# AN 592: Cyclone IV Design Guidelines

© November 2009 AN-592-1.0

This application note provides an easy-to-use set of guidelines and a list of factors to consider in Cyclone® IV designs. Altera recommends following the guidelines listed in this application note throughout the design process. Altera® Cyclone IV devices offer a rich combination of logic, memory, and digital signal processing (DSP) with the lowest power. Cyclone IV devices are ideally suited for cost-sensitive, high-volume applications, including displays, wireless infrastructure equipment, industrial Ethernet, broadcast converters, and chip-to-chip bridging. Planning the FPGA and system early in the design process is crucial to your success.

☞ References to Cyclone IV devices in this application note refer only to Cyclone IV GX devices. Information about Cyclone IV E devices will be included in a future revision of this application note.

This application note describes the Cyclone IV device architecture, as well as aspects of the Quartus® II software and third-party tools that you can use in your design. It does not include all the details about the product. It also refers to other documentation where you can find detailed specifications, device feature descriptions, and additional guidelines.

The guidelines presented in this application note can improve productivity and avoid common design pitfalls. Table 1 describes the various stages of the design flow in the order that each stage is typically performed.

☞ To help verify that you have followed the guidelines described in this application note, refer to the "Design Checklist" on page 51.

**Table 1.** Summary of Design Flow Stages and Guideline Topics

| Stages of Design Flow | Guideline Topics |
|---|---|
| "System Specification" on page 2 | Planning design specifications and IP selection |
| "Device Selection" on page 3 | Device information, determining device density, package offerings, migration, and speed grade |
| "Early System and Board Planning" on page 6 | Early power estimation, planning configuration scheme, and planning for on-chip debugging |
| "Pin Connection Considerations for Board Design" on page 14 | Power-up, power pins, PLL connections, decoupling capacitors, configuration pins, signal integrity, and board-level verification |
| "I/O and Clock Planning" on page 22 | Pin assignments, early pin planning, I/O features and connections, memory interfaces, clock and PLL selection, and simultaneous switching noise (SSN) |
| "Design Entry" on page 34 | Coding styles and design recommendations, SOPC Builder, and planning for hierarchical or team-based design |
| "Design Implementation, Analysis, Optimization, and Verification" on page 41 | Synthesis tool, device utilization, messages, timing constraints and analysis, area and timing optimization, compilation time, verification, and power analysis and optimization |

For more information about the Cyclone IV device architecture, refer to the Literature: Cyclone IV Devices section of the Altera website. For the latest known issues related to Cyclone IV devices, refer to the Knowledge Database.

# System Specification

In systems that contain a Cyclone IV device, the FPGA plays a large role in the overall system and affects the rest of the system design. You must start the design process by creating detailed design specifications for the system and FPGA and determining the FPGA input and output interfaces to the rest of the system.

## Creating Design Specifications

Before you create your logic design or complete your system design, detailed design specifications should define the system, specify the I/O interfaces for the FPGA, identify the different clock domains, and include a block diagram of basic design functions. For suggestions about including intellectual property (IP) blocks, refer to "IP Selection". Taking the time to create these specifications helps to improve design efficiency.

1. ☐ Create detailed design specifications and a test plan, if appropriate.
2. ☐ Plan clock domains, clock resources, and I/O interfaces early with a block diagram.

Create a functional verification plan to ensure your team knows how to verify the system. Creating a test plan at this stage also helps you design for testability and manufacturability. You may need the ability to validate all the design interfaces. For example, if you want to perform built-in-self-test functions to drive the interfaces, you can use an UART interface with a Nios® II processor inside the FPGA device. For guidelines related to analyzing and debugging the device after it is in the system, refer to "Planning for On-Chip Debugging" on page 12.

If your design includes multiple designers, consider a common design directory structure. This eases the design integration stages. For more information about team-based designs, refer to "Planning for Hierarchical and Team-Based Design" on page 39.

## IP Selection

Altera and its third-party intellectual property 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 if 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 the hardware before you purchase the IP license. The evaluation supports an untethered mode, where 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.

3. ☐ Select the IP that affects your system design, especially the I/O interfaces.
4. ☐ If you plan to use OpenCore Plus tethered mode for your IP, ensure that your board design supports this mode of operation.

For more information about the available IP cores, refer to the Intellectual Property Solutions page on the Altera website.

# Device Selection

This section describes the first step in the Cyclone IV design process—choosing the number of transceivers, device density, features, package, and speed grade that best suit your design requirements. Altera recommends targeting FPGA migration devices, which is also described in this section.

5. ☐ Select a device based on transceivers, I/O pin count, LVDS channels, package offering, logic/memory/multiplier density, PLLs, clock routing, and speed grade.

For more information about the features available in each device density, including logic, memory blocks, multipliers, PLLs, package offerings, and I/O pin counts, refer to the *Cyclone IV Device Family Overview* chapter in volume 1 of the *Cyclone IV Device Handbook*.

## High-Speed Transceivers

Cyclone IV devices contain up to eight full duplex high-speed transceivers that can operate independently at data rates up to 3.125 Gbps with physical coding sublayer (PCS) and physical media attachment (PMA) support, and a PCI Express hard IP block. Choose a device density and package that supports enough transceivers for your application; larger densities and package pin counts offer more transceivers.

## Logic, Memory, and Multiplier Density

Cyclone IV devices offer a range of densities that provide different amounts of device logic resources, including LEs, memory, and multipliers. 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. Cyclone IV devices support vertical migration, which provides flexibility, as described in "Vertical Device Migration" on page 5.

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. Consider that coding style, device architecture, and the optimization options used in the Quartus II software can significantly affect a design's resource utilization and timing performance. For more information about determining resource utilization for a compiled design, refer to "Device Resource Utilization Reports" on page 41.

To obtain resource utilization estimates for certain configurations of Altera IP designs, refer to the IP and Megafunctions section of the Altera website.

6. ☐ Reserve device resources for future development and debugging.

Select a device that meets your design requirements with some safety margin, in case you want to add more logic later in the design cycle, upgrade, or expand your design. You may also want additional space in the device to make it easier when creating a design floorplan for incremental or team-based design, as described in "Planning for Hierarchical and Team-Based Design" on page 39. Also, consider reserving resources for debugging, as described in "Planning for On-Chip Debugging" on page 12.

## I/O Pin Count, LVDS Channels, and Package Offering

Cyclone IV devices are available in space-saving Quad Flat Pack No Lead (QFN) and FineLine BGA (FBGA) packages with various I/O pin counts and between 72 and 475 user I/O pins. Determine the required number of I/O pins for your application, considering the design's interface requirements with other system blocks.

Larger densities and package pin counts offer more LVDS channels for serialization and de-serialization; ensure that your device density-package combination includes enough LVDS channels.

Other factors also affect the number of I/O pins required for a design, including SSN concerns, pin placement guidelines, pins used as dedicated inputs, I/O standard availability for each I/O bank, differences between I/O standards and speed for row and column I/O banks, and package migration options. For more information about choosing pin locations, refer to "Pin Connection Considerations for Board Design" on page 14 and "I/O and Clock Planning" on page 22. Also, consider reserving pins for debugging, as described in "Planning for On-Chip Debugging" on page 12.

## PLLs and Clock Routing

Cyclone IV devices include two variations of PLLs—the general-purpose PLLs (GPLLs) and multi-purpose PLLs (MPLLs). Use the GPLLs for general-purpose applications in the FPGA fabric and periphery such as external memory interfaces. Use the MPLLs for clocking the transceiver blocks. If you do not use the MPLLs for transceiver clocking, you can use them for general-purpose clocking. Up to eight GPLLs and MPLLs provide robust clock management and synthesis for device clock management, external system clock management, and high-speed I/O interfaces in Cyclone IV devices.

Cyclone IV devices provide up to 12 dedicated clock pins (`CLK[15..4]`) that drive the global clocks (GCLKs). They also support four dedicated clock pins on each side of the device except the left side of the device. These clock pins can drive up to 30 GCLKs. Check that your chosen device density package combination includes enough PLLs and clock routing resources for your design. GCLK resources are shared between certain PLLs, which can affect which inputs are available for use. For more information and references about clock pins and global routing resources, refer to "I/O and Clock Planning" on page 22.

## Speed Grade

Device speed grade affects device timing performance and timing closure as well as power utilization. Cyclone IV devices are available in three speed grades—6, 7, and 8 (6 is the fastest). Generally, the faster devices cost more. One way to determine which speed grade your design requires is to consider the supported clock rates for the specific I/O interfaces.

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

You may want to consider 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.

## Vertical Device Migration

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

For more information about the list of migration devices, refer to the "Cyclone IV Device Package Offerings" table in the *Cyclone IV Device Family Overview* chapter in volume 1 of the *Cyclone IV Handbook*.

7. ☐ Consider vertical device migration availability and requirements.

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. Altera recommends specifying any potential migration options in the Quartus II software at the beginning of your design cycle. Selecting a migration device can impact the design's pin placement because the Fitter ensures your design is compatible with the selected device or devices.

You can add migration devices later in the design cycle, but this requires extra effort to check pin assignments and may 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.

The Quartus II Pin Planner highlights pins that change function in the migration device when compared with the currently selected device, as described in "Making FPGA Pin Assignments" on page 22.

# Early System and Board Planning

Plan system information related to the FPGA early in the design process, before you have completed your design in the Quartus II software. Early planning allows the FPGA team to provide early information to the PCB board and system designers. This section includes the following topics:

- "Early Power Estimation"
- "Planning for Device Configuration" on page 7
- "Planning for On-Chip Debugging" on page 12

## Early Power Estimation

FPGA power consumption is an important design consideration. You must accurately estimate power consumption 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 provide adequate current to support device operation.

8. ☐ Estimate power consumption with the Altera PowerPlay Early Power Estimator (EPE) spreadsheet to plan the cooling solution and power supplies before the logic design is complete.

Power consumption in FPGA devices depends on the design logic. This dependence can make power estimation challenging during the early board specification and layout stages. The EPE spreadsheet allows you to estimate power utilization before the design is complete by processing information about the device and device resources that you can use when considering design, operating frequency, toggle rates, and environmental conditions. Use the EPE 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 the information manually. The spreadsheet accuracy depends on your input 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 EPE spreadsheet includes the Import Data macro, which parses the information in the Quartus II-generated power estimation file, or alternatively from an older version of the Early Power Estimator, and transfers it into the spreadsheet. If you do not want to use the macro, you can transfer the data into the EPE spreadsheet manually. Altera recommends entering 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, Altera recommends performing a complete power analysis to more accurately check power consumption. The PowerPlay Power Analyzer tool in the Quartus II software provides an accurate estimation of power, ensuring that thermal and supply budgets are not violated. For the most accurate power estimation, use gate-level simulation results with a Verilog Value Change Dump File (**.vcd**) output file from the Quartus II Simulator or a third-party simulation tool. For more information, refer to "Power Analysis" on page 47.

For more information about PowerPlay EPE spreadsheets and user guides for each supported device family, refer to the PowerPlay Early Estimators (EPE) and Power Analyzer page on the Altera website.

For more information about using PowerPlay EPE spreadsheets, refer to the *PowerPlay Early Power Estimator User Guide*.

For more information about power estimation and analysis, refer to the *PowerPlay Power Analysis* chapter in volume 3 of the *Quartus II Handbook*.

## Planning for Device Configuration

Cyclone IV devices are based on SRAM memory that is volatile, so you must download configuration data to the Cyclone IV device each time the device powers up. Consider whether you require multiple configuration schemes, such as one for debug or testing and another for the production environment. Choosing the device configuration method early allows system and board designers to determine what companion devices, if any, are required for the 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 more information about board design guidelines related to configuration pins and connecting devices for configuration, refer to "Pin Connection Considerations for Board Design" on page 14.

In addition, Cyclone IV devices offer advanced configuration features, depending on your configuration scheme. Cyclone IV devices also include optional configuration pins and a reconfiguration option that you must choose early in the design process (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 *Configuration and Remote System Upgrades in Cyclone IV Devices* chapter in volume 1 of the *Cyclone IV 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 Cyclone IV devices with one of four configuration schemes:

- Fast passive parallel (FPP)—A controller supplies the configuration data in a parallel manner to the Cyclone IV device.
- Active serial (AS)—The Cyclone IV device controls the configuration process and receives the configuration data from a serial configuration (EPCS) device.
- Passive serial (PS)—A controller supplies the configuration data serially to the Cyclone IV device.
- JTAG—The Cyclone IV device is configured using the IEEE Standard 1149.1 interface with a download cable or a MAX® II device or microprocessor with flash memory.

You can enable any specific configuration scheme by driving the Cyclone IV device MSEL pins to specific values on the board.

9. ☐ Select a configuration scheme to plan companion devices and board connections.

FPP configuration is only supported in EP4CGX50, EP4CGX75, EP4CGX110, and EP4CGX150 devices.

For Cyclone IV devices to meet the PCIe 100 ms wake-up time requirement, you must use PS configuration mode for the EP4CGX15, EP4CGX22, and EP4CGX30 devices and FPP configuration mode for the EP4CGX50, EP4CGX75, EP4CGX110, and EP4CGX150 devices.

All configuration schemes use at least one of the following: a configuration device, a download cable, or an external controller (for example, a MAX II device or microprocessor).

### Serial Configuration Devices

Use the Altera serial configuration devices (EPCS) in the AS configuration scheme. Serial configuration devices offer a low-cost, low pin-count configuration solution.

For more 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*.

You can programme serial configuration devices with a USB-Blaster™, EthernetBlaster, or ByteBlaster™ II download cable using the Quartus II software. Alternatively, you can use the Altera programming unit (APU), supported third-party programmers such as BP Microsystems and System General, or a microprocessor with the SRunner software driver. SRunner is a software driver developed for embedded serial configuration device programming that you can customize to fit in different embedded systems.

For more information about SRunner, refer to *AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming* and the source code on the Literature: Application Notes section of the Altera website.

Serial configuration devices do not directly support the JTAG interface; however, you can program the device with JTAG download cables and 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.

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

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

### Download Cables

The Quartus II Programmer supports configuring Cyclone IV devices directly using PS or JTAG interfaces with Altera programming download cables. You can download design changes directly to the device with Altera download cables, making prototyping easy and allowing you to create 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 embedded logic analyzer. For more information and references about JTAG debugging tools, refer to "Planning for On-Chip Debugging" on page 12.

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

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

### MAX II Parallel Flash Loader

If your system contains common flash interface (CFI) flash memory, you can use it for Cyclone IV device configuration storage as well. The parallel flash loader (PFL) feature with MAX II devices allows 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 Cyclone IV device and supports compression to reduce the size of your configuration data. Both PS and FPP configuration modes are supported using the PFL feature.

10. ☐ If you want to use a flash device for the PFL, check the list of supported devices.

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

## Configuration Features

This section describes the Cyclone IV configuration features and how they affect your design process.

11. ☐ Ensure your configuration scheme and board supports any required features—data decompression, remote system upgrades, and singe event upset (SEU) mitigation.

For more information about data decompression and remote system upgrades, refer to the *Configuration and Remote System Upgrades in Cyclone IV Devices* chapter in volume 1 of the *Cyclone IV Device Handbook*.

For more information about SEU mitigation, refer to the *SEU Mitigation in Cyclone IV Devices* chapter in volume 1 of the *Cyclone IV Device Handbook*.

### Data Compression

When you enable data compression, the Quartus II software generates configuration files with compressed configuration data. These compressed files reduce the storage requirements in the configuration device or flash memory and decrease the time required to transmit the bitstream to the Cyclone IV device. The time required by a Cyclone IV device to decompress a configuration file is less than the time required to transmit the configuration data to the device.

Cyclone IV devices support decompression in the AS and PS configuration schemes. Altera recommends using the Cyclone IV decompression feature if you use AS or PS mode to reduce configuration time. The Cyclone IV decompression feature is not available in JTAG and FPP configuration schemes.

### Remote System Upgrades

Remote system upgrades help deliver feature enhancements and bug fixes without costly recalls, reduce time-to-market, extend product life, and help to avoid system downtime. Cyclone IV devices feature dedicated remote system upgrade circuitry. Soft logic (either the Nios II embedded processor or user logic) implemented in a Cyclone IV device can download a new configuration image from a remote location, store it in configuration memory, and direct the dedicated remote system upgrade circuitry to initiate a reconfiguration cycle.

Cyclone IV devices support remote update only in the single-device AS configuration scheme. You can implement remote update in conjunction with real-time decompression of configuration data.

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 the ALTREMOTE_UPDATE megafunction, refer to the *Remote Update Circuitry (ALTREMOTE_UPDATE) Megafunction User Guide*.

### SEU Mitigation and CRC Error Checks

Dedicated circuitry is built into Cyclone IV devices for the cyclical redundancy check (CRC) error detection feature that optionally checks for SEUs continuously and automatically. This allows you to confirm that the configuration data stored in a Cyclone IV device is correct and alerts the system to a configuration error. To take advantage of the SEU mitigation features, use the appropriate megafunction for CRC error detection. Use the CRC_ERROR pin to flag errors and design your system to take appropriate action. If not enabled for the CRC function, the CRC_ERROR pin is available as an user I/O pin.

## Quartus II Configuration Settings

This section describes several configuration options that you can set in the Quartus II software before compilation to generate configuration or programming files. Your board and system design are affected by these settings and pins, so consider them in the planning stages. Set the options in the **General** tab of the **Device and Pin Options** dialog box.

### Optional Configuration Pins

Table 2 lists the optional configuration pins you can enable in the Cyclone IV device.

**Table 2.** Optional Configuration Pins

| Configuration Pin | Description |
|---|---|
| CLKUSR | ■ The **Enable user-supplied start-up clock (CLKUSR)** option allows you to select which clock source you use for initialization—either the internal oscillator or an external clock provided on the CLKUSR pin.<br>■ Cyclone IV devices have an option to select CLKUSR (40 MHz maximum) as the external clock source for DCLK during AS configuration. You can change the clock source option in the Quartus II software from the **Configuration** tab of the **Device and Pin Options** dialog box. |
| INIT_DONE | To check if the device has completed initialization and is in user mode, 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 10-kΩ pull-up resistor to the $V_{CCIO}$ power supply of I/O Bank 3. |

12. ☐ Plan your board design to support the optional CLKUSR and INIT_DONE configuration pins, as required.

### Restart Configuration after Error

You can enable the **Auto-restart configuration after error** option so that 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 the $V_{CCIO}$ power supply of I/O Bank 3 unless it is connected to an external configuration device which provides an internal pull-up.

If you directly download design changes to the device with Altera download cables, the Quartus II software disables the **Auto-restart configuration after error** option.

13. ☐ Plan your board design to use the **Auto-restart configuration after error** option.

## Planning for On-Chip Debugging

On-chip debugging is an optional step in the design flow. Different debugging tools work better for different systems and different designers. Evaluate the 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. Early planning can reduce the time spent debugging and eliminates the need to make changes later to accommodate your preferred debugging methodologies. However, 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 or tools, as described in"On-Chip Debugging Tools" on page 12, then refer to "Planning Guidelines for Debugging Tools" on page 13.

### On-Chip Debugging Tools

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

- SignalProbe incremental routing—Quickly routes internal signals to the I/O pins without affecting the routing of the original design. Starting with a fully routed design, select and route the signals for debugging to either previously reserved or currently unused I/O pins.
- SignalTap II Embedded Logic Analyzer—Probes the state of internal and I/O signals without using 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. It 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. The SignalTap II Embedded Logic Analyzer works best for synchronous interfaces. For debugging asynchronous interfaces, consider using SignalProbe or an external logic analyzer to accurately view the signals.
- Logic Analyzer Interface—Allows 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. Use this feature to connect a large set of internal device signals to a small number of output pins for debugging purposes and to multiplex signals with design I/O pins if required.

- In-System Memory Content Editor—Provides read and write access to in-system FPGA memories and constants through the JTAG interface so you can test changes to the memory content and constant values in the FPGA while the device is functioning in the system.
- Virtual JTAG Megafunction—Allows you to build your own system-level debugging infrastructure, including processor-based debugging solutions and debugging tools in the software for system-level debugging. Instantiate the SLD_VIRTUAL_JTAG megafunction 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 *Virtual JTAG (SLD_VIRTUAL_JTAG) Megafunction User Guide* and *Section IV. In-System Design Debugging* in volume 3 of the *Quartus II Handbook*. The section overview provides more information about choosing a debugging solution.

14. ☐ Take advantage of the on-chip debugging features to analyze internal signals and perform advanced debugging techniques.

### Planning Guidelines for Debugging Tools

If you intend to use the on-chip debugging tools, plan for the tool or tools when developing the system board, Quartus II project, and design, as described in the following checklist:

15. ☐ Select the 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 the SignalTap II Embedded Logic Analyzer, Logic Analyzer Interface, In-System Memory Content Editor, or Virtual JTAG Megafunction, plan your system and board with JTAG connections that are available for debugging.
17. ☐ Plan for the small amount of additional logic resources used to implement the JTAG hub logic for the JTAG debugging features.
18. ☐ For debugging with the SignalTap II Embedded Logic Analyzer, reserve device memory resources to capture data during system operation.
19. ☐ Reserve I/O pins for debugging with SignalProbe or the Logic Analyzer Interface so you do not have to change the design or board to accommodate the debugging signals later.
20. ☐ Ensure your board supports debugging mode where the debugging signals do not affect system operation.
21. ☐ Incorporate a pin header or mictor connector as required for an external logic analyzer or mixed signal oscilloscope.
22. ☐ To use debug tools incrementally and reduce compilation time, ensure the incremental compilation feature is on so you do not have to recompile the design to modify the debug tool.
23. ☐ To use the Virtual JTAG megafunction for custom debugging applications, instantiate it in the HDL code as part of the design process.
24. ☐ 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 for the memory block in the MegaWizard Plug-In Manager.

# Pin Connection Considerations for Board Design

When designing the interfaces to the Cyclone IV device, various factors can affect the PCB design. This section contains important guidelines for the following topics:

- "Device Power-Up"
- "Power Pin Connections and Power Supplies" on page 15
- "Configuration Pin Connections" on page 17
- "Board-Related Quartus II Settings" on page 19
- "Signal Integrity Considerations" on page 20
- "Board-Level Simulation and Advanced I/O Timing Analysis" on page 21

The following section, "I/O and Clock Planning" on page 22, describes the I/O signal connections for the FPGA, which also affect the board design.

For more information about board design guidelines, refer to the Board Design Resource Center. This Resource 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

Cyclone IV device I/O pins are hot-socketing compliant without external components. You can insert or remove a Cyclone IV device from a powered-up system board without damaging or interfering with normal system and board operation.

25. ☐ Design your board for power-up—Cyclone IV output buffers are tri-stated until the device is configured and the configuration pins drive out.
26. ☐ Design the voltage supply power ramps to be monotonic.

You can drive signals into the I/O pins before or during power up or power down without damaging the device. Cyclone IV devices support power up or power down of the VCCINT, VCCA, and VCCIO pins in any sequence to simplify the system level design. The individual power supply ramp-up and ramp-down rates can range from 50 µs to 50 ms. The power ramp must be monotonic.

In a hot socketing situation, the Cyclone IV device's output buffers are turned off during system power-up or power-down. Also, the Cyclone IV device does not drive out until the device is configured and working within the 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.

The POR circuit keeps the entire system in reset until the power supply voltage levels have stabilized after power up. After power up, the device does not release nSTATUS until $V_{CCINT}$, $V_{CCA}$, and $V_{CCIO}$ for the I/O banks that contain configuration pins are above the POR trip point of the device. After power down, brown-out occurs if the $V_{CCINT}$ or $V_{CCA}$ voltage sags below the POR trip point.

In Cyclone IV devices, you can select a fast or standard POR time, depending on the MSEL pin settings. The fast POR time is 3 ms < $T_{POR}$ < 9 ms for a fast configuration time. The standard POR time is 50 ms < $T_{POR}$ < 200 ms, which has a lower power-ramp rate.

27. ☐ Set the POR time to ensure the power supplies are stable.

When power is applied to a Cyclone IV device, a POR 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 Cyclone IV devices is 50 ms for standard POR or 3 ms for fast POR, while the minimum power supply ramp time is 50 µs.

Although power sequencing is not a requirement for correct operation, Altera recommends considering 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.

☞ Altera strongly recommends that you connect the GNDs between boards before connecting power to ensure device reliability and compliance to the hot-socketing specification.

28. ☐ Design power sequencing, voltage regulators, and ground connections for best device reliability.

For more information, refer to the *Power Requirements for Cyclone IV Devices* chapter in volume 1 of the *Cyclone IV Device Handbook*.

## Power Pin Connections and Power Supplies

Review the power pin connection guidelines to determine the power supplies that are required in your system and which voltage inputs can share supplies. The Cyclone IV core voltage $V_{CCINT}$ is 1.2 V and $V_{CCA}$ requires 2.5V.

The I/O voltage $V_{CCIO}$ connections depend on the I/O standards of the design and support 1.2, 1.5, 1.8, 2.5, 3.0, and 3.3 V.

☞ 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.

For more information about the supply voltages required for the Cyclone IV device and their recommended operation conditions, refer to the *Power Requirements for Cyclone IV Devices* chapter volume 1 of the *Cyclone IV Device Handbook*.

Voltage reference (VREF) pins serve as voltage references for certain I/O standards. The VREF pin is used mainly for voltage bias and does not source or sink much current. You can create the voltage with a regulator or 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 25.

For more information about power supply types and power supply sharing or isolation, review the *Cyclone IV Device Family Pin Connection Guidelines*.

### Decoupling Capacitors

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

Cyclone IV devices include embedded on-package and on-die decoupling capacitors to provide high-frequency decoupling. 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 the 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 to support optimal transceiver signal integrity and FPGA performance.

For each power supply, you must choose a network of bulk and ceramic decoupling capacitors. While you can use the SPICE simulation 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 *Power Delivery Network (PDN) Tool User Guide*. You can also download the Power Delivery Network (PDN) Tool.

29. ☐ Use the PDN tool to plan your power distribution netlist and decoupling capacitors.

### PLL Board Design Guidelines

For more information about designing your clock and PLL scheme, refer to "Clock and PLL Selection" on page 31 and "PLL Feature Guidelines" on page 32. The following checklist provides several considerations to design a power system for using PLLs and minimizing jitter because PLLs contain analog components embedded in a digital device.

30. ☐ Connect all PLL power pins to reduce noise even if the design does not use all the PLLs—$V_{CCA}$ to 2.5 V and $V_{CCD\_PLL}$ to 1.2 V.
31. ☐ Run a thick trace (at least 20 mils) from the power supply to each PLL power pin.
32. ☐ Connect all PLL digital power pins to the quietest digital supply on the board.
33. ☐ Use ferrite beads to isolate the PLL power supply from the digital power supply.

For more information about the board design guidelines for the PLL power supplies, refer to the **General Board Design Considerations/Guidelines** section of the Board Design Resource Center.

## Configuration Pin Connections

Depending on your configuration scheme, different pull-up and pull-down resistor or signal integrity requirements may apply. Some configuration pins also have specific requirements if unused. You must correctly connect the configuration pins. This section describes guidelines to address common configuration pin connection issues.

34. ☐ Check that all configuration pin connections and pull-up and pull-down resistors are set correctly for your configuration scheme.

For a list of the dedicated and dual-purpose configuration pins, and a description of the function and connection guidelines, refer to the *Configuration and Remote System Upgrades in Cyclone IV Devices* chapter in volume 1 of the *Cyclone IV Device Handbook*.

### DCLK and TCK Signal Integrity

The TCK or DCLK traces, or both, must produce clean signals with no overshoot, undershoot, or ringing.

35. ☐ Design the DCLK and TCK configuration pins to be noise-free.

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 AS, PS, or FPP configuration and cause a CRC error. For a chain of devices, noise on any of the TCK or DCLK pins in the chain could cause the JTAG programming or configuration to fail for the entire chain.

For more information about connecting devices in a chain, refer to the *Configuration and Remote System Upgrades in Cyclone IV Devices* chapter in volume 1 of the *Cyclone IV Device Handbook*.

### JTAG Pins

Because the JTAG configuration takes precedence over all other configuration methods, the JTAG pins must not be left floating or toggling during configuration if you are not using the JTAG interface.

36. ☐ Connect the JTAG pins to a stable voltage level if not in use.

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

#### JTAG Pin Connections

A Cyclone IV device operating in JTAG mode uses four required pins (TDI, TDO, TMS, and TCK). The TCK pin has an internal weak pull-down resistor; and the TDI and TMS pins have internal weak pull-up resistors (typically 25 kΩ). All the JTAG pins are powered by the $V_{CCIO}$ power supply of I/O Bank 9.

37. ☐ Connect the JTAG pins correctly to the download cable header. 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.

38. ☐ To disable the JTAG state machine during power-up, pull the TCK pin low through a resistor to ensure that an unexpected rising edge does not occur on TCK.

39. ☐ Pull TMS high through a resistor.

### 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.

40. ☐ Because the download cable interfaces with the JTAG pins of your device, ensure the download cable and JTAG pin voltages are compatible.

In a JTAG chain containing devices with different voltages, devices with a higher voltage must drive devices with the same or lower voltage. With this device arrangement, a level shifter is required at the end of the chain. If this arrangement is not possible, you must add more level shifters to the chain.

For more information about connecting a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary Scan Testing for Cyclone IV Devices* chapter in volume 1 of the *Cyclone IV 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 the fastest switching JTAG signal. Because cables and board connectors tend to make bad transmission lines and introduce noise to the signals, Altera recommends buffering the signals at the connector. 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. However, prevention of signal deterioration 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 is added to the chain.

Each buffer must drive no more 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.

41. ☐ Buffer the JTAG signals per the recommendations, especially for connectors or if the cable drives more than three devices.

42. ☐ If your device is in a configuration chain, ensure all devices in the chain are connected properly.

### MSEL Configuration Mode Pins

Select the configuration scheme by driving the Cyclone IV device MSEL pins high or low. The JTAG configuration is always available, regardless of MSEL pin selection. MSEL pins are powered by the $V_{CCINT}$ power supply. MSEL pins have 9-kΩ internal pull-down resistors that are always active. To avoid problems detecting an incorrect configuration scheme, hardwire the MSEL pins to $V_{CCA}$ or GND without pull-up or pull-down resistors. You must not drive the MSEL pins with a microprocessor or another device.

43. ☐ Connect the MSEL pins to select the configuration scheme, do not leave them floating. For the flexibility to change between configuration modes during testing or debugging, set up your board to connect each pin to either $V_{CCA}$ or GND with a 0-Ω resistor.

### Other Configuration Pins

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

44. ☐ Hold the nCE chip enable low during configuration, initialization, and user mode.

In a single-device configuration or JTAG programming, tie nCE low. In a 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 must consider during board design. Ensure that these options are set correctly when you create the Quartus II project, and plan for their functionality during your board design.

### Device-Wide Output Enable Pin

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

45. ☐ Turn on the device-wide output enable option, if required.

### Unused Pins

To allow flexibility in your board design, specify the state of the 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
- As input tri-stated with weak pull-up

CAUTION

You must not connect the unused pins with outputs that drive ground to $V_{CC}$ or another signal source because this can create problems that can damage the device output driver.

46. ☐ Specify the reserved state for the unused I/O pins.

To improve signal integrity, set the 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 the clock pins to drive ground and set the other unused I/O pins as inputs that are tri-stated. If you set a reserved state for the unused pins, you must not connect those pins to other devices on the board. To make the appropriate settings, choose one of the five allowable states for the **Reserve all unused pins** option 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.

47. ☐ Carefully check the pin connections in the Quartus II software-generated Pin-Put File (**.pin**). Do not connect RESERVED pins.

When you compile your design, the Quartus II software generates the **.pin** to specify how you must 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 problems that can damage the device output driver. You can connect the `RESERVED_INPUT` I/O pins to a high or low signal on the board and you can leave the `RESERVED_INPUT_WITH_WEAK_PULLUP` and `RESERVED_INPUT_WITH_BUS_HOLD` pins unconnected.

## Signal Integrity Considerations

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

### Voltage Reference Pins

Voltage deviation on a `VREF` pin can affect the threshold sensitivity for inputs.

48. ☐ Design the `VREF` pins 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 24.

### Simultaneous Switching Noise (SSN)

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 the Board Design Resource Center for board layout recommendations that can help reduce some of the noise.

For example, consider the following items:

49. ☐ Break out large bus signals on the board layers close to the device to reduce crosstalk.

50. ☐ Route traces orthogonally if two signal layers are next to each other, whenever possible. Use a separation of two to three times the trace width.

### I/O Termination

Voltage-referenced I/O standards require both an input reference voltage, $V_{REF}$, and a termination voltage ($V_{TT}$). 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 double data rate (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.

Cyclone IV on-chip series termination ($R_S$ OCT) provides the convenience of no external components. Alternatively, you can use external pull-up resistors to terminate 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.

51. ☐ Check I/O termination and impedance matching for the chosen I/O standards, especially for voltage-referenced standards.

For more information about the on-chip termination (OCT) features and limitations, refer to "I/O Features and Pin Connections" on page 24.

## Board-Level Simulation and Advanced I/O Timing Analysis

To ensure that I/O signaling meets the 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.

52. ☐ Perform board-level simulation using IBIS models (when 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. Altera recommends analyzing board level timing as part of I/O and board planning, especially for high-speed designs.

53. ☐ Configure board trace models for Quartus II advanced I/O timing analysis.

You can configure board trace models of selected I/O standards and generate "board-aware" signal integrity reports with the Quartus II software. When you turn on the **Enable Advanced I/O Timing** option, the TimeQuest Timing Analyzer uses the 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.

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

# 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 and transceiver features in Cyclone IV devices. Various considerations are important to effectively plan the available I/O resources to maximize utilization and prevent issues related to signal integrity. Good clock management systems are also crucial to the performance of an FPGA design.

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 describes the following topics:



## Making FPGA Pin Assignments

With the Quartus II Pin Planner GUI, you can identify I/O banks, $V_{REF}$ groups, and differential pin pairings to help you through the I/O planning process. Right-click in the Pin Planner spreadsheet interface and click **Pin Finder** to search for specific pins. If you select migration devices, as described in "Vertical Device Migration" on page 5, the Pin Migration view highlights pins that change function in the migration device when compared with the currently selected device.

54. ☐ Use the Quartus II Pin Planner to create pin assignments.

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 File (**.csv**) containing your I/O assignments for spreadsheet use when all the pins are assigned.

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

55. ☐ Use the Quartus II Fitter messages and reports for sign-off of pin assignments.

Quartus II designers can then pass the pin location information to PCB designers. Pin assignments must 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 you must make changes to the pin-out. 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**. You can use this file to verify that each pin is correctly connected in board schematics.

56. ☐ Verify that the Quartus II pin assignments match those in the schematic and board layout tools.

For more information about using the Pin Planner to create 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, 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 placement and routing software as soon as possible to avoid 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 22. 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 create 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.

57. ☐ Use the **Create Top-Level Design File** command with I/O Assignment Analysis to check the I/O assignments before the design is complete.

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 the system requirements. The Pin Planner **Create/Import Megafunction** feature interfaces with the MegaWizard™ Plug-In Manager and allows you to create or import custom megafunctions and IP cores that use I/O interfaces. Enter PLL, LVDS, and transceiver blocks that can 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, you can pass the preliminary pin location information to the PCB designers as described in the previous section. When the design is complete, use the reports and messages generated by the Quartus II Fitter for the final sign-off of pin assignments.

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

## I/O Features and Pin Connections

Cyclone IV I/Os are designed for ease of use and rapid system integration, while simultaneously providing high bandwidth and support for common interfaces. 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 information about pin connections, refer to the *Cyclone IV Device Family Pin Connection Guidelines*.

### I/O Signaling Type

Cyclone IV 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 for 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 the effects of simultaneous switching outputs (SSO) from pins changing voltage levels at the same time (for example, external memory interface data and address buses). Voltage-referenced signaling also provides an improved logic transition rate with a reduced voltage swing and minimizes noise caused by reflection with a termination requirement. However, additional termination components are required for the reference voltage source, $V_{TT}$.

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. Differential signaling also avoids the requirement for a clean reference voltage. This is possible because of 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 an inverted and non-inverted pair.

Cyclone IV 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 create 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.

58. ☐ Plan the signaling type based on your system requirements.

69. ☐ Allow the software to assign locations for the negative pin in differential pin pairs.

### Selectable I/O Standards and Flexible I/O Banks

Cyclone IV I/O pins are grouped together into I/O banks and each bank has a separate power bus. Cyclone IV devices have up to 10 I/O banks and one configuration bank. Each device I/O pin is associated with one I/O bank. There are three user I/Os in the configuration bank with secondary programming functions that you can use as user I/Os if they are not used as programming I/Os.

60. ☐ Select the suitable signaling type and I/O standard for each I/O pin.

61. ☐ Ensure that the appropriate I/O standard is supported in the targeted I/O bank.

All single-ended I/O standards are supported except HSTL-12 Class II, which is only supported in column I/O banks. All differential I/O standards are supported in top, bottom, and right I/O banks. The only exception is HSTL-12 Class II, which is only supported in column I/O banks. The entire left side of the Cyclone IV devices contain dedicated high-speed transceiver banks for high-speed transceivers interface applications.

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 31.

62. ☐ Place the I/O pins that share voltage levels in the same I/O bank.

63. ☐ Verify that all the output signals in each I/O bank are intended to drive out at the bank's $V_{CCIO}$ voltage level.

64. ☐ Verify that all the voltage-referenced signals in each I/O bank are intended to use the bank's $V_{REF}$ voltage level.

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

Voltage-referenced standards are supported in an I/O bank using any number of single-ended or differential standards, as long as they use the same $V_{REF}$ and $V_{CCIO}$ values. For example, if you choose to implement both SSTL-2 and SSTL-18 in your Cyclone IV devices, the I/O pins using these standards (because they require different $V_{REF}$ values) must be in different banks from each other. However, the same I/O bank can support SSTL-2 and 2.5-V LVCMOS with the $V_{CCIO}$ set to 2.5 V and the $V_{REF}$ set to 1.25 V.

When you use the VREF pins as regular I/Os, they have higher pin capacitance than regular user I/O pins. This has an impact on the timing if the pins are used as inputs and outputs.

For more information about VREF pin capacitance, refer to the "Pin Capacitance" section in the *Cyclone IV Device Data Sheet* chapter in volume 3 of the *Cyclone IV Device Handbook*. For more information about how to identify VREF groups, refer to the Cyclone IV Device Family Pin-Out Files or the Quartus II Pin Planner tool.

65. ☐ Check the I/O bank support for the LVDS and transceiver features.

Different I/O banks include different support for LVDS signaling and the Cyclone IV transceiver banks include additional support.

For information about the number of channels available for the LVDS I/O standard, refer to the *I/O Feature in Cyclone IV Devices* chapter in volume 1 of the *Cyclone IV Device Handbook*. For more information about transceiver bank-related features, refer to the *Cyclone IV Transceivers Architecture* chapter in volume 2 of the *Cyclone IV Device Handbook*.

For more information about I/Os, refer to the *I/O Features in Cyclone IV Devices* chapter in volume 2 of the *Cyclone IV Device Handbook*. Refer to the Cyclone IV I/O banks figures that show the location of each I/O bank and what each bank supports. The figures describing the number of I/Os in each bank provides bank information specific to each device density. When designing LVTTL and LVCMOS inputs with Cyclone IV devices, refer to the section describing the I/O banks guidelines.

For more information about the electrical characteristics of each I/O standard, refer to the *Cyclone IV Device Data Sheet* chapter in volume 3 of the *Cyclone IV Device Handbook*.

### Placement Guidelines Related to Differential I/O Pins

The placement of single-ended I/O pins with respect to differential 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. After compilation, the Quartus II software generates a fitter report that summarizes the guidelines checked by the Quartus II software during compilation.

66. ☐ Use caution and follow the guidelines for pin placement located near the LVDS I/O.

The $V_{CCIO}$ power supply for a bank is susceptible to noise from switching outputs in the bank. To maintain an acceptable noise level on the $V_{CCIO}$ power 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 an acceptable noise level on the $V_{CCIO}$ power supply and to prevent output switching noise from shifting the $V_{REF}$ rail.

### Memory Interfaces

Cyclone IV devices provide an efficient architecture to quickly and easily fit wide external memory interfaces with their small modular I/O bank structure banks. Cyclone IV devices support existing and emerging external DDR memory standards, such as DDR2 SDRAM, DDR SDRAM, and QDRII SRAM. Cyclone IV devices support DDR external memory on the top, bottom, and right I/O banks.

67. ☐ Use the ALTMEMPHY megafunction (or IP core) for each memory interface and follow the connection guidelines and restrictions in the appropriate documentation.

A self-calibrating megafunction (ALTMEMPHY) is optimized to take advantage of the Cyclone IV I/O structure and the Quartus II TimeQuest Timing Analyzer. The megafunction allows you to set the external memory interface features and helps set up the physical interface (PHY) best suited for your system with the highest reliable frequency of operation. When using the Altera memory controller MegaCore functions, the ALTMEMPHY megafunction is instantiated for you.

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

68. ☐ Use dedicated DQ/DQS pins and DQ groups for memory interfaces.

The data strobe DQS and data DQ pin locations are fixed in the Cyclone IV device. Before you design your device pin-out, refer to the memory interface guidelines for details and important restrictions related to the connections for these and other memory-related signals.

For more information about connecting a Cyclone IV device with external memory devices, including the maximum supported clock rate for different memory standards and restrictions on pin placement, refer to the *External Memory Interfaces in Cyclone IV Devices* chapter in volume 1 of the *Cyclone IV Device Handbook*. For additional resources, refer to the External Memory Solutions Center.

For more information about the Cyclone IV PLL, refer to the *Clock Networks and PLLs in Cyclone IV Devices* chapter in the *Cyclone IV Device Handbook*. For more information about the ALTMEMPHY megafunction, refer to the *External DDR Memory PHY Interface (ALTMEMPHY) Megafunction User Guide*.

### Dual-Purpose and Special Pin Connections

Cyclone IV devices allow I/O flexibility with dual-purpose configuration pins. You can use dual-purpose configuration pins as general I/Os after device configuration is complete. Select the necessary 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 GCLK networks, as general-purpose input pins if not used as clock pins. When you use the clock inputs as general inputs, the I/O registers use LE-based registers because the clock input pins do not include dedicated I/O registers.

If not enabled, the device-wide reset and clear pins are available as design I/Os. For more information, refer to "Device-Wide Output Enable Pin" on page 19 and "Register Power-Up Levels and Control Signals" on page 36.

69. ☐ Make the dual-purpose pin settings and check for any restrictions when using these pins as regular I/Os.

### Cyclone IV I/O Features

The Cyclone IV device IOE offers a range of programmable features for an I/O pin. These features increase the flexibility of I/O utilization and provide an alternative to reduce the usage of external discrete components to on-chip, such as a pull-up resistor and diode. Table 3 lists Cyclone IV I/O features, provides usage information and design considerations, and provides references for more information about the features.

**Table 3.** Cyclone IV I/O Features (Part 1 of 2)

| Feature | Usage | Guidelines and More Information |
|---|---|---|
| MultiVolt I/O Interface | Allows all packages to interface with systems of different supply voltages. VCCIO pins can be connected to a 1.5-, 1.8-, 2.5-, 3.0, 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. | For a summary of MultiVolt I/O support, a list of the supported I/O standards and the typical values for input and output $V_{CCIO}$, $V_{REF}$, and board termination voltage ($V_{TT}$), refer to the *I/O Features in Cyclone IV Devices* chapter in volume 1 of the *Cyclone IV Device Handbook*. |
| 3.3-V I/O Interface | Cyclone IV I/O buffers support 3.3-V I/O standards as transmitters or receivers in your system. The output high voltage ($V_{OH}$), output low voltage ($V_{OL}$), input high voltage ($V_{IH}$), and input low voltage ($V_{IL}$) levels meet the 3.3-V I/O standards specifications when the Cyclone IV $V_{CCIO}$ voltage is powered by 3.3 V. | To ensure device reliability and proper operation when interfacing with a 3.3 V I/O system, you must ensure that the absolute maximum ratings of the Cyclone IV devices are not violated.<br>Altera recommends performing IBIS simulation to determine that the overshoot and undershoot voltages are within the guidelines. For more information, refer to the *I/O Features in Cyclone IV Devices* chapter in volume 1 of the *Cyclone IV Device Handbook*. |
| Programmable Output Current Strength | Programmable current-strength control available for certain I/O standards. This 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.<br>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 *I/O Features in Cyclone IV Devices* chapter in volume 1 of the *Cyclone IV 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 the 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.<br>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 ensure zero hold times, minimize setup times, or increase clock-to-output times. You can use delays as deskewing circuitry to ensure that all bits of a bus have the same delay going into or out of the device. | The Quartus II compiler can program these delays to automatically minimize setup time while providing a zero hold time. Programmable delays can increase the register-to-pin delays for output registers.<br>For more information about the delay specifications, refer to the *Cyclone IV Device Data Sheet* chapter in volume 3 of the *Cyclone IV Device Handbook*.<br>For more information about how to set the input and output pin delay, refer to the *Area and Timing Optimization* chapter in volume 2 of the *Quartus II Handbook*. |

**Table 3.** Cyclone IV I/O Features (Part 2 of 2)

| Feature | Usage | Guidelines and More Information |
|---|---|---|
| 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. | An external pull-up resistor is required to provide logic high. |
| Bus Hold | Weakly holds the signal on an I/O pin at its last driven state until the next input signal is present. With this feature, an external pull-up or pull-down resistor to hold a signal level when the bus is tri-stated is not required. The circuitry also pulls non-driven pins away from the input threshold voltage where noise can cause unintended high-frequency switching. | If you enable the bus-hold feature, you cannot use the programmable pull-up option. Disable the bus-hold feature if the I/O pin is configured for differential signals.<br>For more information about sustaining current driven through this resistor and the overdrive current used to identify the next driven input level for each $V_{CCIO}$ voltage level, refer to the *Cyclone IV Device Data Sheet* chapter in volume 3 of the *Cyclone IV Device Handbook*. |
| Programmable Pull-Up Resistor | A pull-up resistor 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 you enable the programmable pull-up option, you cannot use the bus-hold feature. |
| PCI Clamping Diode | Can be used to protect the pin from excessive overshoot voltage in 3.3-V LVTTL, 3.3-V LVCMOS, 3.0-V LVTTL, 3.0-V LVCMOS, 2.5-V LVTTL/LVCMOS, PCI and PCI-X I/O standard interfaces. | If the input I/O standard is 3.3-V LVTTL, 3.3-V LVCMOS, 3.0-V LVTTL, 3.0-V LVCMOS, 2.5-V LVTTL/LVCMOS, PCI, or PCI-X, the PCI clamp diode is enabled by default in the Quartus II software. |
| 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. OCT maintains signal quality, saves board space, and reduces external component costs. Cyclone IV devices support the $R_S$ OCT with or without calibration. Cyclone IV devices provide a series termination value of 25-Ω and 50-Ω. | $R_S$ OCT is supported on any I/O bank. $V_{CCIO}$ and $V_{REF}$ must be compatible for all I/O pins to enable $R_S$ OCT in a given I/O bank. I/O standards that support different $R_S$ values can be located in the same I/O bank as long as their $V_{CCIO}$ and $V_{REF}$ are not conflicting.<br>For more information, refer to the *I/O Features in Cyclone IV Devices* chapter in volume 1 of the *Cyclone IV Device Handbook*. |
| Programmable Pre-Emphasis and VOD | Increases the amplitude of the high frequency component of the output signal and helps to compensate for the frequency dependent attenuation along the transmission line. | For more information, refer to the *Cyclone IV Device I/O Features* chapter in volume 1 of the *Cyclone IV Device Handbook*. |

For more information, consider the following checklist items and refer to the appropriate documentation:

70. ☐ Check the available device I/O features that can help I/O interfaces—current strength, slew rate, I/O delays, open-drain, bus hold, programmable pull-up resistors, PCI clamping diodes, programmable pre-emphasis, and voltage output differential ($V_{OD}$).

71. ☐ Consider the OCT features to save board space.

72. ☐ Check that the required termination scheme is supported for all pin locations.

## 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 and how much logic is driven by a particular clock.

Cyclone IV devices provide up to 12 dedicated clock pins (`CLK[15..4]`) that can drive the GCLKs. Cyclone IV devices support four dedicated clock pins on each side of the device except the left side of the device. These clock pins can drive up to 30 GCLKs.

Cyclone IV devices offer two variations of PLLs—the GPLLs and the MPLLs. The GPLLs are used for general-purpose applications in the FPGA fabric and periphery such as external memory interfaces. The MPLLs are used for clocking the transceiver blocks. When the MPLLs are not used for transceiver clocking, they can be used for general-purpose clocking. There are up to eight GPLLs and MPLLs that provide robust clock management and synthesis for device clock management, external system clock management, and high-speed I/O interfaces in Cyclone IV devices.

For more information about the number of GCLK networks, GPLLs, and MPLLs in each device density, refer to the *Cyclone IV FPGA Device Family Overview* chapter in volume 1 of the *Cyclone IV Device Handbook*.

73. ☐ Use the correct dedicated clock pins and routing signals for the clock and global control signals.

The dedicated clock pins drive the clock network directly, ensuring lower skew than other I/O pins. Use the dedicated routing network to achieve predictable delay with less skew for the high fan-out signals. You can also use the clock pins and clock network to drive the control signals such as asynchronous reset.

74. ☐ Use the device PLLs for clock management.

Specific clock inputs connect to specific PLLs that 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 GCLK networks can drive throughout the entire device, serving as low-skew clock sources for device logic. This clock region has the maximum delay when compared with to other clock regions but allows the signal to reach everywhere within the device. This option is good for routing global reset and clear signals or routing clocks throughout the device.
- IOEs and internal logic can also drive GCLKs to create internally generated GCLKs and other high fan-out control signals; for example, synchronous or asynchronous clears and clock enables.
- PLLs cannot be driven by internally generated GCLKs. The input clock to the PLL must come from dedicated clock input pins, fed by a pin-driven dedicated GCLK, or through a clock control block if the clock control block is fed by an output from another PLL or a pin-driven dedicated GCLK.

75. ☐ Analyze the input and output routing connections for each PLL and clock pin. Ensure that the PLL inputs come from the dedicated clock pins or from another PLL.

For more information about these features and detailed clock connection information, refer to the *Clock Networks and PLLs in Cyclone IV Devices* chapter in volume 1 of the *Cyclone IV Device Handbook*.

If your system requires more clock or control signals than are available in the target device, consider cases when you can spare the dedicated clock resource, particularly low fan-out and low-frequency signals where clock delay and clock skew do not have a significant impact on 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 must not use global routing resources.

## PLL Feature 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 you can implement particular features and input and output frequencies in a particular PLL.

76. ☐ Enable the PLL features and check the settings in the MegaWizard Plug-In Manager.

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*.

Cyclone IV PLLs provide robust clock management and synthesis for device clock management, external system clock management, and high-speed I/O interfaces. All Cyclone IV PLLs support several features for general-purpose clock management. Use the following features when planning your PLL design.

For more information about the PLL features, refer to the *Clock Networks and PLLs in Cyclone IV Devices* chapter in the *Cyclone IV Device Handbook*.

For more 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 (ALTPLL) Megafunction User Guide*.

### Clock Feedback Mode

Cyclone IV PLLs support the following five clock feedback modes:

- Source synchronous mode
- No compensation mode
- Normal mode
- Zero delay buffer (ZDB) mode
- Deterministic latency compensation mode

Each mode compensates for different clock networks and delays so the clocks are aligned differently. Choose the correct feedback mode for your application.

77. ☐ Ensure you select the correct PLL feedback compensation mode.

### 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; for example, when a system 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`).

## Clock Control Block

In Cyclone IV devices, dedicated clock input pins, PLL counter outputs, dual-purpose clock I/O inputs, and internal logic can all feed the clock control block for each GCLK. The output from the clock control block in turn feeds the corresponding GCLK. The GCLK can drive the PLL input if the clock control block inputs are outputs of another PLL or dedicated clock input pins. For EP4CGX15, EP4CGX22, and EP4CGX30 devices, there are five clock control blocks on each side of the device periphery; providing up to 20 clock control blocks in each Cyclone IV device. For EP4CGX50, EP4CGX75, EP4CGX110, and EP4CGX150 devices, there are six clock control blocks on each top, bottom, and right side of the device periphery; 12 clock control blocks on the left side; providing up to 30 clock control blocks in each Cyclone IV device.

The control block has two functions:

- Dynamic GCLK clock source selection (not applicable for the dual-purpose clock [DPCLK] and internal logic input)
- GCLK network power down (dynamic enable and disable)

Use these features to select different clock input signals or power-down clock networks to reduce power consumption without using combinational logic in your design. In Cyclone IV 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. You can select a clock source statically with a setting in the Quartus II software or dynamically using internal logic to drive the multiplexer select inputs.

78. ☐ Use the clock control block for clock selection and power-down.

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

## Simultaneous Switching Noise

SSN becomes a concern when too many pins (in close proximity) change voltage levels at the same time. Consider the following checklist recommendations when planning I/O and clock connections:

79. ☐ Analyze your design for possible SSN problems.
80. ☐ Reduce the number of pins that switch voltage at exactly the same time whenever possible.
81. ☐ Use the differential I/O and lower-voltage standards for high switching I/Os.
82. ☐ Use lower drive strengths for high switching I/Os. The default drive strength setting may be higher than your design requires.
83. ☐ Reduce the number of simultaneously switching output pins within each bank. Spread the output pins across multiple banks if possible.
84. ☐ 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%).
85. ☐ Separate the simultaneously switching pins from the input pins that are susceptible to SSN.
86. ☐ Place the important clock and asynchronous control signals near the ground signals and away from the large switching buses.
87. ☐ Avoid using the I/O pins that are one or two pins away from the PLL power supply pins for high-switching or high drive-strength pins.
88. ☐ Use staggered output delays to shift the output signals through time or use the adjustable slew rate settings.

For more information about the features you can use, refer to "Cyclone IV I/O Features" on page 28.

# Design Entry

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. You can also use megafunctions and SOPC Builder to help design your FPGA system. In addition, when planning and creating the design, plan for a hierarchical or team-based design to improve design productivity.

## Design Recommendations

In a synchronous design, a clock signal triggers all events. When 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.

89. ☐ Use synchronous design practices. Pay attention to the clock signals.

Problems with asynchronous design techniques include reliance on propagation delays in a device, incomplete timing analysis, and possible glitches.

Pay particular attention to your clock signals because they have a strongly 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. For more information, refer to "PLL Board Design Guidelines" on page 16. 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.

90. ☐ Use the Quartus II Design Assistant to check design reliability.

Design Assistant in the Quartus II software is a design-rule checking tool that allows you to check for design issues early in the design flow. The Design Assistant checks your design for adherence to the Altera-recommended design guidelines or design rules. To run Design Assistant, on the Processing menu, point to **Start** and click **Start Design Assistant**. To set 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 feature, 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 of multi-asynchonous clock designs, refer to *Synthesis and Scripting Techniques for Designing Multi-Asynchronous Clock Designs*.

## Using Megafunctions

Altera provides parameterizable megafunctions that are optimized for Altera device architectures. Instead of coding your own logic, you can save design time using megafunctions. Additionally, Altera-provided megafunctions can offer more efficient logic synthesis and device implementation. You can scale the megafunction's size and set various options using 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 2.

The Quartus II MegaWizard Plug-In Manager provides a user interface to customize megafunctions. Altera recommends building or changing megafunction parameters using the MegaWizard Plug-In Manager to ensure you set all the ports and parameters correctly.

91. ☐ Use megafunctions with the MegaWizard Plug-In Manager.

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

## Recommended HDL Coding Styles

HDL coding styles can have a significant effect on the quality of results (QoR) for programmable logic designs. Use the Altera-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 you can take advantage of the dedicated logic block sizes and configurations.

92. ☐ Follow the recommended coding styles, especially for inferring device dedicated logic such as memory and DSP blocks.

For more information about specific HDL coding examples and recommendations, refer to the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*. You can 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

Cyclone IV devices support an optional chip-wide reset that allows you to override all clears on all the device registers, including the memory block registers (but not the memory contents itself). When the `DEV_CLRn` pin is driven low, all registers are cleared or reset to 0.

The following sections describe situations when synthesis performs an optimization called "NOT gate push back", where 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, before compiling your design, turn on **Enable device-wide reset (DEV_CLRn)** in the Quartus II software on the **General** tab of the **Device & Pin Options** dialog box.

93. ☐ Enable chip-wide reset to clear all the registers, if required.

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

For more information about LE and LAB architecture, refer to the *Logic Elements and Logic Array Blocks in Cyclone IV Devices* chapter in volume 1 of the *Cyclone IV Device Handbook*.

If the clock signal is not 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. 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 flipflop, in which case the output of the flipflop could go to a metastable unknown state.

For more information about reset design, refer to industry papers. For a good analysis of the reset architecture, refer to *Asynchronous & Synchronous Reset Design Techniques - Part Deux*.

By default, Quartus II integrated synthesis enables the **Power-Up Don't Care** logic option, 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 may 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 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 if you tap the register during on-chip verification, or view it during simulation, you must 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.

94. ☐ Consider the resources available for the register power-up and control signals. Do not apply both reset and preset signals to a register.

To implement a reset and preset signal on the same register, 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.

## 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 allows 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 the 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 and write protocol that enables flexible topologies for connecting master components to read and/or write 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.

For more information about the Avalon interface, refer to the *Avalon Interface Specifications* manual.

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.

95. ☐ Take advantage of SOPC Builder for system and processor designs.

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

## 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 Cyclone IV designs.

If you want to 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 the lower-level design blocks meet the 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, 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 to ensure that each partition is well placed relative to other partitions in the device.

Follow the Altera recommendations for creating design partitions to improve the overall quality of results. For example, registering partition I/O boundaries keeps critical timing paths inside one partition that can be optimized independently. When you specify design partitions, use the **Incremental Compilation Advisor** to ensure that partitions meet the Altera 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 Verilog Quartus Mapping file (**.vqm**) or EDIF (**.edf**) netlist files 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. For information about support for the Quartus II software incremental compilation, refer to your synthesis tool documentation. To provide more flexibility when partitioning, use hierarchy in your design . 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.

96. ☐ Follow the recommendations to set up your source code and partition your design for incremental compilation; plan early in the design flow.

97. ☐ Perform timing budgeting and resource balancing between the partitions to achieve the best results, especially in team-based flows.

For guidelines to help you create design partitions, refer to the *Best Practices for Incremental Compilation Partitions and Floorplan Assignments* chapter in volume 1 of the *Quartus II Handbook*.

### Planning in Bottom-Up and Team-Based Flows

In bottom-up design flows, the system architect must provide guidance to the designers of the 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, are communicated to the designers of the 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 the 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 well placed relative to the other partitions. When you create different location assignments for each partition, no location conflicts 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 are claimed. Floorplan assignments are recommended for timing-critical partitions in top-down flows.

98. ☐ Create a design floorplan for incremental compilation partitions, if required.

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 allows you to view the 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 to help you create a design floorplan, refer to the *Best Practices for Incremental Compilation Partitions and Floorplan Assignments* chapter in volume 1 of the *Quartus II Handbook*. For more information about creating placement assignments in the floorplan, refer to the *Analyzing and Optimizing the Design Floorplan* chapter in volume 2 of the *Quartus II Handbook*.

# Design Implementation, Analysis, Optimization, and Verification

After you create your design source code and apply the 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. If required, use the Quartus II software to optimize the design's resource utilization and achieve timing closure, preserve the performance of the 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.

## 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 HDL 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.

Altera recommends using 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.

99. ☐ Specify your third-party synthesis tool and use the correct supported version.

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

Your synthesis tool may 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. 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.

## Device Resource Utilization Reports

After compilation in the Quartus II software, 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.

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 usage of logic elements 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. Use this report to estimate device resource utilization for a partial design to ensure that registers were not removed due to missing connections with other parts of the design.

100. ☐ Review the resource utilization and optimization reports after compilation.

## 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, use the Message window tabs to look at only certain types of messages. You can suppress the messages if you have determined that they do not require action from you.

101. ☐ Review all Quartus II messages, especially any warning or error messages.

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 placing and routing 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 Quartus II software optimizes and analyzes your design using different timing models for each device speed grade, so you must perform timing analysis for the correct speed grade. The final programmed device may not operate as expected if the timing paths are not fully constrained, analyzed, and verified to meet the 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 Cyclone IV 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 TimeQuest 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.

Use the `report_datasheet` command to generate a datasheet report that summarizes the I/O timing characteristics of the entire design.

102. ☐ Ensure the timing constraints are complete and accurate, including all clock signals and I/O delays.

103. ☐ Review the TimeQuest Timing Analyzer reports after compilation to ensure there are no timing violations.

104. ☐ Ensure that the input I/O times are not violated when data is provided to the Cyclone IV device.

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 Cyclone IV devices.

105. ☐ Turn on the **Optimize multi-corner timing** option 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 corners. Therefore, turning on this option helps create a design implementation that is more robust across process, temperature, and voltage variations.

106. ☐ Turn on the **Enable multi-corner timing analysis** option during compilation under **TimeQuest Timing Analyzer** in the **Settings** dialog box, or use the `--multicorner` command line option.

This option directs the TimeQuest Timing Analyzer to analyze the design and generate slack reports for the slow and fast corners.

In your TimeQuest Timing Analyzer **.sdc** constraints file, use the following recommended constraints as applicable to your design:

107. ☐ Use `create_clock` and `create_generated_clock` to specify the frequencies and relationships for all the clocks in your design.

108. ☐ Use `set_input_delay` and `set_output_delay` to specify the external device or board timing parameters.

109. ☐ Use `derive_pll_clocks` to create the generated clocks for all the PLL outputs, according to the settings in the PLL megafunctions. Specify the multicycle relationships for the LVDS transmitters or receiver deserialization factors.

110. ☐ Use `derive_clock_uncertainty` to automatically apply inter-clock, intra-clock, and I/O interface uncertainties.

111. ☐ Use `check_timing` to generate a report for any problem with the design or applied constraints, including missing constraints.

## Area and Timing Optimization

This section describes 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 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*.

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 when compared with the final timing results.

112. ☐ Perform Early Timing Estimation if you want timing estimates before running a full compilation.

Physical synthesis optimizations make placement-specific changes to the netlist that improve results for a specific Altera device. You can specify the **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 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) utility 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. If you are interested in optimization for power usage, refer to "Power Optimization" on page 48.

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

The Optimization Advisors provide guidance when selecting 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 your requirements.

113. ☐ Use the Quartus II optimization features to achieve timing closure or improve resource utilization.

114. ☐ Use the Timing and Area Optimization Advisors to suggest optimization settings.

## Preserving Performance and Reducing Compilation Time

Use the incremental compilation feature to preserve logic in unchanged parts of your design, preserve timing performance, and reach timing closure more efficiently. You can speed up design iteration time by an average of 60% when making changes to the design with the incremental compilation feature.

115. ☐ Use incremental compilation to preserve performance for the unchanged blocks in your design and to reduce compilation times.

For guidelines and references, refer to "Planning for Hierarchical and Team-Based Design" on page 39.

116. ☐ Set up parallel compilation if you have multiple processors available for compilation.

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. By default, this option is set to **Use all available processors** so that parallel compilation is turned on by default.

117. ☐ Use the Compilation Time Advisor to suggest settings that reduce compilation time.

The Compilation Time Advisor provides guidance when selecting 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 information, 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 TimeQuest Timing Analyzer, including the delay of different device blocks and placement and routing information. 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.

Altera provides the ModelSim®-Altera simulator, which allows 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, Cadence NC-Sim, and Aldec Active-HDL. Specify your simulation tool in the **EDA Tools Settings** page of the **Settings** dialog box to generate the appropriate output simulation netlist.

118. ☐ Specify your third-party simulation tool and use the correct supported version and simulation models.

If you use a third-party simulation tool, use the software version that is supported with your Quartus II software version. The *Quartus II Software 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 can cause a mismatch with your simulation netlist. To create a testbench, on the Processing menu, point to **Start** and click **Start Testbench Template Writer**.

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 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. 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.

119. ☐ Specify your third-party formal verification tool and use the correct supported version.

120. ☐ If using formal verification, check for support and design limitations.

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

## Power Analysis

Before design completion, estimate power consumption using a spreadsheet as described in "Early Power Estimation" on page 6. 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.

121. ☐ After compilation, analyze power consumption and heat dissipation in the PowerPlay Power Analyzer.

122. ☐ Provide accurate typical signal activities, preferably with a gate-level simulation **.vcd** to generate accurate power analysis results.

You must compile your 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** 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.

123. ☐ Specify the correct operating conditions for power analysis.

You must also specify the operating conditions, including core voltage, device power characteristics, 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, on 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.

The report is a power estimate based on the data provided and is not a power specification. Always refer to the datasheet 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*.

## Power Optimization

Cyclone IV devices use patented architectural power reduction techniques to minimize power and deliver high performance.

To reduce dynamic power consumption in Cyclone IV devices, 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 vary from design to design.

124. ☐ Use the recommended design techniques and the Quartus II options to optimize your design for power consumption, if required.

125. ☐ Use the Power Optimization Advisor to suggest optimization settings.

For more information and additional design techniques to reduce power consumption, refer to the *Power Optimization* chapter in volume 2 of the *Quartus II Handbook*.

#### Clock Power Management

Clocks represent a significant portion of dynamic power consumption because of 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 is in an off state, thereby reducing the overall power consumption of the device.

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

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 by using the clock gating or clock-enable signals in the memory ports. Use the read-enable signal to ensure that read operations occur only when required. For example, if your design does not require read-during-write operations, you can reduce power consumption by de-asserting the read-enable signal during write operations. The Quartus II software automatically places any unused member blocks in low power mode to reduce static power.

### 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, use resistively terminated I/O standards such as SSTL for high-frequency applications. The output load voltage swings by an amount smaller than $V_{CCIO}$ around a 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.

The power used by external devices is not included in the PowerPlay EPE 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. The default setting is **Normal compilation**. You can choose **Extra effort** for additional power optimizations that may 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 source code for area saves power because fewer logic blocks are used; therefore, there is typically less switching activity. Use the DSE and Power Optimization Advisor to provide additional suggestions to reduce power.

For more 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 utility 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 **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. On 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.

## Document Revision History

Table 4 lists the revision history for this application note.

**Table 4.** Document Revision History

| Date and Version Number | Changes Made | Summary of Changes |
|---|---|---|
| November 2009, v1.0 | Initial release. | — |

# Design Checklist

The following checklist is a summary of the guidelines described in this application note. Use the checklist to help you follow these guidelines for each stage of your design.

| Done | N/A | Design Checklist (Part 1 of 6) |
|---|---|---|
| 1. ☐ | ☐ | Create detailed design specifications and a test plan, if appropriate. |
| 2. ☐ | ☐ | Plan clock domains, clock resources, and I/O interfaces early with a block diagram. |
| 3. ☐ | ☐ | Select the IP that affects your system design, especially the I/O interfaces. |
| 4. ☐ | ☐ | If you plan to use OpenCore Plus tethered mode for your IP, ensure that your board design supports this mode of operation. |
| 5. ☐ | ☐ | Select a device based on transceivers, I/O pin count, LVDS channels, package offering, logic/memory/multiplier density, PLLs, clock routing, and speed grade. |
| 6. ☐ | ☐ | Reserve device resources for future development and debugging. |
| 7. ☐ | ☐ | Consider vertical device migration availability and requirements. |
| 8. ☐ | ☐ | Estimate power consumption with the Altera PowerPlay Early Power Estimator (EPE) spreadsheet to plan the cooling solution and power supplies before the logic design is complete. |
| 9. ☐ | ☐ | Select a configuration scheme to plan companion devices and board connections. |
| 10. ☐ | ☐ | If you want to use a flash device for the PFL, check the list of supported devices. |
| 11. ☐ | ☐ | Ensure your configuration scheme and board supports any required features—data decompression, remote system upgrades, and singe event upset (SEU) mitigation. |
| 12. ☐ | ☐ | Plan your board design to support the optional `CLKUSR` and `INIT_DONE` configuration pins, as required. |
| 13. ☐ | ☐ | Plan your board design to use the Auto-restart configuration after error option. |
| 14. ☐ | ☐ | Take advantage of the on-chip debugging features to analyze internal signals and perform advanced debugging techniques. |
| 15. ☐ | ☐ | Select the 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 the SignalTap II Embedded Logic Analyzer, Logic Analyzer Interface, In-System Memory Content Editor, or Virtual JTAG Megafunction, plan your system and board with JTAG connections that are available for debugging. |
| 17. ☐ | ☐ | Plan for the small amount of additional logic resources used to implement the JTAG hub logic for the JTAG debugging features. |
| 18. ☐ | ☐ | For debugging with the SignalTap II Embedded Logic Analyzer, reserve device memory resources to capture data during system operation. |
| 19. ☐ | ☐ | Reserve I/O pins for debugging with SignalProbe or the Logic Analyzer Interface so you do not have to change the design or board to accommodate the debugging signals later. |

| Done | N/A | Design Checklist (Part 2 of 6) |
|---|---|---|
| 20. ☐ | ☐ | Ensure your board supports debugging mode where the debugging signals do not affect system operation. |
| 21. ☐ | ☐ | Incorporate a pin header or mictor connector as required for an external logic analyzer or mixed signal oscilloscope. |
| 22. ☐ | ☐ | To use debug tools incrementally and reduce compilation time, ensure the incremental compilation feature is on so you do not have to recompile the design to modify the debug tool. |
| 23. ☐ | ☐ | To use the Virtual JTAG megafunction for custom debugging applications, instantiate it in the HDL code as part of the design process. |
| 24. ☐ | ☐ | 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 for the memory block in the MegaWizard Plug-In Manager. |
| 25. ☐ | ☐ | Design your board for power-up—Cyclone IV output buffers are tri-stated until the device is configured and the configuration pins drive out. |
| 26. ☐ | ☐ | Design the voltage supply power ramps to be monotonic. |
| 27. ☐ | ☐ | Set the POR time to ensure the power supplies are stable. |
| 28 ☐ | ☐ | Design power sequencing, voltage regulators, and ground connections for best device reliability. |
| 29. ☐ | ☐ | Use the PDN tool to plan your power distribution netlist and decoupling capacitors. |
| 30. ☐ | ☐ | Connect all PLL power pins to reduce noise even if the design does not use all the PLLs—$V_{CCA}$ to 2.5 V and $V_{CCD\_PLL}$ to 1.2 V. |
| 31. ☐ | ☐ | Run a thick trace (at least 20 mils) from the power supply to each PLL power pin. |
| 32. ☐ | ☐ | Connect all PLL digital power pins to the quietest digital supply on the board. |
| 33. ☐ | ☐ | Use ferrite beads to isolate the PLL power supply from the digital power supply. |
| 34. ☐ | ☐ | Check that all configuration pin connections and pull-up and pull-down resistors are set correctly for your configuration scheme. |
| 35. ☐ | ☐ | Design the DCLK and TCK configuration pins to be noise-free. |
| 36. ☐ | ☐ | Connect the JTAG pins to a stable voltage level if not in use. |
| 37. ☐ | ☐ | Connect the JTAG pins correctly to the download cable header. Ensure the pin order is not reversed. |
| 38. ☐ | ☐ | To disable the JTAG state machine during power-up, pull the TCK pin low through a resistor to ensure that an unexpected rising edge does not occur on TCK. |
| 39. ☐ | ☐ | Pull TMS high through a resistor. |
| 40. ☐ | ☐ | Because the download cable interfaces with the JTAG pins of your device, ensure the download cable and JTAG pin voltages are compatible. |
| 41. ☐ | ☐ | Buffer the JTAG signals per the recommendations, especially for connectors or if the cable drives more than three devices. |

| Done | N/A | Design Checklist (Part 3 of 6) |
|---|---|---|
| 42. ☐ | ☐ | If your device is in a configuration chain, ensure all devices in the chain are connected properly. |
| 43. ☐ | ☐ | Connect the MSEL pins to select the configuration scheme, do not leave them floating. For the flexibility to change between configuration modes during testing or debugging, set up your board to connect each pin to either $V_{CCA}$ or GND with a 0-Ω resistor. |
| 44. ☐ | ☐ | Hold the nCE chip enable low during configuration, initialization, and user mode. |
| 45. ☐ | ☐ | Turn on the device-wide output enable option, if required. |
| 46. ☐ | ☐ | Specify the reserved state for the unused I/O pins. |
| 47 ☐ | ☐ | Carefully check the pin connections in the Quartus II software-generated Pin-Put File (.pin). Do not connect RESERVED pins. |
| 48. ☐ | ☐ | Design the VREF pins to be noise-free. |
| 49. ☐ | ☐ | Break out large bus signals on the board layers close to the device to reduce crosstalk. |
| 50. ☐ | ☐ | Route traces orthogonally if two signal layers are next to each other, whenever possible. Use a separation of two to three times the trace width. |
| 51. ☐ | ☐ | Check I/O termination and impedance matching for the chosen I/O standards, especially for voltage-referenced standards. |
| 52. ☐ | ☐ | Perform board-level simulation using IBIS models (when available). |
| 53. ☐ | ☐ | Configure board trace models for Quartus II advanced I/O timing analysis. |
| 54. ☐ | ☐ | Use the Quartus II Pin Planner to create pin assignments. |
| 55. ☐ | ☐ | Use the Quartus II Fitter messages and reports for sign-off of pin assignments. |
| 56. ☐ | ☐ | Verify that the Quartus II pin assignments match those in the schematic and board layout tools. |
| 57. ☐ | ☐ | Use the Create Top-Level Design File command with I/O Assignment Analysis to check the I/O assignments before the design is complete. |
| 58. ☐ | ☐ | Plan the signaling type based on your system requirements. |
| 69. ☐ | ☐ | Allow the software to assign locations for the negative pin in differential pin pairs. |
| 60. ☐ | ☐ | Select the suitable signaling type and I/O standard for each I/O pin. |
| 61. ☐ | ☐ | Ensure that the appropriate I/O standard is supported in the targeted I/O bank. |
| 62. ☐ | ☐ | Place the I/O pins that share voltage levels in the same I/O bank. |
| 63. ☐ | ☐ | Verify that all the output signals in each I/O bank are intended to drive out at the bank's $V_{CCIO}$ voltage level. |
| 64. ☐ | ☐ | Verify that all the voltage-referenced signals in each I/O bank are intended to use the bank's $V_{REF}$ voltage level. |
| 65. ☐ | ☐ | Check the I/O bank support for the LVDS and transceiver features. |

| Done | N/A | Design Checklist (Part 4 of 6) |
|---|---|---|
| 66. ☐ | ☐ | Use caution and follow the guidelines for pin placement located near the LVDS I/O. |
| 67. ☐ | ☐ | Use the ALTMEMPHY megafunction (or IP core) for each memory interface and follow the connection guidelines and restrictions in the appropriate documentation. |
| 68. ☐ | ☐ | Use dedicated DQ/DQS pins and DQ groups for memory interfaces. |
| 69. ☐ | ☐ | Make the dual-purpose pin settings and check for any restrictions when using these pins as regular I/Os. |
| 70. ☐ | ☐ | Check the available device I/O features that can help I/O interfaces—current strength, slew rate, I/O delays, open-drain, bus hold, programmable pull-up resistors, PCI clamping diodes, programmable pre-emphasis, and voltage output differential ($V_{OD}$). |
| 71. ☐ | ☐ | Consider the OCT features to save board space. |
| 72. ☐ | ☐ | Check that the required termination scheme is supported for all pin locations. |
| 73. ☐ | ☐ | Use the correct dedicated clock pins and routing signals for the clock and global control signals. |
| 74. ☐ | ☐ | Use the device PLLs for clock management. |
| 75. ☐ | ☐ | Analyze the input and output routing connections for each PLL and clock pin. Ensure that the PLL inputs come from the dedicated clock pins or from another PLL. |
| 76. ☐ | ☐ | Enable the PLL features and check the settings in the MegaWizard Plug-In Manager. |
| 77. ☐ | ☐ | Ensure you select the correct PLL feedback compensation mode. |
| 78. ☐ | ☐ | Use the clock control block for clock selection and power-down. |
| 79. ☐ | ☐ | Analyze your design for possible SSN problems. |
| 80. ☐ | ☐ | Reduce the number of pins that switch voltage at exactly the same time whenever possible. |
| 81. ☐ | ☐ | Use the differential I/O and lower-voltage standards for high switching I/Os. |
| 82. ☐ | ☐ | Use lower drive strengths for high switching I/Os. The default drive strength setting may be higher than your design requires. |
| 83. ☐ | ☐ | Reduce the number of simultaneously switching output pins within each bank. Spread the output pins across multiple banks if possible. |
| 84. ☐ | ☐ | 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%). |
| 85. ☐ | ☐ | Separate the simultaneously switching pins from the input pins that are susceptible to SSN. |
| 86. ☐ | ☐ | Place the important clock and asynchronous control signals near the ground signals and away from the large switching buses. |
| 87. ☐ | ☐ | Avoid using the I/O pins that are one or two pins away from the PLL power supply pins for high-switching or high drive-strength pins. |
| 88. ☐ | ☐ | Use staggered output delays to shift the output signals through time or use the adjustable slew rate settings. |

| | Done | N/A | Design Checklist (Part 5 of 6) |
|---|---|---|---|
| 89. | ☐ | ☐ | Use synchronous design practices. Pay attention to the clock signals. |
| 90. | ☐ | ☐ | Use the Quartus II Design Assistant to check design reliability. |
| 91. | ☐ | ☐ | Use megafunctions with the MegaWizard Plug-In Manager. |
| 92. | ☐ | ☐ | Follow the recommended coding styles, especially for inferring device dedicated logic such as memory and DSP blocks. |
| 93. | ☐ | ☐ | Enable chip-wide reset to clear all the registers, if required. |
| 94. | ☐ | ☐ | Consider the resources available for the register power-up and control signals. Do not apply both reset and preset signals to a register. |
| 95. | ☐ | ☐ | Take advantage of SOPC Builder for system and processor designs. |
| 96. | ☐ | ☐ | Follow the recommendations to set up your source code and partition your design for incremental compilation; plan early in the design flow. |
| 97. | ☐ | ☐ | Perform timing budgeting and resource balancing between the partitions to achieve the best results, especially in team-based flows. |
| 98. | ☐ | ☐ | Create a design floorplan for incremental compilation partitions, if required. |
| 99. | ☐ | ☐ | Specify your third-party synthesis tool and use the correct supported version. |
| 100. | ☐ | ☐ | Review the resource utilization and optimization reports after compilation. |
| 101. | ☐ | ☐ | Review all Quartus II messages, especially any warning or error messages. |
| 102. | ☐ | ☐ | Ensure the timing constraints are complete and accurate, including all clock signals and I/O delays. |
| 103. | ☐ | ☐ | Review the TimeQuest Timing Analyzer reports after compilation to ensure there are no timing violations. |
| 104. | ☐ | ☐ | Ensure that the input I/O times are not violated when data is provided to the Cyclone IV device. |
| 105. | ☐ | ☐ | Turn on the Optimize multi-corner timing option on the Fitter Settings page in the Settings dialog box. |
| 106. | ☐ | ☐ | Turn on the Enable multi-corner timing analysis option during compilation under TimeQuest Timing Analyzer in the Settings dialog box, or use the `--multicorner` command line option. |
| 107. | ☐ | ☐ | Use `create_clock` and `create_generated_clock` to specify the frequencies and relationships for all the clocks in your design. |
| 108. | ☐ | ☐ | Use `set_input_delay` and `set_output_delay` to specify the external device or board timing parameters. |
| 109. | ☐ | ☐ | Use `derive_pll_clocks` to create the generated clocks for all the PLL outputs, according to the settings in the PLL megafunctions. Specify the multicycle relationships for the LVDS transmitters or receiver deserialization factors. |
| 110. | ☐ | ☐ | Use `derive_clock_uncertainty` to automatically apply inter-clock, intra-clock, and I/O interface uncertainties. |

| Done | N/A | Design Checklist (Part 6 of 6) |
|---|---|---|
| 111. ☐ | ☐ | Use `check_timing` to generate a report for any problem with the design or applied constraints, including missing constraints. |
| 112. ☐ | ☐ | Perform Early Timing Estimation if you want timing estimates before running a full compilation. |
| 113. ☐ | ☐ | Use the Quartus II optimization features to achieve timing closure or improve resource utilization. |
| 114. ☐ | ☐ | Use the Timing and Area Optimization Advisors to suggest optimization settings. |
| 115. ☐ | ☐ | Use incremental compilation to preserve performance for the unchanged blocks in your design and to reduce compilation times. |
| 116. ☐ | ☐ | Set up parallel compilation if you have multiple processors available for compilation. |
| 117. ☐ | ☐ | Use the Compilation Time Advisor to suggest settings that reduce compilation time. |
| 118. ☐ | ☐ | Specify your third-party simulation tool and use the correct supported version and simulation models. |
| 119. ☐ | ☐ | Specify your third-party formal verification tool and use the correct supported version. |
| 120. ☐ | ☐ | If using formal verification, check for support and design limitations. |
| 121. ☐ | ☐ | After compilation, analyze power consumption and heat dissipation in the PowerPlay Power Analyzer. |
| 122. ☐ | ☐ | Provide accurate typical signal activities, preferably with a gate-level simulation .vcd to generate accurate power analysis results. |
| 123. ☐ | ☐ | Specify the correct operating conditions for power analysis. |
| 124. ☐ | ☐ | Use the recommended design techniques and the Quartus II options to optimize your design for power consumption, if required. |
| 125. ☐ | ☐ | Use the Power Optimization Advisor to suggest optimization settings. |

101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support
www.altera.com/support