Stratix III Design Guidelines

May 2008, version 1.1 Application Note 469

Introduction

Stratix® III devices are engineered for high-speed core performance and high-speed I/O with the best signal integrity in the industry, combined with low-static and dynamic-power consumption. The devices also offer increased logic density, so you can integrate more of your product to reduce cost and board space.

It is important to follow Altera recommendations throughout the design process for high-density, high-performance Stratix III designs. Planning the FPGA and system early in the design process is crucial to your success. This document provides an easy-to-use set of guidelines and a list of factors to consider in Stratix III designs, but does not include all the details about the product. It includes pointers to other documentation where you can find detailed specifications, device feature descriptions, and additional guidelines. The material covers the Stratix III device architecture as well as aspects of the Quartus® II software and third-party tools that you might use in your design.

The guidelines presented in this document will help you improve productivity and avoid common design pitfalls. The document discusses various stages of the design flow in the order that each stage is typically performed, as shown in Table 1. You can use the "Design Checklist," on page 65 to help verify that you have followed each of the guidelines.

Table 1. Summary of Design Flow Stages and Guideline Topics (Part 1 of 2)

Stages of Design Flow	Guideline Topics
"Device Selection," on page 2	Device information, determining device density, package offerings, speed grade, core voltage, migration, and HardCopy® ASICs
"Planning for Device Configuration," on page 6	Configuration scheme overview, configuration features, Quartus II settings, optional pins
"Early System Planning," on page 12	Planning: design specifications, IP selection, on-chip debugging, early power estimation
"Board Design Considerations," on page 18	Power-up, power pins, configuration pins, signal integrity, board-level verification
"I/O and Clock Planning," on page 27	Pin assignments, early pin planning, I/O features and connections, clock and PLL selection, SSN
"Design and Compilation," on page 42	Synthesis tools, coding styles and recommendations, planning for hierarchical or team-based design, SOPC Builder

Table 1. Summary of Design Flow Stages and Guideline Topics (Part 2 of 2)	
Stages of Design Flow	**Guideline Topics**
"Timing Closure and Verification," on page 50	Device utilization, timing constraints and analysis, area and timing optimization, verification
"Power Analysis and Optimization," on page 57	Analysis tools, optimization techniques, thermal management options

For complete details about the Stratix III device architecture, refer to the Stratix III Literature page. For the latest known issues related to Stratix III FPGAs, refer to the *Stratix III Device Family Errata Sheet* and the Knowledge Database.

Device Selection

The first step in the Stratix III design process is to choose the device family variant, device density, speed grade, package, and core voltage that best suit your design needs. You should also consider whether you want to target FPGA or ASIC migration devices. Before you begin compiling a design in a third-party synthesis tool or the Quartus II software, set the correct target device.

For information about the features available in each device density, including logic, memory blocks, multipliers, and phase-locked loops (PLLs), as well the various package offerings and I/O pin counts, refer to the *Stratix III Device Family Overview* chapter in volume 1 of the *Stratix III Device Handbook*.

Logic, Memory, and Multiplier Density

The Stratix III logic family (L) offers balanced logic, memory, and multipliers to address a wide range of applications, while the enhanced family (E) offers more memory and multipliers per logic and is ideal for wireless, medical imaging, and military applications. Choose the device family variant with the best resource balance for your design requirements.

Stratix III devices offer a range of densities that provide different amounts of device logic resources, including memory, multipliers, and adaptive logic module (ALM) logic cells. Different device densities may also offer different numbers of features such as PLLs. Determining the required logic density can be a challenging part of the design planning process. Devices with more logic resources can implement larger and potentially more complex designs, but generally have a higher cost. Smaller devices have lower static power utilization. Select a device that meets your design needs with some safety margin, in case you want to add more logic later in the design cycle, or to upgrade or expand your design. You might also

want additional space in the device to ease creating a design floorplan for incremental or team-based design, as described in "Planning for Hierarchical and Team-Based Design," on page 47. Consider reserving resources for debugging, as described in "Planning for On-Chip Debugging," on page 13. Stratix III devices support vertical migration between certain device densities, which provides flexibility as described in "Vertical Device Migration," on page 3.

Many next-generation designs use a current design as a starting point. If you have other designs that target an Altera device, you can use their resource utilization as an estimate for your new design. Compile existing designs in the Quartus II software with the **Auto device selected by the Fitter** option in the **Settings** dialog box. Review the resource utilization to find out which device density fits the design. Keep in mind that coding style, device architecture, and the optimization options used in the Quartus II software can significantly affect a design's resource utilization as well as timing performance. Refer to "Device Resource Utilization Reports," on page 51, for more information about determining resource utilization.

To obtain resource utilization estimates for certain configurations of Altera's intellectual property (IP) designs, refer to the user guides for Altera® megafunctions and IP MegaCores on the IP Megafunctions page in the Literature section at www.altera.com. You can use these numbers to help estimate the resource utilization of your design.

I/O Pin Count and Package Offering

Stratix III devices are available in space-saving FineLine® BGA packages with various I/O pin counts. Certain package sizes support vertical migration within different device densities, as described in "Vertical Device Migration". Determine the required number of I/O pins, considering the design's interface requirements with other system blocks. You can compile any existing designs in the Quartus II software with the **Auto device selected by the Fitter** option in the **Settings** dialog box to determine how many input and output pins are used. Also consider reserving pins for debugging, as described in "Planning for On-Chip Debugging," on page 13.

Vertical Device Migration

Stratix III devices support vertical migration within the same package, which enables you to migrate to different density devices whose dedicated pins, configuration pins, and power pins are the same for a given package. This feature allows future upgrades or changes to your design without any changes to the board layout, because you can replace the Stratix III device on the board with a different density Stratix III

device. You can migrate from the L logic family to the E enhanced family without increasing the amount of logic available, which minimizes the cost of vertical migration.

Determine whether you want the option of migrating your design to another device density. Choose your device density and package to accommodate any possible future device migration to allow flexibility when the design nears completion. You should specify any potential migration options in the Quartus II software at the beginning of your design cycle. Selecting a migration device impacts the design's pin placement to ensure that your design is compatible with the selected device(s). It is possible to add migration devices later in the design cycle, but it requires extra effort to check pin assignments, and can require design or board layout changes to fit into the new target device. It is easier to consider these issues early in the design cycle than at the end, when the design is near completion and ready for migration. To specify the target migration devices, on the Assignments menu, click **Device** and in the **Settings** dialog box, select **Migration Devices**.

As described in "Making FPGA Pin Assignments," on page 28, the **Pin Migration** view in the Quartus II Pin Planner highlights pins that change function in the migration device when compared to the currently selected device.

HardCopy III ASIC Migration

The HardCopy methodology allows you to seamlessly prototype your system with Stratix III FPGAs and completely prepare your system for production, prior to ASIC design handoff. Altera's HardCopy Design Center uses a proven turnkey process to implement the low-cost, low-power, functionally-equivalent, pin-compatible HardCopy III ASIC. The result is a drop-in replacement to the prototyping Stratix III FPGA on your system board.

You can start your HardCopy III ASIC design by targeting your design to an appropriate Stratix III FPGA and choosing the appropriate HardCopy III ASIC companion in the latest version of the Quartus II software.

You can migrate between the FPGA and ASIC revisions of your project when the Quartus II software includes these devices.

If you use a HardCopy device, review the HardCopy guidelines early in the design cycle for any Quartus II software settings that should use or other restrictions you should consider as you complete your design. For example:

- HardCopy III devices use a 0.9-V core voltage (V_{CCL}). So if you use a 1.1-V Stratix III device, your board design must accommodate a core voltage change between devices.
- HardCopy devices do not support a 3.3-V V_{CCIO}, so you should use 3.0-V or less in the Stratix III device as well.
- RAM cannot be initialized to a known value in the HardCopy ASIC like in an FPGA. Therefore, if you plan to migrate to a HardCopy device, your design must write RAM contents during device operation instead of relying on memory initialization values.
- It is especially important to use complete timing constraints if you want to migrate to a HardCopy device because of the rigorous verification requirements for ASICs.
- Altera recommends adding the PLL reconfiguration megafunction to your design so that you can change the PLL settings in-system on the HardCopy III device.

For more information, refer to the *Phase-Locked Loops Reconfiguration Megafunction User Guide (ALTPLL_RECONFIG)*.

In addition, review the HardCopy Readiness Report that is generated during compilation when a HardCopy companion device is selected in the Device Settings. This advises you about any incomplete I/O assignments and provides recommendations for clock pin locations.

For more information and guidelines for HardCopy migrations, refer to the *HardCopy Series Handbook*.

Speed Grade

The device speed grade affects the device timing performance and timing closure, as well as power utilization. Stratix III devices are available in three speed grades, –2, –3, and –4 (–2 is the fastest). Generally, the faster devices have higher cost. As one way to help you determine which speed grade your design requires, refer to the supported clock rates for specific I/O interfaces.

For information about supported clock rates for memory interfaces using I/O pins on different sides of the device in different device speed grades, refer to the *External Memory Interfaces in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*.

Some designers use the fastest speed grade during prototyping to reduce compilation time (because less time is spent optimizing the design to meet timing requirements), and then move to a slower speed grade for production to reduce cost if the design meets its timing requirements.

The Quartus II software optimizes and analyzes your design using different timing models for each speed grade. If you migrate to a device with a different speed grade, you must perform timing analysis to ensure that there are no timing violations resulting from changes in timing performance.

Selectable Core Voltage

Selectable core voltage gives you the option of using a 0.9-V core voltage which increases the static and dynamic power savings compared to the standard 1.1-V core voltage. Designs requiring the highest performance should use the 1.1-V core voltage, while designs requiring minimum power consumption can use the 0.9-V core voltage. Choose the core voltage along with the device and speed grade in the Quartus II software, because the selectable core voltage affects the Stratix III fabric performance. The Quartus II software uses timing and power models specific to the selected core voltage to implement all timing-dependent and power-dependent analysis, and optimization.

If you use the low-power voltage setting, ensure that the 0.9-V option is available for your device density, package, and speed grade combination. There is limited speed grade support with the 0.9-V option.

Planning for Device Configuration

Stratix III devices are based on SRAM cells, and you must download configuration data to the Stratix III device each time the device powers up because SRAM memory is volatile. Choosing your device configuration method early allows system and board designers to determine what companion devices, if any, are needed for your system. Your board layout also depends on the configuration method you plan to use for the programmable device, because different schemes require different connections. For board design guidelines related to configuration pins, refer to "Board Design Considerations," on page 18.

In addition, Stratix III devices offer configuration data decompression to save configuration memory space and time, design security using data encryption to protect your designs, and real-time remote system upgrades. Support for these configuration features varies depending on your configuration scheme. Stratix III devices also include optional configuration pins and a reconfiguration option that you should choose up front and set up in the Quartus II software, so that you have all the information required for your board and system design.

This section includes the following topics:

For more information about configuration, refer to the *Configuring Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*. For more information, refer to the Configuration Center. This web page includes links to JTAG Configuration & ISP Troubleshooter and FPGA Configuration Troubleshooter that you can use to help debug configuration problems.

Configuration Scheme Selection

You can configure Stratix III devices using one of four configuration schemes:

- Fast passive parallel (FPP)
- Fast active serial (AS)
- Passive serial (PS)
- Joint Test Action Group (JTAG)

Select the configuration scheme by driving the Stratix III device MSEL pins either high or low.

For more information about the supported configuration schemes, refer to the Stratix III Device Configuration Center. For complete information about the Stratix III supported configuration schemes, how to execute the required configuration schemes, and all of the necessary option pin settings, including the MSEL pin settings, refer to the *Configuring Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*.

All configuration schemes use a configuration device, a download cable, or an external controller (for example, a MAX® II device or microprocessor).

Configuration Devices

You can use the Altera enhanced configuration devices (EPC) in the FPP and PS configuration schemes. The Altera serial configuration devices (EPCS) are used in the Fast AS configuration scheme. Check whether the configuration device supports the configuration bitstream file size for your Stratix III device. You can also use a MAX II device or a

microprocessor with a flash memory configuration method, or use the compression feature to reduce the configuration file size of large Stratix III devices.

For information about enhanced configuration devices, refer to the *Enhanced Configuration Devices (EPC4, EPC8, and EPC16) Data Sheet* and the *Altera Enhanced Configuration Devices* chapters in volume 2 of the *Configuration Handbook*. For information about serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook*. This datasheet includes a table that lists serial configuration device support for Stratix III devices. If the uncompressed file size is too large for a specific device density, the table indicates that it includes the compression feature to reduce the file size.

Serial configuration devices do not directly support the JTAG interface; however, you can program the device with JTAG download cables using the Serial FlashLoader (SFL) feature in the Quartus II software. This feature uses the FPGA as a bridge between the JTAG interface and the configuration device, allowing both devices to use the same JTAG interface.

This solution is slower than standard AS configuration schemes because the SFL solution must configure the FPGA before programming configuration devices.

For more details about the SFL, refer to *AN 370: Using the Serial FlashLoader with the Quartus II Software*.

Download Cables

The Quartus II programmer supports configuring Stratix III devices directly using PS or JTAG interfaces via Altera programming download cables. You can download design changes directly to the device with Altera download cables, making prototyping easy and enabling you to make multiple design iterations in quick succession. You can use the same download cable to program configuration devices on the board and use JTAG debugging tools such as the SignalTap II Logic Analyzer.

For more information about how to use Altera's current download cables, refer to the following documents:

- *ByteBlaster II Download Cable User Guide*
- *USB Blaster II USB Port Download Cable User Guide*
- *EthernetBlaster Communications Cable User Guide*

MAX II Parallel Flash Loader

If your system already contains common flash interface (CFI) flash memory, you can utilize it for Stratix III device configuration storage as well. The parallel flash loader (PFL) feature with MAX II devices allow you to program CFI flash memory devices through the JTAG interface. It also provides the logic to control configuration from the flash memory device to the Stratix III device and supports compression to reduce the size of your configuration data. Both PS and FPP configuration modes are supported using this PFL feature. If you choose this configuration method, you should check the list of supported flash devices early in your system design cycle and plan accordingly.

For more information about PFL, refer to *AN 386: Using the Parallel Flash Loader with the Quartus II Software*.

Configuration Features

This section describes Stratix III configuration features and how they affect your design process: data compression, design security, and remote system upgrades.

Data Compression

When you enable data compression, the Quartus II software generates configuration files with compressed configuration data. This compressed file reduces the storage requirements in the configuration device or flash memory, and decreases the time needed to transmit the bitstream to the Stratix III device. The time required by a Stratix III device to decompress a configuration file is less than the time needed to transmit the configuration data to the device. Stratix III devices support decompression in the FPP (when using a MAX II device/microprocessor + flash), fast AS, and PS configuration schemes. The Stratix III decompression feature is not available in the FPP (when using enhanced configuration device) or JTAG configuration schemes.

For more information about data compression, refer to the *Configuring Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*.

Design Security Using Configuration Bitstream Encryption

The design security feature ensures that Stratix III designs are protected from copying, reverse engineering, and tampering using configuration bitstream encryption. The design security feature is available when configuring Stratix III devices using the fast passive parallel (FPP) configuration mode with an external host (such as a MAX II device or microprocessor), or when using fast active serial (AS) or passive serial

(PS) configuration schemes. The design security feature is also available in remote update with fast AS configuration mode. The design security feature is not available when you are configuring your Stratix III device using FPP with an enhanced configuration device or JTAG configuration schemes.

For more information about design security using configuration bitstream encryption, refer to the *Design Security in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*.

Remote System Upgrades

Stratix III devices support remote update in the AS configuration scheme. You can implement remote update in conjunction with real-time decompression of configuration data if you need to save configuration memory space in the serial configuration device with AS configuration.

To implement the remote system upgrade interface, you can use the ALTREMOTE_UPDATE megafunction or instantiate a remote system upgrade atom.

For more information about making remote system upgrades, refer to the *Remote System Upgrades with Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*. For more information about the altremote_update megafunction, refer to the *Remote Update Circuitry Megafunction User Guide (ALTREMOTE_UPDATE)*.

Quartus II Configuration Settings

This section covers several configuration options that you can set in the Quartus II software before you compile to generate configuration or programming files. Your board and system design are affected by these settings and pins.

Optional Configuration Pins

You can enable the following optional configuration pins on the **General** tab of the **Device and Pin Options** dialog box:

- `CLKUSR` pin—The **Enable user-supplied start-up clock (CLKUSR)** option enables you to select which clock source is used for initialization, either the internal oscillator or an external clock provided on the `CLKUSR` pin.

- INIT_DONE pin—To check if the device has completed initialization and is in user mode, you can monitor the INIT_DONE pin. Enable the pin with the **Enable INIT_DONE output** option. The INIT_DONE pin is an open-drain output and requires an external pull-up to V_{CCPGM}.

Restart Configuration after Error

You can enable the **Auto-restart after configuration error** option on the **General** tab of the **Device and Pin Options** dialog box. With this option, when a configuration error occurs, the device drives nSTATUS low, which resets the device internally. The device releases its nSTATUS pin after a reset time-out period. The nSTATUS pin requires an external 10-kΩ pull-up resistor to V_{CCPGM}, unless it is connected to an external configuration device which provide an internal pull-up.

Configuration and Programming Files

This section presents information related to the configuration and programming files used for the Stratix III device and any companion devices.

For more information about programming file formats and using the programmer, refer to the *Quartus II Programmer* chapter in volume 3 of the *Quartus II Handbook*.

Programming Files

To store the Stratix III data in configuration devices, you can generate additional files or convert the default SRAM Object File (**.sof**) data into a different file format to program a configuration device. You can set up the software to generate additional programming files during compilation in the **Device and Pin Options** dialog box. To convert the **.sof** file, on the File menu, click **Convert Programming Files**. Programmer Object Files (**.pof**) and JTAG Indirect Configuration files (**.jic**) files are used with the Quartus II Programmer to program configuration devices, while the Hexout (**.hexout**), raw binary file (**.rbf**), Tabular Text File (**.ttf**), and **.rpd** files are used with other programmers.

When working with multi-device configuration chains, combine each device's **.sof** file into one configuration file in the **Convert Programming Files** dialog box. When you set up the single configuration file, list the configuration files in the same order as the devices on the board.

Estimating Configuration File Sizes

The uncompressed **.rbf** file size provides the approximate uncompressed configuration file sizes for Stratix III devices. To calculate the amount of storage space required for multiple device configuration, add the file size of each device together.

Use the data on the uncompressed raw binary file sizes only to estimate the file size before design compilation. Different configuration file formats, such as Hexadecimal (**.hex**) or TTF format, have different file sizes. However, for any specific version of the Quartus II software, any design targeted for the same device has the same uncompressed configuration file size.

Stratix III devices can receive a compressed configuration bitstream and decompress this data in real-time, reducing storage requirements and configuration time in active serial and passive serial schemes. If you are using compression, the file size can vary after each compilation because the compression ratio is dependent on the design.

Early System Planning

In systems that contain a Stratix III device, the FPGA typically plays a large role in the overall system and affects the rest of the system design. Providing FPGA device information early in the design process, before you have completed your design in the Quartus II software, enables earlier planning for the system and board design. It is important to start the design process by creating detailed design specifications. You must decide important aspects of your FPGA that affect the system design, including IP, and on-chip debugging methodologies. You can also perform early power estimation to provide information to PCB board and system designers before you have created any source code, or when you have a preliminary version of the design. This section includes the following topics:

Creating Design Specifications

Before you create your logic design or complete your system design, you should have detailed design specifications. The specifications define what the system should do, specify the I/O interfaces for the FPGA, and include a block diagram of basic design functions. Refer to "IP Selection" for suggestions about including intellectual property blocks. Taking the time to create these specifications will help improve design efficiency.

Creating a test plan at this phase can also help you design for testability and design for manufacturability. For example, do you want to perform any built-in-self test functions to drive interfaces? If so, you could use a UART interface with a Nios® II processor inside the FPGA device. You might require the ability to validate all the design interfaces. Refer to "Planning for On-Chip Debugging," on page 13 for guidelines related to analyzing and debugging the device after it is in the system.

If your design includes multiple designers, it is useful to consider a common design directory structure. This eases the design integration stages. "Planning for Hierarchical and Team-Based Design," on page 47 provides more suggestions for team-based designs.

IP Selection

Altera and its third-party IP partners offer a large selection of off-the-shelf IP cores optimized for Altera devices. You can easily implement these parameterized blocks of IP in your design, reducing your system implementation and verification time, and allowing you to concentrate on adding proprietary value. IP selection often affects system design, especially where the FPGA interfaces with other devices in the system. Consider which I/O interfaces or other blocks in your system design can be implemented using IP cores, and plan to incorporate these cores in your FPGA design.

The OpenCore Plus feature available for many IP cores allows you to program the FPGA to verify your design in hardware before you purchase the IP license. The evaluation supports an untethered mode, in which the design runs for a limited time, or a tethered mode. The tethered mode requires an Altera serial JTAG cable connected between the JTAG port on your board and a host computer running the Quartus II Programmer for the duration of the hardware evaluation period. If you plan to use the tethered mode, ensure that your board design supports this mode of operation.

For descriptions of available IP cores, refer to the Intellectual Property page under Products on Altera's website.

Planning for On-Chip Debugging

On-chip debugging is an optional step in the design flow, and different debugging tools work better for different systems and different designers. Evaluate on-chip debugging options early in your design process to ensure that your system board, Quartus II project, and design are able to support the appropriate options. Planning can reduce time spent debugging, and you will not have to make changes later to accommodate your preferred debugging methodologies. Adding debug

pins may not be enough, because of internal signal accessibility and I/O pin accessibility on the device. First, select your preferred debugging tool(s) described in"On-Chip Debugging Tools" and then refer to "Planning Guidelines for Debugging Tools," on page 15.

On-Chip Debugging Tools

The Quartus II portfolio of verification tools includes the following in-system debugging features:

- SignalProbe incremental routing—This feature makes design verification more efficient by quickly routing internal signals to I/O pins without affecting the routing of the original design. Starting with a fully routed design, you can select and route signals for debugging to either previously reserved or currently unused I/O pins.
- SignalTap® II Embedded Logic Analyzer—This logic analyzer helps you debug an FPGA design by probing the state of internal and I/O signals without the use of external equipment or extra I/O pins, while the design is running at full speed in an FPGA device. Defining custom trigger-condition logic provides greater accuracy and improves the ability to isolate problems. The SignalTap II Embedded Logic Analyzer does not require external probes or changes to the design files to capture the state of the internal nodes or I/O pins in the design; all captured signal data is stored in device memory until you are ready to read and analyze the data. You can use this feature with incremental compilation to save compilation time.
- Logic Analyzer Interface—This interface enables you to connect and transmit internal FPGA signals to an external logic analyzer for analysis, allowing you to take advantage of advanced features in your external logic analyzer or mixed signal oscilloscope. You can use this feature to connect a large set of internal device signals to a small number of output pins for debugging purposes.
- In-System Memory Content Editor—This feature provides read and write access to in-system FPGA memories and constants through the JTAG interface, making it easy to test changes to memory content and constant values in the FPGA while the device is functioning in the system.
- In-System Sources and Probes—This feature sets up customized register chains to drive or sample the instrumented nodes in your logic design, providing an easy way to input simple virtual stimuli and capture the current value of instrumented nodes. You can force trigger conditions set up using the SignalTap II Logic Analyzer, create simple test vectors to exercise your design without the use of external test equipment, and dynamically control run-time control signals with the JTAG chain.

- Virtual JTAG Megafunction—The sld_virtual_jtag megafunction enables you to build your own system-level debugging infrastructure, including both processor-based debugging solutions and debugging tools in software for system-level debugging. This megafunction can be instantiated directly in your HDL code to provide one or more transparent communication channels to access parts of your FPGA design using the JTAG interface of the device.

For more information about these debugging tools, refer to the *sld_virtual_jtag Megafunction User Guide* and *Section V. In-System Design Debugging* in volume 3 of the *Quartus II Handbook*. The section overview provides more information about choosing a debugging solution.

Planning Guidelines for Debugging Tools

If you intend to use any of the on-chip debugging tools, plan for the tools when developing your system board, Quartus II project, and design, as described in this section.

The SignalTap II logic analyzer works best for synchronous interfaces. For debugging asynchronous interfaces, consider using SignalProbe or an external logic analyzer to view the signals most accurately.

The SignalTap II Embedded Logic Analyzer, Logic Analyzer Interface, In-System Memory Content Editor, In-System Sources and Probes, and Virtual JTAG Megafunction all require JTAG connections to perform in-system debugging. Plan your system and board with JTAG ports that are available for debugging.

The JTAG debugging features also require a small amount of additional logic resources to implement the JTAG hub logic. If you set up the appropriate feature early in your design cycle, you can include these device resources in your early resource estimations to prevent over-filling the device with logic.

The SignalTap II Embedded Logic Analyzer uses device memory to capture data during system operation. Consider reserving device memory for use during debugging to ensure that you have enough memory resources to take advantage of this debugging technique.

To use incremental debugging with the SignalTap II Embedded Logic Analyzer, ensure that the **Full incremental compilation** option is on. This option is on by default for projects created in the Quartus II software version 6.1 or later, but is not turned on automatically for existing projects. If incremental compilation is not turned on, you must recompile the entire design when you want to add debugging functions, or when

you make certain changes to SignalTap II settings. Using incremental compilation with the SignalTap II Embedded Logic Analyzer greatly reduces the compilation time required for debugging.

SignalProbe and the Logic Analyzer Interface require I/O pins for debugging. Reserve I/O pins for debugging so you do not have to change the design or board to accommodate debugging signals later. The Logic Analyzer Interface can multiplex signals with design I/O pins if required. Ensure that your board supports some kind of debugging mode, where debugging signals do not affect system operation.

If you use an external logic analyzer or mixed signal oscilloscope, incorporate a pin header or mictor connector for the analyzer.

If you use the Virtual JTAG megafunction for custom debugging applications, you must instantiate it and incorporate it as part of the design process.

The In-System Sources and Probes feature also requires that you instantiate a megafunction in your HDL code. In addition, you have the option to instantiate the SignalTap II Embedded Logic Analyzer as a megafunction so you can connect it to nodes in your design manually and ensure that the tapped node names are not changed during synthesis. You can add the debugging block as a separate design partition for incremental compilation to minimize recompilation times.

To use the In-System Memory Content Editor for RAM or ROM blocks or the LPM_CONSTANT megafunction, turn on the **Allow In-System Memory Content Editor to capture and update content independently of the system clock** option when you create the memory block in the MegaWizard® Plug-In Manager.

Early Power Estimation

FPGA power consumption is an important design consideration. Power consumption must be estimated accurately to develop an appropriate power budget and to design the power supplies, voltage regulators, decoupling, heat sink, and cooling system. Power estimation and analysis have two significant planning requirements:

- Thermal planning—The cooling solution sufficiently dissipates the heat generated by the device. In particular, the computed junction temperature must fall within normal device specifications.
- Power supply planning—Power supplies must provide adequate current to support device operation.

Power consumption in FPGA devices is dependent on the logic design. This dependence can make power estimation challenging during the early board specification and layout stages. The Altera PowerPlay Early Power Estimator (EPE) spreadsheet allows you to estimate power utilization before the design is complete by processing information about the device and the device resources that will be used in the design, as well as the operating frequency, toggle rates, and environmental considerations. You can use the spreadsheet to calculate the device junction temperature by entering the ambient temperature, along with information about the heat sinks, air flow, and board thermal model. The EPE then calculates the power, current estimates, and thermal analysis for the design.

If you do not have an existing design, estimate the number of device resources used in your design and enter it manually. The spreadsheet accuracy depends on your inputs and your estimation of the device resources. If this information changes (during or after your design is complete), your power estimation results are less accurate. If you have an existing design or a partially-completed compiled design, use the **Generate PowerPlay Early Power Estimator File** command in the Quartus II software to provide input to the spreadsheet.

The PowerPlay Early Power Estimator spreadsheet includes the Import Data macro, which parses the information in the power estimation file and transfers it into the spreadsheet. If you do not want to use the macro, you can transfer the data into the Early Power Estimator spreadsheet manually. You should enter additional resources to be used in the final design manually if the existing Quartus II project represents only a portion of your full design. You can edit the spreadsheet and add additional device resources or adjust the parameters after importing the power estimation file information.

When the design is complete, the PowerPlay Power Analyzer tool in the Quartus II software provides an accurate estimation of power that ensures that thermal and supply budgets are not violated. Refer to "Power Analysis," on page 57.

The PowerPlay Early Power Estimator spreadsheets and user guides for each supported device family are available in the Devices section under Support on the Altera website:
www.altera.com/support/devices/estimator/pow-powerplay.html

For more information about using the estimator spreadsheet, refer to the *PowerPlay Early Power Estimator User Guide For Stratix III FPGAs*. For more information about power estimation and analysis, refer to the *PowerPlay Power Analysis* chapter in volume 3 of the *Quartus II Handbook*.

Board Design Considerations

When designing the interfaces to the Stratix III device, various factors can affect the printed circuit board (PCB) design. This section includes important guidelines for the following topics:

"I/O and Clock Planning," on page 27 details the I/O signal connections for the FPGA, which also affect the board design.

For more board design guidelines, refer to Altera's Board Design Guidelines Solution Center at: www.altera.com/support/devices/board/brd-index.html. This Solution Center points to application notes and other documentation that can help you implement successful high-speed PCBs that integrate Altera devices with other elements.

Device Power-Up

Stratix III devices offer hot socketing, which is also known as hot plug-in or hot swap, and power sequencing support without the use of any external devices. You can insert or remove a Stratix III device or a board in a system during system operation without causing undesirable effects to the running system bus or the board inserted into the system. The hot socketing feature helps you use Stratix III devices on printed circuit boards (PCBs) that contain a mixture of voltages.

The system can drive signals into the device I/O before and during power-up. You can power-up or power-down the V_{CCIO}, V_{CC}, V_{CCPGM}, and V_{CCPD} supplies in any sequence. The individual power supply ramp-up and ramp-down rates can range from 50 µs to 100 ms. The power ramp must be monotonic. In a hot socketing situation, the Stratix III device's output buffers are turned off during system power-up or power-down. Also, the Stratix III device does not drive out until the device is configured and working within recommended operating conditions.

Hot socketing circuitry is not available on configuration pins `CONF_DONE`, `nCEO`, and `nSTATUS` because they are required during configuration. Therefore, it is expected behavior for these pins to drive out during power-up and power-down sequences.

When power is applied to a Stratix III device, a power-on-reset event occurs if the power supply reaches the recommended operating range within a certain period of time (specified as a maximum power supply ramp time; t_{RAMP}). The maximum power supply ramp time for Stratix III devices is 100 ms, while the minimum power supply ramp time is 50 µs. The power-on reset (POR) circuit keeps the FPGA in reset until the V_{CC}, V_{CCL}, V_{CCPD}, V_{CCPGM}, and V_{CCPT} power voltage levels monitored by the circuitry have stabilized on power-up. After the device enters user mode, the POR circuitry continues to monitor for brown-out conditions during user mode. In Stratix III devices, a pin-selectable option (PORSEL) is provided that allows you to select between a POR typical delay time of 12 ms or 100 ms. In both cases, you can extend the POR time by using an external component to assert the `nSTATUS` pin low. Extending POR time is required if the board cannot meet the maximum power ramp time specifications, to ensure the device configures properly and enters user mode.

For more information, refer to the *Hot Socketing & Power-On Reset in Stratix III Devices* chapter in the *Stratix III Device Handbook*.

Although power sequencing is not a requirement for correct operation, you should consider the power-up timing of each rail to prevent problems with long-term device reliability when designing a multi-rail-powered system. You can reduce the device in-rush current with proper sequencing and voltage regulator design.

For more information, refer to *AN 448: Stratix III Power Management Design Guide*.

Power Pin Connections

The Stratix III selectable core voltage gives you the option of using a power-saving 0.9-V core voltage V_{CCL} or the standard 1.1-V core voltage V_{CCL}. In either case, a 1.1-V supply is required for the PLL and periphery power supplies V_{CC} and V_{CCD_PLL}. The voltages V_{CC_CLKIN}, V_{CCA_PLL}, V_{CCPT}, and V_{CCBAT} require a 2.5-V supply. The V_{CCPGM} pin powers the configuration pins, and can be 1.8, 2.5, 3.0, or 3.3-V.

The I/O voltage V_{CCIO} connections depend on the design's I/O standards and support 1.2, 1.5, 1.8, 2.5, 3.0, and 3.3-V. Note that the device output pins do not meet the I/O standard specifications if the V_{CCIO} level is out of the recommended operating range for the I/O standard. The `VCCPD` pin must be connected to 3.3-V for a 3.3-V V_{CCIO}, 3.0-V for a 3.0-V V_{CCIO}, and 2.5-V for lower I/O voltages. Voltage reference (VREF) pins serve as voltage references for certain I/O standards. The VREF pin is used mainly for a voltage bias and does not source or sink much current. The

maximum value of this current is 10 µA, and it is independent of the number of I/O pins using a particular VREF pin as their reference voltage. The voltage can be created with a regulator or a resistor divider network. For more information about V_{CCIO} voltages and VREF pins for different I/O banks, refer to "Selectable I/O Standards and Flexible I/O Banks," on page 31.

When designing a board, check that all the power pins are connected correctly, explore any unique requirements for FPGA power pins or other power pins on your board, and determine which devices on your board can share a power rail.

For a list of the supply voltages required for the Stratix III device and their recommended operation conditions, refer to the *DC & Switching Characteristics of Stratix III Devices* chapter in the *Stratix III Device Handbook*. For more details about Stratix III I/O pins, refer to the Stratix III Device Pinout Files at **www.altera.com** and the *Stratix III Pin Connection Guidelines*.

In Stratix III devices, Altera suggests that you use a linear regulator to power the V_{CCA_PLL} and V_{CCPT} power pins, because they power analog circuitry. You can power the digital voltage rails by linear or switching regulators depending on efficiency or cost considerations.

For recommendations about power management system design, including details about power sequencing and power distribution architecture, refer to *AN 448: Stratix III Power Management Design Guide*.

Decoupling Capacitors

Board decoupling becomes more significant to improve overall power supply signal integrity with increased power supply requirements.

Stratix III devices include embedded on-package and on-die decoupling capacitors to provide high-frequency decoupling that external PCB decoupling capacitors and voltage regulator modules cannot support. These low-inductance capacitors suppress power noise for excellent signal integrity performance, and reduce the number of external PCB decoupling capacitors, saving board space, reducing cost, and greatly simplifying PCB design.

Altera has created an easy-to-use power distribution network (PDN) design tool that optimizes the board-level PDN graphically. The purpose of the board-level PDN is to distribute power and return currents from the voltage regulating module (VRM) to the FPGA power supplies, and support optimal transceiver signal integrity and FPGA performance.

For each power supply, PDN designers must choose a network of bulk and ceramic decoupling capacitors. While SPICE simulation could be used to simulate the circuit, the PDN design tool provides a fast, accurate, and interactive way to determine the right number of decoupling capacitors for optimal cost and performance trade-offs.

For more information about the PDN design and optimization process, refer to the *Technical Brief 92: High-Speed Board Design Advisor—Power Distribution Network*.

Contact your Altera sales representative for a copy of the PDN tool.

PLL Board Design Guidelines

For more information about designing your clock and PLL scheme, refer to "Clock and PLL Selection," on page 37 and "PLL Design Guidelines," on page 39. Because PLLs contain analog components embedded in a digital device, the following list provides several considerations to design a board for PLL usage and minimize jitter:

- Ensure that all V_{CCA_PLL} and V_{CCD_PLL} power pins are connected to a 2.5-V and 1.1-V power supply, respectively, even if you do not use all the PLLs in the device.
- Run a thick trace (at least 20 mils) from the power supply to each V_{CCA_PLL} pin.
- Use a isolated linear regulator to power V_{CCA_PLL}.
- Connect all V_{CCD_PLL} power pins to the quietest digital supply on the board.
- Filter each V_{CCA_PLL} and V_{CCD_PLL} pin with a decoupling circuit.
- Use ferrite bead and tantalum parallel capacitor for V_{CCA_PLL} and V_{CCD_PLL} where the power enters the board.

Configuration Pin Connections

Depending on your configuration scheme, different pull-up/pull-down resistor or signal integrity requirements may apply. Some configuration pins also have specific requirements if unused. It is very important to connect the configuration pins correctly. This section presents some guidelines to address common issues.

For a list of the dedicated and dual-purpose configuration pins, and a description of the function and connection guidelines, refer to the *Configuring Stratix III Devices* chapter in the *Stratix III Handbook*. For more details about Stratix III I/O pins, refer to the Stratix III Device Pinout Files at **www.altera.com** and the *Stratix III Pin Connection Guidelines*.

DCLK and TCK Signal Integrity

The TCK and/or DCLK traces should produce clean signals with no overshoot, undershoot, or ringing. When designing the board, lay out the TCK and DCLK traces with the same techniques used to lay out a clock line. Any overshoot, undershoot, ringing, or other noise on the TCK signal can affect JTAG configuration. A noisy DCLK signal can affect configuration and cause a Cyclic Redundancy Check (CRC) error. For a chain of devices, noise on any of the TCK or DCLK pins in the chain could cause JTAG programming or configuration to fail for the entire chain.

For more information about connecting devices in a chain, refer to the *Configuring Stratix III Devices* chapter in the *Stratix III Handbook*.

JTAG Pins

If you are not using the JTAG interface, connect the JTAG pins to a stable voltage level. Because JTAG configuration takes precedence over all other configuration methods, these pins should not be left floating or toggling during configuration.

If you are using the JTAG interface, follow the guidelines in this section.

JTAG Pin Connections

A device operating in JTAG mode uses four required pins—TDI, TDO, TMS, and TCK and one optional pin, TRST. The TCK pin has an internal weak pull-down resistor, while the TDI, TMS, and TRST pins have weak internal pull-up resistors (typically 25 kΩ). JTAG output pin TDO and all JTAG input pins are powered by the 2.5/3.0/3.3-V V_{CCPD}. All of the JTAG pins support only LVTTL I/O standard.

Connect the JTAG pins of the device to the download cable header correctly. Ensure the pin order is not reversed. If you have more than one device in the chain, connect the TDO pin of a device to the TDI pin of the next device in the chain.

Noise on the JTAG pins during configuration, user mode, or power-up can cause the device to go into an undefined state or mode.

To disable the JTAG state machine during power-up, the TCK pin should be pulled low to ensure that an unexpected rising edge does not occur on TCK. Altera recommends pulling the TCK pin low and the TMS pin high through resistors.

Connect TRST to V_{CCPD} through a 1 kΩ resistor. Connecting the pin to ground disables the JTAG circuitry.

Download Cable Operating Voltage

The operating voltage supplied to the Altera download cable by the target board through the 10-pin header determines the operating voltage level of the download cable. The dedicated JTAG pins for all Stratix III devices reside in bank 1A and they are powered by V_{CCPD}. Because the download cable interfaces with the JTAG pins of your device, ensure that the download cable operating voltage and the JTAG pin voltage are compatible.

In a JTAG chain containing devices with different V_{CCIO} levels, the devices with a higher V_{CCIO} level should drive the devices with the same or lower V_{CCIO} level. You need one level shifter at the end of the chain with this device arrangement. If this arrangement is not possible, you have to add more level shifters into the chain.

For recommendations about connecting a JTAG chain with multiple voltages across the devices in the chain, refer to the *IEEE 1149.1 (JTAG) Boundary Scan Testing in Stratix III Devices* chapter in the *Stratix III Device Handbook*.

JTAG Signal Buffering

You may have to add buffers to a JTAG chain, depending on the JTAG signal integrity, especially the `TCK` signal, because it is the JTAG clock and is the fastest switching JTAG signal. Altera recommends buffering the signals at the connector because cables and board connectors tend to make bad transmission lines and introduce noise to the signals. After this initial buffer at the connector, add buffers as the chain gets longer or whenever the signals cross a board connector.

If a cable drives three or more devices, buffer the JTAG signal at the cable connector to prevent signal deterioration. Of course, this also depends on the board layout, loads, connectors, jumpers, and switches on the board. Anything added to the board that affects the inductance or capacitance of the JTAG signals increases the likelihood that a buffer should be added to the chain.

Each buffer should drive no greater than eight loads for the `TCK` and `TMS` signals, which drive in parallel. If jumpers or switches are added to the path, decrease the number of loads.

MSEL Configuration Mode Pins

To select the configuration scheme by driving the Stratix III device MSEL pins either high or low. The MSEL pins are powered by the V_{CCPGM} power supply of the residing bank. The `MSEL[2..0]` pins have 5-kΩ internal pull-down resistors that are always active. During POR and

reconfiguration, the MSEL pins have to be at LVTTL VIL and VIH levels to be considered a logic low and logic high. To avoid any problems with detecting an incorrect configuration scheme, hard-wire the appropriate MSEL pins to V_{CCPGM} or GND. Alternately, set up your board so that you can connect each pin to either V_{CCPGM} or GND with a 0-Ω resistor, so that you can change between configuration modes during testing or debugging. Do not drive the MSEL pins with a microprocessor or another device. Do not leave the MSEL pins floating.

Other Configuration Pins

Ensure all dedicated and dual-purpose configuration pins are connected correctly, including the following pins.

The V_{CCPD} dedicated power pin is used to power the I/O pre-drivers, the JTAG input and output pins, and the design security circuitry. Connect this pin to 3.3 V for 3.3-V I/O standards, 3.0 V for 3.0-V I/O standards, or 2.5 V for 2.5-V or lower I/O standards.

The `nIO_PULLUP` pin chooses whether the internal pull-up resistors on the user I/O pins and dual-purpose I/O pins (`DATA[7..0]`, `CLKUSR`, `INIT_DONE`, `DEV_OE`, `DEV_CLRn`, `CRC_ERROR`) are on or off before and during configuration. Tie the `nIO-PULLUP` directly to V_{CCPGM} or use a 1-k Ω pull-up resistor to turn off the internal pull-up resistors, or tie `nIO-PULLUP` directly to GND to turn on the internal pull-up resistors.

The `nCE` chip enable pin must be held low during configuration, initialization, and user mode. In single device configuration or JTAG programming, tie `nCE` low. In multi-device configuration, tie `nCE` low on the first device and connect its `nCEO` pin to the `nCE` pin on the next device in the chain.

Board-Related Quartus II Settings

The Quartus II software provides options for the FPGA I/O pins that you should consider during board design.

Device-Wide Output Enable Pin

Stratix III devices support an optional chip-wide output enable that allows you to override all tri-states on device I/O. When this `DEV_OE` pin is driven low, all I/O pins are tri-stated; when this pin is driven high, all pins behave as programmed. To use this chip-wide output enable, turn on **Enable device-wide output enable (DEV_OE)** in the Quartus II software before compiling your design on the **General** tab in the **Device & Pin Options** dialog box.

Unused Pins

To allow flexibility in board design, you can specify the state of unused pins as one of the following five states in the Quartus II software: as inputs that are tri-stated, as outputs that drive ground, as outputs that drive an unspecified signal, as input tri-stated with bus-hold, or as input tri-stated with weak pull-up. To improve signal integrity, set unused pins as outputs that drive ground and tie them directly to the ground plane on the board. Doing so reduces inductance by creating a shorter return path and reduces noise on the neighboring I/O. To reduce power dissipation, set clock pins to drive ground and set other unused I/O pins as inputs that are tri-stated. To make the setting that is appropriate for your design, choose one of the five allowable states for **Reserve all unused pins** on the **Unused Pins** tab in the **Device & Pin Options** dialog box, or apply the **Reserve Pin** assignment to specific pins in the Pin Planner.

When you compile your design, the Quartus II software generates the pin report file (**.pin**) to specify how you should connect the device pins. Unused I/O pins are marked in the report file according to the unused pins option you set in the Quartus II software. All I/O pins specified as GND* can either be connected to ground to improve the device's immunity to noise, or left unconnected. Leave all `RESERVED` I/O pins unconnected on your board because these I/O pins drive out unspecified signals. Tying a `RESERVED` I/O pin to V_{CC}, ground, or another signal source can create contention that can damage the device output driver. You can connect `RESERVED_INPUT` I/O pins to a high or low signal on the board, while `RESERVED_INPUT_WITH_WEAK_PULLUP` and `RESERVED_INPUT_WITH_BUS_HOLD` pins can be left unconnected.

Signal Integrity Considerations

This section contains a few board design guidelines related to voltage reference pins, simultaneous switching noise, and I/O termination.

For more information about signal integrity, refer to the *Stratix III FPGA Signal Integrity* white paper.

Voltage Reference Pins

It is important for VREF voltage reference signals to be noise-free. For more information about voltage reference pins and I/O standards, refer to "I/O Features and Pin Connections," on page 30. Voltage deviation on a VREF pin can affect the threshold sensitivity for inputs.

Simultaneous Switching Noise

Simultaneous switching noise (SSN) becomes a concern when too many pins (in close proximity) change voltage levels at the same time. Noise generated by SSN can reduce noise margin and cause incorrect switching. Although SSN is dominant on the device package, refer to the PCB guidelines in Altera's Board Design Guidelines Solution Center at www.altera.com/support/devices/board/brd-index.html for board layout recommendations that can help reduce some of the noise.

Breaking out large bus signals on board layers close to the device can help reduce cross talk. If two signal layers are next to each other, route traces orthogonally, if possible. Use a separation of 2× to 3× the trace width, if possible.

I/O Termination

Voltage-referenced I/O standards require both an input reference voltage, VREF, and a termination voltage, VTT. The reference voltage of the receiving device tracks the termination voltage of the transmitting device. Each voltage-referenced I/O standard requires a unique termination setup. For example, a proper resistive signal termination scheme is critical in SSTL2 standards to produce a reliable DDR memory system with superior noise margin.

Although single-ended, non-voltage-referenced I/O standards do not require termination, impedance matching is necessary to reduce reflections and improve signal integrity.

Stratix III on-chip series and parallel termination provides the convenience of no external components. Alternatively, you can use external pull-up resistors to terminate the voltage-referenced I/O standards such as SSTL and HSTL.

Differential I/O standards typically require a termination resistor between the two signals at the receiver. The termination resistor must match the differential load impedance of the signal line. Stratix III devices provide an optional differential on-chip resistor when using LVDS. Note that certain dedicated clock input pairs do not support differential termination.

For more information about on-chip termination features and limitations, refer to "I/O Features and Pin Connections," on page 30.

Board-Level Simulation and Advanced I/O Timing Analysis

To ensure that the I/O signaling meets receiver threshold levels on your board setup, perform full board routing simulation with third-party board-level simulation tools using an IBIS model.

When this feature is available in the Quartus II software, select **IBIS** under **Board-level signal integrity analysis** on the **EDA Tool Settings** page of the **Settings** dialog box.

The IBIS models will also be listed on Altera's website when they are available.

For more information about this simulation flow, refer to the *Signal Integrity with Third-Party Tools* chapter in volume 3 of the *Quartus II Handbook*.

When you include an FPGA device with high-speed interfaces in a board design, knowing the signal integrity and board routing propagation delay is vital for proper system operation. You should analyze board-level timing as part of I/O and board planning, especially for high-speed designs.

You can configure board trace models of selected I/O standards and generate "board-aware" signal integrity reports with the Quartus II software. When **Enable Advanced I/O Timing** is turned on, the TimeQuest Timing Analyzer uses simulation results for the I/O buffer, package, and board trace model to generate more accurate I/O delays and extra reports to give insight into signal behavior at the system level. You can use these advanced timing reports as a guide to make changes to the I/O assignments and board design to improve timing and signal integrity.

I/O and Clock Planning

Planning and allocating I/O and clock resources is an important task with the high pin counts and advanced clock management features in Stratix III devices. Understanding various considerations to effectively plan the available I/O resources to maximize utilization and prevent issues related to signal integrity is important. Good clock management systems are also crucial to the performance of an FPGA design. Stratix III devices offer a hierarchical clock structure and multiple PLLs with advanced features to provide a complete clock management solution.

The I/O and clock connections of your FPGA affect the rest of your system and board design, so it is important to plan these connections early in your design cycle.

This section details the following topics:

Making FPGA Pin Assignments

Use the Quartus II Pin Planner to make pin assignments. With the Pin Planner GUI, you can identify I/O banks, VREF groups, and differential pin pairings to help you through the I/O planning process. Right-click in the Pin Planner spreadsheet interface and click the **Pin Finder** to search for specific pins.

If migration devices are selected, as described in "Vertical Device Migration," on page 3, the Pin Migration view highlights pins that change function in the migration device when compared to the currently selected device.

You have the option of importing a Microsoft Excel spreadsheet into the Quartus II software to start the I/O planning process if you normally use a spreadsheet in your design flow. You can also export a Comma-Separated Value (**.csv**) file containing your I/O assignments for spreadsheet use when all pins are assigned.

When you compile your design in the Quartus II software, I/O Assignment Analysis in the Fitter validates that the assignments meet all the device requirements and generates messages if there are any problems.

You can then pass the pin location information to PCB designers. It is important that pin assignments match between the Quartus II software and your schematic and board layout tools to ensure the design works correctly on the board where it is placed, especially if changes to the pin-out must be made. The Pin Planner is tightly integrated with certain PCB design EDA tools and can read pin location changes from these tools to check the suggested changes. When you compile your design, the Quartus II software generates the PIN file. You can use this file to verify that each pin is correctly connected in board schematics.

When assigning pins, ensure that targeted pins support the appropriate I/O standard and any other required I/O features, as described in "I/O Features and Pin Connections," on page 30. Certain I/O banks on the top and bottom or left and right of the device support different I/O standards

and voltage levels. You can assign I/O standards and make other I/O-related settings in the Pin Planner. Be sure to use the correct dedicated pin inputs for signals such as clocks and global control signals, as described in "Clock and PLL Selection," on page 37.

For details about using the Pin Planner to make I/O assignments, refer to the *I/O Management* chapter in volume 2 of the *Quartus II Handbook*. For more information about passing I/O information between the Quartus II software and third-party EDA tools, refer to the *Mentor Graphics PCB Design Tools Support* and *Cadence PCB Design Tools Support* chapters in volume 2 of the *Quartus II Handbook*.

Early Pin Planning and I/O Assignment Analysis

In many design environments, the FPGA designers want to plan top-level FPGA I/O pins early so that board designers can start developing the PCB design and layout. The FPGA device's I/O capabilities and board layout guidelines influence pin locations and other types of assignments. In cases where the board design team specifies an FPGA pin-out, it is crucial that you verify pin locations in the FPGA place-and-route software as soon as possible to avoid the need for board design changes.

The Quartus II **Pin Planner** enables easy I/O pin assignment planning, assignment, and validation, as described in "Making FPGA Pin Assignments," on page 28. The Quartus II **Start I/O Assignment Analysis** command checks that the pin locations and assignments are supported in the target FPGA architecture. Checks include reference voltage pin usage, pin location assignments, and mixing of I/O standards. You can use I/O Assignment Analysis to validate I/O-related assignments that you make or modify throughout the design process.

Starting FPGA pin planning early improves the confidence in early board layouts, reduces the chance of error, and improves the design's overall time to market. You can create a preliminary pin-out for an Altera FPGA using the Quartus II Pin Planner before the source code is designed.

Early in the design process, the system architect typically has information about the standard I/O interfaces (such as memory and bus interfaces), IP cores to be used in the design, and any other I/O-related assignments defined by system requirements. The Pin Planner **Create/Import Megafunction** feature interfaces with the MegaWizard Plug-In Manager, and enables you to create or import custom megafunctions and IP cores that use I/O interfaces. Enter PLL and LVDS blocks, including options such as dynamic phase alignment (DPA) because options affect the pin placement rules. When you have entered as much I/O-related information as possible, generate a top-level design netlist file using the **Create Top-Level Design File** command. You can use the I/O analysis

results to change pin assignments or IP parameters and repeat the checking process until the I/O interface meets your design requirements and passes the pin checks in the Quartus II software.

When planning is complete, the pin location information can be passed to PCB designers. The Pin Planner is tightly integrated with certain PCB design EDA tools, and can read pin location changes from these tools to check the suggested changes. It is important that pin assignments match between the Quartus II software and your schematic and board layout tools to ensure the design works correctly on the board where it is placed, especially if changes to the pin-out must be made.

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

When the design is complete, use the reports and messages generated by the Quartus II Fitter for the final sign-off of pin assignments.

I/O Features and Pin Connections

Stratix III I/Os are designed for ease of use and rapid system integration, while simultaneously providing high bandwidth. Independent modular I/O banks with a common bank structure for vertical migration lend efficiency and flexibility to the high speed I/O. This section provides guidelines related to I/O features and pin connections. It describes support for different I/O signal types and I/O standards in device I/O banks, as well as other I/O features available for your design. It also provides information about memory interfaces, pad placement guidelines, and special pin connections.

For more details about Stratix III I/O pins, refer to the Stratix III Device Pinout Files at **www.altera.com** and the *Stratix III Pin Connection Guidelines*.

I/O Signaling Type

Stratix III devices support a wide range of industry I/O standards, including single-ended, voltage-referenced single-ended, and differential I/O standards. This section provides general guidelines on selecting a signaling type.

Single-ended I/O signaling provides a simple rail-to-rail interface. Its speed is limited by the large voltage swing and noise. Single-ended I/Os do not require termination, unless reflection in the system causes undesirable effects.

Voltage-referenced signaling reduces effects of simultaneous switching outputs (SSO) from pins changing voltage levels at the same time (for example, external memory interface data and address buses). It also provides improved logic transition rate with a reduced voltage swing, and minimizes noise caused by reflection with a termination requirement. The only drawback is additional termination components to the reference voltage source, VTT.

Differential signaling eliminates the interface performance barrier of single-ended and voltage-referenced signaling, with superior speed using an additional inverted closely-coupled data pair. It also avoids the need for a clean reference voltage. This is possible with lower swing voltage and noise immunity with common mode noise rejection capability. Considerations for this implementation include the requirements for a dedicated PLL to generate a sampling clock, and matched trace lengths to eliminate the phase difference between inverted and non-inverted pair.

Stratix III I/O pins are organized in pairs to support differential standards. Each I/O pin pair can support differential input or output operations, with the exception of certain clock pins that support differential input operations only. In your design source code, define just one pin to represent a differential pair, and make a pin assignment for this positive end of the pair. When you specify a differential I/O standard, the Quartus II software automatically places the corresponding negative pin.

Selectable I/O Standards and Flexible I/O Banks

Stratix III I/O pins are arranged in groups called modular I/O banks. Depending on the device density, the number of I/O banks ranges from 16 to 24 banks. During migration from a smaller device to a larger device, the bank size increases or remains the same but never decreases. Place pins in I/O banks that support the appropriate I/O standards.

The board must supply each bank with one V_{CCIO} voltage level. Each I/O bank is powered by the V_{CCIO} pins of that particular bank, and is independent of the V_{CCIO} of other I/O banks. A single I/O bank supports output signals that are driving at the same voltage as the V_{CCIO}. An I/O bank can simultaneously support any number of input signals with different I/O standards.

To accommodate voltage-referenced I/O standards, each Stratix III device I/O bank supports multiple VREF pins feeding a common VREF bus. Set the VREF pins to the correct voltage for the I/O standards in the bank. Each I/O bank can only have a single V_{CCIO} voltage level and a

single VREF voltage level at a given time. If the VREF pins are not used as voltage references, they cannot be used as generic I/O pins and should be tied to V_{CCIO} or GND.

An I/O bank including single-ended or differential standards can support voltage-referenced standards as long as all voltage-referenced standards use the same VREF setting. For performance reasons, voltage-referenced input standards use their own V_{CCPD} level as the power source, so you can place voltage-referenced input signals in a bank with any V_{CCIO} voltage. Voltage-referenced bi-directional and output signals must drive out at the I/O bank's V_{CCIO} voltage level.

The placement of single-ended I/O pins with respect to LVDS I/O pins is restricted. Follow the pin placement rules that specify the number of I/O pins that must separate single-ended outputs and LVDS I/O. During compilation, the Quartus II Fitter verifies that these guidelines are satisfied.

For more information, refer to the *Stratix III Device I/O Features* chapter of the *Stratix III Device Handbook*. Refer to the section that discusses Stratix III I/O standards and voltage levels for supported I/O standards and the typical values for input and output V_{CCIO}, V_{CCPD}, VREF, and board VTT. Also, refer to the Stratix III I/O banks figure that shows the location of each I/O bank and what each bank supports. The figures describing the number of I/Os in each bank provide bank information specific to each device density. Refer to the section describing I/O bank restrictions for information about which I/O standards can be combined in each bank, and the section describing I/O placement guidelines for details about LVDS restrictions.

For the electrical characteristics of each I/O standard, refer to the *DC and Switching Characteristics of Stratix III Devices* chapter in volume 2 of the *Stratix III Device Handbook*.

Memory Interfaces

All I/O banks on every side of the device have dedicated registers and phase-shift circuitry to interface with external memory devices. Stratix III devices support interfaces with DDR3, DDR2, and DDR SDRAM, as well as RLDRAM II and QDRII/QDRII + SRAM.

A self-calibrating megafunction (ALTMEMPHY) is optimized to take advantage of the Stratix III I/O structure. The megafunction allows you to set external memory interface features and helps set up the physical interface (PHY) best suited for your system. Altera and third-party IP cores for memory interfaces also use this megafunction.

Depending on the memory specifications, the DQS data strobe pins can be bidirectional single-ended signals, unidirectional differential signals, bidirectional differential signals, or unidirectional complementary signals. Connect the unidirectional read data-strobes or clocks to Stratix III DQS pins and use any available DQ or DQS pins (in the same I/O bank as the data pins) for the unidirectional write data-strobes or clocks. Stratix III devices offer differential input buffers for differential read data-strobe/clock operations and provide an independent DQS logic block for each `CQn` data strobe pin for complementary read data-strobe/clock operations.

The DQ data pins can be bidirectional or unidirectional signals. Connect the unidirectional read data signals to Stratix III DQ pins and the unidirectional write data signals to a different group of DQ pins. Use the defined DQ groups to assign DQ pins for signals in a data bus.

The DQS and DQ pin locations are fixed in the Stratix III device. The memory interface circuitry is available in every Stratix III I/O bank. The top and bottom I/O banks offer higher performance as compared to the left and right side I/O banks, because the left and right I/O pins have higher pin capacitance to support the LVDS I/O standard.

The DQS phase-shift circuitry uses a delay-locked loop (DLL) to dynamically measure the clock period needed by the DQS/CQn pin. The reference clock for each DLL may come from PLL output clocks or any of the two dedicated clock input pins located in either side of the DLL.

If designing multiple memory interfaces into the device using Altera IP, be sure to generate a unique interface for each instance to ensure good results, instead of designing it once and instantiating it multiple times.

For more information about connecting Stratix III device with external memory devices, refer to the *External Memory Interfaces in Stratix III Devices* chapter in the *Stratix III Device Handbook*. For additional resources, refer to the External Memory Solutions Center at: www.altera.com/technology/memory/mem-index.jsp. Refer to the appropriate application note for specific guidelines related to specific memory interfaces.

Pad Placement Guidelines

The V_{CCIO} supply for a bank is susceptible to noise from switching outputs in the bank. To maintain acceptable noise level on the V_{CCIO} supply, there are restrictions on the placement of single-ended I/O pads in relation to differential pads. The Quartus II software automatically checks for these restrictions.

When there are single-ended voltage-referenced inputs in a bank, the Quartus II software automatically checks for restrictions on the placement of outputs in relation to VREF pads and supply pairs (V_{CCIO} and GND). The restriction is in place to maintain acceptable noise level on V_{CCIO} supply and prevent output switching noise from shifting the VREF rail.

Neighboring package pins do not correspond to neighboring bond pads. You can locate the bond pads for a pin with the Quartus II Pin Planner Pad View when planning your I/O pin locations.

In specific applications, you can relax the restrictions check in the Quartus II software. For example, if you have a non-toggling single-ended pin, you can safely place it closer to a differential pin. To indicate such a situation to the Quartus II software, make a 0Mhz **Toggle Rate** assignment for the pin in Assignment Editor.

The **Output Enable Group** assignment is another setting which is useful, especially in external memory interfaces. The logic option assigns an output enable group number, allowing you to specify which pins in the design are driving in and out at the same time, so that the design does not violate the requirements for the maximum number of pins driving out of a VREF group when there is a voltage-referenced input. Use this option when the Fitter cannot detect the output enable group of the pins in the VREF group; for example, when the output enable comes from a state machine or complex logic. This option is useful for specifying the output enable group number of DQ and DQS I/O pins, and DM pins if they are used only during a write operation, for use with the dedicated DDR SDRAM interface. Each individual controller should have a different output enable group number.

Dual-Purpose and Special Pin Connections

Stratix III devices allow I/O flexibility with dual-purpose configuration pins. You can use dual-purpose configuration pins as general I/O after device configuration is complete. Select the desired setting for each of the dual-purpose pins on the **Dual-Purpose Pins** tab of the **Device and Pin Options** dialog box. Depending on the configuration scheme, these pins can be reserved as regular I/O pins, as inputs that are tri-stated, as outputs that drive ground, or as outputs that drive an unspecified signal.

You can also use dedicated clock inputs, which drive to the global clock networks, as general-purpose input pins if not used as clock pins. When you use the clock inputs as general inputs, I/O registers use ALM-based registers because the clock input pins do not include dedicated I/O registers.

The device-wide reset and clear pins are available as design I/O if not enabled. For more information, refer to "Device-Wide Output Enable Pin," on page 24 and "Register Power-Up Levels and Control Signals," on page 45.

Dedicated circuitry for a CRC error detection feature is built into Stratix III devices that can optionally check for single event upsets (SEUs) continuously and automatically. To take advantage of the SEU mitigation features, use the appropriate megafunction for CRC error detection. Use the `CRC_ERROR` or `CRITICAL ERROR` pin to flag errors. If not enabled for their CRC function, these pins are available as design I/O.

For more information about SEU mitigation, refer to the *SEU Mitigation in Stratix III Devices* chapter in the *Stratix III Device Handbook*.

Stratix III I/O Features

The Stratix III bi-directional I/O element (IOE) supports a number of features that can help interface with other devices, reducing the complexity and cost of the PCB. Table 2 lists Stratix III I/O features, provides usage information and design considerations, and provides references to more information about the features.

Table 2. Stratix III I/O Features (Part 1 of 3)

Feature	Usage	Guidelines and More Information
MultiVolt I/O Interface	Allows all packages to interface with systems of different supply voltages. V_{CCIO} pins can be connected to a 1.2-, 1.5-, 1.8-, 2.5-, 3.0-V, or 3.3-V power supply, depending on the output requirements. The output levels are compatible with systems of the same voltage as the power supply. V_{CCPD} power pins must be connected to 3.3-V for 3.3-V V_{CCIO}, 3.0-V for 3.0-V V_{CCIO}, and 2.5-V for other I/O voltages.	Refer to the *Stratix III Device I/O Features* chapter of the *Stratix III Device Handbook* for a summary of MultiVolt I/O support, and additional guidelines for 3.0-V and 3.3-V LVTTL/LVCMOS.
High-Speed Differential I/O with Dedicated DPA Support	Dedicated circuitry for specific differential I/O interconnect standards at speeds up to 1.25 Gbps. Dynamic Phase Alignment (DPA) minimizes bit errors, simplifies PCB layout and timing management, and eliminates channel-to-channel and channel-to-clock skew.	Refer to the *High Speed Differential I/O Interfaces with DPA in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*. If you want to use DPA, be sure to enable the feature so that the design uses the correct PLLs on the right and left sides of the device.

Table 2. Stratix III I/O Features (Part 2 of 3)

Feature	Usage	Guidelines and More Information
Programmable Current Strength	Programmable current-strength control available for certain I/O standards. Can mitigate the effects of high signal attenuation due to a long transmission line or a legacy backplane. A higher current strength increases I/O performance, but also increases noise on the interface, so you can use current strength control to manage noise.	Ensure that the output buffer current strength is sufficiently high, but does not cause excessive overshoot or undershoot that violates voltage threshold parameters for the I/O standard. Altera recommends performing IBIS or SPICE simulations to determine the right current strength setting for your specific application. For a list of standards and settings, refer to the *Stratix III Device I/O Features* chapter of the *Stratix III Device Handbook*.
Programmable Slew Rate Control	Configure each pin for low-noise or high-speed performance. A faster slew rate provides high-speed transitions. You can use faster slew rates to improve the available timing margin in memory-interface applications or when the output pin has a high-capacitive loading. A slow slew rate can help reduce system noise, but adds a nominal delay to rising and falling edges. You can use slew rate to reduce SSN.	Confirm that your interface meets its performance requirements if you use slower slew rates. Altera recommends performing IBIS or SPICE simulations to determine the right slew rate setting for your specific application.
Programmable IOE Delay	Programmable delay chains can be used as deskewing circuitry. Use different input delay values from pin to input register to change setup time, or delay values from output register to output pin to change clock-to-output times, and ensure that all bits of a bus have the same delay going into or out of the device.	Refer to the *DC and Switching Characteristics of Stratix III Devices* chapter in volume 2 of the *Stratix III Device Handbook* for the delay specifications. Refer to *AN 474: Implementing Stratix III Programmable I/O Delay Settings in the Quartus II Software*
Programmable Output Buffer Delay	Delay chains in the single-ended output buffer can independently control the rising and falling edge delays of the output buffer.	You can use delays to adjust the output-buffer duty cycle, compensate channel-to-channel skew, reduce SSO noise by deliberately introducing channel-to-channel skew, and improve high-speed memory-interface timing margins.
Open-Drain Output	When configured as open-drain, the logic value of the output is either high-Z or 0. Used in system-level control signals that can be asserted by multiple devices in the system.	Typically, an external pull-up resistor is required to provide logic high.

Table 2. Stratix III I/O Features (Part 3 of 3)

Feature	Usage	Guidelines and More Information
Bus Hold	Weakly holds the signal on an I/O pin at its last-driven state until the next input signal is present, using a resistor with a nominal resistance (RBH) of approximately 7 kΩ. With this feature, you do not require an external pull-up or pull-down resistor to hold a signal level when the bus is tri-stated. The circuitry also pulls non-driven pins away from the input threshold voltage where noise can cause unintended high-frequency switching.	If the bus-hold feature is enabled, you cannot use the programmable pull-up option. Disable the bus-hold feature if the I/O pin is configured for differential signals. Refer to the *DC and Switching Characteristics of Stratix III Devices* chapter in volume 2 of the *Stratix III Device Handbook* for the specific sustaining current driven through this resistor and the overdrive current used to identify the next-driven input level, for each V_{CCIO} voltage level.
Programmable Pull-Up Resistor	Pull-up resistor (typically 25 kΩ) weakly holds the I/O to the V_{CCIO} level when in user mode. Can be used with open-drain output to eliminate the requirement for an external pull-up resistor.	If the programmable pull-up option is enabled, you cannot use the bus-hold feature.
PCI Clamping Diode	Can be used to protect the pin from excessive overshoot voltage in PCI/PCI-X I/O standard interfaces.	—
On-Chip Termination (OCT)	Driver-impedance matching provides the I/O driver with controlled output impedance that closely matches the impedance of the transmission line to significantly reduce reflections. Support for on-chip series (RS) with or without calibration, parallel (RT) with calibration, and dynamic series and parallel termination for single-ended I/O standards and on-chip differential termination (RD) for differential LVDS I/O standards. You can calibrate the Stratix III I/O bank with any of eight or ten OCT calibration blocks.	OCT RS and RT are supported in the same I/O bank for different I/O standards if they use the same V_{CCIO} supply voltage. Each I/O in an I/O bank can be independently configured to support OCT RS, programmable current strength, or OCT RT. You cannot configure both OCT RS and programmable current strength for the same I/O buffer. Differential on-chip termination RD is available only in row I/O banks; column I/O banks do not support OCT RD. The dedicated clock input pairs `CLK1p`, `CLK1n`, `CLK3p`, `CLK3n`, `CLK8p`, `CLK8n`, `CLK10p`, and `CLK10n` on the row I/O banks of the Stratix III devices do not support RD termination. Refer to the *Stratix III Device I/O Features* chapter of the *Stratix III Device Handbook* for details about the support and implementation of this feature.

Clock and PLL Selection

The first stage in planning your clocking scheme is to determine your system clock requirements. Understand your device's available clock resources and correspondingly plan the design clocking scheme. Consider your requirements for timing performance, as well as how much logic is driven by a particular clock.

Stratix III devices provide dedicated low-skew and high-fanout routing networks. They are organized in a hierarchical structure that provides up to 220 unique clock domains within the device and allows up to 67 unique clock sources per device quadrant. There are up to 12 PLLs per device and up to 10 independently-programmable outputs per PLL. You can use 16 differential dedicated global clock input pins or 32 single-ended clock inputs.

It is important to use the dedicated clock pins and routing for clock signal inputs to your design. The dedicated clock pins drive the clock network directly, ensuring lower skew than other I/O pins. Use the dedicated routing network to have a predictable delay with less skew for high fan-out signals. You can also use the clock pins and clock network to drive control signals like asynchronous reset.

Specific clock inputs connect to specific PLLs, which can drive specific low-skew routing networks. Analyze the global resource availability for each PLL and the PLL availability for each clock input pin.

Use the following descriptions to help determine which clock networks are appropriate for the clock signals in your design:

- The global clock (GCLKs) networks can drive throughout the entire device, serving as low-skew clock sources for device logic. This clock region has the maximum delay compared to other clock regions but allows the signal to reach everywhere within the device. This option is good for routing global reset/clear signals or routing clocks throughout the device.
- The regional clock (RCLK) networks only pertain to the quadrant they drive into. The RCLK networks provide the lowest clock delay and skew for logic contained within a single device quadrant.
- I/O elements (IOEs) and internal logic can also drive GCLKs and RCLKs to create internally generated global or regional clocks and other high fan-out control signals; for example, synchronous or asynchronous clears and clock enables.
- PLLs cannot be driven by internally-generated GCLKs or RCLKs. The input clock to the PLL must come from dedicated clock input pins or pin/PLL-fed GCLKs or RCLKs.
- Periphery clock (PCLK) networks are a collection of individual clock networks driven from the periphery of the Stratix III device. Clock outputs from the DPA block, horizontal I/O pins, and internal logic can drive the PCLK networks. These PCLKs have higher skew compared to GCLK and RCLK networks and can be used instead of general purpose routing to drive signals into and out of the Stratix III device.

For more information about these features, refer to the *Clock Networks and PLLs in Stratix III Devices* chapter in the *Stratix III Device Handbook*.

If your system requires more clock or control signals than are available in the target device, consider cases where the dedicated clock resource could be spared, particularly low-fanout and low-frequency signals where clock delay and clock skew do not have a significant impact on the design performance. Use the **Global Signal** assignment in the Quartus II Assignment Editor to select the type of global routing, or set the assignment to **Off** to specify that the signal should not use any global routing resources.

PLL Design Guidelines

Based on your system requirements, define the required clock frequencies for your FPGA design, and the input frequencies that will be available to the FPGA. Use these specifications to determine your PLL scheme. Use the Quartus II MegaWizard Plug-In Manager to enter your settings for the ALTPLL megafunction, and check the results to verify whether particular features and input/output frequencies can be implemented in a particular PLL.

For more information about setting up your timing constraints to work with the PLL, refer to *AN 471: High-Performance FPGA PLL Analysis with TimeQuest*.

Clock Control Block

Every global and regional clock network has its own clock control block. The control block provides the following features:

- Clock source selection (with dynamic selection for global clocks)
- Global clock multiplexing
- Clock power down (with static or dynamic clock enable or disable)

Use these features to select different clock input signals, or power-down clock networks to reduce power consumption, without using any combinational logic in your design. In Stratix III devices, the clock enable signals are supported at the clock network level instead of at the PLL output counter level, so you can turn off a clock even when a PLL is not being used.

For information about using the ALTCLKCTRL megafunction to set up the clock control block, refer to the *Clock Control Block Megafunction User Guide (ALTCLKCTRL)*.

PLL Features

Stratix III PLLs provide robust clock management and synthesis for device clock management, external system clock management, and high-speed I/O interfaces. Inherent jitter filtration and fine granularity control over multiply, divide ratios, and dynamic phase shift reconfiguration provide the high performance precision required in today's high-speed applications. Stratix III device PLLs are feature rich, supporting advanced capabilities such as clock switchover, dynamic phase shifting, PLL reconfiguration, and reconfigurable bandwidth. Stratix III PLLs also support external feedback mode, spread-spectrum tracking, and post-scale counter cascading features. All Stratix III PLLs have the same core analog structure with only minor differences in features that are supported. You can use some of the following additional features when planning your PLL design.

For information about designing your PLL and using the ALTPLL megafunction to take advantage of the features described in this section, refer to the *Phase-Locked Loops Megafunction User Guide (ALTPLL)*.

Clock Feedback Mode

Stratix III PLLs support up to six different clock feedback modes: Source-synchronous mode, No-compensation mode, Normal mode, Zero-delay buffer (ZDB) mode, External-feedback mode, and LVDS compensation. Each mode compensates for different clock networks and delays, so that the clocks are aligned differently. Choose the correct feedback mode for your application.

Clock Outputs

You can connect clock outputs to dedicated clock output pins or dedicated clock networks. PLLs on the top and bottom of the device provide more dedicated clock outputs than the left and right PLLs.

Clock Switchover

The clock switchover feature allows the PLL to switch between two reference input clocks. Use this feature for clock redundancy or for a dual-clock domain application such as in a system that turns on the redundant clock if the previous clock stops running. The design can perform clock switchover automatically when the clock is no longer toggling or based on a user control signal (`clkswitch`).

Dynamic Reconfiguration

You can reconfigure PLL settings to update the output-clock frequency and the PLL bandwidth and to phase-shift in real time, without reconfiguring the entire Stratix III device. The ability to reconfigure the PLL in real time is useful in applications that operate at multiple frequencies. It is also useful in prototyping environments, allowing you

to sweep PLL output frequencies and adjust the output-clock phase dynamically. You can also use this feature to adjust clock-to-out (t_{CO}) delays in real time by changing the PLL output clock phase shift. This approach eliminates the need to regenerate a configuration file with the new PLL settings.

Enable the dynamic reconfiguration feature with ALTPLL megafunction. You can then make the feature easier to use by instantiating the ALTPLL_RECONFIG megafunction.

For more information about the ALTPLL_RECONFIG MegaWizard Plug-In Manager, refer to the *Phase-Locked Loops Reconfiguration Megafunction Users Guide (ALTPLL_RECONFIG)*.

PLL Control Signals

You can use the following three signals to observe and control the PLL operation and resynchronization:

- `areset`—A reset or resynchronization input for each PLL. You should assert the `areset` signal every time the PLL loses lock to guarantee the correct phase relationship between the PLL input clock and output clocks. You can set up the PLL to automatically reset (self reset) upon a loss-of-lock condition. If the input clock to the PLL is not toggling or is unstable upon power up, assert the `areset` signal after the input clock is stable and within specifications. You should enable this control pin if you enable the dynamic reconfiguration or clock switchover feature.
- `locked`—This output pin for each PLL indicates that the PLL has locked onto the reference clock and the PLL clock outputs are operating at the desired phase and frequency. You can monitor this signal with device core logic.
- `pfdena`—Use this signal to maintain the most recent locked frequency, with some long-term drift to a lower frequency. You should enable this pin if your system requires a certain clock frequency from the PLL even if the input clock is disabled, so your system has time to store its current settings before shutting down.

Simultaneous Switching Noise

SSN becomes a concern when too many pins (in close proximity) change voltage levels at the same time. Analyze the design for possible simultaneous switching noise problems and consider the following recommendations:

- Reduce the number of pins that switch voltage levels at exactly the same time whenever possible.

- Use lower drive strengths for high-switching I/Os. The default drive strength setting might be higher than your design requires. Refer to "Stratix III I/O Features," on page 35.
- Use differential I/O standards and lower-voltage standards for high-switching I/Os.
- Reduce the number of simultaneously switching outputs within each bank. Spread output pins across multiple banks if possible.
- Spread switching I/Os evenly throughout the bank to reduce the number of aggressors in a given area to reduce SSN, when bank usage is substantially below 100%.
- Separate simultaneously-switching pins from input pins that are susceptible to SSN.
- Place important clock and asynchronous control signals near ground signals and away from large switching buses.
- Avoid using I/O pins one or two pins away from PLL power supply pins for high-switching or high drive strength pins.

Use staggered output delays to shift the output signals through time, or use adjustable slew rate settings. Refer to "Stratix III I/O Features," on page 35.

Design and Compilation

After you create your design source code and apply constraints including the device selection and timing requirements, your synthesis tool processes the code and maps it to elements of the device architecture. The Quartus II Fitter then performs placement and routing to implement the design elements in specific device resources. This section provides guidelines for these stages of the design and compilation flow.

Selecting a Synthesis Tool

The Quartus II software includes advanced and easy-to-use integrated synthesis that fully supports Verilog HDL and VHDL, as well as the Altera hardware description language (AHDL) and schematic design entry. You can also use industry-leading third-party EDA synthesis tools to synthesize your Verilog or VHDL design, and then compile the resulting output netlist file in the Quartus II software. Specify any third-party synthesis tool in the New Project Wizard or the **EDA Tools Settings** page of the **Settings** dialog box to use the correct Library Mapping File for your synthesis netlist.

Different synthesis tools can give different results. If you want to select the best-performing tool for your application, you can experiment by synthesizing typical designs for your application and coding style and comparing the results. Be sure to perform placement and routing in the Quartus II software to get accurate timing analysis and logic utilization results.

Altera recommends that you use the most recent version of third-party synthesis tools, because tool vendors are continuously adding new features, fixing tool issues, and enhancing performance for Altera devices.

Your synthesis tool might offer the capability to create a Quartus II project and pass constraints such as the EDA tool setting, device selection, and timing requirements that you specified in your synthesis project. You can use this capability to save time when setting up your Quartus II project for placement and routing.

For more information about supported synthesis tools, refer to the appropriate chapter in *Section III. Synthesis* in volume 1 of the *Quartus II Handbook*. The *Quartus II Release Notes* list the version of each synthesis tool that is officially supported by that version of the Quartus II software.

Design Recommendations and HDL Coding Styles

In the development of complex FPGA designs, design practices and coding styles have an enormous impact on your device's timing performance, logic utilization, and system reliability. Follow Altera's recommendations to achieve the best synthesis and fitting results.

Design Recommendations

Use synchronous design practices to consistently meet your design goals. Problems with other design techniques include reliance on propagation delays in a device, incomplete timing analysis, and possible glitches. In a synchronous design, a clock signal triggers all events. As long as all of the registers' timing requirements are met, a synchronous design behaves in a predictable and reliable manner for all process, voltage, and temperature (PVT) conditions. You can easily target synchronous designs to different device families or speed grades.

Pay particular attention to your clock signals because they have a large effect on your design's timing accuracy, performance and reliability. Problems with clock signals can cause functional and timing problems in your design. Use dedicated clock pins and clock routing for best results. For clock inversion, multiplication and division, use the device PLLs. For clock multiplexing and gating, use the dedicated clock control block or PLL clock switchover feature instead of combinational logic. Refer to "PLL Design Guidelines," on page 39. If you must use internally-generated clock signals, register the output of any combinational logic used as a clock signal to reduce glitches. For example, if you divide a clock using combinational logic, clock the final stage with the clock signal that was used to clock the divider circuit.

The Design Assistant in the Quartus II software is a design-rule checking tool that enables you to check for design issues early in the design flow. The Design Assistant checks your design for adherence to Altera-recommended design guidelines or design rules. To run the Design Assistant, on the Processing menu, point to Start and click **Start Design Assistant**. To set the Design Assistant to run automatically during compilation, turn on **Run Design Assistant during compilation** in the **Settings** dialog box. You can also use third-party "lint" tools to check your coding styles.

For more information about design recommendations and using the Design Assistant, refer to the *Design Recommendations for Altera Devices and the Quartus II Design Assistant* chapter in volume 1 of the *Quartus II Handbook*. You can also refer to industry papers for more information about multiple clock design. For a good analysis, refer to: www.sunburst-design.com/papers/CummingsSNUG2001SJ_AsyncClk.pdf

Using Megafunctions

Altera provides parameterizable megafunctions that are optimized for Altera device architectures. You can save design time by using megafunctions instead of coding your own logic. Additionally, the Altera-provided megafunctions may offer more efficient logic synthesis and device implementation. You can scale the megafunction's size and set various options with parameters. Megafunctions include the library of parameterized modules (LPM) and Altera device-specific megafunctions. You can also take advantage of Altera and third-party IP and reference designs to save design time, as described in "IP Selection," on page 13.

The Quartus II MegaWizard Plug-In Manager provides an easy user interface to customize megafunctions. You should build or change megafunction parameters using the wizard to ensure you set all ports and parameters correctly.

For detailed information about specific megafunctions, refer to Quartus II Help or the megafunction user guides on the User Guides Literature page.

Recommended HDL Coding Styles

HDL coding styles can have a significant effect on the quality of results (QoR) for programmable logic designs. Use Altera's recommended coding styles to achieve optimal synthesis results. When designing memory and digital system processing (DSP) functions, it is helpful to understand the device architecture so that you can take advantage of the

dedicated logic block sizes and configurations. Follow the coding guidelines for inferring megafunctions and targeting dedicated device hardware such as memory and DSP blocks.

For specific HDL coding examples and recommendations, refer to the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*. Refer to your synthesis tool's documentation for any additional tool-specific guidelines. In the Quartus II software, you can use the HDL examples in the Language Templates available from the right-click menu in the text editor.

Register Power-Up Levels and Control Signals

Stratix III devices support an optional chip-wide reset that enables you to override all clears on all device registers, including the registers of the memory blocks (but not the memory contents itself). When this `DEV_CLRn` pin is driven low, all registers are cleared or reset to 0. The following paragraphs describe situations when synthesis performs an optimization called NOT-gate-push back, in which case affected registers behave as though they are preset to a high value when `DEV_CLRn` is driven low. When the `DEV_CLRn` pin is driven high, all registers behave as programmed. To use this chip-wide reset, turn on **Enable device-wide reset (DEV_CLRn)** in the Quartus II software on the **General** tab of the **Device & Pin Options** dialog box before compiling your design.

Each Stratix III logic array block (LAB) also contains dedicated logic for driving register control signals to its ALMs. The control signals include three clocks, three clock enables, two asynchronous clears, a synchronous clear, and a synchronous load. Register control signals restrict how registers are packed into LABs because signals are shared within the LAB. It is important that control signals use the dedicated control signals in the device architecture, so in some cases you might be required to limit the number of different control signals used in your design.

For more information about LAB and ALM architecture, refer to the *Logic Array Blocks and Adaptive Logic Modules in Stratix III Devices* chapter in the *Stratix III Device Handbook*.

If the clock signal might not be available when reset is asserted, an asynchronous reset is typically used to reset the logic. The recommended reset architecture is to allow the reset signal to be asserted asynchronously and de-asserted synchronously. The source of the reset signal is then connected to the asynchronous port of the registers, which can be directly connected to global routing resources. The synchronous de-assertion allows all state machines and registers to start at the same

time. It also avoids the possibility that an asynchronous reset signal is released at or near the active clock edge of a flip-flop, in which case the output of the flip-flop could go to a metastable unknown state.

You can refer to industry papers for more information about reset design. For good analysis of reset architecture, refer to: www.sunburst-design.com/papers/CummingsSNUG2003Boston_Resets.pdf

Quartus II integrated synthesis by default enables the logic option called **Power-Up Don't Care**, which assumes your design does not depend on the power-up state of the device architecture and allows the software to remove registers that become stuck high. Other synthesis tools might use similar assumptions.

Designers typically use an explicit reset signal for the design that forces all registers into their appropriate values after reset but not necessarily at power-up. You can create your design such that the asynchronous reset allows the board to operate in a safe condition. You can then bring up the design with the reset active. Thus, you do not have to depend on the power-up conditions of the device.

If you force a particular power-up condition for your design, use the synthesis options available in your synthesis tool. In Quartus II integrated synthesis, you can apply the **Power-Up Level** logic option in the **Assignment Editor**, with a Tcl assignment, or create an `altera_attribute` assignment in your source code.

Some synthesis tools can also read the default or initial values for registered signals in your source code and implement this behavior in the device. For example, Quartus II integrated synthesis converts HDL default and initial values for registered signals into Power-Up Level settings. That way, the synthesized behavior matches the power-up state of the HDL code during a functional simulation.

The **Power-Up Level** option and the `altera_attribute` assignment are described in the *Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*.

Registers in the device core always power up to a low (0) logic level in the physical device architecture. If you specify a high power-up level or a non-zero reset value (often called a preset signal), synthesis tools typically use the clear signals available on the registers and perform an optimization referred to as NOT-gate push back. NOT-gate push back adds an inverter to the input and the output of the register. The register hardware actually powers up and resets low, but the register output is inverted so the result at all destinations is a high logic value. If synthesis performs a NOT-gate push back optimization, the register behaves like a

high (1) logic level during reset or power-up conditions. Regular register operation is not affected because the signal is inverted twice in the regular data path. This optimization does not negatively affect the fitting or performance of your design, but note that if you tap the register during on-chip verification, or view it during simulation, you should check the signal after the output inversion to obtain the correct value.

If you assign a high power-up level to a register that is reset low, or assign a low power-up value to a register that is preset high, synthesis tools cannot use the NOT-gate push back optimization technique and may ignore the power-up conditions.

Do not apply both a reset and preset signal to the same register. To implement this type of logic, synthesis tools emulate the controls with logic and latches that can be prone to glitches because of the different delays between the different paths to the register. In addition, the power-up value is undefined for these registers.

Planning for Hierarchical and Team-Based Design

The Quartus II incremental compilation feature preserves the results and performance for unchanged logic in your design as you make changes elsewhere, allowing you to perform more design iterations and achieve timing closure more efficiently. In an incremental compilation flow, the system architect splits a large design into smaller partitions that can be designed separately. In a team design environment, team members can work on partitions independently, which simplifies the design process and reduces compilation time. Partitioning your design is optional, but these benefits are important for large Stratix III designs.

The bottom-up design flows are not supported for HardCopy migrations. If you migrate to a HardCopy III ASIC, use the top-down incremental compilation methodology with a consistent set of global assignments for all design blocks, so the same assignments can be recreated in the HardCopy revision.

If you take advantage of the compilation-time savings and performance preservation of Quartus II incremental compilation, plan for an incremental compilation flow from the beginning of your design cycle. Good partition and floorplan design helps lower-level design blocks meet top-level design requirements, reducing the time spent integrating and verifying the timing of the top-level design.

For more information about using the incremental compilation flows in the Quartus II software, as well as important guidelines for creating design partitions and a design floorplan, refer to the *Quartus II Incremental Compilation for Hierarchical and Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*.

Planning Design Partitions

Partitioning a design for an FPGA requires planning to ensure optimal results when the partitions are integrated, and ensure that each partition is placed well, relative to other partitions in the device.

Follow Altera's recommendations for creating design partitions to improve the overall QoR. For example, registering partition I/O boundaries keeps critical timing paths inside one partition that can be optimized independently. When the design partitions are specified, you can use the **Incremental Compilation Advisor** to ensure that partitions meet Altera's recommendations.

Plan your source code so that each design block is defined in a separate file. This allows the software to automatically detect changes to each block separately. If you use a third-party synthesis tool, create separate VQM or EDIF netlists for each design partition in your synthesis tool. If necessary, create separate projects within your synthesis tool so that the tool synthesizes each partition separately and generates separate output netlist files. Refer to your synthesis tool documentation for information about support for Quartus II incremental compilation. Use hierarchy in your design to provide more flexibility when partitioning. Keep your design logic in the leaves of the hierarchy trees; that is, the top level of the hierarchy should have very little logic, and the lower-level design blocks contain the logic.

Timing Budget

Determining a timing budget before designers develop their individual blocks reduces the chance of timing problems during system integration, especially in team-based design. If you optimize lower-level partitions separately, any unregistered paths that cross between partitions are not optimized as an entire path. To ensure that the software correctly optimizes the input and output logic in each partition, you may be required to perform some manual timing budgeting. For each unregistered timing path that crosses between partitions, make timing assignments on the corresponding I/O path in each partition to constrain both ends of the path to the budgeted timing delay. The software optimizes paths appropriately so they meet the top-level design requirements when you assign a timing budget for each part of the connection.

Resource Balancing

It is often important to plan and balance resource utilization. When performing incremental compilation, the software synthesizes each partition separately with no data about the resources used in other partitions. Therefore, synthesis can overuse device resources in the individual partitions, and the design may not fit in the target device when the partitions are merged. In some cases, partitions use conflicting resources when combined at the top level. Device resources include logic, memory, and multipliers, as well as PLLs and global routing signals such as clocks. Balancing and allocating resource utilization among the design partitions avoids any problems with conflicting resources when all the partitions are integrated.

Planning in Bottom-Up and Team-Based Flows

In bottom-up design flows, it is important that the system architect provide guidance to designers of lower-level blocks to ensure that each partition uses the appropriate device resources. Because the designs are developed independently, each lower-level designer has no information about the overall design or how their partition connects with other partitions. This lack of information can lead to problems during system integration. The top-level project information, including pin locations, physical constraints, and timing requirements, should be communicated to the designers of lower-level partitions before they start their design.

The system architect can plan design partitions at the top level and use Quartus II incremental compilation to communicate information to lower-level designers through automatically-generated scripts. The Quartus II software **Generate bottom-up design partition scripts** option automates the process of transferring top-level project information to lower-level modules. The software provides a project manager interface for managing project information in the top-level design.

Creating a Design Floorplan

To take full advantage of incremental compilation, you can create a design floorplan to avoid conflicts between design partitions, and to ensure that each partition is placed well relative to other partitions. When you create different location assignments for each partition, no location conflicts can occur. In addition, a design floorplan helps avoid situations in which the Fitter is directed to place or replace a portion of the design in an area of the device where most resources have already been claimed. Floorplan assignments are recommended for timing-critical partitions in top-down flows.

You can use the Quartus II Chip Planner to create a design floorplan using LogicLock region assignments for each design partition. With a basic design framework for the top-level design, the floorplan editor enables

you to view connections between regions, estimate physical timing delays on the chip, and move regions around the device floorplan. When you have compiled the full design, you can also view logic placement and locate areas of routing congestion to improve the floorplan assignments.

For more information about creating placement assignments in the design floorplan, refer to the *Analyzing and Optimizing the Design Floorplan* chapter in volume 2 of the *Quartus II Handbook*.

SOPC Builder

SOPC Builder is a powerful system development tool for creating systems based on processors, peripherals, and memories. SOPC Builder is an optional tool that enables you to define and generate a complete system-on-a-programmable-chip (SOPC) in much less time than using traditional, manual integration methods. With SOPC Builder, you specify the system components in a GUI, and SOPC Builder generates the interconnect logic automatically. SOPC Builder outputs HDL files that define all components of the system, and a top-level HDL design file that connects all the components together.

SOPC Builder is commonly used as the tool for creating systems based on the Nios II processor. However, SOPC Builder is a general-purpose tool for creating arbitrary SOPC designs that may or may not contain a processor. SOPC Builder components use Avalon interfaces for the physical connection of components, and you can use SOPC Builder to connect any logical device (either on-chip or off-chip) that has an Avalon interface. The Avalon Memory-Mapped interface uses an address-mapped read/write protocol that enables flexible topologies for connecting master components to read and/or write any slave components. The Avalon Streaming interface is a high-speed, unidirectional, system interconnect that enables point-to-point connections between streaming components that send and receive data using source and sink ports.

In addition to its role as a hardware generation tool, SOPC Builder also serves as the starting point for system simulation and embedded software creation. SOPC Builder provides features to ease writing software and to accelerate system simulation.

For information about using SOPC builder to improve your productivity, refer to *Volume 4: SOPC Builder* of the *Quartus II Handbook*.

Timing Closure and Verification

After compiling the completed logic design, check the device utilization and verify that the design meets its timing requirements. Analyze the messages generated during compilation to check for any potential

problems. If required, you can use the Quartus II software to optimize the design's resource utilization and achieve timing closure, preserve the performance of unchanged design blocks, and reduce compilation time for future iterations. You can also verify the design functionality with simulation or formal verification. This section provides guidelines for these stages of the compilation flow.

Device Resource Utilization Reports

After compilation, review the device resource utilization information to determine whether the future addition of extra logic or other design changes will introduce fitting difficulties. If your compilation results in a no-fit error, resource utilization information is important so you can analyze the fitting problems in your design.

To determine resource usage, refer to the **Flow Summary** section of the Compilation Report for a percentage representing the total logic utilization, which includes an estimation of resources that cannot be used due to existing connections or logic use.

For Stratix III devices, a device with low logic utilization does not have the lowest ALM utilization possible. In addition, a design that is reported as close to 100% full might still have space for extra logic. The Fitter uses ALUTs in different ALMs, even when the logic can be placed within one ALM, so that it can achieve the best timing and routability results. Logic might be spread throughout the device when achieving these results. As the device fills up, the Fitter automatically searches for logic that can be placed together in one ALM.

More detailed resource information is available by viewing the reports under **Resource Section** in the **Fitter** section of the Compilation Report. The **Fitter Resource Usage Summary** report breaks down the logic utilization information and indicates the number of fully and partially used ALMs, and provides other resource information including the number of bits in each type of memory block. There are also reports that describe some of the optimizations that occurred during compilation. For example, if you are using Quartus II integrated synthesis, the reports under the **Optimization Results** folder in the **Analysis & Synthesis** section describe information including registers that were removed during synthesis. This report can be useful when estimating device resource utilization for a partial design, to ensure that registers were not removed due to missing connections with other parts of the design.

Quartus II Messages

Each stage of the compilation flow generates messages, including informational notes, warnings, and critical warnings. Review these messages to check for any design problems. Ensure that you understand the significance of any warning messages, and make changes to the design or settings if required. In the Quartus II user interface, you can use the Message window tabs to look at only certain types of messages, and you can suppress messages if you have determined that they do not require any action from you.

For more information about messages and message suppression, refer to the *Managing Quartus II Projects* chapter in volume 2 of the *Quartus II Handbook*.

Timing Constraints and Analysis

In an FPGA design flow, accurate timing constraints allow timing-driven synthesis software and place-and-route software to obtain optimal results. Timing constraints are critical to ensure designs meet their timing requirements, which represent actual design requirements that must be met for the device to operate correctly. The final programmed device might not operate as expected if the timing paths are not fully analyzed and verified to meet requirements.

The Quartus II software includes the Quartus II TimeQuest Timing Analyzer, a powerful ASIC-style timing analysis tool that validates the timing performance of all logic in your design. It supports the industry-standard Synopsys Design Constraints (SDC) format timing constraints, and has an easy-to-use GUI with interactive timing reports. It is ideal for constraining high-speed source-synchronous interfaces and clock multiplexing design structures. (For legacy designs, the Quartus II software also includes the Classic Timing Analyzer, which uses different design constraints and reports. Use the TimeQuest Timing Analyzer for Stratix III and HardCopy III designs.)

The software also supports static timing analysis in the industry-standard Synopsys Primetime software. Specify the tool in the New Project Wizard or the **EDA Tools Settings** page of the **Settings** dialog box to generate the required timing netlist.

A comprehensive static timing analysis includes analysis of register-to-register, I/O, and asynchronous reset paths. It is important to specify the frequencies and relationships for all clocks in your design. Use input and output delay constraints to specify external device or board timing parameters. Specify accurate timing requirements for external interfacing components to reflect the exact system intent. The Timing

Analyzer performs static timing analysis on the entire system, using data required times, data arrival times, and clock arrival times to verify circuit performance and detect possible timing violations. It determines the timing relationships that must be met for the design to correctly function.

Ensure that the input I/O times are not violated when data is provided to the Stratix III device. You can use the `report_datasheet` command to generate a datasheet report that summarizes the I/O timing characteristics of the entire design.

For more information about timing analysis, refer to the *Quartus II TimeQuest Timing Analyzer* and *Synopsys PrimeTime Support* chapters in volume 3 of the *Quartus II Handbook*.

Recommended Timing Optimization and Analysis Assignments

The assignments and settings described in this section are not turned on in the software by default for all designs, but are important for large designs such as those in Stratix III devices.

Turn on **Optimize fast-corner timing** on the **Fitter Settings** page in the **Settings** dialog box. When this option is on, the design is optimized to meet its timing requirements at the Fast Timing process corner and operating condition, as well as at the Slow Timing corner. Therefore, turning on this option helps create a design implementation that is more robust across process, temperature, and voltage variations.

If your design has problems meeting hold times, there may be design problems such as gated clocks adding delay in your clock lines. Otherwise, set the **Optimize hold timing** option to **All paths** to optimize the register-to-register hold time paths.

Turn on **Enable multicorner timing analysis** on the **TimeQuest Timing Analyzer** page under **Timing Analysis Settings** in the **Settings** dialog box, or use the `--multicorner` command line option for TimeQuest. This option directs the TimeQuest Timing Analyzer to analyze the design and generate slack reports for the slow and fast corners.

In your TimeQuest SDC constraints file, use the following recommended constraints as applicable to your design:

- `create_clock`, `create_generated_clock`—Specify the frequencies and relationships for all clocks in your design.
- `derive_pll_clocks`—Create generated clocks for all PLL outputs, according to the settings in the PLL megafunctions. Specify multicycle relationships for LVDS transmitters or receiver deserialization factors.

- `set_input_delay`, `set_output_delay`—Specify external device or board timing parameters.
- `derive_clock_uncertainty`—Automatically apply inter-clock, intra-clock, and I/O interface uncertainties.
- `check_timing`—Generate a report on any problem with the design or applied constraints, including missing constraints.

Area and Timing Optimization

This section highlights some of the features offered in the Quartus II software to help optimize area (or resource utilization) and timing performance. If timing analysis reports that your design requirements were not met, you must make changes to your design or settings and recompile the design to achieve timing closure. If your compilation results in no-fit messages, you must make changes to get a successful placement and routing.

For information about additional optimization features, refer to the *Area and Timing Optimization* chapter in volume 2 of the *Quartus II Handbook*.

You can use the Early Timing Estimation feature to estimate your design's timing results before the software performs full placement and routing. On the Processing menu, point to Start and click **Start Early Timing Estimate** to generate initial compilation results after you have run analysis and synthesis. Using this feature provides a timing estimate up to 45× faster than running a full compilation. The fit is not fully optimized or routed; therefore, timing analysis reports are only estimates. On average, the estimated delays are within 11% of those achieved by a full compilation compared to the final timing results.

Physical synthesis optimizations make placement-specific changes to the netlist that improve results for a specific Altera device. You can specify **Physical synthesis for performance** or **Physical synthesis for fitting** options. These options typically increase compilation time significantly but can provide significant improvements to the QoR with push-button optimizations. If you turn on these options, ensure that they do improve the results for your design. If you do not require these options to meet your design timing requirements, turn off the options to reduce compilation time.

For more information, refer to the *Netlist Optimizations and Physical Synthesis* chapter in volume 2 of the *Quartus II Handbook*.

The Design Space Explorer (DSE) is a utility that automates the process of finding the optimal collection of Quartus II software settings for your design. The **Search for Best Performance** and **Search for Best Area** options under **Exploration Settings** use a predefined exploration space to

target design performance or area improvements with multiple compilations. You can also set the **Optimization Goal** to **Optimize for Speed** or **Optimize for Area** using the **Advanced** tab in the DSE window.

For more information, refer to the *Design Space Explorer* chapter in volume 2 of the *Quartus II Handbook*.

The Optimization Advisors provide guidance in making settings that optimize your design. On the Tools menu, point to Advisors, and click **Resource Optimization Advisor**, or **Timing Optimization Advisor**. Evaluate the options and choose the settings that best suit the requirements.

Preserving Performance

You can use the incremental compilation feature to preserve unchanged parts of your design, thus preserving performance and allowing you to reach timing closure more efficiently. For guidelines and references, refer to "Planning for Hierarchical and Team-Based Design," on page 47.

Reducing Compilation Time

You can speed up design iteration time by an average of 60% when making changes to the design with the incremental compilation feature. For guidelines and references, refer to "Planning for Hierarchical and Team-Based Design," on page 47.

The Quartus II software can run some algorithms in parallel to take advantage of multiple processors and reduce compilation time when more than one processor is available to compile the design. To set the number of processors available for a Quartus II compilation, specify the **Maximum processors allows for parallel compilation** on the **Compilation Process Settings** page of the **Settings** dialog box. The default value for the number of processors is 1, which disables parallel compilation.

The Compilation Time Advisor provides guidance in making settings that reduce your design compilation time. On the Tools menu, point to Advisors, and click **Compilation Time Advisor**. Using some of these techniques to reduce compilation time can reduce the overall QoR.

For more suggestions, refer to the *Area and Timing Optimization* chapter in volume 2 of the *Quartus II Handbook*.

Simulation

The Quartus II software supports both functional and gate-level timing simulations. Perform functional simulation at the beginning of your design flow to check the design functionality or logical behavior of each design block. You do not have to fully compile your design; you can generate a functional simulation netlist that does not contain timing information. Timing simulation uses the timing netlist generated by the Timing Analyzer, including the delay of different device blocks and the placement and routing information. You can perform timing simulation for the top-level design at the end of your design flow to ensure that your design works in the targeted device.

You can use the built-in Quartus II Simulator to perform quick and easy simulations. To debug your design with the Quartus II Simulator, you can set breakpoints to capture certain output conditions, and you can compare your current simulation results against the previous results. Altera also provides the ModelSim-Altera simulator with Quartus II license subscriptions, which enables you to take advantage of advanced testbench capabilities and other features. In addition, the Quartus II EDA Netlist Writer can generate timing netlist files to support other third-party simulation tools such as Synopsys VCS and Cadence NC-Sim. Specify your simulation tool in the **EDA Tools Settings** page of the **Settings** dialog box to generate the appropriate output simulation netlist.

If you use a third-party simulation tool, use the software version that is supported with your Quartus II version. The *Quartus II Release Notes* lists the version of each simulation tool that is officially supported with that particular version of the Quartus II software. Use the model libraries provided with your Quartus II software version because libraries can change between versions, which might cause a mismatch with your simulation netlist.

For more information about simulation tool flows, refer to the appropriate chapter in *Section I. Simulation* in volume 3 of the *Quartus II Handbook*.

Formal Verification

The Quartus II software supports some formal verification flows. Consider whether your desired formal verification flow impacts the design and compilation stages of your design.

Using a formal verification flow can impact performance results because it requires that certain logic optimizations be turned off, such as register retiming, and forces hierarchy blocks to be preserved, which can restrict optimization. Formal verification treats memory blocks as black boxes.

Therefore, it is best to keep memory in a separate hierarchy block so that other logic does not get incorporated into the black box for verification. There are other restrictions that can also limit your design, so consult the documentation for details. If formal verification is important to your design, it is easier to plan for limitations and restrictions in the beginning than to make changes later in the design flow.

The *Quartus II Release Notes* list the version of each formal verification tool that is officially supported with that particular version of the Quartus II software. Specify your formal verification tool in the **EDA Tools Settings** page of the **Settings** dialog box to generate the appropriate output netlist.

For more information about formal verification flows, refer to the appropriate chapter in *Section VI. Formal Verification* in volume 3 of the *Quartus II Handbook*.

Power Analysis and Optimization

Stratix III devices utilize advanced process and circuit techniques, along with major circuit and architecture innovations, to minimize power and deliver high performance. The Quartus II software optimizes and analyzes the design using power models specific to the selected core voltage (0.9-V or 1.1-V). The Programmable Power Technology feature enables every programmable LAB, DSP block, and memory block to deliver either high speed or low power, depending on your design requirements. The Quartus II software automatically takes advantage of the excess slack found on non-critical design paths to minimize power consumption while maintaining high performance for critical paths.

The software offers additional power-optimization techniques to further reduce your power consumption, as described in this section. This section also describes the power analysis tools that enable you to estimate device power consumption and heat dissipation from early design concept through design implementation, and how you can use the Stratix III temperature-sensing diode for thermal management.

For specific architectural features that help reduce power consumption, as well as design optimization techniques, refer to *AN 437: Power Optimization in Stratix III FPGAs*.

Power Analysis

Before design completion, estimate power consumption using a spreadsheet as described in "Early Power Estimation," on page 16. After compiling your design, analyze the power consumption and heat dissipation with the Quartus II PowerPlay Power Analyzer to ensure the design has not violated power supply and thermal budgets.

You must compile a design (to provide information about design resources, placement and routing, and I/O standards) and provide signal activity data (toggle rates and static probabilities) to use the PowerPlay Power Analyzer. You can derive signal activity data from simulation results or a user-defined default toggle rate and vectorless estimation. The signal activities used for the analysis must be representative of the actual operating behavior. For the most accurate power estimation, use gate-level simulation results with a VCD output file from the Quartus II Simulator or a third-party simulation tool. The simulation activity should include typical input vectors over a realistic time period, not the corner cases often used during functional verification. Use the recommended simulator settings (such as glitch filtering) to ensure good results.

You must also specify operating conditions, including core voltage, device power characteristic, ambient and junction temperature, cooling solution and board thermal model. Select the appropriate settings on the **Operating Conditions** page in the **Settings** dialog box.

To calculate the dynamic, static, and I/O thermal power consumption, from the Processing menu, click **PowerPlay Power Analyzer Tool**. The tool also provides a summary of the signal activities used for analysis and a confidence metric that reflects the overall quality of the data sources for the signal activities. Note that the report is a power estimate based on the data provided, and is not a power specification. Always refer to the data sheet for your device.

For more information about power analysis, and recommendations for simulation settings for creating signal activity information, refer to the *PowerPlay Power Analyzer* chapter in volume 3 of the *Quartus II Handbook*. For more information about Signal Activity Files (**.saf**) and how to create them, refer to the *Quartus II Simulator* chapter in volume 3 of the *Quartus II Handbook*. For recommendations and guidelines for power management system design, including creating a power budget spreadsheet, refer to *AN 448: Stratix III Power Management Design Guide*.

Power Optimization

To reduce dynamic power consumption in Stratix III devices, you can use various design and software techniques to optimize your design.

Power optimization in the Quartus II software depends on accurate power analysis results. Use the guidelines in the previous section to ensure the software optimizes the power utilization correctly for the design's operating behavior and conditions.

Device and Design Power Optimization Techniques

This section lists several design techniques that can reduce power consumption. The results of these techniques are different from design to design.

For more details and additional design techniques to reduce power consumption, refer to the *Power Optimization* chapter in volume 2 of the *Quartus II Handbook* and *AN 437: Power Optimization in Stratix III FPGAs*.

Device Speed Grade
When your design includes a lot of critical timing paths that require the high-performance mode, you may be able to reduce power consumption by using a faster speed grade device if available. With a faster device, the software may be able to set more device tiles to use the low-power mode.

Clock Power Management
Clocks represent a significant portion of dynamic power consumption, due to their high switching activity and long paths. The Quartus II software automatically optimizes clock routing power by enabling only the portions of a clock network that are required to feed downstream registers. You can also use clock control blocks to dynamically enable or disable the clock network. When a clock network is powered down, all the logic fed by that clock network does not toggle, thereby reducing the overall power consumption of the device.

For more information about using clock control blocks, refer to the *Clock Control Block Megafunction User Guide (ALTCLKCTRL)*.

To reduce LAB-wide clock power consumption without disabling the entire clock tree, use the LAB-wide clock enable signal to gate the LAB-wide clock. The Quartus II software automatically promotes register-level clock enable signals to the LAB level.

Memory Power Reduction
The key to reducing memory power consumption is to reduce the number of memory clocking events. You can use clock gating described in "Clock Power Management," on page 59 or the clock enable signals in the memory ports.

I/O Power Guidelines
The dynamic power consumed in the I/O buffer is proportional to the total load capacitance, so lower capacitance reduces power consumption.

Non-terminated I/O standards such as LVTTL and LVCMOS have a rail-to-rail output swing equal to the V_{CCIO} supply voltage. Because dynamic power is proportional to the square of the voltage, use lower voltage I/O standards to reduce dynamic power. These I/O standards consume little static power.

Because dynamic power is also proportional to the output transition frequency, for high-frequency applications, use resistively-terminated I/O standard such as SSTL. The output load voltage swings by an amount smaller than V_{CCIO} around some bias point, so dynamic power is lower than for non-terminated I/O under similar conditions.

Resistively-terminated I/O standards dissipate significant static power because current is constantly driven into the termination network. Use the lowest drive strength that meets your speed and waveform requirements to minimize static power when using resistively terminated I/O standards. Note that the power used by external devices is not included in the PowerPlay calculations, so be sure to include it separately in your system power calculations.

Quartus II Power Optimization Techniques

The Quartus II software offers power-optimized synthesis and fitting to reduce core dynamic power. Power-driven compilation works in conjunction with Programmable Power Technology in the Stratix III silicon. The default setting is **Normal compilation**. You can choose **Extra effort** for additional power optimizations that might impact the design's maximum achievable performance. Under the **Analysis and Synthesis Settings** page and the **Fitter Settings** page of the **Settings** dialog box, click **PowerPlay power optimization**.

Optimizing your design for area saves power because fewer logic blocks are used, and therefore there is typically less switching activity. Improving your design source code to optimize for performance can also reduce power usage because more of the design may be placed using low-power tiles instead of requiring the high-performance mode. You can use the DSE and Power Optimization Advisor to provide additional suggestions to reduce power.

For the information about power-driven compilation and the Power Optimization Advisor, refer to the *Power Optimization* chapter in volume 2 of the *Quartus II Handbook*.

DSE

The DSE is a utility that automates the process of finding the optimal collection of Quartus II software settings for your design. The **Search for Lowest Power** option under **Exploration Settings** uses a predefined

exploration space to target overall design power improvements with multiple compilations. You can also set the **Optimization Goal** to **Optimize for Power** using the **Advanced** tab in the DSE window.

For more information, refer to the *Design Space Explorer* chapter in volume 2 of the *Quartus II Handbook*.

Power Optimization Advisor

The Quartus II software includes the Power Optimization Advisor, which provides specific power optimization advice and recommendations based on the current design project settings and assignments. From the Tools menu, point to **Advisors** and click **Power Optimization Advisor**. After making any of the recommended changes, recompile your design and run the Power Play Power Analyzer to check the change in your power results.

Thermal Management

After your design is optimized for power and you have the final power analysis numbers, check that the board cooling solution is sufficient to dissipate the heat generated by the device power consumption. Calculating or measuring the junction temperature is crucial for thermal management. Historically, junction temperature is calculated using ambient or case temperature, junction-to-ambient (θ_{JA}) or junction-to-case (θ_{JC}) thermal resistance, and the device power consumption.

Stratix III devices include a temperature sensing diode (TSD) with embedded analog-to-digital converter (ADC) circuitry, so you do not require an external temperature sensing chip on the board. The Stratix III TSD can self-monitor the device junction temperature and be used with external circuitry for activities such as controlling air flow to the FPGA. You can bypass the ADC if you want to use an external temperature sensor, similar to the solution used for Stratix II device or other devices.

You must include the TSD circuitry in your design if you want to use it. Ensure you make the correct external pin connections, whether you use both the ADC and TSD, or you bypass the ADC and connect the sensing diode to an external temperature sensor.

For more information about these features, refer to the *Programmable Power and Temperature Sensing Diode in Stratix III Devices* chapter in the *Stratix III Device Handbook*.

Conclusion

The design guidelines in this document provide important factors to consider in high-density, high-performance Stratix III designs. It is important to follow Altera's recommendations throughout the design process to achieve good results, avoid common issues, and improve your design productivity. You can use the "Design Checklist," on page 65 to ensure that you have reviewed all the guidelines before completing your Stratix III design.

Referenced Documents

This application note references the following documents:

- *AN 370: Using the Serial FlashLoader with the Quartus II Software*
- *AN 386: Using the Parallel Flash Loader with the Quartus II Software*
- *AN 437: Power Optimization in Stratix III FPGAs*
- *AN 474: Implementing Stratix III Programmable I/O Delay Settings in the Quartus II Software*
- *AN 448: Stratix III Power Management Design Guide*
- *Analyzing and Optimizing the Design Floorplan* chapter in volume 2 of the *Quartus II Handbook*
- *Area and Timing Optimization* chapter in volume 2 of the *Quartus II Handbook*
- *ByteBlaster II Download Cable User Guide*
- *Cadence PCB Design Tools Support* chapter in volume 2 of the *Quartus II Handbook*
- *Clock Control Block Megafunction User Guide (ALTCLKCTRL)*
- *Clock Networks and PLLs in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*
- *Configuring Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*
- *DC and Switching Characteristics of Stratix III Devices* chapter in volume 2 of the *Stratix III Device Handbook*
- *Design Debugging Using the SignalTap II Embedded Logic Analyzer* chapter in volume 3 of the *Quartus II Handbook*
- *Design Debugging Using In-System Sources and Probes* chapter in volume 3 of the *Quartus II Handbook*
- *Design Recommendations for Altera Devices and the Quartus II Design Assistant* chapter in volume 1 of the *Quartus II Handbook*
- *Design Security in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*
- *Design Space Explorer* chapter in volume 2 of the *Quartus II Handbook*
- *Enhanced Configuration Devices Data Sheet* in volume 2 of the *Configuration Handbook*
- *EthernetBlaster Download Cable User Guide*
- *External Memory Interfaces in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*
- *HardCopy Series Handbook*
- *Hot Socketing & Power-On Reset in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*

- *IEEE 1149.1 (JTAG) Boundary Scan Testing in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*
- *I/O Management* chapter in volume 2 of the *Quartus II Handbook*
- *In-System Debugging Using External Logic Analyzers* chapter in volume 3 of the *Quartus II Handbook*
- *In-System Updating of Memory and Constants* chapter in volume 3 of the *Quartus II Handbook*
- *Logic Array Blocks and Adaptive Logic Modules in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*
- *Managing Quartus II Projects* chapter in volume 2 of the *Quartus II Handbook*
- *Mentor Graphics PCB Design Tools Support* chapters in volume 2 of the *Quartus II Handbook*
- *Netlist Optimizations and Physical Synthesis* chapter in volume 2 of the *Quartus II Handbook*
- *Phase-Locked Loops Megafunction User Guide (ALTPLL)*
- *Phase-Locked Loops Reconfiguration Megafunction User Guide (ALTPLL_RECONFIG)*
- *Power Optimization* chapter in volume 2 of the *Quartus II Handbook*
- *Programmable Power and Temperature Sensing Diode in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*
- *PowerPlay Early Power Estimator User Guide For Stratix III FPGAs*
- *PowerPlay Power Analyzer* chapter in volume 3 of the *Quartus II Handbook*
- *Quartus II Incremental Compilation for Hierarchical and Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*
- *Quartus II Programmer* chapter in volume 3 of the *Quartus II Handbook*
- *Quartus II Simulator* chapter in volume 3 of the *Quartus II Handbook*
- *Quick Design Debugging Using SignalProbe* chapter in volume 3 of the *Quartus II Handbook*
- *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*
- *Remote System Upgrades with Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*
- *Remote Update Circuitry Megafunction User Guide (ALTREMOTE_UPDATE)*
- *Section VI. Formal Verification* in volume 3 of the *Quartus II Handbook*
- *SEU Mitigation in Stratix III Devices* chapter in the *Stratix III Device Handbook*
- *Serial Configuration Devices Data Sheet* in volume 2 of the *Configuration Handbook*
- *Signal Integrity with Third-Party Tools* chapter in volume 3 of the *Quartus II Handbook*
- *Section I. Simulation* in volume 3 of the *Quartus II Handbook*
- *sld_virtual_jtag Megafunction User Guide*
- *Stratix III Device Family Overview* chapter in volume 1 of the *Stratix III Device Handbook*

- *Stratix III Device I/O Features* chapter in volume 1 of the *Stratix III Device Handbook*
- Stratix III Device Pinout Files
- *Stratix III Pin Connection Guidelines*
- *Design Security in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*
- *Stratix III FPGA Signal Integrity* white paper
- *Section III. Synthesis* in volume 1 of the *Quartus II Handbook*
- *Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*
- *Synopsys PrimeTime Support* chapter in volume 3 of the *Quartus II Handbook*
- *USB Blaster II USB Download Cable User Guide*
- *Altera Enhanced Configuration Devices* chapters in volume 2 of the *Configuration Handbook*
- *Volume 4: SOPC Builder* of the *Quartus II Handbook*

Document Revision History

Table 3 shows the revision history for this document.

Table 3. Revision History

Date and Document Version	Changes Made	Summary of Changes
May 2008 v1.1	• Updated "Device Selection," on page 2. • Added "HardCopy III ASIC Migration," on page 4 • Updated "Planning for On-Chip Debugging," on page 13. • Updated "Decoupling Capacitors," on page 20. • Updated "Simultaneous Switching Noise," on page 26. • Updated "Register Power-Up Levels and Control Signals," on page 45. • Updated "Planning for Hierarchical and Team-Based Design," on page 47. • Updated "Planning Design Partitions," on page 48. • Updated "Device Resource Utilization Reports," on page 51. • Updated "Referenced Documents," on page 62. • Updated tasks 27, 29, 81, and 98, and added task 3 in "Design Checklist," on page 65.	Updated document to add support for HardCopy III devices. Also, updated reference document links and other minor updates, such as 3.3-V I/O, and information about the PDN tool for decoupling capacitors.
July 2007 v1.0	Initial Release	—

Design Checklist

This checklist provides a summary of the guidelines described in this document. Use the checklist to verify that you have followed the guidelines for each stage of your design.

Project Name:
Date:

"Device Selection," on page 2

	Done	N/A	
1	☐	☐	Select a device based on logic/memory/multiplier density, device features such as PLLs I/O pin count, package offering, and reserve resources for future development.
2	☐	☐	Consider vertical device migration requirements.
3	☐	☐	If you migrate to a HardCopy III ASIC, review the appropriate design considerations.
4	☐	☐	Consider availability of speed grades and selectable core voltage.

"Planning for Device Configuration," on page 6

	Done	N/A	
5	☐	☐	Select a configuration scheme and plan companion devices and board connections.
6	☐	☐	Check that the configuration device supports the configuration bitstream file size for your Stratix III device.
7	☐	☐	Ensure your configuration scheme supports any required features: data decompression, design security, and remote upgrades.
8	☐	☐	If you want to use a flash device for the PFL, check the list of supported devices.
9	☐	☐	Enable optional configuration pins `CLKUSR` and `INIT_DONE`, as required.
10	☐	☐	Use option to Auto-restart after configuration error.
11	☐	☐	Use Convert Programming Files to generate files in different formats or for configuration chains.

"Early System Planning," on page 12

	Done	N/A	
12	☐	☐	Create detailed design specifications, and a test plan if appropriate.
13	☐	☐	Select IP that affects system design, especially I/O interfaces.

14	☐	☐	Take advantage of on-chip debugging features to analyze internal signals and perform advanced debugging techniques.
15	☐	☐	Select on-chip debugging scheme(s) early to plan memory and logic requirements, I/O pin connections, and board connections.
16	☐	☐	If you want to use tethered OpenCore Plus or JTAG debugging features, design the board for JTAG connections.
17	☐	☐	Plan space for any additional device resources used by debugging features.
18	☐	☐	Reserve I/O pins for debugging.
19	☐	☐	Ensure board supports debugging mode, incorporate headers or connectors as required.
20	☐	☐	Ensure incremental compilation is turned on for incremental debugging.
21	☐	☐	Incorporate debugging megafunctions in your design if required.
22	☐	☐	Estimate power consumption with the Early Power Estimator spreadsheet to plan the cooling solution and power supplies.

"Board Design Considerations," on page 18

	Done	N/A	
23	☐	☐	Design voltage supply power ramps to be monotonic.
24	☐	☐	Design board for power-up: Stratix III output buffers are tri-stated until device is configured, and configuration pins drive out.
25	☐	☐	Set POR time to ensure power supplies are stable.
26	☐	☐	Design power sequencing and voltage regulators for best device reliability.
27	☐	☐	Connect all power pins correctly: 0.9-V or 1.1-V core, 1.1-V PLL and periphery, 1.8-V, 2.5-V, 3.0-V, 3.3-V.
28	☐	☐	Connect V_{CCIO} pins to support each bank's I/O standards.
29	☐	☐	Connect V_{CCPD} pin to 3.3-V for a 3.3-V V_{CCIO}, 3.0-V for a 3.0-V V_{CCIO}, and 2.5-V for lower I/O voltages.
30	☐	☐	Connect VREF pins for voltage-referenced standards, and design to be noise-free.
31	☐	☐	Connect all PLL power pins even if the design does not use the PLL.
32	☐	☐	Use a linear regulator to power analog circuitry (V_{CCA_PLL} and V_{CCPT}).

33	☐	☐	Run a thick trace (at least 20 mils) from the power supply to each V_{CCA_PLL} pin.
34	☐	☐	Connect all V_{CCD_PLL} power pins to the quietest digital supply on the board.
35	☐	☐	Filter each V_{CCA_PLL} and V_{CCD_PLL} pin with a decoupling circuit.
36	☐	☐	Use ferrite bead and tantalum parallel capacitor for V_{CCA_PLL} and V_{CCD_PLL} where the power enters the board.
37	☐	☐	Design configuration DCLK and TCLK pins to be noise-free.
38	☐	☐	Connect JTAG pins to a stable voltage level if not in use.
39	☐	☐	Connect JTAG pins correctly to the download cable header. Ensure the pin order is not reversed.
40	☐	☐	Connect TRST to V_{CCPD} through a 1 kΩ resistor.
41	☐	☐	Pull-down TCK and pull-up TMS through resistors.
42	☐	☐	Ensure download cable and JTAG pin voltages are compatible.
43	☐	☐	Buffer JTAG signals per the recommendations, especially for connectors or if the cable drives more than three devices.
44	☐	☐	If your device is in a configuration chain, ensure all devices in the chain are connected properly.
45	☐	☐	Connect MSEL pins to select configuration scheme; do not leave floating. For flexibility, set up your board so you can connect each pin to either V_{CCPGM} or GND with a 0-Ω resistor.
46	☐	☐	Connect nIO_PULLUP correctly to set up internal pull-up resistors.
47	☐	☐	Hold the nCE chip enable low during configuration, initialization, and user mode.
48	☐	☐	Turn on the device wide output enable option, if required.
49	☐	☐	Specify the reserved state for unused I/O pins.
50	☐	☐	Carefully check the pin connections in the Quartus II software-generated PIN file. Do not connect RESERVED pins.
51	☐	☐	Consider SSN during board design: Break out large bus signals on board layers close to the device, route traces orthogonally if two signal layers are next to each, use a separation of 2× to 3× the trace width.
52	☐	☐	Check I/O termination and impedance matching for chosen I/O standards, especially for voltage-referenced standards.

53	☐	☐	Perform board-level simulation using IBIS models (when available).
54	☐	☐	Configure board trace models for Quartus II advanced I/O timing analysis.

"I/O and Clock Planning," on page 27

	Done	N/A	
55	☐	☐	Use the Quartus II Pin Planner to make pin assignments.
56	☐	☐	Use Quartus II Fitter reports after full compilation for sign-off of pin assignments.
57	☐	☐	Verify that Quartus II pin assignments match those in schematic and board layout tools.
58	☐	☐	Create Top-Level Design File command with I/O Assignment Analysis to check I/O assignments before design is complete.
59	☐	☐	Select suitable signaling type and I/O standard for each I/O pin.
60	☐	☐	Ensure that appropriate for I/O standard support is supported in targeted I/O bank.
61	☐	☐	Allow the software to assign locations for the negative pin in differential pin pairs.
62	☐	☐	Place I/O pins that share voltage levels in the same I/O bank.
63	☐	☐	Verify that all output signals in each I/O bank are intended to drive out at the bank's V_{CCIO} voltage level.
64	☐	☐	Verify that all voltage-referenced signals in each I/O bank are intended to use the bank's VREF voltage level.
65	☐	☐	Follow guidelines for placement of LVDS pins.
66	☐	☐	Use ALTMEMPHY megafunction (or IP core) for each memory interface, and follow connection guidelines in the appropriate documentation.
67	☐	☐	Use dedicated DQ pins and DQ groups for memory interfaces.
68	☐	☐	Use the Pin Planner Pad View to help meet pad placement guidelines.
69	☐	☐	Set appropriate I/O-related assignments if required to meet pad placement guidelines.
70	☐	☐	Make dual-purpose pin settings, and check for any restrictions when using these pins as regular I/O.
71	☐	☐	Check available device I/O features that can help interface with other devices: current strength, slew rate, I/O delays, open-drain, bus hold, programmable pull-up resistors, and PCI clamping diodes.

72	☐	☐	Use on-chip termination features to save board space.
73	☐	☐	Check that the required termination scheme is supported for all pin locations. (Column I/O banks and certain differential clock pairs do not support RD termination).
74	☐	☐	If you want to use DPA, be sure to enable the feature so that the design uses the correct PLLs on the right and left sides of the device.
75	☐	☐	Use correct dedicated clock pins and routing signals for clock and global control signals.
76	☐	☐	Use the device PLLs for clock management.
77	☐	☐	Analyze input and output routing connections for each PLL and clock pin. Ensure PLL inputs come from dedicated clock pins or from another PLL.
78	☐	☐	Enable PLL features and check settings in the MegaWizard Plug-In Manager.
79	☐	☐	Use the clock control block for clock selection and power-down.
80	☐	☐	Check that PLL offers the required number of clock outputs, and use dedicated clock output pins.
81	☐	☐	Ensure you select the correct PLL compensation mode.
82	☐	☐	Analyze design for possible simultaneous switching noise problems.
83	☐	☐	Reduce number of pins that switch voltage at exactly the same time whenever possible.
84	☐	☐	Use lower drive strength, differential I/O standards, and lower-voltage standards for high-switching I/Os.
85	☐	☐	Reduce number of simultaneously switching output pins within each bank.
86	☐	☐	Spread switching I/Os evenly throughout the bank.
87	☐	☐	Separate simultaneously-switching pins from input pins that are susceptible to SSN.
88	☐	☐	Place important clock and asynchronous control signals near ground signals and away from large switching buses.
89	☐	☐	Avoid using I/O pins one or two pins away from PLL power supply pins for high-switching or high drive strength pins.
90	☐	☐	Use staggered output delays to shift the output signals through time, or use adjustable slew rate settings

"Design and Compilation," on page 42

	Done	N/A	
91	☐	☐	Specify your third-party synthesis tool and use the correct supported version.
92	☐	☐	Use synchronous design practices. Pay attention to clock signals.
93	☐	☐	Use the Quartus II Design assistant to check design reliability.
94	☐	☐	Use megafunctions with the MegaWizard Plug-In Manager.
95	☐	☐	Follow recommended coding styles, especially for inferring device dedicated logic such as memory blocks.
96	☐	☐	Enable the chip-wide reset to clear all registers if required.
97	☐	☐	Consider resources available for register power-up and control signals. Do not apply both reset and preset signals to a register.
98	☐	☐	Follow recommendations to set up your source code and partition your design for incremental compilation; plan early in the design flow.
99	☐	☐	Perform timing budgeting and resource balancing between partitions to achieve best results, especially in team-based flows.
100	☐	☐	Create a design floorplan for each partition for best results.
101	☐	☐	Take advantage of SOPC Builder for system and processor designs.

"Timing Closure and Verification," on page 50

	Done	N/A	
102	☐	☐	Review resource utilization reports after compilation.
103	☐	☐	Review all Quartus II Messages, especially any warning messages.
104	☐	☐	Ensure timing constraints are complete and accurate, including all clock signals and I/O delays.
105	☐	☐	Turn on Optimize fast-corner timing.
106	☐	☐	Set **Optimize hold timing to All paths** (after checking for any design issues that may cause hold time problems).
107	☐	☐	Turn on **Enable multicorner timing analysis**.
108	☐	☐	Use `derive_pll_clocks` to create generated clocks for PLLs.

109	☐	☐	Use `derive_clock_uncertainty` to apply uncertainties.
110	☐	☐	Use `check_timing` to report on constraint problems.
111	☐	☐	Review TimeQuest Timing Analyzer reports after compilation to ensure there are no timing violations.
112	☐	☐	Ensure that the input I/O times are not violated when data is provided to the Stratix III device.
113	☐	☐	Perform Early Timing Estimation if you want timing estimates before running a full compilation.
114	☐	☐	Use Quartus II optimization features to achieve timing closure, or improve the resource utilization.
115	☐	☐	Use the Timing and Area Optimization Advisors to suggest optimization settings.
116	☐	☐	Use incremental compilation to preserve performance for unchanged blocks in your design, and to reduce compilation times.
117	☐	☐	Set up parallel compilation if you have multiple processors available for compilation.
118	☐	☐	Use the Compilation Time Advisor to suggest settings that reduce compilation time.
119	☐	☐	Specify your third-party simulation tool, and use the correct supported version and simulation models.
120	☐	☐	Specify your third-party formal verification tool and use the correct supported version.

“Power Analysis and Optimization,” on page 57

	Done	N/A	
121	☐	☐	Before the design is complete, estimate power consumption with the Early Power Estimator spreadsheet.
122	☐	☐	After compilation, analyze power consumption and heat dissipation in the PowerPlay Power Analyzer.
123	☐	☐	Provide accurate typical signal activities, preferably with a gate-level simulation VCD file, to get accurate power analysis results.
124	☐	☐	Specify correct operating conditions.
125	☐	☐	Use recommended design techniques and Quartus II options to optimize your design for power consumption, if required.
126	☐	☐	Use the Power Optimization Advisor to suggest optimization settings.
127	☐	☐	Set up the temperature sensing diode in your design to measure the device junction temperature for thermal management.