AN 519: Stratix IV Design Guidelines

Introduction

Altera® Stratix® IV FPGAs deliver a breakthrough level of system bandwidth and power efficiency for high-end applications, allowing designers to innovate without compromise.

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

The guidelines presented in this document can improve productivity and avoid common design pitfalls. The document discusses various stages of the design flow in the order that each stage is typically performed, as shown in Table 1. You can use the "Design Checklist" on page 55 to help verify that you have followed each of the guidelines.

Table 1. Summary of Design Flow Stages and Guideline Topics

Stages of Design Flow	Guideline Topics
"System Specification" on page 2	Planning design specifications, IP selection
"Device Selection" on page 3	Device information, determining device variant and density, package offerings, migration, HardCopy ASICs, speed grade
"Early System and Board Planning" on page 7	Early power estimation, thermal management option, planning for configuration scheme, planning for on-chip debugging
"Pin Connection Considerations for Board Design" on page 15	Power-up, power pins, PLL connections, decoupling capacitors, configuration pins, signal integrity, board-level verification
"I/O and Clock Planning" on page 24	Pin assignments, early pin planning, I/O features and connections, memory interfaces, clock and PLL selection, SSN
"Design Entry" on page 36	Coding styles and design recommendations, SOPC Builder, planning for hierarchical or team-based design
"Design Implementation, Analysis, Optimization, and Verification" on page 42	Synthesis tool, device utilization, messages, timing constraints and analysis, area and timing optimization, compilation time, verification, power analysis and optimization

For complete details about the Stratix IV device architecture, refer to the Stratix IV Literature page. For device errata related to engineering sample (ES) devices, refer to the *Errata Sheet for Stratix IV GX Devices* and the *Errata Sheet for Stratix IV GT Devices*. For the latest known issues related to Stratix IV FPGAs, refer to the Knowledge Center.

System Specification

In systems that contain a Stratix IV device, the FPGA typically plays a large role in the overall system and affects the rest of the system design. It is important to start the design process by creating detailed design specifications for the system and the 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. Refer to "IP Selection" for suggestions about including intellectual property blocks. Taking the time to create these specifications improves design efficiency, but the stage is often skipped by FPGA designers.

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

If your design includes multiple designers, it is useful to consider a common design directory structure. This eases the design integration stages. "Planning for Hierarchical and Team-Based Design" on page 40 provides more suggestions for team-based designs.

IP Selection

Altera and its third-party intellectual property (IP) partners offer a large selection of off-the-shelf IP cores optimized for Altera devices. You can easily implement these parameterized blocks of IP in your design, reducing your system implementation and verification time, and allowing you to concentrate on adding proprietary value.

3. ☐ Select IP that affects system design, especially I/O interfaces.

4. ☐ If you plan to use the Open Core Plus tethered mode for IP, ensure that your board design supports this mode of operation.

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 hardware before you purchase the IP license. The evaluation supports an untethered mode, in which the design runs for a limited time, or a tethered mode. The tethered mode requires an Altera serial JTAG cable connected between the JTAG port on your board and a host computer running the Quartus II Programmer for the duration of the hardware evaluation period.

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

SOPC Builder

SOPC Builder is a system development tool for creating systems based on processors, peripherals, and memories. With SOPC Builder, you specify the system components in a GUI, and SOPC Builder generates the interconnect logic automatically. SOPC Builder outputs HDL files that define all components of the system, and a top-level HDL design file that connects all the components together.

SOPC Builder is commonly used as the tool for creating systems based on the Nios II processor. However, SOPC Builder is a general purpose tool for creating 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 allows a component to use an address mapped read/write protocol that enables flexible topologies for connecting master components to any slave components. The Avalon Streaming interface enables point-to-point connections between streaming components that send and receive data using a high-speed, unidirectional system interconnect between source and sink ports.

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

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

For information about using SOPC builder to improve your productivity, refer to the SOPC Builder Literature page on the Altera website.

Device Selection

This section describes the first step in the Stratix IV design process—choosing the device family variant, device density, features, package, and speed grade that best suit your design requirements. You should also consider whether you want to target FPGA or HardCopy ASIC migration devices, also described in this section.

6. ☐ 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, and phase-locked loops (PLLs), as well as the various package offerings and I/O pin counts, refer to the *Overview for the Stratix IV Device Family* chapter in volume 1 of the *Stratix IV Device Handbook*.

Device Family Variant and High-Speed Transceivers

The Stratix IV device family contains three variants optimized to meet different application requirements. Stratix IV GX devices are transceiver-based FPGAs with full-duplex clock data recovery (CDR)-based transceivers at up to 8.5 Gbps. Stratix IV GT devices offer transceivers at up to 11.3 Gbps, and Stratix IV E devices are enhanced for logic density. If your application requires high-speed transceivers between 8.5 and 11.3 Gbps, choose a GT device variant. For transceivers up to 8.5 Gbps, choose a GX device variant. Otherwise, choose an E variant for increased logic density.

Stratix IV GX FPGAs include between 8 and 32 full-duplex transceiver channels with PCS and PMA support at data rates between 600 Mbps and 8.5 Gbps, and additional full-duplex transceiver channels with PMA-only support at data rates between 600 Mbps and 3.2 Gbps. These devices also include from one to four PCI Express hard IP blocks. Stratix IV GT FPGAs include up to 24 transceivers at rates up to 11.3 Gbps, and higher device densities contain one 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 and PCI Express blocks.

Logic, Memory, and Multiplier Density

Stratix IV devices offer a range of densities that provide different amounts of device logic resources, including memory, multipliers, and adaptive logic module (ALM) logic cells. 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. Stratix IV devices support vertical migration, which provides flexibility, as described in "Vertical Device Migration" on page 6.

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 42.

To obtain resource utilization estimates for certain configurations of Altera's IP designs, refer to the user guides for Altera megafunctions and IP MegaCore functions on the IP and Megafunctions page in the Literature section at www.altera.com.

7. ☐ 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 might also want additional space in the device to ease creating a design floorplan for incremental or team-based design, as described in "Planning for Hierarchical and Team-Based Design" on page 40. Consider reserving resources for debugging, as described in "Planning for On-Chip Debugging" on page 13.

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

Stratix IV devices are available in space-saving FineLine BGA packages with various I/O pin counts between 368 and 1104 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 full-duplex LVDS channels for different signaling; ensure that your device density-package combination includes enough LVDS channels. Other factors can also affect the number of I/O pins required for a design, including simultaneous switching noise (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 details about choosing pin locations, refer to "Pin Connection Considerations for Board Design" on page 15, and "I/O and Clock Planning" on page 24.

You can compile any existing designs in the Quartus II software with the **Auto device selected by the Fitter** option in the **Settings** dialog box to determine how many input and output pins are used. Also consider reserving pins for debugging, as described in "Planning for On-Chip Debugging" on page 13.

PLLs and Clock Routing

Stratix IV devices include up to 12 PLLs (with additional PLLs available from unused transceivers on GX devices). There are up to 16 global GCLKs, 88 regional RCLKs, and 132 additional periphery PCLKs. Check that your chosen device density package combination includes enough PLLs and clock routing resources for your design. Global clock resources are shared between certain PLLs, which can affect which inputs are available for use. For more details and references regarding clock pins and global routing resources, refer to "I/O and Clock Planning" on page 24.

Speed Grade

The device speed grade affects the device timing performance and timing closure, as well as power utilization. Stratix IV devices are available in three speed grades: -2, -3, and -4 (-2 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 specific I/O interfaces.

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

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

Vertical Device Migration

Stratix IV devices support vertical migration within the same package, which enables 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 any changes to the board layout, because you can replace the FPGA on the board with a different density Stratix IV device.

In addition, a vertical migration path exists between Stratix III and Stratix IV E devices, so you can move a Stratix III design to a higher density Stratix IV E device using a common PCB with minimal impact on the power supply design.

For more information about migrating from a Stratix III to Stratix IV E device, refer to *AN 557: Stratix III-to-Stratix IV E Cross-Family Migration Guidelines*.

8. ☐ 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. You should specify 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(s). It is possible to add migration devices later in the design cycle, but it requires extra effort to check pin assignments, and can require design or board layout changes to fit into the new target device. It is easier to consider these issues early in the design cycle than at the end, when the design is near completion and ready for migration.

As described in "Making FPGA Pin Assignments" on page 25, the Quartus II Pin Planner highlights pins that change function in the migration device when compared to the currently selected device.

HardCopy IV ASIC Migration

The HardCopy methodology allows you to seamlessly prototype your system with Stratix IV FPGAs and completely prepare your system for production prior to ASIC design handoff. Altera's HardCopy Design Center uses a proven turnkey process to implement the low-cost, low power, functionally equivalent, pin-compatible HardCopy IV ASIC. Although the Stratix IV device family leads the FPGA industry in low power consumption, HardCopy ASICs offer a typical 50 percent power reduction from the Stratix IV prototype. The result is a drop-in ASIC replacement to the prototyping Stratix IV FPGA on your system board.

You can start your HardCopy IV ASIC design by targeting your design to an appropriate Stratix IV FPGA and choosing the appropriate HardCopy IV ASIC companion in the latest version of the Quartus II software. You can migrate between the FPGA and ASIC revisions of your project when the Quartus II software includes these devices.

9. ☐ If you want to migrate to a HardCopy IV ASIC, review the appropriate design considerations.

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

■ It is especially important to use complete timing constraints if you want to migrate to a HardCopy device because of the rigorous verification requirements for ASICs.

■ RAM cannot be initialized to a known value in the HardCopy ASIC as it can in an FPGA. Therefore, if you plan to migrate to a HardCopy device, your design must write RAM contents during device operation instead of relying on memory initialization values.

■ Altera recommends adding the PLL reconfiguration IP core to your design so that you can change the PLL settings in-system on the HardCopy IV device.

For more information, refer to the *Phase-Locked Loop Reconfiguration (ALTPLL_RECONFIG) IP Core*.

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

Early System and Board Planning

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

■ "Early Power Estimation"

■ "Temperature Sensing for Thermal Management" on page 9

■ "Planning for Device Configuration" on page 9

■ "Planning for On-Chip Debugging" on page 13

Early Power Estimation

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

■ **Thermal planning**—The cooling solution sufficiently dissipates the heat generated by the device. In particular, the computed junction temperature must fall within normal device specifications.

■ **Power supply planning**—Power supplies must provide adequate current to support device operation.

10. ☐ Estimate power consumption with the Early Power Estimator spreadsheet to plan the cooling solution and power supplies before the logic design is complete.

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

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

The PowerPlay Early Power Estimator spreadsheet includes the Import Data macro, which parses the information in the 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 Early Power Estimator spreadsheet manually. You should enter additional resources to be used in the final design manually if the existing Quartus II project represents only a portion of your full design. You can edit the spreadsheet and add additional device resources or adjust the parameters after importing the power estimation file information.

When the design is complete, the PowerPlay Power Analyzer tool in the Quartus II software provides an accurate estimation of power, ensuring that thermal and supply budgets are not violated. For the most accurate power estimation, use gate-level simulation results with a **.vcd** output file from a third-party simulation tool. Refer to "Power Analysis" on page 48.

The PowerPlay Early Power Estimator spreadsheets and user guides for each supported device family are available on the Altera website at:

www.altera.com/support/devices/estimator/pow-powerplay.html

For more information about using the Early Power Estimator spreadsheet, 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*.

Temperature Sensing for Thermal Management

Calculating or measuring the junction temperature is crucial for thermal management. Historically, junction temperature is calculated using ambient or case temperature, junction-to-ambient (θ_{JA}) or junction to case (θ_{JC}) thermal resistance, and the device power consumption. Stratix IV devices include a temperature sensing diode (TSD) with embedded analog-to-digital converter (ADC) circuitry, so you do not require an external temperature sensing chip on the board.

11. ☐ Set up the temperature sensing diode in your design to measure the device junction temperature for thermal management.

The Stratix IV TSD can self-monitor the device junction temperature and be used with external circuitry for activities such as controlling air flow to the FPGA. You can bypass the ADC if you want to use an external temperature sensor, similar to the solution used for a Stratix II device or other devices.

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

For more information about these features, refer to the *Power Management in Stratix IV Devices* chapter in the *Stratix IV Device Handbook*.

Planning for Device Configuration

Stratix IV devices are based on SRAM cells, so you must download configuration data to the Stratix IV device each time the device powers up, because SRAM memory is volatile. 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 board design guidelines related to configuration pins and connecting devices for configuration, refer to "Pin Connection Considerations for Board Design" on page 15.

In addition, Stratix IV devices offer advanced configuration features, depending on your configuration scheme. Stratix IV devices also include optional configuration pins and a reconfiguration option that you should choose early in the design process (and set up in the Quartus II software), so you have all the information required for your board and system design.

This section includes the following topics:

■ "Configuration Scheme Selection"

■ "Configuration Features" on page 11

■ "Quartus II Configuration Settings" on page 13

For more information about configuration, refer to the *Configuration, Design Security, and Remote System Upgrades in Stratix IV Devices* chapter in volume 1 of the *Stratix 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 Stratix IV devices with one of four configuration schemes:

■ **Fast passive parallel (FPP)**—A controller supplies the configuration data in a parallel manner to the Stratix IV FPGA.

■ **Fast active serial (AS)**—The Stratix IV FPGA controls the configuration process and gets the configuration data from a serial configuration (EPCS) device.

■ **Passive serial (PS)**—A controller supplies the configuration data serially to the Stratix IV FPGA.

■ **Joint Test Action Group (JTAG)**—Stratix IV FPGA is configured via the IEEE Standard 1149.1 interface with a download cable, or using a MAX II device or microprocessor with flash memory.

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

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

For more information about the supported configuration schemes, refer to the Configuration Comparison. For complete information about the Stratix IV supported configuration schemes, how to execute the required configuration schemes, and all of the necessary option pin settings, including the MSEL pin settings, refer to the *Configuration, Design Security, and Remote System Upgrades in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

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

Serial Configuration Devices

The Altera serial configuration devices (EPCS) are used in the Fast AS configuration scheme.

13. ☐ If you want to use AS configuration mode with large device densities, confirm there is a configuration device available that is large enough for your target FPGA density.

For information about serial configuration devices, refer to the *Serial Configuration (EPCS) Devices Datasheet* in volume 2 of the *Configuration Handbook*.

Serial configuration devices can be programmed using a USB-Blaster™, EthernetBlaster, or the ByteBlaster™ II download cable via 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 designers 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 Altera website (www.altera.com).

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

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

🐾 For more details about the SFL, refer to *AN 370: Using the Altera Serial Flash Loader Megafunction with the Quartus II Software*.

Download Cables

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

🐾 For more information about how to use Altera's 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 already contains common flash interface (CFI) flash memory, you can utilize it for Stratix 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 Stratix IV device and supports compression to reduce the size of your configuration data. Both PS and FPP configuration modes are supported using this PFL feature.

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

🐾 For more information about PFL, refer to *Parallel Flash Loader IP Core User Guide*.

Configuration Features

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

15.☐ Ensure your configuration scheme and board support any required features: data decompression, design security, remote upgrades, SEU mitigation.

For more information about these features, refer to the *Configuration, Design Security, and Remote System Upgrades in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

Data Compression

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

Stratix IV devices support decompression in the FPP, fast AS, and PS configuration schemes. You should use the Stratix IV decompression feature if you use PS mode to reduce configuration time. In FPP, the host controller must send a DCLK that is ×4 the data rate and the configuration data must be valid for four DCLK cycles. The Stratix IV decompression feature is not available in JTAG configuration schemes.

Design Security Using Configuration Bitstream Encryption

The design security feature ensures that Stratix IV designs are protected from copying, reverse engineering, and tampering. Stratix IV devices have the ability to decrypt configuration bitstreams using the AES algorithm, an industry standard encryption algorithm that is FIPS-197 certified. Stratix IV devices have a design security feature which utilizes a 256-bit security key.

The design security feature is available in the FPP, fast AS, or PS configuration schemes. In FPP, the host controller must send a DCLK that is ×4 the data rate and the configuration data must be valid for four DCLK cycles. The design security feature is not available in JTAG 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. Stratix IV devices feature dedicated remote system upgrade circuitry. Soft logic (either the Nios II embedded processor or user logic) implemented in a Stratix 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.

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

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

For more information about the ALTREMOTE_UPDATE IP core, refer to the *Altera Remote Update IP Core User Guide*.

SEU Mitigation and CRC Error Checks

Dedicated circuitry is built into Stratix IV devices for a cyclic redundancy check (CRC) error detection feature that optionally checks for single event updates (SEUs) continuously and automatically. This allows you to confirm that the configuration data stored in a Stratix 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` or `CRITICAL ERROR` pin to flag errors and design your system to take appropriate action. If not enabled for their CRC function, these pins are available as design I/O.

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

Quartus II Configuration Settings

This section covers several configuration options that designers will 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 on the **General** tab of the **Device and Pin Options** dialog box.

Optional Configuration Pins

You can enable the following optional configuration pins:

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

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

16. ☐ Plan board design to support optional configuration pins `CLKUSR` and `INIT_DONE`, as required.

Restart Configuration after Error

You can enable the **Auto-restart after configuration 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 V_{CCPGM}, unless it is connected to an external configuration device which provides an internal pull-up.

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

Planning for On-Chip Debugging

On-chip debugging is an optional step in the design flow, and different debugging tools work better for different systems and different designers. Evaluate on-chip debugging options early in your design process to ensure that your system board, Quartus II project, and design are able to support the appropriate options. Planning can reduce time spent debugging, and eliminates any need to make changes later to accommodate your preferred debugging methodologies. Adding debug pins might not be enough, because of internal signal accessibility and I/O pin accessibility on the

device. First, select your preferred debugging tool(s) as described in "On-Chip Debugging Tools" on page 14, and then refer to "Planning Guidelines for Debugging Tools" on page 15.

On-Chip Debugging Tools

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

■ **SignalProbe incremental routing**—Quickly routes internal signals to I/O pins without affecting the routing of the original design. Starting with a fully routed design, you can select and route signals for debugging to either previously reserved or currently unused I/O pins.

■ **SignalTap II Embedded Logic Analyzer**—Probes the state of internal and I/O signals without the use of external equipment or extra I/O pins, while the design is running at full speed in an FPGA device. Defining custom trigger-condition logic provides greater accuracy and improves the ability to isolate problems. 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 view the signals most accurately.

■ **Logic Analyzer Interface**—Enables you to connect and transmit internal FPGA signals to an external logic analyzer for analysis, allowing you to take advantage of advanced features in your external logic analyzer or mixed signal oscilloscope. You can use this feature to connect a large set of internal device signals to a small number of output pins for debugging purposes and it can 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 memory content and constant values in the FPGA while the device is functioning in the system.

■ **In-System Sources and Probes**—Sets up customized register chains to drive or sample the instrumented nodes in your logic design, providing an easy way to input simple virtual stimuli and capture the current value of instrumented nodes.

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

For more information about these debugging tools, refer to the *Altera Virtual JTAG (altera_virtual_jtag) IP Core User Guide* and the appropriate debugging chapters in volume 3 of the *Quartus II Handbook*. The section overview provides more information about choosing a debugging solution.

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

Planning Guidelines for Debugging Tools

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

19. ☐ Select on-chip debugging scheme(s) early to plan memory and logic requirements, I/O pin connections, and board connections.

20. ☐ If you want to use the SignalTap II Embedded Logic Analyzer, Logic Analyzer Interface, In-System Memory Content Editor, In-System Sources and Probes, or Virtual JTAG IP core, plan your system and board with JTAG connections that are available for debugging.

21. ☐ Plan for the small amount of additional logic resources used to implement the JTAG hub logic for JTAG debugging features.

22. ☐ For debugging with the SignalTap II Embedded Logic Analyzer, reserve device memory resources to capture data during system operation.

23. ☐ 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 debugging signals later.

24. ☐ Ensure the board supports a debugging mode where debugging signals do not affect system operation.

25. ☐ Incorporate pin header or mictor connector as required for an external logic analyzer or mixed signal oscilloscope.

26. ☐ To use debug tools incrementally and reduce compilation time, ensure incremental compilation is on so you do not have to recompile the design to modify the debug tool.

27. ☐ To use the Virtual JTAG IP core for custom debugging applications, instantiate it in the HDL code as part of the design process.

28. ☐ To use the In-System Sources and Probes feature, instantiate the IP core in the HDL code.

29. ☐ To use the In-System Memory Content Editor for RAM or ROM blocks or the LPM_CONSTANT IP core, 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 IP Catalog.

Pin Connection Considerations for Board Design

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

■ "Device Power-Up"

■ "Power Pin Connections and Power Supplies" on page 17

■ "Configuration Pin Connections" on page 19

■ "Board-Related Quartus II Settings" on page 21

■ "Signal Integrity Considerations" on page 22

■ "Board-Level Simulation and Advanced I/O Timing Analysis" on page 24

"I/O and Clock Planning" on page 24 discusses the I/O signal connections for the FPGA, which also affect the board design.

Device Power-Up

Stratix IV devices offer hot socketing, which is also known as hot plug-in or hot swap, and power sequencing support without the use of any external devices. You can insert or remove a Stratix IV device or a board in a system during system operation without causing undesirable effects to the running system bus or the board inserted into the system. The hot socketing feature helps you use Stratix IV devices on PCBs that contain a mixture of 3.0, 2.5, 1.8, 1.5, and 1.2 V devices.

30. ☐ Design board for power-up: Stratix IV output buffers are tri-stated until the device is configured, and configuration pins drive out.

31. ☐ Design voltage supply power ramps to be monotonic.

You can drive signals into I/O pins, dedicated input pins, and dedicated clock pins of Stratix IV devices before or during power up or power down without damaging the device. Stratix IV devices support power up or power down of the V_{CCIO}, V_{CC}, V_{CCPGM}, and V_{CCPD} power supplies in any sequence to simplify system level design. The individual power supply ramp-up and ramp-down rates can range from 50 μs to 100 ms. The power ramp must be monotonic.

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

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

The power-on-reset (POR) circuit keeps the entire system in reset until the power supply voltage levels V_{CCPT}, V_{CC}, V_{CCPD}, and V_{CCPGM} have stabilized on power-up. The POR block checks the power supply levels, and device configuration cannot start until the release of the POR signal.

In Stratix IV devices, a pin-selectable option (PORSEL) allows you to select between a typical POR time setting of 12 ms or 100 ms. In both cases, you can extend the POR time by using an external component to assert the nSTATUS pin low. Extend POR time if the board cannot meet the maximum power ramp time specifications to ensure the device configures properly and enters user mode.

32. ☐ Set POR time to ensure power supplies are stable.

When power is applied to a Stratix 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 Stratix IV devices is 100 ms, while the minimum power supply ramp time is 50 μs.

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

Altera uses GND as a reference for hot-socketing operations and I/O buffer designs. Connecting the GND between boards before connecting the power supplies prevents the GND on your board from being pulled up inadvertently by a path to power through other components on your board. A pulled up GND could otherwise cause an out-of-specification I/O voltage or current condition with the Altera device.

For more information, refer to the *Power Management in Stratix IV Devices* chapter in the *Stratix IV Device Handbook*.

Power Pin Connections and Power Supplies

The Stratix IV core voltage V_{CC} is 0.9 V and the PLL digital power supply V_{CCD_PLL} is also 0.9 V. The programmable power technology voltage V_{CCPT} requires 1.5 V and several power supplies require a 2.5-V supply. The V_{CCPGM} pin powers the configuration pins, and can be 1.8, 2.5, or 3.0 V. For Stratix IV GX devices, the transceiver voltages require additional power supplies at 0.9 V, 1.1 V, 3.0/2.5 V, and 1.4/1.5 V, depending on the data rate.

The I/O voltage V_{CCIO} connections depend on the design's I/O standards and support 1.2, 1.5, 1.8, 2.5, and 3.0 V. I/O standards at 3.3 V are supported with a 3.0-V V_{CCIO} supply.

The device output pins do not meet the I/O standard specifications if the V_{CCIO} level is out of the recommended operating range for the I/O standard. The VCCPD pin must be connected to 3.0-V for a 3.0-V V_{CCIO} and 2.5-V for 2.5 or lower I/O voltages.

For a list of the supply voltages required for the Stratix IV device and their recommended operation conditions, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter in volume 4 of the *Stratix IV Device Handbook*.

Voltage reference (V_{REF}) pins serve as voltage references for certain I/O standards. The V_{REF} pin is used mainly for a voltage bias and does not source or sink much current. The voltage can be created with a regulator or a resistor divider network. For more information about V_{CCIO} voltages and V_{REF} pins for different I/O banks, refer to "Selectable I/O Standards and Flexible I/O Banks" on page 28.

For details about I/O power pin connections, refer to the *Stratix IV GX Device Family Pin Connection Guidelines*.

33. ☐ Connect all power pins correctly: 0.9-V core, 1.5-V, 2.5 V, and others as required.

34. ☐ Connect V_{CCIO} pins and V_{REF} pins to support each bank's I/O standards.

35. ☐ Connect the VCCPD pin to 3.0-V for a 3.0-V V_{CCIO}, and 2.5-V for lower I/O voltages.

36. ☐ Explore any unique requirements for FPGA power pins or other power pins on your board, and determine which devices on your board can share a power rail.

37. ☐ Use a linear regulator to power analog circuitry (V_{CCA_PLL} and V_{CCPT}). You can power the digital voltage rails by linear or switching regulators depending on efficiency or cost considerations.

38. ☐ Follow the specific guidelines for each pin in the *Stratix IV GX Device Family Pin Connection Guidelines*.

Decoupling Capacitors

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

Stratix 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 PCB design.

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

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

For more information about the PDN design and optimization process, refer to the *Device-Specific Power Delivery Network (PDN) Tool User Guide*.

39. ☐ 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 33 and "PLL Feature Guidelines" on page 34. The following checklist items provide several considerations to design a power system for PLL usage and minimize jitter, because PLLs contain analog components embedded in a digital device.

40. ☐ Connect all PLL power pins to reduce noise even if the design does not use all the PLLs: V_{CCA_PLL} to 2.5-V and V_{CCD_PLL} to 0.9-V.

41. ☐ Run a thick trace (at least 20 mils) from the power supply to each V_{CCA_PLL} pin.

42. ☐ Connect all V_{CCD_PLL} power pins to the quietest digital supply on the board.

43. ☐ Use ferrite beads to isolate the PLL power supply from the digital power supply

For more board design guidelines related to PLL power supplies, refer to the "General Board Design Considerations/Guidelines" section of the Board Design Resource Center.

Transceiver Board Design Guidelines

For guidelines specific to transceiver design, refer to the *Transceiver Architecture* and *Transceiver Configuration Guide* sections in the *Stratix IV Handbook*.

For board design guidelines related to high-speed transceivers, refer to the "Gigahertz Channel Design Considerations" section of the Board Design Resource Center.

Configuration Pin Connections

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

44. ☐ Check that all configuration pin connections and pull-up/pull-down resistors are set correctly for your configuration scheme(s).

For specifics about each configuration pin, refer to the *Stratix IV GX Device Family Pin Connection Guidelines*.

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

DCLK and TCK Signal Integrity

The TCK and/or DCLK traces should produce clean signals with no overshoot, undershoot, or ringing.

45. ☐ Design configuration DCLK and TCLK 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 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 JTAG programming or configuration to fail for the entire chain.

For more information about connecting devices in a chain, refer to the *Configuration, Design Security, and Remote System Upgrades in Stratix IV Devices* chapter in the *Stratix IV Handbook*.

JTAG Pins

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

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

47. ☐ Connect 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.

48. ☐ To disable the JTAG state machine during power-up, pull the `TCK` pin low through a 1-kΩ resistor to ensure that an unexpected rising edge does not occur on `TCK`.

49. ☐ Pull `TMS` and `TDI` high through a 10-kΩ resistor.

50. ☐ Connect `TRST` directly to V_{CCPD} (Connecting the pin to ground disables the JTAG circuitry).

Download Cable Operating Voltage

The operating voltage supplied to the Altera download cable by the target board through the 10-pin header determines the operating voltage level of the download cable. The dedicated JTAG pins for all Stratix IV devices reside in bank 1A and are powered by V_{CCPD}.

51. ☐ 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 V_{CCIO} levels, the devices with a higher V_{CCIO} level should drive the devices with the same or lower V_{CCIO} level. A one-level shifter is required at the end of the chain with this device arrangement. If this arrangement is not possible, you must add more level shifters into the chain.

For recommendations about connecting a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary-Scan Testing in Stratix IV Devices* chapter in the *Stratix IV Device Handbook*.

JTAG Signal Buffering

You might 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. Altera recommends buffering the signals at the connector because cables and board connectors tend to make bad transmission lines and introduce noise to the signals. After this initial buffer at the connector, add buffers as the chain gets longer or whenever the signals cross a board connector.

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

Each buffer should drive no 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.

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

53. ☐ 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 Stratix IV device MSEL pins high or low. JTAG configuration is always available, regardless of MSEL pin selection. The MSEL pins are powered by the V_{CCPGM} power supply of the residing bank. The MSEL[2..0] pins have 5 kΩ internal pull-down resistors that are always active. During POR and reconfiguration, the MSEL pins have to be at LVTTL V_{IL} and V_{IH} levels to be considered a logic low and logic high. To avoid any problems with detecting an incorrect configuration scheme, do not drive the MSEL pins with a microprocessor or another device.

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

Other Configuration Pins

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

55. ☐ Connect nIO_PULLUP correctly to set up internal pull-up resistors.

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

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

In single device configuration or JTAG programming, tie nCE low. In multi-device configuration, tie nCE low on the first device and connect its nCEO pin to the nCE pin on the next device in the chain.

Board-Related Quartus II Settings

The Quartus II software provides options for the FPGA I/O pins that you should consider during board design. Ensure that these options are set correctly when the Quartus II project is created, and plan for the functionality during board design.

Device-Wide Output Enable Pin

Stratix IV devices support an optional chip-wide output enable that allows you to override all tri-states on 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 pins behave as programmed. To use this chip-wide output enable, turn on **Enable device-wide output enable (DEV_OE)** in the Quartus II software before compiling your design on the **General** tab in the **Device & Pin Options** dialog box.

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

Unused Pins

To allow flexibility in board design, you can specify the state of unused pins as one of the following five states in the Quartus II software: as inputs that are tri-stated, as outputs that drive ground, as outputs that drive an unspecified signal, as input tri-stated with bus-hold, or as input tri-stated with weak pull-up.

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

To improve signal integrity, set unused pins as outputs that drive ground and tie them directly to the ground plane on the board. Doing so reduces inductance by creating a shorter return path and reduces noise on the neighboring I/O. Set the unused I/O pins as input tri-stated with weak pull-up (device wide option) to prevent the I/O pins from floating. This option is useful to reduce the number of I/O escape vias that in turn benefits PCB routing.

To make the setting that is appropriate for your design, choose one of the five allowable states for **Reserve all unused pins** on the **Unused Pins** tab in the **Device & Pin Options** dialog box, or apply the **Reserve Pin** assignment to specific pins in the Pin Planner.

For connection requirements for unused clock pins, refer to the *Stratix IV GX Device Family Pin Connection Guidelines*.

59. ☐ Carefully check the pin connections in the Quartus II software-generated **.pin** file. Do not connect RESERVED pins.

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

Signal Integrity Considerations

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

High-Speed Board Design

If your design has high-speed signals, especially with Stratix IV GX or GT device high-speed transceivers, the board design has a major impact on the signal integrity in the system.

For detailed information about signal integrity and board design, refer to the Board Design Resource Center.

For example, Altera provides the following application notes that offer information about high-speed board stack-up and signal routing layers:

■ *AN 528: PCB Dielectric Material Selection and Fiber Weave Effect on High-Speed Channel Routing*

■ *AN 529: Via Optimization Techniques for High-Speed Channel Designs*

■ *AN 530: Optimizing Impedance Discontinuity Caused by Surface Mount Pads for High-Speed Channel Designs*

Voltage Reference Pins

Voltage deviation on a V_{REF} pin can affect the threshold sensitivity for inputs.

60.☐ Design V_{REF} 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 27.

Simultaneous Switching Noise

Simultaneous switching noise (SSN) becomes a concern when too many pins (in close proximity) change voltage levels at the same time. Noise generated by SSN can reduce noise margin and cause incorrect switching. Although SSN is dominant on the device package, refer to the PCB guidelines in Altera's Board Design Guidelines Solution Center for board layout recommendations that can help reduce some of the noise. For example, consider the following items:

61.☐ Break out large bus signals on board layers close to the device to reduce cross talk.

62.☐ Route traces orthogonally if two signal layers are next to each other, if possible. Use a separation of 2 to 3 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 DDR memory system with superior noise margin.

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

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

Differential I/O standards typically require a termination resistor between the two signals at the receiver. The termination resistor must match the differential load impedance of the signal line. Stratix IV devices provide an optional differential on-chip resistor when using LVDS.

☞ Certain dedicated clock input pairs do not support differential termination.

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

For more information about on-chip termination features and limitations, refer to "I/O Features and Pin Connections" on page 27.

Board-Level Simulation and Advanced I/O Timing Analysis

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

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

64. ☐ Perform board-level simulation using IBIS models (when available).

For more information about this simulation flow, refer to the *Signal Integrity Analysis with Third-Party Tools* chapter in volume 2 of the *Quartus II Handbook*.

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

65. ☐ 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 **Enable Advanced I/O Timing** is turned on, the TimeQuest Timing Analyzer uses simulation results for the I/O buffer, package, and board trace model to generate more accurate I/O delays and extra reports to give insight into signal behavior at the system level. You can use these advanced timing reports as a guide to make changes to the I/O assignments and board design to improve timing and signal integrity.

I/O and Clock Planning

Planning and allocating I/O and clock resources is an important task with the high pin counts and advanced clock management features in Stratix 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 details the following topics:

■ "Making FPGA Pin Assignments" on page 25

■ "Early Pin Planning and I/O Assignment Analysis" on page 26

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 the **Pin Finder** to search for specific pins. If migration devices are selected, as described in "Vertical Device Migration" on page 6, the Pin Migration view highlights pins that change function in the migration device when compared to the currently selected device.

66. ☐ Use the Quartus II Pin Planner to make 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 (**.csv**) file containing your I/O assignments for spreadsheet use when all pins are assigned.

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

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

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

For details about using the Pin Planner to make I/O assignments, refer to the *Managing Device I/O Pins* 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 place-and-route 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 25. The Quartus II **Start I/O Assignment Analysis** command checks that the pin locations and assignments are supported in the target FPGA architecture. Checks include reference voltage pin usage, pin location assignments, and mixing of I/O standards. You can use I/O Assignment Analysis to validate I/O-related assignments that you make or modify throughout the design process.

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

69. ☐ Use the Create Top-Level Design File command with I/O Assignment Analysis to check I/O assignments before 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 system requirements. The Pin Planner **Create/Import Megafunction** feature interfaces with the IP Catalog, and enables you to create or import custom megafunctions and IP cores that use I/O interfaces. Enter PLL and LVDS blocks, including options such as dynamic phase alignment (DPA), because options affect the pin placement rules. When you have entered as much I/O-related information as possible, generate a top-level design netlist file using the **Create Top-Level Design File** command. You can use the I/O analysis results to change pin assignments or IP parameters and repeat the checking process until the I/O interface meets your design requirements and passes the pin checks in the Quartus II software.

When planning is complete, the preliminary pin location information can be passed to 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 *Managing Device I/O Pins* chapter in volume 2 of the *Quartus II Handbook*.

I/O Features and Pin Connections

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

For a list of I/O pin locations and connection guidelines, refer to the *Stratix IV GX Device Family Pin Connection Guidelines*.

I/O Signaling Type

Stratix 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 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.

Stratix 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 make a pin assignment for this positive end of the pair. When you specify a differential I/O standard, the Quartus II software automatically places the corresponding negative pin.

70. ☐ Plan signaling type based on system requirements.

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

Selectable I/O Standards and Flexible I/O Banks

Stratix IV I/O pins are arranged in groups called modular I/O banks. Depending on the device density, the number of I/O banks ranges from 16 to 24 banks. In Stratix IV devices, the number of I/O banks per side is four or six, depending on the device density. When migrating between devices with a different number of I/O banks per side, it is the middle or "B" bank that is removed or inserted.

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

73. ☐   Ensure that appropriate I/O standard support is supported in targeted I/O bank.

Certain I/O banks on the top and bottom or left and right of the device support different I/O standards and voltage levels. You can assign I/O standards and make other I/O-related settings in the Pin Planner. Be sure to use the correct dedicated pin inputs for signals such as clocks and global control signals, as described in "Clock and PLL Selection" on page 33.

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

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

76. ☐   Verify that all 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 V_{CCIO} pin in the bank. Each I/O bank is powered by the V_{CCIO} pins of that particular bank, and is independent of the V_{CCIO} of other I/O banks. A single I/O bank supports output signals that are driving at the same voltage as the V_{CCIO}. An I/O bank can simultaneously support any number of input signals with different I/O standards.

To accommodate voltage-referenced I/O standards, each Stratix IV device I/O bank supports multiple V_{REF} pins feeding a common V_{REF} bus. Set the V_{REF} pins to the correct voltage for the I/O standards in the bank. Each I/O bank can only have a single V_{CCIO} voltage level and a single V_{REF} voltage level at a given time. If the V_{REF} pins are not used as voltage references, they cannot be used as generic I/O pins and should be tied to V_{CCIO} of that same bank or GND.

An I/O bank including single-ended or differential standards can support voltage-referenced standards as long as all voltage-referenced standards use the same V_{REF} setting. For performance reasons, voltage referenced input standards use their own V_{CCPD} level as the power source, so you can place voltage-referenced input signals in a bank with a V_{CCIO} of 2.5 V or below. Voltage-referenced bi-directional and output signals must drive out at the I/O bank's V_{CCIO} voltage level.

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

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

👣   For information about the number of channels available for the LVDS I/O standard, refer to the *High-Speed Differential I/O Interfaces and DPA in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*. For more information about transceiver-bank-related features, refer to the *Transceiver Architecture in Stratix IV Devices* chapter in volume 2 of the *Stratix IV Device Handbook*.

For more information about I/O, refer to the *I/O Features in Stratix IV Devices* chapter of the *Stratix IV Device Handbook*. Refer to the Stratix IV I/O banks figure that shows the location of each I/O bank and what each bank supports. The figures describing the number of I/Os in each bank provide bank information specific to each device density. Refer to the section describing I/O bank restrictions for information about which I/O standards can be combined in each bank, and the section describing I/O placement guidelines for details about LVDS restrictions.

For the electrical characteristics of each I/O standard, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter in volume 4 of the *Stratix IV Device Handbook*.

Memory Interfaces

Stratix IV devices provide an efficient architecture to quickly and easily fit wide external memory interfaces with their small modular I/O bank structure banks. Stratix IV devices support existing and emerging external double data rate (DDR) memory standards, such as DDR3, DDR2, DDR SDRAM, QDRII+, QDRII SRAM, and RLDRAM II at frequencies of up to 533 MHz. The Stratix IV FPGA can support DDR external memory on any I/O banks on all sides of the device that do not support transceivers.

78. ☐ Use the ALTMEMPHY IP core for each memory interface, and follow connection guidelines/restrictions in the appropriate documentation.

A self-calibrating megafunction (ALTMEMPHY) is optimized to take advantage of the Stratix IV I/O structure. The megafunction allows you to set external memory interface features and helps set up the physical interface (PHY) best suited for your system. 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.

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

The data strobe DQS and data DQ pin locations are fixed in the Stratix 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 Stratix 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 Stratix IV Devices* chapter in the *Stratix IV Device Handbook*. For additional resources, refer to the External Memory Interface Solutions Center. Refer to the appropriate application note for specific guidelines related to specific memory interfaces.

For more information about the Stratix IV PLL, refer to the *Clock Networks and PLLs in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*. For more information about the ALTMEMPHY megafunction, refer to the *External Memory PHY Interface ((ALTMEMPHY) (nonAFI) Megafunction User Guide*.

Dual-Purpose and Special Pin Connections

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

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

The device-wide reset and clear pins are available as design I/Os if not enabled. For more information, refer to "Device-Wide Output Enable Pin" on page 21 and "Register Power-Up Levels and Control Signals" on page 38.

80.☐   Make dual-purpose pin settings, and check for any restrictions when using these pins as regular I/O.

Stratix IV I/O Features

The Stratix IV bi-directional I/O element (IOE) features support rapid system integration while simultaneously providing the high bandwidth required to maximize internal logic capabilities and system-level performance. Advanced features for device interfaces assist in high-speed data transfer into and out of the device and reduce the complexity and cost of the PCB. Table 2 lists Stratix IV I/O features, provides usage information and design considerations, and provides references for more information about the features.

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

Feature	Usage	Guidelines and More Information
MultiVolt I/O Interface	Allows all packages to interface with systems of different supply voltages. V_{CCIO} pins can be connected to a 1.2-, 1.5-, 1.8-, 2.5-, or 3.0-V power supply, depending on the output requirements. The output levels are compatible with systems of the same voltage as the power supply. V_{CCPD} power pins must be connected to 3.0-V for 3.0-V V_{CCIO}, and 2.5-V for other I/O voltages.	Refer to the previous sections and the *I/O Features in Stratix IV Devices* chapter of the *Stratix IV Device Handbook* for a summary of MultiVolt I/O support and a list of the supported I/O standards and the typical values for input and output V_{CCIO}, V_{CCPD}, V_{REF}, and board termination voltage (V_{TT}). Altera recommends that you use an external clamp diode on the column I/O pins when the input signal is 3.0 V or 3.3 V.
3.3-V I/O Interface	Stratix 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 Stratix IV V_{CCIO} voltage is powered by 3.0 V.	To ensure device reliability and proper operation when interfacing with a 3.3 V I/O system, it is important to make sure that the absolute maximum ratings of the Stratix IV devices are not violated. Altera recommends performing IBIS simulation to determine that the overshoot and undershoot voltages are within the guidelines. For more guidelines, refer to the *I/O Features in Stratix IV Devices* chapter of the *Stratix IV Device Handbook*.

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

Feature	Usage	Guidelines and More Information
Programmable Output Current Strength	Programmable current-strength control available for certain I/O standards. Can mitigate the effects of high signal attenuation due to a long transmission line or a legacy backplane. A higher current strength increases I/O performance, but also increases noise on the interface, so you can use current strength control to manage noise.	Ensure that the output buffer current strength is sufficiently high, but does not cause excessive overshoot or undershoot that violates voltage threshold parameters for the I/O standard. Altera recommends performing IBIS or SPICE simulations to determine the right current strength setting for your specific application. For a list of standards and settings, refer to the *I/O Features in Stratix IV Devices* chapter of the *Stratix 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 rising and falling edges. You can use slew rate to reduce SSN.	Confirm that your interface meets its performance requirements if you use slower slew rates. Altera recommends performing IBIS or SPICE simulations to determine the right slew rate setting for your specific application.
Programmable IOE Delay	Programmable delay chains can 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.	This feature helps read and time margins because it minimizes the uncertainties between signals in the bus. Refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter in volume 4 of the *Stratix IV Device Handbook* for the delay specifications.
Programmable Output Buffer Delay	Delay chains in the single-ended output buffer can independently control the rising and falling edge delays of the output buffer.	You can use delays to adjust the output buffer duty cycle, compensate channel-to-channel skew, reduce SSO noise by deliberately introducing channel-to-channel skew, and improve high-speed memory-interface timing margins.
Open-Drain Output	When configured as open-drain, the logic value of the output is either high-Z or 0. Used in system-level control signals that can be asserted by multiple devices in the system.	Typically, an external pull-up resistor is required to provide logic high.
Bus Hold	Weakly holds the signal on an I/O pin at its last driven state until the next input signal is present, using a resistor with a nominal resistance (RBH) of approximately 7 kΩ. With this feature, 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 the bus-hold feature is enabled, you cannot use the programmable pull-up option. Disable the bus-hold feature if the I/O pin is configured for differential signals. Refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter in volume 4 of the *Stratix IV Device Handbook* for the specific sustaining current driven through this resistor and the overdrive current used to identify the next driven input level for each V_{CCIO} voltage level.
Programmable Pull-Up Resistor	Pull-up resistor (typically 25 kΩ) weakly holds the I/O to the V_{CCIO} level when in user mode. Can be used with open-drain output to eliminate the requirement for an external pull-up resistor.	If the programmable pull-up option is enabled, you cannot use the bus-hold feature.

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

Feature	Usage	Guidelines and More Information
PCI Clamping Diode	Can be used to protect the pin from excessive overshoot voltage in PCI/PCI-X I/O standard interfaces.	Available only in column I/O pins.
On-Chip Termination (OCT)	Driver-impedance matching provides the I/O driver with controlled output impedance that closely matches the impedance of the transmission line to significantly reduce reflections. OCT maintains signal quality, saves board space, and reduces external component costs. Support for on-chip series (RS) with or without calibration, parallel (RT) with calibration, and dynamic series and parallel termination for single-ended I/O standards and on-chip differential termination (RD) for differential LVDS I/O standards.	OCT RS and RT are supported in the same I/O bank for different I/O standards if they use the same V_{CCIO} supply voltage. Each I/O in an I/O bank can be independently configured to support OCT RS, programmable current strength, or OCT RT. You cannot configure both OCT RS and programmable current strength or slew rate control for the same I/O buffer. Differential on-chip termination RD is available in row I/O pins and SERDES block clock pins (CLK 0, 2, 9, 11); it is not supported on column I/O pins, high speed clock pins CLK 1,3,8,10, or the corner PLL clock inputs. Refer to the *I/O Features in Stratix IV Devices* chapter of the *Stratix IV Device Handbook* for details about the support and implementation of this feature.
Programmable Pre-Emphasis	Increases the amplitude of the high frequency component of the output signal, and thus helps to compensate for the frequency dependent attenuation along the transmission line.	Refer to the *High Speed Differential I/O Interfaces and DPA in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.
Programmable Differential Output Voltage	Allows you to adjust output eye height to optimize trace length and power consumption. A higher VOD swing improves voltage margins at the receiver end while a smaller VOD swing reduces power consumption.	Refer to the *High Speed Differential I/O Interfaces and DPA in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.
Dedicated Differential I/O SERDES Circuitry with DPA and Soft-CDR Support	Row I/Os (right and left side) in both Stratix IV GX devices and E devices have built-in SERDES circuitry that supports high-speed LVDS interfaces at data rates between 150 Mbps and 1.6 Gbps. Dynamic Phase Alignment (DPA) circuitry automatically chooses the best phase to compensate the skew between source synchronous clock and received serial data. Soft-CDR mode provides the opportunity for synchronous/asynchronous applications for chip-to-chip and short reach board-to-board applications for SGMII protocols.	If you want to use DPA, be sure to enable the feature in the Parameter Editor so that the design uses the correct PLLs on the right and left sides of the device. (Column input buffers are true LVDS buffers. Column output buffers are single-ended, and must have external termination schemes to support LVDS, mini-LVDS, and RSDS standards.) DPA usage adds some constraints on the placement of high-speed differential channels. Refer to the feature description and placement guidelines in the *High Speed Differential I/O Interfaces and DPA in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

Consider the following checklist items and refer to the appropriate documentation for detailed guidelines:

81. ☐ Check 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 VOD.

82. ☐ Consider on-chip termination features to save board space.

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

84. ☐   Choose the appropriate mode of DPA, non-DPA, or soft-CDR for high-speed LVDS interfaces.

85. ☐   If you want to use DPA, be sure to enable the feature so that the design uses the correct PLLs on the right and left sides of the device, and follow DPA placement guidelines.

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.

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

86. ☐   Use correct dedicated clock pins and routing signals for 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 have a predictable delay with less skew for high fan-out signals. You can also use the clock pins and clock network to drive control signals like asynchronous reset.

87. ☐   Use the device PLLs for clock management.

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

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

■   The global clock (GCLKs) networks can drive throughout the entire device, serving as low-skew clock sources for device logic. This clock region has the maximum delay compared to other clock regions but allows the signal to reach everywhere within the device. This option is good for routing global reset/clear signals or routing clocks throughout the device.

■   The regional clock (RCLK) networks only pertain to the quadrant they drive into. The RCLK networks provide the lowest clock delay and skew for logic contained within a single device quadrant.

■   I/O elements (IOEs) and internal logic can also drive GCLKs and RCLKs to create internally generated global or regional clocks and other high fan-out control signals; for example, synchronous or asynchronous clears and clock enables.

■   PLLs cannot be driven by internally-generated GCLKs or RCLKs. The input clock to the PLL must come from dedicated clock input pins or pin/PLL-fed GCLKs or RCLKs.

■ Periphery clock (PCLK) networks are a collection of individual clock networks driven from the periphery of the Stratix IV device. Clock outputs from the DPA block, PLD-transceiver interface, row I/O pins, and internal logic can drive the PCLK networks. These PCLKs have higher skew compared to GCLK and RCLK networks and can be used instead of general purpose routing to drive signals into and out of the Stratix IV device.

88.☐ Analyze input and output routing connections for each PLL and clock pin. Ensure PLL inputs come from 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 Stratix IV Devices* chapter in the *Stratix IV Device Handbook*.

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

PLL 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 IP Catalog to enter your settings for the ALTPLL IP core, and check the results to verify whether particular features and input/output frequencies can be implemented in a particular PLL.

89.☐ Enable PLL features and check settings in the IP Catalog.

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*.

Stratix IV PLLs provide robust clock management and synthesis for device clock management, external system clock management, and high speed I/O interfaces. Stratix IV device PLLs are feature rich, supporting advanced capabilities such as clock switchover, dynamic phase shifting, PLL reconfiguration, and reconfigurable bandwidth. Stratix IV PLLs also support external feedback mode, spread-spectrum tracking, and post scale counter cascading features. All Stratix IV PLLs have the same core analog structure with only minor differences in features that are supported. You can use some of the following additional features when planning your PLL design.

For more information about the features in top/bottom and side PLLs, refer to the *Clock Networks and PLLs in Stratix IV Devices* chapter in the *Stratix IV Device Handbook*.

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

Clock Feedback Mode

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

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

Clock Outputs

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

91.☐ Check that the PLL offers the required number of clock outputs and use dedicated clock output pins.

Clock Switchover

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

Dynamic Reconfiguration

You can reconfigure PLL settings to update the output-clock frequency and the PLL bandwidth and to phase-shift in real time, without reconfiguring the entire Stratix IV device. The ability to reconfigure the PLL in real time is useful in applications that operate at multiple frequencies. It is also useful in prototyping environments, allowing you to sweep PLL output frequencies and adjust the output-clock phase dynamically. You can also use this feature to adjust clock-to-out (t_{CO}) delays in real time by changing the PLL output clock phase shift. This approach eliminates the need to regenerate a configuration file with the new PLL settings.

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

For more information about the ALTPLL_RECONFIG IP core, refer to the *Phase-Locked Loop Reconfiguration (ALTPLL_RECONFIG) IP Core User Guide*.

Clock Control Block

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

■ Clock source selection (with dynamic selection for global clocks)

■ Global clock multiplexing

■ Clock power down (with static or dynamic clock enable or disable)

Use these features to select different clock input signals, or power-down clock networks to reduce power consumption, without using any combinational logic in your design. In Stratix 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.

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

For information about using the ALTCLKCTRL IP core to set up the clock control block, refer to the *Clock Control Block (ALTCLKCTRL) IP Core 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:

93. ☐ Analyze design for possible simultaneous switching noise problems.

94. ☐ Reduce number of pins that switch voltage at exactly the same time whenever possible.

95. ☐ Use differential I/O standards and lower-voltage standards for high switching I/Os.

96. ☐ Use lower drive strengths for high switching I/Os. The default drive strength setting might be higher than your design requires.

97. ☐ Reduce number of simultaneously switching output pins within each bank. Spread output pins across multiple banks if possible.

98. ☐ 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%).

99. ☐ Separate simultaneously switching pins from input pins that are susceptible to SSN.

100. ☐ Place important clock and asynchronous control signals near ground signals and away from large switching buses.

101. ☐ Avoid using I/O pins one or two pins away from PLL power supply pins for high-switching or high drive strength pins.

102. ☐ Use staggered output delays to shift the output signals through time, or use adjustable slew rate settings.

Refer to "Stratix IV I/O Features" on page 30 for details about the features you can use.

Design Entry

In the development of complex FPGA designs, design practices, coding styles, and megafunction use have an enormous impact on your device's timing performance, logic utilization, and system reliability. In addition, while 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.

103. ☐ Use synchronous design practices. Pay attention to 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 large effect on your design's timing accuracy, performance, and reliability. Problems with clock signals can cause functional and timing problems in your design. Use dedicated clock pins and clock routing for best results. For clock inversion, multiplication, and division, use the device PLLs. For clock multiplexing and gating, use the dedicated clock control block or PLL clock switchover feature instead of combinational logic. Refer to "PLL Board Design Guidelines" on page 18. 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.

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

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

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

Using Megafunctions

Altera provides parameterizable megafunctions that are optimized for Altera device architectures. You can save design time by using megafunctions instead of coding your own logic. Additionally, the Altera-provided megafunctions can offer more efficient logic synthesis and device implementation. You can scale the megafunction's size and set various options with parameters. Megafunctions include the library of parameterized modules (LPM) and Altera device-specific megafunctions. You can also take advantage of Altera and third-party IP and reference designs to save design time, as described in "IP Selection" on page 2.

The Quartus II IP Catalog provides a user interface to customize megafunctions. You should build or change megafunction parameters using the individual megafunction parameter editor to ensure you set all ports and parameters correctly.

105.☐   Use megafunctions with the IP Catalog.

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

Recommended HDL Coding Styles

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

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

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

Register Power-Up Levels and Control Signals

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

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

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

For more information about LAB and ALM architecture, refer to the *Logic Array Blocks and Adaptive Logic Modules in Stratix IV Devices* chapter in the *Stratix 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 deasserted synchronously. The source of the reset signal is then connected to the asynchronous port of the registers, which can be directly connected to global routing resources. The synchronous deassertion 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.

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

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

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

If you force a particular power-up condition for your design, use the synthesis options available in your synthesis tool.

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 to set power-up conditions 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.

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 might ignore the power-up conditions.

108.☐ Consider resources available for 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.

For more information about reset logic and power up conditions, refer to the *Recommended HDL Coding Styles* chapter in volume 1 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 Stratix IV designs.

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

If you 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 lower-level design blocks meet top-level design requirements, reducing the time spent integrating and verifying the timing of the top-level design.

For more information about using the incremental compilation flows in the Quartus II software, refer to the *Quartus II Incremental Compilation for Hierarchical and Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*.

Planning Design Partitions

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

Follow Altera's 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.

Plan your source code so that each design block is defined in a separate file, so that the software can automatically detect changes to each block separately. Use hierarchy in your design to provide more flexibility when partitioning. Keep your design logic in the leaves of the hierarchy trees; that is, the top level of the hierarchy should have very little logic, and the lower-level design blocks contain the logic.

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

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

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

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

The system architect can plan design partitions at the top level and use the Quartus II software **Generate bottom-up design partition scripts** option to automate the process of transferring top-level project information to lower-level modules.

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 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 have already been claimed. Floorplan assignments are recommended for timing-critical partitions in top-down flows.

111. ☐ Create a design floorplan for incremental compilation partitions, if required for your design flow.

You can use the Quartus II Chip Planner to create a design floorplan using LogicLock region assignments for each design partition. With a basic design framework for the top-level design, the floorplan editor enables you to view connections between regions, estimate physical timing delays on the chip, and move regions around the device floorplan. When you have compiled the full design, you can also view logic placement and locate areas of routing congestion to improve the floorplan assignments.

For guidelines 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 with the Chip Planner* chapter in volume 2 of the *Quartus II Handbook*.

Design Implementation, Analysis, Optimization, and Verification

After you create your design source code and apply constraints including the device selection and timing requirements, your synthesis tool processes the code and maps it to elements of the device architecture. The Quartus II Fitter then performs placement and routing to implement the design elements in specific device resources. If required, you can use the Quartus II software to optimize the design's resource utilization and achieve timing closure, preserve the performance of unchanged design blocks, and reduce compilation time for future iterations. You can also verify the design functionality with simulation or formal verification. This section provides guidelines for these stages of the compilation flow.

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

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

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

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

For more information about supported synthesis tools, refer to the appropriate chapters 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.

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

More detailed resource information is available by viewing the reports under **Resource Section** in the **Fitter** section of the Compilation Report. The **Fitter Resource Usage Summary** report breaks down the logic utilization information and indicates the number of fully and partially used ALMs, and provides other resource information including the number of bits in each type of memory block. There are also reports that describe some of the optimizations that occurred during compilation. For example, if you are using Quartus II integrated synthesis, the reports under the **Optimization Results** folder in the **Analysis & Synthesis** section provide 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.

113. ☐   Review resource utilization 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, you can use the Message window tabs to look at only certain types of messages, and you can suppress messages if you have determined that they do not require any action from you.

114. ☐   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 1 of the *Quartus II Handbook*.

Timing Constraints and Analysis

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

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

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

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

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

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

For more information about timing analysis, refer to *The Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Recommended Timing Optimization and Analysis Assignments

The assignments and settings described in this section are important for large designs such as those in Stratix IV devices.

118.☐ Turn on **Optimize fast-corner timing** on the **Fitter Settings** page in the **Settings** dialog box.

When this option is on, the design is optimized to meet its timing requirements at the Fast Timing process corner and operating condition, as well as at the Slow Timing corner. Therefore, turning on this option helps create a design implementation that is more robust across process, temperature, and voltage variations.

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

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

120.☐ Use `set_input_delay`, `set_output_delay` to specify external device or board timing parameters.

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

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

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

👣 For more guidelines about timing constraints, refer to *The Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Area and Timing Optimization

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

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

You can use the Early Timing Estimation feature to estimate your design's timing results before the software performs full placement and routing. On the Processing menu, point to **Start** and click **Start Early Timing Estimate** to generate initial compilation results after you have run analysis and synthesis.

124. ☐ 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 do improve the results for your design. If you do not require these options to meet your design timing requirements, turn off the options to reduce compilation time.

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

The Design Space Explorer (DSE) is a utility that automates the process of finding the optimal collection of Quartus II software settings for your design. The **Search for Best Performance** and **Search for Best Area** options under **Exploration Settings** use a predefined exploration space to target design performance or area improvements with multiple compilations. You can also set the **Optimization Goal** to **Optimize for Speed** or **Optimize for Area** using the **Advanced** tab in the DSE window. If you are interested in optimization for power usage, refer to "Power Optimization" on page 49.

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

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

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

126.☐   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.

127.☐   Use incremental compilation to preserve performance for 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 40.

128.☐   Ensure parallel compilation is enabled 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. Set the **Parallel compilation** option on the **Compilation Process Settings** page of the **Settings** dialog box, or the default setting in the **Options for Processing** from the Tools menu.

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

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

For more suggestions, refer to the *Reducing Compilation Time* 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 the placement and routing information. You can perform timing simulation for the top-level design at the end of your design flow to ensure that your design works in the targeted device.

Altera provides the ModelSim-Altera simulator Starter Edition and offers the higher-performance ModelSim-Altera Edition, which enable 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. The software can also generate scripts to help you setup libraries in your tool with NativeLink integration.

130. ☐ 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 Release Notes* lists the version of each simulation tool that is officially supported with that particular version of the Quartus II software. Use the model libraries provided with your Quartus II software version, because libraries can change between versions, which might cause a mismatch with your simulation netlist. 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 chapters in volume 3 of the *Quartus II Handbook*.

Formal Verification

The Quartus II software supports some formal verification flows. 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. There are other restrictions that can also limit your design; consult the documentation for details. If formal verification is important to your design, it is easier to plan for limitations and restrictions in the beginning than to make changes later in the design flow.

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

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

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

Power Analysis

Before design completion, estimate power consumption using a spreadsheet as described in "Early Power Estimation" on page 7. 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.

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

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

You must compile a design (to provide information about design resources, placement and routing, and I/O standards) and provide signal activity data (toggle rates and static probabilities) to use the PowerPlay Power Analyzer. You can derive signal activity data from simulation results or a user-defined default toggle rate and vectorless estimation. The signal activities used for the analysis must be representative of the actual operating behavior. For the most accurate power estimation, use gate-level simulation results with a **.vcd** output file from 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. Refer to "Simulation" on page 47 for information about third-party simulators.

135. ☐ Specify correct operating conditions for power analysis.

You must also specify 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 data sheet for your device.

For more information about power analysis, and recommendations for simulation settings for creating signal activity information, refer to the *PowerPlay Power Analysis* chapter in volume 3 of the *Quartus II Handbook*. For recommendations and guidelines for power management system design, including creating a power budget spreadsheet, refer to *AN 448: Stratix III Power Management Design Guide*.

Power Optimization

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

For specific architectural features that help reduce power consumption, as well as design optimization techniques, refer to *AN 514: Power Optimization in Stratix IV FPGAs*.

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

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

Device and Design Power Optimization Techniques

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

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

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

For more details and additional design techniques to reduce power consumption, refer to the *Power Optimization* chapter in volume 2 of the *Quartus II Handbook* and *AN 514: Power Optimization in Stratix IV FPGAs*.

Device Speed Grade

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

Clock Power Management

Clocks represent a significant portion of dynamic power consumption, 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 does not toggle, thereby reducing the overall power consumption of the device.

For more information about using clock control blocks, refer to the *Clock Control Block (ALTCLKCTRL) IP Core 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. You can use clock gating described in "Clock Power Management" or the clock enable signals in the memory ports.

I/O Power Guidelines

The dynamic power consumed in the I/O buffer is proportional to the total load capacitance; therefore, 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; therefore, 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 calculations, so be sure to include it separately in your system power calculations.

Quartus II Power Optimization Techniques

The Quartus II software offers power-optimized synthesis and fitting to reduce core dynamic power. Power-driven compilation works in conjunction with Programmable Power Technology in the Stratix IV silicon.

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

👣 For 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 is a utility that automates the process of finding the optimal collection of Quartus II software settings for your design. You can target overall design power improvements with multiple compilations.

For more information, refer to the *Design Optimization Overview* 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.

Conclusion

The design guidelines in this application note provide important factors to consider in high-density, high-performance Stratix IV designs. It is important to follow Altera's recommendations throughout the design process to achieve good results, avoid common issues, and improve your design productivity. The "Design Checklist" on page 55 summarizes the checklist items presented in this document. You can use this separate checklist to ensure that you have reviewed all the guidelines before completing your Stratix IV design.

Referenced Documents

This application note references the following documents:

■ *AN 370: Using the Altera Serial Flash Loader Megafunction with the Quartus II Software*

■ *AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming*

■ *AN 448: Stratix III Power Management Design Guide*

■ *AN 471: High-Performance FPGA PLL Analysis with TimeQuest*

■ *AN 514: Power Optimization in Stratix IV FPGAs*

■ *AN 528: PCB Dielectric Material Selection and Fiber Weave Effect on High-Speed Channel Routing*

■ *AN 529: Via Optimization Techniques for High-Speed Channel Designs*

■ *AN 530: Optimizing Impedance Discontinuity Caused by Surface Mount Pads for High-Speed Channel Designs*

■ *AN 557: Stratix III-to-Stratix IV E Cross-Family Migration Guidelines*

■ *Analyzing and Optimizing the Design Floorplan with the Chip Planner* chapter in volume 2 of the *Quartus II Handbook*

■ *Area Optimization* chapter in volume 2 of the *Quartus II Handbook*

■ *Best Practices for Incremental Compilation Partitions and Floorplan Assignments* chapter in volume 1 of the *Quartus II Handbook*

■ *The Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

■ *ByteBlaster II Download Cable User Guide*

- *Clock Control Block (ALTCLKCTRL) IP Core User Guide*

- *Clock Networks and PLLs in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*

- *Configuration, Design Security, and Remote System Upgrades in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*

- *DC and Switching Characteristics for Stratix IV Devices* chapter in volume 4 of the *Stratix IV Device Handbook*

- *Recommended Design Practices* chapter in volume 1 of the *Quartus II Handbook*

- *Design Optimization Overview* chapter in volume 2 of the *Quartus II Handbook*

- *EthernetBlaster Communications Cable User Guide*

- *External Memory PHY Interface (ALTMEMPHY) (nonAFI) Megafunction User Guide*

- *External Memory Interfaces in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*

- *High-Speed Differential I/O Interfaces and DPA in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*

- *Hot Socketing and Power-On Reset in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*

- *I/O Features in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*

- *Managing Device I/O Pins* chapter in volume 2 of the *Quartus II Handbook*

- *JTAG Boundary Scan Testing in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*

- *Logic Array Blocks and Adaptive Logic Modules in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*

- *Managing Quartus II Projects* chapter in volume 1 of the *Quartus II Handbook*

- *Mentor Graphics PCB Design Tools Support* and *Cadence PCB Design Tools Support* chapters in volume 2 of the *Quartus II Handbook*

- *Netlist Optimizations and Physical Synthesis* chapter in volume 2 of the *Quartus II Handbook*

- *Parallel Flash Loader IP Core User Guide*

- *ALTPLL (Phase-Locked Loop) IP Core User Guide*

- *Phase-Locked Loop Reconfiguration (ALTPLL_RECONFIG) IP Core User Guide*

- *Device-Specific Power Delivery Network (PDN) Tool User Guide*

- *Power Management in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*

- *Power Optimization* chapter in volume 2 of the *Quartus II Handbook*

- *PowerPlay Early Power Estimator User Guide*

- *PowerPlay Power Analysis* chapter in volume 3 of the *Quartus II Handbook*

- *Quartus II Incremental Compilation for Hierarchical and Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*

- *Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*

- *Quartus II Release Notes*

- *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*

- *Altera Remote Update IP Core User Guide*

- *Serial Configuration (EPCS) Devices Datasheet* in volume 2 of the *Configuration Handbook*

- *SEU Mitigation in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Handbook*

- *Signal Integrity Analysis with Third-Party Tools* chapter in volume 2 of the *Quartus II Handbook*

- *Altera Virtual JTAG (altera_virtual_jtag) IP Core User Guide*

- *Overview for the Stratix IV Device Family* chapter in volume 1 of the *Stratix IV Device Handbook*

- *Errata Sheet for Stratix IV GT Devices*

- *Errata Sheet for Stratix IV GX Devices*

- *Stratix IV GX Device Family Pin Connection Guidelines*

- *Transceiver Architecture in Stratix IV Devices* chapter in volume 2 of the *Stratix IV Device Handbook*

- *Technical Brief 92: High-Speed Board Design Advisor (Power Distribution Network)*

- *Section I. Transceiver Architecture* in volume 2 of the *Stratix IV Handbook*

- *Section I. Transceiver Configuration Guide* in volume 3 of the *Stratix IV Handbook*

- *USB-Blaster Download Cable User Guide*

- *SOPC Builder User Guide*

Document Revision History

Table 3 shows the revision history for this application note.

Table 3. Document Revision History

Date and Version Number	Changes Made	Summary of Changes
February 2015, v1.2	■ Updated "Device Power-Up" on page 16 ■ Updated "Unused Pins" on page 22	■ Removed statements about power sequencing. ■ Revised guidelines for dealing with unused pins. ■ Updated links to external documents. ■ Replaced "MegaWizard Plug-In Manager" with "IP Catalog".
May 2009, v1.1	■ Updated "Device Family Variant and High-Speed Transceivers" on page 4 ■ "Device Power-Up" on page 16 ■ "Power Pin Connections and Power Supplies" on page 17 ■ Added "High-Speed Board Design" on page 22 ■ Updated "SOPC Builder" on page 3 ■ Updated "Document Revision History" on page 53	■ Removed incorrect Quartus II Simulator support reference. ■ Updated references and added document. ■ Removed incorrect reference to 3.3 V V_{CCPD} for JTAG pins. ■ Updated Pin leakage current information. ■ Updated device family variant options.
August 2008, v1.0	Initial release.	—

Design Checklist

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

Done N/A

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 IP that affects system design, especially I/O interfaces.

4. ☐ ☐ If you plan to use the Open Core Plus tethered mode for IP, ensure that your board design supports this mode of operation.

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

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

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

8. ☐ ☐ Consider vertical device migration availability and requirements.

9. ☐ ☐ If you want to migrate to a HardCopy IV ASIC, review the appropriate design considerations.

10. ☐ ☐ Estimate power consumption with the Early Power Estimator spreadsheet to plan the cooling solution and power supplies before the logic design is complete.

11. ☐ ☐ Set up the temperature sensing diode in your design to measure the device junction temperature for thermal management.

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

13. ☐ ☐ If you want to use AS configuration mode with large device densities, confirm there is a configuration device available that is large enough for your target FPGA density.

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

15. ☐ ☐ Ensure your configuration scheme and board support any required features: data decompression, design security, remote upgrades, SEU mitigation.

16. ☐ ☐ Plan board design to support optional configuration pins `CLKUSR` and `INIT_DONE`, as required.

17. ☐ ☐ Plan board design to use the Auto-restart after configuration error option.

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

19. ☐ ☐ Select on-chip debugging scheme(s) early to plan memory and logic requirements, I/O pin connections, and board connections.

Done   N/A

20. ☐   ☐   If you want to use the SignalTap II Embedded Logic Analyzer, Logic Analyzer Interface, In-System Memory Content Editor, In-System Sources and Probes, or Virtual JTAG IP core, plan your system and board with JTAG connections that are available for debugging.

21. ☐   ☐   Plan for the small amount of additional logic resources used to implement the JTAG hub logic for JTAG debugging features.

22. ☐   ☐   For debugging with the SignalTap II Embedded Logic Analyzer, reserve device memory resources to capture data during system operation.

23. ☐   ☐   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 debugging signals later.

24. ☐   ☐   Ensure the board supports a debugging mode where debugging signals do not affect system operation.

25. ☐   ☐   Incorporate pin header or mictor connector as required for an external logic analyzer or mixed signal oscilloscope.

26. ☐   ☐   To use debug tools incrementally and reduce compilation time, ensure incremental compilation is on so you do not have to recompile the design to modify the debug tool.

27. ☐   ☐   To use the Virtual JTAG IP core for custom debugging applications, instantiate it in the HDL code as part of the design process.

28. ☐   ☐   To use the In-System Sources and Probes feature, instantiate the IP core in the HDL code.

29. ☐   ☐   To use the In-System Memory Content Editor for RAM or ROM blocks or the LPM_CONSTANT IP core, 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 IP Catalog.

30. ☐   ☐   Design board for power-up: Stratix IV output buffers are tri-stated until the device is configured, and configuration pins drive out.

31. ☐   ☐   Design voltage supply power ramps to be monotonic.

32. ☐   ☐   Set POR time to ensure power supplies are stable.

33. ☐   ☐   Connect all power pins correctly: 0.9-V core, 1.5-V, 2.5 V, and others as required.

34. ☐   ☐   Connect V_{CCIO} pins and V_{REF} pins to support each bank's I/O standards.

35. ☐   ☐   Connect the VCCPD pin to 3.0-V for a 3.0-V V_{CCIO}, and 2.5-V for lower I/O voltages.

36. ☐   ☐   Explore any unique requirements for FPGA power pins or other power pins on your board, and determine which devices on your board can share a power rail.

37. ☐   ☐   Use a linear regulator to power analog circuitry (V_{CCA_PLL} and V_{CCPT}). You can power the digital voltage rails by linear or switching regulators depending on efficiency or cost considerations.

38. ☐   ☐   Follow the specific guidelines for each pin in the Stratix IV GX Device Family Pin Connection Guidelines.

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

Done **N/A**

40. ☐ ☐ Connect all PLL power pins to reduce noise even if the design does not use all the PLLs: V_{CCA_PLL} to 2.5-V and V_{CCD_PLL} to 0.9-V.

41. ☐ ☐ Run a thick trace (at least 20 mils) from the power supply to each V_{CCA_PLL} pin.

42. ☐ ☐ Connect all V_{CCD_PLL} power pins to the quietest digital supply on the board.

43. ☐ ☐ Use ferrite beads to isolate the PLL power supply from the digital power supply

44. ☐ ☐ Check that all configuration pin connections and pull-up/pull-down resistors are set correctly for your configuration scheme(s).

45. ☐ ☐ Design configuration `DCLK` and `TCLK` pins to be noise-free.

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

47. ☐ ☐ Connect JTAG pins correctly to the download cable header. Ensure the pin order is not reversed.

48. ☐ ☐ To disable the JTAG state machine during power-up, pull the `TCK` pin low through a 1-kΩ resistor to ensure that an unexpected rising edge does not occur on `TCK`.

49. ☐ ☐ Pull `TMS` and `TDI` high through a 10-kΩ resistor.

50. ☐ ☐ Connect `TRST` directly to V_{CCPD} (Connecting the pin to ground disables the JTAG circuitry).

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

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

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

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

55. ☐ ☐ Connect `nIO_PULLUP` correctly to set up internal pull-up resistors.

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

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

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

59. ☐ ☐ Carefully check the pin connections in the Quartus II software-generated .pin file. Do not connect RESERVED pins.

60. ☐ ☐ Design V_{REF} pins to be noise-free.

61. ☐ ☐ Break out large bus signals on board layers close to the device to reduce cross talk.

62. ☐ ☐ Route traces orthogonally if two signal layers are next to each other, if possible. Use a separation of 2 to 3 times the trace width.

Done   N/A

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

64. ☐   ☐   Perform board-level simulation using IBIS models (when available).

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

66. ☐   ☐   Use the Quartus II Pin Planner to make pin assignments.

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

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

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

70. ☐   ☐   Plan signaling type based on system requirements.

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

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

73. ☐   ☐   Ensure that appropriate I/O standard support is supported in targeted I/O bank.

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

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

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

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

78. ☐   ☐   Use the ALTMEMPHY IP core for each memory interface, and follow connection guidelines/restrictions in the appropriate documentation.

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

80. ☐   ☐   Make dual-purpose pin settings, and check for any restrictions when using these pins as regular I/O.

81. ☐   ☐   Check 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 VOD.

82. ☐   ☐   Consider on-chip termination features to save board space.

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

84. ☐   ☐   Choose the appropriate mode of DPA, non-DPA, or soft-CDR for high-speed LVDS interfaces.

85. ☐   ☐   If you want to use DPA, be sure to enable the feature so that the design uses the correct PLLs on the right and left sides of the device, and follow DPA placement guidelines.

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

Done **N/A**

87. ☐ ☐ Use the device PLLs for clock management.

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

89. ☐ ☐ Enable PLL features and check settings in the IP Catalog.

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

91. ☐ ☐ Check that the PLL offers the required number of clock outputs and use dedicated clock output pins.

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

93. ☐ ☐ Analyze design for possible simultaneous switching noise problems.

94. ☐ ☐ Reduce number of pins that switch voltage at exactly the same time whenever possible.

95. ☐ ☐ Use differential I/O standards and lower-voltage standards for high switching I/Os.

96. ☐ ☐ Use lower drive strengths for high switching I/Os. The default drive strength setting might be higher than your design requires.

97. ☐ ☐ Reduce number of simultaneously switching output pins within each bank. Spread output pins across multiple banks if possible.

98. ☐ ☐ 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%).

99. ☐ ☐ Separate simultaneously switching pins from input pins that are susceptible to SSN.

100. ☐ ☐ Place important clock and asynchronous control signals near ground signals and away from large switching buses.

101. ☐ ☐ Avoid using I/O pins one or two pins away from PLL power supply pins for high-switching or high drive strength pins.

102. ☐ ☐ Use staggered output delays to shift the output signals through time, or use adjustable slew rate settings.

103. ☐ ☐ Use synchronous design practices. Pay attention to clock signals.

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

105. ☐ ☐ Use megafunctions with the IP Catalog.

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

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

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

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

Done   N/A

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

111. ☐   ☐   Create a design floorplan for incremental compilation partitions, if required for your design flow.

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

113. ☐   ☐   Review resource utilization reports after compilation.

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

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

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

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

118. ☐   ☐   Turn on Optimize fast-corner timing on the Fitter Settings page in the Settings dialog box.

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

120. ☐   ☐   Use `set_input_delay`, `set_output_delay` to specify external device or board timing parameters.

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

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

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

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

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

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

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

128. ☐   ☐   Ensure parallel compilation is enabled if you have multiple processors available for compilation.

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

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

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

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

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

Done **N/A**

134. ☐ ☐ Provide accurate typical signal activities, preferably with a gate-level simulation .vcd file, to get accurate power analysis results.

135. ☐ ☐ Specify correct operating conditions for power analysis.

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

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