Using techniques for achieving the highest design performance is more important with larger density FPGAs. The easy-to-use and effective place-and-route features of the Quartus® II software version 2.2 allow designers to meet performance requirements. To meet performance requirements, it is recommended to perform an initial compilation for fit verification and then optimize for resource utilization, I/O timing, and fMAX timing. See Figure 1.

**Figure 1. Design Optimization Sequence**

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, we recommend you read AN 198: Timing Closure Using the Quartus II Software and AN 161: Using the LogicLock Methodology in the Quartus II Design Software.

The first design optimization stage is to 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 33 for guidelines and recommendations on coding designs for optimal performance.

Compiler Settings

Compiler settings determine how a design is compiled in the Quartus II software. These settings determine what timing information is used, what fitter algorithm is used, and whether or not there may be adjustments to be made in one area or another.

Device Settings

Assigning a specific device determines the timing model referred to during compilation. It is important to choose the correct speed grade in order to obtain accurate results and the optimal 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

The Optimize timing option allows the Quartus II software to optimize the fitting of the design to meet the specified performance requirements for all timing constraints in the design. Use this option when trying to achieve desired timing results.

The Optimize I/O cell register placement for timing option causes the Quartus II software to determine whether or not registers are placed in I/O elements in order to meet a strict $t_{CO}$ or $t_{SU}$ timing requirement. For more information on this option see “I/O Optimization” on page 18.

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

Figure 3. Fitting Settings
Fast Fit

This option reduces the compilation time by about 50% compared to a standard fit depending on the design. This reduction is achieved by allowing the software to stop after the first successful fit is calculated rather than continuing to fit the best fit. The performance achieved using this option is typically slightly lower than that achieved without the Fast Fit option. This option can be selected 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 standard and drive strength employed in a design each affect I/O timing. I/O standards can be assigned through the Assign Pins dialog box by clicking Assign Pins on the Device page of the Settings dialog box (Assignments menu). See Figure 4.

For more information on I/O standards refer to the documents listed under “I/O Standards” on page 33.

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 is a timing-driven fitter and attempts to exceed the desired timing requirements specified. Netlist optimizations are also performed based on the timing constraints specified. See the “Netlist Optimization Options” on page 22 section 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.

Timing settings define which paths should be considered critical in order to meet required performance. Every clock should have an accurate clock setting. All I/Os for which tSU or tCO 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 tradeoffs between paths.

For more information on how to make timing assignments, refer to the documents listed under “Timing Analysis” on page 33.

After appropriate initial compilation assignments have been made and the design compiled, proceed to analyze the design to determine if resource utilization, I/O timing, and/or fMAX timing needs to be optimized.

Design Analysis

The initial compilation establishes whether the initially compiled 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 fMAX timing.
- If the design does not meet performance requirements, check to see if I/O performance is the issue. If the I/O timing performance is not met, see “I/O Optimization” on page 18 before trying to optimize fMAX timing.
- If the design does not meet performance requirements, check to see if fMAX performance is the issue. If fMAX performance is not achieved, see “Improving fMAX” on page 31.
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 floorplan 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 13.

To determine resource usage, refer to the Summary section of the Compilation Report as shown in Figure 5. This section reports how many logic elements (LEs), pins, memory bits and digital signal processing (DSP) block 9-bit elements were used. 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 of the Compilation Report.

Figure 5. Compilation Report Summary

If all resources are 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 quickly, then a resource is 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. See Figure 6.

**Figure 6. Routing Congestion in the Timing Closure Floorplan**

---

**I/O Timing**

To determine whether or not I/O timing has been met, see the **Timing Analyses** folder in the **Compilation Report**. If timing requirements have been entered, there are folders for “tsu Requirements,” “th Requirements,” and “tco Requirements.” Clicking on these folders lists the I/Os and the slack associated with each. The I/Os that have not met the required timing have a negative slack and are seen in red as shown in Figure 7.

For suggestions on how to improve I/O timing, see “I/O Optimization” on page 18.
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. See Figure 8. This is a good method of determining where along the path the greatest delay is located.

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. This can also be done by viewing the path in the Last Compilation Floorplan.
Figure 9. Locate Failing Path in Timing Closure Floorplan

<table>
<thead>
<tr>
<th>Source Name</th>
<th>Destination Name</th>
<th>Source Clock Name</th>
<th>Required Ioc</th>
<th>Actual Ioc</th>
<th>Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>data_in[0]</td>
<td>data_in[1]</td>
<td></td>
<td>0.388 ns</td>
<td>-1.238 ns</td>
<td></td>
</tr>
<tr>
<td>clock</td>
<td>clock</td>
<td></td>
<td>3.154 ns</td>
<td>-1.238 ns</td>
<td></td>
</tr>
<tr>
<td>clock</td>
<td>clock</td>
<td></td>
<td>3.125 ns</td>
<td>-1.195 ns</td>
<td></td>
</tr>
<tr>
<td>in[4]</td>
<td>clock</td>
<td></td>
<td>0.183 ns</td>
<td>-0.153 ns</td>
<td></td>
</tr>
<tr>
<td>rate[0]</td>
<td>s_out[0]</td>
<td>clk2</td>
<td>8.000 ns</td>
<td>7.122 ns</td>
<td>0.823 ns</td>
</tr>
<tr>
<td>rate[0]</td>
<td>s_out[2]</td>
<td>clk2</td>
<td>8.000 ns</td>
<td>7.126 ns</td>
<td>0.854 ns</td>
</tr>
<tr>
<td>rate[2]</td>
<td>s_out[2]</td>
<td>clk2</td>
<td>8.000 ns</td>
<td>7.116 ns</td>
<td>0.882 ns</td>
</tr>
<tr>
<td>rate[3]</td>
<td>s_out[7]</td>
<td>clk2</td>
<td>8.000 ns</td>
<td>7.117 ns</td>
<td>0.883 ns</td>
</tr>
</tbody>
</table>

Figure 10. Failing Path in Timing Closure Floorplan

5.220 ns
Figure 11 shows all failing I/O paths. To view all these paths in the Timing Closure Floorplan use the Show Critical Paths feature.

**Figure 11. Critical I/O Paths in the Timing Closure Floorplan**

---

**f_{MAX} Timing**

To determine whether or not f_{MAX} timing requirements are met, refer to the Timing Analyses Folder in the Compilation Report. For non-complex timing assignments, an f_{MAX} folder and a Register to Register f_{MAX} folder exist. With complex timing assignments, the Clock Requirement folder contains slack information for each clock. The paths that are not meeting timing requirements are shown in red. See Figure 12.

Complex timing assignments, such as in scenarios where there are unrelated clocks in the design, require the Quartus II software to perform complex timing analysis.

For suggestions on how to improve f_{MAX} timing, see “f_{MAX} Optimization” on page 22.
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 to determine the location of the greatest delay.

Visually analyze fMAX 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**

<table>
<thead>
<tr>
<th>Source Name</th>
<th>Destination Name</th>
<th>Source Clock</th>
<th>Destination Clock</th>
<th>Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>data_mem[1][0]</td>
<td>clock</td>
<td>clock</td>
<td>clock</td>
<td>0.115 ns</td>
</tr>
<tr>
<td>store_mem[1][1]</td>
<td>output3[1][10]</td>
<td>clock</td>
<td>clock</td>
<td>0.117 ns</td>
</tr>
<tr>
<td>state_mem[1][1]</td>
<td>output3[1][10]</td>
<td>clock</td>
<td>clock</td>
<td>0.136 ns</td>
</tr>
<tr>
<td>state_mem[1][1]</td>
<td>output3[1][7]</td>
<td>clock</td>
<td>clock</td>
<td>0.177 ns</td>
</tr>
<tr>
<td>tape_mem[1]</td>
<td>output3[1][11]</td>
<td>clock</td>
<td>clock</td>
<td>0.229 ns</td>
</tr>
<tr>
<td>state_mem[1][1]</td>
<td>output3[1][3]</td>
<td>clock</td>
<td>clock</td>
<td>0.259 ns</td>
</tr>
<tr>
<td>state_mem[1][1]</td>
<td>output3[1][3]</td>
<td>clock</td>
<td>clock</td>
<td>0.259 ns</td>
</tr>
<tr>
<td>state_mem[1][1]</td>
<td>output3[1][3]</td>
<td>clock</td>
<td>clock</td>
<td>0.259 ns</td>
</tr>
<tr>
<td>state_mem[1][1]</td>
<td>output3[1][3]</td>
<td>clock</td>
<td>clock</td>
<td>0.321 ns</td>
</tr>
<tr>
<td>state_mem[1][1]</td>
<td>output3[1][3]</td>
<td>clock</td>
<td>clock</td>
<td>0.352 ns</td>
</tr>
</tbody>
</table>

**Figure 15. Failing Path in Timing Closure Floorplan**
It is also possible to 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.

**Figure 16. Critical I/O Paths in the Timing Closure Floorplan**

---

**Resource Utilization Optimization Techniques**

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**

This option is called **Auto Packed Registers** and can be used regardless of which tool was used to synthesize the design. Register packing combines logic cells that only use the register and logic cells that use only the lookup table (LUT) into a single logic cell. Figure 17 shows the packing and the gain of one logic cell.

**Figure 17. Register Packing**
The three most common cases in which register packing can help to optimize a design are:

- A LUT can be implemented in the same cell as an unrelated register with a single data input
- A 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

There are four options available for register packing.

- **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, including those part of carry and cascade chains, 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, go to **Assignments -> Settings -> Packed Registers** option on the **Default Logic Option Settings** page of the **Settings** dialog box (Assignments menu) as shown in Figure 18.
Figure 18. Auto Packed Registers Option

The area reduction and performance results can vary greatly depending on the design. Register packing produces the typical results for Stratix and Cyclone devices shown in Table 1. Results will vary depending on the design.

<table>
<thead>
<tr>
<th>Register Packing Setting</th>
<th>Relative $f_{\text{MAX}}$</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. This case often 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. Then remove any location and/or LogicLock assignments that are 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 compiles by adding small amounts of constraints each run.

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 techniques that can be applied 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 vs. speed. Also, for some synthesis tools, not specifying an f_MAX may result in less logic utilization.

Retarget Memory Blocks

If the design fails to fit because it runs out of memory, 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 an 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.

Retarget 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, so it may be 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. Try this approach if a Quartus II software 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. Consider this option if a larger device is a viable option. Selecting a larger device is often a last resort if the other options do not yield a successful fit. A more difficult option is to re-architect the design for a technique like time domain multiplexing or more efficient RTL.

Table 2 shows the design options that can be used to reduce identified excess resource utilization during the design analysis. Table 2 lists the recommended order to try the options, starting with those requiring the least effort.

The “Resource Utilization Optimization Techniques” section details 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>
<thead>
<tr>
<th>Issue</th>
<th>Design Options to Employ in Order of Least Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Use Register Packing</td>
</tr>
<tr>
<td>Too many logic cells used</td>
<td>1</td>
</tr>
<tr>
<td>Too many memory blocks used</td>
<td>3</td>
</tr>
<tr>
<td>Too many DSP blocks used</td>
<td>3</td>
</tr>
</tbody>
</table>

Alterna Corporation
I/O Optimization

The third design optimization stage is to improve I/O timing. It is good practice to complete this stage before proceeding to the \( f_{\text{MAX}} \) timing optimization stage. This is because changes to the I/O path affect the internal \( f_{\text{MAX}} \). The options presented in this section address how to improve I/O timing, including the setup delay \( (t_{\text{su}}) \), hold time \( (t_{\text{h}}) \), and clock-to-output \( (t_{\text{co}}) \) parameters.

### Timing-driven Compilation (TDC)

The **Optimize Timing** option turns on TDC for I/O. The **Timing-Driven Compilation** option adds the I/O register benefits. **Timing-driven Compilation** is set for 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. **Timing-driven Compilation** for I/Os move registers into I/O elements if required to meet \( t_{\text{su}} \) or \( t_{\text{co}} \) assignments.

---

### Table 2. Techniques for Resolving Resource Utilization Issues (Part 2 of 2)

<table>
<thead>
<tr>
<th>Issue</th>
<th>Design Options to Employ in Order of Least Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Use Register Packing</td>
</tr>
<tr>
<td>Too many I/O resources used</td>
<td></td>
</tr>
<tr>
<td>Too many routing resources used</td>
<td>1</td>
</tr>
</tbody>
</table>

**Notes to Table 2:**

1. The numbers in this table represent the recommended order to try the design option, starting with those requiring the least effort labeled as “1.”
2. In many cases, retargeting DSP blocks is easily done by using the MegaWizard® Plug-In Manager to change the target.
Optimizing FPGA Performance Using the Quartus II Software

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 affect pins that have a t<sub>su</sub> or t<sub>co</sub> 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 combinational 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 and Cyclone devices only)
- Use their synchronous load or asynchronous clear port (APEX and APEX II devices only)

Fast Input, Output & Output Enable Registers

Individual registers can be manually placed in an I/O cell with a Fast I/O assignment. This can be done for an input register with the Fast Input Register option, an output register with the Fast Output Register option and for an output enable register with 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, 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.

Fast I/O options 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 that feeds multiple pins, the register will be duplicated and placed in all relevant I/O elements.

The Fast Input Register, Fast Output Register, and Fast Output Enable Register options can be applied to pins or to registers that feed or are fed by pins. These options can be set in the Assignment Organizer or the Assignment Editor.
Optimizing FPGA Performance Using the Quartus II Software

Programmable Delays

Various programmable delay options can be enabled to minimize the $t_{su}$ and $t_{co}$ times. For Stratix 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.

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 $t_{hi}$</td>
<td>Stratix, Stratix GX, Cyclone, Apex II, Apex 20KE/C, Mercury, MAX$^\text{TM}$ 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 $t_{hi}$</td>
<td>Stratix, Stratix GX, Cyclone, Apex II, Apex 20KE/C, Mercury, Flex 10K$^\text{®}$, Flex$^\text{®}$ 6000, ACEX$^\text{®}$ 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>Decreased $t_{su}$ Increased $t_{hi}$</td>
<td>Stratix, Stratix GX, Apex II, Apex 20KE/C</td>
</tr>
</tbody>
</table>
Optimizing FPGA Performance Using the Quartus II Software

**Fast Regional Clock Option**

Stratix and Stratix GX devices have two fast regional clocks in each 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 can be used for high fan-out control signals. Placing clocks on fast regional clock nets can provide better $t_{co}$ 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 $t_{co}$ at the expense of the $t_{su}$, while shifting it forward will give a better $t_{su}$ at the expense of $t_{co}$ and $t_{h}$. This option can only be used with devices that offer PLLs with the phase shift option. See Figure 19.

**Table 3. Programmable Delays for Altera Devices (Part 2 of 2)**

<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>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>$t_{co}$</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>$t_{co}$</td>
<td>Stratix, Stratix GX</td>
</tr>
<tr>
<td>Increase $t_{z_{pu}}$ 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>$t_{co}$</td>
<td>Stratix, Stratix GX, Apex II, Mercury</td>
</tr>
</tbody>
</table>

**Figure 19. Shift Clock Edges Backwards to Improve $t_{SU}$ at Expense of $t_{CO}$**

Original

With PLL
Optimizing FPGA Performance Using the Quartus II Software

Improving Setup & Clock-to-Output Times

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

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

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

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

\( f_{\text{MAX}} \) Optimization

The fourth design optimization stage is to improve \( f_{\text{MAX}} \) 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 offers a number of Netlist Optimization options to optimize a design after synthesis and before or during place-and-route. These options can make use of fitter timing information in order to modify the synthesis netlist to achieve higher performance. The options offered are:

- Perform what-you-see-is-what-you-get (WYSIWYG) primitive resynthesis
- Perform gate-level register retiming
- Use fitter timing information
- Automatically duplicate LEs
- Perform LE level look-up table (LUT) resynthesis
These options can be tried 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 2.2 software 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 5. Performance by Netlist Optimization Method

<table>
<thead>
<tr>
<th>Optimization Method</th>
<th>$f_{\text{MAX}}$ Gain</th>
<th>Win Ratio (1)</th>
<th>Winner’s $f_{\text{MAX}}$ gain (2)</th>
<th>LE Area Increase</th>
<th>Compile Time Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitive resynthesis</td>
<td>3.1%</td>
<td>64%</td>
<td>7.5%</td>
<td>-0.7%</td>
<td>1.3X</td>
</tr>
<tr>
<td>Duplicate LEs + LUT resynthesis</td>
<td>3.7%</td>
<td>79%</td>
<td>5.0%</td>
<td>1.8%</td>
<td>1.5X</td>
</tr>
<tr>
<td>Primitive resynthesis + fitter timing info</td>
<td>3.9%</td>
<td>70%</td>
<td>7.7%</td>
<td>-3.9%</td>
<td>2.4X</td>
</tr>
<tr>
<td>Primitive resynthesis + retiming</td>
<td>5.7%</td>
<td>69%</td>
<td>10.7%</td>
<td>0.2%</td>
<td>1.3X</td>
</tr>
<tr>
<td>Primitive resynthesis + retiming + duplicate LEs + LUT resynthesis</td>
<td>9.1%</td>
<td>86%</td>
<td>11.5%</td>
<td>1.1%</td>
<td>2.0X</td>
</tr>
<tr>
<td>All options on</td>
<td>9.9%</td>
<td>88%</td>
<td>12.0%</td>
<td>-1.5%</td>
<td>3.2X</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 $f_{\text{MAX}}$ gain is of the designs that showed better performance (ones contributing to Win) what was the average improvement.

Initial Placement Configuration

Changing the Initial placement configuration setting will likely offer different results. To obtain a better $f_{\text{MAX}}$, experiment with different settings. This method should only be attempted if the design is not failing by a very large number of paths and the $f_{\text{MAX}}$ is within range. This range is typically about 3% for Stratix devices.

To change the Initial placement configuration setting, go to the Fitting page of the Settings dialog box (Assignments menu) as shown in Figure 20.
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 see the documentation listed under “Synthesis Design Guidelines” on page 33.

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

Within the synthesis tool, experiment with different options including:

- State machine encodings
- Register balancing
Optimizing FPGA Performance Using the Quartus II Software

- 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, select Manual Logic Duplication option from the Assignment Organizer or Assignment Editor and apply it to the register. For more information on the Manual Logic Duplication option, refer to the Quartus II Online Help.

LogicLock Assignments

LogicLock assignments can be used 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_max paths in the Timing Analyses section of the Compilation Report can provide a helpful method of recognizing patterns.

Location Assignments

If a small number of paths are failing, use hard location assignments to optimize placement. Hard location assignments provide the fitter with more control and the user with less flexibility.

Location assignments are less flexible 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

Hierarchy Assignments

For a design with the hierarchy shown in Figure 21, 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 21. Design Hierarchy

![Design Hierarchy Diagram]

Table 6. Failing Module Paths in Timing Analysis

<table>
<thead>
<tr>
<th>mod_A</th>
<th>reg1</th>
<th>mod_A</th>
<th>reg9</th>
</tr>
</thead>
<tbody>
<tr>
<td>mod_A</td>
<td>reg3</td>
<td>mod_A</td>
<td>reg5</td>
</tr>
<tr>
<td>mod_A</td>
<td>reg4</td>
<td>mod_A</td>
<td>reg6</td>
</tr>
<tr>
<td>mod_A</td>
<td>reg7</td>
<td>mod_A</td>
<td>reg10</td>
</tr>
<tr>
<td>mod_A</td>
<td>reg0</td>
<td>mod_A</td>
<td>reg2</td>
</tr>
</tbody>
</table>

Path Assignments

If a pattern as shown in Figure 22 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 over path-based assignments.

Figure 22. Failing Paths in Timing Analysis

Table 7. Failing Paths in Timing Analysis

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>d_reg[2]</td>
<td>memaddr[0]</td>
</tr>
</tbody>
</table>
Custom Regions

A custom region is a rectangular region that nodes can be assigned to as specified by the user. These assigned nodes are then constrained within the region’s boundaries. Custom regions are hard assignments that cannot be overridden and are very similar to LogicLock regions. Nodes and hierarchies can be assigned to custom regions. Custom regions are commonly used when logic must be constrained to a specific portion of the device.

Manual Place & Route

Before making location assignments, determine whether or not to lock down the location of all nodes in the design. This is called back-annotation. When a design is back-annotated, the nodes are either assigned to LABs or to LEs. It is recommended to back-annotate to LABs. When all the nodes are back-annotated, manually moving nodes does not affect the locations of other design nodes that are locked down and allows predictable timing results. This is referred to as manual-place-and-route.

When performing manual place-and-route on a detailed level, it is suggested to move LABs, not logic cells. The reason for this is that Quartus II places nodes that share the same control signals in appropriate LABs. Moving logic cells may result in a lower probability of achieving a successful route.

When considering the Stratix, Stratix GX and Cyclone architectures, distance is measured in terms of Manhattan distance, which is the distance between nodes as the sum of the x and y distances. Figure 23 shows Manhattan distances or 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 24 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 25, if the source is located in the center, any of the shaded LABs should give approximately the same delay.

![Figure 25. Possible Optimal Placement Ellipse](image)

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 26, moving the destination node closer to the other nodes reduces the delay and may meet timing.
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 column. For the actual delays to and from resources, use the Show Physical Timing Estimate feature in the Timing Closure Floorplan.

**Improving \( f_{\text{MAX}} \)**

Which options to try to improve \( f_{\text{MAX}} \) depend on the failing paths in the design. See Table 8. If the options presented do not achieve performance requirements, design modifications may be required.
### Table 8. Methods for Improving $f_{\text{MAX}}$ (1)

<table>
<thead>
<tr>
<th>Issue</th>
<th>$f_{\text{MAX}}$ Improvement Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical path falling by a small amount</td>
<td>1 2 3 4 5 6 7 8</td>
</tr>
<tr>
<td>Large number of paths failing or paths are failing by large amounts</td>
<td>1 2 3 4 5 6 7</td>
</tr>
</tbody>
</table>

**Note to Table 8:**
(1) The numbers in this table represent the recommended order to try the design option, starting with the option labeled as “1.”

**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

I/O Standards
- 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

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