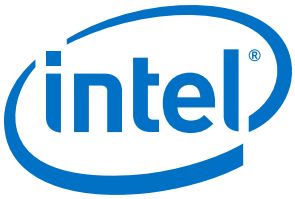


AN 738: Intel® Arria® 10 Device Design Guidelines

AN-738
2017.06.30

Subscribe
Send Feedback

Contents

1 AN 738: Intel® Arria® 10 Device Design Guidelines

This document provides a set of design guidelines, recommendations, and a list of factors to consider for designs that use Intel® Arria® 10 devices. It is important to follow Intel recommendations throughout the design process for high-density, high-performance Arria 10 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. For more information to help verify that you have followed each of the guidelines, use the "Design Checklist" topic in this app note.

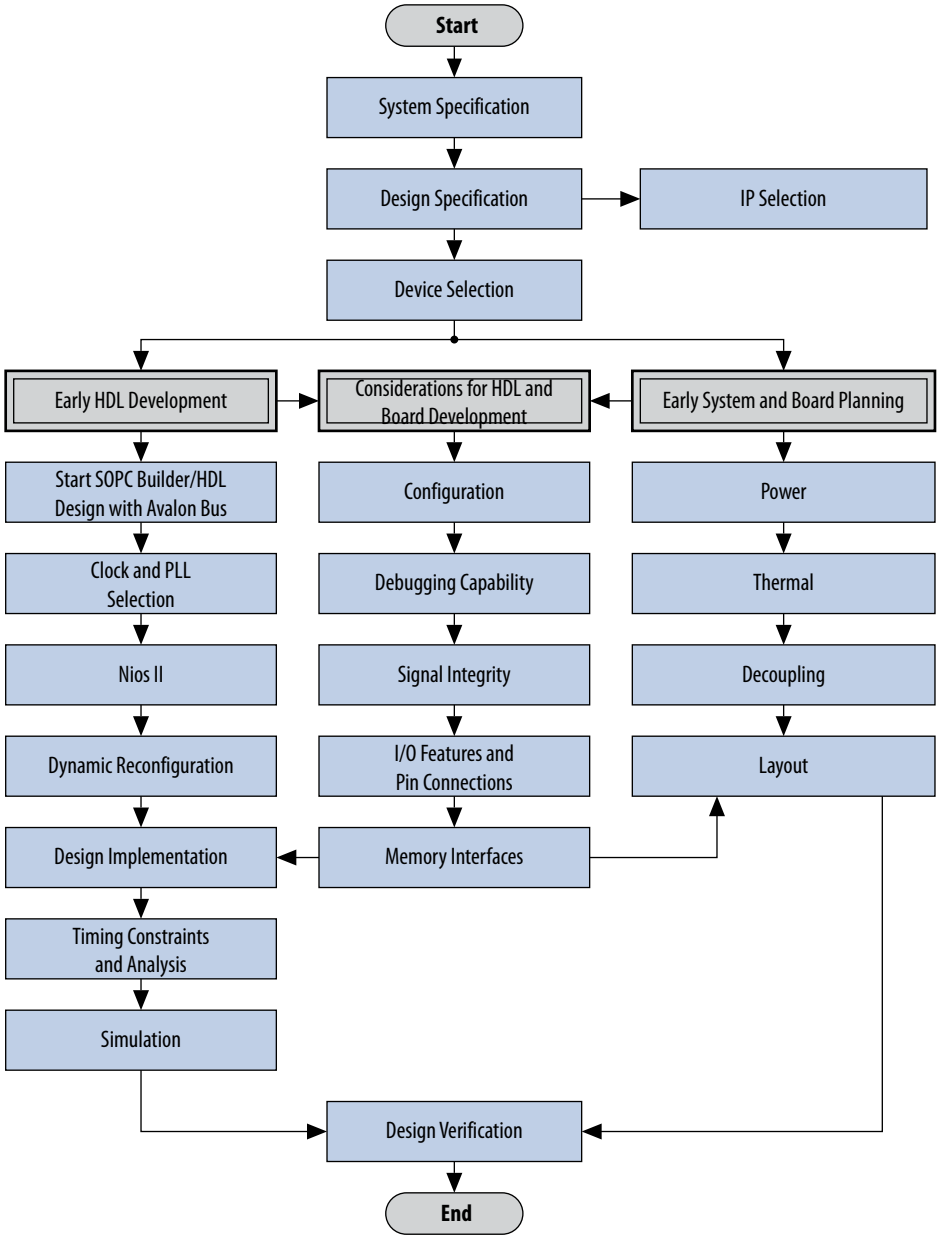
Note: This application note does not include all Arria 10 device details and features. For more information about Arria 10 devices and features, refer to the "Intel Arria 10 Device Design Handbook".

The material references the Arria 10 device architecture as well as aspects of the Quartus® Prime 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. Summary of the Design Flow Stage and Guideline Topics

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

Figure 1. Arria 10 Device Design Flow



Related Links

Arria 10 Core Fabric and General Purpose I/Os Handbook
For more information about Arria 10 devices and features

1.1 System Specification

In systems that contain a Arria 10 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.

1.1.1 Design Specifications

Table 2. Design Specifications Checklist

Number	Done?	Checklist Item
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 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

  Note: 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

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® processor inside the FPGA device. You might require the ability to validate all the design interfaces.

If your design includes multiple designers, it is useful to consider a common design directory structure. This eases the design integration stages.

1.1.2 IP Selection

Table 3. IP Selection Checklist

Number	Done?	Checklist Item
1		Select IP that affects system design, especially I/O interfaces.
2		If you plan to use the OpenCore Plus tethered mode for IP, ensure that your board design supports this mode of operation.

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

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

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

Related Links

https://www.altera.com/support/literature/lit-ip.html
For more information on available IP Cores

1.1.3 Qsys

Table 4. Qsys Checklist

Number	Done?	Checklist Item
1		Take advantage of Qsys for system and processor designs.

Qsys is a system integration tool included as part of the Quartus Prime software. Qsys captures system-level hardware designs at a high level of abstraction and automates the task of defining and integrating customized Hardware Description Language (HDL) components. These components include IP cores, verification IP, and other design modules. Qsys facilitates design reuse by packaging and integrating your custom components with Intel and third-party IP components. Qsys automatically creates interconnect logic from the high-level connectivity you specify, thereby eliminating the error-prone and time-consuming task of writing HDL to specify system-level connections.

Qsys is more powerful if you design your custom components using standard interfaces. By using standard interfaces, your components inter-operate with the components in the Qsys Library. In addition, you can take advantage of bus functional models (BFMs), monitors, and other verification IP to verify your design.

Related Links

Quartus Prime Handbook
For more information about Qsys

1.2 Device Selection

This section describes the first step in the Arria 10 design process—choosing the device family variant, device density, features, package, and speed grade that best suit your design requirements.

Table 5. Device Variant Checklist

Number	Done?	Checklist Item
1		Select a device based on transceivers, I/O pin count, LVDS channels, package offering, logic/ memory/multiplier density, PLLs, clock routing, and speed grade.

Related Links

Arria 10 Device Overview
For more information about the features available in each device density, including logic, memory blocks, multipliers, and phase-locked loops (PLLs)

1.2.1 Device Family Variant and High-Speed Transceivers

The Arria 10 device family currently contains three variants optimized to meet different application requirements.

Table 6. Device Variants and Applications

Device Variant	Transceiver Speed	Applications
GX	12.5 Gbps	For short reach applications and driving 16.0 Gbps backplanes.
GT	17.4 Gbps	For driving 17.4 Gbps backplanes.
	25.8 Gbps	For chip-to-chip and chip-to-module applications, such as interfacing with CFP2 and CFP4 optical modules.
SX SoC	12.5 Gbps	Integrates an ARM*-based HPS and FPGA for short reach applications and driving 16.0 Gbps backplanes.

Related Links

Arria 10 Device Overview
For more information about the device family variants

1.2.2 Logic, Memory, and Multiplier Density

Table 7. Logic, Memory, and Multiplier Density Checklist

Number	Done?	Checklist Item
1		Reserve device resources for future development and debugging.

Arria 10 devices offer a range of densities that provide different amounts of device logic resources, including memory, multipliers, and adaptive logic module (ALM) logic cells. Determining the required logic density can be a challenging part of the design planning process. Devices with more logic resources can implement larger and potentially more complex designs, but generally have a higher cost. Smaller devices have lower static power utilization. Arria 10 devices support vertical migration, which provides flexibility.

Many next-generation designs use a current design as a starting point. If you have other designs that target an Intel 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 Prime software can significantly affect a design's resource utilization and timing performance.

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. Consider reserving resources for debugging.

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

Arria 10 devices are available in space-saving FineLine BGA packages with various I/O pin counts between 288 and 768 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 information on choosing pin locations, refer to "Pin Connection Considerations for Board Design" and "I/O and Clock Planning"

You can compile any existing designs in the Quartus Prime 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".

1.2.4 PLLs and Clock Routing

PLLs provide robust clock management and synthesis for device clock management, external system clock management, and high-speed I/O interfaces.

The Arria 10 device family contains the following PLLs:

- Fractional PLLs—can function as fractional PLLs or integer PLLs
- I/O PLLs—can only function as integer PLLs

The fractional PLLs are located adjacent to the transceiver blocks in the HSSI banks. Each HSSI bank consists of two fractional PLLs. You can configure each fractional PLL independently in conventional integer mode.

In fractional mode, the fractional PLL can operate with third-order delta-sigma modulation. Each fractional PLL has four C counter outputs and one L counter output. The I/O PLLs are located adjacent to the hard memory controllers and LVDS serializer/deserializer (SERDES) blocks in the I/O banks. Each I/O bank consists of one I/O PLL. The I/O PLLs can operate in conventional integer mode. Each I/O PLL has nine C counter outputs.

Arria 10 devices have up to 32 fractional PLLs and 16 I/O PLLs in the largest densities. Arria 10 PLLs have different core analog structure and features support.

For more information about PLLs, refer to "PLLs and Clock Networks".

Related Links

PLLs and Clock Networks
For more information about PLLs

1.2.5 Speed Grade

The device speed grade affects the device timing performance and timing closure, as well as power utilization. Arria 10 GX and SX devices are available in four transceiver speed grades: 1 (fastest), 2, 3, and 4. There are also three core fabric speed grades: 1 (fastest), 2, and 3. Arria 10 GT devices have three transceiver speed grades: 2

(fastest), 3, and 4. There are also three fabric speeds: 1 (fastest), 2, and 3. One way to determine which speed grade your design requires is to consider the supported clock rates for specific I/O interfaces.

You can 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.

Related Links

- External Memory Interfaces in Arria 10 Devices
  For information about supported clock rates for memory interfaces using I/O pins on different sides of the device in different device speed grades
- External Memory Interface Spec Estimator
  For information about comparing the performance of the supported external memory interfaces in Intel FPGA devices

1.2.6 Vertical Device Migration

Table 8. Vertical Device Migration Checklist

Number	Done?	Checklist Item
1		Consider vertical device migration availability and requirements.

Arria 10 devices support vertical migration within the same package, which enables you to migrate to different density devices whose dedicated input pins, configuration pins, and power pins are the same for a given package. This feature allows future upgrades or changes to your design without any changes to the board layout, because you can replace the FPGA on the board with a different density Arria 10 device.

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 Prime software at the beginning of your design cycle or as soon as the device migration selection is possible in the Quartus Prime 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.

The Quartus Prime Pin Planner highlights pins that change function in the migration device when compared to the currently selected device.

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

1.3.1 Early Power Estimation

Table 9. Early Power Estimation Checklist

Number	Done?	Checklist Item
1		Estimate power consumption with the Early Power Estimator (EPE) spreadsheet to plan the cooling solution and power supplies before the logic design is complete.

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.

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 Intel EPE tool 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 tool to perform thermal analysis, including calculation of device junction temperatures derived from the ambient temperature and device power consumption. 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 EPE tool 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 Early Power Estimator File command in the Quartus Prime software to provide input to the spreadsheet.

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

When the design is complete, the Power Analyzer tool in the Quartus Prime software provides more 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.

Note: To obtain the EPE tool, contact your local sales representative.

Related Links

- Early Power Estimator User Guide
  For more information about using the EPE spreadsheet
- Quartus Prime Handbook Volume 3: Verification
  For more information about power estimation and analysis refer to the Power Analysis chapter

1.3.2 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 ($\square_{JA}$) or junction to case ($\square_{JC}$) thermal resistance, and the device power consumption. Arria 10 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.

Table 10. Temperature Sensing Checklist

Number	Done?	Checklist Item
1		Set up the temperature sensing diode in your design to measure the device junction temperature for thermal management.

The Arria 10 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 Arria 10 Devices" chapter in volume 1 of the *Arria 10 Core Fabric and General Purpose I/O Handbook*.

Related Links

Arria 10 Core Fabric and General Purpose I/Os Handbook

1.3.3 Voltage Sensor

Arria 10 devices have an on chip voltage sensor. The sensor provides a 12-bit digital representation of the analog signal being observed. This feature can be used for live monitoring of critical on-chip power supplies and external analog voltage.

Related Links

Arria 10 Core Fabric and General Purpose I/Os Handbook
For more information on voltage sensor feature, refer to Refer to "Power Management in Arria 10 Devices" chapter in the handbook.

1.3.4 Planning for Device Configuration

Arria 10 devices are based on SRAM cells, so you must download configuration data to the Arria 10 device each time the device powers up, because SRAM 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.

In addition, Arria 10 devices offer advanced configuration features, depending on your configuration scheme. Arria 10 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 Prime software), so you have all the information required for your board and system design.

Related Links

- Arria 10 Core Fabric and General Purpose I/Os Handbook
  For information on board design guidelines related to configuration pins and connecting devices for configuration
- Configuration, Design Security, Remote System Upgrades in Arria 10 Devices
  For more information about configuration.
- Configuration Center
  For more information

1.3.4.1 Configuration Scheme Selection

You can configure Arria 10 devices with one of four configuration schemes:

- Fast passive parallel (FPP)—A controller supplies the configuration data in a parallel manner to the Arria 10 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 Arria 10 FPGA controls the configuration process and gets the configuration data from a qual-serial configuration (EPCQ-L) 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 Arria 10 FPGA.
- Joint Test Action Group (JTAG)—Configured using the IEEE Standard 1149.1 interface with a download cable, or using MAX (MAX II, MAX V, MAX 10) devices, or microprocessor with flash memory.

You can enable any specific configuration scheme by driving the Arria 10 device MSEL pins to specific values on the board.

Table 11. Configuration Scheme Selection Checklist

Number	Done?	Checklist Item
1		Select a configuration scheme to plan companion devices and board connections.

All configuration schemes use a configuration device, a download cable, or an external controller (for example, MAX® (MAX II, MAX V, MAX 10) devices or microprocessor).

Related Links

Configuration, Design Security, Remote System Upgrades in Arria 10 Devices
For more information about the Arria 10 device supported configuration schemes, how to execute the required configuration schemes, and all of the necessary option pin settings, including the MSEL pin settings

1.3.4.1.1 Serial Configuration Devices

Intel quad-serial configuration devices (EPCQ-L) are used in the AS configuration scheme.

Serial configuration devices can be programmed using a Intel FPGA Download Cable II or Intel FPGA Ethernet Cable II download cable with the Quartus Prime software through the active serial interface.

Alternatively, you can use the Intel 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.

Table 12. Serial Configuration Devices Checklist

Number	Done?	Checklist Item
1		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.

Serial configuration devices do not directly support the JTAG interface; however, you can program the device with JTAG download cables using the Arria 10 FPGA as a bridge between the JTAG interface and the configuration device, allowing both devices to use the same JTAG interface.

Note: Programming the EPCQ-L using the SFL solution is slower than using the standard AS interface because it must configure the FPGA before programming EPCQ-L configuration devices.

Related Links

- EPCQ-L Serial Configuration Devices Datasheet
  For information about EPCQ-L configuration devices
- AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming
  For more information about the SRunner software
- AN 370: Using the Serial FlashLoader with the Quartus II Software
  For more details about the SFL

1.3.4.1.2 Download Cables

The Quartus Prime programmer supports configuration of the Arria 10 devices directly using JTAG interfaces with Intel programming download cables. You can download design changes directly to the device with Intel 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 Signal Tap Embedded Logic Analyzer.

Related Links

Intel FPGA Parallel Port Cable II User Guide
For more information about how to use Intel's download cables

1.3.4.1.3 Using the Parallel Flash Loader Megafunction with MAX II Devices

Number	Done?	Checklist Item
1		If you want to use a flash device with the parallel flash loader, check the list of supported devices.

If your system already contains common flash interface (CFI) flash memory, you can utilize it for Arria 10 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, MAX V and MAX 10 devices. The PFL also provides the logic to control configuration from the flash memory device to the Arria 10 device and supports compression to reduce the size of your configuration data. Both PS and FPP configuration modes are supported using the PFL feature.

Related Links

Parallel Flash Loader IP Core User Guide
For more information about the PFL

1.3.4.2 Configuration Features

This section describes Arria 10 configuration features and how they affect your design process.

Table 13. Configuration Features Checklist

Number	Done?	Checklist Item
1		Ensure your configuration scheme and board support the required features: design security, remote upgrades, single event upset (SEU) mitigation.

Related Links

Configuration, Design Security, Remote System Upgrades in Arria 10 Devices
For more information about the configuration features

1.3.4.2.1 Data Compression

Data compression is always enabled in Arria 10 configuration, the Quartus Prime 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 configuration bitstream to the Arria 10 device.

Arria 10 devices support decompression in the FPP, AS, and PS configuration schemes. Use the Arria 10 decompression feature if you use the PS mode to reduce configuration time. The Arria 10 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 ×8, FPP ×16, or FPP ×32). To ensure a successful configuration, the configuration controller must send the DCLK that meets the DCLK to DATA ratio.

Related Links

Configuration, Design Security, Remote System Upgrades in Arria 10 Devices
For more information about DCLK to DATA ratio required for your system

1.3.4.2.2 Design Security Using Configuration Bitstream Encryption

The design security feature ensures that Arria 10 designs are protected from copying, reverse engineering, and tampering. Arria 10 devices have the ability to decrypt configuration bitstreams using the AES algorithm, an industry standard encryption algorithm that is FIPS-197 certified. Arria 10 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 ×8, FPP ×16, or FPP ×32). To ensure a successful configuration, the configuration controller must send the DCLK that meets the DCLK to DATA ratio.

Related Links

Configuration, Design Security, Remote System Upgrades in Arria 10 Devices
For more information about DCLK to DATA ratio required for your system

1.3.4.2.3 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. Arria 10 devices feature dedicated remote system upgrade circuitry. Soft logic (either the Nios embedded processor or user logic) implemented in a Arria 10 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.

Arria 10 devices support remote system upgrades only in the single-device AS configuration scheme with EPCQ-L 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.

Related Links

Altera Remote Update IP Core User Guide
For more information about the ALTREMOTE_UPDATE megafunction

1.3.4.2.4 SEU Mitigation and CRC Error Checks

Dedicated circuitry is built into Arria 10 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 Arria 10 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.

Related Links

Arria 10 Core Fabric and General Purpose I/Os Handbook
For more information about SEUs refer to the "SEU Mitigation for Arria 10 Devices" chapter in the handbook.

1.3.4.3 Quartus Prime Configuration Settings

This section covers several configuration options that you can set in the Quartus Prime 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 **Device initialization clock source** 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 100 MHz maximum. You can enable this feature in the **General** page of the **Device and Pins Option** dialog box. The CLKUSR pin is also used as the clock for transceiver calibration, and is a mandatory requirement when using transceivers.
- 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}.

Table 14. Optional Configuration Pins Checklist

Number	Done?	Checklist Item
1		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 kΩ pull-up resistor to V_{CCPGM}.

Table 15. Restart the Configuration After an Error Checklist

Number	Done?	Checklist Item
1		Plan board design to use the **Auto-restart after configuration error** option.

1.3.5 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 Prime 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).

1.3.5.1 On-Chip Debugging Tools

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

- Signal Probe 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.
- Signal Tap 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 Signal Tap Embedded Logic Analyzer works best for synchronous interfaces. For debugging asynchronous interfaces, consider using Signal Probe 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 IP core—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 IP core 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.

Related Links

- Altera Virtual JTAG (altera_virtual_jtag) IP Core User Guide
  For more information about these debugging tools

- Intel SOC FPGA Embedded Design Suite User Guide
  Provides more information about ARM* Development Studio 5* (DS-5*) Intel SoC FPGA Edition debugging.

1.3.5.2 Planning Guidelines for Debugging Tools

Table 16. Planning Guidelines for Debugging Tools Checklist

Number	Done?	Checklist Item
1		Select on-chip debugging scheme(s) early to plan memory and logic requirements, I/O pin connections, and board connections.
2		If you want to use Signal Probe incremental routing, the Signal Tap Embedded Logic Analyzer, Logic Analyzer Interface, In-System Memory Content Editor, In-System Sources and Probes, or Virtual JTAG IP core, plan your system and board with JTAG connections that are available for debugging.
3		Plan for the small amount of additional logic resources used to implement the JTAG hub logic for JTAG debugging features.
4		For debugging with the Signal Tap Embedded Logic Analyzer, reserve device memory resources to capture data during system operation.
5		Reserve I/O pins for debugging with Signal Probe or the Logic Analyzer Interface so you do not have to change the design or board to accommodate debugging signals later.
6		Ensure the board supports a debugging mode where debugging signals do not affect system operation.
7		Incorporate a pin header or mictor connector as required for an external logic analyzer or mixed signal oscilloscope.
8		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.
9		To use the Virtual JTAG IP core for custom debugging applications, instantiate it in the HDL code as part of the design process.
10		To use the In-System Sources and Probes feature, instantiate the IP core in the HDL code.
11		To use the In-System Memory Content Editor for RAM or ROM blocks or the LPM_CONSTANT IP core, turn on the **Allow In-System Memory Content Editor** to capture and update content independently of the system clock option for the memory block in the IP catalog.

If you intend to use any of the on-chip debugging tools, plan for the tool(s) when developing the system board, Quartus Prime project, and design.

1.4 Pin Connection Considerations for Board Design

When designing the interfaces to the Arria 10 device, various factors can affect the PCB design.

1.4.1 Device Power-Up

Table 17. Device Power-Up Checklist

Number	Done?	Checklist Item
1		Design board for power-up: All Arria 10 GPIO pins are tri-stated until the device is configured and configuration pins drive out. The transceiver pins are at high impedance before the device periphery could get programmed. And once the periphery is programmed, the termination and V_{cm} are set immediately after transceiver calibration is complete.
2		Design voltage power supply ramps to be monotonic.
3		Set POR time to ensure power supplies are stable.
4		Design power sequencing and voltage regulators for best device reliability. Connect the GND between boards before connecting the power supplies.

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

The Arria 10 device has Power-On Reset Circuitry, which keeps the device in a reset state until the power supply outputs are within the recommended operating range. The device must reach the recommended operating range within the maximum power supply ramp time. If the ramp time is not met, the device I/O pins and programming registers remain tri-stated and device configuration fails. For the Arria 10 device to exit POR, you must power the V_{CCBAT} power supply even if you do not use the volatile key.

In Arria 10 devices, a pin-selectable option (MSEL) 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.

Arria 10 devices have power-up sequencing and power-down sequencing requirements. You should consider the power-up timing and power-down timing for each rail in order to meet the power sequencing requirements.

Intel uses GND as a reference for 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 Intel device.

Related Links

Quartus II Handbook Volume 2: Design Implementation and Optimization

For more information on the power-up and power-down sequences, refer to the "Power Management in Arria 10 Devices" chapter in this handbook.

1.4.2 Power Pin Connections and Power Supplies

Arria 10 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 *Arria 10 Device Datasheet*.

Arria 10 devices support a wide range of industry I/O standards, such as the following V_{CCIO} voltage levels:

- 3.0 V (only on 3.0 V I/O bank)
- 2.5 V (only on 3.0 V I/O bank)
- 1.8 V
- 1.5 V
- 1.35 V
- 1.25 V
- 1.2 V

Note: 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.

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" chapter.

The V_{REFP_ADC} pin is not a power supply pin. It provides the reference voltage for the ADC for the voltage sensor. For better voltage sensor performance, connect V_{REFP_ADC} to an external reference 1.25 V source. Connecting V_{REFP_ADC} to GND actives an on-chip reference source.

Table 18. Power Pin Connections and Power Supplies Checklist

Number	Done?	Checklist Item
1		Connect all power pins correctly as specified in the *Arria 10 GX and SX Device Family Pin Connection Guidelines*.
2		Connect VCCIO pins and VREF pins to support each bank's I/O standards.
3		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.
4		Follow the suggested power supply sharing and isolation guidance, and the specific guidelines for each pin in the *Arria 10 GX and SX Device Family Pin Connection Guidelines*.

Related Links

- Arria 10 Device Datasheet
  For a list of the supply voltages required for Arria 10 devices and their recommended operation conditions
- Arria 10 Core Fabric and General Purpose I/Os Handbook
  For a complete list of the supported I/O standards and VCCIO voltages, refer to the "I/O and High Speed I/O in Arria 10 Devices" chapter in this handbook.

- Arria 10 GX, GT, and SX Device Family Pin Connection Guidelines
  For details about I/O power pin connections

1.4.2.1 Decoupling Capacitors

Table 19. Decoupling Capacitors Checklist

Number	Done?	Checklist Item
1		Use the PDN tool to plan your power distribution netlist and decoupling capacitors.

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

Arria 10 devices include 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.

Intel has created an easy-to-use power distribution network (PDN) design tool that optimizes the board-level PDN graphically. The purpose of the board-level PDN is to distribute power and return currents from the voltage regulating module (VRM) to the FPGA power supplies. By using the PDN tool, you can quickly arrive at an optimized PDN decoupling solution for your 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.

1.4.2.2 PLL and Transceiver Board Design Guidelines

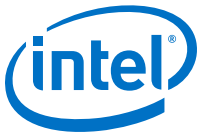
Table 20. PLL Board Design Guidelines Checklist

Number	Done?	Checklist Item
1		Connect all PLL power pins to reduce noise even if the design does not use all the PLLs.
2		Power supply nets should be provided by an isolated power plane, a power plane cut out, or thick trace of at least 20 mils.

Plan your board design when you design a power system for PLL usage and to minimize jitter, because PLLs contain analog components embedded in a digital device.

Related Links

- Board Design Resource Center
  For more board design guidelines related to PLL power supplies
- Arria 10 Transceiver PHY User Guide
  For guidelines specific to transceiver design, refer to the "Arria 10 Transceiver PHY Architecture " chapter in this User Guide.
- Gigahertz Channel Design Considerations
  For board design guidelines related to high-speed transceivers



1.4.3 Configuration Pin Connections

Table 21. Configuration Pin Connections Checklist

Number	Done?	Checklist Item
1		Check that all configuration pin connections and pull-up/pull-down resistors are set correctly for your configuration scheme(s).

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.

Related Links

- Arria 10 GX, GT, and SX Device Family Pin Connection Guidelines
  For specifics about each configuration pin.
- Arria 10 Core Fabric and General Purpose I/Os Handbook
  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 in Arria 10 Devices" chapter in this handbook.

1.4.3.1 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.

Table 22. DCLK and TCK Signal Integrity Checklist

Number	Done?	Checklist Item
1		Design configuration DCLK and TCK pins to be noise-free.

Related Links

Arria 10 Core Fabric and General Purpose I/Os Handbook
For more information about connecting devices in a chain, refer to the "Configuration, Design Security, Remote System Upgrades in Arria 10 Devices" of this handbook.

1.4.3.2 JTAG Pins

Table 23. JTAG Pins Checklist

Number	Done?	Checklist Item
1		Connect JTAG pins to a stable voltage level if not in use.

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.

JTAG Pin Connections

A device operating in JTAG mode uses four required pins—TDI, TDO, TMS, and TCK—and one optional pin, TRST. The TCK pin has an internal weak pull-down resistor, while the TDI, TMS, and TRST pins have weak internal pull-up resistors. The JTAG output pin TDO and all JTAG input pins are powered by the 1.2 V, 1.5 V, and 1.8 V V_{CCPGM}. All JTAG pins are tri-stated during JTAG reconfiguration. Do not drive voltage lower than 1.8 V, 1.5 V, and 1.2-V V_{CCPGM} supply for the TDI, TMS, TCK, and TRST pins. The voltage supplies for TDI, TMS, TCK, and TRST input pins must be the same as set for the V_{CCPGM} supply.

Table 24. JTAG Pin Connections Checklist

Number	Done?	Checklist Item
1		Connect JTAG pins correctly to the download cable header. Ensure the pin order is not reversed.
2		To disable the JTAG state machine during power-up, pull the TCK pin low through a 1 kΩ resistor to ensure that an unexpected rising edge does not occur on TCK.
3		Pull TMS and TDI high through a 1 kΩ to 10 kΩ resistor.
4		Connect TRST directly to V_{CCPGM} (Connecting the pin low disables the JTAG circuitry).

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.

Download Cable Operating Voltage

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

JTAG pins in the Arria 10 device are powered up by V_{CCIO_SDM}. In a JTAG chain containing devices with different V_{CCIO} levels, ensure that the V_{IL} max, V_{IH} min, and the maximum V_I specifications of the device JTAG input pins are not violated. Level shifter might be required between devices to meet the voltage specifications of the devices input pin.

Table 25. Download Cable Operating Voltage Checklist

Number	Done?	Checklist Item
1		Ensure the download cable and JTAG pin voltages are compatible because the download cable interfaces with the JTAG pins of your device.

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. Intel 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. 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.

Table 26. JTAG Signal Buffering Checklist

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

Related Links

Arria 10 Core Fabric and General Purpose I/Os Handbook

For recommendations about connecting a JTAG chain with multiple voltages across the devices in the chain, refer to the "JTAG Boundary-Scan Testing in Arria 10 Devices" chapter in this handbook.

1.4.3.3 MSEL Configuration Mode Pins

Table 27. MSEL Configuration Mode Pins Checklist

Number	Done?	Checklist Item
1		Connect the SDM pins with MSEL function to select the configuration scheme; do not leave them floating. Do not hardwire the pins to V_{CCIO_SDM} or GND if they have other configuration functions based on the configuration scheme selected.

Select the configuration scheme by driving the Arria 10 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 kΩ internal pull-down resistors that are always active.

During POR and reconfiguration, the MSEL pins must be at LVTTL 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.

1.4.3.4 Other Configuration Pins

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

Table 28. Other Configuration Pins Checklist

Number	Done?	Checklist Item
1		Connect `nIO_PULLUP` directly to GND.
2		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.

1.4.4 Board-Related Quartus Prime Settings

The Quartus Prime 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 Prime project is created, and plan for the functionality during board design.

1.4.4.1 Device-Wide Output Enable Pin

Table 29. Device-Wide Output Enable Pin Checklist

Number	Done?	Checklist Item
1		Turn on the device-wide output enable option, if required.

Arria 10 devices support an optional chip-wide output enable that allows you to override all tri-states on the device I/Os. When this DEV_0E 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 Prime 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 Prime software. Do not leave this pin floating.

1.4.4.2 Unused Pins

Table 30. Unused Pins Checklist

Number	Done?	Checklist Item
1		Specify the reserved state for unused I/O pins.
2		Carefully check the pin connections in the Quartus Prime software-generated `.pin` file. Do not connect RESERVED pins.

You can specify the state of unused pins in the Quartus Prime 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**

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.

1.4.5 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.

1.4.5.1 High-Speed Board Design

If your design has high-speed signals, especially with Arria 10 GX/SX device high-speed transceivers, the board design has a major impact on the signal integrity in the system.

Related Links

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

1.4.5.2 Voltage Reference Pins

Table 31. Voltage Reference Pins Checklist

Number	Done?	Checklist Item
1		Design VREF pins to be noise-free.

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

For more information about VREF pins and I/O standards, refer to "I/O Features and Pin Connections".

1.4.5.3 Simultaneous Switching Noise

Table 32. Simultaneous Switching Noise Checklist

Number	Done?	Checklist Item
1		Break out large bus signals on board layers close to the device to reduce cross talk.
2		Route traces orthogonally if two signal layers are next to each other, if possible. Use a separation of two to three times the trace width.

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, plan the board layout according to the board layout recommendations in the PCB guidelines can help with noise reduction.

1.4.5.4 I/O Termination

Voltage-referenced I/O standards require both an V_{REF} and a termination voltage (V_{TT}). The reference voltage of the receiving device tracks the termination voltage of the transmitting device. Each voltage-referenced I/O standard requires a unique termination setup.

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

Arria 10 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. Arria 10 devices provide an optional on-chip differential resistor when using LVDS.

Note:

Table 33. I/O Termination Checklist

Number	Done?	Checklist Item
1		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".

Related Links

I/O and High Speed I/O in Arria 10 Devices
For a complete list of on-chip termination (OCT) support for each I/O standard

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

Table 34. Board-Level Simulation and Advanced I/O Timing Analysis Checklist

Number	Done?	Checklist Item
1		Perform board-level simulation using IBIS models (when available).
2		Configure board trace models for Quartus Prime 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 Prime software, select **IBIS** under **Board-level signal integrity analysis** on the **Board-Level** page in **EDA Tool Settings** of the **Settings** dialog box.

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.

You can configure board trace models of selected I/O standards and generate "board-aware" signal integrity reports with the Quartus Prime 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.

Related Links

Quartus II Handbook Volume 2: Design Implementation and Optimization
For more information about this simulation flow, refer to the "Signal Integrity with Third-Party Tools" chapter in this handbook.

1.5 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 Arria 10 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.

1.5.1 Making FPGA Pin Assignments

Table 35. Making FPGA Pin Assignments Checklist

Number	Done?	Checklist Item
1		Use the Quartus Prime Pin Planner to make pin assignments.
2		Use Quartus Prime Fitter messages and reports for sign-off of pin assignments.
3		Verify that the Quartus Prime pin assignments match those in the schematic and board layout tools.

With the Quartus Prime 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, the Pin Migration view highlights pins that change function in the migration device when compared to the currently selected device.

You have the option of importing a Microsoft Excel spreadsheet into the Quartus Prime 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 Prime 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.

Quartus Prime designers can then pass the pin location information to PCB designers. Pin assignments between the Quartus Prime 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 Prime software generates the `.pin` file. You can use this file to verify that each pin is correctly connected in the board schematics.

Related Links

- Quartus Prime Handbook Volume 2: Design Implementation and Optimization
  For details about using the Pin Planner to make I/O assignments refer to the "Managing Device I/O Pins" chapter in this handbook.
- Quartus Prime Handbook Volume 2: Design Implementation and Optimization
  For more information about passing I/O information between the Quartus Prime software and third-party EDA tools, refer to the "Mentor Graphics PCB Design Tools Support" chapter in this handbook.
- Quartus Prime Handbook Volume 2: Design Implementation and Optimization
  For more information about passing I/O information between the Quartus Prime software and third-party EDA tools, refer to the "Cadence PCB Design Tools Support" chapter in this handbook.

1.5.2 Early Pin Planning and I/O Assignment Analysis

Table 36. Early Pin Planning and I/O Assignment Analysis Checklist

Number	Done?	Checklist Item
1		Use the **Create Top-Level Design File** command with I/O Assignment Analysis to check the I/O assignments before the design is complete.

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 Prime Pin Planner for I/O pin assignment planning, assignment, and validation, as described in "Making FPGA Pin Assignments". The Quartus Prime **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 Intel FPGA using the Quartus Prime Pin Planner before the source code is designed.

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 IP Core feature interfaces with the IP catalog, and enables you to create or import custom 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 Prime software.

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

You can use the Quartus Prime Pin Planner for I/O pin assignment planning, assignment, and validation, as described in "Making FPGA Pin Assignments". The Quartus Prime **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.

Related Links

Quartus Prime Handbook Volume 2: Design Implementation and Optimization
For details about using the Pin Planner to make I/O assignments refer to the "Managing Device I/O Pins" chapter in this handbook.

1.5.3 I/O Features and Pin Connections

Arria 10 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.

Related Links

Arria 10 GX, GT, and SX Device Family Pin Connection Guidelines
For a list of I/O pin locations and connection guidelines.

1.5.3.1 I/O Signaling Type

Table 37. I/O Signaling Type Checklist

Number	Done?	Checklist Item
1		Plan the I/O signaling type based on the system requirements.
2		Allow the software to assign locations for the negative pin in differential pin pairs.

Arria 10 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.

Arria 10 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 Prime software automatically places the corresponding negative pin.

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

Arria 10 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 6 to 16 banks, with up to eight I/O banks per side, depending on the device density.

Table 38. Selectable Standards and Flexible I/O Banks Checklist

Number	Done?	Checklist Item
1		Select a suitable signaling type and I/O standard for each I/O pin. The I/O banks are located in I/O columns. Each I/O bank contains its own PLL, DPA, and SERDES circuitries.
2		Ensure that the appropriate I/O standard support is supported in the targeted I/O bank.
3		Place I/O pins that share voltage levels in the same I/O bank.
4		Verify that all output signals in each I/O bank are intended to drive out at the bank's V_{CCIO} voltage level.
5		Verify that all voltage-referenced signals in each I/O bank are intended to use the bank's V_{REF} voltage level.
6		Check the I/O bank support for LVDS and transceiver features.

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 described in the *Clock and PLL Selection* section .

The board must supply each bank with one V_{CCIO} voltage level for every VCCIO pin in a bank. Each I/O bank is powered by the VCCIO pins of that particular bank, and is independent of the VCCIO pins 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 V_{REF} 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 V_{REF} setting. 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 Arria 10 transceiver banks include additional support. There are two types of I/O banks, LVDS and 3 V.

The LVDS I/O bank supports differential and single-ended I/O standards up to 1.8 V. The LVDS I/O pins form pairs of true differential LVDS channels. Each pair supports a parallel input/output termination between the two pins. You can use each LVDS channel as transmitter or receiver.

The 3 V I/O bank supports only single-ended I/O standards up to 3 V. Each adjacent I/O pair also supports Differential SSTL and Differential HSTL I/O standards. The single-ended output of the 3 V I/O has the same set of features as the single-ended output of the DDR I/O IP, except the programmable pre-emphasis feature.

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

Related Links

- Clock and PLL Selection on page 38
- Arria 10 Core Fabric and General Purpose I/Os Handbook
  For information about the number of channels available for the LVDS I/O standard, refer to the "I/O and High Speed I/O in Arria 10 Devices" chapter of this handbook.
- Arria 10 Transceiver PHY User Guide
  For more information about transceiver-bank-related features, refer to the "Arria 10 Transceiver PHY Architecture " chapter of this User Guide.
- Arria 10 Core Fabric and General Purpose I/Os Handbook
  For more information about I/Os, refer to the "I/O and High Speed I/O in Arria 10 Devices" chapter of this handbook.
- Arria 10 Device Datasheet
  For the electrical characteristics of each I/O standard

1.5.3.3 Memory Interfaces

Table 39. Memory Interfaces Checklist

Number	Done?	Checklist Item
1		Use the Arria 10 External Memory Interfaces IP core for each memory interface, and follow connection guidelines/restrictions in the appropriate documentation.
2		For a given bank, most memory pins are tied to a dedicated location. Refer to the *Arria 10 GX and SX Device Family Pin Connection Guidelines* for pin assignments.

Arria 10 devices provide an efficient architecture to quickly and easily fit wide external memory interfaces with their small modular I/O banks. The Arria 10 FPGA can support DDR external memory on any I/O banks on all sides of the device that do not support transceivers.

The self-calibrating Arria 10 External Memory Interfaces IP core is optimized to take advantage of the Arria 10 I/O structure. The Arria 10 External Memory Interfaces IP core 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 Intel memory controller Intel FPGA IP functions, the Arria 10 External Memory Interfaces IP core is instantiated automatically. If you design multiple memory interfaces into the device using Intel FPGA IP core, generate a unique interface for each instance to ensure good results instead of designing it once and instantiating it multiple times.

The data strobe DQS and data DQ pin locations are fixed in Arria 10 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.

You can implement a protocol that is not supported by Arria 10 External Memory Interfaces IP core by using the Altera PHYLite for Parallel Interfaces IP core.

Address and command pins within the address/command bank must follow a fixed pin-out scheme, as defined in the `<variation_name>_readme.txt` file generated with your IP core. The pin-out scheme varies according to the topology of the memory interface. The pin-out scheme is a hardware requirement that you must follow. Some schemes require three lanes to implement address and command pins, while others require four lanes.

Related Links

- Arria 10 Core Fabric and General Purpose I/Os Handbook
  For more information about supported memory protocols, refer to the "External Memory Interfaces in Arria 10 Devices" chapter of this handbook.
- Altera PHYLite for Parallel Interfaces IP Core User Guide
- External Memory Interface Handbook Volume 3: Reference Material
  For more information regarding the Arria 10 solution, refer to the "Functional Description—Arria 10 EMIF" chapter in this handbook.
- External Memory Interface Spec Estimator
  For supported Arria 10 EMIF specs.

1.5.3.4 Dual-Purpose and Special Pin Connections

Table 40. Dual-Purpose and Special Pin Connections Checklist

Number	Done?	Checklist Item
1		Make dual-purpose pin settings and check for any restrictions when using these pins as regular I/O.

Arria 10 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 programmable clock routing 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 the *Device-Wide Output Enable Pin* section.

For more information, refer to the *Register Power-Up Levels and Control Signals* section.

Related Links

- Device-Wide Output Enable Pin on page 26
- Register Power-Up Levels and Control Signals on page 43

1.5.3.5 Arria 10 I/O Features

The Arria 10 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 41. Arria 10 I/O Features

Feature	Usage	Guidelines and More Information
MultiVolt I/O Interface	Allows all packages to interface with systems of different supply voltages. VCCIO pins can be connected to a 1.2-, 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. VCCPD power pins must be connected to a 2.5- or 3.0-V power supply.	Refer to the previous sections and the I/O and High Speed I/O in Arria 10 Devices chapter of the Arria 10 Core Fabric and General Purpose I/O Handbook for a summary of MultiVolt I/O support and a list of the supported I/O standards and the typical values for input and output V_{CCIO}, V_{CCPD}, VREF, and board termination voltage (V_{TT}). Intel

continued...

Feature	Usage	Guidelines and More Information
		recommends that you use an external clamp diode on the all I/O pins when the input signal is 3.0 V.
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. Intel 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/O and High Speed I/O in Arria 10 Devices chapter of the Arria 10 Core Fabric and General Purpose I/O Handbook.
Programmable Slew Rate Control	Configure each pin for low-noise or high-speed performance. A faster slew rate provides high speed transitions. You can use faster slew rates to improve the available timing margin in memory-interface applications or when the output pin has 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. Intel 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 Arria 10 Device Datasheet.
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 kΩ . 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 Arria 10 Device Datasheet.
Programmable Pull-Up Resistor	Pull-up resistor (typically 25 kΩ) weakly holds the I/O to the V_{CCIO} level when in user mode. Can be used with 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.
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	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

continued...

Feature	Usage	Guidelines and More Information
	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.	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/O and High Speed I/O in Arria 10 Devices chapter of the Arria 10 Core Fabric and General Purpose I/O Handbook.
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/O and High Speed I/O in Arria 10 Devices chapter of the Arria 10 Core Fabric and General Purpose I/O Handbook.
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/O and High Speed I/O in Arria 10 Devices chapter of the Arria 10 Core Fabric and General Purpose I/O Handbook.
Dedicated Differential I/O SERDES Circuitry with DPA and Soft-CDR Support	All the I/Os in Arria 10 GX devices and E devices have built-in SERDES circuitry that supports high-speed LVDS interfaces at data rates of up to 1600 Mbps. 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 parameter editor. DPA usage adds some constraints on the placement of high-speed differential channels. Refer to the feature description and placement guidelines in the I/O and High Speed I/O in Arria 10 Devices chapter of the Arria 10 Core Fabric and General Purpose I/O Handbook.

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

Consider the following checklist items and refer to the appropriate documentation in Table 3 for detailed guidelines:

Table 42. Arria 10 I/O Features Checklist

Number	Done?	Checklist Item
1		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}.
2		Consider on-chip termination (OCT) features to save board space.
3		Verify that the required termination scheme is supported for all pin locations.
4		Choose the appropriate mode of DPA, non-DPA, or soft-CDR for high-speed LVDS interfaces.

Related Links

- Arria 10 Core Fabric and General Purpose I/Os Handbook
  For more information about I/Os, refer to the "I/O and High Speed I/O in Arria 10 Devices" chapter of this handbook.
- Arria 10 Device Datasheet
  For a list of the supply voltages required for Arria 10 devices and their recommended operation conditions.

1.5.4 Clock and PLL Selection

Table 43. Clock and PLL Selection Checklist

Number	Done?	Checklist Item
1		Use the correct dedicated clock pins and routing signals for clock and global control signals.
2		Use the device PLLs for clock management.
3		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.

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

Arria 10 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.

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.

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 Arria 10 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 Arria 10 device.

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 Prime 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.

Related Links

PLLs and Clock Networks
For more information about these features and detailed clock connection information.

1.5.5 PLL Feature Guidelines

Table 44. PLL Feature Guidelines Checklist

Number	Done?	Checklist Item
1		Enable PLL features and check settings in the parameter editor.

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 Prime parameter editor to enter your settings in Intel I/O PLL IP core, and check the results to verify whether particular features and input/output frequencies can be implemented in a particular PLL.

Arria 10 devices contain fractional PLLs in addition to I/O PLLs. You can configure fractional PLLs as integers or as enhanced fractional PLLs.

You can use I/O PLLs and 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 transmit clocking for transceivers.

Arria 10 device PLLs are feature rich, and support advanced capabilities such as clock feedback modes, switchover, and dynamic phase shifting.

Related Links

PLLs and Clock Networks
For more information about the fractional PLL features and a description of the clock network.

1.5.5.1 Clock Feedback Mode

The default clock feedback mode is direct compensation mode. Fractional PLLs support the following clock feedback modes:

- Direct compensation
- Feedback compensation bonding

I/O PLLs support the following clock feedback modes:

- Direct compensation
- Normal compensation
- Source synchronous compensation
- LVDS compensation
- ZDB compensation
- External feedback compensation

Table 45. Clock Feedback Mode Checklist

Number	Done?	Checklist Item
1		Ensure you select the correct PLL feedback compensation mode.

1.5.5.2 Clock Outputs

Table 46. Clock Outputs Checklist

Number	Done?	Checklist Item
1		Check that the PLL offers the required number of clock outputs and use dedicated clock output pins.

You can connect clock outputs to dedicated clock output pins or dedicated clock networks. There is no dedicated clock out pin for fractional PLL. I/O PLL can connect to a clock network or a dedicated clock pin.

1.5.6 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 Arria 10 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.

Table 47. Clock Control Features Checklist

Number	Done?	Checklist Item
1		Use the clock control block for clock selection and power-down.

Related Links

Clock Control Block (ALTCLKCTRL) IP Core User Guide

For information about using the ALTCLKCTRL megafunction to set up the clock control block refer to this User Guide.

1.5.7 I/O Simultaneous Switching Noise

Table 48. I/O Simultaneous Switching Noise Checklist

Number	Done?	Checklist Item
1		Analyze the design for possible SSN problems.
2		Reduce the number of pins that switch the voltage level at exactly the same time whenever possible.
3		Use differential I/O standards and lower-voltage standards for high-switching I/Os.
4		Use lower drive strengths for high-switching I/Os. The default drive strength setting might be higher than your design requires.
5		Reduce the number of simultaneously switching output pins within each bank. Spread output pins across multiple banks if possible.
6		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%).
7		Separate simultaneously switching pins from input pins that are susceptible to SSN.
8		Place important clock and asynchronous control signals near ground signals and away from large switching buses.
9		Avoid using I/O pins one or two pins away from PLL power supply pins for high-switching or high-drive strength pins.
10		Use staggered output delays to shift the output signals through time, or use adjustable slew rate settings.

SSN is a concern when too many I/Os (in close proximity) change voltage levels at the same time. Plan the I/O and clock connections according to the recommendations.

For more information, refer to "Arria 10 I/O Features" for details about the features you can use.

1.6 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.

1.6.1 Design Recommendations

Table 49. Design Recommendations Checklist

Number	Done?	Checklist Item
1		Use synchronous design practices. Pay attention to clock signals.

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.

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. 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.

For more information, refer to “PLL Board Design Guidelines”.

Arria 10 devices do not support the Quartus Prime Design Assistant design-rule checking tool.

Related Links

Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation
For more information about design recommendations, refer to the "Design Recommendations " chapter of this handbook. You can also refer to industry papers for more information about multiple clock design.

1.6.2 Using IP Cores

Table 50. Using IP Cores Checklist

Number	Done?	Checklist Item
1		Use IP cores with the parameter editor.

Intel provides parameterizable IP cores that are optimized for Intel device architectures. You can save design time by using IP cores instead of coding your own logic. Additionally, the Intel-provided IP cores can offer more efficient logic synthesis and device implementation. You can scale the IP core’s size and set various options with parameters. IP cores include the library of parameterized modules (LPM) and Intel device-specific IP cores. You can also take advantage of Intel and third-party IP cores and reference designs to save design time. The Quartus Prime IP catalog provides a user interface to customize IP cores. You should build or change IP core parameters using the parameter editor to ensure you set all ports and parameters correctly.

For more information, refer to “IP Selection”.

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

1.6.3 Reconfiguration

Arria 10 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.

1.6.3.1 Dynamic Reconfiguration

Arria 10 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.

Related Links

Reconfiguration Interface and Dynamic Reconfiguration
For more information about dynamic reconfiguration, refer to the "Reconfiguration Interface and Dynamic Reconfiguration" chapter of Arria 10 Transceiver PHY User Guide.

1.6.3.2 Partial Reconfiguration

Partial reconfiguration is an advanced feature of the device family. If you are interested in using partial reconfiguration, contact Intel for support.

Related Links

- Increasing Design Functionality with Partial and Dynamic Reconfiguration in 28-nm FPGAs
  For more information about partial reconfiguration
- Partial Reconfiguration IP Core User Guide
  For more information about partial reconfiguration with Arria 10 devices

1.6.4 Recommended HDL Coding Styles

Table 51. Recommended HDL Coding Styles Checklist

Number	Done?	Checklist Item
1		Follow recommended coding styles, especially for inferring device dedicated logic such as memory and DSP blocks.

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

Refer to your synthesis tool's documentation for any additional tool-specific guidelines. In the Quartus Prime software, you can use the HDL examples in the Language Templates available from the right-click menu in the text editor.

Related Links

Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation
For specific HDL coding examples and recommendations, refer to the "Recommended HDL Coding Styles" chapter of this handbook

1.6.5 Register Power-Up Levels and Control Signals

Table 52. Register Power-Up Levels and Control Signals Checklist

Number	Done?	Checklist Item
1		Enable the chip-wide reset to clear all registers if required.
2		Consider resources available for register power-up and control signals. Do not apply both reset and preset signals to a register.

Arria 10 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, the 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 Prime software on the **General** category of the **Device and Pin Options** dialog box before compiling your design.

Each Arria 10 logic array block (LAB) also contains dedicated logic for driving register control signals to its ALMs. 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.

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.

By default, the Quartus Prime 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, the Quartus Prime 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.

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.

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.

Related Links

- Arria 10 Core Fabric and General Purpose I/Os Handbook
  For more information about LAB and ALM architecture, refer to the "Logic Array Blocks and Adaptive Logic Modules in Arria 10 Devices" chapter of this handbook
- Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation
  For more information, refer to the **Power-Up Level** option and the `altera_attribute` assignment that set power-up conditions are described in the "Quartus Prime Integrated Synthesis" chapter of this handbook
- Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation
  For more information about reset logic and power up conditions, refer to the "Recommended HDL Coding Styles" chapter of this handbook

1.6.6 Planning for Hierarchical and Team-Based Design

The Quartus Prime incremental compilation feature preserves the results and performance for unchanged logic in your design as you make changes elsewhere, allowing you to perform more design iterations and achieve timing closure more efficiently. In an incremental compilation flow, the system architect splits a large design into smaller partitions that can be designed separately. In a team design environment, team members can work on partitions independently, which simplifies the design process and reduces compilation time. Partitioning your design is optional, but these benefits are important for large Arria 10 designs.

If you want to take advantage of the compilation-time savings and performance preservation of Quartus Prime 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.

Related Links

Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation
For more information about using the incremental compilation flows in the Quartus Prime software, refer to the "Quartus Prime Incremental Compilation for Hierarchical and Team-Based Design" chapter of this handbook

1.6.6.1 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 Intel'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.

Table 53. Planning Design Partitions Checklist

Number	Done?	Checklist Item
1		Follow recommendations to set up your source code and partition your design for incremental compilation; plan early in the design flow.

Related Links

Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation
For guidelines to help you create design partitions, refer to the "Best Practices for Incremental Compilation Partitions and Floorplan Assignments" chapter of this handbook

1.6.6.2 Planning in Bottom-Up and Team-Based Flows

If your design is created in multiple Quartus Prime 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.

Number	Done?	Checklist Item
1		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 Prime 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.

1.6.6.3 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.

Table 54. Creating a Design Floorplan Checklist

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

You can use the Quartus Prime 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.

Related Links

- Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation
  For guidelines to help you create a design floorplan, refer to the "Best Practices for Incremental Compilation Partitions and Floorplan Assignments" chapter of this handbook
- Quartus Prime Pro Edition Handbook Volume 2: Design Implementation and Optimization
  For more information about creating placement assignments in the floorplan, refer to the "Analyzing and Optimizing the Design Floorplan" chapter of this handbook

1.7 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 Prime Fitter then performs placement and routing to implement the design elements in specific device resources. If required, you can use the Quartus Prime 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.

1.7.1 Selecting a Synthesis Tool

Table 55. Selecting a Synthesis Tool Checklist

Number	Done?	Checklist Item
1		Specify your synthesis tool and use the correct supported version.

The Quartus Prime software includes advanced and easy-to-use integrated synthesis that fully supports Verilog HDL and VHDL, as well as the Intel 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 Prime 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.

Intel recommends using the most recent version of third-party synthesis tools, because tool vendors are continuously adding new features, fixing tool issues, and enhancing performance for Intel 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 Prime software to get accurate timing analysis and logic utilization results.

Your synthesis tool might offer the capability to create a Quartus Prime 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 Prime project for placement and routing.

Related Links

- Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation
  For more information about supported synthesis tools, refer to the appropriate chapter of this handbook
- Quartus Prime Pro Edition Software and Device Support Release Notes
  Provides more information on the version of each synthesis tool that is officially supported by that version of the Quartus Prime software.

1.7.2 Device Resource Utilization Reports

Table 56. Device Resource Utilization Reports Checklist

Number	Done?	Checklist Item
1		Review resource utilization reports after compilation.

After compilation in the Quartus Prime 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 Arria 10 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 the Quartus Prime 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.

1.7.3 Quartus Prime Messages

Table 57. Quartus Prime Messages Checklist

Number	Done?	Checklist Item
1		Review all Quartus Prime messages, especially warning or error 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 Prime 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.

Related Links

Quartus Prime Pro Edition Handbook Volume 2: Design Implementation and Optimization

For more information about messages and message suppression, refer to the "Managing Quartus Prime Projects" chapter of this handbook

1.7.4 Timing Constraints and Analysis

Table 58. Design Specifications Checklist

Number	Done?	Checklist Item
1		Ensure timing constraints are complete and accurate, including all clock signals and I/O delays.
2		Review the TimeQuest Timing Analyzer reports after compilation to ensure there are no timing violations.
3		Ensure that the input I/O times are not violated when data is provided to the Arria 10 device.

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 Prime 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 Prime software includes the Quartus Prime 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.

Related Links

- Quartus Prime Pro Edition Handbook Volume 3: Verification
  For more information about timing analysis, refer to the "The Quartus Prime TimeQuest Timing Analyzer" chapter of this handbook
- Quartus Prime Pro Edition Handbook Volume 3: Verification
  For more information about timing analysis, refer to the "Synopsys PrimeTime Support" chapter of this handbook

1.7.4.1 Recommended Timing Optimization and Analysis Assignments

Table 59. Recommended Timing Optimization and Analysis Assignments Checklist

Number	Done?	Checklist Item
1		Turn on **Optimize multi-corner timing** on the **Fitter Settings** page in the **Settings** dialog box.
2		Use `create_clock` and `create_generated_clock` to specify the frequencies and relationships for all clocks in your design.
3		Use `set_input_delay` and `set_output_delay` to specify the external device or board timing parameters.
4		Use `derive_pll_clocks` to create generated clocks for all PLL outputs, according to the settings in the PLL IP cores. Specify multicycle relationships for LVDS transmitters or receiver deserialization factors.
5		Use `derive_clock_uncertainty` to automatically apply inter-clock, intra-clock, and I/O interface uncertainties.
6		Use `check_timing` to generate a report on any problem with the design or applied constraints, including missing constraints.

The assignments and settings described in this section are important for large designs such as those in Arria 10 devices.

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, apply the recommended constraints to your design.

Related Links

Quartus Prime Pro Edition Handbook Volume 3: Verification
For more guidelines about timing constraints, refer to the "Best Practices for the Quartus Prime TimeQuest Timing Analyzer" chapter of this handbook

1.7.5 Area and Timing Optimization

Table 60. Area and Timing Optimization Checklist

Number	Done?	Checklist Item
1		Perform Early Timing Estimation if you want timing estimates before running a full compilation.
2		Use Quartus Prime optimization features to achieve timing closure or improve the resource utilization.
3		Use the Timing and Area Optimization Advisors to suggest optimization settings.

This section highlights some of the features offered in the Quartus Prime 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.

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.

Physical synthesis optimizations make placement-specific changes to the netlist that improve results for a specific Intel 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.

The Design Space Explorer (DSE) is a utility that automates the process of finding the optimal collection of the Quartus Prime 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.

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.

Related Links

Quartus Prime Pro Edition Handbook Volume 2: Design Implementation and Optimization

For information about additional optimization features, refer to the "Area and Timing Optimization" chapter of this handbook

1.7.6 Preserving Performance and Reducing Compilation Time

Table 61. Preserving Performance and Reducing Compilation Time Checklist

Number	Done?	Checklist Item
1		Use incremental compilation to preserve performance for unchanged blocks in your design and to reduce compilation times.
2		Ensure parallel compilation is enabled if you have multiple processors available for compilation.
3		Use the Compilation Time Advisor to suggest settings that reduce 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.

The Quartus Prime 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.

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.

Related Links

Quartus Prime Pro Edition Handbook Volume 2: Design Implementation and Optimization

For more suggestion on "Area and Timing Optimization"

1.7.7 Simulation

Table 62. Simulation Checklist

Number	Done?	Checklist Item
1		Specify your simulation tool, and use the correct supported version and simulation models.

The Quartus Prime software supports both RTL and gate level functional 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.

Intel provides the ModelSim* - Intel FPGA Starter Edition and offers the higher-performance ModelSim - Intel FPGA Edition, which enable you to take advantage of advanced testbench capabilities and other features. In addition, the Quartus Prime 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.

If you use a third-party simulation tool, use the software version that is supported with your Quartus Prime software version. The Quartus Prime Software Release Notes list the version of each simulation tool that is officially supported with that particular version of the Quartus Prime software. Use the model libraries provided with your Quartus Prime 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**.

Related Links

- Quartus Prime Software Release Notes.
  For more information about third party tool version support
- Quartus Prime Pro Edition Handbook Volume 3: Verification
  For more information about simulation tool flows, refer to the appropriate chapter of this handbook

1.7.8 Formal Verification

Table 63. Formal Verification Checklist

Number	Done?	Checklist Item
1		Specify your formal verification tool and use the correct supported version.
2		If you use formal verification, check for support and design limitations.

The Quartus Prime software supports some formal verification flows. Using a formal verification flow can impact performance results because it requires that certain logic optimizations be turned off, such as register retiming, and forces hierarchy blocks to be preserved, which can restrict optimization. There are other restrictions that can also limit your design; consult the documentation for details.

If formal verification is important to your design, it is easier to plan for limitations and restrictions in the beginning than to make changes later in the design flow.

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

Related Links

- Quartus Prime Pro Edition Handbook Volume 3: Verification
  For more information about formal verification flows, refer to the "Formal Verification" chapter of this handbook
- Quartus Prime Software Release Notes.
  For more information about third party tool version support

1.7.9 Power Analysis

Table 64. Power Analysis Checklist

Number	Done?	Checklist Item
1		After compilation, analyze power consumption and heat dissipation in the Power Analyzer.
2		Provide accurate signal activities, preferably with a gate-level simulation `.vcd`, to get accurate power analysis results.
3		Specify the correct operating conditions for power analysis.

Before design completion, estimate power consumption using the EPE spreadsheet. After compiling your design, analyze the power consumption and heat dissipation with the Quartus Prime Power Analyzer to ensure the design has not violated power supply and thermal budgets.

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 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.

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 **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.

Related Links

Quartus Prime Pro Edition Handbook Volume 3: Verification
For more information about power analysis and recommendations for simulation settings for creating signal activity information, refer to the "Power Analyzer" chapter of this handbook

1.7.10 Power Optimization

Arria 10 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 Prime 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 Arria 10 devices, you can use various design and software techniques to optimize your design.

Power optimization in the Quartus Prime 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.

1.7.10.1 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.

Arria 10 devices also offer the following power saving techniques:

- SmartVID
- V_{CC} Power Manager
- Programmable Power Technology
- Low Stratix Power Device Grades

Table 65. Device and Design Power Optimization Techniques Checklist

Number	Done?	Checklist Item
1		Use recommended design techniques and Quartus Prime options to optimize your design for power consumption, if required.
2		Use the Power Optimization Advisor to suggest optimization settings.

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.

Related Links

- Arria 10 Core Fabric and General Purpose I/Os Handbook
  For more details on Power Reduction Techniques
- Quartus Prime Pro Edition Handbook Volume 2: Design Implementation and Optimization
  For more details and additional design techniques to reduce power consumption, refer to the "Power Optimization" chapter in this handbook

1.7.10.1.1 Clock Power Management

Clocks represent a significant portion of dynamic power consumption, because of their high switching activity and long paths. The Quartus Prime 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 features 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.

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 Prime software automatically promotes register-level clock enable signals to the LAB level.

Related Links

Clock Control Block (ALTCLKCTRL) IP Core User Guide
For more information about using clock control blocks

1.7.10.1.2 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.

1.7.10.1.3 I/O Power Guidelines

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

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

Because dynamic power is also proportional to the output transition frequency, use resistively-terminated I/O standards such as SSTL for high-frequency applications. The output load voltage swings by an amount smaller than 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 EPE calculations, so be sure to include it separately in your system power calculations.

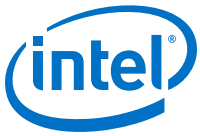
1.7.10.2 Quartus Prime Power Optimization Techniques

The Quartus Prime software offers power-optimized synthesis and fitting to reduce core dynamic power. Power-driven compilation works in conjunction with Programmable Power Technology in Arria 10 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.

Related Links

Quartus Prime Pro Edition Handbook Volume 2: Design Implementation and Optimization
For more information about power-driven compilation and the Power Optimization Advisor, refer to the "Power Optimization" chapter of this handbook



1.7.10.2.1 Power Optimization Advisor

The Quartus Prime 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 Analyzer to check the change in your power results.

1.8 Conclusion

The design guidelines in this application note provide important factors to consider in high-density, high-performance Arria 10 designs. It is important to follow Intel's recommendations throughout the design process to achieve good results, avoid common issues, and improve your design productivity. The "Design Checklist" 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 Arria 10 design.

1.9 Document Revision History

Table 66. Document Revision History

Date	Version	Changes
June 2017	2017.06.30	Made the following changes: • Changed "EPCQ" to "EPCQ-L" globally. • Changed the description of the CLKUSR optional configuration pin to match the GUI in the "Quartus Prime Configuration Settings" section.
March 2017	2017.03.20	Minor formatting changes.
March 2017	2017.03.15	Rebranded as Intel.
July 2016	2.3	Made the following changes: • Changed the I/O pin counts in the "I/O Pin Count, LVDS Channels, and Package Offering" section. • Changed the transceiver speed grade availabilities in the "Speed Grade" section. • Updated the URLs to PCIe* documentation in the "PCIe" section. • Updated the URLs for *Low Latency 40- and 100-Gbps Ethernet MAC and PHY MegaCore Function User Guide* and *DisplayPort IP Core User Guide* in the "Transceiver Design Flow" section. • Updated the supported modes in the "Clock Feedback Mode" section. • Changed the minimum data rate in the "PCS Types Supported by GX Transceiver Channels" table. • Updated the URLs in the "Calibration" section. • Changed the transceiver speeds in the "Device Variants and Applications" table.
June 2016	2.2	Made the following changes: • Changed "28.3 Gbps" to "25.8 Gbps" globally. • Added the "GT Transceiver Bank Architecture for Bank GXBL1G" figure. • Added the "GT Transceiver Bank Architecture for Banks GXBL1E and GXBL1H" figure. • Changed the description in the "Simulation" section.
		continued...

Date	Version	Changes
May 2016	2.1	Removed the step to select the Design Assistant option in the "Design Recommendations" section. The Design Assistant is not supported by Arria 10 devices.
May 2015	2.0	Added further description for the requirement of CLKUSR in the "Optional Configuration Pins" section. Changed "MegaWizard Plug-In Manager" to "IP Catalog" or "parameter editor" as appropriate, globally.
August 2014	1.0	Initial release.

1.10 Design Checklist

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

Number	Done?	N/A	Checklist Item
1.			"Create detailed design specifications and a test plan if appropriate."
2.			"Plan clock domains, clock resources, and I/O interfaces early with a block diagram."
3.			"Select IP that affects system design, especially I/O interfaces."
4.			"If you plan to use the OpenCore Plus tethered mode for IP, ensure that your board design supports this mode of operation."
5.			"Take advantage of Qsys for system and processor designs."
6.			"Select a device based on transceivers, I/O pin count, LVDS channels, package offering, logic/memory/multiplier density, PLLs, clock routing, and speed grade."
7.			"Reserve device resources for future development and debugging."
8.			"Consider vertical device migration availability and requirements."
9.			"Estimate power consumption with the Early Power Estimator (EPE) spreadsheet to plan the cooling solution and power supplies before the logic design is complete."
10.			"Set up the temperature sensing diode in your design to measure the device junction temperature for thermal management."
11.			"Select a configuration scheme to plan companion devices and board connections."
12.			"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."
13.			"If you want to use a flash device with the parallel flash loader, check the list of supported devices."
14.			"Ensure your configuration scheme and board support the following required features: data decompression, design security, remote upgrades, single event updates (SEU) mitigation."
15.			"Plan the board design to support optional configuration pins CLKUSR and INIT_DONE, as required."
16.			"Plan board design to use the **Auto-restart after configuration error** option."
17.			"Take advantage of on-chip debugging features to analyze internal signals and perform advanced debugging techniques. The ARM DS-5 <keyword keyref="companyname-tm"/> Edition offers you a variety of debugging features for SoC designs."
18.			"Select on-chip debugging scheme(s) early to plan memory and logic requirements, I/O pin connections, and board connections."

continued...

19.			"If you want to use Signal Probe 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."
20.			"Plan for the small amount of additional logic resources used to implement the JTAG hub logic for JTAG debugging features."
21.			"For debugging with the SignalTap II Embedded Logic Analyzer, reserve device memory resources to capture data during system operation."
22.			"Reserve I/O pins for debugging with Signal Probe or the Logic Analyzer Interface so you do not have to change the design or board to accommodate debugging signals later."
23.			"Ensure the board supports a debugging mode where debugging signals do not affect system operation."
24.			"Incorporate a pin header or mictor connector as required for an external logic analyzer or mixed signal oscilloscope."
25.			"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."
26.			"To use the Virtual JTAG megafunction for custom debugging applications, instantiate it in the HDL code as part of the design process."
27.			"To use the In-System Sources and Probes feature, instantiate the megafunction in the HDL code."
28.			"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 IP catalog."
29.			"Design board for power-up: Arria 10 output buffers are tri-stated until the device is configured and configuration pins drive out."
30.			"Design voltage power supply ramps to be monotonic."
31.			"Set POR time to ensure power supplies are stable."
32.			"Design power sequencing and voltage regulators for best device reliability. Connect the GND between boards before connecting the power supplies."
33.			"Connect all power pins correctly as specified in the Arria 10 GX, GT, and SX Device Family Pin Connection Guidelines."
34.			"Connect V_{CCIO} pins and VREF pins to support each bank's I/O standards."
35.			"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."
36.			"Follow the suggested power supply sharing and isolation guidance, and the specific guidelines for each pin in the Arria 10 GX, GT, and SX Device Family Pin Connection Guidelines."
37.			"Use the PDN tool to plan your power distribution netlist and decoupling capacitors."
38.			"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 Arria 10 GX, GT, and SX Device Family Pin Connection Guidelines."
39.			"Power supply nets should be provided by an isolated power plane, a power plane cut out, or thick trace of at least 20 mils."
40.			"Check that all configuration pin connections and pull-up/pull-down resistors are set correctly for your configuration scheme(s)."
41.			"Design configuration DCLK and TCK pins to be noise-free."

continued...

42.			"Connect JTAG pins to a stable voltage level if not in use."
43.			"Connect JTAG pins correctly to the download cable header. Ensure the pin order is not reversed."
44.			"To disable the JTAG state machine during power-up, pull the TCK pin low through a 1-kW resistor to ensure that an unexpected rising edge does not occur on TCK."
45.			"Pull TMS and TDI high through a 1-k to 10-kW resistor."
46.			"Connect TRST directly to V_{CCPGM} (Connecting the pin low disables the JTAG circuitry)."
47.			"Because the download cable interfaces with the JTAG pins of your device, ensure the download cable and JTAG pin voltages are compatible."
48.			"Buffer JTAG signals per the recommendations, especially for connectors or if the cable drives more than three devices."
49.			"If your device is in a configuration chain, ensure all devices in the chain are connected properly."
50.			"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."
51.			"Connect nIO_PULLUP directly to GND."
52.			"Hold the nCE (chip enable) pin low during configuration, initialization, and user mode."
53.			"Turn on the device-wide output enable option, if required."
54.			"Specify the reserved state for unused I/O pins."
55.			"Carefully check the pin connections in the Quartus Prime software-generated **.pin**. Do not connect RESERVED pins."
56.			"Design VREF pins to be noise-free."
57.			"Break out large bus signals on board layers close to the device to reduce cross talk."
58.			"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."
59.			"Check I/O termination and impedance matching for chosen I/O standards, especially for voltage-referenced standards."
60.			"Perform board-level simulation using IBIS models (when available)."
61.			"Configure board trace models for Quartus Prime advanced I/O timing analysis."
62.			"Use the Quartus Prime Pin Planner to make pin assignments."
63.			"Use Quartus Prime Fitter messages and reports for sign-off of pin assignments."
64.			"Verify that the Quartus Prime pin assignments match those in the schematic and board layout tools."
65.			"Use the **Create Top-Level Design File** command with I/O Assignment Analysis to check the I/O assignments before the design is complete."
66.			"Plan the I/O signaling type based on the system requirements."
67.			"Allow the software to assign locations for the negative pin in differential pin pairs."
68.			"Select a suitable signaling type and I/O standard for each I/O pin. The I/O banks are located in I/O columns. Each I/O bank contains its own PLL, DPA, and SERDES circuitries."

continued...

69.			"Ensure that the appropriate I/O standard support is supported in the targeted I/O bank."
70.			"Place I/O pins that share voltage levels in the same I/O bank."
71.			"Verify that all output signals in each I/O bank are intended to drive out at the bank's V_{CCIO} voltage level."
72.			"Verify that all voltage-referenced signals in each I/O bank are intended to use the bank's VREF voltage level."
73.			"Check the I/O bank support for LVDS and transceiver features."
74.			"Use the Arria 10 EMIF IP megafunction (or IP core) for each memory interface, and follow connection guidelines/restrictions in the appropriate documentation."
75.			"Use dedicated DQ/DQS pins and DQ groups for memory interfaces."
76.			"Make dual-purpose pin settings and check for any restrictions when using these pins as regular I/O."
77.			"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}."
78.			"Consider OCT features to save board space."
79.			"Verify that the required termination scheme is supported for all pin locations."
80.			"Choose the appropriate mode of DPA, non-DPA, or soft-CDR for high-speed LVDS interfaces."
81.			"Use the correct dedicated clock pins and routing signals for clock and global control signals."
82.			"Use the device fractional PLLs for clock management."
83.			"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."
84.			"Enable PLL features and check settings in the parameter editor."
85.			"Ensure you select the correct PLL feedback compensation mode."
86.			"Check that the PLL offers the required number of clock outputs and use dedicated clock output pins."
87.			"Use the clock control block for clock selection and power-down."
88.			"Analyze the design for possible SSN problems."
89.			"Reduce the number of pins that switch the voltage level at exactly the same time whenever possible."
90.			"Use differential I/O standards and lower-voltage standards for high-switching I/Os."
91.			"Use lower drive strengths for high-switching I/Os. The default drive strength setting might be higher than your design requires."
92.			"Reduce the number of simultaneously switching output pins within each bank. Spread output pins across multiple banks if possible."
93.			"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%)."
94.			"Separate simultaneously switching pins from input pins that are susceptible to SSN."
95.			"Place important clock and asynchronous control signals near ground signals and away from large switching buses."

continued...

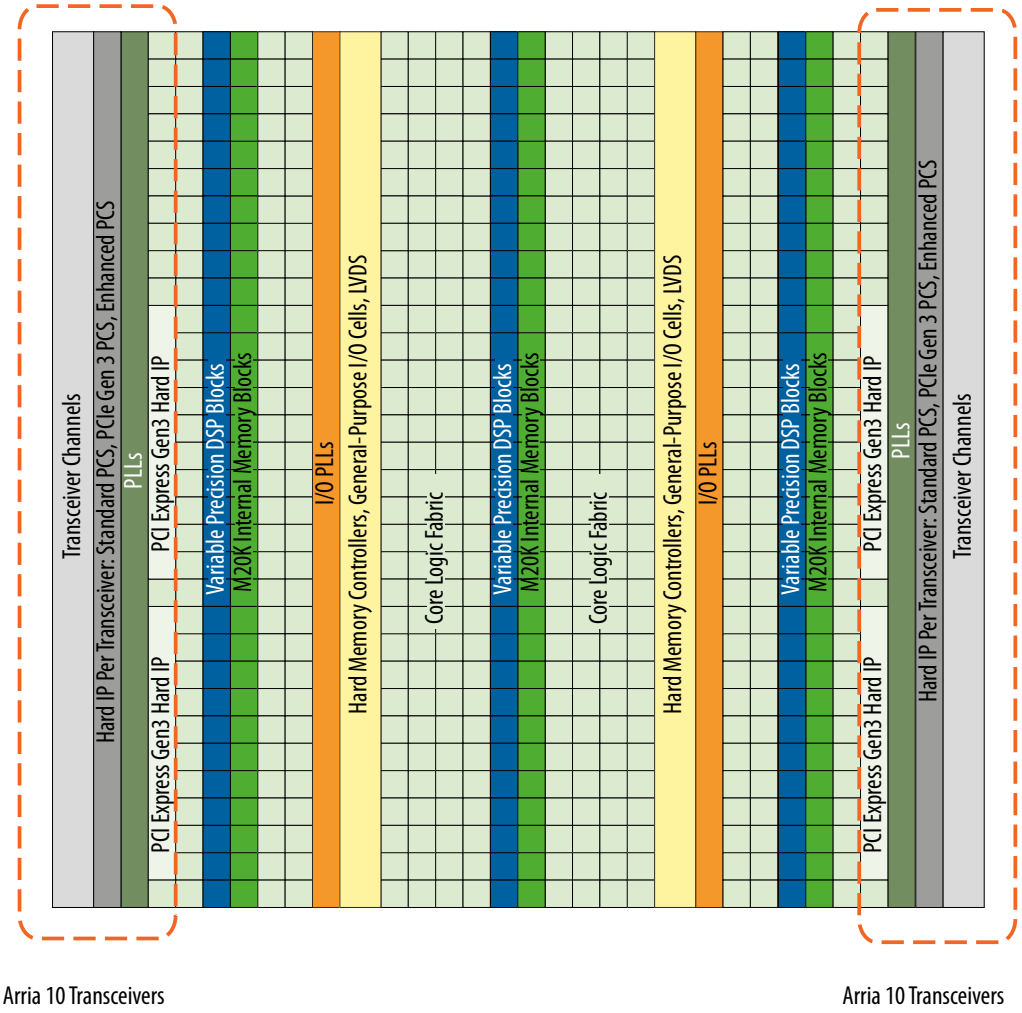
96.			"Avoid using I/O pins one or two pins away from PLL power supply pins for high-switching or high-drive strength pins."
97.			"Use staggered output delays to shift the output signals through time, or use adjustable slew rate settings."
98.			"Use synchronous design practices. Pay attention to clock signals."
99.			"Use the Quartus II Design Assistant to check design reliability."
99.			"Use megafunctions with the parameter editor."
100.			"Follow recommended coding styles, especially for inferring device dedicated logic such as memory and DSP blocks."
101.			"Enable the chip-wide reset to clear all registers if required."
102.			"Consider resources available for register power-up and control signals. Do not apply both reset and preset signals to a register."
103.			"Follow recommendations to set up your source code and partition your design for incremental compilation; plan early in the design flow."
104.			"Perform timing budgeting and resource balancing between partitions to achieve best results, especially in team-based flows."
105.			"Create a design floorplan for incremental compilation partitions, if required for your design flow."
106.			"Specify your third-party synthesis tool and use the correct supported version."
107.			"Review resource utilization reports after compilation."
108.			"Review all Quartus Prime messages, especially warning or error messages."
109.			"Ensure timing constraints are complete and accurate, including all clock signals and I/O delays."
110.			"Review the TimeQuest Timing Analyzer reports after compilation to ensure there are no timing violations."
111.			"Ensure that the input I/O times are not violated when data is provided to the Arria 10 device."
112.			"Turn on **Optimize multi-corner timing** on the **Fitter Settings** page in the **Settings** dialog box."
113.			"Use `create_clock` and create_generated_clock to specify the frequencies and relationships for all clocks in your design."
114.			"Use `set_input_delay` and set_output_delay to specify the external device or board timing parameters."
115.			"Use `derive_pll_clocks` to create generated clocks for all PLL outputs, according to the settings in the PLL megafunctions. Specify multicycle relationships for LVDS transmitters or receiver deserialization factors."
116.			"Use `derive_clock_uncertainty` to automatically apply inter-clock, intra-clock, and I/O interface uncertainties."
117.			"Use `check_timing` to generate a report on any problem with the design or applied constraints, including missing constraints."
118.			"Perform Early Timing Estimation if you want timing estimates before running a full compilation."
119.			"Use Quartus Prime optimization features to achieve timing closure or improve the resource utilization."
120.			"Use the Timing and Area Optimization Advisors to suggest optimization settings."

continued...

121.			"Use incremental compilation to preserve performance for unchanged blocks in your design and to reduce compilation times."
122.			"Ensure parallel compilation is enabled if you have multiple processors available for compilation."
123.			"Use the Compilation Time Advisor to suggest settings that reduce compilation time."
124.			"Specify your third-party simulation tool, and use the correct supported version and simulation models."
125.			"Specify your third-party formal verification tool and use the correct supported version."
126.			"If you use formal verification, check for support and design limitations."
127.			"After compilation, analyze power consumption and heat dissipation in the Power Analyzer."
128.			"Provide accurate typical signal activities, preferably with a gate-level simulation **.vcd**, to get accurate power analysis results."
129.			"Specify the correct operating conditions for power analysis."
130.			"Use recommended design techniques and Quartus Prime options to optimize your design for power consumption, if required."
131.			"Use the Power Optimization Advisor to suggest optimization settings."

1.11 Appendix: Arria 10 Transceiver Design Guidelines

Figure 2. Arria 10 FPGA Architecture Block Diagram



Note: The transceiver channels are placed on the left side periphery in most Arria 10 devices. For larger Arria 10 devices, additional transceiver channels are placed on the right side periphery.

1.11.1 Transceiver PHY Architecture Overview

A link is defined as a single entity communication port. A link can have one or more transceiver channels. A transceiver channel is synonymous with a transceiver lane.

For example, a 10GBASE-R link has one transceiver channel or lane with a data rate of 10.3125 Gbps. A 40GBASE-R link has four transceiver channels. Each transceiver channel operates at a lane data rate of 10.3125 Gbps. Four transceiver channels give a total collective link bandwidth of 41.25 Gbps (40 Gbps before and after 64B/66B PCS encoding and decoding).

1.11.2 Transceiver Bank Architecture

The transceiver bank is the fundamental unit that contains all the functional blocks related to the device's high speed serial transceivers.

Each transceiver bank includes six transceiver channels in all devices except for the devices with 66 transceiver channels. These devices (with 66 transceiver channels) have both six channel and three channel transceiver banks. The uppermost transceiver bank on the left and the right side of these devices is a three channel transceiver bank. All other devices contain six channel transceiver banks.

The figures below show the transceiver bank architecture with the phase locked loop (PLL) and clock generation block (CGB) resources available in each bank.

Figure 3. Three-Channel GX Transceiver Bank Architecture

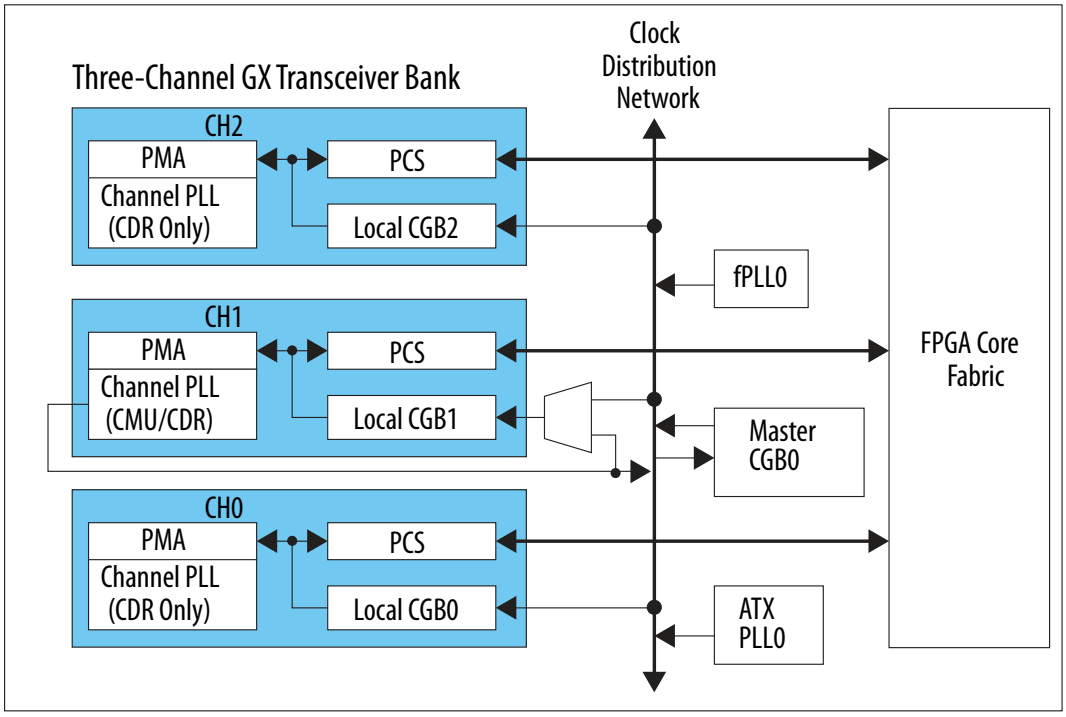


Figure 4. Six-Channel GX Transceiver Bank Architecture

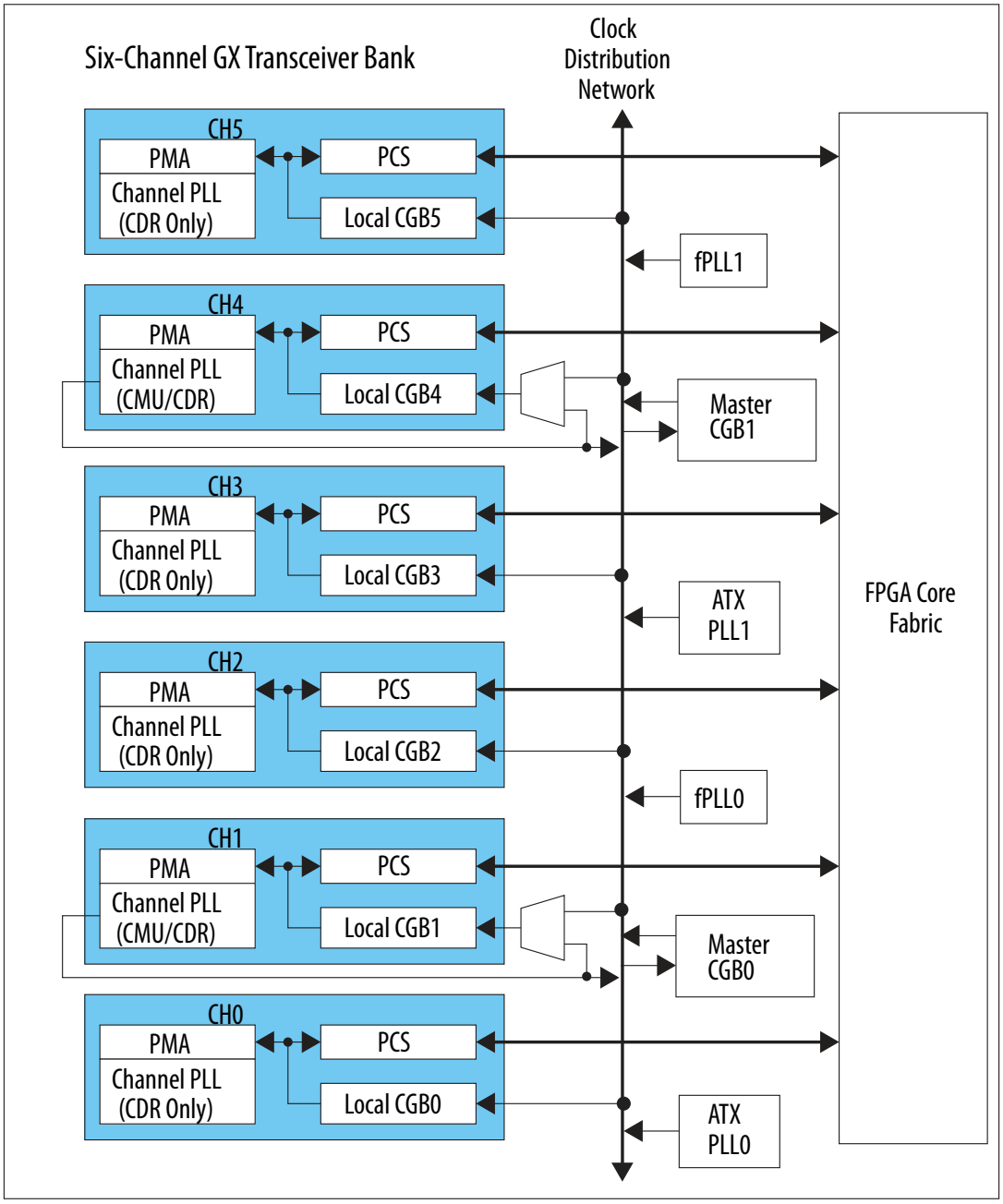
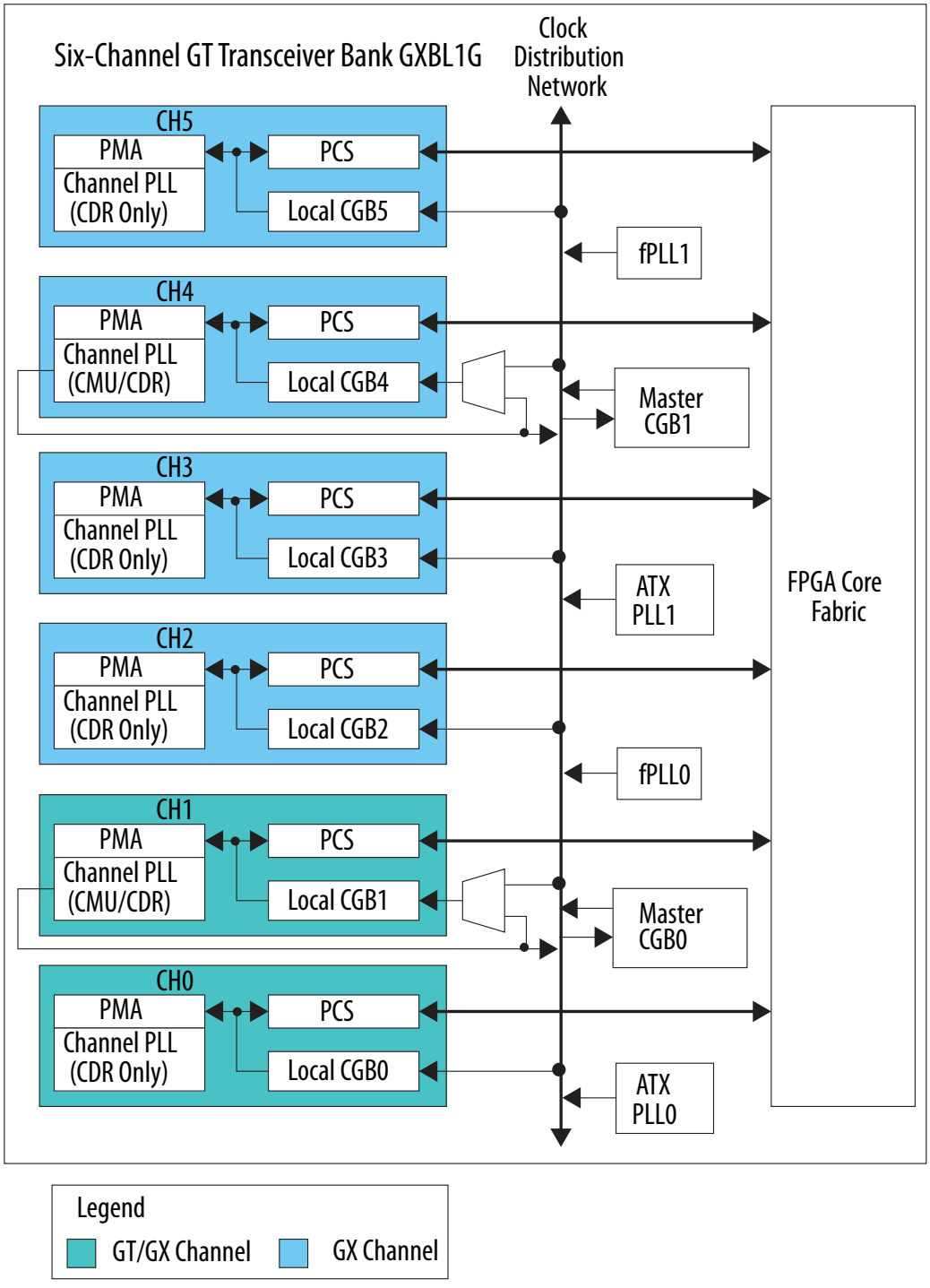
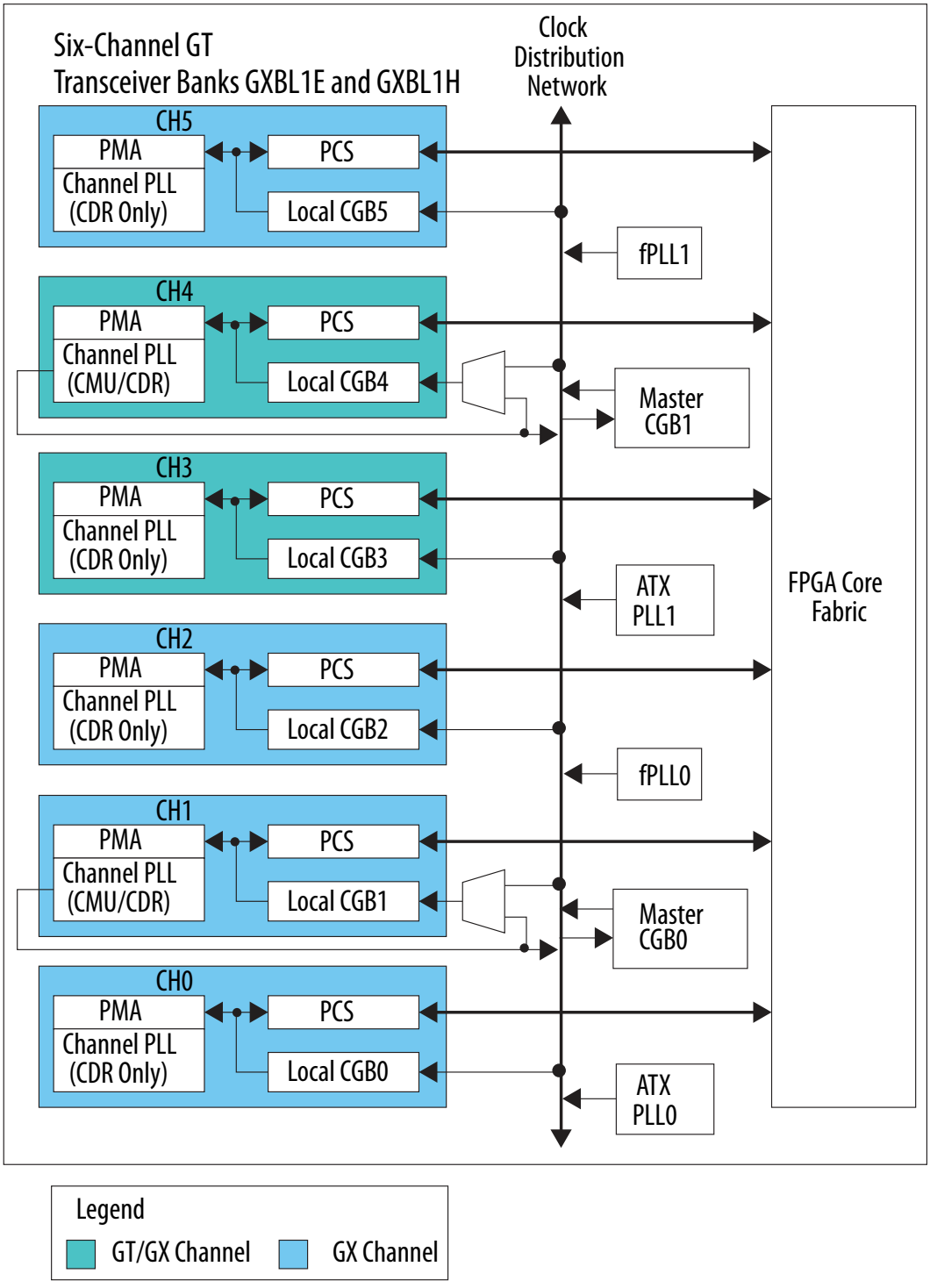


Figure 5. GT Transceiver Bank Architecture for Bank GXBL1G



Note: In GT devices, the transceiver banks GXBL1E, GXBL1G, and GXBL1H include GT channels.

Figure 6. GT Transceiver Bank Architecture for Banks GXBL1E and GXBL1H



The transceiver channels perform all the required PHY layer functions between the FPGA fabric and the physical medium. The high speed clock required by the transceiver channels is generated by the transceiver PLLs. The master and local clock generation blocks (CGBs) provide the necessary high speed serial and low speed parallel clocks to drive the non-bonded and bonded channels in the transceiver bank.

Related Links

Transceiver Basics
For an online training course for transceivers

1.11.3 PHY Layer Transceiver Components

Transceivers in Arria 10 devices support both Physical Medium Attachment (PMA) and Physical Coding Sublayer (PCS) functions at the physical (PHY) layer.

A PMA is the transceiver's electrical interface to the physical medium. The transceiver PMA consists of standard blocks such as:

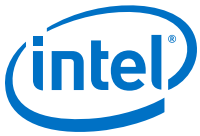
- serializer/deserializer (SERDES)
- clock and data recovery PLL
- analog front end transmit drivers
- analog front end receive buffers

The PCS can be bypassed with a PCS-Direct configuration. Both the PMA and PCS blocks are fed by multiple clock networks driven by high performance PLLs. In PCS-Direct configuration, the data flow is through the PCS block, but all the internal PCS blocks are bypassed. In this mode, the PCS functionality is implemented in the FPGA fabric.

1.11.3.1 The GX Transceiver Channel

Related Links

- PLLs
  For more information about transceiver PLLs in Arria 10 devices
- ATX PLL
  For more information about ATX PLL
- ATX PLL IP
  For details about implementing the ATX PLL IP
- fPLL
  For more information about fPLL
- fPLL IP
  For details about implementing the fPLL IP
- CMU PLL
  For more information about CMU PLL
- CMU PLL IP
  For details about implementing the CMU PLL IP
- Clock Generation Block
  For more information about the clock generation block
- design examples



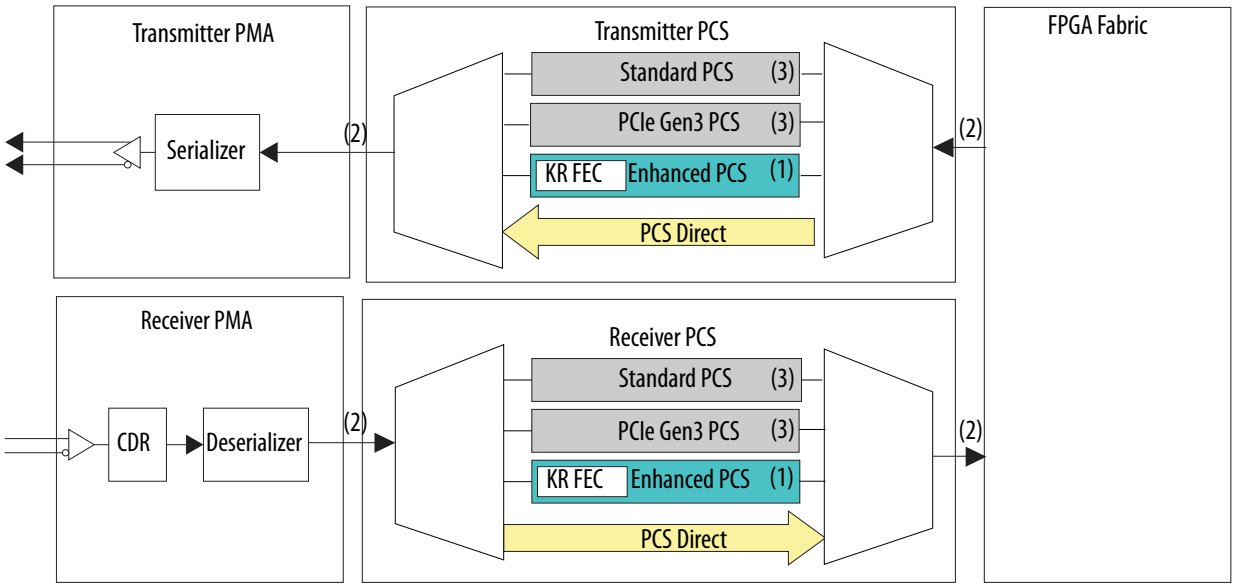
For more information about design examples

- CPRI MegaCore Function User Guide
  For more information about the Arria 10 protocols
- DisplayPort IP Core Function User Guide
  For more information about the Arria 10 protocols
- Low Latency 40- and 100-Gbps Ethernet MAC and PHY MegaCore Function User Guide
  For more information about the Arria 10 protocols
- 100G Interlaken MegaCore Function User Guide
  For more information about the Arria 10 and Interlaken
- 50G Interlaken MegaCore Function User Guide
  For more information about the Arria 10 and Interlaken
- Arria 10 Avalon-MM DMA Interface for PCIe* Solutions User Guide
  For more information about the Arria 10 and PCIe*
- Arria 10 Avalon-MM Interface for PCIe Solutions User Guide
  For more information
- Arria 10 Avalon-ST Interface for PCIe Solutions User Guide
  For more information
- SerialLite III Streaming MegaCore Function User Guide
  For more information
- RapidIO II MegaCore Function User Guide
  For more information

1.11.3.2 The GT Transceiver Channel

The GT transceiver channels are used for supporting data rates from 17.4 Gbps to 25.8 Gbps. The GT transceiver channels can also be reconfigured as GX transceiver channels. When they are reconfigured as GX transceiver channels, the Standard PCS, Enhanced PCS, and PCIe Gen3 PCS are available and they support data rates from 1 Gbps to 17.4 Gbps.

Figure 7. GT Transceiver Channel in Full Duplex Mode Operating Between 17.4 Gbps and 25.8 Gbps



Notes:
(1) The Enhanced PCS must be configured in lowl latency mode to support data rate range from 17.4 Gbps to 25.8 Gbps.
(2) The FPGA Fabric - PCS and PCS-PMA interface widths are configurable.
(3) The Standard PCS and PCIe Gen3 PCS blocks are available when the GT channel is reconfigured as a GX transceiver channel.

Table 67. PCS Types and Data Rates Supported by GT Channel Configurations

GT Channel Configuration	PCS Type	Data Rates Supported
GT	Standard PCS	Not available
	Enhanced PCS	17.4 Gbps to 25.8 Gbps (1)
	PCIe Gen3 PCS	Not available
GX	Standard PCS	1 Gbps to 10 Gbps
	Enhanced PCS	1 Gbps to 17.4 Gbps
	PCIe Gen3 PCS	8 Gbps
Notes: 1. The Enhanced PCS must be configured in low latency mode to support data rate range from 17.4 Gbps to 25.8 Gbps. 2. The GT channels can also operate in PCS-Direct configuration for data rates between 1 Gbps to 25.8 Gbps.		

1.11.4 Transceiver Phase-Locked Loops

Each transceiver channel in Arria 10 devices has direct access to three types of high performance PLLs:

- Advanced Transmit (ATX) PLL
- Fractional PLL (fPLL)
- Channel PLL / Clock Multiplier Unit (CMU) PLL.

These transceiver PLLs along with the Master or Local Clock Generation Blocks (CGB) drive the transceiver channels.

Related Links

Arria 10 Transceiver PHY User Guide
For more information about transceiver PLLs in Arria 10 devices, refer to the "PLLs" section of the "PLLs and Clock Networks" chapter of this user guide

1.11.4.1 Advanced Transmit (ATX) PLL

An advanced transmit (ATX) PLL is a high performance PLL. It supports both integer frequency synthesis and coarse resolution fractional frequency synthesis. The ATX PLL is the transceiver channel's primary transmit PLL. It can operate over the full range of supported data rates required for high data rate applications.

Related Links

- Arria 10 Transceiver PHY User Guide
  For more information about ATX PLL, refer to the "ATX PLL" section of the "PLLs and Clock Networks" chapter of this user guide
- Arria 10 Transceiver PHY User Guide
  For details about implementing the ATX PLL IP, refer to the "ATX PLL IP" section of the "PLLs and Clock Networks" chapter of this user guide

1.11.4.2 Fractional PLL (fPLL)

A fractional PLL (fPLL) is an alternate transmit PLL used for generating low clock frequencies for low data rate applications. fPLLs support both integer frequency synthesis and fine resolution fractional frequency synthesis. Unlike the ATX PLL, the fPLL can be used to synthesize frequencies that can drive the core through the FPGA fabric clock networks.

Related Links

- Arria 10 Transceiver PHY User Guide
  For more information about fPLL, refer to the "fPLL" section of the "PLLs and Clock Networks" chapter of this user guide
- Arria 10 Transceiver PHY User Guide
  For details about implementing the fPLL IP, refer to the "fPLL IP" section of the "PLLs and Clock Networks" chapter of this user guide

1.11.4.3 Channel PLL (CMU/CDR PLL)

A channel PLL resides locally within each transceiver channel. Its primary function is clock and data recovery in the transceiver channel when the PLL is used in CDR mode. The channel PLLs of channel 1 and 4 can be used as a transmit PLL when reconfigured in CMU mode. The channel PLLs of channel 0, 2, 3, and 5 cannot be reconfigured in CMU mode and therefore cannot be used as a transmit PLL.

Related Links

- Arria 10 Transceiver PHY User Guide
  For more information about CMU PLL, refer to the "CMU PLL" section of the "PLLs and Clock Networks" chapter in this user guide

- Arria 10 Transceiver PHY User Guide
  For details about implementing the CMU PLL IP, refer to the "CMU PLL IP" section of the "PLLs and Clock Networks" chapter of this user guide

1.11.5 Clock Generation Block (CGB)

In Arria 10 devices, there are two types of clock generation blocks (CGBs):

- Master CGB
- Local CGB

Transceiver banks with six transceiver channels have two master CGBs. Master CGB1 is located at the top of the transceiver bank and master CGB0 is located at the bottom of the transceiver bank. Transceiver banks with three channels have only one master CGB. The master CGB divides and distributes bonded clocks to a bonded channel group. It also distributes non-bonded clocks to non-bonded channels across the x6/xN clock network.

Each transceiver channel has a local CGB. The local CGB is used for dividing and distributing non-bonded clocks to its own PCS and PMA blocks.

Related Links

Arria 10 Transceiver PHY User Guide
For more information about the clock generation block, refer to the "Clock Generation Block" section of the "PLLs and Clock Networks" chapter in this user guide

1.11.6 Calibration

Arria 10 FPGA devices contain a dedicated calibration engine to compensate for process variations.

The calibration engine calibrates the analog portion of the transceiver to allow both the transmitter and receiver to operate at maximum performance. Each Arria 10 device contains two calibration engines and each engine resides on either side of the device. A hard NIOS II processor controls the calibration flow.

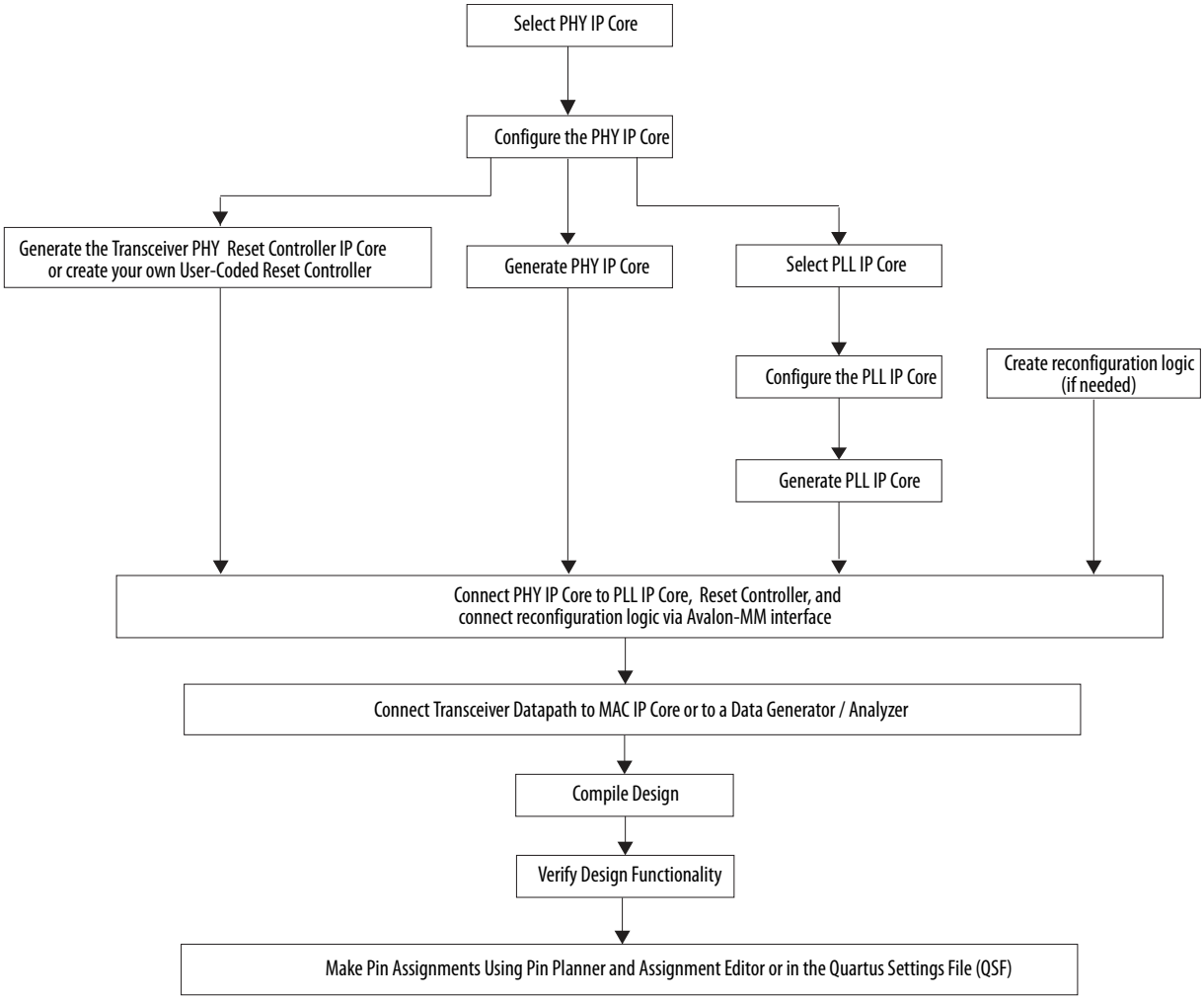
The CLKUSR pin clocks the calibration engine. All transceiver reference clocks and the CLKUSR clock must be free running and stable upon device power-up to successfully complete the calibration process and for optimal transceiver performance.

Related Links

- Arria 10 Device Datasheet
  For more information about CLKUSR pin requirements
- Configuration, Design Security, and Remote System Upgrades in Arria 10 Devices
  For information about configuration requirements for the CLKUSR pin

1.11.7 Transceiver Design Flow

Figure 8. Transceiver Design Flow



Related Links

- Arria 10 Transceiver PHY Design Examples
  For more information about design examples refer to the Intel FPGA Wiki page that provides useful guidance for developing your own design. However, the wiki is not guaranteed by Intel.
- CPRI MegaCore Function User Guide
- DisplayPort IP Core Function User Guide
- Low Latency 40- and 100-Gbps Ethernet MAC and PHY MegaCore Function User Guide
- 100G Interlaken MegaCore Function User Guide
- 50G Interlaken MegaCore Function User Guide
- Arria 10 Avalon-MM DMA Interface for PCIe* Solutions User Guide

- Arria 10 Avalon-MM Interface for PCIe Solutions User Guide
- Arria 10 Avalon-ST Interface for PCIe Solutions User Guide
- SerialLite III Streaming MegaCore Function User Guide
- RapidIO II MegaCore Function User Guide