Intel® Quartus® Prime Standard Edition User Guide

Power Analysis and Optimization

Updated for Intel® Quartus® Prime Design Suite: **18.1**

Subscribe
Send Feedback

UG-20184 | 2018.09.24
Latest document on the web: PDF | HTML

Contents

Send Feedback

1. Power Analysis

The Intel® Quartus® Prime Design Suite provides tools to estimate power consumption in a FPGA design at different stages of the design process. The Intel Quartus Prime Power Analyzer allows you to estimate power consumption for a post-fit design. To estimate the power consumption before you compile the design, use the Early Power Estimator (EPE) spreadsheet.

Note: Do not use the results of the Power Analyzer as design specifications.

As designs grow larger and process technology continues to shrink, power becomes an increasingly important design consideration. When designing a PCB, you must estimate the power consumption of a device accurately to develop an appropriate power budget, and to design the power supplies, voltage regulators, heat sink, and cooling system.

The Intel Quartus Prime Power Analyzer helps you establish guidelines for the power budget of your design. Make sure to verify the actual power during device operation, because this information is sensitive to the actual device design and the environmental operating conditions.

This chapter describes the Intel Quartus Prime Power Analyzer tool. For details about the EPE spreadsheets, refer to the Early Power Estimator page in the Altera website.

Note: The Intel Quartus Prime Power Analyzer does not support the Intel Arria® 10 HPS IP. You can obtain a power estimation for this Intel FPGA IP with the EPE spreadsheet.

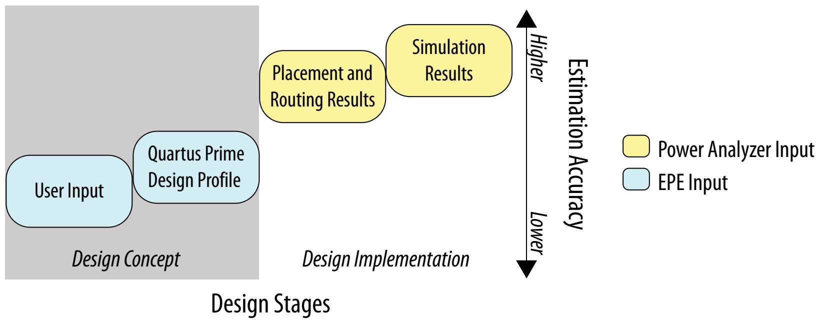
Related Information

- Early Power Estimator Page
- Power Analyzer Support Resources

1.1. Comparison of the EPE and the Intel Quartus Prime Power Analyzer

The following figure shows the design stages in which you use power analysis tools, and compares the accuracy of the estimations for different input types:

Figure 1. Estimation Accuracy for Different Inputs and Power Analysis Tools



The following table lists the differences between the EPE and the Intel Quartus Prime Power Analyzer.

Table 1. Comparison of the EPE and the Intel Quartus Prime Power Analyzer

Characteristic	EPE	Intel Quartus Prime Power Analyzer
When to use	Any time *Note:* For post-fit power analysis, you get better results with the Intel Quartus Prime Power Analyzer.	Post-fit
Software requirements	Spreadsheet program	The Intel Quartus Prime software
Accuracy	Medium	Medium to very high
Data inputs	• Resource usage estimates • Clock requirements • Environmental conditions • Toggle rate	• Post-fit design • Clock requirements • Signal activity defaults • Environmental conditions • Register transfer level (RTL) simulation results (optional) • Post-fit simulation results (optional) • Signal activities per node or entity (optional)
Data outputs *Note:* The EPE and Power Analyzer outputs vary by device family.	• Total thermal power dissipation • Thermal static power • Thermal dynamic power • Off-chip power dissipation • Current drawn from voltage supplies	• Total thermal power • Thermal static power • Thermal dynamic power • Thermal I/O power • Thermal power by design hierarchy • Thermal power by block type • Thermal power dissipation by clock domain • Off-chip (non-thermal) power dissipation • Device supply currents
Estimation of transceiver power for dynamic reconfiguration features	Includes an estimation of the incremental power consumption by these features.	Not included

1.2. Power Estimations and Design Requirements

Power estimation and analysis helps you satisfy two important planning requirements:

- **Thermal**—Thermal power is the power that dissipates as heat from the FPGA. Devices use a heatsink or fan to act as a cooling solution. This cooling solution must be sufficient to dissipate the heat that the device generates. Additionally, the computed junction temperature must fall within normal device specifications.
- **Power supply**—Power supply is the power that the device needs to operate. Power supplies must provide adequate current to support device operation.

  Note: For power supply planning, use the EPE at early stages of the design cycle. When the design is complete, you can use the Power Analyzer reports for an estimate of design power requirement.

Thermal and supply power requirements are similar, but not identical, because there are elements outside the device that also contribute to power dissipation, such as terminator resistors.

1.3. Power Analyzer Walkthrough

The Intel Quartus Prime Power Analyzer requires post-fit design.

You must either provide timing assignments for all clocks in the design, or generate activity data from a simulation-based flow. You must specify the I/O standard on each device input and output, and the board trace model on each output in the design.

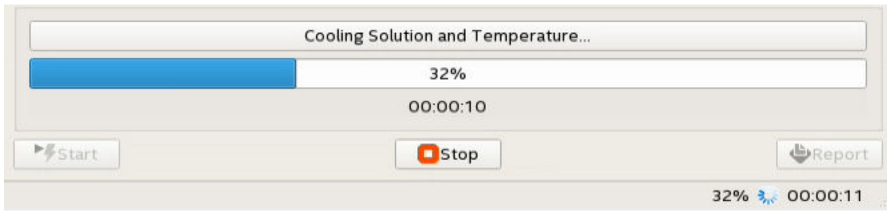
To run the Power Analyzer:

1. From the Intel Quartus Prime Software, open the Power Analyzer tool by clicking **Processing ➤** **Power Analyzer Tool**.
2. If you have signal activity information for the project, turn on **Use input files to initialize toggle rates and static probabilities during power analysis**, and then click **Add Power Input Files** to specify input files.
   For more information about generating those input files, refer to *Sources for Signal Activity Data*.
3. To direct the Power Analyzer to write a Signal Activity (`.saf`) output file, turn on **Write out signal activities used during power analysis**, and specify the file name.
4. To direct the Power Analyzer to generate an Early Power Estimation file, turn on **Write out Early Power Estimation file**, and specify the file name.

   With this file, you can import design information into the Early Power Estimator spreadsheet, and perform what-if analyses.
5. Specify the default toggle rate for input I/O signals.

   The Power Analyzer uses the default toggle rate when no other method specifies the signal-activity data.
6. Specify the default toggle rate for the remaining (non input) signals.
7. Define the cooling solution and temperature.
8. Click **Start**.

Figure 2. Progress Bar in Intel Quartus Prime Power Analyzer



9. When the tool finishes, click **Report** to open the Power Analyzer report.

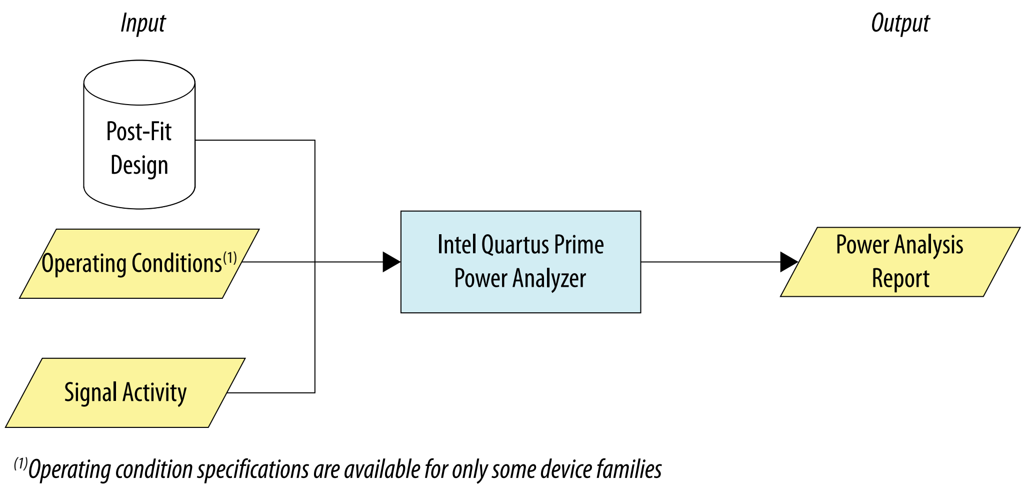
Related Information

Inputs for the Power Analyzer on page 7

1.4. Inputs for the Power Analyzer

The Power Analyzer supports accurate power estimations by allowing you to specify the important design factors affecting power consumption. The following figure shows the high-level flow of the Power Analyzer:

Figure 3. Power Analyzer High-Level Flow


Input
Output
Post-Fit Design
Operating Conditions(1)
Signal Activity
Intel Quartus Prime Power Analyzer
Power Analysis Report
(1)Operating condition specifications are available for only some device families

To obtain accurate I/O power estimates, the Power Analyzer requires you to synthesize the design and then fit the design to the target device. Additionally, you must specify:

- The electrical standard on each I/O cell.
- The board trace model on each I/O standard in the design.
- Timing assignments for all the clocks in your design, or use a simulation-based flow to generate activity data.

1.4.1. Operating Settings and Conditions

You can specify device power characteristics, operating voltage conditions, and operating temperature conditions for power analysis in the Intel Quartus Prime software.

The **Operating Settings and Conditions** page of the **Settings** dialog box allows you to specify whether the device has typical power consumption characteristics or maximum power consumption characteristics.

The **Voltage** page of the **Settings** dialog box allows you to view the operating voltage conditions for each power rail in the device, and specify supply voltages for power rails with selectable supply voltages.

If the specifications in the **Voltage** page conflict with the voltage required for operation (for example, supply voltages for some transceiver depend on the data rate), the Fitter overrides the voltage for relevant rails with the value that the Fitter calculates. The Intel Quartus Prime Power Analyzer also uses the calculated value.

On the **Temperature** page of the **Settings** dialog box, you can specify the thermal operating conditions of the device.

Related Information

- Operating Settings and Conditions Page (Settings Dialog Box)
  In *Intel Quartus Prime Help*
- Voltage Page (Settings Dialog Box)
  In *Intel Quartus Prime Help*
- Temperature Page (Settings Dialog Box)
  In *Intel Quartus Prime Help*

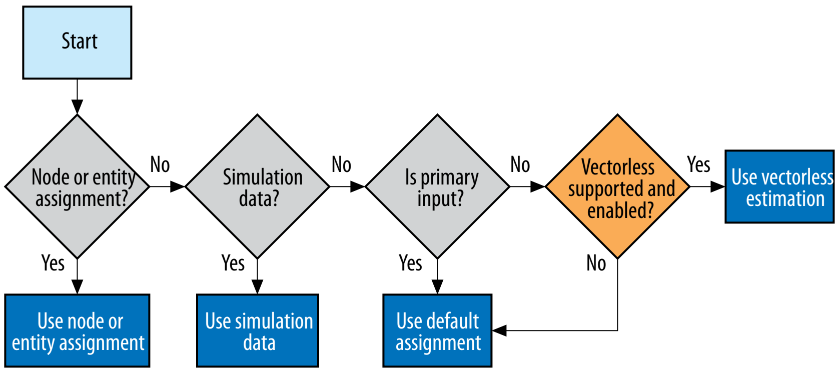
1.4.2. Sources for Signal Activity Data

The accuracy of the power estimation depends on how representative signal-activity data is during power analysis. The Power Analyzer allows you to specify signal activities from the following sources:

- Simulation results
- User-entered node, entity, and clock assignments
- User-entered default toggle rate assignment
- Vectorless estimation (selected devices)

You can mix and match the signal-activity data sources on a signal-by-signal basis. The following figure shows the priority scheme applied to each signal.

Figure 4. Signal-Activity Data Source Priority Scheme


Start
Node or entity assignment?
No
Yes
Use node or entity assignment
Simulation data?
No
Yes
Use simulation data
Is primary input?
No
Yes
Use default assignment
Vectorless supported and enabled?
Yes
No
Use vectorless estimation

Send Feedback

1.4.2.1. Waveforms from Supported Simulators

The Power Analyzer can read waveforms generated by a supported design simulation. From the simulation waveform, the Power Analyzer calculates static probability and toggle rate can be calculated for each signal.

Note: Power analysis is most accurate when you use representative input stimuli to generate simulations.

The following simulators generate outputs that the Power Analyzer supports:

- ModelSim®
- ModelSim - Intel FPGA Edition
- QuestaSim
- Active-HDL
- NCSim
- VCS*
- VCS MX
- Riviera-PRO*

Related Information

Signal Activity on page 27

1.4.2.2. .vcd Files from Third-Party Simulation Tools

The Intel Quartus Prime Power Analyzer supports `.vcd` files generated by a simulation tool as the source of activity data.

A Verilog Value Change Dump File (`.vcd`) provides signal activity and static probability information. These files include all the routing resources and the exact logic array resource usage.

For third-party simulators, use the **EDA Tool Settings** to specify the Generate Value Change Dump (VCD) file script option in the **Simulation** page of the **Settings** dialog box. These scripts instruct the third-party simulators to generate a `.vcd` that encodes the simulated waveforms. The Intel Quartus Prime Power Analyzer reads this file directly to derive the toggle rate and static probability data for each signal.

Note: The Intel Quartus Prime software does not support a built-in simulator.

Other third-party EDA simulators can generate `.vcd` files that the Power Analyzer can read. For those simulators, you must manually create a simulation script to generate the `.vcd` file.

1.4.2.2.1. Generating a .vcd in a EDA Simulation Tool

To create a `.vcd` for the design, follow these steps:

1. On the **Assignments ➤ Settings**.
2. In the **Category** list, under **EDA Tool Settings**, click **Simulation**.
3. In the **Tool name** list, select the EDA simulator.

4. In the **Format for output netlist** list, select **Verilog HDL**, or **SystemVerilog HDL**, or **VHDL**.
5. Turn on **Generate Value Change Dump (VCD) file script**.

   This option turns on the **Map illegal HDL characters** and **Enable glitch filtering** options.

   The **Map illegal HDL characters** option ensures that all signals have legal names and makes signal toggle rates available to the Power Analyzer.
   The **Enable glitch filtering** option directs the EDA Netlist Writer to perform glitch filtering when generating VHDL Output Files, Verilog Output Files, and the corresponding Standard Delay Format Output Files for use with other EDA simulation tools. This option is available regardless of whether or not you want to generate `.vcd` scripts.

   Note: For ModelSim simulations , the `+nospecify` option in the `vsim` command disables the **Specify path delays and timing checks** option. By enabling glitch filtering on the **Simulation** page, the simulation models include specified path delays. Thus, ModelSim might fail to simulate a design. As a best practice, remove the `+nospecify` option from the ModelSim `vsim` command to ensure accurate simulation for power estimation.
6. Click **Script Settings**. Select the signals that you want to write to the `.vcd`.
   — If you choose **All signals**, the generated script instructs the third-party simulator to write all connected output signals to the `.vcd` file.
   — If you choose **All signals except combinational lcell outputs**, the generated script instructs the third-party simulator to write all connected output signals to the `.vcd`, except logic cell combinational outputs.

   Note: The file can become extremely large if you write all output signals to the file, because the file size depends on the number of output signals being monitored and the number of transitions that occur.
7. Click **OK**.
8. In the **Design instance name** box, type a name for the testbench.
9. Compile the design with the Intel Quartus Prime software, and generate the necessary EDA netlist and script that instructs the third-party simulator to generate a `.vcd`.
10. In the third-party EDA simulation tool, call the generated script in the simulation tool before running the simulation.
11. Perform the simulation.

The simulation tool generates the `.vcd` file in the project directory.

1.4.2.2.2. Generating a .vcd from ModelSim Software

To generate a `.vcd` with the ModelSim software, follow these steps:

1. In the Intel Quartus Prime software, on the Assignments menu, click **Settings**.
2. In the **Category** list, under **EDA Tool Settings**, click **Simulation**.
3. In the **Tool name** list, select your preferred EDA simulator.
4. In the **Format for output netlist** list, select **Verilog HDL**, or **SystemVerilog HDL**, or **VHDL**.
5. Turn on **Generate Value Change Dump (VCD) file script**.

6. To generate the `.vcd`, perform a full compilation.
7. In the ModelSim software, compile the files necessary for simulation.
8. Load your design by clicking **Start Simulation** on the Tools menu, or use the `vsim` command.
9. Use the `.vcd` script created in 6 on page 11 using the following command: `source <design>_dump_all_vcd_nodes.tcl`
10. Run the simulation (for example, run 2000ns or run -all).
11. Quit the simulation using the `quit -sim` command, if required.
12. Exit the ModelSim software.
    If you do not exit the software, the ModelSim software might end the writing process of the **.vcd** improperly, resulting in a corrupt `.vcd`.

1.4.2.3. Signal Activities from RTL (Functional) Simulation, Supplemented by Vectorless Estimation

In the functional simulation flow, simulation provides toggle rates and static probabilities for all pins and registers in your design. Vectorless estimation fills in the values for all the combinational nodes between pins and registers, giving good results. This flow usually provides a compilation time benefit when you use the third-party RTL simulator.

1.4.2.3.1. RTL Simulation Limitation

RTL simulation may not provide signal activities for all registers in the post-fitting netlist because synthesis loses some register names. For example, synthesis might automatically transform state machines and counters, thus changing the names of registers in those structures.

1.4.2.4. Signal Activities from Vectorless Estimation and User-Supplied Input Pin Activities

The vectorless estimation flow provides a low level of accuracy, because vectorless estimation for registers is not entirely accurate.

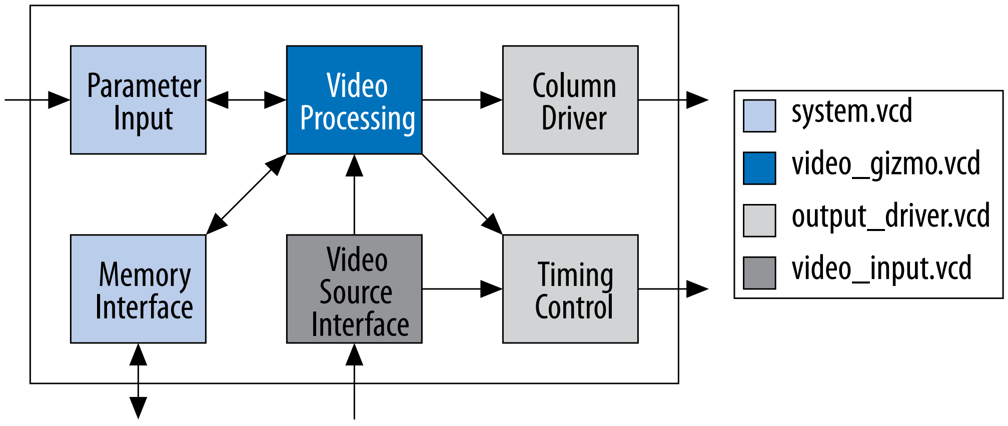
1.4.2.5. Signal Activities from User Defaults Only

The user defaults only flow provides the lowest degree of accuracy.

1.5. Power Analysis in Modular Design Flows

A common design practice is to create modular or hierarchical designs in which you develop each design entity separately, and then instantiate these modules in a higher-level entity to form a complete design. You can perform simulation on a complete design or on each module for verification. The Power Analyzer supports modular design flows when reading the signal activities from simulation files. The following figure shows an example of a modular design flow.

Figure 5. Modular Simulation Flow

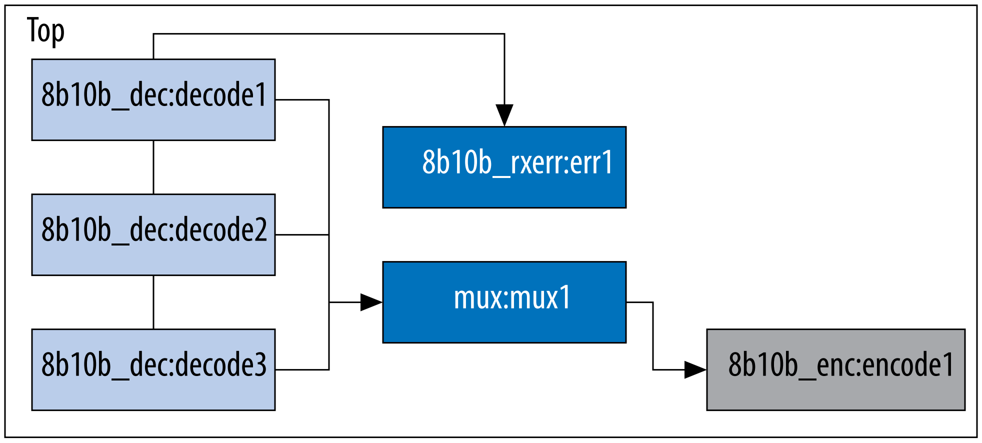


The Power Analyzer supports associating multiple `.vcd` simulation files to specific entity names, enabling the integration of partial design simulations into a complete design power analysis. When specifying multiple `.vcd` files for the design, more than one simulation file can contain signal-activity information for the same signal. In those cases, the Power analyzer follows these rules:

- When you apply multiple `.vcd` files to the same design entity, the Power Analyzer calculates the signal activity as the equal-weight arithmetic average of each `.vcd`.
- When you apply multiple simulation files to design entities at different levels in the design hierarchy, the signal activity in the power analysis derives from the simulation file that applies to the most specific design entity.

The following figure shows an example of a hierarchical design:

Figure 6. Example Hierarchical Design



The top-level module of the design, called `Top`, consists of three 8b/10b decoders, followed by a `mux`. The software encodes the output of the `mux` to produce the final output of the top-level module. An error-handling module handles any 8b/10b decoding errors. The Top module contains the top-level entity of the design and any logic not defined as part of another module. The design file for the top-level module can be a wrapper for the hierarchical entities or can contain its own logic.

The following usage scenarios show common ways that you can simulate the design and import the .vcd into the Power Analyzer:

1.5.1. Complete Design Simulation

You can simulate the entire design and generate a .vcd from a third-party simulator. The Power Analyzer can then import the .vcd (specifying the top-level design). The resulting power analysis uses the signal activities information from the generated .vcd, including those that apply to submodules, such as decode [1-3], err1, mux1, and encode1.

1.5.2. Modular Design Simulation

You can independently simulate of the top-level design, and then import all the resulting .vcd files into the Power Analyzer. For example, you can simulate the 8b10b_dec independent of the entire design and mux, 8b10b_rxerr, and 8b10b_enc. You can then import the .vcd files generated from each simulation by specifying the appropriate instance name. For example, if the files produced by the simulations are 8b10b_dec.vcd, 8b10b_enc.vcd, 8b10b_rxerr.vcd, and mux.vcd, you can use the import specifications in the following table:

Table 2. Import Specifications

File Name	Entity
8b10b_dec.vcd	Top|8b10b_dec:decode1
8b10b_dec.vcd	Top|8b10b_dec:decode2
8b10b_dec.vcd	Top|8b10b_dec:decode3
8b10b_rxerr.vcd	Top|8b10b_rxerr:err1
8b10b_enc.vcd	Top|8b10b_enc:encode1
mux.vcd	Top|mux:mux1

The resulting power analysis applies the simulation vectors in each file to the assigned entity. Simulation provides signal activities for the pins and for the outputs of functional blocks. If the inputs to an entity instance are input pins for the entire design, the simulation file associated with that instance does not provide signal activities for the inputs of that instance. For example, an input to an entity such as mux1 has its signal activity specified at the output of one of the decode entities.

1.5.3. Multiple Simulations on the Same Entity

You can perform multiple simulations of an entire design or specific modules of a design. For example, in the process of verifying the top-level design, you can have three different simulation testbenches: one for normal operation, and two for corner cases. Each of these simulations produces a separate .vcd. In this case, apply the different .vcd file names to the same top-level entity, as shown in the following table.

Table 3. Multiple Simulation File Names and Entities

File Name	Entity
`normal.vcd`	`Top`
`corner1.vcd`	`Top`
`corner2.vcd`	`Top`

The resulting power analysis uses an arithmetic average of the signal activities calculated from each simulation file to obtain the final signal activities used. If a signal `err_out` has a toggle rate of zero transition per second in `normal.vcd`, 50 transitions per second in `corner1.vcd`, and 70 transitions per second in `corner2.vcd`, the final toggle rate in the power analysis is 40 transitions per second.

If you do not want the Power Analyzer to read information from multiple instances and take an arithmetic average of the signal activities, use a `.vcd` that includes only signals from the instance that you care about.

1.5.4. Overlapping Simulations

You can perform a simulation on the entire design, and more exhaustive simulations on a submodule, such as `8b10b_rxerr`. The following table lists the import specification for overlapping simulations.

Table 4. Overlapping Simulation Import Specifications

File Name	Entity
`full_design.vcd`	`Top`
`error_cases.vcd`	`Top|8b10b_rxerr:err1`

In this case, the software uses signal activities from `error_cases.vcd` for all the nodes in the generated `.vcd` and uses signal activities from `full_design.vcd` for only those nodes that do not overlap with nodes in `error_cases.vcd`. In general, the more specific hierarchy (the most bottom-level module) derives signal activities for overlapping nodes.

1.5.5. Partial Simulations

You can perform a simulation in which the entire simulation time is not applicable to signal-activity calculation. For example, if you run a simulation for 10,000 clock cycles and reset the chip for the first 2,000 clock cycles. If the Power Analyzer performs the signal-activity calculation over all 10,000 cycles, the toggle rates are only 80% of their steady state value (because the chip is in reset for the first 20% of the simulation). In this case, you must specify the useful parts of the `.vcd` for power analysis. The **Limit VCD Period** option enables you to specify a start and end time when performing signal-activity calculations.

1.5.5.1. Specifying Start and End Time when Performing Signal-Activity Calculations using the Limit VCD Period Option

To specify a start and end time when performing signal-activity calculations using the **Limit VCD period** option, follow these steps:

1. In the Intel Quartus Prime software, click **Assignments ➤ Settings**.
2. Under the **Category** list, click **Power Analyzer Settings**.
3. Turn on the **Use input file(s) to initialize toggle rates and static probabilities during power analysis** option.
4. Click **Add**.
5. In the **File name** and **Entity** fields, browse to the necessary files.
6. Under **Simulation period**, turn on **VCD file** and **Limit VCD period** options.
7. In the **Start time** and **End time** fields, specify the desired start and end time.
8. Click **OK**.

You can also use the following tcl or qsf assignment to specify .vcd files:

```
set_global_assignment -name POWER_INPUT_FILE_NAME "test.vcd" -section_id
test.vcd
set_global_assignment -name POWER_VCD_FILE_START_TIME "10 ns" -section_id
test.vcd
set_global_assignment -name POWER_VCD_FILE_END_TIME "1000 ns" -section_id
test.vcd
set_instance_assignment -name POWER_READ_INPUT_FILE test.vcd -to test_design
```

Related Information

- set_power_file_assignment
  In *Intel Quartus Prime Help*
- Add/Edit Power Input File Dialog Box
  In *Intel Quartus Prime Help*

1.5.6. Node Name Matching Considerations

Node name mismatches happen when you have .vcd applied to entities other than the top-level entity. In a modular design flow, the gate-level simulation files created in different Intel Quartus Prime projects might not match their node names with the current Intel Quartus Prime project.

For example, you may have a file named 8b10b_enc.vcd, which the Intel Quartus Prime software generates in a separate project called 8b10b_enc while simulating the 8b10b encoder. If you import the .vcd into another project called Top, you might encounter name mismatches when applying the .vcd to the 8b10b_enc module in the Top project. This mismatch happens because the Intel Quartus Prime software might name all the combinational nodes in the 8b10b_enc.vcd differently than in the Top project.

You can avoid name mismatching with only RTL simulation data, in which register names do not change, or with an incremental compilation flow that preserves node names along with a gate-level simulation.

Note: To ensure accuracy, Intel FPGA recommends that you use an incremental compilation flow to preserve the node names of your design.

1.5.7. Glitch Filtering

The Power Analyzer defines a glitch as two signal transitions so closely spaced in time that the pulse, or glitch, occurs faster than the logic and routing circuitry can respond. The output of a transport delay model simulator contains glitches for some signals. The logic and routing structures of the device form a low-pass filter that filters out glitches that are tens to hundreds of picoseconds long, depending on the device family.

Some third-party simulators use different models than the transport delay model as the default model. Different models cause differences in signal activity and power estimation. The inertial delay model, which is the ModelSim default model, filters out more glitches than the transport delay model and usually yields a lower power estimate.

Note: Intel FPGA recommends that you use the transport simulation model when using the Intel Quartus Prime software glitch filtering support with third-party simulators. Simulation glitch filtering has little effect if you use the inertial simulation model.

Glitch filtering in a simulator can also filter a glitch on one logic element (LE) (or other circuit element) output from propagating to downstream circuit elements to ensure that the glitch does not affect simulated results. Glitch filtering prevents a glitch on one signal from producing non-physical glitches on all downstream logic, which can result in a signal toggle rate and a power estimate that are too high. Circuit elements in which every input transition produces an output transition, including multipliers and logic cells configured to implement XOR functions, are especially prone to glitches. Therefore, circuits with such functions can have power estimates that are too high when glitch filtering is not used.

Note: Intel FPGA recommends that you use the glitch filtering feature to obtain the most accurate power estimates. For `.vcd` files, the Power Analyzer flows support two levels of glitch filtering.

1.5.7.1. Enabling Tool Based Glitch Filtering

To enable the first level of glitch filtering in the Intel Quartus Prime software for supported third-party simulators, follow these steps:

1. In the Intel Quartus Prime software, click **Assignments ➤ Settings**.
2. In the **Category** list, select **Simulation under EDA Tool Settings**.
3. Select the **Tool name** to use for the simulation.
4. Turn on **Enable glitch filtering**.

1.5.7.2. Enabling Glitch Filtering During Power Analysis

The second level of glitch filtering occurs while the Power Analyzer is reading the `.vcd` generated by a third-party simulator. To enable the second level of glitch filtering, follow these steps:

1. In the Intel Quartus Prime software, click **Assignments ➤ Settings**.
2. In the **Category** list, select **Power Analyzer Settings**.
3. Under **Input File(s)**, turn on **Perform glitch filtering on VCD files**.

The `.vcd` file reader performs filtering complementary to the filtering performed during simulation and is often not as effective. While the `.vcd` file reader can remove glitches on logic blocks, the file reader cannot determine how a given glitch affects downstream logic and routing, and may eliminate the impact of the glitch completely. Filtering the glitches during simulation avoids switching downstream routing and logic automatically.

Note: When running simulation for design verification (rather than to produce input to the Power Analyzer), Intel recommends that you turn off the glitch filtering option to produce the most rigorous and conservative simulation from a functionality viewpoint. When performing simulation to produce input for the Power Analyzer, Intel FPGA recommends that you turn on the glitch filtering to produce the most accurate power estimates.

1.5.8. Node and Entity Assignments

You can assign toggle rates and static probabilities to individual nodes and entities in the design. These assignments have the highest priority, overriding data from all other signal-activity sources.

You must use the Assignment Editor or Tcl commands to create the **Power Toggle Rate** and **Power Static Probability** assignments. You can specify the power toggle rate as an absolute toggle rate in transitions per second using the **Power Toggle Rate** assignment, or you can use the **Power Toggle Rate Percentage** assignment to specify a toggle rate relative to the clock domain of the assigned node for a more specific assignment made in terms of hierarchy level.

Note: If you use the **Power Toggle Rate Percentage** assignment, and the node does not have a clock domain, the Intel Quartus Prime software issues a warning and ignores the assignment.

Assigning toggle rates and static probabilities to individual nodes and entities is appropriate for signals in which you have knowledge of the signal or entity being analyzed. For example, if you know that a 100 MHz data bus or memory output produces data that is essentially random (uncorrelated in time), you can directly enter a 0.5 static probability and a toggle rate of 50 million transitions per second.

The Power Analyzer treats bidirectional I/O pins differently. The combinational input port and the output pad for a pin share the same name. However, those ports might not share the same signal activities. For reading signal-activity assignments, the Power Analyzer creates a distinct name <*node_name~output*> when configuring the bidirectional signal as an output and <*node_name~result*> when configuring the signal as an input. For example, if a design has a bidirectional pin named `MYPIN`, assignments for the combinational input use the name `MYPIN~result`, and the assignments for the output pad use the name `MYPIN~output`.

Note: When you create the logic assignment in the Assignment Editor, you cannot find the `MYPIN~result` and `MYPIN~output` node names in the Node Finder. Therefore, to create the logic assignment, you must manually enter the two differentiating node names to create the assignment for the input and output port of the bidirectional pin.

1.5.8.1. Timing Assignments to Clock Nodes

For clock nodes, the Power Analyzer uses timing requirements to derive the toggle rate when neither simulation data nor user-entered signal-activity data is available. f_{MAX} requirements specify full cycles per second, but each cycle represents a rising transition and a falling transition. For example, a clock f_{MAX} requirement of 100 MHz corresponds to 200 million transitions per second for the clock node.

1.5.9. Default Toggle Rate Assignment

You can specify a default toggle rate for primary inputs and other nodes in your design. The Power Analyzer uses the default toggle rate when no other method specifies the signal-activity data.

The Power Analyzer specifies the toggle rate in absolute terms (transitions per second), or as a fraction of the clock rate in effect for each node. The toggle rate for a clock derives from the timing settings for the clock. For example, if the Power Analyzer specifies a clock with an f_{MAX} constraint of 100 MHz and a default relative toggle rate of 20%, nodes in this clock domain transition in 20% of the clock periods, or 20 million transitions occur per second. In some cases, the Power Analyzer cannot determine the clock domain for a node because either the Power Analyzer cannot determine a clock domain for the node, or the clock domain is ambiguous. For example, the Power Analyzer may not be able to determine a clock domain for a node if the user did not specify sufficient timing assignments. In these cases, the Power Analyzer substitutes and reports a toggle rate of zero.

Related Information

Toggle Rate on page 28

1.5.10. Vectorless Estimation

For some device families, the Power Analyzer automatically derives estimates for signal activity on nodes with no simulation or user-entered signal-activity data.

Vectorless estimation statistically estimates the signal activity of a node based on the signal activities of nodes feeding that node, and on the actual logic function that the node implements. Vectorless estimation cannot derive signal activities for primary inputs. Vectorless estimation is accurate for combinational nodes, but not for registered nodes. Therefore, the Power Analyzer requires simulation data for at least the registered nodes and I/O nodes for accuracy.

1.6. Power Analyzer Compilation Report

The Power Analyzer Compilation Report contains the following sections:

Summary

The Summary section of the report shows the estimated total thermal power consumption of your design. This includes dynamic, static, and I/O thermal power consumption. The I/O thermal power includes the total I/O power drawn from the V_{CCIO} and V_{CCPD} power supplies and the power drawn from V_{CCINT} in the I/O subsystem including I/O buffers and I/O registers. The report also includes a confidence metric that reflects the overall quality of the data sources for the signal activities. For example, a **Low** power estimation confidence value reflects that you have provided insufficient toggle rate data, or most of the signal-activity information used for power estimation is from default or vectorless estimation settings. For more information about the input data, refer to the Power Analyzer Confidence Metric report.

Settings

The Settings section of the report shows the Power Analyzer settings information of your design, including the default input toggle rates, operating conditions, and other relevant setting information.

Simulation Files Read

The Simulation Files Read section of the report lists the simulation output file that the `.vcd` used for power estimation. This section also includes the file ID, file type, entity, VCD start time, VCD end time, the unknown percentage, and the toggle percentage. The unknown percentage indicates the portion of the design module unused by the simulation vectors.

Operating Conditions Used

The Operating Conditions Used section of the report shows device characteristics, voltages, temperature, and cooling solution, if any, during the power estimation. This section also shows the entered junction temperature or auto-computed junction temperature during the power analysis.

Thermal Power Dissipated by Block

The Thermal Power Dissipated by Block section of the report shows estimated thermal dynamic power and thermal static power consumption categorized by atoms. This information provides you with estimated power consumption for each atom in your design.

By default, this section does not contain any data, but you can turn on the report with the **Write power dissipation by block to report file** option on the **Power Analyzer Settings** page.

Thermal Power Dissipation by Block Type (Device Resource Type)

This Thermal Power Dissipation by Block Type (Device Resource Type) section of the report shows the estimated thermal dynamic power and thermal static power consumption categorized by block types. This information is further categorized by estimated dynamic and static power and provides an average toggle rate by block type. Thermal power is the power dissipated as heat from the FPGA device.

Thermal Power Dissipation by Hierarchy

This Thermal Power Dissipation by Hierarchy section of the report shows estimated thermal dynamic power and thermal static power consumption categorized by design hierarchy. This information is further categorized by the dynamic and static power that was used by the blocks and routing in that hierarchy. This information is useful when locating modules with high power consumption in your design.

Core Dynamic Thermal Power Dissipation by Clock Domain

The Core Dynamic Thermal Power Dissipation by Clock Domain section of the report shows the estimated total core dynamic power dissipation by each clock domain, which provides designs with estimated power consumption for each clock domain in the design. If the clock frequency for a domain is unspecified by a constraint, the clock frequency is listed as "unspecified." For all the combinational logic, the clock domain is listed as no clock with zero MHz.

Current Drawn from Voltage Supplies

The Current Drawn from Voltage Supplies section of the report lists the current drawn from each voltage supply. The V_{CCIO} and V_{CCPD} voltage supplies are further categorized by I/O bank and by voltage. This section also lists the minimum safe power supply size (current supply ability) for each supply voltage. Minimum current requirement can be higher than user mode current requirement in cases in which the supply has a specific power up current requirement that goes beyond user mode requirement, such as the V_{CCPD} power rail in Stratix® III and Stratix IV devices, and the V_{CCIO} power rail in Stratix IV devices.

The I/O thermal power dissipation on the summary page does not correlate directly to the power drawn from the V_{CCIO} and V_{CCPD} voltage supplies listed in this report. This is because the I/O thermal power dissipation value also includes portions of the V_{CCINT} power, such as the I/O element (IOE) registers, which are modeled as I/O power, but do not draw from the V_{CCIO} and V_{CCPD} supplies.

The reported current drawn from the I/O Voltage Supplies (ICCIO and ICCPD) as reported in the Power Analyzer report includes any current drawn through the I/O into off-chip termination resistors. This can result in ICCIO and ICCPD values that are higher than the reported I/O thermal power, because this off-chip current dissipates as heat elsewhere and does not factor in the calculation of device temperature. Therefore, total I/O thermal power does not equal the sum of current drawn from each V_{CCIO} and V_{CCPD} supply multiplied by V_{CCIO} and V_{CCPD} voltage.

For SoC devices or for Arria V SoC and Cyclone® V SoC devices, there is no standalone ICC_AUX_SHARED current drawn information. The ICC_AUX_SHARED is reported together with ICC_AUX.

Confidence Metric Details

The Confidence Metric is defined in terms of the total weight of signal activity data sources for both combinational and registered signals. Each signal has two data sources allocated to it; a toggle rate source and a static probability source.

The Confidence Metric Details section also indicates the quality of the signal toggle rate data to compute a power estimate. The confidence metric is low if the signal toggle rate data comes from poor predictors of real signal toggle rates in the device during an operation. Toggle rate data that comes from simulation, user-entered assignments on specific signals or entities are reliable. Toggle rate data from default toggle rates (for example, 12.5% of the clock period) or vectorless estimation are relatively inaccurate. This section gives an overall confidence rating in the toggle rate data, from low to high. This section also summarizes how many pins, registers, and combinational nodes obtained their toggle rates from each of simulation, user entry, vectorless estimation, or default toggle rate estimations. This detailed information helps you understand how to increase the confidence metric, letting you determine your own confidence in the toggle rate data.

Signal Activities

The Signal Activities section lists toggle rates and static probabilities assumed by power analysis for all signals with fan-out and pins. This section also lists the signal type (pin, registered, or combinational) and the data source for the toggle rate and static probability. By default, this section does not contain any data, but you can turn on the report with the **Write signal activities to report file** option on the **Power Analyzer Settings** page.

Intel recommends that you keep the **Write signal activities to report file** option turned off for a large design because of the large number of signals present. You can use the Assignment Editor to specify that activities for individual nodes or entities are reported by assigning an on value to those nodes for the **Power Report Signal Activities** assignment.

Messages

The Messages section lists the messages that the Intel Quartus Prime software generates during the analysis.

1.7. Scripting Support

You can run procedures and create settings described in this chapter in a Tcl script. Alternatively, you can run procedures at a command prompt. For more information about scripting command options, refer to the Intel Quartus Prime Command-Line and Tcl API Help browser. To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp
```

Related Information

- Tcl Scripting
  In *Intel Quartus Prime Standard Edition Handbook Volume 2*
- API Functions for Tcl
  In *Intel Quartus Prime Help*

1.7.1. Running the Power Analyzer from the Command–Line

The executable to run the Power Analyzer is `quartus_pow`. For a complete listing of all command–line options supported by `quartus_pow`, type the following command at a system command prompt:

```
quartus_pow --help
```

or

```
quartus_sh --qhelp
```

The following lists the examples of using the `quartus_pow` executable. Type the command listed in the following section at a system command prompt:

Note: These examples assume that operations are performed on Intel Quartus Prime project called *sample*.

- To instruct the Power Analyzer to generate a EPE File:

```
quartus_pow sample --output_epe=sample.csv ←
```

- To instruct the Power Analyzer to generate a EPE File without performing the power estimate:

```
quartus_pow sample --output_epe=sample.csv --estimate_power=off  ←
```

- To instruct the Power Analyzer to use a `.vcd` as input (`sample.vcd`):

```
quartus_pow sample --input_vcd=sample.vcd ←
```

- To instruct the Power Analyzer to use two `.vcd` files as input files (`sample1.vcd` and `sample2.vcd`), perform glitch filtering on the `.vcd` and use a default input I/O toggle rate of 10,000 transitions per second:

```
quartus_pow sample --input_vcd=sample1.vcd --input_vcd=sample2.vcd \
--vcd_filter_glitches=on --\
default_input_io_toggle_rate=10000transitions/s
```

- To instruct the Power Analyzer to not use an input file, a default input I/O toggle rate of 60%, no vectorless estimation, and a default toggle rate of 20% on all remaining signals:

```
quartus_pow sample --no_input_file --default_input_io_toggle_rate=60% \
--use_vectorless_estimation=off --default_toggle_rate=20%
```

Note: No command–line options are available to specify the information found on the **Power Analyzer Settings Operating Conditions** page. Use the Intel Quartus Prime GUI to specify these options.

The `quartus_pow` executable creates a report file, *<revision name>*`.pow.rpt`. You can locate the report file in the main project directory. The report file contains the same information that the Power Analyzer Compilation Report.

Related Information

Power Analyzer Compilation Report on page 18

1.8. Power Analysis Revision History

The following revision history applies to this chapter:

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1.0	• Initial release in Intel Quartus Prime Standard Edition User Guide. • General chapter reorganization. • Moved *Factors Affecting Power Consumption* to chapter: *Power Optimization*.

continued...

Document Version	Intel Quartus Prime Version	Changes
		• Updated figure: *Power Analyzer High-level Flow*. • Divided topic: *Types of Power Analysis* into two topics: *Power Estimations and Design Requirements* and *Design Activity and Power Analysis*. • Updated figure: *Power Analysis Tools from Design Concept through Design Implementation* and renamed to: *Estimation Accuracy for Different Inputs and Power Analysis Tools*
2018.06.11	18.0.0	• In *Comparison of the EPE and the Intel Quartus Prime Power Analyzer*, updated the data output types that the Power Analyzer supports. • In *Comparison of the EPE and the Intel Quartus Prime Power Analyzer*, added row about estimation of transceiver power for features that you enable only through dynamic reconfiguration. • Specified features not supported by the Power Analyzer.
2017.05.08	17.0.0	Removed references to PowerPlay name. Power analysis occurs in the Intel Quartus Prime Power Analyzer.
2015.11.02	15.1.0	Changed instances of *Quartus II* to *Intel Quartus Prime*.
2014.12.15	14.1.0	• Removed Signal Activities from Full Post-fit Netlist (Timing) Simulation and Signal Activities from Full Post-fit Netlist (Zero Delay) Simulation sections as these are no longer supported. • Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings.
2014.08.18	14.0a10.0	Updated "Current Drawn from Voltage Supplies" to clarify that for SoC devices or for Arria V SoC and Cyclone V SoC devices, there is no standalone ICC_AUX_SHARED current drawn information. The ICC_AUX_SHARED is reported together with ICC_AUX.
November 2012	12.1.0	• Updated "Types of Power Analyses" on page 8–2, and "Confidence Metric Details" on page 8–23. • Added "Importance of .vcd" on page 8–20, and "Avoiding Power Estimation and Hardware Measurement Mismatch" on page 8–24
June 2012	12.0.0	• Updated "Current Drawn from Voltage Supplies" on page 8–22. • Added "Using the HPS Power Calculator" on page 8–7.
November 2011	10.1.1	• Template update. • Minor editorial updates.
December 2010	10.1.0	• Added links to Quartus II Help, removed redundant material. • Moved "Creating PowerPlay EPE Spreadsheets" to page 8–6. • Minor edits.
July 2010	10.0.0	• Removed references to the Quartus II Simulator. • Updated Table 8–1 on page 8–6, Table 8–2 on page 8–13, and Table 8–3 on page 8–14. • Updated Figure 8–3 on page 8–9, Figure 8–4 on page 8–10, and Figure 8–5 on page 8–12.
November 2009	9.1.0	• Updated "Creating PowerPlay EPE Spreadsheets" on page 8–6 and "Simulation Results" on page 8–10. • Added "Signal Activities from Full Post-fit Netlist (Zero Delay) Simulation" on page 8–19 and "Generating a .vcd from Full Post-fit Netlist (Zero Delay) Simulation" on page 8–21. • Minor changes to "Generating a .vcd from ModelSim Software" on page 8–21. • Updated Figure 11–8 on page 11–24.

continued...

Document Version	Intel Quartus Prime Version	Changes
March 2009	9.0.0	• This chapter was chapter 11 in version 8.1. • Removed Figures 11-10, 11-11, 11-13, 11-14, and 11-17 from 8.1 version.
November 2008	8.1.0	• Updated for the Quartus II software version 8.1. • Replaced Figure 11-3. • Replaced Figure 11-14.
May 2008	8.0.0	• Updated Figure 11–5. • Updated "Types of Power Analyses" on page 11–5. • Updated "Operating Conditions" on page 11–9. • Updated "PowerPlay Power Analyzer Compilation Report" on page 11–31. • Updated "Current Drawn from Voltage Supplies" on page 11–32.

Related Information

Documentation Archive
For previous versions of the *Intel Quartus Prime Handbook*, search the documentation archives.

2. Power Optimization

The Intel Quartus Prime software offers power-driven compilation to fully optimize device power consumption. Power-driven compilation focuses on reducing the design's total power consumption in synthesis and place-and-route stages.

This chapter describes the power-driven compilation feature and flow in detail, as well as low power design techniques that can further reduce power consumption in your design. The techniques primarily target Arria, Stratix, and Cyclone series of devices. These devices utilize a low-k dielectric material that dramatically reduces dynamic power and improves performance. Arria series, Stratix IV, and Stratix V device families include efficient logic structures called adaptive logic modules (ALMs) that obtain maximum performance while minimizing power consumption. Cyclone device families offer the optimal blend of high performance and low power in a low-cost FPGA.

This chapter focuses on design optimization options and techniques that help reduce core dynamic power and I/O power. In addition to these techniques, there are additional power optimization techniques available for specific devices, including Programmable Power Technology and Device Speed Grade Selection.

Related Information

- AN 514: Power Optimization in Stratix IV FPGAs
- Power Analysis on page 4
- AN 711: Power Reduction Features in Intel Arria 10 Devices
- Intel FPGA Literature and Technical Documentation

2.1. Factors Affecting Power Consumption

Understanding the following factors that affect power consumption allows you to use the Power Analyzer and interpret its results effectively:

Design Activity and Power Analysis on page 25

Device Selection on page 26

Environmental Conditions on page 26

Device Resource Usage on page 27

Signal Activity on page 27

2.1.1. Design Activity and Power Analysis

Power consumption of a device also depends on the design's activity over time. Static power (P_{STATIC}) is the thermal power that a chip dissipates independent of user clocks. P_{STATIC} includes leakage power from all FPGA functional blocks, except for I/O DC bias power and transceiver DC bias power, which are accounted for in the I/O and transceiver sections. Dynamic power is the additional power consumption of a device due to signal activity or switching.

2.1.2. Device Selection

Device families have different power characteristics. Many parameters affect the device family power consumption, including choice of process technology, supply voltage, electrical design, and device architecture.

Power consumption also varies in a single device family. A larger device with more transistors consumes more static power than a smaller device in the same family. In devices that employ global routing architectures, dynamic power can also increase with device size.

The choice of device package also affects the ability of the device to dissipate heat, and you may need to use a different cooling solution to comply with junction temperature constraints.

Process variation can affect power consumption. Process variation primarily impacts static power, because sub-threshold leakage current varies exponentially with changes in transistor threshold voltage. Therefore, you must consult device specifications for static power, and not rely on empirical observation. Process variation has a weak effect on dynamic power.

2.1.3. Environmental Conditions

The main environmental parameters affecting junction temperature are operating temperature and the cooling solution. Operating temperature primarily affects device static power consumption. Higher junction temperatures result in higher static power consumption. The device thermal power and cooling solution that you use must keep the device junction temperature within the maximum operating range for the device.

The following table lists the environmental conditions that influence power consumption.

Table 5. Environmental Conditions that Affect Power Consumption

Environmental Condition	Description
Airflow	Measures how quickly the device replaces heated air from the vicinity of the device with air at ambient temperature. You can either specify airflow as "still air" when you are not using a fan, or as the linear feet per minute rating of the fan in the system. Higher airflow decreases thermal resistance.
Heat Sink and Thermal Compound	A heat sink allows more efficient heat transfer from the device to the surrounding area because of its large surface area exposed to the air. The thermal compound that interfaces the heat sink to the device also influences the rate of heat dissipation. The case-to-ambient thermal resistance (θ_{CA}) parameter describes the cooling capacity of the heat sink and thermal compound employed at a given airflow. Larger heat sinks and more effective thermal compounds reduce θ_{CA}.
Junction Temperature	The junction temperature of a device is equal to: $T_{Junction}=T_{Ambient}+P_{Thermal}\cdot{}_{JA}$

continued...

Environmental Condition	Description
	in which θ_{JA} is the total thermal resistance from the device transistors to the environment, in degrees Celsius per watt. The value θ_{JA} is equal to the sum of the junction-to-case (package) thermal resistance (θ_{JC}), and the case-to-ambient thermal resistance (θ_{CA}) of the cooling solution.
Board Thermal Model	The junction-to-board thermal resistance (θ_{JB}) is the thermal resistance of the path through the board, in degrees Celsius per watt. To compute junction temperature, you can use this board thermal model along with the board temperature, the top-of-chip θ_{JA} and ambient temperatures.

2.1.4. Device Resource Usage

Power consumption depends on the number and types of device resources that a design uses.

2.1.4.1. Number, Type, and Loading of I/O Pins

Output pins drive off-chip components, resulting in high-load capacitance that leads to a high-dynamic power per transition. Terminated I/O standards require external resistors that draw constant (static) power from the output pin.

2.1.4.2. Number and Type of Hard Logic Blocks

A design with more logic elements (LEs), multiplier elements, memory blocks, transceiver blocks, or HPS system tends to consume more power than a design with fewer circuit elements. The operating mode of each circuit element also affects its power consumption.

For example, a DSP block performing 18 × 18 multiplications and a DSP block performing multiply-accumulate operations consume different amounts of dynamic power, because of different amounts of charging internal capacitance on each transition. The operating mode of a circuit element also affects static power.

2.1.4.3. Number and Type of Global Signals

Global signal networks span large portions of the device and have high capacitance, resulting in significant dynamic power consumption. The type of global signal is important as well. Global clocks cover the entire device, whereas quadrant clocks only span one-fourth of the device. For example, Stratix V devices support global clocks and quadrant (regional) clocks. Clock networks that span smaller regions have lower capacitance and tend to consume less power. The location of the logic array blocks (LABs) driven by the clock network can also have an impact because the Intel Quartus Prime software automatically disables unused branches of a clock.

2.1.5. Signal Activity

The behavior of each signal in the design is an important factor in estimating power consumption. To get accurate results from the power analysis, the signal activity must represent the actual operating behavior of the design.

The two most important behaviors of a signal are toggle rate and static probability.

2.1.5.1. Toggle Rate

The toggle rate of a signal is the average number of times that the signal changes value per unit of time. The units for toggle rate are transitions per second, and a transition is a change from `1` to `0`, or `0` to `1`.

Note: Inaccurate signal toggle rate data is the largest source of power estimation error.

Dynamic power increases linearly with the toggle rate as you charge the board trace model more frequently for logic and routing. The Intel Quartus Prime software models full rail-to-rail switching. For high toggle rates, especially on circuit output I/O pins, the circuit can transition before fully charging the downstream capacitance. The result is a slightly conservative prediction of power by the Power Analyzer.

Related Information

Default Toggle Rate Assignment on page 18

2.1.5.2. Static Probability

The static probability of a signal is the fraction of time that the signal is logic `1` during device operation. Static probability ranges from `0` (always at ground) to `1` (always at logic-high).

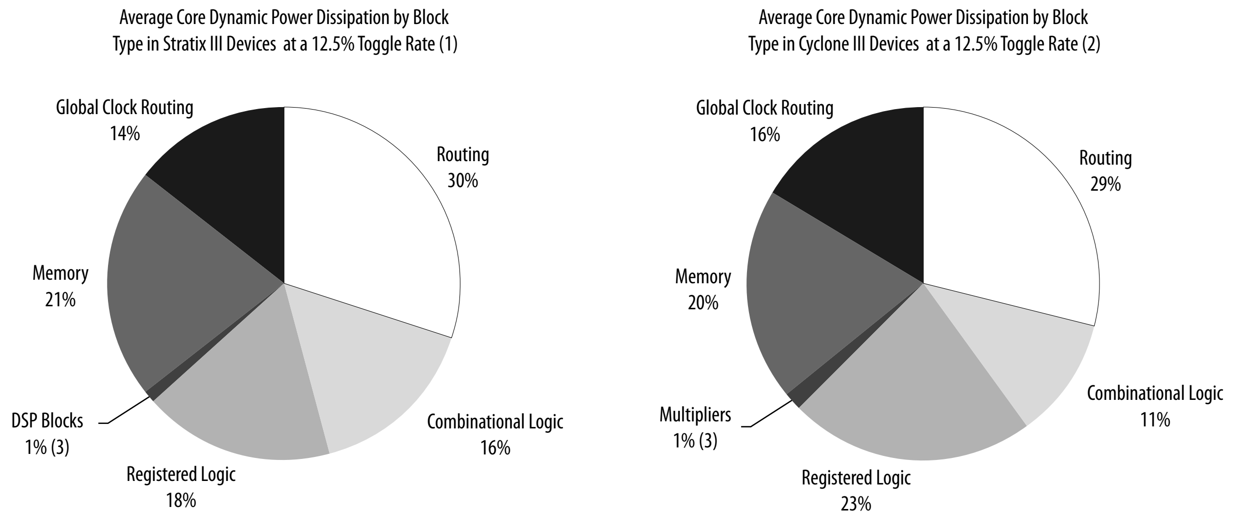
The static probability of input signals impacts the design's static power consumption, due to state-dependent leakage in routing and logic. This effect becomes more important for smaller geometries. In output I/O standards that drive termination resistors, the static power also depends on the static probability on I/O pins.

2.2. Power Dissipation

You can refine techniques that reduce power consumption in a design by understanding the sources of power dissipation.

The following figure shows the power dissipation of Stratix and Cyclone devices in different designs. The analysis considers a fixed clock rate of 100 MHz and exhibits varied logic resource utilization across available resources.

Figure 7. Average Core Dynamic Power Dissipation



Notes:

1. These results originate from 103 designs.
2. These results originate from 96 designs.
3. In designs using DSP blocks, DSPs consumed 5% of core dynamic power.

In Stratix and Cyclone device families, a series of column and row interconnect wires of varying lengths provide signal interconnections between logic array blocks (LABs), memory block structures, and digital signal processing (DSP) blocks or multiplier blocks. These interconnects dissipate the largest component of device power.

FPGA combinational logic is another source of power consumption. For more information about ALMs and LEs in Cyclone or Stratix devices, refer to the respective device handbook.

Memory and clock resources are other major consumers of power in FPGAs. Stratix devices feature the TriMatrix memory architecture. TriMatrix memory includes 512-bit M512 blocks, 4-Kbit M4K blocks, and 512-Kbit M-RAM blocks, which are configurable to support many features. Stratix IV TriMatrix on-chip memory is an enhancement based upon the Stratix II FPGA TriMatrix memory and includes three sizes of memory blocks: MLAB blocks, M9K blocks, and M144K blocks. Stratix IV and Stratix V devices feature Programmable Power Technology, an advanced architecture that enables a smooth trade-off between speed and power. The core of each Stratix IV and Stratix V device is divided into tiles, each of which may be put into a high-speed or low-power mode. The primary benefit of Programmable Power Technology is to reduce static power, with a secondary benefit being a small reduction in dynamic power. Cyclone IV GX devices have 9-Kbit M9K memory blocks.

2.3. Design Space Explorer II for Power-Driven Optimization

The Design Space Explorer II (DSE II) tool allows you to find and implement the project settings that result in best power behavior.

The DSE II offers two options in **Exploration mode** that target power optimization: **Power (High Effort)** and **Power (Aggressive)**. In both cases, the target is an overall improvement in the design's power; specifically, reducing the total thermal power in the design.

When the optimization targets power, the DSE II runs the Intel Quartus Prime Power Analyzer for every group of settings. The resultant reports help you debug the design and determine trade-offs between power requirements and performance optimization.

Related Information

- Design Space Explorer II
  In *Intel Quartus Prime Standard Edition User Guide: Design Optimization*
- Launch Design Space Explorer Command (Tools Menu)
  In *Intel Quartus Prime Help*

2.4. Power-Driven Compilation

The standard Intel Quartus Prime compilation flow consists of Analysis and Synthesis, placement and routing, Assembly, and Timing Analysis. Power-driven compilation takes place at the Analysis and Synthesis and Place-and-Route stages.

Intel Quartus Prime software settings that control power-driven compilation are located in the **Power optimization during synthesis** list in the **Advanced Settings (Synthesis)** dialog box, and the **Power optimization during fitting** list on the **Advanced Fitter Settings** dialog box. The following sections describes these power optimization options at the Analysis and Synthesis and Fitter levels.

2.4.1. Power-Driven Synthesis

Synthesis netlist optimization occurs during the synthesis stage of the compilation flow. You can apply these settings on a project or entity level.

The **Power Optimization During Synthesis** logic option determines how aggressively Analysis & Synthesis optimizes the design for power. To access this option at a project level, click **Assignments ➤ Settings ➤ Compiler Settings ➤ Advanced Settings (Synthesis)**.

Table 6. Power Optimization During Synthesis Options

Settings	Description	Optimization Techniques Included
Off	The Compiler does not perform netlist, placement, or routing optimizations to minimize power.	-
Normal compilation (Default)	The Compiler applies low compute effort algorithms to minimize power through netlist optimizations that do not reduce design performance.	• Memory block optimization • Power-aware logic mapping
Extra effort	Besides the techniques in the **Normal compilation** setting, the Compiler applies high-compute-effort algorithms to minimize power through netlist optimizations. Selecting this option might impact performance.	• Memory block optimization • Power-aware logic mapping • Power-aware memory balance

You can also control memory optimization options from the Intel Quartus Prime **Settings** dialog box. The **Default Parameters** page allows you to edit the **Low_Power_Mode** parameter. The settings for this parameter are equivalent to the values of the **Power Optimization During Synthesis** logic options. The **Low_Power_Mode** parameter always takes precedence over the **Optimize Power for Synthesis** option for power optimization on memory.

Table 7. Low Power Mode Parameter Options

Parameter Value	Equivalent Setting in Power Optimization During Synthesis Logic Option
None	**Off**
Auto	**Normal compilation**
All	**Extra effort**

Related Information

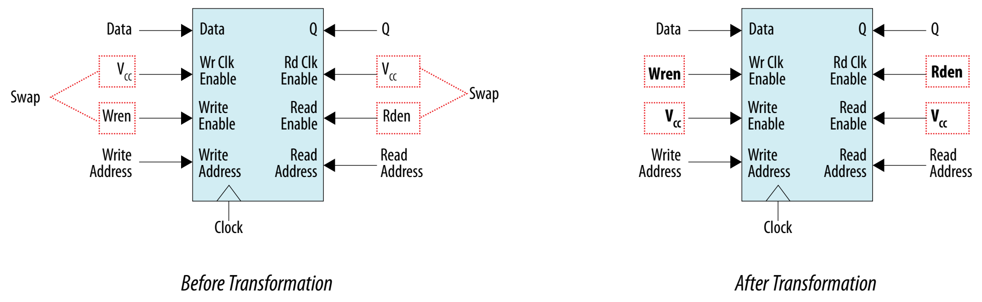
- Clock Enable in Memory Blocks on page 38
- Intel Quartus Prime Compiler Settings on page 35

2.4.1.1. Memory Block Optimization

Memory optimization involves moving user-defined read/write enable signals to associated read-and-write clock enable signals for all memory types.

Memory blocks can represent a large fraction of total design dynamic power. Minimizing the number of memory blocks accessed during each clock cycle can significantly reduce memory power.

Figure 8. Memory Block Transformation



In the default implementation of a simple dual-port memory block, write-clock enable signals and read-clock enable signals connect to V_{CC}, making both read and write memory ports active during each clock cycle.

Memory transformation moves the read-enable and write-enable signals to the respective read-clock enable and write-clock enable signals. This technique reduces the design’s memory power consumption, because memory ports are shut down when they are not accessed.

For Stratix IV and Stratix V devices, the memory transformation takes place at the Fitter level by selecting the **Normal compilation** settings for the power optimization option.

In Cyclone IV GX and StratixIV devices, the read-during-write behavior impacts the power of single-port and bidirectional dual-port RAMs. As a best practice, you can allow optimization by setting the read-during-write parameter to "Don't care" at the HDL level, and set the `read-enable` signal to the inversion of the existing `write-enable` signal (if one exists). This allows the core of the RAM to shut down, which prevents switching, saving a significant amount of power.

2.4.1.2. Power-Aware Logic Mapping

Power-aware logic mapping reduces power by rearranging the logic during synthesis to eliminate nets with high switching rates.

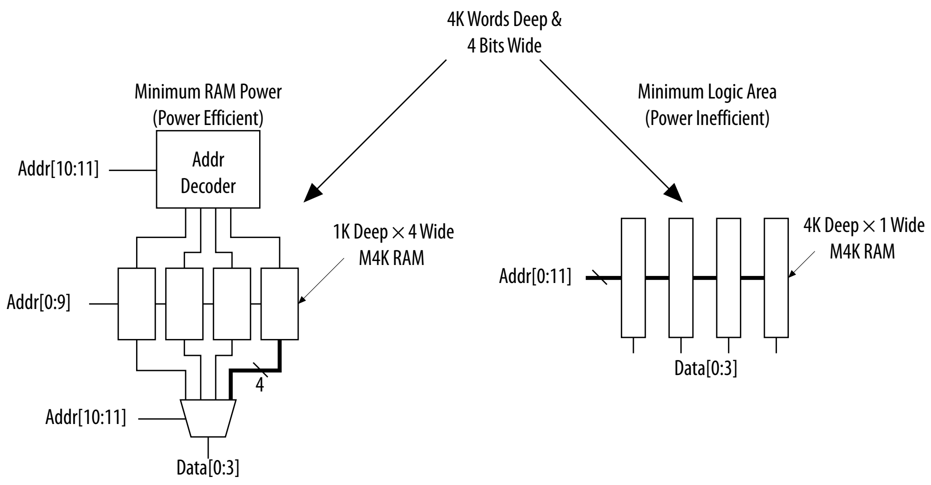
2.4.1.3. Power-Aware Memory Balancing

Power-aware memory balancing chooses the best configuration for a memory implementation and provides optimal power saving by determining the required number of memory blocks, decoder, and multiplexer circuits. When the design does not specify target-embedded memory blocks for the design's memory functions, the power-aware balancer automatically selects them during memory implementation.

The Compiler includes this optimization technique when the **Power Optimization During Synthesis** logic option is set to **Extra effort**.

The following figure is an example of a 4k × 4 (4k deep and 4 bits wide) memory implementation in two different configurations using M4K memory blocks available in some Stratix devices.

Figure 9. 4K × 4 Memory Implementation Using Multiple M4K Blocks



The minimum logic area implementation configures M4K blocks as 4k × 1. The Intel Quartus Prime software uses this implementation as the default, because the resulting design has the minimum logic area (0 logic cells) and the highest speed. However, all four M4K blocks are active on each memory access, which increases RAM power.

The minimum RAM power implementation configures four M4K blocks as 1k × 4 for optimal power saving. The RAM IP core includes an address decoder to select which of the four M4K blocks is active on a given cycle, based on the state of the top two user address bits. The RAM IP core implements a multiplexer to feed the downstream logic by choosing the appropriate M4K output. This implementation reduces RAM power because only one M4K block is active on any cycle, but it requires extra logic cells, costing logic area and potentially impacting design performance.

There is a trade-off between power saved by accessing fewer memories and power consumed by the extra decoder and multiplexor logic. The Intel Quartus Prime software automatically balances the power savings against the costs to choose the lowest power configuration for each logical RAM. The benchmark data shows that the power-driven synthesis can reduce memory power consumption by as much as 60% in Stratix devices.

You can also set the **MAXIMUM_DEPTH** parameter manually to configure the memory for low power optimization. This technique is the same as the power-aware memory balancer, but it is manual rather than automatic like the **Extra effort** setting in the **Power optimization** list. The **MAXIMUM_DEPTH** parameter always takes precedence over the **Optimize Power for Synthesis** options for power optimization on memory optimization. You can set the **MAXIMUM_DEPTH** parameter for memory modules manually in the Intel FPGA IP instantiation or in the IP Catalog.

Related Information

Maximum Block Depth Configuration
In *Embedded Memory (RAM: 1-PORT, RAM: 2-PORT, ROM: 1-PORT, and ROM: 2-PORT) User Guide*

2.4.2. Power-Driven Fitter

The Intel Quartus Prime software allows you to control the power-driven compilation setting of the Fitter on a project-wide basis. The **Advanced Fitter Settings** dialog box page provides the **Power optimization during Fitting** logic option, that determines how aggressively the Fitter optimizes the design for power.

Table 8. Power-Driven Fitter Option

Option	Description
Off	The Fitter does not perform optimizations to minimize power.
Normal compilation (Default)	The Fitter applies low compute effort algorithms to minimize power through placement and routing optimizations. These techniques do not reduce design performance. Includes DSP optimizations that create power-efficient DSP block configurations for DSP functions.
Extra effort	Besides the optimization techniques of the **Normal Compilation** option, the Fitter applies high compute effort algorithms to minimize power through placement and routing optimizations. These techniques might impact performance. The **Extra effort** setting for the Fitter requires extensive effort to optimize the design for power and can increase compilation time.

For Stratix IV and Stratix V devices, the **Normal compilation** setting enables the Programmable Power Technology to configure tiles as high-speed mode or low-power mode. Programmable Power Technology is always turned **ON** even when the **OFF** setting is selected for the **Power optimization** option. Tiles are the combination of LAB and MLAB pairs (including the adjacent routing associated with LAB and MLAB), which can be configured to operate in high-speed or low-power mode. This level of power optimization does not have any affect on the fitting, timing results, or compile time.

The **Extra effort** setting the Fitter works to minimize power even after the design meets timing requirements by moving the logic closer during placement to localize high-toggling nets and choosing routes with low capacitance.

The **Extra effort** setting uses a Value Change Dump (`.vcd`) file that guides the Fitter to fully optimize the design for power, based on the signal activity of the design. The best power optimization during fitting results from using the most accurate signal activity information. If there is no `.vcd` file, the Intel Quartus Prime software estimates the signal activities from the settings in the **Power Analyzer Settings** page in the **Settings** dialog box, such as assignments, clock assignments, and vectorless estimation values. The benchmark data shows that the power-driven Fitter technique can reduce power consumption by as much as 19% in Stratix devices. On average, you can reduce core dynamic power by 16% with the Extra effort synthesis and Extra effort fitting settings, as compared to the **Off** settings in both synthesis and Fitter options for power-driven compilation.

Related Information

- AN 514: Power Optimization in Stratix IV FPGAs
- Power Analyzer Settings Page (Settings Dialog Box)
  In *Intel Quartus Prime Help*
- Value Change Dump File (.vcd) Definition
  In *Intel Quartus Prime Help*
- Assignment Editor Options on page 36

2.4.3. Area-Driven Synthesis

Using area optimization rather than timing or delay optimization during synthesis saves power because you use fewer logic blocks. Using less logic usually means less switching activity.

The Intel Quartus Prime software provides **Speed**, **Balanced**, or **Area** for the **Optimization Technique** option. You can also specify this logic option for specific modules in your design with the Assignment Editor in cases where you want to reduce area using the **Area** setting (potentially at the expense of register-to-register timing performance) while leaving the default **Optimization Technique** setting at **Balanced** (for the best trade-off between area and speed for certain device families). The **Speed Optimization Technique** can increase the resource usage of your design if the constraints are too aggressive and can also result in increased power consumption.

The benchmark data shows that the area-driven technique can reduce power consumption by as much as 31% in Stratix devices and as much as 15% in Cyclone devices.

Related Information

Assignment Editor Options on page 36

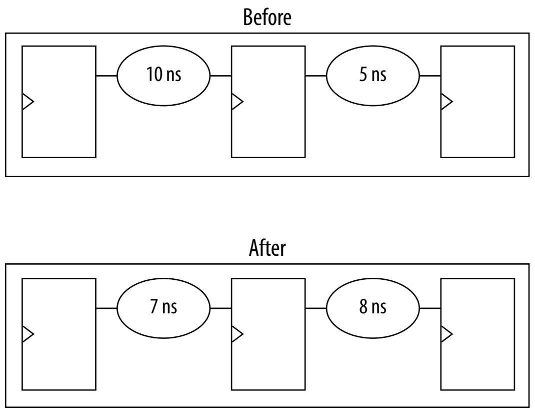
2.4.4. Gate-Level Register Retiming

You can also use gate-level register retiming to reduce circuit switching activity. Retiming shuffles registers across combinational blocks without changing design functionality.

The **Perform gate-level register retiming** option in the Intel Quartus Prime software enables the movement of registers across combinational logic to balance timing, allowing the software to trade off the delay between critical and noncritical paths.

Retiming uses fewer registers than pipelining. In this example of gate-level register retiming, the 10 ns critical delay is reduced by moving the register relative to the combinational logic, resulting in the reduction of data depth and switching activity.

Figure 10. Gate-Level Register Retiming



Gate-level register retiming makes changes at the gate level. If you are using an atom netlist from a third-party synthesis tool, you must also select the **Perform WYSIWYG primitive resynthesis** option to undo the atom primitives to gates mapping (so that register retiming can be performed), and then to remap gates to Intel primitives.

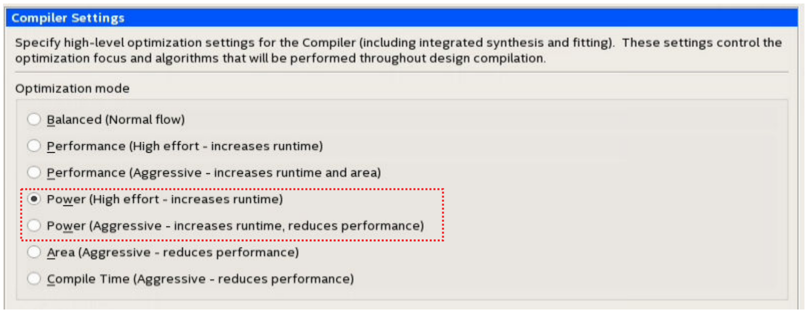
When using Intel Quartus Prime integrated synthesis, retiming occurs during synthesis before the design is mapped to Intel primitives. The benchmark data shows that the combination of WYSIWYG remapping and gate-level register retiming techniques can reduce power consumption by as much as 6% in Stratix devices and as much as 21% in Cyclone devices.

2.4.5. Intel Quartus Prime Compiler Settings

The Intel Quartus Prime software provides settings that optimize power for the full design.

To set the optimization mode on the Intel Quartus Prime software, click **Assignments ➤ Settings ➤ Compiler Settings**.

Figure 11. Compiler Settings for Power Optimization


Compiler Settings

Specify high-level optimization settings for the Compiler (including integrated synthesis and fitting). These settings control the optimization focus and algorithms that will be performed throughout design compilation.

Optimization mode

- ○ Balanced (Normal flow)
- ○ Performance (High effort - increases runtime)
- ○ Performance (Aggressive - increases runtime and area)
- ● Power (High effort - increases runtime)
- ○ Power (Aggressive - increases runtime, reduces performance)
- ○ Area (Aggressive - reduces performance)
- ○ Compile Time (Aggressive - reduces performance)

The two power optimization modes direct the Compiler to prioritize one optimization metric.

Power (High effort—increases runtime)

High effort modes enable additional optimizations that increase compilation time and do not affect design performance. High Power Effort mode guides the Compiler to spend additional compilation time reducing routing utilization, which saves dynamic power.

Power (Aggressive—increases runtime, reduces performance)

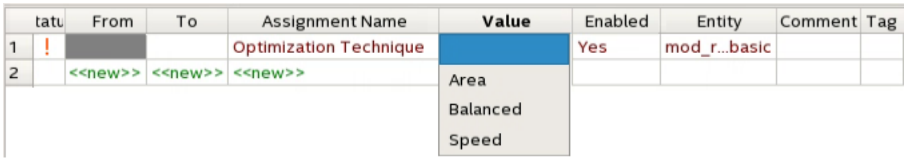
Aggressive modes increase compilation time and make trade-offs that may harm other optimization metrics (performance, area, etc.). In Aggressive Power mode, the Compiler attempts to reduce the routing usage of signals with the highest specified (via Signal Activity File) or estimated toggle rates, saving additional dynamic power but potentially affecting performance.

2.4.6. Assignment Editor Options

The Assignment Editor allows you to select Optimization Technique & Synthesis Power Optimization for individual modules. With this feature, you can focus on the parts of the design that require more work.

The **Optimization Technique** logic option specifies the overall optimization goal for Analysis & Synthesis: attempt to maximize performance or minimize logic usage.

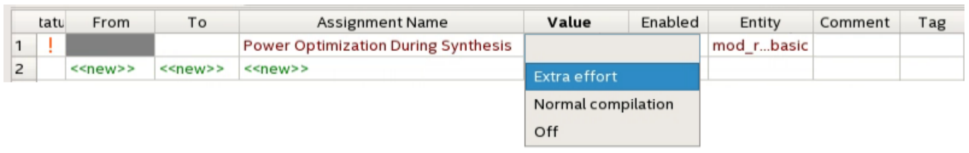
Figure 12. Optimization Technique Options



	tatu	From	To	Assignment Name	Value	Enabled	Entity	Comment	Tag
1	!			Optimization Technique		Yes	mod_r...basic		
2		<<new>>	<<new>>	<<new>>					

The **Power Optimization During Synthesis** logic option determines how aggressively Analysis & Synthesis optimizes the design for power.

Figure 13. Power Optimization During Synthesis Options



tatu	From	To	Assignment Name	Value	Enabled	Entity	Comment	Tag
1 !			Power Optimization During Synthesis			mod_r...basic		
2	<<new>>	<<new>>	<<new>>					

Table 9. Power Optimization During Synthesis Options

Settings	Description	Optimization Techniques Included
Off	The Compiler does not perform netlist, placement, or routing optimizations to minimize power.	-
Normal compilation (Default)	The Compiler applies low compute effort algorithms to minimize power through netlist optimizations that do not reduce design performance.	• Memory block optimization • Power-aware logic mapping
Extra effort	Besides the techniques in the **Normal compilation** setting, the Compiler applies high-compute-effort algorithms to minimize power through netlist optimizations. Selecting this option might impact performance.	• Memory block optimization • Power-aware logic mapping • Power-aware memory balance

Related Information

- Area-Driven Synthesis on page 34
- Power-Driven Fitter on page 33

2.5. Design Guidelines

During FPGA design implementation, you can apply the following design techniques to reduce power consumption. This section provides detailed design techniques for Cyclone IV GX devices that affect overall design power. The results of these techniques are different from design to design.

2.5.1. Clock Power Management

Clocks represent a significant portion of dynamic power consumption due to their high switching activity and long paths. Actual clock-related power consumption is higher, because the power consumption of a block includes local clock distribution within logic, memory, and DSP or multiplier blocks.

The Intel Quartus Prime software optimizes clock routing power automatically, enabling only those portions of the clock network that are necessary to feed downstream registers.

Related Information

Clock Control Block IP Core User Guide (ALTCLKCTRL)

2.5.1.1. Clock Enable in Memory Blocks

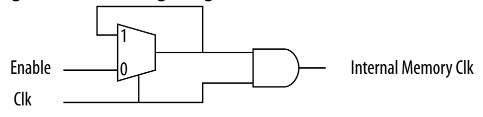
In memory blocks, power consumption is tied to the clock rate, and is insensitive to the toggle rate on the data and address lines. Memory consumes approximately 20% of the core dynamic power in typical designs.

When a memory block is clocked, a sequence of timed events occur within the block to execute a read or write. The circuitry that the clock controls consumes the same amount of power, independent of changes in address or data from one cycle to the next. Thus, the toggle rate of input data and the address bus have no impact on memory power consumption.

The key to reducing memory power consumption is to reduce the number of memory clocking events. You can achieve this reduction through network-wide clock gating, or on a per-memory basis through use of the clock enable signals on the memory ports.

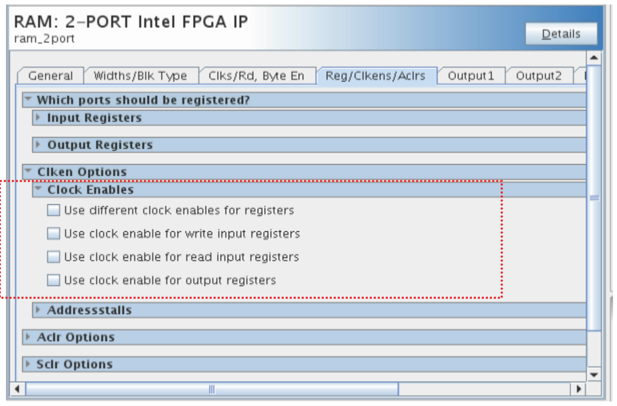
Figure 14. Memory Clock Enable Signal

Logical view of the internal clock of the memory block. Use the appropriate enable signals on the memory to make use of the clock enable signal instead of gating the clock.



The clock enable signal enables the memory only when necessary, and shuts down for the rest of the time, reducing the overall memory power consumption. You include these enable signals when generating the memory block function.

Figure 15. Clock Enable in RAM 2-Port


RAM: 2-PORT Intel FPGA IP
ram_2port

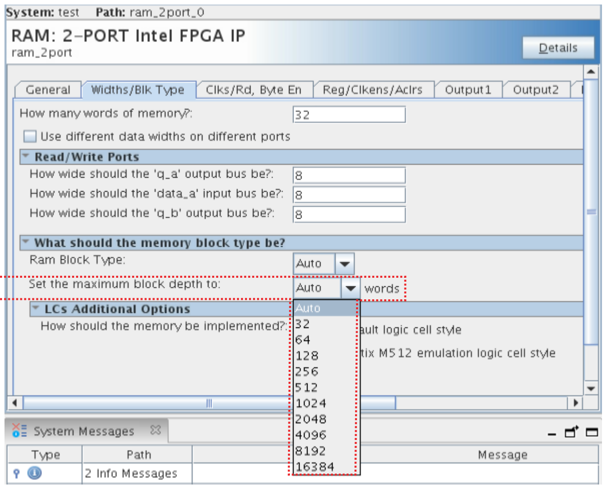
☐ Use different clock enables for registers
☐ Use clock enable for write input registers
☐ Use clock enable for read input registers
☐ Use clock enable for output registers

For example, consider a design that contains a 32-bit-wide M4K memory block in ROM mode that is running at 200 MHz. Assuming that the output of this block is only required approximately every four cycles, this memory block consumes 8.45 mW of dynamic power according to the demands of the downstream logic. By adding a small amount of control logic to generate a read clock enable signal for the memory block only on the relevant cycles, the power can be cut 75% to 2.15 mW.

You can also use the `MAXIMUM_DEPTH` parameter in your memory IP core to save power in Cyclone IV GX, Stratix IV, and Stratix V devices; however, this approach might increase the number of LEs required to implement the memory and affect design performance.

The Intel Quartus Prime software automatically chooses the best design memory configuration for optimal power. However, you can set the **MAXIMUM_DEPTH** parameter for memory modules during the IP core instantiation.

Figure 16. RAM 2-Port Maximum Depth



System: test **Path:** ram_2port_0

RAM: 2-PORT Intel FPGA IP
ram_2port

How many words of memory?: 32

Use different data widths on different ports

Read/Write Ports

How wide should the 'q_a' output bus be?: 8
How wide should the 'data_a' input bus be?: 8
How wide should the 'q_b' output bus be?: 8

What should the memory block type be?

Ram Block Type: Auto
Set the maximum block depth to: Auto words

Auto
32
64
128
256
512
1024
2048
4096
8192
16384

LCs Additional Options

How should the memory be implemented?: ault logic cell style
tix M512 emulation logic cell style

2 Info Messages

Related Information

- Power-Driven Compilation on page 30
- Clock Power Management on page 37
- Clocking Modes and Clock Enable
  In *Embedded Memory (RAM: 1-PORT, RAM: 2-PORT, ROM: 1-PORT, and ROM: 2-PORT) User Guide*

2.5.1.1.1. Memory Power Reduction Example

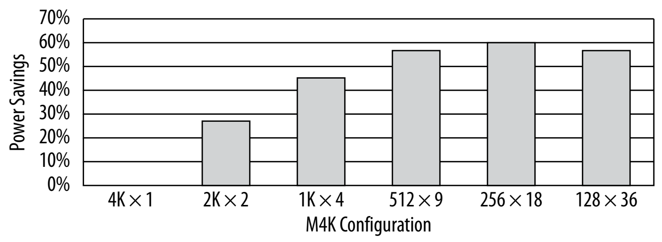
Power usage measurements for a 4K × 36 simple dual-port memory implemented using multiple M4K blocks in a Stratix device. For each implementation, the M4K blocks are configured with a different memory depth.

Table 10. 4K × 36 Simple Dual-Port Memory Implemented Using Multiple M4K Blocks

M4K Configuration	Number of M4K Blocks	ALUTs
4K × 1 (Default setting)	36	0
2K × 2	36	40
1K × 4	36	62
512 × 9	32	143
256 × 18	32	302
128 × 36	32	633

Using the `MAXIMUM_DEPTH` parameter can save power. For all implementations, a user-provided read enable signal is present to indicate when read data is required. Using this power-saving technique can reduce power consumption by as much as 60%.

Figure 17. Power Savings Using the MAXIMUM_DEPTH Parameter



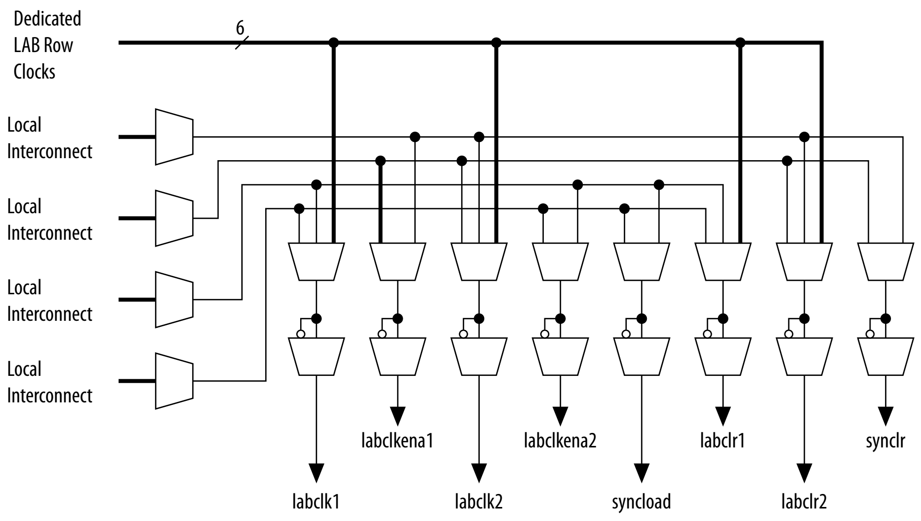
As the memory depth becomes more shallow, memory dynamic power decreases because unaddressed M4K blocks can be shut off using a decoded combination of address bits and the read enable signal. For a 128-deep memory block, power used by the extra LEs starts to outweigh the power gain achieved by using a more shallow memory block depth. The power consumption of the memory blocks and associated LEs depends on the memory configuration.

Note: The SOPC Builder and Platform Designer (Standard) system do not offer specific power savings control for on-chip memory block. There is no read enable, write enable, or clock enable that you can enable in the on-chip RAM megafunction to shut down the RAM block in the SOPC Builder and Platform Designer (Standard) system.

2.5.1.2. LAB Clock Power

Another contributor to clock power consumption are LAB clocks, which distribute clock to the registers within a LAB. LAB clock power can be the dominant contributor to overall clock power.

Figure 18. LAB-Wide Control Signals



To reduce LAB-wide clock power consumption without disabling the entire clock tree, use the LAB-wide clock enable to gate the LAB-wide clock. The Intel Quartus Prime software automatically promotes register-level clock enable signals to the LAB-level. A shared gated clock controls all registers within an LAB that share a common clock and clock enable. To take advantage of these clock enables, use a clock enable construct in the relevant HDL code for the registered logic.

2.5.1.2.1. LAB-Wide Clock Enable Example

This VHDL code makes use of a LAB-wide clock enable. This clock-gating logic is automatically turned into an LAB-level clock enable signal.

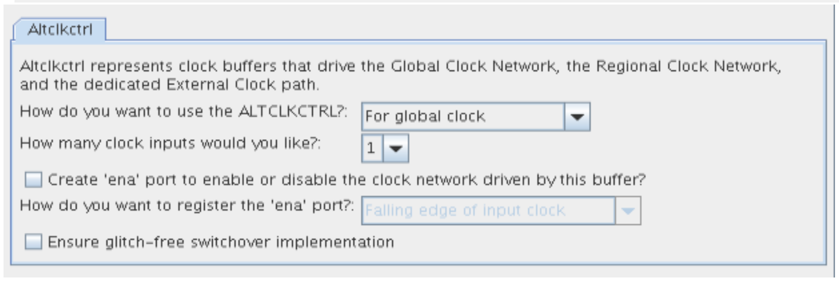
```
IF clk'event AND clock = '1' THEN
    IF logic_is_enabled = '1' THEN
        reg <= value;
    ELSE
        reg <= reg;
    END IF;
END IF;
```

2.5.1.3. Clock Enables

Use clock enables instead of gated clocks:

```
assign clk_gate = clk1 & gateA & gateB;
always @ (posedge clk_gate) begin
   sr[N-1:1] <= sr[N-2:0];
   sr[0]<=din1;
end
```

```
assign enable = gateA & gateB;
always @(posedge clk2) begin
   if (enable) begin
      sr[N-1:1] <= sr[N-2:0];
      sr[0]<=din2;
   end
end
```


Altclkctrl

Altclkctrl represents clock buffers that drive the Global Clock Network, the Regional Clock Network, and the dedicated External Clock path.

How do you want to use the ALTCLKCTRL?: For global clock

How many clock inputs would you like?: 1

☐ Create 'ena' port to enable or disable the clock network driven by this buffer?

How do you want to register the 'ena' port?: Falling edge of input clock

☐ Ensure glitch-free switchover implementation

Reduce LAB-wide clock power consumption without disabling the entire clock tree, use the LAB-wide clock enable to gate the LAB-wide clock.

```
always @(posedge clk)
begin
   if (ena)
      temp <= dataa;
   else
      temp <= temp;
   end
end
```

2.5.1.4. Global Signals

Intel FPGAs have different kinds of global signal resources available. Global signals can span the entire chip or smaller regions. Choose the clock networks that can cover all the fanout on a specific domain. For example, you can reduce clock power by switching from a clock network that spans the entire chip to one quarter of the chip, provided all the fanout for that clock is within that region of the chip.

Related Information

Device Page (Settings Dialog Box)
In *Intel Quartus Prime Help*

2.5.1.4.1. Viewing Clock Details in the Chip Planner

1. Open the Chip Planner (**Tools ➤ Chi Planner**).
2. In the **Task** pane, under **Clock Reports**, double-click **Report Clock Details**.

Figure 21. Chip Planner Task Pane

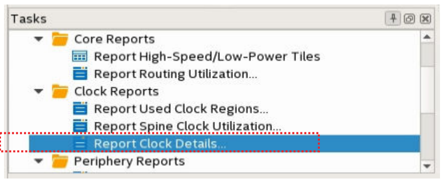
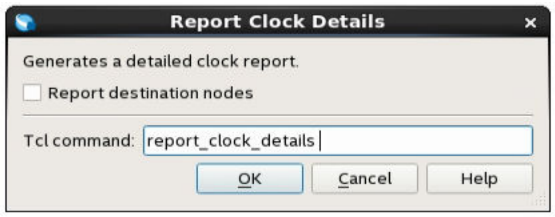

Tasks
Core Reports
Report High-Speed/Low-Power Tiles
Report Routing Utilization...
Clock Reports
Report Used Clock Regions...
Report Spine Clock Utilization...
Report Clock Details...
Periphery Reports

Figure 22. Report Clock Details


Report Clock Details

Generates a detailed clock report.

☐ Report destination nodes

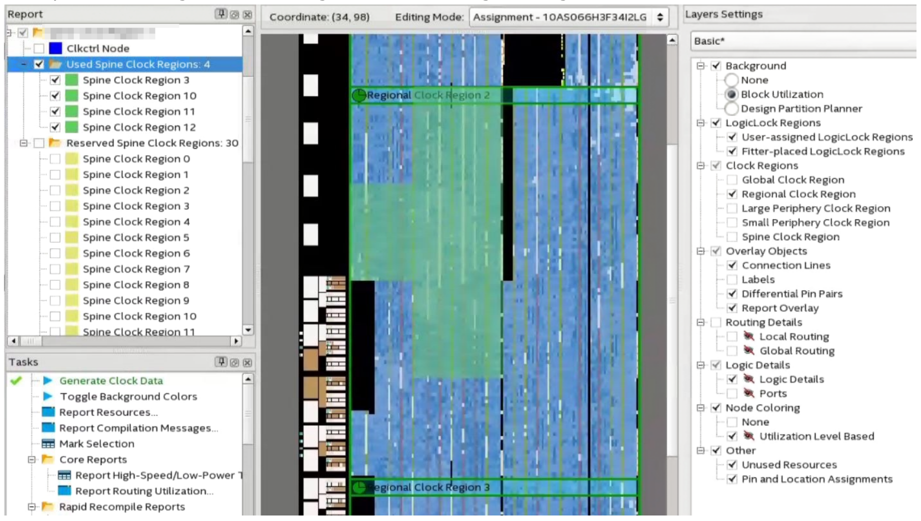
Tcl command: report_clock_details

OK Cancel Help

3. Click **OK**.
The **Report** pane generates the **Clock** folder.
4. Expand the **Clock** folder and select **Used spine clock regions** to highlight on the Chip planner.
5. In the **Layers Settings** pane, turn on **Regional/Periphery clock region** to see whether used spine clock regions are within.

Figure 23. Clock Highlight in Chip Planner

This example uses a Regional clock Region instead of a global signal.



2.5.1.5. Merge Clocks

Evaluate the possibility of merging clocks and PLLs in the design.

Design	2clks & 2PLLs	1 Clk & 1 PLL
Oc_dma_stamp25	6.079W	5.46W

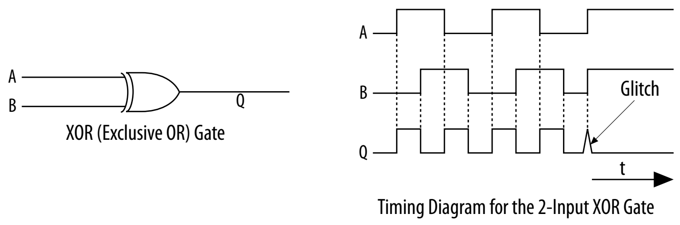
- 2clks & 2PLLs
  Clk1:350Mhz, Fanout 46788
  Clk2: 365Mhz, Fanout 2450
- 1Clk & 1PLL
  Merge clks
  clk: 365Mhz, Fanout 51277

2.5.2. Pipelining and Retiming

Glitches are unnecessary and unpredictable temporary logic switches at the output of combinational logic. Designs with glitches consume more power, because of faster switching activity. A glitch usually occurs when there is a mismatch in input signal timing, leading to unequal propagation delay.

For example, consider a 2-input XOR gate where one input changes from `1` to `0`, and moments later the other input changes from `0` to `1` For a short time, both inputs become `1` (high), resulting in `0` (low) at the output of the XOR gate. Then, when the second input transition takes place, the XOR gate output becomes `1` (high). Therefore, before the output becomes stable, the input delay produces a glitch in the output.

Figure 24. XOR Gate Showing Glitch at the Output

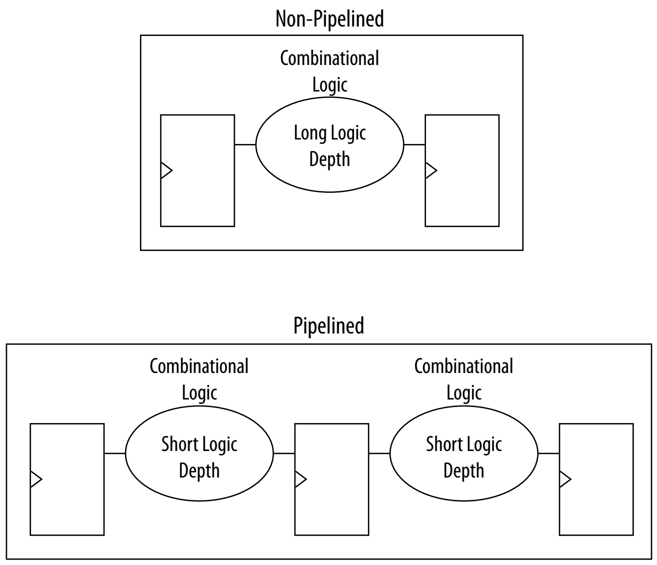


A glitch can propagate to subsequent logic and create unnecessary switching activity, increasing power consumption. Circuits with many XOR functions, such as arithmetic circuits or cyclic redundancy check (CRC) circuits, tend to have many glitches if there are several levels of combinational logic between registers.

Registers stop glitches from propagating through combinational paths. Pipelining is a technique that breaks combinational paths by inserting registers. By reducing logic-level numbers between registers, pipelining can result in higher clock speed operations. However, pipelining increases the latency of a circuit in terms of the number of clock cycles to a first result.

The following figure shows how pipelining breaks a long combinational path.

Figure 25. Pipelining Example



This reduction in switching activity lowers power dissipation in combinational logic. However, for designs with few glitches, pipelining can increase power consumption by adding unnecessary registers. Pipelining can also increase resource utilization. Benchmark data shows that pipelining can reduce dynamic power consumption by as much as 30% in Cyclone and Stratix devices.

2.5.3. Architectural Optimization

Design-level architectural optimizations allow you to take advantage of device architecture features. These features include dedicated memory, DSPs, or multiplier blocks that can perform memory or arithmetic-related functions. You can reduce power consumption by choosing blocks in place of LUTs. For example, you can build large shift registers from RAM-based FIFO buffers instead of building the shift registers from the LE registers.

The Stratix device family allows you to efficiently target small, medium, and large memories with the TriMatrix memory architecture. Each TriMatrix memory block is optimized for a specific function. M512 memory blocks are more power-efficient than the distributed memory structures in some competing FPGAs. With M4K memory blocks you can implement buffers for a wide variety of applications, including processor code storage, large look-up table implementation, and large memory applications. The M-RAM blocks are useful in applications when storing a large volume of data on-chip is necessary. Effective utilization of these memory blocks can have a significant impact on power reduction in the design.

The latest Stratix and Cyclone device families have configurable M9K memory blocks that provide memory functions such as RAM, FIFO buffers, and ROM.

2.5.4. I/O Power Guidelines

The Power Analyzer calculates I/O power using the default capacitive load set for the I/O standard in the **Capacitive Loading** page of the **Device and Pin Options** dialog box. Any other components defined in the board trace model are not taken into account for the power measurement. For Cyclone IV GX, Stratix IV, and Stratix V devices, Advanced I/O Timing considers the full board trace model.

Nonterminated I/O Standards

Nonterminated I/O standards such as LVTTL and LVCMOS have a rail-to-rail output swing. The voltage difference between logic-high and logic-low signals at the output pin is equal to the V_{CCIO} supply voltage. If the capacitive loading at the output pin is known, the following expression determines the dynamic power consumed in the I/O buffer:

$$P = \frac{F\,C\,V^2}{2}$$

where:

- F is the output transition frequency
- C is the total load capacitance being switched
- V is equal to V_{CCIO} supply voltage

Because of the quadratic dependence on V_{CCIO}, lower voltage standards consume significantly less dynamic power.

Transistor-to-transistor logic (TTL) I/O buffers consume very little static power. As a result, the total power that a LVTTL or LVCMOS output consumes is highly dependent on load and switching frequency.

Resistively Terminated I/O Standards

In resistively terminated I/O standards like SSTL and HSTL, the output load voltage swings by a small amount around a bias point. The dynamic power equation above is valid as well, but V is the actual load voltage swing. This voltage is much smaller than V_{CCIO}, resulting in lower dynamic power when comparing to nonterminated I/O under similar conditions.

Resistively terminated I/O standards dissipate significant static (frequency-independent) power, because the I/O buffer is constantly driving current into the resistive termination network. However, the lower dynamic power of these I/O standards means they often have lower total power than LVCMOS or LVTTL for high-frequency applications. As a best practice, when using resistively terminated standards choose the lowest drive strength I/O setting that meets the speed and waveform requirements to minimize I/O power.

You can save a small amount of static power by connecting unused I/O banks to the lowest possible V_{CCIO} voltage.

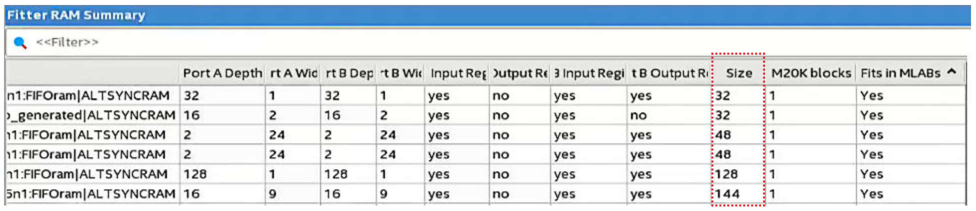
Related Information

Stratix Series FPGA I/O Connectivity

2.5.5. Memory Optimization (M20K/MLAB)

M20K memory blocks represent a big part of the power consumption in a design. The Fitter RAM Summary Report displays the utilization of the memory blocks in different parts of the design.

Figure 26. Fitter RAM Summary Report



Fitter RAM Summary

<<Filter>>

	Port A Depth	rt A Wid	rt B Dep	rt B Wid	Input Reg	Output Re	3 Input Regi	t B Output R	Size	M20K blocks	Fits in MLABs
n1:FIFOram|ALTSYNCRAM	32	1	32	1	yes	no	yes	yes	32	1	Yes
_generated|ALTSYNCRAM	16	2	16	2	yes	no	yes	no	32	1	Yes
11:FIFOram|ALTSYNCRAM	2	24	2	24	yes	no	yes	yes	48	1	Yes
11:FIFOram|ALTSYNCRAM	2	24	2	24	yes	no	yes	yes	48	1	Yes
n1:FIFOram|ALTSYNCRAM	128	1	128	1	yes	no	yes	yes	128	1	Yes
5n1:FIFOram|ALTSYNCRAM	16	9	16	9	yes	no	yes	yes	144	1	Yes

Some guidelines to optimize the use of memories are:

- Port shallow memories from M20K to MLAB.

  For example, implement in HDL with `ramstyle` attribute:

  ```
  (* ramstyle = "MLAB" *) reg [0:7] my_ram[0:63];
  ```

- Avoid read-during-write behavior and set to **Don’t care** (at the HDL level) wherever possible.

  Read-during-write behavior impact the power of single-port and bidirectional dual-port RAMs. **Don’t care** allows an optimization that sets the read-enable signal to the inversion of the existing write-enable signal (if one exists). This allows the core of the RAM to shut down, which prevents switching, saving a significant amount of power.

- Pack input/output registers in M20K.

2.5.5.1. Implementation

Table 11. Single-port Embedded Memory Configurations for Devices

This table lists the maximum configurations that single-port RAM and ROM modes support.

Memory Block	Depth (bits)	Programmable Width
MLAB	32	x16, x18, or x20
	64 [1]	x8, x9, x10
M20K	512	x40, x32
	1K	x20, x16
	2K	x10, x8
	4K	x5, x4
	8K	x2
	16K	x1

[1] Supported through software emulation and consumes additional MLAB blocks.

Figure 27. Power numbers from EPE

						Port A				Port B					Thermal Power (W)		
Module	RAM Type	# RAM Blocks	Data Width	RAM Depth	RAM Mode	Clock Freq (MHz)	Enable %	Read %	Write %	Clock Freq (MHz)	Enable %	Read %	Write %	Toggle %	Routing	Block	Total
	M20K	1	40	32	Simple Dual Port	384.0	100%	0%	100%	384.0	100%	100%	0%	12.5%	0.000	0.005	0.005
	MLAB	2	20	32	Simple Dual Port	384.0	100%	0%	100%	384.0	100%	100%	0%	12.5%	0.001	0.002	0.002

2.5.5.2. Rd/Wr Enables

Dedicated RAM blocks dissipate most energy whenever the RAM is accessed for a read or write cycle. You can save power by adding Read/Write enable.

						Port A				Port B					Thermal Power (W)		
Module	RAM Type	# RAM Blocks	Data Width	RAM Depth	RAM Mode	Clock Freq (MHz)	Enable %	Read %	Write %	Clock Freq (MHz)	Enable %	Read %	Write %	Toggle %	Routing	Block	Total
168	M20K	144	40	512	Simple Dual Port	384.0	100%	0%	100%	384.0	100%	100%	0%	12.5%	0.046	0.668	0.714
168	M20K	144	40	512	Simple Dual Port	384.0	100%	0%	50%	384.0	100%	50%	0%	12.5%	0.023	0.362	0.385

2.5.6. DDR Memory Controller Settings

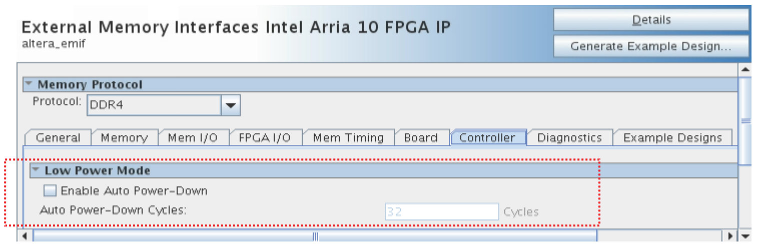
The Intel Arria 10 EMIF IP DDR3 controller provides low power mode options. These options put DDR in Power saving mode when the controller is idle, providing power savings on External Memory DDR. These options are **Enable Auto Power-Down** and **Auto Power-Down Cycles**.

Power-Down Mode

Enable Auto Power-Down directs the controller to place the memory device in power-down mode after a specific number of idle controller clock cycles. You can configure the idle wait time. All ranks must be idle to enter auto power-down.

Auto Power-Down Cycles Number of cycles the controller must be IDLE before entering power down state. You determine the number based on the traffic pattern. If the number is too small, the control enters power down too frequently, affecting efficiency. The Intel Arria 10 device family supports from 1 to 65534 cycles.

Figure 28. Intel Arria 10 EMIF Controller Parameters


External Memory Interfaces Intel Arria 10 FPGA IP
altera_emif

Memory Protocol
Protocol: DDR4

Low Power Mode
Enable Auto Power-Down
Auto Power-Down Cycles: 32 Cycles

Self-Refresh

Directs the Controller to self-refresh when not sending traffic for very long period. Self-refresh takes more time compared to power down, but the power saving is higher than power down.

Related Information

Intel Arria 10 EMIF IP DDR3 Parameters: Controller
In *External Memory Interfaces Intel Arria 10 FPGA IP User Guide*

2.5.7. DSP Implementation

When you maximize the packing of DSP blocks, you reduce Logic Utilization, power consumption, and increase efficiency. The HDL coding style grants you control of the DSP resources available in the FPGA.

Example 1. Implement Multiplier + Accumulator in 1 DSP

```
always @ (posedge clk)
begin
   if (ena)
   begin
      dataout <= dataa * datab + datac * datad;
   end
end
```

Example 2. Implement multiplication in 2 DSPs and the adder in LABs

```
always @ (posedge clk)
begin
   if (ena)
   begin
      mult1 <= dataa * datab;
      mult2 <= datac * datad;
   end
end
always @(posedge clk)
begin
   if (ena)
   begin
      dataout <= mult1 + mult2
   end
end
```

Related Information

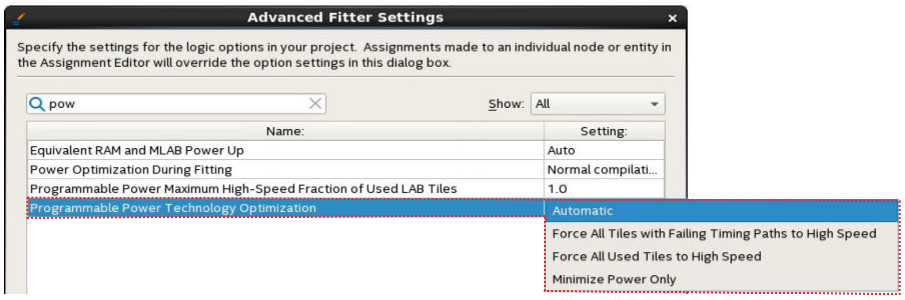
Inferring Multipliers and DSP Functions
In *Intel Quartus Prime Standard Edition User Guide: Design Recommendations*

2.5.8. Reducing High-Speed Tile (HST) Usage

High-Speed tiles are available in Stratix V and Intel Arria 10 device families.

1. In the **Advanced Fitter Settings** pane, The **Programmable Power Technology Optimization** logic option controls how the fitter configures tiles to operate in high-speed mode or low-power mode. Select **Minimize Power Only**.

Figure 29. Programmable Power Technology Optimization



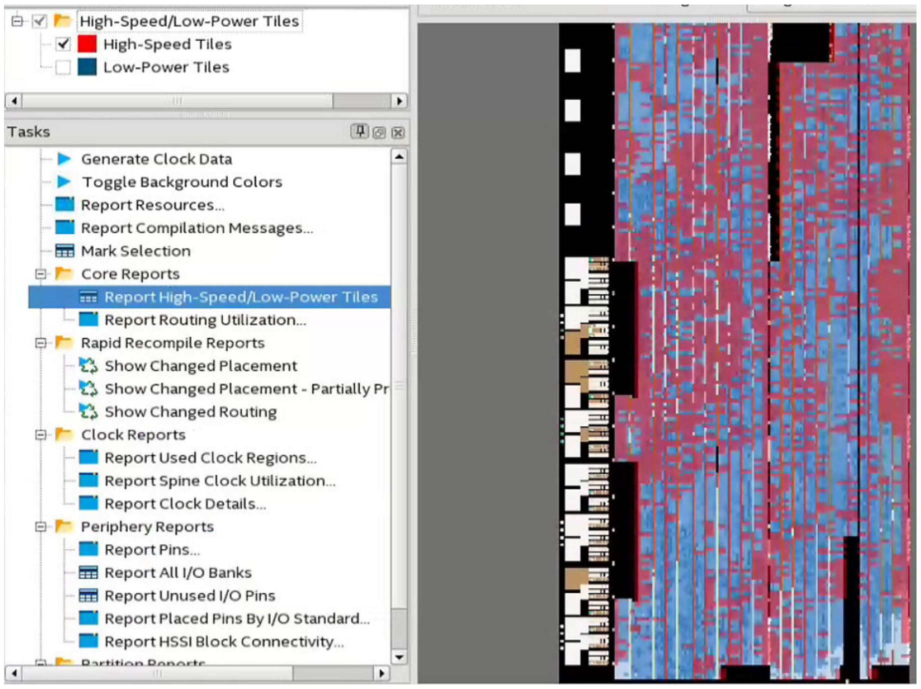
Specify the settings for the logic options in your project. Assignments made to an individual node or entity in the Assignment Editor will override the option settings in this dialog box.

Name:	Setting:
Equivalent RAM and MLAB Power Up	Auto
Power Optimization During Fitting	Normal compilati...
Programmable Power Maximum High-Speed Fraction of Used LAB Tiles	1.0
Programmable Power Technology Optimization	Automatic

Automatic
Force All Tiles with Failing Timing Paths to High Speed
Force All Used Tiles to High Speed
Minimize Power Only

2. Identify entity modules that use HST by plotting entity modules and HST heatmap on the Chip Planner and modify the floorplan to reduce usage.

Figure 30. Entity Modules and HST Heatmap on the Chip Planner



2.5.9. Unused Transceiver Channels

Transceivers in the device degrade over time unless you preserve them. The Intel Quartus Prime software generates a warning message if a design contains unused XCVRs.

You do not need to preserve transceivers under 8Gbps. For transceivers over 8Gbps, the best practice is to preserve if there is a possibility for future usage. Otherwise, you can turn the transceivers off. You enable unused transceivers through dynamic reconfiguration or a new device programming file.

2.5.10. Periphery Power reduction XCVR Settings

2.5.10.1. Transceiver Settings

- Use min VCCR/T possible (depending on data rate).
- Certain devices have DFE ON by default. If possible, turn off the channel, This depends on the how lossy is the channel.
- Turn off PDN compensation.
  This setting induces jitter, which is necessary to check system tolerance.
- Use one equalizer stage.

DFE	Adaptation	Equalizer Stage	Transmitter High-Speed Compensation
Disabled	Disabled	Non-S1 Mode	Disabled
Disabled	Disabled	Non-S1 Mode	Enabled
Disabled	Disabled	N/A	Enabled

2.5.10.2. I/O Current Strength

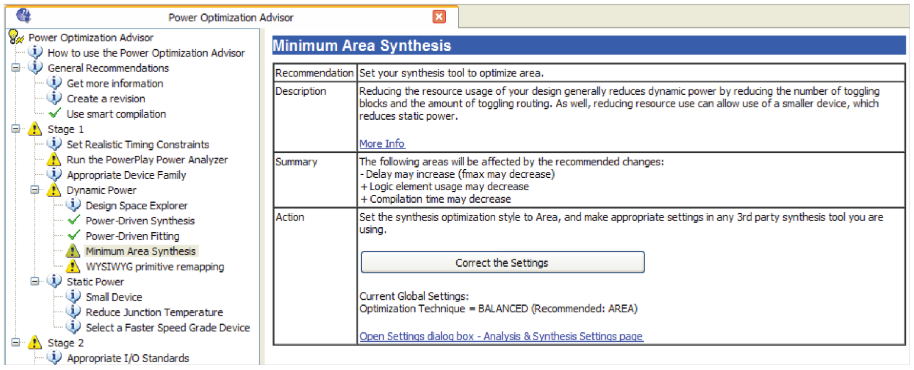
As a best practice, choose a low voltage I/O standard and the lowest drive strength that meets the speed requirements.

2.6. Power Optimization Advisor

The Intel Quartus Prime Power Optimization Advisor provides advice and recommendations based on the current design project settings and assignments. You run the Advisor after the Power Analyzer.

Figure 31. Power Optimization Advisor

Power Optimization Advisor after compiling a design that is not fully optimized for power.


Power Optimization Advisor
Power Optimization Advisor
How to use the Power Optimization Advisor
General Recommendations
Get more information
Create a revision
Use smart compilation
Stage 1
Set Realistic Timing Constraints
Run the PowerPlay Power Analyzer
Appropriate Device Family
Dynamic Power
Design Space Explorer
Power-Driven Synthesis
Power-Driven Fitting
Minimum Area Synthesis
WYSIWYG primitive remapping
Static Power
Small Device
Reduce Junction Temperature
Select a Faster Speed Grade Device
Stage 2
Appropriate I/O Standards
Correct the Settings

Minimum Area Synthesis

Recommendation	Set your synthesis tool to optimize area.
Description	Reducing the resource usage of your design generally reduces dynamic power by reducing the number of toggling blocks and the amount of toggling routing. As well, reducing resource use can allow use of a smaller device, which reduces static power. More Info
Summary	The following areas will be affected by the recommended changes: - Delay may increase (fmax may decrease) + Logic element usage may decrease + Compilation time may decrease
Action	Set the synthesis optimization style to Area, and make appropriate settings in any 3rd party synthesis tool you are using. Correct the Settings Current Global Settings: Optimization Technique = BALANCED (Recommended: AREA) Open Settings dialog box - Analysis & Synthesis Settings page

The Power Optimization Advisor organizes the recommendations into stages that suggest the implementation order. Each recommendation includes a description, summary of the effect of the recommendation, and the action required to make the appropriate setting.

An icon indicates whether each recommended setting is made in the current project. Checkmark icons appear next to recommendations that are already implemented, warning icons appear next to recommendations that are not followed for this compilation. Information icons indicate general suggestions.

Recommendations include a link to the location in the Intel Quartus Prime GUI where you can change the setting. After implementing the recommended changes, recompile your design. You can verify power results with the Power Analyzer.

Related Information

Advisors in the Intel Quartus Prime Software
In *Intel Quartus Prime Help*

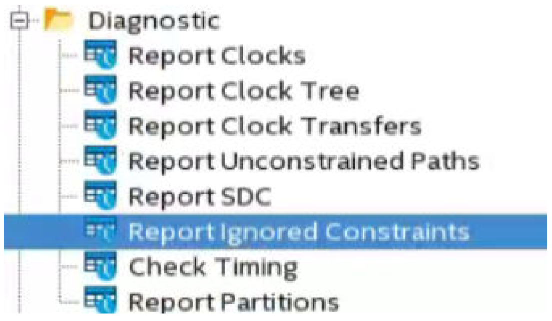
2.6.1. Set Realistic Timing Constraints

Timing requirements are too high, the Compiler increases HST Usage. In addition, the Fitter efforts focus more in timing than power optimization.

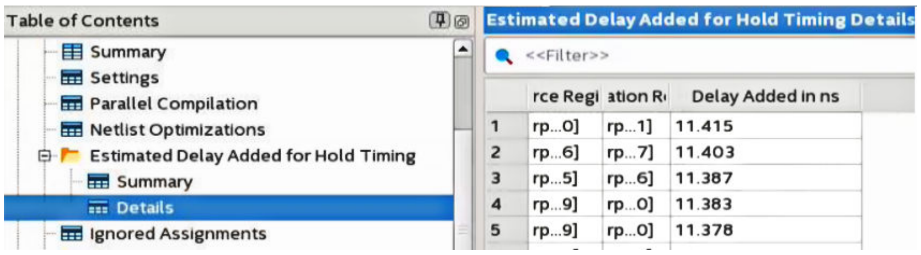
2.6.1.1. Find Timing Information

- To find False or Multi-Cycle Paths, click **Report Ignored Constraints** in the Timing Analyzer **Tasks** pane.

Figure 32. Report Ignored Constraints



- To see a list of the 10 paths with highest delay in the design, in the **Reports** pane find **Fitter Summary Report ➤ Estimate Delay Added for Hold Timing ➤ Details**.



	rce Regi	ation R	Delay Added in ns
1	rp...0]	rp...1]	11.415
2	rp...6]	rp...7]	11.403
3	rp...5]	rp...6]	11.387
4	rp...9]	rp...0]	11.383
5	rp...9]	rp...0]	11.378

2.6.2. Appropriate Device Family

Choose a device family with the dynamic and static power characteristics best suited to your application.

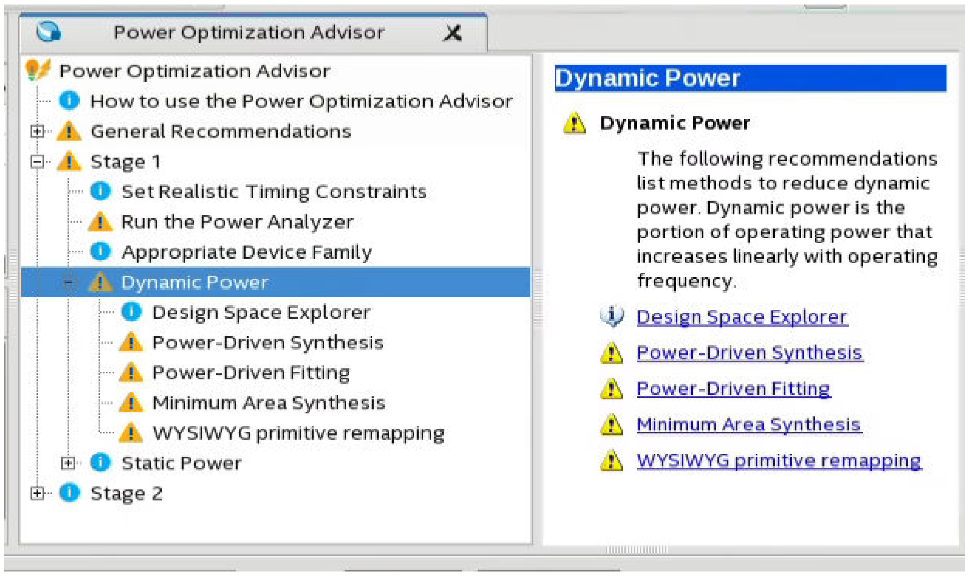
Related Information

- Device Selection on page 26
- Device Page (Settings Dialog Box)
  In *Intel Quartus Prime Help*

2.6.3. Dynamic Power

The recommendations in this section can reduce dynamic power.

Figure 33. Dynamic Power Recommendations in the Power Optimization Advisor



The following recommendations list methods to reduce dynamic power. Dynamic power is the portion of operating power that increases linearly with operating frequency.

Related Information

- Design Space Explorer II for Power-Driven Optimization on page 29
- Power-Driven Synthesis on page 30
- Power-Driven Fitter on page 33
- Area-Driven Synthesis on page 34

2.6.4. Static Power

The recommendations in this section can reduce static power dissipation. Static power is the frequency independent power that a design dissipates, even when the design clocks are stopped.

Small Device

Use the smallest device which can fit your design.

Related Information

- Device Selection on page 26
- Device Page (Settings Dialog Box)
  In *Intel Quartus Prime Help*

2.6.5. Appropriate I/O Standards

Choose appropriate I/O Standards to minimize design power.

Related Information

I/O Power Guidelines on page 45

2.6.6. Use RAM Blocks

Implement RAMs and medium to large shift registers in RAM blocks instead of logic cell registers.

Related Information

Memory Optimization (M20K/MLAB) on page 46

2.6.7. Shut Down RAM Blocks

Use the clock enable, read enable and write enable ports on RAM blocks to shut them down during cycles in which the RAM is not read or written. If your design does not depend on a specific read result when reading and writing the same address, then specify "don't care" for the read-during-write parameter in the RAM IP Catalog.

Related Information

- Clock Enable in Memory Blocks on page 38
- Memory Optimization (M20K/MLAB) on page 46

2.6.8. Clock Enables on Logic

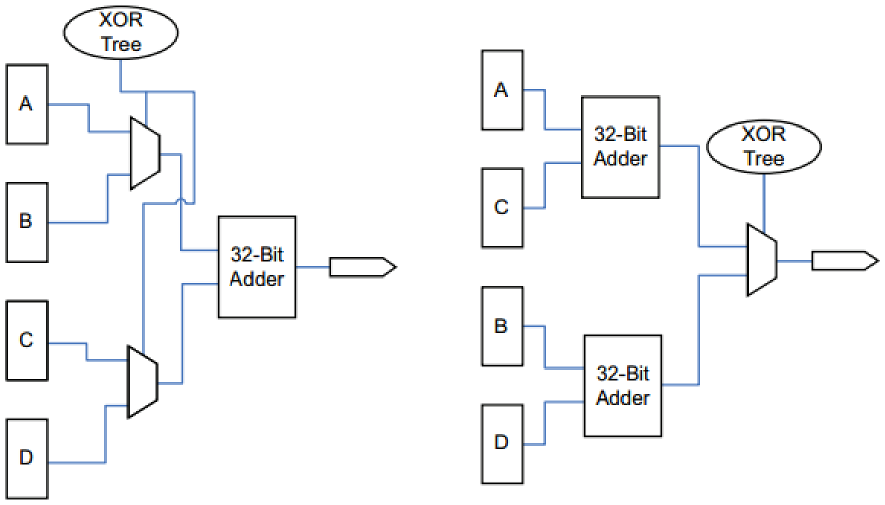
Another technique for power reduction is gating clocks when the logic does not require them. Even though you can build clock-gating logic, this approach can generate clock glitches in FPGAs using ALMs or LEs.

2.6.9. Pipeline Logic to Reduce Glitching

Long chains of cascaded logic blocks can create glitches due to path delay differences between the input signals. Inserting Flip-Flops to cut these long chains terminates the propagation of glitches to consecutive logic cells.

Circuits that heavily use of XIO functions (for example, Cyclic redundancy check) tend to glitch significantly when cascaded. Add pipeline registers or re-architect to reduce signal toggling

Example 3. Glitch Prone Design



Related Information

Pipelining and Retiming on page 43

2.7. Power Optimization Revision History

The following revision history applies to this chapter:

Table 12. Document Revision History

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1.0	• Initial release in Intel Quartus Prime Standard Edition User Guide. • Added topic: *Factors Affecting Power Consumption*, moved from chapter: *Power Analysis* • Extended content about Power Optimization Advisor with a description of recommendations. • Added design guidelines: *DDR Memory Controller Settings*, *DSP Implementation*, *Reducing High-Speed Tile (HST) Usage*, *Unused Transceiver Channels*, *Periphery Power reduction XCVR Settings*
2018.06.11	18.0.0	• Moved general information about the Design Space Explorer (DSE II) to the *Design Optimization Overview* chapter, left a section about using DSE II for Power-Driven Optimization.
2018.05.07	18.0.0	• Moved general information about the Design Space Explorer (DSE II) to the *Design Optimization Overview* chapter, left a section about using DSE II for Power-Driven Optimization.
2016.10.31	16.1.0	• Removed statement of support for gate-level timing simulation.
2015.11.02	15.1.0	Changed instances of *Quartus II* to *Intel Quartus Prime*. • Updated screenshot for DSE II GUI. • Added information about remote hosts for DSE II.
2014.12.15	14.1.0	• Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings. • Updated DSE II GUI and optimization settings.

continued...

Document Version	Intel Quartus Prime Version	Changes
2014.06.30	14.0.0	Updated the format.
May 2013	13.0.0	Added a note to "Memory Power Reduction Example" on Qsys and SOPC Builder power savings limitation for on-chip memory block.
June 2012	12.0.0	Removed survey link.
November 2011	10.0.2	Template update.
December 2010	10.0.1	Template update.
July 2010	10.0.0	• Was chapter 11 in the 9.1.0 release • Updated Figures 14-2, 14-3, 14-6, 14-18, 14-19, and 14-20 • Updated device support • Minor editorial updates
November 2009	9.1.0	• Updated Figure 11-1 and associated references • Updated device support • Minor editorial update
March 2009	9.0.0	• Was chapter 9 in the 8.1.0 release • Updated for the Quartus II software release • Added benchmark results • Removed several sections • Updated Figure 13–1, Figure 13–17, and Figure 13–18
November 2008	8.1.0	• Changed to 8½" × 11" page size • Changed references to altsyncram to RAM • Minor editorial updates
May 2008	8.0.0	• Added support for Stratix IV devices • Updated Table 9–1 and 9–9 • Updated "Architectural Optimization" on page 9–22 • Added "Dynamically-Controlled On-Chip Terminations" on page 9–26 • Updated "Referenced Documents" on page 9–29 • Updated references

Related Information

Documentation Archive
For previous versions of the *Intel Quartus Prime Handbook*, search the documentation archives.

A. Intel Quartus Prime Standard Edition User Guides

Refer to the following user guides for comprehensive information on all phases of the Intel Quartus Prime Standard Edition FPGA design flow.

Related Information

- Intel Quartus Prime Standard Edition User Guide: Getting Started
  Introduces the basic features, files, and design flow of the Intel Quartus Prime Standard Edition software, including managing Intel Quartus Prime Standard Edition projects and IP, initial design planning considerations, and project migration from previous software versions.
- Intel Quartus Prime Standard Edition User Guide: Platform Designer
  Describes creating and optimizing systems using Platform Designer (Standard), a system integration tool that simplifies integrating customized IP cores in your project. Platform Designer (Standard) automatically generates interconnect logic to connect intellectual property (IP) functions and subsystems.
- Intel Quartus Prime Standard Edition User Guide: Design Recommendations
  Describes best design practices for designing FPGAs with the Intel Quartus Prime Standard Edition software. HDL coding styles and synchronous design practices can significantly impact design performance. Following recommended HDL coding styles ensures that Intel Quartus Prime Standard Edition synthesis optimally implements your design in hardware.
- Intel Quartus Prime Standard Edition User Guide: Design Compilation
  Describes set up, running, and optimization for all stages of the Intel Quartus Prime Standard Edition Compiler. The Compiler synthesizes, places, and routes your design before generating a device programming file.
- Intel Quartus Prime Standard Edition User Guide: Design Optimization
  Describes Intel Quartus Prime Standard Edition settings, tools, and techniques that you can use to achieve the highest design performance in Intel FPGAs. Techniques include optimizing the design netlist, addressing critical chains that limit retiming and timing closure, and optimization of device resource usage.
- Intel Quartus Prime Standard Edition User Guide: Programmer
  Describes operation of the Intel Quartus Prime Standard Edition Programmer, which allows you to configure Intel FPGA devices, and program CPLD and configuration devices, via connection with an Intel FPGA download cable.
- Intel Quartus Prime Standard Edition User Guide: Partial Reconfiguration
  Describes Partial Reconfiguration, an advanced design flow that allows you to reconfigure a portion of the FPGA dynamically, while the remaining FPGA design continues to function. Define multiple personas for a particular design region, without impacting operation in other areas.

- Intel Quartus Prime Standard Edition User Guide: Third-party Simulation
  Describes RTL- and gate-level design simulation support for third-party simulation tools by Aldec*, Cadence*, Mentor Graphics*, and Synopsys* that allow you to verify design behavior before device programming. Includes simulator support, simulation flows, and simulating Intel FPGA IP.
- Intel Quartus Prime Standard Edition User Guide: Third-party Synthesis
  Describes support for optional synthesis of your design in third-party synthesis tools by Mentor Graphics*, and Synopsys*. Includes design flow steps, generated file descriptions, and synthesis guidelines.
- Intel Quartus Prime Standard Edition User Guide: Debug Tools
  Describes a portfolio of Intel Quartus Prime Standard Edition in-system design debugging tools for real-time verification of your design. These tools provide visibility by routing (or “tapping”) signals in your design to debugging logic. These tools include System Console, Signal Tap logic analyzer, Transceiver Toolkit, In-System Memory Content Editor, and In-System Sources and Probes Editor.
- Intel Quartus Prime Standard Edition User Guide: Timing Analyzer
  Explains basic static timing analysis principals and use of the Intel Quartus Prime Standard Edition Timing Analyzer, a powerful ASIC-style timing analysis tool that validates the timing performance of all logic in your design using an industry-standard constraint, analysis, and reporting methodology.
- Intel Quartus Prime Standard Edition User Guide: Power Analysis and Optimization
  Describes the Intel Quartus Prime Standard Edition Power Analysis tools that allow accurate estimation of device power consumption. Estimate the power consumption of a device to develop power budgets and design power supplies, voltage regulators, heat sink, and cooling systems.
- Intel Quartus Prime Standard Edition User Guide: Design Constraints
  Describes timing and logic constraints that influence how the Compiler implements your design, such as pin assignments, device options, logic options, and timing constraints. Use the Pin Planner to visualize, modify, and validate all I/O assignments in a graphical representation of the target device.
- Intel Quartus Prime Standard Edition User Guide: PCB Design Tools
  Describes support for optional third-party PCB design tools by Mentor Graphics* and Cadence*. Also includes information about signal integrity analysis and simulations with HSPICE and IBIS Models.
- Intel Quartus Prime Standard Edition User Guide: Scripting
  Describes use of Tcl and command line scripts to control the Intel Quartus Prime Standard Edition software and to perform a wide range of functions, such as managing projects, specifying constraints, running compilation or timing analysis, or generating reports.