optimization)	53
Step 2: Setting Up the KC705 Evaluation Board	54
Step 3: Setting Up the Fusion Digital Power Designer Software	55
Step 4: Programming the Bitstream.....	56
Step 5: Measuring the Hardware Power Rails	59
Step 6: Estimating Vectorless Power with Junction Temperature	61
Conclusion	63
Lab 4: Measuring Hardware Power Using the KCU105 Evaluation Board	64
Introduction	64
Step 1: Generating a Bit File from the Implemented Design.....	64
Step 2: Setting up the KCU105 Evaluation Board	65
Step 3: Configuring the Maxim Digital Power Tool Software	66
Step 4: Programming the Bitstream.....	66
Step 5: Measuring the Hardware Power Rails	71
Step 6: Estimating the Vectorless Power with Junction Temperature.....	73
Conclusion	75
Lab 5: Performing Power Optimization	76
Introduction	76
Step 1: Setting Up Options to Run Power Optimization.....	76
Step 2: Running <code>report_power_opt</code> to Examine User/Design Specific Power Optimizations.....	79
Step 3: Running <code>report_power</code> to Examine Power Savings	81
Step 4: Turning Off Optimizations on Specific Signals and Rerunning the Implementation.....	82
Step 5: Running <code>report_power_opt</code> to Examine Tool Optimizations Again	84
Step 6: Saving Power using UltraScale BRAM in Cascaded Mode	85
Conclusion	87
Legal Notices.....	88
Please Read: Important Legal Notices	88

This tutorial introduces the power analysis and optimization use model recommended for use with the Xilinx® Vivado® Integrated Design Environment (IDE). The tutorial describes the basic steps involved in taking a small example design from RTL to implementation, estimating power through the different stages, and using simulation data to enhance the accuracy of the power analysis. It also describes the steps involved in using the power optimization tools in the design.



VIDEO: The [Vivado Design Suite QuickTake Video: Power Estimation and Analysis Using Vivado](#) shows how the Vivado Design Suite can help you to estimate power consumption in your design and reviews best practices for getting the most accurate estimation.



VIDEO: The [Vivado Design Suite QuickTake Video: Power Optimization Using Vivado](#) describes the factors that affect power consumption in an FPGA, shows how the Vivado Design Suite helps to minimize power consumption in your design, and looks at some advanced control and best practices for getting the most out of Vivado power optimization.

Software Requirements

This tutorial requires that the Vivado Design Suite software 2017.1 release or later is installed. For installation instructions and information, see the *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* (UG973).

For hardware power measurement of 7 series devices, the tutorial requires Texas Instruments Fusion Design Power Designer software, which can be downloaded from the following location:
http://www.ti.com/tool/fusion_digital_power_designer

For hardware power measurement of UltraScale™ devices, the tutorial requires Maxim Digital Power Tool software, which can be downloaded from the following location:
<https://www.maximintegrated.com/en/design/tools/applications/evkit-software/index.mvp?id=1183>

Hardware Requirements

Supported Operating Systems to run the Vivado Design Suite, and memory recommendations when using the Vivado tools, are described in the *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* (UG973).

Hardware Requirements for 7 Series Devices

- The hardware power measurements for 7 series devices
(needed in [Lab 3: Measuring Hardware Power Using the KC705 Evaluation Board](#)), require a Xilinx Kintex®-7 FPGA KC705 Evaluation Kit. You can find information on the Evaluation Kit at this location:
[Xilinx Kintex®-7 FPGA KC705 Evaluation Kit](#)
- For power measurements through TI Power Regulators
(needed in [Lab 3: Measuring Hardware Power Using the KC705 Evaluation Board](#)), use the Texas Instruments USB Interface Adapter. You can find information on the USB Interface Adaptor at this location:
www.ti.com/lit/ml/sllu093/sllu093.pdf

Hardware Requirements for UltraScale Devices

- The hardware power measurements in UltraScale devices
(needed in [Lab 4: Measuring Hardware Power Using the KCU105 Evaluation Board](#)), require a Xilinx Kintex® UltraScale FPGA KCU105 Evaluation Kit. You can find information on the Evaluation Kit at the following location:
[Xilinx Kintex® UltraScale FPGA KCU105 Evaluation Kit](#)
- For power measurements through Maxim Digital Power Tool
(needed in [Lab 4: Measuring Hardware Power Using the KCU105 Evaluation Board](#)), use the Maxim Power interface adapter.
You can find information on the interface adapter at the following location:
www.maximintegrated.com/en/products/power/power-switching/MAXPOWERTOOL002.html

Locating Tutorial Design Files

1. Download the reference design files from the Xilinx website:

[ug997-vivado-power-analysis-optimization-tutorial.zip](#)

2. Extract the zip file contents into any write-accessible location.

This tutorial refers to the location of the extracted `ug997-vivado-power-analysis-optimization-tutorial.zip` file contents as `<Extract_Dir>`.



IMPORTANT: You will modify the tutorial design data while working through this tutorial. Use a new copy of the original data each time you start this tutorial.

The `ug997-vivado-power-analysis-optimization-tutorial.zip` file includes a readme file which contains the details and version history of the design files along with the folders of 7 series and UltraScale design files.

7 Series Tutorial Design Files

You can find a separate 7 series folder containing the 7 series tutorial design files in the contents of the zip file.

The following table describes the contents of the 7 series tutorial design files:

Directories/Files	Description
/src	Contains the design HDL and testbench for the functional simulation.
/src/dut_fpga.v	Top module for the design.
/src/bram_tdp.v /src/bram_top.v /src/dut.v	Other design blocks - synthesized module.
dut_fpga_kc705.xdc	Contains clocking and timing constraints for the design.
/src/testbench.v	Testbench for simulating the design.

UltraScale Device Tutorial Design Files

You can find a separate UltraScale folder containing the UltraScale device tutorial design files in the contents of the zip file.

The following table describes the contents of the UltraScale device tutorial design files:

Directories/Files	Description
/src	Contains the design HDL and testbench for the simulation.
/src/dut_fpga.v	Top module for the design.

Directories/Files	Description
/src/dut.v /src/Cascade_bram.v /src/Noncascade_bram.v /src/bram_top_cascade.v /src/bram_top_noncascade.v /src/bram_tdp_cas.v /src/bram_tdp_noncas.v	Other design blocks.
dut_fpga_kcu105.xdc	Contains clocking and timing constraints for the design.
/src/testbench.v	Testbench for simulating the design.

Introduction

In this lab, you will learn about the Power Analysis and Optimization features in the Vivado® IDE. The lab will take you through the steps of project creation and power analysis at the synthesis stage, using the Vivado Report Power feature in vectorless mode. It will also demonstrate using the SAIF file generated from behavioral simulation for Vivado Report Power Analysis.

You will analyze power in the Vivado IDE. Then you will examine some of the major features in the Power Report window and closely examine some power specific Tcl commands. You will also learn to create a SAIF file by simulating the design in the timing simulation stage using both the Vivado simulator and Questa Advanced Simulator.

You will also learn how to achieve Power Optimization after `opt_design` in the Vivado IDE. You will examine the power optimization report and selectively turn power optimizations ON or OFF on specific signals, nets, modules, or hierarchy.

Step 1: Creating a New Project

To create a project, use the New Project wizard to name the project, to add RTL source files and constraints, and to specify the target device.

Note: Throughout this tutorial, Xilinx® 7 series example design is used to explain the process of configuring, implementing, estimating the power through different stages, and using simulation data to enhance the accuracy of the power analysis. For UltraScale™ device design, most of the steps are similar to 7 series. Additional information, wherever necessary, is provided for UltraScale devices.

- On Linux,
 1. Go to the directory where the lab materials are stored:

```
cd <Extract_Dir>/7_series (for 7 series devices)
or
cd <Extract_Dir>/UltraScale (for UltraScale devices)
```
 2. Launch the Vivado IDE: `vivado`

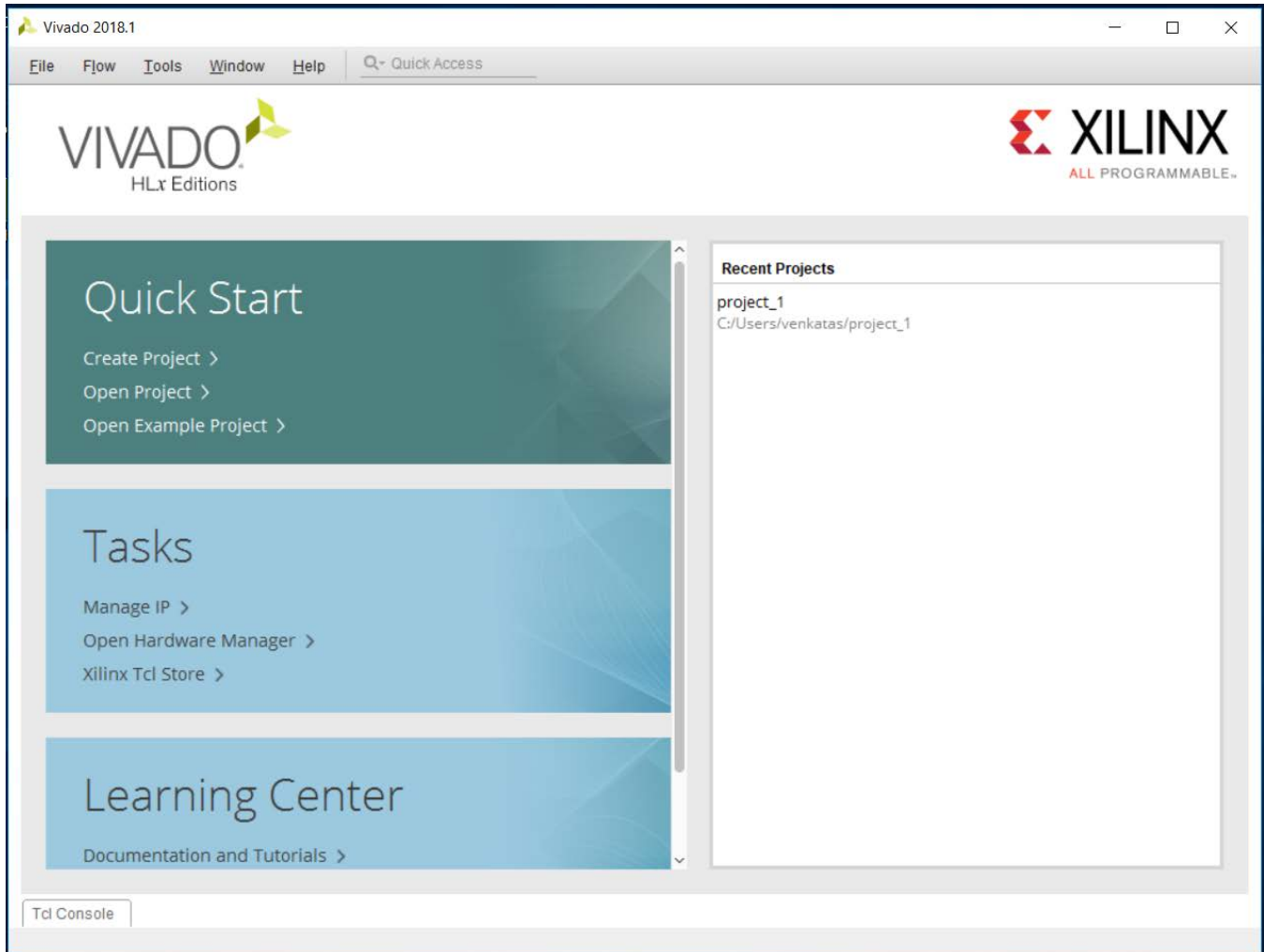


Figure 1: Vivado IDE – Getting Started Page

- On Windows,
 3. Launch the Vivado IDE:
Start > All Programs > Xilinx Design Tools > Vivado 2018.x > Vivado 2018.x
(x denotes the latest version of Vivado 2018 IDE)
 4. As an alternative, click the **Vivado 2018.x** Desktop icon to start the Vivado IDE.
 The Vivado IDE Getting Started page contains links to open or create projects and to view documentation.
 5. In the Getting Started page, click **Create New Project** to start the New Project wizard.
 6. Click **Next** to continue to the next screen.

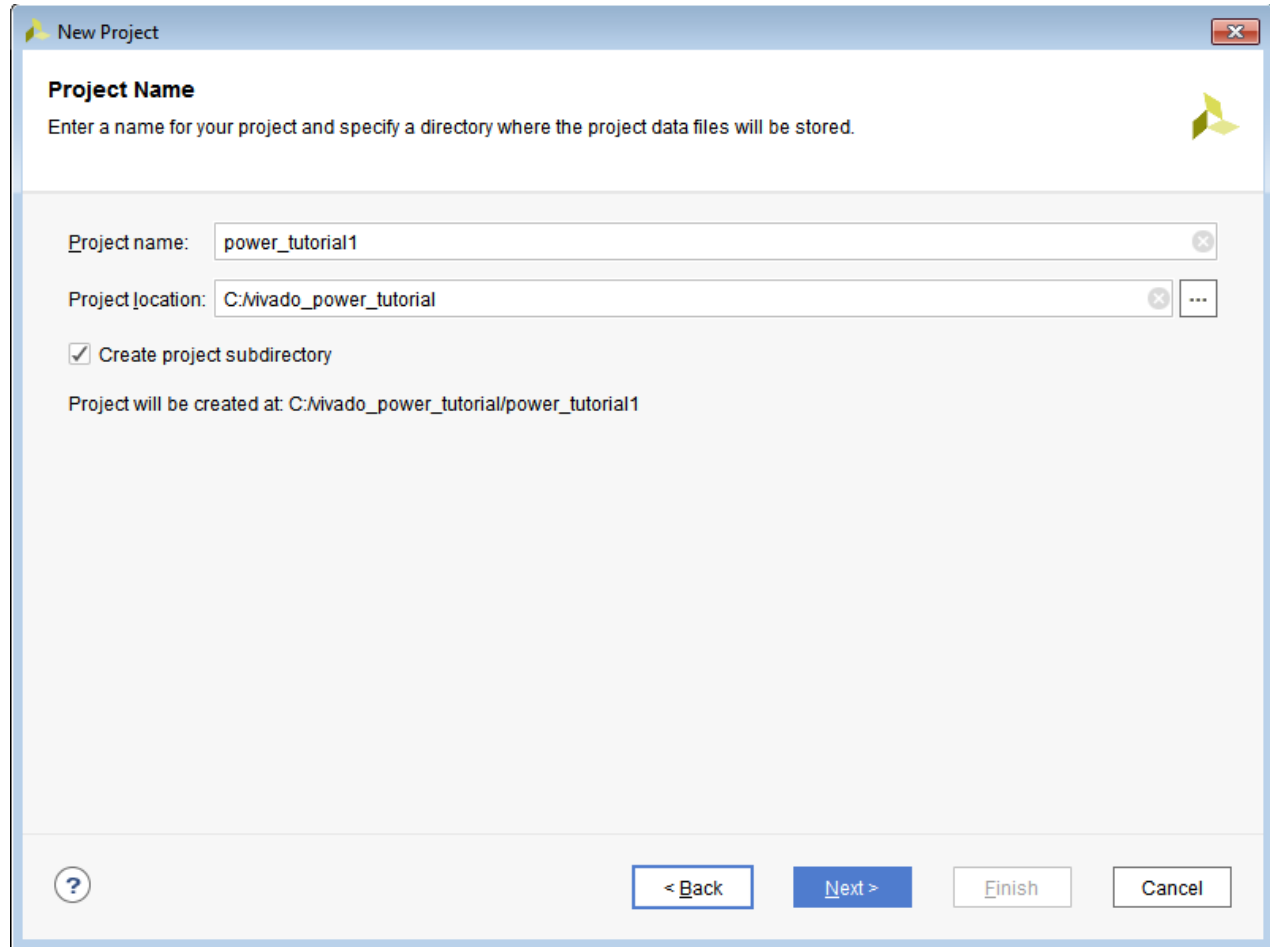


Figure 2: Creating a New Project

7. In the Project Name page, name the new project `power_tutorial1` and enter the project location (`C:\Vivado_Power_Tutorial`). Make sure to check the **Create project subdirectory** option and click **Next**.
8. In the Project Type page, specify the **type of project** to create as **RTL Project**, make sure to uncheck the **Do not specify sources at this time** option, and click **Next**.
9. In the Add Sources page:
 - a. Set **Target Language** to **Verilog** and **Simulator language** to **Mixed**.
 - b. Click the **Add Files** button.
 - c. In the Add Source Files dialog box, navigate to the `<Extract_Dir>/7_series/src` directory for 7 Series devices or `<Extract_Dir>/UltraScale/src` for UltraScale devices.
 - d. Select all of the Verilog (`.v`) source files, and click **OK**.
 - e. In the Add Sources page, change the **HDL Source For** the `testbench.v` file to **Simulation Only**.

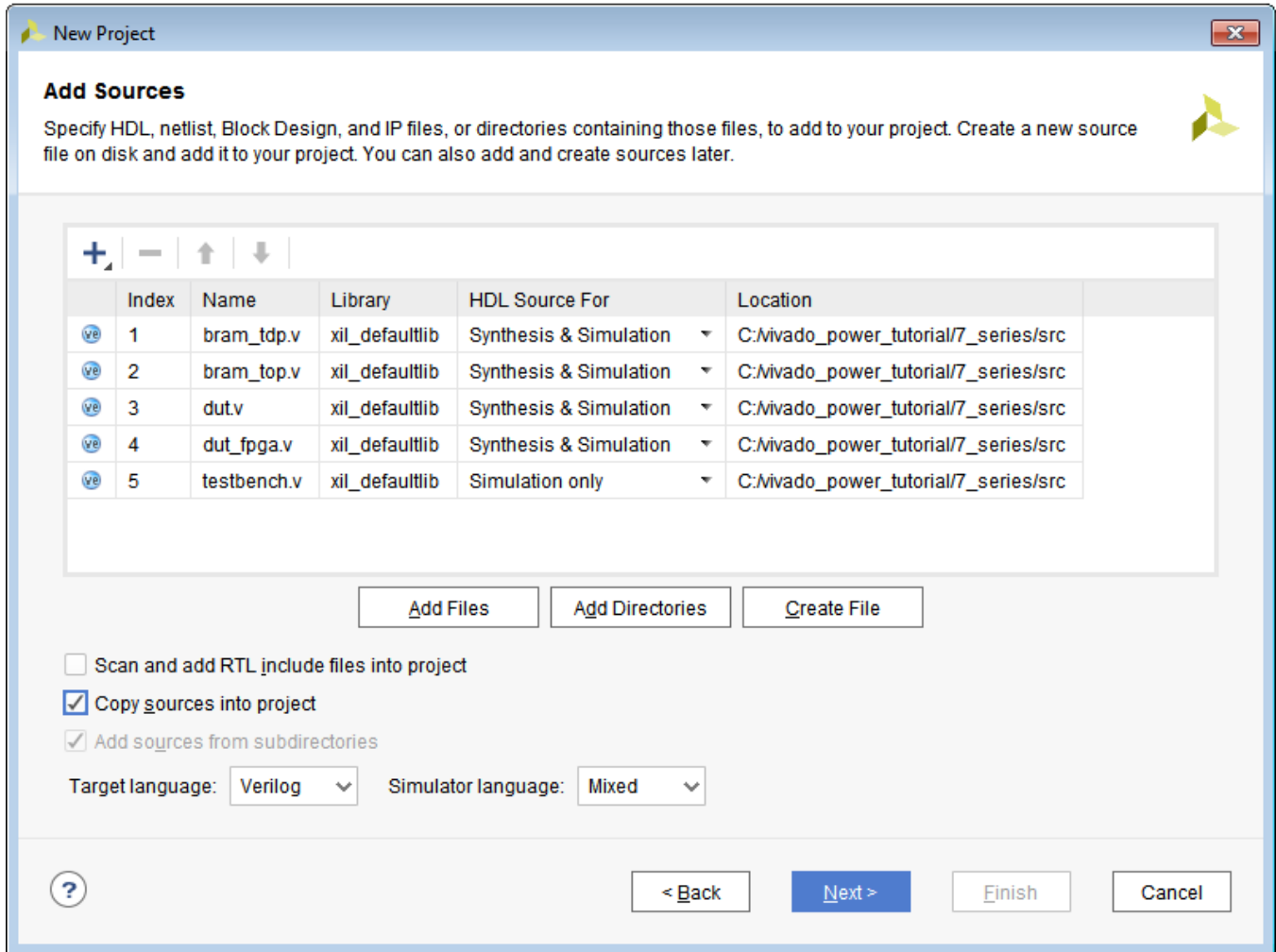


Figure 3: Setting HDL Source Type for testbench.v (7 Series)

- f. Verify that the files are added and **Copy sources into project** is checked. Click **Next**.
10. In the Add Existing IP (optional) dialog box, click **Next**.
11. In the Add Constraints (optional) dialog box, click **Add Files** and select `dut_fpga_kc705.xdc` in the file browser. In the directory structure, you will find the `dut_fpga_kc705.xdc` file below the `/src` folder.

For UltraScale devices, select `dut_fpga_kcu105.xdc` in the file browser. In the directory structure, you will find the `dut_fpga_kcu105.xdc` file below the `/src` folder.
12. Click **Next** to continue.
13. In the Default Part dialog box, click **Boards** to specify the board for the target device and select **Kintex-7 KC705 Evaluation Platform** for 7 Series or **Kintex-UltraScale KCU105 Evaluation Platform** for UltraScale devices. Then click **Next**.



TIP: When you specify a **Board**, you are also specifying the part you are targeting for your design, in this case an xc7k325tffg900-2 FPGA for 7 Series or xcku040-ffva156-2-e FPGA for UltraScale devices.

14. Review the New Project Summary page. Verify that the data appears as expected, per the steps above, and click **Finish**.

Note: It might take a moment for the project to initialize in the Vivado IDE.

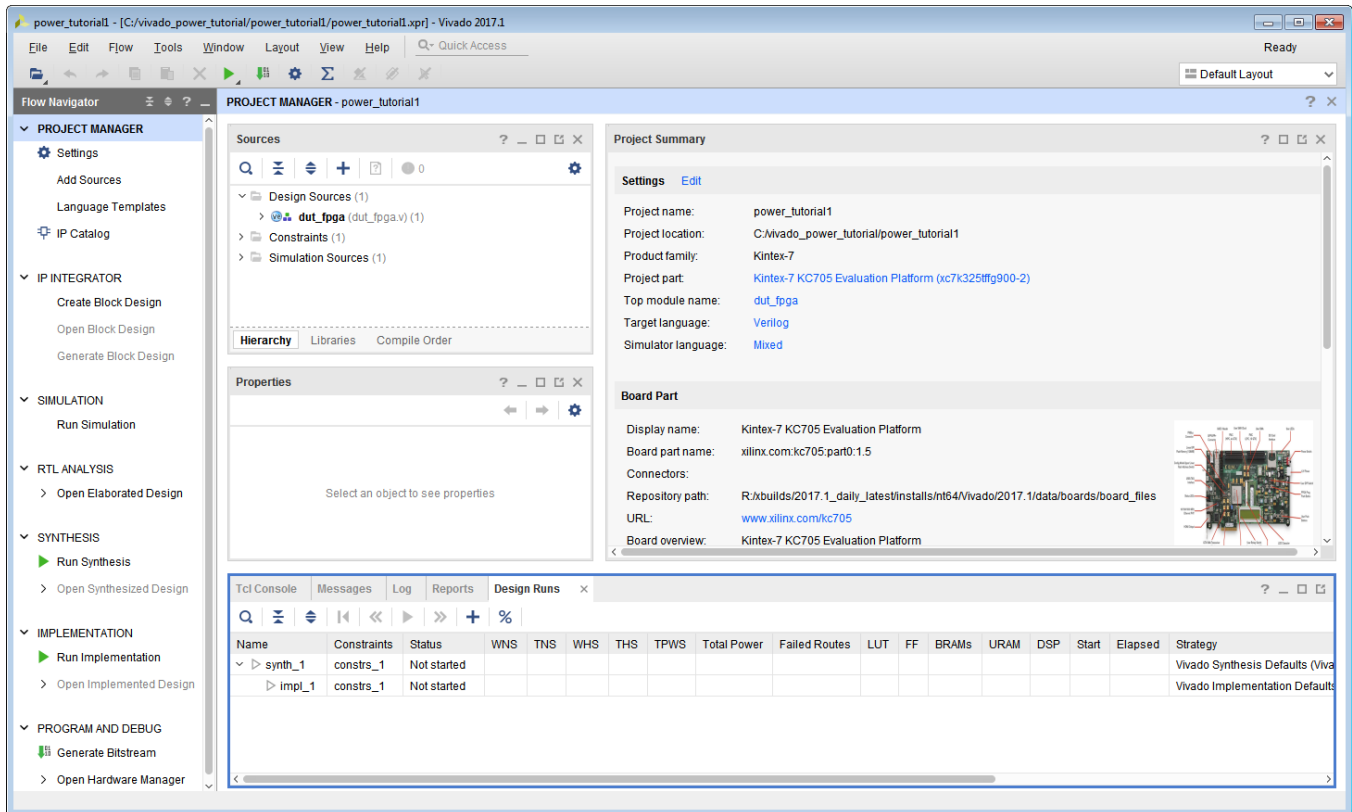


Figure 4: Project in Vivado IDE (7 Series)

15. In the Vivado Options dialog box (**Tools > Settings > Tool Settings > Project**), enter the tutorial project directory in the **Specify project directory** box, so that all reports are saved in the tutorial project directory. Then click **OK**.

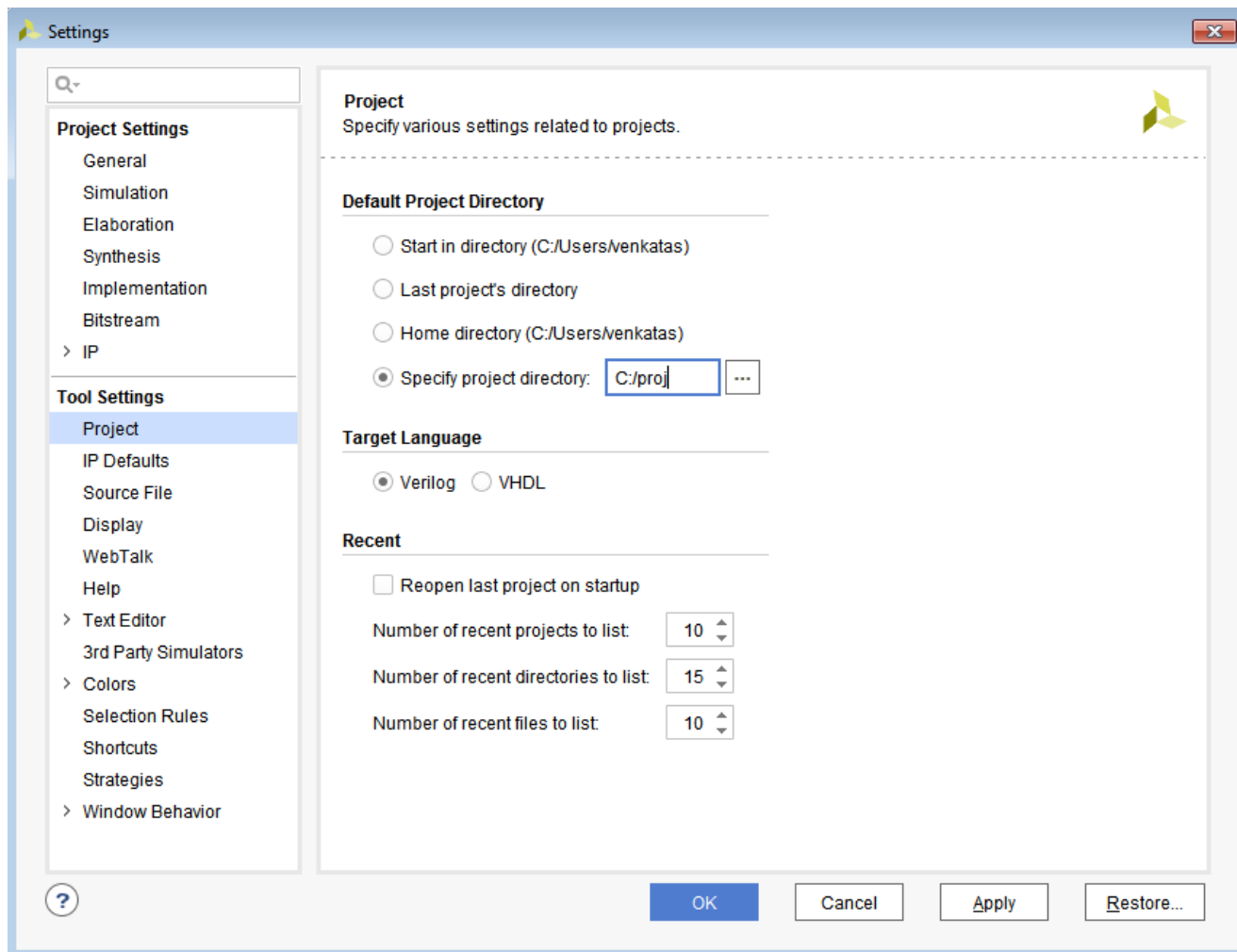


Figure 5: Setting the Project Directory

Now, the design is ready for Synthesis.

Step 2: Synthesizing the Design

1. Click **Run Synthesis** in the Flow Navigator. In the Launch Runs Dialog box that appears click **OK**.

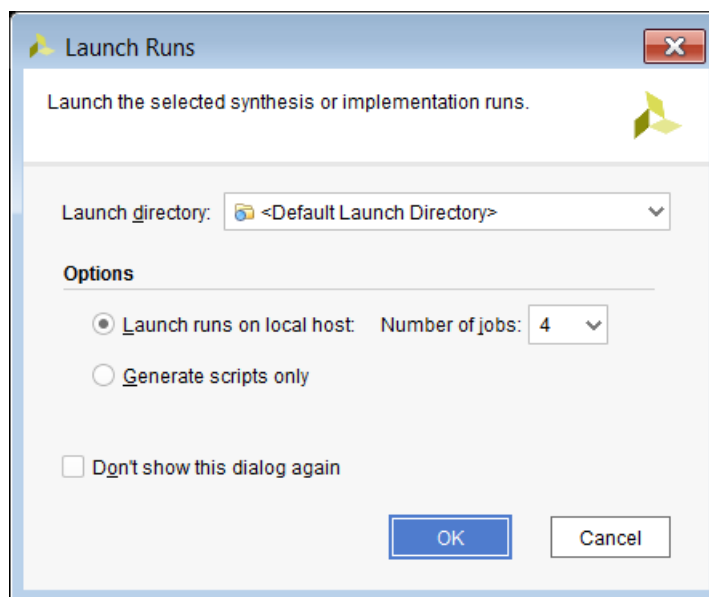


Figure 6: Launch Runs Dialog Box

2. The Synthesis Completed dialog box appears after synthesis has completed on the design.

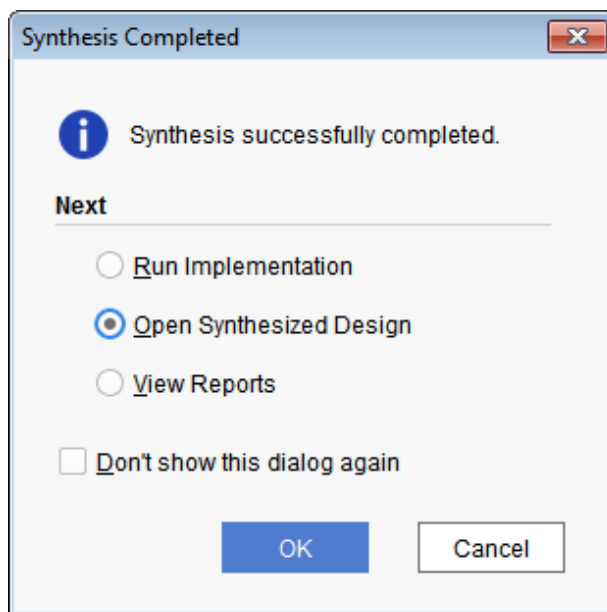


Figure 7: Synthesis Completed Dialog Box

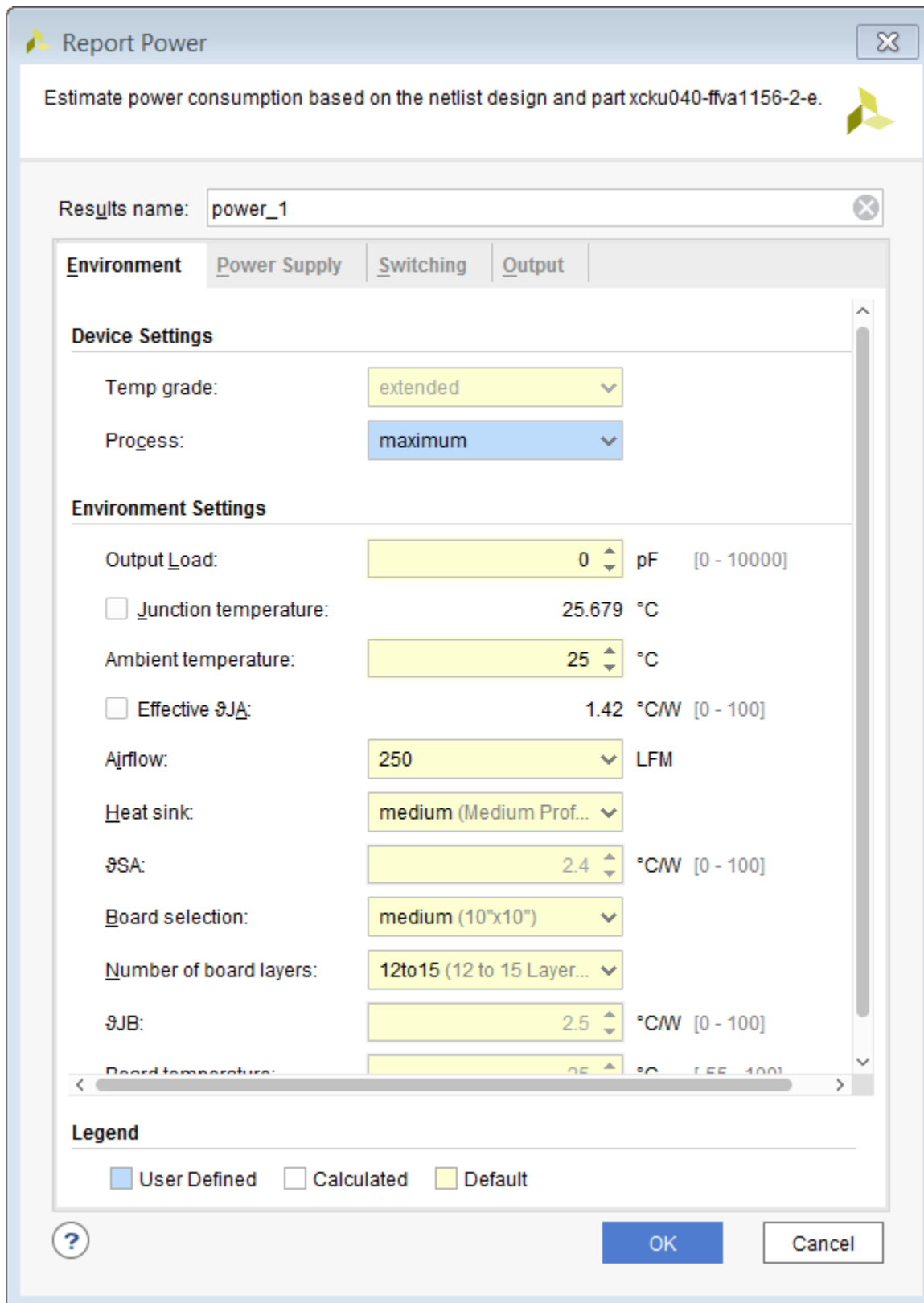
3. Open the synthesized design by selecting **Open Synthesized Design** in the Synthesis Completed dialog box and clicking **OK**.

Step 3: Setting Up the Report Power

The Vivado IDE allows you to specify input data to the Report Power tool to enhance the accuracy of the power analysis.

In the Vivado IDE, you can configure thermal, environmental, and power supply options to mimic the board level settings as closely as possible. For information on setting these options, see the *Vivado Design Suite User Guide: Power Analysis and Optimization* ([UG907](#)).

1. In the main menu bar, select **Reports > Report Power**.
2. Examine the **Environment** tab in the Report Power dialog box ([Figure 8](#)).



Report Power

Estimate power consumption based on the netlist design and part xcku040-fva1156-2-e.

Results name: power_1

Environment | Power Supply | Switching | Output

Device Settings

Temp grade: extended

Process: maximum

Environment Settings

Output Load: 0 pF [0 - 10000]

☐ Junction temperature: 25.679 °C

Ambient temperature: 25 °C

☐ Effective θ_{JA} : 1.42 °C/W [0 - 100]

Airflow: 250 LFM

Heat sink: medium (Medium Prof...)

θ_{SA} : 2.4 °C/W [0 - 100]

Board selection: medium (10"x10")

Number of board layers: 12to15 (12 to 15 Layer...)

θ_{JB} : 2.5 °C/W [0 - 100]

Board temperature: 25 °C [55 - 100]

Legend

☒ User Defined ☐ Calculated ☐ Default

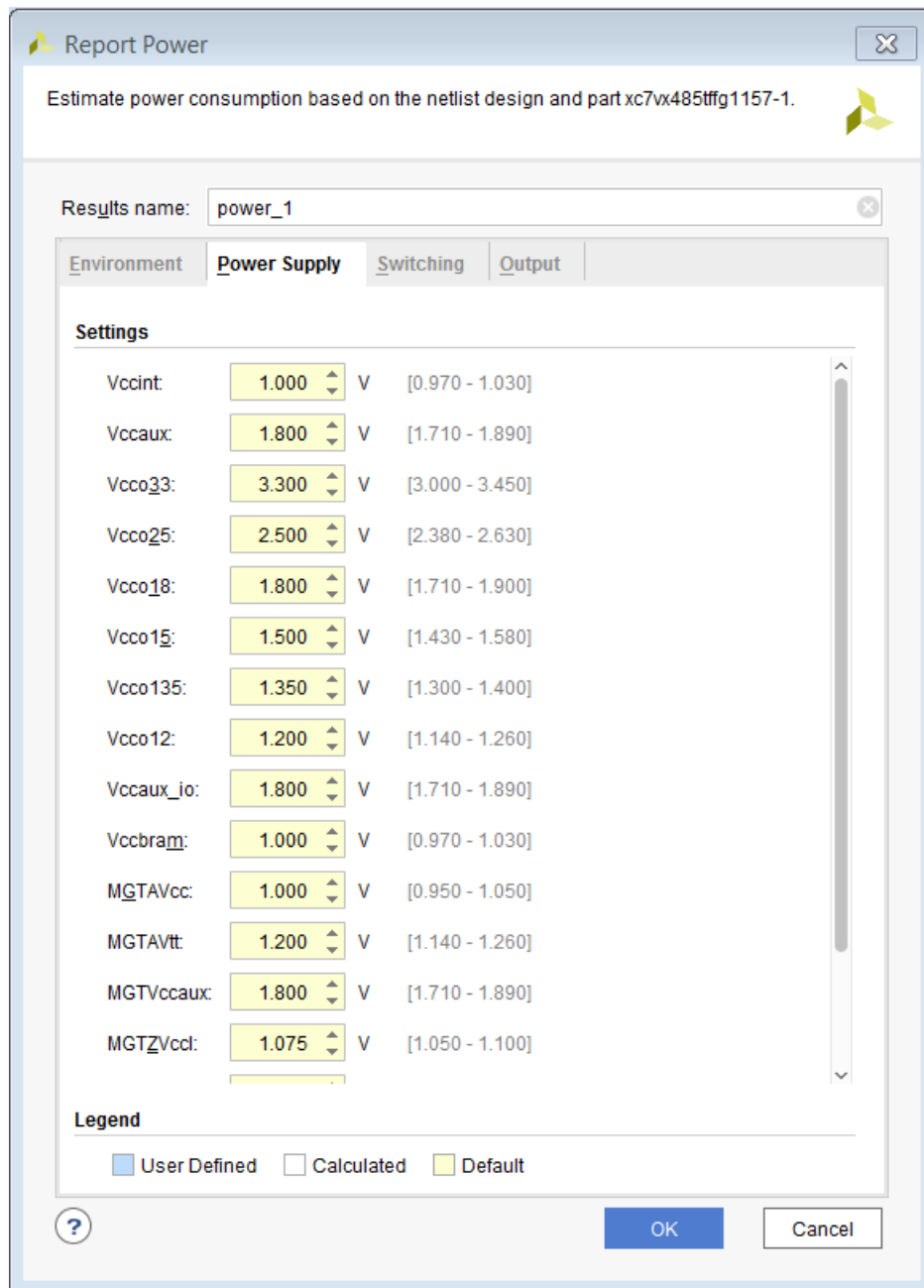
OK Cancel

Figure 8: Report Power Dialog Box (Environment Tab)

- In the **Environment** tab, set **Process** to **maximum** for a worst case power analysis. Examine the **Power Supply** tab (Figure 9).



IMPORTANT: By default, Vivado Report Power uses nominal values for voltage supply sources. Voltage is a large factor contributing to both static and dynamic power. For the most accurate analysis, ensure that actual voltage values are entered for each supply. Similarly, ensure temperature and other environmental factors match actual operating conditions.



Report Power

Estimate power consumption based on the netlist design and part xc7vx485tffg1157-1.

Results name: power_1

Environment Power Supply Switching Output

Settings

Vccint:	1.000	V	[0.970 - 1.030]
Vccaux:	1.800	V	[1.710 - 1.890]
Vcco33:	3.300	V	[3.000 - 3.450]
Vcco25:	2.500	V	[2.380 - 2.630]
Vcco18:	1.800	V	[1.710 - 1.900]
Vcco15:	1.500	V	[1.430 - 1.580]
Vcco135:	1.350	V	[1.300 - 1.400]
Vcco12:	1.200	V	[1.140 - 1.260]
Vccaux_io:	1.800	V	[1.710 - 1.890]
Vccbram:	1.000	V	[0.970 - 1.030]
MGTAVcc:	1.000	V	[0.950 - 1.050]
MGTAVtt:	1.200	V	[1.140 - 1.260]
MGTVccaux:	1.800	V	[1.710 - 1.890]
MGTZVcd:	1.075	V	[1.050 - 1.100]

Legend

☒ User Defined ☐ Calculated ☐ Default

OK Cancel

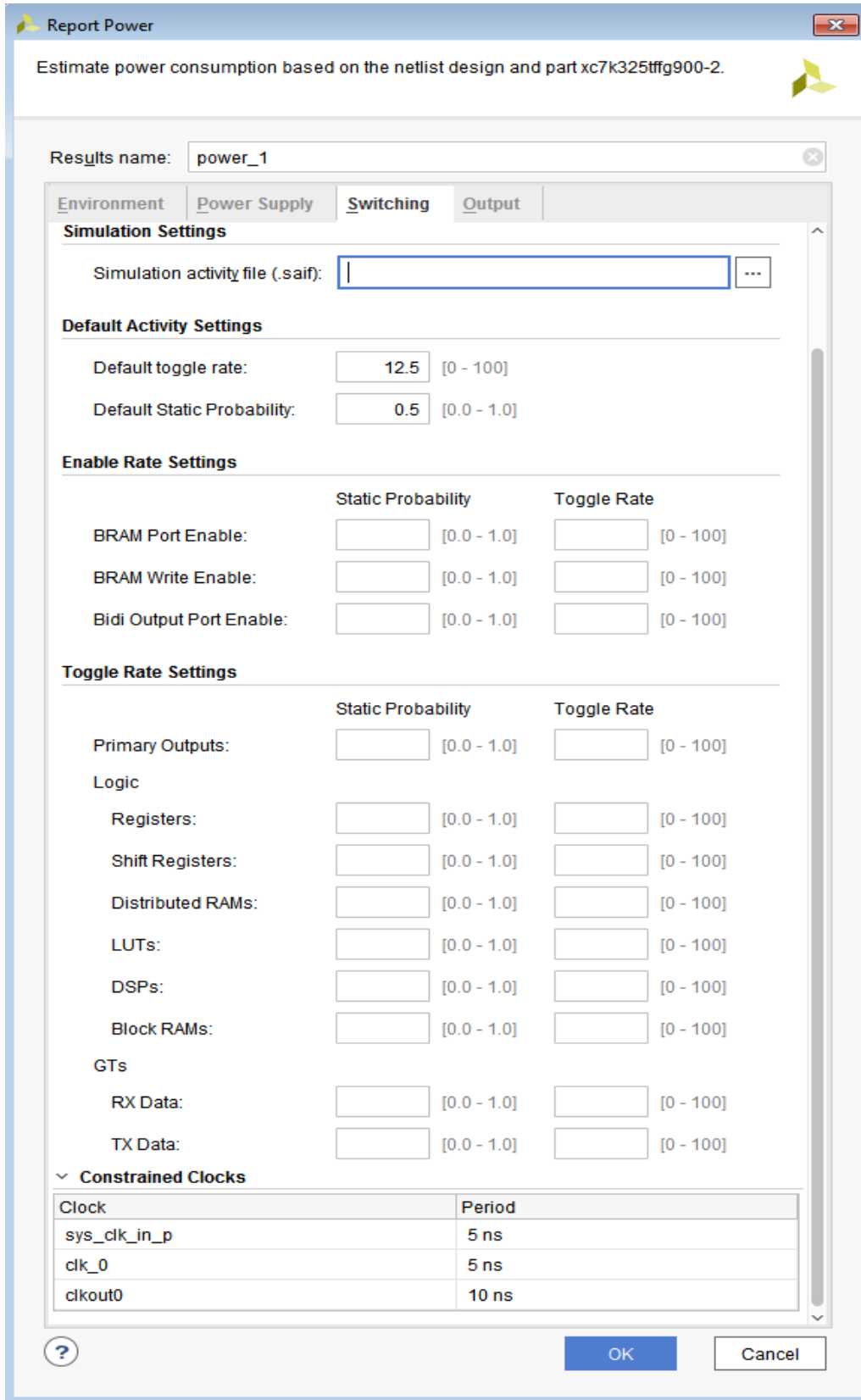
Figure 9: Report Power Dialog Box – Power Supply Tab

4. In the **Switching** tab, expand Constrained **Clocks** and examine the constrained clocks in the design (Figure 10).

IMPORTANT: Make sure all the relevant clocks in the design are constrained. All the design clocks must be defined using `create_clock` or `create_generated_clock` XDC constraints, so that Report Power recognizes the clocks.



Default toggle rate is set to 12.5% and **Default Static Probability** is set to 0.5. This will be applied to primary input ports (non-clock) and block box outputs.



Report Power

Estimate power consumption based on the netlist design and part xc7k325tffg900-2.

Results name: power_1

Environment Power Supply Switching Output

Simulation Settings

Simulation activity file (.saif):

Default Activity Settings

Default toggle rate: 12.5 [0 - 100]

Default Static Probability: 0.5 [0.0 - 1.0]

Enable Rate Settings

	Static Probability	Toggle Rate
BRAM Port Enable:	[0.0 - 1.0]	[0 - 100]
BRAM Write Enable:	[0.0 - 1.0]	[0 - 100]
Bidi Output Port Enable:	[0.0 - 1.0]	[0 - 100]

Toggle Rate Settings

	Static Probability	Toggle Rate
Primary Outputs:	[0.0 - 1.0]	[0 - 100]
Logic		
Registers:	[0.0 - 1.0]	[0 - 100]
Shift Registers:	[0.0 - 1.0]	[0 - 100]
Distributed RAMs:	[0.0 - 1.0]	[0 - 100]
LUTs:	[0.0 - 1.0]	[0 - 100]
DSPs:	[0.0 - 1.0]	[0 - 100]
Block RAMs:	[0.0 - 1.0]	[0 - 100]
GTs		
RX Data:	[0.0 - 1.0]	[0 - 100]
TX Data:	[0.0 - 1.0]	[0 - 100]

Constrained Clocks

Clock	Period
sys_clk_in_p	5 ns
clk_0	5 ns
clkout0	10 ns

OK Cancel

Figure 10: Report Power Dialog Box – Switching tab expanded to display Constrained Clocks

5. In the **Output** tab of Report Power dialog box, specify the **Output text file** as `power_1.pwr`.
6. Specify the **Output XPE file** as `power_1.xpe`. After creating this file when Report Power runs, you can import the file and results into the Xilinx Power Estimator (XPE). For information on importing the file in to the Xilinx Power Estimator (XPE), see the *Xilinx Power Estimator User Guide* (UG440).
7. Specify the RPX file to write the results of the Report command in to the specified report file (RPX). The saved RPX file can be reloaded using the **Reports > Open Interactive Report** command to provide interaction/cross-probing with the open design.

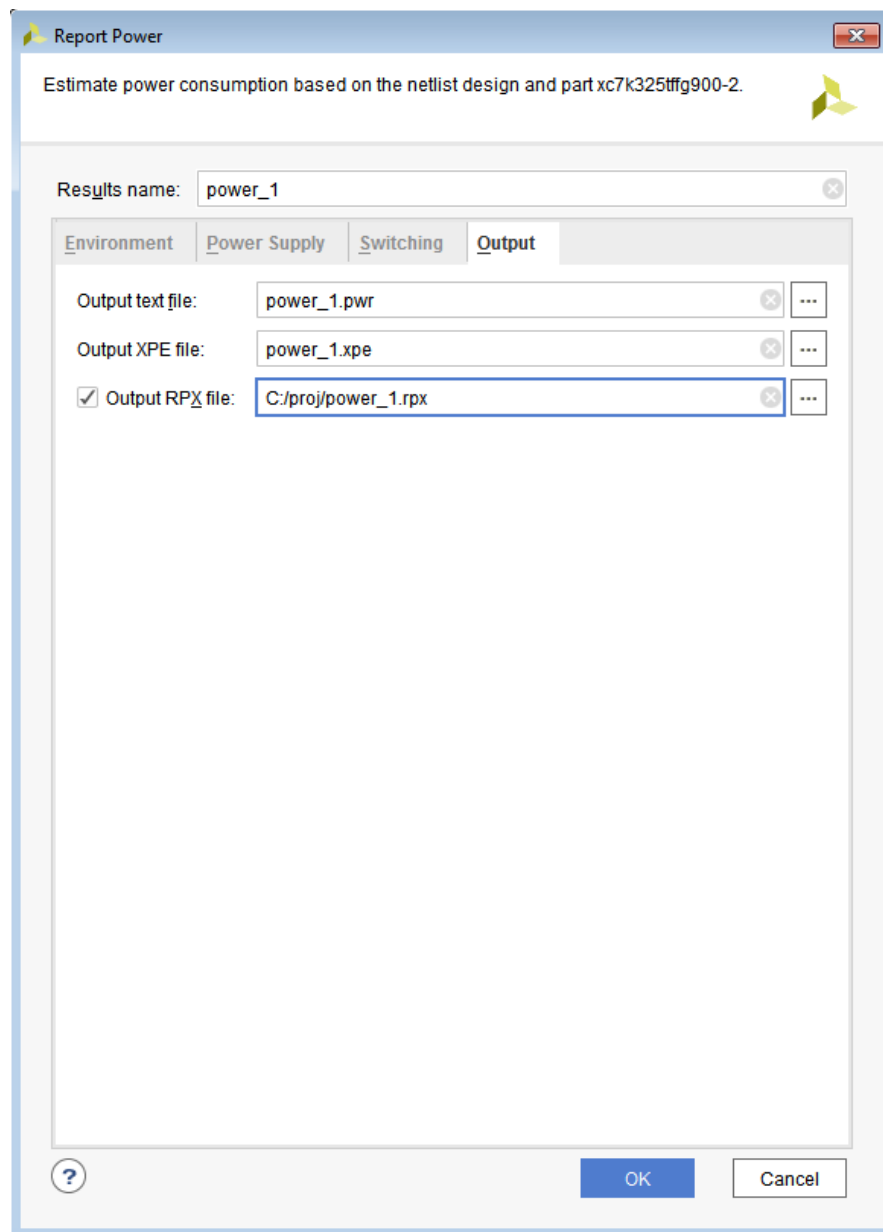


Figure 11: Report Power Dialog Box (Output Tab)

Legends in Report Power Tool

The following legends appear consistently in the Report Power tool:

Constraint	Displays when the Nets are defined as clock with timer constraints. The defined frequency of a clock determines the switching activity.
Simulation	Displays when the Nets with switching activities are derived from simulation's .saif file
User Defined	Displays when the Nets with user set switching activities are derived from set_switching_activity power tcl command.
Estimated	Displays when the Nets with switching activities are generated by report_power vectorless propagation engine.
Default	Displays when the Nets include default switching activities. If you use set_switching_activity on input port nets or on internal nets before running report_power (vectorless propagation), the report tool displays the default.

Step 4: Running Report Power

- Click **OK** on the Report Power dialog box.
This runs the `report_power` command.
- Examine the Power Report **power_1** generated in the Power tab of the results windows area in the Vivado IDE.



TIP: Due to continuous accuracy improvements in the Vivado tools, the actual power numbers you see might be slightly different than the ones that appear in the following figures.

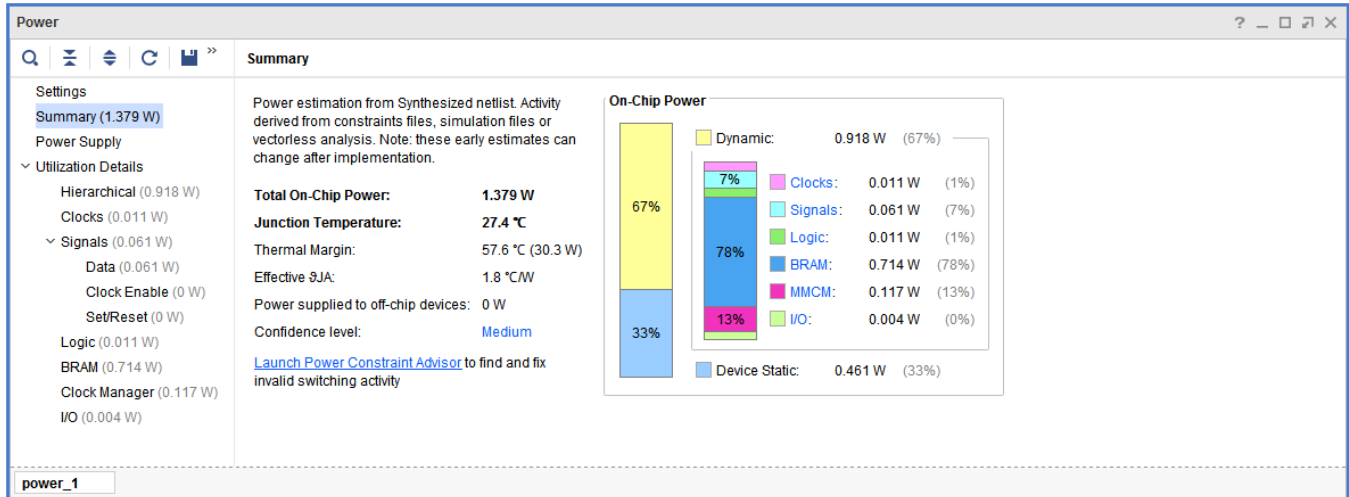


Figure 12: Power Report

- Examine the power breakdown in the Power Report by block type (**Logic**, **BRAM**, **I/O**, etc.).
- Examine the power supply breakdown in the **Power Supply** view.

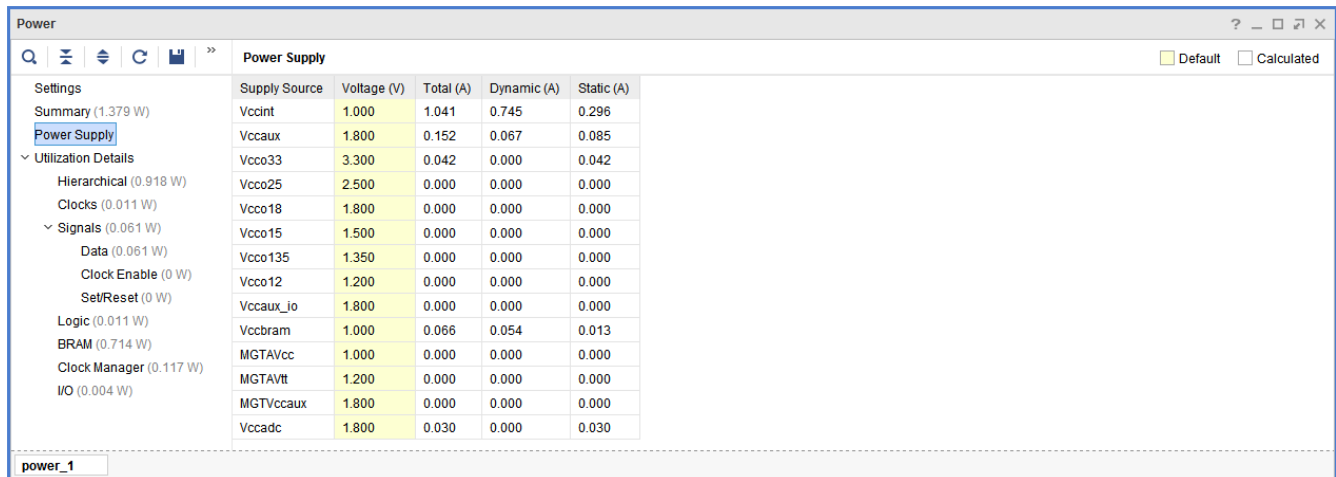


Figure 13: Power Report – Power Supply Display

- Examine the Hierarchical breakdown of the power in the **Utilization Details > Hierarchical** view.

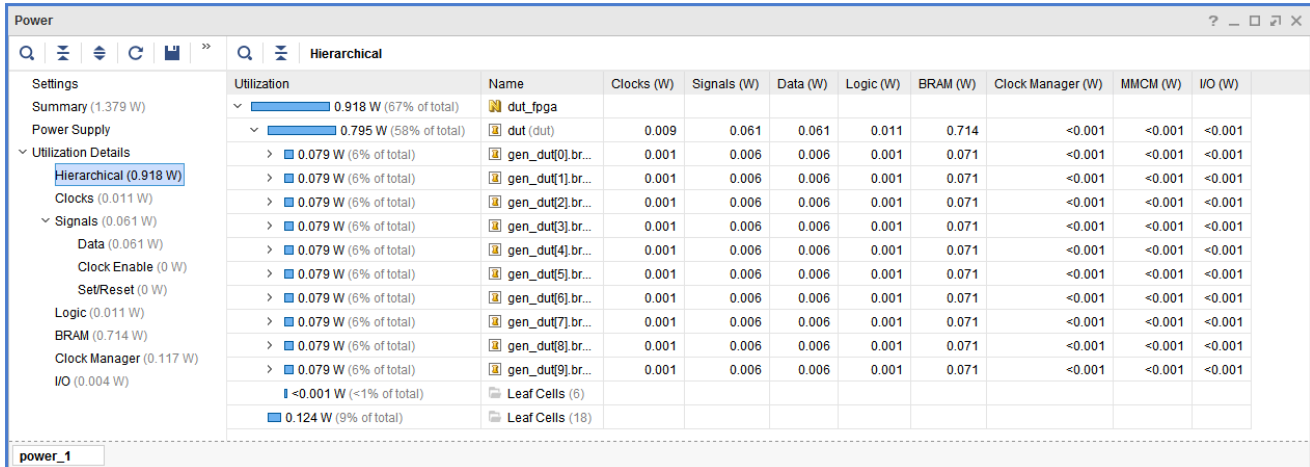


Figure 14: Power Report – Hierarchical Display

6. Examine the **Clocks** view and the various **Signals** views (**Data**, **Clock Enable**, and **Set/Reset**).

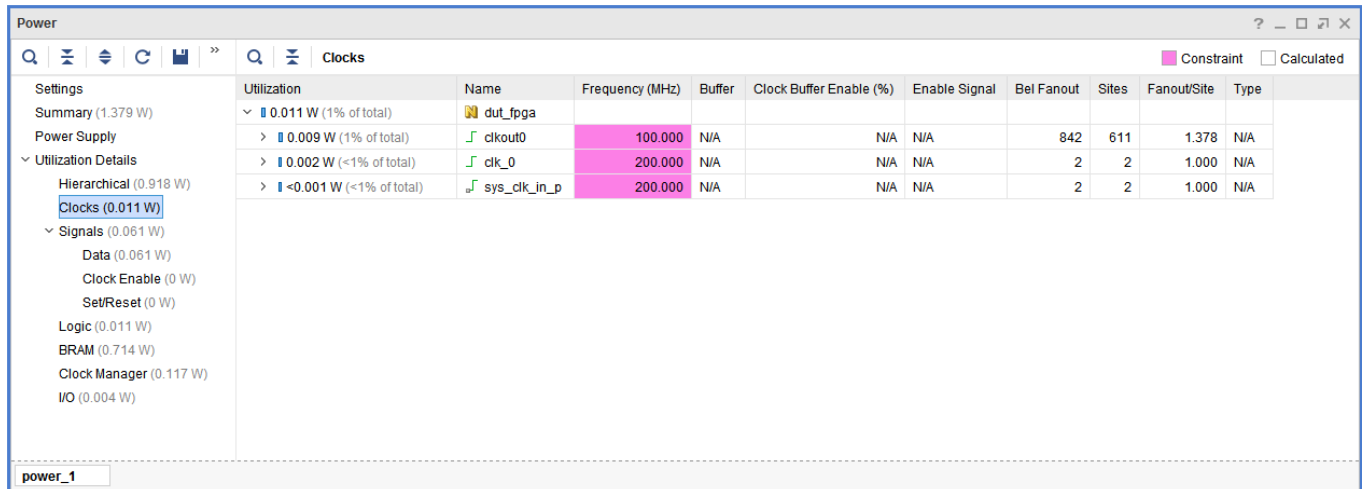


Figure 15: Power Report – Clocks Display

Step 5: Viewing the Power Properties

This step will show you how you can get the display of static probability and toggle rate for a signal in property window.

1. Note the total power (**Total On-Chip Power**) in the Power Report **Summary** view.
2. Click the **Set/Reset** item in the Power Report.
3. Click on the **dut/dut_reset** signal.

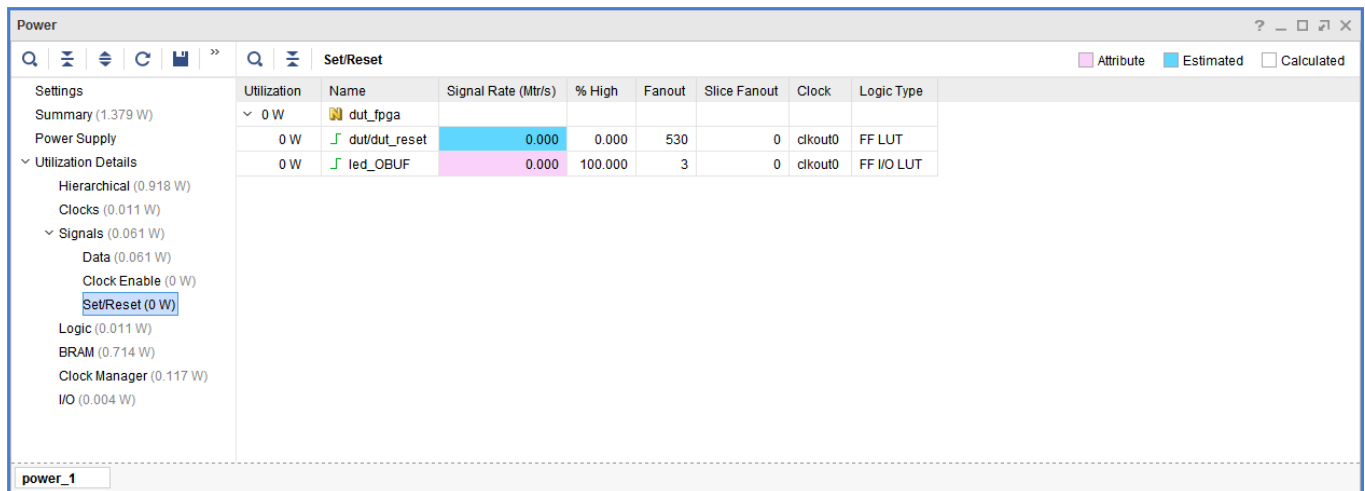


Figure 16: Power Report – Set/Reset Display

4. Note that there is a **Power** tab in the Net Properties window, displaying Net Properties for the **dut/dut_reset** signal. Click on **Load Power Properties** to get the power information the first time.

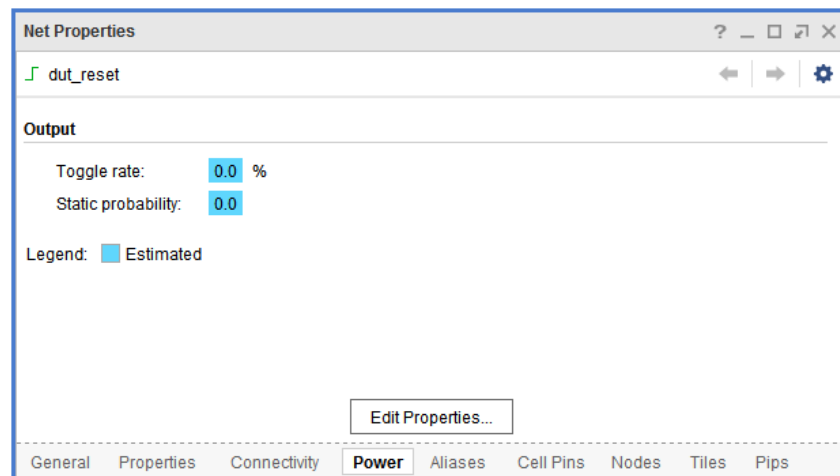


Figure 17: Properties Window – Set/Reset Net Properties

5. Note the **Toggle rate** is 0% and the **Static probability** is 0 for the **dut/dut_reset** signal, which indicates that Reset is always de-asserted in the design.

Step 6: Editing Power Properties and Refining the Power Analysis

Assume the Reset is asserted for 10% of the cycles in this design. Switching activity can be set accordingly to re-estimate the power.

1. In the Net Properties window, click the **Edit Properties** button.
2. In the Edit Power Properties dialog box, change the **Toggle rate** to 4% and the **Static probability** to 0.1.

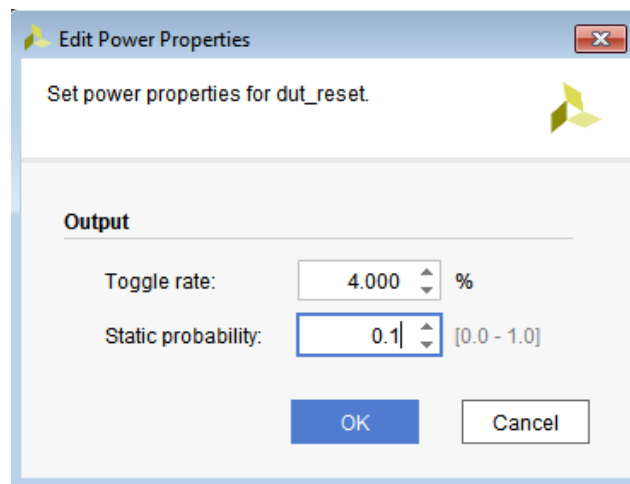


Figure 18: Edit Power Properties Dialog Box

3. Click **OK**.
4. In the Net Properties window, observe that the **Toggle Rate** and **Static Probability** values turn a different color to indicate that they are user defined.

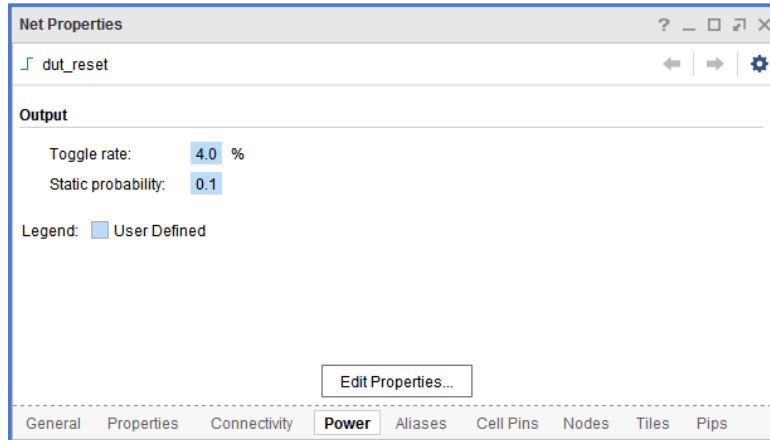


Figure 19: User Defined Color in Net Properties Window

You can also observe the equivalent Tcl command executed in the Tcl Console.

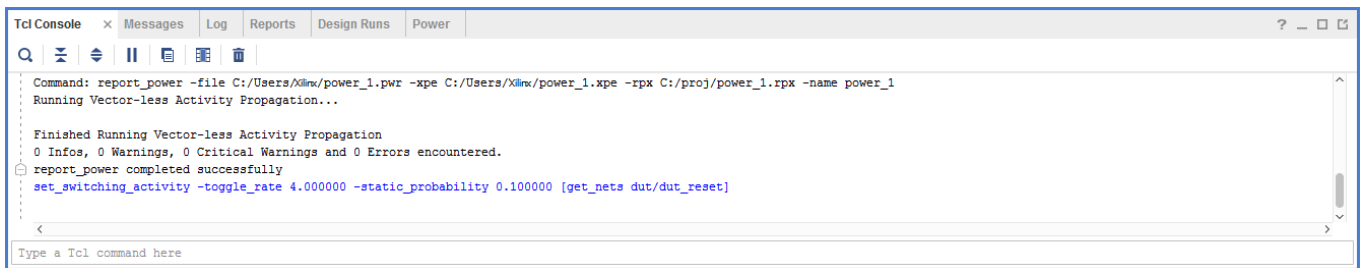


Figure 20: Equivalent Tcl Command for Net Property Update

5. Rerun Report Power (**Reports> Report Power**).
6. Change the **Output text File** and **Output XPE File** in the **Output** tab to `power_2.pwr` and `power_2.xpe` respectively.
7. In the **Switching** tab, set **Switching Activity for Resets:** to **None**. Then click **OK**.
8. In the Power window, note the change in total power reported in the **power_2** report compared to the **power_1** report. The total power has *decreased* due to the change in the **Signal Rate** for the **dut/dut_reset** signal. Since the signal is a **reset** signal, an increase in its activity will significantly reduce the activity of other signals in the design. Note that the **Signal Rate** of the **dut/dut_reset** signal is now color coded as being **User Defined** in both the properties window and the Set/Reset view of the Power Report.

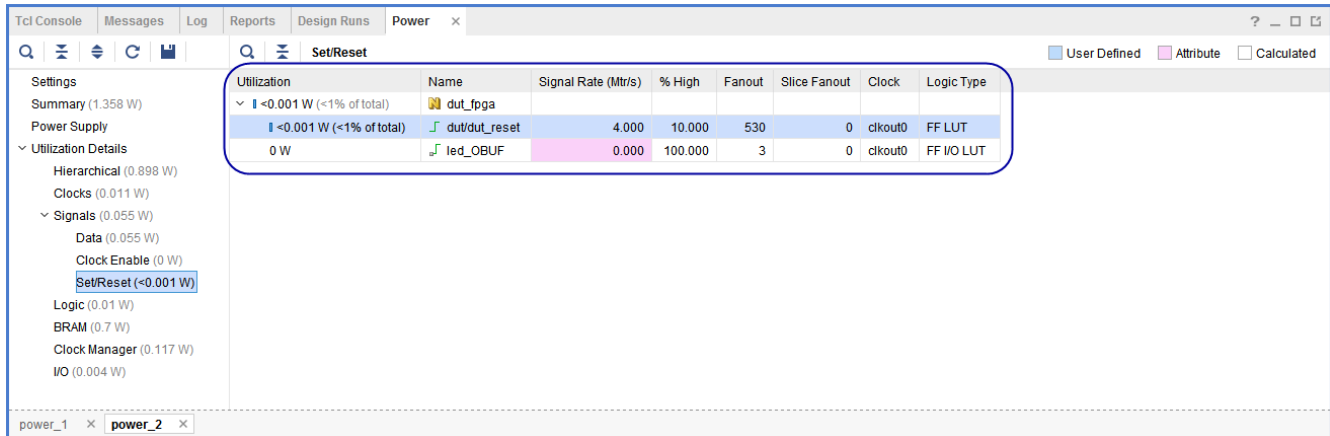


Figure 21: Power Report – User Defined Signal Rate

Xilinx recommends you to double-check the signal rates and percentage high (%high) values of high impact I/O ports, control signals (such as resets and clock enables) and high fanout nets. This is an opportunity to guide the Report Power tool to the right estimation scenario.

See the *Vivado Design Suite User Guide: Power Analysis and Optimization* ([UG907](#)) for more information on switching activity.

TIP: In Tcl, use the `set_switching_activity` command to change the signal rate and static probability of signals and use `report_switching_activity` to query the values that were set on the signals.

```
set_switching_activity -signal_rate 4 -static_probability 0.1 [get_nets dut/dut_reset]
report_switching_activity [get_nets dut/dut_reset]
```

IMPORTANT: Switching activity can also be specified in terms of toggle rate. Toggle rate is always associated with a clock. The primary ports can be associated with a specific clock using the `set_input_delay` and `set_output_delay` commands. If no clock association is found, Report Power will associate the ports with respect to the capturing clock. For a clock of 100 MHz and a toggle rate of 4, the equivalent signal rate will be 4 MTr/s ($\text{signal_rate} = \text{toggle_rate} * \text{Freq} = 4 * 100\text{MHz}$).

Step 7: Running Functional Simulation with SAIF Output

Now that you have created a Vivado Design Suite project for the tutorial design, you can set up and launch the Vivado simulator to run post-synthesis functional simulation. Simulation will generate a switching activity values file (SAIF) that will enable you to do more accurate power estimation on your design.

1. In the Flow Navigator, click **Settings** to open the **Project Settings** dialog box and set the simulation properties in **Simulation** section.
2. In the Simulation section of Project Settings dialog box, note that the following Simulation defaults are automatically set for you based on the design files:

Simulator language: `Mixed`

Simulation set: `sim_1`

Simulation top-module name: `testbench`

3. In the Elaboration tab of Simulation, make sure the **xsim.elaborate.debug_level** is set to **typical**, which is the default value.

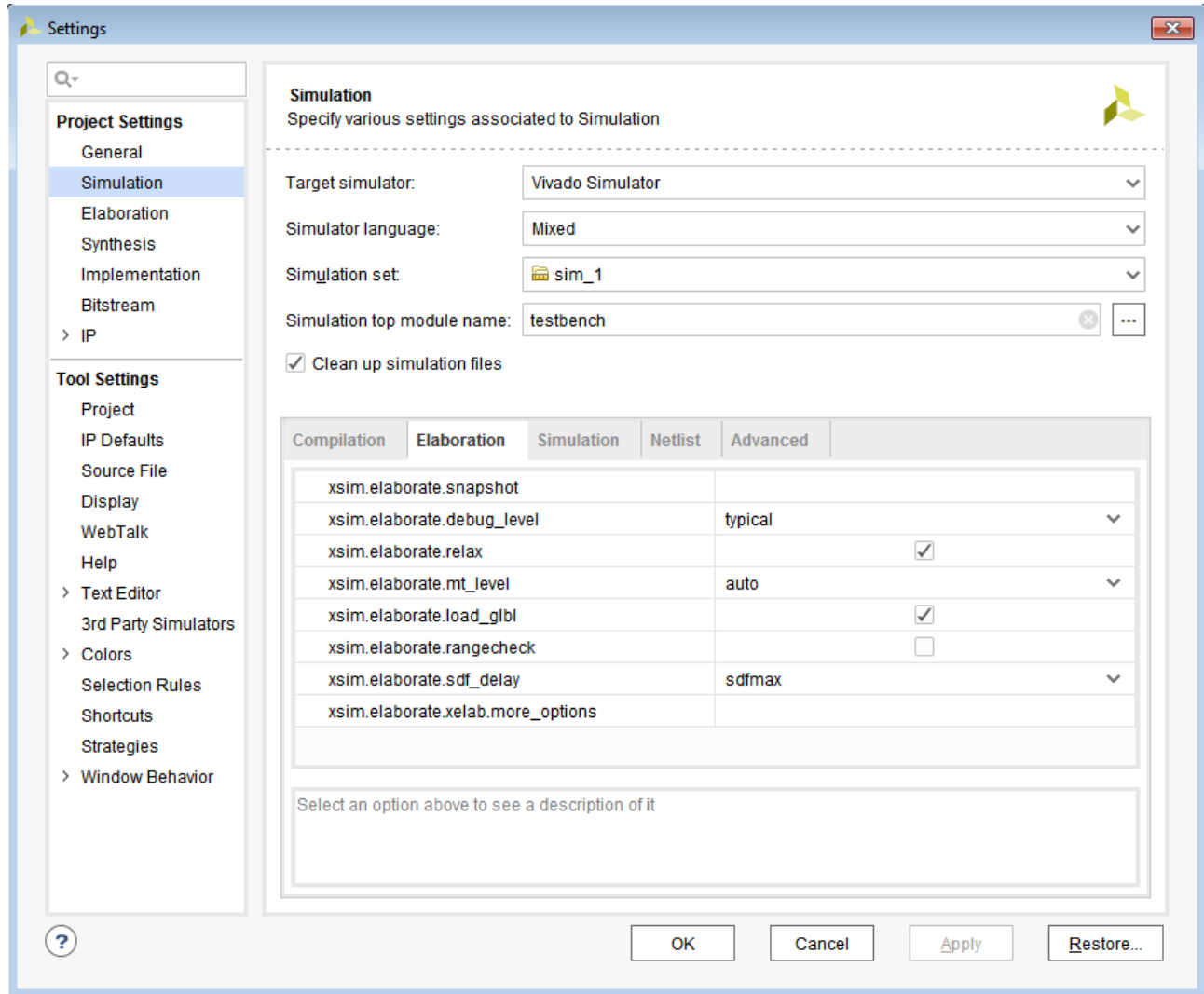


Figure 22: Simulation Settings – Elaboration Tab

4. In the Simulation tab enter the SAIF file name as `power_tutorial_func.saif` for **xsim.simulate.saif**. Observe that the **xsim.simulate.runtime** is 1000ns.
5. Click **OK**.

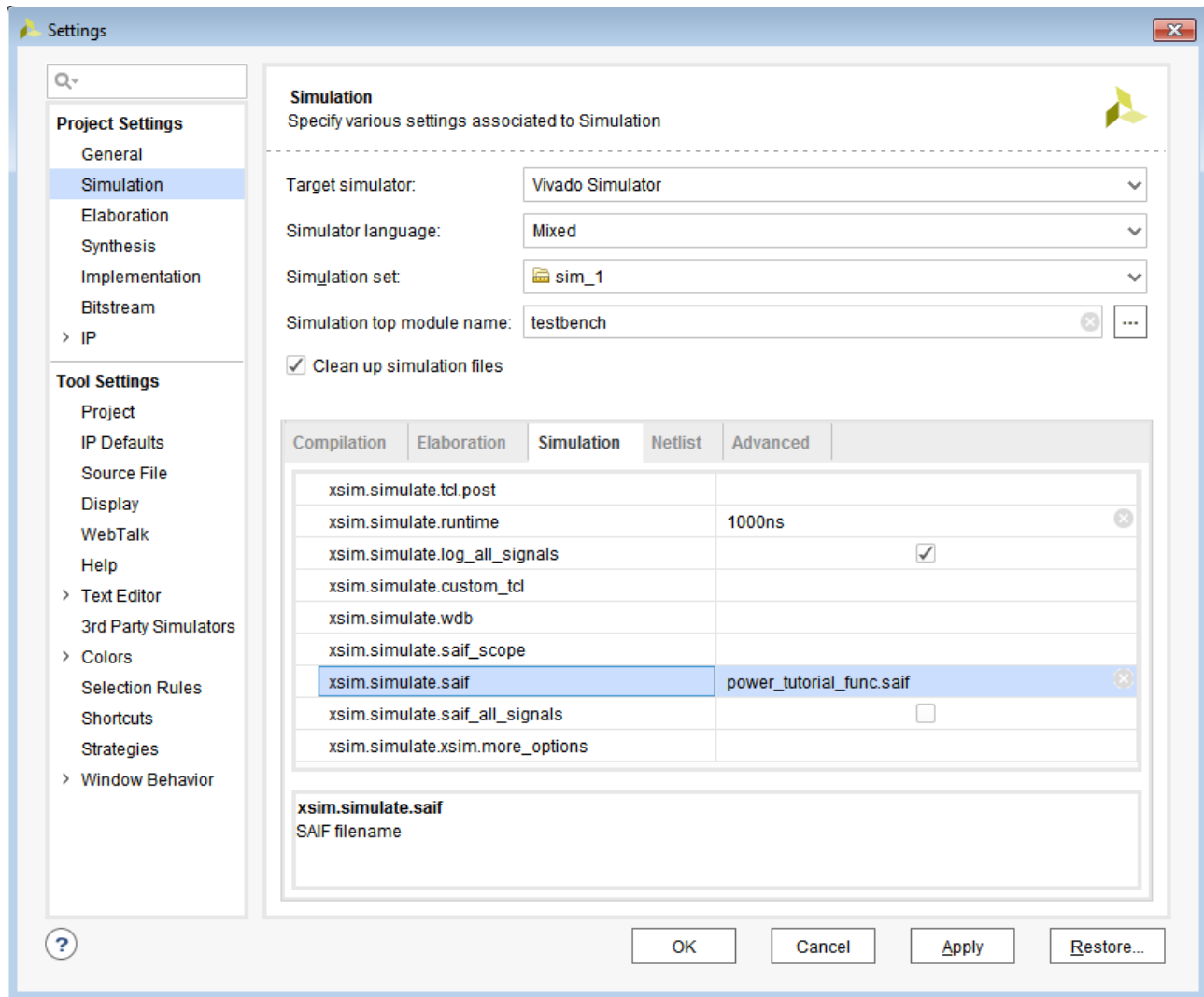


Figure 23: Simulation Settings – Entering SAIF File Setting

With the simulation settings properly configured, you can launch the Vivado simulator to perform a post-synthesis functional simulation of the design.

Note: The power reporting and analysis are not performed at the RTL level. They are performed at the gate level.

6. In the Flow Navigator, click **Run Simulation > Run Post-Synthesis Functional Simulation**.

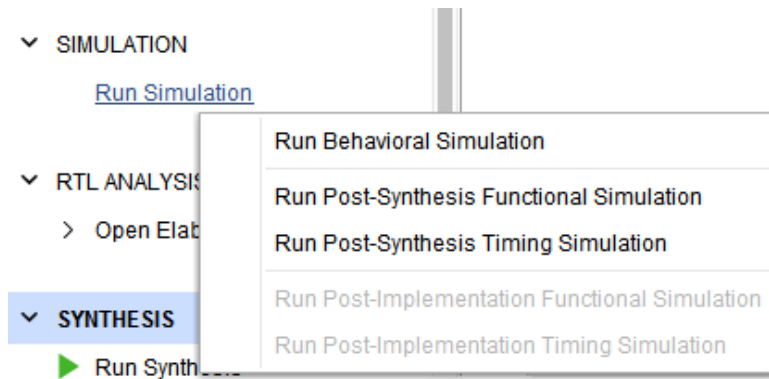


Figure 24: Running Post-Synthesis Functional Simulation from Flow Navigator

When you launch the Run Post-Synthesis Functional Simulation command, the Vivado simulator is invoked to run the simulation.

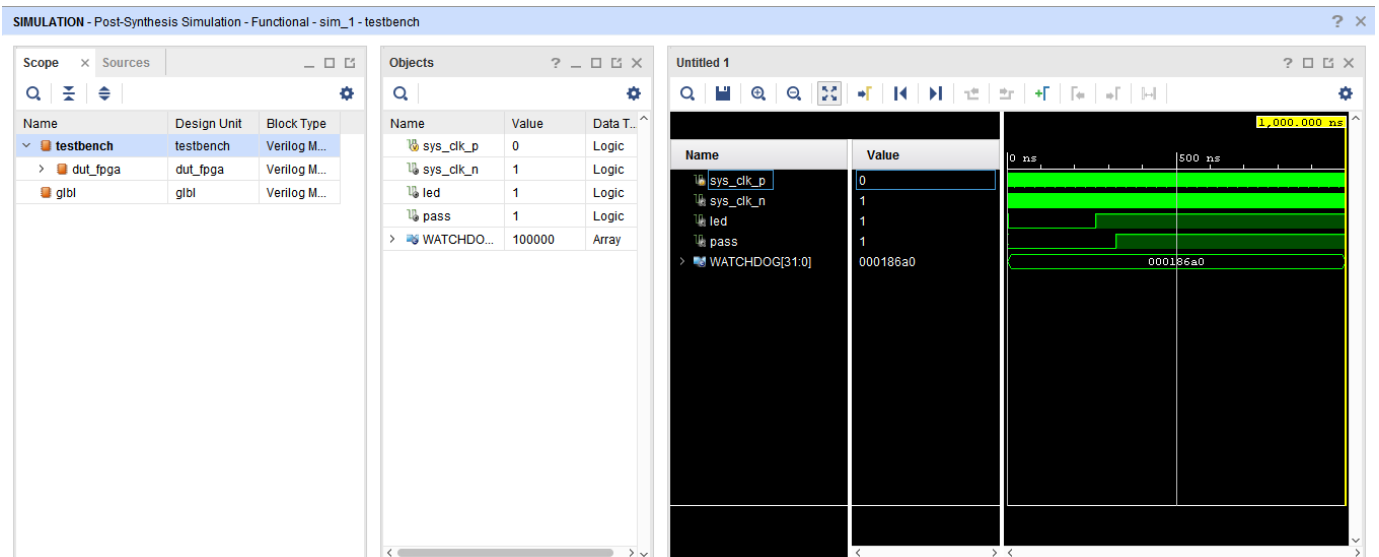


Figure 25: Post-Synthesis Functional Simulation Window in the Vivado IDE

After the simulation completes, click **x** at the top right corner to close the simulation window.

Step 8: Incorporating SAIF Data into Power Analysis

The SAIF output file requested in the simulation run is generated in the project directory. To further guide the power analysis algorithm we use this SAIF file.

1. Ensure the SAIF file requested is generated. Check to see that the SAIF file requested in the simulation settings prior to running simulation appears in this directory:


```
<project_directory>/power_tutorial1/power_tutorial1.sim/sim_1/synth/
func/power_tutorial_func.saif
```

2. In the Flow Navigator window, click on **Open Synthesized Design** to expand options.
3. From the Synthesized Design options, select **Report Power**.
4. In the Report Power dialog box, set the **Results name** to `power_3`
5. In the **Output** tab of Report Power dialog box, make the following changes:
 - o Set the **Output text File** to `power_3.pwr`
 - o Set the **Output XPE File** to `power_3.xpe`
6. In the **Environment** tab of Report Power dialog box, make sure that the **Process** is set to **maximum**.
7. In the **Switching** tab of Report Power dialog box, specify the SAIF file location.

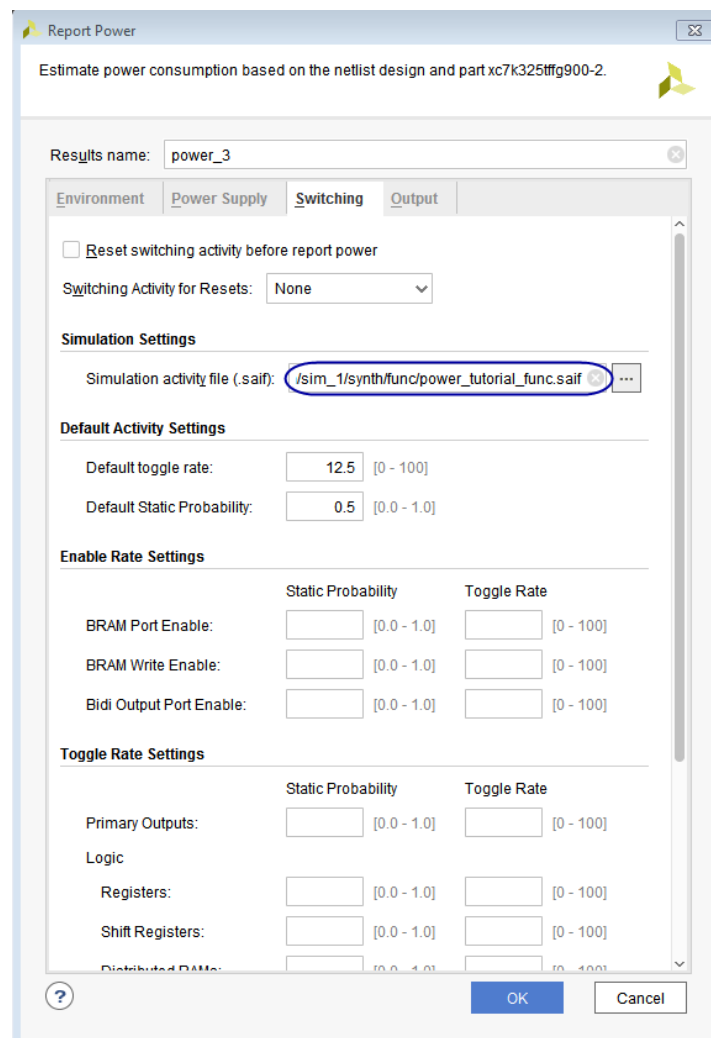


Figure 26: Specifying SAIF File Location

- Click **OK** in the Report Power dialog box.

The `report_power` command runs, and the Power Report **power_3** is generated in the Power tab of the results windows area.

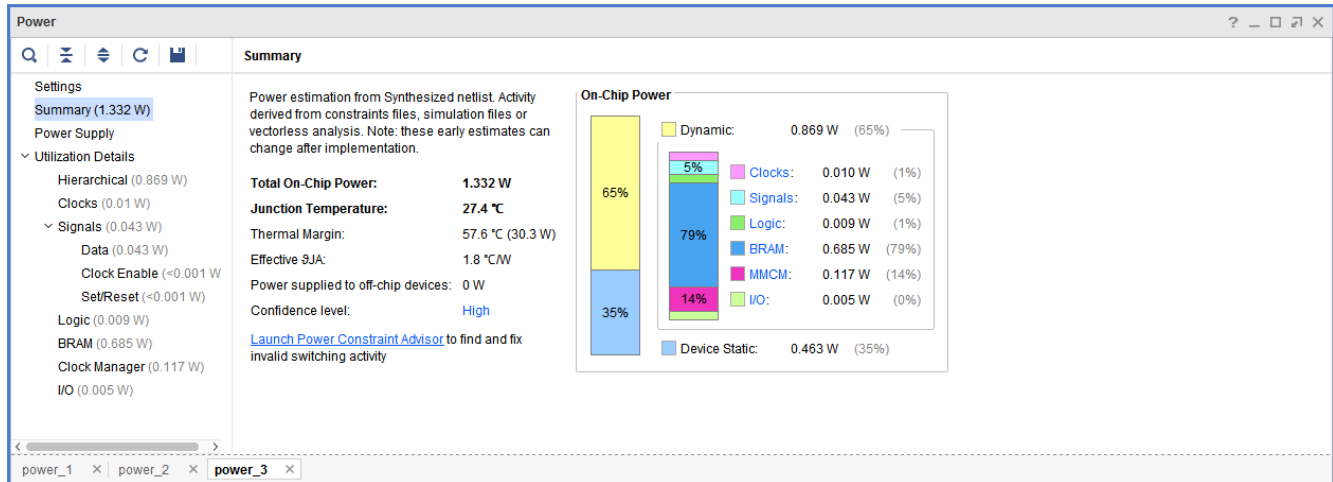


Figure 27: Power Report – Summary

Note: The SAIF annotation results are displayed in the Tcl Console. Make sure that all the design nets are matched with simulation nets, to achieve better accuracy by including Simulation data. For 7 series devices, the number of design nets and simulation nets may vary due to various reasons. The most common reason is that their hierarchical separators are different. Sometimes, the simulation nets may be lower down in the hierarchy level. However, they should match 100%.

Example: INFO: [Power 33-26] Design nets matched = 1894 of 1894

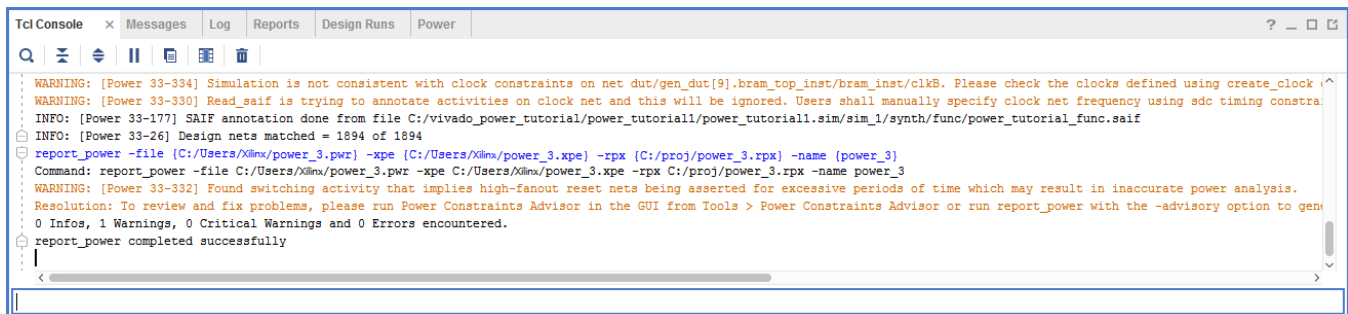


Figure 28: Tcl Console Log – Reading SAIF Data

- Go to the **I/O** view in the Power Report. Note that all the I/O port activity data has been set from simulation data we specified. The data is color coded to indicate activity rates read from the **Simulation** output file.

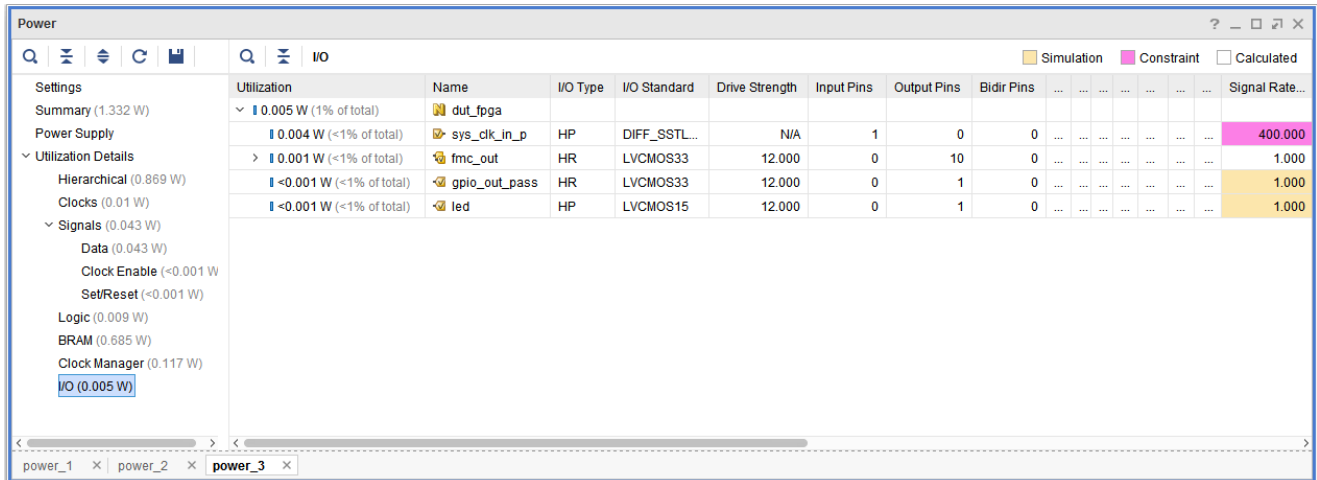


Figure 29: Power Report – I/O Display

10. Note the difference in total power numbers (**Total On-Chip Power** in the **Summary** view) between a pure vectorless run in the **power_1** results versus with the post synthesis functional simulation data in the **power_3** results. Also note that the **dut/dut_reset** signal rates are overwritten by simulation SAIF data.

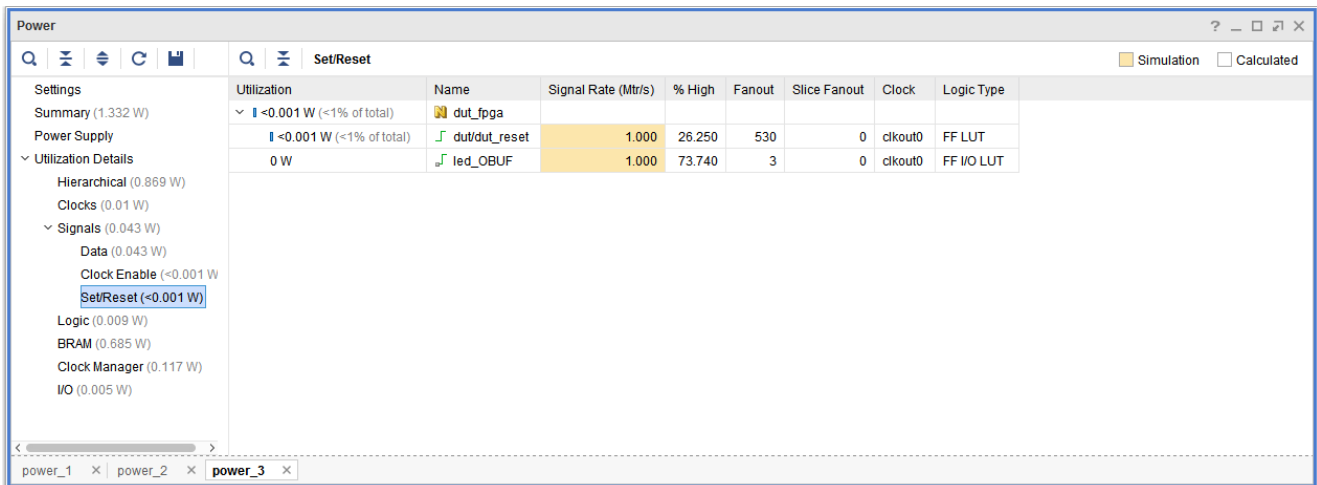


Figure 30: Power Report – Set/Reset Display

Step 9: Implementing the Design

This tutorial helps you understand Power Analysis with and without Power Optimization. In this step, you will run Implementation *without* Power Optimization.

1. In the Flow Navigator, right-click on **Implementation** and select **Implementation Settings**.
2. In the **Opt Design** settings, select the **NoBramPowerOpt** option for `-directive` and click **OK**.

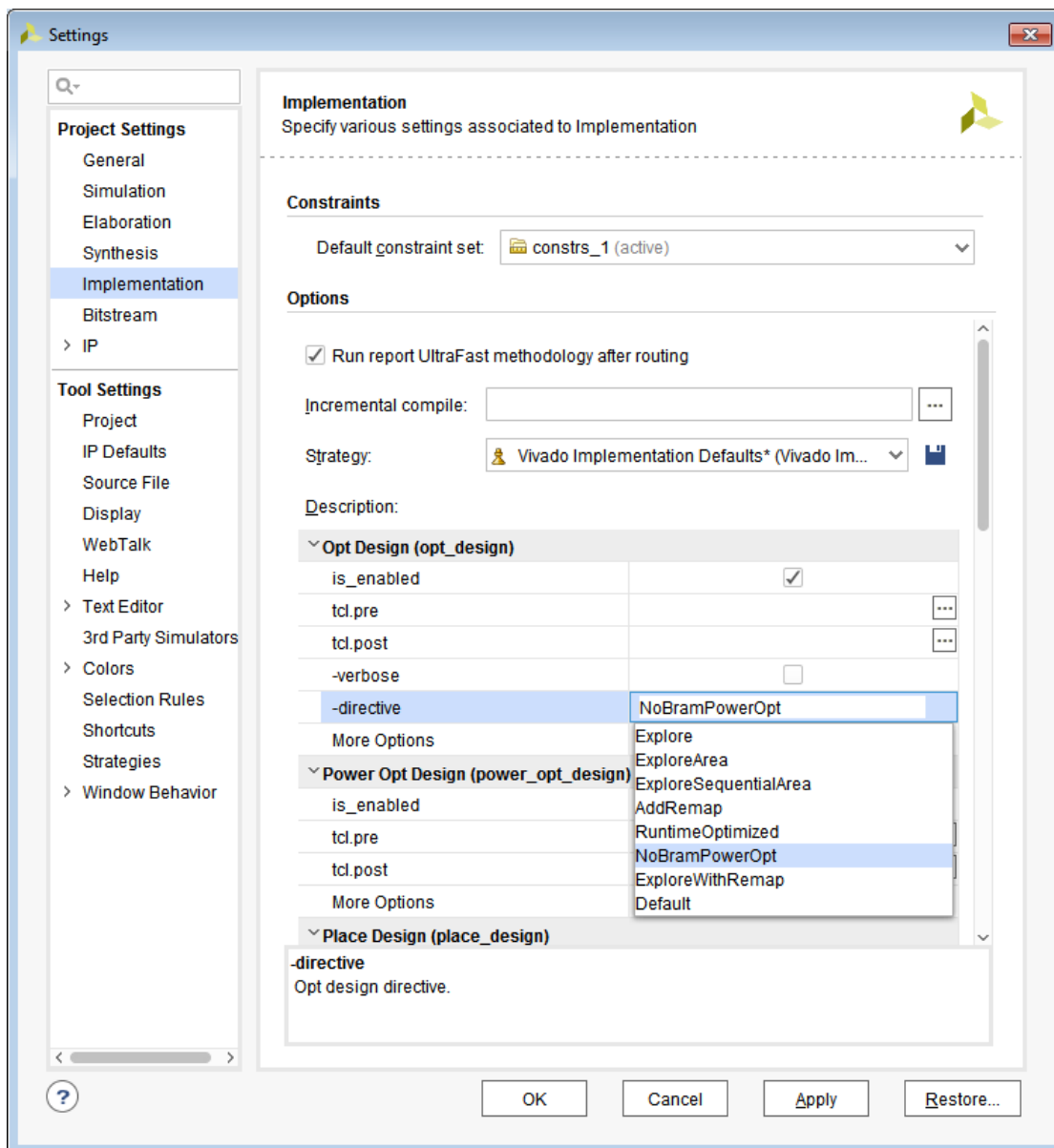


Figure 31: Disabling Power Optimization

3. In the Flow Navigator, click **Run Implementation**.

4. When prompted to **Save project before launching implementation** click **Don't Save**.

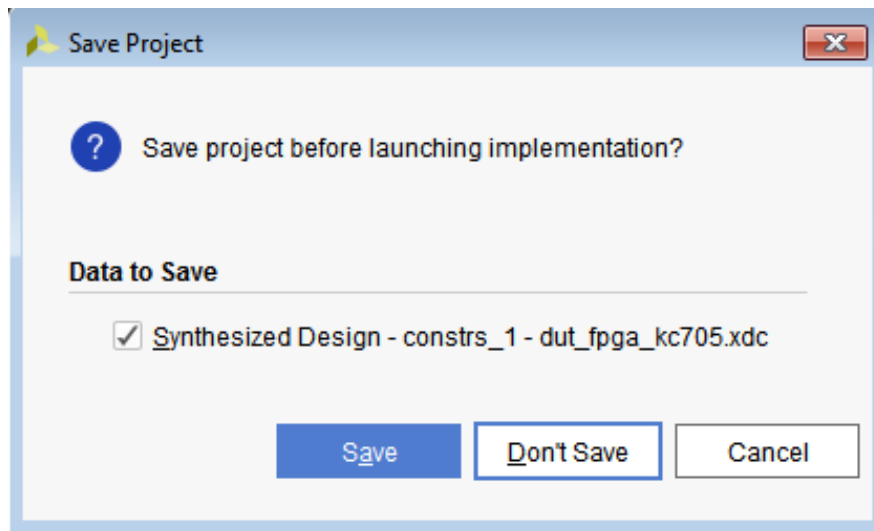


Figure 32: Click “Don't Save” Here

Conclusion

In Lab 1, you have learned how to set the power analysis in the Vivado IDE. In lab 2, you will learn about the timing simulation and its effect on the power analysis.

Lab 2: Running Timing Simulation and Estimating Power

Introduction

In this lab, you will learn about generating a SAIF file after running a timing level simulation using the Vivado® simulator and Questa Advanced Simulator. The lab will take you through the steps for SAIF file creation, running timing simulation, and estimating power using the SAIF data.

Step 1: Configuring and Running the Timing Simulation using Vivado Simulator

1. In the Implementation Complete dialog box, select **Open Implemented Design** and click **OK** to open the implemented design. When prompted to **Save project before opening an implemented design**, click **Don't Save**.

Now you are ready to set up and launch the Vivado simulator to run post implementation timing simulation. You will set the timing simulation properties in the Vivado IDE, then run the timing simulation.

2. In the Flow Navigator, click **Settings** and select **Simulation** tab to set the timing simulation properties.
3. In the Project Settings dialog box, note that the following defaults are automatically set:

Simulation set: `sim_1`

Simulation top-module name: `testbench`

4. In the Elaboration tab, make sure that **debug_level** is set to **typical**, which is the default value.
5. In the Simulation tab set the SAIF filename **xsim.simulate.saif** to `power_tutorial_timing_xsim.saif`.
6. Set the **xsim.simulate.saif_scope** to `testbench/dut_fpga`.
7. Observe that the **simulation run time xsim.simulate.runtime** is 1000ns.
8. Click **OK**.

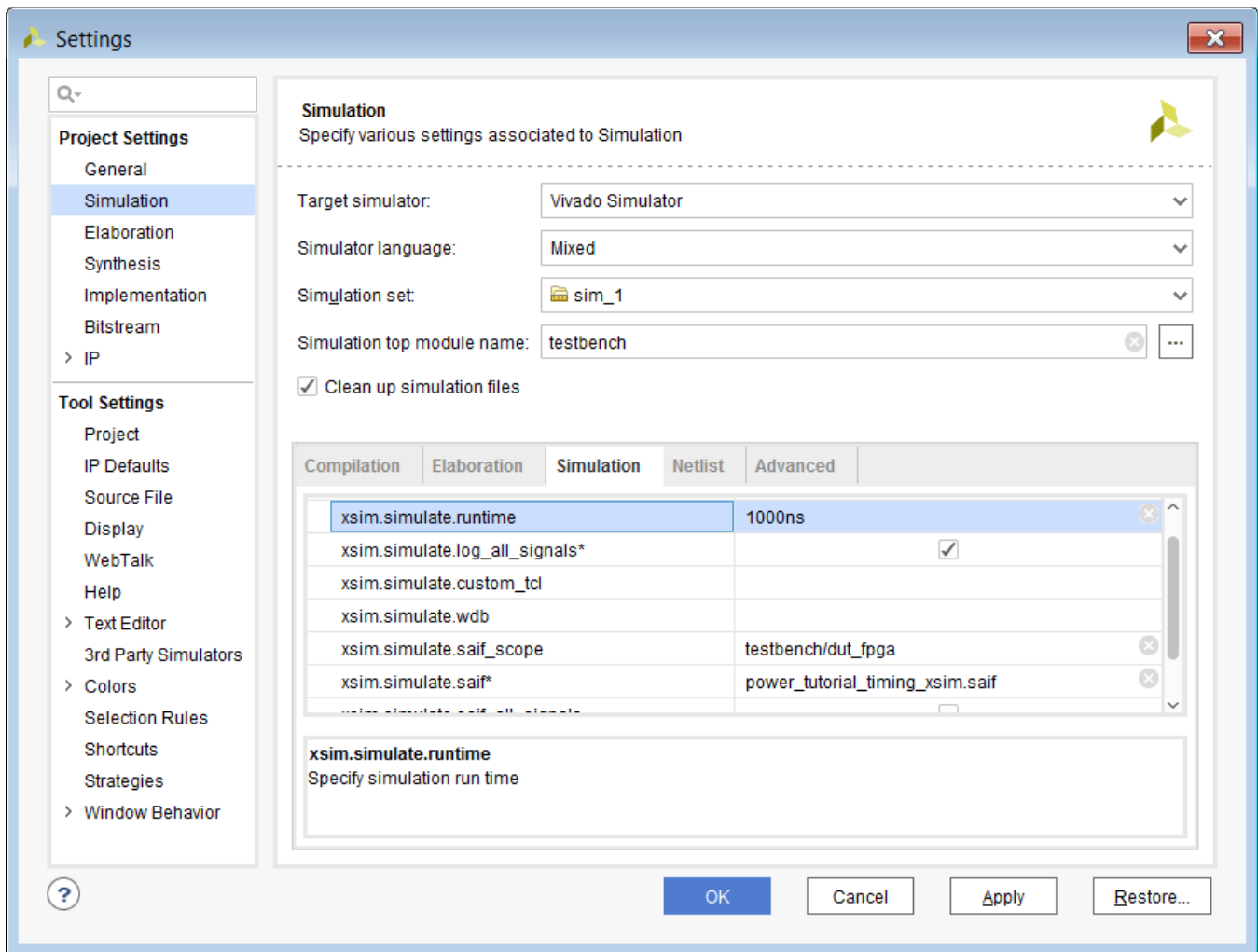


Figure 33: Timing Simulation Settings – Simulation Tab

With the simulation settings properly configured, you can launch the Vivado simulator to perform a timing simulation of the post implemented design.

9. In the Flow Navigator, click **Run Simulation > Run Post-Implementation Timing Simulation**.

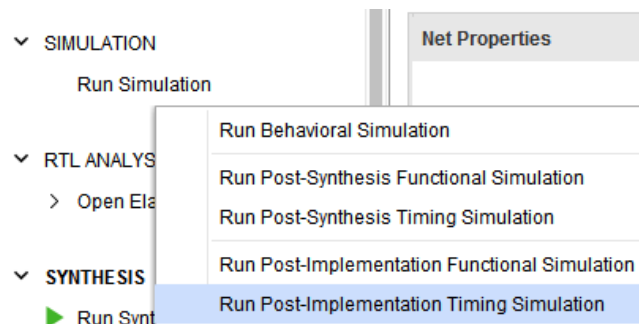


Figure 34: Running Post-Implementation Timing Simulation

10. After the Vivado simulator has finished simulating the design, ensure that the SAIF file requested has been generated. Check to see that the SAIF file requested in the simulation settings prior to running simulation appears in this directory:

```
<project_directory>/power_tutorial1/power_tutorial1.sim/  
sim_1/impl/timing/power_tutorial_timing_xsim.saif
```

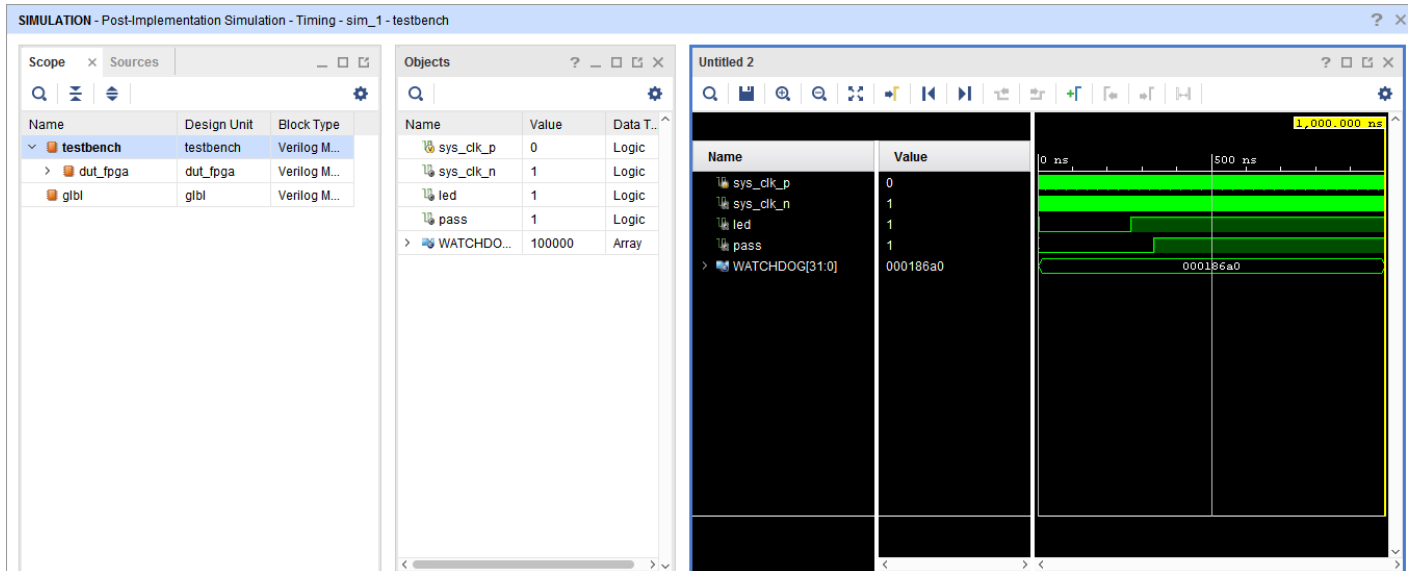


Figure 35: Post-Implementation Timing Simulation using the Vivado Simulator

Step 2: Running Report Power in Vectorless Mode

1. In the Flow Navigator, select **Open Implemented Design > Report Power** to open the Report Power dialog box.

You can also select **Reports > Report Power** from the main menu bar.

2. In the Report Power dialog box **Environment** tab, make sure the **Process** is set to **maximum** and click **OK**.

The Report Power command creates a Power Report under the **power_1** tab in the results windows area.

3. Note the total power (**Total On-Chip Power**) in the power report **Summary** page (Figure 36).

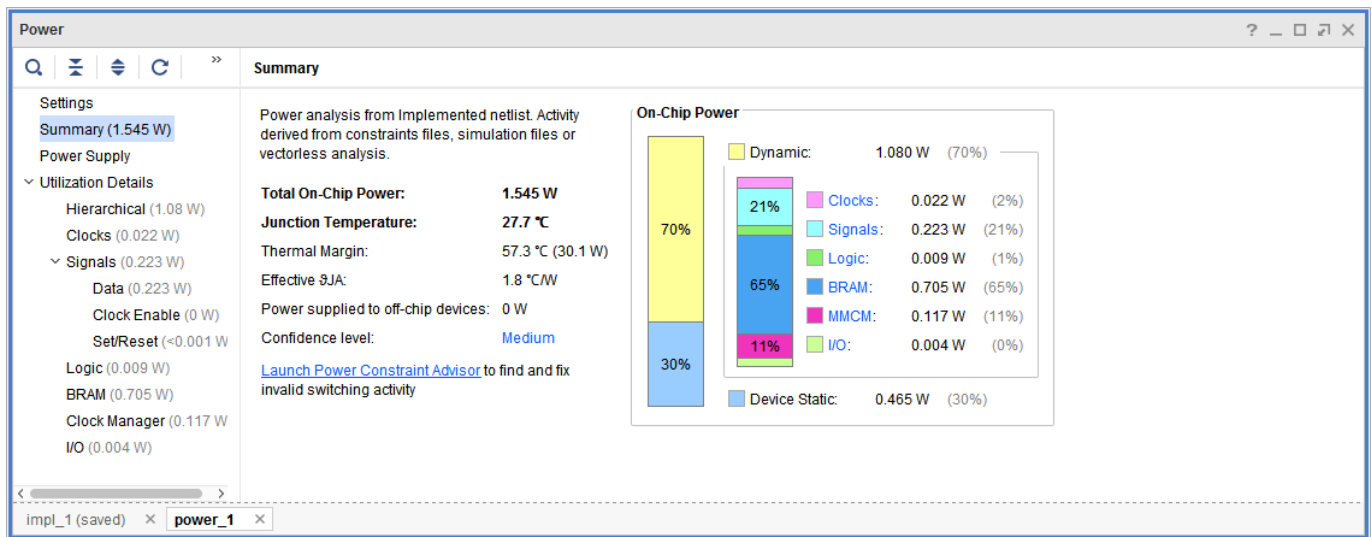


Figure 36: Running Post-Implementation Report Power in Vectorless Mode

Vectorless Analysis is done based on default switching activity specification on the primary ports and the design clocks.

Refer to the *Vivado Design Suite User Guide: Power Analysis and Optimization* ([UG907](#)) for more information on Vectorless Power Analysis.

Step 3: Running Report Power with Vivado Simulator SAIF Data

The project directory contains the SAIF output file requested in the previous timing simulation run. We use this SAIF file – a “Switching Activity Interchange Format” file – to further guide the power analysis algorithm.

1. In the main menu bar, select **Reports > Report Power**.
2. In the Report Power dialog box, specify the SAIF file location in the **Switching** tab.

The SAIF file, which was requested in the simulation settings prior to running timing simulation, should appear here:

```
<project_directory>/power_tutorial1/power_tutorial1.sim/  
sim_1/impl/timing/power_tutorial_timing_xsim.saif
```

3. Click **OK** in the Report Power dialog box.

After the Report Power command completes, the **Power** tab in the results windows area displays Power Report **power_2**.

In the Tcl console, observe that the SAIF file is read successfully and that 100% of the design nets are matched. This assures you that the generated SAIF file is correct and matched with all design nets.

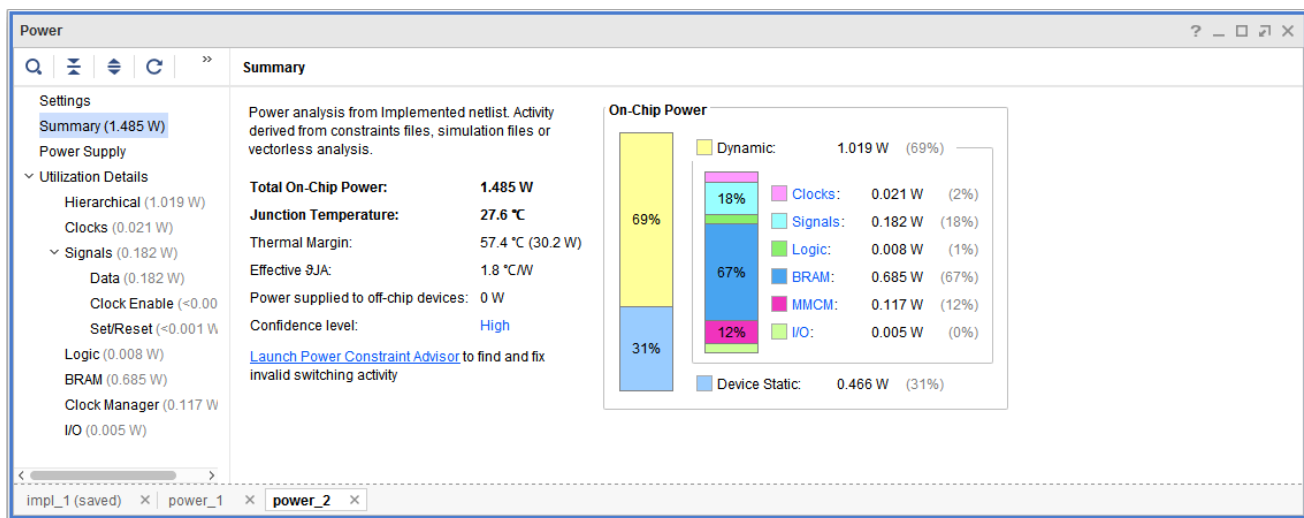
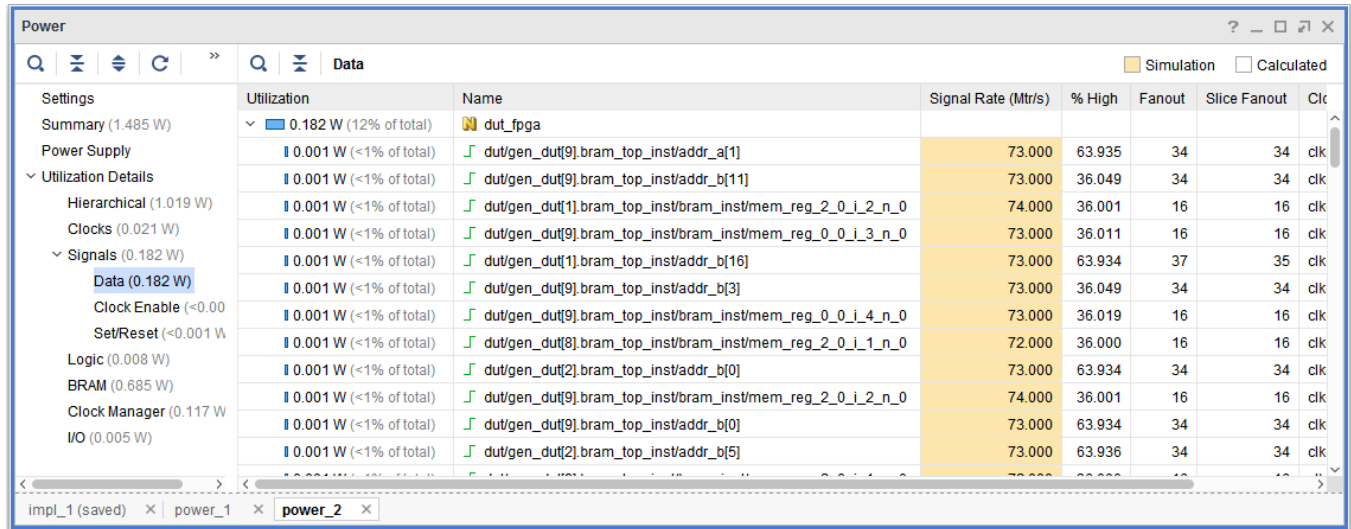


Figure 37: Power Report with SAIF Data

4. Note the change in total power (**Total On-Chip Power** in the **Summary** view) in the **power_2** report compared to the **power_1** report. The total power estimated in the report generated with SAIF file data will be different than the total power estimated in the vectorless run (**power_1** results).
5. Examine the summary and block level (**On-Chip Power**) power distribution in the **Summary** view of the Power Report.

- Go to the **Utilization Details > Signals > Data** view in the Power Report. Note that all the **Signal Rate** data has been set from simulation data the SAIF file provided.

The data is color coded to indicate activity rates read from the **Simulation** output file.



Settings	Utilization	Name	Signal Rate (Mtr/s)	% High	Fanout	Slice Fanout	Clk
Summary (1.485 W)	0.182 W (12% of total)	dut_fpga					
Power Supply	0.001 W (<1% of total)	dut/gen_dut[9].bram_top_inst/addr_a[1]	73.000	63.935	34	34	clk
Utilization Details	0.001 W (<1% of total)	dut/gen_dut[9].bram_top_inst/addr_b[11]	73.000	36.049	34	34	clk
Hierarchical (1.019 W)	0.001 W (<1% of total)	dut/gen_dut[1].bram_top_inst/bram_inst/mem_reg_2_0_i_2_n_0	74.000	36.001	16	16	clk
Clocks (0.021 W)	0.001 W (<1% of total)	dut/gen_dut[9].bram_top_inst/bram_inst/mem_reg_0_0_i_3_n_0	73.000	36.011	16	16	clk
Signals (0.182 W)	0.001 W (<1% of total)	dut/gen_dut[1].bram_top_inst/addr_b[16]	73.000	63.934	37	35	clk
Data (0.182 W)	0.001 W (<1% of total)	dut/gen_dut[9].bram_top_inst/addr_b[3]	73.000	36.049	34	34	clk
Clock Enable (<0.00 W)	0.001 W (<1% of total)	dut/gen_dut[9].bram_top_inst/bram_inst/mem_reg_0_0_i_4_n_0	73.000	36.019	16	16	clk
Set/Reset (<0.001 W)	0.001 W (<1% of total)	dut/gen_dut[8].bram_top_inst/bram_inst/mem_reg_2_0_i_1_n_0	72.000	36.000	16	16	clk
Logic (0.008 W)	0.001 W (<1% of total)	dut/gen_dut[2].bram_top_inst/addr_b[0]	73.000	63.934	34	34	clk
BRAM (0.685 W)	0.001 W (<1% of total)	dut/gen_dut[9].bram_top_inst/bram_inst/mem_reg_2_0_i_2_n_0	74.000	36.001	16	16	clk
Clock Manager (0.117 W)	0.001 W (<1% of total)	dut/gen_dut[9].bram_top_inst/addr_b[0]	73.000	63.934	34	34	clk
I/O (0.005 W)	0.001 W (<1% of total)	dut/gen_dut[2].bram_top_inst/addr_b[5]	73.000	63.936	34	34	clk

Figure 38: Power Report Signals > Data View - Simulation SAIF Data for Signal Rates

- In the **Summary** view of the **power_1** report (the report generated by the vectorless analysis), click on **Confidence level (High)** in Figure 39).

The Confidence Level is a measurement of the accuracy and the completeness of the input data that the Report Power uses while performing power analysis.

Notice that the **Confidence Level** is High for the vectorless analysis since less than 25% of internal nodes are user specified for **Internal Activity**.

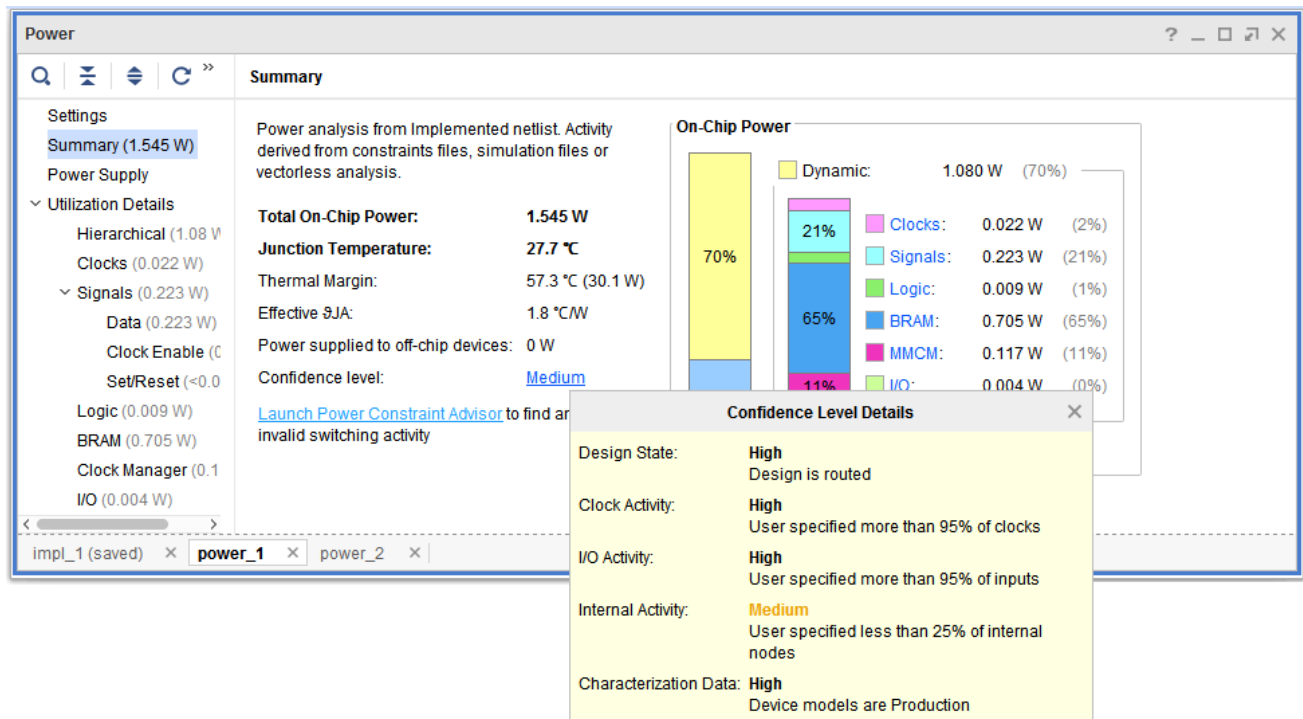


Figure 39: Confidence Level for Vectorless Power Analysis (power_1)

- In the **Summary** view of the **power_2** report (the report generated by the analysis for which you specified a SAIF file as input), click on **Confidence level (High)** in [Figure 40](#).

Notice that the **Confidence Level** has increased to **High**, since more than 25% of internal nodes are user specified for **Internal Activity**.

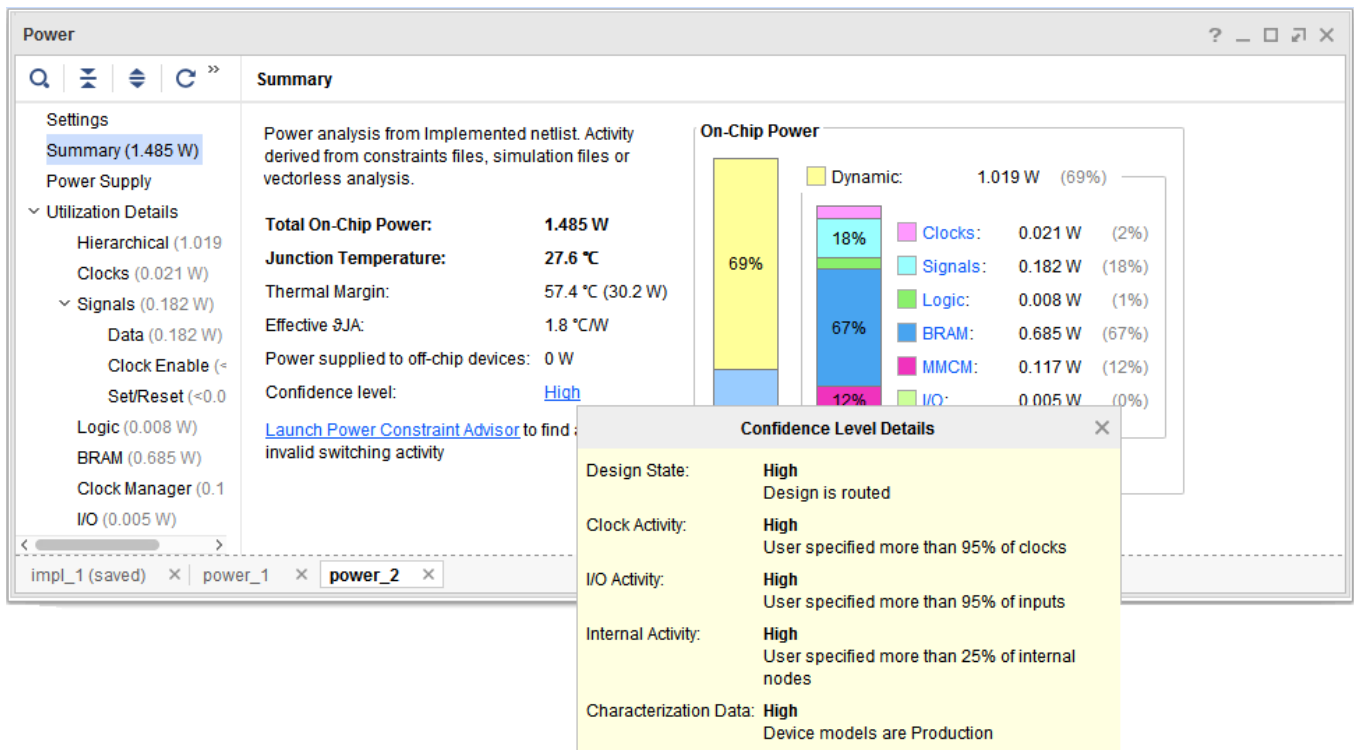


Figure 40: Confidence Level for Power Analysis with SAIF Input (power_2)

Generating a SAIF File using Questa Advanced Simulator

The following steps will take you through the process of SAIF file creation, running timing simulation, and estimating power using the SAIF data using Questa Advanced Simulator.

IMPORTANT: Make sure the Vivado Design Suite knows where to pick up the Questa Advanced Simulator tool. You can either:

Manually set the path to ModelSim/Questa Advanced Simulator using the `$PATH` environment variable

or

From the **Tools > Settings > Tool Settings** section, define the path to the simulator in the Vivado IDE under the **3rd Party Tools** tab: **Questa Advanced Simulator install path**.

Make sure the **Compiled library location** points to a valid location for the compiled Xilinx simulation libraries.

To create new compiled libraries:

1. In the 3rd Party Simulators tab, specify the **Compiled library location to Questa Advanced Simulator**. Use the **Compiled library location** specified during the compiled library generation and enter it in the **Compiled library location** field of 3rd Party Simulators tab. It should point to the `compile_simlib` directory.
2. Click **OK** to define the path and generate compiled libraries.

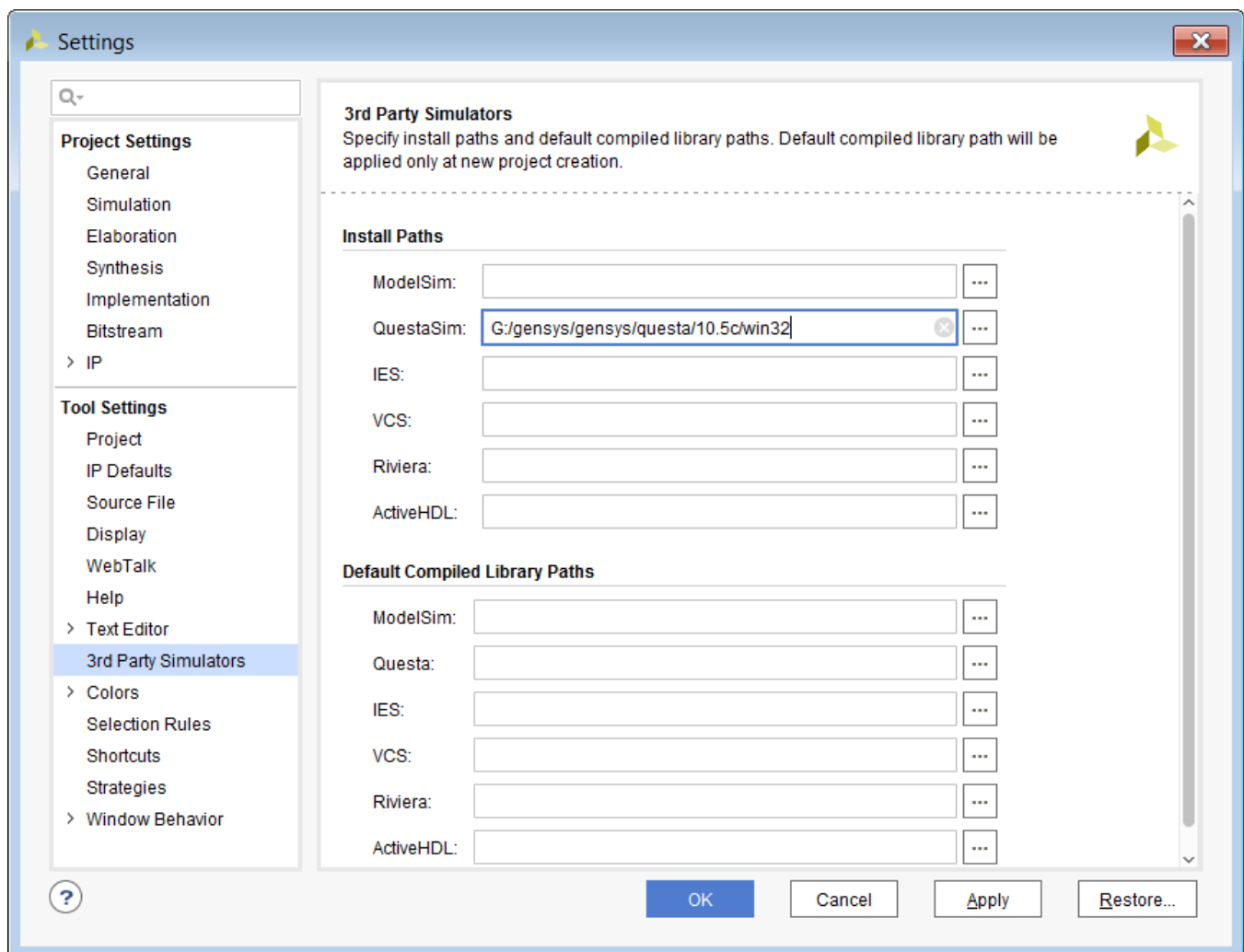


Figure 41: Setting the Questa Advanced Simulator Installation Path and compiled libraries

Step 1: Configuring and Running Timing Simulation in Questa Advanced Simulator

Now you are ready to set up and launch the Questa Advanced Simulator to run post-implementation timing simulation. You will set the timing simulation properties in the Vivado IDE, and run the timing simulation.

1. In the Flow Navigator, right-click **Simulation** to select **Simulation Settings**. Set the timing simulation properties.
2. In the Simulation Settings tab, set the **Target simulator** to **Questa Advance Simulator**.
3. Click **Yes** to **change your target simulator to Questa Advance Simulator**.

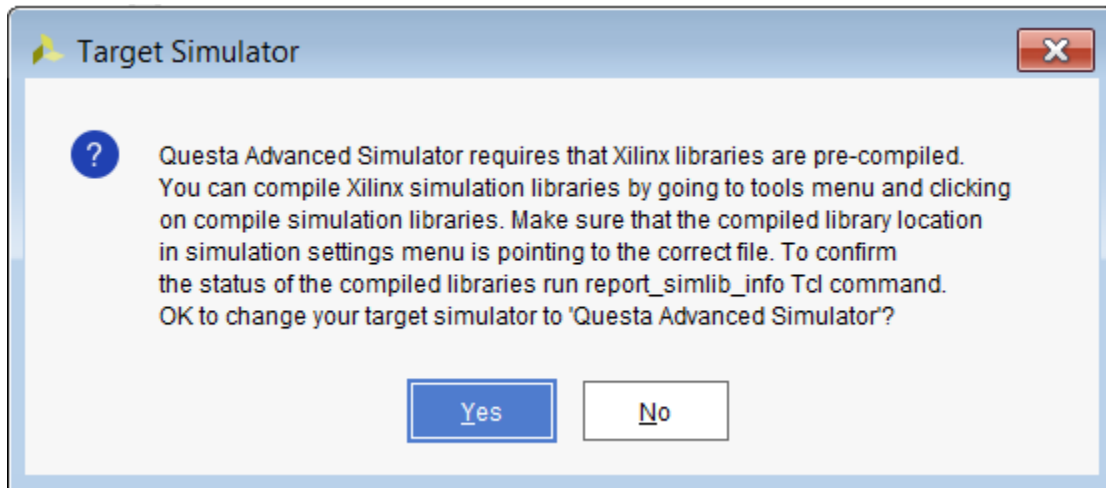


Figure 42: Simulation Settings – Changing the Target Simulator

4. Set the **questa.simulate.saif** to `power_tutorial_timing_questasim.saif`.
5. Set the **questa.simulate.saif_scope** to `testbench/dut_fpga`.
6. Note that the **questa.simulate.runtime** is 1000ns.

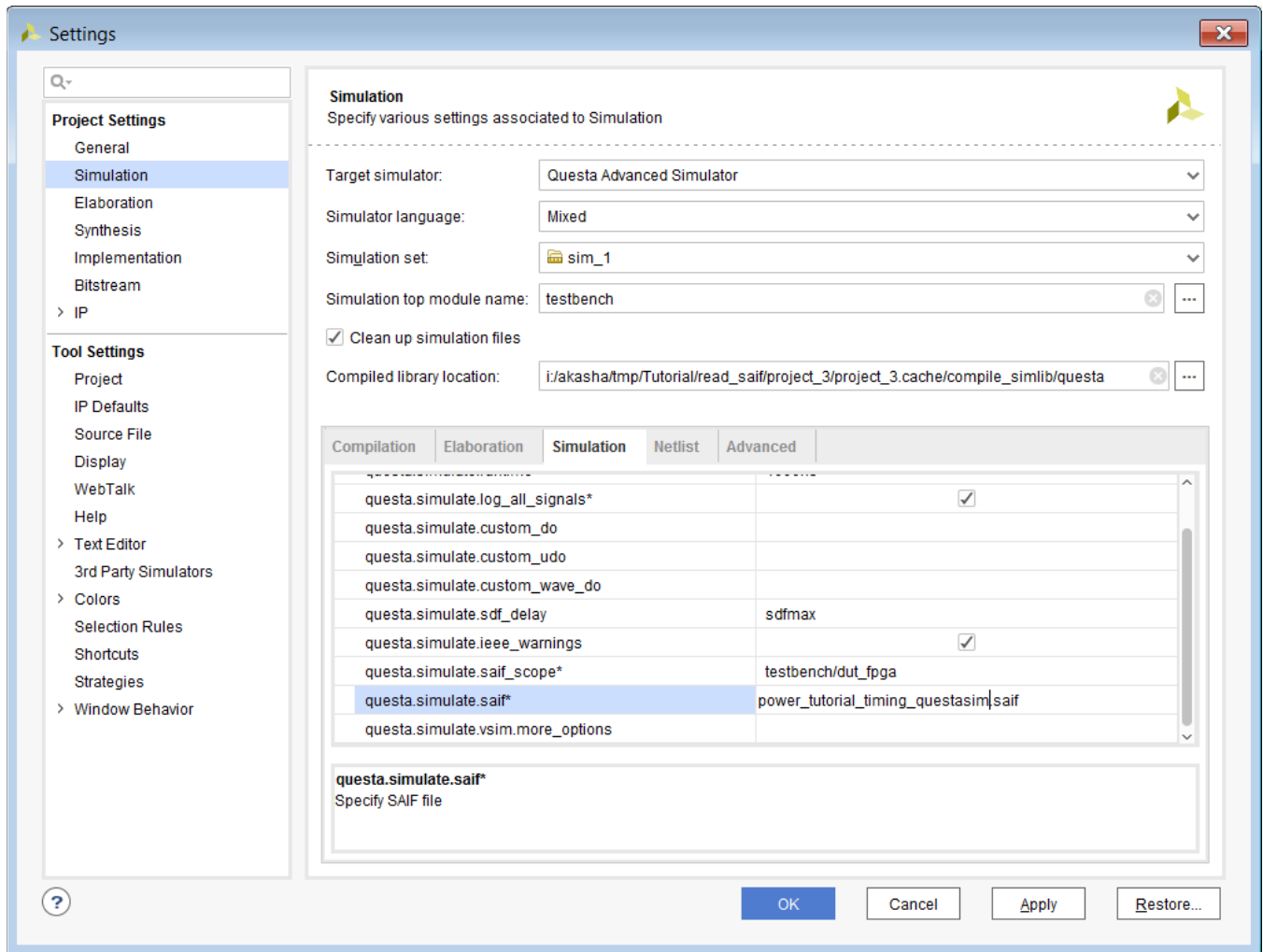


Figure 43: Configuring SAIF file Settings

7. Click **OK**. With the simulation settings properly configured, you can launch the Questa Advanced Simulator to perform a timing simulation of the design.
8. In the Flow Navigator, click **Run Simulation > Run Post-Implementation Timing Simulation**.

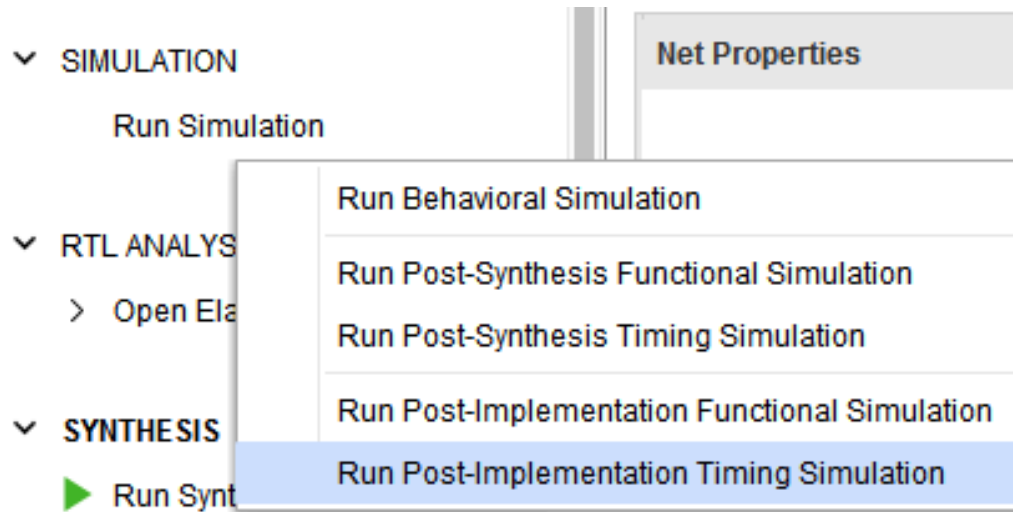


Figure 44: Running Post-Implementation Timing Simulation

A separate Questa Advanced Simulator GUI opens and starts simulating the design.

9. After the Questa Advanced Simulator has finished simulating the design, make sure the SAIF file requested has been generated. Check to see that the SAIF file requested in the simulation settings prior to running simulation appears in this directory:

```
<project_directory>/power_tutorial1/power_tutorial1.sim/  
sim_1/impl/timing/power_tutorial_timing_questasim.saif
```

Step 2: Running Report Power in Vectorless Mode



IMPORTANT: If SAIF based report_power has already been run in this session, run the `reset_switching_activity -all` command in the Tcl Console. This will clear the SAIF data in the power engine from the earlier runs.

1. Close any open Report Power views.
2. In the Flow Navigator, select **Implemented Design > Report Power** to open the Report Power dialog box.

Alternatively, select **Reports > Report Power** in the main menu bar.

3. In the Report Power dialog box, make the following settings:
 - Specify the **Results name** as `power_1`
 - In the **Environment** tab, Set the **Process** to **maximum**
 - In the **Switching** tab, leave the **Simulation activity file** empty
4. Verify that all the input settings are correct and click **OK**.

The Report Power command creates a Power Report under the **power_1** tab in the results windows area. Note that the total power for Vectorless Analysis runs with default switching rates.

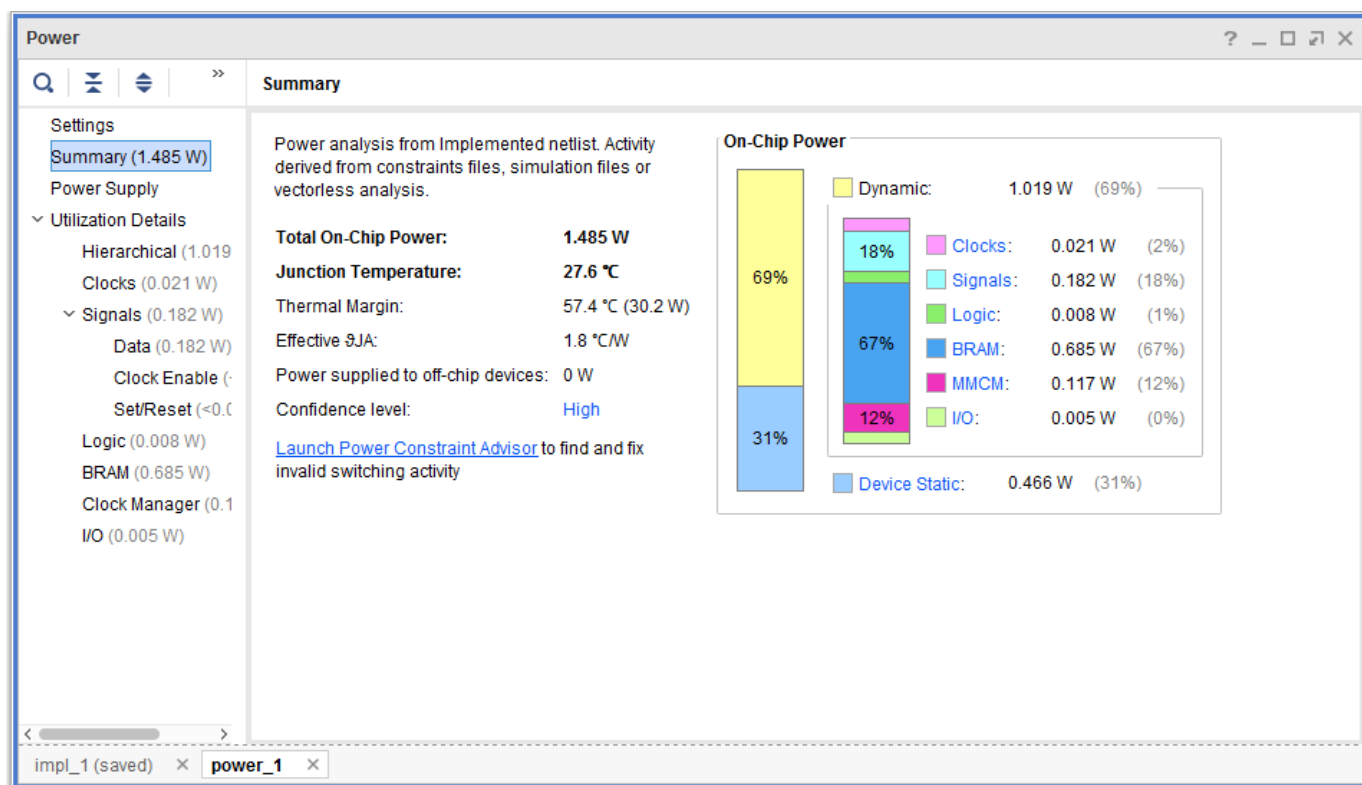


Figure 45: Power Report – Vectorless Mode Run

Step 3: Running Report Power with Questa Advanced Simulator SAIF Data

The SAIF output file requested in the simulation run has been generated under the project directory. We use this SAIF file – a “Switching Activity Interchange Format” file – to further guide the power estimation algorithm.

1. In the main menu bar, select **Reports > Report Power**.
2. In the Report Power dialog box, specify the SAIF file location in the **Switching** tab.

The SAIF file, which was requested in the simulation settings prior to running simulation, should appear here:

```
<project_directory>/power_tutorial1/power_tutorial1.sim/  
sim_1/impl/timing/power_tutorial_timing_questasim.saif
```

3. Click **OK** in the Report Power dialog box.

The Report Power command runs, and the Power Report **power_2** is generated in the **Power** tab of the results windows area.

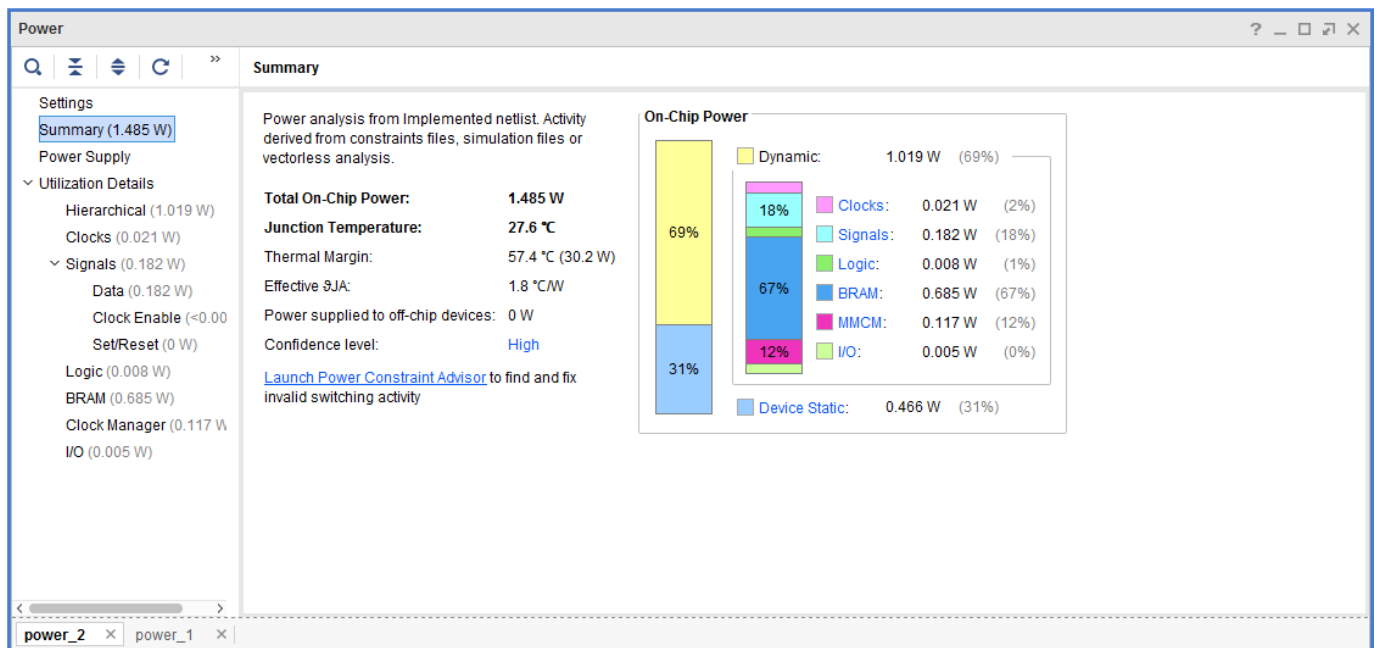


Figure 46: Power Report – With SAIF File Data

4. In the Tcl console, observe the read_saif results. This shows the percentage of design nets matched with simulation SAIF. This is important for accurate power analysis.
5. Go to the **Signals > Data** view in the Power Report and scroll to the right. Note that all the Signal Rate data has been set from simulation SAIF data you provided.

The data is color coded to indicate activity rates read from the **Simulation** output file.

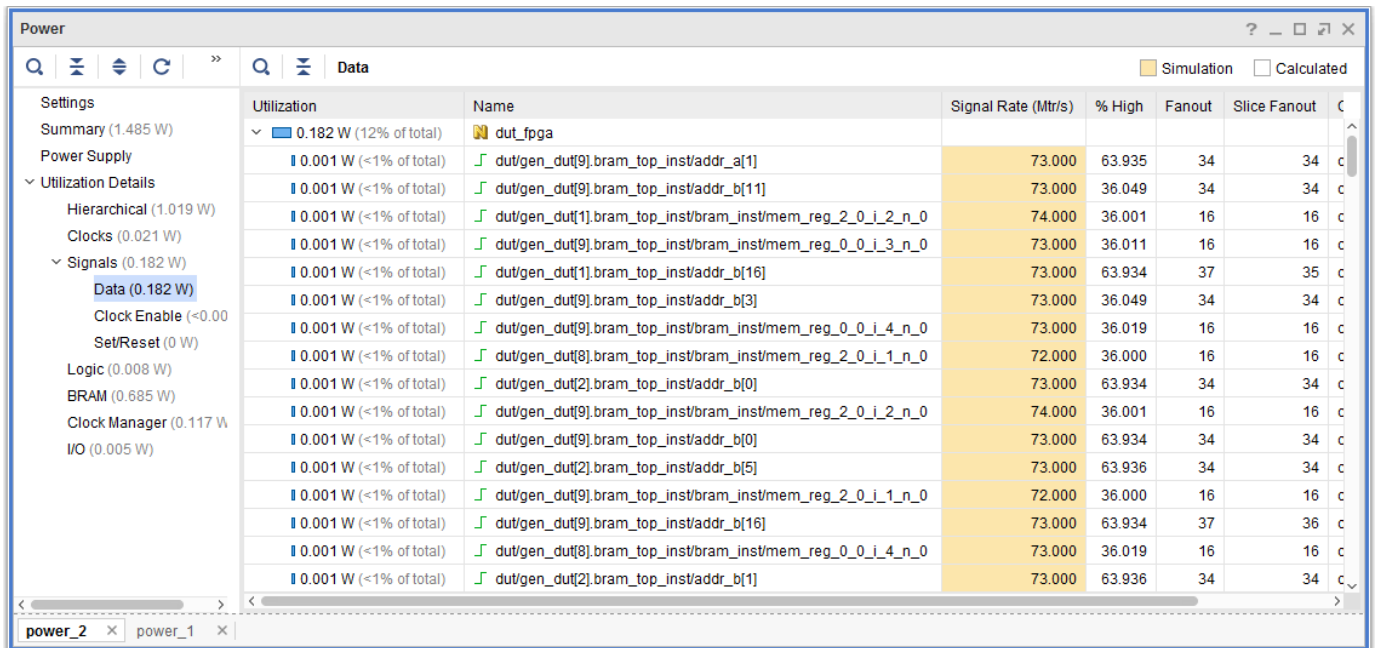


Figure 47: Signals > Data View with SAIF File Data

- Note the change in total power (**Total On-Chip Power** in the **Summary** view) in the **power_2** report compared to the **power_1** report. The total power estimated in the report generated with SAIF file data will be different than the total power estimated in the vectorless run (**power_1** results).

Conclusion

In Lab 2, you have learned how to generate a SAIF file after running a timing level simulation using a Vivado Simulator and Questa Advanced Simulator.

In Lab 3, you will learn about basic hardware power measurement technique using the KC705 Evaluation Board and correlating the hardware power numbers with the numbers generated by Vivado Report Power.

Lab 3: Measuring Hardware Power Using the KC705 Evaluation Board

Introduction

In this lab, you will learn about basic hardware power measurement technique and correlating the hardware power numbers with the numbers generated by Vivado® Report Power using KC705 evaluation board for 7 series devices. The lab will take you through the steps for setting up the hardware measurement, programing a bit file using Vivado Hardware Manager and power measurement through Texas Instruments (TI) Fusion Design Software. It also includes Junction Temperature reading from Vivado System Monitor.

Step 1: Generating a Bit File from the Implemented Design (Non-Power Optimization)

1. In the Vivado Design Suite, open the 7 series Implemented design.
2. In the Flow Navigator, click **Generate Bitstream**.

The Bitstream Generation Completed dialog box appears after the bitstream has been generated.

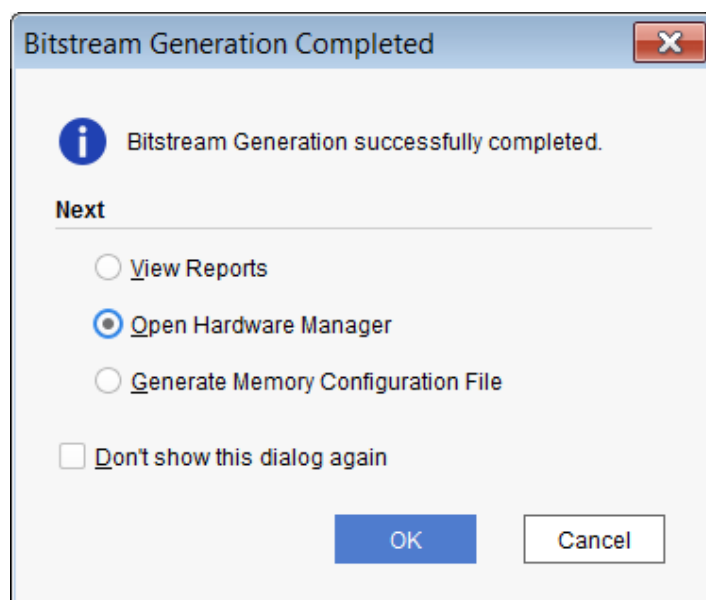


Figure 48: Bitstream Generation Completed Dialog Box

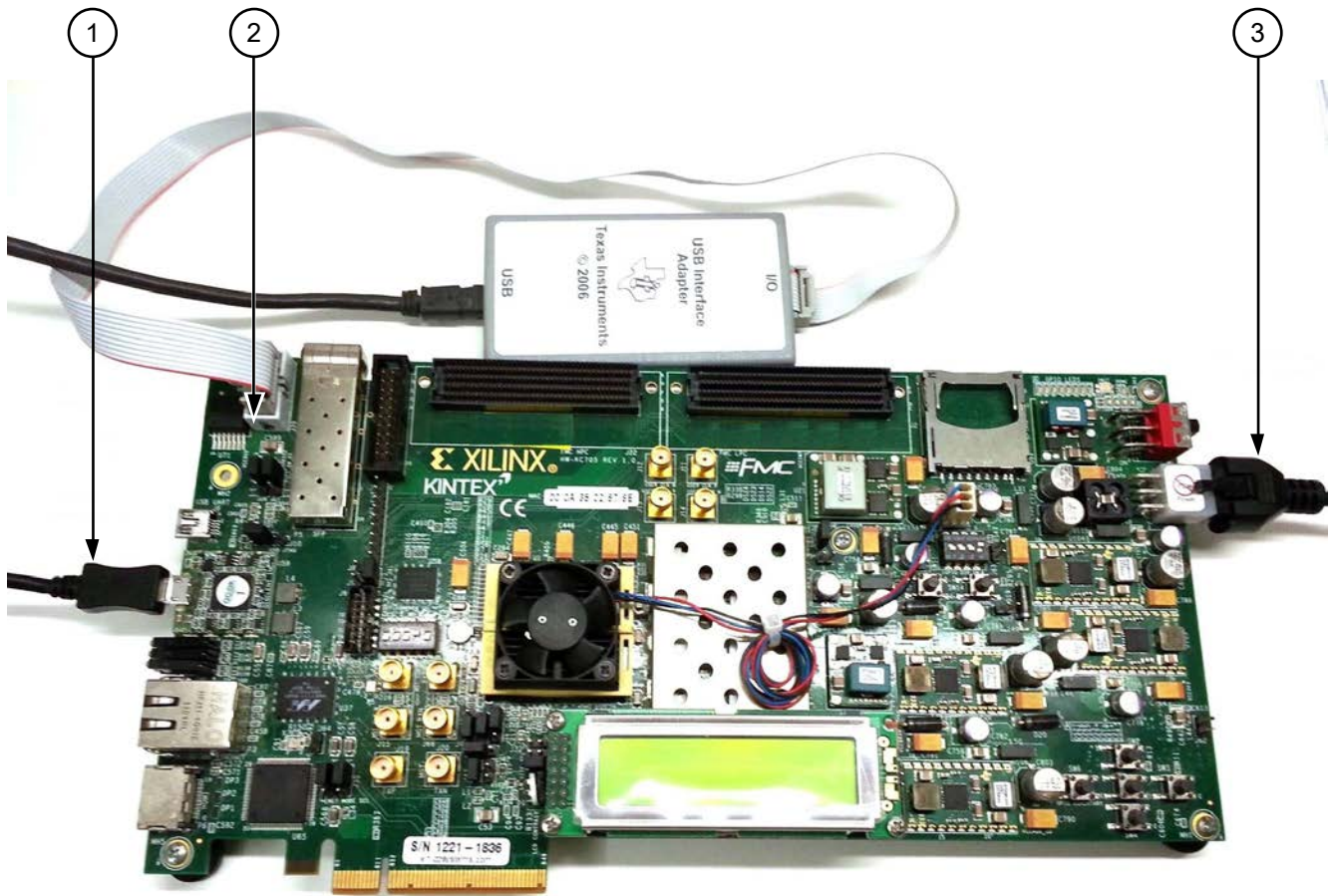
- Open the Hardware Manager by selecting **Open Hardware Manager** in the Bitstream Generation Completed dialog box and clicking **OK**.

Step 2: Setting Up the KC705 Evaluation Board



IMPORTANT: This project is created for the KC705 Rev 1.0 Evaluation Board. The pin constraints are set based on this Evaluation Board. If you are using any other Revisions, update the XDC file `dut_fpga_kc705.xdc` with the correct pin constraints.

- Connect the Digilent Cable (or Platform USB Cable) for Programming.
- Connect the TI USB Interface Adapter to the PMBus port on the KC705 Evaluation Board.
- Connect the Power Cable.



X14505

Figure 49: KC705 Hardware Setup

4. Install the TI Fusion Digital Power Designer software on the PC from this location:

www.ti.com/tool/fusion_digital_power_designer

Step 3: Setting Up the Fusion Digital Power Designer Software

1. Power ON the KC705 Evaluation Board.
2. Open the Fusion Digital Power Designer.

The software detects the USB adapter and brings up the GUI.

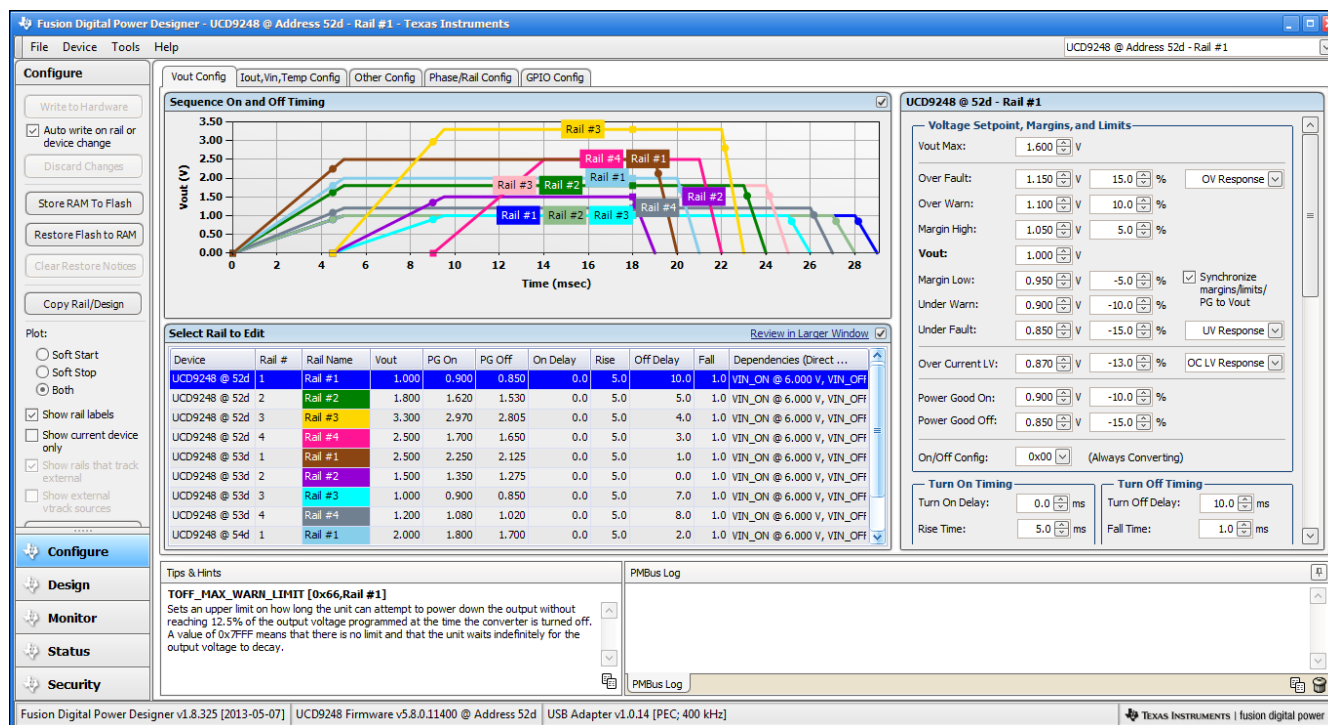


Figure 50: Fusion Digital Power Designer

Step 4: Programming the Bitstream

1. Power up the KC705 Evaluation Board.
2. In the Vivado Hardware Manager, click **Open Target** in the green alert bar and select **Open New Target**.

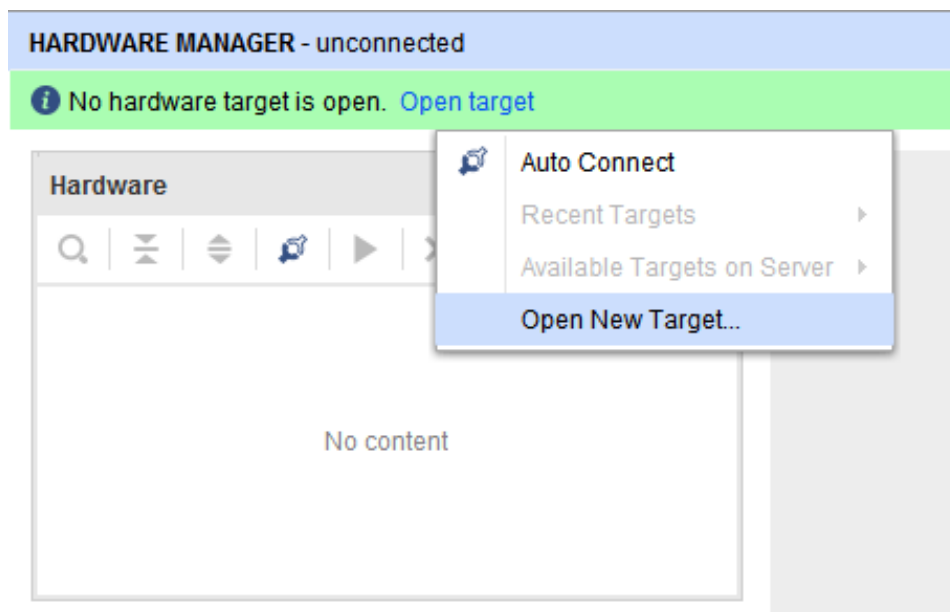


Figure 51: Opening the New Hardware Target

3. In the Open New Hardware Target dialog box, click **Next** to go to the **Hardware Server Settings**.
4. Select the server to which the board is connected.
 - If the board is connected to the local PC, select **Local Server** and click **Next**.
 - If you are connecting to a remote server, see Connecting to a Hardware Target Using `hw_server` in the *Vivado Design Suite User Guide: Programming and Debugging* (UG908).

When the hardware is detected successfully, the part information will be displayed in the Open New Hardware Target dialog box.

5. Verify the part information, then click **Next** and **Finish**.

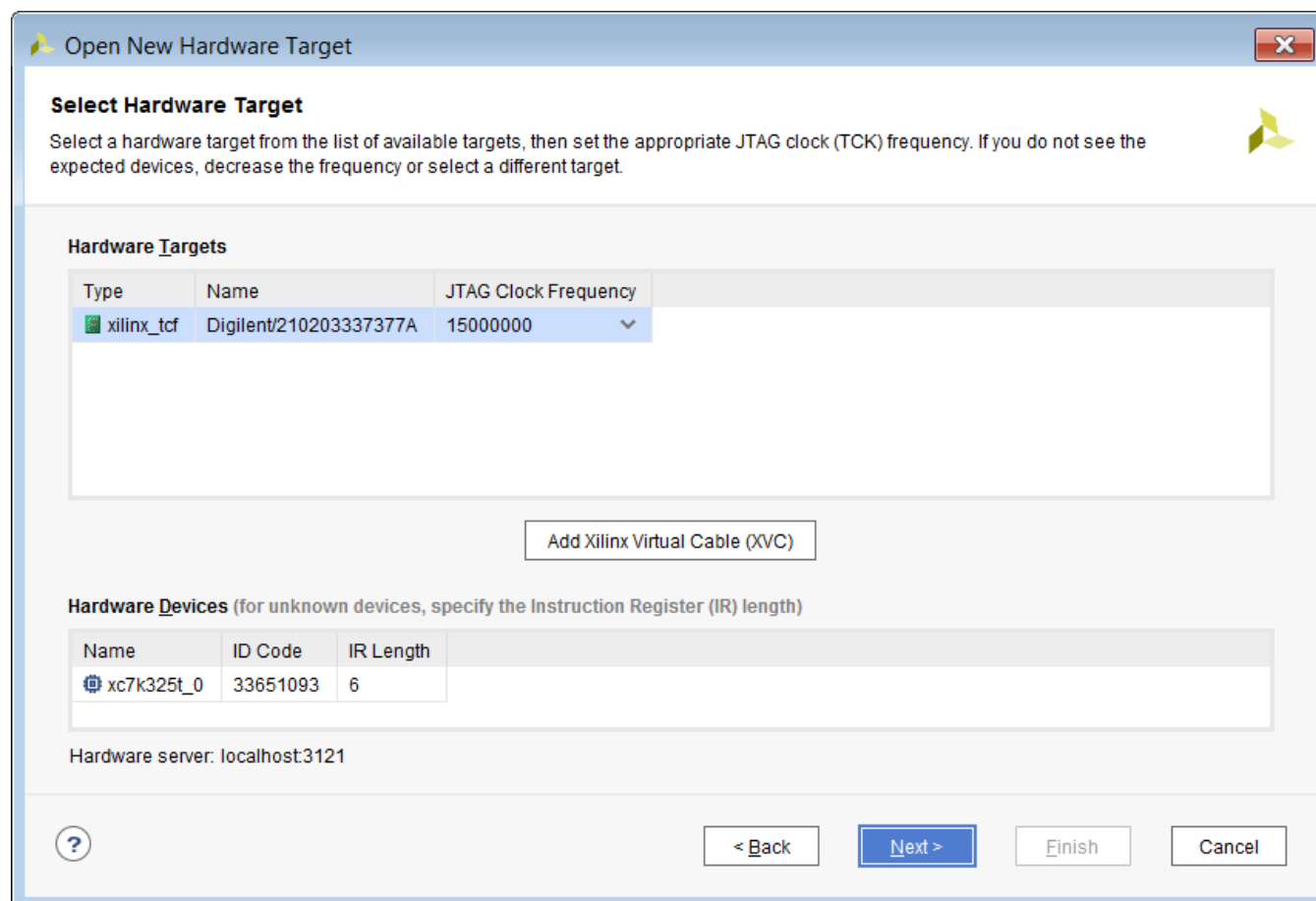


Figure 52: Hardware Target Detected

6. In the Hardware window, right-click the part and select **Program Device**.

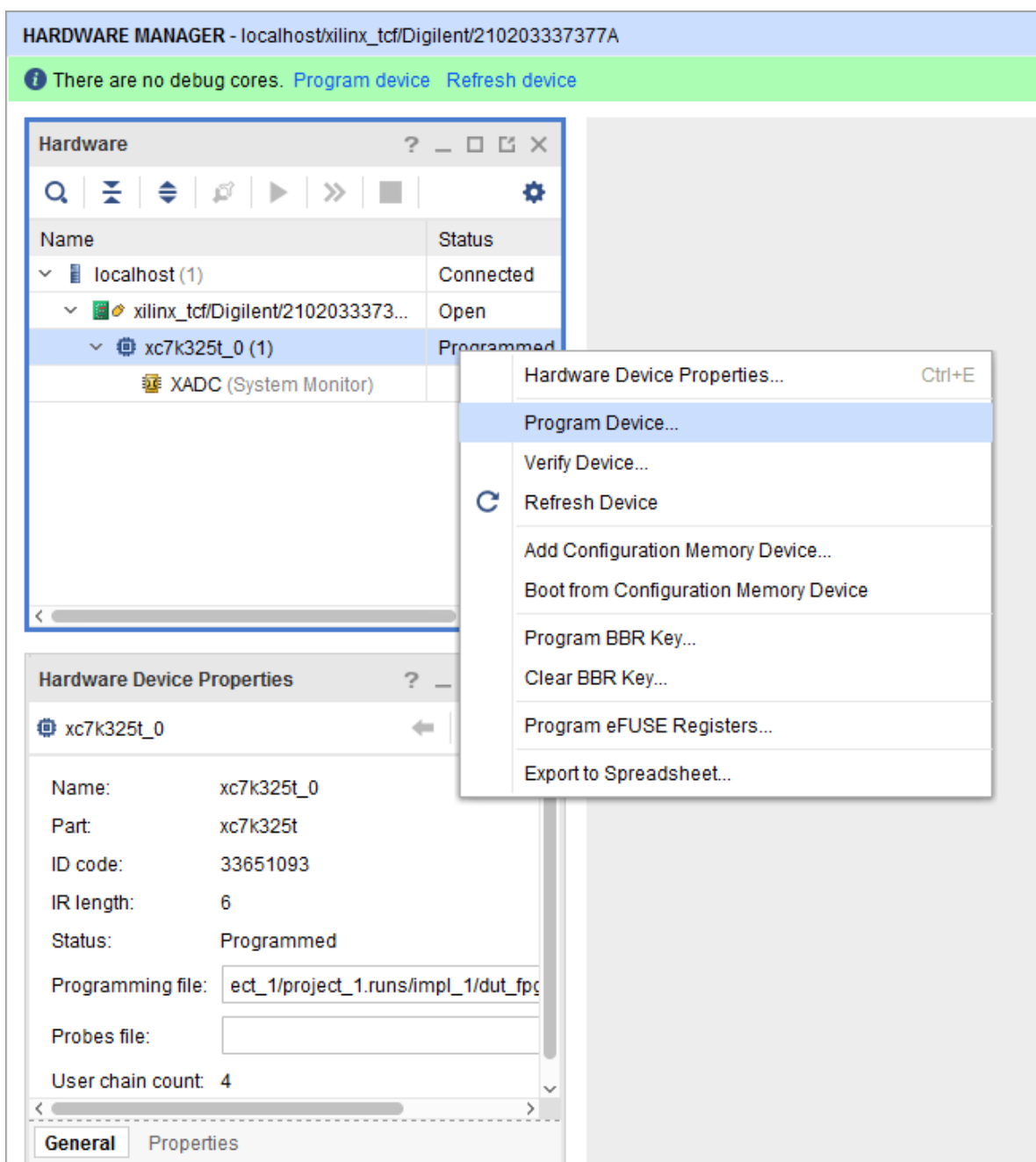


Figure 53: Program Device

7. Select the bit file `<project_dir>/power_tutorial1/power_tutorial1.runs/impl_1/dut_fpga.bit` and click **Program**.
8. After the program completes successfully, select **XADC (System Monitor)** in the Hardware window, then right-click and select **Dashboard**, and then select **New Dashboard**.
9. Click **OK**. The System Monitor window opens and plots Die Temperature (Junction Temperature) in the Graph Window.

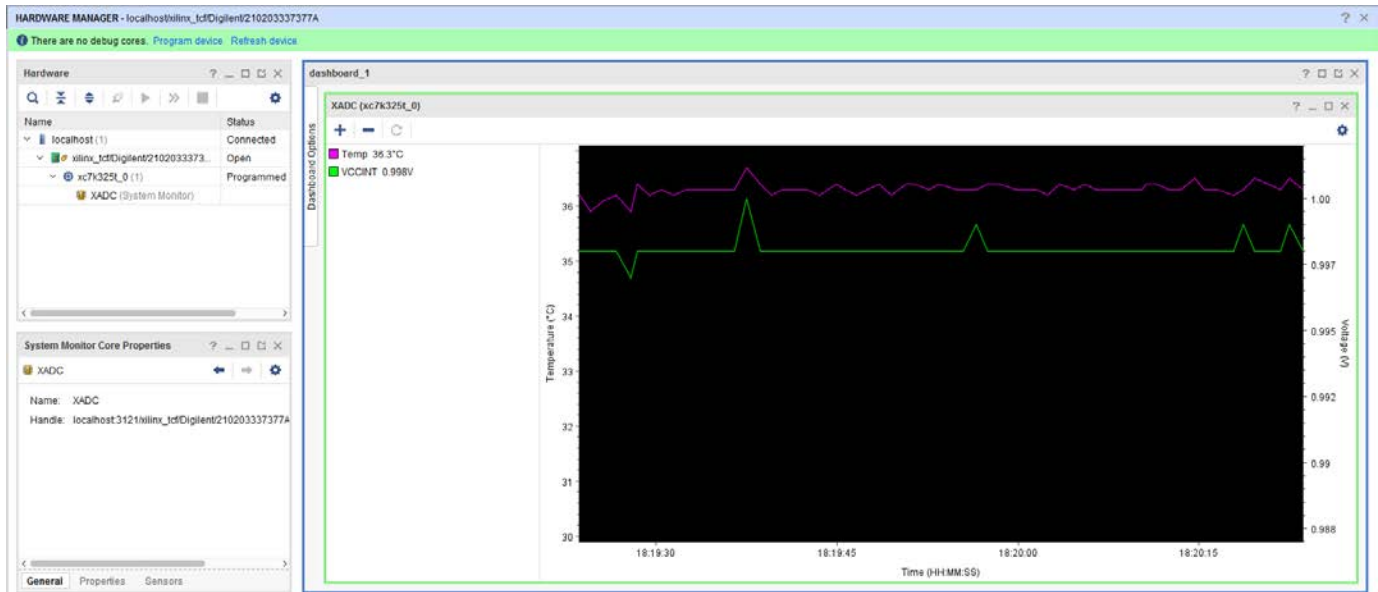


Figure 54: System Monitor – Junction Temperature Plot

Step 5: Measuring the Hardware Power Rails

1. In the Fusion Digital Power Designer, select a rail in the Configure view and click **Monitor**.

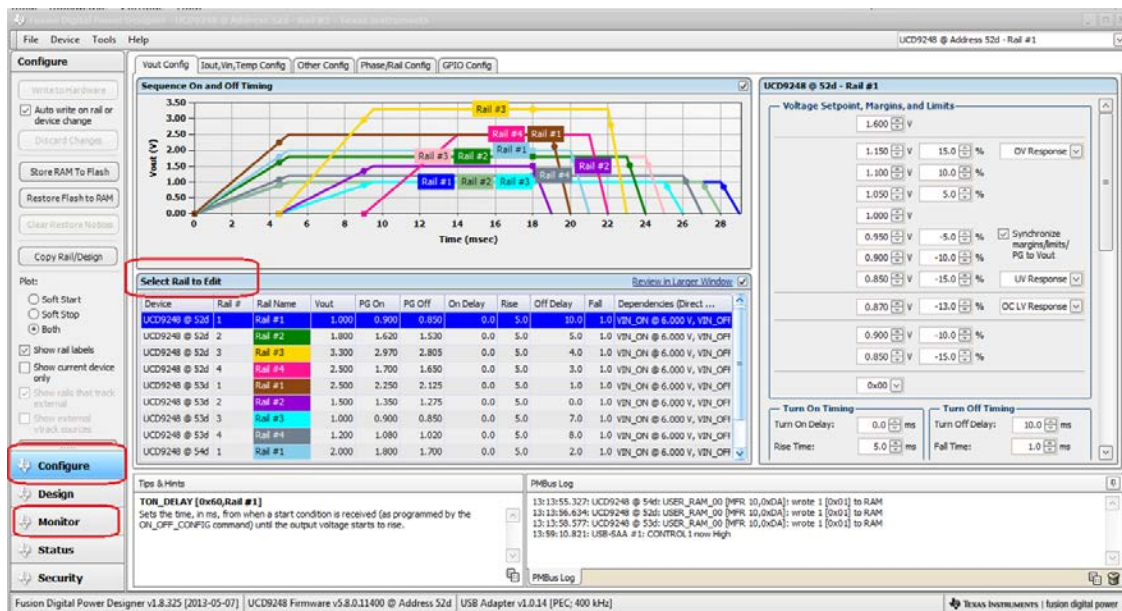


Figure 55: Select a Rail to Monitor

2. Configure the parameters to be monitored. An Output Power graph will be plotted in the Monitor window.

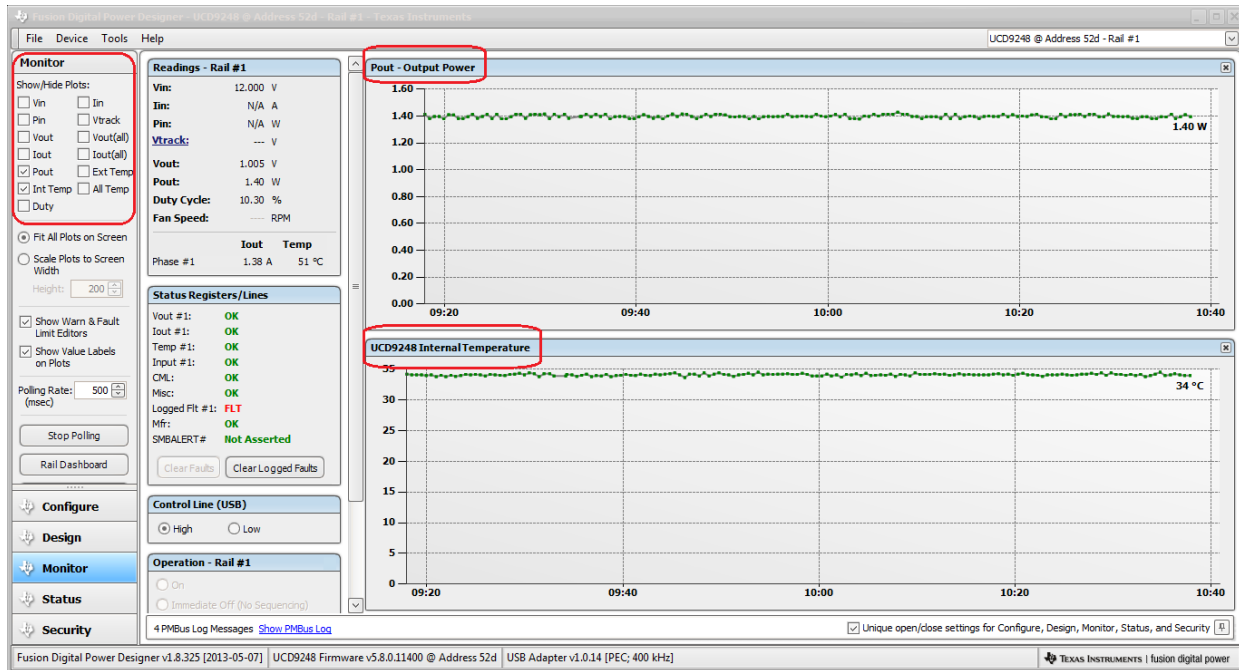


Figure 56: Monitor a Rail – Power and Junction Temperature

- Repeat the steps above to monitor the power information for each rail supplied to the device. Note that rail information is displayed in terms of Regulator address.

		KC705 (UCD9248)			
device	rail	purpose	voltage	IOUT_CAL_GAIN	IOUT_CAL_OFFSET
52	1	VCCINT	1.0V	0xEBDC	0x8000
	2	VCCAUX	1.8V	0xEBDC	0x8000
	3	VCC3V3	3.3V	0xEBDC	0x8000
	4	VADJ	2.5V	0xEBDC	0x8000
53	1	VCC2V5	2.5V	0xEBDC	0x8000
	2	VCC1V5	1.5V	0xEBDC	0x8000
	3	MGT_AVCC	1.0V	0xEBDC	0x8000
	4	MGT_AVTT	1.2V	0xEBDC	0x8000
54	1	VCCAUX_IO	1.8V	0xEBDC	0x8000
	2	VCCBRAM	1.0V	0xEBDC	0x8000
	3	MGT_VCCAUX	1.8V	0xEBDC	0x8000
	4	N/A	N/A	N/A	N/A

Figure 57: KC705 Rail Address Mapping

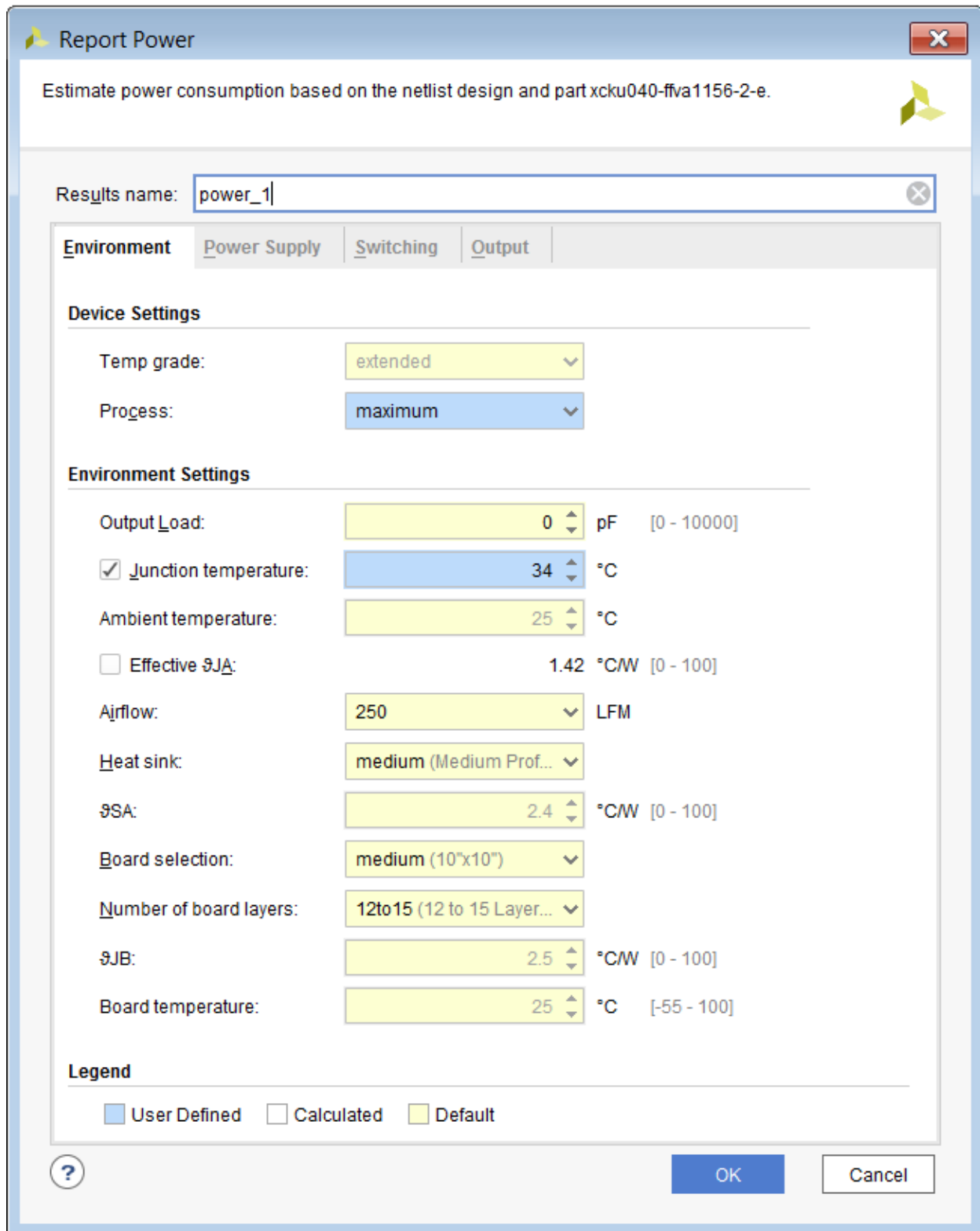
- Note the Junction Temperature value either from the Vivado Hardware Manager or from the Fusion Digital Power Designer.

Step 6: Estimating Vectorless Power with Junction Temperature

For further Power Analysis, you can use the measured Junction Temperature and other thermal settings to feed into Vivado Report Power for better accuracy.

1. In the Vivado Design Suite, open the tutorial project and click **Open Implemented Design** to display the implemented design.
2. In the Tcl Console, run the following command to reset any user defined or SAIF file defined settings.

```
reset_switching_activity -all
```
3. In the main menu bar, select **Reports > Report Power**.
4. In the Report Power dialog box **Environment** tab, enter the **Junction Temperature** value supplied by the hardware power measurement.
5. Set the **Process** to **maximum**.
6. In the **Switching** tab, make sure the **Simulation activity file (saif)** is blank.
7. Click **OK**.



Report Power

Estimate power consumption based on the netlist design and part xcku040-fvva1156-2-e.

Results name:

Environment | **Power Supply** | **Switching** | **Output**

Device Settings

Temp grade:

Process:

Environment Settings

Output Load: pF [0 - 10000]

☒ Junction temperature: °C

Ambient temperature: °C

☐ Effective θ_{JA} : 1.42 °C/W [0 - 100]

Airflow: LFM

Heat sink:

θ_{SA} : °C/W [0 - 100]

Board selection:

Number of board layers:

θ_{JB} : °C/W [0 - 100]

Board temperature: °C [-55 - 100]

Legend

☒ User Defined ☐ Calculated ☐ Default

Figure 58: Report Power Settings – Updating Junction Temperature

8. In the Power Report, observe that the power numbers increase slightly as compared to the vectorless power analysis using a default **Junction Temperature** value. Note that the **Junction Temperature** is now color coded as being user defined in the Power Report.

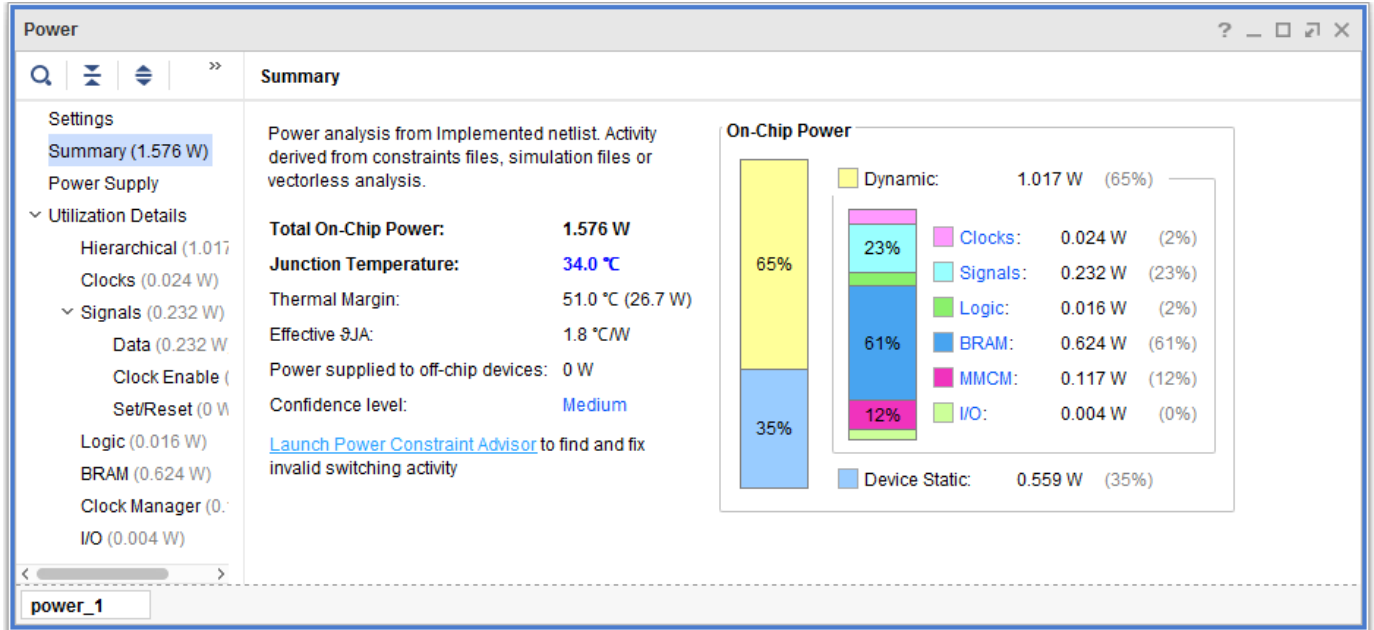


Figure 59: Power Report – Vectorless Analysis with Junction Temperature Supplied

9. Similarly, you can overwrite the Junction Temperature setting and do a SAIF based power analysis. Note the power numbers measured and estimated on non-power Optimized design.

Conclusion

In this lab, you have completed a Vivado Report Power analysis on post-synthesis and post-implementation netlist designs without Power Optimization. You also experimented with hardware power measurement using the KC705 Evaluation Board and with reading Junction Temperature for software analysis.

In lab 4, you will learn to experiment with hardware power measurement using the KCU105 Evaluation Board and with reading Junction Temperature for software analysis.

Lab 4: Measuring Hardware Power Using the KCU105 Evaluation Board

Introduction

In this lab, you will learn about the basic hardware power measurement technique and correlating the hardware power numbers with the numbers generated by Vivado® Report Power using the KCU105 evaluation board for UltraScale™ devices. The lab will take you through the steps for setting up the hardware measurement, programming a bit file using the Vivado Hardware Manager and power measurement through the Maxim Digital Power Tool. It also includes the Junction Temperature reading from the Vivado System Monitor.

Step 1: Generating a Bit File from the Implemented Design

1. In the Vivado Design Suite, open the UltraScale Implemented design.
2. In the Flow Navigator, click **Generate Bitstream**.
3. When prompted to **Save project before generating bitstream**, click **Don't Save**.

The Bitstream Generation Completed dialog box appears after the bitstream has been generated.

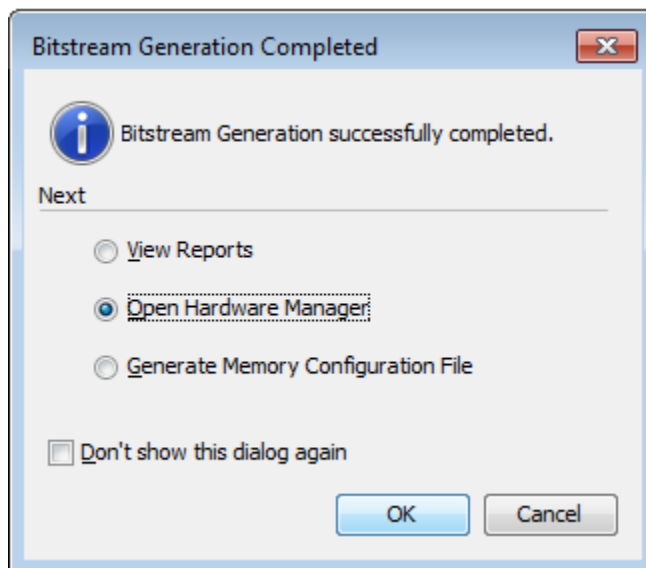


Figure 60: BitStream Generation Dialog Box

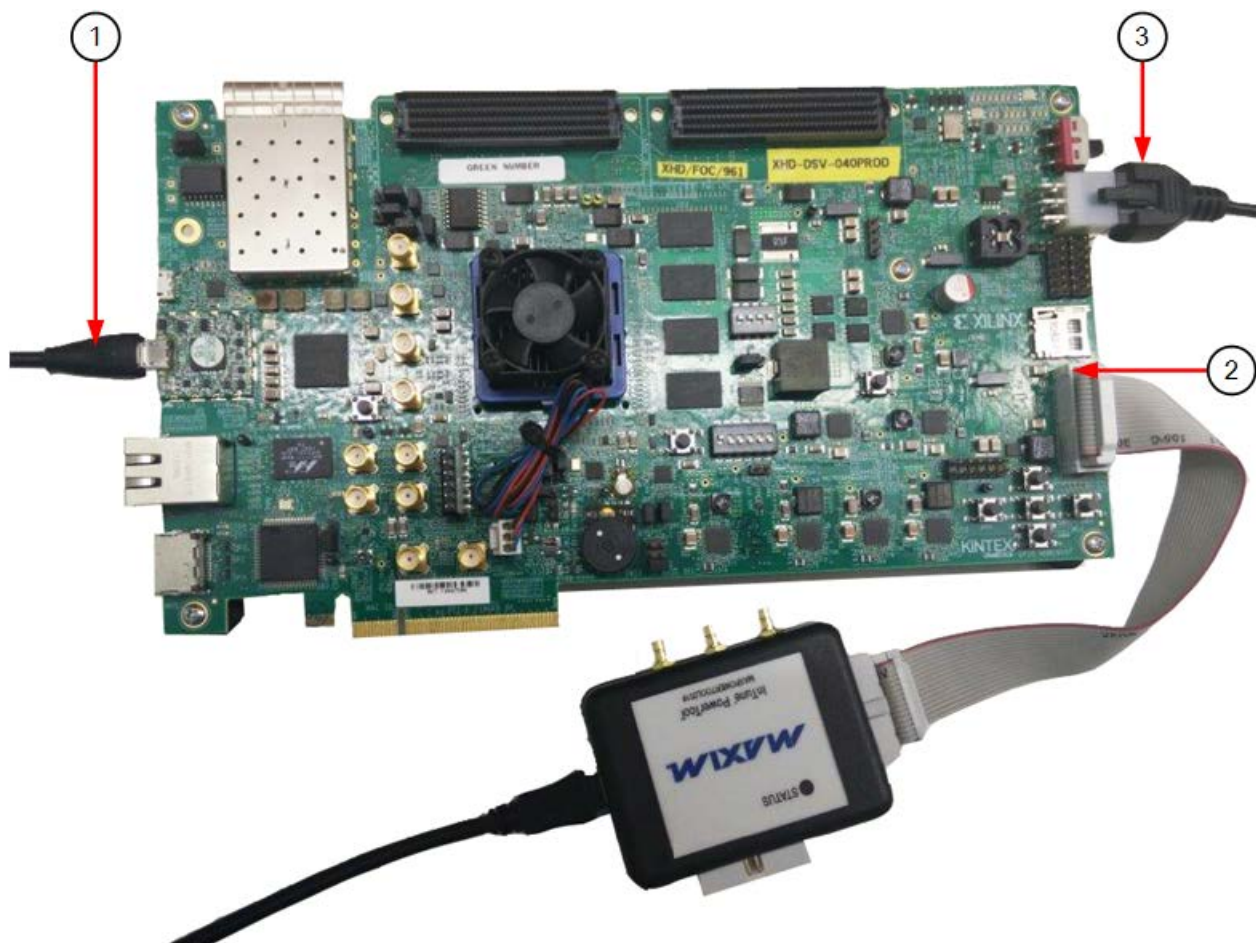
4. Select **Open Hardware Manager** in the **Bitstream Generation Dialog box** and click **OK** to open the Hardware Manager.

Step 2: Setting up the KCU105 Evaluation Board



IMPORTANT: This project is created for the KCU105 Rev B Evaluation Board. The pin constraints are set based on this Evaluation Board. If you are using any other Revisions, update the XDC file `dut_fpga_kcu105.xdc` with the correct pin constraints.

1. Connect the Digilent Cable (or Platform USB Cable) for Programming.
2. Connect the MAXPOWERTOOL002# Interface Adapter to the PMBus port on the KCU105 Evaluation Board.
3. Connect the Power Cable.



X16390-032416

Figure 61: KCU105 Hardware Setup

4. Install the Maxim Digital Power Designer software on the PC from this location.

<https://www.maximintegrated.com/en/design/tools/applications/evkit-software/index.mvp?id=1183>

Step 3: Configuring the Maxim Digital Power Tool Software

1. Power ON the KCU105 Evaluation board.
2. Open the Maxim Digital Power Tool. The software detects the Interface adapter and brings up the GUI.

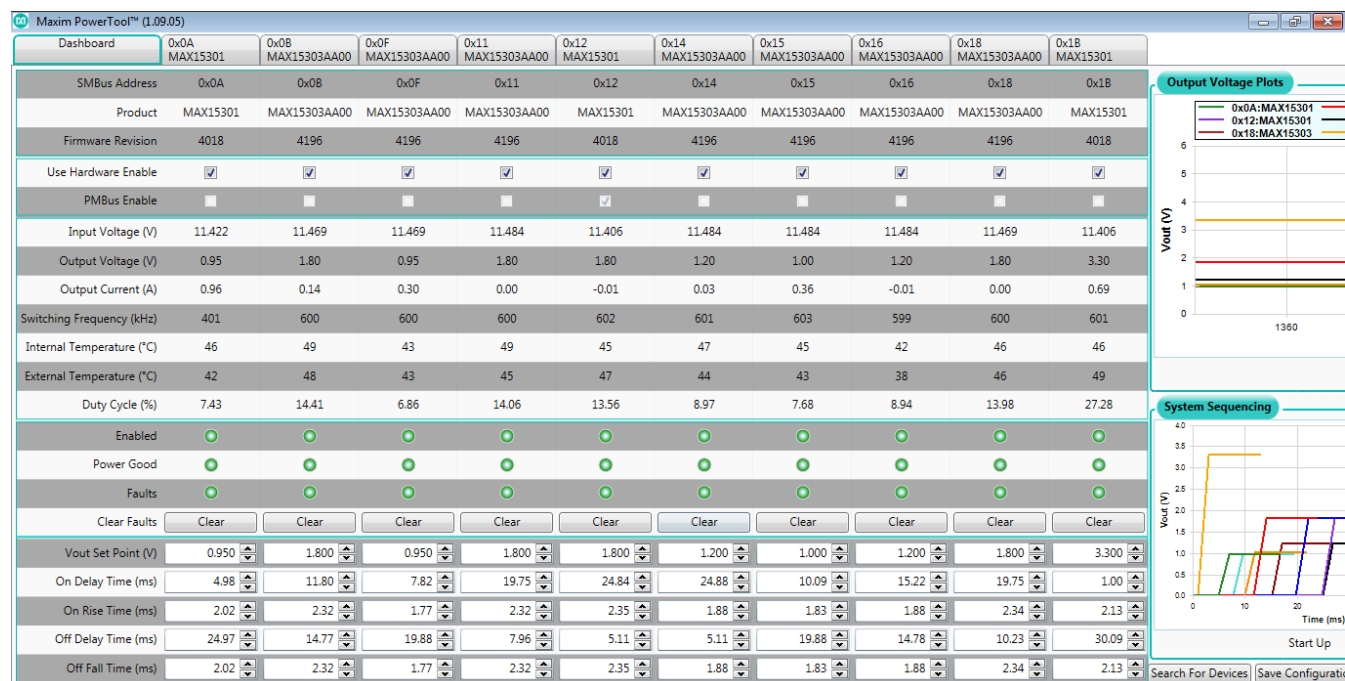


Figure 62: Maxim Digital Power Tool

Step 4: Programming the Bitstream

1. In the Vivado Hardware Manager, click **Open Target** in the green alert bar and select **Open New Target**.
2. In the Open New Hardware Target dialog box, click **Next** to go to the **Hardware Server Settings**.
3. Select the server to which the board is connected.
 - o If the board is connected to the local PC, select **Local Server** and click **Next**.
 - o If you are connecting to a remote server, see Connecting to a Hardware Target Using hw_server in the *Vivado Design Suite User Guide: Programming and Debugging* (UG908).

When the hardware is detected successfully, the part information will be displayed in the Open New Hardware Target dialog box.

4. Verify the part information, then click **Next** and **Finish**.

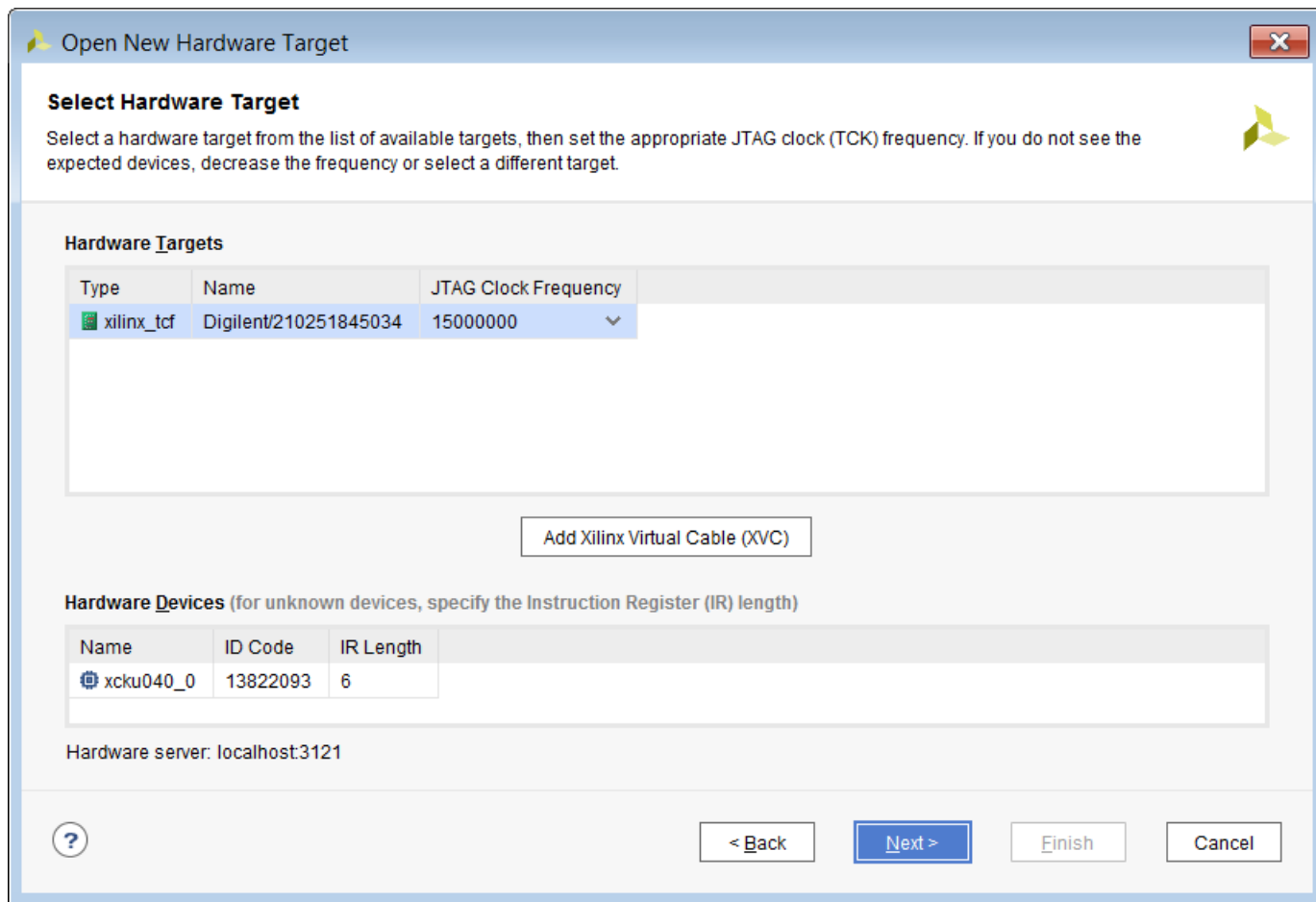


Figure 63: Detecting the Hardware Target

5. In the Hardware Devices window, right-click the part and select **Program Device**.

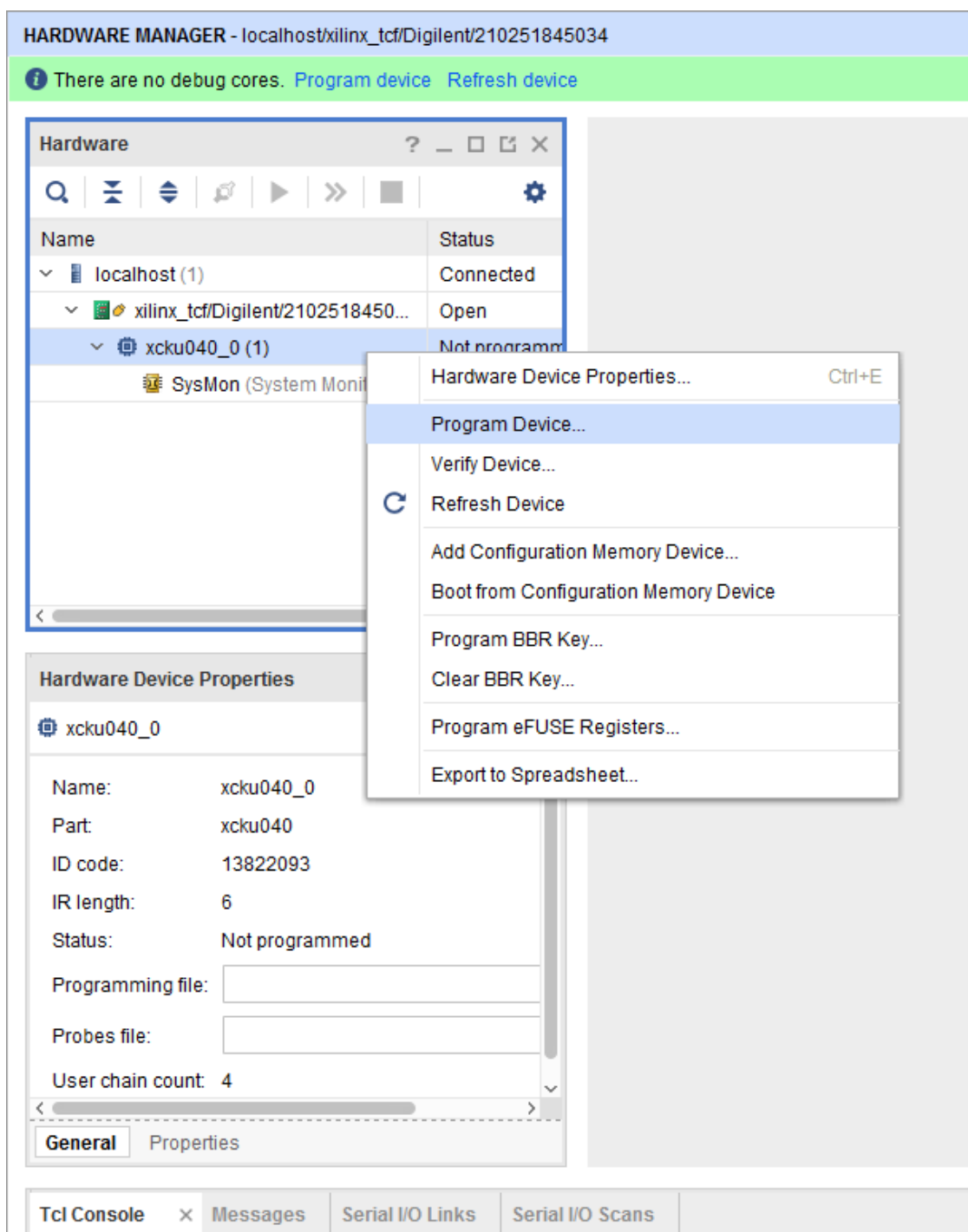


Figure 64: Program Device setup

6. Select the bit file from the implementation runs directory of the project which is created in [Lab 2](#) for UltraScale design <project_dir>/power_tutorial2/power_tutorial2.runs/impl_1/ dut_fpga.bit and click **Program**.
7. After the program completes successfully, select **XADC (System Monitor)** in the Hardware window, then right-click and select **Dashboard**, and then select **New Dashboard**.

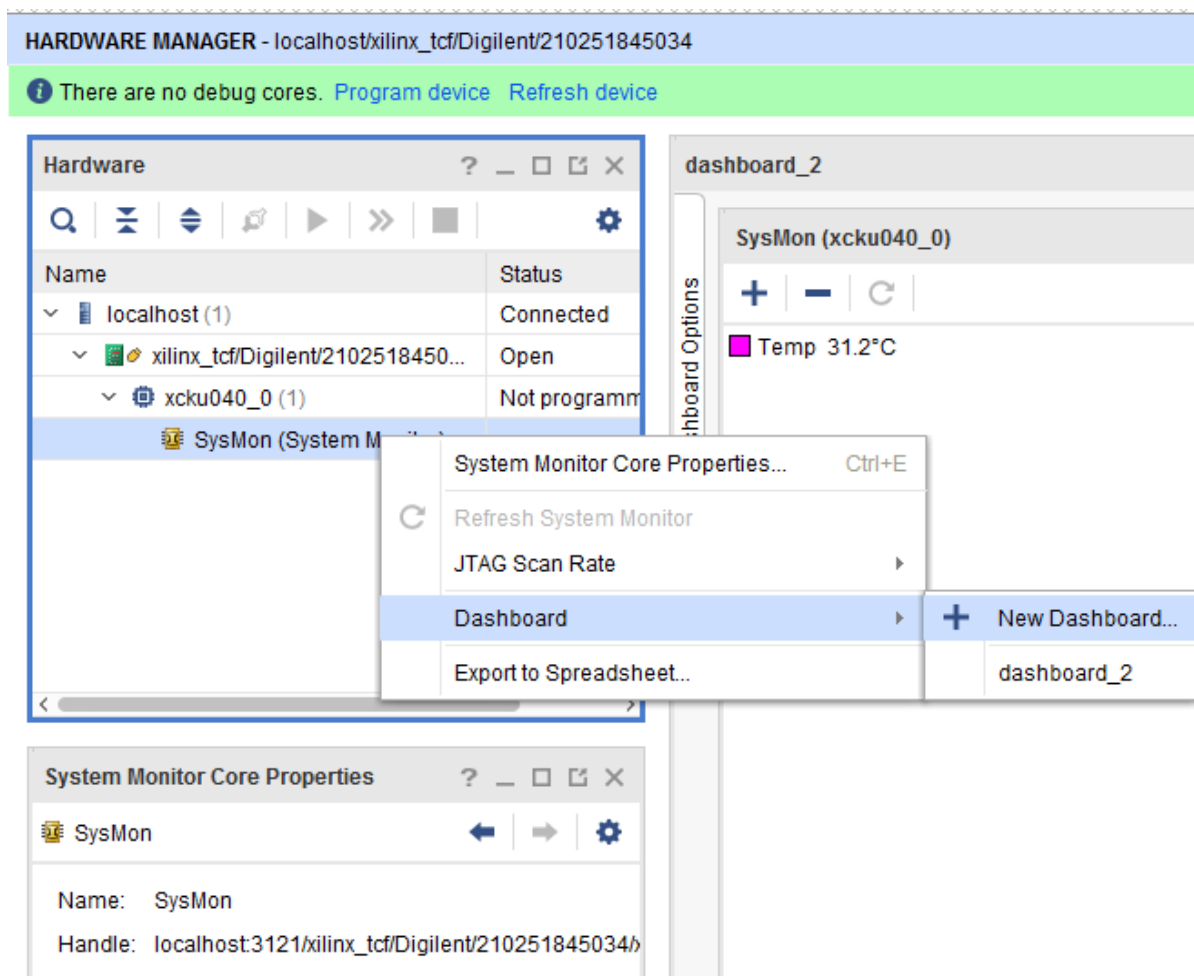


Figure 65: Creating a New Dashboard

- Click **OK**. The System Monitor window opens and plots Die Temperature (Junction Temperature) in the Graph Window.

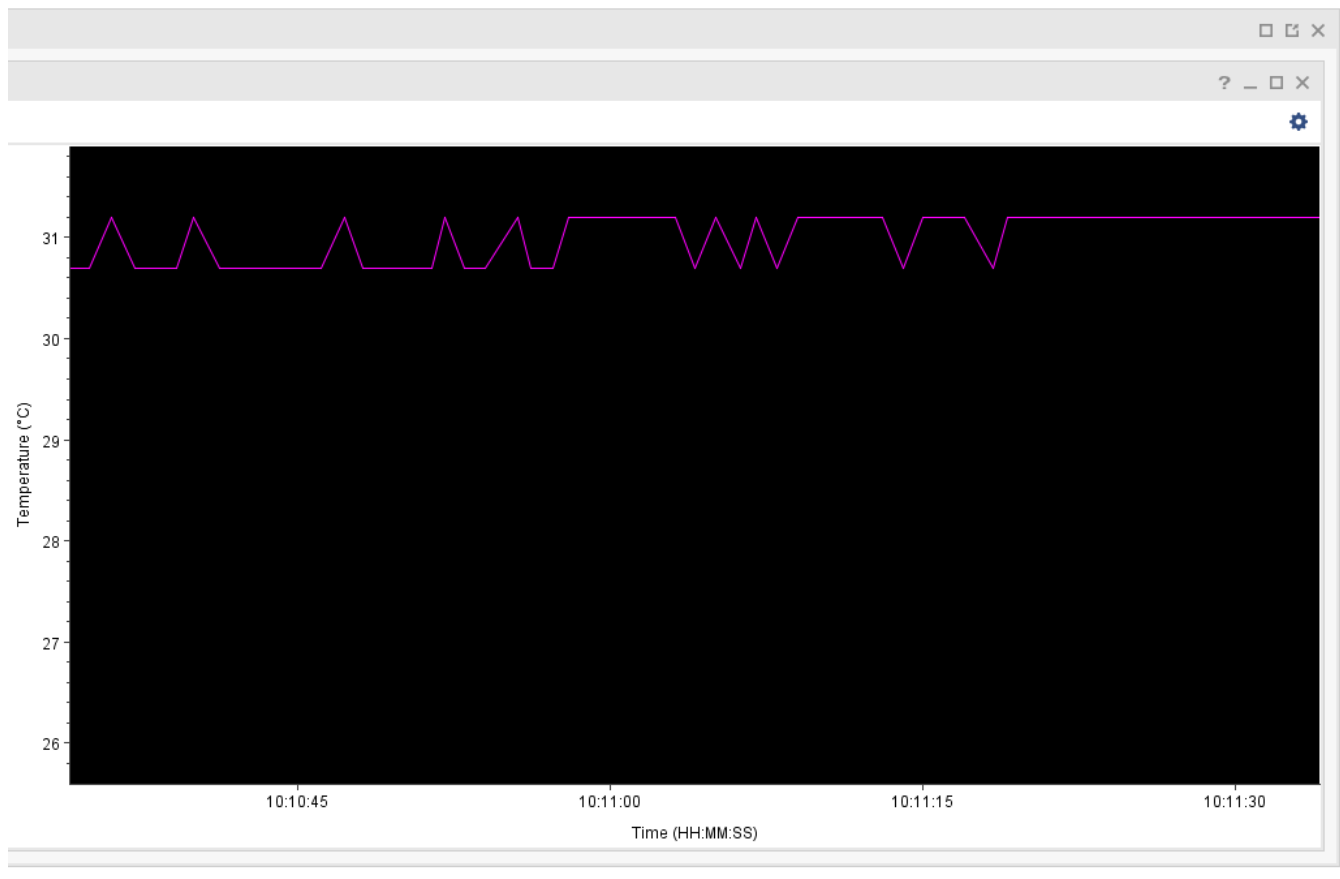


Figure 66: System Monitor- Junction Temperature Plot for UltraScale Device

Step 5: Measuring the Hardware Power Rails

1. In the Maxim Digital Power GUI, select a rail to monitor the power information.

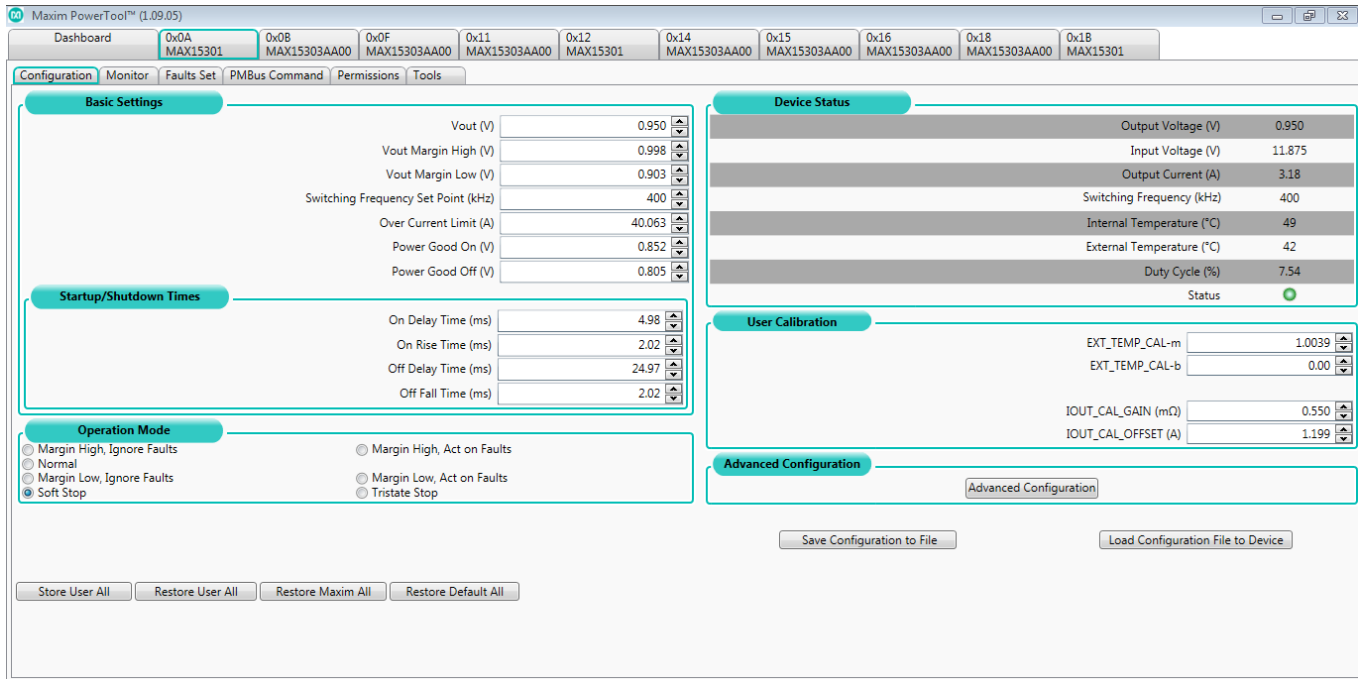


Figure 67: Selecting a Rail to Monitor

2. In **Configuration** tab, you can observe the basic settings and device status.
3. Click **Monitor** tab to observe the voltage and current plots.

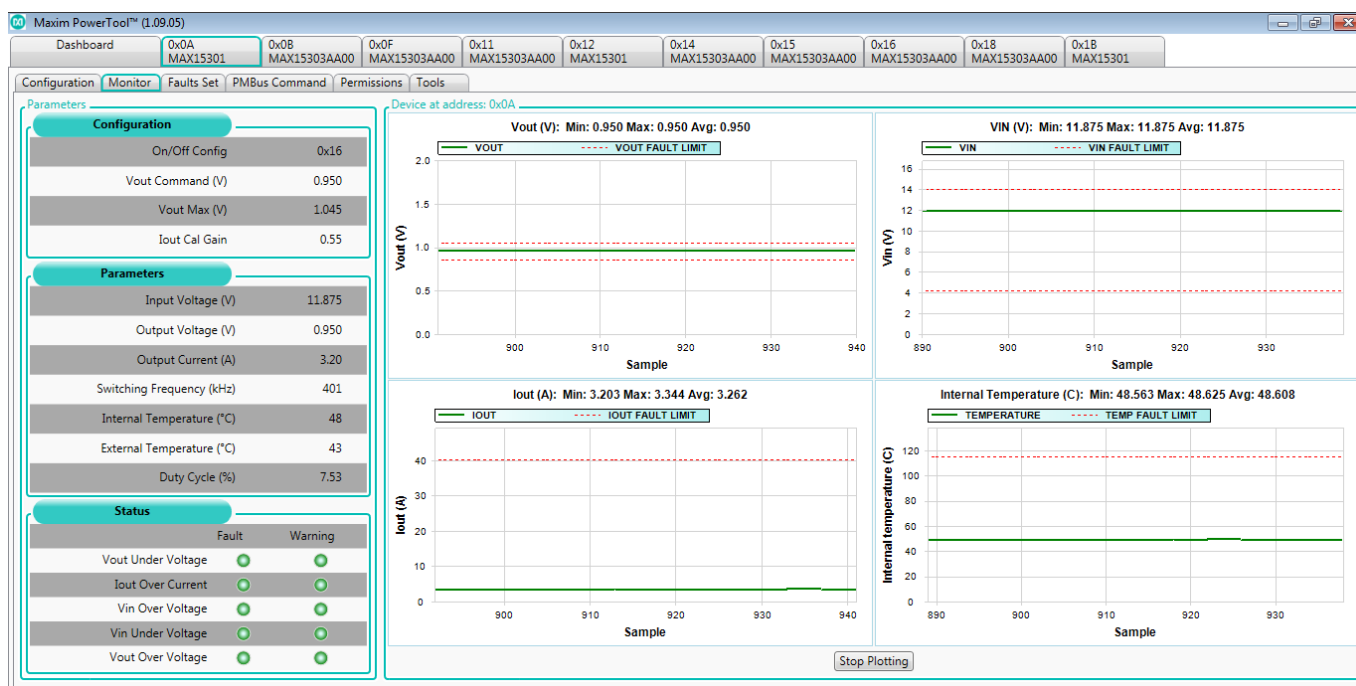


Figure 68: Monitoring a Rail - Voltage and Current

- Repeat the steps mentioned above to monitor the power information for each rail supplied to the device.

Note that the rail information is displayed in terms of Regulator address.

RAIL	VOLTAGE	PMBUS ADDR
VCCINT	0.95V	0x0A
VCCAUX	1.8V	0x0B
VCCBRAM	0.95V	0x0F
VCC1V8	1.8V	0x11
VADJ_1V8	1.8V	0x12
VCC1V2	1.2V	0x14
MGTAVCC	1V	0x15
MGTAVTT	1.2V	0x16
MGTAVCCAUX	1.8V	0x18
UTIL_3V3	3.3V	0x1B

- Note the Junction Temperature value from the Vivado Hardware Manager (System Monitor).

Step 6: Estimating the Vectorless Power with Junction Temperature

For further Power Analysis, you can use the measured Junction Temperature and other thermal settings to feed into Vivado Report Power for better accuracy.

6. In the Vivado Design Suite, open the tutorial project and click **Open Implemented Design** to display the implemented design.
7. In the Tcl Console, run the following command to reset any user defined or SAIF file defined settings.

```
reset_switching_activity -all
```

8. In the main menu bar, select **Reports > Report Power**.
9. In the **Environment** tab of Report Power dialog box, enter the **Junction Temperature** value supplied by the hardware power measurement.
10. Set the **Process** to **maximum**.
11. In the **Switching** tab, make sure that the **Simulation activity file (saif)** is blank.
12. Click **OK**.

Report Power

Estimate power consumption based on the netlist design and part xcku040-ffva1156-2-e.

Results name:

Environment | Power Supply | Switching | Output

Device Settings

Temp grade:

Process:

Environment Settings

Output Load: pF [0 - 10000]

☒ Junction temperature: °C

Ambient temperature: °C

☐ Effective θ_{JA} : °C/W [0 - 100]

Airflow: LFM

Heat sink:

θ_{SA} : °C/W [0 - 100]

Board selection:

Number of board layers:

θ_{JB} : °C/W [0 - 100]

Board temperature: °C [-55 - 100]

Legend

☒ User Defined ☐ Calculated ☐ Default

Figure 69: Reporting Power Settings – Updating Junction Temperature

13. In the Power Report, observe that the power numbers increase slightly as compared to the vectorless power analysis using a default **Junction Temperature** value.

Note that the **Junction Temperature** is now color coded as being user defined in the Power Report.

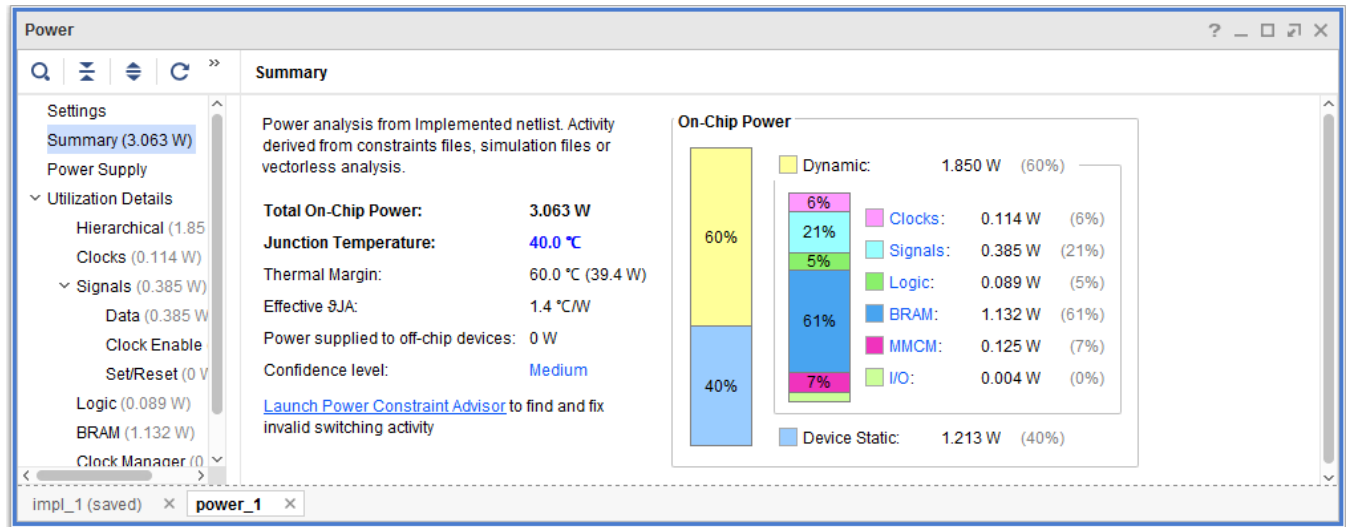


Figure 70: Power Report – Vector Analysis with Junction Temperature Supplied

14. Similarly, you can overwrite the Junction Temperature setting and do a SAIF based power analysis.

Conclusion

In this lab, you have learned to experiment with hardware power measurement using the KCU105 Evaluation Board.

In lab 5, you will learn about using the Power Optimization features in the Vivado IDE.

Lab 5: Performing Power Optimization

Introduction

In this lab, you will learn about using the Power Optimization features in Vivado® for 7 series devices. The lab will take you through the steps for invoking Power Optimization after synthesizing the design. It will also guide you on how to use the power optimization report, make decisions and selectively turn off power optimization on signals, blocks, and hierarchies.



TIP: When you run Implementation on your design, the Vivado tools may perform BRAM power optimizations by default during `opt_design`. These optimizations will not affect performance, and will have little impact on area and runtime. In the previous Lab, the default BRAM power optimization was disabled ([Step-9 of Lab1](#)) by setting a `NoBramPowerOpt` directive to `opt_design`.

Step 1: Setting Up Options to Run Power Optimization

1. In the Flow Navigator, right click **Implementation** and select **Implementation Settings**.
2. In the Project Settings dialog box, select **Implementation** tab to make the following settings:

- In the **Opt Design** settings, set the `-directive` option to **Default**.

BRAM optimization runs in the Default setting for **Opt Design** during Implementation. BRAM optimization was disabled in the previous lab. It is now re-enabled when the design runs Power Optimization.

- In the **Power Opt Design** settings, check the **is_enabled** box.

This ensures Power Optimization runs after `opt_design`. Enabling the **Power Opt Design** option prior to `place_design` results in a complete power optimization to be performed. This option yields the best possible power saving from the Vivado tools.

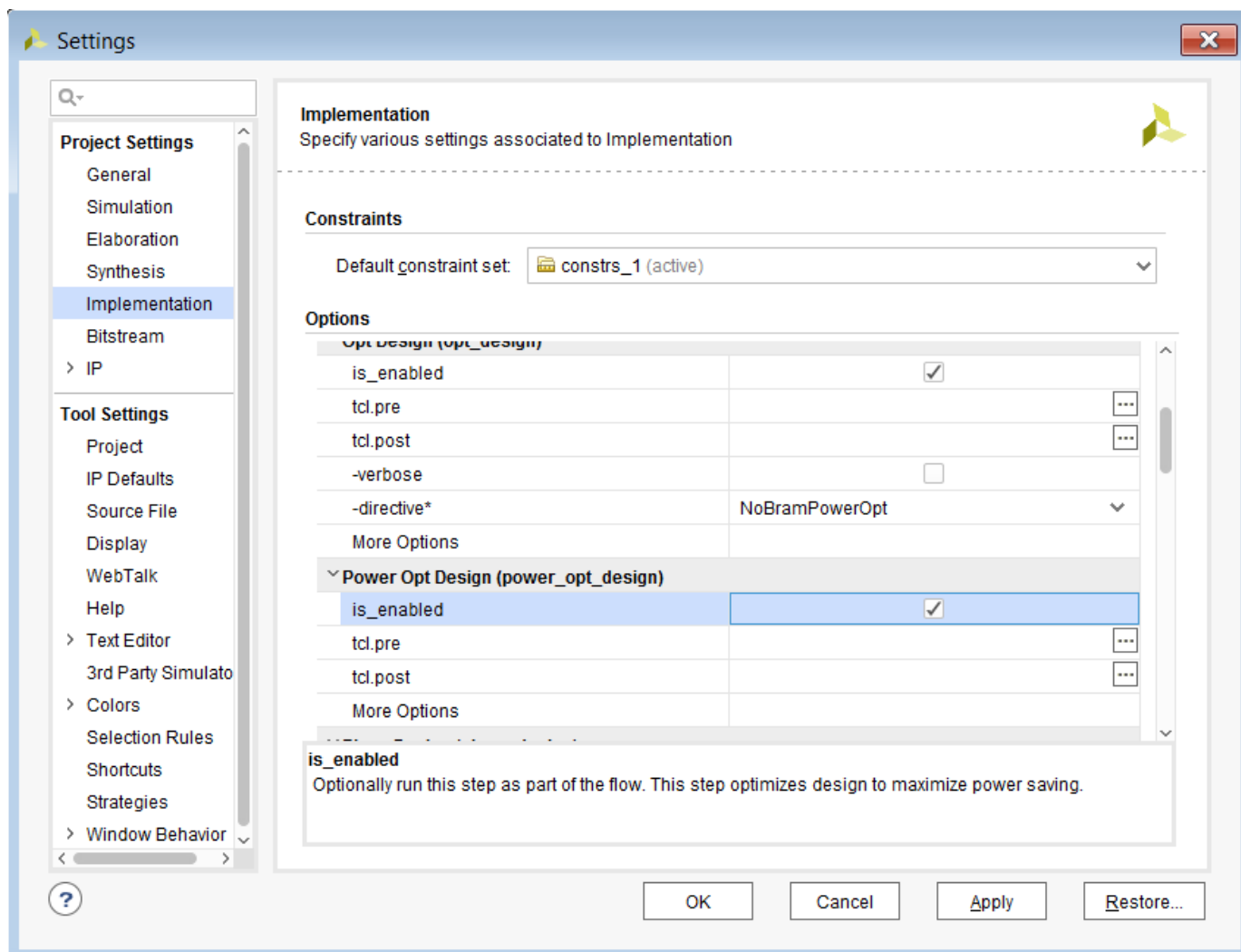


Figure 71: Implementation Settings – Enabling Power Optimization

3. Click **OK**.

4. In the Create New Run dialog box, click **Yes** to “Properties for the completed run ‘impl_1’ have been modified. Do you want to preserve the state of ‘impl_1’ and apply these changes to a new run?”.

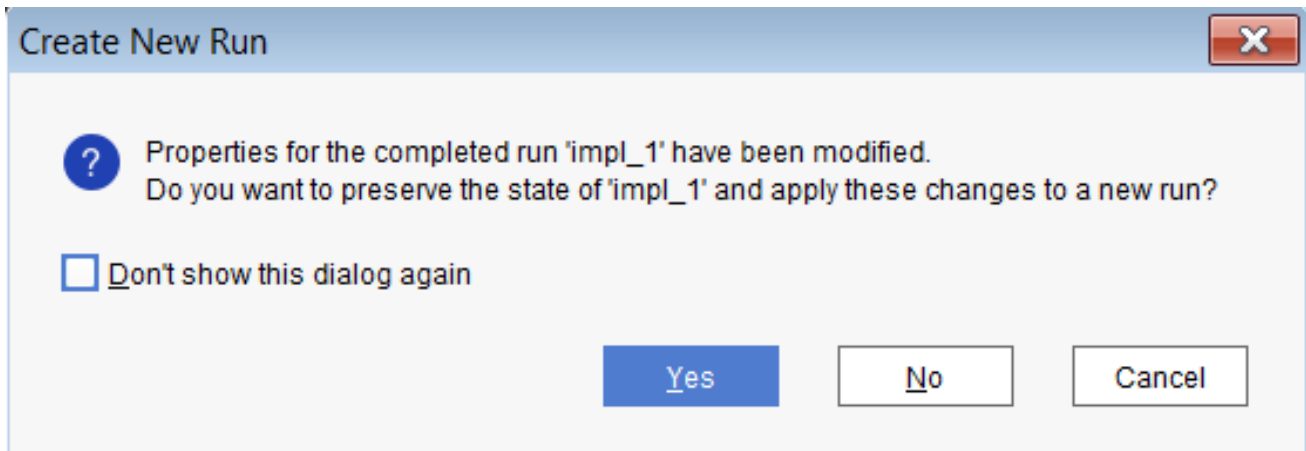


Figure 72: Create New Run Dialog Box

5. In the Create Run dialog box, set the **Run Name** to impl_2.
6. Click **OK**.
7. In the Flow Navigator, select **Run Implementation**. Click **Don't Save** when the Save Project window pops up to save both Synthesis and Implementation constraints.

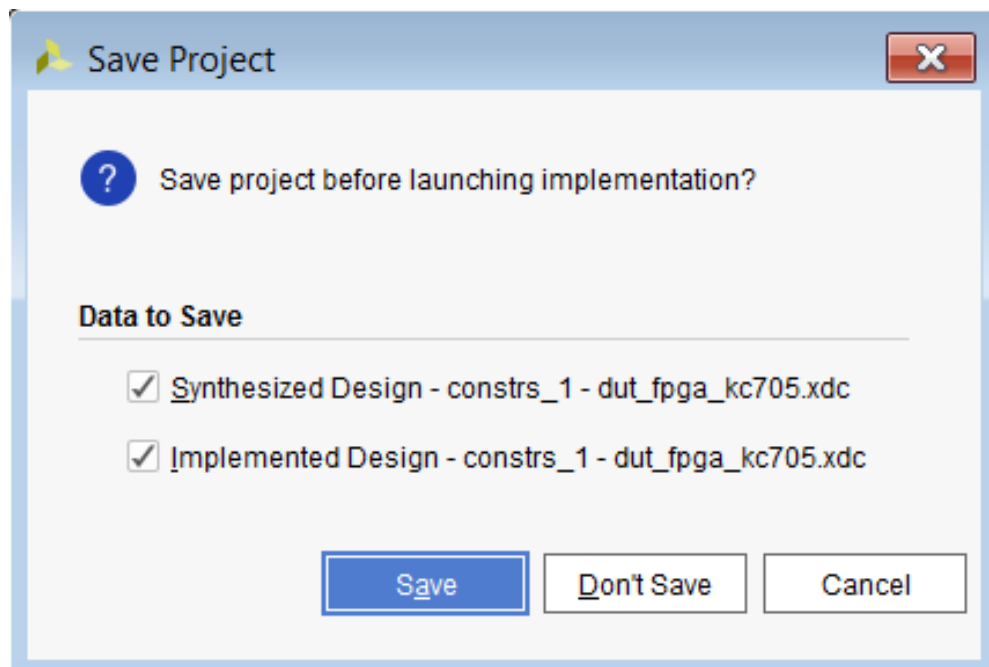


Figure 73: Click “Don’t Save” Here

You are running Implementation with Power Optimization turned on.

8. In the Implementation Completed dialog box, select **Open Implemented Design** and click **OK**. Click **Don't Save** when the Save Project window pops up to save both Synthesis and Implementation constraints.

Step 2: Running report_power_opt to Examine User/Design Specific Power Optimizations

1. In the Flow Navigator, select **Implemented Design**.
2. In the main menu bar, select **Reports > Report Power Optimization**.

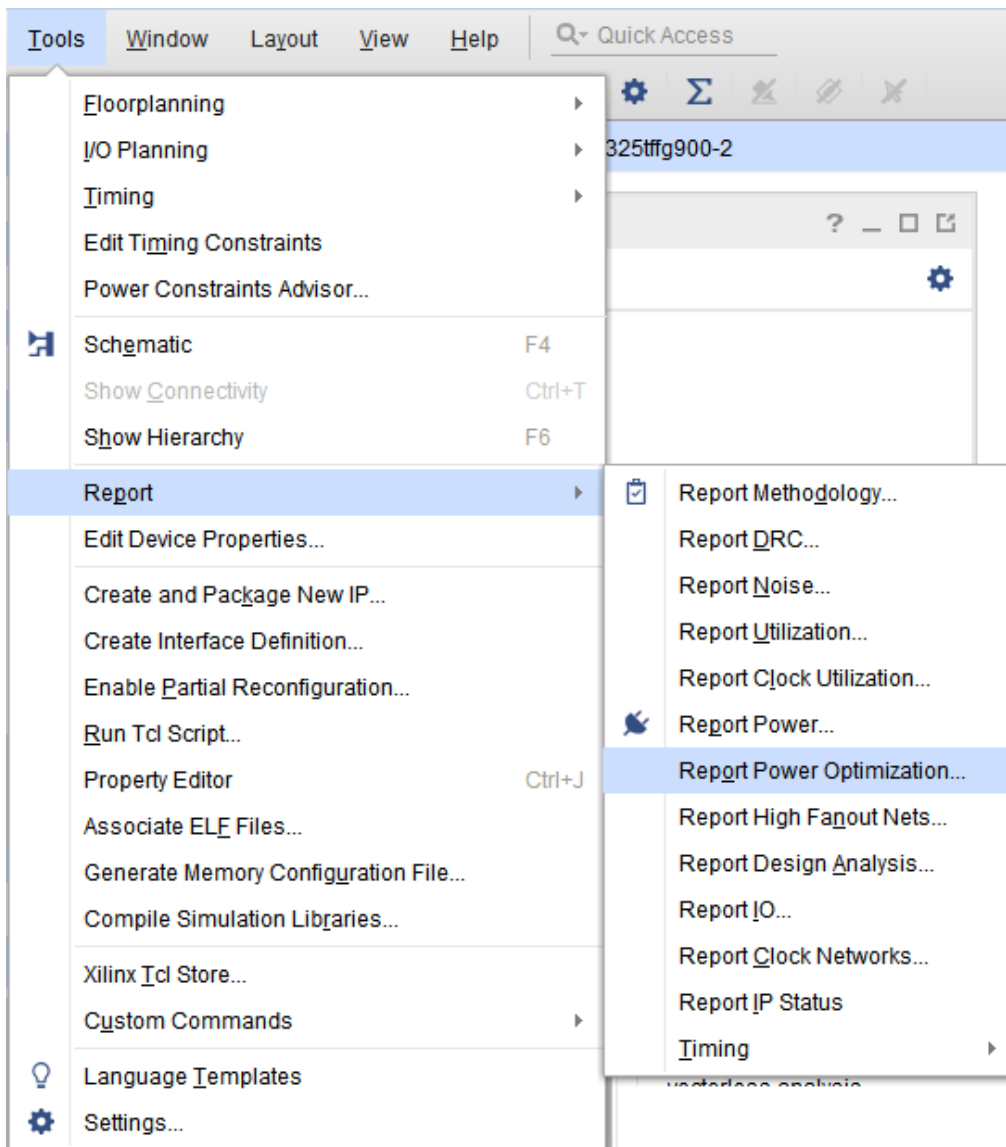


Figure 74: Generating a Power Optimization Report

- The Report Power Optimization dialog box appears, as shown in the following figure.

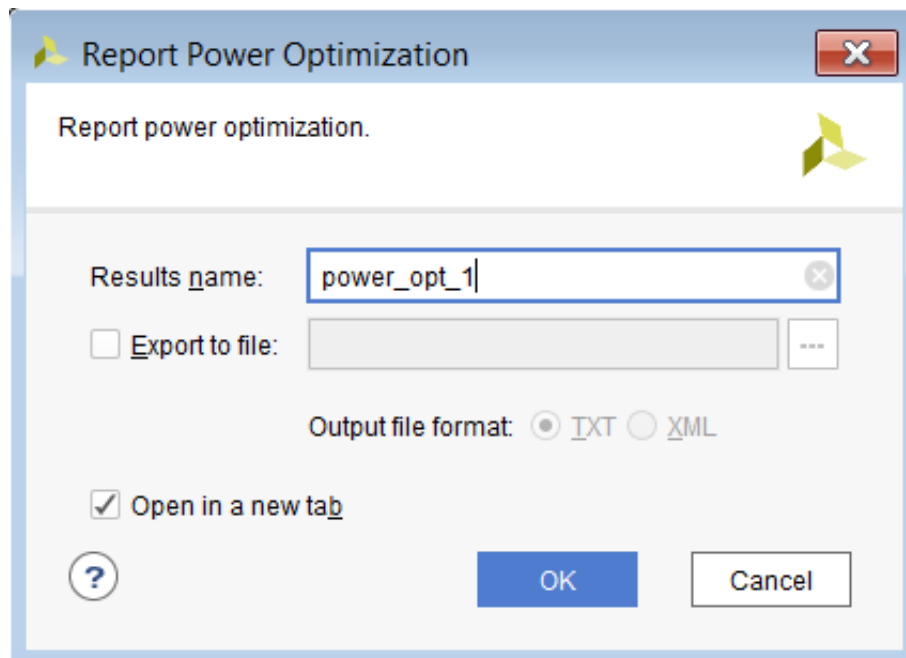


Figure 75: Report Power Optimization Dialog Box

- Enter `power_opt_1` for the **Results name**.
- Ensure that the **Open in a new tab** option is checked.
- Click **OK**. Alternatively, in the Tcl Console execute this Tcl command:

```
report_power_opt -name power_opt_1
```
- Observe the report **power_opt_1** is generated in the **Power Opt** tab of the results windows area. When the report opens, the **Summary** view is displayed in the report.
- In the **Summary** view, note that 50% of the BRAMs are clock gated by the tool during Power Optimization.

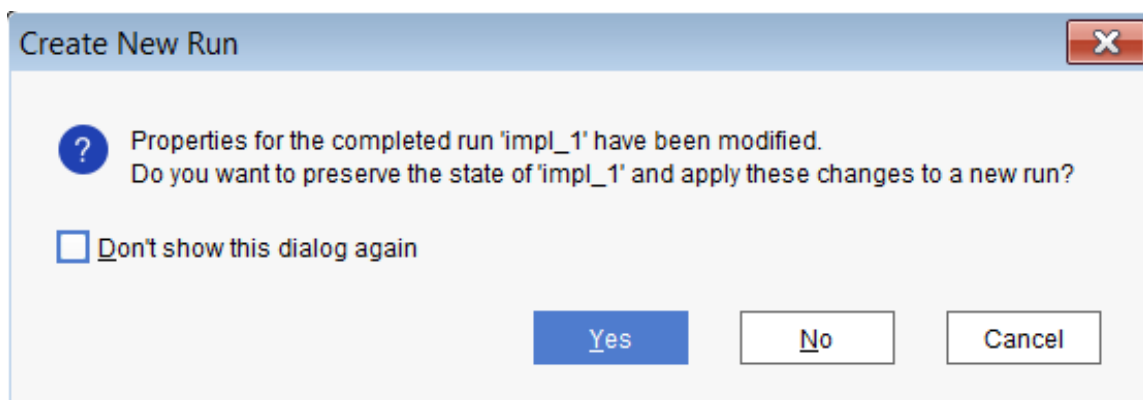


Figure 76: Power Optimization Report – Tool Gated BRAMs

9. In the Power Optimization Report, select **Hierarchical Information > BRAMs > Tool Gated BRAMs** and observe the BRAM cells and its CE ports which are gated by the tool during the power optimization.

Step 3: Running report_power to Examine Power Savings

1. In the main menu bar, select **Reports > Report Power**.
2. In the Report Power dialog box, make the following settings:
 - o Specify the **Results name** as power_1
 - o In the **Environment** tab, make sure the **Process** is set to **maximum**.
3. Click **OK**. Alternatively, in the Tcl Console execute this Tcl command:


```
report_power -name power_1
```
4. In the **Summary** view of the Power Report, observe an approximately 100-200mW power savings compared to the non-optimized power run in the previous lab.

You can generate a bitstream to program the hardware and measure its power, to observe the power saving in hardware. See [Lab 3: Measuring Hardware Power Using the KC705 Evaluation](#) for hardware power measurement instructions.

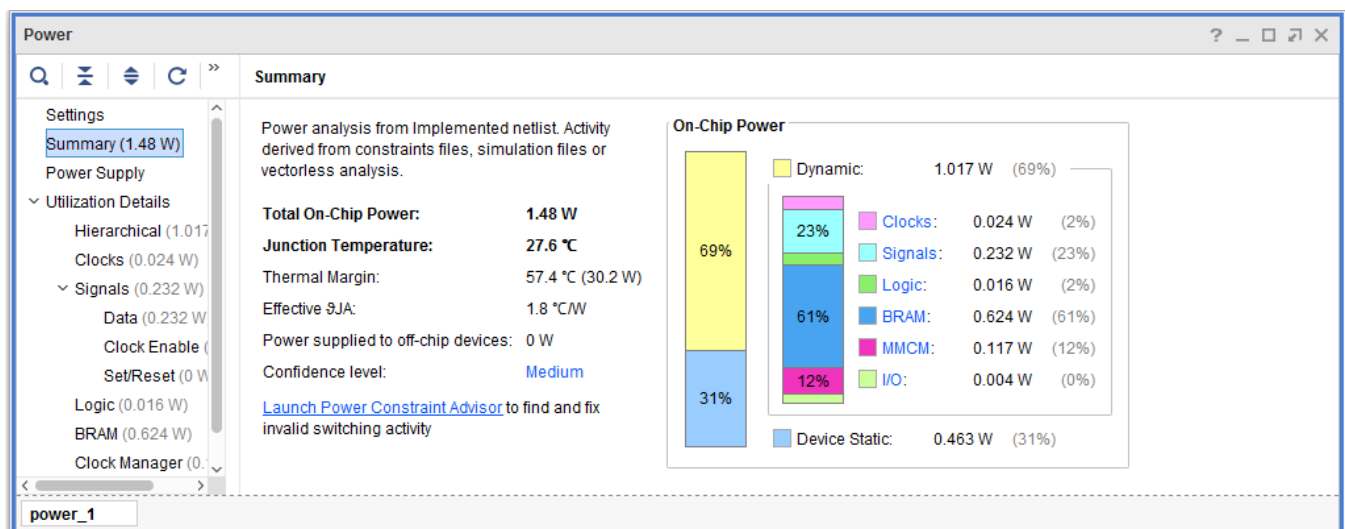


Figure 77: Power Savings with Power Optimization

Step 4: Turning Off Optimizations on Specific Signals and Rerunning the Implementation

In this step you will learn how to turn off the power optimization on specific BRAMs.



IMPORTANT: Power optimization works to minimize the impact on timing while maximizing power savings. However, in certain cases, if timing degrades after power optimization, you can identify and apply power optimizations only on non-timing critical clock domains or modules using the `set_power_opt` XDC command.

See the Vivado Design Suite User Guide: Power Analysis and Optimization ([UG907](#)) for more information on the `set_power_opt` command.

Assume that this BRAM is in the critical path:

```
dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_0
```

This step makes sure the tool does not gate this BRAM.

1. In the Tcl Console, type this command:

```
set_power_opt -exclude_cells [get_cells dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_0]
```

This will prevent the tool from gating this BRAM.

2. From the Flow Navigator choose **Run Implementation**, which in turn reruns `power_opt_design`.

3. Click **Save** in the Save Project dialog box to save the Synthesized Design and Implemented Design constraints before launching implementation.

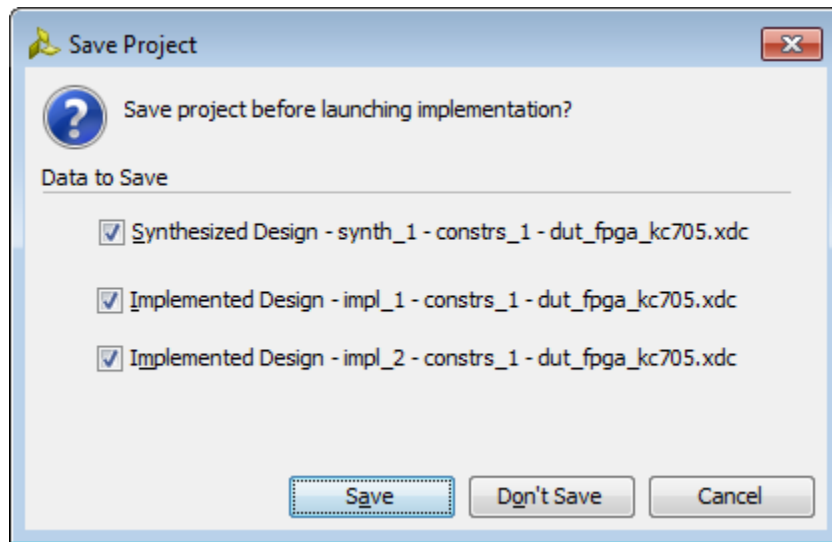


Figure 78: Save Design Constraints Before Rerunning Implementation

Also, select **Implemented Design – impl_2** in the Save Constraints Conflict dialog box to save the changes in constraints from the `set_power_opt` command.

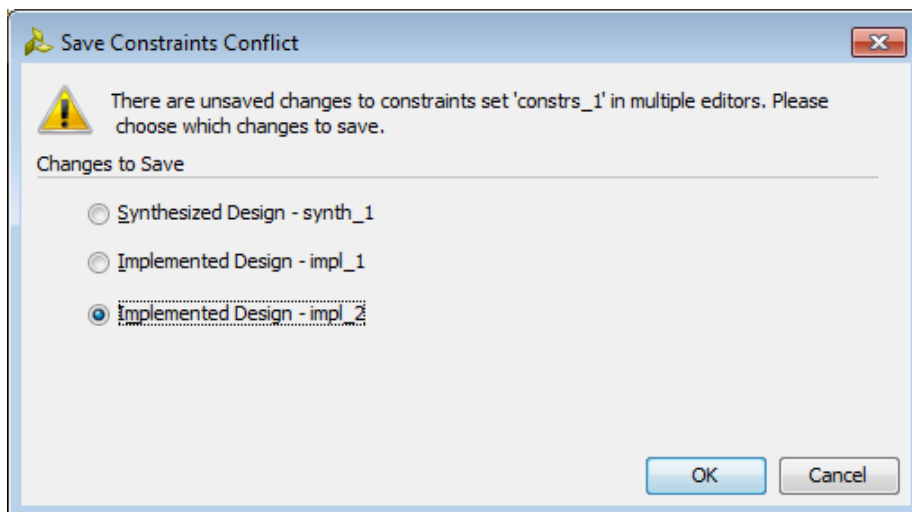


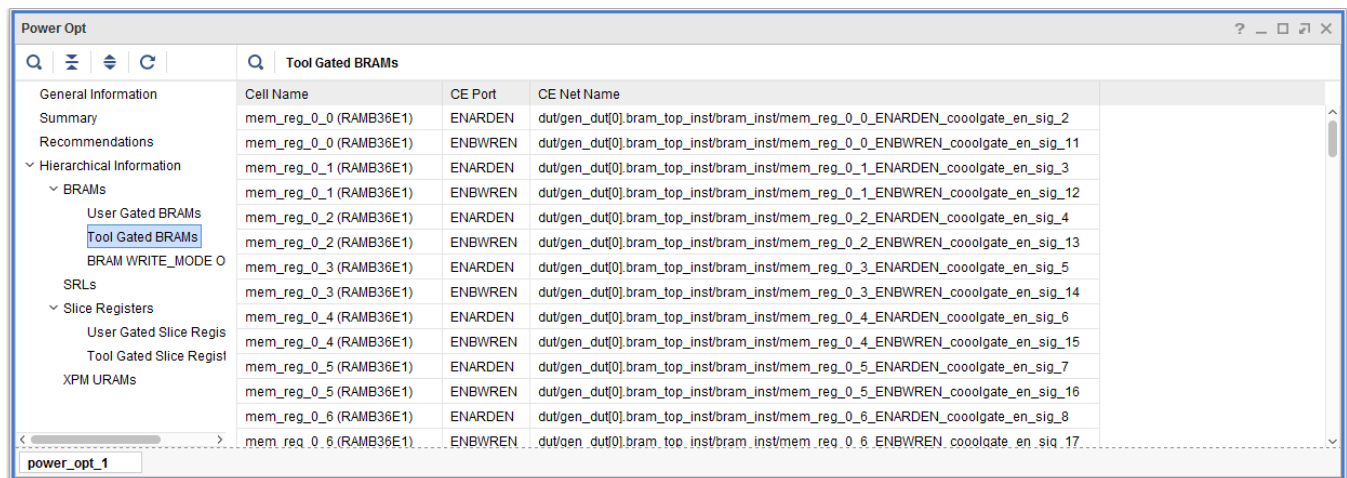
Figure 79: Saving Design Constraints for impl_2

4. In the Implementation Completed dialog box, select **Open Implemented Design** and click **OK**.

Step 5: Running report_power_opt to Examine Tool Optimizations Again

1. In the main menu bar, select **Reports > Report Power Optimization**.
2. In the Report Power Optimization dialog box, type in the **Results name** as power_opt_2. Alternatively, execute this Tcl command in the Tcl Console:

```
report_power_opt -name power_opt_2
```
3. In the generated report **power_opt_2** in the Power Opt window, display **Tool Gated BRAMs**.



Cell Name	CE Port	CE Net Name
mem_reg_0_0 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_0
mem_reg_0_1 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_1
mem_reg_0_2 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_2
mem_reg_0_3 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_3
mem_reg_0_4 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_4
mem_reg_0_5 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_5
mem_reg_0_6 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_6
mem_reg_0_7 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_7
mem_reg_0_8 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_8
mem_reg_0_9 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_9
mem_reg_0_10 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_10
mem_reg_0_11 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_11
mem_reg_0_12 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_12
mem_reg_0_13 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_13
mem_reg_0_14 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_14
mem_reg_0_15 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_15
mem_reg_0_16 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_16
mem_reg_0_17 (RAMB36E1)	ENBWRN	dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_17

Figure 80: Displaying Tool Gated BRAMs

4. Note that this BRAM is no longer in the list of **Tool Gated BRAMs**:

```
dut/gen_dut[0].bram_top_inst/bram_inst/mem_reg_0_0
```

Step 6: Saving Power using UltraScale BRAM in Cascaded Mode

UltraScale architecture-based devices provide the capability to cascade the data out from one BRAM to the next BRAM serially. This will enable the devices to create a deeper block RAM in a bottom-up fashion. When used in cascaded mode, the power consumption is considerably low compared to the BRAM used in non-cascaded mode.

1. Run the steps mentioned in [Step 1: Creating a New Project](#) shown in [Lab 1: Running Power Analysis in the Vivado Tools](#).
 - a. In the **Add Source Files** dialog box, navigate to the `<Extract_Dir>/UltraScale/src` for UltraScale devices.
 - b. In the **Add Constraints** (optional) dialog box, click Add Files and select `dut_fpga_kcu105.xdc` in the file browser. In the directory structure, you will find the `dut_fpga_kcu105.xdc` file below the `/src` folder.
 - c. Select the Kintex-UltraScale KCU105 Evaluation Platform (xcku040-ffva156-2-e FPGA), click **Next**.
2. Review the New Project Summary page. Verify that the data appears as expected and click **Finish**.
3. In the Vivado Options dialog box (**Tools > Options > General**), enter the tutorial project directory in the **Specify project directory** box, so that all reports are saved in the tutorial project directory. Then click **OK**.
4. Click **Run Synthesis** in the Flow Navigator.

The Synthesis Completed dialog box appears after synthesis has completed on the design.

5. Select **Run Implementation** in the Synthesis Completed dialog box and click **OK**.
6. After the Implementation completes, click **Open Implemented Design**.
7. You can see the automatically generated power report `impl_1` in the power tab, which shows as a saved report. This is an autogenerated vectorless power report.
8. Note the total power (**Total On-Chip Power**) in the power report **Summary** page.

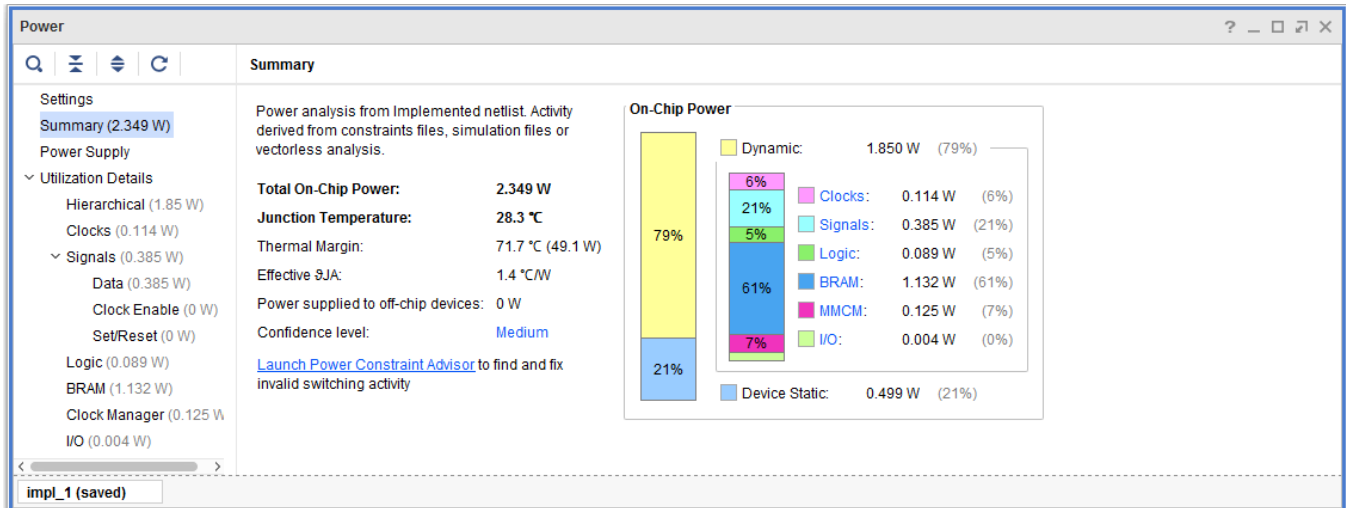


Figure 81: Power Report Summary

9. Select **Hierarchical view** in the **Utilization Details** option on the left panel and observe the Cascaded and Non-cascaded BRAM power.

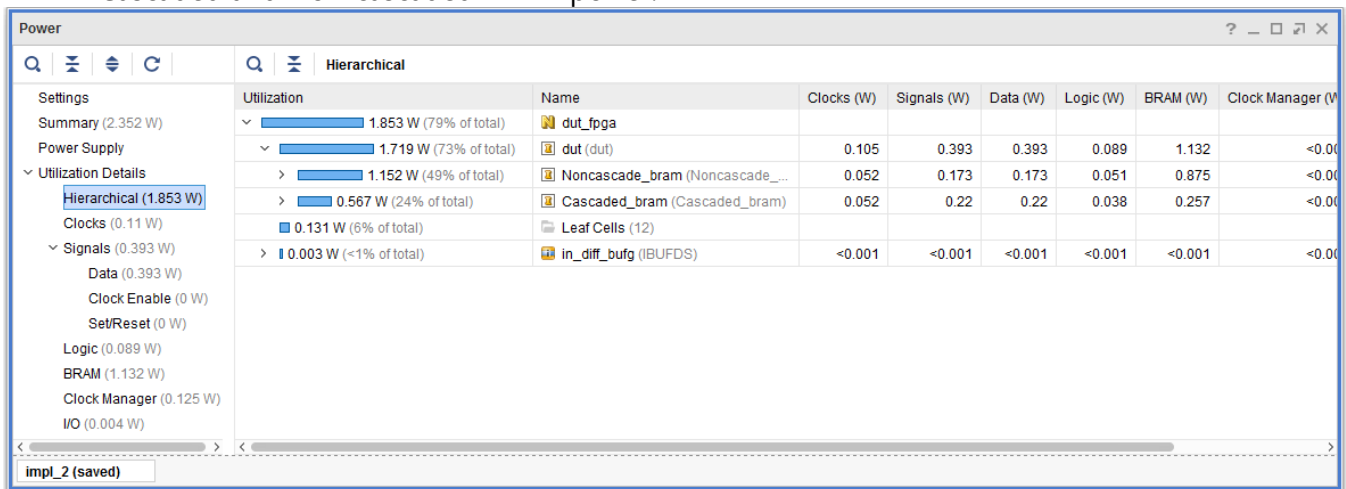


Figure 82: Cascaded and Non Cascaded BRAM Power view

10. You can see 50% to 60% saving in Cascaded BRAM compared to Non Cascaded BRAM.
11. Use the same steps as specified in [Step 1](#), [Step 2](#), and [Step 3](#) to perform SAIF based power analysis using Vivado Simulator.

Conclusion

In this tutorial, we have accomplished the following:

- Used the Report Power dialog box to verify and set device, thermal, and environmental conditions that contribute to power estimation.
- Synthesized the design and estimated the power after synthesis.
- Set switching activities on an I/O port and reran Report Power.
- Ran functional simulation using the Vivado simulator and generated a SAIF file that is input to Report Power for a more accurate power analysis.
- Implemented the design, ran post-implementation timing simulation using the Vivado simulator, and generated a SAIF file that is input to report power for a more accurate power analysis.
- Ran Questa Advanced Simulator post-implementation timing simulation and generated a SAIF file that is input to report power for a more accurate power analysis.
- Performed power measurement on the design implemented in a KC705 and KCU105 Evaluation Boards. Compared the hardware power numbers with the numbers generated by Vivado Report Power.
- Learned how to achieve power optimization as part of an implementation run.
- Examined the power optimization report and selectively turned off power optimizations on a cell in the design.
- Examined the power saving of UltraScale BRAMs in cascaded mode when compared to BRAMs in Non-cascaded mode.

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

© Copyright 2012-2018 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.