Techniques for achieving the highest design performance are important when designing for larger density FPGAs. The tools that facilitate these techniques must provide the highest level of flexibility without compromising ease-of-use. The place-and-route features of the Quartus® II software allow designers to meet performance requirements by facilitating optimization at multiple points in the design process.

This application note explores the various compiler settings and explains techniques that can optimize resource allocation to maximize the performance of Altera® devices. Before using this document, be sure you are familiar with the topics discussed in AN 198: Timing Closure Using the Quartus II Software and AN 161: Using the LogicLock Methodology in the Quartus II Design Software.

Table Of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Compilation</td>
<td>1</td>
</tr>
<tr>
<td>Design Analysis</td>
<td>6</td>
</tr>
<tr>
<td>Resource Utilization Optimization Techniques</td>
<td>14</td>
</tr>
<tr>
<td>I/O Optimization Techniques</td>
<td>20</td>
</tr>
<tr>
<td>$f_{\text{MAX}}$ Optimization Techniques</td>
<td>25</td>
</tr>
<tr>
<td>Conclusion</td>
<td>36</td>
</tr>
</tbody>
</table>

Initial Compilation

To best meet performance requirements, perform an initial compilation for fit verification, then optimize for resource utilization, I/O timing, and $f_{\text{MAX}}$ timing, in that order (Figure 1).
Figure 1. Design Optimization Sequence

![Diagram showing the design optimization sequence]

Make appropriate initial compilation assignments before compiling a design in the Quartus II software. Significantly different compilation results can occur depending on what assignments are made. This section describes the basic assignments to make before an initial compilation.

Additional settings may be required for best results depending on the design. See the Altera documentation listed on page 36 for guidelines and recommendations on coding designs for optimal performance.

**Compiler Settings**

Compiler settings determine what timing information is used, what fitter algorithm is used, and whether or not to make adjustments in one area or another when a design is compiled.

**Device Settings**

Assigning a specific device determines the timing model used during compilation. It is important to chose the correct speed grade in order to obtain accurate results and the best optimization. The device size and the package determine how many resources are used. Figure 2 shows the Device page of the Settings dialog box (Assignments menu).
Timing Driven Compilation

Set the **Optimize timing** and the **Optimize I/O cell register placement for timing** options on the **Fitting** page of the **Settings** dialog box (Assignments menu) as shown in **Figure 3**.

The **Optimize timing** option directs the Quartus II software to optimize the fitting of the design to meet the specified performance requirements for all maximum delay timing constraints in a design, including $t_{SU}$, $t_{CO}$, and $f_{MAX}$.

The **Optimize hold timing** option directs the Quartus II software to optimize minimum delay timing constraints, such as hold times ($t_H$) on device input pins, minimum delays from input to output pin ($t_{PD}$) and minimum $t_{CO}$ to an output pin. The Quartus II software adds delay to connections as needed to guarantee the minimum delays required by these constraints are satisfied. This option is available only for the Stratix™, Stratix GX, and Cyclone™ device families.
The **Optimize I/O cell register placement for timing** option allows the Quartus II software to determine whether or not registers are placed in I/O elements in order to meet a strict tIO or tSU timing requirement. For more information on this option see “I/O Optimization Techniques” on page 20.

**Figure 3. Fitting Settings**

![Figure 3. Fitting Settings](image)

**Fast Fit**

Fast Fit reduces the amount of optimization effort for each algorithm employed during fitting. This reduces the compilation time by about 50%, while resulting in a fit that has, on average, 10% lower \( f_{MAX} \) than the \( f_{MAX} \) achieved by a regular fit. For a small minority of hard to fit circuits, the reduced optimization of Fast Fit can result in the first fitting attempt being unroutable, resulting in multiple fitting attempts and a long fitting time. Select this option on the **Fitting** page of the **Settings** dialog box as shown in Figure 3. The default in the Quartus II software is best fit through the default fit option.
I/O Assignments

The I/O standards and drive strengths employed in a design each affect I/O timing. Assign I/O standards using the Assign Pins dialog box by clicking Assign Pins on the Device page of the Settings dialog box (Assignments menu). See Figure 4. You can also make I/O assignments using the Assignment Editor (Assignment menu).

For more information on I/O standards, see the following documents:

- AN 253: Using Selectable I/O Standards in Cyclone Devices
- AN 166: Using High-Speed I/O Standards in APEX II Devices
- AN 134: Programmable I/O Standards in Mercury Devices
- AN 117: Using Selectable I/O Standards in APEX 20KE, APEX 20KC & MAX 7000B Devices
- the Using Selectable I/O Standards in the Stratix & Stratix GX Devices chapter of the Stratix Device Handbook

Figure 4. Assigning I/O Standards

Timing Settings

An important step in obtaining the highest performance is to apply detailed timing constraints. The Quartus II PowerFit™ fitter attempts to exceed specified timing requirements. Netlist optimizations are also performed based on the timing constraints specified. See “Netlist Optimization Options” on page 25 for more information.
Timing settings are used during the timing analysis. The compilation report states whether or not timing requirements were met and provides detailed timing information on which paths violate the timing requirements.

Every clock should have an accurate clock setting. All I/Os for which \( t_{SU} \) or \( t_{CO} \) is to be optimized should also have settings. It is important to make any complex timing assignments according to the needs of the design, including multicycle and cut-timing path assignments. This information allows the Quartus II software to make appropriate trade-offs between paths.

When there are timing constraints in the design, the Quartus II software will not attempt to optimize clocks that are unconstrained. Specify timing constraints in your design wherever possible for best results.

For more information on how to make timing assignments, refer to the following:

- AN 123: Using Timing Analysis in the Quartus II Software
- Quartus II Online Help

After appropriate initial compilation assignments have been made and the design is compiled, proceed to analyze the design to determine if resource utilization, I/O timing, or \( f_{MAX} \) timing need to be optimized.

### Design Analysis

The initial compilation establishes whether the design achieved a successful fit and met the specified performance. The Quartus II software provides a **Compilation Report** to analyze the design implementation.

- If the design does not fit, apply one of the techniques described under “Resource Utilization” on page 6 before trying to optimize I/O timing or \( f_{MAX} \) timing.
- If the I/O timing performance requirements are not met, see “I/O Optimization Techniques” on page 20 before trying to optimize \( f_{MAX} \) timing.
- If \( f_{MAX} \) performance requirements are not met, see “\( f_{MAX} \) Improvement Summary” on page 35.

### Resource Utilization

Determining device utilization is important regardless of whether a successful fit was achieved. In the event of a successful fit, review this information to determine whether the future addition of extra logic or any other design changes may introduce fitting difficulties. This
information also gives an indication of how easy it will be to perform manual floorplanning in the device, should it be necessary to improve design performance in the future.

For suggestions on how to reduce resource utilization, see “Resource Utilization Optimization Techniques” on page 14.

To determine resource usage, refer to the Summary section of the Compilation Report as shown in Figure 5. This section reports how many pins are used, as well as other device resources such as memory bits, digital signal processing (DSP) block 9-bit elements, and PLLs. This section provides a good indication of where the design exceeded the available device resources. More detailed information is available by viewing the Resource Section in the Fitter section of the Compilation Report (Processing menu).

Figure 5. Compilation Report Summary

If resource usage is reported under 100% and a successful fit was still not achieved, then it is likely that there were not enough routing resources or that some assignments were illegal.

If the fitter ends very quickly, then a resource may be over-utilized or there is an illegal assignment. If the Quartus II software runs for a long time, then it is likely that a legal placement or route cannot be found. Look for a compilation message giving an indication of the problem.

Use the Timing Closure floorplan to view areas of routing congestion. The colors represent the amount of routing congestion. The red area in Figure 6 denotes an area of the device with very congested routing.
I/O Timing

To determine whether or not I/O timing has been met, see the Timing Analyzer folder in the Compilation Report (Processing menu). The $t_{ss}$, $t_{th}$, and $t_{co}$ reports list the I/O paths and the slack associated with each. The I/O paths that have not met the required timing are reported with a negative slack and are displayed in red as shown in Figure 7.

Figure 7. I/O Requirements
To see why timing requirements are not met, right-click a particular I/O entry and choose List Paths. A message appears in the System tab of the Message window. You can expand a selection by clicking the ‘+’ icon at the beginning of the line, as shown in Figure 8. This is a good method of determining where along the path the greatest delay is located.

Figure 8. I/O Slack Report

To visually analyze I/O timing, right-click on an I/O entry in the report and select Locate in Timing Closure Floorplan as shown in Figures 9 and 10. The Timing Closure Floorplan is displayed, highlighting the I/O path. Note that you can set the level of detail in the floorplan with the selection under the View menu. This can also be done by viewing the path in the Last Compilation Floorplan.

Figure 9. Locate Failing Path in Timing Closure Floorplan
In Figure 11 the red arrows begin at the start point of a critical path (i.e., a register) to the end point of that critical path (i.e., another register) and the times shown are the slack figures for each path. Negative slack indicates paths that failed to meet their timing requirements.

To see all the intermediate nodes (i.e., combinatorial logic cells) on a path and the delay for each level of logic, right-click the title bar above a path’s slack number and choose Expand (right button pop-up menu). To view all these paths in the Timing Closure Floorplan choose Routing > Show Critical Paths (View menu).
**f<sub>MAX</sub> Timing**

To determine whether or not f<sub>MAX</sub> timing requirements are met, refer to the Timing Analyzer folder in the Compilation Report (Processing menu). The Clock Setup folder of the Compilation Report provides figures for slack and register-to-register f<sub>MAX</sub>. The paths that are not meeting timing requirements are shown in red. See Figure 12.

For suggestions on how to improve f<sub>MAX</sub> timing, see “f<sub>MAX</sub> Optimization Techniques” on page 25.
To analyze why timing was not met, right click on a particular path reported on in the System tab of the Message window (Figure 13) and select List Paths (right button pop-up menu) to determine the location of the greatest delay. You can expand a selection by clicking the "+" icon at the beginning of the line.

Figure 13. \( f_{\text{max}} \) Slack Report

Visually analyze \( f_{\text{max}} \) paths by right-clicking on a path in the report and selecting Locate in Timing Closure Floorplan to display the Timing Closure Floorplan, which highlights the path. See Figure 14. Figure 15 shows the Timing Closure Floorplan displaying a failing path.

Figure 14. Locate Failing Path in Timing Closure Floorplan
Figure 15. Failing Path in Timing Closure Floorplan

View all failing paths in the **Timing Closure Floorplan** using the **Show Critical Paths** feature. Figure 16 shows critical $f_{\text{max}}$ paths in the **Timing Closure Floorplan**.
After design analysis, the second design optimization stage is to improve resource utilization. Complete this stage before proceeding to the I/O timing optimization stage. If a design is not fitting into a specified device, employ the techniques in this section to achieve a successful fit.

**Use Register Packing**

The *Auto Packed Registers* option is available regardless of the tool used to synthesize the design. Register packing combines logic cells where only the register is used with logic cells where only the lookup table (LUT) is used into a single logic cell. *Figure 17* shows the packing and the gain of one logic cell.
Figure 17. Register Packing

Registers may also be packed into DSP blocks as shown in Figure 18.

Figure 18. Register Packing in DSP blocks

The six most common cases in which register packing can help to optimize a design are:

- An LUT can be implemented in the same cell as an unrelated register with a single data input
- An LUT that feeds a register with a single data input and also feeds other logic can be implemented in the same cell as the register that it feeds
- A register and the LUT that it feeds can be implemented in the same cell
- A register can be packed into a RAM block
- A register can be packed into a DSP block
- A register can be packed into an I/O Element (IOE)

There are four options available for register packing, as follows.
Optimizing FPGA Performance Using the Quartus II Software

Resource Utilization Optimization

- **Off** – Does not pack registers.
- **Normal** – Default setting packs registers when this is not expected to hurt timing.
- **Minimize Area** – Aggressively packs registers to reduce area.
- **Minimize Area with Chains** – Aggressively packs registers to reduce area. This option will pack registers with carry chains, and convert registers into register cascade chains and pack them with other logic to reduce area. This option is only available for Stratix, Stratix GX and Cyclone devices.

Turning register packing on decreases the number of LEs in the design but could decrease performance. To turn on register packing, turn on the **Packed Registers** option on the **Default Logic Option Settings** page of the **Settings** dialog box (Assignments menu) as shown in **Figure 19**.

**Figure 19. Auto Packed Registers Option**

![Figure 19. Auto Packed Registers Option](image-url)
The area reduction and performance results can vary greatly depending on the design. Register packing produces the typical results for Stratix, Stratix GX, and Cyclone devices shown in Table 1. Results will vary depending on the design.

### Table 1. Typical Register Packing Results for Stratix, Stratix GX, & Cyclone Devices

<table>
<thead>
<tr>
<th>Register Packing Setting</th>
<th>Relative f&lt;sub&gt;MAX&lt;/sub&gt;</th>
<th>Relative LE Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>1.0</td>
<td>1.12</td>
</tr>
<tr>
<td>Normal (default)</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Minimize Area</td>
<td>0.97</td>
<td>0.93</td>
</tr>
<tr>
<td>Minimize Area with Chains</td>
<td>0.94</td>
<td>0.90</td>
</tr>
</tbody>
</table>

*Remove Fitting Constraints*

A design with too many user constraints may not fit the targeted device. This occurs when the location or LogicLock™ assignments are too strict and there are not enough routing resources. In this case, use the Routing Congestion viewer to locate routing problems in the floorplan and remove any location and/or LogicLock assignments in that area. If the design still does not fit, the design is over constrained. To correct the over constraint, remove all location and LogicLock assignments and run successive compilations, incrementally constraining the design before each compilation.

For more information on the Routing Congestion viewer, see Quartus II Help.

*Modify Quartus II Synthesis Options*

If the design fails to fit because it uses too many LEs, memory, or DSP blocks and using a larger device is not an option, there are several ways to reduce the resource utilization. For information on specific Quartus II synthesis options, refer to *AN 238: Using Quartus II Verilog HDL & VHDL Integrated Synthesis*.

*Modify EDA Synthesis Tool Options*

Most synthesis tools have an option to optimize for area or speed. Also, for some synthesis tools, not specifying an f<sub>MAX</sub> may result in less logic utilization.
For more information on synthesis tools, see the following documents:

- AN 238: Using Quartus II Verilog HDL & VHDL Integrated Synthesis
- AN 226: Synplify & Quartus II Design Methodology
- AN 225: LeonardoSpectrum & Quartus II Design Methodology
- AN 222: Using Precision RTL Synthesis in the Quartus II Design Methodology

**Perform WYSIWYG Re-mapping For Area**

If you initially use a third-party synthesis tool and wish to see if the Quartus II software can re-map the circuit such that fewer LEs are used, perform the following steps.

1. Turn on Perform WYSIWYG Primitive un-mapping on the Netlist Optimizations page of the Settings dialog box (Assignments menu).
2. Choose Area for Optimization technique on the Default Logic Options Settings page of the Settings dialog box (Assignments menu).
3. Re-compile the design.

WYSIWYG re-mapping for area in this way typically reduces $f_{\text{MAX}}$.

**Re-target Memory Blocks**

If the design fails to fit because it runs out of device memory resources, it may be due to a lack of a certain type of memory. For example, a design may require two M-RAM blocks and be targeted for a Stratix EP1S10 device, which has only one. By building one of the memories with a different size memory block, such as M4K memory blocks, it may be possible to obtain a fit.

**Re-target DSP Blocks**

A design may not fit because it requires too many DSP blocks. All of the functions of DSP blocks can be performed with logic cells, making it possible to retarget some of the DSP blocks towards LEs in order to obtain a fit.
Use a Larger Package/Change Pin Assignments

If a design with pin assignments fails to fit, run a compilation without the pin assignments to see whether or not a fit is possible for the design in the specified device and package. Also try this approach if a Quartus II error message indicates fit problems due to pin assignments.

If the design fits when all pin assignments are ignored or when several pin assignments are ignored or moved, then it may be necessary to modify the pin assignments for the design or choose a larger package.

If the design fails to fit due to unavailable I/Os, a successful fit can be obtained by using a larger package.

Use a Larger Device

If a successful fit cannot be achieved due to a shortage of LEs, memory, or DSP blocks, use a larger device or re-architect the design using time domain multiplexing or more efficient RTL coding techniques.

Resolving Resource Utilization Issues

The previous sections detail each of the design options. If these methods do not sufficiently reduce the amount of resources used by the design, modify the design at the source to achieve the desired performance.

Table 2 shows design options used to reduce excess resource utilization and the recommended order to try the options, starting with those requiring the least effort.

The Quartus II software includes a design space explorer (DSE) Tcl/Tk script for automating successive compilations of a design, each employing different design options. For more information on the DSE script, see AN 198: Timing Closure Using the Quartus II Software.

<table>
<thead>
<tr>
<th>Issue</th>
<th>Design Options to Employ (in Order from Right to Left)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Too many logic cells used</td>
<td>Use Register Packing</td>
</tr>
<tr>
<td></td>
<td>Remove Fitter Constraints</td>
</tr>
<tr>
<td></td>
<td>Modify Quartus II Synthesis Options</td>
</tr>
<tr>
<td></td>
<td>Modify EDA Synthesis Tool Options</td>
</tr>
<tr>
<td></td>
<td>Use a Larger Device</td>
</tr>
<tr>
<td>Too many memory blocks used</td>
<td>Modify Quartus II Synthesis Options</td>
</tr>
<tr>
<td></td>
<td>Modify EDA Synthesis Tool Options</td>
</tr>
<tr>
<td></td>
<td>Remove Fitter Constraints</td>
</tr>
<tr>
<td></td>
<td>Retarget Memory Blocks</td>
</tr>
<tr>
<td></td>
<td>Use a Larger Device</td>
</tr>
</tbody>
</table>
Once resource utilization has been optimized, proceed to optimize I/O timing, as described in the following section.

### I/O Optimization Techniques

The third design optimization stage focuses on I/O timing. Because changes to the I/O path affect the internal f_{MAX}, complete this stage before proceeding to the f_{MAX} timing optimization stage.

The options presented in this section address how to improve I/O timing, including the setup delay (t_{su}), hold time (t_{th}), and clock-to-output (t_{co}) parameters.

### Timing-driven Compilation

The following Timing-driven Compilation options are available on the Fitting page in the Compiler settings category of the Settings dialog box (Assignment menu).

Turning on the **Optimize Timing** option and selecting **Timing Driven Compilation** (Fitting page, Settings Dialog box) moves registers into I/O elements if required to meet t_{su} or t_{co} assignments.

Perform I/O timing optimization using the **Optimize I/O cell register placement for timing** assignment located on the Fitting page of the Settings dialog box (Assignments menu). This option is ON by default when using the PowerFit fitter.

The **Optimize I/O cell register placement for timing** option is a global setting.
For APEX™ 20KE/C devices, if the I/O register is not available, the fitter tries to move the register into the logic array block (LAB) adjacent to the I/O element.

This option only affects pins that have a \( t_{\text{tsu}} \) or \( t_{\text{tco}} \) requirement. Using the I/O register is only possible if the register directly feeds a pin or gets fed directly by a pin. This setting does not affect registers that:

- Have combinatorial logic between the register and the pin
- Are part of a carry or cascade chain
- Have an overriding location assignment
- Use the synchronous load or asynchronous load port, and the value is not “1” (Stratix, Stratix GX, and Cyclone devices only)
- Use their synchronous load or asynchronous clear port (APEX and APEX II devices only)

**Fast Input, Output & Output Enable Registers**

Manually place individual registers in an I/O cell by making a fast I/O assignment using the Assignment Editor (Assignments menu). For an input register use the Fast Input Register option, for an output register use the Fast Output Register option, and for an output enable register use the Fast Output Enable Register option.

If the fast I/O setting is ON, the register is always placed in the I/O element. If the fast I/O setting is OFF, the register is not placed in the I/O element. This is true even if the Optimize I/O cell register placement for timing option, located on the Fitting page of the Settings dialog box (Assignments menu), is turned ON. If there is no Fast I/O assignment, the Quartus II software determines whether or not to place registers in I/O elements if the Optimize I/O cell register placement for timing option is turned ON.

The three fast I/O options (Fast Input Register, Fast Output Register, and Fast Output Enable Register) can also be used to override a register that is in a LogicLock region and force it into an I/O cell. If this assignment is applied to a register (including an output enable register) that feeds multiple pins, the register will be duplicated and placed in all relevant I/O elements.

**Programmable Delays**

Various programmable delay options can be enabled to minimize the \( t_{\text{tsu}} \) and \( t_{\text{tco}} \) times. For Stratix, Stratix GX, and Cyclone devices, the Quartus II software automatically adjusts the programmable delays to help meet
timing requirements. Programmable delays are advanced options that should only be used after a project is compiled and the I/O timing checked and determined to be unsatisfactory.

Assign programmable delay options to qualified nodes using the Assignment Editor (Assignments menu.) After an assignment has been made and the design is compiled, you can view the delay chain in the target device using the Quartus II chip editor and resource property editors. Figure 20 shows the resource property editor displaying a programmable delay implemented in the delay chain of a Stratix device.

**Figure 20. Delay Chain Shown in the Quartus II Resource Property Editor**

For more information on using the Quartus II chip and resource property editors, see *AN 310: Using the Quartus II Chip Editor.*
Table 3 summarizes the programmable delays available for Altera devices.

<table>
<thead>
<tr>
<th>Programmable Delay</th>
<th>Description</th>
<th>I/O Timing Impact</th>
<th>Device Families</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decrease Input delay to Input Register</td>
<td>Decreases propagation delay from an input pin to the input register in the I/O cell associated with the pin. Applied to input/bidirectional pin or register it feeds.</td>
<td>Decreased $t_{su}$ Increased $t_{h}$</td>
<td>Stratix, Stratix GX, Cyclone, Apex II, Apex 20KE/C, Mercury, MAX® 7000B</td>
</tr>
<tr>
<td>Decrease Input delay to Internal Cells</td>
<td>Decreases the propagation delay from an input or bidirectional pin to logic cells and embedded cells within the device. Applied to input/bidirectional pin or register it feeds.</td>
<td>Decreased $t_{su}$ Increased $t_{h}$</td>
<td>Stratix, Stratix GX, Cyclone, Apex II, Apex 20KE/C, Mercury, Flex 10K®, Flex® 6000, ACEX® 1K</td>
</tr>
<tr>
<td>Decrease Input Delay to Output Register</td>
<td>Decreases the propagation delay from the interior of the device to the output register in an I/O cell. Applied to input/bidirectional pin or register it feeds.</td>
<td>N/A</td>
<td>Stratix, Stratix GX, Apex II, Apex 20KE/C</td>
</tr>
<tr>
<td>Increase Delay to Output Enable Pin</td>
<td>Increases the propagation delay through the tristate output to the pin. The signal can either come from internal logic or the output enable register in an I/O cell. Applied to output/bidirectional pin or register feeding it.</td>
<td>Increased $t_{co}$</td>
<td>Stratix, Stratix GX, Apex II, Mercury</td>
</tr>
<tr>
<td>Increase Delay to Output Pin</td>
<td>Increases the propagation delay to the output or bidirectional pin from internal logic or the output register in an I/O cell. Applied to output/bidirectional pin or register feeding it.</td>
<td>Increased $t_{co}$</td>
<td>Stratix, Stratix GX, Cyclone, Apex II, Apex 20KE/C, Mercury</td>
</tr>
<tr>
<td>Increase Input Clock Enable Delay</td>
<td>Increases the propagation delay from the interior of the device to the clock enable input of an I/O input register.</td>
<td>N/A</td>
<td>Stratix, Stratix GX, Apex II, Apex 20KE/C</td>
</tr>
<tr>
<td>Increase Output Clock Enable Delay</td>
<td>Increases the propagation delay from the interior of the device to the clock enable input of the I/O output register and output enable register.</td>
<td>N/A</td>
<td>Stratix, Stratix GX, Apex II, Apex 20KE/C</td>
</tr>
<tr>
<td>Increase Output Enable Clock Enable Delay</td>
<td>Increases the propagation delay from the interior of the device to the clock enable input of an output enable register.</td>
<td>N/A</td>
<td>Stratix, Stratix GX</td>
</tr>
<tr>
<td>Increase $t_{zx}$ Delay to Output Pin</td>
<td>Used for zero bus-turnaround (ZBT) by increasing the propagation delay of the falling edge of the output enable signal.</td>
<td>Increased $t_{co}$</td>
<td>Stratix, Stratix GX, Apex II, Mercury</td>
</tr>
</tbody>
</table>
Fast Regional Clock Option

Stratix EP1S25, EP1S20, and EP1S10 devices and Stratix GX EP1SGX10 and EP1SGX25 devices contain two fast regional clock networks, FCLK[1..0], within each quadrant, fed by input pins that can connect to fast regional clock networks. In Stratix EP1S30, Stratix GX EP1SGX40, as well as larger devices in both families, there are two fast regional clock networks within each half-quadrant. Dedicated FCLK input pins can directly feed these clock nets. Fast regional clocks have less of a delay to I/O elements than regional or global clocks and are used for high fan-out control signals. Placing clocks on fast regional clock nets can provide better tCO performance.

Using PLLs to Shift Clock Edges

Using a PLL should automatically help I/O timing. If the timing is still not met, most devices allow the PLL to be phase shifted in order to change the I/O timing. Shifting the clock backwards will give a better tCO at the expense of the tSU, while shifting it forward will give a better tSU at the expense of tCO and tI. This option can only be used with devices that offer PLLs with the phase shift option. See Figure 21.

Figure 21. Shift Clock Edges Forward to Improve tSU at the Expense of tCO

Original

With PLL
Improving Setup & Clock-to-Output Times

Table 4 shows a generally recommended order in which to use options to reduce $t_{su}$ and $t_{co}$ times. Keep in mind that reducing $t_{su}$ times increases hold ($t_H$) times.

<table>
<thead>
<tr>
<th>Option</th>
<th>$t_{su}$ (2)</th>
<th>$t_{co}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set option to ensure that the appropriate constraints are set for the failing I/Os</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Set option to use timing-driven compilation for I/O</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Set option to use fast input register</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Set option to use fast output register and fast output enable register</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Set Decrease Input Delays to Input Register = ON</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Set Decrease Input Delays to Internal Cells = ON</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Set Increase Delay to Output Pin = OFF</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Set option to use PLLs to shift clock edges</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Set option to use the Fast Regional Clock option</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Notes to Table 4:
(1) These options may not apply for all device families.
(2) Reducing $t_{su}$ times increases hold ($t_H$) times.

Once I/O timing has been optimized, proceed to optimize $f_{MAX}$, as described in the following section.

$f_{MAX}$ Optimization Techniques

The fourth design optimization stage is to improve $f_{MAX}$ timing. In the case when performance cannot be met after compiling with the Quartus II software, there are a number of options available to the user.

Netlist Optimization Options

The Quartus II software includes netlist optimization options to optimize your design further than the standard compilation flow, regardless of the synthesis tool used. Depending on your design, some options may have more of an effect than others, and the options can be applied in combination to provide optimal results. The Quartus II netlist optimizations are applied at different stages of the design flow, either during synthesis or during fitting.
The synthesis netlist optimizations occur during the synthesis stage of the Quartus II compilation flow. Operating on output from either a third-party synthesis tool or the Quartus II standard integrated synthesis, these optimizations make changes to the synthesis netlist that are beneficial in terms of area or speed, depending on your selected optimization technique.

The following synthesis netlist optimizations are available:

- WYSIWYG Primitive Re-synthesis
- Gate-level Register Re-timing

The fitter netlist optimizations take place during the fitter stage of the Quartus II compilation flow. These optimizations involve Quartus II re-synthesis of parts of the design as fitting progresses and routing delays become more apparent. Since routing delays are a significant part of the total critical path delay, this allows re-synthesis (and hence re-optimization) of the true critical paths in a design.

This tight integration of fitting and synthesis process is known as physical synthesis. Physical synthesis netlist optimizations make placement-specific changes to the netlist that improve results for a specific Altera device.

The following fitter netlist optimizations are available:

- Physical synthesis for combinatorial logic
- Physical synthesis for registers:
  - Register duplication
  - Register retiming

View and modify the netlist optimization options on the Netlist Optimizations page of the Settings dialog box (Assignments Menu).

For more information on timing closure, refer to AN 198: Timing Closure with the Quartus II Software.
fMAX Optimization Techniques

Optimizing FPGA Performance Using the Quartus II Software

Try these options in different combinations to provide the best results. Performance results are design dependant. Typical benchmark results with netlists from leading third-party synthesis tools and compiled with the Quartus II software version 3.0 are shown in Table 5. These results were obtained using various designs and numbers of LEs, using two different FPGA synthesis tools for average performance gains.

<table>
<thead>
<tr>
<th>Optimization Method</th>
<th>fMAX Gain</th>
<th>Win Ratio (1)</th>
<th>Winner’s fMAX Gain (2)</th>
<th>Increase in LE Area</th>
<th>Increase in Compile Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitive Re-synthesis</td>
<td>3%</td>
<td>60%</td>
<td>7%</td>
<td>-1%</td>
<td>1.2X</td>
</tr>
<tr>
<td>Physical Synthesis for Combinatorial Logic and Registers</td>
<td>10%</td>
<td>86%</td>
<td>12%</td>
<td>5%</td>
<td>2.1X</td>
</tr>
<tr>
<td>Primitive Re-Synthesis as well as Physical Synthesis for Combinatorial Logic and Registers</td>
<td>13%</td>
<td>87%</td>
<td>14%</td>
<td>4%</td>
<td>2.1X</td>
</tr>
<tr>
<td>All options on (Primitive Re-Synthesis, Gate-level Register Re-timing, and Physical Synthesis for Combinatorial Logic and Registers)</td>
<td>13%</td>
<td>82%</td>
<td>15%</td>
<td>3%</td>
<td>2.2X</td>
</tr>
</tbody>
</table>

Notes to Table 5:
(1) Win is the percentage of designs that showed better performance with the option on, than without the option on.
(2) Winner’s fMAX gain refers to the average improvement for the designs that showed better performance with these settings (designs considered a Win).

Seed (Initial Placement Configuration)

Changing the seed affects the initial placement configuration and often causes different fitter results. To obtain a better fMAX value, experiment with different settings. This method should only be attempted if the design is failing timing on a small number of paths and the fMAX is within range. This range is typically about 3% for Stratix devices.

The initial placement configuration is controlled by the Seed setting. To change the Seed setting, choose Settings (Assignments menu) and from the Category list, choose Fitting under Compiler Settings.
Synthesis Options

How a design is synthesized can have a large impact on the performance of the design. Performance can vary depending on the way the design is coded, what synthesis tool is used, and what options are specified when synthesizing. Synthesis options should be used if a large number of paths are failing or specific paths are failing by a large amount and have many levels of logic.

For more information on synthesis options, see the following documents:

- AN 238: Using Quartus II Verilog HDL & VHDL Integrated Synthesis
- AN 226: Synplify & Quartus II Design Methodology
- AN 225: LeonardoSpectrum & Quartus II Design Methodology
- AN 222: Using Precision RTL Synthesis in the Quartus II Design Methodology

Within synthesis tools, it is important to ensure that the frequency is specified for each clock; otherwise the tool optimizes for area. For best performance for push-button compiles:

- Flatten the hierarchy
- Optimize for speed, not area
- Set the effort to high (where applicable)
Within the synthesis tool, experiment with different options including:

- State machine encoding
- Register balancing
- Pipelining
- Fanout control
- Logic replication

It is possible to manually duplicate registers in the Quartus II software regardless of the synthesis tool used. This technique is often used if moving a register in a failing path creates other failing paths or there are timing problems based on fanout of the registers. To duplicate a register, apply the Manual Logic Duplication option to the register using the Assignment Editor. For more information on the Manual Logic Duplication option, refer to the Quartus II Online Help.

The Quartus II software includes a design space explorer (DSE) Tcl/Tk script for automating successive compilations of a design, each employing different design options. For more information on the DSE script, see AN 198: Timing Closure Using the Quartus II Software.

**LogicLock Assignments**

Make LogicLock assignments to optimize based on nodes, design hierarchy, or critical paths. This method should be used if a large number of paths are failing but recoding the design is thought to be unnecessary.

When making LogicLock assignments, it is important to consider what level of flexibility to leave the fitter. LogicLock assignments provide more flexibility than hard location assignments. Assignments that are more flexible require higher fitter effort. The LogicLock assignments possible in order of decreasing flexibility are as follows:

- Soft LogicLock regions
- Auto size, floating location regions
- Fixed size, floating location regions
- Fixed size, locked location regions

To determine what to put into a LogicLock region, refer to the timing analysis results and the Timing Closure Floorplan. The register-to-register f\(_{\text{MAX}}\) paths in the Timing Analyses section of the Compilation Report can provide a helpful method of recognizing patterns. The following sections outline cases where LogicLock regions can help to optimize a design.

For more information on the LogicLock design methodology, refer to AN 161: Using the LogicLock Methodology in the Quartus II Design Software.
Hierarchy Assignments

For a design with the hierarchy shown in Figure 23, which has failing paths in the timing analysis results similar to those shown in Table 6, mod_A is likely a problem module. In this case, mod_A could be placed in a LogicLock region to attempt to put all the nodes in the module closer together in the floorplan.

Figure 23. Design Hierarchy

![Design Hierarchy Diagram]

<table>
<thead>
<tr>
<th>Table 6. Failing Module Paths in Timing Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
</tr>
<tr>
<td>mod_A</td>
</tr>
<tr>
<td>mod_A</td>
</tr>
<tr>
<td>mod_A</td>
</tr>
<tr>
<td>mod_A</td>
</tr>
<tr>
<td>mod_A</td>
</tr>
</tbody>
</table>

Path Assignments

If a pattern as shown in Figure 24 is seen, it is likely an indication of common “problem” paths. In this case, a path-based assignment could be made from all d_reg registers to all memaddr registers. A path-based assignment can be made to place all source registers, destination registers and the nodes between them in a LogicLock region.
Alternatively, the nodes of a critical path can explicitly be placed in a LogicLock region. There may be alternate paths between the source and destination registers that could become critical if this method is used instead of path-based assignments.

**Figure 24. Failing Paths in Timing Analysis**

<table>
<thead>
<tr>
<th>Location Assignments &amp; Back Annotation</th>
</tr>
</thead>
</table>

**Table 7. Failing Paths in Timing Analysis**

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>d_reg[2]</td>
<td>memaddr[0]</td>
</tr>
</tbody>
</table>
If a small number of paths are failing, you can use hard location
assignments to optimize placement. Location assignments are less
flexible for the the Quartus II Fitter than LogicLock assignments. The
commonly used location assignments, in order of decreasing flexibility,
are as follows:

- Custom regions
- Back-annotated LAB location assignments
- Back-annotated LE location assignments

**Custom Regions**

A custom region is a rectangular region containing user-assigned nodes.
These assigned nodes are then constrained within the region’s
boundaries. Custom regions are hard location assignments that cannot be
overridden and are very similar to fixed-size, locked-location LogicLock
regions. Custom regions are commonly used when logic must be
constrained to a specific portion of the device.

**Manual Place & Route: Back Annotation**

Fixing the location of nodes in a design to the location from the last
compilation is known as back-annotation. When all the nodes are back-
annotated, manually moving nodes does not affect the locations of other
design nodes that are locked down, allowing for predictable timing
results. This is referred to as manual-place-and-route. Before making
location assignments, determine whether or not to lock down the location
of all nodes in the design. When a design is back-annotated, the nodes are
either assigned to LABs (this is preferred because of increased flexibility)
or LEs.

Using a Routing Constraint File (.rcf) with fitter netlist
optimizations may cause a routing failure.

When performing manual place-and-route on a detailed level, it is
suggested to move LABs, not logic cells. Quartus II places nodes that
share the same control signals in appropriate LABs. Successful place-and-
route is more difficult when moving logic cells.

For more information on back-annotation of routing, refer to Quartus II
Help.

When considering the Stratix, Stratix GX and Cyclone architectures,
distance is measured in terms of Manhattan distance, which is the sum of
the x and y distance between nodes. Figure 25 shows Manhattan
distances of 3 and 5.
In the Stratix, Stratix GX, and Cyclone architectures, the row interconnect delay is slightly faster than the column interconnect delay. As a result, the path with less column interconnects shown in Figure 26 should be faster than the path with more column interconnects even though they have the same Manhattan distance.

When placing nodes, optimal placement is an ellipse as measured in Manhattan distance around the source or destination node. In Figure 27, if the source is located in the center, any of the shaded LABs should give approximately the same delay.
Optimizing FPGA Performance Using the Quartus II Software

Optimizing FPGA Performance Using the Quartus II Software

To determine the actual delays to and from a resource, use the **Show Physical Timing Estimate** feature in the **Timing Closure Floorplan**.

In general, when using manual place-and-route, it is best to fix all I/O paths first. This is because there are often fewer options available to meet I/O timing. After I/O timing has been met, focus on manually placing \( f_{\text{MAX}} \) paths.

The best way to meet performance is to move nodes closer together. For a critical path as shown in **Figure 28**, moving the destination node closer to the other nodes reduces the delay and may meet timing.

---

**Figure 27. Possible Optimal Placement Ellipse**

---

**Figure 28. Reducing Delay of Critical Path**
For the Mercury, Apex II, and Apex 20KE/C architectures, reducing the Manhattan distance may not reduce the delay. The delay for paths should be reduced by placing the source and destination nodes in the same geographical resource location. From fastest to slowest, the resources are as follows:

- LAB
- MegaLAB™
- MegaLAB column
- Row

For example, if the nodes cannot be placed in the same MegaLAB to reduce the delay, they should be placed in the same MegaLAB column. For the actual delays to and from resources, use the Show Physical Timing Estimate feature in the Timing Closure Floorplan.

**f**\text{MAX Improvement Summary}

The choice of options and the adjustment of settings to improve \( f_{\text{MAX}} \) depends on the failing paths in the design. To achieve the best results relative to your performance requirements, apply the following options, compiling after each.

1. Apply netlist optimization options.
2. Modify the seed. (This step may be omitted if a large number of critical paths are failing, or if paths are failing by large amounts.)
3. Apply synthesis options.
4. Make LogicLock assignments.
5. Make location assignments, or perform manual place-and-route by back-annotating the design.

If the options presented above do not achieve performance requirements, design modifications may be required.

The Quartus II software includes a design space explorer (DSE) Tcl/Tk script for automating successive compilations of a design, each time employing different combinations of netlist optimizations, initial placement configuration settings, and synthesis options. For more information on the DSE script, see *AN 198: Timing Closure Using the Quartus II Software*. 

---

Altera Corporation
Conclusion

The Quartus II software provides many features to effectively achieve optimal performance. Designers who understand the techniques presented in this application note can quickly optimize a design.

References

Synthesis Design Guidelines

AN 238: Using Quartus II Verilog HDL & VHDL Integrated Synthesis
AN 226: Synplify & Quartus II Design Methodology
AN 225: LeonardoSpectrum & Quartus II Design Methodology
AN 222: Using Precision RTL Synthesis in the Quartus II Design Methodology

I/O Standards

AN 253: Using Selectable I/O Standards in Cyclone Devices
Using Selectable I/O Standards in Stratix & Stratix GX Devices chapter of the Stratix Device Handbook
AN 166: Using High-Speed I/O Standards in APEX II Devices
AN 134: Programmable I/O Standards in Mercury Devices
AN 117: Using Selectable I/O Standards in APEX 20KE, APEX 20KC & MAX 7000B Devices

Timing Analysis

AN 123: Using Timing Analysis in the Quartus II Software

Quartus II Online Help

Timing Closure

AN 198: Timing Closure with the Quartus II Software
AN 161: Using the LogicLock Methodology in the Quartus II Design Software
Revision History

The information contained in AN 297: Optimizing FPGA Performance Using the Quartus II Software version 1.2 supersedes information published in previous versions.

Version 1.2

This version of the document contains the following changes:

- Added “Perform WYSIWYG Re-mapping For Area” on page 18.
- Made changes throughout to update the document for the Quartus II software version 3.0.
- Made minor text updates throughout the document.