

This application note provides a set of design guidelines, recommendations, and a list of factors to consider for designs that use Altera® Stratix® V FPGAs. It is important to follow Altera recommendations throughout the design process for high-density, high-performance Stratix V designs. This document also assists you with planning the FPGA and system early in the design process, which is crucial to successfully meet design requirements. You can use the [“Design Checklist” on page 53](#) to help verify that you have followed each of the guidelines.

This application note does not include all Stratix V device details and features. For more information about Stratix V devices and features, refer to the [Stratix V Device Handbook](#).

The material references the Stratix V device architecture as well as aspects of the Quartus® II software and third-party tools that you might use in your design. The guidelines presented in this document can improve productivity and avoid common design pitfalls. [Table 1](#) shows the stages in the design flow and contains a brief description of the guidelines covered in each stage.

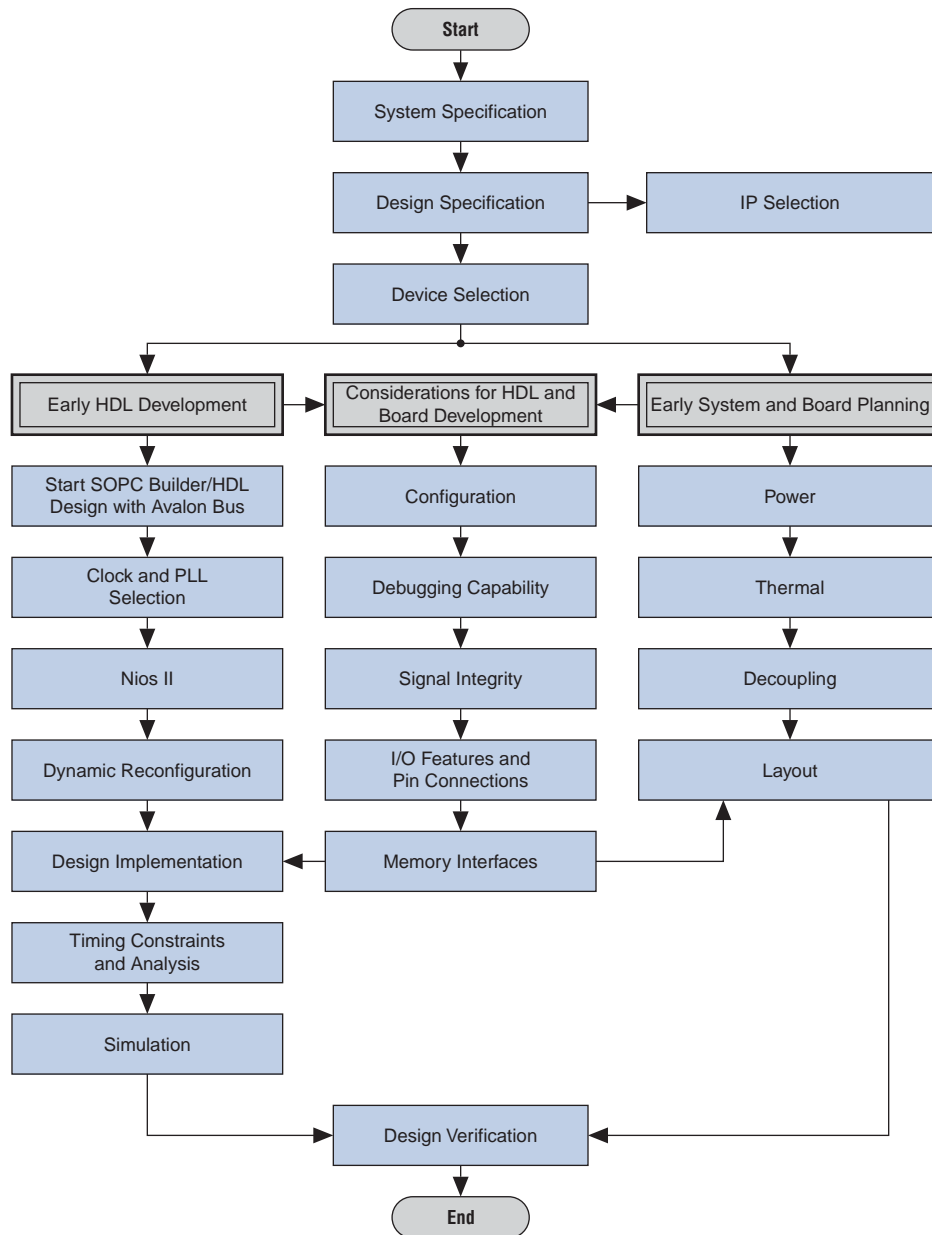
Table 1. Summary of the Design Flow Stage and Guideline Topics

Stages of the Design Flow	Guidelines
“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, HardCopy ASICs, speed grade
“Early System and Board Planning” on page 8	Early power estimation, thermal management option, planning for configuration scheme, planning for on-chip debugging
“Pin Connection Considerations for Board Design” on page 16	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, simultaneous switching noise (SSN)
“Design Entry” on page 37	Coding styles and design recommendations, SOPC Builder, planning for hierarchical or team-based design
“Design Implementation, Analysis, Optimization, and Verification” on page 43	Synthesis tool, device utilization, messages, timing constraints and analysis, area and timing optimization, compilation time, verification, power analysis and optimization

For complete details about the Stratix V device architecture, refer to the [Stratix V Literature](#) page. For the latest known issues related to [Stratix V FPGAs](#), refer to the [Knowledge Database](#).

Figure 1 shows the various stages of the design flow in the order that each stage is typically performed.

Figure 1. Stratix V Device Design Flow



System Specification

In systems that contain a Stratix V 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.

Design Specifications

Create detailed design specifications that define the system before you create your logic design or complete your system design, by performing the following:

- Specify the I/O interfaces for the FPGA
- Identify the different clock domains
- Include a block diagram of basic design functions
- Include intellectual property (IP) blocks

For suggestions about including intellectual property blocks, refer to [“IP Selection”](#). Taking the time to create these specifications improves design efficiency, but this stage is often skipped by FPGA designers.

- Create a functional verification/test plan
 - Consider a common design directory structure
1. Create detailed design specifications and a test plan if appropriate.
 2. Plan clock domains, clock resources, and I/O interfaces early with a block diagram.

Create a functional verification plan to ensure the team knows how to verify the system. Creating a test plan at this stage can also help you design for testability and design for manufacturability. For example, do you want to perform built-in-self test (BIST) functions to drive interfaces? If so, you could use a UART interface with a Nios® II processor inside the FPGA device. You might require the ability to validate all the design interfaces. Refer to [“Planning for On-Chip Debugging”](#) on page 15 for guidelines related to analyzing and debugging the device after it is in the system.

If your design includes multiple designers, it is useful to consider a common design directory structure. This eases the design integration stages. [“Planning for Hierarchical and Team-Based Design”](#) on page 41 provides more suggestions for team-based designs.

IP Selection

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

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

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

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

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


SOPC Builder

The SOPC Builder is a system development tool you can use to create systems based on processors, peripherals, and memories. With the SOPC Builder, you specify the system components in a GUI, and the SOPC Builder generates the interconnect logic automatically. The 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.

The 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 the 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 the source and sink ports.

- Take advantage of the 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 the SOPC Builder to improve your productivity, refer to the [SOPC Builder Literature](#) page on the Altera website.

Device Selection

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

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

-  For more information about the features available in each device density, including logic, memory blocks, multipliers, and phase-locked loops (PLLs), as well as the various package offerings and I/O pin counts, refer to the [Stratix V Device Family Overview](#).

Device Family Variant and High-Speed Transceivers

The Stratix V device family currently contains two variants optimized to meet different application requirements. Both transceiver-based device variants contain full-duplex clock data recovery (CDR)-based transceivers at up to 12.5 Gbps. Stratix V GS devices have transceivers on one side of the device, while Stratix V GX devices have transceivers on two sides.


For information about transceiver board design guidelines, refer to “[Appendix: Stratix V Transceiver Design Guidelines](#)” on page 59.

Logic, Memory, and Multiplier Density

Stratix V devices offer a range of densities that provide different amounts of device logic resources, including memory, multipliers, and adaptive logic module (ALM) logic cells. Determining the required logic density can be a challenging part of the design planning process. Devices with more logic resources can implement larger and potentially more complex designs, but generally have a higher cost. Smaller devices have lower static power utilization. Stratix V 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. Review the resource utilization to find out which device density fits the design. Consider that the coding style, device architecture, and 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 43.

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

 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](#) page in the [Literature](#) section at www.altera.com.

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 design floorplan creation for an incremental or team-based design, as described in “[Planning for Hierarchical and Team-Based Design](#)” on page 41. Consider reserving resources for debugging, as described in “[Planning for On-Chip Debugging](#)” on page 15.

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

Stratix V devices are available in space-saving FineLine BGA packages with various I/O pin counts between 264 and 1,032 user I/O pins. Determine the required number of I/O pins for your application, considering the design’s interface requirements with other system blocks.

Larger densities and package pin counts offer more full-duplex LVDS channels for different signaling; ensure that your device density-package combination includes enough LVDS channels. Other factors can also affect the number of I/O pins required for a design, including simultaneous switching noise (SSN) concerns, pin placement guidelines, pins used as dedicated inputs, I/O standard availability for each I/O bank, differences between I/O standards and speed for row and column I/O banks, and package migration options. For more details about choosing pin locations, refer to [“Pin Connection Considerations for Board Design” on page 16](#), and [“I/O and Clock Planning” on page 25](#).

You can compile any existing designs in the Quartus II software to determine how many I/O pins are used. Also consider reserving I/O pins for debugging, as described in [“Planning for On-Chip Debugging” on page 15](#).

PLLs and Clock Routing

Stratix V devices include up to 24 fractional PLLs (with additional PLLs available from unused transceivers). There are up to 16 global clocks (GCLKs), 92 regional clocks (RCLKs), and 309 periphery clocks (PCLKs). Verify that your chosen device density package combination includes enough PLLs and clock routing resources for your design.

GCLK resources are shared between certain PLLs, which can affect which inputs are available for use. For more details and references regarding clock pins and global routing resources, refer to [“I/O and Clock Planning” on page 25](#).

Speed Grade

The device speed grade affects the device timing performance and timing closure, as well as power utilization. Stratix V devices are available in three speed grades: -2, -3, and -4 (-2 is the fastest). One way to determine which speed grade your design requires is to consider the supported clock rates for specific I/O interfaces.



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



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

Vertical Device Migration

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

- Consider vertical device migration availability and requirements.

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

As described in ["Making FPGA Pin Assignments"](#) on page 26, the Quartus II Pin Planner highlights pins that change function in the migration device when compared to the currently selected device.

HardCopy V ASIC Migration

The HardCopy methodology allows you to seamlessly prototype your system with Stratix V FPGAs, and completely prepares your system for production prior to ASIC design handoff. Altera's HardCopy Design Center uses a proven turnkey process to implement the low-cost, low power, functionally equivalent, pin-compatible HardCopy V ASIC.

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

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

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

- Use complete timing constraints if you want to migrate to a HardCopy device because of the rigorous verification requirements for ASICs.
- RAM cannot be initialized to a known value in the HardCopy ASIC as it can in an FPGA. Therefore, your design must write RAM contents during device operation instead of relying on memory initialization values.

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

Early System and Board Planning

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

- “Early Power Estimation”
- “Temperature Sensing for Thermal Management” on page 9
- “Planning for Device Configuration” on page 10
- “Planning for On-Chip Debugging” on page 15

Early Power Estimation

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

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



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

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

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

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

When the design is complete, the PowerPlay Power Analyzer tool in the Quartus II software provides an accurate estimation of power, ensuring that thermal and supply budgets are not violated. For the most accurate power estimation, use gate-level simulation results with an output file (.vcd) output file from a third-party simulation tool. Refer to “Power Analysis” on page 49.

-  The PowerPlay EPE spreadsheets and user guides for each supported device family are available on the Altera website at:
www.altera.com/support/devices/estimator/pow-powerplay.html
-  For more information about using the 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*.


Temperature Sensing for Thermal Management

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

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

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

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

-  For more information about these features, refer to the *Power Management in Stratix V Devices* chapter in volume 1 of the *Stratix V Device Handbook*.

Planning for Device Configuration

Stratix V devices are based on SRAM cells, so you must download configuration data to the Stratix V device each time the device powers up, because SRAM memory is volatile. Consider whether you require multiple configuration schemes, such as one for debugging 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 16](#).

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

This section includes the following topics:

- [“Configuration Scheme Selection” on page 10](#)
- [“Configuration Features” on page 12](#)
- [“Quartus II Configuration Settings” on page 14](#)



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


Configuration Scheme Selection

You can configure Stratix V devices with one of four configuration schemes:

- Fast passive parallel (FPP)—A controller supplies the configuration data in a parallel manner to the Stratix V FPGA. FPP is supported in an 8-bit (FPP ×8), 16-bit (FPP ×16) or 32-bit data width (FPP ×32).
- Active serial (AS)—The Stratix V FPGA controls the configuration process and gets the configuration data from a serial configuration (EPCS) device or from a qual-serial configuration (EPCS) device. AS is supported in 1-bit (AS ×1) or 4-bit data width (AS ×4).
- Passive serial (PS)—An external host supplies the configuration data serially to the Stratix V FPGA.
- Joint Test Action Group (JTAG)—Configured using the IEEE Standard 1149.1 interface with a download cable, or using a MAX II device or microprocessor with flash memory.

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


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

 For complete information about the Stratix V device 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, Remote System Upgrades with Stratix V Devices* chapter in volume 1 of the *Stratix V Device Handbook*.

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

Serial Configuration Devices


Altera serial configuration devices (EPCS) and quad-serial configuration devices (EPCQ) are used in the AS configuration scheme.

 For information about EPCS and EPCQ configuration devices, refer to volume 2 of the *Configuration Handbook*.


Serial configuration devices can be programmed using a USB-Blaster™, EthernetBlaster, or the ByteBlaster™ II download cable with 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.

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

 For more information about the SRunner software, refer to *AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming* and the source code on the Altera website (www.altera.com).


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

 Programming the EPCS using the SFL solution is slower than using the standard AS interface because it must configure the FPGA before programming EPCS or EPCQ configuration devices.

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

Download Cables

The Quartus II programmer supports configuration of Stratix V devices directly using PS or JTAG interfaces with Altera programming download cables. You can download design changes directly to the device with Altera download cables, making prototyping easy and 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](#)

Using the Parallel Flash Loader Megafunction with MAX II Devices

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

14. If you want to use a flash device with the parallel flash loader, check the list of supported devices.

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

Configuration Features

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

15. Ensure your configuration scheme and board support the following required features: data decompression, design security, remote upgrades, single event updates (SEU) mitigation.

 For more information about these features, refer to the [Configuration, Design Security, Remote System Upgrades with Stratix V Devices](#) chapter in volume 1 of the *Stratix V Device Handbook*.

Data Compression

When you enable data compression, the Quartus II software generates configuration files with compressed configuration data. This compressed file reduces the storage requirements in the configuration device or flash memory, and decreases the time required to transmit the bitstream to the Stratix V device.

Stratix V devices support decompression in the FPP, AS, and PS configuration schemes. Use the Stratix V decompression feature if you use the PS mode to reduce configuration time. The Stratix V decompression feature is not available in the JTAG configuration scheme.

When compression is turned on, the DCLK to DATA ratio changes accordingly based on the FPP configuration scheme selected (FPP $\times 8$, FPP $\times 16$, or FPP $\times 32$). To ensure a successful configuration, the configuration controller must send the DCLK that meets the DCLK to DATA ratio.

- For more information about DCLK to DATA ratio required for your system, refer to the [Configuration, Design Security, Remote System Upgrades with Stratix V Devices](#) chapter in volume 1 of the *Stratix V Device Handbook*.

Design Security Using Configuration Bitstream Encryption

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

The design security feature is available in the FPP, AS, or PS configuration schemes. The design security feature is not available in JTAG configuration scheme.

When the compression is turned on, the DCLK to DATA ratio changes accordingly based on the FPP configuration scheme selected (FPP $\times 8$, FPP $\times 16$, or FPP $\times 32$). To ensure a successful configuration, the configuration controller must send the DCLK that meets the DCLK to DATA ratio.

- For more information about DCLK to DATA ratio required for your system, refer to the [Configuration, Design Security, Remote System Upgrades with Stratix V Devices](#) chapter in volume 1 of the *Stratix V Device Handbook*.

Remote System Upgrades

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

Stratix V devices support remote system upgrades only in the single-device AS configuration scheme with EPCS and EPCQ devices. You can implement remote system upgrades in conjunction with design security and real-time decompression of configuration data. To implement the remote system upgrade interface, use the ALTREMOTE_UPDATE megafunction.

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

SEU Mitigation and CRC Error Checks

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



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

Quartus II Configuration Settings

This section covers several configuration options that you can set in the Quartus II software before compilation to generate configuration or programming files. Your board and system design are affected by these settings and pins, so consider them in the planning stages. Set the options on the **General** category 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. `CLKUSR` also allow you to drive the AS configuration clock (`DCLK`) at 125 MHz maximum. You can enable this feature in the **Configuration** page of the **Device and Pins Option** dialog box.
- **INIT_DONE**—To check if the device has completed initialization and is in user mode, you can monitor the `INIT_DONE` pin. Enable the `INIT_DONE` pin with the **Enable INIT_DONE output** option. During the reset stage, after the device exits POR, and during the beginning of configuration, the `INIT_DONE` pin is tri-stated and pulled high due to an external pull-up resistor. The `INIT_DONE` pin is an open-drain output and requires an external pull-up to V_{CCPGM} .

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

Restart the Configuration After an 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. This enables you to re-initiate the configuration cycle. The `nSTATUS` pin requires an external 10-k Ω pull-up resistor to V_{CCPGM} .

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


Planning for On-Chip Debugging

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

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 the 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 more 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 the 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 IV. In-System Design Debugging* in volume 3 of the *Quartus II Handbook*. The section overview provides more information about choosing a debugging solution.

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

Planning Guidelines for Debugging Tools

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

19. Select on-chip debugging scheme(s) early to plan memory and logic requirements, I/O pin connections, and board connections.
20. If you want to use SignalProbe incremental routing, the SignalTap II Embedded Logic Analyzer, Logic Analyzer Interface, In-System Memory Content Editor, In-System Sources and Probes, or Virtual JTAG megafunction, plan your system and board with JTAG connections that are available for debugging.
21. Plan for the small amount of additional logic resources used to implement the JTAG hub logic for JTAG debugging features.
22. For debugging with the SignalTap II Embedded Logic Analyzer, reserve device memory resources to capture data during system operation.
23. Reserve I/O pins for debugging with SignalProbe or the Logic Analyzer Interface so you do not have to change the design or board to accommodate debugging signals later.
24. Ensure the board supports a debugging mode where debugging signals do not affect system operation.
25. Incorporate a pin header or mictor connector as required for an external logic analyzer or mixed signal oscilloscope.
26. To use debug tools incrementally and reduce compilation time, ensure incremental compilation is on so you do not have to recompile the design to modify the debug tool.
27. To use the Virtual JTAG megafunction for custom debugging applications, instantiate it in the HDL code as part of the design process.
28. To use the In-System Sources and Probes feature, instantiate the megafunction in the HDL code.
29. 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 Stratix V device, various factors can affect the PCB design. This section includes important guidelines for the following topics:

- [“Device Power-Up”](#)
- [“Power Pin Connections and Power Supplies” on page 18](#)
- [“Configuration Pin Connections” on page 20](#)
- [“Board-Related Quartus II Settings” on page 22](#)
- [“Signal Integrity Considerations” on page 23](#)


- “Board-Level Simulation and Advanced I/O Timing Analysis” on page 25
- “I/O and Clock Planning” on page 25 discusses the I/O signal connections for the FPGA, which also affect the board design.

Device Power-Up

Stratix V devices offer hot socketing, which is also known as hot plug-in or hot swap, and power sequencing support without the use of external devices. You can insert or remove a Stratix V device or a board in a system during system operation without causing undesirable effects to the running system bus or the board inserted into the system. The hot socketing feature helps you use Stratix V devices on PCBs that contain a mixture of I/O standards powered by different voltages.

 For more information about the hot socketing capabilities of I/O pins, and for power-on reset circuitry details that must be considered for all Stratix V designs, refer to the *Hot Socketing and Power-On Reset in Stratix V Devices* chapter.

- 30. Design board for power-up: Stratix V output buffers are tri-stated until the device is configured and configuration pins drive out.
- 31. Design voltage power supply ramps to be monotonic.

 The minimum current requirement for the power-on-reset (POR) supplies must be available during device power up.


Stratix V devices do not exit POR if V_{CCIO} and V_{CCPD} are powered by the same regulator and not enough current is available during power up. If the V_{CCIO} , V_{CCPD} , and V_{CCPGM} are powered by the same regulator and not enough current is available during power up, Stratix V devices do not enter POR. For recommendation about combining supplies, refer to the *Stratix V Pin Connection Guidelines*, and for minimum current requirement, use the PowerPlay Early Power Estimator spreadsheet or refer to the Quartus II PowerPlay Power Analyzer report file.

In Stratix V devices, a pin-selectable option (PORSEL) allows you to select between a typical POR time setting of 4 ms or 100 ms. In both cases, you can extend the POR time by using an external component to assert the $nSTATUS$ pin low. Extend POR time if the board cannot meet the maximum power ramp time specifications to ensure the device configures properly and enters user mode.

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


Although power sequencing is not a requirement for correct operation, you should consider the power-up timing of each rail to prevent problems with long-term device reliability when designing a multi-rail powered system.

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



33. Design power sequencing and voltage regulators for best device reliability. Connect the GND between boards before connecting the power supplies.
-  For more information, refer to the *Hot Socketing and Power-On Reset in Stratix V Devices* chapter in the *Stratix V Device Handbook*.

Power Pin Connections and Power Supplies

Stratix V devices require various voltage supplies depending on your design requirements. To verify the core voltage, PLL digital power supply, programmable technology voltage, and other voltage supply levels, refer to the *Stratix V Device Family Pin Connection Guidelines*.

-  For a list of the supply voltages required for Stratix V devices and their recommended operation conditions, refer to the *DC and Switching Characteristics for Stratix V Devices* chapter in volume 3 of the *Stratix V Device Handbook*.

Stratix V devices support a wide range of industry I/O standards, such as V_{CCIO} voltage levels of 3.0, 2.5, 1.8, 1.5, 1.35, 1.25, and 1.2 V.

-  The device output pins do not meet the I/O standard specifications if the V_{CCIO} level is out of the recommended operating range for the I/O standard. The V_{CCPD} pin must be connected to 3.0 V for a 3.0-V V_{CCIO} and 2.5 V for 2.5 V or lower I/O voltages.
-  For a complete list of the supported I/O standards and V_{CCIO} voltages, refer to the *I/O Features in Stratix V Devices* chapter.

Voltage reference (VREF) pins serve as voltage references for certain I/O standards. The VREF pin is used mainly for a voltage bias and does not source or sink much current. The voltage can be created with a regulator or a resistor divider network. For more information about V_{CCIO} voltages and VREF pins for different I/O banks, refer to “*Selectable I/O Standards and Flexible I/O Banks*” on page 29.

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

34. Connect all power pins correctly as specified in the *Stratix V Device Family Pin Connection Guidelines*.
35. Connect V_{CCIO} pins and VREF pins to support each bank's I/O standards.
36. Connect the V_{CCPD} pin to 3.0 V for a 3.0-V V_{CCIO} , and 2.5 V for lower I/O voltages.
37. Explore unique requirements for FPGA power pins or other power pins on your board, and determine which devices on your board can share a power rail.
38. Follow the suggested power supply sharing and isolation guidance, and the specific guidelines for each pin in the *Stratix V Device Family Pin Connection Guidelines*.

Decoupling Capacitors

Board decoupling is important for improving overall power supply integrity while ensuring the rated device performance.

Stratix V devices include embedded on-package and on-die decoupling capacitors to provide high-frequency decoupling. These low-inductance capacitors suppress power noise for excellent power 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 (V_{RM}) to the FPGA power supplies. By using the PDN tool, designers can quickly arrive at an optimized PDN decoupling solution for their specific design.

For each power supply, PDN designers must choose a network of bulk and 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.

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



PLL Board Design Guidelines

For more information about designing your clock and PLL scheme, refer to “[Clock and PLL Selection](#)” on page 33 and “[PLL Feature Guidelines](#)” on page 35. Consider the following checklist items when you design a power system for PLL usage and to minimize jitter, because PLLs contain analog components embedded in a digital device.

- 40. Connect all PLL power pins to reduce noise even if the design does not use all the PLLs. For pin voltage requirements, refer to the [Stratix V Device Family Pin Connection Guidelines](#).
 - 41. Power supply nets should be provided by an isolated power plane, a power plane cut out, or thick trace of at least 20 mils.
-  For more board design guidelines related to PLL power supplies, refer to the “General Board Design Considerations/Guidelines” section of the [Board Design Resource Center](#).

Transceiver Board Design Guidelines


For information about transceiver board design guidelines, refer to “[Appendix: Stratix V Transceiver Design Guidelines](#)” on page 59.


-  For guidelines specific to transceiver design, refer to the [Stratix V Device Handbook, Volume 2](#).
-  For board design guidelines related to high-speed transceivers, refer to the “Gigahertz Channel Design Considerations” section of the [Board Design Resource Center](#).

Configuration Pin Connections

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

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


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

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

DCLK and TCK Signal Integrity

The TCK and/or DCLK traces should produce clean signals with no overshoot, undershoot, or ringing. When designing the board, lay out the TCK and DCLK traces with the same techniques used to lay out a clock line. Any overshoot, undershoot, ringing, or other noise on the TCK signal can affect JTAG configuration. A noisy DCLK signal can affect configuration and cause a 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.

43. Design configuration DCLK and TCK pins to be noise-free.

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

JTAG Pins

Because JTAG configuration takes precedence over all other configuration methods, the JTAG pins should not be left floating or toggling during configuration if you do not use the JTAG interface. If you use the JTAG interface, follow the guidelines in this section.

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

JTAG Pin Connections

A device operating in JTAG mode uses four required pins—TDL, TDO, TMS, and TCK—and one optional pin, TRST. The TCK pin has an internal weak pull-down resistor, while the TDL, TMS, and TRST pins have weak internal pull-up resistors. The JTAG output pin TDO and all JTAG input pins are powered by the 2.5 and 3.0 V_{CCPD}. All JTAG pins are tri-stated during JTAG reconfiguration.

45. 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.

- 46. To disable the JTAG state machine during power-up, pull the TCK pin low through a 1-k Ω resistor to ensure that an unexpected rising edge does not occur on TCK.
- 47. Pull TMS and TDI high through a 1-k to 10-k Ω resistor.
- 48. Connect TRST directly to V_{CCPD} (Connecting the pin low 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.

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

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



For recommendations about connecting a JTAG chain with multiple voltages across the devices in the chain, refer to the *JTAG Boundary-Scan Testing in Stratix V Devices* chapter in volume 1 of the *Stratix V Device Handbook*.

JTAG Signal Buffering

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

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

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

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

MSEL Configuration Mode Pins

Select the configuration scheme by driving the Stratix V device MSEL pins high or low. JTAG configuration is always available, regardless of the MSEL pin selection. The MSEL pins are powered by the V_{CCPGM} power supply of the residing bank. The MSEL[4..0] pins have 5 k- Ω internal pull-down resistors that are always active.

During POR and reconfiguration, the MSEL pins must be at LVTTTL V_{IL} and V_{IH} levels to be considered a logic low and logic high, respectively. To avoid problems with detecting an incorrect configuration scheme, hardwire the MSEL pins to V_{CCPGM} or GND without pull-up or pull-down resistors. Do not drive the MSEL pins with a microprocessor or another device.

52. Connect the MSEL pins to a select configuration scheme; do not leave them floating. For flexibility to change between configuration modes during testing or debugging, set up the board to connect each pin to either V_{CCPGM} or GND without pull-up or pull-down resistors.

Other Configuration Pins

Ensure all dedicated and dual-purpose configuration pins are connected correctly.

53. Connect nIO_PULLUP correctly to set up the 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. Although nIO_PULLUP is powered by V_{CC} , Altera recommends connecting this pin to V_{CCPGM} or GND directly without using a pull-up or pull-down resistor.

54. Hold the nCE (chip enable) pin low during configuration, initialization, and user mode.

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

Board-Related Quartus II Settings

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

Device-Wide Output Enable Pin

Stratix V devices support an optional chip-wide output enable that allows you to override all tri-states on the device I/Os. When this `DEV_OE` pin is driven low, all I/O pins are tri-stated; when this pin is driven high, all 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 in the **General** category of the **Device and Pin Options** dialog box. Ensure this pin is driven to a valid logic level on your board if you enable this pin in the Quartus II software. Do not leave this pin floating.

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

Unused Pins

You can specify the state of unused pins in the Quartus II software to allow flexibility in the board design by choosing one of the five allowable states for **Reserve all unused pins** on the **Unused Pins** category in the **Device and Pin Options** dialog box:

- **As inputs tri-stated**
- **As output driving ground**
- **As outputs driving an unspecified signal**
- **As input tri-stated with bus-hold circuitry**
- **As input tri-stated with weak pull-up**

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

The common setting is to set unused pins **As inputs tri-stated with weak pull-up**. To improve signal integrity, set the unused pins to **As output driving ground**. Doing this reduces inductance by creating a shorter return path and reduces noise on the neighboring I/Os. This approach should not be used if this results in many via paths causing congestion for signals under the device.

To reduce power dissipation, set clock pins and other unused I/O pins **As inputs tri-stated**, and tie them to ground.


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

Signal Integrity Considerations

This section contains references to detailed board design guidelines, as well as a few guidelines related to VREF pins, SSN, and I/O termination.

High-Speed Board Design

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

 For detailed information about signal integrity and board design, refer to the [Board Design Resource Center](#). For example, Altera provides the following application notes that offer information about high-speed board stack-up and signal routing layers:

- [AN 528: PCB Dielectric Material Selection and Fiber Weave Effect on High-Speed Channel Routing](#)
- [AN 529: Via Optimization Techniques for High-Speed Channel Designs](#)
- [AN 530: Optimizing Impedance Discontinuity Caused by Surface Mount Pads for High-Speed Channel Designs](#)

Voltage Reference Pins

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

58. Design VREF pins to be noise-free.

For more information about VREF pins and I/O standards, refer to “[I/O Features and Pin Connections](#)” on page 28.

Simultaneous Switching Noise

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

59. Break out large bus signals on board layers close to the device to reduce cross talk.
60. Route traces orthogonally if two signal layers are next to each other, if possible. Use a separation of 2 to 3 times the trace width.



I/O Termination

Voltage-referenced I/O standards require both an VREF 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 SSTL-2 standards to produce a reliable DDR memory system with superior noise margin.

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

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


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

-  Certain dedicated clock input pairs do not support differential termination.
 -  For a complete list of on-chip termination (OCT) support for each I/O standard, refer to the *I/O Features in Stratix V Devices* chapter.
61. Check I/O termination and impedance matching for chosen I/O standards, especially for voltage-referenced standards.
- For more information about OCT features and limitations, refer to “*I/O Features and Pin Connections*” on page 28.

Board-Level Simulation and Advanced I/O Timing Analysis

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

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

62. Perform board-level simulation using IBIS models (when available).
-  For more information about this simulation flow, refer to the *Signal Integrity with Third-Party Tools* chapter in volume 2 of the *Quartus II Handbook*.

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

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

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

I/O and Clock Planning

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

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

This section details the following topics:

- “Making FPGA Pin Assignments” on page 26
- “Early Pin Planning and I/O Assignment Analysis” on page 27
- “I/O Features and Pin Connections” on page 28
- “Clock and PLL Selection” on page 33
- “PLL Feature Guidelines” on page 35
- “Clock Control Block” on page 36

Making FPGA Pin Assignments

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

64. 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 spreadsheet compatible (.csv) file containing your I/O assignments 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.

65. 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. Pin assignments between the Quartus II software and your schematic and board layout tools must match 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 integrated with certain PCB design EDA tools and can read pin location changes from these tools to check the suggested changes. When you compile your design, the Quartus II software generates the **.pin**. You can use this file to verify that each pin is correctly connected in the board schematics.

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

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

Early Pin Planning and I/O Assignment Analysis

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

You can use the Quartus II Pin Planner for I/O pin assignment planning, assignment, and validation, as described in “*Making FPGA Pin Assignments*” on page 26. The Quartus II **Start I/O Assignment Analysis** command checks that the pin locations and assignments are supported in the target FPGA architecture. Checks include reference voltage pin usage, pin location assignments, and mixing of I/O standards. You can use I/O Assignment Analysis to validate I/O-related assignments that you make or modify throughout the design process.

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

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

Early in the design process, the system architect typically has information about the standard I/O interfaces (such as memory and bus interfaces), IP cores to be used in the design, and any other I/O-related assignments defined by system requirements.

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

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

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



For a list of I/O pin locations and connection guidelines, refer to the [Stratix V Device Family Pin Connection Guidelines](#).

I/O Signaling Type

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

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

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

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

Stratix V 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.

- 68. Plan the I/O signaling type based on the system requirements.
- 69. Allow the software to assign locations for the negative pin in differential pin pairs.

Selectable I/O Standards and Flexible I/O Banks

Stratix V I/O pins are arranged in groups called modular I/O banks. Depending on the device density, the number of I/O banks ranges from 16 to 26 banks, with up to eight I/O banks per side, depending on the device density.

- 70. Select a suitable signaling type and I/O standard for each I/O pin.
- 71. Ensure that the appropriate I/O standard support is supported in the targeted I/O bank.

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

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

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

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


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


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

- 75. Check the I/O bank support for LVDS and transceiver features.



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

 For more information about I/Os, refer to the *I/O Features in Stratix V Devices* chapter in volume 1 of the *Stratix V Device Handbook*.

 For the electrical characteristics of each I/O standard, refer to the *DC and Switching Characteristics for Stratix V Devices* chapter in volume 3 of the *Stratix V Device Handbook*.

Memory Interfaces

Stratix V devices provide an efficient architecture to quickly and easily fit wide external memory interfaces with their small modular I/O banks. Stratix V devices support DDR3, DDR2, DDR SDRAM, QDR II+, QDR II SRAM, and RLDRAM II. The Stratix V FPGA can support DDR external memory on any I/O banks on all sides of the device that do not support transceivers.


76. Use the UniPHY megafunction (or IP core) for each memory interface, and follow connection guidelines/restrictions in the appropriate documentation.

The self-calibrating UniPHY megafunction is optimized to take advantage of the Stratix V I/O structure. The UniPHY megafunction allows you to set external memory interface features and helps set up the physical interface (PHY) best suited for your system. When you use the Altera memory controller MegaCore functions, the UniPHY megafunction is instantiated automatically.

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

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

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

 Refer to the following documents for more information about a specific external memory interface topic:

- For connecting a Stratix V device with external memory devices and pin planning, refer to *Volume 2: Device, Pin, and Board Layout Guidelines* of the *External Memory Interface Handbook*.
- For memory speeds and maximum clock rates, refer to the [External Memory Interface Spec Estimator](#).
- For information about the UniPHY megafunction, refer to the following user guides:
 - [DDR2 and DDR3 SDRAM Controller with UniPHY User Guide](#)
 - [QDR II and QDR II+ SRAM Controller with UniPHY User Guide](#)
 - [RLDRAM II Controller with UniPHY IP User Guide](#)
- For additional resources, refer to the [External Memory Solutions Center](#).

Dual-Purpose and Special Pin Connections

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

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

The device-wide reset and clear pins are available as design I/Os if they are not enabled. For more information, refer to [“Device-Wide Output Enable Pin” on page 23](#) and [“Register Power-Up Levels and Control Signals” on page 39](#).

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

Stratix V I/O Features

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

Table 2. Stratix V I/O Features

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-, 1.25-, 1.35-, 1.5-, 1.8-, 2.5-, or 3.0-V power supply, depending on the output requirements. The output levels are compatible with systems of the same voltage as the power supply. V _{CCPD} power pins must be connected to a 2.5- or 3.0-V power supply.	Refer to the previous sections and the I/O Features in Stratix V Devices chapter in volume 1 of the <i>Stratix V Device Handbook</i> for a summary of MultiVolt I/O support and a list of the supported I/O standards and the typical values for input and output V _{CCIO} , V _{CCPD} , VREF, and board termination voltage (V _{TT}). Altera recommends that you use an external clamp diode on the all I/O pins when the input signal is 3.0 V or 3.3 V.
3.3-V I/O Interface	Stratix V I/O buffers support 3.3-V I/O standards as transmitters or receivers in your system. The output high voltage (V _{OH}), output low voltage (V _{OL}), input high voltage (V _{IH}), and input low voltage (V _{IL}) levels meet the 3.3-V I/O standards specifications when the Stratix V V _{CCIO} voltage is powered by 3.0 V.	To ensure device reliability and proper operation when interfacing with a 3.3-V I/O system, it is important to make sure that the absolute maximum ratings of the Stratix V device are not violated. Altera recommends performing an IBIS or SPICE simulation to determine that the overshoot and undershoot voltages are within the specifications. For more information, refer to the I/O Features in Stratix V Devices chapter in volume 1 of the <i>Stratix V Device Handbook</i> .

Table 2. Stratix V I/O Features

Feature	Usage	Guidelines and More Information
Programmable Output Current Strength	Programmable current-strength control is available for certain I/O standards. You 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 an 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 Stratix V Devices</i> chapter in volume 1 of the <i>Stratix V 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 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 the 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 delay specifications, refer to the <i>DC and Switching Characteristics for Stratix V Devices</i> chapter in volume 3 of the <i>Stratix V 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 an 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 (R_{BH}) of approximately 7 k Ω . With this feature, you do not require an external pull-up or pull-down resistor to hold a signal level when the bus is tri-stated. The circuitry also pulls non-driven pins away from the input threshold voltage where noise can cause unintended high-frequency switching.	If the bus-hold feature is enabled, you cannot use the programmable pull-up option. Disable the bus-hold feature if the I/O pin is configured for differential signals. For the specific sustaining current driven through this resistor and the overdrive current used to identify the next driven input and level for each V_{CCIO} voltage, refer to the <i>DC and Switching Characteristics for Stratix V Devices</i> chapter in volume 3 of the <i>Stratix V Device Handbook</i> .
Programmable Pull-Up Resistor	Pull-up resistor (typically 25 k Ω) weakly holds the I/O to the V_{CCIO} level when in user mode. Can be used with the open-drain output to eliminate the requirement for an external pull-up resistor.	If the programmable pull-up option is enabled, you cannot use the bus-hold feature.

Table 2. Stratix V I/O Features

Feature	Usage	Guidelines and More Information
On-Chip Termination (OCT)	Driver-impedance matching provides the I/O driver with controlled output impedance that closely matches the impedance of the transmission line to significantly reduce reflections. OCT maintains signal quality, saves board space, and reduces external component costs. Support for on-chip series (R_S) with or without calibration, parallel (R_T) with calibration, and dynamic series and parallel termination for single-ended I/O standards and on-chip differential termination (R_D) for differential LVDS I/O standards.	OCT R_S and R_T are supported in the same I/O bank for different I/O standards if they use the same V_{CCIO} supply voltage. Each I/O in an I/O bank can be independently configured to support OCT R_S , programmable current strength, or OCT R_T . You cannot configure both OCT R_S and programmable current strength or slew rate control for the same I/O buffer. Differential OCT R_D is available in all I/O pins. For details about the support and implementation of this feature, refer to the <i>I/O Features in Stratix V Devices</i> chapter in volume 1 of the <i>Stratix V Device Handbook</i> .
Programmable Pre-Emphasis	Increases the amplitude of the high frequency component of the output signal, and thus helps to compensate for the frequency-dependent attenuation along the transmission line.	Refer to the <i>High-Speed Differential I/O Interfaces and DPA in Stratix V Devices</i> chapter in volume 1 of the <i>Stratix V 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.	Refer to the <i>High-Speed Differential I/O Interfaces and DPA in Stratix V Devices</i> chapter in volume 1 of the <i>Stratix V Device Handbook</i> .
Dedicated Differential I/O SERDES Circuitry with DPA and Soft-CDR Support	All the I/Os in Stratix V GX devices and E devices have built-in SERDES circuitry that supports high-speed LVDS interfaces at data rates of up to 1.424 Gbps. DPA circuitry automatically chooses the best phase to compensate the skew between the source synchronous clock and received serial data. The soft-CDR mode provides the opportunity for synchronous/asynchronous applications for chip-to-chip and short reach board-to-board applications for SGMII protocols.	If you want to use DPA, enable the feature in the MegaWizard Plug-In Manager. DPA usage adds some constraints on the placement of high-speed differential channels. Refer to the feature description and placement guidelines in the <i>High-Speed Differential I/O Interfaces and DPA in Stratix V Devices</i> chapter in volume 1 of the <i>Stratix V Device Handbook</i> .

Consider the following checklist items and refer to the appropriate documentation in [Table 2](#) for detailed guidelines:

- 79. 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 V_{OD} .
- 80. Consider OCT features to save board space.
- 81. Verify that the required termination scheme is supported for all pin locations.
- 82. Choose the appropriate mode of DPA, non-DPA, or soft-CDR for high-speed LVDS interfaces.

Clock and PLL Selection

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

Stratix V devices provide dedicated low-skew and high fan-out routing networks. They are organized in a hierarchical structure that provides up to 417 unique clock domains within the device (16 GCLKs + 92 RCLKs + 309 PCLKs). There are up to 28 fractional PLLs per device and up to 18 independently-programmable outputs per PLL. You can use 16 differential dedicated GCLK input pins or 48 to 56 single-ended clock inputs.

83. Use the 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.

84. Use the device fractional PLLs for clock management.

Connect clock inputs to specific PLLs to drive specific low-skew routing networks. Analyze the global resource availability for each PLL and the PLL availability for each clock input pin.

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

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

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



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

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

PLL Feature Guidelines

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

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

Stratix V devices contain fractional PLLs in addition to integer PLLs. You can configure fractional PLLs as integers or as enhanced fractional PLLs. One fractional PLL can use up to 18 output counters and all external clock outputs. Two adjacent fractional PLLs share the 18 output counters.

You can use fractional PLLs to reduce the number of oscillators required on the board, as well as to reduce the clock pins used in the FPGA by synthesizing multiple clock frequencies from a single reference clock source. In addition, you can use fractional PLLs for clock network delay compensation, zero-delay buffering, and transmit clocking for transceivers.

Stratix V device PLLs are feature rich, and support advanced capabilities such as clock feedback modes, switchover, and dynamic phase shifting.



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

Clock Feedback Mode

Stratix V PLLs support up to six different clock feedback modes: source synchronous mode, source synchronous mode for LVDS compensation, direction compensation mode, normal mode, zero-delay buffer (ZDB) mode, and external-feedback mode. Each mode allows clock multiplication and division, phase shifting, and programmable duty cycle. Choose the correct feedback mode for your application.

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

Clock Outputs

You can connect clock outputs to dedicated clock output pins or dedicated clock networks.

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

Clock Control Block

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

- Clock source selection (with dynamic selection for GCLKs)
- GCLK multiplexing
- Clock power down (with static or dynamic clock enable or disable)

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

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

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

I/O Simultaneous Switching Noise

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

90. Analyze the design for possible SSN problems.
91. Reduce the number of pins that switch the voltage level at exactly the same time whenever possible.
92. Use differential I/O standards and lower-voltage standards for high-switching I/Os.
93. Use lower drive strengths for high-switching I/Os. The default drive strength setting might be higher than your design requires.
94. Reduce the number of simultaneously switching output pins within each bank. Spread output pins across multiple banks if possible.
95. 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%).
96. Separate simultaneously switching pins from input pins that are susceptible to SSN.
97. Place important clock and asynchronous control signals near ground signals and away from large switching buses.
98. Avoid using I/O pins one or two pins away from PLL power supply pins for high-switching or high-drive strength pins.
99. Use staggered output delays to shift the output signals through time, or use adjustable slew rate settings.

Refer to “[Stratix V I/O Features](#)” on page 31 for details about the features you can use.

Design Entry

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

Design Recommendations

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

100. 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 19. 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.

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

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



For more information about design recommendations and using the Design Assistant, refer to the [Design Recommendations for Altera Devices and the Quartus II Design Assistant](#) chapter in volume 1 of the *Quartus II Handbook*. You can also refer to industry papers for more information about multiple clock design. For a good analysis, refer to www.sunburstdesign.com.

Using Megafunctions

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

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

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

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

Reconfiguration

Stratix V devices allow you to easily modify your transceivers and FPGA-core while other portions of your design are still running by using dynamic reconfiguration and partial reconfiguration, respectively.

Dynamic Reconfiguration

Stratix V devices allow you to dynamically reconfigure different portions of the transceivers for different protocols, data rates, and PMA settings without powering down any part of the device or interrupting adjacent transceiver channels.

 For more information about dynamic reconfiguration, refer to the [Dynamic Reconfiguration in Stratix V Devices](#) chapter in volume 2 of the *Stratix V Device Handbook*.

Partial Reconfiguration


Partial reconfiguration is suitable for designs with many permutations that do not operate simultaneously. In such systems, one section of the FPGA continues to operate, while the other section is reconfigured for new functionality. Partial reconfiguration improves logic density by removing the need to implement functions that do not operate simultaneously in the FPGA and can potentially reduce the device resource count.

 For more information about partial reconfiguration, refer to the [Increasing Design Functionality with Partial and Dynamic Reconfiguration in 28-nm FPGAs](#) white paper.

Recommended HDL Coding Styles

HDL coding styles can have a significant effect on the quality of results 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.

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

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

Register Power-Up Levels and Control Signals


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

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

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

 For more information about LAB and ALM architecture, refer to the *Logic Array Blocks and Adaptive Logic Modules in Stratix V Devices* chapter in volume 1 of the *Stratix V 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 allows 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: www.sunburst-design.com.

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

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

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

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


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

Registers in the device core always power up to a low (0) logic level in the physical device architecture. If you specify a high power-up level or a non-zero reset value (often called a preset signal), synthesis tools typically use the clear signals available on the registers and perform an optimization referred to as NOT-gate push back.

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


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

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


 For more information about reset logic and power up conditions, refer to the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*.

Planning for Hierarchical and Team-Based Design

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

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

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

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

Planning Design Partitions

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

Follow Altera's recommendations for creating design partitions to improve the overall quality of results. For example, registering partition I/O boundaries keeps critical timing paths inside one partition that can be optimized independently.

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

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

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

Planning in Bottom-Up and Team-Based Flows

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

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


The system architect can plan design partitions at the top level and use the Quartus II software **Generate Bottom-Up Design Partition Scripts** option under the Project menu to automate the process of transferring top-level project information to lower-level modules.

Creating a Design Floorplan

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

108. 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 a third-party synthesis tool in the New Project Wizard or the **EDA Tools Settings** page of the **Settings** dialog box to use the correct Library Mapping File (.lmf) for your synthesis netlist.

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

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

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

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 Software 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 introduce fitting difficulties. If your compilation results in a no-fit error, resource utilization information is important so you can analyze the fitting problems in your design.

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

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

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

- 110. Review resource utilization reports after compilation.

Quartus II Messages

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

- 111. Review all Quartus II messages, especially 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.

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.

- 112. Ensure timing constraints are complete and accurate, including all clock signals and I/O delays.
- 113. Review the TimeQuest Timing Analyzer reports after compilation to ensure there are no timing violations.
- 114. Ensure that the input I/O times are not violated when data is provided to the Stratix V 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 important for large designs such as those in Stratix V devices.


- 115. Turn on **Optimize multi-corner timing** on the **Fitter Settings** page in the **Settings** dialog box.

When the **Optimize multi-corner timing** option is on, the design is optimized to meet its timing requirements at all timing process corners and operating conditions. Therefore, turning on this option helps create a design implementation that is more robust across PVT variations.

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


- 116. Use `create_clock` and `create_generated_clock` to specify the frequencies and relationships for all clocks in your design.
- 117. Use `set_input_delay` and `set_output_delay` to specify the external device or board timing parameters.

- 118. Use `derive_pll_clocks` to create generated clocks for all PLL outputs, according to the settings in the PLL megafunctions. Specify multicycle relationships for LVDS transmitters or receiver deserialization factors.
- 119. Use `derive_clock_uncertainty` to automatically apply inter-clock, intra-clock, and I/O interface uncertainties.
- 120. Use `check_timing` to generate a report on any problem with the design or applied constraints, including missing constraints.

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

Area and Timing Optimization

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

 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.

- 121. 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 optimize for performance and fitting in the **Physical Synthesis Optimizations** page of the **Settings** dialog box. The options in the **Physical Synthesis Optimizations** page typically increase compilation time significantly but can provide significant improvements to the quality of results 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 the 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 50*.

-  For more information, refer to *About Design Space Explorer* in the Quartus II Help.

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 your requirements.

122. Use Quartus II optimization features to achieve timing closure or improve the resource utilization.
123. 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.

124. 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 41.

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

The Quartus II software can run some algorithms in parallel to take advantage of multiple processors and reduce compilation time when more than one processor is available to compile the design. Set the **Parallel compilation** option on the **Compilation Process Settings** page of the **Settings** dialog box, or change the default setting in the **Options** dialog box in the **Processing** page from the Tools menu.

126. 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 quality of results.



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 placement and routing information. You can perform timing simulation for the top-level design at the end of your design flow to ensure that your design works in the targeted device.

Altera provides the ModelSim®-Altera simulator Starter Edition and offers the higher-performance ModelSim-Altera Edition, which enable you to take advantage of advanced testbench capabilities and other features. In addition, the Quartus II EDA Netlist Writer can generate timing netlist files to support other third-party simulation tools such as Synopsys VCS, Cadence NC-Sim, and Aldec Active-HDL. Specify your simulation tool in the **EDA Tools Settings** page of the **Settings** dialog box to generate the appropriate output simulation netlist. The software can also generate scripts to help you setup libraries in your tool with NativeLink integration.

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

If you use a third-party simulation tool, use the software version that is supported with your Quartus II software version. The *Quartus II Software Release Notes* list 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. Using a formal verification flow can impact performance results because it requires that certain logic optimizations be turned off, such as register retiming, and forces hierarchy blocks to be preserved, which can restrict optimization. There are other restrictions that can also limit your design; consult the documentation for details.



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

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 Software 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.

- 128. Specify your third-party formal verification tool and use the correct supported version.
- 129. If you use formal verification, check for support and design limitations.

Power Analysis

Before design completion, estimate power consumption using a spreadsheet as described in “*Early Power Estimation*” on page 8. 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.

- 130. After compilation, analyze power consumption and heat dissipation in the PowerPlay Power Analyzer.
- 131. Provide accurate typical signal activities, preferably with a gate-level simulation **.vcd**, 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 analysis must be representative of the actual operating behavior. For the most accurate power estimation, use gate-level simulation results with a **.vcd** output file from a third-party simulation tool. The simulation activity should include typical input vectors over a realistic time period and not the corner cases often used during functional verification. Use the recommended simulator settings (such as glitch filtering) to ensure good results. Refer to “*Simulation*” for information about third-party simulators.

- 132. Specify the correct operating conditions for power analysis.

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



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

Power Optimization

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

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

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

Device and Design Power Optimization Techniques

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

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



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

Device Speed Grade

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

Clock Power Management

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



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

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

Memory Power Reduction

Reduce the number of memory clocking events to reduce memory power consumption. You can use clock gating described in “[Clock Power Management](#)” or the clock enable signals in the memory ports.

I/O Power Guidelines

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

Non-terminated I/O standards such as 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 a amount smaller than the V_{CCIO} around a bias point; therefore, dynamic power is lower than for non-terminated I/O under similar conditions.

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



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

Quartus II Power Optimization Techniques

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

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



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

Power Optimization Advisor

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

Conclusion

The design guidelines in this application note provide important factors to consider in high-density, high-performance Stratix V designs. It is important to follow Altera's recommendations throughout the design process to achieve good results, avoid common issues, and improve your design productivity. The ["Design Checklist" on page 53](#) summarizes the checklist items presented in this document. You can use this separate checklist to ensure that you have reviewed all the guidelines before completing your Stratix V design.

Document Revision History

[Table 3](#) shows the revision history for this document.

Table 3. Document Revision History

Date	Version	Changes
December 201010.1	1.0	Initial release.

Design Checklist

Use the checklist to verify that you have followed the guidelines for each stage of your design.

- | Done | N/A | |
|------|--------------------------|--|
| 1. | <input type="checkbox"/> | <input type="checkbox"/> “Create detailed design specifications and a test plan if appropriate.” |
| 2. | <input type="checkbox"/> | <input type="checkbox"/> “Plan clock domains, clock resources, and I/O interfaces early with a block diagram.” |
| 3. | <input type="checkbox"/> | <input type="checkbox"/> “Select IP that affects system design, especially I/O interfaces.” |
| 4. | <input type="checkbox"/> | <input type="checkbox"/> “If you plan to use the OpenCore Plus tethered mode for IP, ensure that your board design supports this mode of operation.” |
| 5. | <input type="checkbox"/> | <input type="checkbox"/> “Take advantage of the SOPC Builder for system and processor designs.” |
| 6. | <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.” |
| 7. | <input type="checkbox"/> | <input type="checkbox"/> “Reserve device resources for future development and debugging.” |
| 8. | <input type="checkbox"/> | <input type="checkbox"/> “Consider vertical device migration availability and requirements.” |
| 9. | <input type="checkbox"/> | <input type="checkbox"/> “If you want to migrate to a HardCopy V ASIC, review the appropriate design considerations.” |
| 10. | <input type="checkbox"/> | <input type="checkbox"/> “Estimate power consumption with the PowerPlay Early Power Estimator (EPE) spreadsheet to plan the cooling solution and power supplies before the logic design is complete.” |
| 11. | <input type="checkbox"/> | <input type="checkbox"/> “Set up the temperature sensing diode in your design to measure the device junction temperature for thermal management.” |
| 12. | <input type="checkbox"/> | <input type="checkbox"/> “Select a configuration scheme to plan companion devices and board connections.” |
| 13. | <input type="checkbox"/> | <input type="checkbox"/> “If you want to use the AS configuration mode with large device densities, confirm there is a configuration device available that is large enough for your target FPGA density.” |
| 14. | <input type="checkbox"/> | <input type="checkbox"/> “If you want to use a flash device with the parallel flash loader, check the list of supported devices.” |
| 15. | <input type="checkbox"/> | <input type="checkbox"/> “Ensure your configuration scheme and board support the following required features: data decompression, design security, remote upgrades, single event updates (SEU) mitigation.” |
| 16. | <input type="checkbox"/> | <input type="checkbox"/> “Plan the board design to support optional configuration pins CLKUSR and INIT_DONE, as required.” |
| 17. | <input type="checkbox"/> | <input type="checkbox"/> “Plan board design to use the Auto-restart after configuration error option.” |
| 18. | <input type="checkbox"/> | <input type="checkbox"/> “Take advantage of on-chip debugging features to analyze internal signals and perform advanced debugging techniques.” |
| 19. | <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.” |
| 20. | <input type="checkbox"/> | <input type="checkbox"/> “If you want to use SignalProbe incremental routing, the SignalTap II Embedded Logic Analyzer, Logic Analyzer Interface, In-System Memory Content Editor, In-System Sources and Probes, or Virtual JTAG megafunction, plan your system and board with JTAG connections that are available for debugging.” |

- 21. “Plan for the small amount of additional logic resources used to implement the JTAG hub logic for JTAG debugging features.”
- 22. “For debugging with the SignalTap II Embedded Logic Analyzer, reserve device memory resources to capture data during system operation.”
- 23. “Reserve I/O pins for debugging with SignalProbe or the Logic Analyzer Interface so you do not have to change the design or board to accommodate debugging signals later.”
- 24. “Ensure the board supports a debugging mode where debugging signals do not affect system operation.”
- 25. “Incorporate a pin header or mictor connector as required for an external logic analyzer or mixed signal oscilloscope.”
- 26. “To use debug tools incrementally and reduce compilation time, ensure incremental compilation is on so you do not have to recompile the design to modify the debug tool.”
- 27. “To use the Virtual JTAG megafunction for custom debugging applications, instantiate it in the HDL code as part of the design process.”
- 28. “To use the In-System Sources and Probes feature, instantiate the megafunction in the HDL code.”
- 29. “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.”
- 30. “Design board for power-up: Stratix V output buffers are tri-stated until the device is configured and configuration pins drive out.”
- 31. “Design voltage power supply ramps to be monotonic.”
- 32. “Set POR time to ensure power supplies are stable.”
- 33. “Design power sequencing and voltage regulators for best device reliability. Connect the GND between boards before connecting the power supplies.”
- 34. “Connect all power pins correctly as specified in the Stratix V Device Family Pin Connection Guidelines.”
- 35. “Connect V_{CCIO} pins and VREF pins to support each bank’s I/O standards.”
- 36. “Connect the V_{CCPD} pin to 3.0 V for a 3.0-V V_{CCIO} , and 2.5 V for lower I/O voltages.”
- 37. “Explore unique requirements for FPGA power pins or other power pins on your board, and determine which devices on your board can share a power rail.”
- 38. “Follow the suggested power supply sharing and isolation guidance, and the specific guidelines for each pin in the Stratix V Device Family Pin Connection Guidelines.”
- 39. “Use the PDN tool to plan your power distribution netlist and decoupling capacitors.”
- 40. “Connect all PLL power pins to reduce noise even if the design does not use all the PLLs. For pin voltage requirements, refer to the Stratix V Device Family Pin Connection Guidelines.”
- 41. “Power supply nets should be provided by an isolated power plane, a power plane cut out, or thick trace of at least 20 mils.”

- 42. “Check that all configuration pin connections and pull-up/pull-down resistors are set correctly for your configuration scheme(s).”
- 43. “Design configuration `DCLK` and `TCK` pins to be noise-free.”
- 44. “Connect JTAG pins to a stable voltage level if not in use.”
- 45. “Connect JTAG pins correctly to the download cable header. Ensure the pin order is not reversed.”
- 46. “To disable the JTAG state machine during power-up, pull the `TCK` pin low through a 1-k Ω resistor to ensure that an unexpected rising edge does not occur on `TCK`.”
- 47. “Pull `TMS` and `TDI` high through a 1-k to 10-k Ω resistor.”
- 48. “Connect `TRST` directly to `VCCPD` (Connecting the pin low disables the JTAG circuitry).”
- 49. “Because the download cable interfaces with the JTAG pins of your device, ensure the download cable and JTAG pin voltages are compatible.”
- 50. “Buffer JTAG signals per the recommendations, especially for connectors or if the cable drives more than three devices.”
- 51. “If your device is in a configuration chain, ensure all devices in the chain are connected properly.”
- 52. “Connect the `MSEL` pins to a select configuration scheme; do not leave them floating. For flexibility to change between configuration modes during testing or debugging, set up the board to connect each pin to either `VCCPGM` or `GND` without pull-up or pull-down resistors.”
- 53. “Connect `nIO_PULLUP` correctly to set up the internal pull-up resistors.”
- 54. “Hold the `nCE` (chip enable) pin low during configuration, initialization, and user mode.”
- 55. “Turn on the device-wide output enable option, if required.”
- 56. “Specify the reserved state for unused I/O pins.”
- 57. “Carefully check the pin connections in the Quartus II software-generated `.pin`. Do not connect `RESERVED` pins.”
- 58. “Design `VREF` pins to be noise-free.”
- 59. “Break out large bus signals on board layers close to the device to reduce cross talk.”
- 60. “Route traces orthogonally if two signal layers are next to each other, if possible. Use a separation of 2 to 3 times the trace width.”
- 61. “Check I/O termination and impedance matching for chosen I/O standards, especially for voltage-referenced standards.”
- 62. “Perform board-level simulation using IBIS models (when available).”
- 63. “Configure board trace models for Quartus II advanced I/O timing analysis.”
- 64. “Use the Quartus II Pin Planner to make pin assignments.”
- 65. “Use Quartus II Fitter messages and reports for sign-off of pin assignments.”

- 66. “Verify that the Quartus II pin assignments match those in the schematic and board layout tools.”
- 67. “Use the Create Top-Level Design File command with I/O Assignment Analysis to check the I/O assignments before the design is complete.”
- 68. “Plan the I/O signaling type based on the system requirements.”
- 69. “Allow the software to assign locations for the negative pin in differential pin pairs.”
- 70. “Select a suitable signaling type and I/O standard for each I/O pin.”
- 71. “Ensure that the appropriate I/O standard support is supported in the targeted I/O bank.”
- 72. “Place I/O pins that share voltage levels in the same I/O bank.”
- 73. “Verify that all output signals in each I/O bank are intended to drive out at the bank’s V_{CCIO} voltage level.”
- 74. “Verify that all voltage-referenced signals in each I/O bank are intended to use the bank’s VREF voltage level.”
- 75. “Check the I/O bank support for LVDS and transceiver features.”
- 76. “Use the UniPHY megafunction (or IP core) for each memory interface, and follow connection guidelines/restrictions in the appropriate documentation.”
- 77. “Use dedicated DQ/DQS pins and DQ groups for memory interfaces.”
- 78. “Make dual-purpose pin settings and check for any restrictions when using these pins as regular I/O.”
- 79. “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 V_{OD} .”
- 80. “Consider OCT features to save board space.”
- 81. “Verify that the required termination scheme is supported for all pin locations.”
- 82. “Choose the appropriate mode of DPA, non-DPA, or soft-CDR for high-speed LVDS interfaces.”
- 83. “Use the correct dedicated clock pins and routing signals for clock and global control signals.”
- 84. “Use the device fractional PLLs for clock management.”
- 85. “Analyze input and output routing connections for each PLL and clock pin. Ensure PLL inputs come from the dedicated clock pins or from another PLL.”
- 86. “Enable PLL features and check settings in the MegaWizard Plug-In Manager.”
- 87. “Ensure you select the correct PLL feedback compensation mode.”
- 88. “Check that the PLL offers the required number of clock outputs and use dedicated clock output pins.”
- 89. “Use the clock control block for clock selection and power-down.”
- 90. “Analyze the design for possible SSN problems.”
- 91. “Reduce the number of pins that switch the voltage level at exactly the same time whenever possible.”

-
- 92. “Use differential I/O standards and lower-voltage standards for high-switching I/Os.”
 - 93. “Use lower drive strengths for high-switching I/Os. The default drive strength setting might be higher than your design requires.”
 - 94. “Reduce the number of simultaneously switching output pins within each bank. Spread output pins across multiple banks if possible.”
 - 95. “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%).”
 - 96. “Separate simultaneously switching pins from input pins that are susceptible to SSN.”
 - 97. “Place important clock and asynchronous control signals near ground signals and away from large switching buses.”
 - 98. “Avoid using I/O pins one or two pins away from PLL power supply pins for high-switching or high-drive strength pins.”
 - 99. “Use staggered output delays to shift the output signals through time, or use adjustable slew rate settings.”
 - 100. “Use synchronous design practices. Pay attention to clock signals.”
 - 101. “Use the Quartus II Design Assistant to check design reliability.”
 - 102. “Use megafunctions with the MegaWizard Plug-In Manager.”
 - 103. “Follow recommended coding styles, especially for inferring device dedicated logic such as memory and DSP blocks.”
 - 104. “Enable the chip-wide reset to clear all registers if required.”
 - 105. “Consider resources available for register power-up and control signals. Do not apply both reset and preset signals to a register.”
 - 106. “Follow recommendations to set up your source code and partition your design for incremental compilation; plan early in the design flow.”
 - 107. “Perform timing budgeting and resource balancing between partitions to achieve best results, especially in team-based flows.”
 - 108. “Create a design floorplan for incremental compilation partitions, if required for your design flow.”
 - 109. “Specify your third-party synthesis tool and use the correct supported version.”
 - 110. “Review resource utilization reports after compilation.”
 - 111. “Review all Quartus II messages, especially warning or error messages.”
 - 112. “Ensure timing constraints are complete and accurate, including all clock signals and I/O delays.”
 - 113. “Review the TimeQuest Timing Analyzer reports after compilation to ensure there are no timing violations.”
 - 114. “Ensure that the input I/O times are not violated when data is provided to the Stratix V device.”
 - 115. “Turn on Optimize multi-corner timing on the Fitter Settings page in the Settings dialog box.”

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

Appendix: Stratix V Transceiver Design Guidelines

Quartus II software support for Stratix V transceivers is different from previous transceiver support models because it accounts for how designers use processors to handle data flow. High-speed transceivers are represented in the software by PHY IP cores. The PHY IP cores are missing transceiver voltage, and termination and PLL settings which are now handled by the Quartus II Settings File (.qsf). This section describes the requirements for simulating a transceiver design using the custom PHY IP core, moving to a design, and how to change settings in the .qsf.



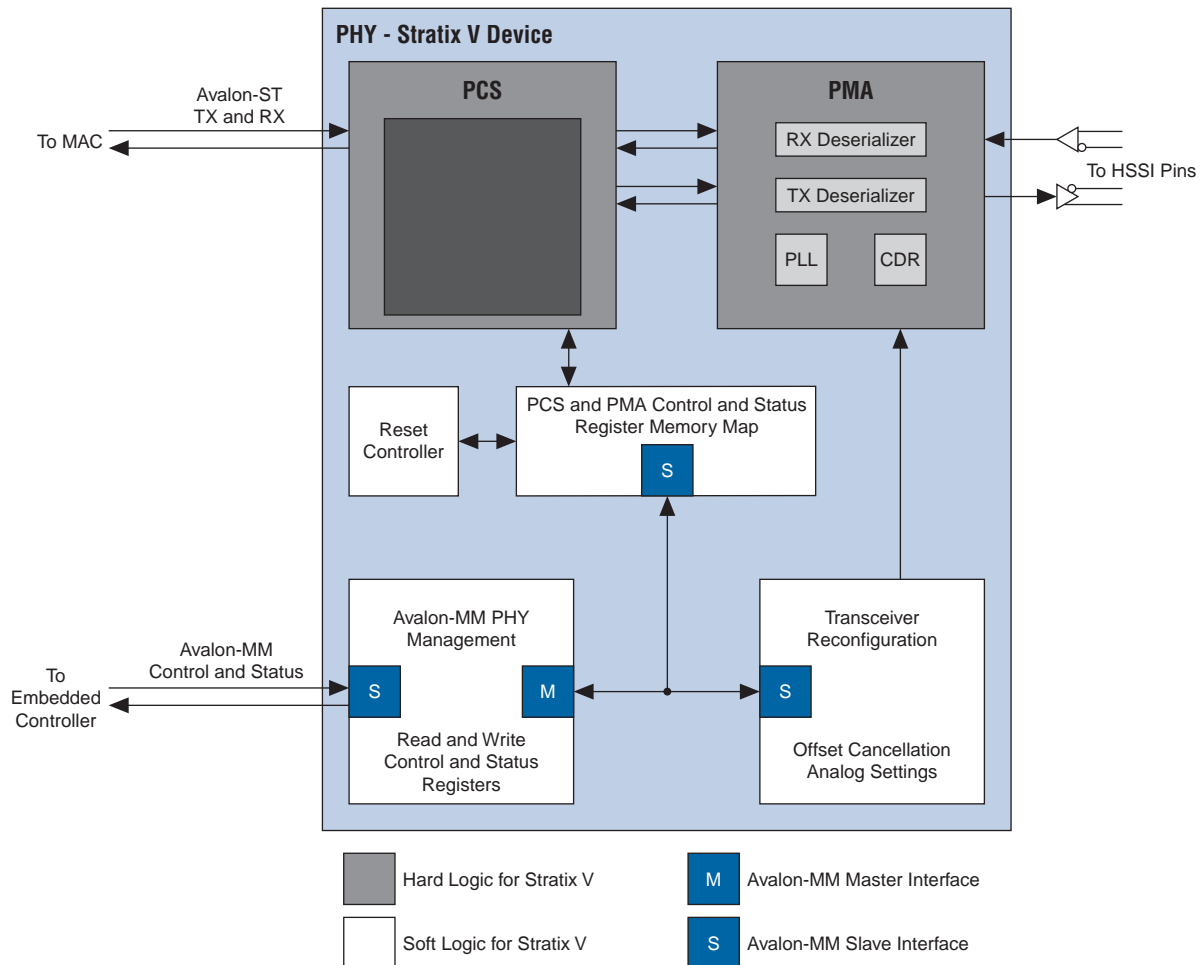
For more information about Stratix V transceivers, refer to the following documents:

- *Altera Transceiver PHY IP Core User Guide*—This document contains information about the transceiver PHY IP options. It also contains a getting started section describing the design flow options: custom Avalon master or Qsys. The Qsys option is not supported for the high-speed transceiver PHY IP in the ADCS version 10.1.
- *Stratix V Device Handbook, Volume 2: Transceiver*—Volume 2 of the *Stratix V Device Handbook* describes how the transceiver features work and the features supported in protocol specific PHY IP.
- *Avalon Interface Specifications*—PHY IP design is modular and uses standard interfaces. All PHY IPs include an Avalon[®] Memory-Mapped (Avalon-MM) interface or conduit interface to access the control and status registers, and an Avalon Streaming (Avalon-ST) interface to connect to the MAC layer design for data transfer. This document discusses the Avalon-MM and Avalon-ST protocols, including timing diagrams.

What is in the PHY IP?

In previous devices containing transceivers, the user had to build the transceiver PMA and PCS blocks, the reconfiguration clock, offset cancellation, status and control block, and the reset timing block for each transceiver group instantiation. In the PHY IP, these all are in the same instantiation (Figure 2).


Figure 2. Altera Modular PHY Design



The control and status registers store device-dependent information about the PCS and PMA modules. You can access the device-dependent information with the device independent Avalon-MM interface, reducing overall complexity of the design and the number of device-dependent signals that you must expose in your top-level module.

The Avalon-MM interface is a synchronous protocol. Each Avalon-MM port is synchronized with an associated clock interface. A design that interfaces with the Avalon-MM can include an embedded controller with an Avalon-MM master interface, a state machine, or μ -controller (simulation model). An Avalon-MM master is required for each PHY IP core instantiation.

The transceiver reconfiguration control and reset control are managed by the Avalon-MM interface. The reset controller manages the `rx_analogreset`, `rx_digitalreset`, `tx_digitalreset`, and `pll_powerdown` signals. The reconfiguration controller controls the writes and reads.

 For information about the addresses of the Avalon-MM registers and the corresponding register name and description, refer to the *Altera Transceiver PHY IP Core User Guide*.

All addressing is in the form of 32-bit registers. Table 5 has a few of the addresses for custom PHY IP. To address these registers, note which registers are for direct addressing (have a `<n>` in the description in the *Altera Transceiver PHY IP Core User Guide*) or indirect addressing.

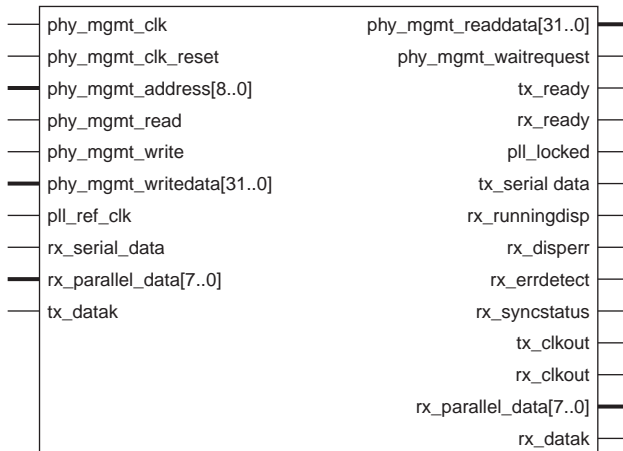
Simulating a Stratix V Design

Create a design. The custom PHY IP in a matching top level with the register map addressing of the Avalon master is handled by the simulator test bench file.

For this example, a custom PHY IP is used with one duplex channel, 8B/10B is enabled, and the interface to the FPGA fabric is 8 bits. The 8B/10B status ports are enabled (Figure 3). The automatic synchronization state machine with the default settings is used with the word aligner, but the status ports are enabled and the pattern length is 10.

Figure 3. Custom PHY

Custom PHY



When the custom PHY IP MegaWizard Plug-In Manager is finished, the software creates two directories in the project directory: `<PHY IP name>` and `<PHY IP name>_sim`.

If you called the IP “customphy,” then the `customphy` directory has the Quartus II synthesis files. The `customphy_sim\altera_xcvr_custom_phy` directory has the simulation files. These directories can be the same files but they do not have to match and they are auto generated by the PHY IP MegaWizard Plug-In Manager.

An example tcl script in the `customphy_sim\altera_xcvr_custom_phy` directory is auto generated to show how to run a transceiver in the ModelSim simulation. Change the name of the tcl script for your project. If you plan to run it from the equivalent directory (`custom_phy_sim\altera_xcvr_custom_phy`), make a change in the tcl script at row 51 to point to your Quartus II directory and your project names. For example, [Table 4](#) shows the required changes for this project:

Table 4. Changes to the tcl Script

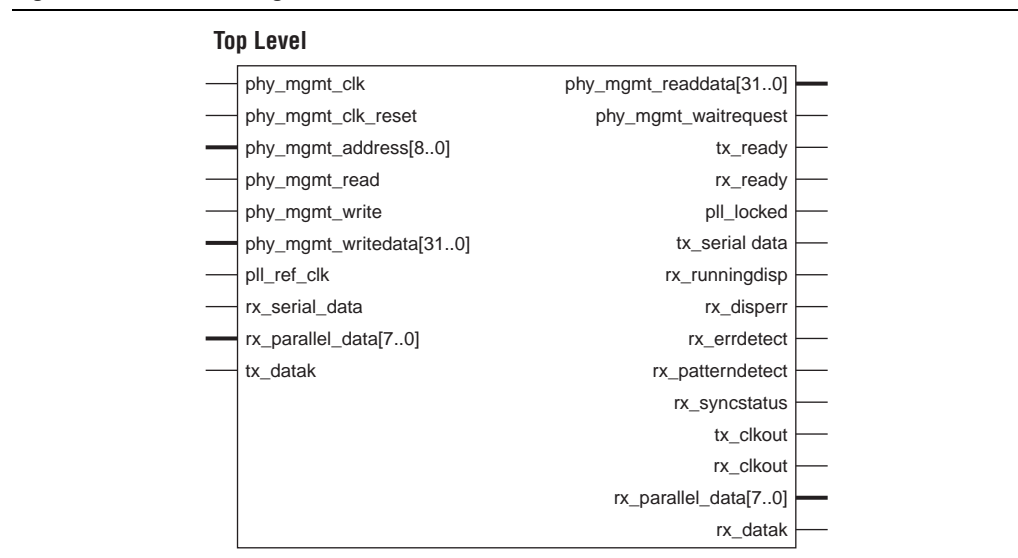
Original	<pre>#set QUARTUS_ROOTDIR \$env(QUARTUS_ROOTDIR) # dut_name = top-level Verilog variant name as generated by Qmegawiz #set tb_name <top level Verilog/VHDL testbench name></pre>
New	<pre>set QUARTUS_ROOTDIR C:/altera/10.1/quartus/ set dut_name CustomPHY set tb_name TopLevel_tb</pre>



The files generated by the PHY IP MegaWizard Plug-In Manager are overwritten the next time the PHY IP is modified in the MegaWizard Plug-In Manager. If a modification is required in any of the files for a custom design, the name of the files should be modified so that they are not overwritten accidentally.

Create a top file design and bring out all the signals to the top level. The top level matches the custom PHY IP signals.

Figure 4. TOP Level Design



Create a test bench (`TopLevel_tb`) with the following:

- A clock generator for the `phy_mgmt_clk`
- A clock generator for the transceiver refclk (`pll_ref_clk`)
- A data generator for the transmit parallel side (plus required align pattern)
- Plan to use a required memory map control and status registers

Working with the Interface

You can run your design, test bench, and tcl script in ModelSim and simulate the transceiver interface. Verified the details for the interface using the *Altera Transceiver PHY IP Core User Guide*. Table 5 lists the registers tested in this exercise.

Table 5. CUSTOM PHY IP CORE REGISTER MAP

Signal	Word Address	Bit
pma_tx_pll_is_locked	h022	—
reset_ch_bitmask	h041	—
reset_control	h042;	—
reset_fine_control	h044;	31-4,0
reset_tx_digital	h044;	1 only
reset_rx_analog	h044;	2 only
Reset_rx_digital	h044;	3 only
phy_serial_loopback	h061;	—
pma_rx_signaldetect	h063;	—
pma_rx_is_lockedtodata	h066;	—
pma_rx_is_lockedtoref	h067;	—
pcs8g_rx_status	h081;	31-6
rx_phase_comp_fifo_error	h081;	0
rx_bitslipboundaryselectout	h081;	5-1
pcs8g_tx_status	h082;	31-1
tx_phase_comp_fifo_error	h082;	0
pcs8g_tx_control	h083;	31-6
tx_invpolarity	h083;	0
tx_bitslipboundaryselect	h083;	5-1
pcs8g_rx_control	h084;	31-1
rx_invpolarity	h084;	0

The following is an example of using the signals in Table 5 for the custom PHY IP with channel 0:

1. Read the pll_is_locked state for all channels (Figure 5):
 - a. Set phy_mgmt_address = 9'h022;
 - b. Send a read pulse
 - c. Read the phy_mgmt_readdata switches to 32'h00000001;

Result: This means channel 0 is locked to the PLL.

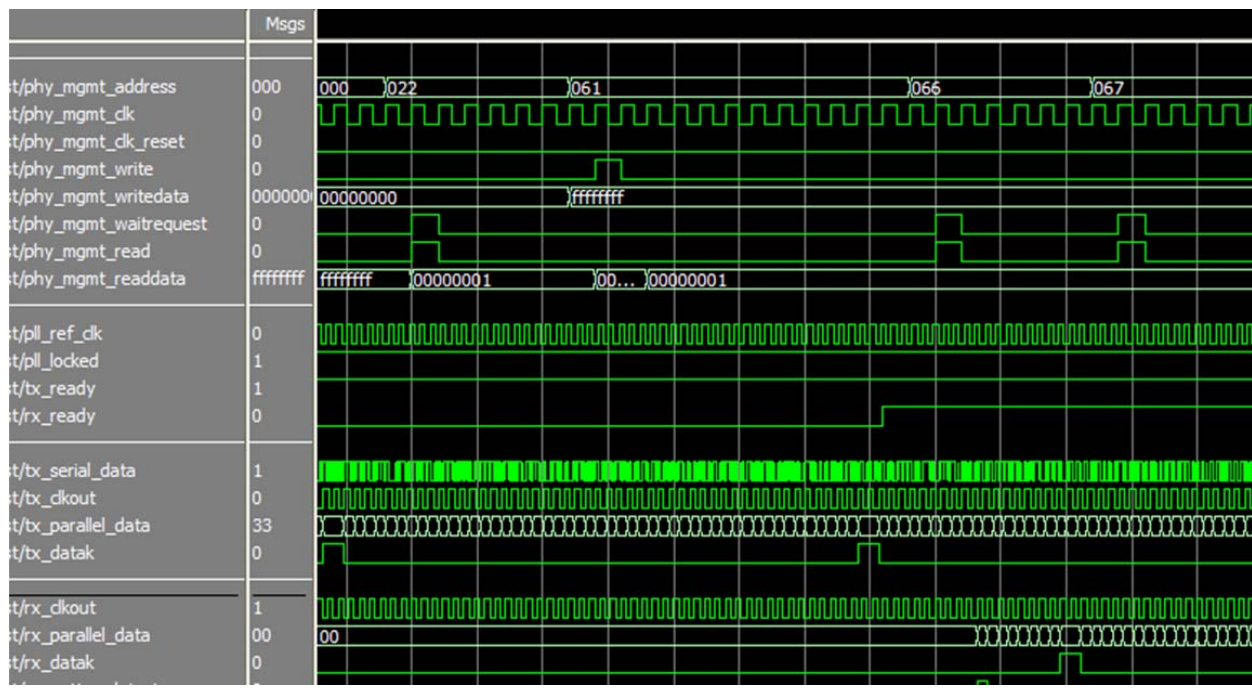
2. Enable serial loopback (Figure 5):
 - a. Set `phy_mgmt_address = 9'h061`;
 - b. Set `phy_mgmt_writedata = 32'hfffffff` //only `32'h00000001` is required
 - c. Send a write pulse

Result: The `rx_parallel` data (2nd from the bottom in Figure 5) starts outputting data some time later after loopback is enabled. This delay represents the receiver functional block delay within the PHY IP until the loopbacked data is accessible in the FPGA fabric.

3. Read `pma_is_lockedtoref` and `pma_is_lockedtodata` (Figure 5):
 - a. Set `phy_mgmt_address = 9'h066`; // `lockedtoref`
 - b. Send a read pulse
 - c. `Phy_mgmt_readdata` is `32'h00000001`; // for channel 0
 - d. Set `phy_mgmt_address = 9'h067`; // `lockedtodata`
 - e. Send a read pulse
 - f. `Phy_mgmt_readdata` is `32'h00000001`; // for channel 0

Result: The read addresses for `pma_rx_is_lockedtoref` (66) and `pma_rx_is_lockedtodata` (67) are both high for channel 0. These signals are not mutually exclusive. If `pma_rx_is_lockedtodata` is high, the CDR is locked to data no matter the state of `pma_rx_is_lockedtoref`.

Figure 5. Direct Read (PLL Locked) and Write (Serial Loopback), rx_is_lockedto (Reference and Data)



4. Set the mask to reset the individual receivers and transmitters (Figure 6):

- Set `phy_mgmt_address = 9h041; // Reset_Chip_Mask`
- Set `phy_mgmt_writedata = 32'h00000001; // all F' to reset all`
- Send a write pulse

Result: This sets the reset chip mask to focus on only channel 0.

5. Reset the channel 0 transmitter (Figure 6):

- Set `phy_mgmt_address = 9'h042; //reset control`
- Set `phy_mgmt_writedata = 32'h00000001; // select the TX reset =bit 0`
- Send a write pulse

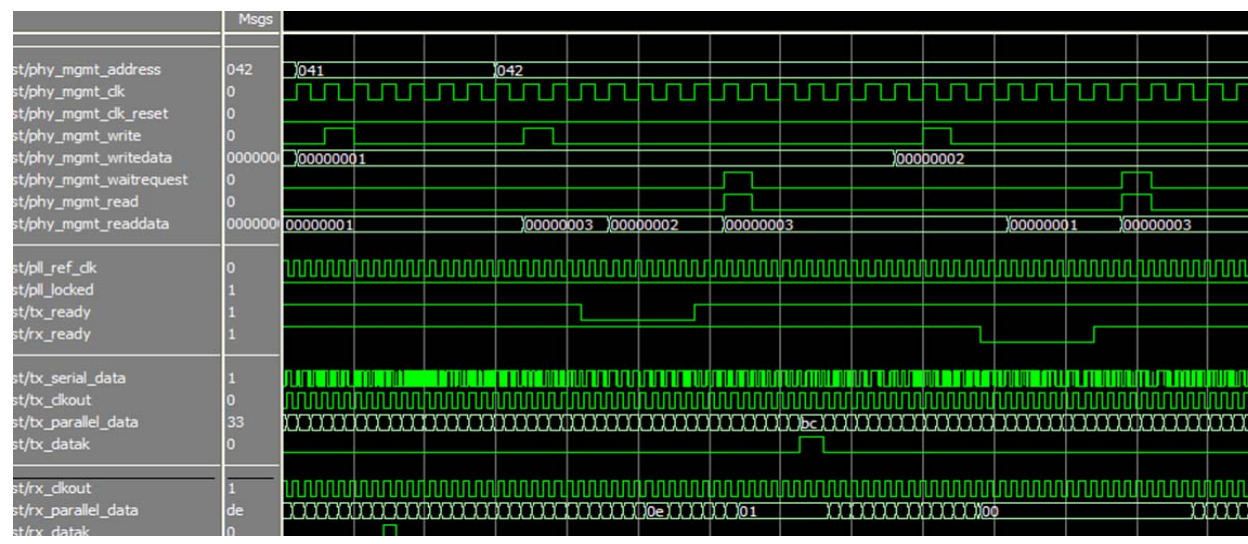
Result: The readdata output for this location has `tx_ready` in bit location 0 and the `rx_ready` in bit location 1. The write pulse `tx_ready` bit goes low and then comes back (data goes from 11 to 10 and back to 11). This information is delayed from the `tx_ready` output transitions. This is the normal latency through the management interface because readdata is a registered output. This is the automatic clear reset so you do not have to write a 0 in the register.

6. Reset the channel 0 receiver (Figure 6):

- Keep `phy_mgmt_address = 9'h042;`
- Set `phy_mgmt_writedata = 32'h00000002; // RX reset = bit 1`
- Send a write pulse

Result: readdata for this location has `tx_ready` in bit 0 and `rx_ready` in bit 1. After the write pulse, `rx_ready` goes low and then comes back (data goes from 11 to 01 and back to 11). This information is delayed from the `rx_ready` output transitions. This is the normal latency through the management interface because readdata is a registered output. This is the automatic clear reset so you do not have to write a 0 in the register.

Figure 6. Reset Control with Automatic Clear



7. Flexible fine reset control (bypasses the reset controller) shown in [Figure 7](#):

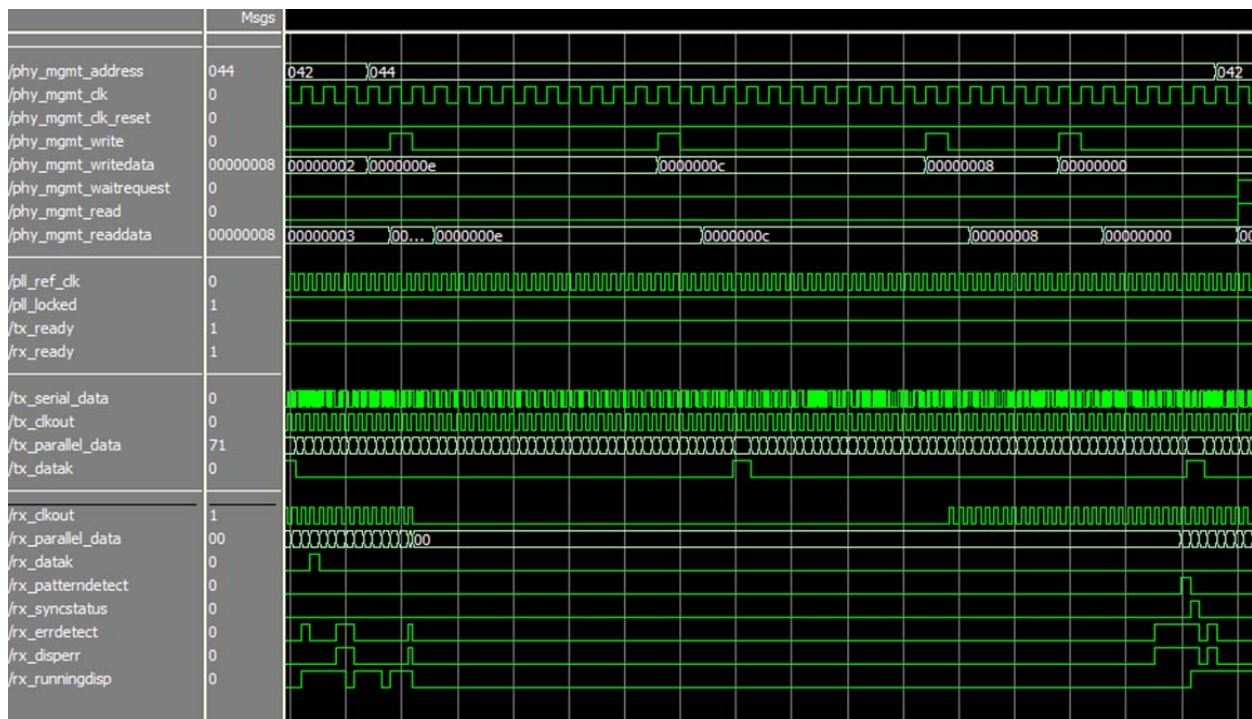
- a. Set `phy_mgmt_address = 9'h044;` // Fine Reset Control
- b. Set `phy_mgmt_writedata = 32'h0000000e;` // 1110 to reset all resets
- c. Send a write pulse



The `tx_ready` and `rx_ready` signals come from the automatic reset controller and can not be used to monitor the readiness using `fine_reset_control`. In the next section, writing a 0 in the reset location turns off the reset after user delay for a user flexible reset sequence.

- d. Custom delay for all reset:
 - Set `phy_mgmt_writedata = 32'h0000000c;` // 1100 releases the TX digital reset bit
- e. Send a write pulse
- f. Custom delay for the RX reset:
 - Set `phy_mgmt_writedata = 32'h00000008;` // 1000 deselects RX analog reset bit
- g. Send a write pulse
- h. Custom delay for the RX digital reset:
 - Set `phy_mgmt_writedata = 32'h00000000;` // deselects the RX digital reset bit

Figure 7. Manual Reset Requires Clear



Compiling a Real Design

After you have simulated the design, work on a real Quartus II design. If you have used Stratix IV transceivers, you may have noticed that many selections in the ALTGX megafunction are not made in the PHY IP. The following section discusses how the selections are moved to the Assignment Editor or `.qsf`.

Set the Quartus II Settings File Using the Assignment Editor and the Settings Dialog Box

Beginning with Stratix V devices, the analog settings (the settings that cannot be observed in digital simulation) are set using `.qsf` assignments applied in the Assignment Editor or in the **Settings** dialog box. This approach avoids modifying the RTL when such a setting requires an update. Specifically, you do not have to re-synthesize the design when analog parameters change.

There are two modes of use for these `.qsf` assignments:

- When a `.qsf` assignment is present, the assigned value is validated by the Quartus II software. Assuming the value is valid, it is applied to a block and becomes a configuration.
- When a `.qsf` assignment does not exist, the Quartus II software computes a valid value for the parameter. This is the default and preferred use model.

To set Termination

Table 6 shows the available setting to set termination with GXB pins.

Table 6. GXB Pin Termination Options in the Assignment Editor

QSF Variable Name	Default Value	Valid Values
GXB_IO_PIN_TERMINATION	100_OHMS	<ul style="list-style-type: none"> ■ 150_OHMS ■ 120_OHMS ■ 100_OHMS ■ 85_OHMS ■ EXTERNAL_RESISTOR
GXB_DEDICATED_REFCLK_PIN_TERMINATION	AC_COUPLING	<ul style="list-style-type: none"> ■ DC_COUPLING_INTERNAL_100_OHMS ■ DC_COUPLING_EXTERNAL_RESISTOR ■ AC_COUPLING

For example, setting the receiver serial data (`rx_serial_data`) channel to have 100 ohms termination in the Assignment Editor writes the following to the `.qsf`:

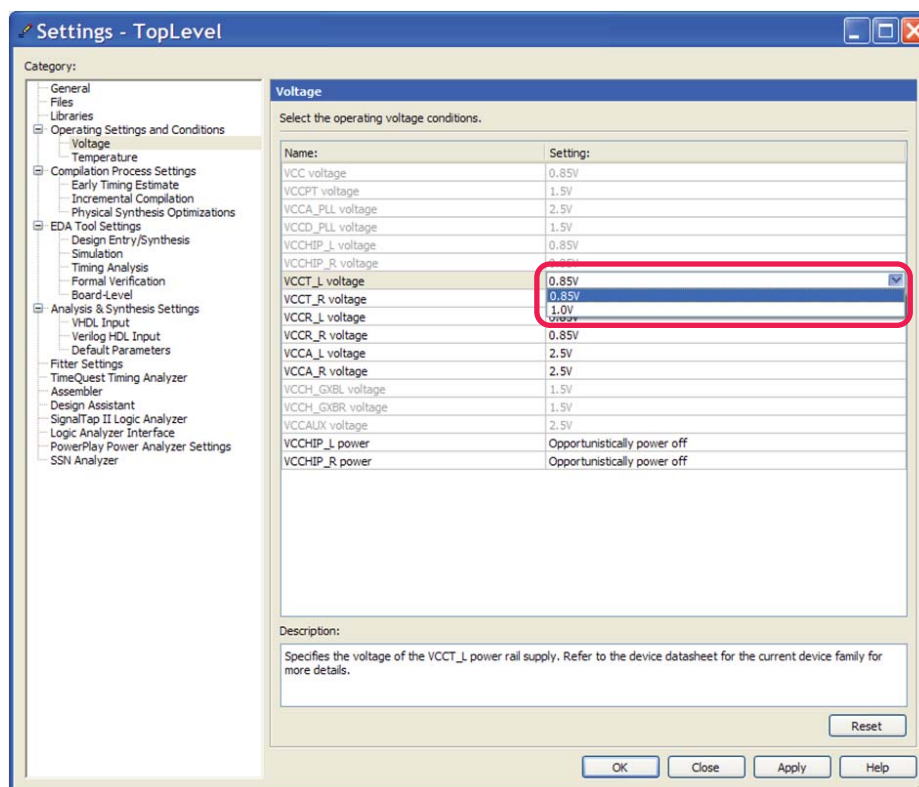
```
set_instance_assignment -name GXB_IO_PIN_TERMINATION 100_OHMS -to rx_serial_data
```

Setting the Transceiver Voltage Levels

To find the default values or settings for transceiver voltage levels in the current project, on the Assignments menu, click **Settings**. In the **Category** list, expand **Operating Settings and Conditions** and select **Voltage**. Use the pull-down menu to select the voltage value you want to change in the **Setting** column. When the values are selected, click **Apply**. The value is correctly written to the **.qsf**. Verify the values you enter match the *Stratix V Pin Connection Guidelines* voltage levels for the transceiver data rate or it may not compile. You can always use the voltage levels for a higher data rate configuration.

For example, `set_global_assignment -name VCCT_L_USER_VOLTAGE 1.0V` is written to the **.qsf** if you select 1.0 V in the **Voltage** page of the **Setting** dialog box (Figure 8).

Figure 8. Selecting 1.0 V in the Voltage Page of the Settings Dialog Box



Setting PLL Assignments

The Quartus II software automatically picks the best internal settings and the best location for the PLL. These settings are available in the Assignment Editor if they are required. Selecting the PLL type as ATX PLL is the most common transceiver PLL option. The default PLL type is CMU PLL. Table 7 shows the .qsf names for the different PLLs.

Table 7. PLL Assignments

.qsf Name	FPLL	CMU PLL	ATX PLL
MERGE_TX_PLL_DRIVEN_BY_REGISTERS_WITH_SAME_CLEAR	✓	✓	✓
PLL_CHANNEL_SPACING	✓	—	—
PLL_COMPENSATION	✓	—	—
PLL_COMPENSATION_MODE	✓	✓	✓
PLL_OUTPUT_CLOCK_FREQUENCY	✓	✓	✓
PLL_PFD_CLOCK_FREQUENCY	✓	✓	✓
PLL_VCO_CLOCK_FREQUENCY	✓	✓	✓
MATCH_PLL_COMPENSATION_CLOCK	✓	—	—
PLL_TYPE	✓	✓	✓

Setting the PLL TYPE to ATX PLL

In the last example directory, an auto generated file called **custom_phy_assignments.qip** has a comment showing the PLL type assignment required for each PLL in the .qsf:

```
#set_instance_assignment -name PLL_TYPE ATX -to
"<top_level|...|<my_name>:<my_name>_inst|sv_xcvr_generic_top:xcvr_custom_phy|sv_xcvr_g
eneric:transceiver_core|pll[*].tx_pll"
```

You can use the Assignment Editor to automatically write the comment in the .qsf. Search for the *.tx_pll in the Node Finder for the list of tx_pll signals used in this design.

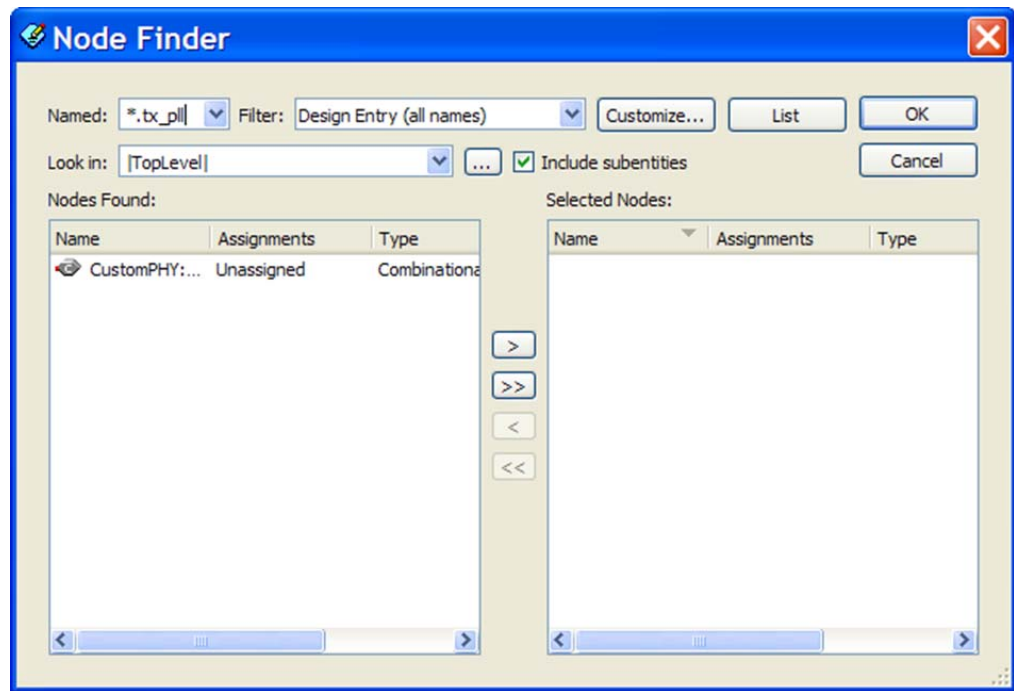
In the previous example, when using the Node Finder for design entry and searching for *.tx_pll in the search field, the single TX PLL location was as follows:

```
CustomPHY:CustomPHY_Inst|altera_xcvr_custom_phy:customphy_inst|sv_xcvr_custom_phy:S5|s
v_xcvr_generic_top:xcvr|sv_xcvr_generic:transceiver_core|pll[0].tx_pll
```

To automatically write the ATX PLL assignment for the PLL location, use the Node Finder and Assignment Editor.

First, locate the `tx_pll`s in the Node Finder as shown in [Figure 9](#).

Figure 9. Using the Node Finder to Locate the `tx_pll`s



Use the Assignment Editor to assign the name and value to the node, as shown in [Figure 10](#).

Figure 10. Using the Assignment Editor to Assign a `tx_pll` as a ATX PLL

To	Assignment Name	Value	Enabled	Entity
rx_dataout	Virtual Pin	On	Yes	TopLevel
rx_clkout	Virtual Pin	On	Yes	TopLevel
rx_ctrlidetect	Virtual Pin	On	Yes	TopLevel
rx_disperr	Virtual Pin	On	Yes	TopLevel
rx_errdetect	Virtual Pin	On	Yes	TopLevel
rx_syncstatus	Virtual Pin	On	Yes	TopLevel
XCVR:XCVR_Inst rx_patterndetect	Virtual Pin	On	Yes	TopLevel
CustomPHY:Custo...customphy_inst	PLL Type	ATX	Yes	TopLevel

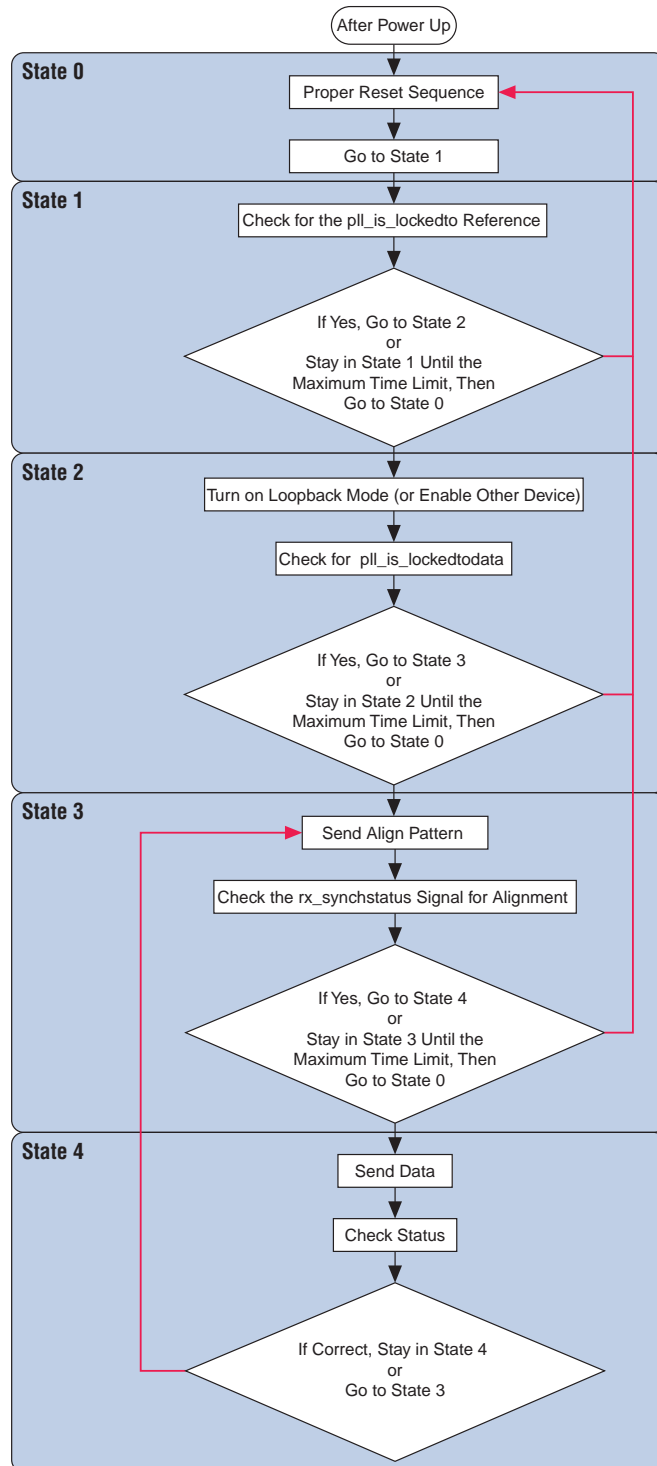
This design is only 1250 Mbps (the default data rate for this configuration), so the ATX PLL cannot be used. The design may fail when it is compiled because the settings for the ATX PLL could not attain the VCO rate of 625 Mbps. The ATX PLL has data rate restrictions and the valid choices should be reviewed in the [DC and Switching Characteristics for Stratix V Devices](#) chapter.

Moving the Test Bench into the Design

In the simulation section, the master for the transceiver block Avalon-MM control and status registers was assigned through the test bench. In a real design, this function is driven by a microprocessor but in a simple design, this function is driven by a state machine.

For the state machine, there should be a reset state 0, and a check for the correct status in each state before moving to the next state (for example, refer to [Figure 11](#)).

Figure 11. Example of Moving Through Each State in a State Machine



The state machine can have many more status checks and can be smarter on what it does if the desired outcome is not achieved to closely match the system or protocol requirements.