

This application note provides an easy-to-use set of guidelines and a list of factors to consider in Arria® II designs. It is important to follow Altera recommendations throughout the design process. Altera® Arria II FPGAs are designed for ease-of-use, low cost, and low power, while supporting common interfaces with streamlined transceivers and I/Os. Planning the FPGA and system early in the design process is crucial to your success.


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

These guidelines improves productivity and avoid common design pitfalls. You can use the “[Design Checklist](#)” on page 54 to help verify that you have followed each of the guidelines.

[Table 1](#) lists the various stages of the design flow in the order that each stage is typically performed.

Table 1. Summary of Design Flow Stages and Guideline Topics

Stages of Design Flow	Guideline Topics
“ System Specification ” on page 2	Planning design specifications, IP selection
“ Device Selection ” on page 4	Device information, determining device variant and density, package offerings, migration, speed grade
“ Early System and Board Planning ” on page 7	Early power estimation, planning configuration scheme, planning for on-chip debugging
“ Pin Connection Considerations for Board Design ” on page 15	Power up, power pins, PLL connections, decoupling capacitors, configuration pins, signal integrity, board-level verification
“ I/O and Clock Planning ” on page 25	Pin assignments, early pin planning, I/O features and connections, memory interfaces, clock and PLL selection, SSN
“ Design Entry ” on page 38	Coding styles and design recommendations, SOPC Builder, planning for hierarchical or team-based design
“ Design Implementation, Analysis, Optimization, and Verification ” on page 44	Synthesis tool, device utilization, messages, timing constraints and analysis, area and timing optimization, compilation time, verification, power analysis and optimization

 For complete information about the Arria II device architecture, refer to the [Arria II Literature](#) page. For the latest known issues related to Arria II FPGAs, refer to the [Knowledge Database](#) page of the Altera website.

System Specification

In systems that contain an Arria II device, the FPGA typically plays a large role in the overall system and affects the rest of the system design. It is important to start the design process by creating detailed design specifications for the system and the FPGA, and determining the FPGA input and output interfaces to the rest of the system.

Creating Design Specifications

Before you create your logic design or complete your system design, detailed design specifications must define the system, specify the I/O interfaces for the FPGA, identify the different clock domains, and include a block diagram of basic design functions. For suggestions about including intellectual property blocks, refer to [“IP Selection”](#). Taking the time to create these specifications improves design efficiency, but the stage is often skipped by FPGA designers.

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

Create a functional verification plan to ensure the team knows how to verify the system. Creating a test plan at this stage can also help you design for testability and manufacturability. For example, if you want to perform any built-in-self-test functions to drive interfaces, use an UART interface with a Nios[®] II processor inside the FPGA device. You might require the ability to validate all the design interfaces. For guidelines related to analyzing and debugging the device after it is in the system, refer to [“Planning for On-Chip Debugging”](#) on page 13.

If your design includes multiple designers, it is useful to consider a common design directory structure. This eases the design integration stages. [“Planning for Hierarchical and Team-Based Design”](#) on page 42 provides more suggestions for team-based designs.


IP Selection

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

IP selection often affects system design, especially if the FPGA interfaces with other devices in the system. Consider which I/O interfaces or other blocks in your system design can be implemented using IP cores, and plan to incorporate these cores in your FPGA design.

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

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

 For descriptions of available IP cores, refer to the [Intellectual Property](#) page of the Altera website.

SOPC Builder

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

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

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


 For more information about the Avalon interface, refer to the [Avalon Interface Specifications](#) manual.

 For information about using SOPC builder to improve your productivity, refer to the [SOPC Builder Literature](#) page of the Altera website.

Device Selection

This section describes the first step in the Arria II design process—choosing the number of transceivers, device density, features, package, and speed grade that best suit your design requirements. You must also consider whether you want to target FPGA migration devices, which is also described in this section.

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

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

High-Speed Transceivers

Arria II FPGAs include between 4 and 24 full-duplex transceiver channels at data rates between 600 Mbps and 6.375 Gbps with PCS and PMA support, and a PCI Express (PCIe) hard IP block. Choose a device density and package that supports enough transceivers for your application; larger densities and package pin counts offer more transceivers.

Logic, Memory, and Multiplier Density

Arria II devices offer a range of densities that provide different amounts of device logic resources, including memory, multipliers, and adaptive logic module (ALM) logic cells. Determining the required logic density can be a challenging part of the design planning process. Devices with more logic resources can implement larger and potentially more complex designs, but generally have a higher cost. Smaller devices have lower static power utilization. Arria II devices support vertical migration, which provides flexibility, as described in [“Vertical Device Migration”](#) on page 6.

Many next-generation designs use a current design as a starting point. If you have other designs that target an Altera device, you can use their resource utilization as an estimate for your new design. Compile existing designs in the Quartus II software with the **Auto device selected by the Fitter** option in the **Settings** dialog box. Review the resource utilization to find out which device density fits the design. Consider that coding style, device architecture, and the optimization options used in the Quartus II software can significantly affect a design’s resource utilization and timing performance. For more information about determining resource utilization for a compiled design, refer to [“Device Resource Utilization Reports”](#) on page 45.

To obtain resource utilization estimates for certain configurations of Altera’s IP designs, refer to the user guides for Altera megafunctions and IP MegaCores on the [IP Megafunctions Literature](#) page on the Altera website.

7. Reserve device resources for future development and debugging.

Select a device that meets your design requirements with some safety margin, in case you want to add more logic later in the design cycle, upgrade, or expand your design. You might also want additional space in the device to ease creating a design floorplan for incremental or team-based design, as described in [“Planning for Hierarchical and Team-Based Design” on page 42](#). Consider reserving resources for debugging, as described in [“Planning for On-Chip Debugging” on page 13](#).

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

Arria II devices are available in space-saving FineLine BGA (FBGA), Ultra Fineline BGA (UBGA), and Hybrid FineLine BGA (HBGA) packages with various I/O pin counts between 150 and 726 user I/O pins. Determine the required number of I/O pins for your application, considering the design’s interface requirements with other system blocks.

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

Other factors can also affect the number of I/O pins required for a design, including simultaneous switching noise (SSN) concerns, pin placement guidelines, pins used as dedicated inputs, I/O standard availability for each I/O bank, differences between I/O standards and speed for row and column I/O banks, and package migration options. For more information about choosing pin locations, refer to [“Pin Connection Considerations for Board Design” on page 15](#) and [“I/O and Clock Planning” on page 25](#).


Consider reserving pins for debugging, as described in [“Planning for On-Chip Debugging” on page 13](#).

PLLs and Clock Routing

Arria II devices include 4 or 8 PLLs. There are up to 16 global GCLKs, 88 regional RCLKs, and 88 additional periphery PCLKs. Check that your chosen device density package combination includes enough PLLs and clock routing resources for your design. Global clock resources are shared between certain PLLs, which can affect which inputs are available for use. For more information and references regarding clock pins and global routing resources, refer to [“I/O and Clock Planning” on page 25](#).

Speed Grade

The device speed grade affects the device timing performance and timing closure, as well as power utilization. Arria II devices are available in four speed grades: 3, 4, 5, and 6 (3 is the fastest). Generally, the faster devices cost more. One way to determine which speed grade your design requires is to consider the supported clock rates for specific I/O interfaces.


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

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

Vertical Device Migration

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

8. Consider vertical device migration availability and requirements.

 For a list of migration devices, refer to the “Package Options and I/O Information for Arria II GX Devices” and “Package Options and I/O Information for Arria II GZ Devices” tables in the *Overview for the Arria II Device Family* chapter in volume 1 of the *Arria II Device Handbook*.

Determine whether you want the option of migrating your design to another device density. Choose your device density and package to accommodate any possible future device migration to allow flexibility when the design nears completion. You must specify any potential migration options in the Quartus II software at the beginning of your design cycle. Selecting a migration device can impact the design’s pin placement, because the Fitter ensures your design is compatible with the selected device or devices. It is possible to add migration devices later in the design cycle, but it requires extra effort to check pin assignments, and can require design or board layout changes to fit into the new target device. It is easier to consider these issues early in the design cycle than at the end, when the design is near completion and ready for migration.

As described in “[Making FPGA Pin Assignments](#)” on page 25, the Quartus II Pin Planner highlights pins that change function in the migration device when compared with the currently selected device.

Early System and Board Planning

You must plan the system information related to the FPGA early in the design process, before designers have completed the design in the Quartus II software. Early planning allows the FPGA team to provide early information to PCB board and system designers.

This section includes the following topics:

- “Early Power Estimation” on page 7
- “Planning for Device Configuration” on page 8
- “Planning for On-Chip Debugging” on page 13

Early Power Estimation

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

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


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

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

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

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

When the design is complete, you must perform a complete power analysis to check the power consumption more accurately. The PowerPlay Power Analyzer tool in the Quartus II software provides an accurate estimation of power, ensuring that thermal and supply budgets are not violated. For the most accurate power estimation, use gate-level simulation results with a .vcd output file from the Quartus II Simulator or a third-party simulation tool. Refer to [“Power Analysis” on page 50](#).

 The PowerPlay EPE spreadsheets and user guides for each supported device family are available at the [PowerPlay Early Power Estimators \(EPE\) and Power Analyzer literature page](#) on the Altera website. For more information about using the PowerPlay EPE spreadsheet, refer to the [PowerPlay Early Power Estimator User Guide](#). For more information about power estimation and analysis, refer to the [PowerPlay Power Analysis](#) chapter in volume 3 of the *Quartus II Handbook*.


Planning for Device Configuration

Arria II devices are based on SRAM cells, so you must download configuration data to the Arria II device each time the device powers up, because SRAM memory is volatile. Consider whether or not you require multiple configuration schemes, such as one for debug or testing and another for the production environment. Choosing the device configuration method early allows system and board designers to determine what companion devices, if any, are required for the system. Your board layout also depends on the configuration method you plan to use for the programmable device, because different schemes require different connections. For board design guidelines related to configuration pins and connecting devices for configuration, refer to [“Pin Connection Considerations for Board Design” on page 15](#).

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

This section includes the following topics:

- [“Configuration Scheme Selection” on page 9](#)
- [“Configuration Features” on page 11](#)
- [“Quartus II Configuration Settings” on page 12](#)

- 
- For more information about configuration, refer to the [Configuration, Design Security, and Remote System Upgrades in Arria II Devices](#) chapter in volume 1 of the *Arria II Device Handbook*. For more information, refer to the [Configuration Center](#). This web page includes links to [JTAG Configuration & ISP Troubleshooter](#) and [FPGA Configuration Troubleshooter](#) that you can use to help debug configuration problems.


Configuration Scheme Selection

You can configure Arria II devices with one of four configuration schemes:

- **Fast passive parallel (FPP)**—A controller supplies the configuration data in a parallel manner to the Arria II FPGA.
- **Fast active serial (AS)**—The Arria II FPGA controls the configuration process and gets the configuration data from a serial configuration (EPCS) device.
- **Passive serial (PS)**—A controller supplies the configuration data serially to the Arria II FPGA.
- **Joint Test Action Group (JTAG)**—Arria II FPGA is configured through the IEEE Standard 1149.1 interface with a download cable, or using a MAX[®] II or a MAX V device or microprocessor with flash memory.

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


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

- 
- For more information about the supported configuration schemes, refer to the [Configuration Center](#). For complete information about the Arria II supported configuration schemes, how to execute the required configuration schemes, and all of the necessary option pin settings, including the MSEL pin settings, refer to the [Configuration, Design Security, and Remote System Upgrades in Arria II Devices](#) chapter in volume 1 of the *Arria II Device Handbook*.

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

Serial Configuration Devices

The Altera EPCS devices are used in the Fast AS configuration scheme. Serial configuration devices offer a low-cost, low pin-count configuration solution.

- 
- For information about EPCS devices, refer to the [Serial Configuration Devices \(EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128\) Datasheet](#) in volume 2 of the *Configuration Handbook*.

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

- For more information about SRunner, refer to [AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming](#) and the source code on the Altera website.

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

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

- For more information about the SFL, refer to [AN 370: Using the Serial Flash Loader with the Quartus II Software](#).

Download Cables

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

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

- [ByteBlaster II Download Cable User Guide](#)
- [USB Blaster Download Cable User Guide](#)
- [EthernetBlaster Communications Cable User Guide](#)

MAX II and MAX V Parallel Flash Loader

If your system already contains common flash interface (CFI) flash memory, you can use it for Arria II device configuration storage. The parallel flash loader (PFL) feature with MAX II and MAX V devices allows you to program CFI flash memory devices through the JTAG interface. It also provides the logic to control configuration from the flash memory device to the Arria II device and supports compression to reduce the size of your configuration data. Both PS and FPP configuration modes are supported using this PFL feature.

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

- For more information about PFL, refer to [Parallel Flash Loader Megafunction User Guide](#).

Configuration Features

This section describes Arria II configuration features and how they affect your design process.

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

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

Data Compression

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

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

Design Security Using Configuration Bitstream Encryption

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

The design security feature is available in the FPP, fast AS, or PS configuration schemes. In FPP, the host controller must send a DCLK that is $\times 4$ the data rate and the configuration data must be valid for four DCLK cycles. The design security feature is not available in JTAG configuration schemes.

Remote System Upgrades

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

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

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

- For more information about the ALTREMOTE_UPDATE megafunction, refer to the *Remote Update Circuitry (ALTREMOTE_UPDATE) Megafunction User Guide*.

SEU Mitigation and CRC Error Checks

Dedicated circuitry is built into Arria II devices for a cyclic redundancy check (CRC) error detection feature that optionally checks for single event updates (SEUs) continuously and automatically. This allows you to confirm that the configuration data stored in a Arria II device is correct and alerts the system to a configuration error. To take advantage of the SEU mitigation features, use the appropriate megafunction for CRC error detection. Use the CRC_ERROR or CRITICAL_ERROR pin to flag errors and design your system to take appropriate action. If not enabled for their CRC function, these pins are available as design I/O.

- For more information about SEUs, refer to the *SEU Mitigation in Arria II Devices* chapter in volume 1 of the *Arria II Device Handbook*.

Quartus II Configuration Settings

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

Optional Configuration Pins

You can enable the following optional configuration pins:

- CLKUSR —The **Enable user-supplied start-up clock (CLKUSR)** option enables you to select which clock source is used for initialization, either the internal oscillator or an external clock provided on the CLKUSR pin.
- INIT_DONE—To check if the device has completed initialization and is in user mode, you can monitor the INIT_DONE pin. Enable the pin with the **Enable INIT_DONE output** option. The INIT_DONE pin is an open-drain output and requires an external pull-up to V_{CCPGM} .

- Plan board design to support optional configuration pins CLKUSR and INIT_DONE, as required.

Restart Configuration after Error

You can enable the **Auto-restart after configuration error** option so that when a configuration error occurs, the device drives nSTATUS low, which resets the device internally. The device releases its nSTATUS pin after a reset time-out period. The nSTATUS pin requires an external 10-k Ω pull-up resistor to V_{CCPGM} , unless it is connected to an external configuration device which provides an internal pull-up resistor.

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

Planning for On-Chip Debugging


On-chip debugging is an optional step in the design flow, and different debugging tools work better for different systems and different designers. Evaluate on-chip debugging options early in your design process to ensure that your system board, Quartus II project, and design are able to support the appropriate options. Planning can reduce time spent debugging, and eliminates any need to make changes later to accommodate your preferred debugging methodologies. Adding debug pins might not be enough, because of internal signal accessibility and I/O pin accessibility on the device. First, select your preferred debugging tool(s) described in [“On-Chip Debugging Tools” on page 13](#) and then refer to [“Planning Guidelines for Debugging Tools” on page 14](#).

On-Chip Debugging Tools

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

- **SignalProbe incremental routing**—Quickly routes internal signals to I/O pins without affecting the routing of the original design. Starting with a fully routed design, you can select and route signals for debugging to either previously reserved or currently unused I/O pins.
- **SignalTap II Embedded Logic Analyzer**—Probes the state of internal and I/O signals without the use of external equipment or extra I/O pins, while the design is running at full speed in an FPGA device. Defining custom trigger-condition logic provides greater accuracy and improves the ability to isolate problems. It does not require external probes or changes to the design files to capture the state of the internal nodes or I/O pins in the design; all captured signal data is stored in device memory until you are ready to read and analyze the data. The SignalTap II Embedded Logic Analyzer works best for synchronous interfaces. For debugging asynchronous interfaces, consider using SignalProbe or an external logic analyzer to view the signals most accurately.
- **Logic Analyzer Interface**—Enables you to connect and transmit internal FPGA signals to an external logic analyzer for analysis, allowing you to take advantage of advanced features in your external logic analyzer or mixed signal oscilloscope. You can use this feature to connect a large set of internal device signals to a small number of output pins for debugging purposes and it can multiplex signals with design I/O pins if required.
- **In-System Memory Content Editor**—Provides read and write access to in-system FPGA memories and constants through the JTAG interface, so you can test changes to memory content and constant values in the FPGA while the device is functioning in the system.
- **In-System Sources and Probes**—Sets up customized register chains to drive or sample the instrumented nodes in your logic design, providing an easy way to input simple virtual stimuli and capture the current value of instrumented nodes.

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

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

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

Planning Guidelines for Debugging Tools


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

16. Select on-chip debugging scheme(s) early to plan memory and logic requirements, I/O pin connections, and board connections.
17. If you want to use the SignalTap II Embedded Logic Analyzer, Logic Analyzer Interface, In-System Memory Content Editor, In-System Sources and Probes, or Virtual JTAG Megafunction, plan your system and board with JTAG connections that are available for debugging.
18. Plan for the small amount of additional logic resources used to implement the JTAG hub logic for JTAG debugging features.
19. For debugging with the SignalTap II Embedded Logic Analyzer, reserve device memory resources to capture data during system operation.
20. Reserve I/O pins for debugging with SignalProbe or the Logic Analyzer Interface so you do not have to change the design or board to accommodate debugging signals later.
21. Ensure board supports debugging mode where debugging signals do not affect system operation.
22. Incorporate pin header or mictor connector as required for an external logic analyzer or mixed signal oscilloscope.
23. 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.
24. To use the SLD_VIRTUAL_JTAG megafunction for custom debugging applications, instantiate it in the HDL code as part of the design process.
25. To use the In-System Sources and Probes feature, instantiate the megafunction in the HDL code.
26. To use the In-System Memory Content Editor for RAM or ROM blocks or the LPM_CONSTANT megafunction, turn on the **Allow In-System Memory Content Editor to capture and update content independently of the system clock** option for the memory block in the MegaWizard™ Plug-In Manager.

Pin Connection Considerations for Board Design

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

- “Device Power Up” on page 15
- “Power Pin Connections and Power Supplies” on page 17
- “Configuration Pin Connections” on page 18
- “Board-Related Quartus II Settings” on page 21
- “Signal Integrity Considerations” on page 22
- “Board-Level Simulation and Advanced I/O Timing Analysis” on page 24
- “I/O and Clock Planning” on page 25 details the I/O signal connections for the FPGA, which also affect the board design.

 For detailed board design guidelines, refer to [Board Design Resource Center](#) page of the Altera website. This Resource Center points to application notes and other documentation that can help you implement successful high-speed PCBs that integrate Altera devices with other elements.

Device Power Up

Arria II device I/O pins are hot-socketing compliant without the need for external components. You can insert or remove an Arria II device from a powered-up system board without damaging or interfering with normal system and board operation.

27. Design board for power up: Arria II output buffers are tri-stated until the device is configured, and configuration pins drive out.
28. Design voltage supply power ramps to be monotonic.

You can drive signals into I/O pins before or during power up or power down without damaging the device. Arria II GX devices support power up or power down of the V_{CCIO} , V_{CC} , and V_{CCPD} power supplies to simplify system level design. Arria II GZ devices support power up and power down of V_{CCIO} , V_{CC} , V_{CCPD} , and V_{CCPGM} supplies in any sequence (provided that the V_{CC} powers up fully before V_{CCAUX}) to simplify system level design. The individual power supply ramp-up and ramp-down rates can range from 50 μ s to 100 ms for Arria II devices. The power ramp must be monotonic.

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

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

The power-on-reset (POR) circuit keeps the entire system in reset until the power supply voltage levels have stabilized on power up. After power up, the device does not release `nSTATUS` until V_{CCCB} , V_{CCA_PLL} , V_{CC} , V_{CCPD} , and V_{CCIO} for I/O banks 3C and 8C (for Arria II GX devices) and V_{CC} , V_{CCAUX} , V_{CCCB} , V_{CCPGM} , and V_{CCPD} (for Arria II GZ devices) are above the device's POR trip point. On power down for

Arria II GX devices, brown-out occurs if the V_{CC} drops below the POR trip point and any of the V_{CC} , V_{CCPD} , or V_{CCIO} for I/O banks 3C and 8C drops below the threshold level of hot-socket circuitry. On power down for Arria II GZ devices, brown-out occurs if the V_{CC} , V_{CCAUX} , V_{CCCB} , V_{CCPGM} , or V_{CCPD} voltages drops below the threshold voltage.

In Arria II devices, you can select between a fast or standard POR time. For Arria II GX devices, selection depends on the MSEL pin settings. For Arria II GZ devices, selection depends on the PORSEL input pin. PORSEL = L is set as standard POR time. PORSEL = H is set as fast POR time. Fast POR time is $4\text{ ms} < T_{POR} < 12\text{ ms}$ for a fast configuration time. Standard POR time is $100\text{ ms} < T_{POR} < 300\text{ ms}$ for a lower power-ramp rate.

The fast POR time is typically 4 ms for fast configuration time. The standard POR time is typically 100 ms, which has a lower power-ramp rate. In both cases, you can extend the POR time with an external component to assert the nSTATUS pin low. Extend POR time if the board cannot meet the maximum power ramp time specifications to ensure the device configures properly and enters user mode.

29. Set POR time to ensure power supplies are stable.

When power is applied to an Arria II device, a POR event occurs if the power supply reaches the recommended operating range within a certain period of time (specified as a maximum power supply ramp time; t_{RAMP}). The maximum power supply ramp time for Arria II GX devices is 100 ms for normal POR or 4 ms for fast POR, while the minimum power supply ramp time is 50 μs . The maximum power supply ramp time for Arria II GZ devices is 100 ms for normal POR (PORSEL = 0) or 4 ms for fast POR (PORSEL = 1), while the minimum power supply ramp time is 50 μs .

-  For more information, refer to the *Power Management in Arria II Devices* chapter in volume 1 of the *Arria II Device Handbook*.

Altera does not provide reliability data or in-rush current specifications when the device power rails do not meet the published t_{RAMP} specifications. Altera recommends the power-up timing of each rail is compliant with the t_{RAMP} specification when designing a multi-rail powered system. Altera device specifications and reliability data are based on power supplies following the t_{RAMP} specifications. The device may consume higher current during power up when the t_{RAMP} specifications are not met.

-  For Arria II GZ devices, Altera requires powering up V_{CC} before V_{CCAUX} .

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

-  For more information, refer to the *Power Management in Arria II Devices* chapter in volume 1 of the *Arria II Device Handbook*.

Power Pin Connections and Power Supplies

Review the power pin connection guidelines to determine the power supplies that are required in your system and which voltage inputs can share supplies. The Arria II core voltage V_{CC} is 0.90 V and other voltage inputs require 1.1, 1.5, and 2.5 V.

For Arria II GX devices, the I/O voltage V_{CCIO} connections depend on the I/O standards and support 1.2, 1.5, 1.8, 2.5, 3.0, and 3.3V. The I/O voltage V_{CCPD} connection depends on the V_{CCIO} voltage and supports 2.5V, 3.0V, and 3.3V.

For Arria II GZ devices, the I/O voltage V_{CCIO} connections depend on the I/O standards and support 1.2, 1.5, 1.8, 2.5, and 3.0V. The I/O voltage V_{CCPD} connection depends on the V_{CCIO} voltage and supports 2.5V and 3.0V.



The device output pins do not meet the I/O standard specifications if the V_{CCIO} level is out of the recommended operating range for the I/O standard.



For a list of the supply voltages required for the Arria II device and their recommended operation conditions, refer to the [Power Management in Arria II Devices](#) chapter in volume 1 of the *Arria II Device Handbook*.

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



For guidelines on power supply types and power supply sharing or isolation, review the [Arria II Device Family Pin Connection Guidelines](#) and the [Device Schematic Review Worksheets](#) page of the Altera website.

Decoupling Capacitors

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

Arria II devices include embedded on-die decoupling capacitors to provide high-frequency decoupling. These low-inductance capacitors suppress power noise for excellent signal integrity performance, and reduce the number of external PCB decoupling capacitors, saving board space, reducing cost, and greatly simplifying PCB design.

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

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

- For more information about the PDN design and optimization process, refer to the [Power Delivery Network \(PDN\) Tool User Guide](#). You can also download the [Power Delivery Network \(PDN\) Tool](#).

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

PLL Board Design Guidelines

For more information about designing your clock and PLL scheme, refer to “[Clock and PLL Selection](#)” on page 34 and “[PLL Feature Guidelines](#)” on page 35. The following checklist items provide several considerations to design a power system for PLL usage and minimize jitter, because PLLs contain analog components embedded in a digital device.

32. Connect all PLL power pins to reduce noise even if the design does not use all the PLLs: V_{CCA_PLL} to 2.5-V and V_{CCD_PLL} to 0.9-V.
33. Run a thick trace (at least 20 mils) from the power supply to each PLL power pin.
34. Connect all PLL digital power pins to the quietest digital supply on the board.
35. Use ferrite beads to isolate the PLL analog power supply from the digital power supply.

- For more board design guidelines for PLL power supplies, refer to the **General Board Design Considerations/Guidelines** section of the [Board Design Resource Center](#).

Transceiver Board Design Guidelines

- For detailed guidelines specific to transceiver design, refer to the [Transceiver Architecture in Arria II Devices](#) section in volume 2 of the *Arria II Device Handbook*.

Configuration Pin Connections

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

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

- For a list of the dedicated and dual-purpose configuration pins, and a description of the function and connection guidelines, refer to the [Configuration, Design Security, and Remote System Upgrades in Arria II Devices](#) chapter in volume 1 of the *Arria II Device Handbook*.

DCLK and TCK Signal Integrity

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

37. Design configuration DCLK and TCLK pins to be noise-free.

When designing the board, lay out the TCK and DCLK traces with the same techniques used to lay out a clock line. Any overshoot, undershoot, ringing, or other noise on the TCK signal can affect JTAG configuration. A noisy DCLK signal can affect configuration and cause a CRC error. For a chain of devices, noise on any of the TCK or DCLK pins in the chain could cause JTAG programming or configuration to fail for the entire chain.

- For more information about connecting devices in a chain, refer to the *Configuration, Design Security, and Remote System Upgrades in Arria II Devices* chapter in volume 1 of the *Arria II Device Handbook*.

JTAG Pins

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

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

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

JTAG Pin Connections

A device operating in JTAG mode uses four required pins—TDI, TDO, TMS, and TCK—and one optional pin, TRST.(for Arria II GZ devices only). The TCK pin has an internal weak pull-down resistor, and the TDI, TMS, and TRST pins have weak internal pull-up resistors (typically 25 k Ω).

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

If you have more than one device in the chain, connect the TDO pin of a device to the TDI pin of the next device in the chain.

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

- To disable the JTAG state machine during power-up, pull the TCK pin low through a resistor to ensure that an unexpected rising edge does not occur on TCK.
- Pull TMS high through a resistor. The pull-up resistor value can either be 1 k Ω or 10 k Ω .
- For Arria II GZ devices, connect TRST to V_{CCPD} through a 1 k Ω resistor. (Connecting the pin to ground disables the JTAG circuitry.)

Download Cable Operating Voltage

The operating voltage supplied to the Altera download cable by the target board through the 10-pin header determines the operating voltage level of the download cable.

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

In a JTAG chain containing devices with different voltage, the devices with a higher voltage must drive the devices with the same or lower voltage. A 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 *JTAG Boundary-Scan Testing in Arria II Devices* chapter in volume 1 of the *Arria II Device Handbook*.

JTAG Signal Buffering

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

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

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

- Buffer JTAG signals per the recommendations, especially for connectors or if the cable drives more than three devices.
- If your device is in a configuration chain, ensure all devices in the chain are connected properly.

MSEL Configuration Mode Pins

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

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

Other Configuration Pins

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

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

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

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

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

Board-Related Quartus II Settings

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

Device-Wide Output Enable Pin

Arria II devices support an optional chip-wide output enable that allows you to override all tri-states on device I/Os. When this `DEV_OE` pin is driven low, all I/O pins are tri-stated; when this pin is driven high, all pins behave as programmed. To use this chip-wide output enable, turn on **Enable device-wide output enable (DEV_OE)** in the Quartus II software before compiling your design on the **General** tab in the **Device & Pin Options** dialog box. If you enable this option in the Quartus II software, be sure to connect the pin on the board properly and do not leave it floating.

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

Unused Pins

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

50. Specify the reserved state for unused I/O pins.

To improve signal integrity, set unused pins as outputs that drive ground and tie them directly to the ground plane on the board. Doing so reduces inductance by creating a shorter return path and reduces noise on the neighboring I/O. To reduce power dissipation, connect unused dedicated clock input pins to ground on the board, and set unused I/O pins and dual purpose clock I/O pins as inputs tri-stated with weak pull-up resistor enabled. The weak pull-up resistor prevents the I/O pin from floating. If you set a reserved state for unused pins, be sure not to connect those pins to other devices on the board. To make the setting that is appropriate for your design, choose one of the five allowable states for **Reserve all unused pins** on the **Unused Pins** tab in the **Device & Pin Options** dialog box, or apply the **Reserve Pin** assignment to specific pins in the Pin Planner.

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

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

Signal Integrity Considerations

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

High-Speed Board Design

If your design has high-speed signals, especially with Arria II device high-speed transceivers, the board design has a major impact on the signal integrity in the system.

 For detailed information about signal integrity and board design, refer to the [Board Design Resource Center](#) page of 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](#)

Voltage Reference Pins

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

52. Design VREF pins to be noise-free.

For more information about voltage reference pins and I/O standards, refer to [“I/O Features and Pin Connections”](#) on page 27.

Simultaneous Switching Noise

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

- 53. Break out large bus signals on board layers close to the device to reduce crosstalk.
- 54. Route traces orthogonally if two signal layers are next to each other, whenever possible. Use a separation of 2 to 3 times the trace width.


I/O Termination

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

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

Arria II on-chip series termination (R_S OCT) and on-chip parallel termination (R_T OCT) (for Arria II GZ devices only) provide the convenience of no external components. Alternatively, you can use external pull-up resistors to terminate voltage-referenced I/O standards such as SSTL and HSTL.

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

 Certain dedicated clock input pairs do not support differential termination. For more information, refer to the [Arria II Device Family Pin Connection Guidelines](#).

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

For more information about on-chip termination features and limitations, refer to [“I/O Features and Pin Connections” on page 27](#).

Board-Level Simulation and Advanced I/O Timing Analysis

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

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

56. Perform board-level simulation using IBIS models.



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

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

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

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



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

I/O and Clock Planning

Planning and allocating I/O and clock resources is an important task with the high pin counts and advanced clock management and transceiver features in Arria II devices. Various considerations are important to effectively plan the available I/O resources to maximize utilization and prevent issues related to signal integrity. Good clock management systems are also crucial to the performance of an FPGA design.

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

This section describes the following topics:

- [“Making FPGA Pin Assignments” on page 25](#)
- [“Early Pin Planning and I/O Assignment Analysis” on page 26](#)
- [“I/O Features and Pin Connections” on page 27](#)
- [“Clock and PLL Selection” on page 34](#)
- [“PLL Feature Guidelines” on page 35](#)
- [“Clock Control Block” on page 36](#)
- [“Simultaneous Switching Noise” on page 37](#)

Making FPGA Pin Assignments

With the Quartus II Pin Planner GUI, you can identify I/O banks, V_{REF} groups, and differential pin pairings to help you through the I/O planning process. Right-click in the Pin Planner spreadsheet interface and click the **Pin Finder** to search for specific pins. If migration devices are selected, as described in [“Vertical Device Migration” on page 6](#), the Pin Migration view highlights pins that change function in the migration device when compared with the currently selected device.

58. Use the Quartus II Pin Planner to make pin assignments.


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

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

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

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

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

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

Early Pin Planning and I/O Assignment Analysis

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

The Quartus II Pin Planner enables easy I/O pin assignment planning, assignment, and validation, as described in “[Making FPGA Pin Assignments](#)” on page 25. The Quartus II **Start I/O Assignment Analysis** command checks that the pin locations and assignments are supported in the target FPGA architecture. Checks include reference voltage pin usage, pin location assignments, and mixing of I/O standards. You can use I/O Assignment Analysis to validate I/O-related assignments that you make or modify throughout the design process.

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

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

Early in the design process, the system architect typically has information about the standard I/O interfaces (such as memory and bus interfaces), IP cores that you can use in the design, and any other I/O-related assignments defined by system requirements. The Pin Planner **Create/Import Megafunction** feature interfaces with the MegaWizard Plug-In Manager, and enables you to create or import custom megafunctions and IP cores that use I/O interfaces. Enter PLL, LVDS, and transceiver blocks, including options such as dynamic phase alignment (DPA), because options


affect the pin placement rules. When you have entered as much I/O-related information as possible, generate a top-level design netlist file using the **Create Top-Level Design File** command. You can use the I/O analysis results to change pin assignments or IP parameters and repeat the checking process until the I/O interface meets your design requirements and passes the pin checks in the Quartus II software.

When planning is complete, you can pass the preliminary pin location information to PCB designers as described in the previous section. When the design is complete, use the reports and messages generated by the Quartus II Fitter for the final sign-off of pin assignments.

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

I/O Features and Pin Connections

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

 For guidelines about pin connection, review the *Arria II Device Family Pin Connection Guidelines*.

I/O Signaling Type

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

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

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

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

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

- 62. Plan signaling type based on system requirements.
- 63. Allow the software to assign locations for the negative pin in differential pin pairs.

Selectable I/O Standards and Flexible I/O Banks

Arria II I/O pins are arranged in groups called modular I/O banks. Depending on the device density, the number of I/O banks are up to 20, while the number of transceiver banks range from 1 to 12 banks. In Arria II devices, the maximum number of I/O banks per side is four or six, depending on the device density. When migrating between devices with a different number of I/O banks per side, it is the “B” bank that is removed or inserted.

- 64. Select suitable signaling type and I/O standard for each I/O pin.
- 65. Ensure that appropriate I/O standard is supported in targeted I/O bank.

Certain I/O banks on the top and bottom or left and right of the device support different I/O standards. For Arria II GX devices, the left side I/O banks contain high-speed transceiver banks with dedicated configuration banks at banks 3C and 8C. LVDS with DPA is supported in banks 5A, 5B, 6A, and 6B. For Arria II GZ devices, the dedicated configuration pins are located in Bank 1A and 1C. However, these banks are not dedicated configuration banks; therefore, user I/O pins are available in Bank 1A and Bank 1C.

You can assign I/O standards and make other I/O related settings in the Pin Planner. Be sure to use the correct dedicated pin inputs for signals such as clocks and global control signals, as described in [“Clock and PLL Selection” on page 34](#).

- 66. Place I/O pins that share voltage levels in the same I/O bank.
- 67. Verify that all output signals in each I/O bank are intended to drive out at the bank's V_{CCIO} voltage level.
- 68. Verify that all voltage-referenced signals in each I/O bank are intended to use the bank's V_{REF} voltage level.




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

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

An I/O bank including single-ended or differential standards can support voltage-referenced standards as long as all voltage-referenced standards use the same V_{REF} setting. For performance reasons, voltage-referenced input standards use their own V_{CCPD} level as the power source, so you can place voltage-referenced input signals in a bank with a V_{CCIO} of 2.5 V or below. However, voltage-referenced inputs that use $R_{T/OCT}$ require the V_{CCIO} of the I/O bank to match the voltage of the input standard, because parallel OCT uses the output buffer connected to V_{CCIO} . Voltage-referenced bi-directional and output signals must drive out at the I/O bank's V_{CCIO} voltage level.

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

Different I/O banks include different support for LVDS signaling, and the Arria II transceiver banks include additional support.

-  For information about the number of channels available for the LVDS I/O standard, refer to the *High-Speed Differential I/O Interfaces and DPA in Arria II Devices* chapter in volume 1 of the *Arria II Device Handbook*. For more information about transceiver bank-related features, refer to the *Transceiver Architecture in Arria II Devices* chapter in volume 1 of the *Arria II Device Handbook*.
-  For more information about I/O, refer to the *I/O Features in Arria II Devices* chapter in volume 1 of the *Arria II Device Handbook*. Refer to the Arria II I/O banks figure that shows the location of each I/O bank and what each bank supports. The figures describing the number of I/Os in each bank provide bank information specific to each device density. For information about which I/O standards you can combine in each bank, refer to the section describing I/O bank restrictions. For more information about LVDS restrictions, refer to the section describing I/O placement guidelines.
-  For the electrical characteristics of each I/O standard, refer to the *Device Datasheet for Arria II Devices* chapter in volume 3 of the *Arria II Device Handbook*.

Placement Guidelines Related to Differential I/O Pins

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

70. Use caution and follow guidelines for placement of pins located near LVDS I/O.

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

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

Memory Interfaces

Arria II devices provide an efficient architecture to quickly and easily fit wide external memory interfaces with their small modular I/O bank structure banks. Arria II devices support existing and emerging external DDR memory standards, such as DDR3, DDR2, DDR SDRAM, QDR II SRAM, QDR II+ SRAM, and RLDRAM II at frequencies of up to 400 MHz. Arria II GX devices support external memory interfaces on the top, bottom, and right I/O banks, while Arria II GZ devices support external memory interfaces on all sides of the I/O banks (except for the EP2AGZ300 and EP2AGZ350 devices in 780-pin FBGA, which only support external memory interfaces on the top and bottom I/O banks).


71. Use the ALTMEMPHY megafunction (or IP core) for Arria II GX devices and the UniPHY megafunction for Arria II GZ devices for each memory interface, and follow connection guidelines/restrictions in the appropriate documentation.


Self-calibrating megafunctions (ALTMEMPHY and UniPHY) are optimized to take advantage of the Arria II I/O structure and the Quartus II TimeQuest Timing Analyzer. The megafunctions allow you to set external memory interface features and helps set up the physical interface (PHY) best suited for your system with the highest reliable frequency of operation. When using the Altera memory controller MegaCore[®] functions, the ALTMEMPHY and UniPHY megafunctions are instantiated for you.

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

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

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

-  For more information about connecting an Arria II device with external memory devices, including the maximum supported clock rate for different memory standards and restrictions on pin placement, refer to the *External Memory Interfaces in Arria II Devices* chapter in volume 1 of the *Arria II Device Handbook*. For additional resources, refer to the [External Memory Solutions Center](#) page of the Altera website. For specific guidelines related to specific memory interfaces, refer to the appropriate application note.

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

Dual-Purpose and Special Pin Connections

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

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

If not enabled, the device-wide reset and clear pins are available as design I/Os. For more information, refer to [“Device-Wide Output Enable Pin” on page 21](#) and [“Register Power-Up Levels and Control Signals” on page 39](#).

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

Arria II I/O Features

The Arria II bidirectional I/O element (IOE) features for device interfaces assist in high-speed data transfer into and out of the device and reduce the complexity and cost of the PCB.

[Table 2](#) lists Arria II I/O features, provides usage information and design considerations, and provides references for more information about the features.

Table 2. Arria II I/O Features (Part 1 of 3)

Feature	Usage	Guidelines and More Information
MultiVolt I/O Interface	Allows all packages to interface with systems of different supply voltages. V_{CCIO} pins can be connected to a 1.2-, 1.5-, 1.8-, 2.5-, 3.0, or 3.3-V (Arria II GX devices only) power supply, depending on the output requirements. The output levels are compatible with systems of the same voltage as the power supply. You must connect V_{CCPD} power pins to 3.3-V for 3.3-V V_{CCIO} (for Arria II GX devices only) 3.0-V for 3.0-V V_{CCIO} , and 2.5-V for other I/O voltages.	For a summary of MultiVolt I/O support, a list of the supported I/O standards and the typical values for input and output V_{CCIO} , V_{CCPD} , V_{REF} , and board termination voltage (V_{TT}), refer to the previous sections and the I/O Features in Arria II Devices chapter in volume 1 of the Arria II Device Handbook . Altera recommends that you use an external clamp diode on the column I/O pins when the input signal is 3.0 V or 3.3 V.
3.3-V I/O Interface	Arria II I/O buffers support 3.3-V I/O standards as transmitters or receivers in your system. The output high voltage (V_{OH}), output low voltage (V_{OL}), input high voltage (V_{IH}), and input low voltage (V_{IL}) levels meet the 3.3-V I/O standards specifications when the V_{CCIO} voltage is powered by 3.3 V (Arria II GX) or 3.0 V (Arria II GZ).	To ensure device reliability and proper operation when interfacing with a 3.3 V I/O system, it is important to make sure that the absolute maximum ratings of the Arria II devices are not violated. Altera recommends performing IBIS simulation to determine that the overshoot and undershoot voltages are within the guidelines. For more guidelines, refer to the I/O Features in Arria II Devices chapter in volume 1 of the Arria II Device Handbook .

Table 2. Arria II I/O Features (Part 2 of 3)

Feature	Usage	Guidelines and More Information
Programmable Output Current Strength	Programmable current-strength control available for certain I/O standards. Can mitigate the effects of high signal attenuation due to a long transmission line or a legacy backplane. A higher current strength increases I/O performance, but also increases noise on the interface, so you can use current strength control to manage noise.	Ensure that the output buffer current strength is sufficiently high, but does not cause excessive overshoot or undershoot that violates voltage threshold parameters for the I/O standard. Altera recommends performing IBIS or SPICE simulations to determine the right current strength setting for your specific application. For a list of standards and settings, refer to the <i>I/O Features in Arria II Devices</i> chapter in volume 1 of the <i>Arria II Device Handbook</i> .
Programmable Slew Rate Control	Configure each pin for low-noise or high-speed performance. A faster slew rate provides high speed transitions. You can use faster slew rates to improve the available timing margin in memory-interface applications or when the output pin has a high-capacitive loading. A slow slew rate can help reduce system noise, but adds a nominal delay to rising and falling edges. You can use slew rate to reduce SSN.	Confirm that your interface meets its performance requirements if you use slower slew rates. Altera recommends performing IBIS or SPICE simulations to determine the right slew rate setting for your specific application.
Programmable IOE Delay	Programmable delay chains can ensure zero hold times, minimize setup times, or increase clock-to-output times. You can use delays as deskewing circuitry to ensure that all bits of a bus have the same delay going into or out of the device.	This feature helps read and time margins because it minimizes the uncertainties between signals in the bus. For the delay specifications, refer to the <i>Device Datasheet for Arria II Devices</i> chapter in volume 3 of the <i>Arria II Device Handbook</i> .
Programmable Output Buffer Delay	Delay chains in the single-ended output buffer can independently control the rising and falling edge delays of the output buffer.	You can use delays to adjust the output buffer duty cycle, compensate channel-to-channel skew, reduce SSO noise by deliberately introducing channel-to-channel skew, and improve high-speed memory-interface timing margins.
Open-Drain Output	When configured as open-drain, the logic value of the output is either high-Z or 0. Used in system-level control signals that can be asserted by multiple devices in the system.	Typically, an external pull-up resistor is required to provide logic high.
Bus Hold	Weakly holds the signal on an I/O pin at its last driven state until the next input signal is present, using a resistor with a nominal resistance (RBH) of approximately 7 k Ω . With this feature, an external pull-up or pull-down resistor to hold a signal level when the bus is tri-stated is not required. The circuitry also pulls non-driven pins away from the input threshold voltage where noise can cause unintended high-frequency switching.	If the bus-hold feature is enabled, you cannot use the programmable pull-up option. Disable the bus-hold feature if the I/O pin is configured for differential signals. For the specific sustaining current driven through this resistor and the overdrive current used to identify the next driven input level for each V _{CCIO} voltage level, refer to the <i>Device Datasheet for Arria II Devices</i> chapter in volume 1 of the <i>Arria II Device Handbook</i> .
Programmable Pull-Up Resistor	A pull-up resistor (typically 25 k Ω) weakly holds the I/O to the V _{CCIO} level when in user mode. You can use with open-drain output to eliminate the requirement for an external pull-up resistor.	If the programmable pull-up option is enabled, you cannot use the bus-hold feature.

Table 2. Arria II I/O Features (Part 3 of 3)

Feature	Usage	Guidelines and More Information
PCI Clamping Diode	You can use to protect the pin from excessive overshoot voltage in PCI/PCI-X I/O standard interfaces.	—
On-Chip Termination (OCT)	Driver-impedance matching provides the I/O driver with controlled output impedance that closely matches the impedance of the transmission line to significantly reduce reflections. OCT maintains signal quality, saves board space, and reduces external component costs. The Arria II devices support R_S OCT with or without calibration, R_D OCT without calibration, and R_T OCT with calibration (Arria II GZ devices only). The Arria II devices provide R_S OCT, R_D OCT, and R_T OCT value of 25- Ω , 50- Ω , and a 100- Ω for LVDS. For more information, refer to the <i>I/O Features in Arria II Devices</i> and <i>High-Speed Differential I/O Interfaces and DPA in Arria II Devices</i> chapters in volume 1 of the <i>Arria II Device Handbook</i> .	R_S OCT is supported in the same I/O bank for different I/O standards if they use the same V_{CCIO} supply voltage. You can independently configure each I/O in an I/O bank to support R_S OCT or programmable current strength. You cannot configure both R_S OCT and programmable current strength or slew rate control for the same I/O buffer. Not all I/O pins support R_D OCT. For more information, refer to the <i>I/O Features in Arria II Devices</i> and <i>High-Speed Differential I/O Interfaces and DPA in Arria II Devices</i> chapters in volume 1 of the <i>Arria II Device Handbook</i> , and <i>Arria II Device Family Pin Connection Guidelines</i>
Programmable Pre-Emphasis and V_{OD}	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.	For more information, refer to the <i>High-Speed Differential I/O Interfaces and DPA in Arria II Devices</i> chapter in volume 1 of the <i>Arria II Device Handbook</i> .
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 <i>High-Speed Differential I/O Interfaces and DPA in Arria II Devices</i> chapter in volume 1 of the <i>Arria II Device Handbook</i> .
Dedicated Differential I/O SERDES Circuitry with DPA Support	Arria II GX devices have built-in SERDES circuitry that supports high-speed LVDS interfaces on the right side of the device, while Arria II GZ devices supports high-speed LVDS interfaces on both sides of the device. Dynamic Phase Alignment (DPA) circuitry automatically chooses the best phase to compensate the skew between source synchronous clock and received serial data.	If you want to use DPA, be sure to enable the feature in the MegaWizard Plug-In Manager so that the design uses the correct PLLs. DPA usage adds constraints on the placement of high-speed differential channels. For more information, refer to the detailed feature description and placement guidelines in the <i>High-Speed Differential I/O Interfaces and DPA in Arria II Devices</i> chapter in volume 1 of the <i>Arria II Device Handbook</i> .

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

- 74. Check available device I/O features that can help I/O interfaces: current strength, slew rate, I/O delays, open-drain, bus hold, programmable pull-up resistors, PCI clamping diodes, programmable pre-emphasis, and VOD.
- 75. Consider on-chip termination features to save board space.
- 76. Check that the required termination scheme is supported for all pin locations.

- 77. Choose the appropriate mode of DPA, non-DPA, or soft-CDR for high-speed LVDS interfaces.
- 78. If you want to use DPA, be sure to enable the feature so that the design uses the correct PLLs and follow DPA placement guidelines.

Clock and PLL Selection

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

Arria II devices provide dedicated low-skew and high fan-out routing networks. They are organized in a hierarchical structure that provides up to 192 unique clock domains within the device and allows up to 60 unique clock sources per device quadrant. There are up to six PLLs per Arria II GX device and up to seven independently-programmable outputs per PLL. Arria II GZ devices offer up to eight PLLs that provide robust clock management and synthesis for device clock management, external system clock management, and high-speed I/O interfaces. For Arria II GX devices, you can use up to six differential dedicated global clock input pins or 12 single-ended clock inputs that can drive global or regional clock networks. For Arria II GZ devices, you can use up to 16 differential dedicated global clock input pins or 32 single-ended clock inputs that can drive either the global or regional clock networks.

- 79. Use correct dedicated clock pins and routing signals for clock and global control signals.

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

- 80. Use the device PLLs for clock management.

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

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

- The global clock (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 regional clock (RCLK) networks only pertain to the quadrant they drive into. The RCLK networks provide the lowest clock delay and skew for logic contained within a single device quadrant.
- The internal logic drives the GCLKs and RCLKs to create internally generated global or regional clocks and other high fan-out control signals; for example, synchronous or asynchronous clears and clock enables.

- PLLs cannot be driven by internally-generated GCLKs or RCLKs. The input clock to the PLL must come from dedicated clock input pins or pin/PLL-fed GCLKs or RCLKs.
 - Periphery clock (PCLK) networks are a collection of individual clock networks driven from the periphery of the Arria II device. Clock outputs from the DPA block, PLD-transceiver interface, row I/O pins, and internal logic can drive the PCLK networks. These PCLKs have higher skew compared to GCLK and RCLK networks and can be used instead of general purpose routing to drive signals into and out of the Arria II device.
81. Analyze input and output routing connections for each PLL and clock pin. Ensure PLL inputs come from dedicated clock pins or from another PLL.

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

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


PLL Feature Guidelines


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

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

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

Arria II PLLs provide robust clock management and synthesis for device clock management, external system clock management, and high speed I/O interfaces. All Arria II PLLs have the same core analog structure and support features with minor differences in the features that are supported for Arria II GZ devices. You can use some of the following additional features when planning your PLL design.

 For more information about PLL features, refer to the *Clock Networks and PLLs in Arria II Devices* chapter in volume 1 of the *Arria II Device Handbook*.

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

Clock Feedback Mode

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

83. Ensure you select the correct PLL feedback compensation mode.

Clock Switchover

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

Dynamic Reconfiguration

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

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

-  For more information about the `ALTPLL_RECONFIG` megafunction, refer to the *Phase-Locked Loops Reconfiguration (ALTPLL_RECONFIG) Megafunction Users Guide*.

Clock Control Block

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

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

Use these features to select different clock input signals, or power-down clock networks to reduce power consumption, without using any combinational logic in your design. In Arria II devices, the clock enable signals are supported at the clock network level instead of at the PLL output counter level, so you can turn off a clock even when a PLL is not being used. You can select a clock source statically with a setting in the Quartus II software, or dynamically using internal logic to drive the multiplexer select inputs.

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

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

Simultaneous Switching Noise

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

85. Analyze design for possible simultaneous switching noise problems.
86. Reduce number of pins that switch voltage at exactly the same time whenever possible.
87. Use differential I/O standards and lower-voltage standards for high switching I/Os.
88. Use lower drive strengths for high switching I/Os. The default drive strength setting might be higher than your design requires.
89. Reduce number of simultaneously switching output pins within each bank. Spread output pins across multiple banks if possible.
90. Spread switching I/Os evenly throughout the bank to reduce the number of aggressors in a given area to reduce SSN (when bank usage is substantially below 100%).
91. Separate simultaneously switching pins from input pins that are susceptible to SSN.
92. Place important clock and asynchronous control signals near ground signals and away from large switching buses.
93. Avoid using I/O pins one or two pins away from PLL power supply pins for high-switching or high drive strength pins.
94. Use staggered output delays to shift the output signals through time, or use adjustable slew rate settings.

For more information about the features you can use, refer to [“Arria II I/O Features” on page 31](#).

Design Entry

In the development of complex FPGA designs, design practices and coding styles have an enormous impact on your device's timing performance, logic utilization, and system reliability. You can also use megafunctions and SOPC Builder to help design your FPGA system. In addition, while planning and creating the design, plan for a hierarchical or team-based design to improve design productivity.

Design Recommendations

In a synchronous design, a clock signal triggers all events. When all of the registers' timing requirements are met, a synchronous design behaves in a predictable and reliable manner for all process, voltage, and temperature (PVT) conditions. You can easily target synchronous designs to different device families or speed grades.

95. Use synchronous design practices. Pay attention to clock signals.

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

Pay particular attention to your clock signals, because they have a large effect on your design's timing accuracy, performance, and reliability. Problems with clock signals can cause functional and timing problems in your design. Use dedicated clock pins and clock routing for best results. For clock inversion, multiplication, and division, use the device PLLs. For clock multiplexing and gating, use the dedicated clock control block or PLL clock switchover feature instead of combinational logic. Refer to ["PLL Board Design Guidelines" on page 18](#). If you must use internally generated clock signals, register the output of any combinational logic used as a clock signal to reduce glitches. For example, if you divide a clock using combinational logic, clock the final stage with the clock signal that was used to clock the divider circuit.

96. Use the Quartus II Design Assistant to check design reliability.

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

-  For more information about design recommendations and using the Design Assistant, refer to the [Design Recommendations for Altera Devices and the Quartus II Design Assistant](#) chapter in volume 1 of the *Quartus II Handbook*. You can also refer to industry papers for more information about multiple clock design. For a good analysis, refer to [Synthesis and Scripting Techniques for Designing Multi-Asynchronous Clock Designs](#).

Using Megafunctions

Altera provides parameterizable megafunctions that are optimized for Altera device architectures. You can save design time by using megafunctions instead of coding your own logic. Additionally, the Altera-provided megafunctions can offer more efficient logic synthesis and device implementation. You can scale the megafunction's size and set various options with parameters. Megafunctions include the library of parameterized modules (LPM) and Altera device-specific megafunctions. You can also take advantage of Altera and third-party IP and reference designs to save design time, as described in [“IP Selection” on page 3](#).

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


97. Use megafunctions with the MegaWizard Plug-In Manager.

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

Recommended HDL Coding Styles

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

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

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

Register Power-Up Levels and Control Signals


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

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

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

 For more information about LAB and ALM architecture, refer to the *Logic Array Blocks and Adaptive Logic Modules in Arria II Devices* chapter in volume 1 of the *Arria II Device Handbook*.

If the clock signal is not available when reset is asserted, an asynchronous reset is typically used to reset the logic. The recommended reset architecture is to allow the reset signal to be asserted asynchronously and de-asserted synchronously. The source of the reset signal is then connected to the asynchronous port of the registers, which can be directly connected to global routing resources. The synchronous de-assertion allows all state machines and registers to start at the same time. It also avoids the possibility that an asynchronous reset signal is released at or near the active clock edge of a flipflop, in which case the output of the flipflop could go to a metastable unknown state.


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

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

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

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

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

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

Registers in the device core always power up to a low (0) logic level in the physical device architecture. If you specify a high power-up level or a non-zero reset value (often called a preset signal), synthesis tools typically use the clear signals available on the registers and perform an optimization referred to as NOT-gate push back. NOT-gate push back adds an inverter to the input and the output of the register. The register hardware actually powers up and resets low, but the register output is inverted so the result at all destinations is a high logic value. If synthesis performs a NOT-gate push back optimization, the register behaves like a high (1) logic level during reset or power-up conditions. Regular register operation is not affected because the signal is inverted twice in the regular data path. This optimization does not negatively affect the fitting or performance of your design, but if you tap the register during on-chip verification, or view it during simulation, you must check the signal after the output inversion to obtain the correct value.

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

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

To implement a reset and preset signal on the same register, synthesis tools emulate the controls with logic and latches that can be prone to glitches because of the different delays between the different paths to the register. In addition, the power up value is undefined for these registers.

SOPC Builder

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

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



For more information about the Avalon interface, refer to the [Avalon Interface Specifications](#) manual.

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

- 101.□ Take advantage of SOPC Builder for system and processor designs.



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

Planning for Hierarchical and Team-Based Design

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

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



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

Planning Design Partitions

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

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

Plan your source code so that each design block is defined in a separate file. This allows the software to automatically detect changes to each block separately. If you use a third-party synthesis tool, create separate Verilog Quartus Mapping file (.vqm) or EDIF (.edf) netlist files for each design partition in your synthesis tool. If necessary, create separate projects within your synthesis tool so that the tool synthesizes each partition separately and generates separate output netlist files. Refer to your synthesis

tool documentation for information about support for Quartus II incremental compilation. Use hierarchy in your design to provide more flexibility when partitioning. Keep your design logic in the leaves of the hierarchy trees; that is, the top level of the hierarchy must have very little logic, and the lower-level design blocks contain the logic.

- 102. Follow recommendations to set up your source code and partition your design for incremental compilation; plan early in the design flow.
- 103. Perform timing budgeting and resource balancing between partitions to achieve best results, especially in team-based flows.

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

Planning in Bottom-Up and Team-Based Flows

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


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

Creating a Design Floorplan

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

- 104. Create a design floorplan for incremental compilation partitions, if required for your design flow.

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

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

Design Implementation, Analysis, Optimization, and Verification

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

Selecting a Synthesis Tool


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

Altera recommends 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.

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

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

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

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

Device Resource Utilization Reports

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

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

For FPGAs that use ALMs, a device with low logic utilization does not have the lowest ALM utilization possible. In addition, a design that is reported as close to 100% full might still have space for extra logic. The Fitter uses ALUTs in different ALMs, even when the logic can be placed within one ALM, so that it can achieve the best timing and routability results. Logic might be spread throughout the device when achieving these results. As the device fills up, the Fitter automatically searches for logic that can be placed together in one ALM. Other factors can also affect the logic fitting, including the number of different control signals used in the design as described in [“Register Power-Up Levels and Control Signals”](#) on page 39.

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

106. Review resource utilization and optimization reports after compilation.

Quartus II Messages

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

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



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

Timing Constraints and Analysis

In an FPGA design flow, accurate timing constraints allow timing-driven synthesis software and place-and-route software to obtain optimal results. Timing constraints are critical to ensure designs meet their timing requirements, which represent actual design requirements that must be met for the device to operate correctly. The Quartus II software optimizes and analyzes your design using different timing models for each device speed grade, so you must perform timing analysis for the correct speed grade. The final programmed device might not operate as expected if the timing paths are not fully constrained, analyzed, and verified to meet requirements.

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

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

A comprehensive static timing analysis includes analysis of register-to-register, I/O, and asynchronous reset paths. It is important to specify the frequencies and relationships for all clocks in your design. Use input and output delay constraints to specify external device or board timing parameters. Specify accurate timing requirements for external interfacing components to reflect the exact system intent. The TimeQuest Timing Analyzer performs static timing analysis on the entire system, using data required times, data arrival times, and clock arrival times to verify circuit performance and detect possible timing violations. It determines the timing relationships that must be met for the design to correctly function.

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

- 108. Ensure timing constraints are complete and accurate, including all clock signals and I/O delays.
- 109. Review the TimeQuest Timing Analyzer reports after compilation to ensure there are no timing violations.
- 110. Ensure that the input I/O times are not violated when data is provided to the Arria II device.



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

Recommended Timing Optimization and Analysis Assignments

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

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

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

112. Turn on **Enable multicorner timing analysis** under **Timing Analysis Settings** in the **Settings** dialog box, or use the `--multicorner` command line option.


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

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

113. Use `create_clock`, `create_generated_clock` to specify the frequencies and relationships for all clocks in your design.
114. Use `set_input_delay`, `set_output_delay` to specify external device or board timing parameters.
115. Use `derive_pll_clocks` to create generated clocks for all PLL outputs, according to the settings in the PLL megafunctions. Specify multi-cycle relationships for LVDS transmitters or receiver deserialization factors.
116. Use `derive_clock_uncertainty` to automatically apply inter-clock, intra-clock, and I/O interface uncertainties.
117. Use `check_timing` to generate a report for any problem with the design or applied constraints, including missing constraints.

Area and Timing Optimization

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

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

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

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

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

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

The Design Space Explorer (DSE) is a utility that automates the process of finding the optimal collection of Quartus II software settings for your design. The **Search for Best Performance** and **Search for Best Area** options under **Exploration Settings** use a predefined exploration space to target design performance or area improvements with multiple compilations. You can also set the **Optimization Goal** to **Optimize for Speed** or **Optimize for Area** using the **Advanced** tab in the DSE window. If you are interested in optimization for power usage, refer to *“Power Optimization” on page 51*.

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

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

- 119. Use Quartus II optimization features to achieve timing closure or improve the resource utilization.
- 120. Use the Timing and Area Optimization Advisors to suggest optimization settings.

Preserving Performance and Reducing Compilation Time

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

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

For guidelines and references, refer to *“Planning for Hierarchical and Team-Based Design” on page 42*.

- 122. Set up parallel compilation if you have multiple processors available for compilation.

The Quartus II software can run some algorithms in parallel to take advantage of multiple processors and reduce compilation time when more than one processor is available to compile the design. To set the number of processors available for a Quartus II compilation, specify the **Maximum processors allows for parallel compilation** on the **Compilation Process Settings** page of the **Settings** dialog box. By default, this option is set to **Use all available processors** so that parallel compilation is turned on by default.

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

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

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


Simulation

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

Altera provides the ModelSim-Altera simulator, which enables you to take advantage of advanced testbench capabilities and other features. In addition, the Quartus II EDA Netlist Writer can generate timing netlist files to support other third-party simulation tools such as Synopsys VCS, Cadence NC-Sim, and Aldec Active-HDL. Specify your simulation tool in the **EDA Tools Settings** page of the **Settings** dialog box to generate the appropriate output simulation netlist.

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

If you use a third-party simulation tool, use the software version that is supported with your Quartus II software version. The *Quartus II Release Notes* lists the version of each simulation tool that is officially supported with that particular version of the Quartus II software. Use the model libraries provided with your Quartus II software version, because libraries can change between versions, which might cause a mismatch with your simulation netlist. To create a testbench, on the Processing menu, point to **Start** and click **Start Testbench Template Writer**.

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

Formal Verification

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

Using a formal verification flow can impact performance results because it requires that certain logic optimizations be turned off, such as register retiming, and forces hierarchy blocks to be preserved, which can restrict optimization. Formal verification treats memory blocks as black boxes. Therefore, it is best to keep memory in a separate hierarchy block so that other logic does not get incorporated into the black box for verification. There are other restrictions that can also limit your design; consult the documentation for details. If formal verification is important to your design, it is easier to plan for limitations and restrictions in the beginning than to make changes later in the design flow.

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

- 125. Specify your third-party formal verification tool and use the correct supported version.
- 126. If using formal verification, check for support and design limitations.



For more information about formal verification flows, refer to [Section VI. Formal Verification](#) in volume 3 of the *Quartus II Handbook*.

Power Analysis

Before design completion, estimate power consumption using a spreadsheet as described in “[Early Power Estimation](#)” on [page 7](#). After compiling your design, analyze the power consumption and heat dissipation with the Quartus II PowerPlay Power Analyzer to ensure the design has not violated power supply and thermal budgets.

- 127. After compilation, analyze power consumption and heat dissipation in the PowerPlay Power Analyzer.
- 128. Provide accurate typical signal activities, preferably with a gate-level simulation **.vcd** file, to get accurate power analysis results.

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

- 129. Specify correct operating conditions for power analysis.

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

To calculate the dynamic, static, and I/O thermal power consumption, on the Processing menu, click **PowerPlay Power Analyzer Tool**. The tool also provides a summary of the signal activities used for analysis and a confidence metric that reflects the overall quality of the data sources for the signal activities.



The report is a power estimate based on the data provided, and is not a power specification. Always refer to the data sheet for your device.



For more information about power analysis, and recommendations for simulation settings for creating signal activity information, refer to the *PowerPlay Power Analyzer* chapter in volume 3 of the *Quartus II Handbook*. For more information about Signal Activity Files (.saf) and how to create them, refer to the *Quartus II Simulator* chapter in volume 3 of the *Quartus II Handbook*.

Power Optimization

Arria II devices utilize advanced architectural power reduction techniques to minimize power and deliver high performance.

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

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

Device and Design Power Optimization Techniques

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

- 130. Use recommended design techniques and Quartus II options to optimize your design for power consumption, if required.
- 131. Use the Power Optimization Advisor to suggest optimization settings.



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

Clock Power Management

Clocks represent a significant portion of dynamic power consumption because of their high switching activity and long paths. The Quartus II software automatically optimizes clock routing power by enabling only the portions of a clock network that are required to feed downstream registers. You can also use clock control blocks to dynamically enable or disable the clock network. When a clock network is powered down, all the logic fed by that clock network is in an off state, thereby reducing the overall power consumption of the device.

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

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

Memory Power Reduction

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


I/O Power Guidelines

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

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

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

Resistively terminated I/O standards dissipate significant static power because current is constantly driven into the termination network. Use the lowest drive strength that meets your speed and waveform requirements to minimize static power when using resistively terminated I/O standards.

 The power used by external devices is not included in the PowerPlay calculations, so be sure to include it separately in your system power calculations.

Quartus II Power Optimization Techniques

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

Optimizing your design source code for area saves power because fewer logic blocks are used; therefore, there is typically less switching activity. You can use the DSE and Power Optimization Advisor to provide additional suggestions to reduce power.

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

DSE

The DSE is a utility that automates the process of finding the optimal collection of Quartus II software settings for your design. The **Search for Lowest Power** option under **Exploration Settings** uses a predefined exploration space to target overall design power improvements with multiple compilations. You can also set **Optimization Goal to Optimize for Power** using the **Advanced** tab in the DSE window.

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

Power Optimization Advisor

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

Document Revision History

Table 3 lists the revision history for this document.

Table 3. Document Revision History

Date	Version	Changes
March 2011	2.0	<ul style="list-style-type: none"> Updated for the Quartus II software version 10.1 release Added Arria II GZ devices information Added “SOPC Builder”, “Dynamic Reconfiguration”, and “High-Speed Board Design” sections.
February 2009	1.0	Initial release

Design Checklist

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

- | Done | N/A | |
|--------------------------|--------------------------|--|
| <input type="checkbox"/> | <input type="checkbox"/> | Create detailed design specifications, and a test plan if appropriate. |
| <input type="checkbox"/> | <input type="checkbox"/> | Plan clock domains, clock resources, and I/O interfaces early with a block diagram. |
| <input type="checkbox"/> | <input type="checkbox"/> | Select IP that affects system design, especially I/O interfaces. |
| <input type="checkbox"/> | <input type="checkbox"/> | If you plan to use the Open Core Plus tethered mode for IP, ensure that your board design supports this mode of operation. |
| <input type="checkbox"/> | <input type="checkbox"/> | Take advantage of SOPC Builder for system and processor designs. |
| <input type="checkbox"/> | <input type="checkbox"/> | Select a device based on transceivers, I/O pin count, LVDS channels, package offering, logic/memory/multiplier density, PLLs, clock routing, and speed grade. |
| <input type="checkbox"/> | <input type="checkbox"/> | Reserve device resources for future development and debugging. |
| <input type="checkbox"/> | <input type="checkbox"/> | Consider vertical device migration availability and requirements. |
| <input type="checkbox"/> | <input type="checkbox"/> | Estimate power consumption with the Early Power Estimator spreadsheet to plan the cooling solution and power supplies before the logic design is complete. |
| <input type="checkbox"/> | <input type="checkbox"/> | Select a configuration scheme to plan companion devices and board connections. |
| <input type="checkbox"/> | <input type="checkbox"/> | If you want to use a flash device for the PFL, check the list of supported devices. |
| <input type="checkbox"/> | <input type="checkbox"/> | Ensure your configuration scheme and board support any required features: data decompression, design security, remote upgrades, SEU mitigation. |
| <input type="checkbox"/> | <input type="checkbox"/> | Plan board design to support optional configuration pins <code>CLKUSR</code> and <code>INIT_DONE</code> , as required. |
| <input type="checkbox"/> | <input type="checkbox"/> | Plan board design to use the Auto-restart after configuration error option. |
| <input type="checkbox"/> | <input type="checkbox"/> | Take advantage of on-chip debugging features to analyze internal signals and perform advanced debugging techniques. |
| <input type="checkbox"/> | <input type="checkbox"/> | Select on-chip debugging scheme(s) early to plan memory and logic requirements, I/O pin connections, and board connections. |
| <input type="checkbox"/> | <input type="checkbox"/> | If you want to use the SignalTap II Embedded Logic Analyzer, Logic Analyzer Interface, In-System Memory Content Editor, In-System Sources and Probes, or Virtual JTAG Megafunction, plan your system and board with JTAG connections that are available for debugging. |
| <input type="checkbox"/> | <input type="checkbox"/> | Plan for the small amount of additional logic resources used to implement the JTAG hub logic for JTAG debugging features. |
| <input type="checkbox"/> | <input type="checkbox"/> | For debugging with the SignalTap II Embedded Logic Analyzer, reserve device memory resources to capture data during system operation. |
| <input type="checkbox"/> | <input type="checkbox"/> | Reserve I/O pins for debugging with SignalProbe or the Logic Analyzer Interface so you do not have to change the design or board to accommodate debugging signals later. |
| <input type="checkbox"/> | <input type="checkbox"/> | Ensure board supports debugging mode where debugging signals do not affect system operation. |
| <input type="checkbox"/> | <input type="checkbox"/> | Incorporate pin header or mictor connector as required for an external logic analyzer or mixed signal oscilloscope. |
| <input type="checkbox"/> | <input type="checkbox"/> | 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. |
| <input type="checkbox"/> | <input type="checkbox"/> | To use the <code>SLD_VIRTUAL_JTAG</code> megafunction for custom debugging applications, instantiate it in the HDL code as part of the design process. |

Done N/A

- 25. To use the In-System Sources and Probes feature, instantiate the megafunction in the HDL code.
- 26. To use the In-System Memory Content Editor for RAM or ROM blocks or the LPM_CONSTANT megafunction, turn on the Allow In-System Memory Content Editor to capture and update content independently of the system clock option for the memory block in the MegaWizard™ Plug-In Manager.
- 27. Design board for power up: Arria II output buffers are tri-stated until the device is configured, and configuration pins drive out.
- 28. Design voltage supply power ramps to be monotonic.
- 29. Set POR time to ensure power supplies are stable.
- 30. Design power sequencing, voltage regulators, and ground connections for best device reliability.
- 31. Use the PDN tool to plan your power distribution netlist and decoupling capacitors.
- 32. Connect all PLL power pins to reduce noise even if the design does not use all the PLLs: V_{CCA_PLL} to 2.5-V and V_{CCD_PLL} to 0.9-V.
- 33. Run a thick trace (at least 20 mils) from the power supply to each PLL power pin.
- 34. Connect all PLL digital power pins to the quietest digital supply on the board.
- 35. Use ferrite beads to isolate the PLL analog power supply from the digital power supply.
- 36. Check that all configuration pin connections and pull-up or pull-down resistors are set correctly for your configuration scheme(s).
- 37. Design configuration $DCLK$ and $TCLK$ pins to be noise-free.
- 38. Connect JTAG pins to a stable voltage level if not in use.
- 39. Connect JTAG pins correctly to the download cable header. Ensure the pin order is not reversed.
- 40. To disable the JTAG state machine during power-up, pull the TCK pin low through a resistor to ensure that an unexpected rising edge does not occur on TCK .
- 41. Pull TMS high through a resistor. The pull-up resistor value can either be 1 k Ω or 10 k Ω .
- 42. For Arria II GZ devices, connect $TRST$ to V_{CCPD} through a 1 k Ω resistor. (Connecting the pin to ground disables the JTAG circuitry.)
- 43. Because the download cable interfaces with the JTAG pins of your device, ensure the download cable and JTAG pin voltages are compatible.
- 44. Buffer JTAG signals per the recommendations, especially for connectors or if the cable drives more than three devices.
- 45. If your device is in a configuration chain, ensure all devices in the chain are connected properly.
- 46. Connect $MSEL$ pins to select the configuration scheme; do not leave floating. For flexibility to change between configuration modes during testing or debugging, set up board to connect each pin to either V_{CCPGM} or GND with a 0- Ω resistor.
- 47. Connect nIO_PULLUP correctly to set up internal pull-up resistors.
- 48. Hold the nCE chip enable low during configuration, initialization, and user mode.
- 49. Turn on the device-wide output enable option, if required.
- 50. Specify the reserved state for unused I/O pins.
- 51. Carefully check the pin connections in the Quartus II software-generated .pin file. Do not connect RESERVED pins.
- 52. Design $VREF$ pins to be noise-free.
- 53. Break out large bus signals on board layers close to the device to reduce crosstalk.

- | Done | N/A |
|------|---|
| 54. | <input type="checkbox"/> Route traces orthogonally if two signal layers are next to each other, whenever possible. Use a separation of 2 to 3 times the trace width. |
| 55. | <input type="checkbox"/> Check I/O termination and impedance matching for chosen I/O standards, especially for voltage-referenced standards. |
| 56. | <input type="checkbox"/> Perform board-level simulation using IBIS models. |
| 57. | <input type="checkbox"/> Configure board trace models for Quartus II advanced I/O timing analysis. |
| 58. | <input type="checkbox"/> Use the Quartus II Pin Planner to make pin assignments. |
| 59. | <input type="checkbox"/> Use Quartus II Fitter messages and reports for sign-off of pin assignments. |
| 60. | <input type="checkbox"/> Verify that Quartus II pin assignments match those in schematic and board layout tools. |
| 61. | <input type="checkbox"/> Use the Create Top-Level Design File command with I/O Assignment Analysis to check I/O assignments before the design is complete. |
| 62. | <input type="checkbox"/> Plan signaling type based on system requirements. |
| 63. | <input type="checkbox"/> Allow the software to assign locations for the negative pin in differential pin pairs. |
| 64. | <input type="checkbox"/> Select suitable signaling type and I/O standard for each I/O pin. |
| 65. | <input type="checkbox"/> Ensure that appropriate I/O standard is supported in targeted I/O bank. |
| 66. | <input type="checkbox"/> Place I/O pins that share voltage levels in the same I/O bank. |
| 67. | <input type="checkbox"/> Verify that all output signals in each I/O bank are intended to drive out at the bank's V_{CCIO} voltage level. |
| 68. | <input type="checkbox"/> Verify that all voltage-referenced signals in each I/O bank are intended to use the bank's V_{REF} voltage level. |
| 69. | <input type="checkbox"/> Check I/O bank support for LVDS and transceiver features. |
| 70. | <input type="checkbox"/> Use caution and follow guidelines for placement of pins located near LVDS I/O. |
| 71. | <input type="checkbox"/> Use the ALTMEMPHY megafunction (or IP core) for Arria II GX devices and the UniPHY megafunction for Arria II GZ devices for each memory interface, and follow connection guidelines/restrictions in the appropriate documentation. |
| 72. | <input type="checkbox"/> Use dedicated DQ/DQS pins and DQ groups for memory interfaces. |
| 73. | <input type="checkbox"/> Make dual-purpose pin settings, and check for any restrictions when using these pins as regular I/O. |
| 74. | <input type="checkbox"/> Check available device I/O features that can help I/O interfaces: current strength, slew rate, I/O delays, open-drain, bus hold, programmable pull-up resistors, PCI clamping diodes, programmable pre-emphasis, and VOD. |
| 75. | <input type="checkbox"/> Consider on-chip termination features to save board space. |
| 76. | <input type="checkbox"/> Check that the required termination scheme is supported for all pin locations. |
| 77. | <input type="checkbox"/> Choose the appropriate mode of DPA, non-DPA, or soft-CDR for high-speed LVDS interfaces. |
| 78. | <input type="checkbox"/> If you want to use DPA, be sure to enable the feature so that the design uses the correct PLLs and follow DPA placement guidelines. |
| 79. | <input type="checkbox"/> Use correct dedicated clock pins and routing signals for clock and global control signals. |
| 80. | <input type="checkbox"/> Use the device PLLs for clock management. |
| 81. | <input type="checkbox"/> Analyze input and output routing connections for each PLL and clock pin. Ensure PLL inputs come from dedicated clock pins or from another PLL. |
| 82. | <input type="checkbox"/> Enable PLL features and check settings in the MegaWizard Plug-In Manager. |
| 83. | <input type="checkbox"/> Ensure you select the correct PLL feedback compensation mode. |

- | Done | N/A |
|------|---|
| 84. | <input type="checkbox"/> Use the clock control block for clock selection and power-down. |
| 85. | <input type="checkbox"/> Analyze design for possible simultaneous switching noise problems. |
| 86. | <input type="checkbox"/> Reduce number of pins that switch voltage at exactly the same time whenever possible. |
| 87. | <input type="checkbox"/> Use differential I/O standards and lower-voltage standards for high switching I/Os. |
| 88. | <input type="checkbox"/> Use lower drive strengths for high switching I/Os. The default drive strength setting might be higher than your design requires. |
| 89. | <input type="checkbox"/> Reduce number of simultaneously switching output pins within each bank. Spread output pins across multiple banks if possible. |
| 90. | <input type="checkbox"/> Spread switching I/Os evenly throughout the bank to reduce the number of aggressors in a given area to reduce SSN (when bank usage is substantially below 100%). |
| 91. | <input type="checkbox"/> Separate simultaneously switching pins from input pins that are susceptible to SSN. |
| 92. | <input type="checkbox"/> Place important clock and asynchronous control signals near ground signals and away from large switching buses. |
| 93. | <input type="checkbox"/> Avoid using I/O pins one or two pins away from PLL power supply pins for high-switching or high drive strength pins. |
| 94. | <input type="checkbox"/> Use staggered output delays to shift the output signals through time, or use adjustable slew rate settings. |
| 95. | <input type="checkbox"/> Use synchronous design practices. Pay attention to clock signals. |
| 96. | <input type="checkbox"/> Use the Quartus II Design Assistant to check design reliability. |
| 97. | <input type="checkbox"/> Use megafunctions with the MegaWizard Plug-In Manager. |
| 98. | <input type="checkbox"/> Follow recommended coding styles, especially for inferring device dedicated logic such as memory and DSP blocks. |
| 99. | <input type="checkbox"/> Enable the chip-wide reset to clear all registers if required. |
| 100. | <input type="checkbox"/> Consider resources available for register power-up and control signals. Do not apply both reset and preset signals to a register. |
| 101. | <input type="checkbox"/> Take advantage of SOPC Builder for system and processor designs. |
| 102. | <input type="checkbox"/> Follow recommendations to set up your source code and partition your design for incremental compilation; plan early in the design flow. |
| 103. | <input type="checkbox"/> Perform timing budgeting and resource balancing between partitions to achieve best results, especially in team-based flows. |
| 104. | <input type="checkbox"/> Create a design floorplan for incremental compilation partitions, if required for your design flow. |
| 105. | <input type="checkbox"/> Specify your third-party synthesis tool and use the correct supported version. |
| 106. | <input type="checkbox"/> Review resource utilization and optimization reports after compilation. |
| 107. | <input type="checkbox"/> Review all Quartus II messages, especially any warning or error messages. |
| 108. | <input type="checkbox"/> Ensure timing constraints are complete and accurate, including all clock signals and I/O delays. |
| 109. | <input type="checkbox"/> Review the TimeQuest Timing Analyzer reports after compilation to ensure there are no timing violations. |
| 110. | <input type="checkbox"/> Ensure that the input I/O times are not violated when data is provided to the Arria II device. |
| 111. | <input type="checkbox"/> Turn on Optimize fast-corner timing on the Fitter Settings page in the Settings dialog box. |
| 112. | <input type="checkbox"/> Turn on Enable multicorner timing analysis under Timing Analysis Settings in the Settings dialog box, or use the <code>--multicorner</code> command line option. |

- | Done | N/A |
|------|--|
| 113. | <input type="checkbox"/> Use <code>create_clock</code> , <code>create_generated_clock</code> to specify the frequencies and relationships for all clocks in your design. |
| 114. | <input type="checkbox"/> Use <code>set_input_delay</code> , <code>set_output_delay</code> to specify external device or board timing parameters. |
| 115. | <input type="checkbox"/> Use <code>derive_pll_clocks</code> to create generated clocks for all PLL outputs, according to the settings in the PLL megafunctions. Specify multi-cycle relationships for LVDS transmitters or receiver deserialization factors. |
| 116. | <input type="checkbox"/> Use <code>derive_clock_uncertainty</code> to automatically apply inter-clock, intra-clock, and I/O interface uncertainties. |
| 117. | <input type="checkbox"/> Use <code>check_timing</code> to generate a report for any problem with the design or applied constraints, including missing constraints. |
| 118. | <input type="checkbox"/> Perform Early Timing Estimation if you want timing estimates before running a full compilation. |
| 119. | <input type="checkbox"/> Use Quartus II optimization features to achieve timing closure or improve the resource utilization. |
| 120. | <input type="checkbox"/> Use the Timing and Area Optimization Advisors to suggest optimization settings. |
| 121. | <input type="checkbox"/> Use incremental compilation to preserve performance for unchanged blocks in your design and to reduce compilation times. |
| 122. | <input type="checkbox"/> Set up parallel compilation if you have multiple processors available for compilation. |
| 123. | <input type="checkbox"/> Use the Compilation Time Advisor to suggest settings that reduce compilation time. |
| 124. | <input type="checkbox"/> Specify your third-party simulation tool, and use the correct supported version and simulation models. |
| 125. | <input type="checkbox"/> Specify your third-party formal verification tool and use the correct supported version. |
| 126. | <input type="checkbox"/> If using formal verification, check for support and design limitations. |
| 127. | <input type="checkbox"/> After compilation, analyze power consumption and heat dissipation in the PowerPlay Power Analyzer. |
| 128. | <input type="checkbox"/> Provide accurate typical signal activities, preferably with a gate-level simulation <code>.vcd</code> file, to get accurate power analysis results. |
| 129. | <input type="checkbox"/> Specify correct operating conditions for power analysis. |
| 130. | <input type="checkbox"/> Use recommended design techniques and Quartus II options to optimize your design for power consumption, if required. |
| 131. | <input type="checkbox"/> Use the Power Optimization Advisor to suggest optimization settings. |