Arria V and Cyclone V Design Guidelines

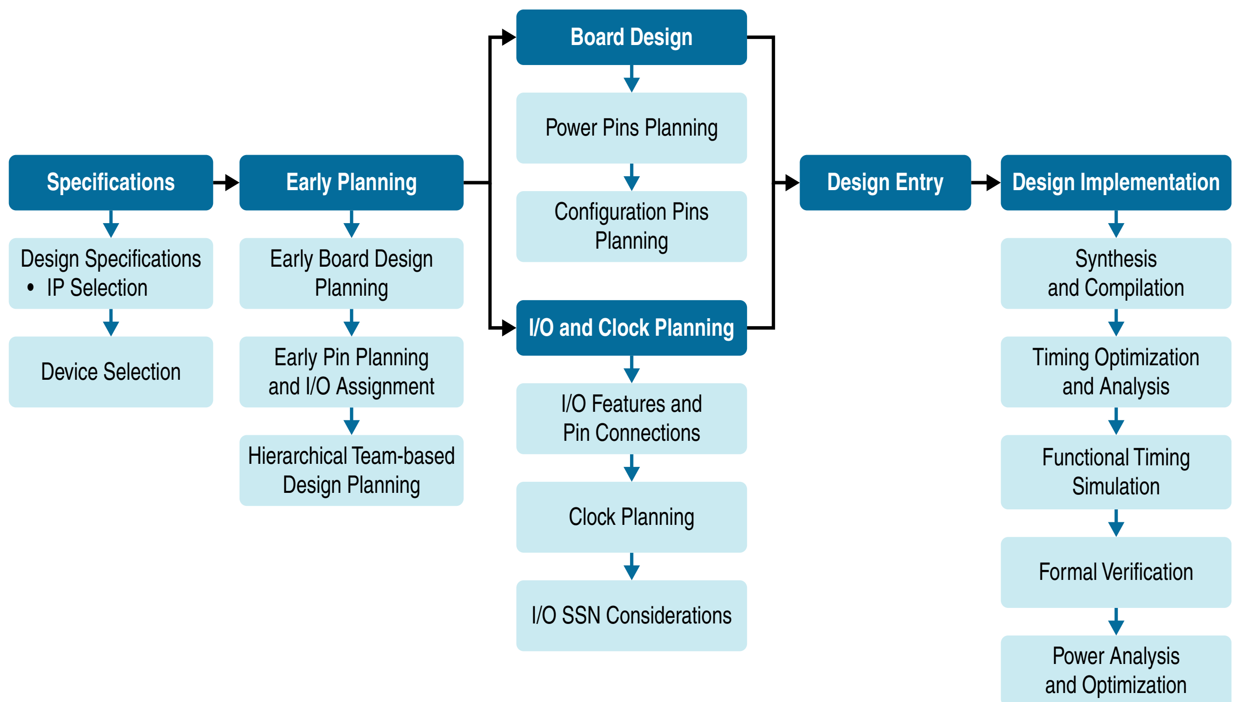
AN-662-1.0 Application Note

This application note provides a set of checklists that consist of design guidelines, recommendations, and factors to consider when you create designs using the Arria® V or Cyclone® V FPGAs.

- Use this document to help you plan the FPGA and system early in the design process, which is crucial for a successful design.
- Follow Altera's recommendations throughout the design process to achieve good results, avoid common issues, and improve your design productivity.

Figure 1 shows the Arria V and Cyclone V design flow. The sections in this document provide the checklists and guidelines for each part of the design flow.

Figure 1. Arria V and Cyclone V Design Flow



For the Arria V and Cyclone V SoC FPGA device variants, the guidelines in this document are applicable only to the FPGA portion of the devices.

101 Innovation Drive
San Jose, CA 95134
www.altera.com

Before You Begin

Before you begin planning and designing your FPGA system, familiarize yourself with the FPGA device features, and the design tools and IP that are available for the Arria V or Cyclone V device family.

Table 1. Prerequisites Checklist (Part 1 of 2)

No.	✓	Checklist items
1.	☐	*Read through the Device Overview of the FPGA* The Device Overview provides an overview of the capabilities and options available for a device family. Read through the document to familiarize yourself with the device family offerings and general features. For an overview of each FPGA device family, refer to the following documents: ■ *Arria V Device Overview* ■ *Cyclone V Device Overview*
2.	☐	*Estimate design requirements* Create a rough estimate of the design in the following terms: ■ Basic functions of the product ■ Similar previous designs ■ General device requirements
3.	☐	*Review available design tools* Consider the available design, estimators, system builders, and verification tools. The following items are some of the available tools provided by Altera: ■ The Quartus® II software for design, synthesis, simulation, and programming; including integration with Qsys, simulation tools, and verification tools. ■ The Qsys system integration tool—next generation SOPC Builder that automatically generates interconnect logic to connect intellectual property (IP) functions and subsystems. ■ The Mentor Graphics® ModelSim®-Altera® simulation software. ■ The TimeQuest Timing Analyzer for static timing analysis with support for Synopsys® Design Constraints (SDC) format. ■ The PowerPlay Power Analyzer for power analysis and optimization. ■ The SignalProbe and SignalTap II Logic Analyzer debugging tools. ■ The External Memory Interface Toolkit available in the Quartus II software. ■ The Transceiver Toolkit for real-time validation of transceiver link signal integrity. For more information, visit the following pages on the Altera website: ■ Design Tools & Services ■ Design Software Support ■ Transceiver Toolkit For a guideline to migrate from SOPC Builder to Qsys, refer to *AN 632: SOPC Builder to Qsys Migration Guidelines*.

Table 1. Prerequisites Checklist (Part 2 of 2)

No.	✓	Checklist items
4.	☐	*Review available IP* Altera and its third-party IP partners offer a large selection of parameterized blocks of IP cores optimized for Altera devices that you can implement to reduce your implementation and verification time. Based on your estimated requirements, refer to the All Intellectual Property page on Altera's website to check if there are available IP that can provide the functions that you need.

Design Specifications

Typically, the FPGA is an important part of the overall system and affects the rest of the system design. Use the following checklist to start your design process.

Table 2. Design Specifications Checklist (Part 1 of 2)

No.	✓	Checklist items
1.	☐	*Create detailed design specifications* Before you create your logic design or complete your system design, perform the following: ■ Specify the I/O interfaces for the FPGA ■ Identify the different clock domains ■ Include a block diagram of basic design functions ■ Consider a common design directory structure—if your design includes multiple designers, a common design directory structure eases the design integration stages.
2.	☐	*Create detailed functional verification or test plan* A functional verification plan ensures the team knows how to verify the system. Creating a test plan at an early stage helps you design for testability and manufacturability. For example, if you plan to perform built-in-self test (BIST) functions to drive interfaces, you can plan to use a UART interface with a Nios® II processor inside the FPGA device. For more information, refer to "Review available on-chip debugging tools" on page 10.
3.	☐	*Select IP that affects system design, especially I/O interfaces* Include intellectual property (IP) blocks in your detailed design specifications. Taking the time to create these specifications improves design efficiency. For a list of available IP offered by Altera and third-party IP partners, refer to the All Intellectual Property page on Altera's website.

Table 2. Design Specifications Checklist (Part 2 of 2)

No.	✓	Checklist items
4.	☐	*Ensure your board design supports the OpenCore Plus tethered mode* You can program your FPGA and verify your design in hardware before you purchase an IP license by using the OpenCore Plus feature available for many IP cores. OpenCore Plus supports the following modes: ■ Untethered—your design runs for a limited time. ■ Tethered—your design runs for the duration of the hardware evaluation period. This mode requires an Altera download cable connected to the JTAG port on your board and a host computer that runs the Quartus II Programmer. If you plan to use this mode, ensure that your board design supports this mode.
5.	☐	*Review available system development tools* For more information, visit the following pages on the Altera website: ■ Design Tools & Services ■ Design Software Support

Device Selection

Use the following checklist to determine the device variant, density, and package combination that is suitable for your design.

Table 3. Device Selection Checklist (Part 1 of 3)

No.	✓	Checklist items
1.	☐	*Consider the available device variants* The Arria V and Cyclone V device families consist of several device variants that are optimized for different application requirements. 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, refer to the following documents: ■ *Arria V Device Overview* ■ *Cyclone V Device Overview*
2.	☐	*Estimate the required logic, memory, and multiplier density* Arria V or Cyclone V devices offer a range of densities that provide different amounts of device logic resources. 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.

Table 3. Device Selection Checklist (Part 2 of 3)

No.	✓	Checklist items
3.	☐	*Consider vertical device migration availability and requirements* Determine whether you want the flexibility 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. To verify the pin migration compatibility, use the **Pin Migration View** window in the Quartus II software Pin Planner. The **Pin Migration View** window helps you identify the difference in pins that can exist between migration devices: ■ If one device has pins for connection to V_{CC} or GND but are I/O pins on a different device, the Quartus II software ensures these pins are not used for I/O. For migration, ensure that these pins are connected to the correct PCB plane. ■ If you are migrating between two devices in the same package, connect the pins that are not connected to the smaller die to V_{CC} or GND on the larger die in your original design. For more information about verifying the pin migration compatibility, refer to the "I/O Management" chapter of the *Quartus II Handbook*.
4.	☐	*Review resource utilization reports of similar designs* If you have other designs that target an Altera device, you can use their resource utilization as an estimate for your new design. Coding style, device architecture, and optimization options used in the Quartus II software can significantly affect resource utilization and timing performance of a design. To estimate resource utilization for certain configurations of Altera's IP designs, refer to the relevant Altera megafunctions and IP MegaCores user guides at the IP and Megafunctions page on the Altera website.
5.	☐	*Reserve device resources for future development and debugging* Select a device that meets your design requirements with some safety margin in case you want to add more logic later in the design cycle, upgrade, or expand your design. You may also want additional space in the device to ease design floorplan creation for an incremental or team-based design. Consider reserving resources for debugging, as described in "Consider the guidelines to plan for debugging tools" on page 10.
6.	☐	*Estimate the number of I/O pins that you require* Determine the required number of I/O pins for your application, considering the design's interface requirements with other system blocks. You can compile any existing designs in the Quartus II software to determine how many I/O pins are used. 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 relevant topics under "Board Design" on page 7 and "I/O and Clock Planning" on page 20.
7.	☐	*Consider the I/O pins you need to reserve for debugging* Consider reserving I/O pins for debugging, as described in "Consider the guidelines to plan for debugging tools" on page 10.

Table 3. Device Selection Checklist (Part 3 of 3)

No.	✓	Checklist items
8.	☐	*Verify that the number of LVDS channels are enough* Larger densities and package pin counts offer more full-duplex LVDS channels for differential signaling. Ensure that your device density-package combination includes enough LVDS channels.
9.	☐	*Verify the number of PLLs and clock routing resources* Verify that your chosen device density package combination includes enough PLLs and clock routing resources for your design. GCLK resources are shared between certain PLLs, which can affect the inputs that 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 20.
10.	☐	*Determine the device speed grade that you require* The device speed grade affects the device timing performance and timing closure, as well as power utilization. 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, use the estimator tool on the External Memory Interface Spec Estimator page. You can use the fastest speed grade while prototyping to reduce compilation time because less time is spent optimizing the design to meet timing requirements. If the design meets the timing requirements, you can then move to a slower speed grade for production to reduce cost. For information about the available speed grades, refer to the following documents: ■ *Arria V Device Datasheet* ■ *Cyclone V Device Datasheet*

Board Design

Use the checklists in this section as guidelines to design your board.

Early Board Design

Early planning allows the FPGA team to provide early information to PCB board and system designers.

Table 4. Early Board Design Planning Checklist (Part 1 of 5)

No.	✓	Checklist items
1.	☐	*Review the available configuration schemes* You can configure the Arria V or Cyclone V devices with one of several configuration schemes. For the list of available configuration schemes for each device family, refer to the "Enhanced Configuration and Configuration via Protocol" sections in the following documents: ■ *Arria V Device Overview* ■ *Cyclone V Device Overview*
2.	☐	*Select a configuration scheme* For information about the configuration schemes, the execution of the required configuration schemes, and the necessary and optional pin settings, including the MSEL pin settings, refer to the following documents: ■ *Configuration, Design Security, and Remote System Upgrades in Arria V Devices* ■ *Configuration, Design Security, and Remote System Upgrades in Cyclone V Devices* For more information about selecting the configuration schemes, refer to the *Configuring Altera FPGAs* document.

Table 4. Early Board Design Planning Checklist (Part 2 of 5)

No.	✓	Checklist items
3.	☐	*Consider the configuration device support and availability* ■ All configuration schemes use a configuration device, a download cable, or an external controller (for example, a MAX® II device or microprocessor). For the active serial (AS) configuration scheme, you can use the Altera serial configuration devices (EPCS) and quad-serial configuration devices (EPCQ). ■ The Quartus II programmer supports configuration of devices directly using PS or JTAG interfaces with the USB-Blaster™, EthernetBlaster II, or the ByteBlaster™ II download cable. You can download design changes directly to the device and 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. ■ Serial configuration devices do not directly support the JTAG interface but 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. However, programming the EPCS using the SFL solution is slower than standard AS interface because it must configure the FPGA before programming EPCS or EPCQ configuration devices. ■ If your system contains common flash interface (CFI) flash memory, you can use it for device configuration storage. You can program CFI flash memory devices through the JTAG interface with the parallel flash loader (PFL) megafunction in MAX II and MAX V devices. The PFL can also control configuration from the flash memory device to the Arria V or Cyclone V device and it supports data compression. PS and FPP configuration modes are supported using PFL. ■ 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—a software driver developed for embedded serial configuration device programming that designers can customize to fit in different embedded systems. For a list of documents about configuration devices, including the SRunner software and Altera download cables, refer to the Configuration Devices page on the Altera website.

Table 4. Early Board Design Planning Checklist (Part 3 of 5)

No.	✓	Checklist items
4.	☐	*Ensure the configuration scheme and board support the required features:* ■ Data decompression—if you enable data compression, the storage requirement and the time to transmit the configuration bitstream are reduced. The `DCLK` to `DATA` ratio changes accordingly, based on the selected FPP configuration scheme. For successful configuration, the configuration controller must send the `DCLK` that meets the `DCLK` to `DATA` ratio. ■ Design security—this feature utilizes a 256-bit security key. The devices can decrypt configuration bitstreams using the AES algorithm that is FIPS-197 certified. Design security is available for the FPP, AS, or PS configuration schemes but not available for the JTAG configuration scheme. ■ Remote system upgrade—this feature is supported only in the single-device AS configuration scheme with EPCS and EPCQ devices. To implement the remote system upgrade interface, use the ALTREMOTE_UPDATE megafunction. ■ SEU mitigation—dedicated circuitry in the devices perform cyclic redundancy check (CRC) error detection and optionally check for SEU errors automatically. To detect SEU errors, use the `CRC_ERROR` pin to flag errors and design your system to take appropriate action. If you do not enable the CRC error detection feature, you can also use the `CRC_ERROR` pin as a design I/O pin. For more information, refer to the following documents: ■ *Configuration, Design Security, and Remote System Upgrades in Arria V Devices* ■ *Configuration, Design Security, and Remote System Upgrades in Cyclone V Devices* ■ *Remote System Upgrade (ALTREMOTE_UPDATE) Megafunction User Guide* ■ *SEU Mitigation in Arria V Devices* ■ *SEU Mitigation in Cyclone V Devices*
5.	☐	*Plan for support of optional configuration pins (*`CLKUSR` *and* `INIT_DONE`*), if required* You can enable the following optional configuration pins: ■ `CLKUSR`—the **Enable user-supplied start-up clock (CLKUSR)** option allows you to select the clock source to use for initialization: the internal oscillator or an external clock provided on the `CLKUSR` pin. `CLKUSR` also allow you to drive the AS configuration clock (`DCLK`) at 125 MHz maximum. In the Quartus II software, enable this feature in the **Configuration** page of the **Device and Pins Option** dialog box. ■ `INIT_DONE`—you can monitor the `INIT_DONE` pin to check if the device has completed initialization and is in user mode. The `INIT_DONE` pin is an open drain output and requires an external pull-up to V_{CCPGM}. During reset, after the device exits POR and in the beginning of configuration, the `INIT_DONE` pin is tri-stated and pulled high by the external pull-up resistor. To enable the `INIT_DONE` pin, turn on the **Enable INIT_DONE output** option.
6.	☐	*Plan for the* ***Auto-restart after configuration error*** *option* To reset the device internally by driving the `nSTATUS` pin low when a configuration error occurs, enable the **Auto-restart after configuration error** option. The device releases its `nSTATUS` pin after the reset time-out period. This behavior allows you to re-initiate the configuration cycle. The `nSTATUS` pin requires an external 10-kΩ pull-up resistor to V_{CCPGM}.

Table 4. Early Board Design Planning Checklist (Part 4 of 5)

No.	✓	Checklist items
7.	☐	*Review available on-chip debugging tools* Take advantage of on-chip debugging features to analyze internal signals and perform advanced debugging techniques. Different debugging tools work better for different systems and different designers. Early planning can reduce the time spent debugging, and eliminates design changes later to accommodate your preferred debugging methodologies. Adding debug pins may not be enough, because of internal signal and I/O pin accessibility on the device. For more information about in-system debugging tools in the Quartus II software, refer to the following documents: ■ *System Debugging Tools Overview* in the *Quartus II Handbook* ■ *Virtual JTAG (sld_virtual_jtag) Megafunction User Guide*
8.	☐	*Consider the guidelines to plan for debugging tools* ■ Select on-chip debugging schemes early to plan memory and logic requirements, I/O pin connections, and board connections. ■ If you want to use SignalProbe incremental routing, the SignalTap II Embedded Logic Analyzer, Logic Analyzer Interface, In-System Memory Content Editor, In-System Sources and Probes, or Virtual JTAG megafunction, plan your system and board with JTAG connections that are available for debugging. ■ Plan for the small amount of additional logic resources used to implement the JTAG hub logic for JTAG debugging features. ■ For debugging with the SignalTap II Embedded Logic Analyzer, reserve device memory resources to capture data during system operation. ■ Reserve I/O pins for debugging with SignalProbe or the Logic Analyzer Interface so that you do not have to change the design or board to accommodate debugging signals later. ■ Ensure the board supports a debugging mode where debugging signals do not affect system operation. ■ Incorporate a pin header or micro connector as required for an external logic analyzer or mixed signal oscilloscope. ■ 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. ■ To use the Virtual JTAG megafunction for custom debugging applications, instantiate it in the HDL code as part of the design process. ■ To use the In-System Sources and Probes feature, instantiate the megafunction in the HDL code. ■ To use the In-System Memory Content Editor for RAM or ROM blocks or the LPM_CONSTANT megafunction, turn on the **Allow In-System Memory Content Editor** option to capture and update content independently of the system clock option for the memory block in the MegaWizard Plug-In Manager.

Table 4. Early Board Design Planning Checklist (Part 5 of 5)

No.	✓	Checklist items
9.	☐	*Use the PowerPlay Early Power Estimator (EPE) to estimate power supplies and cooling solution* FPGA power consumption depends on logic design and is challenging to estimate during early board specification and layout. However, it is an important design consideration and must be estimated accurately to develop an appropriate power budget to design the power supplies, voltage regulators, decoupling capacitors, heat sink, and cooling system. Use the Altera PowerPlay EPE spreadsheet to estimate power, current, and device junction temperature before your have a complete design. The EPE calculates the estimated information based on device information, planned device resources, operating frequency, toggle rates, ambient temperature, heat sinks information, air flow, board thermal model, and other environmental considerations. ■ If you have an existing or partially-completed and compiled design—use the **Generate PowerPlay Early Power Estimator File** command in the Quartus II software to provide input to the EPE spreadsheet. ■ If you do not have an existing design—estimate manually the number of device resources used in your design and input into the EPE spreadsheet. If the device resources information changes during or after the design phase, your power estimation results will be less accurate. For more information, the EPE user guide, and to download the appropriate PowerPlay EPE spreadsheet for your device, visit the PowerPlay Early Power Estimators (EPE) and Power Analyzer page on the Altera website. For guidelines about proper power supply design, refer to "Use PDN tool to plan for power distribution and decoupling capacitor selection" on page 14.
10.	☐	*Review the transceiver design guidelines* The Quartus II software support models for Arria V and Cyclone V transceivers takes into account the way designers use processors to handle data flow. In the Quartus II software, high-speed transceivers are represented by PHY IP cores. Instead of the PHY IP cores, the transceiver voltage, termination, and PLL settings are handled by the Quartus II Settings File (.qsf). For more information about the Arria V and Cyclone V transceivers, refer to the following documents: ■ *Arria V Device Handbook, Volume 2: Transceivers* ■ *Cyclone V Device Handbook, Volume 2: Transceivers* PHY IP design is modular and uses standard interfaces. All PHY IP include an Avalon® Memory-Mapped (Avalon-MM) interface or conduit interface to access the control and status registers, and an Avalon Streaming (Avalon-ST) interface to connect to the MAC layer design for data transfer. For more information, refer to the following documents: ■ *Altera Transceiver PHY IP Core User Guide* ■ *Avalon Interface Specifications* For a description of the requirements for simulating a transceiver design using the custom PHY IP core, moving to a design, and how to change settings in the **.qsf**, refer to the "Appendix: Stratix® V Transceiver Design Guidelines" section in the *Stratix V Device Design Guidelines*. The guidelines in the document are applicable to the Arria V and Cyclone V transceivers.

Power Pin Connections

The Arria V and Cyclone V devices require various voltage supplies depending on your design requirements. Use the following checklist to design the board for the FPGA power pin connections.

Table 5. Power Pin Connections Checklist (Part 1 of 3)

No.	✓	Checklist items

1.	☐	*Design the board for power-up*
		The Arria V and Cyclone V devices support hot socketing (hot plug-in/hot swap) and power sequencing without the use of external devices. Consider the following guidelines:
		■ The output buffers are tri-stated with weak pull-up resistors enabled until the device is configured and configuration pins drive out.
		■ Design the voltage power supply ramps to be monotonic—ensure that the minimum current requirement for the power-on-reset (POR) supplies is available during device power up. If the following voltages share the same power supply and there is not enough current available during power up, these results apply:
		  ■ V_{CCIO}, V_{CCPD}, and V_{CCPGM}—the devices do not enter POR
		  ■ V_{CCIO} and V_{CCPD}—the devices do not exit POR
		■ Set the POR delay to ensure power supplies are stable—use the MSEL pin settings to select between a typical POR delay of 4 ms or 100 ms. In both cases, you can extend the POR delay by using an external component to assert the `nSTATUS` pin low. To ensure the device configures properly and enters user mode, extend the POR delay if the board cannot meet the maximum power ramp time specifications.
		■ For the device to exit POR, you must power the V_{CCBAT} power supply even if you do not use the volatile key.
		■ Design power sequencing and voltage regulators for the best device reliability—although power sequencing is not required for correct operation, consider the power-up timing of each rail to prevent problems with long-term device reliability if you are designing a multi-rail powered system.
		■ Connect the GND between boards before connecting the power supplies—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 device.
		For more information, refer to the following documents:
		■ *Power Management in Arria V Devices*
		■ *Power Management in Cyclone V Devices*
		■ *Arria V Device Family Pin Connection Guidelines*
		■ *Cyclone V Device Family Pin Connection Guidelines*
		■ *Configuration, Design Security, and Remote System Upgrades in Arria V Devices*
		■ *Configuration, Design Security, and Remote System Upgrades in Cyclone V Devices*

Table 5. Power Pin Connections Checklist (Part 2 of 3)

No.	✓	Checklist items
2.	☐	*Review the list of required supply voltages* For a list of the required supply voltages and the recommended operating conditions, refer to the following documents: ■ *Arria V Device Datasheet* ■ *Cyclone V Device Datasheet*
3.	☐	*Ensure I/O power pin compatibility with I/O standards* The output pins of the devices will not conform to the I/O standard specifications if the V_{CCIO} level is out of the recommended operating range of the I/O standard. For a complete list of the supported I/O standards and V_{CCIO} voltages, refer to the following documents: ■ *I/O Features in Arria V Devices* ■ *I/O Features in Cyclone V Devices*

4.	☐	*Ensure correct power pin connections*
		■ Connect all power pins correctly as specified in the following documents:
		  ■ *Arria V Device Family Pin Connection Guidelines*
		  ■ *Cyclone V Device Family Pin Connection Guidelines*
		■ Connect V_{CCIO} pins and V_{REF} pins to support the I/O standards of each bank.
		■ For unused supplies, consider whether there is a need to ground, open, or retain the power connection.

5.	☐	*Determine power rail sharing*
		■ Explore 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. It is especially important for you to consider the power supply sharing ability of devices from different device families.
		■ Follow the suggested power supply sharing and isolation guidance, and the specific guidelines for each pin in the following documents:
		  ■ *Arria V Device Family Pin Connection Guidelines*
		  ■ *Cyclone V Device Family Pin Connection Guidelines*
		  ■ *AN 583: Designing Power Isolation Filters with Ferrite Beads for Altera FPGAs*

Table 5. Power Pin Connections Checklist (Part 3 of 3)

No.	✓	Checklist items
6.	☐	*Use PDN tool to plan for power distribution and decoupling capacitor selection* Arria V and Cyclone V devices include embedded on-package and on-die decoupling capacitors to provide high-frequency decoupling. To plan power distribution and return currents from the voltage regulating module (VRM) to the FPGA power supplies, you can use the power distribution network (PDN) design tool that optimizes the board-level PDN graphically. Although you can use SPICE simulation to simulate the circuit, the PDN design tool provides a fast, accurate, and interactive way to determine the right number of decoupling capacitors for optimal cost and performance trade-off. Download the appropriate PDN tool and documentation from the Altera website: ■ Power Delivery Network (PDN) Tool Version 12.0 for Arria V, Cyclone V, Cyclone IV, and Arria II GZ Devices ■ *AN 574: Printed Circuit Board (PCB) Power Delivery Network (PDN) Design Methodology* ■ *Device-Specific Power Delivery Network (PDN) Tool User Guide* ■ *Power Delivery Network (PDN) Tool User Guide* (device agnostic)

7.	☐	*Review the following guidelines for PLL board design*
		■ Connect all PLL power pins to reduce noise even if the design does not use all the PLLs. For pin voltage requirements, refer to the following documents:
		  ■ *Arria V Device Family Pin Connection Guidelines*
		  ■ *Cyclone V Device Family Pin Connection Guidelines*
		■ Power supply nets should be provided by an isolated power plane, a power plane cut out, or a thick trace.

8.	☐	*Review the transceiver design guidelines* For more information about the Arria V and Cyclone V transceivers, refer to the following documents: ■ *Arria V Device Handbook, Volume 2: Transceivers* ■ *Cyclone V Device Handbook, Volume 2: Transceivers* ■ *Altera Transceiver PHY IP Core User Guide* ■ *Avalon Interface Specifications* For a description of the requirements for simulating a transceiver design using the custom PHY IP core, moving to a design, and how to change settings in the **.qsf**, refer to the “Appendix: Stratix V Transceiver Design Guidelines” section in the *Stratix V Device Design Guidelines*. The guidelines in the document are applicable to the Arria V and Cyclone V transceivers.

Configuration Pin Connections

Depending on your configuration scheme, different pull-up or pull-down resistor, signal integrity, and specific pin requirements apply. Connecting the configurations pins correctly is important. Use the following checklist to address common issues.

Table 6. Configuration Pin Connections Checklist (Part 1 of 3)

No.	✓	Checklist items
1.	☐	*Verify configuration pin connections and pull-up or pull-down resistors are correct for your configuration schemes* For specifics about each configuration pin, refer to the following documents: ■ *Arria V Device Family Pin Connection Guidelines* ■ *Cyclone V Device Family Pin Connection Guidelines*
2.	☐	*Design configuration `DCLK` and `TCK` pins to using the same technique as in designing high-speed signal or system clock* ■ Noise on the `TCK` signal can affect JTAG configuration. ■ Noisy `DCLK` signal can affect configuration and cause a CRC error. ■ For a chain of devices, noise on the `TCK` or `DCLK` pins in the chain can cause JTAG programming or configuration to fail for the entire chain.
3.	☐	*Verify the JTAG pins are connected to a stable voltage level if not in use* JTAG configuration takes precedence over all configuration methods. If you do not use the JTAG interface, do not leave the JTAG pins floating or toggling during configuration.
4.	☐	*Verify the JTAG pin connections to the download cable header* A device operating in JTAG mode uses the required `TDI`, `TDO`, `TMS`, and `TCK` pins, and the optional `TRST` pin. The `TCK` pin has an internal weak pull-down resistor. The `TDI`, `TMS`, and `TRST` pins have weak internal pull-up resistors. The JTAG output pin (`TDO`) and all JTAG input pins are powered by the 2.5 or 3.0-V V_{CCPD}.
5.	☐	*Review the following JTAG pin connections guidelines:* ■ If you have multiple devices in the chain, connect the `TDO` pin of a device to the `TDI` pin of the next device in the chain. ■ Noise on the JTAG pins during configuration, user mode, or power-up can cause the device to go into an undefined state or mode. ■ To disable the JTAG state machine during power-up, pull the `TCK` pin low through a 1-kΩ resistor to ensure that an unexpected rising edge does not occur on `TCK`. ■ Pull `TMS` and `TDI` high through a 1-kΩ to 10-kΩ resistor. ■ Connect `TRST` directly to V_{CCPD}. Connecting the pin low disables the JTAG circuitry.

Table 6. Configuration Pin Connections Checklist (Part 2 of 3)

No.	✓	Checklist items
6.	☐	*Ensure the download cable and JTAG pin voltages are compatible* The download cable interfaces with the JTAG pins of your device. 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 JTAG pins are powered by V_{CCPD}. 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 have to add more level shifters into the chain. For recommendations about connecting a JTAG chain with multiple voltages across the devices in the chain, refer to the following documents: ■ *JTAG Boundary-Scan Testing in Arria V Devices* ■ *JTAG Boundary-Scan Testing in Cyclone V Devices*
7.	☐	*Buffer the JTAG signal according to the following guidelines:* ■ If a cable drives three or more devices, buffer the JTAG signal at the cable connector to prevent signal deterioration. ■ 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.
8.	☐	*Ensure all devices in the chain are connected properly* If your device is in a configuration chain, ensure all devices in the chain are connected properly.
9.	☐	*Ensure that the MSEL pins are not left floating and do not use weak pull-up resistors* ■ Connect the MSEL pins directly to the power or GND planes. ■ If you are using pull-up or pull-down resistors, use 0-Ω resistors. ■ If the MSEL pins are floating or weakly pulled, you may not be able to configure the device.

10.	☐	*Review the following guidelines for other configuration pins:*
		■ Hold the `nCE` (chip enable) pin low during configuration, initialization, and user mode:
		  ■ In single device configuration or JTAG programming—tie `nCE` low
		  ■ In multi-device configuration chain—tie `nCE` low on the first device and connect the `nCEO` pin of the first device to the `nCE` pin of the next device

Table 6. Configuration Pin Connections Checklist (Part 3 of 3)

No.	✓	Checklist items
11.	☐	*Determine if you need to turn on device-wide output enable* The Arria V or Cyclone V device support an optional chip-wide output enable that allows you to override all tri-states on the device I/Os. When the `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 the chip-wide output enable feature: ■ Turn on **Enable device-wide output enable (DEV_OE)** under the **General** category of the **Device and Pin Options** dialog box in the Quartus II software before compiling your design ■ Ensure that the `DEV_OE` pin is driven to a valid logic level on your board ■ Do not leave the `DEV_OE` pin floating

General I/O Pin Connections

Use the following checklist to plan your general I/O pin connections and to improve signal integrity.

Table 7. General I/O Pin Connections Checklist (Part 1 of 4)

No.	✓	Checklist items
1.	☐	*Specify the state of unused I/O pins according to the following guidelines:* ■ To reduce power dissipation, set clock pins and other unused I/O pins **As inputs tri-stated**. By default, the Quartus II software set the input pins tri-stated with weak pull-up resistor enabled. ■ To improve signal integrity, in the **Reserve all unused pins** option under the **Unused Pins** category of the **Device and Pin Options** dialog box of the Quartus II software, set the unused pins **As output driving ground**. This setting reduces inductance by creating a shorter return path and reduces noise on the neighboring I/Os. However, do not use this approach if it results in many via paths that causes congestion for signals under the device. ■ Carefully check the pin connections in the pin report file (**.pin**) generated by the Quartus II software when you compile your design. The **.pin** specifies how you should connect the device pins. I/O pins specified as GND can be left unconnected or connected to ground for improved noise immunity. Do not connect `RESERVED` pins.

Table 7. General I/O Pin Connections Checklist (Part 2 of 4)

No.	✓	Checklist items
2.	☐	*Refer to the Board Design Resource Center* If your design has high-speed signals, especially with Arria V or Cyclone V 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 on the Altera website. 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* You can also refer to the I/O Management, Board Development Support, and Signal Integrity Analysis Resource Center on the Altera website for board-level signal integrity information related to the Quartus II software.
3.	☐	*Design VREF pins to be noise free* Voltage deviation on a `VREF` pin can affect the threshold sensitivity for inputs. For more information about `VREF` pins and I/O standards, refer to "I/O Features and Pin Connections" on page 21.
4.	☐	*Refer to the Board Design Guideline Solution Center* Noise generated by SSN—when too many pins in close proximity change voltage levels at the same time—can reduce the noise margin and cause incorrect switching. For example, consider these board layout recommendations: ■ Break out large bus signals on board layers close to the device to reduce cross talk. ■ If possible, route traces orthogonally if two signal layers are next to each other, and use a separation of two to three times the trace width. For more board layout recommendations that can help with noise reduction, refer to the PCB guidelines in the Board Design Guidelines Solution Center on the Altera website. For a list of recommendations for I/O and clock connections, refer to "I/O Simultaneous Switching Noise" on page 29.

Table 7. General I/O Pin Connections Checklist (Part 3 of 4)

No.	✓	Checklist items
5.	☐	*Verify I/O termination and impedance matching* Voltage-referenced I/O standards require both a VREF and a termination voltage (V_{TT}). The reference voltage of the receiving device tracks the termination voltage of the transmitting device. Consider the following items: ■ Each voltage-referenced I/O standard requires a unique termination setup. For example, a proper resistive signal termination scheme is critical in **SSTL-2** 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. ■ 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. Arria V and Cyclone V devices provide an optional differential on-chip resistor when using LVDS. The Arria V and Cyclone V on-chip series and parallel termination provides the convenience of no external components. You can also use external pull-up resistors to terminate the voltage-referenced I/O standards such as SSTL and HSTL. For a complete list of on-chip termination (OCT) support for each I/O standard, refer to the following documents: ■ *I/O Features in Arria V Devices* ■ *I/O Features in Cyclone V Devices*
6.	☐	*Perform full board routing simulation using IBIS models* 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. To select the IBIS output in the Quartus II software, on the Assignments menu, click **Settings**. Navigate to the **Board-Level** page of the **EDA Tool Settings** category. Under the **Board-level signal integrity analysis** section, in the **Format** option, select **IBIS**. For more information, refer to the *Signal Integrity Analysis with Third-Party Tools* chapter of the *Quartus II Handbook*.
7.	☐	*Configure board trace models for Quartus II advanced timing analysis* The signal integrity and board routing propagation delay is important for you to design proper system operation. If you use and FPGA with high-speed interfaces in your board design, analyze the board level timing as part of the I/O and board planning. To generate a more accurate I/O delays and extra reports to gain better insights into the signal behavior at the system level, turn on **Enable Advanced I/O Timing** under the **TimeQuest Timing Analyzer** category in the **Settings** dialog box of your Quartus II project. With this option turned on, the TimeQuest Timing Analyzer uses simulation results for the I/O buffer, package, and board trace model to generate the I/O delays. You can use the advanced timing reports as a guide to make changes to the I/O assignments and board design to improve timing and signal integrity.

Table 7. General I/O Pin Connections Checklist (Part 4 of 4)

No.	✓	Checklist items
8.	☐	*Review your pin connections* Altera provides schematic review worksheets based on the device Pin Connection Guidelines and other board-level pin connections literature that you need to consider when you finalize your schematics. Use the following worksheets to help you find mistakes in your schematic and adhere to Altera's guidelines: ■ *Arria V Device Schematic Review Worksheet* ■ *Cyclone V Schematic Review Worksheet*

I/O and Clock Planning

Use the checklists in this section as guidelines to plan your I/O and clocking.

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.

Table 8. Early Pin Planning and I/O Assignment Analysis Checklist (Part 1 of 2)

No.	✓	Checklist items
1.	☐	*Verify pin locations in the FPGA place-and-route software early* The FPGA I/O capabilities and board layout guidelines influence pin locations and other types of assignments. Starting FPGA pin planning early improves the confidence in early board layouts, reduces the chance of error, and improves the overall time-to-market.

Table 8. Early Pin Planning and I/O Assignment Analysis Checklist (Part 2 of 2)

No.	✓	Checklist items
2.	☐	*Use the Quartus II Pin Planner for I/O pin planning, assignment, and validation* 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. You can use the Quartus II Pin Planner for I/O pin assignment planning, assignment, and validation: ■ 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. ■ The Create/Import Megafunction feature of the Pin Planner interfaces with the MegaWizard Plug-In Manager, and enables you to create or import custom megafunctions and IP cores that use I/O interfaces. ■ Enter PLL and LVDS blocks, including options such as dynamic phase alignment (DPA) because the options affect the pin placement rules. Then, use the **Create Top-Level Design File** command to generate a top-level design netlist file. ■ 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. ■ You can create a transceiver instance together with its interface and check the transceiver pin or bank placement. After planning is complete, you can pass the preliminary pin location information to PCB designers. After the design is complete, you can use the reports and messages generated by the Quartus II Fitter for the final sign-off of the pin assignments. For more information about I/O assignment and analysis, refer to the *I/O Management* chapter of the Quartus II Handbook.

I/O Features and Pin Connections

Use the checklist in this section for guidelines related to I/O features and pin connections:

- Support for different I/O signal types and I/O standards in device I/O banks, and other I/O features available for your design
- 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 following documents:

- *Arria V Device Family Pin Connection Guidelines*

- *Cyclone V Device Family Pin Connection Guidelines*

Table 9. I/O Features and Pin Connections Checklist (Part 1 of 5)

No.	✓	Checklist items
1.	☐	*Determine if your system requires single-ended I/O signaling* ■ Single-ended I/O signaling provides a simple rail-to-rail interface. ■ The 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.
2.	☐	*Determine if your system requires voltage-referenced signaling* ■ 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 provides an improved logic transition rate with a reduced voltage swing, and minimizes noise caused by reflection with a termination requirement. ■ Additional termination components are required for the reference voltage source (V_{TT}).
3.	☐	*Determine if your system requires differential signaling* ■ 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 avoids the requirement for a clean reference voltage. This is possible because of lower swing voltage and noise immunity with a 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. ■ Allow the software to assign locations for the negative pin in differential pin pairs.
4.	☐	*Select a suitable signaling type and I/O standard for each I/O pin* Ensure that the appropriate I/O standard support is supported in the targeted I/O bank. For more information, refer to the following documents: ■ *I/O Features in Arria V Devices* ■ *High-Speed Differential I/O Interfaces and DPA in Arria V Devices* ■ *I/O Features in Cyclone V Devices*
5.	☐	*Place I/O pins that share voltage levels in the same I/O bank* ■ Certain I/O banks on 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. ■ Use the correct dedicated pin inputs for signals such as clocks and global control signals.

Table 9. I/O Features and Pin Connections Checklist (Part 2 of 5)

No.	✓	Checklist items
6.	☐	*Verify that all output signals in each I/O bank are intended to drive out at the bank's VCCIO voltage level* ■ The board must supply each bank with one V_{CCIO} voltage level for every `VCCIO` pin in a bank. ■ Each I/O bank is powered by the `VCCIO` 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.
7.	☐	*Verify that all voltage-referenced signals in each I/O bank are intended to use the bank's VREF voltage* ■ To accommodate voltage-referenced I/O standards, each I/O bank supports multiple VREF pins feeding a common VREF bus. Set the VREF pins to the correct voltage for the I/O standards in the bank. ■ Each I/O bank can only have a single V_{CCIO} voltage level and a single VREF voltage level at a given time. If the VREF pins are not used as voltage references, the pins cannot be used as generic I/O pins and must be tied to the 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 VREF setting. ■ For performance reasons, voltage-referenced input standards use their own V_{CCPD} level as the power source. You can place voltage-referenced input signals in a bank with a VCCIO of 2.5 V or below. ■ Voltage-referenced bidirectional and output signals must drive out at the V_{CCIO} voltage level of the I/O bank.
8.	☐	*Check the I/O bank support for LVDS and transceiver features* Different I/O banks include different support for LVDS signaling. The Arria V and Cyclone V transceiver banks include additional support. For more information, refer to the following documents: ■ *High-Speed Differential I/O Interfaces and DPA in Arria V Devices* ■ *Transceiver Architecture in Arria V Devices* ■ *I/O Features in Cyclone V Devices* ■ *Transceiver Architecture in Cyclone V Devices*
9.	☐	*Verify the usage of the VREF pins that are used as regular I/Os* VREF pins have higher pin capacitance that results in a different I/O timing: ■ Do not use these pins in a grouped interface such as a bus. ■ Do not use these pins for high edge rate signals such as clocks.

Table 9. I/O Features and Pin Connections Checklist (Part 3 of 5)

No.	✓	Checklist items
10.	☐	*Use the UniPHY megafunction (or IP core) for each memory interface, and follow connection guidelines* The self-calibrating UniPHY megafunction is optimized to take advantage of the Arria V or Cyclone V I/O structure. The UniPHY megafunction allows you to set external memory interface features and helps set up the physical interface (PHY) best suited for your system. When you use the Altera memory controller MegaCore functions, the UniPHY megafunction is instantiated automatically. 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. For more information, refer to the *Planning Pin and FPGA Resources* chapter in the *External Memory Interface Handbook*.
11.	☐	*Use dedicated DQ/DQS pins and DQ groups for memory interfaces* The data strobe DQS and data DQ pin locations are fixed in Arria V and Cyclone V devices. 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 specific external memory interface topic, refer to the following documents: ■ *Volume 2: Design Guidelines* in the *External Memory Interface Handbook* ■ External Memory Interface Spec Estimator on the Altera website ■ *Introduction to UniPHY IP* in the *External Memory Interface Handbook* ■ External Memory Solutions Center on the Altera website
12.	☐	*Make dual-purpose pin settings and check for any restrictions when using these pins as regular I/O* You can use dual-purpose configuration pins as general I/Os after device configuration is complete. Select the desired setting for each of the dual-purpose pins on the **Dual-Purpose Pins** category 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 the GCLK networks, as general purpose input pins if they are not used as clock pins. If you use the clock inputs as general inputs, the 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 they are not enabled. For more information, refer to “Determine if you need to turn on device-wide output enable” on page 17 and “Enable the chip-wide reset to clear all registers if required” on page 31.

Table 9. I/O Features and Pin Connections Checklist (Part 4 of 5)

No.	✓	Checklist items
13.	☐	*Review available device I/O features that can help I/O interfaces* Check the available I/O features and consider the following guidelines: ■ Programmable current strength—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 an IBIS or SPICE simulations to determine the right current strength setting for your specific application. ■ Programmable slew rate—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 input/output element (IOE) delays—helps read and time margins by minimizing the uncertainties between signals in the bus. For delay specifications, refer to the relevant device datasheet. ■ Open-drain output—if configured as an open-drain, the logic value of the output is either high-Z or 0. This feature is 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—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. For the specific sustaining current driven through this resistor and the overdrive current used to identify the next driven input and level for each V_{CCIO} voltage, refer to the relevant device datasheet. ■ Programmable pull-up resistors—weakly holds the I/O to the V_{CCIO} level when in user mode. This feature can be used with the open-drain output to eliminate the need for an external pull-up resistor. If the programmable pull-up option is enabled, you cannot use the bus-hold 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. ■ Programmable differential output voltage—allows you to adjust output eye height to optimize trace length and power consumption. A higher V_{OD} swing improves voltage margins at the receiver end while a smaller V_{OD} swing reduces power consumption. For more information, refer to the following document: ■ *I/O Features in Arria V Devices* ■ *I/O Features in Cyclone V Devices*

Table 9. I/O Features and Pin Connections Checklist (Part 5 of 5)

No.	✓	Checklist items
14.	☐	*Consider OCT features to save board space* 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. ■ OCT R_S and R_T 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 R_S, programmable current strength, or OCT R_T ■ You cannot configure both OCT R_S and programmable current strength or slew rate control for the same I/O buffer ■ Differential OCT R_D is available in all I/O pins For details about the support and implementation of this feature, refer to the following documents: ■ *I/O Features in Arria V Devices* ■ *High-Speed Differential I/O Interfaces and DPA in Arria V Devices* ■ *I/O Features in Cyclone V Devices*
15.	☐	*Verify that the required termination scheme is supported for all pin locations*
16.	☐	*Choose the appropriate mode of DPA, non-DPA, or soft-CDR for high-speed LVDS interfaces* For more information, refer to the following documents: ■ *High-Speed Differential I/O Interfaces and DPA in Arria V Devices* ■ *I/O Features in Cyclone V Devices*

Clock Planning

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.
- Based on your system requirements, define the required clock frequencies for your FPGA design and the input frequencies available to the FPGA. Use these specifications to determine your PLL scheme.

- Use the Quartus II MegaWizard Plug-In Manager to enter your settings in Altera PLL megafunctions, and check the results to verify whether particular features and input/output frequencies can be implemented in a particular PLL.

Table 10. Clock Planning Checklist (Part 1 of 2)

No.	✓	Checklist items
1.	☐	*Use the device fractional PLLs for clock management* Connect clock inputs to specific PLLs to drive specific low-skew routing networks. Analyze the global resource availability for each PLL and the PLL availability for each clock input pin. Use the following descriptions to help determine which clock networks are appropriate for the clock signals in your design: ■ The GCLK networks can drive throughout the entire device, serving as low-skew clock sources for device logic. This clock region has the maximum delay 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 RCLK networks only pertain to the quadrant they drive into and provide the lowest clock delay and skew for logic contained within a single device quadrant. ■ IOEs and internal logic can also drive GCLKs and RCLKs to create internally generated GCLKs or RCLKs 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 from another pin/PLL-fed GCLK or RCLK. ■ PCLK networks are a collection of individual clock networks driven from the periphery of the device. Clock outputs from the DPA block, PLD-transceiver interface, 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 device.
2.	☐	*Enable PLL features and check settings in the MegaWizard Plug-In Manager* You can configure the fractional PLLs in the Arria V and Cyclone V to function in integer or enhanced fractional mode. Take note that to drive LVDS channels, you must use the PLLs in integer mode. For more information about the fractional PLL features, refer to the “Fractional PLL Architecture” section in the following documents: ■ *Clock Networks and PLLs in Arria V Devices* ■ *Clock Networks and PLLs in Cyclone V Devices*
3.	☐	*Ensure that you select the correct PLL feedback compensation mode* Arria V and Cyclone V PLLs support six different clock feedback modes. For more information, refer to the “Clock Feedback Modes” section in the following documents: ■ *Clock Networks and PLLs in Arria V Devices* ■ *Clock Networks and PLLs in Cyclone V Devices*
4.	☐	*Check that the PLL offers the required number of clock outputs and use dedicated clock output pins* You can connect clock outputs to dedicated clock output pins or clock networks.

Table 10. Clock Planning Checklist (Part 2 of 2)

No.	✓	Checklist items
5.	☐	*Use the clock control block for clock selection and power-down* Every GCLK and RCLK network has its own clock control block. The control block provides the following features that you can use to select different clock input signals or power-down clock networks to reduce power consumption without using any combinational logic in your design: ■ Clock source selection (with dynamic selection for GCLKs) ■ GCLK multiplexing ■ Clock power down (with static or dynamic clock enable or disable) In Arria V and Cyclone V devices, the `clkena` signals are supported at the clock network level instead of at the PLL output counter level. This allows you to gate off the clock even when you are not using a PLL. You can also use the `clkena` signals to control the dedicated external clocks from the PLLs. For more information, refer to the *Clock Control Block (ALTCLKCTRL) Megafunction User Guide*.

I/O Simultaneous Switching Noise

SSN is a concern when too many I/Os (in close proximity) change voltage levels at the same time. Use the checklist in this section for recommendations to plan I/O and clock connections.

Table 11. I/O Simultaneous Switching Noise Checklist

No.	✓	Checklist item
1.	☐	*Consider the following recommendations to mitigate I/O simultaneous switching noise:* ■ Analyze the design for possible SSN problems. ■ Reduce the number of pins that switch the voltage level at exactly the same time whenever possible. ■ Use differential I/O standards and lower-voltage standards for high-switching I/Os. ■ Use lower drive strengths for high-switching I/Os. The default drive strength setting might be higher than your design requires. ■ Reduce the number of simultaneously switching output pins within each bank. Spread output pins across multiple banks if possible. ■ Spread the switching I/Os evenly throughout the bank to reduce the number of aggressors in a given area to reduce SSN if bank usage is substantially below 100%. ■ Separate simultaneously switching pins from input pins that are susceptible to SSN. ■ Place important clock and asynchronous control signals near ground signals and away from large switching buses. ■ Avoid using I/O pins one or two pins away from PLL power supply pins for high-switching or high-drive strength pins. ■ Use staggered output delays to shift the output signals through time, or use adjustable slew rate settings. For information and guidelines on using available I/O features, refer to "I/O Features and Pin Connections" on page 21. For signal integrity design techniques to mitigate SSN, view the Signal & Power Integrity Design Techniques for SSN webcast on the Altera website.

Design Entry

In complex FPGA design development, design practices, coding styles, and megafunction usage 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.

Table 12. Design Entry Checklist (Part 1 of 6)

No.	✓	Checklist items
1.	☐	*Use synchronous design practices* 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.

Table 12. Design Entry Checklist (Part 2 of 6)

No.	✓	Checklist items

2.	☐	*Consider the following recommendations to avoid clock signals problems:*
		■ Use dedicated clock pins and clock routing for best results—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.
		■ 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.
		■ 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.
		■ Consider the following recommendations for transceivers:
		  ■ Use the transceiver dedicated `refclk` pins.
		  ■ For easier timing closure, clock the transmit logic in the fabric using the transmitter recovered clock, and the receive logic using the receiver recovered clock.

3.	☐	*Use the Quartus II Design Assistant to check design reliability* The Design Assistant is a design-rule checking tool that checks your design according to Altera recommendations and helps you avoid design issues early in your design flow: ■ To run the tool, 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, refer to the "Checking Design Violations with the Design Assistant" section in the *Recommended Design Practices* chapter of the *Quartus II Handbook*. You can also refer to industry papers for more information about multiple clock design. For a good analysis, refer to www.sunburst-design.com/papers.
4.	☐	*Use megafunctions with the MegaWizard Plug-In Manager* Instead of coding your own logic, save your design time by using Altera's megafunctions—a library of parameterized modules and device-specific megafunctions. The megafunctions are optimized for Altera device architectures and can offer more efficient logic synthesis and device implementation. To ensure that you set all ports and parameters correctly, use the Quartus II Megawizard Plug-In Manager to build or change megafunctions parameters. For detailed information about a specific megafunction, refer to the relevant user guide on the IP and Megafunctions page at the Altera website.

Table 12. Design Entry Checklist (Part 3 of 6)

No.	✓	Checklist items
5.	☐	*Review the information on dynamic and partial reconfiguration features* The Arria V and Cyclone V devices support dynamic and partial reconfiguration: ■ Dynamic reconfiguration—dynamically change the transceiver data rates, PMA settings, or protocols of a channel affecting data transfer on adjacent channels. ■ Partial reconfiguration—reconfigure part of the device while other sections of the device remain operational. For more information, refer to *Increasing Design Functionality with Partial and Dynamic Reconfiguration in 28-nm FPGAs*.
6.	☐	*Consider the Altera's recommended coding styles to achieve optimal synthesis results* HDL coding styles can have a significant impact on the quality of results for programmable logic designs. For example, when designing memory and digital system processing (DSP) functions, understanding the device architecture helps you to take advantage of the dedicated logic block sizes and configurations. ■ For specific HDL coding examples and recommendations, refer to the *Recommended HDL Coding Styles* chapter in the *Quartus II Handbook*. ■ You can use the HDL templates provided in the Quartus II software as examples for your reference. To access the templates, right click the editing area in the Quartus II text editor and click **Insert Template**. ■ For additional tool-specific guidelines, refer to the documentation of your synthesis tool.
7.	☐	*Enable the chip-wide reset to clear all registers if required* Arria V and Cyclone V 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). ■ `DEV_CLRn` pin is driven low—all registers are cleared or reset to 0. The affected register behave as if they are preset to a high value when synthesis performs an optimization called NOT-gate-push back due to register control signals. ■ `DEV_CLRn` pin is driven high—all registers behave as programmed. To enable chip-wide reset, before compiling your design, turn on **Enable device-wide reset (DEV_CLRn)** under the **Options** list of the **General** category in the **Device and Pin Options** dialog box of the Quartus II software.
8.	☐	*Use device architecture-specific register control signals* Each Arria V and Cyclone V logic array block (LAB) contains dedicated logic for driving register control signals to its ALMs. It is important that the control signals use the dedicated control signals in the device architecture. In some cases, you may be required to limit the number of different control signals in your design. For more information about LAB and ALM architecture, refer to the following documents: ■ *Logic Array Blocks and Adaptive Logic Modules in Arria V Devices* ■ *Logic Array Blocks and Adaptive Logic Modules in Cyclone V Devices*

Table 12. Design Entry Checklist (Part 4 of 6)

No.	✓	Checklist items
9.	☐	*Review recommended reset architecture* ■ 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 allows the reset signal to be asserted asynchronously and deasserted synchronously. ■ The source of the reset signal is 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. ■ Synchronous deassertion avoids an asynchronous reset signal from being released at, or near, the active clock edge of a flipflop that can cause the output of the flipflop to go to a metastable unknown state. For more information about good reset design, refer to industry papers such as the analysis of reset architecture at www.sunburst-design.com/papers.
10.	☐	*Review the synthesis options available in your synthesis tool* If you force a particular power-up condition for your design, use the synthesis options available in your synthesis tool: ■ By default, the Quartus II software Integrated Synthesis turns on the **Power-Up Don't Care** logic option that assumes your design does not depend on the power-up state of the device architecture. 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 with asynchronous reset that allows you to power up the design safely with the reset active, regardless of the power-up conditions of the device. ■ Some synthesis tools can also read the default or initial values for registered signals in your source code and implement the behavior in the device. For example, the Quartus II software Integrated Synthesis converts HDL default and initial values for registered signals into **Power-Up Level** settings. The synthesized behavior matches the power-up conditions of the HDL code during a functional simulation. ■ 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 (preset signal), synthesis tools typically use the clear signals available on the registers and perform the NOT-gate push back optimization technique. 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. For more information about the **Power-Up Level** settings and the `altera_attribute` assignment that sets the power-up state, refer to the *Quartus II Integrated Synthesis* chapter of the *Quartus II Handbook*.

Table 12. Design Entry Checklist (Part 5 of 6)

No.	✓	Checklist items
11.	☐	*Consider resources available for register power-up and control signals* 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 the *Quartus II Handbook*.
12.	☐	*Consider Altera's recommendations for creating design partitions* Partitioning a design for an FPGA requires planning to ensure optimal results when the partitions are integrated and ensures 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. 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. For guidelines to help you create design partitions, refer to the *Best Practices for Incremental Compilation Partitions and Floorplan Assignments* chapter in the *Quartus II Handbook*.
13.	☐	*Perform timing budgeting and resource balancing between partitions* If your design is created in multiple 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, which can lead to problems during system integration. ■ The top-level project information, including pin locations, physical constraints, and timing requirements, should be communicated to the designers of lower-level partitions before they start their design. ■ The system architect can plan design partitions at the top level and use the Quartus II software **Generate Bottom-Up Design Partition Scripts** option on the Project menu to automate the process of transferring top-level project information to lower-level modules.

Table 12. Design Entry Checklist (Part 6 of 6)

No.	✓	Checklist items
14.	☐	*Create a design floorplan for incremental compilation partitions* ■ A design floorplan avoids conflicts between design partitions and ensure that each partition is well-placed relative to other partitions. When you create different location assignments for each partition, no location conflicts occur. ■ A design floorplan helps avoid situations in which the Fitter is directed to place or replace a portion of the design in an area of the device where most resources have already been claimed. ■ Floorplan assignments are recommended for timing-critical partitions in top-down flows. You can use the Quartus II Chip Planner to create a design floorplan using LogicLock region assignments for each design partition. ■ With a basic design framework for the top-level design, the floorplan editor enables you to view connections between regions, estimate physical timing delays on the chip, and move regions around the device floorplan. ■ After you compiled the full design, you can also view logic placement and locate areas of routing congestion to improve the floorplan assignments. For more information and guidelines in creating a design floorplan and placement assignments in the floorplan, refer to the following chapters in the *Quartus II Handbook*: ■ *Best Practices for Incremental Compilation Partitions and Floorplan Assignments* ■ *Analyzing and Optimizing the Design Floorplan with the Chip Planner*

Design Implementation

Use the checklists in the this section as guidelines while implementing your design.

Synthesis and Compilation

Table 13. Synthesis and Compilation Checklist (Part 1 of 3)

No.	✓	Checklist items
1.	☐	*Specify your synthesis tool and use correct supported version* The Quartus II software includes integrated synthesis that fully supports Verilog HDL, VHDL, 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 (**.lmf**) 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. ■ 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. ■ Perform placement and routing in the Quartus II software to get accurate timing analysis and logic utilization results. ■ Your synthesis tool may offer the capability to create a Quartus II project and pass constraints such as the EDA tool setting, device selection, and timing requirements that you specified in your synthesis project. 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 following chapters of the Quartus II Handbook: ■ *Quartus II Integrated Synthesis* ■ *Synopsys Synplify Support* ■ *Mentor Graphics Precision Synthesis Support* ■ *Mentor Graphics LeonardoSpectrum Support* For information about the officially supported version of each synthesis tool in a Quartus II software version, refer to the relevant Quartus II software release notes on the Release Notes page at the Altera website.

Table 13. Synthesis and Compilation Checklist (Part 2 of 3)

No.	✓	Checklist items
2.	☐	*Review resource utilization reports after compilation* After compilation in the Quartus II software, review the device resource utilization information: ■ Use the information to determine whether the future addition of extra logic or other design changes introduce fitting difficulties. ■ If your compilation results in a no-fit error, use the information to analyze fitting problems. ■ 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 usage. ■ For more detailed resource information, view 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 use the Quartus II Integrated Synthesis, the reports under the **Optimization Results** folder in the **Analysis & Synthesis** section provide information that includes 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. Low logic utilization does not mean the lowest possible ALM utilization. A design that is reported to be close to 100% may 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.
3.	☐	*Review all Quartus II messages, especially warning or error messages* Each stage of the compilation flow generates messages, including informational notes, warnings, and critical warnings. Understand the significance of 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. You can suppress the messages if you have determined that your action is not required. For more information about messages and message suppression, refer to the *Managing Quartus II Projects* chapter in the *Quartus II Handbook*.
4.	☐	*Consider using incremental compilation* 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.

Table 13. Synthesis and Compilation Checklist (Part 3 of 3)

No.	✓	Checklist items
5.	☐	*Ensure parallel compilation is enabled* 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 change the default setting in the **Options** dialog box in the **Processing** page from the Tools menu.
6.	☐	*Use the Compilation Time Advisor* 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 quality of results. For more information, refer to the *Area and Timing Optimization* chapter in the Quartus II Handbook.

Timing Optimization and Analysis

Use the guidelines in the following checklist for analyzing your design timing and optimizing your timing performance.

Table 14. Timing Optimization and Analysis Checklist (Part 1 of 2)

No.	✓	Checklist items
1.	☐	*Ensure timing constraints are complete and accurate* 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. For more information, refer to the *Timing Analysis Overview* chapter of the *Quartus II Handbook*.
2.	☐	*Review the TimeQuest Timing Analyzer reports after compilation* 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. The software also supports static timing analysis in the industry-standard Synopsys Primetime software. To generate the required timing netlist, specify the tool in the New Project Wizard or the **EDA Tools Settings** page of the **Settings** dialog box.

Table 14. Timing Optimization and Analysis Checklist (Part 2 of 2)

No.	✓	Checklist items
3.	☐	*Ensure that the I/O timings are not violated when data is provided to the FPGA* 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.
4.	☐	*Perform Early Timing Estimation before running a full compilation* If the timing analysis reports that your design requirements were not met, you must make changes to your design or settings and recompile the design to achieve timing closure. If your compilation results in no-fit messages, you must make changes to get successful placement and routing. You can use the Early Timing Estimation feature in the Quartus II software 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.
5.	☐	*Consider the following recommendations for timing optimization and analysis assignment:* ■ Turn on **Optimize multi-corner timing** on the Fitter Settings page in the Settings dialog box. ■ Use `create_clock` and `create_generated_clock` to specify the frequencies and relationships for all clocks in your design. ■ Use `set_input_delay` and `set_output_delay` to specify the external device or board timing parameters ■ 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. ■ Use `derive_clock_uncertainty` to automatically apply inter-clock, intra-clock, and I/O interface uncertainties. ■ Use `check_timing` to generate a report on any problem with the design or applied constraints, including missing constraints ■ Use the Quartus II optimization features to achieve timing closure or improve the resource utilization. ■ Use the Timing and Area Optimization Advisors to suggest optimization settings. For more guidelines about timing constraints, refer to *The Quartus II TimeQuest Timing Analyzer* chapter in the *Quartus II Handbook*.

Functional and Timing Simulation

Use the following checklist for guidelines about functional and timing simulation.

Table 15. Functional and Timing Simulation Checklist

No.	✓	Checklist items
1.	☐	*Perform functional simulation at the beginning of your design flow* Perform the simulation 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.
2.	☐	*Perform timing simulation to ensure your design works in targeted device* Timing simulation uses the timing netlist generated by the TimeQuest Timing Analyzer, which includes the delay of different device blocks and 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.
3.	☐	*Specify your simulation tool and use correct supported version* ■ Altera provides the ModelSim®-Altera simulator Starter Edition and offers the higher-performance ModelSim-Altera Edition that 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. ■ If you use a third-party simulation tool, use the software version that is supported with your Quartus II software version. ■ 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. ■ Use only the model libraries provided with your Quartus II software version. Libraries may change between versions and this can cause a mismatch with your simulation netlist. ■ To create a testbench in the Quartus II software, on the **Processing** menu, point to **Start** and click **Start Testbench Template Writer**. For information about the officially supported version of each simulation tool in a Quartus II software version, refer to the relevant Quartus II software release notes on the Release Notes page at the Altera website. For more information, refer to the following documents in the *Quartus II Handbook*: ■ *Simulating Altera Designs* ■ *Mentor Graphics ModelSim and QuestaSim Support* ■ *Synopsys VCS and VCS MX Support* ■ *Cadence Incisive Enterprise Simulator Support* ■ *Aldec Active-HDL and Rivera-PRO Support*

Formal Verification

Use the following guidelines if your design requires formal verification.

Table 16. Formal Verification Checklist

No.	✓	Checklist items
1.	☐	*Determine if you require formal verification for your design* If formal verification is required for your design, it is easier to plan for limitations and restrictions in the beginning than to make changes later in the design flow.
2.	☐	*Check for support and design limitations for 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. For more information, refer to the *Cadence Encounter Conformal Support* chapter in the *Quartus II Handbook*.
3.	☐	*Specify your formal verification tool and use correct supported version* Specify your formal verification tool in the **EDA Tools Settings** page of the **Settings** dialog box to generate the appropriate output netlist. For information about the officially supported version of each formal verification tool in a Quartus II software version, refer to the relevant Quartus II software release notes on the Release Notes page at the Altera website.

Power Analysis and Optimization

After compiling your design, analyze the power consumption and heat dissipation with the Quartus II PowerPlay Power Analyzer to calculate the dynamic, static, and I/O thermal power consumption and ensure the design has not violated power supply and thermal budgets.

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

Table 17. Power Analysis and Optimization Checklist (Part 1 of 3)

No.	✓	Checklist items
1.	☐	*Provide accurate typical signal activities to get accurate power analysis result* You need to provide accurate typical signal activities to PowerPlay Power Analyzer: ■ Compile a design to derive the information about design resources, placement and routing, and I/O standards. ■ Derive signal activity data (toggle rates and static probabilities) from simulation results or a user-defined default toggle rate and vectorless estimation. The signal activities used for 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 and not the corner cases often used during functional verification. Use the recommended simulator settings, such as glitch filtering, to ensure good results.
2.	☐	*Specify the correct operating conditions for power analysis* Specify the operating conditions, including the core voltage, device power characteristics, ambient and junction temperature, cooling solution, and the board thermal model. In the Quartus II software, select the appropriate settings on the **Operating Settings and Conditions** page in the **Settings** dialog box.
3.	☐	*Analyze power consumption and heat dissipation in the PowerPlay Power Analyzer* In the Quartus II software, 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 signal activities. For more information about power analysis and recommendations for simulation settings for creating signal activity information, refer to the *PowerPlay Power Analysis* chapter in the *Quartus II Handbook*. The PowerPlay Power Analyzer report is a power estimate and is not a power specification. Always refer to the device datasheet for the power specification.
4.	☐	*Review recommended design techniques and Quartus II options to optimize power consumption* For information about design techniques to optimize power consumption, refer to the *Power Optimization* chapter of the Quartus II Handbook.

Table 17. Power Analysis and Optimization Checklist (Part 2 of 3)

No.	✓	Checklist items
5.	☐	*Use the Power Optimization Advisor to suggest optimization settings* The Power Optimization Advisor provides specific power optimization advice and recommendations based on the current design project settings and assignments. For more information, refer to the *Power Optimization* chapter in the Quartus II Handbook.
6.	☐	*Consider using a faster speed grade device* If your design includes many 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.
7.	☐	*Optimize the 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) Megafunction User Guide*. To reduce LAB-wide clock power consumption without disabling the entire clock tree, use the LAB-wide clock enable signal to gate the LAB wide clock. The Quartus II software automatically promotes register-level clock enable signals to the LAB level.
8.	☐	*Reduce the number of memory clocking events* Reduce the number of memory clocking events to reduce memory power consumption. You can use clock gating or the clock enable signals in the memory ports.

Table 17. Power Analysis and Optimization Checklist (Part 3 of 3)

No.	✓	Checklist items
9.	☐	*Consider the following I/O power guidelines:* ■ The dynamic power consumed in the I/O buffer is proportional to the total load capacitance—lower capacitance reduces power consumption. ■ Dynamic power is proportional to the square of the voltage. Use lower voltage I/O standards to reduce dynamic power. Non-terminated I/O standards such as **LVTTL** and **LVCMOS** have a rail-to-rail output swing equal to the V_{CCIO} supply voltage and consume little static power. ■ Dynamic power is 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 the V_{CCIO} around a bias point. Because of this, the 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 Power Analyzer calculations. Ensure that you include the external devices power separately in your system power calculations.
10.	☐	*Review the information on power-driven compilation and Power Optimization Advisor* For more information, refer to the *Power Optimization* chapter in the Quartus II Handbook.

Document Revision History

Table 18 lists the revision history for this document.

Table 18. Document Revision History

Date	Version	Changes
January 2013	1.0	Initial release.