

Vivado Design Suite User Guide

Power Analysis and Optimization

UG907 (v2021.2) October 22, 2021

Xilinx is creating an environment where employees, customers, and partners feel welcome and included. To that end, we're removing non-inclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You may still find examples of non-inclusive language in our older products as we work to make these changes and align with evolving industry standards.



Revision History

The following table shows the revision history for this document.

Section	Revision Summary
10/22/2021 Version 2021.2	
General updates	Editorial updates only, No technical content updates.
06/16/2021 Version 2021.1	
Setting Power and Current Budget for Xilinx Devices	Added recommendation.
Power Rail Creation and Management	Added note.
Network on Chip (NoC)	Updated.
MRMAC	Updated figure.

Table of Contents

Revision History	2
Chapter 1: Power in Xilinx Devices	5
Navigating Content by Design Process.....	5
Introduction.....	6
Power Terminology.....	6
Power Supplies in Xilinx Devices.....	9
Xilinx Device Power and the Overall Design Process.....	10
Xilinx Power Estimation, Analysis, and Optimization Tools.....	12
Chapter 2: Estimating Power - Initial Evaluation Stage	15
Introduction.....	15
Power Budgeting.....	15
Chapter 3: Estimating Power - Vivado Design Flow Stage	16
Introduction.....	16
Power Estimation Expectations.....	16
Estimating Power in the Vivado Integrated Design Environment.....	16
Configuring HBM for report_power.....	32
Configuring GTM for report_power.....	34
Chapter 4: Power Analysis and Optimization in the Vivado Design Suite	37
Introduction.....	37
Power Analysis in the Vivado Integrated Design Environment.....	37
Power Optimization Feature.....	64
Chapter 5: Achieving an Accurate Power Analysis Using Vivado Report Power	77
Introduction.....	77
Chapter 6: Versal ACAP and Report Power	92
Introduction to Versal ACAP.....	92

Report Power for Versal ACAP.....	93
AI Engine.....	94
Network on Chip (NoC).....	94
DDR Memory Controller (DDRMC).....	96
MRMAC.....	97
PCIe.....	97
DSP.....	98
PS-PMC.....	99
IO.....	101
CPM.....	102
GTY.....	103
Chapter 7: Tips and Techniques for Power Reduction.....	104
Introduction.....	104
System-Level Power Reduction.....	104
Measuring Power and Temperature.....	106
Design Level Power Reduction.....	108
Appendix A: Additional Resources and Legal Notices.....	115
Xilinx Resources.....	115
Documentation Navigator and Design Hubs.....	115
References.....	115
Training Resources.....	117
Please Read: Important Legal Notices.....	117

Power in Xilinx Devices

Navigating Content by Design Process

Xilinx® documentation is organized around a set of standard design processes to help you find relevant content for your current development task. All Versal™ ACAP design process [Design Hubs](#) can be found on the Xilinx.com website. This document covers the following design processes:

- **System and Solution Planning:** Identifying the components, performance, I/O, and data transfer requirements at a system level. Includes application mapping for the solution to PS, PL, and AI Engine. Topics in this document that apply to this design process include:
 - [Chapter 2: Estimating Power - Initial Evaluation Stage](#)
 - [Chapter 3: Estimating Power - Vivado Design Flow Stage](#)
 - [Chapter 6: Versal ACAP and Report Power](#)
- **Embedded Software Development:** Creating the software platform from the hardware platform and developing the application code using the embedded CPU. Also covers XRT and Graph APIs. Topics in this document that apply to this design process include:
 - [Chapter 2: Estimating Power - Initial Evaluation Stage](#)
 - [Chapter 3: Estimating Power - Vivado Design Flow Stage](#)
 - [Chapter 6: Versal ACAP and Report Power](#)
- **AI Engine Development:** Creating the AI Engine graph and kernels, library use, simulation debugging and profiling, and algorithm development. Also includes the integration of the PL and AI Engine kernels. Topics in this document that apply to this design process include:
 - [Chapter 6: Versal ACAP and Report Power](#)
- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, functional simulation, and evaluating the Vivado® timing, resource use, and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
 - [Chapter 3: Estimating Power - Vivado Design Flow Stage](#)
 - [Chapter 4: Power Analysis and Optimization in the Vivado Design Suite](#)

- [Chapter 6: Versal ACAP and Report Power](#)
- **System Integration and Validation:** Integrating and validating the system functional performance, including timing, resource use, and power closure. Topics in this document that apply to this design process include:
 - [Chapter 3: Estimating Power - Vivado Design Flow Stage](#)
 - [Chapter 4: Power Analysis and Optimization in the Vivado Design Suite](#)
 - [Chapter 6: Versal ACAP and Report Power](#)
- **Board System Design:** Designing a PCB through schematics and board layout. Also involves power, thermal, and signal integrity considerations. Topics in this document that apply to this design process include:
 - [Chapter 3: Estimating Power - Vivado Design Flow Stage](#)
 - [Chapter 4: Power Analysis and Optimization in the Vivado Design Suite](#)
 - [Chapter 7: Tips and Techniques for Power Reduction](#)

Introduction

This chapter provides the terminology used in describing power when implementing Xilinx® devices on a board. It also puts the device development in the greater context of the system being designed and provides a high level description of what to expect at each stage of the design flow. The chapter then describes the Xilinx® tools used for power estimation, analysis, and optimization.



VIDEO: The [Vivado Design Suite QuickTake Video Tutorial: Power Estimation and Analysis Using Vivado](#) shows how Vivado® 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 Tutorial: Power Optimization Using Vivado](#) describes the factors that affect power consumption in a Xilinx device and how Vivado 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.

Power Terminology


The following terminology is used in this guide.

- **Device Static Power:** Device static power is the power from transistor leakage on all connected voltage rails and the circuits required for the device to operate normally, post configuration. This is normally measured by programming a blank bitstream into the device. Device static power is a function of process, voltage, and temperature. This represents the steady state, intrinsic leakage in the device.
- **Design Power:** Design power is the dynamic power of the user design, due to the input data pattern and the design internal activity. This power is instantaneous and varies at each clock cycle. It depends on voltage levels, logic, and routing resources used. This also includes static current from I/O terminations, clock managers, and other circuits that need power when used. It does not include power supplied to off-chip devices.
- **Total On-Chip Power:** Total on-chip power is the power consumed internally within the device, equal to the sum of device static power and design power. It is also known as thermal power.
- **Off-Chip Power:** Off-chip power is the current that flows from the supply source through the device power pins, then out of the I/Os and dissipated in external board components. The currents supplied by the device are generally consumed in off-chip components such as I/O terminations, LEDs, or the I/O buffers of other chips, and therefore do not raise the device junction temperature.

Note: Negative off-chip power dissipated is the power that is sourced from external source and dissipated inside our device.

- **Power-On Current:** Power-on current is transient current that occurs when power is first applied to the device. This current varies for each voltage supply and depends on the device construction as well as the ability of the power supply source to ramp up to the nominal voltage. This current also depends on the device's operating conditions, such as temperature and sequencing between the different supplies. Power-on current is generally lower than operating current due to architectural enhancements as well as adherence to proper power-on sequencing.
- **Ambient Temperature (°C):** Ambient temperature is the temperature of the air immediately surrounding the device under the expected system operating conditions.
- **Effective Thermal Resistance to Air (Θ_{JA} (°C/W)):** Effective thermal resistance to air is also known as *Theta-JA* and *TJA*. This coefficient defines how power is dissipated from the device silicon to the environment (device junction to ambient air). It includes contributions from all elements, from the silicon chip dimensions to the surrounding air, plus any material in between, such as the package, the PCB, any heat sink, and airflow. Typically this combines thermal resistance and interdependencies from the two main paths by which the generated heat can escape onto the environment:
 - Upward from the die to the air (junction to air or Θ_{JA}).
 - Downward from the die through the board and into the air (junction to board or Θ_{JB}).

For detailed information on thermal resistance, refer to *7 Series FPGAs Packaging and Pinout Product Specification (UG475)*, *UltraScale and UltraScale+ FPGAs Packaging and Pinouts Product Specification (UG575)* and for thermal resistance in Versal devices, refer to *Versal ACAP Packaging and Pinouts Architecture Manual (AM013)*.

 **IMPORTANT!** *The thermal data mentioned in this user guide is for the device/package comparison only. Do not use these values for thermal simulations. Use the thermal models provided on Xilinx.com.*

Device Characterization

- **Advance:** Devices with the Advance designation have data models primarily based on simulation results or measurements from early production device lots. This data is typically available within a year of product launch. The Power model data with this designation is considered relatively stable and conservative, although some under or over-reporting can occur. Advance data accuracy is considered lower than the Preliminary and Production data.
- **Preliminary:** Devices with the Preliminary designation are based on complete early production silicon. Almost all the blocks in the device fabric are characterized. The probability of accurate power reporting is improved compared to Advance data.
- **Production:** Devices with the Production designation are released after enough production silicon of a particular device family member has been characterized to provide full power correlation over numerous production lots. Device models with this characterization data are not expected to evolve further.

The accuracy of any power estimation is derived from the information input to the models. Report Power uses the following models based on the device characterization:

- **Advance:** +/-25%
- **Preliminary:** +/-20%
- **Production:** +/-10%

These models are same as those used in Xilinx® Power Estimator, however these models have more accurate inputs such as exact resource usage as well as actual trace lengths, so the accuracy of a high confidence report power estimate is 5% better than that of XPE.

Note: When the maximum process is used and the junction temperature (Tj) is accurate based on thermal simulation, the accuracy of the model only needs to be applied to the dynamic portion of the estimation.

Signal Rate

Signal rate is the number of times an element changes state (high-to-low and low-to-high) per second. Xilinx tools express this as millions of transitions per seconds (Mtr/s). For example, if a signal changes at every four clocks cycle with respect to a 100 MHz (10 ns) Clock, then the Signal Rate is: $1/(4*10\text{ ns}) = 25\text{ Mtr/s}$.

Toggle Rate

Toggle rate (%) is the rate at which the output of a synchronous logic element switches with respect to a given clock input. It is modeled as a percentage between 0 - 100%. A toggle rate of 100% means that on average the output toggles once during every clock cycle. As an example, if a signal changes at every four clock cycles with respect to a clock of any frequency, then the Toggle Rate is: $(1/4) * 100 = 25\%$.



IMPORTANT! The toggle rate for clock nets is always 200%, which means that the net toggles twice in a cycle.



TIP: Ideally a synchronous net changes at the most once per clock (except DDR nets); thus the maximum toggle rate is 100%. If a synchronous net is prone to glitches, use Signal Rate to specify the switching activity.

For asynchronous elements such as nets and logic that are not synchronized with a clock, the toggle rate cannot be computed. The Vivado® power tools expect the use of Signal Rate for these kinds of elements.

By default the primary inputs of the design are not associated with a specific clock. Use the `set_input_delay` constraint to associate a clock with the primary inputs. If you do not associate a clock, the power tools compute the toggle rate with respect to either the capturing clock or the fastest clock in the design.

Static Probability

Static probability defines the fraction of time during which the considered element is driven at a high (1'b1) logic level and the valid range is 0 to 1. As an example, if a signal is at Logic 1 for 40 ns in a duration of 100 ns, the static probability = $40/100 = 0.4$.



TIP: Static Probability = 1 represents that the considered element is held at Logic 1 throughout the analysis duration and never toggles. Similarly, Static Probability=0 represents that the considered element is held at Logic 0 throughout the analysis duration and never toggles.

Power Supplies in Xilinx Devices

Multiple power supplies are required to power Xilinx devices. Separate sources provide the required power for different resources of the device. This allows different resources to work at different voltage levels for increased performance or signal strength while preserving a high immunity to noise and parasitic effects.

For power supplies available in Versal ACAP, refer Versal ACAP Resources and Corresponding Power Supply table in *Xilinx Power Estimator User Guide for Versal ACAP* ([UG1275](#)).

For power supplies available in other Xilinx® devices, refer FPGA Resources and their Power Supply table in *Xilinx Power Estimator User Guide (UG440)*.



TIP: Xilinx offers optimized solutions to help you find the right power delivery solution for your application. The hardware verified reference designs ensure that all Xilinx power specifications are optimally met and follow the supported power up/down sequencing. To explore these solutions, see the [Power Delivery Solutions](#) tab of the [Power](#) page on the Xilinx website.

Xilinx Device Power and the Overall Design Process

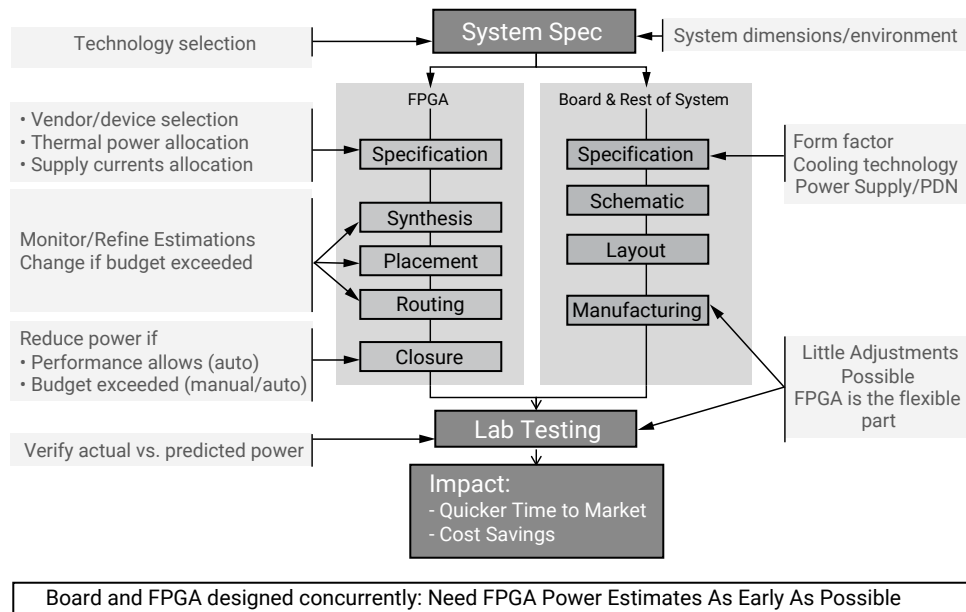
From project conception to completion there are many different factors to consider that influence power. Omitting for a moment all other constraints (functionality, performance, cost, and time to market), power related tasks can be sorted into two separate classes.

- **Physical domain:** Enclosure, board shape, power supply and power distribution network (PDN), thermal power dissipation system.
- **Functional domain:** Area, performance, I/O interfaces signal integrity.

The next chapters demonstrate the interdependencies between these two classes. These classes differ in that the physical domain involves hardware decisions, while the functional domain mostly involves design creation. Typically, hardware selection and sizing occurs very early in the design flow to allow time to build prototype boards. The effect of a device functionality on power consumption can be estimated early on, then refined as more and more of the design logic is completed. The following figure illustrates a typical system design process, and highlights power-related decision points. The figure demonstrates that, at the time you select your device and associated cooling parts, the device logic is not yet available. Therefore, a careful methodology to estimate the device logic power requirements is needed. Methodologies are discussed in:

- [Chapter 2: Estimating Power - Initial Evaluation Stage](#)
- [Chapter 3: Estimating Power - Vivado Design Flow Stage](#)

Figure 1: Power in the Device Design Process

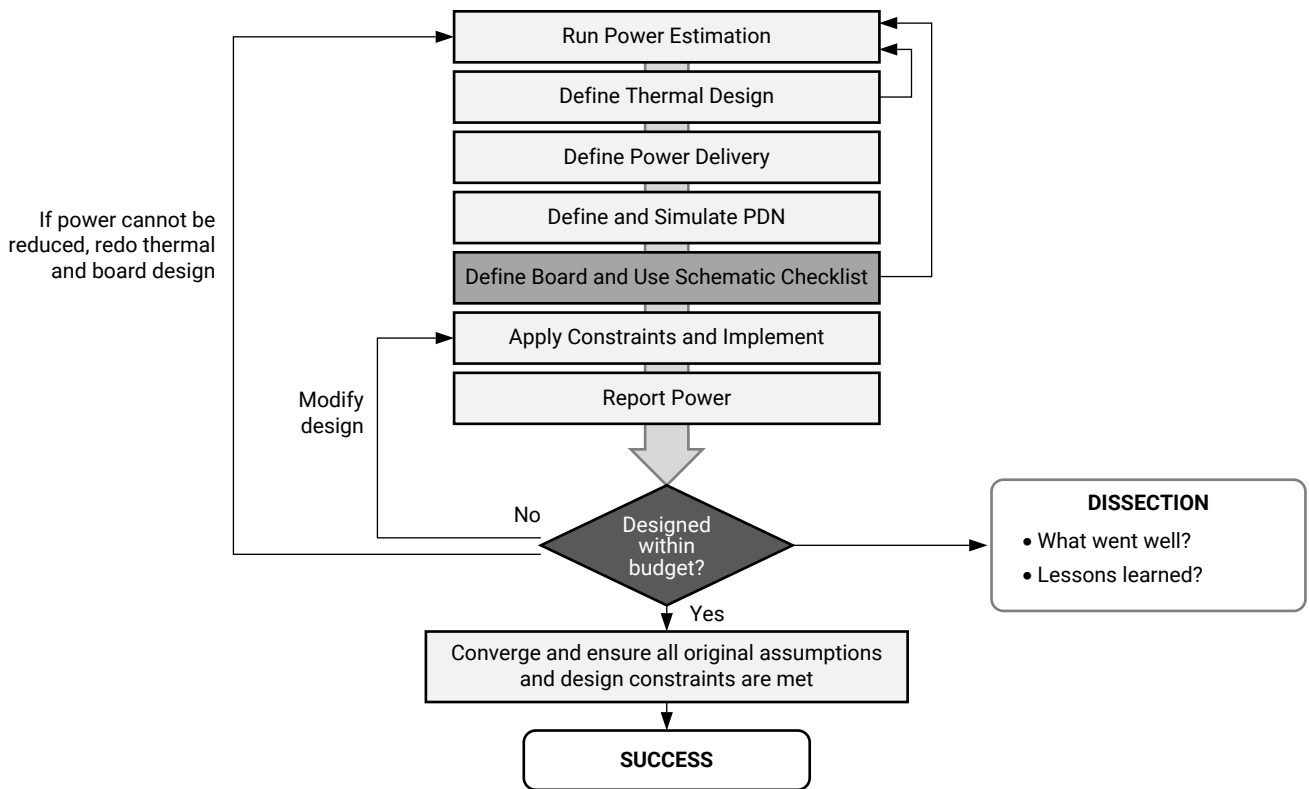


The following sections provide methodologies to analyze and reduce power consumption throughout the design process.

Power Estimation in System Design

Power estimation is a critical starting point in system/board design flow. The initial estimation is the starting point for thermal and power delivery design. The result of all these stages should then be used as design constraints in Vivado® or Vitis™ project. The following figure shows an overview of this flow. For more information, see Power and Thermal Considerations section in *Versal ACAP Design Guide* (UG1273).

Figure 2: Power and Thermal Methodology

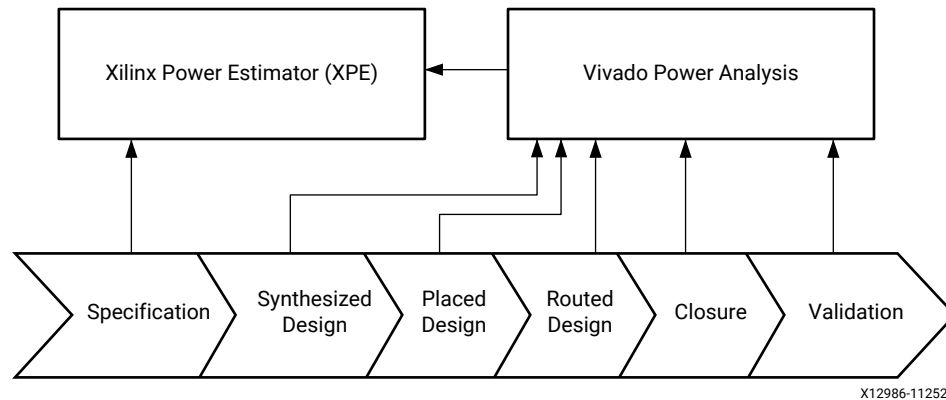


X24843-111920

Xilinx Power Estimation, Analysis, and Optimization Tools

Xilinx® provides a suite of software tools and documentation to help you evaluate the thermal and power supply requirements of your device throughout the design cycle. The following figure shows the tools available at each stage of the device design cycle. Some of the tools are standalone while others are integrated into the implementation software, to align with the environment and information available to you at each stage of the design process. All tools have communication channels so you can exchange information back and forth to be most efficient with your analysis.

Figure 3: Vivado Power Estimation and Analysis Tools in the Design Process



Xilinx Power Estimator (XPE)

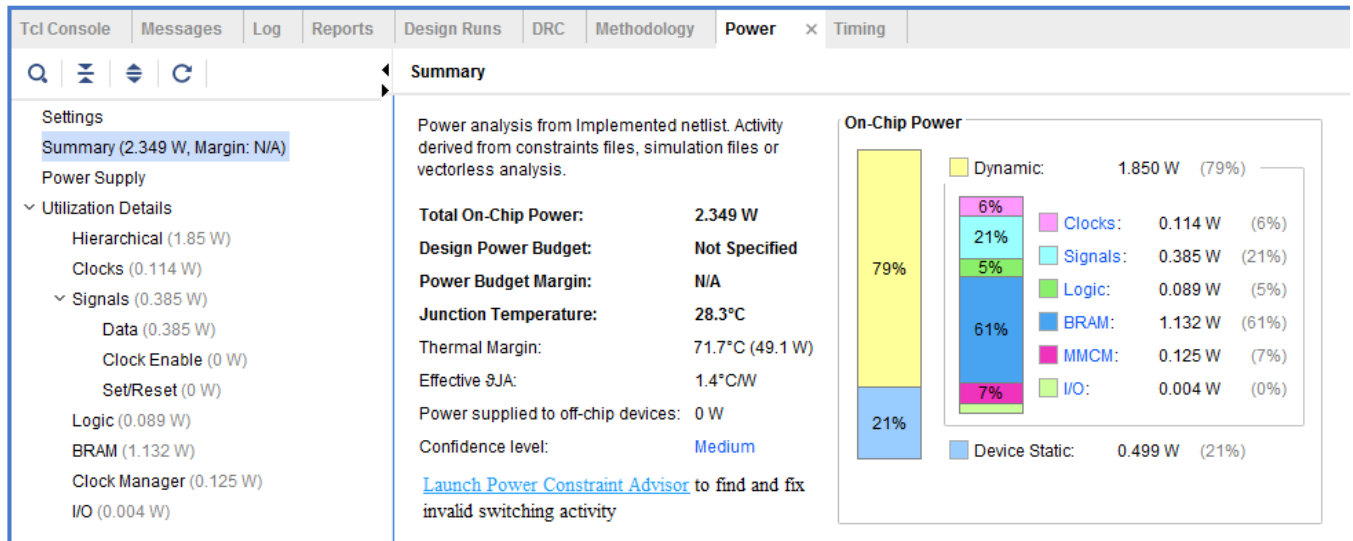
The Xilinx® Power Estimator (XPE) spreadsheet is a power estimation tool typically used in the pre-design and pre-implementation phases of a project. XPE assists with architecture evaluation and device selection and helps in selecting the appropriate power supply and thermal management components that may be required for your application. The XPE interface lets you specify design resource usage, activity rates, I/O loading, and many other factors which XPE then combines with the device models to calculate the estimated power distribution.

XPE is also commonly used later in the design cycle during implementation and power closure to, for example, evaluate power implications of engineering change orders (ECO). For large designs implemented by multiple teams, the project leader can use XPE to import usage and activity for each team's module, then monitor the total power and reallocate the power budget to ensure constraints are met. For more information on using the Xilinx Power Estimator, see *Xilinx Power Estimator User Guide* (UG440) and *Xilinx Power Estimator User Guide for Versal ACAP* (UG1275).

Vivado Power Analysis

The Vivado® power analysis feature performs power estimation through all stages of the flow: post-synthesis, post-placement, and post-routing. It is most accurate at post-route because it can read the exact logic and routing resources from the implemented design. The following figure presents the Summary power report and the different views of your design that you can navigate: by clock domain, by type of resource, and by design hierarchy. Within the Vivado Integrated Design Environment (IDE), you can adjust environment settings and design activity so you can evaluate how to reduce your design supply and thermal power consumption. You can also cross-probe into the design from the power report, which aids in identifying and evaluating high power consuming hierarchy/resources used in the design.

Figure 4: Vivado Power Analysis



Vivado Power Optimization

The Vivado® design tools offer a variety of power optimizations to minimize dynamic power consumption by up to 30% in your design. These optimizations use the equivalent techniques of a complex ASIC clock gating to minimize switching activity without affecting the design functionality. The power optimizations can be applied on the entire design or on selected portions of the design. In Vivado, you can perform power optimization using the Vivado IDE or using Tcl commands.

Estimating Power - Initial Evaluation Stage

Introduction

This chapter describes a methodology to evaluate your design's power consumption during the initial evaluation stage of the design cycle. You will work in Xilinx[®] Power Estimator during this stage of the design cycle. If you have already completed the initial evaluation stage, go to the next chapter, which describes a methodology to evaluate your design's power consumption in the later stage of the design cycle. At this stage, you will use the Vivado[®] Design Suite, which automates and simplifies power estimation.

Power Budgeting

At this stage you have determined that Xilinx[®] device is the most effective technology for your application. Now you need to define which vendor, family, and package can best fit your functionality, performance, cost, and power budgets. In terms of power, you must estimate the total device power requirements even before any logic is developed. Understanding the total power requirements will help you define your power delivery and cooling system specifications.

For more information on power estimation, see *Seven Steps to an Accurate Worst-Case Power Analysis using the Xilinx Power Estimator* ([XAPP1348](#)).

Estimating Power - Vivado Design Flow Stage

Introduction

This chapter describes tool features in the Vivado® Design Suite that automate or simplify power estimation during the design flow stage. Once you generate and analyze a power estimation in the Vivado Design Suite, see [Chapter 7: Tips and Techniques for Power Reduction](#) for techniques to investigate and modify your system, to minimize the device power consumption.

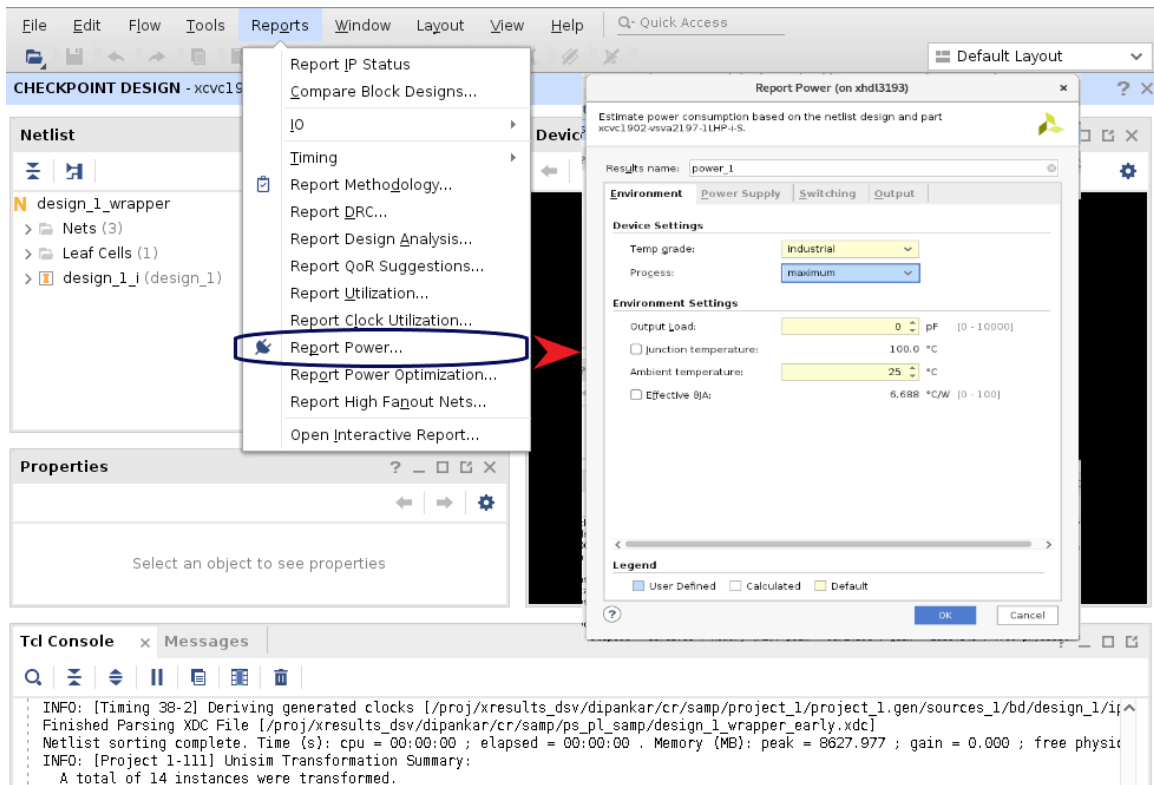
Power Estimation Expectations

As your design flow progresses through synthesis and implementation you will want to monitor and verify the power consumption regularly. You must ensure that thermal dissipation remains within budget so that you can detect and act early on if any area approaches your constraints. The accuracy of the power estimates varies depending on the design stage when the power was estimated.

Estimating Power in the Vivado Integrated Design Environment

This section covers power analysis using Report Power in the Vivado® integrated design environment. These instructions assume this is the first time you are setting up a power analysis after Synthesis. Therefore, provide the tool with the relevant activity information. For subsequent runs, you can choose whether to use Report Power in the Vivado integrated design environment to navigate your Power report or use the Tcl equivalent (`report_power`) to bypass the Vivado integrated design environment and review the text power report directly. The following figure shows the Vivado power analysis.

Figure 5: Vivado Power Analysis - Supplying Relevant Input Data for Analysis



Setting Up Power Analysis from the Vivado IDE

Perform the following steps to specify the environment, activity, supply, and tool defaults in the Report Power dialog box.

1. Select **Flow** → **Open Synthesized Design** or **Flow** → **Open Implemented Design**.
Alternatively, you can make this selection in the Flow Navigator.
2. Select **Reports** → **Report Power**.
Alternatively, you can select Report Power in the Flow Navigator.
3. In the Report Power dialog box, adjust device environment and tool settings.
 - Navigate to different tabs in the Report Power dialog box and adjust all settings to closely match your environment.
 - Environment and voltage settings have a large influence on device static power.
 - Activity rates and voltage settings largely influence dynamic power calculations.
 - When unsure of a particular setting, use the default value.
 - If you have an activity file from simulation results, you can specify it in this dialog box.

For more information on these settings, see [Review Device/Design Settings and Adjust Activity for Known Elements](#).

4. Specify the name of the report.

Running Power Analysis from the Vivado IDE

In the Report Power dialog box, click **OK** to start the power analysis. The tool does the following.

1. Takes into account the environment, device, and tool options.
2. Reads the netlist connectivity and configuration.
3. Applies activity factors for the nodes you defined.

A node is a component such as a net, pin, or port.

4. Determines activity for any remaining undefined nodes before computing the thermal and supply power.

Power analysis uses different sources of information for activity definition, including:

- Simulation files (SAIF)
- Automatic calculations using a vectorless power analysis methodology
- Manual definition using the `set_switching_activity` Tcl command

For more information, see [Running Power Analysis from the Tcl Prompt](#).

Vectorless (Probabilistic) Estimation

When design node activity is not provided either from you or from the simulation results, the vectorless power estimation algorithms are capable of predicting this activity. The vectorless engine assigns initial *seeds* (default signal rates and static probability) to all undefined nodes. Then, starting from the design primary inputs it propagates activity to the output of internal nodes, and repeats this operation until the primary outputs are reached. The algorithm understands the design connectivity and resource functionality and configuration. Its heuristics can even approximate the glitching rate for any nodes in the netlist. Glitching occurs when design elements change states multiple times in between active clock edges before settling to a final value. The vectorless propagation engine is not as accurate as a post-route simulation with a reasonably long duration and realistic stimulus, but it is an excellent compromise between accuracy and compute efficiency.

Note: The vectorless power estimator does not propagate activity to the output ports of GTs. If any design logic depends on these activity rates, you must explicitly specify the activity rates on GT outputs using `set_switching_activity -type <rx_data|tx_data>` commands to achieve an accurate analysis.




TIP: The vectorless power estimation is an average power estimation for the design, unless you have specifically overridden switching rates and static probability for the design.

User Input to Improve Vectorless Estimation

In any design, users typically know the activity of specific nodes because they are imposed by the system specification or the interfaces with which the device communicates. Providing this information to the tools, especially for nodes which drive multiple cells in the device (Set, Reset, Clock Enable, or clock signals), will help guide the power estimation algorithms. These nodes include:

- **Clock Activity:** Users typically know the exact frequency of all device clock domains, whether externally provided (input ports), internally generated, or externally supplied to the printed circuit board (output ports). The design should have at least one clock specified using the `create_clock` constraint. If no clock is defined, then Report Power issues a warning message and uses a 10 GHz clock frequency for switching activity computations.
- **I/O Data Ports:** With your knowledge of the exact protocols and format of the data flowing in and out of the device, you can usually specify signal transition rate and/or signal static probability rate in the tools for at least some of the I/Os. For example, some protocols have a DC balanced requirement (signal static probability rate = 50%) or you may know how often data is written or read from your memory interface, so you can set the data rate of strobe and data signals. If no user activity rate is specified on primary inputs, Report Power assigns a default static probability of 0.5 and a default toggle rate of 12.5%.
- **I/O and Internal Control Signals:** With your knowledge of the system and the expected functionality you may be able to predict the activity on control signals such as Set, Reset and Clock Enable. These signals typically can turn on or off large pieces of the design logic, so providing this activity information increases the power estimation accuracy. If a primary input is found to be reset (that is, directly connected to the RESET pin of sequential elements), then the tool assigns a default static probability of 0 and a default signal rate of 0. Similarly, if a primary input is found to be Clock Enable (that is, directly connected to the CE pin of sequential elements), then the tool assigns a default static probability of 0.99 and a default signal rate of 2.

 **RECOMMENDED:** *Providing node activity information to the tools, especially for nodes which drive multiple cells in the device (Set, Reset, Clock Enable, or clock signals), helps guide the power estimation algorithms.*

Note: The vectorless power estimator does not propagate activity to the output ports of GTs. If any design logic depends on these activity rates, you must explicitly specify the activity rates on GT outputs using `set_switching_activity -type gt_txdata|gt_rxdata` commands to achieve an accurate analysis.

Vector (SAIF) Based Power Estimation

In parallel with all stages of the design development, perform simulations to verify that the design behaves as expected. Different verification techniques are available depending on the design development state, the design complexity, or company policy. The following sections highlight the valuable data you can capture and common pitfalls related to using this data to perform power analysis. An important factor for getting an accurate power estimation is that the design activity needs to be realistic. It should represent typical or worst case scenario for data

coming into the simulated block. This type of information is not necessarily provided while performing verification or validating functions. Sometimes, invalid data is given as input to verify that the system can handle it and remain stable even when invalid data or commands are given to it. Using such test cases to perform power analysis may result in inaccurate power estimation because the design logic is not stimulated as it would be under typical system operation.

- **System Transaction Level:** Very early in the design cycle, you may have created a description of transactions which occur between devices on a PCB or between the different functions of your device application. You can extract from this the expected activity per functional block for certain I/O ports and most of the clock domains. This information helps you fill in the Xilinx® Power Estimator spreadsheet.
- **Device Description Level:** While defining the RTL for your application you may want to verify the functionality by performing behavioral simulations. This helps you verify the data flow and the validity of calculations to the clock cycle. At this stage, exact device resources used, count, and configuration data is not available. You can manually extrapolate resource usage and extract activity for I/O ports or internal control signals (Set, Reset, Clock Enable). This information can be applied to refine the Xilinx Power Estimator spreadsheet information. Your simulator should be able to extract node activity and export it in the form of a SAIF file. You can save this file for more accurate power analysis in the Vivado® design flow, for example after place and route, if you do not plan to run post-implementation simulations.
- **Device Implementation Level :** Simulation can be performed at different stages in the implementation process with different outcomes in terms of the power-related information which can be extracted. This additional information may also be used to refine the Xilinx Power Estimator spreadsheet and the Vivado power analysis as well. It may also save I/O ports and specific module activity, which can later be reused in the Vivado power analysis feature at any stage of the design completion (post-synthesis, post-placement, or post-route).
 - **Post Synthesis:** The netlist is mapped to the actual resources available in the target device.
 - **Post Placement:** The netlist components are placed into the actual device resources. With this packing information the final logic resource count and configuration becomes available and you can update the Xilinx Power Estimator spreadsheet for your design.
 - **Post Routing:** After routing is complete all the details about routing resources used and exact timing information for each path in the design are defined. In addition to verifying the implemented circuit functionality under best and worst case gate and routing delays, the simulator can also report the exact activity of internal nodes and include glitching. Power analysis at this level provides you the most accurate power estimation before you actually measure power on your prototype board.

Specifying Switching Activity for the Analysis

Simulation Results (SAIF File)

Vivado® Report Power matches nets in the design database with names in the simulation results netlist. The simulation results netlist is a SAIF (Switching Activity Interchange Format) file. For all nets matched, Vivado Report Power will apply switching activity and static probability to calculate the design power. Simulation results may have been generated early in the design flow, before synthesis or placement and routing. In this case it is preferable to capture from the simulation results only module I/O ports activity and let the vectorless engine estimate internal node activity. Functional simulations do not capture glitch activity. Also, Report Power may not be able to match all nodes between the design and the simulation netlist because of logic transformations which happen during implementation (optimizations, replications, gating, retiming, etc.). Nevertheless most primary ports and control signals will be matched and this information provides the tool with realistic activity for the matched nodes. The activity is propagated by the vectorless engine onto the unmatched design portion and increase the accuracy of the power estimation. Make sure to use the following type of simulation results:

- Ensure test vectors and inputs to the simulation represent the typical or expected behavior of the design. Error handling and corner case simulations do not typically stimulate the logic in the way it would be stimulated under normal operation.
- Post-implementation simulation results are preferred over behavioral simulation results. Full timing simulation would be much more accurate, because it helps with capturing timing glitch information into the SAIF results.

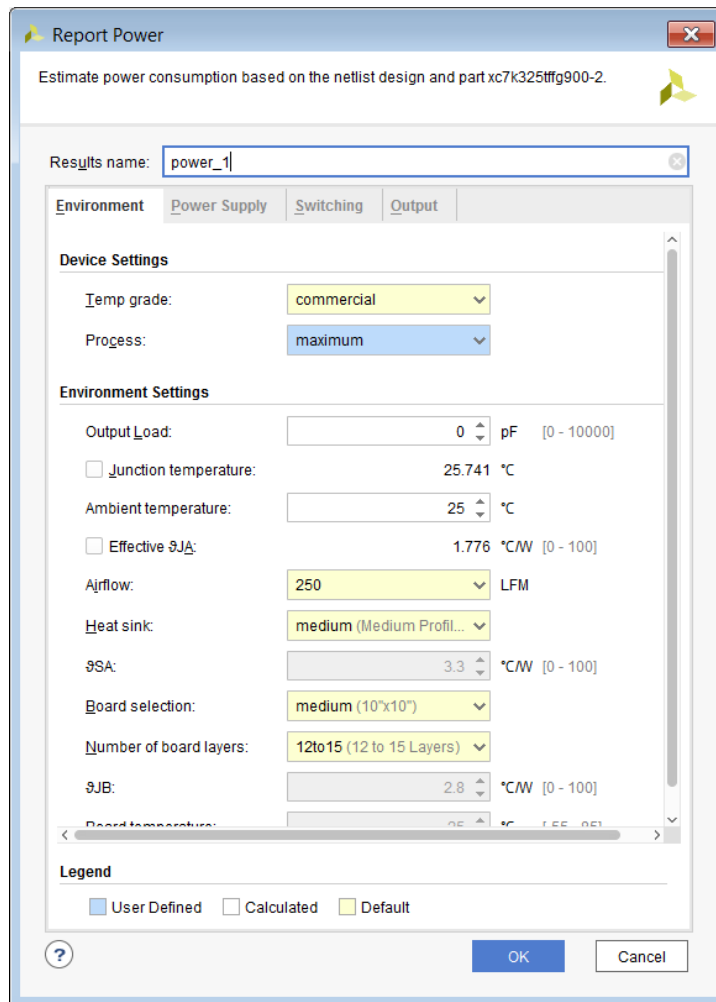
★ **IMPORTANT!** Report power uses vectorless algorithm and default switching rates to compute the activity on un-matched design nets with the given SAIF file. This results in different toggle rates in Power Report and it eventually reflects in XPE too. It is recommended not to use VHDL generated `.saif` files as the timing simulation is supported in Verilog only.

★ **IMPORTANT!** To generate a SAIF file from the Vivado simulator for power analysis, refer to the Vivado Design Suite User Guide: Logic Simulation (UG900). To generate a SAIF file from the Mentor Graphics ModelSim simulator for power analysis within the Vivado® Design Suite, see Xilinx® Answer Record 53544. For full timing simulation, generate a design timing information (SDF) file using the `write_sdf` command and annotate it while running simulation.

Review Device/Design Settings and Adjust Activity for Known Elements

You can open the Report Power dialog box from the Flow Navigator window in the Vivado® integrated design environment. In this dialog box, you can review power settings and adjust activity for known elements in your design as shown in the following figure.

Figure 6: Report Power Dialog Box



Review the different input tabs to make sure they accurately represent your expected system. The following Input Tabs are available in Report Power Dialog box:

- Environment Tab
- Power Supply Tab
- Switching Tab
- Output Tab

Environment Tab

Review the different user-editable selections in the Environment tab. Make sure the process, voltage and environment data closely match your expected environment. These settings have a significant influence on the total estimated power. The user-editable selections in the Environment tab are:

- **Device Settings:**
 - **Temp grade:** Select the appropriate grade for the device (typically Commercial or Industrial). Some devices may have different device static power specifications depending on this setting. Setting this properly will also allow for the proper display of junction temperature limits for the chosen device.
 - **Process:** For the purposes of a worst-case analysis, the recommended process setting is Maximum. The default setting of Typical will give a closer picture to what would be measured statistically, but changing the setting to Maximum will modify the power specification to worst-case values.
- **Environment Settings:**
 - **Output Load (pF):** The board and other external capacitance driven by the outputs in the I/O ports.
 - **Junction temperature (°C):** Specify a value to force the device junction temperature to a specific value. For worst-case analysis, force this value using user override check-box to TJ (Max) based on the temperature grade of the part.
 - **Ambient temperature (°C):** Specify the maximum possible temperature expected inside the enclosure that will house the device design. This, along with airflow and other thermal dissipation paths (for example, the heatsink), will allow an accurate calculation of Junction Temperature which in turn allows more accurate calculation of the device static power.
 - **Effective Θ_{JA} (°C/W):** Specify the value for custom Θ_{JA} which is generally derived from thermal modeling. Ambient Temperature and Effective Θ_{JA} are to be set if the values are derived from thermal simulations for better accuracy in estimation.
 - **Airflow (LFM):** The airflow across the chip is measured in Linear Feet per Minute (LFM). LFM can be calculated from the fan output in CFM (Cubic Feet per Minute) divided by the cross sectional area through which the air passes. Specific placement of the device and/or fan may have an effect on the effective air movement across the device and thus the thermal dissipation. Note that the default for this parameter is 250 LFM. If you plan to operate the device without active air flow (still air operation) then the 250 LFM default has to be changed to 0 LFM.
 - **Heat sink (if available):** If a heatsink is used and more detailed thermal dissipation information is not available, choose an appropriate profile for the type of heatsink used. This, along with other entered parameters, will be used to help calculate an effective Θ_{JB} , resulting in a more accurate junction temperature and quiescent power calculation. Note that some types of sockets may act as heatsinks, depending on the design and construction of the socket.
 - **Board selection and Number of board layers (if available):** Selecting an approximate size and stack of the board will help calculate the effective Θ_{JB} by taking into account the thermal conductivity of the board itself.

- **Θ_{JB}** : In the event more accurate thermal modeling of the board and system is available, Θ_{JB} (printed circuit board thermal resistance) should be used to specify the amount of heat dissipation expected from the device.

Note: For most accurate power estimation, Xilinx recommends specifying maximum ambient temperature. The application needs to support T_a and Θ_{JA} of the thermal solution, this allows the power estimation to more accurately represent Junction Temperature (T_j) and the static power of the device assuming maximum process is specified.

Power Supply Tab

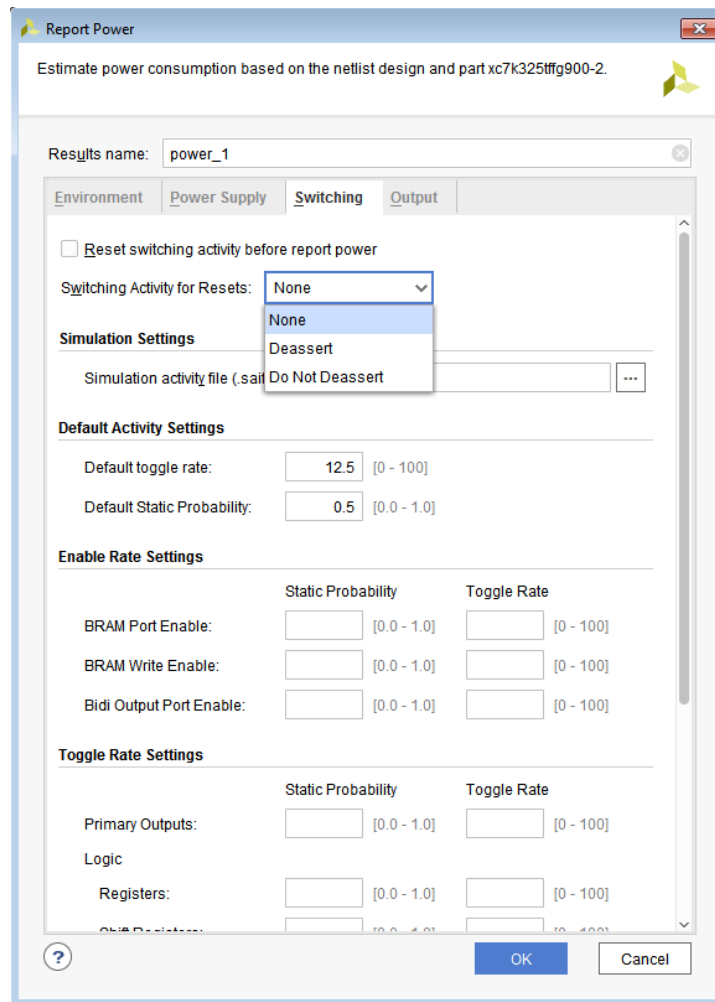
If this information is known, ensure to set the voltage levels correctly for different power supply sources in the Power Supply tab. Voltage is a large factor contributing to both static and dynamic power.

Note: Xilinx recommends that the TYP voltage is used for power estimation as this means that the power delivery solution has a balanced positive and negative range to allow DC tolerance and AC Ripple.

Switching Tab

In the Switching tab, review the design's Simulation and Default Activity Settings. The clocks constrained in the design can also be viewed on this page as shown in the following figure.

Figure 7: Report Power Switching Settings



- **Reset switching activity before report power:** This check-box if enabled, clears/resets all the switching activity applied before running report power.
- **Switching Activity for Resets:** Sets the Switching Activity for control sets. See [Deassertion of Resets](#) for more information.
- **Simulation Settings:**
 - **Simulation activity file (.saif):** Vivado® Report Power takes input SAIF simulation data generated for the design. Report Power then matches nets in the design database with names in the simulation results netlist. See [Specifying Switching Activity for the Analysis](#), for a description of how input from a simulation results (SAIF) file can be used for a more accurate power analysis.
- **Default Activity Settings:**

- **Default toggle rate:** The default toggle rate to be used in power analysis on the primary inputs of the design. The default toggle rate is set on those primary input nets whose switching activity is not specified by the user, simulation data or constraints of the design. On asynchronous inputs the toggle rate is set with respect to the capturing clock in the design. Valid values are: $0 \leq \text{value} < 100$. The default value is 12.5.
- **Default Static Probability:** The default static probability to be used in power analysis on the design. The default static probability is set on those primary inputs whose switching activity is not specified by the user, simulation data or constraints of the design. Valid values are: $0 \leq \text{value} \leq 1$. The default value is 0.5.
- **Enable Rate Settings:**
 - **Block RAM Port Enable:** Sets the activity rate of all the Block RAM enable signals of the design to the value specified.
 - **Block RAM Write Enable:** Sets the activity rate of all the Block RAM write enable signals of the design to the value specified.
 - **Bidi Output Port Enable:** Sets the activity rate of all the Bidirectional I/O enable signals (T pin of IOBUF) of the design to the value specified.

Note: Specify Static Probability and the Toggle Rate together.
- **Toggle Rate Settings:**
 - **Primary Outputs:** Sets the switching activity rate of all the enable signals (i.e., T pin of OBUFT) of the primary outputs of the design to the value specified.
 - **Logic:**
 - **Registers:** Sets switching activity rate on Output pins of all the Registers in the design.
 - **Shift Registers:** Sets switching activity rate on Output pins of all the Shift Registers in the design.
 - **Distributed RAMs:** Sets switching activity rate on Data Outputs pins of all the Distributed RAMs in the design.
 - **LUTs:** Sets switching activity rate on Outputs pins of all the LUTs in the design.
 - **DSPs:** Sets switching activity rate on Data Outputs pins of all the DSPs in the design.
 - **Block RAMs:** Sets switching activity rate on Data Outputs pins of all the Block RAMs in the design.
 - **GTs (Serial Transceivers):**
 - **RX Data:** Sets switching activity rate on RX Data Output pins of all the GTs in the design.

- TX Data:** Sets switching activity rate on TX Data Output pins of all the GTs in the design.

Note: Specify *Static Probability* and *Toggle Rate* together. See the description of the `set_switching_activity` command under [Netlist Element Activity](#), for more information and guidelines.
- Constrained Clocks:** Expanding *Constrained Clocks* lists all the clocks that are constrained in the design. Review the clock frequencies and ensure they are accurate.



TIP: Make sure all primary clocks are specified. The design clocks are identified based only on `create_clock` or `create_generated_clock` constraints.



RECOMMENDED: Xilinx® recommends that you use the exact clock frequencies in your design for more accurate power calculation.

Output Tab

Output tab displays various power result files. The Output tab contains the following settings:

- Output Text File:** For project documentation you may want to save the power estimation results. In other circumstances you may be experimenting with different mapping, placement, and routing options to close on performance or area constraints. Saving power results for each experiment will help you select the most power-effective solution when several experiments meet your requirements.
- Output XPE file (for Xilinx® Power Estimator):** This file, when selected, saves all the environment information, device usage, and design activity in a file (.xpe) which you can later import into the Xilinx Power Estimator spreadsheet. This proves quite useful when your power budget is exceeded and you don't think that software optimization features alone will be able to meet your budgets. In this case, import the current implementation results into Xilinx Power Estimator, explore different mapping, gating, folding, and other strategies, and estimate their impact on power before modifying the RTL code or rerunning the implementation. You can also compare your assumptions in the Xilinx Power Estimator spreadsheet with these synthesis results and adjust XPE where appropriate.
- Output RPX file:** This file saves the power report in RPX format, which can later be opened in Vivado® Integrated Design Environment (IDE) by using `open_report` command.

Run the Analysis

Once you have provided Report Power with the relevant input data, run the analysis. The tool starts annotating the netlist with activity from files and user inputs, then apply the tool defaults for the remaining undefined nodes. Next, through an iterative process, the tool propagates this initial activity from the primary inputs to the primary outputs of your design to refine the activity estimate for the undefined nodes. Finally, it calculates the dynamic power for each resource used and deduce the additional static power this switching activity generates, to compute the expected junction temperature and total power requirements for the design.

Retaining the Switching Activity Constraints

All the inputs to report_power tool are saved in the XDC constraints of the project and will be populated if report_power tool is invoked again in the flow. This is useful for the what-if analysis. The most recent switching activity constraints are retained and appear in the tool. Even if you provide inputs through XDC based commands in Tcl console of Vivado® Integrated Design Environment or through the Net Properties window (Edit Properties in Power tab), these input values will reflect in report_power tool. XDC constraints for switching activity will be in sync with the report_power tool. Any change made in the tool will reflect in the XDC constraints and vice-versa.

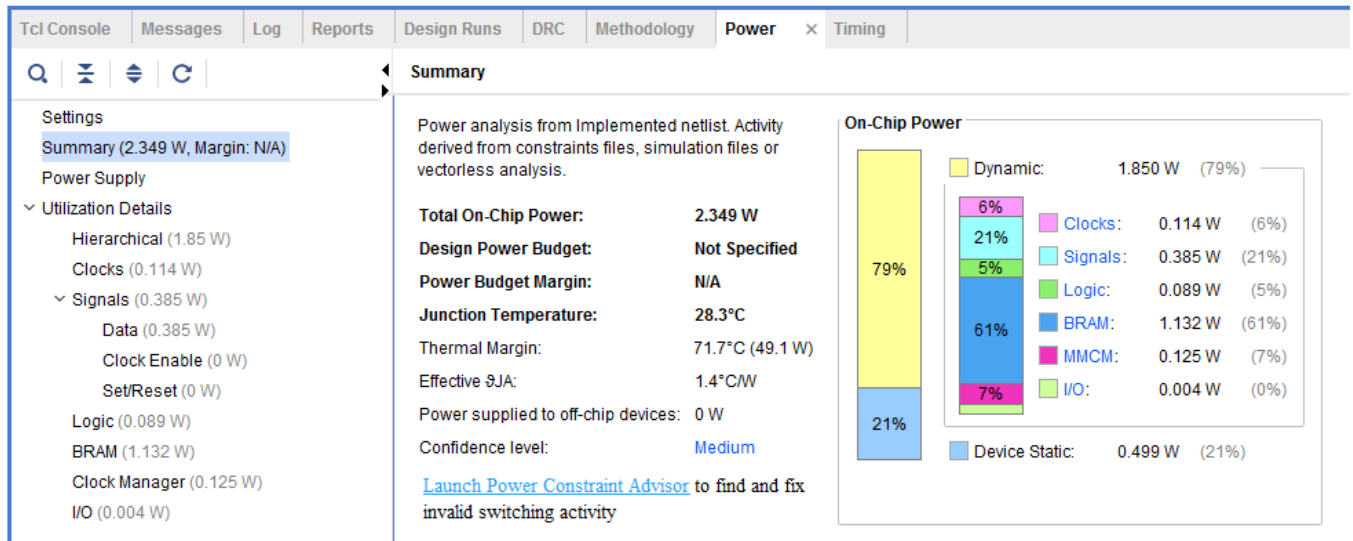
This is also helpful, if you want to override the default switching activity in the report_power tool. In this case, you can create XDC constraints with desired default values and run report_power.

Review Your Design Power Distribution

Once the power analysis is complete you can view the Summary view to review the *Total On-Chip Power* and thermal properties. The *On-Chip Power* graph shows the power dissipated in each of the device resource types. With this high-level view you can determine which parts of your design contribute most to the total power as shown in the following figure.

The Summary view also displays a *Confidence Level* for the power analysis. The *Confidence Level* is a measurement of the accuracy and the completeness of the input data Report Power uses as it performs a power analysis. If you click the Confidence level value (Low, Medium, or High), *Confidence level* details are displayed, and these details can suggest ways of increasing the accuracy of the power analysis. For example, you might increase the accuracy of the power analysis by specifying activity rates for more of the clocks or more of the I/O inputs in the design.

Figure 8: Vivado Power Analysis - Report Power in the Vivado Integrated Design Environment



The Power Supply section shows the current drawn for each supply source and breaks down this total current between static and dynamic current.

From the Utilization Details section you can get more details of the power at the resource level by clicking on the different resource types in the graph as shown in the following figure. The different resources views are organized as a tree table. You can drag a column header to reorder the column arrangement. You can also click on a column header to change the sorting order.

Figure 9: Vivado Power Analysis – Utilization Details

Utilization	Name	I/O Type	I/O Standard	Drive ...	Input Pins	Output Pins	Bidir Pins	IO LOGIC SERI
0.004 W (<1% of...)	dut_fpga							
0.004 W (<1...)	sys_clk_...	HP	DIFF_SSTL15	N/A	1	0	0	No
> 0.001 W (<1...)	fmc_out	HR	LVCMOS33	12.000	0	10	0	No
0 W	gpio_out...	HR	LVCMOS33	12.000	0	1	0	No
0 W	led	HP	LVCMOS15	12.000	0	1	0	No

If the reported power exceeds your thermal or supply budget, you can refer to [Chapter 7: Tips and Techniques for Power Reduction](#), for a list of available techniques to reduce the device power. These techniques depend on the completeness of your design and your development process's tolerance to change.

★ IMPORTANT! When *Maximum Process* is selected in the *Device* table and any power-on supply current values exceed the estimated operating current requirements, the *Power Supply* panel displays the minimum power-on supply requirements, in blue. If any of the current values appear in blue, the total power indicated in the *Power Supply* panel will not match the *Total On-Chip* power in the *Summary* section of *Vivado Power Report*.

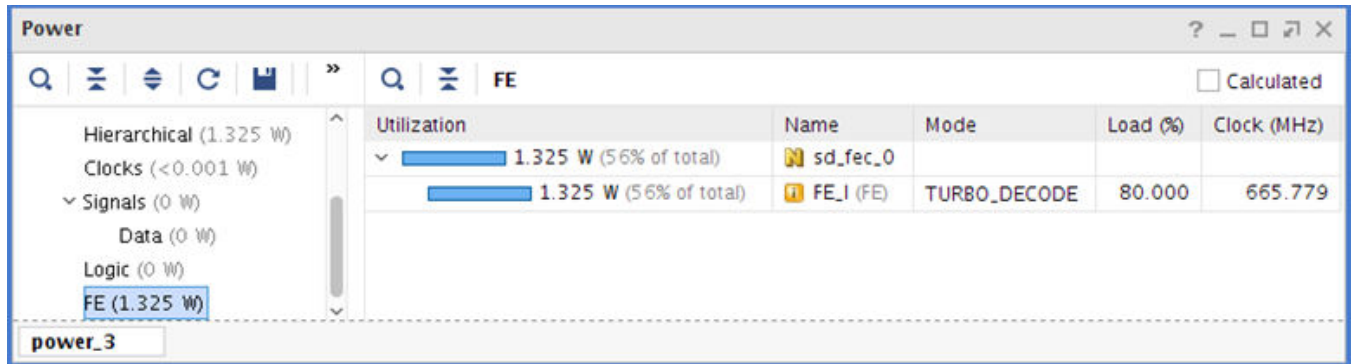
Alert for Maximum Package Current

The Total Iccint current value field in power supply section turns in to red, when estimated current exceeds the maximum specification limit of a selected package. This is applicable only for UltraScale+™ devices.

Power Estimation of SD-FEC Core

Report Power supports the power estimation of Soft-Decision FEC core available in Zynq® UltraScale+™ RFSocS. When the design containing the SD-FEC IP is implemented, Report Power displays the power estimation as shown in the following figure.

Figure 10: Report Power with SD-FEC Power Estimation



Following properties can be modified before running the Report Power for the SD-FEC object after implementation:

- LD_PERCENT_LOAD: Percentage utilization for LDPC Decoder core
- LE_PERCENT_LOAD: Percentage utilization for LDPC Encoder core
- TD_PERCENT_LOAD: Percentage utilization for Turbo Decoder core

These three properties can also be provided during SD-FEC IP customization and using `set_property` commands on an implemented design. Also, the generated `.xpe` file by Report Power command can be imported to XPE spreadsheet for further what-if analysis.

Power Estimation of RF Converter

Zynq® UltraScale+™ RFSoc device family includes RF data converter subsystem. Report Power support is available for power estimation of these cores. Cores can be generated by the RF Data Converter IP which is part of the Xilinx® IP catalog in Vivado®. This facilitates for different configurations available. Using the design implemented with these IPs, Report Power can be run to generate the power report as shown in the following figures.

Figure 11: Report Power for RFADC

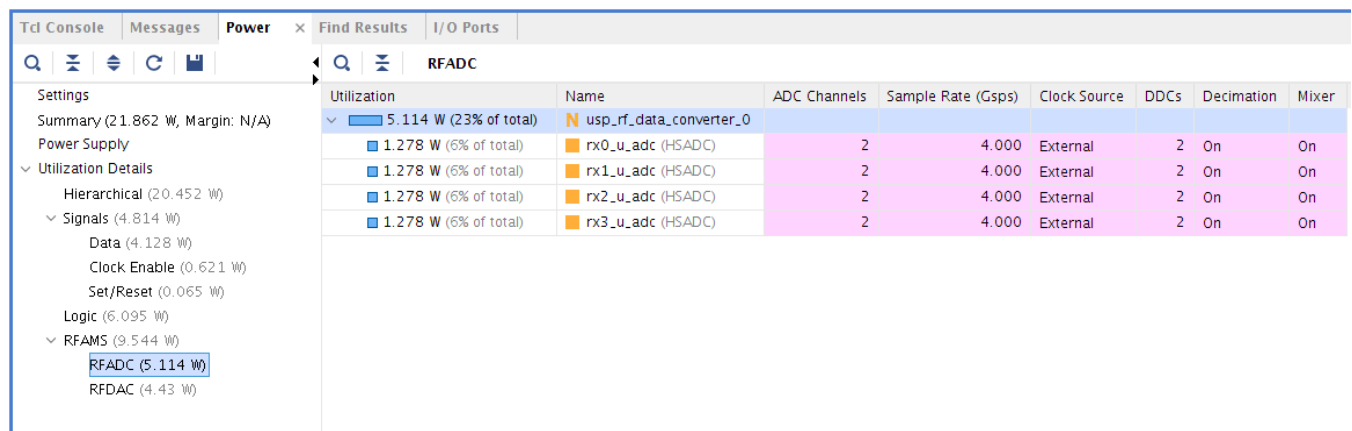


Figure 12: Report Power for RFDAC

Utilization	Name	DAC Channels	Sample Rate (Gbps)	Clock Source	DUCs	Interpolation	Mixer
4.43 W (20% of total)	usp_rf_data_converter_0						
2.215 W (10% of total)	tx0_u_dac (HSDAC)	4	6.400	External	4	On	On
2.215 W (10% of total)	tx1_u_dac (HSDAC)	4	6.400	External	4	On	On

Use the RF data converter IP customization to set all the user configuration values such as ADC/DAC channel count, sample rate, clock source, decimation, mixer etc. Also, the power data can be imported back to XPE sheet for further analysis of estimated power.

Configuring HBM for report_power

The HBM is configured using the HBM IP wizard within an IP integrator block design or the IP catalog. Set the cell properties on HBM instance to access further HBM settings for report_power. Cell properties can be used to fine-tune power analysis. The following types of cells are available in an HBM instance:

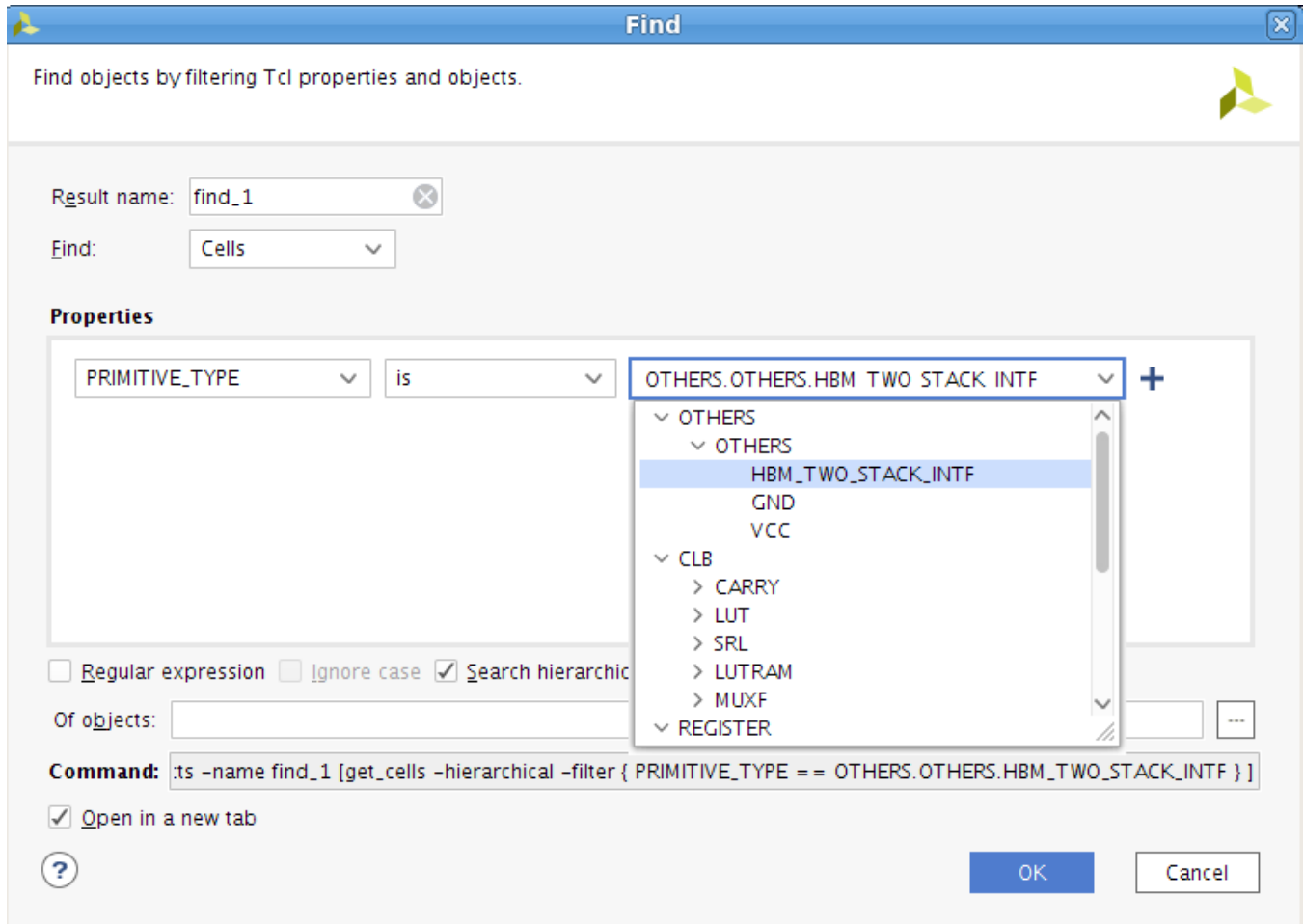
- HBM_ONE_STACK_INTF: if targeting a single HBM stack
- HBM_TWO_STACK_INTF: if targeting two HBM stacks

Use the `get_cells` command to locate the HBM instance.

```
set hbm_inst [get_cells -hier -filter {REF_NAME == HBM_TWO_STACK_INTF}]
```

You can also locate HBM instance using Find in the Vivado® Integrated Design Environment as shown in the following figure:

Figure 13: HBM Instance Using Find in Vivado



The property values can be modified before running report_power. The following properties are used for power analysis:

- PAGEHIT_PERCENT_00, PAGEHIT_PERCENT_01: The percentage of cycles that an HBM transaction accesses an open page, which results in the fastest access. For example, sequential memory accesses are more likely to occur within an open page which reduces power and increases efficiency.
- READ_PERCENT_00 to READ_PERCENT_15 (Stack 0), READ_PERCENT_16 to READ_PERCENT_31 (Stack 1): The percentage of cycles that a pseudo-channel is reading from the HBM.
- WRITE_PERCENT_00 to WRITE_PERCENT_15 (Stack 0), WRITE_PERCENT_16 to WRITE_PERCENT_31 (Stack 1): The percentage of cycles that a pseudo-channel is writing to the HBM.

Ensure reasonable values for READ_PERCENT and WRITE_PERCENT based on PAGEHIT_PERCENT. Use the following guidelines:

- PAGEHIT_PERCENT < 75%: READ_PERCENT + WRITE_PERCENT should be 50% or less
- PAGEHIT_PERCENT >= 75%: READ_PERCENT + WRITE_PERCENT should be 90% or less

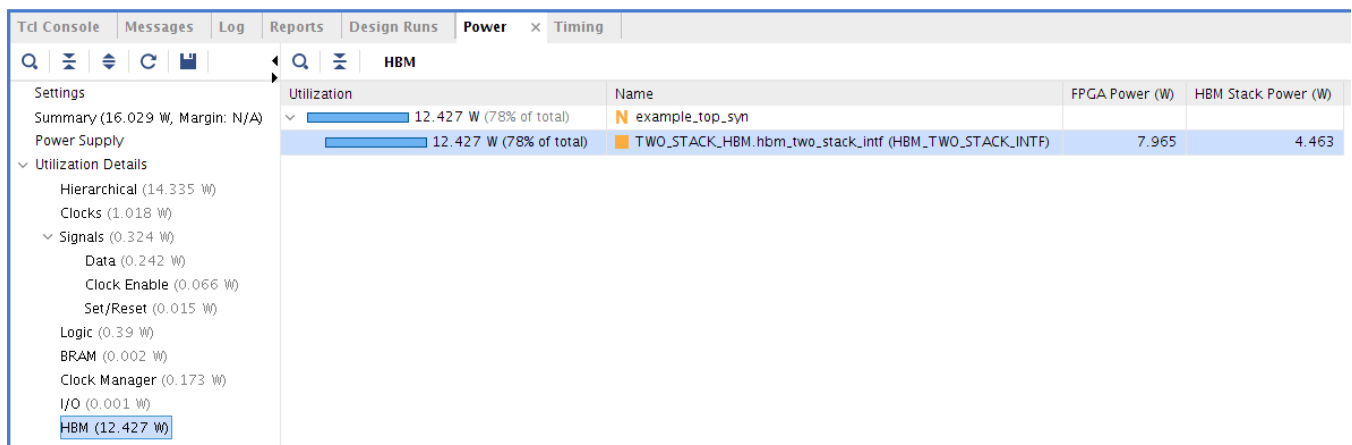
Note: In the current release, PAGEHIT_PERCENT_00 and PAGEHIT_PERCENT_01 have a default value of 50. The default value will be corrected to 75 in a future release.

The following properties are assigned by HBM IP configuration and are not modified.

- DATARATE_00 to DATARATE_15: Data rate for each memory controller in Gbps. Properties 00 to 07 apply to Stack 0 and 08 to 15 apply to Stack 1.
- SWITCH_ENABLE_00, SWITCH_ENABLE_01: Reflects whether the dedicated AXI switch is enabled or disabled for a stack.

The following figure is an example of Report Power output for HBM, showing the breakdown of power between the device and HBM stacks.

Figure 14: Report Power for HBM



Configuring GTM for report_power

GTM is configured using the GTM IP wizard within IP integrator block design of the IP catalog as shown in the following figure:

Figure 15: Configuring GTM for report_Power

Component Name

Basic | Physical_Resources | Optional_features | FEC_Options | AM_50G | AM_100G

System

GT Type

Transceiver configuration preset

Transmitter	Receiver
TX Line rate (Gb/s) <input type="text" value="53.125"/>	RX Line rate (Gb/s) <input type="text" value="53.125"/>
Transmitter PAM mode selection <input type="text" value="PAM4"/>	Receiver PAM mode selection <input type="text" value="PAM4"/>
TX User data width <input type="text" value="128"/>	RX User data width <input type="text" value="128"/>
TX Internal data width <input type="text" value="128"/>	RX Internal data width <input type="text" value="128"/>
TXOUTCLK source <input type="text" value="TXPROGDIVCLK"/>	RXOUTCLK source <input type="text" value="RXPROGDIVCLK"/>
Differential swing and emphasis mode <input type="text" value="Custom"/>	

Reference clock Frequency

Requested reference clock(MHz)

Actual Reference clock(MHz)

Resulting Fractional divider Enable

All major parameters required for Report Power estimation can be set using UNISIM properties. The UNISIM properties are as follows:

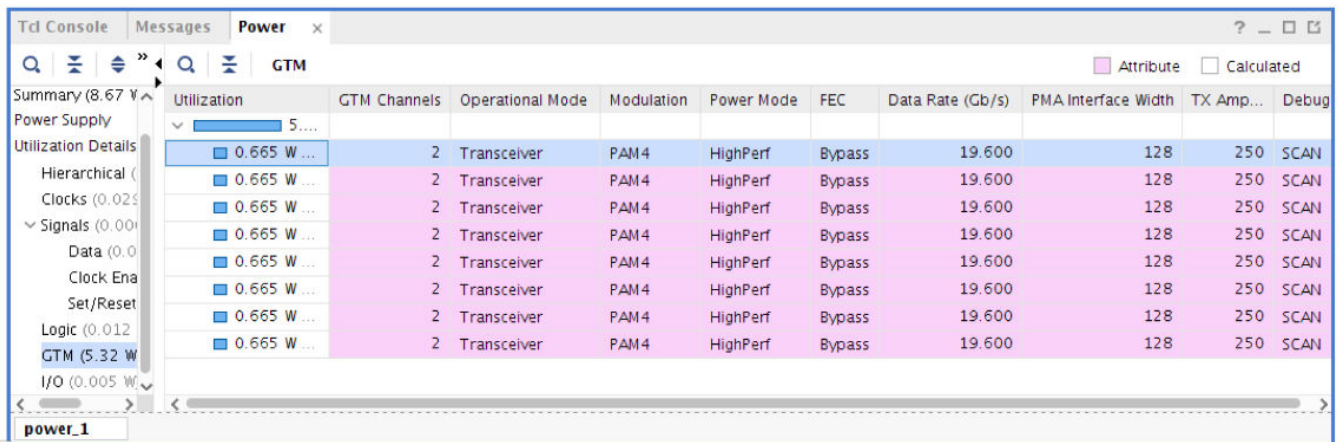
- **MODULATION_MODE:** This is the GTM signal modulation scheme and is used to select NRZ and PAM4 signaling.
- **DATARATE:** This is the GTM channel line rate for given modulation scheme. For PAM4, the values range from 19.6 Gb/s to 58 Gb/s and for NRZ GTM linerate range is from 9.8 Gb/s to 29 Gb/s.
- **FEC_MODE:** This is the hardened RS-FEC usage. If this parameter is set to BYPASS, GTM bypasses the hardened FEC block. To use FEC, set this parameter to KP4.
- **INTERFACE_WIDTH:** This is the GTM interface width and this property is added for future use. As of now, the interface width is derived from MODULATION_MODE.
- **INS_LOSS_NYQ:** This is the equalization mode. The value of this parameter should be less than or equal to 10 dB for Low Power mode and greater 10 dB for High Performance mode.
- **TX_AMPLITUDE_SWING:** This is the amplitude of TX driver's differential swing and the valid values are 250, 275, 300, ..., 1000, and 1025.

The GTM Debug mode is determined by the attribute `CH*_RX_PAD_CFG1[10]`.
`CH*_RX_PAD_CFG1[10]` is the `ACJTAG_EN` bit for each channel and is used to determine whether ACJTAG is active or not.

Note: When the `FEC_MODE` parameter is set to `KP4`, GTM cannot bypass the hardened FEC block when PAM4 signaling is used. You should ensure that the `FEC_MODE` parameter is set to `KP4` using PAM4.

The following figure is an example of report Power output for GTM.

Figure 16: Report Power Output for GTM



Utilization	GTM Channels	Operational Mode	Modulation	Power Mode	FEC	Data Rate (Gb/s)	PMA Interface Width	TX Amp...	Debug
0.665 W ...	2	Transceiver	PAM4	HighPerf	Bypass	19.600	128	250	SCAN
0.665 W ...	2	Transceiver	PAM4	HighPerf	Bypass	19.600	128	250	SCAN
0.665 W ...	2	Transceiver	PAM4	HighPerf	Bypass	19.600	128	250	SCAN
0.665 W ...	2	Transceiver	PAM4	HighPerf	Bypass	19.600	128	250	SCAN
0.665 W ...	2	Transceiver	PAM4	HighPerf	Bypass	19.600	128	250	SCAN
0.665 W ...	2	Transceiver	PAM4	HighPerf	Bypass	19.600	128	250	SCAN
0.665 W ...	2	Transceiver	PAM4	HighPerf	Bypass	19.600	128	250	SCAN

Power Analysis and Optimization in the Vivado Design Suite

Introduction

This chapter discusses the power-related features and flows available in the Vivado[®] Design Suite to get you quickly started with power estimation, analysis, and optimization. You can perform power analysis after synthesis, optimization, placement or routing. It is not supported after RTL elaboration. You can perform power optimization only before and after placement. Using either the Vivado Integrated Design Environment or the Tcl prompt, you can perform power analysis and optimization, and can experiment with *What If?* scenarios in a dynamic manner.

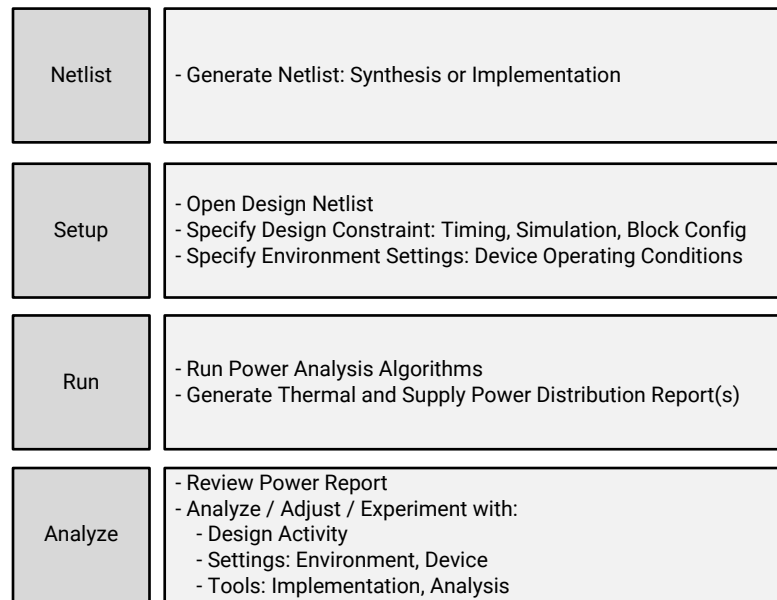
Power Analysis in the Vivado Integrated Design Environment

The Vivado[®] Integrated Design Environment power-related capabilities enable the following estimation and analysis features throughout the implementation of your design.

- Reporting the thermal characteristics that impact the static power of the design, including:
 - Thermal statistics, such as junction and ambient temperature values
 - Data on board selection, including number of board layers and board temperature
 - Data on the selection of airflow and the heat sink profile used by the design
- Reporting the device current requirements from the different power supply sources
- Allowing detailed power distribution analysis to guide power saving strategies to reduce dynamic, thermal or off-chip power

The following figure shows the typical power estimation and analysis flow. This includes the main steps required to ensure appropriate tool input and settings before running the estimation or analysis, which ensures the most accurate results. You can run power estimation and analysis commands from the Vivado Integrated Design Environment or the Tcl prompt.

Figure 17: Power Estimation and Analysis Flow



X12401-010320

Supported Device Architecture

Vivado® Design Suite architecture support is described in the *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* ([UG973](#)).

Supported Inputs

- XDC constraints file to specify timing constraints.
- Simulation output activity file results from behavioral or timing simulation results (SAIF files).
- XDC/Tcl file commands to specify environment, operating conditions, tool defaults, and individual netlist nodes activity. For UltraScale+™ devices, XPE dumps the XDC files that are sourced from Vivado® Integrated Design Environment.
- The Vivado power analysis tool has multiple mechanisms to enter default values and node activity rates. The list below presents the different mechanisms; the list is sorted from highest priority to lowest.
 1. Static (constant tied to GND or VCC).
 2. User entered value in any of the Utilization Details views in the Power Results window.

3. Imported simulation activity file (SAIF).
4. Imported constraint files – Clock constraints imported from constraint files (XDC) or the design netlist.
5. Vectorless estimation – For any node not defined in any of the previously listed inputs, the vectorless estimation will try to estimate activity based on default values combined with the activity of inputs to the node.
6. A default value – For nodes that cannot be estimated by the vectorless estimation a default is assigned, as in the case of design primary inputs and black box outputs.

Note: You can adjust default values in the Report Power dialog box. See [Review Device/Design Settings and Adjust Activity for Known Elements](#) for more information.

Supported Outputs

- GUI I/O Bus, Net, and Cell Power properties
- GUI and text power reports
- XML based power report that can be imported into the Xilinx® Power Estimator spreadsheet tool
- Reporting activity rates and operating conditions through Tcl commands

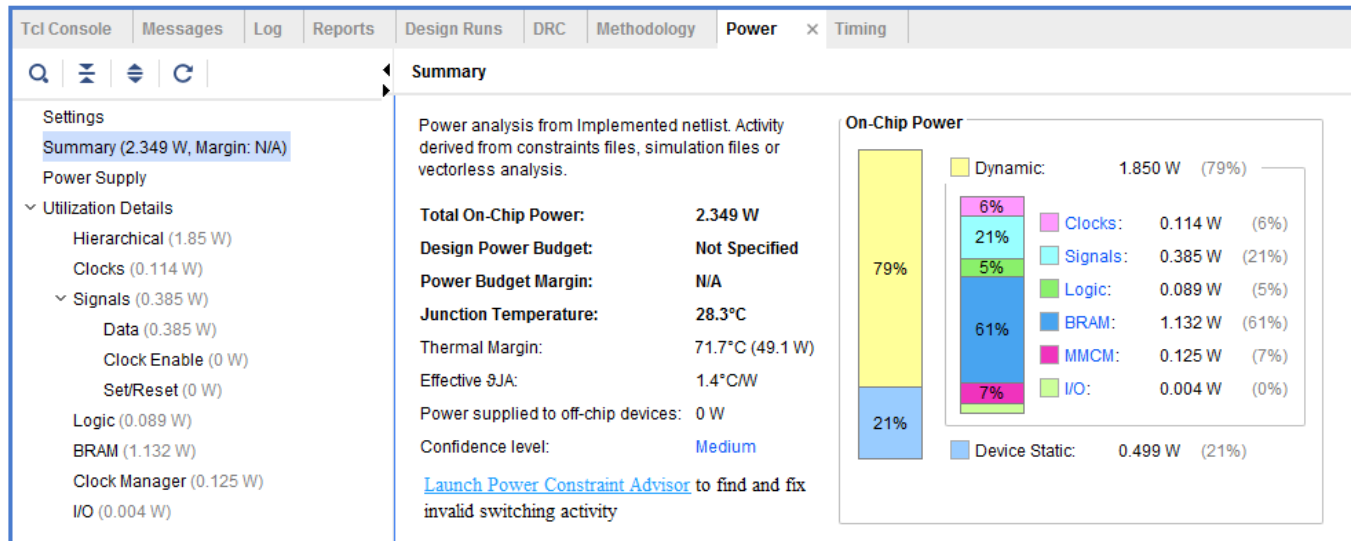
Further Refining Control Signal Activity

When SAIF-based annotation has not been used for accurate power analysis, you can fine-tune the power analysis after doing the first level analysis. Report Power extracts and lists all the different control signals in the Signal view. You may know from the expected behavior of your application that some Set/Reset signals are not active in normal design operation. In that case, you may want to adjust the activity for these signals. Similarly, some signals in your application may disable entire blocks of the design when the blocks are not in use. Adjust their activity according to the expected functionality. Because synthesis tool and place and route algorithms can infer or remap control signals to optimize your RTL description, many of the signals listed in these views may be unfamiliar. If you are unsure of what these signals are, let the tool determine the activity.

Analyzing Power Reports from the Vivado Integrated Design Environment

Power report and analysis windows are integrated into the Vivado® Integrated Design Environment workspace as shown in the following figure. These windows enable navigation across the different power views and cross probing to the existing view.

Figure 18: Power Analyzer in the Vivado Integrated Design Environment



- The Power settings section displays all the device, tool, and environment settings used with power calculations.
- The Summary section displays a concise view of the most important thermal and supply power results.

Navigate through your design by type of resources with the Utilization Details section or Netlist view to review configuration, utilization, and activity details for the selected elements in the Statistics tab of the Properties window. You can generate multiple reports to estimate power under different operating conditions or different activity patterns. Some of the values in the Utilization Details views (for example, Frequency in the Clocks view or Signal Rate in the I/O view) are color coded as shown in the following figure to indicate the source of the value used by Report Power to perform the power analysis. A legend at the bottom of the window indicates the source specified by each color (for example, the value was supplied by a Simulation activity file, or was User Defined, or a Default value was assigned by the vectorless propagation engine).

Figure 19: Color Coding in the Power Window

Utilization	Name	I/O Type	I/O St...	Drive ...	I...	Clock (MHz)	Signal Rate (Mtr/s)	D...
0.004 W (<1% o...	dut_fpga															
0.004 W (<1...	sys_clk...	HP	DIFF_...	N/A	1	0	0							200.000	400.000	Cl
> 0.001 W (<1...	fmc_out	HR	LVCM...	12.000	0		0							100.000	0.000	St
0 W	gpio_out...	HR	LVCM...	12.000	0	1	0							100.000	0.000	St
0 W	led	HP	LVCM...	12.000	0	1	0							100.000	0.000	St

★ IMPORTANT! Report Power supports Zynq®-7000 SoC and Zynq® UltraScale+™ MPSoC power analysis on Zynq-7000/Zynq® UltraScale+™ MPSoC blocks configured through the IP integrator. You configure the PS usage and functionality through the IP integrator. Report Power estimates power based on these configuration settings. The power estimate within Vivado® is read-only; you cannot edit the Signal Rate or Static Probability of the PS specific processor, interfaces or memory at this time. For more details on the individual fields in the PS tab of Xilinx® Power Estimator, refer to the PS Sheet section in the Xilinx Power Estimator User Guide (UG440).

★ IMPORTANT! Report Power supports power estimation of VCU (Video Codec Unit) for Zynq UltraScale+ EV devices. VCU is configured through the IP integrator for resolution, color format and other properties. Report Power estimates power based on these configuration settings. For more details, refer to the Other Sheet section in the Xilinx Power Estimator User Guide (UG440).

Setting Power and Current Budget for Xilinx Devices

It is a good practice to specify the power and supply current budget for your design before generating the power report. The design power budget is the power budget for the entire device while the current budget is specified per power supply or power rail. To specify supply current budget, use the commands mentioned in [Power Rail Creation and Management](#) section. To specify power budget, use the commands mentioned in the following recommendation:

✓ RECOMMENDED: To ensure design closure, correct power constraints should be applied. Following are some of the recommended constraints:

- **Minimum recommended constraints:** These constraints will ensure the power estimation checks, the power budget, and uses the worst-case maximum process for static power analysis.

```
set_operating_conditions -design_power_budget <Power in Watts>
```

```
set_operating_conditions -process maximum
```

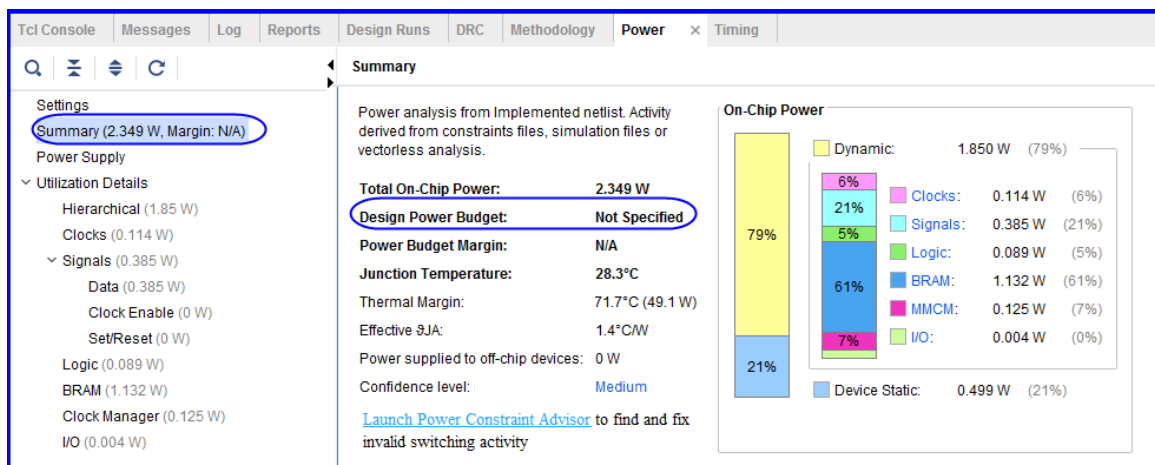
- **Additional recommended constraints:** These constraints will define the thermal solution and based on this, Report Power will estimate the Junction Temperature and therefore the static power more accurately.

```
set_operating_conditions -ambient_temp <max Ambient requested for application is Celsius>
```

```
set_operating_conditions -thetaja <the rise in junction temperature for every watt dissipated, obtained from thermal simulation, C/W>
```

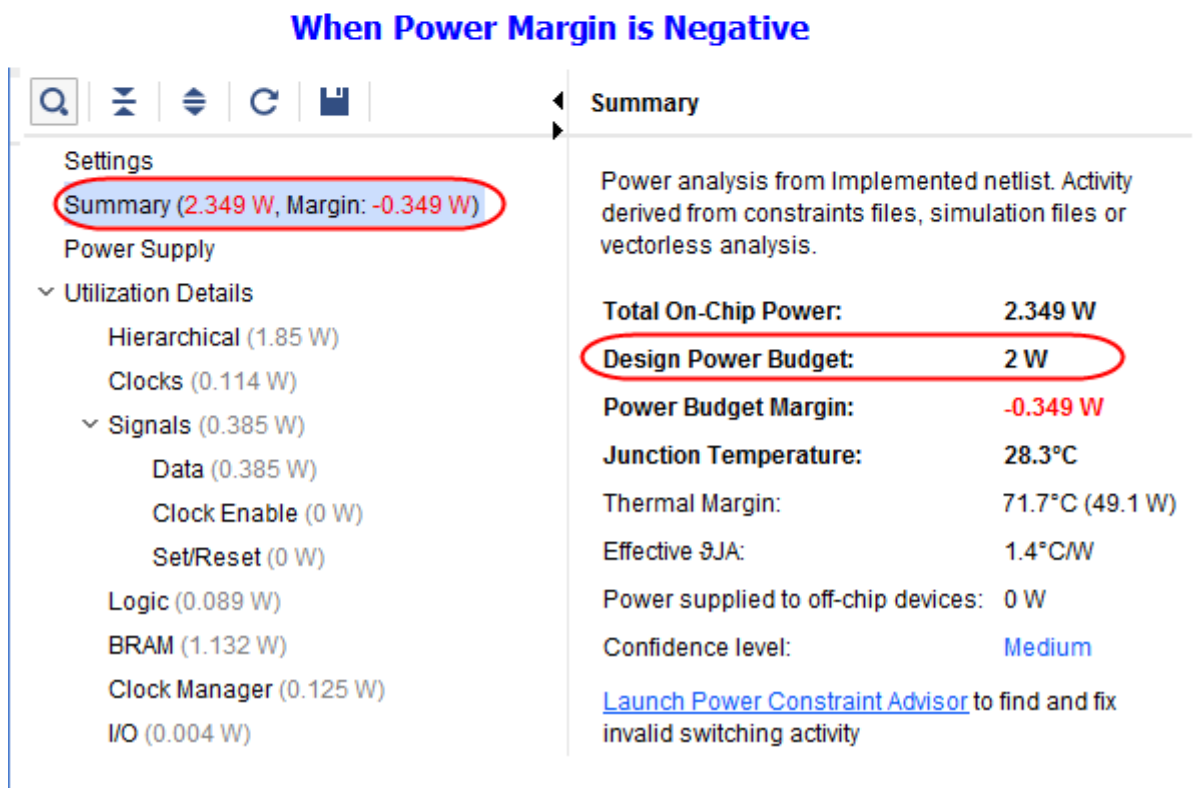
While running Report Power, the design power and the individual power supply current are compared with this power and supply current budgets. Report Power (GUI/Text) indicates the power budget margin. It displays the positive margin if the design power is less than the power budget or a red negative margin, if design power exceeds power budget. If you have not provided a power budget, the report displays N/A for the margin. The following figure shows the power report when you do not specify any design power budget:

Figure 20: Power Report Without Design Power Budget



The following figure shows the power report when the design power budget is specified as 4 Watts and power margin is positive. It also displays the power margin in a negative state when the design power budget is specified as 2 Watts.

Figure 21: Power Report With Power Budget at Positive and Negative Margins



The supply current budget for each power supply is displayed in the Power Supply section. For each supply, Report Power displays positive margin when the supply current is less than the specified budget and a negative margin appears in red when supply current exceeds the specified current budget. If the current budget is not specified for any supply, then Report Power displays the current budget as unspecified and margin as N/A for that supply rail as shown in the following figure:

Figure 22: Supply Current Budget

Supply Source	Voltage (V)	Total (A)	Dynamic (A)	Static (A)	Budget (A)	Margin (A)
Vccint	0.850	37.647	32.270	5.376	42.000	4.353
Vccint_io	0.850	0.751	0.001	0.750	0.500	-0.251
Vccbram	0.850	0.143	0.000	0.143	1.620	1.477
Vccaux	1.800	0.635	0.160	0.475	1.190	0.555
Vccaux_io	1.800	0.027	0.003	0.024	0.056	0.029
Vcco33	3.300	0.000	0.000	0.000	Unspecified	NA
Vcco25	2.500	0.000	0.000	0.000	Unspecified	NA
Vcco18	1.800	0.004	0.004	0.000	0.014	0.010
Vcco15	1.500	0.000	0.000	0.000	Unspecified	NA
Vcco135	1.350	0.000	0.000	0.000	Unspecified	NA
Vcco12	1.200	0.000	0.000	0.000	Unspecified	NA
Vcco10	1.000	0.000	0.000	0.000	Unspecified	NA
Vccadc	1.800	0.016	0.000	0.016	0.020	0.004
VCC_IO_HBM	1.200	0.164	0.000	0.164	3.840	3.676
VCC_HBM	1.200	0.188	0.000	0.188	4.160	3.972
VCCAUX_HBM	2.500	0.025	0.000	0.025	0.200	0.175
MGTYAVcc	0.900	0.000	0.000	0.000	Unspecified	NA
MGTYAVtt	1.200	0.000	0.000	0.000	Unspecified	NA

Power Rail Creation and Management

A rail or power rail refers to a group of supplies or power rails. In this section, a supply or power supply refers to an individual device power supply such as Vccint and Vccram. An auxiliary power supply can have voltage regulators that are fed by multiple sources in different phases. To handle this, user may need to add a power rail in multiple groups to account for voltage sources drawing current from different phases. Power rail definitions, the number of phases for a power rail as well as phase distribution on a rail and their current budgets are typically supplied with the target board files. Xilinx supports Tcl commands to overwrite board data or to create rail definitions, update phase information, and current budgets. Following are some useful commands:

- **create_power_rail:** Creates a new power rail.

```
create_power_rail <power rail name> -power_sources {supply1, supply2, ...}
create_power_rail <power rail name> -power_sources {supply1, supply2, ...}
-num_phases <number of phases>

create_power_rail <power rail name> -power_sources {supply1, supply2, ...}
-phased_power_source {<power rail name with phase info> <num_phase>}
```

- **delete_power_rail:** Deletes an existing power rail.

```
delete_power_rail <power rail name>
```

- **add_to_power_rail:** Add power sources to an existing power rail.

```
add_to_power_rail <power rail name> -power_sources {supply1, supply2, ...}
add_to_power_rail <power rail name> -power_sources {supply1, supply2, ...}
-phased_power_source { <power_rail_with_phase_info> <num_phases> }
```

- **remove_from_power_rail:** Removes power sources from a power rail.

```
remove_from_power_rail <power rail name> -power_sources {supply1,
supply2, ...}
```

When defined, the supply current budget can be specified for the Power Rails along with the Rail voltage. The power budget of the Rail is the supply current budget multiplied by the Rail voltage. Use the following command to specify the power and supply current budget:

```
set_operating_conditions -supply_current_budget {<supply rail name>
<current budget in Amp>} -voltage {<supply rail name> <voltage>}
```



TIP: Power rail reporting is not fully supported in the Report Power GUI, to view the complete results, ensure to generate a `.txt` output file.

Report Power compares the current on each Power Rail against their current budget and indicates positive and negative margin in the power text report as discussed in the previous section. The following figure shows a power text report with Power Rails and their budgets.

Figure 23: Power Text Report

Source	Voltage (V)	Total (A)	Dynamic (A)	Static (A)	Powerup (A)	Budget (A)	Margin (A)
12V_PEX	12.000	1.218	0.841	0.377	NA	5.500	4.282 (MET)
BOARD_VCCINT	0.850	9.104	5.708	3.396	NA	60.000	50.896 (MET)
Vccint	0.850	9.104	5.708	3.396	NA	60.000	50.896 (MET)
VCCINT_BRAM	0.850	1.121	0.507	0.613	NA	60.000	58.879 (MET)
Vccint_io	0.850	0.951	0.429	0.522	NA	15.000	14.049 (MET)
Vccbram	0.850	0.170	0.078	0.091	NA	45.000	44.830 (MET)
MGT0V9AVCC	0.900	0.804	0.680	0.124	NA	4.000	3.196 (MET)
MGTAVVcc	0.900	0.804	0.680	0.124	NA	4.000	3.196 (MET)
MGTAVTT	1.200	2.842	2.804	0.038	NA	4.000	1.158 (MET)
MGTAVTt	1.200	2.842	2.804	0.038	NA	4.000	1.158 (MET)
VCC1V8	1.800	0.995	0.464	0.531	NA	4.000	3.005 (MET)
Vccaux	1.800	0.840	0.353	0.487	NA	1.000	0.160 (MET)
Vccaux_io	1.800	0.035	0.011	0.024	NA	1.000	0.965 (MET)
MGTVccaux	1.800	0.103	0.100	0.004	NA	1.000	0.897 (MET)
Vcco18	1.800	0.000	0.000	0.000	NA	0.500	0.500 (MET)
Vccadc	1.800	0.016	0.000	0.016	NA	0.500	0.484 (MET)
3V3_PEX	3.300	0.332	0.213	0.119	NA	3.000	2.668 (MET)
2V5_VPP	2.500	0.030	0.006	0.024	NA	0.250	0.220 (MET)
VCCAUX_HBM	2.500	0.030	0.006	0.024	NA	Unspecified	NA
1V2_HBM	1.200	0.852	0.574	0.278	NA	10.000	9.148 (MET)
VCC_HBM	1.200	0.447	0.297	0.150	NA	Unspecified	NA
VCC_IO_HBM	1.200	0.405	0.277	0.127	NA	Unspecified	NA
Vcco33	3.300	0.000	0.000	0.000	NA	Unspecified	NA
Vcco25	2.500	0.000	0.000	0.000	NA	Unspecified	NA
Vcco15	1.500	0.000	0.000	0.000	NA	Unspecified	NA
Vcco135	1.350	0.000	0.000	0.000	NA	Unspecified	NA
Vcco12	1.200	0.000	0.000	0.000	NA	Unspecified	NA
Vcco10	1.000	0.000	0.000	0.000	NA	Unspecified	NA

Note: As of 2020.2 Vivado release, only U50 board files support Power meta-data such as Power Rail definition and their current budgets. Additional boards will support power rails in future releases.

Save and Restore Power Reports

Save and restore power reports is a new feature introduced in Vivado®. This feature allows you to save the power reports from Vivado Integrated Design Environment and then reopen them when required. The report will be saved in the rpx format and can be opened at any time using the following Vivado Tcl command:

```
open_report
```

When you run and open implemented design in the project mode, you see that the power report `impl_1` opens up by default like a timing report. In the checkpoint flow, you can save the report using `-rpx` option with `report_power` tcl command:

```
report_power -rpx design_1_power.rpx
```

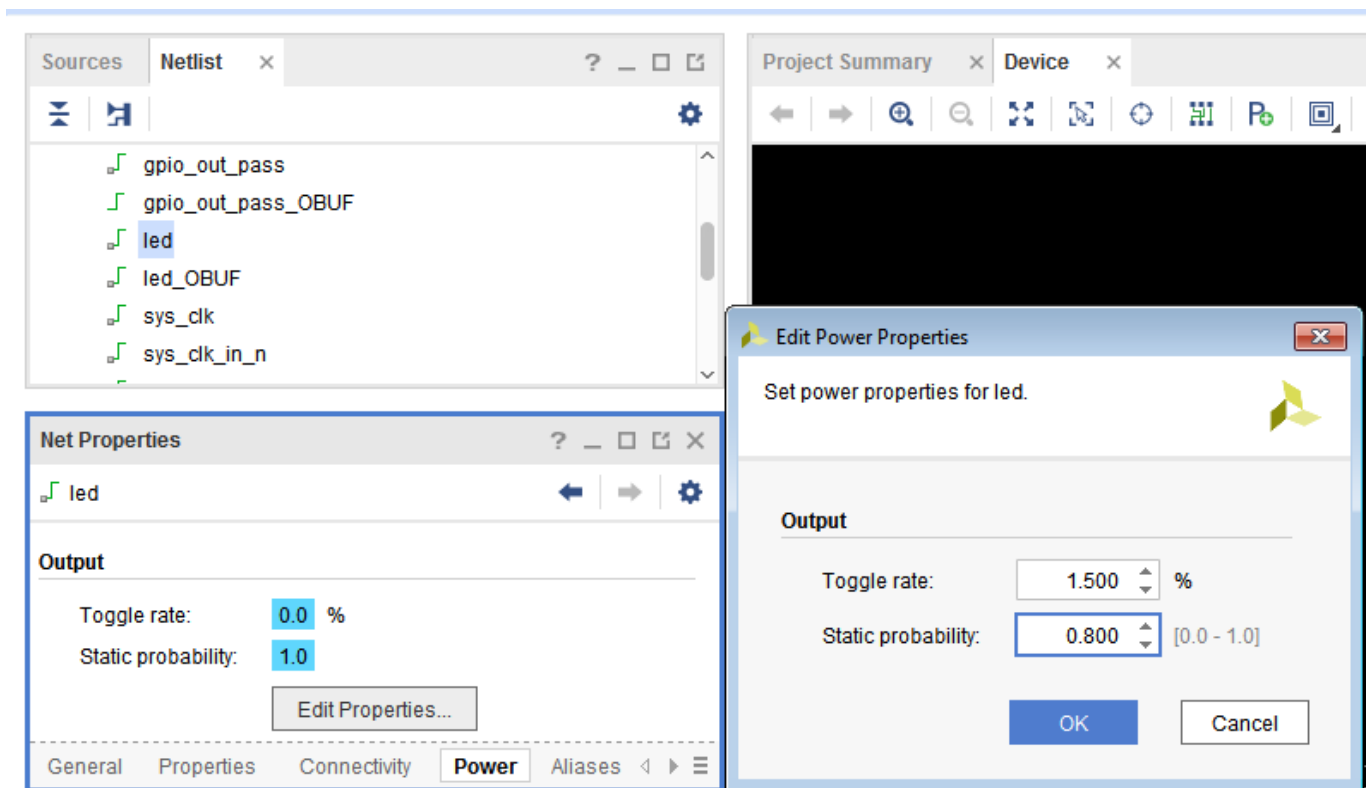
This saved report can be restored in Vivado Integrated Design Environment using the following Tcl command:

```
open_report -name rpx1 ./design_1_power.rpx
```


Performing What If? Analysis in the Vivado Integrated Design Environment

To perform What If? analysis, you can set toggle rates and static probability on nets and cells in the design. To make these settings, right-click any net or cell from the Netlist window or Power Report and click **Properties**, and go to the Power view in the Properties window as shown in the following figure. Then click the **Load Properties** button in the Power view. Click **Edit Properties** and set the Toggle Rate and Static Probability in the Edit Power Properties dialog box that appears, and click **OK**.

Figure 24: Power View in Net Properties Window



In the example above the Toggle Rate has been set to 1.5% and Static Probability is set to 0.8. On the Tcl console the following XDC constraint will be displayed when the Vivado Integrated Design Environment commits the change on OK.

```
set_switching_activity -toggle_rate 1.500000 -static_probability 0.800000
[get_nets led]
```



IMPORTANT! This XDC constraint makes your design out of date so use Force up-to-date to restore the design status.

Power Constraints Advisor

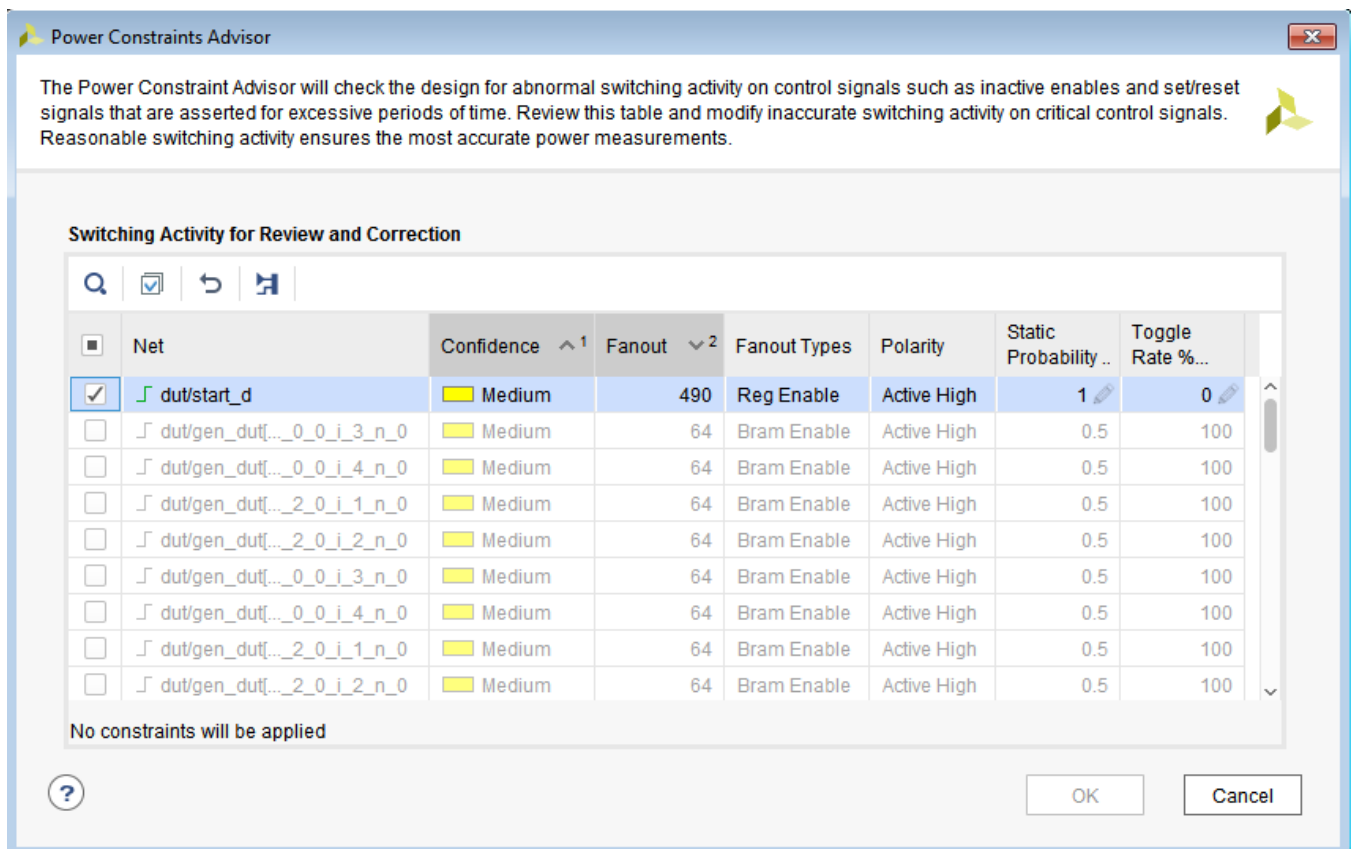
Power Constraints Advisor reports the tool-computed switching activity on all control signals in the design. Control signals include resets and enables such as Reset, Set, Clear, and Preset. Providing reasonable switching activity ensures the most accurate power estimation.

In the Vivado® IDE, select **Tools** → **Power Constraints Advisor** to run the Power Constraints Advisor.



TIP: Power constraints advisor also allows you to validate the state of enables and resets to ensure worst case power analysis. You should ensure that the highest fanout nets are correctly defined.

Figure 25: Power Constraints Advisor



Review the report table and modify inaccurate switching activity on critical control signals such as inactive enables and reset signals that are asserted for excessive periods of time. The following constraints are available in the Power Constraints Advisor report:

- **Net:** The nets are control sets, block RAM enables or Reg Enables.
- **Confidence:** This field shows how accurate the switching activity is for a particular net. Following are the thresholds used by the power tools when computing the confidence level for nets:

- **Set / Reset / Preset / Clear:**

Table 1: Power tool Thresholds

Confidence	Static Probability
Low	> 8%
Medium	Between 5% -8%
High	< 5%

- **Block RAM Enables:**

Table 2: Block RAM Enable

Confidence	Static Probability
Low	< 1%

Low confidence means that the block RAM is not active in the design and should be revisited to check the possibility of removing it.

- **Reg Enables:**

Table 3: Reg Enable

Confidence	Static Probability
Low	< 3%
Medium	> 3%

Low Confidence informs you that the Register in the design is not active and should be revisited. Medium Confidence informs you that the registers are enabled with reasonable amount of time either defined by you or propagated by tool.

- **Fanout:** This field shows the fanout for each control signal, which is the number of driven leaf-level primitives. Signals with higher fanout are the most important for review and correction because they are capable of disabling downstream switching of large portions of the design. This may result in severe under-reporting of power. Low-fanout signals with inaccurate switching will have less impact and are therefore not important.
- **Fanout Types:** This field specifies if the nets are control sets (set, reset, clear, preset) or block RAM enable. If there are multiple entries for any control net, it means that those particular nets have multiple fanouts and they are driving different pins in fanout cells.
- **Polarity:** This field identifies the polarity for the control set. You should pay attention to the polarity while setting the static probability of a net.
- **Static Probability:** This is editable field and you need to enter the correct activity based on the fanout type and polarity of the net.

- **Toggle Rate:** Toggle rate for the net. This is also editable and you need to enter this field based on the static probability.

Note: By default, PCA will be sorted by Confidence as Low and Fanout as high to low. Also, the column filtering is enabled for PCA wizard. To use column filtering, right-click on header row and click **Enable Column Filtering**.

The following process is recommended for using the Power Constraints Advisor:

1. Click the **Confidence** column to sort it so that LOW signals are in top.
2. Hold down the Ctrl key and click the **Fanout** column twice to sort it by descending values.
3. Review and define new **Static Probability** and **Toggle Rate** for all the control nets which are LOW in confidence with fanout greater than 200.
4. Click **OK** to apply the constraints to the design and rerun the Report Power command.

The following are some of the examples which help you to set accurate switching activity for the control sets and block RAM enables:

Active high reset with Static Probability 0.9

This indicates that the reset is high (active) 90% of the time. This means that the load cells are reset for 90% of the time, which is excessive. Change the switching activity to indicate that the reset is inactive, a more realistic condition, by setting the Static Probability to 0 and Toggle Rate to 0.

Block RAM Enable with Static Probability 0 and Toggle Rate of 0

This indicates that the block RAM is never enabled, which is overly pessimistic. Assign a more reasonable switching activity on the block RAM Enable such as a 25% enable rate, setting the Static Probability to 0.25 and Toggle Rate to 50. Use the following command to generate the text report for the power constraints advisor:

```
report_power -advisory -file power_report.pwr
```

Advisory table is added at the end of the this report file.

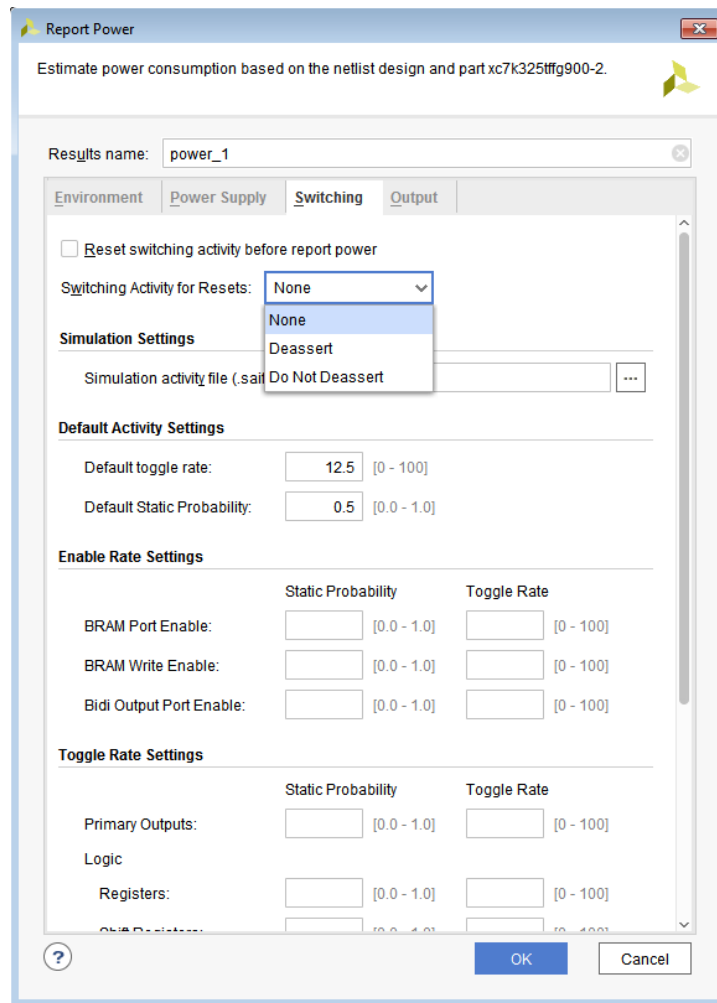
Deassertion of Resets

You can deassert all the resets in the design. This option allows you to match the power number from tools much closely with the hardware number. You will have the option to enable/disable this option from the report power tool or in Tcl mode. There are three options for setting the Switching Activity for Resets:

- **None:** This is the default mode. In this mode, the report power tool will not set any value and leave the activities as comes after vector-less propagation.

- **Deassert:** When you select this option, the report power tool will deassert all the resets in the design.
- **Do Not Deassert:** In this mode, changes of deassert option will be reverted back to original value.

Figure 26: Deassertion of Resets



The following Tcl options are introduced in `set_switching_activity` and `reset_switching_activity` commands:

```
set_switching_activity -deassert_resets
```

This is equivalent to Deassert option for Switching Activity for Resets.

```
reset_switching_activity -no_deassert_resets
```

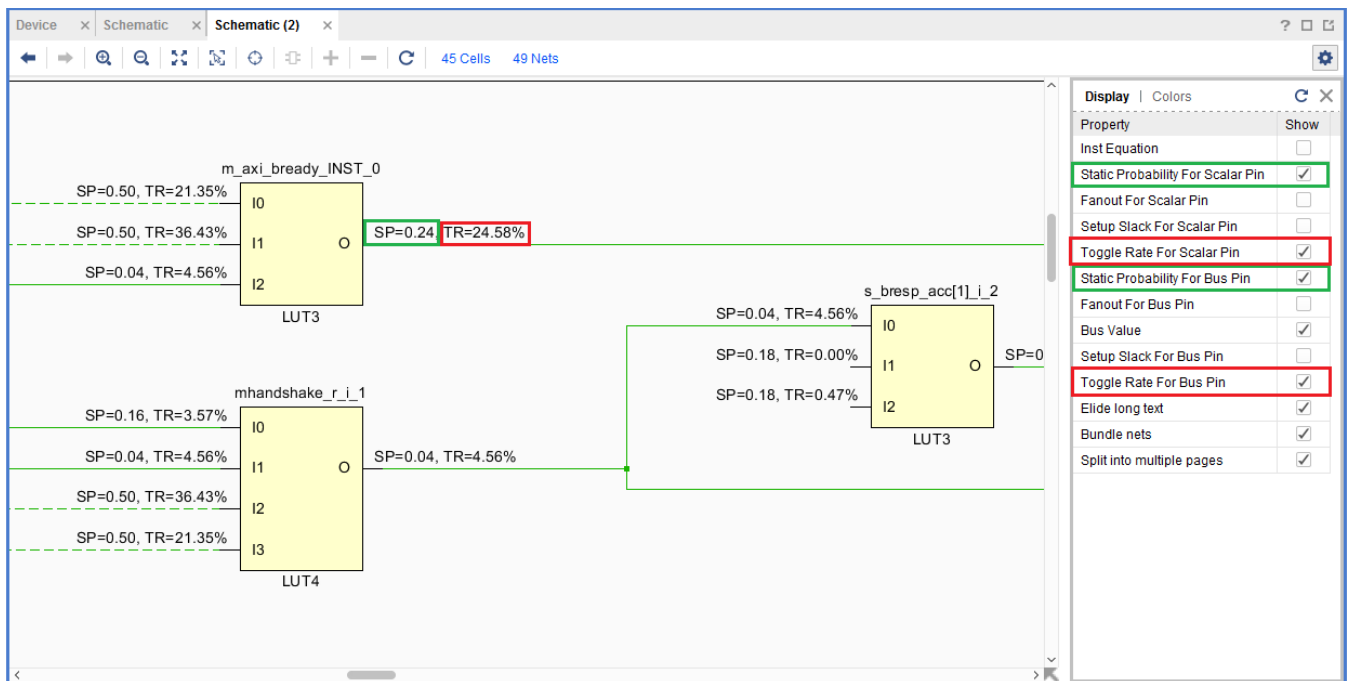
This is equivalent to Do Not Deassert option for Switching Activity for Resets. The Deassert option will not be set in the following exceptional conditions:

- If a reset net is connected to pins of different polarity. For example, if a reset net is connected to both the active-High reset pin and active-Low reset pin, then the command would not try to set value on this net.
- If a net connected to active-High reset pin is also connected to an active-High enable pin at the same time, then this command does not do anything.
- Nets connected with synchronizer circuits which provide an asynchronous clear and synchronous deassert functionality to avoid meta-stability issue crossing different clock domains.

Viewing Switching Activity on Schematics

This feature allows you to observe the Static Probability and Toggle Rate information for any particular nets in the schematic.

Figure 27: Switching Activity on Schematics



To enable the switching activity reporting on schematics, click on the setting icon at the top right hand corner on schematic view and select the SP/TR for scalar or bus pins.

Power Analysis Using the Tcl Interface

This section describes each step in a typical power analysis flow using the Tcl interface. The following section, [Power Analysis Tcl Commands](#), lists the commands related to power analysis. For information on options, properties, applicable elements, or returned values for a specific command:

- Type `<command_name> -help`, or
- See, *Vivado Design Suite Tcl Command Reference Guide* ([UG835](#)).
- See, *Vivado Design Suite User Guide: Using Constraints* ([UG903](#)).

Power Analysis Tcl Commands

- `read_saif`
- `set_switching_activity`
- `set_operating_conditions`
- `report_switching_activity`
- `report_operating_conditions`
- `report_power`
- `reset_switching_activity`
- `reset_operating_conditions`
- `set_units`

Timing Constraints that Influence Power Analysis

- `create_clock`
- `create_generated_clock`
- `set_input_delay`
- `set_case_analysis`

Setting Up Power Analysis from the Tcl Prompt

Before running power estimations, you must provide the tool with information about the device environment and the known switching activity rates for the design netlist. This ensures the accuracy of the power estimation.

- [Device Environment](#)
- [Netlist Element Activity](#)
- [set_case_analysis](#)

Device Environment

Specify all device operating conditions settings such as:

- Thermal, for example:
 - Ambient temperature
 - Heat sink
- Voltage, for example:
 - VCCINT
 - VCCAUX
 - VCCO
- Device, for example:
 - Temperature grade
 - Process corner

Use the following commands:

- `report_operating_conditions`

Report all or the specified operating condition settings. Examples are:

```
report_operating_conditions    # Reports all
report_operating_conditions -voltage
```

- `set_operating_conditions`

Modify the specified operating condition parameters. Examples are:

```
set_operating_conditions -process maximum -junction_temperature 50
set_operating_conditions -voltage {vccint 0.97 vccaux 1.71}
```

- `reset_operating_conditions`

Return all or the specified operating condition parameters to the default values for the selected device. Examples are:

```
reset_operating_conditions    # Resets all
reset_operating_conditions -voltage
```

Netlist Element Activity

Use the following commands to define the switching activity, including signal or toggle rate and static probability, and the clock waveform information for known netlist elements.

- `set_switching_activity`


Set the activity of the specified elements. You can set either static probability and signal rate or static probability and toggle rate. Examples are:

- To set default switching activity on primary ports and black box outputs of the entire design:

```
set_switching_activity -default_static_probability 0.5 -
default_toggle_rate 12.5
```

- To set the signal rate on a port/net/pin:

```
set_switching_activity -static_probability 0.5 -signal_rate 50
[get_ports din*]
```

 **IMPORTANT!** Signal rate must be > 0 when static probability is > 0 and < 1. Similarly, static probability must be 0 or 1 when signal rate is 0. Static probability and signal rate must be specified together.

- To set toggle rate on port/net/pin:

```
set_switching_activity -static_probability 0.5 -toggle_rate 25 [get_nets
din_int*]
```

Note that the toggle rate is specific to the clock associated with the element and the valid range is 0 to 100.

- Setting Switching activity on a group of nodes:

The `set_switching_activity` command can also be used to set activity rates on a group of nodes (called types), using the `-type` option. The supported types are listed in the following table:

Table 4: Types (-type option) in Switching Activity Tcl Commands

Type Name	Switching Activity Applied To	PIN Name(s)	Cell Name(s)
bram_enable	Enable pins of block RAM	ENARDEN/ENBWREN	RAMB36/18
bram	All the active data outputs of block RAM	DOADO/DOBDO	RAMB36/18
bram_wr_enable	Write enable pins of block RAM	WEA/WEBWE	RAMB36/18
register	Output pin of FF/Latch	Q	FD*
shift_register	Output pin of Shift-Registers	Q	SRL*
lut_ram	Output data pin of RAM	O	RAM(32 64 128 256)*
lut	Output pin	O	LUT*
dsp	All the data outputs of DSPs	P/ACOUT/BCOUT/PCOUT	DSP48
gt_txdata	TX data in port	TXDATA	GT*_CHANNEL
gt_rxdata	RX data out port	RXDATA	GT*_CHANNEL
io_output	Primary outputs	get_ports -filter {DIRECTION = OUT} && 'I' pin of OBUF* & IOBUF*	OBUF*
io_bidir_enable	Enable pin of Bidir ports	T	OBUF*

The following section describes usage in the `set_switching_activity` command. To set the specified switching activity on all LUTs in the design top scope:

```
set_switching_activity -type lut -static_probability 0.5 -toggle_rate 25
[get_cells]
```

To set the specified toggle rate and static probability on all registers in the hierarchy of CPU/MEM:

```
set_switching_activity -type register -toggle_rate 0.4 -static_probability
0.5 [get_cells CPU/MEM]
```

To set the specified toggle rate and static probability on all registers in the hierarchy of CPU/ and the hierarchy underneath:

```
set_switching_activity -type register -toggle_rate 0.4 -static_probability
0.5 -hier [get_cells CPU]
```

To Set the specified switching activity on all primary outputs

```
set_switching_activity -type io_output -static_probability 0.5 -toggle_rate
0.4 -all
```

★ **IMPORTANT!** Ideally, toggle rate should not include glitch rate in it, which implies that the following condition must be satisfied, $(toggle_rate/200) \leq static_probability \leq 1 - (toggle_rate/200)$. Use the signal rate setting for considering glitch switching, along with actual activity rate.

★ **IMPORTANT!** The `set_switching_activity` command will not have any effect on design clock nets. To change the activity on the clock nets, please use timing constraints (`create_clock`, `create_generated_clock`, `set_case_analysis`, etc).

- `report_switching_activity`

Reports the activity of the specified elements. Displays static probability, signal rate and toggle rate. The command also displays the source of the assigned switching activity.

Examples of `report_switching_activity` commands are:

- Report static probability, signal rate, and toggle rate for a single net:

```
Vivado% report_switching_activity -static_probability [get_ports clk_p]
clk_p: static probability = 0.5 (C)

Vivado% report_switching_activity [get_ports clk_p]
clk_p: static probability = 0.5 (C) signal rate = 400 (C) toggle rate
= 200 (C)
```

The source of the assigned switching activity is expressed as: (C)=XDC Constraints, (D)=Tool Default, (S)= SAIF Annotated, (A)=User Assigned.

- Report on group nodes:

To report switching activity for all distributed RAMs in the hierarchy CPU/:

```
report_switching_activity -type lut_ram [get_cells CPU/*]
```

To report switching activity for all GT RXDATA in the design:

```
report_switching_activity -type gt_rxdata -all
```

See [Table 4: Types \(-type option\) in Switching Activity Tcl Commands](#) table for information on the supported types.

- Report average switching activity over a list of nets:
 - **Switching activity of a bus:** `report_switching_activity -average [get_nets bussed_signal[*]]`
 - **Switching activity of an instance:** `report_switching_activity -average [get_nets -top -hier -filter {TYPE==SIGNAL && NAME =~ "Instance_Name"}]`
 - **Switching activity of whole design:** `report_switching_activity -average [get_nets -top -hier -filter {TYPE==SIGNAL}]`

The `-average` option can calculate typical switching activity of existing designs, and that activity can be applied in XPE to estimate power of similar designs.

- `reset_switching_activity`

Resets the activity rates (static probability, signal rate, and toggle rate) on specific netlist elements to the tool default value. The command resets both user specified values and Simulation activity rate settings. Examples are:

- To reset default switching activity on primary ports and black box outputs of the entire design:

```
reset_switching_activity -default
```

- To reset activity rates on the entire design:

```
reset_switching_activity -all
```

- To reset activity rate on specific port/net/pin:

```
reset_switching_activity [get_ports din*]
```

- To reset activity rates on a group of nodes:

To reset the switching activity for all block RAM enables (ENARDEN/ENBWREN) in the entire design:

```
reset_switching_activity -type bram_enable -all
```

To reset the switching activity for all LUTs in the hierarchy CPU/ and levels underneath:

```
reset_switching_activity -type lut -hier [get_cells CPU/MEM]
```

See [Table 4: Types \(-type option\) in Switching Activity Tcl Commands](#) table for information on the supported types.

- `read_saif`

Read an SAIF simulation output file and annotate matched netlist elements with the switching activity described in the file. An examples is:

```
read_saif -out_file read_saif.rpt -strip_path tb/tb_core/core -file
routed.saif
```

Options to `read_saif` are:

- `out_file`: Dumps the unmatched simulation and design nets list into a file.
- `strip_path`: By default it is assumed that the design top is instantiated in the test bench. Thus the first two levels of hierarchy are stripped while annotating SAIF data into the design. If the simulation setup has multiple hierarchy levels, then you are expected to specify the hierarchy to be stripped off from SAIF to better match the actual design.

The `read_saif` command also displays the SAIF annotation summary to show the number of design nets matched. Ideally 100% design net match is expected for an accurate analysis.



IMPORTANT! *If your design contains any encrypted IP/Blocks, your simulator will not dump the SAIF information for those IP/Blocks and for any internal blocks within the encrypted hierarchy. This incomplete SAIF information might affect the power estimation accuracy. The `read_saif` command will not modify the activities on the design clock nets. Clock nets activities will be driven by the timing constraints.*

`read_saif` command can be executed multiple times with each SAIF file. This will enable you to read multiple SAIF files for different blocks in design. Report power then estimates the power by considering the switching activity information from all the SAIF files. If common nets exist in multiple SAIF files, then the switching activity will be applied from the last read SAIF file using `read_saif` command.

- `create_clock`

Synthesis and implementation constraint to specify clock waveforms. An example is:

```
create_clock -name clk -period 5 [get_ports clk]; # 200MHz
```

- `create_generated_clock`

Synthesis and implementation constraint to specify generated clock waveforms. An example is:

```
create_generated_clock -name gen_clk -source clk1 -divide_by 2 [get_net -
hier sys_clk]
```

- `set_input_delay`

Associates primary inputs to the specific clock. This is very important in a multi-clock design, especially if the primary port is launched at a different clock. An example is:

```
create_clock -name clk1 -period 5 [get_ports clk]
set_input_delay -clock clk1 1 [get_ports d]
```

Note: If the primary ports are not associated with any clock, then the switching rate is computed based on the capturing clock in the path.

By default, `create_clock` and `create_generated_clock` are defined in the XDC file and you need not rerun them. However, to do What If? analysis, such as by changing the clock frequency for Report Power, `create_clock` or `create_generated_clock` must be used to reflect the change.

set_case_analysis

For global clock primitives (BUFG, BUFGCE, BUFGCE_DIV, BUFG_GT, BUFGCTRL), the enable/selection of clock is determined by `set_case_analysis` command. This command guides the timing analyzer to identify the clocks across clocking logic. For example the select signal of BUFGMUX must be set using `set_case_analysis` to guide the timing analyzer's clock selection. This in turn helps Report Power to estimate power using the right clock. For BUFGCE block, CE input must be set using `set_case_analysis` to enable or disable the clock output.

Minimum Input Set

Before performing power estimation:

- Make sure the activity is defined for all clocks in your netlist.
- If possible, specify the activity of all primary input ports in your design using the Tcl commands or reading a simulation output file. These port activity rates determine the internal logic activity rates. Therefore, if the tool's default settings do not match your application, the internal logic activity may be overestimated or underestimated.
- If known, specify the activity of any high fanout nets that you defined in your HDL code, such as global set, reset, and clock enable signals.

When reading the simulation result file, make sure the activity is representative of the worst case design functional activity (that is, the simulation result at which the maximum design code coverage is achieved). Using simulation results from basic and corner case tests can lead to inaccurate power estimations.

Running Power Analysis from the Tcl Prompt

After all environment and activity settings are defined, you can run the power analysis algorithm using the `report_power` command. An example is:

```
Vivado% report_power -file routed.pwr -xpe design_top.xpe
```

The tool does the following:

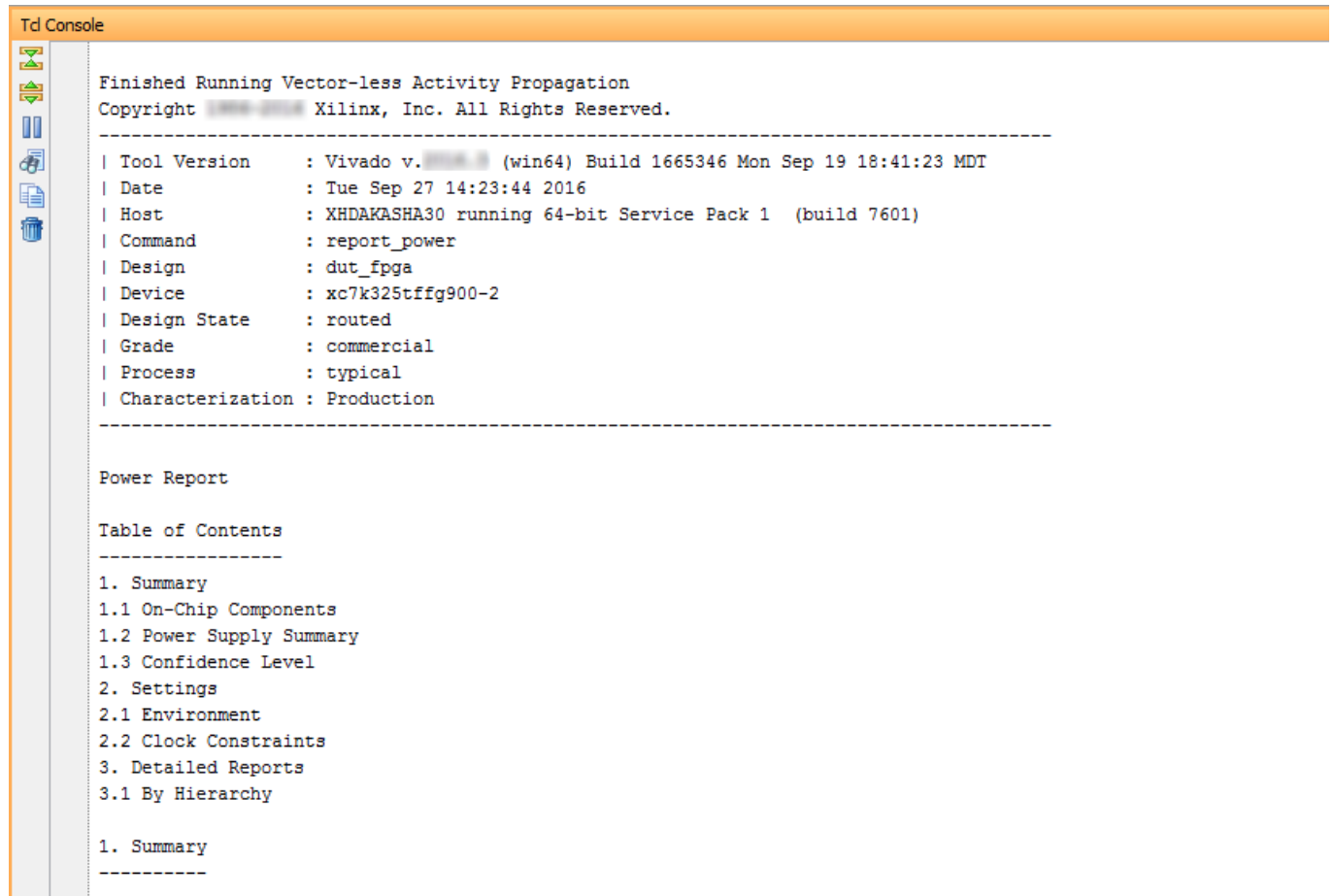
1. Loads your environment settings and design netlist.
2. Annotates activity for any netlist element you specified with input files or Tcl commands.
Note: For all undefined nodes, the tool uses the vectorless propagation engine to estimate activity, taking into account activity of known elements and logic configuration and connectivity.
3. Calculates and reports the design thermal and supply power.

Analyzing Power Reports from the Tcl Prompt

To analyze the design power, start by reviewing the total thermal and supply power information in the power report as shown in the following figure. Then, depending on your design margin against requirements, you can review the resource or hierarchy sections. These sections show the design power distribution at a more detailed level. As a result of your analysis, you may want to return to Xilinx® Power Estimator and perform design architectural scenarios. You can perform What If? scenarios to evaluate the impact of changes in the settings for:

- Environment
- Device
- Implementation
- Power tool

Figure 28: Text Report Generated for Power and Thermal Information



```

Tcd Console
Finished Running Vector-less Activity Propagation
Copyright (c) 2016 Xilinx, Inc. All Rights Reserved.
-----
| Tool Version      : Vivado v.2016.2 (win64) Build 1665346 Mon Sep 19 18:41:23 MDT
| Date             : Tue Sep 27 14:23:44 2016
| Host             : XHDAKASHA30 running 64-bit Service Pack 1 (build 7601)
| Command          : report_power
| Design           : dut_fpga
| Device           : xc7k325tffg900-2
| Design State     : routed
| Grade            : commercial
| Process          : typical
| Characterization  : Production
-----

Power Report

Table of Contents
-----
1. Summary
1.1 On-Chip Components
1.2 Power Supply Summary
1.3 Confidence Level
2. Settings
2.1 Environment
2.2 Clock Constraints
3. Detailed Reports
3.1 By Hierarchy

1. Summary
-----
    
```

Use Model Examples

Using the `cpu_hdl` design included with the Vivado tools, the following scripts provide examples for most of the commands discussed in the previous sections. You can perform power reporting dynamically using Tcl commands. For example:

```
vivado -mode batch -source power_analysis.tcl
```

You can also use a Tcl script. The script examples below assume that you are using the batch mode for sourcing the script.

Example 1: Post-Synthesis and Post-Implementation Power Estimation and Comparison in Project Mode

```

#----- Setup estimation -----

# Open example project with HDL source files and timing constraints
create_project project_1 $work_dir/project_1 -part xc7k70tffg676-2 -force
set_property target_language VHDL [current_project]
    
```

```

instantiate_example_design -template      xilinx.com:design:cpu_hdl:1.0
#----- Run Synthesis then Power estimation
-----

# Run Vivado Design Suite synthesis and automatically
launch_runs synth_1
wait_on_run synth_1

#open design
open_run synth_1

# Display tool default assumed operating conditions
report_operating_conditions -all

# Set specific device and environment operating conditions
set_operating_conditions -ambient 25
set_operating_conditions -voltage {vccint 1.0 vccaux 1.71}

# Generate verbose post-synthesis power report
report_power -verbose -file ex1_post-synthesis.pwr

#----- Run Implementation then Power estimation
-----
launch_runs impl_1
wait_on_run impl_1

#open design
open_run impl_1

# Generate post-implementation verbose power report
report_power -file ex1_post-implementation.pwr

# Return operating conditions to default for device
reset_operating_conditions -ambient -voltage {vccint vccaux}
    
```

Example 2: Post-Synthesis and Post-Implementation Power Estimation and Comparison in Projectless Mode

```

#----- Setup estimation -----

# Open netlist in projectless mode
read_edif -name top.edf

# AND link the design
link_design

# OR open Vivado checkpoint
open_checkpoint -file post_synth.dcp

# read design constraints, if it is not part of a design checkpoint (DCP)
read_xdc -name top_full.xdc

# Display tool default assumed operating conditions
report_operating_conditions -all

# Set specific device and environment operating conditions
set_operating_conditions -ambient 25
set_operating_conditions -voltage {vccint 0.95 vccaux 1.71}

#----- Power estimation at post synthesis -----
    
```

```
# Generate verbose post-synthesis power report
report_power -verbose -file ex1_post-synthesis.pwr

#----Run various Implementation steps then run Power estimation after every
step ----
opt_design
report_power -verbose -file ex1_post-opt_design.pwr
power_opt_design ;# Optional
report_power -verbose -file ex1_post_pwr_opt_design.pwr
place_design
report_power -verbose -file ex1_post_place_design.pwr
phys_opt_design ;# Optional
report_power -verbose -file ex1_post_phys_opt_design.pwr
route_design

# Generate post-route verbose power report
report_power -verbose -file ex1_post_route_design.pwr

# Return operating conditions to default for device
reset_operating_conditions -ambient -voltage {vccint vccaux}
```

Example 3: Examine and ensure Static Probability values on resets are accurate

```
# Query the Static Probability value of the reset
report_switching_activity -static_probability [get_ports reset]
# Output is - reset: static probability = 0.5 (D)

# Set Static Probability value and signal rate of reset to 0
set_switching_activity -static_probability 0.0 -toggle_rate 0 [get_ports
reset]

# Generate post-route verbose power report
report_power -verbose -file ex1_post_route_design.pwr
```

Example 4: What If? Design Analysis/Report, Edit, and Reset Design Activity

Working with power analysis can be very dynamic, allowing you to explore What If? scenarios on the fly. Open the previously implemented design, and enter or source the following commands. This modifies activity for control signals (clock enable and reset) in submodule `fftEngine` to evaluate the impact on power for this hierarchical level and the entire design.

```
#----- Report power and activity with default settings
-----

# Report power
report_power -file ex3_power_before.pwr

# Get activity of signals of interest
report_switching_activity [get_nets {fftEngine/reset fftEngine/wb_we_i_reg}]

#----- scenario with no reset and higher CE activity -----

# disable reset and enable clock enables in module fftEngine most of the
time
set_switching_activity -static_probability 0 -signal_rate 0 [get_nets
```

```
fftEngine/reset_reg]
set_switching_activity -static_probability 1 -toggle_rate 0 [get_nets
fftEngine/wb_we_i_reg]
report_power -file ex3_power_no_reset_activ.pwr
report_switching_activity [get_nets fftEngine/reset_reg fftEngine/
wb_we_i_reg]

#----- scenario with active reset and low CE activity -----

# enable reset and disable clock enable in module fftEngine most of the time
set_switching_activity -static_probability 1 -toggle_rate 0 [get_nets
fftEngine/reset_reg]
set_switching_activity -static_probability 0 -signal_rate 0 [get_nets
fftEngine/wb_we_i_reg]
report_power -file ex3_power_reset_activ.pwr
report_switching_activity [get_nets fftEngine/reset_reg fftEngine/
wb_we_i_reg]
```

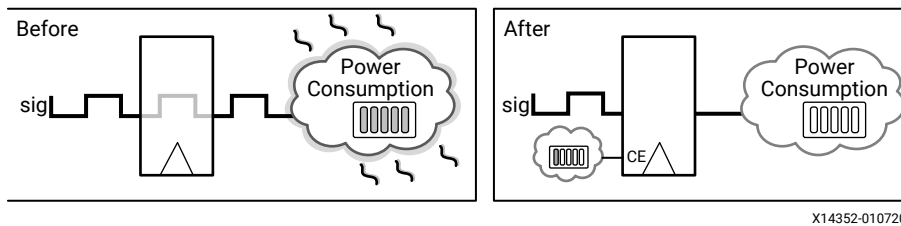
Power Optimization Feature

The Vivado® design tools offer a variety of power optimizations to minimize dynamic power consumption by up to 30% in your design. These optimizations use ASIC style clock gating techniques to minimize activity on portions of the design that do not contribute to the design output or those that do not require design state update for that clock cycle. These optimizations can be applied on the entire design or on selected portions of the design as shown in the following figure.

The dynamic power consumption of a device is determined by the operating clock frequency (f), node capacitance (C), device operating voltage (V), and the activity (α) on various nodes in the design. For most designs, several of the above parameters are typically fixed either by the device technology (for example, voltage) or by design requirements (for example, operating frequency). However, there are several nodes in the design that do not affect the output of the device but still continue to toggle. This constitutes a significant portion of wasted dynamic power. You can use the clock enables (CE) in the device for gating such nodes. While this is possible through optimal coding techniques, this is rarely done by the designer either because the design contains intellectual property (IP) from other sources or because of the amount of effort involved in performing such fine grained clock gating. Vivado automates these power optimizations under a single command to maximize power savings while minimizing your effort.

Vivado performs an analysis on the entire design, including legacy and third-party IP blocks, for potential power savings. It looks at the output logic of sourcing registers that do not contribute to the result for each clock cycle and then creates fine-grained clock gating and/or logic gating signals that neutralize unnecessary switching activity.

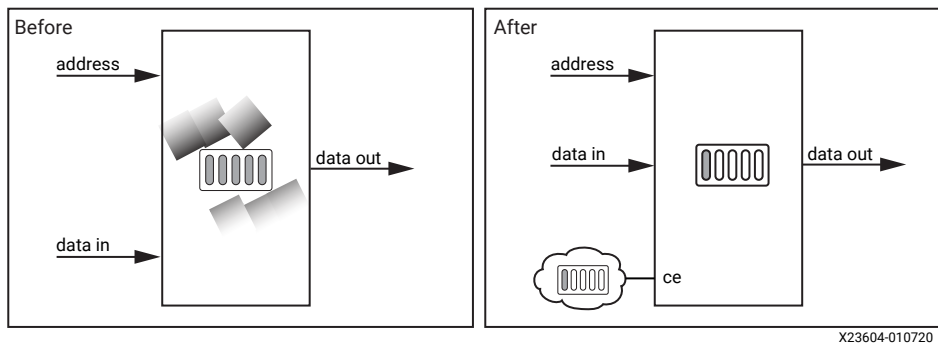
Figure 29: Intelligent Clock Gating



X14352-010720

The intelligent clock gating optimization also reduces power for dedicated block RAM in either simple dual-port or true dual-port mode as shown in the following figure. These blocks provide several enables: an array enable, a write enable, and an output register clock enable. Most of the power savings comes from using the array enable, and the software implements functionality to reduce power when no data is being written and when the output is not being used.

Figure 30: Clock Gating Optimizations Using Block RAM Enables



X23604-010720

Xilinx® intelligent clock gating optimizations do not modify user logic but instead create additional gating logic. Therefore the functionality of the design is preserved at all times. However, this optimization could impact timing, especially if the optimization is applied on critical paths.

Block RAM WRITE_MODE Power Optimizations

In Xilinx® 7 series devices, for block RAMs in true dual port (TDP) mode, the WRITE_MODE can be changed from WRITE_FIRST to NO_CHANGE safely if the output of that port is not connected or the corresponding output is not needed during write operation. Similarly, for block RAM in true dual port (TDP) mode, the WRITE_MODE can be changed from READ_FIRST to NO_CHANGE safely if the corresponding output port is not connected.

In UltraScale™ devices, in addition to the above optimization, for block RAM in Simple Dual Port (SDP) mode, WRITE_MODE of both the read and write ports can be changed to NO_CHANGE safely if the read and write port clocks are asynchronous. These changes help to save power in the write cycle by not updating the output port of the block RAM. This optimization will be performed only when there is no impact to user defined functionality and performance.

Block RAM Cascade Optimizations

In Xilinx® 7 series devices, if block RAMs are found to be cascaded, because only one block RAM can be active at any time, the rest of the block RAMs can be disabled based on address and any existing enable conditions. This enables large power savings. These optimizations are performed by default in the `opt_design` phase in the Vivado® Design Suite.

Performing Power Optimization in the Vivado Integrated Design Environment

Power optimizations are performed during two stages in Vivado:

- `opt_design`
- `power_opt_design`

Optimizations that are performed during the `opt_design` phase occur without user intervention. These optimizations primarily focus on power savings on block RAMs.



IMPORTANT! *The power optimization might impact the timing performance of your design during `opt_design`, `power_opt_design`, or both.*

For UltraScale™ devices, the more aggressive block RAM power optimizations that may negatively impact timing are included only in `power_opt_design`. This allows performance to be traded for power savings. For UltraScale+™ devices, XPM-URAM power optimization occurs in `power_opt_design`.

By default the `opt_design` command performs block RAM power optimization. Block RAM power optimization can also be run explicitly and standalone by using the `-bram_power_opt` option:

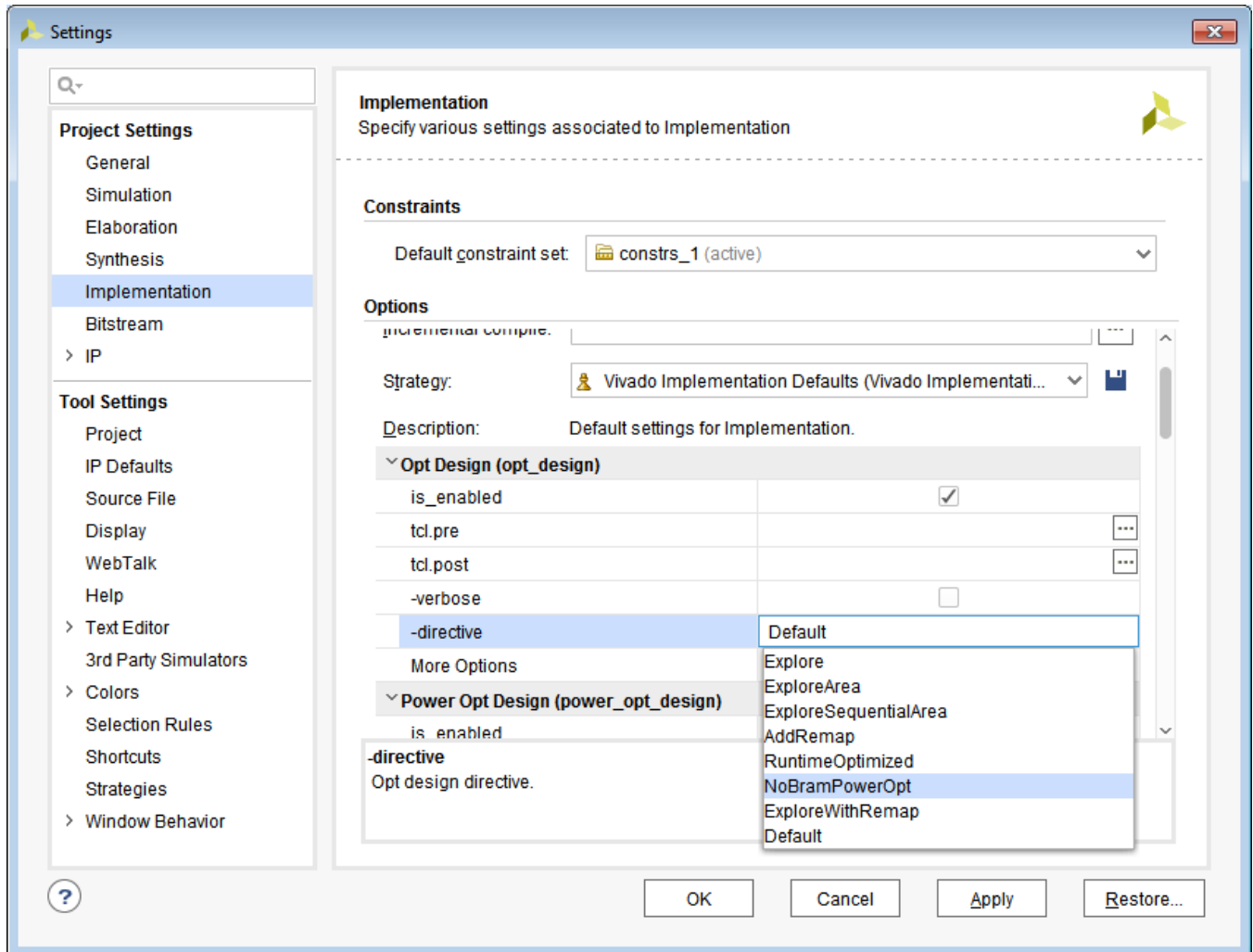
```
opt_design -bram_power_opt
```

To disable block RAM power optimization from the default `opt_design` flow, set the `NoBramPowerOpt` directive to the `opt_design` command:

```
opt_design -directive NoBramPowerOpt
```

You can also set this directive in the Implementation settings window as shown in the following figure.

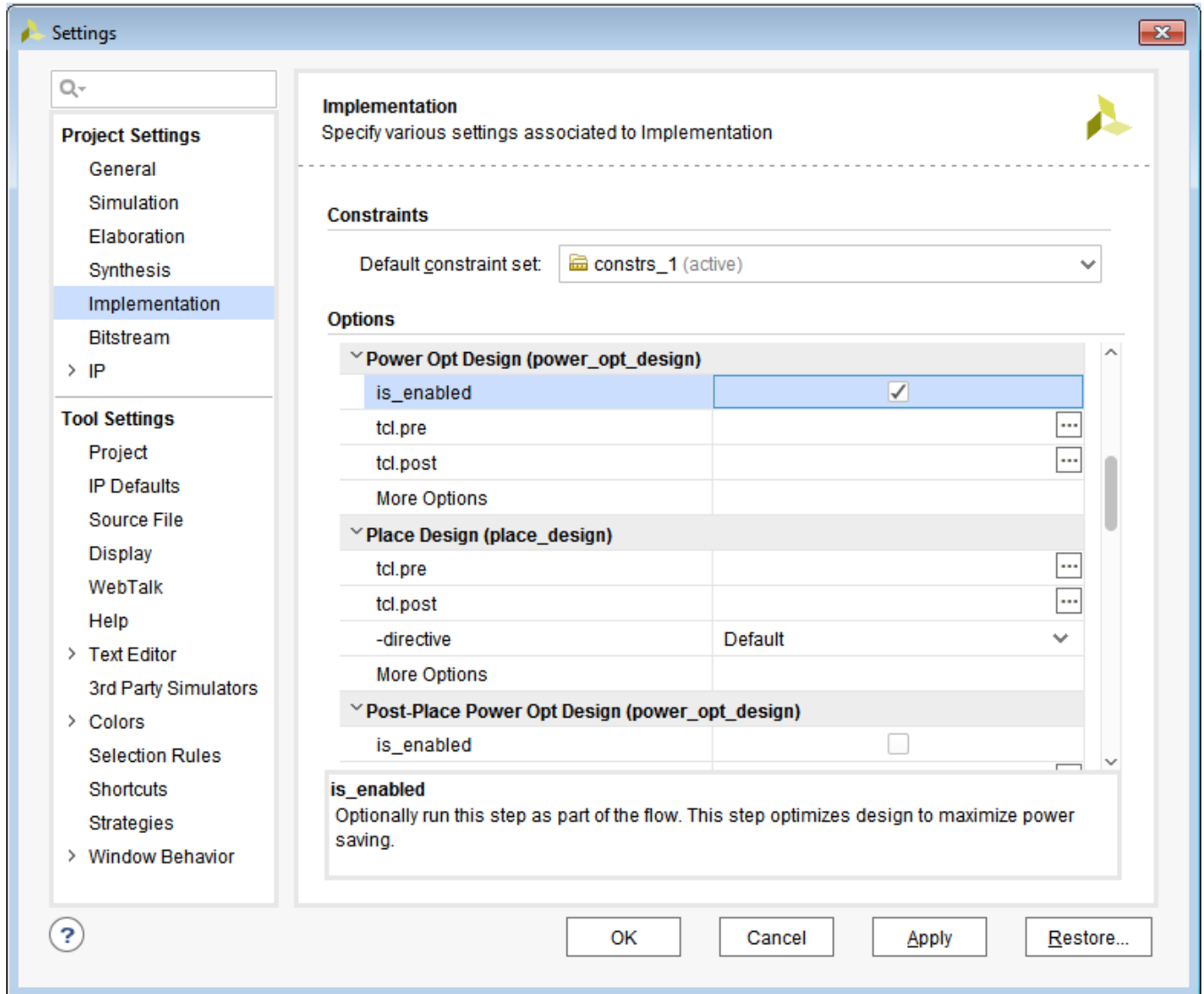
Figure 31: Disabling block RAM Power Optimization During Opt Design



To enable power optimization through `power_opt_design` in the Vivado® Integrated Design Environment, check the `is_enabled` option available by selecting **Tools** → **Project Settings** → **Implementation** → **Power Opt Design** as shown in the following figure. Once enabled, power optimization is run as a part of the implementation step in the Vivado Integrated Design Environment. To set fine grained control over optimization and to report the result of the optimization, refer to the [Power Analysis Tcl Commands](#) section.

★ **IMPORTANT!** Power Opt Design can be enabled either pre-place or post-place in the design flow, but not in both places. See [Running Power Optimization](#) for more details.

Figure 32: Power Optimization Option



Displaying a Power Optimization Report in the Vivado IDE

In the Vivado® IDE you can display a power optimization report that describes the power optimizations that have been performed on your design. You can display the power optimization report after synthesis or after implementation.

★ IMPORTANT! In Vivado, power optimization is performed during the *opt_design* and *power_opt_design* stages of the Vivado design flow. Both of these stages occur during implementation, which occurs after the design has been synthesized. If you generate a power optimization report on the synthesized design, the report will only contain information about the power optimization features that were coded into your original design (for example, gating a block RAM using a clock enable (CE)). The report will not detail power optimizations performed later by the tools, during implementation.

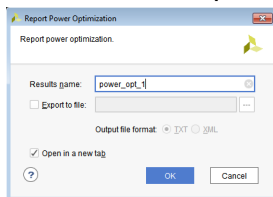
To display a Power Optimization Report in the Vivado integrated design environment:

1. In the Flow Navigator, select **Open Synthesized Design** or **Open Implemented Design**.
2. Select **Reports** → **Report Power Optimization**.

The equivalent Tcl command to perform this operation is:

```
report_power_opt -name <report_name>
```

3. In the Report Power Optimization dialog box, specify the following options.
 - **Results name:** Specify the name under which the power optimization report appears in the Vivado IDE.
 - **Export to file:** Check this box to generate a text report in addition to the power optimization report in the Vivado IDE. Specify a file name and location for the text report, and select whether this is a TXT or XML file.
 - **Open in a new tab:** Check this box to add this new power optimization report to any other power optimization reports currently displayed in the Vivado IDE. Leave this box unchecked to replace any power optimization reports currently displayed in the Vivado



4. Click **OK**.

A power optimization report appears in the results windows area of the IDE with this new power optimization report.

Elements	TOTAL	USER GATED	TOOL GATED	% GATED(Total)
Number of BRAMs	320	320	0	100.000
Number of SRLs	0	0	0	0.000
Number of Slice Registers	521	490	0	94.050
Number of XPM URAMs	0	0	0	0.000
BRAM write mode changes	640	0	0	0.000

You can select from different views of the power optimization report.

- **General Information:** Information about your design, the Xilinx® device into which your design is implemented, and the Tcl command that generated this power optimization report.
- **Summary:** Count of block RAMs, SRLs, and Slice Registers that were optimized by the user in the design and by the power optimization tool.
- **Recommendations:** Things you can do to further optimize your design for power.
- **Hierarchical Information:** Details of the block RAMs, SRLs, and Slice Registers for which Vivado has performed power optimization.

For a description of the power optimizations Vivado Integrated Design performs, see [Power Optimization Feature](#) and [Block RAM WRITE_MODE Power Optimizations](#).



TIP: If any hierarchical module or instance is tagged with a `DONT_TOUCH` attribute, Power Optimization does not optimize this logic.

Performing Power Optimization Using the Tcl Interface

There are four power optimization Tcl commands in Vivado®:

- `set_power_opt`
- `opt_design -bram_power_opt`
- `power_opt_design`
- `report_power_opt`

These commands can be used to enable power optimization as well as control portions of the design that are to be optimized, and to generate a report that shows the effect of the optimizations performed. For information on options, properties, applicable elements, or returned values for a specific command:

- Type `<command_name> -help`
- See the *Vivado Design Suite Tcl Command Reference Guide* ([UG835](#)) and the *Vivado Design Suite User Guide: Using Constraints* ([UG903](#))

Setting Power Optimization Constraints

Prior to running power optimization, you can optionally set power optimization constraints to identify portions of the design that need to be optimized for power. The `set_power_opt` command provides you the option to include or exclude cell types, hierarchy levels or clock domains for power optimization.



TIP: You need to use the `power_opt_design` command to enable the power optimization step. The `set_power_opt` command is used only for targeting the optimization.

The syntax for the `set_power_opt` command is:

```
set_power_opt [-include_cells <args>] [-exclude_cells <args>] [-clocks <args>] [-cell_types <args>] [-quiet] [-verbose]
```

Table 5: `set_power_opt` Options

Option Name	Optional	Default	Description
<code>-include_cells</code>	Yes	All	Include only the listed cells for clock gating
<code>-exclude_cells</code>	Yes	None	Exclude the listed cells from clock gating
<code>-clocks</code>	Yes	All clocks	Clock gate the cells clocked by the listed clocks only
<code>-cell_types</code>	Yes	All	Clock gate the following cell types only: [all bram uram reg srl none]
<code>-quiet</code>	Yes	N/A	Ignore command errors
<code>-verbose</code>	Yes	N/A	Suspend message limits during command execution

Examples

The following example sets power optimization for block RAM and REG type cells, then adds SRLs:

```
set_power_opt -cell_types {bram reg}
set_power_opt -cell_types {srl}
```

The following example sets power optimization for block RAM cells only, then excludes the cpuEngine block from optimization, but then includes the cpuEngine/cpu_dbg_dat_i block:

```
set_power_opt -cell_types bram
set_power_opt -exclude_cells cpuEngine
set_power_opt -include_cells cpuEngine/cpu_dbg_dat_i
```

Running Power Optimization

Power optimization works on the entire design or on portions of the design (when `set_power_opt` is used) to minimize power consumption. Power optimization can be run pre-place or post-place in the design flow, but not in both places. The pre-place power optimization step focuses on maximizing power saving. This could result in timing degradation in rare cases. If preserving timing is the primary goal, the post-place power optimization step is the recommended option. This step performs only those power optimizations that preserve timing. You could also run `phys_opt_design -bram_enable_opt` at post-place to revert some of the block RAM enable optimizations which affect timing. A typical pre-place power optimization script would be:

```
synth_design
opt_design
power_opt_design
place_design
route_design
report_power
```

The syntax for this command is:

```
power_opt_design [-quiet] [-verbose]
```

Table 6: power_opt_design Options

Option Name	Optional	Default	Description
-quiet	Yes	N/A	Ignore command errors
-verbose	Yes	N/A	Suspend message limits during command execution

Generating a Power Optimization Text Report

The `report_power_opt` command provides you with a text report containing a hierarchical breakdown of all the cells including block RAMs, SRLs, and registers that have been optimized for power. It provides information on the enables used for each cell and if the enables were created by Vivado® (or by the user). The syntax for this command is:

```
report_power_opt [-cell <arg>] [-file <arg>] [-quiet] [-verbose]
```

Table 7: `report_power_opt` Options

Option Name	Optional	Default	Description
<code>-cell</code>	Yes	Top level	Report power optimization for a specific cell
<code>-file</code>	Yes	None	Write the report into the specified file. The specified file will be overwritten if one already exists
<code>-quiet</code>	Yes	N/A	Ignore command errors
<code>-verbose</code>	Yes	N/A	Suspend message limits during command execution

Examples

The following example creates a file named `myopt.rep` and reports power optimization for the entire design:

```
report_power_opt -file myopt.rep
```

The following example creates a file named `myopt2.rep` and reports power optimization for the `mctrl0` sub-hierarchy of the design:

```
report_power_opt -file myopt2.rep -cell mcore0/mctrl0
```

Guidelines to Maximize Power Saving with Power Optimization

To maximize power savings when you run power optimization in the Vivado® tools, you should run power optimization on the entire design and not exclude portions of the design. If you do not see anticipated power savings after enabling power optimization, make sure the design is properly constrained. Check to see if all registers in the design have been constrained, using the `check_timing` command.

If the design has been constrained correctly, then review the design for potential coding styles that could impact power optimizations. The three areas of potential debug are the global set and reset signals, block RAM enable generation, and register clock gating. A low number of power optimization generated enables could indicate the need to review coding practices or options/properties set for design synthesis and implementation.

- Global set and reset signals

Where possible, minimize the use of asynchronous set/reset signals especially to datapath or pipeline flip-flops as well as block RAMs (block RAM).

You should also consider constraining the global set and reset signals as `dont_touch` during the `power_opt_design` step to avoid their use as enables. Note that setting `dont_touch` property in HDL will cause every step in the flow to obey this property. It is recommended that this option is set up as an XDC constraint only for the power optimization step. Here is an example of how to do this:

```
set_property DONT_TOUCH true [get_cells u1]
```

Finally, ensure that the signal rate and probabilities of the global set and reset signals are set correctly prior to running power optimization and vectorless power estimation.

- Slice registers and SRLs

A number of different reasons could explain why `power_opt_design` might not be able to generate clock enables for slice registers or SRLs in the design. Some examples are:

- Having combinatorial loops in the design
- Using set/reset signals at the flip-flops and SRLs that are sourced from primary inputs to the design
- Using asynchronous set/reset signals at the datapath flip-flops
- Large number of clock domains in the design preventing enables being generated due to clock domain crossing issues
- SRL sizes: Typically the larger the number of shift register stages in the SRLs, the more difficult it is to generate a single clock enable for all stages

- Block RAMs

Block RAM rich designs are excellent candidates for power savings. Vivado uses a variety of optimization techniques to generate enables and save power. If block RAM gating coverage is low after using `power_opt_design`, some of the possible reasons could be:

- Block RAMs are mainly FIFO18/FIFO36 cells. These cannot be optimized by the tool.
- Memories inferred or instantiated are mainly in true dual port (TDP) mode using asynchronous clocks on their A and B ports that cannot be optimized by `power_opt_design`.

- Use of asynchronous reset signals to either the block RAM themselves or to the address/write-enable flip-flops feeding the block RAMs.

Preserving Timing After Power Optimization

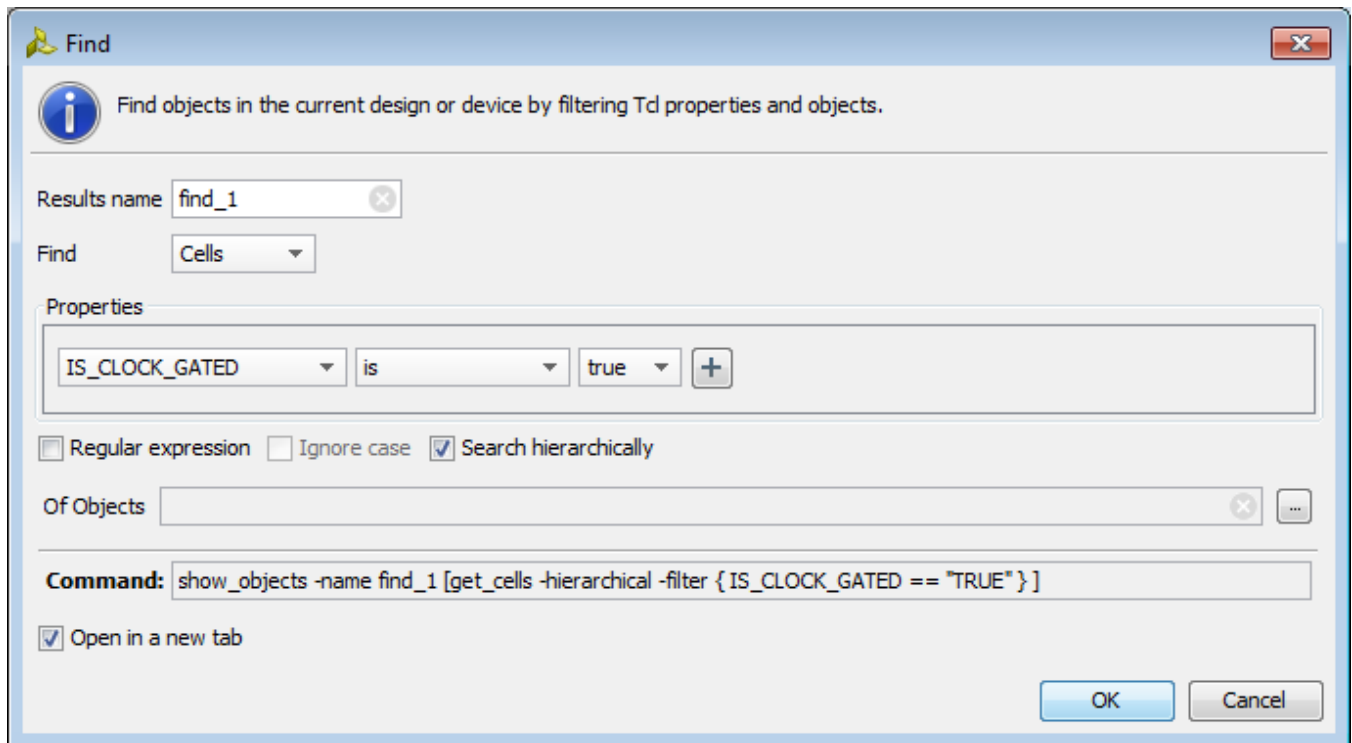
Power optimization works to minimize the impact on timing while maximizing power savings. However, in certain cases, if timing degrades after power optimization, you can employ a few techniques to offset this impact.

Where possible, identify and apply power optimizations only on non-timing critical clock domains or modules using the `set_power_opt` XDC command. If the most critical clock domain happens to cover a large portion of the design or consumes the most power, review critical paths to see if any cells in the critical path were optimized by power optimization. Note that objects optimized by power optimization have an `IS_CLOCK_GATED` property on them. Exclude these cells from power optimization. To locate clock gated cells, you can use the following Tcl command:

```
get_cells -hier -filter {IS_CLOCK_GATED==1}
```

You can use the Find dialog box to locate these cells as shown in the following figure.

Figure 33: Finding Power Optimized Cells



A simpler alternative is to limit power optimization to block RAMs. This minimizes the timing impact but its effectiveness is dependent on the number of block RAMs present in the design and how effectively they have been gated. To limit power optimization to block RAMs, run a `set_power_opt -cell_types {bram}` command before running the `opt_design` or `power_opt_design` commands.

Achieving an Accurate Power Analysis Using Vivado Report Power

Introduction

Accurate power estimation is always challenging for the software tools, because the tools have to assume various factors on their own. If you can guide the tool as much as possible to minimize these assumptions, you can achieve a more accurate power estimation. For an accurate power analysis, the following factors must be considered:

- [Thermal Settings](#)
- [Power Supply Settings](#)
- [Clock Specifications](#)
- [Control Signals](#)
- [Primary Inputs](#)
- [Component Level](#)

Thermal Settings

Ideally, static power is the sum of source to drain and gate leakage power in the transistor. Static power is purely dependent on Thermal conditions. Providing more accurate thermal information is a basic requirement for accurate power estimation.

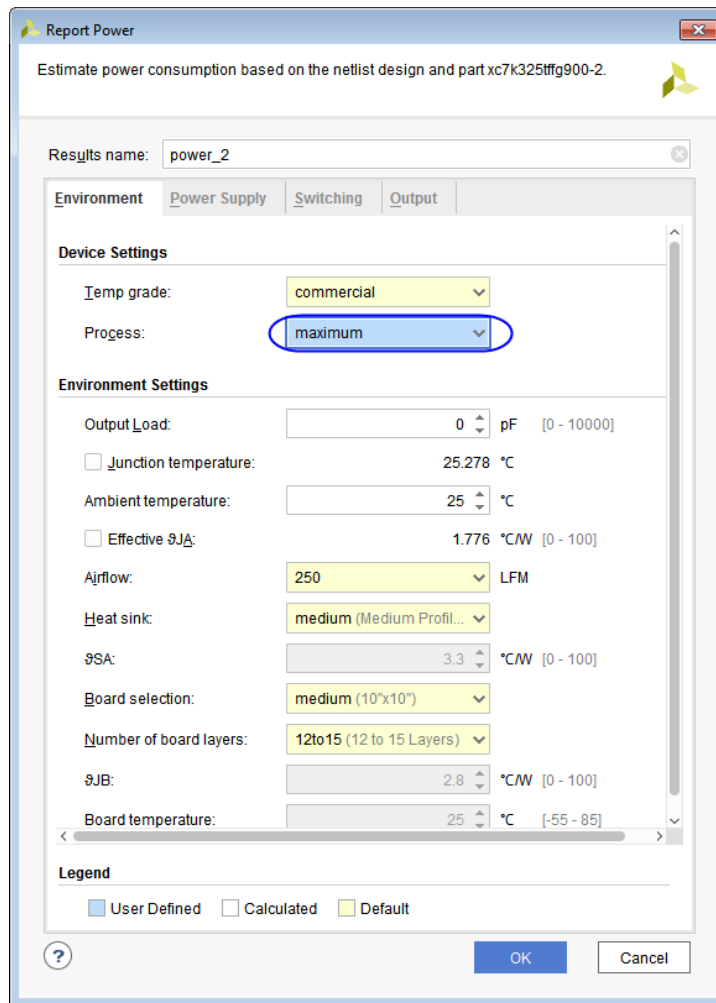
Process Corners

When devices are fabricated, each device has variations of performance and power consumption, due to the manufacturing process. Report Power offers static power estimation for two process corners, typical and maximum. Ideally all devices should meet the typical estimation value. But process variations result in a distribution of devices, which needs to be centered on the typical value, adjusted manually based on process variation for any particular device. A maximum setting, however, guarantees that the reported numbers are within operating range and closer to hardware measurements. At a fixed Junction Temperature, the expected variation in static power from typical to maximum would be ~2.5X on Commercial devices.

★ IMPORTANT! Use the maximum Process setting to achieve worst-case static power accuracy.

In Vivado®, the default Process is typical in Report Power. This can be changed to maximum in the Environment tab of the Report Power dialog box:

Figure 34: Setting the Process for Report Power



Equivalent Tcl command:

```
set_operating_conditions -process maximum
```

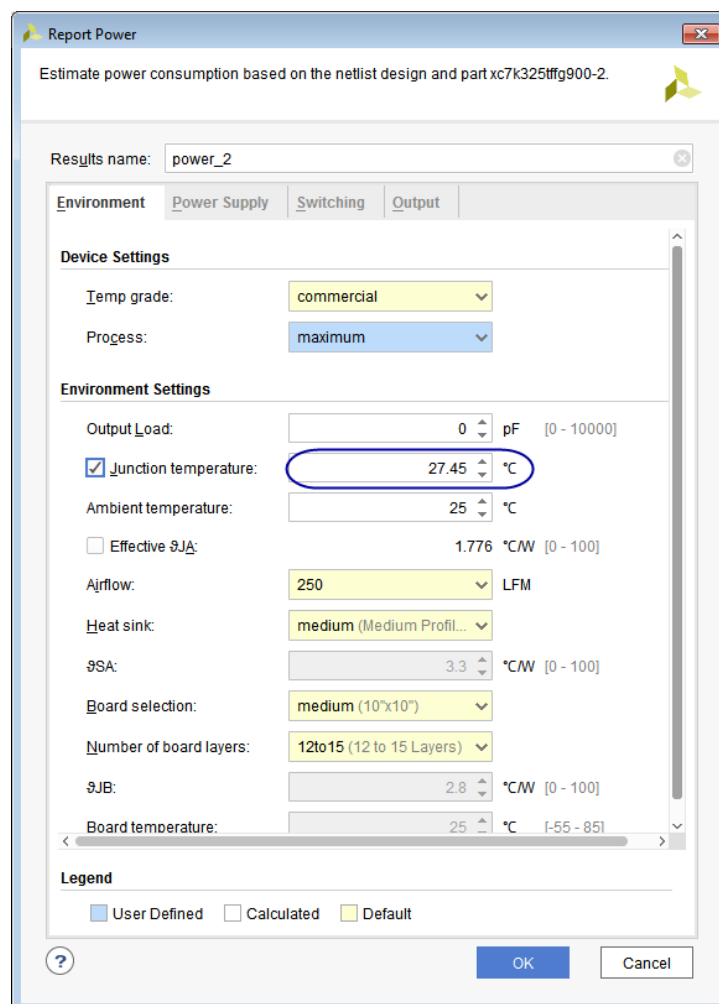
Junction Temperature

Leakage current increases exponentially with Junction Temperature, which results in higher static power. Junction Temperature depends on various factors: the total power of the device, the cooling system, board selection, and ambient conditions. By default the Junction Temperature is computed based on other Thermal setup inputs: Ambient Temperature, Heat Sink, Board Selection, etc. Because Junction Temperature is directly proportional to total power, it varies when dynamic power increases. It is very important to specify the right Junction Temperature to estimate accurate static power.

★ IMPORTANT! Read the Junction Temperature at the time when power is measured on the hardware and overwrite the existing setting in the Report Power dialog box.

To set Junction Temperature in the Vivado® IDE, enable the Junction Temperature check box in the Environment tab of the Report Power dialog box and enter the value.

Figure 35: Setting Junction Temperature for Report Power



Equivalent Tcl command:

```
set_operating_conditions -junction_temp 45
```

You can measure approximate Junction Temperature by placing a simple thermistor or other hand-held temperature measurement device on the Xilinx device. If one of the Xilinx Hardware Programming tools is used to program the devices, then you can read the Die Temperature values. For example, ISE-Impact reads Die Temperature values when you select **Debug → Read Status Register**. Vivado Hardware Manager graphically drafts the Die Temperature plots in the System Monitor Window.

Power Supply Settings

Voltage scaling is one of the prominent power saving approaches in Xilinx® devices. There are different voltage rails to supply specific voltages to logic in the device. Each Device Data Sheet lists the recommended operating conditions of these rails. For example, Kintex®-7 devices can operate with a VCCINT rail between 0.97V and 1.03V. You can make use of voltage scaling to meet your power budget. In the Hardware setup, these rails are supplied by external Power Regulators with fine-grained controls. Because power increases when supply voltage increases, you must provide the exact supply voltage to estimate power accurately.



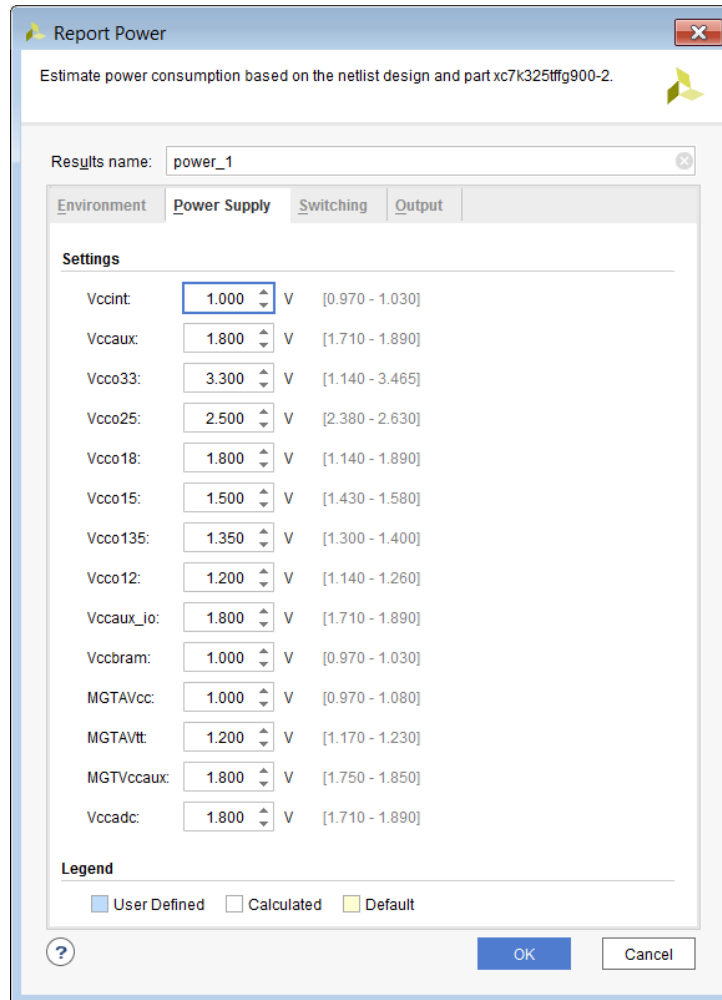
IMPORTANT! Specify accurate power supply values in the Power Supply tab of the Report Power dialog box.

To specify power supply voltages in the Vivado® Integrated Design Environment, enter the values in the Power Supply tab of the Report Power dialog box.



TIP: While moving away from the nominal voltage, ensure that the MIN and MAX voltage specification for the rail can still be met. This does not impact the positive or negative margin available for the power delivery and PDN to supply the required power for transient events. The AC ripple and DC tolerance must remain within the data sheet specification.

Figure 36: Setting Power Supply Values for Report Power



Equivalent Tcl command:

```
set_operating_conditions -voltage {vccint 0.98 vccaux 1.8}
```

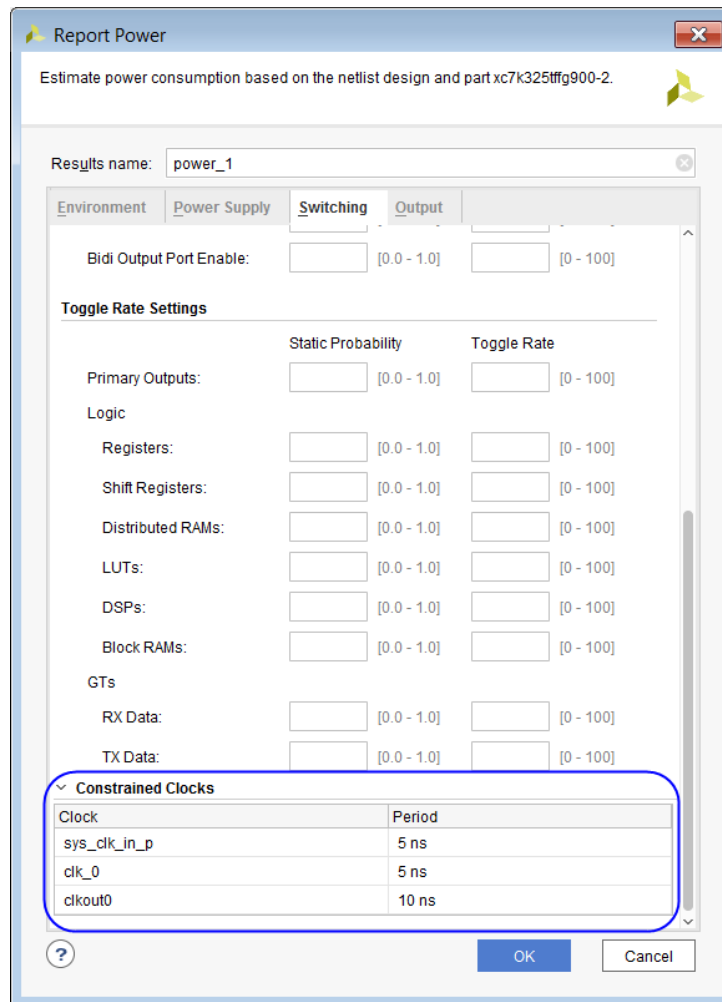
Clock Specifications

Design clocks are the main component for dynamic power computation. If no clocks are defined, switching activity estimates will be inaccurate, resulting in inaccurate power estimates. A clock node is identified from timing constraints which are defined using `create_clock` or `create_generated_clock` XDC commands.

Note: All the required clocks in the design must be defined using `create_clock` or `create_generated_clock` commands.

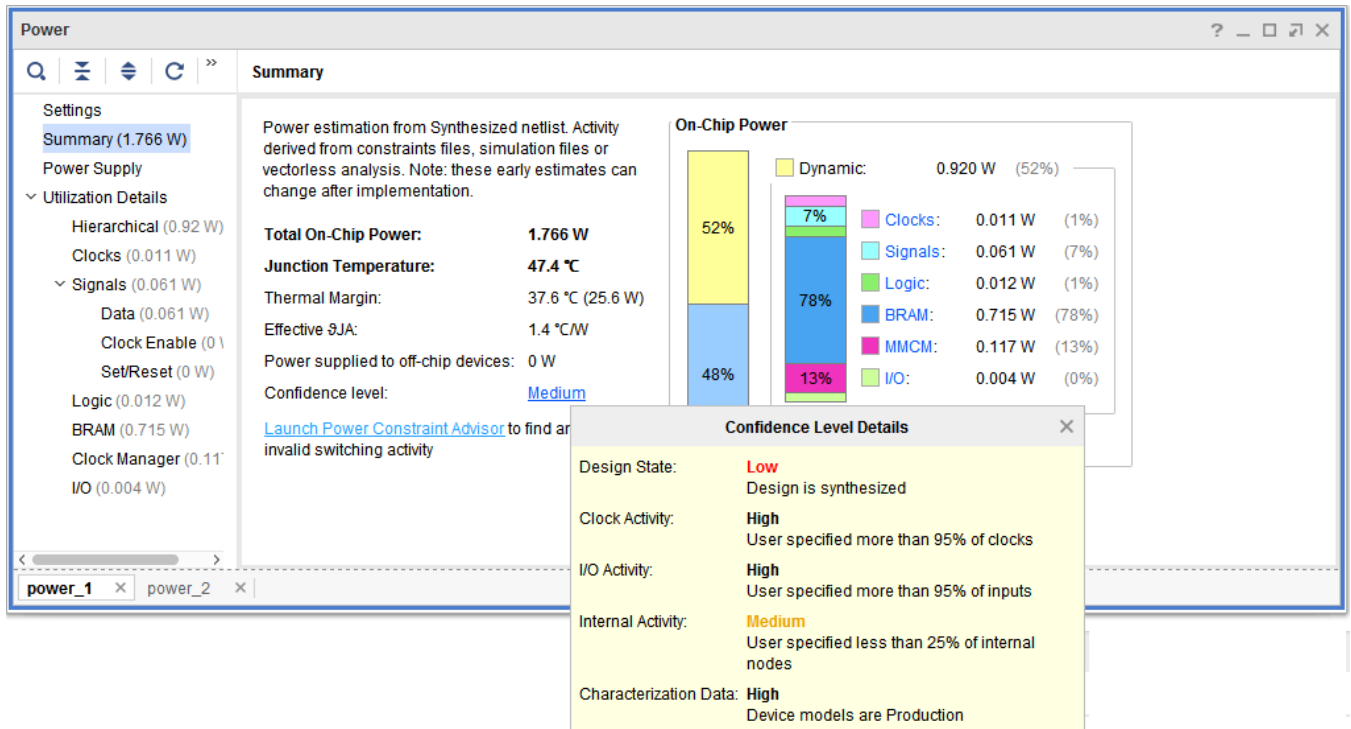
The Switching tab of the Report Power dialog box displays all the clocks defined in the design.

Figure 37: Constrained Clocks for Report Power



Make sure all the clocks defined in the design are displayed. Once Report Power runs, the Power Report confirms the percentage of clocks defined in the design when you view the Confidence Level details from the Summary page. This guides you to make sure there is a HIGH confidence level on Clock Activity.

Figure 38: Confidence Level and Specified Clocks



In Tcl mode, use the `get_clocks` and `report_clocks` commands to get the list of defined clocks. The text report gives the Confidence Level on Clock Activity:

```
report_power -file power.rpt
```

Figure 39: Text Report - Confidence Level for Clock Activity

```
1.3 Confidence Level
```

User Input Data	Confidence	Details	Action
Design implementation	Low	Design is synthesized	Accuracy of the tool is not optimal until design is fully placed and routed
Clock node activity	High	User specified more than 95% of clocks	
I/O node activity	High	User specified more than 95% of inputs	
Internal node activity	Medium	User specified less than 25% of internal nodes	Provide missing internal node activity with simulation results or by editing
Device models	High	Device models are Production	
Overall confidence level	Medium		

Control Signals

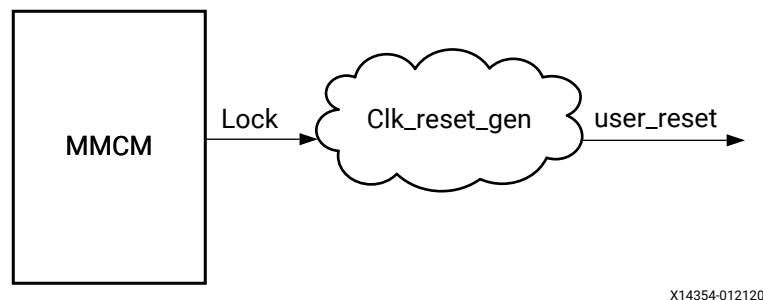
Global and Regional Resets

The Activity rate on Global Resets could change the power estimation dramatically. It conveys the state of each logic block in the design and the probability of logic output changes. If it is not set with the right switching information, you can get unrealistic power estimates. For example, ideally Reset is expected to be asserted (active) at the beginning of the run for a few cycles and remains inactive the rest of the time. This could be denoted in terms of switching activity as:

```
set_switching_activity -static_probability 0.01 -signal_rate 2 [get_ports
glb_reset]
```

Report Power identifies primary ports which are found to be global resets and applies the above switching activity. It uses a very conservative and safe way to identify the global resets - the ports which are directly connected to Reset pins of leaf primitives. However this does not help much on complex designs where the Reset logic is generated internally through special logic circuits (reset generator, debouncer, reset stretching, etc). When there is logic involved to generate Reset, Report Power is not aware of design intent and does not apply any default switching information on it.

Figure 40: Reset Logic



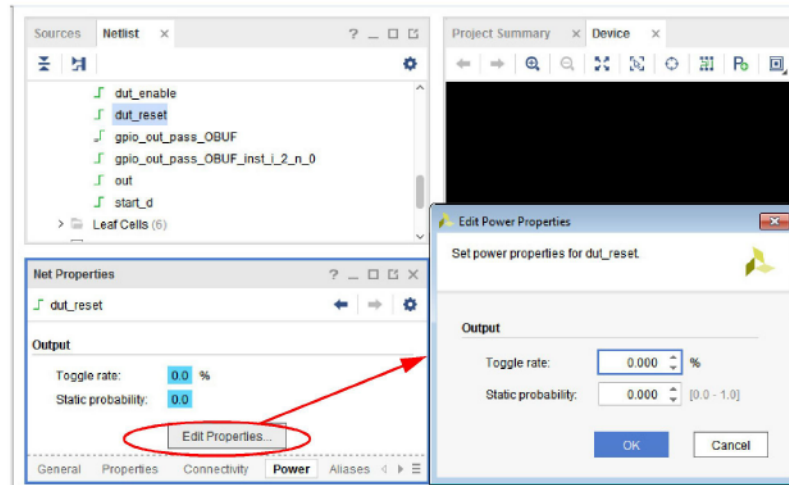
In this situation, the Reset activity information is derived from the generated logic using a probabilistic computation and propagation algorithm. Probabilistic computation is done at the leaf primitive level of logic. At times, the probabilistic algorithm lags handling of specific logic blocks, such as deep nested feedback logic. This results in unexpected switching activity on Reset nets.



RECOMMENDED: Make sure to supply the correct switching information on global/regional Reset nets.

The designer is expected to be aware of such global reset nets in the design. Set activity rates directly on these nets in the Power tab of the Net Properties window.

Figure 41: Setting Net Activity Rate



Equivalent Tcl command:

```
set_switching_activity -static_probability 0.01 -signal_rate 2 [get_nets u1/clkRst_gen/user_reset]
```

The Power Report also helps identify the Reset nets in the design, so you can verify the switching information on these nets and take corrective action. You can run a first trial run of Report Power using the default settings to analyze the activity on Reset nets.

Figure 42: Reset Net in the Power Report

Utilization	Name	Signal Rate (Mtr/s)	% High	Fanout	Slice Fanout	Clock	Logic Type
0 W	dut_fpga						
0 W	dut/dut_reset	0.000	0.000	530	0	clkout0	FF LUT
0 W	led_OBUF	0.000	100.000	3	0	clkout0	FF I/O LUT

Note that the Power Report also shows the number of logic cells that are affected by this Reset net: Fanout. If the initial switching activity estimation does not seem correct, you can select the net in the Power Report (as shown above) and edit the Power properties in the Net Properties window.

Note: Report Power displays both Preset/Set and Reset nets combined in the design. The above guidelines for Reset nets also apply to Preset/Set nets.

Global Clock Enables

In general, dealing with Clock Enables is less complex than dealing with Reset. In most of the design usage, it is obvious and straightforward. However, Clock Enables can grow as complex as Reset on power aware designs in which Clock Enables are extensively used and are controlled using special logic circuits. Dynamic power on logic cells depends on the switching activity of Clock Enables. If the activity rates are not set properly, it will easily result in inaccurate numbers.

For example, Enable is expected to be asserted (active) throughout the run and remains inactive only when the logic cell is not being used - if at all explicitly controlled to save power. This could be denoted in terms of switching activity as:

```
set_switching_activity -static_probability 0.99 -signal_rate 2 [get_ports glb_enable]
```

Report Power identifies primary ports which are found to be global enables and applies the above switching activity. It uses a very conservative and safe way to identify the global enables: the ports which are directly connected to CE pins of leaf primitives.



RECOMMENDED: Make sure to supply the correct switching information on global/regional Enable nets.

The Power Report also helps identify such Enable nets in the design, so that you can quickly validate the switching information on these nets and take corrective action. You can run a first trial run of Report Power using the default settings to analyze the activity on Enable nets.

Figure 43: Clock Enable Net in the Power Report

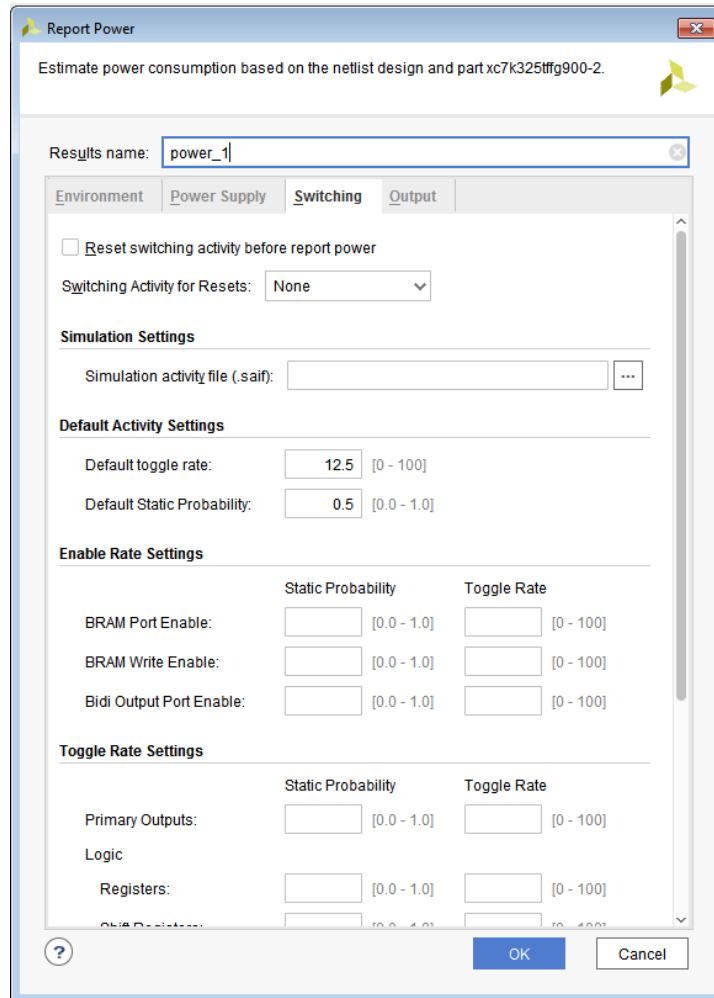
Utilization	Name	Signal Rate (Mtr/s)	% High	Fanout	Slice Fanout	Clock	Logic Type
0 W	dut_fpga						
0 W	dut/start_d	0.000	100.000	490	0	clkout0	FF
0 W	led_OBUF	0.000	100.000	3	0	clkout0	FF I/O LUT

Note that the Power Report also gives information about the number of logic cells that are affected by this Enable net, in the Fanout and Logic Type columns. If the initial switching activity estimation does not seem correct, you can select the net in the Power Report and edit Power properties in the Net Properties window.

Primary Inputs

Common nodes are taken care of with the above recommendations. However, design specific handshaking (protocols, memory interface, etc.) and data ports also need attention. Ideally, the activity rates on primary ports decide the overall activity of the design, which influences the dynamic power accuracy. Report Power assigns a default switching activity of Toggle_rate=12.5 and Static_Probability=0.5 on primary inputs (except clock and control ports). These values mean that the port will toggle once in eight clock cycles, and 50% of time the port stays at High (Logic 1). This assumption works fairly well on data ports. But it will have a huge accuracy impact when it is applied to handshaking nodes. This emphasizes the importance of correct switching information settings on primary inputs. The default activity settings can be found in the Switching tab of the Report Power dialog box:

Figure 44: Setting Default Switching Activity



You can change the default values which will be applied to all primary inputs (non-clock and non-control). Equivalent Tcl command:

```
set_switching_activity -default_static_probability 0.5 -default_toggle_rate 25
```

The same activity rate is applied to all the primary inputs - Report Power does not understand and distinguish handshaking ports from data ports. So it is important to specify the activity rates manually for the handshaking ports. This can be done either through the Vivado® Integrated Design Environment or a Tcl command.

Note: Make sure correct switching values are set on primary I/O Ports.

In the Power Report, the I/O section lists all the ports and corresponding switching activity information.

Figure 45: I/O View in the Power Report

Utilization	Name	I/O Type	I/O Standard	Drive ...	Input Pins	Output Pins	Bidir Pins	IO LOGIC SERI
0.004 W (<1% of...)	dut_fpga							
0.004 W (<1...)	sys_clk_...	HP	DIFF_SSTL15	N/A	1	0	0	No
<0.001 W (<1...)	fmc_out	HR	LVCMOS33	12.000	0	10	0	No
0 W	gpio_out...	HR	LVCMOS33	12.000	0	1	0	No
0 W	led	HP	LVCMOS15	12.000	0	1	0	No

Verify the activity rates on I/O ports. To change the activity rate, select the input port in the Power Report and edit the Power properties in the I/O Port Properties window.

Equivalent Tcl commands:

```
set_switching_activity -static_probability 0.25 -toggle_rate 10 [get_ports im_fcx_sync_in]

set_switching_activity -static_probability 0.5 -toggle_rate 50 [get_ports im_fcx_data_in]
```

Component Level

Finally, monitor the activity rates across major power consuming primitives in the design. After all the above points are taken care of, the activity rates across the hard blocks such as block RAMs, GTs, and DSPs should reflect meaningful values. However, Xilinx® recommends you to double-check these values, to make sure that there are no internal logic propagation or modeling issues in the tool.

For example, one known limitation is that the Report Power does not propagate activity rates across GTs. If any GT data outputs are consumed by logic, you must set activity rates explicitly on GT TX/RX outputs.

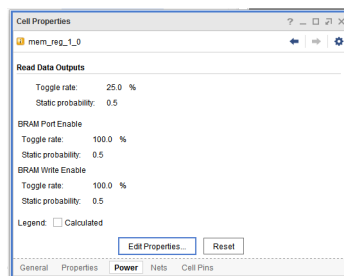
Report Power offers a simple interface in the Report Power dialog box to set the output activity rates on various types: registers, shift registers, LUTs, RAMs, block RAMs, DSPs, and GTs. These settings are the equivalent of the `-type` argument of `set_switching_activity` command. After a value is set, it is retained for subsequent power reporting runs. Global settings affect all the instances of hard primitives in the design. For example, a Toggle Rate set on block RAMs will be applied to all the block RAMs in the design. Alternatively, the Cell Properties window could also be used to change the activity rates. In the Power Report, review block RAM, DSP, and GT sections:

Figure 46: Activity Rate for a Block RAM

Utilization	Name	Mode	Signal Rate	Clock Name A
0.715 W (40% of total)	dut_tpga			
> 0.005 W (<1% of total)	Cascade Group (2 BRAMs)	CASC	3.456	dut_clk
> 0.005 W (<1% of total)	Cascade Group (2 BRAMs)	CASC	3.456	dut_clk
> 0.005 W (<1% of total)	Cascade Group (2 BRAMs)	CASC	3.456	dut_clk
> 0.005 W (<1% of total)	Cascade Group (2 BRAMs)	CASC	3.456	dut_clk
> 0.005 W (<1% of total)	Cascade Group (2 BRAMs)	CASC	3.456	dut_clk
> 0.005 W (<1% of total)	Cascade Group (2 BRAMs)	CASC	3.456	dut_clk
> 0.005 W (<1% of total)	Cascade Group (2 BRAMs)	CASC	3.456	dut_clk
> 0.005 W (<1% of total)	Cascade Group (2 BRAMs)	CASC	3.456	dut_clk
> 0.005 W (<1% of total)	Cascade Group (2 BRAMs)	CASC	3.456	dut_clk
> 0.005 W (<1% of total)	Cascade Group (2 BRAMs)	CASC	3.456	dut_clk
> 0.005 W (<1% of total)	Cascade Group (2 BRAMs)	CASC	3.456	dut_clk
> 0.005 W (<1% of total)	Cascade Group (2 BRAMs)	CASC	3.456	dut_clk
> 0.005 W (<1% of total)	Cascade Group (2 BRAMs)	CASC	3.456	dut_clk

To change the activity rate, select a hard block instance in the Power Report and edit the Power properties in the Cell Properties window.

Figure 47: Power Properties View for Block RAM Cell



Equivalent Tcl commands to change the activity rates on types:

- To set activity rates on all block RAMs in the specific design hierarchy instance `u1/transmit`:

```
set_switching_activity -static_probability 0.25 -toggle_rate 10 -type
bram [get_cells u1/transmit]
```

- To set activity rates on all the GTs present in the design:

```
set_switching_activity -static_probability 0.5 -toggle_rate 50 -type gt -all
```

Versal ACAP and Report Power

Introduction to Versal ACAP

Versal™ adaptive compute acceleration platforms (ACAPs) combine Scalar Engines, Adaptable Engines, and Intelligent Engines with leading-edge memory and interfacing technologies to deliver powerful heterogeneous acceleration for any application. Most importantly, Versal ACAP hardware and software are targeted for programming and optimization by data scientists and software and hardware developers. Versal ACAPs are enabled by a host of tools, software, libraries, IP, middleware, and frameworks to enable all industry-standard design flows.

Built on the TSMC 7 nm FinFET process technology, the Versal portfolio is the first platform to combine software programmability and domain-specific hardware acceleration with the adaptability necessary to meet today's rapid pace of innovation. The portfolio includes six series of devices uniquely architected to deliver scalability and AI inference capabilities for a host of applications across different markets—from cloud—to networking—to wireless communications—to edge computing and endpoints.

The Versal architecture combines different engine types with a wealth of connectivity and communication capability and a network on chip (NoC) to enable seamless memory-mapped access to the full height and width of the device. Intelligent Engines are SIMD VLIW AI Engines for adaptive inference and advanced signal processing compute, and DSP Engines for fixed point, floating point, and complex MAC operations. Adaptable Engines are a combination of programmable logic blocks and memory, architected for high-compute density. Scalar Engines, including Arm® Cortex®-A72 and Cortex-R5F processors, allow for intensive compute tasks.

The Versal AI Edge series focuses on AI performance per watt for real-time systems in automated drive, predictive factory and healthcare systems, multi-mission payloads in aerospace & defense, and a breadth of other applications. More than just AI, the Versal AI Edge series accelerates the whole application from sensor to AI to real-time control, all with the highest levels of safety and security to meet critical standards such as ISO26262 and IEC 61508.

The Versal AI Core series delivers breakthrough AI inference acceleration with AI Engines that deliver over 100x greater compute performance than current server-class of CPUs. This series is designed for a breadth of applications, including cloud for dynamic workloads and network for massive bandwidth, all while delivering advanced safety and security features. AI and data scientists, as well as software and hardware developers, can all take advantage of the high-compute density to accelerate the performance of any application.

The Versal Prime series is the foundation and the mid-range of the Versal platform, serving the broadest range of uses across multiple markets. These applications include 100G to 200G networking equipment, network and storage acceleration in the Data Center, communications test equipment, broadcast, and aerospace & defense. The series integrates mainstream 58G transceivers and optimized I/O and DDR connectivity, achieving low-latency acceleration and performance across diverse workloads.

The Versal Premium series provides breakthrough heterogeneous integration, very high-performance compute, connectivity, and security in an adaptable platform with a minimized power and area footprint. The series is designed to exceed the demands of high-bandwidth, compute-intensive applications in wired communications, data center, test & measurement, and other applications. Versal Premium series ACAPs include 112G PAM4 transceivers and integrated blocks for 600G Ethernet, 600G Interlaken, PCI Express® Gen5, and high-speed cryptography.

The Versal architecture documentation suite is available at: <https://www.xilinx.com/versal>.

Report Power for Versal ACAP

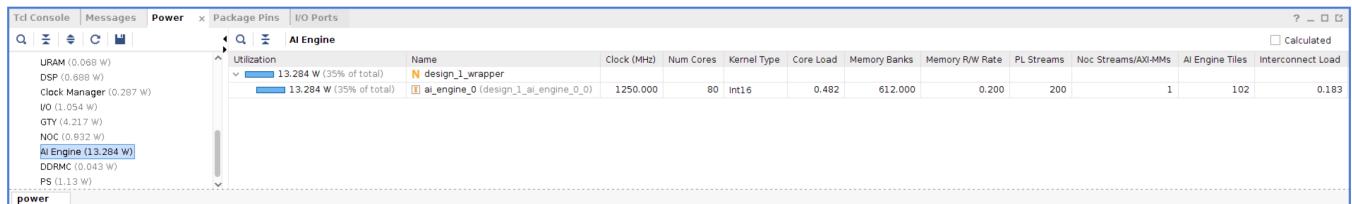
Report power for Versal™ ACAP features similar options and methodologies listed in previous chapters which apply to all Xilinx® devices. This following sections describe specific features of Versal ACAP:

- [AI Engine](#)
- [Network on Chip \(NoC\)](#)
- [DDR Memory Controller \(DDRMC\)](#)
- [MRMAC](#)
- [PCIe](#)
- [DSP](#)
- [PS-PMC](#)
- [IO](#)
- [CPM](#)
- [GTY](#)

AI Engine

The AI Engine is an array of VLIW SIMD high-performance processors that cater to solutions with high compute or complex DSP intensive applications, for example, 5G Wireless or Machine Learning algorithms. The AI Engine is available in all AI core series ACAP devices. AI Engine based designs can be generated using the Vitis™ integrated design environment (IDE). For more information, see *Versal ACAP AI Engine Programming Environment User Guide (UG1076)*. During the compilation of the AI Engine design using Vitis, a Vivado® Design Suite compatible project and design checkpoint (post route stage) is created by default. This generated `.dcp` file can be opened using Vivado and used for power estimation. `report_power` also reports AI Engine parameters, such as cores and tiles used, vector type and load, memory read and write rates as per the design. The maximum AI Engine frequency varies according to the device speed grade. The `.xpe` file generated using Vitis software platform can also be imported into the XPE tool. Also, there is a method to generate `.xpe` file for simulation flow from the Vitis software platform. For more information about XPE import feature, see *Xilinx Power Estimator User Guide for Versal ACAP (UG1275)*.

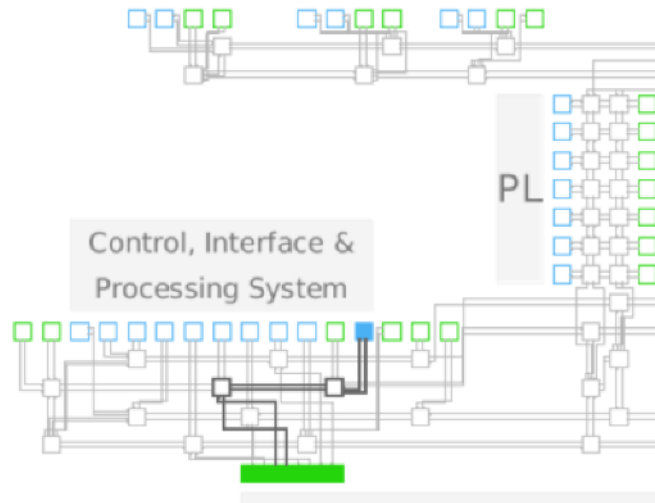
Figure 48: AI Engine Report Power



Network on Chip (NoC)

The Versal™ Network on Chip (NoC) is a high bandwidth interface within a Versal ACAP. It allows high bandwidth and Quality of Service (QoS) support for all resources on a Versal ACAP. NoC is an AXI-interconnecting network used for sharing the data among IP endpoints in the programmable logic (PL), the processing system (PS), and other hard blocks. A single physical NoC channel can be shared by traffic with different latency or bandwidth requirements. The NoC is configured using the NoC compiler to setup the required connections and the QoS. Report power uses the output of the NoC compiler automatically to estimate the power for your NoC configuration. For more information, see *Versal ACAP Programmable Network on Chip and Integrated Memory Controller LogiCORE IP Product Guide (PG313)*. The following figure shows the NoC view in the Vivado® IP integrator:

Figure 49: NoC View



The following figure shows the NoC QoS view:

Figure 50: NoC QoS View

Name	Traffic Class	Read Bandwidth Required (MB/s)	Read Bandwidth Estimate (MB/s)	Read Latency Estimate (cycles)	Traffic Class	Write Bandwidth Required (MB/s)	Write Bandwidth Estimate (MB/s)	Write Latency Estimate (cycles)
design_1_i/axi_noc_0/inst/500_AXI_nmu0d_5b05_500_AXI_nmu_0_top_INST/NOCC_NMU128_INST	Read	100	100 (0)	300	Write	100	100 (0)	0
design_1_i/axi_noc_0/inst/NOCC_ddrc/inst/moc_ddr4_phy/inst/ku_ddrmc_main	Best Effort				Best Effort			

The following figure shows the NoC Power view generated by the report power command:

Figure 51: NoC Power View

Utilization	Name	Data P...	Read B...	Write ...	Read Tran...	Write Tr...	Switches	Interconnect Distance
2.479 W (14% of total)	design_1_wrapper							
2.479 W (14% of total)	design_1_i/axi_noc_0/inst/M00_AXI	PL_TO_...	5.000	5.000	256.000	256.000	4	6.000

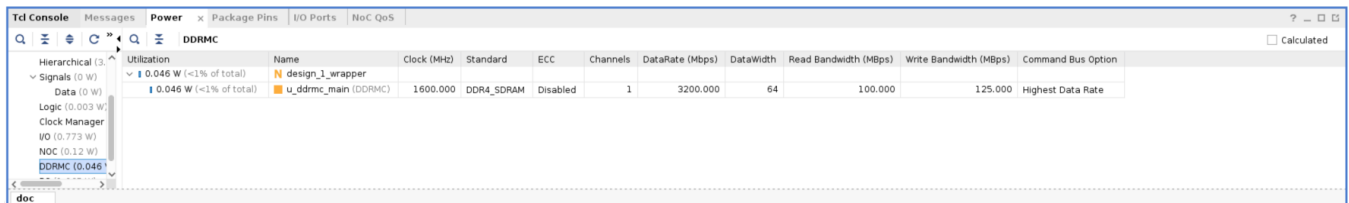
Note: NoC clock gating is enabled by default for `xvc1902`, `xvm1802`, and `xvcp1202`. Hence, power is reported only for the used clock buffers in the design. Versal premium stacked silicon interconnect technology (SSIT) devices do not support NoC clock gating in the present release. Hence, all the clock buffers are used in every design.

Vivado/Vitis generates .xpe file during implementation runs as well as during NoC compiler runs. These .xpe files can be imported to the XPE tool using import functionality for performing detailed what-if analysis. For more information, see *Xilinx Power Estimator User Guide for Versal ACAP* (UG1275). The generated .xpe file can be found in the following path: `project_1.gen/sources_1/common/nsln/NOC_Power.xpe`.

DDR Memory Controller (DDRMC)

Versal™ ACAP devices have a hardened DRAM controller available up to a maximum of 4. This can be accessed from the PS or the PL. However, the connection must go through the NoC. The DRAM memory controller can only be accessed using NoC. Each memory controller instance is connected to the NoC using four system ports. Each of these ports are bi-directional 128-bit data paths with three traffic types. A maximum data rate of 4266 Mb/s, per data pin, is supported for LPDDR4/4X and a maximum data rate of 3200 Mb/s is supported for DDR4/4E. For more information, see Memory Interface Controller section of *Versal Prime Series Data Sheet: DC and AC Switching Characteristics* (DS956). The DDRMC configuration is performed using NoC IP and the dependent power parameters are computed using NoC compiler. For more information, see *Versal ACAP Programmable Network on Chip and Integrated Memory Controller LogiCORE IP Product Guide* (PG313). The following figure shows the DDRMC Power View.

Figure 52: DDRMC Power View



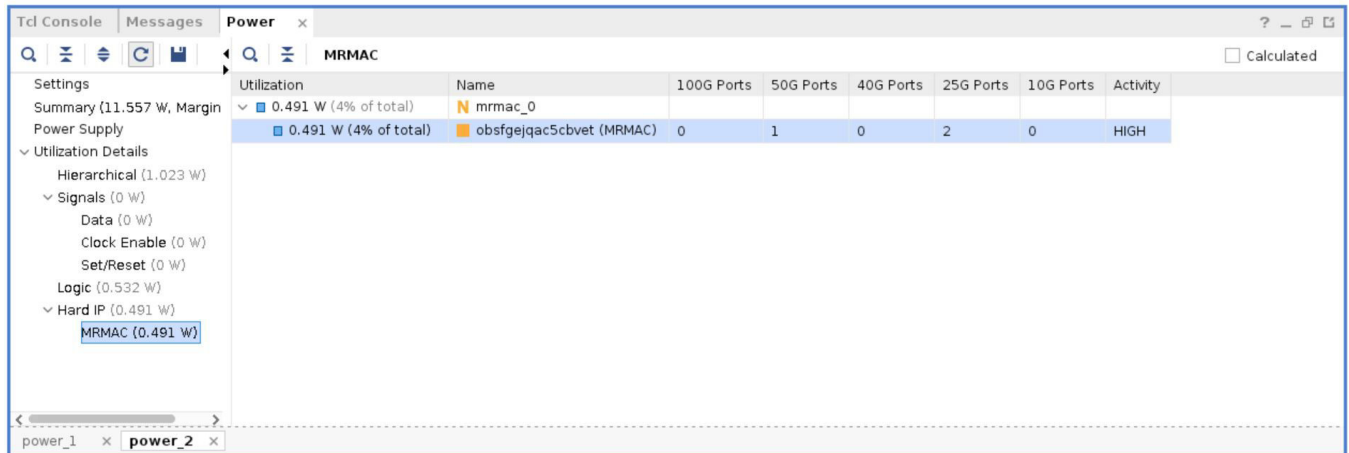
Utilization	Name	Clock (MHz)	Standard	ECC	Channels	DataRate (Mbps)	DataWidth	Read Bandwidth (Mbps)	Write Bandwidth (Mbps)	Command Bus Option
0.046 W (<1% of total)	design_1_wrapper									
0.046 W (<1% of total)	u_ddrmc_main (DDRMC)	1600.000	DDR4_SDRAM	Disabled	1	3200.000	64	100.000	125.000	Highest Data Rate

Note: NoC and DDRMC power can be reported during the post synthesis and post implementation stages of the design. In the project flow, Xilinx® suggests that you should use the Tcl command `update_noc_qos` after opening a synthesized or implemented design to report NoC and DDRMC Power.

MRMAC

Versal™ ACAP devices have an integrated 100G Multirate Ethernet MAC (MRMAC), a high performance, low latency, adaptable Ethernet integrated hard IP targeted for networking applications. This block can be configured for up to four ports with independent MAC and PHY functions according to the IEEE Standard MAC Rates from 10GE to 100GE, and at an overall maximum bandwidth of 100GE. The IP supports various FECs and IEEE 1588 Standards for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems (IEEE 1588) hardware time-stamping. The total power (combined for all 4 ports) is reported as MRMAC Power under the Hard IP's section. For more information, see *Versal Devices Integrated 100G Multirate Ethernet MAC (MRMAC) LogiCORE IP Product Guide* (PG314).

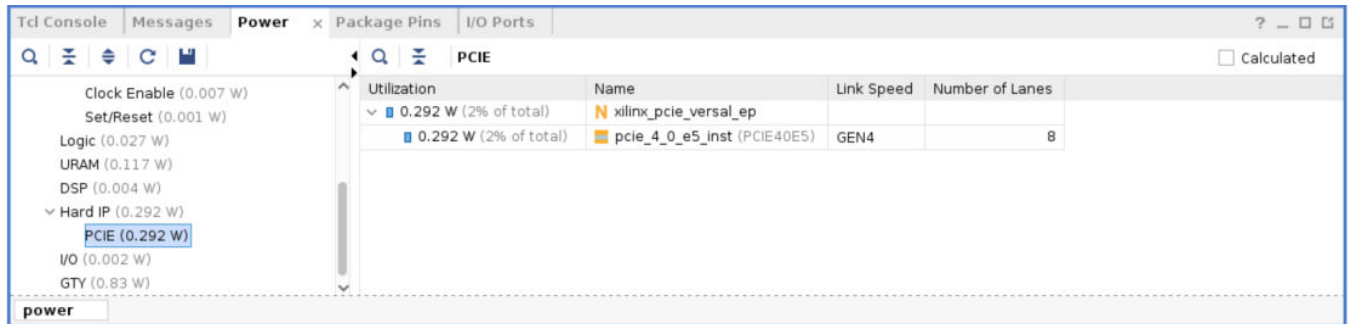
Figure 53: MRMAC View



PCIe

Versal™ ACAP devices have a dedicated PCIe® core in the MAC column, denoted as PCIe. It is a standalone Gen4x8 core without an embedded DMA engine. The core supports Gen1, Gen2, Gen3 and Gen4 line rates. The link widths are x1, x2, x4, x8 or x16 link widths (x16 configuration supported only for Gen1-3 speeds). The combination of PCIe® block, block RAMs/UltraRAMs, GTs, and fabric clocking implements all three layers of the PCI Express protocol, which are the physical layer, data link layer and transaction layer. PCIe Power is reported under the Hard IP's section. For more information, see *Versal ACAP Integrated Block for PCI Express LogiCORE IP Product Guide* (PG343).

Figure 54: PCIe View



DSP

Unlike previous generation DSP blocks, the Versal™ ACAP DSP block can be implemented as a complex multiplier and a floating-point adder and multiplier. It can implement a wide variety of arithmetic and logic functions like previous generation DSP blocks. DSP operations which are reported as DSP Mode in power report are listed here. For more information, see *Versal ACAP DSP Engine Architecture Manual* ([AM004](#)).

- **INT24:** Legacy mode is compatible with the DSP48 from previous generations. INT24 indicates that the DSP block is configured as a 27x24 signed, fixed point multiplier.
- **INT8:** DSP58 uses the Vector Fixed Point ALU mode in this configuration. This mode is used for computing three-element 9x8 vector dot products with accumulate or post add options.
- **CINT18:** This mode indicates that two adjacent DSP58 blocks are configured to implement an 18-bit complex multiplier.
- **FP32:** In this mode, DSPFP32 primitive is used as a floating-point multiplier and adder. This mode is used for FP32 single precision or FP16 half precision with accumulate or post add options.

Note: It is recommended to re-target your existing designs to DSP58 for taking complete advantage of the architectural improvements in the DSP58 blocks.



TIP: DSP output toggle rate does not change much with the data input toggle rate. It mainly depends on specific DSP operations (Multiply-Only, Multiply-Add, and Multiply-Accumulate) and various DSP modes (INT24, INT8, CINT18, and FP32).

The following figure shows report_power results for DSP primitive:

Figure 55: DSP View

Utilization	Name	Clock A (MHz)	Clock Name	DSP No...	DSP Config	MULT Used?	Multiplier Pipeline Used?	Pre-Adder Used?	AD Reg Used?	Signal Rate
0.553 mW	accum_reg	100.000	clk_BUF_BUF0	INT0	3x(S&E) Dot-Product + 50	Yes	Yes	No	No	20.000

PS-PMC

Versal™ devices feature a Control Interface and Processing System IP core. This subsystem is used to configure PL, NoC, and AI Engine. The Control Interface & Processing System is divided into the following power domains. These power domains can operate independently.

- **Platform Management Controller (PMC):** PMC is always on. It is used for device configuration and management. The PMCIOS are a part of this domain. PMC is powered by VCC_PMC rail.
- **Full Power Domain (FPD):** This domain consists of application processing units (APUs) such as Dual A72 Core, L2 Cache, FPD Interconnect, and CCIX. FPD is powered by VCC_PSFP rail.
- **Low Power Domain (LPD):** This domain consists of real-time processing units such as Dual R5, TCM, OCM, and LPD Interconnect. LPD is powered by VCC_PSLP. PSIOs (Dual GEM, USB, and other IO's) are a part of this domain. The IO supply required for the LPD is VCCO_502 and it depends on the type of PSIO used.

Following are the parameters which can be modified using Control Interface and Processing System (CIPS) IP core configuration wizard in Vivado.

- PMC
 - PMC subsystem clock frequency and peripheral IO clocks can be modified.
- FPD
 - A72 operation clock and FPD interconnect clocks.
- LPD
 - Dual R5 Operating Clock and LPD Interconnect Clocks.
 - PS IO clocks like GEM and USB. IO standards for each bank can also be configured.

For more information, see *Control, Interface and Processing System LogiCORE IP Product Guide* (PG352).

Power numbers reported by report_power are for a default operating load of 90% for all processing systems including A72, R5, and PMC (also the interconnects). For an IO power estimate, IOs need to be configured from the IP configuration wizard, if not it reports zero power for the IOs. The power reported for a PS instance is the sum of all the PS rails (VCC_PSFP, VCC_PSLP, and IOs).

Figure 56: Control Interface and Processing System IP Core Configuration Wizard

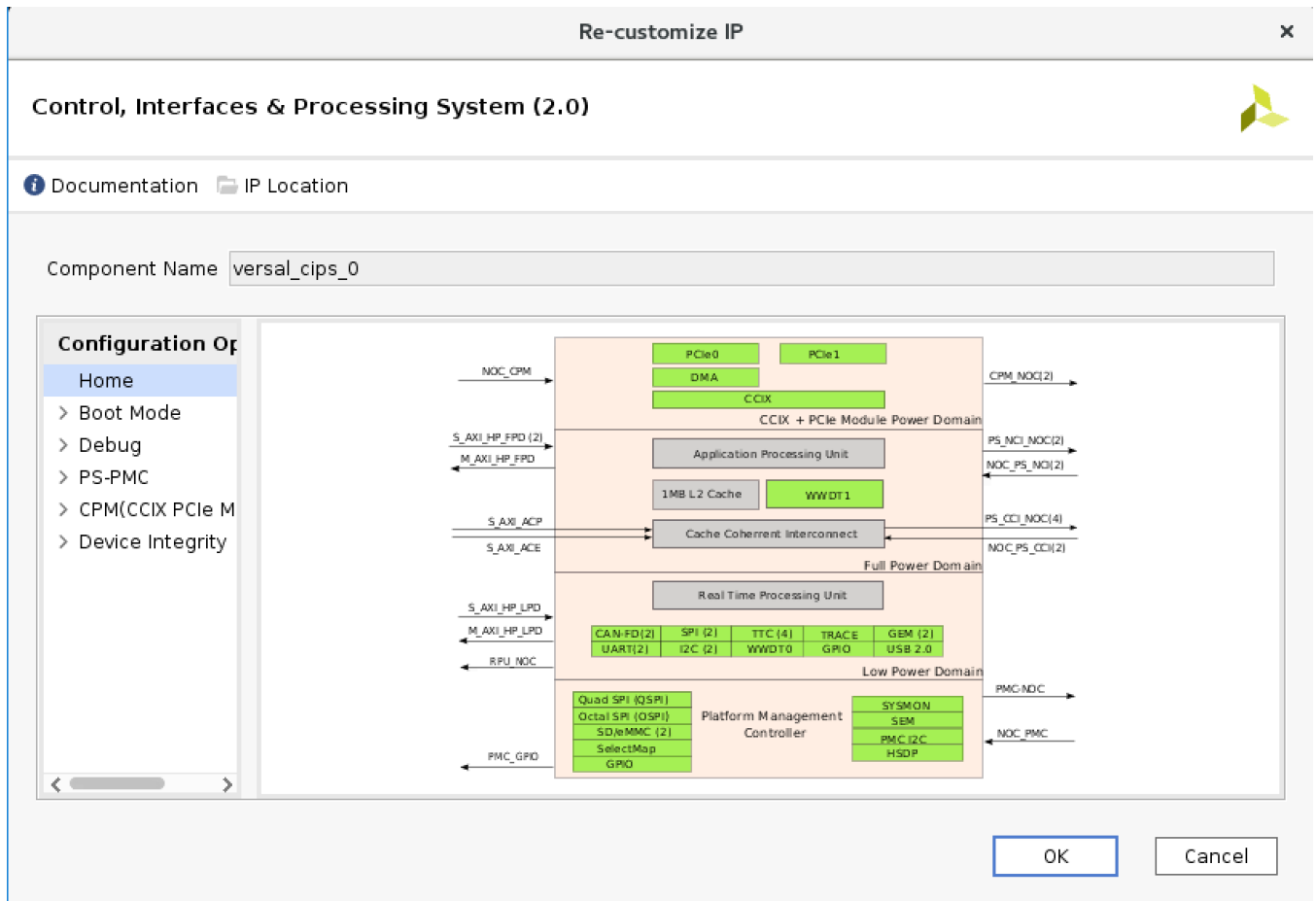


Figure 57: Control Interface & Processing System IP Core Clock Configuration

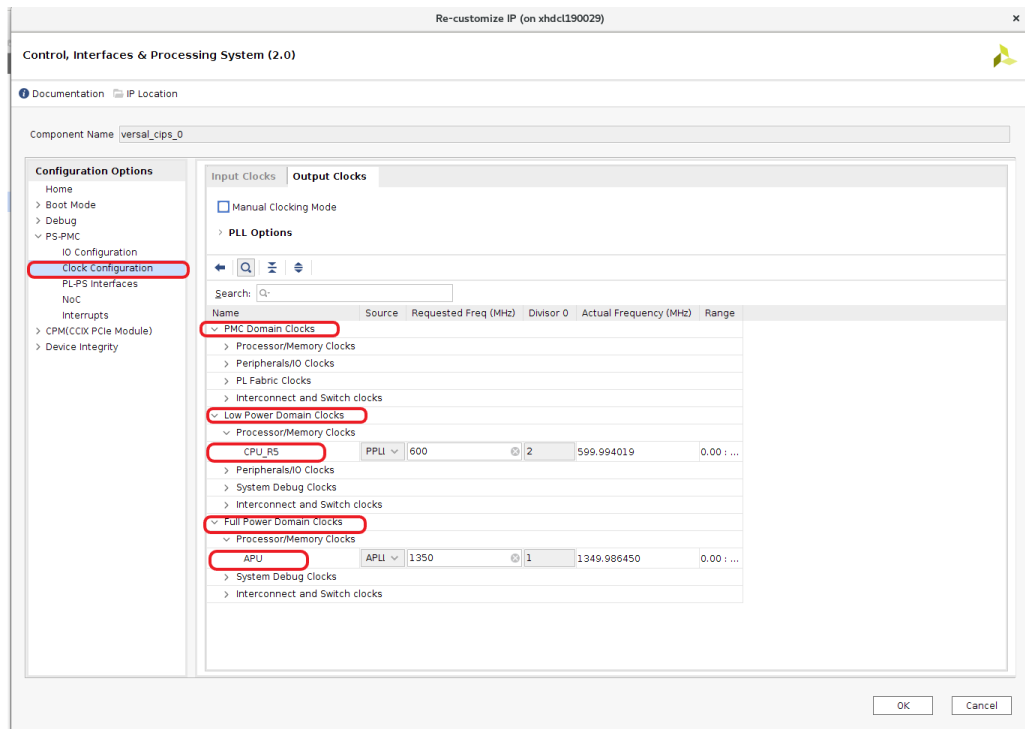
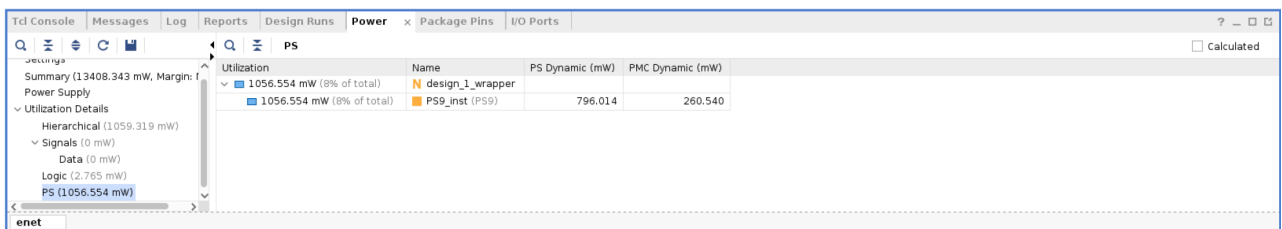


Figure 58: PS View



IO

Versal™ device IOs are categorized into the following two categories:

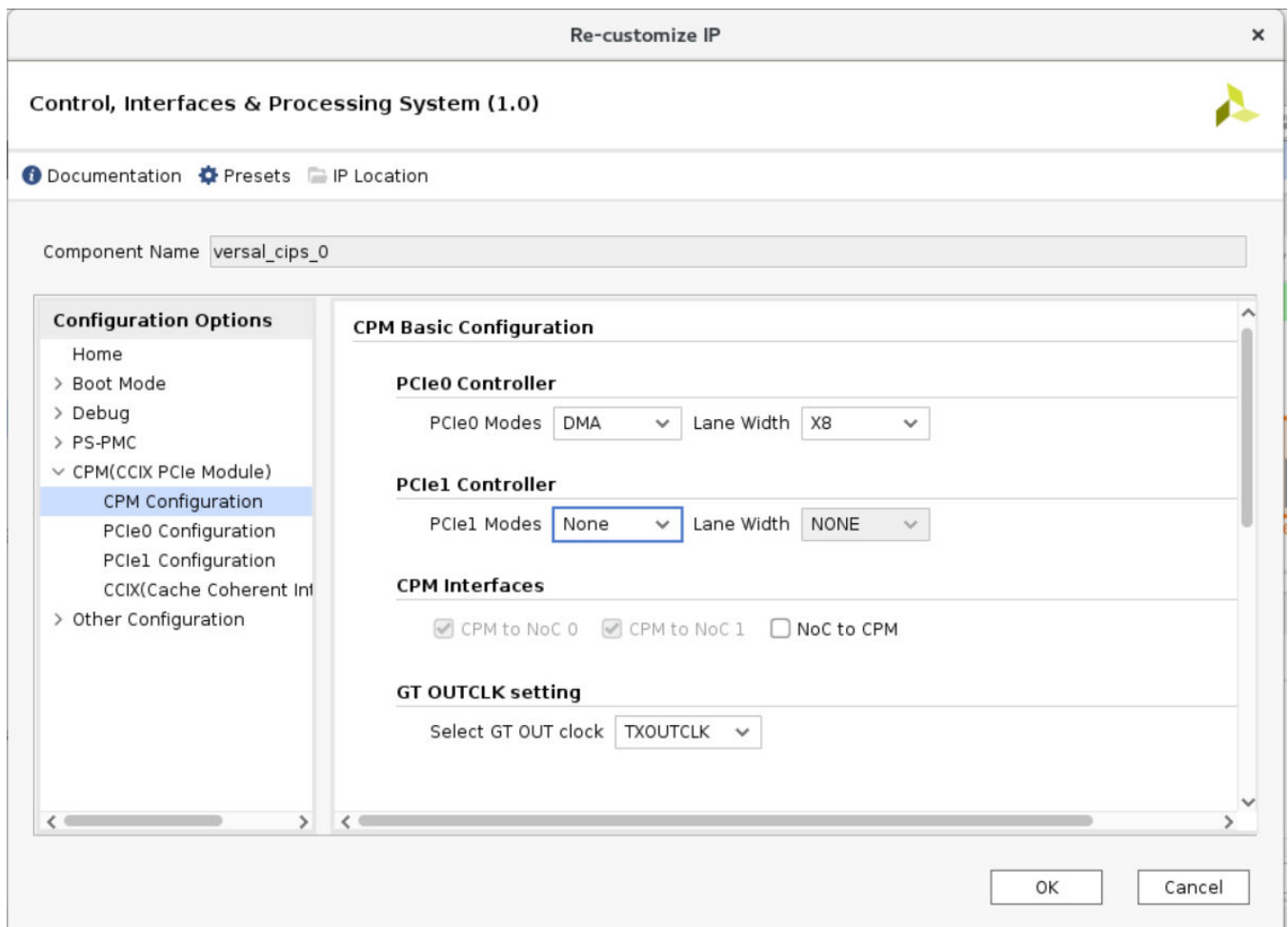
- **HDIO:** It is designed as a high density IO that is cost effective and flexible for implementation. XPIO is designed to replace HPIO (UltraScale+™) and PS dedicated DDR memory interface IO (PS-DDRIO).
- **XPIO:** It supports IO standards with maximum VCCO of 1.5V. HDIO is intended to cover IO standards not offered by XPIO with maximum VCCO beyond 1.5V. It is important to define proper IO standard for an accurate on-chip and off-chip power estimation for IO interfaces. If an IO standard is not defined then zero power is reported.

For more information, see *Versal ACAP SelectIO Resources Architecture Manual* ([AM010](#)).

CPM

CCIX PCIe Module (CPM) in a Versal™ device is an independent subsystem. CPM contains a Type-A PCIe® (Gen-4x16) controller that contains two instances of PCI Express controller cores. It has a hardened component to allow a fabric accelerator to act as a cache coherent interconnect for accelerators (CCIX) compliant accelerator. PCIe®-A block interfaces with GTs through the XPIPE interfaces. The controller supports Gen1, Gen2, Gen3, Gen4 PCIe® modes, up to 16 lanes. It also supports a CCIX only ESM mode (20 or 25 Gb/s). For more information, see *Versal ACAP CPM CCIX Architecture Manual* ([AM016](#)). CPM can be configured in the IP integrator from the Control, Interfaces and Processing System IP as shown in the following figure.

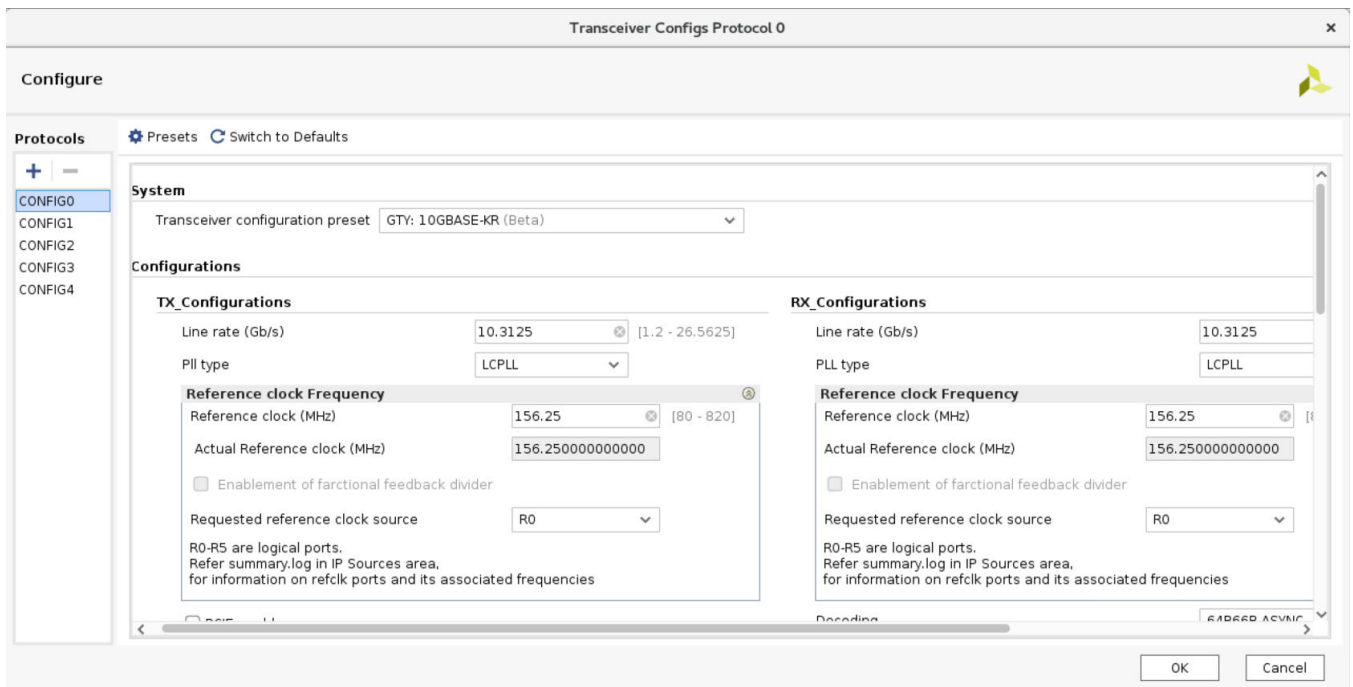
Figure 59: CPM Configuration



GTY

The GTY transceiver performs basic serialization and deserialization functions. The highest supported data rate is 32.75 Gb/s and the lowest supported data rate is 1.2 Gb/s. Versal ACAP transceivers wizard IP supports 16 user configurations per lane. For more information, see *Versal ACAP GTY and GTYP Transceivers Architecture Manual (AM002)*.

Figure 60: Transceiver Configuration Protocol Dialog Box



Power is reported for any specific configuration depending on CH*_TXRATE and CH*_RXRATE port connections. If CH*_TXRATE and CH*_RXRATE are connected to some logic other than GND/VCC, the configuration with worst case power among all configurations is reported.

Figure 61: GTY

Utilization	Channel	Operational Mode	PLL Sharing	Power Mode	RX Clock Source	RX Data Rate (Gbps)	RX Data Path Width	TX Clock Source	TX Data Rate (Gbps)	TX Data Path Width	TX Data Mode	TX QP Amp (mV)	Vccint (mW)	MGTYVccaux (mW)	MGTYVcc (mW)	MGTYVAT (mW)	
2111.505 mW																	
527.876 mW	gt_qua...	TRANSCEIVER	Yes	DFE	LCPLL	24.330	64	Raw	LCPLL	24.330	64	Raw	250.000	116.854	0.204	105.969	304.849
527.876 mW	gt_qua...	TRANSCEIVER	Yes	DFE	LCPLL	24.330	64	Raw	LCPLL	24.330	64	Raw	250.000	116.854	0.204	105.969	304.849
527.876 mW	gt_qua...	TRANSCEIVER	Yes	DFE	LCPLL	24.330	64	Raw	LCPLL	24.330	64	Raw	250.000	116.854	0.204	105.969	304.849
527.876 mW	gt_qua...	TRANSCEIVER	Yes	DFE	LCPLL	24.330	64	Raw	LCPLL	24.330	64	Raw	250.000	116.854	0.204	105.969	304.849

Tips and Techniques for Power Reduction

Introduction

This chapter describes power reduction techniques and their expected effect on total device power. This information will help you evaluate your best options depending on your time, power budget, available resources, and freedom to change your design.

System-Level Power Reduction

Cooling Strategy

Cooling strategy ensures that heat generated from the device is extracted and absorbed by the environment. These cooling strategies, which are generally available at the beginning of the design and become less feasible in the later stages, have a significant impact on the device static power:

- Increase the airflow.
- Lower the ambient temperature.
- Use a heat sink (or a larger heat sink), or select a different regulator.

Supply Strategy

Voltage has a large effect on both static and dynamic power. Active control of the voltage level ensures the desired voltage is applied to the device.

- Use switching regulators.
Switching regulators are more power efficient than linear regulators, at the expense of requiring a higher component count.
- Use adjustable regulators.

Ensure voltage is as close as possible to the device and to the highest consuming device if the same supply powers multiple devices.

- Select regulators with tight tolerances.

Device Selection

- Select the best device for the product.

Increasingly, power is becoming one of the primary factors for selecting a device. Select the device that best meets your density, functionality, and performance requirements and will also meet your power budget.

- Minimize the number of devices.

This saves space, I/O interconnect power, total leakage, and other factors. Typically, replacing multiple components (for example, processor and device) with a single larger device consumes less static power.

- Select the smallest device possible.

This reduces leakage. Typically in a device family the same package may be available with different die sizes. You can, for instance, use a larger die during the prototyping and pre-series phase, then move to a smaller die for volume production.

- Select the largest package possible.

This increases heat dissipation. A larger package has a larger area to dissipate the die heat into the environment. A larger heat sink can be attached to the package upper side and more heat can escape onto the PCB via the bottom ball grid array.

- Use low voltage devices.

Some device families are available with a lower power option. The lower core voltage requirements translate into significant static and dynamic power savings.

- Use low leakage devices.

Some device families are available with a lower leakage or static power options in the form of specific speed or temperature grades. These devices may cost a bit more to purchase but you or the end user may be able to more than offset this with savings on the electricity bill or cooling hardware and system maintenance.

Measuring Power and Temperature

This section describes the techniques for measuring a device power consumption and heat dissipation. Some of these techniques use internal device resources. Other techniques use board or external components. Some applications require power and temperature to be actively monitored and adjusted after deployment. Other applications use these measurement techniques in the lab during prototyping and validation phases.

Power Measurement Techniques

Power measurement techniques include:

- [Using a Current Sense Resistor](#)
- [Using Advanced Regulators and Digital Power Controllers](#)
- [Performing On-Board Monitoring](#)
- [Having Separate Voltage Rails](#)

Using a Current Sense Resistor

Inserting a Current Sense Resistor in series between the regulator output and the device creates a small voltage drop which, by Ohm's Law, is proportional to the flowing current. Measuring this voltage through an XADC/SYSMON gives you the current being supplied to the device. To understand the connections needed to obtain the desired accuracy of measurements, refer to the user guide of the respective device family, *Versal ACAP System Monitor Architecture Manual (AM006)*, *UltraScale Architecture System Monitor User Guide (UG580)*, and *7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide (UG480)* (also known as the XADC User Guide). See *Driving the Xilinx Analog-to-Digital Converter (XAPP795)* for more information on how to use a current sense resistor.

Using Advanced Regulators and Digital Power Controllers

The latest evaluation kits include advanced regulator and digital power controllers that you can use to capture regulator output currents and voltages, then send this information to a monitoring computer over a USB interface. This is the simplest and most convenient way to monitor the power rails. Most Xilinx® development boards have integrated power controllers that can be accessed with a GUI software on a PC using a PMBus (I2C) to USB interface module.

Performing On-Board Monitoring

Xilinx® 7 series and above device families provide internal sensors and at least one analog-to-digital converter to measure supplied voltages and device temperature. The Vivado® hardware manager provides real-time JTAG access to measure the different supply source voltages or device junction temperature before and after device configuration. You can also instantiate a System Monitor or XADC component in your code to access these measurements from your device application.

Having Separate Voltage Rails

When possible, have separate voltage rails for each of the supply voltages. If voltage rails are tied together, note it and account for it when power is measured across these rails.

Thermal Measurement Techniques

Thermal measurement techniques include:

- [Performing External Monitoring](#)
- [Performing On-Board Monitoring](#)

Performing External Monitoring

Because the device package prevents access to the silicon, junction temperature cannot be measured directly. Junction temperature can be approximated by measuring the temperature of the package, the heat sink, and other locations with a thermocouple. Thermal cameras are also used to visualize the device temperature and thermal dissipation interactions with neighboring components and the larger environment.

Performing On-Board Monitoring

Thermal measurements are possible using the same techniques as power measurements. You can use the Vivado® hardware manager before and after device configuration. You can also use the System Monitor/XADC primitive within your design to read the device junction temperature.

Methodology for Power and Temperature Measurement

To evaluate the three factors contributing to the design total power, you must control the device junction temperature and let it stabilize before making measurements. This control and stabilization is required because the device and design static power is heavily dependent on the device junction temperature. The three factors contributing to the design total power are:

- [Device Static](#)
- [Design Static](#)
- [Design Dynamic](#)

Device Static

Download a blank design to ensure that: (1) no input noise is captured; and (2) all internal logic and configuration circuits are in a known state.

Note: A blank design is a design with a single gate or flip-flop that never toggles, and in which all outputs are in a 3-state configuration.

Wait for the junction temperature to stabilize, then measure VCCINT, VCCAUX, and any other supply source of interest. With special equipment, a simple heat gun, or cold spray, you can force temperature changes to evaluate the influence of the environment on the device static power. VCCADC should always be connected to VCCAUX.

Design Static

Download the design onto the device and do not start any input or internal activity (input data and external and internal clock generation). Wait for the device temperature to stabilize, then measure power on all supply rails of interest. Subtracting the device static measurement from these values gives you the additional static power from the specific logic resources and configuration used in your design (design static power).

Design Dynamic

Download the design onto the device and provide clocks and input stimulus representative of the design. Wait for the junction temperature to stabilize before measuring all supply sources of interest. This power represents the instantaneous total power of the design. It will vary with the change in activity at each clock cycle.

Design Level Power Reduction

The following sections describe tips and techniques that can be applied to the design to meet your design's power budget. Your design cycle will include a power closure phase under two main circumstances:

- You want to further optimize your design after constraints are met.
- Or
- Your design has exceeded its power budget.

Further Optimizing the Design After it Meets Constraints

Typically at this stage in the development process you want to minimize changes to the RTL, board power supply, and cooling parameters, because this involves a lot of verification or PCB respin costs. However, you can experiment with different software options and constraints to optimize your logic and routing resource counts, configuration, and activity. This minimizes dynamic power and reduces static power at the same time. Depending on your design margins a 15% to 20% savings for your core dynamic power is a reasonable expectation, with some designs showing even more power reduction.

My Power Budget is Exceeded! What Can I Do?

Typically at this stage the pressure to get the system to market is getting high and many parameters in your system are well defined, such as the board environment and cooling options. Even though this restricts the type of engineering rework you can do, the following methodology should help identify and focus on the highest potential areas for power reduction.

Step 1: Which Power Budget is Exceeded?

GUI users can review the Summary view in the Vivado® Power Analysis report, and command line users can use the Summary section of the report file. The On-Chip and Supply Power tables provide a high-level view of the power distribution. Navigate the Summary view to determine the type and amount of power that exceeds your budget.

Step 2: Identify the Area on Which to Focus

Review the different detailed views in the Vivado® Power Analysis report or Xilinx® Power Estimator. Analyze the environment parameters and the power distribution across the different resources used, the design hierarchy, and clock domains. When you find an area of the design where power seems high, the information following should help you determine the likely contributing factors.

Step 3: Experiment

With the list of candidate areas in your design for power optimizations derived from the previous step, you can now sort this list from easiest to most involved and decide which optimization or experiment to perform next. The power tools allow you to do What If? analysis so you can quickly enter design changes and estimate the power implications without having to actually edit any code or constraint or rerun the implementation tools.

Use Device Resources More Efficiently

- **Block RAM:**

- The amount of power block RAM consumes is directly proportional to the amount of time it is enabled. To save power, the block RAM enable can be driven Low on clock cycles when the block RAM is not used in the design. Block RAM Enable Rate, along with Clock rate, is an important parameter that must be considered for power optimization.
- Use the NO_CHANGE mode in the TDP mode if the output latches remain unchanged during a write operation. This mode is the most power efficient. This mode is not available in the SDP mode because it is identical in behavior to WRITE_FIRST mode.
- **I/O:** I/O interfaces have to drive long distances with potentially more parasitic effects, hence they typically represent a large portion of the device power requirements.
 - **VCCAUX:** Use the lowest VCCAUX possible. This minimizes both the static and dynamic power for this voltage supply.
 - **Inputs:** Limit usage of internally referenced input standards.
 - **IODELAY:** Set the HIGH_PERFORMANCE_MODE property on the IDELAY2 to FALSE. When FALSE, this property increases the output jitter, but consumes less power.
 - **IBUF_LOW_PWR:** Set the IBUF_LOW_PWR property to TRUE on bidirectional and input I/Os. Make sure the design performance allows for this setting.
 - **I/O Configuration:** Review the I/O standard, drive strength, and on-chip termination settings in the context of your performance needs and evaluate if you can use lower drive strength using tristatable DCI I/O standards (T_DCI), get by without terminations, or use external terminations.
 - **Outputs:**
 - Use the lowest slew/drive/voltage level supported by the receiving chip(s).
 - No termination or series termination are preferred over parallel terminations. Signal integrity simulation tools can help with this determination.
 - Consider whether using on-chip or off-chip termination is the best option given your device thermal budget, system cost, and board real estate requirements.
 - Evaluate using lower voltage swing differential standards.
 - Evaluate if your application allows you to use transceivers instead of large parallel busses.
 - Evaluate the requirements of I/O features such as IBUF, IO DELAY, and others, and disable when performance allows.
- **Transceivers:**
 - The GTX/GTH/GTP transceiver supports a range of power-down modes that may save power if applicable.

- There are two types of adaptive filtering available to the GTX/GTH receiver depending on system level trade-offs between power and performance. Optimized for power with lower channel loss, the GTX/GTH/GTP receiver has a power-efficient adaptive mode named the low-power mode (LPM).
- Each GTX/GTH/GTP transceiver provides support for generating the out-of-band (OOB) sequences described in the Serial ATA (SATA), Serial Attach SCSI (SAS) specification, and beaconing described in the PCI™ Express specification. If OOB sequence is not used, this could further save power.
- Pack the maximum number of transceivers into a single tile to minimize duplicating supporting circuits.
- **XADC:**
 - The XADC can be powered down by writing to its Configuration register #2 (Address 0x42) from the DRP port during run time. Bits DI4 and DI5 of this register control the power-down for each channel. To statically emulate power-down behavior in Vivado®, the configuration registers can be set by entering this command in the Vivado Tcl console:

```
set_property INIT_42 {16'h0430} [get_cells <inst>]
```

where `<inst>` is the XADC instance. The above command powers down both channels of the XADC.

- **Logic:**

You can optimize the design description using these methods:

- Minimize asynchronous control signals which prevent logic optimization and use more routing resources.
- Minimize the number of control sets. A control set consists of the unique grouping of a clock, clock enable, set, reset, and, in the case of LUT RAM, write enable signals. Control set information is important because count limits or sharing of signals within a slice may occur. This varies with the device architecture, and when the limit is reached can prevent proximity packing of related logic, which would increase routing resources.
- Add pipeline levels to minimize the size of combinatorial logic cones. This minimizes the propagation of glitches between registers until signals reach their final state at each clock cycle.
- Use resource time sharing. These techniques minimize device resource usage by time multiplexing different functions to the same hardware resources. This allows you to use a smaller device or can reduce placement and routing congestion, which will lower both static and dynamic core power.

- Processes which are slow and similar can be performed on the same resources instead of separate resources. This requires careful thinking for how to buffer, multiplex, initialize, and control the data to be processed. Typical applications for such optimization are similar parallel processes, such as processing multiple input sensors. Instead of having as many processing units as inputs, you could use a single processing unit and make it run faster, so it processes input channels one after the other while ensuring the same response time for each output. A Xilinx® Power Estimator What If? estimation can help you decide whether the power savings are worth the engineering effort.
- Use the DSP and block RAM optional registers. For example, in DSP blocks the multiplier or MREG registers, when enabled, are the most power efficient implementation as they minimize the propagation of internal glitches between clock cycles.

Experiment Using the Vivado Power Optimizer Feature

To maximize power savings when you run the power optimizer in the Vivado® tools, you should run power optimization on the entire design and not exclude portions of the design. If you do not see anticipated power savings after enabling power optimization, the three areas of potential debug are the global set and reset signals, block RAM enable generation, and register clock gating. A low number of power optimization generated enables in this area could indicate the need to review coding practices or options/properties set for design synthesis and implementation.

Note: In the Vivado tools, power optimization works to minimize the impact on timing while maximizing power savings. However, in certain cases, timing may degrade after power optimization. For techniques to offset this impact, see [Preserving Timing After Power Optimization](#).

- **Global set and reset signals:**

Where possible, minimize the use of asynchronous set/reset signals especially to datapath or pipeline flip-flops as well as block RAMs. You should also consider constraining the global set and reset signals as `dont_touch` during the `power_opt_design` step to avoid their use as enables. Note that setting the `dont_touch` property in HDL will cause every step in the flow to obey this property. It is recommended that this option is set up as an XDC constraint only for the power optimization step. Here is an example of how to do this:

```
set_property DONT_TOUCH true [get_cells u1]
```

Finally, ensure that the signal rate and probabilities of the global set and reset signals are set correctly prior to running power optimizer and vectorless power estimation.

- **Slice registers and SRLs:** A number of different reasons could explain why `power_opt_design` might not be able to generate clock enables for slice registers or SRLs in the design. Some examples are:
 - Having combinational loops in the design
 - Using set/reset signals at the flip-flops and SRLs that are sourced from primary inputs to the design

- Using asynchronous set/reset signals at the datapath flip-flops
- Large number of clock domains in the design preventing enables being generated due to clock domain crossing issues
- SRL sizes: Typically the larger the number of shift register stages in the SRLs, the more difficult it is to generate a single clock enable for all stages
- **Block RAMs:** Block RAM rich designs are excellent candidates for power savings. Vivado uses a variety of optimization techniques to generate enables and save power. If block RAM gating coverage is low after using `power_opt_design`, some of the possible reasons could be:
 - Block RAMs are mainly FIFO18/FIFO36 cells. These cannot be optimized by the tool.
 - Memories inferred or instantiated are mainly in true dual port (TDP) mode using asynchronous clocks on their A and B ports that cannot be optimized by `power_opt_design`.
 - Use of asynchronous reset signals to either the block RAMs themselves or to the address/write-enable flip-flops feeding the block RAMs.

Experiment within the Vivado Power Analysis Feature

In the Vivado® Report Power dialog box you can make adjustments then rerun the analysis to review the power implications for these factors:

- **Environment:** Includes thermal parameters, process, or voltages.
- **Design Activity:** Adjust the activity of nets or cells in the design. Change one item or change multiple items at a time. You can also change:
 - **Clock domains:** Adjust the switching frequency.
 - **Device logic:** Adjust the dynamic activity rate.
 - **I/Os:** Adjust both static and dynamic activity probabilities. You can also adjust parameters for the external components connected to the device outputs, such as the load capacitance or the near-end board termination details.
 - **Signals:** Adjust the dynamic activity rate for data signals. For control signals you can also adjust the static probability to evaluate power under different Clock Enable, Set, or Reset scenarios.
- **Specific blocks:** In addition to the dynamic activity probability you can also adjust the activity of control ports such as port enables or write enables on block RAMs.

Experiment within Xilinx Power Estimator (XPE)

In XPE you can import the Vivado® power analysis results from modules developed by multiple sources to review the total power once these separate IP blocks are implemented in the device. You can also evaluate situations where you would have to change the netlist, and evaluate the power implications, without having to actually make the code changes. For your design core logic, XPE works at a coarser resolution than the Vivado power analysis, because you cannot adjust each logic element or signal individually in XPE. In XPE, you can also experiment with:

- **Resource usage:** Explore reducing the resource count. Try remapping pieces of logic from slice logic to dedicated blocks such as block RAM or DSP, and vice versa.
- **Resource configuration:** Explore using different configuration settings for the design I/Os, block RAMs, clock generators, and other resources.

Experiment within RTL Code

If you need to modify your RTL code to reduce power you can experiment with adding a pipeline or performing power retiming around high-activity logic such as carry chains and XOR functions. Although long paths with carry chains tend to be on slower clock domains, they exhibit more glitching activity, which increases the design power. Retiming or pipelining these paths is often beneficial.

Step 4: Implement the Changes and Review the Power Saving

Once you have determined the best changes to make given your time, performance, and resource constraints, proceed with implementing them. It is worth mentioning that trying too many options or changes at once may not yield the best results because of potential conflicts or interactions between them. Best practice, if time allows, is to experiment with a few options at a time so you can evaluate their effect on power and other constraints before adding on other changes.

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see [Xilinx Support](#).

Documentation Navigator and Design Hubs

Xilinx[®] Documentation Navigator (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the Vivado[®] IDE, select **Help** → **Documentation and Tutorials**.
- On Windows, select **Start** → **All Programs** → **Xilinx Design Tools** → **DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the [Design Hubs](#) page.

Note: For more information on DocNav, see the [Documentation Navigator](#) page on the Xilinx website.

References

These documents provide supplemental material useful with this guide:

1. *UltraScale Architecture System Monitor User Guide* ([UG580](#))
2. *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* ([UG973](#))
3. *Vivado Design Suite Tcl Command Reference Guide* ([UG835](#))
4. *Vivado Design Suite User Guide: Using Constraints* ([UG903](#))
5. *Xilinx Power Estimator User Guide* ([UG440](#))
6. *Vivado Design Suite Tutorial: Power Analysis and Optimization* ([UG997](#))
7. *Vivado Design Suite User Guide: Logic Simulation* ([UG900](#))
8. *7 Series FPGAs Packaging and Pinout Product Specification* ([UG475](#))
9. *UltraScale and UltraScale+ FPGAs Packaging and Pinouts Product Specification* ([UG575](#))
10. *7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide* ([UG480](#))
11. *Versal ACAP AI Engine Programming Environment User Guide* ([UG1076](#))
12. *Versal ACAP Design Guide* ([UG1273](#))
13. *Xilinx Power Estimator User Guide for Versal ACAP* ([UG1275](#))
14. *Versal ACAP Programmable Network on Chip and Integrated Memory Controller LogiCORE IP Product Guide* ([PG313](#))
15. *Versal ACAP Integrated Block for PCI Express LogiCORE IP Product Guide* ([PG343](#))
16. *Control, Interface and Processing System LogiCORE IP Product Guide* ([PG352](#))
17. *Versal Prime Series Data Sheet: DC and AC Switching Characteristics* ([DS956](#))
18. *Versal ACAP GTY and GTYP Transceivers Architecture Manual* ([AM002](#))
19. *Versal ACAP DSP Engine Architecture Manual* ([AM004](#))
20. *Versal ACAP System Monitor Architecture Manual* ([AM006](#))
21. *Versal ACAP Packaging and Pinouts Architecture Manual* ([AM013](#))
22. *Versal ACAP CPM CCIX Architecture Manual* ([AM016](#))
23. *Seven Steps to an Accurate Worst-Case Power Analysis using the Xilinx Power Estimator* ([XAPP1348](#))
24. *Driving the Xilinx Analog-to-Digital Converter* ([XAPP795](#))
25. [Vivado Design Suite Documentation](#)

Training Resources

Xilinx® provides a variety of training courses and QuickTake videos to help you learn more about the concepts presented in this document. Use these links to explore related training resources:

- [Vivado Design Suite QuickTake Video: Power Estimation and Analysis](#)
- [Vivado Design Suite QuickTake Video: Power Optimization](#)
- [Vivado Design Suite QuickTake Video Tutorials](#)

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 <https://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 <https://www.xilinx.com/legal.htm#tos>.

This document contains preliminary information and is subject to change without notice. Information provided herein relates to products and/or services not yet available for sale, and provided solely for information purposes and are not intended, or to be construed, as an offer for sale or an attempted commercialization of the products and/or services referred to herein.

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

© Copyright 2012-2021 Xilinx, Inc. Xilinx, the Xilinx logo, Alveo, Artix, Kintex, Spartan, Versal, Virtex, Vivado, 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.