

This chapter describes techniques to reduce resource usage when designing for Altera® devices.

This chapter includes the following topics:

- “Resource Utilization” on page 14–1
- “Optimizing Resource Utilization (LUT-Based Devices)” on page 14–2
- “Optimizing Resource Utilization (Macrocell-Based CPLDs)” on page 14–12
- “Scripting Support” on page 14–17

Resource Utilization

Determining device utilization is important regardless of whether your design achieved a successful fit. If your compilation results in a no-fit error, resource utilization information is important for analyzing the fitting problems in your design. If your fitting is successful, review the resource utilization information to determine whether the future addition of extra logic or other design changes might introduce fitting difficulties. Also, review the resource utilization information to determine if it is impacting timing performance.

To determine resource usage, refer to the **Flow Summary** section of the Compilation Report. This section reports resource utilization, including pins, memory bits, digital signal processing (DSP) blocks, and phase-locked loops (PLLs). **Flow Summary** indicates whether your design exceeds the available device resources. More detailed information is available by viewing the reports under **Resource Section** in the **Fitter** section of the Compilation Report.

Flow Summary shows the overall logic utilization. The Fitter can spread logic throughout the device, which may lead to higher overall utilization.

As the device fills up, the Fitter automatically searches for logic functions with common inputs to place in one ALM. The number of packed registers also increases. Therefore, a design that has high overall utilization might still have space for extra logic if the logic and registers can be packed together more tightly.

The reports under the **Resource Section** in the **Fitter** section of the Compilation Report provide more detailed resource information. The Fitter Resource Usage Summary report breaks down the logic utilization information and provides other resource information, including the number of bits in each type of memory block. This panel also contains a summary of the usage of global clocks, PLLs, DSP blocks, and other device-specific resources.

You can also view reports describing some of the optimizations that occurred during compilation. For example, if you use Quartus® II integrated synthesis, the reports in the Optimization Results folder in the **Analysis & Synthesis** section include information about registers that integrated synthesis removed during synthesis. Use this report to estimate device resource utilization for a partial design to ensure that registers were not removed due to missing connections with other parts of the design.

If a specific resource usage is reported as less than 100% and a successful fit cannot be achieved, either there are not enough routing resources or some assignments are illegal. In either case, a message appears in the **Processing** tab of the Messages window describing the problem.

If the Fitter finishes unsuccessfully and runs much faster than on similar designs, a resource might be over-utilized or there might be an illegal assignment. If the Quartus II software seems to run for an excessively long time compared to runs on similar designs, a legal placement or route probably cannot be found. In the Compilation Report, look for errors and warnings that indicate these types of problems.

You can use the Chip Planner to find areas of the device that have routing congestion on specific types of routing resources. If you find areas with very high congestion, analyze the cause of the congestion. Issues such as high fan-out nets not using global resources, an improperly chosen optimization goal (speed versus area), very restrictive floorplan assignments, or the coding style can cause routing congestion. After you identify the cause, modify the source or settings to reduce routing congestion.

- ❓ For more information about Fitter Resources Report, refer to *Fitter Resources Report* in Quartus II Help. For information about how to view routing congestion, refer to *Displaying Resources and Information* in Quartus II Help. For information about using the Chip Planner tool, refer to *About the Chip Planner* in Quartus II Help. For details about using the Chip Planner tool, refer to the *Analyzing and Optimizing the Design Floorplan with the Chip Planner* chapter of the *Quartus II Handbook*.

Optimizing Resource Utilization (LUT-Based Devices)

After design analysis, the next stage of design optimization is to improve resource utilization. Complete this stage before proceeding to I/O timing optimization or register-to-register timing optimization. Ensure that you have already set the basic constraints “Initial Compilation: Required Settings” section in the *Design Optimization Overview* chapter of the *Quartus II Handbook* before proceeding with the resource utilization optimizations described in this section. If a design does not fit into a specified device, use the techniques in this section to achieve a successful fit.

- After you optimize resource utilization and your design fits in the desired target device, optimize I/O timing as described in the *I/O Timing Optimization Techniques (LUT-Based Devices)* section in the *Timing Closure and Optimization* chapter of the *Quartus II Handbook*. These tips are valid for all FPGA families and the MAX® II family of CPLDs.

Using the Resource Optimization Advisor

The Resource Optimization Advisor provides guidance in determining settings that optimize resource usage. To run the Resource Optimization Advisor, on the Tools menu, point to **Advisors**, and click **Resource Optimization Advisor**.

The Resource Optimization Advisor provides step-by-step advice about how to optimize resource usage (logic element, memory block, DSP block, I/O, and routing) of your design. Some of the recommendations in these categories might conflict with each other. Altera recommends evaluating the options and choosing the settings that best suit your requirements.

- ❓ For more information about the Resource Optimization Advisor, refer to *Resource Optimization Advisor Command (Tools Menu)* in Quartus II Help.

Resolving Resource Utilization Issues Summary

Resource utilization issues can be divided into the following three categories:

- Issues relating to I/O pin utilization or placement, including dedicated I/O blocks such as PLLs or LVDS transceivers (refer to “*I/O Pin Utilization or Placement*”).
- Issues relating to logic utilization or placement, including logic cells containing registers and LUTs as well as dedicated logic, such as memory blocks and DSP blocks (refer to “*Logic Utilization or Placement*” on page 14-4).
- Issues relating to routing (refer to “*Routing*” on page 14-9).

I/O Pin Utilization or Placement

Use the suggestions in the following sections to help you resolve I/O resource problems.

Use I/O Assignment Analysis

To help with pin placement, on the Processing menu, point to **Start** and click **Start I/O Assignment Analysis**. The **Start I/O Assignment Analysis** command allows you to check your I/O assignments early in the design process. You can use this command to check the legality of pin assignments before, during, or after compilation of your design. If design files are available, you can use this command to accomplish more thorough legality checks on your design’s I/O pins and surrounding logic. These checks include proper reference voltage pin usage, valid pin location assignments, and acceptable mixed I/O standards.

Common issues with I/O placement relate to the fact that differential standards have specific pin pairings and certain I/O standards might be supported only on certain I/O banks.

If your compilation or I/O assignment analysis results in specific errors relating to I/O pins, follow the recommendations in the error message. Right-click the message in the Messages window and click **Help** to open the Quartus II Help topic for this message.

Modify Pin Assignments or Choose a Larger Package

If a design that has pin assignments fails to fit, compile the design without the pin assignments to determine whether a fit is possible for the design in the specified device and package. You can use this approach if a Quartus II error message indicates fitting problems due to pin assignments.

If the design fits when all pin assignments are ignored or when several pin assignments are ignored or moved, you might have to modify the pin assignments for the design or select a larger package.

If the design fails to fit because insufficient I/O pins are available, a successful fit can often be obtained by using a larger device package (which can be the same device density) that has more available user I/O pins.



For more information about I/O assignment analysis, refer to the *I/O Management* chapter in volume 2 of the *Quartus II Handbook*.

Logic Utilization or Placement

Use the suggestions in the following sections to help you resolve logic resource problems, including logic cells containing registers and LUTs, as well as dedicated logic such as memory blocks and DSP blocks.

Optimize Source Code


If your design does not fit because of logic utilization, evaluate and modify the design at the source to achieve the desired results. You can often improve logic significantly by making design-specific changes to your source code. This is typically the most effective technique for improving the quality of your results.

If your design does not fit into available logic elements (LEs) or ALMs, but you have unused memory or DSP blocks, check if you have code blocks in your design that describe memory or DSP functions that are not being inferred and placed in dedicated logic. You might be able to modify your source code to allow these functions to be placed into dedicated memory or DSP resources in the target device.

Ensure that your state machines are recognized as state machine logic and optimized appropriately in your synthesis tool. State machines that are recognized are generally optimized better than if the synthesis tool treats them as generic logic. In the Quartus II software, you can check for the State Machine report under **Analysis & Synthesis** in the Compilation Report. This report provides details, including the state encoding for each state machine that was recognized during compilation. If your state machine is not being recognized, you might have to change your source code to enable it to be recognized.



For coding style guidelines, including examples of HDL code for inferring memory and DSP functions, refer to the “Instantiating Altera Megafunctions” and the “Inferring Multiplier and DSP Functions from HDL Code” sections of the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*. For guidelines and sample HDL code for state machines, refer to the “General Coding Guidelines” section of the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*.



-  For additional HDL coding examples, refer to *AN 584: Timing Closure Methodology for Advanced FPGA Designs*.

Optimize Synthesis for Area, Not Speed

If your design fails to fit because it uses too much logic, resynthesize the design to improve the area utilization. First, ensure that you have set your device and timing constraints correctly in your synthesis tool. Particularly when area utilization of the design is a concern, ensure that you do not over-constrain the timing requirements for the design. Synthesis tools generally try to meet the specified requirements, which can result in higher device resource usage if the constraints are too aggressive.

If resource utilization is an important concern, some synthesis tools offer an easy way to optimize for area instead of speed. If you are using Quartus II integrated synthesis, select **Balanced** or **Area** for the **Optimization Technique**. You can also specify an **Optimization Technique** 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) or **Speed**. You can also use the **Speed Optimization Technique for Clock Domains** logic option to specify that all combinational logic in or between the specified clock domain(s) is optimized for speed.



In some synthesis tools, not specifying an f_{MAX} requirement can result in less resource utilization.

-  In the Quartus II software, the **Balanced** setting typically produces utilization results that are very similar to those produced by the **Area** setting, with better performance results. The **Area** setting can give better results in some cases.
-  For information about setting the timing requirements and synthesis options in Quartus II integrated synthesis and other synthesis tools, refer to the appropriate chapter in *Synthesis* in volume 1 of the *Quartus II Handbook*, or your synthesis software's documentation.

The Quartus II software provides additional attributes and options that can help improve the quality of your synthesis results.

Restructure Multiplexers

Multiplexers form a large portion of the logic utilization in many FPGA designs. By optimizing your multiplexed logic, you can achieve a more efficient implementation in your Altera device.

-  For more information about this option, refer to *Restructure Multiplexers logic option* in Quartus II Help.
-  For design guidelines to achieve optimal resource utilization for multiplexer designs, refer to the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*.

Perform WYSIWYG Primitive Resynthesis with Balanced or Area Setting

The **Perform WYSIWYG Primitive Resynthesis** logic option specifies whether to perform WYSIWYG primitive resynthesis during synthesis. This option uses the setting specified in the **Optimization Technique** logic option. The **Perform WYSIWYG Primitive Resynthesis** logic option is useful for resynthesizing some or all of the WYSIWYG primitives in your design for better area or performance. However, WYSIWYG primitive resynthesis can be done only when you use third-party synthesis tools.



The **Balanced** setting typically produces utilization results that are very similar to the **Area** setting with better performance results. The **Area** setting can give better results in some cases. Performing WYSIWYG resynthesis for area in this way typically reduces register-to-register timing performance.



For information about this logic option, refer to *Perform WYSIWYG Primitive Resynthesis logic option* in Quartus II Help.

Use Register Packing

The **Auto Packed Registers** option implements the functions of two cells into one logic cell by combining the register of one cell in which only the register is used with the LUT of another cell in which only the LUT is used.



For more information, refer to *Auto Packed Registers logic option* in Quartus Help.

Remove Fitter Constraints

A design with conflicting constraints or constraints that are difficult to meet may not fit in the targeted device. For example, a design might fail to fit if the location or LogicLock assignments are too strict and not enough routing resources are available on the device.

To resolve routing congestion caused by restrictive location constraints or LogicLock region assignments, use the **Routing Congestion** task in the Chip Planner to locate routing problems in the floorplan, then remove any internal location or LogicLock region assignments in that area. If your design still does not fit, the design is over-constrained. To correct the problem, remove all location and LogicLock assignments and run successive compilations, incrementally constraining the design before each compilation. You can delete specific location assignments in the Assignment Editor or the Chip Planner. To remove LogicLock assignments in the Chip Planner, in the LogicLock Regions Window, or on the Assignments menu, click **Remove Assignments**. Turn on the assignment categories you want to remove from the design in the **Available assignment categories** list.



For more information about the **Routing Congestion** task in the Chip Planner, refer to *Analyzing and Optimizing the Design Floorplan with the Chip Planner* of the *Quartus II Handbook*.

Flatten the Hierarchy During Synthesis

Synthesis tools typically provide the option of preserving hierarchical boundaries, which can be useful for verification or other purposes. However, the Quartus II software optimizes across hierarchical boundaries so as to perform the most logic minimization, which can reduce area in a design with no design partitions.

If you are using Quartus II incremental compilation, you cannot flatten your design across design partitions. Incremental compilation always preserves the hierarchical boundaries between design partitions, and the synthesis does not flatten it across partitions. Follow Altera's recommendations for design partitioning, such as registering partition boundaries to reduce the effect of cross-boundary optimizations.



For more information about using incremental compilation and recommendations for design partitioning, refer to the *Quartus II Incremental Compilation for Hierarchical and Team-Based Design* and *Best Practices for Incremental Compilation Partitions and Floorplan Assignments* chapters in volume 1 of the *Quartus II Handbook*.

Retarget Memory Blocks

If your design fails to fit because it runs out of device memory resources, your design may require a certain type of memory that the device does not have. For example, a design that requires two M-RAM blocks cannot be targeted to a Stratix® EP1S10 device, which has only one M-RAM block. You might be able to obtain a fit by building one of the memories with a different size memory block, such as an M4K memory block.

If the memory block was created with the MegaWizard™ Plug-In Manager, open the MegaWizard Plug-In Manager and edit the RAM block type so it targets a new memory block size.

ROM and RAM memory blocks can also be inferred from your HDL code, and your synthesis software can place large shift registers into memory blocks by inferring the ALTSHIFT_TAPS megafunction. This inference can be turned off in your synthesis tool to cause the memory or shift registers to be placed in logic instead of in memory blocks. Also, for improved timing performance, you can turn this inference off to prevent registers from being moved into RAM.

- ② For more information, refer to *Auto RAM Replacement logic option*, *Auto ROM Replacement logic option*, and *Auto Shift Register Replacement logic option* in Quartus II Help.

Depending on your synthesis tool, you can also set the RAM block type for inferred memory blocks. In Quartus II integrated synthesis, set the **ramstyle** attribute to the desired memory type for the inferred RAM blocks, or set the option to **logic**, to implement the memory block in standard logic instead of a memory block.

Consider the Resource Utilization by Entity report in the report file and determine whether there is an unusually high register count in any of the modules. Some coding styles can prevent the Quartus II software from inferring RAM blocks from the source code because of the blocks' architectural implementation, and force the software to implement the logic in flipflops. As an example, a function such as an asynchronous reset on a register bank might make the register bank incompatible with the RAM blocks in the device architecture, so that the register bank is implemented in flipflops. It is often possible to move a large register bank into RAM by slight modification of associated logic.



For more information about memory inference control in other synthesis tools, refer to the appropriate chapter in *Synthesis* in volume 1 of the *Quartus II Handbook*, or your synthesis software's documentation. For more information about coding styles and HDL examples that ensure memory inference, refer to the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*.

Use Physical Synthesis Options to Reduce Area

The physical synthesis options for fitting help you decrease resource usage. When you enable these options, the Quartus II software makes placement-specific changes to the netlist that reduce resource utilization for a specific Altera device.



The compilation time might increase considerably when you use physical synthesis options.

With the Quartus II software, you can apply physical synthesis options to specific instances, which can reduce the impact on compilation time. Physical synthesis instance assignments allow you to enable physical synthesis algorithms for specific portions of your design.

The following physical synthesis optimizations for fitting are available:

- Physical synthesis for combinational logic
- Map logic into memory



For more information, refer to *Physical Synthesis Optimizations Page (Settings Dialog Box)* in Quartus II Help.

Retarget or Balance DSP Blocks

A design might not fit because it requires too many DSP blocks. You can implement all DSP block functions with logic cells, so you can retarget some of the DSP blocks to logic to obtain a fit.

If the DSP function was created with the MegaWizard Plug-In Manager, open the MegaWizard Plug-In Manager and edit the function so it targets logic cells instead of DSP blocks. The Quartus II software uses the `DEDICATED_MULTIPLIER_CIRCUITRY` megafunction parameter to control the implementation.

DSP blocks also can be inferred from your HDL code for multipliers, multiply-adders, and multiply-accumulators. You can turn off this inference in your synthesis tool. When you are using Quartus II integrated synthesis, you can disable inference by turning off the **Auto DSP Block Replacement** logic option for your entire project. On the Assignments menu, click **Settings**. In the **Category** list, select **Analysis & Synthesis Settings**, click **More Settings**, and turn off **Auto DSP Block Replacement**. Alternatively, you can disable the option for a specific block with the Assignment Editor.



For more information about disabling DSP block inference in other synthesis tools, refer to the appropriate chapter in *Synthesis* in volume 1 of the *Quartus II Handbook*, or your synthesis software's documentation.

The Quartus II software also offers the **DSP Block Balancing** logic option, which implements DSP block elements in logic cells or in different DSP block modes. The default **Auto** setting allows DSP block balancing to convert the DSP block slices automatically as appropriate to minimize the area and maximize the speed of the design. You can use other settings for a specific node or entity, or on a project-wide basis, to control how the Quartus II software converts DSP functions into logic cells and DSP blocks. Using any value other than **Auto** or **Off** overrides the `DEDICATED_MULTIPLIER_CIRCUITRY` parameter used in megafunction variations.

- For more details about the Quartus II logic options described in this section, refer to *Auto DSP Block Replacement logic option* and *DSP Block Balancing logic option* in Quartus II Help.

Use a Larger Device

If a successful fit cannot be achieved because of a shortage of LEs, ALMs, memory, or DSP blocks, you might require a larger device.

Routing

Use the suggestions in the following sections to help you resolve routing resource problems.

Set Auto Packed Registers to Sparse or Sparse Auto

The **Auto Packed Registers** option reduces LE or ALM count in a design. You can set this option in the Assignment Editor or by clicking **More Settings** on the **Fitter Settings** page in the **Settings** dialog box.

- For more information, refer to *Auto Packed Registers logic option* in Quartus II Help.

Set Fitter Aggressive Routability Optimizations to Always

The **Fitter Aggressive Routability Optimization** option is useful if your design does not fit due to excessive routing wire utilization.

If there is a significant imbalance between placement and routing time (during the first fitting attempt), it might be because of high wire utilization. Turning on the **Fitter Aggressive Routability Optimizations** option can reduce your compilation time.

On average, this option can save up to 6% wire utilization, but can also reduce performance by up to 4%, depending on the device.

- For more information, refer to *Fitter Aggressive Routability Optimizations logic option* in Quartus II Help.

Increase Router Effort Multiplier

The Router Effort Multiplier controls how quickly the router tries to find a valid solution. The default value is 1.0 and legal values must be greater than 0. Numbers higher than 1 help designs that are difficult to route by increasing the routing effort. Numbers closer to 0 (for example, 0.1) can reduce router runtime, but usually reduce routing quality slightly. Experimental evidence shows that a multiplier of 3.0 reduces overall wire usage by approximately 2%. Using a Router Effort Multiplier higher than the default value could be beneficial for designs with complex datapaths with more than five levels of logic. However, congestion in a design is primarily due to placement, and increasing the Router Effort Multiplier does not necessarily reduce congestion.

- Any Router Effort Multiplier value greater than 4 only increases by 10% for every additional 1. For example, a value of 10 is actually 4.6.

- For more information, refer to *Router Effort Multiplier logic option* in Quartus II Help.

Remove Fitter Constraints

A design with conflicting constraints or constraints that are difficult to achieve may not fit the targeted device. Conflicting or difficult-to-achieve constraints can occur when location or LogicLock assignments are too strict and there are not enough routing resources.

In this case, use the **Routing Congestion** task in the Chip Planner to locate routing problems in the floorplan, then remove all location and LogicLock region assignments from that area. If the local constraints are removed, and the design still does not fit, the design is over-constrained. To correct the problem, remove all location and LogicLock assignments and run successive compilations, incrementally constraining the design before each compilation. You can delete specific location assignments in the Assignment Editor or the Chip Planner. To remove LogicLock assignments in the Chip Planner, in the LogicLock Regions Window, or on the Assignments menu, click **Remove Assignments**. Turn on the assignment categories you want to remove from the design in the **Available assignment categories** list.


 For more information about the **Routing Congestion** task in the Chip Planner, refer to the *Analyzing and Optimizing the Design Floorplan with the Chip Planner* chapter in volume 2 of the *Quartus II Handbook*. You can also refer to *About the Chip Planner* in Quartus II Help.

Optimize Synthesis for Area, Not Speed


In some cases, resynthesizing the design to improve the area utilization can also improve the routability of the design. First, ensure that you have set your device and timing constraints correctly in your synthesis tool. Ensure that you do not overconstrain the timing requirements for the design, particularly when the area utilization of the design is a concern. Synthesis tools generally try to meet the specified requirements, which can result in higher device resource usage if the constraints are too aggressive.

If resource utilization is important to improve the routing results in your design, some synthesis tools offer an easy way to optimize for area instead of speed. If you are using Quartus II integrated synthesis, on the Assignments menu, click **Settings**. In the **Category** list, select **Analysis & Synthesis Settings** and select **Balanced** or **Area** under **Optimization Technique**.

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). You can apply the setting to specific modules while leaving the default **Optimization Technique** setting at **Balanced** (for the best trade-off between area and speed for certain device families) or **Speed**. You can also use the **Speed Optimization Technique for Clock Domains** logic option to specify that all combinational logic in or between the specified clock domain(s) is optimized for speed.

 In the Quartus II software, the **Balanced** setting typically produces utilization results that are very similar to those obtained with the **Area** setting, with better performance results. The **Area** setting can yield better results in some unusual cases.

In some synthesis tools, not specifying an f_{MAX} requirement can result in less resource utilization, which can improve routability.

 For information about setting the timing requirements and synthesis options in Quartus II integrated synthesis and other synthesis tools, refer to the appropriate chapter in *Synthesis* in volume 1 of the *Quartus II Handbook*, or your synthesis software's documentation.

Optimize Source Code

If your design does not fit because of routing problems and the methods described in the preceding sections do not sufficiently improve the routability of the design, modify the design at the source to achieve the desired results. You can often improve results significantly by making design-specific changes to your source code, such as duplicating logic or changing the connections between blocks that require significant routing resources.

Use a Larger Device

If a successful fit cannot be achieved because of a shortage of routing resources, you might require a larger device.

Optimizing Resource Utilization (Macrocell-Based CPLDs)

The following recommendations help you take advantage of the macrocell-based architecture in the MAX 7000 and MAX 3000 devices to yield maximum speed, reliability, and device resource utilization while minimizing fitting difficulties.

After design analysis, the first stage of design optimization is to improve resource utilization. Complete this stage before proceeding to timing optimization. First, ensure that you have set the basic constraints described in “Initial Compilation: Required Settings” section in the *Design Optimization Overview* chapter of the *Quartus II Handbook*. If your design is not fitting into a specified device, use the techniques in this section to achieve a successful fit.

Use Dedicated Inputs for Global Control Signals

MAX 7000 and MAX 3000 devices have four dedicated inputs that you can use for global register control. Because the global register control signals can bypass the logic cell array and directly feed registers, product terms can be preserved for primary logic. Also, because each signal has a dedicated path into the LAB, global signals also can bypass logic and data path interconnect resources.

Because the dedicated input pins are designed for high fan-out control signals and provide low skew, always assign global signals (such as clock, clear, and output enable) to the dedicated input pins.

You can use logic-generated control signals for global control signals instead of dedicated inputs. However, the following list shows the disadvantages of using logic-generated control signals:

- More resources are required (logic cells, interconnect).
- More data skew is introduced.
- If the logic-generated control signals have high fan-out, the design can be more difficult to fit.

By default, the Quartus II software uses dedicated inputs for global control signals automatically. You can assign control signals to dedicated input pins in one of the following ways:

- In the Assignment Editor, select one of the two following methods:
 - Assign pins to dedicated pin locations.
 - Assign a **Global Signal** setting to the pins.
- On the Assignments menu, click **Settings**. On the **Analysis & Synthesis Settings** page, click **More Settings**, and in the **Existing Option settings** section, select **Auto Global Register Control Signals**.
- Insert a GLOBAL primitive after the pins.

Reserve Device Resources

Because pin and logic option assignments can be necessary for board layout and performance requirements, and because full utilization of the device resources can increase the difficulty of fitting the design, Altera recommends leaving 10% of the logic cells and 5% of the I/O pins unused to accommodate future design modifications. Following the Altera-recommended device resource reservation guidelines for macrocell-based CPLDs increases the chance that the Quartus II software can fit the design during recompilation after changes or assignments have been made.

Pin Assignment Guidelines and Procedures

Sometimes user-specified pin assignments are necessary for board layout. This section describes pin assignment guidelines and procedures.

To minimize fitting issues with pin assignments, follow these guidelines:

- Assign speed-critical control signals to dedicated inputs.
- Assign output enables (OE) pin to appropriate locations.
- Estimate fan-in to assign output pins to the appropriate LAB.
- Assign output pins that require parallel expanders to macrocells numbered 4 to 16.



Altera recommends allowing the Quartus II software to select pin assignments automatically when possible. You can use the Quartus II Pin Advisor feature (accessible from the Tools menu) for pin connection guidelines.



For more information about the Pin Advisor, refer to *Pin Advisor Command (Tools Menu)* in Quartus II Help.

Control Signal Pin Assignments

Assign speed-critical control signals to dedicated input pins. Every MAX 7000 and MAX 3000 device has four dedicated input pins (GCLK1, OE2/GCLK2, OE1, and GCLRn). You can assign clocks to global clock dedicated inputs (GCLK1 and OE2/GCLK2), assign clear signals to the global clear dedicated input (GCLRn), and speed-critical OE signals to global OE dedicated inputs (OE1 and OE2/GCLK2).

Output Enable Pin Assignments

Occasionally, because the total number of required output enable pins is more than the dedicated input pins, output enable signals must be assigned to I/O pins.

To minimize possible fitting errors when assigning the output enable pins for MAX 7000 and MAX 3000 devices, refer to [Pin-Out Files for Altera Devices](http://www.altera.com) on the Altera website (www.altera.com).

Estimate Fan-In When Assigning Output Pins

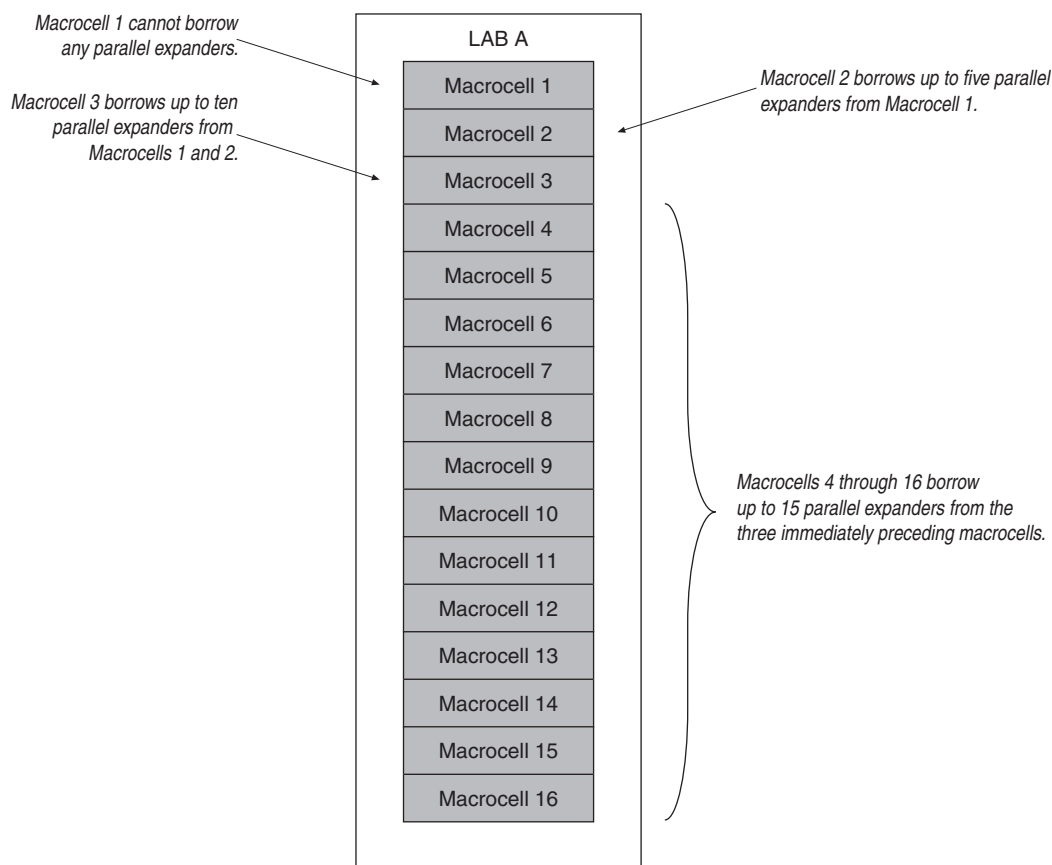
Macrocells with high fan-in can cause more placement problems for the Quartus II Fitter than those with low fan-in. The maximum fan-in per LAB should not exceed 36 in MAX 7000 and MAX 3000 devices. Therefore, estimate the fan-in of logic (such as an x -input AND gate) that feeds each output pin. If the total fan-in of logic that feeds each output pin in the same LAB exceeds 36, compilation can fail. To save resources and prevent compilation errors, avoid assigning pins that have high fan-in.

Outputs Using Parallel Expander Pin Assignments

Figure 14-1 shows how parallel expanders are used within a LAB. MAX 7000 and MAX 3000 devices contain chains that can lend or borrow parallel expanders. The Quartus II Fitter places macrocells in a location that allows them to lend and borrow parallel expanders appropriately.

As shown in Figure 14-1, only macrocells 2 through 16 can borrow parallel expanders. Therefore, assign output pins that might require parallel expanders to pins adjacent to macrocells 4 through 16. Altera recommends using macrocells 4 through 16 because they can borrow the largest number of parallel expanders.

Figure 14-1. LAB Macrocells and Parallel Expander Associations



Resolving Resource Utilization Problems

Excessive macrocell usage and lack of routing resources can cause resource utilization problems. Macrocell usage errors occur when the total number of macrocells in the design exceed the available macrocells in the device. Routing errors occur when the available routing resources are insufficient to implement the design. Check the Message window for the compilation results.



Messages in the Messages window are also copied in the Report Files. For more information about the message, right-click a message and click **Help**.

Resolving Macrocell Usage Issues

Occasionally, a design requires more macrocell resources than are available in the selected device, which results in the design not fitting. The following list provides tips for resolving macrocell usage issues as well as tips to minimize the number of macrocells used:

- On the Assignments menu, click **Settings**. In the **Category** list, select **Analysis & Synthesis Settings**, click **More Settings**, and turn off **Auto Parallel Expanders**. If the design's clock frequency (f_{MAX}) is not an important design requirement, turn off parallel expanders for all or part of the project. The design usually requires more macrocells if parallel expanders are turned on.
- Change **Optimization Technique** from **Speed** to **Area**. Selecting **Area** instructs the compiler to give preference to area utilization rather than speed (f_{MAX}). On the Assignments menu, click **Settings**. In the **Category** list, change the **Optimization Technique** option in the **Analysis & Synthesis Settings** page.
- Use D-type flipflops instead of latches. Altera recommends always using D-type flipflops instead of latches in your design because D-type flipflops can reduce the macrocell fan-in, and thus reduce macrocell usage. The Quartus II software uses extra logic to implement latches in MAX 7000 and MAX 3000 designs because MAX 7000 and MAX 3000 macrocells contain D-type flipflops instead of latches.
- Use asynchronous clear and preset instead of synchronous clear and preset. To reduce product term usage, use asynchronous clear and preset in your design whenever possible. Using other control signals such as synchronous clear produces macrocells and pins with higher fan-out.



After following the suggestions in this section, if your project still does not fit the targeted device, consider using a larger device. When upgrading to a different density, the vertical package-migration feature of the MAX 7000 and MAX 3000 device families allows pin assignments to be maintained.

Resolving Routing Issues

Routing is another resource that can cause design fitting issues. For example, if the total fan-in into a LAB exceeds the maximum allowed, a no-fit error can occur during compilation. If your design does not fit the targeted device because of routing issues, consider the following suggestions:

- Use dedicated inputs/global signals for high fan-out signals. The dedicated inputs in MAX 7000 and MAX 3000 devices are designed for speed-critical and high fan-out signals. Always assign high fan-out signals to dedicated inputs/global signals.
- Change the **Optimization Technique** option from **Speed** to **Area**. This option can resolve routing resource and macrocell usage issues. Refer to [“Resolving Macrocell Usage Issues” on page 14-15](#).
- Reduce the fan-in per cell. If you are not limited by the number of macrocells used in the design, you can use the **Fan-in per cell (%)** option to reduce the fan-in per cell. The allowable values are 20–100%; the default value is 100%. Reducing fan-in can reduce localized routing congestion but increase the macrocell count. You can set this logic option in the Assignment Editor or under **More Settings** in the **Analysis & Synthesis Settings** page of the **Settings** dialog box.
- On the Assignments menu, click **Settings**. In the **Category** list, select **Analysis & Synthesis Settings**, click **More Options**, and turn off **Auto Parallel Expanders**. By turning off the parallel expanders, you give the Quartus II software more fitting flexibility for each macrocell, allowing macrocells to be relocated. For example, you can move each macrocell (previously grouped together in the same LAB) to a different LAB to reduce routing constraints.
- Insert logic cells. Inserting logic cells reduces fan-in and shared expanders used per macrocell, increasing routability. By default, the Quartus II software automatically inserts logic cells when necessary. Otherwise, you can disable **Auto Logic Cell** as follows:
 1. On the Assignments menu, click **Settings**.
 2. In the **Category** list, select **Analysis & Synthesis Settings**.
 3. Under **More Settings**, turn off **Auto Logic Cell Insertion**. For more information, refer to [“Using LCELL Buffers to Reduce Required Resources”](#).
- Change pin assignments. If you want to discard your pin assignments, you can let the Quartus II Fitter ignore some or all of the assignments.



If you prefer reassigning pins to increase routing efficiency, refer to [“Pin Assignment Guidelines and Procedures” on page 14-13](#).

Using LCELL Buffers to Reduce Required Resources

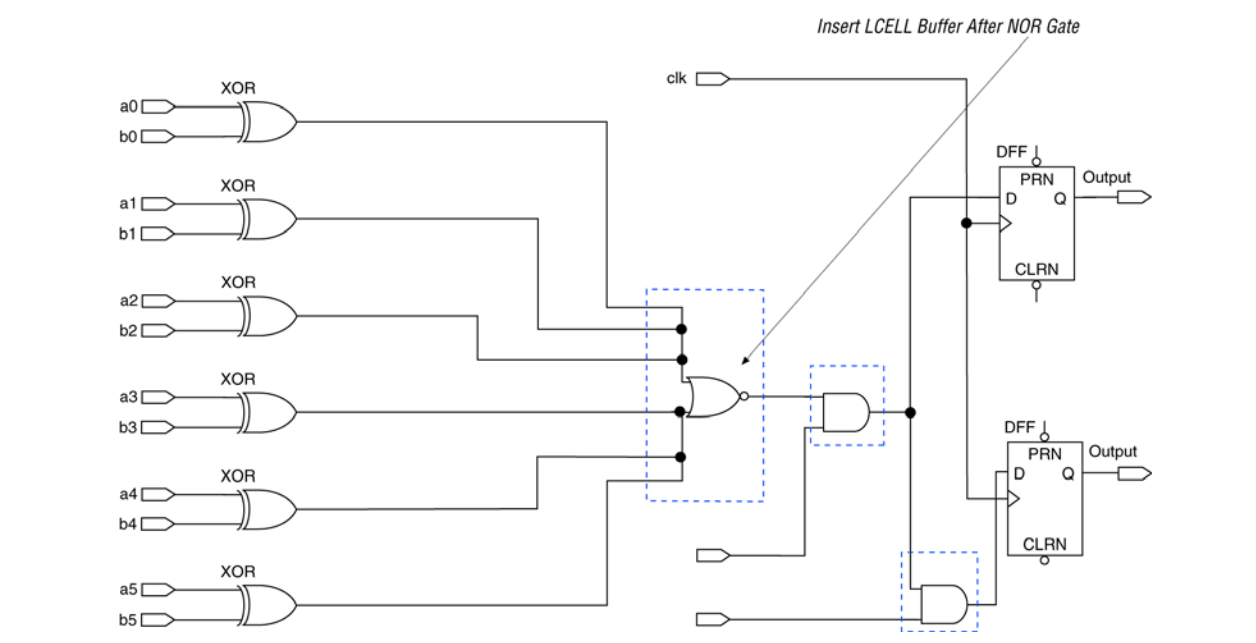
Complex logic, such as multilevel XOR gates, are often implemented with more than one macrocell. When this occurs, the Quartus II software automatically allocates shareable expanders—or additional macrocells (called synthesized logic cells)—to supplement the logic resources that are available in a single macrocell. You can also break down complex logic by inserting logic cells in the project to reduce the average fan-in and the total number of shareable expanders required. Manually inserting logic cells can provide greater control over speed-critical paths.

Instead of using the **Auto Logic Cell Insertion** option, you can manually insert logic cells. However, Altera recommends using the **Auto Logic Cell Insertion** option unless you know which part of the design is causing the congestion.

A good location to manually insert LCELL buffers is where a single complex logic expression feeds multiple destinations in your design. You can insert an LCELL buffer just after the complex expression; the Fitter extracts this complex expression and places it in a separate logic cell. Rather than duplicate all the logic for each destination, the Quartus II software feeds the single output from the logic cell to all destinations.

To reduce fan-in and prevent no-fit compilations caused by routing resource issues, insert an LCELL buffer after a NOR gate (Figure 14-2). The design in Figure 14-2 was compiled for a MAX 7000AE device. Without the LCELL buffer, the design requires two macrocells and eight shareable expanders, and the average fan-in is 14.5 macrocells. However, with the LCELL buffer, the design requires three macrocells and eight shareable expanders, and the average fan-in is just 6.33 macrocells.

Figure 14-2. Reducing the Average Fan-In by Inserting LCELL Buffers



Scripting Support

You can run procedures and make settings described in this chapter in a Tcl script. You can also run some procedures at a command prompt. For detailed information about scripting command options, refer to the Quartus II command-line and Tcl API Help browser. To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp ←
```



For more information about Tcl scripting, refer to the *Tcl Scripting* chapter in volume 2 of the *Quartus II Handbook*. For more information about all settings and constraints in the Quartus II software, refer to the *Quartus II Settings File Manual*. For more information about command-line scripting, refer to the *Command-Line Scripting* chapter in volume 2 of the *Quartus II Handbook*.

You can specify many of the options described in this section either in an instance, or at a global level, or both.

Use the following Tcl command to make a global assignment:

```
set_global_assignment -name <.qsf variable name> <value> ←
```

Use the following Tcl command to make an instance assignment:

```
set_instance_assignment -name <.qsf variable name> <value> \  
-to <instance name> ←
```



If the <value> field includes spaces (for example, 'Standard Fit'), you must enclose the value in straight double quotation marks.

Initial Compilation Settings

Use the Quartus II Settings File (.qsf) variable name in the Tcl assignment to make the setting along with the appropriate value. The **Type** column indicates whether the setting is supported as a global setting, an instance setting, or both.

Table 14-1 lists the advanced compilation settings.

Table 14-1. Advanced Compilation Settings

Setting Name	.qsf File Variable Name	Values	Type
Placement Effort Multiplier	PLACEMENT_EFFORT_MULTIPLIER	Any positive, non-zero value	Global
Router Effort Multiplier	ROUTER_EFFORT_MULTIPLIER	Any positive, non-zero value	Global
Router Timing Optimization level	ROUTER_TIMING_OPTIMIZATION_LEVEL	NORMAL, MINIMUM, MAXIMUM	Global
Final Placement Optimization	FINAL_PLACEMENT_OPTIMIZATION	ALWAYS, AUTOMATICALLY, NEVER	Global

Resource Utilization Optimization Techniques (LUT-Based Devices)

Table 14-2 lists the .qsf file variable name and applicable values for the settings described in “Optimizing Resource Utilization (LUT-Based Devices)” on page 14-2.

Table 14-2. Resource Utilization Optimization Settings (Part 1 of 2)

Setting Name	.qsf File Variable Name	Values	Type
Auto Packed Registers ⁽¹⁾	AUTO_PACKED_REGISTERS_<device family name>	OFF, NORMAL, MINIMIZE AREA, MINIMIZE AREA WITH CHAINS, AUTO	Global, Instance
Perform WYSIWYG Primitive Resynthesis	ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP	ON, OFF	Global, Instance

Table 14-2. Resource Utilization Optimization Settings (Part 2 of 2)

Setting Name	.qsf File Variable Name	Values	Type
Physical Synthesis for Combinational Logic for Reducing Area	PHYSICAL_SYNTHESIS_COMBO_LOGIC_FOR_AREA	ON, OFF	Global, Instance
Physical Synthesis for Mapping Logic to Memory	PHYSICAL_SYNTHESIS_MAP_LOGIC_TO_MEMORY_FOR_AREA	ON, OFF	Global, Instance
Optimization Technique	<device family name>_OPTIMIZATION_TECHNIQUE	AREA, SPEED, BALANCED	Global, Instance
Speed Optimization Technique for Clock Domains	SYNTH_CRITICAL_CLOCK	ON, OFF	Instance
State Machine Encoding	STATE_MACHINE_PROCESSING	AUTO, ONE-HOT, GRAY, JOHNSON, MINIMAL BITS, ONE-HOT, SEQUENTIAL, USER-ENCODE	Global, Instance
Auto RAM Replacement	AUTO_RAM_RECOGNITION	ON, OFF	Global, Instance
Auto ROM Replacement	AUTO_ROM_RECOGNITION	ON, OFF	Global, Instance
Auto Shift Register Replacement	AUTO_SHIFT_REGISTER_RECOGNITION	ON, OFF	Global, Instance
Auto Block Replacement	AUTO_DSP_RECOGNITION	ON, OFF	Global, Instance
Number of Processors for Parallel Compilation	NUM_PARALLEL_PROCESSORS	Integer between 1 and 16 inclusive, or ALL	Global

Note to Table 14-2:

- (1) Allowed values for this setting depend on the device family that you select.

Document Revision History

Table 14-3 lists the revision history for this document.

Table 14-3. Document Revision History

Date	Version	Changes
May 2013	1.0	Initial release.

