## Contents

1. **Power Analysis** ............................................................................................................... 4  
   1.1. Power Analysis Tools.................................................................................................. 5  
   1.2. Running the Power Analyzer...................................................................................... 6  
   1.3. Specifying Power Analyzer Input.............................................................................. 7  
      1.3.1. Device Operating Condition Settings for Power Analysis.................................. 8  
      1.3.2. Specifying Signal Activity Data.................................................................... 9  
      1.3.3. Specifying the Default Toggle Rate............................................................. 15  
      1.3.4. Specifying Toggle Rates for Specific Nodes................................................... 16  
      1.3.5. Avoiding Simulation Node Name Match....................................................... 17  
   1.4. Viewing Power Analysis Reports............................................................................. 17  
   1.5. Power Analysis in Modular Design Flows.................................................................. 21  
      1.5.1. Complete Design Simulation Power Analysis Flow......................................... 22  
      1.5.2. Modular Design Simulation Power Analysis Flow........................................... 22  
      1.5.3. Multiple Simulation Power Analysis Flow...................................................... 23  
      1.5.4. Overlapping Simulation Power Analysis Flow................................................ 23  
      1.5.5. Partial Design Simulation Power Analysis Flow.............................................. 23  
      1.5.6. Vectorless Estimation Power Analysis Flow................................................... 24  
   1.6. Scripting Support................................................................................................. 24  
      1.6.1. Running the Power Analyzer from the Command–Line................................... 25  
   1.7. Power Analysis Revision History............................................................................. 26  

2. **Power Optimization** ...................................................................................................... 28  
   2.1. Factors Affecting Power Consumption..................................................................... 28  
      2.1.1. Design Activity and Power Analysis............................................................. 28  
      2.1.2. Device Selection...................................................................................... 28  
      2.1.3. Environmental Conditions.......................................................................... 29  
      2.1.4. Device Resource Usage............................................................................. 29  
      2.1.5. Signal Activity......................................................................................... 30  
   2.2. Design Space Explorer II for Power-Driven Optimization............................................ 31  
   2.3. Power-Driven Compilation..................................................................................... 31  
      2.3.1. Power-Driven Synthesis............................................................................ 31  
      2.3.2. Power-Driven Fitter.................................................................................. 34  
      2.3.3. Area-Driven Synthesis.............................................................................. 34  
      2.3.4. Gate-Level Register Retiming..................................................................... 35  
      2.3.5. Intel Quartus Prime Compiler Settings........................................................ 35  
      2.3.6. Assignment Editor Options........................................................................ 36  
   2.4. Design Guidelines................................................................................................ 37  
      2.4.1. Clock Power Management......................................................................... 37  
      2.4.2. Pipelining and Retiming............................................................................. 42  
      2.4.3. Architectural Optimization......................................................................... 43  
      2.4.4. I/O Power Guidelines................................................................................ 44  
      2.4.5. Dynamically Controlled On-Chip Terminations (OCT)..................................... 45  
      2.4.6. Memory Optimization (M20K/MLAB)........................................................... 45  
      2.4.7. DDR Memory Controller Settings............................................................... 47  
      2.4.8. DSP Implementation................................................................................ 47  
      2.4.9. Reducing High-Speed Tile (HST) Usage....................................................... 48  
      2.4.10. Unused Transceiver Channels................................................................... 49
## Contents

2.4.11. Periphery Power reduction XCVR Settings ................................................................. 49  
2.5. Power Optimization Advisor ........................................................................................ 50  
  2.5.1. Set Realistic Timing Constraints .............................................................................. 50  
  2.5.2. Appropriate Device Family ...................................................................................... 51  
  2.5.3. Dynamic Power ........................................................................................................ 51  
  2.5.4. Static Power ............................................................................................................. 52  
  2.5.5. Appropriate I/O Standards ....................................................................................... 52  
  2.5.6. Use RAM Blocks ....................................................................................................... 52  
  2.5.7. Shut Down RAM Blocks ......................................................................................... 53  
  2.5.8. Clock Enables on Logic ............................................................................................. 53  
  2.5.9. Pipeline Logic to Reduce Glitching .......................................................................... 53  
2.6. Power Optimization Revision History ........................................................................... 54  

3. Power Analysis and Optimization Document Archive ........................................................ 56  

A. Intel Quartus Prime Pro Edition User Guides .................................................................... 57
1. Power Analysis

Power consumption is a critical design consideration. When designing a PCB, you must determine the power consumption of the FPGA device to develop an accurate power budget, and to design the power supplies, voltage regulators, heat sink, and cooling system.

The Intel® Quartus® Prime Design Suite provides the Early Power Estimator (EPE) spreadsheet and Power Analyzer for estimating the power consumption in your design.

**Figure 1. Power Analyzer Tool**

- **Specify Power Input Files**
  - Use input file(s) to initialize toggle rates and static probabilities during power analysis
  - Add Power Input File(s)

- **Write Out Signal Activities**
  - Write out signal activities used during power analysis
  - Output file name

- **Write Out EPE File**
  - Write out Early Power Estimation file
  - Output file name

- **Specify Toggle Rates**
  - Default toggle rates for unspecified signals
    - Default toggle rate used for input I/O signals: 12.5%
    - Default toggle rate used for remaining signals
      - Use default value: 12.5%
      - Use vectorless estimation

- **Specify Cooling/Temp Solutions**
  - Cooling Solution and Temperature
    - 0%
    - 00:00:00

- **Start/Stop or View Reports**
Power estimation and analysis allows you to confirm that your design does not exceed thermal or power supply requirements throughout the design process:

- **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:* Do not use the results of the Power Analyzer as design specifications. You must also verify the actual power during device operation to account for actual environmental operating conditions.

**Related Information**
- Early Power Estimator Page
- Power Analyzer Support Resources

### 1.1. Power Analysis Tools

The Intel Quartus Prime Design Suite provides tools to analyze the power consumption of your FPGA design at different stages of the design process.

- Early Power Estimator (EPE) spreadsheet—estimates power consumption for power supply planning before compiling the design.
- Intel Quartus Prime Power Analyzer—estimates power consumption for a post-fit design, allowing you to establish guidelines for the power budget.

#### Figure 2. Estimation Accuracy for Different Inputs and Power Analysis Tools

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

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>EPE</th>
<th>Intel Quartus Prime Power Analyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>When to use</td>
<td>Any time</td>
<td>Post-fit</td>
</tr>
<tr>
<td><strong>Note:</strong> For post-fit power analysis, you get better results with the Intel Quartus Prime Power Analyzer.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software requirements</td>
<td>Spreadsheet program</td>
<td>The Intel Quartus Prime software</td>
</tr>
</tbody>
</table>

*continued...*
### Characteristic

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>EPE</th>
<th>Intel Quartus Prime Power Analyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>Medium</td>
<td>Medium to very high</td>
</tr>
</tbody>
</table>

#### 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
- 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
- Thermal power dissipation by block type
- Device supply currents

**Note:** The EPE and Power Analyzer outputs vary by device family.

#### Estimation of transceiver power for dynamic reconfiguration features
- Includes an estimation of the incremental power consumption by these features.
  - Not included

---

**Note:**
The Intel Quartus Prime Power Analyzer does not support analysis of the following Intel FPGA IP: Intel Stratix® 10 High Bandwidth Memory 2 (HBM2) IP, Intel Stratix 10 HPS IP, Intel Arria® 10 HPS IP. You can obtain a power estimation for these Intel FPGA IPs with the EPE spreadsheet.

### 1.2. Running the Power Analyzer

Before running the Power Analyzer you must run full compilation of your design to generate the post-fit netlist. In addition, you must either provide timing assignments for all clocks in the design, or specify signal activity data for power analysis. 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. To specify device power characteristics, operating voltage, and temperature conditions for power analysis, click **Assignments ➤ Settings ➤ Operating Settings and Conditions**, as Device Operating Condition Settings for Power Analysis on page 8 describes.
2. To run full compilation of your design, click **Processing ➤ Start Compilation**.
3. Click **Processing ➤ Power Analyzer Tool**.
4. Specify the source of signal activity data, as Generating Signal Activity Data for Power Analysis on page 10 describes.
5. To generate a Signal Activity (.saf) file during analysis, turn on **Write out signal activities used during power analysis**, and specify the file name.
6. 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. The Early Power Estimation file summarizes the resource utilization and allows you to perform what-if analyses in EPE.

7. Specify the **Default toggle rates for unspecified signals**, as Specifying the Default Toggle Rate on page 15 describes.

8. To specify temperature range and cooling options, click **Cooling Solution and Temperature**.

9. Click **Start**.

**Figure 3. Progress Bar in Intel Quartus Prime Power Analyzer**

10. When power analysis is complete, click **Report** to open the Power Analyzer reports that Viewing Power Analysis Reports on page 17 describes.

**Related Information**

Specifying Power Analyzer Input on page 7

**1.3. Specifying Power Analyzer Input**

The Power Analyzer accuracy is driven by design factors, operating conditions, and signal activity data that affect power consumption. The following figure shows how the Power Analyzer interprets these inputs and generates results in the Power Analysis report:

**Figure 4. Power Analyzer High-Level Flow**

---

(2) Operating condition specifications are available for only some device families
To obtain accurate I/O power estimates, the Power Analyzer requires full compilation of your design, in addition to specifying the following settings:

- 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.

1.3.1. Device Operating Condition Settings for Power Analysis

You can specify device power characteristics, operating voltage conditions, and operating temperature conditions for power analysis in the Operating Settings and Conditions page of the Settings dialog box.

Figure 5. Operating Settings and Conditions

The Power Analyzer reads the following settings to determine the device operating conditions for power analysis:

Table 2. Device Operating Condition Settings

<table>
<thead>
<tr>
<th>Option</th>
<th>Settings</th>
</tr>
</thead>
</table>
| Device power characteristics | • Typical—specifies average power consumed by typical silicon at nominal operating conditions.  
                                • Maximum—specifies maximum power consumed by worst-case device.       |
| Voltage tab                 | Specifies the operating voltage conditions for each power rail in the device, and the supply voltages for power rails with selectable supply voltages. |
| Temperature tab             | Specifies the thermal operating conditions of the device, including:      |
### 1.3.2. Specifying 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 activity data from the following sources:

- `.vcd` files generated by supported third-party simulators
- 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 6. Signal Activity Data Source Priority Scheme**

1.3.2.1. Using Simulation Signal Activity Data in Power Analysis

You can specify a Verilog Value Change Dump File (.vcd) generated by supported simulators as the source of signal activity data for power analysis.

Third-party simulators can output a `.vcd` that contains signal activity and static probability information that inform the power analysis. The generated `.vcd` includes all of the routing resources and the exact logic array resource usage.

---

(1) ModelSim*, ModelSim - Intel FPGA Edition, QuestaSim, Active-HDL, NCSim, VCS*, VCS MX, Riviera-PRO*
To improve accuracy of power analysis, you can generate a Standard Delay Output (.sdo) file that includes back-annotated delay estimates of the instances of core atoms for ModelSim simulation. ModelSim simulation can then output a more accurate .vcd for use as power analysis input. You must run the Fitter (Finalize) command before generating the .sdo. **Note:** To improve accuracy of power analysis, the Intel Quartus Prime EDA Netlist writer can generate a Standard Delay Output (.sdo) file that includes back-annotation of delays for a design's netlist for use during simulation in ModelSim. Although the .sdo only contains delay estimates and imprecise timing information, including the .sdo in simulation results in a more accurate output .vcd for power analysis.

**Note:** The EDA Netlist Writer currently supports .sdo file generation only for Verilog .vo simulation in the ModelSim simulator (not ModelSim - Intel FPGA Edition) for Intel Stratix 10 designs. The EDA Netlist Writer does not currently support .sdo file generation for any other simulator or device family.

### 1.3.2.1.1. Generating Signal Activity Data for Power Analysis

Follow these steps to generate and use simulation signal activity data for power analysis:

1. To run full compilation on your design, click **Processing ➤ Start Compilation**.
2. To specify settings for output of simulation files, click **Assignments ➤ Settings ➤ EDA Tool Settings ➤ Simulation**. Select your simulator in **Tool name** and the **Format for output netlist** and **Output directory**.
3. Turn on **Map illegal HDL characters**. This setting directs the EDA Netlist Writer to map illegal characters for VHDL or Verilog HDL, and results in more accurate data for power analysis.
4. For Intel Stratix 10 designs, to generate a Standard Delay Output (.sdo) file that includes back-annotation of delays for power analysis, refer to Generating Standard Delay Output for Power Analysis on page 11.

5. In the Intel Quartus Prime software, click Processing > Power Analyzer Tool. The Power Analyzer tab appears.

6. Under Input file, turn on Use input files to initialize toggle rates and static probabilities during power analysis, and then click Add Power Input Files. The Power Analyzer Settings page appears.

7. To specify a .vcd for power analysis, click Add and specify the File name, Entity, and Simulation period for the .vcd.

8. To enable glitch filtering during power analysis with the .vcd you generate, turn on Perform glitch filtering on VCD files.

9. To run the power analysis, click Start on the Power Analyzer tab. View the toggle rates in the power analysis results.

1.3.2.1.2. Generating Standard Delay Output for Power Analysis

To improve accuracy of power analysis, you can generate a Standard Delay Output (.sdo) file that includes back-annotated delay estimates for ModelSim simulation. ModelSim simulation can then output a more accurate .vcd for use as power analysis input. You must run Fitter (Finalize) before generating the .sdo.
Note: The EDA Netlist Writer currently supports .sdo file generation only for Verilog .vo simulation in the ModelSim simulator (not ModelSim - Intel FPGA Edition) for Intel Stratix 10 designs. The EDA Netlist Writer does not currently support .sdo file generation for any other simulator or device family.

**Figure 10. Using an SDO in Power Analysis**

1. Click **Assignments ➤ Settings ➤ EDA Tool Settings ➤ Simulation**. In **Tool name** select **ModelSim** and **Verilog** for **Format for output netlist**.
2. Click **More EDA Netlist Writer Settings**. Set **Enable SDO Generation for Power Estimation** to **On**. Set **Generate Power Estimate Scripts** to **ALL NODES**.

**Figure 11. More EDA Netlist Writer Settings**

3. To run the Fitter, click **Processing ➤ Start ➤ Start Fitter (Finalize)**.
4. Create a representative testbench (.vt) that exercises the design functions appropriately.
5. To specify the appropriate hierarchy level for signals in the output .vcd, add the following line to the project .qsf file:

```
set_global_assignment -name EDA_TEST_BENCH_DESIGN_INSTANCE_NAME
    <DUT instance path> -section_id eda_simulation
```

6. After Fitter processing is complete, click Processing ▶ Start ▶ Start EDA Netlist Writer. EDA Netlist Writer generates the following files in /<project>/simulation/modelsim/power/:
   - `<project>.vo` (contains a reference to the .sdo file by default)
   - `<project>_dump_all_vcd_nodes.tcl`—specifies nodes to save in .vcd
   - `<project>_v.sdo`—back-annotated delay estimates

7. Create a ModelSim script (.do) to load the design and testbench, start ModelSim, and then source the .do script.

8. To specify the signals ModelSim includes in the .vcd file, source `_dump_all_vcd_nodes.tcl` in ModelSim.

9. To generate the .vcd file, simulate the test bench and netlist in ModelSim. The .vcd file generates according to your specifications.

10. Specify the .vcd as an input to power analysis, as Generating Signal Activity Data for Power Analysis on page 10 describes.

### 1.3.2.1.3. Simulation Glitch Filtering

You can enable glitch filtering in the .vcd that you generate in a third-party simulator for use in power analysis by turning on the **Perform glitch filtering on VCD files** option.

---

(2) Specify the full hierarchical path in the testbench, not just the instance name. For example, specify a|b|c, not just c.
Figure 12. Enabling Glitch Filtering for VCD

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.
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.

The .vcd file reader performs glitch filtering that is complementary to simulation glitch filtering, but is often less precise. While the .vcd file reader has the ability to remove glitches on logic blocks, the file reader cannot determine how a given glitch potentially affects downstream logic and routing. Filtering the glitches during simulation avoids switching downstream routing and logic automatically.

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.

1.3.2.2. 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.

1.3.2.2.1. 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.

1.3.2.3. 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.

1.3.2.4. Signal Activities from User Defaults Only

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

1.3.3. Specifying the Default Toggle Rate

You can specify the Default toggle rates for unspecified signals in your design for power analysis. The Power Analyzer uses the default toggle rate when no other method specifies the signal activity data.
You specify 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 $f_{\text{MAX}}$ 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 the clock domain is ambiguous. For example, the Power Analyzer cannot determine a clock domain for a node unless you specify sufficient timing constraints for the clock domains. If the Power Analyzer cannot determine the clock domain for a node, the Power Analyzer substitutes and reports a toggle rate of zero.

**Related Information**

Toggle Rate on page 30

### 1.3.4. Specifying Toggle Rates for Specific Nodes

You can assign toggle rates and static probabilities to individual nodes 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 is appropriate for signals in which you have knowledge of the signal 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.

1.3.4.1. Clock Node Toggle Rates

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.

1.3.5. Avoiding Simulation Node Name Match

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.

1.4. Viewing Power Analysis Reports

Following successful power analysis, click the Power Analyzer Reports button to view the Power Analysis section of the Compilation Report.
The Power Analysis reports 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.

**Power Savings Summary**

Lists any savings (in mW) and the type of savings method, such as SmartVID Power Savings.

**Parallel Compilation**

When you enable parallel compilation, the Parallel Compilation report list the number of processors you use during Power Analysis.

**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.

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, 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.

1.5. Power Analysis in Modular Design Flows

In modular or hierarchical design flows you develop each design block separately, and then instantiate these blocks into a higher-level design to form a complete design. The Intel Quartus Prime software supports simulation and power analysis of the top-level design or individual blocks with the design.

Figure 15. Modular Simulation Flow

You can associate multiple .vcd simulation output files with specific node names, enabling the integration of partial design simulations into a complete design power analysis. When specifying multiple .vcd files for a node, 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 node, the Power Analyzer calculates the signal activity as the equal-weight arithmetic average of each .vcd.
- When you apply multiple simulation files to design nodes 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 node.

The following figure shows an example of a hierarchical design:

Figure 16. 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 Power Analysis Flow

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 Power Analysis Flow

You can independently simulate 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>
<thead>
<tr>
<th>File Name</th>
<th>Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>8b10b_dec.vcd</td>
<td>Top</td>
</tr>
<tr>
<td>8b10b_dec.vcd</td>
<td>Top</td>
</tr>
<tr>
<td>8b10b_dec.vcd</td>
<td>Top</td>
</tr>
<tr>
<td>8b10b_rxerr.vcd</td>
<td>Top</td>
</tr>
<tr>
<td>8b10b_enc.vcd</td>
<td>Top</td>
</tr>
<tr>
<td>mux.vcd</td>
<td>Top</td>
</tr>
</tbody>
</table>

The resulting power analysis applies the simulation vectors in each file to the assigned instance. Simulation provides signal activities for the pins and for the outputs of functional blocks. If the inputs to an 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 instance such as mux1 has its signal activity specified at the output of one of the decode instances.
1.5.3. Multiple Simulation Power Analysis Flow

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 4. Multiple Simulation File Names and Entities

<table>
<thead>
<tr>
<th>File Name</th>
<th>Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal.vcd</td>
<td>Top</td>
</tr>
<tr>
<td>corner1.vcd</td>
<td>Top</td>
</tr>
<tr>
<td>corner2.vcd</td>
<td>Top</td>
</tr>
</tbody>
</table>

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 Simulation Power Analysis Flow

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 5. Overlapping Simulation Import Specifications

<table>
<thead>
<tr>
<th>File Name</th>
<th>Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>full_design.vcd</td>
<td>Top</td>
</tr>
<tr>
<td>error_cases.vcd</td>
<td>Top</td>
</tr>
</tbody>
</table>

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 Design Simulation Power Analysis Flow

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).
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 for Signal Activity Calculations

To specify a start and end time for 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:

```tcl
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
```

### 1.5.6. Vectorless Estimation Power Analysis Flow

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. 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**

1.6.1. 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 not to use an input file, specify a default input I/O toggle rate of 60%, with vectorless estimation off, 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**

[Viewing Power Analysis Reports on page 17](#)
# 1.7. Power Analysis Revision History

The following revision history applies to this chapter:

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019.12.04</td>
<td>19.1.0</td>
<td>• Removed references to entity-specific toggle rates in &quot;Specifying Toggle Rates for Specific Nodes.&quot; Toggle rates must be either global or node specific.</td>
</tr>
</tbody>
</table>
| 2019.08.02       | 19.1.0                     | • Clarified wording of statements about .vcd files in "Simulation Glitch Filtering" topic.  
                        • Corrected typo in "Specifying the Default Toggle Rate" topic.  
                        • Corrected typo in "Running the Power Analyzer from the Command Line" topic.  
                        • Improved explanation in "Generating Standard Delay Output for Power Analysis" topic. |
| 2019.07.03       | 19.1.0                     | • Corrected broken links to Help. |
| 2019.04.01       | 19.1.0                     | • Described new support for generation of SDO for use in power analysis.  
                        • Retitled some topic headings for greater clarity.  
                        • Changed the order of some topics for improved flow of information.  
                        • Added descriptions of Power Savings Summary and Parallel Compilation power analysis reports.  
                        • Added new Power Analysis flow diagrams. |
| 2018.09.24       | 18.1.0                     | • General chapter reorganization.  
                        • Moved Factors Affecting Power Consumption to chapter: Power Optimization.  
                        • 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  
                        • Removed content referring to device families not supported in Intel Quartus Prime Pro Edition. |
| 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. |
| 2016.10.31       | 16.1.0                     | • Implemented Intel rebranding.  
                        • Removed support for .vcd generation by the Compiler. Generate .vcd files for power estimation in your EDA simulator. |
| 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. |
<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014.08.18</td>
<td>14.0a10.0</td>
<td>Updated &quot;Current Drawn from Voltage Supplies&quot; 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.</td>
</tr>
</tbody>
</table>
| 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. |
| 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 "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 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
- 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 28
Device Selection on page 28
Environmental Conditions on page 29
Device Resource Usage on page 29
Signal Activity on page 30

2.1.1. Design Activity and Power Analysis

Power consumption of a device also depends on the design's activity over time. Static power ($P_{\text{STATIC}}$) is the thermal power that a chip dissipates independent of user clocks. $P_{\text{STATIC}}$ 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 6. Environmental Conditions that Affect Power Consumption

<table>
<thead>
<tr>
<th>Environmental Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airflow</td>
<td>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 &quot;still air&quot; 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.</td>
</tr>
<tr>
<td>Heat Sink and Thermal Compound</td>
<td>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 ( \theta_{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 ( \theta_{CA} ).</td>
</tr>
<tr>
<td>Junction Temperature</td>
<td>The junction temperature of a device is equal to: [ T_{Junction} = T_{Ambient} + P_{Thermal} \cdot \theta_{JA} ] in which ( \theta_{JA} ) is the total thermal resistance from the device transistors to the environment, in degrees Celsius per watt. The value ( \theta_{JA} ) is equal to the sum of the junction-to-case (package) thermal resistance ( \theta_{JC} ), and the case-to-ambient thermal resistance ( \theta_{CA} ) of the cooling solution.</td>
</tr>
<tr>
<td>Board Thermal Model</td>
<td>The junction-to-board thermal resistance ( \theta_{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 ( \theta_{JA} ) and ambient temperatures.</td>
</tr>
</tbody>
</table>

### 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. 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
Specifying the Default Toggle Rate on page 15
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. 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 Pro Edition User Guide: Design Optimization
- Launch Design Space Explorer Command (Tools Menu)
  In Intel Quartus Prime Help

2.3. Power-Driven Compilation


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.

2.3.1. 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 7. Power Optimization During Synthesis Options

<table>
<thead>
<tr>
<th>Settings</th>
<th>Description</th>
<th>Optimization Techniques Included</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>The Compiler does not perform netlist, placement, or routing optimizations to minimize power.</td>
<td>-</td>
</tr>
</tbody>
</table>
| Normal compilation     | 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 8. Low Power Mode Parameter Options

<table>
<thead>
<tr>
<th>Parameter Value</th>
<th>Equivalent Setting in Power Optimization During Synthesis Logic Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Off</td>
</tr>
<tr>
<td>Auto</td>
<td>Normal compilation</td>
</tr>
<tr>
<td>All</td>
<td>Extra effort</td>
</tr>
</tbody>
</table>

### Related Information
- Clock Enable in Memory Blocks on page 37
- Intel Quartus Prime Compiler Settings on page 35

#### 2.3.1.1. 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.
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.

### 2.3.1.2. Power-Aware Logic Mapping

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

### 2.3.1.3. 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**.

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**

- **RAM and ROM Parameter Settings**
  - In *Intel Stratix 10 Embedded Memory User Guide*
2.3.2. 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 9. Power-Driven Fitter Option

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>The Fitter does not perform optimizations to minimize power.</td>
</tr>
<tr>
<td>Normal compilation</td>
<td>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.</td>
</tr>
<tr>
<td>Extra effort</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

Related Information
Assignment Editor Options on page 36

2.3.3. 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.
2.3.4. 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.

![Gate-Level Register Retiming](image)

**Figure 18. 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.

2.3.5. 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.**
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.

### 2.3.6. 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.

**Power Optimization During Synthesis** logic option determines how aggressively Analysis & Synthesis optimizes the design for power.
Table 10. Power Optimization During Synthesis Options

<table>
<thead>
<tr>
<th>Settings</th>
<th>Description</th>
<th>Optimization Techniques Included</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>The Compiler does not perform netlist, placement, or routing optimizations to minimize power.</td>
<td>-</td>
</tr>
<tr>
<td>Normal compilation</td>
<td>The Compiler applies low compute effort algorithms to minimize power through netlist optimizations that do not reduce design performance.</td>
<td>• Memory block optimization&lt;br&gt;• Power-aware logic mapping</td>
</tr>
<tr>
<td>Extra effort</td>
<td>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.</td>
<td>• Memory block optimization&lt;br&gt;• Power-aware logic mapping&lt;br&gt;• Power-aware memory balance</td>
</tr>
</tbody>
</table>

Related Information
- Area-Driven Synthesis on page 34
- Power-Driven Fitter on page 34

2.4. Design Guidelines

During FPGA design implementation, you can apply the following design techniques to reduce power consumption. The results of these techniques are different from design to design.

2.4.1. 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.

2.4.1.1. 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 22. 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.

Enable

Internal Memory Clk

Ck

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 23. Clock Enable in RAM 2-Port**

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.
2. Power Optimization

Figure 24. RAM 2-Port Maximum Depth

Related Information
- Power-Driven Compilation on page 31
- Clock Power Management on page 37
- Clocking Modes and Clock Enable

2.4.1.2. 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 25. 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.

2.4.1.2.1. 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.

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

2.4.1.3. Clock Enables

Use clock enables instead of gated clocks:

```vhdl
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
```

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

```vhdl
always @(posedge clk)
begin
  if (ena)
    temp <= dataa;
  else
    temp <= temp;
end
```
2.4.1.4. 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.

2.4.1.4.1. Viewing Clock Details in the Chip Planner

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

Figure 28. Chip Planner Task Pane

Figure 29. Report Clock Details

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 30. Clock Highlight in Chip Planner
This example uses a Regional clock Region instead of a global signal.

2.4.1.5. Merge Clocks

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

<table>
<thead>
<tr>
<th>Design</th>
<th>2clks &amp; 2PLLs</th>
<th>1 Clk &amp; 1 PLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oc_dma_stamp25</td>
<td>6.079W</td>
<td>5.46W</td>
</tr>
</tbody>
</table>

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

2.4.2. 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.
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.

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.

### 2.4.3. 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.

Related Information
Timing Closure and Optimization
In Intel Quartus Prime Pro Edition User Guide: Design Optimization

2.4.4. 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.

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:

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

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.
2.4.5. Dynamically Controlled On-Chip Terminations (OCT)

Dynamic OCT enables series termination (RS) and parallel termination (RT) to dynamically turn on/off during the data transfer. This feature is especially useful in FPGAs with external memory interfaces, such as interfacing with DDR memories.

Dynamic OCT eliminates the constant DC power that parallel termination consumes when transmitting data, reducing power consumption when compared to conventional termination. Parallel termination is extremely useful for applications that interface with external memories where I/O standards, such as HSTL and SSTL, are used. Parallel termination supports dynamic OCT, which is useful for bidirectional interfaces.

For more information about dynamic OCT in specific devices, refer to the Intel Stratix 10 General Purpose I/O User Guide or the Intel Arria 10 Core Fabric and General Purpose I/O Handbook.

Example 1. Example: Power Saving for a DDR3 Interface with OCT

The static current consumed by parallel OCT is equal to the $V_{CCIO}$ voltage divided by 100 W. For DDR3 interfaces with SSTL-15, the static current per pin is:

$$\frac{1.5V}{100W} = 15mA$$

Therefore, the static power is:

$$1.5V \times 15mA = 22.5mW$$

For an interface with 72 DQ and 18 DQS pins, the static power is:

$$90 \text{pins} \times 2.25mW = 2.025W$$

Dynamic parallel OCT disables parallel termination during write operations, so if writing occurs 50% of the time, the power saved by dynamic parallel OCT is:

$$50\% \times 2.025W = 1.0125W$$

For more information about dynamic OCT in Stratix IV devices, refer to the chapter in the Stratix IV Device Handbook.

Related Information

- Dynamic OCT
  In Intel Arria 10 Core Fabric and General Purpose I/O Handbook
- Dynamic OCT
  In Intel Stratix 10 General Purpose I/O User Guide

2.4.6. 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.
Some guidelines to optimize the use of memories are:

- Port shallow memories from M20K to MLAB.

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

```vhdl
(* 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.

### 2.4.6.1. Implementation

#### Table 11. Single-port Embedded Memory Configurations for Intel Arria 10 Devices

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

<table>
<thead>
<tr>
<th>Memory Block</th>
<th>Depth (bits)</th>
<th>Programmable Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLAB</td>
<td>32</td>
<td>x16, x18, or x20</td>
</tr>
<tr>
<td></td>
<td>64 (3)</td>
<td>x8, x9, x10</td>
</tr>
<tr>
<td>M20K</td>
<td>512</td>
<td>x40, x32</td>
</tr>
<tr>
<td></td>
<td>1K</td>
<td>x20, x16</td>
</tr>
<tr>
<td></td>
<td>2K</td>
<td>x10, x8</td>
</tr>
<tr>
<td></td>
<td>4K</td>
<td>x5, x4</td>
</tr>
<tr>
<td></td>
<td>8K</td>
<td>x2</td>
</tr>
<tr>
<td></td>
<td>16K</td>
<td>x1</td>
</tr>
</tbody>
</table>

#### Figure 34. Power numbers from EPE

(3) **Supported through software emulation and consumes additional MLAB blocks.**
2.4.6.2. 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.

<table>
<thead>
<tr>
<th>Module</th>
<th>RAM Type</th>
<th># RAM Banks</th>
<th>Data Width</th>
<th>RAM Depth</th>
<th>Port A</th>
<th>Port B</th>
<th>Thermal Power (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16B</td>
<td>M20K</td>
<td>144, 40</td>
<td>512</td>
<td></td>
<td>384.0</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>16B</td>
<td>M20K</td>
<td>144, 40</td>
<td>512</td>
<td></td>
<td>384.0</td>
<td>100%</td>
<td>50%</td>
</tr>
</tbody>
</table>

2.4.7. DDR Memory Controller Settings

The External Memory Interfaces Intel Arria 10 FPGA IP provides low power mode settings. These settings put DDR memory in power saving mode when the controller is idle, providing power savings on external memory DDR. The **Enable Auto Power-Down** and **Auto Power-Down Cycles** settings enable this capability.

**Low Power Mode Settings**

- **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**—specifies the number of cycles the controller must be IDLE before entering the 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.

**Example 2. Implement Multiplier + Accumulator in 1 DSP**

```verilog
always @ (posedge clk)
begin
  if (ena)
  begin
```

**Related Information**

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

2.4.8. 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 3. Implement multiplication in 2 DSPs and the adder in LABs

```verilog
dataout <= dataa * datab + datac * datad;
end
end
```

```verilog
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 Pro Edition User Guide: Design Recommendations

2.4.9. Reducing High-Speed Tile (HST) Usage

High-Speed tiles are available in the Intel Arria 10 design family.

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.

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.
2.4.10. 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.

2.4.11. Periphery Power reduction XCVR Settings

2.4.11.1. 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 how lossy is the channel.
- Turn off PDN compensation.
  This setting induces jitter, which is necessary to check system tolerance.
- Use one equalizer stage.
<table>
<thead>
<tr>
<th>DFE</th>
<th>Adaptation</th>
<th>Equalizer Stage</th>
<th>Transmitter High-Speed Compensation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabled</td>
<td>Disabled</td>
<td>Non-S1 Mode</td>
<td>Disabled</td>
</tr>
<tr>
<td>Disabled</td>
<td>Disabled</td>
<td>Non-S1 Mode</td>
<td>Enabled</td>
</tr>
<tr>
<td>Disabled</td>
<td>Disabled</td>
<td>N/A</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

2.4.11.2. 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.

2.5. 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 38. Power Optimization Advisor

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.

2.5.1. 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.
2.5.1.1. Find Timing Information

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

**Figure 39. 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.

2.5.2. 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 28

2.5.3. Dynamic Power

The recommendations in this section can reduce dynamic power.
2.5.4. 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 28

2.5.5. Appropriate I/O Standards

Choose appropriate I/O Standards to minimize design power.

Related Information

I/O Power Guidelines on page 44

2.5.6. Use RAM Blocks

Implement RAMs and medium to large shift registers in RAM blocks instead of logic cell registers.
2.5.7. 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 37
- Memory Optimization (M20K/MLAB) on page 45

2.5.8. 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.

2.5.9. 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 4. Glitch Prone Design

Related Information
- Pipelining and Retiming on page 42
### 2.6. Power Optimization Revision History

The following revision history applies to this chapter:

#### Table 12. Document Revision History

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 2019.08.02       | 18.1.0                     | • Corrected typo in "Viewing Clock Details in the Chip Planner" topic.  
|                  |                            | • Corrected typo in "Pipelining and Retiming" topic.  
|                  |                            | • Corrected typo in "Implementation" topic.  
|                  |                            | • Updated "DDR Memory Controller Settings" topic for latest IP name and to correct typos.  
|                  |                            | • Corrected typo in "Pipeline Logic to Reduce Glitching" topic.  |
| 2018.09.24       | 18.1.0                     | • 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: Memories (M20K/MLAB), DDR Memory Controller Settings, DSP Implementation, Reducing High-Speed Tile (HST) Usage, Unused Transceiver Channels, Periphery Power reduction XCVR Settings  
|                  |                            | • Removed content referring to device families not supported in Intel Quartus Prime Pro Edition.  |
| 2018.06.11       | 18.0.0                     | • Moved general information about the Design Space Explorer (DSE II) to the Design Optimization User Guide, 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 User Guide, left a section about using DSE II for Power-Driven Optimization.  |
| 2016.10.31       | 16.1.0                     | • Implemented Intel rebranding.  
|                  |                            | • 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.  |
| 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  |

*continued...*
<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 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.
3. Power Analysis and Optimization Document Archive

If the table does not list a software version, the user guide for the previous software version applies.

<table>
<thead>
<tr>
<th>Intel Quartus Prime Version</th>
<th>User Guide</th>
</tr>
</thead>
</table>
A. Intel Quartus Prime Pro Edition User Guides

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

Related Information

- **Intel Quartus Prime Pro Edition User Guide: Getting Started**
  Introduces the basic features, files, and design flow of the Intel Quartus Prime Pro Edition software, including managing Intel Quartus Prime Pro Edition projects and IP, initial design planning considerations, and project migration from previous software versions.

  Describes creating and optimizing systems using Platform Designer, a system integration tool that simplifies integrating customized IP cores in your project. Platform Designer automatically generates interconnect logic to connect intellectual property (IP) functions and subsystems.

  Describes best design practices for designing FPGAs with the Intel Quartus Prime Pro Edition software. HDL coding styles and synchronous design practices can significantly impact design performance. Following recommended HDL coding styles ensures that Intel Quartus Prime Pro Edition synthesis optimally implements your design in hardware.

- **Intel Quartus Prime Pro Edition User Guide: Design Compilation**
  Describes set up, running, and optimization for all stages of the Intel Quartus Prime Pro Edition Compiler. The Compiler synthesizes, places, and routes your design before generating a device programming file.

  Describes Intel Quartus Prime Pro 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, optimizing device resource usage, device floorplanning, and implementing engineering change orders (ECOs).

  Describes operation of the Intel Quartus Prime Pro 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 Pro Edition User Guide: Block-Based Design**
  Describes block-based design flows, also known as modular or hierarchical design flows. These advanced flows enable preservation of design blocks (or logic that comprises a hierarchical design instance) within a project, and reuse of design blocks in other projects.
• Intel Quartus Prime Pro 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 Pro 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 Pro 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 Pro Edition User Guide: Third-party Logic Equivalence
  Checking Tools
  Describes support for optional logic equivalence checking (LEC) of your design
  in third-party LEC tools by OneSpin*.

• Intel Quartus Prime Pro Edition User Guide: Debug Tools
  Describes a portfolio of Intel Quartus Prime Pro 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.

  Explains basic static timing analysis principals and use of the Intel Quartus
  Prime Pro 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.

  Describes the Intel Quartus Prime Pro 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 Pro 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 Interface Planner to prototype
  interface implementations, plan clocks, and quickly define a legal device
  floorplan. Use the Pin Planner to visualize, modify, and validate all I/O
  assignments in a graphical representation of the target device.

  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 Pro Edition User Guide: Scripting
  Describes use of Tcl and command line scripts to control the Intel Quartus
  Prime Pro Edition software and to perform a wide range of functions, such as
  managing projects, specifying constraints, running compilation or timing
  analysis, or generating reports.